Support Vector Machines

Problem Definition

Consider a training set of n iid samples

$$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$$

where $\boldsymbol{x_i}$ is a vector of length m and

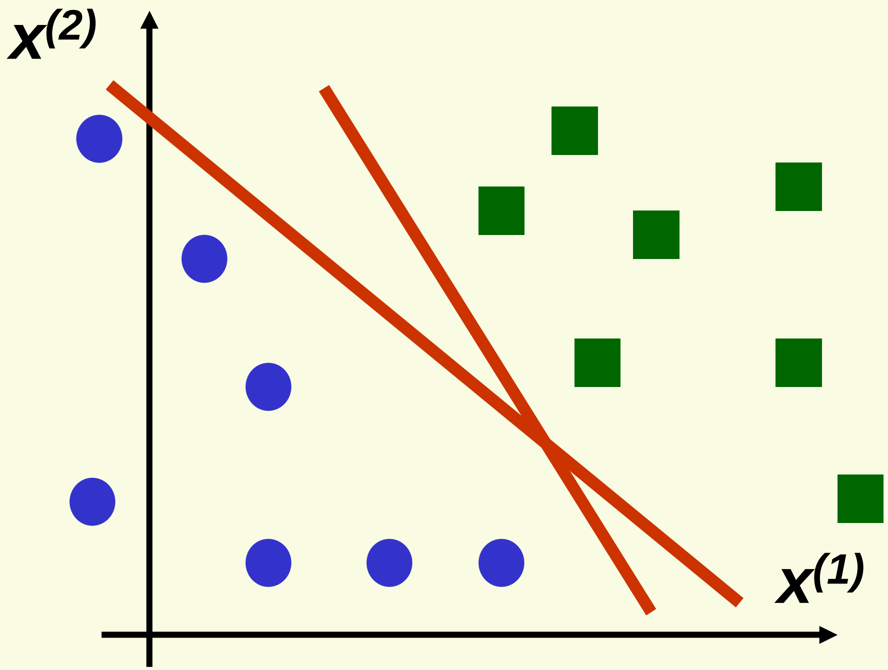
$y_i \in \{+1, -1\}$ is the class label for data point $\boldsymbol{x_i}$.

Find a separating hyperplane $w \cdot x + b = 0$

corresponding to the decision function

$$f(x) = sign(w \cdot x + b)$$

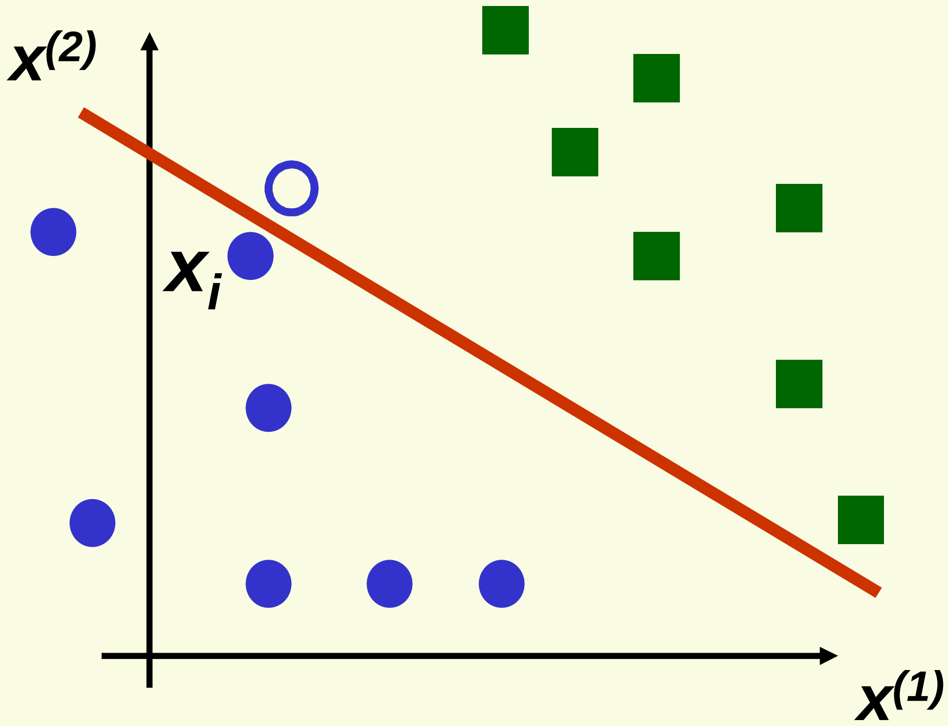
Separating Hyperplanes



- which separating hyperplane should we choose?

Separating Hyperplanes

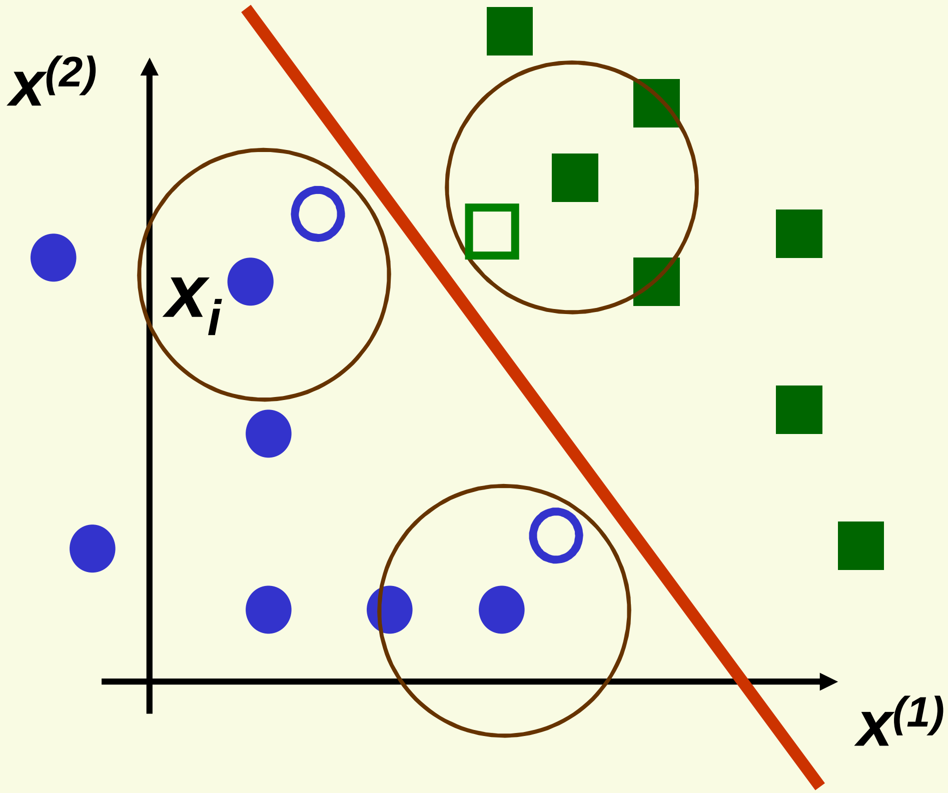
- Training data is just a subset of of all possible data
- Suppose hyperplane is close to sample $\boldsymbol{x_i}$
- If we see new sample close to sample ***i***, it is likely to be on the wrong side of the hyperplane



- Poor generalization (performance on unseen data)

Separating Hyperplanes

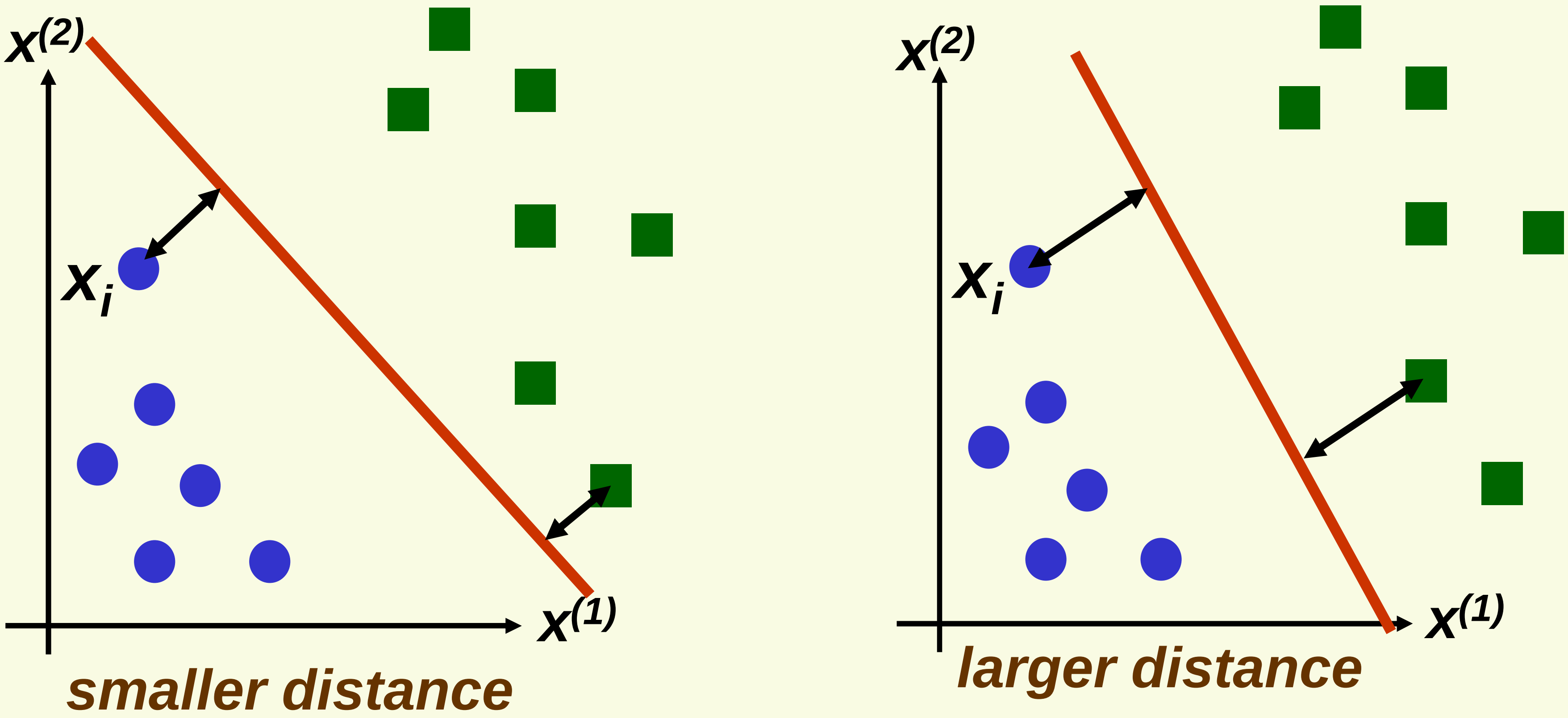
- Hyperplane as far as possible from any sample



- New samples close to the old samples will be classified correctly
- Good generalization

SVM

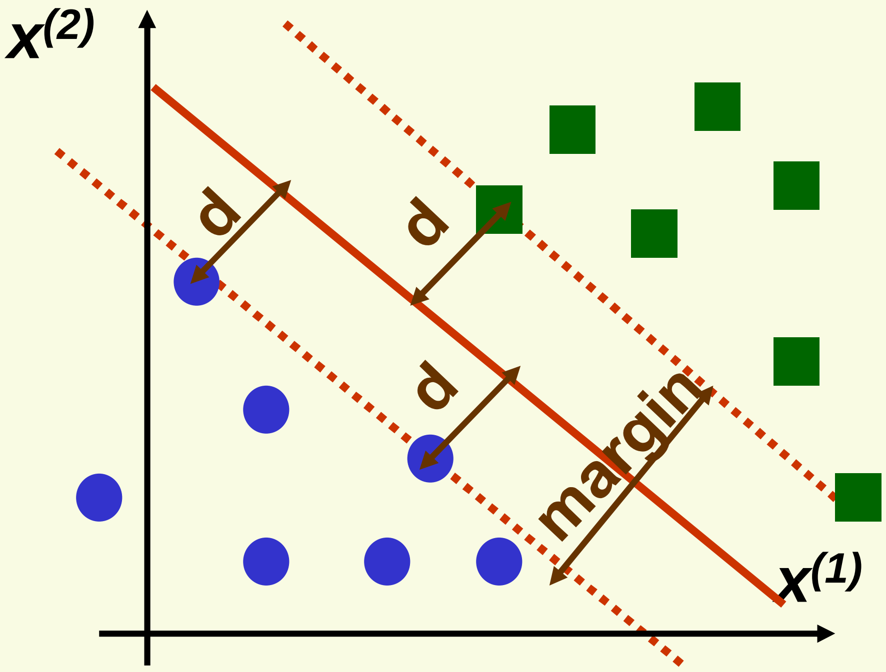
- Idea: maximize distance to the closest example



- For the optimal hyperplane
  - distance to the closest negative example = distance to the closest positive example

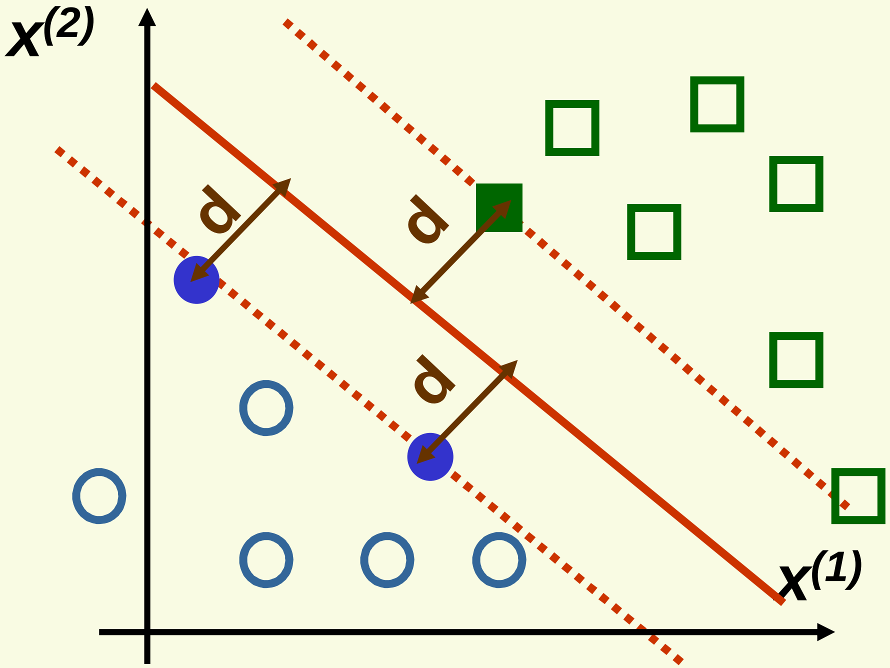
SVM: Linearly Separable Case

- SVM: maximize the ***margin***



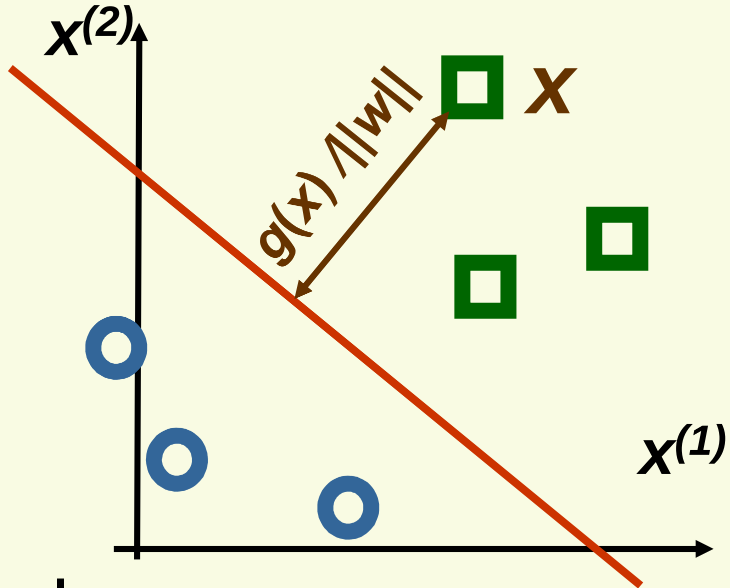
- ***margin*** is twice the absolute value of distance ***d*** of the closest examples to the separating hyperplane
- Better generalization (performance on test data)
  - in practice
  - and in theory

SVM: Linearly Separable Case



- ***Support vectors*** are the samples closest to the separating hyperplane
  - they are the most difficult patterns to classify
  - Optimal hyperplane is completely defined by support vectors
    - of course, we do not know which samples are support vectors without finding the optimal hyperplane

SVM: Formula for the Margin



- $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{w}^t\boldsymbol{x} + \boldsymbol{b}$
- absolute distance between $\boldsymbol{x}$ and the boundary $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{0}$

$$\frac{\left|\mathbf{w}^t\mathbf{x}+b\right|}{\left\|\mathbf{w}\right\|}$$

- distance is unchanged for hyperplane $\boldsymbol{g}_1(\boldsymbol{x})=\alpha\boldsymbol{g}(\boldsymbol{x})$

$$\frac{\left|\alpha\,\mathbf{w}^t\mathbf{x}+\alpha\,b\right|}{\left\|\alpha\,\mathbf{w}\right\|} = \frac{\left|\mathbf{w}^t\mathbf{x}+b\right|}{\left\|\mathbf{w}\right\|}$$

- Let $\mathbf{x}_i$ be an example closest to the boundary. Set

$$\left|\mathbf{w}^t\mathbf{x}_i+b\right| = 1$$

- Now the largest margin hyperplane is unique

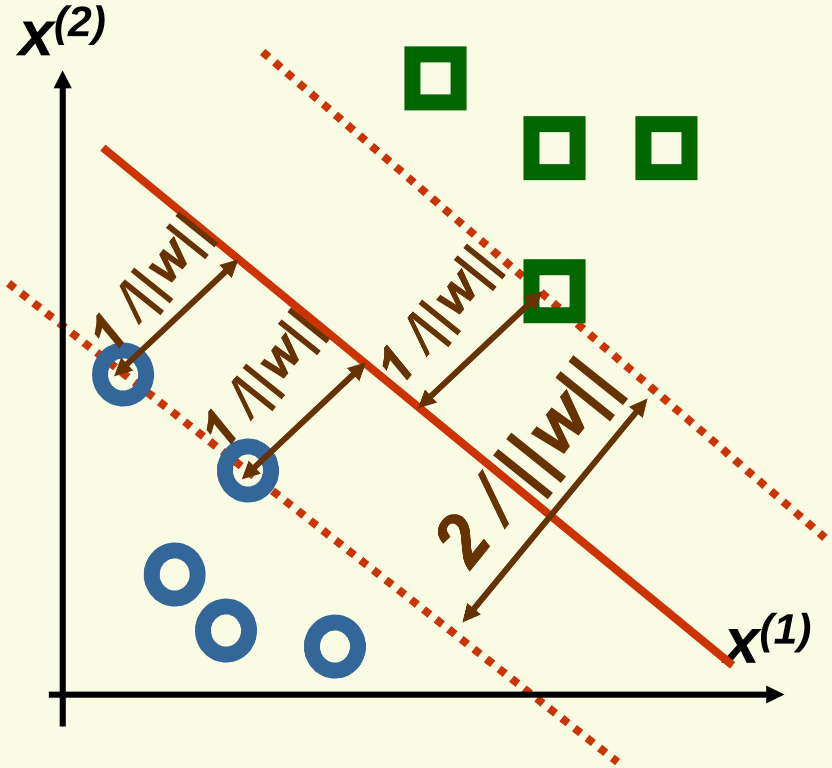
SVM: Formula for the Margin

- For uniqueness, set $\left|\mathbf{w}^t\mathbf{x}_i + b\right| = 1$ for any example $\boldsymbol{x_i}$ closest to the boundary
- now distance from closest sample $\boldsymbol{x_i}$ to $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{0}$ is

$$\frac{\left|\mathbf{w}^t\mathbf{x}_i + b\right|}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|}$$

- Thus the margin is

$$\boxed{m = \frac{2}{\|w\|}}$$



SVM: Optimal Hyperplane

- Maximize margin $m = \frac{2}{\|\mathbf{w}\|}$

  subject to constraints

$$\begin{cases} \mathbf{w}^t\mathbf{x}_i + b \geq \mathbf{1} & \mathbf{y}_i = \mathbf{1} \\ \mathbf{w}^t\mathbf{x}_i + b \leq -\mathbf{1} & \mathbf{y}_i = \mathbf{-1} \end{cases}$$

- Can convert our problem to

$$J(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 \qquad \text{s.t} \quad y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1$$

- ***J***(***w***) is a quadratic function, thus there is a single global minimum

Constrained Quadratic Programming

Primal Problem:

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

subject to $y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq \mathbf{1},\ \forall \mathbf{i}$

- Introduce Lagrange multipliers $\alpha_i \geq \mathbf{0}$ associated with the constraints
- The solution to the primal problem is equivalent to determining the saddle point of the function

$$L_P \equiv L(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - \mathbf{1} \right)$$

Solving Constrained QP

- At saddle point, L_P has minimum requiring

$$\frac{\partial L_P}{\partial w} = w - \sum_i \alpha_i y_i \mathbf{x}_i = \mathbf{0} \Rightarrow w = \sum_i \alpha_i y_i \mathbf{x}_i$$

$$\frac{\partial L_P}{\partial b} = \sum_i \alpha_i y_i = \mathbf{0}$$

Primal-Dual

Primal:

$$L_P = \frac{1}{2}\|\mathrm{w}\|^2 - \sum_{i=1}^{n} \alpha_i y_i (\mathrm{x}_i \cdot \mathrm{w} + b) + \sum_{i=1}^{n} \alpha_i$$

minimize L_P with respect to w,b,

subject to $\alpha_i \geq 0$

$$\mathrm{w} = \sum_i \alpha_i y_i \mathrm{x}_i \qquad \sum_i \alpha_i y_i = 0$$

substitute

Dual:

$$L_D = \sum_{i=1}^{l} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathrm{x}_i \mathrm{x}_j$$

maximize L_D with respect to $\boldsymbol{\alpha}$

subject to $\alpha_i \geq 0, \quad \sum_i \alpha_i y_i = 0$

Solving QP using dual problem

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

$$\text{constrained to} \quad \alpha_i \geq 0 \quad \forall i \quad \textit{and} \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

- $\alpha = \{\alpha_1, \ldots, \alpha_n\}$ are new variables, one for each sample
- $L_D(\alpha)$ can be optimized by quadratic programming
- $L_D(\alpha)$ formulated in terms of α
  - it depends on ***w*** and ***b*** indirectly
- $L_D(\alpha)$ depends on the number of samples, not on dimension of samples

Threshold

- ***b*** can be determined from the optimal α and Karush-Kuhn-Tucker (KKT) conditions

$$\alpha_i \left[y_i \left(\mathbf{w} \cdot \mathbf{x}_i + b \right) - \mathbf{1} \right] = \mathbf{0}, \quad \forall \mathbf{i}$$

- $\alpha_i > \mathbf{0}$ implies

$$y_i (w \cdot \mathbf{x}_i + b) = \mathbf{1} \;\Rightarrow\; \mathbf{w} \cdot \mathbf{x_i} + b = y_i$$

$$b = y_i - w \cdot x_i$$

Support Vectors

- For every sample ***i***, one of the following must hold
  - $\alpha_i = 0$
  - $\alpha_i > 0$ and $\boldsymbol{y_i}(\boldsymbol{w} \cdot \boldsymbol{x_i} + \boldsymbol{b} - \boldsymbol{1}) = \boldsymbol{0}$

- Many $\alpha_i = \mathbf{0} \quad \Rightarrow \quad \mathrm{w} = \sum_i \alpha_i y_i \mathrm{x}_i$ sparse solution

- Samples with $\alpha_i > \mathbf{0}$ are ***Support Vectors*** and they are the closest to the separating hyperplane

- Optimal hyperplane is completely defined by support vectors

SVM: Classification

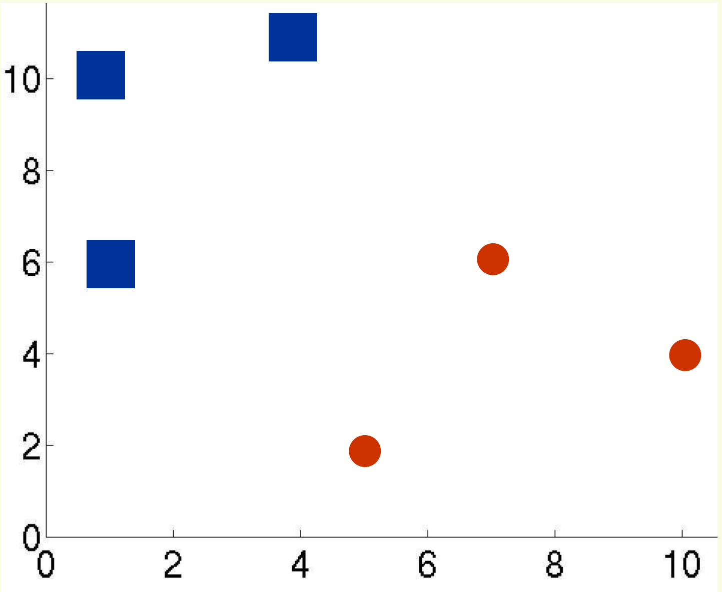
- Given a new sample x, finds its label y

$$y = sign(\mathbf{w} \cdot \mathbf{x} + b)$$

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i x_i$$

SVM: Example

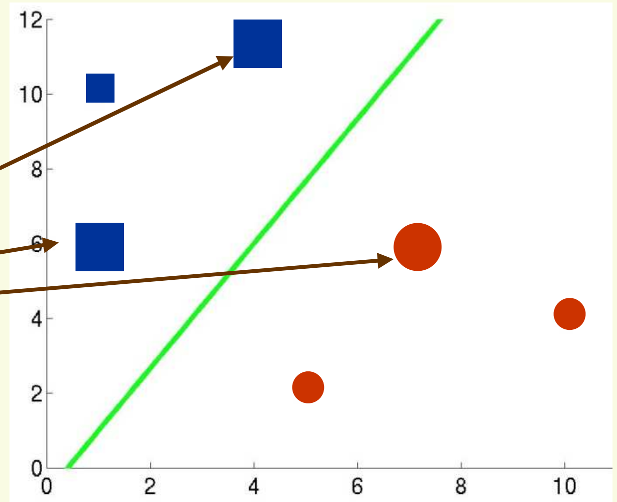
- Class 1: [1,6], [1,10], [4,11]
- Class 2: [5,2], [7,6], [10,4]



SVM: Example

- Solution $\alpha = \begin{bmatrix} \mathbf{0.036} \\ \mathbf{0} \\ \mathbf{0.039} \\ \mathbf{0} \\ \mathbf{0.076} \\ \mathbf{0} \end{bmatrix}$

support vectors

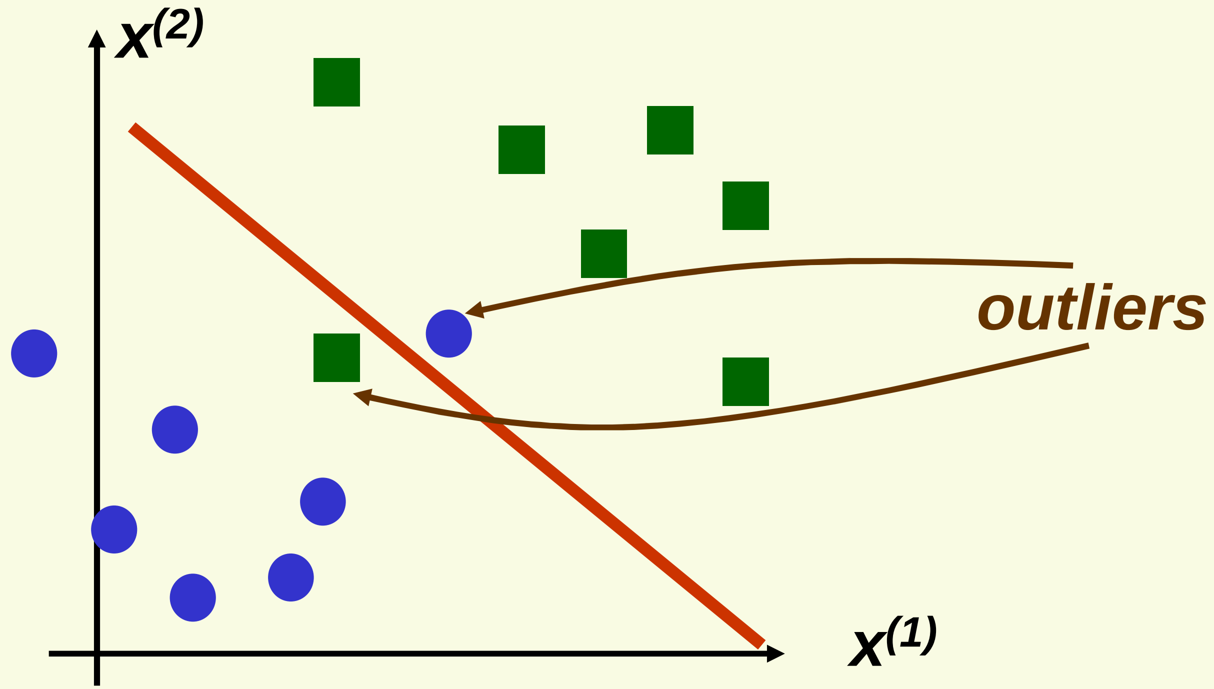


- find ***w*** using $w = \sum_{i=1}^{n} \alpha_i y_i x_i = (\alpha .* y)^t x = \begin{bmatrix} -\mathbf{0.33} \\ \mathbf{0.20} \end{bmatrix}$
- since α_1 >0, can find ***b*** using

$$b = y_1 - w^t x_1 = \mathbf{0.13}$$

SVM: Non Separable Case

- Data is most likely to be not linearly separable, but linear classifier may still be appropriate



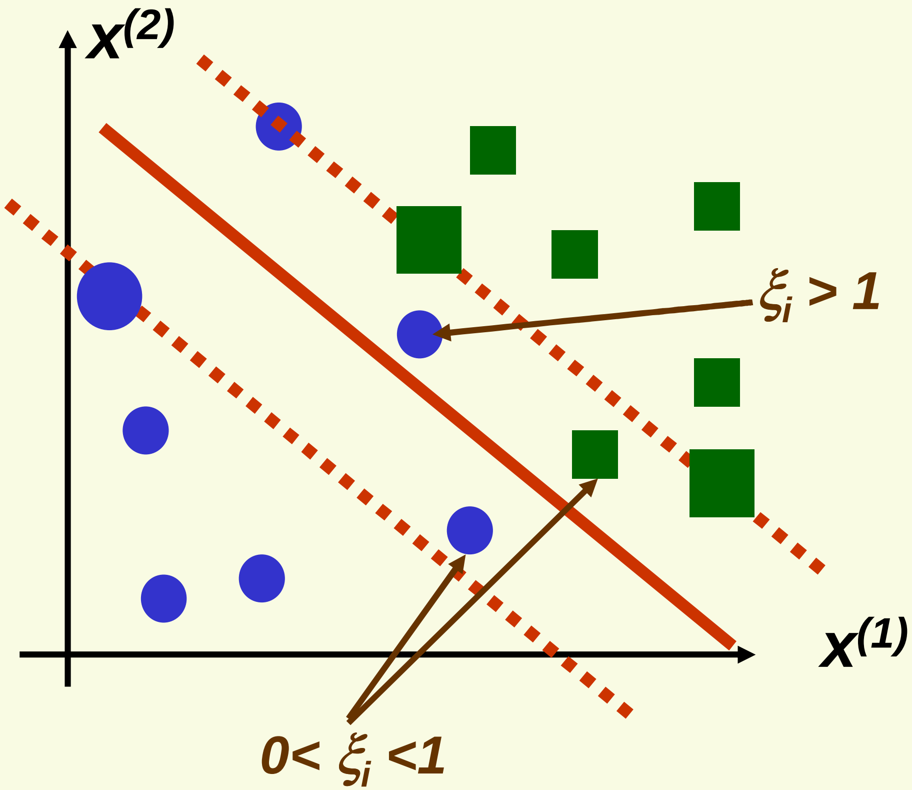
- Can apply SVM in non linearly separable case
  - data should be "almost" linearly separable for good performance

SVM with slacks

- Use nonnegative "slack" variables $\xi_1, \ldots, \xi_n$ (one for each sample)
- Change constraints from $y_i\left(\mathbf{w}^t\mathbf{x}_i + b\right) \geq \mathbf{1} \quad \forall i$ to

$$y_i\left(\mathbf{w}^t\mathbf{x}_i + b\right) \geq \mathbf{1} - \xi_i \quad \forall i$$

- ξ_i is a measure of deviation from the ideal position for sample ***i***
  - ξ_i >1 sample ***i*** is on the wrong side of the separating hyperplane
  - 0< ξ_i <1 sample ***i*** is on the right side of separating hyperplane but within the region of maximum margin



SVM with slacks

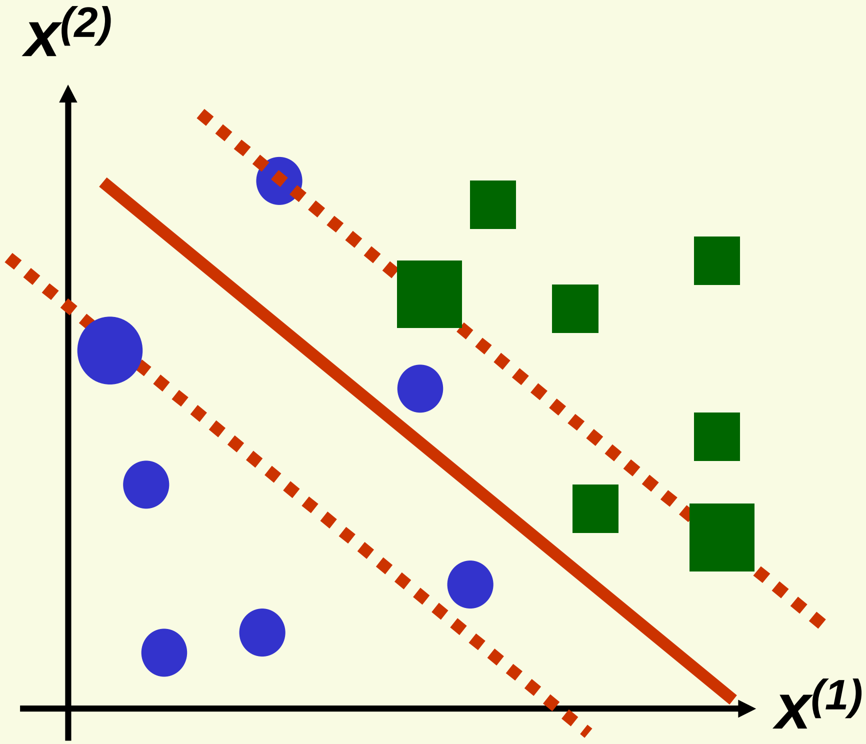
- Would like to minimize

$$J(w,\xi_1,\dots,\xi_n)=\frac{1}{2}\|w\|^2+C\sum_{i=1}^{n}\xi_i$$

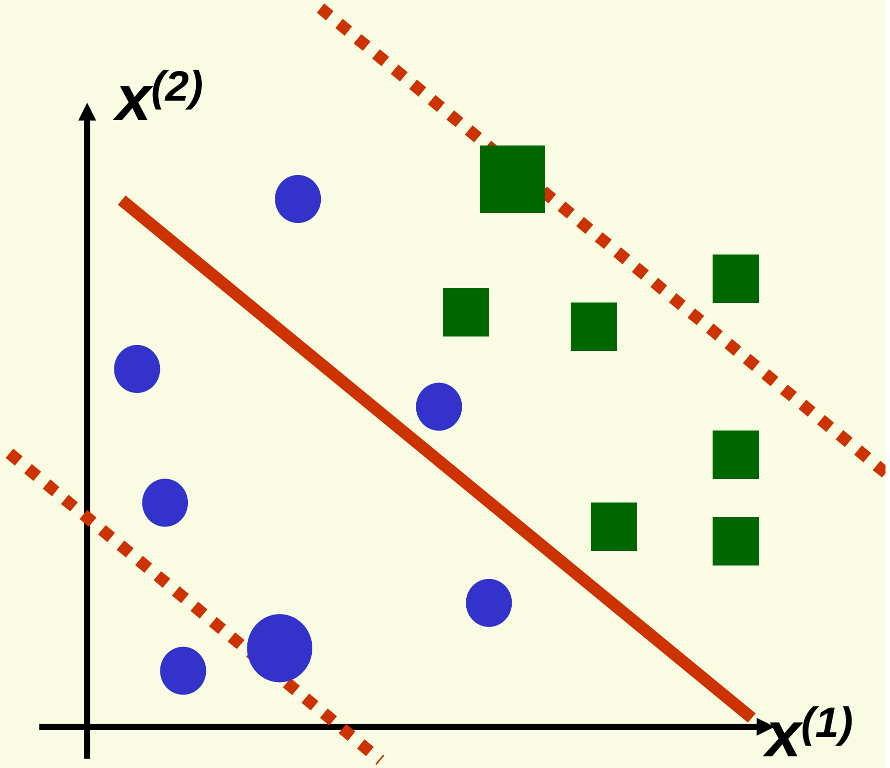
  - constrained to $y_i(\mathbf{w}^t\mathbf{x}_i+b)\geq 1-\xi_i$ and $\xi_i\geq 0 \quad \forall i$
  - **C** > **0** is a constant which measures relative weight of the first and second terms
    - if C is small, we allow a lot of samples not in ideal position
    - if C is large, we want to have very few samples not in ideal position

SVM with slacks

$$J(w,\xi_1,\ldots,\xi_n)=\frac{1}{2}\|w\|^2+C\sum_{i=1}^{n}\xi_i$$



large C, few samples not in ideal position



small C, a lot of samples not in ideal position

SVM with slacks– Dual Formulation

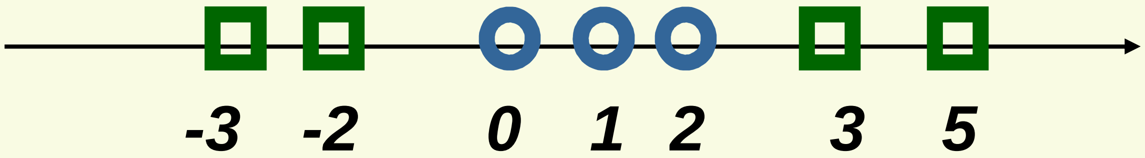
maximize $L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_i y_i y_j x_i^t x_j$

constrained to $0 \le \alpha_i \le C \quad \forall i \quad and \quad \sum_{i=1}^{n} \alpha_i y_i = 0$

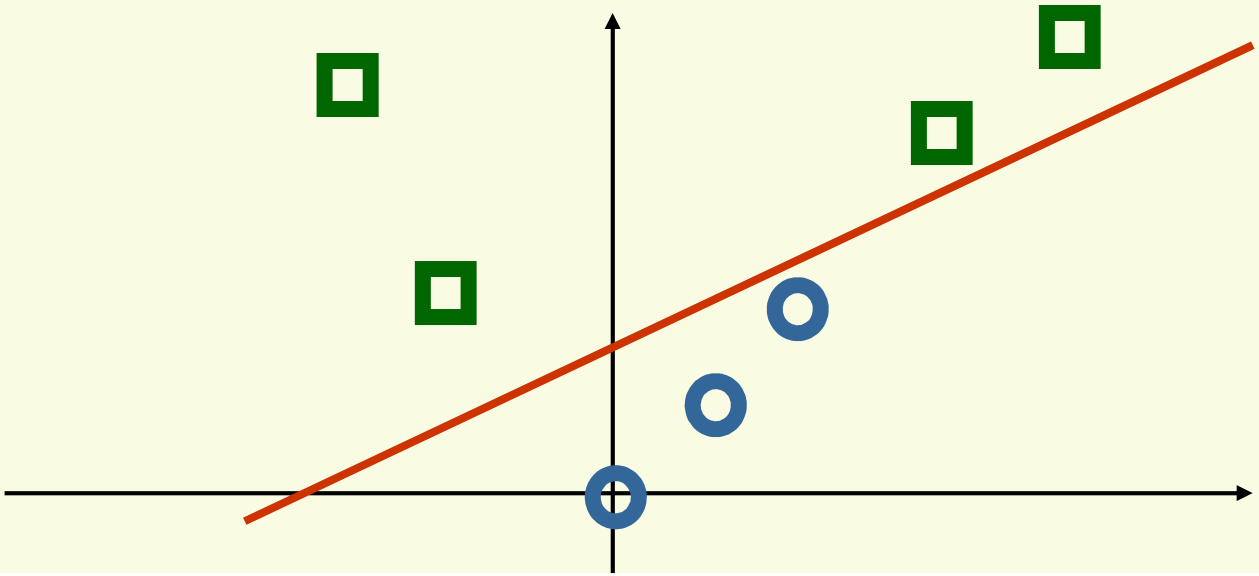
- find ***w*** using $w = \sum_{i=1}^{n} \alpha_i y_i x_i$
- solve for ***b*** using any $0 < \alpha_i < C$ and $\alpha_i [y_i (\mathbf{w}^t \mathbf{x}_i + b) - 1] = 0$

Non Linear Mapping

- Cover's theorem:
  - "*pattern-classification problem cast in a high dimensional space non-linearly is more likely to be linearly separable than in a low-dimensional space*"
- One dimensional space, not linearly separable

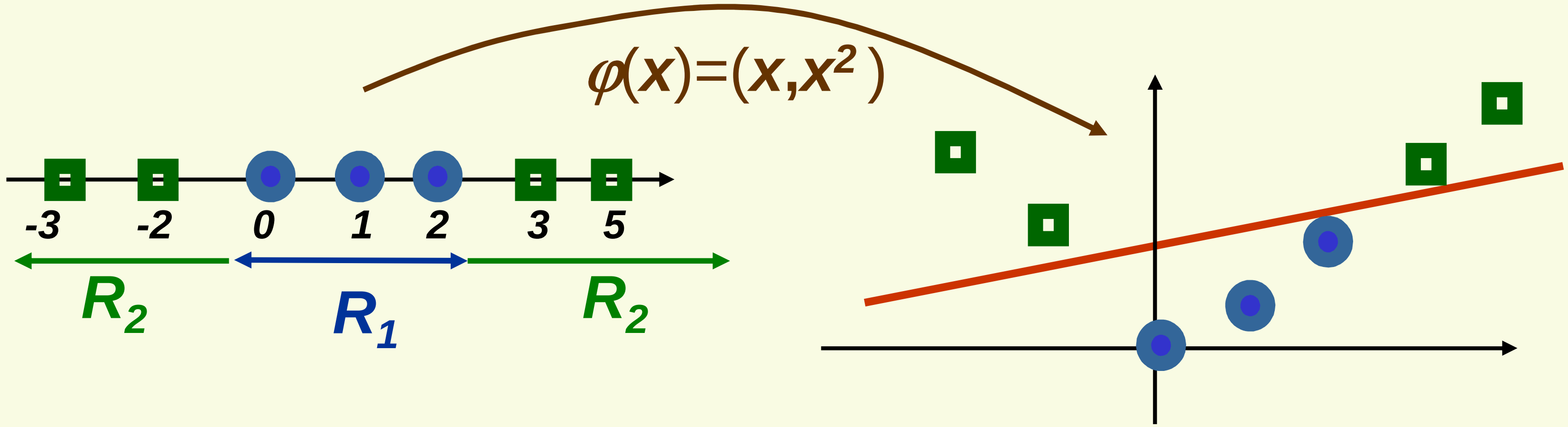


- Lift to two dimensional space with $\varphi(\boldsymbol{x})=(\boldsymbol{x},\boldsymbol{x}^2)$



Non Linear Mapping

- Solve a non linear classification problem with a linear classifier
  1. Project data $\boldsymbol{x}$ to high dimension using function $\varphi(\boldsymbol{x})$
  2. Find a linear discriminant function for transformed data $\varphi(\boldsymbol{x})$
  3. Final nonlinear discriminant function is $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{w}^t \varphi(\boldsymbol{x}) + \boldsymbol{w}_0$

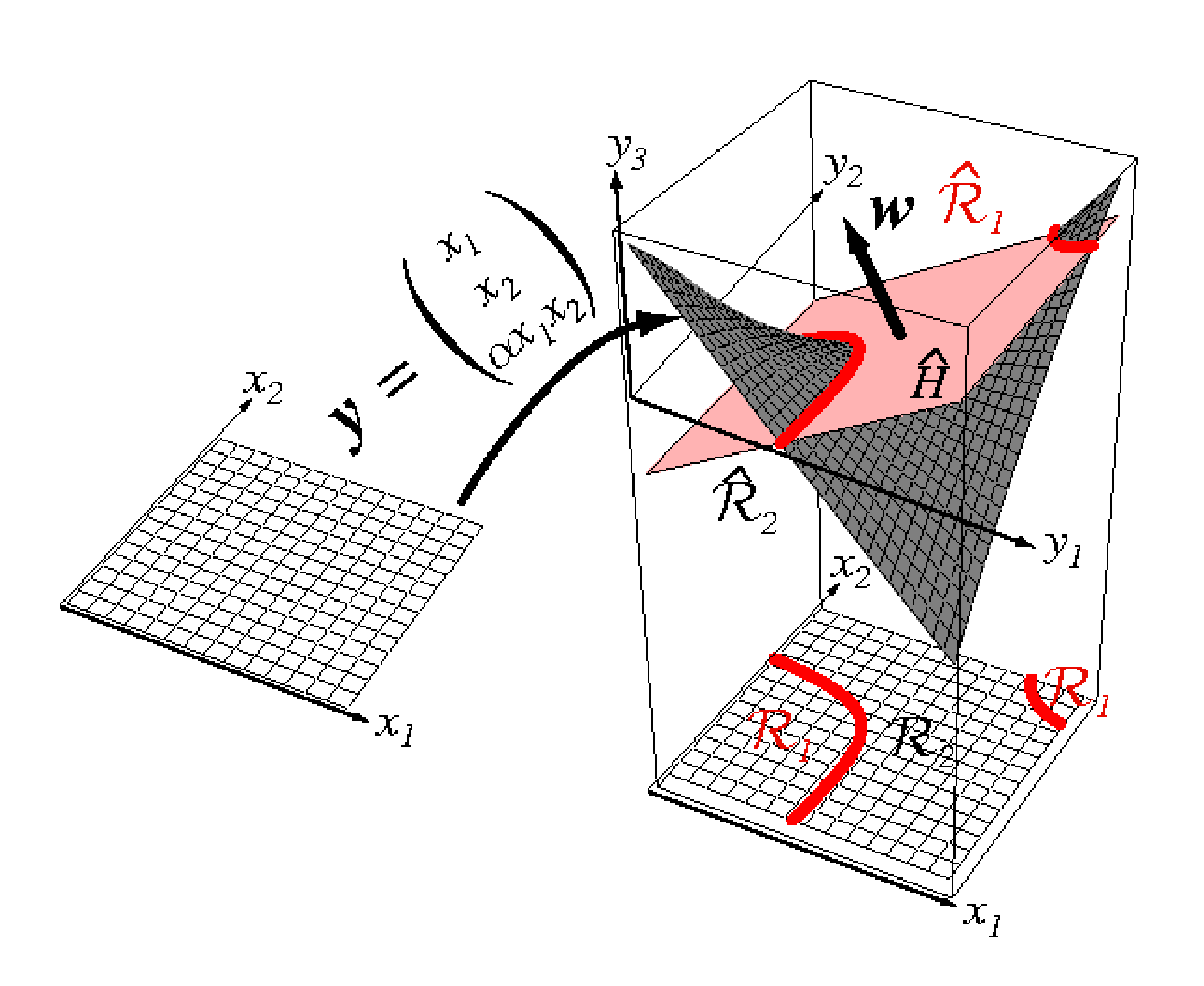


- In 2D, discriminant function is linear

$$g\left(\begin{bmatrix} x^{(1)} \\ x^{(2)} \end{bmatrix}\right) = \begin{bmatrix} w_1 & w_2 \end{bmatrix} \begin{bmatrix} x^{(1)} \\ x^{(2)} \end{bmatrix} + w_0$$

- In 1D, discriminant function is not linear $\quad g(x) = w_1 x + w_2 x^2 + w_0$

Non Linear Mapping: Another Example



Non Linear SVM

- Can use any linear classifier after lifting data into a higher dimensional space. However we will have to deal with the "curse of dimensionality"
  1. poor generalization to test data
  2. computationally expensive

- SVM handles the "curse of dimensionality" problem:
  1. enforcing largest margin permits good generalization
     - It can be shown that generalization in SVM is a function of the margin, independent of the dimensionality
  2. computation in the higher dimensional case is performed only implicitly through the use of ***kernel*** functions

Non Linear SVM: Kernels

- Recall SVM optimization

maximize $$L_D(\alpha)=\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_i y_i y_j x_i^t x_j$$

and classification $$y=sign\left(\sum_{i=1}^{n}\alpha_i y_i \mathbf{x_i}\cdot\mathbf{x}+b\right)$$

- Note that samples $\boldsymbol{x_i}$ appear only through the dot products $\boldsymbol{x_i^t x_j}$, $\boldsymbol{x_i^t x}$.

- If we lift $\boldsymbol{x_i}$ to high dimensional space F using $\varphi(\boldsymbol{x})$, need to compute high dimensional product $\varphi(\boldsymbol{x_i})^t\varphi(\boldsymbol{x_j})$

maximize $$L_D(\alpha)=\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_i y_i y_j \varphi(x_i)^t\varphi(x_j)$$

  - The dimensionality of space F not necessarily important. May not even know the map φ.

Kernel

- A function that returns the value of the dot product between the images of the two arguments:

  $K(x,y)=\varphi(\boldsymbol{x_i})^t\varphi(\boldsymbol{x_j})$

- Given a function K, it is possible to verify that it is a kernel.

$$\text{maximize} \quad L_D(\alpha)=\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_i y_i y_j \underbrace{\varphi(x_i)^t\varphi(x_j)}_{K(x_i,x_j)}$$

- Now we only need to compute $\boldsymbol{K(x_i,x_j)}$ instead of $\varphi(\boldsymbol{x_i})^t\varphi(\boldsymbol{x_j})$
  - "kernel trick": do not need to perform operations in high dimensional space explicitly

Kernel Matrix

- (aka the Gram matrix):

K=

K(1,1)	K(1,2)	K(1,3)	…	K(1,m)
K(2,1)	K(2,2)	K(2,3)	…	K(2,m)
…	…	…	…	…
K(m,1)	K(m,2)	K(m,3)	…	K(m,m)

- The central structure in kernel machines
- Contains all necessary information for the learning algorithm
- Fuses information about the data AND the kernel
- Many interesting properties:

From www.support-vector.net

Mercer's Theorem

- The kernel matrix is Symmetric Positive Definite
- Any symmetric positive definite matrix can be regarded as a kernel matrix, that is as an inner product matrix in some space

Every (semi)positive definite, symmetric function is a kernel: i.e. there exists a mapping φ such that it is possible to write:

$$\boldsymbol{K}(\boldsymbol{x},\boldsymbol{y}) = \varphi(\boldsymbol{x})^t \varphi(\boldsymbol{y})$$

Positive definite $\displaystyle \int_{\forall f \in L_2} K(x,y) f(x) f(y)\,dx\,dy \geq 0$

From www.support-vector.net

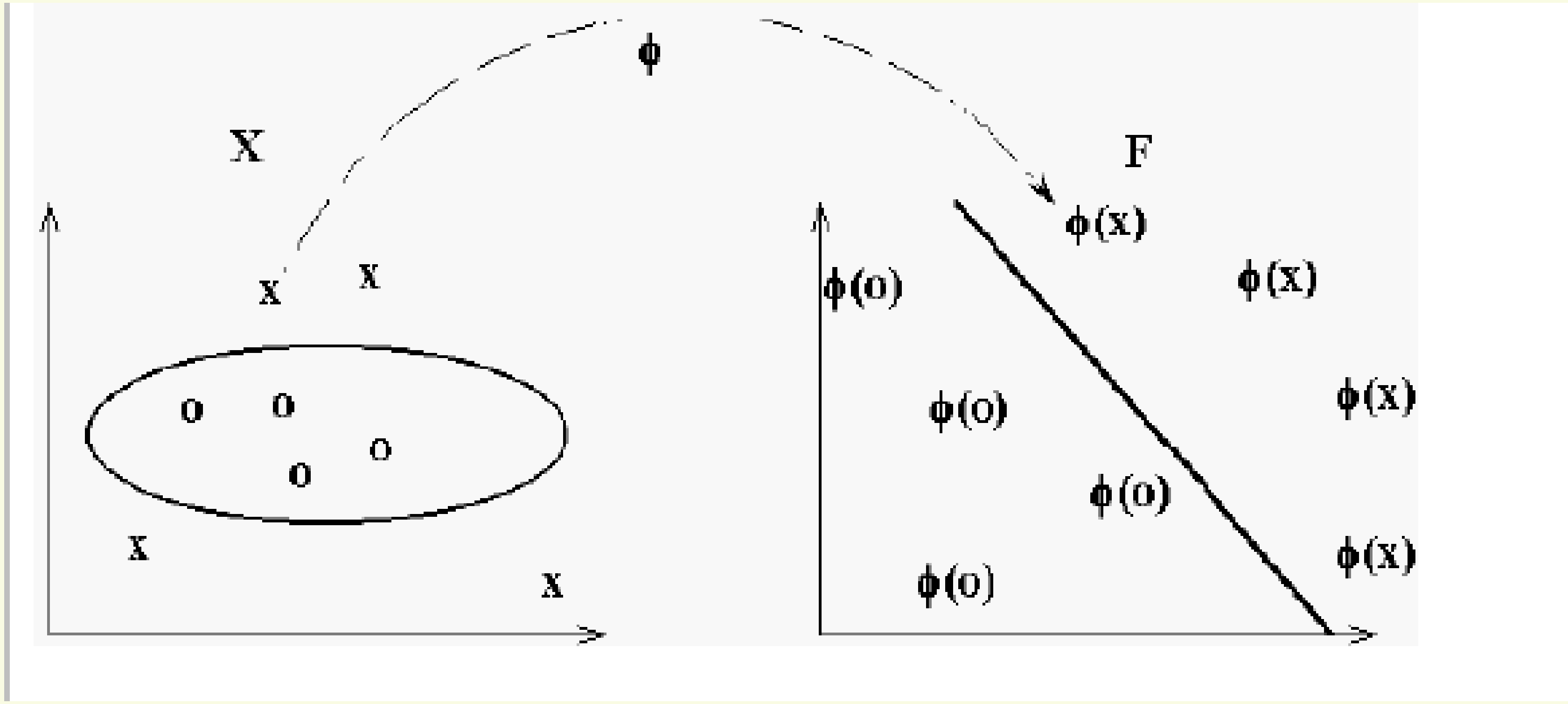
Examples of Kernels

- Some common choices (both satisfying Mercer's condition):
  - Polynomial kernel $K(x_i, x_j) = (x_i^t x_j + 1)^p$
  - Gaussian radial Basis kernel (data is lifted in infinite dimension)

$$K(x_i, x_j) = \exp\left(-\frac{1}{2\sigma^2}\left\|x_i - x_j\right\|^2\right)$$

From www.support-vector.net

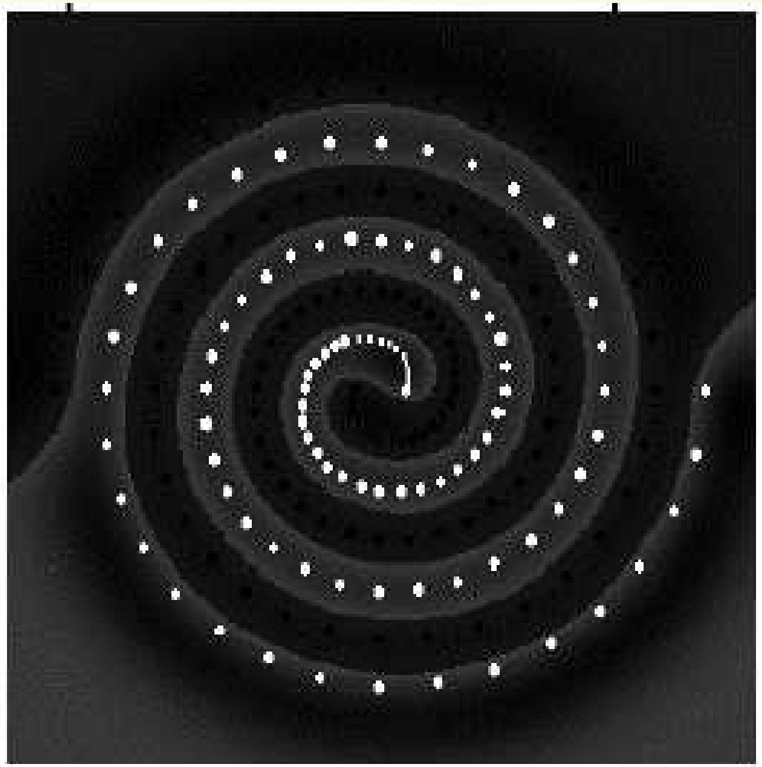
Example Polynomial Kernels



From www.support-vector.net

Example: the two spirals

- Separated by a hyperplane in feature space (gaussian kernels)



From www.support-vector.net

Making Kernels

- The set of kernels is closed under some operations. If K, K' are kernels, then:
- K+K' is a kernel
- cK is a kernel, if c>0
- aK+bK' is a kernel, for a,b >0
- Etc etc etc……
- can make complex kernels from simple ones: modularity !

Non Linear SVM Recepie

- Start with data $\mathbf{x_1},\ldots,\mathbf{x_n}$ which lives in feature space of dimension $\boldsymbol{d}$
- Choose kernel $\boldsymbol{K}(\mathbf{x_i},\mathbf{x_j})$ corresponding to some function $\varphi(\mathbf{x_i})$ which takes sample $\mathbf{x_i}$ to a higher dimensional space
- Find the largest margin linear discriminant function in the higher dimensional space by using quadratic programming package to solve:

$$\text{maximize} \quad L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

$$\text{constrained to} \quad 0 \le \alpha_i \le C \quad \forall i \quad \textit{and} \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

Non Linear SVM Recipe

- Weight vector ***w*** in the high dimensional space:

$$w = \sum_{i}^{n} \alpha_i y_i \varphi(x_i)$$

- Linear discriminant function of largest margin in the high dimensional space:

$$\boldsymbol{g}(\varphi(\boldsymbol{x})) = \boldsymbol{w}^t \varphi(\boldsymbol{x}) = \left(\sum_{x_i \in S} \alpha_i y_i \varphi(x_i) \right)^t \varphi(x)$$

- Non linear discriminant function in the original space:

$$g(x) = \left(\sum_{x_i \in S} \alpha_i y_i \varphi(x_i) \right)^t \varphi(x) = \sum_{x_i \in S} \alpha_i y_i \varphi^t(x_i) \varphi(x) = \sum_{x_i \in S} \alpha_i y_i K(x_i, x)$$

- decide class 1 if ***g*** (***x***) > 0, otherwise decide class 2

Non Linear SVM

- Nonlinear discriminant function

$$g(\boldsymbol{x}) = \sum_{\boldsymbol{x}_i \in S} \boxed{\alpha_i} \boxed{z_i} \boxed{K(\boldsymbol{x}_i, \boldsymbol{x})}$$

$$g(\boldsymbol{x}) = \sum \boxed{\text{weight of support vector } \boldsymbol{x}_i} \boxed{\mp 1} \boxed{\text{“inverse distance” from } \boldsymbol{x} \text{ to support vector } \boldsymbol{x}_i}$$

most important
training samples,
i.e. support vectors

$$K(\boldsymbol{x}_i, \boldsymbol{x}) = \exp\left(-\frac{1}{2\sigma^2}\|\boldsymbol{x}_i - \boldsymbol{x}\|^2\right)$$

Higher Order Polynomials
Taken from Andrew Moore

Poly-nomial	$\phi(\boldsymbol{x})$	Cost to build *H* matrix traditionally	Cost if d=100	$\phi(\boldsymbol{a})^t\phi(\boldsymbol{b})$	Cost to build *H* matrix sneakily	Cost if d=100
Quadratic	All $d^2/2$ terms up to degree 2	$d^2\, n^2 /4$	2,500 n^2	$(\boldsymbol{a}^t\boldsymbol{b}+1)^2$	$d\, n^2 / 2$	50 n^2
Cubic	All $d^3/6$ terms up to degree 3	$d^3\, n^2 /12$	83,000 n^2	$(\boldsymbol{a}^t\boldsymbol{b}+1)^3$	$d\, n^2 / 2$	50 n^2
Quartic	All $d^4/24$ terms up to degree 4	$d^4\, n^2 /48$	1,960,000 n^2	$(\boldsymbol{a}^t\boldsymbol{b}+1)^4$	$d\, n^2 / 2$	50 n^2

n is the number of samples, ***d*** is number of features

SVM Summary

- Advantages:
  - Based on nice theory
  - excellent generalization properties
  - objective function has no local minima
  - can be used to find non linear discriminant functions
  - Complexity of the classifier is characterized by the number of support vectors rather than the dimensionality of the transformed space

- Disadvantages:
  - It’s not clear how to select a kernel function in a principled manner
  - tends to be slower than other methods (in non-linear case).