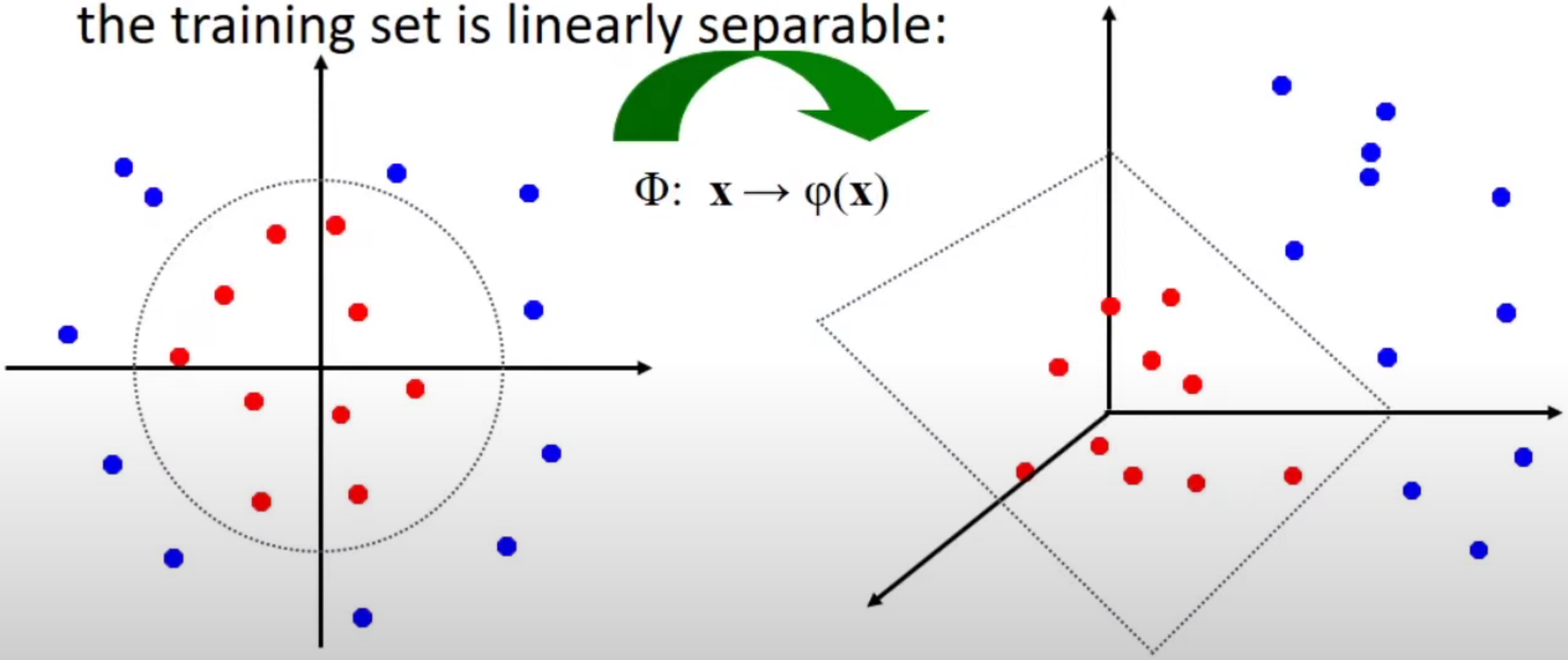
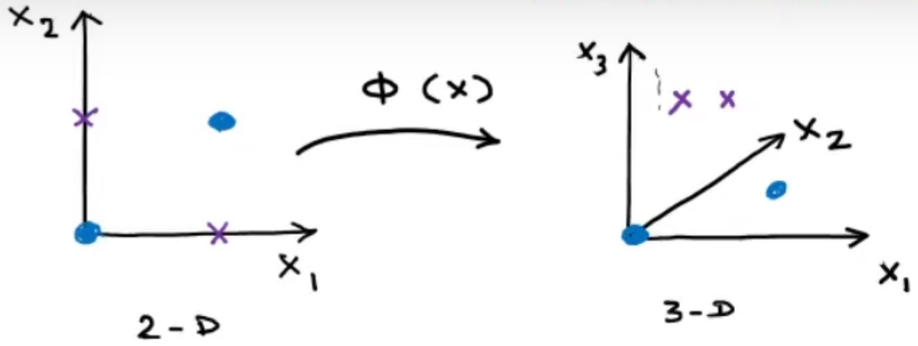
Making SVM capable of solving nonlinear classification task

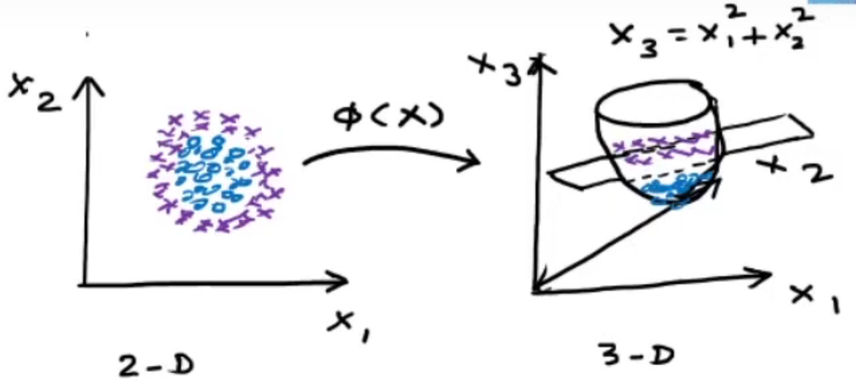
- Idea: The original nonlinear input feature space can always be mapped to some higher-dimensional feature space where the training set is linearly separable:



$$L = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y^{(i)} y^{(j)} \langle \phi(\vec{s}^{(i)}) \cdot \phi(\vec{s}^{(j)}) \rangle$$

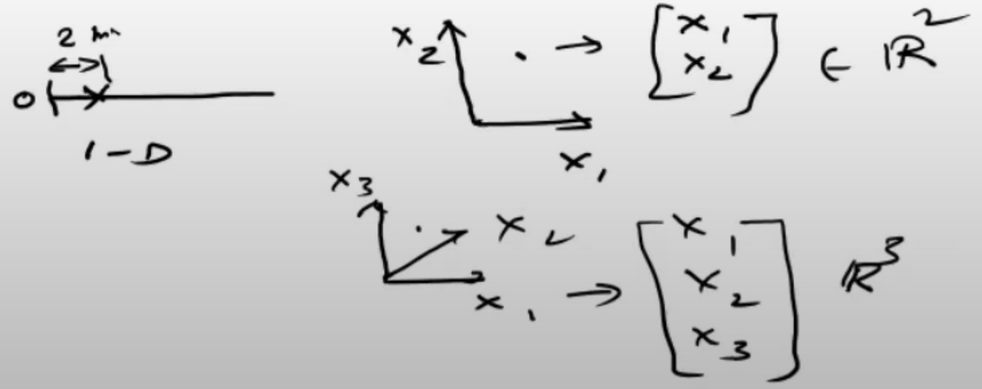
Kernel method in SVM





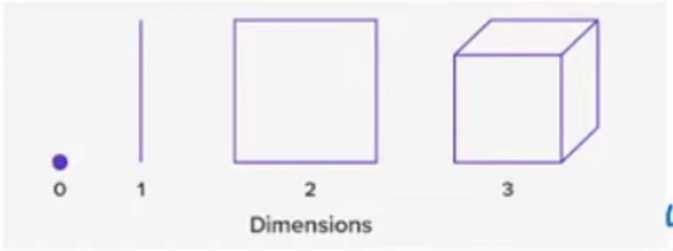
Physical Dimension.

Representational dimension.



Making SVM capable of solving nonlinear classification task

- Concept of Physical and Representational dimensions
- Concept of higher dimensional feature space
- Transformation function $\phi : x \rightarrow \phi(x)$;



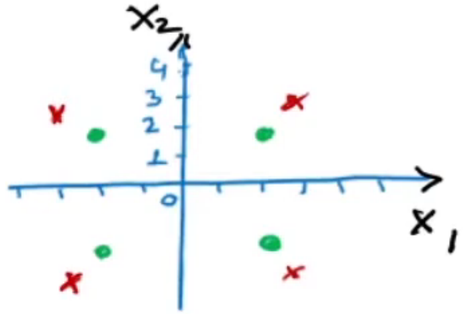
Say we have samples :

$$\bar{s}_+^{(i)} = \left\{ \begin{pmatrix} 3 \\ 3 \end{pmatrix}, \begin{pmatrix} 3 \\ -3 \end{pmatrix}, \begin{pmatrix} -3 \\ -3 \end{pmatrix}, \begin{pmatrix} -3 \\ 3 \end{pmatrix} \right\} \rightarrow Y^{(i)} = 1$$

$$\bar{s}_-^{(i)} = \left\{ \begin{pmatrix} 2 \\ 2 \end{pmatrix}, \begin{pmatrix} 2 \\ -2 \end{pmatrix}, \begin{pmatrix} -2 \\ -2 \end{pmatrix}, \begin{pmatrix} -2 \\ 2 \end{pmatrix} \right\}$$

$$\downarrow$$

$$Y^{(i)} = -1 .$$



$$Y^{(i)} (\bar{w} \cdot \bar{s}^{(i)} + b) = 1$$

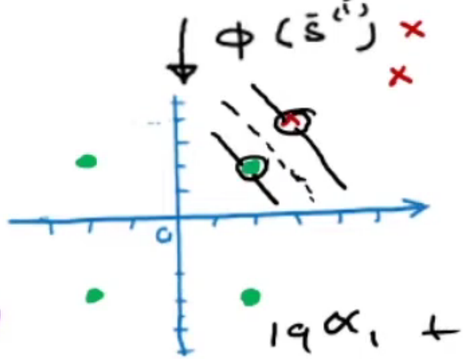
$$\sum_{i=1}^{2} \alpha_i Y^{(i)} (\tilde{w} \tilde{s}^{(i)}) = 1$$

$$\downarrow$$

Let us define a nonlinear mapping function $\phi(\bar{s}^{(i)}) = \phi\begin{pmatrix} \bar{s}^{(1)} \\ \bar{s}^{(2)} \end{pmatrix}$

$64 \times 64 \times 3 = 12288$

$$\begin{cases} 6 - \bar{s}^{(2)} + |\bar{s}^{(1)} - \bar{s}^{(2)}| & \text{if } \sqrt{\bar{s}^{(1)2} + \bar{s}^{(2)2}} > 3 \\ 6 - \bar{s}^{(1)} + |\bar{s}^{(1)} - \bar{s}^{(2)}| \\ \bar{s}^{(1)} \\ \bar{s}^{(2)} \end{cases} \quad \text{otherwise .}$$



$$\alpha_1 \begin{pmatrix} 3 \\ 3 \\ 1 \end{pmatrix} \begin{pmatrix} 3 \\ 3 \\ 1 \end{pmatrix} + \alpha_2 \begin{pmatrix} 3 \\ 3 \\ 1 \end{pmatrix} \begin{pmatrix} 2 \\ 2 \\ 1 \end{pmatrix} = 1$$

$$\alpha_1 \begin{pmatrix} 3 \\ 3 \\ 1 \end{pmatrix} \begin{pmatrix} 2 \\ 2 \\ 1 \end{pmatrix} + \alpha_2 \begin{pmatrix} 2 \\ 2 \\ 1 \end{pmatrix} \begin{pmatrix} 2 \\ 2 \\ 1 \end{pmatrix} = -1$$

After transformation, sample-1 $\rightarrow \begin{matrix} 6-3+0 = 3 \\ 6-3+3 = 3 \end{matrix}$

sample-2 $\rightarrow \begin{matrix} 6-3+6 = 12 \\ 6-3+6 = 12 \end{matrix} \rightarrow Y^{(i)} = 1$

sample-3 $\rightarrow \begin{pmatrix} 15 \\ 15 \end{pmatrix}$

sample-4 $\rightarrow \begin{pmatrix} 9 \\ 15 \end{pmatrix}$

$$19\alpha_1 + 13\alpha_2 = -1$$
$$13\alpha_1 + 9\alpha_2 = 1$$

Solving $\rightarrow \alpha_1 = -8, \alpha_2 = 11.67$

$1000 \times 1000 \times 3 = 3M$

Sample-1 $\rightarrow \begin{pmatrix} 2 \\ 2 \end{pmatrix}$, sample-2 $\rightarrow \begin{pmatrix} 2 \\ -2 \end{pmatrix}$, sample-3 $\rightarrow \begin{pmatrix} -2 \\ -2 \end{pmatrix} \rightarrow Y^{(i)} = -1$

sample-4 $\rightarrow \begin{pmatrix} -2 \\ 2 \end{pmatrix}$

$$\tilde{w} = \sum_{i=1}^{2} \alpha_i \tilde{s}^{(i)} Y^{(i)}$$
$$= -8 \begin{pmatrix} 3 \\ 3 \\ 1 \end{pmatrix} + 11.67 \begin{pmatrix} 2 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} -0.66 \\ -0.66 \\ 3.67 \end{pmatrix}$$

$$Y = w_1 x_1 + w_2 x_2 + b$$
$$= -0.66 x_1 - 0.66 x_2 + 3.67$$

Making SVM capable of solving nonlinear classification task

Kernel Trick

- In general it is difficult to design data specific mapping function ϕ
- So we apply Kernel trick -

We write $K(x,z) = \phi(x) \cdot \phi(z)$ ——— (1)

Now let $\phi(x) = \phi\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_{2\times 1} = \begin{bmatrix} x_1x_1 \\ x_1x_2 \\ x_2x_1 \\ x_2x_2 \end{bmatrix}_{4\times 1}$ &

$$\phi(z) = \phi\begin{bmatrix} z_1 \\ z_2 \end{bmatrix}_{2\times 1} = \begin{bmatrix} z_1z_1 \\ z_1z_2 \\ z_2z_1 \\ z_2z_2 \end{bmatrix}_{4\times 1}$$

Then from (1) $\rightarrow$ $K(x,z) = \begin{bmatrix} x_1^2 \\ x_1x_2 \\ x_2x_1 \\ x_2^2 \end{bmatrix} \cdot \begin{bmatrix} z_1^2 \\ z_1z_2 \\ z_2z_1 \\ z_2^2 \end{bmatrix} = (x_1z_1)^2 + x_1x_2z_1z_2 + x_1x_2z_1z_2 + (x_2z_2)^2$

$= (x_1z_1 + x_2z_2)^2$

$= (x \cdot z)^2$

We call K is a quadratic kernel.

Kernel Trick

Take another example :

$$K(x, z) = \phi(x) \cdot \phi(z)$$

Let $\phi(x) = \phi\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} x_1^3 \\ \sqrt{3} \cdot x_1 x_2^2 \\ \sqrt{3} \cdot x_1^2 x_2 \\ x_3^3 \end{bmatrix}$

$$\phi(z) = \phi\begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} z_1^3 \\ \sqrt{3} \cdot z_1 z_2^2 \\ \sqrt{3} \cdot z_2 z_1^2 \\ z_3^3 \end{bmatrix}$$

Calculate

$$\Rightarrow \phi(x) \cdot \phi(z)$$

$$\Rightarrow (x_1 z_1 + x_2 z_2)^3$$

Different types of Kernels:

1. Polynomial kernel $K(x, z) = (x \cdot z + 1)^d$, where d is the degree of polynomial.
2. Gaussian kernel, $K(x, z) = \exp\left(-\frac{\|x - z\|^2}{2\sigma^2}\right)$
3. Gaussian Radial Basis function (RBF), $K(x, z) = \exp\left(-\gamma \|x - z\|^2\right)$
4. Laplace RBF kernel, $K(x, z) = \exp\left(-\frac{\|x - z\|}{\sigma}\right)$
5. Hyperbolic tangent kernel, $K(x, z) = \tanh\left(\beta_0 x \cdot z + \beta_1\right)$.

Kernel Trick

SVM= Optimal Classifier + Kernel Trick

$\bar{s}^{(i)} \to x^{(i)} \to x$; $\bar{s}^{(j)} \to x^{(j)} \to z$

$x \xrightarrow{\Phi} \Phi(x)$; $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \in \mathbb{R}^{n=2} \to \Phi(x) = \begin{bmatrix} x_1x_1 \\ x_1x_2 \\ x_2x_1 \\ x_2x_2 \end{bmatrix} \in \mathbb{R}^{n=4}$

$O(n)$ $\quad O(n^2)$ $\quad 4\times 1$

Considering all pairs of monomial terms.

$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \in \mathbb{R}^{n=3} \to \Phi(x) = \begin{bmatrix} x_1x_1 \\ x_1x_2 \\ x_1x_3 \\ x_2x_1 \\ x_2x_2 \\ x_2x_3 \\ x_3x_1 \\ x_3x_2 \\ x_3x_3 \end{bmatrix} \in \mathbb{R}^{n=9}$

Similarly $\Phi(z) = \begin{bmatrix} z_1z_1 \\ z_1z_2 \\ z_1z_3 \\ z_2z_1 \\ z_2z_2 \\ z_2z_3 \\ z_3z_1 \\ z_3z_2 \\ z_3z_3 \end{bmatrix} \in \mathbb{R}^{n=9}$

Because there are n^2 elements, you need

$O(n^2)$ time to compute $\Phi(x)$ or

$\Phi(x)^T \Phi(z)$

How kernel can reduce the complexity?

Let, $K(x,z) = \Phi(x)^T \Phi(z) = (x^T z)^2$

$$(x^T z)^2 = \underbrace{\left(\sum_{i=1}^{n} x_i z_i\right)}_{x^T z} \underbrace{\left(\sum_{j=1}^{n} x_j z_j\right)}_{x^T z} = \sum_{i=1}^{n}\sum_{j=1}^{n} x_i z_i x_j z_j$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n} (x_i \cdot x_j)(z_i \cdot z_j) \to O(n)$$

Kernel Trick

IF we choose $\phi(x)^T \cdot \phi(z) = K(x,z) = (x^T z + c)^d$; For $d = 2$

$\phi(x) = \begin{bmatrix} x_1x_1 \\ x_1x_2 \\ x_1x_3 \\ x_2x_1 \\ x_2x_2 \\ x_2x_3 \\ x_3x_1 \\ x_3x_2 \\ x_3x_3 \\ \sqrt{2c}\,x_1 \\ \sqrt{2c}\,x_2 \\ \sqrt{2c}\,x_3 \\ c \end{bmatrix}$; $\phi(z) = \begin{bmatrix} z_1z_1 \\ z_1z_2 \\ z_1z_3 \\ z_2z_1 \\ z_2z_2 \\ z_2z_3 \\ z_3z_1 \\ z_3z_2 \\ z_3z_3 \\ \sqrt{2c}\,z_1 \\ \sqrt{2c}\,z_2 \\ \sqrt{2c}\,z_3 \\ c \end{bmatrix}$

$\phi(x) = \begin{bmatrix} x_1x_1 \\ x_1x_2 \\ x_1x_3 \\ x_2x_1 \\ x_2x_2 \\ x_2x_3 \\ x_3x_1 \\ x_3x_2 \\ x_3x_3 \\ \vdots \end{bmatrix} = \begin{bmatrix} n+d \\ d \end{bmatrix}$ $\phi(z) = \begin{bmatrix} z_1z_1 \\ z_1z_2 \\ z_1z_3 \\ z_2z_1 \\ z_2z_2 \\ z_2z_3 \\ z_3z_1 \\ z_3z_2 \\ z_3z_3 \\ \vdots \end{bmatrix}$;

(Can have all $\binom{n+d}{d}$ features of monomial terms upto the order of d. Roughly it can have $(n+d)^d$ features. It is large.

But with our quadratic kernel, the computation is still $O(n)$

Kernel Trick

How to make Kernels?

From Intuition if x and z are similar then

$k(x,z) = \phi(x)^T \phi(z)$ is large

If x and z are dissimilar then k(x,z) is small

Gram matrix

$$K = \begin{bmatrix} \phi(x_1)^T\phi(x_1) & \phi(x_1)^T\phi(x_2) & \cdots & \phi(x_1)^T\phi(x_n) \\ \phi(x_2)^T\phi(x_1) & \phi(x_2)^T\phi(x_2) & \cdots & \phi(x_2)^T\phi(x_n) \\ \vdots & \vdots & & \\ \phi(x_m)^T\phi(x_1) & \phi(x_m)^T\phi(x_2) & \cdots & \phi(x_m)^T\phi(x_n) \end{bmatrix}$$

$$= \begin{bmatrix} k(x_1,x_1) & k(x_1,x_2) & \cdots & k(x_1,x_n) \\ k(x_2,x_1) & k(x_2,x_2) & \cdots & k(x_2,x_n) \\ \vdots & & & \\ k(x_m,x_1) & k(x_m,x_2) & \cdots & k(x_m,x_n) \end{bmatrix}$$

$K \geq 0 \rightarrow$ Symmetric

SVM as linear machine:

$$\max_{\alpha} W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle$$

$\langle x, z \rangle$

SVM as nonlinear machine

$$\max_{\alpha} W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle \phi(x^{(i)}), \phi(x^{(j)}) \rangle$$

$\phi(x)^T \phi(z)$

SVM as nonlinear classifier with kernel trick

$$\max_{\alpha} W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j$$

$K(x,z)$

How to design your own Kernel?

All Kernels more or less satisfy the small and large concept.

For example 1. Gaussian Kernel → $K(x,z) = \exp\left(-\frac{\|x-z\|^2}{2\sigma^2}\right)$

If x & z are similar, then $K(x,z) \to 1$

If x & z are dissimilar, then $K(x,z) \to 0$

2. Radial basis Function kernel → $K(x,z) = \exp\left(-\gamma \|x-z\|^2\right)$

Same conditions are obeyed.

All kernels must obey Mercer's theorem which states that :

For K to be a valid kernel function, there must exists Φ such that $K(x,z) = \phi(x)^T \phi(z)$ and the corresponding kernel matrix K must be symmetric and positive semidefinite i.e. $K \geq 0$, being constituted from a finite set of m points and having dimension $m \times m$ with its (i,j) entry is given by $K_{ij} = K(x^{(i)}, x^{(j)})$.

How to make your own kernel

Proof :

Let K is a valid Kernel corresponding to some feature mapping Φ.

Now consider some finite set of m points $\{x^{(1)}, x^{(2)}, \ldots x^{(m)}\}$ and let a square matrix $K_{m \times m}$ be defined such that its (i, j) – entry is given by $K_{ij} = K(x^{(i)}, x^{(j)})$. This matrix is Kernel matrix.

Now if K is a valid Kernel then $K_{ij} = K(x^{(i)}, x^{(j)}) = \Phi(x^{(i)})^T \Phi(x^{(j)}) =$

$$\Phi(x^{(j)})^T \Phi(x^{(i)}) = K(x^{(j)}, x^{(i)})$$

Also letting $\Phi_k(x)$ to denote the k-th coordinate of the vector $\Phi(x)$, then for any arbitrary vector z, we can write :

$$z^T K z = \sum_i \sum_j z_i K_{ij} z_j \qquad \text{(By the definition of matrix multiplication)}$$

$$= \sum_i \sum_j z_i \Phi(x^{(i)})^T \Phi(x^{(j)}) z_j$$

$$= \sum_i \sum_j z_i \sum_k \Phi_k(x^{(i)}) \Phi_k(x^{(j)}) z_j \qquad \because a^T b = \sum_k a_k b_k$$

$$= \sum_k \sum_i \sum_j z_i \Phi_k(x^{(i)}) \cdot \Phi_k(x^{(j)}) z_j = \sum_k \left(\sum z_i \Phi_k(x^{(i)}) \right)^2 \geq 0$$

$\Rightarrow$ Hence 'K' is positive semi-definite i.e all its eigenvalues ≥ 0

Generalised Kernel method

Kernel method is very general and can be applied to all the discriminative models you have learnt so far like:

- **Linear Regression**
- **Logistic Regression**
- **Perceptron learning**

Kernel based methods can also be applied to generative models like **PCA & LDA**

Properties of SVM

- Out of a large data set normally there are only a handful of support vectors and only support vectors are used to specify the separating hyper plane.
- Capable of handling large feature spaces:
  - computational complexity does not depend on the dimensionality of the feature space.
- Some flexibility of choosing kernel functions.
- Overfitting can be controlled by soft margin approach
- Has a nice mathematical foundation: The classification problem has been formulated as a constrained quadratic optimization problem, guaranteeing that the training parameters are tuned using global optimal values.

Limitations

- **It is sensitive to noise**

  - A relatively small number of mislabeled examples can dramatically decrease the performance

- **It only considers two classes**

  - how to do multi-class classification with SVM?

1) with output m, learn m SVM's

- **SVM 1 learns "Output==1" vs "Output != 1"**
- **SVM 2 learns "Output==2" vs "Output != 2"**
- **:**
- **SVM m learns "Output==m" vs "Output != m"**

2)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.

SVM Applications

- SVM has been used successfully in many real-world problems:
  - Text (and hypertext) categorization.
  - Image classification.
  - Bioinformatics (Protein classification, Cancer classification, etc.).
  - Hand-written character recognition.

Comparison of SVM with other classifiers

S. No.	Name of the Classifier	Test error rate (%)
1.	K - Nearest Neighbours (well formulated)	3 - 5
2.	SVM (Gaussian Kernel / 4 degree polynomial)	0.5 -1.5
3.	NN (2 - layers, 300 Hidden units)	3 - 4.5
	6 - layers running on GPU	1
4.	CNN augmented with SVM	0.5