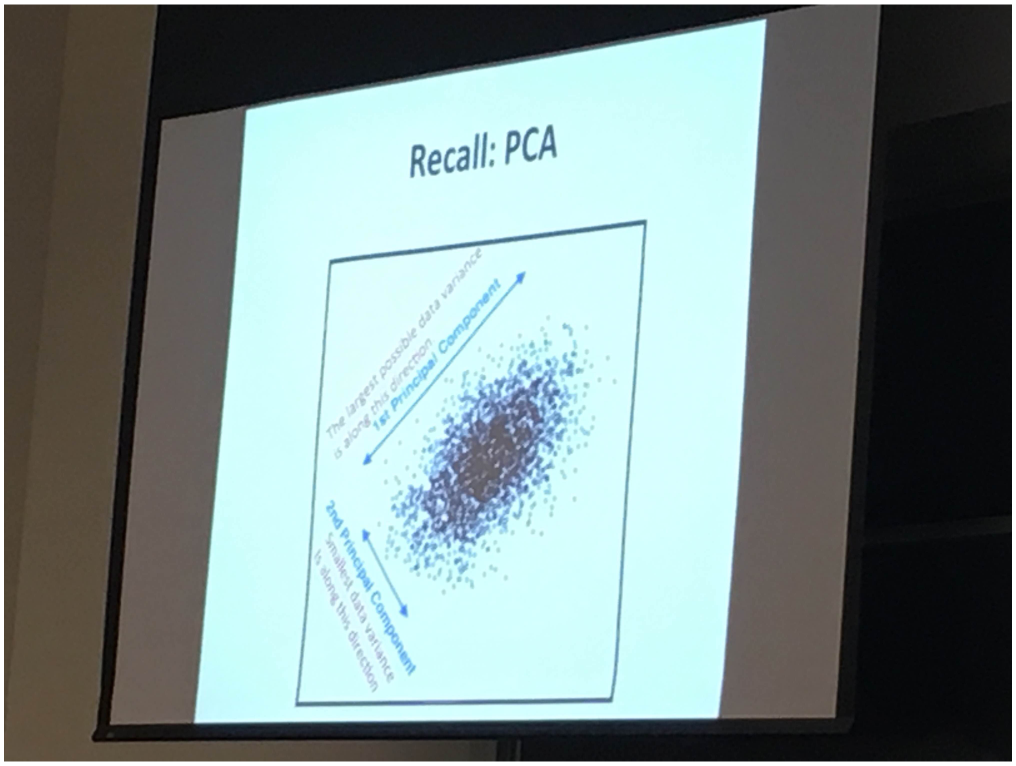
Mathematics of Big Data I

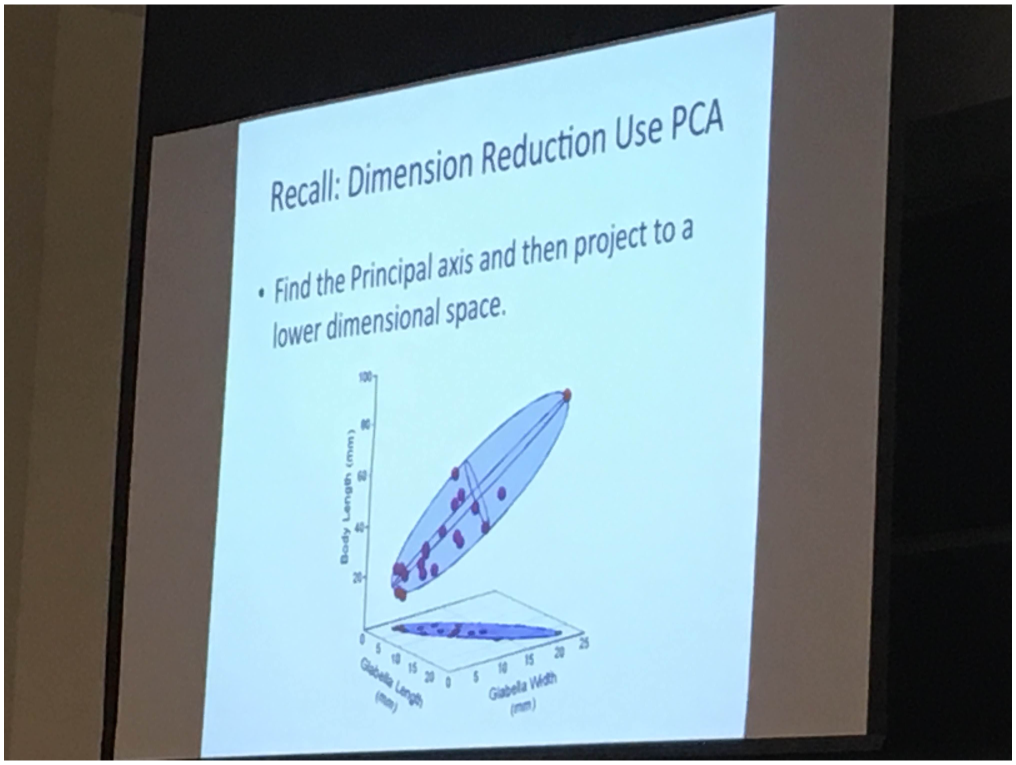
Lecture 8: Kernel PCA, One Class SVMs, and Learning Theory

October 31st, 2016

1 Kernel PCA



Recall that in principle component analysis we find an orthogonal transformation to convert a set of observed (possibly) correlated values into a set of linearly uncorrelated values. We find the principle directions first by taking the covariance matrix and finding the means μ_1 and μ_2 if we are working in two dimensions. We subtract the x values by μ_1 and the y-values by μ_2. Projection works very well here in that we maintain an ellipsoid in the subspace.



Recall: Dimension Reduction Use PCA

- Find the Principal axis and then project to a lower dimensional space.

Body Length (mm)
Glabella Length (mm)
Glabella Width (mm)

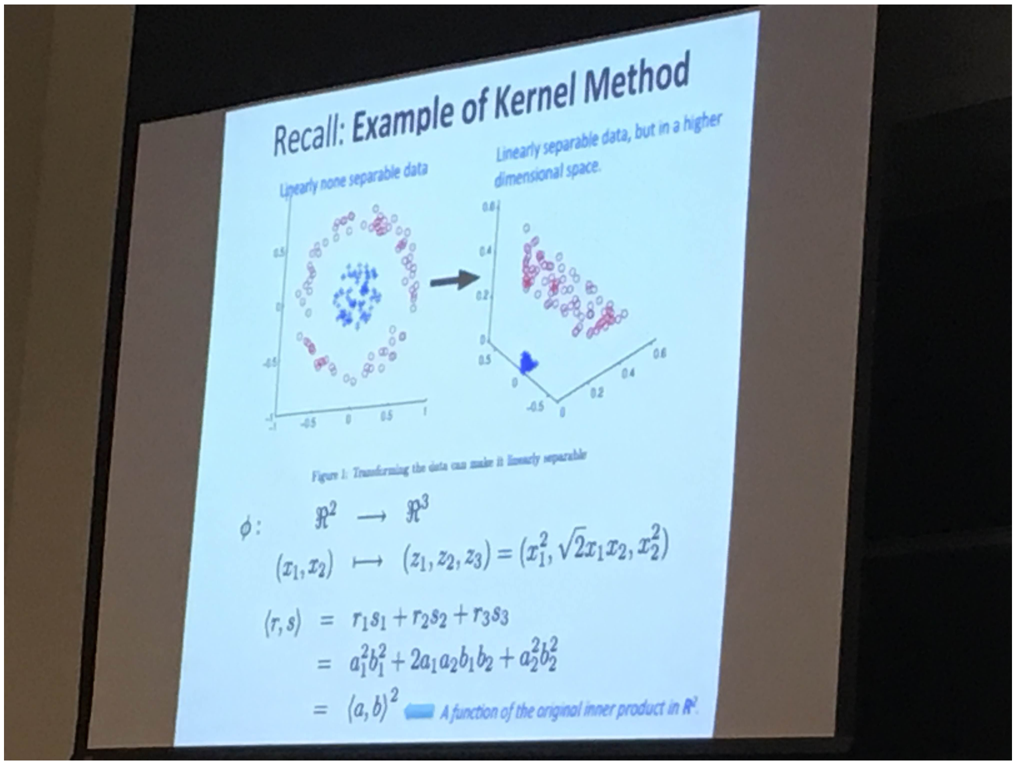
We will now implement the concept of a kernel by taking a function $K = k(x,y)$ which can be written as an inner product of another function $k(x,y) = \langle \Phi(x), \Phi(y) \rangle = \Phi(x)^T \Phi(y)$. For example we can have $k(x,y) = (x^T y + 1)^2$ or the Gaussian kernel

$$k(x,y) \;=\; \exp\left[-\frac{\|x-y\|^2}{2\sigma^2}\right].$$

Recall that we usually take the function $\Phi : \mathbb{R}^n \to \mathbb{R}^N$ each Euclidean space containing it's own inner product. We call the space $\left(\mathbb{R}^N,\ \langle \cdot, \cdot \rangle\right)$ the **feature space**. We can unambiguously define the cosine of the angle between two vectors by

$$\cos\theta \;=\; \frac{\langle \Phi(x), \Phi(y) \rangle}{\sqrt{\langle \Phi(x), \Phi(x) \rangle}\sqrt{\langle \Phi(y), \Phi(y) \rangle}}.$$

As we will see later, we know that a function $k(x,y)$ can be written as a kernel if and only if it is symmetric and positive definite.



Recall: Example of Kernel Method

Linearly none separable data

Linearly separable data, but in a higher dimensional space.

Figure 1: Transforming the data can make it linearly separable

$$\begin{aligned} \phi : \quad \Re^2 &\longrightarrow \Re^3 \\ (x_1, x_2) &\longmapsto (z_1, z_2, z_3) = (x_1^2, \sqrt{2}x_1x_2, x_2^2) \end{aligned}$$

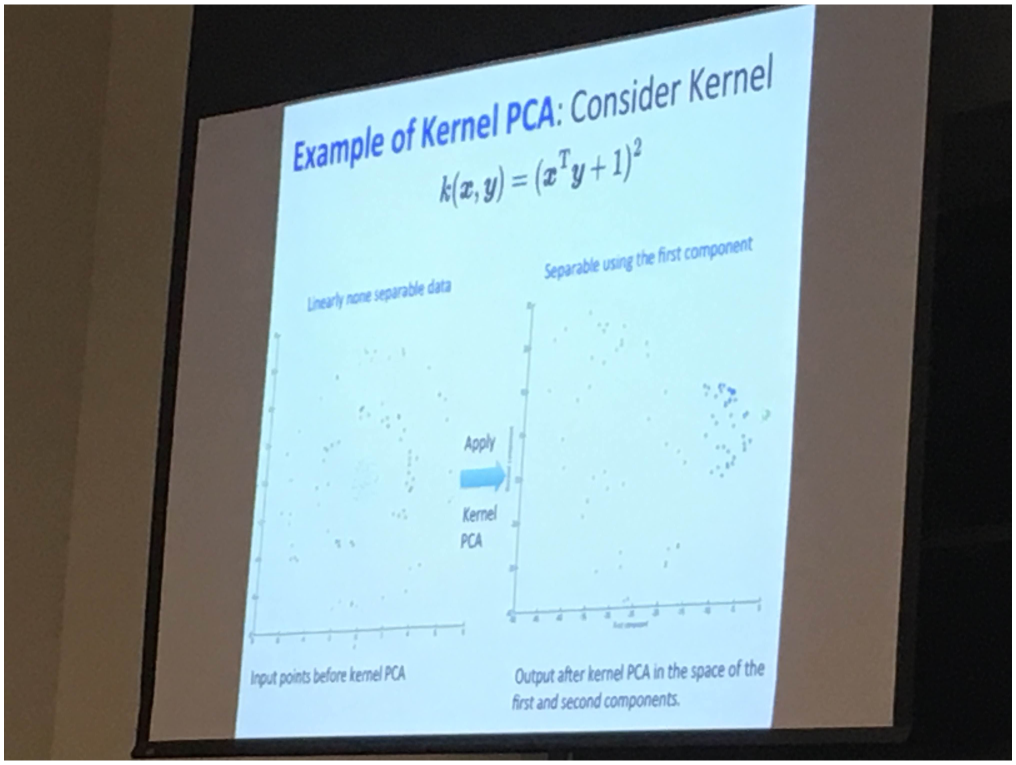
$$\begin{aligned} \langle r, s \rangle &= r_1s_1 + r_2s_2 + r_3s_3 \\ &= a_1^2b_1^2 + 2a_1a_2b_1b_2 + a_2^2b_2^2 \\ &= \langle a, b \rangle^2 \end{aligned}$$

A function of the original inner product in $\Re^2$.

What if we have the case where our data is not separable, as in the image above? What we want to do in this case is to transform the data so that it is separable in another space, often of higher dimension. Assume for instance we can have $\phi : \mathbb{R}^2 \to \mathbb{R}^3$ given by

$$\phi(x_1, x_2) = (z_1, z_2, z_3) = (x_1^2, \sqrt{2}x_1x_2, x_2^2) = \langle x_1, x_2 \rangle^2.$$

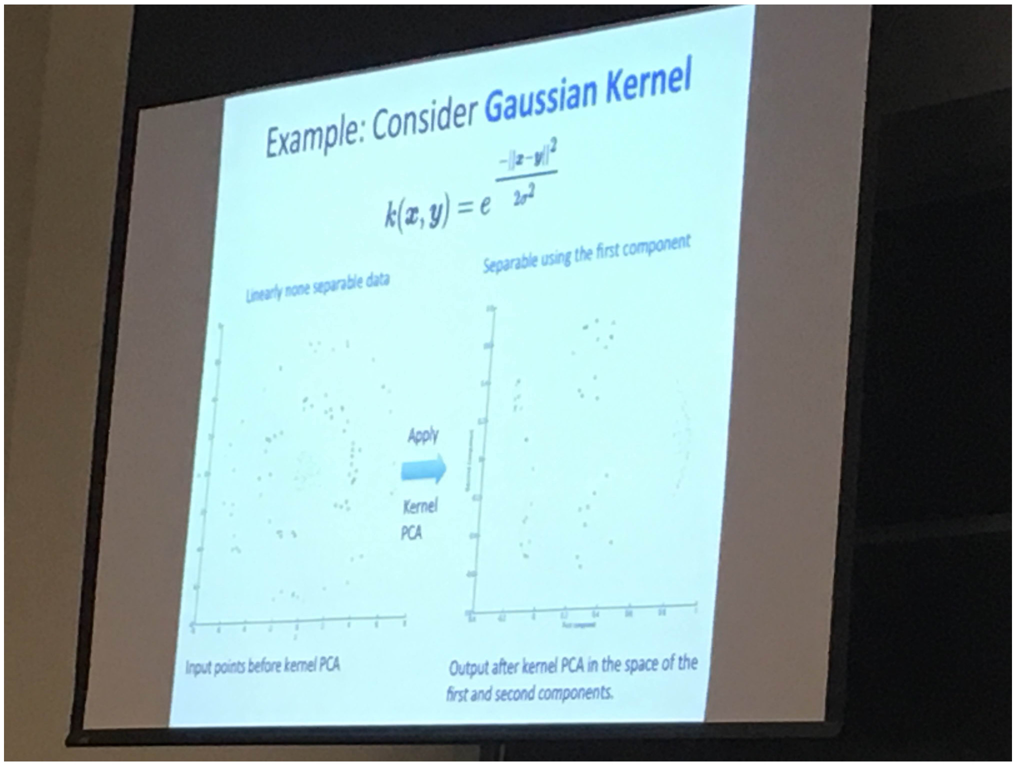
The key concept of *kernel PCA* is that it computes the principal components but projects our data onto these components. We can for instance have the kernel $k(x, y) = (x^T y + 1)^2$ and data that is in concentric circles will be formed into a cone shape:



Example of Kernel PCA: Consider Kernel

$$k(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x}^T \boldsymbol{y} + 1)^2$$

Using a Gaussian kernel gives us a smoother solution:



Example: Consider **Gaussian Kernel**

$$k(\boldsymbol{x}, \boldsymbol{y}) = e^{\frac{-\|\boldsymbol{x}-\boldsymbol{y}\|^2}{2\sigma^2}}$$

An important thing is that in addition to our data being separable, we would like our kernel to be smooth. Here is the key idea. We can cleverly avoid working directly in the feature space, since this will likely be high-dimensional. Instead of solving for the principle components themselves, we actually end up solving the projections of our data onto the PCA components.

We start with a mapping that takes our data into a higher dimensional space where the data can be linearly separated. We claim that the eigenvectors can be expressed in the following manner. If v is an eigenvector, we can write

$$v = \sum_{i=1}^{N} \alpha_i \phi(x_i).$$

Proof: Let C_F be the covariance matrix of the data in the feature space, and suppose we can write it as

$$C_F = \frac{1}{N}\sum_{i=1}^{N} \phi(x_i)\phi(x_i)^T.$$

Observe then that C_F acting on the eigenvector v is written

$$C_F v = \lambda v = \frac{1}{N}\sum_{i=1}^{N} \phi(x_i)\left[\phi(x_i)^T v\right]$$

and therefore

$$v = \sum_{i=1}^{N} \underbrace{\left(\frac{\phi(x_i)^T v}{\lambda N}\right)}_{=\alpha_i} \phi(x_i)$$

thus proving the claim $\square$

How can we find the vector of coefficients $\alpha = (\alpha_1, \ldots, \alpha_N)$? We claim that $K\alpha^T = (N\lambda)\alpha^T$ where K is the **kernel matrix**

$$K = \begin{bmatrix} \phi(x_1)^T\phi(x_1) & \phi(x_1)^T\phi(x_2) & \ldots & \phi(x_1)^T\phi(x_N) \\ \phi(x_2)^T\phi(x_1) & \phi(x_2)^T\phi(x_2) & \ldots & \phi(x_2)^T\phi(x_N) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(x_N)^T\phi(x_1) & \phi(x_N)^T\phi(x_2) & \ldots & \phi(x_N)^T\phi(x_N) \end{bmatrix}.$$

Proof: Observe that the covariance matrix satisfies $C_F = \frac{1}{N}K$, and therefore

$$C_F v = \lambda v = \frac{1}{N} K v$$

and therefore $(N\lambda)v = Kv$ but since $v = \alpha_1\phi(x_1) + \ldots + \alpha_N\phi(x_N)$ we can write it as

$$v = \begin{bmatrix} \alpha_1 & \ldots & \alpha_N \end{bmatrix}\begin{bmatrix} \phi(x_1) \\ \vdots \\ \phi(x_N) \end{bmatrix}$$

and thus $N\lambda\alpha\phi(x) = K\alpha\phi(x)$. If we multiply on the right by $\phi(x)^T$ we find

$$N\lambda\alpha\phi(x)\phi(x)^T = N\lambda\alpha K = K\alpha\phi(x)\phi(x)^T = K\alpha K$$

hence $N\lambda\alpha K = K\alpha K$ and so $(N\lambda\alpha - K\alpha)K = 0$. Now, without loss of generality we can assume (and it is quite safe to do so) that K is positive definite. In this case, this forces the term in parentheses to be the zero matrix and thus $K\alpha = N\lambda\alpha$ proving that α is an eigenvector associated to the eigenvalue $N\lambda$. In general we need to normalize (or *centralize*) the data in the ϕ-space since $\phi(x_i)$ may not have zero mean. When we centralize this we obtain

$$\hat{\phi}(x_i) = \phi(x_i) - \sum_{j=1}^{N} \phi(x_j)$$

and so we get a normalized kernel function

$$\begin{aligned}
\hat{K}(x_i, x_j) &= \left[\hat{\phi}(x_i)\right]^T \hat{\phi}(x_j) \\
&= \left[\phi(x_i) - \sum_{k=1}^{N} \phi(x_k)\right]^T \left[\phi(x_j) - \sum_{l=1}^{N} \phi(x_l)\right] \\
&= \phi(x_i)^T \phi(x_j) - \frac{1}{N}\sum_{l=1}^{N} \phi(x_i)^T \phi(x_l) - \frac{1}{N}\sum_{k=1}^{N} \phi(x_k)^T \phi(x_j) + \frac{1}{N^2}\sum_{k=1}^{N}\sum_{l=1}^{N} \phi(x_k)^T \phi(x_l) \\
&= K(x_i, x_j) - \frac{1}{N}\sum_{l=1}^{N} K(x_{i,l}) - \frac{1}{N}\sum_{k=1}^{N} K(x_{k,j}) + \frac{1}{N^2}\sum_{k=1}^{N}\sum_{l=1}^{N} K(x_k, x_l).
\end{aligned}$$

It's worth asking, why should we multiply all of this out? Because we want to **avoid** working directly in the high-dimensional feature space, the kernel function of the PCA is restricted in that it computes **not** the principal components themselves, but the projection of our data onto those components. To evaluate the projection from a point in the feature space $\Phi(X)$ onto the jth principal component v_j:

$$v_j^T \phi(x) = \left[\sum_{i=1}^{N} \alpha_i \phi(x_i)\right]^T \phi(x)$$

where we observe that $\phi(x_i)^T \phi(x) = K(x_i, x)$ which is an element of the kernel. Therefore we find

$$v_j^T \phi(x) = \sum_{i=1}^{N} \alpha_j K(x, x_i) \triangleq y_j.$$

We can summarize the above as follows:

- Choose a kernel function,
- Construct the normalized kernel matrix,
- Solve an eigenvalue problem where $K\alpha_j = N\lambda_j \alpha_j$ and

  $$\alpha_j = \begin{bmatrix} \alpha_{j1} \\ \alpha_{j2} \\ \vdots \\ \alpha_{jN} \end{bmatrix}$$

  is an eigenvector associated to λ of the kernel matrix K. Finally set

  $$y_j = \sum_{i=1}^{N} \alpha_{ji} K(x, x_i)$$

  for $j \in \{1, \ldots, N\}$.

An important theorem is **Mercer's theorem** which shows that as long as the function $k(x, y)$ is symmetric positive-definite then it can be written as a kernel.

Theorem 1 (Mercer's Theorem) *A symmetric function $K(x, y)$ can be expressed as an inner product $K(x, y) = \langle \phi(x), \phi(y) \rangle$ for some ϕ if and only if $K(x, y)$ is positive semi-definite meaning*

$$\int K(x, y) g(x), g(y) dx dy \geq 0$$

for all suitable functions g, or equivalently semi-definite in the discrete case as a matrix.

There are numerous applications of kernel methods:

1. Turn any linear model into a non-linear model applying the kernel trick to the model.
2. Support vector machines,
3. Gaussian processes,
4. Principal component analysis,
5. Canonical correlational analysis,
6. Ridge regression,
7. Spectral clustering,
8. Linear adaptive filters,
9. The kernel perceptron,

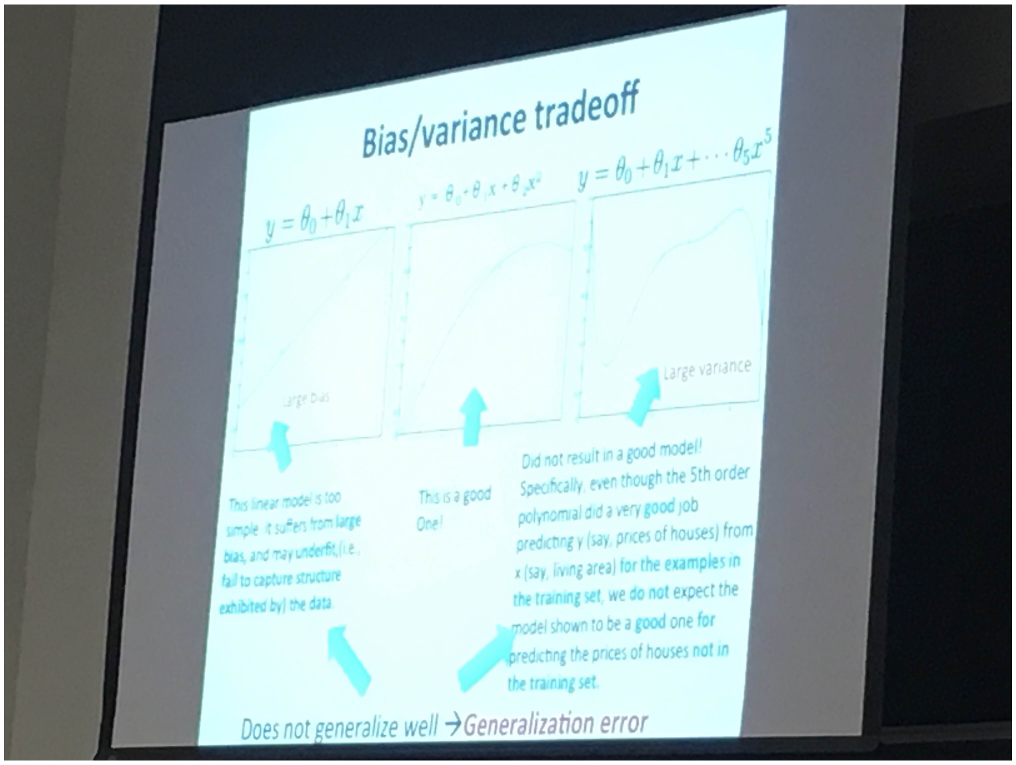
and many other algorithms.

2 One-Class Support Vector Machines

The goal of machine learning is to distinguish test data. But if we only have one class of data and the goal is to test new data, then how well are we able to train our machine. One method for this is called **one-class support vector machine**. This is only here as a side note since most of this is covered in the textbook.

3 Learning Theory

We will be discussing bias/variance trade-off, union and Chernoff/Hoeffding bounds, and briefly on the VC dimension. This section will closely follow Prof. Ng's notes of Stanford University.


$y = \theta_0 + \theta_1 x$
$y = \theta_0 + \theta_1 x + \theta_2 x^2$
$y = \theta_0 + \theta_1 x + \cdots \theta_5 x^5$
Large bias
Large variance

Bias/variance tradeoff

This linear model is too simple: it suffers from large bias, and may underfit (i.e., fail to capture structure exhibited by) the data.

This is a good One!

Did not result in a good model! Specifically, even though the 5th order polynomial did a very good job predicting y (say, prices of houses) from x (say, living area) for the examples in the training set, we do not expect the model shown to be a good one for predicting the prices of houses not in the training set.

Does not generalize well → Generalization error

Observe in the above fitting problem, we see that if we underfit the data we end up incurring a large bias in our fit, but overfitting yields a similar problem. If we overfit the data then we pass through every data point but the model is now very sensitive to new inputs. Fitting the curve with a degree-2 polynomial is a good fit. We see that we incur a large *generalization error* in the case of underfitting or overfitting. There are two different types of error. What **learning theory** does is relates bias error to large variance. To illustrate this, say we have a training set with lots of spurious data. We may obtain a model with large generalization error particularly with large variance. There is a tradeoff here. If our model is too "simple" with few parameters, we may have a large bias but very small variance. Similarly, if the model is too "complex" then our model may have too many parameters and may be prone to high variance (but smaller bias). To do this analysis we need two bounds.

Definition 1 *Let $A_1, \dots, A_k$ be k many different events. Then the **union bound** is that*

$$p(A_1 \cup \dots, \cup A_k) \;\; \le \;\; \sum_{i=1}^{k} P(A_i).$$

Definition 2 *Let $Z_1, \dots, Z_m$ be m-many IID random variables drawn from a Bernoulli distribution with parameter ϕ, meaning $P\{Z_i = 1\} = \phi$. Define*

$$\hat{\phi} \;\; = \;\; \frac{1}{m}\sum_{i=1}^{m} Z_i$$

*which is the mean of the random variables. Let $\gamma > 0$ be fixed, then the **Chernoff/Hoeffding bound** is given by*

$$P\left\{\left|\phi - \hat{\phi}\right| > \gamma\right\} \;\; \le \;\; 2\exp\left(-2\gamma^2 m\right).$$

The meaning of the Chernoff/Hoeffding inequality is that the average of the Bernoulli random variable then our probability of being far from the mean decreases exponentially with the number of random samples drawn from the distribution. As an example, if someone has a biased coin whose probability of landing on heads is $\phi \ge 0.5$, then if you toss it m times and calculate the fraction of times that it came up heads will be a good estimate of ϕ with high probability (provided that m is large).

In our further analysis we will need these two bounds. To make our discussion easier, let's say that our parameter $\gamma \in \{0, 1\}$. Everything we do with this restriction does in fact generalize to other classes of problems. Suppose we have a training set $S = \left\{\left(x^{(i)}, y^{(i)}\right)\right\}$ for $i \in \{1, \dots, m\}$. The training samples are drawn IID from a distribution $\mathcal{D}$. For a hypothesis h, we define the **training error** to be

$$\hat{\varepsilon}(h) \;\; = \;\; \frac{1}{m}\sum_{i=1}^{m} \mathbb{1}\left\{h(x^{(i)}) \ne y^{(i)}\right\}.$$

which is just the fraction of the training samples that the hypothesis h misclassifies. The **generalization error** is defined as

$$\varepsilon(h) \;\; = \;\; P_{(x,y)\sim\mathcal{D}}(h(x) \ne y)$$

which is the probability that given a new training point (x, y) drawn from the same distribution, that h will misclassify it. Our assumption that the new data point is drawn fom the same distribution is one of the PAC assumptions (*"probably approximately correct"*). In a certain sense the training error will be close to the generalization error with high probability. In the setting of linear classification, let $h_\theta(x) = \mathbb{1}\{\theta^T x \ge 0\}$. What's a reasonable fitting of the parameters θ? One approach is to minimize the training error and pick

$$\hat{\theta} \;\; = \;\; \arg\min_{\theta} \hat{\varepsilon}(h_\theta).$$

This process is called **empirical risk minimization** (ERM). We think of ERM as the most basic learning algorithm, and the one we focus on in the following treatment.

We define the **hypothesis class** $\mathcal{H}$ as

$$\mathcal{H} \;=\; \left\{h_\theta \,:\, h_\theta(x) = \mathbb{1}\{\theta^T x \geq 0\},\; \theta \in \mathbb{R}^{n+1}\right\}.$$

We can now think of ERM as minimization over the class of function on $\mathcal{H}$. For the sake of this class, we will assume that $\mathcal{H}$ is finite. We would like to give guarantees on the generalization error of h. Our strategy for doing so will be in two parts: we will show that $\hat{\varepsilon}(h)$ is a reliable estimation of $\varepsilon(h)$ for all h. We will also show that this implies an upper bound on the generalization error of $\hat{h}$. We find that $\hat{\varepsilon}(h)$ is exactly the mean of m random variables drawn IID from a Bernoulli distribution, hence we can use the Chernoff bounds. If we generalize for all the h_i on $\mathcal{H}$, we find that

$$P\{\exists\, h \in \mathcal{H} \,:\, |\varepsilon(h_i) - \hat{\varepsilon}(h_i)| > \gamma\} \;\leq\; 2k \exp\left(-2\gamma^2 m\right).$$

What we've shown is that this is true simultaneously for any $h \in \mathcal{H}$. Subtracting both sides by 1, we find that

$$P\{\neg\exists\, h \in \mathcal{H} \,:\, |\varepsilon(h_i) - \hat{\varepsilon}(h_i)| > \gamma\} \;\geq\; 1 - 2k \exp\left(-2\gamma^2 m\right).$$

where $\neg\exists$ means "there does not exist". We will have that $\varepsilon(h)$ will be within γ of $\hat{\varepsilon}(h)$ for all $h \in \mathcal{H}$. This is called *uniform convergence* since this holds for everything in $\mathcal{H}$. The quantities m, γ, δ can be adjusted and we can solve for bounds in terms of one variable in terms of the other two. For instance, given γ and $\delta > 0$, then in order to have a probability of at least $1 - \delta$ for training error to be within γ is given by

$$m \;\geq\; \frac{1}{2\gamma^2} \log \frac{2k}{\delta}$$

where $k = |\mathcal{H}|$. We can find similar inequalities for the other cases.

Theorem 2 *Let $|\mathcal{H}| = k$ and let any m, δ be fixed. Then with probability at least $1 - \delta$ we have that*

$$\varepsilon\left(\hat{h}\right) \;\leq\; \left(\min_{h \in \mathcal{H}} \varepsilon(h)\right) + 2\sqrt{\frac{1}{2m} \log \frac{2k}{\delta}}.$$

Corollary 1 *Let $|\mathcal{H}| = k$ and let δ, γ be fixed. Then for $\varepsilon\left(\hat{h}\right) \leq \min_{h \in \mathcal{H}} \varepsilon(h) + 2\gamma$ to hold with probability at least $1 - \delta$, suffices that*

$$m \;\geq\; \frac{1}{2\gamma^2} \log \frac{2k}{\delta} \;=\; O\left(\frac{1}{\gamma^2} \log \frac{k}{\delta}\right).$$