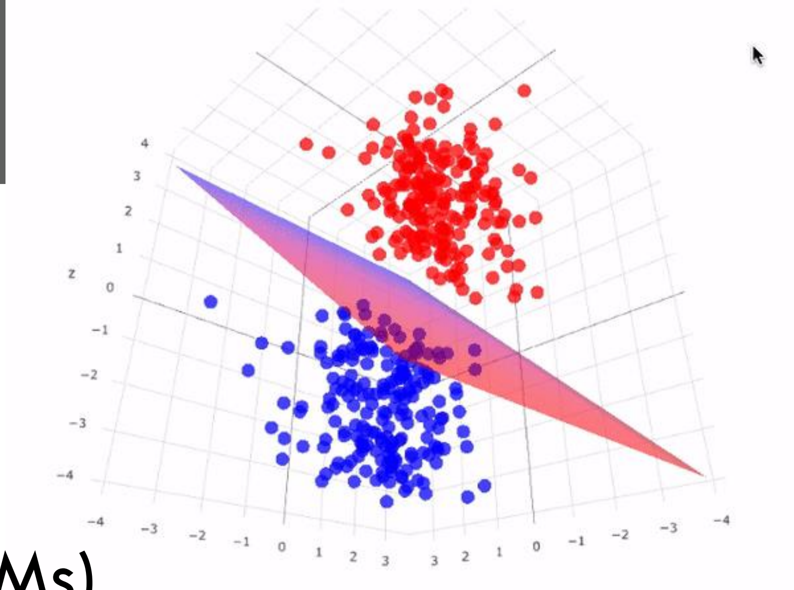
CHAPTER 13:

KERNEL MACHINES

Stella Grasshof

Overview



- Revisit
- Kernel Machines:
  Support Vector Machines (SVMs)
- Supervised classification of:
  (non)linearly separable classes
- Regression
- Unsupervised learning: one-class classification
- Kernel PCA

https://medium.com/nyu-a3sr-data-science-team/support-vector-machines-and-wine-cef59ad38b41

Reminder: Multivariate Derivatives

- A function with vector input and scalar output:

$$f : \mathbb{R}^2 \rightarrow \mathbb{R},\ \boldsymbol{v} = (v_1, v_2)^{\mathrm{T}}$$

$$f(\boldsymbol{v}) = \boldsymbol{v}^T \boldsymbol{v} + 2 = v_1^2 + v_2^2 + 2$$

- Its gradient is defined by its partial derivatives as

$$\nabla f : \mathbb{R}^2 \rightarrow \mathbb{R}^2$$

$$\nabla f(\boldsymbol{v}) = \begin{pmatrix} \frac{\partial}{\partial v_1} f(\boldsymbol{v}) \\ \frac{\partial}{\partial v_2} f(\boldsymbol{v}) \end{pmatrix} = \begin{pmatrix} 2v_1 \\ 2v_2 \end{pmatrix} = 2\boldsymbol{v}$$

Reminder: Multivariate Derivatives

- Compute the gradient of a function

$$g(\boldsymbol{v}) = \sum_{i=1}^{2}\sum_{j=1}^{2} v_i v_j = v_1^2 + v_2^2 + 2v_1 v_2$$

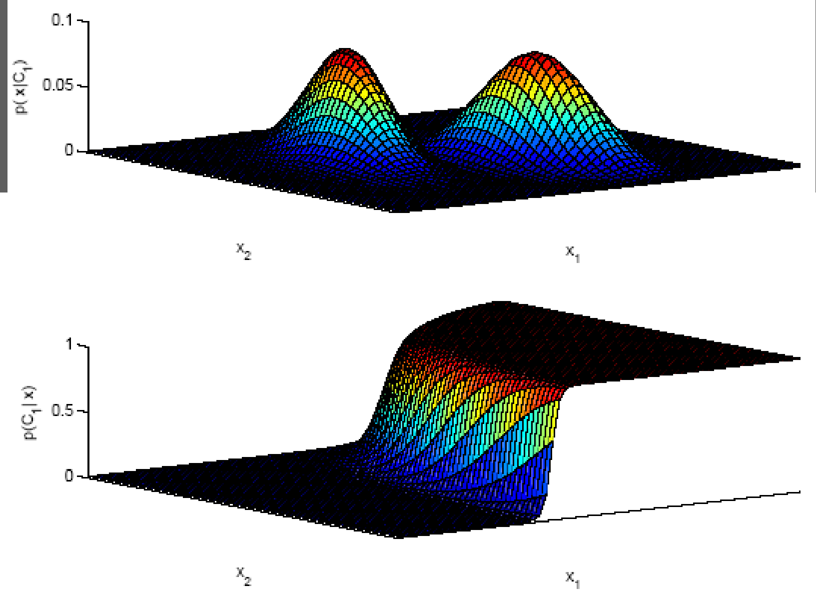
$$\nabla g(\boldsymbol{v}) = \begin{pmatrix} 2v_1 + 2v_2 \\ 2v_2 + 2v_1 \end{pmatrix} = 2\begin{pmatrix} v_1 + v_2 \\ v_1 + v_2 \end{pmatrix}$$

$$= 2\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}\begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$$

- Example of rewriting (helpful for exercise)

$$\begin{pmatrix} v_1 + 4v_2 \\ 2v_1 + 3v_2 \end{pmatrix} = \begin{pmatrix} 1 & 4 \\ 2 & 3 \end{pmatrix}\begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$$

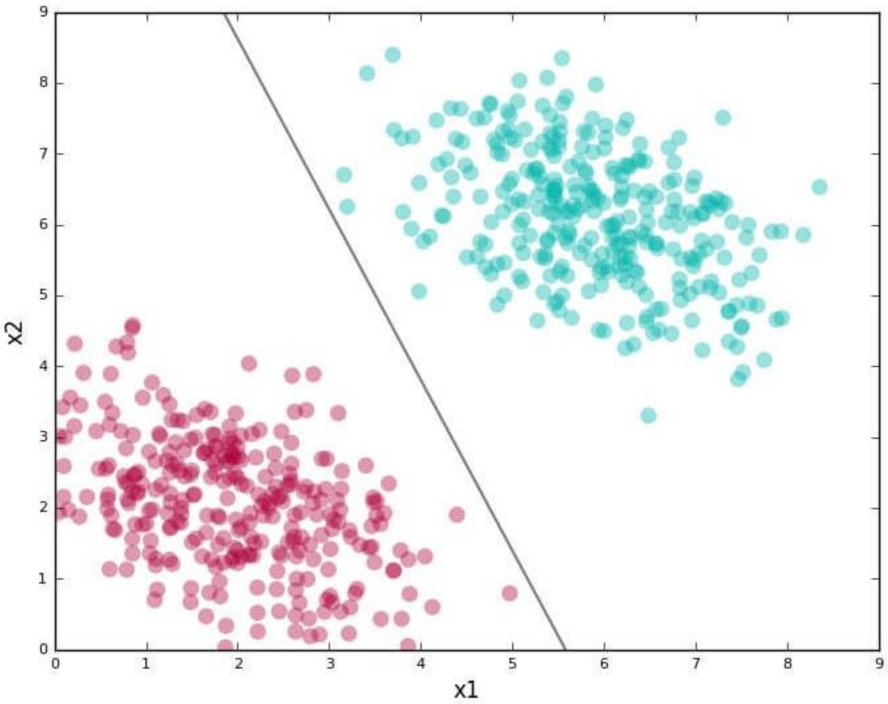
Kernel Machines

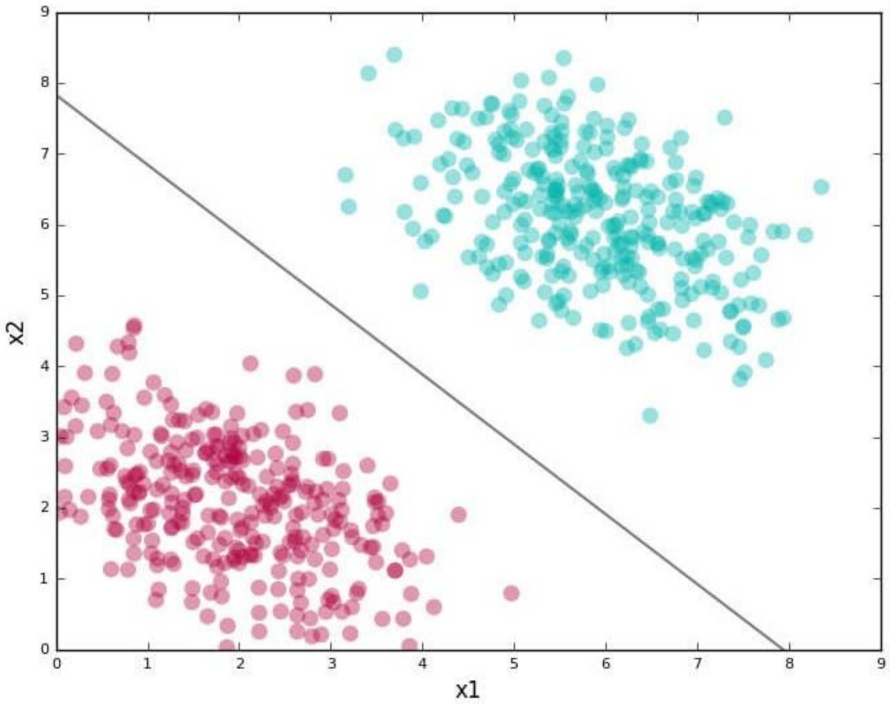


- Discriminant-based:
  previously defined by densities,
  here:
  we want few training points aka **support vectors**
- Kernel functions:
  application-specific measures of similarity
- No need to represent instances as vectors
- Convex optimization problems with a unique solution

Separating two classes by one line

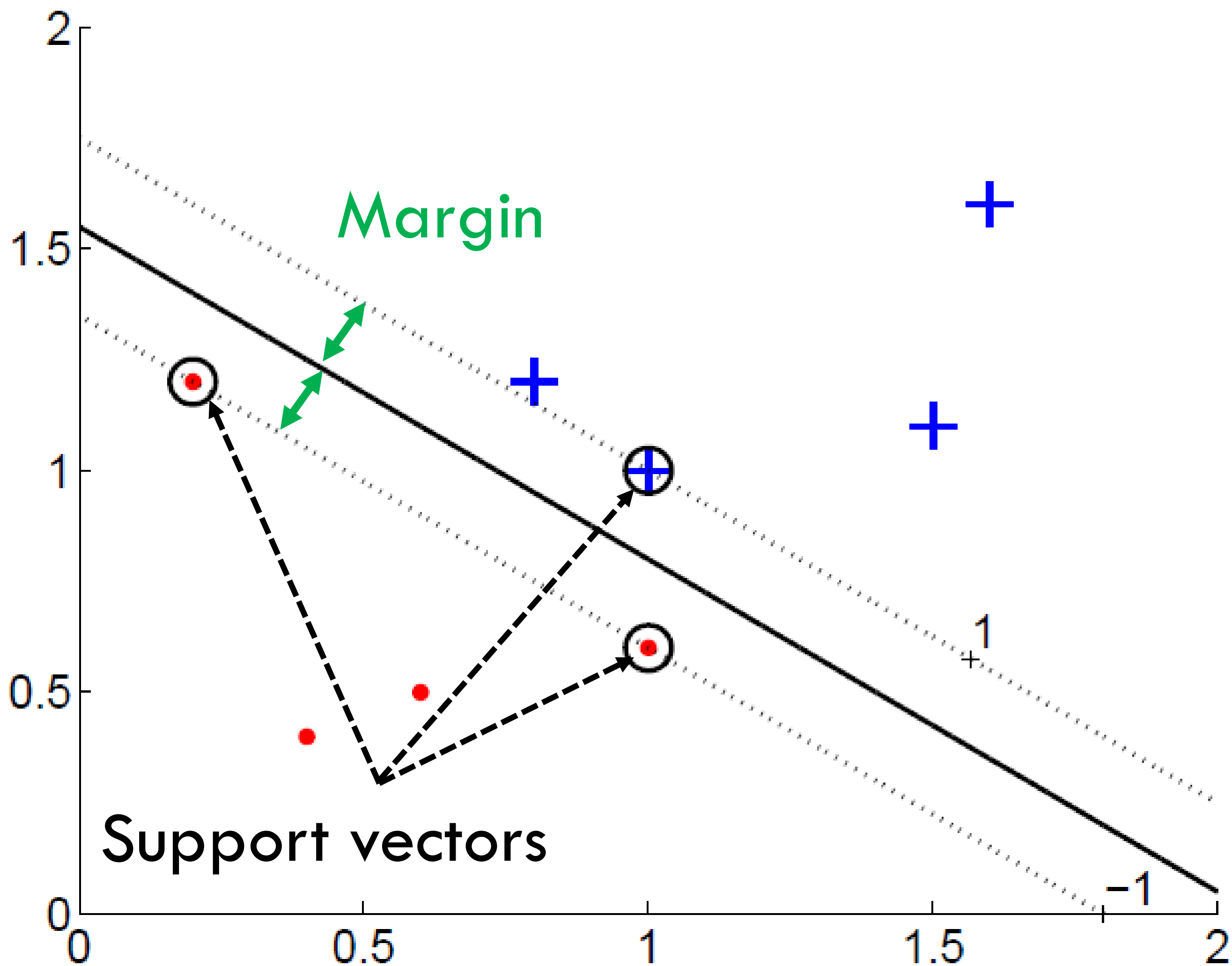
Which is better?





https://blog.statsbot.co/support-vector-machines-tutorial-c1618e635e93

Support Vector Machines (SVMs)



Support Vector Machines (SVMs)

Support Vector Machines (SVMs)

find the **optimal separating hyperplane** which:

- Separates the two classes
- Maximizes the margin between them, i.e. maximize distance between plane and the points closest to it
- Hyperplane is defined by few training samples

Repeat: discriminant function

Classification:

- Discriminant function require the probability density function (pdf):

$$g_i(\boldsymbol{x}) = \log p(\boldsymbol{x}|C_i) + \log P(C_i)$$

$$g_k(\boldsymbol{x}) = \max_i g_i(\boldsymbol{x})$$

- Separates the two classes

$$g_1(\boldsymbol{x}_i) = \boldsymbol{w}_1^{\mathrm{T}} \boldsymbol{x}_i + w_{10}$$

$$g_2(\boldsymbol{x}_i) = \boldsymbol{w}_2^{\mathrm{T}} \boldsymbol{x}_i + w_{20}$$

$$\Rightarrow g(\boldsymbol{x}_i) = g_1(\boldsymbol{x}_i) - g_2(\boldsymbol{x}_i) = (\boldsymbol{w}_1 - \boldsymbol{w}_2)^{\mathrm{T}} \boldsymbol{x}_i + (w_{10} - w_{20})$$

$$g(\boldsymbol{x}_i) = \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i + w_0 \begin{cases} > 0 & , \text{ class } C_1 \\ \leq 0 & , \text{ class } C_2 \end{cases}$$

Optimal Separating Hyperplane

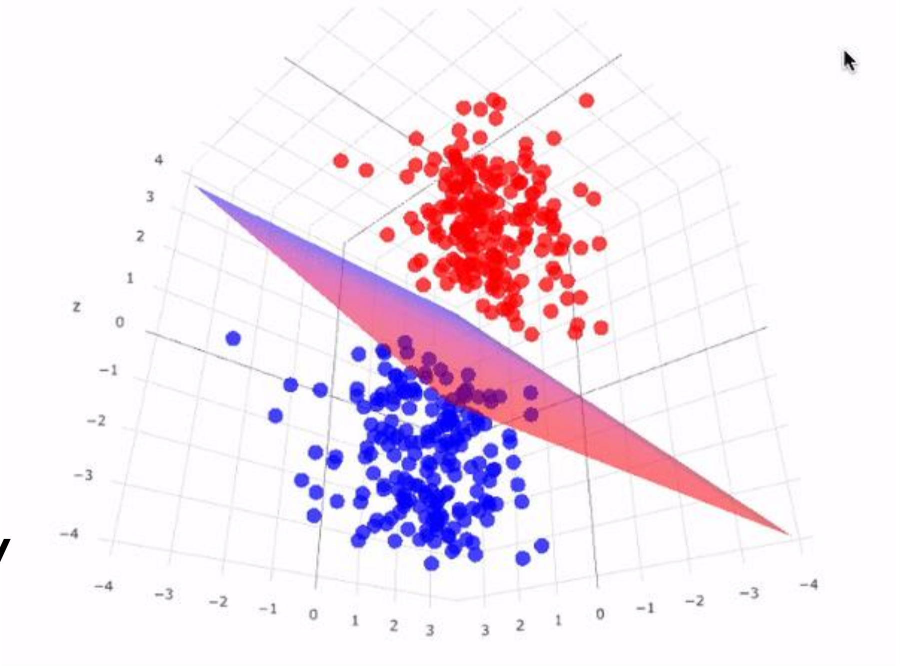
- Given points $\boldsymbol{x}_i \in \mathbb{R}^D,\ i = 1, \ldots, N$ of two classes with labels:

$$r_i = \begin{cases} +1, & \text{if } \boldsymbol{x}_i \in C_1 \\ -1, & \text{if } \boldsymbol{x}_i \in C_2 \end{cases}$$

- Points on the hyperplane:

$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + w_0 = 0$$

- Class decision as previously



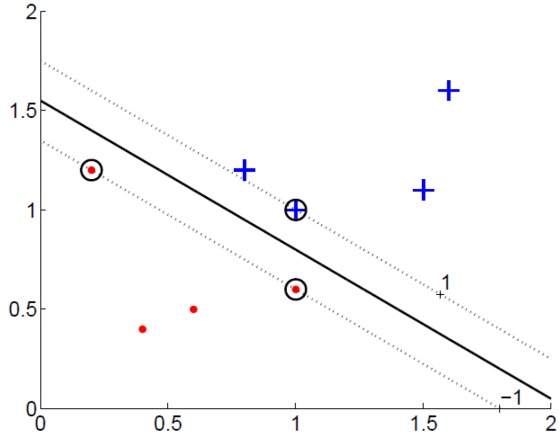
$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + w_0 \begin{cases} > 0, & \text{class } C_1 : \ r_i = +1 \\ \leq 0, & \text{class } C_2 : \ r_i = -1 \end{cases}$$

https://medium.com/nyu-a3sr-data-science-team/support-vector-machines-and-wine-cef59ad38b41

Optimal Separating Hyperplane

- Previous class decision

$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + w_0 \begin{cases} > 0, & \text{class } C_1 : \; r_i = +1 \\ \leq 0, & \text{class } C_2 : \; r_i = -1 \end{cases}$$



- Now: we want points to be some distance away of the hyperplane

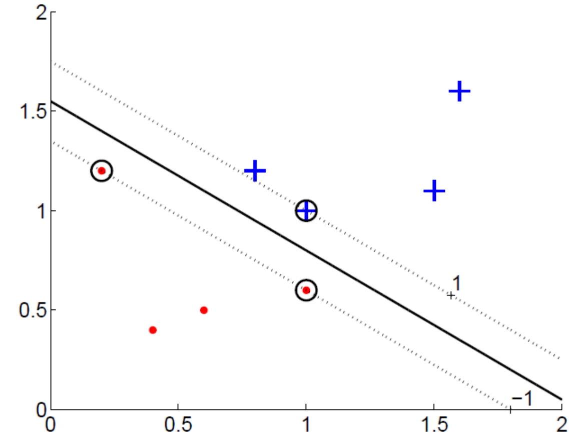
$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + w_0 \begin{cases} \geq 1, & r_i = +1 \\ \leq -1, & r_i = -1 \end{cases}$$

This can be rewritten

$$r_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + w_0) \geq 1$$

Optimal Separating Hyperplane

- Data points $\boldsymbol{x}_i \in \mathbb{R}^D$ with labels $r_i \in \{-1, 1\}$ then decision boundary is
$$r_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + w_0) \geq 1$$
unknown: $\boldsymbol{w},\ w_0$



- Distance of points to the hyperplane should be large:

$$\max \frac{\left|\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + w_0\right|}{\|\boldsymbol{w}\|_2} \geq \rho, \quad \forall i$$

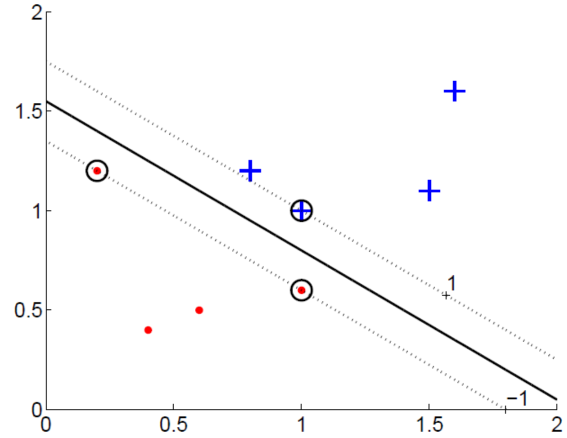
$$\max \frac{r_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + w_0)}{\|\boldsymbol{w}\|_2} \geq \rho, \quad \forall i$$

Optimal Separating Hyperplane

- Data points $\boldsymbol{x}_i \in \mathbb{R}^D$ with labels $r_i \in \{-1, 1\}$ then decision boundary is

$$r_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + w_0) \geq 1$$

unknown: $\boldsymbol{w},\ w_0$



- Distance of points to the hyperplane should be large:

maximize

$$\max \frac{r_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + w_0)}{\|\boldsymbol{w}\|_2} \geq \rho, \quad \forall i$$

minimize

- Fix $\|\boldsymbol{w}\|_2\, \rho = 1$ then:

$$\min \|\boldsymbol{w}\|_2^2, \text{ subject to } r_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + w_0) \geq 1, \quad \forall i$$

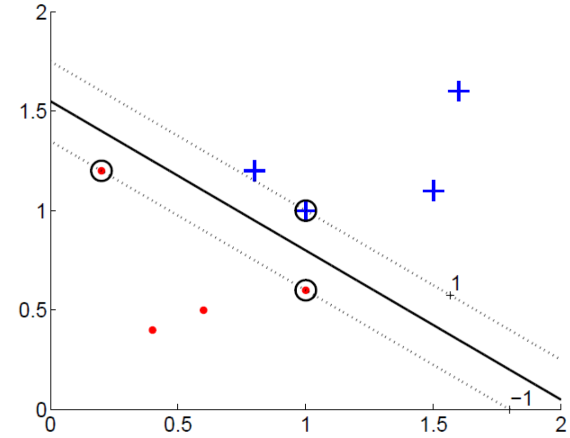
Optimal Separating Hyperplane

- Data points $\boldsymbol{x}_i \in \mathbb{R}^D$ with labels $r_i \in \{-1, 1\}$

$$r_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + w_0) \geq 1$$

unknown: $\boldsymbol{w},\ w_0$

- Distance of points to the hyperplane should be large:



maximize

$$\max \frac{r_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + w_0)}{\|\boldsymbol{w}\|_2} \geq \rho, \quad \forall i$$

minimize

- Lagrange function

$$L(\boldsymbol{w}, w_0, \alpha_i) = \frac{1}{2}\|\boldsymbol{w}\|_2^2 - \sum_{i=1}^{N} \alpha_i (r_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + w_0) - 1)$$

$$= \frac{1}{2}\|\boldsymbol{w}\|_2^2 - \sum_{i=1}^{N} \alpha_i r_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + w_0) + \sum_{i=1}^{N} \alpha_i,\ \alpha_i \geq 0$$

Optimal Separating Hyperplane

- Data points $\boldsymbol{x}_i \in \mathbb{R}^D$ with labels $r_i \in \{-1, 1\}$

$$r_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + w_0) \geq 1$$

unknown: $\boldsymbol{w},\ w_0$

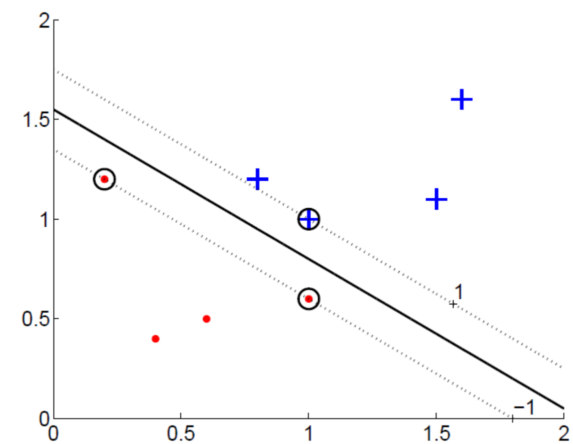
- Lagrange function

$$L(\boldsymbol{w}, w_0, \alpha_i) = \frac{1}{2}\|\boldsymbol{w}\|_2^2 - \sum_{i=1}^{N} \alpha_i r_i \left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + w_0\right) + \sum_{i=1}^{N} \alpha_i$$

- Minimize *L* for $\boldsymbol{w},\ w_0$

$$\nabla_{\boldsymbol{w}} L(\boldsymbol{w}, w_0, \alpha_i) = \boldsymbol{w} - \sum_{i=1}^{N} \alpha_i r_i \boldsymbol{x}_i \stackrel{!}{=} 0$$

$$\frac{\partial}{\partial w} L(\boldsymbol{w}, w_0, \alpha_i) = -\sum_{i=1}^{N} \alpha_i r_i \stackrel{!}{=} 0$$



Optimal Separating Hyperplane

- Data points $\boldsymbol{x}_i \in \mathbb{R}^D$ with labels $r_i \in \{-1, 1\}$

$$r_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + w_0) \geq 1$$

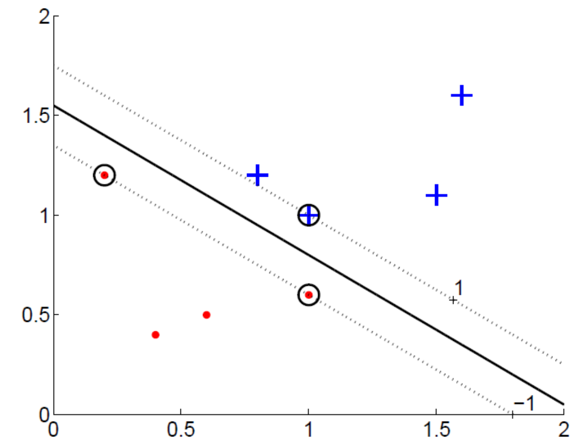
unknown: $\boldsymbol{w},\ w_0$

- Lagrange function

$$L(\boldsymbol{w}, w_0, \alpha_i) = \frac{1}{2}\|\boxed{\boldsymbol{w}}\|_2^2 - \boxed{\sum_{i=1}^{N} \alpha_i r_i}(\boxed{\boldsymbol{w}^{\mathrm{T}}}\boldsymbol{x}_i + w_0) + \sum_{i=1}^{N} \alpha_i$$

- Minimize *L* for $\boldsymbol{w},\ w_0$

$$\boxed{\boldsymbol{w} = \sum_{i=1}^{N} \alpha_i r_i \boldsymbol{x}_i} \qquad \boxed{\sum_{i=1}^{N} \alpha_i r_i = 0}$$



Optimal Separating Hyperplane

- Data points $\boldsymbol{x}_i \in \mathbb{R}^D$ with labels $r_i \in \{-1, 1\}$

$$r_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + w_0) \geq 1$$

  unknown: $\boldsymbol{w},\ w_0$

- Lagrange function

$$\boxed{\boldsymbol{w} = \sum_{i=1}^{N} \alpha_i r_i \boldsymbol{x}_i} \qquad \boxed{\sum_{i=1}^{N} \alpha_i r_i = 0}$$

$$L(\boldsymbol{w}, w_0, \alpha_i) = \frac{1}{2}\|\boldsymbol{w}\|_2^2 - \sum_{i=1}^{N} \alpha_i r_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + w_0) + \sum_{i=1}^{N} \alpha_i$$

$$= \frac{1}{2}\boxed{\boldsymbol{w}}^{\mathrm{T}}\boxed{\boldsymbol{w}} - \boxed{\boldsymbol{w}}^{\mathrm{T}}\boxed{\sum_{i=1}^{N} \alpha_i r_i \boldsymbol{x}_i} - w_0 \boxed{\sum_{i=1}^{N} \alpha_i r_i} + \sum_{i=1}^{N} \alpha_i$$

$$= -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j r_i r_j \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j + \sum_{i=1}^{N} \alpha_i$$

Optimal Separating Hyperplane

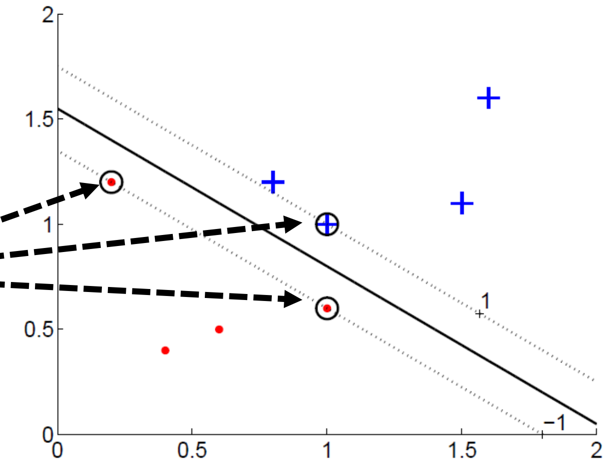
- Data points $\boldsymbol{x}_i \in \mathbb{R}^D$ with labels $r_i \in \{-1, 1\}$

$$r_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + w_0) \geq 1$$

- **Known:** $\boldsymbol{w} = \sum_{i=1}^{N} \alpha_i r_i \boldsymbol{x}_i$

  and if x_k **support vector,** then

$$w_0 = r_k - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_k$$



- **unknown:** $\alpha_i > 0$ are **support vectors**
- Max Lagrange function for α_i, with: $\sum_{i=1}^{N} \alpha_i r_i = 0, \quad \alpha_i \geq 0$

$$L(\boldsymbol{w}, w_0, \alpha_i) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j r_i r_j \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j + \sum_{i=1}^{N} \alpha_i$$

Sequential Minimal Optimization (SMO)

$$\max_{\alpha_i} \; L(\boldsymbol{w}, w_0, \alpha_i) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j r_i r_j \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j + \sum_{i=1}^{N}\alpha_i$$

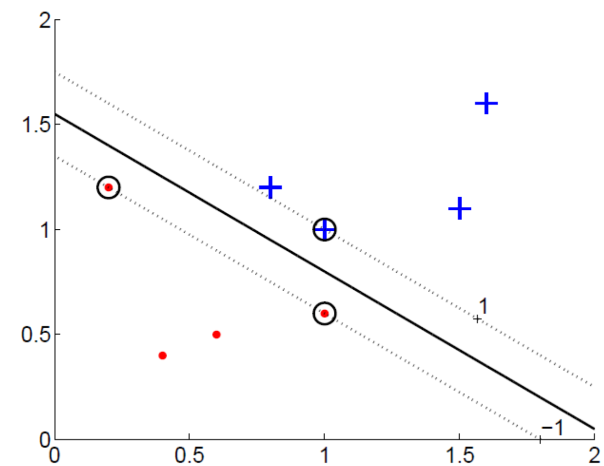
$$\sum_{i=1}^{N}\alpha_i r_i = 0, \quad 0 \leq \alpha_i \leq C$$

How to estimate the Lagragian Multipliers α_i ?

By: **SMO**

Idea: estimate two: α_k, α_l , keep the rest fixed

1) Find α_k that violates conditions
2) Pick a second multiplier α_l and optimize the pair α_k, α_l
3) Repeat 1) and 2)

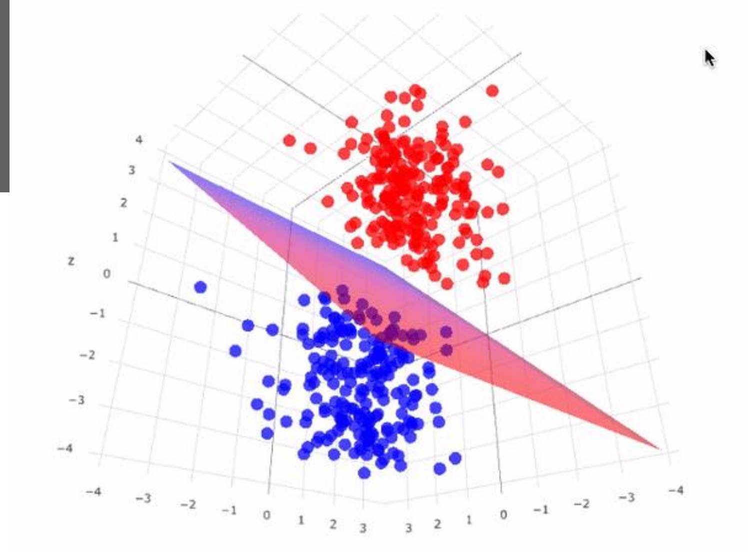


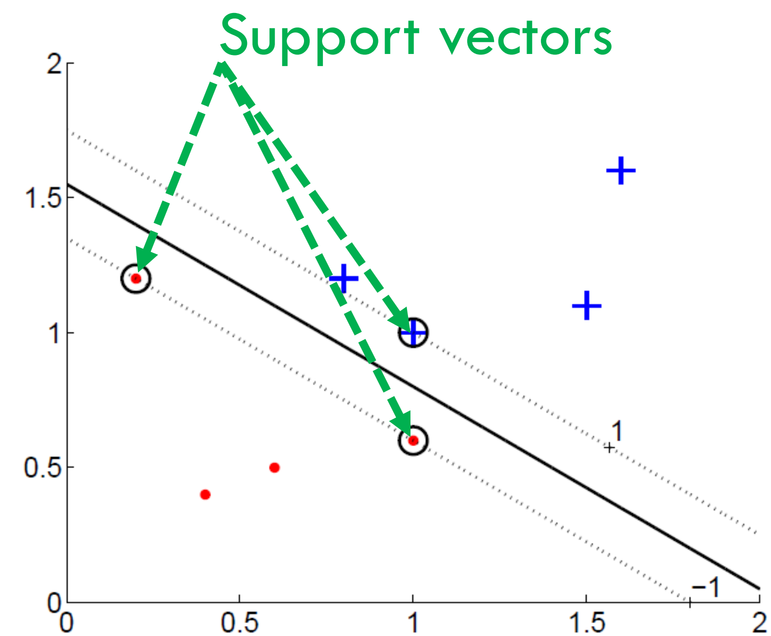
Summary SVM

- Data points $\boldsymbol{x}_i \in \mathbb{R}^D$
  with labels $r_i \in \{-1, 1\}$
- SVM:
  (1) find support vectors,
  defined by $\alpha_i > 0$
  which then define $\boldsymbol{w},\ w_0$
  (2) the parameters
  needed to classify new points

$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + w_0 \begin{cases} \geq 1, & r_i = +1 \\ \leq -1, & r_i = -1 \end{cases}$$

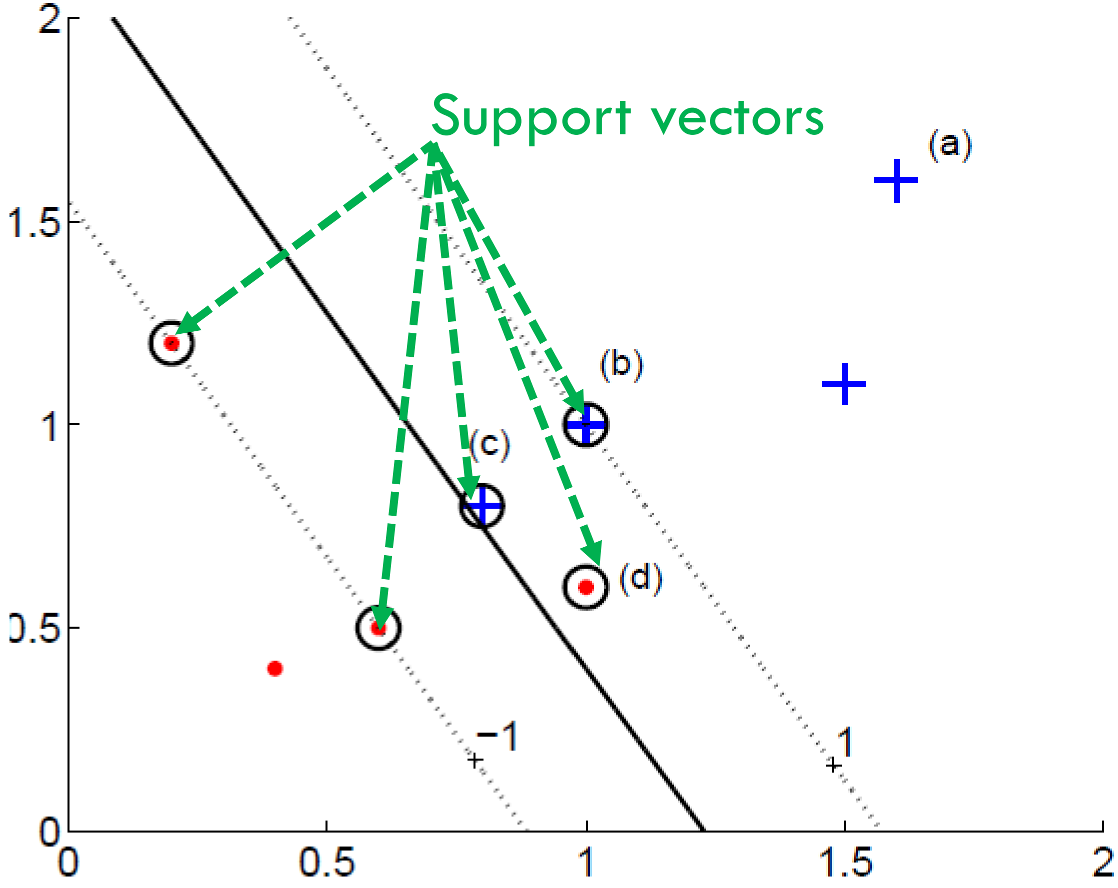
- Problem:
  - Assumption linearly separable
  - Two classes only





Soft Margin Hyperplane

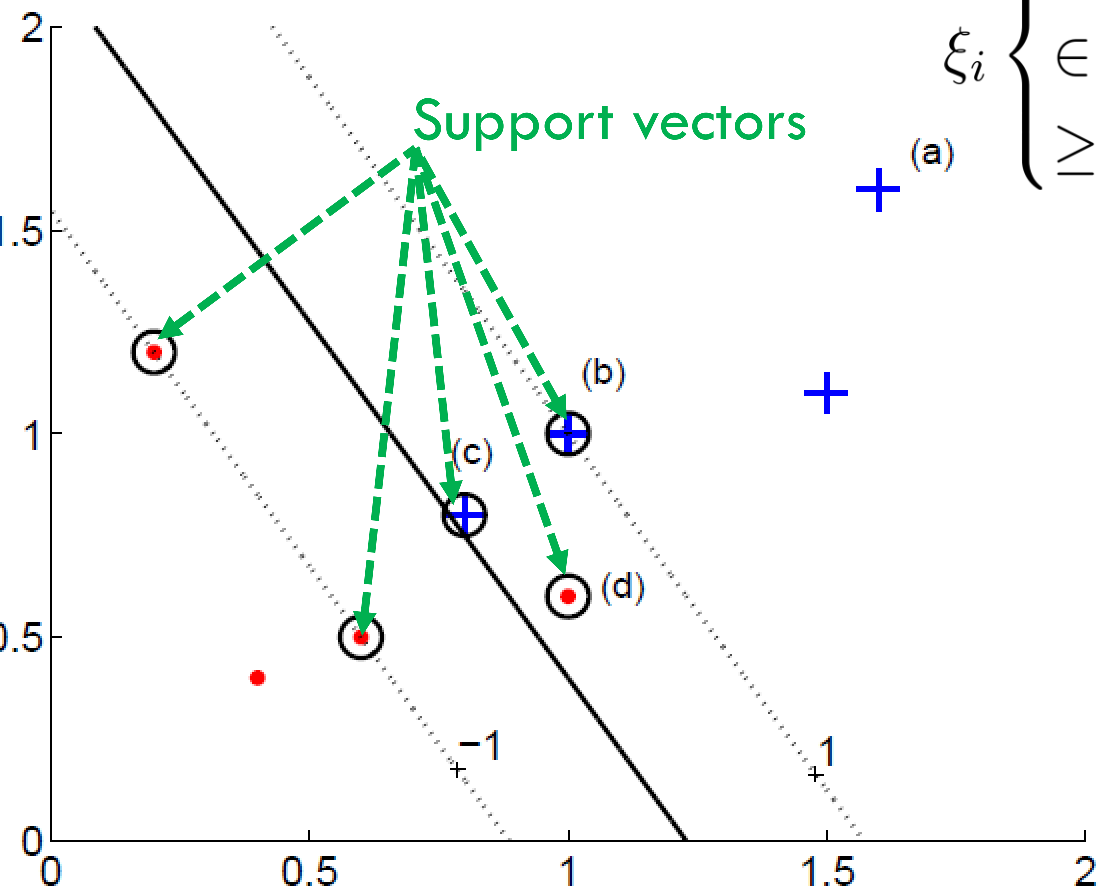
There is no Separating Hyperplane



Soft Margin Hyperplane

- Linearly separable $r_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + w_0) \geq 1$
- Not linearly separable $r_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + w_0) \geq 1 - \xi$

$$\xi_i \begin{cases} = 0 & , \ \boldsymbol{x}_i \text{ ok} \ \text{(a), (b)} \\ \in (0,1) & , \ \boldsymbol{x}_i \text{ in margin} \ \text{(c)} \\ \geq 1 & , \ \boldsymbol{x}_i \text{ falsely classified} \ \text{(d)} \end{cases}$$



Soft Margin Hyperplane

- Linearly separable $r_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + w_0) \geq 1$
- Not linearly separable $r_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + w_0) \geq 1 - \xi_i$

$$\xi_i \begin{cases} = 0 & , \ \boldsymbol{x}_i \text{ ok} \\ \in (0,1) & , \ \boldsymbol{x}_i \text{ in margin} \\ \geq 1 & , \ \boldsymbol{x}_i \text{ falsely classified} \end{cases}$$

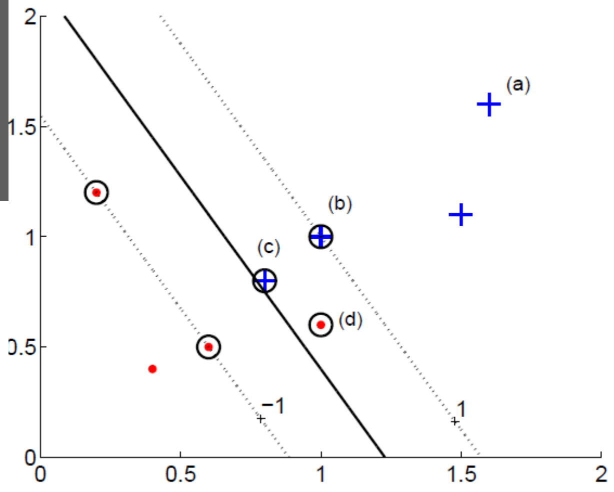
- Soft error $\sum_{i=1}^{N} \xi_i$

- New Optimization function, where C is must be set

$$L(\boldsymbol{w}, w_0, \alpha_i, \beta_i, \xi_i) = \frac{1}{2}\|\boldsymbol{w}\|_2^2 - \sum_{i=1}^{N} \alpha_i r_i \left((\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + w_0) - 1 + \xi_i \right)$$

$$+ C \sum_{i} \xi_i - \sum_{i=1}^{N} \beta_i \xi_i$$

Hinge Loss



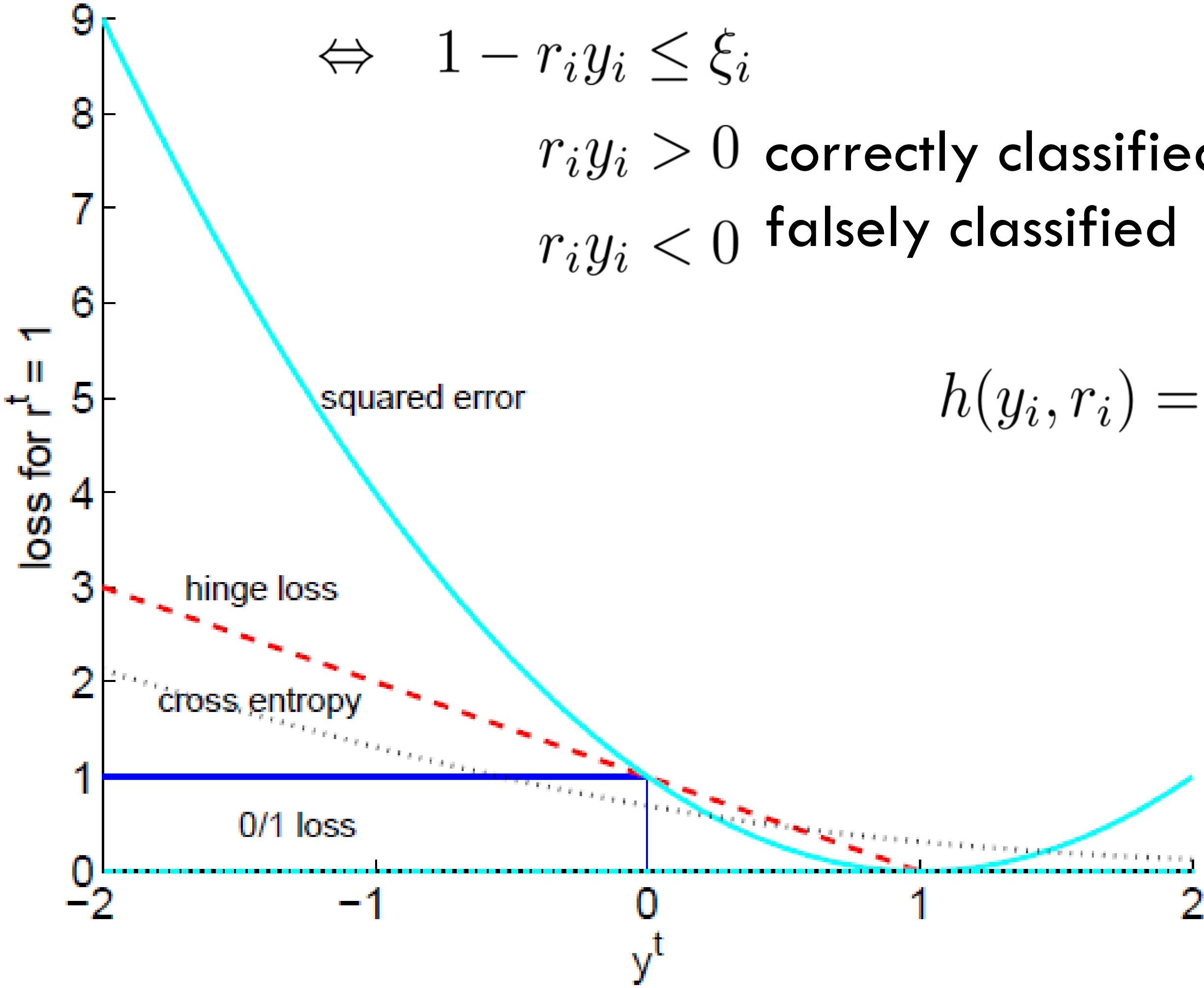
$$r_i \underbrace{\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + w_0\right)}_{=y_i} \geq 1 - \xi_i$$

$$\Leftrightarrow \quad 1 - r_i y_i \leq \xi_i$$

$r_i y_i > 0$ correctly classified

$r_i y_i < 0$ falsely classified

$$h(y_i, r_i) = \begin{cases} 0, & \text{if } y_i r_i \geq 1 \\ 1 - y_i r_i, & \text{otherwise} \end{cases}$$



Hinge loss is more robust than square error

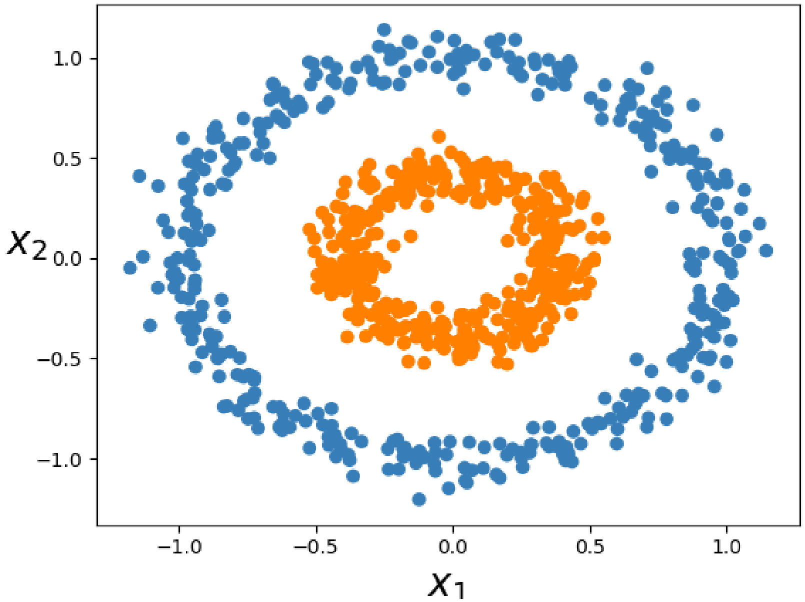
Kernel Trick

For non-linearly separable data:
transfer in higher dimensional space

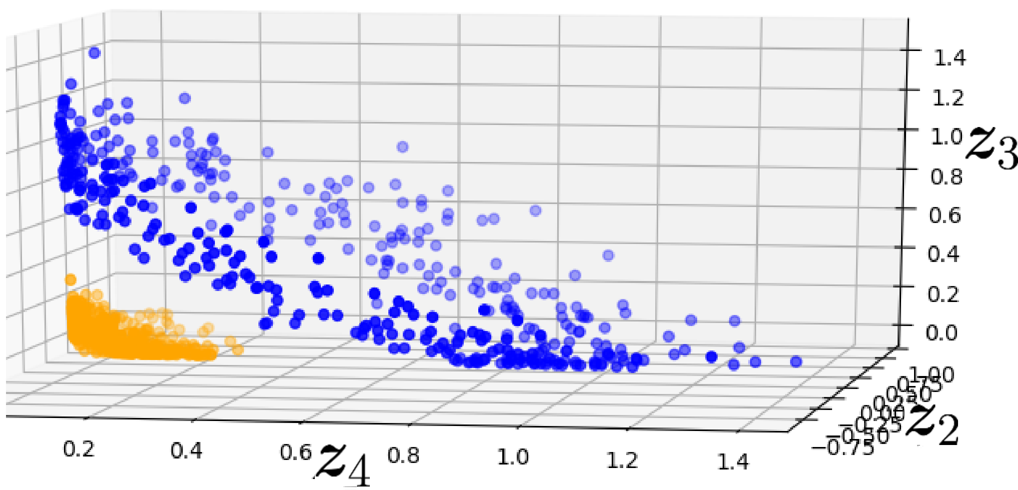
$$\mathbf{\Phi} : \mathbb{R}^2 \mapsto \mathbb{R}^4$$

$$\boldsymbol{z}_i = \mathbf{\Phi}(\boldsymbol{x}_i) = (1, x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

before



after



https://towardsdatascience.com/the-kernel-trick-c98cdbcaeb3f

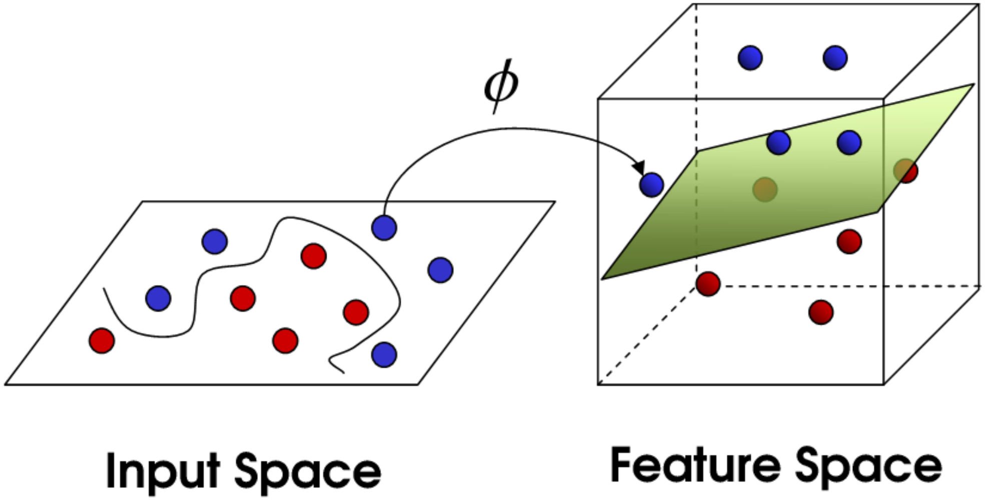
Kernel Trick

- Basis function as preprocessing

$$\mathbf{\Phi} : \mathbb{R}^D \to \mathbb{R}^L, D \ll L$$

$$\boldsymbol{z}_i = \mathbf{\Phi}(\boldsymbol{x}_i)$$

$$g(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\mathbf{\Phi}(\boldsymbol{x})$$



- Kernel function K(.,.)

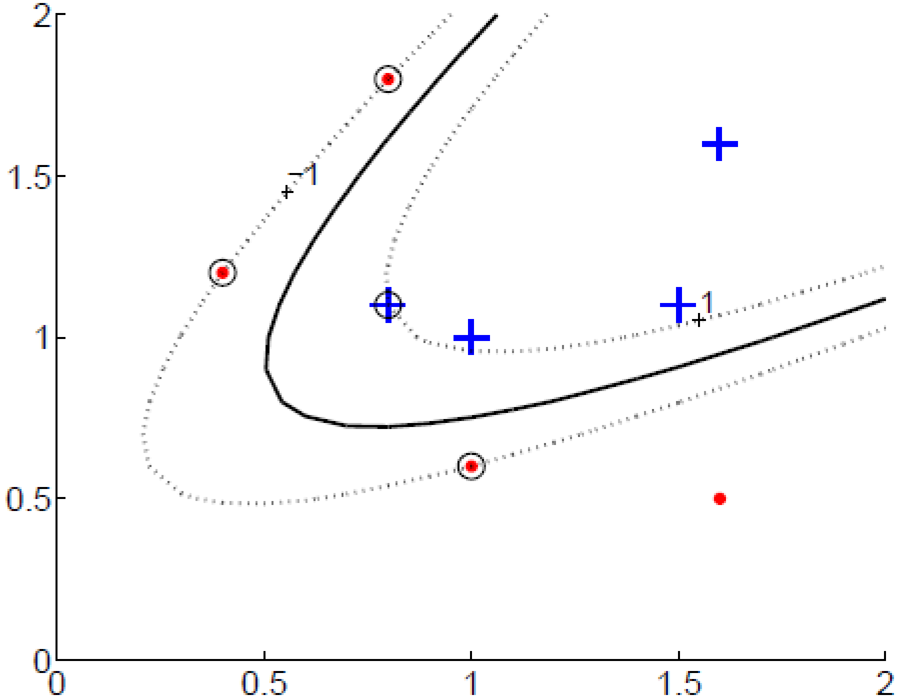
$$g(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\mathbf{\Phi}(\boldsymbol{x}) = \sum_i \alpha_i r_i \boxed{\mathbf{\Phi}(\boldsymbol{x}_i)^{\mathrm{T}}\mathbf{\Phi}(\boldsymbol{x})}$$

$$= \sum_i \alpha_i r_i \boxed{\boldsymbol{K}(\boldsymbol{x}_i, \boldsymbol{x})}$$

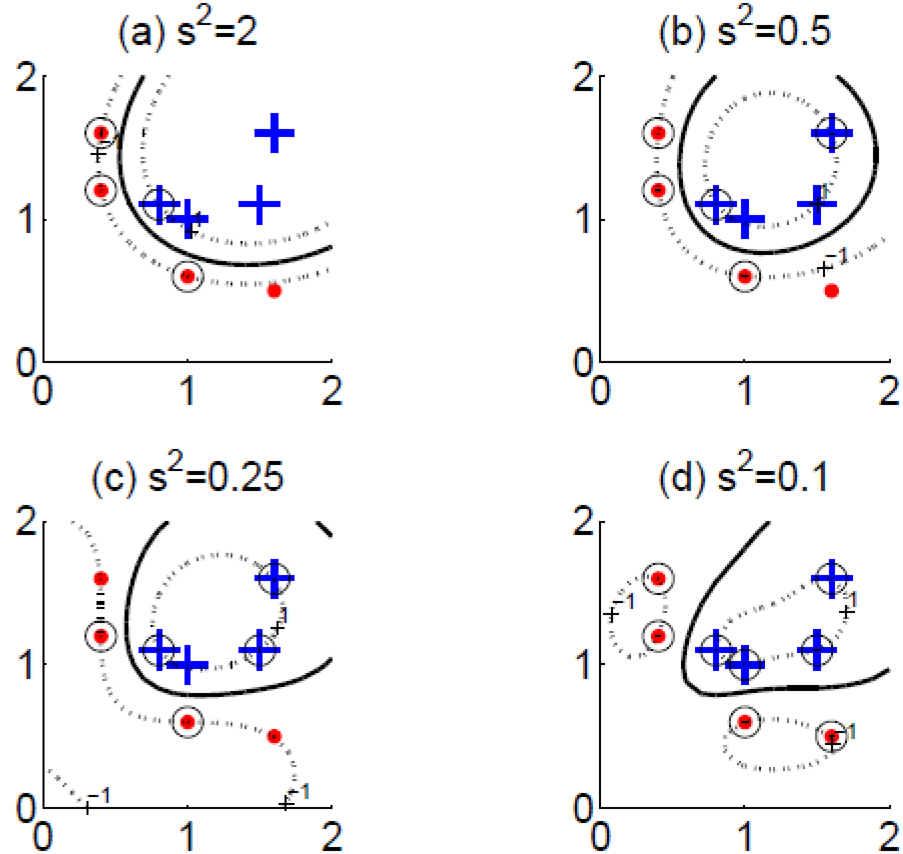
https://towardsdatascience.com/the-kernel-trick-c98cdbcaeb3f

Kernel: examples

Polynomial



Radial-basis functions

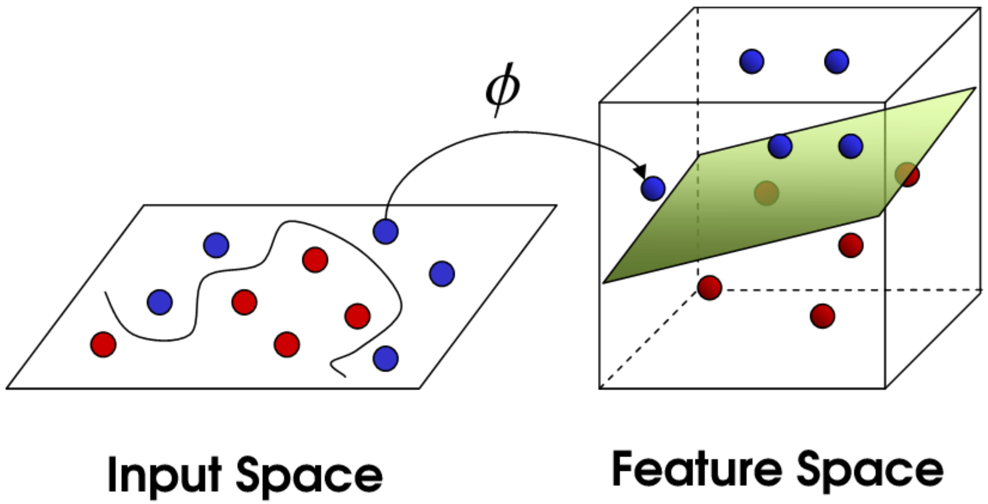


How to define a Kernel

$$\mathbf{\Phi} : \mathbb{R}^D \to \mathbb{R}^L, D \ll L$$

$$g(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}} \mathbf{\Phi}(\boldsymbol{x}) = \sum_i \alpha_i r_i \mathbf{\Phi}(\boldsymbol{x}_i)^{\mathrm{T}} \mathbf{\Phi}(\boldsymbol{x}) = \sum_i \alpha_i r_i \boldsymbol{K}(\boldsymbol{x}_i, \boldsymbol{x})$$

- Application dependent!
- Kernel function is a measure of similarity: the more similar the input, the higher: (e.g. p.d.f.)



https://towardsdatascience.com/the-kernel-trick-c98cdbcaeb3f

How to define a Kernel

$$\mathbf{\Phi} : \mathbb{R}^D \rightarrow \mathbb{R}^L, D \ll L$$

$$g(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}} \mathbf{\Phi}(\boldsymbol{x}) = \sum_i \alpha_i r_i \mathbf{\Phi}(\boldsymbol{x}_i)^{\mathrm{T}} \mathbf{\Phi}(\boldsymbol{x}) = \sum_i \alpha_i r_i \boldsymbol{K}(\boldsymbol{x}_i, \boldsymbol{x})$$

- Kernel function = measure of similarity
- Empirical kernel map:
  Define a set of templates $\boldsymbol{m}_i$ and score function s

$$\mathbf{\Phi}(\boldsymbol{x}) = [s(\boldsymbol{x}, \boldsymbol{m}_1), \ldots, s(\boldsymbol{x}, \boldsymbol{m}_M)]$$

- Combine kernels

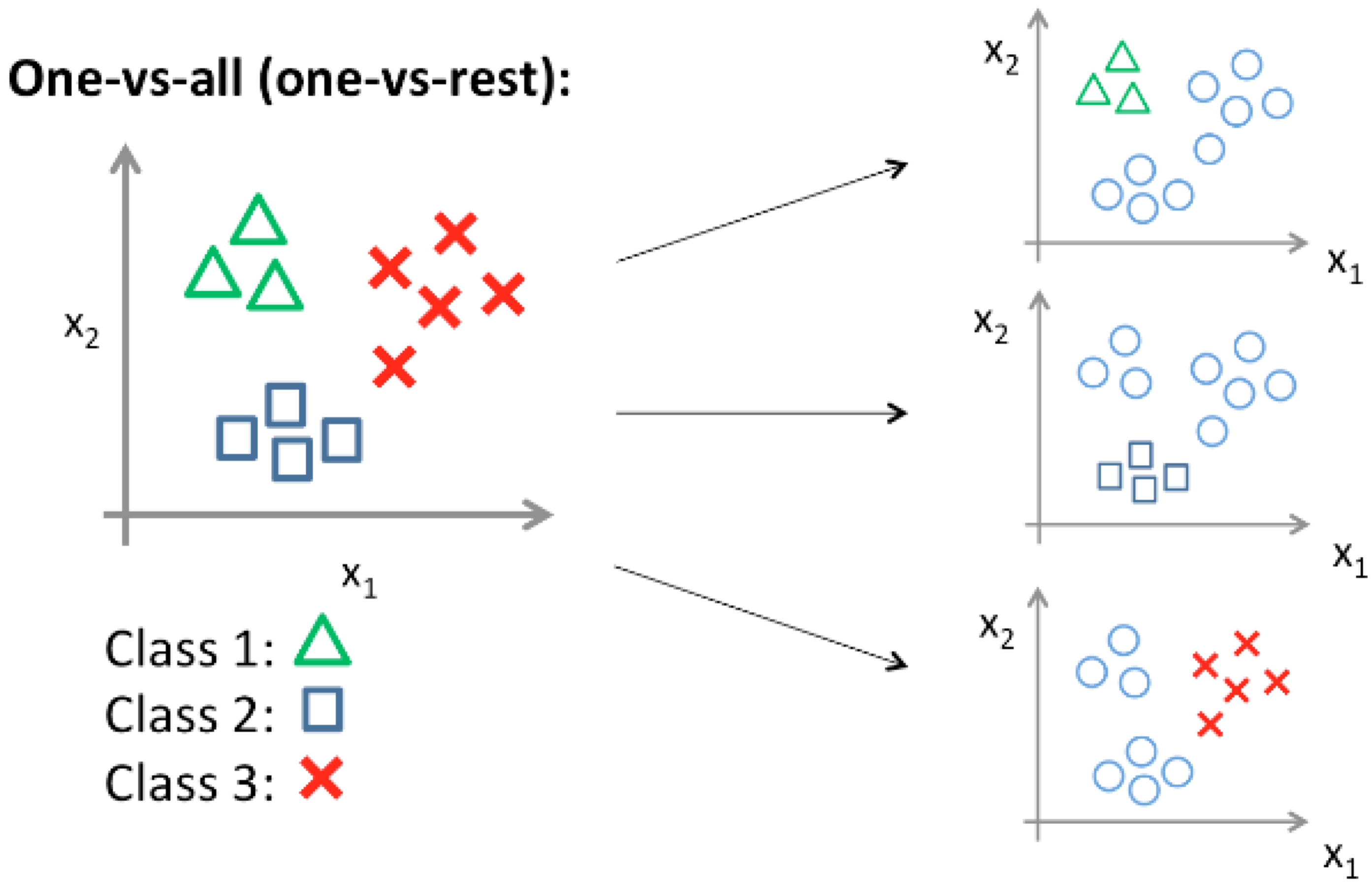
$$K(\boldsymbol{x}, \boldsymbol{x}_i) = \sum_n \lambda_n K_n(\boldsymbol{x}, \boldsymbol{x}_i)$$

- Combination of two different representations, e.g. image and sound

Multiclass Kernel Machines

Consider K classes

- One-vs-all: K subproblems

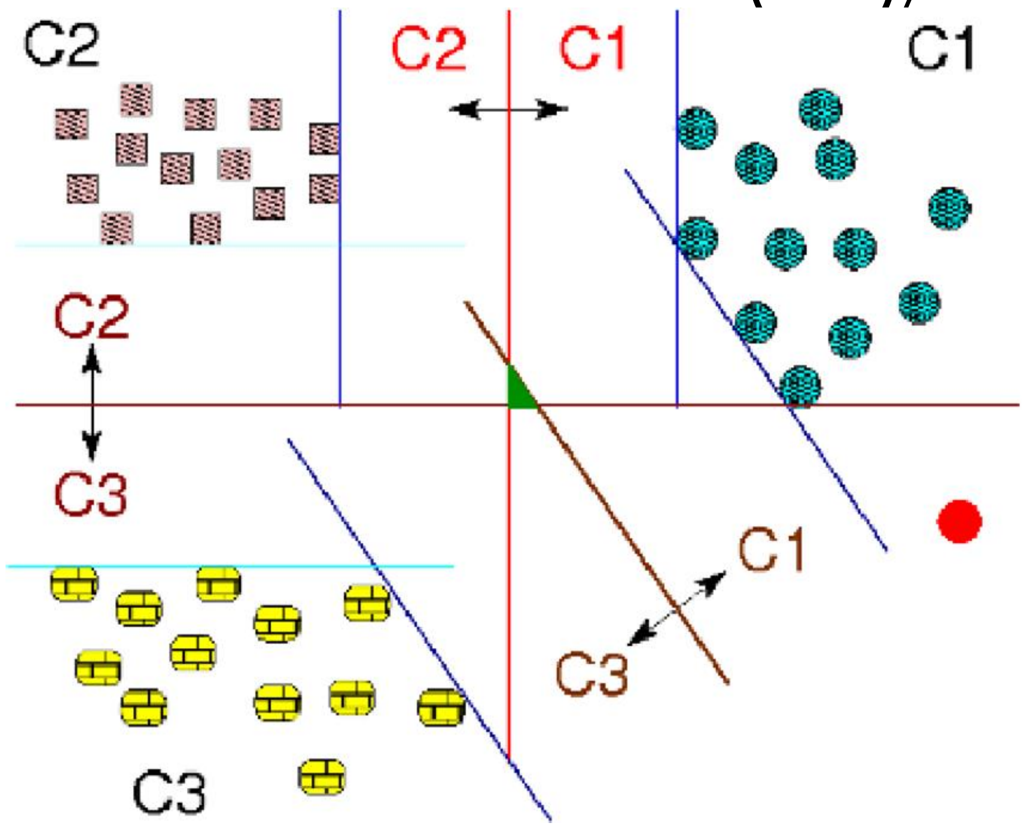


Multiclass Kernel Machines

Consider K classes

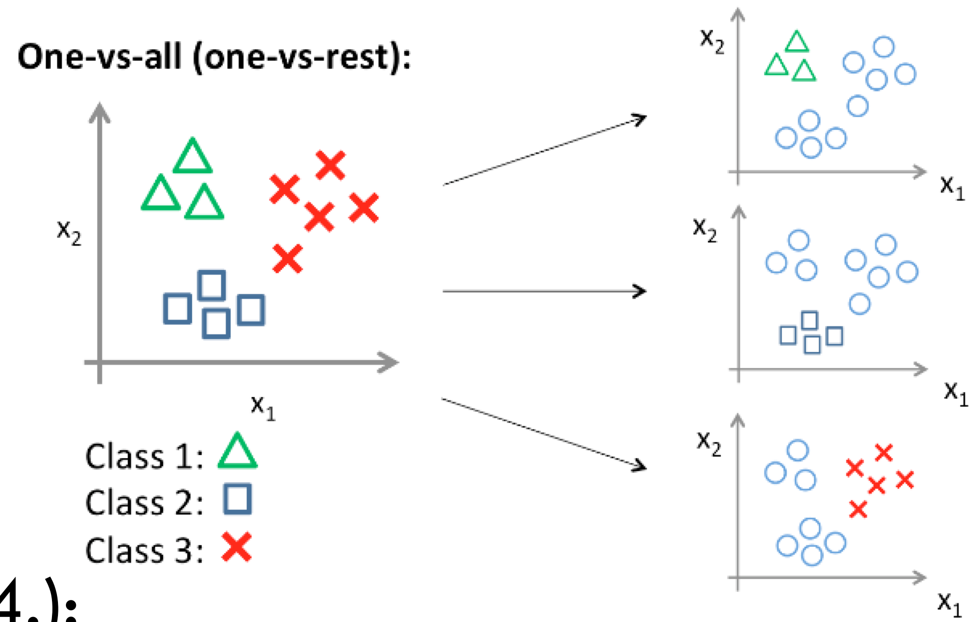
- Pairwise separation (10.4.):

  focus on two classes at a time: K(K-1)/2



https://www.researchgate.net/figure/Illustration-of-the-SVM-principle-and-of-the-one-versus-one-multiclass-classification_fig2_220098164

Multiclass Kernel Machines



Consider K classes

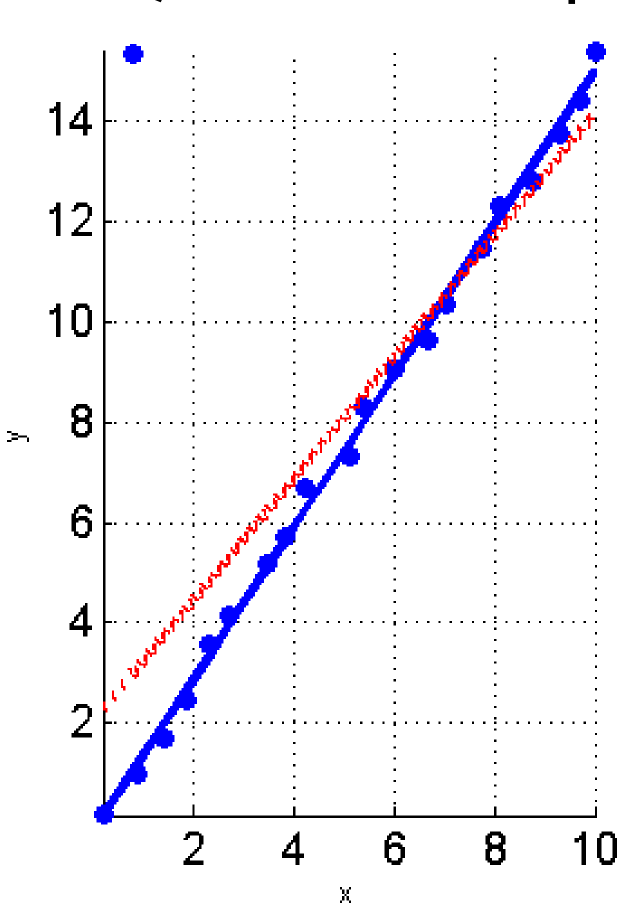
- 1-vs-all
  define K problems
- Pairwise separation (10.4.):
  focus on two classes at a time: K(K-1)/2
- Error-Correcting Output Codes (Sec. 17.5)
  consider sets of 2 classes
- Single multiclass optimization
  involves all classes

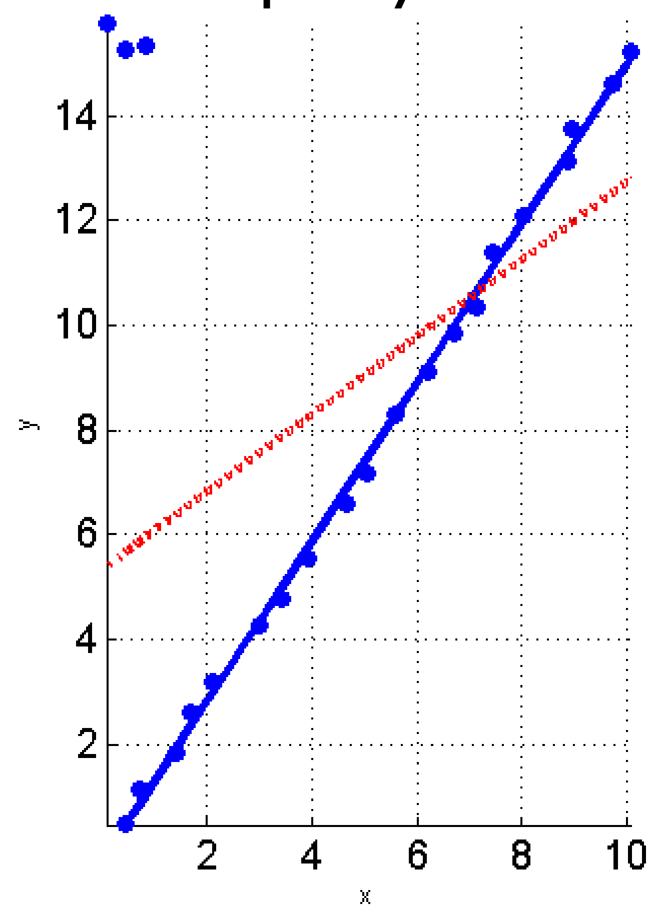
Regression

Fit a line / hyperplane through the data

previous problem:

outliers, because all points contribute equally

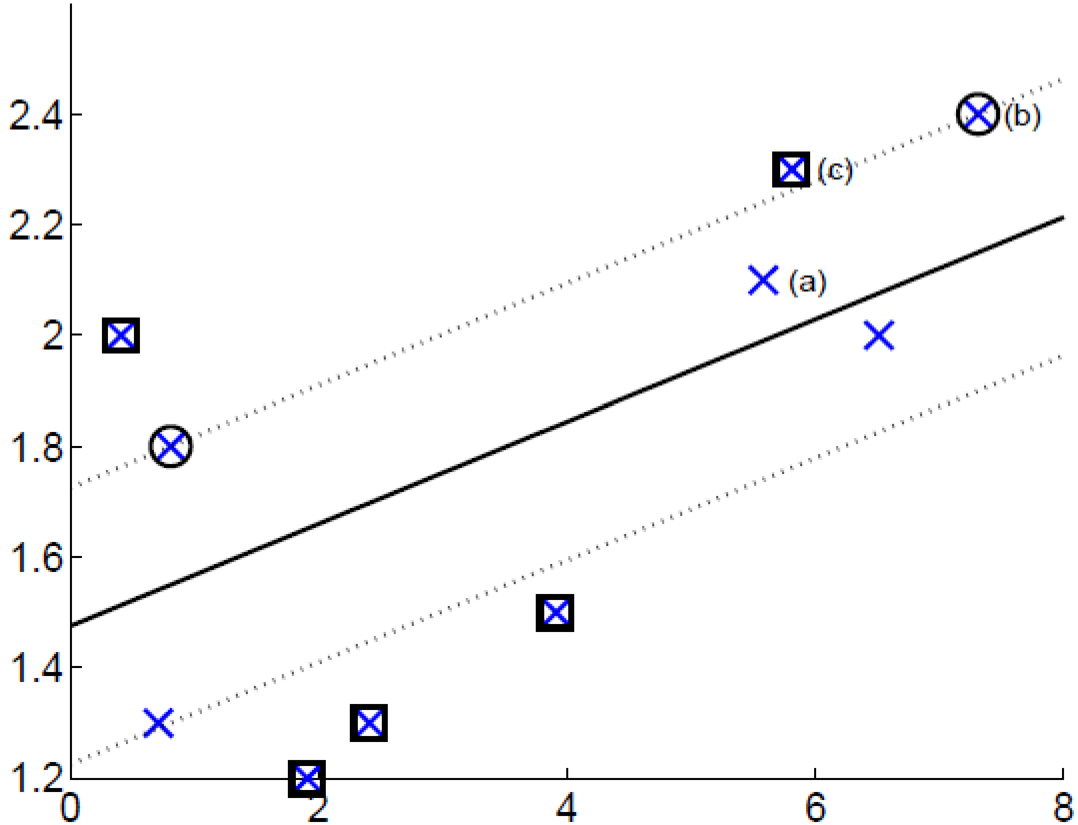




SVM for Regression

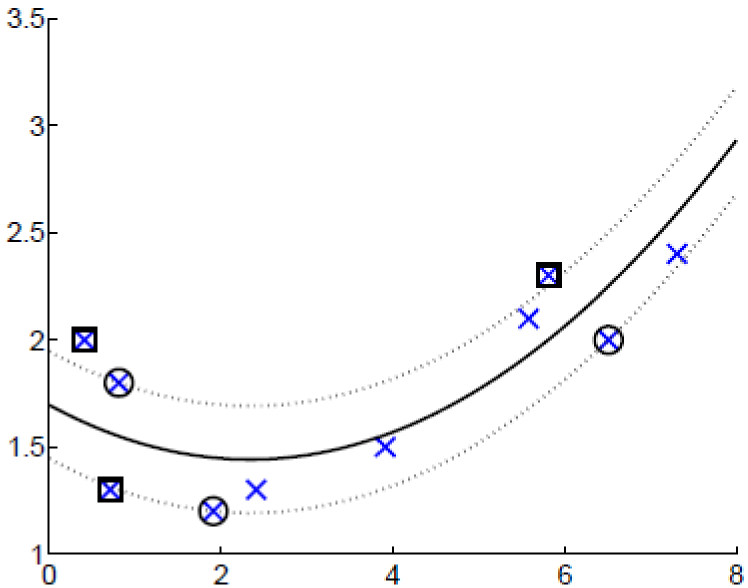
Fitted line is weighted sum of support vectors: □ ○

$$f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + w_0 = \sum_i (\alpha_{i+} - \alpha_{i-})\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x} + w_0$$

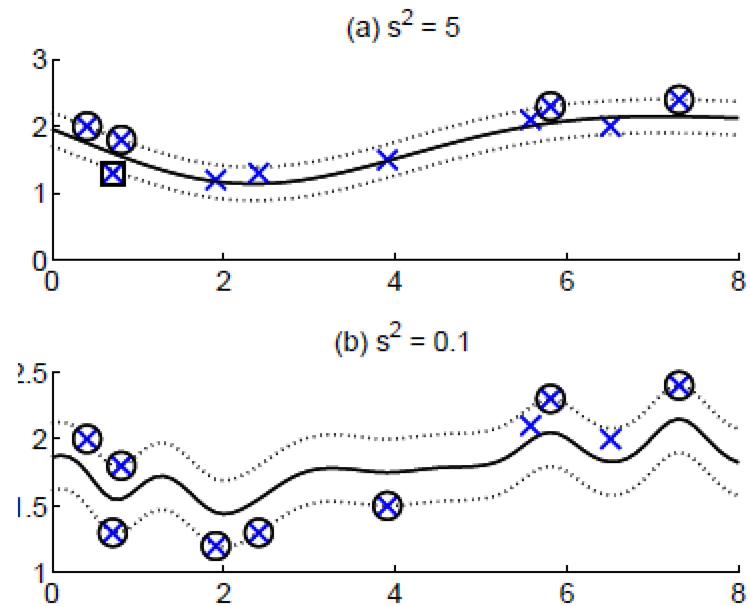


Kernel Regression

- Polynomial kernel



- Gaussian kernel



support vectors

○ inside of tube

□ outside of tube

36

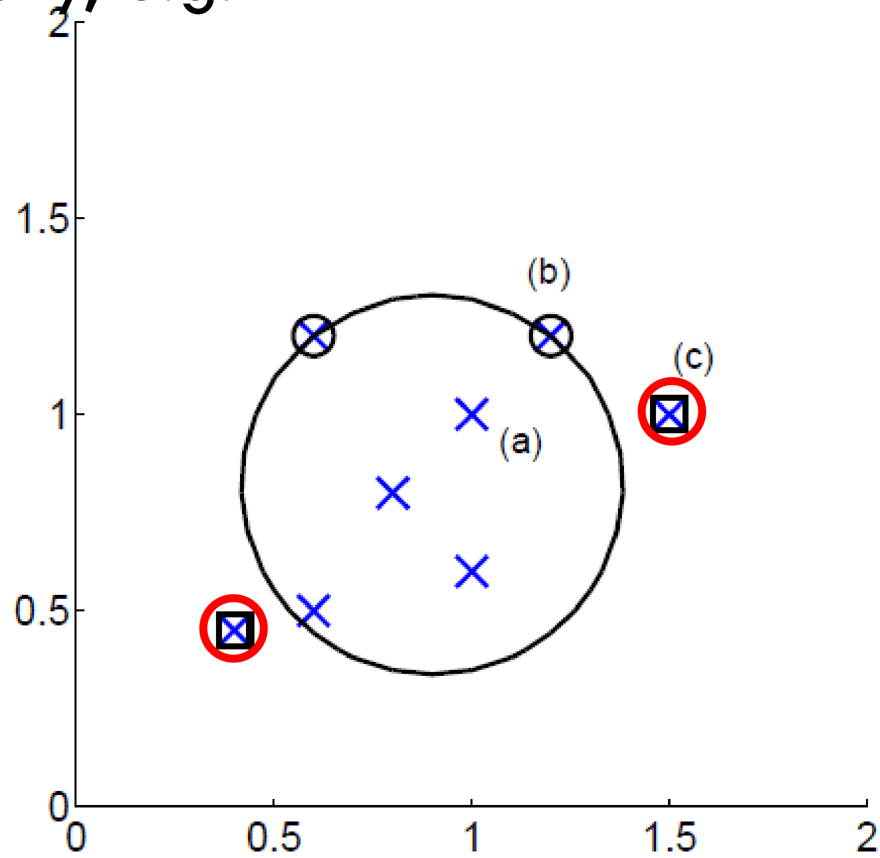
One-Class Kernel Machines

Unsupervised learning:

- Estimate regions of high density, e.g. to define outliers
- Smoothest boundary, enclosing as many points as possible
- Consider a sphere with center ***a*** and radius *R*

$$\min R^2 + C \sum_i \xi_i$$

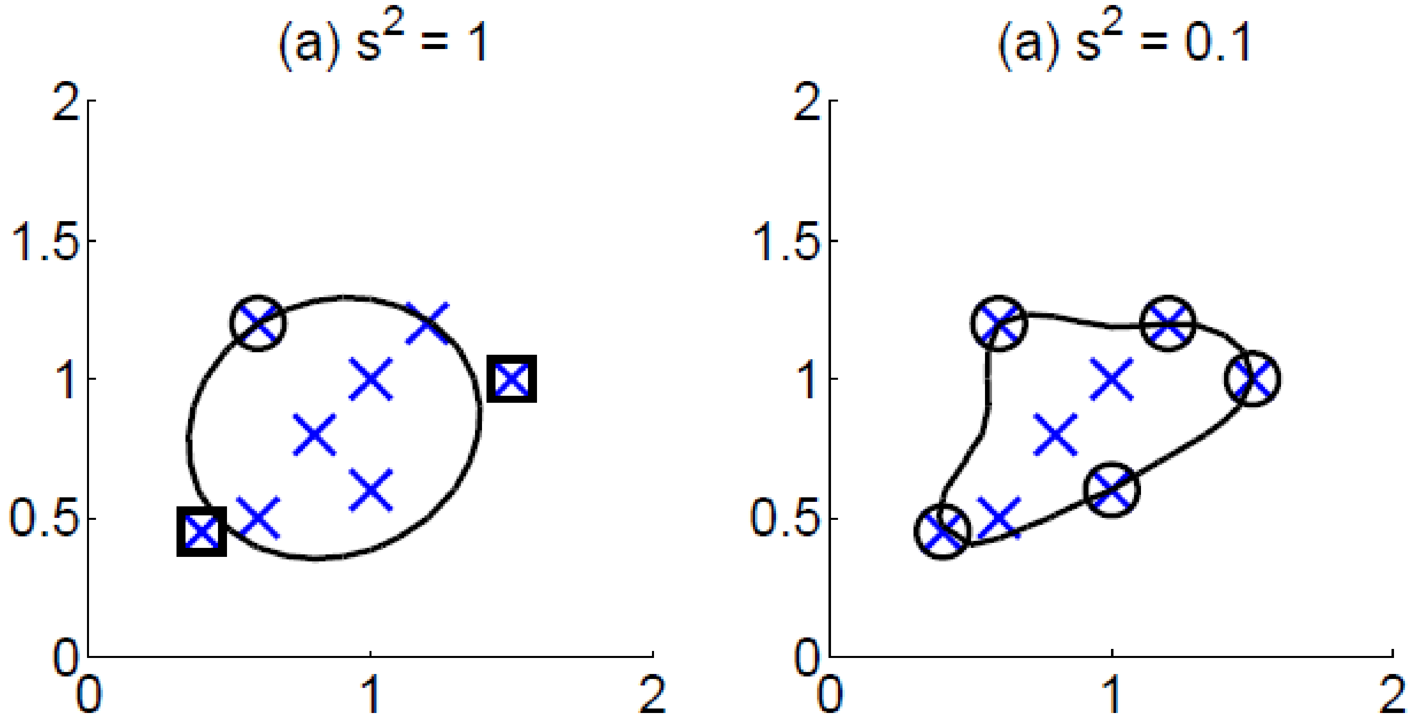
$$\|\boldsymbol{x}_i - \boldsymbol{a}\|_2^2 \leq R^2 + \xi_i$$



Result of linear kernel

One-Class Kernel Machines

Gaussian Kernel

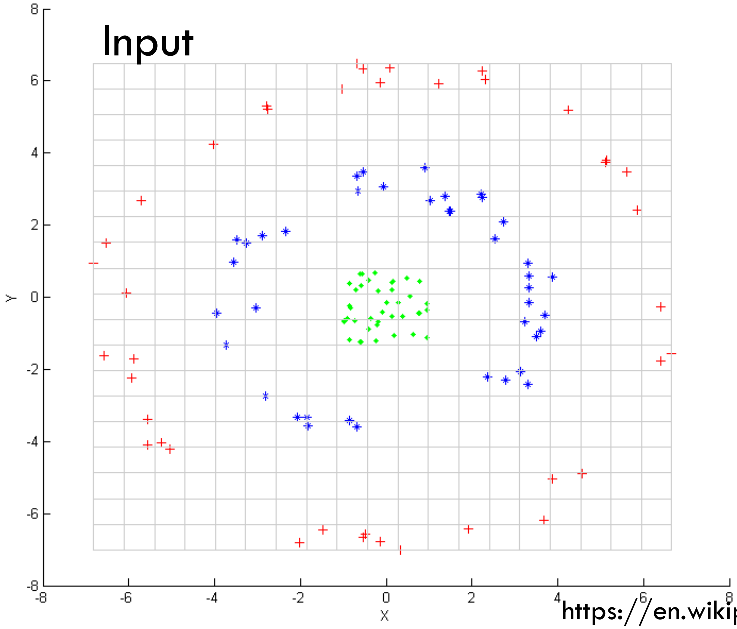


Kernel Dimensionality Reduction

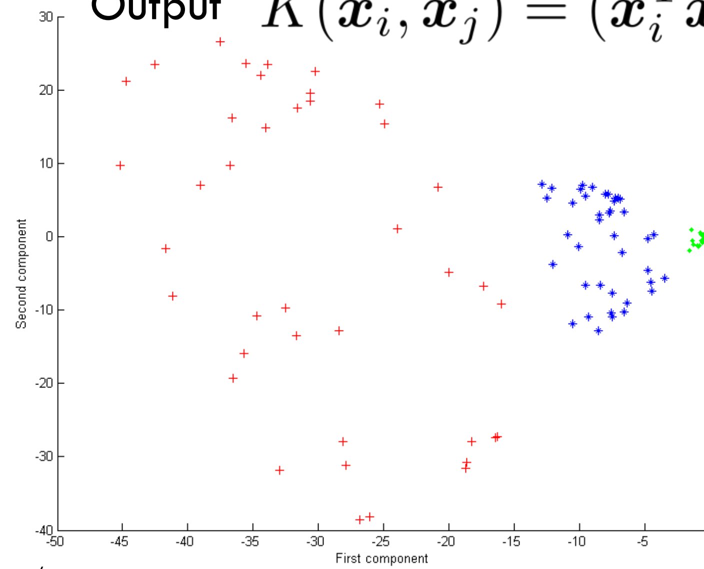
- **Kernel PCA**
  - Previously: PCA on data covariance matrix $\widehat{\boldsymbol{\Sigma}} = \boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$
  - NOW PCA on kernel matrix : NxN $\widehat{\boldsymbol{\Sigma}} = (K(\boldsymbol{x}_i, \boldsymbol{x}_j))_{i,j=1}^{N}$

$$\widehat{\boldsymbol{\Sigma}}\boldsymbol{w}_k = \lambda_k \boldsymbol{w}_k, \;\; \boldsymbol{w}_i^{\mathrm{T}}\boldsymbol{w}_j = \begin{cases} 1 & , i = j \\ 0 & , i \neq j \end{cases}$$

- also: Kernel LDA, CCA



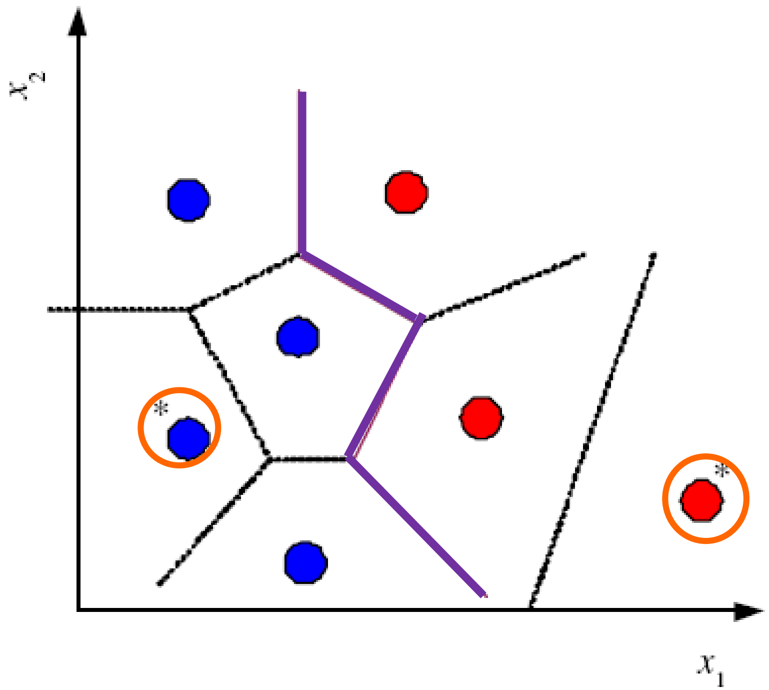
Output $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j + 1)^2$



https://en.wikipedia.org/wiki/Kernel_principal_component_analysis

Summary: Kernel Machines

- Many „kernelized“ methods
- Kernelized: if x cannot be linearly separated, use basis function $\Phi(x)$ or kernel function $K(x, x_i)$
- Various applications:
  - Classification
  - Regression
  - Dimensionality reduction
  - Outlier detection
- Few training points suffice for definition of hyperplane
- More robust to outliers



APPENDIX