Foundations of Machine Learning

Module 5: Support Vector Machine
Part E: Nonlinear SVM and Kernel function

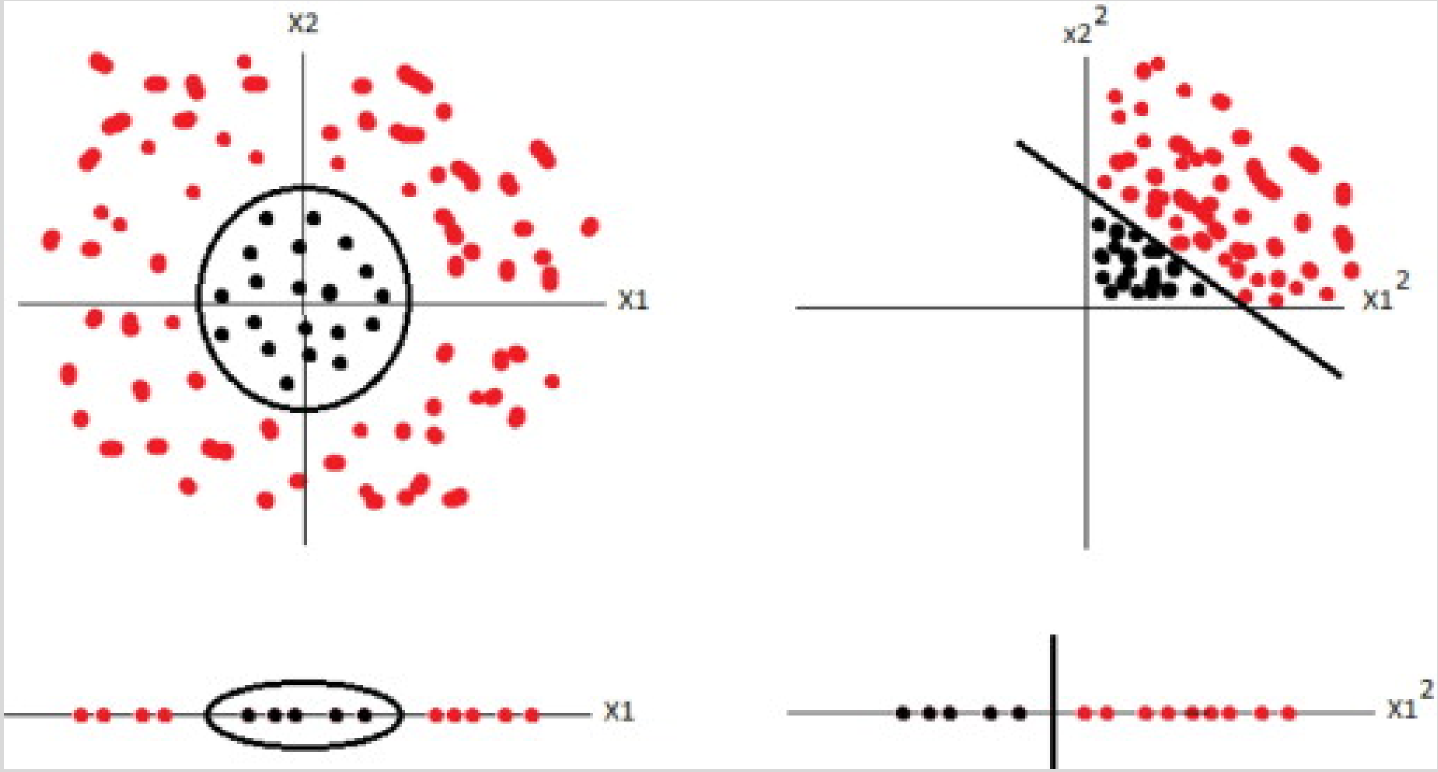
Sudeshna Sarkar
IIT Kharagpur

Non-linear decision surface

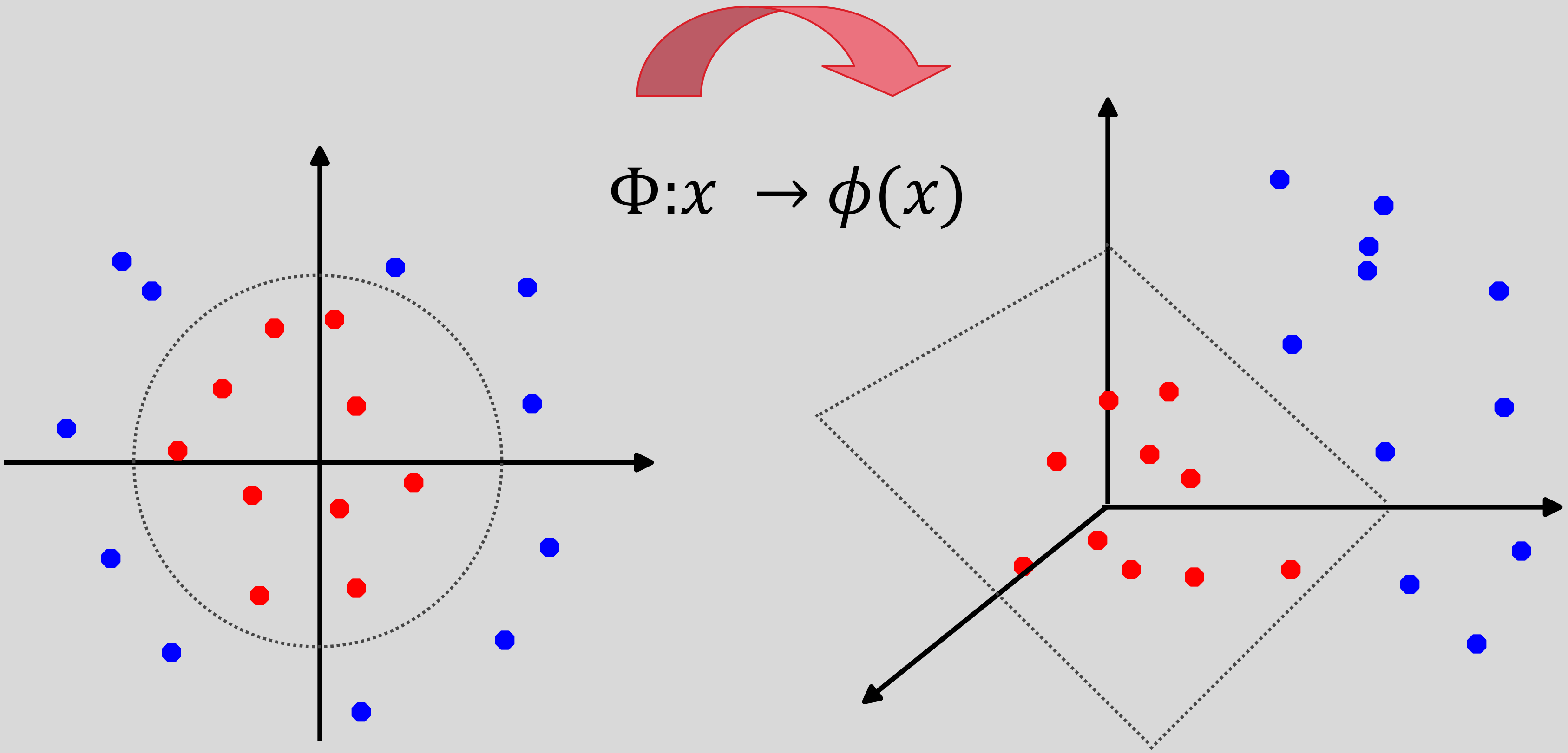
- We saw how to deal with datasets which are linearly separable with noise.
- What if the decision boundary is truly non-linear?
- Idea: Map data to a high dimensional space where it is linearly separable.
  - Using a bigger set of features will make the computation slow?
  - The “kernel” trick to make the computation fast.

Non-linear SVMs: Feature Space

$$\Phi : x \rightarrow \phi(x)$$



Non-linear SVMs: Feature Space



This slide is from *www.iro.umontreal.ca/~pift6080/documents/papers/**svm**_tutorial.**ppt***

Kernel

- Original input attributes is mapped to a new set of input features via feature mapping Φ.
- Since the algorithm can be written in terms of the scalar product, we replace $x_a . x_b$ with $\phi(x_a).\phi(x_b)$
- For certain Φ's there is a simple operation on two vectors in the low-dim space that can be used to compute the scalar product of their two images in the high-dim space

$$K(x_a, x_b) = \phi(x_a).\phi(x_b)$$

Let the kernel do the work rather than do the scalar product in the high dimensional space.

Nonlinear SVMs: The Kernel Trick

- With this mapping, our discriminant function is now:

$$g(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b = \sum_{i \in SV} \alpha_i \boxed{\phi(\mathbf{x}_i)^T \phi(\mathbf{x})} + b$$

- We only use the dot product of feature vectors in both the training and test.
- A *kernel function* is defined as a function that corresponds to a dot product of two feature vectors in some expanded feature space:

$$K(x_a, x_b) = \phi(x_a) . \phi(x_b)$$

The kernel trick

$K(x_a, x_b) = \phi(x_a).\phi(x_b)$

Often $K(x_a, x_b)$ may be very inexpensive to compute even if $\phi(x_a)$ may be extremely high dimensional.

Kernel Example

2-dimensional vectors $\bar{\boldsymbol{x}} = [x_1 x_2]$

let $\boldsymbol{K}\left(\boldsymbol{x_i}, \boldsymbol{x_j}\right) = (\mathbf{1} + \boldsymbol{x_i} . \boldsymbol{x_j})^{\mathbf{2}}$

We need to show that $K\left(x_i, x_j\right) = \phi\,(x_i) . \phi(x_j)$

$K(x_i, x_j) = (1 + x_i x_j)^2,$

$= 1 + x_{i1}^2 x_{j1}^2 + 2\, x_{i1} x_{j1}\, x_{i2} x_{j2} + x_{i2}^2 x_{j2}^2 + 2 x_{i1} x_{j1} + 2 x_{i2} x_{j2}$

$= [1 \;\; x_{i1}^2 \;\; \sqrt{2}\, x_{i1} x_{i2} \;\; x_{i2}^2 \;\; \sqrt{2} x_{i1} \;\; \sqrt{2} x_{i2}] . [1 \;\; x_{j1}^2 \;\; \sqrt{2}\, x_{j1} x_{j2} \;\; x_{j2}^2 \;\; \sqrt{2} x_{j1} \;\; \sqrt{2} x_{j2}]$

$= \phi(x_i) . \phi(x_j),$

where $\phi(x) = [1 \;\; x_1^2 \;\; \sqrt{2}\, x_1 x_2 \;\; x_2^2 \;\; \sqrt{2} x_1 \;\; \sqrt{2} x_2]$

Commonly-used kernel functions

- Linear kernel: $K\left(x_i, x_j\right) = x_i . x_j$
- Polynomial of power *p*:

$$K\left(x_i, x_j\right) = (1 + x_i . x_j)^p$$

- Gaussian (radial-basis function):

$$K\left(x_i, x_j\right) = e^{-\frac{\left\|x_i - x_j\right\|^2}{2\sigma^2}}$$

- Sigmoid

$$K\left(x_i, x_j\right) = \tanh(\beta_0 x_i . x_j + \beta_1)$$

In general, functions that satisfy *Mercer's condition* can be kernel functions.

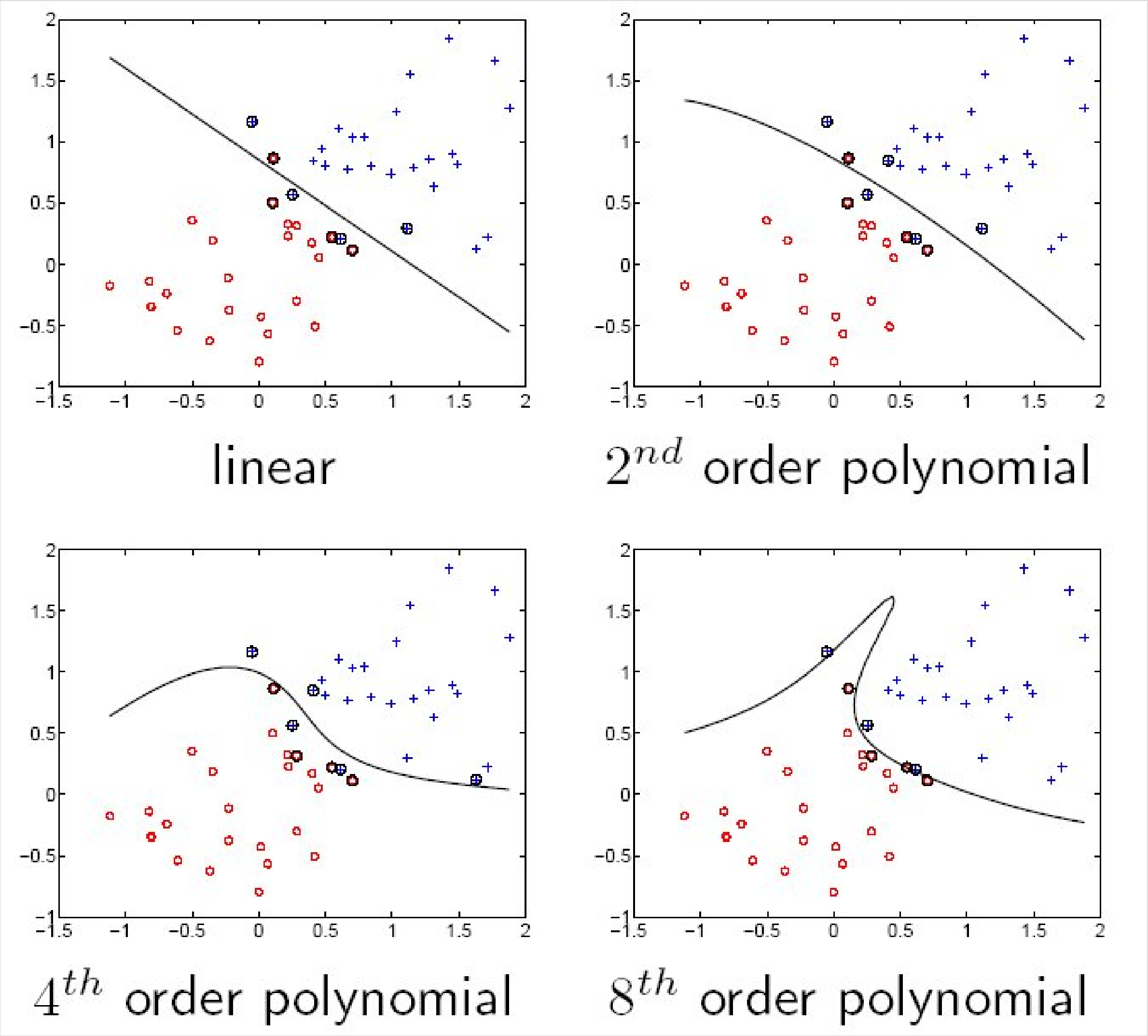
Kernel Functions

- Kernel function can be thought of as a similarity measure between the input objects
- Not all similarity measure can be used as kernel function.
- Mercer's condition states that any positive semi-definite kernel *K*(*x*, *y*), i.e.

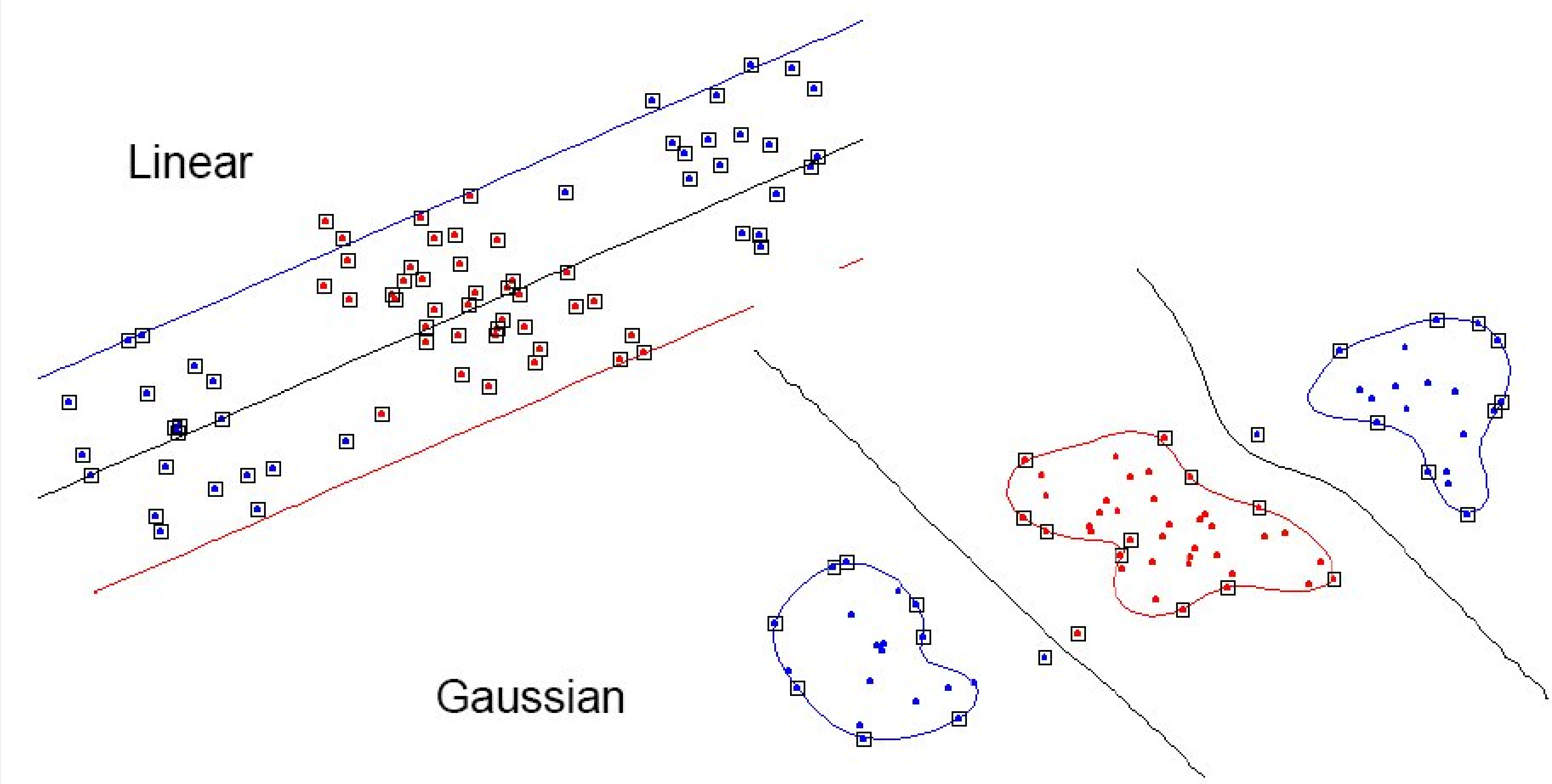
$$\sum_{i,j} K(x_i, x_j) c_i c_j \geq 0$$

- can be expressed as a dot product in a high dimensional space.

SVM examples



Examples for Non Linear SVMs –Gaussian Kernel



Nonlinear SVM: Optimization

- Formulation: (Lagrangian Dual Problem)

$$\text{maximize} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

such that

$$0 \le \alpha_i \le C$$

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

- The solution of the discriminant function is

$$g(x) = \sum_{i \epsilon SV} \alpha_i K\left(x_i, x_j\right) + b$$

Performance

- Support Vector Machines work very well in practice.
  - The user must choose the kernel function and its parameters
- They can be expensive in time and space for big datasets
  - The computation of the maximum-margin hyper-plane depends on the *square of the number of training cases.*
  - We need to store all the support vectors.
- The kernel trick can also be used to do PCA in a much higher-dimensional space, thus giving a non-linear version of PCA in the original space.

Multi-class classification

- SVMs can only handle two-class outputs
- Learn N SVM's
  - SVM 1 learns Class1 vs REST
  - SVM 2 learns Class2 vs REST
  - :
  - SVM N learns ClassN vs REST
- Then to predict the output for a new input, just predict with each SVM and find out which one puts the prediction the furthest into the positive region.

Thank You