- In this course so far, we have discussed various strategies for learning classifiers/regression models.
- We looked at simple nearest neighbour classifier.
- We have discussed generative models for classification (such as Bayes and Naive-bayes).
- We have considered linear discriminative models (such as logistic regression, linear discriminant analysis).
- We have discussed nonlinear discriminative models (such as neural networks, SVMs)
- We looked at generative models such as graphical models, RBMs, VAEs
- We also looked at Bagging and Boosting and classifier ensembles.
- All these involve supervised learning.

- We did not discuss any specific methods to decide on the features to be used.
- The neural network viewpoint is that ideally we should automatically learn the features.
- Then one can give the 'raw pattern' as input.
- But, in many cases, we have to decide on the features.
- Measure many possible features and 'pick good ones'
- Often, performance is critically dependent on features used
- There are general methods for supervised learning of feature subsets.

Feature Selection

- Feature selection refers to learning of best subset of features.
- If we totally have n features then there are 2^n possible subsets of features.
- We can think of feature selection as being similar to model selection.
- So, we can use holdout validation or cross validation for this.
- However, generally n is large and hence this is not computationally feasible.
- So, we choose among only some of the subsets.

- There are two general approaches to feature selection – wrapper methods and filter methods.
- Wrapper methods essentially do model selection. That is, they train classifiers with different subsets of features and then select among them through validation.
- Here, feature selection is wrapped around classifier learning.
- Filter methods try to rank different features based on some score function and then filter out least relevant features.

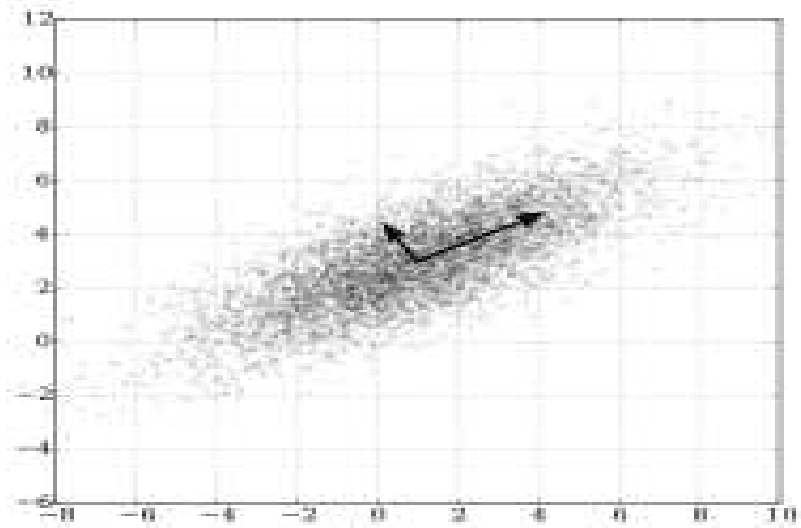
- Here is a simple wrapper method
- At iteration k, we have a current feature subset $\mathcal{F}_k$. We want to find which is the best feature to add to the set.
- For this, we add any one of the remaining features and learn a classifier. We do this for each of the remaining ones and thus find the best one to add.
- We can start with the null feature set and run the iterations n times.
- Thus we need to learn classifiers with only $O(n^2)$ feature subsets.
- We choose the best subset based on estimate of generalization error of classifier obtained through validation.

- Here is a simple filter-based method for feature selection.
- We assign a score to each feature based on how well it correlates with class label.
- The score can be mutual information between x, the feature, and y, the class label.
- Given two random variables, x, y, the mutual information is the KL divergence between the joint distribution $f(x, y)$ and the product of the marginals $f(x)f(y)$:

$$MI(x, y) = \sum_{x,y} f(x, y) \ln \left(\frac{f(x, y)}{f(x)f(y)} \right)$$

- Then we choose the K best features where K itself is chosen through, e.g., cross-validation.

- We can also think of ‘feature learning’ in an unsupervised fashion.
- We can ask: what are the best ‘coordinates’ to represent some given data.



(Image taken from Wikipedia)

- Here we are looking for techniques to transform the original feature vector into a new feature vector
- We want to determine this transformation based on a set of data points given. (Unsupervised)
- We may want to do this to reduce the dimensionality of the feature vector without loosing too much information.
- We may want to do this to improve the features (e.g., make them uncorrelated).
- One such technique is the Principal Component Analysis (PCA).
- This is a general-purpose method useful in many problems of data analysis and machine learning.

Principal Component Analysis

- We can think of PCA as a useful linear transformation of the feature vector.
- For dimensionality reduction, we essentially want to project the data onto a lower dimensional subspace.
- We can define PCA as projection onto a subspace such that
  - variance of projected data is maximized, or
  - mean-square error (in approximating a feature vector with its projection) is minimized.
- Here we do not take into consideration the class-label information.

PCA as dimensionality reduction

- Let $\{X_1, \cdots, X_n\}$, $X_i \in \Re^d$ be the given data.
- Suppose we want to project it onto an m-dimensional subspace.
- Let $U_1, \cdots, U_m$ denote an orthonormal basis for the m-dimensional subspace.
- Let $U_1, \cdots, U_m, U_{m+1}, \cdots, U_d$ denote the extension of this basis to whole of $\Re^d$.
- Note that

$$X_i = \sum_{j=1}^{d} \left(X_i^T U_j\right) U_j$$

  Note that we would think of the parenthetic expression as a symbol for a scalar. (We would not open the bracket by multiplying!)

- Let $\tilde{X}_i$ (which would be in the m-dimensional subspace), denote the approximation of X_i.
- Then we can write

$$\tilde{X}_i = \sum_{j=1}^{m} z_{ij}\, U_j \; + \; \sum_{j=m+1}^{d} \beta_j\, U_j$$

- Note that the second term does not depend on i.
- We need to find the z_{ij} and β_j to get an approximation with least mean-square error.

We want z_{ij} and β_j to minimize

$$
\begin{aligned}
J &= \frac{1}{n}\sum_{i=1}^{n} ||X_i - \tilde{X}_i||^2 \\
&= \frac{1}{n}\sum_{i=1}^{n} \left|\left| \sum_{j=1}^{d} (X_i^T U_j) U_j - \sum_{j=1}^{m} z_{ij} U_j - \sum_{j=m+1}^{d} \beta_j U_j \right|\right|^2 \\
&= \frac{1}{n}\sum_{i=1}^{n} \left|\left| \sum_{j=1}^{m} (X_i^T U_j - z_{ij}) U_j + \sum_{j=m+1}^{d} (X_i^T U_j - \beta_j) U_j \right|\right|^2 \\
&= \frac{1}{n}\sum_{i=1}^{n} \left[\sum_{j=1}^{m} (X_i^T U_j - z_{ij})^2 + \sum_{j=m+1}^{d} (X_i^T U_j - \beta_j)^2 \right]
\end{aligned}
$$

$$J = \frac{1}{n} \sum_{i=1}^{n} \left[\sum_{j=1}^{m} (X_i^T U_j - z_{ij})^2 + \sum_{j=m+1}^{d} (X_i^T U_j - \beta_j)^2 \right]$$

- For any indices s and t ($1 \leq s \leq n$, $1 \leq t \leq m$),

$$\frac{\partial J}{\partial z_{st}} = 0 \Rightarrow 2(X_s^T U_t - z_{st}) = 0 \Rightarrow z_{st} = X_s^T U_t$$

- Similarly, for any index t, $t \geq m + 1$,

$$\frac{\partial J}{\partial \beta_t} = 0 \Rightarrow \frac{1}{n} \sum_{i=1}^{n} 2(X_i^T U_t - \beta_t) = 0$$

$$\Rightarrow \quad \beta_t = \left(\frac{1}{n} \sum_{i=1}^{n} X_i \right)^T U_t = \bar{X}^T U_t$$

where $\bar{X}$ is the mean of the data vectors.

- Thus, for a given basis $\{U_j\}$, we get

$$\begin{aligned}
\tilde{X}_i &= \sum_{j=1}^{m} (X_i^T U_j)\, U_j + \sum_{j=m+1}^{d} (\bar{X}^T U_j)\, U_j \\
&= \sum_{j=1}^{m} (X_i^T U_j - \bar{X}^T U_j)\, U_j + \sum_{j=1}^{d} (\bar{X}^T U_j)\, U_j \\
&= \sum_{j=1}^{m} ((X_i - \bar{X})^T U_j)\, U_j + \bar{X}
\end{aligned}$$

- Hence given any subspace (that is, the $\{U_j\}$) we know the projections and hence the errors.
- We now need to find the subspace that minimizes the error.

- Recall that

$$X_i = \sum_{j=1}^{d} (X_i^T U_j)\, U_j$$

- We have

$$\tilde{X}_i = \sum_{j=1}^{m} (X_i^T U_j)\, U_j \; + \; \sum_{j=m+1}^{d} (\bar{X}^T U_j)\, U_j$$

- Hence,

$$X_i - \tilde{X}_i = \sum_{j=m+1}^{d} (X_i^T U_j - \bar{X}^T U_j) U_j$$

$$||X_i - \tilde{X}_i||^2 = \sum_{j=m+1}^{d} ((X_i - \bar{X})^T U_j)^2$$

- Hence we have

$$
\begin{aligned}
J &= \frac{1}{n}\sum_{i=1}^{n}\sum_{j=m+1}^{d}((X_i-\bar{X})^T U_j)^2 \\
&= \sum_{j=m+1}^{d}\frac{1}{n}\sum_{i=1}^{n} U_j^T(X_i-\bar{X})(X_i-\bar{X})^T U_j \\
&= \sum_{j=m+1}^{d} U_j^T\left(\frac{1}{n}\sum_{i=1}^{n}(X_i-\bar{X})(X_i-\bar{X})^T\right)U_j
\end{aligned}
$$

- Let

$$S = \frac{1}{n} \sum_{i=1}^{n} (X_i - \bar{X})(X_i - \bar{X})^T$$

- This is the data covariance matrix which is a real symmetric matrix.
- Now, our mean square error is

$$J = \sum_{j=m+1}^{d} U_j^T S U_j$$

- We need to find U_j (that is the correct subspace) to minimize J.

- Suppose we want to find vector U, to

$$\begin{aligned} \min_{U} \quad & U^T S U \\ \text{subject to} \quad & U^T U = 1 \end{aligned}$$

- The lagrangian for the problem is

$$U^T S U + \lambda(1 - U^T U)$$

- Equating the gradient of Lagrangian to zero,

$$SU = \lambda U$$

- This means U should be an eigen vector of S.
- The corresponding value is: $U^T S U = U^T \lambda U = \lambda$.
- The minimizing U is eigen vector corresponding to least eigen value.

- We want to minimize $J = \sum_{j=m+1}^{d} U_j^T S U_j$.
- Since S is real symmetric, all its eigen values are real and it would have a set of orthonormal eigen vectors that span the space.
- So, to minimize J, we should choose, $U_{m+1}, \cdots, U_d$ to be the $(d-m)$ eigen vectors corresponding to the least $(d-m)$ eigen values.
- Since the vectors $U_{m+1}, \cdots, U_d$ span the orthogonal complement of our desired m-dimensional space, that space has orthonormal basis $U_1, \cdots, U_m$ which are the remaining eigen vectors of S.

The final transformation

- To sum up, we get our lower dimensional representation as follows.
- We form the data covariance matrix

$$S = \frac{1}{n} \sum_{i=1}^{n} (X_i - \bar{X})(X_i - \bar{X})^T$$

- Let $U_1, \cdots, U_d$ be the orthogonal set of eigen vectors of S, arranged in decreasing order of the corresponding eigen values.
- Now our approximation is

$$\tilde{X}_i = \sum_{j=1}^{m} (X_i^T U_j)\, U_j \; + \sum_{j=m+1}^{d} (\bar{X}^T U_j)\, U_j$$

- We can rewrite this as

$$\begin{aligned}
\tilde{X}_i &= \sum_{j=1}^{m} (X_i^T U_j - \bar{X}^T U_j)\, U_j \; + \sum_{j=1}^{d} (\bar{X}^T U_j)\, U_j \\
&= \sum_{j=1}^{m} (X_i^T U_j - \bar{X}^T U_j)\, U_j \; + \; \bar{X} \\
\tilde{X}_i - \bar{X} &= \sum_{j=1}^{m} ((X_i - \bar{X})^T U_j)\, U_j
\end{aligned}$$

- We can assume $\bar{X} = 0$ without loss of generality.
- This is because we can always work with mean-subtracted data.

- In our representation, $\tilde{X}_i$ is a d-dimensional vector which is in an m-dimensional subspace of $\Re^d$.
- We can write it as a m-component vector as

$$\tilde{X}_i = [U_1 \cdots U_m]^T X_i = A^T X_i$$

  where A is a $d \times m$ matrix whose columns are $U_1, \cdots, U_m$.
- We are projecting X_i onto the space spanned by the eigen vectors $U_1, \cdots, U_m$.
- The projections are called the principal components.

- Suppose A is the $d \times d$ matrix whose columns are the (orthonormal) eigen vectors of S.
- Now A represents an orthonormal transformation of $\Re^d$ onto itself.
- Consider $\tilde{X} = A^T X$
- This transforms the data into their principal components.

PCA as capturing most of the variance

- Now let us look at the alternate viewpoint.
- We want to find a m-dimensional subspace in which the projected data would have maximum variance.
- The solution turns out to be the same.
- That is, we need to project data onto the space spanned by $U_1, \cdots, U_m$, the eigen vectors corresponding to the top m eigen values of the data covariance matrix.

- First consider the case $m = 1$.
- Let U_1 be the unit vector for the one dimensional subspace.
- The projected data would be $(X_i^T U_1)U_1$. Its mean is $(\bar{X}^T U_1)U_1$.
  The variance is

$$\begin{aligned}\frac{1}{n}\sum_{i=1}^{n}(X_i^T U_1 - \bar{X}^T U_1)^2 &= \frac{1}{n}\sum_{i=1}^{n} U_1^T (X_i - \bar{X})(X_i - \bar{X})^T U_1 \\ &= U_1^T S U_1\end{aligned}$$

- From our earlier analysis, this variance is maximized when U_1 is the eigen vector corresponding to the highest eigen value of S.
- Now suppose $m = 2$ and hence we want to add another direction.
- So, we need U_2, which has unit norm and such that $U_1^T U_2 = 0$ and such that the projected variance would be maximum.
- It is easy to see that U_2 would be the eigen vector corresponding to the second highest eigen value.

- Thus, if we want an m-dimensional space so that the projected data would have highest variance then that space is the one spanned by the m eigen vectors of S corresponding to top m eigen values.
- These directions U_i are called the principal directions.
- The projected values are called the principal components.

- Thus, PCA is essentially a projection of the data vectors onto a space spanned by the eigen vectors of the covariance matrix.
- This is a linear transform of the original data: $\tilde{X} = A^T X$.
- The columns of the matrix A are the eigen vectors.
- By using only top m eigen vectors we get dimensionality reduction.
- The residual error is the sum of the least $(d - m)$ eigen values.
- This is a good way to decide on the value of m.

PCA and Whitening Transform

- While PCA is mainly used for dimensionality reduction, there are other uses too.
- We can use PCA to find a linear transform of the feature vector so that the transformed features are uncorrelated.
- We discussed this earlier in the context of normalizing inputs for a neural network.
- We recall this transform to see its relation to PCA.

- As earlier, let $S = \frac{1}{n} \sum_{i=1}^{n} (X_i - \bar{X})(X_i - \bar{X})^T$ be the data covariance matrix.
- Let $\lambda_1, \cdots, \lambda_d$ be the eigenvalues of S arranged in a decreasing order.
- Let $U_1, \cdots, U_d$ be the corresponding eigen vectors (which are orthonormal).
- Let L be a diagonal matrix with λ_i being the diagonal entries.
- Let $\tilde{U}$ be a $d \times d$ matrix whose columns are U_j.

- Now the eigen value equations for S are

$$S\tilde{U} = \tilde{U}L$$

- Now define a transformation of the data vectors given by

$$Z_i = L^{-0.5}\tilde{U}^T(X_i - \bar{X})$$

  where $\bar{X}$ is the mean of the data vectors.
- We are essentially scaling each principal component by square root of the corresponding eigen value.
- It is easy to see that mean of Z_i is zero:

- The covariance matrix for the data Z_i is now given by

$$
\begin{aligned}
S_Z &= \frac{1}{n}\sum_{i=1}^{n} Z_i\, Z_i^T \\
&= \frac{1}{n}\sum_{i=1}^{n} L^{-0.5}\, \tilde{U}^T\, (X_i - \bar{X})\, (X_i - \bar{X})^T\, \tilde{U}\, L^{-0.5} \\
&= L^{-0.5}\, \tilde{U}^T\, S\, \tilde{U}\, L^{-0.5} \\
&= L^{-0.5}\, \tilde{U}^T\, \tilde{U}\, L\, L^{-0.5} \\
&= I
\end{aligned}
$$

- Thus, if we transform X_i to Z_i by

$$Z_i = L^{-0.5}\tilde{U}^T(X_i - \bar{X})$$

  then the transformed data are zero-mean, unit variance and uncorrelated.
- This is the transformation we saw earlier for normalizing inputs to a neural network.
- This idea comes from PCA.
- This is called PCA whitening.

ZCA Whitening

- As earlier let $\tilde{U}$ be matrix of eigen vectors, L diagonal matrix of eigen values and assume mean of X is zero..
- The PCA whitening transform is: $Z = L^{-0.5}\tilde{U}^T X$
- If R is any orthogonal matrix then $Z_1 = RZ$ would also be a whitening transform:

$$E[Z_1 Z_1^T] = E[RZZ^T R^T] = RIR^T = I$$

- Thus the PCA whitening transform is not unique.
- If we take $R = \tilde{U}$ then it is called ZCA whitening.

ZCA Whitening

- Recall that PCA transform is: $\tilde{X} = \tilde{U}^T X$.
- So, PCA whitening is $L^{-0.5}\tilde{X}$. This need not be close to X in mean-square sense.
- Suppose we are looking for a transformation $\tilde{X} = A^T X$ such that the output is whitened and, in addition, it is close to the original data in mean-square sense. That transform is ZCA whitening.
- The ZCA whitening is more useful when data are images.
- The ZCA whitened images look similar to the original but the PCA whitened ones may not.

High dimensional PCA

- To implement PCA, we need to find eigen vectors of S which is an $d \times d$ matrix.
- There can be situations where the feature vector dimension, d, is large (and $n < d$).
- For example, image-based pattern recognition. Here we may have $d >> n$.
- When d is large, finding eigen vectors of S can be computationally expensive.
- In such situations we can reformulate PCA so that we find eigen vectors of only a $n \times n$ matrix.

- Recall

$$S = \frac{1}{n} \sum_{i=1}^{n} (X_i - \bar{X})(X_i - \bar{X})^T$$

- Since S is sum of n rank-1 matrices, its rank can not exceed n.
- Thus, anyway, $d - n$ eigen values of S would be zero and hence we do not need those eigen vectors.

- Let A be the $n \times d$ matrix whose i^{th} row is $(X_i - \bar{X})^T$.
- Then, we have $S = \frac{1}{n} A^T A$.
- Let U_i be an eigen vector of S for eigen value $\lambda_i > 0$.

$$\frac{1}{n} A^T A \, U_i = \lambda_i \, U_i$$

- This implies

$$\lambda_i \, (AU_i) = A \left(\frac{1}{n} \, A^T A \, U_i \right) = \frac{1}{n} \, A A^T (AU_i)$$

- Thus, λ_i is also an eigen value of the $n \times n$ matrix $\frac{1}{n} A A^T$ and AU_i is its corresponding eigenvector.

- Thus, for PCA now, we do not need to find the eigenvectors of S.
- It is enough to find eigenvectors of the $n \times n$ matrix $\frac{1}{n} A A^T$.

- Specifically, let V_i be the eigenvector for an eigenvalue λ_i of the $n \times n$ matrix $\frac{1}{n} A A^T$.
- Then we have

$$\frac{1}{n} A A^T \, V_i = \lambda_i \, V_i, \qquad \text{and hence}$$

$$\lambda_i \, \left(A^T V_i \right) = A^T \left(\frac{1}{n} A A^T \, V_i \right) = \left(\frac{1}{n} A^T A \right) \, \left(A^T V_i \right)$$

- Thus, we get all required eigenvectors of S as $A^T V_i$.

Kernel PCA

- PCA is essentially a linear transform of the data.
- We can think of a non-linear analogue through the kernel trick.
- Suppose we use $\phi : \Re^d \rightarrow \Re^M$ to map the data to a new (possibly high dimensional) feature space.
- Suppose we want to do PCA in $\Re^M$.
- We can use the Kernel trick for that.

- Our data in the new space is: $\phi(X_i),\ i = 1, \cdots, n$.
- For now, let us assume $\sum \phi(X_i) = 0$.
- We assume

$$k(X, X') = \phi(X)^T \phi(X')$$

- The data covariance matrix now is

$$C = \frac{1}{n} \sum_{i=1}^{n} \phi(X_i)\phi(X_i)^T$$

- Our task is to find eigen vectors of C without working in the new feature space.

- The eigen vectors of C satisfy: $CV_i = \lambda_i V_i$.
- Substituting for C gives us

$$CV_i = \frac{1}{n} \sum_{j=1}^{n} \phi(X_j) \left(\phi(X_j)^T V_i \right) = \lambda_i V_i$$

- Implies V_i is a linear combination of $\phi(X_j)$.
- Let

$$V_i = \sum_{j=1}^{n} a_{ij} \phi(X_j), \quad i = 1, \cdots, M.$$

- Substituting for C and V_i in the eigen value equation:

$$CV_i = \lambda_i V_i$$

$$\frac{1}{n} \sum_{j=1}^{n} \phi(X_j)\phi(X_j)^T \sum_{m=1}^{n} a_{im}\phi(X_m) = \lambda_i \sum_{m=1}^{n} a_{im}\phi(X_m)$$

- Premultiplying both sides by $\phi(X_l)^T$ for some l,

$$\sum_{j=1}^{n} \phi(X_l)^T\phi(X_j)\phi(X_j)^T \sum_{m=1}^{n} a_{im}\phi(X_m) = n\lambda_i\phi(X_l)^T \sum_{m=1}^{n} a_{im}\phi(X_m)$$

$$\sum_{j=1}^{n} \phi(X_l)^T \phi(X_j) \phi(X_j)^T \sum_{m=1}^{n} a_{im} \phi(X_m) = n\lambda_i \phi(X_l)^T \sum_{m=1}^{n} a_{im} \phi(X_m)$$

Hence

$$\sum_{j=1}^{n} \phi(X_l)^T \phi(X_j) \sum_{m=1}^{n} a_{im} \phi(X_j)^T \phi(X_m) = n\lambda_i \sum_{m=1}^{n} a_{im} \phi(X_l)^T \phi(X_m)$$

Now, using the kernel function we get

$$\sum_{j=1}^{n} k(X_l, X_j) \sum_{m=1}^{n} a_{im} k(X_j, X_m) = n\lambda_i \sum_{m=1}^{n} a_{im} k(X_l, X_m)$$

- Using the kernel function we got (for all l)

$$\sum_{j=1}^{n} k(X_l, X_j) \sum_{m=1}^{n} a_{im} k(X_j, X_m) = n\lambda_i \sum_{m=1}^{n} a_{im} k(X_l, X_m)$$

$$\sum_{m=1}^{n} \left(\sum_{j=1}^{n} k(X_l, X_j) k(X_j, X_m) \right) a_{im} = n\lambda_i \sum_{m=1}^{n} a_{im} k(X_l, X_m)$$

- Let

$$K = [K_{ij}] \text{ where } K_{ij} = k(X_i, X_j)$$

- Then

$$\left(K^2\right)_{lm} = \sum_{j} K_{lj} K_{jm}$$

- Then we have

$$\sum_{m=1}^{n} K_{lm}^2 \, a_{im} = n\lambda_i \sum_{m=1}^{n} K_{lm} \, a_{im}, \; \forall l$$

- We can write this in matrix notation as

$$K^2 \, \mathbf{a_i} = n\lambda_i \, K \, \mathbf{a_i}, \; i = 1, \cdots, n$$

  where $\mathbf{a_i}$ is a column vector whose m^{th} component is a_{im}
- If we know all λ_i and $\mathbf{a_i}$ then we have eigen values and eigen vectors of C.
- So, we need to find all λ_i and $\mathbf{a_i}$ satisfying the above.

- Suppose $\mathbf{a_i}, \lambda_i$ satisfy

$$K\ \mathbf{a_i} = n\lambda_i\ \mathbf{a_i},\ i = 1, \cdots, n$$

- These will then satisfy

$$K^2\ \mathbf{a_i} = n\lambda_i\ K\ \mathbf{a_i},\ i = 1, \cdots, n$$

- Does it give all $\mathbf{a_i}$ that we want?
- The difference is essentially in terms of eigen vectors of K having zero eigen values.

- We can find the relevant $\mathbf{a_i}$ by solving

$$K\ \mathbf{a_i} = n\lambda_i\ \mathbf{a_i}$$

  which is same as finding eigen values and eigen vectors of K.
- From the eigen spectrum of K, we can find all the needed λ_i and $\mathbf{a_i}$.
- That is, if $\mathbf{b}$ is an eigen vector of K corresponding to eigen value μ then we have

$$K\ \mathbf{b} = \mu\mathbf{b}$$

  and hence one of the λ_i is μ/n and the corresponding $\mathbf{a}_i$ is $\mathbf{b}$.

- We want λ_i, the eigen values, and V_i, the eigen vectors, of C, the data covariance matrix in the new feature space.
- We know V_i are linear combination of $\phi(X_j)$ with weights given by a_{ij}.
- Hence, once we get all $\mathbf{a_i}$, we can calculate all V_i.

- Since $\mathbf{a_i}$ are eigenvectors of K, we need to know how to normalize it. This is determined by the requirement that $V_i^T V_i = 1$.
- We normalize $\mathbf{a_i}$ using

$$1 = V_i^T V_i = \sum_{j,m=1}^{n} a_{ij} a_{im} \phi(X_j)^T \phi(X_m) = \mathbf{a_i}^T K \mathbf{a_i} = n\lambda_i \mathbf{a_i}^T \mathbf{a_i}.$$

- We cannot do this normalization for eigenvectors corresponding to zero eigen value. (But we can ignore them)

- For PCA in the new feature space we do not need V_i explicitly.
- Given an X we only need $\phi(X)^T V_i$. This is given by

$$\phi(X)^T V_i = \sum_{j=1}^{n} a_{ij}\ \phi(X)^T \phi(X_j) = \sum_{j=1}^{n} a_{ij}\ k(X, X_j).$$

- Thus we can compute the PCA in the new feature space without ever needing to evaluate $\phi(X)$.

- We can sum up the process as follows.
- From the data we form the gram matrix K.
- We find the eigen vectors, $\mathbf{a_i}$, of K and properly normalize them.
- Using these, we get the principal components corresponding to the transformed data.
- The whole process avoids ever calculating $\phi(X)$ through the kernel trick.
- However, there is one issue that we still need to take care of.

- In the analysis presented we have assumed that mean of $\phi(X_i)$ is zero which, in general, is not true.
- We cannot do a 'mean subtraction' because we do not want to compute $\phi(X_i)$.
- So, we need a trick for that too.

- Define

$$\tilde{\phi}(X_i) = \phi(X_i) - \frac{1}{n}\sum_{j=1}^{n}\phi(X_j)$$

- Let $\tilde{K}$ be the gram matrix corresponding to 'tilde' variables. We need the eigen vectors of this matrix.
- But we can calculate only K the gram matrix of original variables.
- So, we need $\tilde{K}$ in terms of K.

- We have
$$\tilde{K}_{jm} = \tilde{\phi}(X_j)^T \tilde{\phi}(X_m)$$
- By substituting for $\tilde{\phi}$ we get
$$\tilde{K}_{jm} = \left(\phi(X_j) - \frac{1}{n} \sum_{s=1}^{n} \phi(X_s) \right)^T \left(\phi(X_m) - \frac{1}{n} \sum_{s=1}^{n} \phi(X_s) \right)$$
- Algebraic simplification gives
$$\tilde{K}_{jm} = K_{jm} - \frac{1}{n} \sum_{t=1}^{n} K_{jt} - \frac{1}{n} \sum_{l=1}^{n} K_{ml} + \frac{1}{n^2} \sum_{t,l=1}^{n} K_{lt}$$
- We can write this in matrix notation as
$$\tilde{K} = K - A_n K - K A_n + A_n K A_n$$
where A_n is a $n \times n$ matrix all of whose entries are $\frac{1}{n}$.

- Thus, we can calculate $\tilde{K}$ from the data gram matrix K.
- From the eigen vectors of $\tilde{K}$ we can now find the PCA in the new feature space.
- This completes the kernel PCA algorithm.
- Kernel PCA is one possible nonlinear analogue for PCA.