Support Vector Machines

SVM

- Said to start in 1979 with Vladimir Vapnik's paper
- Major developments throughout 1990's
- Elegant theory
  - Has good generalization properties
- Have been applied to diverse problems very successfully in the last 10-15 years
- One of the most important developments in pattern recognition in the last 10 years

Problem Definition

Consider a training set of n iid samples

$$(\boldsymbol{x}_1, \boldsymbol{y}_1), (\boldsymbol{x}_2, \boldsymbol{y}_2), \ldots, (\boldsymbol{x}_n, \boldsymbol{y}_n)$$

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

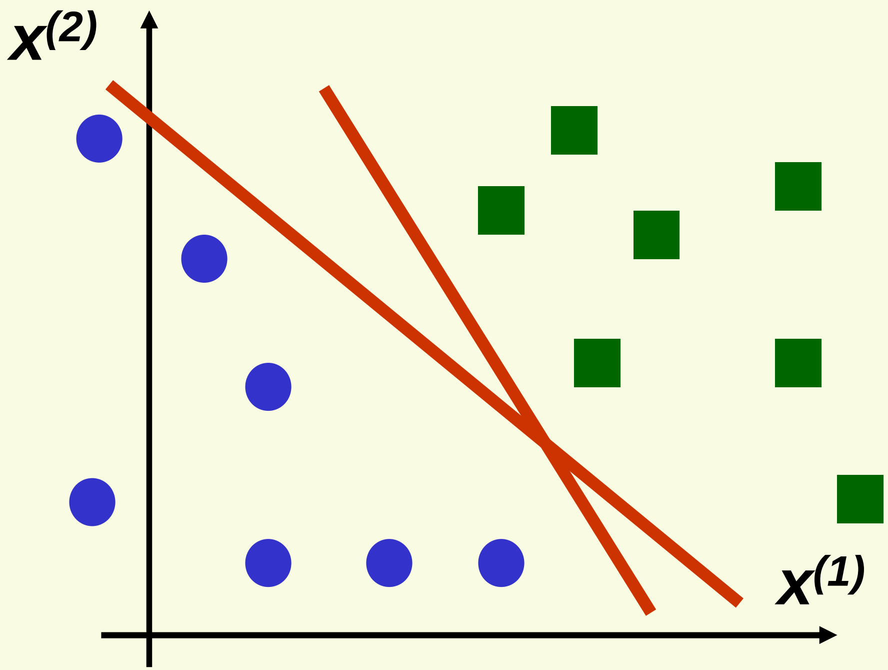
$\boldsymbol{y}_i \in \{+1, -1\}$ is the class label for data point $\boldsymbol{x}_i$.

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

corresponding to the decision function

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

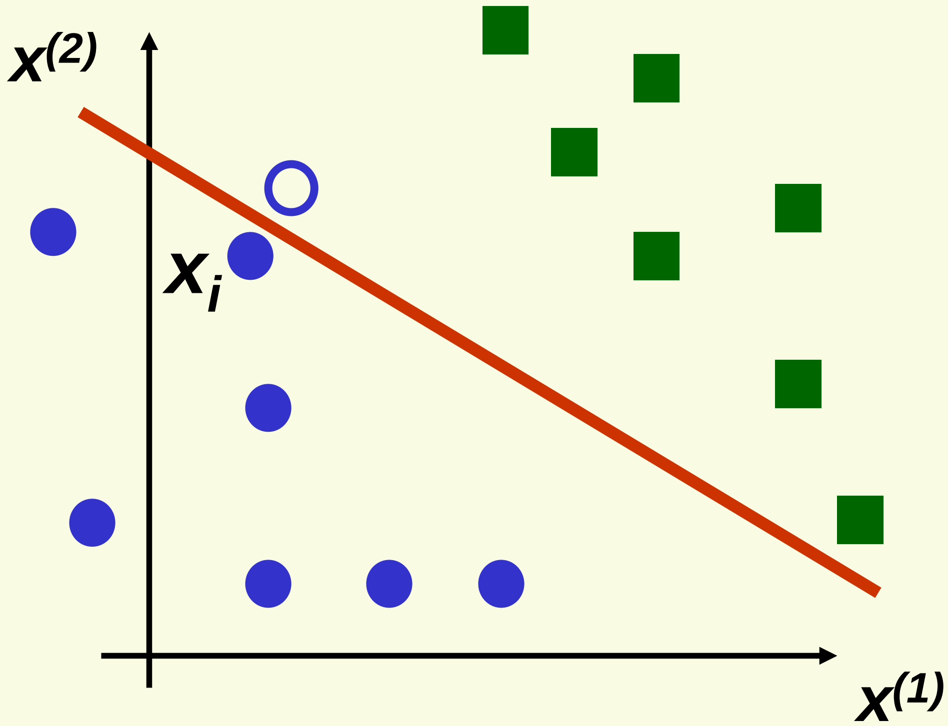
Separating Hyperplanes



- which separating hyperplane should we choose?

Separating Hyperplanes

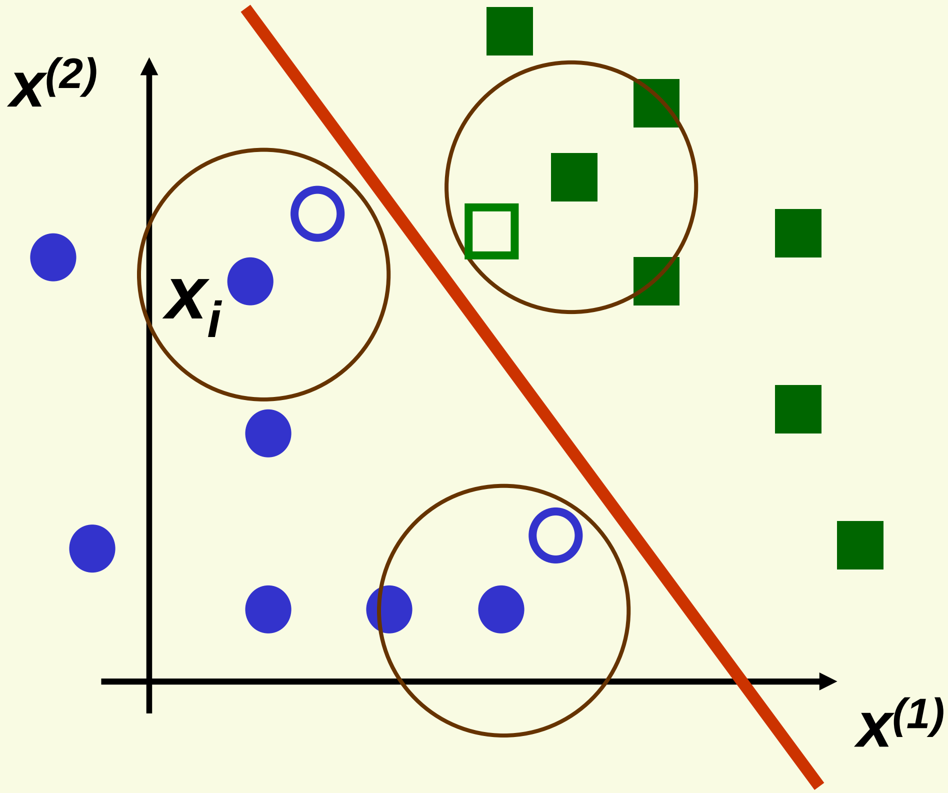
- Training data is just a subset of of all possible data
- Suppose hyperplane is close to sample $\boldsymbol{x_i}$
- If we see new sample close to sample ***i***, it is likely to be on the wrong side of the hyperplane



- Poor generalization (performance on unseen data)

Separating Hyperplanes

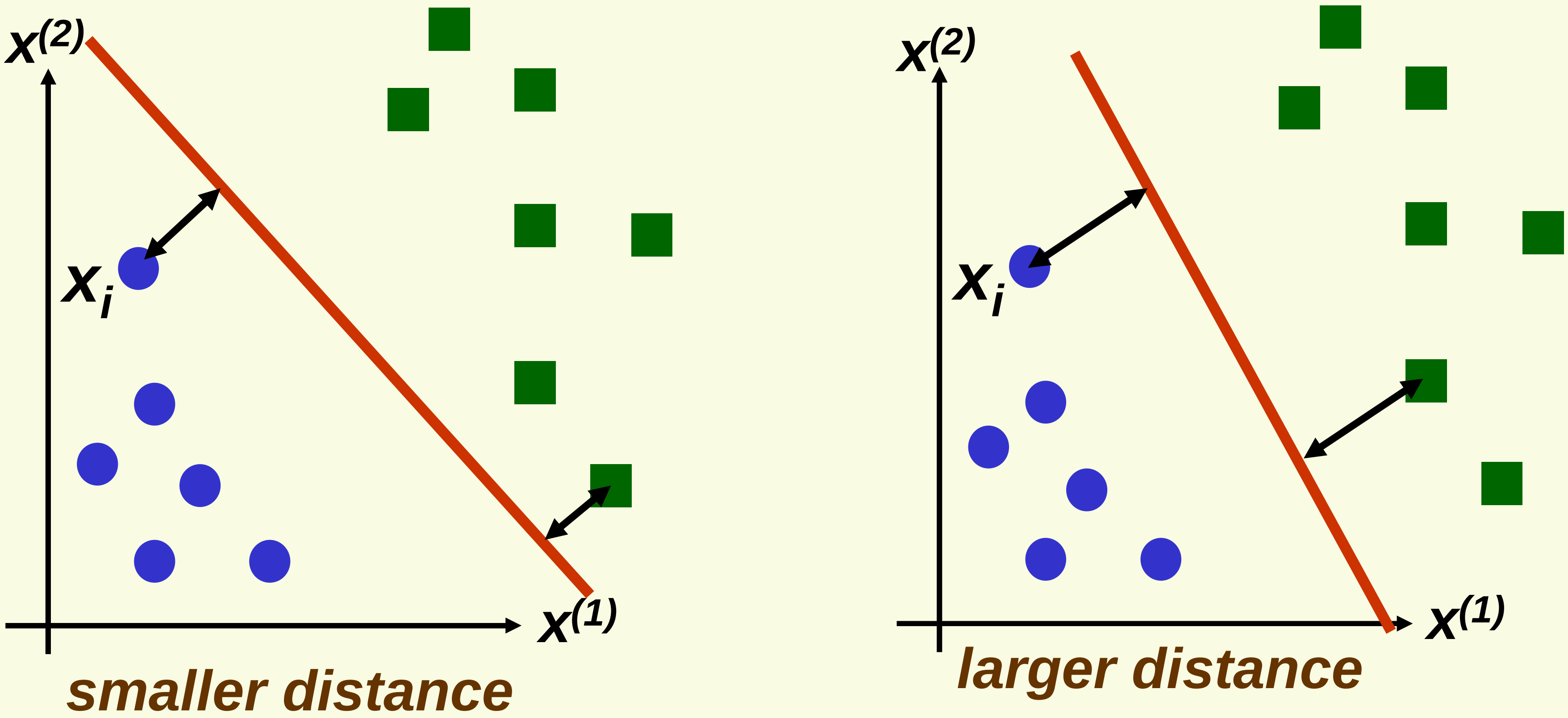
- Hyperplane as far as possible from any sample



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

SVM

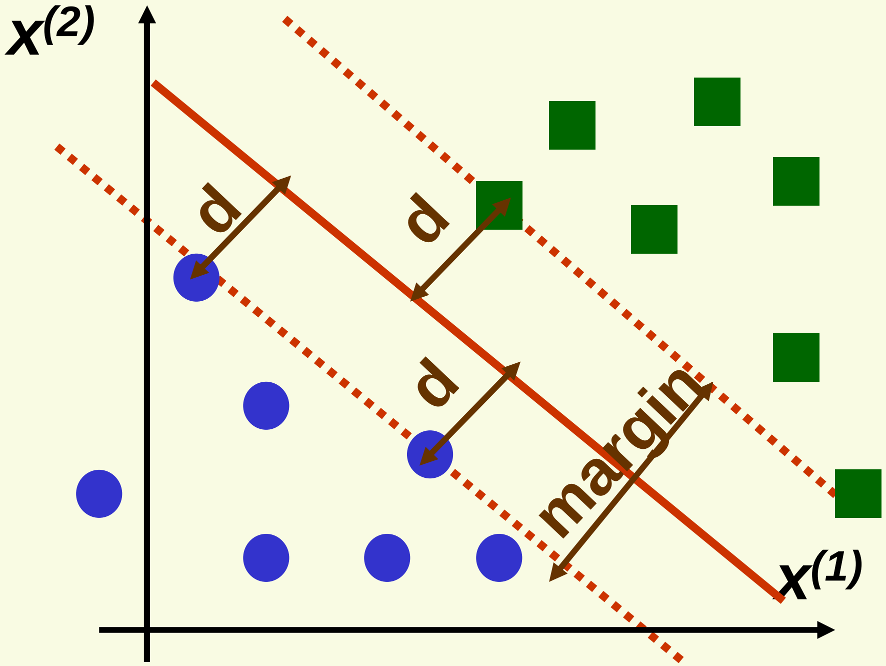
- Idea: maximize distance to the closest example



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

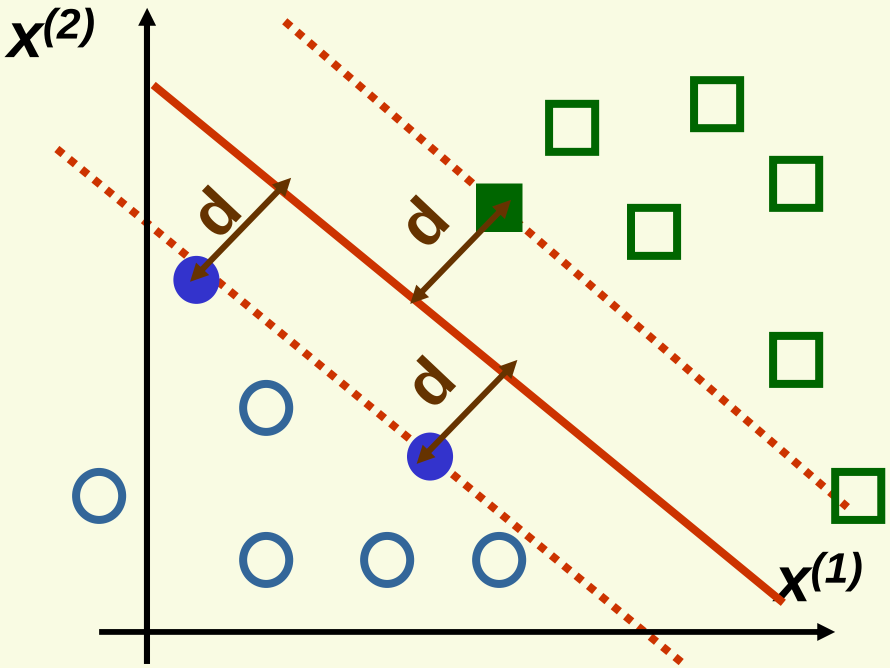
SVM: Linearly Separable Case

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



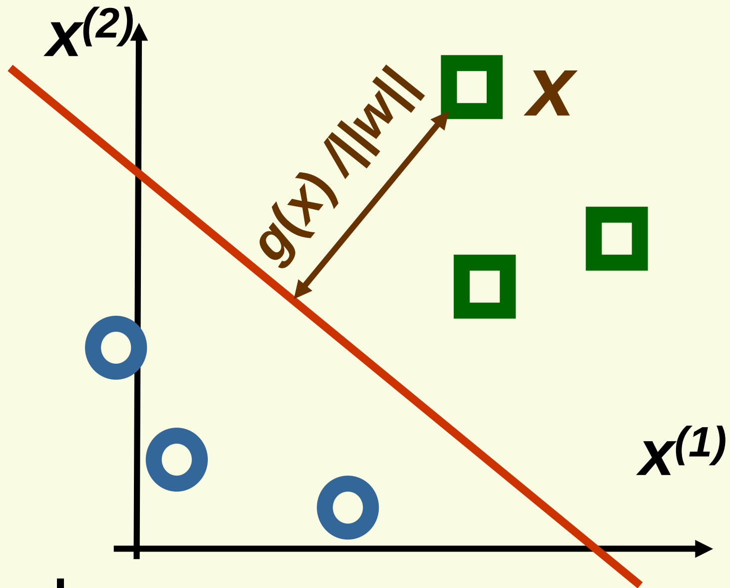
- ***margin*** is twice the absolute value of distance ***d*** of the closest example to the separating hyperplane
- Better generalization (performance on test data)
  - in practice
  - and in theory

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



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

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

- distance is unchanged for hyperplane $g_1(\mathbf{x}) = \alpha g(\mathbf{x})$

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

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

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

- Now the largest margin hyperplane is unique

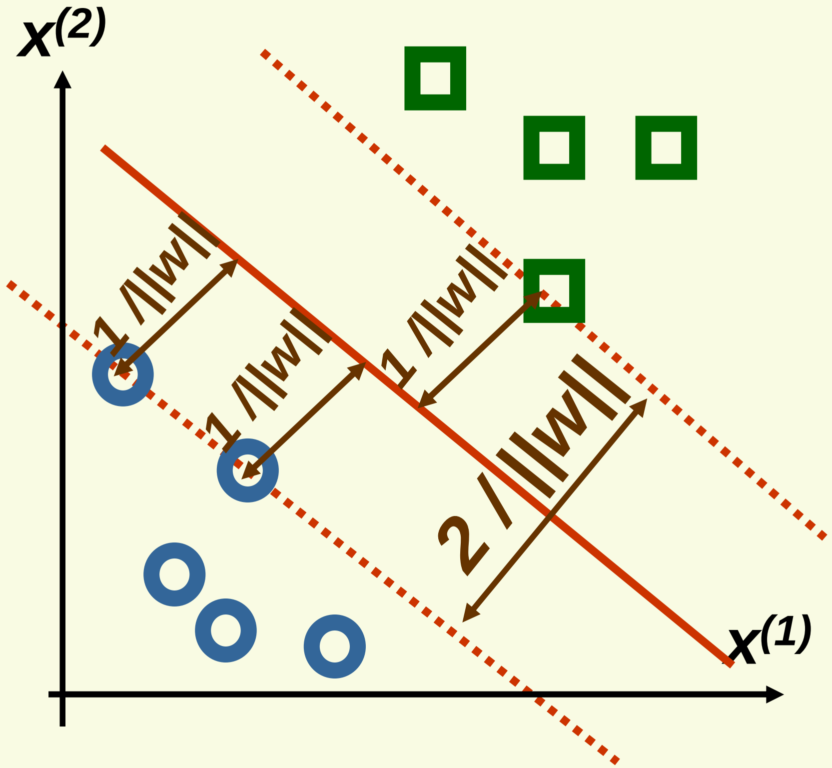
SVM: Formula for the Margin

- 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

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

- Thus the margin is

$$m = \frac{2}{\left\|w\right\|}$$



SVM: Optimal Hyperplane

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

  subject to constraints

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

- Can convert our problem to

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

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

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

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

Solving Constrained QP

- At saddle point, L_P has minimum requiring

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

$$\frac{\partial \boldsymbol{L_P}}{\partial \boldsymbol{b}} = \sum_i \alpha_i \boldsymbol{y}_i = \mathbf{0}$$

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

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

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

- $\alpha = \{\alpha_1, \ldots, \alpha_n\}$ 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

Threshold

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

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

- $\alpha_i > 0$ implies

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

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

Support Vectors

- For every sample ***i***, one of the following must hold
  - $\alpha_i = 0$
  - $\alpha_i > 0$ and $y_i(\mathbf{w} \cdot \mathbf{x}_i + b - 1) = 0$
- Many $\alpha_i = \mathbf{0} \;\Rightarrow\; \mathrm{w} = \sum_i \alpha_i y_i \mathrm{x}_i$ 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

More on dual problem

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

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

- $L_D(\alpha)$ depends on the number of samples, not on dimension of samples
- samples appear only through the dot products $x_i^t x_j$
- This will become important when looking for a ***nonlinear*** discriminant function, as we will see soon

SVM: Classification

- Given a new sample x, finds its label y

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

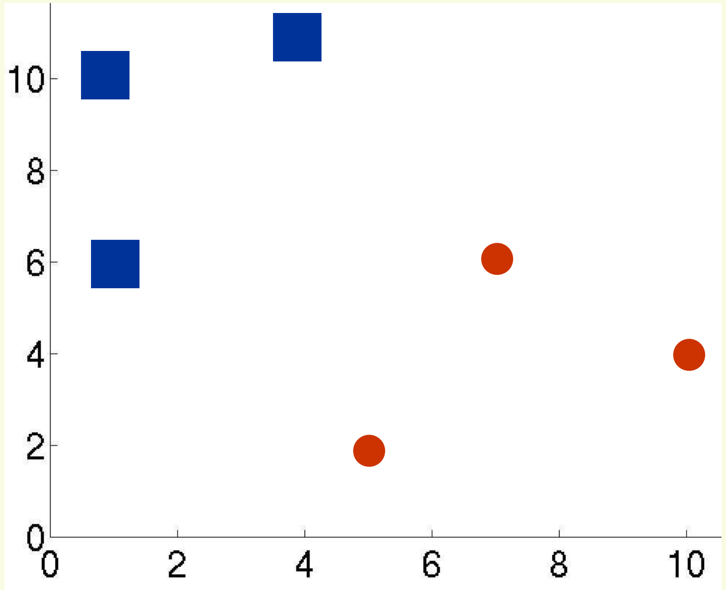
$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i x_i$$

duality

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

SVM: Example

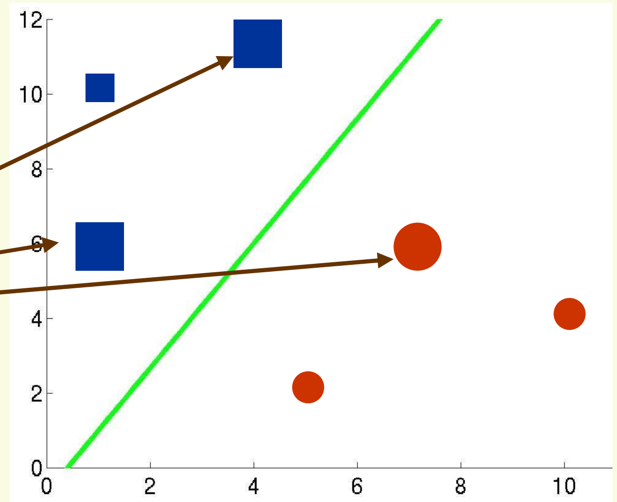
- Class 1: [1,6], [1,10], [4,11]
- Class 2: [5,2], [7,6], [10,4]



SVM: Example

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

support vectors



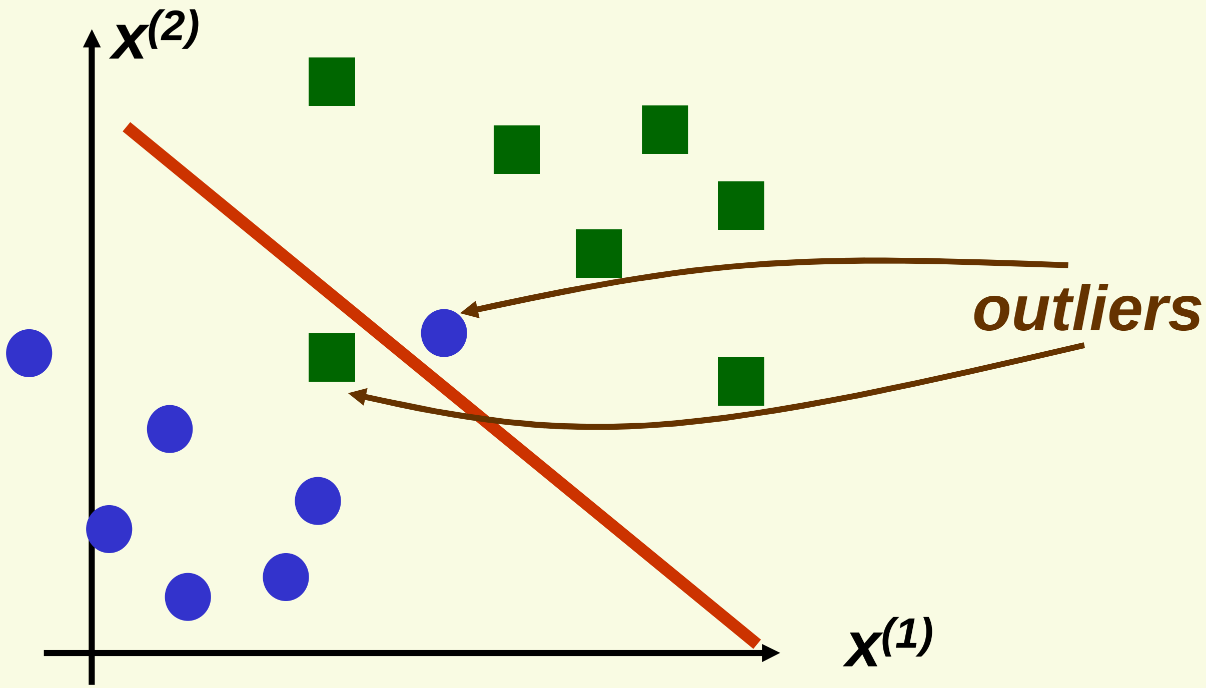
- find ***w*** using $\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i = (\alpha .* y)^t \boldsymbol{x} = \begin{bmatrix} -0.33 \\ 0.20 \end{bmatrix}$

- since α_1 >0, can find ***b*** using

$$b = \frac{1}{y_1} - w^t x_1 = 0.13$$

SVM: Non Separable Case

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



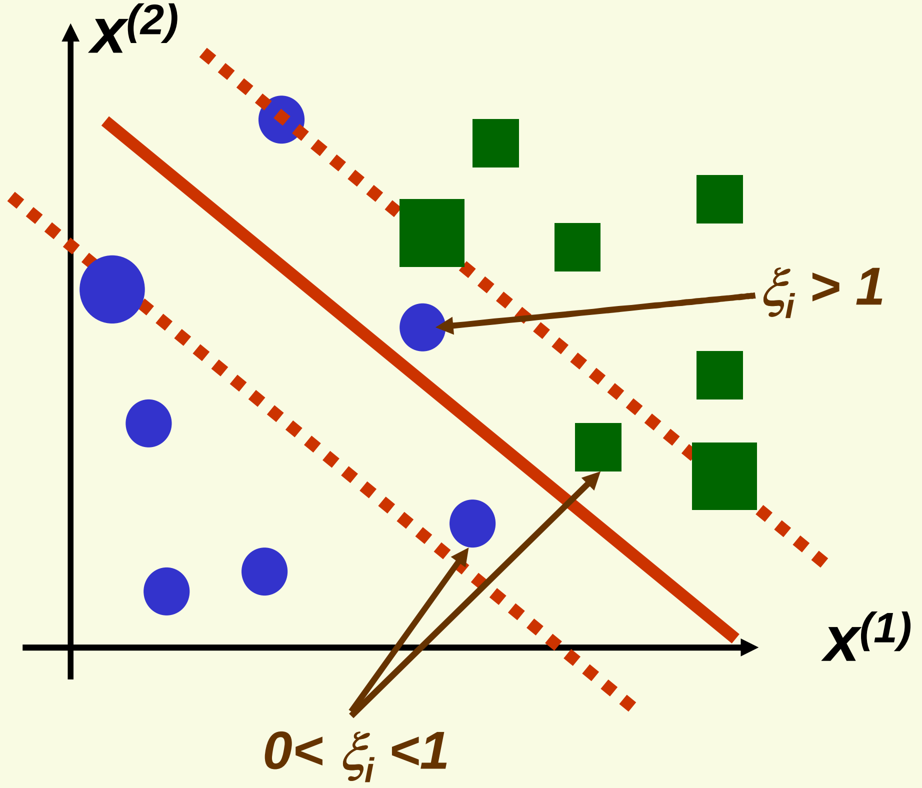
- Can apply SVM in non linearly separable case
  - data should be "almost" linearly separable for good performance

SVM with slacks

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

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

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



SVM with slacks

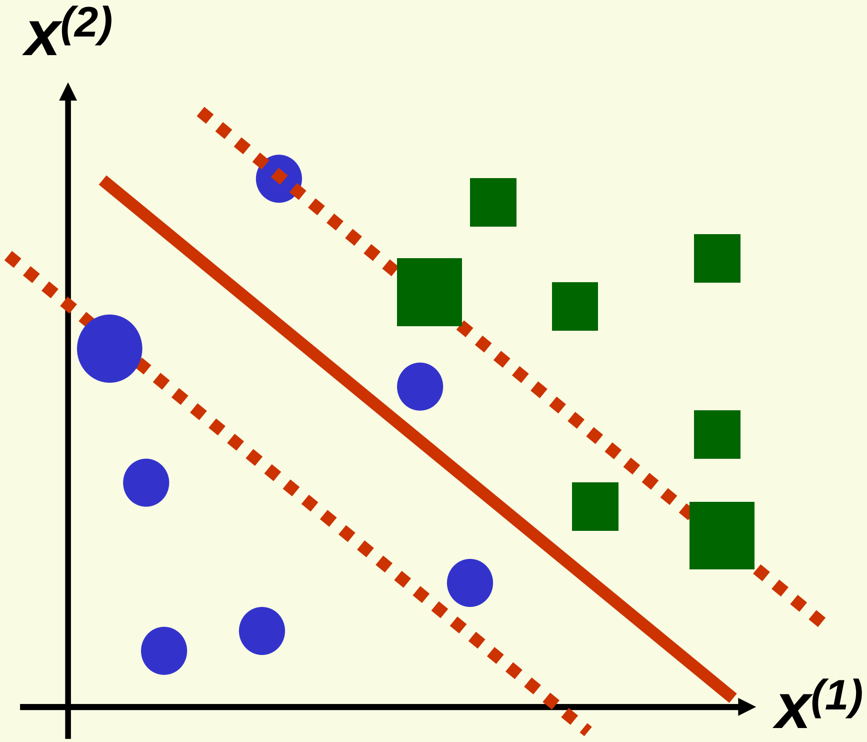
- Would like to minimize

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

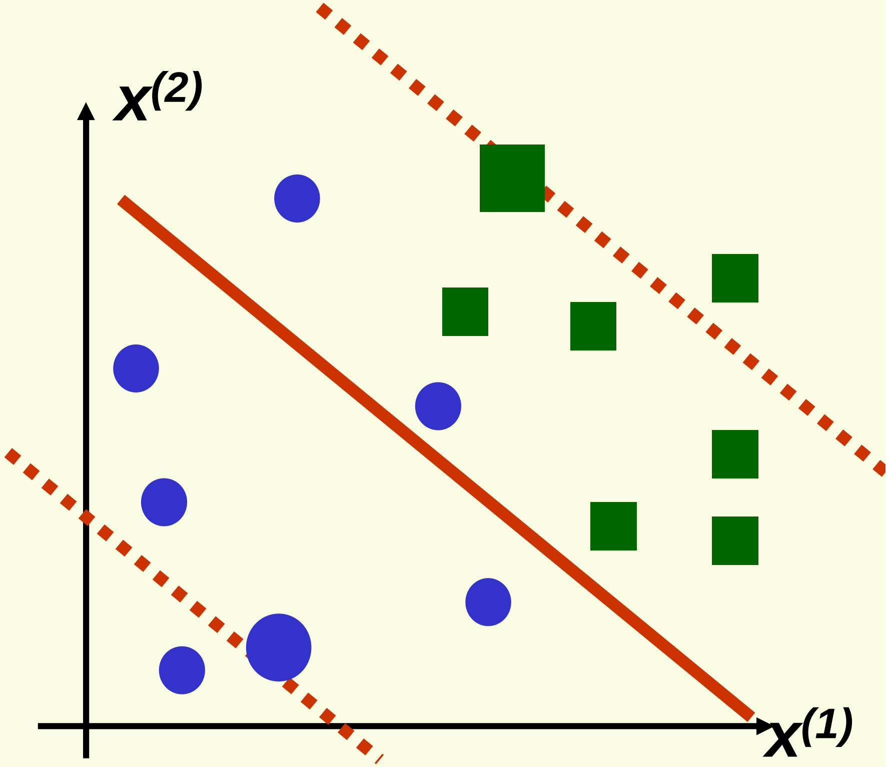
  - constrained to $y_i(\mathbf{w}^t \mathbf{x}_i + b) \geq 1 - \xi_i$ and $\xi_i \geq 0 \quad \forall i$
  - $\mathbf{C} > \mathbf{0}$ is a constant which measures relative weight of the first and second terms
    - if C is small, we allow a lot of samples not in ideal position
    - if C is large, we 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 $L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^t x_j$

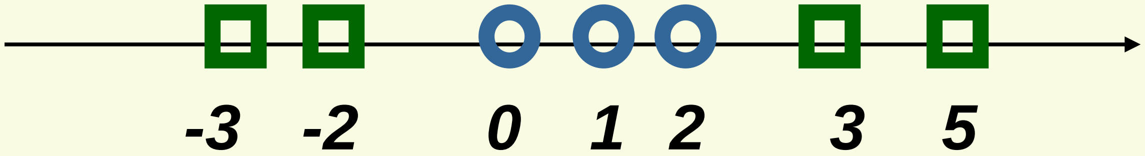
constrained to $0 \le \alpha_i \le C \quad \forall i \quad \textit{and} \quad \sum_{i=1}^{n} \alpha_i y_i = 0$

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

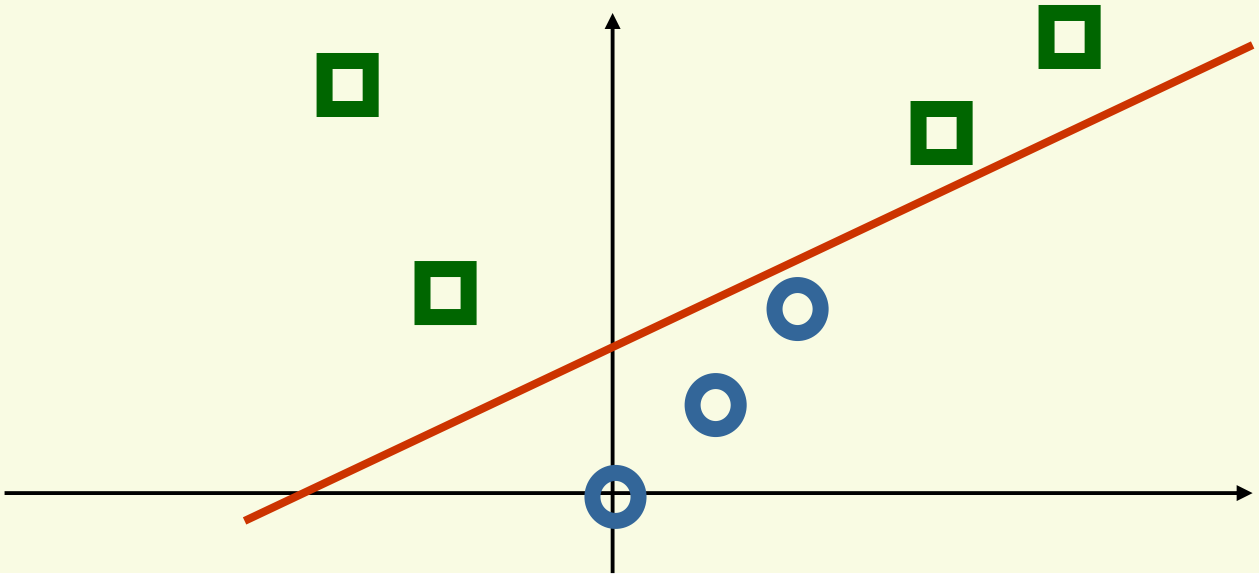
Non Linear Mapping

- 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

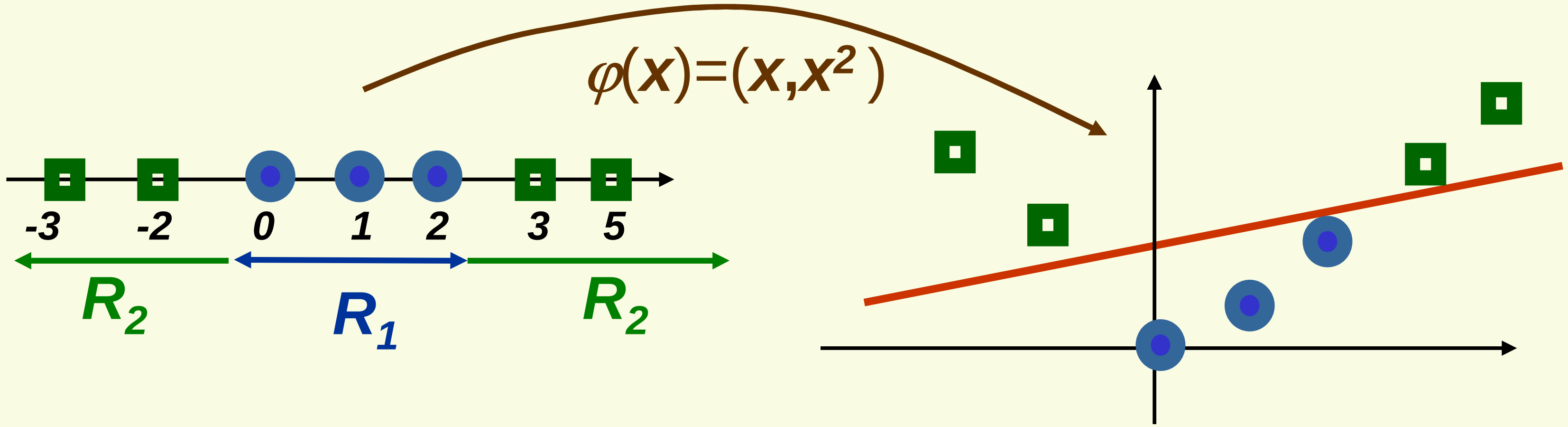


- Lift to two dimensional space with $\varphi(\boldsymbol{x})=(\boldsymbol{x},\boldsymbol{x}^2)$



Non Linear Mapping

- To 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 $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{w}^t \varphi(\boldsymbol{x}) + \boldsymbol{w}_0$

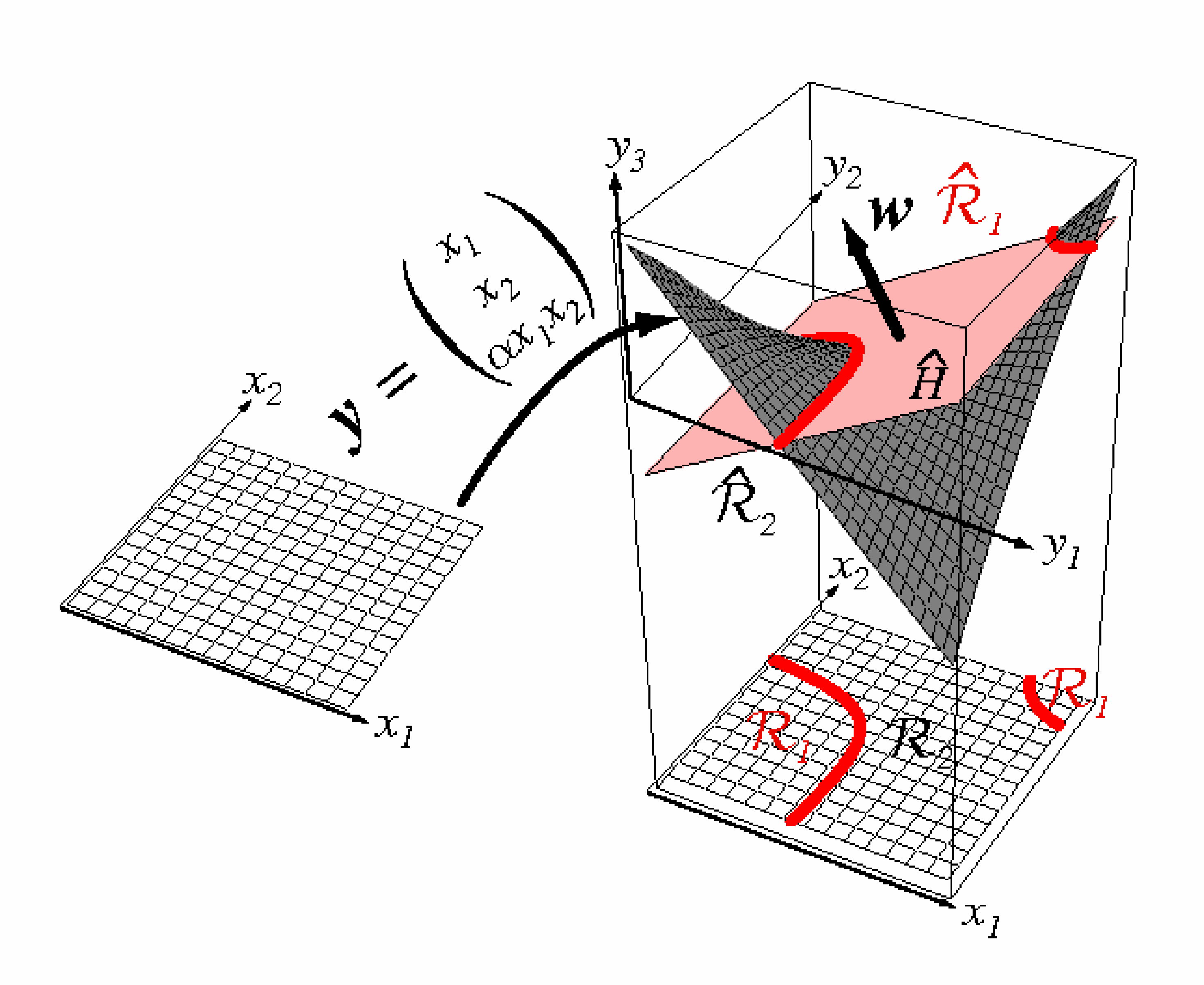


- In 2D, discriminant function is linear

$$\boldsymbol{g}\left(\begin{bmatrix} \boldsymbol{x}^{(1)} \\ \boldsymbol{x}^{(2)} \end{bmatrix}\right) = \begin{bmatrix} \boldsymbol{w}_1 & \boldsymbol{w}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{x}^{(1)} \\ \boldsymbol{x}^{(2)} \end{bmatrix} + \boldsymbol{w}_0$$

- In 1D, discriminant function is not linear $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{w}_1\boldsymbol{x} + \boldsymbol{w}_2\boldsymbol{x}^2 + \boldsymbol{w}_0$

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 avoids the "curse of dimensionality" problems by
  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 $$L_D(\alpha)=\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_i y_i y_j \mathbf{x}_i^t\mathbf{x}_j$$

- Note this optimization depends on samples $\mathbf{x}_i$ only through the dot product $\mathbf{x}_i^t\mathbf{x}_j$
- If we lift $\mathbf{x}_i$ to high dimensional space F using $\varphi(\mathbf{x})$, need to compute high dimensional product $\varphi(\mathbf{x}_i)^t\varphi(\mathbf{x}_j)$

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

- 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.

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

- Now we only need to compute $\boldsymbol{K}(\boldsymbol{x}_i,\boldsymbol{x}_j)$ 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:

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

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

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

$$\mathbf{x} = (x_1, x_2);$$

$$\mathbf{z} = (z_1, z_2);$$

$$K(\mathbf{x}, \mathbf{z}) \equiv \langle \mathbf{x}, \mathbf{z} \rangle^2 = (x_1 z_1 + x_2 z_2)^2 =$$

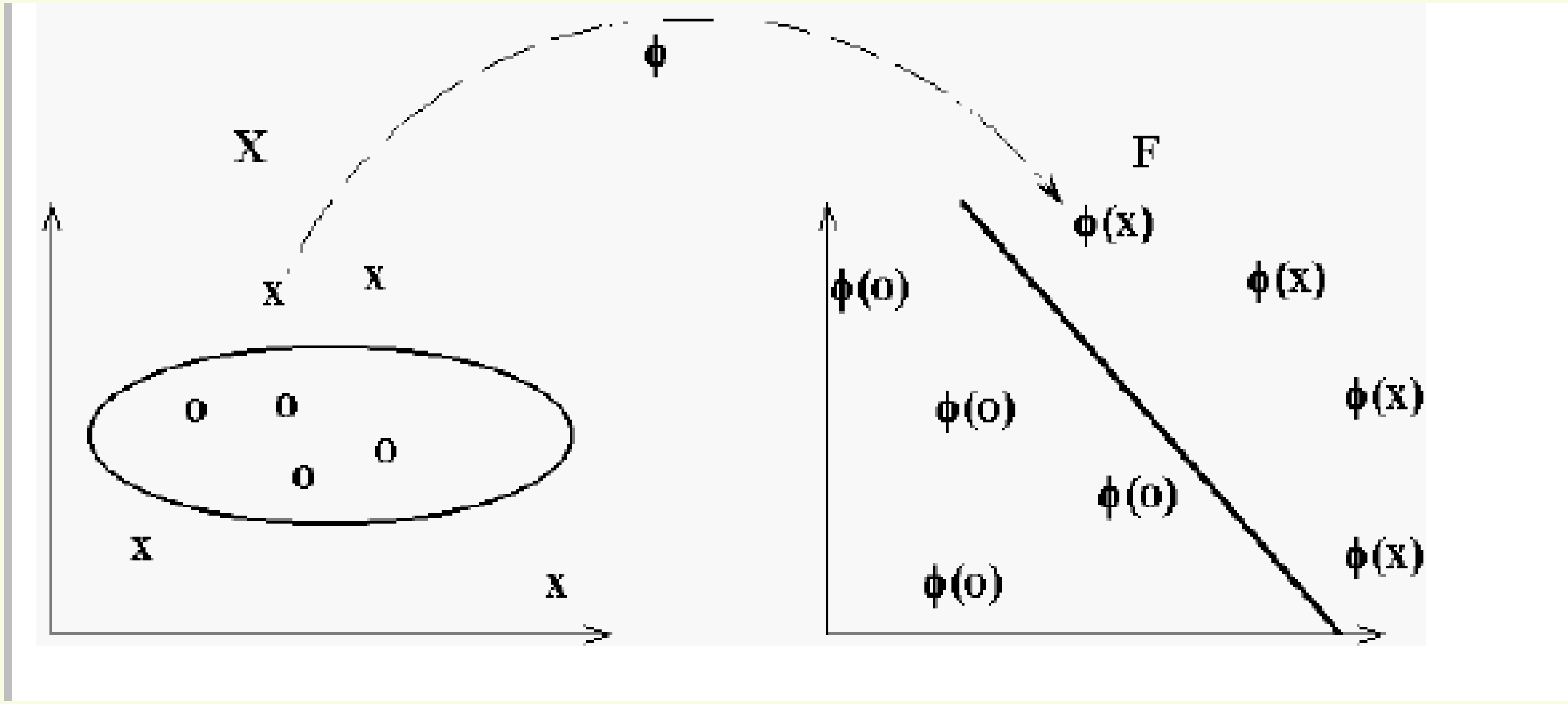
$$x_1^2 z_1^2 + x_2^2 z_2^2 + 2 x_1 z_1 x_2 z_2 =$$

$$\left\langle (x_1^2, x_2^2, \sqrt{2} x_1 x_2), (z_1^2, z_2^2, \sqrt{2} z_1 z_2) \right\rangle =$$

$$\langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle$$

From www.support-vector.net

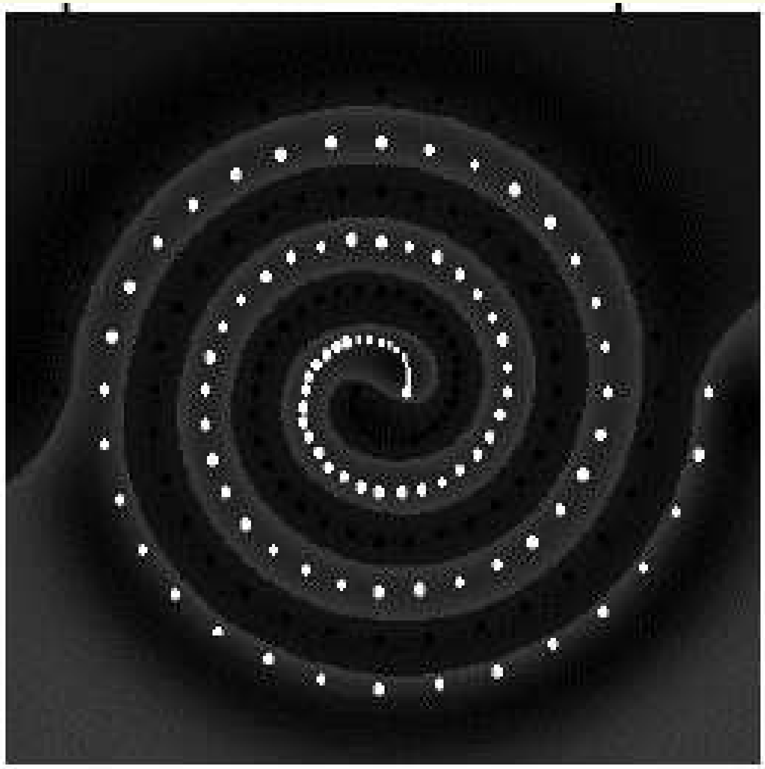
Example Polynomial Kernels



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

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

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

Non Linear SVM Recipe

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

$$\boldsymbol{w} = \sum_{i}^{n} \alpha_i \boldsymbol{y}_i \varphi(\boldsymbol{x}_i)$$

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

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

- Non linear discriminant function in the original space:

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

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

Non Linear SVM

- Nonlinear discriminant function

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

$$g(x) = \sum$$

most important training samples, i.e. support vectors

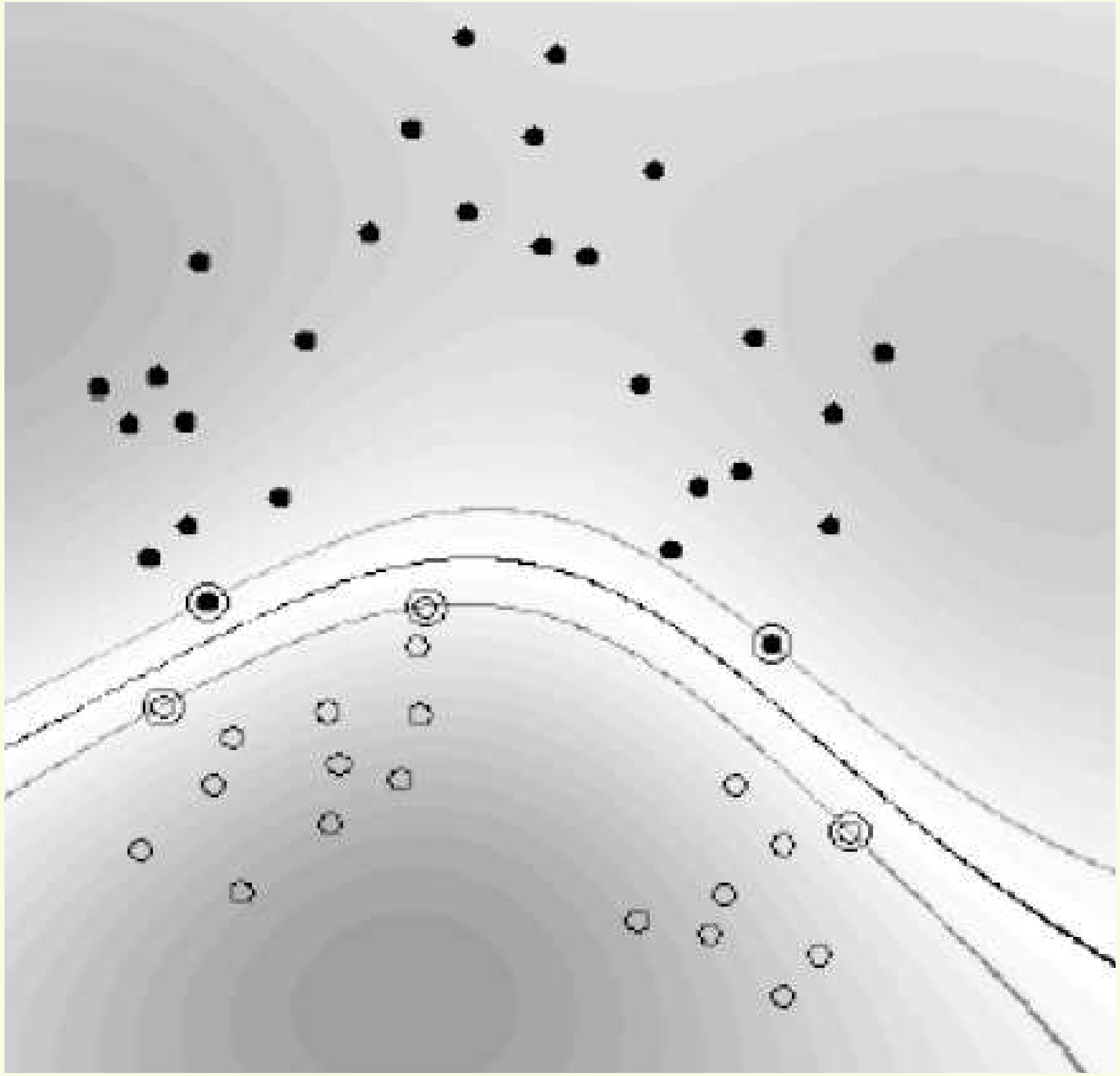
weight of support vector x_i

∓ 1

"inverse distance" from x to support vector x_i

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

Toy Example with a Gaussian Kernel

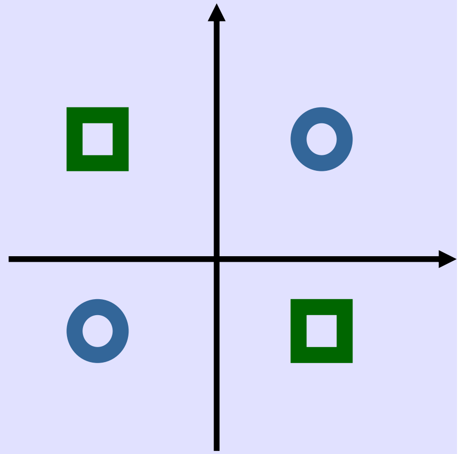


Higher Order Polynomials
Taken from Andrew Moore

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 Example: XOR Problem



- Class 1: $\mathbf{x_1} = [1,-1]$, $\mathbf{x_2} = [-1,1]$
- Class 2: $\mathbf{x_3} = [1,1]$, $\mathbf{x_4} = [-1,-1]$
- Use polynomial kernel of degree 2:
  - $K(x_i,x_j) = (x_i^t x_j + 1)^2$
  - This kernel corresponds to mapping

$$\varphi(x) = \left[1 \quad \sqrt{2}x^{(1)} \quad \sqrt{2}x^{(2)} \quad \sqrt{2}x^{(1)}x^{(2)} \quad \left(x^{(1)}\right)^2 \quad \left(x^{(2)}\right)^2\right]^t$$

- Need to maximize

$$L_D(\alpha) = \sum_{i=1}^{4}\alpha_i - \frac{1}{2}\sum_{i=1}^{4}\sum_{j=1}^{4}\alpha_i\alpha_j z_i z_j \left(x_i^t x_j + 1\right)^2$$

constrained to $0 \le \alpha_i \quad \forall i \quad \textit{and} \quad \alpha_1 + \alpha_2 - \alpha_3 - \alpha_4 = 0$

SVM Example: XOR Problem

- Can rewrite $L_D(\alpha)=\sum_{i=1}^{4}\alpha_i-\frac{1}{2}\alpha^t H\alpha$
  - $H_{ij}=z_i z_j K(x_i,x_j)$ where $K(x_i,x_j) = (x_i^t x_j + 1)^2$
  - Thus $H=\begin{bmatrix} 9 & 1 & -1 & -1 \\ 1 & 9 & -1 & -1 \\ -1 & -1 & 9 & 1 \\ -1 & -1 & 1 & 9 \end{bmatrix}$
- Take derivative with respect to α and set it to 0

$$\frac{d}{da}L_D(\alpha)=\begin{bmatrix}1\\1\\1\\1\end{bmatrix}-\begin{bmatrix} 9 & 1 & -1 & -1 \\ 1 & 9 & -1 & -1 \\ -1 & -1 & 9 & 1 \\ -1 & -1 & 1 & 9 \end{bmatrix}\alpha=0$$

- Solution to the above is $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = 0.25$
  - satisfies the constraints $\forall i,\ 0\le\alpha_i$ *and* $\alpha_1+\alpha_2-\alpha_3-\alpha_4=0$
  - all samples are support vectors

SVM Example: XOR Problem

$$\varphi(\mathbf{x}) = \begin{bmatrix} 1 & \sqrt{2}\mathbf{x}^{(1)} & \sqrt{2}\mathbf{x}^{(2)} & \sqrt{2}\mathbf{x}^{(1)}\mathbf{x}^{(2)} & \left(\mathbf{x}^{(1)}\right)^2 & \left(\mathbf{x}^{(2)}\right)^2 \end{bmatrix}^t$$

- Class 1: $\mathbf{x}_1 = [1,-1]$, $\mathbf{x}_2 = [-1,1]$
- Class 2: $\mathbf{x}_3 = [1,1]$, $\mathbf{x}_4 = [-1,-1]$

- Weight vector ***w*** is:

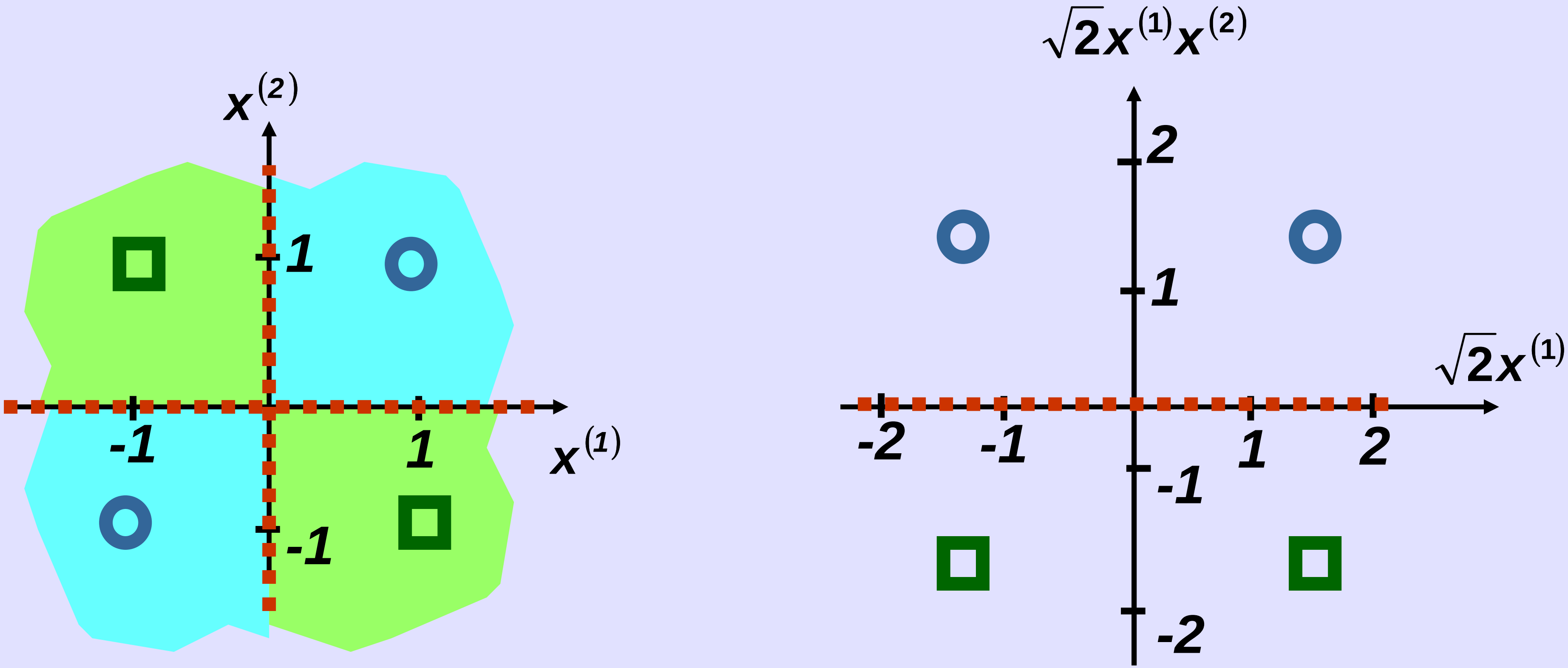
$$\mathbf{w} = \sum_{i=1}^{4} \alpha_i z_i \varphi(\mathbf{x}_i) = 0.25\left(\varphi(\mathbf{x}_1) + \varphi(\mathbf{x}_2) - \varphi(\mathbf{x}_3) - \varphi(\mathbf{x}_4)\right)$$
$$= \begin{bmatrix} 0 & 0 & 0 & -\sqrt{2} & 0 & 0 \end{bmatrix}$$

- Thus the nonlinear discriminant function is:

$$g(\mathbf{x}) = \mathbf{w}\varphi(\mathbf{x}) = \sum_{i=1}^{6} w_i \varphi_i(\mathbf{x}) = -\sqrt{2}\left(\sqrt{2}\mathbf{x}^{(1)}\mathbf{x}^{(2)}\right) = -2\mathbf{x}^{(1)}\mathbf{x}^{(2)}$$

SVM Example: XOR Problem

$$g(x) = -2x^{(1)}x^{(2)}$$



decision boundaries nonlinear

decision boundary is linear

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