Lecture 18: 21 March, 2023

Madhavan Mukund

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

Data Mining and Machine Learning
January–April 2023

Soft margin optimization

Minimize $\frac{|w|}{2} + \sum_{i=1}^{N} \xi_i^2$

Add $C \cdot \sum \xi_i^2$

Use higher power than 2

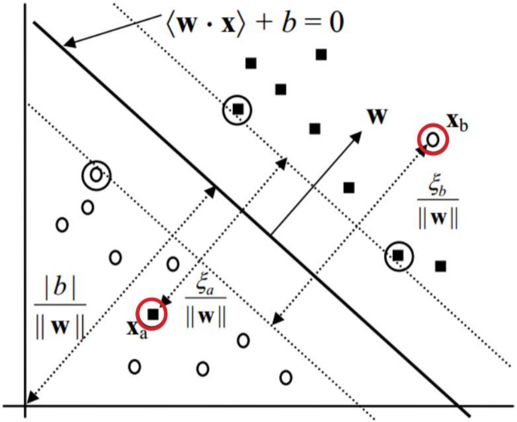
Subject to

$\xi_i \geq 0$

$w \cdot x_i + b > 1 - \xi_i$, if $y_i = 1$

$w \cdot x_i + b < -1 + \xi_i$, if $y_i = 1$ -1

- Constraints include requirement that error terms are non-negative
- Again the objective function is quadratic



Dualization

Wolfe dual

only dot products of x_i's

Maximize $\sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j)$

$\varphi(x_i) \cdot \varphi(x_j)$

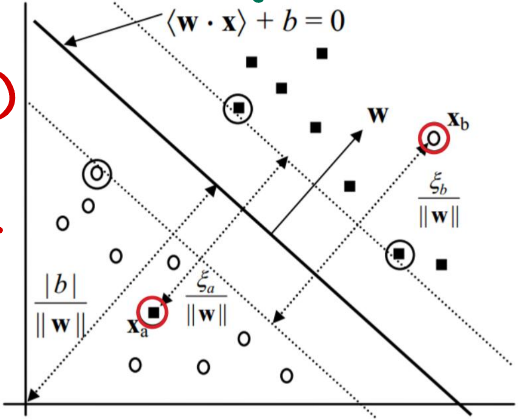
Subject to

$0 \leq \alpha_i \geq 1$

$\sum_i \alpha_i y_i = 0$

Lagrange Multipliers

No x_i's here

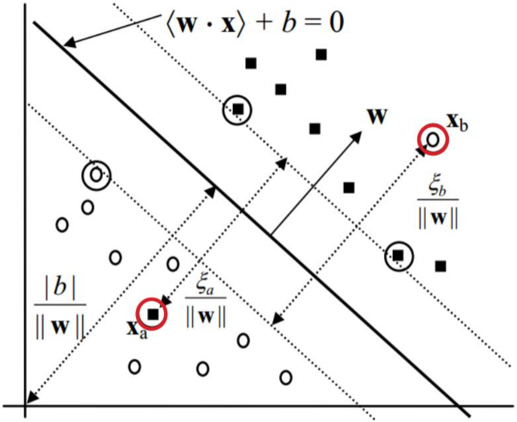


Soft margin optimization

- Can again be solved using convex optimization theory
- Form of the solution turns out to be the same as the hard margin case
  - Expression in terms of Lagrange multipliers α_i
  - Only terms corresponding to support vectors are actively used

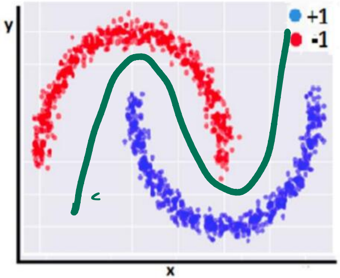
$$\text{sign}\left[\sum_{i \in sv} y_i \alpha_i (x_i \cdot z) + b\right]$$

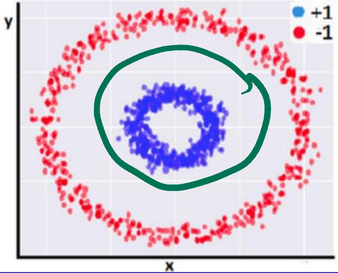
Only dot prod.



The non-linear case

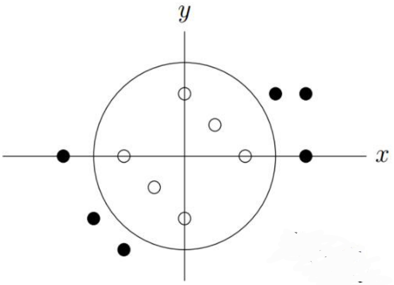
- How do we deal with datasets where the separator is a complex shape?
- Geometrically transform the data
  - Typically, add dimensions
- For instance, if we can "lift" one class, we can find a planar separator between levels





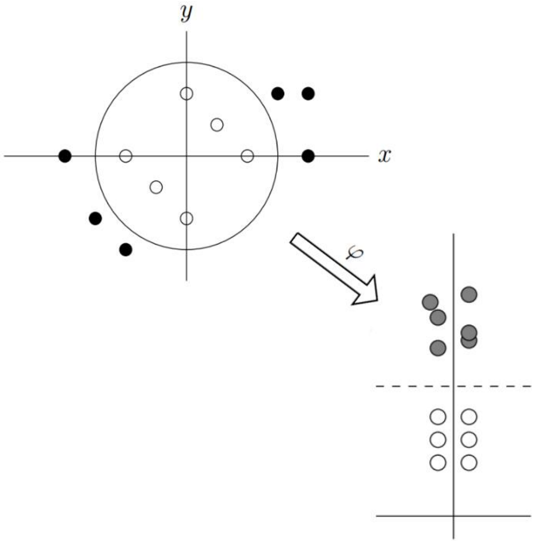
Geometric tranformation

- Consider two sets of points separated by a circle of radius 1
- Equation of circle is $x^2 + y^2 = 1$



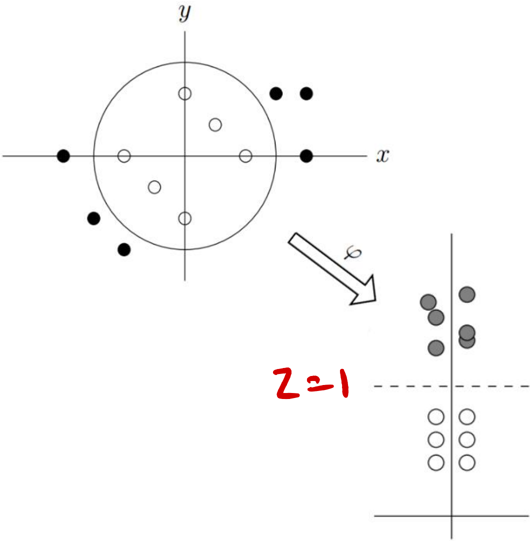
Geometric tranformation

- 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, $x^2 + y^2 > 1$



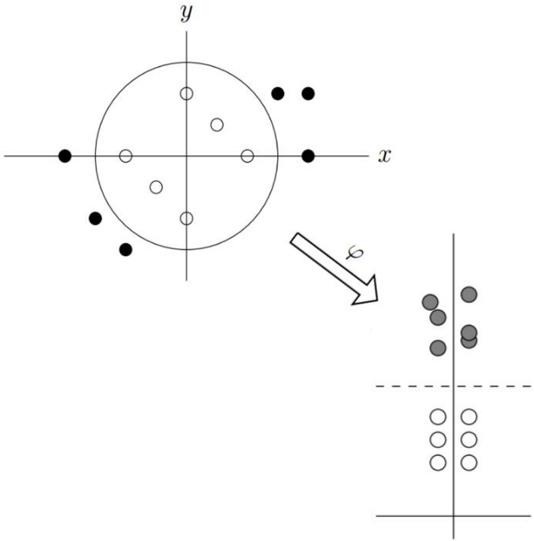
Geometric tranformation

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



Geometric tranformation

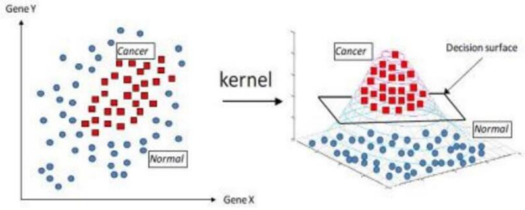
- Consider two sets of points separated by a circle of radius 1
- Equation of circle is $x^2 + y^2 = 1$
- Points inside the circle, $x^2 + y^2 < 1$
- Points outside circle, $x^2 + y^2 > 1$
- Transformation
  $$\varphi : (x, y) \mapsto (x, y, x^2 + y^2)$$
- Points inside circle lie below $z = 1$
- Point outside circle lifted above $z = 1$



SVM after transformation

- SVM in original space

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



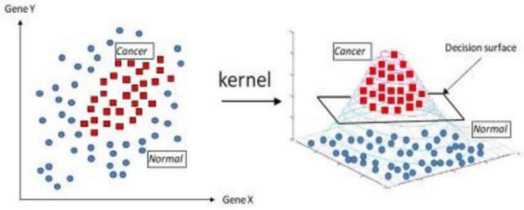
SVM after transformation

- SVM in original space

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

- After transformation

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



SVM after transformation

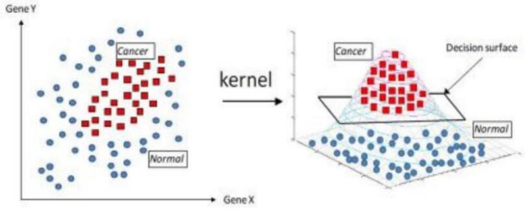
- SVM in original space

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

- After transformation

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

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



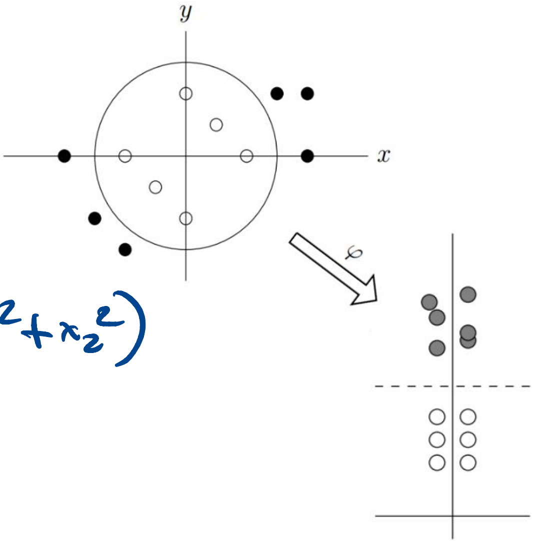
Dot products

- Consider the transformation

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

$x_2^2)$

$(x_1, x_1) \mapsto (x_1, x_2, x_1^2 + x_2^2)$



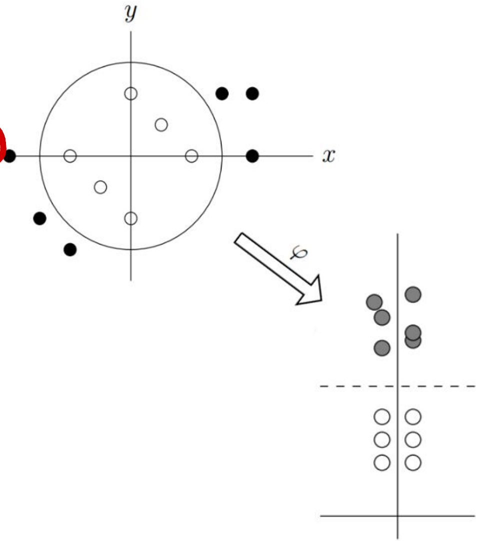
Dot products

- Consider the transformation

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

- Dot product in transformed space

$$\begin{aligned}\varphi(x) \cdot \varphi(z) &= 1 + 2x_1z_1 + 2x_2z_2 + x_1^2z_1^2 \\ &\quad +2x_1x_2z_1z_2 + x_2^2z_2^2 \\ &= (1 + x_1z_1 + x_2z_2)^2\end{aligned}$$



Dot products

- Consider the transformation

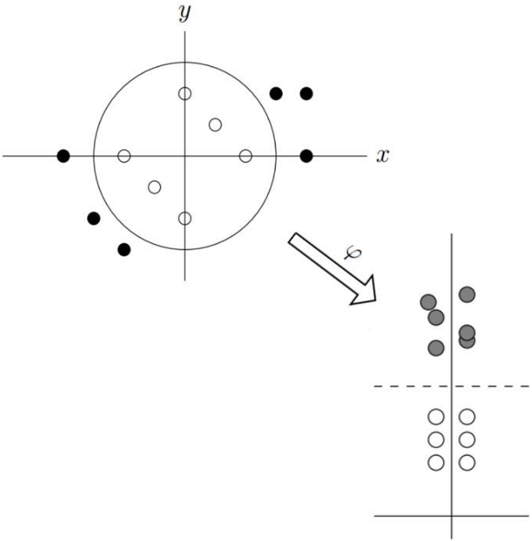
  $\varphi : (x_1, x_2) \mapsto (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1x_2, x$

- Dot product in transformed space

$$\begin{aligned} \varphi(x) \cdot \varphi(z) &= 1 + 2x_1z_1 + 2x_2z_2 + x_1^2z_1^2 \\ &\quad + 2x_1x_2z_1z_2 + x_2^2z_2^2 \\ &= (1 + x_1z_1 + x_2z_2)^2 \end{aligned}$$

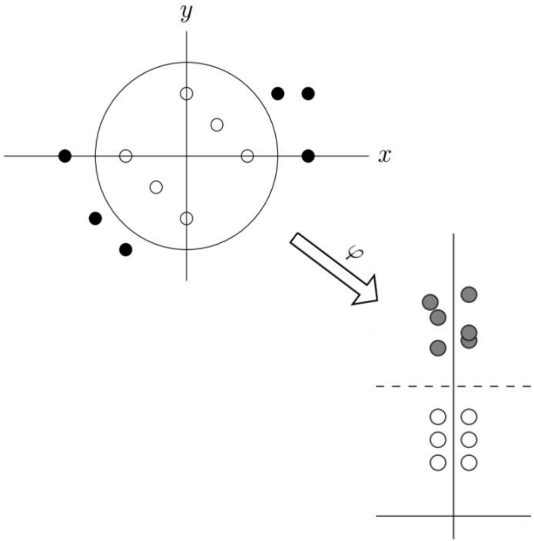
- Transformed dot product can be expressed in terms of original inputs

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



Kernels

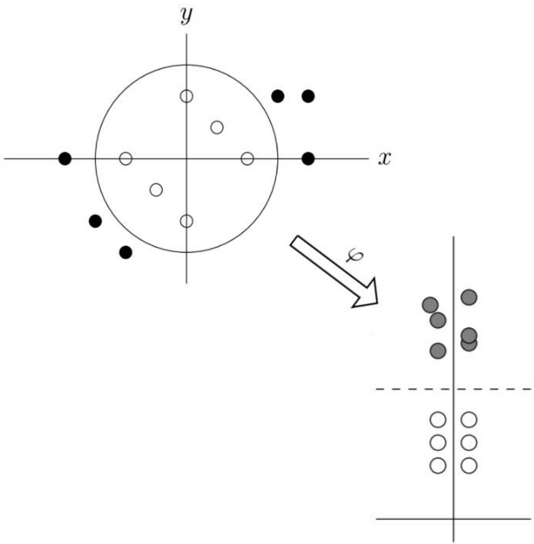
- K is a kernel for transformation φ if $K(x,z) = \varphi(x) \cdot \varphi(z)$



Kernels

- K is a kernel for transformation φ if $K(x,z) = \varphi(x) \cdot \varphi(z)$
- 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

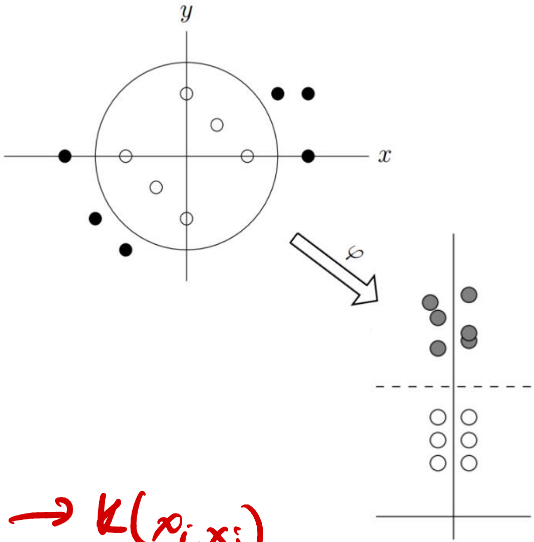
$$\text{sign}\left[\sum_{i \in sv} y_i \alpha_i (\varphi(x_i) \cdot \varphi(z)) + b\right]$$



- K is a kernel for transformation φ if $K(x, z) = \varphi(x) \cdot \varphi(z)$
- 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

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

Also in $\varphi(x_i).\varphi(x_j)$ of dual $\rightarrow K(x_i, x_j)$

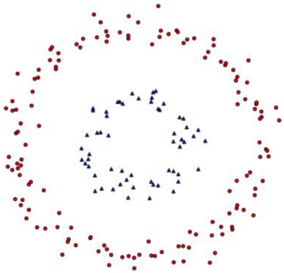


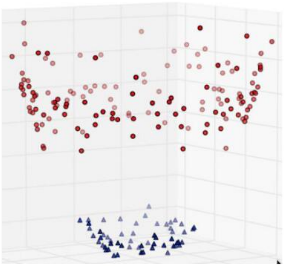
Kernels

- If we know K is a kernel for some transformation φ, we can blindly use K without even knowing what φ looks like!

$K(x_i, x_j)$ in training

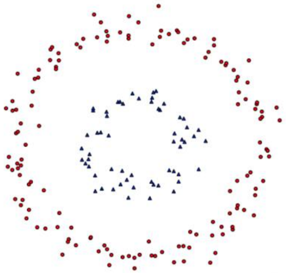
$K(x_i, z)$ for classification

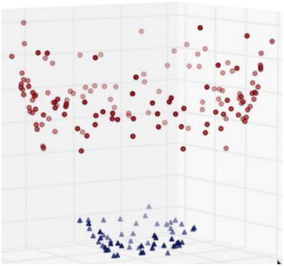




Kernels

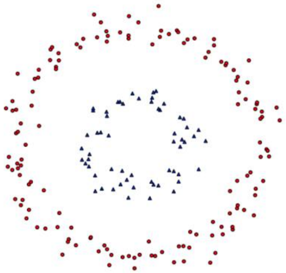
- If we know K is a kernel for some transformation φ, we can blindly use K without even knowing what φ looks like!
- When is a function a valid kernel?

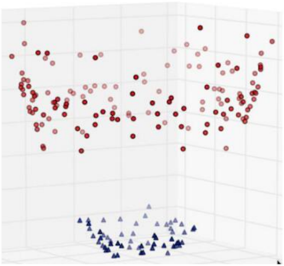




Kernels

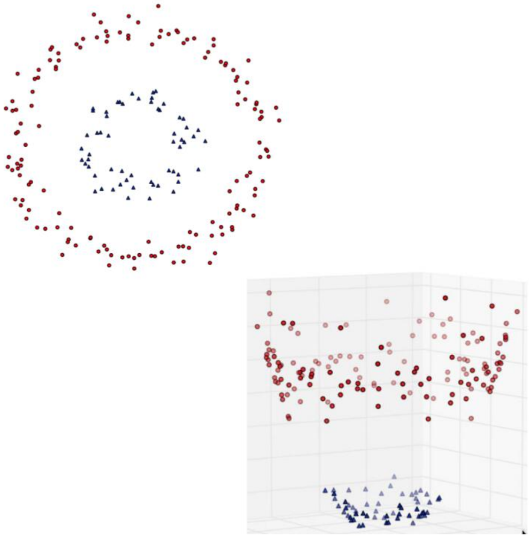
- If we know K is a kernel for some transformation φ, we can blindly use K without even knowing what φ looks like!
- When is a function a valid kernel?
- Has been studied in mathematics — Mercer's Theorem
  - Criteria are non-constructive





Kernels

- 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

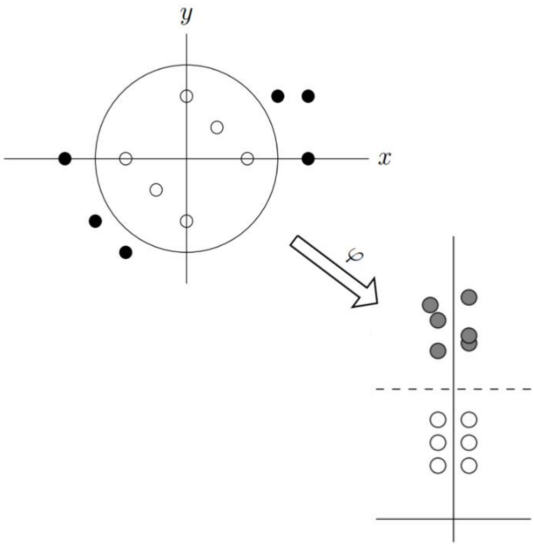


Kernels

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

$$K = \begin{array}{c} \\ x_1 \\ x_2 \\ \vdots \\ x_N \end{array} \begin{array}{c} x_1 \quad x_2 \quad \cdots \quad x_N \\ \left[\begin{array}{cccc} & & & \\ & & & \\ & & & \\ & & & \end{array} \right] \end{array}$$

- Entries are values $K(x_i, x_j)$

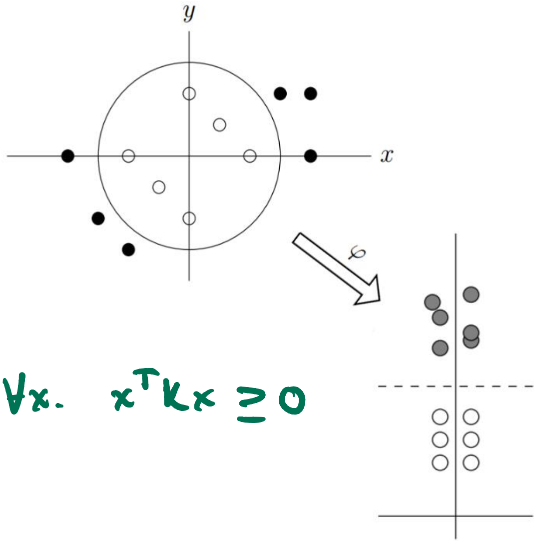


Kernels

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

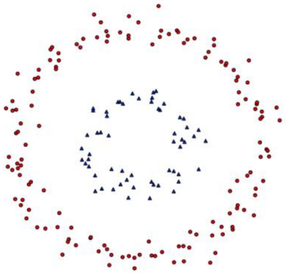
$$K = \begin{array}{c} \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{array} \begin{array}{c} \begin{array}{cccc} x_1 & x_2 & \cdots & x_n \end{array} \\ \left[\begin{array}{cccc} & & & \\ & & & \\ & & & \\ & & & \end{array} \right] \end{array}$$

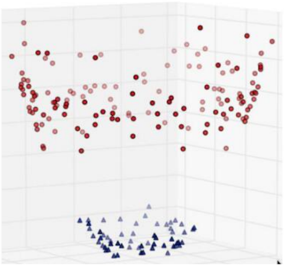
- Entries are values $K(x_i, x_j)$
- Gram matrix should be positive semi-definite for all $x_1, x_2, \ldots, x_N$ → $\forall x.\ x^T K x \geq 0$



Known kernels

- Fortunately, there are many known kernels

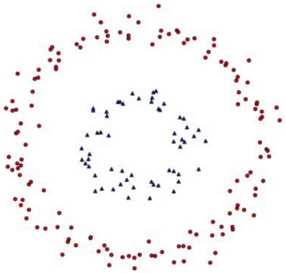


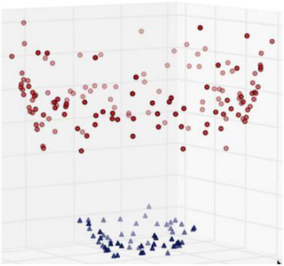


Known kernels

- Fortunately, there are many known kernels
- Polynomial kernels

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



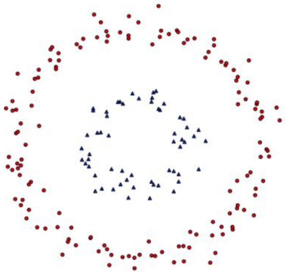


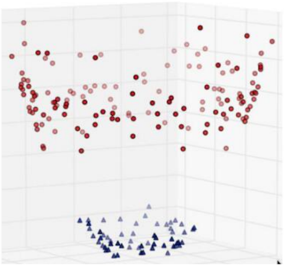
$$(1 + x_1 z_1 + x_2 z_2)^2$$

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

Known kernels

- Fortunately, there are many known kernels
- Polynomial kernels
$$K(x,z) = (1 + x \cdot z)^k$$
- Any $K(x,z)$ representing a similarity measure



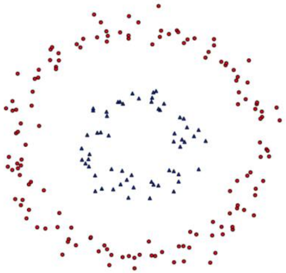


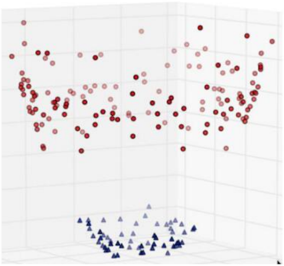
- Fortunately, there are many known kernels
- Polynomial kernels

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

- Any $K(x, z)$ representing a similarity measure
- Gaussian radial basis function — similarity based on inverse exponential distance

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





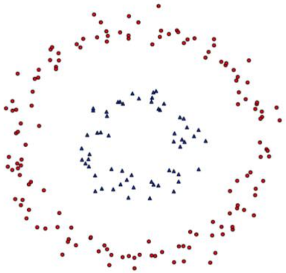
Known kernels

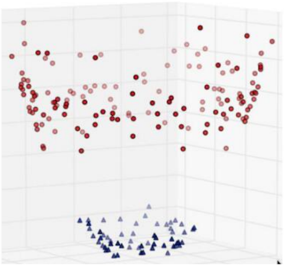
- Fortunately, there are many known kernels
- Polynomial kernels

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

- Any $K(x, z)$ representing a similarity measure
- Gaussian radial basis function — similarity based on inverse exponential distance

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





Till about 2010

SVM + manually constructed kernels

were "best" classifiers

Then came neural networks

This happens "automatically"