Machine Learning (CS60050)
Spring 2020-2021
Instructor: Dr. Aritra Hazra

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
Scribed by: Aditya Devasthale(20CS60R26) and Sagar Gupta(20CS60R30)

18 March 2021

1 Introduction to Dimensionality Reduction

For a problem as the number of attributes(dimensions) increases, time taken to compute solution as well as memory required to store inputs and intermediate solution also increases. Sometimes searching for solution in high dimensional space becomes infeasible. Dimensionality reduction refers to techniques for reducing the number of attributes(dimensions) in training data. There are two components in dimensionality reduction:
1)**Feature selection**: finding a subset of the original set of variables, or features, to get a smaller subset which can be used to model the problem.
2)**Feature extraction**: project original data points with d dimension, to new data points with k dimensions where $k < d$.

2 Feature selection

If the feature size is initially d, the number of possible subsets are 2^d. A criteria is required to find which subset is best. One such measure is Kullback–Leibler divergence

2.1 Kullback–Leibler divergence

Kullback–Leibler divergence can be used to measure class separability between two classes. If two classes have probability distribution as p_1 and p_2 respectively, their KL divergence is given as:

$$KL(p_1, p_2) = \sum_{x \in TE} p_1(x) log \frac{p_1(x)}{p_2(x)} + p_2(x) log \frac{p_2(x)}{p_1(x)}$$

where x belongs to training data.

2.2 Feature selection methods

For selecting d features, from D available features where $d < D$, number of possible options are ${}^{D}C_{d} \approx D^{d}$. Therefore time complexity is D^{d}. To reduce time complexity following two search methods can be used:
1)**Forward search** : Start with empty set of features. Add a single feature to set that gives best KL divergence value. In each iteration a one feature is added to set based on KL divergence value till we get d features in set. Time complexity of this methods is $O(D * d)$. This method does not go over entire search space.
2)**Backward search** : Start with a set containing all features in it . Remove a single feature from set that results best KL divergence value. In each iteration a one feature is removed from set based on KL divergence value till we have d features in set. Time complexity of this methods is $O(D * d)$.

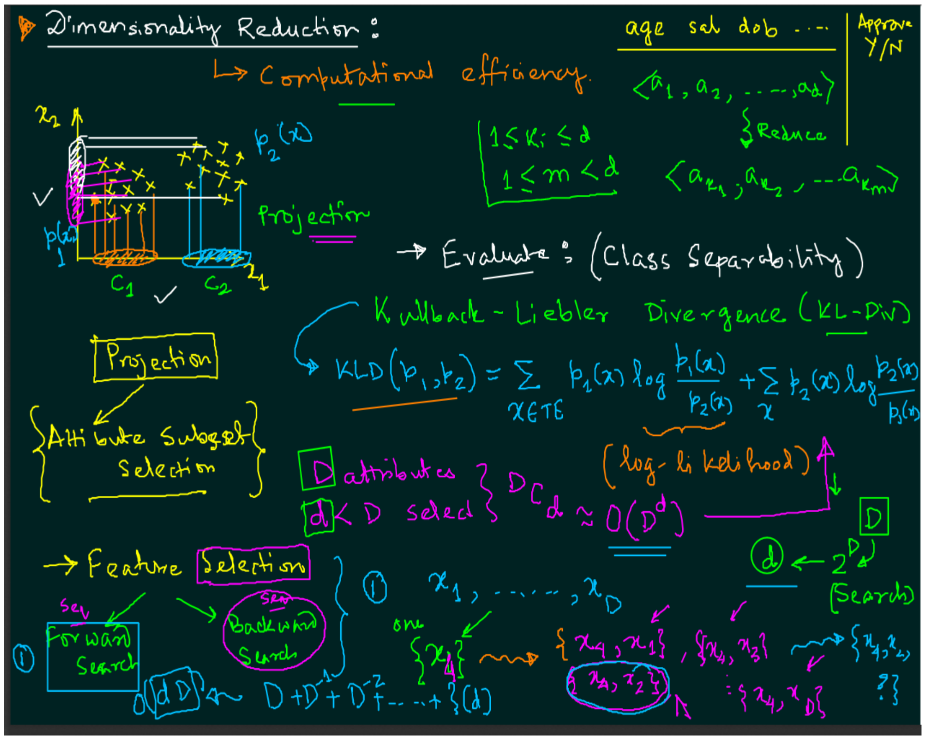


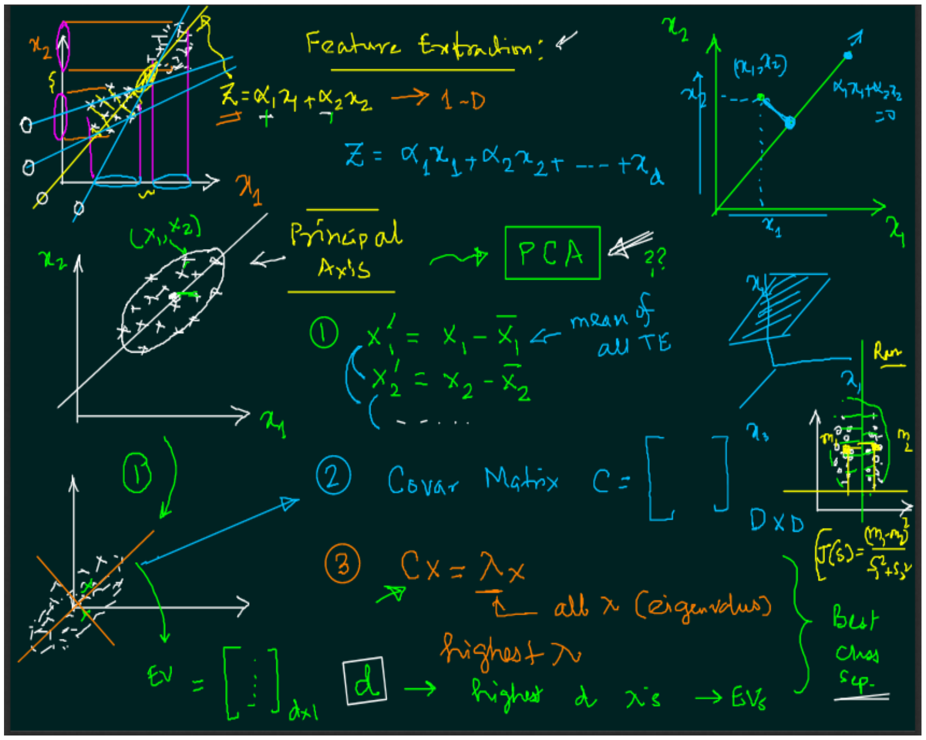
Figure 1: Reference 1

3 Feature extraction

Selecting x_1 or x_2 as subset of features is same as projecting points to x_1 and x_2 axis. Instead of projecting on x_1 or x_2 axis, points can be projected on a line.

3.1 Principal component analysis

Line representing the direction of maximum variation through the data gives us the best line for projecting data. This line is called as first principal component. For projecting in d dimesions, d principal component are found out. This called as Principal component analysis.
Steps involved in finding Principal components are:
1)Data points are centered around origin. Each feature of data point is subtracted with mean value of that feature.
2)A covariance matrix of dimension DXD (where D is the number of dimensions) is computed. It has as entries the covariances associated with all possible pairs of the initial variables.
3) Eigenvectors and eigenvalues of the covariance matrix is computed to identify the principal components. By ranking eigenvectors in order of their eigenvalues, highest to lowest, we get the principal components in order of significance. To project data to d dimensions, eigenvectors corresponding to top d eigenvalues is selected.



Feature Extraction:
$z = \alpha_1 x_1 + \alpha_2 x_2 \rightarrow 1\text{-D}$
$z = \alpha_1 x_1 + \alpha_2 x_2 + \dots + x_d$
Principal Axis
PCA ??
① $x_1' = x_1 - \bar{x}_1$ ← mean of all TE
$x_2' = x_2 - \bar{x}_2$
② Covar Matrix C = [ ] DXD
③ $Cx = \lambda x$
all λ (eigenvalues)
highest λ
EV = [ ] d×1
d → highest d λ's → EVs
Best Class Sep.
$J(S) = \frac{(m_1-m_2)^2}{S_1^2+S_2^2}$
x_1
x_2

Figure 2: Reference 2

4 Kernel Trick

Let us recall what we did in Kernel Method:

In a situation where we need to classify between points which are like :
We had a X-dimensional point.

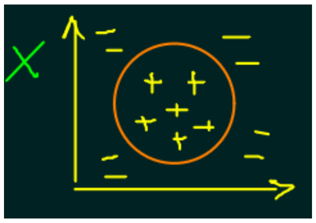


Figure 3: No linear separation possible

as we can see that there is no linear classification in possible.

so, we make a transformation and put it into a Z dimension (higher dimension)(x^2), because that transformation resulted us a linear separation.
And finally upon getting that we transformed back again to ϕ^{-1}

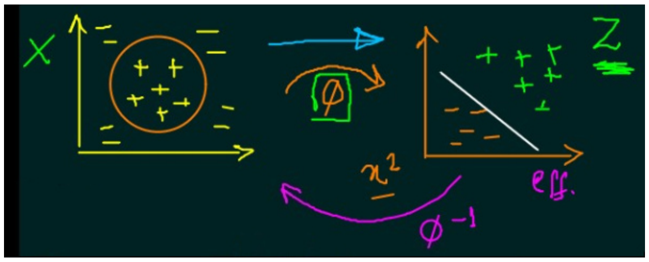


Figure 4: Transformation and returning back

This implies that not always the increase in dimension is bad.

Now we will see that how Kernel trick helps us to compute the linear separability without visiting the higher dimension space.

Let us recall the support vector machine, where we did the SVM to minimise our loss w.r.t.

$$\alpha(\alpha) = \sum_n \alpha_n - \frac{1}{2} \sum_n \sum_n y_n y_m \alpha_n \alpha_m Z_n^T Z_m$$

In support vector machine, this is computed from Hessian Matrix.(**H**)
Here, $Z^T Z$ is nothing but $K(x, x^{'})$
In our example,

$$Z = \phi(x) = (1, x_1, x_2, x_1^2, x_2^2, x_1 x_2)$$

This is complete transformation of Z from the point x where, x is two attribute point. i.e., $x = (x_1, x_2)$
Then,

$$Z^T Z = K(x, x^{'}) = (1, x_1 x_1^{'}, x_2 x^{'}{}_2, x_1^2 x^{'2}{}_1, x_2^2 x^{'2}{}_2, x_1 x^{'}{}_1 x_2 x^{'}{}_2)$$

and now, if in order to compute this if we have to visit these spaces, then it is a huge burden.
Rather, the Kernel Trick suggest that indeed we do not have to visit to all these dimensions to compute this.
We only need to know about the kernel, we do not need to know about the ϕ, in order to compute this.

Point to note is that the Kernel should be suitable because not everything represents a space.

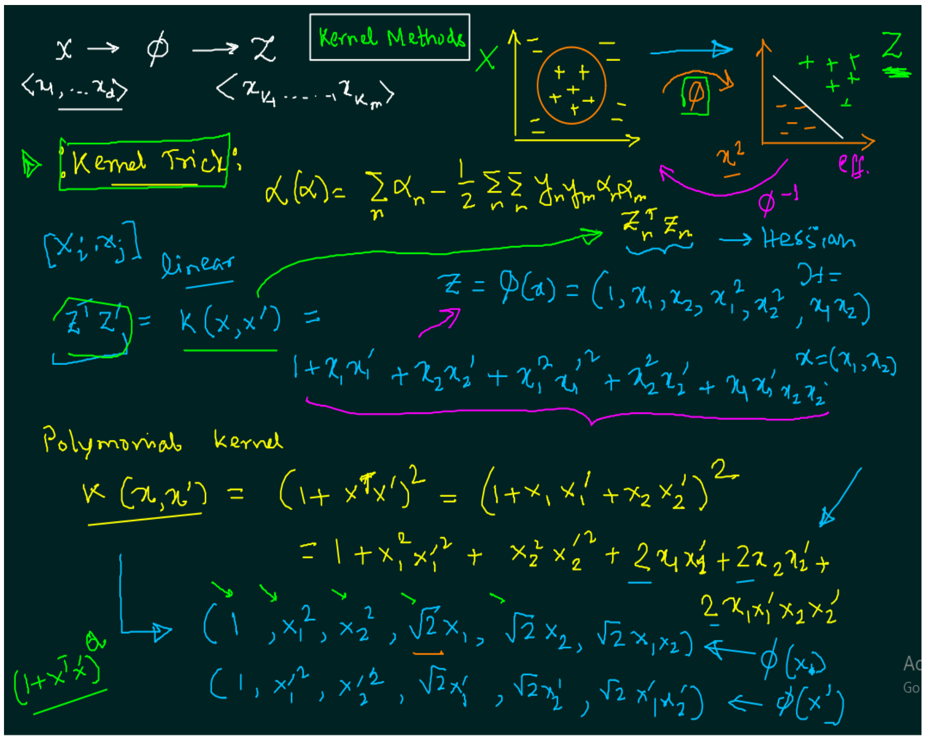


Figure 5: Reference 3

With the help of Kernel trick we can also visit to infinite dimension too, we do not have to bother where are we visiting, For example:
Let us take an infinite dimensional kernel

$$K(x, x^{'}) = e^{-\gamma||x-x^{'}||^2}$$

when we do the Taylor's Series expansion, we get,

$$K(x, x^{'}) = e^{-x^2} e^{-x^{'2}} \sum_{k=0}^{\infty} \frac{2^x x^x x^{'k}}{k!}$$

Here, also , we just need to put the kernel value into the Hessian Matrix in place of $x_1 X_2$.
Mostly, we use this trick to get discriminant through support vector machine. and from,

$$\mathbb{E}[E_{out}] \leq \frac{\mathbb{E}[No.\ of\ support\ vectors]}{(N-1)}, \text{ where N = number of points}$$

we see that only the number of support vectors matters.
The only part that is governing this kind of Kernel trick is the fact that when we transfer it to higher dimensional space we are thinking that our generalization boundary is getting too much bad but, it is indeed not because sometimes the

discriminant is only with respect to support vectors and generalization bound with respect to less number of support vectors makes our dimension to be very restricted.

But it's a very nice trick because now we are in a situation that we could visit anywhere and knock the door and come back with the result. Replace it with a new separator in there.

And indeed this kernel function is called the Radial Basis Function
And as we can see it is a Gaussian function other than exponential function where the distance between the term has a negative impact on how the dot products or the relation product will be taken.

Radial Basis Function is a very Universal function, which can be used as a model for for classification for cluster, support Vector machine, regularization etc.
It's totally a new dimensional thing which is a Radial Basis Function, which will be covered in next class in very details and we will see the ramifications of taking such kind of a kernel functions.

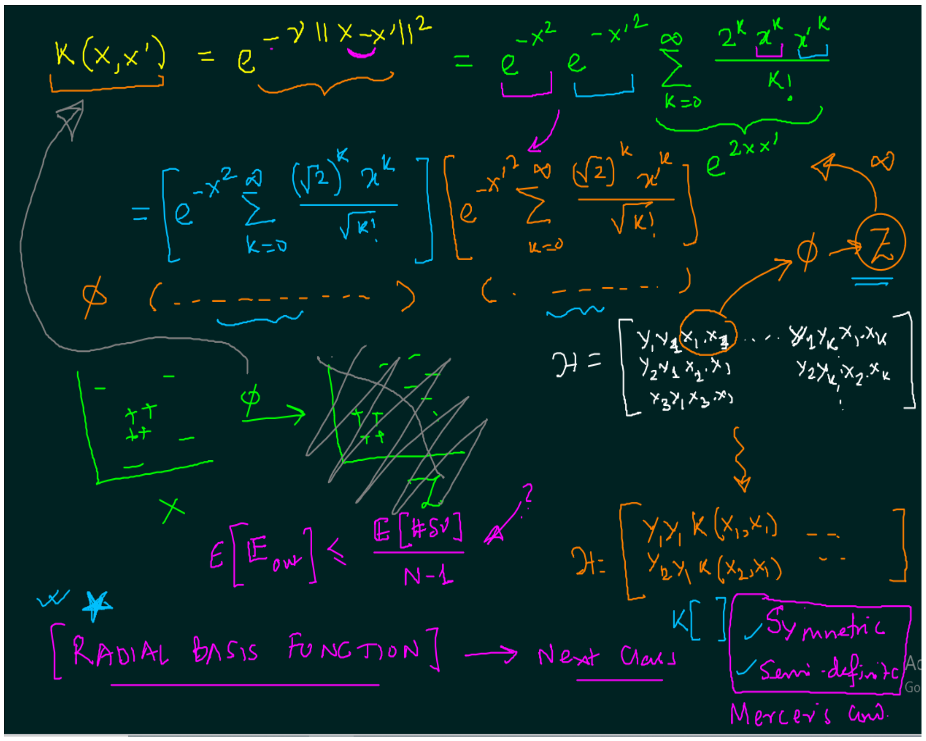

$K(x,x') = e^{-\gamma\|x-x'\|^2} = e^{-x^2} e^{-x'^2} \sum_{k=0}^{\infty} \frac{2^k x^k x'^k}{k!}$
$= \left[e^{-x^2} \sum_{k=0}^{\infty} \frac{(\sqrt{2})^k x^k}{\sqrt{k!}}\right] \left[e^{-x'^2} \sum_{k=0}^{\infty} \frac{(\sqrt{2})^k x'^k}{\sqrt{k!}}\right]$
$e^{2xx'}$
ϕ
∞
Z
$\mathcal{H} =$
$\mathcal{H} =$
$y_1 y_1 K(x_1,x_1)$
$y_2 y_1 K(x_2,x_1)$
X
$\mathbb{E}[E_{out}] \le \frac{\mathbb{E}[\#SV]}{N-1}$
[RADIAL BASIS FUNCTION] → Next Class
K[ ]
Symmetric
Semi-definite
Mercer's Cond.

Figure 6: Reference 4