Machine Learning

Lect 10

Dr. Susana
Mukherjee

Non linear SVM and kernel functions

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

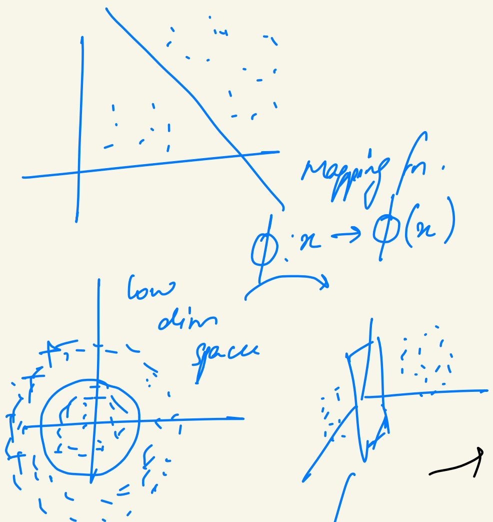
Original feature space can be mapped into new feature space where it is linearly separable → higher dimensional

But what about computational cost ??

Linear SVM : $\sum \alpha_i \alpha_j y_i y_j \underbrace{x_i}_{d\text{-dimensional}} \cdot \underbrace{x_j}_{d\text{-dimensional}}$

we need to find pairwise dot products

if we have m training examples we need to do m^2 computations like this



→ Computation time $O(d^2)$

⇓ Transform to high dimensional space $D >> d$

SVM types
- Gaussian RBF
- linear kernel
- polynomial kernel

$K(x_i, x_j)$

$\phi(x_i) = x_i \quad \phi(x_j) = x_j$

$e^{-\frac{\|x_i - x_j\|^2}{\sigma^2}}$

$(1 + x_i x_j)^{p=2}$ qua

$$\sum \alpha_i \alpha_j y_i y_j \phi(x_i) \cdot \phi(x_j)$$

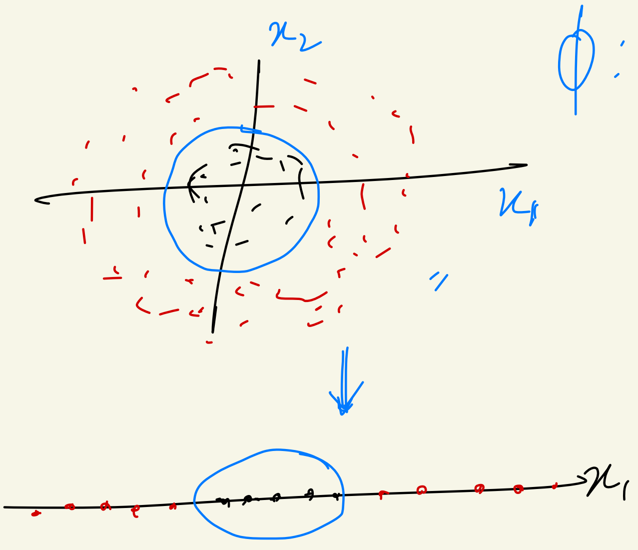
$O(D^2)$

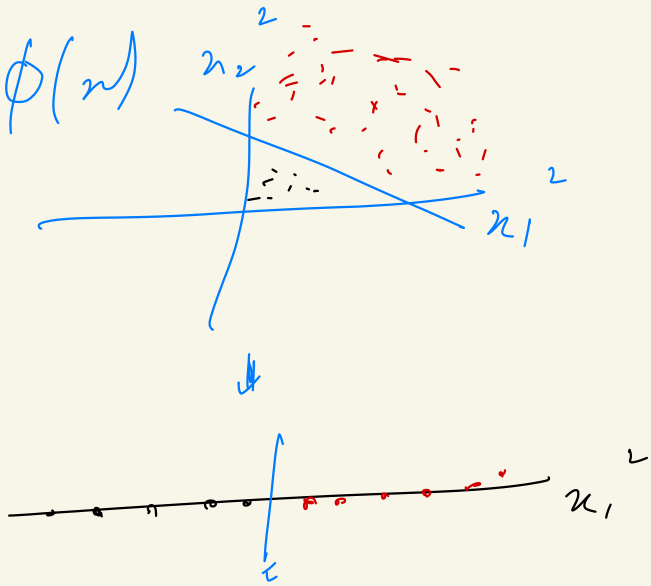
⇓ more computational cost

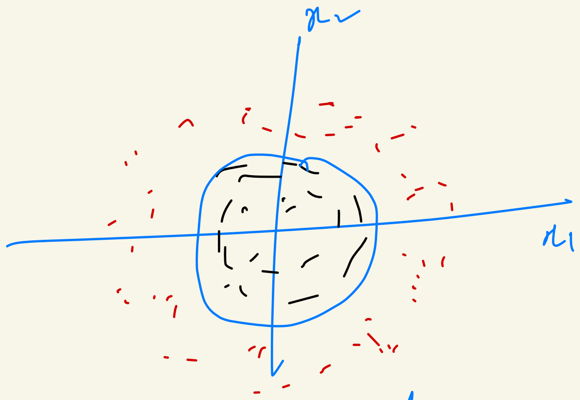
Soln: Use kernel function to reduce computational costs

Visualize Feature Space

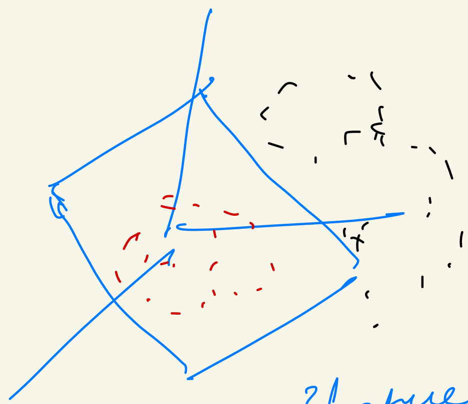
$\phi : x \to \phi(x)$







2 features



3 features
↓
linearly separable

In certain feature transformations where after transformation, SVM can be solved efficiently.

Kernel function:

x_a $\quad x_b$ $\qquad\qquad x_a \cdot x_b$

$\phi(x_a)$ $\quad \phi(x_b)$ $\qquad\qquad \phi(x_a) \cdot \phi(x_b)$

[   ].

[   ]

$$\underline{K(x_a, x_b)} = \phi(x_a) \cdot \phi(x_b)$$

↳ may be easy to compute as it is a fn. of x_a & x_b without expanding $\phi(x_a)$ or $\phi(x_b)$

also called kernel trick

What is Kernel: i) Original input attributes is mapped to a new set of input features via feature mapping ϕ

ii) Since the algo can be written in terms of the scalar product, we replace $x_a \cdot x_b$ with $\phi(x_a) \cdot \phi(x_b)$

iii) For certain ϕ; there is a simple operation on 2 vectors in the low-dim space that can be used to compute the scalar product of their 2 images in the high dim space

dot product ~~scalar~~ or vector??

$$k(x_a, x_b) = \phi(x_a) \cdot \phi(x_b)$$

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

Nonlinear SVM: The kernel Trick

→ With this mapping, our discriminant function is now:

$$g(x) = w^T \phi(x) + b$$

$$= \sum_{i \in SV} \alpha_i \boxed{\phi(x_i)^T \phi(x)} + b$$

$w^T x + b$

$w^T \phi(x) + b$

In dual form

→ We only use the dot product of feature vectors in both the training & test

→ A kernel function is defined as a function that corresponds to a dot product of two feature vectors in some expanded feature space:

$$k(x_a, x_b) = \phi(x_a) \cdot \phi(x_b)$$

This $k(x_a, x_b)$ may be very inexpensive to compute even if $\phi(x_a)$ may be extremely high dimensional

x_i $[x_{i1} \ x_{i2}]$ x_j $[x_{j1} \ x_{j2}]$

<u>Kernel Example</u>

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

Let $k(x_i, x_j) = (1 + x_i \cdot x_j)^2$

$$k(x_i, x_j) = (1 + x_i \cdot x_j)^2$$

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

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

$= \phi(x_i) \cdot \phi(x_j)$

where $\phi(x) = [1 \quad x_1^2 \quad \sqrt{2}x_1x_2 \quad x_2^2 \quad \sqrt{2}x_1 \quad \sqrt{2}x_2]$

$(1 + x_i \cdot x_j)^2$

$[x_{i1} \quad x_{i2}]$

$[x_{j1} \quad x_{j2}]$

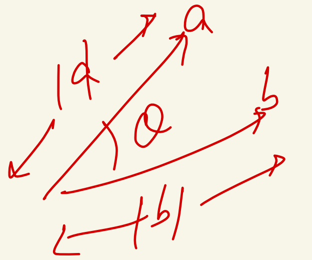
$$[1 + x_{i1}x_{j1} + x_{i2}x_{j2}]^2$$

$$(a+b+c)^2 = a^2 + b^2 + c^2 + 2ab + 2bc + 2ca$$

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

Dot product

$$a \cdot b = |a|\,|b| \cos\theta$$

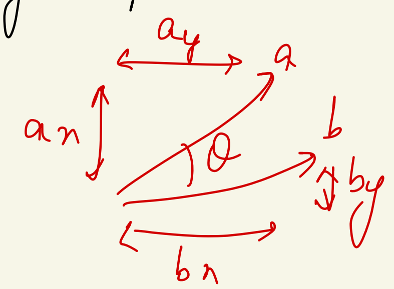


or

$$a \cdot b = a_x \times b_x + a_y \times b_y$$ then y-components

multiply the x components of 2 vectors

Scalar quantity



$x_i = [x_{i1} \quad x_{i2}]$

$x_j = [x_{j1} \quad x_{j2}]$

$x_i \cdot x_j = x_{i1} \times x_{j1} + x_{i2} \times x_{j2}$

$(1 + x_i \cdot x_j)^2 = (1 + x_{i1}x_{j1} + x_{i2}x_{j2})^2 \Rightarrow (a+b+c)^2$

Commonly used kernel functions

→ Linear SVM corresponds to linear kernel:

$$k(x_i, x_j) = x_i \cdot x_j \quad \text{where } \phi(x_i) = x_i \ \& \ \phi(x_j) = x_j$$

→ Polynomial of power p:

$$k(x_i, x_j) = (1 + x_i \cdot x_j)^p$$

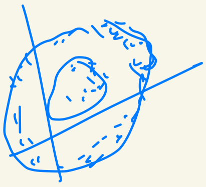
Identify mapping

→ Gaussian (radial basis function):

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

→ Sigmoid:

$$K(x_i, x_j) = \tanh(\beta_0 x_i \cdot x_j + \beta_1)$$



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

Eg: Text classification we can use kernels based on similarity of words of text

Kernel Functions

→ kernel function can be thought of as a similarity measure b/w the input objects

→ Not all similarity measure can be used as kernel function

→ Mercer's condition states that any positive semi-definite kernel $k(x, y)$ ie

$$\sum k(x_i, x_j) \underbrace{c_i c_j}_{\text{any real no.}} \geq 0$$

positive semi-definite matrix

Generally this similarity measure is symmetric

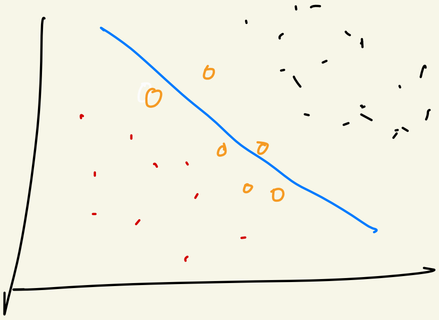
Any matrix A is symmetric & all the eigen values are +ve

$$k(x_i, x_j) = k(x_j, x_i)$$

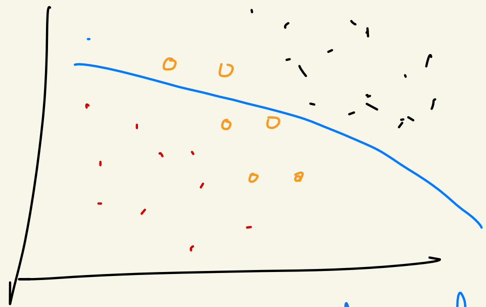
If you can write similarity b/w points as a matrix & this matrix is symmetric +ve semi definite then a kernel function will exist

→ Such functions can be expressed as a dot product in high dimensional space

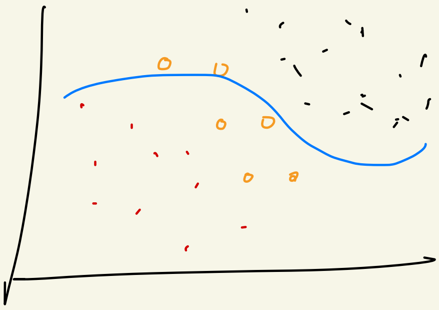
SVM examples



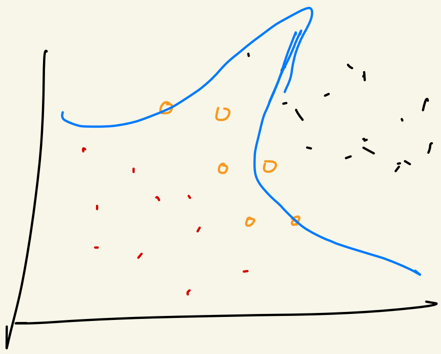
linear kernel with noise



2nd order polynomial



4th order polynomial



8th order polynomial

Non Linear SVM optimization Formulation (Langragian Dual Problem)

$\phi(x_i) \cdot \phi(x_j)$
$\Uparrow$
$x_i \cdot x_j$

maximize $\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \, k(x_i, x_j)$

s.t. $0 \le \alpha_i \le C \rightarrow$ hyperparameter

$C = 0.001$

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

The solution of the discriminant function is any test point x

$$g(x) = \sum_{i \in SV} \alpha_i \, k(x_i, x_j) + b$$

$\hookrightarrow \phi(x_i) \cdot x$

$\phi(x_i)^T \cdot \phi(x)$

Performance:

- SVM work very well in practice
  User must choose the kernel function & its parameters (C) appropiately
- They can be expensive in time & space for big datasets
- Computation of the maximum margin hyperplane depends on the square of number of training cases

$O(d^k)$

n^2

- We need to store all 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

Multiclass classification

SVMs can only handle 2-class outputs

Learn N-SVMs

- SVM 1 learns Class 1 vs Rest
- SVM 2 learns Class 2 vs Rest

.

SVM N learns Class N vs Rest

Then to predict o/p for a new input, just predict with each SVM & find out which one put the prediction the furthest into the positive region

majority voting

Class 1 Class 2 Class 3 . . . Class N

OVO
one vs one

OVA
one vs all

- SVM 1 Class 1 vs Rest
  Class 2 vs Rest
-
-

Class 1 — negative

sk-learn

svm predict
svm.classify

python
matlab → lib-svm library