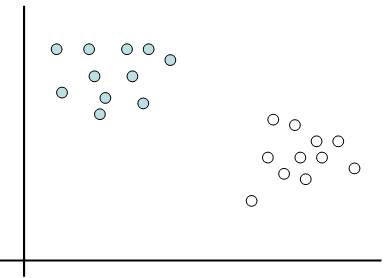
**Lecture 7 – Whiteboard notes**

## **Support Vector Machines**

* Also called as Sparse Kernel machines, Maximum margin classifiers

What is Kernel in Sparse Kernel machines?

[](https://www.google.com/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwiK9d6RipjKAhVBSCYKHQtUDCMQjRwIBw&url=https://onionesquereality.wordpress.com/2009/03/22/why-are-support-vectors-machines-called-so/&psig=AFQjCNHwK6ttOaYsUysyHs-sIq5kuZQebQ&ust=1452267683897456)

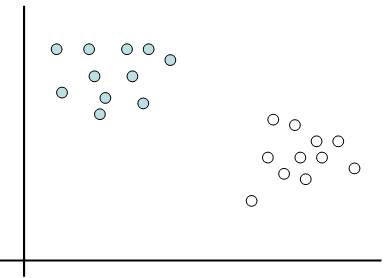
* Translating linear classification problem in terms of inner product:
* The Kernel function can be written as an inner product. We use kernel machines to predict and hence the term Kernel in Sparse Kernel machines.

Why Sparse?

* Not all training points are used to create the model.

### **Linear Support Vector Machines**

* Used for data that is linearly separable

[](https://www.google.com/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwiK9d6RipjKAhVBSCYKHQtUDCMQjRwIBw&url=https://onionesquereality.wordpress.com/2009/03/22/why-are-support-vectors-machines-called-so/&psig=AFQjCNHwK6ttOaYsUysyHs-sIq5kuZQebQ&ust=1452267683897456)

* There can be multiple lines drawn to separate the above points in training dataset. Concept of Margin is used to find the best hyperplane.

#### **Margin**

* Margin of a linear classifier is defined as the width that the boundary could be increased by before hitting a data point. Margin is highlighted in green in the below figure.

Support Vectors

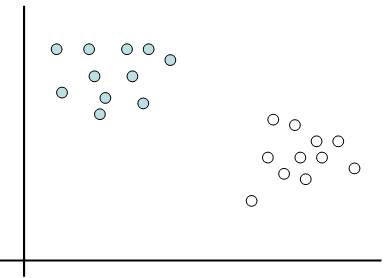
[](https://www.google.com/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwiK9d6RipjKAhVBSCYKHQtUDCMQjRwIBw&url=https://onionesquereality.wordpress.com/2009/03/22/why-are-support-vectors-machines-called-so/&psig=AFQjCNHwK6ttOaYsUysyHs-sIq5kuZQebQ&ust=1452267683897456)

Figure Maximum Margin Linear Classifier

* Points on the boundary are called support vectors.

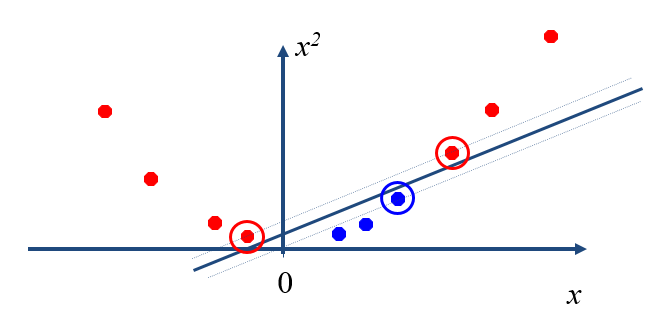
### **Non-Linear SVM**

* Changing the non-linear data points to be linearly separable by applying transformation Ф.

Input space

**C:\Users\Thavaselvi\Desktop\CSC 529\Non-linear SVM1.PNG**

Ф

****

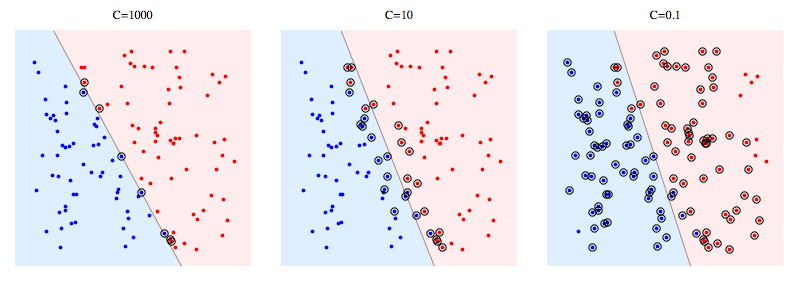
Feature space

* Linear classifier uses dot product between vectors to classify the data points.
* In a non-linear space, transformation Ф is applied to the data points.

* By applying the Kernel trick, the dimensions of the data points are increased.
* For example,

#### **Soft Margin**

* As the complexity parameter is increased, the soft margin is increased.

[](https://www.google.com/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwij-r-H-47LAhUDuoMKHexOAjIQjRwIBw&url=http://stackoverflow.com/questions/4629505/svm-hard-or-soft-margins&psig=AFQjCNEWGEhFMvFqymntbQP1M10QGopGgQ&ust=1456353818010721)

### **Multi-class SVM**

* A separate SVM is built for each class. The distance between the test data point and each of the hyperplanes is calculated.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | SVM1 (Class 1) | SVM2 (Class 2) | … | SVMm (Class m) |
| Class X | d(x, h1) | d(x, h2) | … | d(x, hm) |

* The final class for the new test point is given by:

### **Applications of SVM**

Handwritten Digit Recognition: US Postal Service (USPS handwritten digits

* + Training set: 7300, Testing set: 2000
  + Number of classes: 10
  + Image converted to attributes as shown in below figure

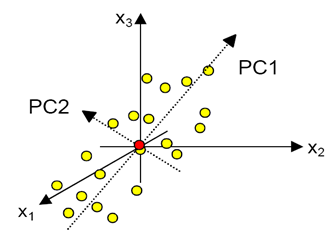
|  |  |  |
| --- | --- | --- |
|  | 16 | |
| 16 |  |  |
|  |  |
|  |  |
|  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | X1 | X2 | .. | X256 | Class |
| Instance 1 | 1 | 2 | 3 | … 256 |  |
| Instance 2 |  |  |  |  |  |
| … |  |  |  |  |  |
| Instance 7300 |  |  |  |  |  |

* 10 SVMs are built. One for each class and then assign the test instance to the class which has the highest distance from the hyperplane.

## **PRINCIPAL COMPONENT ANALYSIS (PCA)**

* Orthogonal projection of data onto lower-dimension linear space that:
  + maximizes variance of projected data
  + minimizes mean squared distance between
    - data point and
    - projections (sum of blue lines)

****

* PC1 = Eigen vector 1 ( Eigen value 1), PC2 = Eigen vector 2 ( Eigen value 2)

=

=

=

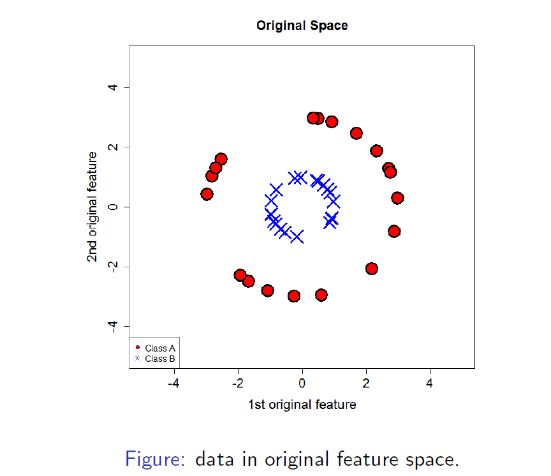
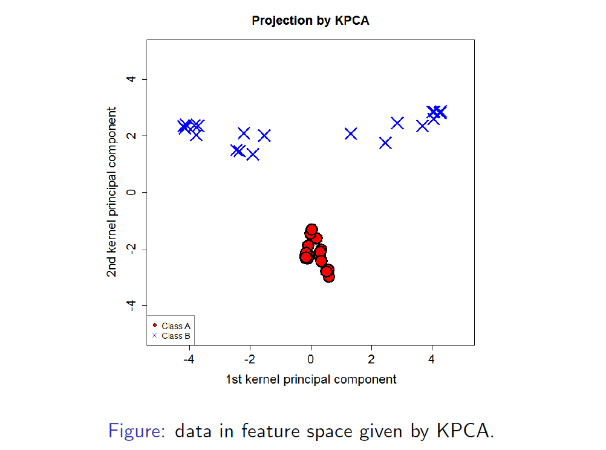
* Eigen values are computed from the co-variance matrix.
* Covariance matrix when there is no relationship between x1 and x2.
  + PCA cannot be used in this case.
* Covariance matrix when there is a relationship between x1 and x2
  + PCA can be applied in this case.
* (x1, x2) 🡪 (PC1, PC2)
* Solve 🡺 to obtain the Eigen values.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | PC1 | PC2 | .. | PCp |
| Case 1 |  |  |  |  |
| Case 2 |  |  |  |  |
| … |  |  |  |  |
| Case m |  |  |  |  |

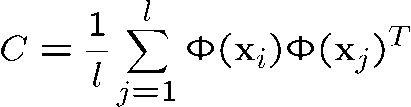
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | X1 | X2 | .. | Xk |
| Case 1 |  |  |  |  |
| Case 2 |  |  |  |  |
| … |  |  |  |  |
| Case m |  |  |  |  |

## **Kernel PCA**

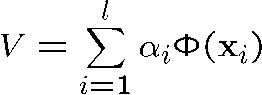
* To capture non-linear structure

* Covariance matrix:

****

* Solve for eigenvalue using CV=λV

****