Course : CM-072

Presentation 5

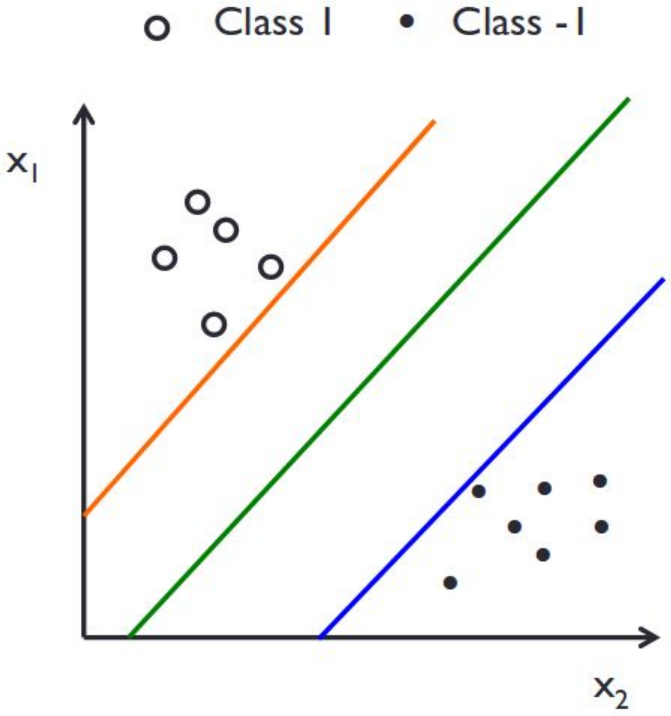
Support Vector Machine: history

- SVMs introduced in COLT-92 by Boser, Guyon & Vapnik. Became rather popular since.
- Theoretically well motivated algorithm: developed from Statistical Learning Theory (Vapnik & Chervonenkis) since the 60s.
- Empirically good performance: successful applications in many fields (bioinformatics, text, image recognition, ...)
- A large and diverse community work on them: from machine learning, optimization, statistics, neural networks, functional analysis, etc.

Support Vector Machine: 3 key ideas

- Use **optimization** to find solution (i.e. a hyperplane) with few errors
- Seek **large margin** separator to improve generalization
- Use **kernel trick** to make large feature spaces computationally efficient.

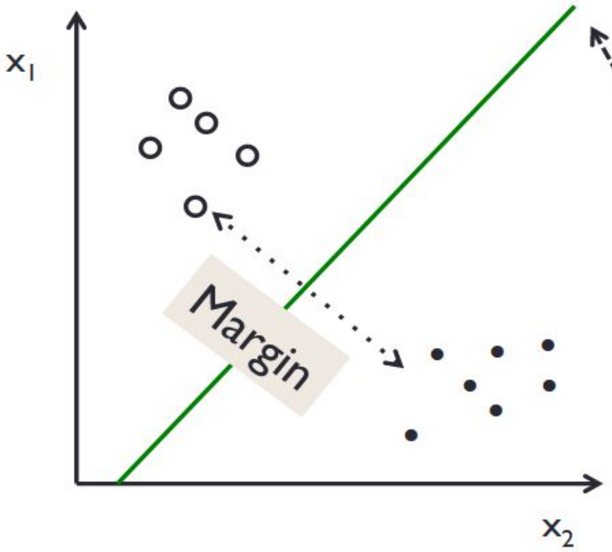
Support Vector Machine



Many possible lines..

Which one is the best?

Support Vector Machine



Many possible lines..

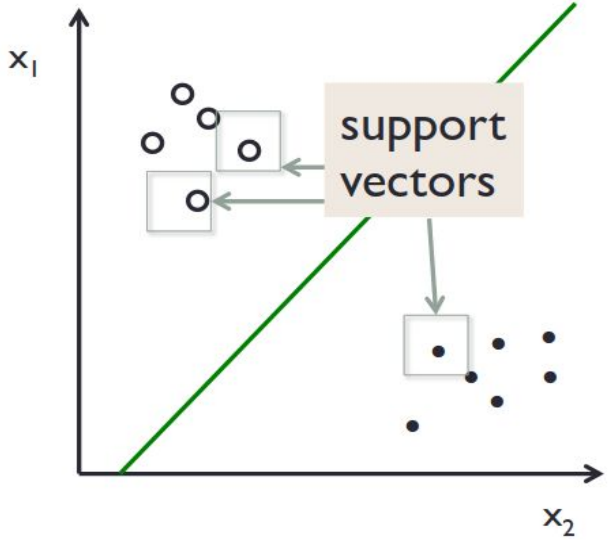
Which one is the best?

Intuitively the one far away from the data points…

for which the margin is maximized.

Margin: defined by the distance of the closest data point from the decision boundary.

Support Vector Machine

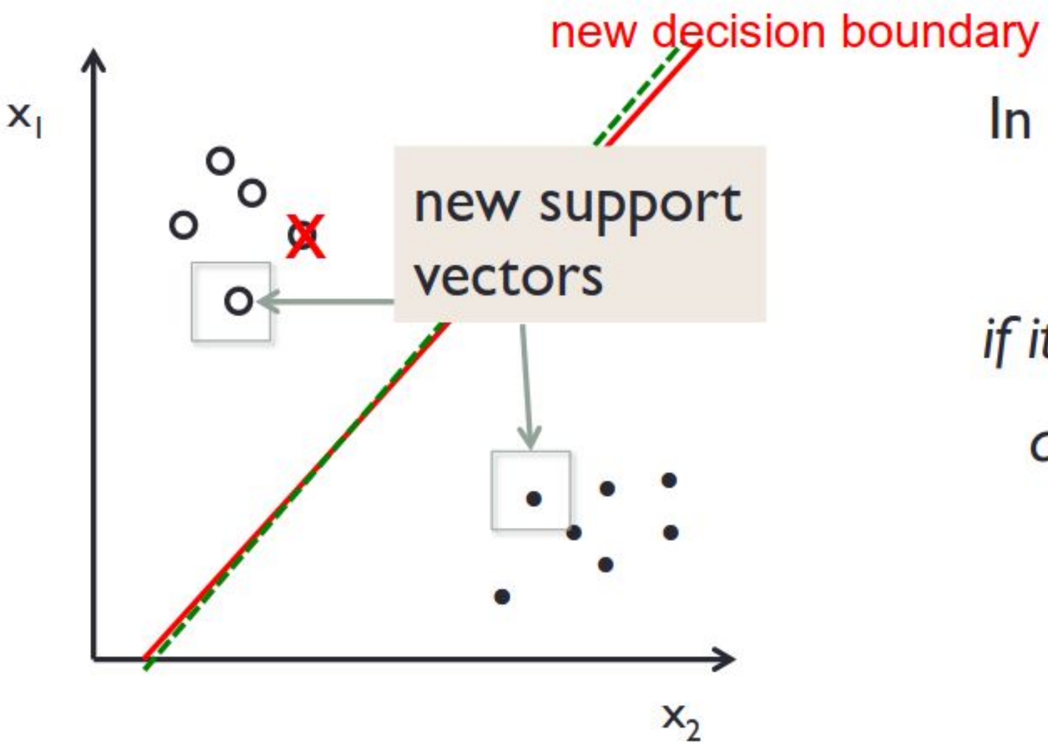


The data points (vectors) that define the margin are called support vectors.

They form a small subset (2 or 3 points in a 2 dimensional space) of the dataset used to define the decision boundary.

*The rest of the data points do not participate in the decision.

Support Vector Machine



In other words, a point is a

support vector

if its removal from the dataset

changes the decision line.

Support Vector Machine

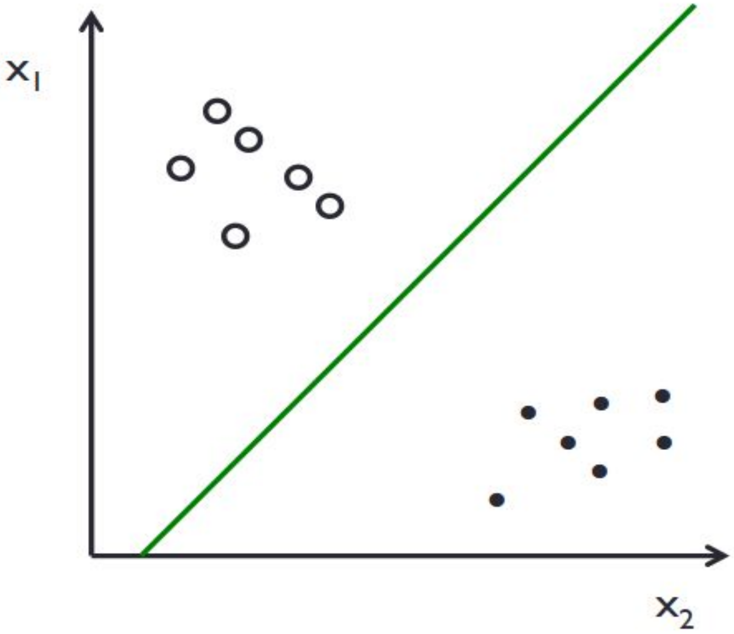
Finding support vectors graphically

Example: 2 points (en face)

Example: 3 points (• closest to the line defined by o in the convex hull)

Convex hull (grey line),
support vector (grey square),
decision boundary (green line),
gutter (dashed line)

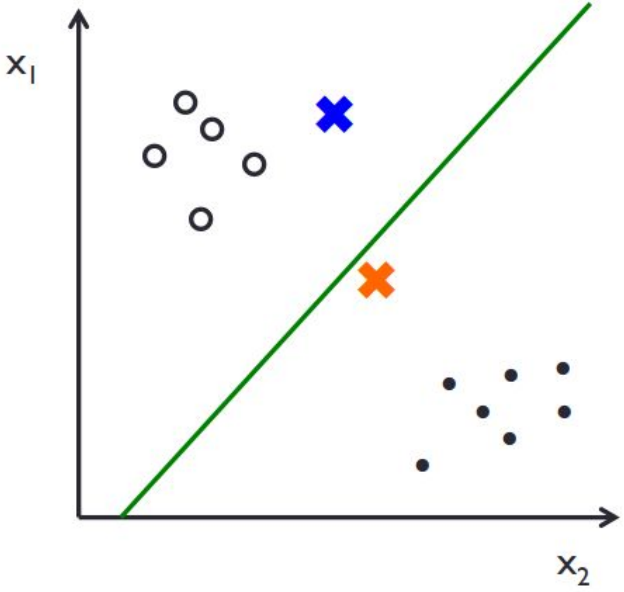
Support Vector Machine



Classification decision:

The furthest the point is from the decision boundary the more confident we are about the class assignment.

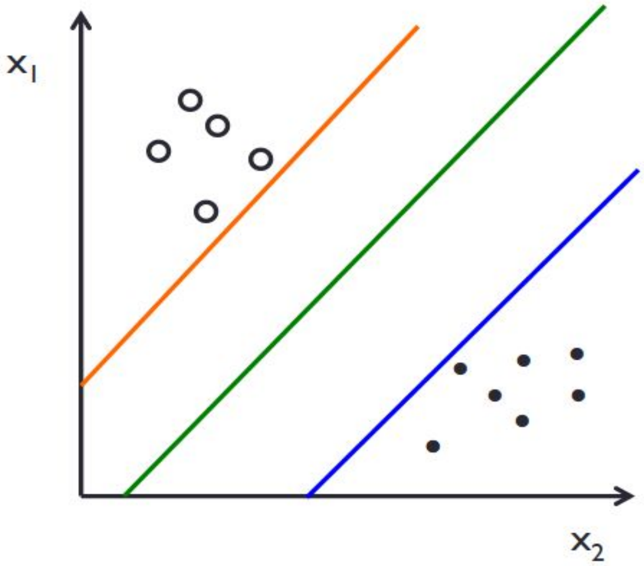
Support Vector Machine



The furthest the point is from the decision boundary the more confident we are about the class assignment.

We are more confident for the class assignment for ✖ than for ✖ .

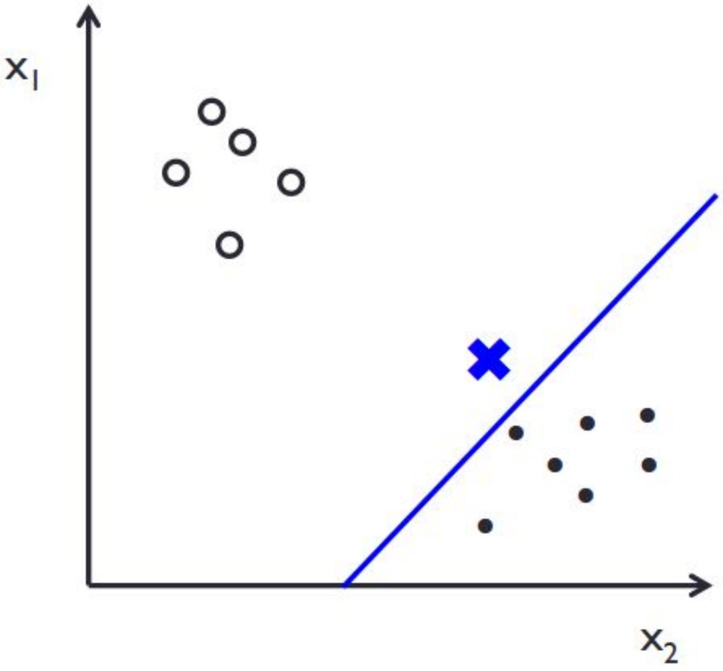
Support Vector Machine



So, picking the green line (by the SVM classifier) gives a classification safety margin: a slight error in the measurement will not affect the result.

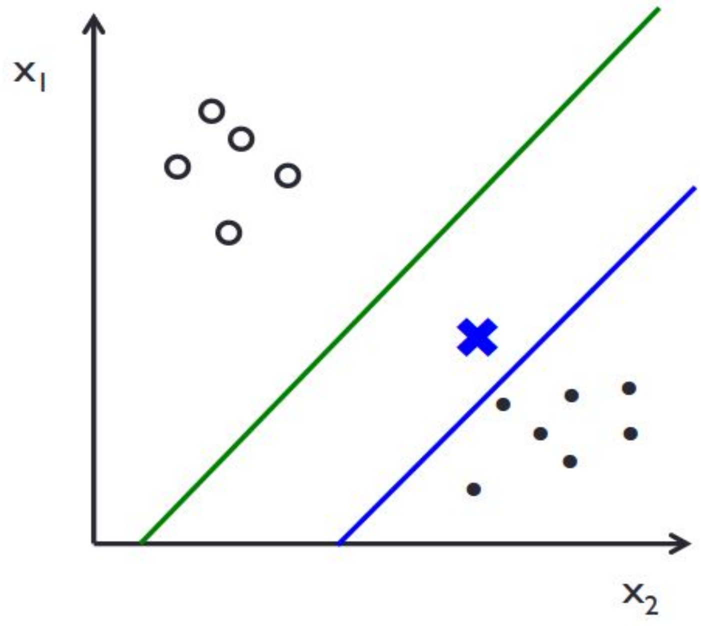
Larger margin classifier →
Better generalization ability and noise-tolerant.

Support Vector Machine



For example, if we had chosen the blue line, ✖ would have been assigned to class 1(∘)

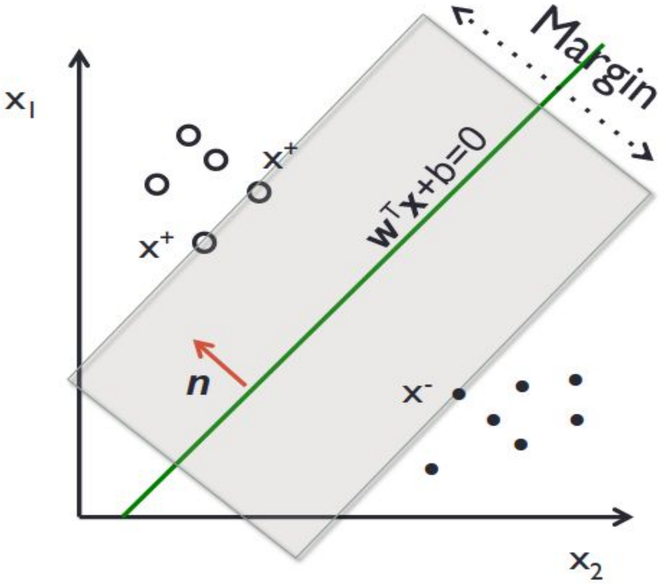
Support Vector Machine



For example, if we had chosen the blue line, ✖ would have been assigned to class 1 (∘) whereas with the green line it is correctly classified in class -1 (•)

Support Vector Machine

Learning phase



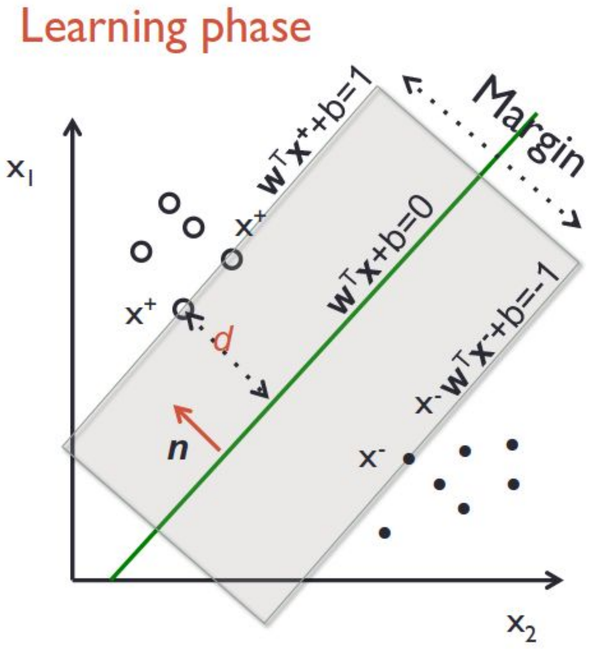
We search for w (weight vector) and b (intercept) to define the decision boundary. For the decision hyperplane it holds that (*assumption 1*):

$$\mathbf{w}^T\mathbf{x} + b = 0$$

$$\mathbf{n} = \frac{\mathbf{w}}{\|\mathbf{w}\|}$$

Unit-length normal vector of the hyperplane

Support Vector Machine



$$\mathbf{n} = \frac{\mathbf{w}}{\|\mathbf{w}\|}$$ Unit-length normal vector of the hyperplane

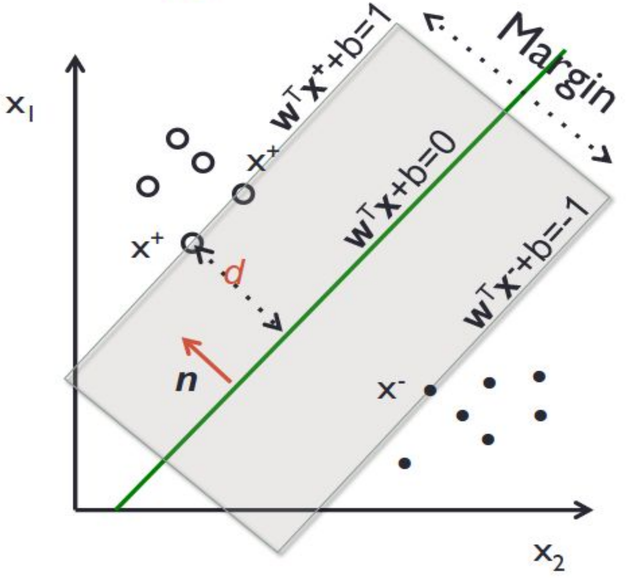
- For the support vectors $\mathbf{x}^+$ and $\mathbf{x}^-$ it holds that (*assumption* 2)

$$\mathbf{w}^T\mathbf{x}^+ + b = 1$$

$$\mathbf{w}^T\mathbf{x}^- + b = -1$$

Support Vector Machine

Learning phase



$$\mathbf{n} = \frac{\mathbf{w}}{\|\mathbf{w}\|}$$ Unit-length normal vector of the hyperplane

- For the support vectors $\mathbf{x}^+$ and $\mathbf{x}^-$ it holds that (*assumption* 2)

$$\mathbf{w}^T\mathbf{x}^+ + b = 1$$

$$\mathbf{w}^T\mathbf{x}^- + b = -1$$

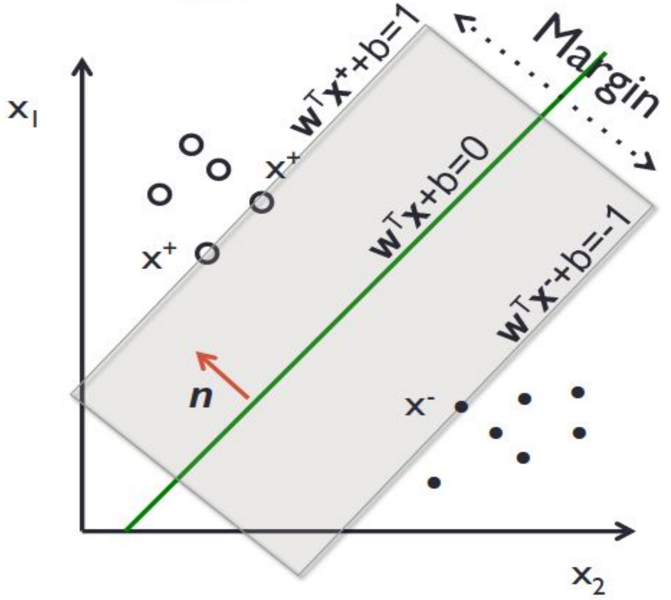
- Then, for the margin width it holds that

$$\boxed{M = 2d = 2\frac{(\mathbf{w}^T\mathbf{x}^+ + b)}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|}}$$

(*d* is the distance of a support vector from the decision boundary)

Support Vector Machine

Learning phase



Optimization Problem:

Maximize margin width

$$\frac{2}{\|\mathbf{w}\|}$$

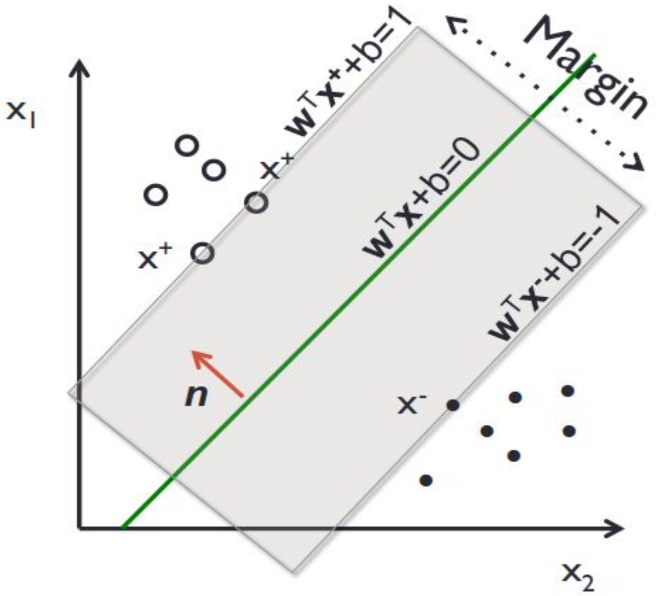
such that

For $y_i = +1$, $\mathbf{w}^T \mathbf{x}_i + b \geq 1$

For $y_i = -1$, $\mathbf{w}^T \mathbf{x}_i + b \leq -1$

Support Vector Machine

Learning phase



Optimization Problem (alternatively):

Minimize

$$\frac{1}{2}\|\mathbf{w}\|^2$$

such that

$$y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1$$

$$y_i \in \{1, -1\}$$

Support Vector Machine

Learning phase

Quadratic programming with linear constraints

Minimize

$$\frac{1}{2}\|\mathbf{w}\|^2$$

such that

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 \geq 0$$

Lagrangian Function

$$\text{minimize } L_p(\mathbf{w}, b, \alpha_i) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 \right)$$

s.t. $\alpha_i \geq 0$ ← *Lagrangian multipliers*

Support Vector Machine

Learning phase

$$\text{minimize} \quad L_p(\mathbf{w}, b, \alpha_i) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 \right)$$

$$\text{s.t.} \quad \alpha_i \geq 0$$

$$\frac{\partial L_p}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

$$\frac{\partial L_p}{\partial b} = 0 \Rightarrow \sum_{i=1}^{n} \alpha_i y_i = 0$$

Support Vector Machine

Learning phase

$$\text{minimize } L_p(\mathbf{w}, b, \alpha_i) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 \right)$$

$$\text{s.t.} \quad \alpha_i \geq 0$$

Lagrangian Dual Problem

$$\text{maximize } \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \boxed{\mathbf{x}_i^T \mathbf{x}_j}$$

$$\text{s.t.} \quad \alpha_i \geq 0 \text{ , and } \sum_{i=1}^{n} \alpha_i y_i = 0$$

Property of α_i when we introduce the lagrange multipliers.

The result when we differentiate the original Lagrangian w.r.t. b.

Support Vector Machine

Learning phase

- From KKT* conditions, we know:

$$\alpha_i[y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1] = 0$$

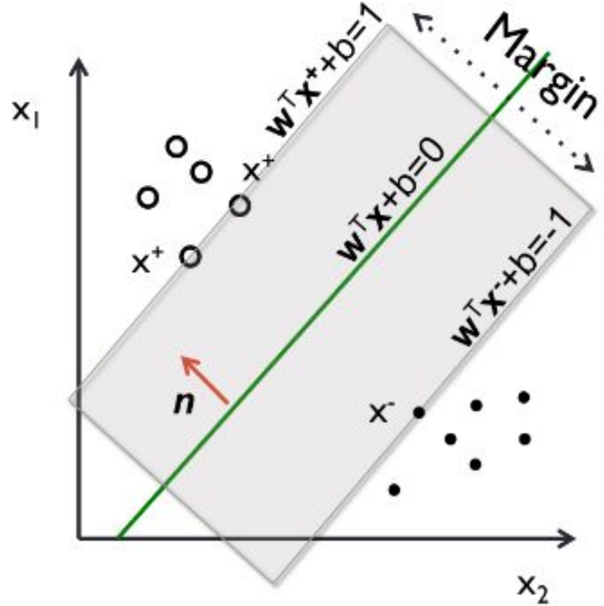
- Thus, only support vectors can have $\alpha_i \neq 0$

- The solution has the form:

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i = \sum_{i \in SV} \alpha_i y_i \mathbf{x}_i$$

$$b = y_i - \mathbf{w}^T x_i$$

where $\mathbf{x}_i$ is a support vector



Support Vector Machine

Testing phase

- The linear discriminant function is:

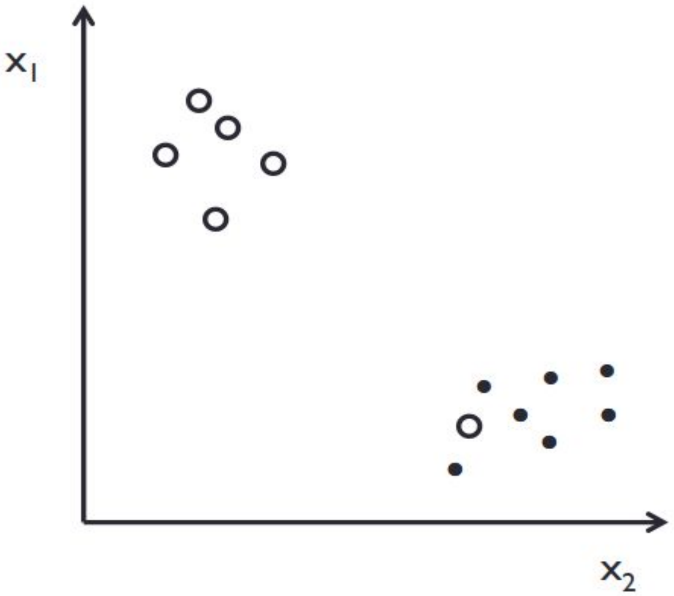
$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = \sum_{i \in SV} \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b$$

- Classification rule for point $\mathbf{x}$: $class(\mathbf{x}) = sign(g(\mathbf{x}))$

- Note: it relies on a *dot product* between the test point $\mathbf{x}$ and the support vectors $\mathbf{x}_i$

- Also keep in mind that solving the optimization problem involved computing the dot products $\mathbf{x}_i^T\mathbf{x}_j$ between all pairs of training points

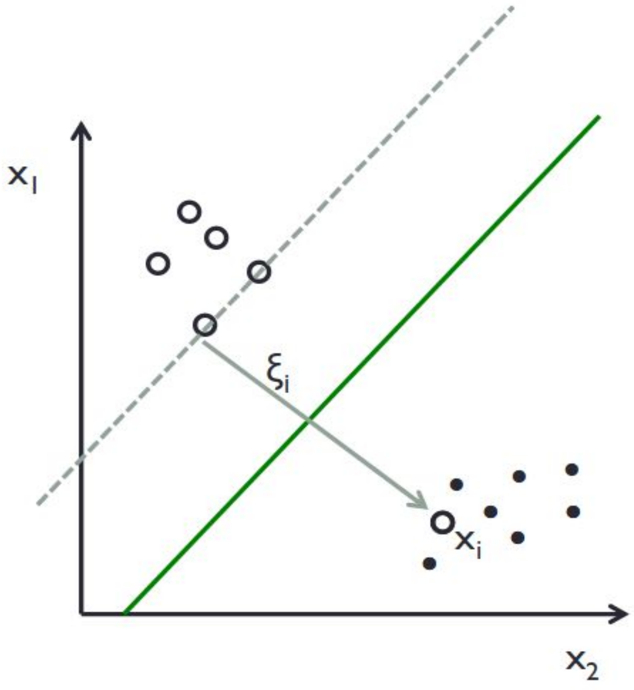
Soft margin classification



What if a circle point exists among the dots?

Then the data are not linearly separable!

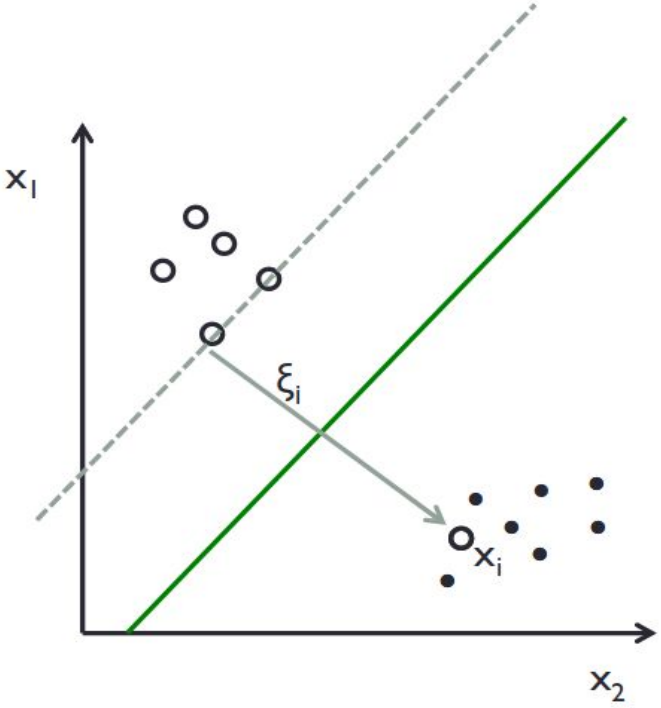
Soft margin classification



Allow for small classification errors by introducing slack variables ξ_i.

A non-zero value for ξ_i allows x_i to not meet the margin requirement at a cost proportional to the value of ξ_i.

Soft margin classification



Optimization problem

$$\min_{\mathbf{w},\xi_i} \|\mathbf{w}\|^2 + C\sum_{i=1}^{N} \xi_i$$

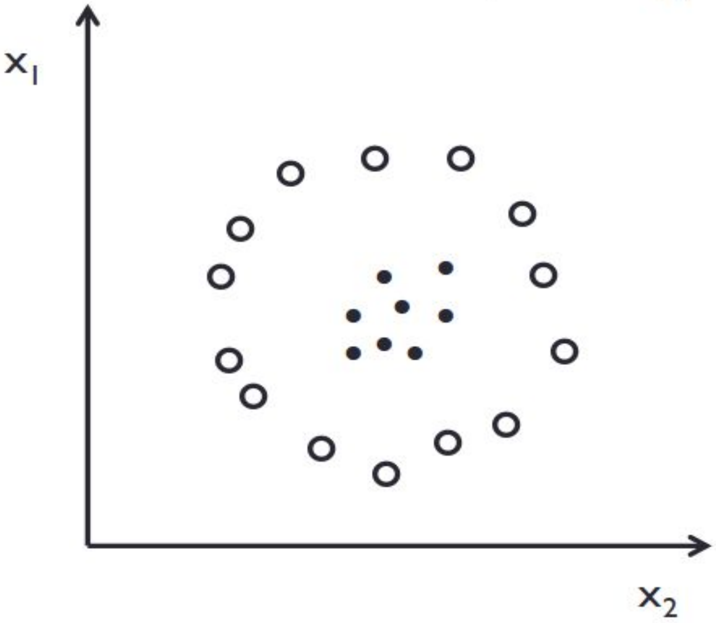
subject to

$$y_i(\mathbf{w}^{\mathrm{T}} x_i + b) \geq 1 - \xi_i \text{ for } i = 1 \dots N$$

C is the penalty that we pay
for each error.

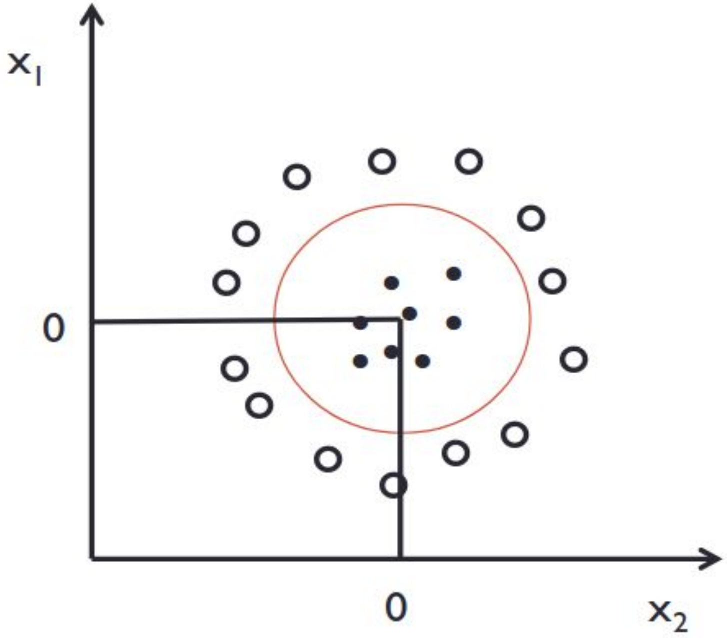
What about non-linearly separable data?

Here, a straight line cannot separate well the data!

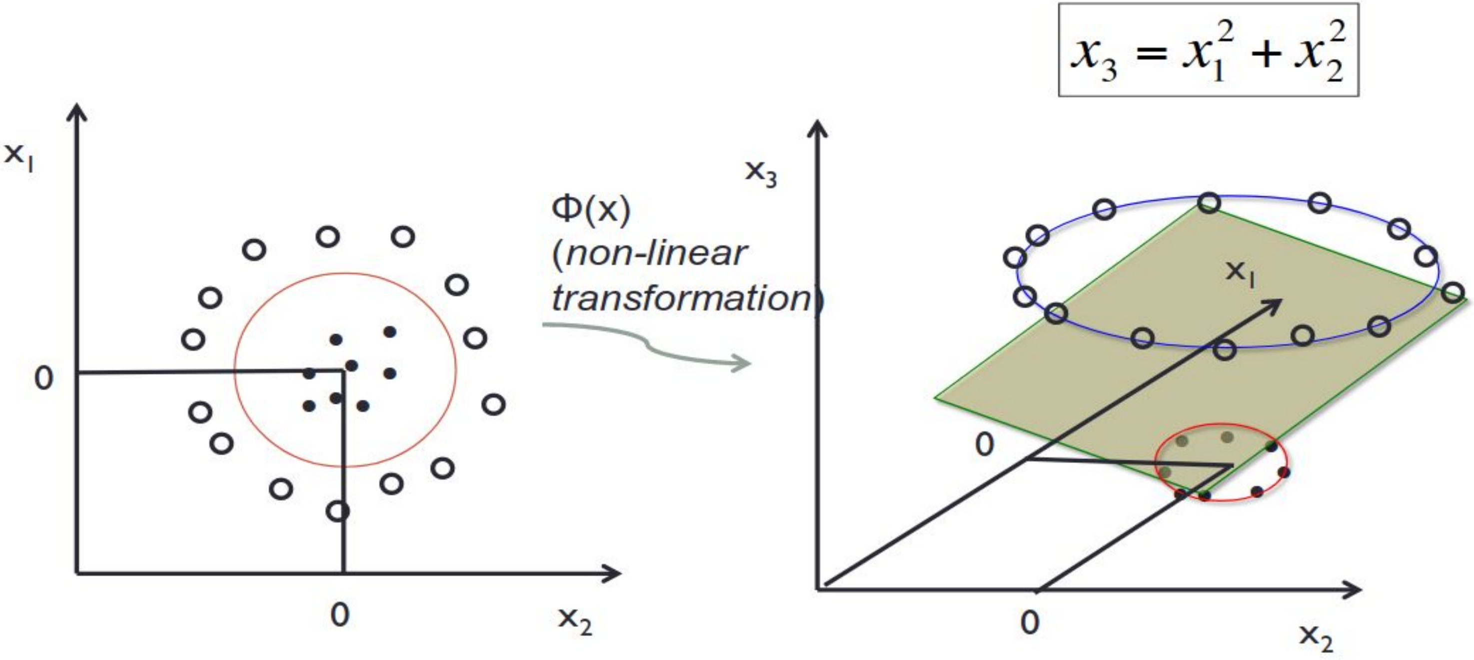


What about non-linearly separable data?

The circle describes the separation better..



What about non-linearly separable data?



Now, the data in the transformed space defined by (x_1, x_2, x_3) are linearly separable! So, let's use $\Phi(\mathbf{x})$ instead of $\mathbf{x}$!

Kernels

Intuitively, you can think of a kernel as mapping a set data from one coordinate system to another coordinate system.

In the original coordinate system the data may not be linearly separable at all, whereas in the new coordinate system if we choose the correct kernel, we should get a set a data set is very easily linearly separable.

Support Vector machine-Kernels

Learning phase

$$\text{maximize} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j)$$

$$\text{s.t.} \quad \alpha_i \geq 0 \text{ , and } \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

- $K(x_i,x_j)=\Phi(x_i)^T\Phi(x_j)$ is called the kernel function
- Kernel trick: Instead of having to transform each data point to the new space, we can directly replace with the dot product.

Popular Kernels

- Polynomial kernel

$$K(\mathbf{x},\mathbf{y}) = (\mathbf{x}^T\mathbf{y} + 1)^d$$

  - d=1: linear kernel
  - d=2: quadratic kernel

- (Gaussian) Radial basis function (RBF)

$$K(\mathbf{x},\mathbf{y}) = \exp(-\gamma \|\mathbf{x} - \mathbf{y}\|^2)$$

Kernels in Practice

- Dual coefficients less interpretable
- Long runtime for large datasets (100k samples)
- Real power in infinite-dimensional spaces: RBF
- RBF is universal kernel - can learn anything.

Preprocessing

- Kernel use inner products or distances.
- StandardScaler or MinMaxScaler
- Gamma parameter in RBF directly relates to scaling of data – default only works with zero-mean, unit variance.

Kernel Approximation in Practice

- SVM: only when 100000 >> n_samples, but works for n_features large
- RBFSampler, Nystroem can allow making anything kernelized!
- Some kernels (like chi2 and intersection) have really fast approximation.

Importance of Kernels

- Kernels work best for “small” n_samples
- Approximate kernels or random features for many samples
- Could do even SGD / streaming with kernel approximations