Lecture 10

IDS575: Statistical Models and Methods
Theja Tulabandhula

Notes derived from the book titled "Elements of Statistical Learning [2nd Ed.] (Chapter 12 and Section 14.2)

We will finish Support Vector Machines and discuss unsupervised learning methods.

1 Support Vector Machines

Continued.

1.1 Computing the Dual Formulation

Since the optimization problem is a convex program, we can write another equivalent optimization problem with a different set of variables, called the *dual*.

The motivation for doing so is that in this dual formulation, data appears in a nice way (inner products) which lets us relax linearity. Giving us the chance to use a trick called the *kernel trick*.

Lets first write the Lagrangian:

$$L_P = \frac{1}{2}\|\beta\|_2^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i(y_i(x_i^T\beta+\beta_0) - 1 + \xi_i) - \sum_{i=1}^{N}\mu_i\xi_i,$$

where C, α_i, μ_i are the non-negative Lagrange multipliers. Taking the derivative with respect to β, β_0 and ξ_i we get:

$$\beta = \sum_i \alpha_i y_i x_i, \text{ (this is a key equation)}$$

$$0 = \sum_{i=1}^{N}\alpha_i y_i$$

$$\alpha_i = C - \mu_i.$$

We can now replace the occurrence of β, β_0 and ξ_i in L_P to get the dual problem, which will just be a function of α_i (μ_i is exactly $C - \alpha_i$). The problem is to maximize

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j,$$

subject to $0 \leq \alpha_i \leq C$ and $0 = \sum_{i=1}^{N} \alpha_i y_i$. These α_i are called the dual variables. At its maximum, $L_D = L_P$ (this is called strong duality in the optimization literature).

It turns out[1] that for the optimal $\{\alpha_i\}_{i=1}^N$ and β, β_0 and $\{\xi_i\}_{i=1}^N$, the following additional conditions hold:

$$\alpha_i(y_i(x_i^T\beta + \beta_0) - 1 + \xi_i) = 0, \text{ and}$$
$$(C - \alpha_i)\xi_i = 0.$$

Since $\beta = \sum_{i=1}^{N} \alpha_i y_i x_i$, we can try interpreting what αs mean. In particular, α_i is non-zero when $y_i(x_i^T\beta + \beta_0) - 1 + \xi_i$ is zero. The observations where this holds are known as *support vectors*[2].

Example 1. An example decision boundary and the support vector boundary is shown for two different values of C in Figure 1. As can be seen margin is larger when C is smaller. In both settings, a linear model may not be appropriate.

1.2 The Kernel Trick and the Support Vector Machine

We will now extend the support vector classifier to a more flexible setting where we call the method a *support vector machine*.

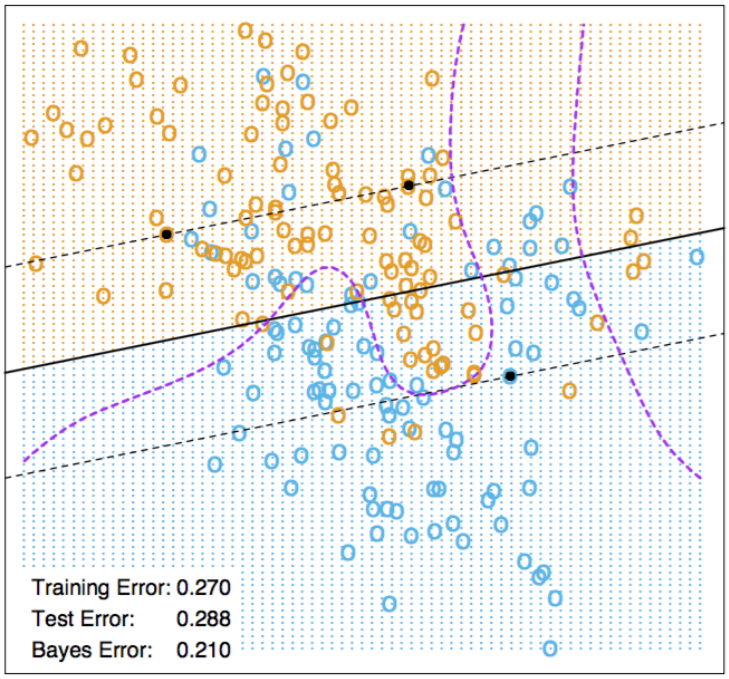
The idea here is to work with high-dimensional transformed input spaces in a succinct way using the kernel trick.

Recall that in the dual formulation, only terms of the form $x_i^T x_j$ appeared. Further, to make a prediction at a point x, we also only need inner products because:

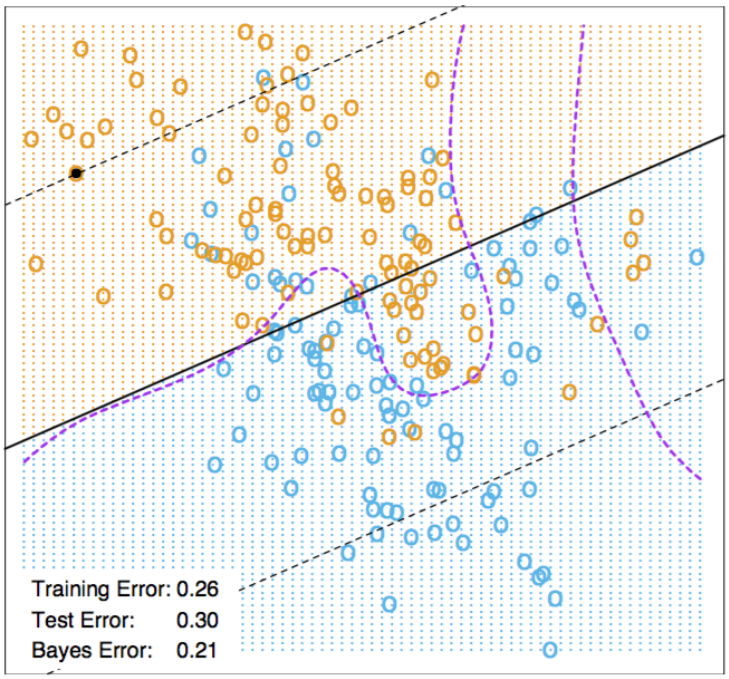
$$f(x) = \beta^T x = \sum_{i=1}^{N} \alpha_i y_i x_i^T x + \beta_0.$$

[1] These are part of what are known as the KKT conditions.

[2] Hence the name!



$C = 10000$



$C = 0.01$

Figure 1: Linear support vector classifier on a 2-dimensional dataset.

We replace these inner products with kernel functions $k(x, z)$. These functions implicitly compute inner products in arbitrary transformed spaces.

Example 2. The following are some popular choices for kernels:

- A d^{th}-degree polynomial $k(x, z) = (1 + x^T z)^d$, and
- Radial basis $k(x, z) = \exp(-\gamma \|x - z\|_2^2)$.

For the same 2-dimensional data as before, the decision boundaries and support vector boundaries are shown in Figure 2. These seem to be a better fit.

Note 1. Beyond two classes: there is no really good answer to extending SVMs to setting where the number of classes $K > 2$. One straightforward way is to build multiple two-class SVMs and use them simultaneously.

1.3 Support Vector Regression

There is a *penalization* view of SVM that alllows us to reconcile it with other methods we have seen earlier. This view is as follows:

$$\min_{\beta, \beta_0} \frac{1}{2}\|\beta\|_2^2 + C \sum_{i=1}^{N} [1 - y_i(\beta^T x_i + \beta_0)]_+,$$

where $[\cdot]_+ = max(0, \cdot)$. This formulation looks like a penalty plus a loss function. In particular, the loss function above is known as the *hinge loss*.

If we have a regression setting, the y_i is now going to be a real number. Then, we can suitably change the loss function above to get a version of support vector regression method:

$$\min_{\beta, \beta_0} \frac{1}{2}\|\beta\|_2^2 + C \sum_{i=1}^{N} L(y_i, \beta^T x_i + \beta_0),$$

where for example, $L(y, w) = 0$ if $|y - w| < \epsilon$ and $|y - w| - \epsilon$ otherwise. This is called the ϵ-insensitive loss[3].

2 Unsupervised Learning with Association Rules

As the name implies, unsupervised learning is when you don't have an explicit notion of a target or dependent variable. So what is the task? If you recall supervised learning, it was all about the conditional distribution of Y/G given X (or functions of it). Here, assuming data is denoted by X, we are interested in inferring properties of $P(X)$ (the distribution of X).

What properties are we interested in learning? Some examples are:

[3] ϵ is a parameter of the loss function here.

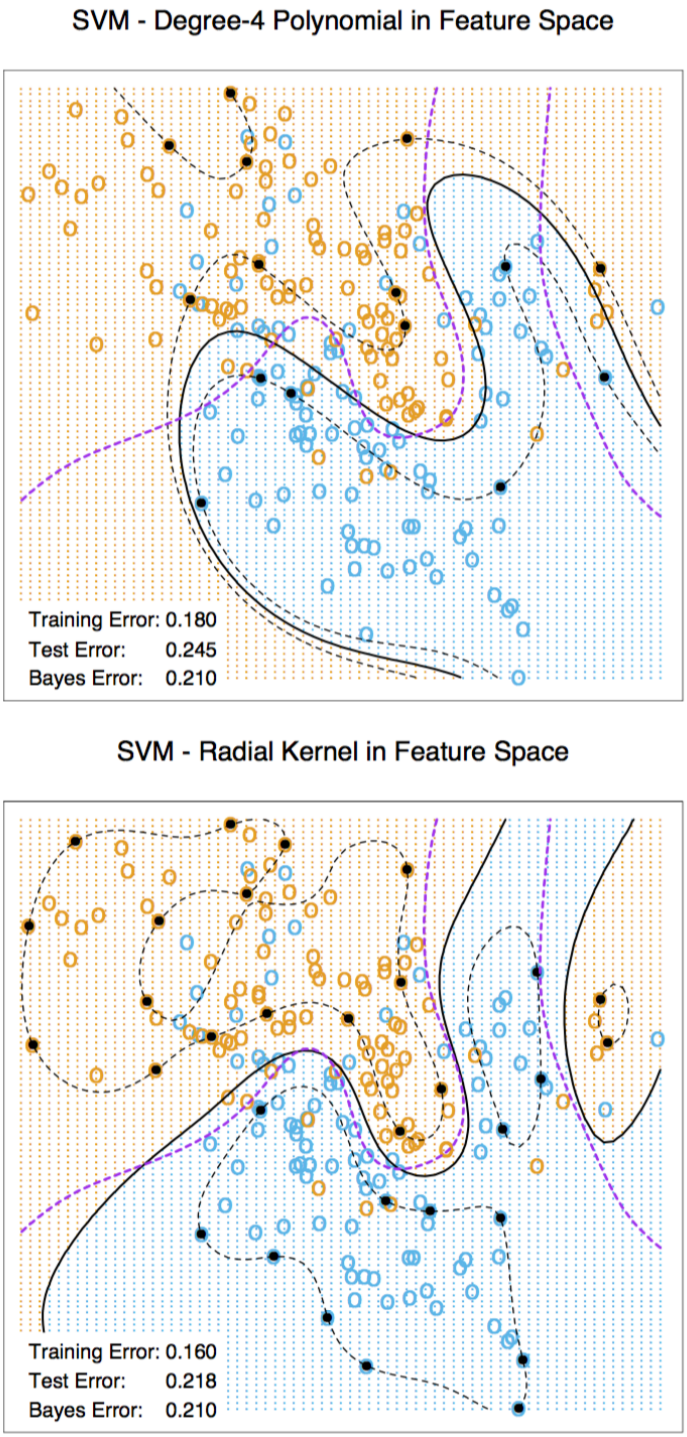


Figure 2: Support vector machine with two different kernels on a 2-dimensional dataset.

- Estimating a mixture of Gaussians model

- Descriptive statistics that characterize $P(X)$.
  - For example, methods such as principal components, multidimensional scaling, self-organizing maps etc (which we will look at later) focus on finding association between variables and reduce dimensionality.
  - Another example is cluster analysis, which finds regions in space that contain modes of $P(X)$.
  - A final example is that of association rules. These try to construct description-s/rules that describe regions of high density when the data is all binary.

There is no measure of success for unsupervised learning similar to test error notion that we studied for supervised learning. Algorithms are motivated heuristically and the results are also judged heuristically.

2.1 Association Rules

The goal of this method is to find joint values of *subsets* of $X = (X_1, ..., X_p)$ that appear most frequently in *binary* valued datasets.

Example 3. Consider sales transaction records at a retail store. Given lots of observations, you can find out which products are likely to be purchased together. Here p would be the number of all products sold in the store, which can be very large. Finding such patterns can help manage inventory, promotions, customer segmentation and a variety of business tasks.

Note 2. We care about those values in subsets of X which have high probability mass.

Note 3. This is a difficult problem when p is large, because there are exponentially many such subsets. And for each subset, there are exponential number of values X can take.

Let $s_j \subseteq \{0, 1\}$ be a subset corresponding to the j^{th} input variable. We want to find $(s_1, ..., s_p)$ such that

$$Pr[\cap_{j=1}^{p}(X_j \in s_j)]$$

is large.

Note 4. If we relax the binary assumption, the above definition and goal still stands. One can always convert data from quantitative and categorical to binary through the notion of *dummy* variables.

Example 4. if each X_j takes values in S_j, then the total number of dummy variabls is $\sum_{j=1}^{p} |S_j|$.

Coming back to the binary setting, let $\mathcal{K}$ be a subset of $\{1, ..., p\}$. Then the probability mass definition above can be rewritten as:

$$Pr[\cap_{j=1}^{p}(X_j \in s_j)] = Pr[\prod_{k \in \mathcal{K}} X_k = 1].$$

The set $\mathcal{K}$ is called an itemset and $|\mathcal{K}|$ is its size.

Its estimate $\frac{1}{N}\sum_{i=1}^{N}\prod_{k\in\mathcal{K}} x_{ik}$ is called the support of $\mathcal{K}$ and is denoted as $T(\mathcal{K})$.

We want to only identify those itemsets whose support is at least $t > 0$, and this collection can be written using set notation as: $\{\mathcal{K}_j \,|T(\mathcal{K}_j) \geq t\}$.

2.2 Apriori Algorithm

This is one of the first algorithms for finding the collection $\{\mathcal{K}_j \,|T(\mathcal{K}_j) \geq t\}$.

The key idea in the algorithm is that for a set $\mathcal{K} \leq \mathcal{K}$ then $T(\mathcal{K}) \geq T(\mathcal{K})$.

The algorithm works in rounds. In each round, it reads through the dataset once.

- In the first round, compute support of single-item sets. Discard those with $T(\mathcal{K}) < t$.
- In the second, it computes support of two-item sets that can be formed from pairs of single-items left from previous round.
- In round m, an itemset of size m is considered only if *all* its $\binom{m}{m-1}$ subsets have enough support in the previous round and the remaining single-item was left over after the first round.

Till now, we have only mined what are called *frequent itemsets*. These can be turned into *rules* by breaking the itemset into two parts: an antecedent and a consequent and checking for additional desired statistics (e.g., confidence defined below).

An association rule is of the form $p \to q$ where p and q are sets such that $p \cup q = \mathcal{K}$.

- Support of a rule is $T(p \cup q)$.
- Confidence $c(p \to q)$ of a rule is defined as $T(p \cup q)/T(p)$, which can be viewed as an estimate of $Pr(q|p)$, where for any set $\mathcal{K}$, $Pr(\mathcal{K}) = Pr(\prod_{k\in\mathcal{K}} X_k = 1)$.

Note 5. We would like to have rules that have high confidence and support. That is $\{p \to q : T(p \cup q) > t, c(p \to q) > c\}$ for some thresholds t and c.

Some of the issues with association rule mining include:

- Only for binary data.
- Computationally expensive.
- Rules with high confidence but low support will not be discovered. This may be important in some situations.

Example 5. An example rule: {'number in household = 1','number of children = 0'} → { 'language in home = English'}.

3 Summary

We learned the following things:

- The kernel trick for Support Vector Machine.
- Statistical basis for association rules.

A Sample Exam Questions

1. What are support vectors? Express the prediction model in terms of the support vectors.
2. What is the kernel trick?
3. Describe the pros and cons of association rules? What is their time complexity?