
Advanced Prediction Models

Deep Learning, Graphical Models and Reinforcement
Learning

Today's Outline

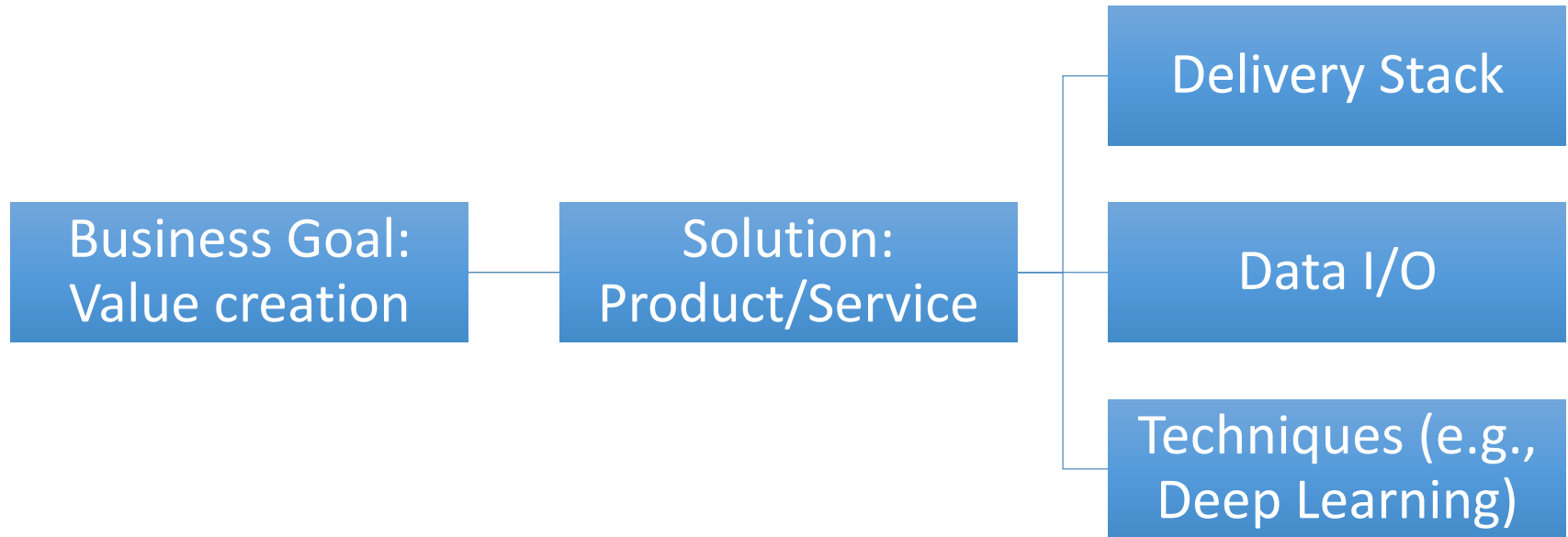
- Course Logistics
- Introduction to the Course
- Getting Started with Neural Nets
 - Classification
 - Backpropagation
 - Feedforward Neural Nets

Course Topics

- We will cover several tools under the umbrella of
 - Deep Learning
 - Probabilistic Graphical Models
 - Online and Reinforcement Learning

Introduction to the Course

20000 Ft View



- You need a critical understanding of the domain to be successful in shipping solutions
- Before venturing into a complex technique, try a shallow/easy technique

A Business Analyst's Toolkit

- Techniques
 - Prediction
 - Decision Trees
 - Linear classifiers and logistic regression
 - Naïve Bayes classifier
 - SVMs
 - Neural networks (and deep learning)
 - Graphical models
 - Online/reinforcement learning
 - Exploration
 - Clustering
 - Market basket analysis

Example I

- You are an online fashion retailer
- Want to adaptively recommend products
- Cannot measure certain quantities directly
 - Substitution behavior
 - Stock-level sensitivities

Example I

- You are an online fashion retailer
- Want to adaptively recommend products
- Cannot measure certain quantities directly
 - Substitution behavior
 - Stock-level sensitivities
- Build a personalization system that infers the most likely product that would be bought given censored/partial information
 - Recommend products
 - Tweak prices

Example II

- You are a home insurance provider
- Want to check houses for risks and opportunities
- Manually checking houses and neighborhood does not scale

Example II

- You are a home insurance provider
- Want to check houses for risks and opportunities
- Manually checking houses and neighborhood does not scale
- Fly a helicopter/drone and capture video
- Tag objects in the video
 - Classify if a outdoor pool is present or not
 - Classify greenery
 - Segment the house from the background
- Figure out insurance premiums across neighborhoods

Example III

- Fashion retailing
 - The customer dislikes our recommendation
 - The customer finds the price too high
 - How to update our recommendations and prices?
- Home insurance
 - Prices the premium too low for this year
 - Had to payout a lot
 - How to update the premium for next year?

Example III

- Fashion retailing
 - The customer dislikes our recommendation
 - The customer finds the price too high
 - How to update our recommendations and prices?

Data Variety

- Structured data
 - Examples:
 - Medical/healthcare data
 - Advertising data
 - Have ordinal, integer, binary or categorical fields
 - Among other tools, one can use graphical models

Data Variety

- Structured data
 - Examples:
 - Medical/healthcare data, advertising data
 - Have ordinal, integer, binary or categorical fields
 - If there is missing/noisy data, one can use **graphical models**
- Unstructured data
 - Examples:
 - Images (tensor, i.e., typically a 3 dimensional array) and videos (a sequence of images), text strings/documents
 - **Deep learning** reduces feature engineering effort

Complex Decisions

- Decisions
 - Examples:
 - which articles to show, how to price products
 - May use many predictions, may need to be taken repeatedly for different contexts
 - Online and reinforcement learning methods address this 'learning on the go' problem

Two Themes of the Course

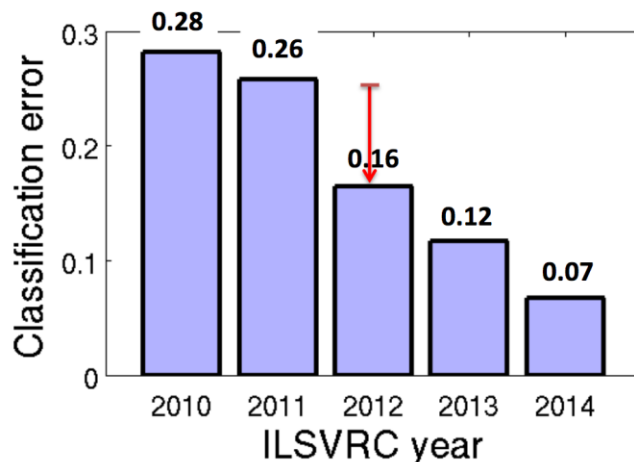
- Data Variety
 - Images and Videos
 - Speech
 - Text and Language
- Complex Decisions
 - Sequential Decision Making

Three Techniques covered in the Course

- To address data variety and complex decision problems, we will look at:
 - Deep Learning
 - Probabilistic Graphical Models
 - Online and Reinforcement Learning

Deep Learning

- One example (in vision) of its success is at the ILSVRC¹
- ImageNet dataset has 22000 categories across 14 million images
- ILSVRC Task 1 was a classification challenge
 - Given 1000 categories and 1.5 million images, predict 5 categories for a test image



¹ImageNet Large Scale Visual Recognition Challenge

²Figure: Russakovsky et al. arxiv:1409.0575

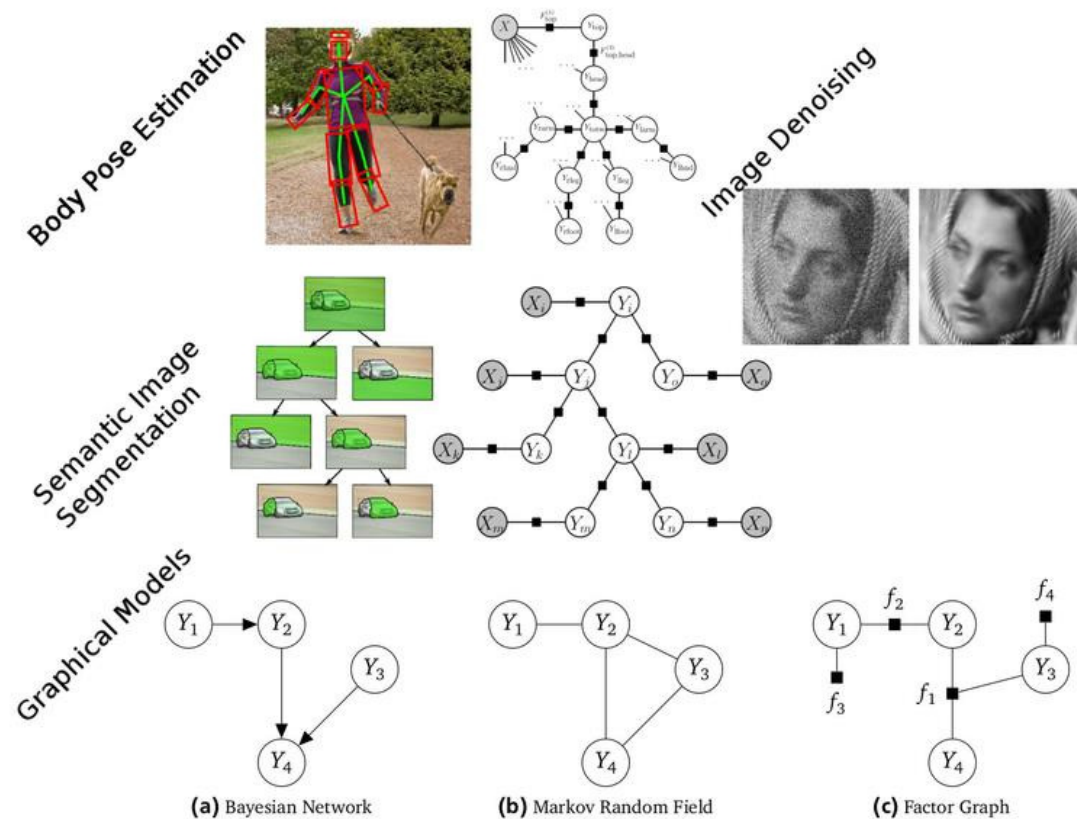
Deep Learning

- Neural nets are not new (1960s). Applied to handwritten digit recognition back in 1998
- Were not mainstream till around 2010/2012*
 - What changed? [Access to GPUs and Data](#)
- Caveat:
 - Deep learning achieves good performance on some tasks
 - Typically has not worked well beyond classification...
 - There is a lot of scope for improvement, engineering, system building, model building

*Context-Dependent Pre-trained Deep Neural Networks for Large Vocabulary Speech Recognition, Dahl et al. 2010
Imagenet classification with deep convolutional neural networks, Krizhevsky et al. 2012

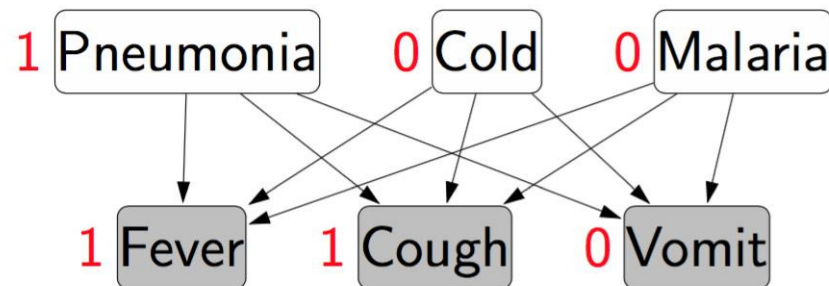
Graphical Models

- Probability distribution and graphs!
- Method of choice for complex machine learning models



Graphical Models

Question: If patient has a cough and fever, what disease(s) does he/she have?



Probabilistic program: diseases and symptoms

For each disease $i = 1, \dots, m$:

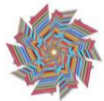
Generate activity of disease $D_i \sim p(D_i)$

For each symptom $j = 1, \dots, n$:

Generate activity of symptom $S_j \sim p(S_j \mid D_{1:m})$

Graphical Models

Question: given a text document, what topics is it about?



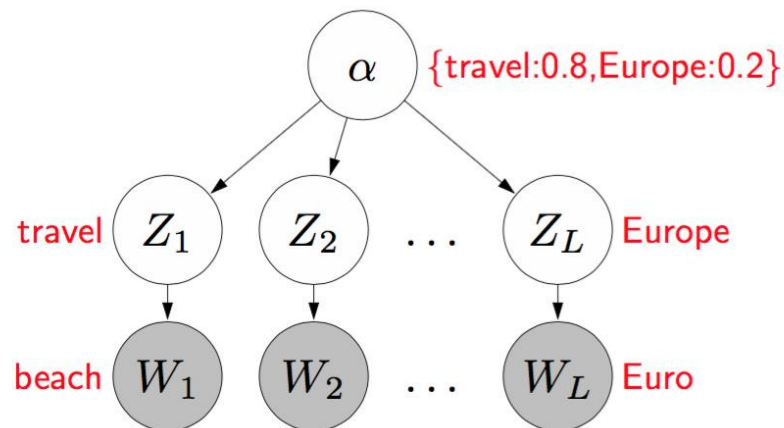
Probabilistic program: latent Dirichlet allocation

Generate a distribution over topics $\alpha \in \mathbb{R}^K$

For each position $i = 1, \dots, L$:

Generate a topic $Z_i \sim p(Z_i \mid \alpha)$

Generate a word $W_i \sim p(W_i \mid Z_i)$



Graphical Models vs Deep Learning

Graphical Models

- Probabilistic
- Dependencies btw. RVs
- Low capacity
- Domain knowledge: easy to encode

Deep Neural Networks

- Deterministic
- Input/Output Mapping
- High capacity
- Domain knowledge: hard

Combinations:

D. Kingma and M. Welling: Auto-encoding variational Bayes. ICLR, 2014.

L. Chen, A. Schwing and R. Urtasun: Learning Deep Structured Models. ICML, 2015.

J. Domke: Learning graphical model parameters with approximate marginal inference. PAMI, 2013, Vol. 35, no. 10, pp. 2454–2467.

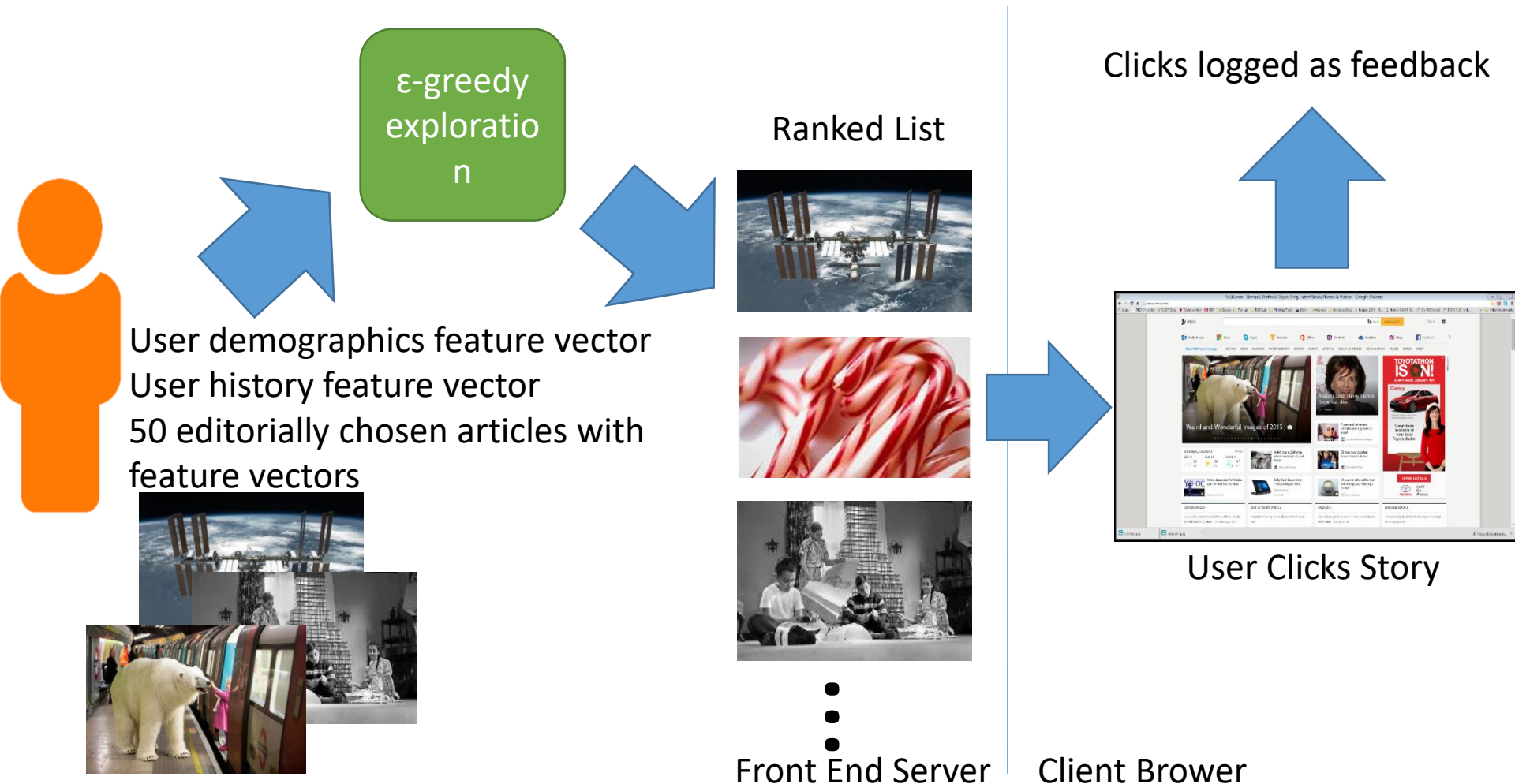
Online/Reinforcement Learning

The image shows the MSN homepage with a navigation bar at the top containing links to Outlook.com, Store, Skype, Rewards, Office, OneNote, OneDrive, Maps, and Facebook. Below the navigation bar are category links: Make MSN my homepage, DATING, NEWS, WEATHER, ENTERTAINMENT, SPORTS, MONEY, LIFESTYLE, HEALTH & FITNESS, FOOD & DRINK, TRAVEL, AUTOS, and VIDEO.

The main content area features several sections:

- Best of Late Night Videos:** A carousel of three videos. The first shows a woman eating Buffalo wings with the headline "Models devour Buffalo wings" (CNN). The second shows Sanders talking with Trump and Clinton with the headline "Sanders talks Trump, Clinton" (Newsy). The third shows Stewart returning to 'The Daily Show' with the headline "Stewart returns to 'The Daily Show'" (NowThis News).
- Marjorie Lord, 'Danny Thomas Show' star, dies:** A news article with a photo of Marjorie Lord and a source credit to Variety.
- 7 year-end retirement mistakes you may want to avoid:** A news article with a photo of people and a source credit to U.S. News & World Report.
- Clinton vows to defeat Islamic State if elected:** A news article with a photo of Clinton and a source credit to AP Associated Press.
- 15 ways to drink coffee that will change your mornings forever:** A lifestyle article with a photo of a cup of coffee and a source credit to Gourmandize.
- Wife's role in California attack raises fear of jihad brides:** A news article with a photo of a couple and a source credit to AP Associated Press.
- Daily Deal:** An advertisement for an Asus TP550LA laptop for \$399, sponsored by Microsoft.
- Weather:** A section for Montreal, Canada, showing forecasts for Saturday (50°/33°), Sunday (38°/35°), and Monday (50°/41°).
- Advertisements:** A large Toyota advertisement for the "TOYOTATHON IS ON!" event, featuring a red Camry and a woman in a Santa hat. The ad promotes great deals available at local Toyota dealers and includes a link to "OFFER DETAILS".
- Editors' Picks:** A section with two articles: "How police duty belt went from Officer Friendly to Mad Max in 30 years" (The Washington Post) and "Bruce Springsteen Fans Upset About 'River Tour' Ticket Prices, Resale Scams" (Gossip Cop).
- Best of Week's Video:** A section with two videos: "Reporter covering storm blows Internet away" (CNN) and "Epic fails: How not to fit a rear wiper blade on your car" (Rumble).
- Careers:** A section with two articles: "The 50 best places to work in 2016, according to employees" (Business Insider) and "15 blue-collar jobs for adrenaline junkies" (RWM.org).
- Weekend Reads:** A section with two articles: "The haunting link between two mass shootings" (The Washington Post) and "Newborns die after being sent home with drug-dependent mothers" (Reuters).

Online/Reinforcement Learning



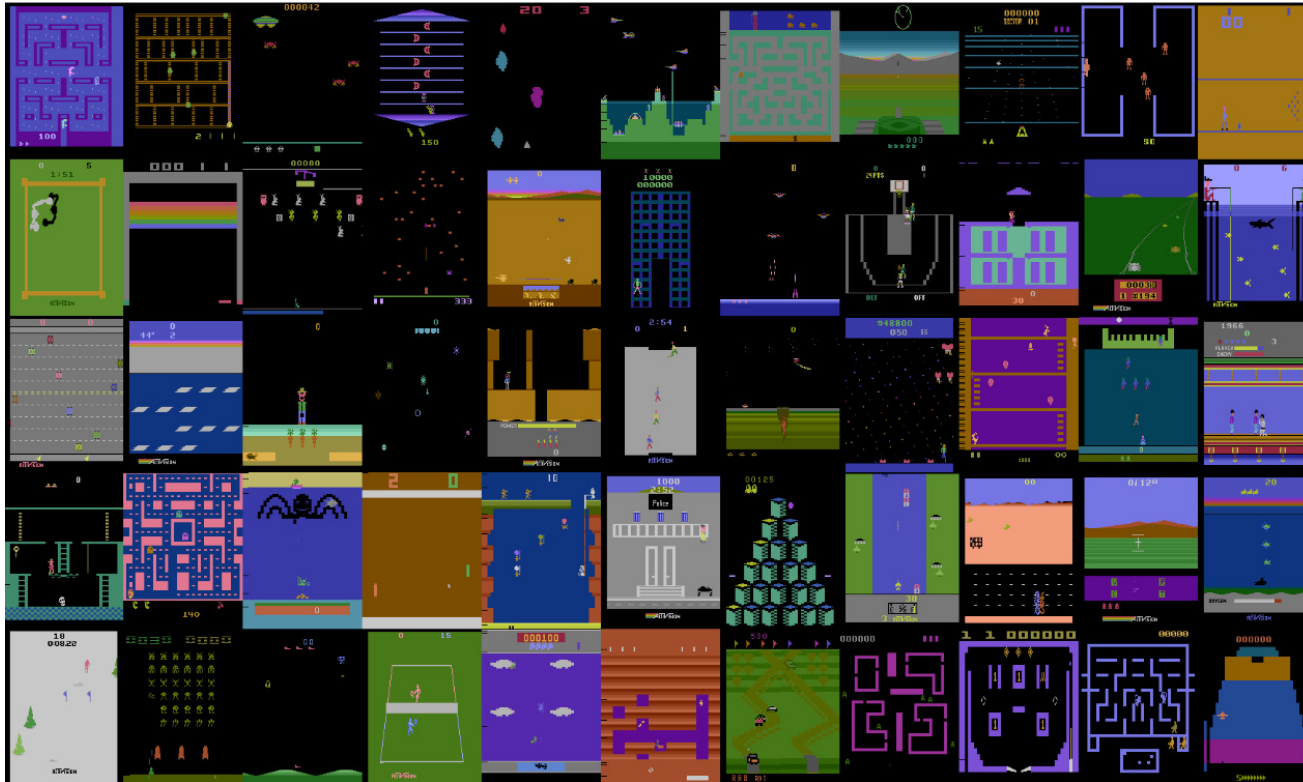
¹Reference: Alekh Agarwal et al., <http://arxiv.org/abs/1606.03966>

Online/Reinforcement Learning



¹Reference: DeepMind, March 2016

Online/Reinforcement Learning



Caveat

- Measurable metrics of business success take priority over technical success metrics
- Need to ask:
 - Does a $Y\%$ increase in classification accuracy help in $X\%$ increase in sales?
 - Does a $Z\%$ increase in classification accuracy due to using a deep learning solution help the bottom-line?
 - What is the technical debt incurred? Who will maintain?

Questions?

Today's Outline

- Course Logistics
- Introduction to the Course
- Getting Started with Neural Nets
 - Classification
 - Backpropagation

Classification

- Classification
 - Data
 - Model
 - Loss
 - Optimization

Classification



- To design the classifier, we need
 - Training data
 - Model specification for the classifier
 - Loss function to define the best model
 - Optimization to get to the best model

Data (I)

- Lets pick a domain: vision
- What is an image?
 - A bunch of numbers between 0 to 255
 - A 3 dimensional array
 - The same object can look different based on
 - Location of the camera
 - Location of the light source
 - Rigidity of the object
 - Occluding objects
 - Background
 - Variation across objects of the same category

Data (II)

- Say we have N training examples $(x_i, y_i), i = 1, \dots, N$
 - x_i is the feature vector for the i^{th} example
 - y_i is the label for the i^{th} example
- Before deep learning
 - Carefully designed features
 - Histogram of colors
 - Histogram of Oriented Gradients (HOG)
 - Scale Invariant Feature Transform (SIFT)
 - Various types of filters
- With deep learning
 - Almost no feature engineering

Model (I)

- Parametric vs non-parametric
- Example:
 - Logistic classifier is parametric
 - K-Nearest Neighbor is a non-parametric classifier
- We will focus on parametric models
- A fixed set of **parameters** and **hyper-parameters** determine a model completely

Model (II)

- Pick a concrete parametric model $f(x, W, b)$
 - x is the input ($d \times 1$ dimensional)
 - Vectorize the image or get features
 - W is a parameter ($p \times d$ dimensional)
 - b is also a parameter ($p \times 1$ dimensional)
- Let $f(x, W, b) = Wx + b$
 - This is a linear model
 - We will change this later
 - The output of the linear model is a vector of scores

Model (III)

- Given a model (i.e., a fixed W, b pair) our classifier can be
 - Pick the index with the highest ‘score’
 - $\hat{l} = \operatorname{argmax}_{\{j=1,\dots,p\}} f(x, W, b)$
 - Pick the index with the highest ‘probability’
 - Need a map/function from scores to probabilities
- We want to use the best model. How?
 - Define best: Loss function
 - Find the best: Optimization

Loss functions (I)

- Let the j^{th} coordinate of $f(x, W, b)$ be s_j
- Loss L_{data} is defined over the training data
- Is chosen to be decomposable over N terms, one per example
 - $L_{data} = \sum_{i=1}^N L_i$

Loss functions (I)

- Let the j^{th} coordinate of $f(x, W, b)$ be s_j
- Loss L_{data} is defined over the training data
- Is chosen to be decomposable over N terms, one per example
 - $L_{data} = \sum_{i=1}^N L_i$
- Logistic loss (**Cross-entropy** or **softmax**) for example i
 - $L_i = -\log P(Y = y_i | X = x_i)$ where
 - $P(Y = j | X = x_i) = \frac{e^{s_j}}{\sum_k e^{s_k}}$
- SVM loss (2 class, W is a row vector) for example i
 - $L_i = \max(0, 1 - y_i s_{y_i})$

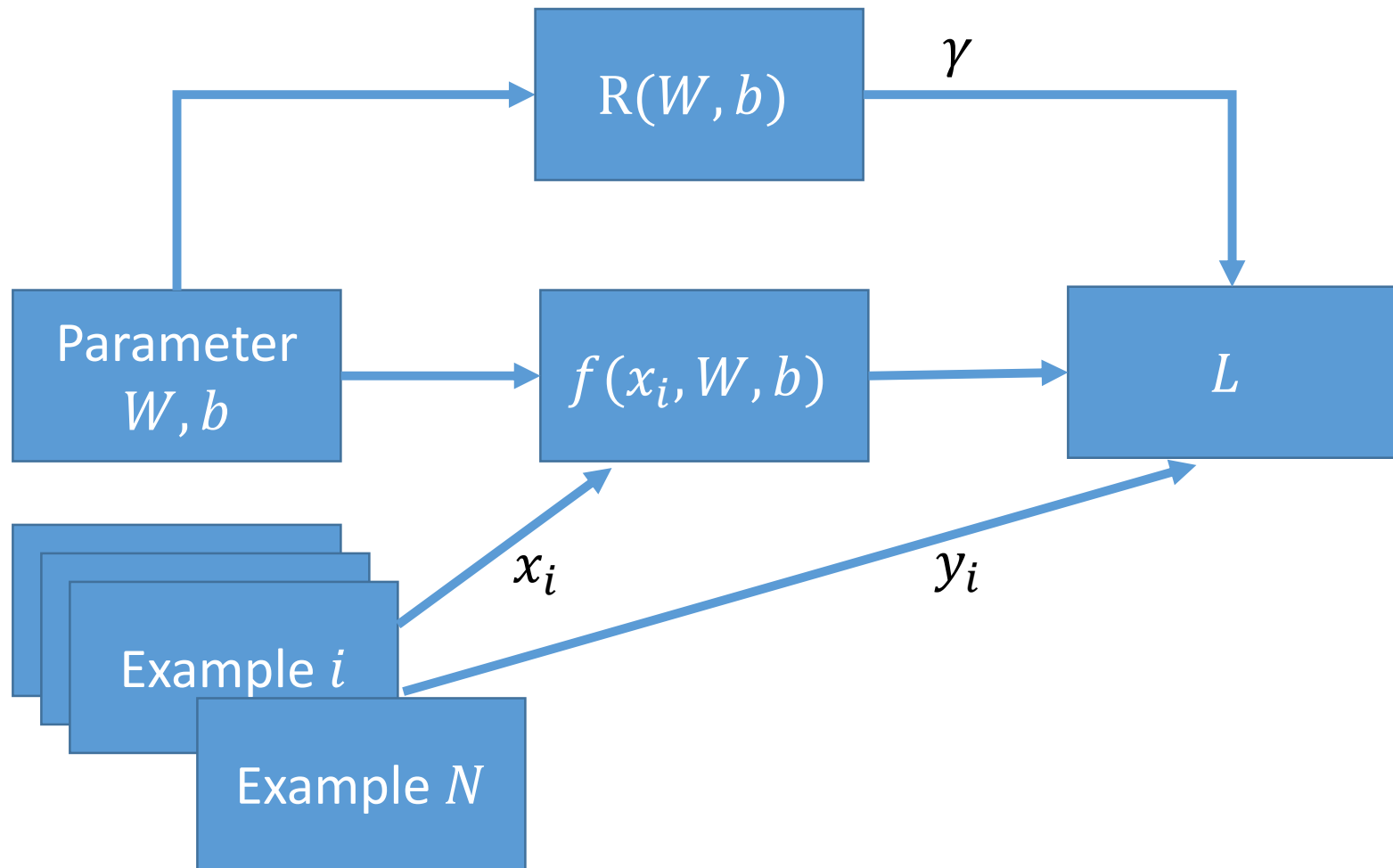
Loss functions (II)

- Need for regularization
 - Unique model
 - Desired model
 - Control overfitting
- Final loss $L = L_{data} + \lambda R(W, b)$
- $R(W, b)$ can be just a function of W or b or both

Loss functions (III)

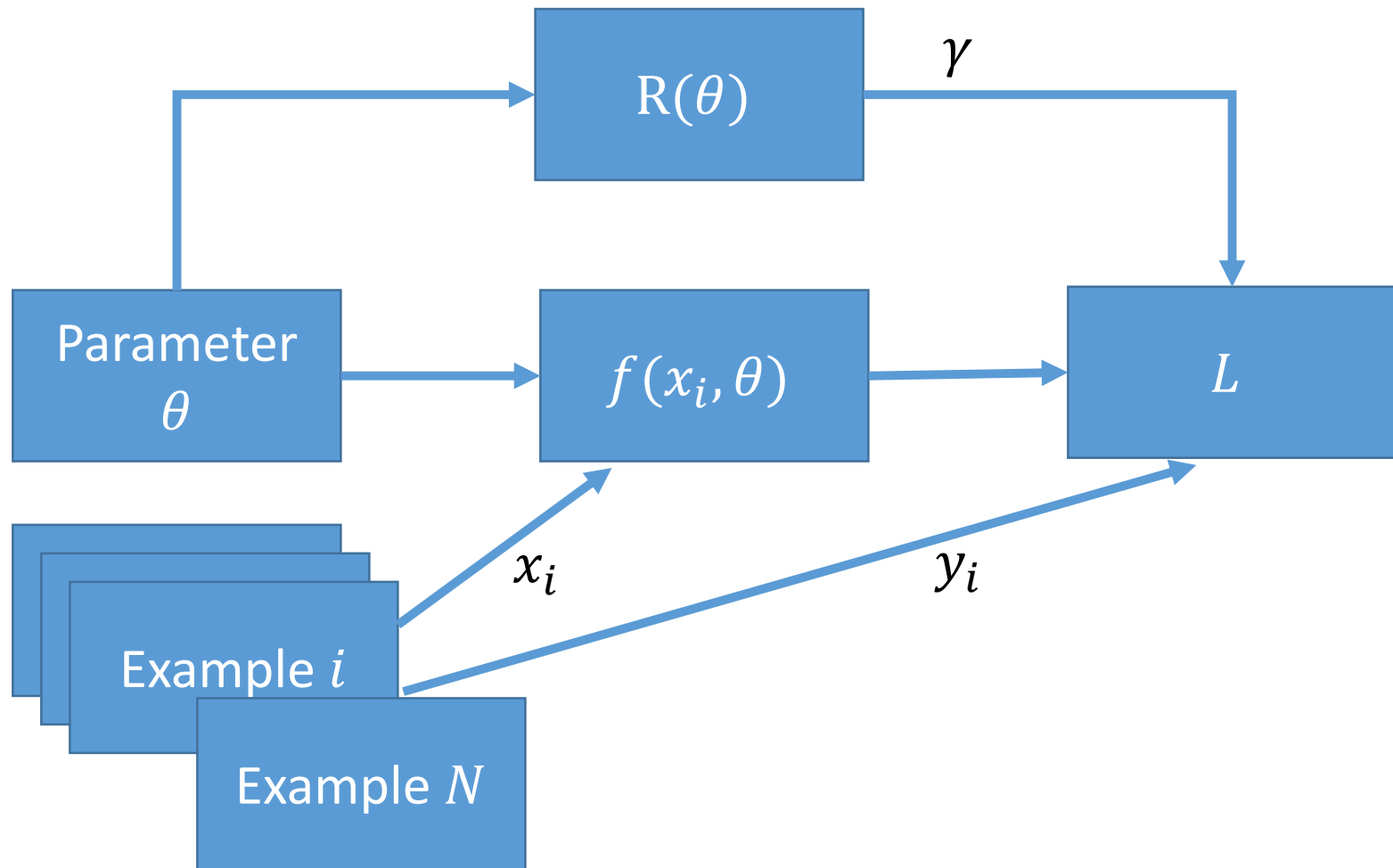
- L2 regularization: $||W||_2^2 = \sum_i \sum_j W_{ij}^2$
- L1: $||W||_1 = \sum_i \sum_j |W_{ij}|$
- Elastic net: $\alpha ||W||_1 + (1 - \alpha) ||W||_2^2$
- Regularization may not always be an explicit function of the parameters
 - We will see **dropout** later

Optimization (I)



Need to find parameters W, b and hyper-parameter γ

Optimization (I)



Need to find parameters θ and hyper-parameter γ

Optimization (II)

- Many ways to optimize
- We will focus on first order methods
 - Key ingredient: Gradient
- Gradient is the vector of partial derivatives of a function
- Can be computed
 - Numerically: $\lim_{h \rightarrow 0} \frac{f(z+h) - f(z)}{h}$
 - Analytically: Calculus and chain rules

Optimization (III)

- Build on the intuition
 - Start with a model (i.e., W_0, b_0)
 - Evaluate L for this model on the training data
 - Change W_0, b_0 to W_1, b_1 such that the new L is smaller
 - Repeat
- This intuition is the essence of Gradient Descent methods
 - Gradient of L with respect to the parameters is used to change W_0, b_0 to W_1, b_1

Optimization (IV)

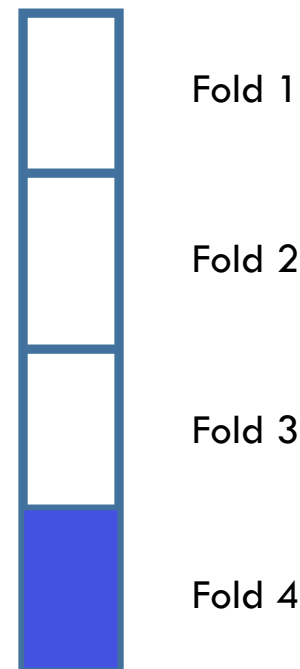
- Example method: **Batched Gradient Descent**
- Get a sample of training data
 - Example: AlexNet¹ used 256 examples as one batch
- Get gradient of L with respect to parameters W, b
- Update
 - $W_{k+1} \leftarrow W_k - \alpha \nabla_W L$
 - $b_{k+1} \leftarrow b_k - \beta \nabla_b L$
- Step sizes (**learning rates**) α, β need careful choice

¹Krizhevsky et al. NIPS Deep Learning Workshop 2012

Optimization (V)

- Tuning the hyper-parameter(s)
 - Break dataset into two parts: test and train
 - Remove test data access while you are tuning the parameters of your model
 - Do cross validation to tune **hyper-parameters**

Essentially cycle through the choice of validation fold
Optimize **parameters** over the remaining folds



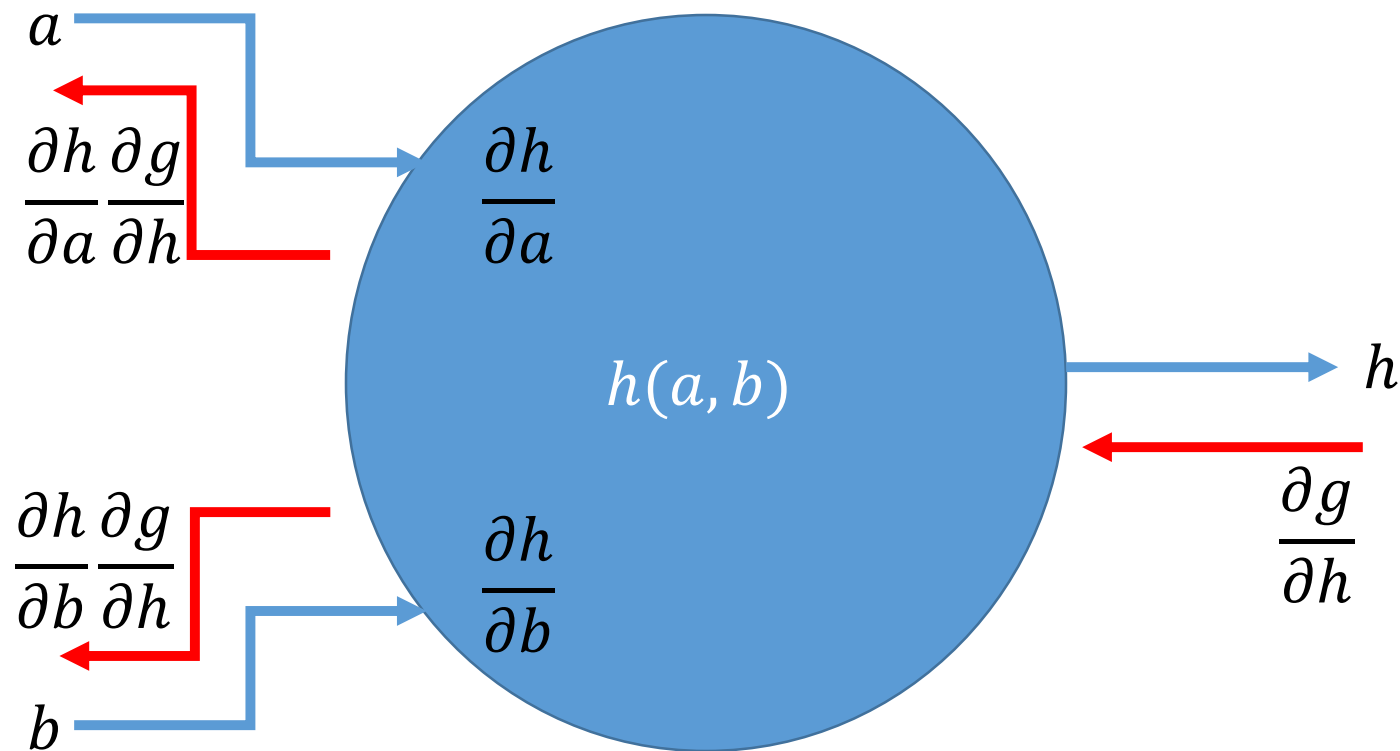
Questions?

Today's Outline

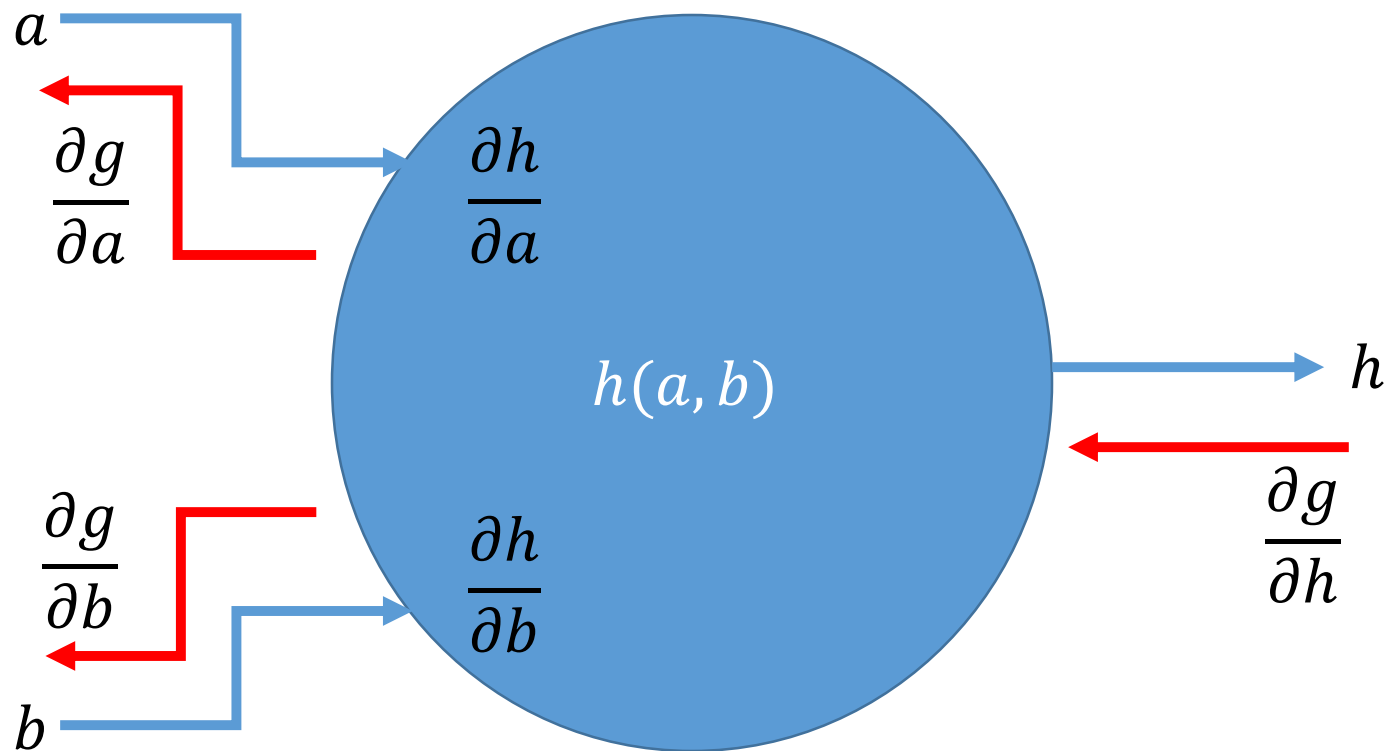
- Course Logistics
- Introduction to the Course
- Getting Started with Neural Nets
 - Classification
 - Backpropagation

Backpropagation

- An efficient way to get the gradient needed for optimization

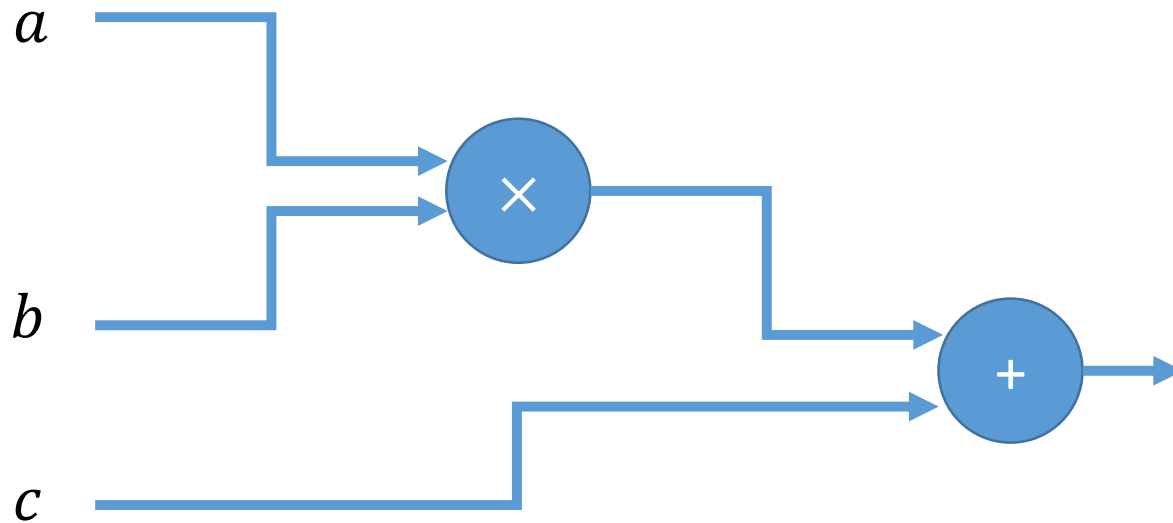


Backpropagation



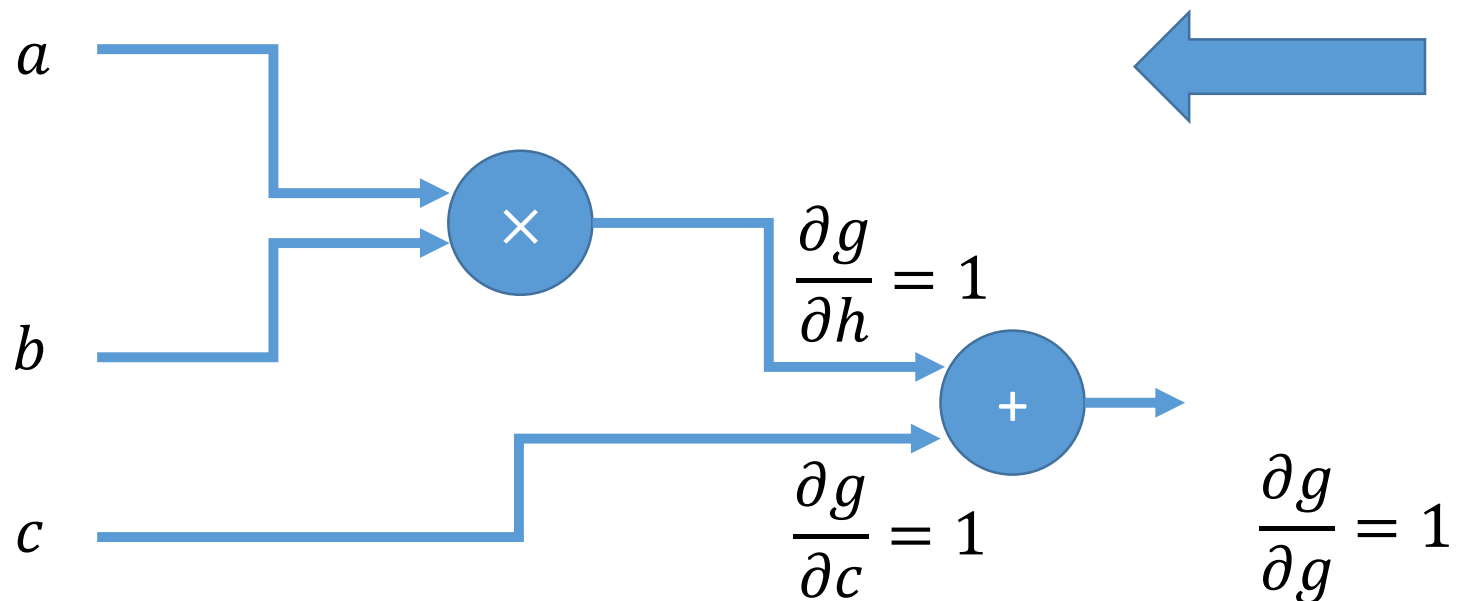
Notion of a Computational Graph

- Consider a function $g(a, b, c) = a * b + c$
- Draw a graph



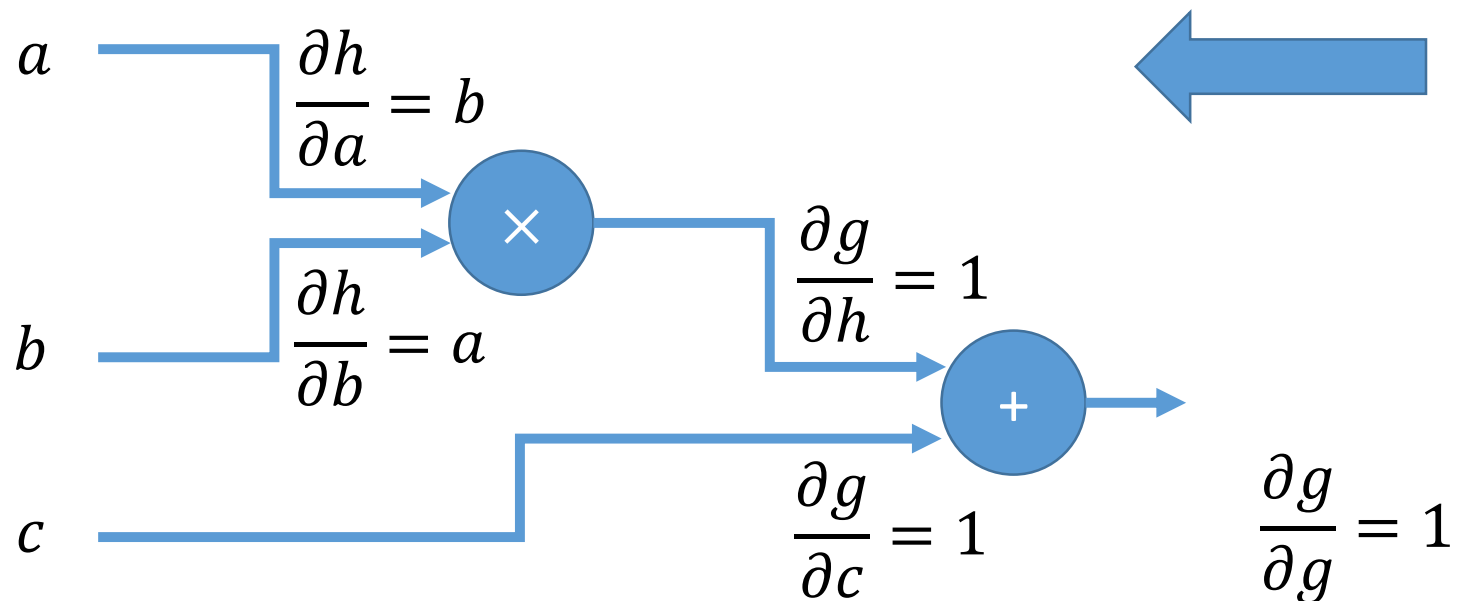
Backprop Example 1

- The circles represent compute nodes
- Let $h = a * b$. Then $g = h + c$



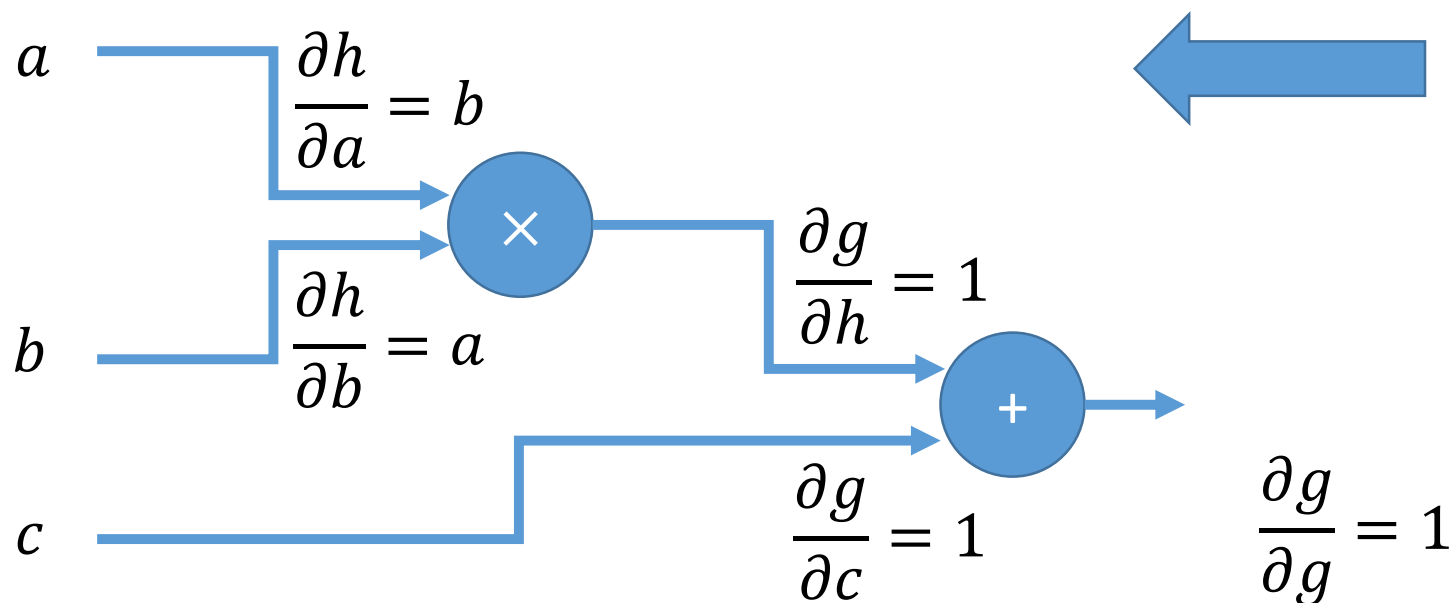
Backprop Example 1

- The circles represent compute nodes
- Let $h = a * b$. Then $g = h + c$



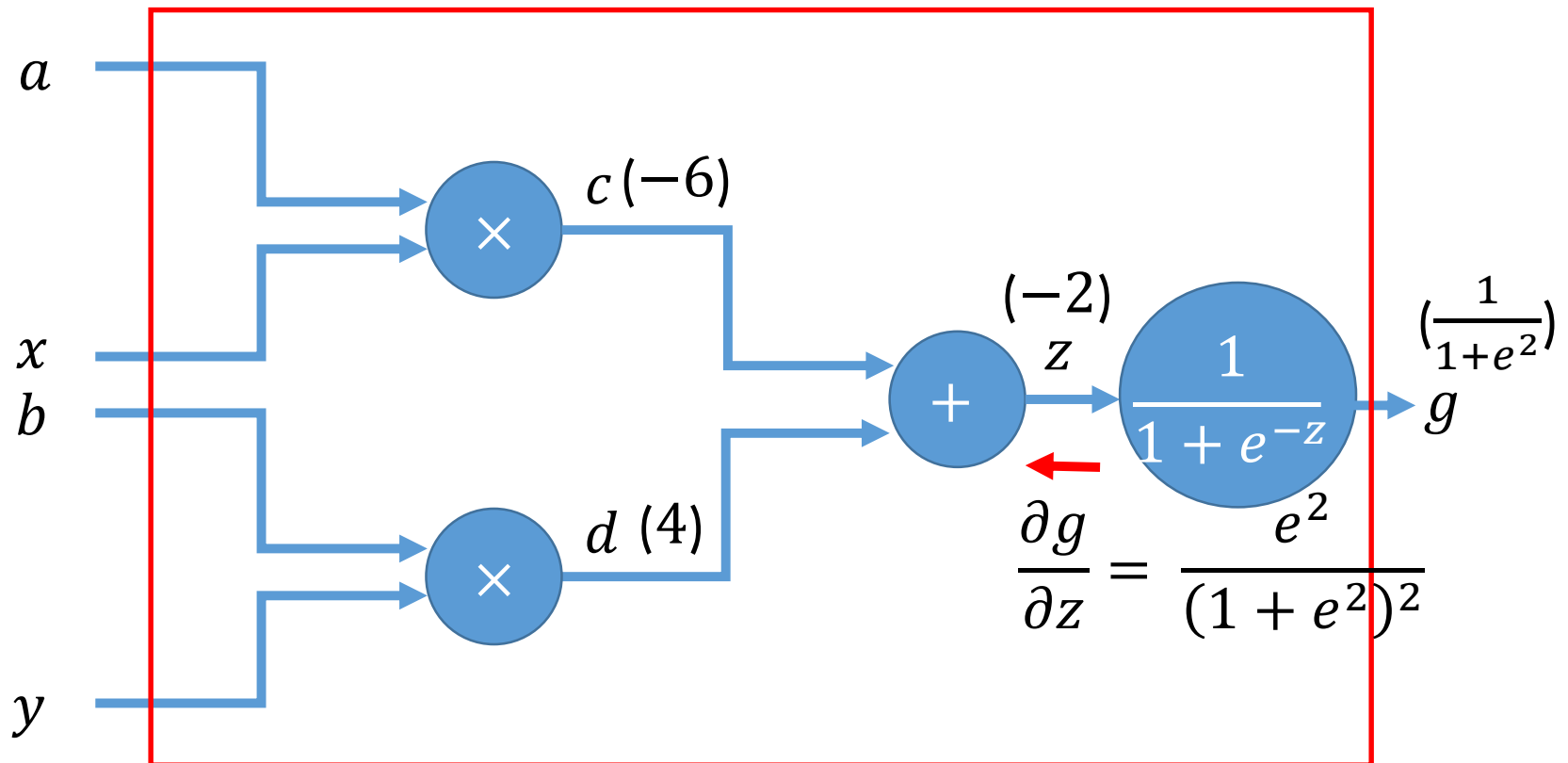
Backprop Example 1

- We can find $\frac{\partial g}{\partial a}$, $\frac{\partial g}{\partial b}$ and $\frac{\partial g}{\partial c}$ by chain rule!



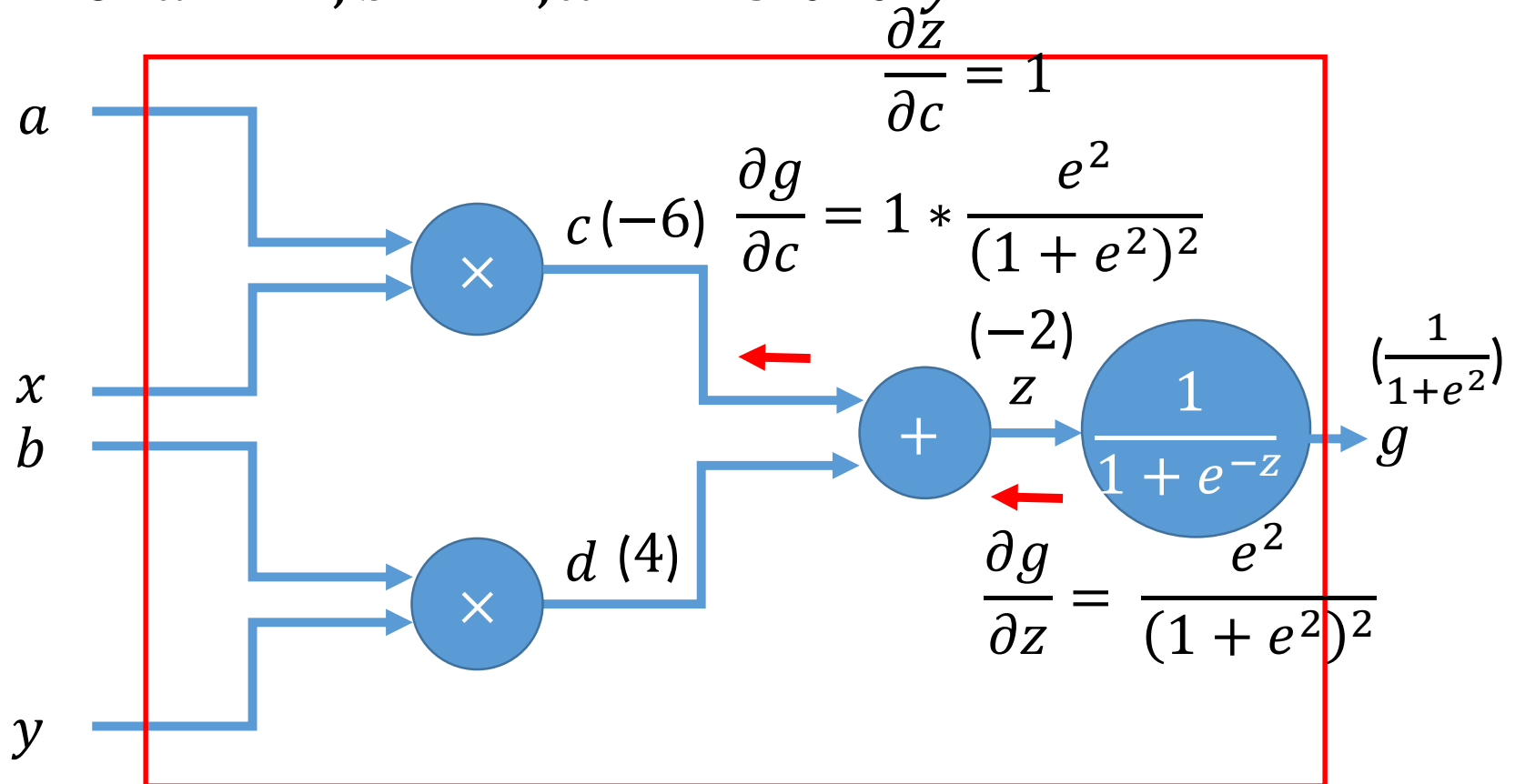
Backprop Example 2

- Consider a function $g(a, b, x, y) = \frac{1}{1+e^{-(ax+by)}}$
- Let $a = 2, b = 1, x = -3$ and $y = 4$



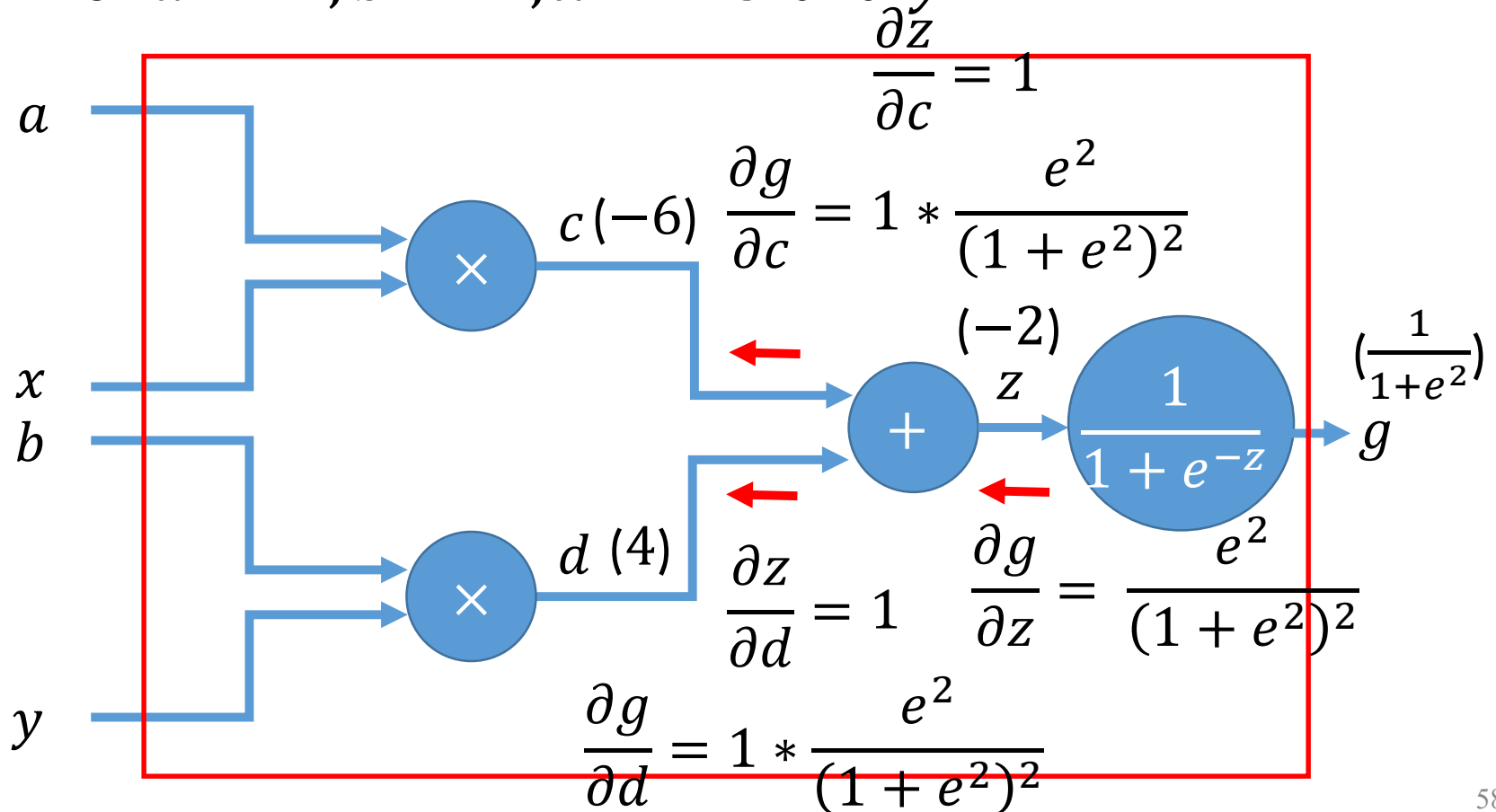
Backprop Example 2

- Consider a function $g(a, b, x, y) = \frac{1}{1+e^{-(ax+by)}}$
- Let $a = 2, b = 1, x = -3$ and $y = 4$



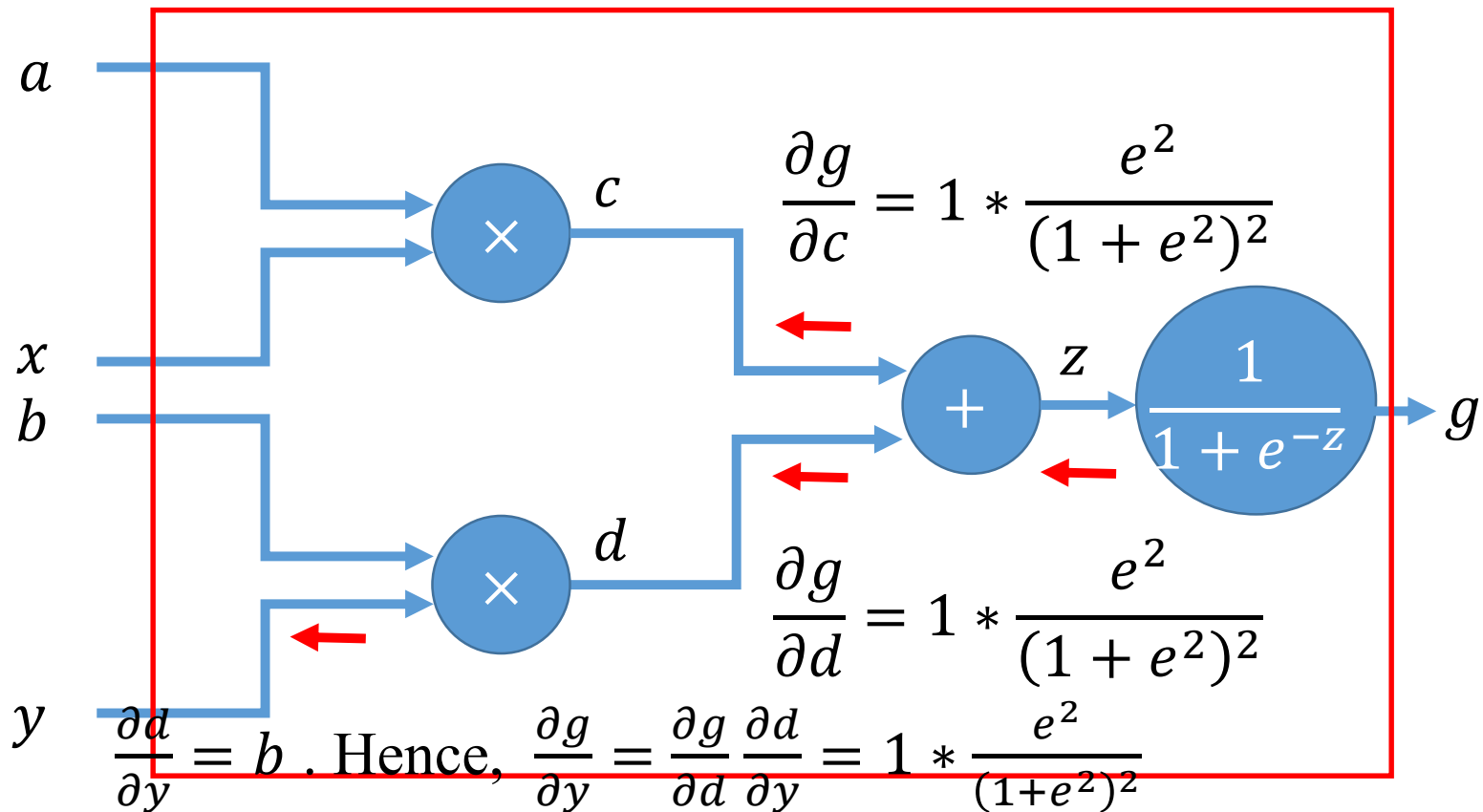
Backprop Example 2

- Consider a function $g(a, b, x, y) = \frac{1}{1+e^{-(ax+by)}}$
- Let $a = 2, b = 1, x = -3$ and $y = 4$

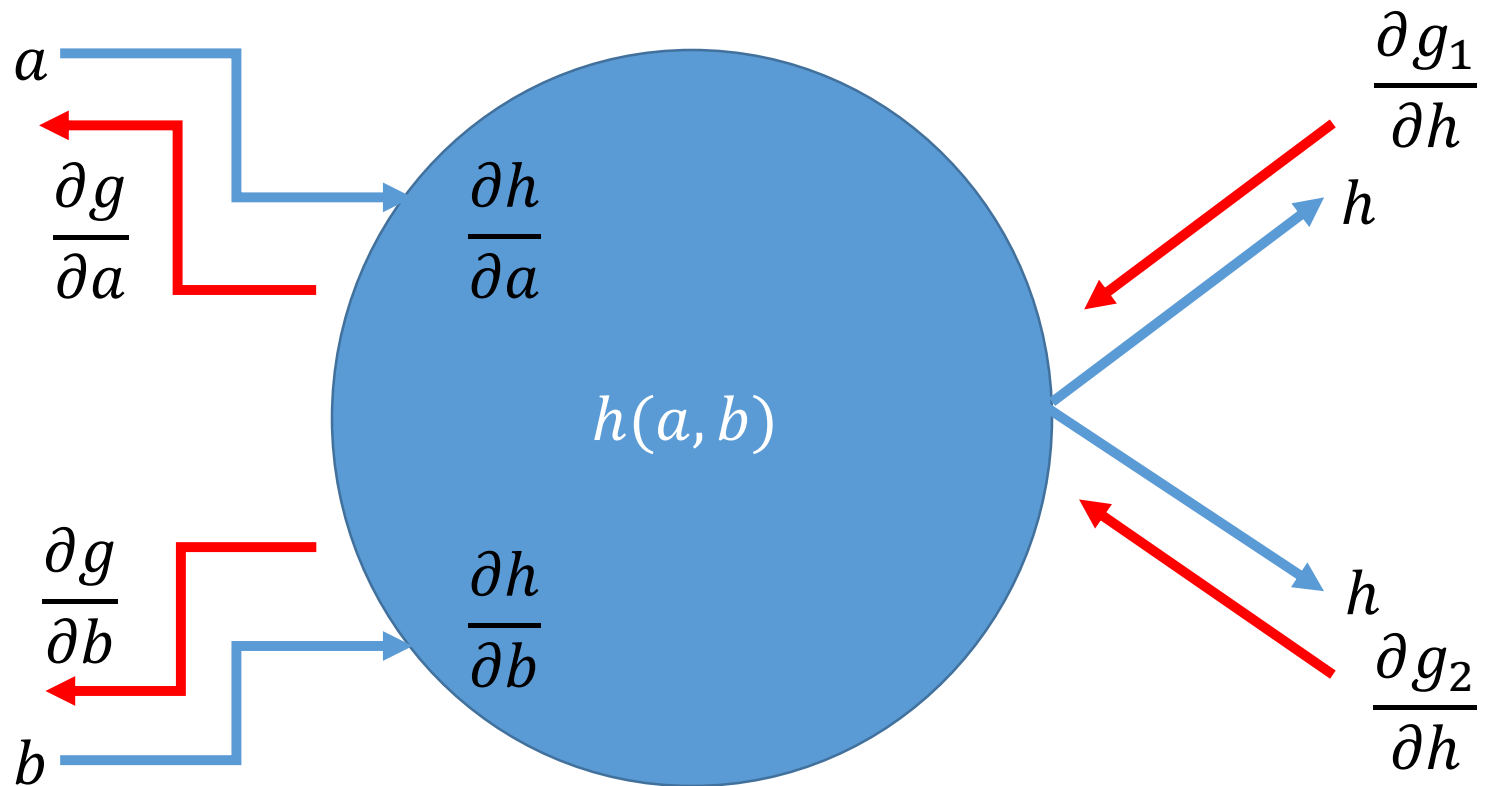


Backprop Example 2

- Consider a function $g(a, b, x, y) = \frac{1}{1+e^{-(ax+by)}}$
- Let $a = 2, b = 1, x = -3$ and $y = 4$

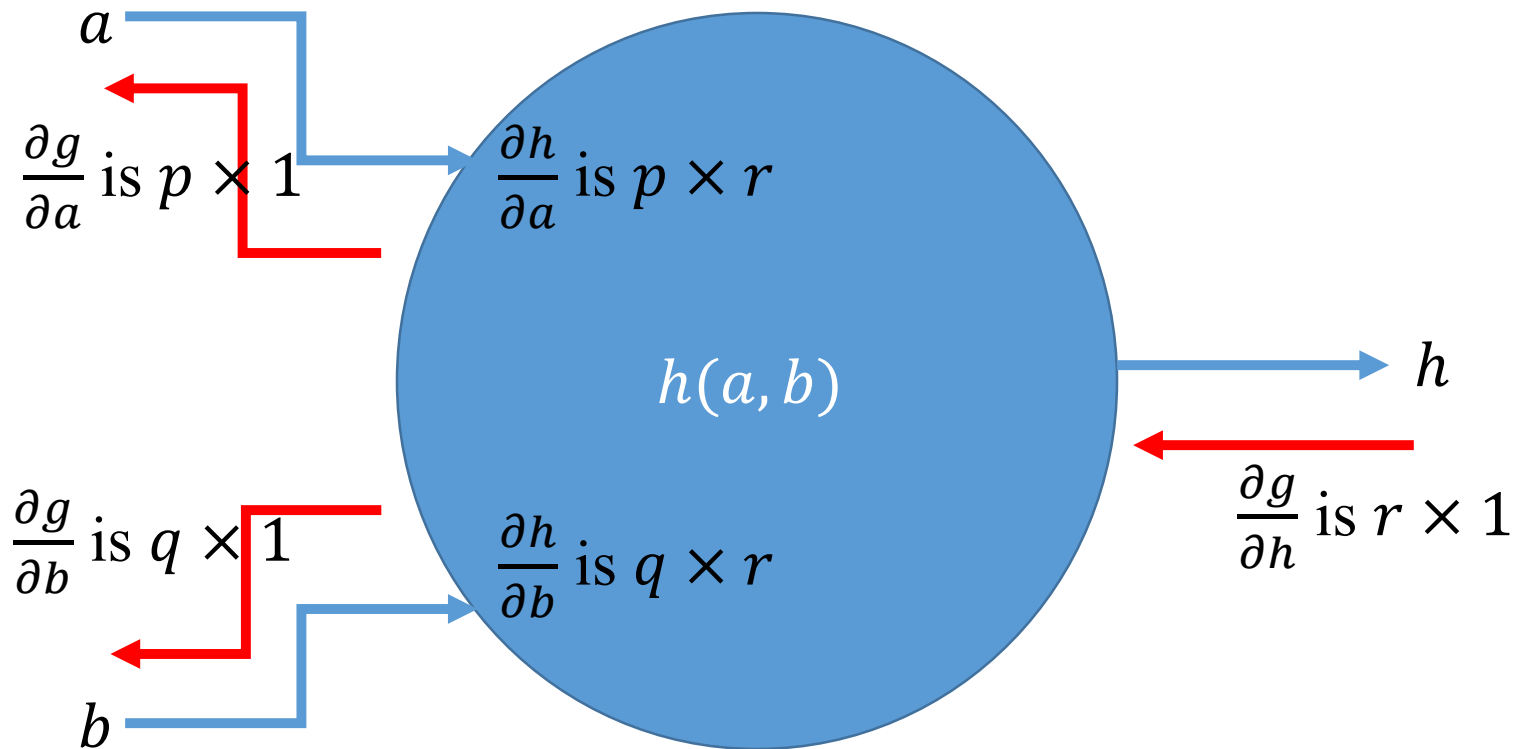


Backprop for multiple outputs



Backprop for vectors

- Say, a is $p \times 1$ dimensional, b is $q \times 1$ dimensional and h is $r \times 1$ dimensional and g is scalar



Node tracks matrices (cleverly)

Backprop API for a node

- Implement two functions
 - Forward
 - Backward
- Forward
 - Get input from preceding node(s)
 - Track inputs and local gradients
 - Return computation
- Backward
 - Get gradient from succeeding node(s)
 - Compute gradients (simple multiplication)
 - Return gradients to preceding node(s)

Computational Graph API

- Data structure a graph (nodes and directed edges)
- Implement two functions for it
 - Forward
 - Backward
- Forward
 - Recursively pass the inputs to the **next** nodes
 - Return L
- Backward
 - Recursively traverse the graph backwards
 - Return gradients

Backprop and batched Gradient Descent

- Choose a mini-batch (sample) of size B
- Forward propagate through the computation graph
 - Compute losses $L_{i_1}, L_{i_2}, \dots, L_{i_B}$ and $R(W, b)$
 - Get loss L for the batch
- Backprop to compute gradients with respect to W, b
- Update parameters W, b
 - In the direction of the negative gradient

Linear Classifier in Python

```
#Example modified from http://cs231n.github.io/neural-networks-case-study/

#Imports
import numpy as np #Represent ndarrays a.k.a. tensors
import matplotlib.pyplot as plt #For plotting
np.random.seed(0) #For repeatability of the experiment
import pickle #To read data for this experiment

#Setup
%matplotlib inline
plt.rcParams['figure.figsize'] = (10.0, 8.0) # set default size of plots
plt.rcParams['image.interpolation'] = 'nearest'
plt.rcParams['image.cmap'] = 'gray'
```


Linear Classifier in Python

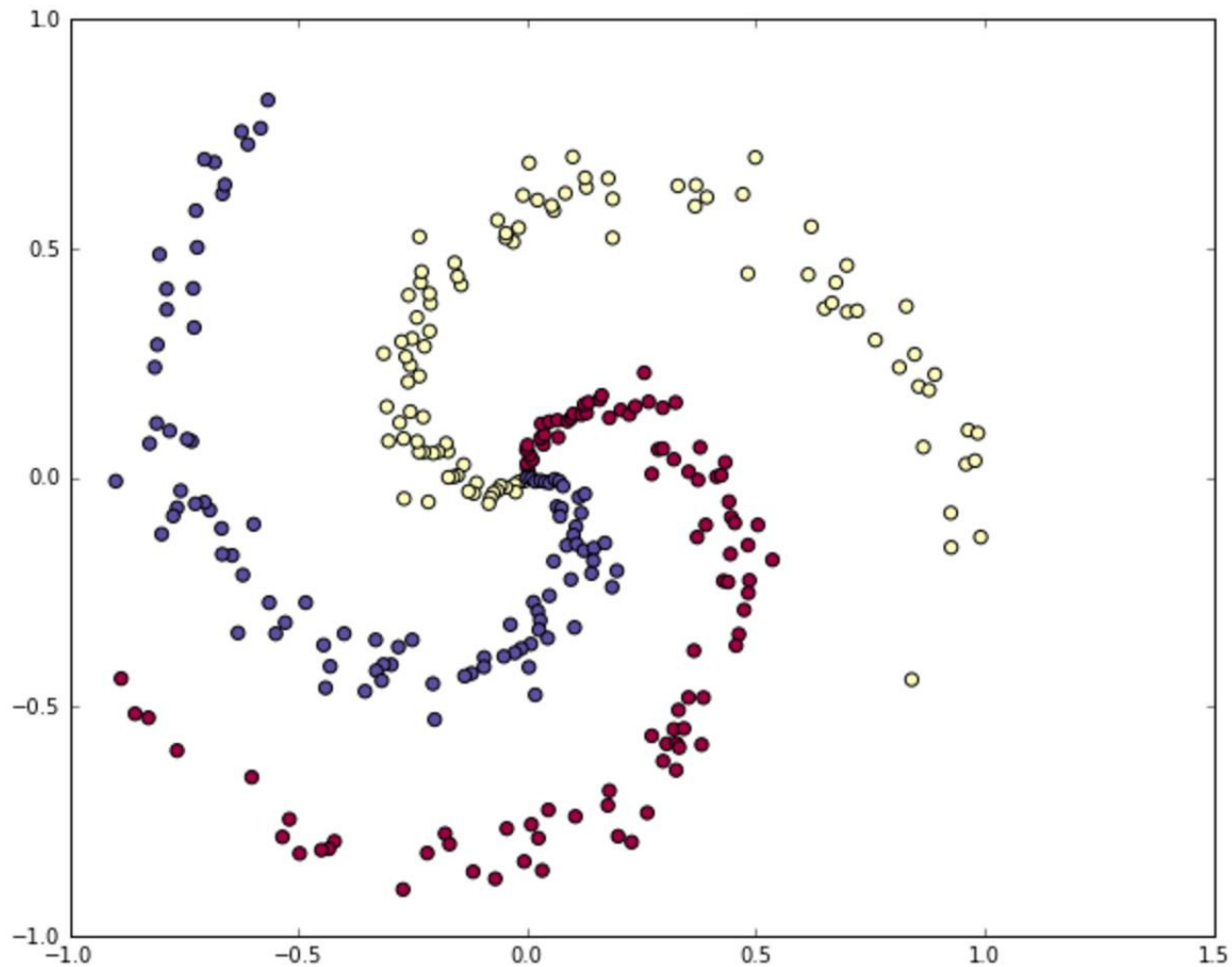
Data

```
#Read data
X = pickle.load(open('dataX.pickle','rb'))
y = pickle.load(open('dataY.pickle','rb'))

#Define some local variables
D = X.shape[1] #Number of features
K = max(y)+1 #Number of classes assuming class index starts from 0

#Plot the data
fig = plt.figure()
plt.scatter(X[:, 0], X[:, 1], c=y, s=40, cmap=plt.cm.Spectral)
```


Linear Classifier in Python



Linear Classifier in Python

Model

```
# Linear model

# Start with an initialize parameters randomly
W = 0.01 * np.random.randn(D,K)
b = np.zeros((1,K))

# Initial values from hyperparameter
reg = 1e-3 # regularization strength

#For simplicity, we will not optimize this using grid search here.
```


Linear Classifier in Python

```
#Perform batch SGD using backprop

#For simplicity we will take the batch size to be the same as number of examples
num_examples = X.shape[0]

#Initial value for the Gradient Descent Parameter
step_size = 1e-0 #Also called learning rate

#For simplicity, we will not hand tune this algorithm parameter as well.
```


Linear Classifier in Python

```
# gradient descent loop
for i in xrange(200):

    # evaluate class scores, [N x K]
    scores = np.dot(X, W) + b

    # compute the class probabilities
    exp_scores = np.exp(scores)
    probs = exp_scores / np.sum(exp_scores, axis=1, keepdims=True) # [N x K]

    # compute the loss: average cross-entropy loss and regularization
    corect_logprobs = -np.log(probs[range(num_examples),y])
    data_loss = np.sum(corect_logprobs)/num_examples
    reg_loss = 0.5*reg*np.sum(W*W)
    loss = data_loss + reg_loss
    if i % 10 == 0:
        print "iteration %d: loss %f" % (i, loss)

    # compute the gradient on scores
    dscores = probs
    dscores[range(num_examples),y] -= 1
    dscores /= num_examples

    # backpropate the gradient to the parameters (W,b)
    dW = np.dot(X.T, dscores)
    db = np.sum(dscores, axis=0, keepdims=True)

    dW += reg*W # regularization gradient

    # perform a parameter update
    W += -step_size * dW
    b += -step_size * db
```


Linear Classifier in Python

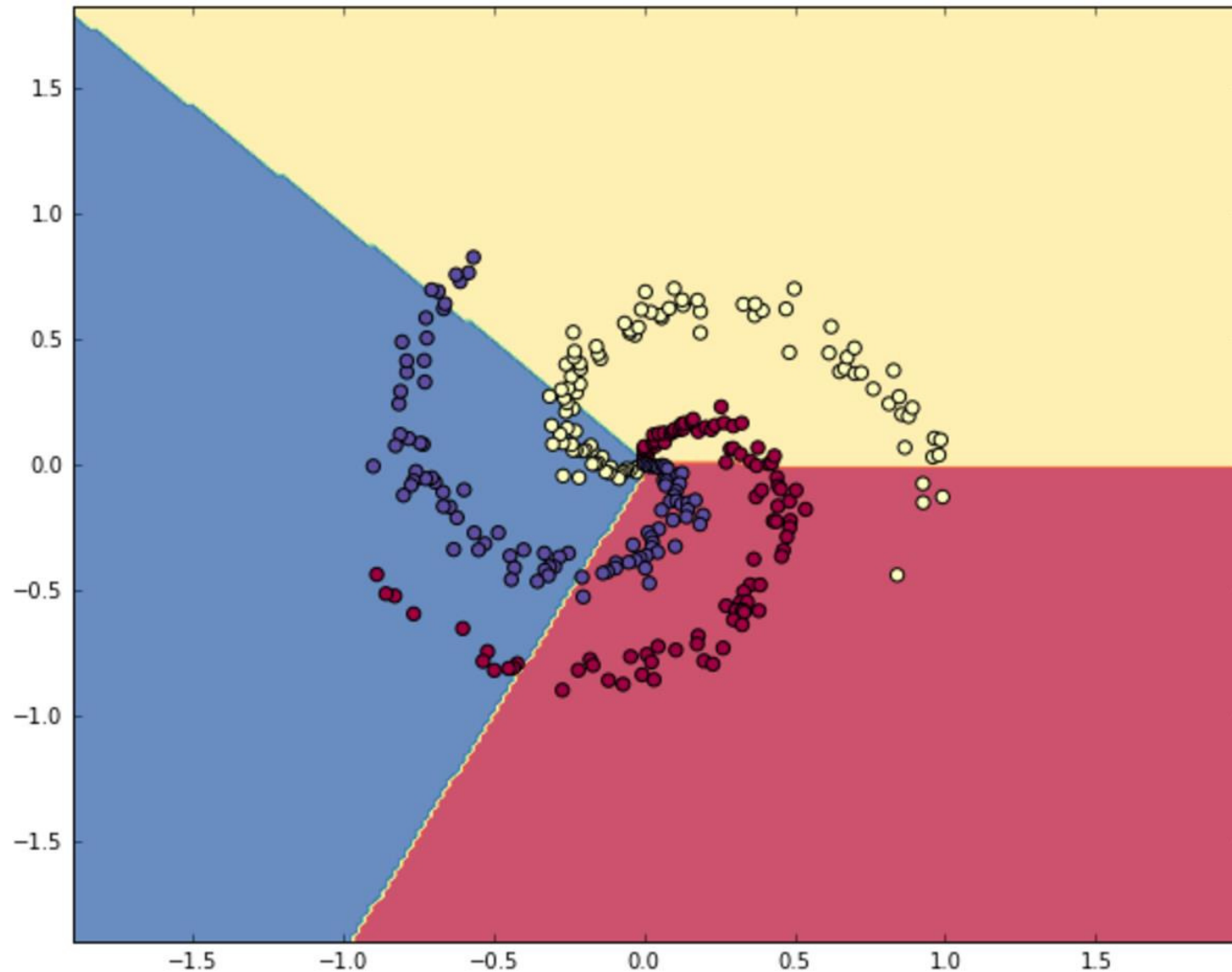
Post Training

```
# Post-training: evaluate test set accuracy

#For simplicity, we will use training data as proxy for test. Do not do this.
X_test = X
y_test = y

scores = np.dot(X_test, W) + b
predicted_class = np.argmax(scores, axis=1)
print 'test accuracy: %.2f' % (np.mean(predicted_class == y_test))
```


Linear Classifier in Python



Questions?

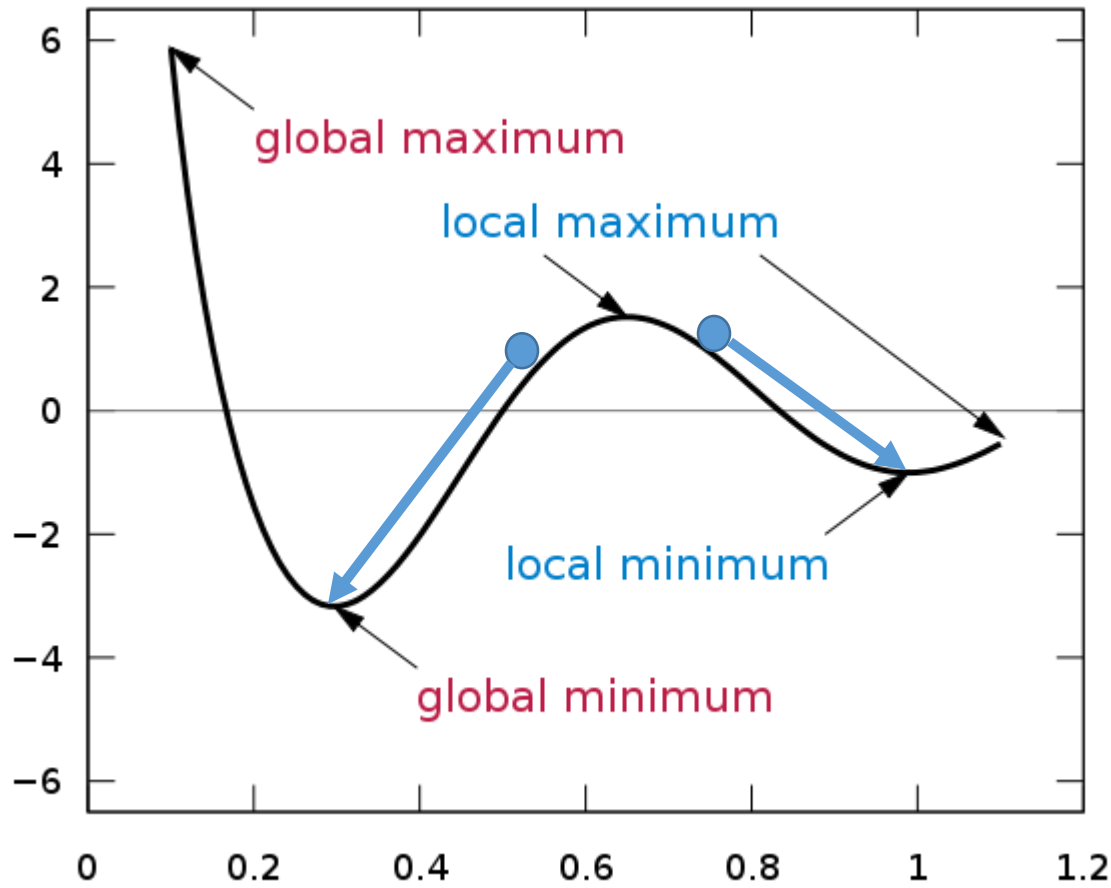
Summary

- Data variety poses challenges
 - Missing
 - Noisy
- Complex decisions poses challenges
 - Learning on the go
- We reviewed classification
 - Regression would have similar considerations
- Discussed backpropagation
 - A useful method for optimizing for the best model parameters

Appendix

Gradient Descent

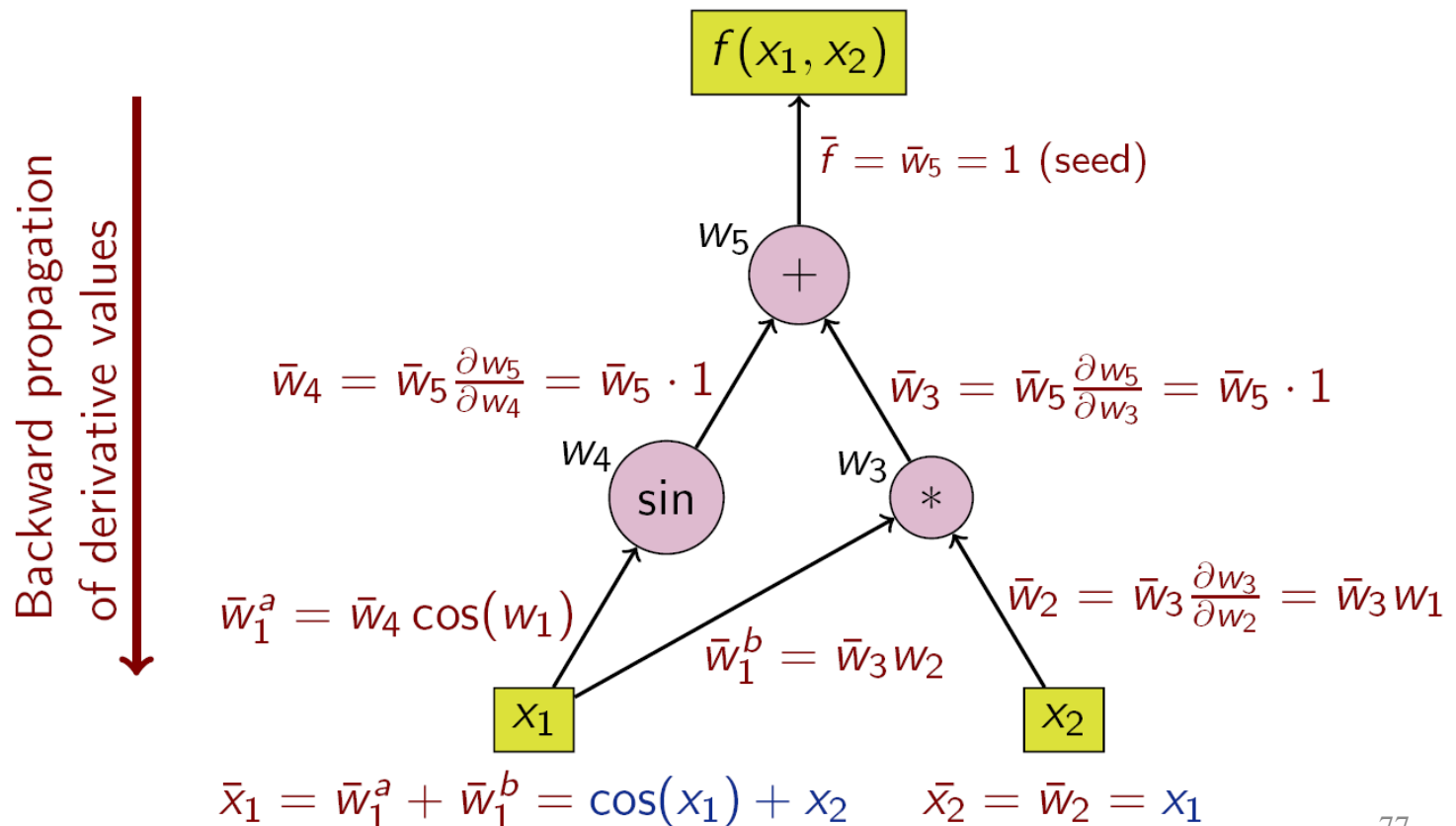
Gradient descent can only reach local optima



Reverse mode AutoDiff

- Backpropagation is a case of reverse accumulation automatic differentiation¹

An example from wikipedia



¹See https://en.wikipedia.org/wiki/Automatic_differentiation

Advanced Prediction Models

Deep Learning, Graphical Models and Reinforcement
Learning

Today's Outline

- Python Walkthrough
- Feedforward Neural Nets
- Convolutional Neural Nets
 - Convolution
 - Pooling

Python Walkthrough

Python Setup (I)

- Necessary for the programming portions of the assignments
- More precisely, use Ipython (ipython.org)

IP[y]: IPython
Interactive Computing

[Install](#) · [Documentation](#) · [Project](#) · [Jupyter](#) · [News](#) · [Cite](#) · [Donate](#) · [Books](#)

IPython provides a rich architecture for interactive computing with:

- A powerful interactive shell.
- A kernel for [Jupyter](#).
- Support for interactive data visualization and use of [GUI toolkits](#).
- Flexible, [embeddable](#) interpreters to load into your own projects.
- Easy to use, high performance tools for [parallel computing](#).

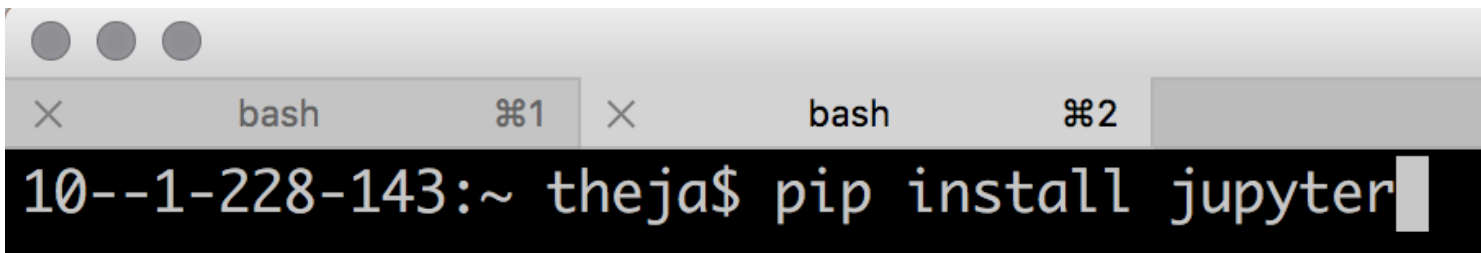
Python Setup (II)

- Install Python
 - Use Anaconda
(<https://www.continuum.io/downloads>)
 - Python 2 vs Python 3 (your choice)



Python Setup (III)

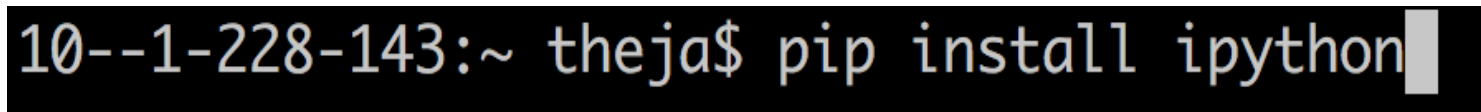
- Install Ipython/Jupyter
 - If you installed the Anaconda distribution, you are all set
 - Else use the command on the command-line



A screenshot of a macOS terminal window. The window has a title bar with three colored buttons (red, yellow, green) and two tabs. The first tab is labeled 'bash' and '⌘1', and the second tab is labeled 'bash' and '⌘2'. The terminal content shows the prompt '10--1-228-143:~ theja\$' followed by the command 'pip install jupyter' with a cursor at the end.

```
10--1-228-143:~ theja$ pip install jupyter
```

or

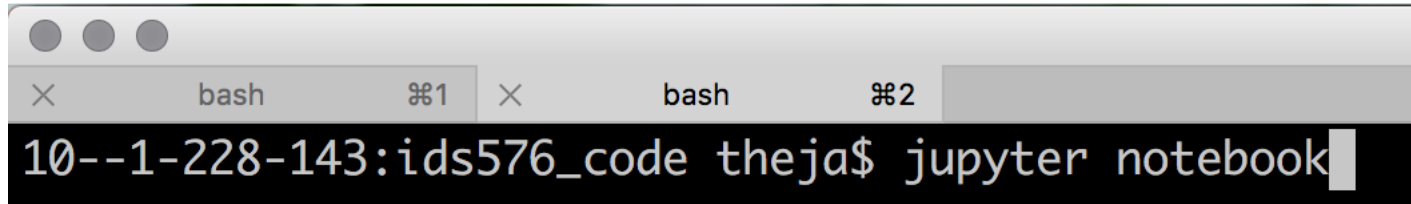


A screenshot of a terminal window showing the command to install ipython. The prompt is '10--1-228-143:~ theja\$' followed by 'pip install ipython' with a cursor at the end.

```
10--1-228-143:~ theja$ pip install ipython
```

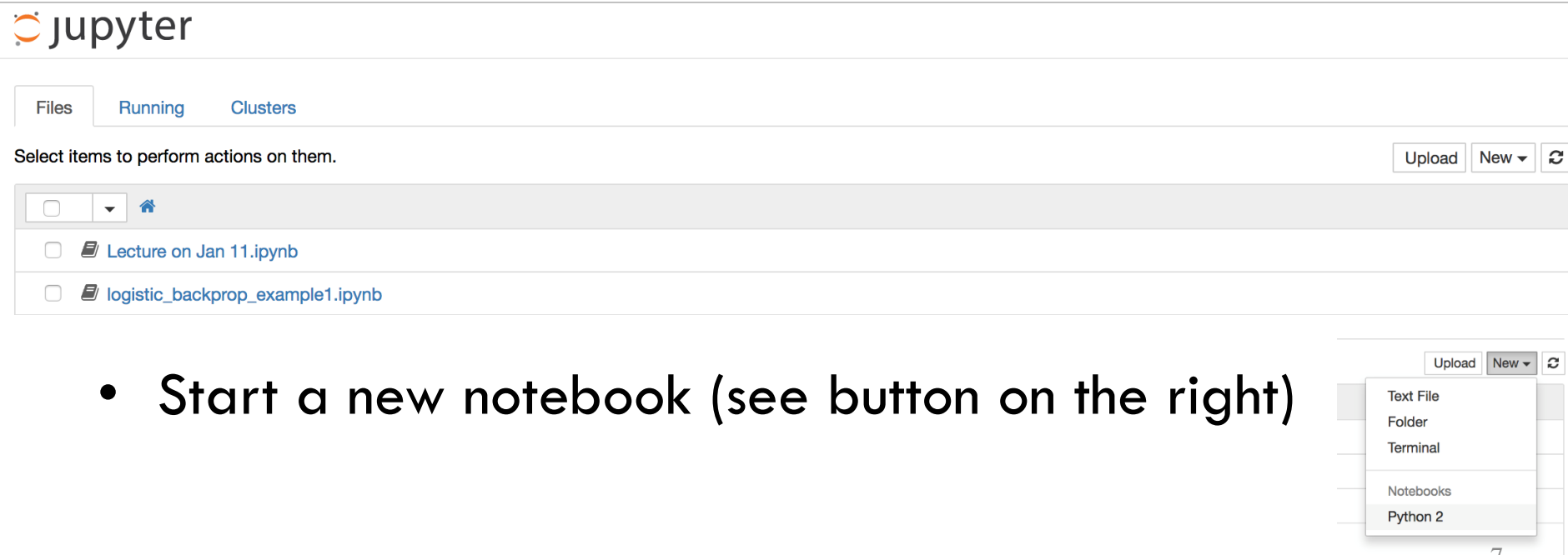

Python Setup (IV)

- Run Jupyter (or ipython)



```
10--1-228-143:ids576_code theja$ jupyter notebook
```

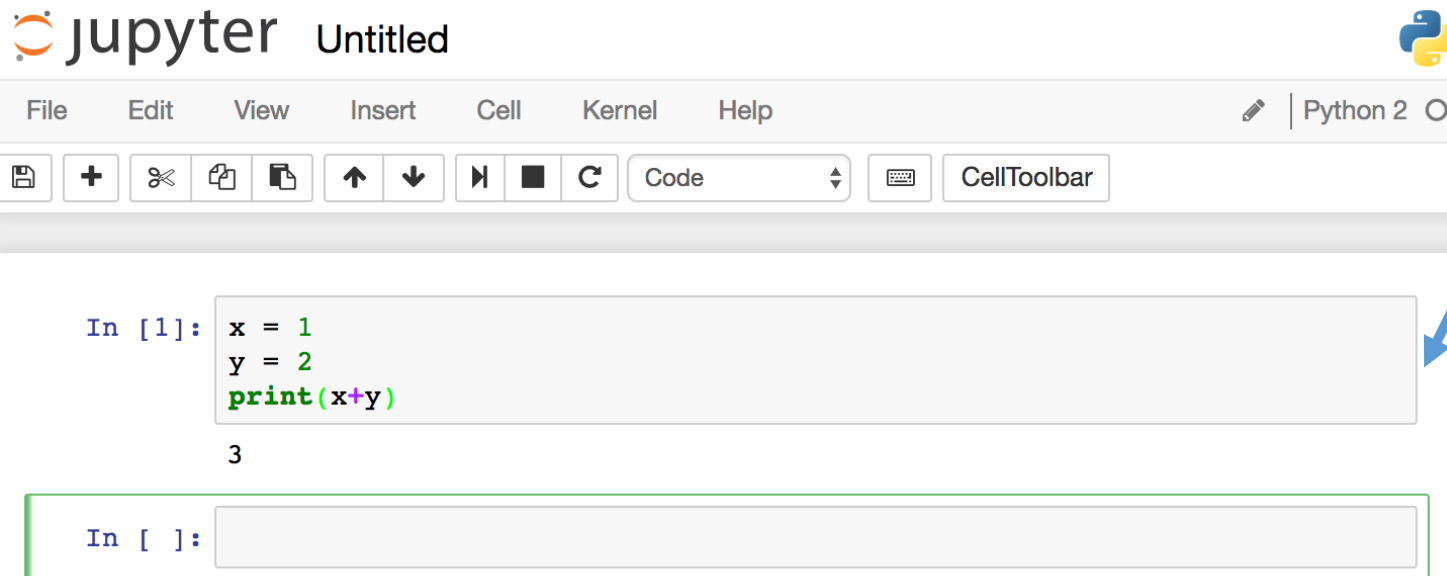
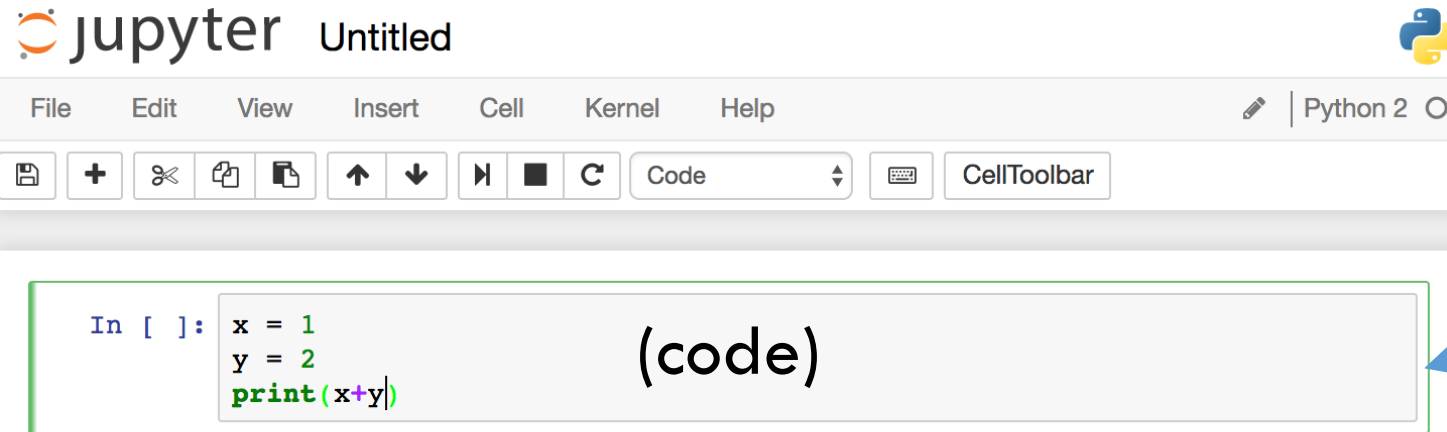
- Your browser with open a page like this



The screenshot shows the Jupyter web interface. At the top, there's a navigation bar with 'Files', 'Running', and 'Clusters' tabs. Below this, a message says 'Select items to perform actions on them.' followed by 'Upload', 'New', and a refresh icon. The file list shows two notebooks: 'Lecture on Jan 11.ipynb' and 'logistic_backprop_example1.ipynb'. On the right, a 'New' button is open, showing a dropdown menu with options: 'Text File', 'Folder', 'Terminal', 'Notebooks', and 'Python 2'.

- Start a new notebook (see button on the right)

Python Setup (V)



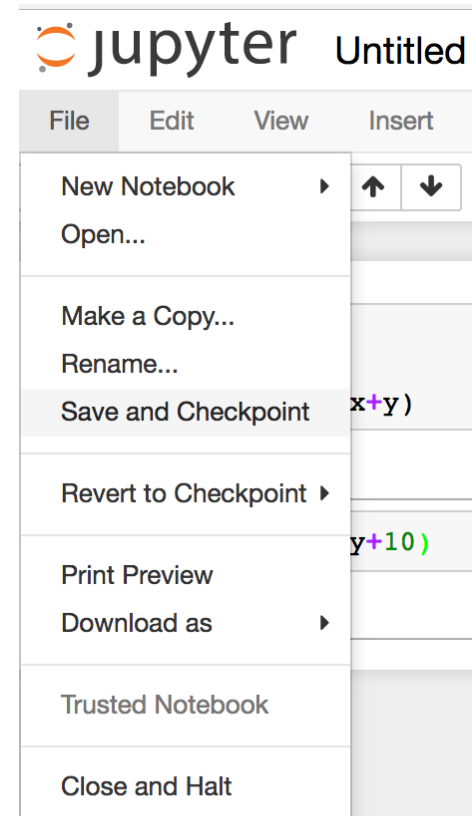
Python Setup (VI)

- Global variables are shared between cells
- Cells are typically run from top to bottom

```
In [1]: x = 1
        y = 2
        print(x+y)
        3

In [2]: print(y+10)
        12
```

- Save changes using the save button



Python Review

- General purpose programming language
- 2 vs 3 (3 is backward incompatible)
- Very similar to Matlab (and better) for scientific computing
- It is dynamically typed

Python Review: Data Types

```
In [1]: x = 3
        y = 3.0
        z = 2
        print(x)
        print(y)
        print type(x)
        print type(y)
        print(x/z)
        print(y/z)

3
3.0
<type 'int'>
<type 'float'>
1
1.5
```


Python Review: Data Types

```
|: x +=1 #This is a comment. No unary operators (x++ will not work)
   print(x)
   y **=2
   print y
```

4

9.0

```
|: a,b = True,False
   mystring = 'ids676'
   print a,b,mystring,'. In upper case: ' + mystring.upper()
```

True False ids676 . In upper case: IDS676

Python Review: List and Tuple

Dictionary, List, Tuple, Set

```
: mylist = ['i', 'd', 's']  
   mytuple = (5, 7, 6)  
   print mylist, mytuple  
  
['i', 'd', 's'] (5, 7, 6)
```

```
: mylist[0] = 'c'  
   mylist[1] = 'b'  
   mylist[2] = 'a'  
   mylist.append(5)  
   mylist.extend([7, 6])  
   print mylist  
  
['c', 'b', 'a', 5, 7, 6]
```


Python Review: Dictionary & Set

```
mylist[:2] = 'a','a'  
print mylist  
print set(mylist) #a set object will have unique elements
```

```
['a', 'a', 'a', 5, 7, 6]  
set(['a', 5, 6, 7])
```

```
course = {} #An empty dictionary/hash-map  
course[mytuple] = 'Advanced Prediction Models'  
course['572'] = 'Data Mining'  
print course
```

```
{(5, 7, 6): 'Advanced Prediction Models', '572': 'Data Mining'}
```


Python Review: Naïve for-loop

```
for x in mylist: #A for loop  
    print x
```

```
a  
a  
a  
5  
7  
6
```


Python Review: Function

Functions

```
import math, numpy
def softmax(z):
    return (1.0/(1+math.e**(-z)))
print softmax(-20)
print softmax(numpy.asarray([-1,0,1]))
```

2.06115361819e-09

[0.26894142 0.5 0.73105858]

Python Review: Numpy

Numpy

```
a = numpy.array([-1,0,1])
print a,type(a),a.shape,a.dtype
b = numpy.array([[1.0,2,3],[1,2,3]])
print b, type(b), b.shape,b.dtype
```

```
[-1  0  1] <type 'numpy.ndarray'> (3,) int64
[[ 1.  2.  3.]
 [ 1.  2.  3.]] <type 'numpy.ndarray'> (2, 3) float64
```

```
c1 = b[1:,0:2]#note the slice indexing
print c1,c1.shape
c2= b[1,0:2] #note the integer indexing
print c2,c2.shape
```

```
[[ 1.  2.]] (1, 2)
[ 1.  2.] (2,)
```


Python Review: Numpy

```
print b>2, b[b>2]
```

```
[[False False  True]
 [False False  True]] [ 3.  3.]
```

```
x = numpy.array([[1,2],[3,4]])
y = numpy.array([[1,1],[1,1]])
z = numpy.array([1,1])
print x*y #elementwise product
print x.dot(z) #matrix vector product
```

```
[[1 2]
 [3 4]]
[3 7]
```

```
print x.sum(), x.T
```

```
10 [[1 3]
     [2 4]]
```

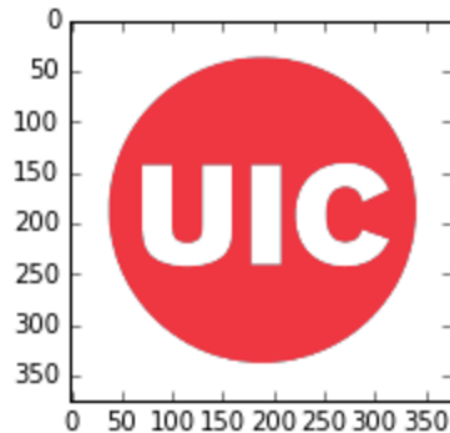

Python Review: Scipy Images

Scipy images

```
from scipy.misc import imread, imresize
%matplotlib inline
import matplotlib.pyplot as plt

img = imread('uic-logo-circle-red.jpg')

# Show the original image
plt.subplot(1, 2, 1)
plt.imshow(numpy.uint8(img))
plt.show()
```



Some Relevant Packages in Python

- Keras
 - An open-source neural network library running on top of various deep learning frameworks.
- Tensorflow
 - A programming system to represent computations as graphs
 - Two steps:
 - Construct the graph
 - Execute (via session)

Questions?

Today's Outline

- Python Walkthrough
- Feedforward Neural Nets
- Convolutional Neural Nets
 - Convolution
 - Pooling

Feedforward Neural Network

- Linear model $f(x, W, b) = Wx + b$
- A feedforward neural network model will include nonlinearities
- Two layer model
 - $f(x, W_1, b_1, W_2, b_2) = W_2 \max(0, W_1 x + b_1) + b_2$
 - Say x is d dimensional
 - W_1 is $d \times q$ dimensional
 - W_2 is $q \times p$ dimensional
 - Then the number of hidden nodes is q
 - The number of labels is p
 - The notion of layer is for vectorizing/is conceptual

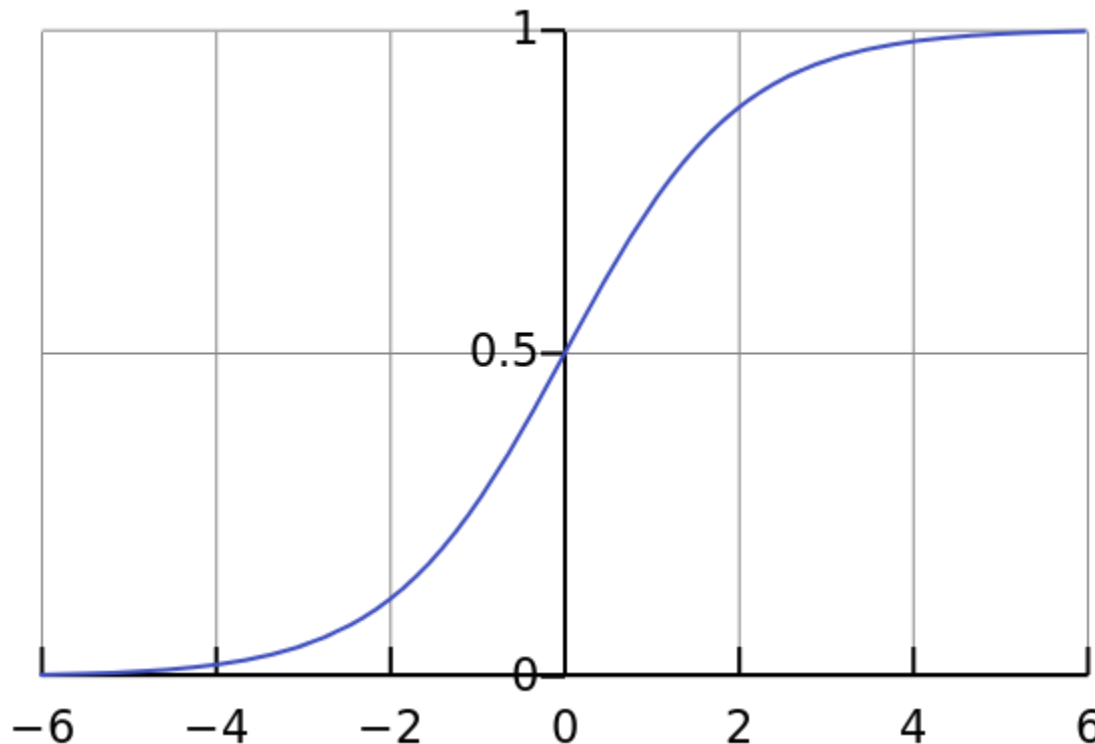
Nonlinearities (I)

Name	Formula	Year
none	$y = x$	-
sigmoid	$y = \frac{1}{1+e^{-x}}$	1986
tanh	$y = \frac{e^{2x}-1}{e^{2x}+1}$	1986
ReLU	$y = \max(x, 0)$	2010
(centered) SoftPlus	$y = \ln(e^x + 1) - \ln 2$	2011
LReLU	$y = \max(x, \alpha x), \alpha \approx 0.01$	2011
maxout	$y = \max(W_1x + b_1, W_2x + b_2)$	2013
APL	$y = \max(x, 0) + \sum_{s=1}^S a_i^s \max(0, -x + b_i^s)$	2014
VReLU	$y = \max(x, \alpha x), \alpha \in 0.1, 0.5$	2014
RReLU	$y = \max(x, \alpha x), \alpha = \text{random}(0.1, 0.5)$	2015
PRReLU	$y = \max(x, \alpha x), \alpha \text{ is learnable}$	2015
ELU	$y = x, \text{ if } x \geq 0, \text{ else } \alpha(e^x - 1)$	2015

- How to pick the nonlinearity/**activation function?**

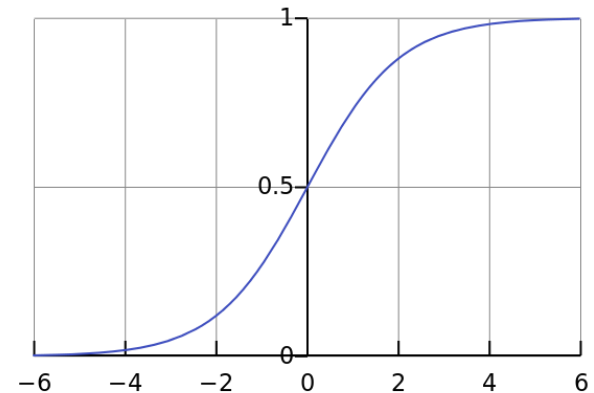
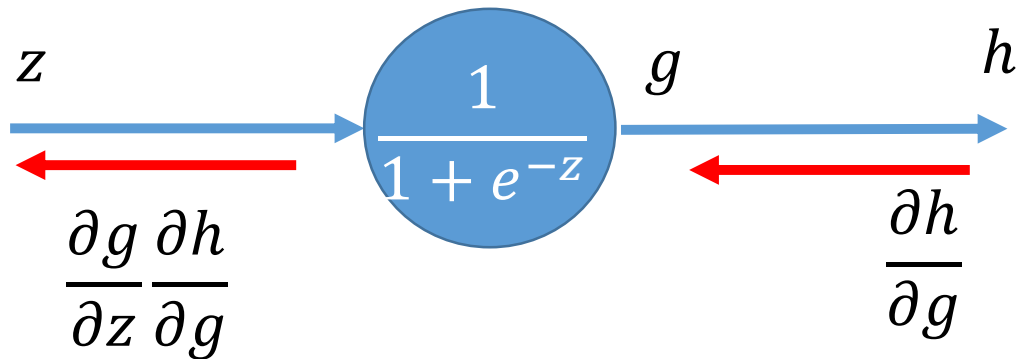
Nonlinearities (II)

- Sigmoid
 - Is a map whose range is $[0,1]$



Nonlinearities (III)

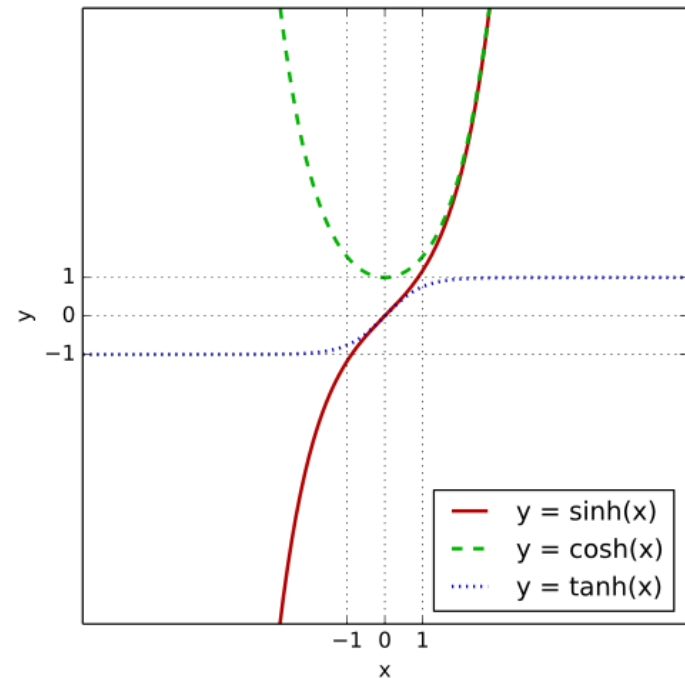
- **Saturated** node/neuron makes gradients vanish



- Not zero-centered
 - Empirically may lead to slower convergence

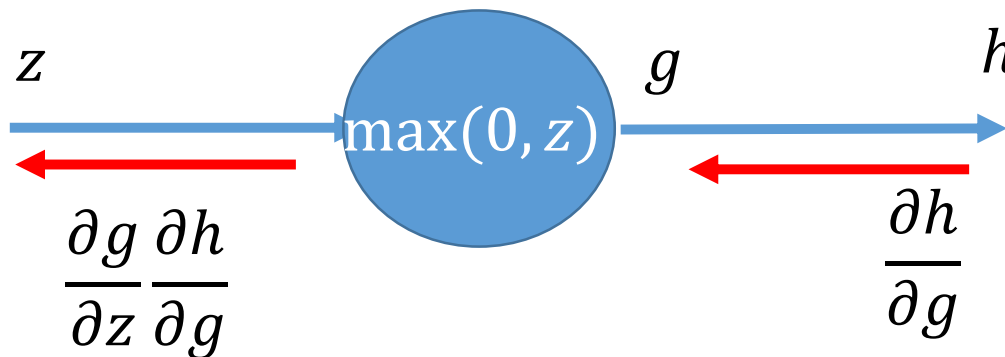
Nonlinearities (IV)

- $\tanh()$ addresses the zero-centering problem. So will typically give better results
- Still gradients vanish

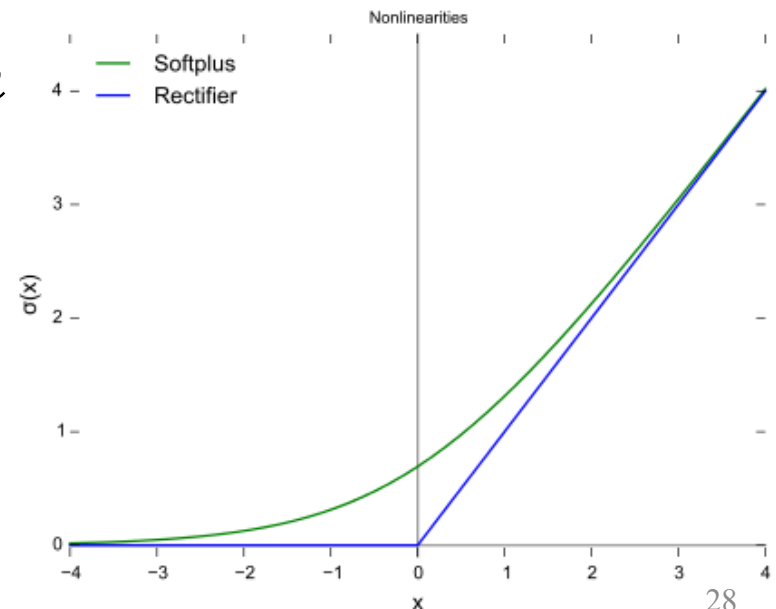


Nonlinearities (V)

- ReLU (2012 Krizhevsky et al.)
- No vanishing gradient on the positive side
- Empirically observed to be very good
- Initialization/high learning rate may lead to permanently dead ReLUs (diagnosable)

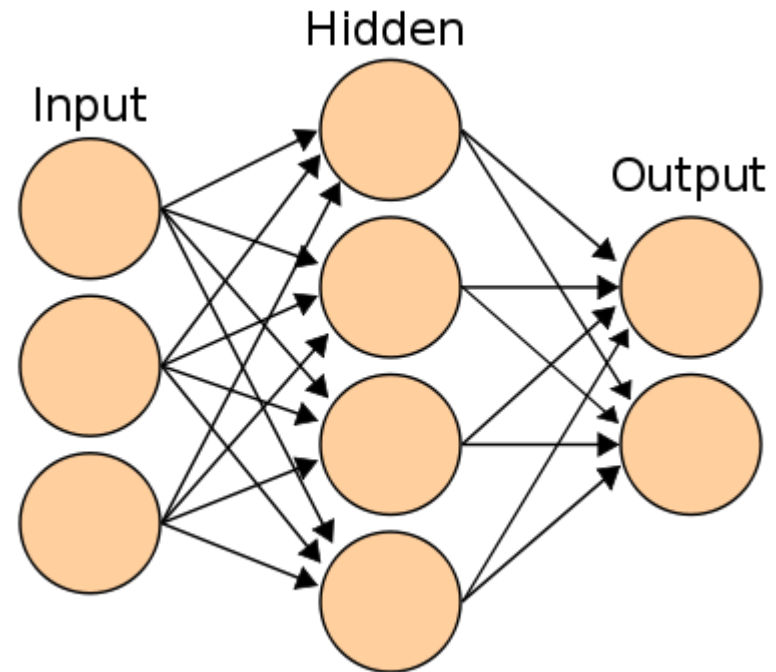


Is a gradient gate!



Feedforward Neural Net

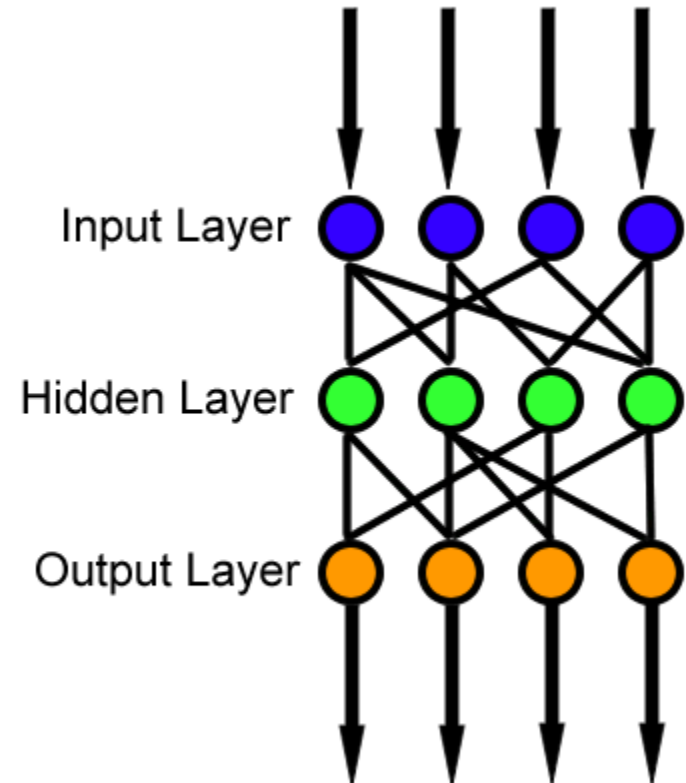
- Lets focus on a 2-layer net
- Layers
 - Input
 - Hidden
 - Output
- Node
- Nonlinearity
 - Activation



$$f(x, W_1, b_1, W_2, b_2) = W_2 \max(0, W_1 x + b_1) + b_2$$

Feedforward Net: Two Layer Model

- Number of layers is the number of W, b pairs
- Some questions to think about:
 - How to pick the number of layers?
 - How to pick the number of hidden units in each layer?



Feedforward Net and Backprop

- Choose a mini-batch (sample) of size B
- Forward propagate through the computation graph
 - Compute losses $L_{i_1}, L_{i_2}, \dots, L_{i_B}$ and $R(W_1, b_1, W_2, b_2)$
 - Get loss L for the batch
- Backprop to compute gradients with respect to W_1, b_1, W_2 and b_2
- Update parameters W_1, b_1, W_2 and b_2
 - In the direction of the negative gradient

Feedforward Net in Python

```
# Feedforward neural net model

# Start with an initial set of parameters randomly
h = 100 # size of hidden layer
W = 0.01 * np.random.randn(D,h)
b = np.zeros((1,h))
W2 = 0.01 * np.random.randn(h,K)
b2 = np.zeros((1,K))

# Initial values from hyperparameter
reg = 1e-3 # regularization strength

#For simplicity, we will not optimize this using grid search here.
```


Feedforward Net in Python

```
#Perform batch SGD using manual backprop

#For simplicity we will take the batch size to be the same as number of examples
num_examples = X.shape[0]

#Initial value for the Gradient Descent Parameter
step_size = 1e-0 #Also called learning rate

#For simplicity, we will not hand tune this algorithm parameter as well.

# gradient descent loop
for i in xrange(10000):

    # evaluate class scores, [N x K]
    hidden_layer = np.maximum(0, np.dot(X, W) + b) # note, ReLU activation
    scores = np.dot(hidden_layer, W2) + b2

    # compute the class probabilities
    exp_scores = np.exp(scores)
    probs = exp_scores / np.sum(exp_scores, axis=1, keepdims=True) # [N x K]

    # compute the loss: average cross-entropy loss and regularization
    corect_logprobs = -np.log(probs[range(num_examples), y])
    data_loss = np.sum(corect_logprobs)/num_examples
    reg_loss = 0.5*reg*np.sum(W*W) + 0.5*reg*np.sum(W2*W2)
    loss = data_loss + reg_loss
    if i % 1000 == 0:
        print "iteration %d: loss %f" % (i, loss)
```


Feedforward Net in Python

```
# compute the gradient on scores
dscores = probs
dscores[range(num_examples),y] -= 1
dscores /= num_examples

# backpropate the gradient to the parameters
# first backprop into parameters W2 and b2
dW2 = np.dot(hidden_layer.T, dscores)
db2 = np.sum(dscores, axis=0, keepdims=True)
# next backprop into hidden layer
dhidden = np.dot(dscores, W2.T)
# backprop the ReLU non-linearity
dhidden[hidden_layer <= 0] = 0
# finally into W,b
dW = np.dot(X.T, dhidden)
db = np.sum(dhidden, axis=0, keepdims=True)

# add regularization gradient contribution
dW2 += reg * W2
dW += reg * W

# perform a parameter update
W += -step_size * dW
b += -step_size * db
W2 += -step_size * dW2
b2 += -step_size * db2
```


Feedforward Net in Python

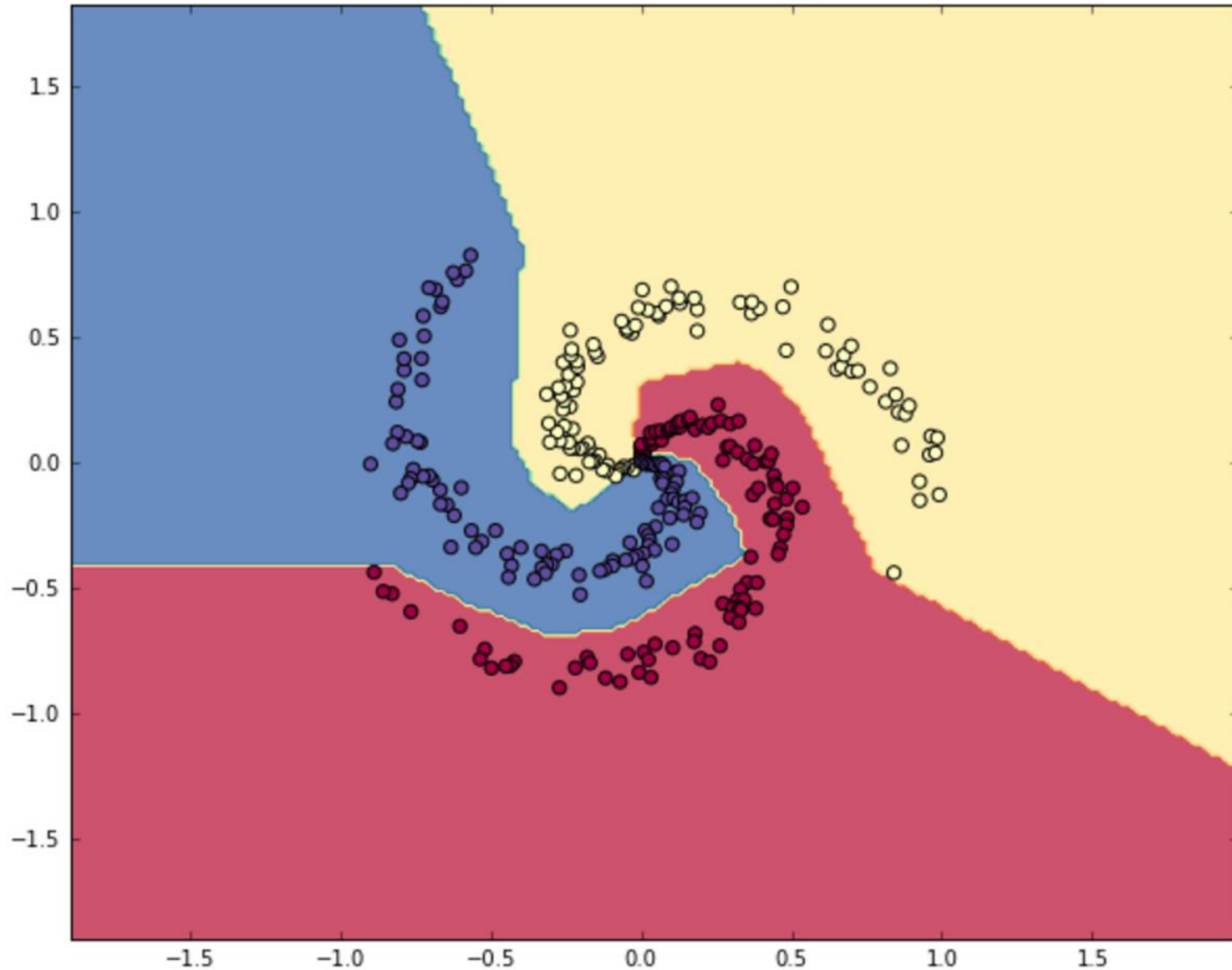
Post Training

```
# Post-training: evaluate test set accuracy

#For simplicity, we will use training data as proxy for test. Do not do this.
X_test = X
y_test = y

hidden_layer = np.maximum(0, np.dot(X_test, W) + b)
scores = np.dot(hidden_layer, W2) + b2
predicted_class = np.argmax(scores, axis=1)
print 'test accuracy: %.2f' % (np.mean(predicted_class == y_test))
```


Feedforward Net in Python



FNN in the Browser

- See playground.tensorflow.org

Questions?

Today's Outline

- Python Walkthrough
- Feedforward Neural Nets
- Convolutional Neural Nets
 - Convolution
 - Pooling

Convolutional Neural Network

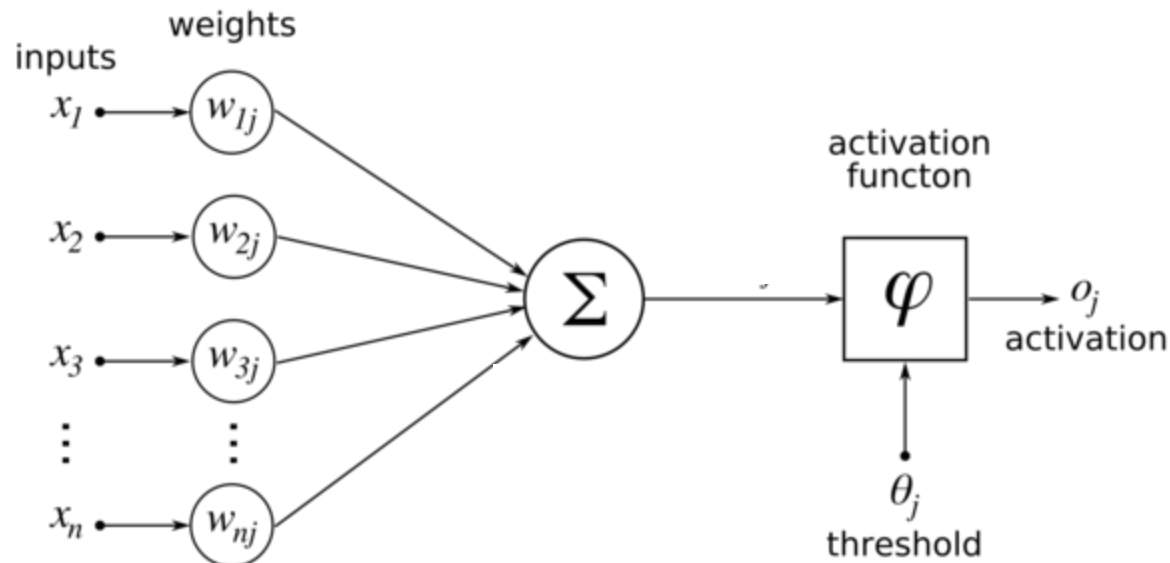
Similar to Feedforward NN

- Similar to feedforward neural networks
- Each neuron/node is associated with weights and a bias
- Node receives input
 - Performs dot product of vectors
 - Applies non-linearity
- The difference:
 - Number of parameters is reduced!

How? That is the content of this lecture!

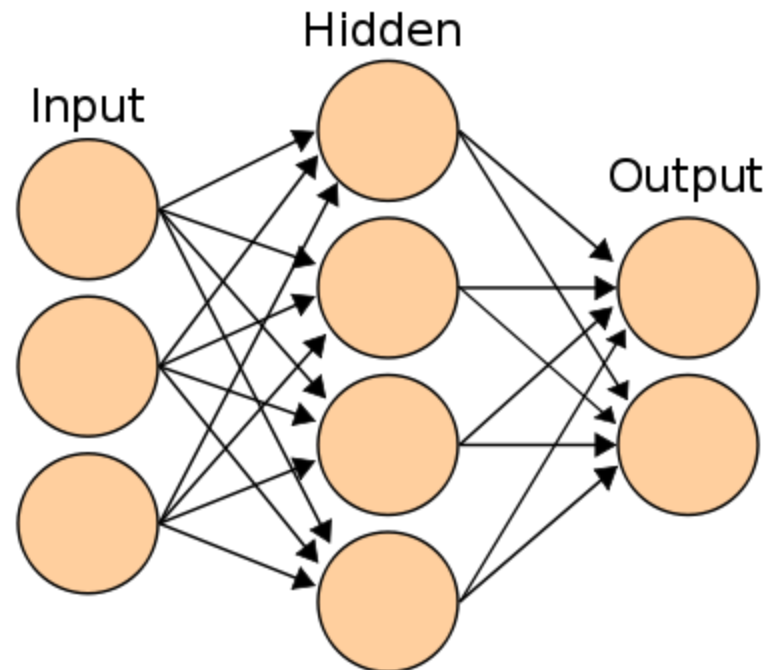
Similar to Feedforward NN

- Recall a Feedforward net:
 - Get a vector x_i and transform it to a score vector by passing through a sequence of hidden layers
 - Each hidden layer has neurons
 - Each neuron is fully connected to previous layer



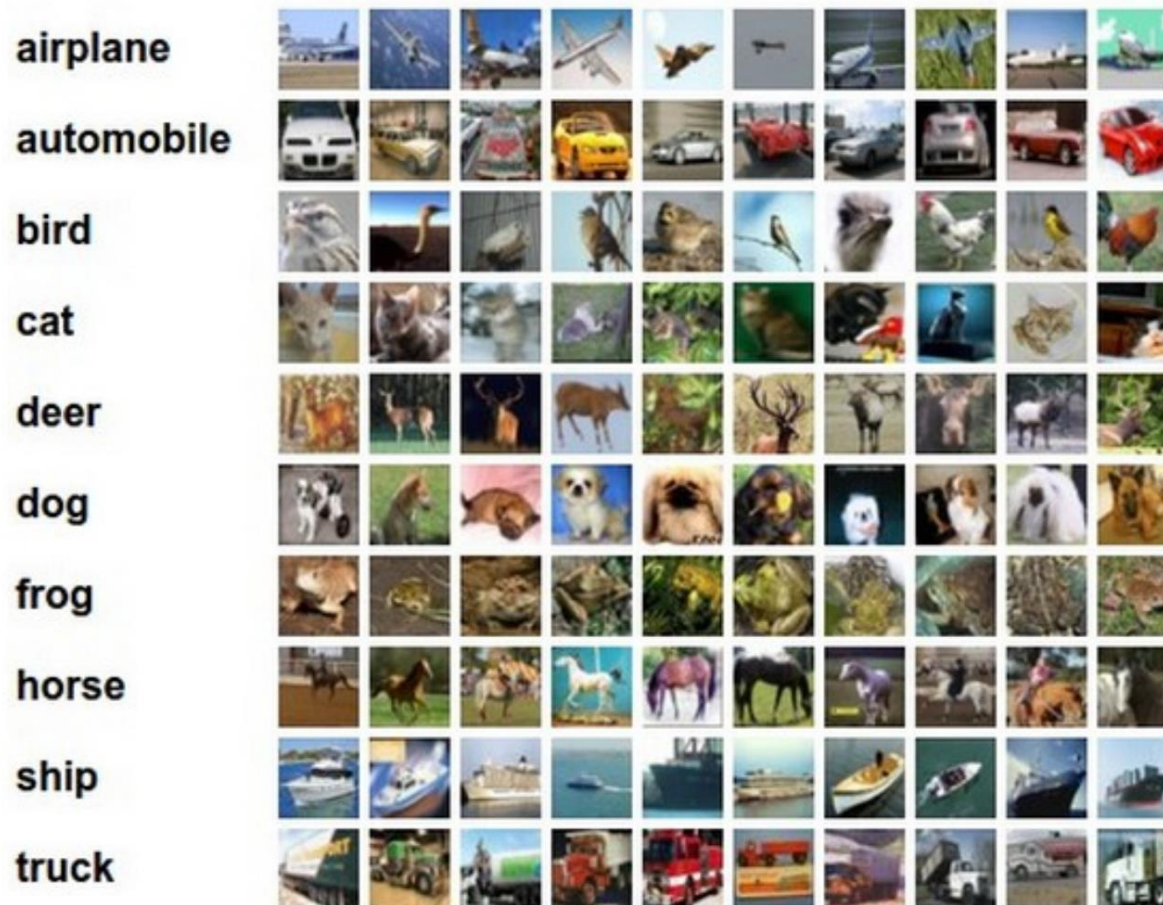
Towards CNNs (I)

- Feedforward net:
 - Can you visualize the connections for an arbitrary neuron here?



Towards CNNs (II)

- Consider the CIFAR-10 Dataset. Images are $32 \times 32 \times 3$ in size



¹Figure: <http://cs231n.github.io/classification/>

Towards CNNs (III)

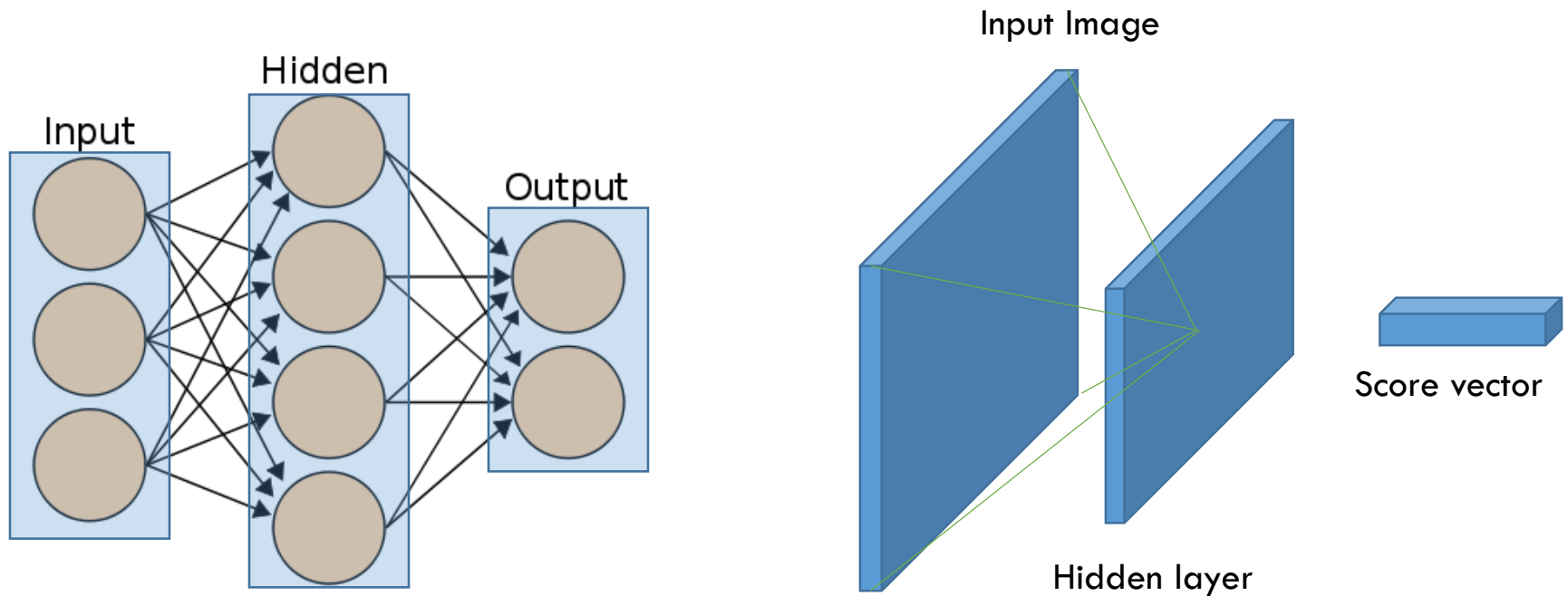
- First fully connected feedforward neuron would have $32*32*3$ weights associated with it (+1 bias parameter)
- What if the images were $1280*800*3$?
- Clearly, we also need many neurons in each hidden layer. This leads to explosion in the total number of parameters (or the dimension of W s and b s)

CNN Architecture

- We will look at it from layers point of view
- The new idea is that layers have **width** and **depth**!
 - (In contrast, Feedforward NN layers only had height)
 - (depth here does NOT correspond to number of layers of a network)

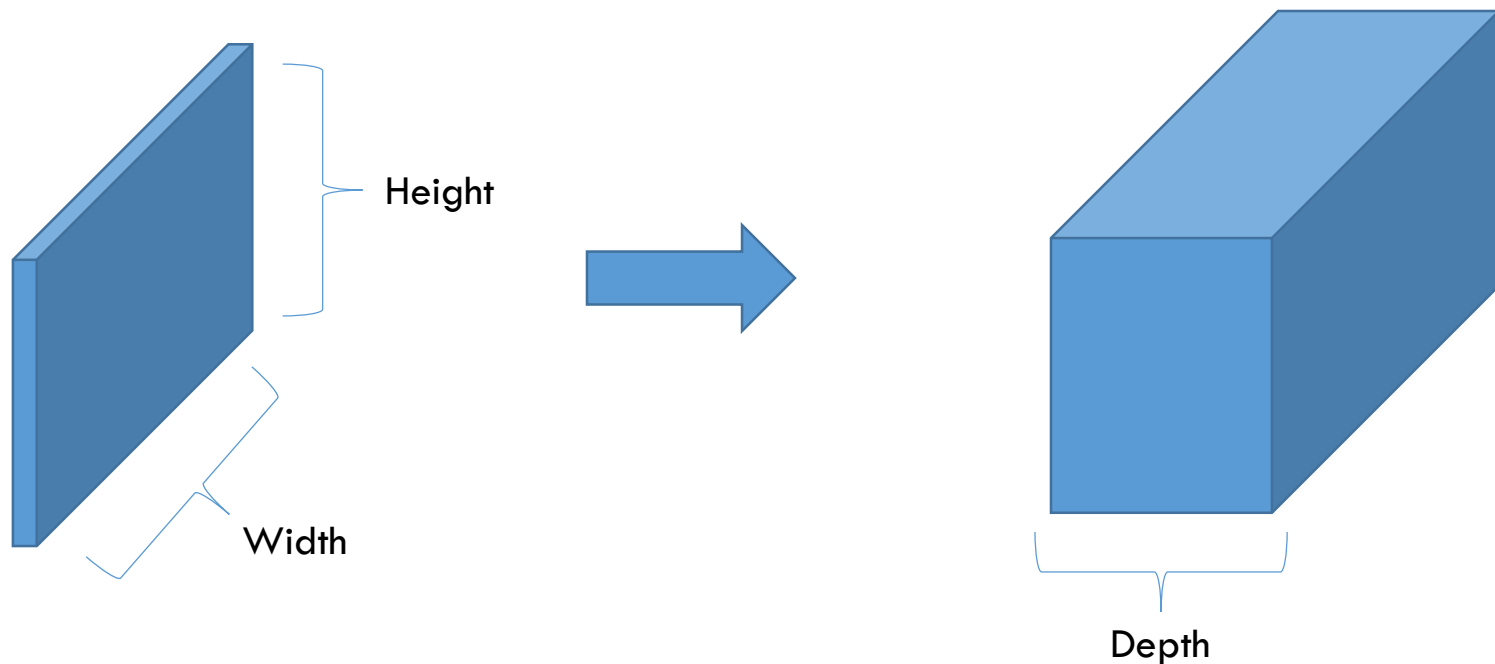
CNN Architecture

- View FFN layers as having width and height



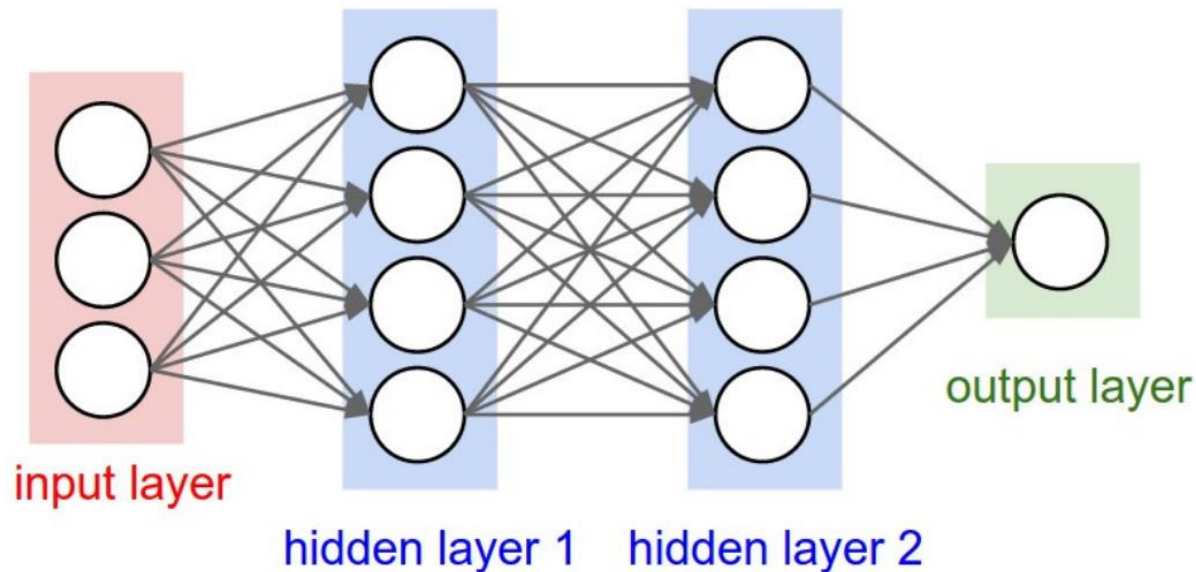
CNN Architecture

- The new idea is that CNN layers have **depth**!
 - (depth here does NOT correspond to number of layers of a network)



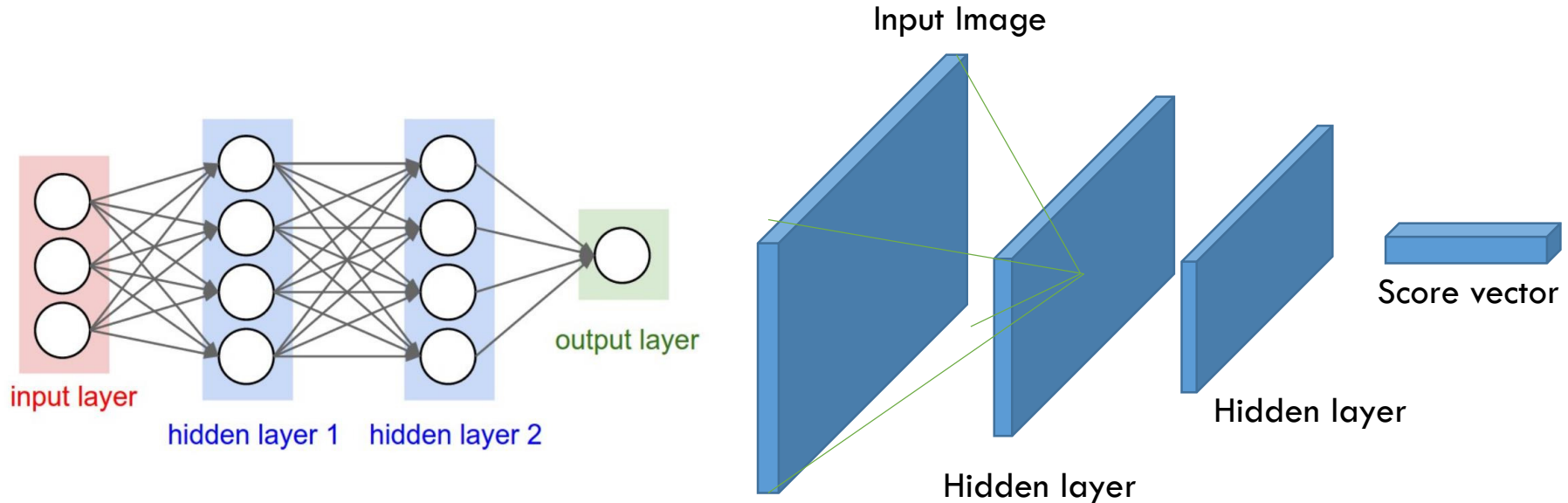
3D Volumes of Neurons

- Input has dimension $32*32*3$ (for CIFAR-10 dataset)
- Final output has dimension $1*1*10$ (10 classes)
- Previously,



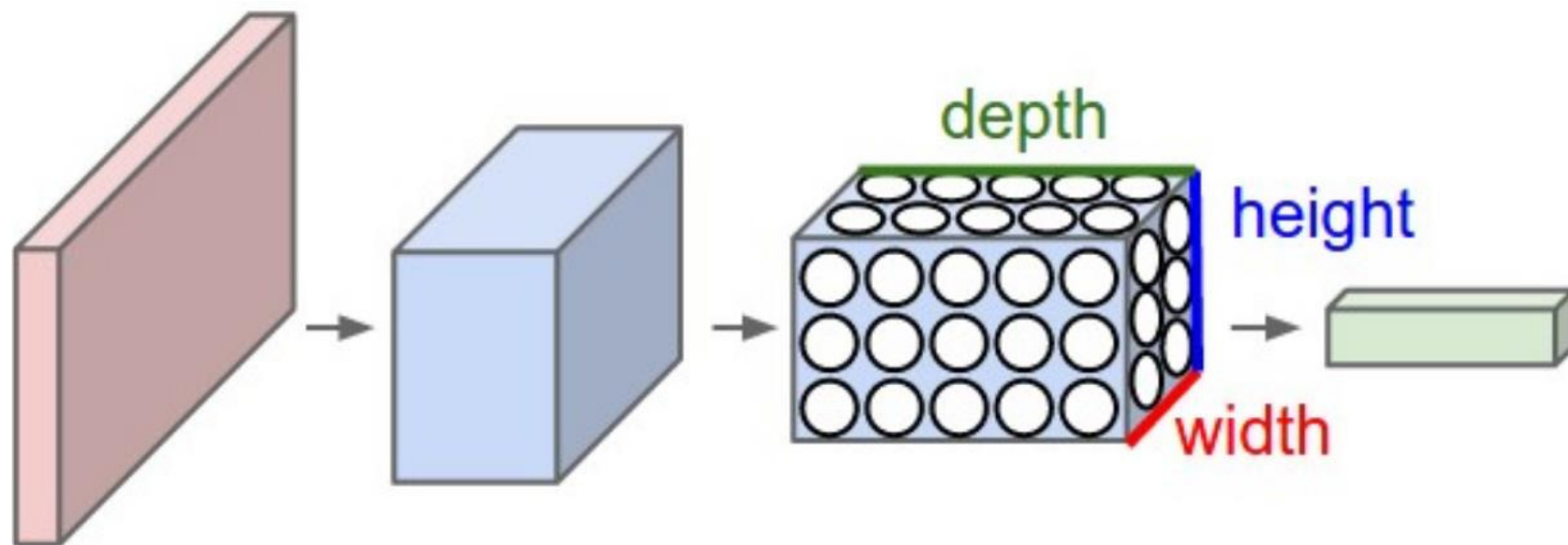
3D Volumes of Neurons

- Input has dimension $32*32*3$ (for CIFAR-10 dataset)
- Final output has dimension $1*1*10$ (10 classes)
- So assuming 2 hidden layers, previously we had,



3D Volumes of Neurons

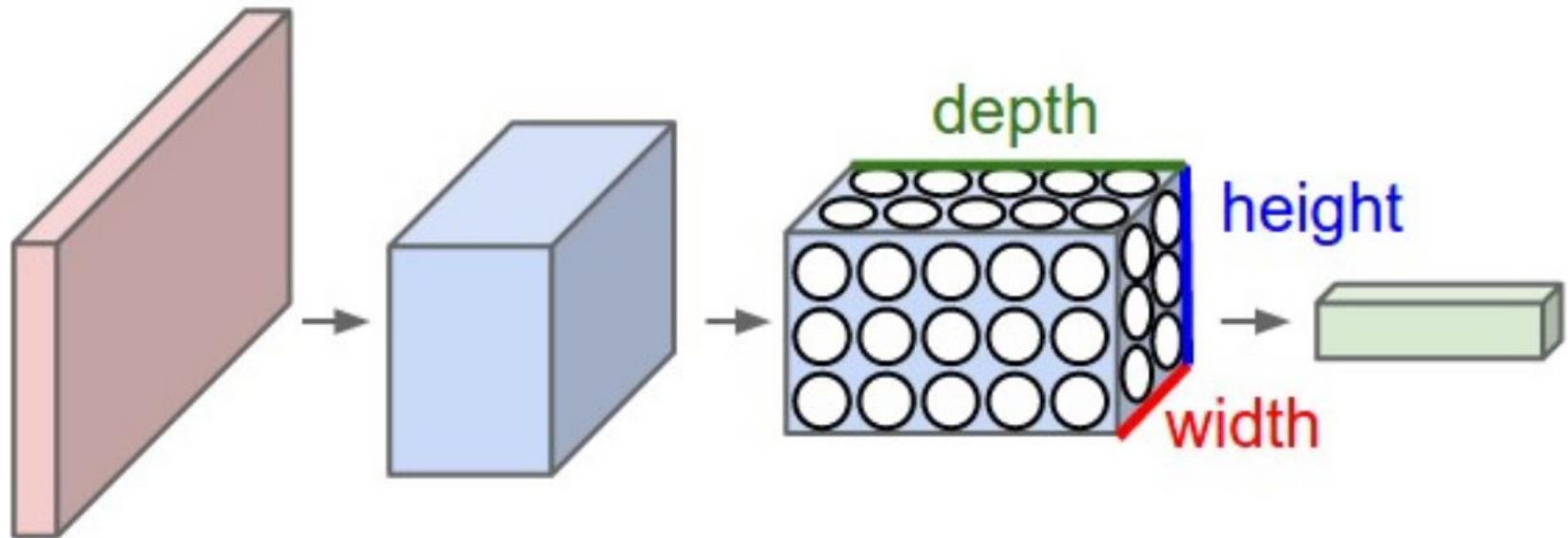
- Now,



- Each layer simply does this: transforms an input tensor (3D volume) to an output tensor using some function

3D Volumes of Neurons

- Now,



- Each layer simply does this: transforms an input tensor (3D volume) to an output tensor using some function

CNN Layers

- Three types
 - Convolutional Layer (CONV)
 - Pooling Layer (POOL)
 - Fully Connected Layer (same as Feedforward neural network, i.e., $1 * 1 * \text{\#Neurons}$ is the layer's output tensor)
- Stack these in various ways

CNN Example Architecture

- Say our classification dataset is CIFAR-10
- Let the architecture be as follows:
 - INPUT \rightarrow CONV \rightarrow POOL \rightarrow FC
- INPUT:
 - This layer is nothing but $32 \times 32 \times 3$ in dimension (width*height*3 color channels)

CNN Example Architecture

- Say our classification dataset is CIFAR-10
- Let the architecture be as follows:
 - INPUT \rightarrow CONV \rightarrow POOL \rightarrow FC
- INPUT:
 - This layer is nothing but $32 \times 32 \times 3$ in dimension (width*height*3 color channels)
- CONV:
 - Neurons compute like regular feedforward neurons (sum the product of inputs with weights and add bias).
 - May output a different shaped tensor, say with dimension $32 \times 32 \times 12$

CNN Example Architecture

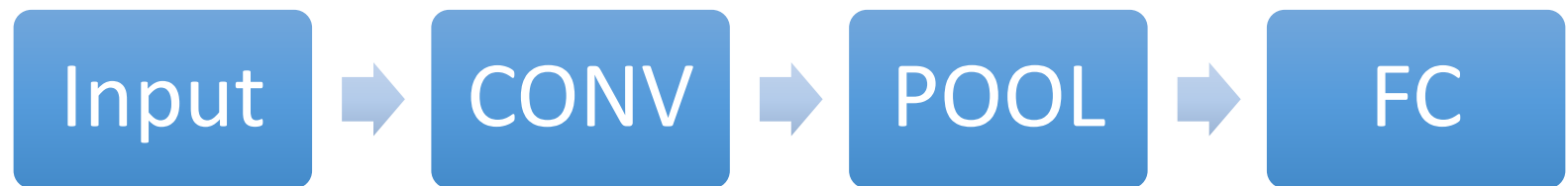
- POOL:
 - Performs a down-sampling in the spatial dimension
 - Outputs a tensor with the depth dimension the same as input
 - If input is $32*32*12$, then output could be $16*16*12$

CNN Example Architecture

- POOL:
 - Performs a down-sampling in the spatial dimension
 - Outputs a tensor with the depth dimension the same as input
 - If input is $32*32*12$, then output could be $16*16*12$
- FC:
 - This is the fully connected layer. Input can be any tensor (say $16*16*12$) but the output will have only one effective dimension ($1*1*10$ since this is the last layer and CIFAR-10 has 10 classes)

CNN Example Architecture

- So we went from pixels (32×32 RGB images) to scores (10 in number)



- Some layers have parameters (CONV and FC), other layers do not (POOL)
- Optimization of these parameters still for achieving scores consistent with image labels

The Convolution Layer (CONV)

- Layer's parameters correspond to a set of filters
- What is a filter?
 - A linear function parameterized by a tensor
 - Outputs a scalar
 - The parameter tensor is learned during training
- Example
 - First layer filter may be of dimension $3 \times 3 \times 3$
 - 3 pixels wide
 - 3 pixels high
 - 3 unit filter-depth for three color channels
- We slide (convolve) the filter across the width and height of the input volume and compute the scalar output to be passed into the nonlinearity

CONV: Sliding/Convolving

- We slide (convolve) the filter across the width and height of the input volume and compute the scalar output to be passed into the nonlinearity

1 _{x1}	1 _{x0}	1 _{x1}	0	0
0 _{x0}	1 _{x1}	1 _{x0}	1	0
0 _{x1}	0 _{x0}	1 _{x1}	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

Convolved
Feature

Also see <http://setosa.io/ev/image-kernels/>

The Convolution Layer (CONV)

- Three things to notice
 - Filters are small along width and height
 - Same **filter-depth** as the input tensor (3D volume)
 - If the input is $x * y * z$, then filter could be $3 * 3 * z$
 - As we slide, we produce a **2D** activation map

The Convolution Layer (CONV)

- Three things to notice
 - Filters are small along width and height
 - Same **filter-depth** as the input tensor (3D volume)
 - If the input is $x * y * z$, then filter could be $3 * 3 * z$
 - As we slide, we produce a **2D** activation map
- Filters (i.e., filter parameters) will be learned during training that ‘detect’ certain visual features
 - Example:
 - Oriented edges, colors, etc. at the first layer
 - Specific patterns in higher layers

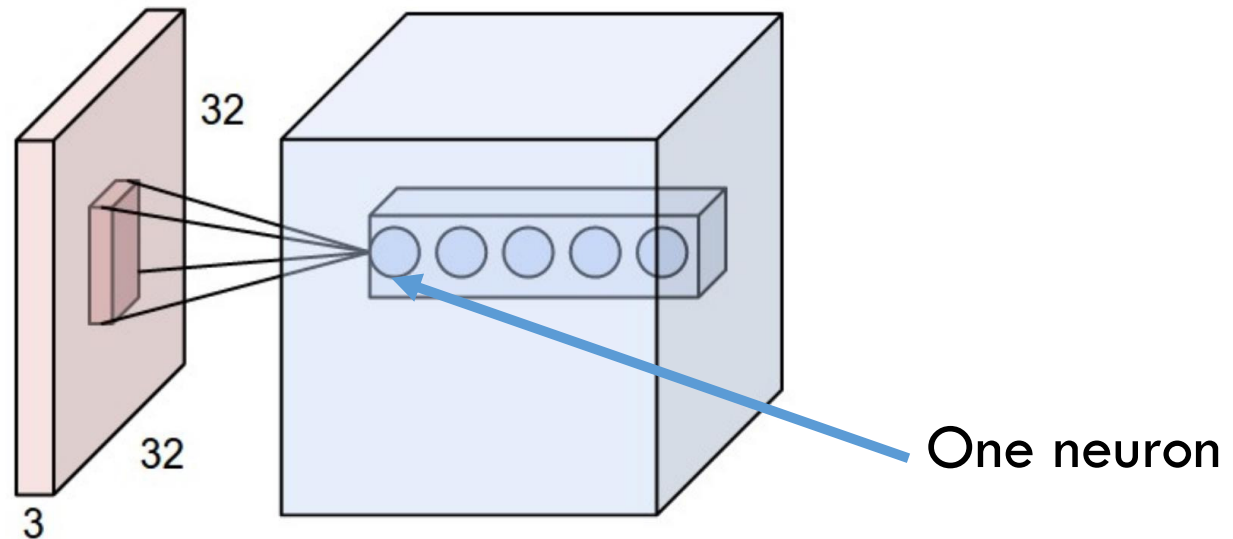
CONV: Filters

- Before we look at the patterns ...
- Lets now look at the neurons themselves
 - How are they connected?
 - How are they arranged?
 - How can we get **reduced parameters**?

CONV: Local Connectivity

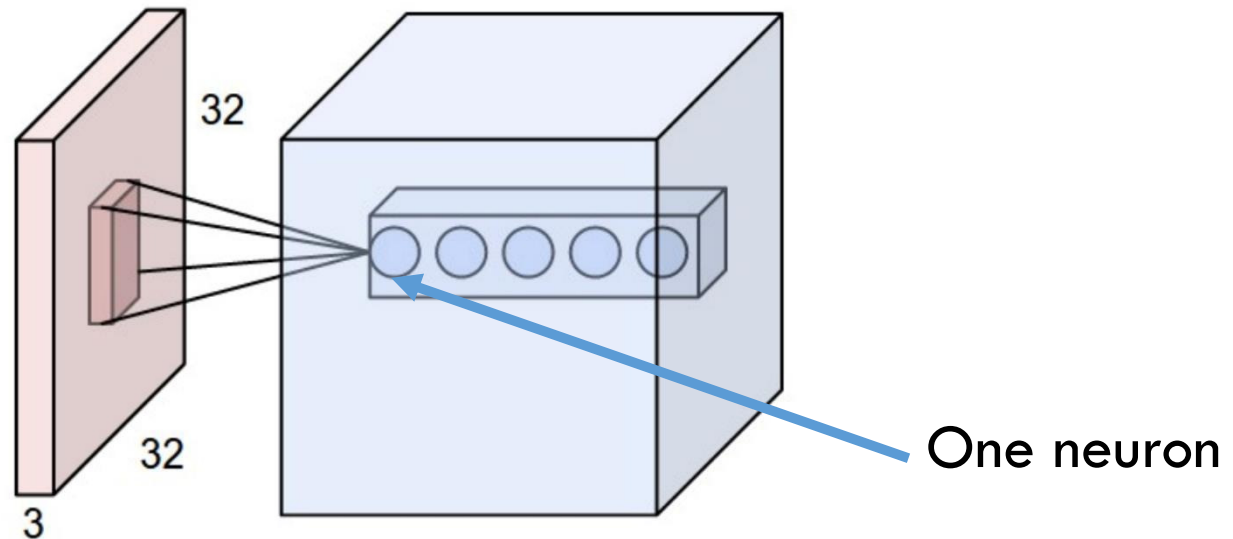
- Connect each neuron to a local (spatial) region of the input tensor
- Spatial extent of this connectivity is called **receptive field**
- Depth connectivity is the same as input depth

CONV: Local Connectivity



- Example: If input tensor is $32 \times 32 \times 3$ and filter is $3 \times 3 \times 3$ then
 - the number of weight parameters is 27, and
 - there is 1 bias parameter

CONV: Local Connectivity

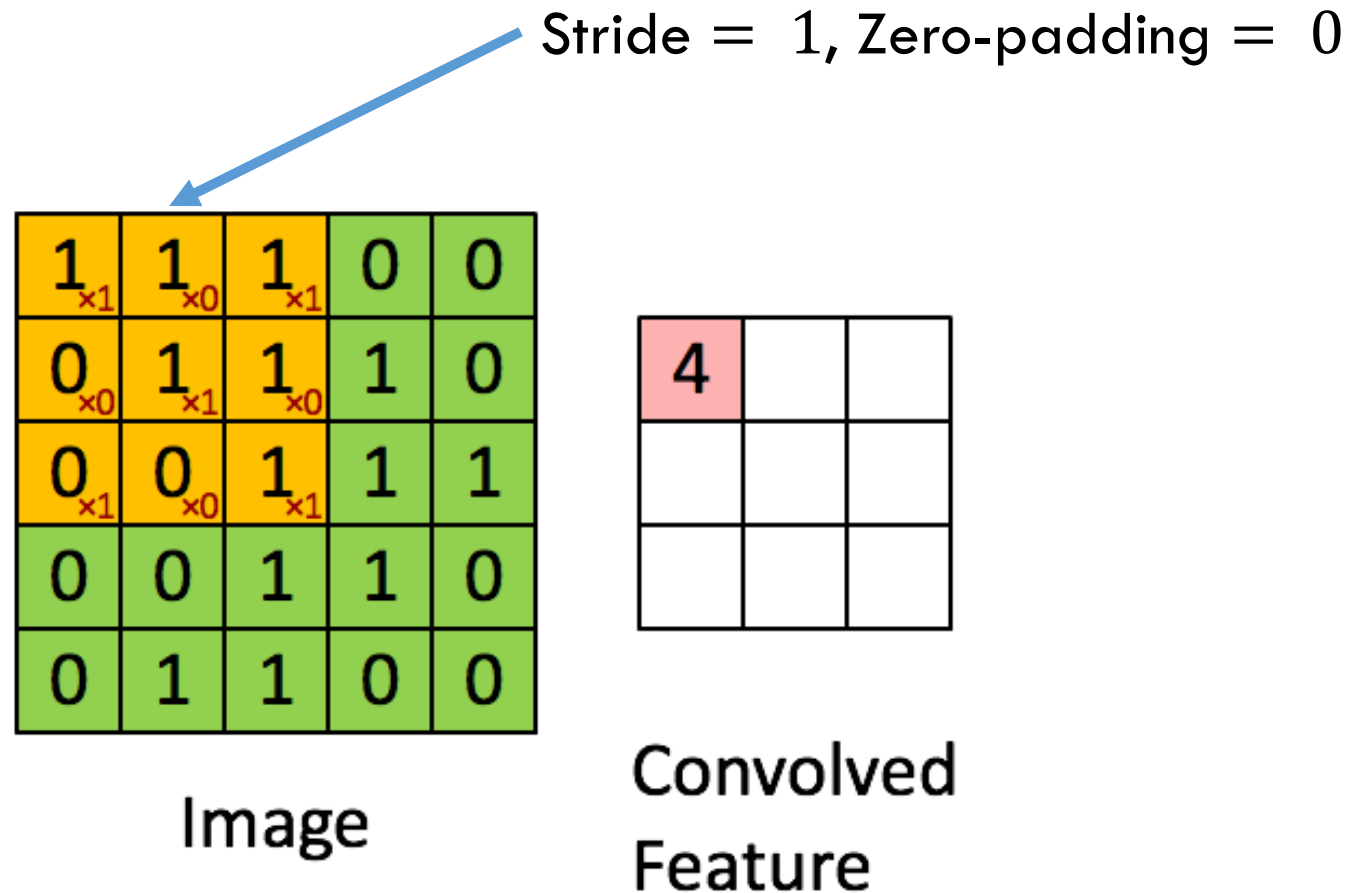


- All 5 neurons are looking at the same spatial region
- Each neuron belongs to a different filter

CONV: Spatial Arrangement

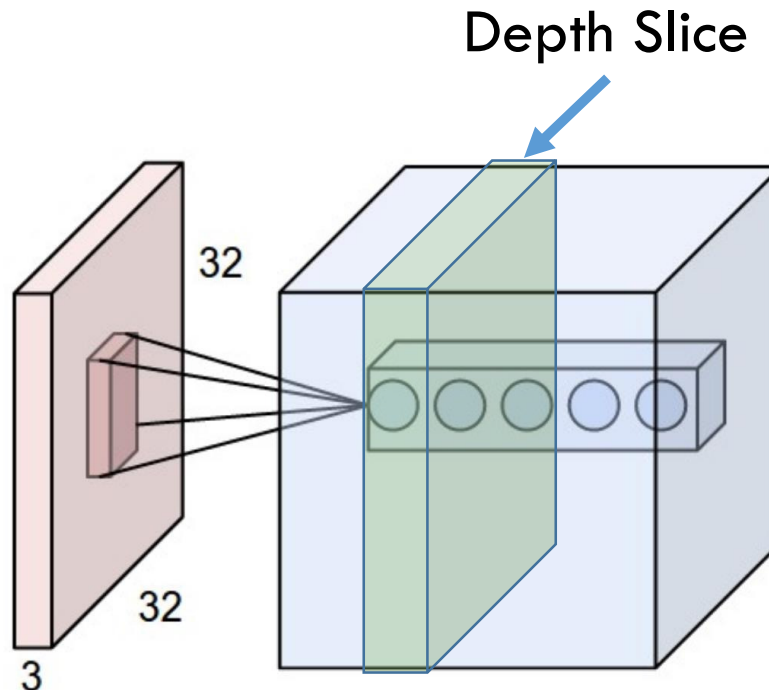
- Back to layer point of view
- Size of output **tensor** depends on three numbers:
 - **Layer Depth**
 - Corresponds to the number of filters
 - **Stride** (how much the filter is moved spatial)
 - Example: If stride is 1, then filter is moved 1 pixel at a time
 - **Zero-padding**
 - Deals with boundaries (is usually 1 or 2)

CONV: Stride/Zero-pad



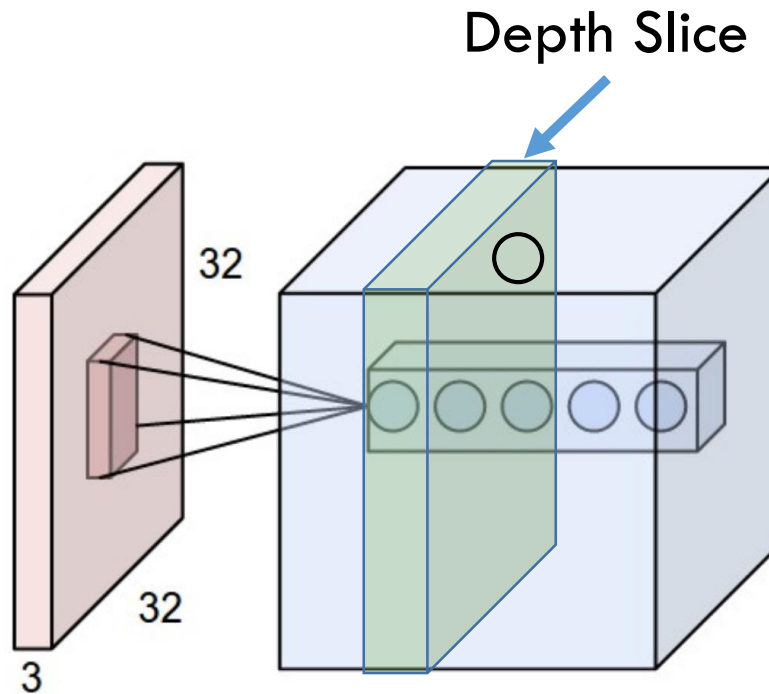
CONV: Parameter Sharing

- Key assumption: If a filter is useful for one region, it should also be useful for another region
- Denote a single 2D slice of depth of a layer as **depth slice**



CONV: Parameter Sharing

- Then, all neurons in each depth slice use the same weight and bias parameters!

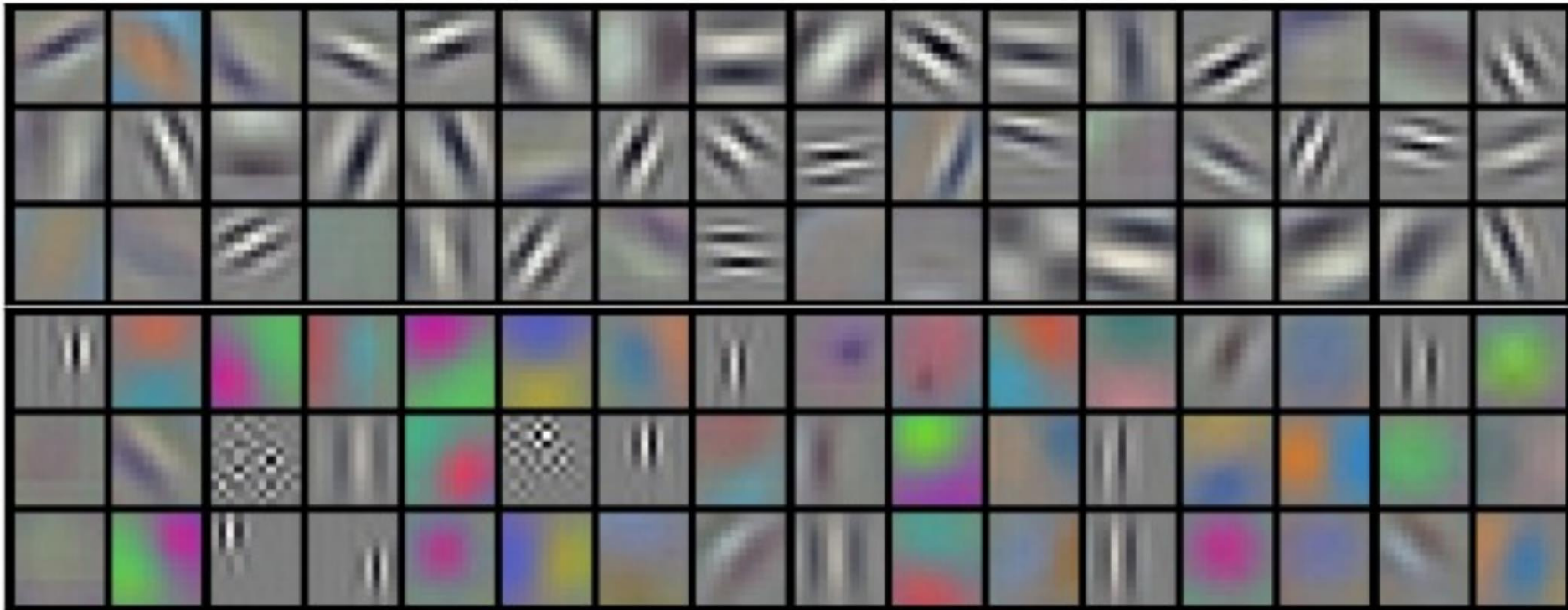


CONV: Parameter Sharing

- Number of parameters is reduced!
- Example:
 - Say the number of filters is M (= Layer Depth)
 - Then, this layer will have $M * (3 * 3 * 3 + 1)$ parameters
- Gradients will get added up across neurons of a depth slice

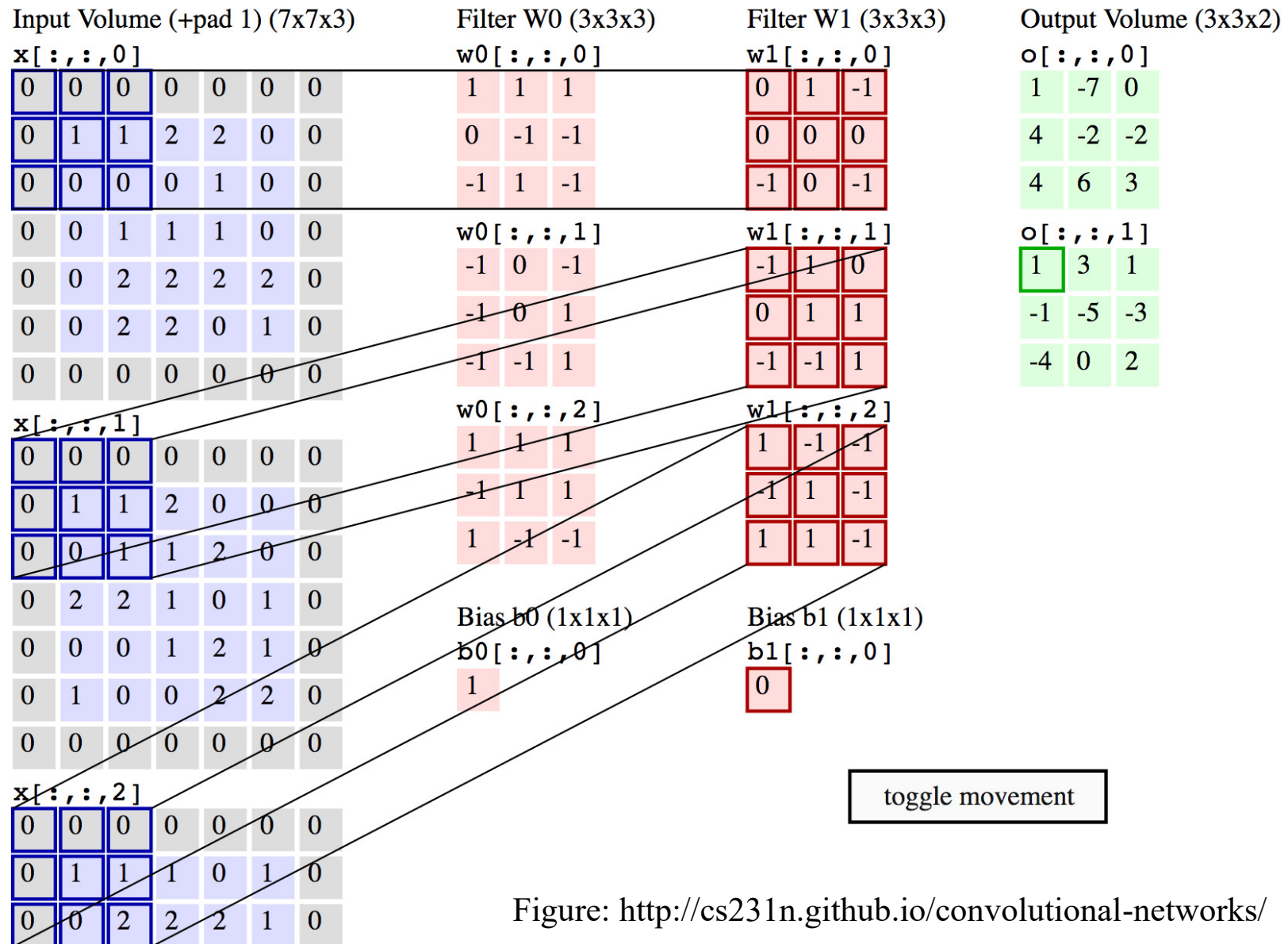
CONV: Parameter Sharing

- AlexNet's first layer has $11 \times 11 \times 3$ sized filters 96 in number. The filter weights are plotted below:



- Intuition: If capturing an edge is important, then important everywhere

Example: CONV Layer Computation

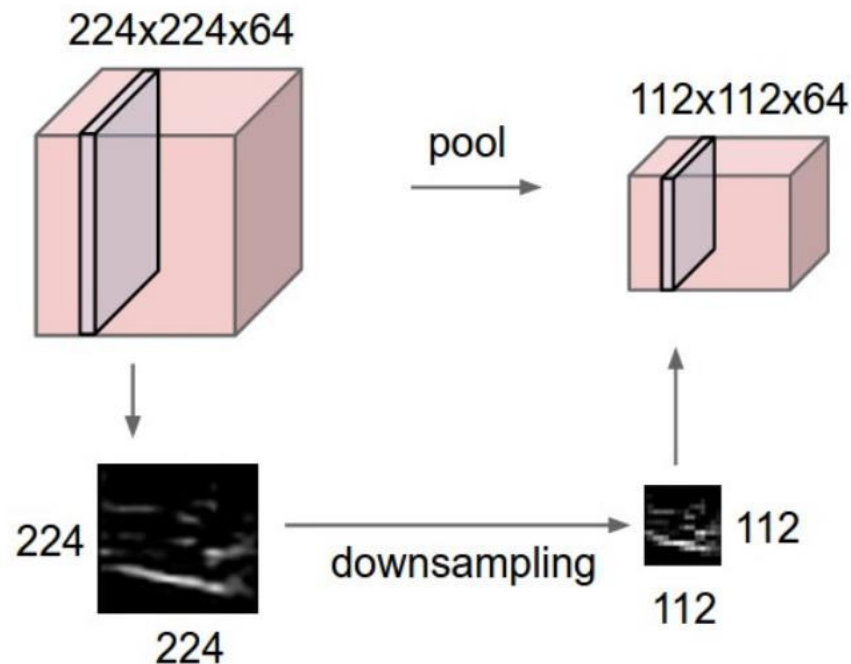


The Pooling Layer: POOL

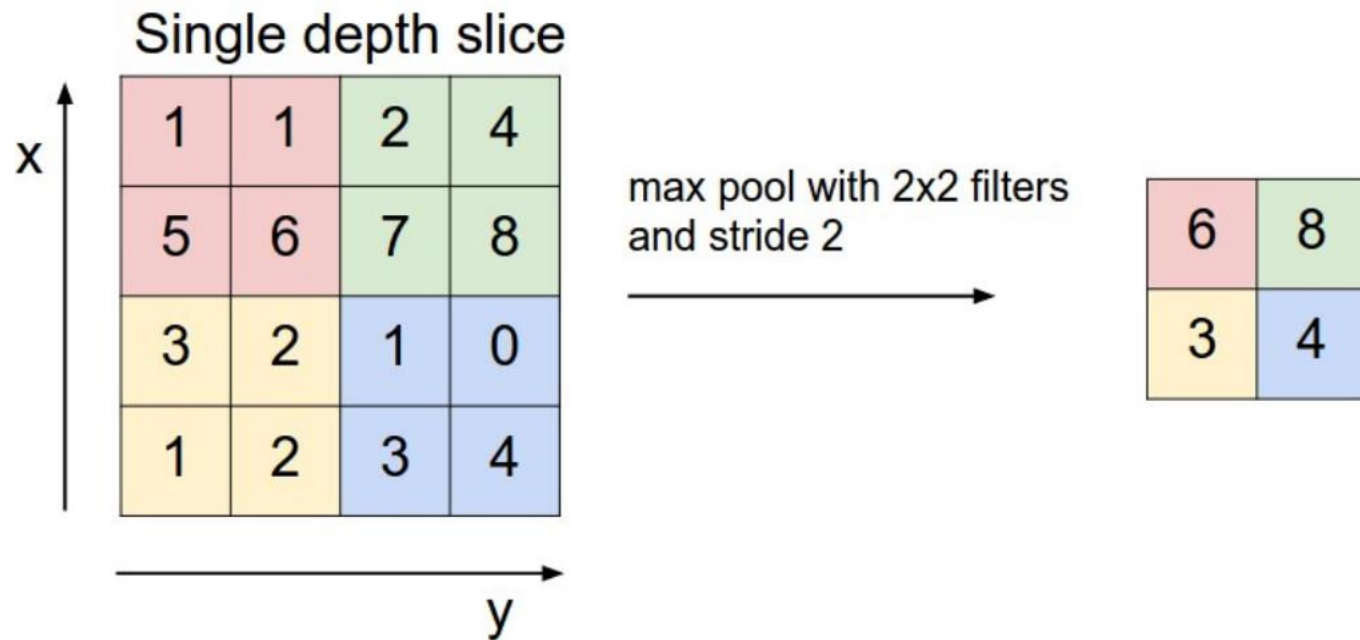
- Vastly more simpler than CONV
- Reduce the **spatial** size by using a MAX or similar operation
- Operate independently for each depth slice

POOL: Example

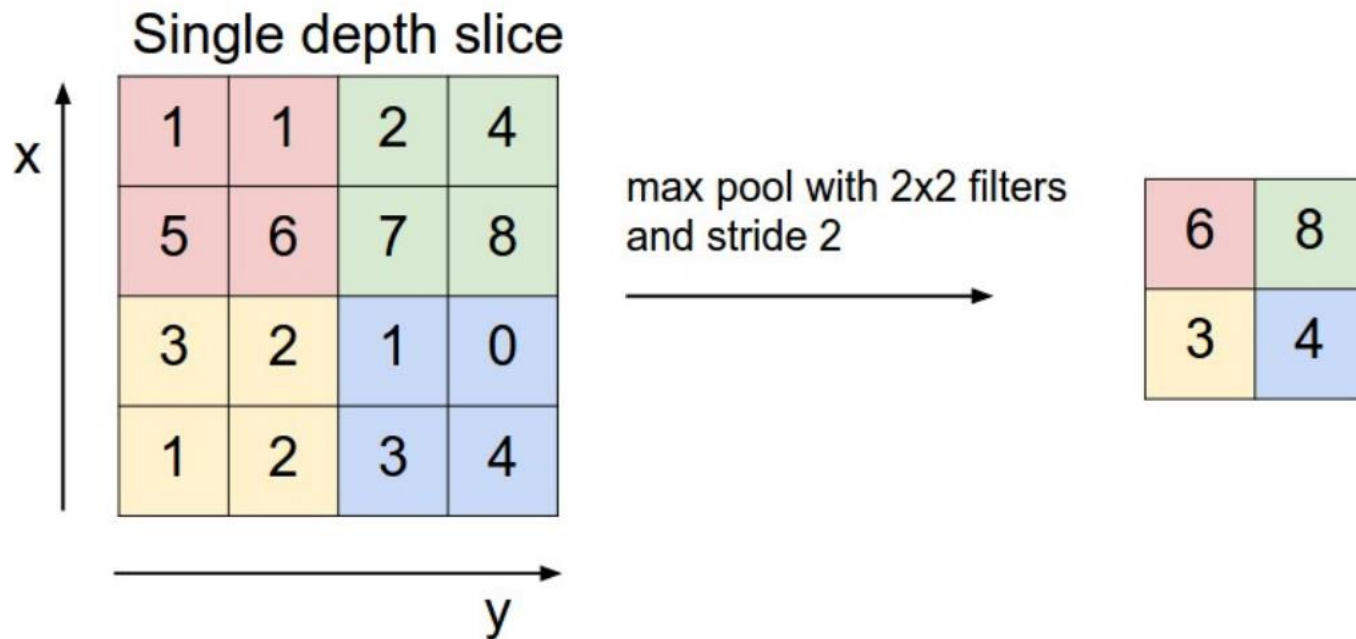
- Input depth is retained



POOL: Example



POOL: Example



- Recent research is showing that you may not need a pooling layer

Fully Connected Layer: FC

- Essentially a fully connected layer
- Already seen while discussing feedforward neural networks

CNN in the Browser

- Dataset: CIFAR-10
- <http://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html>

Summary

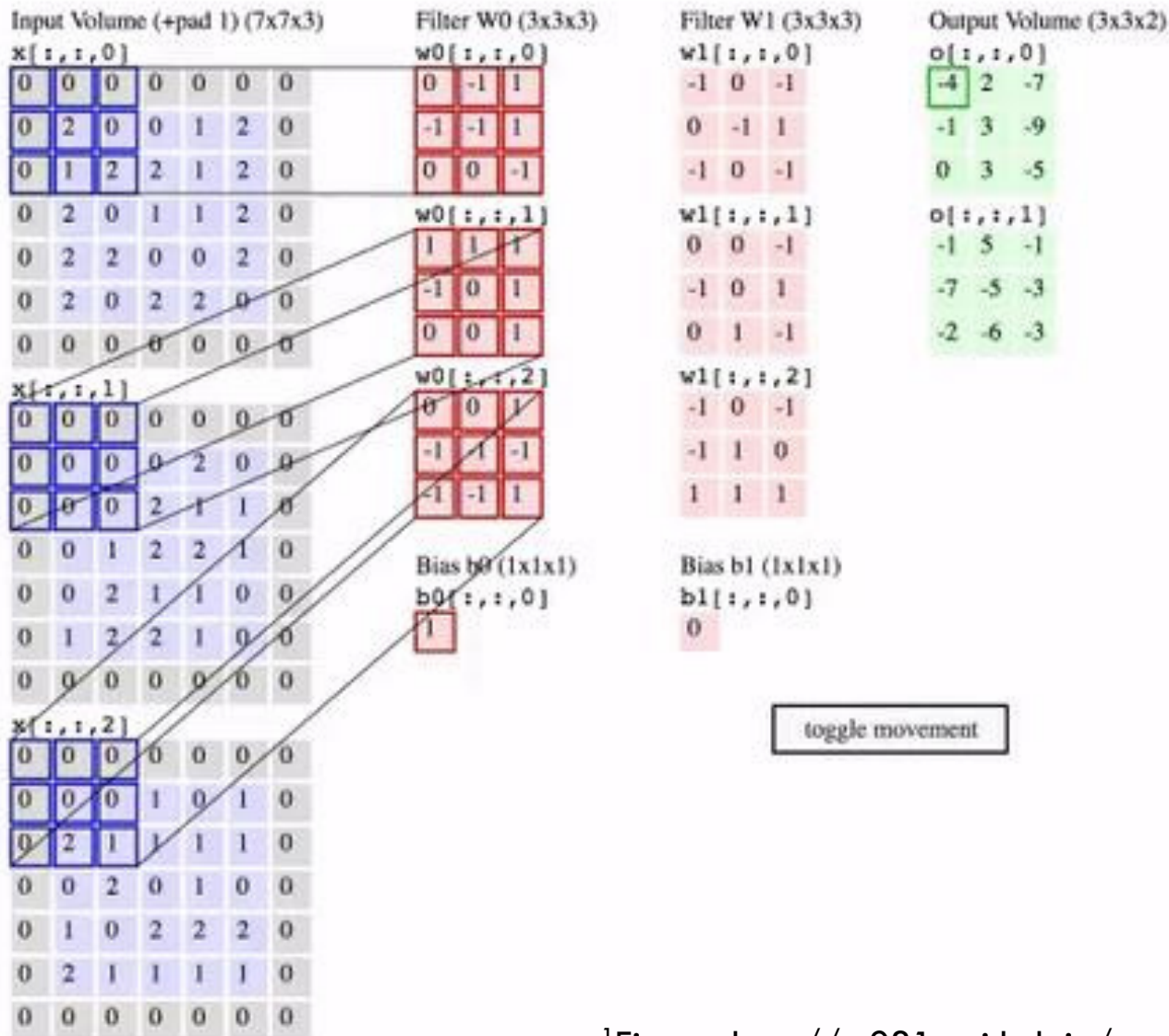
- Feedforward neural nets can do better than linear classifiers (saw this for a low-dimensional small synthetic example)
- CNN have been very effective in image related applications.
- Exploit specific properties of images
 - Hierarchy of features
 - Locality
 - Spatial invariance
- Lots of **design choices** that have been empirically validated and are intuitive. Still, there is room for improvement.

Appendix

Naming: Why 'Neural'

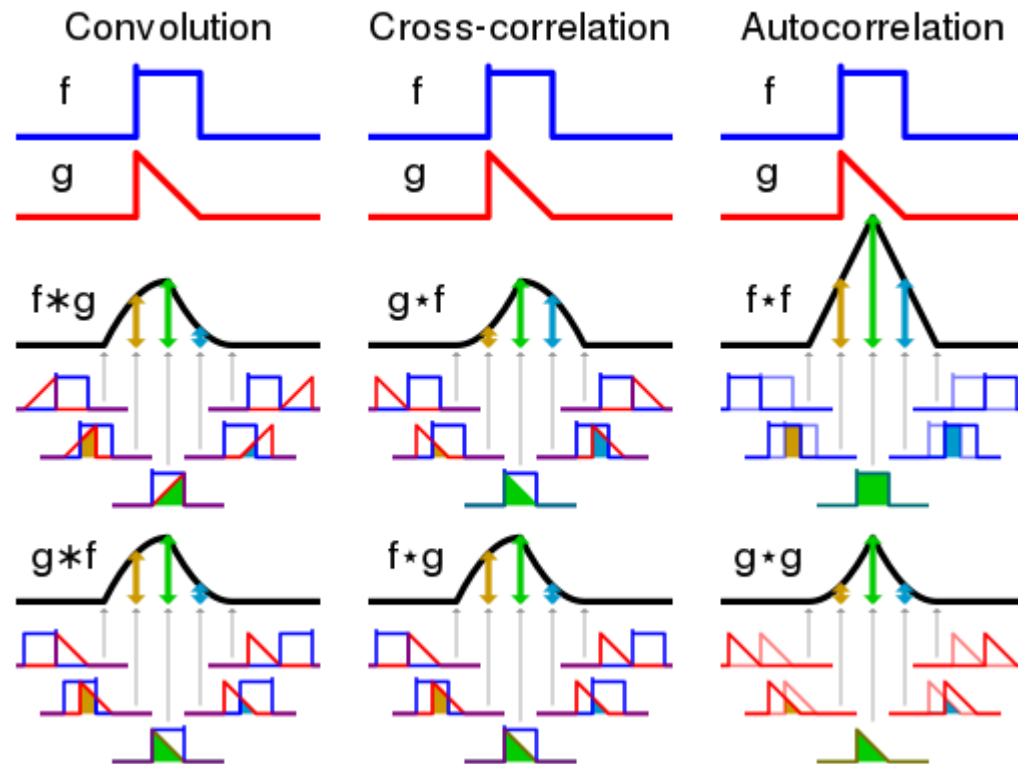
- Historical
- Let $f(x) = w \cdot x + b$
- Perceptron from 1957: $h(x) = \begin{cases} 0, & f(x) < 0 \\ 1, & \text{otherwise} \end{cases}$
- Update rule was $w_{k+1} = w_k + \alpha(y - h(x))x$ similar to gradient update rules we see today
- Passing the score through a sigmoid was likened to how a neuron fires
 - Firing rate $= \frac{1}{1 + e^{-yf(x)}}$

Naming: Why 'Convolution'



The name 'convolution' comes from the convolution operation in signal processing that is essentially a matrix matrix product.

Naming: Why 'Convolution'



Advanced Prediction Models

Deep Learning, Graphical Models and Reinforcement
Learning

Today's Outline

- Visualizing CNNs
- Transfer Learning
- Neural Net Training Tricks
 - Data Augmentation
 - Weight Initialization/Batch Normalization/Dropout

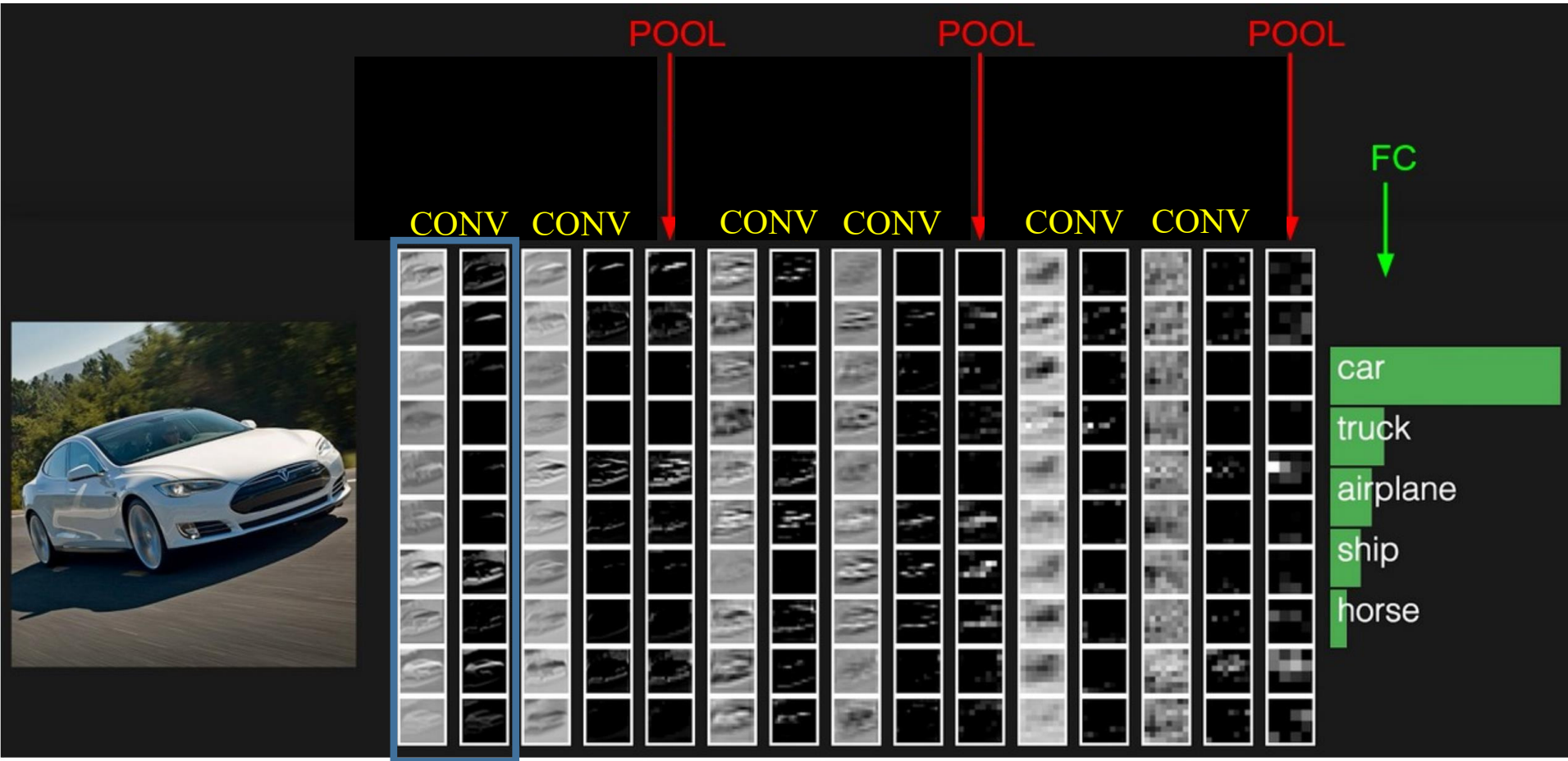
Quick Review: Convolutional Neural Networks

Recap of CNN Architecture

- Typically a CONV is followed by a POOL
- Closer to the output, use FC layers
- In CONV, smaller filters are preferred (say $3 * 3 * z$)
- Input image should ideally be divisible by 2 many times



Example: A CNN Architecture



- A different sequence of layers
- Number of filters (layer depth) is 10
- Activation tensors (flattened along depth) are shown

¹Figure: <http://cs231n.github.io/convolutional-networks/>

Example: CONV Layer Parameter Count

- Input tensor of size $90 * 90 * 10$
- Say we have 5 filters, each is $3 * 3 * 10$
- Stride is 1 and zero padding is 1
- Then output tensor will be $90 * 90 * 5$
- We can calculate manually for other strides and padding values
- Number of parameters is $5 * (3 * 3 * 10 + 1) = 455$
- Contrast with Fully connected net:
 - Number of inputs is 81000
 - Number of hidden layer neurons is 40500
 - Hence, the number of parameters is $> 3,280,500,000$

CNN and Backpropagation

- Backpropagation through a CONV layer
 - Constitutes a set of matrix-matrix products and whatever is the behavior for the nonlinearity
- Backpropagation through a POOL layer
 - Essentially like ReLU where one can keep track of the index of the maximum
- (You will not have to do this by hand in real-life)

Questions?

Visualizing CNNs

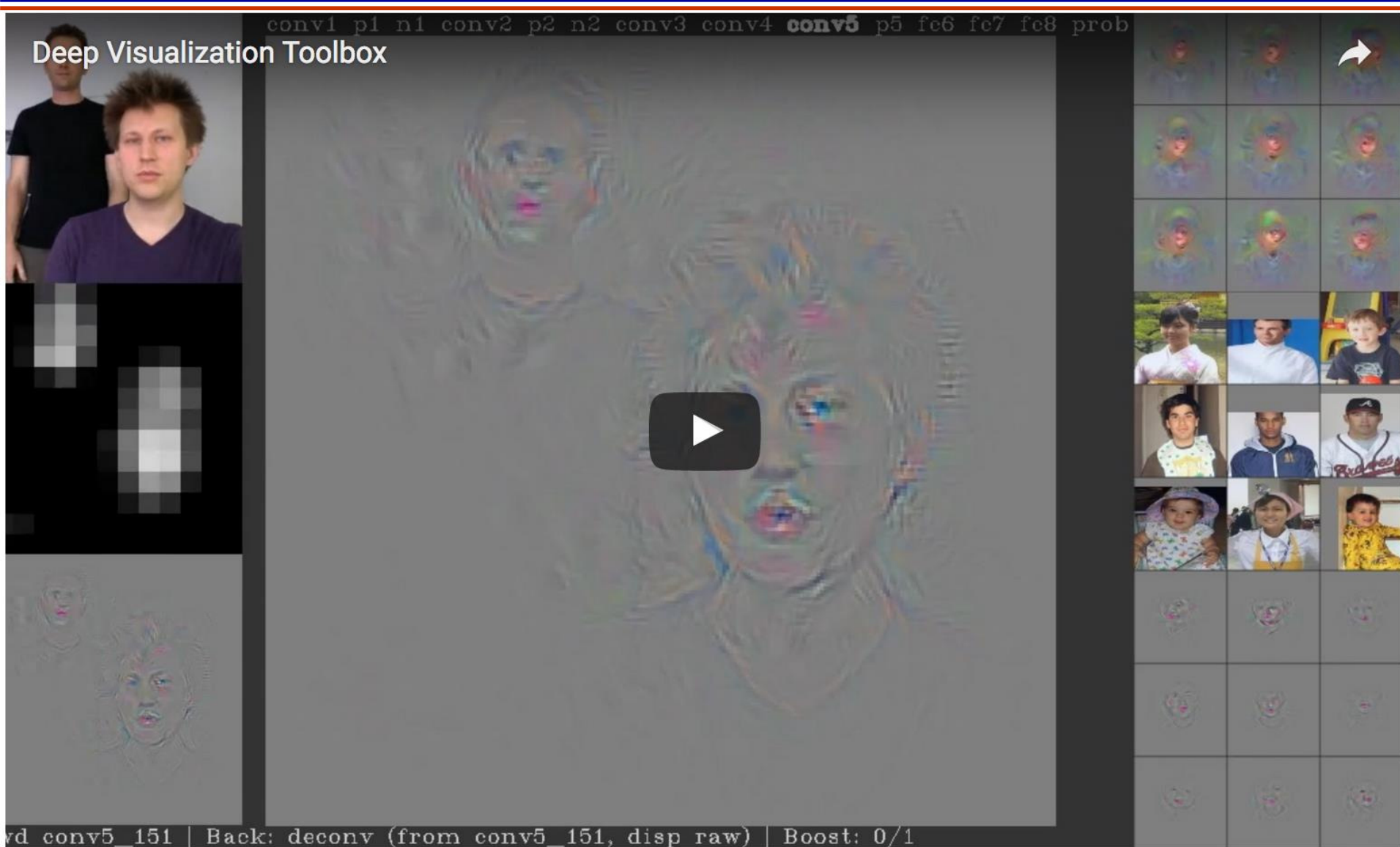
Combating Non-Interpretability

- Common criticism: learned features are not interpretable
- We will look at 4 attempts
 - Look at activations
 - Look at weights
 - Look at images in an embedded space
 - Look at impact of occlusion
 - Look at images that activate neurons highly

An Example CNN Visualization Tool

- Online tool by Adam Harley
 - <http://scs.ryerson.ca/~aharley/vis/conv/flat.html>

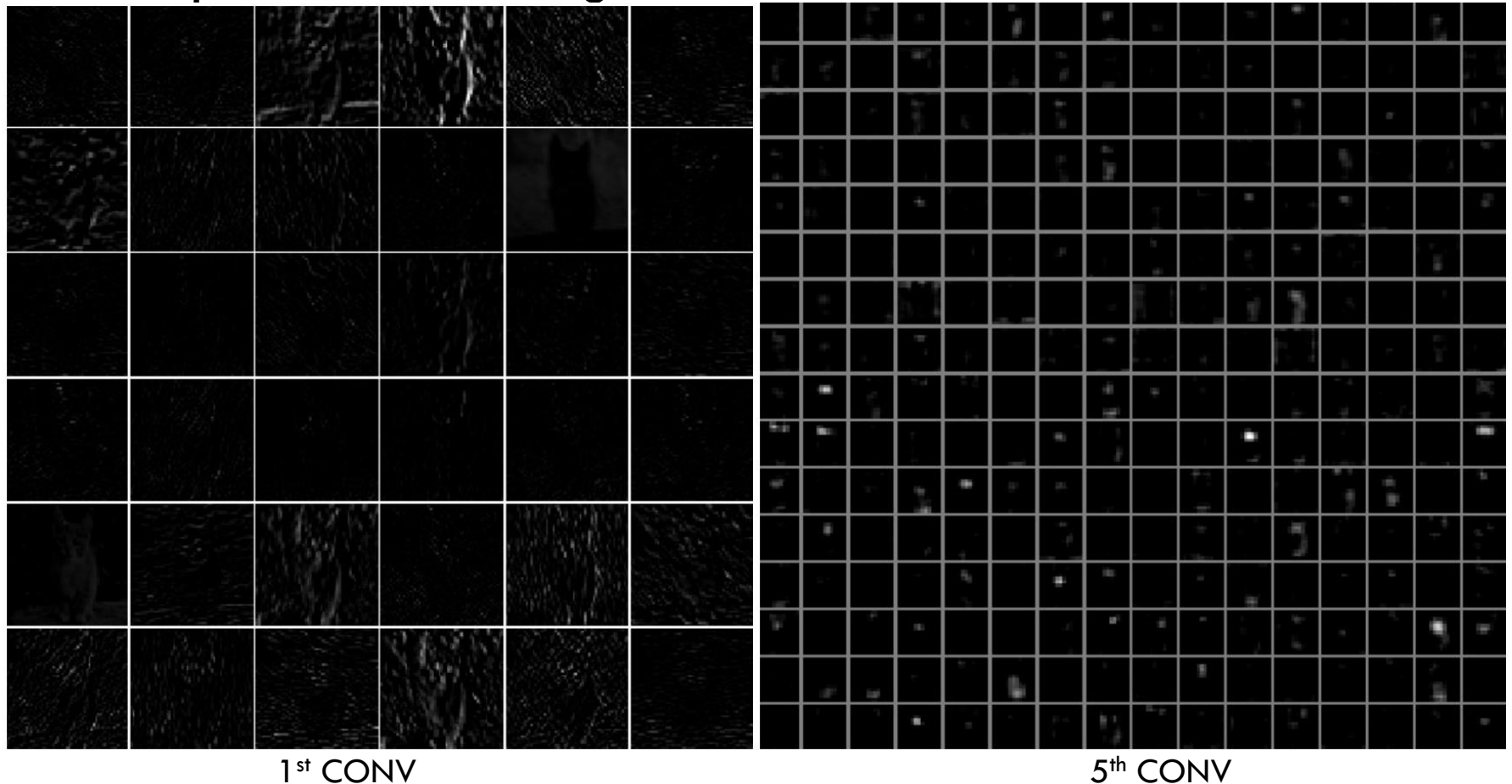
Another Example Tool



¹Figure: <http://yosinski.com/deepvis>

Visualize: Activations

- Useful to debug ‘dead’ filters (e.g., when using ReLU)
- Input is a cat image

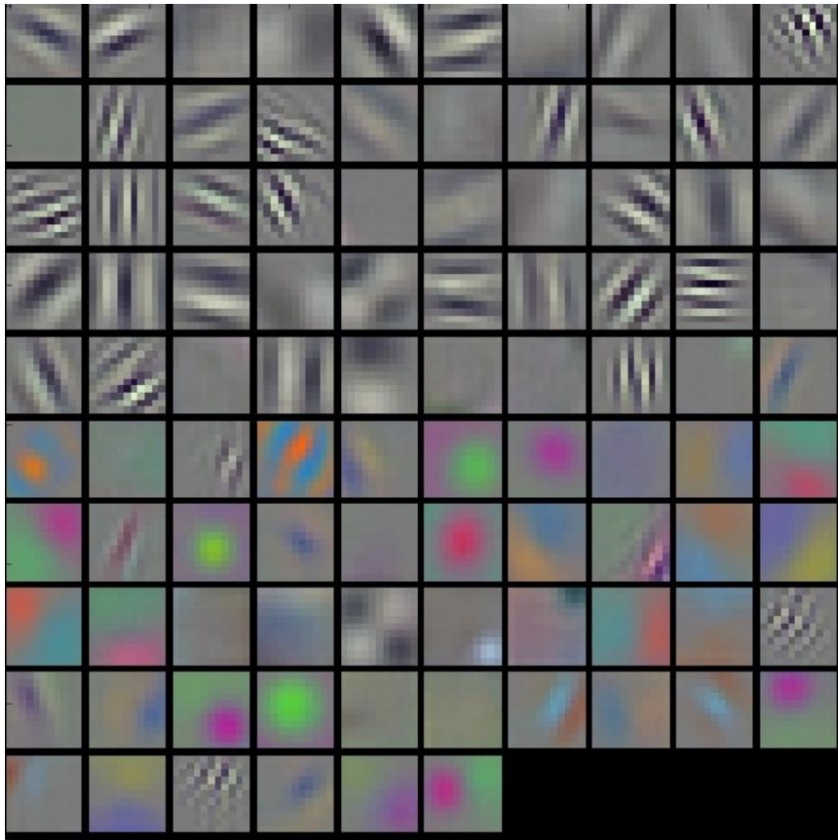


1st CONV

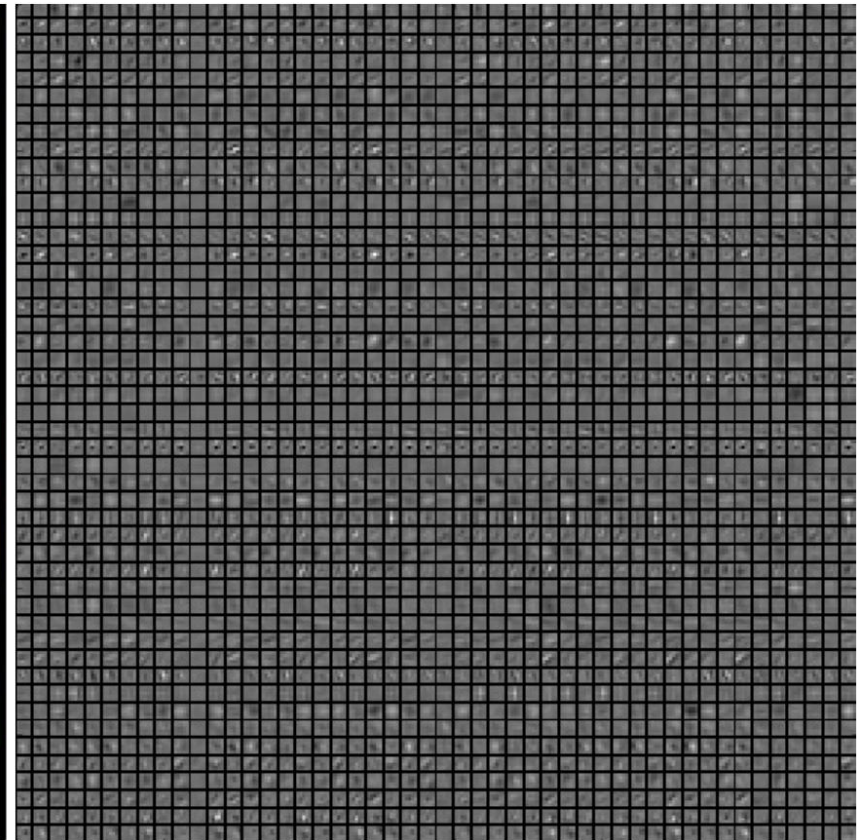
5th CONV

Visualize: Weights

- Useful to debug if training needs to be run more (if patterns are noisy)



1st CONV



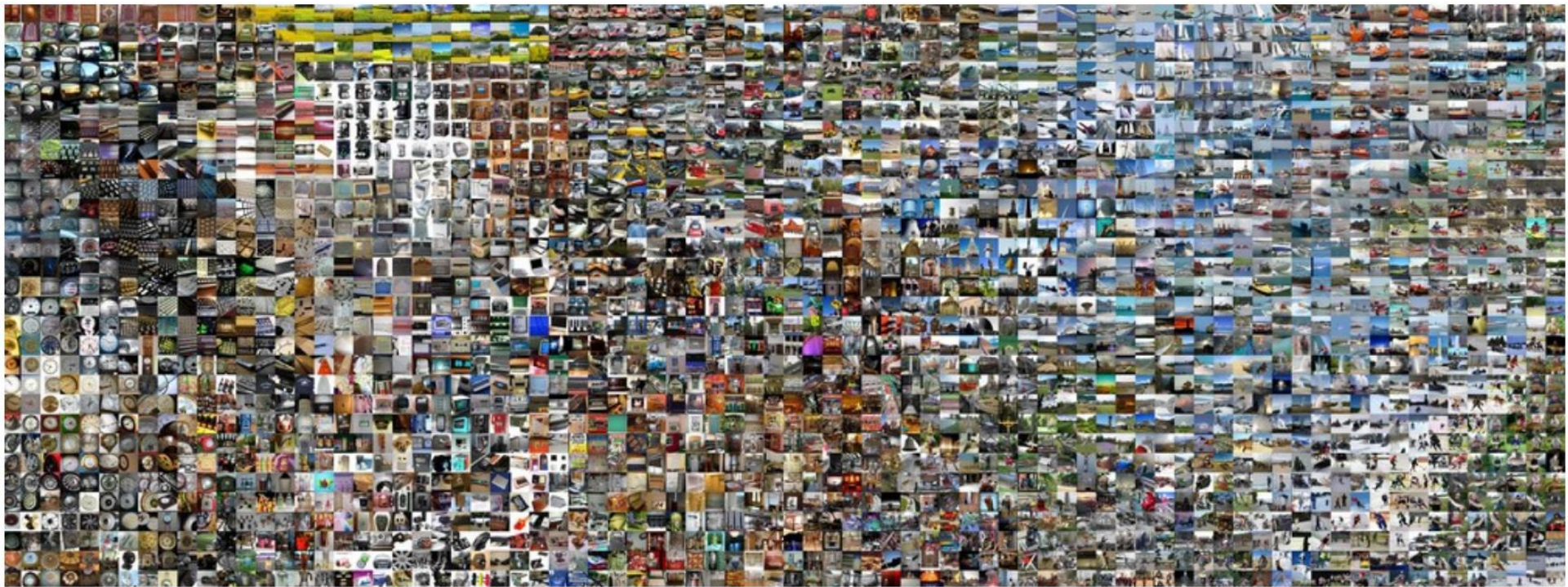
2nd CONV

Visualize: Low-Dimensional Embeddings

- CNN
 - Input: Image
 - Output: Scores
- The input to the layer that computes scores:
 - $s = W \max(0, h) + b = Wa + b$
- Activation a can be considered as a representation of the input image
- Embed a 's into a 2D space
 - Such that distance properties are preserved

Visualize: Low-Dimensional Embeddings

- In Alexnet, the output of layer before FC layer is 4096 dim
- The t-SNE embedding is shown below:



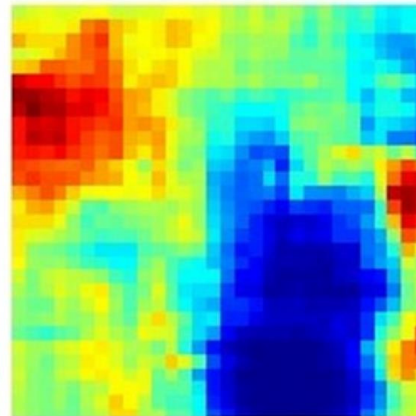
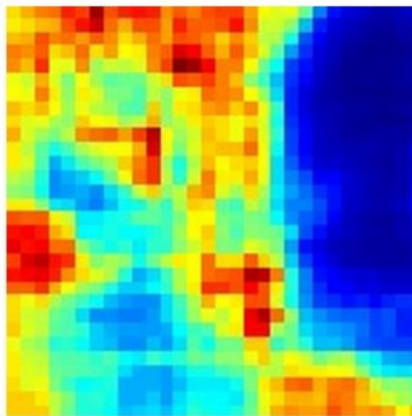
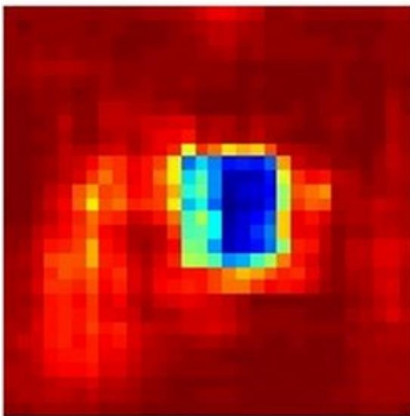
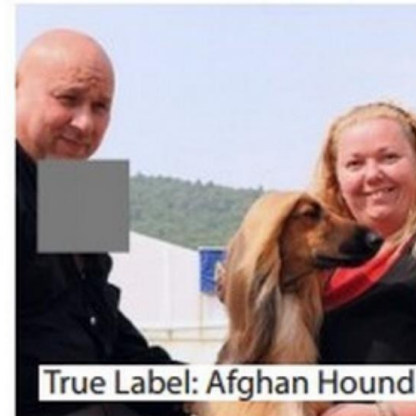
- Similarities are class-based and semantic rather than color and pixel based
 - Implies: images close to each other are similar for the CNN

Visualize: By Occlusion

- To figure out which part of the image is leading to a certain classification
- Plot the probability of class of interest as a function of occlusion

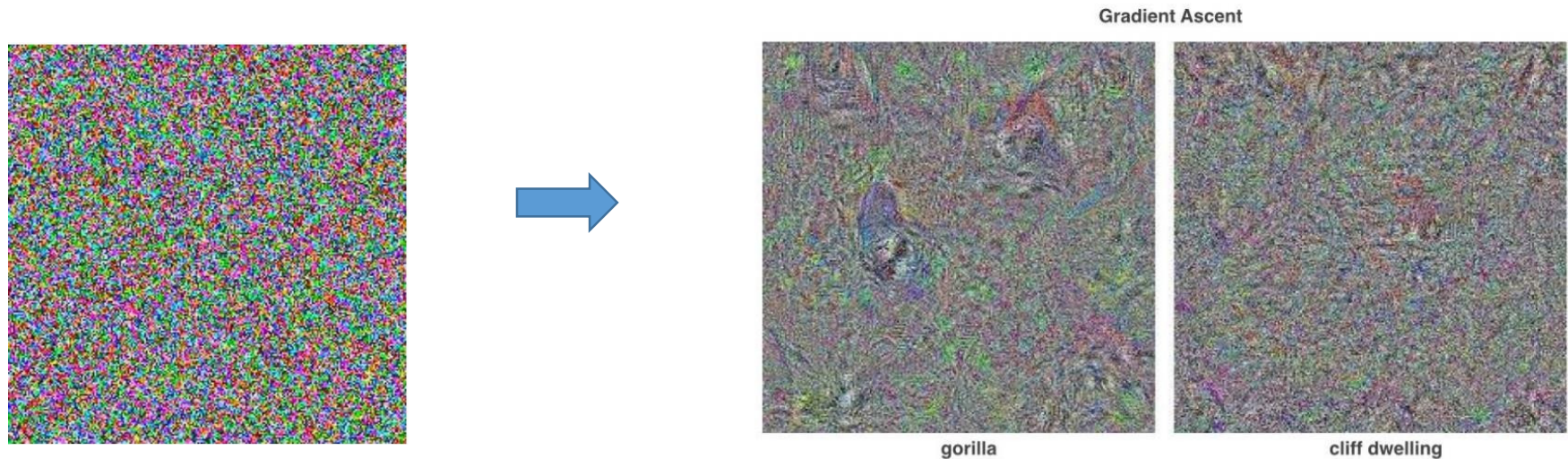
Visualize: By Occlusion

- Occlusion in grey is slid over the images and plot probability of correct class



Visualize: Synthesize Images

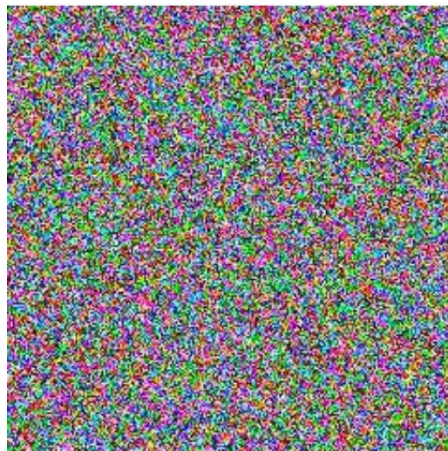
- Find images that activate a neuron the most



- Seed with 'natural' image priors

Visualize: Synthesize Images

- Find images that activate a neuron the most



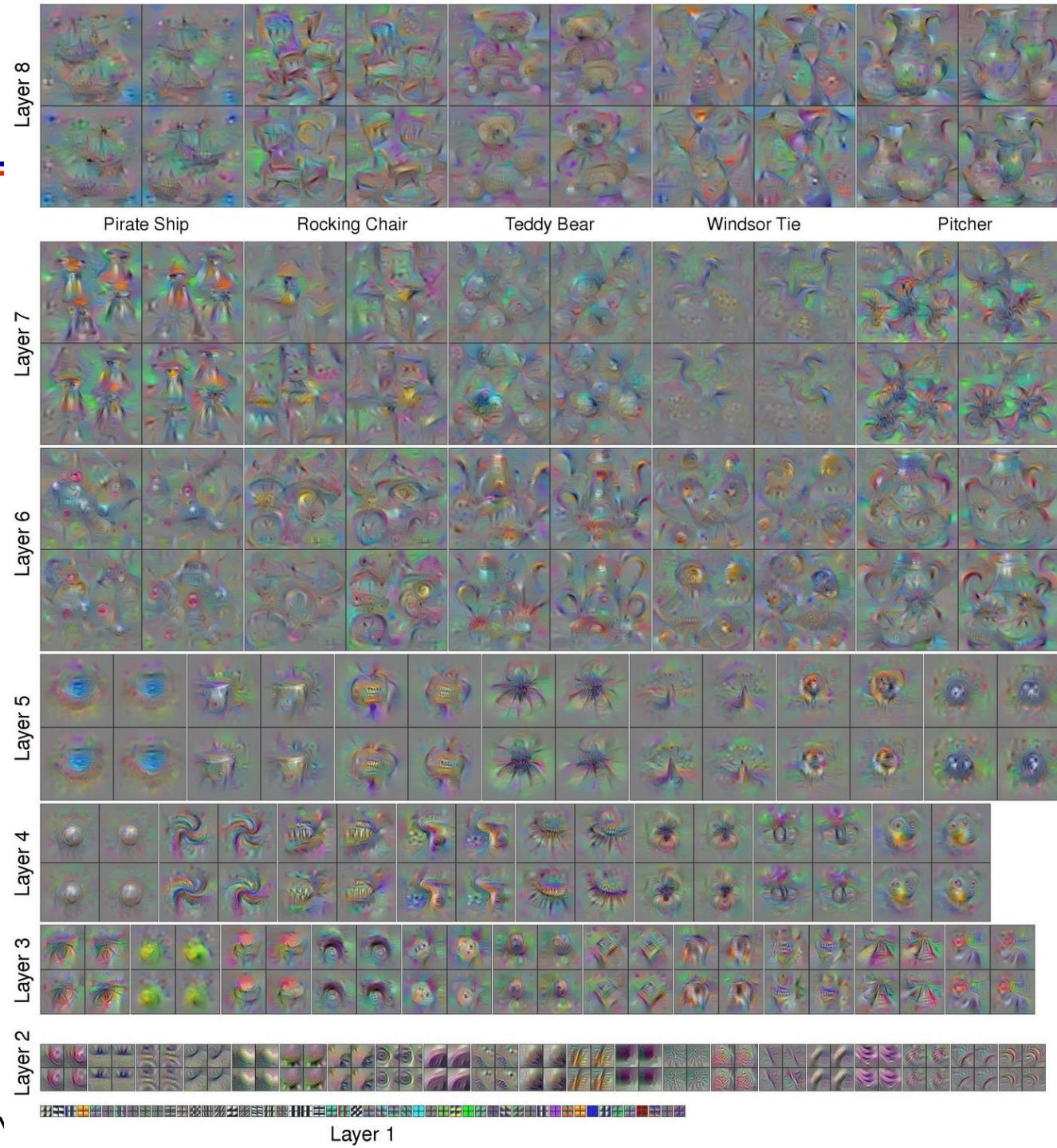
goose



ostrich

- Seed with 'natural' image priors

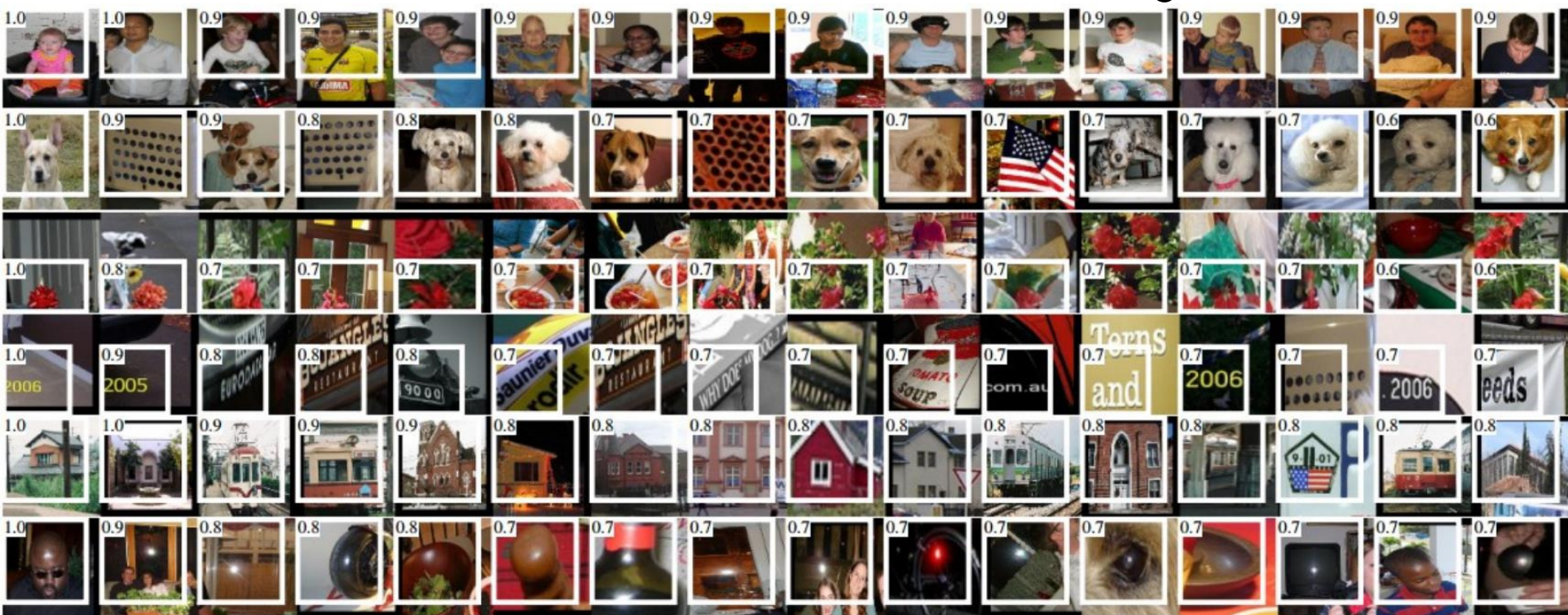
Visualize: Synthesize images



¹Figure: <http://yosinski.com/deep>

Visualize: Images that Activate a Neuron

- Track which images maximally activate a neuron
 - Understand what the neuron is tracking



5th POOL Activation values and receptive fields of some neurons in Alexnet
(May not be a good idea...)

Questions?

Today's Outline

- Visualizing CNNs
- Transfer Learning
- Neural Net Training Tricks
 - Data Augmentation
 - Weight Initialization/Batch Normalization/Dropout

Transfer Learning

Transfer Learning

- Very few people train a deep feedforward net or a CNN from scratch
- **Myth:** “We need a lot of data to use Deep Neural Networks”
- We will see two approaches if we have small data
 - Feature extraction
 - Fine-tuning
- Both these are loosely termed as **Transfer learning**

Transfer by Feature Extraction (I)

- Get a pretrained CNN
 - Example: VGG or AlexNet that was trained on Imagenet
- Remove the last FC (that outputs 1000 dim score)
- Pass new training data to get **embeddings**

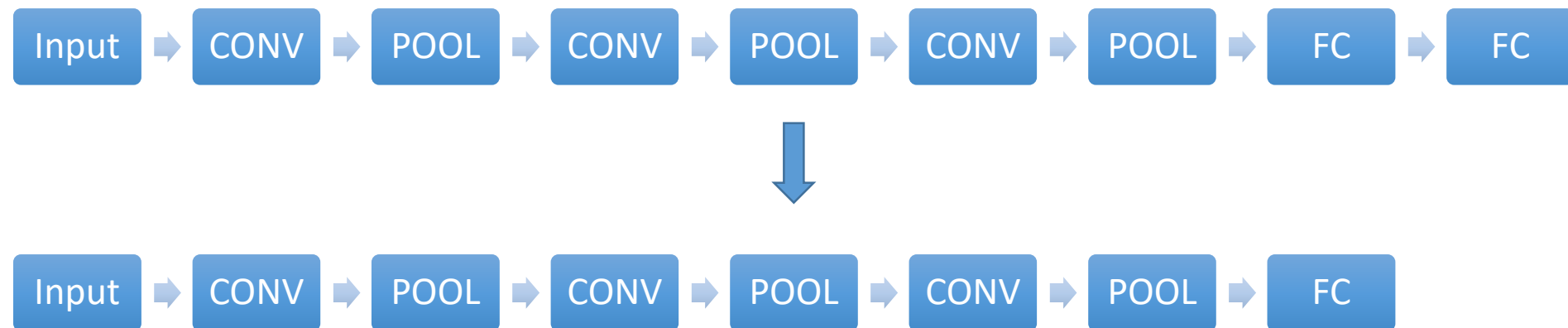
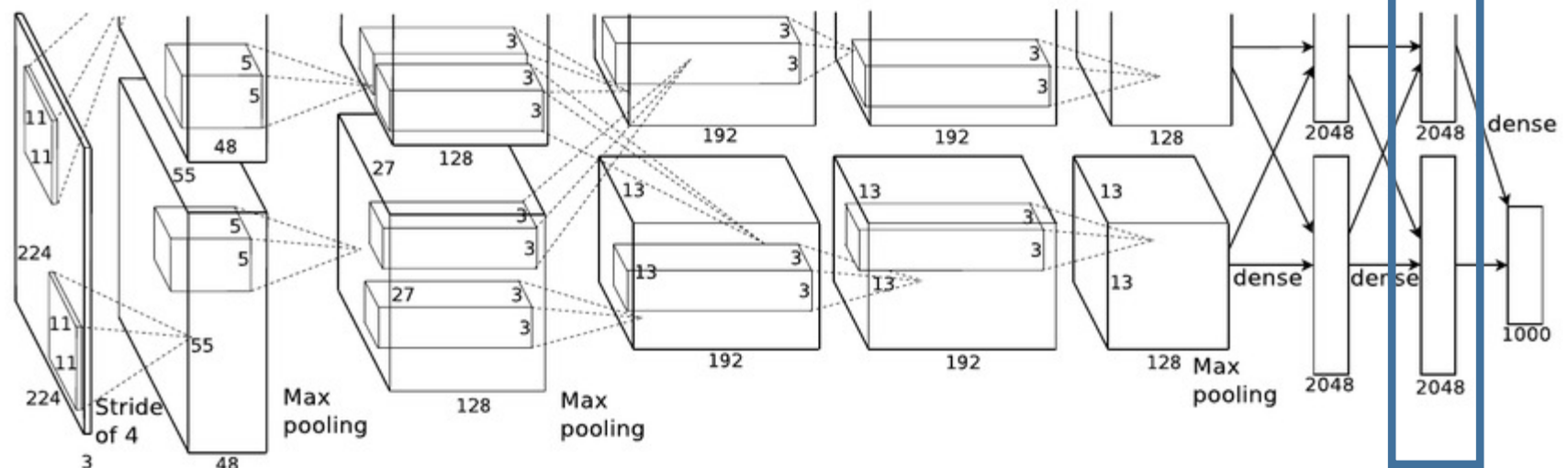


Image Embeddings

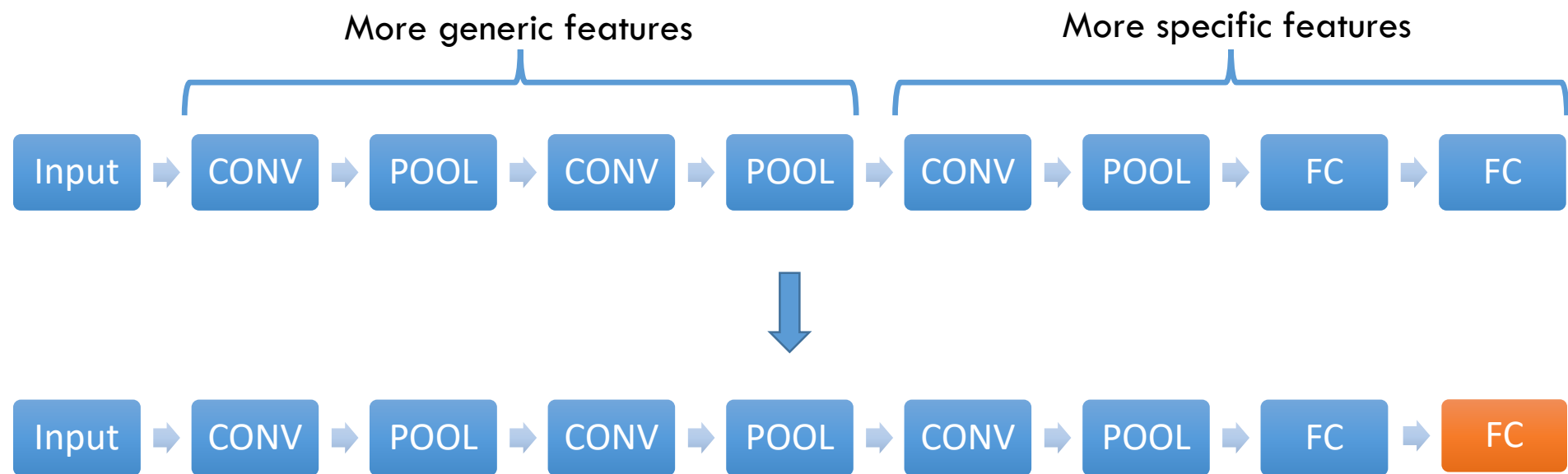
- We can think of the penultimate hidden layer activations (a 4096 dim vector) as an **embedding** of the image



- This is the **activation vector** or the **representation** or the **CNN code** of the image

Transfer by Feature Extraction (II)

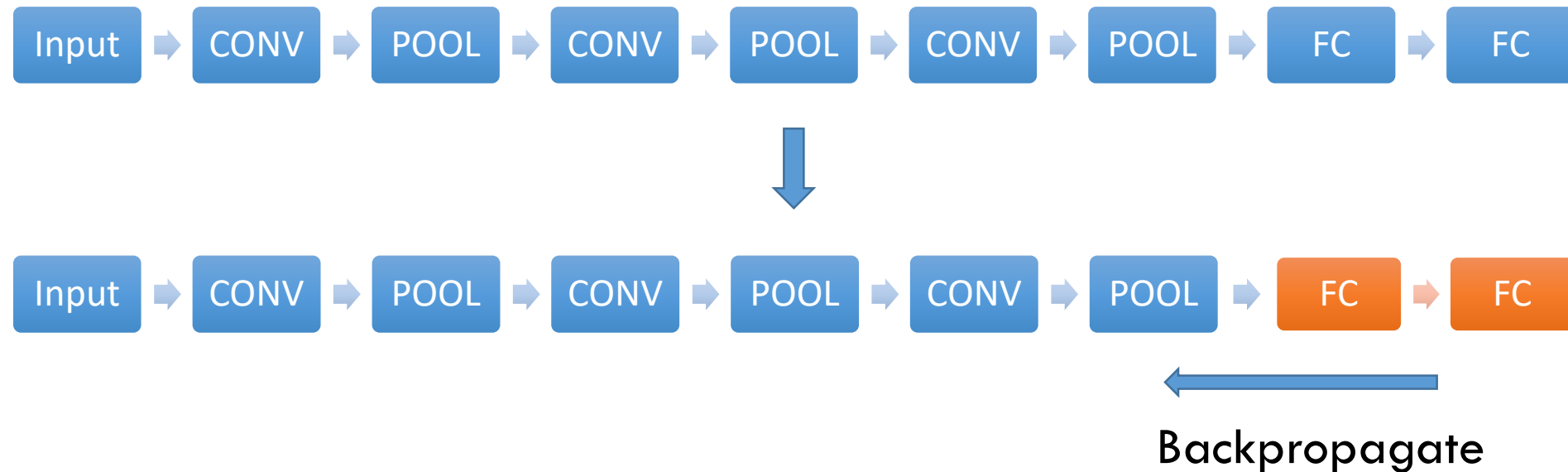
- Input these to a linear or non-linear classifier!



- For example, for imagenet output 1000 dim scores
- For our data, output say 2 scores (cat vs dog)

Transfer by Fine-tuning

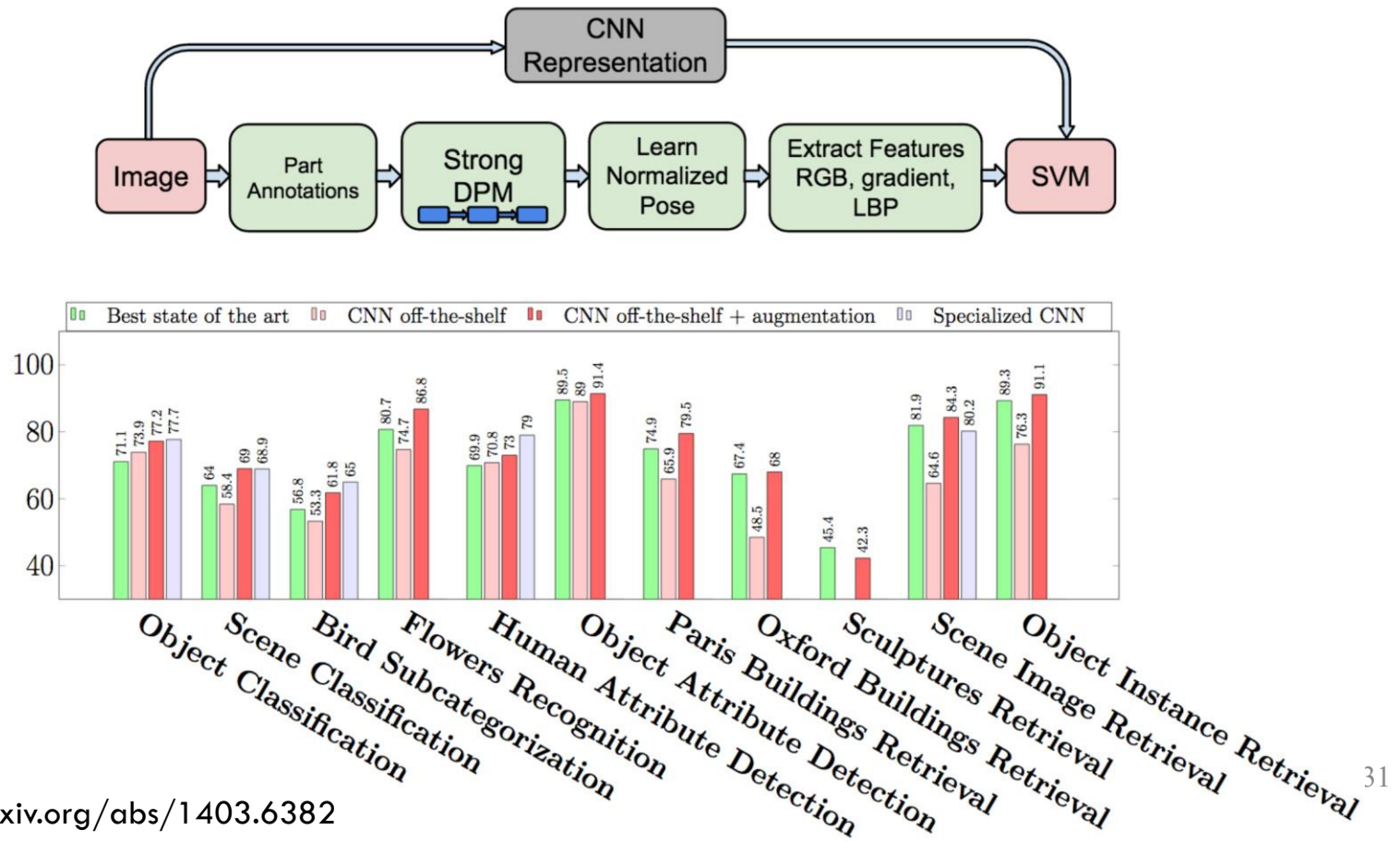
- Retrain or **finetune** additional layers of the pre-trained if we have more data



- We can even go all the way back to the first layer if there is a lot of training data available

Benefits of Transfer

- We can get a significant boost in performance compared to hand engineered classification/machine-learning pipelines



Aside: Other Vision Tasks

- Some example vision tasks are given below

Classification



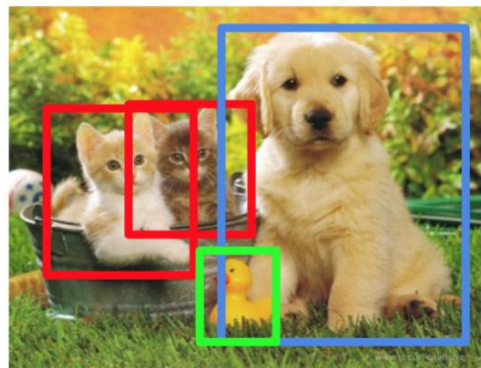
CAT

**Classification
+ Localization**



CAT

Object Detection



CAT, DOG, DUCK

**Instance
Segmentation**



CAT, DOG, DUCK

Single object

Multiple objects

Transfer Learning Choices

- When to transfer

	Similar dataset	Different dataset
Small data	Feature extract	NA
Large data	Fine-tune a bit	Fine-tune a lot

- How to transfer

- Get pre-trained models for popular software systems
 - Tensorflow Models
 - Keras Model Zoo
 - Caffe Model Zoo

This is key for projects!

VGG Net in Keras

- 2nd in the 2014 ILSVRC classification task
- 3x3 conv filters with stride 1
- ReLU non-linearity
- 5 POOL layers
- 3 FC layers

<https://gist.github.com/baraldilorenzo/07d7802847aaad0a35d3>



Questions?

Today's Outline

- Visualizing CNNs
- Transfer Learning
- Neural Net Training Tricks
 - Data Augmentation
 - Weight Initialization/Batch Normalization/Dropout

Neural Net Training Tricks

Neural Nets in Practice

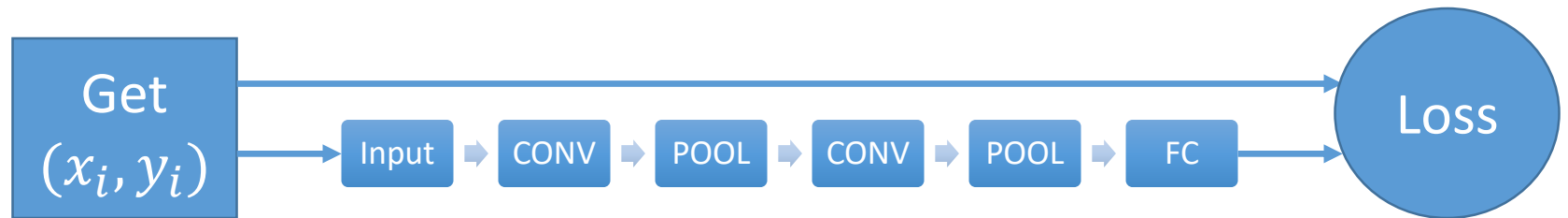
- There are a few **empirically validated** techniques that improve the performance (classification accuracy) of feedforward nets and CNNs
- We will look at some of these
 - Data: data augmentation
 - Model: initialization, batch normalization, dropout
- For our discussion, we will fix the optimization technique to be a gradient based method. We will revisit related algorithmic enhancements later.

Data

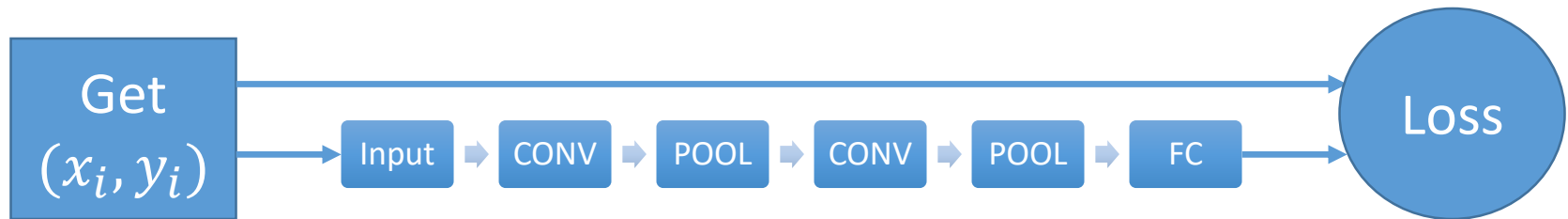
- Data:
 - How is it handled?
 - What is its quality?
- Handling:
 - Deep nets may need to read lots of data (images), so keep them in contiguous spaces of hard-disk
- Quality:
 - Collect as much clean data as possible. At the same time, unclean may also be good enough

Next: Extract the most out of existing data for CNNs

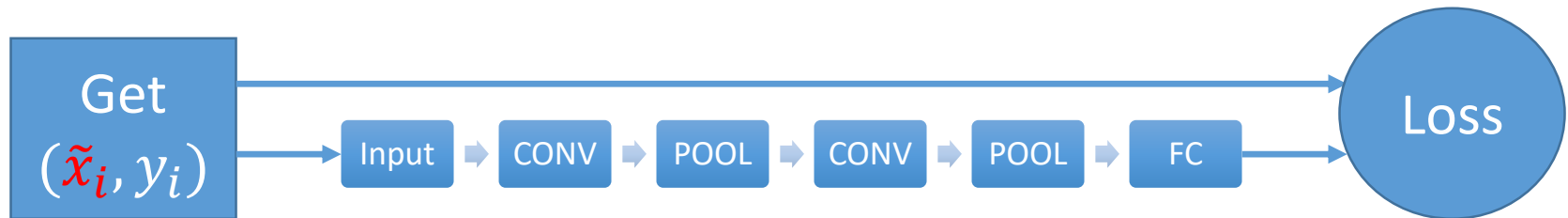
Augmenting Data (I)



Augmenting Data (I)



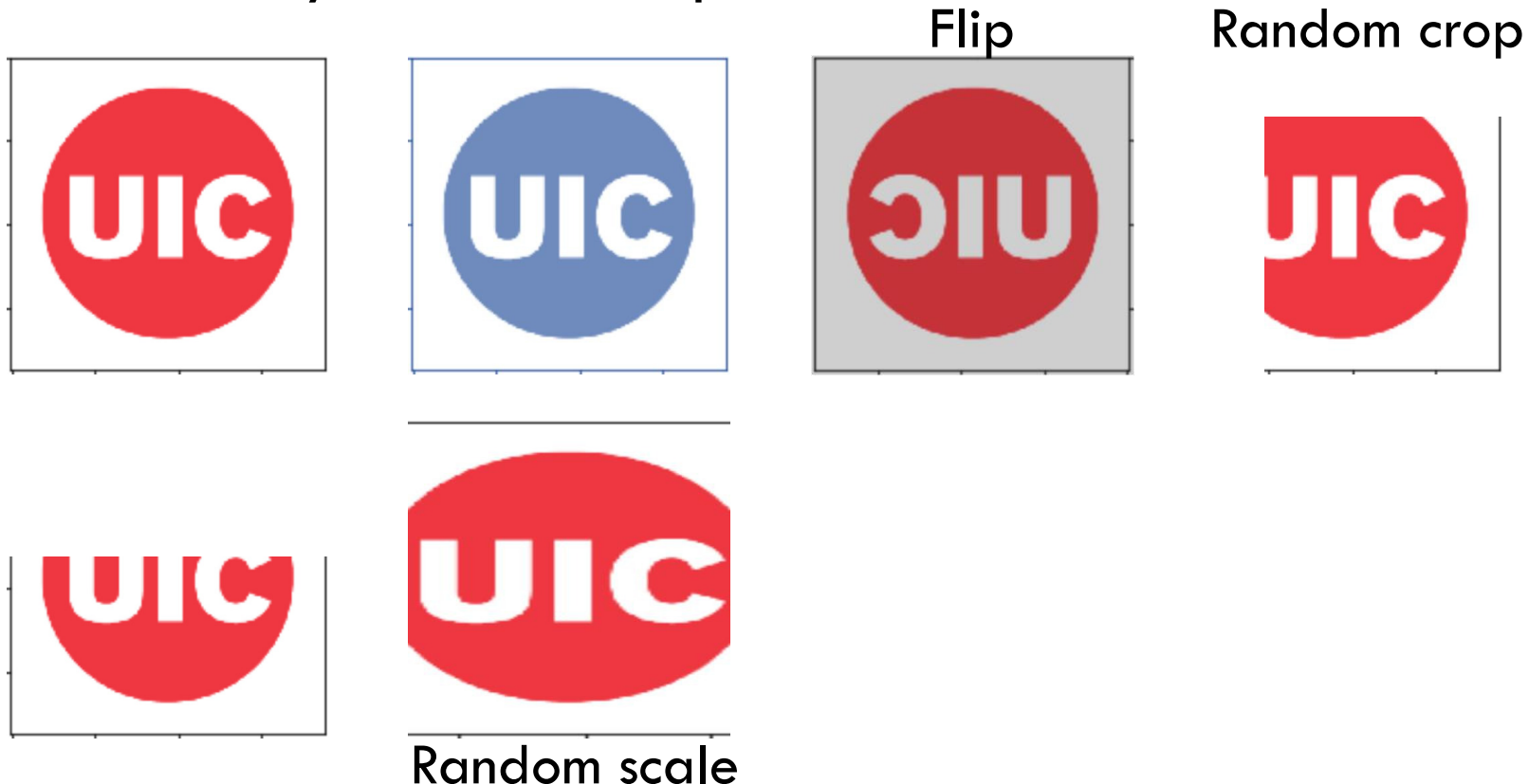
And



Where $\tilde{x}_i = g(x)$ is a transformation

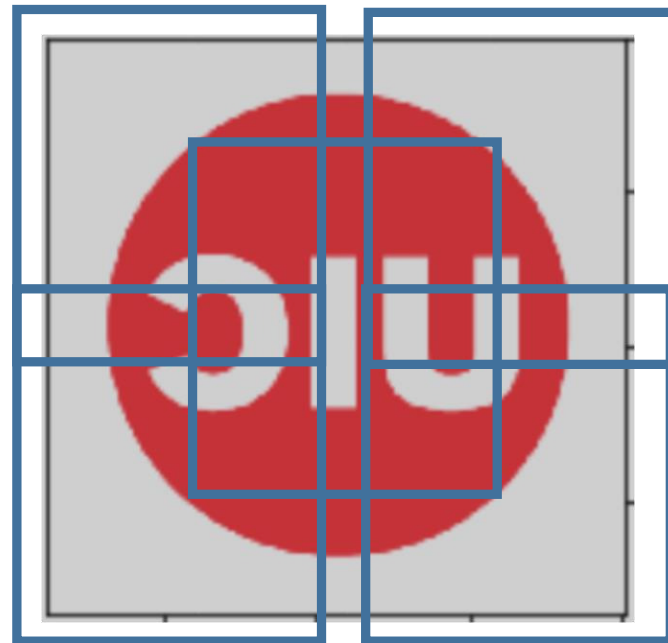
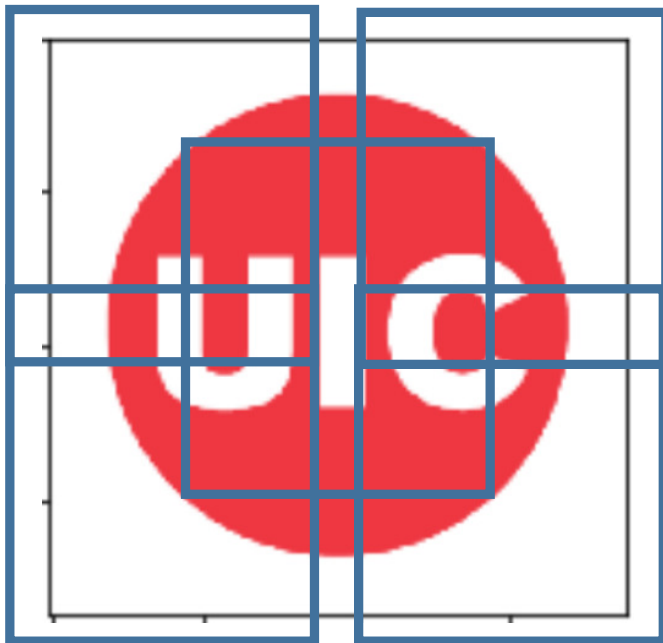
Augmenting Data (II)

- We are changing the input without changing the label
- We then add this new example to our training set
- Widely used technique!



Augmenting Data (III)

- At test time, **average** the predictions of a fixed set of transformations
- Example (for Resnet, the ILSVRC 2015 winner):
 - Image at 5 scales: 224, 256, 384, 460 and 640
 - At each scale, get 10 224×224 crops



Augmenting Data (IV)

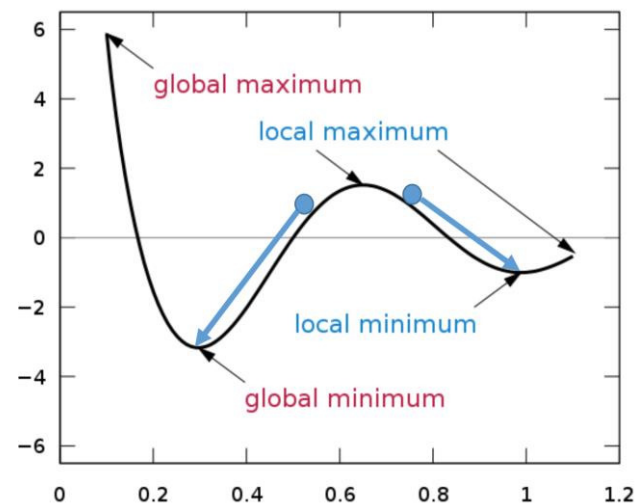
- Other ways to augment data include
 - Changing contrast and color
 - Mix translations, rotations, stretching, shearing, distortions
- This is very useful for small datasets
- From one point of view, this is essentially
 - Adding some noise during training
 - Marginalizing noise out at test

Model

- We have already seen few choices
 - Activation function or nonlinearities
 - Number of layers and number of neurons per layer
 - CNN filter choices ...
- There are other choices while training deep neural nets (including CNNs) that also **make a difference**
 - Weight initialization
 - Batch normalization
 - Dropout

Model: Weight Initialization

- Weight initialization plays a key role in training deep networks
 - Example: $W = 0$ may be bad
- Not just the issue of local optima
- But also the magnitudes of gradients in backprop
 - Activation statistics (mean and variance) influence gradients
- Heuristics available in the literature to initialize W



Model: Batch Normalization

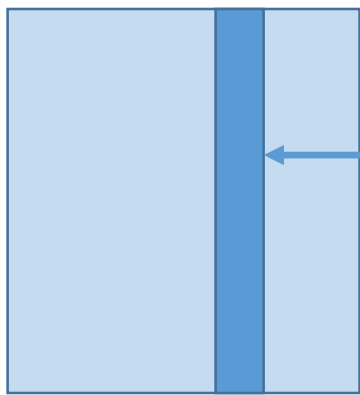
- Activations magnitudes and their statistics depend on the dataset, the network and the nonlinearity used
- Their statistics influence gradient propagation, hence also learning
- Is there a way to control them?
 - Yes, through batch normalization!

Model: Batch Normalization

- Idea: Make each activation unit-Gaussian by subtracting the mean and then dividing by standard deviation

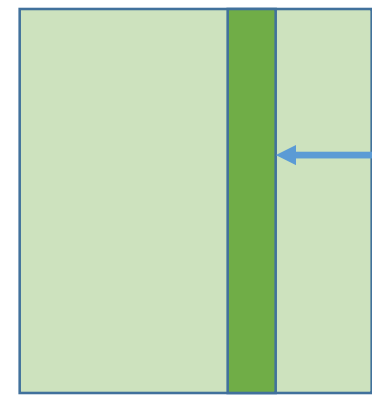
Batch-size = N

Number of output neurons = D



$N \times D$

$$\hat{x} = \frac{\gamma(x - E[x])}{\sqrt{Var[x]}} + \beta$$

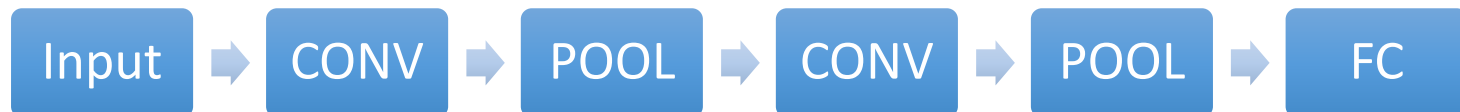


$N \times D$

- Is a differentiable function: hence no issue with backpropagation
- At test time, there is no batch. Use the training data means and variances

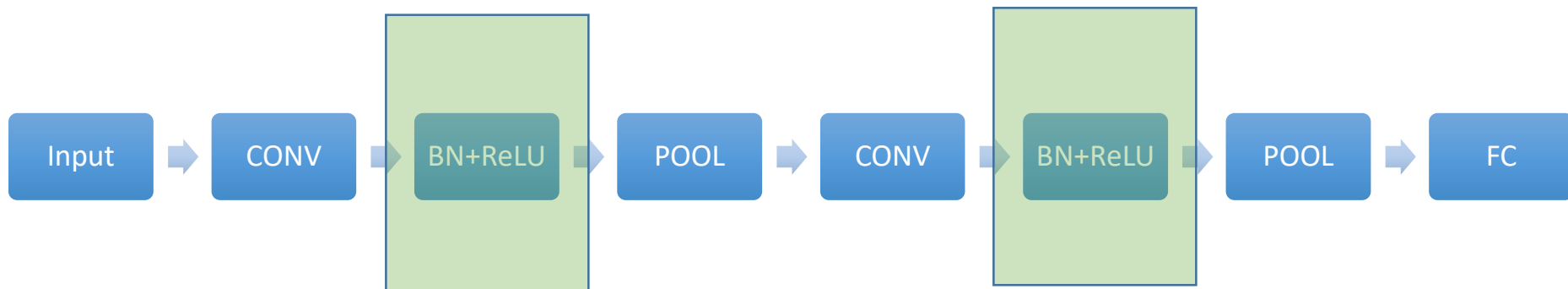
Model: Batch Normalization

- Previously,



- Now

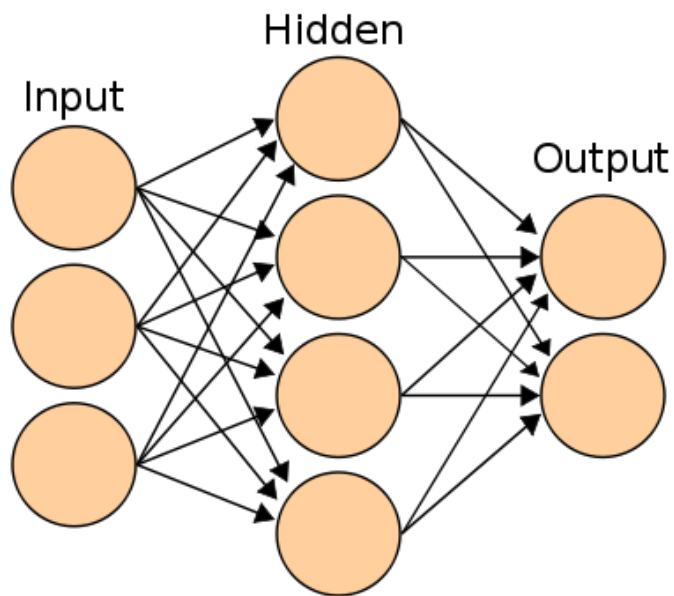
- Insert a Batch Normalization layer between CONV and nonlinearity (ReLU)



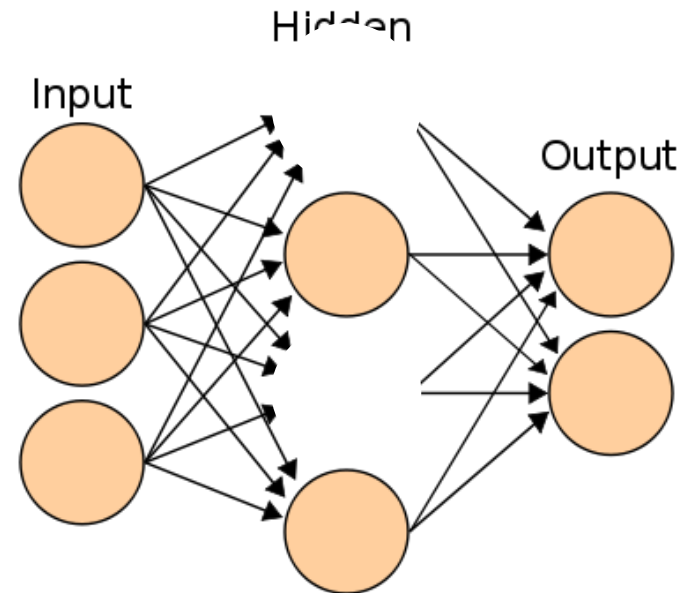
- Empirically observed: improved gradient flows, less sensitive to **initialization**.

Model: Dropout (Regularization)

- Idea: During training, every time we forward pass, we set the output of a few neurons to zero with some probability



Without dropout



One pass with dropout

Model: Dropout (Regularization)

- Intuitively, it is
 - Making us use smaller capacity of the network. Hence, can think of it as a **regularization**
 - Forcing all the neurons to be useful. Hence there is over-representation or redundancy
- Also think of it as
 - Subsampling a part of the network for each example
 - Thus, we get an ensemble of neural networks that share parameters

Model: Dropout (Regularization)

- Higher probability means stronger regularization
- At test time,
 - Instead of doing many forward passes
 - Perform no dropout
 - Scale all activations by the probability of dropout
- Example:
 - Say dropout with probability p
 - Originally: $f(x, W_1, b_1, W_2, b_2) = W_2 \max(0, W_1 x + b_1) + b_2$
 - With dropout: $W_2 * p * \max(0, W_1 x + b_1) + b_2$

Summary (I)

- CNN are very effective in image related applications.
 - State of the art!
- Exploit specific properties of images
 - Hierarchy of features
 - Locality
 - Spatial invariance
- Lots of **design choices** that have been empirically validated and are intuitive. Still, there is room for improvement.

Summary (II)

- We saw
 - Visualizations to understand how CNNs work
 - Transfer learning applied to CNNs (important for applications)
 - An excellent way to get a deep learning solution working
 - There is no need for large datasets to get started

Summary (III)

- Neural Nets Training Tricks
 - Revisited data: data augmentation
 - Revisited models: initialization, batch norm, dropout
- To train state of the art deep learning systems, you have to rethink:
 - (a) data, (b) models, and (c) optimization¹
 - What is the most bang per buck for your business?
- If the deep learning system is core to the business, look at engineering best practices (we saw some today)

¹We did not cover this in this lecture

Appendix

Sample Questions

- How does a 2 layer feedforward net differ from a linear classifier?
- Describe why nonlinearities are introduced in a neural network? Why is the ReLU non-linearity called a gradient gate?
- Describe the parameter sharing property of a convolutional layer
- How is backpropagation used while optimizing the parameters of a neural network?

Advice

- In spite of all these design choices, for 90% of the applications, pick an architecture that works well on an established dataset (e.g., Imagenet)
- Focus on the application and business considerations, not architectural decisions!

Practical Considerations

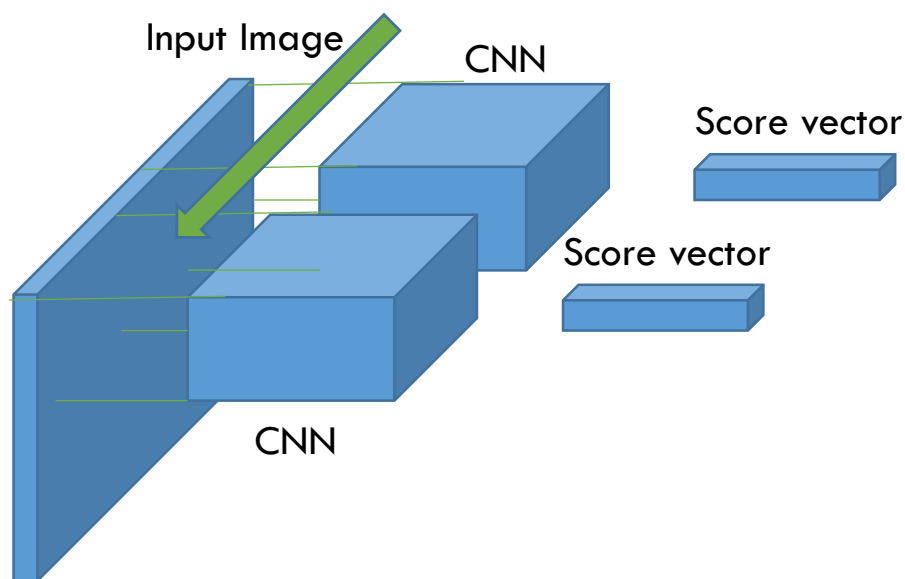
- Model choice: nonlinearity, number of layers, number of neurons
- Data preprocessing: batch normalization, subtracting mean of inputs
- Parameter initialization: random or zeroes?
- Learning rate: How to change?
- Batch normalization: re-normalizing activations
- Monitoring learning: plot graphs of training and validation
- Cross validation: hyper-parameter tuning is non-trivial

Partial Robustness to Input Size

- The input image size determines the tensors in intermediate stages
- Example
 - Alexnet requires $224 \times 224 \times 3$ sized images
- What if we have a larger sized image?
 - We can 'convert' FC layers to equivalent CONV layers for efficiency
 - Then slide the original CNN over the larger image!
 - This leads to a 'single' forward pass

Partial Robustness to Input Size

- Instead of a single vector of scores, now we get a bunch of scores



Advanced Prediction Models

Deep Learning, Graphical Models and Reinforcement
Learning

CVPR 2017 Best Paper

Densely Connected Convolutional Networks

Gao Huang*
Cornell University
gh349@cornell.edu

Zhuang Liu*
Tsinghua University
liuzhuangthu@gmail.com

Kilian Q. Weinberger
Cornell University
kqw4@cornell.edu

Laurens van der Maaten
Facebook AI Research
lvdmaaten@fb.com

Abstract

Recent work has shown that convolutional networks can be substantially deeper, more accurate, and efficient to train if they contain shorter connections between layers close to the input and those close to the output. In this paper, we embrace this observation and introduce the Dense Convolutional Network (DenseNet), which connects each layer to every other layer in a feed-forward fashion. Whereas traditional convolutional networks with L layers have L connections—one between each layer and its subsequent layer—our network has $\frac{L(L+1)}{2}$ direct connections. For each layer, the feature-maps of all preceding layers are used as inputs, and its own feature-maps are used as inputs into all subsequent layers. DenseNets have several compelling advantages: they alleviate the vanishing-gradient problem, strengthen feature propagation, encourage feature reuse, and substantially reduce the number of parameters. We evaluate our proposed architecture on four highly competitive object recognition benchmark tasks (CIFAR-10, CIFAR-100, SVHN, and ImageNet). DenseNets obtain significant improvements over the state-of-the-art on most of them, whilst requiring less memory and computation to achieve high performance. Code is available at <https://github.com/liuzhuang13/DenseNet>.

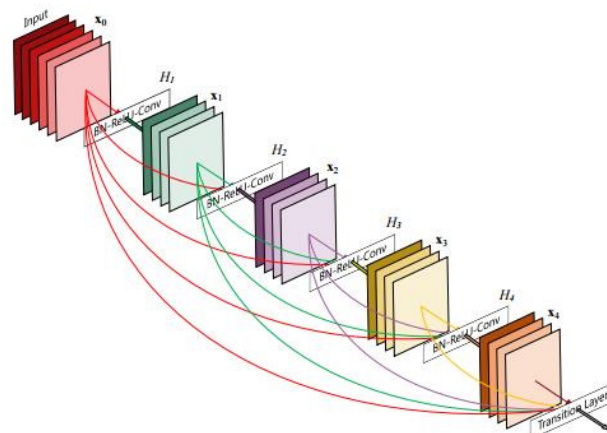


Figure 1. A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

Networks [33] and Residual Networks (ResNets) [11] have surpassed the 100-layer barrier.

As CNNs become increasingly deep, a new research problem emerges: as information about the input or gradient passes through many layers, it can vanish and “wash out” by the time it reaches the end (or beginning) of the network. Many recent publications address this or related

Today's Outline

- Introduction to Natural Language Processing
- Models with Simple Representations
- Word Embeddings and Word2Vec

Introduction to Natural Language Processing

Natural Language Processing (NLP)

- Concerns will all aspects of natural languages
- *We will only sample a very narrow set of topics in this area*
- We will sample a few ways to deal with text
 - Text is a sequence of symbols
 - Naïve way: represent them as one-hot encoded vectors
 - We will see some better methods today

Motivation: Machine Translation



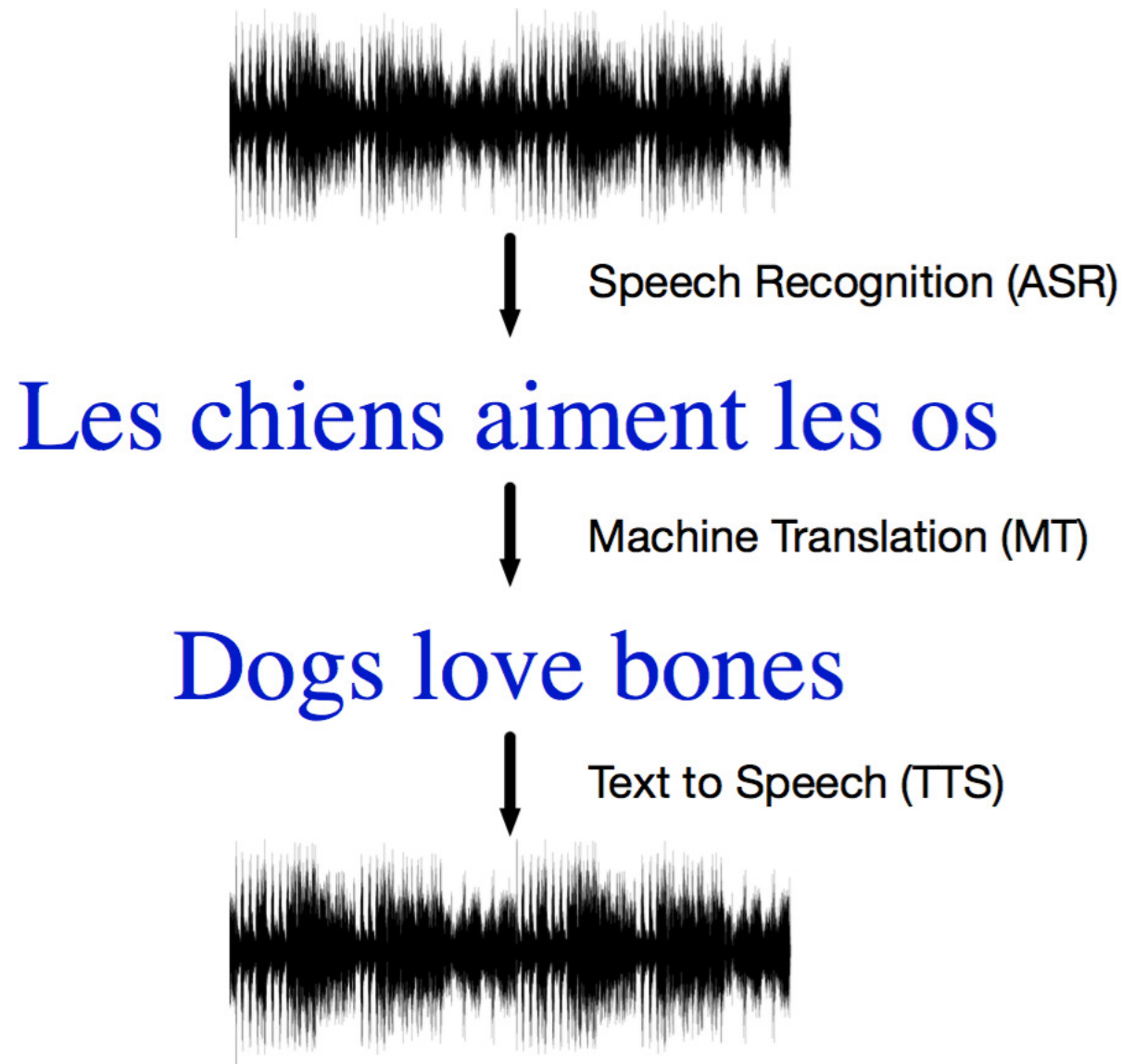
Motivation: Query Answering

Document The BBC producer allegedly struck by Jeremy Clarkson will not press charges against the “Top Gear” host, his lawyer said Friday. Clarkson, who hosted one of the most-watched television shows in the world, was dropped by the BBC Wednesday after an internal investigation by the British broadcaster found he had subjected producer Oisin Tymon “to an unprovoked physical and verbal attack.” ...

Query Who does the article say will not press charges against Jeremy Clarkson?

Answer Oisin Tymon

Motivation: Speech to Speech



Motivation: Visual Query Answering



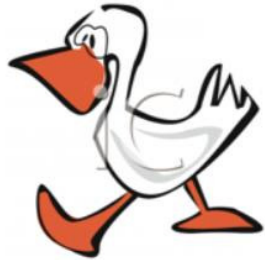
What is the man holding?

Does it appear to be raining?

Does this man have 20/20 vision?

Motivation: Harder Text Problems

Sense



I saw her duck



Idioms

He kicked a goal
He kicked the ball
He caught the ball
He kicked the bucket

Reference

The ball did not fit in the box because it was too
[big/small].

Models with Simple Representations

Side-stepping Word-word Relationships

- We will look at a few models that
 - Don't explicitly account for word-word relationships
- These are:
 - Naïve Bayes Spam Filter
 - Markov Language Model
 - Latent Dirichlet Allocation
 - Conditional Random Field based Classifier
 - CNN based Sentence Classifier

Naïve Bayes Spam Filter

- **Key assumption**

Words occur independently of each other given the label of the document

$$p(w_1, \dots, w_n | \text{spam}) = \prod_{i=1}^n p(w_i | \text{spam})$$

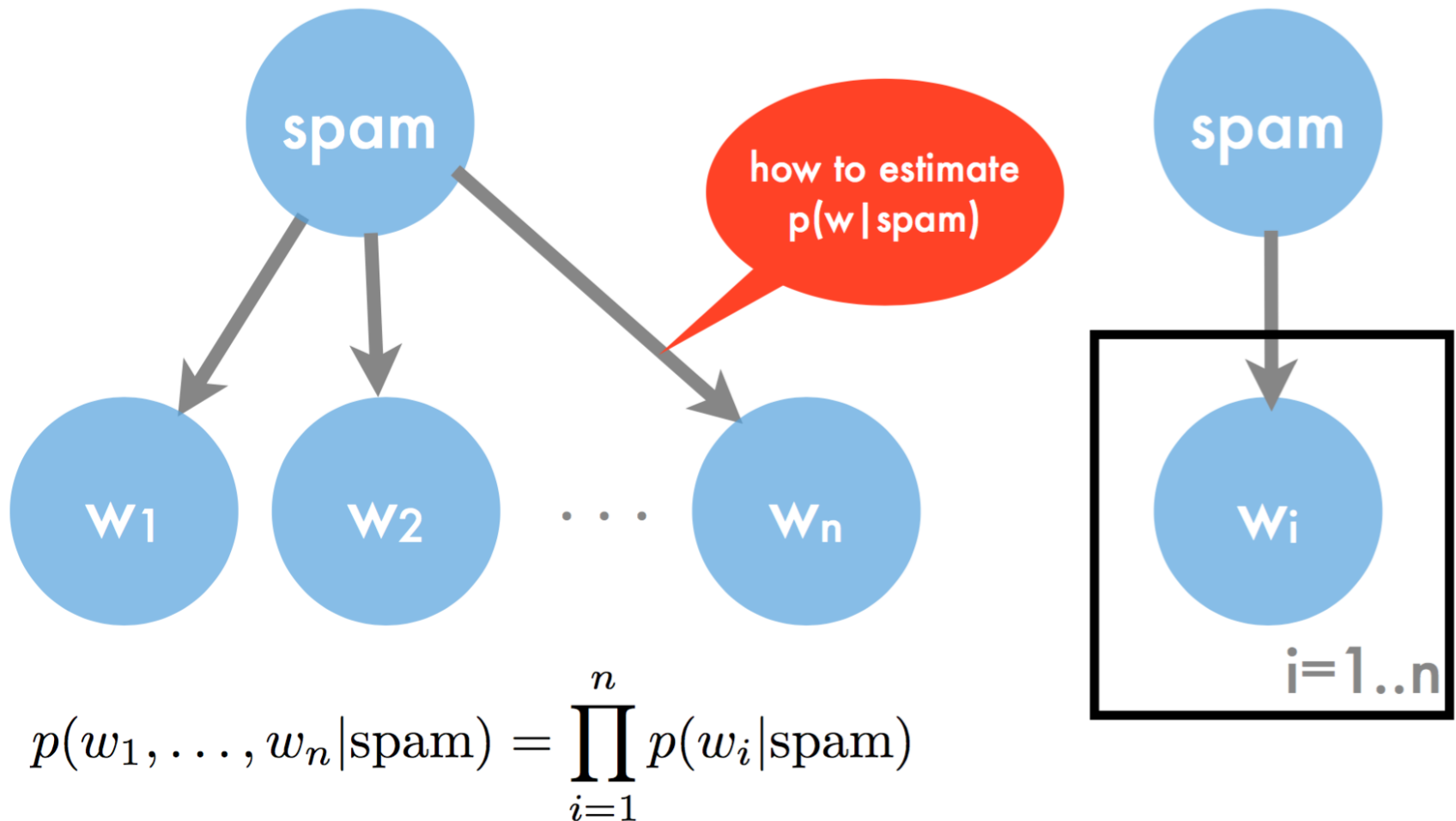
- **Spam classification via Bayes Rule**

$$p(\text{spam} | w_1, \dots, w_n) \propto p(\text{spam}) \prod_{i=1}^n p(w_i | \text{spam})$$

- **Parameter estimation**

Compute spam probability and word distributions for spam and ham

Naïve Bayes Spam Filter



Naïve Bayes Spam Filter

- Two classes (spam/ham)
- Binary features (e.g. presence of \$\$\$, viagra)
- Simplistic Algorithm
 - Count occurrences of feature for spam/ham
 - Count number of spam/ham mails

feature probability

$$p(x_i = \text{TRUE}|y) = \frac{n(i, y)}{n(y)} \text{ and } p(y) = \frac{n(y)}{n}$$

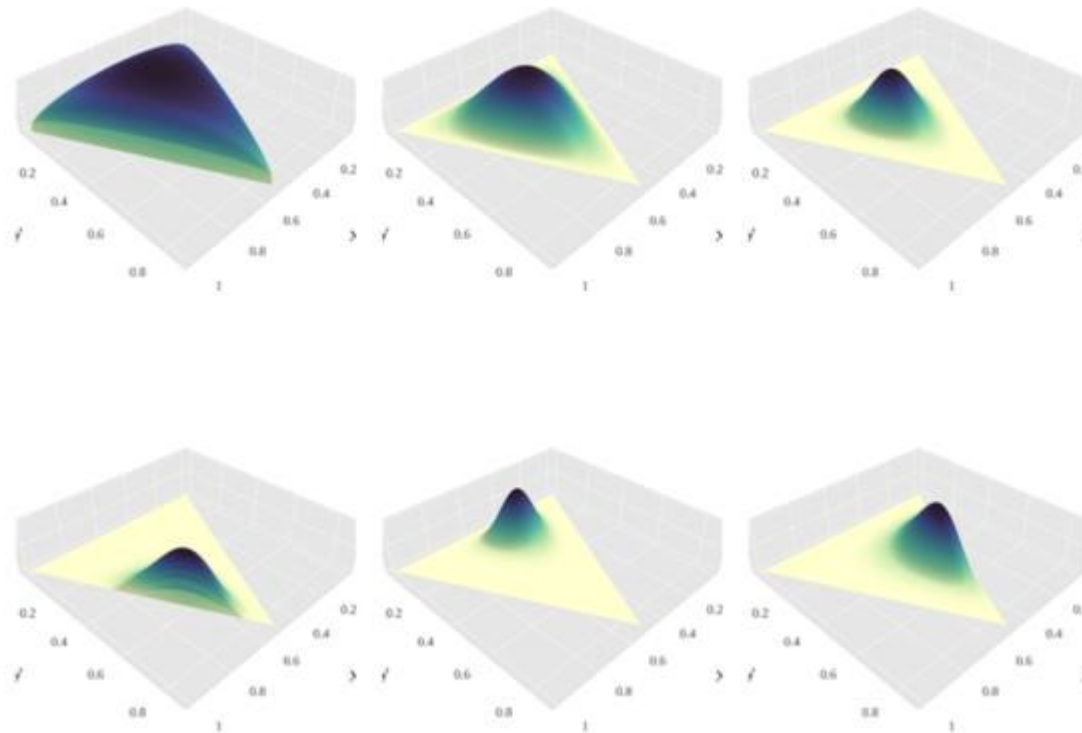
spam probability

$$p(y|x) \propto \frac{n(y)}{n} \prod_{i:x_i=\text{TRUE}} \frac{n(i, y)}{n(y)} \prod_{i:x_i=\text{FALSE}} \frac{n(y) - n(i, y)}{n(y)}$$

A Character-level Language Markov Model

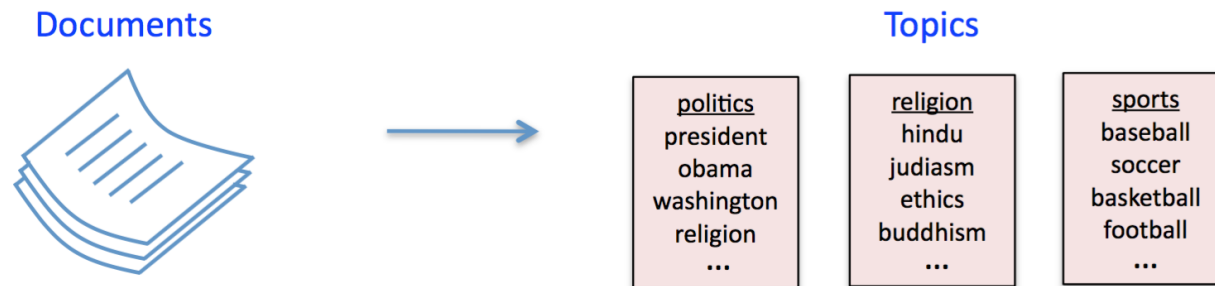
- Character-level language model allows you to generate new text
- It can be modeled using a maximum likelihood based method
- Pick a fixed order = 2
- For a training sequence, e.g., $\{h, e, l, l, o\}$
 - Compute $P(\{l\}|\{h, e\}) = \frac{\#\{l, h, e\}}{\#\{h, e\}}$
 - Do this for every three characters in the vocabulary
- Generate new text by sampling!

Aside: Dirichlet Distribution



Latent Dirichlet Allocation

- **Topic models** are powerful tools for exploring large data sets and for making inferences about the content of documents



- Many applications in information retrieval, document summarization, and classification



- LDA is one of the simplest and most widely used topic models

Latent Dirichlet Allocation

- 1 Sample the document's **topic distribution** θ (aka topic vector)

$$\theta \sim \text{Dirichlet}(\alpha_{1:T})$$

where the $\{\alpha_t\}_{t=1}^T$ are fixed hyperparameters. Thus θ is a distribution over T topics with mean $\theta_t = \alpha_t / \sum_{t'} \alpha_{t'}$

- 2 For $i = 1$ to N , sample the **topic** z_i of the i 'th word

$$z_i | \theta \sim \theta$$

- 3 ... and then sample the actual **word** w_i from the z_i 'th topic

$$w_i | z_i \sim \beta_{z_i}$$

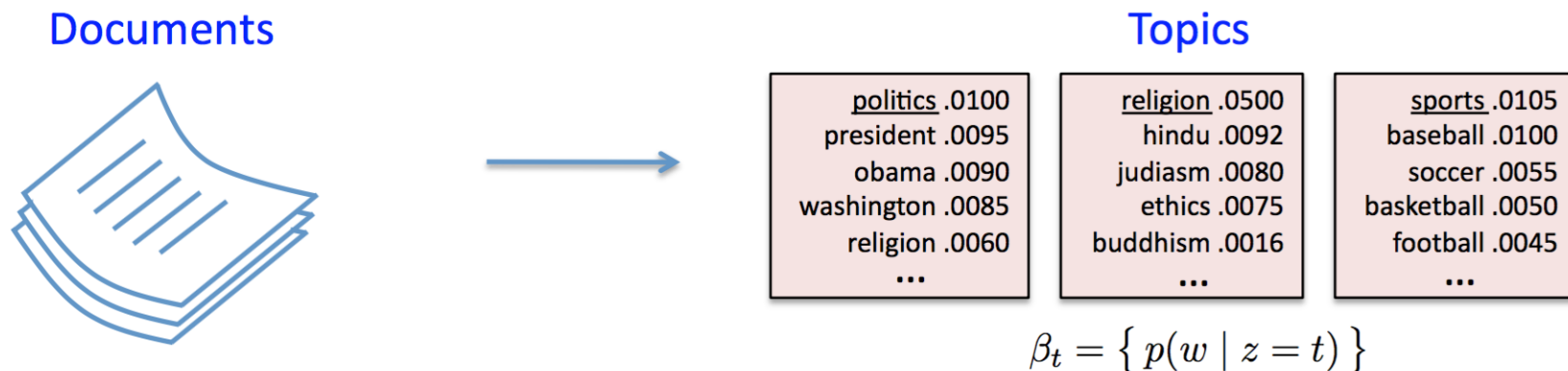
where $\{\beta_t\}_{t=1}^T$ are the *topics* (a fixed collection of distributions on words)

Latent Dirichlet Allocation

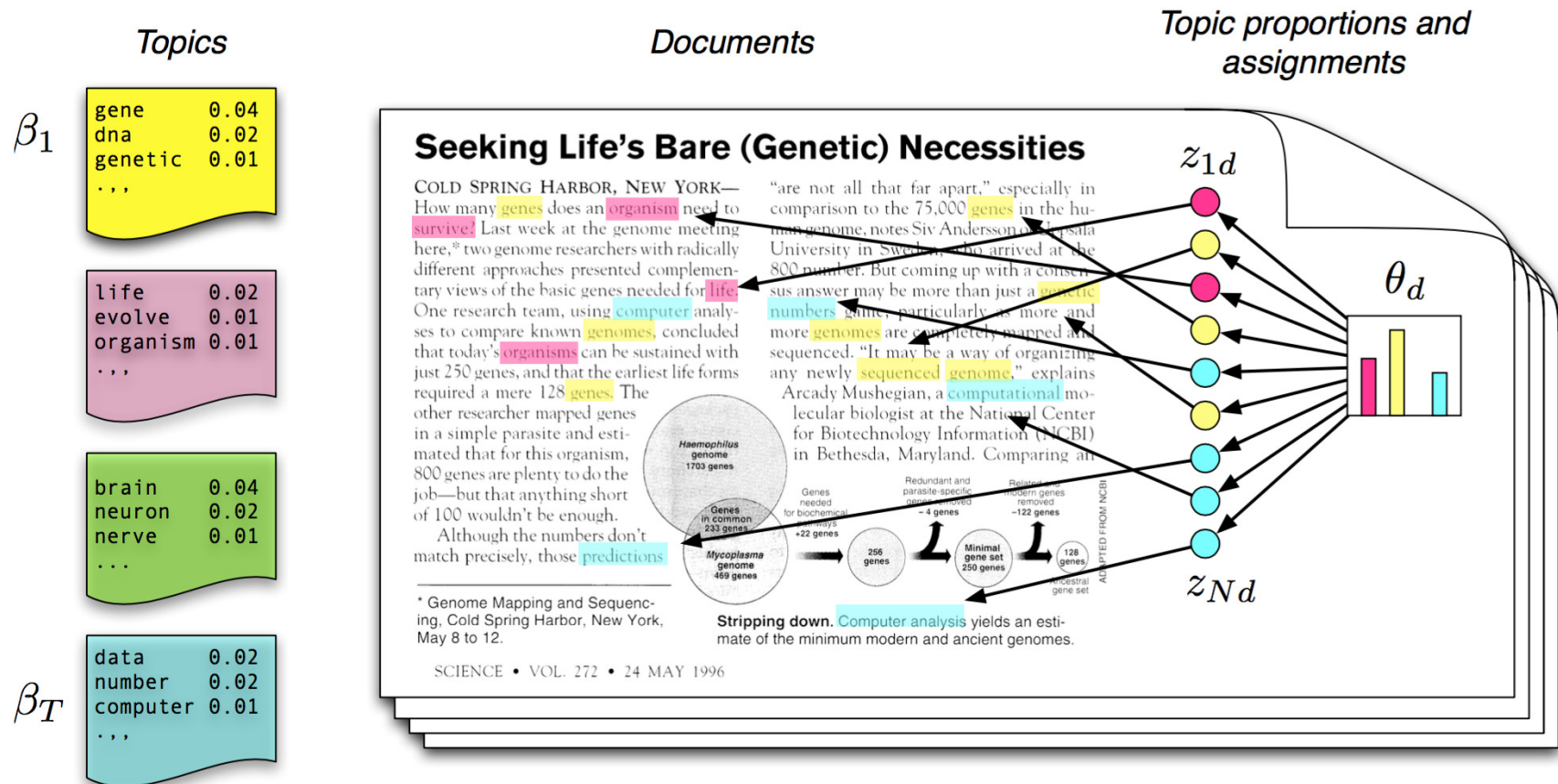
... and then sample the actual **word** w_i from the z_i 'th topic

$$w_i | z_i \sim \beta_{z_i}$$

where $\{\beta_t\}_{t=1}^T$ are the *topics* (a fixed collection of distributions on words)

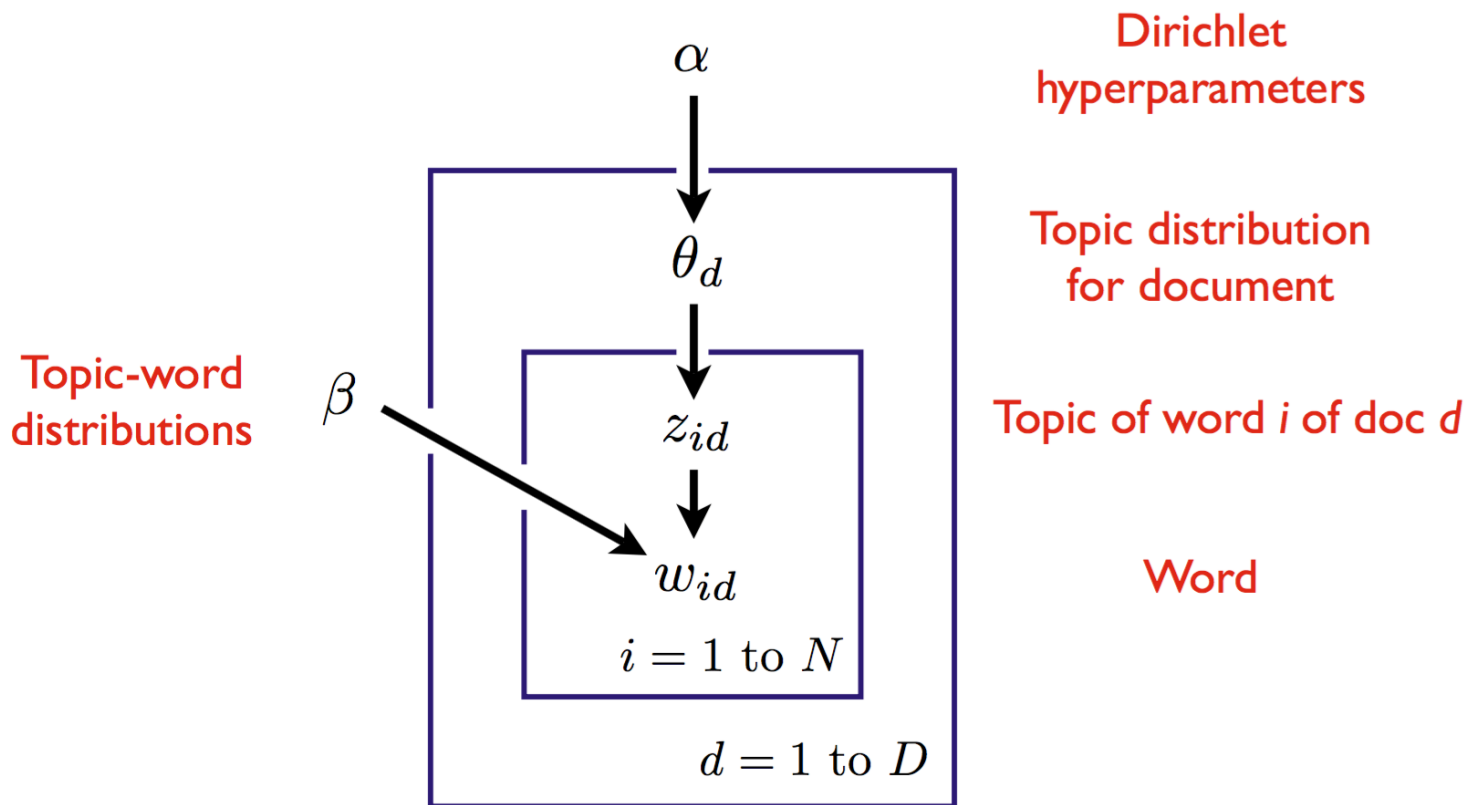


Latent Dirichlet Allocation



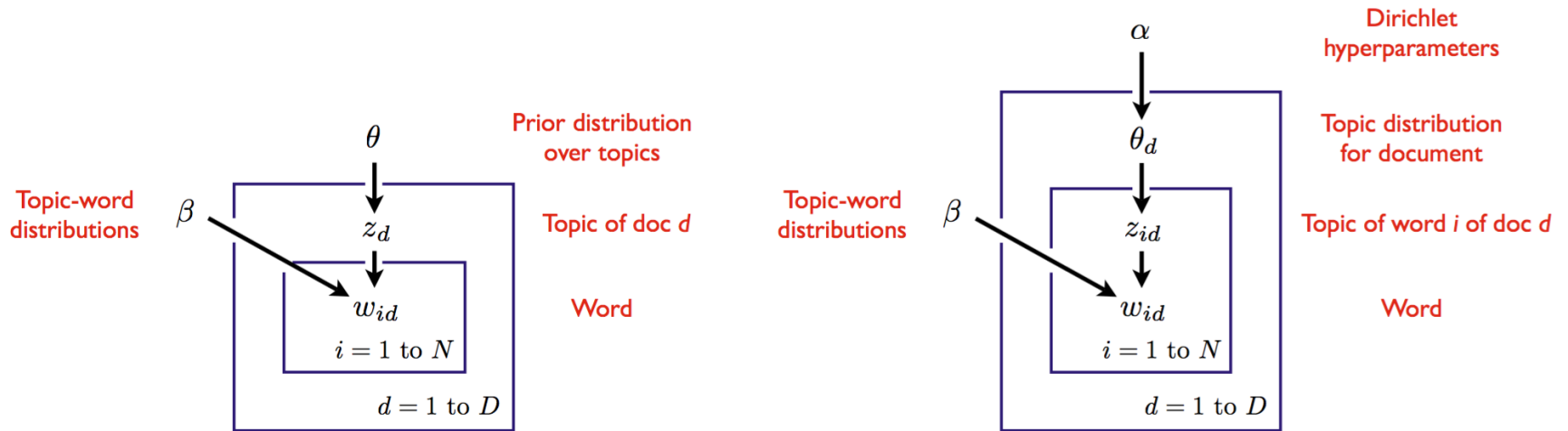
(Blei, *Introduction to Probabilistic Topic Models*, 2011)

Latent Dirichlet Allocation



Variables within a plate are replicated in a conditionally independent manner

Latent Dirichlet Allocation



- Model on left is a **mixture model**
 - Called *multinomial* naive Bayes (a word can appear multiple times)
 - Document is generated from a single topic
- Model on right (LDA) is an **admixture model**
 - Document is generated from a distribution over topics

Conditional Random Field based Classifier

- **Conditional random fields** are undirected graphical models of conditional distributions $p(\mathbf{Y} \mid \mathbf{X})$
 - \mathbf{Y} is a set of **target variables**
 - \mathbf{X} is a set of **observed variables**
- We typically show the graphical model using just the \mathbf{Y} variables
- Potentials are a function of \mathbf{X} and \mathbf{Y}

Conditional Random Field based Classifier

- A CRF is a Markov network on variables $\mathbf{X} \cup \mathbf{Y}$, which specifies the conditional distribution

$$P(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{c \in C} \phi_c(\mathbf{x}_c, \mathbf{y}_c)$$

with partition function

$$Z(\mathbf{x}) = \sum_{\hat{\mathbf{y}}} \prod_{c \in C} \phi_c(\mathbf{x}_c, \hat{\mathbf{y}}_c).$$

- As before, two variables in the graph are connected with an undirected edge if they appear together in the scope of some factor
- The only difference with a standard Markov network is the normalization term – before marginalized over \mathbf{X} and \mathbf{Y} , now only over \mathbf{Y}

CRF for NLP: Log-linear Terms

- Factors may depend on a large number of variables
- We typically parameterize each factor as a log-linear function,

$$\phi_c(\mathbf{x}_c, \mathbf{y}_c) = \exp\{\mathbf{w} \cdot \mathbf{f}_c(\mathbf{x}_c, \mathbf{y}_c)\}$$

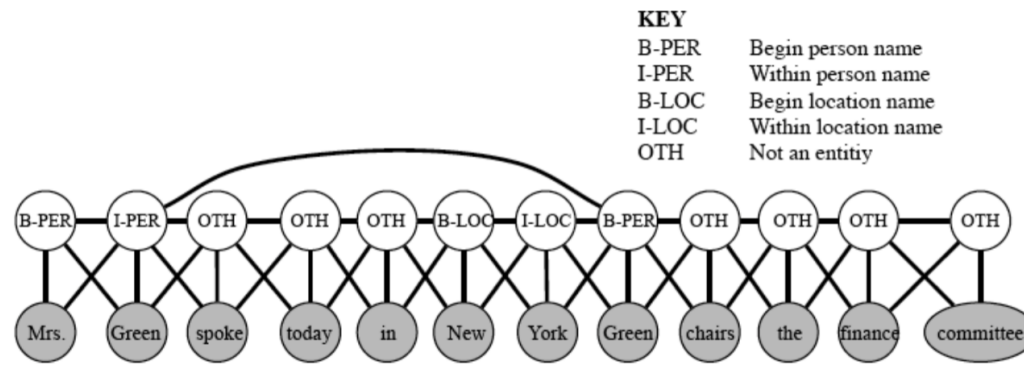
- $\mathbf{f}_c(\mathbf{x}_c, \mathbf{y}_c)$ is a feature vector
- \mathbf{w} is a weight vector which is typically learned – we will discuss this extensively in later lectures

CRF for NLP: The Task

- Given a sentence, determine the people and organizations involved and the relevant locations:
“Mrs. Green spoke today in New York. Green chairs the finance committee.”
- Entities sometimes span multiple words. Entity of a word not obvious without considering its *context*
- CRF has one variable X_i for each word, and Y_i encodes the possible labels of that word
- The labels are, for example, “B-person, I-person, B-location, I-location, B-organization, I-organization”
 - Having beginning (B) and within (I) allows the model to segment adjacent entities

CRF for NLP: The Task

The graphical model looks like (called a *skip-chain CRF*):



There are three types of potentials:

- $\phi^1(Y_t, Y_{t+1})$ represents dependencies between neighboring target variables [analogous to transition distribution in a HMM]
- $\phi^2(Y_t, Y_{t'})$ for all pairs t, t' such that $x_t = x_{t'}$, because if a word appears twice, it is likely to be the same entity
- $\phi^3(Y_t, X_1, \dots, X_T)$ for dependencies between an entity and the word sequence [e.g., may have features taking into consideration capitalization]

Notice that the graph structure changes depending on the sentence!

CNN based Sentence Classification

- Input is a sequence of words (variable)
- Output is a class label (fixed)

- Baseline 1:
 - Ignore sequence
 - Ignore semantic information
 - Treat input as a fixed length bag of words
 - This is a fixed size input and output classifier

CNN based Sentence Classification

- Baseline 2:
 - Weighted average of the word vectors as a vector for the sentence
 - Still loses word order
 - Retains some semantic information
 - Again, a fixed size input and output classifier

CNN based Sentence Classification

- Can also use Convolutional Neural Network!
 - For NLP, they became popular 2014
 - Less prominent currently due to other techniques
- Recall

1 _{x1}	1 _{x0}	1 _{x1}	0	0
0 _{x0}	1 _{x1}	1 _{x0}	1	0
0 _{x1}	0 _{x0}	1 _{x1}	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

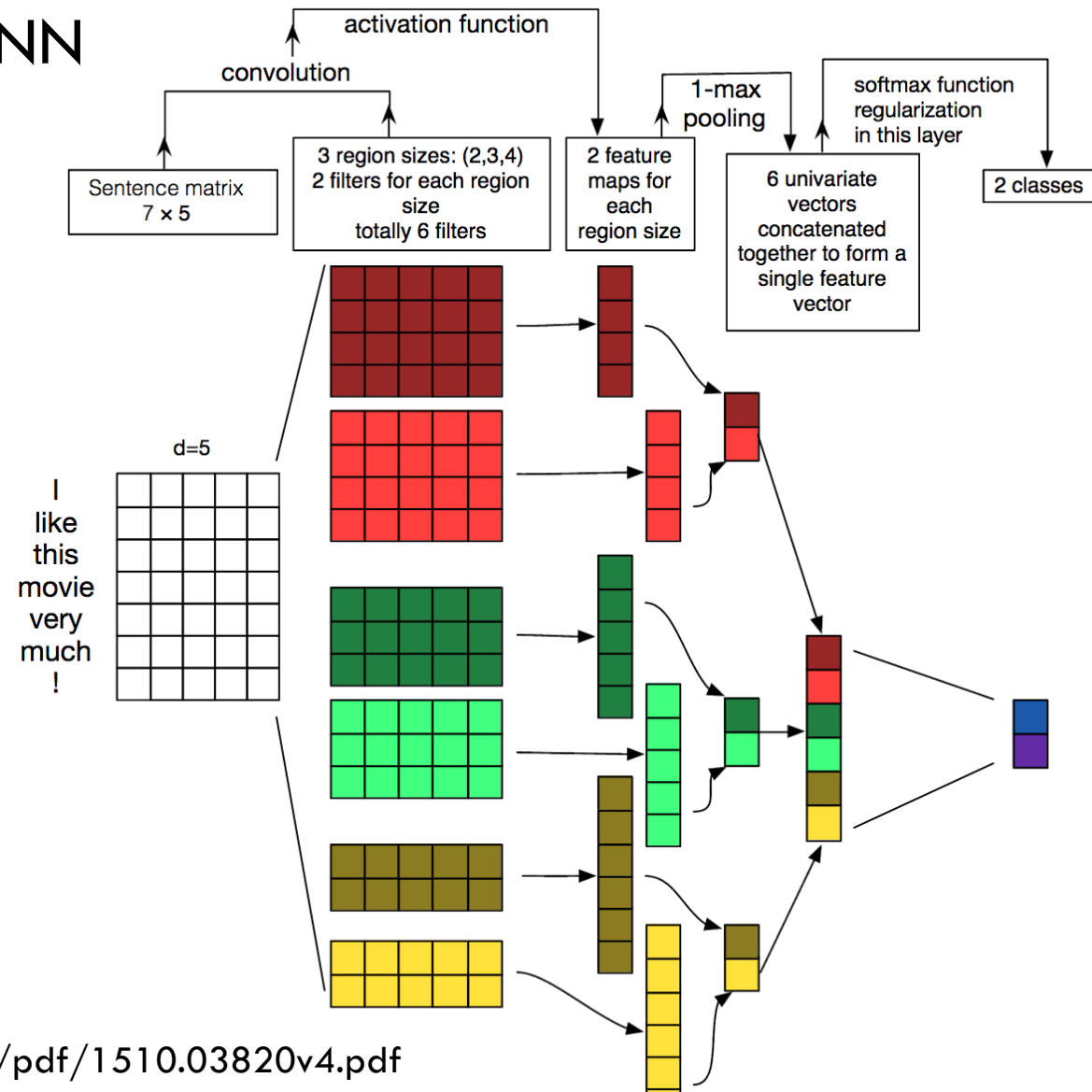
Convolved
Feature

Example I: Sentence Classification

- As you already know, CNNs capture
 - Location invariance
 - Example: In images, don't care where the 'cat' is in the input
 - Compositionality
 - Example: In images, lower level features to higher level patterns
- We will represent the sentence as a matrix
 - Each row for one word

Example I: Sentence Classification

- Example CNN



Embeddings

A Different Way of Dealing with Words

- Want semantically similar words to be represented similarly
 - This is the idea behind Vector Space Models in NLP
- Distributional Hypothesis (Firth 1957)
 - Words that appear in the same contexts share semantic meaning
- Two types of approaches:
 - Count based (PCA based)
 - Prediction based (creating auxiliary task etc)

Dealing with Words

- A word embedding $W: \text{words} \rightarrow \mathbb{R}^n$ is a function
 - Parametric
- Dimension n can be high: e.g., 300
- Example:
 - $W(\text{'university'}) = (0.3, -0.1, 2.0, 1.1, -1.5, \dots)$
 - $W(\text{'class'}) = (0.5, 1.1, -0.7, 2.5, 0.2, \dots)$

Learning an Embedding

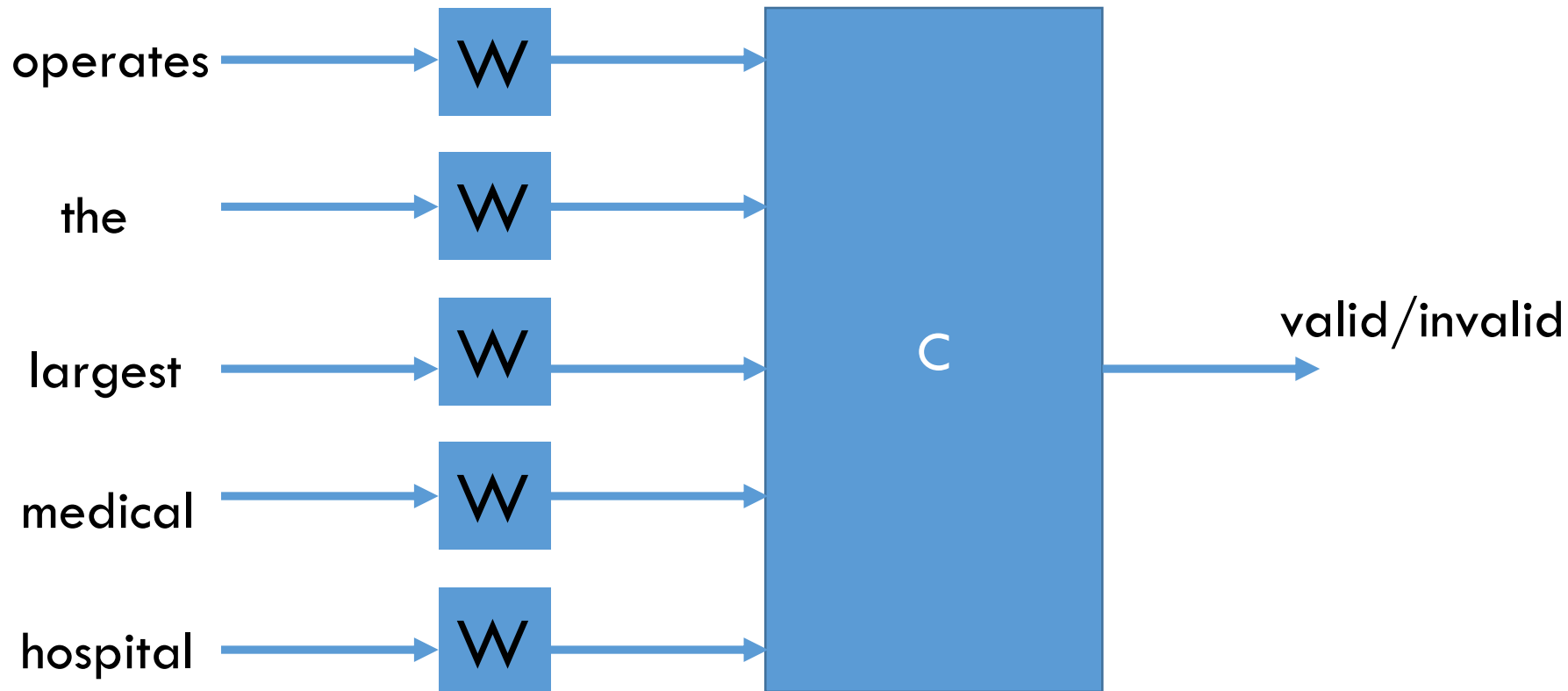
- How do we learn a good W ?
- Start with the same intuitive idea as before
 - Initialize such that W outputs random vectors for each word
 - Change the parameters such that the embedding vectors are **meaningful for a task**

Which Task? (I)

- Train a network to classify whether an input sequence of 5 words is valid or not
 - The input sequence is called an N-gram (5-gram)
- We get data, say from Wikipedia
 - Example: “operates the largest medical school”
- Break ‘half’ of them by replacing a word in each sequence with a random word
 - Example: “operates the **consistently** medical school”

Which Task? (II)

- Pass each word through W to get the vectors
- Pass the vectors through C (a classifier)



Which Task? (III)

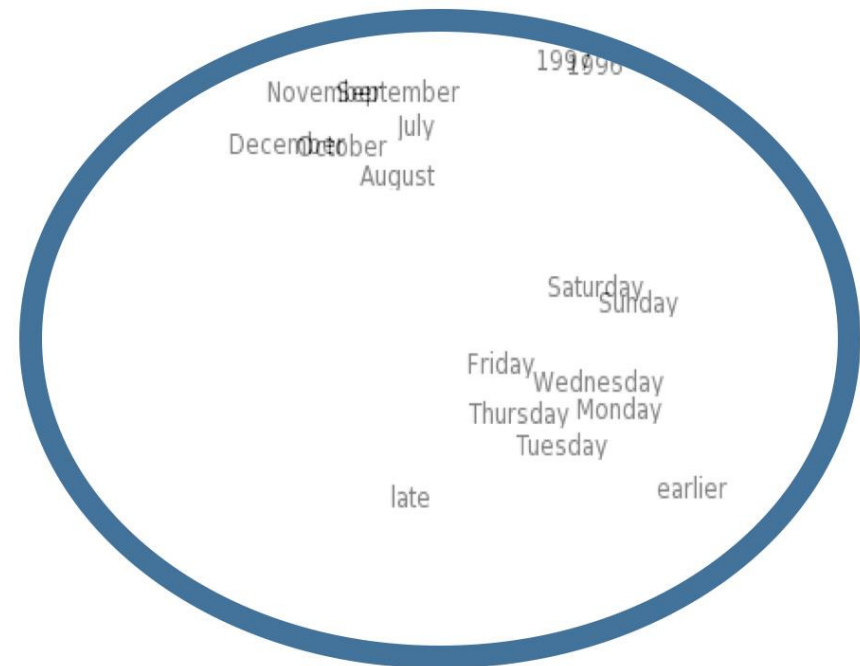
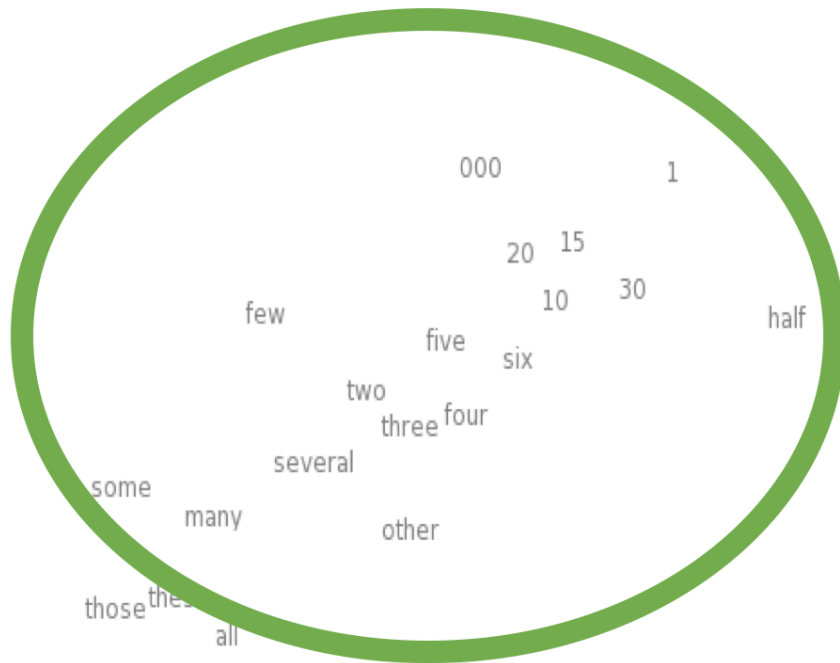
- In order to do the classification correctly, parameters for W and C should be good
- The task itself is **uninteresting** and **inconsequential**
 - We could have defined a different task
- Our objective is to learn a good W

Quality of Embedding (I)

- Say, we learned a good W
 - See Word2Vec or GloVe (for pretrained embeddings)
- How to visualize? Use t-SNE

Quality of Embedding (II)

- We see similar words are close together



Quality of Embedding (III)

- Look at words closest in the embedding to a given word
- 10 nearest neighbors are listed here

FRANCE 454	JESUS 1973	XBOX 6909	REDDISH 11724	SCRATCHED 29869	MEGABITS 87025
AUSTRIA	GOD	AMIGA	GREENISH	NAILED	OCTETS
BELGIUM	SATI	PLAYSTATION	BLUISH	SMASHED	MB/S
GERMANY	CHRIST	MSX	PINKISH	PUNCHED	BIT/S
ITALY	SATAN	IPOD	PURPLISH	POPPED	BAUD
GREECE	KALI	SEGA	BROWNISH	CRIMPED	CARATS
SWEDEN	INDRA	PSNUMBER	GREYISH	SCRAPED	KBIT/S
NORWAY	VISHNU	HD	GRAYISH	SCREWED	MEGAHERTZ
EUROPE	ANANDA	DREAMCAST	WHITISH	SECTIONED	MEGAPIXELS
HUNGARY	PARVATI	GEFORCE	SILVERY	SLASHED	GBIT/S
SWITZERLAND	GRACE	CAPCOM	YELLOWISH	RIPPED	AMPERES

An Attempt at Intuition (I)

- Is it natural for words with similar meanings to have similar vectors (hence nearest neighbors)?
- Example:
 - Change “operates the **largest** medical school” to “operates the **biggest** medical school”
 - If W maps **biggest** and **largest** close by
 - Then classifier C should still be able to work

An Attempt at Intuition (II)

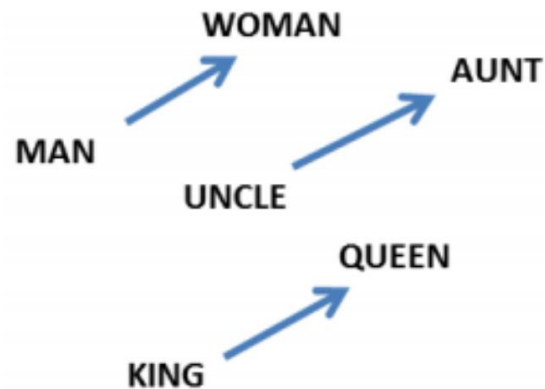
- Similar words getting mapped to close by vectors is great!
- We are not just limited to synonyms
- Example 1: “the inside wall is blue” to “the inside wall is red”
- Example 2: “the inside wall is blue” to “the inside ceiling is red”

How Much Data?

- Clearly, we have to see all words (for whom we need embeddings)
- But we need not see their combinations
- Analogies allow us to generalize to new combination of words
- This is similar to humans: we have seen all words but have not seen all sentences with those words

Difference of Vectors Property (I)

- Many word embeddings exhibit the following property as a **side-effect**:
 - Analogies are encoded in **difference vectors**



$$W(\text{"woman"}) - W(\text{"man"}) \simeq W(\text{"aunt"}) - W(\text{"uncle"})$$

$$W(\text{"woman"}) - W(\text{"man"}) \simeq W(\text{"queen"}) - W(\text{"king"})$$

Difference of Vectors Property (II)

- For a pair of words, subtract their difference and add to another word. For example,
 - $W(\text{"France"}) - W(\text{"Paris"}) + W(\text{"Rome"}) \approx W(\text{"Italy"})$

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

Use of Embeddings (I)

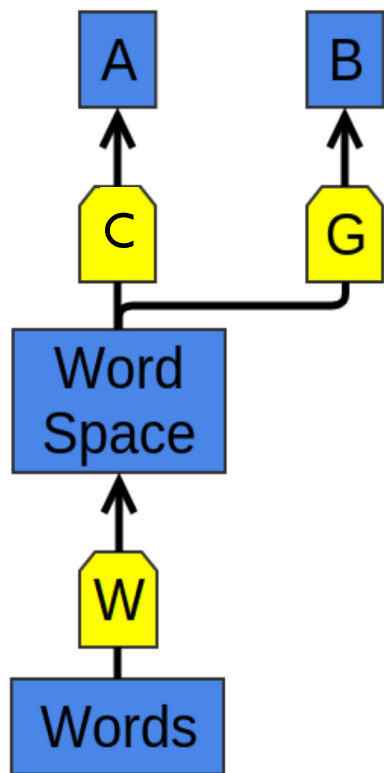
- Embeddings represent **unstructured** data automatically in such a way that a subsequent task's performance is good
 - We have already seen **image embeddings**
 - Here, we are seeing **word embeddings**

Use of Embeddings (I)

- Embeddings represent **unstructured** data automatically in such a way that a subsequent task's performance is good
 - We have already seen **image embeddings**
 - Here, we are seeing **word embeddings**
- Once a word embedding W is learned, we can use it for many other NLP (Natural Language Processing) tasks
 - **Transfer learning** (just like for images!)

Use of Embeddings (II)

- Learn a good representation (i.e., W) on some task and use it for other tasks



“

The use of word representations... has become a key “secret sauce” for the success of many NLP systems in recent years, across tasks including named entity recognition, part-of-speech tagging, parsing, and semantic role labeling. (Luong et al. (2013))

W and F learn to perform task A. Later, G can learn to perform B based on W .

Use of Embeddings (III)

- Key benefit of W
 - Can train using more than one kind of data
- Thus, we can learn a way to **map multiple kinds of data into a single representation!**
- Example: Bilingual word embedding¹
 - English words
 - Mandarin words
 - Embed both words in the **same space**

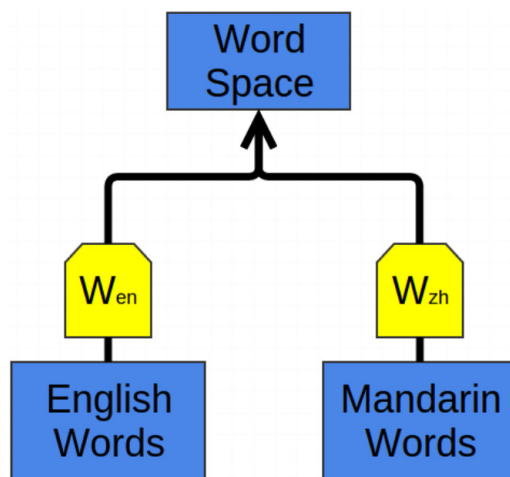
¹Reference: http://ai.stanford.edu/~wzou/emnlp2013_ZouSocherCerManning.pdf

Bilingual Word Embedding (I)

- Train W_{en} and W_{zh} simultaneously
 - Impose the following: words that we know are close translations should be **close** together
 - Example:

$$W_{en}(\text{'university'}) = (0.3, -0.1, 2.0, 1.1, -1.5, \dots)$$

$$W_{zh}(\text{'大学'}) = (0.2, -0.1, 2.2, 1.0, -1.4, \dots)$$



¹Pronunciation of 大学: Dàxué

²Figure: <http://colah.github.io/posts/2014-07-NLP-RNNs-Representations/>

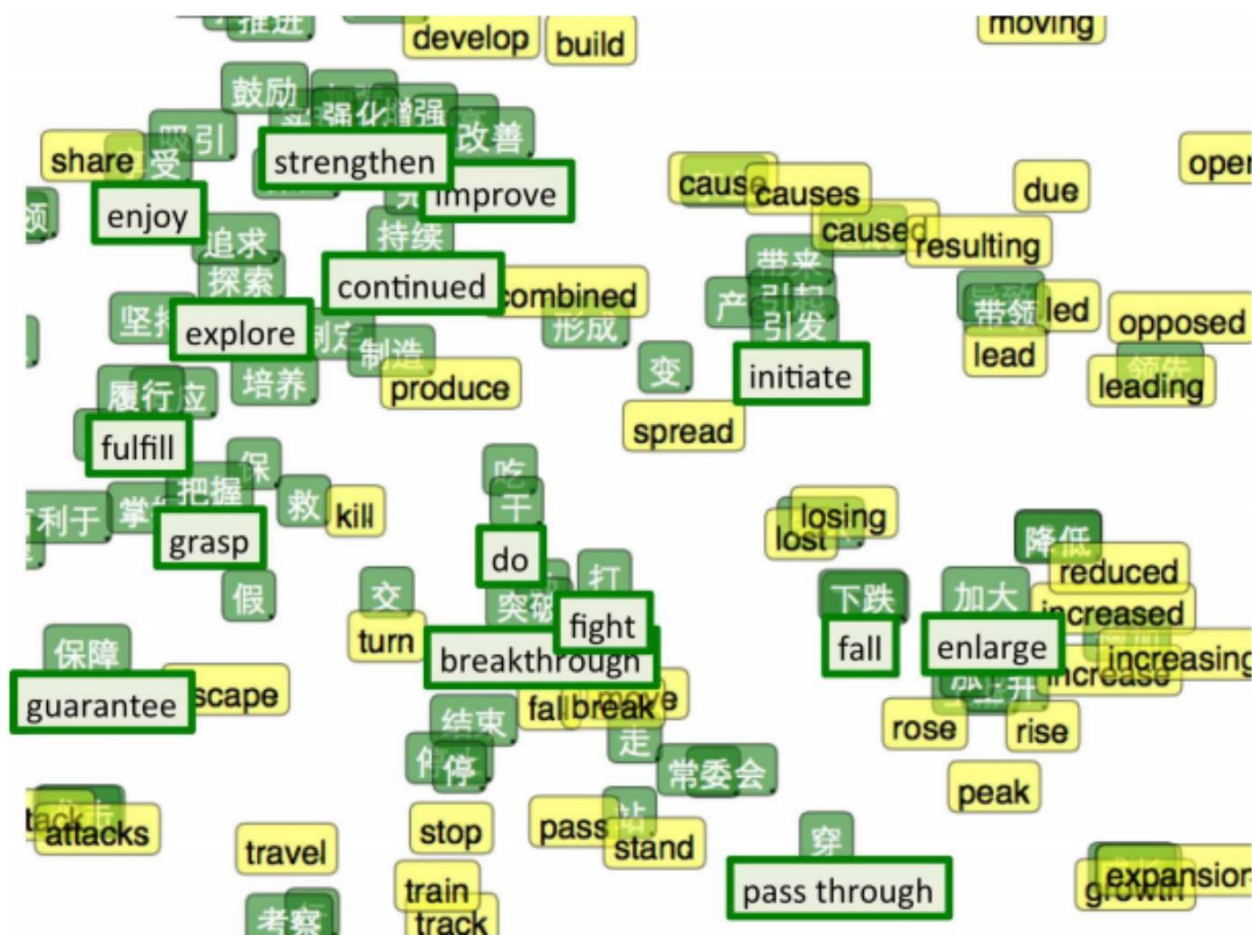
Bilingual Word Embedding (II)

- After training W_{en} and W_{zh} we observe:
 - Words that we didn't know were translations end up close together
- Example:
 - We did not know 商业 and business are translations. Still we get:

$$W_{en}(\text{'business'}) = (0.7, -0.4, 1.0, 1.8, -0.8, \dots)$$

$$W_{zh}(\text{'商业'}) = (0.8, -0.3, 0.9, 2.0, -0.9, \dots)$$

Bilingual Word Embedding (III)



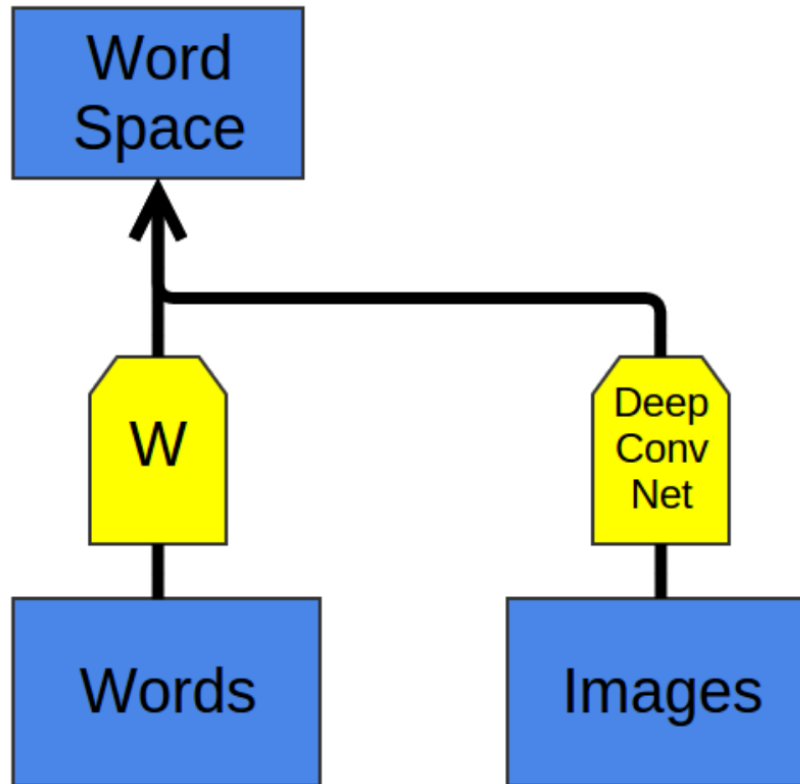
t-SNE visualization of the bilingual word embedding. Green is Chinese, Yellow is English.

General Shared Embeddings

- We can also embed **very** different kinds of data into the same space
- Example:
 - Images and words
 - Map the image of an **object** near the **object** word vector
 - Map the image of a **dog** near the **dog** word vector

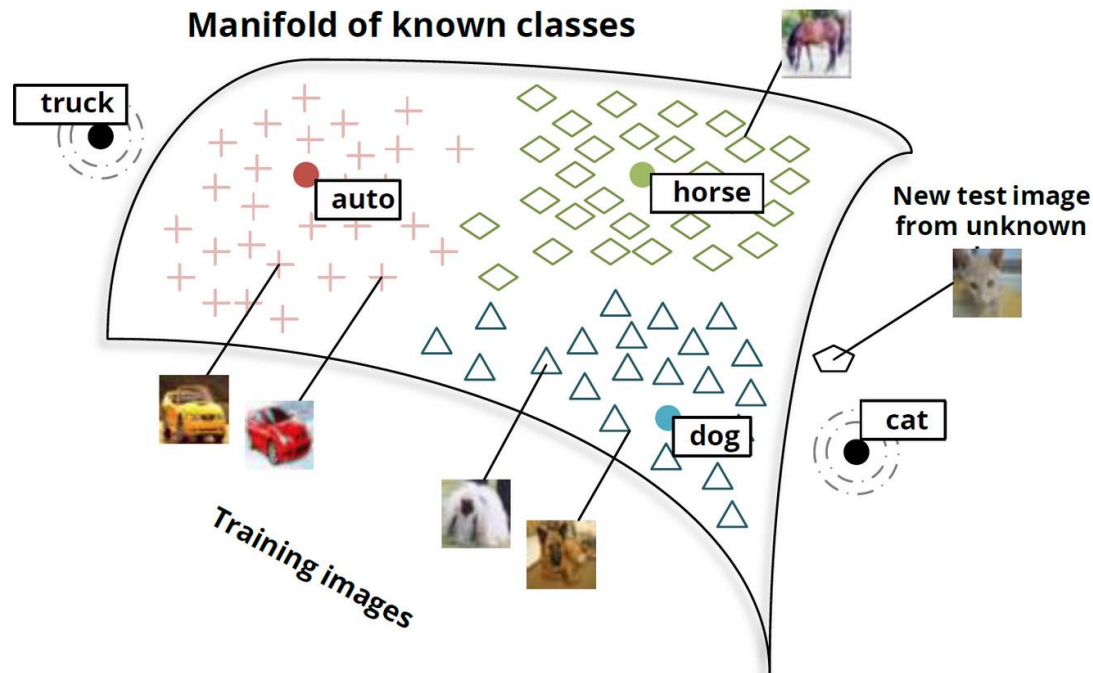
General Shared Embeddings

- Essentially the output of the image classifier is not a score vector but a vector in the range(W)



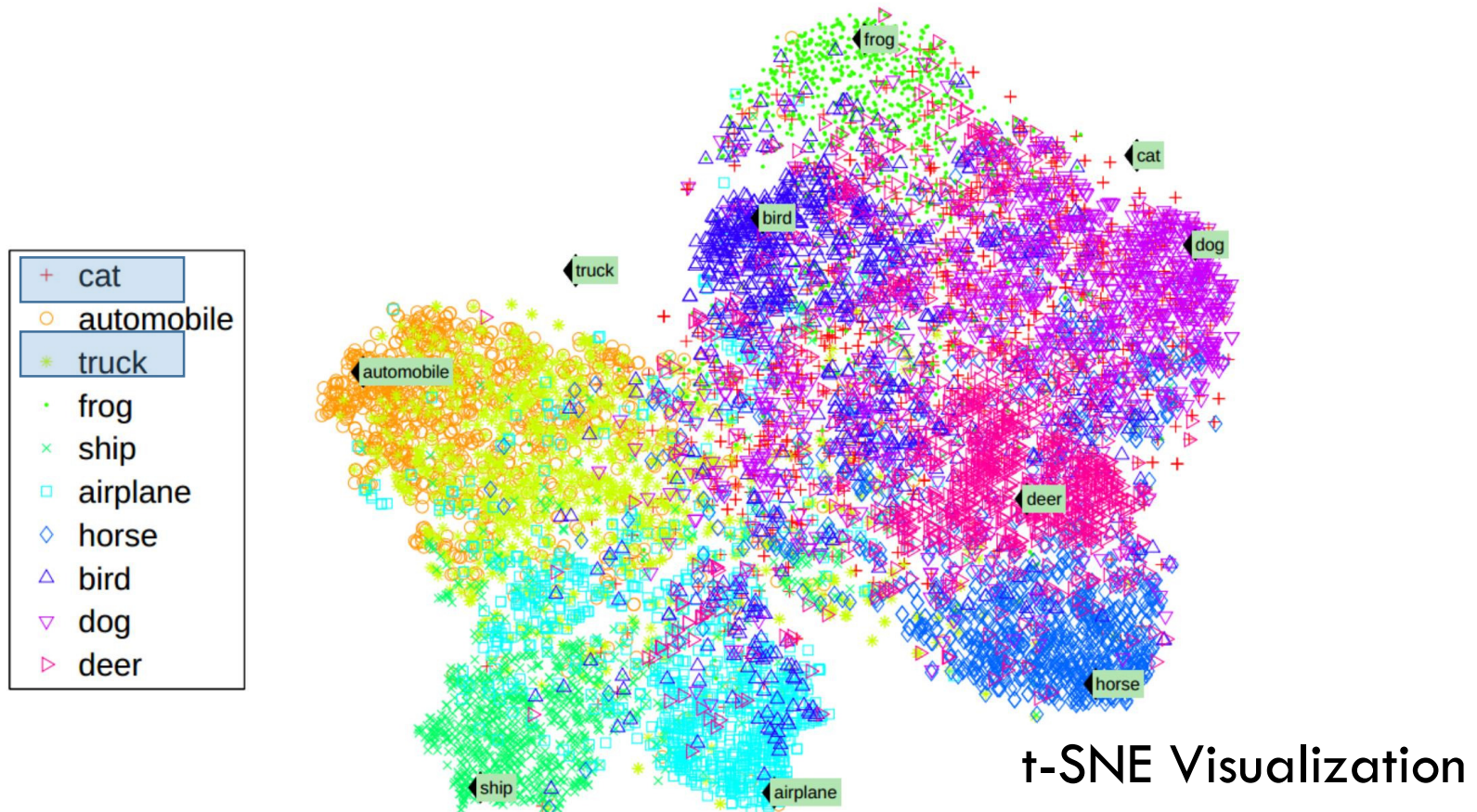
General Shared Embeddings

- When we test the model on **new** classes of images
 - Note: new means *not seen in training*
- For example, we didn't have images of cats



General Shared Embeddings

- Images of cats are mapped to regions where dog vectors are!*



Questions?

Word2Vec In Detail

Word2Vector

- A technique proposed by Google in 2013
- Is a predictive method rather than a count based method
- Objective: Vector representations of words that capture their co-occurrence statistics

Word2Vector: Two Versions

- Continuous Bag of Words and Skip-Gram
- Lets go through the skip-gram model in some detail now

W2V: The Skip-Gram Version

- This is a very simple neural network model to learn W
- We will train a single hidden layer NN to perform an auxiliary task
- The goal will be to just learn the weights of the network
 - This will give us W

W2V: The Auxiliary Task

- Task:
 - Pick a word in the middle of a sentence
 - Pick one of the **nearby** words at random
 - Make network learn probability of every word in our vocab of being this nearby word
- Input: a word pair (one hot encoded)
- Output: normalized scores (of length: vocab size)
- Meaning of 'nearby':
 - Essentially defined using a window size
 - Example: Window size 2 means 2 words to left and 2 to right of the input word are nearby

W2V: The Auxiliary Task

- Feed word pairs
- Example: “The quick brown fox jumps over the lazy dog”

Source Text	Training Samples					
<table><tr><td>The</td><td>quick</td><td>brown</td></tr></table> fox jumps over the lazy dog. ➡	The	quick	brown	(the, quick) (the, brown)		
The	quick	brown				
The <table><tr><td>quick</td><td>brown</td><td>fox</td></tr></table> jumps over the lazy dog. ➡	quick	brown	fox	(quick, the) (quick, brown) (quick, fox)		
quick	brown	fox				
The quick <table><tr><td>brown</td><td>fox</td><td>jumps</td></tr></table> over the lazy dog. ➡	brown	fox	jumps	(brown, the) (brown, quick) (brown, fox) (brown, jumps)		
brown	fox	jumps				
The <table><tr><td>quick</td><td>brown</td><td>fox</td><td>jumps</td><td>over</td></tr></table> the lazy dog. ➡	quick	brown	fox	jumps	over	(fox, quick) (fox, brown) (fox, jumps) (fox, over)
quick	brown	fox	jumps	over		

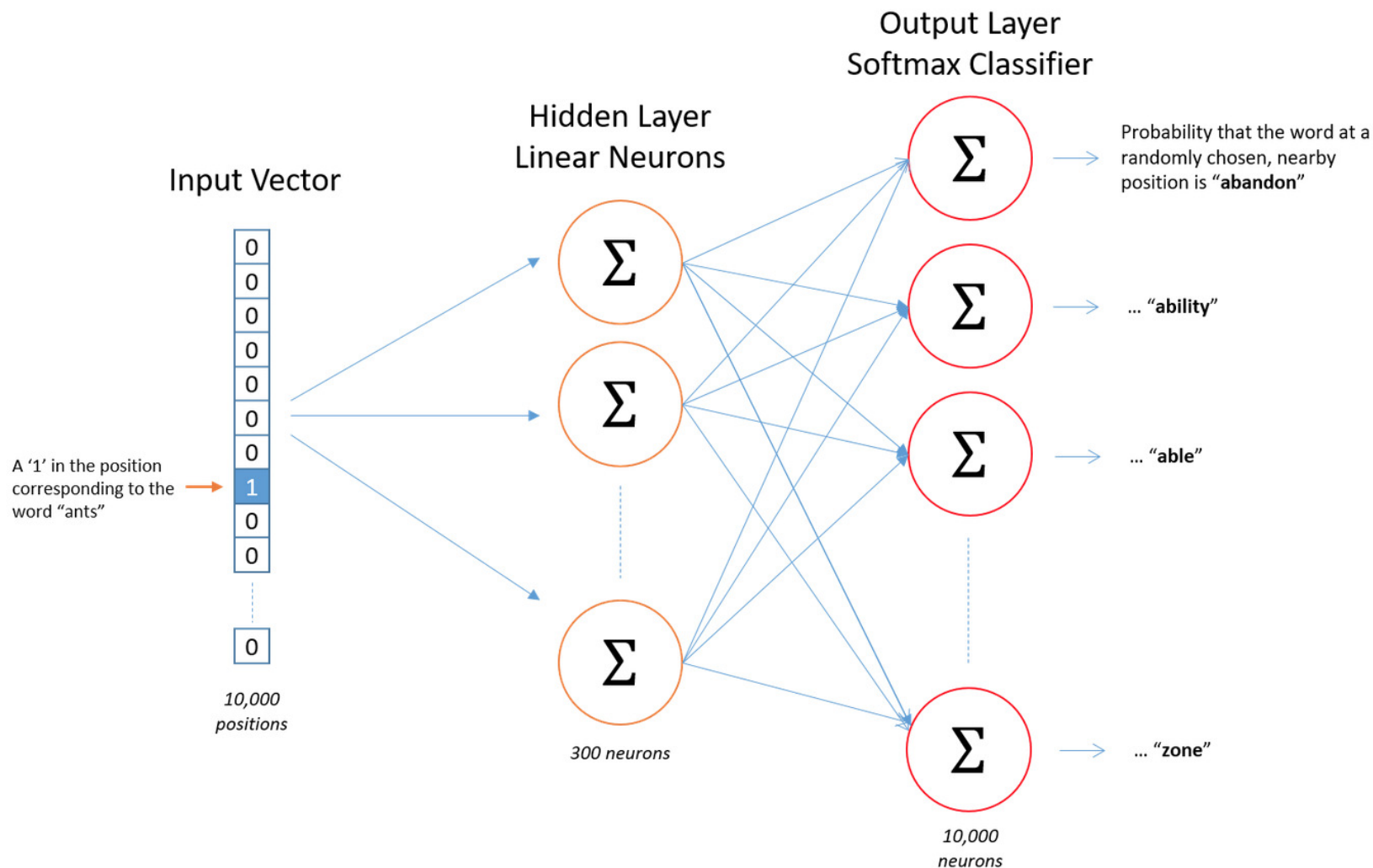
mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

W2V: The Network

- Say we have 10000 words in our vocab
- Then the input word is 10000 dimensional vector
 - Example: Cat word will have 'Cat' coordinate 1, everything else 0
- The true label (word) is also 10000 dimensional vector
- Network outputs 10000 scores which pass through softmax
 - Each coordinate is the probability that a particular word is the randomly selected nearby word

W2V: The Network

- Notice: No nonlinearity in the hidden layer!



W2V: The Objective

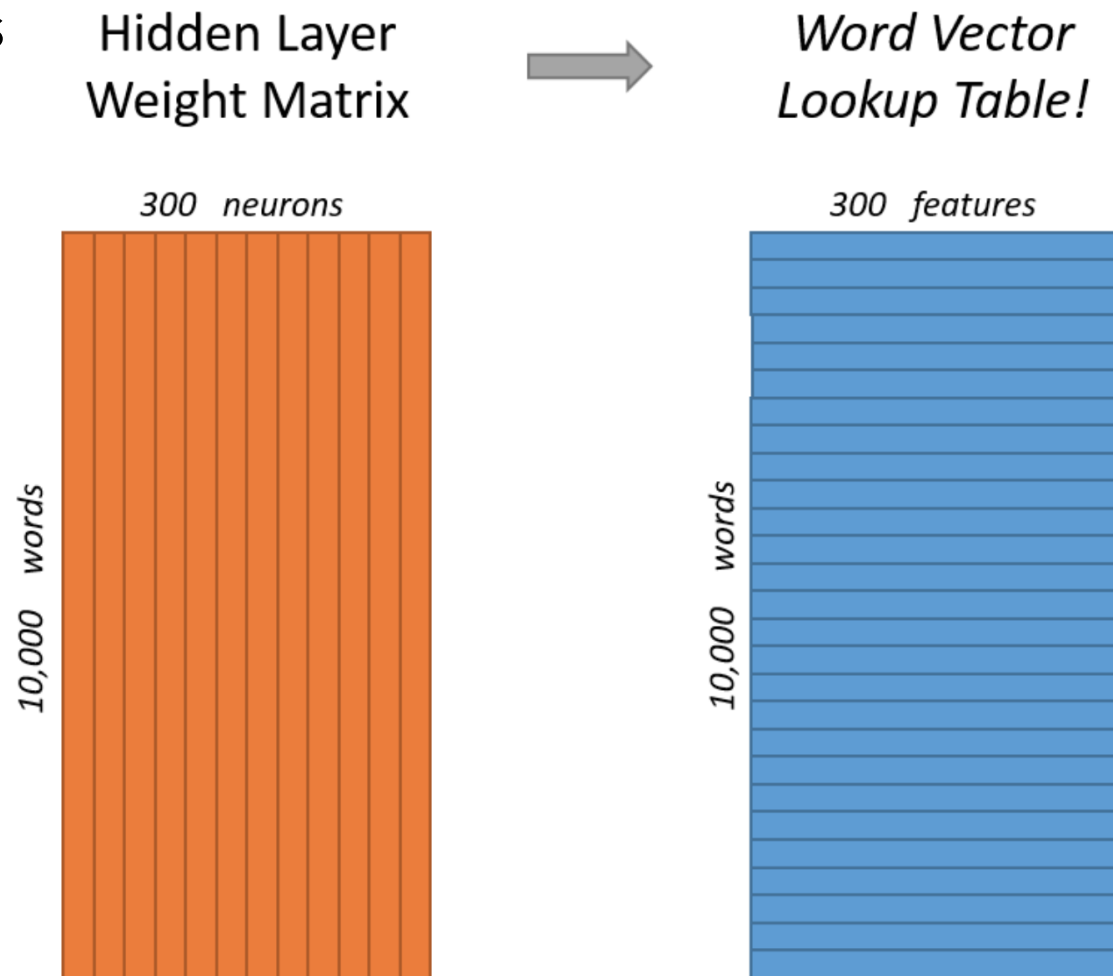
- The objective is to maximize the normalized score (recall normalization by softmax operation) of the correct context word
- Say our training data is made with T words, each having a context window size 2
 - That is, each word is associated with 4 other words
 - Total training data is $4T$
- The objective is $\frac{1}{T} \sum_{t=1}^T \sum_{j \in \{-2, -1, 1, 2\}} \log p(w_{t+j} | w_t)$

W2V: The Hidden Layer

- Is represented by a weight matrix W_0
 - Lets represent it by its transpose (just for convenience)
 - $h^T = x^T W_0^T = x^T W$ for each example
 - Number of rows of W is 10000
 - Number of columns of W is 300
- Then the rows of W are our word vectors!

W2V: The Hidden Layer

- Our real goal was just to learn the hidden layer weights



W2V: The Lookup

- Say word 'Cat' has coordinate c for some $c \in \{1, \dots, 10,000\}$
- If we multiply the 1×10000 dim one hot vector for the word 'Cat' with W
- It will just select the c^{th} row of W
 - The output of the hidden layer is the word vector!

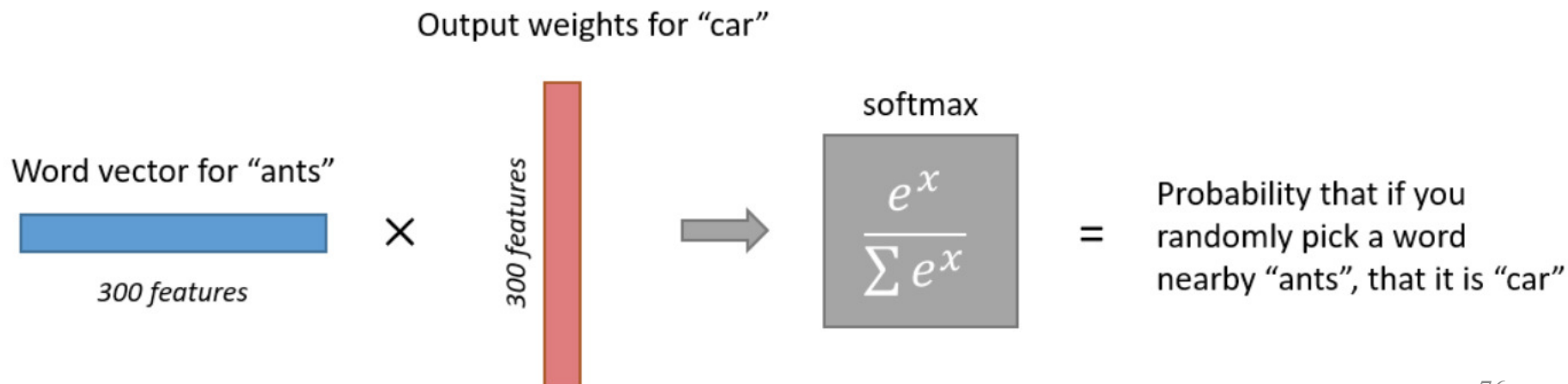
W2V: The Lookup

- Say word 'Cat' has coordinate c for some $c \in \{1, \dots, 10,000\}$
- If we multiply the 1×10000 dim one hot vector for the word 'Cat' with W
- It will just select the c^{th} row of W
 - The output of the hidden layer is the word vector!
- Example visualization

$$[0 \quad 0 \quad 0 \quad 1 \quad 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = [10 \quad 12 \quad 19]$$

W2V: Auxiliary Task Again

- Output of the network is a bunch of normalized scores (i.e., probabilities)
 - Denote the probability that the this word is a nearby word
- Example:
 - Pick the word vector for 'ants'
 - Pic the output neuron for word 'car'



W2V: Intuition for Vectors

- If two different words have similar “contexts”
 - Words that are likely to appear around them
 - Then the output probability vector should be similar
- For output vector to be similar
 - The word vector (weights of hidden layer) should be similar
 - Since the inputs are 1-hamming distance apart always

W2V: Intuition for Vectors

- Word2Vec is capturing nothing but the co-occurrence statistics!
- Example:
 - Words like 'university' and 'masters' would have similar contexts, hence similar word vectors
- This will also handle stemming!
 - Example: words like 'car' and 'cars' will have similar vectors because contexts would be similar

W2V: Practice

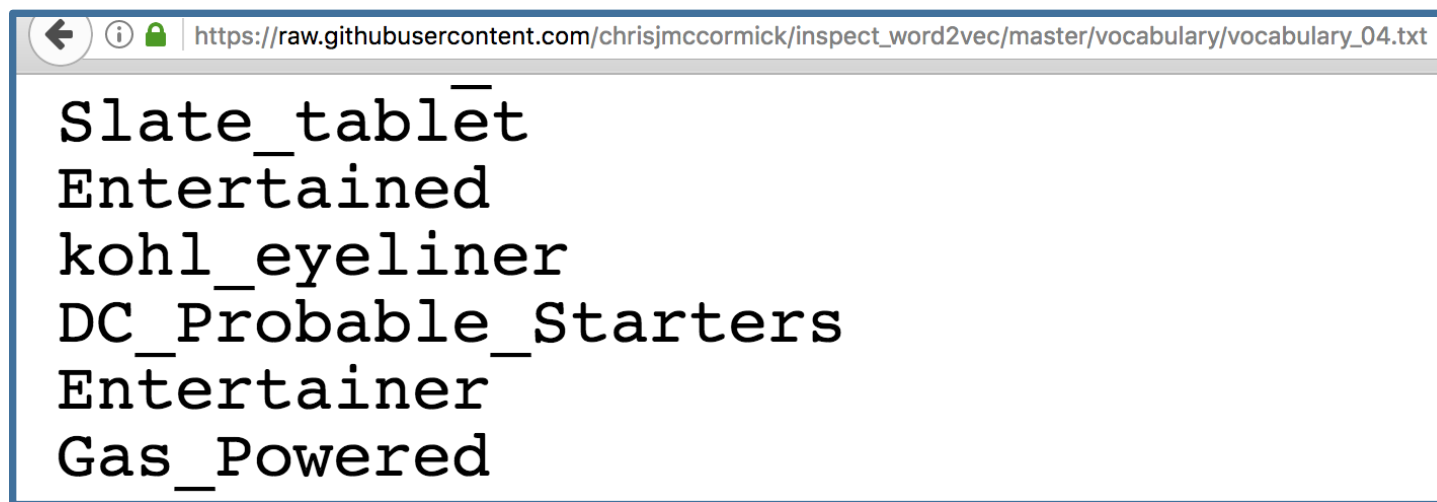
- The network is relatively large
 - Two weight matrices
 - 300×10000 parameters each
- Need a lot of data to train
- And engineering tricks are needed to deal with data

W2V: Engineering Tricks

- Subsample frequent words
 - Example: Too many pairs like ('the',...). So delete them proportional to how frequent they are
- Treat common phrases as single 'words'
- Optimization trick: negative sampling
 - Only update the weights of neurons corresponding to a few (5-20) non-nearby words
 - These few are sampled inversely proportional to their frequency

W2V: From Google

- 100 Billion words from the Google News Dataset
- Vocab size totals about 3 million!

A screenshot of a web browser window displaying a raw GitHub file. The address bar shows the URL: https://raw.githubusercontent.com/chrisjmccormick/inspect_word2vec/master/vocabulary/vocabulary_04.txt. The main content area shows a list of words in a monospaced font, each on a new line: Slate_tablet, Entertained, kohl_eyeliner, DC_Probable_Starters, Entertainer, and Gas_Powered. The words are left-aligned and appear to be part of a larger dataset.

Slate_tablet
Entertained
kohl_eyeliner
DC_Probable_Starters
Entertainer
Gas_Powered

¹Reference: https://github.com/chrisjmccormick/inspect_word2vec/tree/master/vocabulary

²word2Vec file: <https://drive.google.com/file/d/0B7XkCwpl5KDYNINUTTISS21pQmM/edit>

W2V: From Google

- 100 Billion words from the Google News Dataset
- Vocab size totals about 3 million!

```
In [1]: import gensim
```

```
In [3]: model = gensim.models.Word2Vec.load_word2vec_format('./GoogleNews-vectors-negative300.bin.gz', binary=True)
```

```
In [7]: model.most_similar('good')
```

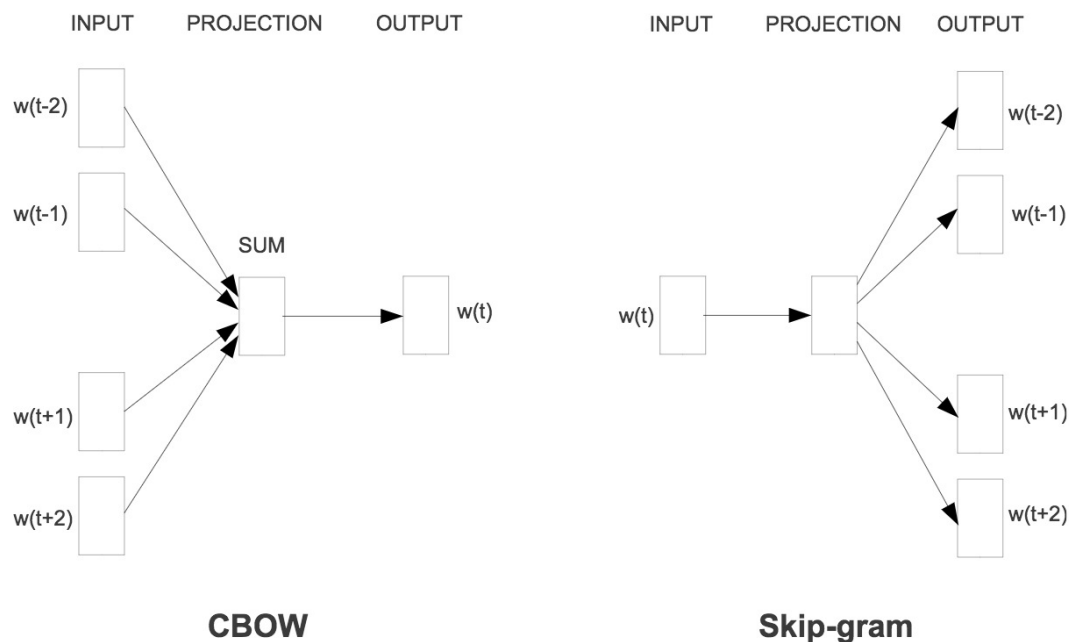
```
Out[7]: [(u'great', 0.7291509509086609),  
         (u'bad', 0.7190051078796387),  
         (u'terrific', 0.6889115571975708),  
         (u'decent', 0.6837348341941833),  
         (u'nice', 0.6836091876029968),  
         (u'excellent', 0.6442928910255432),  
         (u'fantastic', 0.6407778859138489),  
         (u'better', 0.6120729446411133),  
         (u'solid', 0.5806034803390503),  
         (u'lousy', 0.5764203071594238)]
```

¹Reference: https://github.com/chrisjmccormick/inspect_word2vec/tree/master/vocabulary

²word2Vec file: <https://drive.google.com/file/d/0B7XkCwpl5KDYNINUTTISS21pQmM/edit>

W2V: The CBOW Version

- Continuous Bag of Words (CBOW) version of Word2Vec
 - Essentially, a slightly different prediction task



- There is another popular embedding called GLoVe

¹Figure: <https://arxiv.org/pdf/1301.3781.pdf>

²Reference for GLoVe: <http://nlp.stanford.edu/projects/glove/>

Word2Vec Example Code

- For an implementation in Python see
 - See https://github.com/tensorflow/tensorflow/blob/master/tensorflow/examples/tutorials/word2vec/word2vec_basic.py
- Many other pretrained embeddings are also available
 - See <https://github.com/3Top/word2vec-api>

NLP Ecosystem in Python

- There are many tools to choose from
 - Gensim, NLTK, SpaCy, TextBlob, Pattern
- Also, there are many traditional NLP tasks and techniques that may be helpful to know about:
 - These include tokenizing, stop words, stemming, Parts-of-speech tagging, chunking and chinking, Named Entity Recognition, lemmatizing and knowing the wordnet ecosystem among others.

Questions?

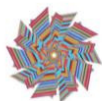
Summary

- Text processing is very useful in multiple applications
- We saw some models that did not need to understand word meanings
 - Naïve bayes, Markov assumption based, CRF, LDA
- The notion of embedding words (or characters or phrases ...) is useful and such embeddings can be learned
- We saw how Word2Vec embeddings were created

Appendix

LDA: More Intuition (different notation)

Question: given a text document, what topics is it about?



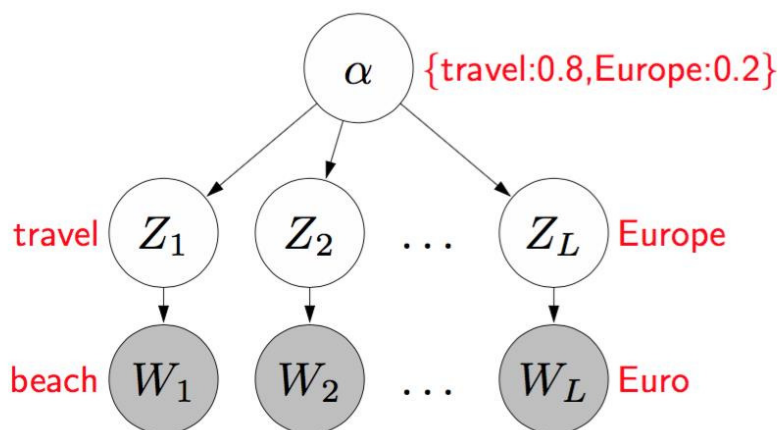
Probabilistic program: latent Dirichlet allocation

Generate a distribution over topics $\alpha \in \mathbb{R}^K$

For each position $i = 1, \dots, L$:

Generate a topic $Z_i \sim p(Z_i \mid \alpha)$

Generate a word $W_i \sim p(W_i \mid Z_i)$



Advanced Prediction Models

Deep Learning, Graphical Models and Reinforcement
Learning

Today's Outline

- Recurrent Neural Networks
- Long-Short Term Memory based RNNs
- Sequence to Sequence Learning and other RNN Applications

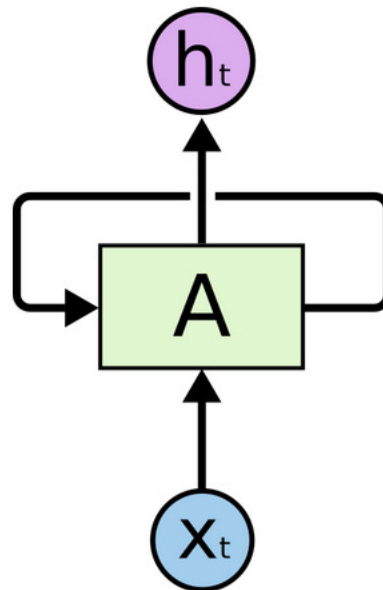
Recurrent Neural Network

The Idea of Persistence (I)

- Our thoughts have persistence
- We understand the present given what we have seen in the past
- Feedforward neural networks and CNNs don't have persistence
 - Example:
 - classify every scene in a movie
 - Output size (number of classes) is fixed
 - Number of layers is fixed
 - Unclear how a CNN can use information from previous scenes

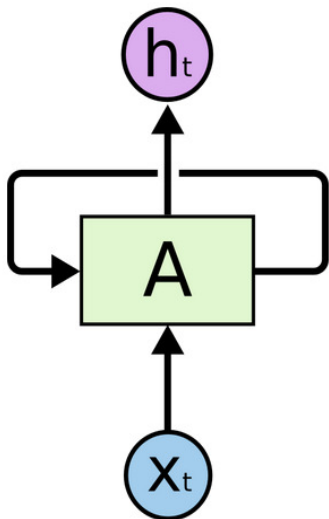
The Idea of Persistence (II)

- Architectures called Recurrent Neural Networks address the idea of persistence
- They are networks with ‘**loops**’ that try to persist past information



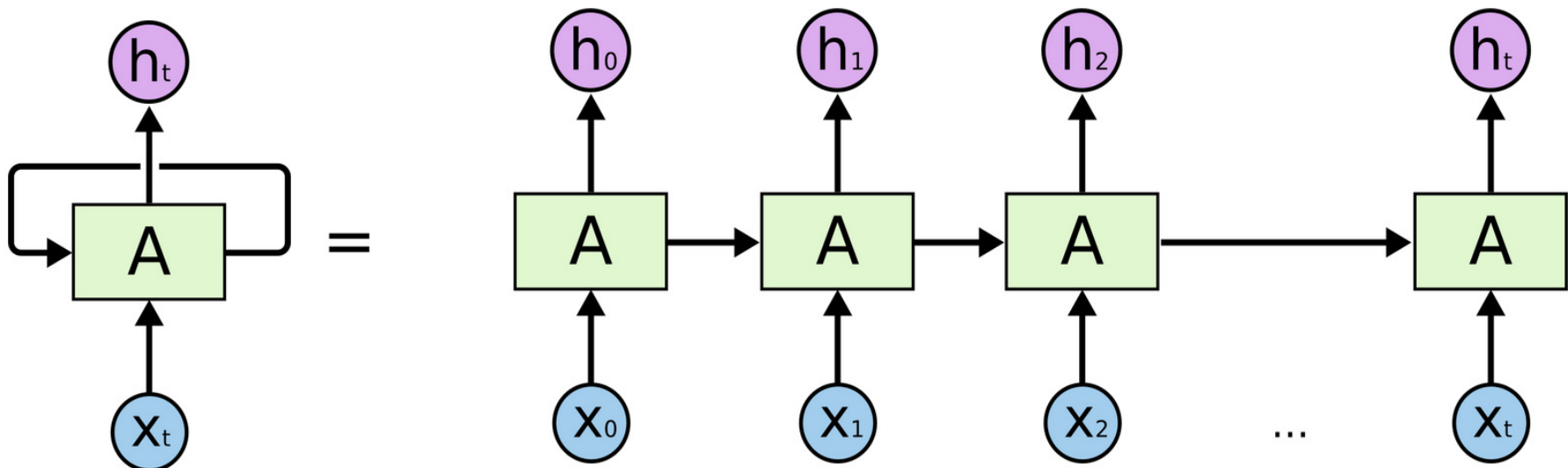
Unrolled Diagrams (I)

- Here, A is a **network** with two inputs and two outputs
- But the **loop** is just for drawing



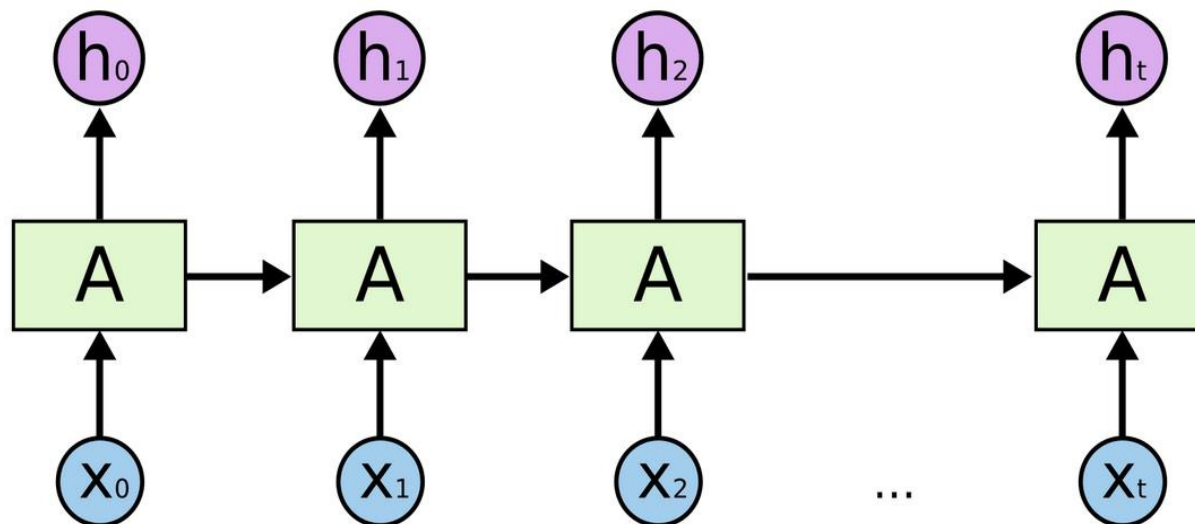
Unrolled Diagrams (II)

- There is no self-loop
- Here is the unrolled representation



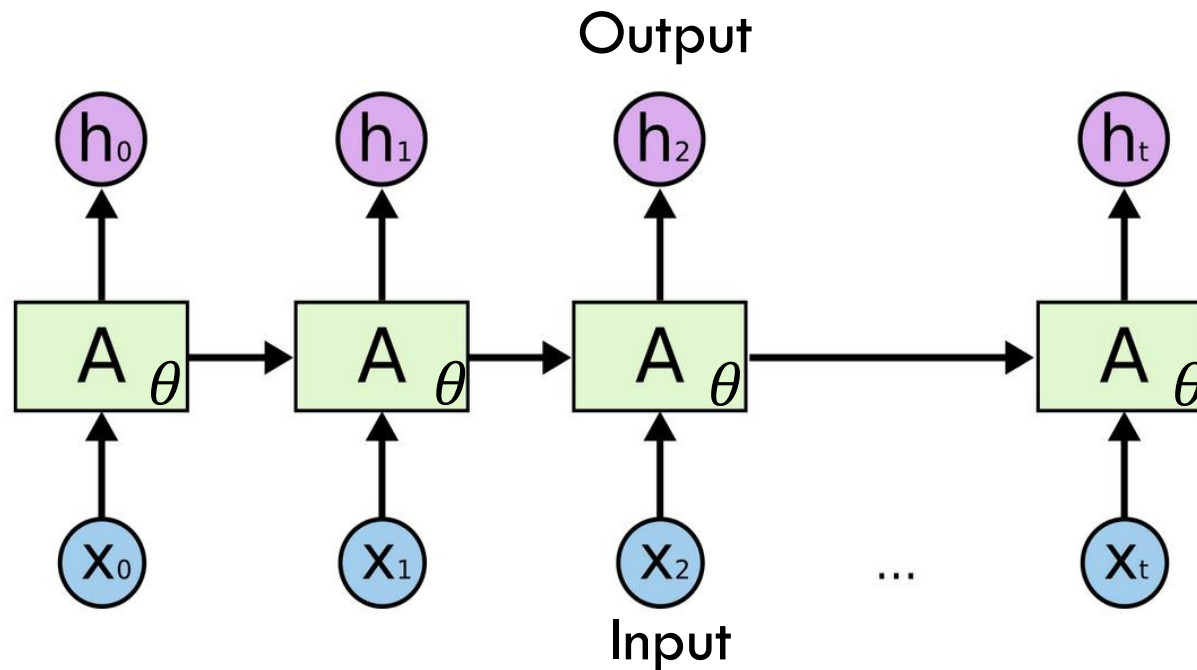
Unrolled Diagrams (III)

- This sequential or repetitive structure is useful for working with sequences
 - Of images
 - Of words



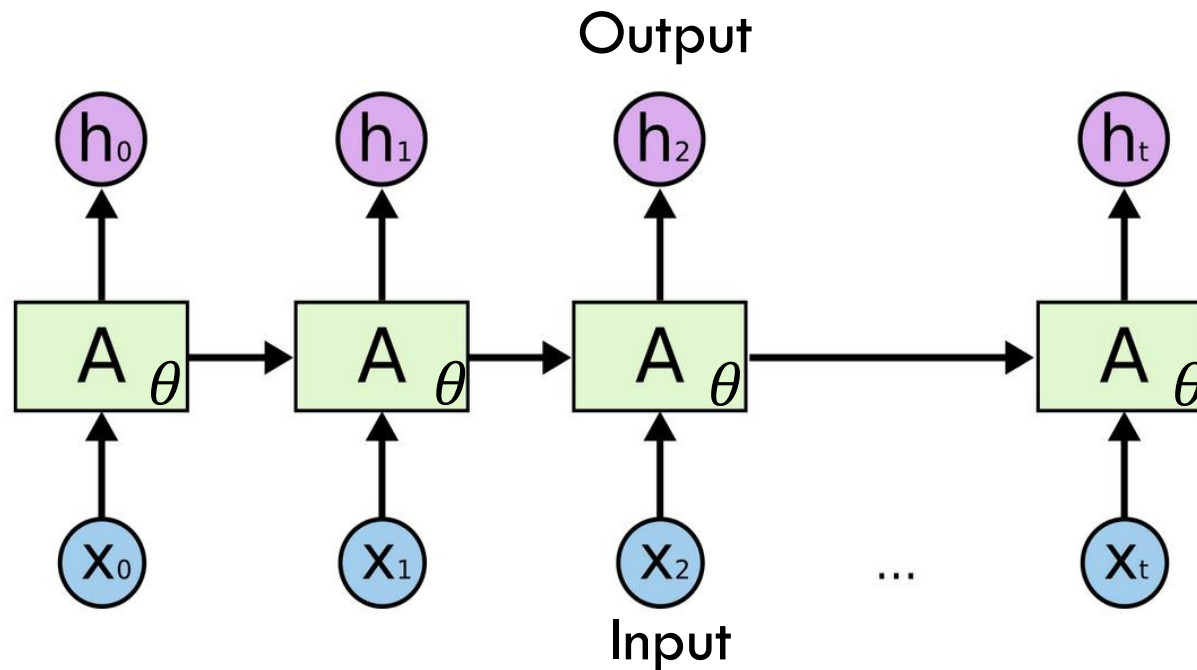
Unrolled Diagrams (V)

- At a **stage**, they accept an input and give an output, which are parts of sequences



Vanilla RNN (I)

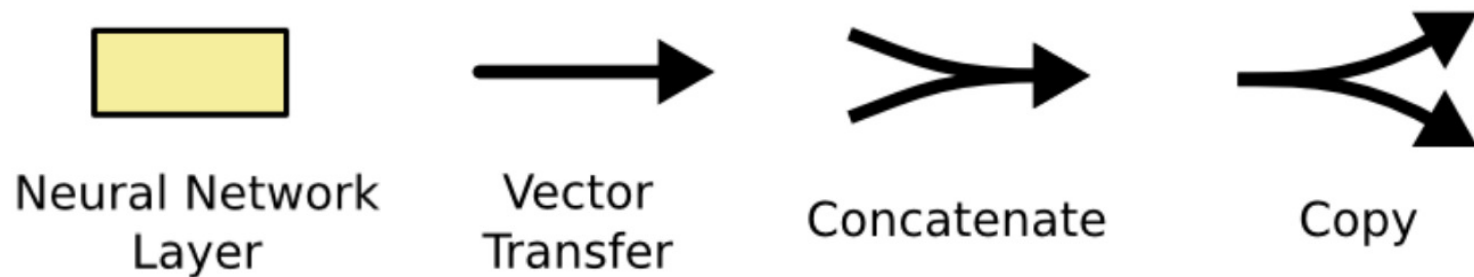
- Unrolled representation is key to understanding
 - For vanilla RNN it is:



- Assuming a single hidden layer with tanh nonlinearity

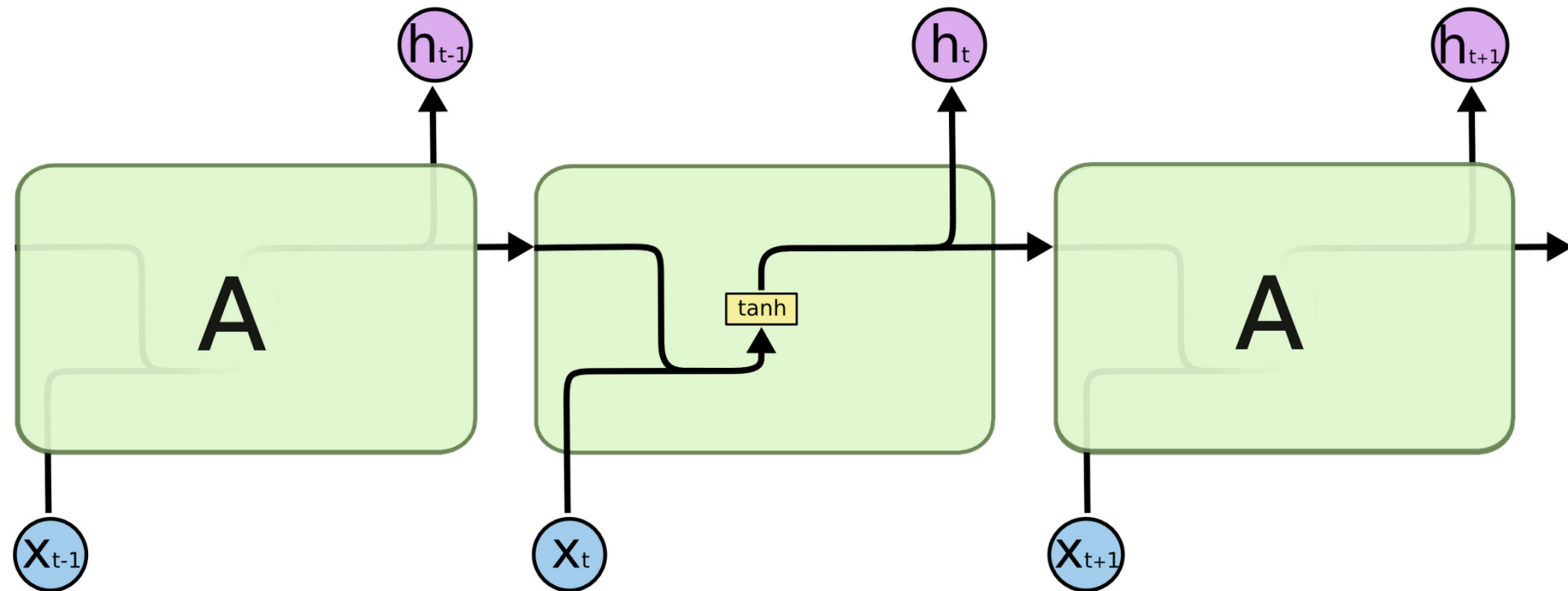
Vanilla RNN (II)

- Some quick notation
 - Dark arrow represents a vector
 - Box represents a (fully connected hidden) layer



Vanilla RNN (III)

- Unrolled representation is key to understanding
 - For vanilla RNN it is:



- Assuming a single hidden layer with tanh nonlinearity

Vanilla RNN in Code

- Training an RNN means finding θ (e.g., W and b) that give rise to a desired behavior quantified by a loss function

```
import numpy as np

class RNN:
    #...
    def __init__(self, len_h, len_x):
        self.h = np.zeros(len_h)
        self.W = np.random.randn(len_h, len_h + len_x)
        self.bias = np.random.randn(len_h)
        #...
    def step(self, x_t):
        activation = np.dot(self.W, np.hstack((self.h, x_t))) + self.bias
        self.h = np.tanh(activation)
        return self.h #could have returned g(self.h) for some function g

rnn = RNN(3, 4)
for _ in range(5):
    x_t = np.random.randn(4)
    h_t = rnn.step(x_t)
    print h_t
```


Language Model (LM) Example

- Build a character-level language model
 - Give RNN a large text dataset
 - Model the probability of the next character given a sequence of previous characters
- Allows us to generate new text!
- Note: This is a toy example where better methods exist

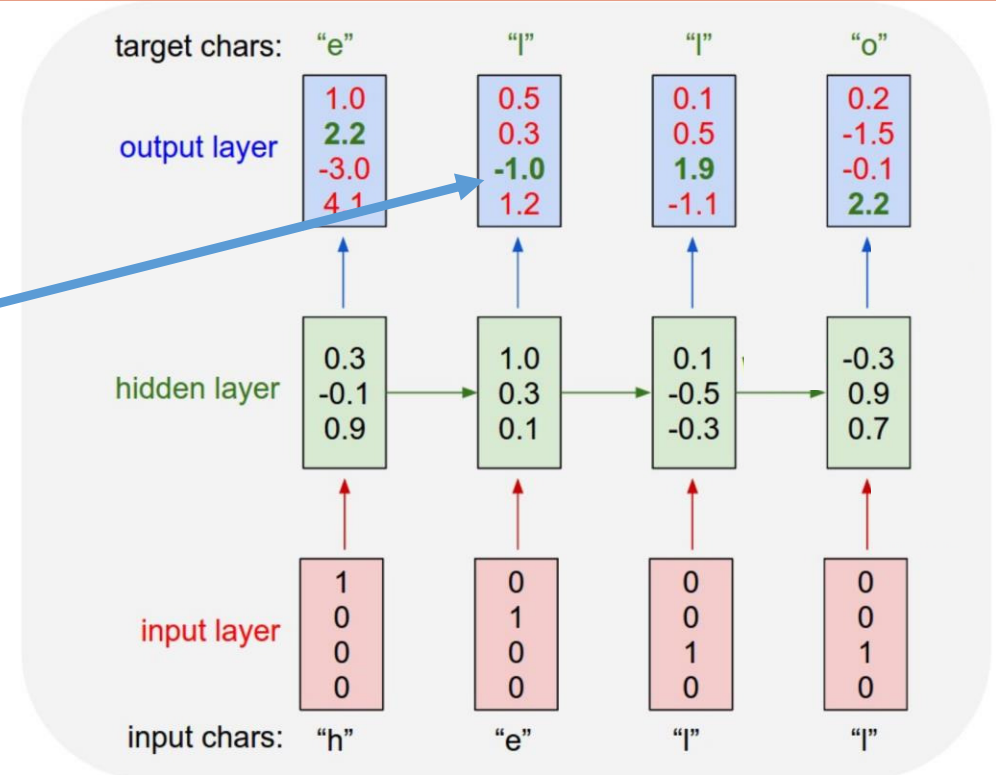
LM Example: Data and Embedding

- Vocabulary: $\{h, e, l, o\}$
- Training sequence: $\{h, e, l, l, o\}$
 - Four training examples:
 - $P(e | h)$ should be high
 - $P(l | he)$ should be high
 - $P(l | hel)$ should be high
 - $P(o | hell)$ should be high
- Embedding:
 - Encode each character as a 4-dimensional vector

¹Reference: <http://karpathy.github.io/2015/05/21/rnn-effectiveness/>

LM Example: RNN

We want green numbers to be high and red numbers to be low



¹Figure: <http://karpathy.github.io/2015/05/21/rnn-effectiveness/>

- Feed each vector into the RNN
- Output is a sequence of vectors
 - Let dimension be 4
 - Interpret as the confidence that the corresponding character is the next in sequence

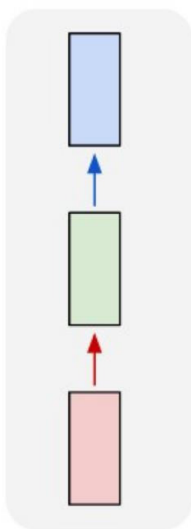
LM Example: RNN

- Define loss as the cross entropy loss (i.e., multiclass logistic) on every output vector simultaneously
- When first time $\{l\}$ is input, the next character should be $\{l\}$
- When the second time $\{l\}$ is input, the next character should be $\{o\}$
- Hence, we need state/persistence, which the RNN hopefully captures

RNN Application Categories

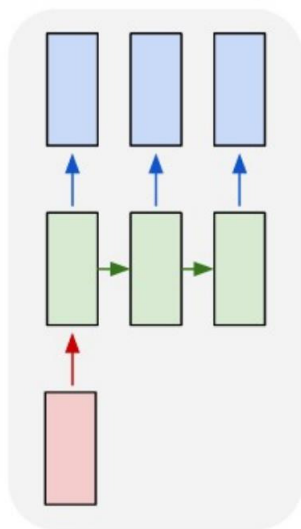
- Input: Red, Output: Blue, RNN's state: Green

one to one



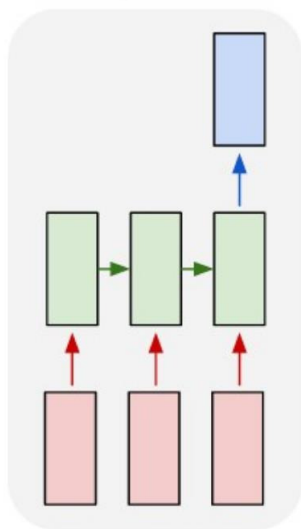
Classifier
Fixed input
Fixed output

one to many



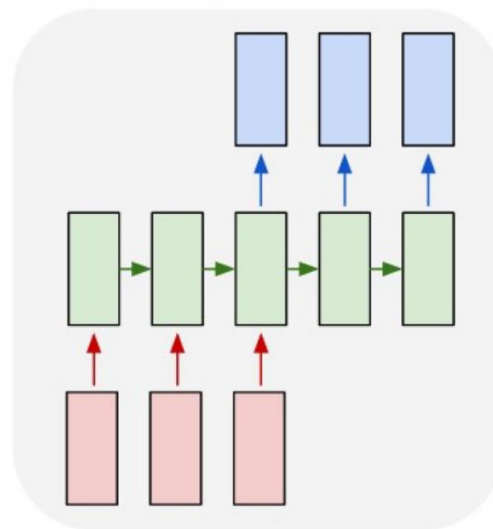
Sequence output
E.g.: Image captioning

many to one



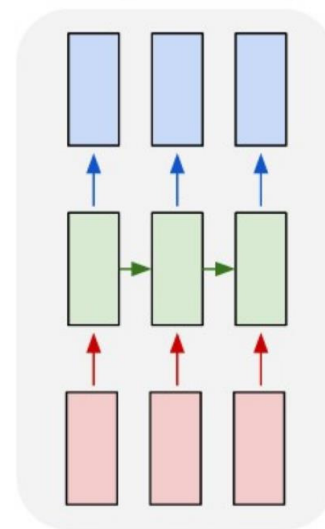
Sequence input
E.g.: Sentiment analysis

many to many



Sequence input
Sequence output
E.g.: Machine translation

many to many



Sequence input
Sequence output
E.g.: Video classification

Questions?

Today's Outline

- Recurrent Neural Networks
- Long-Short Term Memory based RNNs
- Sequence to Sequence Learning and other RNN Applications

Long-Short Term Memory RNNs

Long Term vs Short Term (I)

- Why are we looking at RNN?
 - Hypothesis: the network can connect past information to the current
 - Can they?
 - It depends...

Long Term vs Short Term (II)

- Consider a model predicting next word based on previous words
- Case A:
 - $R(\text{"... advanced prediction"}) = \text{"models"}$
 - Here, the **immediate** preceding words are helpful

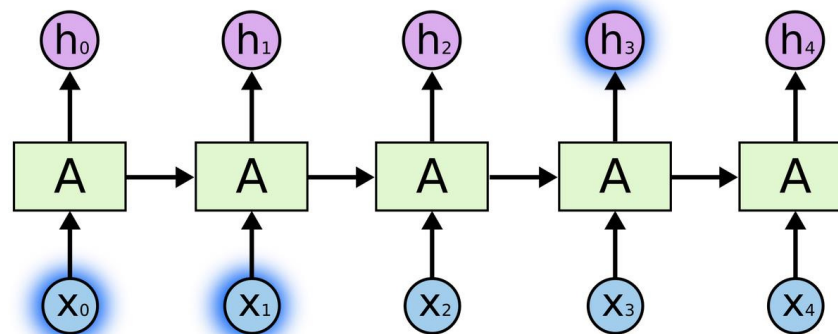
Long Term vs Short Term (II)

- Consider a model predicting next word based on previous words
- Case A:
 - $R(\text{"... advanced prediction"}) = \text{"models"}$
 - Here, the **immediate** preceding words are helpful
- Case B:
 - $R(\text{"I went to UIC... I lived in [?]"}) = \text{"Chicago"}$
 - Here, more context is needed
 - Recent info suggests [?] is a place.
 - Need the context of UIC from further back

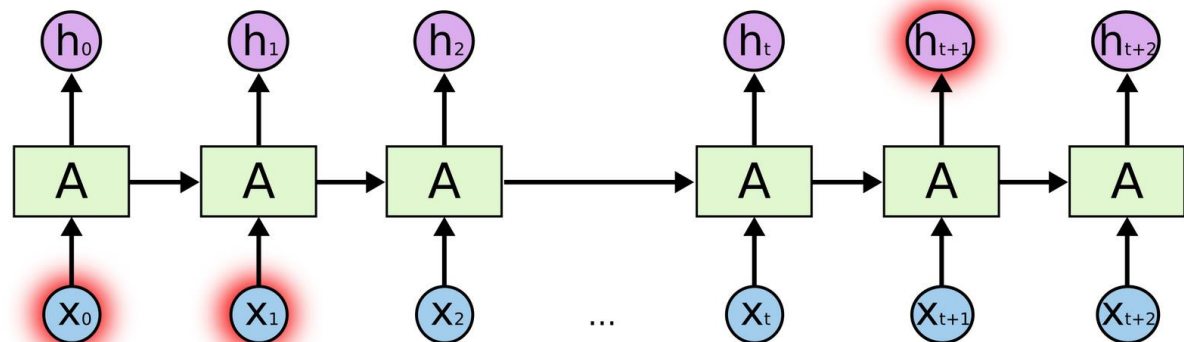
Long Term vs Short Term (III)

- Consider a model predicting next word based on previous words

- Case A:



- Case B:



A Special RNN: LSTM

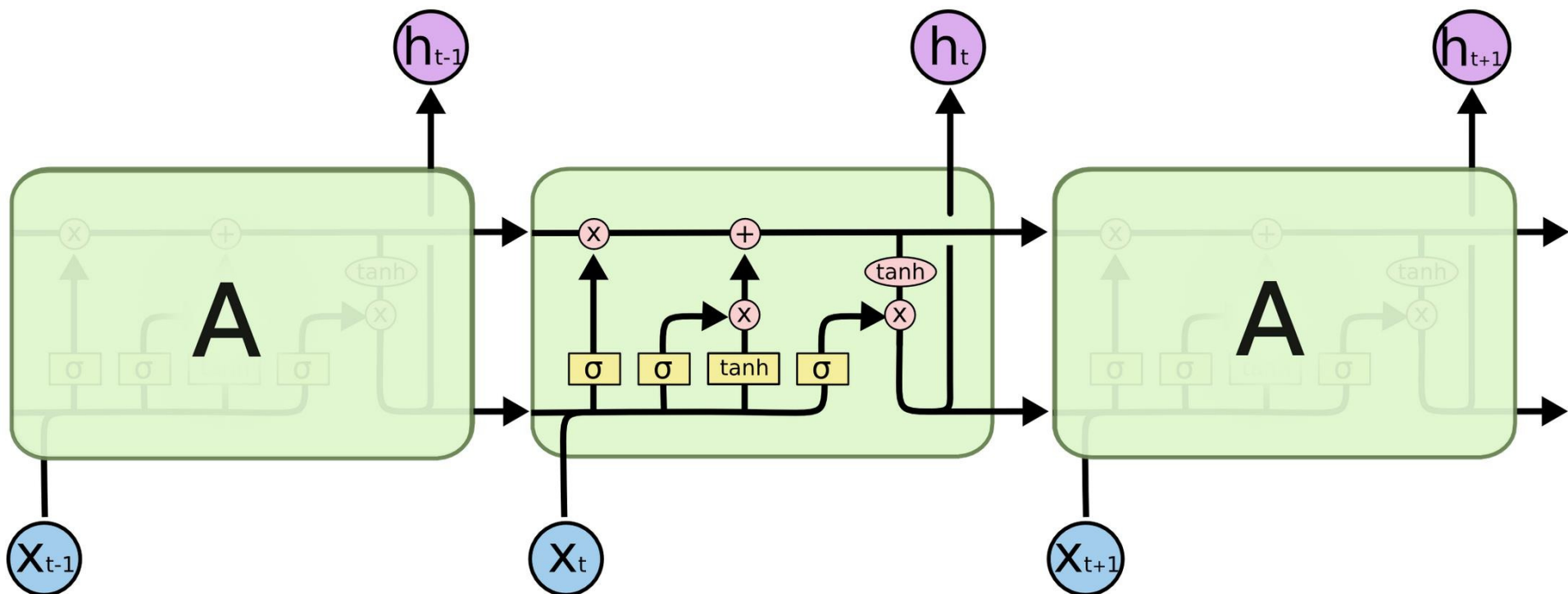
- The gap between the relevant information and the point where it is needed can **become unbounded**
- **Empirical observation:** RNNs are unable to learn to connect long range information.
- Thus, we look at LSTMs (Long Short Term Memory Cells)

LSTM: Long Short Term Memory based RNN

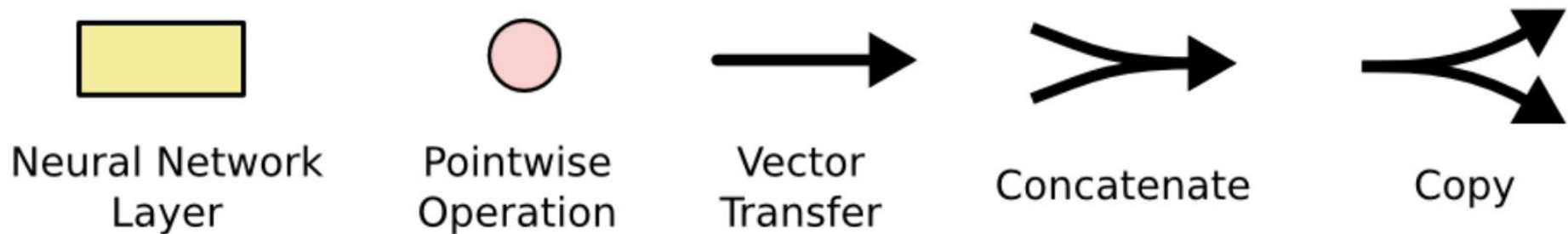
- Capable of learning long-term dependencies
- Designed to avoid the long range issue that a vanilla RNN faces
 - How do they do that? We will address that now

LSTM: Block Level

- LSTM RNN have a similar structure to vanilla RNNs
- Only the repeating module is different
- Instead of a single neural layer, they have **four**



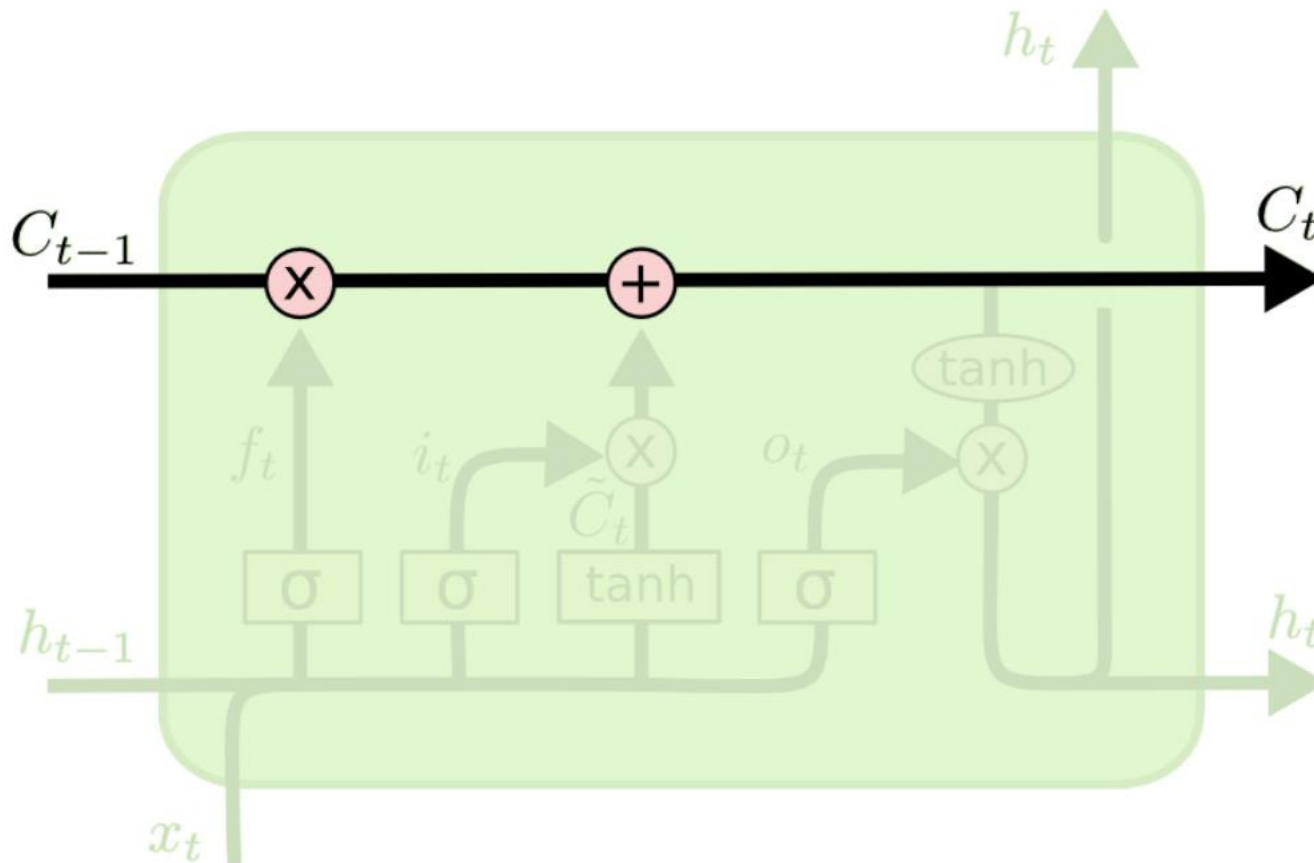
LSTM: Recall Notation



- Dark arrow represents a vector, output from one layer and input to another
- Circle represents element-wise operations
 - Example: sum of two vectors
- Box represents a (fully connected) hidden layer

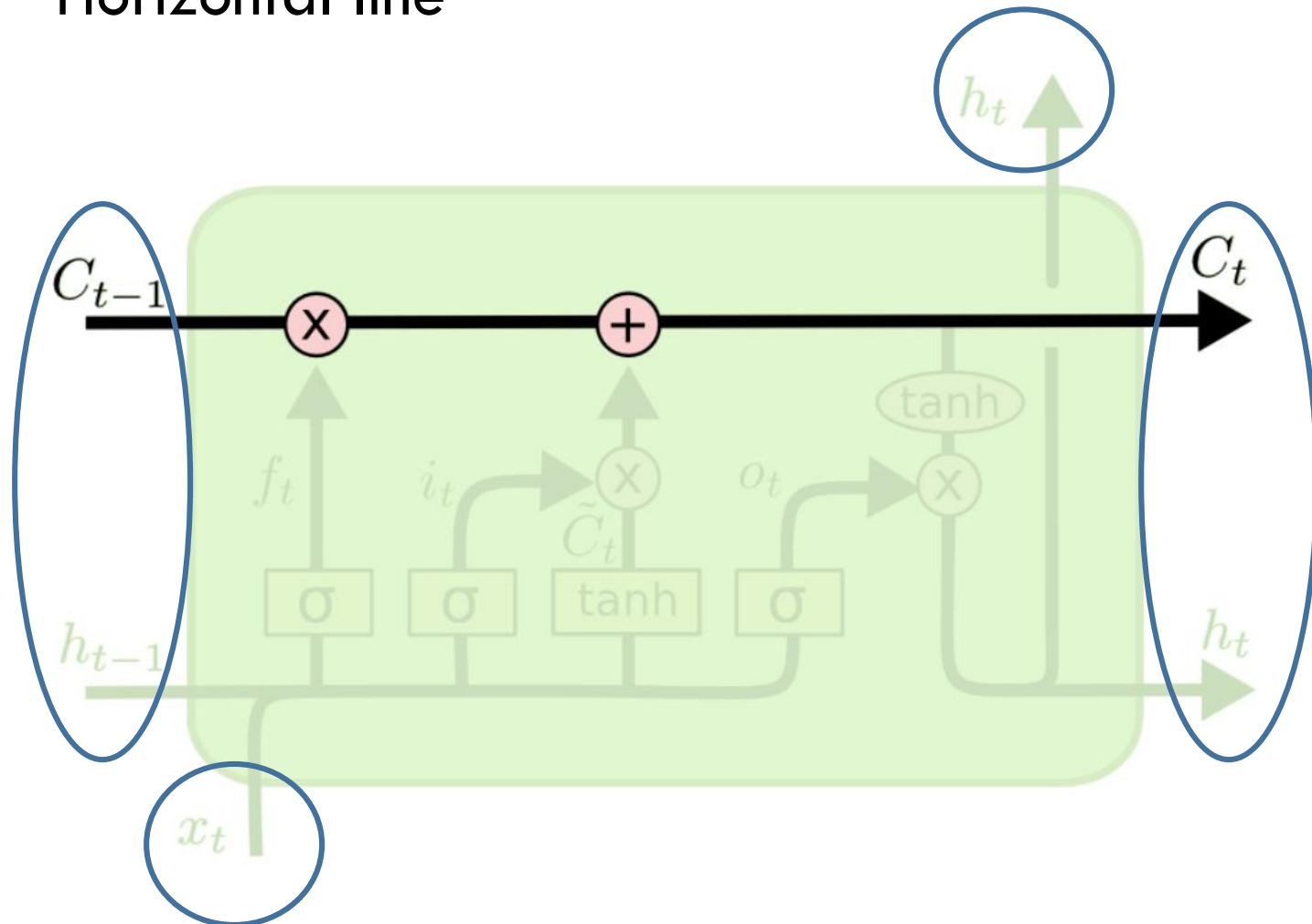
LSTM: Cell State (I)

- There is a notion of **cell state**
 - Horizontal line



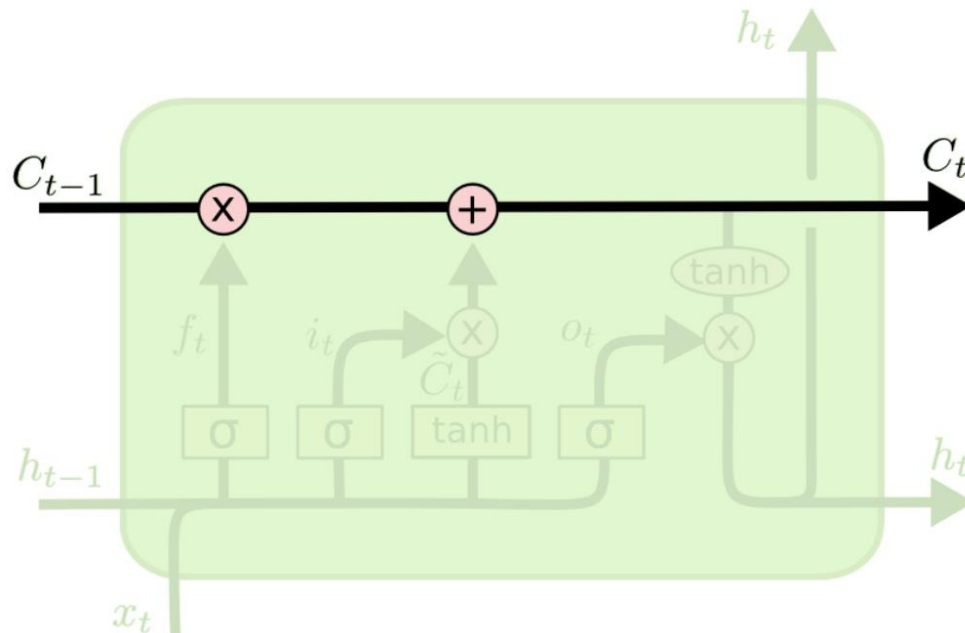
LSTM: Cell State (I)

- There is a notion of **cell state**
 - Horizontal line



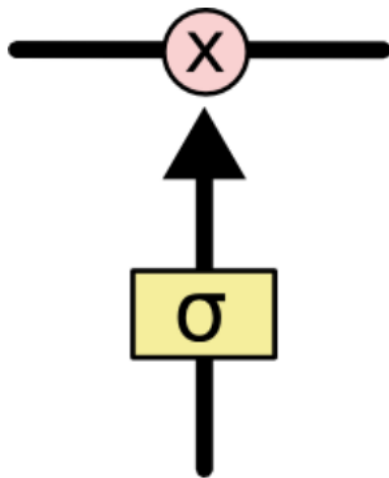
LSTM: Cell State (II)

- Cell state:
 - Runs straight down the unrolled network
 - Minor interactions
 - Information could **flow** along it unchanged



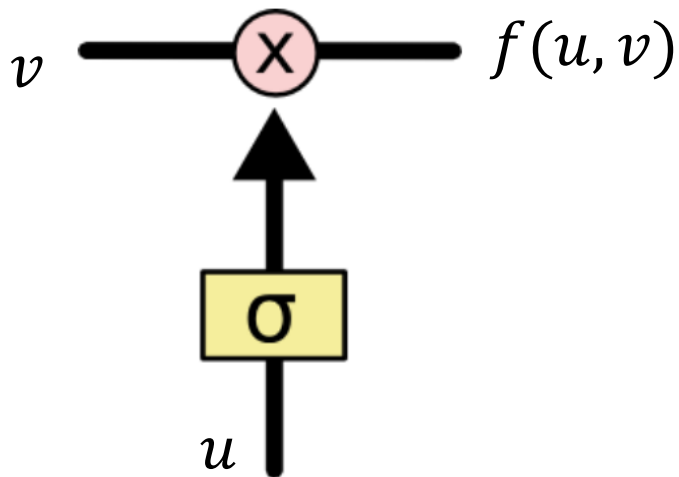
LSTM: Gates (I)

- The LSTM can add or remove information to the cell state by regulating **gates**
- Gates optionally let information through
 - Made of a sigmoid NN layer and a pointwise multiplication



LSTM: Gates (I)

- The LSTM can add or remove information to the cell state by regulating **gates**
- Gates optionally let information through
 - Made of a sigmoid NN layer and a pointwise multiplication

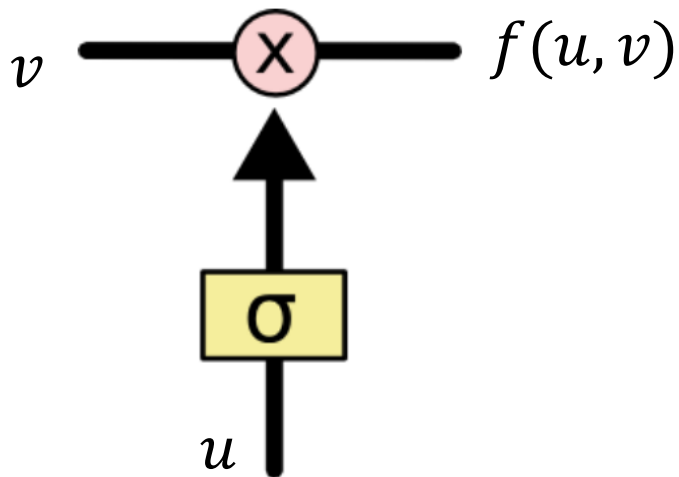


Mathematically,

$$f(u, v) = v \otimes \sigma(Wu + b)$$

LSTM: Gates (II)

- Gate:
 - The sigmoid layer outputs numbers in (0,1)
 - Determines how much of each component to let through
 - 0 means 'do not let input through'
 - 1 means 'let input through'

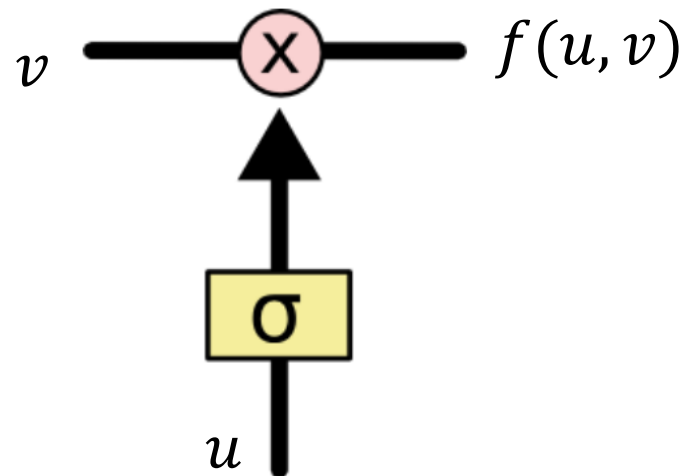


Mathematically,

$$f(u, v) = v \otimes \sigma(Wu + b)$$

LSTM: Gates (III)

- LSTM has three gates to **control** the **cell state**

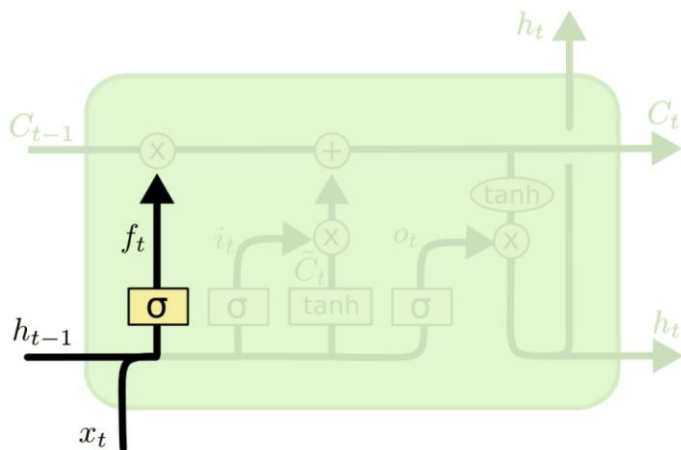


LSTM: Forget Old Information

- First Step: what information to throw away from cell state
- Decided by forget gate layer
 - Input: h_{t-1} and x_t
 - Output: a vector with entries in $(0,1)$ corresponding to entries in C_{t-1}
 - 1 corresponds to keep the input
 - 0 corresponds to get rid of the input

LSTM: Forget Old Information

- Example: In the task of predicting the next word based on all previous ones
 - Cell state **may** include gender of current subject
 - This will be useful to predict/use correct pronouns (male: he, female: she)
 - When a new subject is observed
 - Need to forget the gender of old subject



$$f_t = \sigma (W_f \cdot [h_{t-1}, x_t] + b_f)$$

LSTM: Remember New Information

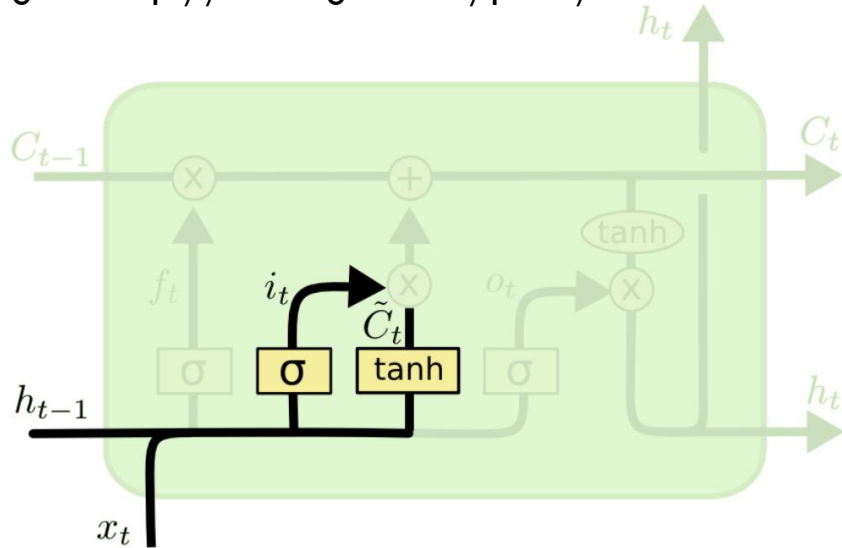
- Next step: decide what new information we will store in cell state
- Two ingredients
 - Input gate layer
 - Tanh layer
- Input gate layer
 - Decides which values to update

LSTM: Remember New Information

- Next step: decide what new information we will store in cell state
- Two ingredients
 - Input gate layer
 - Tanh layer
- Input gate layer
 - Decides which values to update
- Tanh layer
 - Creates a vector of new candidate values \tilde{C}_t that can be added to the cell state

LSTM: Remember New Information

¹Figure: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

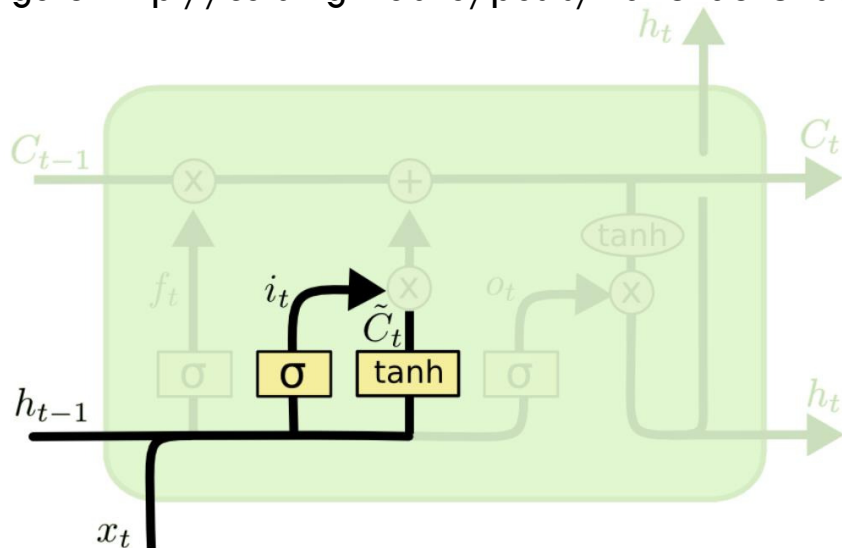


$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

- Input gate layer
 - Decides which values to update
- Tanh layer
 - Creates a vector of new candidate values \tilde{C}_t that can be added to the cell state

LSTM: Remember New Information

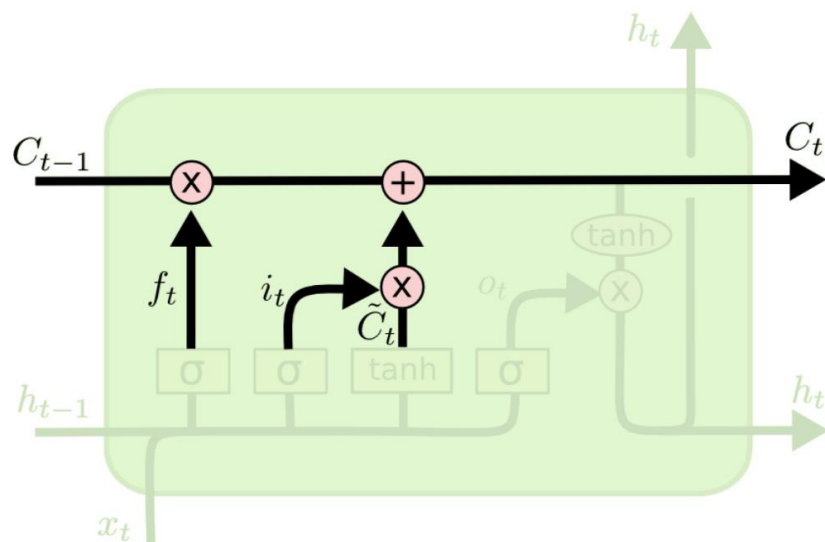
¹Figure: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>



- Combine \tilde{C}_t with the output i_t of the input gate layer to get $i_t \otimes \tilde{C}_t$
- In the language model example
 - Add the gender of the new subject to the cell state (this replaces the old one we are forgetting)

LSTM: Forget and Remember

- Last step:
 - Modify the cell state



$$C_t = f_t \otimes C_{t-1} + i_t \otimes \tilde{C}_t$$

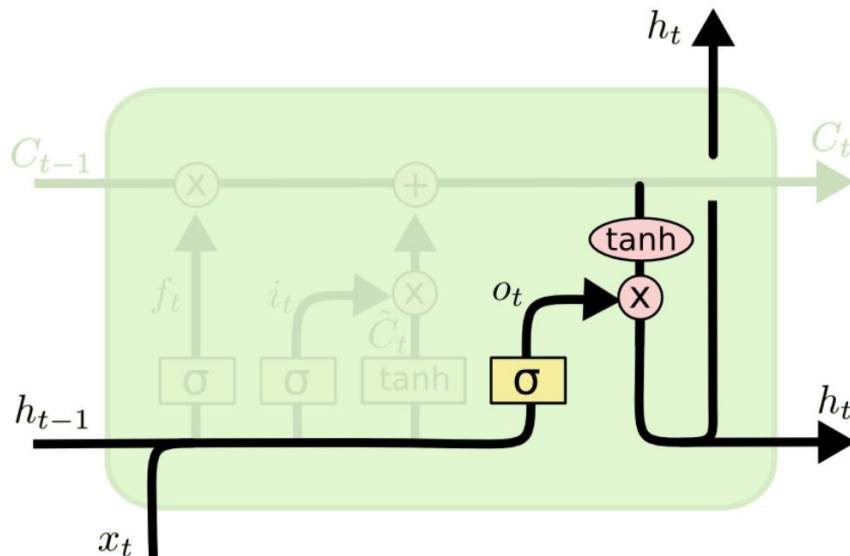
- $i_t \otimes \tilde{C}_t$ are the new values, scaled by how much we want to update each coordinate of cell state

LSTM: Output

- Output a filtered or transformed version of cell state
- Two stages:
 - Pass the cell state through a tanh layer
 - Scale it with a sigmoid layer output
 - The sigmoid layer decides what parts of the cell state we will output

LSTM: Output

- In the language model example
 - Since it just saw a new subject, it may output information related to actions (verbs)
 - Output whether the subject is singular or plural so verb can be modified appropriately

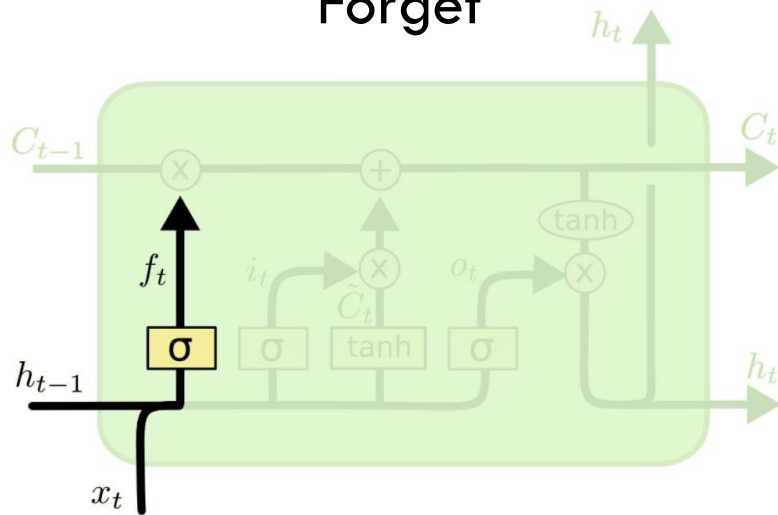


$$o_t = \sigma (W_o [h_{t-1}, x_t] + b_o)$$

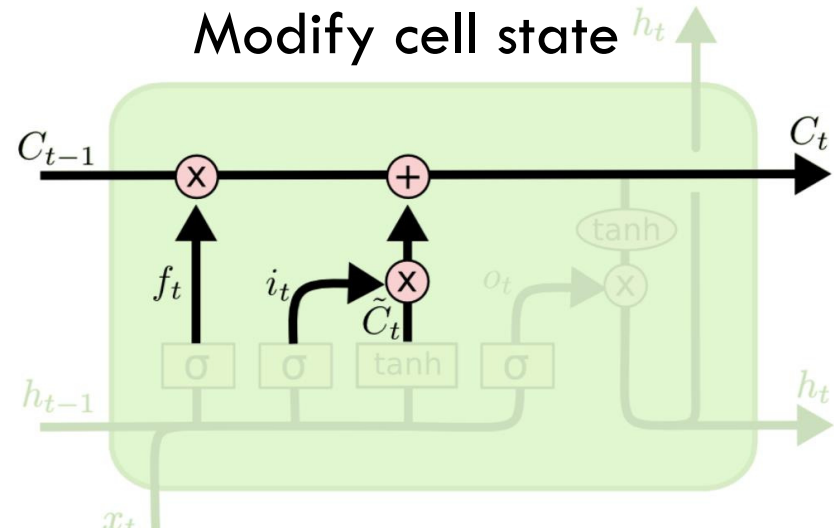
$$h_t = o_t \otimes \tanh (C_t)$$

LSTM: Architecture Summary

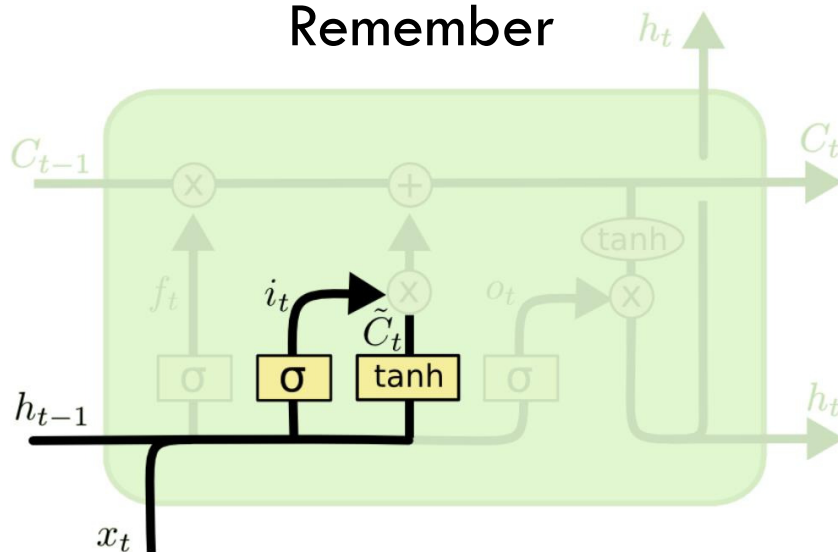
Forget



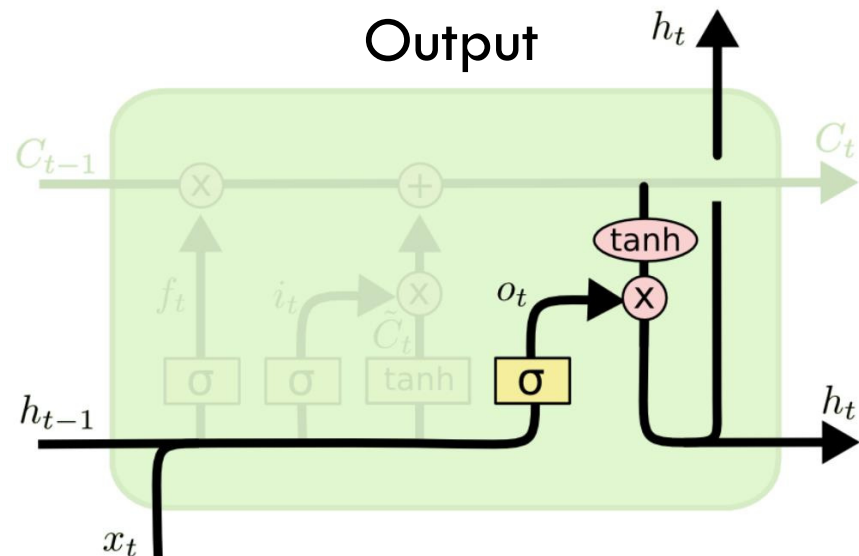
Modify cell state



Remember

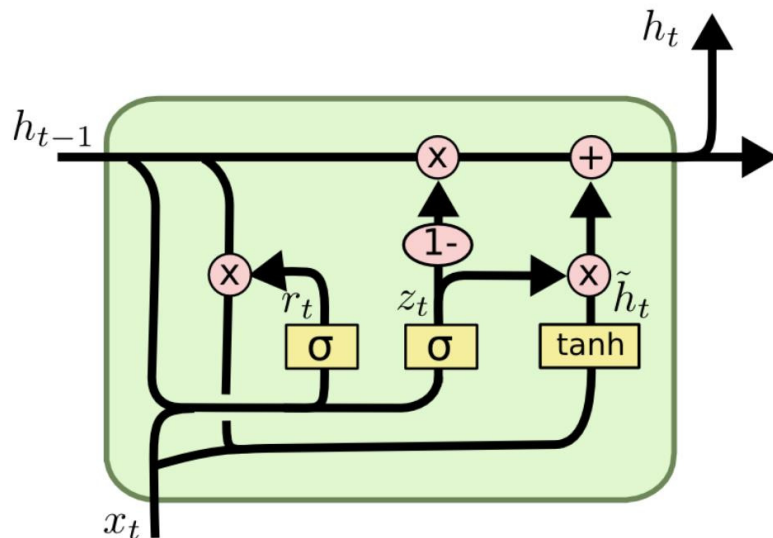


Output



Other Variations in the Family of RNNs (I)

- The vanilla RNN and the LSTM we saw are just one of many variations
- Example: Gated Recurrent Unit (GRU)
 - Combines the forget and input gates
 - Merges the cell state and hidden state
 - ...



$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$

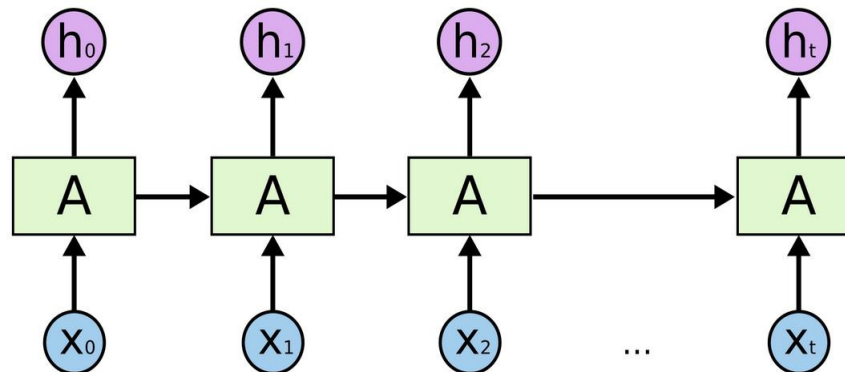
$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$

$$\tilde{h}_t = \tanh(W \cdot [r_t \otimes h_{t-1}, x_t])$$

$$h_t = (1 - z_t) \otimes h_{t-1} + z_t \otimes \tilde{h}_t$$

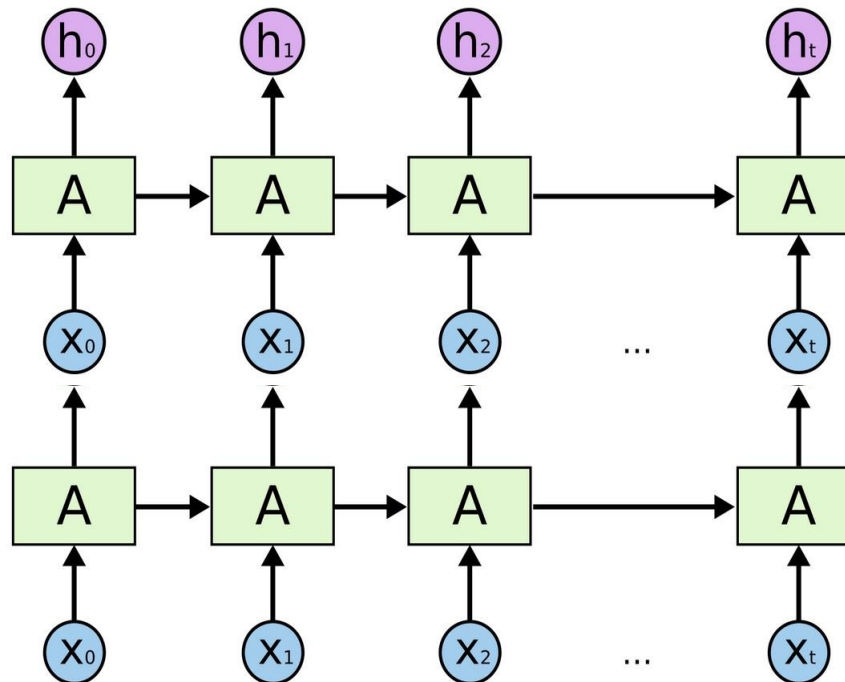
Other Variations in the Family of RNNs (II)

- One can also go deep by stacking RNNs on top of each other



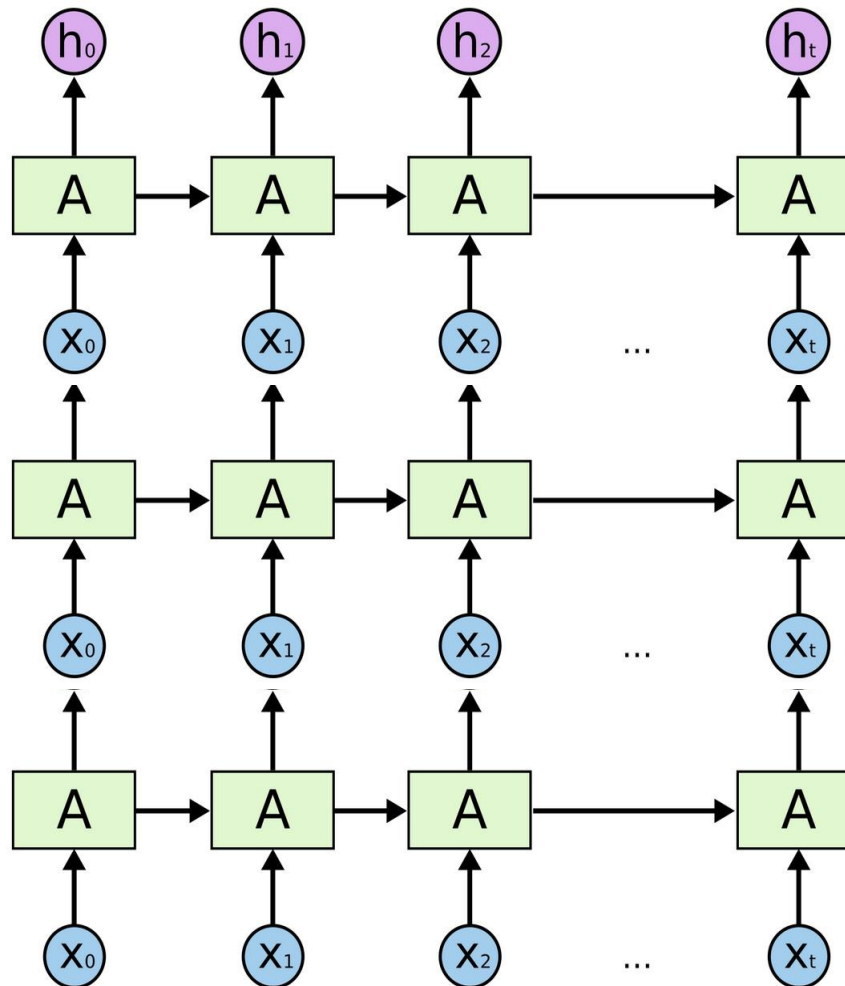
Other Variations in the Family of RNNs (II)

- One can also go deep by stacking RNNs on top of each other



Other Variations in the Family of RNNs (II)

- One can also go deep by stacking RNNs on top of each other



Other Variations in the Family of RNNs (III)

- Extensive investigation has been done to see which variations are the best^{1,2}
- As a practitioner, just use the popular architectures
- To recap, we are studying RNNs because we:
 - Want a notion of state/persistence to capture long term dependence
 - Want to process variable length sequences

¹Reference: <http://arxiv.org/pdf/1503.04069.pdf>

²Reference: <http://jmlr.org/proceedings/papers/v37/jozefowicz15.pdf>

Training RNNs

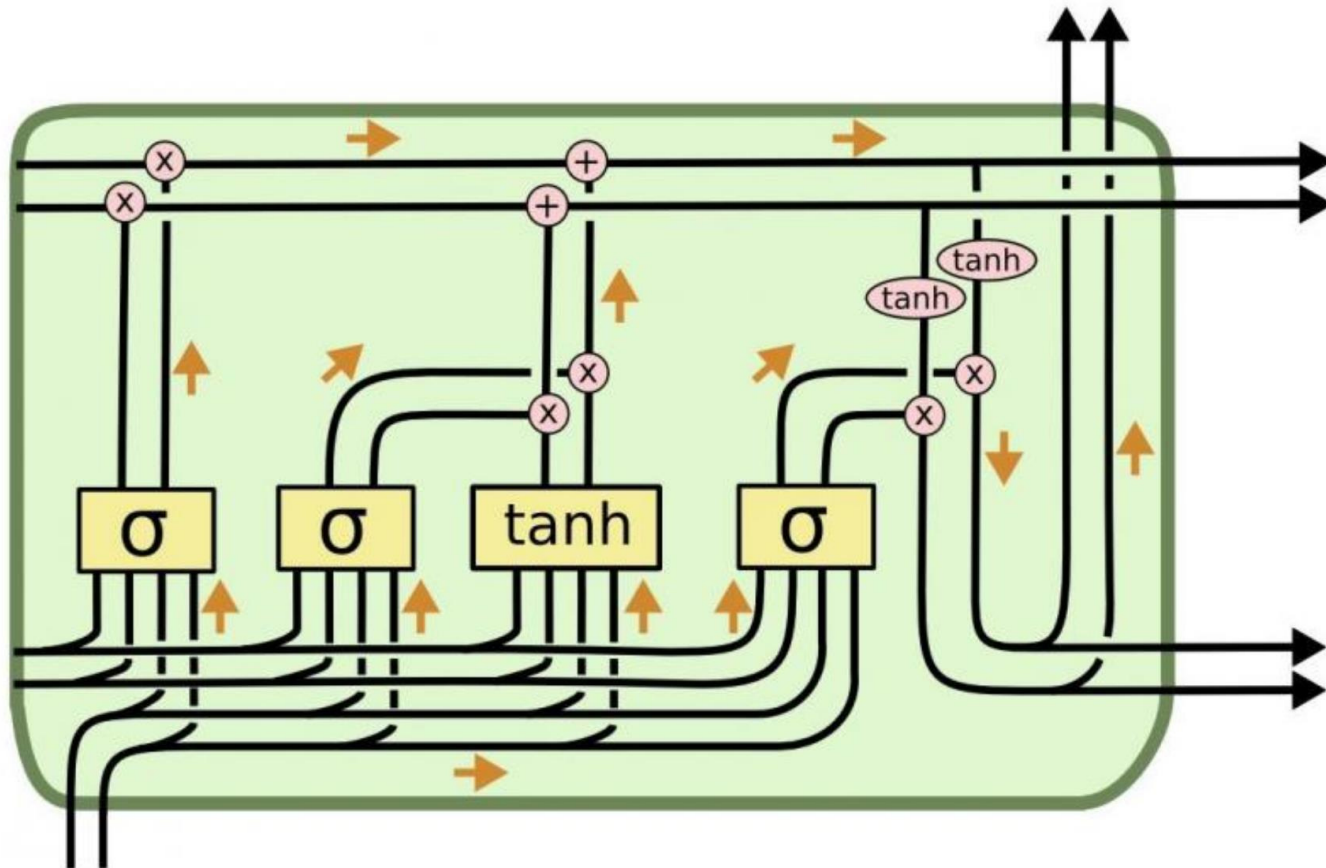
- These networks consist of differentiable operations
- Suitably define loss
- Run backpropagation to find best parameters

Example Code

- LSTM applied to IMDB sentiment classification task
 - See https://github.com/fchollet/keras/blob/master/examples/imdb_lstm.py (Keras)
 - Or see <https://github.com/tflearn/tflearn/blob/master/examples/nlp/lstm.py> (tflearn with Tensorflow)
- Dataset:
<http://ai.stanford.edu/~amaas/data/sentiment/>

LSTM: Accounting for Dimensions

- Think of h_t as 2 dimensional and cell state as 2 dimensional



Questions?

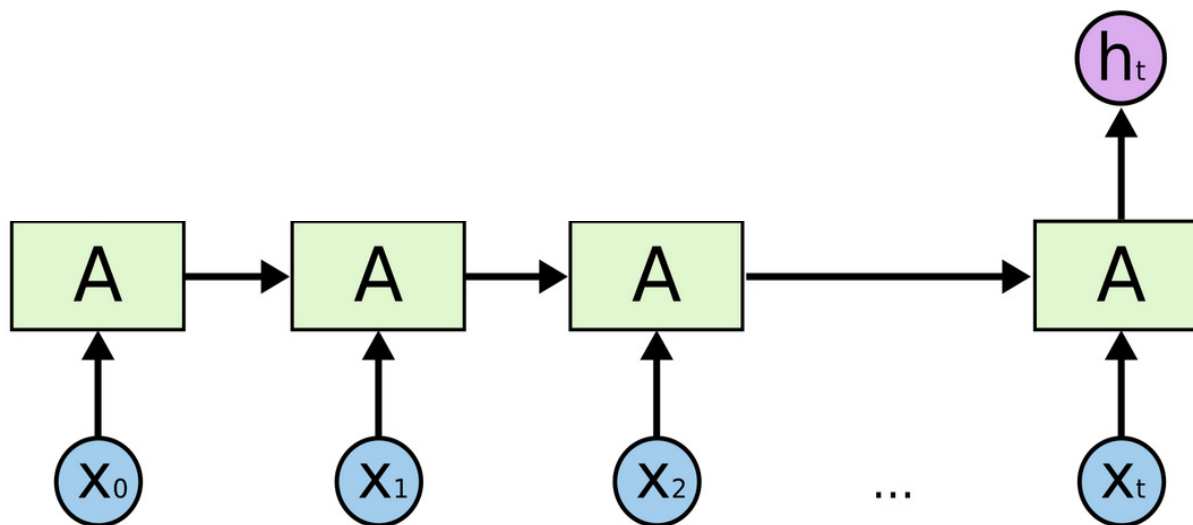
Today's Outline

- Recurrent Neural Networks
- Long-Short Term Memory based RNNs
- Sequence to Sequence Learning and other RNN Applications

Sequence to Sequence Learning and other RNN Applications

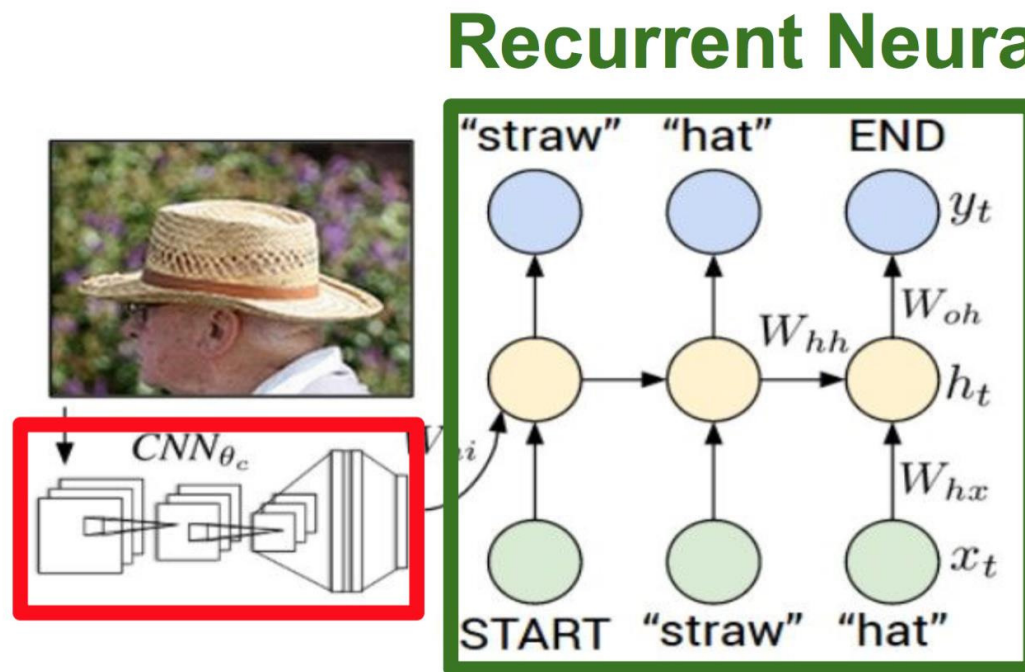
Example I: Sentence Classification

- We saw how to use a CNN for this task.
- We can use the unrolled RNN:



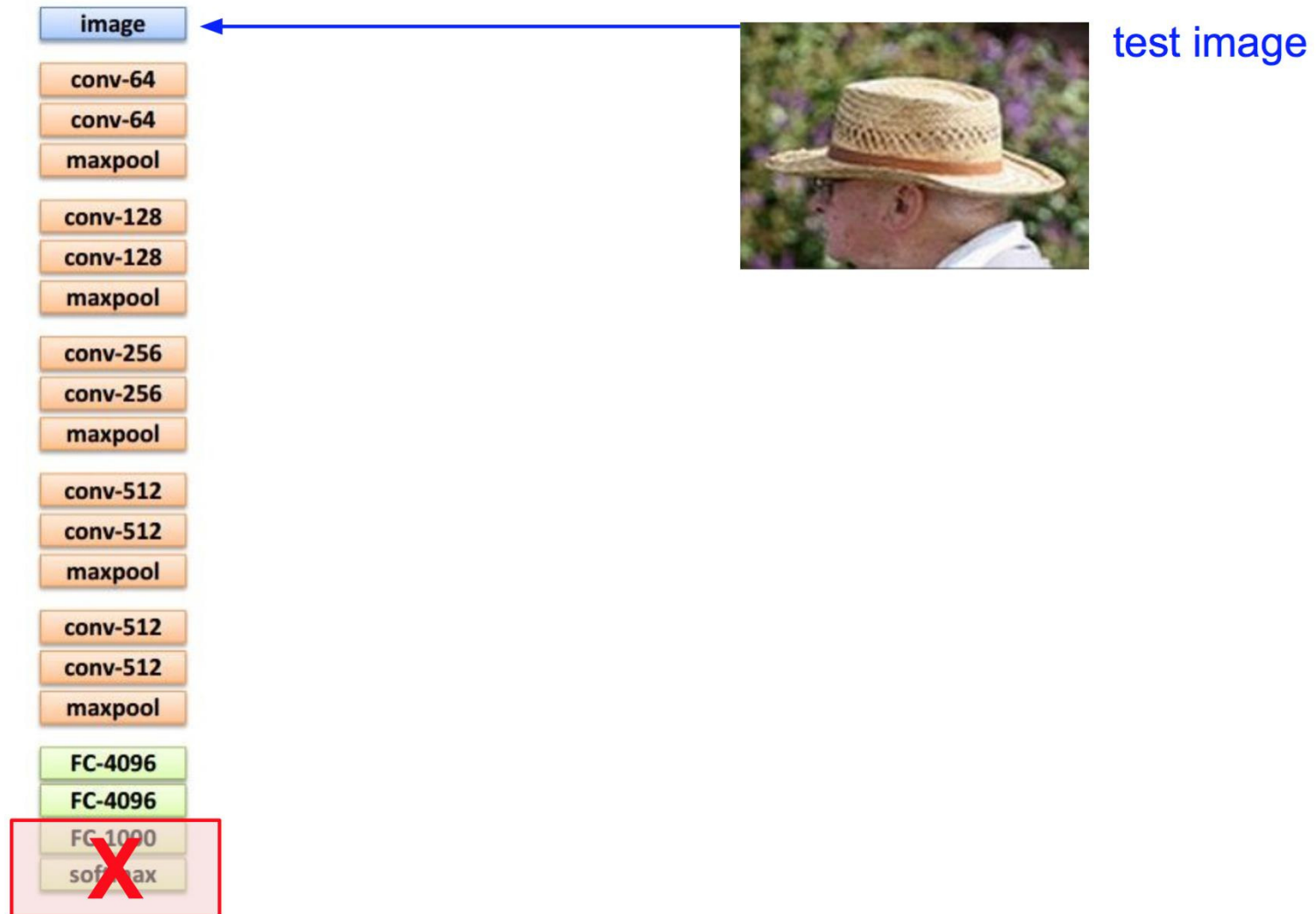
Example II: Image Captioning

- Use CNNs and RNNs together to go from one data type to another

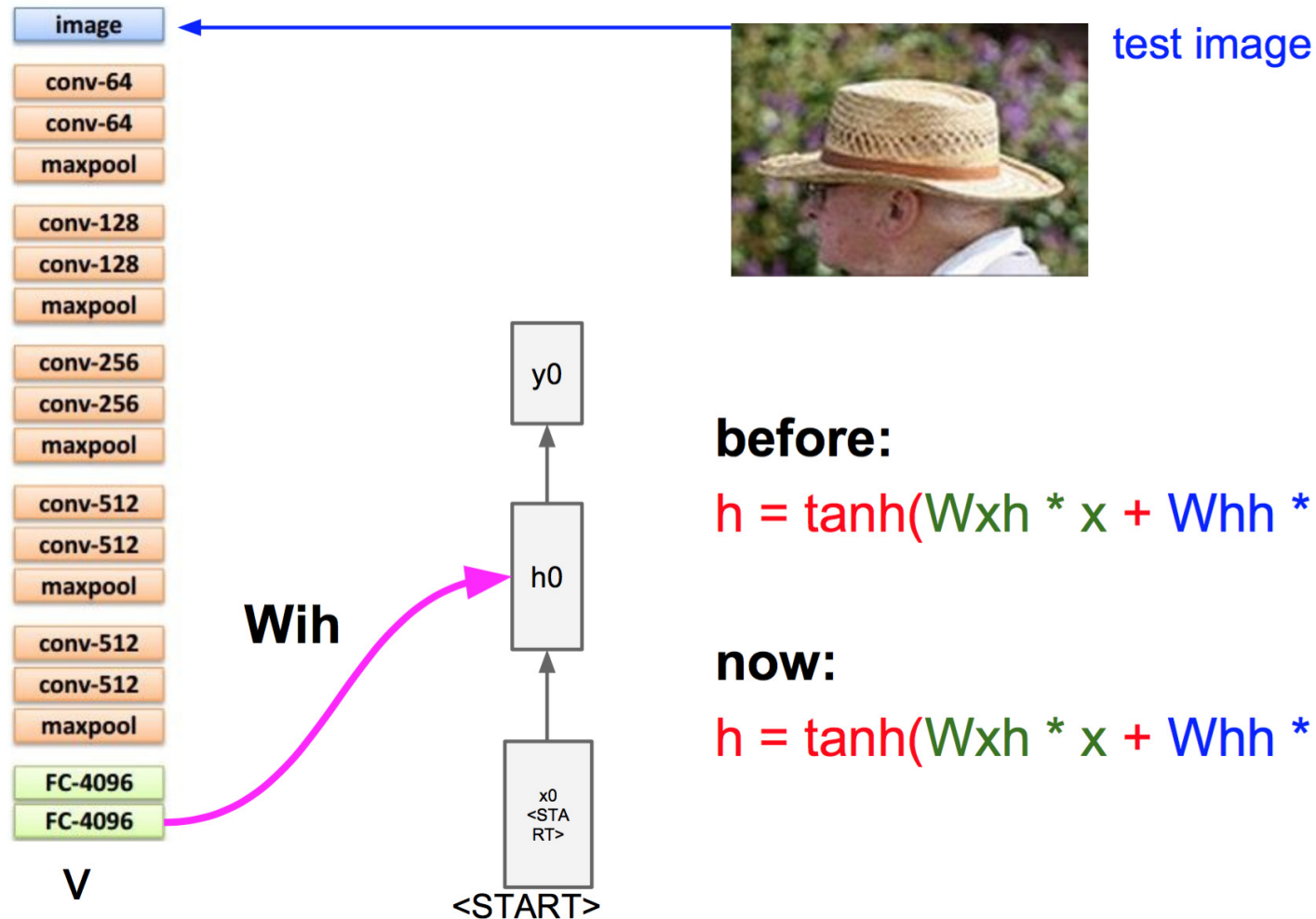


Convolutional Neural Network

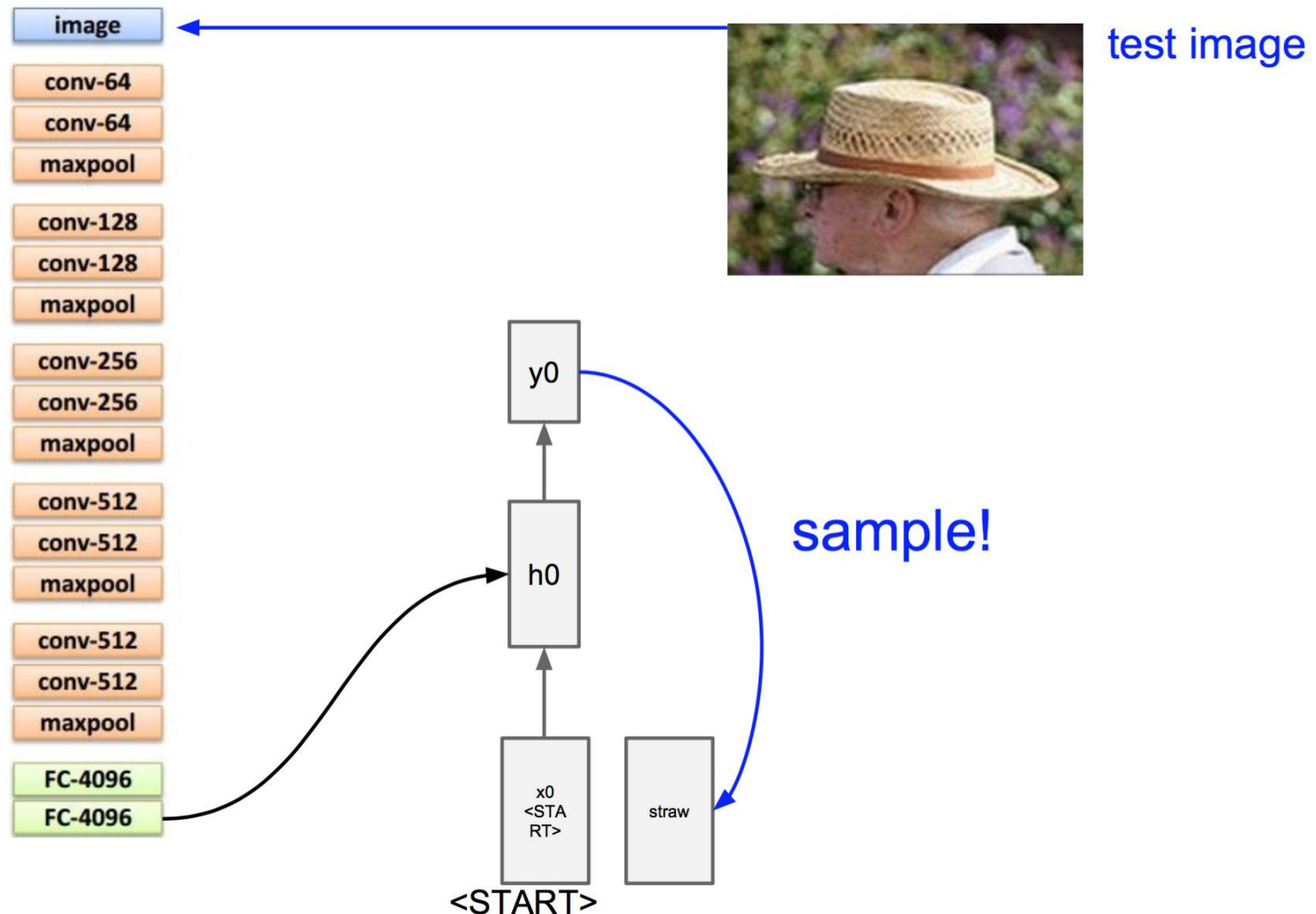
Example II: Image Captioning



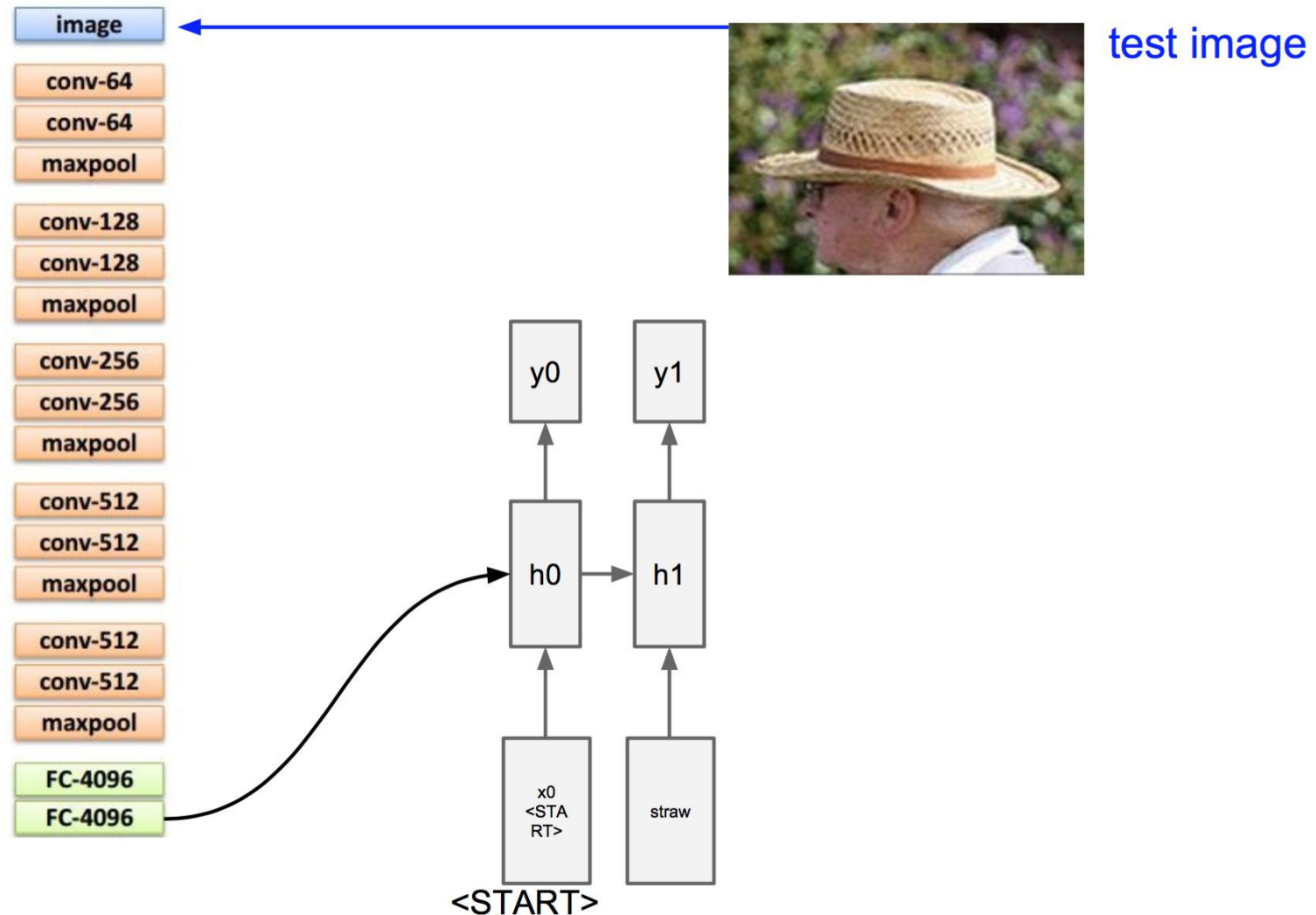
Example II: Image Captioning



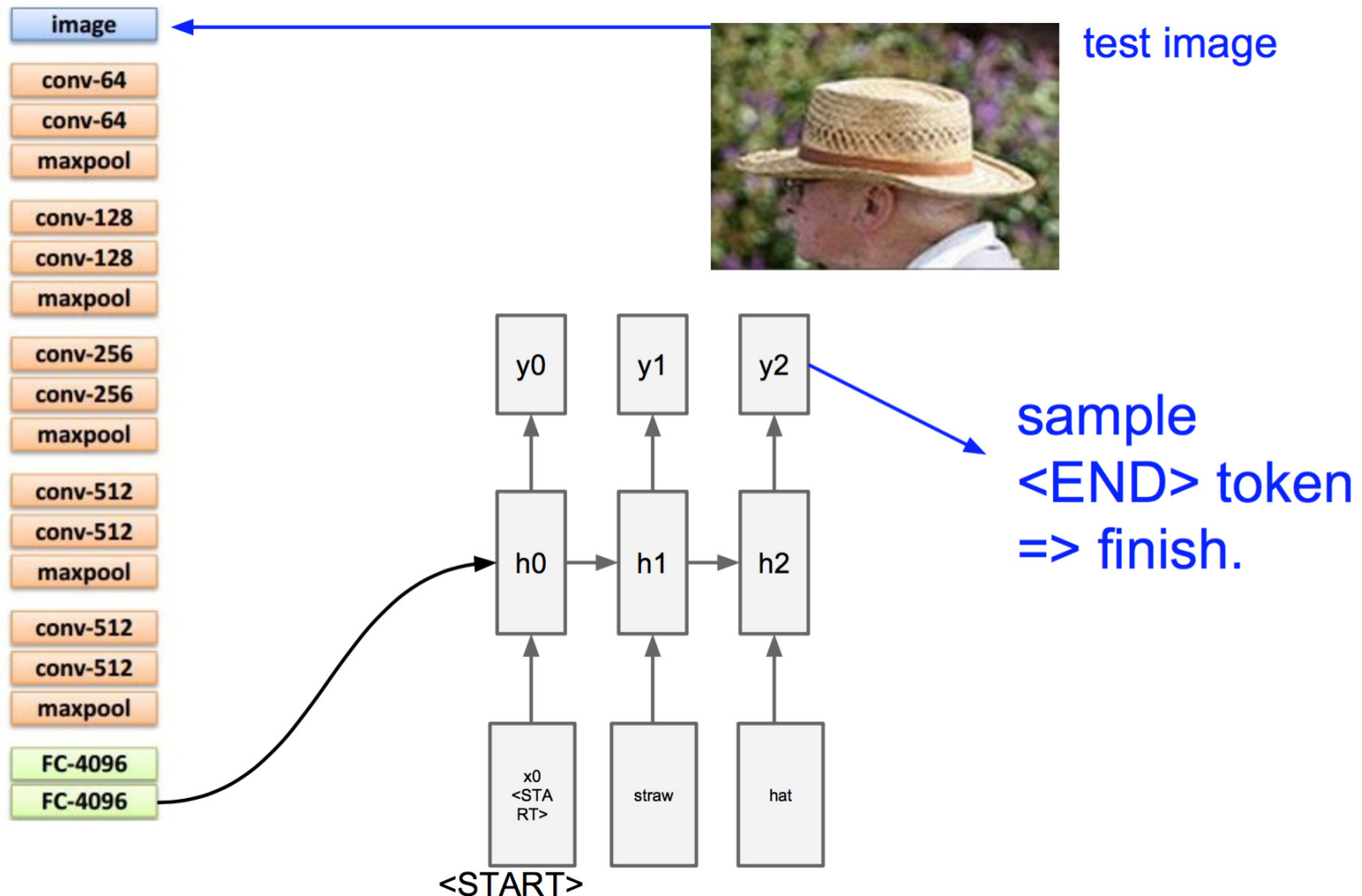
Example II: Image Captioning



Example II: Image Captioning



Example II: Image Captioning

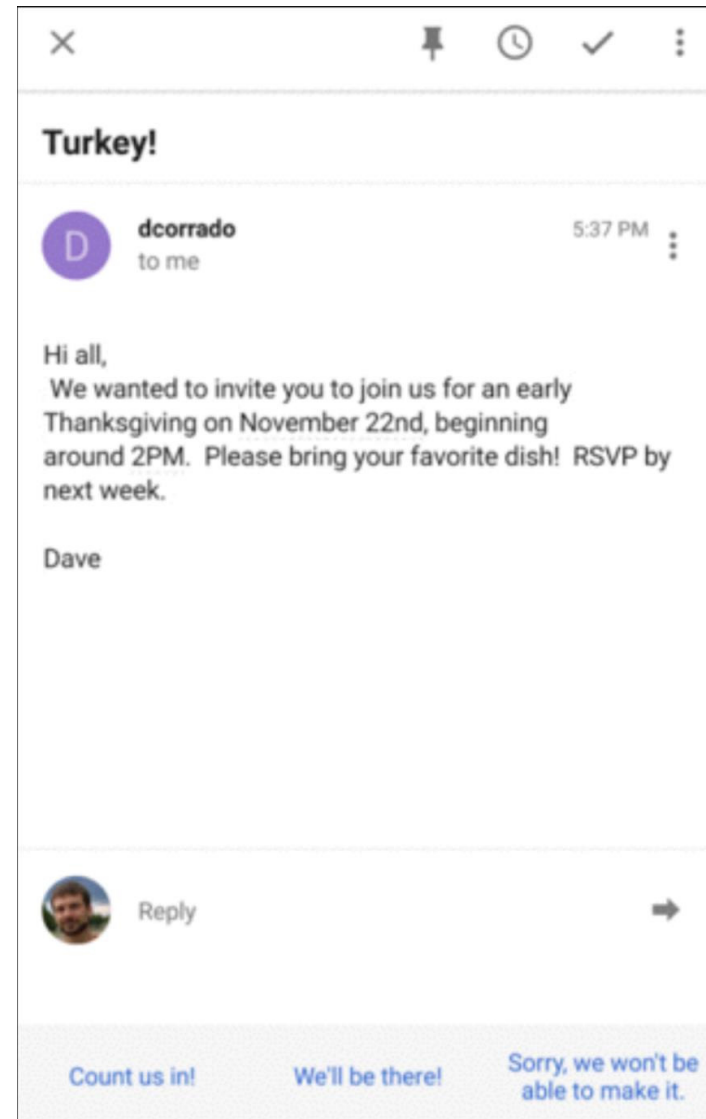


Example III: Auto-Reply

- In this family of applications, we want mapping between variable length inputs to variable length outputs
- Other applications:
 - Translation
 - Summarizing
 - Speech transcription
 - Question answering

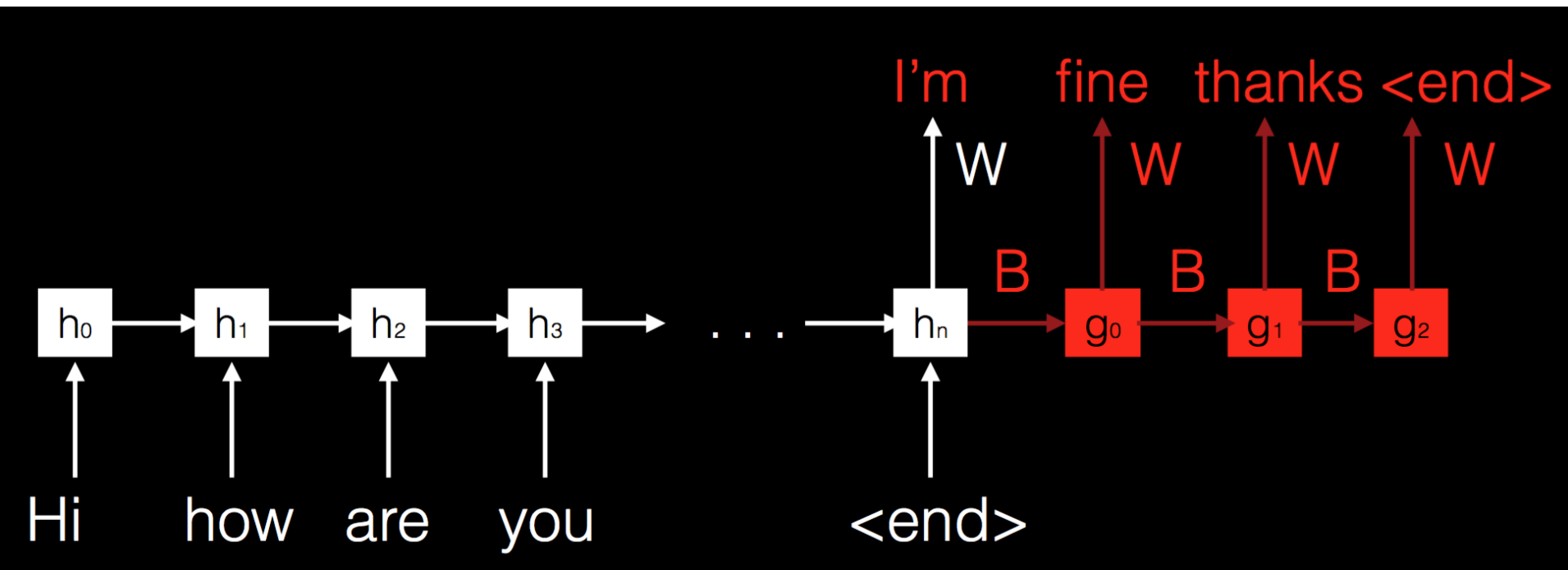
Example III: Auto-Reply

- Auto-reply is a feature where the computer reads your email and responds appropriately



Example III: Auto-Reply

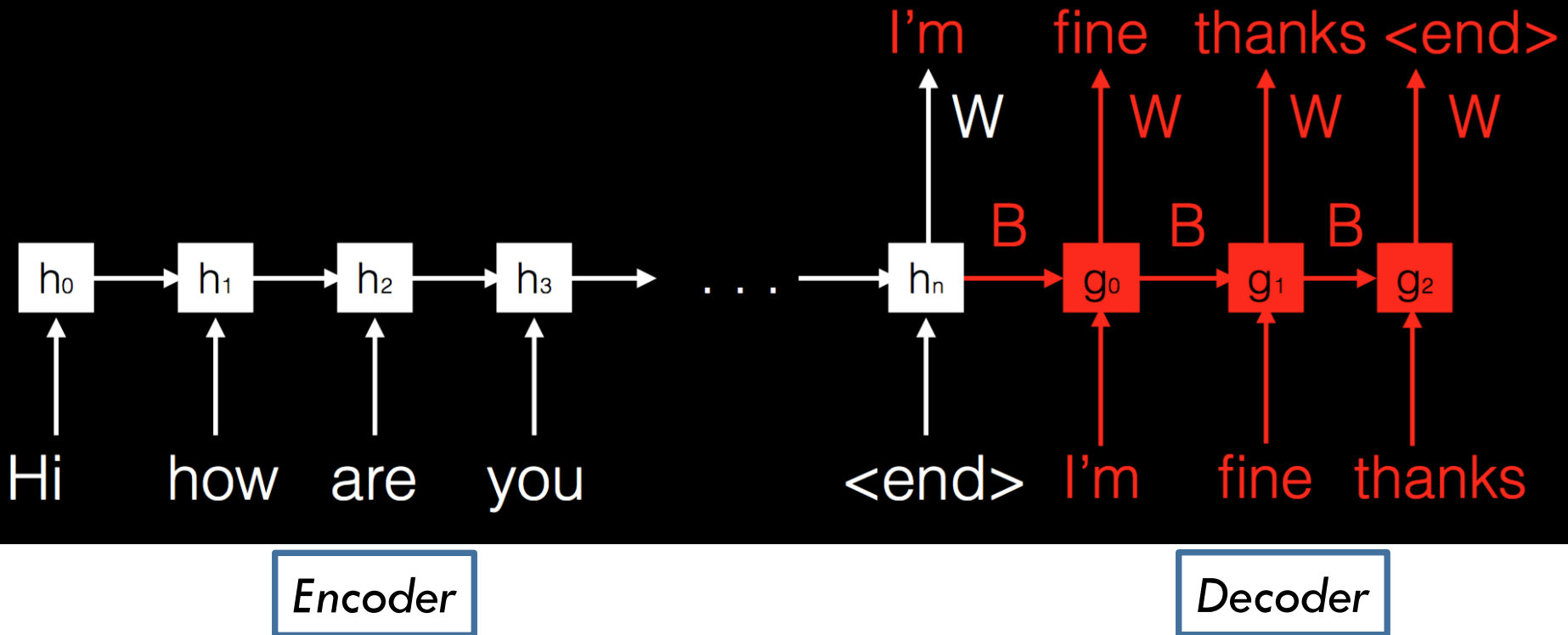
- First version



- Note that the number of classes in output is the number of words in the vocab!

Example III: Auto-Reply

- Second version



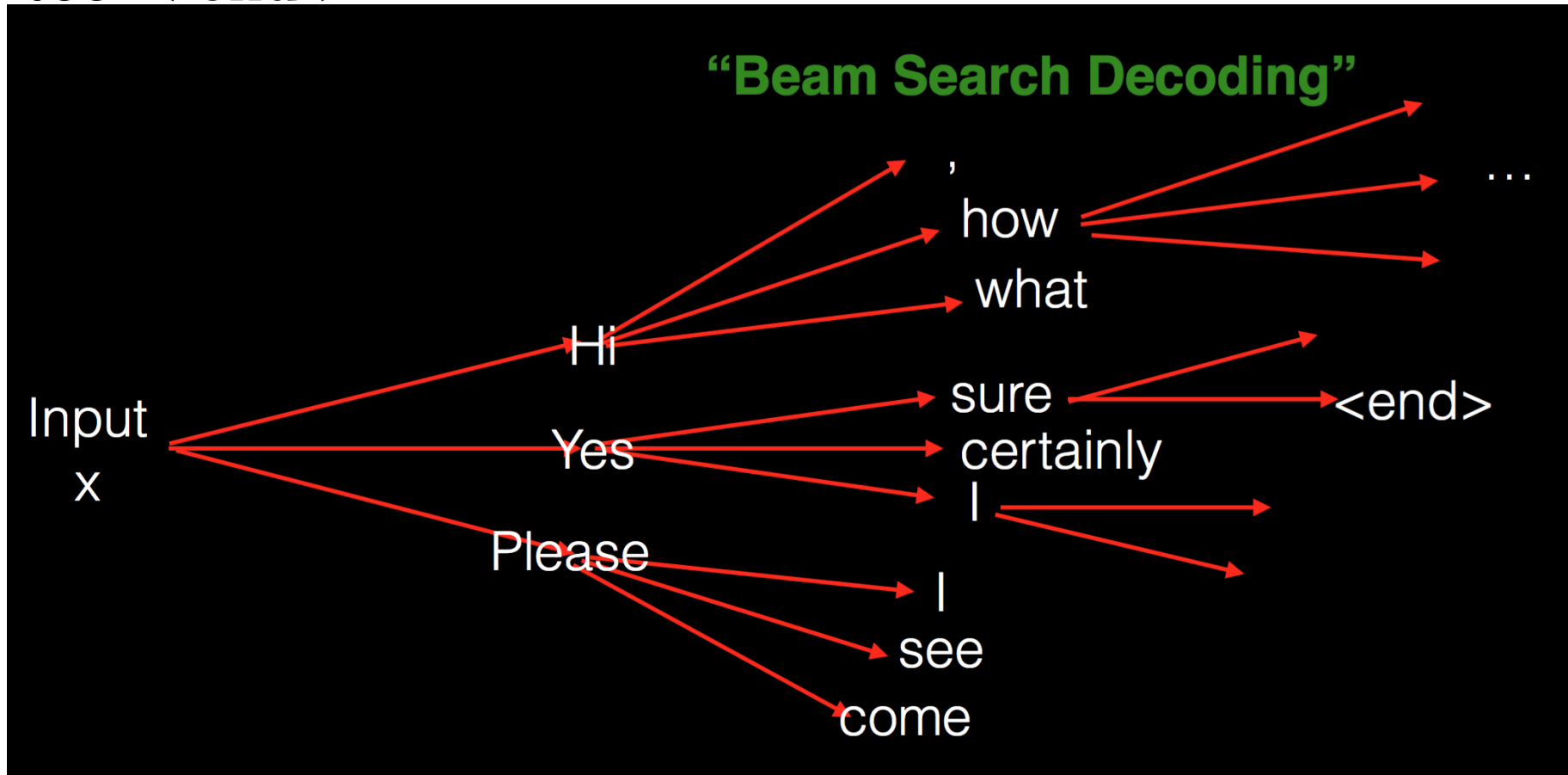
- Feed back the true output at each stage

Example III: Auto-Reply

- As we saw with image captioning example,
- Given input sequence x , we first output y_0 which has the highest probability
- Given x and y_0 , we output y_1 , which has the highest probability
- This is greedy
 - Does not correct for mistakes

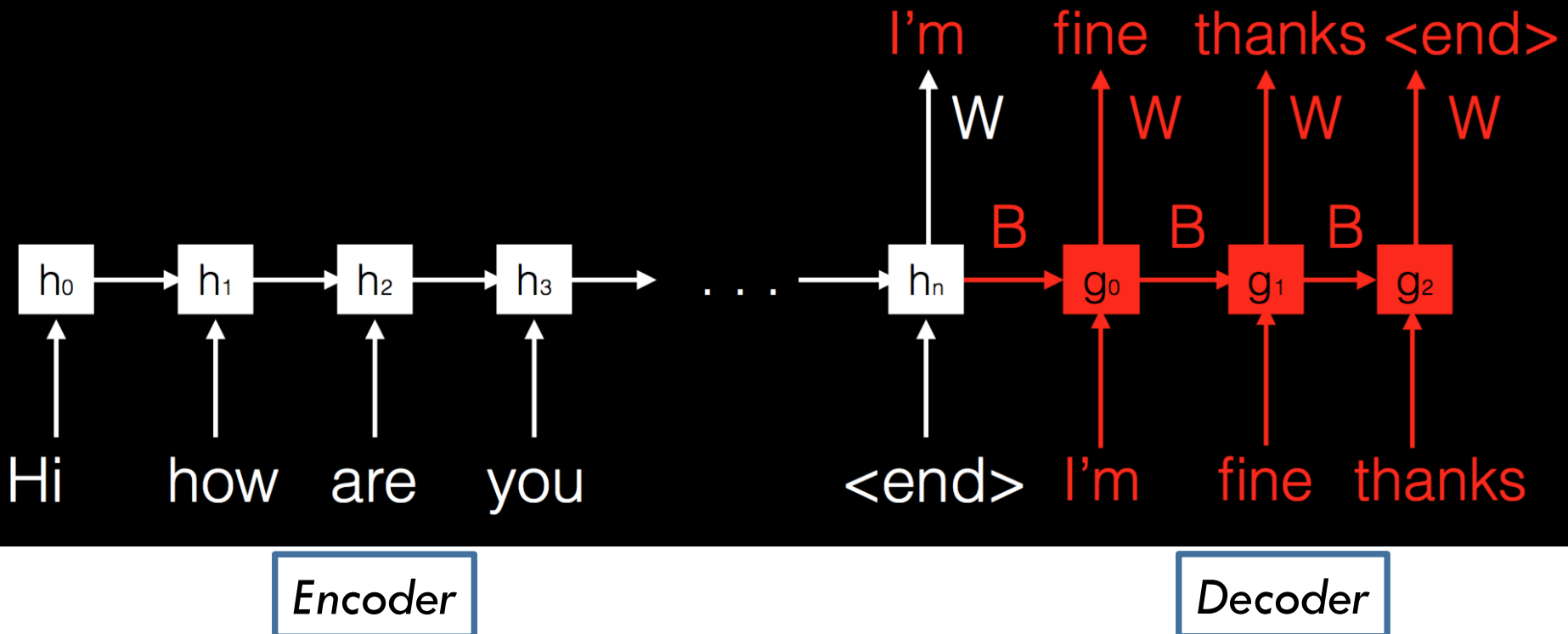
Example III: Auto-Reply

- Beam Search Decoding
- Retain k best candidate output sequences up to the time we see `< end >`



Example III: Auto-Reply

- Issue with second version: h_n is the only link
 - In fact, it is a fixed length vector. Whereas input is variable length
- Can be fixed with an 'attention' layer



Example IV: Speech Transcription

- Traditional pipeline has
 - Acoustic model $P(output|word)$
 - Language model $P(word)$
 - Feature engineering
 - ...
- Sequence to sequence learning can do ‘end-to-end’ without much feature engineering or blockwise modeling

Example IV: Speech Transcription

- What we want is the following



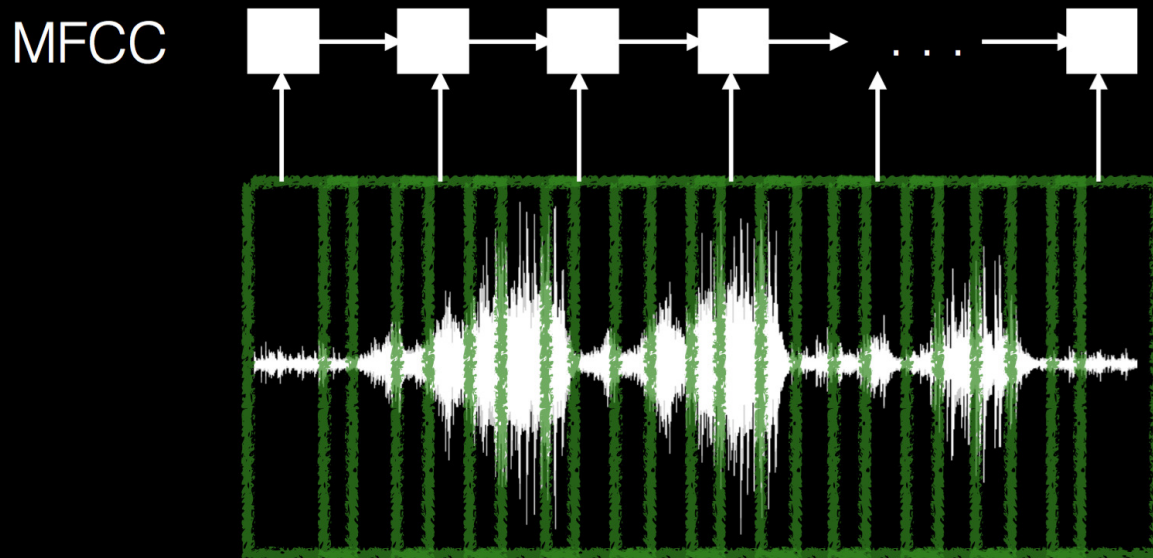
Example IV: Speech Transcription

- Step 1: Get some fixed length vectors



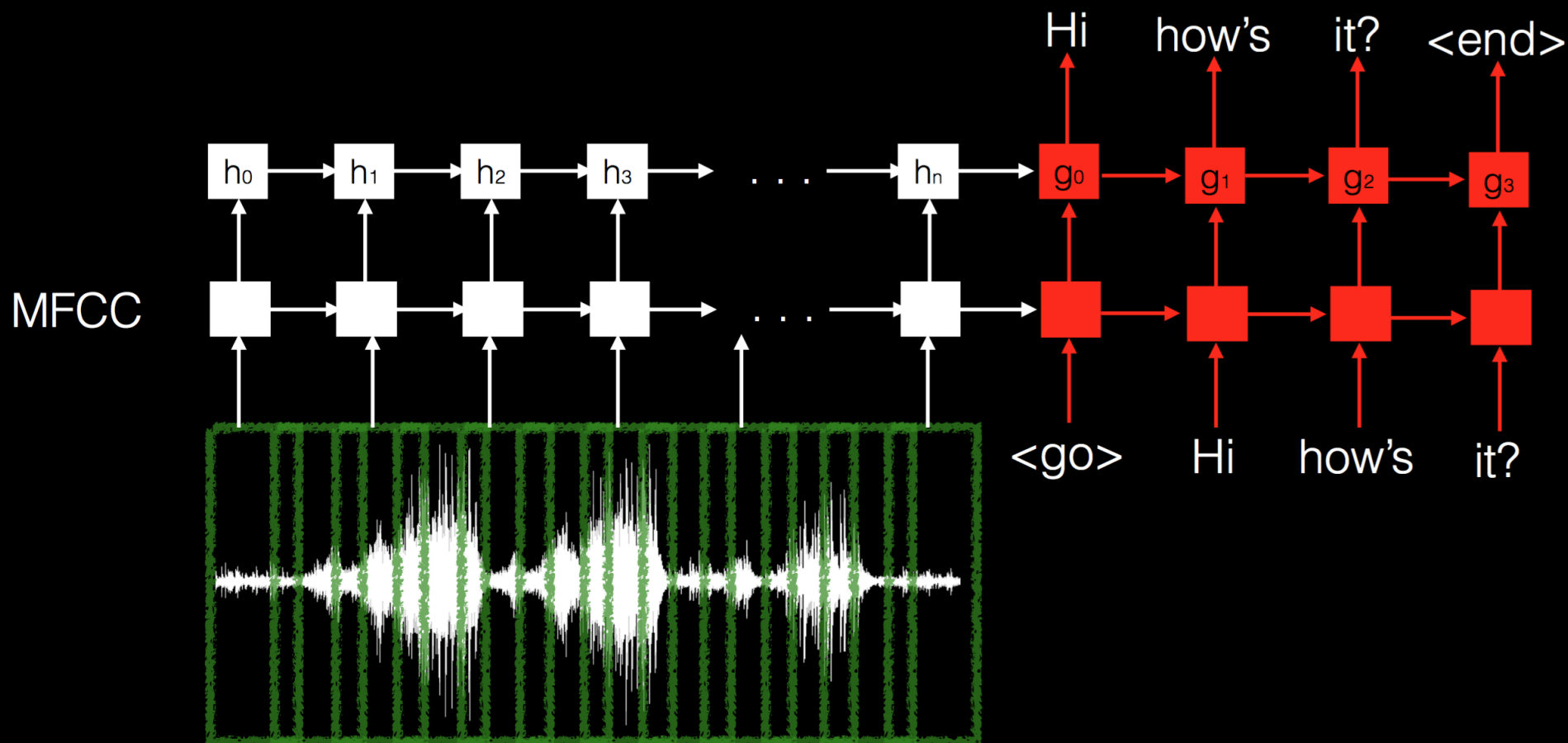
Example IV: Speech Transcription

- Step 2: Pass through an encoder



Example IV: Speech Transcription

- Step 3: Decode
- This is only a high level idea. Many many challenges.



Questions?

Summary

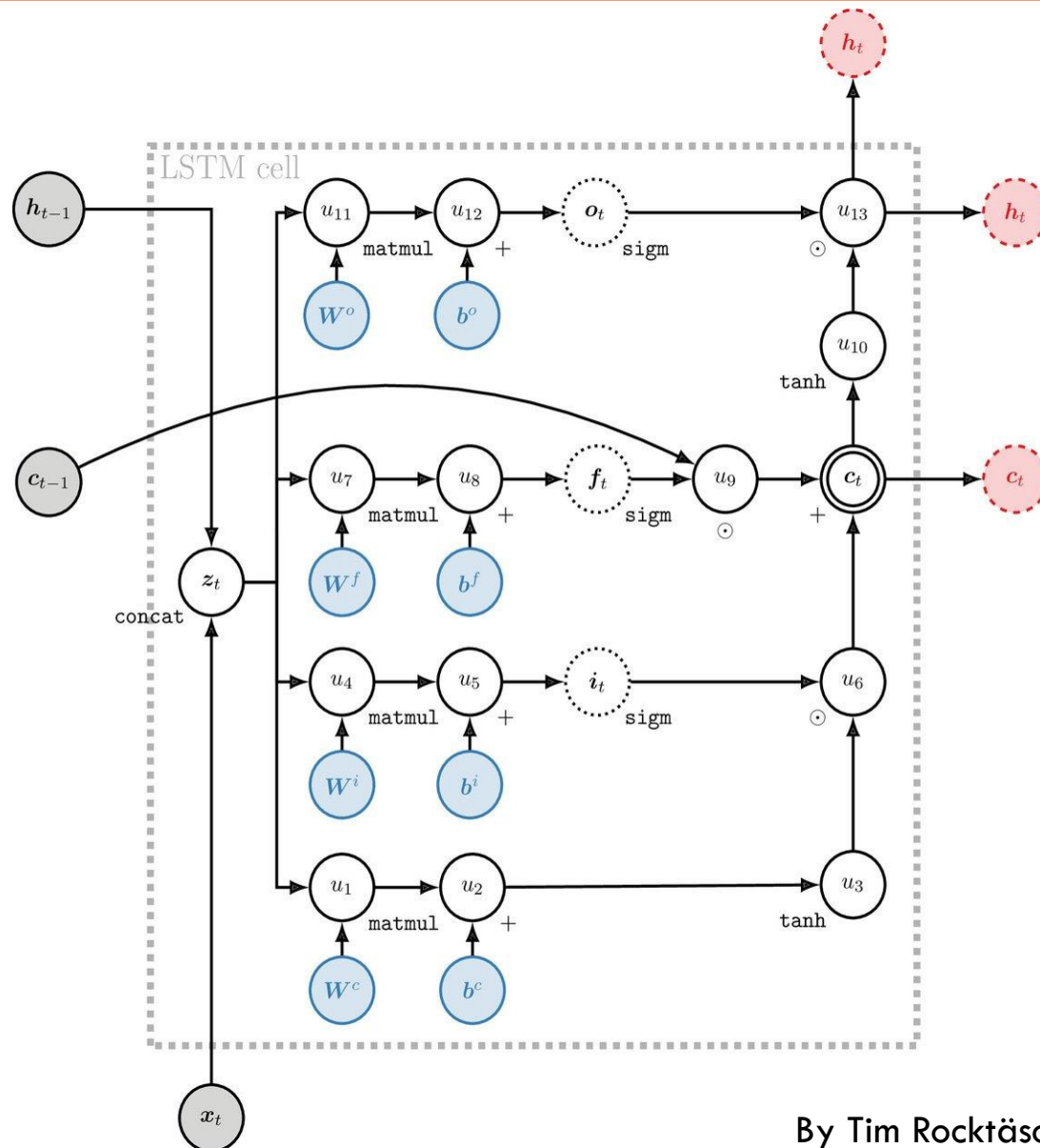
- Need for RNNs
- Understood the internals of RNNs (incl. LSTMs)
- Looked at some implementation details for the ‘sequence to sequence’ family of problems.
 - These significantly extend beyond classification

Appendix

Sample Exam Questions

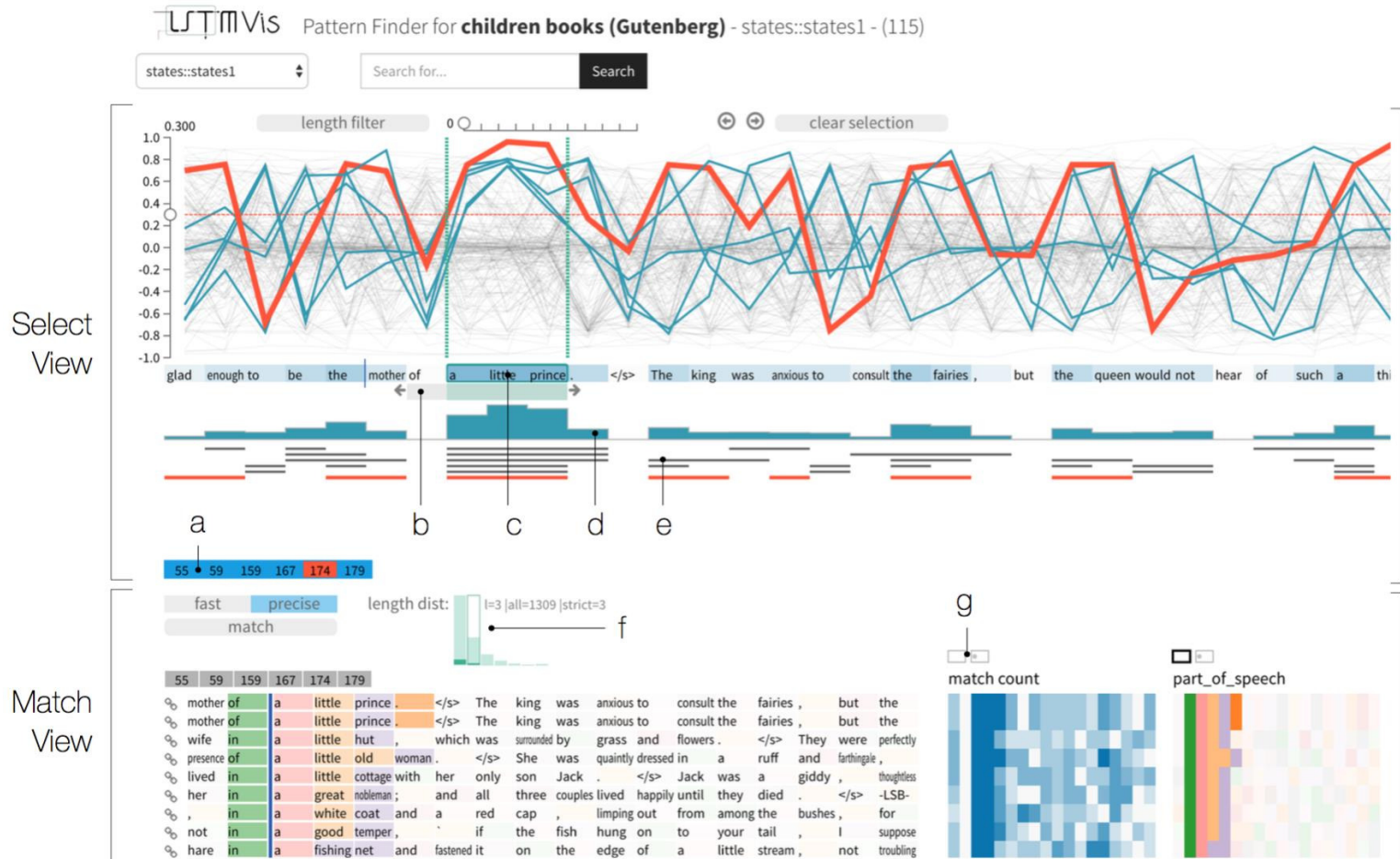
- What is the need for an RNN architecture?
- What shortcoming of vanilla RNNs does an LSTM RNN attempt to fix?
- Describe how sentence classification can be done with both an RNN and a CNN.

Yet Another Diagram of LSTM



Understanding LSTM: LSTMVis

- A visual tool to see which cell states do what

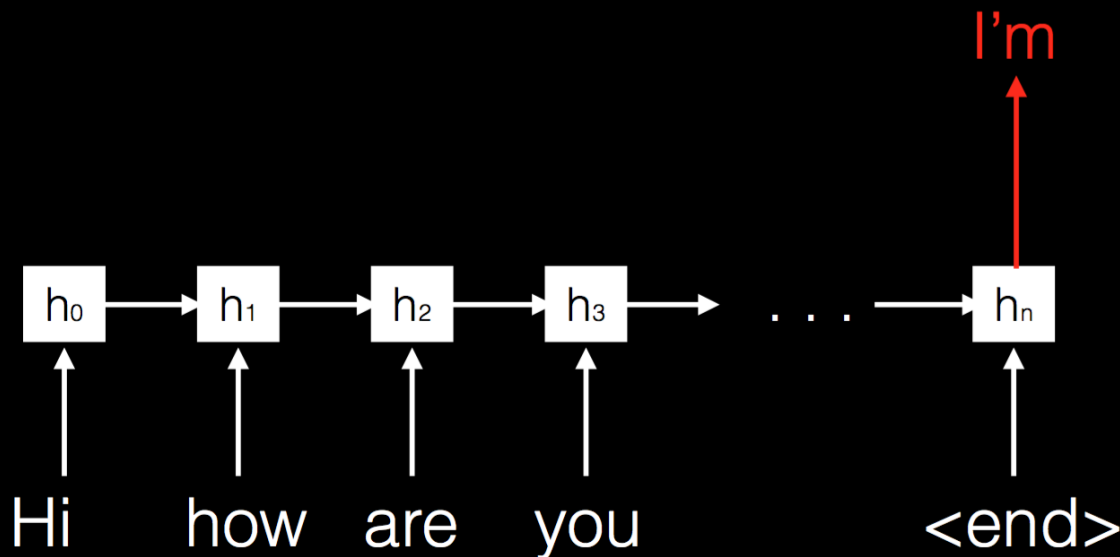


Tensorflow Seq2Seq/RNN Models

- For sequence to sequence modeling nuances, especially about how to deal with variable length training input and output data, see <https://www.tensorflow.org/tutorials/seq2seq/>

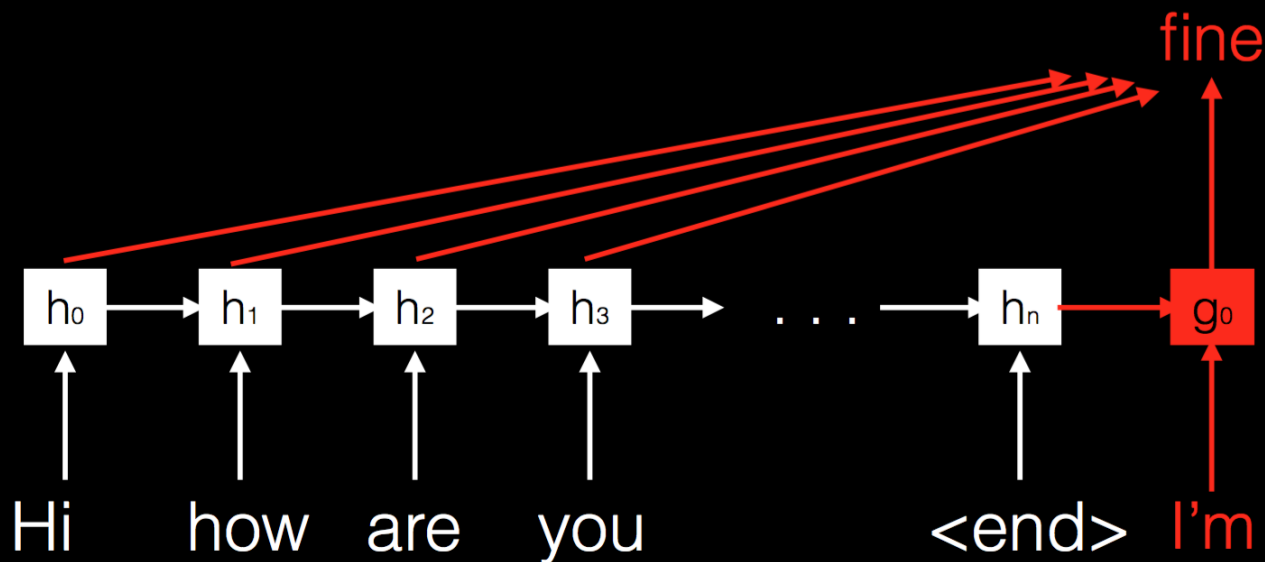
Example III (Extension): Auto-Reply

- Third version: Attention Mechanism
- Ideally output could consider ‘attention’ to parts of history



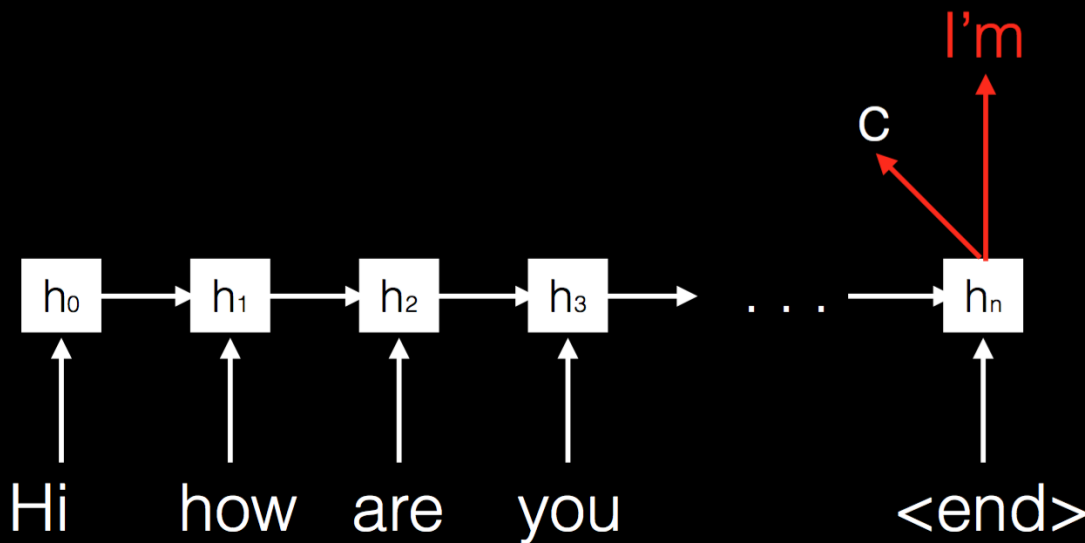
Example III (Extension): Auto-Reply

- Could look at every state in the past



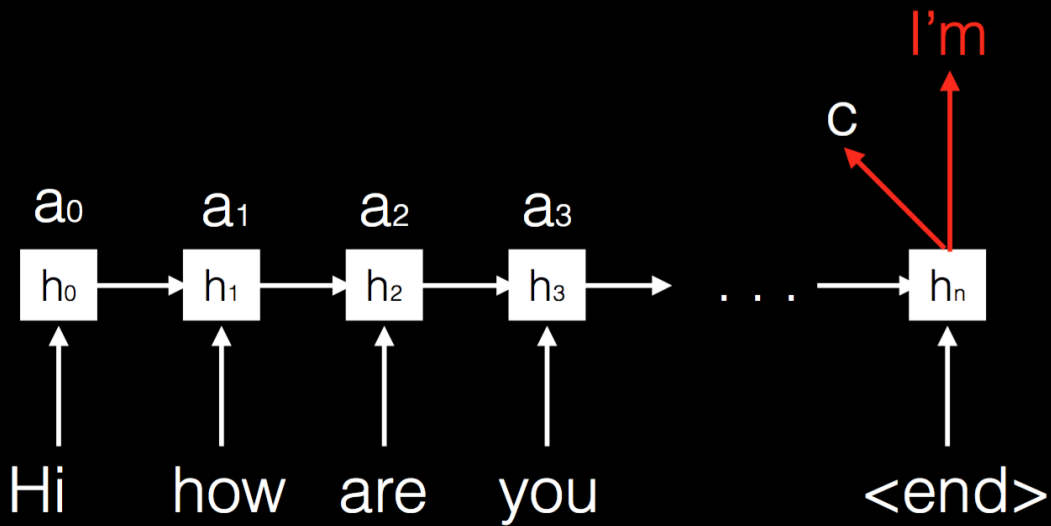
Example III (Extension): Auto-Reply

- So instead of returning a word, output the current state



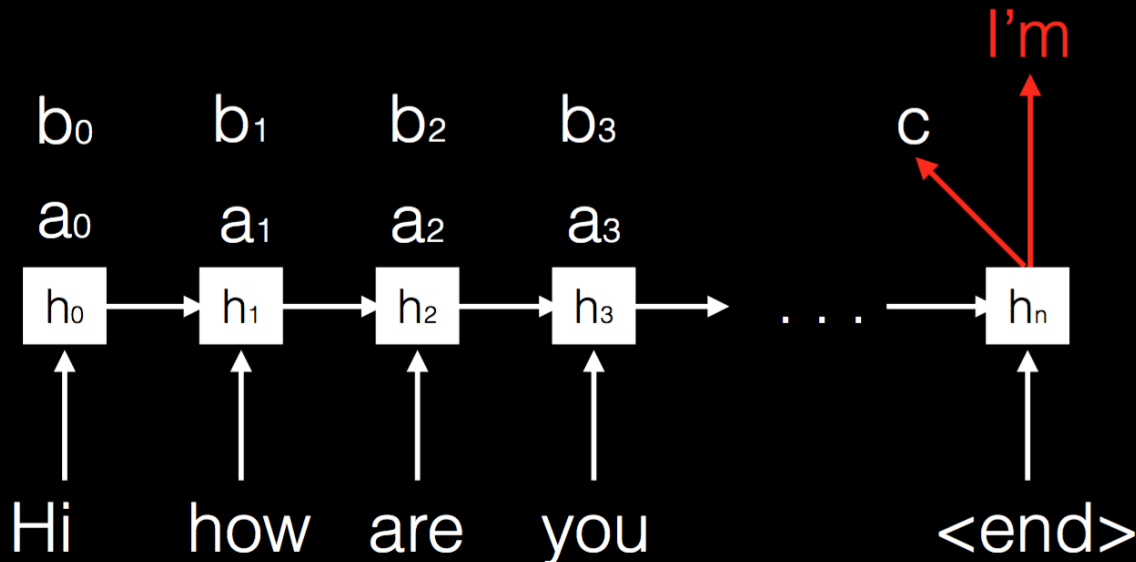
Example III (Extension): Auto-Reply

- Take inner products with previous states



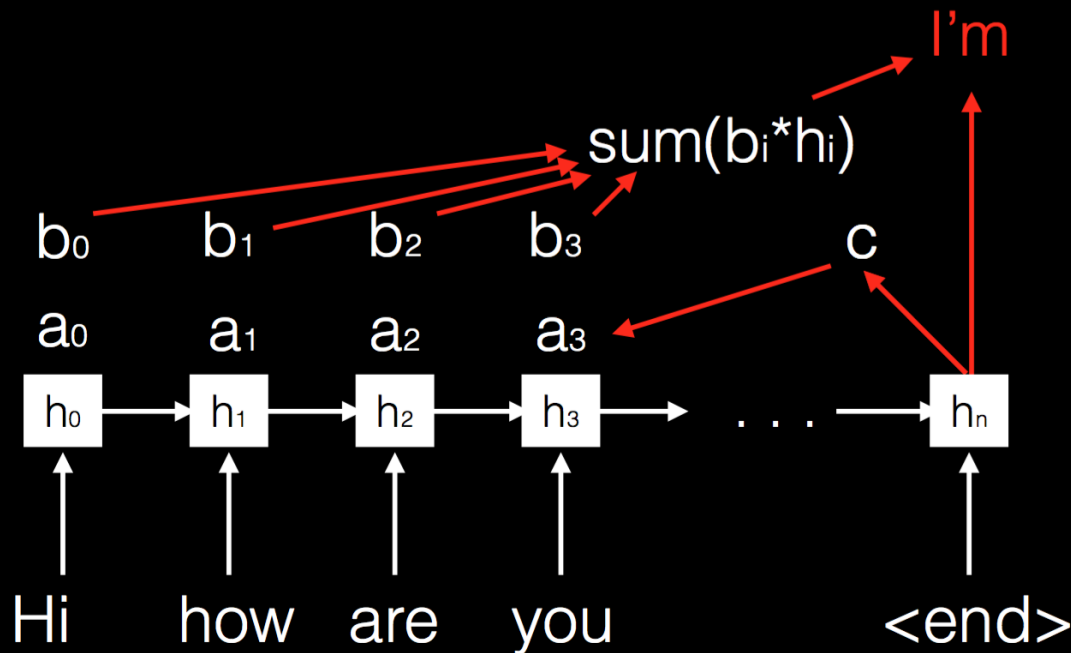
Example III (Extension): Auto-Reply

- Take inner products with previous states



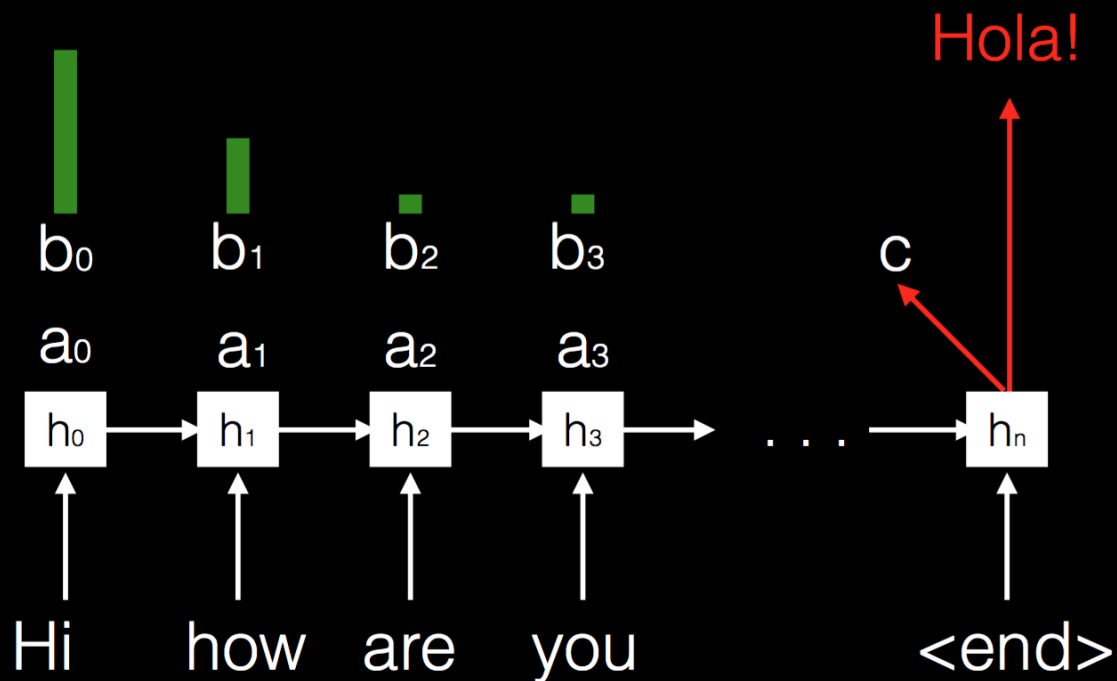
Example III (Extension): Auto-Reply

- Pass through a neural net layer to predict final word



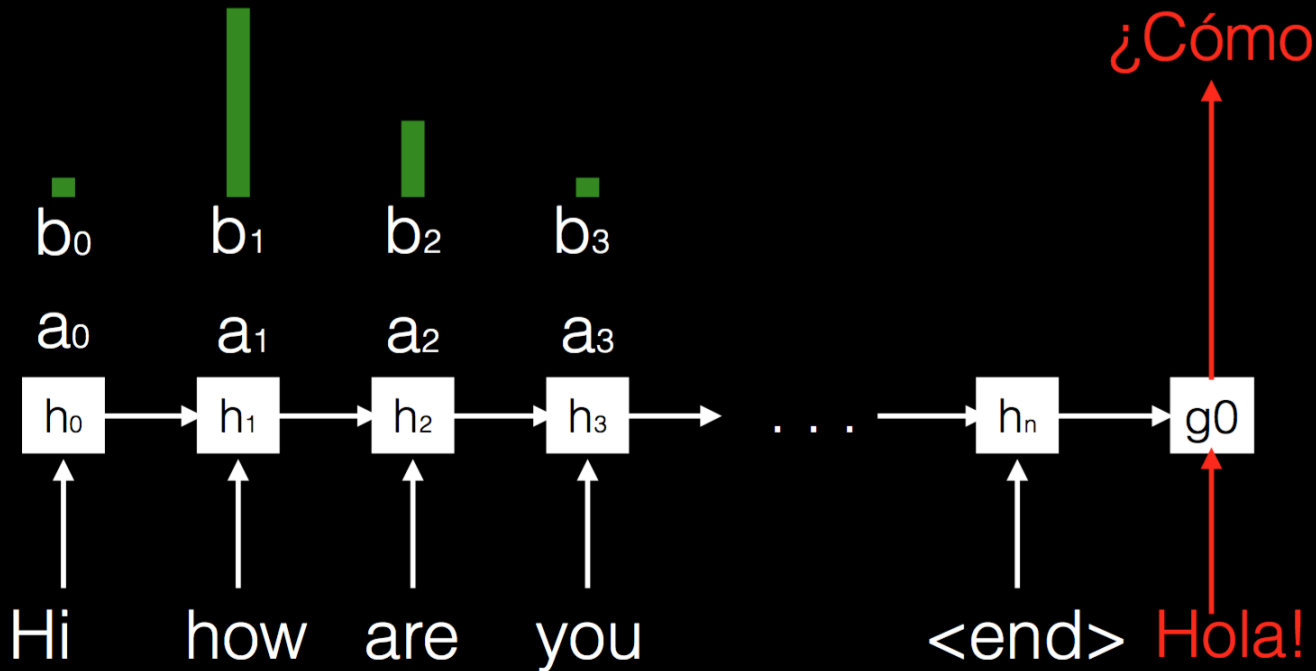
Example III (Extension): Same with Translation!

- Same principle also applies for translation. The first prediction learns to focus on certain part of the input



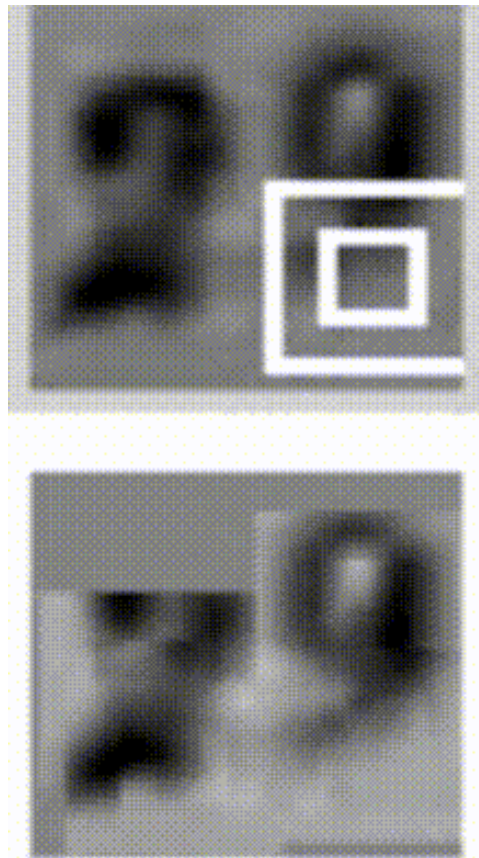
Example III (Extension): Auto-Reply

- The second prediction learns to focus on certain part of the input



Example V: Object Recognition with Visual Attention

- Even if we do not have sequences, we can still use RNNs to process the single fixed input in a sequence



¹Figure: <http://karpathy.github.io/2015/05/21/rnn-effectiveness/>

²Reference: <http://arxiv.org/abs/1412.7755>

Advanced Prediction Models

Deep Learning, Graphical Models and Reinforcement
Learning

Today's Outline

- Unsupervised Learning Landscape
- Autoencoders and Variational Autoencoders (VAE)
- Generative Adversarial Networks (GAN)

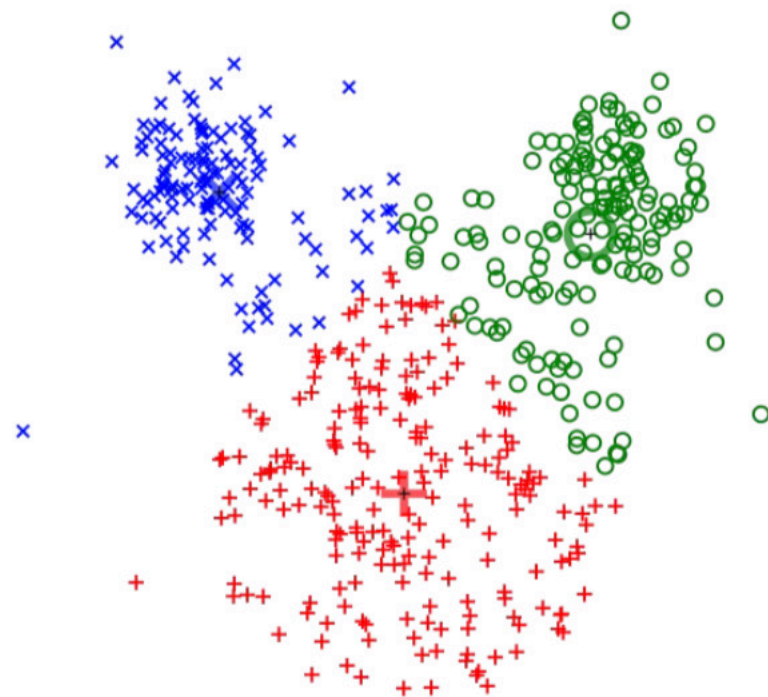
Unsupervised Learning Landscape

Unsupervised Learning

- Supervised learning
 - Involves feature and label pairs as training data
 - Goal is to find a map from feature to label/value
- Unsupervised learning
 - Involves only feature vectors
 - Example: images
 - Goal is to learn some patterns of data
 - There is **no objective measure of success**

Unsupervised Learning Tasks

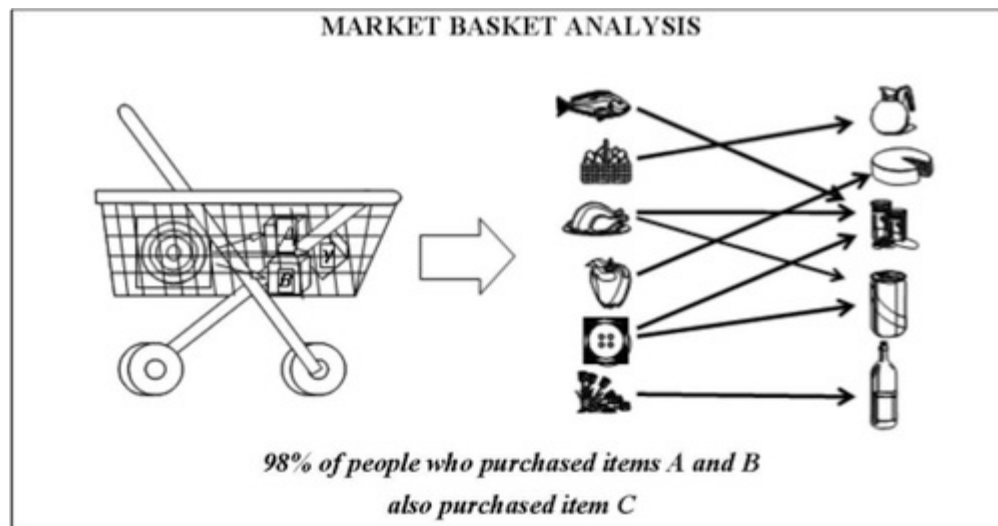
- Clustering
- Association rules
- Dimensionality reduction
- Density estimation
- Embedding
- Sampling



K-means clustering

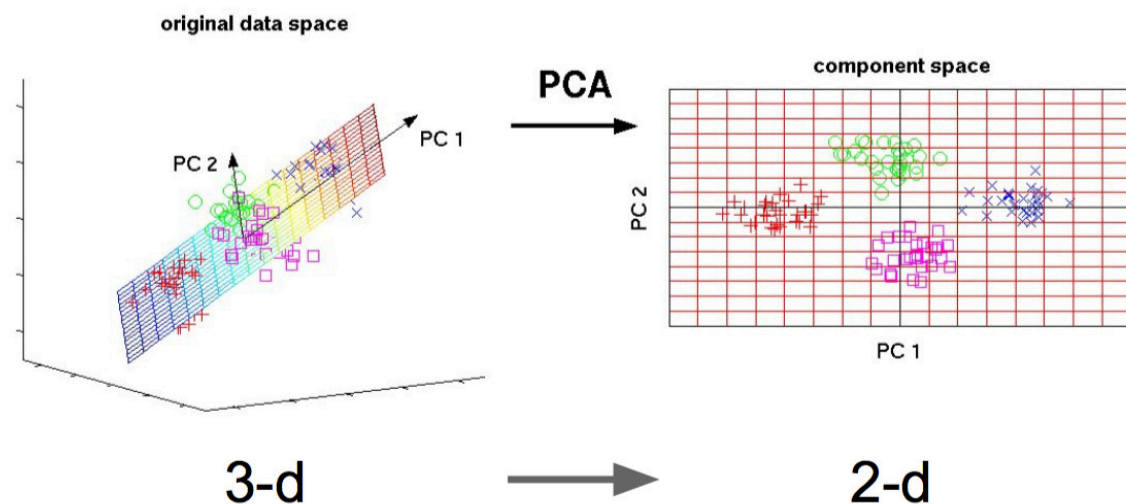
Unsupervised Learning Tasks

- Clustering
- Association rules
- Dimensionality reduction
- Density estimation
- Embedding
- Sampling



Unsupervised Learning Tasks

- Clustering
- Association rules
- Dimensionality reduction
- Density estimation
- Embedding
- Sampling



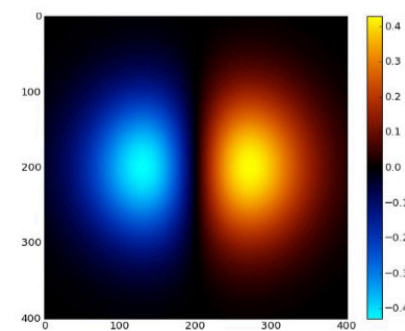
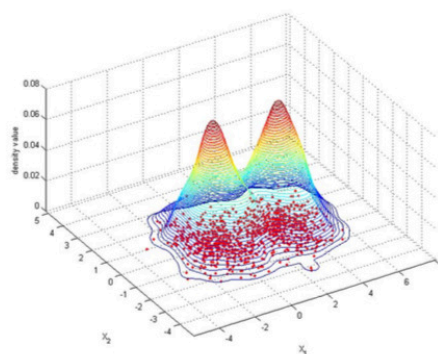
Unsupervised Learning Tasks

- Clustering
- Association rules
- Dimensionality reduction
- Density estimation
- Embedding
- Sampling



Figure copyright Ian Goodfellow, 2016. Reproduced with permission.

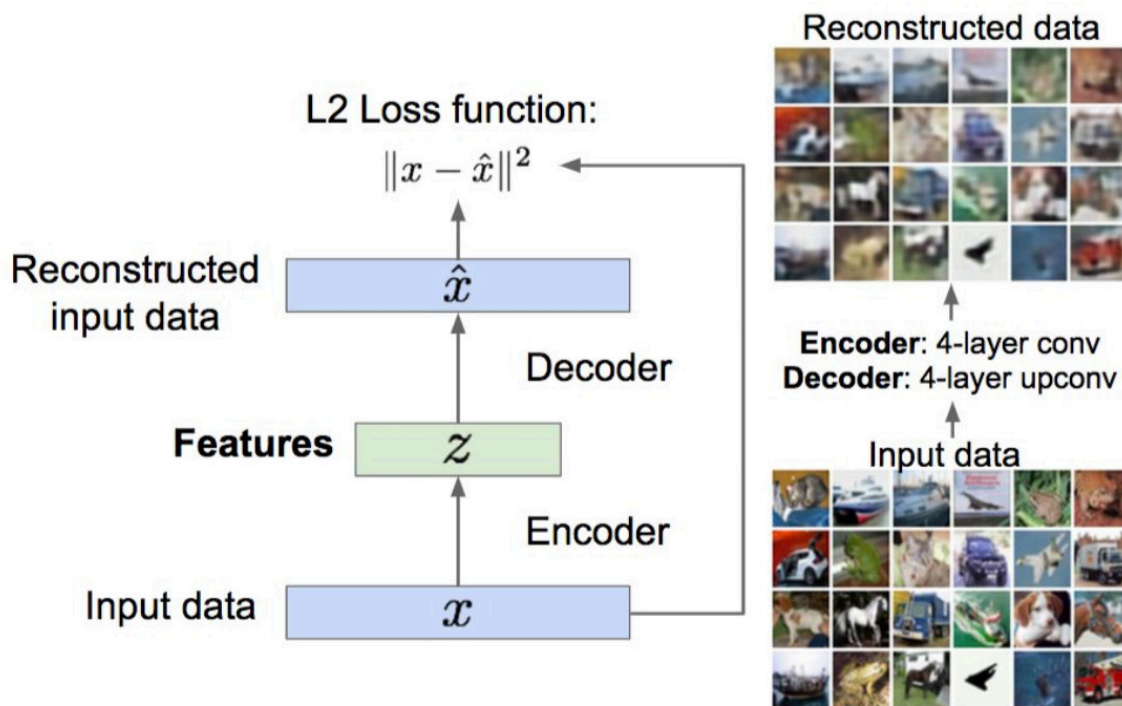
1-d density estimation



2-d density estimation

Unsupervised Learning Tasks

- Clustering
- Association rules
- Dimensionality reduction
- Density estimation
- Embedding
- Sampling



Unsupervised Learning Tasks

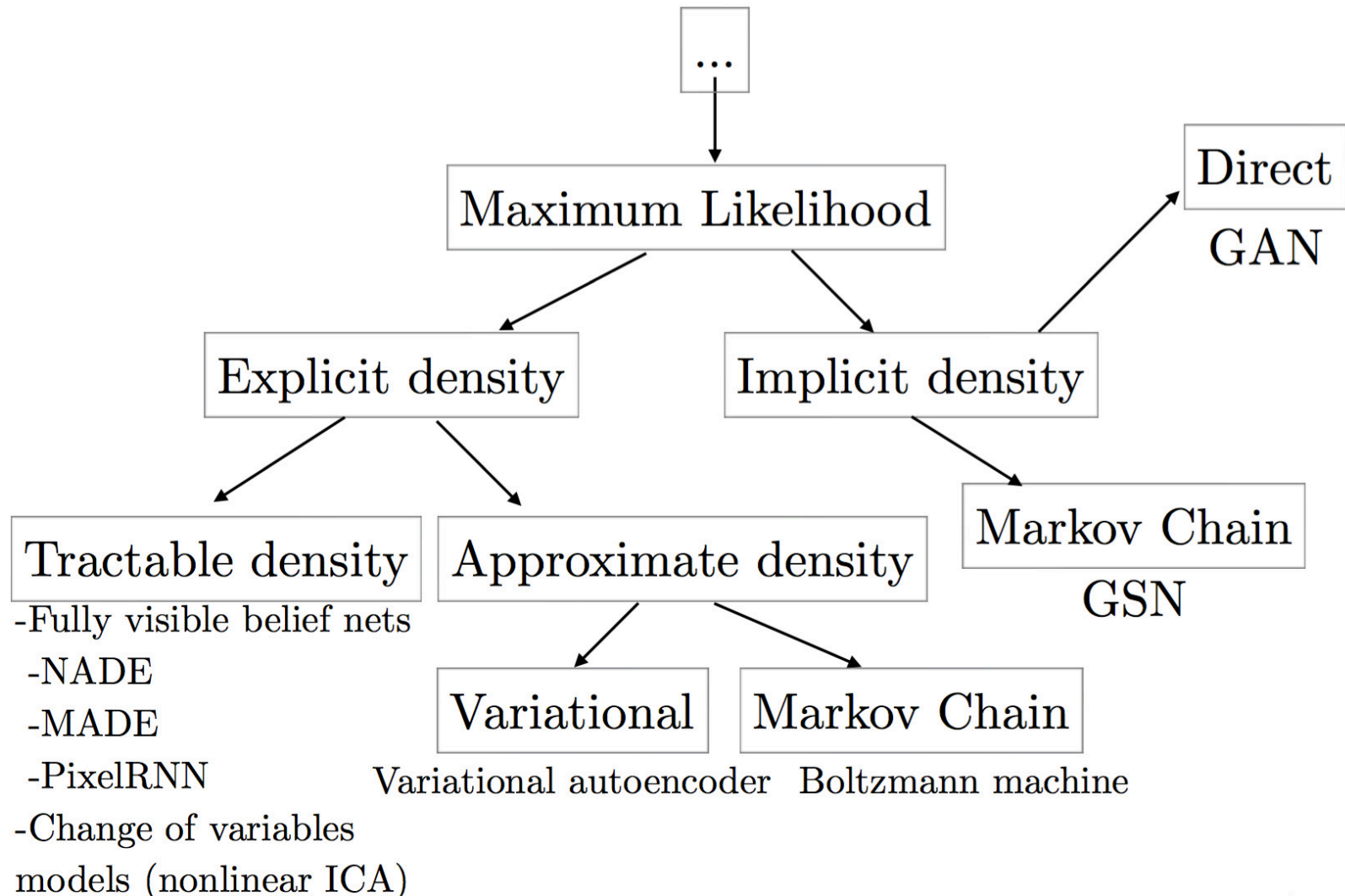
- Clustering
- Association rules
- Dimensionality reduction
- Density estimation
- Embedding
- Sampling



Learning a Distribution

- Given (large amount of) data drawn from P_d , we want to estimate P_m such that samples from P_m are as similar as possible to samples from P_d
- Two approaches:
 - Explicit
 - If we construct P_m explicitly, we can address all the other tasks mentioned
 - Implicit
 - We can directly generate a sample from P_m without explicitly defining it!

Explicit and Implicit Approaches



Explicit and Implicit Approaches

- When would we be okay with an implicit approach
 - Simulate possible futures for planning
 - When samples themselves are useful for other tasks...



Explicit and Implicit Approaches

- When would we be okay with an implicit approach
 - Simulate possible futures for planning
 - When samples themselves are useful for other tasks...

original



bicubic
(21.59dB/0.6423)



SRResNet
(23.44dB/0.7777)

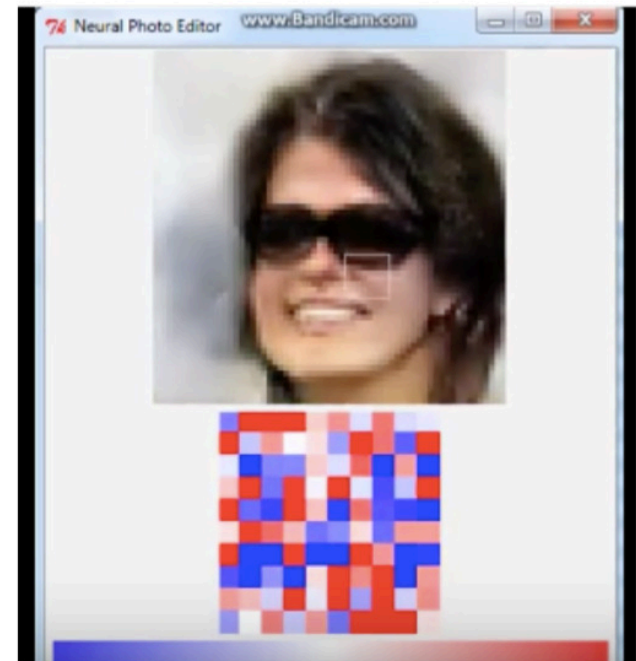
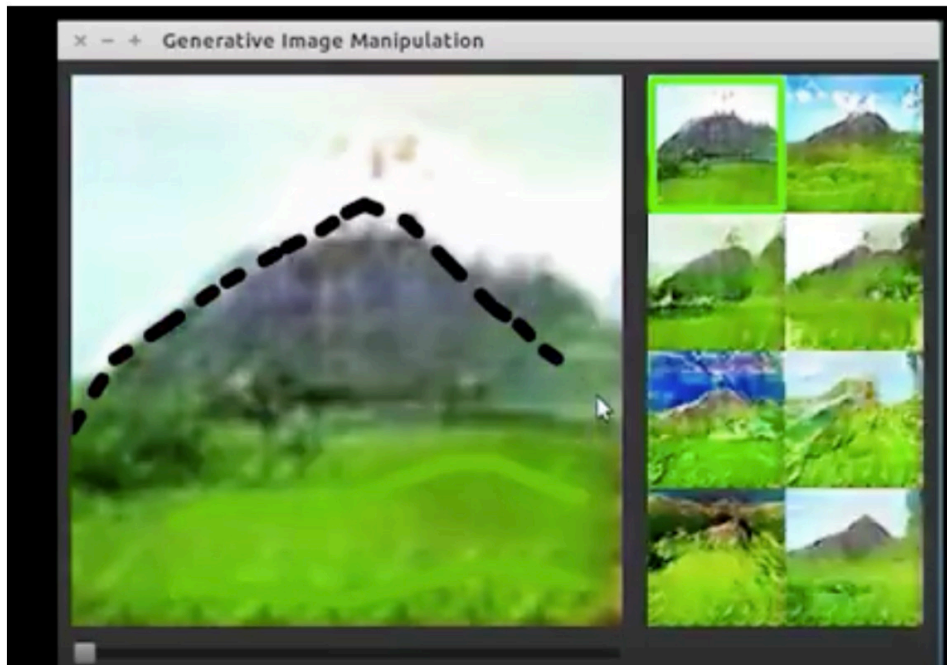


SRGAN
(20.34dB/0.6562)



Explicit and Implicit Approaches

- When would we be okay with an implicit approach
 - Simulate possible futures for planning
 - When samples themselves are useful for other tasks...



Explicit and Implicit Approaches

- When would we be okay with an implicit approach
 - Simulate possible futures for planning
 - When samples themselves are useful for other tasks...

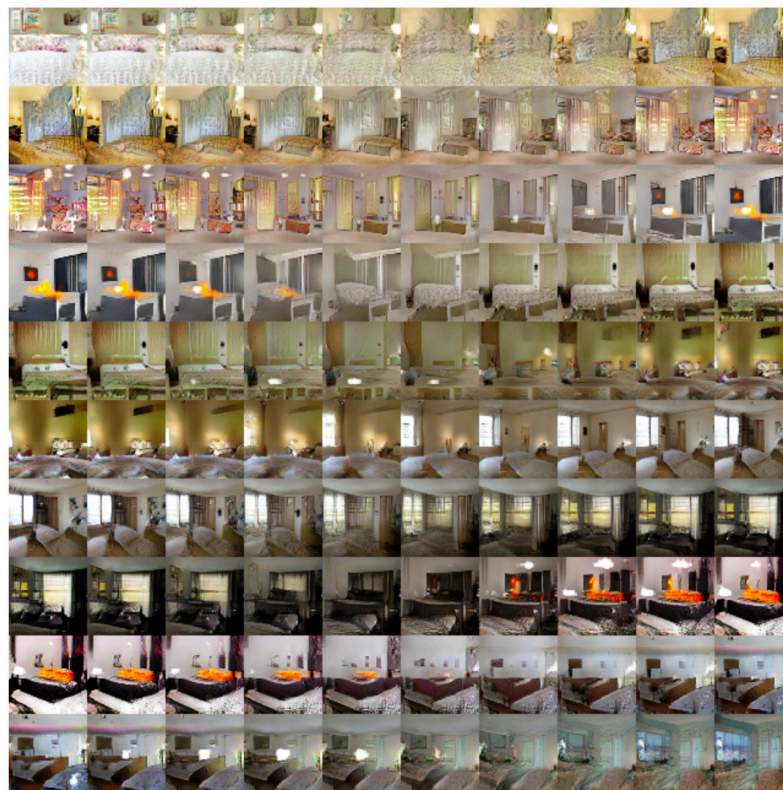


Explicit and Implicit Approaches

- We will look at one model under each approach and work with **image** data
 - Explicit: Variational Autoencoders (VAE)
 - Implicit: Generative Adversarial Networks (GAN)
- Both use **neural networks** as a core object
 - Their number of parameters should be smaller than the amount of training data, so the models discover some intrinsic aspect of the data

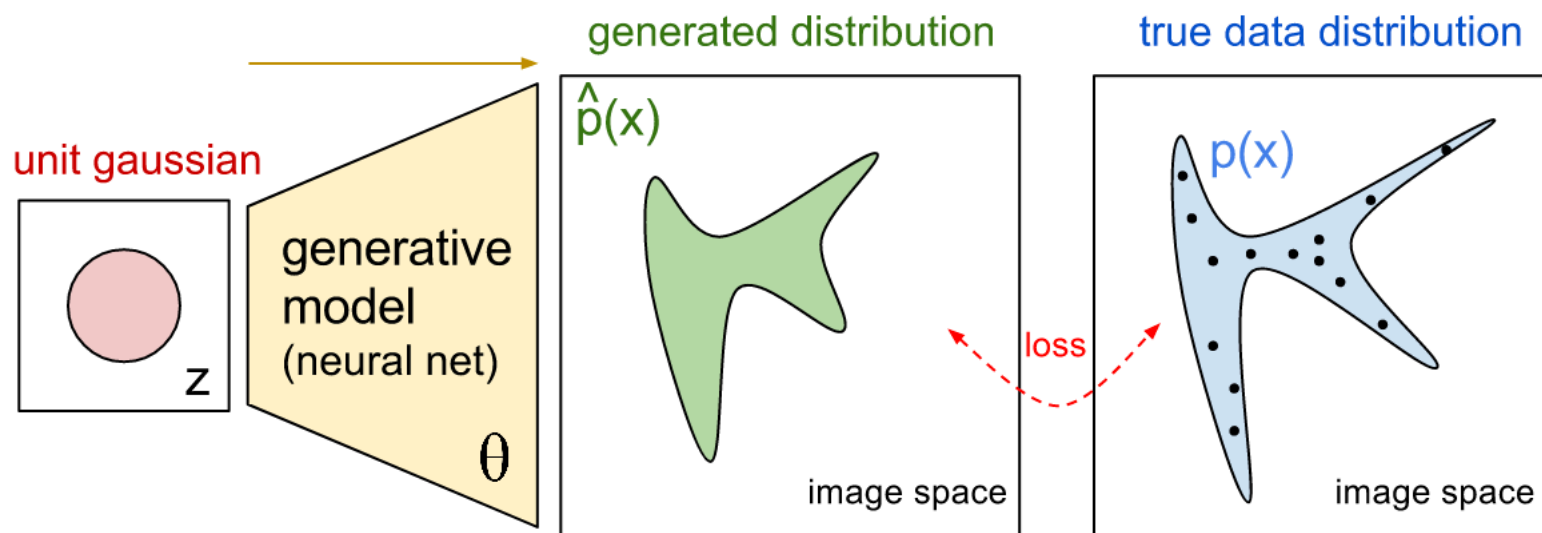
More than Memorization

- Consider Deep Convolutional Generative Adversarial Network (DCGAN)
 - You can walk from one point to another in the bedroom latent space (e.g., 6th and 10th rows)



More than Memorization

- Either model (VAE or GAN) will essentially build the yellow box below:



Questions?

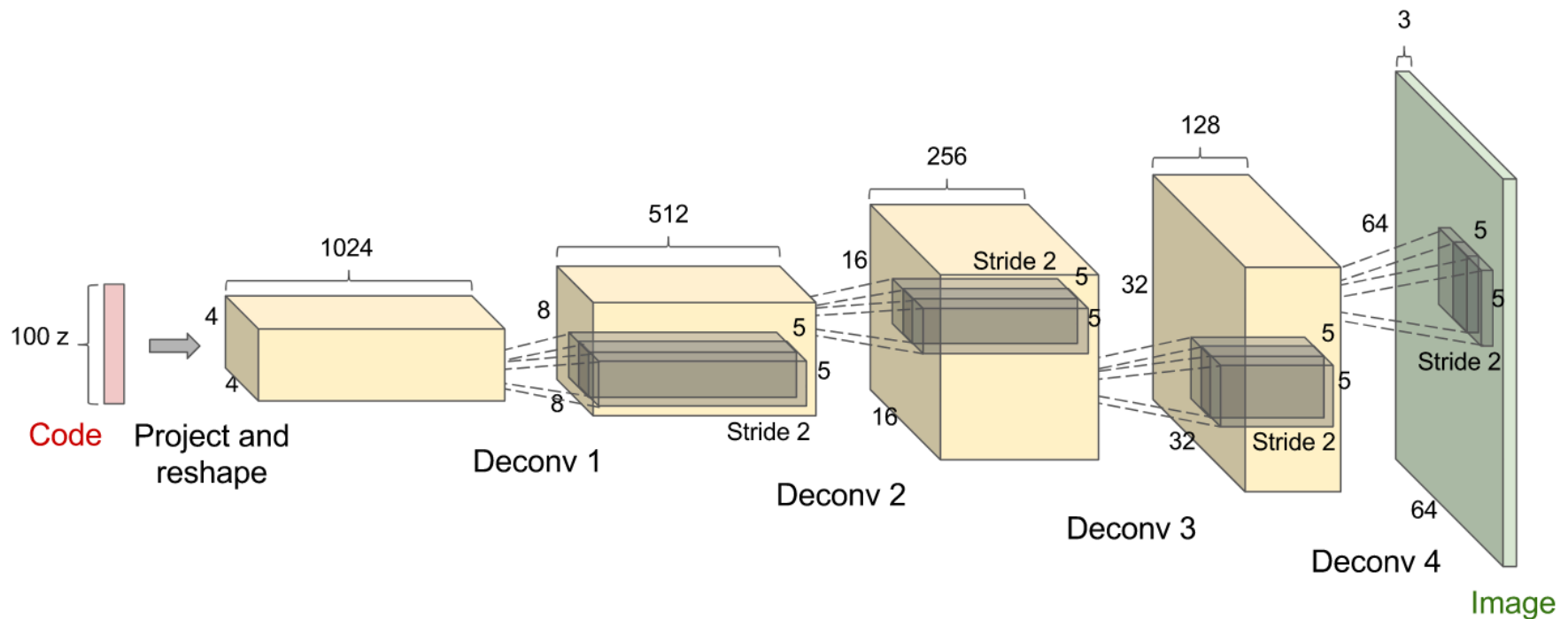
Today's Outline

- Unsupervised Learning Landscape
- Autoencoders and Variational Autoencoders (VAE)
- Generative Adversarial Networks (GAN)

Autoencoders and Variational Autoencoders

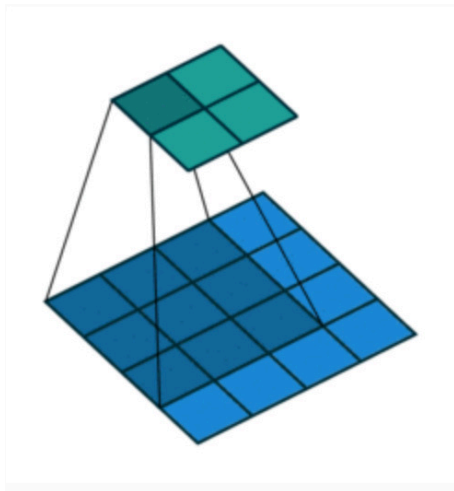
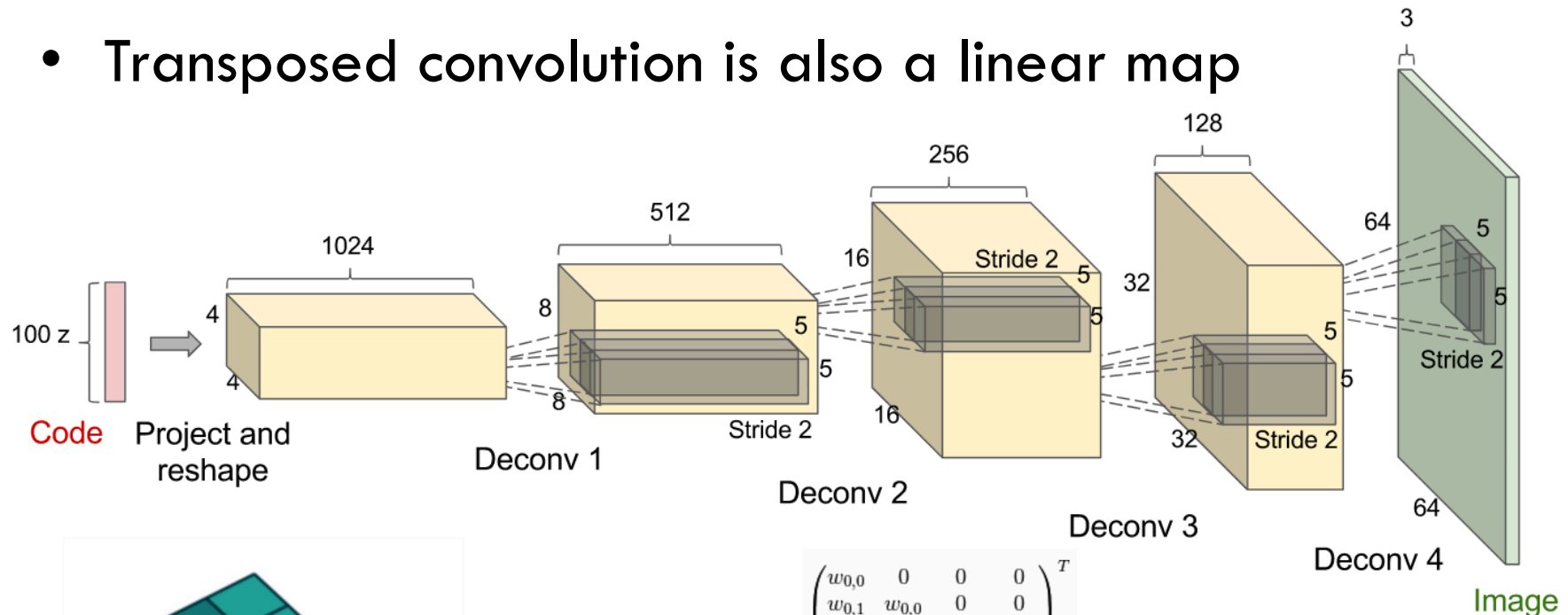
Neural Net as a Transformation Map

- NN is a function that maps an input to output
- Here is a deconvolutional/transposed-convolutional network



Neural Net as a Transformation Map

- Transposed convolution is also a linear map



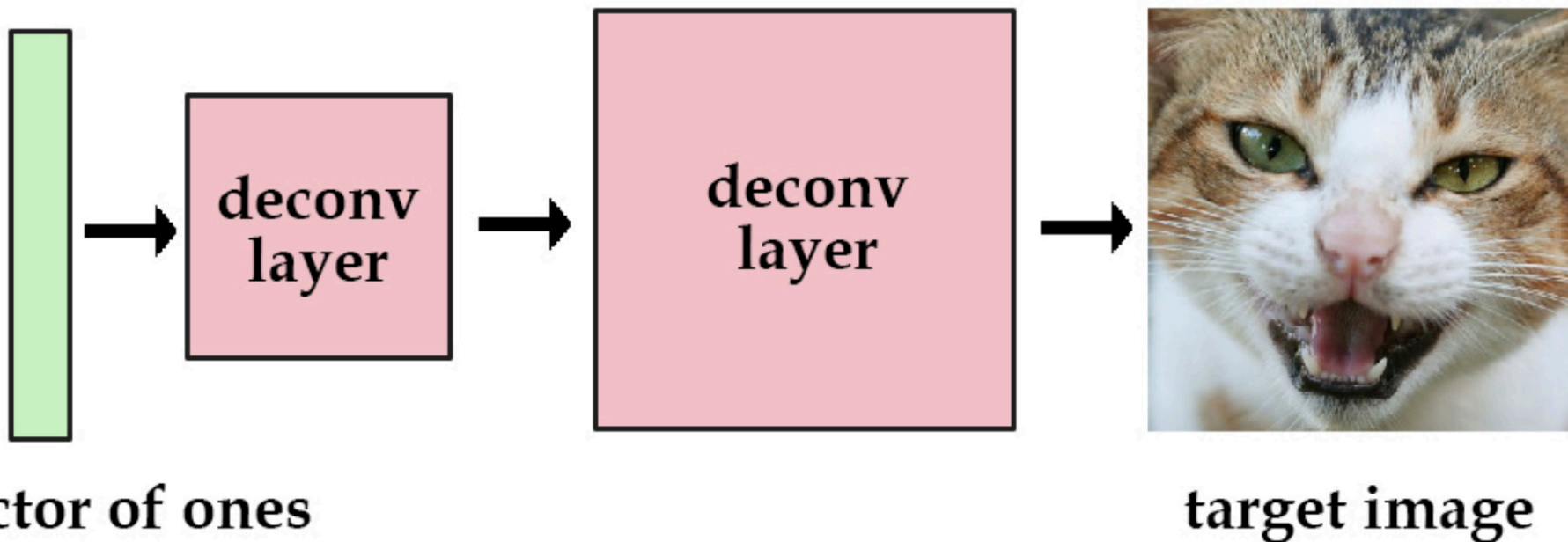
16-dim vec =

$$\begin{pmatrix} w_{0,0} & 0 & 0 & 0 \\ w_{0,1} & w_{0,0} & 0 & 0 \\ w_{0,2} & w_{0,1} & 0 & 0 \\ 0 & w_{0,2} & 0 & 0 \\ w_{1,0} & 0 & w_{0,0} & 0 \\ w_{1,1} & w_{1,0} & w_{0,1} & w_{0,0} \\ w_{1,2} & w_{1,1} & w_{0,2} & w_{0,1} \\ 0 & w_{1,2} & 0 & w_{0,2} \\ w_{2,0} & 0 & w_{1,0} & 0 \\ w_{2,1} & w_{2,0} & w_{1,1} & w_{1,0} \\ w_{2,2} & w_{2,1} & w_{1,2} & w_{1,1} \\ 0 & w_{2,2} & 0 & w_{1,2} \\ 0 & 0 & w_{2,0} & 0 \\ 0 & 0 & w_{2,1} & w_{2,0} \\ 0 & 0 & w_{2,2} & w_{2,1} \\ 0 & 0 & 0 & w_{2,2} \end{pmatrix}^T$$

*4-dim vec

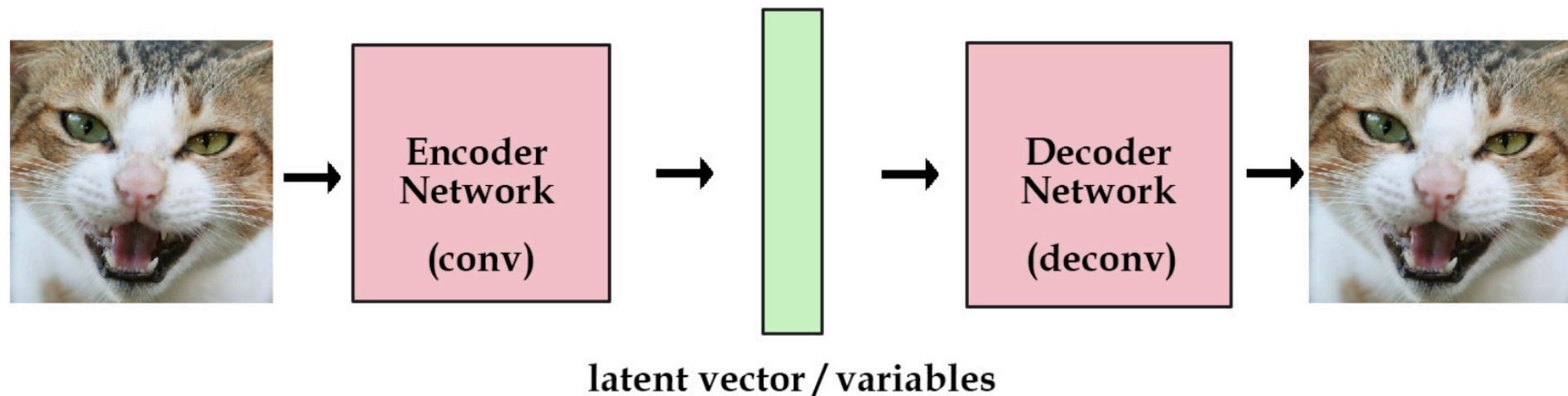
Transformation from a Single Vector

- For example, set inputs to all ones
- Train network to reduce MSE between its output and target image
- Then information related to image is captured in network parameters



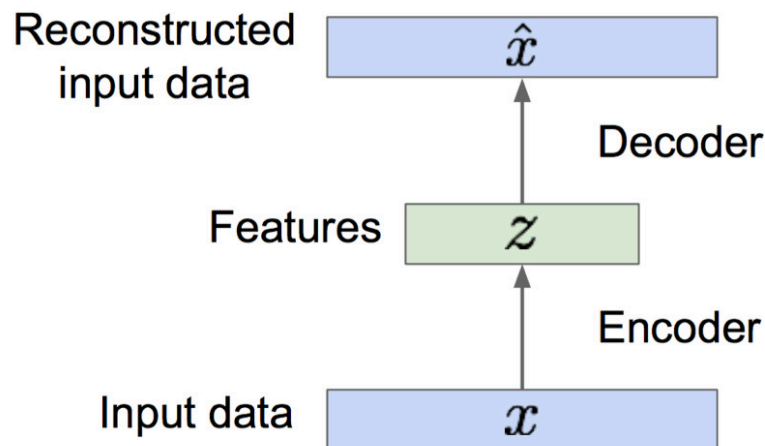
Transformation from Multiple Vectors

- Do the same with multiple input vectors (e.g., one hot encoded)
- These input vectors are called codes. The network is called a decoder.
- In an autoencoder, we also have an 'encoder' that takes original images and 'codes' them



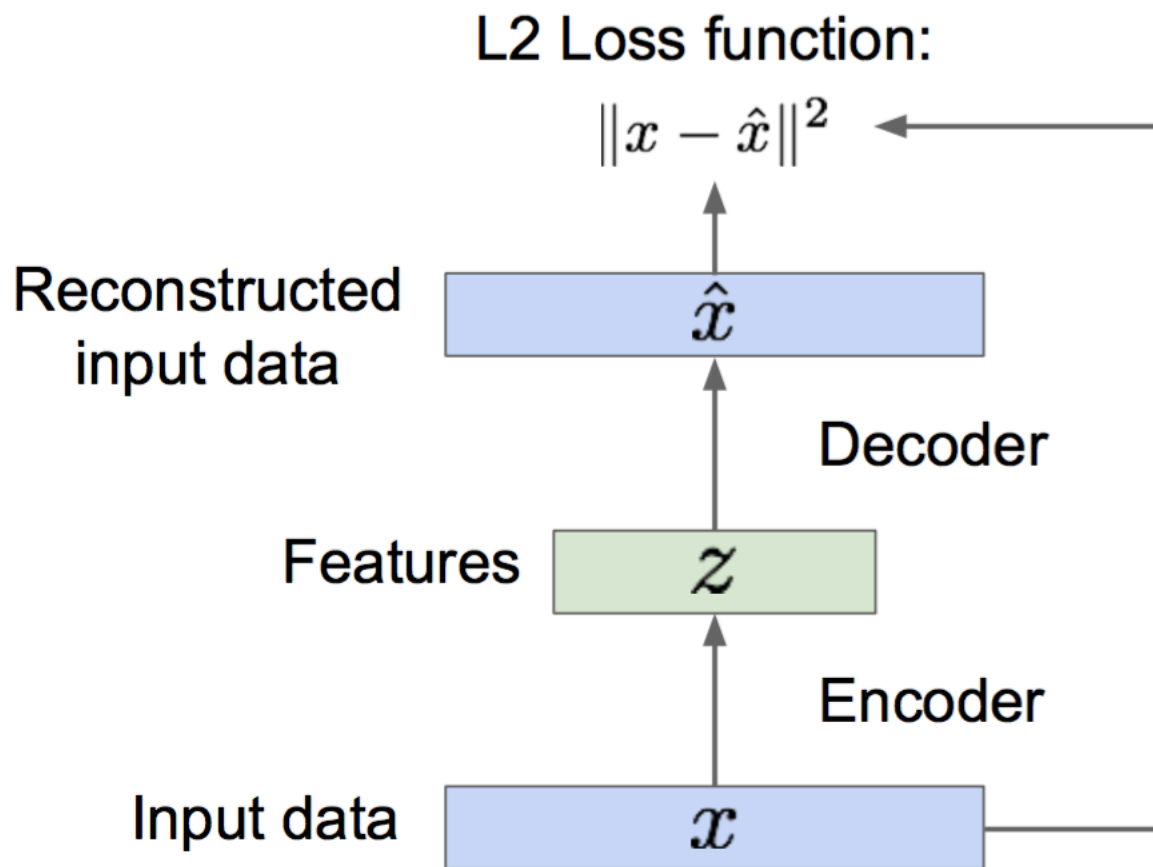
Autoencoder: The Objective

- Captures information in training data
- The latent variable z (also called code) can be thought of as embedding
- Keep the dimension of z smaller than input x , otherwise we have a trivial solution
 - If we choose a larger dimension, add noise to x during training (this is called a denoising autoencoder)

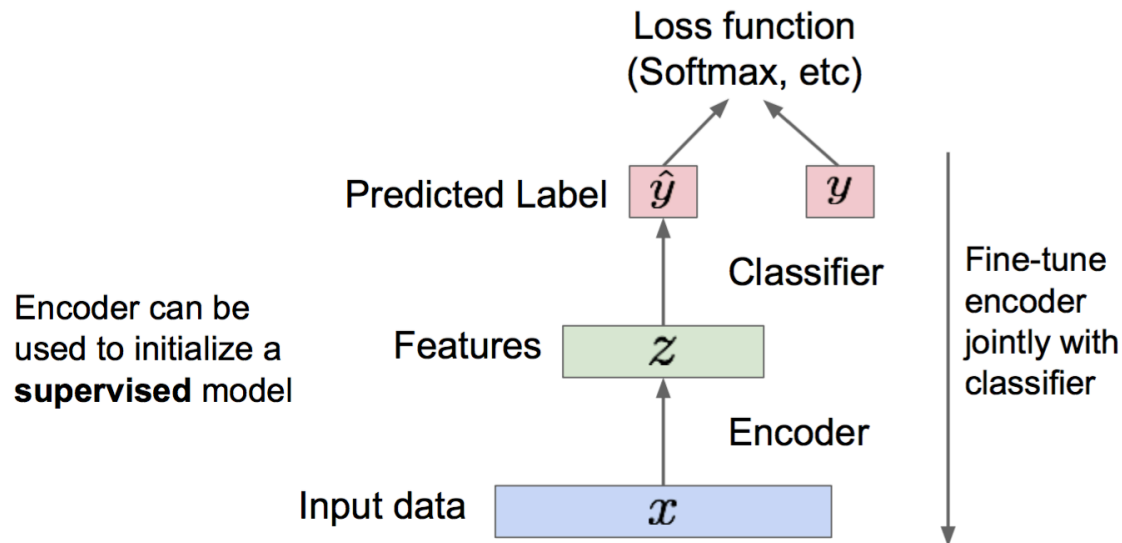


Autoencoder: The Architecture

- No labels are needed here



Autoencoder: Uses



- Reduction in dimension achieved by the encoder is useful
 - Just like PCA
 - Captures meaningful variations in the data via the embeddings
- Named 'autoencoder' because it attempts to reconstruct original data
- **Cannot generate new samples yet!**

Variational Autoencoder

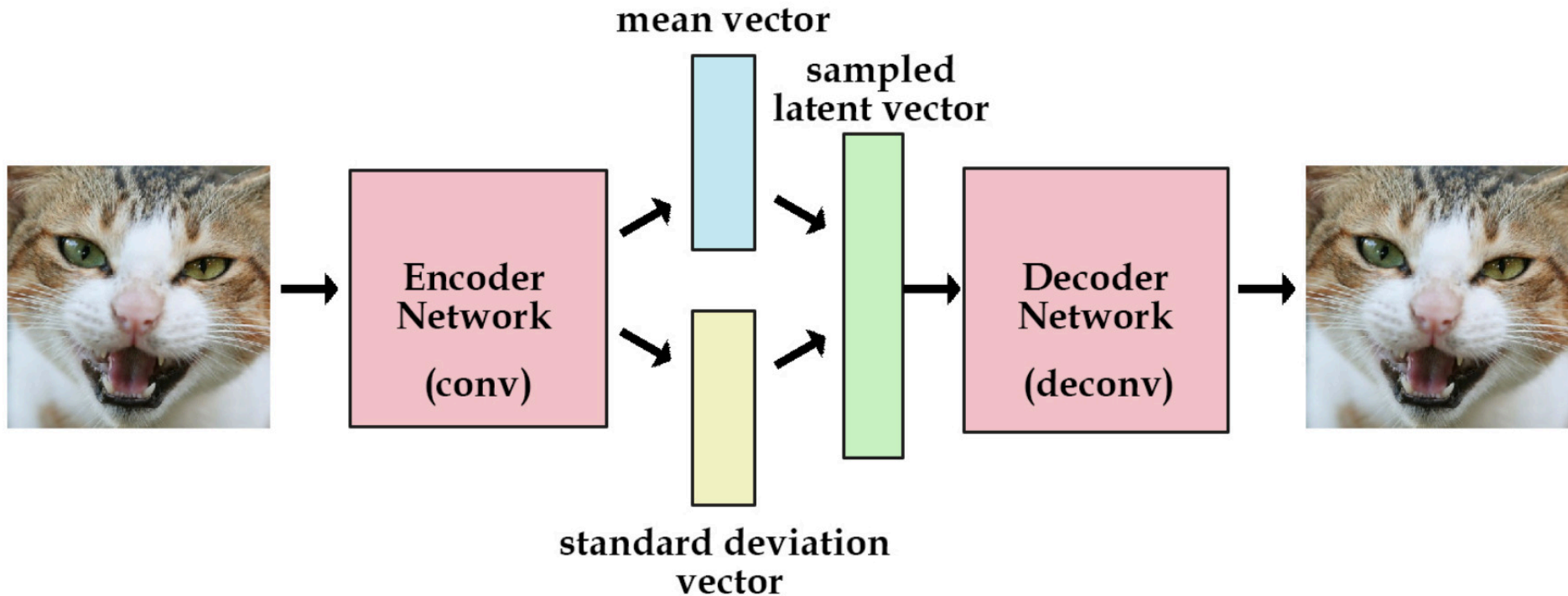
- Probabilistic extension of autoencoding
- The intuitive idea is to make z random, and in particular make P_z a Gaussian
 - If we can manage this, then we can sample from P_z and generate new images
- Two high level changes needed
 - Architecture
 - Loss function

Variational Autoencoder: Loss

- Loss will be sum of two losses
 - Reconstruction loss
 - Latent loss (how far from Gaussian the distribution obtained from encoder is)
 - Measured using KL divergence
 - Encoder generates the mean and covariance of the Gaussian
- We will look at the math behind this shortly

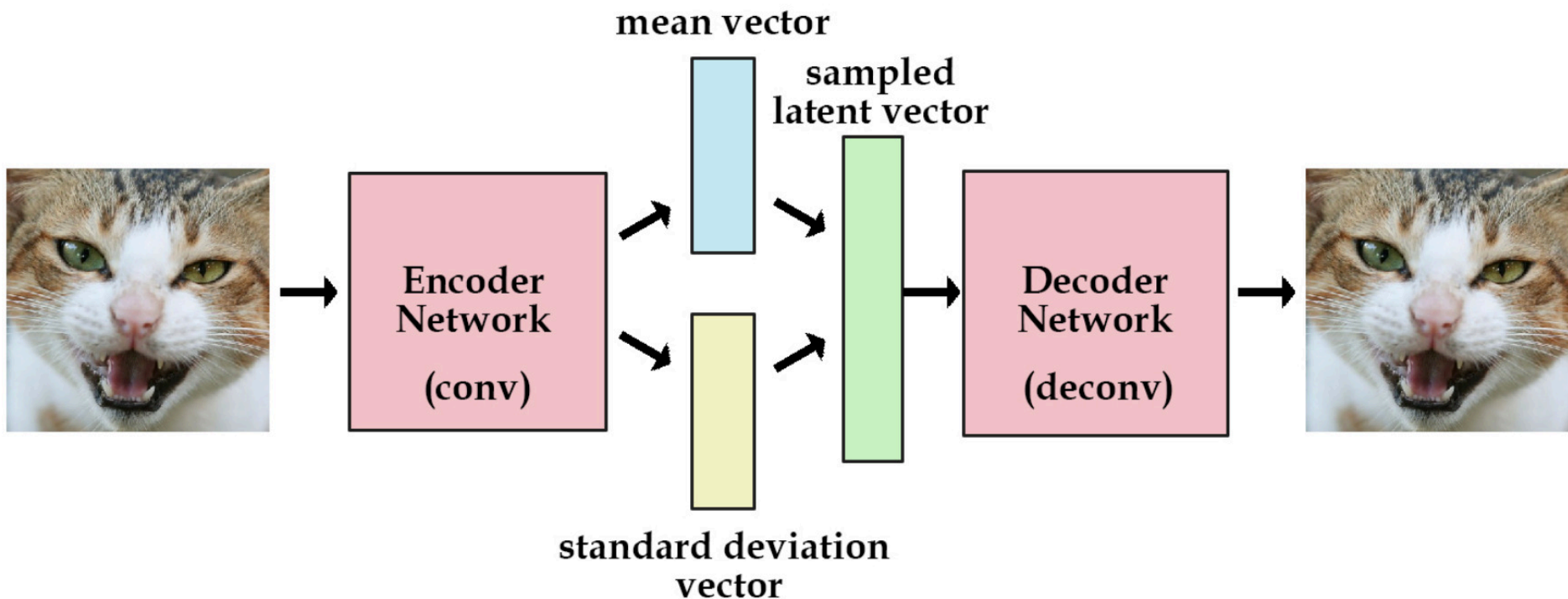
Variational Autoencoder: Architecture

- Architecture involves a sampling in between



Variational Autoencoder: Architecture

- Architecture involves a sampling in between
- Can still backprop given realized sample



Variational Autoencoder: Generalization

- This sampling allows for generalization
 - Gaussian noise ensures we are not remembering only the training data
 - More noise means less information is stored in the decoder
- Once we have trained, we can sample from a Gaussian and pass it through the decoder to get a new image

Variational Autoencoder: Samples

- Experiments on MNIST
 - Samples generated during training (left, center) and original data



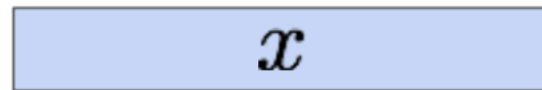
Source: <https://github.com/kvfrans/variational-autoencoder>

VAE: Derivation

- Assume a model as below
- Variable x represents image, z represents the latent variable
- We want to estimate θ^*

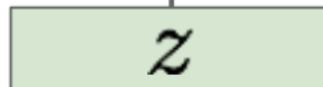
Sample from
true conditional

$$p_{\theta^*}(x \mid z^{(i)})$$



Sample from
true prior

$$p_{\theta^*}(z)$$

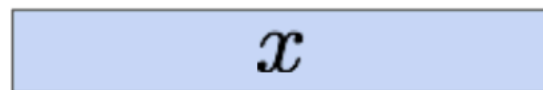


VAE: Derivation

- Let P_z be Gaussian
- Let $P(x|z)$ be a neural network: decoder
- We can train by maximizing likelihood of training data $p_{\theta}(x) = \int p_{\theta}(z)p_{\theta}(x|z)dz$

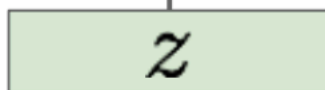
Sample from
true conditional

$$p_{\theta^*}(x | z^{(i)})$$



Sample from
true prior

$$p_{\theta^*}(z)$$

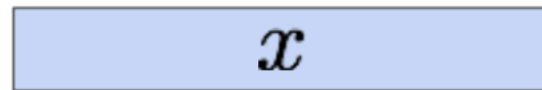


VAE: Derivation

- Let P_z be Gaussian
- Let $P(x|z)$ be a neural network: decoder
- We can train by maximizing likelihood of training data $p_{\theta}(x) = \int p_{\theta}(z)p_{\theta}(x|z)dz$

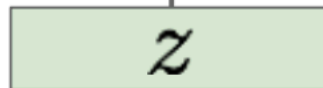
Sample from
true conditional

$$p_{\theta^*}(x | z^{(i)})$$



Sample from
true prior

$$p_{\theta^*}(z)$$



VAE: Derivation

- Unfortunately, maximizing $p_{\theta}(x) = \int p_{\theta}(z)p_{\theta}(x|z)dz$ is hard
 - Because of the decoder network

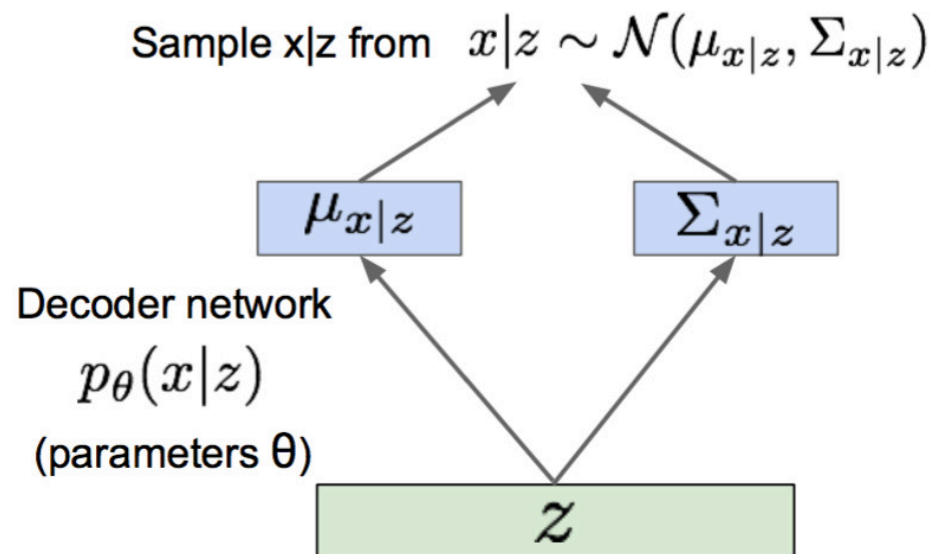
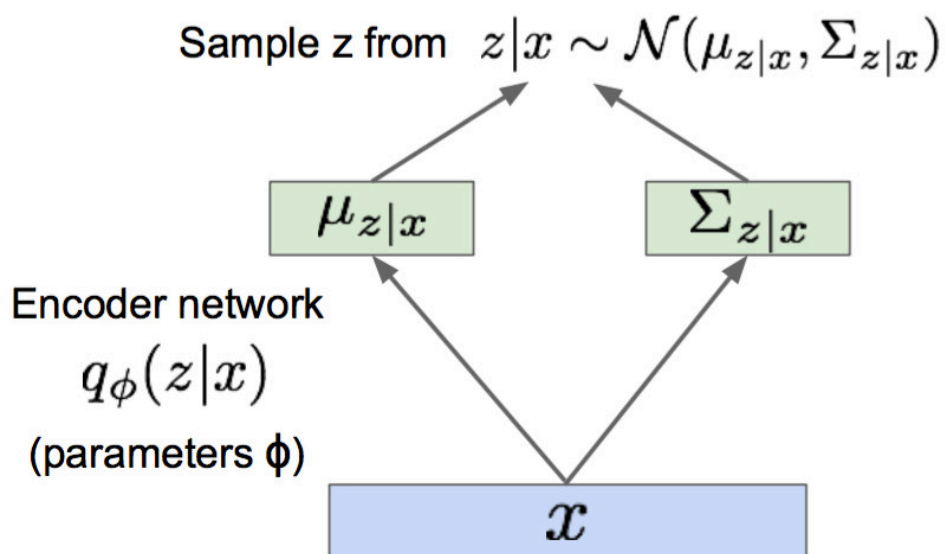
Data likelihood: $p_{\theta}(x) = \int p_{\theta}(z)p_{\theta}(x|z)dz$

Posterior density also intractable: $p_{\theta}(z|x) = p_{\theta}(x|z)p_{\theta}(z)/p_{\theta}(x)$

- We fix this by assuming an approximation ‘encoder’ $q_{\phi}(z|x)$ to $p_{\theta}(z|x)$

VAE: Derivation

- The encoder and decoder are probabilistic

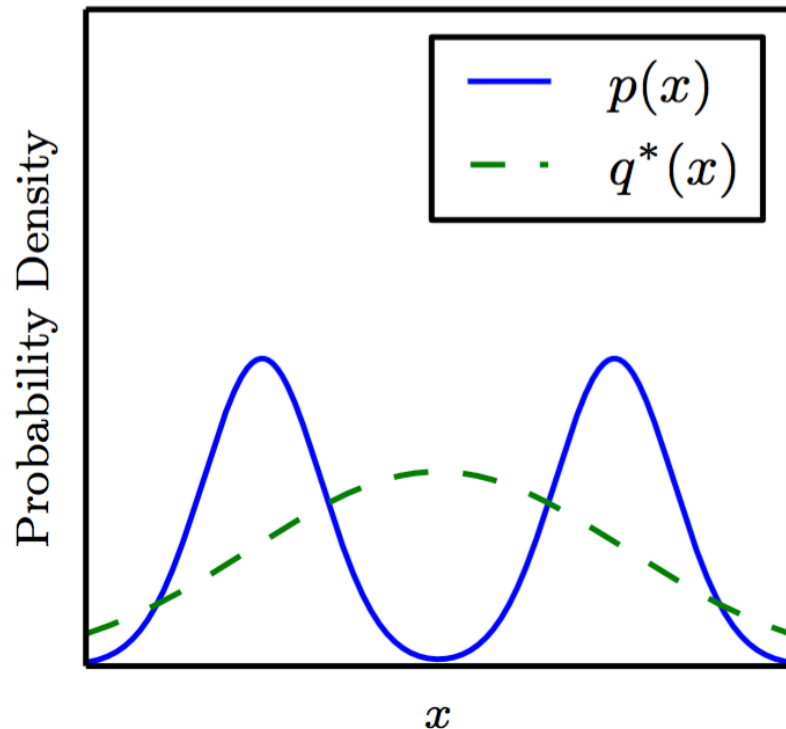


Aside: Notion of Information

- Information: $-\log P(x)$
- Entropy: $-\sum P(x) \log P(x)$
- KL divergence:
 - A notion of dissimilarity between two distributions
 - $D_{KL}(p||q) = \sum P(x) \log \frac{P(x)}{Q(x)}$

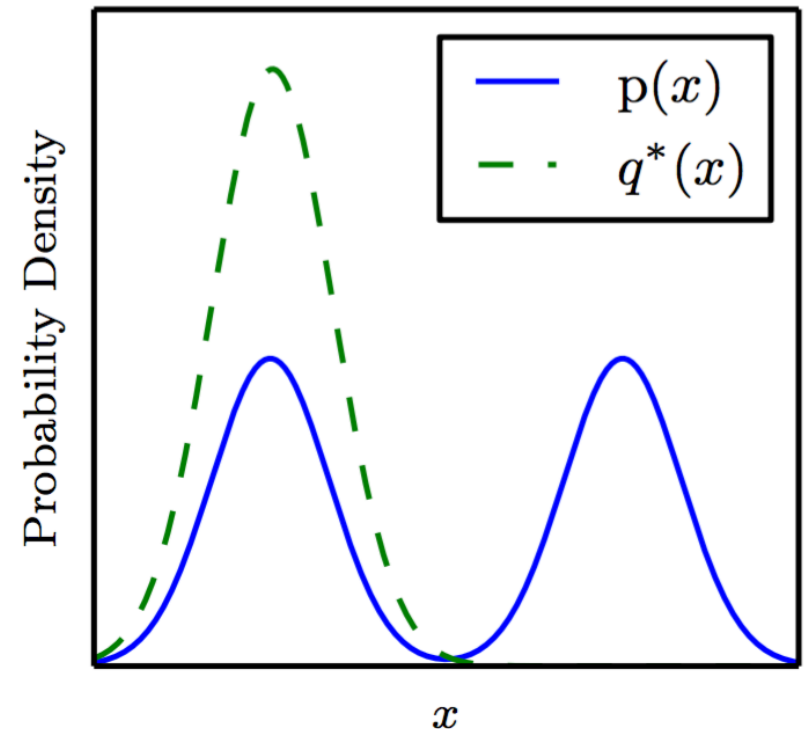
Aside: Notion of Information

$$q^* = \operatorname{argmin}_q D_{\text{KL}}(p||q)$$



Maximum likelihood

$$q^* = \operatorname{argmin}_q D_{\text{KL}}(q||p)$$



Reverse KL

VAE: Derivation

$$\log p_{\theta}(x^{(i)}) = \mathbf{E}_{z \sim q_{\phi}(z|x^{(i)})} \left[\log p_{\theta}(x^{(i)}) \right] \quad (p_{\theta}(x^{(i)})) \text{ Does not depend on } z$$

VAE: Derivation

$$\begin{aligned}\log p_{\theta}(x^{(i)}) &= \mathbf{E}_{z \sim q_{\phi}(z|x^{(i)})} \left[\log p_{\theta}(x^{(i)}) \right] && (p_{\theta}(x^{(i)})) \text{ Does not depend on } z \\ &= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z)p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \right] && (\text{Bayes' Rule})\end{aligned}$$

VAE: Derivation

$$\begin{aligned}\log p_{\theta}(x^{(i)}) &= \mathbf{E}_{z \sim q_{\phi}(z|x^{(i)})} \left[\log p_{\theta}(x^{(i)}) \right] && (p_{\theta}(x^{(i)}) \text{ Does not depend on } z) \\ &= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \right] && (\text{Bayes' Rule}) \\ &= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \frac{q_{\phi}(z | x^{(i)})}{q_{\phi}(z | x^{(i)})} \right] && (\text{Multiply by constant})\end{aligned}$$

VAE: Derivation

$$\begin{aligned}\log p_{\theta}(x^{(i)}) &= \mathbf{E}_{z \sim q_{\phi}(z|x^{(i)})} \left[\log p_{\theta}(x^{(i)}) \right] && (p_{\theta}(x^{(i)}) \text{ Does not depend on } z) \\&= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \right] && (\text{Bayes' Rule}) \\&= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \frac{q_{\phi}(z | x^{(i)})}{q_{\phi}(z | x^{(i)})} \right] && (\text{Multiply by constant}) \\&= \mathbf{E}_z \left[\log p_{\theta}(x^{(i)} | z) \right] - \mathbf{E}_z \left[\log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z)} \right] + \mathbf{E}_z \left[\log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z | x^{(i)})} \right] && (\text{Logarithms})\end{aligned}$$

VAE: Derivation

$$\begin{aligned}\log p_{\theta}(x^{(i)}) &= \mathbf{E}_{z \sim q_{\phi}(z|x^{(i)})} \left[\log p_{\theta}(x^{(i)}) \right] && (p_{\theta}(x^{(i)}) \text{ Does not depend on } z) \\&= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \right] && (\text{Bayes' Rule}) \\&= \mathbf{E}_z \left[\log \frac{p_{\theta}(x^{(i)} | z) p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \frac{q_{\phi}(z | x^{(i)})}{q_{\phi}(z | x^{(i)})} \right] && (\text{Multiply by constant}) \\&= \mathbf{E}_z \left[\log p_{\theta}(x^{(i)} | z) \right] - \mathbf{E}_z \left[\log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z)} \right] + \mathbf{E}_z \left[\log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z | x^{(i)})} \right] && (\text{Logarithms}) \\&= \mathbf{E}_z \left[\log p_{\theta}(x^{(i)} | z) \right] - D_{KL}(q_{\phi}(z | x^{(i)}) || p_{\theta}(z)) + D_{KL}(q_{\phi}(z | x^{(i)}) || p_{\theta}(z | x^{(i)}))\end{aligned}$$

VAE: Derivation

- The first two terms constitute a lower bound for the data likelihood that we can maximize tractably

$$= \underbrace{\mathbf{E}_z \left[\log p_\theta(x^{(i)} | z) \right] - D_{KL}(q_\phi(z | x^{(i)}) || p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)} + \underbrace{D_{KL}(q_\phi(z | x^{(i)}) || p_\theta(z | x^{(i)}))}_{> 0}$$

$$\log p_\theta(x^{(i)}) \geq \mathcal{L}(x^{(i)}, \theta, \phi)$$

Variational lower bound ("ELBO")

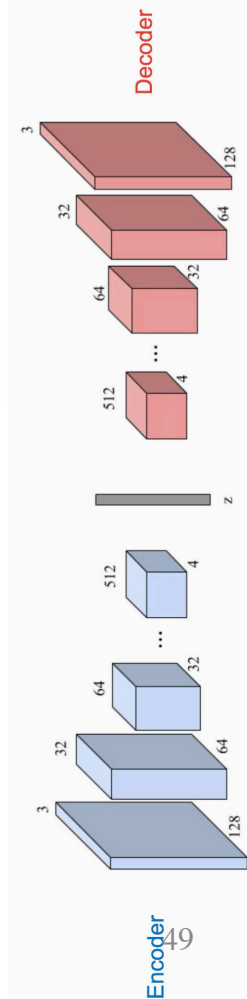
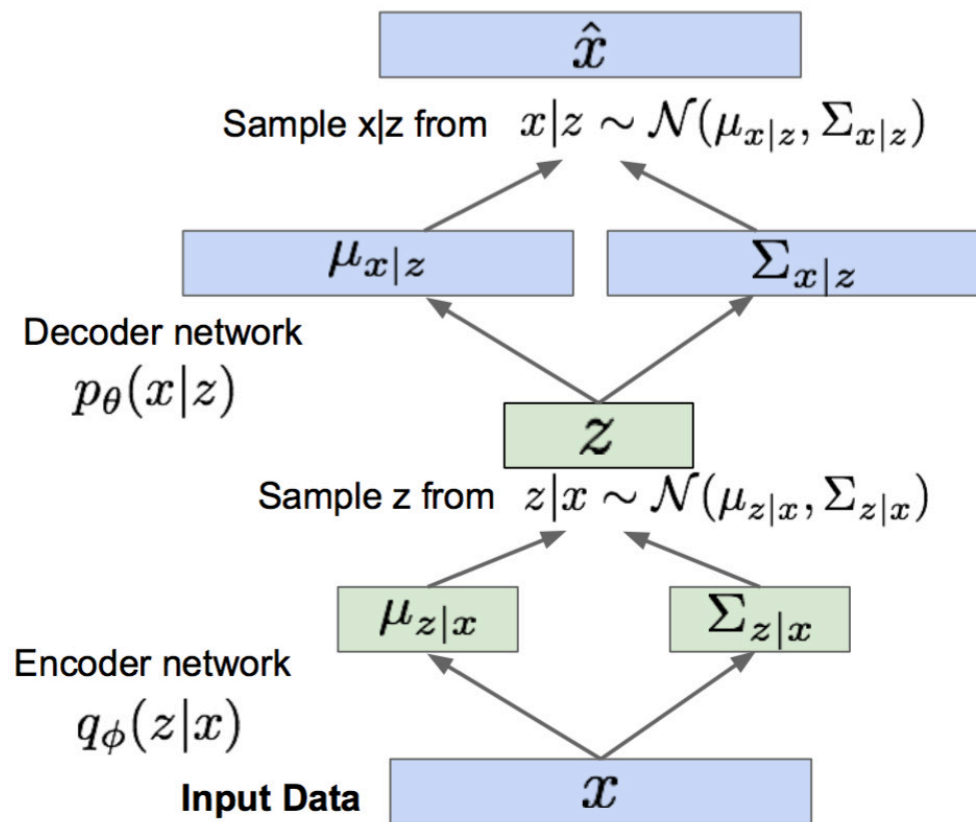
$$\theta^*, \phi^* = \arg \max_{\theta, \phi} \sum_{i=1}^N \mathcal{L}(x^{(i)}, \theta, \phi)$$

Training: Maximize lower bound

- The first term of \mathcal{L} is essentially reconstruction error
- The second term of \mathcal{L} is making the encoder network close to Gaussian prior

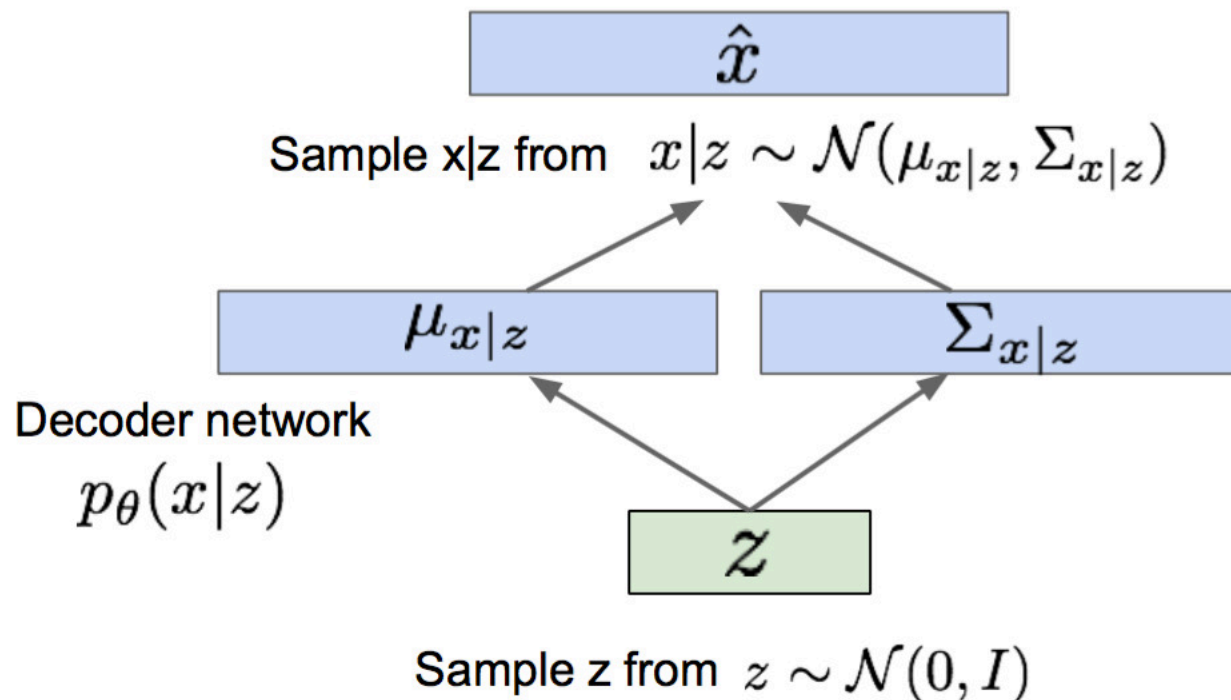
VAE: Derivation

- In summary,



VAE: Samples (Recap)

- We can create new samples!

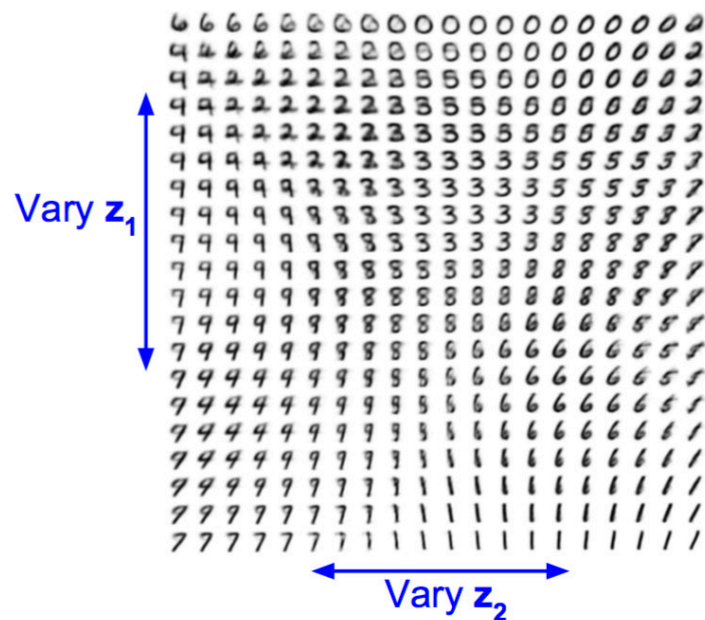


Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

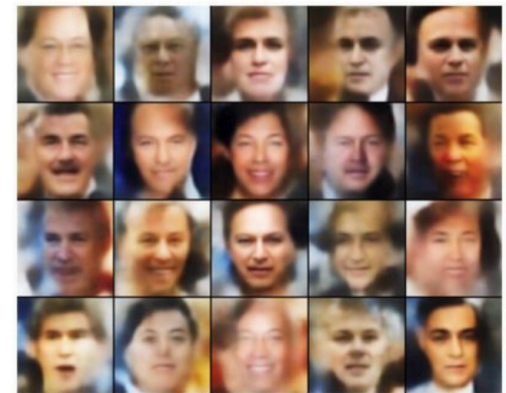
VAE: Experiments

- Some generated samples

Data manifold for 2-d \mathbf{z}



32x32 CIFAR-10



Labeled Faces in the Wild

Further reading: <https://arxiv.org/pdf/1606.05908.pdf>

Questions?

Today's Outline

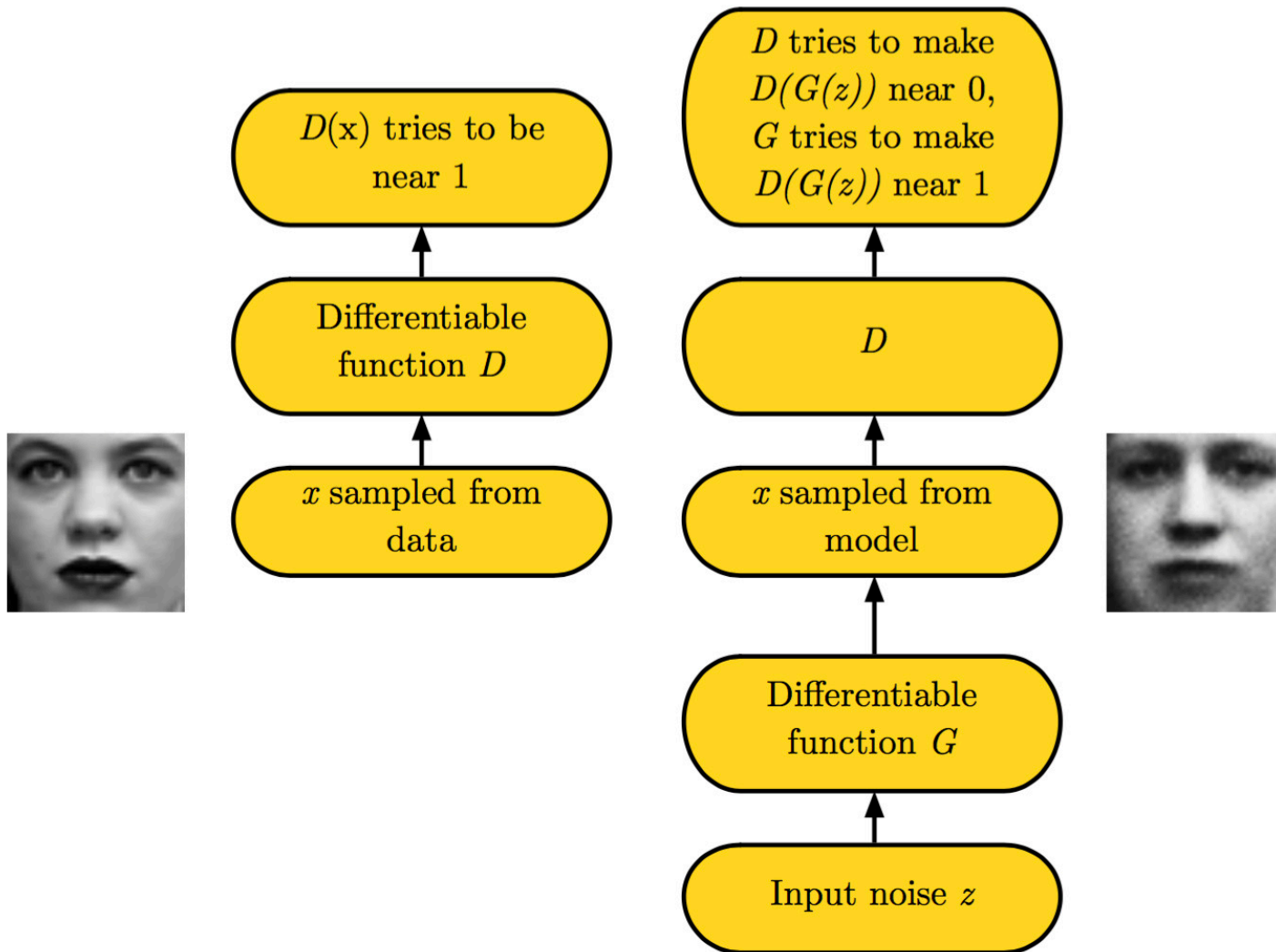
- Unsupervised Learning Landscape
- Autoencoders and Variational Autoencoders (VAE)
- Generative Adversarial Networks (GAN)

Generative Adversarial Networks

GANs: Two Scenarios

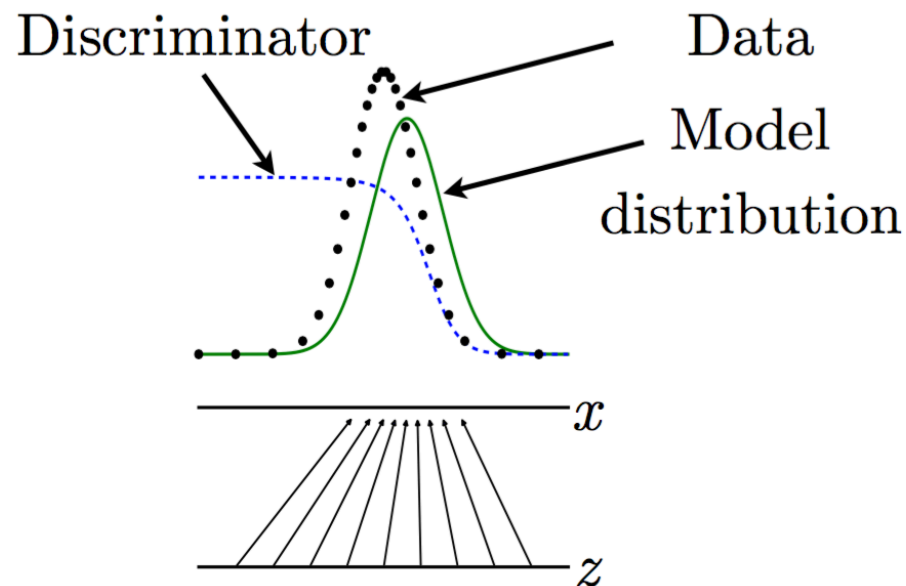
- Instead of working with an explicit density function, GANs take an ‘adversarial’ or ‘game-theoretic’ approach

GANs: Two Scenarios



The Generator and the Discriminator

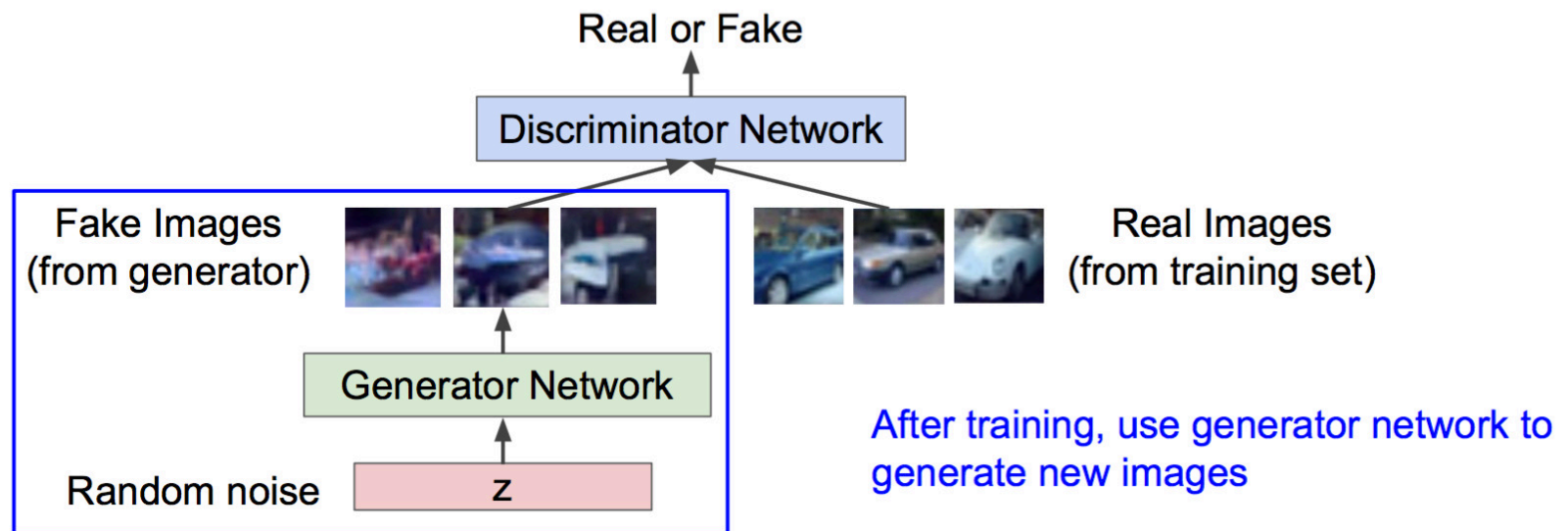
- Assume $X = G_{\theta_g}(z)$
- Differentiable
- $D_{\theta_d}(X)$ takes values in $\{0,1\}$



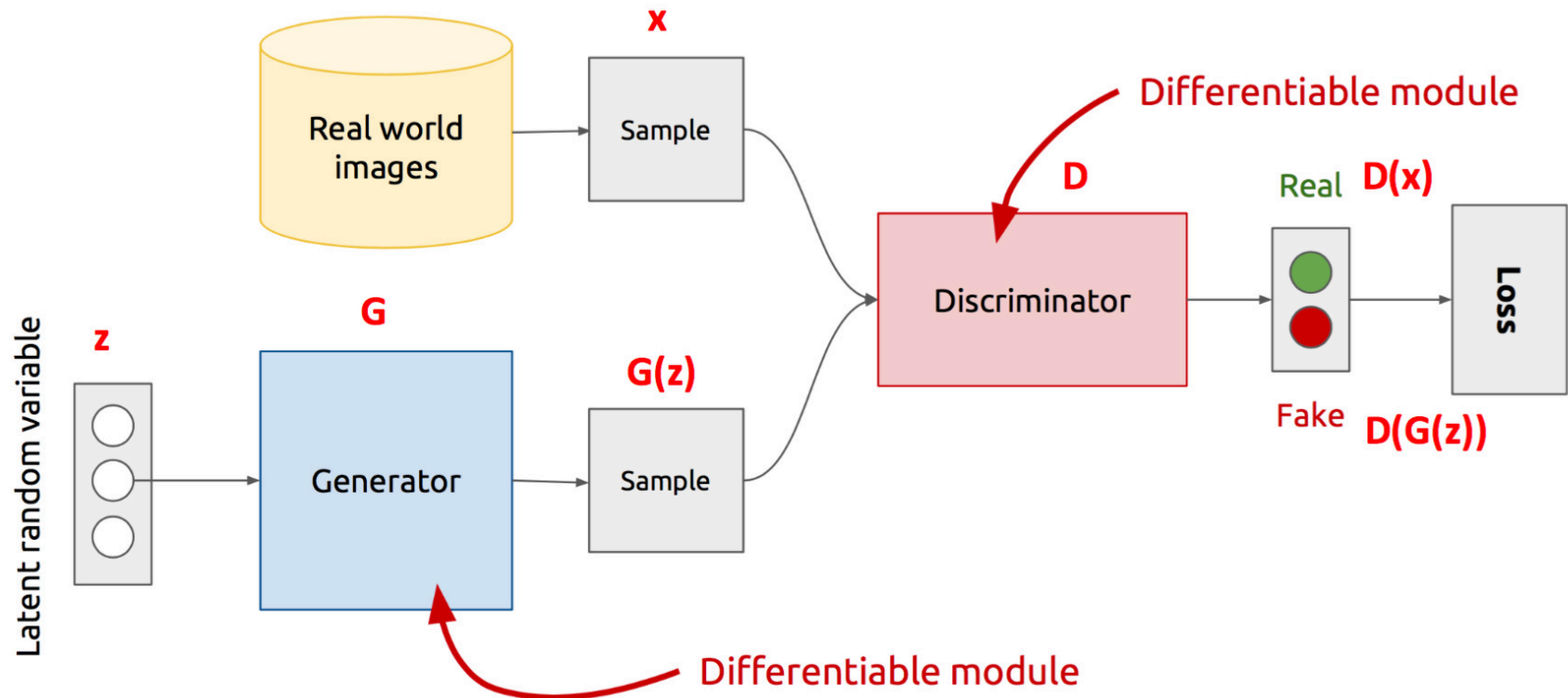
The Generator and the Discriminator

Generator network: try to fool the discriminator by generating real-looking images

Discriminator network: try to distinguish between real and fake images



The Generator and the Discriminator



The Objectives

- There are many variations. We will look at the original one.
- The generator and the discriminator are playing a minimax game.
- $J(D) = -E_{P_d} \log D(x) - E_{P_m} \log(1 - D(x))$
 - Where $P_m(x)$ is the derived distribution using $G(z)$ and P_z
- $J(G) = -J(D)$

The Objectives

- The optimal strategy for the discriminator at equilibrium is

- $$D(x) = \frac{P_d(x)}{P_d(x) + P_m(x)}$$

The Objectives

- The optimal strategy for the discriminator at equilibrium is
 - $D(x) = \frac{P_d(x)}{P_d(x) + P_m(x)}$
- The optimal strategy for the generator is to find parameters such that
 - $P_d = P_m$
- Caveat: Other variations of GANs are non minimax in nature and often times work better
 - Example: no saturation issue

The Training Procedure

- Create a minibatch of real data
- Create a minibatch of generated data
- Score the discriminator
- Backprop to update the parameter θ_d
- Score the generator
- Backprop to update the parameter θ_g

The Training Procedure

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Alternate between:

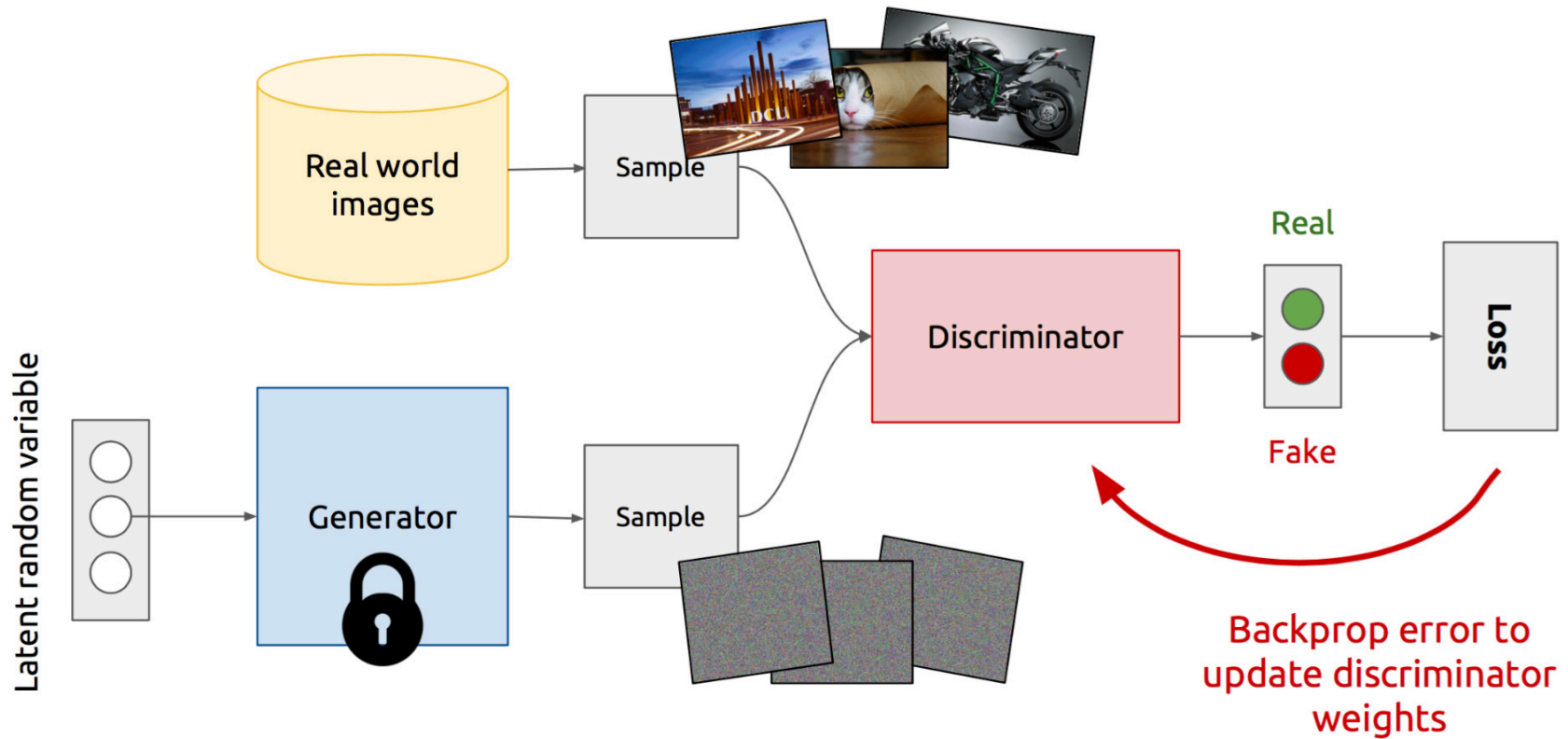
1. **Gradient ascent** on discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

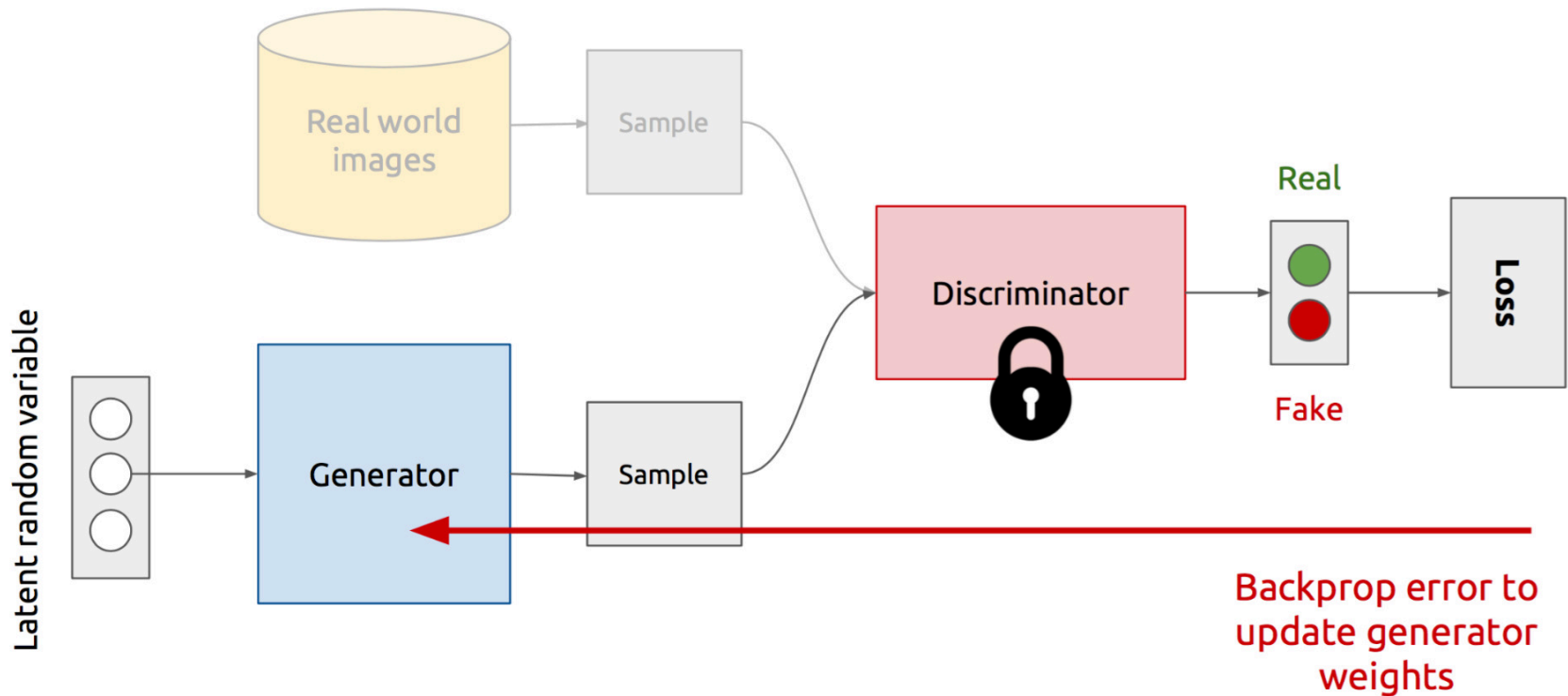
2. **Gradient descent** on generator

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

The Training Procedure

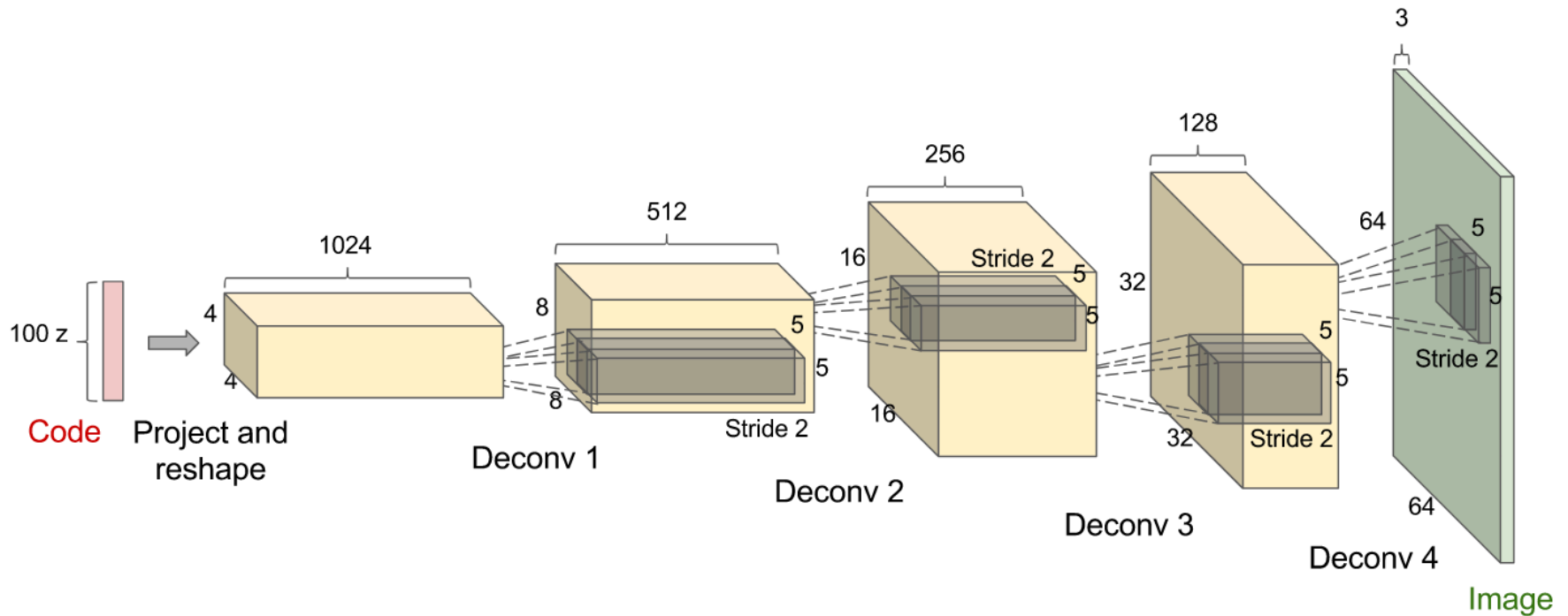


The Training Procedure

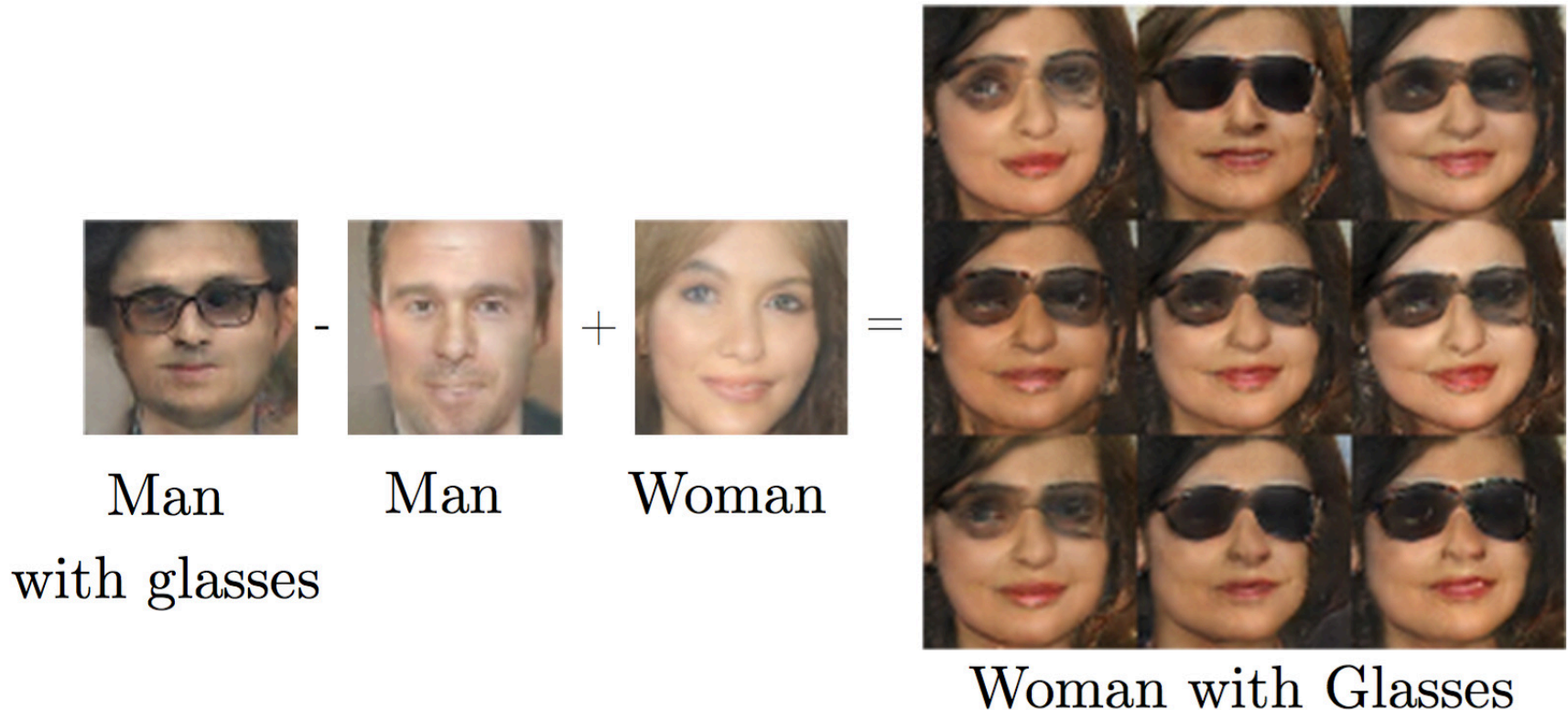


Example Generator Architecture

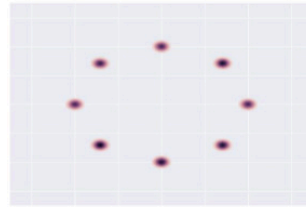
- DCGAN



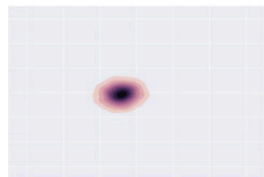
GAN Properties: Latent Space Arithmetic as a Byproduct



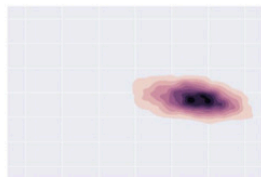
GAN Properties: Mode Collapse Issue



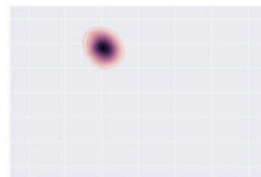
Target



Step 0



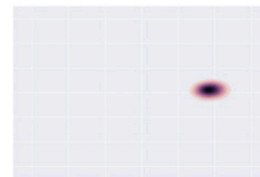
Step 5k



Step 10k



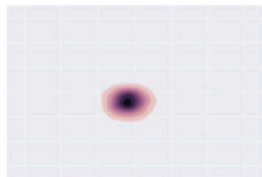
Step 15k



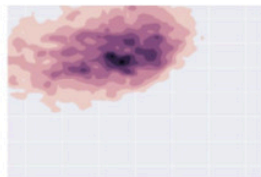
Step 20k



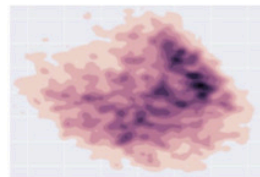
Step 25k



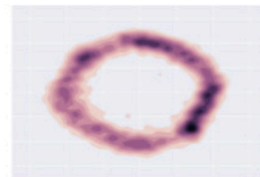
Step 0



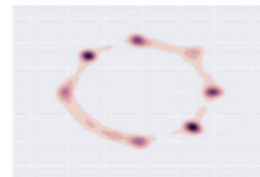
Step 5k



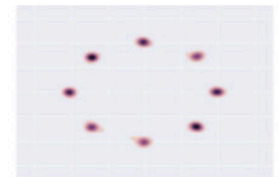
Step 10k



Step 15k



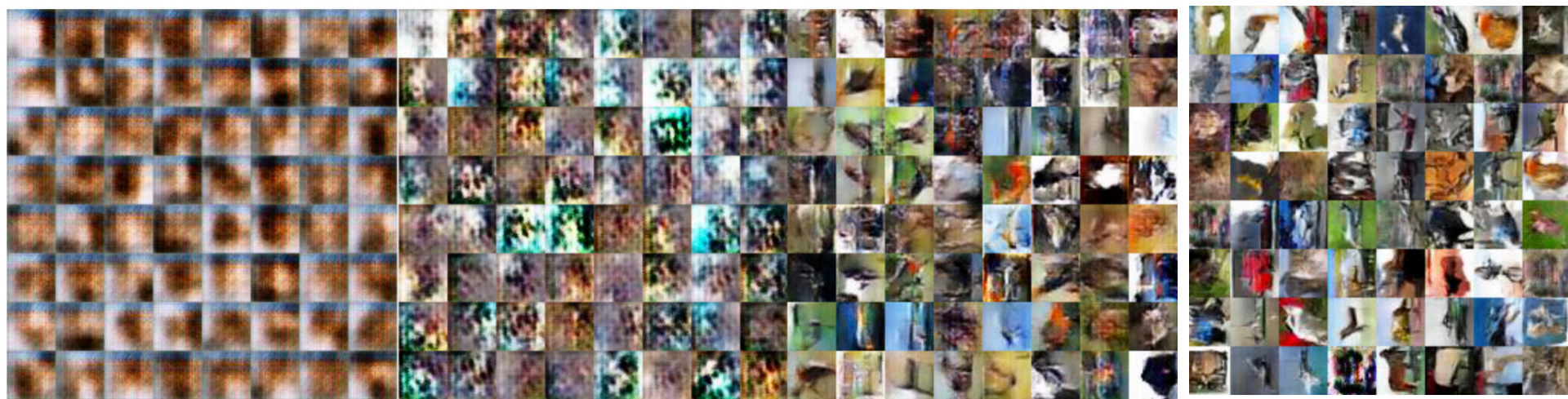
Step 20k



Step 25k

GAN: Experiments

- Experiments on CIFAR-10 (only generated images below)
 - Code: <https://github.com/kvfrans/generative-adversarial>



Questions?

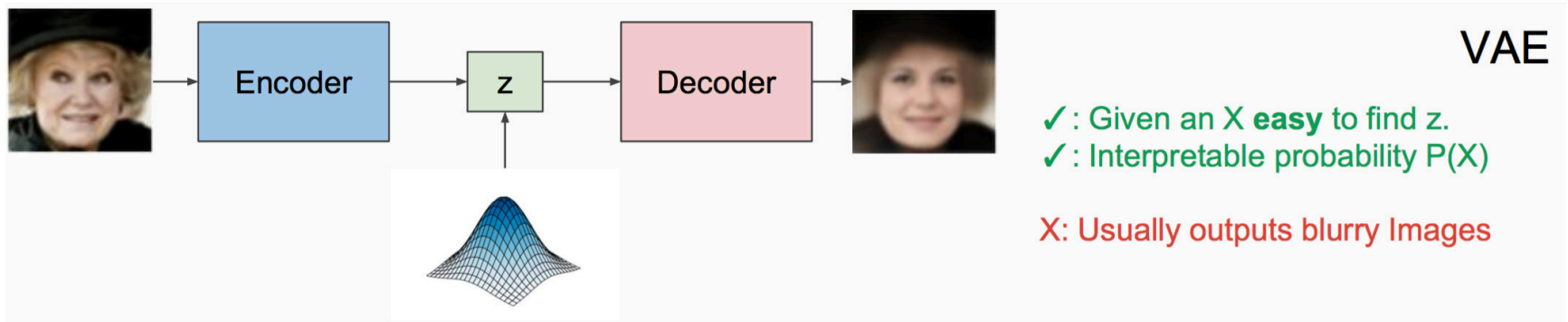
VAE and GAN

- VAEs
 - Are generative models that use regularized log likelihood to approximate performance score
 - Tend to achieve higher likelihoods of data, but the generated samples don't have real-world properties like sharpness
 - Can compare generated images with original images, which is not possible with GANs
 - Part of graphical models with principled theory

VAE and GAN

- GANs
 - Are generative models that use a supervised learning classifier to approximate performance score
 - No constraint that a ‘bed’ should look like a ‘bed’
 - Try to solve an intractable game, vastly more difficult to train
 - Tend to have sharper image samples
 - Start with latent variables and transform them deterministically
 - There is no Markov chain style of sampling required
 - They are asymptotically consistent (will converge to P_d), whereas VAEs are not
 - Many many variations have been proposed in the past 3 years (>150!)

VAE and GAN

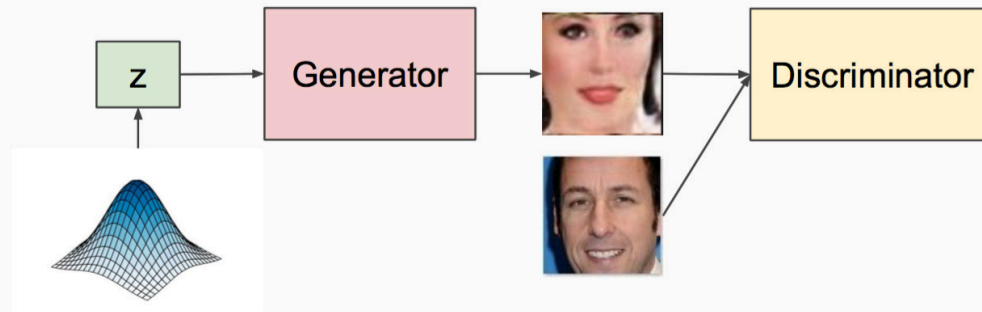


GAN

✓ : Very sharp images

X: Given an X **difficult** to find z . (Need to backprop.)

✓/X: No explicit $P(X)$.



Summary

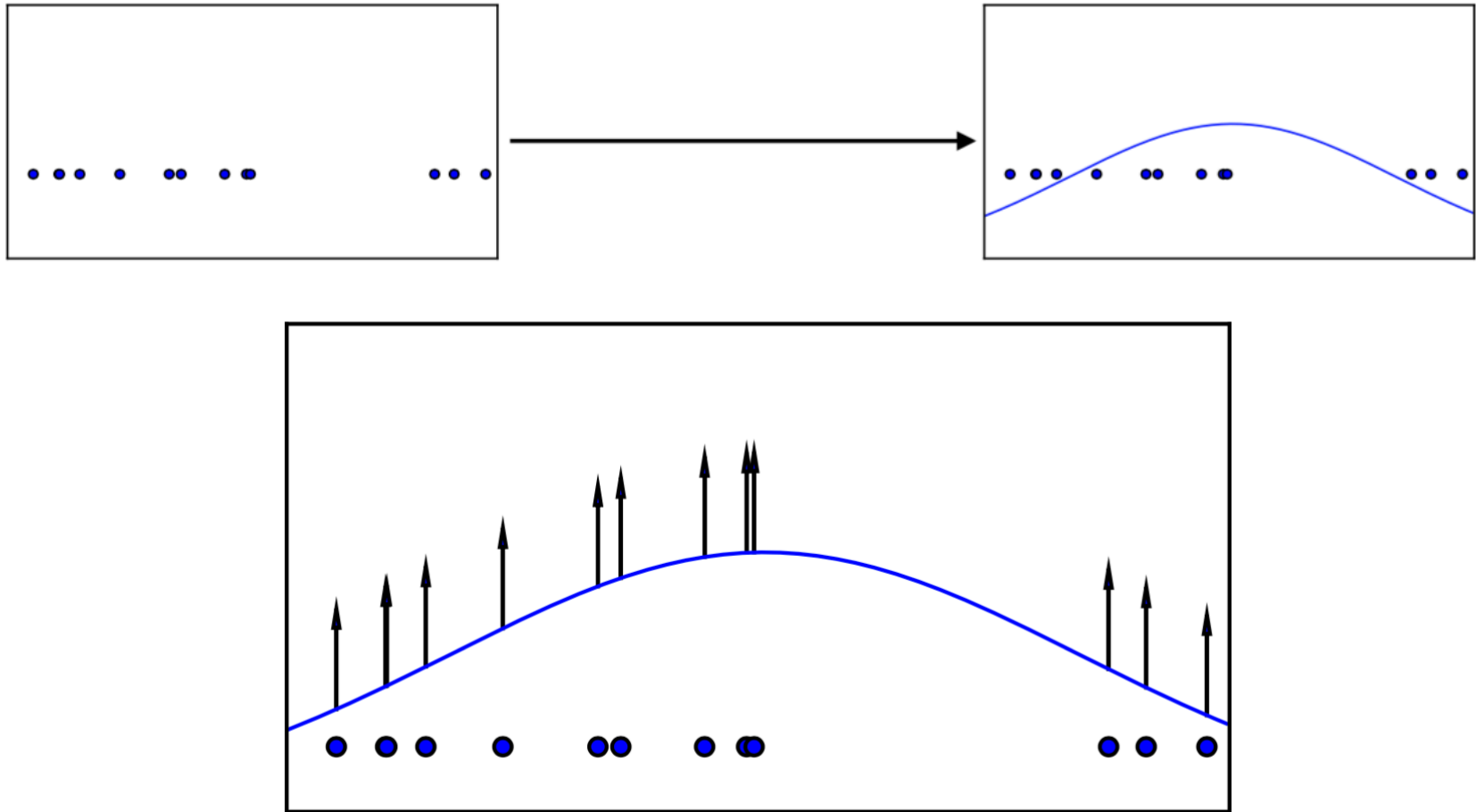
- Both models are recent (VAEs from 2013, GANs from 2014) and have initiated very exciting new directions in machine learning and AI
- Useful in many applications such as
 - Image denoising
 - Image Super-resolution
 - Reinforcement learning
 - Generating embeddings
 - Artistic help
- Eventually help the computer understand the world better

Appendix

Sample Exam Questions

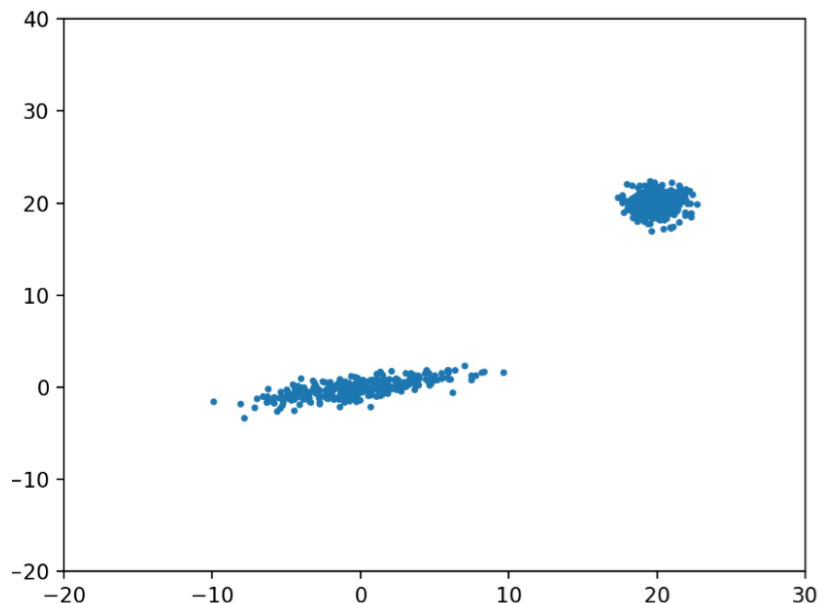
- What are the uses of generative models?
- What is the difference between a regular autoencoder and a variational autoencoder?
- What is the qualitative objective of the discriminator in a GAN? What is the qualitative objective of the generator?
- Describe some differences between a VAE model and a GAN.

Maximum Likelihood Estimation I



Maximum Likelihood Estimation II

Step 1: observe a set of samples



Step 2: assume a GMM model

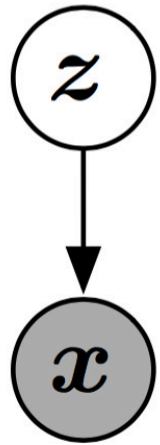
$$p(x|\theta) = \sum_i \pi_i \mathcal{N}(x|\mu_i, \Sigma_i)$$

Step 3: perform maximum likelihood learning

$$\max_{\theta} \sum_{x^{(j)} \in \text{Dataset}} \log p(\theta|x^{(j)})$$

VAE: Original Work

(Kingma and Welling 2013, Rezende et al 2014)



$$\begin{aligned}\log p(\mathbf{x}) &\geq \log p(\mathbf{x}) - D_{\text{KL}}(q(\mathbf{z}) \| p(\mathbf{z} | \mathbf{x})) \\ &= \mathbb{E}_{\mathbf{z} \sim q} \log p(\mathbf{x}, \mathbf{z}) + H(q)\end{aligned}$$



CIFAR-10 samples

Recent Work: Plug and Play Generative Models

- A recent work combines GANs with denoising autoencoders and other techniques to generate realistic samples
- There are many other such attempts in the past 2 years



redshank

ant

monastery



volcano



oranges on a table next to a liquor bottle

¹Reference: Ian Goodfellow (NIPS 2016 Tutorial)

Advanced Prediction Models

Deep Learning, Graphical Models and Reinforcement
Learning

Today's Outline

- Quick Review
- CNN Visualization for Fun
- Deeper Nets: Recent Advances
- Practical Tips

Review of Deep Learning

Key Topics

- Basics of machine learning
 - Classification pipeline
 - Regularization
- Neural networks
 - Backpropagation
 - CNNs: convolution, pooling
 - RNNs (including LSTMs)
 - VAEs and GANs
- Data domains
 - Vision (images)
 - Text
 - Embeddings: word2vec
 - Language models

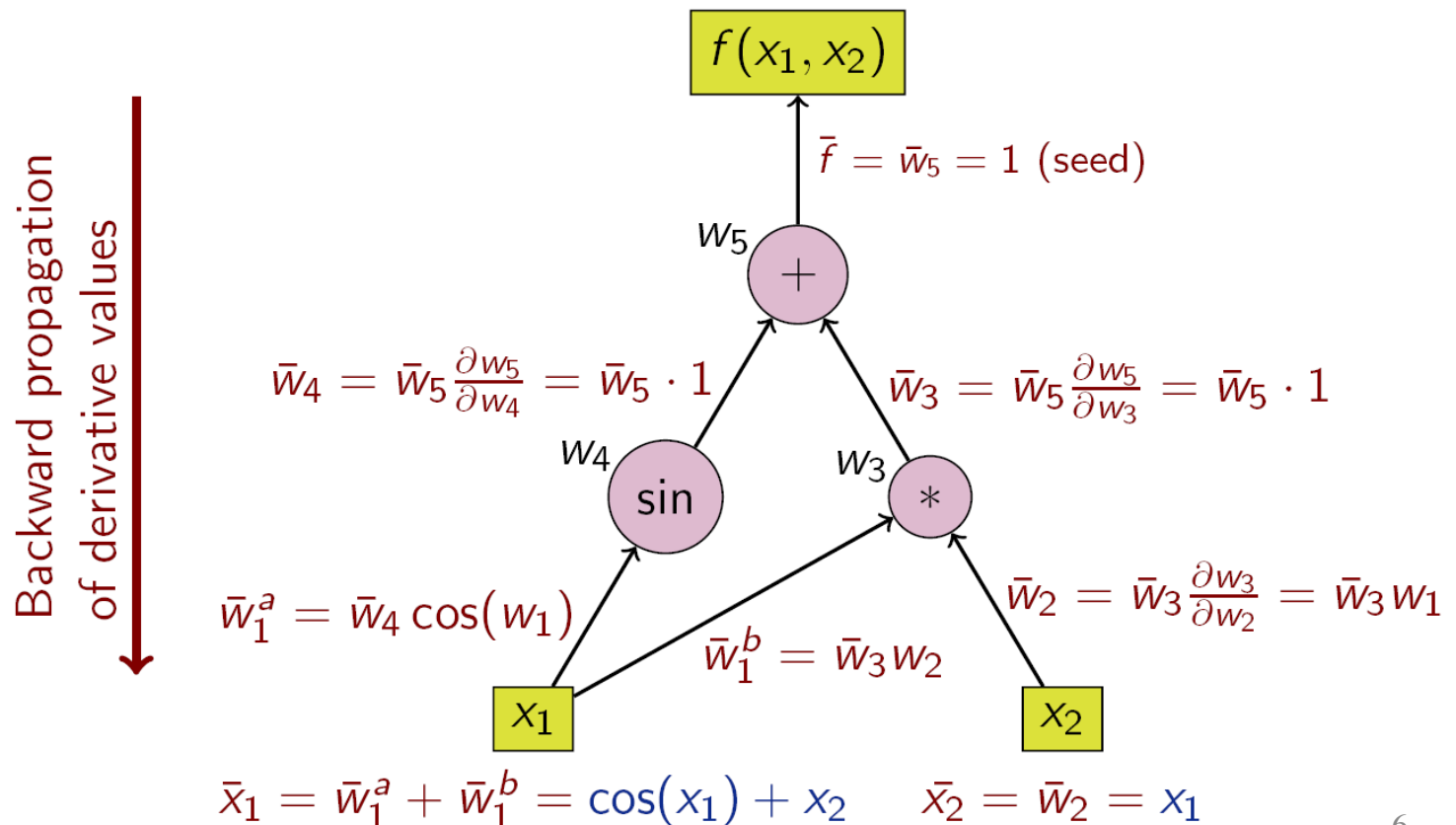
Sample Concepts

- Backpropagation
- Convolution, Pooling and Batch Normalization in CNNs
- LSTMs
- VAEs and GANs
- Word2Vec

Backpropagation Example I

- Backpropagation is a case of reverse accumulation automatic differentiation¹

An example from wikipedia



¹See https://en.wikipedia.org/wiki/Automatic_differentiation

Backpropagation Example II

- Compute the gradient using backpropagation

Consider a three layer network:

$$h^1 = \sigma(W^1 x), h^2 = \sigma(W^2 h^1), f(x) = \langle w^3, h^2 \rangle.$$

See Figure 1 for an illustration. Compute $\frac{\partial f}{\partial W^1_{i,j}}$.

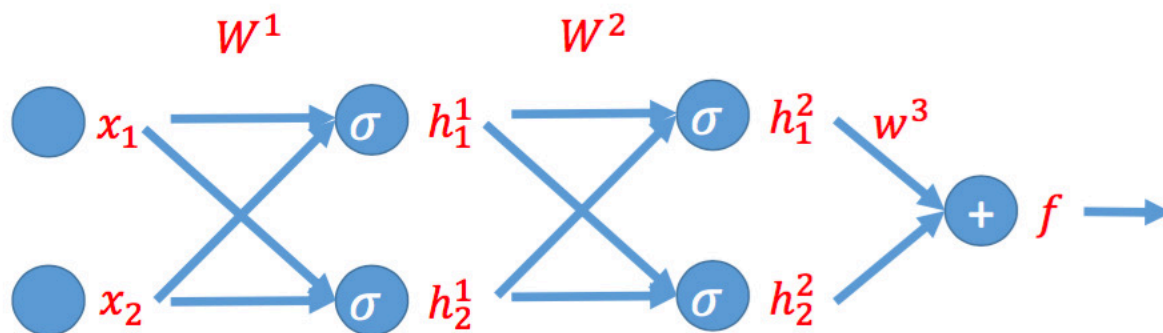


Figure 1: An illustration of the three layer network

Sample Concepts

- Backpropagation
- Convolution, Pooling and Batch Normalization in CNNs
- LSTMs
- VAEs and GANs
- Word2Vec

CNN: Computing Convolutions

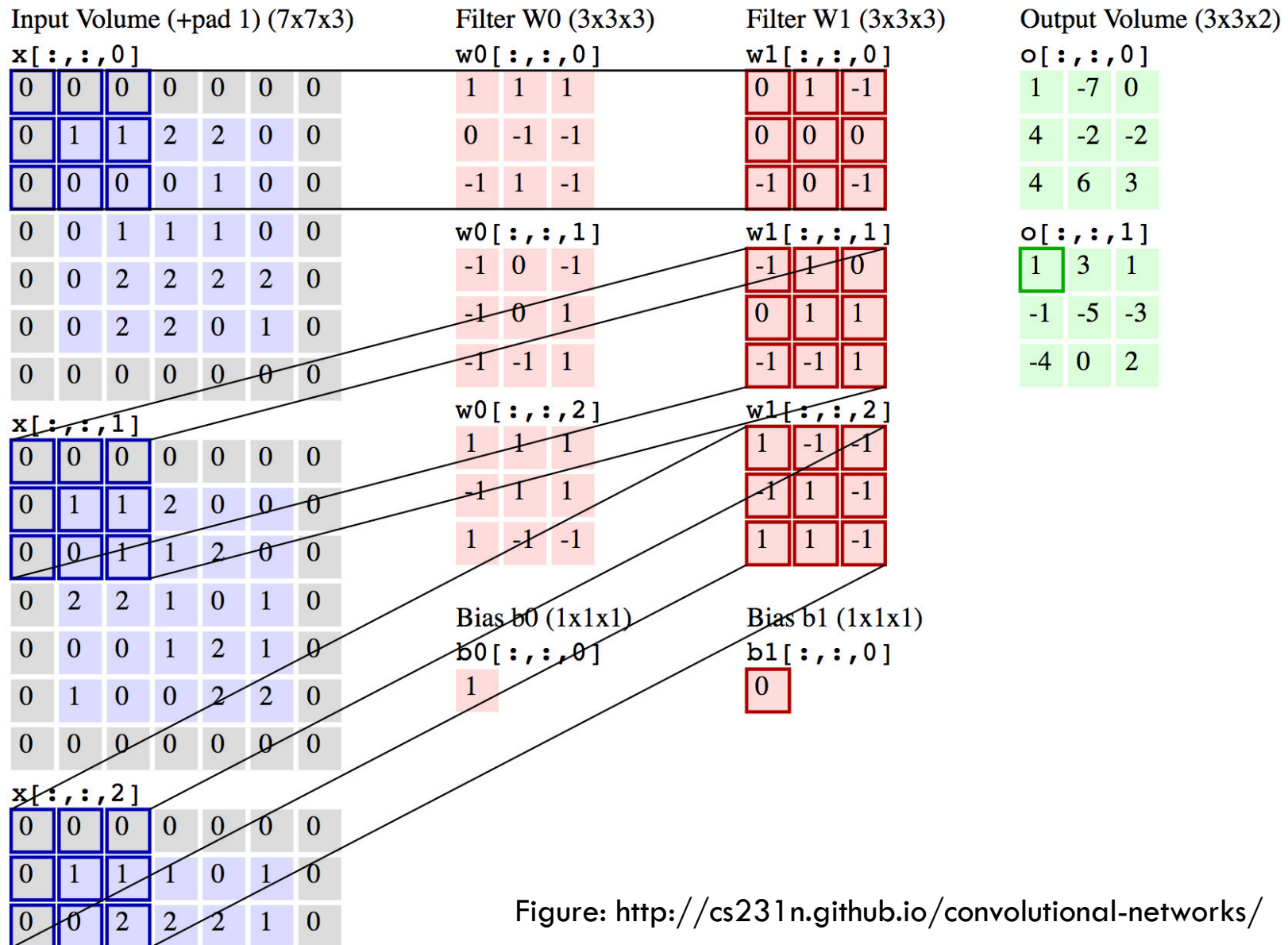
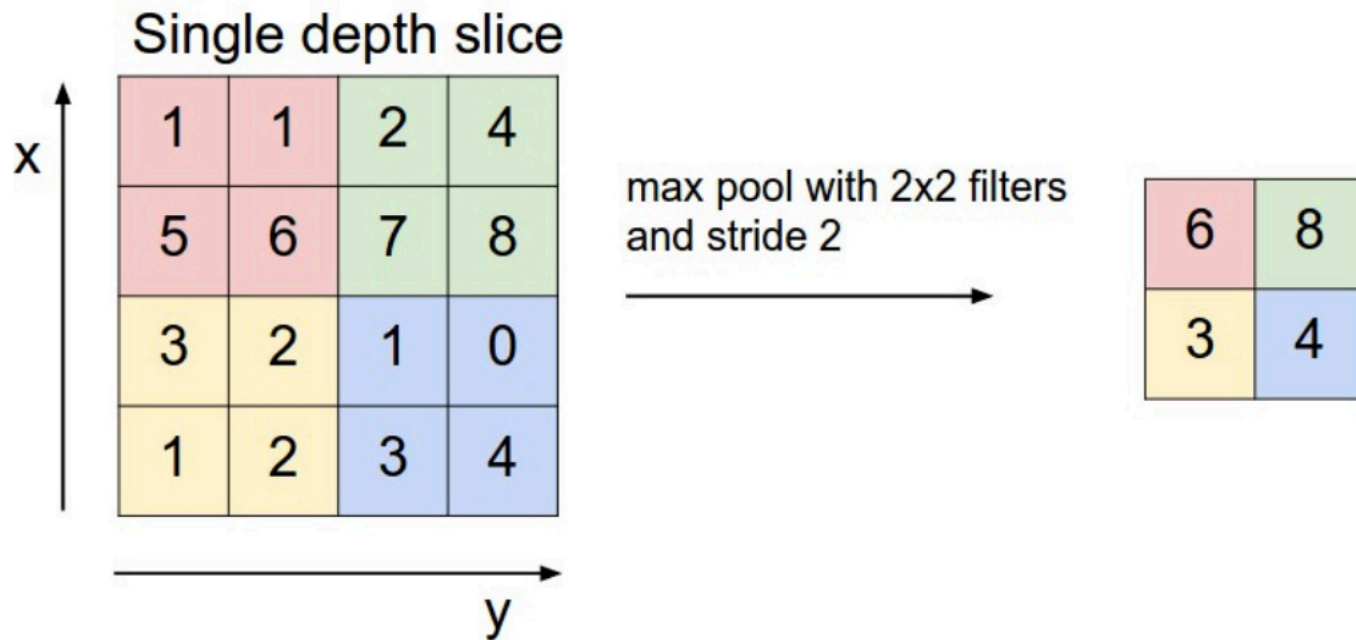


Figure: <http://cs231n.github.io/convolutional-networks/>

CNN: Computing Pooled Outputs

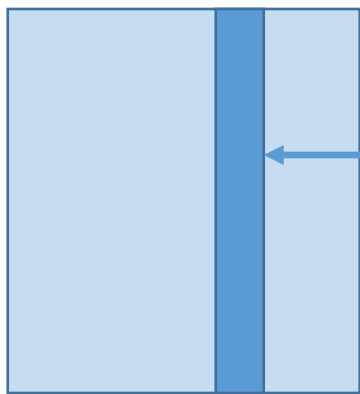


CNN: Batch Normalization

- Idea: Make each activation unit-Gaussian by subtracting the mean and then dividing by standard deviation

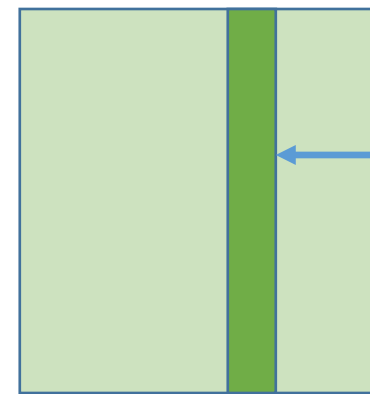
Batch-size = N

Number of output neurons = D



$N \times D$

$$\hat{x} = \frac{\gamma(x - E[x])}{\sqrt{Var[x]}} + \beta$$



$N \times D$

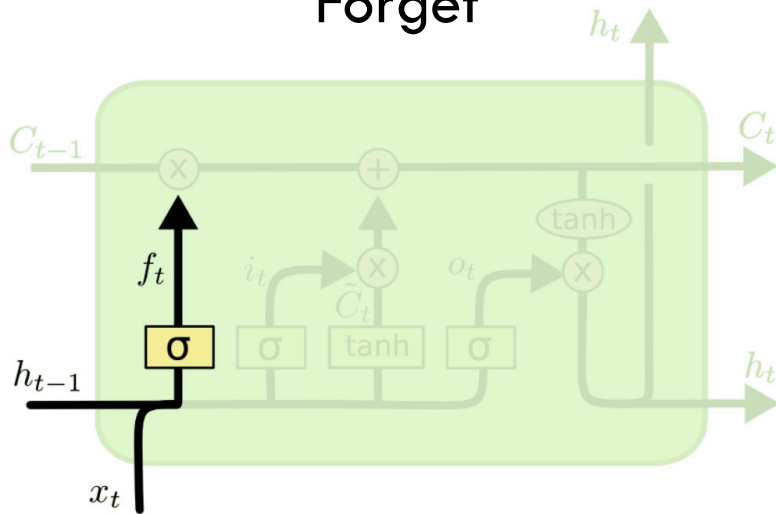
- Is a differentiable function: hence no issue with backpropagation
- At test time, there is no batch. Use the training data means and variances

Sample Concepts

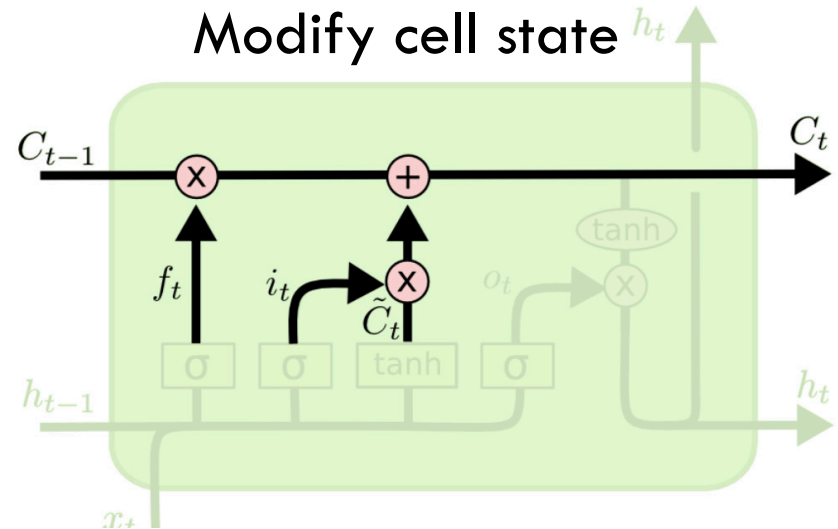
- Backpropagation
- Convolution, Pooling and Batch Normalization in CNNs
- LSTMs
- VAEs and GANs
- Word2Vec

LSTMs

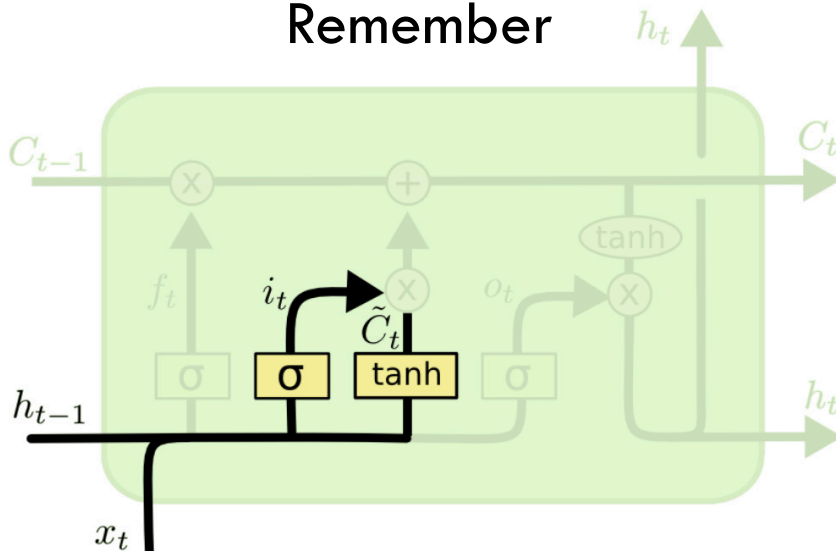
Forget



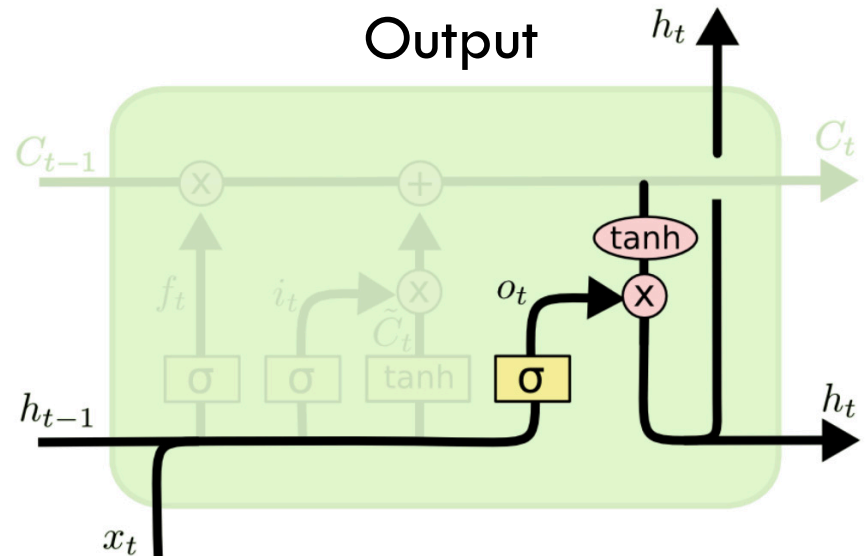
Modify cell state



Remember



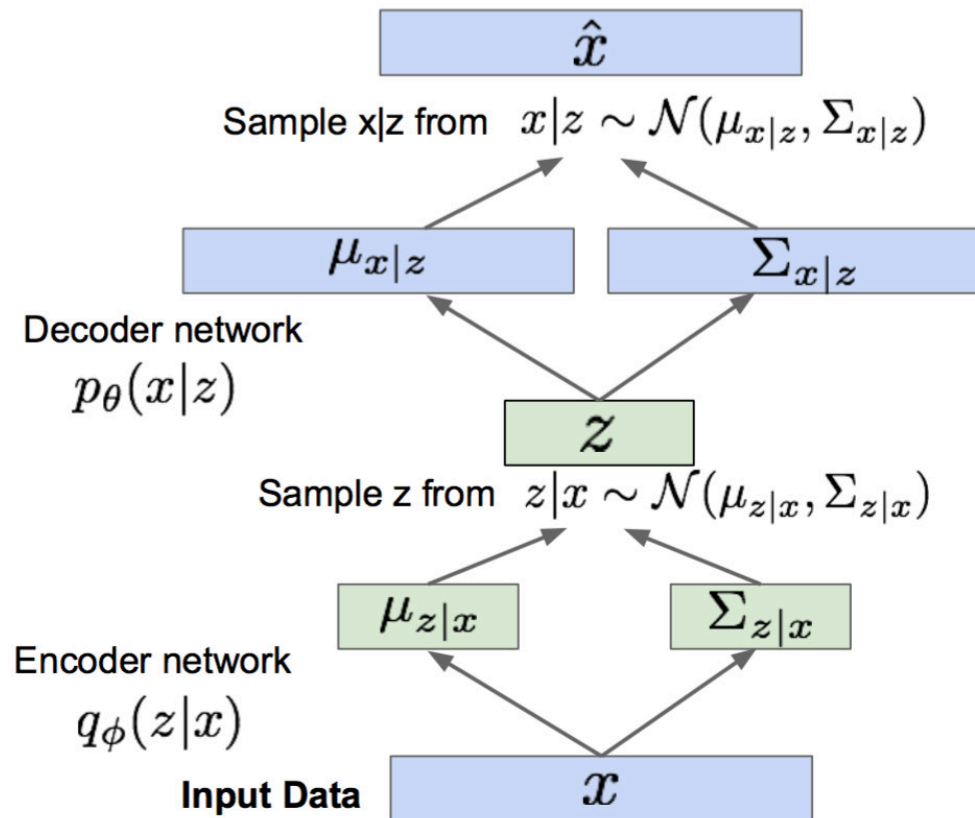
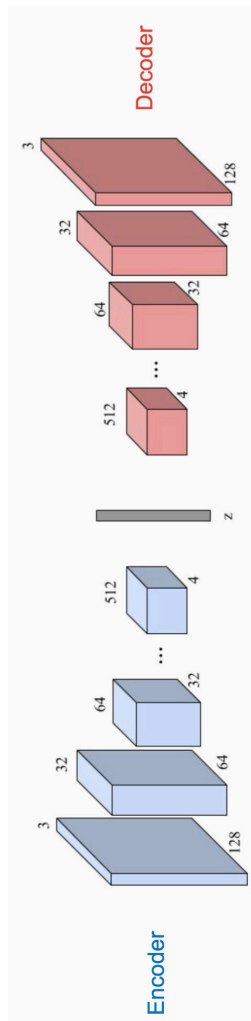
Output



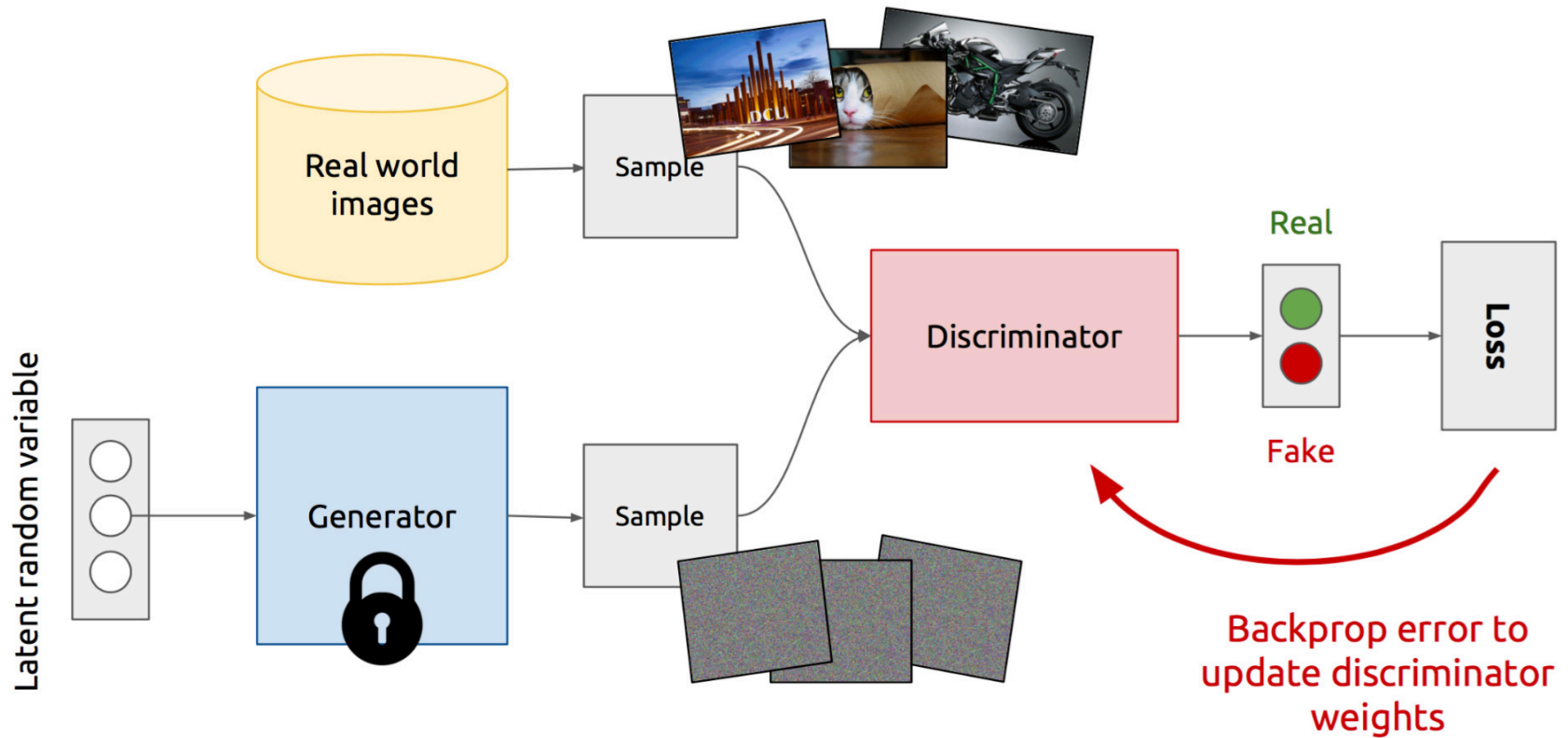
Sample Concepts

- Backpropagation
- Convolution, Pooling and Batch Normalization in CNNs
- LSTMs
- VAEs and GANs
- Word2Vec

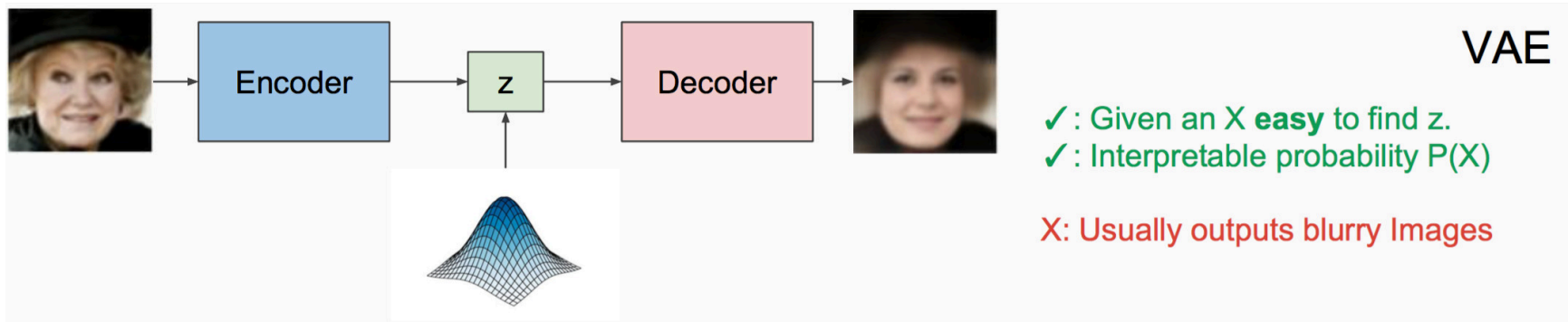
VAE: The Training Procedure



GAN: The Training Procedure



VAE and GAN

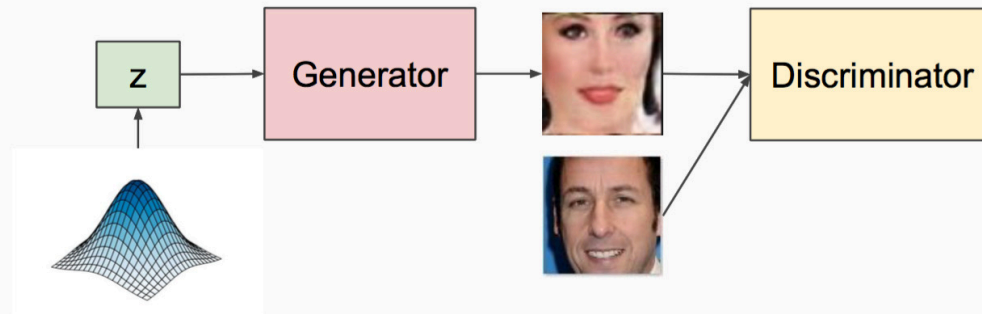


GAN

✓ : Very sharp images

X: Given an X **difficult** to find z . (Need to backprop.)

✓/X: No explicit $P(X)$.



VAE and GAN

- VAEs
 - Are generative models that use regularized log likelihood to approximate performance score
 - Tend to achieve higher likelihoods of data, but the generated samples don't have real-world properties like sharpness
 - Can compare generated images with original images, which is not possible with GANs
 - Part of graphical models with principled theory

VAE and GAN

- GANs
 - Are generative models that use a supervised learning classifier to approximate performance score
 - No constraint that a ‘bed’ should look like a ‘bed’
 - Try to solve an intractable game, vastly more difficult to train
 - Tend to have sharper image samples
 - Start with latent variables and transform them deterministically
 - There is no Markov chain style of sampling required
 - They are asymptotically consistent (will converge to P_d), whereas VAEs are not
 - Many many variations have been proposed in the past 3 years (>150!)

Sample Concepts

- Backpropagation
- Convolution, Pooling and Batch Normalization in CNNs
- LSTMs
- GANs
- Word2Vec

Word2Vector Word Embeddings

- An embedding of words proposed by Google in 2013
- Is based on a predictive method rather than on a count based method
- Objective: Vector representations of words that capture their co-occurrence statistics

Word2Vector: Two Versions

- Continuous Bag of Words and Skip-Gram
- Lets go through the skip-gram model in some detail now

W2V: The Skip-Gram Version

- This is a very simple neural network model to learn W
- We will train a single hidden layer NN to perform an auxiliary task
- The goal will be to just learn the weights of the network
 - This will give us W

W2V: The Auxiliary Task

- Task:
 - Pick a word in the middle of a sentence
 - Pick one of the **nearby** words at random
 - Make network learn probability of every word in our vocab of being this nearby word
- Input: a word (one hot encoded)
- Output: normalized scores (of length: vocab size)
- Meaning of 'nearby':
 - Essentially defined using a window size
 - Example: Window size 2 means 2 words to left and 2 to right of the input word are nearby

W2V: The Auxiliary Task

- Word pairs as feature-label pairs
- Example: “The quick brown fox jumps over the lazy dog”

Source Text	Training Samples					
<table><tr><td>The</td><td>quick</td><td>brown</td></tr></table> fox jumps over the lazy dog. ➡	The	quick	brown	(the, quick) (the, brown)		
The	quick	brown				
The <table><tr><td>quick</td><td>brown</td><td>fox</td></tr></table> jumps over the lazy dog. ➡	quick	brown	fox	(quick, the) (quick, brown) (quick, fox)		
quick	brown	fox				
The quick <table><tr><td>brown</td><td>fox</td><td>jumps</td></tr></table> over the lazy dog. ➡	brown	fox	jumps	(brown, the) (brown, quick) (brown, fox) (brown, jumps)		
brown	fox	jumps				
The <table><tr><td>quick</td><td>brown</td><td>fox</td><td>jumps</td><td>over</td></tr></table> the lazy dog. ➡	quick	brown	fox	jumps	over	(fox, quick) (fox, brown) (fox, jumps) (fox, over)
quick	brown	fox	jumps	over		

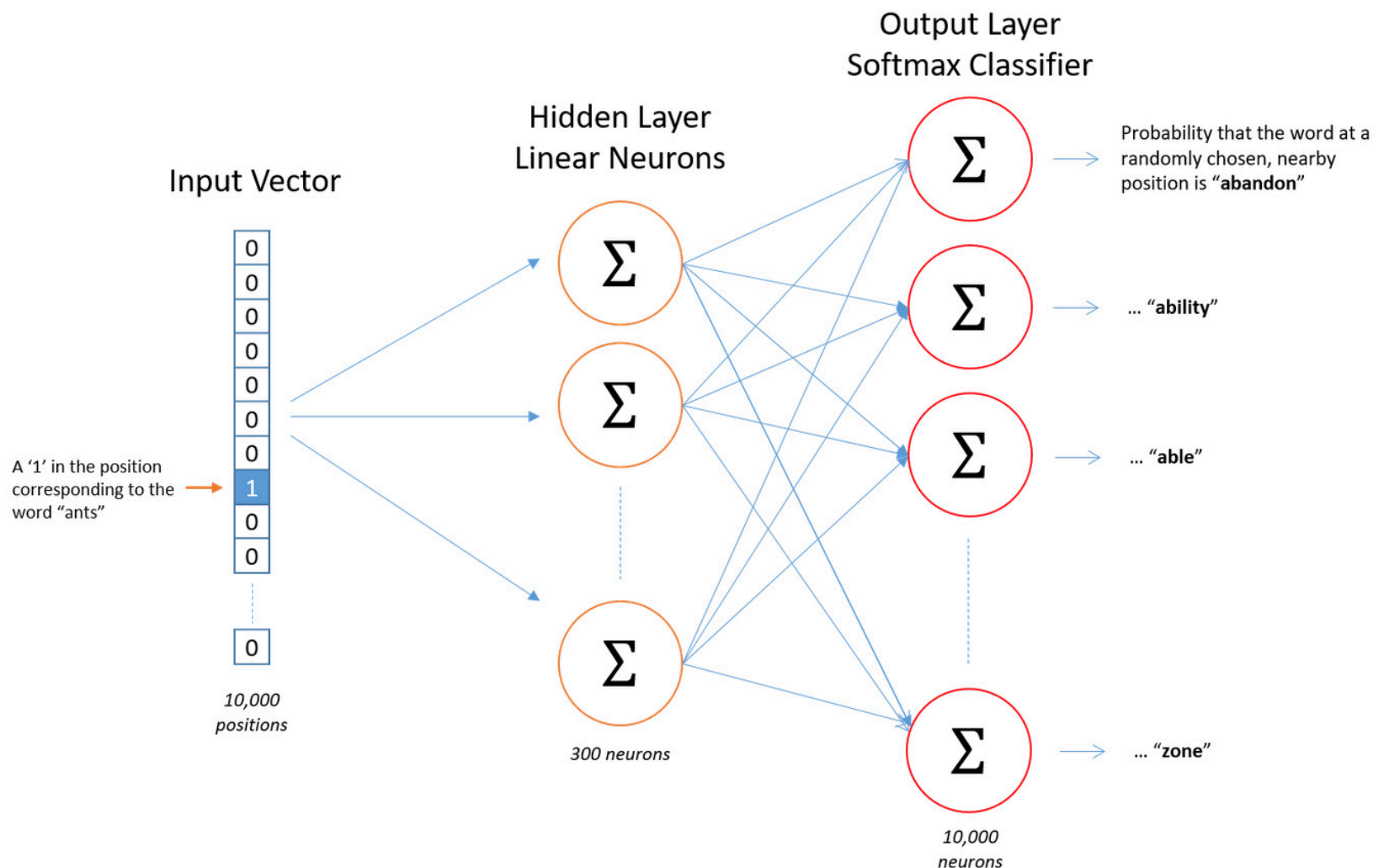
mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

W2V: The Network

- Say we have 10000 words in our vocab
- Then the input word is 10000 dimensional vector
 - Example: Cat word will have 'Cat' coordinate 1, everything else 0
- The true label (word) is also 10000 dimensional vector
- Network outputs 10000 scores which pass through softmax
 - Each coordinate is the probability that a particular word is the randomly selected nearby word

W2V: The Network

- Notice: No nonlinearity in the hidden layer!



W2V: The Objective

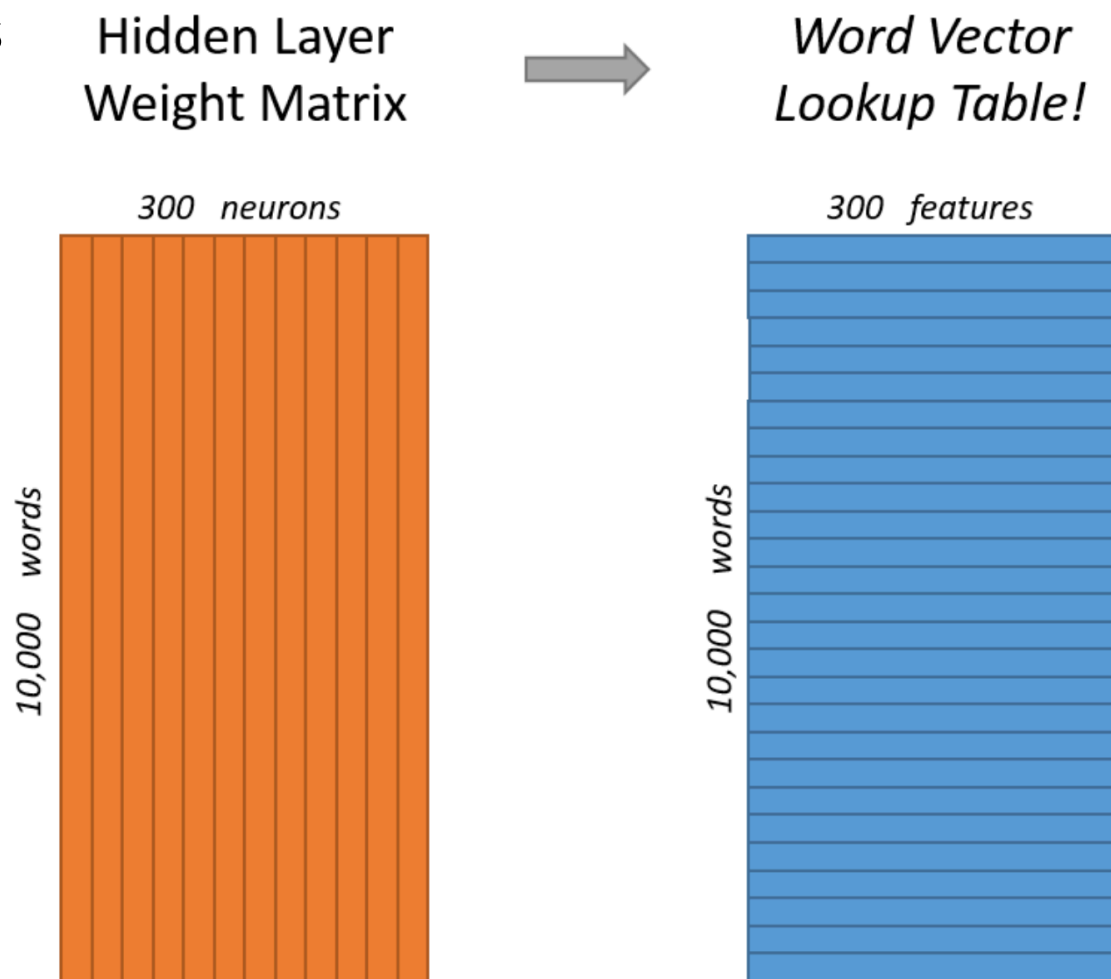
- The objective is to maximize the normalized score (recall normalization by softmax operation) of the correct context word
- Say our training data is made with T words, each having a context window size 2
 - That is, each word is associated with 4 other words
 - Total training data is $4T$
- The objective is $\frac{1}{T} \sum_{t=1}^T \sum_{j \in \{-2, -1, 1, 2\}} \log p(w_{t+j} | w_t)$

W2V: The Hidden Layer

- Is represented by a weight matrix W_0
 - Lets represent it by its transpose (just for convenience)
 - $h^T = x^T W_0^T = x^T W$ for each example
 - Number of rows of W is 10000
 - Number of columns of W is 300
- Then the rows of W are our word vectors!

W2V: The Hidden Layer

- Our real goal was just to learn the hidden layer weights



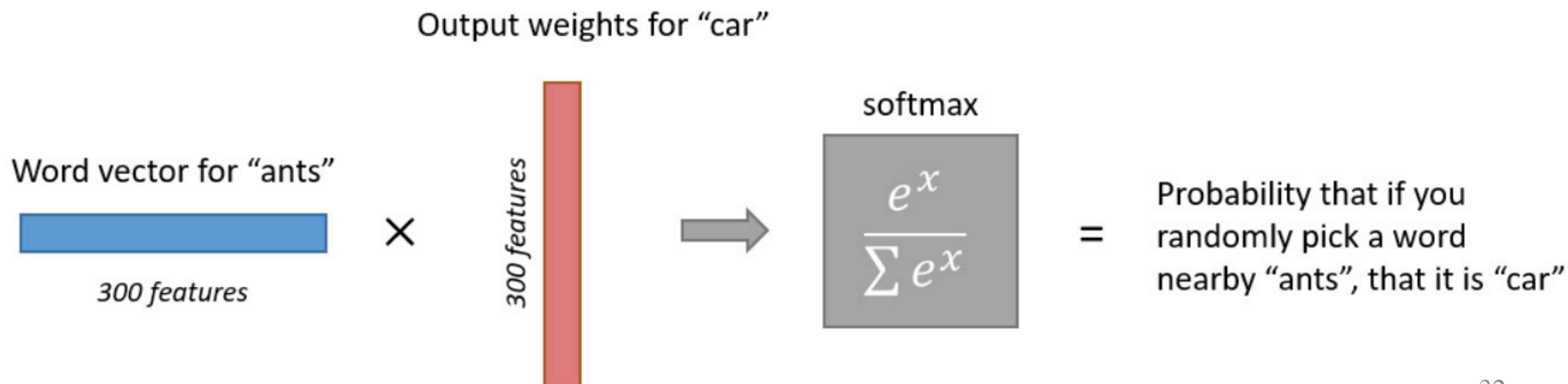
W2V: The Lookup

- Say word 'Cat' has coordinate c for some $c \in \{1, \dots, 10,000\}$
- If we multiply the 1×10000 dim one hot vector for the word 'Cat' with W
- It will just select the c^{th} row of W
 - The output of the hidden layer is the word vector!
- Example visualization

$$[0 \quad 0 \quad 0 \quad 1 \quad 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = [10 \quad 12 \quad 19]$$

W2V: Auxiliary Task Again

- Output of the network is a bunch of normalized scores (i.e., probabilities)
 - Denote the probability that the this word is a nearby word
- Example:
 - Pick the word vector for 'ants'
 - Pic the output neuron for word 'car'



W2V: Intuition for Vectors

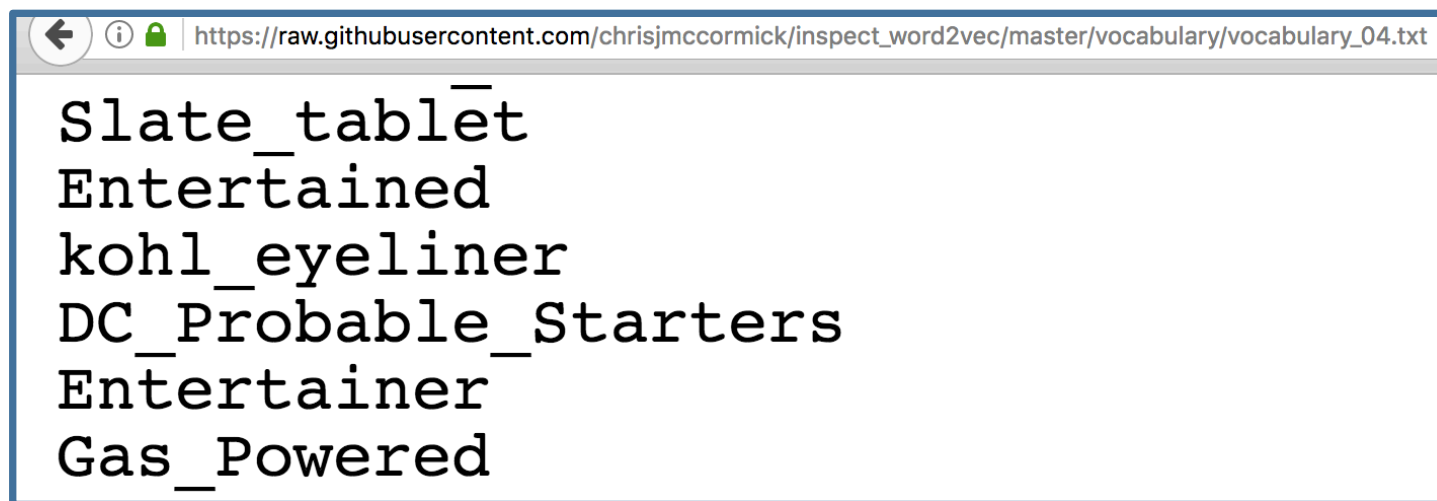
- If two different words have similar “contexts”
 - Similar words are likely to appear around them
 - Then the output probability vectors should be similar
 - For output vectors to be similar the word vector (weights of hidden layer) should be similar
 - Since the inputs are 1-hamming distance apart always

W2V: Intuition for Vectors

- Word2Vec is capturing nothing but the co-occurrence statistics!
- Example:
 - Words like 'university' and 'masters' would have similar contexts, hence similar word vectors
- This will also handle stemming!
 - Example: words like 'car' and 'cars' will have similar vectors because contexts would be similar

W2V: From Google

- 100 Billion words from the Google News Dataset
- Vocab size totals about 3 million!



A screenshot of a web browser window displaying a list of words from a vocabulary file. The address bar shows the URL: https://raw.githubusercontent.com/chrisjmccormick/inspect_word2vec/master/vocabulary/vocabulary_04.txt. The list of words is as follows:

```
Slate_tablet
Entertained
kohl_eyeliner
DC_Probable_Starters
Entertainer
Gas_Powered
```

¹Reference: https://github.com/chrisjmccormick/inspect_word2vec/tree/master/vocabulary

²word2Vec file: <https://drive.google.com/file/d/0B7XkCwpl5KDYNINUTTISS21pQmM/edit>

W2V: From Google

- 100 Billion words from the Google News Dataset
- Vocab size totals about 3 million!

```
In [1]: import gensim
```

```
In [3]: model = gensim.models.Word2Vec.load_word2vec_format('./GoogleNews-vectors-negative300.bin.gz', binary=True)
```

```
In [7]: model.most_similar('good')
```

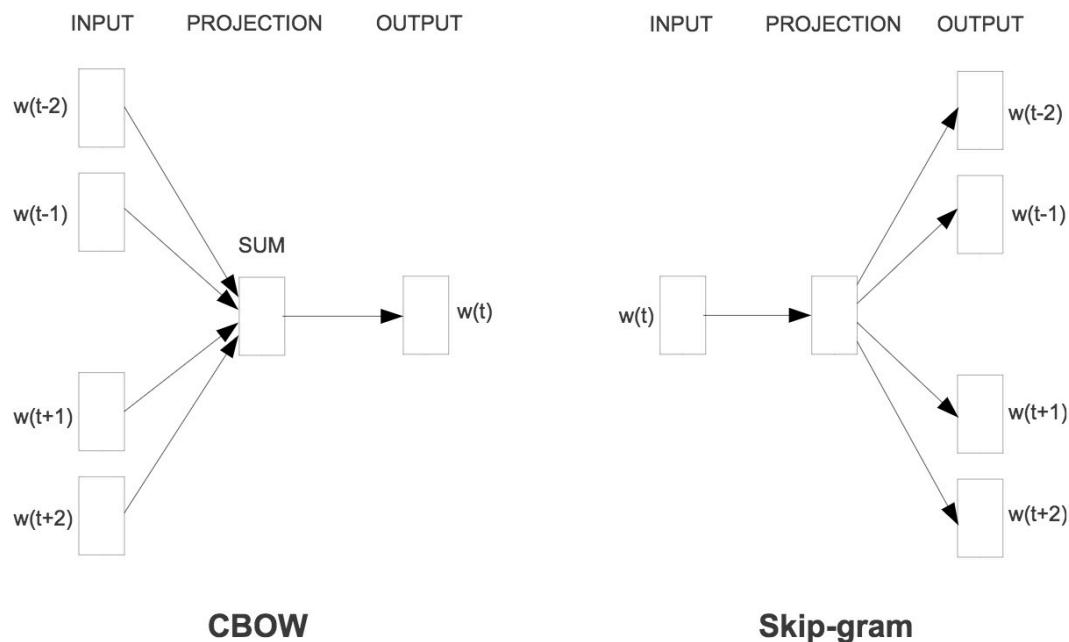
```
Out[7]: [(u'great', 0.7291509509086609),  
         (u'bad', 0.7190051078796387),  
         (u'terrific', 0.6889115571975708),  
         (u'decent', 0.6837348341941833),  
         (u'nice', 0.6836091876029968),  
         (u'excellent', 0.6442928910255432),  
         (u'fantastic', 0.6407778859138489),  
         (u'better', 0.6120729446411133),  
         (u'solid', 0.5806034803390503),  
         (u'lousy', 0.5764203071594238)]
```

¹Reference: https://github.com/chrisjmccormick/inspect_word2vec/tree/master/vocabulary

²word2Vec file: <https://drive.google.com/file/d/0B7XkCwpl5KDYNINUTTISS21pQmM/edit>

W2V: The CBOW Version

- Continuous Bag of Words (CBOW) version of Word2Vec
 - Essentially, a slightly different prediction task



- There is another popular embedding called GLoVe

¹Figure: <https://arxiv.org/pdf/1301.3781.pdf>

²Reference for GLoVe: <http://nlp.stanford.edu/projects/glove/>

Questions?

Today's Outline

- Quick Review
- CNN Visualization for Fun
- Deeper Nets: Recent Advances
- Practical Tips

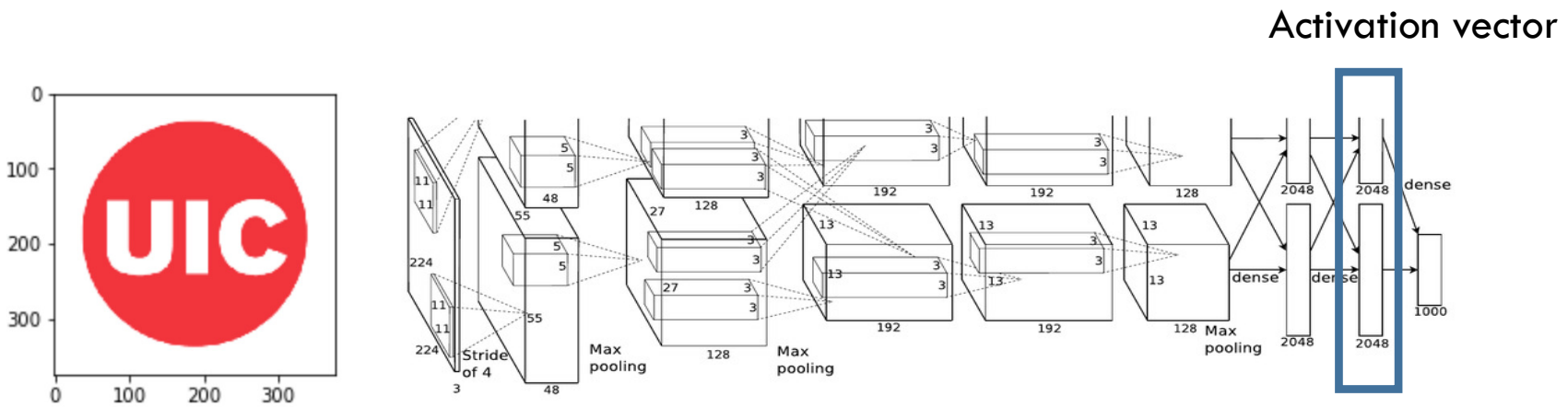
CNN Visualization for Fun

CNN Visualization for Fun

- What we know
 - Each layer extracts features of increasing complexity
 - First layer may look for basic things like edges
 - Intermediate layers may look for components
 - Final layer neurons activate for very complex things like a face or a building
- Lets look at a fun way to further our intuition of what goes on in a CNN ...

Neural Style Transfer (I)

- Image transformation task (merge an image style onto another)

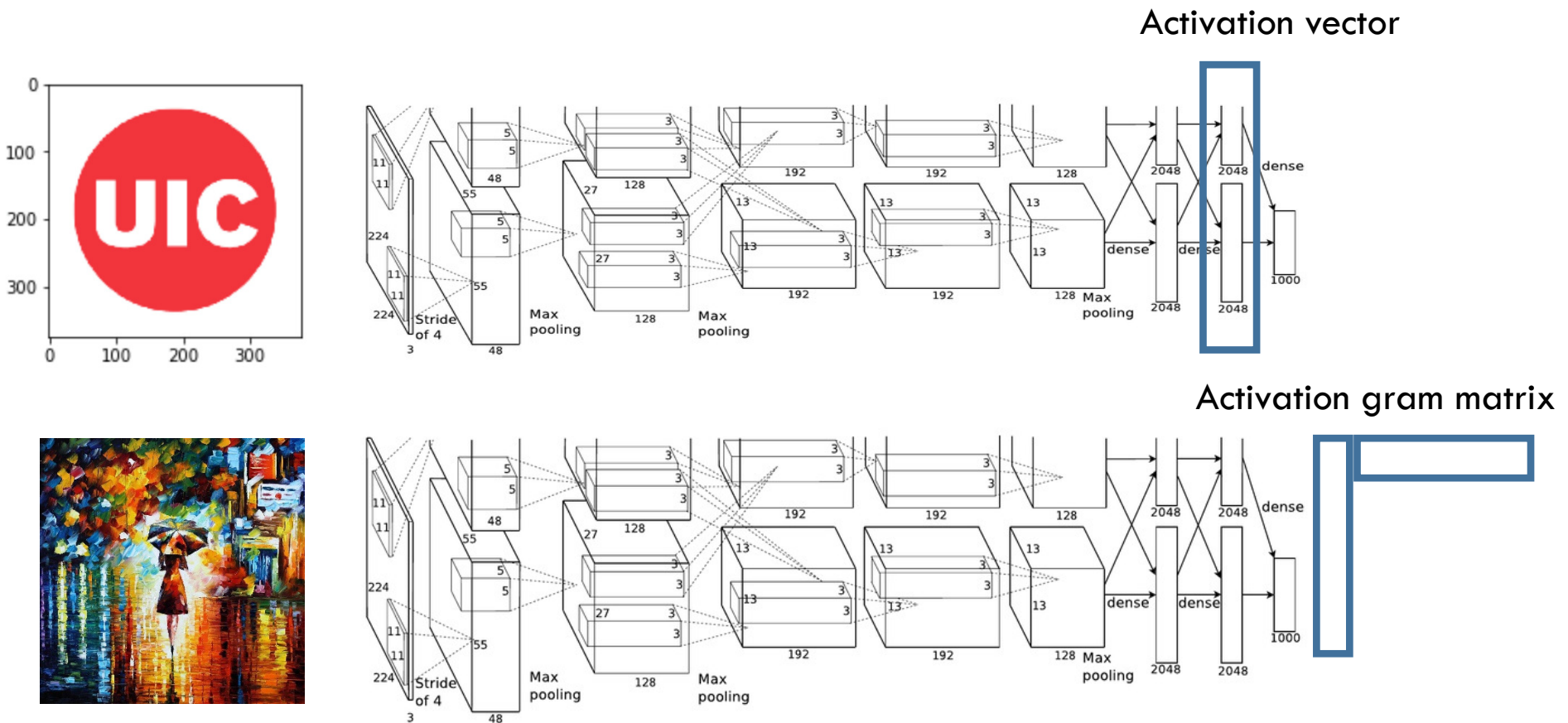


¹Figure: <https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>

²Work: <http://cs.stanford.edu/people/jcjohns/papers/eccv16/JohnsonECCV16.pdf>

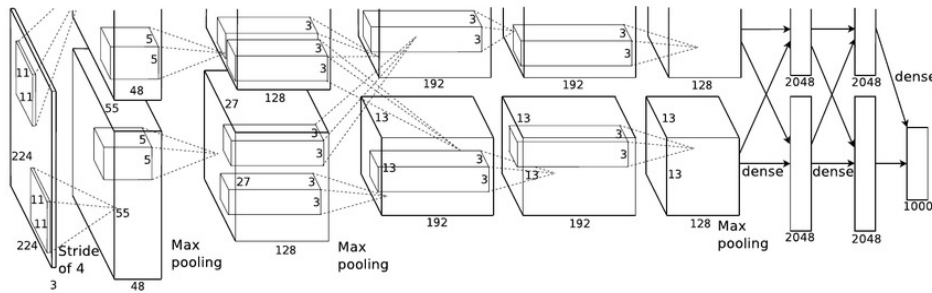
Neural Style Transfer (II)

- Image transformation task (merge an image style onto another)

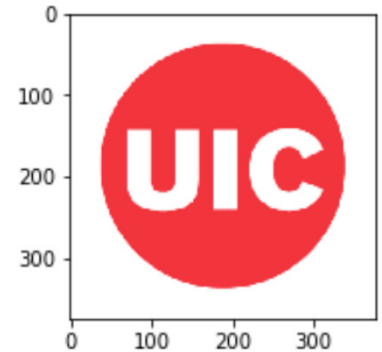


Neural Style Transfer (III)

- Optimizing for an image that satisfies: match content of first image and style of second image



Match activation vector of

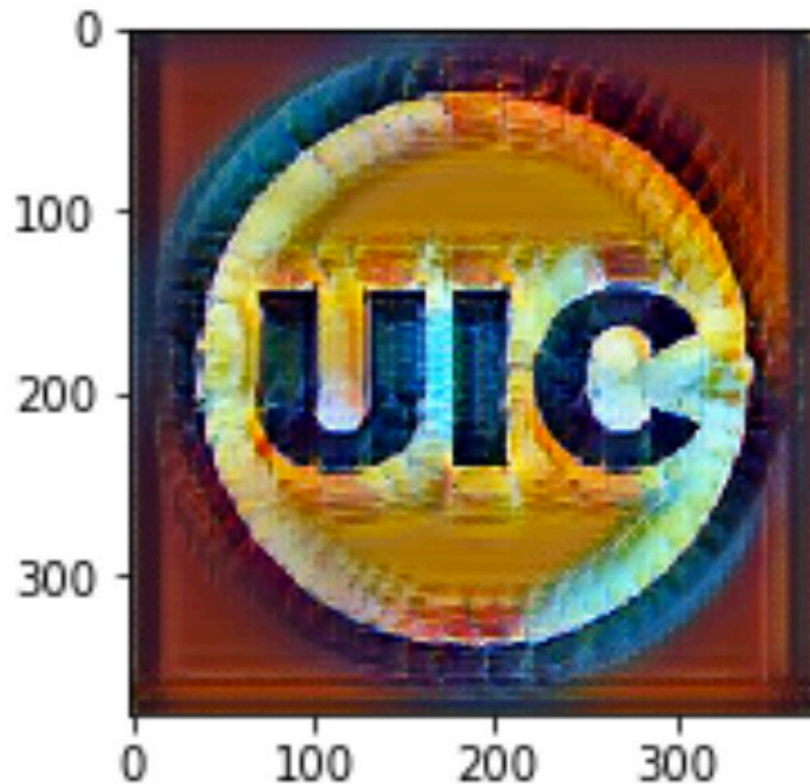


And match activation gram
matrices of

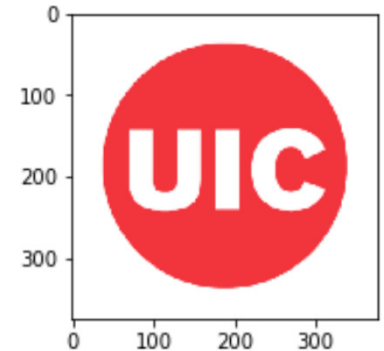


Neural Style Transfer (IV)

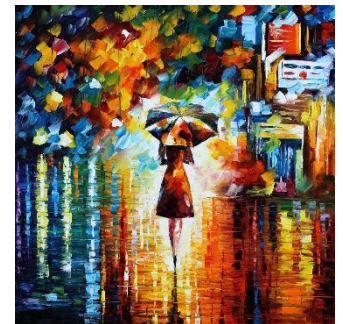
- Results in



Match activation vector of



And match activation gram matrices of



Neural Style Transfer (V)



Questions?

Today's Outline

- Quick Review
- CNN Visualization for Fun
- Deeper Nets: Recent Advances
- Practical Tips

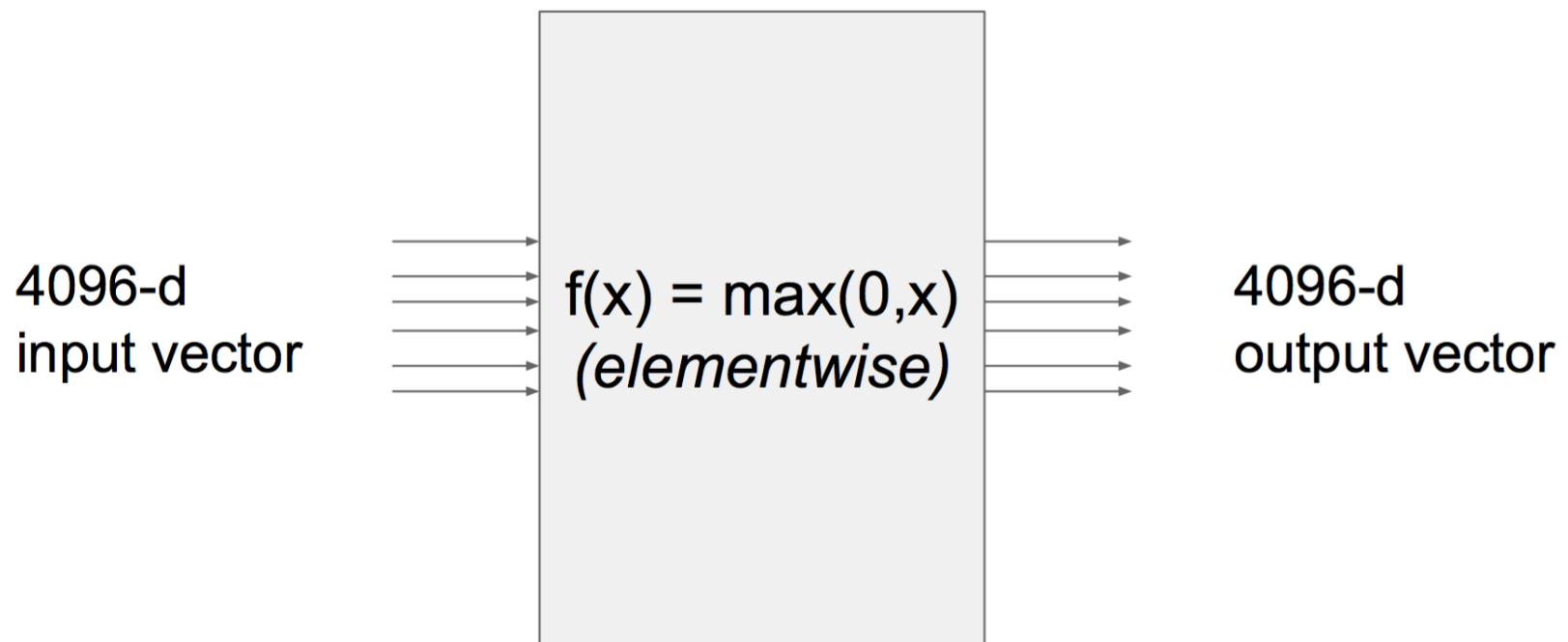
Engineering Deeper Nets

Engineering Deeper Nets

- Remember, we generally have millions of parameters in deep neural networks
 - What if we want deeper networks?
 - How to handle computation?
 - How to handle memory?
- Several computational and other best practices have been proposed and investigated
- We will look at some:
 - Vectorization
 - Architectures
 - Parallelism
 - Precision

Deeper Nets: Vectorization

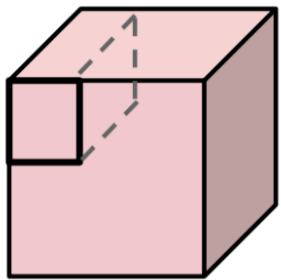
- Vectorization
 - We have already seen this in the example classifiers



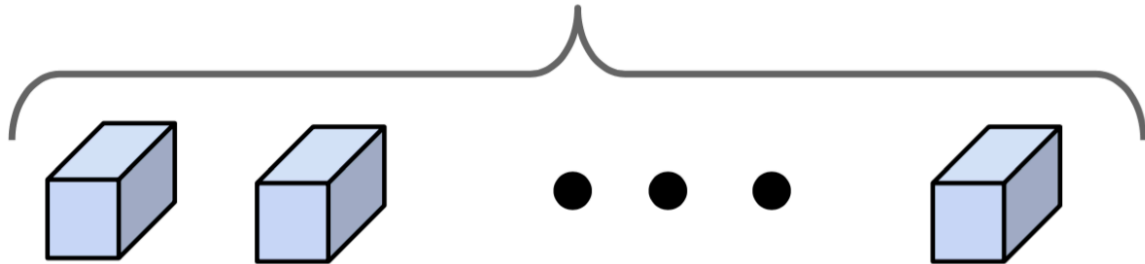
Deeper Nets: Vectorization

- Even the filter computations in the CONV layers

Feature map: $H \times W \times C$



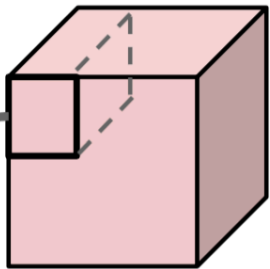
Conv weights: D filters, each $K \times K \times C$



Deeper Nets: Vectorization

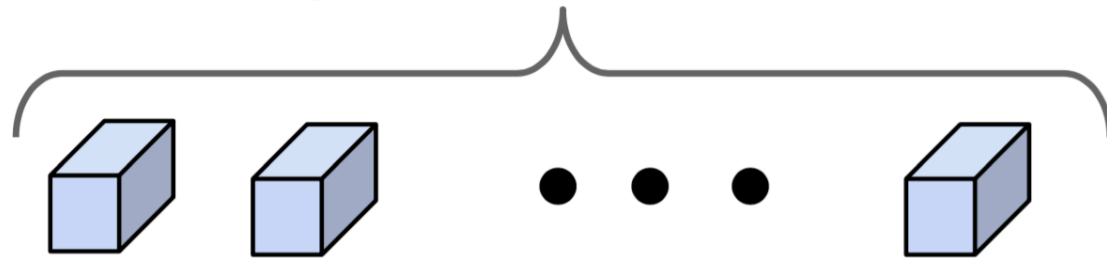
- Even the filter computations in the CONV layers

Feature map: $H \times W \times C$



Reshape $K \times K \times C$
receptive field to column
with K^2C elements

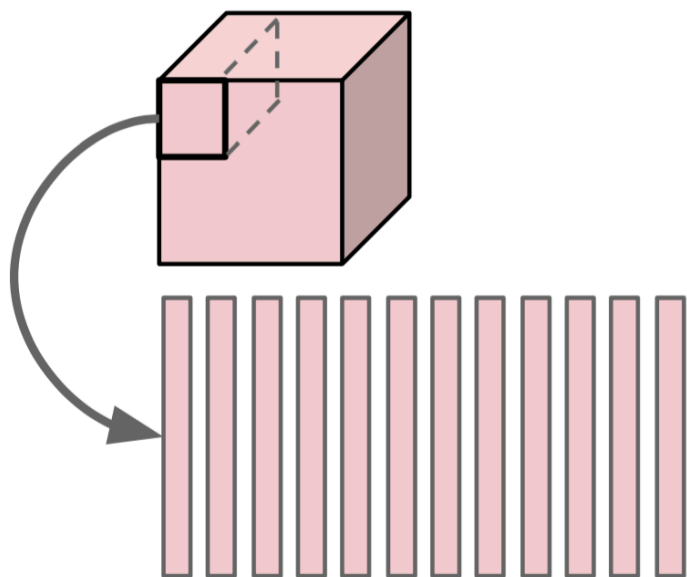
Conv weights: D filters, each $K \times K \times C$



Deeper Nets: Vectorization

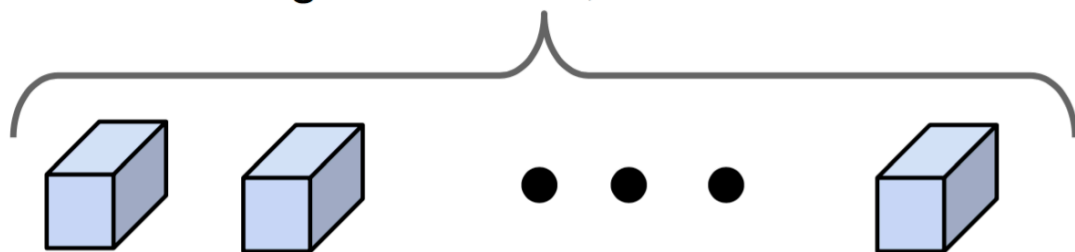
- Even the filter computations in the CONV layers

Feature map: $H \times W \times C$



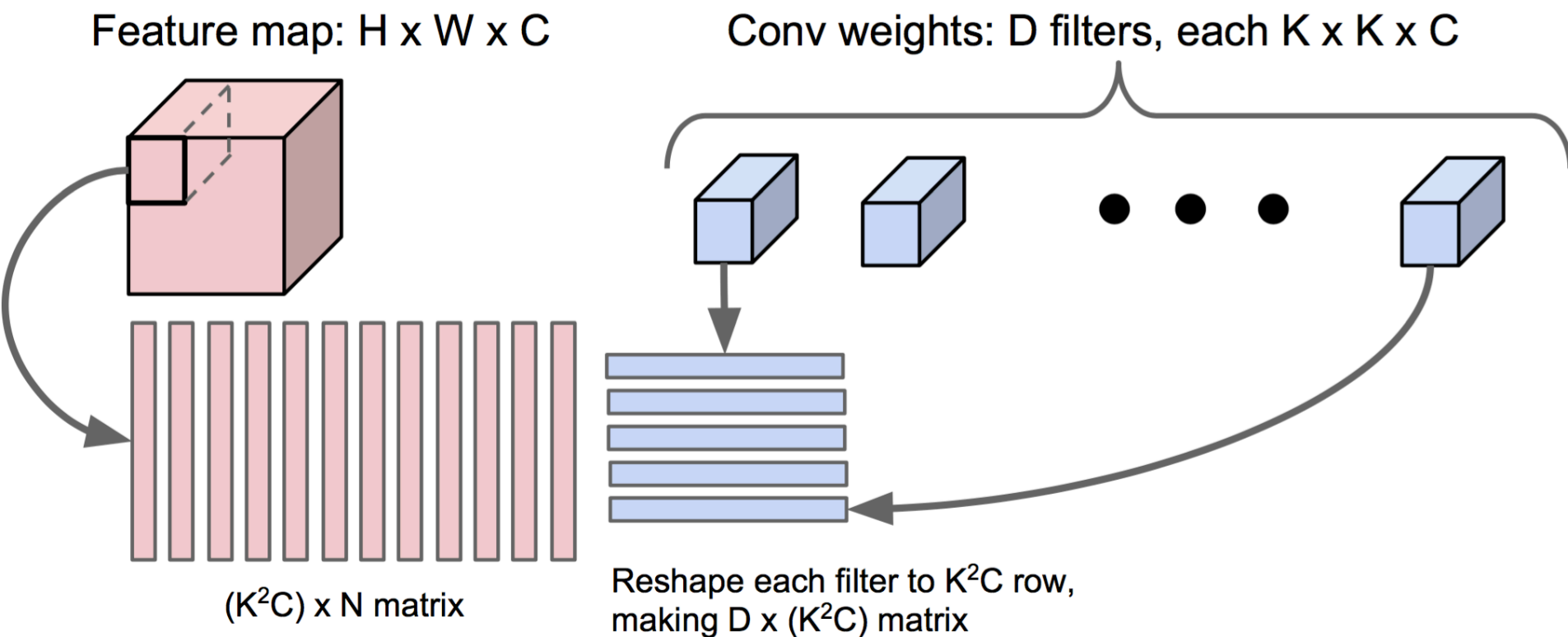
Repeat for all columns to get $(K^2C) \times N$ matrix
(N receptive field locations)

Conv weights: D filters, each $K \times K \times C$

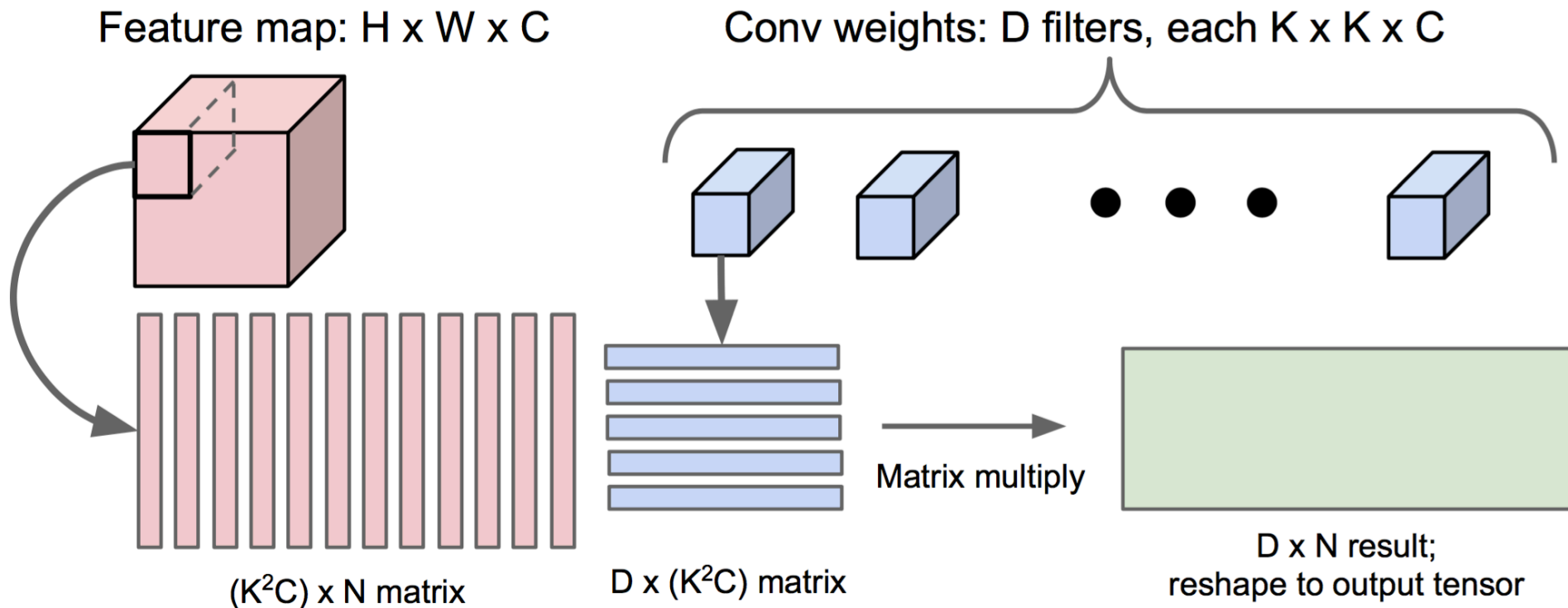


Deeper Nets: Vectorization

- Even the filter computations in the CONV layers



Deeper Nets: Vectorization



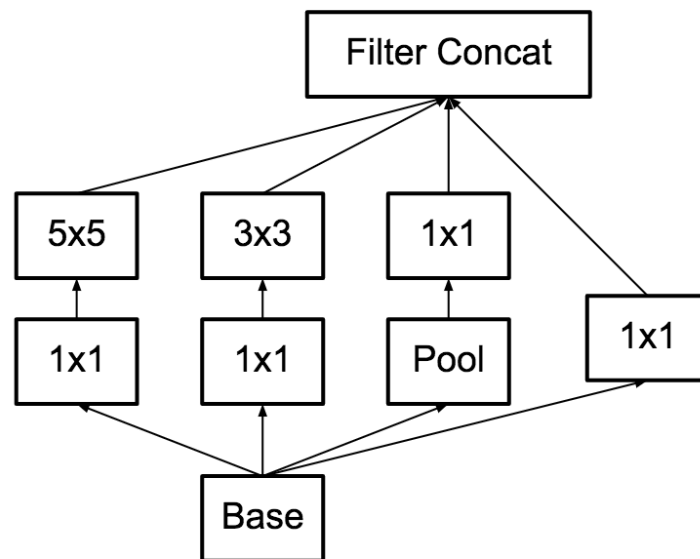
- Extensions and alternatives:
 - Faster matrix multiplies (e.g., Strassen's algorithm)
 - **Fast Fourier Transform** routines

¹Related work: <https://arxiv.org/abs/1509.09308>

²Figure: <http://cs231n.stanford.edu/>

Deeper Nets: New Architectures

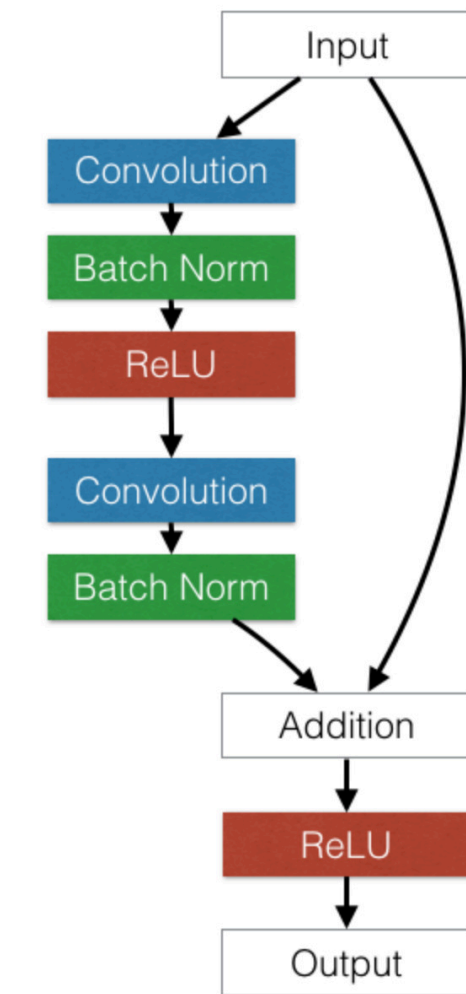
- GoogLeNet (2014 ILSVRC winner)
 - 22 layers
 - Notion of modules



- It has 5 million parameters
 - AlexNet (2012 winner) has 60 million
 - VGG Net has 180 million

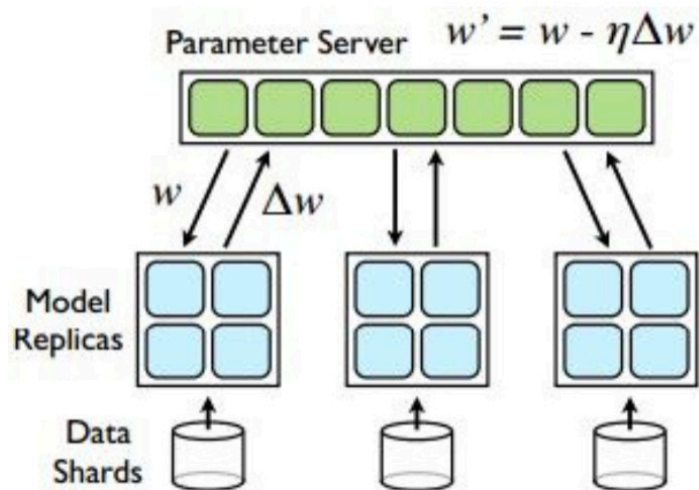
Deeper Nets: New Architectures

- ResNet (2015 ILSVRC winner):
 - Add skip connections
 - Notion of a residual block
- Overcome optimization difficulties
- 152 layers (8x VGG depth)



Deeper Nets: Parallelism

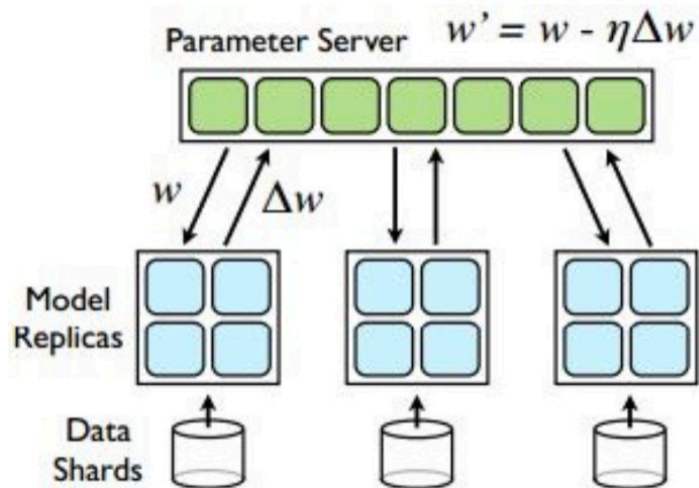
- Software systems such as Tensorflow can take care of data and model parallelism



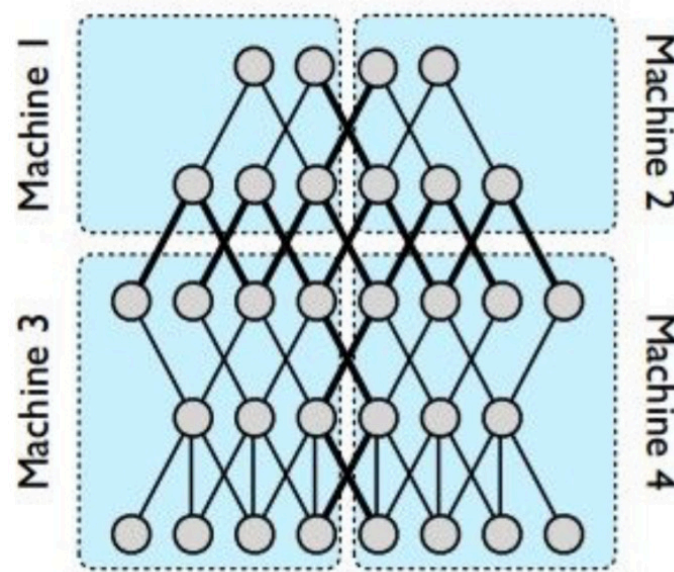
Data parallelism

Deeper Nets: Parallelism

- Software systems such as Tensorflow can take care of data and model parallelism



Data parallelism



Model parallelism

Deeper Nets: Compute Precision

- Smaller precision does not significantly impact performance
 - For example, check effect on computation time when using numpy by changing dtype to float32, float64 etc)

AlexNet (One Weird Trick paper) - Input 128x3x224x224

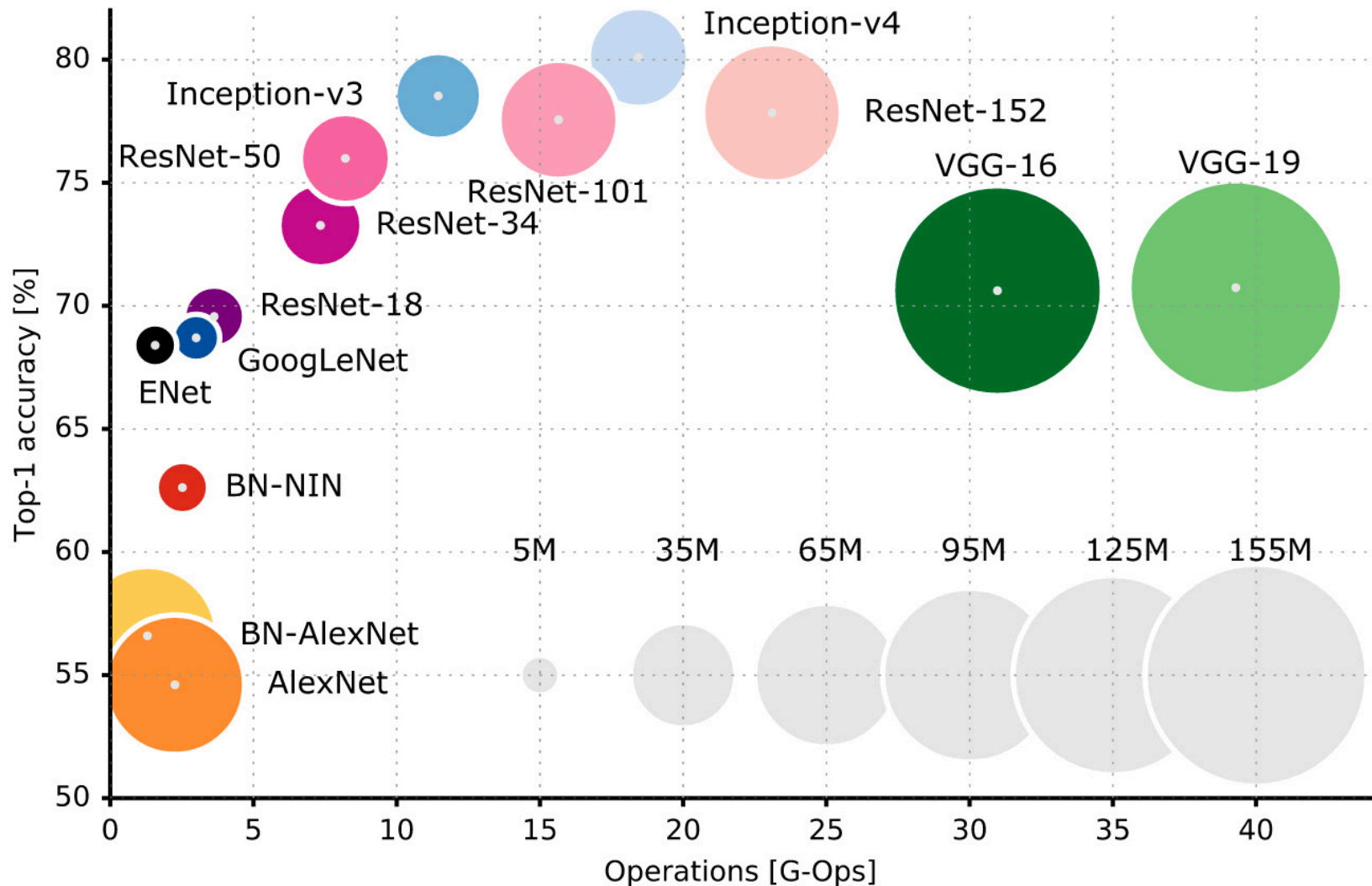
Library	Class	Time (ms)	forward (ms)	backward (ms)
CuDNN[R4]-fp16 (Torch)	cudnn.SpatialConvolution	71	25	46
Nervana-neon-fp16	ConvLayer	78	25	52
CuDNN[R4]-fp32 (Torch)	cudnn.SpatialConvolution	81	27	53
TensorFlow	conv2d	81	26	55

- One can also use **1-bit** activations and weights (retain higher precision for gradients)

¹Related work: <https://arxiv.org/abs/1602.02830>

²Figure: <https://github.com/soumith/convnet-benchmarks>

New Architectures (Addendum)



Questions?

Today's Outline

- Quick Review
- CNN Visualization for Fun
- Deeper Nets: Recent Advances
- Practical Tips

Deep Learning in Practice

From the “Nuts and Bolts of Applying Deep Learning” talk by Andrew Ng (Oct 2016)

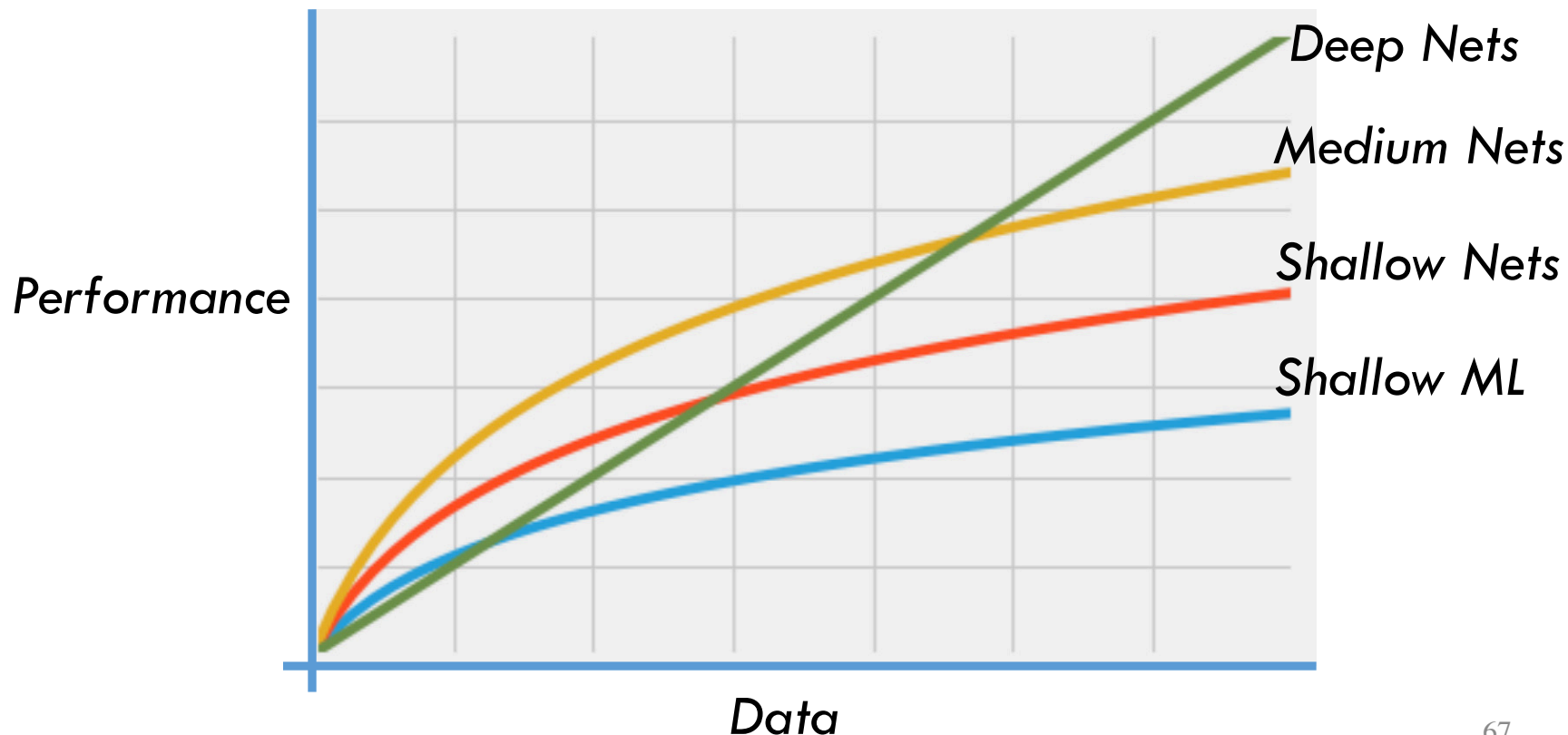
The Practice of Deep Learning

- Major DL Trends in the Industry
- End-to-end DL
- Bias/Variance
- Human Level Performance

DL Research is different from DL for production!

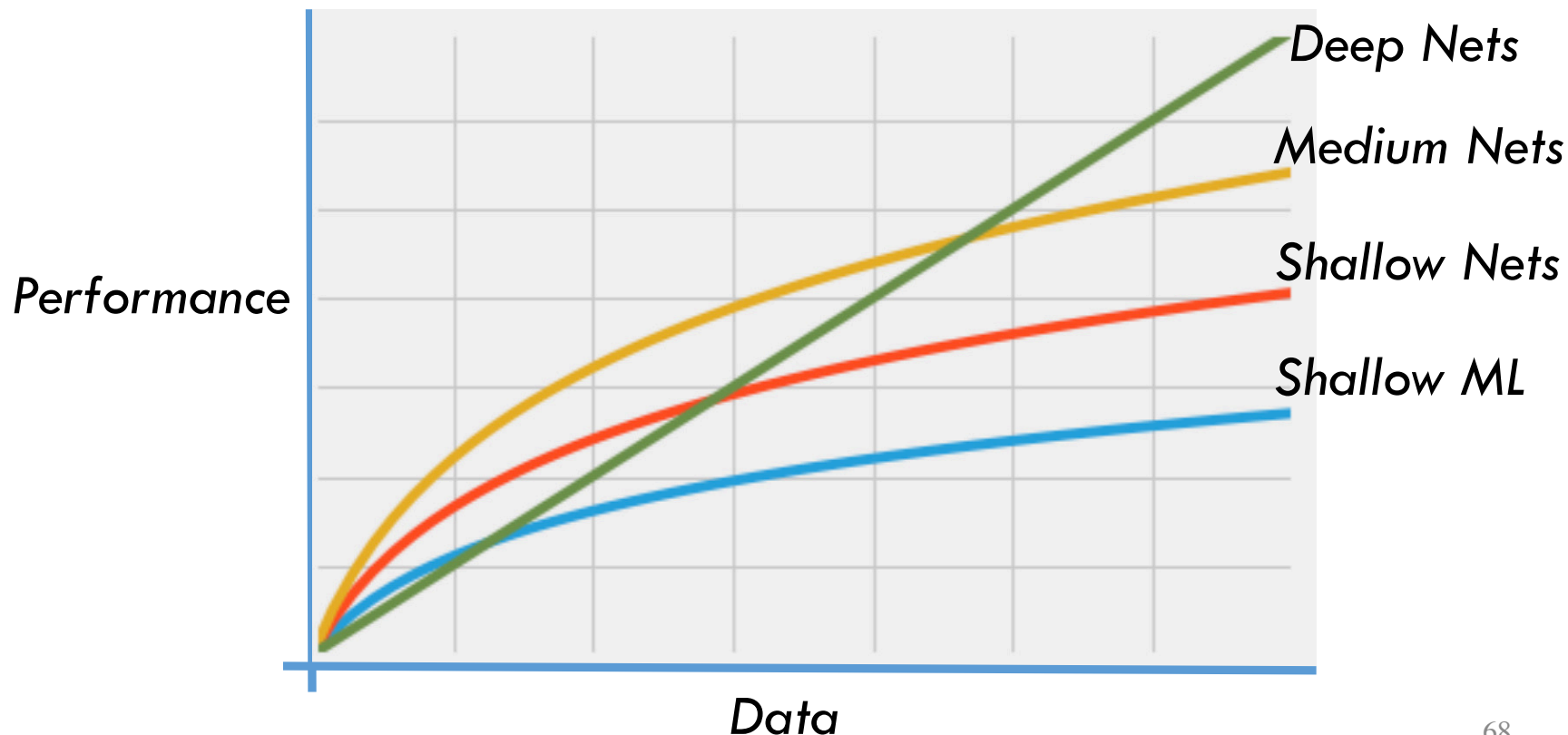
Major DL Trends

- Scale is what is behind the success of deep learning methods
- And it has impact on the workflow of an ML project



Major DL Trends

- Deep Nets capacity can absorb all the data
- Small data regime: Just use shallow ML



Production Ready DL models

- Fully connected models: FFNs
- Sequence models: RNNs
- Image models: CNNs

So focus on these for now.

- Not-so-production-ready techniques:
 - Unsupervised
 - Reinforcement learning

Rise of End-to-End DL

- Shallow ML:
 - Outputs were numbers
 - Example: sentiment is $+/-$
- DL:
 - Output can be very rich
 - Example:
 - Image captioning, input is image, output is text string
 - Machine translation
 - Speech recognition: speech to text directly , no need for phonemes!
- Unfortunately, this cannot be done always!

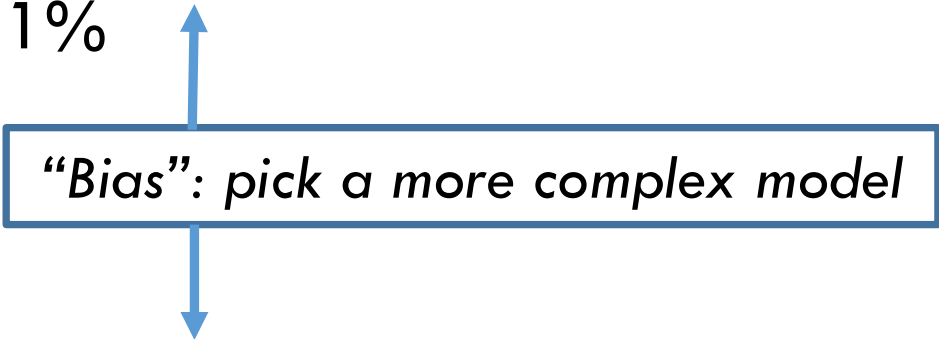
Rise of End-to-End DL

- Unfortunately, this cannot be done always!
- Example
 - Scenario 1 (less realistic):
 - Image to steering action
 - Scenario 2 (more realistic):
 - Image to detect pedestrians, other cars
 - Explicit trajectory planner plans a path
 - Then steering control is actuated
- Scenario 1 may work if you have **lots** of data

What to do after Training?

- Lots of decisions to potentially improve
- How to take these decisions?
- Lets look at Bias-Variance tradeoff

What to do after Training?

- Example Goal: Build a human level speech recognition system
 - Measure (error)
 - Human-performance: 1%
 - Training error: 5%
 - Validation error: 6%
- 
- The diagram consists of a rectangular box with a blue border containing the text *"Bias": pick a more complex model*. From the top center of the box, a blue arrow points upwards towards the 'Human-performance: 1%' bullet point. From the bottom center of the box, a blue arrow points downwards towards the 'Training error: 5%' bullet point.

What to do after Training?

- Example Goal: Build a human level speech recognition system
- Measure (error)
 - Human-performance: 1%

- Training error: 2%

“Variance”: Get more data, try regularizing, early stopping

- Validation error: 6%

What to do after Training?

- Example Goal: Build a human level speech recognition system

- Measure (error)

- Human-performance: 1%

“Bias”: pick a more complex model

- Training error: 5%

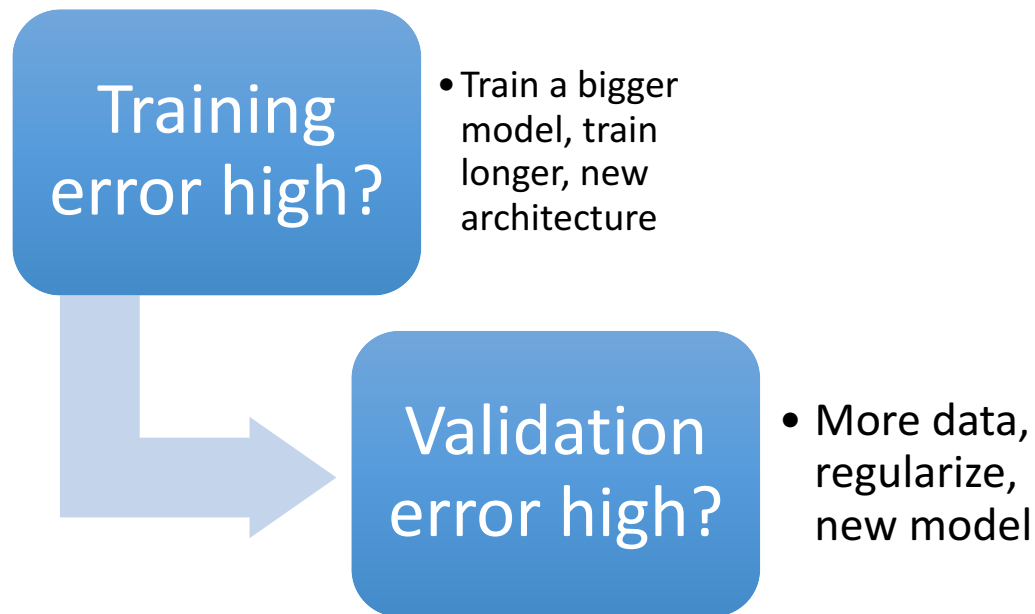
“Variance”: Get more data, try regularizing, early stopping

- Validation error: 10%

- Always good to compute all these numbers!

ML Workflow

- Different from the shallow ML era, we always have a way out



- In shallow ML, there was a tradeoff of 'bias' and 'variance', here the coupling is weaker
- More data, bigger model → scale!

More Data Cheaply

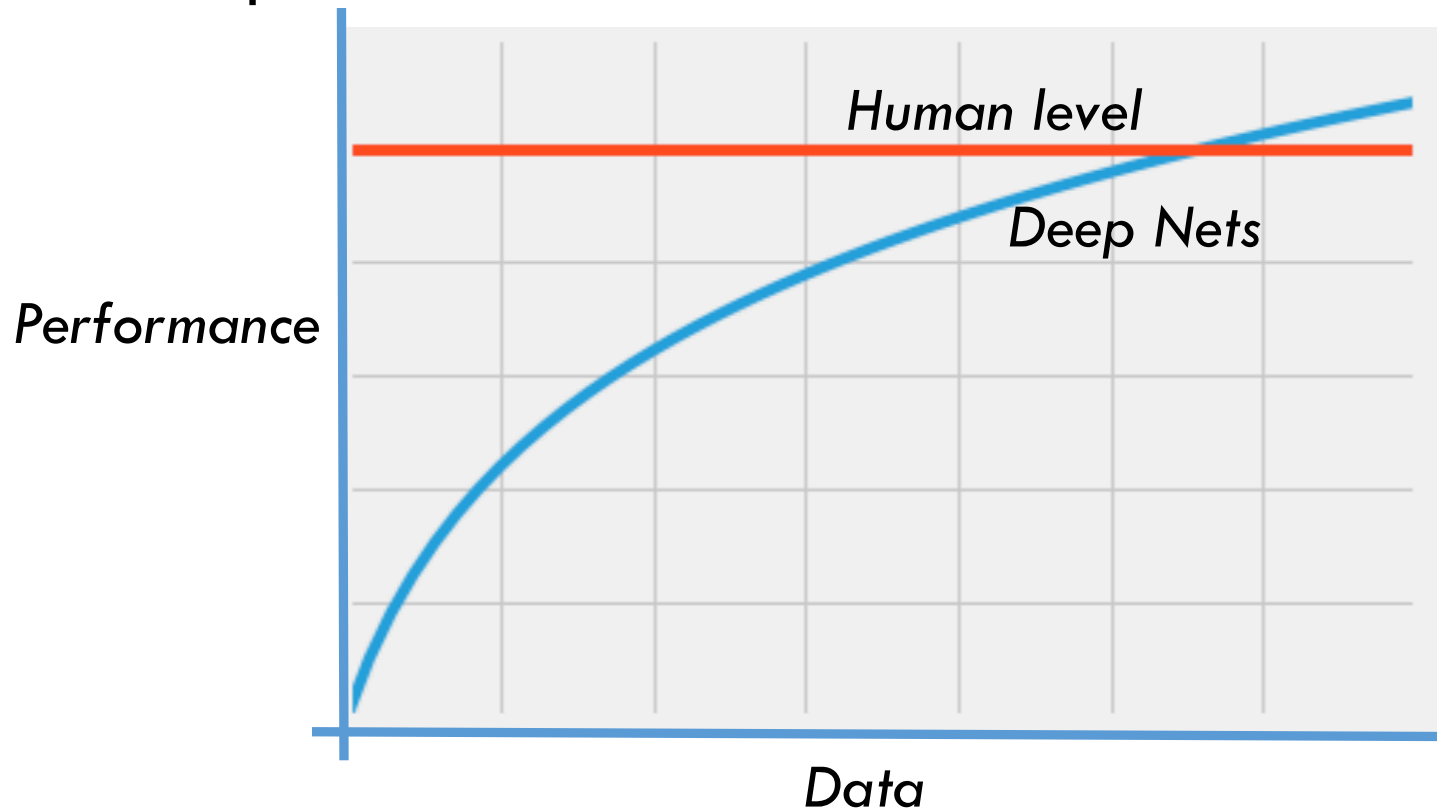
- Shift towards hand engineering of data rather than features
 - Example:
 - For OCR data: get an image, put text in a random font
 - For speech data: add speech to car, outdoor or crowded noise as needed
 - Video games data for reinforcement learning
- Sometimes this is not possible...
- Also have a (logical) unified data warehouse with proper access management.

Validation and Test

- Validation examples should be very similar to test data
 - Validation data is the problem spec for the ML project.
 - If it is not the same as test, then months of effort wasted
- Validation performance
 - Break down the error to figure out where errors are being made
 - There will be a risk of overfitting on this compared to test eventually.

Human Level Performance

- Progress after human-level becomes difficult
 - Labels come from humans
 - No more knowledge of the theoretical limit of performance



Human Level Performance

- When worse than humans, we can do better by
 - Getting more data
 - Doing error analysis
 - Estimating bias and variance
- Example (error values)
 - Training error: 5%
 - Validation error: 10%

Human Level Performance

- When worse than humans, we can do better by
 - Getting more data
 - Doing error analysis
 - Estimating bias and variance
- Example (error values)
 - Human-performance: 4%
 - Training error: 5%
 - Validation error: 10%

Human Level Performance

- When human performance is lower, it is not clear what to do.
- When its way higher and you are using shallow ML, again it is not clear what to do.
- Example (error values)
 - Human-performance: 15%
 - Training error: 5%
 - Validation error: 10%

What can DL do?

- How should a product manager specify
 - a self-driving car?
 - what accuracy is needed for sentiment classifier?
- Rules of Thumb: DL can do
 - Anything ...

What can DL do?

- How should a product manager specify
 - a self-driving car?
 - what accuracy is needed for sentiment classifier?
- Rules of Thumb: DL can do
 - Anything that a typical person can do in less than 1 sec
 - Example: Look at a picture and tell the expression of the person
 - Predicting outcome of next from a sequence of events

Questions?

Deep Learning Not Discussed

- Covered only core blocks: CNNs, RNNs, Embeddings, GANs
- Lots of aspects we have not touched
 - Optimization: Adagrad, momentum ...
 - Unsupervised methods: autoencoders, sparse coding, ...
 - Many **variations** of state of the art models in speech, text and vision
- Later
 - In graphical models, we will touch VAE again.
 - In reinforcement learning, we will touch CNN again.

Summary

- Recapped some salient aspects of modern deep learning
- Saw a style transfer application
- Considered some of the engineering tricks that are crucial to obtain state of the art performances
- Looked at the practical considerations of ML workflows in workplaces
 - As you pursue business analytics/data science careers, good to have this perspective in mind.

Appendix

Sample Exam Questions

- How does style transfer take place?
- What are a few computer engineering tricks needed to make deep networks work?
- Describe the auxiliary task designed for word2vec.
- When training error is high, what could be a possible approach to fix it? What if the validation error is high?

Advanced Prediction Models

Deep Learning, Graphical Models and Reinforcement
Learning

Today's Outline

- Motivation
- Primer on Graphs
- Directed Graphical Models
- Undirected Graphical Models

Why Graphical Models

- We have seen deep learning techniques for unstructured data
 - Predominantly vision and text/audio
 - We will see control in the last part of the course
 - (Reinforcement Learning)

Why Graphical Models

- We have seen deep learning techniques for unstructured data
 - Predominantly vision and text/audio
 - We will see control in the last part of the course
 - (Reinforcement Learning)
- For structured data, graphical models are the most versatile framework
 - Successfully applications:
 - Kalman filtering in engineering
 - Decoding in cell phones (channel codes)
 - Hidden Markov models for time series
 - ...

Graphical Models Landscape

- What are graphical models good for?
 - Representation
 - Capture uncertainty (joint distribution)
 - Capture **conditional independences** (metadata)
 - Visualization of metadata for a distribution

Graphical Models Landscape

- What are graphical models good for?
 - Representation
 - Capture uncertainty (joint distribution)
 - Capture **conditional independences** (metadata)
 - Visualization of metadata for a distribution
 - Inference
 - Create data structures for computing marginal or conditional distributions **quickly**

Graphical Models Landscape

- What are graphical models good for?
 - Representation
 - Capture uncertainty (joint distribution)
 - Capture **conditional independences** (metadata)
 - Visualization of metadata for a distribution
 - Inference
 - Create data structures for computing marginal or conditional distributions **quickly**
 - Learning
 - Learning the **parameters of the distribution** can be aided by graph techniques

Graphical Models Landscape

- We are learning them in three parts
- Part 1: Representation ([this lecture](#))
 - How does a directed or undirected probabilistic graphical model (DPGM/UPGM) look like? What does it encode?


Graphical Models Landscape

- We are learning them in three parts
- Part 1: Representation ([this lecture](#))
 - How does a directed or undirected probabilistic graphical model (DPGM/UPGM) look like? What does it encode?
- Part 2: Inference
 - How to compute marginal and conditional probabilities efficiently? ([next lecture](#))


Graphical Models Landscape

- We are learning them in three parts
- Part 1: Representation ([this lecture](#))
 - How does a directed or undirected probabilistic graphical model (DPGM/UPGM) look like? What does it encode?
- Part 2: Inference
 - How to compute marginal and conditional probabilities efficiently? ([next lecture](#))
- Part 3: Learning
 - How to estimate the parameters of a DPGM/UPGM? ([next to next lecture](#))


Turing Award




MORE ACM AWARDS



A.M. TURING CENTENARY CELEBRATION WEBCAST





A.M. TURING AWARD WINNERS BY...

ALPHABETICAL LISTING

YEAR OF THE AWARD

RESEARCH SUBJECT




Photo-Essay


BIRTH:
September 4, 1936, Tel Aviv.


EDUCATION:
B.S., Electrical Engineering (Technion, 1960); M.S., Electronics (Newark College of Engineering, 1961); M.S., Physics (Rutgers University, 1965); Ph.D., Electrical Engineering (Polytechnic Institute of Brooklyn, 1965).


EXPERIENCE:
Research Engineer, New York University Medical School (1960–1961); Instructor,


JUDEA PEARL
United States – 2011


CITATION
For fundamental contributions to artificial intelligence through the development of a calculus for probabilistic and causal reasoning.

 SHORT ANNOTATED BIBLIOGRAPHY

 ACM DL AUTHOR PROFILE

 ACM TURING AWARD LECTURE VIDEO

 RESEARCH SUBJECTS

 ADDITIONAL MATERIALS

Judea Pearl created the representational and computational foundation for the processing of information under uncertainty.

He is credited with the invention of *Bayesian networks*, a mathematical formalism for defining complex probability models, as well as the principal algorithms used for inference in these models. This work not only revolutionized the field of artificial intelligence but also became an important tool for many other branches of engineering and the natural sciences. He later created a mathematical framework for *causal inference* that has had significant impact in the social sciences.

Judea Pearl was born on September 4, 1936, in Tel Aviv, which was at that time administered under the British Mandate for Palestine. He grew up in *Bnei Brak*, a Biblical town his grandfather went to reestablish in 1924. In 1956, after serving in the Israeli army and joining a Kibbutz, Judea decided to study engineering. He attended the Technion, where he met his wife, Ruth, and received a B.S. degree in Electrical Engineering in 1960. Recalling the Technion faculty members in a 2012 interview in the *Technion Magazine*, he emphasized the thrill of discovery:

¹Reference: David Sontag, (2013)

Graphical Models vs Deep Learning

Graphical Models

- Probabilistic
- Dependencies btw. RVs
- Low capacity
- Domain knowledge: easy to encode

Deep Neural Networks

- Deterministic
- Input/Output Mapping
- High capacity
- Domain knowledge: hard

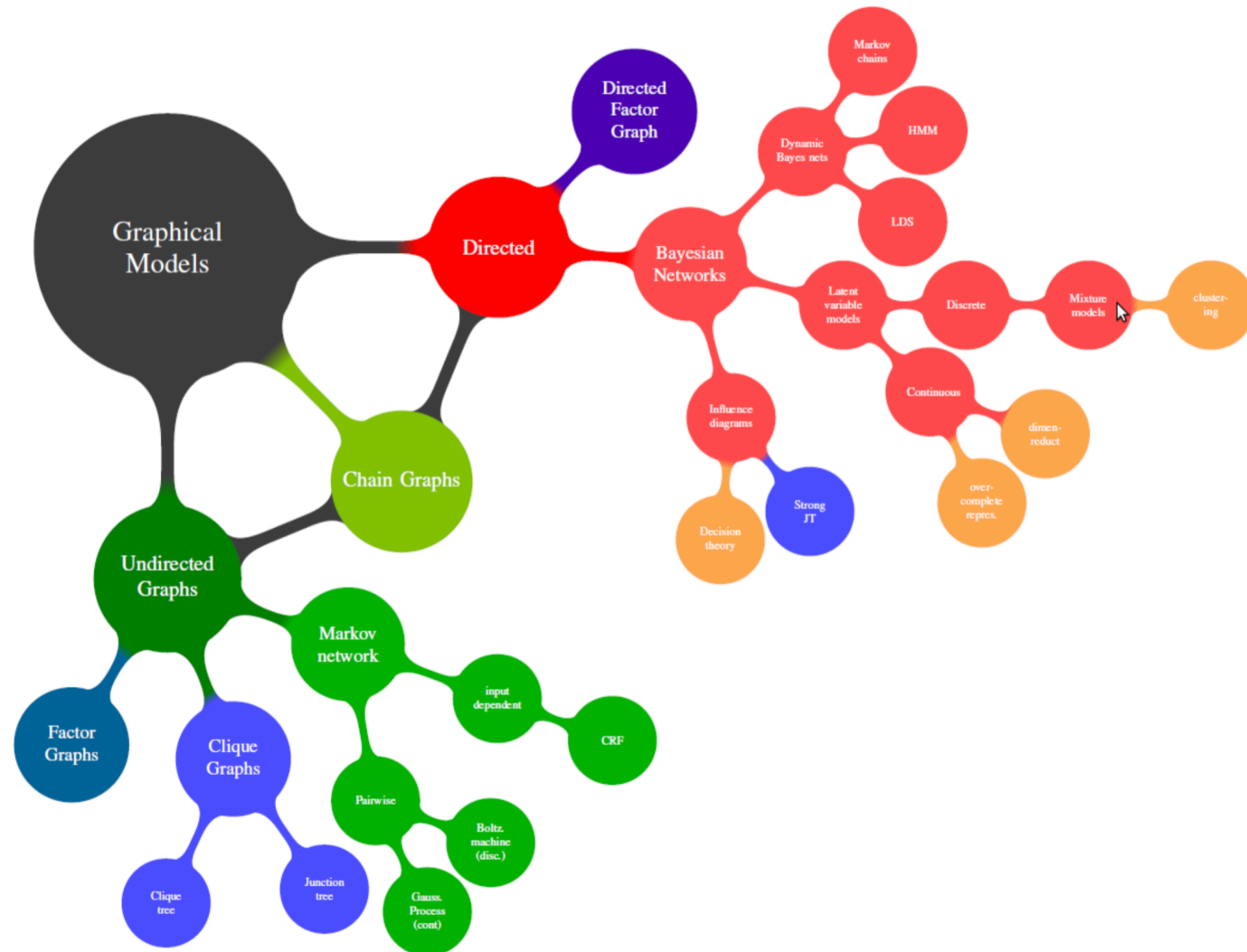
Combinations:

D. Kingma and M. Welling: Auto-encoding variational Bayes. ICLR, 2014.

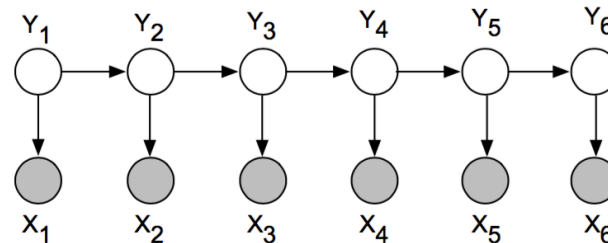
L. Chen, A. Schwing and R. Urtasun: Learning Deep Structured Models. ICML, 2015.

J. Domke: Learning graphical model parameters with approximate marginal inference. PAMI, 2013, Vol. 35, no. 10, pp. 2454–2467.

Graphical Models Landscape



Application 1: Hidden Markov Model

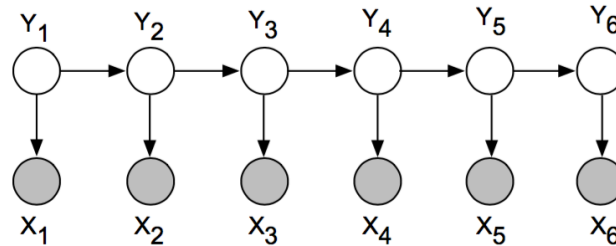


- Frequently used for speech recognition and part-of-speech tagging
- Joint distribution factors as:

$$p(\mathbf{y}, \mathbf{x}) = p(y_1)p(x_1 | y_1) \prod_{t=2}^T p(y_t | y_{t-1})p(x_t | y_t)$$

- $p(y_1)$ is the distribution for the starting state
 - $p(y_t | y_{t-1})$ is the *transition* probability between any two states
 - $p(x_t | y_t)$ is the *emission* probability
- What are the conditional independencies here? For example,
 $Y_1 \perp \{Y_3, \dots, Y_6\} \mid Y_2$

Application 1: Hidden Markov Model



- Joint distribution factors as:

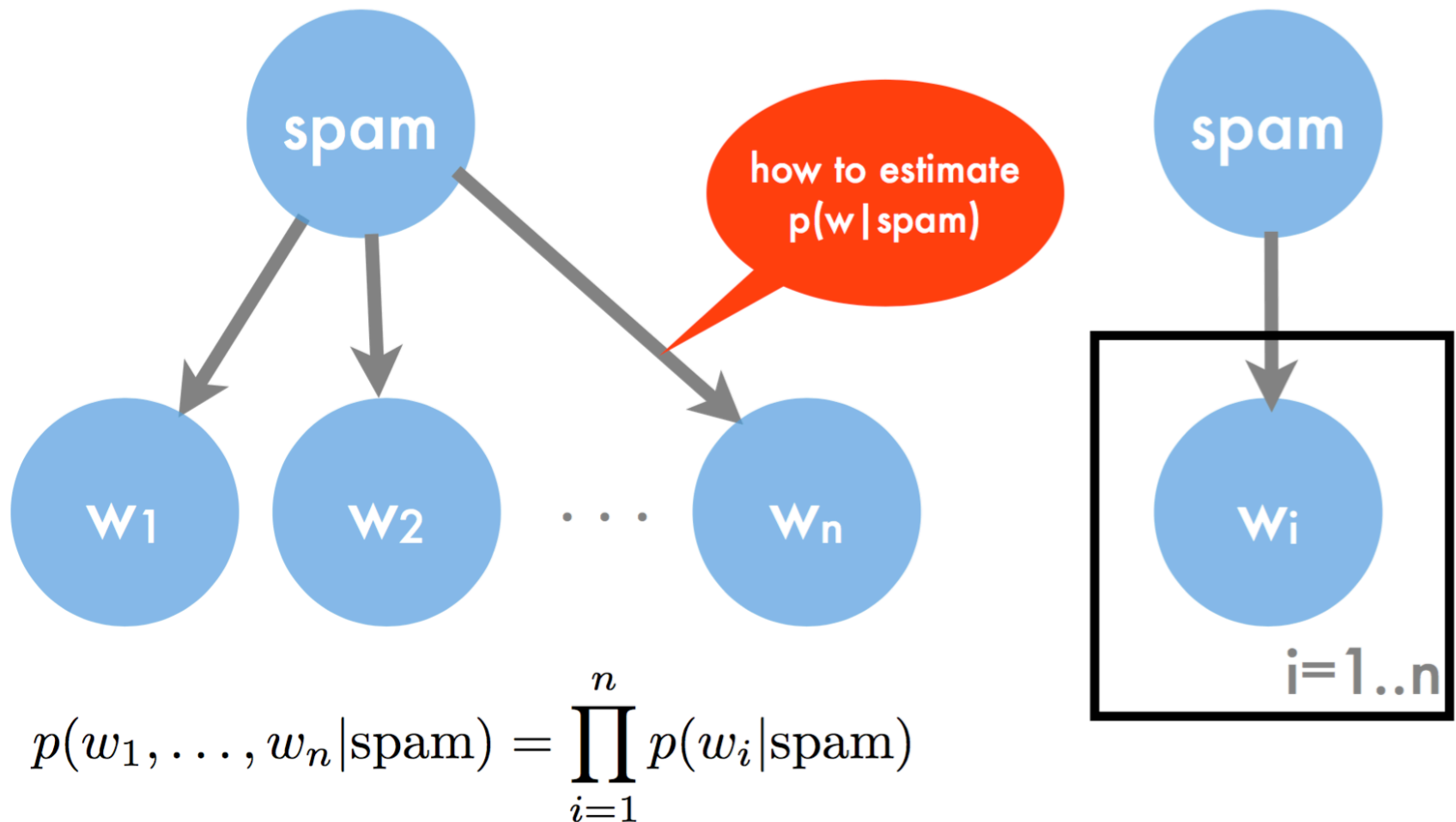
$$p(\mathbf{y}, \mathbf{x}) = p(y_1)p(x_1 | y_1) \prod_{t=2}^T p(y_t | y_{t-1})p(x_t | y_t)$$

- A **homogeneous** HMM uses the same parameters (β and α below) for each transition and emission distribution (**parameter sharing**):

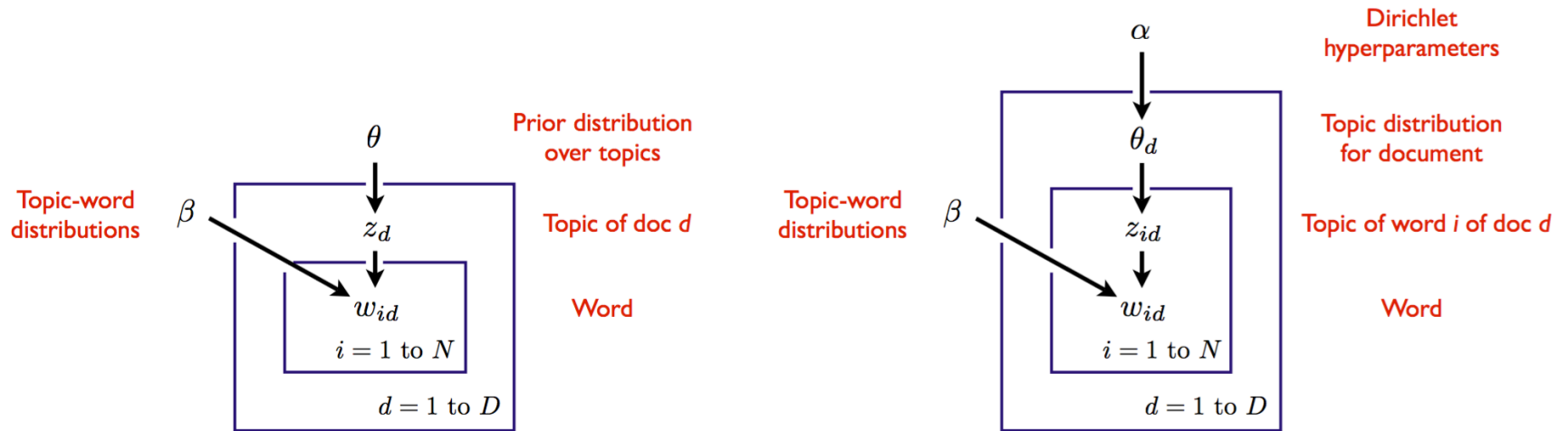
$$p(\mathbf{y}, \mathbf{x}) = p(y_1)\alpha_{x_1, y_1} \prod_{t=2}^T \beta_{y_t, y_{t-1}} \alpha_{x_t, y_t}$$

How many parameters need to be learned?

Application 2: Naïve Bayes Spam Filter



Application 3: Latent Dirichlet Allocation



- Model on left is a **mixture model**
 - Called *multinomial* naive Bayes (a word can appear multiple times)
 - Document is generated from a single topic
- Model on right (LDA) is an **admixture model**
 - Document is generated from a distribution over topics

Application 4: Conditional Random Field

- **Conditional random fields** are undirected graphical models of conditional distributions $p(\mathbf{Y} \mid \mathbf{X})$
 - \mathbf{Y} is a set of **target variables**
 - \mathbf{X} is a set of **observed variables**
- A CRF is a Markov network on variables $\mathbf{X} \cup \mathbf{Y}$, which specifies the conditional distribution

$$P(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{c \in C} \phi_c(\mathbf{x}_c, \mathbf{y}_c)$$

with partition function

$$Z(\mathbf{x}) = \sum_{\hat{\mathbf{y}}} \prod_{c \in C} \phi_c(\mathbf{x}_c, \hat{\mathbf{y}}_c).$$

Questions?

Today's Outline

- Motivation
- Primer on Graphs
- Directed Graphical Models
- Undirected Graphical Models

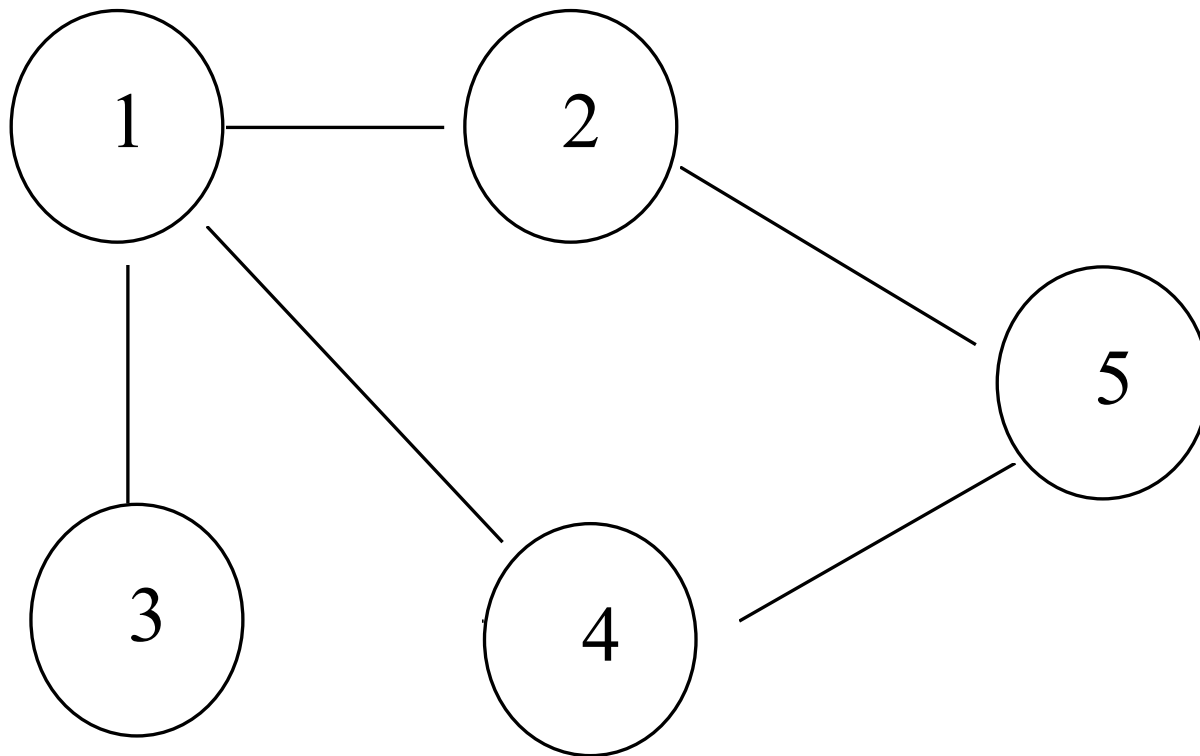
Primer on Graphs

Graph

- A network with
 - Edges (links)
 - Vertices (nodes)
- Heavily used in Computer Science for algorithms and data structures
- Here, we will only need the terminology of graphs.
 - As we will see, their primary purpose will be visualization

Undirected Graph

- An undirected graph
 - Edges have no direction information

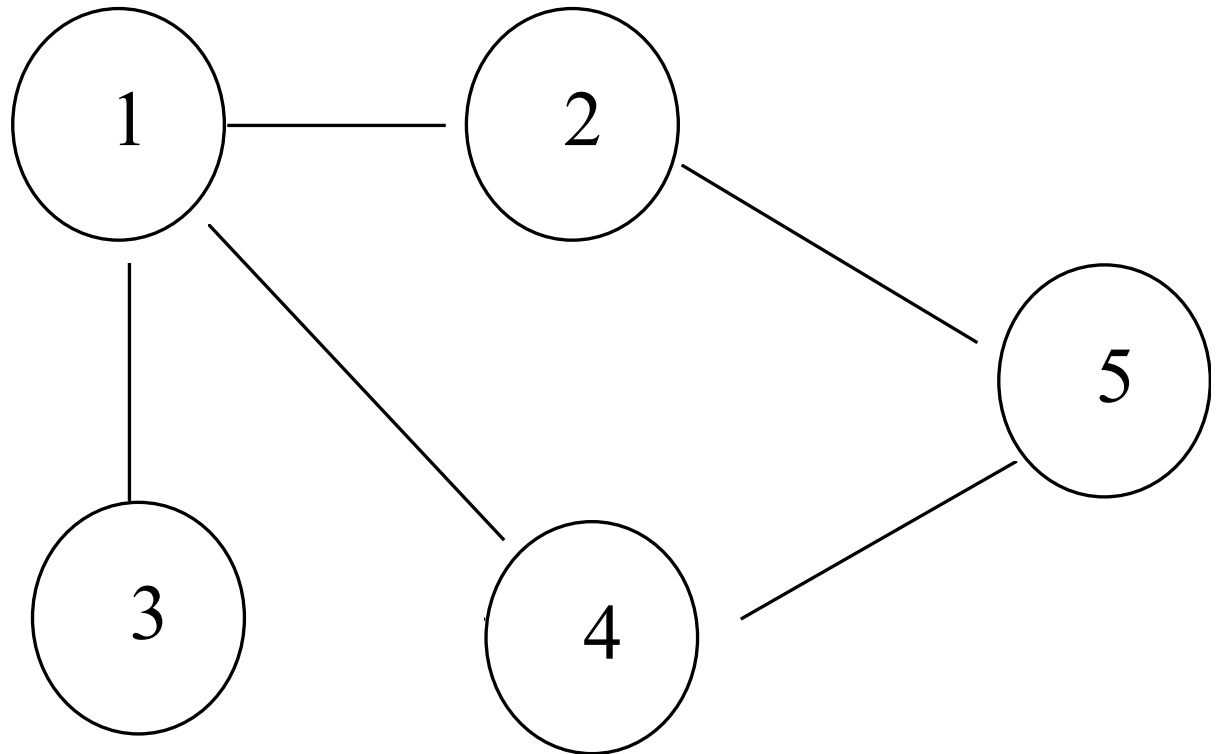


Notation for Undirected Graphs

- Set of vertices denoted $1, \dots, N$
- Size of graph is N
- Edge is an (unordered) pair (i, j)
 - (i, j) is the same as (j, i)
 - indicates that i and j are directly connected
- Maximum number of edges: $N(N - 1)/2$ (order N^2)
- i and j connected if there is a path of edges between them
- Subgraph of G :
 - restrict attention to certain vertices and edges between them

Path

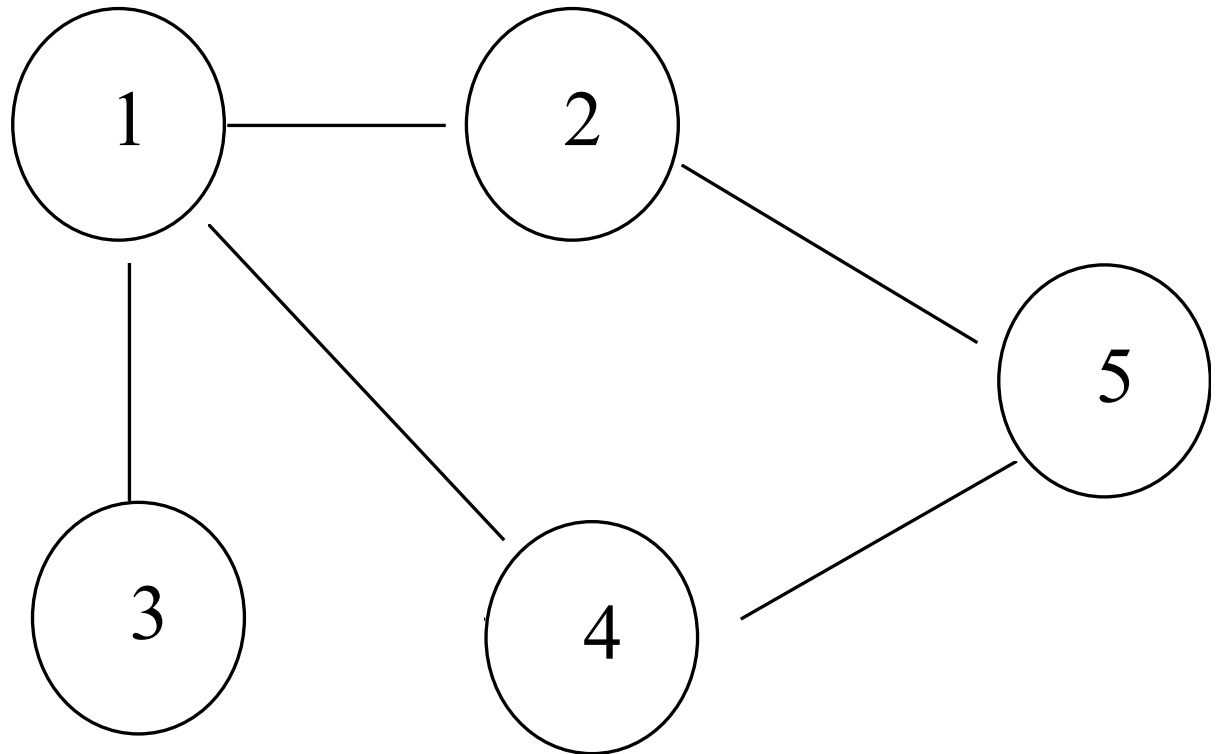
- A sequence of vertices where each successive pair are connected by an edge



- For example, $(3,4,5)$ is not a path. $(3,1,4,5)$ is a path

Neighbor

- All vertices that share an edge with the node are its neighbors. Denote as $nbhd(X)$



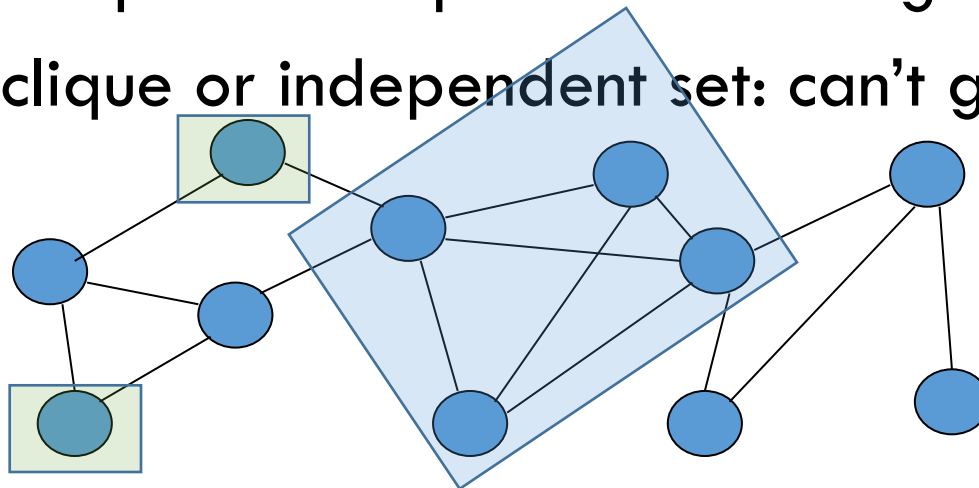
- For example, (3,4,2) are neighbors of 1.
 - $nbhd(1) = (3,4,2)$

Cliques and Independent Sets

- A clique in a graph G is a set of vertices:
 - informal: that are all directly connected to each other
 - formal: whose induced subgraph is complete
 - an edge is a clique of just 2 vertices

Cliques and Independent Sets

- A clique in a graph G is a set of vertices:
 - informal: that are all directly connected to each other
 - formal: whose induced subgraph is complete
 - an edge is a clique of just 2 vertices
- Independent set:
 - set of vertices whose induced subgraph is empty (no edges)
- Maximum clique or independent set: largest in the graph
- Maximal clique or independent set: can't grow any larger

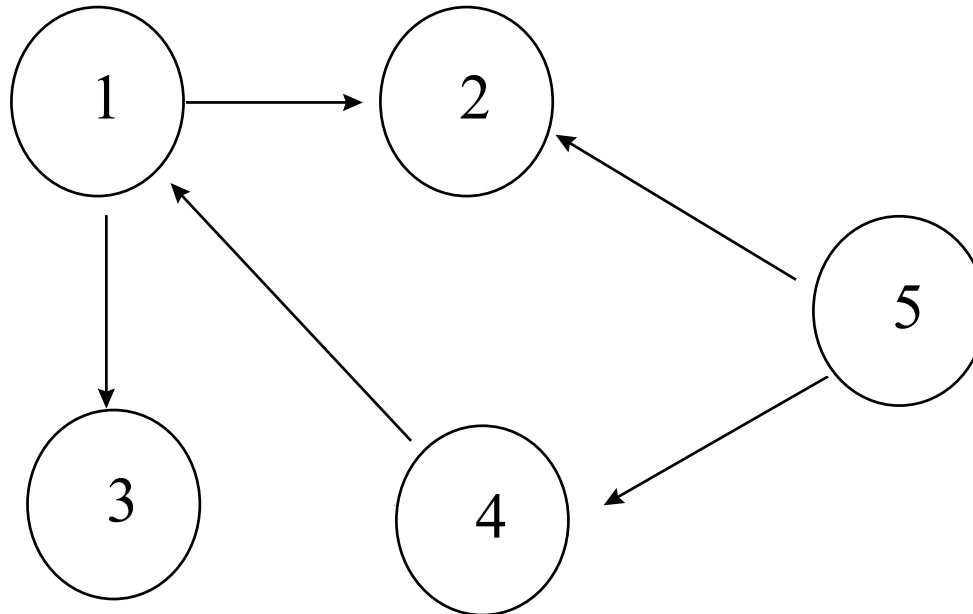


Directed Acyclic Graph

- A directed graph
 - Edges have directions or orientations
 - Edge (u,v) means $u \rightarrow v$
 - May also have edge (v,u)
 - Common for capturing asymmetric relations
- A directed acyclic graph (DAG)
 - No directed cycles
 - No way to follow the oriented edges and come back to the starting node

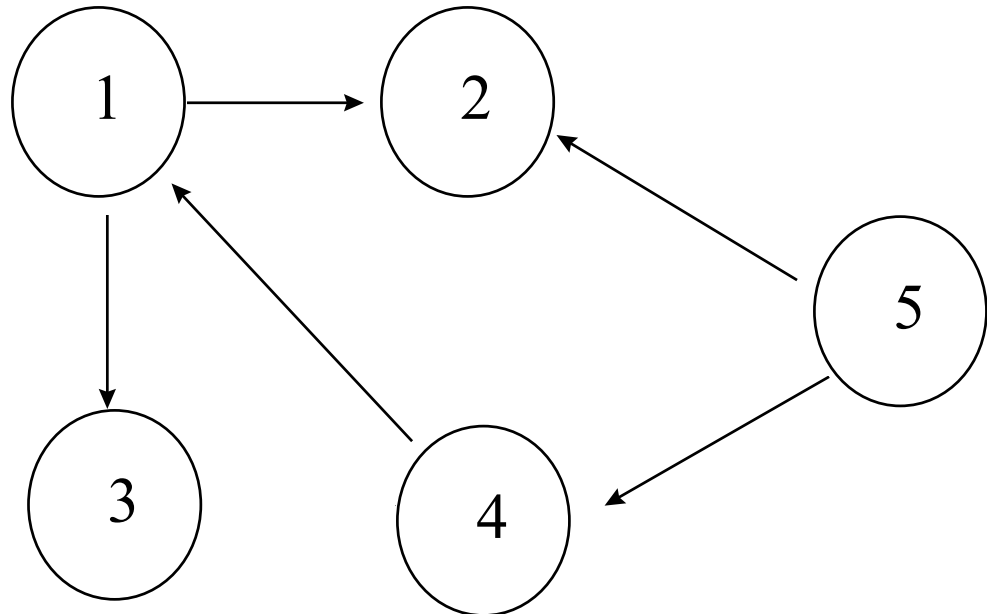
DAG

- A directed acyclic graph (DAG)



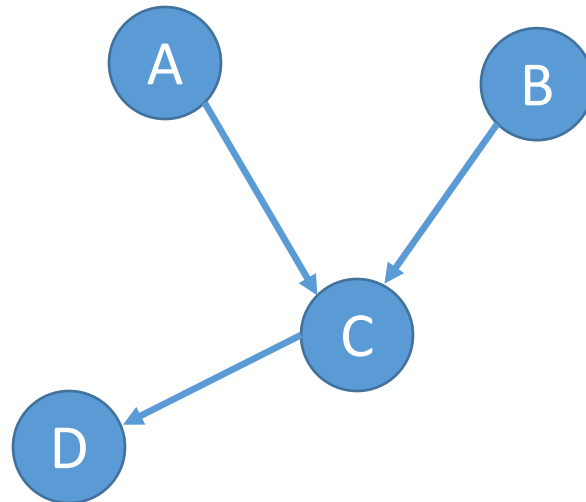
DAG Paths vs Directed Paths

- Path:
 - Same as undirected graph. Ignore directions
 - Example: (3,1,2) is a path
- Directed path
 - Take direction into account. E.g., (5,4,1,3) is a directed path



Parents of a Node

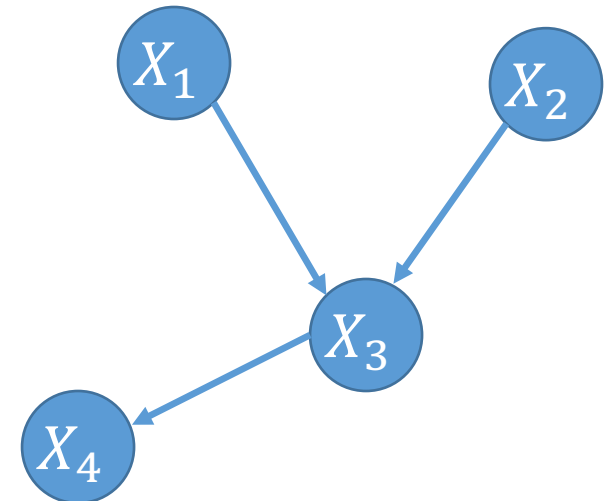
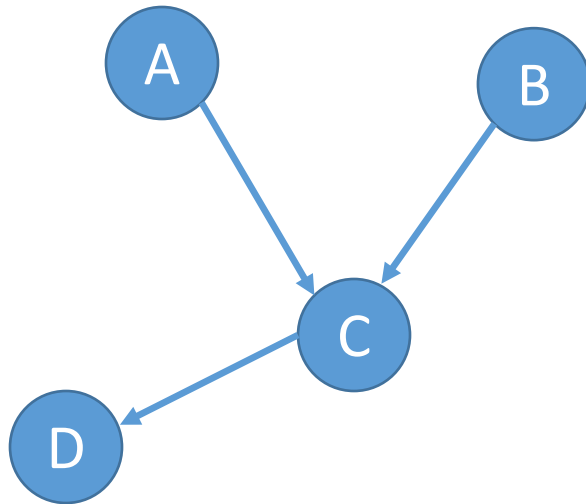
- Notation:
 - $pa(J)$ = Parents of node J



- In this graph, parents of C are (A, B)
 - Neighbors of that vertex that point to that vertex

Parents of a Node

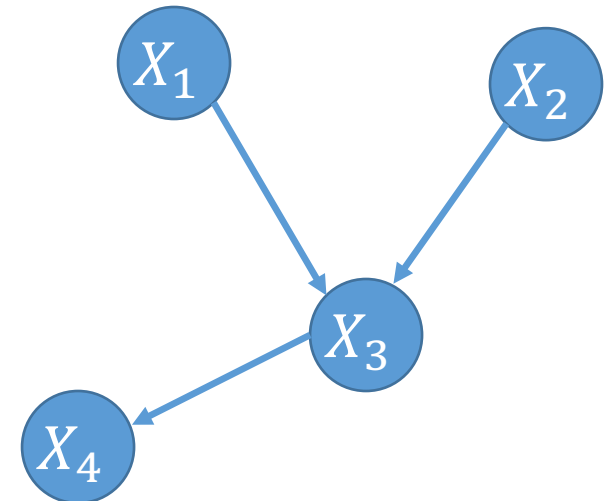
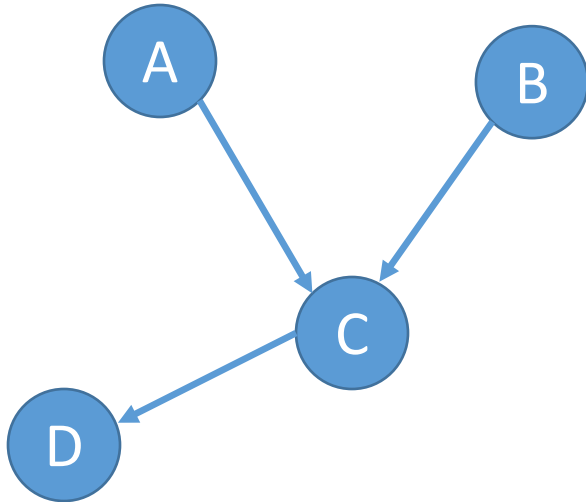
- In the below graph, parents of A is the empty set ϕ



- In the graph on the right, $pa(X_4) = (X_3)$

Descendants of a Node

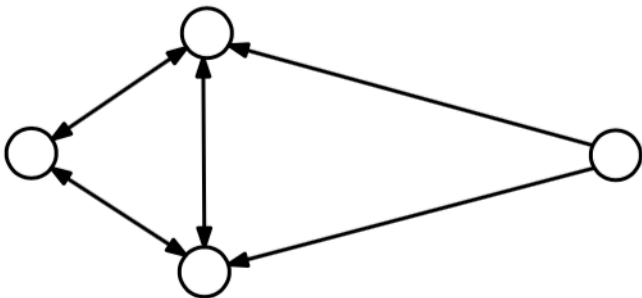
- All nodes that can be reached by following the arrow directions
- In the below graph, descendants of A are $\{C, D\}$



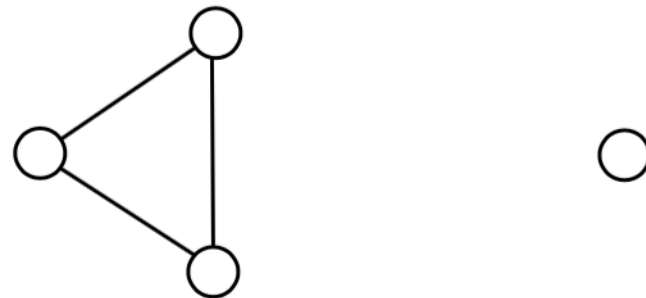
- In the graph on the right, $\text{Desc}(X_3) = (X_4)$

Connected Graphs

- G (directed or undirected) is connected if there is a path between any two vertices.
- Otherwise, we have connected components.
 - subgraphs determined by mutual connectivity
- Connected graph: only one connected component
- Complete graph: edge between all pairs of vertices



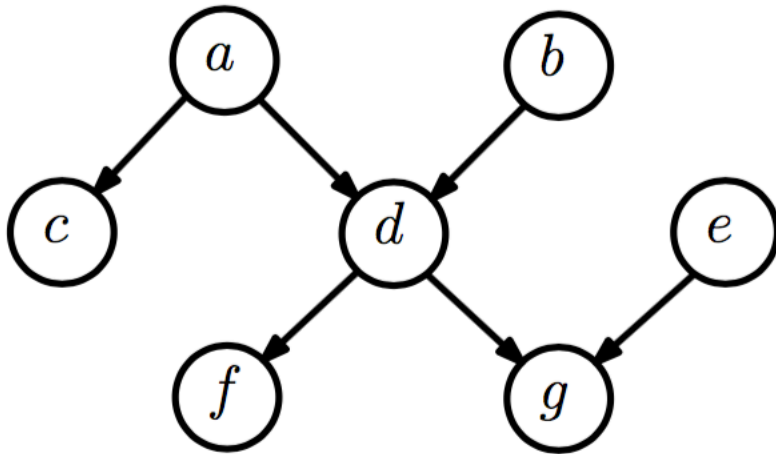
connected graph



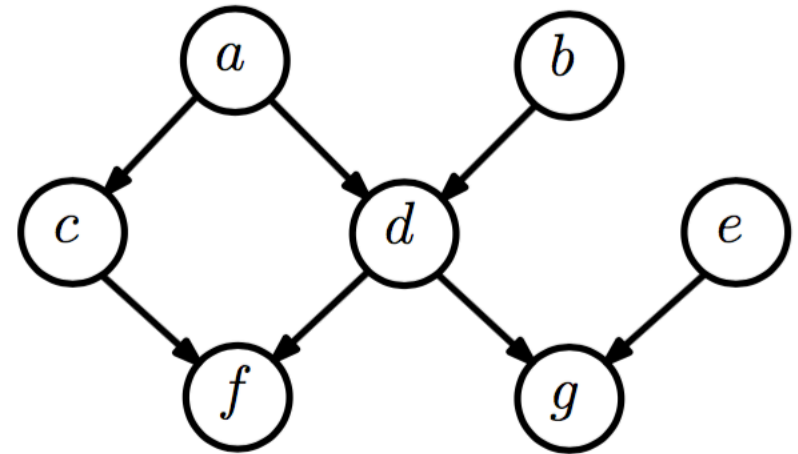
graph with
two connected components³⁵

Singly-Connected

- If for any vertex pair, there is no more than one path between them. This is also called a **tree**.



singly-connected



multiply-connected

- Otherwise, it is multiply-connected. Also called **loopy**.
- Similar definition for undirected graphs as well.

Questions?

Today's Outline

- Motivation
- Primer on Graphs
- Directed Graphical Models
- Undirected Graphical Models

Directed (Probabilistic) Graphical Models

Based on notes by *MathematicalMonk*¹

¹Reference: <https://www.youtube.com/playlist?list=PLD0F06AA0D2E8FFBA>

DPGM

- DPGM: Directed Probabilistic Graphical Model
- Also called a Bayesian Network or Belief Net
 - Nothing Bayesian here
- These are conditional independence diagrams
- So we have directed graphs that tell about conditional independence properties of a probability distribution

Factorization

- Key idea:
 - Factorization
 - The diagrams tell how the joint distribution factors
- Graphs
 - Will be used as a notational gadget
 - For visualizing
 - Conditional independence properties
 - Inference algorithms
 - Dynamic programming, Markov Chain Monte Carlo

Tractable Inference Objective

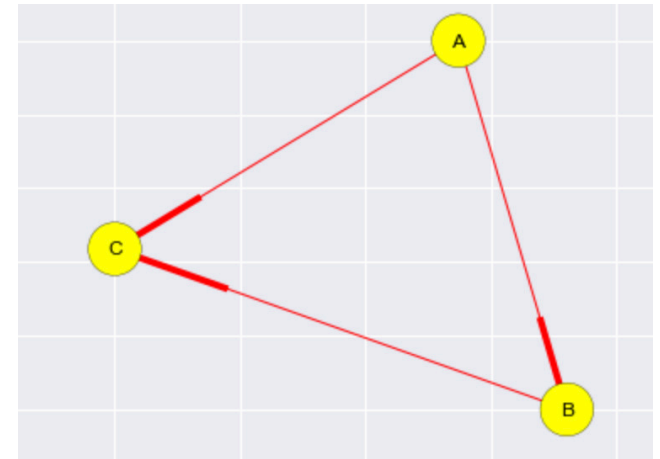
- Why do we care about conditional independence?
- Because we can perform tractable or efficient inference (we will address this next lecture!)
- Lets associate a graph with a probability distribution

Joint Distribution

- Let A, B, C be RVs
- Joint distribution
 - $P(A = a, B = b, C = c)$
 - $= P(c|a, b)P(a, b)$
 - $= P(c|a, b)P(b|a)P(a)$
 - This is a factorization
 - We can always do this

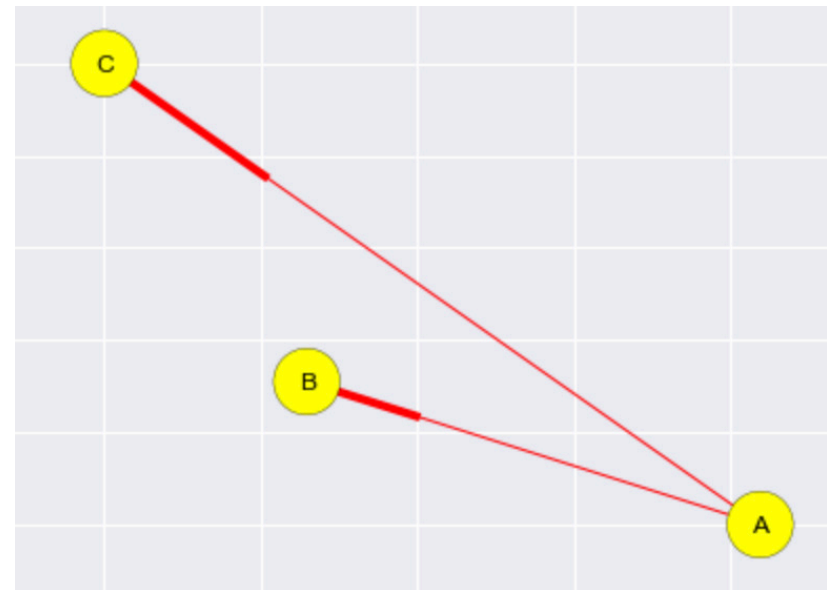
Non-unique Factorizations

- $P(c|a, b)P(b|a)P(a)$
- Create a node for each factor
- Graph has directed edges
- No cycles
 - Can't return to a node
- Nothing special about this factorization
 - We could have factored in a completely different way



Distribution to a DAG

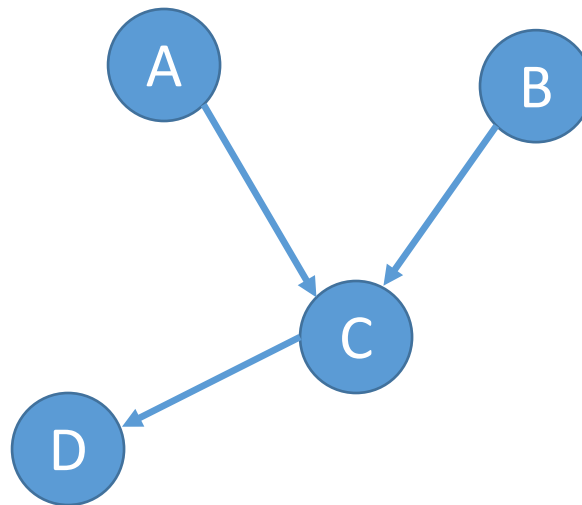
- If C is conditionally independent of B given A
- Use notation $C \perp B \mid A$
- Then $P(c|a, b) = P(c|a)$
- So we got a different graph



- Not every distribution could have lead to this graph.

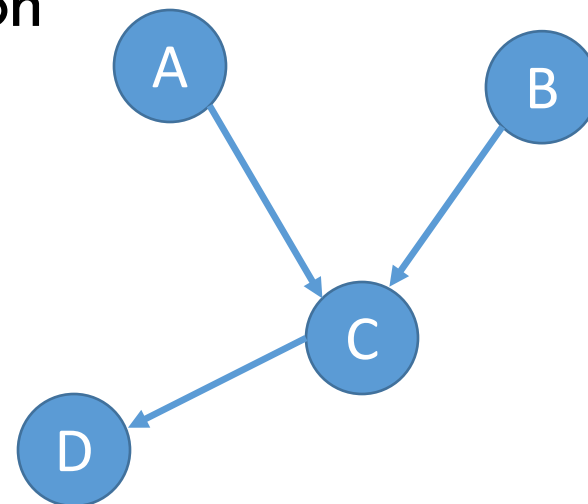
DAG to a Distribution

- What if we start with a graph?
- Can we construct a probability distribution?
- This is a DAG



DAG to a Distribution

- Write one term per node
- $P(a, b, c, d) = P(d|c)P(c|a, b)P(a)P(b)$
- This may/may not be a distribution
- This is a product of factors
- One factor per node



Non-unique Graphs

- Given $X = (X_1, \dots, X_n) \sim P$, and a DAG G
- We say X respects G (or P respects G) if
 - $P(x_1, \dots, x_n) = \prod P(x_i | pa(x_i)) \quad \forall i = 1, \dots, n$
- Formalizes the ideas in the examples we saw earlier

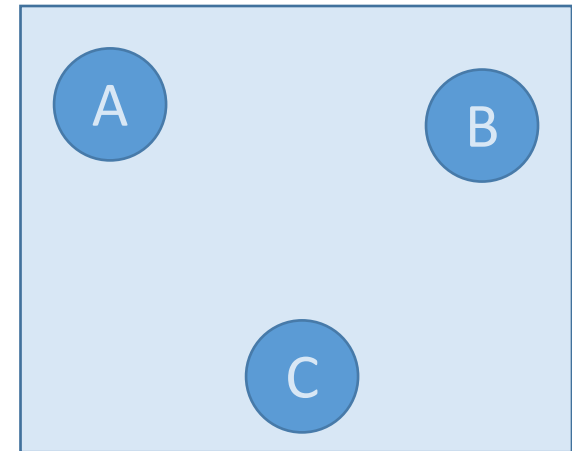
Non-unique Graphs

- Given $X = (X_1, \dots, X_n) \sim P$, and a DAG G
- We say X respects G (or P respects G) if
 - $P(x_1, \dots, x_n) = \prod P(x_i | pa(x_i)) \quad \forall i = 1, \dots, n$
- Formalizes the ideas in the examples we saw earlier
- The graph G does not imply that any RVs are conditionally dependent.
 - At most, it will imply is conditional independence
- The graph G does not uniquely determine the probability distribution P

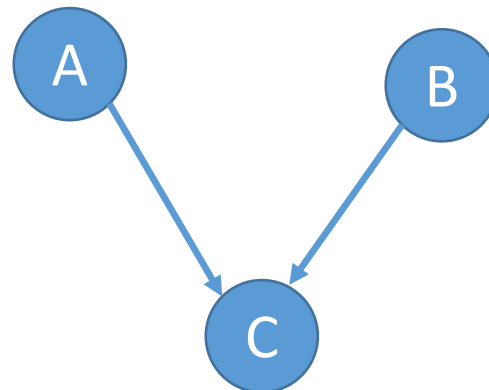
Non-unique Graphs

- Say A,B,C are independent
 - $P(a,b,c) = P(a)P(b)P(c)$

- Let $X = (A,B,C)$
- Then X respects the graph G

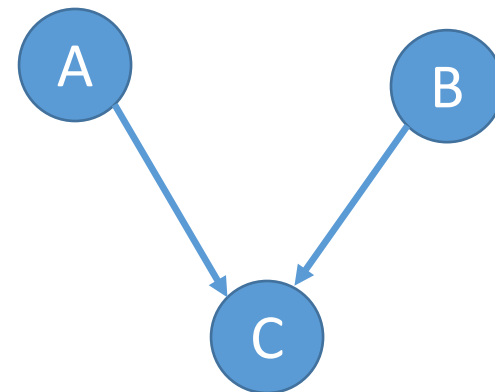


- X also respects G'

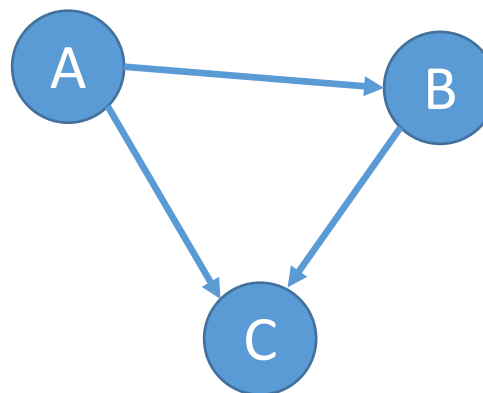


Non-unique Graphs

- Graph G' is not saying C depends on A and B



- It only says that the distribution of $X = (A, B, C)$ factors in a way that can be represented by G'
- X also respects G''

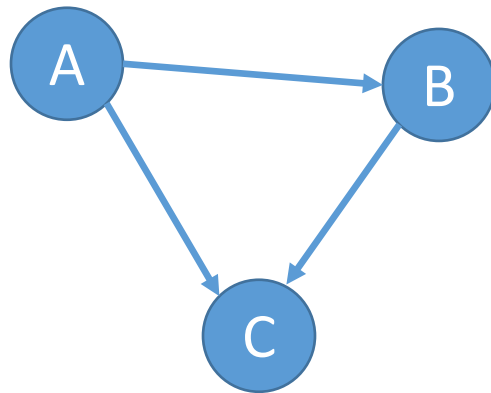


Aside: Complete DAG

- A complete directed acyclic graph is a graph where all vertices are connected to all (permissible) others
- Any distribution on n variables respects a complete directed acyclic graph with n nodes

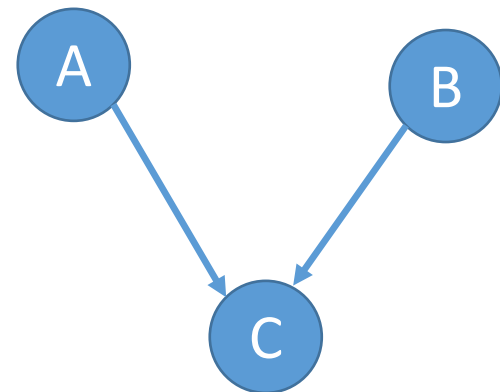
Aside: Variables to Vectors

- We can combine RVs into random vectors and nothing discussed so far changes.
- For example, A, B, C can be random vectors



Advantage of Graphs

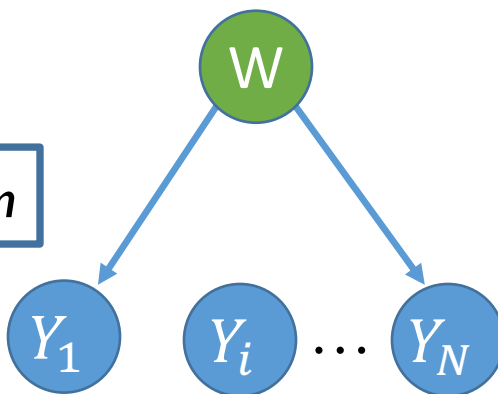
- Key fact: if factors are normalized conditional distributions, then the joint distribution is a valid distribution
- Summary till now:
 - Connected a graph to a joint distribution
- Two (of many) advantages of DPGMs:
 - Visualization
 - Sampling (we will see this later)
 - We can sample sequentially according to the graph



Example: Linear Regression

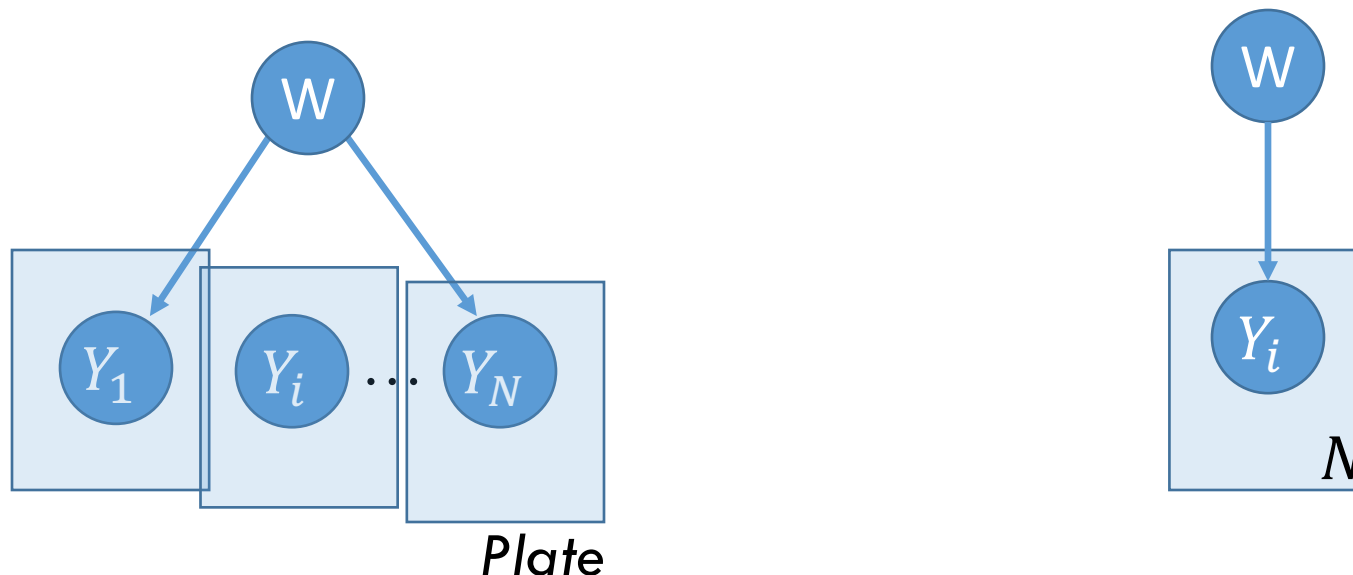
- Graphical model for (Bayesian) linear regression
 - Data: $\{x_i, y_i\}_{i=1}^N$ where x_i is d dimensional
 - Model: $f_W(x) = W^T \phi(x)$
 - Linear in W (not a matrix, a random vector)
- Let $W \sim N(0, \sigma_0^2 I)$
- Let $Y_i \sim N(W^T \phi(x_i), \sigma^2)$
 - Let Y_i be conditionally independent of Y_j given W

σ_0, σ and x_i are not random



Example: Linear Regression

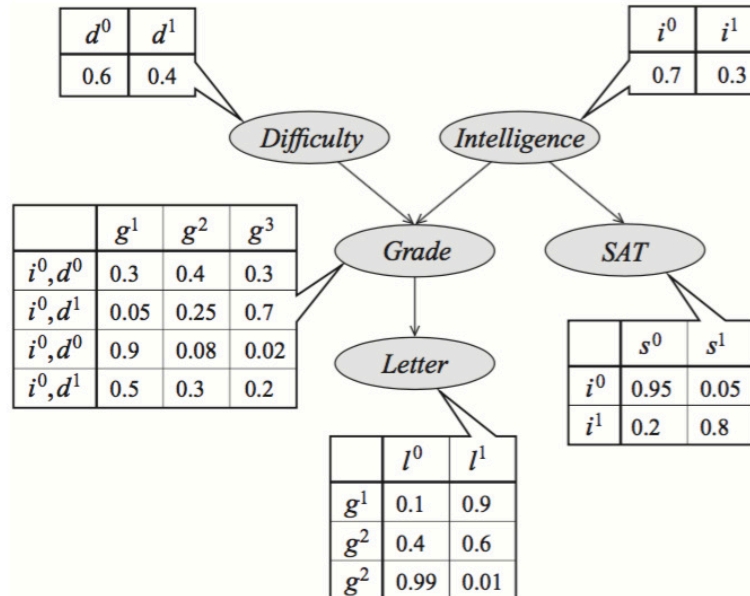
- $P(w, y_1, \dots, y_N) = P(w) \prod P(y_i | w)$
- Can also use a plate notation
 - Stack the plates on top of each other



- Variable W is called a latent or hidden variable
- Variables Y_i are called observed variables

Example: Student Network

- Consider the following Bayesian network:



- What is its joint distribution?

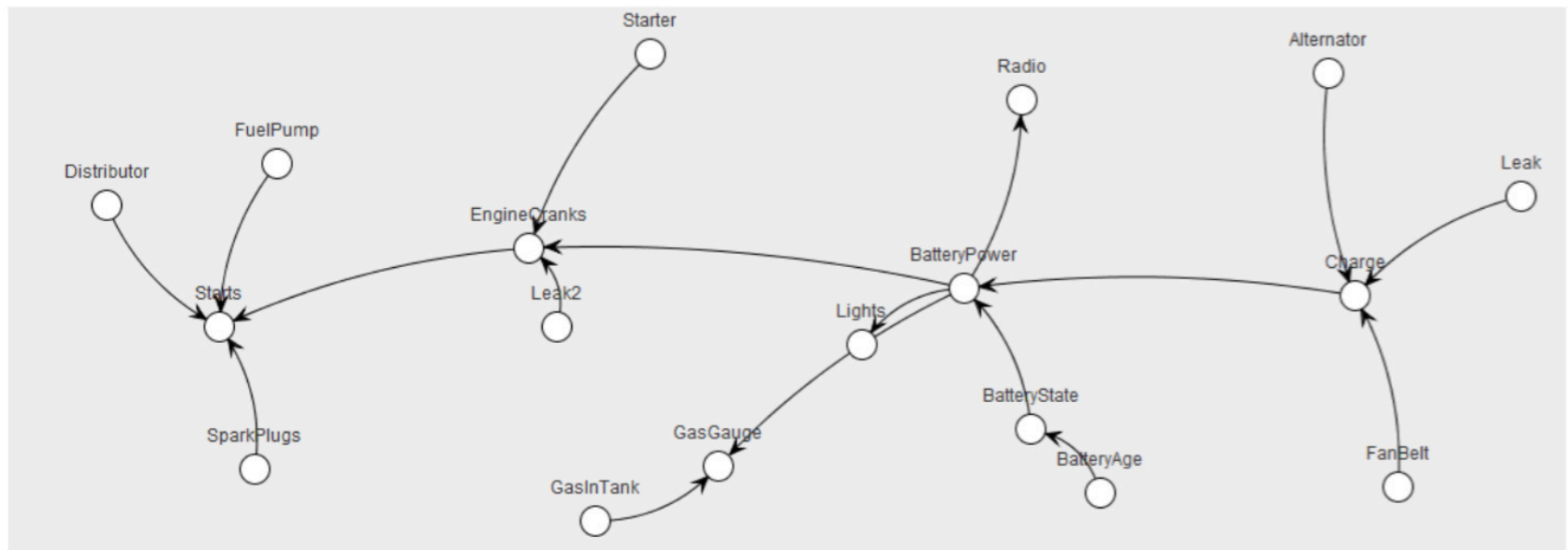
$$p(x_1, \dots, x_n) = \prod_{i \in V} p(x_i \mid \mathbf{x}_{\text{Pa}(i)})$$

$$p(d, i, g, s, l) = p(d)p(i)p(g \mid i, d)p(s \mid i)p(l \mid g)$$

Example: Car Network

$$p(x_1, \dots, x_n) = \prod_{i \in V} p(x_i \mid \mathbf{x}_{\text{Pa}(i)})$$

Will my car start this morning?



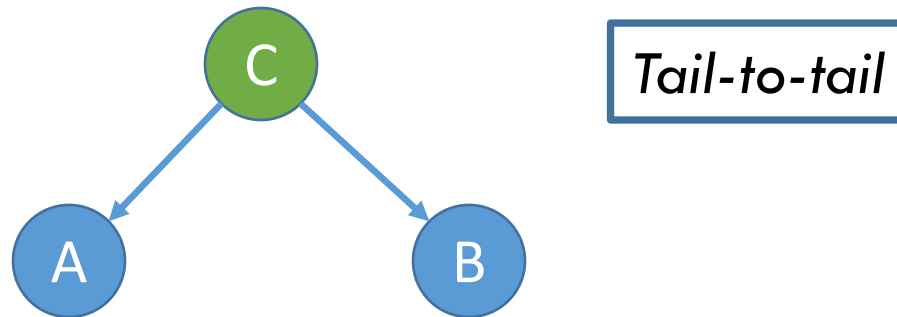
Heckerman *et al.*, Decision-Theoretic Troubleshooting, 1995

Aside: Observed vs Hidden

- Observed variables:
 - Sometimes no need to model their density
 - For example, inputs in regression or classification
 - This leads to conditional density estimation
- Unobserved variables:
 - Called hidden or latent
 - Can be marginalized out
 - Can make the density modeling of observed variables easier

Conditional Independence (I)

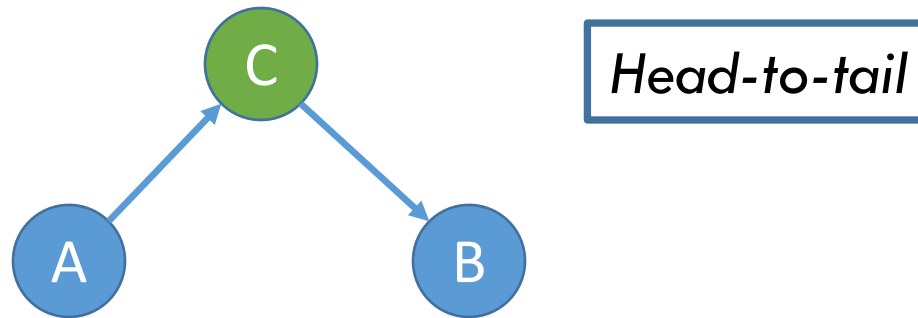
- Given a graphical model, we can determine if two sets of RVs are conditionally independent or not



- $P(a, b, c) = P(a|c)P(b|c)P(c)$ is the joint distribution that respects this graph
- What happens when we condition on C?
 - $P(a, b|c) = \frac{P(a, b, c)}{P(c)} = P(a|c)P(b|c)$
 - Thus, A and B are conditionally independent given C
 - Use notation $A \perp B \mid C$

Conditional Independence (II)

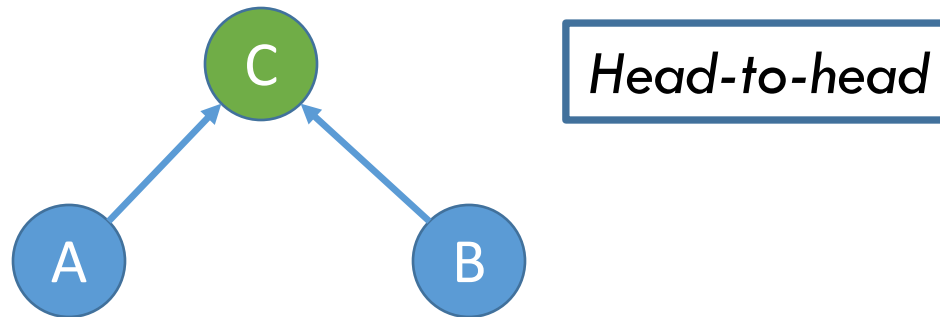
- Given a graphical model, we can determine if two sets of RVs are conditionally independent or not



- $P(a, b, c) = P(a)P(c|a)P(b|c) = [P(a|c)P(c)]P(b|c)$ is the joint distribution that respects this graph
- What happens when we condition on C?
 - $P(a, b|c) = \frac{P(a, b, c)}{P(c)} = P(a|c)P(b|c)$
 - Thus, A and B are conditionally independent given C
 - Use notation $A \perp B \mid C$

Conditional Independence (III)

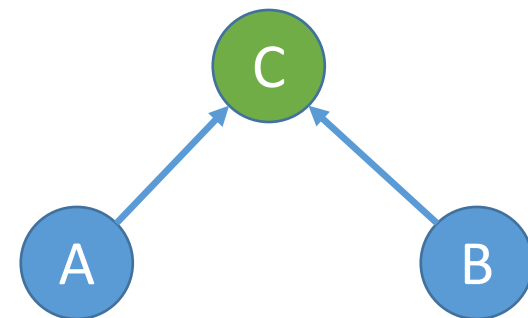
- Given a graphical model, we can determine if two sets of RVs are conditionally independent or not



- $P(a, b, c) = P(a)P(b)P(c|a, b)$ is the joint distribution that respects this graph
- What happens when we condition on C?
 - $P(a, b|c) = \frac{P(a, b, c)}{P(c)} \neq P(a|c)P(b|c)$
 - A and B are **not** conditionally independent given C ⁶²

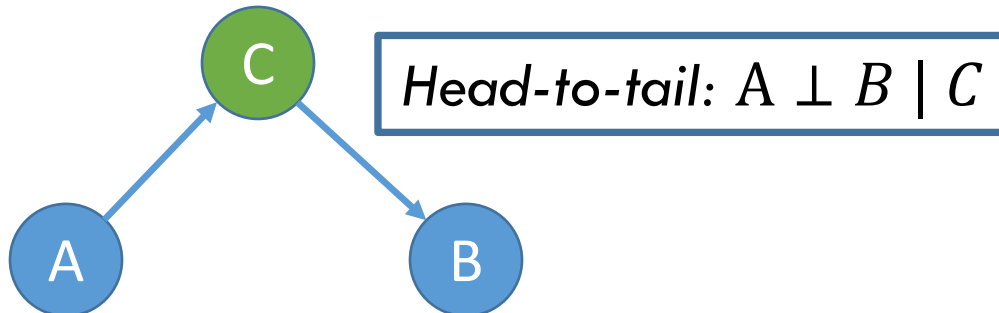
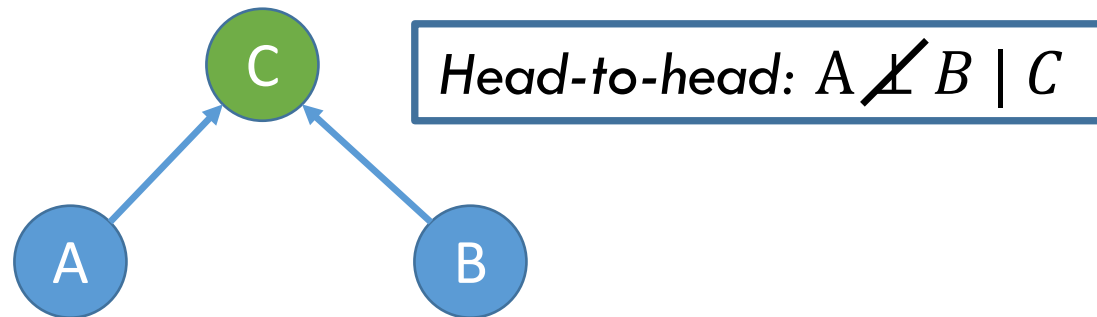
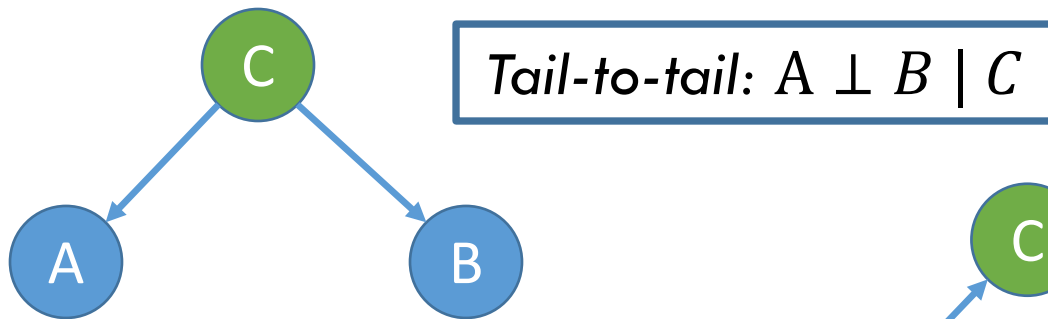
Head-to-Head Example

- Say $A \sim \text{Bern}\left(\frac{1}{2}\right), B \sim \text{Bern}\left(\frac{1}{2}\right)$
- Say $C = 1$ if $A = B$ and 0 otherwise
- Conditioned on C
 - If we know A, we know B.
 - They are dependent!
 - Similarly, if we know B, we know A.
- Hence, $A \not\perp B \mid C$ (i.e., not true for every distribution that respects the graph)
- But unconditionally, $A \perp B$
 - $P(a, b) = \sum_c P(a, b, c) = \sum_c P(a)P(b)P(c|a, b)$
 - $= P(a)P(b) \sum_c P(c|a, b) = P(a)P(b)$



Conditional Independence: Summary of 3 Node Setting

- Given a graphical model, we can determine if two sets of RVs are conditionally independent or not



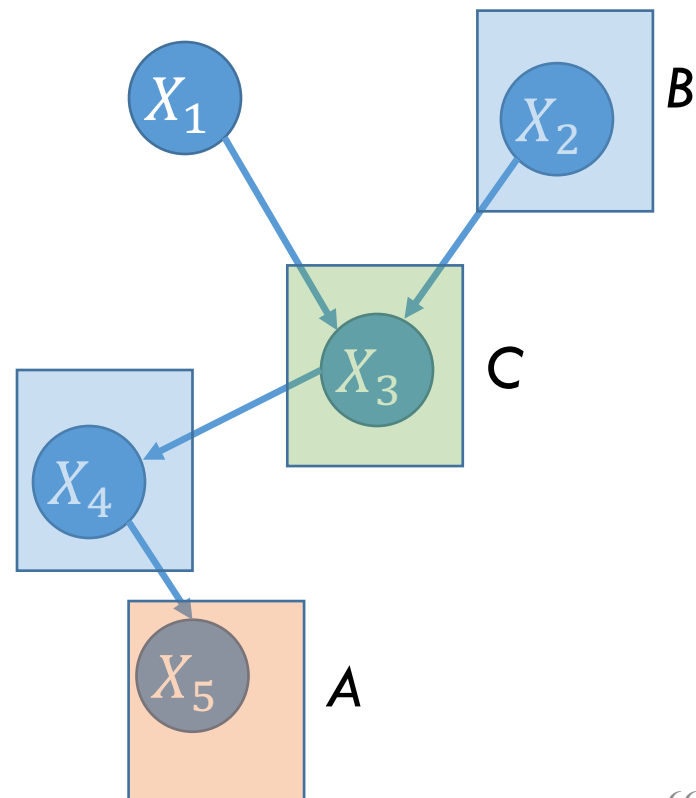
D-Separation Criterion: N Node Setting

- We saw how conditional independence properties unfold due to graph structure
 - This was only for three node graphs
- We will now move to larger DAGs
- We will look at the idea of d-separation

D-Separation (I)

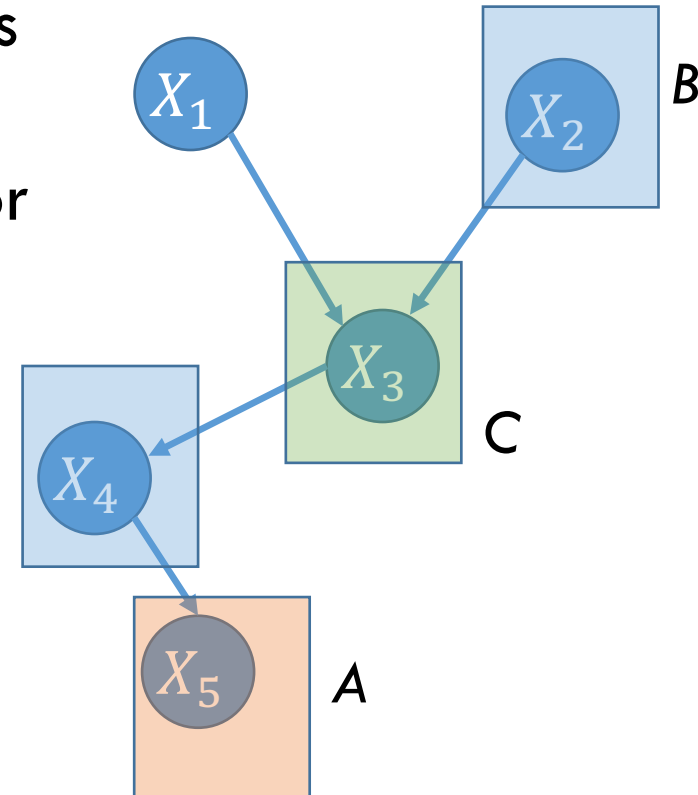
- Helps you read off the conditional independence properties

- Notation
 - Sets of RVs A, B and C
 - Disjoint
 - Not necessarily covering all



D-Separation (II)

- A path between two vertices is **blocked** with respect to \mathcal{C} if it passes through a node v such that
 - $v \in \mathcal{C}$, arrows are head-to-tail or tail-to-tail
 - OR, $v \notin \mathcal{C}$, arrows are head-to-head, and $\text{Descendants}(v) \notin \mathcal{C}$
- Example
 - X_4, X_3 and X_2 are in head-tail
 - So path is blocked

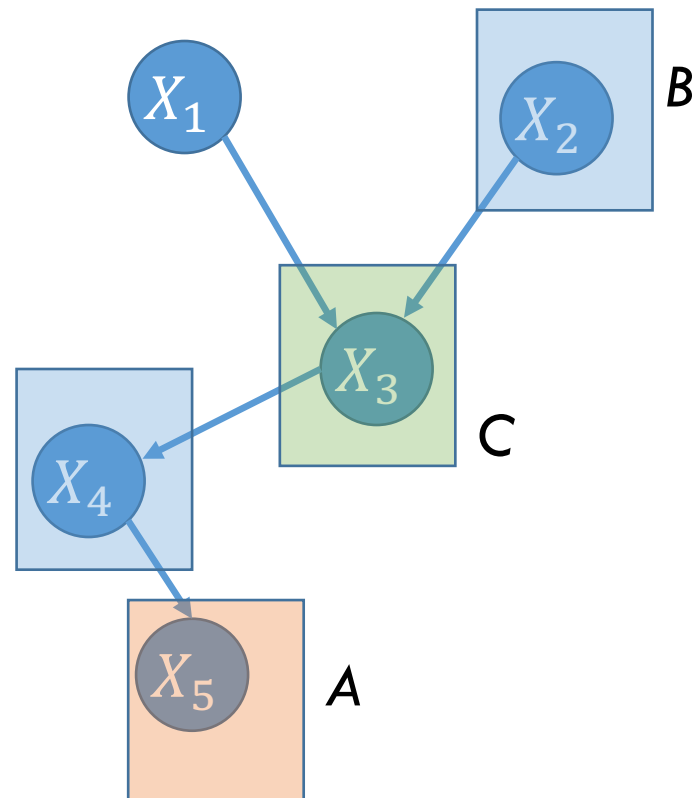


D-Separation (III)

- Definition of D-Separation
 - A and B are d-separated by C if all paths from vertices in A to vertices in B are blocked with respect to C
- Key result
 - If A and B are d-separated by C, then $A \perp B \mid C$
- Note: the above result is only 'necessary' not 'sufficient'

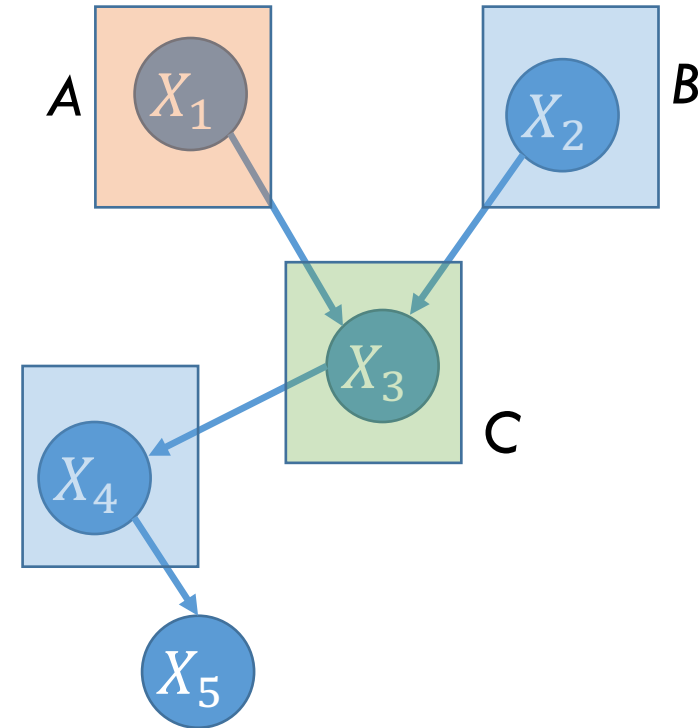
D-Separation Example I

- Let $C = \{X_3\}$
- Is $A \perp B \mid C$?
- We can check that by checking d-separation for all pairs of vertices $X_i \perp X_j \mid C$?
 - $i = \{5\}$
 - $j = \{2,4\}$
- Easy to see that
 - X_2, X_5 are blocked by C
 - X_4, X_5 are not blocked by C
- Hence, not d-separated
- Hence cannot say $A \perp B \mid C$



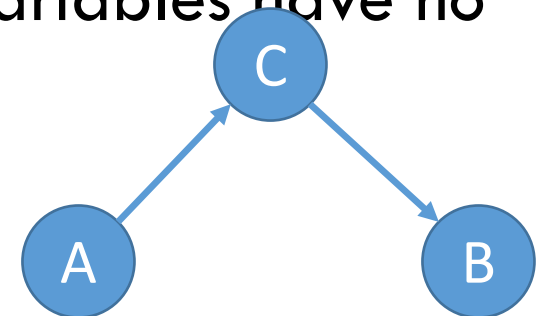
D-Separation Example II

- Let $C = \{X_3\}$
- Is $A \perp B \mid C$?
- We can check that by checking d-separation for all pairs of vertices $X_i \perp X_j \mid C$?
 - $i = \{1\}$
 - $j = \{2,4\}$
- We can see that
 - X_1, X_2 are not blocked by C
 - X_1, X_4 are blocked by C
- Hence, not d-separated
- Hence cannot say $A \perp B \mid C$



DAG and Probability (I)

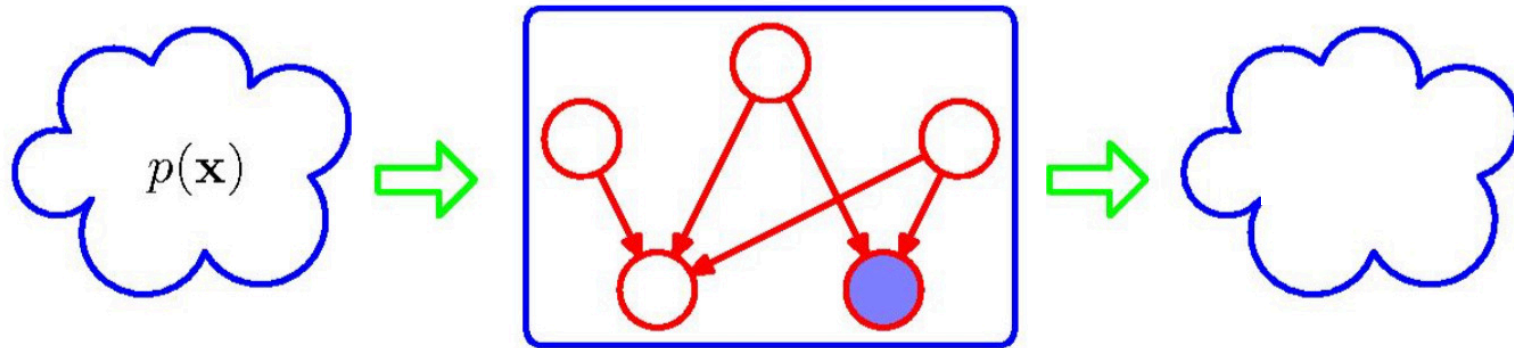
- We have showed that the structure of the DAG corresponds to a set of conditional independence assumptions
 - We can read conditional independence easily!
- We just need to specify $P(X_i|pa(X_i))$
- This does not mean that non-parent variables have no influence
 - Thus, the DAG does not imply
 - $P(c|a, b) = P(c|a)$



DAG and Probability (II)

- DPGMs are good for representing independence, not for representing dependence
- We have seen this
 - Multiple graphs for the same distribution
 - D-separation only says conditional independence if true. If not true, then no conclusion is drawn.

Filter view of DPGM



- Only distributions that satisfy conditional independences are allowed to pass
- One graph can describe many probability distributions
- Edge cases:
 - When DAG is fully connected, all distributions pass
 - When DAG is fully disconnected, only the product distribution ($\prod_i P(X_i)$) passes

Continuous Distributions

- We never had to state whether $P(X|Y)$ was continuous or discrete
- The graph is agonistic to the support of the random variables!

Questions?

Today's Outline

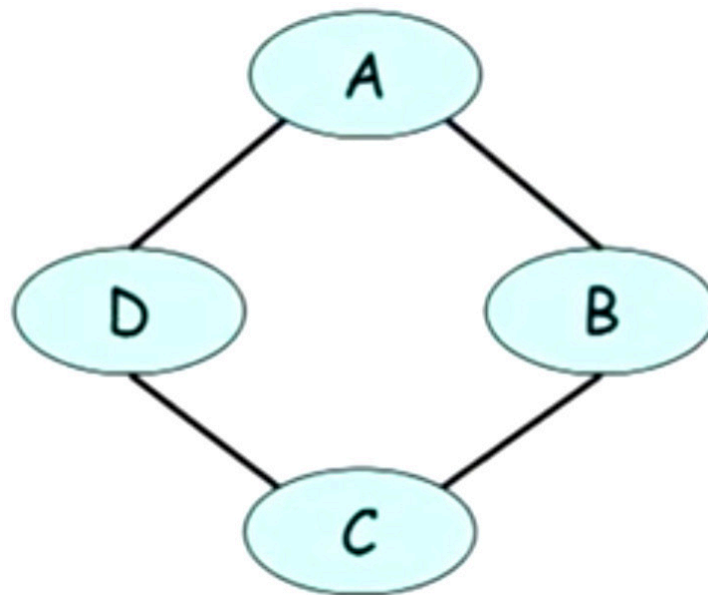
- Motivation
- Primer on Graphs
- Directed Graphical Models
- Undirected Graphical Models

Undirected (Probabilistic) Graphical Models

Based on notes from Bjoern Andres and Bernt Schiele (2016)

UPGM

- Also called Markov Networks or Markov Random Fields
- No edge directions
- Again, diagrams of probability distributions that capture conditional independences

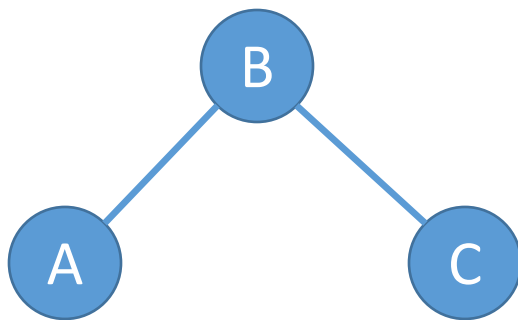


UPGM vs DPGM

- DPGMs are more used in data analytics, ML, statistics
- UPGMs have been used in computer vision and physics but have applications in data analytics as well
- DPGM
 - Factor of the distribution was a (cond.) distribution
- UPGM
 - Factor (also called **potential**) need not be a distribution
 - Let $P(a, b, c) = \frac{1}{Z} \phi_1(a, b) \phi_2(b, c)$
 - Here Z is the normalization constant or **partition function**.
 $Z = \sum_{a,b,c} \phi_1(a, b) \phi_2(b, c)$

Notion of a Potential

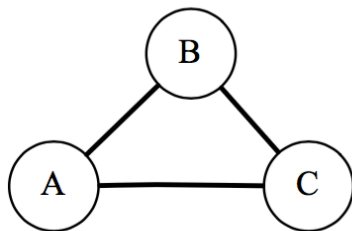
- Potential $\phi(x)$ is a non-negative function of variable x . Joint potential $\phi(x_1, \dots, x_D)$ is a non-negative function of a set of variables.
- Let $P(a, b, c) = \frac{1}{Z} \phi_1(a, b) \phi_2(b, c)$



Potentials Over Cliques

- For RVs X_1, \dots, X_D , an UPGM is defined as a product of potentials over the **cliques** of graph G
- $P(X_1, \dots, X_D) = \frac{1}{Z} \prod_c \phi_c(\mathcal{X}_c)$
 - Here $Z = \sum_{x_1, \dots, x_D} \prod_c \phi_c(\{x_i: X_i \in \mathcal{X}_c\})$
- Special cases:
 - When cliques are of size 2: the UPGM is called a pairwise UPGM
 - When all potentials are strictly positive: the distribution is called a Gibbs distribution

Example Potentials



		$\phi_{A,B}(a,b)$		$\phi_{B,C}(b,c)$		$\phi_{A,C}(a,c)$	
		B		C		C	
A	0	10	1	B	0	10	1
	1	1	10		1	1	10

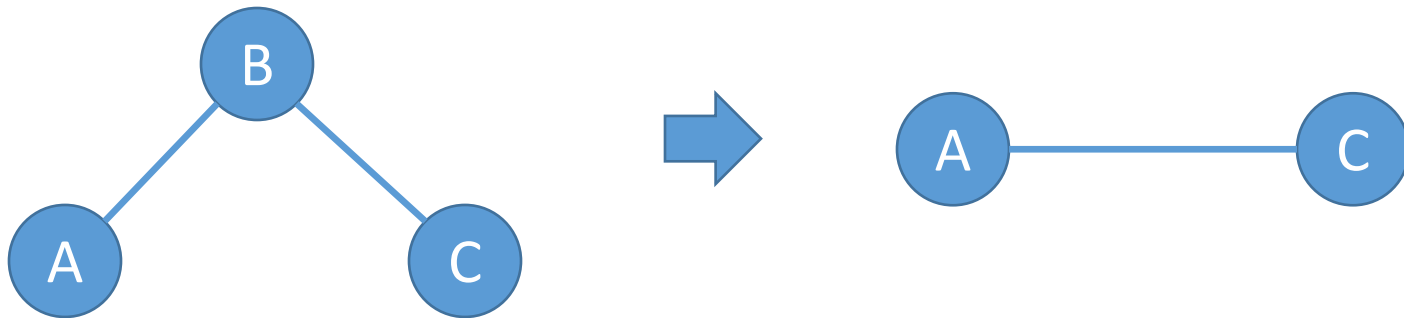
$$p(a, b, c) = \frac{1}{Z} \phi_{A,B}(a, b) \cdot \phi_{B,C}(b, c) \cdot \phi_{A,C}(a, c),$$

where

$$Z = \sum_{\hat{a}, \hat{b}, \hat{c} \in \{0,1\}^3} \phi_{A,B}(\hat{a}, \hat{b}) \cdot \phi_{B,C}(\hat{b}, \hat{c}) \cdot \phi_{A,C}(\hat{a}, \hat{c}) = 2 \cdot 1000 + 6 \cdot 10 = 2060.$$

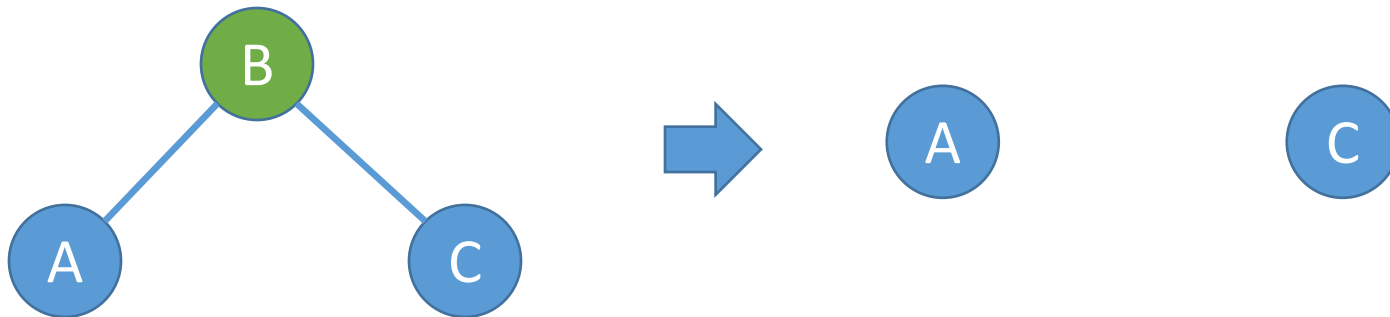
Marginalization

- Marginalizing over B makes A and C graphically dependent
- $P(a, c) = \sum_b P(a, b, c) = \frac{1}{Z} \phi_3(a, c)$



Conditional Independence (I)

- Conditioning on B makes A and C independent
- $P(a, c|b) = P(a|b)P(c|b)$



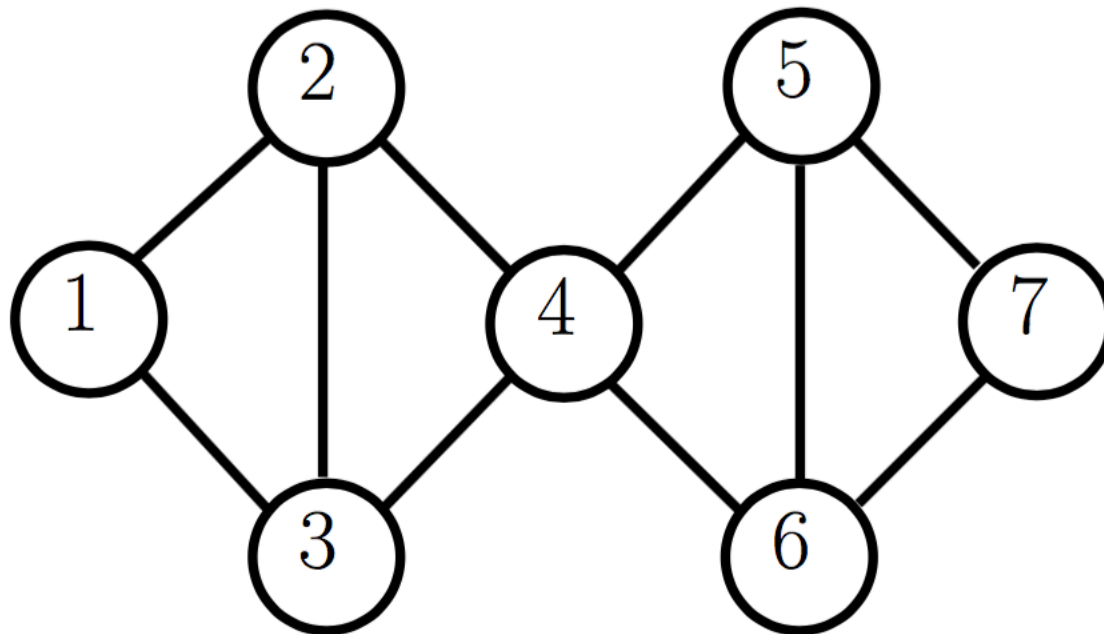
- Key: This is different from the head-to-head directed graph example, where conditioning introduced dependency!

Conditional Independence (II)

- Global Markov property
 - Two sets of nodes (say A and B) are conditionally independent given a third set C if
 - All nodes in A and B are connected through nodes in C
- Local Markov property
 - Conditioning on the neighbors of X makes X independent of the rest of the graph.
 - $P(X_i | X_1, \dots, X_{i-1}, X_{i+1}, \dots, X_D) = P(X_i | nbhd(X_i))$

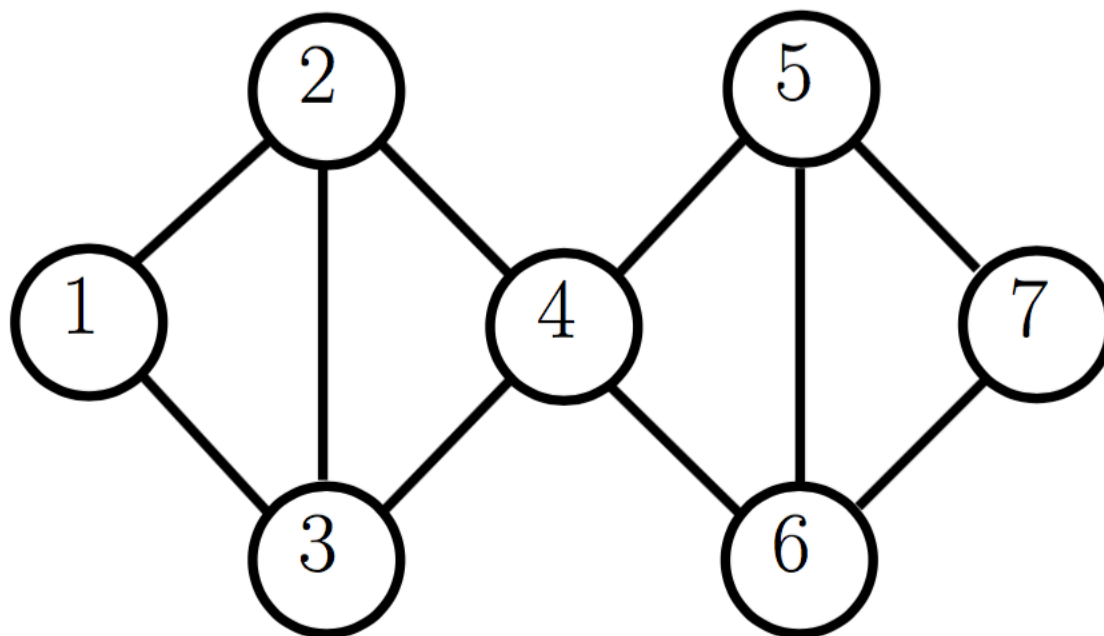
Global Markov Property

- In the following graph G , as a consequence of global Markov property:
 - $\{X_1, X_2, X_3\} \perp \{X_5, X_6, X_7\} | X_4$



Local Markov Property

- In the following graph G , as a consequence of local Markov property:
 - $X_4 \perp \{X_1, X_7\} | \{X_2, X_3, X_5, X_6\}$
 - $X_1 \perp \{X_4, X_5, X_6, X_7\}$

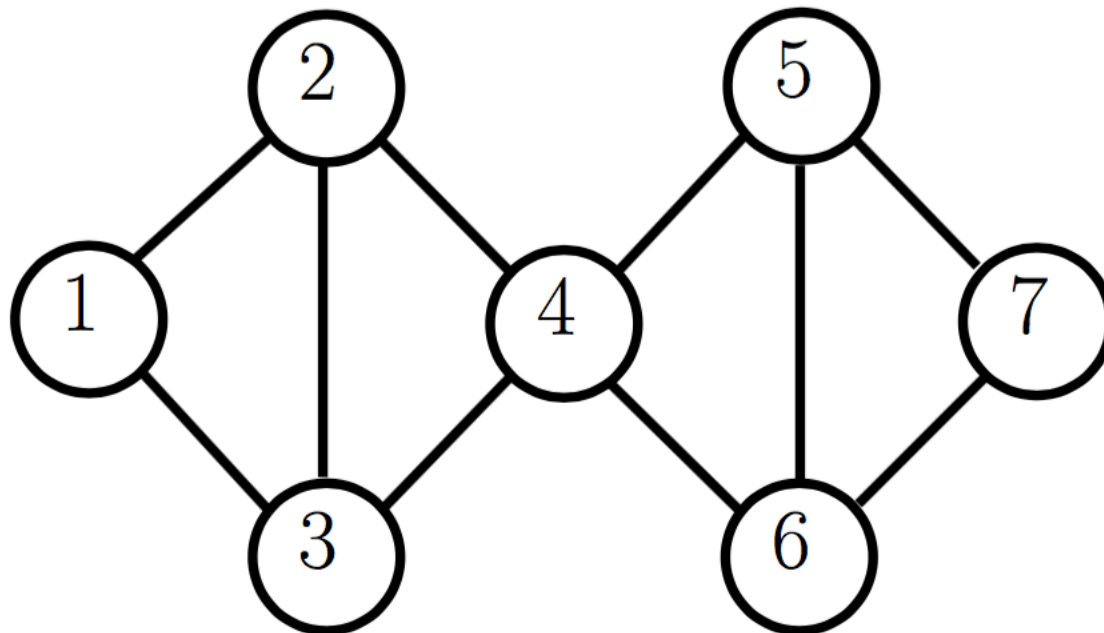


Graph to Distribution

- So, clearly the undirected graph specifies a set of conditional independence statements
- We can write down a joint distribution using the graph
- For example, we may consider a factorization involving maximal cliques.

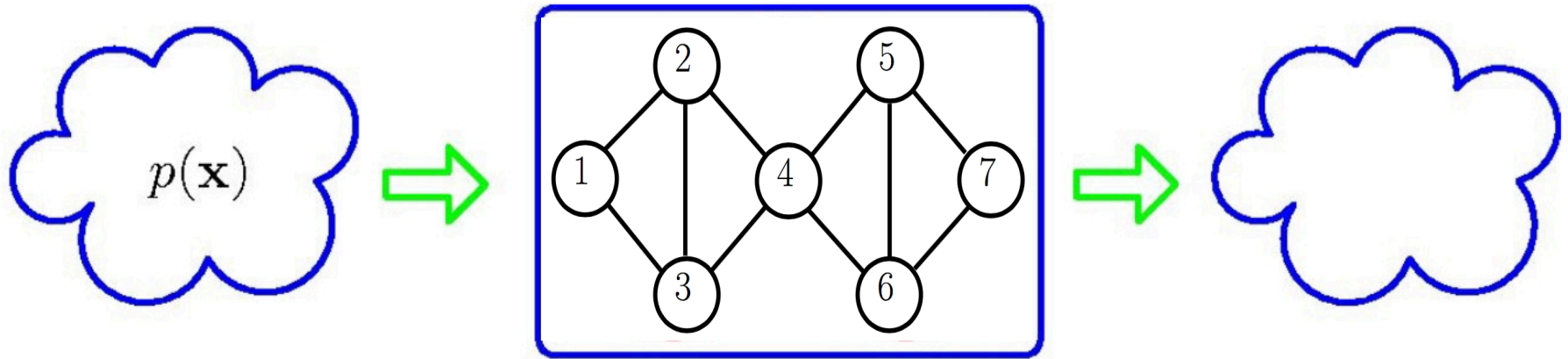
Graph to Distribution

- $P(x_1, \dots, x_7) = \frac{1}{Z} \phi_1(x_1, x_2, x_3) \phi_2(x_2, x_3, x_4) \phi_3(x_4, x_5, x_6) \phi_4(x_5, x_6, x_7)$



- But, we could have also considered some other factorization

Filter view of UPGM



- Only distributions that satisfy conditional independences are allowed to pass

Limitations of DPGM and UPGM

- Cannot always represent all conditional independences of a given joint distribution
- Example: we cannot draw a DPGM for the following distribution
 - $P(A, B, C, D)$ with $A \perp C | \{B, D\}$ and $B \perp D | \{A, C\}$
- Another example: we cannot represent the following using a UPGM
 - $P(A, B, C)$ with $A \not\perp C | \{B\}$ and $A \perp C$
- You should verify this yourself

DPGM vs UPGM

Property	UPGMs	DPGMs
Form	Prod. potentials	Prod. potentials
Potentials	Arbitrary	Cond. probabilities
Cycles	Allowed	Forbidden
Partition func.	$Z = ?$	$Z = 1$
Indep. check	Graph separation	D-separation
Indep. props.	Some	Some
Inference	MCMC, BP, etc.	Convert to UPGM

Questions?

Summary

- What are graphical models good at?
 - Capture complexity and uncertainty
 - Capture conditional independences
 - We can visualize what's happening with a distribution
- They unify many probabilistic techniques: mixture models, factor analysis, hidden Markov models, Kalman filters etc.
- Today we saw: visualization, conditional independence properties
- Next: computations (**inference** and **learning**)

Appendix

Additional Resources

- Book 1: *Graphical models, exponential families, and variational inference* by Martin J. Wainwright and Michael I. Jordan
 - See https://people.eecs.berkeley.edu/~wainwrig/Papers/WaiJor08_FTM.pdf
- Book 2: *Bayesian Reasoning and Machine Learning* by David Barber
 - See <http://web4.cs.ucl.ac.uk/staff/D.Barber/pmwiki/pmwiki.php?n=Brml.Online>

Review: Probability

Based on Sam Roweis's slides (2002)

Probability

- We use probabilities $p(x)$ to represent our beliefs $B(x)$ about the states x of the world.
- There is a formal calculus for manipulating uncertainties represented by probabilities.

Probability

- We use probabilities $p(x)$ to represent our beliefs $B(x)$ about the states x of the world.
- There is a formal calculus for manipulating uncertainties represented by probabilities.
- Any consistent set of beliefs obeying the *Cox Axioms* can be mapped into probabilities.
 1. Rationally ordered degrees of belief:
if $B(x) > B(y)$ and $B(y) > B(z)$ then $B(x) > B(z)$
 2. Belief in x and its negation \bar{x} are related: $B(x) = f[B(\bar{x})]$
 3. Belief in conjunction depends only on conditionals:
 $B(x \text{ and } y) = g[B(x), B(y|x)] = g[B(y), B(x|y)]$

Probability

- An **outcome space** specifies the possible outcomes that we would like to reason about, e.g.

$$\Omega = \{ \text{, \text{} \} \quad \text{Coin toss}$$

$$\Omega = \{ \text{, , , , , } \} \quad \text{Die toss}$$

- We specify a **probability** $p(\omega)$ for each outcome ω such that

$$p(\omega) \geq 0, \quad \sum_{\omega \in \Omega} p(\omega) = 1$$

$$\text{E.g., } p(\text{) = .6$$

$$p(\text{) = .4$$

Probability

- An **event** is a subset of the outcome space, e.g.

$$E = \{ \text{die 1}, \text{die 2}, \text{die 3} \} \quad \text{Even die tosses}$$

$$O = \{ \text{die 4}, \text{die 5}, \text{die 6} \} \quad \text{Odd die tosses}$$

- The **probability** of an event is given by the sum of the probabilities of the outcomes it contains,

$$p(E) = \sum_{\omega \in E} p(\omega)$$

$$\begin{aligned} \text{E.g., } p(E) &= p(\text{die 1}) + p(\text{die 2}) + p(\text{die 3}) \\ &= 1/2, \text{ if fair die} \end{aligned}$$

Random Variables

- Random variables X represents outcomes or states of world.
Instantiations of variables usually in lower case: x
We will write $p(x)$ to mean probability($X = x$).
- Sample Space: the space of all possible outcomes/states.
(May be discrete or continuous or mixed.)

Random Variables

- Random variables X represents outcomes or states of world.
Instantiations of variables usually in lower case: x
We will write $p(x)$ to mean $\text{probability}(X = x)$.
- Sample Space: the space of all possible outcomes/states.
(May be discrete or continuous or mixed.)
- Probability mass (density) function $p(x) \geq 0$
Assigns a non-negative number to each point in sample space.
Sums (integrates) to unity: $\sum_x p(x) = 1$ or $\int_x p(x)dx = 1$.
Intuitively: how often does x occur, how much do we believe in x .
- Ensemble: random variable + sample space + probability function

Expectation

- Expectation of a function $a(x)$ is written $E[a]$ or $\langle a \rangle$

$$E[a] = \langle a \rangle = \sum_x p(x) a(x)$$

e.g. mean = $\sum_x x p(x)$, variance = $\sum_x (x - E[x])^2 p(x)$

Expectation

- Expectation of a function $a(x)$ is written $E[a]$ or $\langle a \rangle$

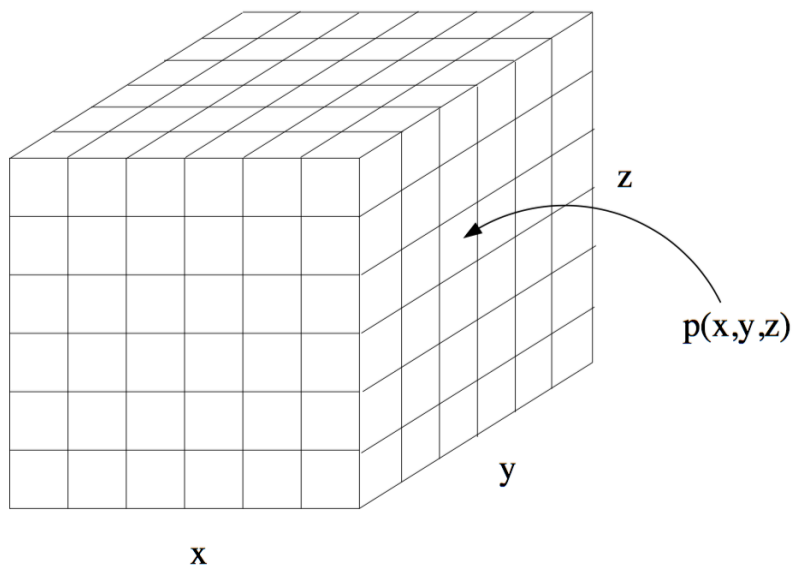
$$E[a] = \langle a \rangle = \sum_x p(x) a(x)$$

e.g. mean = $\sum_x x p(x)$, variance = $\sum_x (x - E[x])^2 p(x)$

- Moments are expectations of higher order powers.
(Mean is first moment. Autocorrelation is second moment.)
- Centralized moments have lower moments subtracted away
(e.g. variance, skew, kurtosis).
- Deep fact: Knowledge of all orders of moments completely defines the entire distribution.

Joint Probability

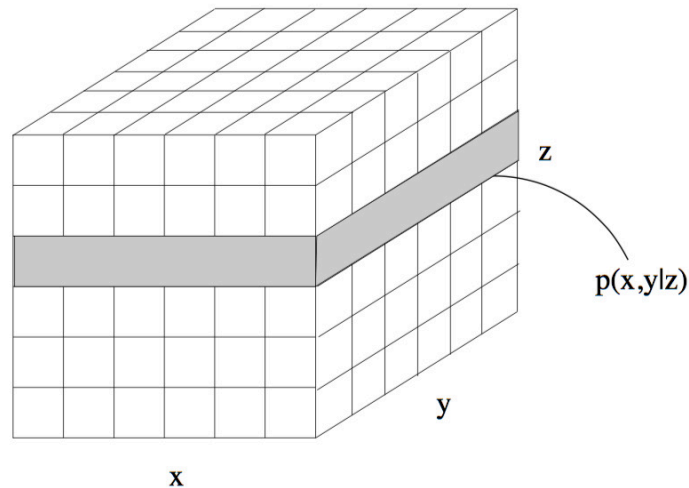
- Key concept: two or more random variables may interact.
Thus, the probability of one taking on a certain value depends on which value(s) the others are taking.
- We call this a joint ensemble and write
$$p(x, y) = \text{prob}(X = x \text{ and } Y = y)$$



Conditional Probability

- If we know that some event has occurred, it changes our belief about the probability of other events.
- This is like taking a "slice" through the joint table.

$$p(x|y) = p(x, y) / p(y)$$

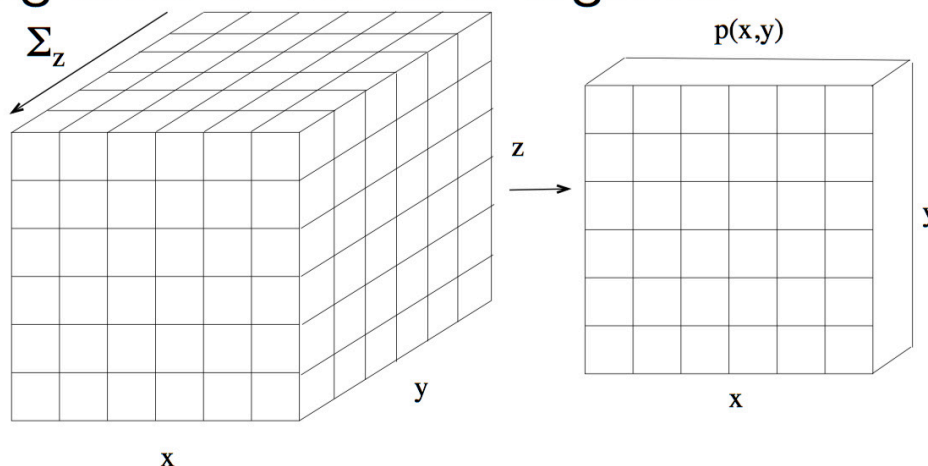


Marginal Probability

- We can "sum out" part of a joint distribution to get the *marginal distribution* of a subset of variables:

$$p(x) = \sum_y p(x, y)$$

- This is like adding slices of the table together.



- Another equivalent definition: $p(x) = \sum_y p(x|y)p(y)$.

Bayes Rule

- Manipulating the basic definition of conditional probability gives one of the most important formulas in probability theory:

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} = \frac{p(y|x)p(x)}{\sum_{x'} p(y|x')p(x')}$$

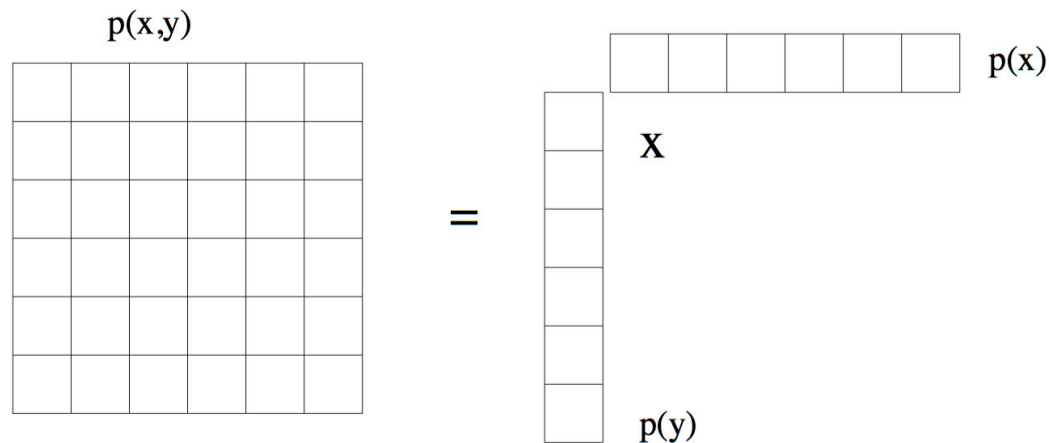
- This gives us a way of "reversing" conditional probabilities.
- Thus, all joint probabilities can be factored by selecting an ordering for the random variables and using the "chain rule":

$$p(x, y, z, \dots) = p(x)p(y|x)p(z|x, y)p(\dots |x, y, z)$$

Conditional Independence

- Two variables are independent iff their joint factors:

$$p(x, y) = p(x)p(y)$$



- Two variables are conditionally independent given a third one if for all values of the conditioning variable, the resulting slice factors:

$$p(x, y|z) = p(x|z)p(y|z) \quad \forall z$$

Independent Event Examples

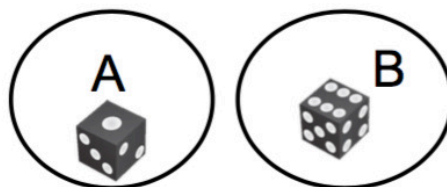
- Independent event example
 - Hardware failures events in different data centers
- Dependent event examples
 - Queries to a search engine and news
 - Tweets and news
 - IM and email communications

Independent Event Examples

- Two events A and B are **independent** if

$$p(A \cap B) = p(A)p(B)$$

- Are these two events independent?



No! $p(A \cap B) = 0, \quad p(A)p(B) = \left(\frac{1}{6}\right)^2$

- Now suppose our outcome space had two different die:

$$\Omega = \{ \text{brown die}, \text{blue die}, \text{brown die}, \text{blue die}, \text{brown die}, \text{blue die}, \dots, \text{brown die}, \text{blue die} \} \quad \text{2 die tosses}$$

$6^2 = 36 \text{ outcomes}$

and the probability distribution is such that each die is independent,

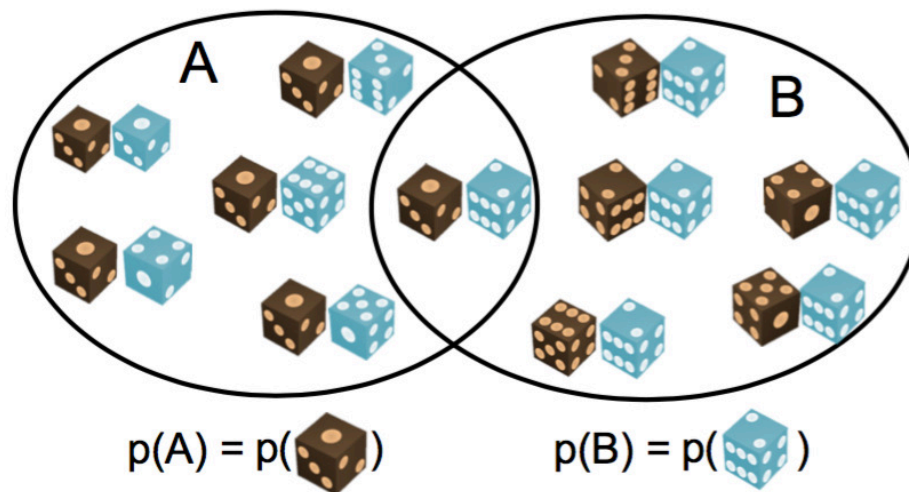
$$p(\text{brown die}, \text{blue die}) = p(\text{brown die}) p(\text{blue die}) \quad p(\text{brown die}, \text{blue die}) = p(\text{brown die}) p(\text{blue die})$$

Independent Event Examples

- Two events A and B are **independent** if

$$p(A \cap B) = p(A)p(B)$$

- Are these two events independent?



Yes!

$$p(A \cap B) = p(\text{brown die with 3 and blue die with 2})$$

$$p(A)p(B) = p(\text{brown die with 3}) p(\text{blue die with 2})$$

Relation to Statistics

- Probability: inferring probabilistic quantities for data given fixed models (e.g. prob. of events, marginals, conditionals, etc).

Relation to Statistics

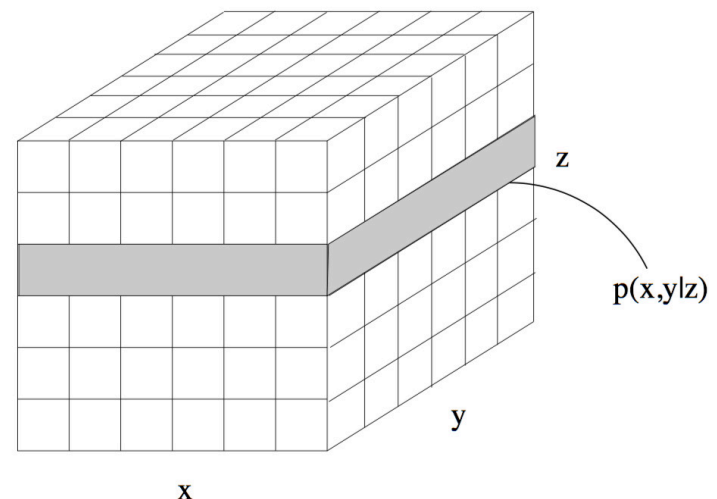
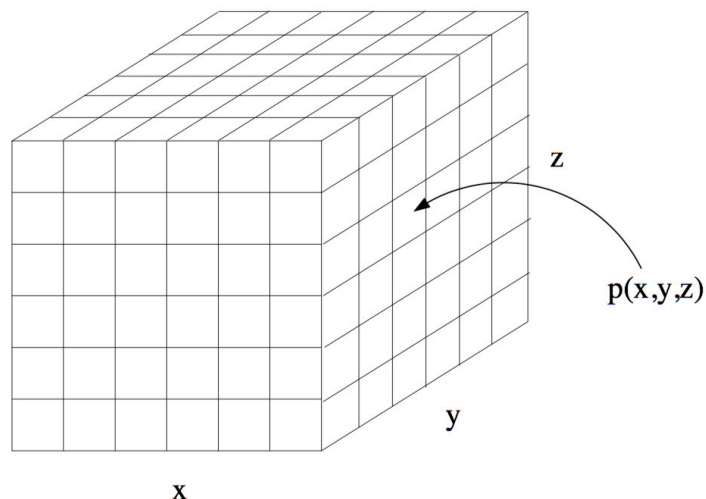
- Probability: inferring probabilistic quantities for data given fixed models (e.g. prob. of events, marginals, conditionals, etc).
- Statistics: inferring a model given fixed data observations (e.g. clustering, classification, regression).
- Many approaches to statistics:
frequentist, Bayesian, decision theory, ...

Conditional Probability Table

- For discrete (categorical) quantities, the most basic parametrization is the probability table which lists $p(x_i = k^{th} \text{ value})$.
- Since PTs must be nonnegative and sum to 1, for k -ary variables there are $k - 1$ free parameters.

Conditional Probability Table

- For discrete (categorical) quantities, the most basic parametrization is the probability table which lists $p(x_i = k^{th} \text{ value})$.
- Since PTs must be nonnegative and sum to 1, for k -ary variables there are $k - 1$ free parameters.
- If a discrete variable is conditioned on the values of some other discrete variables we make one table for each possible setting of the parents: these are called *conditional probability tables* or CPTs.



Likelihood Function

- So far we have focused on the (log) probability function $p(\mathbf{x}|\theta)$ which assigns a probability (density) to any joint configuration of variables \mathbf{x} given fixed parameters θ .
- But in learning we turn this on its head: we have some fixed data and we want to find parameters.
- Think of $p(\mathbf{x}|\theta)$ as a function of θ for fixed \mathbf{x} :

$$L(\theta; \mathbf{x}) = p(\mathbf{x}|\theta)$$

$$\ell(\theta; \mathbf{x}) = \log p(\mathbf{x}|\theta)$$

This function is called the (log) “likelihood”.

- Chose θ to maximize some cost function $c(\theta)$ which includes $\ell(\theta)$:

$$c(\theta) = \ell(\theta; \mathcal{D}) \quad \text{maximum likelihood (ML)}$$

$$c(\theta) = \ell(\theta; \mathcal{D}) + r(\theta) \quad \text{maximum a posteriori (MAP)/penalized ML}$$

(also cross-validation, Bayesian estimators, BIC, AIC, ...)

Complete Data, IID Sampling

- A single observation of the data \mathbf{X} is rarely useful on its own.
- Generally we have data including many observations, which creates a *set of random variables*: $\mathcal{D} = \{\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^M\}$
- Two very common assumptions:
 1. Observations are independently and identically distributed according to joint distribution of graphical model: IID samples.
 2. We observe all random variables in the domain on each observation: complete data.

Maximum Likelihood

- For IID data:

$$p(\mathcal{D}|\theta) = \prod_m p(\mathbf{x}^m|\theta)$$

$$\ell(\theta; \mathcal{D}) = \sum_m \log p(\mathbf{x}^m|\theta)$$

- Idea of maximum likelihood estimation (MLE): pick the setting of parameters most likely to have generated the data we saw:

$$\theta_{\text{ML}}^* = \operatorname{argmax}_{\theta} \ell(\theta; \mathcal{D})$$

- Very commonly used in statistics.
Often leads to “intuitive”, “appealing”, or “natural” estimators.

What to do with a Distribution

- *Generate data*: draw samples from the distribution. This often involves generating a uniformly distributed variable in the range $[0,1]$ and transforming it. For more complex distributions it may involve an iterative procedure that takes a long time to produce a single sample (e.g. Gibbs sampling, MCMC).
- *Compute log probabilities*.
When all variables are either observed or marginalized the result is a single number which is the log prob of the configuration.

What to do with a Distribution

- *Generate data*: draw samples from the distribution. This often involves generating a uniformly distributed variable in the range $[0,1]$ and transforming it. For more complex distributions it may involve an iterative procedure that takes a long time to produce a single sample (e.g. Gibbs sampling, MCMC).
- *Compute log probabilities*.
When all variables are either observed or marginalized the result is a single number which is the log prob of the configuration.
- *Inference*: Compute expectations of some variables given others which are observed or marginalized.
- *Learning*.
Set the parameters of the density functions given some (partially) observed data to maximize likelihood or penalized likelihood.

Advanced Prediction Models

Deep Learning, Graphical Models and Reinforcement
Learning

Recap: Why Graphical Models

- We have seen deep learning techniques for unstructured data
 - Predominantly vision and text/audio
 - We will see control in the last part of the course
 - (Reinforcement Learning)
- For structured data, graphical models are the most versatile framework
 - Successfully applications:
 - Kalman filtering in engineering
 - Decoding in cell phones (channel codes)
 - Hidden Markov models for time series
 - Clustering, regression, classification ...

Recap: Graphical Models Landscape

- Three key parts:
 - Representation
 - Capture uncertainty (joint distribution)
 - Capture **conditional independences** (metadata)
 - Visualization of metadata for a distribution
 - Inference
 - Efficient methods for computing marginal or conditional distributions **quickly**
 - Learning
 - Learning the **parameters of the distribution** can deal with prior knowledge and missing data

Today's Outline

- Inference
 - Factor Graph
 - Variable Elimination
- Inference using Belief Propagation
- Inference using Markov Chain Monte Carlo

Inference

Based on notes from Bjoern Andres and Bernt Schiele (2016)

Inference Objectives

- Let $\bar{X} = X_1, \dots, X_D$ be a random vector.
- Let $\bar{X} \in \mathfrak{X}$ and $X_i \in \mathfrak{X}_i$
- Given $P(\bar{X})$ compute functions of it

Inference Objectives

- Let $\bar{X} = X_1, \dots, X_D$ be a random vector.
- Let $\bar{X} \in \mathfrak{X}$ and $X_i \in \mathfrak{X}_i$
- Given $P(\bar{X})$ compute functions of it
 - Example, find
 - Mode $\bar{x}^* \in \operatorname{argmax}_{\bar{x} \in \mathfrak{X}} P(\bar{x})$
 - Mean $E[g(\bar{x})] = \sum_{\bar{x} \in \mathfrak{X}} g(\bar{x})P(\bar{x})$
 - A marginal $\operatorname{argmax}_{x_i \in \mathfrak{X}_i} \sum_{x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_D} P(\bar{x})$
 - A conditional $P(X_i | x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_D)$

Algorithms for Inference

- Variable Elimination
- Belief Propagation
- Sampling based methods (MCMC)

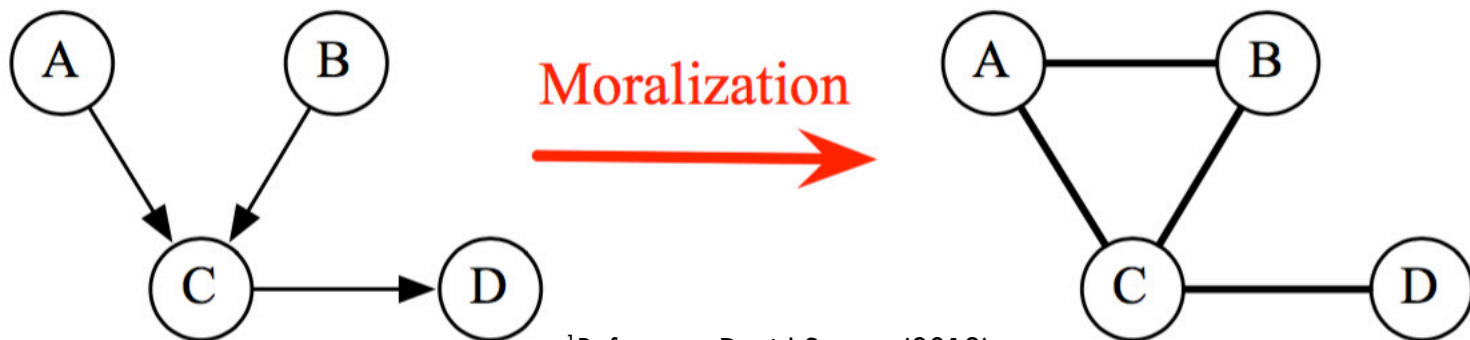
¹Note: There are others, but we will not discuss them here

DPGMs and UPGMs

- Inference algorithms can typically run on both graphs
- For convenience, we will construct a UPGM from a DPGM and discuss inference on UPGM

DPGMs and UPGMs

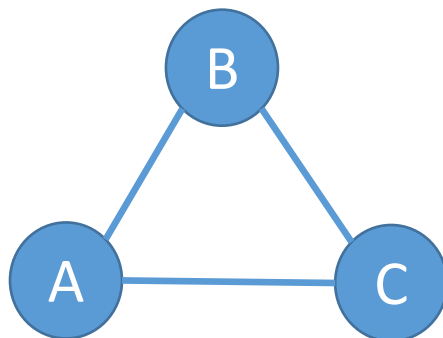
- Inference algorithms can typically run on both graphs
- For convenience, we will construct a UPGM from a DPGM and discuss inference on UPGM
- The construction is straightforward
 - For each factor in DPGM, call it a potential now
 - Moralize the DPGM and remove directions
 - (We lose some information in the graph)



¹Reference: David Sontag (2013)

Factor Graphs

- For both DPGM and UPGMs, factorization is simply not specified by the graph!
- Consider the following example graph

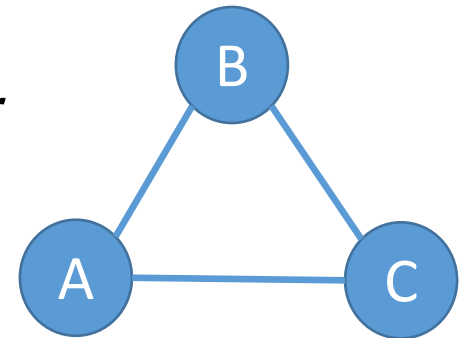


- It could be $P(a, b, c) = \frac{1}{Z} \phi(a, b, c)$
- Or it could be $P(a, b, c) = \frac{1}{Z} \phi_1(a, b) \phi_2(b, c) \phi_3(c, a)$

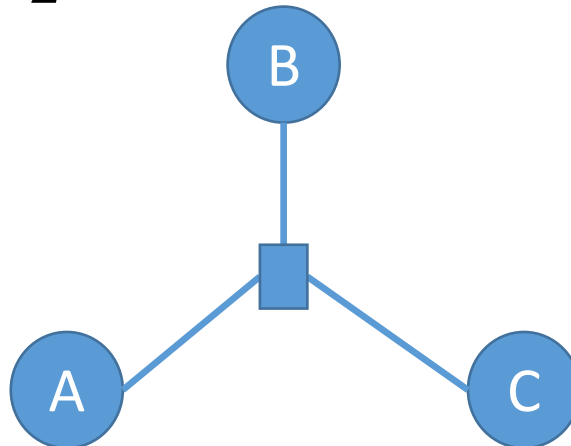
Factor Graph for UPGM

- Hence, we define new graphs called **factor graphs**

- Consider a square node for each factor



- Then, $P(a, b, c) = \frac{1}{Z} \phi(a, b, c)$ can be represented by

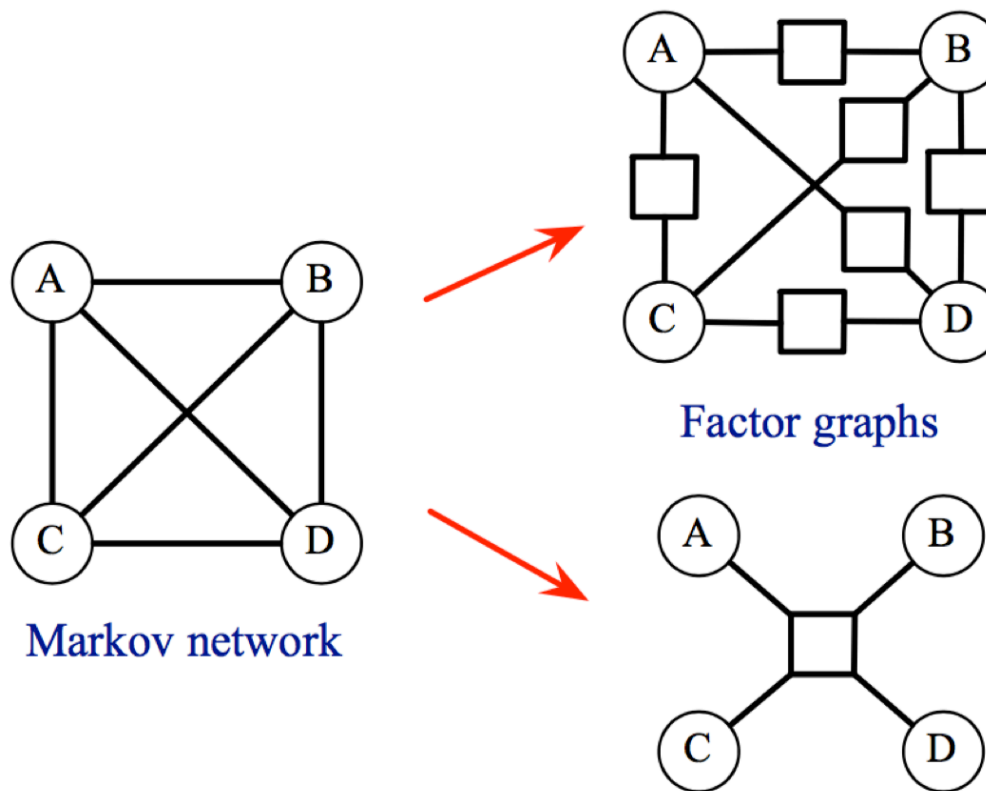


Factor Graph

- We define new graphs called **factor graphs** to capture the factorization in the graph itself
- For a function $f(x_1, \dots, x_D) = \prod_i \phi_i(\mathcal{X}_i)$ the factor graph has a **square** node for each factor $\phi_i(\mathcal{X}_i)$ and a **circular** variable node for each variable x_j
- Factor graphs will allow us to define inference algorithms for both DPGMs and UPGMs
 - Just a more richer way of drawing graphs for $P(X)$

Factor Graphs for a UPGM

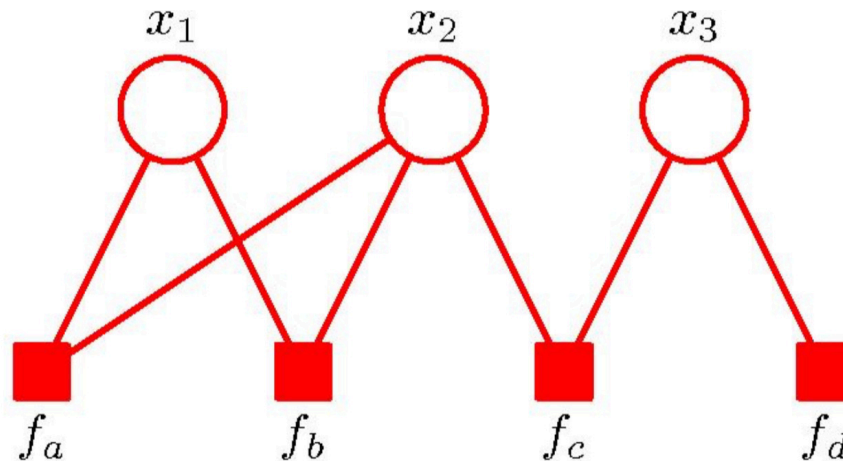
- The following example shows two factor graphs for the same UPGM



Factor Graph Example (I)

- Which distribution does the following graph correspond to?

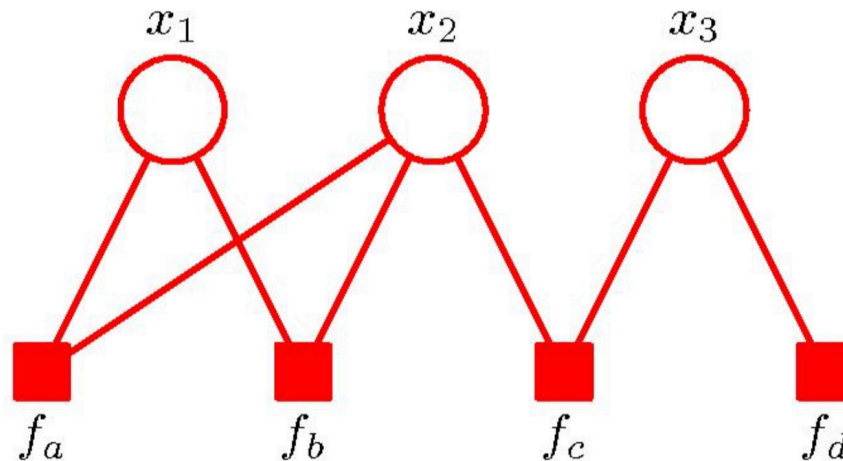
We will use f or ϕ to denote factors



We will use lower case to minimize notation clutter

Factor Graph Example (I)

- Which distribution does the following graph correspond to?



- It corresponds to
 - $P(x_1, x_2, x_3) = \frac{1}{Z} f_a(x_1, x_2) f_b(x_1, x_2) f_c(x_2, x_3) f_d(x_3)$

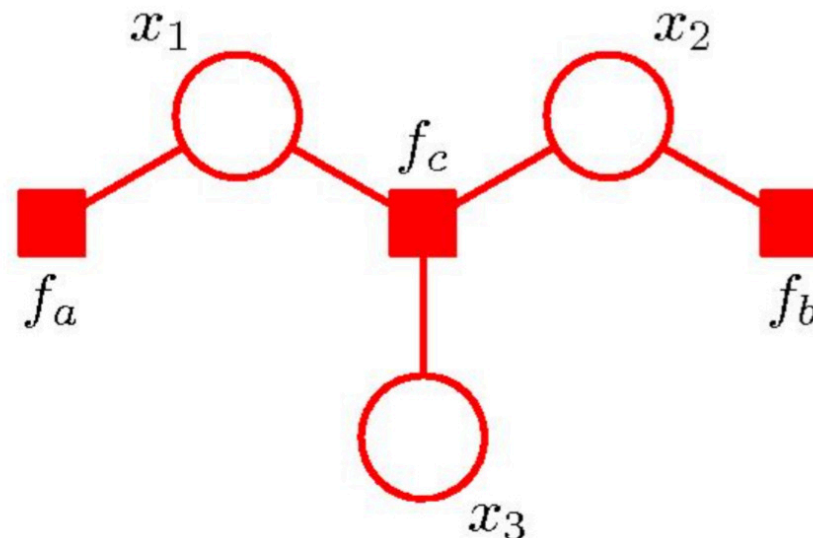
Factor Graph Example (II)

- What is the factor graph for the distribution
 - $P(x_1, x_2, x_3) = \frac{1}{Z} f_c(x_3 | x_1, x_2) f_a(x_1) f_b(x_2)$

Factor Graph Example (II)

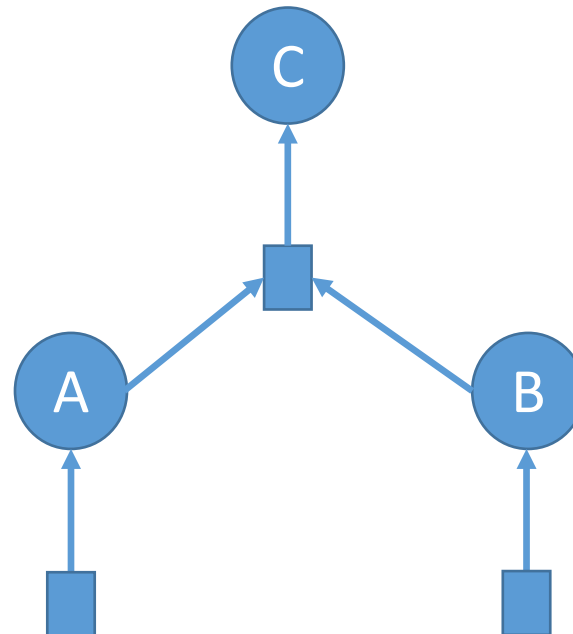
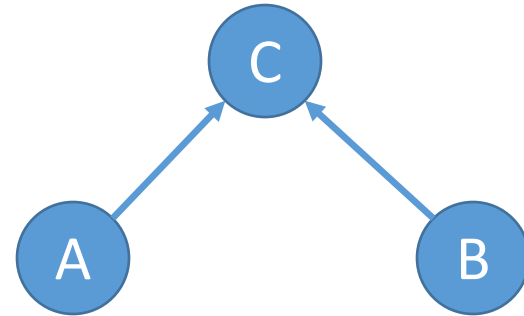
- What is the factor graph for the distribution
 - $P(x_1, x_2, x_3) = \frac{1}{Z} f_c(x_3 | x_1, x_2) f_a(x_1) f_b(x_2)$
- The following is the desired factor graph

¹Reference: Bjoern Andres and Bernt Schiele, MPI (2016)



Factor Graph for DPGM

- We can do this for DPGMs as well (although redundant)
- Consider the graph on the right
- Its factor graph representation is shown below

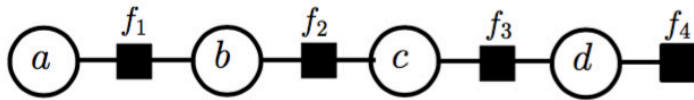


Inference using Variable Elimination

- An algorithm that uses dynamic programming to avoid enumerating all configurations/assignments
- Works for DPGMs and UPGMs
- It is a very simple idea, which is
 - Don't sum over all configurations simultaneously
 - Do it one variable at a time

Variable Elimination Example

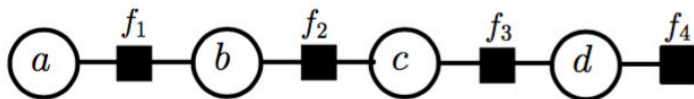
We will use lower case to minimize notation clutter



This can be for a DPGM or a UPGM

Variable Elimination Example

¹Reference: Bjoern Andres and Bernt Schiele, MPI (2016)



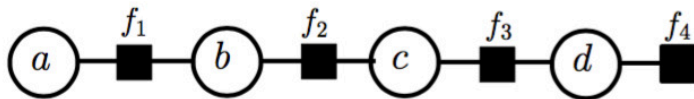
This can be for a DPGM or a UPGM

$$p(a, b, c, d) = \frac{1}{Z} f_1(a, b) f_2(b, c) f_3(c, d) f_4(d)$$

Objective: Find $p(a, b)$

Variable Elimination Example

¹Reference: Bjoern Andres and Bernt Schiele, MPI (2016)



$$p(a, b, c, d) = \frac{1}{Z} f_1(a, b) f_2(b, c) f_3(c, d) f_4(d)$$

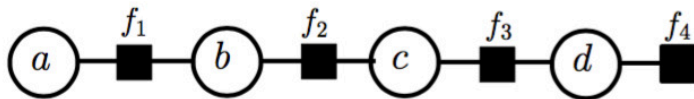
Objective: Find $p(a, b)$

$$p(a, b, c) = \sum_d p(a, b, c, d)$$

$$= \frac{1}{Z} f_1(a, b) f_2(b, c) \underbrace{\sum_d f_3(c, d) f_4(d)}_{\mu_{d \rightarrow c}(c)} \quad (\text{compute this for all } c)$$

Variable Elimination Example

¹Reference: Bjoern Andres and Bernt Schiele, MPI (2016)



$$p(a, b, c, d) = \frac{1}{Z} f_1(a, b) f_2(b, c) f_3(c, d) f_4(d)$$

Objective: Find $p(a, b)$

$$p(a, b, c) = \sum_d p(a, b, c, d)$$

$$= \frac{1}{Z} f_1(a, b) f_2(b, c) \underbrace{\sum_d f_3(c, d) f_4(d)}_{\mu_{d \rightarrow c}(c)} \quad (\text{compute this for all } c)$$

$$p(a, b) = \sum_c p(a, b, c) = \frac{1}{Z} f_1(a, b) \underbrace{\sum_c f_2(b, c) \mu_{d \rightarrow c}(c)}_{\mu_{c \rightarrow b}(b)} \quad (\text{compute this for all } b)$$

Questions?

Today's Outline

- Inference
 - Factor Graph
 - Variable Elimination
- Inference using Belief Propagation
- Inference using Markov Chain Monte Carlo

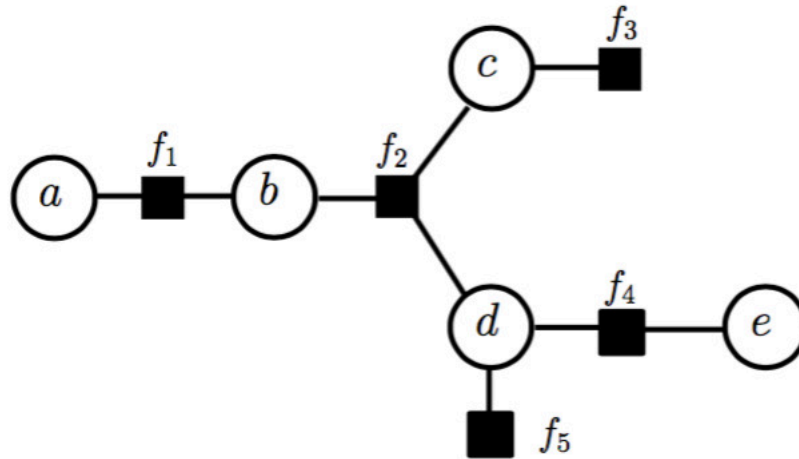
Inference using Belief Propagation

Belief Propagation (BP)

- Generalizes the idea of Variable Elimination
- Also called the Sum-Product Algorithm
- Will give exact answers (marginals, conditionals) on factor graphs that are trees
- Can also be used for general graphs but may give wrong answers

BP Example: Compute a Marginal

consider a branching graph:



with factors

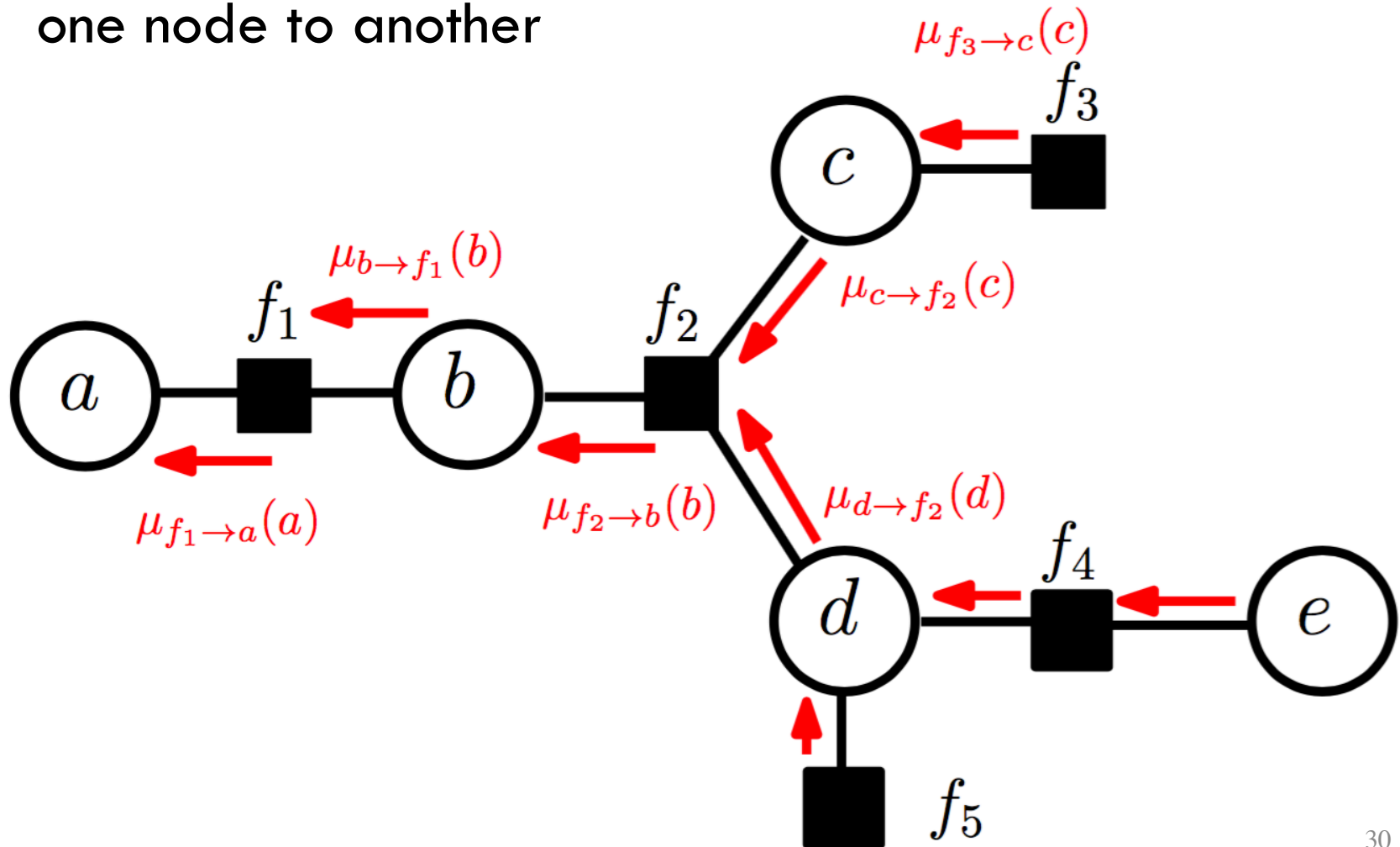
$$f_1(a, b)f_2(b, c, d)f_3(c)f_4(d, e)f_5(d)$$

For example: find marginal $p(a, b)$

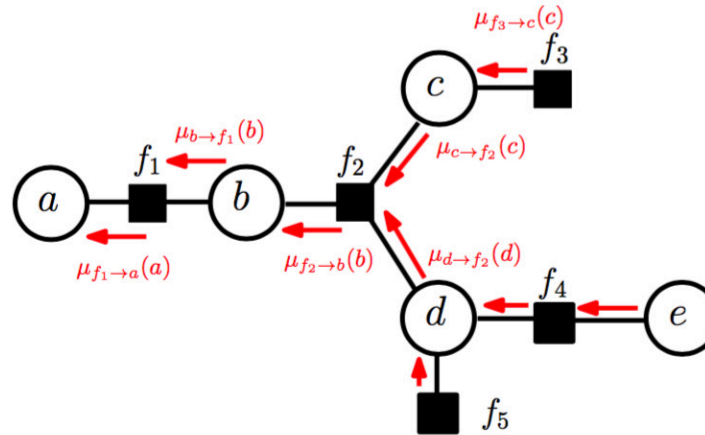
- We will introduce the notion of
 - messages, and
 - message passing

BP Example: Messages

- Messages are functions (vectors) that are passed from one node to another



BP Example: Messages



$$p(a, b) = \frac{1}{Z} f_1(a, b) \underbrace{\sum_{c, d, e} f_2(b, c, d) f_3(c) f_5(d) f_4(d, e)}_{\mu_{f_2 \rightarrow b}(b)}$$

$$\mu_{f_2 \rightarrow b}(b) = \sum_{c, d} f_2(b, c, d) \underbrace{f_3(c)}_{\mu_{c \rightarrow f_2}(c)} \underbrace{f_5(d) \sum_e f_4(d, e)}_{\mu_{d \rightarrow f_2}(d)}$$

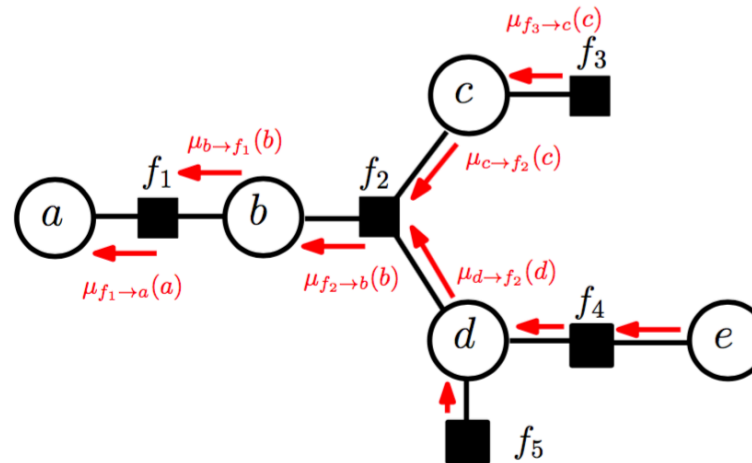
BP Example: Message from Factor to Variable

Here (repeated from last slide):

$$\mu_{f_2 \rightarrow b}(b) = \sum_{c,d} f_2(b, c, d) \underbrace{f_3(c)}_{\mu_{c \rightarrow f_2}(c)} \underbrace{f_5(d) \sum_e f_4(d, e)}_{\mu_{d \rightarrow f_2}(d)}$$

$$\mu_{f_2 \rightarrow b}(b) = \sum_{c,d} f_2(b, c, d) \mu_{c \rightarrow f_2}(c) \mu_{d \rightarrow f_2}(d)$$

BP Example: Message from Factor to Variable



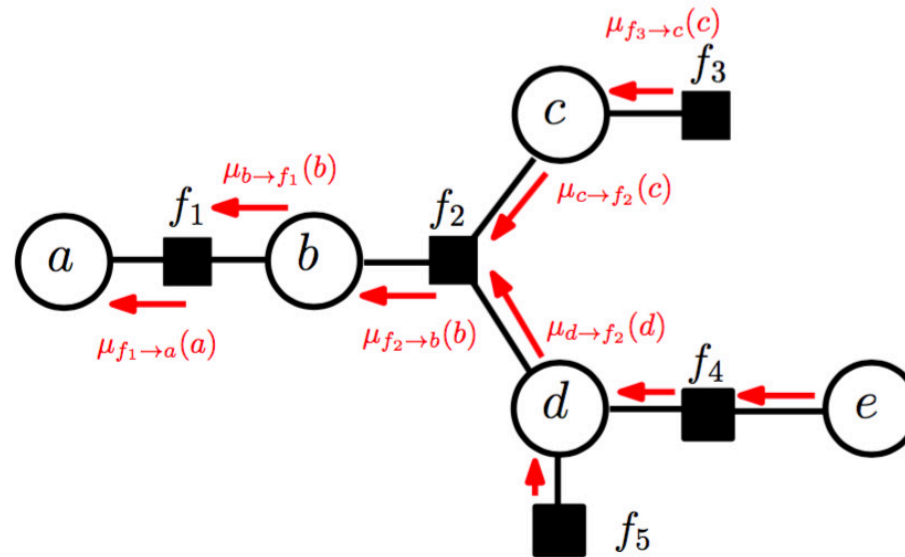
Here (repeated from last slide):

$$\mu_{f_2 \rightarrow b}(b) = \sum_{c,d} f_2(b, c, d) \mu_{c \rightarrow f_2}(c) \mu_{d \rightarrow f_2}(d)$$

more general:

$$\mu_{f \rightarrow x}(x) = \sum_{y \in \mathcal{X}_f \setminus x} \phi_f(\mathcal{X}_f) \prod_{y \in \{\text{ne}(f) \setminus x\}} \mu_{y \rightarrow f}(y)$$

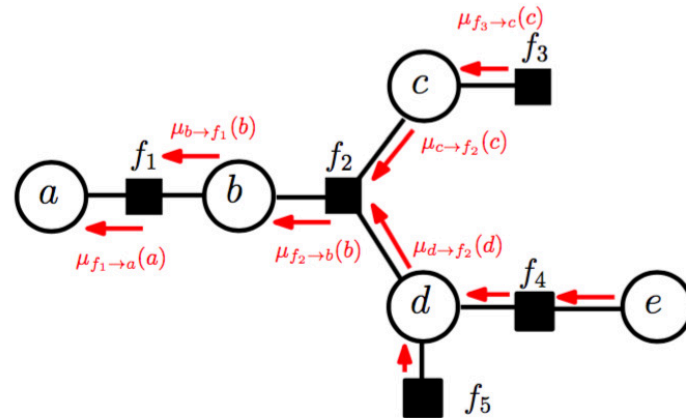
BP Example: Message from Variable to Factor



$$\mu_{d \rightarrow f_2}(d) = \underbrace{f_5(d)}_{\mu_{f_5 \rightarrow d}(d)} \underbrace{\sum_e f_4(d, e)}_{\mu_{f_4 \rightarrow d}(d)}$$

$$\mu_{\textcolor{red}{d} \rightarrow f_2}(\textcolor{red}{d}) = \mu_{f_5 \rightarrow \textcolor{red}{d}}(\textcolor{red}{d}) \mu_{f_4 \rightarrow \textcolor{red}{d}}(\textcolor{red}{d})$$

BP Example: Message from Variable to Factor



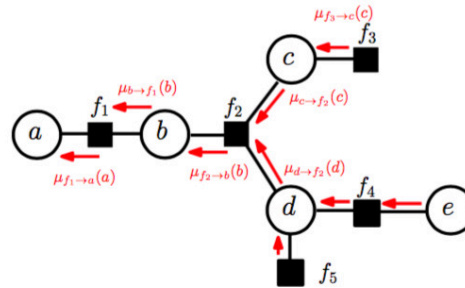
Here (repeated from last slide):

$$\mu_{d \rightarrow f_2}(d) = \mu_{f_5 \rightarrow d}(d) \mu_{f_4 \rightarrow d}(d)$$

General:

$$\mu_{x \rightarrow f}(x) = \prod_{g \in \{\text{ne}(x) \setminus f\}} \mu_{g \rightarrow x}(x)$$

BP Example: Compute a Different Marginal



If we want to compute the marginal $p(a)$
(use factor-to-variable message):

$$p(a) = \frac{1}{Z} \mu_{f_1 \rightarrow a}(a) = \underbrace{\sum_b f_1(a, b) \mu_{b \rightarrow f_1}(b)}_{\mu_{f_1 \rightarrow a}(a)} \frac{1}{Z}$$

which we could also view as

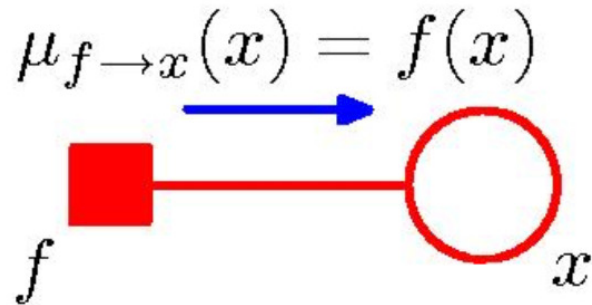
$$p(a) = \frac{1}{Z} \sum_b f_1(a, b) \underbrace{\mu_{b \rightarrow f_1}(b)}_{\mu_{f_2 \rightarrow b}(b)}$$

Belief Propagation Algorithm

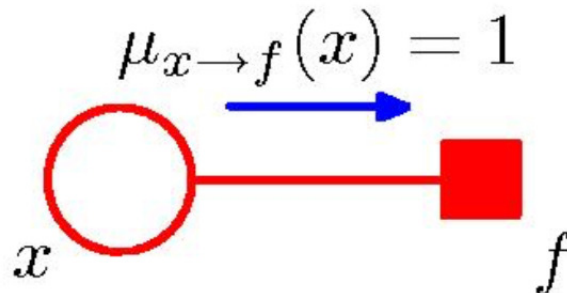
- We described the concept of ‘messages’ via an example (computing marginals for a given factor graph)
- Now we will summarize the algorithm in general
- It has three key ingredients
 - Initialization
 - Variable to factor message
 - Factor to variable message
- Don’t forget the original objective: efficient inference

BP: Initialization

- Messages from extremal/leaf node factors are initialized to be the factor itself

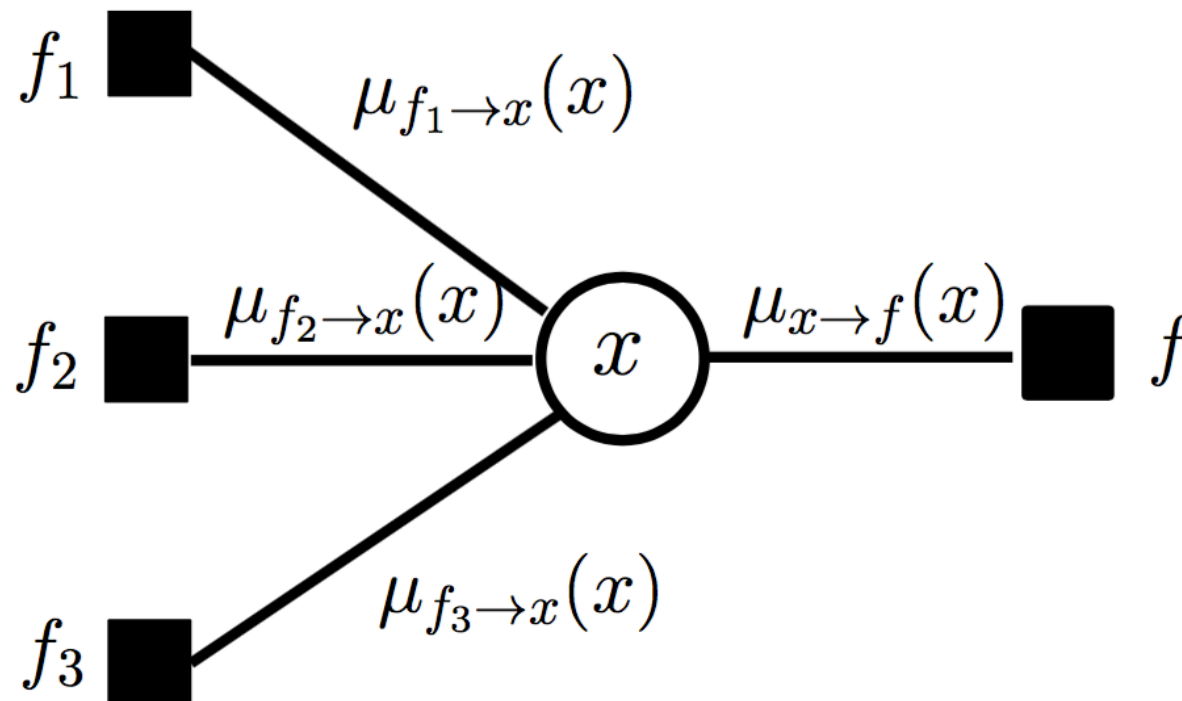


- Messages from extremal/leaf node variables are initialized to value 1



BP: Variable to Factor Message

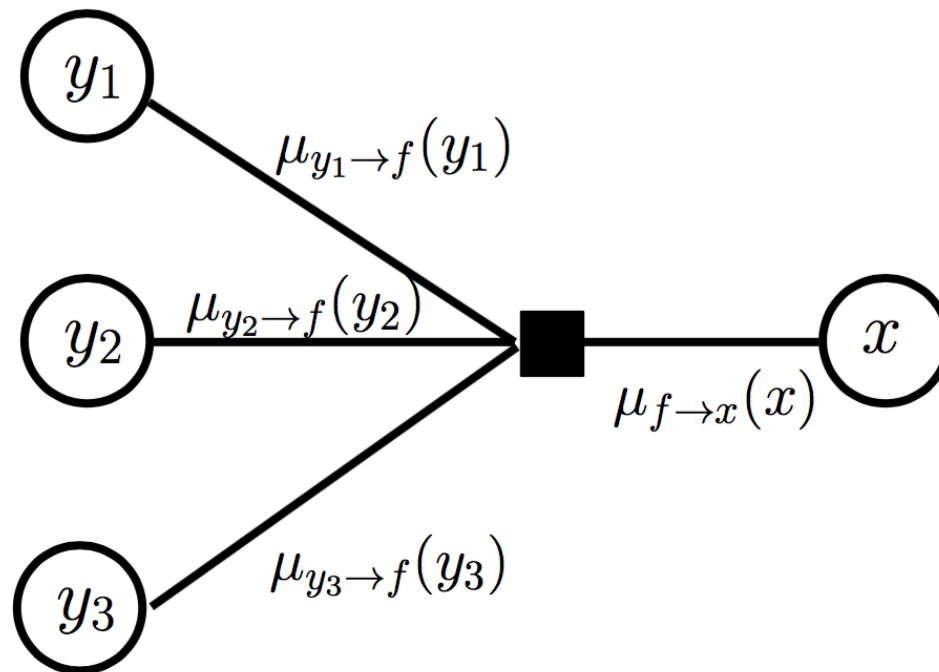
$$\mu_{x \rightarrow f}(x) = \prod_{g \in \{\text{ne}(x) \setminus f\}} \mu_{g \rightarrow x}(x)$$



BP: Factor to Variable Message

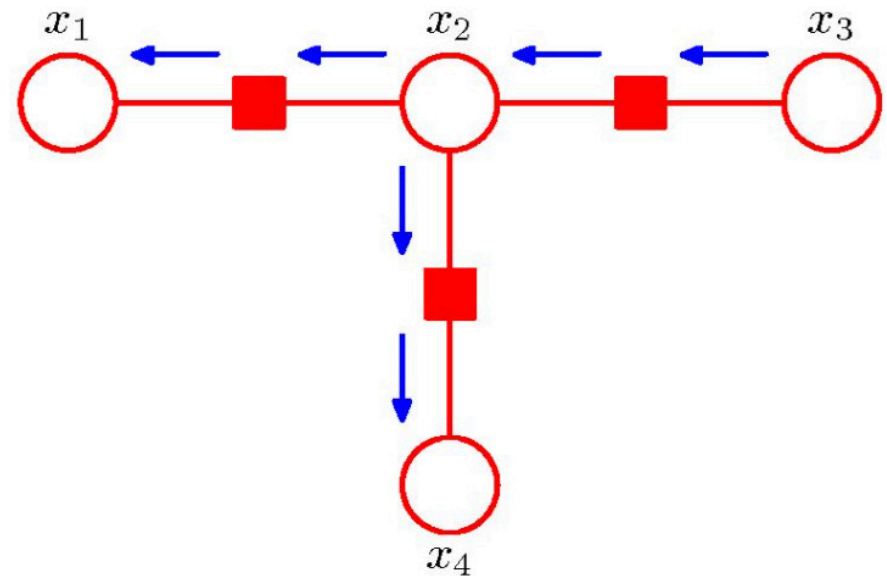
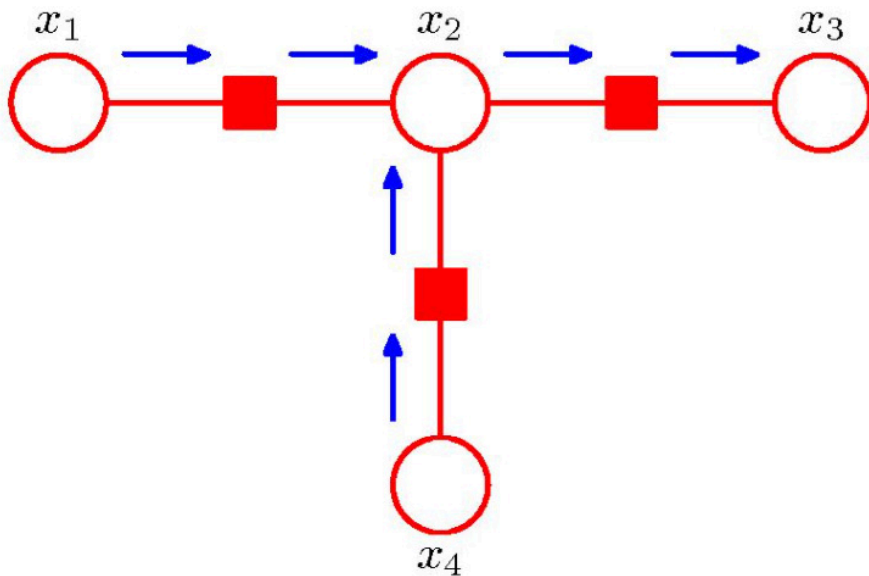
- We sum over all values possible in the scope of the factor

$$\mu_{f \rightarrow x}(x) = \sum_{y \in \mathcal{X}_f \setminus x} \phi_f(\mathcal{X}_f) \prod_{y \in \{\text{ne}(f) \setminus x\}} \mu_{y \rightarrow f}(y)$$



BP: Ordering of Messages

- Messages depend on all incoming messages
- To compute all messages
 - Go from leaves to a designated root (say x_3)
 - Go from the designated root back to leaves

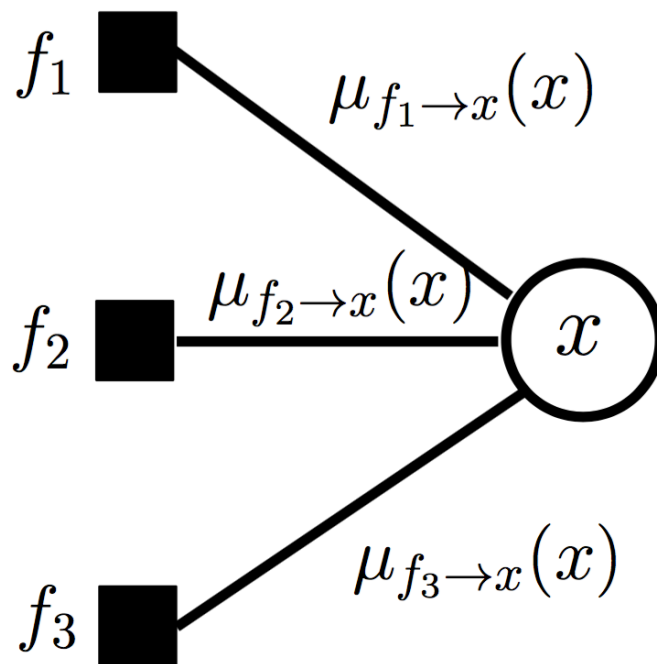


Designated root: x_3

BP: Computing a Marginal

- Marginal is simply the product of messages the variable of interest receives

$$p(x) \propto \prod_{f \in \text{ne}(x)} \mu_{f \rightarrow x}(x)$$



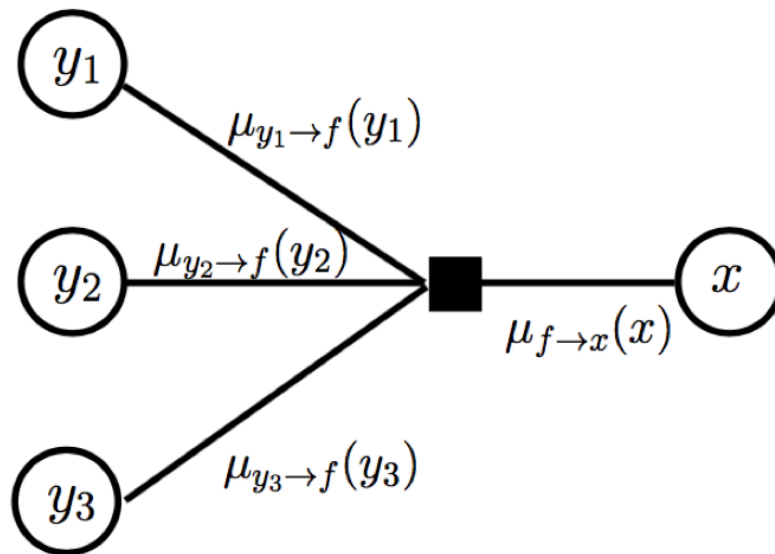
BP: Computing Maximal State

- BP variant can also solve for the maximal state $\bar{x}^* \in \operatorname{argmax}_{\bar{x} \in \mathcal{X}} P(\bar{x})$
- This version is called **Max-Product Belief Propagation**
- Has three ingredients just as before
 - Initialization (same as before)
 - Variable to factor message (same as before)
 - Factor to variable message

BP: Computing Maximal State

- Factor to variable message is different from Sum-Product

$$\mu_{f \rightarrow x}(x) = \max_{y \in \mathcal{X}_f \setminus x} \phi_f(\mathcal{X}_f) \prod_{y \in \{\text{ne}(f) \setminus x\}} \mu_{y \rightarrow f}(y)$$



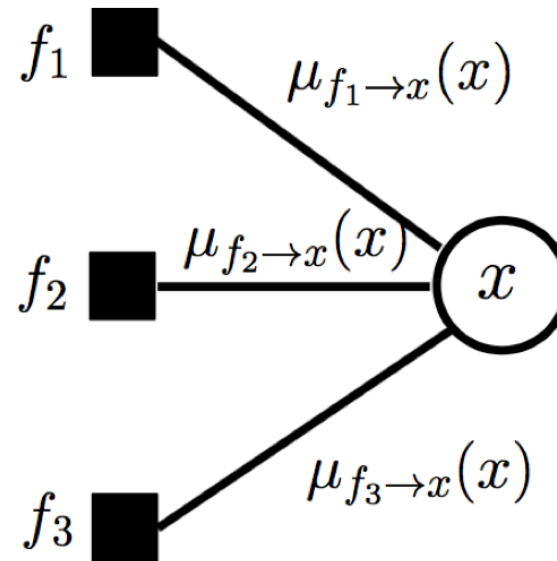
- Additionally, we need to track values achieving maximums as well

¹Reference: Bjoern Andres and Bernt Schiele, MPI (2016)

BP: Computing Maximal State

- Maximal state of a variable is

$$x^* = \operatorname{argmax}_x \prod_{f \in \operatorname{ne}(x)} \mu_{f \rightarrow x}(x)$$



BP: General Factor Graphs

- Is in-exact
- Since it is not clear whether BP is a clear winner for inference with general graphs (among competing algorithms), we will not explore this further.
- See https://en.wikipedia.org/wiki/Belief_propagation for more details

Questions?

Today's Outline

- Inference
 - Factor Graphs
 - Variable Elimination
- Inference using Belief Propagation
- Inference using Markov Chain Monte Carlo

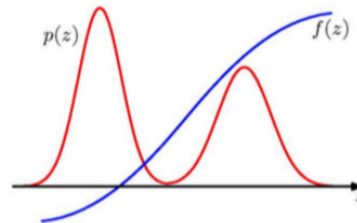
Inference using Markov Chain Monte Carlo

See https://en.wikipedia.org/wiki/Markov_chain_Monte_Carlo

Approximate Inference

- BP and Variable Elimination are exact algorithms
- They work for tree structured factor graphs
- We will resort to **numerical sampling** to perform approximate inference for general graphical models
 - Essentially, use random sampling to approximate

Monte Carlo Methods



We want to evaluate

$$\mathbb{E}[f] = \int f(x)p(x)dx \quad \text{or} \quad \mathbb{E}[f] = \sum_{x \in \mathcal{X}} f(x)p(x)$$

Sampling idea:

- ▶ draw L independent samples x^1, x^2, \dots, x^L from $p(\cdot)$: $x^l \sim p(\cdot)$
- ▶ replace the integral/sum with the finite set of samples

$$\hat{f} = \frac{1}{L} \sum_{l=1}^L f(x^l)$$

- ▶ as long as $x^l \sim p(\cdot)$ then

$$\mathbb{E}[\hat{f}] = \mathbb{E}[f]$$

¹Reference: Bjoern Andres and Bernt Schiele, MPI (2016)

²Reference: https://en.wikipedia.org/wiki/Monte_Carlo_method

Sampling

- Many methods in the literature
- Monte Carlo methods
 - Rejection sampling
 - Importance sampling
- Markov Chain Monte Carlo methods
 - Gibbs sampling
 - Metropolis-Hastings sampling
- ...

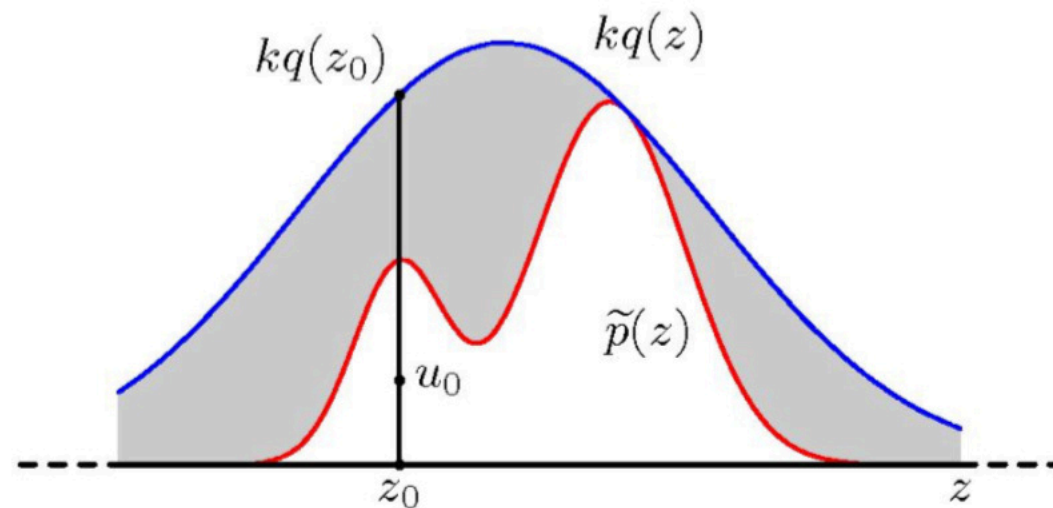
Rejection Sampling

Sample two random variables:

1. $z_0 \sim q(x)$
2. $u_0 \sim [0, kq(z_0)]$ uniform

reject sample z_0 if $u_0 > \tilde{p}(z_0)$

*$q(x)$ is a proposal distribution
such that $kq(x) \geq p(x) \forall x$*



¹Reference: Bjoern Andres and Bernt Schiele, MPI (2016)

²Reference: https://en.wikipedia.org/wiki/Rejection_sampling

Rejection Sampling

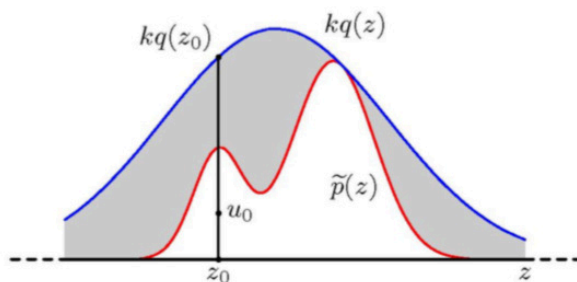
Sample z drawn from q and accepted with probability $\tilde{p}(z)/kq(z)$

So (overall) acceptance probability

$$p(\text{accept}) = \int \frac{\tilde{p}(z)}{kq(z)} q(z) dz = \frac{1}{k} \int \tilde{p}(z) dz$$

So the lower k the better (more acceptance)

- ▶ subject to constraint $kq(z) \geq \tilde{p}(z)$



- Impractical in high dimensions (lots of samples will get rejected)

Rejection Sampling

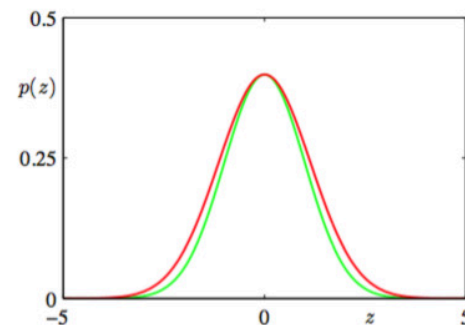
Example:

- ▶ assume $p(x)$ is Gaussian with covariance matrix: $\sigma_p^2 I$
- ▶ assume $q(x)$ is Gaussian with covariance matrix: $\sigma_q^2 I$
- ▶ clearly: $\sigma_q^2 \geq \sigma_p^2$
- ▶ in D dimensions: $k = \left(\frac{\sigma_q}{\sigma_p}\right)^D$

assume:

- ▶ σ_q is 1% larger than σ_p , $D = 1000$
- ▶ then $k = 1.01^{1000} \geq 20000$
- ▶ and $p(\text{accept}) \leq \frac{1}{20000}$

therefore: often impractical to find good proposal distribution $q(x)$ for high dimensions



Importance Sampling

- This is a variance reduction technique to Monte Carlo averaging
- A clever way to estimate expectations

- Objective is $\mathbb{E}[f] = \int f(z)p(z)dz$

- Solution

- Sample L points
- Compute $\mathbb{E}[f] \simeq \sum_{l=1}^L f(z^l)p(z^l)$

¹Reference: Bjoern Andres and Bernt Schiele, MPI (2016)

²Reference: https://en.wikipedia.org/wiki/Importance_sampling

Importance Sampling

use a proposal distribution $q(z)$ from which it is easy to draw samples
express expectation in the form of a finite sum over samples $\{z^l\}$
drawn from $q(z)$:

$$\begin{aligned}\mathbb{E}[f] &= \int f(z)p(z)dz = \int f(z)\frac{p(z)}{q(z)}q(z)dz \\ &\simeq \frac{1}{L} \sum_{l=1}^L \frac{p(z^l)}{q(z^l)} f(z^l)\end{aligned}$$

with importance weights: $r^l = \frac{p(z^l)}{q(z^l)}$

¹Reference: Bjoern Andres and Bernt Schiele, MPI (2016)

²Reference: https://en.wikipedia.org/wiki/Importance_sampling

Importance Sampling

- If we can only evaluate up to a normalizing constant, then additional tricks needed.

$p(z)$ can be only evaluated up to a normalization constant (unknown):

$$p(z) = \tilde{p}(z)/Z_p$$

$q(z)$ can be also treated in a similar way:

$$q(z) = \tilde{q}(z)/Z_q$$

then:

$$\begin{aligned}\mathbb{E}[f] &= \int f(z)p(z)dz = \frac{Z_q}{Z_p} \int f(z) \frac{\tilde{p}(z)}{\tilde{q}(z)} q(z)dz \\ &\simeq \frac{Z_q}{Z_p} \frac{1}{L} \sum_{l=1}^L \tilde{r}^l f(z^l)\end{aligned}$$

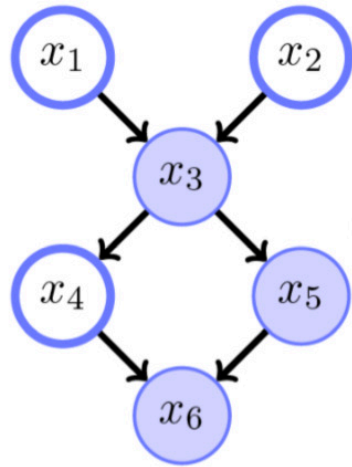
with: $\tilde{r}^l = \frac{\tilde{p}(z^l)}{\tilde{q}(z^l)}$

For example, $\frac{Z_p}{Z_q} \simeq \frac{1}{L} \sum_{l=1}^L \tilde{r}^l$

¹Reference: Bjoern Andres and Bernt Schiele, MPI (2016)

²Reference: https://en.wikipedia.org/wiki/Importance_sampling

Gibbs Sampling MCMC



Sample from this distribution $p(x)$

Idea: Sample sequence x^0, x^1, x^2, \dots by updating one variable at a time

Eg. update x_4 by conditioning on the set of shaded variables **Markov blanket**

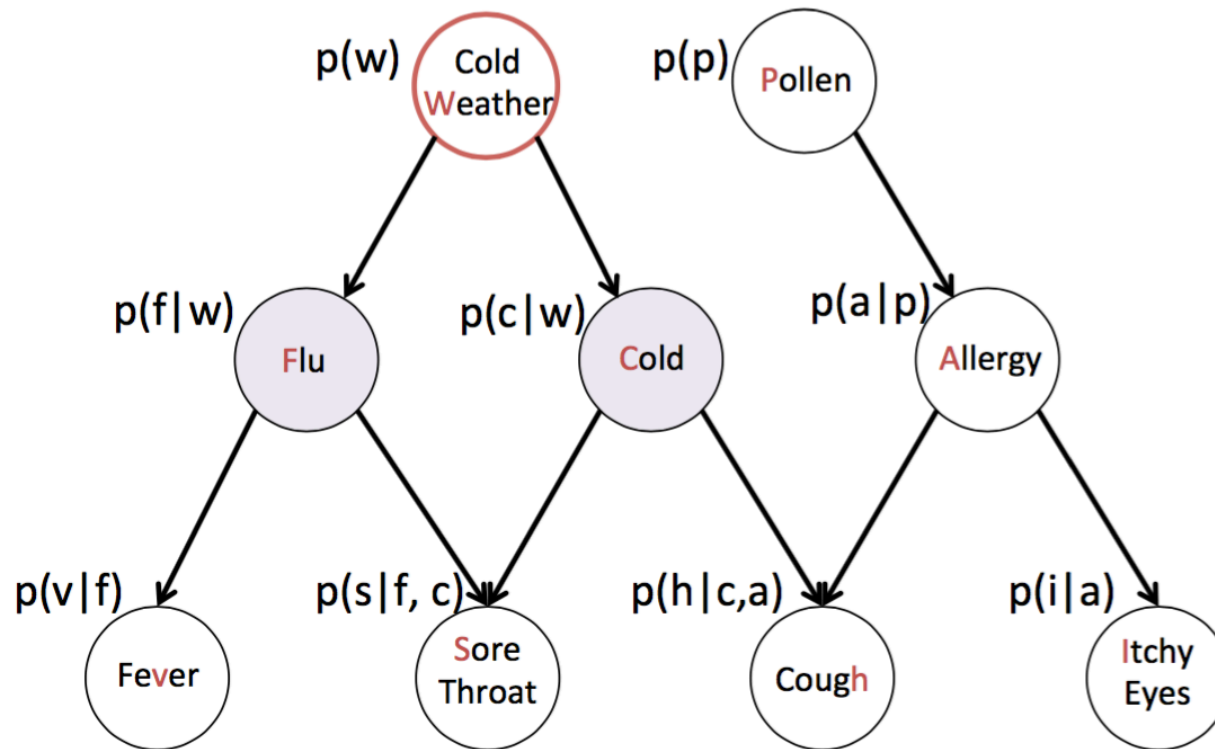
$$p(x_4 \mid x_1, x_2, x_3, x_5, x_6) = p(x_4 \mid x_3, x_5, x_6)$$

¹Reference: Bjoern Andres and Bernt Schiele, MPI (2016)

²Reference: https://en.wikipedia.org/wiki/Gibbs_sampling

Gibbs Sampling Example I

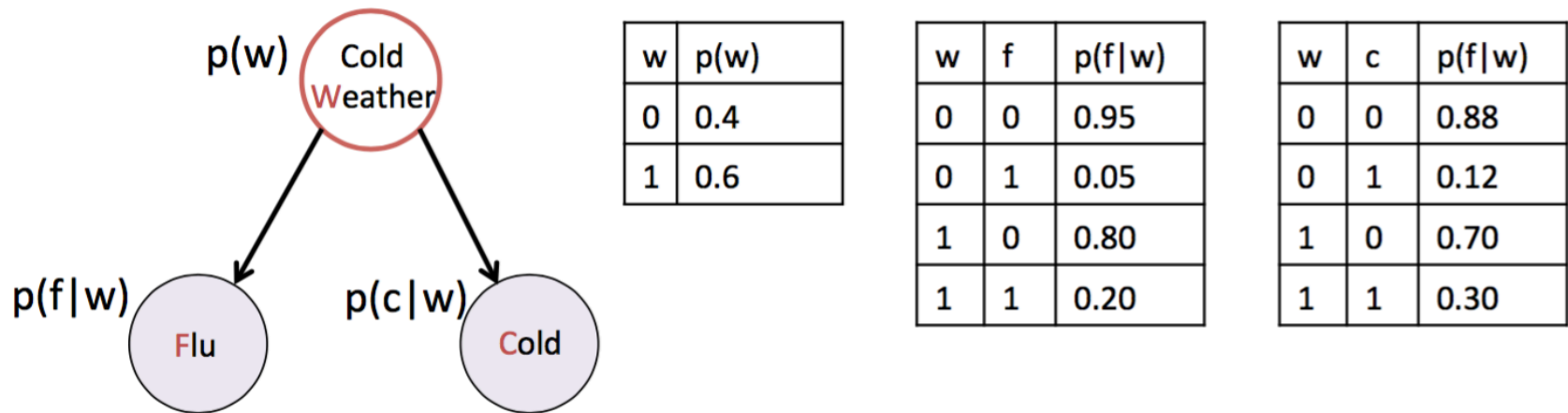
How do we sample a new value for W?



$$\begin{aligned} &P(W=w|F=1, P=1, C=0, \dots, I=0) \\ &= P(W=w|F=1, C=0) \end{aligned}$$

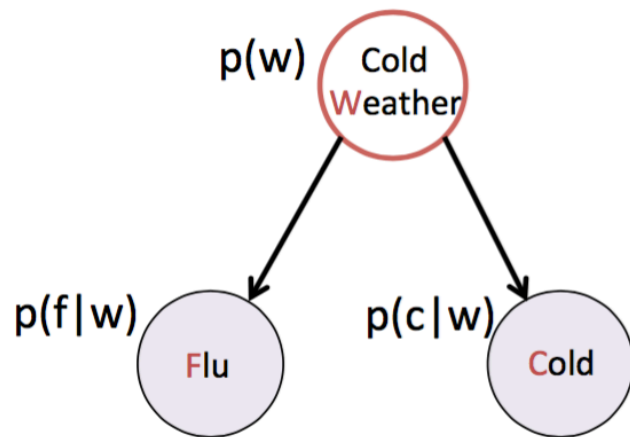
Markov Blanket!

Gibbs Sampling Example I



$$\begin{aligned} & P(W=w|F=1, P=1, C=0, \dots, I=0) \\ &= P(W=w|F=1, C=0) \\ &\propto P(F=1|W=w) * P(C=0|W=w) * P(W=w) \end{aligned}$$

Gibbs Sampling Example I



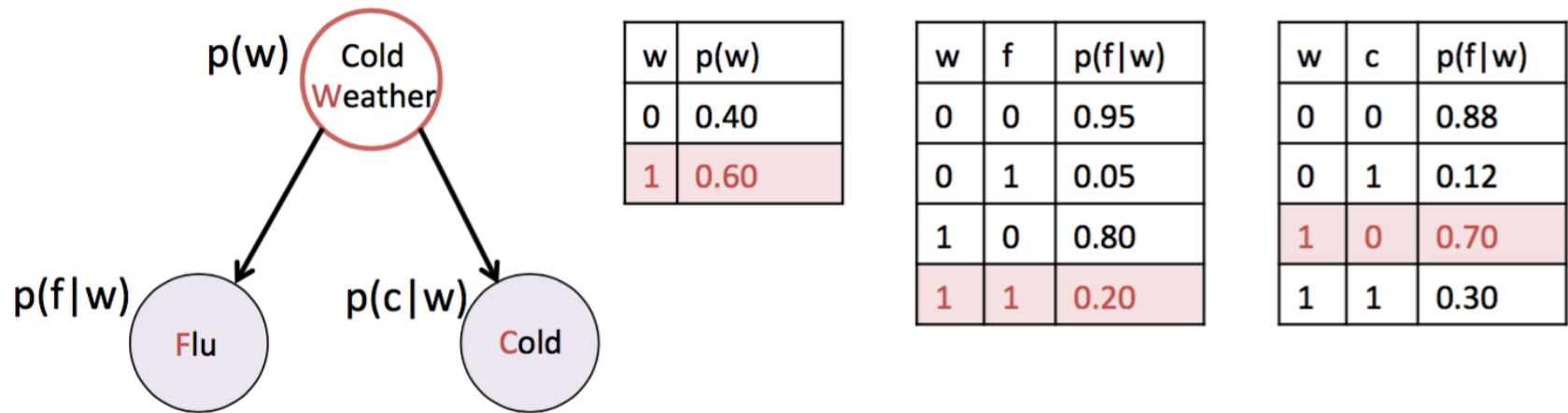
w	p(w)
0	0.40
1	0.60

w	f	p(f w)
0	0	0.95
0	1	0.05
1	0	0.80
1	1	0.20

w	c	p(c w)
0	0	0.88
0	1	0.12
1	0	0.70
1	1	0.30

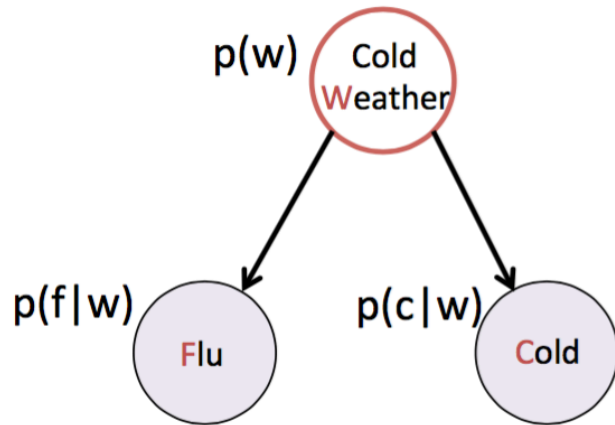
$$\begin{aligned}
 & P(W=w|F=1, P=1, C=0, \dots, I=0) \\
 &= P(W=w|F=1, C=0) \\
 &\propto P(F=1|W=w) * P(C=0|W=w) * P(W=w) \\
 &= \begin{cases} 0.05 * 0.88 * 0.40, & W = 0 \\ \end{cases}
 \end{aligned}$$

Gibbs Sampling Example I



$$\begin{aligned}
 & P(W=w|F=1, C=0, \dots, I=0) \\
 &= P(W=w|F=1, C=0) \\
 &\propto P(F=1|W=w) * P(C=0|W=w) * P(W=w) \\
 &= \begin{cases} 0.05 * 0.88 * 0.40, & W = 0 \\ 0.20 * 0.70 * 0.60, & W = 1 \end{cases}
 \end{aligned}$$

Gibbs Sampling Example I



w	p(w)
0	0.40
1	0.60

w	f	p(f w)
0	0	0.95
0	1	0.05
1	0	0.80
1	1	0.20

w	c	p(c w)
0	0	0.88
0	1	0.12
1	0	0.70
1	1	0.30

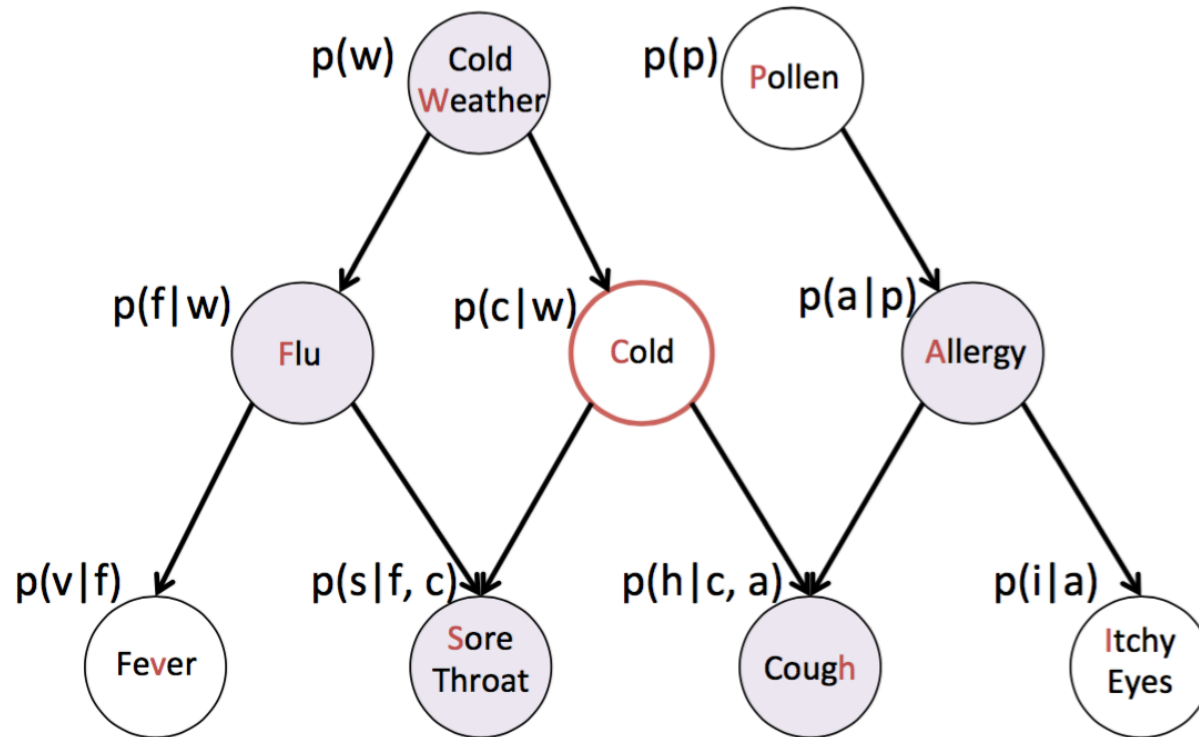
$$\begin{aligned}
 & P(W=w|F=1, C=0, \dots, I=0) \\
 &= P(W=w|F=1, C=0) \\
 &\propto P(F=1|W=w) * P(C=0|W=w) * P(W=w) \\
 &= \begin{cases} 0.05 * 0.88 * 0.40, & W = 0 \\ 0.20 * 0.70 * 0.60, & W = 1 \end{cases}
 \end{aligned}$$

$$\begin{aligned}
 & P(W = w | F = 1, C = 0) \\
 &= \begin{cases} 0.0176 / (0.0176 + 0.084), & w = 0 \\ 0.084 / (0.0176 + 0.084), & w = 1 \end{cases} \\
 &= \begin{cases} 0.173, & w = 0 \\ 0.827, & w = 1 \end{cases}
 \end{aligned}$$

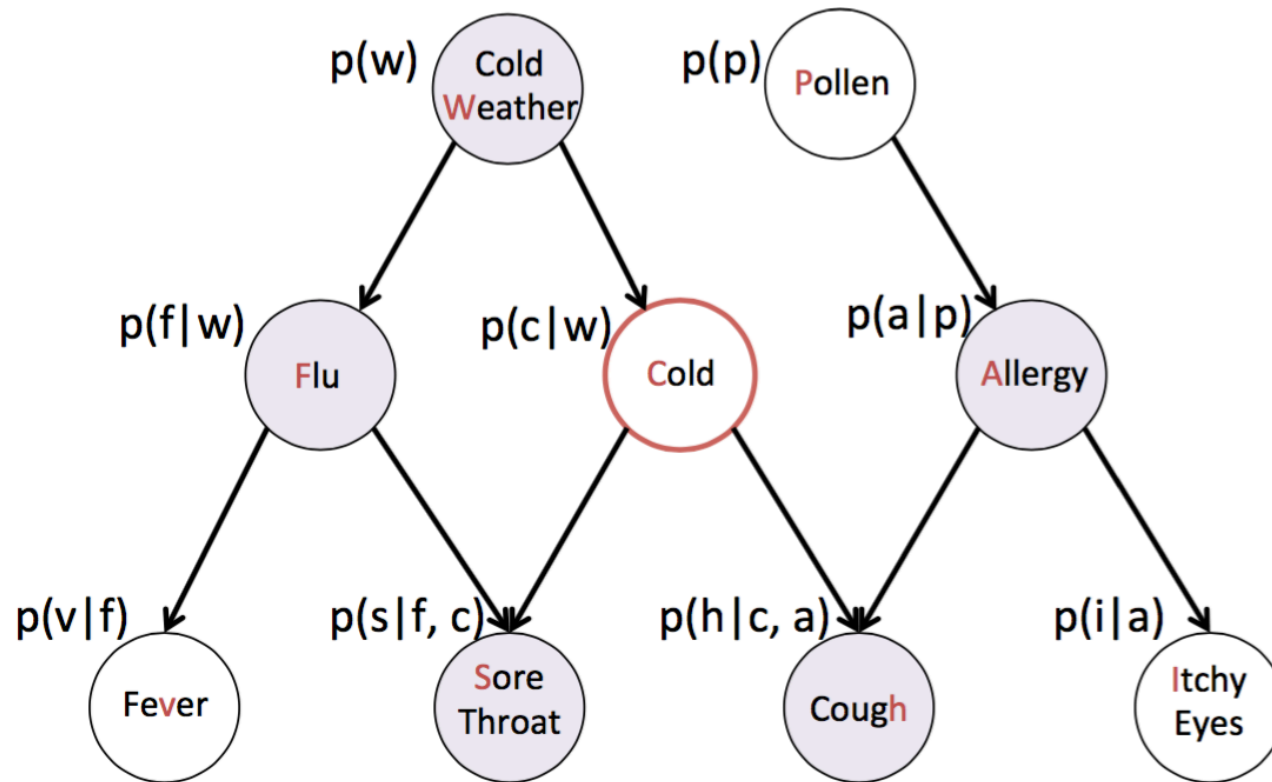
Sample a new w!

Gibbs Sampling Example II

How do we sample a new value for C?

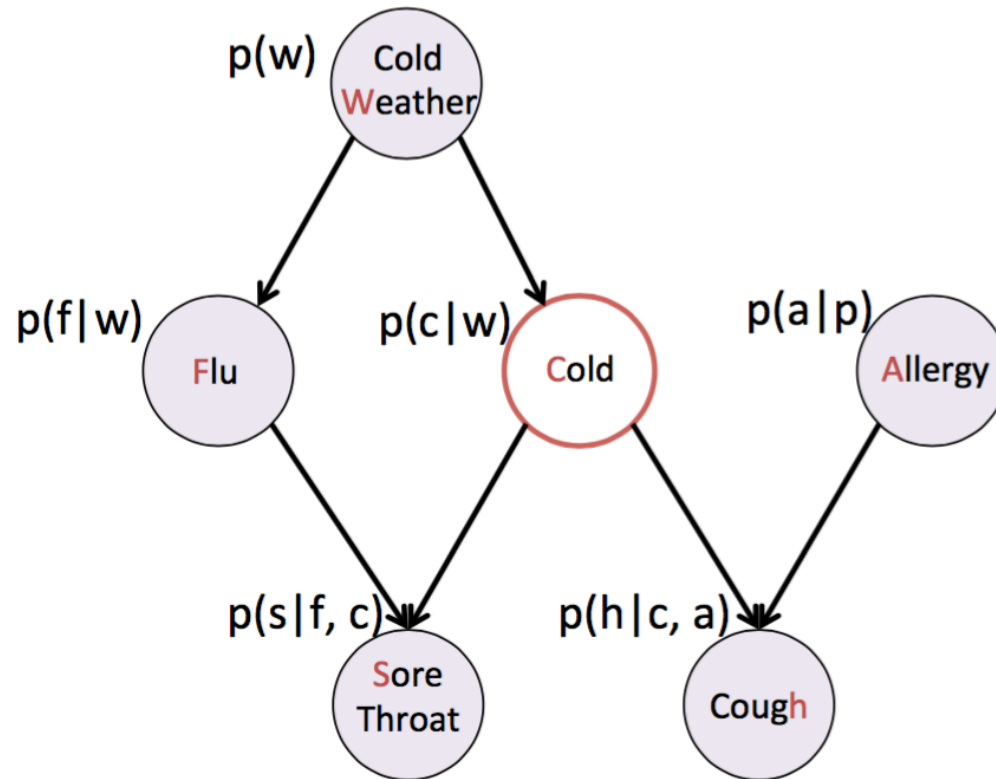


Gibbs Sampling Example II



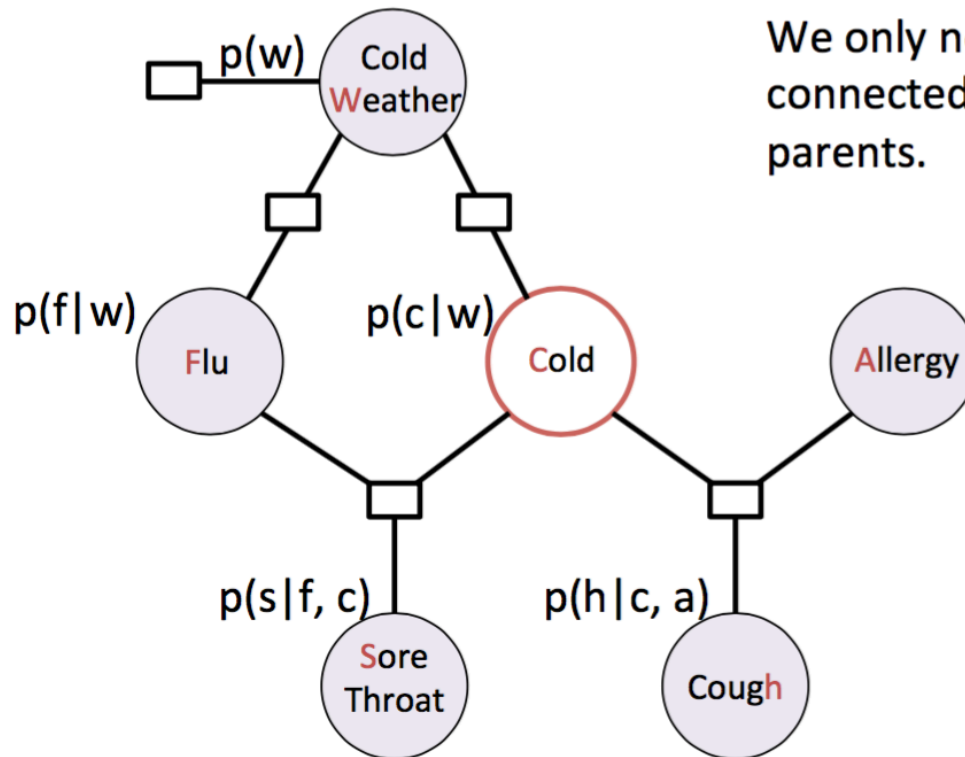
$$\begin{aligned} &P(C=c \mid W=1, F=1, P=1, \dots, I=0) \\ &= P(C=c \mid W=1, F=1, S=0, H=1, A=1) \quad \text{Markov Blanket!} \end{aligned}$$

Gibbs Sampling Example II



$$\begin{aligned} & P(C=c \mid W=1, F=1, P=1, \dots, I=0) \\ &= P(C=c \mid W=1, F=1, S=0, H=1, A=1) \end{aligned}$$

Gibbs Sampling Example II



$$\begin{aligned} &P(C=c \mid W=1, F=1, P=1, \dots, I=0) \\ &= P(C=c \mid W=1, F=1, S=0, H=1, A=1) \\ &= p(w) p(f \mid w) p(c \mid w) p(s \mid f, c) p(h \mid c, a) \end{aligned}$$

Gibbs Sampling MCMC

Update x_i

$$p(x_i \mid x_{\setminus i}) = \frac{1}{Z} p(x_i \mid pa(x_i)) \prod_{j \in \text{ch}(i)} p(x_j \mid \text{pa}(x_j))$$

and the normalisation constant is

$$Z = \sum_{x_i} p(x_i \mid pa(x_i)) \prod_{j \in \text{ch}(i)} p(x_j \mid \text{pa}(x_j))$$

¹Reference: Bjoern Andres and Bernt Schiele, MPI (2016)

²Reference: https://en.wikipedia.org/wiki/Gibbs_sampling

Gibbs Sampling MCMC

Think of Gibbs sampling as

$$x^{l+1} \sim q(\cdot \mid x^l)$$

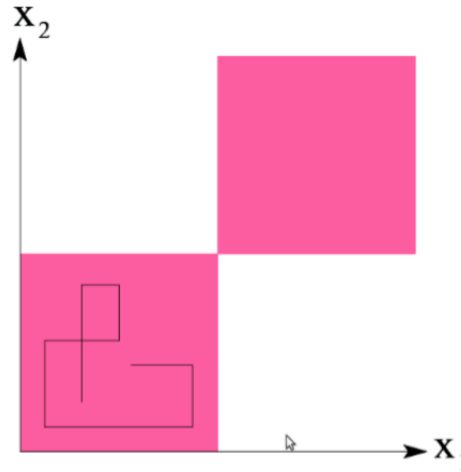
Problem: States are highly dependent (x^1, x^2, \dots)

Need a long time to run Gibbs sampling to *forget* the initial state, this is called **burn in** phase

¹Reference: Bjoern Andres and Bernt Schiele, MPI (2016)

²Reference: https://en.wikipedia.org/wiki/Gibbs_sampling

Gibbs Sampling MCMC



In this example the samples stay in the lower left quadrant

Some technical requirements to Gibbs sampling

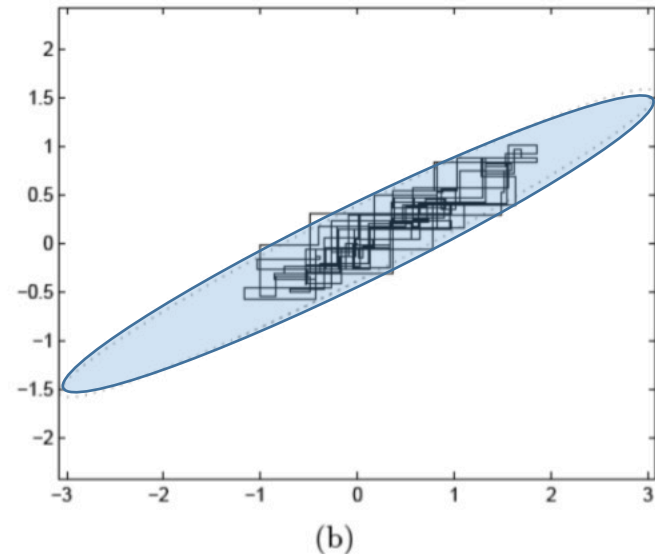
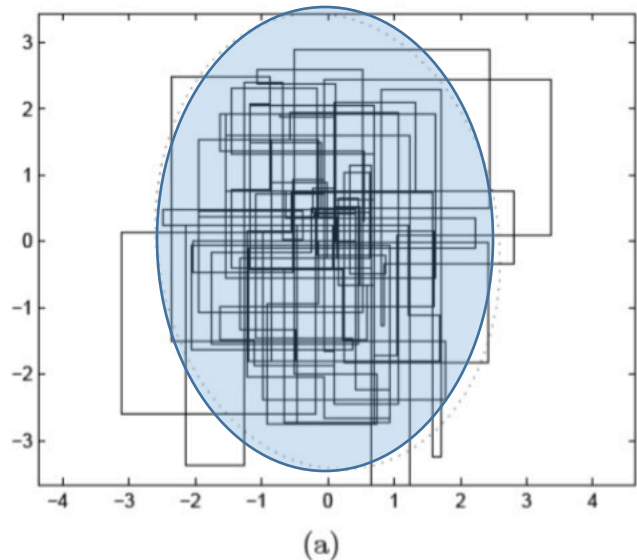
The Markov Chain $q(x^{l+1} | x^l)$ needs to be able to traverse the entire state-space (no matter where we start)

- ▶ This property is called **irreducible**
- ▶ Then $p(x)$ is the stationary distribution of $q(x' | x)$

¹Reference: Bjoern Andres and Bernt Schiele, MPI (2016)

²Reference: https://en.wikipedia.org/wiki/Gibbs_sampling

Gibbs Sampling Iterations



Gibbs sampling is more efficient if states are not correlated

- ▶ Left: Almost isotropic Gaussian
- ▶ Right: correlated Gaussian

¹Reference: Bjoern Andres and Bernt Schiele, MPI (2016)

²Reference: https://en.wikipedia.org/wiki/Gibbs_sampling

Markov Chain Analysis

Sample from a multi-variate distribution

$$p(x) = \frac{1}{Z} p^*(x)$$

with Z intractable to calculate

Idea: Sample from some $q(\mathbf{x} \rightarrow \mathbf{x}')$ with a **stationary distribution**

$$\pi(\mathbf{x}') = \sum_{\mathbf{x}} \pi(\mathbf{x}) q(\mathbf{x} \rightarrow \mathbf{x}') \quad \text{for all } \mathbf{x}'$$

Given $p(x)$ find $q(\mathbf{x} \rightarrow \mathbf{x}')$ such that $\pi(\mathbf{x}) = p(x)$

Gibbs sampling is one instance (that is why it is working)

¹Reference: Bjoern Andres and Bernt Schiele, MPI (2016)

²Reference: https://en.wikipedia.org/wiki/Metropolis%E2%80%93Hastings_algorithm

Markov Chain Terminology

Transition probability $q(\mathbf{x} \rightarrow \mathbf{x}')$

Occupancy probability $\pi_t(\mathbf{x})$ at time t

Equilibrium condition on π_t defines stationary distribution $\pi(\mathbf{x})$

Note: stationary distribution depends on choice of $q(\mathbf{x} \rightarrow \mathbf{x}')$

Pairwise **detailed balance** on states guarantees equilibrium

Gibbs sampling transition probability:

sample each variable given current values of all others

\Rightarrow detailed balance with the true posterior

For Bayesian networks, Gibbs sampling reduces to sampling conditioned on each variable's Markov blanket

Stationary Distribution of a MC

$\pi_t(\mathbf{x})$ = probability in state \mathbf{x} at time t

$\pi_{t+1}(\mathbf{x}')$ = probability in state \mathbf{x}' at time $t + 1$

π_{t+1} in terms of π_t and $q(\mathbf{x} \rightarrow \mathbf{x}')$

$$\pi_{t+1}(\mathbf{x}') = \sum_{\mathbf{x}} \pi_t(\mathbf{x}) q(\mathbf{x} \rightarrow \mathbf{x}')$$

Stationary distribution: $\pi_t = \pi_{t+1} = \pi$

$$\pi(\mathbf{x}') = \sum_{\mathbf{x}} \pi(\mathbf{x}) q(\mathbf{x} \rightarrow \mathbf{x}') \quad \text{for all } \mathbf{x}'$$

If π exists, it is unique (specific to $q(\mathbf{x} \rightarrow \mathbf{x}')$)

In equilibrium, expected “outflow” = expected “inflow”

Detailed Balance Equation

“Outflow” = “inflow” for each pair of states:

$$\pi(\mathbf{x})q(\mathbf{x} \rightarrow \mathbf{x}') = \pi(\mathbf{x}')q(\mathbf{x}' \rightarrow \mathbf{x}) \quad \text{for all } \mathbf{x}, \mathbf{x}'$$

Detailed balance \Rightarrow stationarity:

$$\begin{aligned}\sum_{\mathbf{x}} \pi(\mathbf{x})q(\mathbf{x} \rightarrow \mathbf{x}') &= \sum_{\mathbf{x}} \pi(\mathbf{x}')q(\mathbf{x}' \rightarrow \mathbf{x}) \\ &= \pi(\mathbf{x}') \sum_{\mathbf{x}} q(\mathbf{x}' \rightarrow \mathbf{x}) \\ &= \pi(\mathbf{x}')\end{aligned}$$

MCMC algorithms typically constructed by designing a transition probability q that is in detailed balance with desired π

Gibbs Satisfies Detailed Balance

Sample each variable in turn, given **all other variables**

Sampling X_i , let $\bar{\mathbf{X}}_i$ be all other nonevidence variables

Current values are x_i and $\bar{\mathbf{x}}_i$; \mathbf{e} is fixed

Transition probability is given by

$$q(\mathbf{x} \rightarrow \mathbf{x}') = q(x_i, \bar{\mathbf{x}}_i \rightarrow x'_i, \bar{\mathbf{x}}_i) = P(x'_i | \bar{\mathbf{x}}_i, \mathbf{e})$$

This gives detailed balance with true posterior $P(\mathbf{x} | \mathbf{e})$:

$$\begin{aligned} \pi(\mathbf{x})q(\mathbf{x} \rightarrow \mathbf{x}') &= P(\mathbf{x} | \mathbf{e})P(x'_i | \bar{\mathbf{x}}_i, \mathbf{e}) = P(x_i, \bar{\mathbf{x}}_i | \mathbf{e})P(x'_i | \bar{\mathbf{x}}_i, \mathbf{e}) \\ &= P(x_i | \bar{\mathbf{x}}_i, \mathbf{e})P(\bar{\mathbf{x}}_i | \mathbf{e})P(x'_i | \bar{\mathbf{x}}_i, \mathbf{e}) \quad (\text{chain rule}) \\ &= P(x_i | \bar{\mathbf{x}}_i, \mathbf{e})P(x'_i, \bar{\mathbf{x}}_i | \mathbf{e}) \quad (\text{chain rule backwards}) \\ &= q(\mathbf{x}' \rightarrow \mathbf{x})\pi(\mathbf{x}') = \pi(\mathbf{x}')q(\mathbf{x}' \rightarrow \mathbf{x}) \end{aligned}$$

Metropolis-Hasting MCMC

- We will now mention one other MCMC method in passing.
 - Metropolis-Hasting (MH)
 - A special case is called Metropolis sampling.

¹Reference: Bjoern Andres and Bernt Schiele, MPI (2016)

²Reference: https://en.wikipedia.org/wiki/Metropolis%E2%80%93Hastings_algorithm

MH MCMC Algorithm

Slightly more general MCMC method when the proposal distribution is *not* symmetric

Sample x' and accept with probability

$$A(x', x) = \min \left(1, \frac{\tilde{q}(x | x') p^*(x')}{\tilde{q}(x' | x) p^*(x)} \right)$$

Note: when the proposal distribution is symmetric,
Metropolis-Hastings reduces to standard Metropolis sampling

¹Reference: Bjoern Andres and Bernt Schiele, MPI (2016)

²Reference: https://en.wikipedia.org/wiki/Metropolis%E2%80%93Hastings_algorithm

MH MCMC Special Case: Metropolis Sampling

Special case of MCMC method (proposal distribution) with the following proposal distribution

- ▶ symmetric: $q(x' | x) = q(x | x')$

Sample x' and accept with probability

$$A(x', x) = \min \left(1, \frac{p^*(x')}{p^*(x)} \right) \in [0, 1]$$

- ▶ If new state x' is more probable always accept
- ▶ If new state is less probable accept with $\frac{p^*(x')}{p^*(x)}$

¹Reference: Bjoern Andres and Bernt Schiele, MPI (2016)

²Reference: https://en.wikipedia.org/wiki/Metropolis%E2%80%93Hastings_algorithm

Questions?

Summary

- Inference computations on joint distributions is a hard problem
- Graphical models help us do this in efficient ways
 - For tree models, this is linear time!
- We discussed two exact methods
 - Variable Elimination
 - Belief propagation
- We discussed one family of approximate methods
 - Based on **sampling** via Markov Chain Monte Carlo techniques

Appendix

Sample Exam Questions

- What is a factor graph? How is it related to DPGMs? How is it related to UPGMs?
- What are the key steps of Belief propagation?
- What is the use of BP? Can it be used for inference over general factor graphs?
- How would one use sampling for inference?
- Why is Gibbs sampling a MCMC technique?
- Why does BP do better than variable elimination?

Advanced Prediction Models

Deep Learning, Graphical Models and Reinforcement
Learning

Recap: Why Graphical Models

- We have seen deep learning techniques for unstructured data
 - Predominantly vision and text/audio
 - We will see control in the last part of the course
 - (Reinforcement Learning)
- For structured data, graphical models are the most versatile framework
 - Successfully applications:
 - Kalman filtering in engineering
 - Decoding in cell phones (channel codes)
 - Hidden Markov models for time series
 - Clustering, regression, classification ...

Recap: Graphical Models Landscape

- Three key parts:
 - Representation
 - Capture uncertainty (joint distribution)
 - Capture **conditional independences** (metadata)
 - Visualization of metadata for a distribution
 - Inference
 - Efficient methods for computing marginal or conditional distributions **quickly**
 - Learning
 - Learning the **parameters of the distribution** can deal with prior knowledge and missing data

Today's Outline

- Applications
- Learning
 - DPGM/UPGM
 - Parameter Estimation
 - Structure Estimation
 - Complete/Incomplete Data

Applications

Applications of Graphical Models

- Given all that we have learned up to now, we will sample the following applications

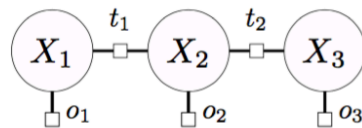
Hidden Markov Models	time series, tracking
Gaussian Mixture Models	clustering
Latent Dirichlet Allocation	topic modeling
Conditional Random Fields	structured classification/regression

Example Graphical Model I



Problem: person tracking

Sensors reports positions: 0, 2, 2. Objects don't move very fast and sensors are a bit noisy. What path did the person take?

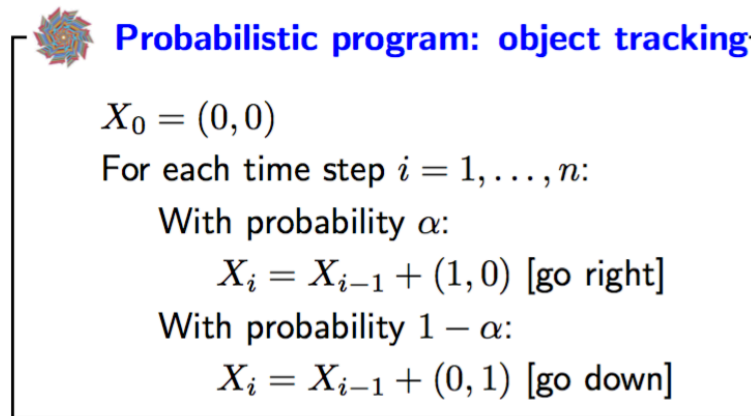


- Variables X_i : location of object at position i
- Transition factors $t_i(x_i, x_{i+1})$: incorporate physics
- Observation factors $o_i(x_i)$: incorporate sensors

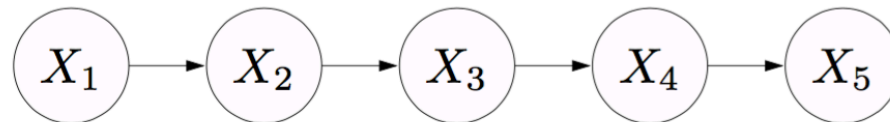
Example Graphical Model II

- A generative process is nothing but a description of the joint distribution in terms of how the random variables realize

Probabilistic program:



Bayesian network:



Mathematical definition:

$$p(x_i \mid x_{i-1}) = \alpha \cdot \underbrace{[x_i = x_{i-1} + (1, 0)]}_{\text{right}} + (1 - \alpha) \cdot \underbrace{[x_i = x_{i-1} + (0, 1)]}_{\text{down}}$$

Example Graphical Model III

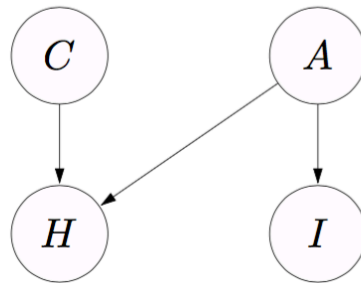


Problem: cold or allergies?

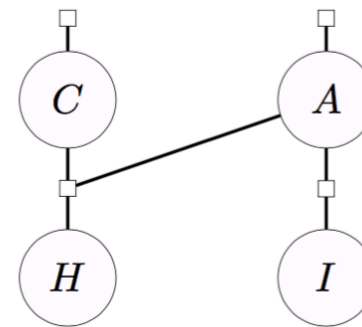
You are coughing and have itchy eyes. What do you have?

Variables: **C**old, **A**llergy, Cough, Itchy eyes

Bayesian network:

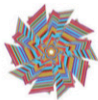
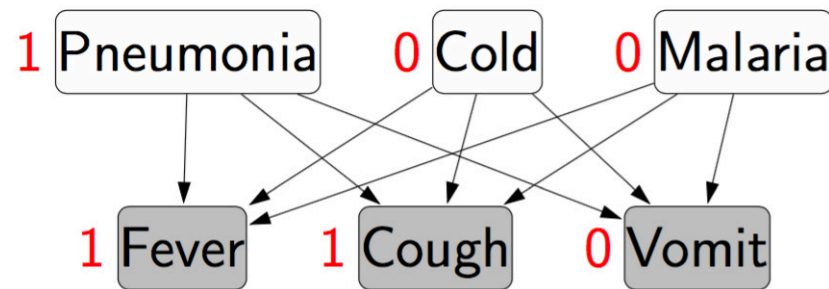


Factor graph:



Example Graphical Model IV

Question: If patient has a cough and fever, what disease(s) does he/she have?



Probabilistic program: diseases and symptoms

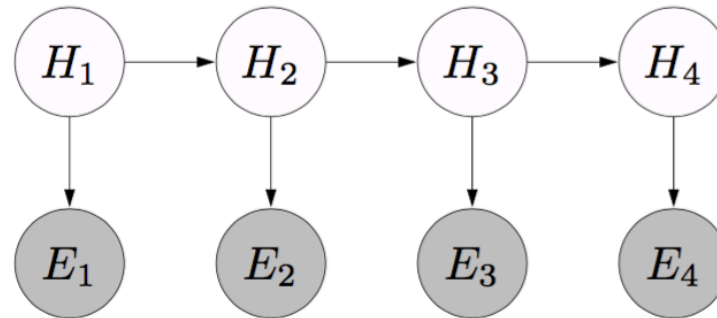
For each disease $i = 1, \dots, m$:

Generate activity of disease $D_i \sim p(D_i)$

For each symptom $j = 1, \dots, n$:

Generate activity of symptom $S_j \sim p(S_j \mid D_{1:m})$

Object Tracking via Hidden Markov Model



Problem: object tracking

$H_i \in \{1, \dots, K\}$: location of object at time step i

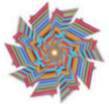
$E_i \in \{1, \dots, K\}$: sensor reading at time step i

Start: $p(h_1)$: uniform over all locations

Transition $p(h_i | h_{i-1})$: uniform over adjacent loc.

Emission $p(e_i | h_i)$: uniform over adjacent loc.

Generative Program for HMM

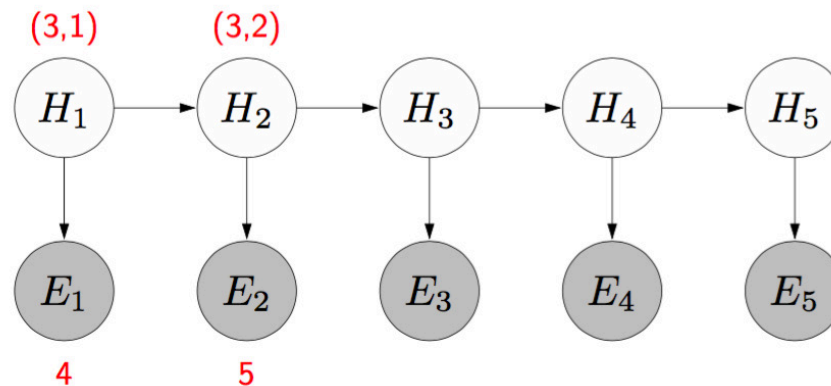


Probabilistic program: hidden Markov model (HMM)

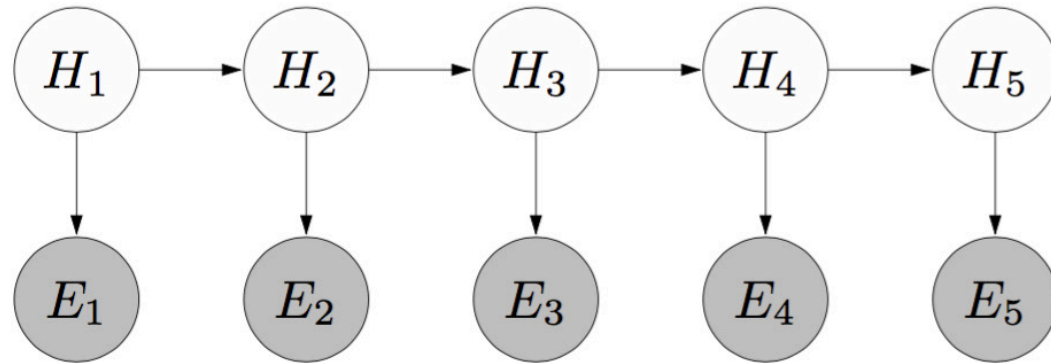
For each time step $t = 1, \dots, T$:

Generate object location $H_t \sim p(H_t \mid H_{t-1})$

Generate sensor reading $E_t \sim p(E_t \mid H_t)$



Object Tracking via HMM



$$\mathbb{P}(H = h, E = e) = \underbrace{p(h_1)}_{\text{start}} \prod_{i=2}^n \underbrace{p(h_i | h_{i-1})}_{\text{transition}} \prod_{i=1}^n \underbrace{p(e_i | h_i)}_{\text{emission}}$$

Query (**filtering**):

$$\mathbb{P}(H_3 \mid E_1 = e_1, E_2 = e_2, E_3 = e_3)$$

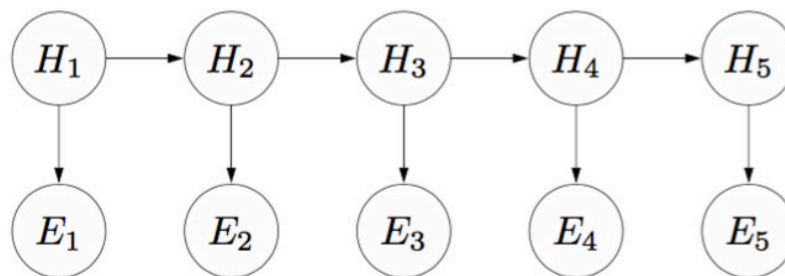
Query (**smoothing**):

$$\mathbb{P}(H_3 \mid E_1 = e_1, E_2 = e_2, E_3 = e_3, E_4 = e_4, E_5 = e_5)$$

HMM Parameter Sharing

Variables:

- H_1, \dots, H_n (e.g., actual positions)
- E_1, \dots, E_n (e.g., sensor readings)



$$\mathbb{P}(H = h, E = e) = \prod_{i=1}^n p_{\text{trans}}(h_i \mid h_{i-1}) p_{\text{emit}}(e_i \mid h_i)$$

Parameters: $\theta = (p_{\text{trans}}, p_{\text{emit}})$

$\mathcal{D}_{\text{train}}$ is a set of full assignments to (H, E)

Mixture Models

- “Standard” distributions (e.g., multivariate Gaussian) are too limited
- How do we represent and learn more complex ones?
- One answer: Mixtures of “standard” distributions
- In the limit, can approximate any distribution this way
- Also good (and widely used) as a clustering method

Gaussian Mixture Models

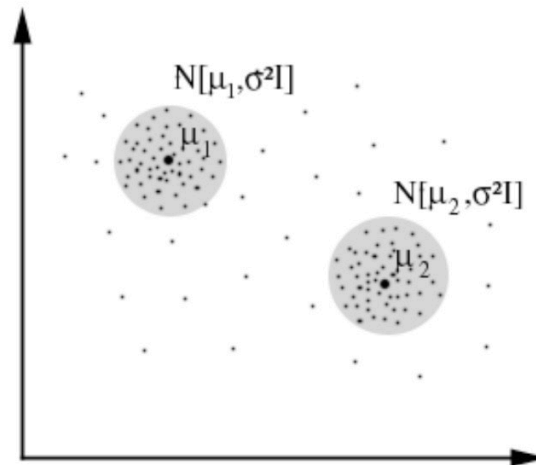
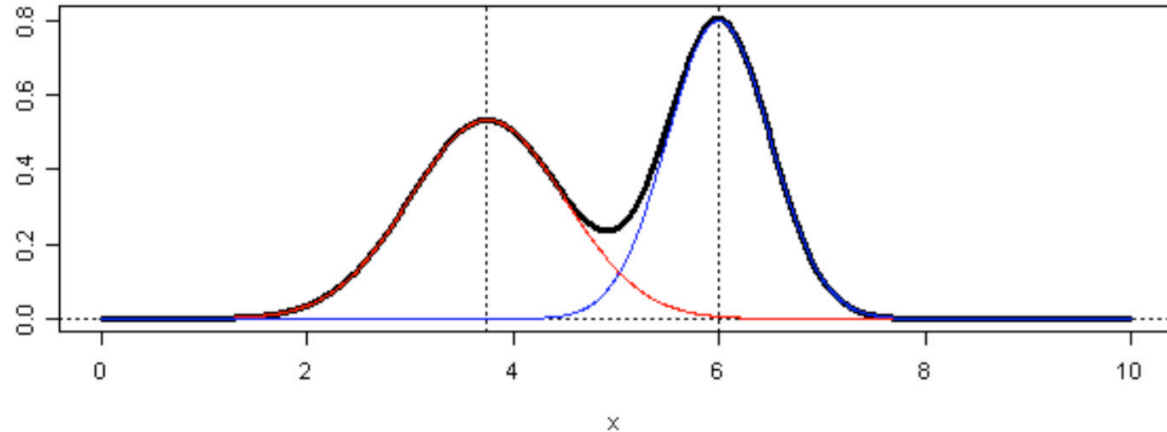
- The N -dim. multivariate normal distribution, $\mathcal{N}(\mu, \Sigma)$, has density:

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{N/2} |\Sigma|^{1/2}} \exp \left(-\frac{1}{2} (\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu) \right)$$

- Suppose we have k Gaussians given by μ_k and Σ_k , and a distribution θ over the numbers $1, \dots, k$
- Mixture of Gaussians distribution $p(y, \mathbf{x})$ given by
 - 1 Sample $y \sim \theta$ (specifies which Gaussian to use)
 - 2 Sample $\mathbf{x} \sim \mathcal{N}(\mu_y, \Sigma_y)$

Gaussian Mixture Model in 1D and 2D

- The marginal distribution over \mathbf{x} looks like:



Learning a GMM

Initialize parameters ignoring missing information

Repeat until convergence:

E step: Compute expected values of unobserved variables, assuming current parameter values

M step: Compute new parameter values to maximize probability of data (observed & estimated)

(Also: Initialize expected values ignoring missing info)

Learning a 1D GMM

Initialization: Choose means at random, etc.

E step: For all examples x_k :

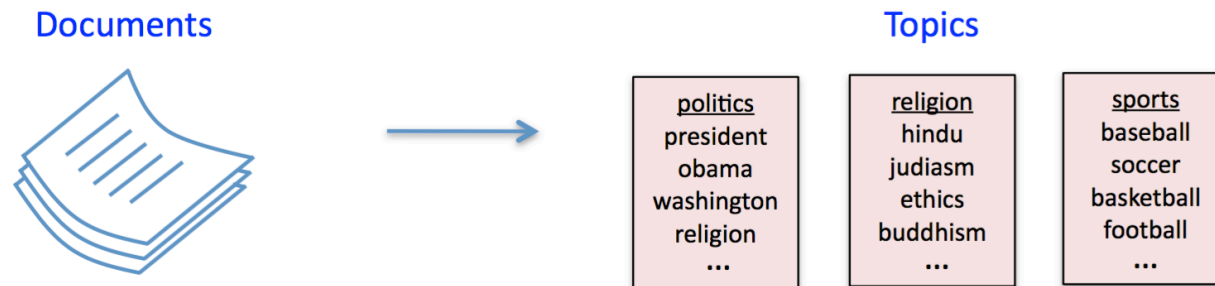
$$P(\mu_i | x_k) = \frac{P(\mu_i)P(x_k | \mu_i)}{P(x_k)} = \frac{P(\mu_i)P(x_k | \mu_i)}{\sum_{i'} P(\mu_{i'})P(x_k | \mu_{i'})}$$

M step: For all components c_i :

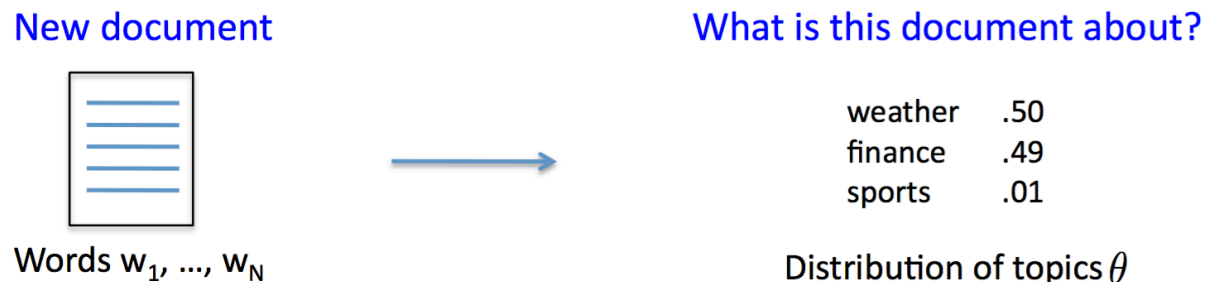
$$\begin{aligned} P(c_i) &= \frac{1}{n_e} \sum_{k=1}^{n_e} P(\mu_i | x_k) \\ \mu_i &= \frac{\sum_{k=1}^{n_e} x_k P(\mu_i | x_k)}{\sum_{k=1}^{n_e} P(\mu_i | x_k)} \\ \sigma_i^2 &= \frac{\sum_{k=1}^{n_e} (x_k - \mu_i)^2 P(\mu_i | x_k)}{\sum_{k=1}^{n_e} P(\mu_i | x_k)} \end{aligned}$$

Latent Dirichlet Allocation

- **Topic models** are powerful tools for exploring large data sets and for making inferences about the content of documents



- Many applications in information retrieval, document summarization, and classification



- LDA is one of the simplest and most widely used topic models

Latent Dirichlet Allocation

- 1 Sample the document's **topic distribution** θ (aka topic vector)

$$\theta \sim \text{Dirichlet}(\alpha_{1:T})$$

where the $\{\alpha_t\}_{t=1}^T$ are fixed hyperparameters. Thus θ is a distribution over T topics with mean $\theta_t = \alpha_t / \sum_{t'} \alpha_{t'}$

- 2 For $i = 1$ to N , sample the **topic** z_i of the i 'th word

$$z_i | \theta \sim \theta$$

- 3 ... and then sample the actual **word** w_i from the z_i 'th topic

$$w_i | z_i \sim \beta_{z_i}$$

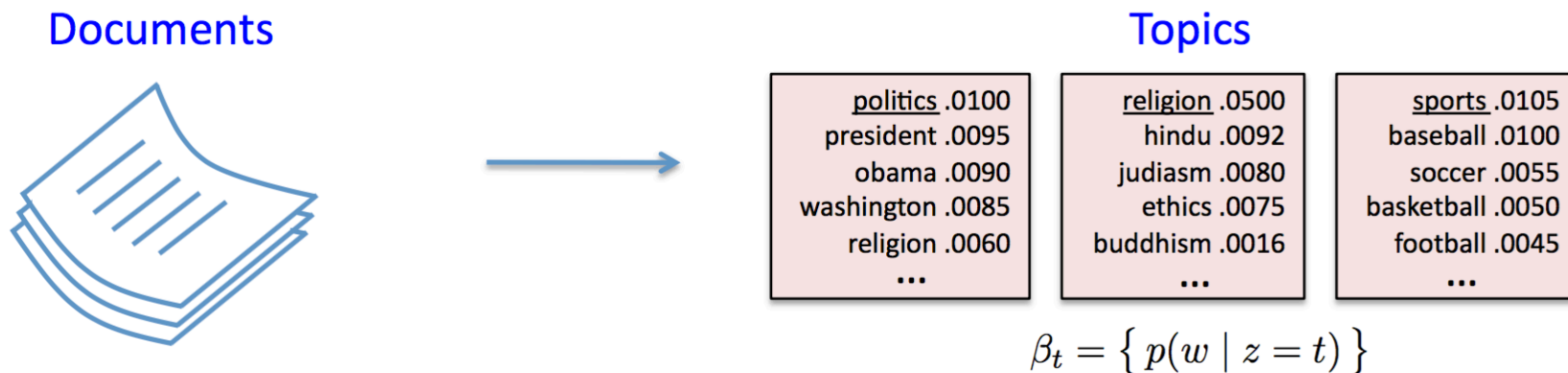
where $\{\beta_t\}_{t=1}^T$ are the *topics* (a fixed collection of distributions on words)

Latent Dirichlet Allocation

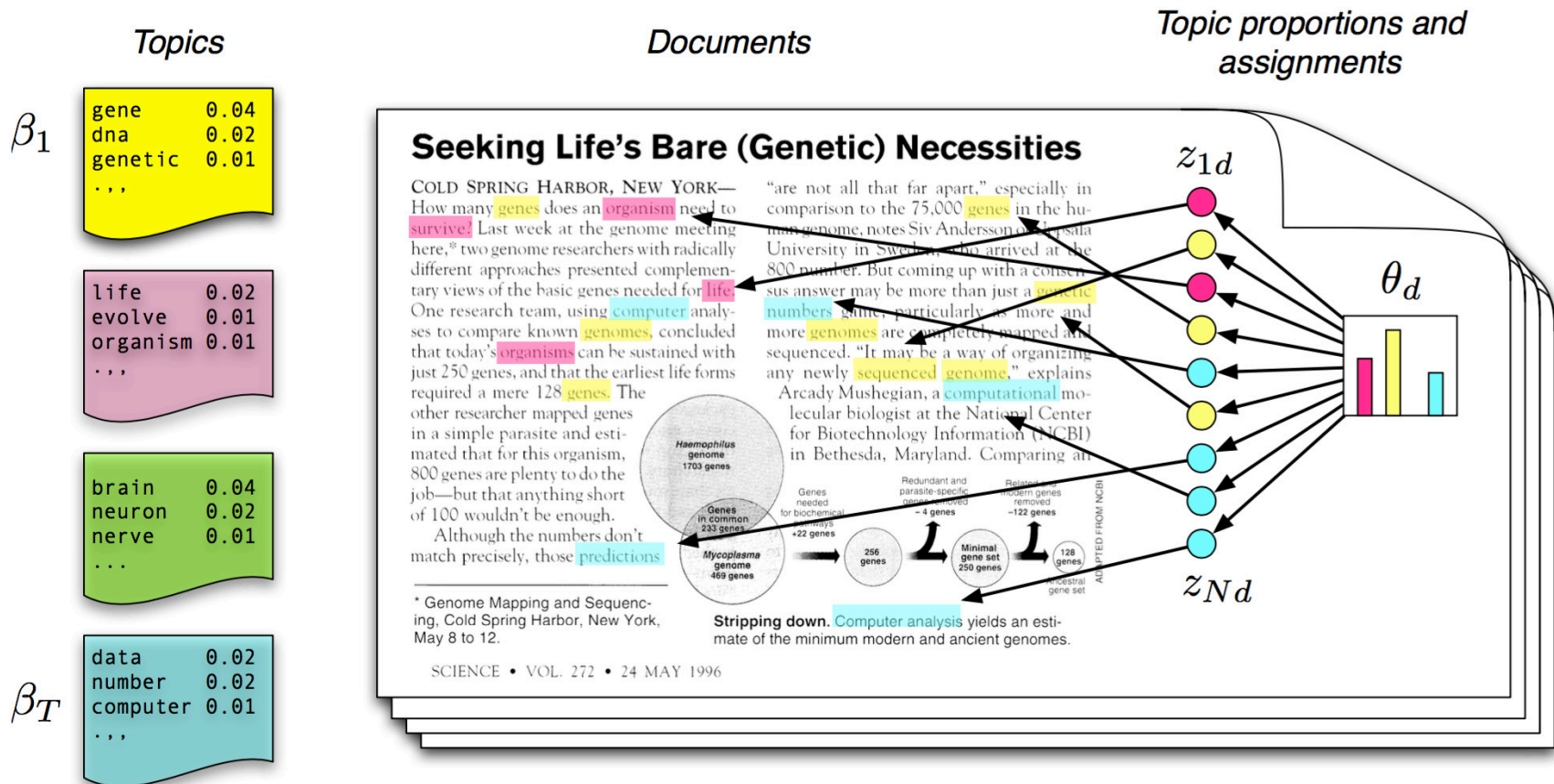
... and then sample the actual **word** w_i from the z_i 'th topic

$$w_i | z_i \sim \beta_{z_i}$$

where $\{\beta_t\}_{t=1}^T$ are the *topics* (a fixed collection of distributions on words)

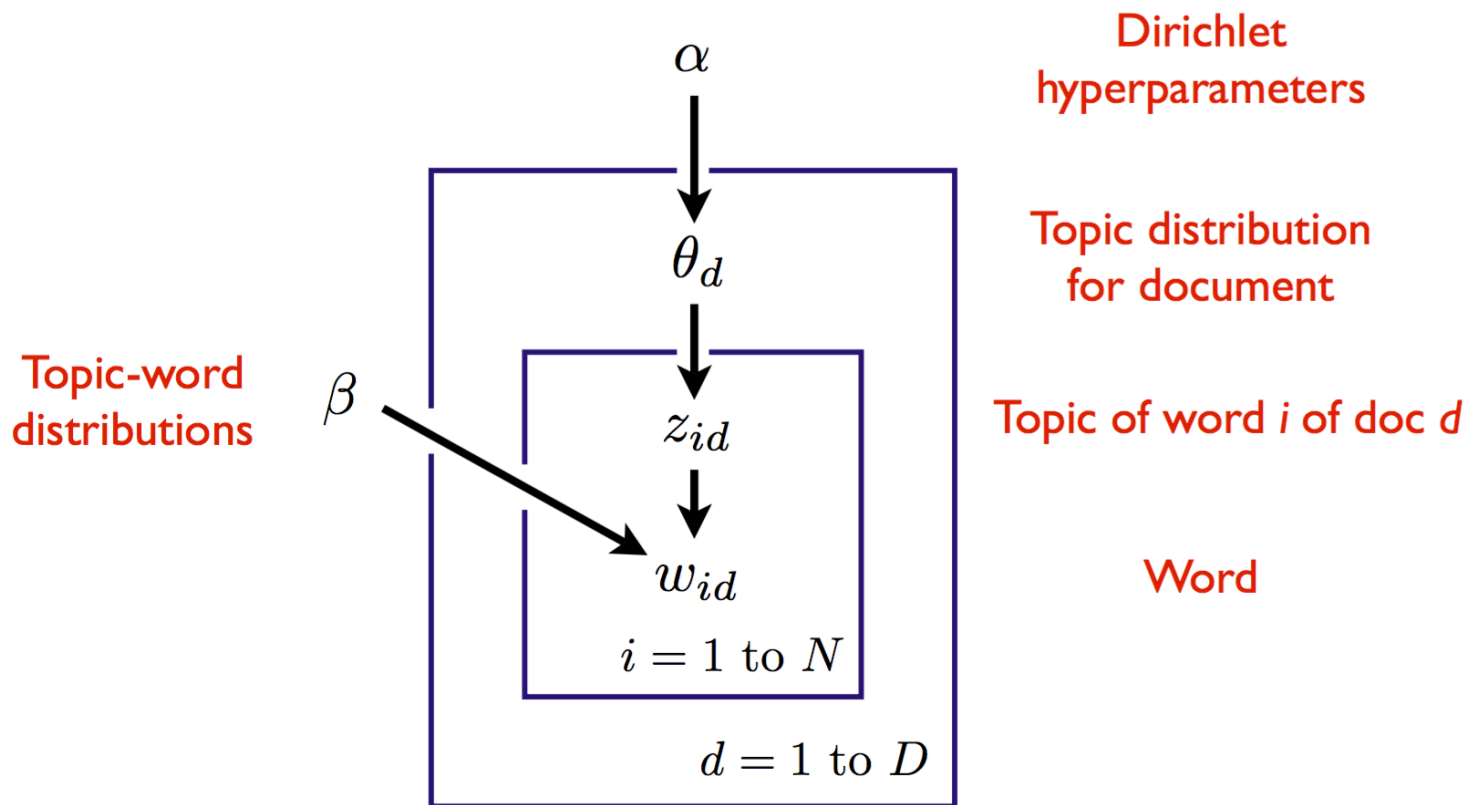


Latent Dirichlet Allocation



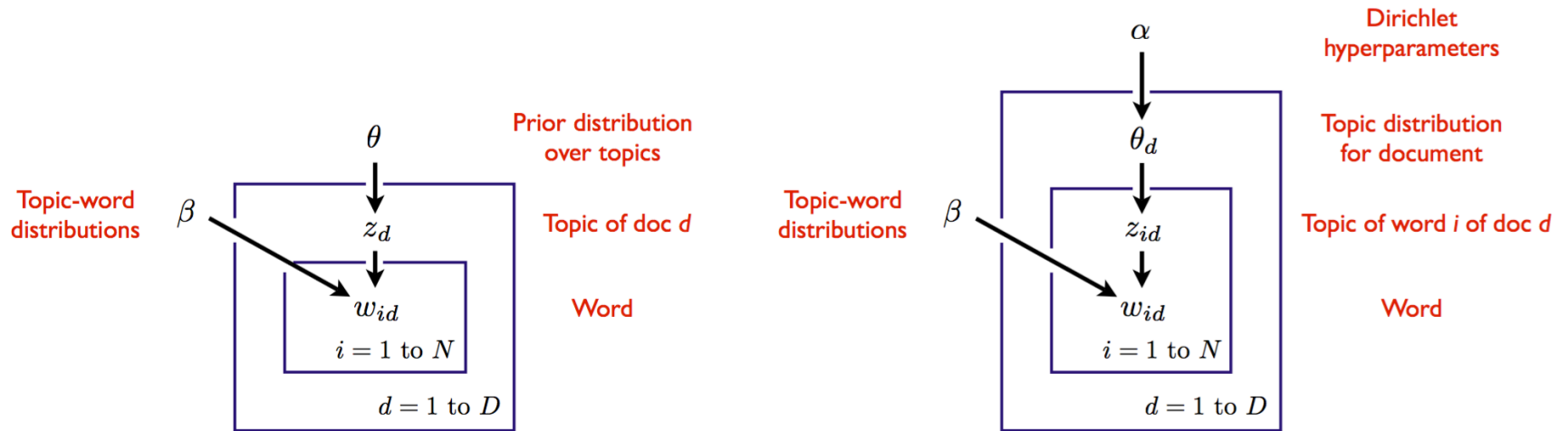
(Blei, *Introduction to Probabilistic Topic Models*, 2011)

Latent Dirichlet Allocation



Variables within a plate are replicated in a conditionally independent manner

Latent Dirichlet Allocation



- Model on left is a **mixture model**
 - Called *multinomial* naive Bayes (a word can appear multiple times)
 - Document is generated from a single topic
- Model on right (LDA) is an **admixture model**
 - Document is generated from a distribution over topics

Conditional Random Field based Classifier

- **Conditional random fields** are undirected graphical models of conditional distributions $p(\mathbf{Y} \mid \mathbf{X})$
 - \mathbf{Y} is a set of **target variables**
 - \mathbf{X} is a set of **observed variables**
- We typically show the graphical model using just the \mathbf{Y} variables
- Potentials are a function of \mathbf{X} and \mathbf{Y}

Conditional Random Field based Classifier

- A CRF is a Markov network on variables $\mathbf{X} \cup \mathbf{Y}$, which specifies the conditional distribution

$$P(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{c \in C} \phi_c(\mathbf{x}_c, \mathbf{y}_c)$$

with partition function

$$Z(\mathbf{x}) = \sum_{\hat{\mathbf{y}}} \prod_{c \in C} \phi_c(\mathbf{x}_c, \hat{\mathbf{y}}_c).$$

- As before, two variables in the graph are connected with an undirected edge if they appear together in the scope of some factor
- The only difference with a standard Markov network is the normalization term – before marginalized over \mathbf{X} and \mathbf{Y} , now only over \mathbf{Y}

CRF for Natural Language Processing: Log-linear Terms

- Factors may depend on a large number of variables
- We typically parameterize each factor as a log-linear function,

$$\phi_c(\mathbf{x}_c, \mathbf{y}_c) = \exp\{\mathbf{w} \cdot \mathbf{f}_c(\mathbf{x}_c, \mathbf{y}_c)\}$$

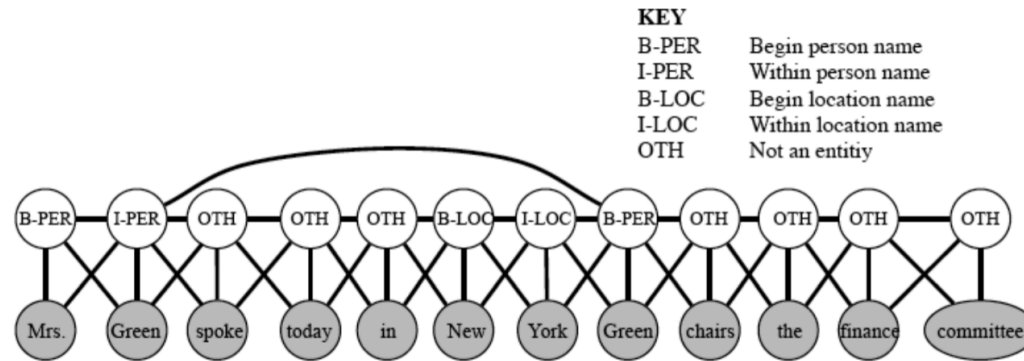
- $\mathbf{f}_c(\mathbf{x}_c, \mathbf{y}_c)$ is a feature vector
- \mathbf{w} is a weight vector which is typically learned – we will discuss this extensively in later lectures

CRF for Natural Language Processing: The Task

- Given a sentence, determine the people and organizations involved and the relevant locations:
“Mrs. Green spoke today in New York. Green chairs the finance committee.”
- Entities sometimes span multiple words. Entity of a word not obvious without considering its *context*
- CRF has one variable X_i for each word, and Y_i encodes the possible labels of that word
- The labels are, for example, “B-person, I-person, B-location, I-location, B-organization, I-organization”
 - Having beginning (B) and within (I) allows the model to segment adjacent entities

CRF for Natural Language Processing: The Task

The graphical model looks like (called a *skip-chain CRF*):



There are three types of potentials:

- $\phi^1(Y_t, Y_{t+1})$ represents dependencies between neighboring target variables [analogous to transition distribution in a HMM]
- $\phi^2(Y_t, Y_{t'})$ for all pairs t, t' such that $x_t = x_{t'}$, because if a word appears twice, it is likely to be the same entity
- $\phi^3(Y_t, X_1, \dots, X_T)$ for dependencies between an entity and the word sequence [e.g., may have features taking into consideration capitalization]

Notice that the graph structure changes depending on the sentence!

Questions?

Today's Outline

- Applications
- Learning
 - Parameter Estimation in DPGMs with Complete/Incomplete Data
 - Structure Estimation in DPGMs
 - Parameter Estimation in UPGMs with Complete/Incomplete Data

Estimation/Learning

Different Estimation/Learning Problems

- There are many variants

Model	DPGM	UPGM
Data	Complete	Incomplete
Structure	Known	Unknown
Objective	Generative	Discriminative

Different Estimation/Learning Problems

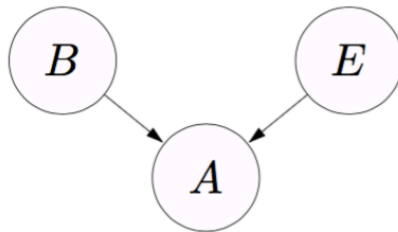
- We will look at the following problems
 - Learning DPGMs with complete data and known structure
 - MLE via counting and normalizing
 - Learning DPGMs with incomplete data and known structure
 - EM
 - Learning DPGM structure
 - Learning UPGMs in a generative setting
 - Learning UPGM in a discriminative setting

Different Estimation/Learning Problems

- There are many variants

Model	DPGM	UPGM
Data	Complete	Incomplete
Structure	Known	Unknown
Objective	Generative	Discriminative

Learning in DPGM: Parameters



b	$p(b)$
1	?
0	?

e	$p(e)$
1	?
0	?

b	e	a	$p(a \mid b, e)$
0	0	0	?
0	0	1	?
0	1	0	?
0	1	1	?
1	0	0	?
1	0	1	?
1	1	0	?
1	1	1	?

Learning in DPGM: Parameter Estimation

Training data

$\mathcal{D}_{\text{train}}$ (an example is an assignment to X)



Parameters

θ (local conditional probabilities)

Learning in DPGM: One Variable Example

Setup:

- One variable R representing the rating of a movie $\{1, 2, 3, 4, 5\}$

$$\textcircled{R} \quad \mathbb{P}(R = r) = p(r)$$

Parameters:

$$\theta = (p(1), p(2), p(3), p(4), p(5))$$

Training data:

$$\mathcal{D}_{\text{train}} = \{1, 3, 4, 4, 4, 4, 4, 5, 5, 5\}$$

Learning in DPGM: One Variable Example

Learning:

$$\mathcal{D}_{\text{train}} \Rightarrow \theta$$

Intuition: $p(r) \propto$ number of occurrences of r in $\mathcal{D}_{\text{train}}$

Example:

$$\mathcal{D}_{\text{train}} = \{1, 3, 4, 4, 4, 4, 4, 5, 5, 5\}$$



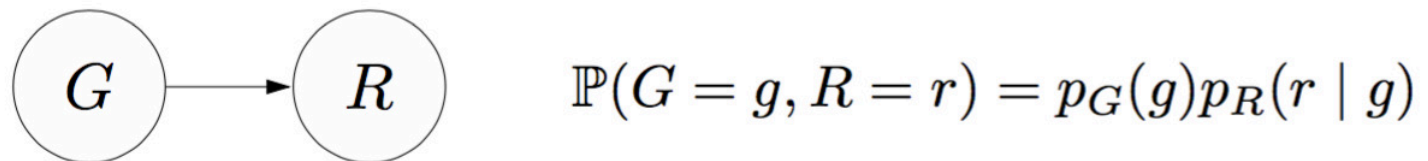
θ :

r	$p(r)$
1	0.1
2	0.0
3	0.1
4	0.5
5	0.3

Learning in DPGM: Two Variables Example

Variables:

- Genre $G \in \{\text{drama}, \text{comedy}\}$
- Rating $R \in \{1, 2, 3, 4, 5\}$



$$\mathcal{D}_{\text{train}} = \{(\text{d}, 4), (\text{d}, 4), (\text{d}, 5), (\text{c}, 1), (\text{c}, 5)\}$$

Parameters: $\theta = (p_G, p_R)$

Learning in DPGM: Two Variables Example

$$\mathcal{D}_{\text{train}} = \{(d, 4), (d, 4), (d, 5), (c, 1), (c, 5)\}$$

Intuitive strategy:

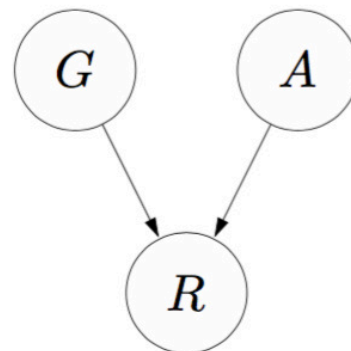
- Estimate each local conditional distribution (p_G and p_R) separately
- For each value of conditioned variable (e.g., g), estimate distribution over values of unconditioned variable (e.g., r)

θ :	g	$p_G(g)$	g	r	$p_R(r g)$
	d	3/5	d	4	2/3
	c	2/5	d	5	1/3
			c	1	1/2
			c	5	1/2

Learning in DPGM: Three Variables Example I

Variables:

- Genre $G \in \{\text{drama}, \text{comedy}\}$
- Won award $A \in \{0, 1\}$
- Rating $R \in \{1, 2, 3, 4, 5\}$



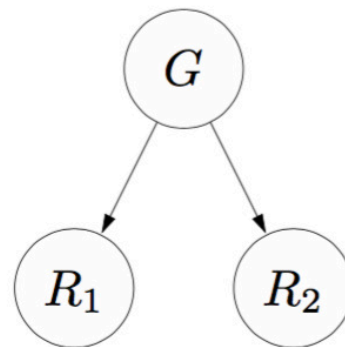
$$\mathbb{P}(G = g, A = a, R = r) = p_G(g)p_A(a)p_R(r \mid g, a)$$

$$\mathcal{D}_{\text{train}} = \{(d, 0, 3), (d, 1, 5), (c, 0, 1), (c, 0, 5), (c, 1, 4)\}$$

Learning in DPGM: Three Variables Example II

Variables:

- Genre $G \in \{\text{drama, comedy}\}$
- Jim's rating $R_1 \in \{1, 2, 3, 4, 5\}$
- Martha's rating $R_2 \in \{1, 2, 3, 4, 5\}$



$$\mathbb{P}(G = g, R_1 = r_1, R_2 = r_2) = p_G(g)p_{R_1}(r_1 \mid g)p_{R_2}(r_2 \mid g)$$

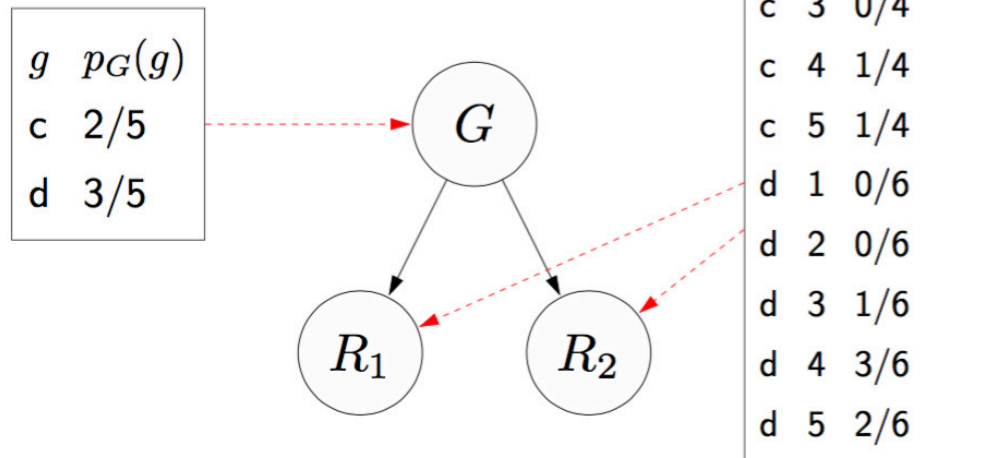
$$\mathcal{D}_{\text{train}} = \{(\text{d}, 4, 5), (\text{d}, 4, 4), (\text{d}, 5, 3), (\text{c}, 1, 2), (\text{c}, 5, 4)\}$$

Learning in DPGM: Parameter Sharing



Key idea: parameter sharing

The local conditional distributions of different variables use the same parameters.



Learning in DPGM: Maximum Likelihood via Counting and Normalizing

Input: training examples $\mathcal{D}_{\text{train}}$ of full assignments

Output: parameters $\theta = \{p_d : d \in D\}$



Algorithm: maximum likelihood for Bayesian networks

Count:

For each $x \in \mathcal{D}_{\text{train}}$:

For each variable x_i :

Increment count $_{d_i}(x_{\text{Parents}(i)}, x_i)$

Normalize:

For each d and local assignment $x_{\text{Parents}(i)}$:

Set $p_d(x_i \mid x_{\text{Parents}(i)}) \propto \text{count}_d(x_{\text{Parents}(i)}, x_i)$

Learning in DPGM: Maximum Likelihood via Counting and Normalizing

Maximum likelihood objective:

$$\max_{\theta} \prod_{x \in \mathcal{D}_{\text{train}}} \mathbb{P}(X = x; \theta)$$

Algorithm on previous slide exactly computes maximum likelihood parameters (closed form solution).

Learning in DPGM: Maximum Likelihood via Counting and Normalizing

$$\mathcal{D}_{\text{train}} = \{(d, 4), (d, 5), (c, 5)\}$$

$$\max_{p_G(\cdot)} (p_G(d)p_G(d)p_G(c)) \max_{p_R(\cdot|c)} p_R(5 | c) \max_{p_R(\cdot|d)} (p_R(4 | d)p_R(5 | d))$$

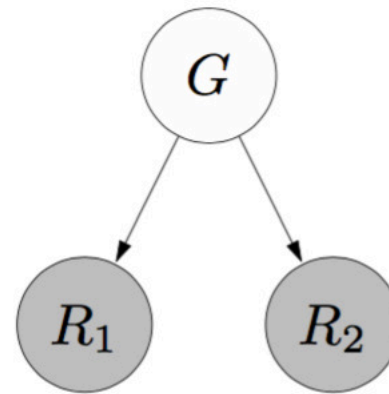
- **Key:** decomposes into subproblems, one for each distribution d and assignment x_{Parents}
- For each subproblem, solve in closed form (Lagrange multipliers for sum-to-1 constraint)

Different Estimation/Learning Problems

- What if we have missing data?

Model	DPGM	UPGM
Data	Complete	Incomplete
Structure	Known	Unknown
Objective	Generative	Discriminative

Learning in DPGM: Latent Variables



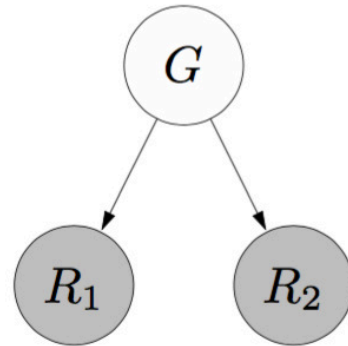
What if we **don't observe** some of the variables?

$$\mathcal{D}_{\text{train}} = \{(\textcolor{red}{?}, 4, 5), (\textcolor{red}{?}, 4, 4), (\textcolor{red}{?}, 5, 3), (\textcolor{red}{?}, 1, 2), (\textcolor{red}{?}, 5, 4)\}$$

DPGM: Maximizing Marginal Likelihood

Variables: H is hidden, $E = e$ is observed

Example:



$$H = G \quad E = (R_1, R_2) \quad e = (4, 5) \\ \theta = (p_G, p_R)$$

Maximum marginal likelihood objective:

$$\begin{aligned} & \max_{\theta} \prod_{e \in \mathcal{D}_{\text{train}}} \mathbb{P}(E = e; \theta) \\ &= \max_{\theta} \prod_{e \in \mathcal{D}_{\text{train}}} \sum_h \mathbb{P}(H = h, E = e; \theta) \end{aligned}$$

Expectation Maximization

Inspiration: K-means

Variables: H is hidden, E is observed (to be e)



Algorithm: Expectation Maximization (EM)

E-step:

- Compute $q(h) = \mathbb{P}(H = h \mid E = e; \theta)$ for each h (use any probabilistic inference algorithm)
- Create weighted points: (h, e) with weight $q(h)$

M-step:

- Compute maximum likelihood (just count and normalize) to get θ

Repeat until convergence.

EM: Revisiting K-Means

- EM tries to maximize marginal likelihood
- K-means
 - Is a special case of EM (for GMMs with variance tending to 0)
 - Objective: Estimate cluster centers

EM: Revisiting K-Means

- EM tries to maximize marginal likelihood
- K-means
 - Is a special case of EM (for GMMs with variance tending to 0)
 - Objective: Estimate cluster centers
 - But don't know which points belong to which clusters
 - Take an alternating optimization approach
 - Find the best cluster assignment given current cluster centers
 - Find the best cluster centers given assignments

The Two Steps of EM

- E-step
 - Here, we don't know what the hidden variables are, so compute their distribution given the current parameters ($P(H|E = e; \theta)$)
 - Need inference! (BP/Gibbs MCMC)
 - This distribution provides a weight $q(h)$ (temp variable in the EM algo) to every value H can take
- Conceptually, the E-step generates a set of weighted full realizations/configurations (h, e) with weights $q(h)$

The Two Steps of EM

- M-step
 - Just do MLE (i.e., counting and normalizing) to re-estimate parameters
- If we repeat E-step and M-step again and again, eventually we will converge to a **local optima** of parameters

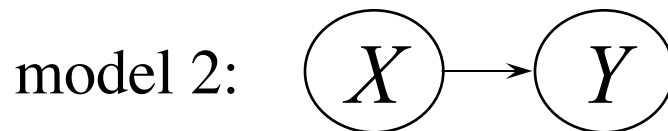
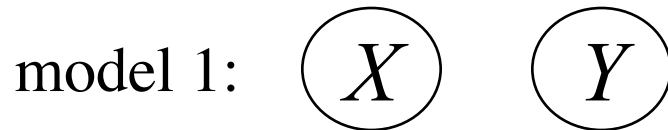
Different Estimation/Learning Problems

- What if the structure is unknown?

Model	DPGM	UPGM
Data	Complete	Incomplete
Structure	Known	Unknown
Objective	Generative	Discriminative

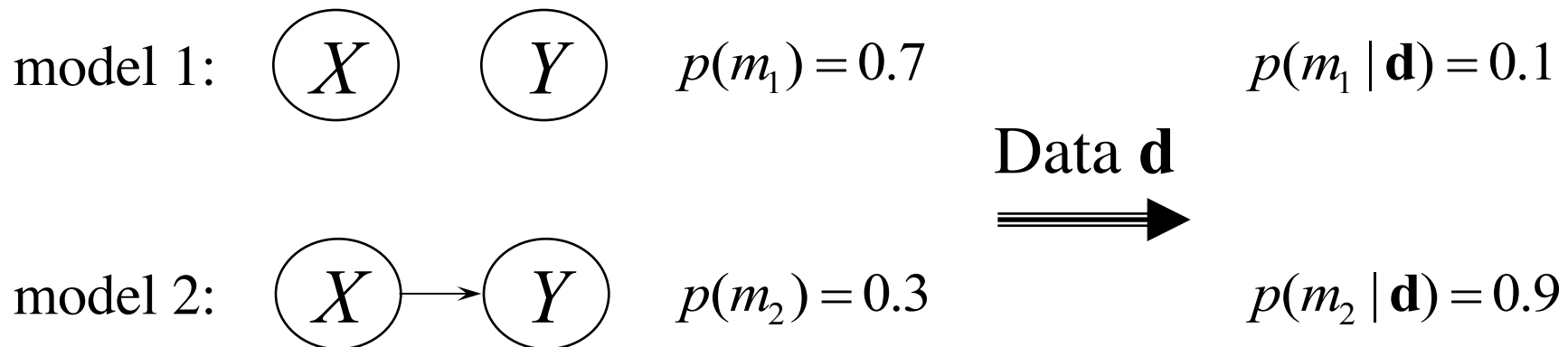
Learning Structure: Bayesian Approach

- Given data, which model is correct?



Learning Structure: Bayesian Approach

- Given data, which model is ~~correct?~~ more likely?



- Can do model averaging
- Can do model selection to pick a model that is
 - tractable, understandable, explainable

Learning Structure: Model Scoring

- Use Baye's theorem to score a model

Given data \mathbf{d} :

model score $\rightsquigarrow p(m | \mathbf{d}) \propto p(m) \underline{p(\mathbf{d} | m)}$

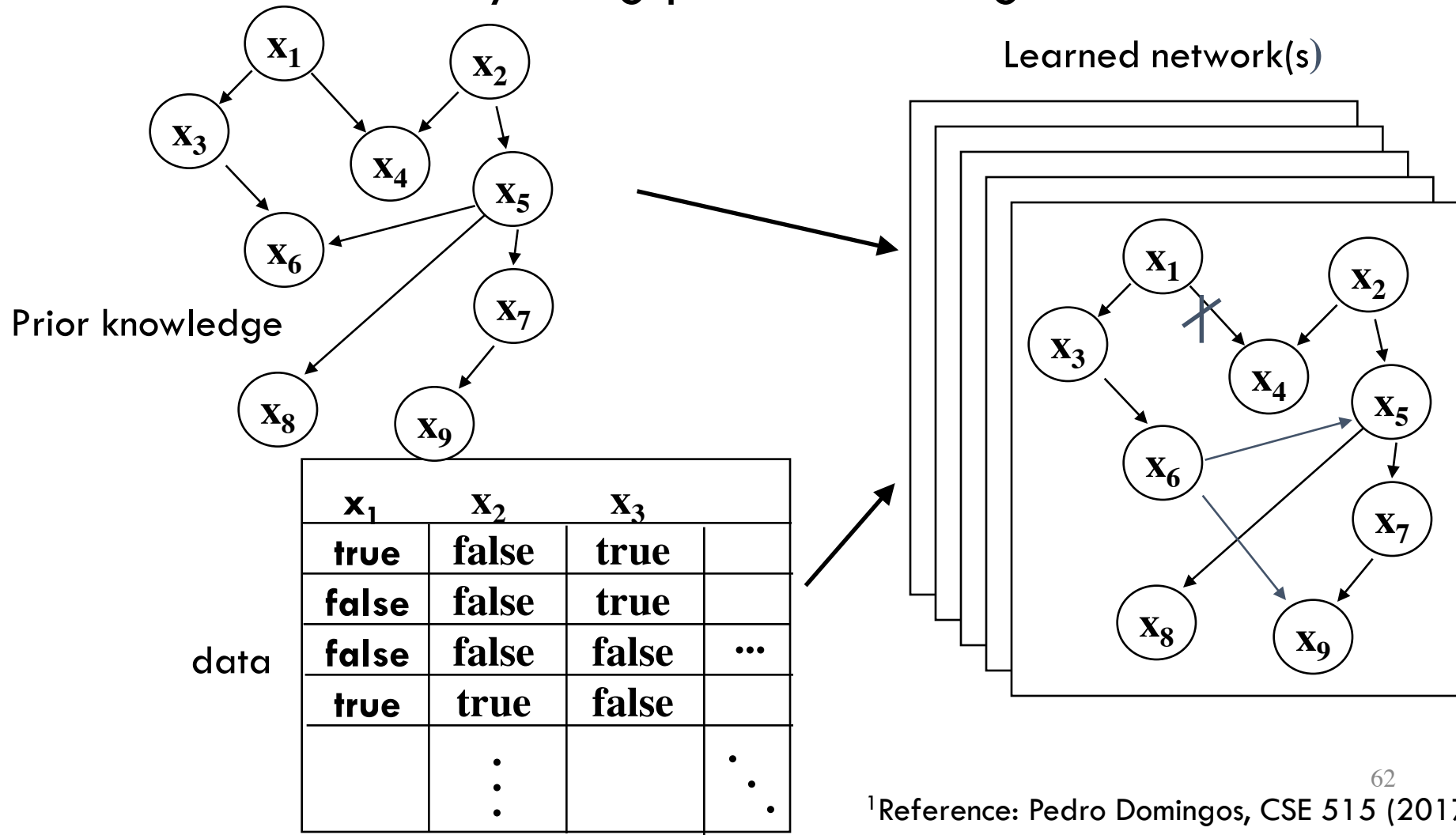
"marginal
likelihood"

likelihood

$\rightsquigarrow p(\mathbf{d} | m) = \int p(\mathbf{d} | \theta, m) p(\theta | m) d\theta$

Combined Learning

- Although structure learning is hard in general, still useful to do it by using prior knowledge and data

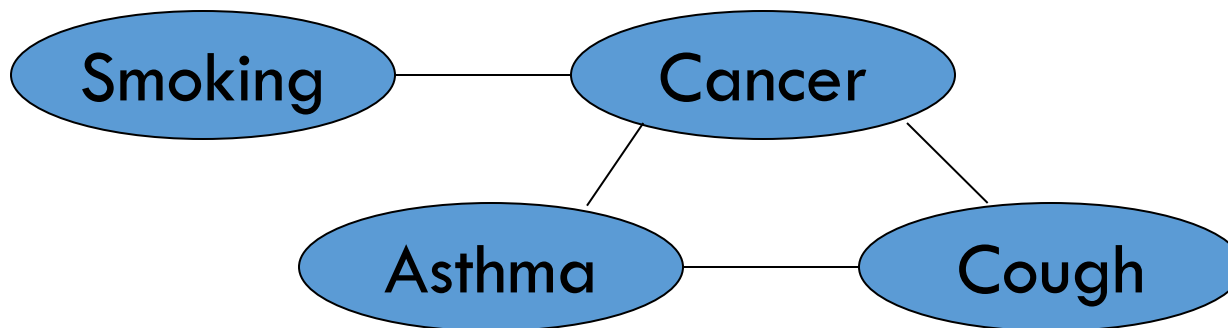


Different Estimation/Learning Problems

- There are many variants

Model	DPGM	UPGM
Data	Complete	Incomplete
Structure	Known	Unknown
Objective	Generative	Discriminative

Learning in UPGM



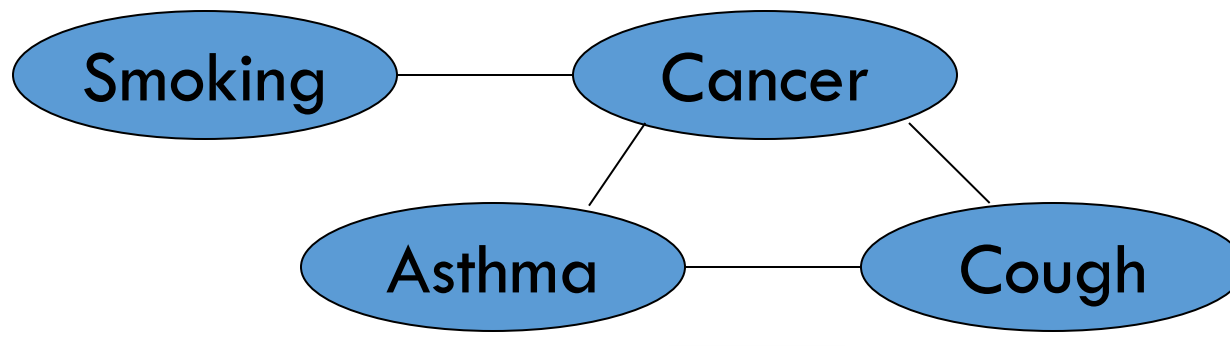
Potential functions defined over cliques

$$P(x) = \frac{1}{Z} \prod_c \Phi_c(x_c)$$

$$Z = \sum_x \prod_c \Phi_c(x_c)$$

Smoking	Cancer	$\Phi(S,C)$
False	False	4.5
False	True	4.5
True	False	2.7
True	True	4.5

Learning in UPGM



Can be thought in terms of a log-linear representation

$$P(x) = \frac{1}{Z} \exp \left(\sum_i w_i f_i(x) \right)$$

Weight of Feature i Feature i

$$f_1(\text{Smoking}, \text{Cancer}) = \begin{cases} 1 & \text{if } \neg \text{Smoking} \vee \text{Cancer} \\ 0 & \text{otherwise} \end{cases}$$

$$w_1 = 0.51$$

Learning in UPGM: Generative

- Maximize likelihood or posterior probability
- Numerical optimization (gradient or 2nd order)

$$\frac{\partial}{\partial w_i} \log P_w(x) = \boxed{n_i(x)} - \boxed{E_w[n_i(x)]}$$

No. of times feature i is true in data

Expected no. times feature i is true according to model

- Requires inference at each step (slow!)

Learning in UPGM: Pseudo-likelihood

$$PL(x) \equiv \prod_i P(x_i \mid \textit{neighbors}(x_i))$$

- Likelihood of each variable given its neighbors in the data
- Does not require inference at each step
- Consistent estimator
- Widely used in vision, spatial statistics, etc.
- But PL parameters may not work well for long inference chains

Different Estimation/Learning Problems

- There are many variants

Model	DPGM	UPGM
Data	Complete	Incomplete
Structure	Known	Unknown
Objective	Generative	Discriminative

Learning in UPGM: Discriminative

- This is related to Conditional Random Fields (CRFs)
- Maximize conditional likelihood of query (y) given evidence (x)

$$\frac{\partial}{\partial w_i} \log P_w(y | x) = \boxed{n_i(x, y)} - \boxed{E_w[n_i(x, y)]}$$

No. of true values of feature i in data

Expected no. of true values according to model

- Inference is easier, but still hard

Different Estimation/Learning Problems

- There are many variants

Model	DPGM	UPGM
Data	Complete	Incomplete
Structure	Known	Unknown
Objective	Generative	Discriminative

Learning in UPGM: Missing Data

- Gradient of likelihood is now difference of expectations

$$\frac{\partial}{\partial w_i} \log P_w(x) = E_w[n_i(y | x)] - E_w[n_i(x, y)]$$

Exp. no. true values given observed data

x : Observed

y : Missing

Expected no. true values given no data

- Can use gradient descent or EM

Learning Summary

- We looked at the following problems
 - Learning DPGMs with complete data and known structure
 - MLE via counting and normalizing
 - Learning DPGMs with incomplete data and known structure
 - EM
 - Learning DPGM structure
 - Learning UPGMs in a generative setting
 - Learning UPGM in a discriminative setting

Learning Summary

- There are many other variants
- Some of these tasks necessarily rely on heuristics
- Many ways have been proposed in research, and as practitioners, we have to pick and choose.

Questions?

Summary

- We discussed some of the applications where they have been successfully applied
- We looked at parameter and structure estimation of these graphical models
- Bottom line: When there is structure in the inputs and outputs of a ML pipeline, consider DPGMs/UPGMs
 - An unified way of thinking about supervised and unsupervised learning

Appendix

Sample Exam Questions

- In which settings would one use MLE and EM for learning in graphical models? Give examples.
- How is the graph structure learned? Can it be specified as prior information?
- Mention 3 applications of graphical models and specify their descriptions. Explain how learning happens in one of these models.

Which is computationally more expensive for Bayesian networks?

probabilistic inference given the parameters

learning the parameters given fully labeled data

Gibbs Sampling when Observations/Evidence are Given

“State” of network = current assignment to all variables.

Generate next state by sampling one variable given Markov blanket

Sample each variable in turn, keeping evidence fixed

```
function GIBBS-SAMPLING( $X, \mathbf{e}, bn, N$ ) returns an estimate of  $P(X|\mathbf{e})$ 
  local variables:  $\mathbf{N}[X]$ , a vector of counts over  $X$ , initially zero
                    $\mathbf{Z}$ , the nonevidence variables in  $bn$ 
                    $\mathbf{x}$ , the current state of the network, initially copied from  $\mathbf{e}$ 

  initialize  $\mathbf{x}$  with random values for the variables in  $\mathbf{Y}$ 
  for  $j = 1$  to  $N$  do
    for each  $Z_i$  in  $\mathbf{Z}$  do
      sample the value of  $Z_i$  in  $\mathbf{x}$  from  $P(Z_i|mb(Z_i))$ 
        given the values of  $MB(Z_i)$  in  $\mathbf{x}$ 
       $\mathbf{N}[x] \leftarrow \mathbf{N}[x] + 1$  where  $x$  is the value of  $X$  in  $\mathbf{x}$ 
  return NORMALIZE( $\mathbf{N}[X]$ )
```

Can also choose a variable to sample at random each time

Additional Applications: Naïve Bayes Spam Filter

- **Key assumption**

Words occur independently of each other given the label of the document

$$p(w_1, \dots, w_n | \text{spam}) = \prod_{i=1}^n p(w_i | \text{spam})$$

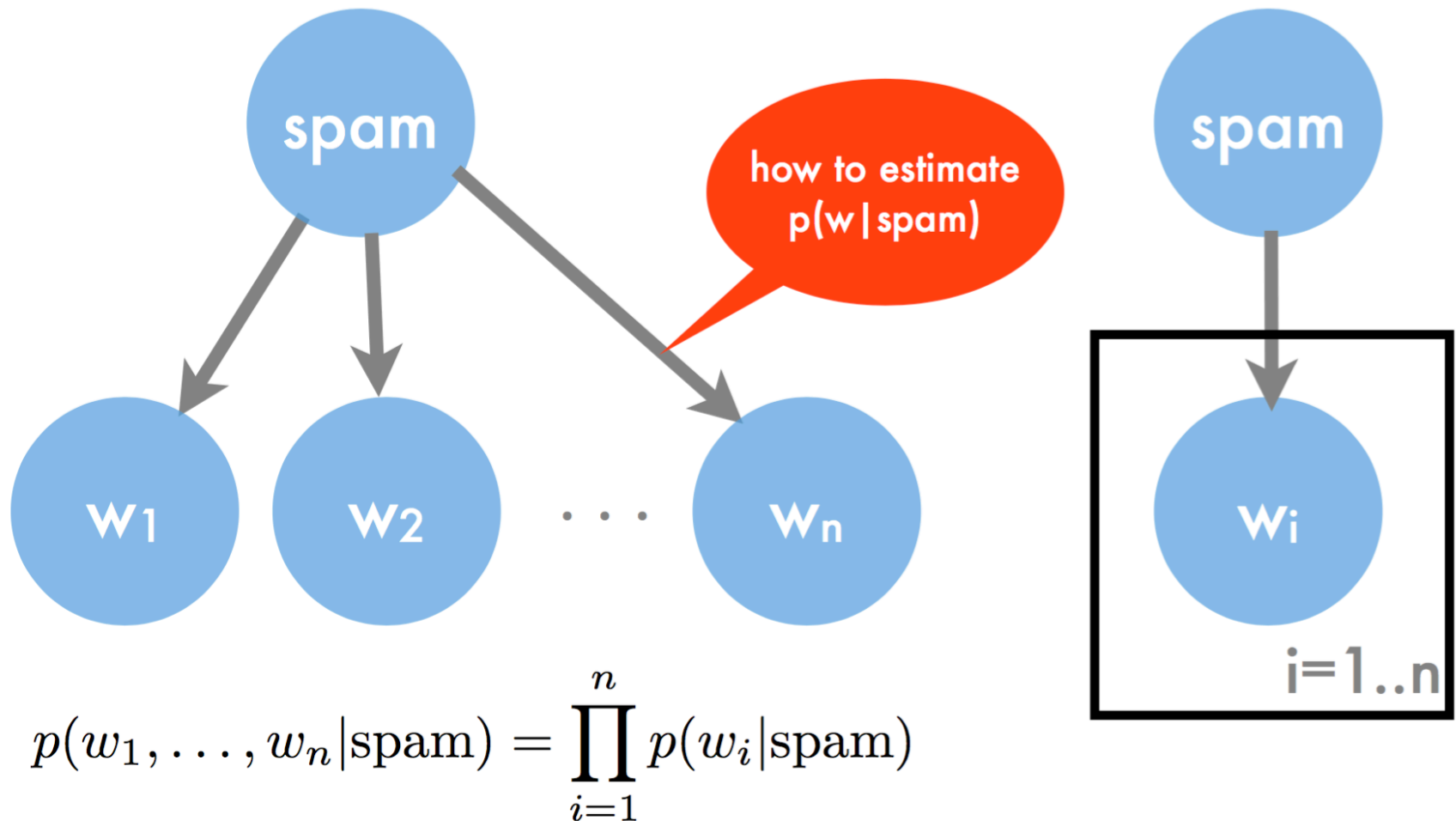
- **Spam classification via Bayes Rule**

$$p(\text{spam} | w_1, \dots, w_n) \propto p(\text{spam}) \prod_{i=1}^n p(w_i | \text{spam})$$

- **Parameter estimation**

Compute spam probability and word distributions for spam and ham

Additional Applications: Naïve Bayes Spam Filter



Additional Applications: Naïve Bayes Spam Filter

- Two classes (spam/ham)
- Binary features (e.g. presence of \$\$\$, viagra)
- Simplistic Algorithm
 - Count occurrences of feature for spam/ham
 - Count number of spam/ham mails

feature probability

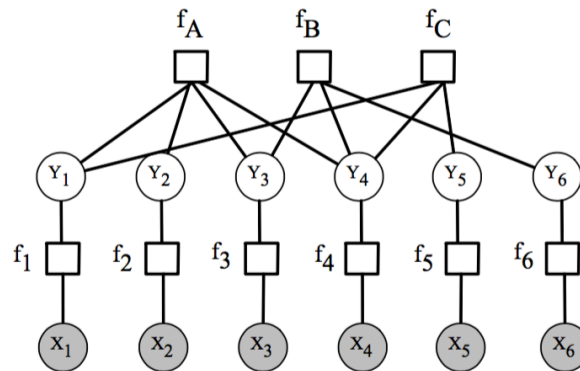
$$p(x_i = \text{TRUE}|y) = \frac{n(i, y)}{n(y)} \text{ and } p(y) = \frac{n(y)}{n}$$

spam probability

$$p(y|x) \propto \frac{n(y)}{n} \prod_{i:x_i=\text{TRUE}} \frac{n(i, y)}{n(y)} \prod_{i:x_i=\text{FALSE}} \frac{n(y) - n(i, y)}{n(y)}$$

Additional Applications: MAP Problem in Low Density Parity Check Codes

- Error correcting codes for transmitting a message over a noisy channel (invented by Gallager in the 1960's, then re-discovered in 1996)



- Each of the top row factors enforce that its variables have even parity:

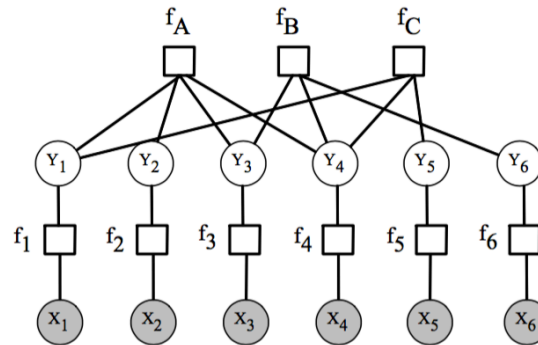
$$f_A(Y_1, Y_2, Y_3, Y_4) = 1 \text{ if } Y_1 \otimes Y_2 \otimes Y_3 \otimes Y_4 = 0, \text{ and } 0 \text{ otherwise}$$

- Thus, the only assignments \mathbf{Y} with non-zero probability are the following (called **codewords**):
3 bits encoded using 6 bits

000000, 011001, 110010, 101011, 111100, 100101, 001110, 010111

- $f_i(Y_i, X_i) = p(X_i | Y_i)$, the likelihood of a bit flip according to noise model

Additional Applications: MAP Problem in Low Density Parity Check Codes



- The *decoding* problem for LDPCs is to find

$$\operatorname{argmax}_{\mathbf{y}} p(\mathbf{y} \mid \mathbf{x})$$

This is called the **maximum a posteriori** (MAP) assignment

- Since Z and $p(\mathbf{x})$ are constants with respect to the choice of \mathbf{y} , can equivalently solve (taking the log of $p(\mathbf{y}, \mathbf{x})$):

$$\operatorname{argmax}_{\mathbf{y}} \sum_{c \in C} \theta_c(\mathbf{x}_c),$$

where $\theta_c(\mathbf{x}_c) = \log \phi_c(\mathbf{x}_c)$

Advanced Prediction Models

Deep Learning, Graphical Models and Reinforcement
Learning

Beyond Prediction

- Recall from the introductory class
 - We discussed complex prediction problems and addressed them using
 - Deep learning architectures
 - Graphical models
 - We also discussed complex decisions, especially in the presence of feedback
- A way to make data-driven decisions: we will look at
 - Online machine learning (this lecture)
 - Reinforcement learning (next)
 - Deep reinforcement learning (next to next)

Examples of Complex Decisions

- ▶ Inventory Management
 - ▶ Observations: current inventory levels
 - ▶ Actions: number of units of each item to purchase
 - ▶ Rewards: profit
- ▶ Resource allocation: who to provide customer service to first
- ▶ Routing problems: in management of shipping fleet, which trucks / truckers to assign to which cargo

Reinforcement Learning: The Next Frontier in Data Science

<https://www.technologyreview.com/s/603501/10-breakthrough-technologies-2017-reinforcement-learning/>

**MIT
Technology
Review**

Past Lists+

Topics+

Top Stories

10 Breakthrough Technologies

The List x

Years +

Reversing Paralysis

Self-Driving Trucks

Paying with Your Face

Practical Quantum Computers

The 360-Degree Selfie

Hot Solar Cells

Gene Therapy 2.0

The Cell Atlas

Botnets of Things

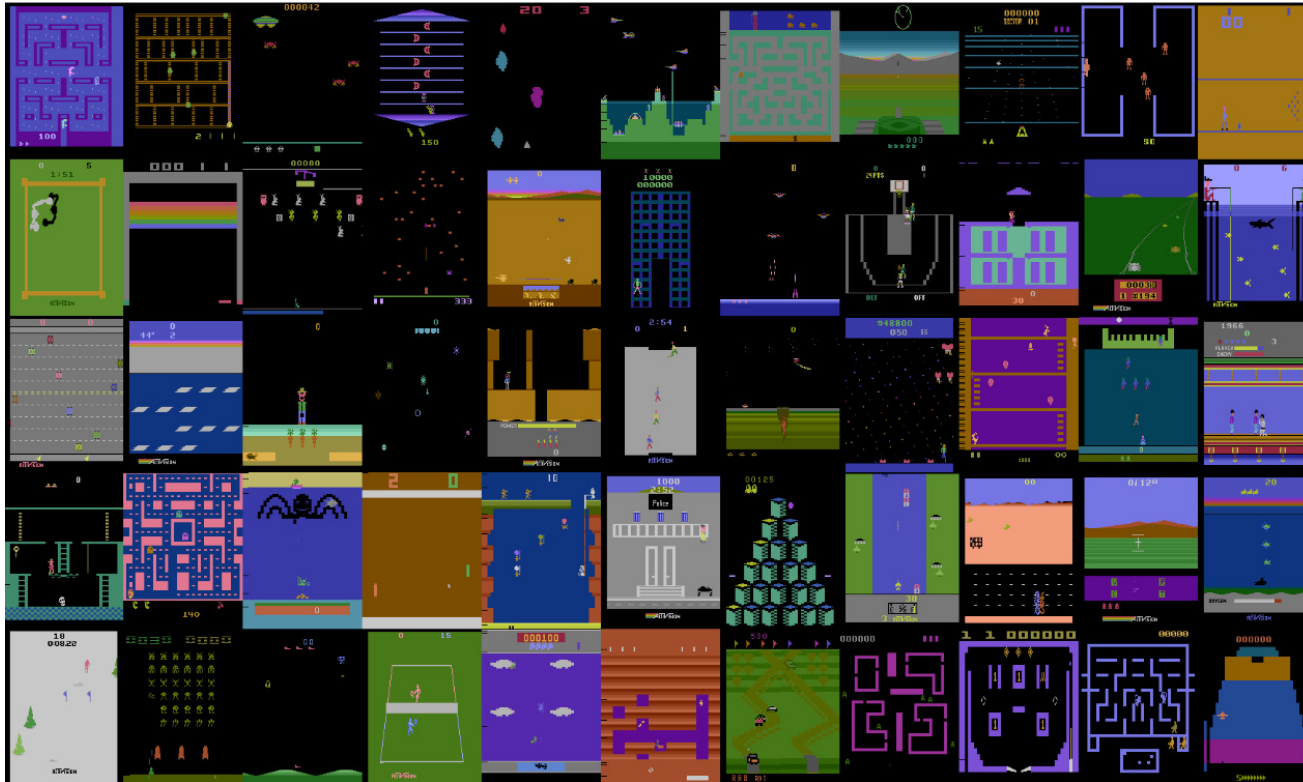
Reinforcement Learning

Reinforcement

By experimenting
figuring out how
no programmer could teach them.

March/April 2017 Issue

Reinforcement Learning: The Next Frontier in Data Science



¹Figure: Defazio Graepel, Atari Learning Environment

Reinforcement Learning: The Next Frontier in Data Science



¹Reference: DeepMind, March 2016

Today's Outline

- Online Machine Learning
- A/B Testing
- Multi-armed bandits
- Contextual bandits

Online Machine Learning

The Gist of Online (Machine) Learning

1. (Optionally) observe the state of the world (aka **context**)
2. Choose an action
3. Obtain feedback on the chosen action

Repeat

The Gist of Online (Machine) Learning

1. (Optionally) observe the state of the world (aka **context**)
2. Choose an action
3. Obtain feedback on the chosen action

Repeat

Goal: Optimize feedback (e.g. maximize reward) for chosen actions

Assumption: Agent's actions do not influence future contexts

MSN Deployment for Personalized News

The screenshot displays the MSN homepage with a personalized news layout. At the top, there's a navigation bar with the MSN logo, a search bar, and links to various services like Outlook.com, Store, Skype, Rewards, Office, OneNote, OneDrive, Maps, and Facebook. Below this, a horizontal menu lists categories: Make MSN my homepage, DATING, NEWS, WEATHER, ENTERTAINMENT, SPORTS, MONEY, LIFESTYLE, HEALTH & FITNESS, FOOD & DRINK, TRAVEL, AUTOS, and VIDEO.

The main content area features several personalized news tiles:

- Best of Late Night Videos:** A carousel of video thumbnails with titles like "Models devour Buffalo wings", "Sanders talks Trump, Clinton", and "Stewart returns to 'The Daily Show'".
- Marjorie Lord, 'Danny Thomas Show' star, dies:** A news tile featuring a photo of Marjorie Lord.
- 7 year-end retirement mistakes you may want to avoid:** A news tile from U.S. News & World Report.
- Clinton vows to defeat Islamic State if elected:** A news tile from Associated Press.
- Wife's role in California attack raises fear of jihad brides:** A news tile from Associated Press.
- 15 ways to drink coffee that will change your mornings forever:** A lifestyle tile from Gourmandize.
- Daily Deal:** A promotional tile for an Asus TP550LA laptop.
- Weather:** A section for Montreal, Canada, showing forecasts for Saturday, Sunday, and Monday.
- Advertisements:** A large Toyota advertisement for the "TOYOTATHON IS ON!" event, featuring a Camry and a woman in a red dress.

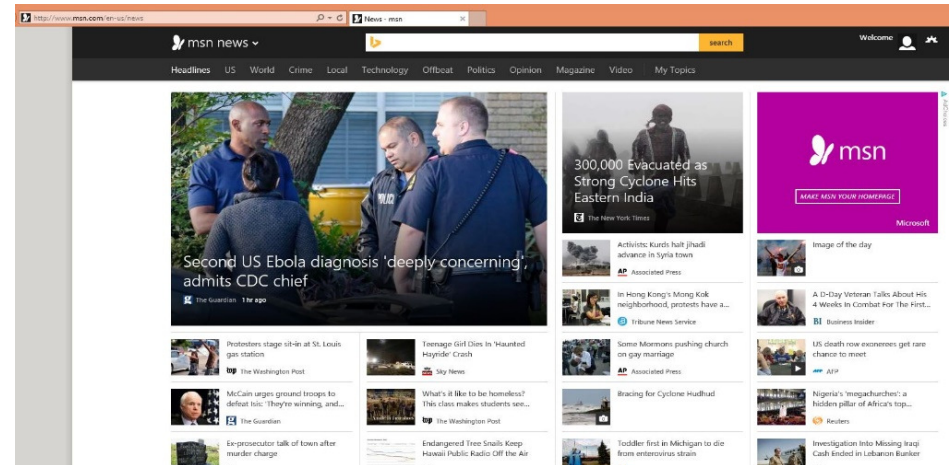
At the bottom, there are four sections:

- EDITORS' PICKS:** Articles like "How police duty belt went from Officer Friendly to Mad Max in 30 years" and "Bruce Springsteen Fans Upset About 'River Tour' Ticket Prices, Resale Scams".
- BEST OF WEEK'S VIDEO:** Videos like "Reporter covering storm blows Internet away" and "Epic fails: How not to fit a rear wiper blade on your car".
- CAREERS:** Articles like "The 50 best places to work in 2016, according to employees" and "15 blue-collar jobs for adrenaline junkies".
- WEEKEND READS:** Articles like "The haunting link between two mass shootings" and "Newborns die after being sent home with drug-dependent mothers".

MSN Deployment for Personalized News

Loop:

1. User **arrives** at MSN with browsing history, user account, previous visits,...
2. Microsoft **chooses** news stories, ...
3. User **responds** to content (clicks, navigation, etc)

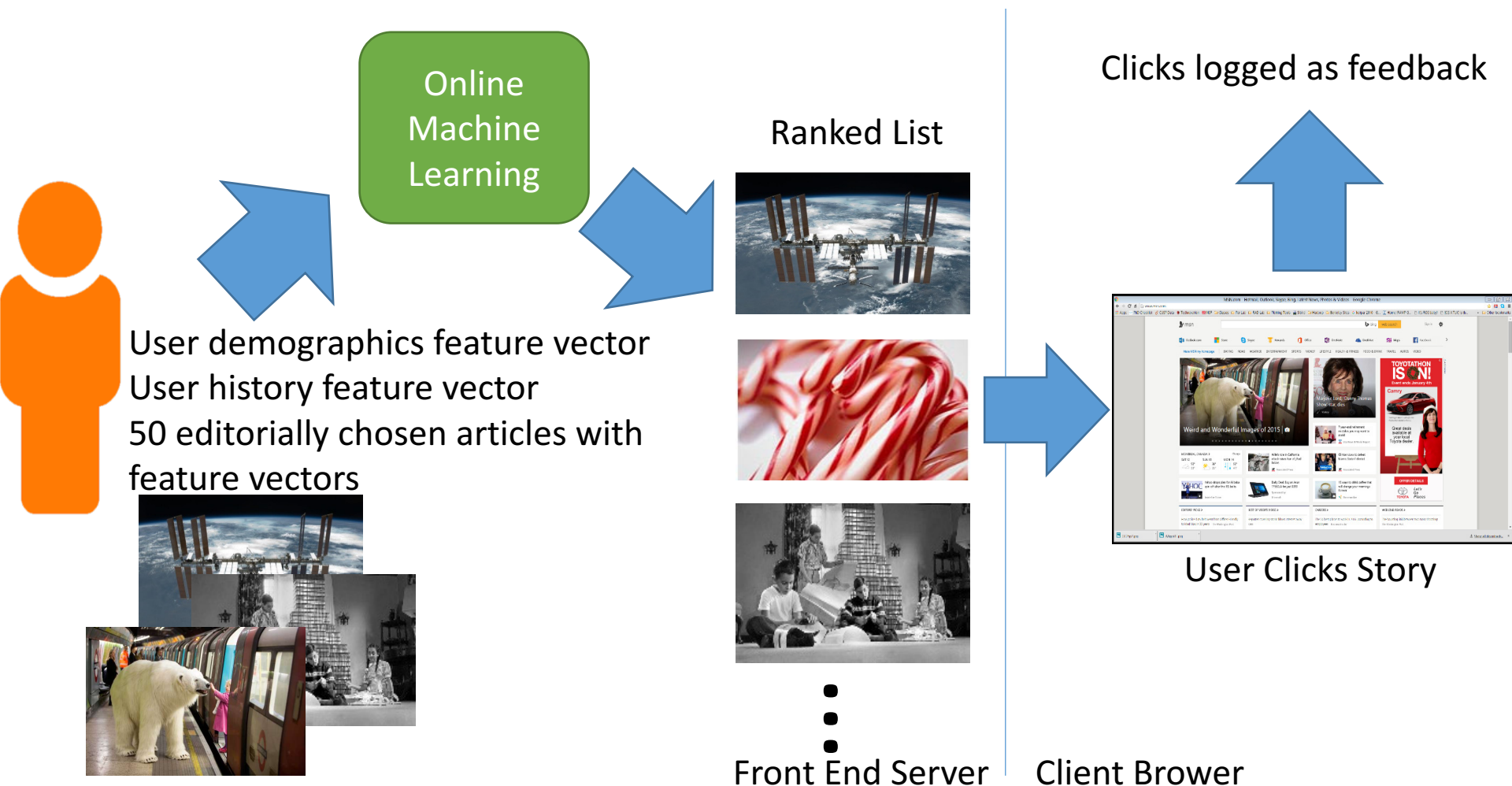


Goal: Choose content to yield desired user behavior

Assumption: Recommendations to one user do not affect other users

¹Reference: Alekh Agarwal et al., <http://arxiv.org/abs/1606.03966>

MSN Deployment for Personalized News



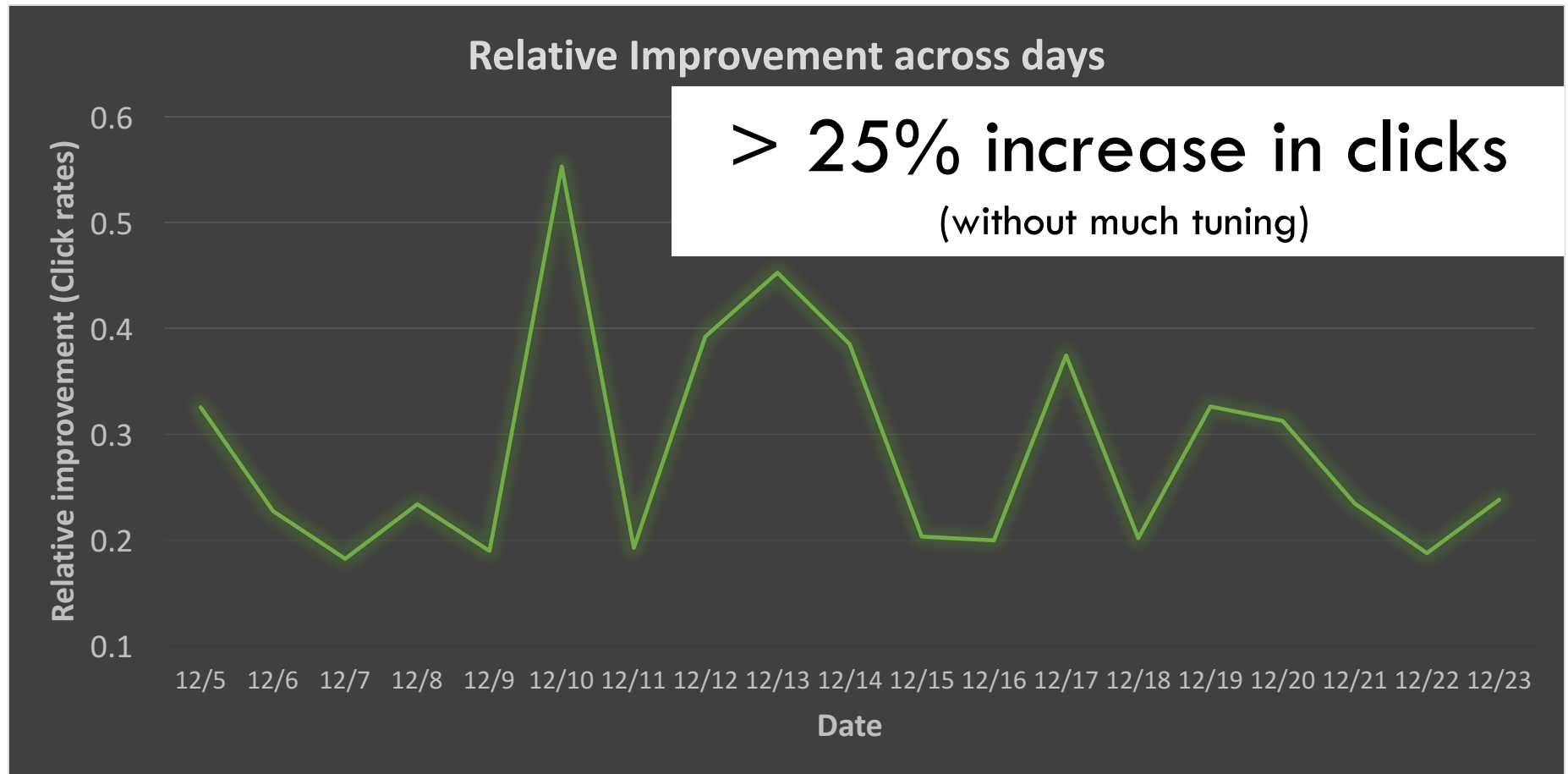
MSN Deployment for Personalized News

- 10 million+ users
- 1000s of requests per second
- 5% overhead on front end machines
- 10s of servers for training
- 5 minute model update frequency



MSN Deployment for Personalized News

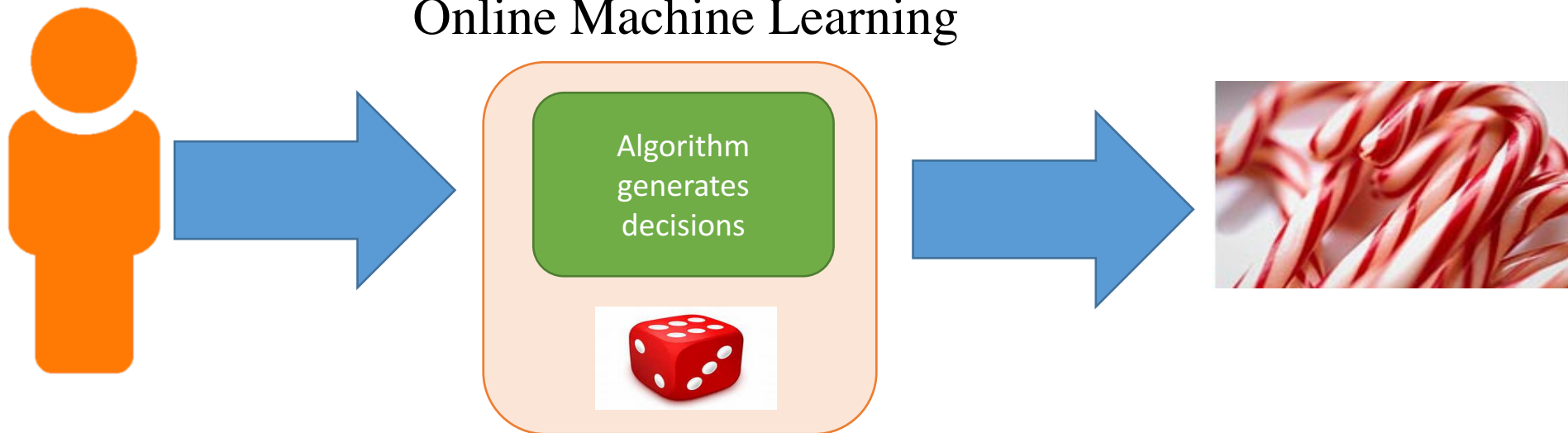
- Relative gains observed



Multitude of Applications

- Content Recommendation: Apps, Movies, Books, ...
- Personalization of search results
- Customer churn prevention
- Adaptive UI personalization
- ...

Online Machine Learning



Questions?

Today's Outline

- Online Machine Learning
- A/B Testing
- Multi-armed bandits
- Contextual bandits

A/B Testing

Motivation for A/B Tests

- Typical business scenario
 - Say there is a meeting to decide on how to improve a product or service
 - Multiple competing ideas emerge
 - Want to make this decision after making some field observations.
 - How to pick one?
- Use A/B testing (this is related to two-sample hypothesis testing)

Motivation for A/B Tests

- Full time companies such as Optimizely, Apptimize, APT, Monetate, etc. provide A/B testing services
- Extensively used at
 - Microsoft for Bing.com (see <http://exp-platform.com>)
 - Google, Facebook, Amazon, Airbnb, LinkedIn ...
- Marketing tools
- Clinical trials (\$11b+ market)

Example with Two Solutions

- Which page has a higher conversion rate?

Doctor FootCare™ Shopping Cart

Home | Products | Learn More | Tips | Testimonials | FAQ | About Us | Contact Us | 1-866-211-9733

Shop With Confidence

- ✓ Satisfaction Guaranteed
- ✓ 30-day, hassle-free Returns
- ✓ 100% Safe, Secured shopping
- ✓ We assure your Privacy

100% Secured Checkout

Continue Shopping > Proceed To Checkout

Item Name	Item Number	Quantity	Remove	Unit Price	Subtotal
Trial Kit	FFCS	1		\$0.00	\$0.00

Update

Total: \$0.00

Select Shipping Method: Standard (\$5.95)

100% Secured Checkout

Continue Shopping > Proceed To Checkout

Home | Products | Learn More | Tips | Testimonials | FAQ | About Us | Contact Us | Shopping Cart

A

Doctor FootCare™ Shopping Cart

Home | Products | Learn More | Tips | Testimonials | FAQ | About Us | Contact Us | 1-866-211-9733

Shop With Confidence

- ✓ Satisfaction Guaranteed
- ✓ 30-day, hassle-free Returns
- ✓ 100% Safe, Secured shopping
- ✓ We assure your Privacy

100% Secured Checkout

Continue Shopping > Proceed To Checkout

Item Name	Item Number	Quantity	Remove	Unit Price	Subtotal
Trial Kit	FFCS	1		\$0.00	\$0.00

Discount: \$0.00

Total: \$0.00

Enter Coupon Code

Select Shipping Method: Standard (\$5.95)

100% Secured Checkout

Recalculate Continue Shopping > Proceed To Checkout

Home | Products | Learn More | Tips | Testimonials | FAQ | About Us | Contact Us | Shopping Cart

B

Kumar et al. 2009

Example with Two Solutions

- Which page has a higher conversion rate?

Doctor FootCare™ Shopping Cart

Home | Products | Learn More | Tips | Testimonials | FAQ | About Us | Contact Us | 1-866-211-9733

Shop With Confidence

- ✓ Satisfaction Guaranteed
- ✓ 30-day, hassle-free Returns
- ✓ 100% Safe, Secured shopping
- ✓ We assure your Privacy

100% Secured Checkout

Continue Shopping > Proceed To Checkout

Item Name	Item Number	Quantity	Remove	Unit Price	Subtotal
Trial Kit	FFCS	1		\$0.00	\$0.00

Update

Total: \$0.00

Select Shipping Method: Standard (\$5.95)

100% Secured Checkout

Continue Shopping > Proceed To Checkout

Home | Products | Learn More | Tips | Testimonials | FAQ | About Us | Contact Us | Shopping Cart

A

Doctor FootCare™ Shopping Cart

Home | Products | Learn More | Tips | Testimonials | FAQ | About Us | Contact Us | 1-866-211-9733

Shop With Confidence

- ✓ Satisfaction Guaranteed
- ✓ 30-day, hassle-free Returns
- ✓ 100% Safe, Secured shopping
- ✓ We assure your Privacy

100% Secured Checkout

Continue Shopping > Proceed To Checkout

Item Name	Item Number	Quantity	Remove	Unit Price	Subtotal
Trial Kit	FFCS	1		\$0.00	\$0.00

Discount: \$0.00

Total: \$0.00

Enter Coupon Code

Select Shipping Method: Standard (\$5.95)

100% Secured Checkout

Recalculate Continue Shopping > Proceed To Checkout

Home | Products | Learn More | Tips | Testimonials | FAQ | About Us | Contact Us | Shopping Cart

B

Kumar et al. 2009

- With B, site lost 90% of revenue: users want to find coupons to reduce price

A/B Testing Setup

- First we will ignore the online aspect of the problem
- That is, we will ignore instantaneous feedback
- We will only use these feedbacks at the end of a period
- In particular,
 - They will be used to decide on good recommendation policies

A/B Testing Setup

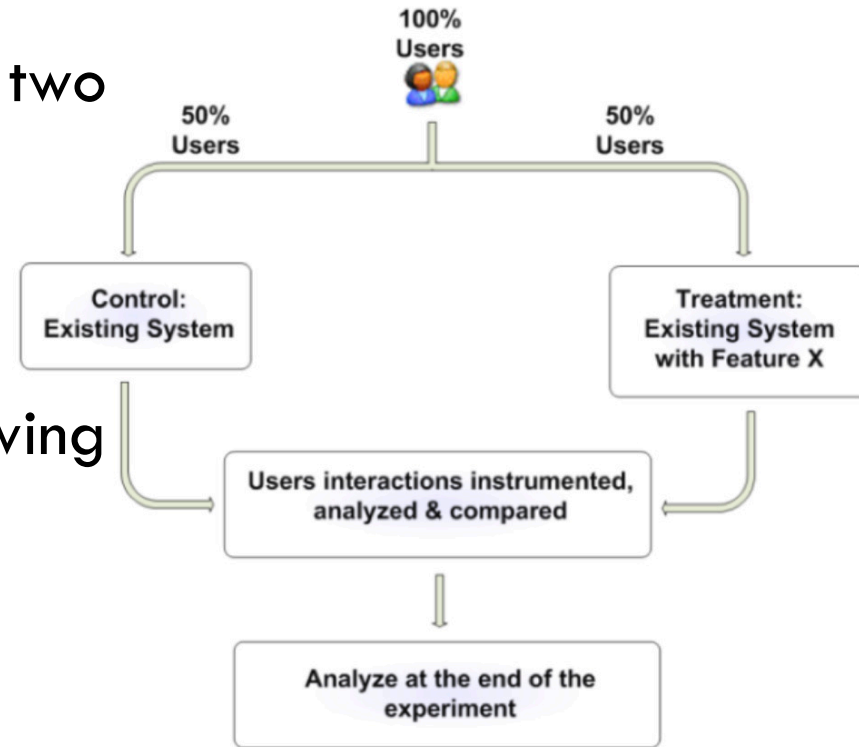
- A/B testing is about showing users two solutions



- And figuring out if solution A is different than solution B

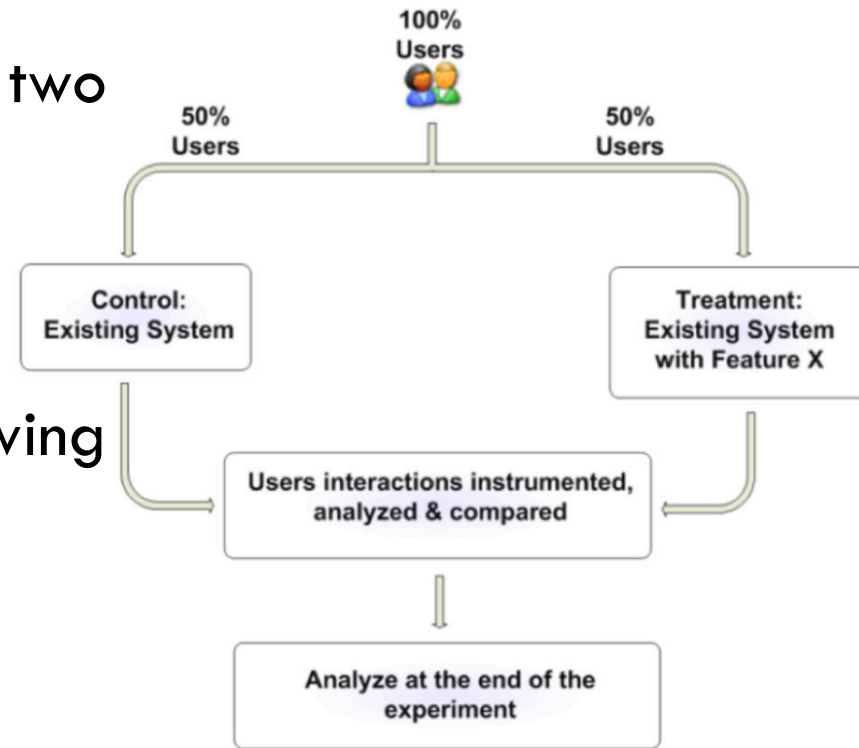
A/B Testing Setup

- A/B testing is about showing users two solutions
 - A (control)
 - B (treatment)
- Randomly split the users while showing



A/B Testing Setup

- A/B testing is about showing users two solutions
 - A (control)
 - B (treatment)
- Randomly split the users while showing
- Collect the outcomes and decide which option was better
 - Best scientific way to establish cause-effect relationship
 - Compared to offline data analysis (error prone)



A/B Testing is Two Sample Testing

- A/B testing is about collecting statistics across two groups
- Randomized assignment of the two solutions to each user is a key requirement
 - Eliminates biases and confounding
- Say each group of users has true mean effect μ_1 and μ_2
- From data, we want to infer whether
 - These are different (statistical significance)?
 - Same?
 - Which is larger?

Types of Hypothesis Tests

- Fisher
 - Reject H_0 (no acceptance as such)
 - More data typically leads to rejection
- Neyman-Pearson
 - Compare H_0 to H_1
 - Find likelihood ratio $P(Data|H_0)/P(Data|H_1)$
- Bayesian
 - Compute $P(H_0|Data)/P(H_1|Data)$
 - Similar to Neyman-Pearson when $P(H_0) = P(H_1)$

A/B Testing Pros

- Very intuitive setup and conclusions
- Field experiment decides the worth of a feature/offering, not gut instinct
- Most used in industry! (compared to bandit techniques)
 - Also called split or bucket testing
- Need not be a one time process
 - Can repeat if you think users have changed in terms of their preferences

A/B Testing Cons

- Has many bells and whistles to make it work
 - Especially because most treatment effects show small incremental improvement
 - See <http://exp-platform.com> for an extensive list of issues that affect A/B testing
- What if we can change who sees what treatment (action) dynamically?
 - Leads to Multi-Armed Bandit problems.
- What if we want to **optimize** over several options dynamically depending on context?
 - Leads to Contextual Bandit problems.

Questions?

Today's Outline

- Online Machine Learning
- A/B Testing
- Multi-armed bandits
- Contextual bandits

Bandit Problems

The Multi-armed Bandit Problem

- Multi-armed bandit (MAB) problem involves the following in each interaction



- pulling an arm = making a choice (which ad/color to display)
- reward/regret = measure of success (user-click, item-buy)

The Formal Setting

Problem Formulation

Consider K arms (actions) each correspond to an unknown distribution $\{\nu_k\}_{k=1}^K$ with values bounded in $[0, 1]$.

- At each time t , the agent pulls an arm $I_t \in \{1, \dots, K\}$ and observes a reward $x_t \sim \nu_{I_t}$ (i.i.d. sample from ν_{I_t}).
- The objective is to maximize the expected sum of rewards.

Notations

- mean of each arm: $\mu_k = \mathbb{E}_{X \sim \nu_k}[X]$
- mean of the best arm: $\mu^* = \max_k \mu_k$

MAB Performance

- It is an online problem.
- We need to come up with algorithms/strategies.
 - Example:
 - a round-robin strategy
 - A constant strategy (bad idea!)

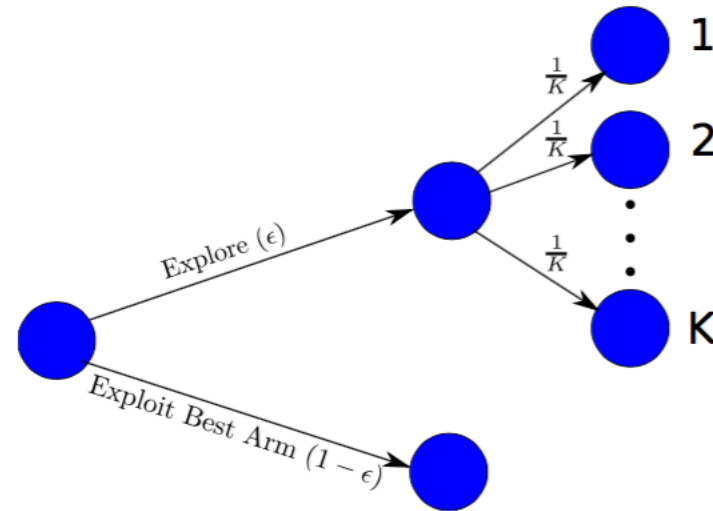
To evaluate the performance of a strategy

Cumulative Regret

$$R_n = n\mu^* - \sum_{t=1}^n x_t$$

Objective: find a strategy with small *expected cumulative regret* $\mathbb{E}[R_n]$

The Epsilon-Greedy Algorithm



Strategy = $\epsilon \cdot \text{Scientist} + (1 - \epsilon) \cdot \text{Businessman}$

At each time t

- With probability $1 - \epsilon$, pick the subjectively best arm
- With probability $\frac{\epsilon}{K}$, pick a random arm

The Epsilon-Greedy Algorithm Intuition

- How can we do well? We need to explore the arms. We also need to exploit what we have learned so far.

Scientist View

- Explore new ideas



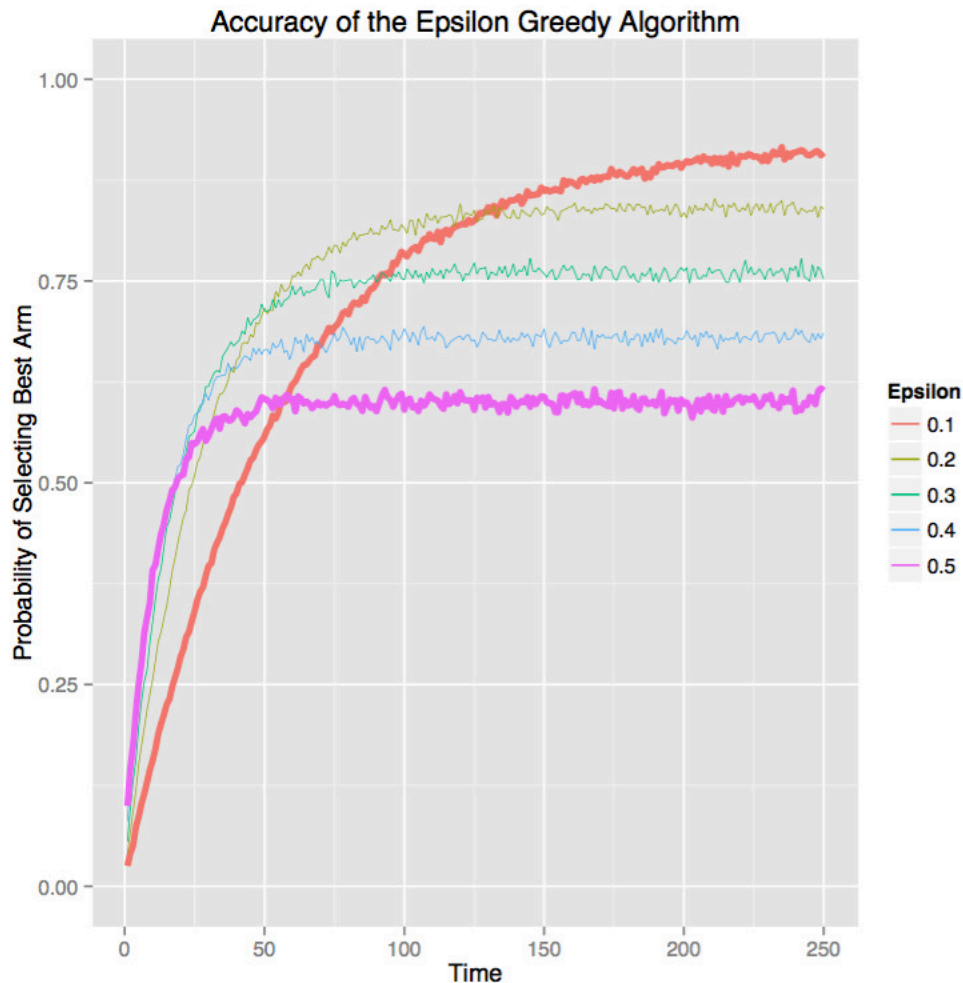
Businessman View

- Exploit best idea found so far



Epsilon-Greedy Synthetic Experiment

5 Bernoulli arms with reward probabilities **0.1, 0.1, 0.1, 0.1, 0.9**



$\epsilon = 0.1$ (Businessman)

- Learns slowly
- Does well at the end

$\epsilon = 0.5$ (Scientist)

- Learns quickly
- Doesn't exploit at the end

The Upper Confidence Bound (UCB) Algorithm

- Lets look at a slightly more involved algorithm: UCB

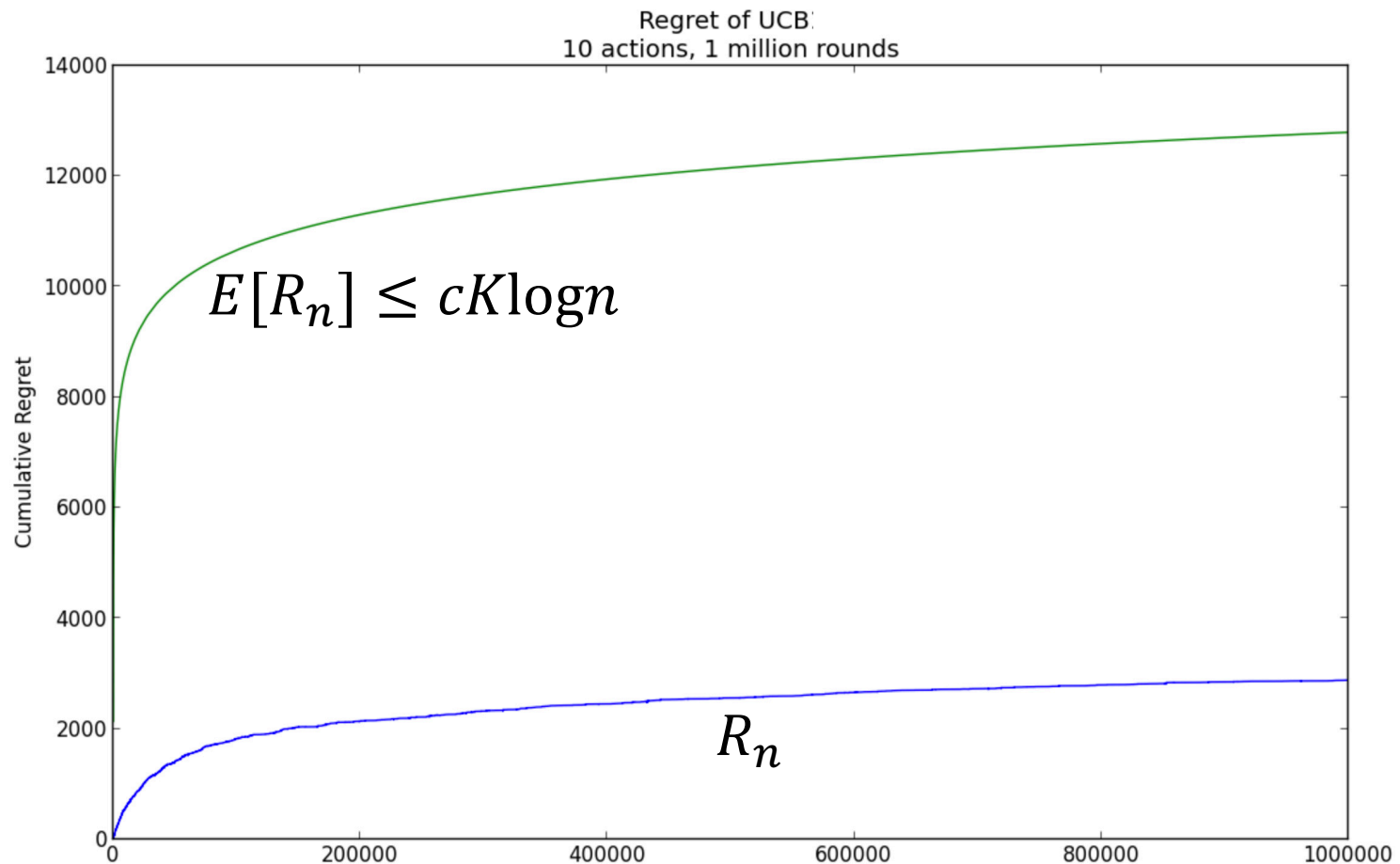
Upper confidence bound (UCB) strategy selects an arm at time t that

$$I_t = \arg \max_k B_{t, T_k(t-1)}(k) \quad , \quad B_{t,s}(k) = \hat{\mu}_{k,s} + \sqrt{\frac{2 \log t}{s}}$$

$\hat{\mu}_{k,s} = \frac{1}{s} \sum_{i=1}^s x_{k,i}$ is the **empirical mean** of arm k at time s

UCB Synthetic Experiment

- 10 actions, 10^6 interactions (is this realistic?)
- Reward for each action has mean $0.5/k$ ($5 \leq k \leq 15$)



The Thompson Sampling Algorithm

- A Bayesian algorithm for MAB problems is as follows

In Thompson [1933] the following strategy was proposed for the case of Bernoulli distributions:

- Assume a **uniform prior** on the parameters $\mu_i \in [0, 1]$.
- Let $\pi_{i,t}$ be the **posterior distribution** for μ_i at the t^{th} round.
- Let $\theta_{i,t} \sim \pi_{i,t}$ (independently from the past given $\pi_{i,t}$).
- $I_t \in \operatorname{argmax}_{i=1,\dots,K} \theta_{i,t}$.

Thompson Sampling: Conjugate Priors

A family of prior distribution

$$\mathcal{P}_A = \{p_\alpha(\theta) \mid \alpha \in A\}$$

is said to be **conjugate** to a model \mathcal{P}_Θ , if, for a sample

$$X^{(1)}, \dots, X^{(n)} \stackrel{\text{i.i.d.}}{\sim} p_\theta \quad \text{with} \quad p_\theta \in \mathcal{P}_\Theta,$$

the distribution q defined by

$$q(\theta) = p(\theta | x^{(1)}, \dots, x^{(n)}) = \frac{p_\alpha(\theta) \prod_i p_\theta(x^{(i)})}{\int p_\alpha(\theta) \prod_i p_\theta(x^{(i)}) d\theta}$$

is such that

$$q \in \mathcal{P}_A.$$

Thompson Sampling: Conjugate Priors

We say that $\boldsymbol{\theta} = (\theta_1, \dots, \theta_K)$ follows the Dirichlet distribution and note

$$\boldsymbol{\theta} \sim \text{Dir}(\boldsymbol{\alpha})$$

for $\boldsymbol{\theta}$ in the simplex $\triangle_K = \{\mathbf{u} \in \mathbb{R}_+^K \mid \sum_{k=1}^K u_k = 1\}$ and

Thompson Sampling: Conjugate Priors

We say that $\boldsymbol{\theta} = (\theta_1, \dots, \theta_K)$ follows the Dirichlet distribution and note

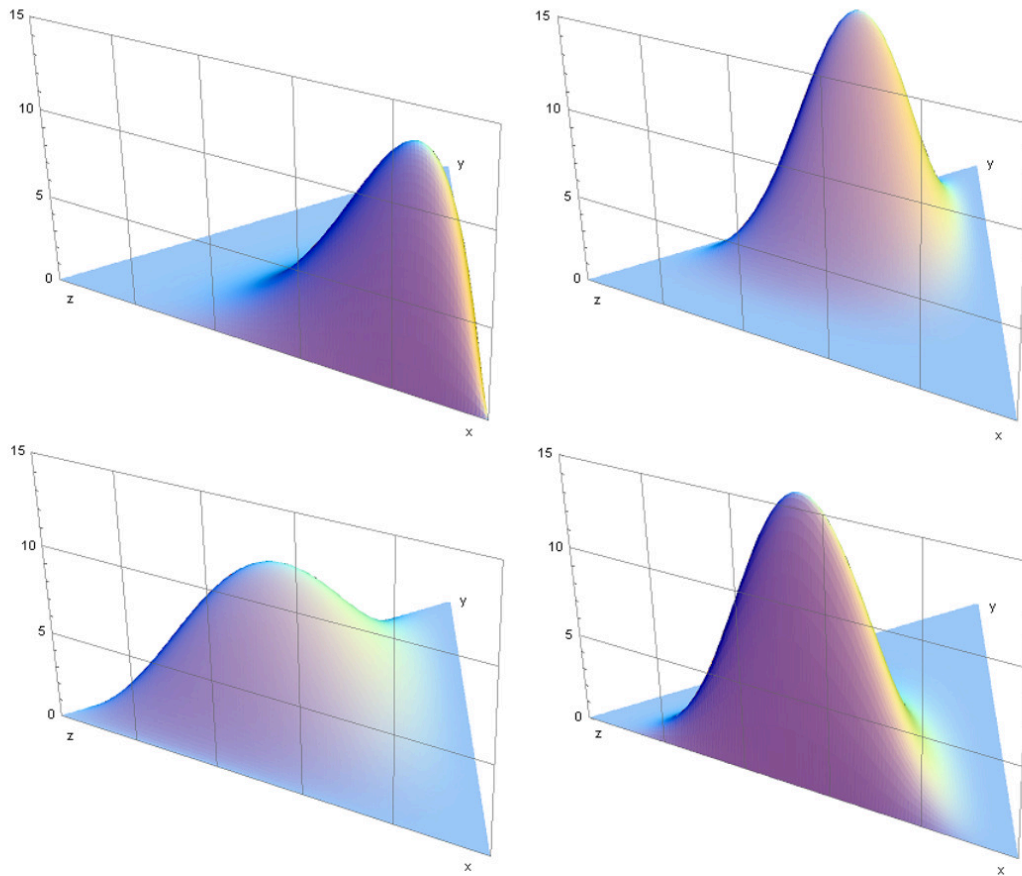
$$\boldsymbol{\theta} \sim \text{Dir}(\boldsymbol{\alpha})$$

for $\boldsymbol{\theta}$ in the simplex $\Delta_K = \{\mathbf{u} \in \mathbb{R}_+^K \mid \sum_{k=1}^K u_k = 1\}$ and admitting the density

$$p(\boldsymbol{\theta}; \boldsymbol{\alpha}) = \frac{\Gamma(\alpha_0)}{\prod_k \Gamma(\alpha_k)} \theta_1^{\alpha_1-1} \dots \theta_K^{\alpha_K-1}$$

$$\alpha_0 = \sum_k \alpha_k \quad \text{and} \quad \Gamma(x) := \int_0^\infty t^{x-1} e^{-t} dt$$

Thompson Sampling: Conjugate Priors



¹Reference: http://imagine.enpc.fr/%7Eeobozinsg/stats_review.html

Thompson Sampling:

Categorical-Dirichlet Conjugacy

Consider the simple Bayesian Dirichlet-Multinomial model with

- A Dirichlet prior on the parameter of the multinomial: $\boldsymbol{\theta} \sim \text{Dir}(\boldsymbol{\alpha})$
- A multinomial random variable $\mathbf{z} \sim \mathcal{M}(1, \boldsymbol{\theta})$

$$p(\boldsymbol{\theta}) \propto \prod_{k=1}^K \theta_k^{\alpha_k - 1} \quad \text{and} \quad p(\mathbf{z}|\boldsymbol{\theta}) = \prod_{k=1}^K \theta_k^{z_k}$$

Thompson Sampling:

Categorical-Dirichlet Conjugacy

Consider the simple Bayesian Dirichlet-Multinomial model with

- A Dirichlet prior on the parameter of the multinomial: $\boldsymbol{\theta} \sim \text{Dir}(\boldsymbol{\alpha})$
- A multinomial random variable $\mathbf{z} \sim \mathcal{M}(1, \boldsymbol{\theta})$

$$p(\boldsymbol{\theta}) \propto \prod_{k=1}^K \theta_k^{\alpha_k - 1} \quad \text{and} \quad p(\mathbf{z}|\boldsymbol{\theta}) = \prod_{k=1}^K \theta_k^{z_k}$$

Let $\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(N)}$ be an i.i.d. sample distributed like \mathbf{z} .

We have

$$p(\boldsymbol{\theta}|\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(N)}) = \frac{p(\boldsymbol{\theta}) \prod_n p(\mathbf{z}^{(n)}|\boldsymbol{\theta})}{p(\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(N)})}$$

Thompson Sampling:

Categorical-Dirichlet Conjugacy

Consider the simple Bayesian Dirichlet-Multinomial model with

- A Dirichlet prior on the parameter of the multinomial: $\boldsymbol{\theta} \sim \text{Dir}(\boldsymbol{\alpha})$
- A multinomial random variable $\mathbf{z} \sim \mathcal{M}(1, \boldsymbol{\theta})$

$$p(\boldsymbol{\theta}) \propto \prod_{k=1}^K \theta_k^{\alpha_k - 1} \quad \text{and} \quad p(\mathbf{z}|\boldsymbol{\theta}) = \prod_{k=1}^K \theta_k^{z_k}$$

Let $\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(N)}$ be an i.i.d. sample distributed like \mathbf{z} .

We have

$$p(\boldsymbol{\theta}|\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(N)}) = \frac{p(\boldsymbol{\theta}) \prod_n p(\mathbf{z}^{(n)}|\boldsymbol{\theta})}{p(\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(N)})} \propto \prod_k \theta_k^{\alpha_k + \sum_n z_{nk} - 1}$$

So that $(\boldsymbol{\theta}|\mathbf{Z}) \sim \text{Dir}((\alpha_1 + N_1, \dots, \alpha_K + N_K))$ with $N_k = \sum_n z_{nk}$

Non-Probabilistic Setting

- Why do we need to assume that the rewards are i.i.d.?
- Can we drop the stochastic assumptions on the rewards?
- Reason #1: These rewards may be the output of a complex process
- Reason #2: These rewards may be generated by an ‘adversary’ (someone who is not random)

Non-Probabilistic Setting

- We can in fact drop the probabilistic reward assumption!
- Template
 - Adversary selects rewards $x_t(1), \dots, x_t(K)$, which are not known to the player (us)
 - Player selects arm I_t
 - In full information, player sees $x_t(1), \dots, x_t(K)$
 - In bandit information setup, player only sees $x_t(I_t)$

Exp3 Algorithm

Initialization: $w_1(k) = 1$ for all $k = 1, \dots, K$

Exp3 Algorithm

Initialization: $w_1(k) = 1$ for all $k = 1, \dots, K$

At each time $t = 1, \dots, n$: the player selects an arm $I_t \sim p_t$, where

$$p_t(k) = (1 - \gamma) \frac{w_t(k)}{\sum_{i=1}^K w_t(i)} + \frac{\gamma}{K}$$

Exp3 Algorithm

Initialization: $w_1(k) = 1$ for all $k = 1, \dots, K$

At each time $t = 1, \dots, n$: the player selects an arm $I_t \sim p_t$, where

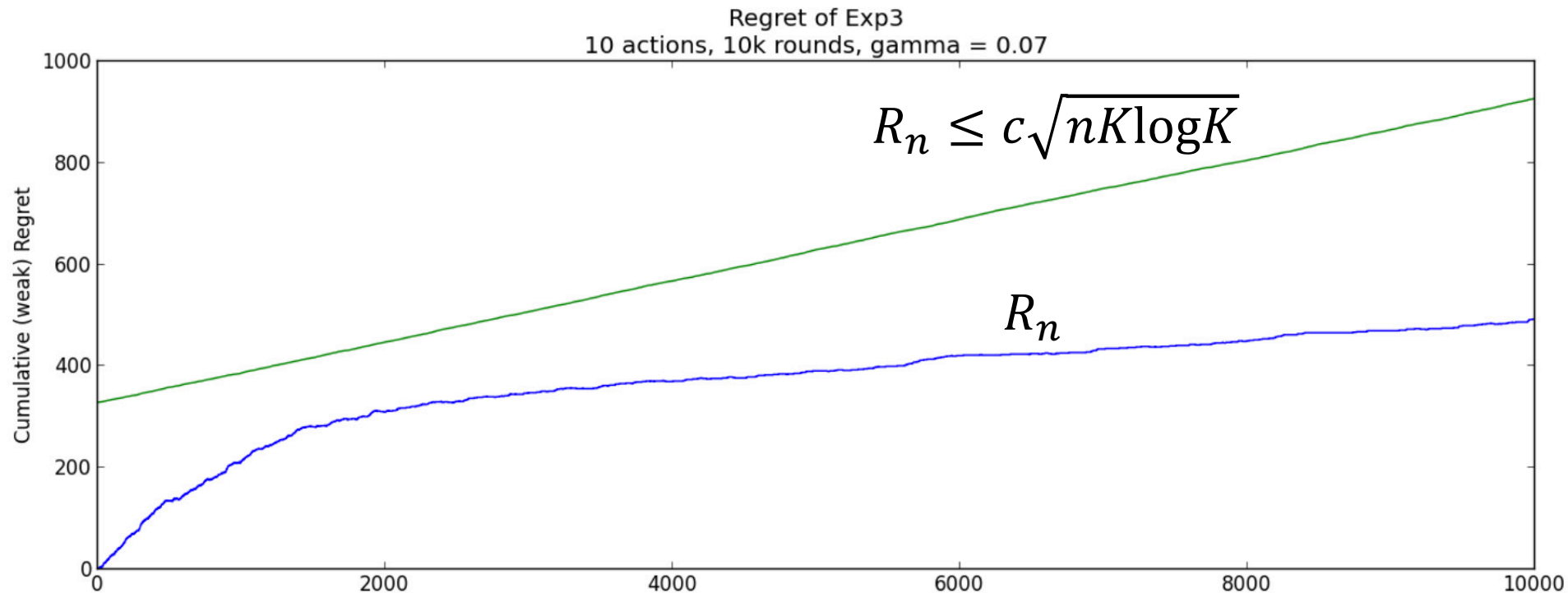
$$p_t(k) = (1 - \gamma) \frac{w_t(k)}{\sum_{i=1}^K w_t(i)} + \frac{\gamma}{K}$$

with $w_t(k) = e^{\eta \sum_{s=1}^{t-1} \tilde{x}_s(k)}$, where $\tilde{x}_s(k) = \frac{x_s(k)}{p_s(k)} \mathbf{1}\{I_s = k\}$.

$\eta > 0$ and $\gamma > 0$ are the parameters of the algorithm.

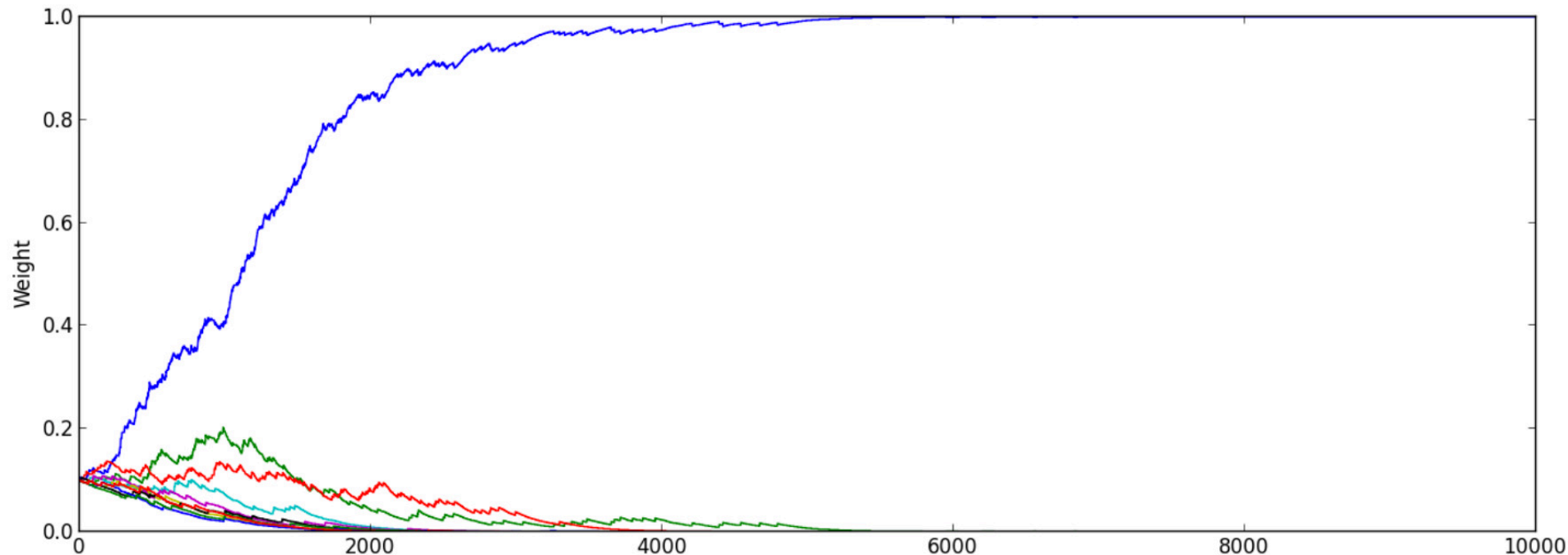
Exp3 Synthetic Experiment

- 10 actions, 10^3 interactions
- Reward for each action is Bernoulli with means $1/k$ ($2 \leq k < 12$)



Exp3 Synthetic Experiment

- 10 actions, 10^3 interactions
- Reward for each action is Bernoulli with means $1/k$ ($2 \leq k < 12$)



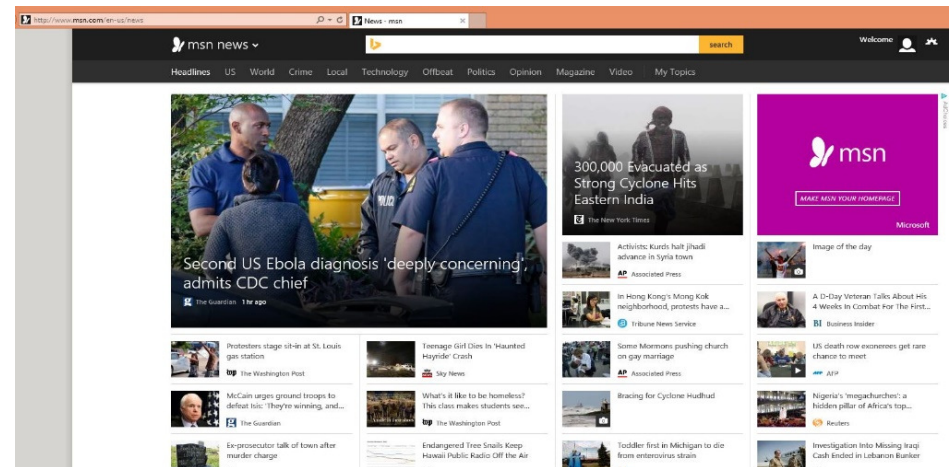
Questions?

Bandits with Contexts

Recall: MSN Deployment for Personalized News

Loop:

1. User **arrives** at MSN with browsing history, user account, previous visits,...
2. Microsoft **chooses** news stories, ...
3. User **responds** to content (clicks, navigation, etc)



Goal: Choose content to yield desired user behavior

Assumption: Recommendations to one user do not affect other users

¹Reference: Alekh Agarwal et al., <http://arxiv.org/abs/1606.03966>

Previous Bandit Models are not Enough

- No context!
- No-carry over effect from one interaction to the next
 - Say users can change behavior by seeing recommendations
 - Can be captured by Reinforcement Learning

The Contextual Bandit Problem

- In the Contextual Bandit problem,
 - Every round, we get context
 - We want to find the best policy (what to do in each context)
 - May not see the same context twice!
- Different from MAB setting because in MAB problems
 - No context
 - We were finding a single best action

Benefit of Context

- Say we have 5 ads

$a_1 =$ “buy pet lizards”

$a_2 =$ “1-800-petunias”

$a_3 =$ “cheap mp3 players”

$a_4 =$ “find local florists”

$a_5 =$ “affordable dragon souls”

- Say we have 4 policies
 - These map context to ads

- Now, lets look at one round of Exp3
 - For Exp3, it is as if it has 4 “arms” (one per policy)

Benefit of Context

- In round t say the policies recommend the following:

e_1 chose a_2

$a_1 =$ “buy pet lizards”

e_2 chose a_2

$a_2 =$ “1-800-petunias”

e_3 chose a_4

$a_3 =$ “cheap mp3 players”

$a_4 =$ “find local florists”

e_4 chose a_4

$a_5 =$ “affordable dragon souls”

- Say Exp3 chose “arm” e_1 by sampling from weights
- And, say e_1 ’s ad choice a_2 was clicked

Benefit of Context

- Exp3 assigns reward $\tilde{x}_s(e_1) = \frac{x_s(e_1)}{p_s(e_1)}$
- Rest of the arms all get reward 0
- Can we do better?
 - Yes! e_2 also was recommending a_2
 - We should better estimate reward of e_2

e_1 chose a_2

e_2 chose a_2

e_3 chose a_4

e_4 chose a_4

Exp4 Algorithm

Initialization: $\forall \pi \in \Pi : w_t(\pi) = 1$

For each $t = 1, 2, \dots$:

1. Observe x_t and let for $a = 1, \dots, K$

$$p_t(a) = \frac{\sum_{\pi} \mathbf{1}[\pi(x_t) = a] w_t(\pi)}{\sum_{\pi} w_t(\pi)}.$$

Exp4 Algorithm

Initialization: $\forall \pi \in \Pi : w_t(\pi) = 1$

For each $t = 1, 2, \dots$:

1. Observe x_t and let for $a = 1, \dots, K$

$$p_t(a) = \frac{\sum_{\pi} \mathbf{1}[\pi(x_t) = a] w_t(\pi)}{\sum_{\pi} w_t(\pi)} + p_{\min},$$

Exp4 Algorithm

Initialization: $\forall \pi \in \Pi : w_t(\pi) = 1$

For each $t = 1, 2, \dots$:

1. Observe x_t and let for $a = 1, \dots, K$

$$p_t(a) = (1 - Kp_{\min}) \frac{\sum_{\pi} \mathbf{1}[\pi(x_t) = a] w_t(\pi)}{\sum_{\pi} w_t(\pi)} + p_{\min},$$

Exp4 Algorithm

Initialization: $\forall \pi \in \Pi : w_t(\pi) = 1$

For each $t = 1, 2, \dots$:

1. Observe x_t and let for $a = 1, \dots, K$

$$p_t(a) = (1 - Kp_{\min}) \frac{\sum_{\pi} \mathbf{1}[\pi(x_t) = a] w_t(\pi)}{\sum_{\pi} w_t(\pi)} + p_{\min},$$

where $p_{\min} = \sqrt{\frac{\ln |\Pi|}{KT}}$.

Exp4 Algorithm

Initialization: $\forall \pi \in \Pi : w_t(\pi) = 1$

For each $t = 1, 2, \dots$:

1. Observe x_t and let for $a = 1, \dots, K$

$$p_t(a) = (1 - Kp_{\min}) \frac{\sum_{\pi} \mathbf{1}[\pi(x_t) = a] w_t(\pi)}{\sum_{\pi} w_t(\pi)} + p_{\min},$$

where $p_{\min} = \sqrt{\frac{\ln |\Pi|}{KT}}$.

2. Draw a_t from p_t , and observe reward $r_t(a_t)$.

Exp4 Algorithm

Initialization: $\forall \pi \in \Pi : w_t(\pi) = 1$

For each $t = 1, 2, \dots$:

1. Observe x_t and let for $a = 1, \dots, K$

$$p_t(a) = (1 - Kp_{\min}) \frac{\sum_{\pi} \mathbf{1}[\pi(x_t) = a] w_t(\pi)}{\sum_{\pi} w_t(\pi)} + p_{\min},$$

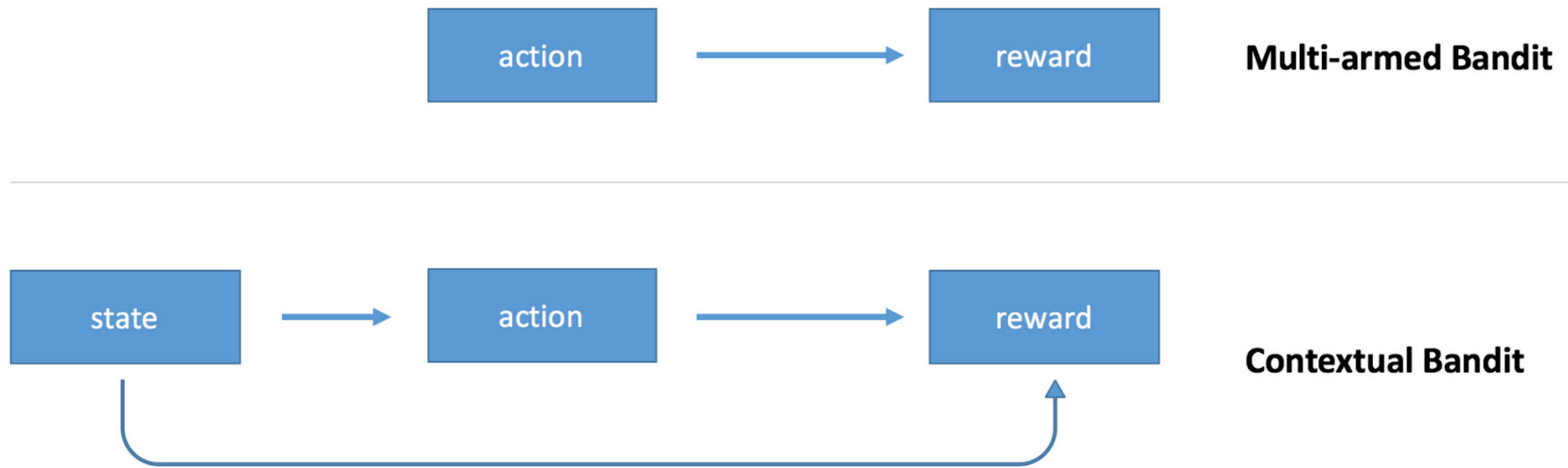
where $p_{\min} = \sqrt{\frac{\ln |\Pi|}{KT}}$.

2. Draw a_t from p_t , and observe reward $r_t(a_t)$.
3. Update for each $\pi \in \Pi$

$$w_{t+1}(\pi) = \begin{cases} w_t(\pi) \exp \left(p_{\min} \frac{r_t(a_t)}{p_t(a_t)} \right) & \text{if } \pi(x_t) = a_t \\ w_t(\pi) & \text{otherwise} \end{cases}$$

Questions?

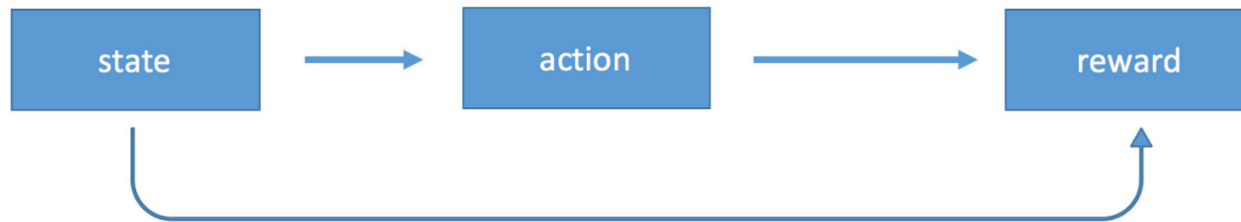
Reinforcement Learning: Because Contextual Bandit Formulation is not Enough



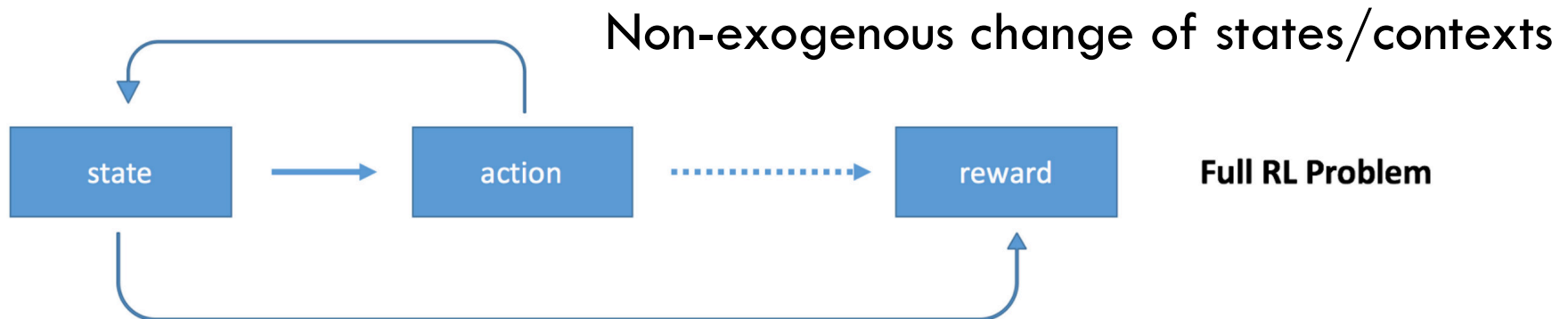
Reinforcement Learning: Because Contextual Bandit Formulation is not Enough



Multi-armed Bandit



Contextual Bandit



Full RL Problem

Summary

- We looked at A/B testing as a way to introduce enhancements in a business product/service
 - May need a lot of examples
 - Is based on the idea of randomized control trials
- We also looked at two new online ML problems
 - Multi-Armed Bandits
 - Contextual Bandits
- Contextual bandits are a special case of reinforcement learning, which we will study next time.

Appendix

Sample Exam Questions

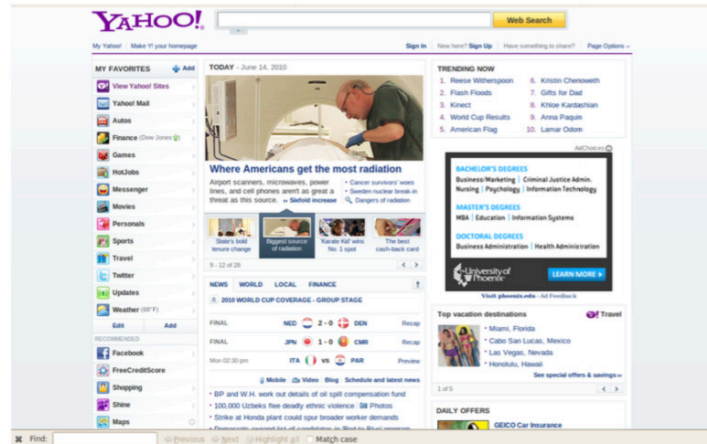
- What is the difference between A/B testing and Multi-armed bandits?
- Can we do A/B testing when we have more than two options?
- What is the role of exploration in the Bandit problems?
- Can Exp3 be used in a stochastic setting?
- How does the contextual problem differ from the non-contextual problem?

Online ML is Difficult to Deploy

- Separate teams for each part of the process
- Faulty logging
 - Logging just choice, not probabilities
 - Features not logged and change in time
- Runtime behavior incompatible with the ML
 - Business logic overriding randomization
 - Using the probability as feature for downstream ML
- Subtle errors that are difficult to find in complex systems!

¹Reference: Alekh Agarwal et al., <http://arxiv.org/abs/1606.03966>

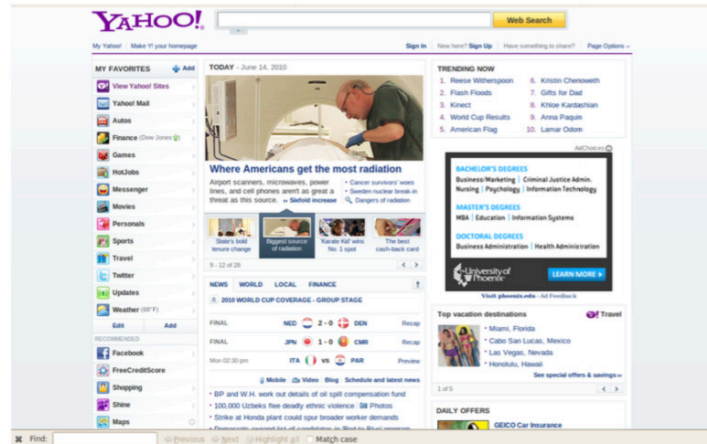
Contextual Bandit: Website Example



Repeatedly:

1. A user comes to Yahoo! (with history of previous visits, IP address, data related to his Yahoo! account)

Contextual Bandit: Website Example



Repeatedly:

1. A user comes to Yahoo! (with history of previous visits, IP address, data related to his Yahoo! account)
2. Yahoo! chooses information to present (from urls, ads, news stories)

Contextual Bandit: Website Example



Repeatedly:

1. A user comes to Yahoo! (with history of previous visits, IP address, data related to his Yahoo! account)
2. Yahoo! chooses information to present (from urls, ads, news stories)
3. The user reacts to the presented information (clicks on something, clicks, comes back and clicks again, et cetera)

Contextual Bandit: Website Example



Repeatedly:

1. A user comes to Yahoo! (with history of previous visits, IP address, data related to his Yahoo! account)
2. Yahoo! chooses information to present (from urls, ads, news stories)
3. The user reacts to the presented information (clicks on something, clicks, comes back and clicks again, et cetera)

Yahoo! wants to interactively choose content and use the observed feedback to improve future content choices.

¹Reference: John Langford (2011)

Contextual Bandit: Clinical Example



Repeatedly:

1. A patient comes to a doctor with symptoms, medical history, test results
2. The doctor chooses a treatment
3. The patient responds to it

The doctor wants a policy for choosing targeted treatments for individual patients.

Additional Resources

- Course at UWash:
 - <http://courses.cs.washington.edu/courses/cse599s/12sp/scribes.html> (lectures 13,14)
- Course at UCSD:
 - <http://cseweb.ucsd.edu/~kamalika/teaching/CSE291W11/> (lecture5)
- Tutorial by Bygelzimer and Langford:
 - http://hunch.net/~exploration_learning/
- Course at UAlberta:
 - <https://sites.ualberta.ca/~szepesva/CMPUT654/>

Note: These are optional. May be slightly theoretical in nature.

Advanced Prediction Models

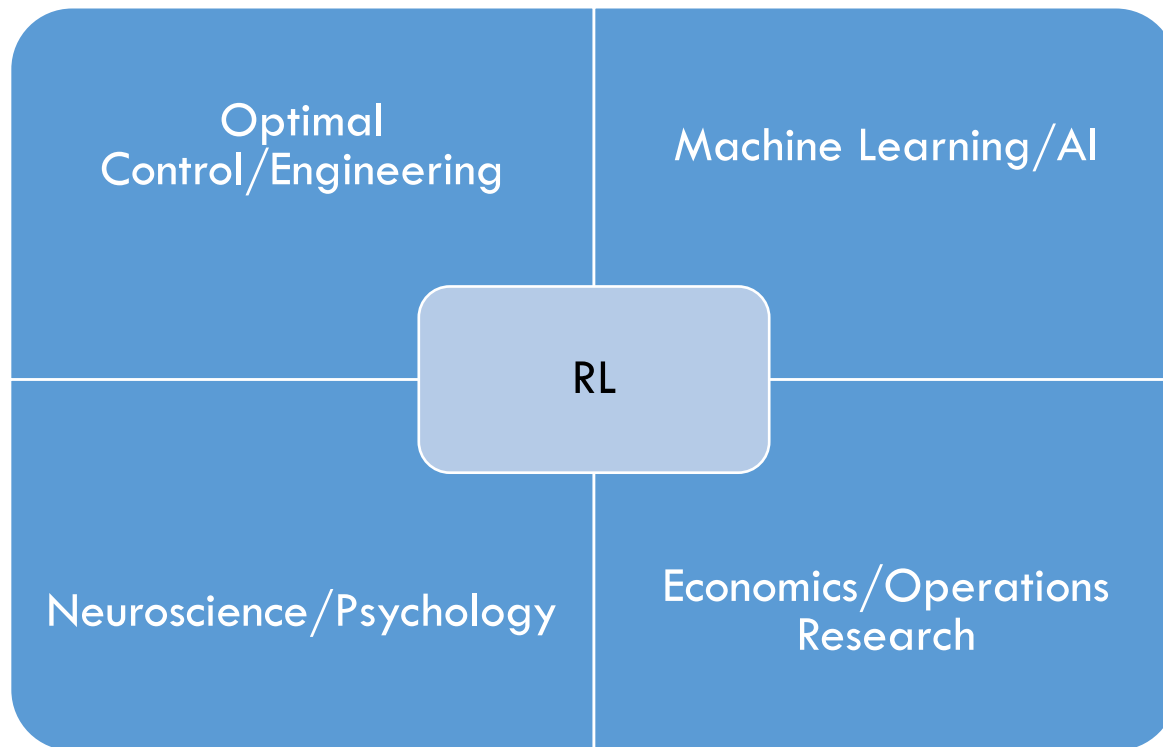
Deep Learning, Graphical Models and Reinforcement
Learning

Today's Outline

- Complex Decisions
- Reinforcement Learning Basics
 - Markov Decision Process
 - (State Action) Value Function
- Q Learning Algorithm

Complex Decisions

Complex Decisions Making is Everywhere



Control

- Fly drones
- Autonomous driving

Operations

- Retain customers, UX
- Inventory management

Logistics

- Schedule transportation
- Resource allocation

Games

- Chess, Go, Atari

Complex Decisions Making is Everywhere

Computer Go



Brain computer interface



Medical trials



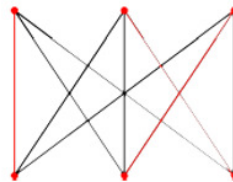
Packets routing



Ads placement



Dynamic allocation



Credit: Sebastien Bubeck

Control

- Fly drones
- Autonomous driving

Operations

- Retain customers, UX
- Inventory management

Logistics

- Schedule transportation
- Resource allocation

Games

- Chess, Go, Atari

Complex Decision Making can be addressed using RL

<https://www.technologyreview.com/s/603501/10-breakthrough-technologies-2017-reinforcement-learning/>

**MIT
Technology
Review**

Past Lists+

Topics+

Top Stories

10 Breakthrough Technologies

The List x

Years +

Reversing Paralysis

Self-Driving Trucks

Paying with Your Face

Practical Quantum Computers

The 360-Degree Selfie

Hot Solar Cells

Gene Therapy 2.0

The Cell Atlas

Botnets of Things

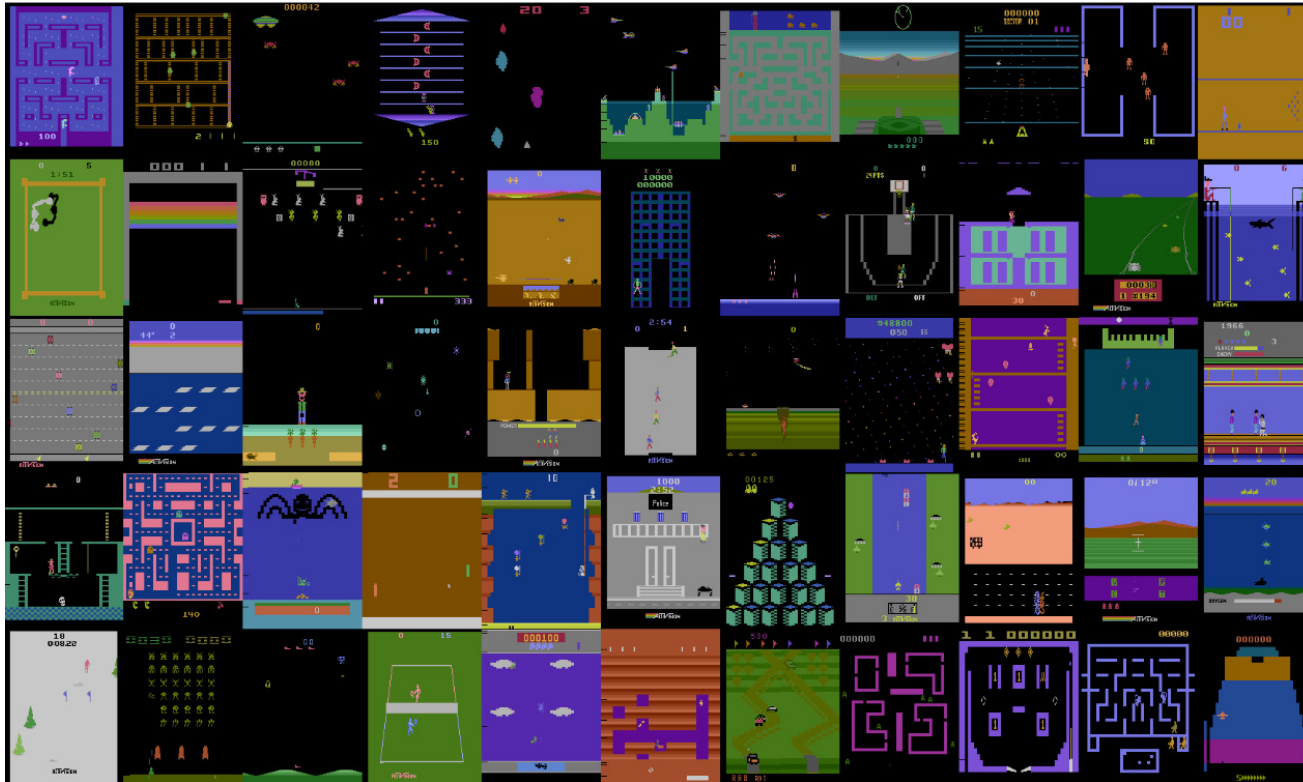
Reinforcement Learning

Reinforcement

By experimenting
figuring out how
no programmer could teach them.

March/April 2017 Issue

Playing Atari Using RL (2013)

¹Figure: Defazio Graepel, Atari Learning Environment

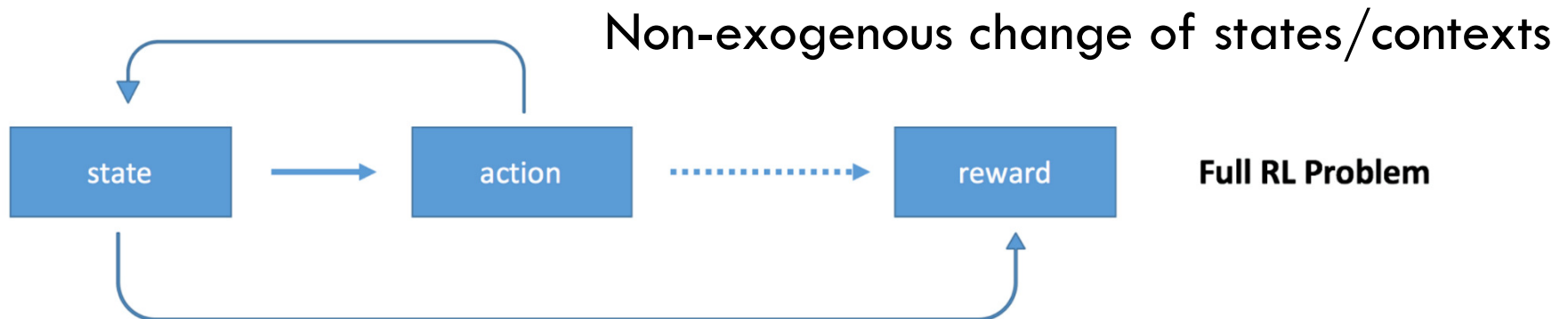
AlphaGo Conquers Go (2016)



¹Reference: DeepMind, March 2016

-
- Videos

Need for Reinforcement Learning



Questions?

Today's Outline

- Complex Decisions
- Reinforcement Learning Basics
 - Markov Decision Process
 - (State Action) Value Function
- Q Learning Algorithm

RL Overview

- Reinforcement Learning (RL) addresses a version of the problem of **sequential decision making**
- Ingredients:
 - There is an **environment**
 - Within which, an **agent** takes actions
 - This action **influences the future**
 - Agent gets a (potentially **delayed**) feedback signal
- How to select actions to maximize total reward?
- RL provides several sound answers to this question

The Environment

- Sees Agent's action A_t and generates an observation S_{t+1} and a reward R_{t+1}
- Subscript t indexes time. Current observation S_t is called **state**
- Assume the future (at times $t + 1, t + 2, \dots$) is independent of the past ($\dots, t - 2, t - 1$) given the present (t): this is called the Markov assumption

$$P(S_{t+1}|S_t) = P(S_{t+1}|S_1, S_2, \dots S_t)$$

- Assume everything relevant is observed

The Agent

- Agent observes R_{t+1}, S_{t+1} and these are not i.i.d. across time
- Agent's objective is to maximize expected total future reward $E[R_{t+1} + \gamma R_{t+2} + \dots]$
- Agent's actions affect what it sees in the future (S_{t+1})
- Maybe better to trade off current reward R_{t+1} to gain more rewards in the future

The Reward

- A **reward** R_t is a scalar feedback signal
- Indicates how well agent is doing at step t
- The agent's job is to maximise cumulative reward

Reinforcement learning is based on the **reward hypothesis**

Definition (Reward Hypothesis)

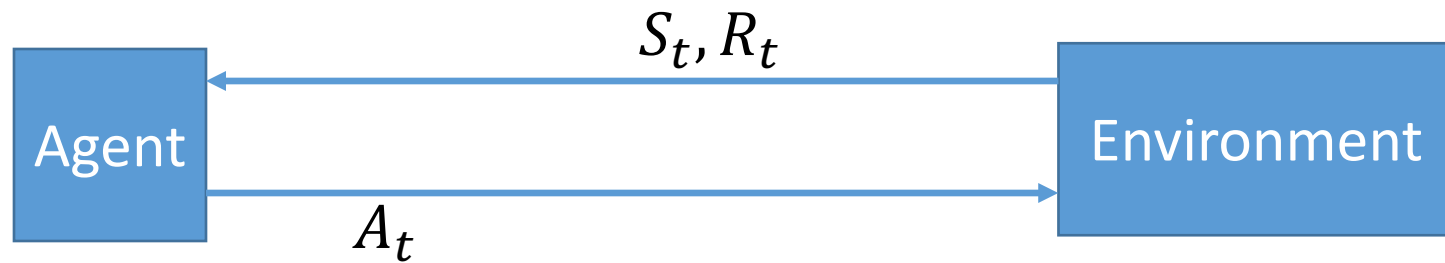
All goals can be described by the maximisation of expected cumulative reward

The Goal

- Goal: *select actions to maximise total future reward*
- Actions may have long term consequences
- Reward may be delayed
- It may be better to sacrifice immediate reward to gain more long-term reward
- Examples:
 - A financial investment (may take months to mature)
 - Refuelling a helicopter (might prevent a crash in several hours)
 - Blocking opponent moves (might help winning chances many moves from now)

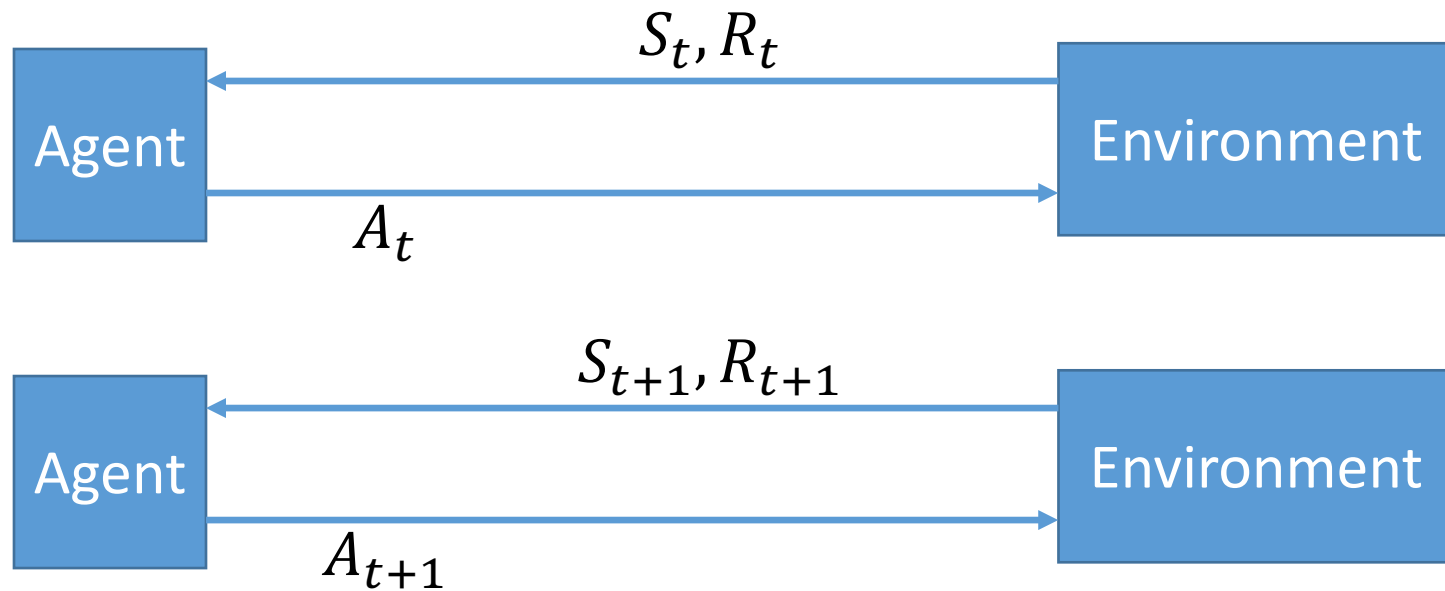
The Interactions

- Pictorially



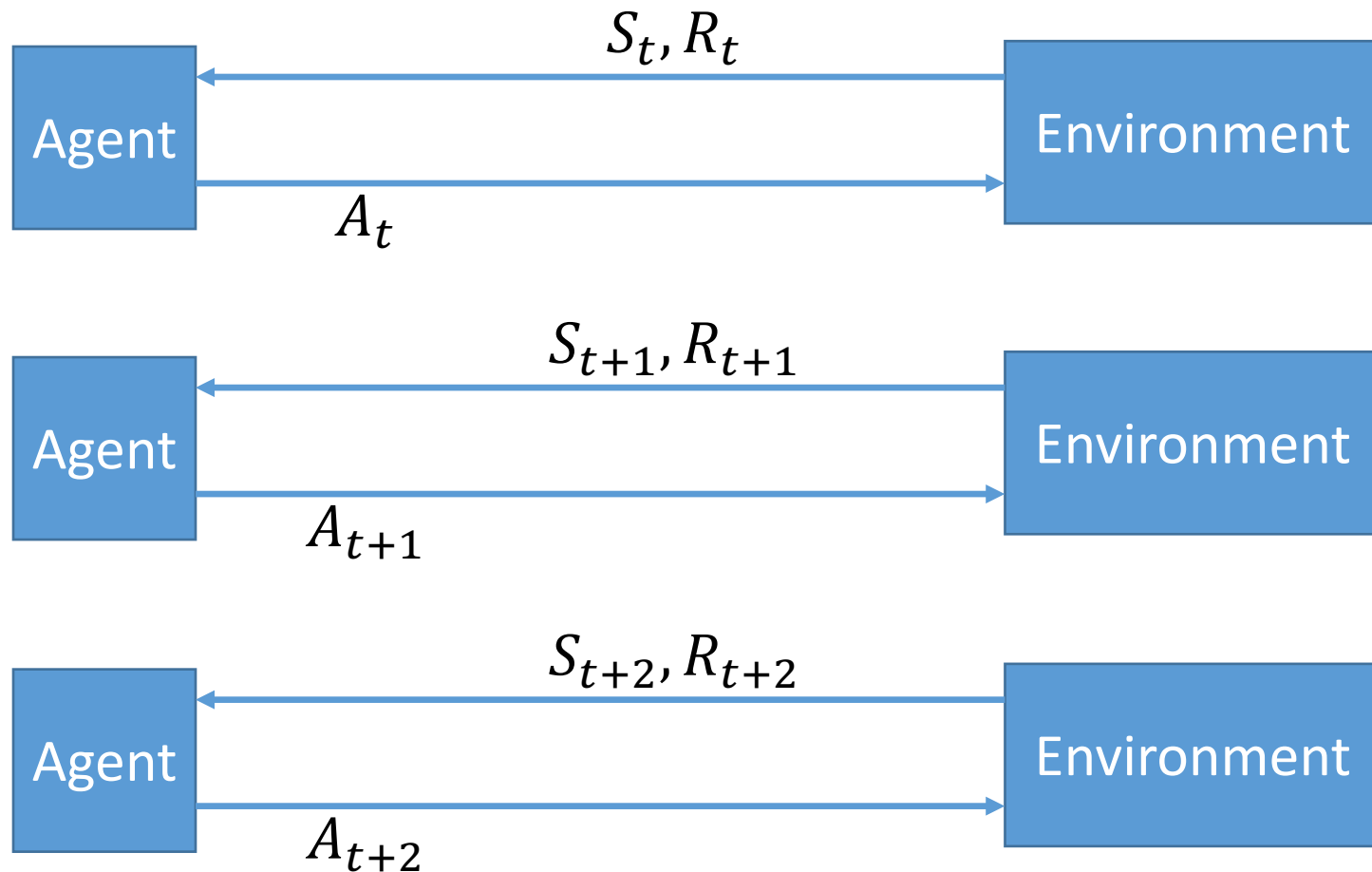
The Interactions

- Pictorially



The Interactions

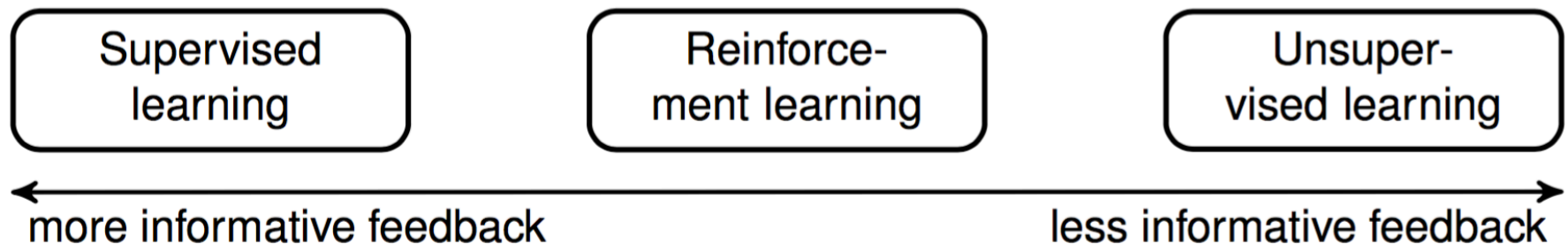
- Pictorially



RL versus other Machine Learning Settings

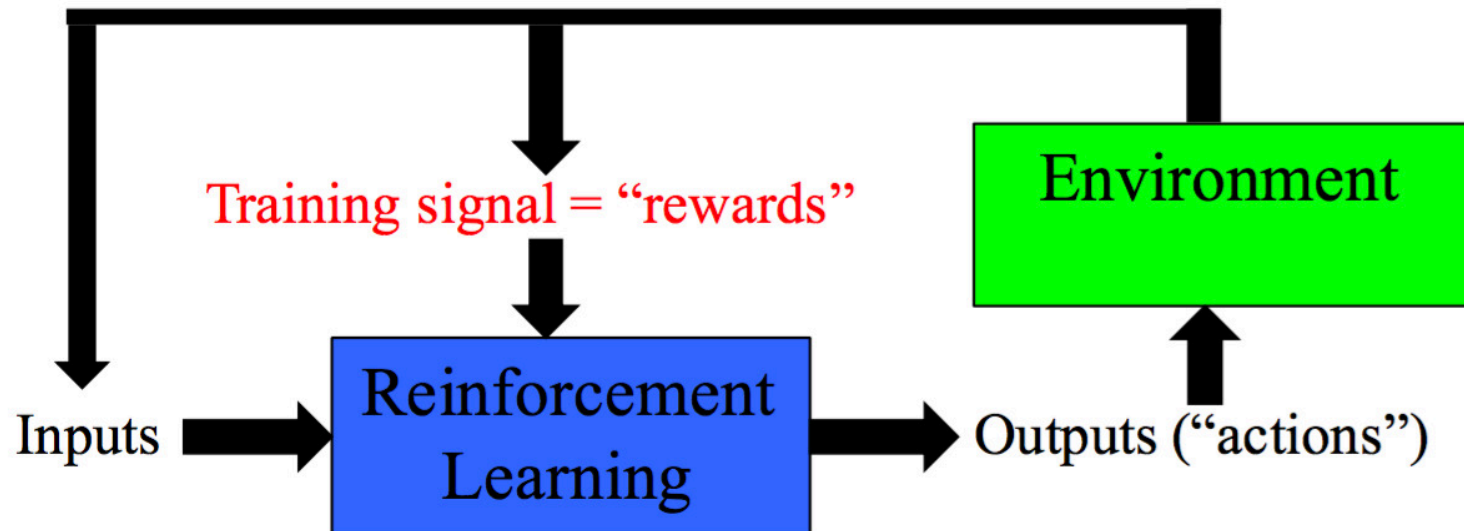
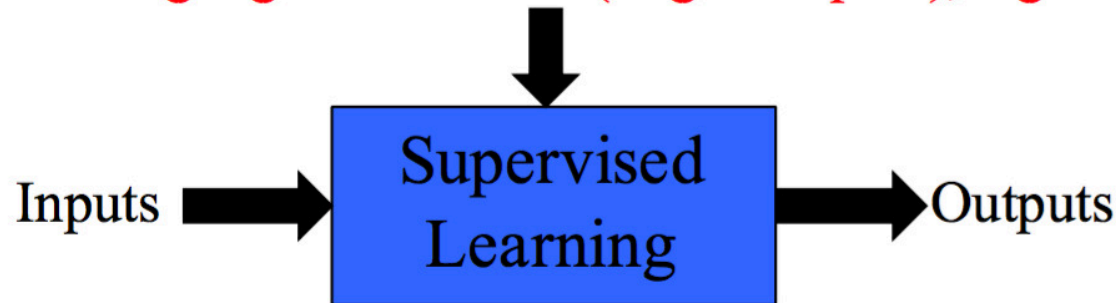
What makes reinforcement learning different from other machine learning paradigms?

- There is no supervisor, only a *reward* signal
- Feedback is delayed, not instantaneous
- Time really matters (sequential, non i.i.d data)
- Agent's actions affect the subsequent data it receives



RL versus other Machine Learning Settings

Training signal = desired (target outputs), e.g. class



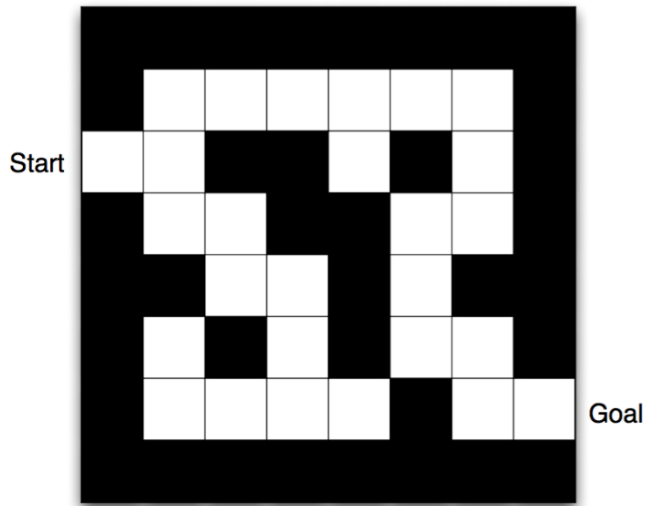
Components of an RL Agent

- An RL agent may include one or more of these components:
 - Policy: agent's behaviour function
 - Value function: how good is each state and/or action
 - Model: agent's representation of the environment

Components of RL: Policy

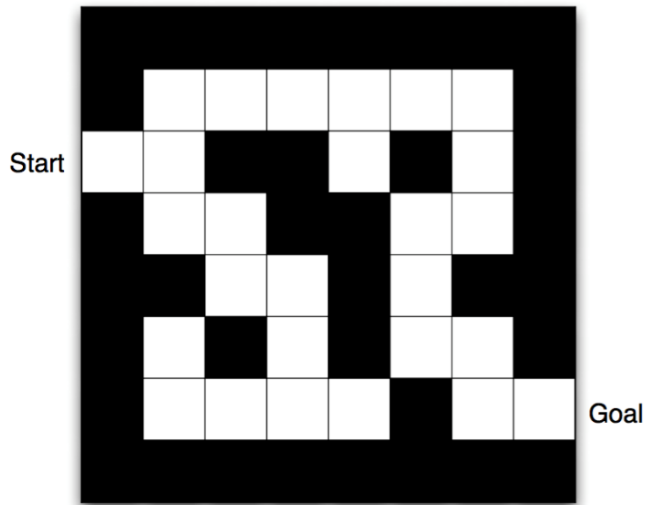
- A **policy** is the agent's behaviour
- It is a map from state to action, e.g.
- Deterministic policy: $a = \pi(s)$
- Stochastic policy: $\pi(a|s) = \mathbb{P}[A_t = a | S_t = s]$

Components of RL: Policy

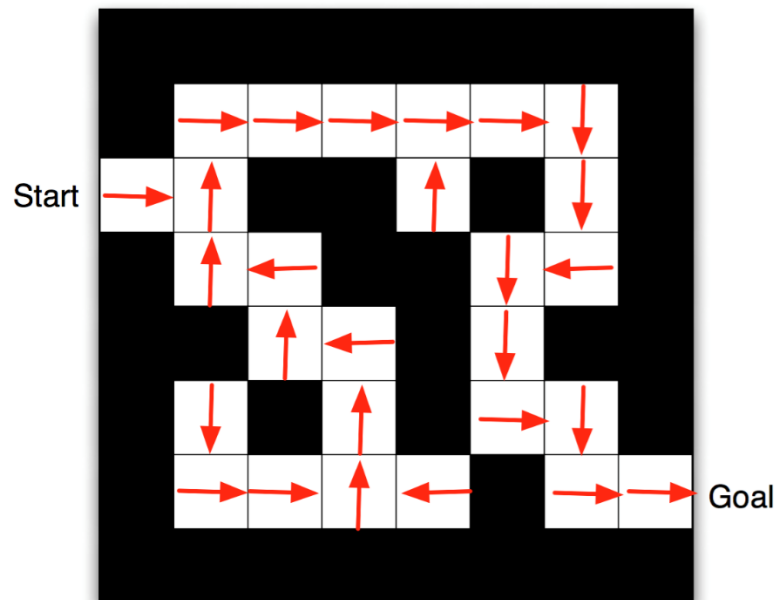


- Rewards: -1 per time-step
- Actions: N, E, S, W
- States: Agent's location

Components of RL: Policy



- Rewards: -1 per time-step
- Actions: N, E, S, W
- States: Agent's location

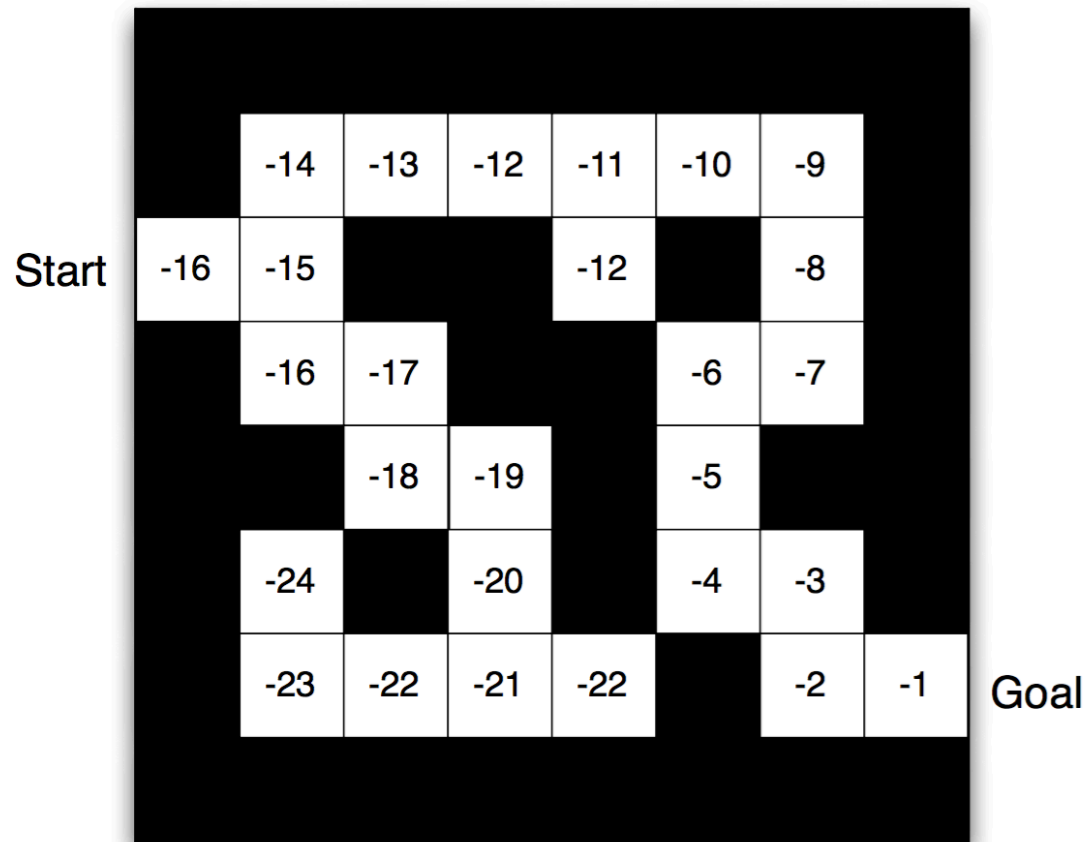


Components of RL: Value Function

- Value function is a prediction of future reward
- Used to evaluate the goodness/badness of states
- And therefore to select between actions, e.g.

$$v_{\pi}(s) = \mathbb{E}_{\pi} [R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \mid S_t = s]$$

Components of RL: Value Function



- Numbers represent value $v_{\pi}(s)$ of each state s

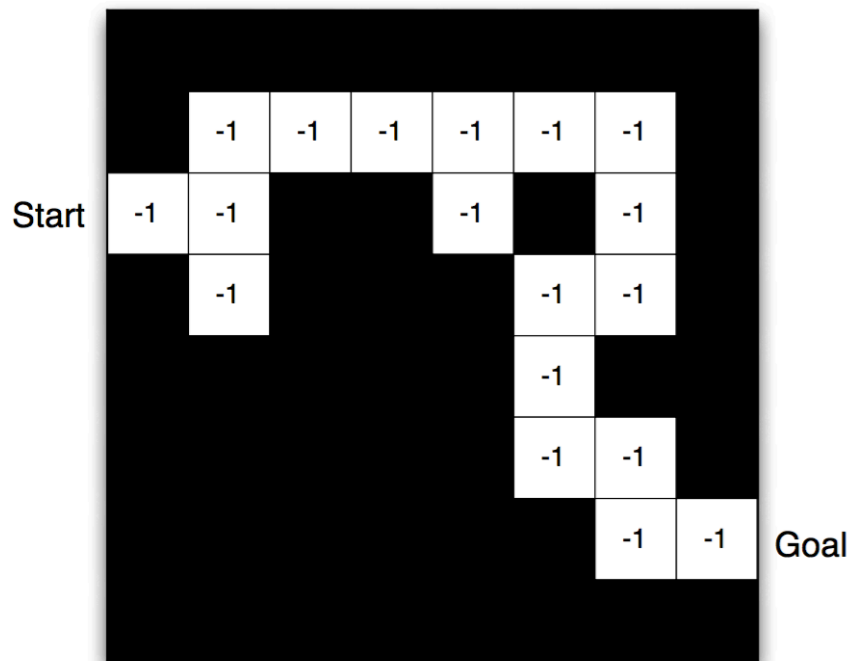
Components of RL: Model

- A **model** predicts what the environment will do next
- \mathcal{P} predicts the next state
- \mathcal{R} predicts the next (immediate) reward, e.g.

$$\mathcal{P}_{ss'}^a = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$$

$$\mathcal{R}_s^a = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$$

Components of RL: Model



- Dynamics: how actions change the state
- Rewards: how much reward from each state

- Grid layout represents transition model $\mathcal{P}_{ss'}^a$
- Numbers represent immediate reward \mathcal{R}_s^a from each state s (same for all a)

Questions?

Today's Outline

- Complex Decisions
- Reinforcement Learning Basics
 - Markov Decision Process
 - (State Action) Value Function
- Q Learning Algorithm

Components of RL: MDP Framework

- We will now revisit these components formally
 - Policy $\pi(a|s)$
 - Value function $v_{\pi}(s)$
 - Model $\mathcal{P}_{ss'}^a$ and \mathcal{R}_s^a
- In the framework of Markov Decision Processes
- And then we will address the question of optimizing for the best π in realistic environments

Towards a Markov Decision Process

- MDPs are a useful way to describe the RL problem
- MDPs can be understood via the following progression
 - Start with a Markov Chain
 - State transitions happen autonomously
 - Add Rewards
 - Becomes a Markov Reward Process
 - Add Actions that influences state transitions
 - Becomes a Markov Decision Process

Markov Chain/Process

For a Markov state s and successor state s' , the *state transition probability* is defined by

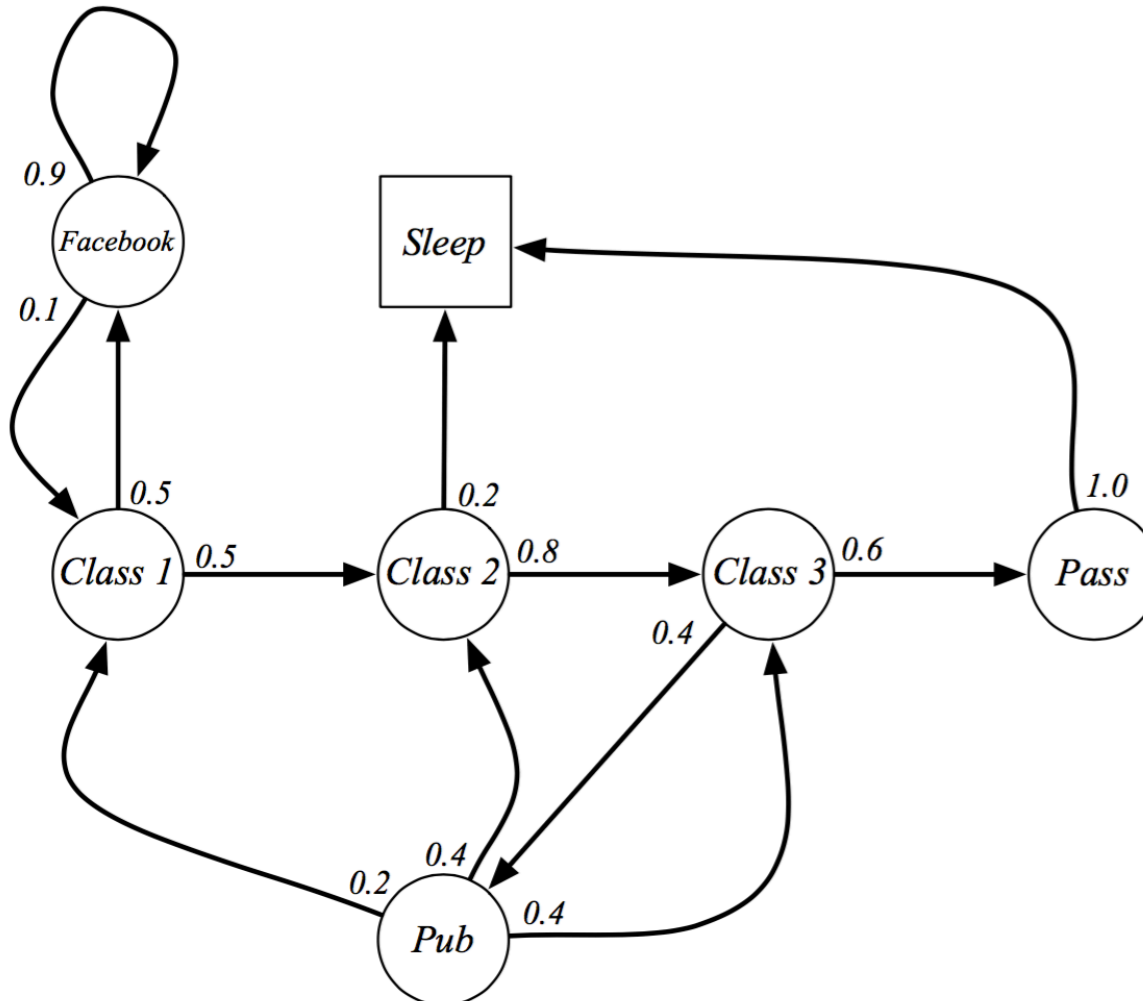
$$\mathcal{P}_{ss'} = \mathbb{P} [S_{t+1} = s' \mid S_t = s]$$

State transition matrix \mathcal{P} defines transition probabilities from all states s to all successor states s' ,

$$\mathcal{P} = \begin{matrix} & \begin{matrix} \text{to} \\ \mathcal{P}_{11} & \dots & \mathcal{P}_{1n} \\ \vdots & & \\ \mathcal{P}_{n1} & \dots & \mathcal{P}_{nn} \end{matrix} \\ \begin{matrix} \text{from} \end{matrix} & \left[\begin{array}{ccc} \mathcal{P}_{11} & \dots & \mathcal{P}_{1n} \\ \vdots & & \\ \mathcal{P}_{n1} & \dots & \mathcal{P}_{nn} \end{array} \right] \end{matrix}$$

where each row of the matrix sums to 1.

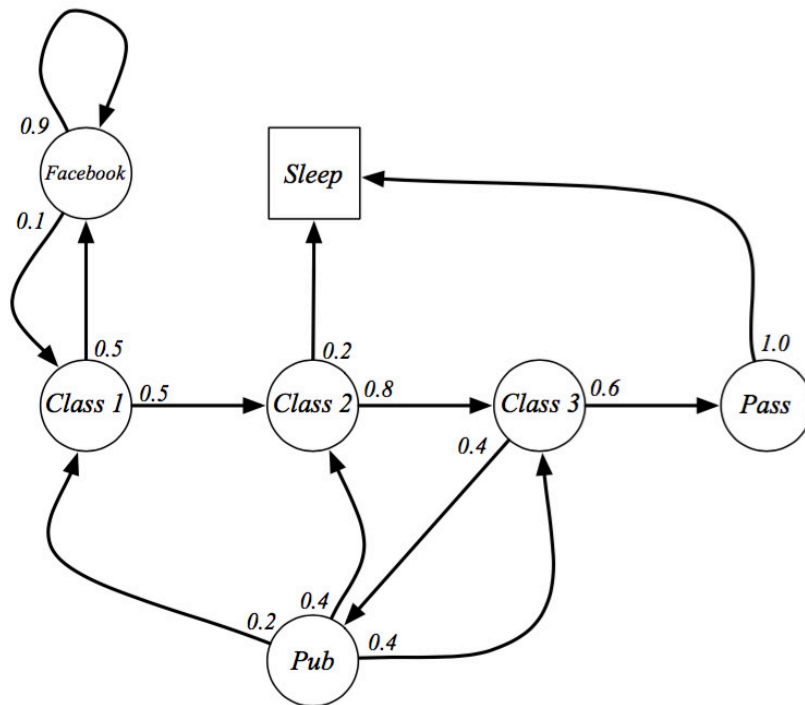
Example Markov Chain



Example Markov Chain

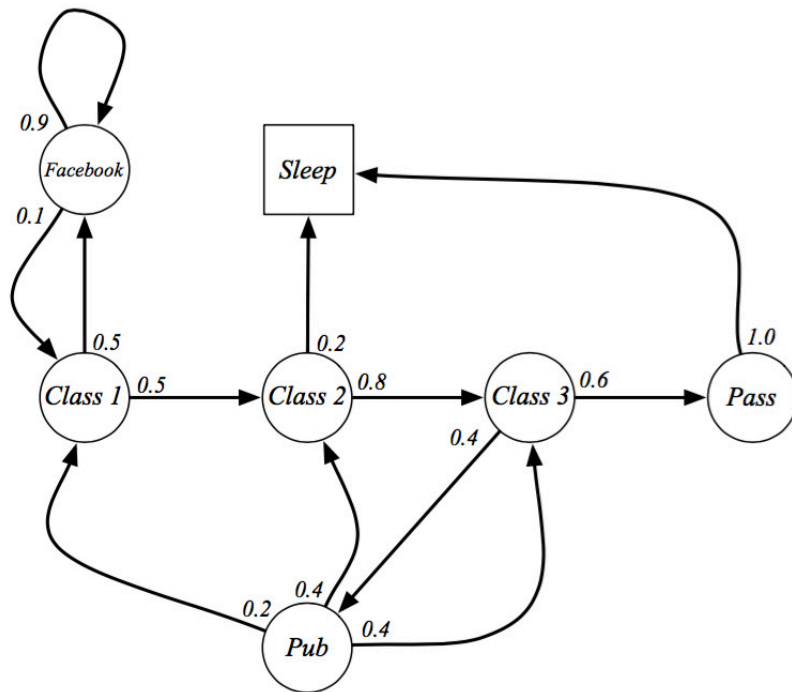
Sample **episodes** for Student Markov Chain starting from $S_1 = C1$

S_1, S_2, \dots, S_T



- C1 C2 C3 Pass Sleep
- C1 FB FB C1 C2 Sleep
- C1 C2 C3 Pub C2 C3 Pass Sleep
- C1 FB FB C1 C2 C3 Pub C1 FB FB
FB C1 C2 C3 Pub C2 Sleep

Example Markov Chain



$$\mathcal{P} = \begin{matrix} & \begin{matrix} C1 & C2 & C3 & Pass & Pub & FB & Sleep \end{matrix} \\ \begin{matrix} C1 \\ C2 \\ C3 \\ Pass \\ Pub \\ FB \\ Sleep \end{matrix} & \begin{bmatrix} & & & & & 0.5 & \\ & 0.5 & & & & & 0.2 \\ & & 0.8 & & & & \\ & & & 0.6 & 0.4 & & 1.0 \\ 0.2 & 0.4 & 0.4 & & & & \\ 0.1 & & & & & 0.9 & \\ & & & & & & 1 \end{bmatrix} \end{matrix}$$

Markov Chain with Rewards

A Markov reward process is a Markov chain with values.

Definition

A *Markov Reward Process* is a tuple $\langle \mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

- \mathcal{S} is a finite set of states
- \mathcal{P} is a state transition probability matrix,
 $\mathcal{P}_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s]$
- \mathcal{R} is a reward function, $\mathcal{R}_s = \mathbb{E}[R_{t+1} \mid S_t = s]$
- γ is a discount factor, $\gamma \in [0, 1]$

Markov Chain with Rewards

Definition

The *return* G_t is the total discounted reward from time-step t .

$$G_t = R_{t+1} + \gamma R_{t+2} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Markov Chain with Rewards

Definition

The *return* G_t is the total discounted reward from time-step t .

$$G_t = R_{t+1} + \gamma R_{t+2} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

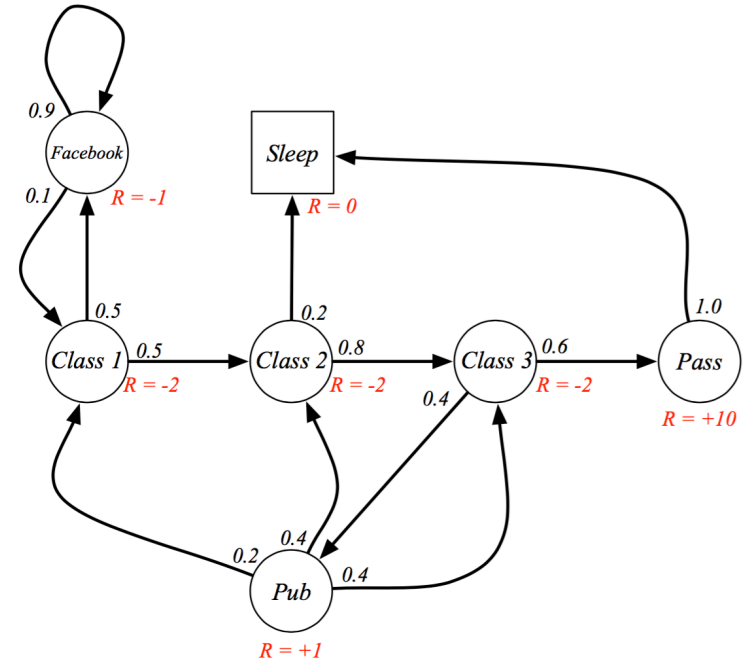
The value function $v(s)$ gives the long-term value of state s

Definition

The *state value function* $v(s)$ of an MRP is the expected return starting from state s

$$v(s) = \mathbb{E} [G_t \mid S_t = s]$$

Example Markov Reward Process



Sample **returns** for Student MRP:
Starting from $S_1 = C1$ with $\gamma = \frac{1}{2}$

$$G_1 = R_2 + \gamma R_3 + \dots + \gamma^{T-2} R_T$$

C1 C2 C3 Pass Sleep	$v_1 = -2 - 2 * \frac{1}{2} - 2 * \frac{1}{4} + 10 * \frac{1}{8}$	=	-2.25
C1 FB FB C1 C2 Sleep	$v_1 = -2 - 1 * \frac{1}{2} - 1 * \frac{1}{4} - 2 * \frac{1}{8} - 2 * \frac{1}{16}$	=	-3.125
C1 C2 C3 Pub C2 C3 Pass Sleep	$v_1 = -2 - 2 * \frac{1}{2} - 2 * \frac{1}{4} + 1 * \frac{1}{8} - 2 * \frac{1}{16} \dots$	=	-3.41
C1 FB FB C1 C2 C3 Pub C1 ...	$v_1 = -2 - 1 * \frac{1}{2} - 1 * \frac{1}{4} - 2 * \frac{1}{8} - 2 * \frac{1}{16} \dots$	=	-3.20
FB FB FB C1 C2 C3 Pub C2 Sleep			

Recursions in Markov Reward Process

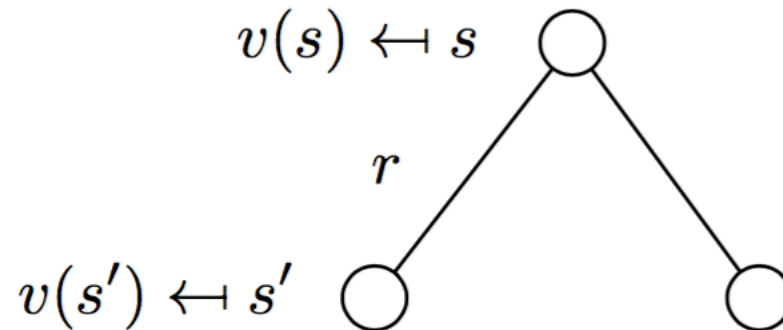
The value function can be decomposed into two parts:

- immediate reward R_{t+1}
- discounted value of successor state $\gamma v(S_{t+1})$

$$\begin{aligned} v(s) &= \mathbb{E}[G_t \mid S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \mid S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma (R_{t+2} + \gamma R_{t+3} + \dots) \mid S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s] \end{aligned}$$

Recursions in Markov Reward Process

$$v(s) = \mathbb{E} [R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s]$$



$$v(s) = \mathcal{R}_s + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'} v(s')$$

Markov Decision Process

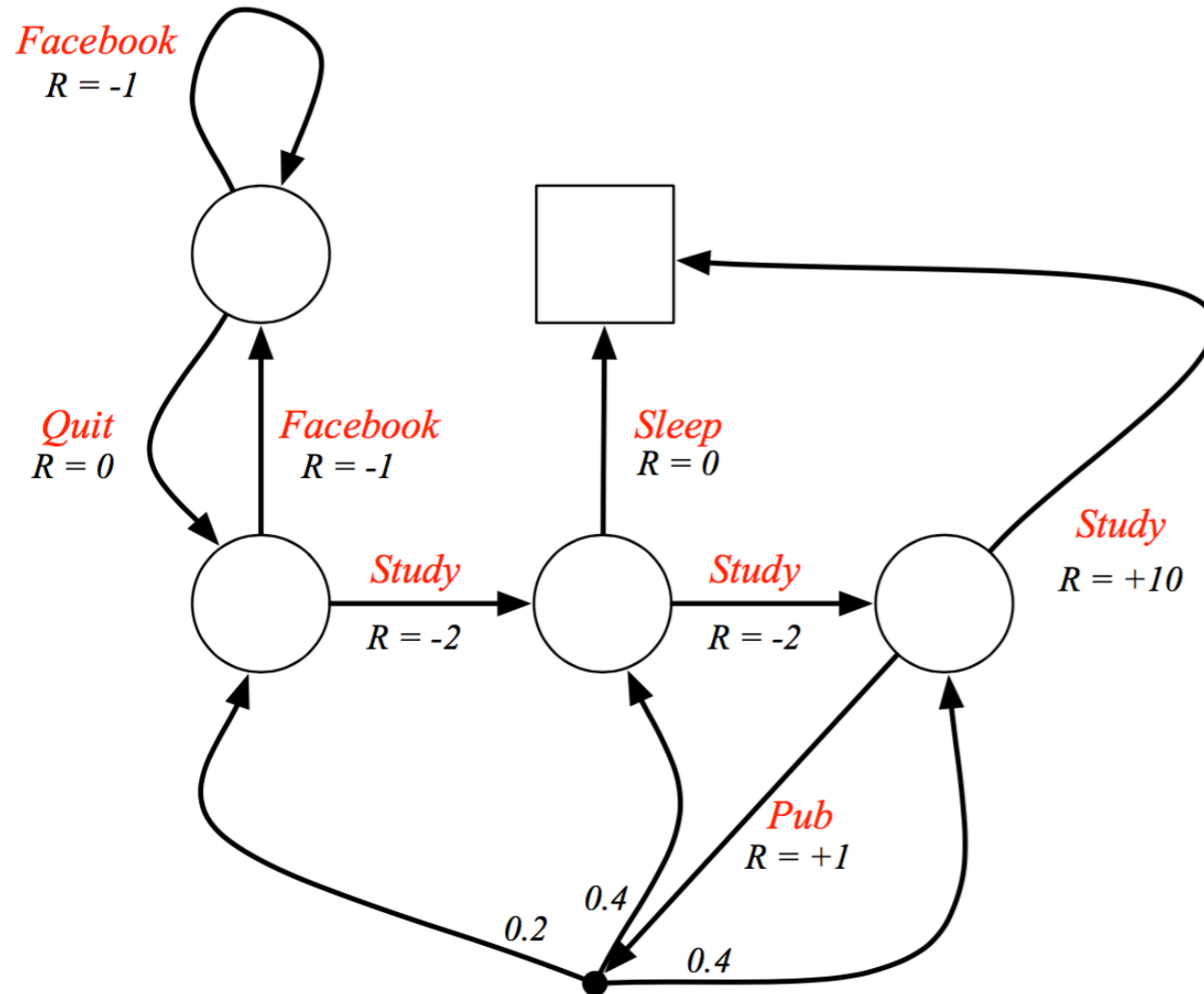
A Markov decision process (MDP) is a Markov reward process with decisions. It is an *environment* in which all states are Markov.

Definition

A *Markov Decision Process* is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

- \mathcal{S} is a finite set of states
- \mathcal{A} is a finite set of actions
- \mathcal{P} is a state transition probability matrix,
$$\mathcal{P}_{ss'}^a = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$$
- \mathcal{R} is a reward function, $\mathcal{R}_s^a = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$
- γ is a discount factor $\gamma \in [0, 1]$.

Example Markov Decision Process



Markov Decision Process: Policy

- Now that we have introduced **actions**, we can discuss **policies** again
- Recall

Definition

A *policy* π is a distribution over actions given states,

$$\pi(a|s) = \mathbb{P}[A_t = a \mid S_t = s]$$

- A policy fully defines the behaviour of an agent

MDP is an MRP for a Fixed Policy

- Given an MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ and a policy π
- The state sequence S_1, S_2, \dots is a Markov process $\langle \mathcal{S}, \mathcal{P}^\pi \rangle$
- The state and reward sequence S_1, R_2, S_2, \dots is a Markov reward process $\langle \mathcal{S}, \mathcal{P}^\pi, \mathcal{R}^\pi, \gamma \rangle$

MDP is an MRP for a Fixed Policy

- Given an MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ and a policy π
- The state sequence S_1, S_2, \dots is a Markov process $\langle \mathcal{S}, \mathcal{P}^\pi \rangle$
- The state and reward sequence S_1, R_2, S_2, \dots is a Markov reward process $\langle \mathcal{S}, \mathcal{P}^\pi, \mathcal{R}^\pi, \gamma \rangle$
- where

$$\mathcal{P}_{s,s'}^\pi = \sum_{a \in \mathcal{A}} \pi(a|s) \mathcal{P}_{ss'}^a$$

$$\mathcal{R}_s^\pi = \sum_{a \in \mathcal{A}} \pi(a|s) \mathcal{R}_s^a$$

Markov Decision Process: Value Function

- We can also talk about the **value function(s)**

Definition

The *state-value function* $v_{\pi}(s)$ of an MDP is the expected return starting from state s , and then following policy π

$$v_{\pi}(s) = \mathbb{E}_{\pi} [G_t \mid S_t = s]$$

Markov Decision Process: Value Function

- We can also talk about the value function(s)

Definition

The *state-value function* $v_\pi(s)$ of an MDP is the expected return starting from state s , and then following policy π

$$v_\pi(s) = \mathbb{E}_\pi [G_t \mid S_t = s]$$

Definition

The *action-value function* $q_\pi(s, a)$ is the expected return starting from state s , taking action a , and then following policy π

$$q_\pi(s, a) = \mathbb{E}_\pi [G_t \mid S_t = s, A_t = a]$$

Recursions in MDP

*Also called the **Bellman Expectation Equations**

The state-value function can again be decomposed into immediate reward plus discounted value of successor state,

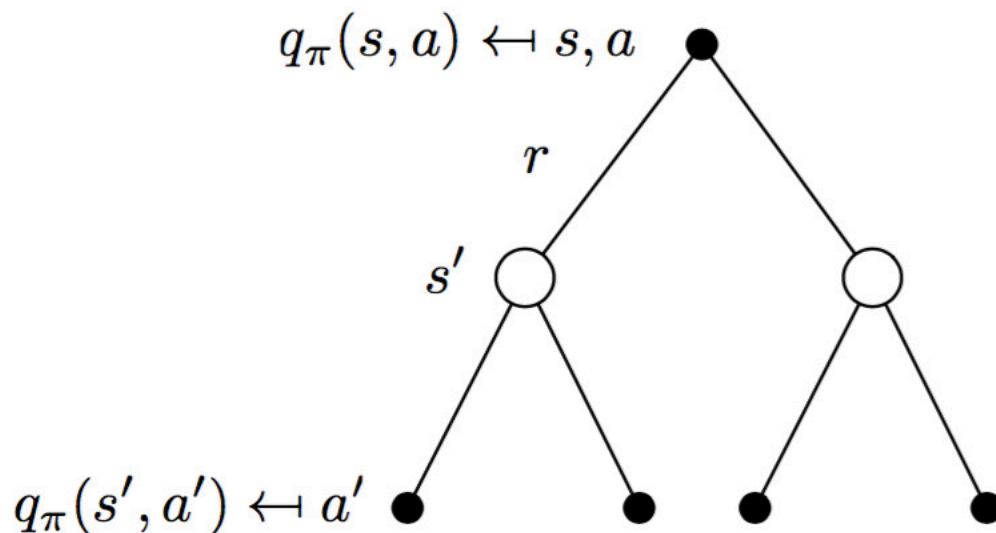
$$v_{\pi}(s) = \mathbb{E}_{\pi} [R_{t+1} + \gamma v_{\pi}(S_{t+1}) \mid S_t = s]$$

The action-value function can similarly be decomposed,

$$q_{\pi}(s, a) = \mathbb{E}_{\pi} [R_{t+1} + \gamma q_{\pi}(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a]$$

Recursions in MDP

*Also called the **Bellman Expectation Equations**



$$q_{\pi}(s, a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \sum_{a' \in \mathcal{A}} \pi(a'|s') q_{\pi}(s', a')$$

Markov Decision Process: Objective

The *optimal state-value function* $v_*(s)$ is the maximum value function over all policies

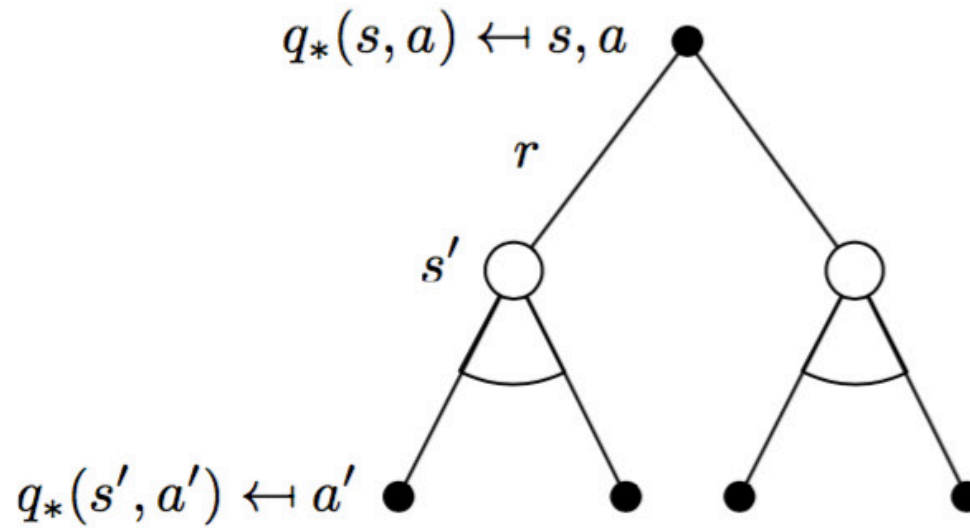
$$v_*(s) = \max_{\pi} v_{\pi}(s)$$

The *optimal action-value function* $q_*(s, a)$ is the maximum action-value function over all policies

$$q_*(s, a) = \max_{\pi} q_{\pi}(s, a)$$

Markov Decision Process: Objective

*Also called the **Bellman Optimality Equation**



$$q_*(s, a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \max_{a'} q_*(s', a')$$

Markov Decision Process: Optimal Policy

An optimal policy can be found by maximising over $q_*(s, a)$,

$$\pi_*(a|s) = \begin{cases} 1 & \text{if } a = \operatorname{argmax}_{a \in \mathcal{A}} q_*(s, a) \\ 0 & \text{otherwise} \end{cases}$$

Questions?

Today's Outline

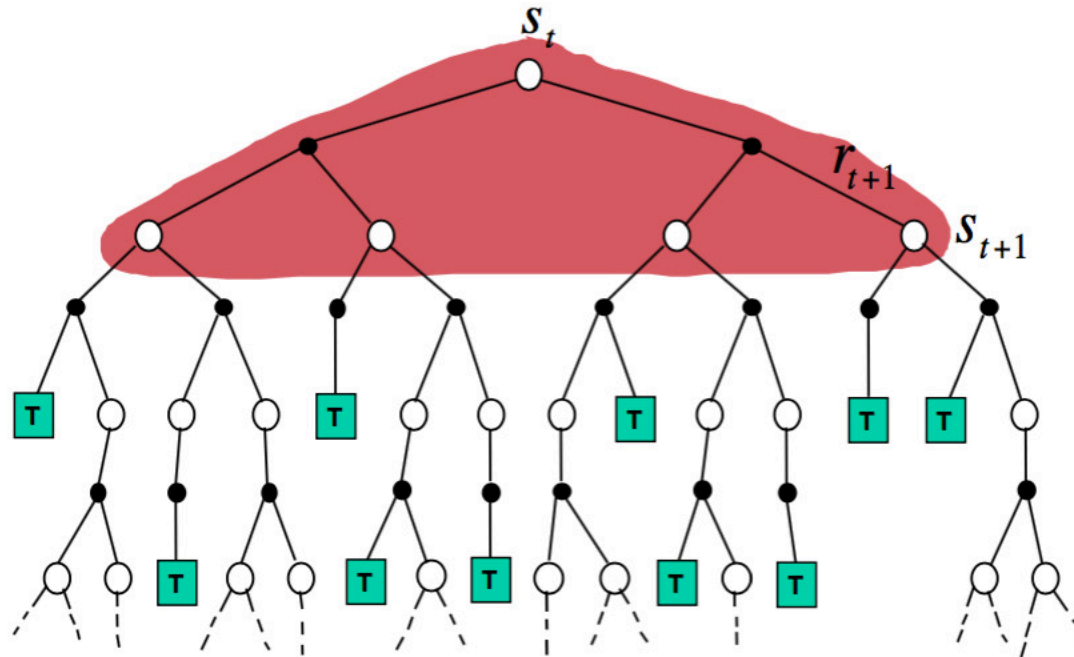
- Complex Decisions
- Reinforcement Learning Basics
 - Markov Decision Process
 - (State Action) Value Function
- Q Learning Algorithm

Finding the Best Policy

- Need to be able to do two things ideally
 - Prediction:
 - For a given policy, evaluate how good it is
 - Compute $q_{\pi}(s, a)$
 - Control:
 - And make an improvement from π
- We will focus on the Q Learning algorithm
 - It does prediction and control ‘simultaneously’

Intuition for an Iterative Algorithm

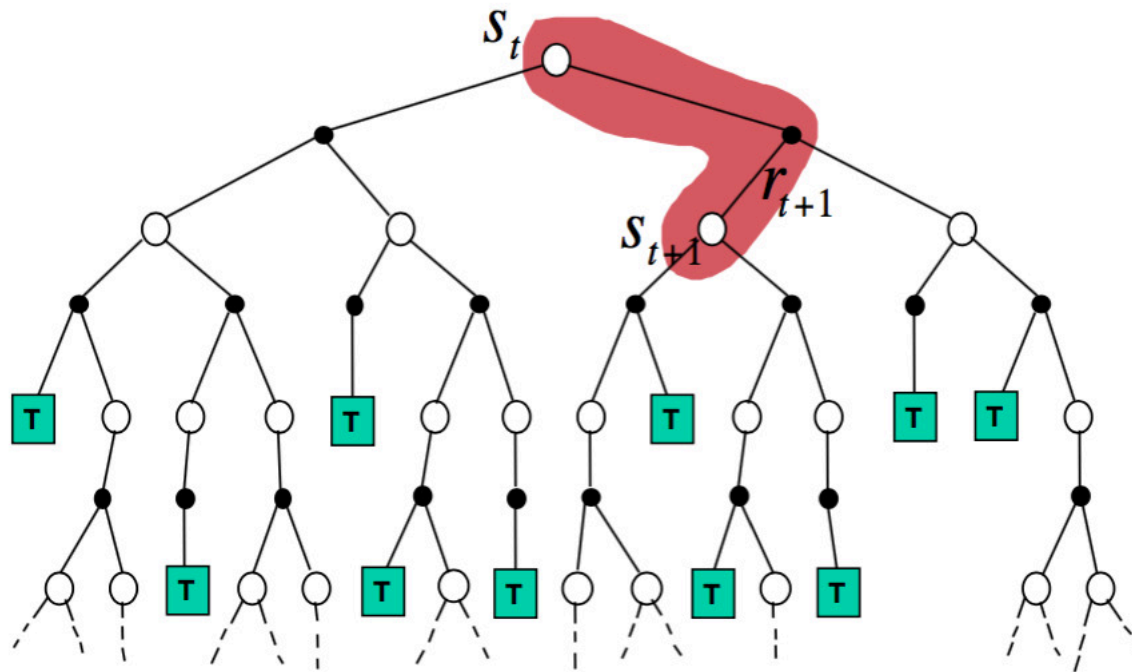
$$V(S_t) \leftarrow \mathbb{E}_\pi [R_{t+1} + \gamma V(S_{t+1})]$$



$$Q(s, a) \leftarrow \mathbb{E} \left[R + \gamma \max_{a' \in \mathcal{A}} Q(S', a') \mid s, a \right]$$

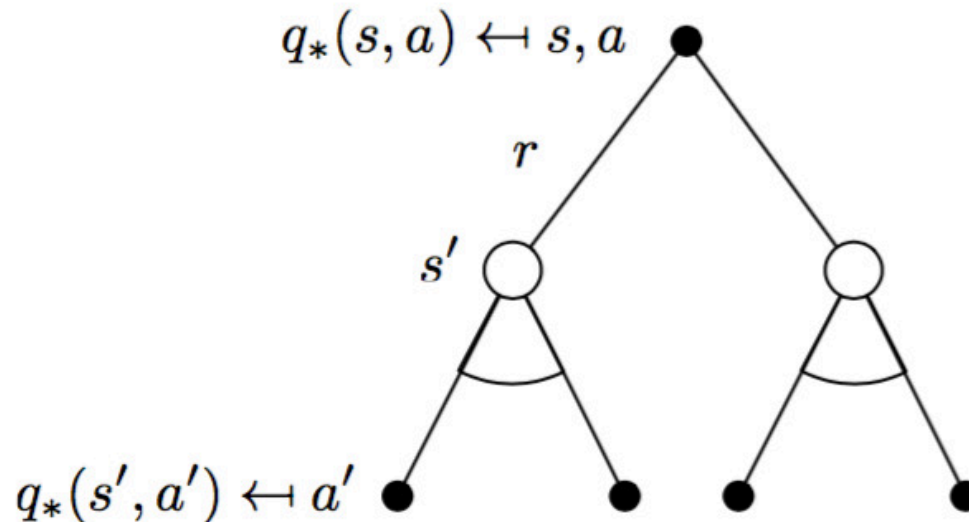
Intuition for an Iterative Algorithm

$$V(S_t) \leftarrow V(S_t) + \alpha (R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$$



The Q Learning Algorithm

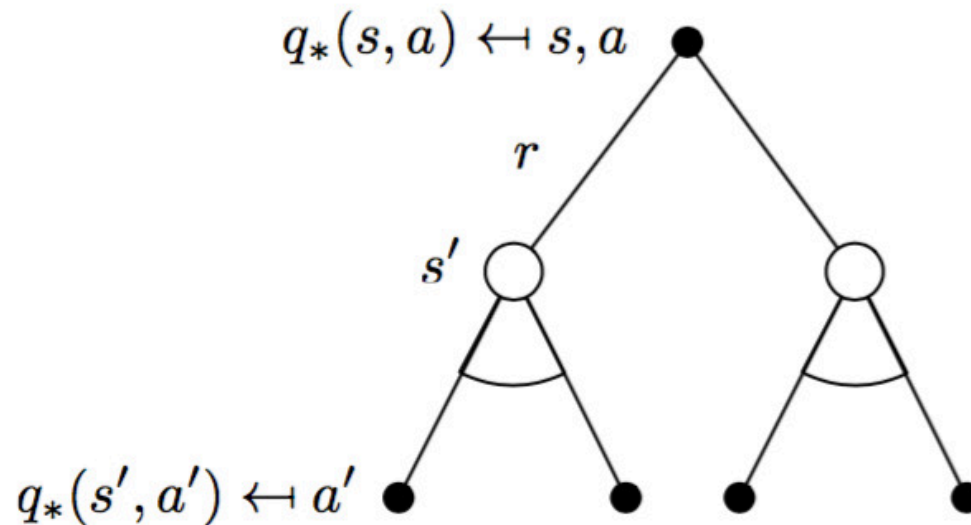
- If we know the model
 - Turn the Bellman Optimality Equation into an **iterative update**
 - This is called Value Iteration



$$q_*(s, a) \boxed{=} \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \max_{a'} q_*(s', a')$$

The Q Learning Algorithm

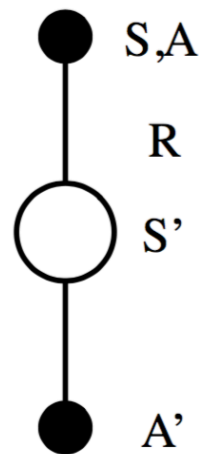
- If we do not know the model
 - Do **sampling** to get an **incremental** iterative update
 - Choose next actions to ensure **exploration**



$$q_*(s, a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \max_{a'} q_*(s', a')$$

The Q Learning Algorithm

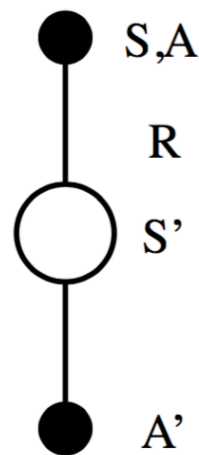
- If we do not know the model
 - Do **sampling** to get an **incremental** iterative update
 - Choose next actions to ensure **exploration**



$$Q(S, A) \leftarrow Q(S, A) + \alpha (R + \gamma Q(S', A') - Q(S, A))$$

The Q Learning Algorithm

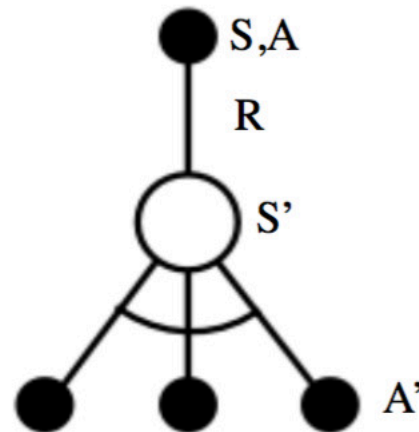
- If we do not know the model
 - Do **sampling** to get an **incremental** iterative update
 - Choose next actions to ensure **exploration**



$$Q(S, A) \leftarrow Q(S, A) + \alpha (R + \gamma Q(S', A') - Q(S, A))$$

The Q Learning Algorithm

- If we do not know the model
 - Do **sampling** to get an **incremental** iterative update
 - Choose next actions to ensure **exploration**



$$Q(S, A) \leftarrow Q(S, A) + \alpha \left(R + \gamma \max_{a'} Q(S', a') - Q(S, A) \right)$$

The Q Learning Algorithm

- Initialize Q , which is a **table** of size $\#states \times \#actions$
- Start at state s_1

The Q Learning Algorithm

- Initialize Q , which is a **table** of size $\#states \times \#actions$
- Start at state s_1
- For $t = 1, 2, 3, \dots$
 - Take A_t chosen uniformly at random with probability ϵ

Explore

The Q Learning Algorithm

- Initialize Q , which is a **table** of size $\#states \times \#actions$
- Start at state s_1
- For $t = 1, 2, 3, \dots$
 - Take A_t chosen uniformly at random with probability ϵ
 - Take $\operatorname{argmax}_{a \in A} Q(S_t, a)$ with probability $1 - \epsilon$

Explore

Exploit

The Q Learning Algorithm

- Initialize Q , which is a **table** of size $\#states \times \#actions$
- Start at state s_1
- For $t = 1, 2, 3, \dots$
 - Take A_t chosen uniformly at random with probability ϵ Explore
 - Take $\operatorname{argmax}_{a \in A} Q(S_t, a)$ with probability $1 - \epsilon$ Exploit
 - Update Q :
 - $Q(S_t, A_t) = Q(S_t, A_t) + \alpha_t \underbrace{(R_{t+1} + \gamma \max_{a \in A} Q(S_{t+1}, a) - Q(S_t, A_t))}_{\text{Temporal difference error}}$

The Q Learning Algorithm

- Initialize Q , which is a **table** of size $\#states \times \#actions$
- Start at state s_1
- For $t = 1, 2, 3, \dots$
 - Take A_t chosen uniformly at random with probability ϵ Explore
 - Take $\operatorname{argmax}_{a \in A} Q(S_t, a)$ with probability $1 - \epsilon$ Exploit
 - Update Q :
 - $Q(S_t, A_t) = Q(S_t, A_t) + \alpha_t \underbrace{(R_{t+1} + \gamma \max_{a \in A} Q(S_{t+1}, a) - Q(S_t, A_t))}_{\text{Temporal difference error}}$
- Parameter ϵ is the exploration parameter
- Parameter α_t is the learning rate

The Q Learning Algorithm

- Initialize Q , which is a **table** of size $\#states \times \#actions$
- Start at state s_1
- For $t = 1, 2, 3, \dots$
 - Take A_t chosen uniformly at random with probability ϵ Explore
 - Take $\operatorname{argmax}_{a \in A} Q(S_t, a)$ with probability $1 - \epsilon$ Exploit
 - Update Q :
 - $Q(S_t, A_t) = Q(S_t, A_t) + \alpha_t \underbrace{(R_{t+1} + \gamma \max_{a \in A} Q(S_{t+1}, a) - Q(S_t, A_t))}_{\text{Temporal difference error}}$
- Parameter ϵ is the exploration parameter
- Parameter α_t is the learning rate
- Under appropriate assumptions¹, $\lim_{t \rightarrow \infty} Q = Q^*$

¹Reference: Christopher J. C. H. Watkins and Peter Dayan, 1992

The Q Learning Algorithm: Recap

- Bellman Optimality Equation gives rise to the Q-Value Iteration algorithm
- Making this algorithm incremental, sampled and adding ϵ -greedy exploration gives Q Learning Algorithm

Q-Value Iteration		Q-Learning
$Q(s, a) \leftarrow \mathbb{E} \left[R + \gamma \max_{a' \in \mathcal{A}} Q(S', a') \mid s, a \right]$		$Q(S, A) \stackrel{\alpha}{\leftarrow} R + \gamma \max_{a' \in \mathcal{A}} Q(S', a')$
		where $x \stackrel{\alpha}{\leftarrow} y \equiv x \leftarrow x + \alpha(y - x)$

Questions?

Summary

- RL is a great framework to make agents intelligent
- Specify goals and provide feedback
- Many challenges still remain: exciting opportunity to contribute towards next generation of artificially intelligent and autonomous agents.
- In the next lecture, we will see that **deep learning function approximation based RL** agents show promise in large complex tasks: **representations** matter!
 - Applications such as
 - Self-driving cars
 - Intelligent virtual agents

Appendix

Additional Resources

- An Introduction to Reinforcement Learning by Richard Sutton and Andrew Barto
 - <http://incompleteideas.net/sutton/book/the-book.html>
- Course on Reinforcement Learning by David Silver at UCL (includes video lectures)
 - <http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html>
- Research Papers
 - Deep RL collection: <https://github.com/junhyukoh/deep-reinforcement-learning-papers>
 - [MKS RVBGR FOPBSAKKWLH2015] Mnih et al. Human-level control through deep reinforcement learning. Nature, 518:529–533, 2015.
 - [SHMGSDSAPLDGNKSLLKGH2016] Silver et al. Mastering the game of Go with deep neural networks and tree search. Nature, 529: 484–489, 2016.

Cons of RL

- Reinforcement Learning requires experiencing the environment many many times
- This is because it is a trial and error based approach
- Impractical for many complex tasks
- Unless one has access to simulators where an RL agent can practice a billion times

RL versus other Machine Learning Settings

- There is a notion of exploration and exploitation, similar to Multi-armed bandits and Contextual bandits
 - *Exploration* finds more information about the environment
 - *Exploitation* exploits known information to maximise reward
 - It is usually important to explore as well as exploit
- Key difference: actions influence future contexts
 - Reinforcement learning is like trial-and-error learning
 - The agent should discover a good policy
 - From its experiences of the environment
 - Without losing too much reward along the way

RL versus other Sequential Decision Making Settings

Two fundamental problems in sequential decision making

- Reinforcement Learning:

- The environment is initially unknown
- The agent interacts with the environment
- The agent improves its policy

- Planning:

- A model of the environment is known
- The agent performs computations with its model (without any external interaction)
- The agent improves its policy
- a.k.a. deliberation, reasoning, introspection, pondering, thought, search

Types of RL Agents

- There are many ways to design them, so we roughly categorize them as below:

- Value Based

- No Policy (Implicit)
- Value Function

- Policy Based

- Policy
- No Value Function

- Actor Critic

- Policy
- Value Function

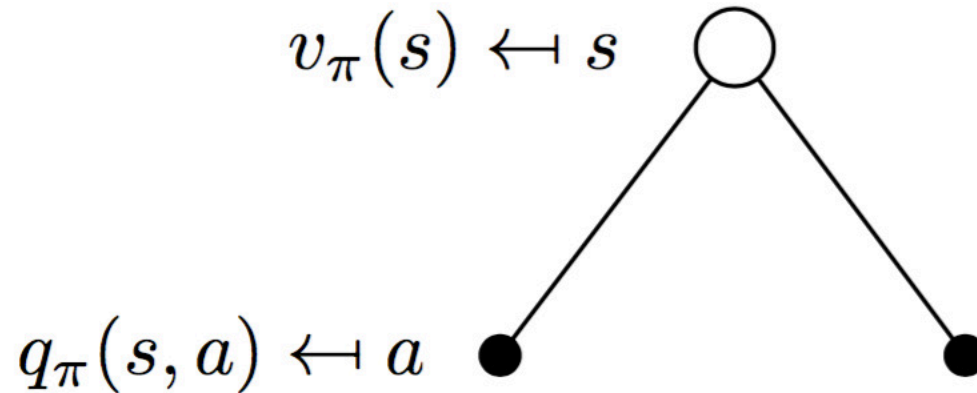
- Model Free

- Policy and/or Value Function
- No Model

- Model Based

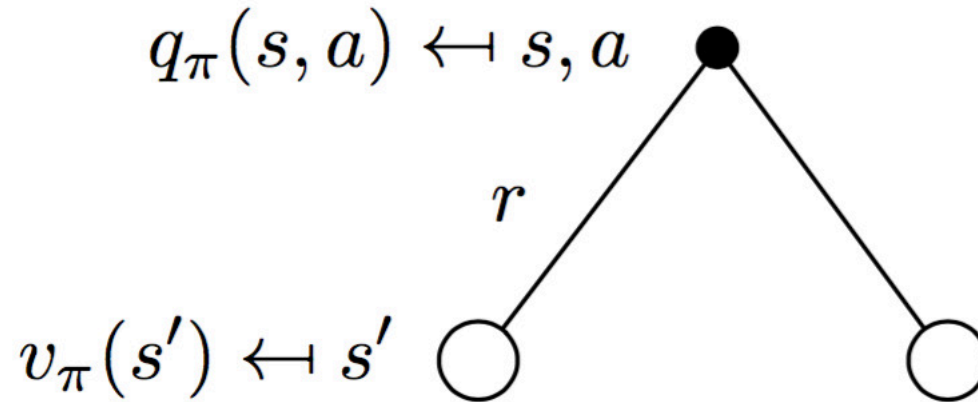
- Policy and/or Value Function
- Model

Relating the Two Value Functions I



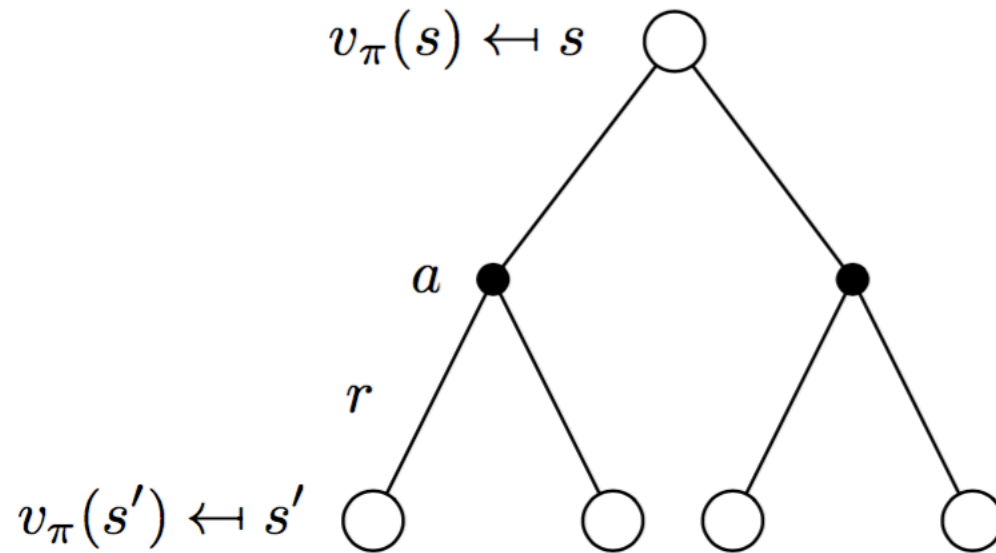
$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s, a)$$

Relating the Two Value Functions II



$$q_\pi(s, a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s')$$

Recursion in MDP: Value Function Version



$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s') \right)$$

Relating Policy and Value Function

An optimal policy can be found by maximising over $q_*(s, a)$,

$$\pi_*(a|s) = \begin{cases} 1 & \text{if } a = \operatorname{argmax}_{a \in \mathcal{A}} q_*(s, a) \\ 0 & \text{otherwise} \end{cases}$$