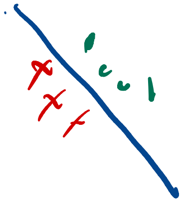
Lecture 20: 17 June, 2021

Madhavan Mukund

https://www.cmi.ac.in/~madhavan

Data Mining and Machine Learning
April–July 2021

Perceptrons
SVM

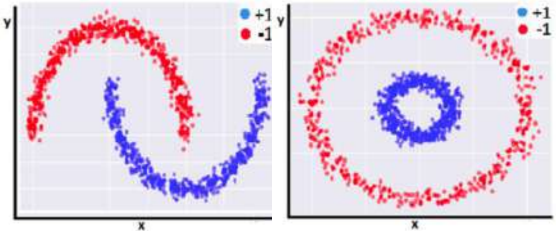


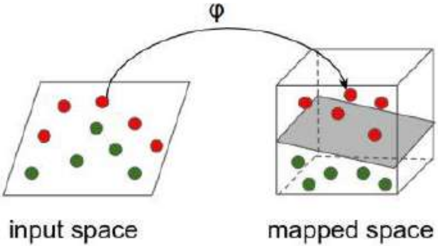
The non-linear case

- How do we deal with datasets where the separator is a complex shape?

- Geometrically transform the data
  - Typically, add dimensions

- For instance, if we can "lift" one class, we can find a planar separator between levels



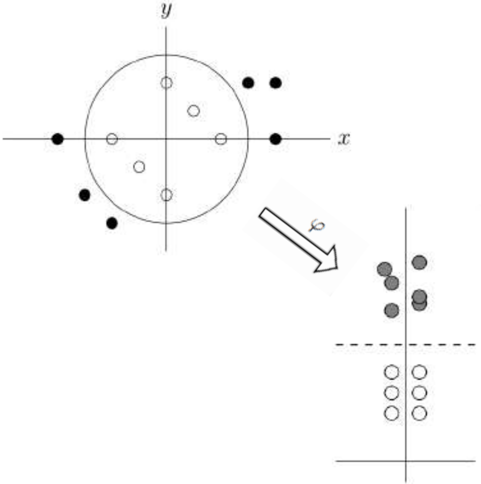


Geometric tranformation

- Consider two sets of points separated by a circle of radius 1
- Equation of circle is $x^2 + y^2 = 1$
- Points inside the circle $x^2 + y^2 < 1$
- Points outside circle $x^2 + y^2 > 1$
- Transformation

$$\varphi : (x, y) \mapsto (x, y, x^2 + y^2)$$

- Points inside circle lie below z = 1
- Point outside circle lifted above z = 1



SVM after transformation

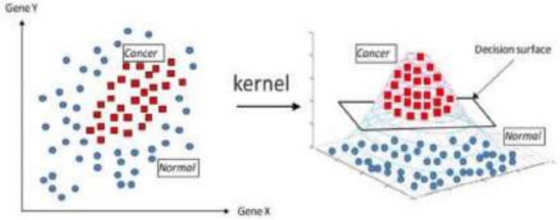
- SVM in original space

$$\operatorname{sign}\left[\sum_{i \in sv} y_i \alpha_i \langle x_i \cdot z \rangle + b\right]$$

- After transformation

$$\operatorname{sign}\left[\sum_{i \in sv'} y_i \alpha_i \langle \varphi(x_i) \cdot \varphi(z) \rangle + b\right]$$

- All we need to know is how to compute dot products in transformed space



Dot products

$\sqrt{2}x_1 \cdot \sqrt{2}z_1,$

$x = \langle x_1, x_2 \rangle$

- Consider the transformation

$$\varphi : (x_1, x_2) \mapsto (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1x_2, x_2^2)$$

- Dot product in transformed space

$\sqrt{2}z_2 z_2$

$$\langle \varphi(x) \cdot \varphi(z) \rangle = 1 + 2x_1z_1 + 2x_2z_2 + x_1^2z_1^2 + 2x_1x_2z_1z_2 + x_2^2z_2^2$$

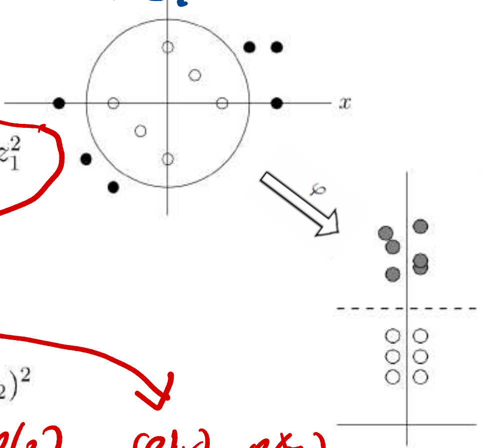
$$= (1 + x_1z_1 + x_2z_2)^2$$

x: (x_1, x_2) z: (z_1, z_2)

- Transformed dot product can be expressed in terms of original inputs

$$\langle \varphi(x) \cdot \varphi(z) \rangle = K(x, z) = (1 + x_1z_1 + x_2z_2)^2$$

$x \to \varphi(x)$, $z \to \varphi(z)$, $\varphi(x) \cdot \varphi(z)$



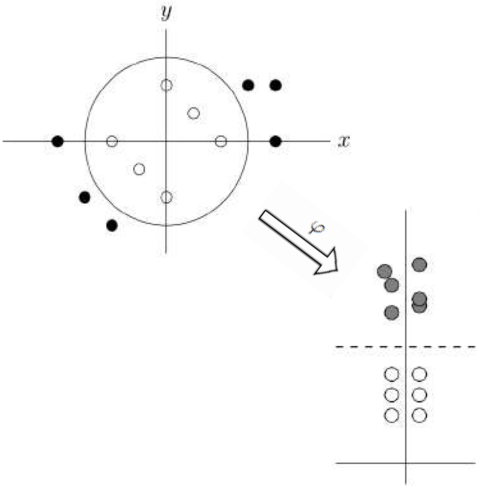
Kernels

- K is a *kernel* for transformation φ if

$$K(x,z) = \langle \varphi(x) \cdot \varphi(z) \rangle$$

- If we have a kernel, we don't need to explicitly compute transformed points
- All dot products can be computed implicitly using the kernel on original data points

$$\text{sign}\left[\sum_{i \in sv'} y_i \alpha_i \langle \varphi(x_i) \cdot \varphi(z) \rangle + b\right]$$



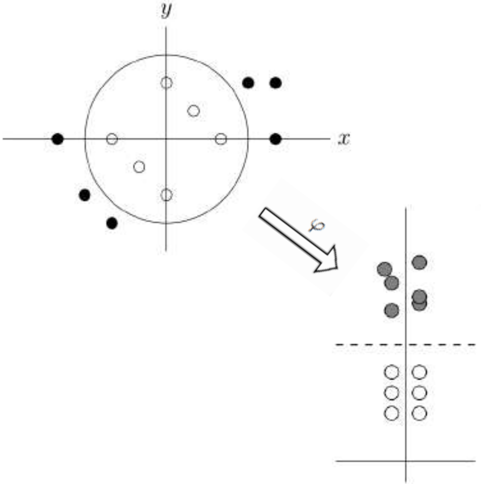
Kernels

- K is a *kernel* for transformation φ if

$$K(x, z) = \langle \varphi(x) \cdot \varphi(z) \rangle$$

- If we have a kernel, we don't need to explicitly compute transformed points
- All dot products can be computed implicitly using the kernel on original data points

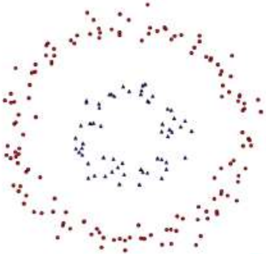
$$\mathrm{sign}\left[\sum_{i \in sv'} y_i \alpha_i K(x_i, z) + b\right]$$

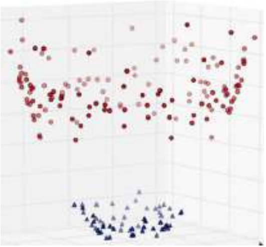


Kernels

$K(x,z)$ is it the kernel of some φ?

- If we know K is a kernel for some transformation φ, we can blindly use K without even knowing what φ looks like!
- When is a function a valid kernel?
- Has been studied in mathematics – Mercer's Theorem
  - Criteria are non-constructive
- Can define sufficient conditions from linear algebra





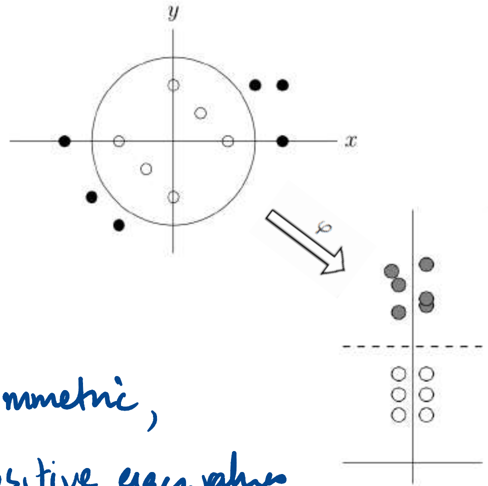
Kernels

- Kernel over training data $x_1, x_2, \ldots, x_N$ can be represented as a *gram matrix*

$$K = \begin{array}{c c} & \begin{array}{cccc} x_1 & x_2 & \cdots & x_N \end{array} \\ \begin{array}{c} x_1 \\ x_2 \\ \vdots \\ x_n \end{array} & \left[\begin{array}{cccc} & & & \\ & & & \\ & K(x_i, x_j) & & \\ & & & \end{array} \right] \end{array}$$

- Entries are values $K(x_i, x_j)$
- Gram matrix should be *positive semi-definite* for all $x_1, x_2, \ldots, x_N$

Symmetric,
Positive eigenvalues



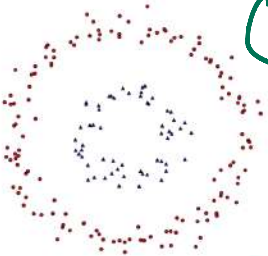
Known kernels

$K(x,z) = (1 + x_1 z_1 + x_2 z_2)^2$

- Fortunately, there are many known kernels
- Polynomial kernels

$$K(x,z) = (1 + \langle x \cdot z \rangle)^k$$

$(1+(x \cdot z))^2$



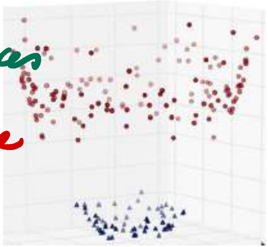
- Any K(x,z) representing a similarity measure

Gene sequences

Edit distance

- Gaussian radial basis function – similarity based on inverse exponential distance

$$K(x,z) = e^{-c|x-z|^2}$$



Kernel Methods

K_1, K_2 kernels $\rightarrow$ $K_1 + K_2$

$\rightarrow$ $K_1 \cdot K_2$

Try out a kernel & evaluate the result

Build up a library of kernels

"Manual" exercise