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

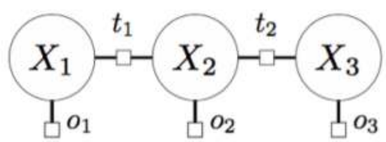
- Given all that we have learned up to now, we will sample the following applications

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

[1]Reference: Percy Liang, CS221 (2015)

Example Graphical Model II

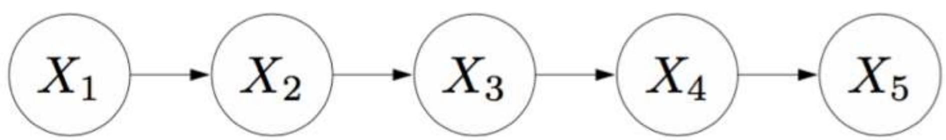
- A generative process is nothing but a description of the joint distribution in terms of how the random variables realize

Probabilistic program:

Probabilistic program: object tracking

$X_0 = (0,0)$
For each time step $i = 1, \ldots, n$:
  With probability α:
    $X_i = X_{i-1} + (1,0)$ [go right]
  With probability $1 - \alpha$:
    $X_i = X_{i-1} + (0,1)$ [go down]

Bayesian network:

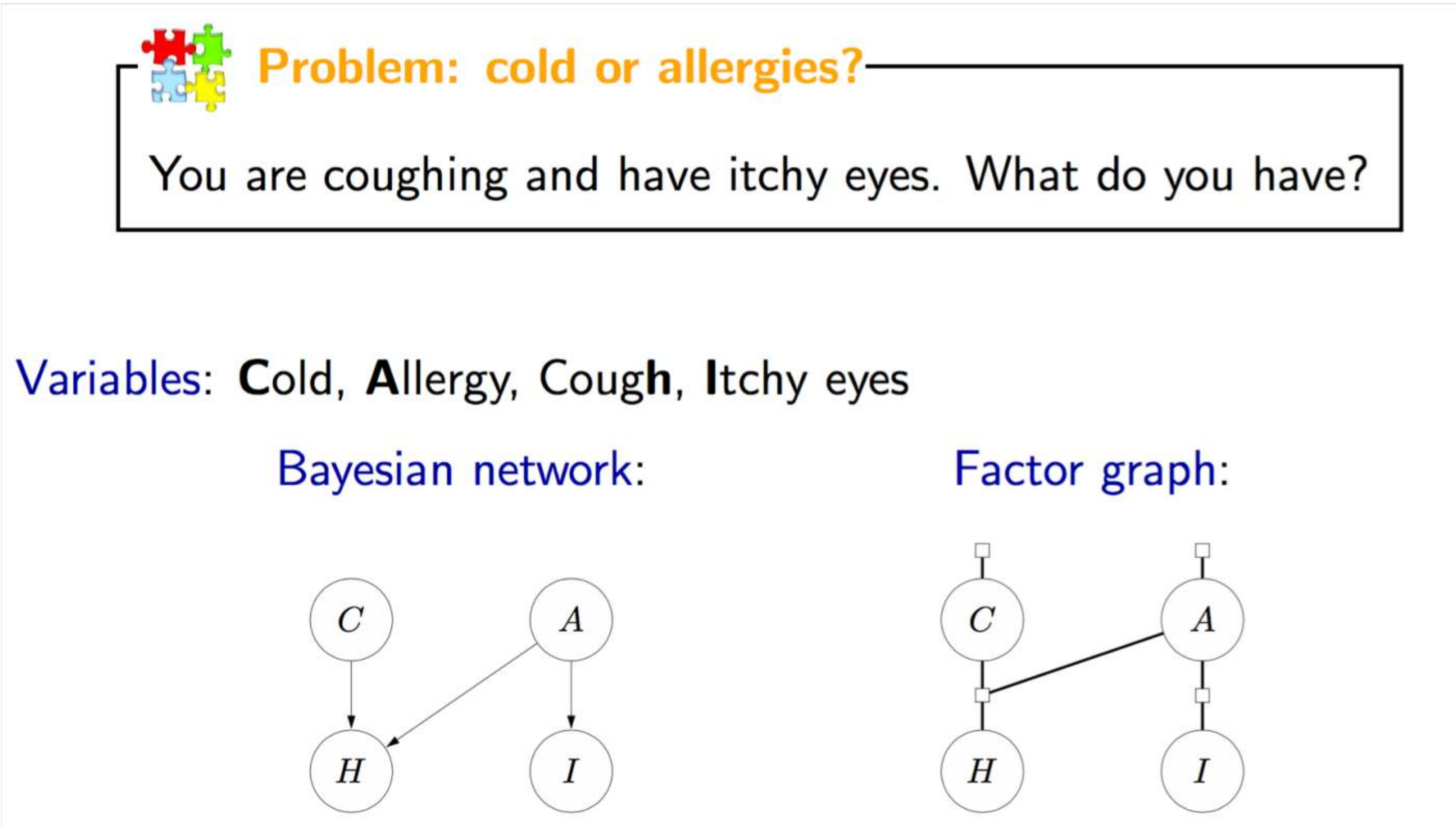


Mathematical definition:

$$p(x_i \mid x_{i-1}) = \alpha \cdot \underbrace{[x_i = x_{i-1} + (1,0)]}_{\text{right}} + (1-\alpha) \cdot \underbrace{[x_i = x_{i-1} + (0,1)]}_{\text{down}}$$

[1]Reference: Percy Liang, CS221 (2015)

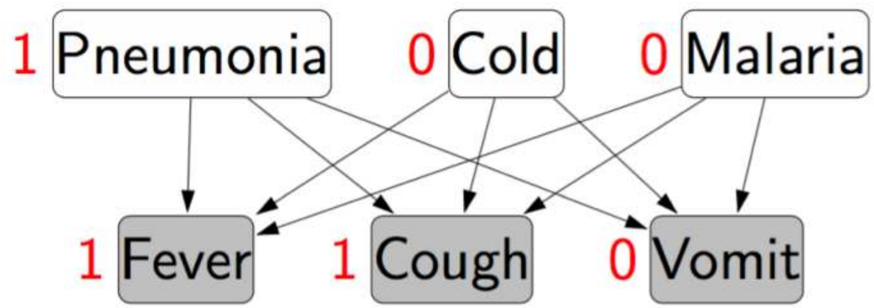
Example Graphical Model III



[1]Reference: Percy Liang, CS221 (2015)

Example Graphical Model IV

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



Probabilistic program: diseases and symptoms

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

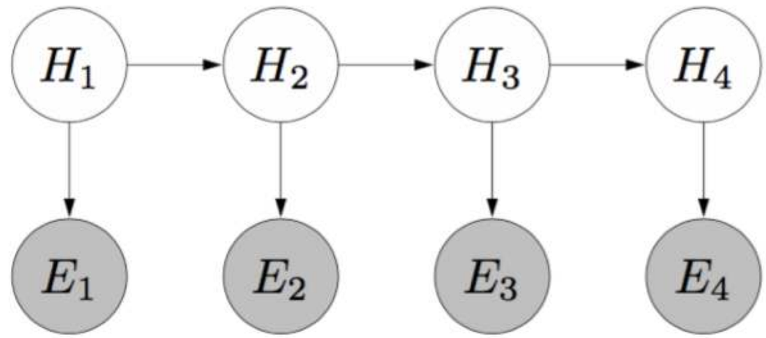
  Generate activity of disease $D_i \sim p(D_i)$

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

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

[1]Reference: Percy Liang, CS221 (2015)

Object Tracking via Hidden Markov Model



Problem: object tracking

$H_i \in \{1, \ldots, K\}$: location of object at time step i

$E_i \in \{1, \ldots, K\}$: sensor reading at time step i

Start: $p(h_1)$: uniform over all locations

Transition $p(h_i \mid h_{i-1})$: uniform over adjacent loc.

Emission $p(e_i \mid h_i)$: uniform over adjacent loc.

[1]Reference: Percy Liang, CS221 (2015)

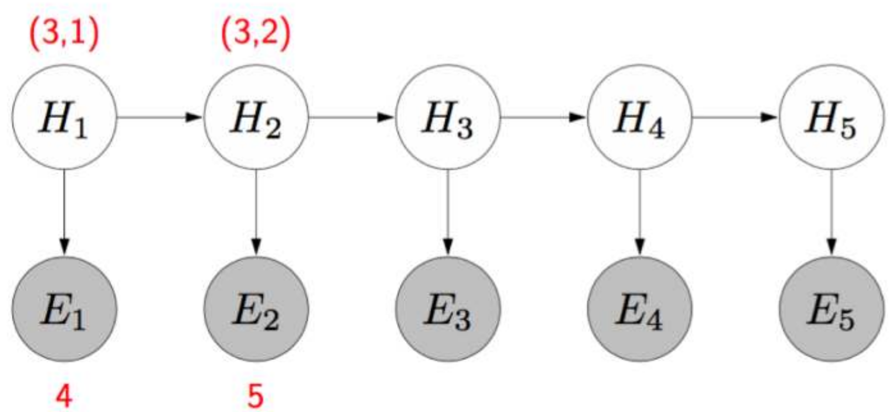
Generative Program for HMM

Probabilistic program: hidden Markov model (HMM)

For each time step $t = 1, \ldots, T$:

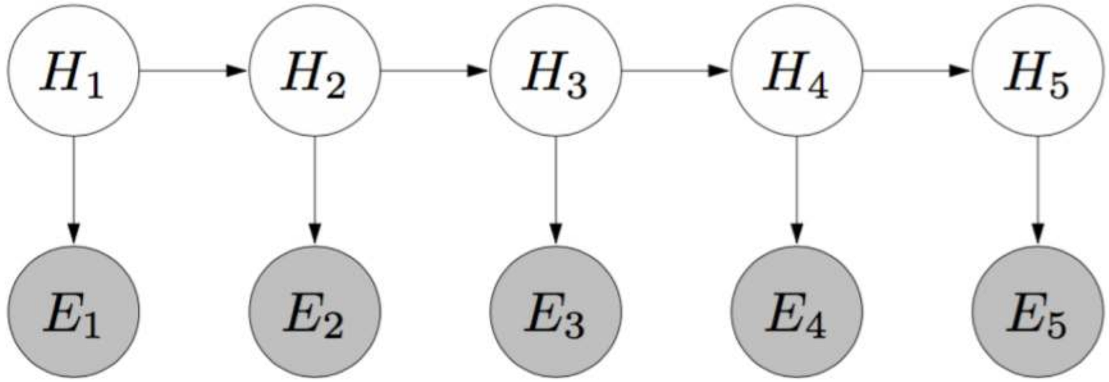
Generate object location $H_t \sim p(H_t \mid H_{t-1})$

Generate sensor reading $E_t \sim p(E_t \mid H_t)$



[1]Reference: Percy Liang, CS221 (2015)

Object Tracking via HMM



$$\mathbb{P}(H = h, E = e) = \underbrace{p(h_1)}_{\text{start}} \prod_{i=2}^{n} \underbrace{p(h_i \mid h_{i-1})}_{\text{transition}} \prod_{i=1}^{n} \underbrace{p(e_i \mid h_i)}_{\text{emission}}$$

Query (**filtering**):

$$\mathbb{P}(H_3 \mid E_1 = e_1, E_2 = e_2, E_3 = e_3)$$

Query (**smoothing**):

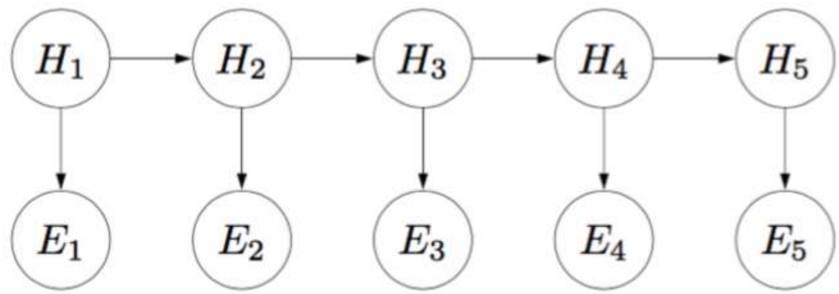
$$\mathbb{P}(H_3 \mid E_1 = e_1, E_2 = e_2, E_3 = e_3, E_4 = e_4, E_5 = e_5)$$

[1]Reference: Percy Liang, CS221 (2015)

HMM Parameter Sharing

Variables:

- $H_1, \ldots, H_n$ (e.g., actual positions)
- $E_1, \ldots, E_n$ (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)

[1]Reference: Percy Liang, CS221 (2015)

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

[1] Reference: Pedro Domingos, CSE 515 (2017)

Gaussian Mixture Models

- The N-dim. multivariate normal distribution, $\mathcal{N}(\mu, \Sigma)$, has density:

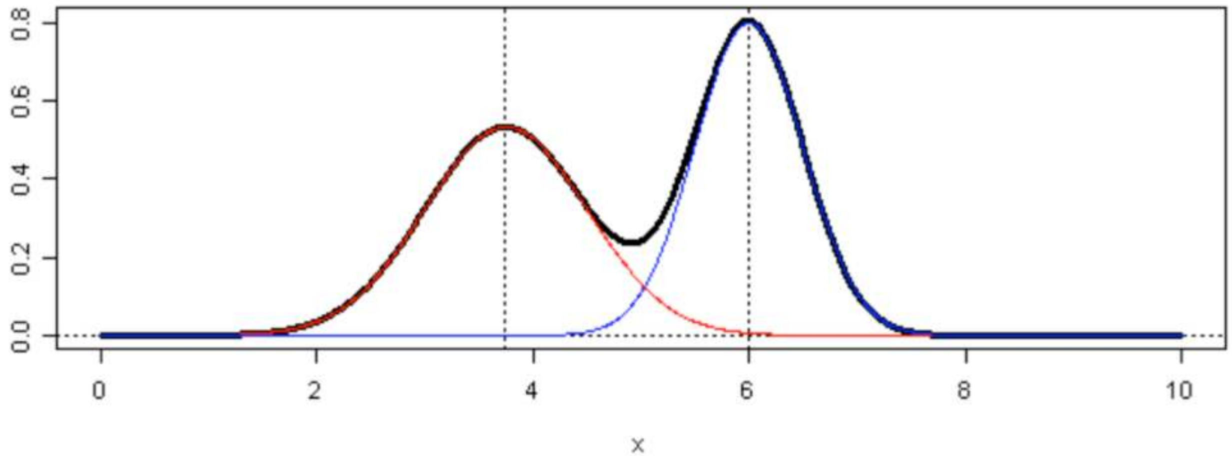
$$p(\mathbf{x}) = \frac{1}{(2\pi)^{N/2}|\Sigma|^{1/2}} \exp\Big(-\frac{1}{2}(\mathbf{x}-\mu)^T \Sigma^{-1}(\mathbf{x}-\mu)\Big)$$

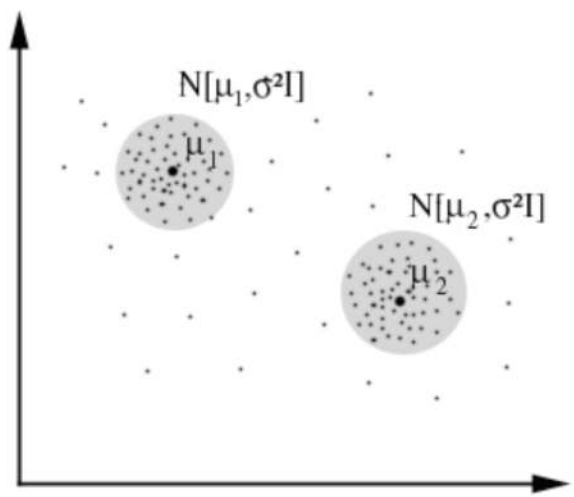
- Suppose we have k Gaussians given by μ_k and Σ_k, and a distribution θ over the numbers $1, \ldots, k$
- Mixture of Gaussians distribution $p(y, \mathbf{x})$ given by
  1. Sample $y \sim \theta$ (specifies which Gaussian to use)
  2. Sample $x \sim \mathcal{N}(\mu_y, \Sigma_y)$

[1]Reference: David Sontag (2013)

Gaussian Mixture Model in 1D and 2D

- The marginal distribution over **x** looks like:





[1]Reference: David Sontag (2013)

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)

[1]Reference: Pedro Domingos, CSE 515 (2017)

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:

$$P(c_i) = \frac{1}{n_e}\sum_{k=1}^{n_e} P(\mu_i|x_k)$$

$$\mu_i = \frac{\sum_{k=1}^{n_e} x_k\, P(\mu_i|x_k)}{\sum_{k=1}^{n_e} P(\mu_i|x_k)}$$

$$\sigma_i^2 = \frac{\sum_{k=1}^{n_e} (x_k - \mu_i)^2\, P(\mu_i|x_k)}{\sum_{k=1}^{n_e} P(\mu_i|x_k)}$$

[1]Reference: Pedro Domingos, CSE 515 (2017)

Latent Dirichlet Allocation

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

Documents

Topics

politics	religion	sports
president	hindu	baseball
obama	judiasm	soccer
washington	ethics	basketball
religion	buddhism	football
...	...	...

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

New document

Words $w_1, \ldots, w_N$

What is this document about?

weather	.50
finance	.49
sports	.01

Distribution of topics θ

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

[1]Reference: David Sontag (2013)

Latent Dirichlet Allocation

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

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

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)

[1]Reference: David Sontag (2013)

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)

Documents

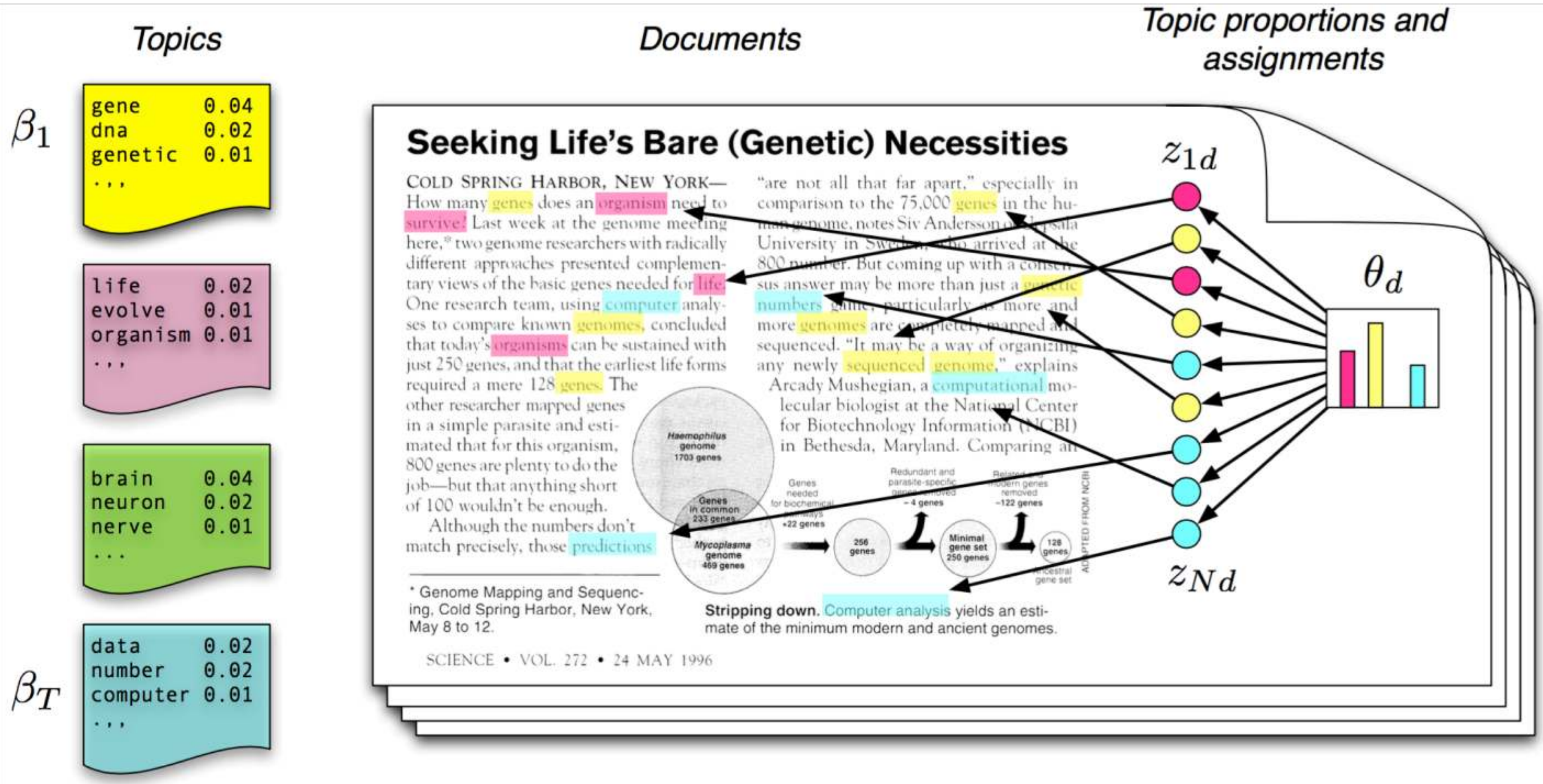
Topics

politics .0100	religion .0500	sports .0105
president .0095	hindu .0092	baseball .0100
obama .0090	judiasm .0080	soccer .0055
washington .0085	ethics .0075	basketball .0050
religion .0060	buddhism .0016	football .0045
...	...	...

$$\beta_t = \{ p(w \mid z = t) \}$$

[1]Reference: David Sontag (2013)

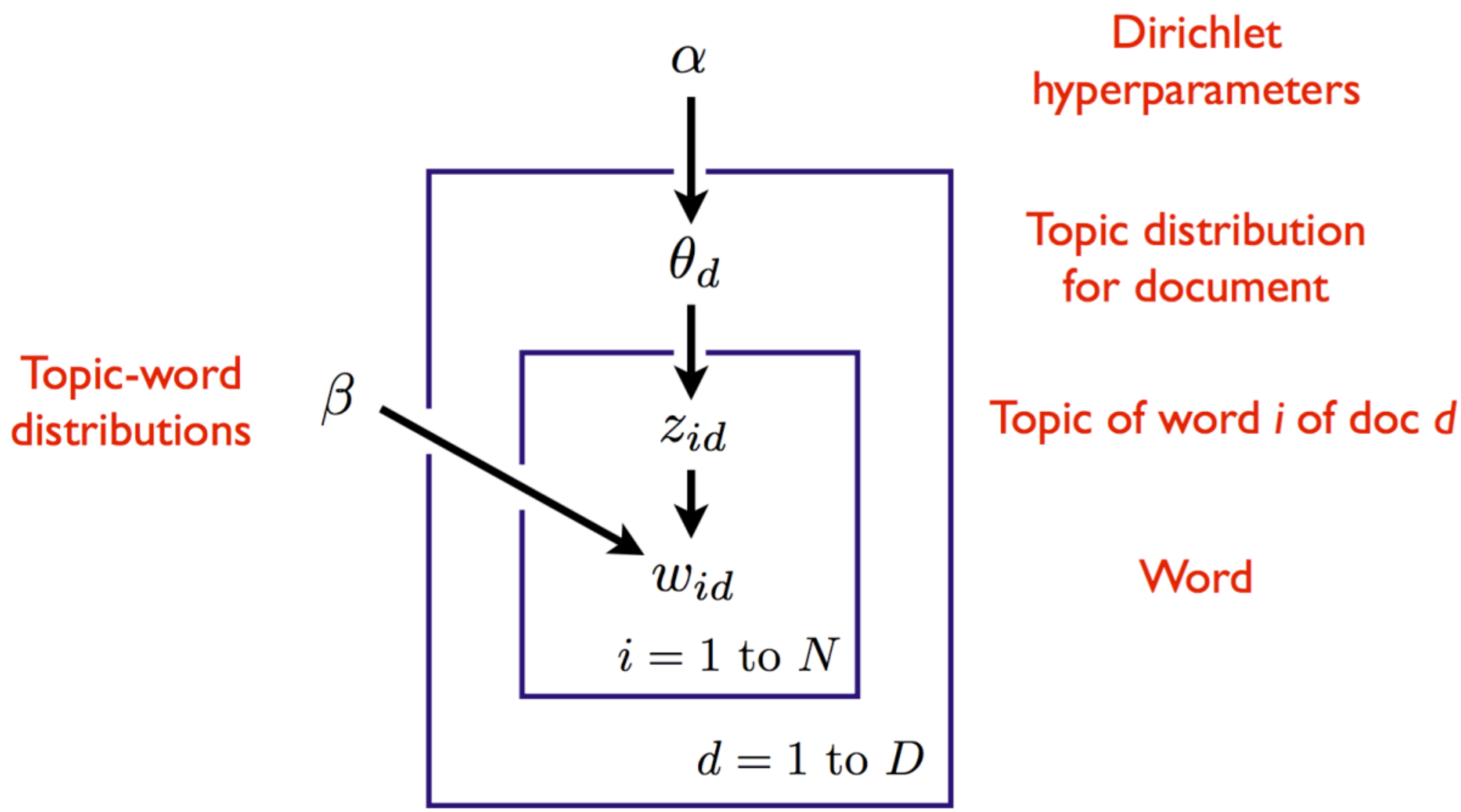
Latent Dirichlet Allocation



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

[1]Reference: David Sontag (2013)

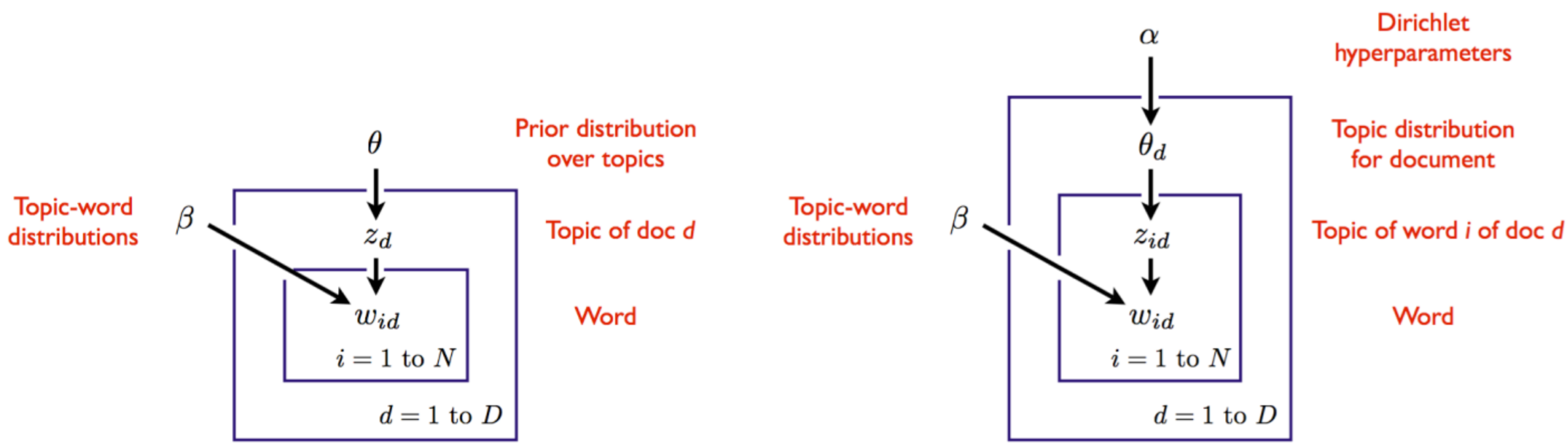
Latent Dirichlet Allocation



Variables within a plate are replicated in a conditionally independent manner

[1]Reference: David Sontag (2013)

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

[1]Reference: David Sontag (2013)

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 $\mathbf{Y}$

[1]Reference: David Sontag (2013)

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 $\mathbf{Y}$

[1]Reference: David Sontag (2013)

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

[1]Reference: David Sontag (2013)

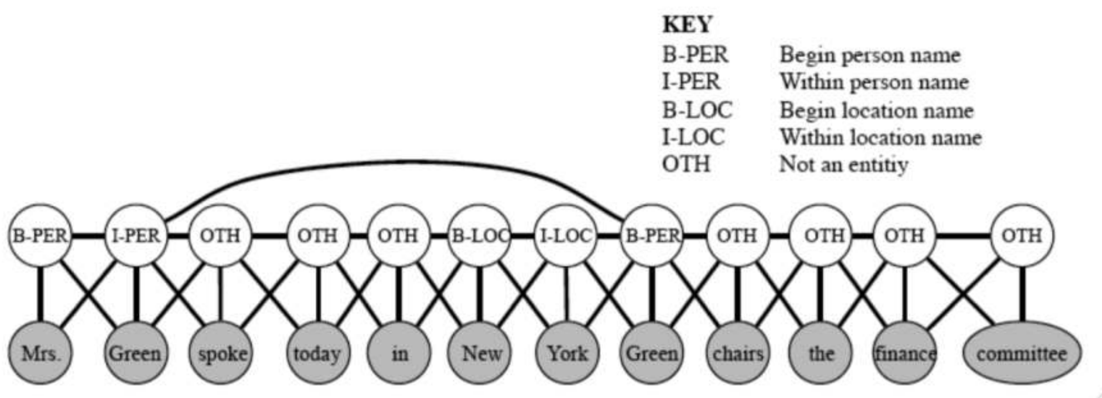
CRF for Natural Language Processing: The Task

- 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

[1]Reference: David Sontag (2013)

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
- $\phi^3(Y_t, X_1, \cdots, X_T)$ 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!

[1]Reference: David Sontag (2013)

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

[1]Reference: Pedro Domingos, CSE 515 (2017)

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

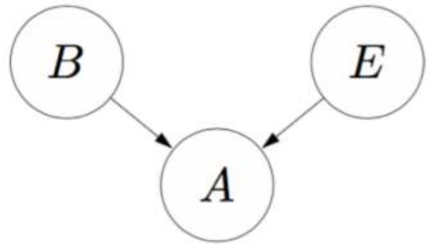
Different Estimation/Learning Problems

- There are many variants

Model	DPGM	UPGM
Data	Complete	Incomplete
Structure	Known	Unknown
Objective	Generative	Discriminative

[1]Reference: Pedro Domingos, CSE 515 (2017)

Learning in DPGM: Parameters



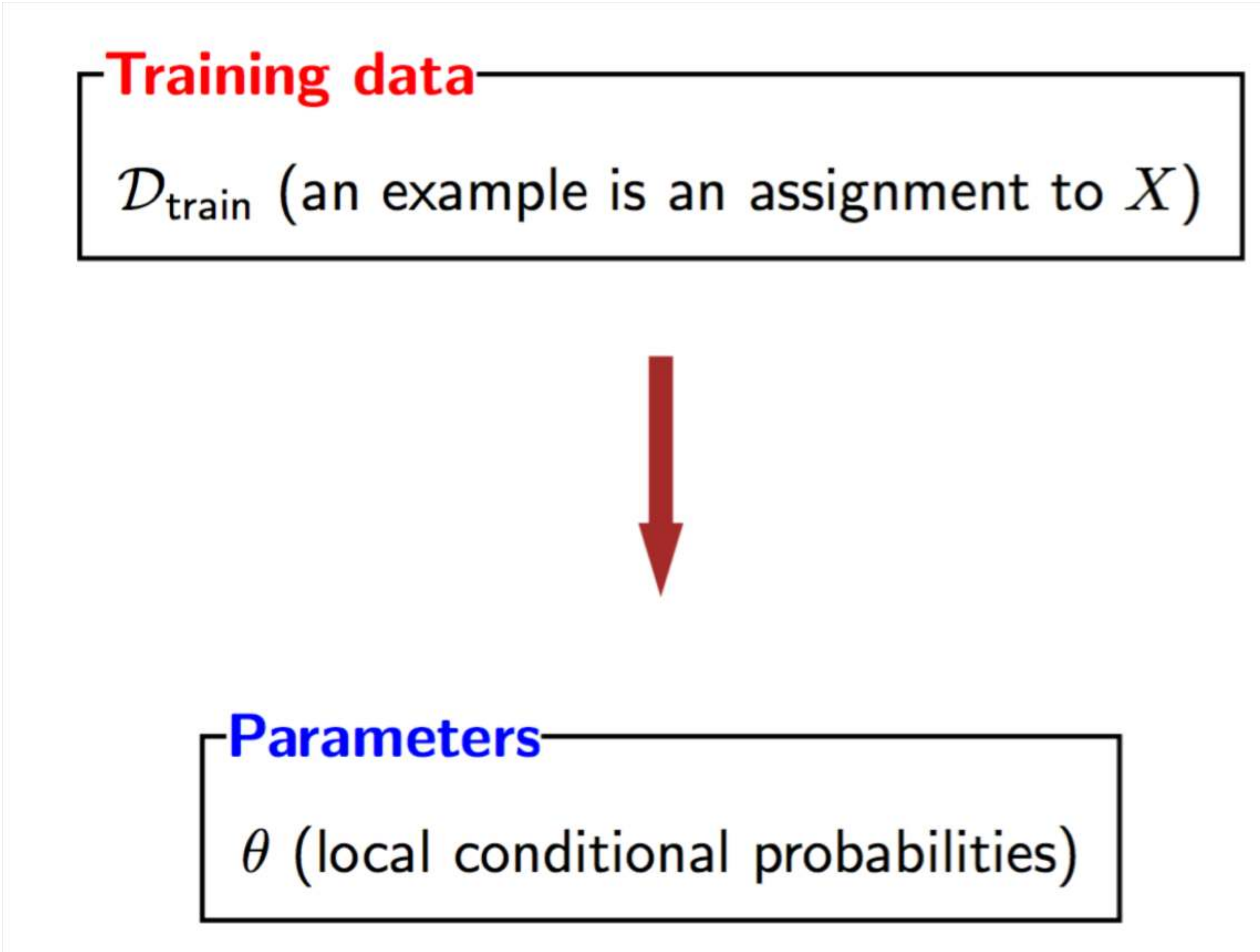
b	$p(b)$
1	?
0	?

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	?

[1]Reference: Percy Liang, CS221 (2015)

Learning in DPGM: Parameter Estimation



[1]Reference: Percy Liang, CS221 (2015)

Learning in DPGM: One Variable Example

Setup:

- One variable R representing the rating of a movie $\{1, 2, 3, 4, 5\}$

$$R \qquad \mathbb{P}(R = r) = p(r)$$

Parameters:

$$\theta = (p(1), p(2), p(3), p(4), p(5))$$

Training data:

$$\mathcal{D}_{\text{train}} = \{1, 3, 4, 4, 4, 4, 4, 5, 5, 5\}$$

[1] Reference: Percy Liang, CS221 (2015)

Learning in DPGM: One Variable Example

Learning:

$$\mathcal{D}_{\text{train}} \quad \Rightarrow \quad \theta$$

Intuition: $p(r) \propto$ number of occurrences of r in $\mathcal{D}_{\text{train}}$

Example:

$$\mathcal{D}_{\text{train}} = \{1, 3, 4, 4, 4, 4, 4, 5, 5, 5\}$$

$$\downarrow$$

θ:

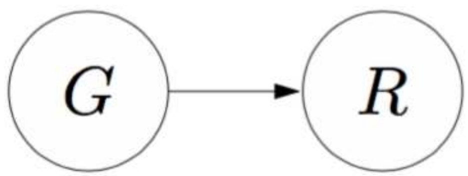
r	$p(r)$
1	0.1
2	0.0
3	0.1
4	0.5
5	0.3

[1]Reference: Percy Liang, CS221 (2015)

Learning in DPGM: Two Variables Example

Variables:

- Genre $G \in \{\text{drama}, \text{comedy}\}$
- Rating $R \in \{1, 2, 3, 4, 5\}$



$$\mathbb{P}(G = g, R = r) = p_G(g) p_R(r \mid g)$$

$$\mathcal{D}_{\text{train}} = \{(\text{d}, 4), (\text{d}, 4), (\text{d}, 5), (\text{c}, 1), (\text{c}, 5)\}$$

Parameters: $\theta = (p_G, p_R)$

[1]Reference: Percy Liang, CS221 (2015)

Learning in DPGM: Two Variables Example

$$\mathcal{D}_{\text{train}} = \{(\text{d}, 4), (\text{d}, 4), (\text{d}, 5), (\text{c}, 1), (\text{c}, 5)\}$$

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	$p_G(g)$
d	3/5
c	2/5

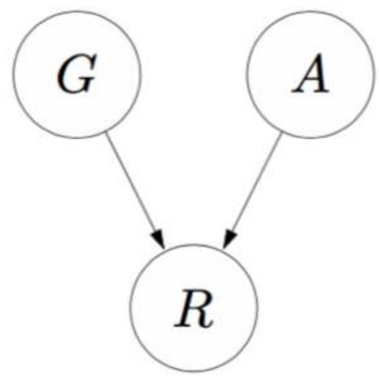
g	r	$p_R(r \mid g)$
d	4	2/3
d	5	1/3
c	1	1/2
c	5	1/2

[1]Reference: Percy Liang, CS221 (2015)

Learning in DPGM: Three Variables Example I

Variables:

- Genre $G \in \{\text{drama}, \text{comedy}\}$
- Won award $A \in \{0, 1\}$
- Rating $R \in \{1, 2, 3, 4, 5\}$



$$\mathbb{P}(G = g, A = a, R = r) = p_G(g)p_A(a)p_R(r \mid g, a)$$

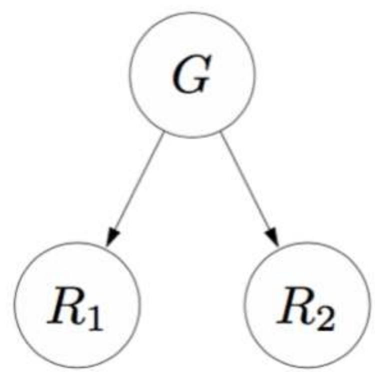
$$\mathcal{D}_{\text{train}} = \{(\text{d}, 0, 3), (\text{d}, 1, 5), (\text{c}, 0, 1), (\text{c}, 0, 5), (\text{c}, 1, 4)\}$$

[1]Reference: Percy Liang, CS221 (2015)

Learning in DPGM: Three Variables Example II

Variables:

- Genre $G \in \{\text{drama}, \text{comedy}\}$
- Jim's rating $R_1 \in \{1, 2, 3, 4, 5\}$
- Martha's rating $R_2 \in \{1, 2, 3, 4, 5\}$



$$\mathbb{P}(G = g, R_1 = r_1, R_2 = r_2) = p_G(g) p_{R_1}(r_1 \mid g) p_{R_2}(r_2 \mid g)$$

$$\mathcal{D}_{\text{train}} = \{(\text{d}, 4, 5), (\text{d}, 4, 4), (\text{d}, 5, 3), (\text{c}, 1, 2), (\text{c}, 5, 4)\}$$

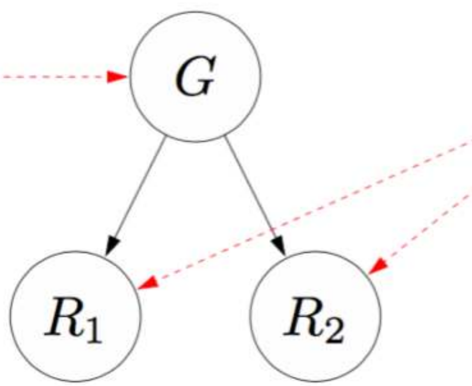
[1]Reference: Percy Liang, CS221 (2015)

Learning in DPGM: Parameter Sharing

Key idea: parameter sharing

The local conditional donstributions of different variables use the same parameters.

g	$p_G(g)$
c	2/5
d	3/5



g	r	$p_R(r \mid g)$
c	1	1/4
c	2	1/4
c	3	0/4
c	4	1/4
c	5	1/4
d	1	0/6
d	2	0/6
d	3	1/6
d	4	3/6
d	5	2/6

[1]Reference: Percy Liang, CS221 (2015)

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 $\text{count}_{d_i}(x_{\text{Parents}(i)}, x_i)$

Normalize:

For each d and local assignment $x_{\text{Parents}(i)}$:

Set $p_d(x_i \mid x_{\text{Parents}(i)}) \propto \text{count}_d(x_{\text{Parents}(i)}, x_i)$

[1]Reference: Percy Liang, CS221 (2015)

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).

[1] Reference: Percy Liang, CS221 (2015)

Learning in DPGM: Maximum Likelihood via Counting and Normalizing

$\mathcal{D}_{\text{train}} = \{(\mathsf{d}, 4), (\mathsf{d}, 5), (\mathsf{c}, 5)\}$

$$\max_{p_G(\cdot)}(p_G(\mathsf{d})p_G(\mathsf{d})p_G(\mathsf{c})) \max_{p_R(\cdot|\mathsf{c})} p_R(5 \mid \mathsf{c}) \max_{p_R(\cdot|\mathsf{d})} (p_R(4 \mid \mathsf{d})p_R(5 \mid \mathsf{d}))$$

- 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)

[1] Reference: Percy Liang, CS221 (2015)

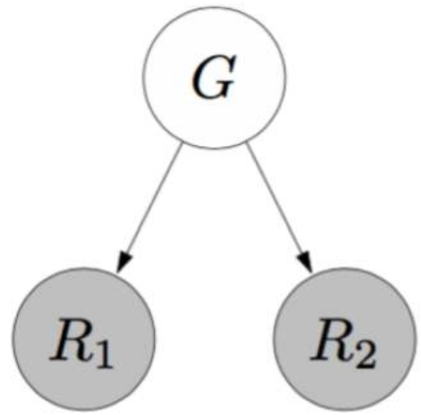
Different Estimation/Learning Problems

- What if we have missing data?

Model	DPGM	UPGM
Data	Complete	Incomplete
Structure	Known	Unknown
Objective	Generative	Discriminative

[1]Reference: Pedro Domingos, CSE 515 (2017)

Learning in DPGM: Latent Variables



What if we **don't observe** some of the variables?

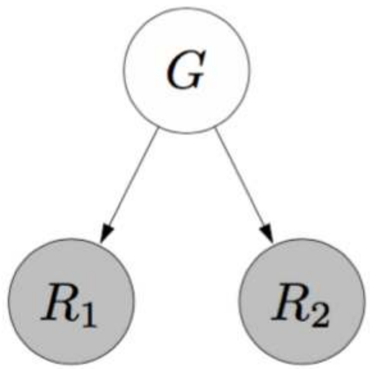
$$\mathcal{D}_{\text{train}} = \{(?, 4, 5), (?, 4, 4), (?, 5, 3), (?, 1, 2), (?, 5, 4)\}$$

[1]Reference: Percy Liang, CS221 (2015)

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:

$$\max_{\theta} \prod_{e \in \mathcal{D}_{\text{train}}} \mathbb{P}(E = e; \theta)$$

$$= \max_{\theta} \prod_{e \in \mathcal{D}_{\text{train}}} \sum_{h} \mathbb{P}(H = h, E = e; \theta)$$

[1]Reference: Percy Liang, CS221 (2015)

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.

[1]Reference: Percy Liang, CS221 (2015)

[2]Note: EM was first proposed in 1977

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

[1]Reference: Percy Liang, CS221 (2015)

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

[1]Reference: Percy Liang, CS221 (2015)

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 $q(h)$

[1]Reference: Percy Liang, CS221 (2015)

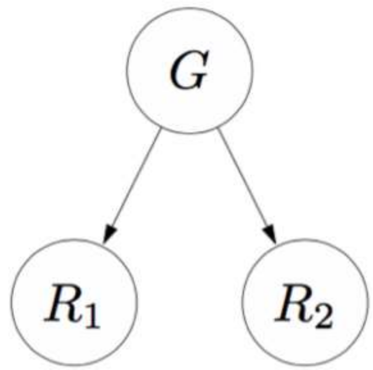
The Two Steps of EM

- 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

[1]Reference: Percy Liang, CS221 (2015)

EM: Example



$\mathcal{D}_{\text{train}} = \{(?, 2, 2), (?, 1, 2)\}$

θ:

g	$p_G(g)$
c	0.5
d	0.5

g	r	$p_R(r \mid g)$
c	1	0.4
c	2	0.6
d	1	0.6
d	2	0.4

E-step

(r_1, r_2)	g	$\mathbb{P}(G = g, R_1 = r_1, R_2 = r_2)$	$q(g)$
(2, 2)	c	$0.5 \cdot 0.6 \cdot 0.6$	$\frac{0.18}{0.18+0.08} = 0.69$
(2, 2)	d	$0.5 \cdot 0.4 \cdot 0.4$	$\frac{0.08}{0.18+0.08} = 0.31$
(1, 2)	c	$0.5 \cdot 0.4 \cdot 0.6$	0.5
(1, 2)	d	$0.5 \cdot 0.6 \cdot 0.4$	0.5

M-step

θ:

g	count	$p_G(g)$
c	$0.69 + 0.5$	0.59
d	$0.31 + 0.5$	0.41

g	r	count	$p_R(r \mid g)$
c	1	0.5	0.21
c	2	$0.5 + 0.69 + 0.69$	0.79
d	1	0.5	0.31
d	2	$0.5 + 0.31 + 0.31$	0.69

[1] Reference: Percy Liang, CS221 (2015)

Different Estimation/Learning Problems

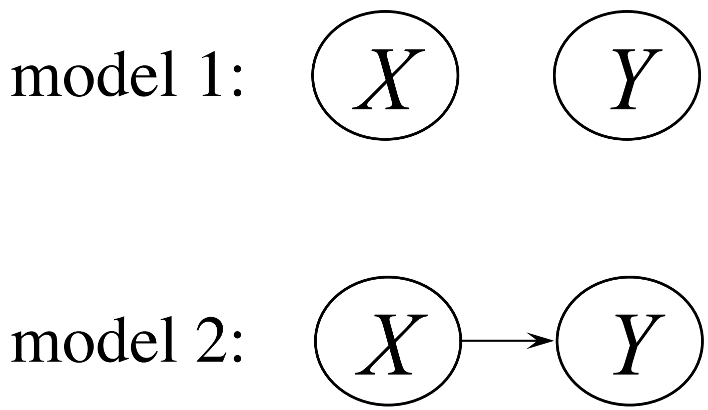
- What if the structure is unknown?

Model	DPGM	UPGM
Data	Complete	Incomplete
Structure	Known	Unknown
Objective	Generative	Discriminative

[1]Reference: Pedro Domingos, CSE 515 (2017)

Learning Structure: Bayesian Approach

- Given data, which model is correct?



[1]Reference: Pedro Domingos, CSE 515 (2017)

Learning Structure: Bayesian Approach

- Given data, which model is ~~correct?~~ more likely?

model 1: $X \quad Y$ $\qquad p(m_1) = 0.7$ $\qquad p(m_1 \mid \mathbf{d}) = 0.1$

$$\xrightarrow{\text{Data } \mathbf{d}}$$

model 2: $X \rightarrow Y$ $\qquad p(m_2) = 0.3$ $\qquad p(m_2 \mid \mathbf{d}) = 0.9$

- Can do model averaging
- Can do model selection to pick a model that is
  - tractable, understandable, explainable

[1]Reference: Pedro Domingos, CSE 515 (2017)

Learning Structure: Model Scoring

- Use Baye's theorem to score a model

Given data $\mathbf{d}$:

model score

$$p(m \mid \mathbf{d}) \propto p(m)\underline{p(\mathbf{d} \mid m)}$$

"marginal likelihood"

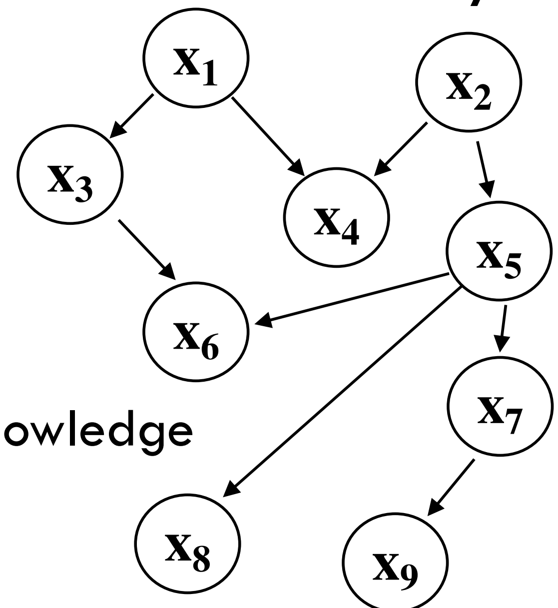
likelihood

$$p(\mathbf{d} \mid m) = \int p(\mathbf{d} \mid \theta, m)\, p(\theta \mid m)\, d\theta$$

[1]Reference: Pedro Domingos, CSE 515 (2017)

Combined Learning

- Although structure learning is hard in general, still useful to do it by using prior knowledge and data

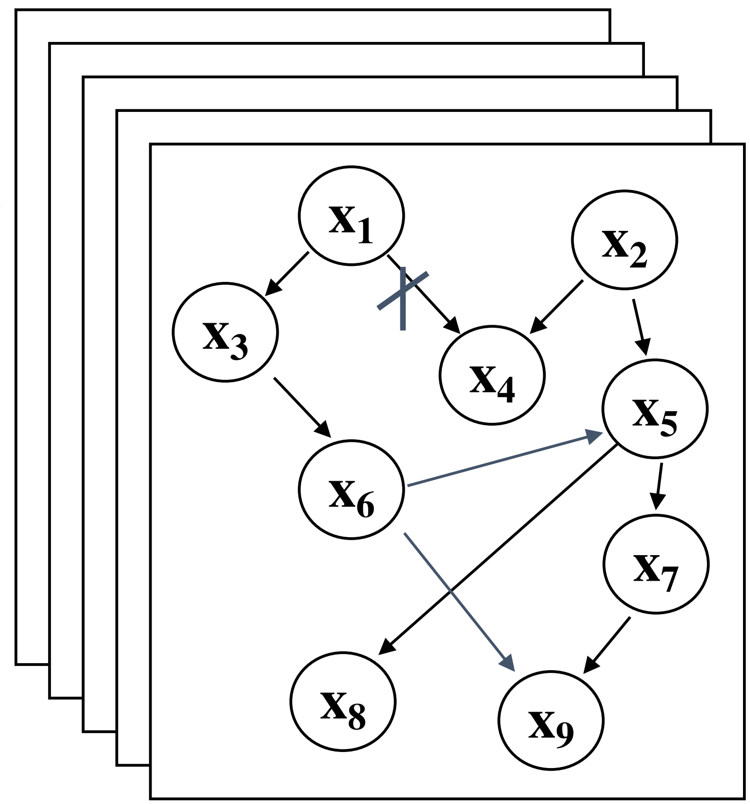


Prior knowledge

data

x_1	x_2	x_3	
true	false	true	
false	false	true	
false	false	false	...
true	true	false	
	⋮		⋱

Learned network(s)



[1]Reference: Pedro Domingos, CSE 515 (2017)

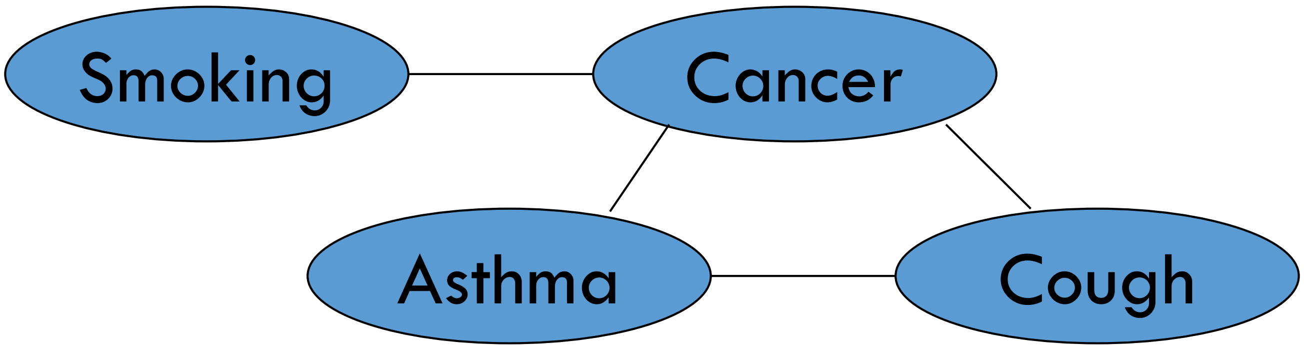
Different Estimation/Learning Problems

- There are many variants

Model	DPGM	UPGM
Data	Complete	Incomplete
Structure	Known	Unknown
Objective	Generative	Discriminative

[1]Reference: Pedro Domingos, CSE 515 (2017)

Learning in UPGM



Potential functions defined over cliques

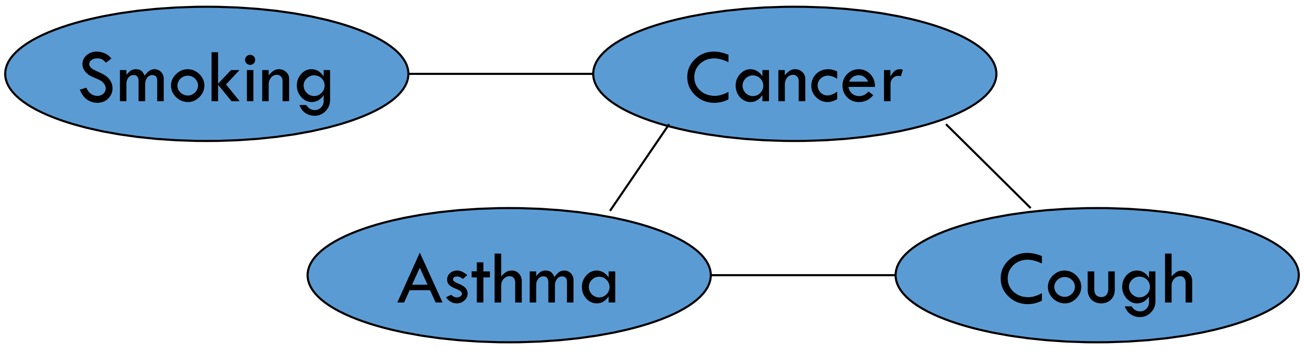
$$P(x) = \frac{1}{Z} \prod_c \Phi_c(x_c)$$

$$Z = \sum_x \prod_c \Phi_c(x_c)$$

Smoking	**Cancer**	**Φ(S,C)**
False	False	4.5
False	True	4.5
True	False	2.7
True	True	4.5

[1]Reference: Pedro Domingos, CSE 515 (2017)

Learning in UPGM



Can be thought in terms of a log-linear representation

$$P(x) = \frac{1}{Z}\exp\left(\sum_i \underbrace{\boxed{w_i}}_{\text{Weight of Feature } i} . \underbrace{\boxed{f_i(x)}}_{\text{Feature } i}\right)$$

$$f_1(\text{Smoking, Cancer}) = \begin{cases} 1 & \text{if } \neg \text{ Smoking } \vee \text{ Cancer} \\ 0 & \text{otherwise} \end{cases}$$

$$w_1 = 0.51$$

[1]Reference: Pedro Domingos, CSE 515 (2017)

Learning in UPGM: Generative

- Maximize likelihood or posterior probability
- Numerical optimization (gradient or 2nd order)

$$\frac{\partial}{\partial w_i} \log P_w(x) = \boxed{n_i(x)} - \boxed{E_w\left[n_i(x)\right]}$$

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!)

[1]Reference: Pedro Domingos, CSE 515 (2017)

Learning in UPGM: Pseudo-likelihood

$$PL(x) \equiv \prod_i P(x_i \mid neighbors(x_i))$$

- 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

[1]Reference: Pedro Domingos, CSE 515 (2017)

Different Estimation/Learning Problems

- There are many variants

Model	DPGM	UPGM
Data	Complete	Incomplete
Structure	Known	Unknown
Objective	Generative	Discriminative

[1]Reference: Pedro Domingos, CSE 515 (2017)

Learning in UPGM: Discriminative

- This is related to Conditional Random Fields (CRFs)
- Maximize conditional likelihood of query (y) given evidence (x)

$$\frac{\partial}{\partial w_i} \log P_w(y \mid x) = \boxed{n_i(x,y)} - \boxed{E_w\left[n_i(x,y)\right]}$$

No. of true values of feature *i* in data

Expected no. of true values according to model

- Inference is easier, but still hard

[1]Reference: Pedro Domingos, CSE 515 (2017)

Different Estimation/Learning Problems

- There are many variants

Model	DPGM	UPGM
Data	Complete	Incomplete
Structure	Known	Unknown
Objective	Generative	Discriminative

[1]Reference: Pedro Domingos, CSE 515 (2017)

Learning in UPGM: Missing Data

- Gradient of likelihood is now difference of expectations

$$\frac{\partial}{\partial w_i} \log P_w(x) = \boxed{E_w\left[n_i(y \mid x)\right]} - \boxed{E_w\left[n_i(x, y)\right]}$$

Exp. no. true values given observed data

Expected no. true values given no data

x: Observed
y: Missing

- Can use gradient descent or EM

[1]Reference: Pedro Domingos, CSE 515 (2017)

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.

[1] Reference: Pedro Domingos, CSE 515 (2017)

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

[1]Reference: Percy Liang, CS221 (2015)

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 $\mathbf{P}(Z_i|mb(Z_i))$
        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

[1]Reference: Pedro Domingos, CSE 515 (2017)

Additional Applications: Naïve Bayes Spam Filter

- Key assumption
  Words occur independently of each other given the label of the document

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

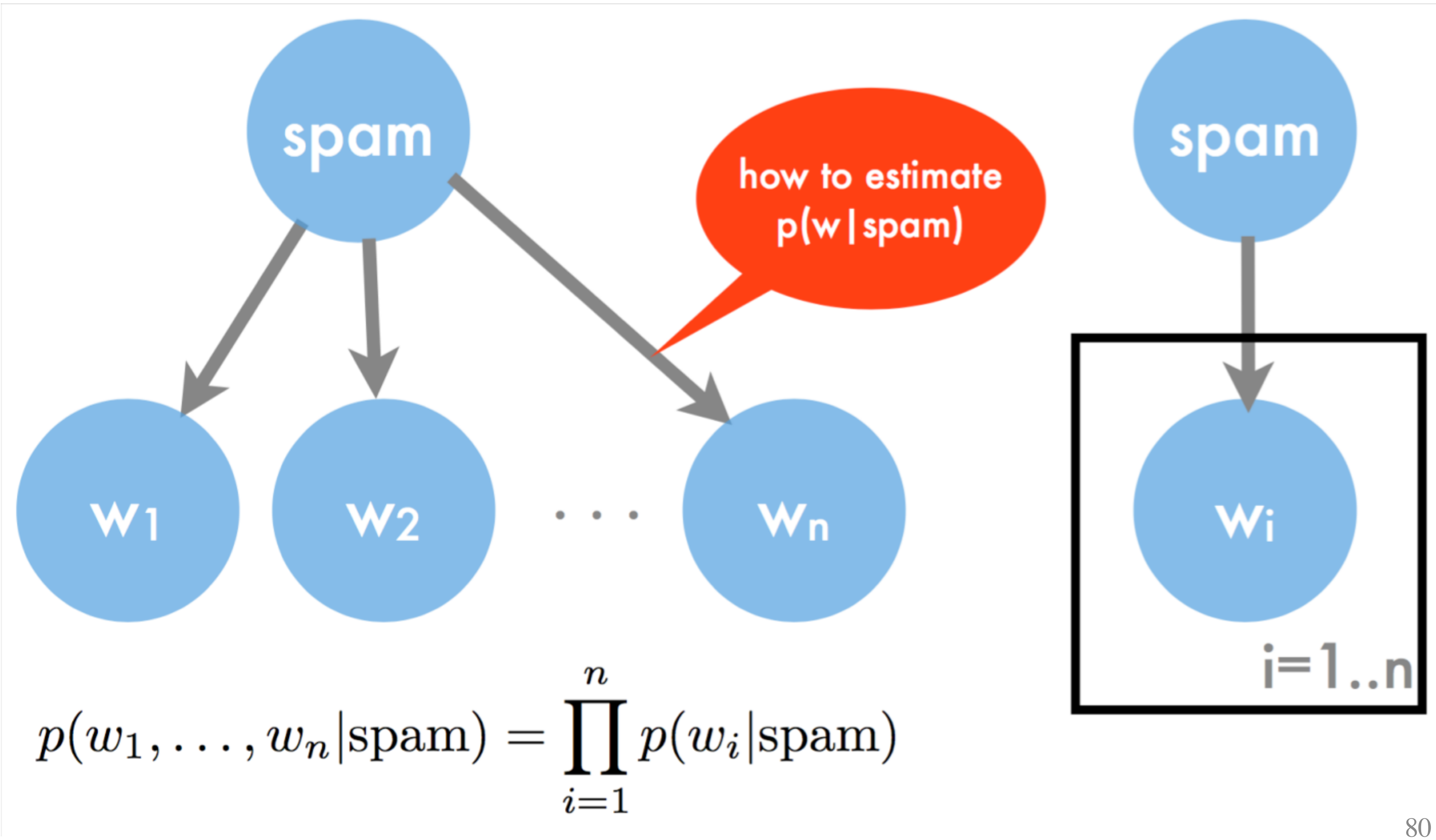
- Spam classification via Bayes Rule

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

- Parameter estimation
  Compute spam probability and word distributions for spam and ham

[1] Reference: Alex Smola (2011)

Additional Applications: Naïve Bayes Spam Filter



[1]Reference: Alex Smola (2011)

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

spam probability

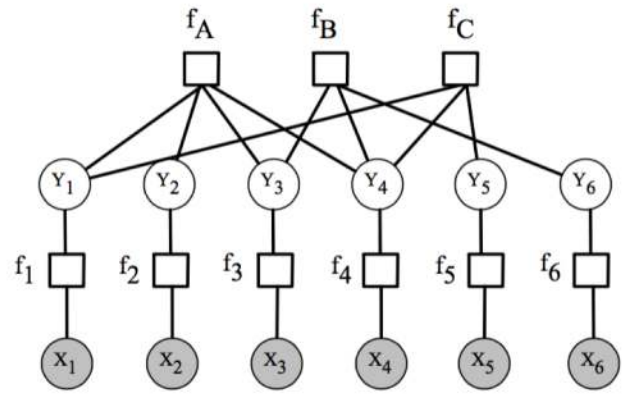
$$p(x_i = \mathrm{TRUE}|y) = \frac{n(i,y)}{n(y)} \text{ and } p(y) = \frac{n(y)}{n}$$

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

[1]Reference: Alex Smola (2011)

Additional Applications: MAP Problem in Low Density Parity Check Codes

- Error correcting codes for transmitting a message over a noisy channel (invented by Galleger in the 1960's, then re-discovered in 1996)



- Each of the top row factors enforce that its variables have even parity:

$$f_A(Y_1, Y_2, Y_3, Y_4) = 1 \text{ if } Y_1 \otimes Y_2 \otimes Y_3 \otimes Y_4 = 0, \text{ and } 0 \text{ otherwise}$$

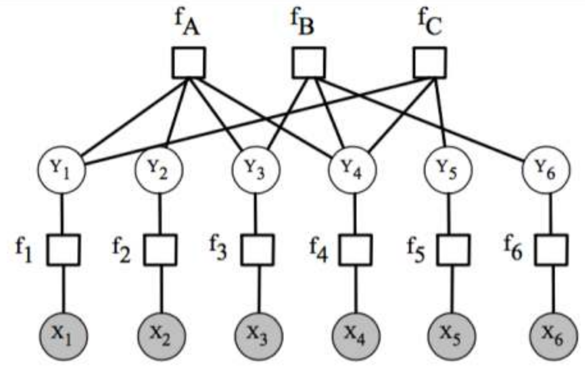
- Thus, the only assignments **Y** with non-zero probability are the following (called **codewords**): *3 bits encoded using 6 bits*

$$000000,\ 011001,\ 110010,\ 101011,\ 111100,\ 100101,\ 001110,\ 010111$$

- $f_i(Y_i, X_i) = p(X_i \mid Y_i)$, the likelihood of a bit flip according to noise model

[1]Reference: David Sontag (2013)

Additional Applications: MAP Problem in Low Density Parity Check Codes



- The *decoding* problem for LDPCs is to find

$$\mathrm{argmax}_{\mathbf{y}} p(\mathbf{y} \mid \mathbf{x})$$

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 $p(\mathbf{y}, \mathbf{x})$):

$$\mathrm{argmax}_{\mathbf{y}} \sum_{c \in C} \theta_c(\mathbf{x}_c),$$

where $\theta_c(\mathbf{x}_c) = \log \phi_c(\mathbf{x}_c)$

[1]Reference: David Sontag (2013)