CS840a
Learning and Computer Vision
Prof. Olga Veksler

Lecture 8

SVM

Some pictures from C. Burges

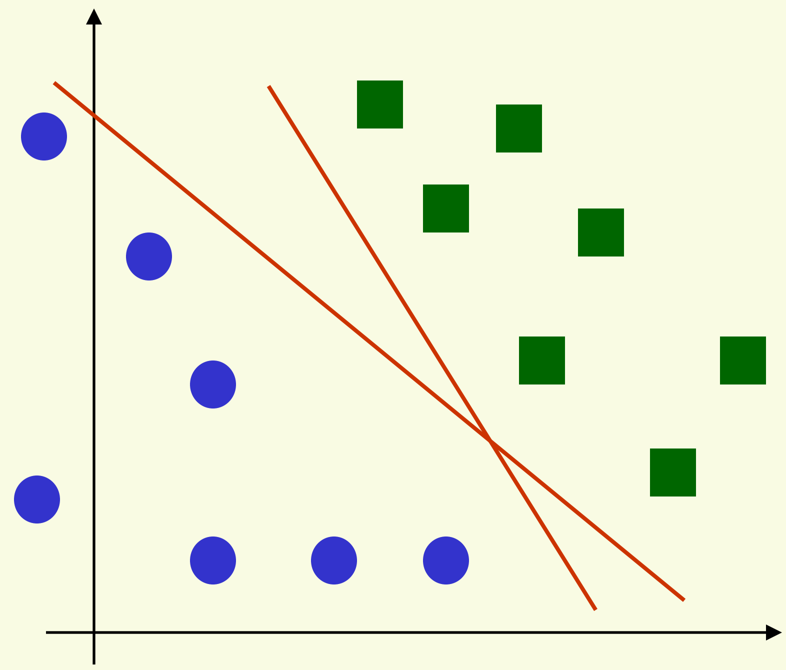
SVM

- 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 15-20 years

Linear Discriminant Functions

$$g(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + w_0$$

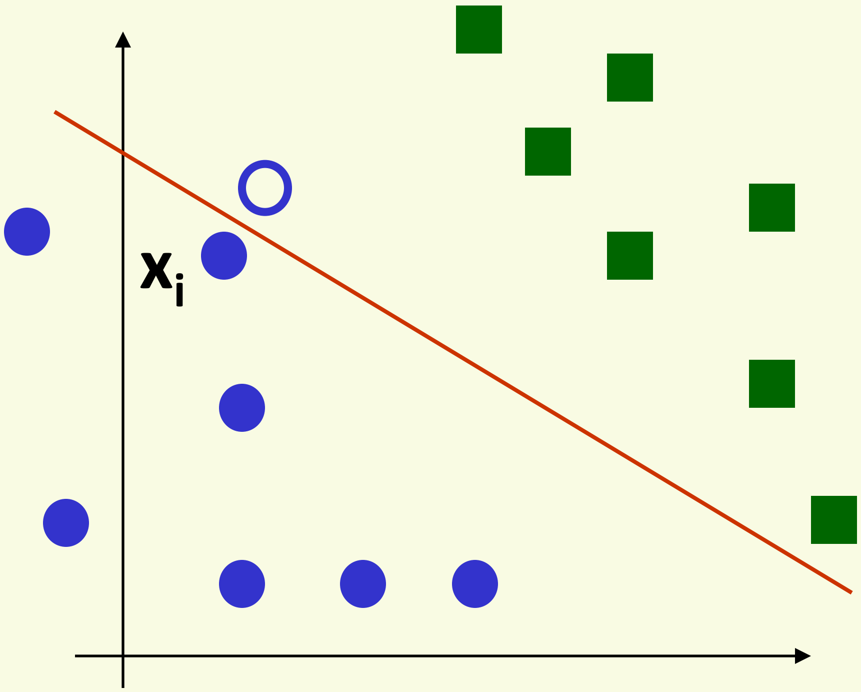
$$\boxed{\begin{array}{l} g(\mathbf{x}) > 0 \quad \Rightarrow \mathbf{x} \in \textbf{class } 1 \\ g(\mathbf{x}) < 0 \quad \Rightarrow \mathbf{x} \in \textbf{class } 2 \end{array}}$$



- which separating hyperplane should we choose?

Margin Intuition

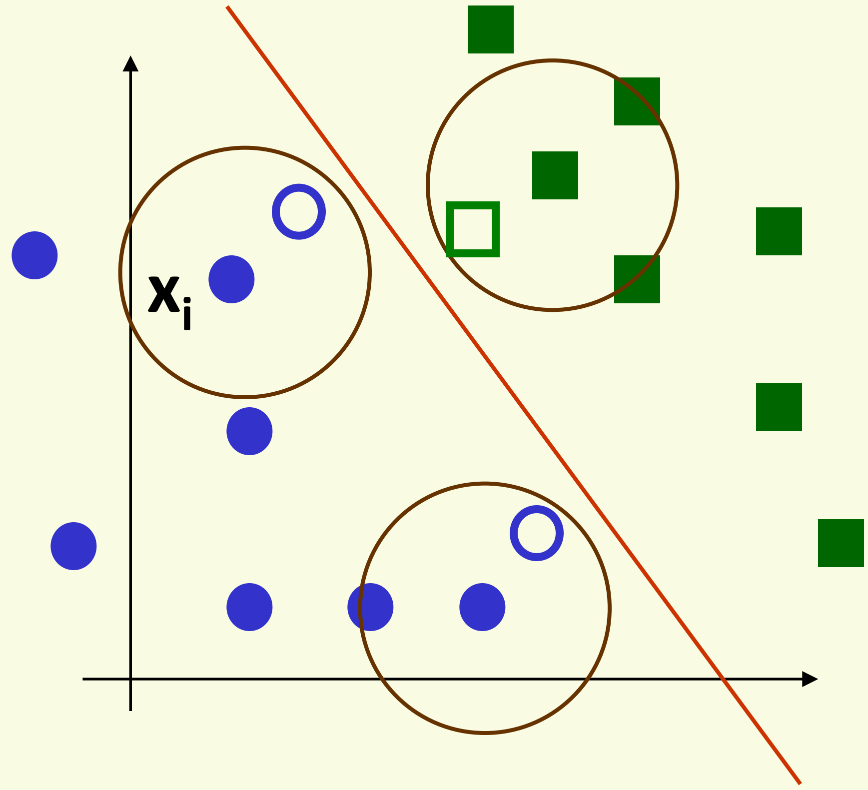
- Training data is just a subset of of all possible data
- Suppose hyperplane is close to sample $\mathbf{x}_i$
- If sample is close to sample $\mathbf{x}_i$, it is likely to be on the wrong side



- Poor generalization

Margin Intuition

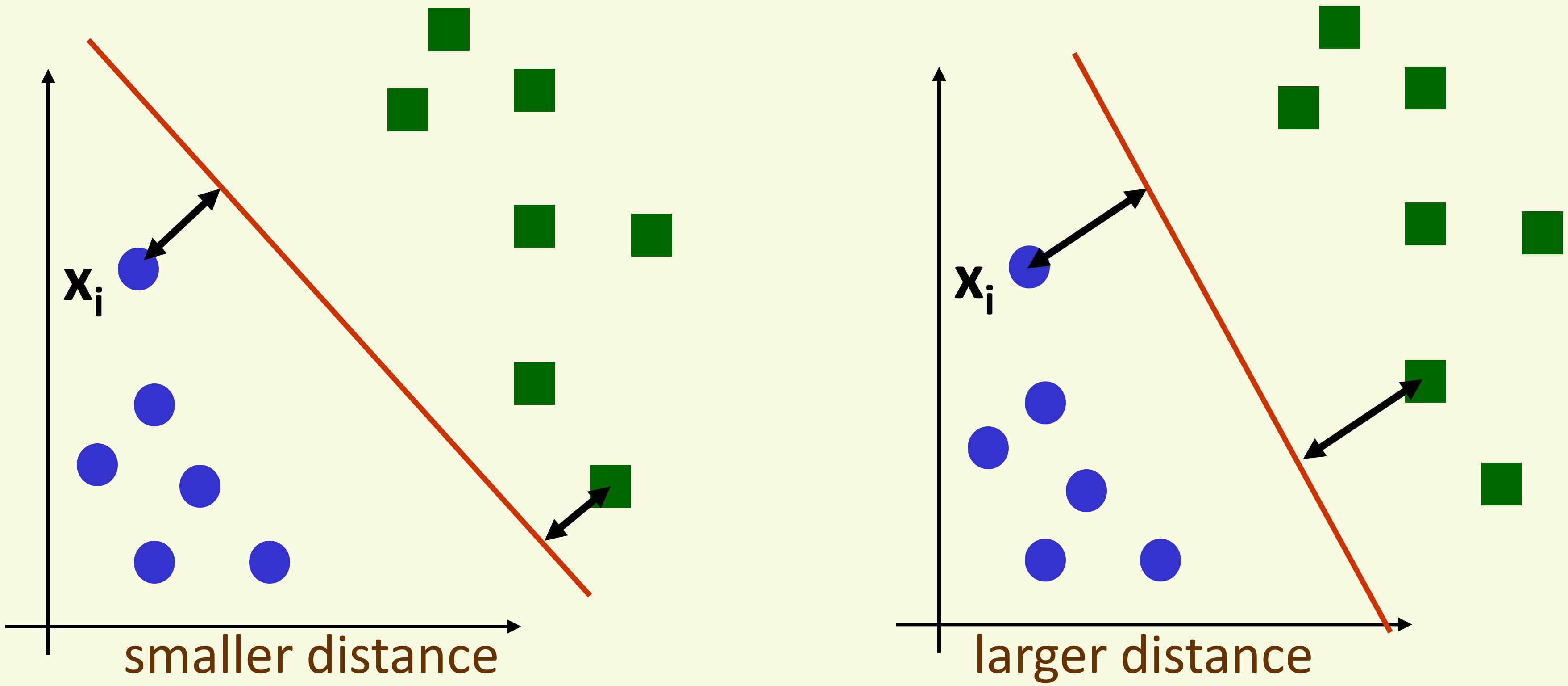
- Hyperplane as far as possible from any sample



- More likely that new samples close to old samples classified correctly
- Good generalization

SVM

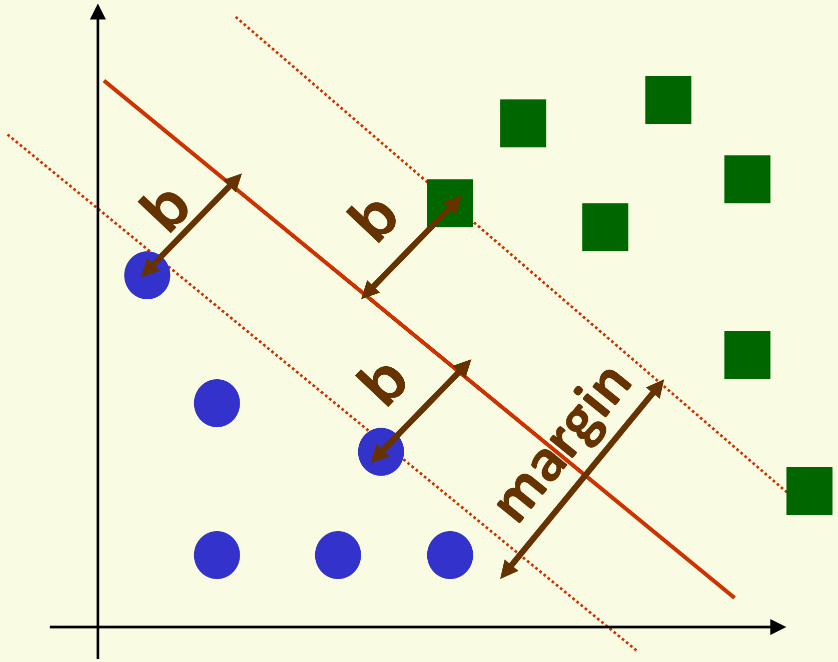
- Idea: maximize distance to the closest example



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

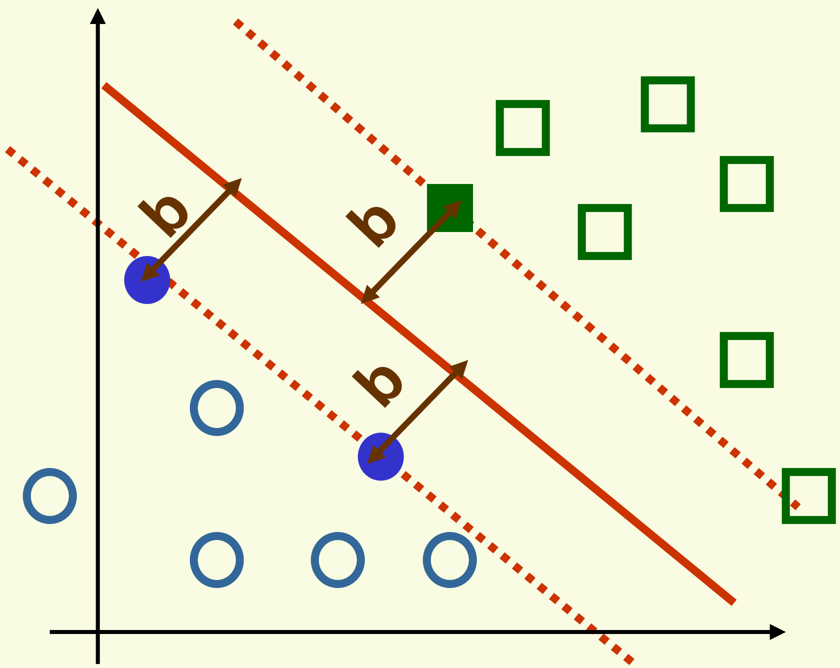
SVM: Linearly Separable Case

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



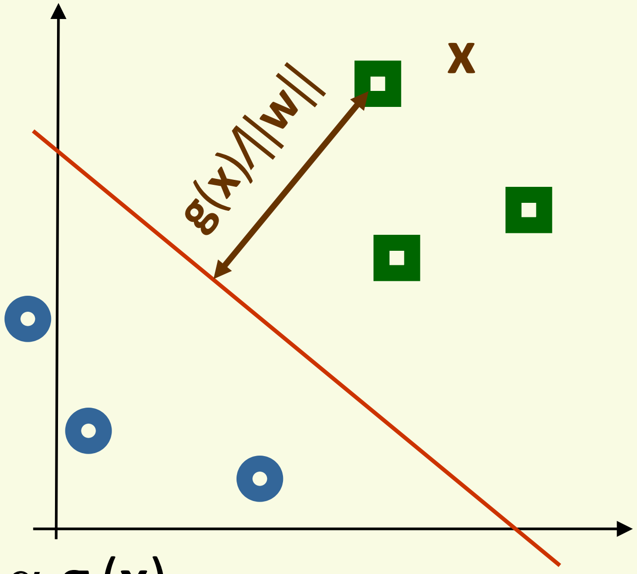
- ***margin*** is twice the absolute value of distance ***b*** of the closest example to the separating hyperplane
- Better generalization
  - in practice and in theory

SVM: Linearly Separable Case



- ***Support vectors*** are samples closest to separating hyperplane
  - they are the most difficult patterns to classify, intuitively
  - optimal hyperplane is completely defined by support vectors
    - do not know which samples are support vectors beforehand

SVM: Formula for the Margin



- $\mathbf{g}(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + \mathbf{w_0}$
- absolute distance between **x** and the boundary $\mathbf{g}(\mathbf{x}) = \mathbf{0}$

$$\frac{\left|w^t x + w_0\right|}{\left\|w\right\|}$$

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

$$\frac{\left|\alpha\,\mathbf{w}^t\mathbf{x} + \alpha\,\mathbf{w}_0\right|}{\left\|\alpha\,\mathbf{w}\right\|} = \frac{\left|\mathbf{w}^t\mathbf{x} + \mathbf{w}_0\right|}{\left\|\mathbf{w}\right\|}$$

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

$$\left|\mathbf{w}^t\mathbf{x}_i + \mathbf{w}_0\right| = 1$$

- Now the largest margin hyperplane is unique

SVM: Formula for the Margin

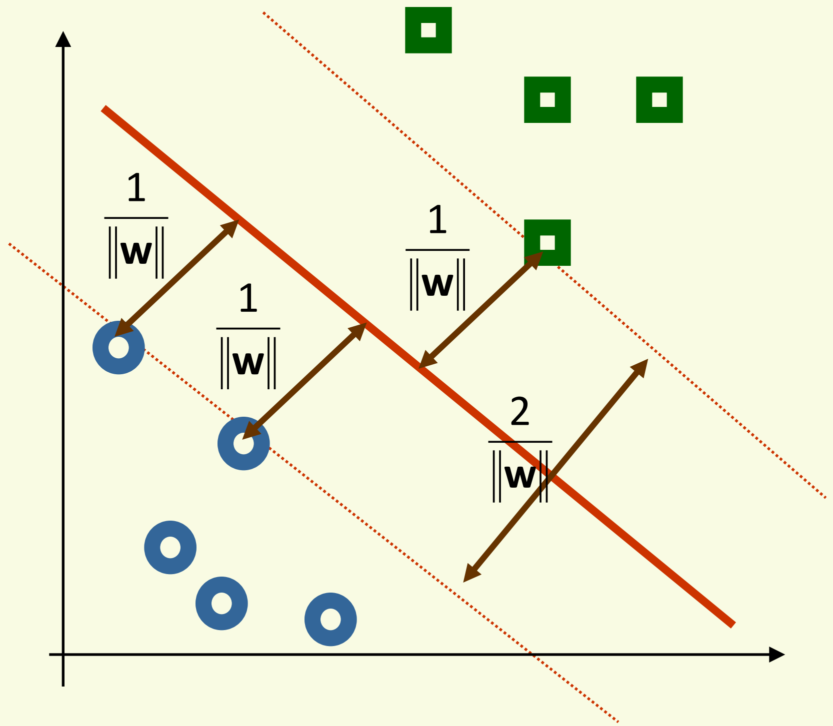
- For uniqueness, set $\left|\mathbf{w}^t\mathbf{x}_i + \mathbf{w}_0\right| = 1$ for any example $\mathbf{x}_i$ closest to the boundary

- now distance from closest sample $\mathbf{x}_i$ to $\mathbf{g}(\mathbf{x}) = 0$ is

$$\frac{\left|\mathbf{w}^t\mathbf{x}_i + \mathbf{w}_0\right|}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|}$$

- Thus the margin is

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



SVM: Optimal Hyperplane

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

$$\begin{cases} \mathbf{w}^t \mathbf{x}_i + \mathbf{w}_0 \geq 1 & \textbf{if } \mathbf{x}_i \textbf{ is positive example} \\ \mathbf{w}^t \mathbf{x}_i + \mathbf{w}_0 \leq -1 & \textbf{if } \mathbf{x}_i \textbf{ is negative example} \end{cases}$$

- Let $\begin{cases} \mathbf{z}_i = 1 & \textbf{if } \mathbf{x}_i \textbf{ is positive example} \\ \mathbf{z}_i = -1 & \textbf{if } \mathbf{x}_i \textbf{ is negative example} \end{cases}$

- Convert our problem to

$$\boxed{\begin{array}{ll} \text{minimize} & \mathbf{J}(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{constrained to} & \mathbf{z}^i\left(\mathbf{w}^t \mathbf{x}_i + \mathbf{w}_0\right) \geq 1 \quad \forall \mathbf{i} \end{array}}$$

- **J**(**w**) is a convex function, thus it has a single global minimum

SVM: Optimal Hyperplane

- Use Kuhn-Tucker theorem to convert our problem to:

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

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

- $\alpha = \{\alpha_1, \ldots, \alpha_n\}$ are new variables, one for each sample
- Rewrite $L_D(\alpha)$ using n by n matrix H:

$$L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\begin{bmatrix}\alpha_1 \\ \vdots \\ \alpha_n\end{bmatrix}^t H \begin{bmatrix}\alpha_1 \\ \vdots \\ \alpha_n\end{bmatrix}$$

  - where the value in the i th row and j th column of H is

$$H_{ij} = z_i z_j x_i^t x_j$$

SVM: Optimal Hyperplane

- Use Kuhn-Tucker theorem to convert our problem to:

maximize	$L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \mathbf{z}_i \mathbf{z}_j \mathbf{x}_i^t \mathbf{x}_j$
constrained to	$\alpha_i \geq 0 \quad \forall \mathbf{i} \quad \textbf{and} \quad \sum_{i=1}^{n} \alpha_i \mathbf{z}_i = 0$

- $\boldsymbol{\alpha} = \{\alpha_1, \dots, \alpha_n\}$ are new variables, one for each sample
- $\mathbf{L}_D(\boldsymbol{\alpha})$ can be optimized by quadratic programming
- $\mathbf{L}_D(\boldsymbol{\alpha})$ formulated in terms of $\boldsymbol{\alpha}$
  - depends on $\mathbf{w}$ and $\mathbf{w}_0$

SVM: Optimal Hyperplane

- After finding the optimal $\alpha = \{\alpha_1, \dots, \alpha_n\}$
  - for every sample **i**, one of the following must hold
    - $\alpha_i = 0$ (sample ***i*** is not a support vector)
    - $\alpha_i \neq 0$ and $z_i(w^t x_i + w_0 - 1) = 0$ (sample **i** is support vector)
  - compute $\mathbf{w} = \sum_{i=1}^{n} \alpha_i z_i \mathbf{x}_i$
  - solve for w_0 using any $\alpha_i > 0$ and $\alpha_i \left[z_i \left(\mathbf{w}^t \mathbf{x}_i + w_0 \right) - 1 \right] = 0$

$$w_0 = \frac{1}{z_i} - \mathbf{w}^t \mathbf{x}_i$$

- Final discriminant function:

$$g(\mathbf{x}) = \left(\sum_{\mathbf{x}_i \in S} \alpha_i z_i \mathbf{x}_i \right)^t \mathbf{x} + w_0$$

  - where ***S*** is the set of support vectors

$$S = \{ \mathbf{x}_i \mid \alpha_i \neq 0 \}$$

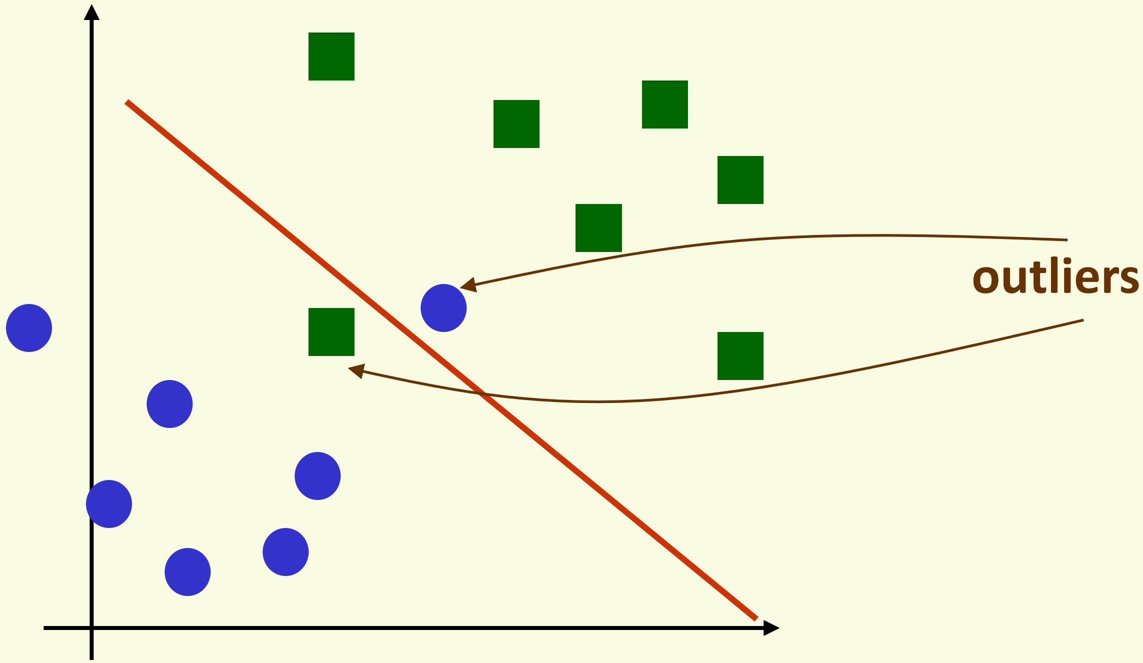
SVM: Optimal Hyperplane

maximize	$L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j z_i z_j \mathbf{x}_i^t \mathbf{x}_j$
constrained to	$\alpha_i \geq 0 \quad \forall i$ **and** $\sum_{i=1}^{n} \alpha_i z_i = 0$

- $L_D(\alpha)$ depends on the number of samples, not on dimension of samples
- samples appear only through the dot products $\mathbf{x}_i^t \mathbf{x}_j$
- Will become important when looking for a ***nonlinear*** discriminant function

SVM: Non Separable Case

- Linear classifier still be appropriate when data is not linearly separable, but almost linearly separable



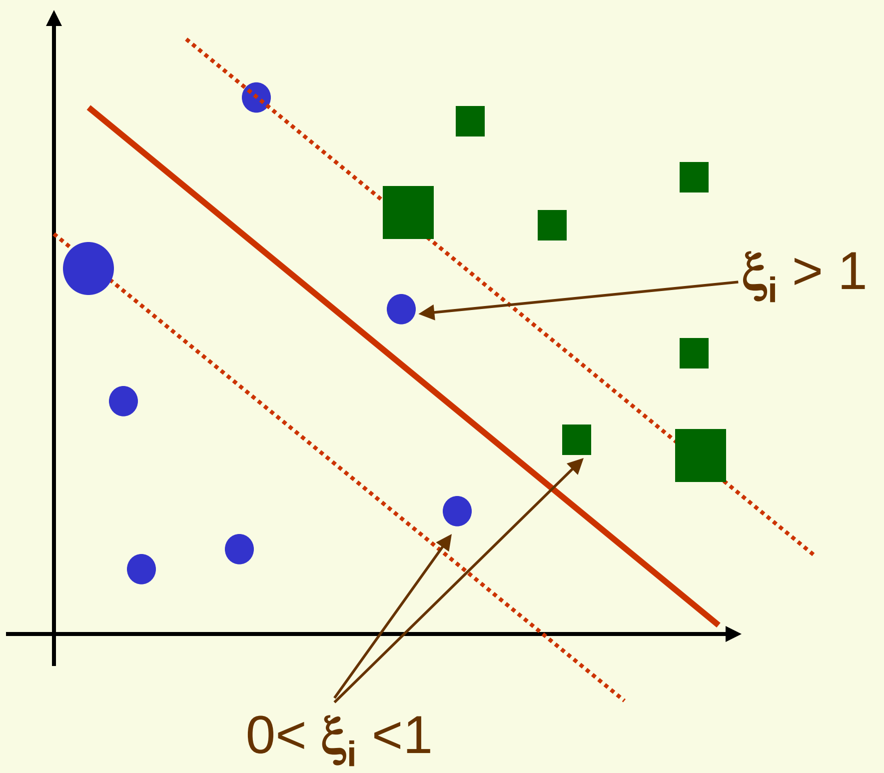
- Can adapt SVM to almost linearly separable case

SVM: Non Separable Case

- Introduce non-negative *slack* variables $\xi_1, \ldots, \xi_n$
  - one for each sample
- Change constraints from $z_i(\mathbf{w}^t \mathbf{x}_i + w_0) \geq 1 \quad \forall i$ to

$$z_i(\mathbf{w}^t \mathbf{x}_i + w_0) \geq 1 - \xi_i \quad \forall i$$

- ξ_i measures deviation from the ideal position for sample $\mathbf{x}_i$
  - $\xi_i > 1$: $\mathbf{x}_i$ is on the wrong side of the hyperplane
  - $0 < \xi_i < 1$: $\mathbf{x}_i$ is on the right side of the hyperplane but within the region of maximum margin



SVM: Non Separable Case

- Wish to minimize

$$J(\mathbf{w},\xi_1,\ldots,\xi_n)=\frac{1}{2}\|\mathbf{w}\|^2+\beta\sum_{i=1}^{n}I(\xi_i>0)$$

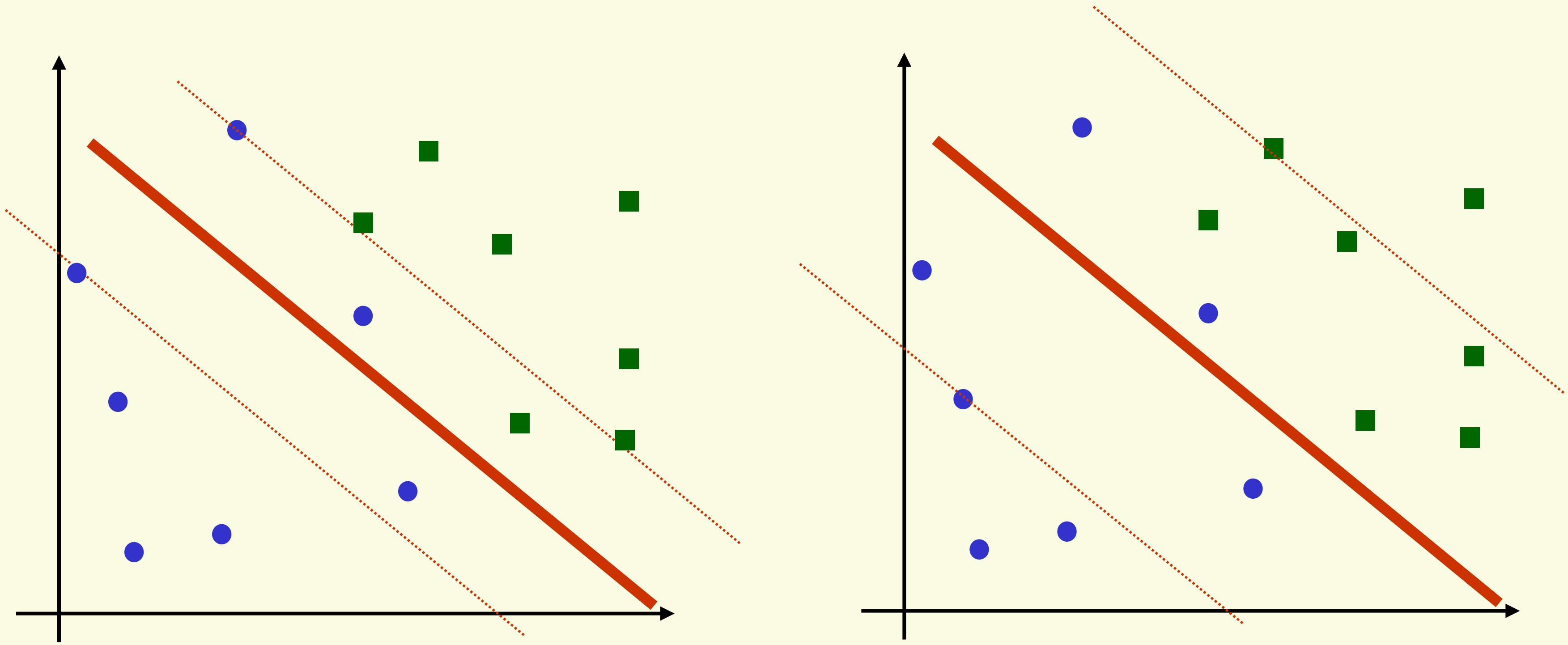
# of samples not in ideal location

- where $I(\xi_i>0)=\begin{cases}1 & \textbf{if } \xi_i>0\\ 0 & \textbf{if } \xi_i\le 0\end{cases}$
- constrained to $z_i(\mathbf{w}^t\mathbf{x}_i+w_0)\ge 1-\xi_i$ and $\xi_i\ge 0\ \ \forall i$
- β measures relative weight of first and second terms
  - if β is small, we allow a lot of samples not in ideal position
  - if β is large, we allow very few samples not in ideal position
  - choosing β appropriately is important

SVM: Non Separable Case

$$J(\mathbf{w}, \xi_1, \ldots, \xi_\mathbf{n}) = \frac{1}{2}\|\mathbf{w}\|^2 + \beta \boxed{\sum_{\mathbf{i}=1}^{\mathbf{n}} \mathbf{I}(\xi_\mathbf{i} > 0)}$$

# of samples
not in ideal location



large β, few samples not in ideal position

small β, many samples not in ideal position

SVM: Non Separable Case

- Minimization problem is NP-hard due to discontinuity of $\mathbf{I}(\xi_i)$

$$J(\mathbf{w}, \xi_1, \dots, \xi_n) = \frac{1}{2}\|\mathbf{w}\|^2 + \beta \sum_{i=1}^{n} \mathbf{I}(\xi_i > 0)$$

\# of samples
not in ideal location

- where $\mathbf{I}(\xi_i > 0) = \begin{cases} 1 & \textbf{if } \xi_i > 0 \\ 0 & \textbf{if } \xi_i \le 0 \end{cases}$
- constrained to $\mathbf{z}_i(\mathbf{w}^t \mathbf{x}_i + \mathbf{w}_0) \ge 1 - \xi_i$ and $\xi_i \ge 0 \quad \forall \mathbf{i}$

SVM: Non Separable Case

- Instead we minimize

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

a measure of
# of misclassified
examples

- constrained to $\begin{cases} z_i(\mathbf{w}^t\mathbf{x}_i+w_0)\geq 1-\xi_i & \forall i \\ \xi_i\geq 0 & \forall i\end{cases}$

- Use Kuhn-Tucker theorem to converted to

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

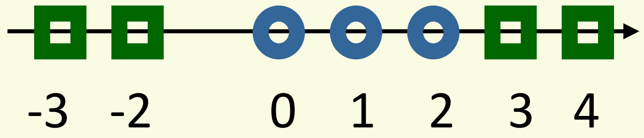
constrained to $\quad 0\leq\alpha_i\leq\beta \quad \forall i$ **and** $\sum_{i=1}^{n}\alpha_i z_i=0$

- find **w** using $\quad \mathbf{w}=\sum_{i=1}^{n}\alpha_i z_i \mathbf{x}_i$
- solve for $\mathbf{w_0}$ using any $0<\alpha_i<\beta$ and $\alpha_i\left[z_i(\mathbf{w}^t\mathbf{x}_i+w_0)-1\right]=0$

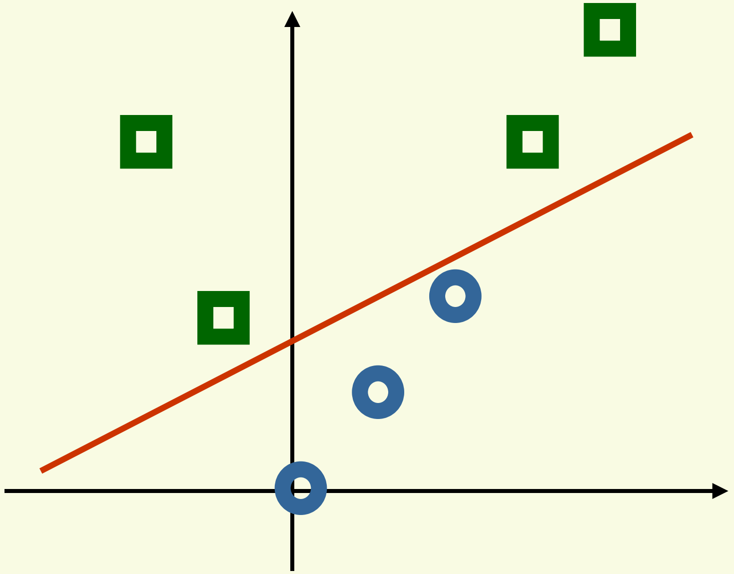
Non Linear Mapping

- 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"*

- Not linearly separable in 1D

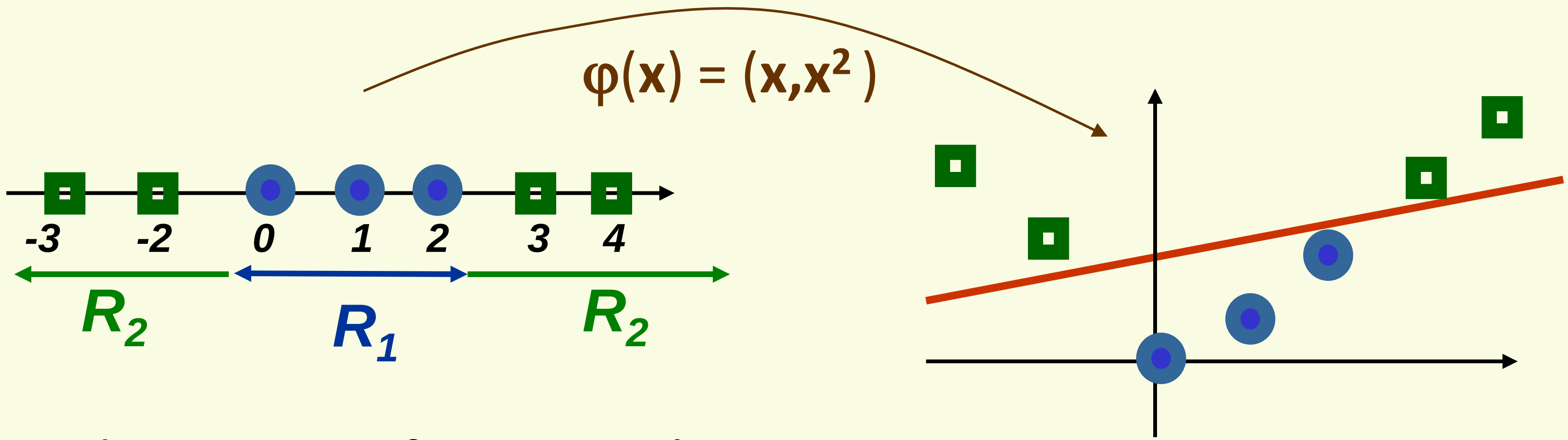


- Lift to 2D space with $\mathbf{h}(\mathbf{x}) = (\mathbf{x},\mathbf{x}^2)$



Non Linear Mapping

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

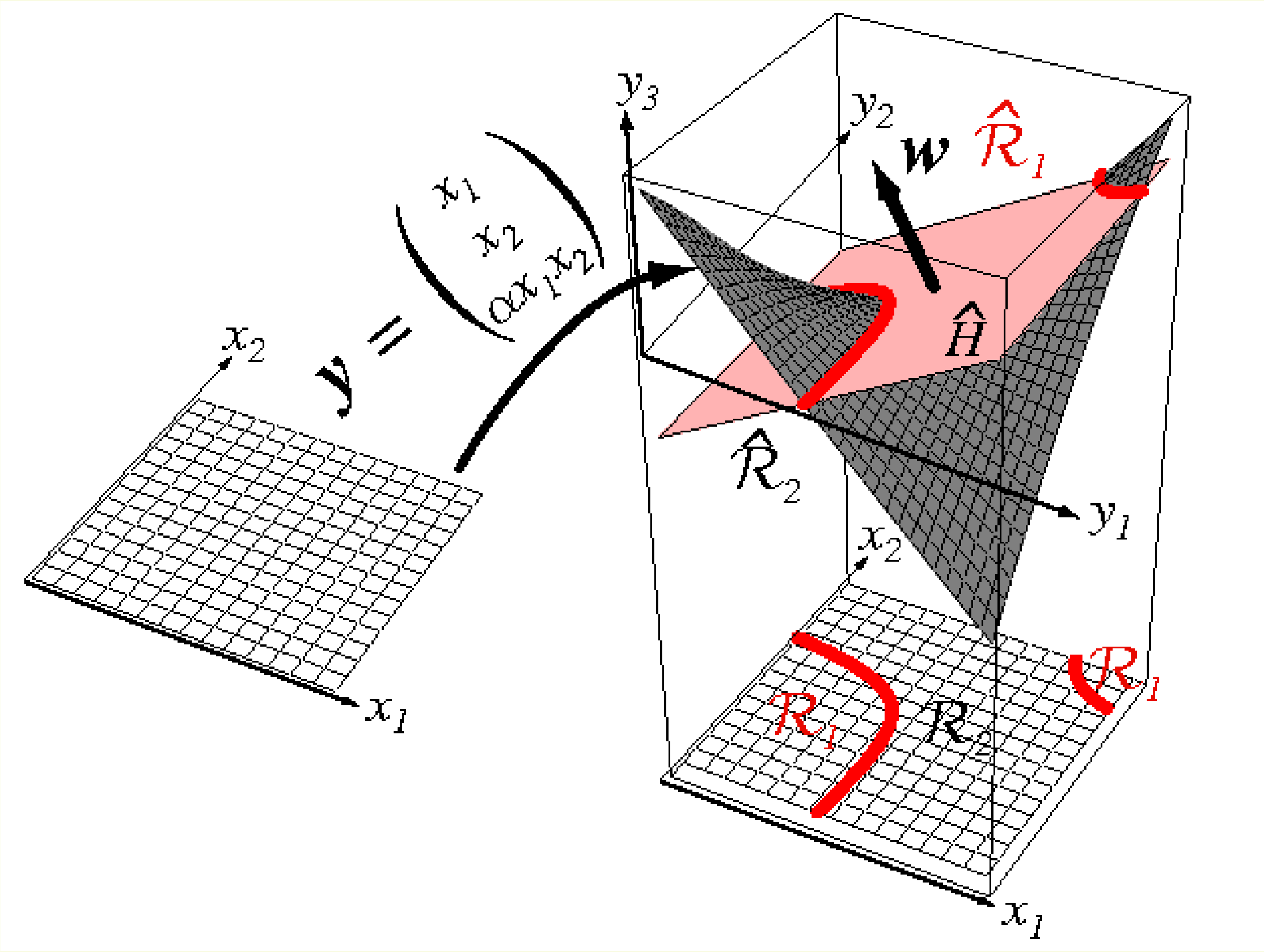


- In 2D, discriminant function is linear

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

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

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” by
  - enforcing largest margin permits good generalization
  - computation in the higher dimensional case is performed only implicitly through the use of ***kernel*** functions

Non Linear SVM: Kernels

- Recall SVM optimization

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

- Optimization depends on samples $\boldsymbol{x}_i$ only through the dot product $\boldsymbol{x}_i^t\boldsymbol{x}_j$

- If we lift $\boldsymbol{x}_i$ to high dimension using $\boldsymbol{\varphi}(\mathbf{x})$, need to compute high dimensional product $\boldsymbol{\varphi}(\boldsymbol{x}_i)^t\boldsymbol{\varphi}(\boldsymbol{x}_j)$

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

- Idea: find ***kernel*** function $\mathbf{K}(\mathbf{x}_i,\mathbf{x}_j)$ s.t. $\mathbf{K}(\mathbf{x}_i,\mathbf{x}_j) = \boldsymbol{\varphi}(\mathbf{x}_i)^t\boldsymbol{\varphi}(\mathbf{x}_j)$

Non Linear SVM: Kernels

maximize $$\mathbf{L_D}(\alpha) = \sum_{\mathbf{i}=1}^{\mathbf{n}} \alpha_{\mathbf{i}} - \frac{1}{2}\sum_{\mathbf{i}=1}^{\mathbf{n}}\sum_{\mathbf{j}=1}^{\mathbf{n}} \alpha_{\mathbf{i}}\alpha_{\mathbf{i}}\mathbf{z}_{\mathbf{i}}\mathbf{z}_{\mathbf{j}}\underbrace{\varphi(\mathbf{x}_{\mathbf{i}})^{\mathbf{t}}\varphi(\mathbf{x}_{\mathbf{j}})}_{\mathbf{K}(\mathbf{x}_{\mathbf{i}},\mathbf{x}_{\mathbf{j}})}$$

- Kernel trick
  - only need to compute $\mathbf{K}(\mathbf{x}_i,\mathbf{x}_j)$ instead of $\boldsymbol{\varphi}(\mathbf{x}_i)^{\mathbf{t}}\boldsymbol{\varphi}(\mathbf{x}_j)$
  - no need to lift data in high dimension explicitely, computation is performed in the original dimension

Non Linear SVM: Kernels

- Suppose we have 2 features and $\mathbf{K(x,y)} = \mathbf{(x^t y)^2}$

- Which mapping $\boldsymbol{\varphi}(\mathbf{x})$ does it correspond to?

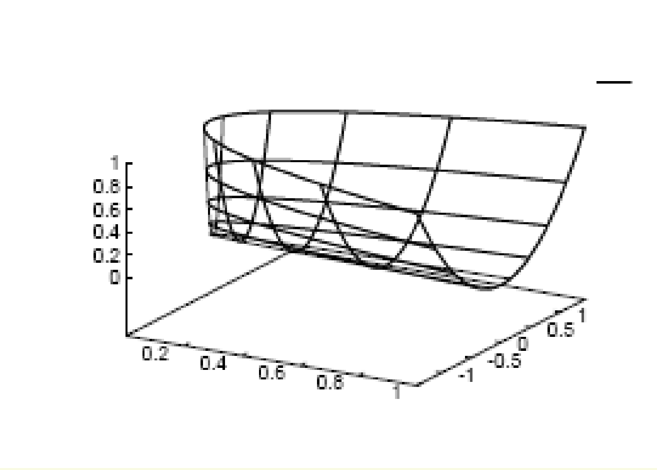
$$\mathbf{K}(\mathbf{x},\mathbf{y}) = \left(\mathbf{x}^t\mathbf{y}\right)^2 = \left(\begin{bmatrix}\mathbf{x}^{(1)} & \mathbf{x}^{(2)}\end{bmatrix}\begin{bmatrix}\mathbf{y}^{(1)} \\ \mathbf{y}^{(2)}\end{bmatrix}\right)^2 = \left(\mathbf{x}^{(1)}\mathbf{y}^{(1)} + \mathbf{x}^{(2)}\mathbf{y}^{(2)}\right)^2$$

$$= \left(\mathbf{x}^{(1)}\mathbf{y}^{(1)}\right)^2 + 2\left(\mathbf{x}^{(1)}\mathbf{y}^{(1)}\right)\left(\mathbf{x}^{(2)}\mathbf{y}^{(2)}\right) + \left(\mathbf{x}^{(2)}\mathbf{y}^{(2)}\right)^2$$

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

- Thus

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



Non Linear SVM: Kernels

- How to choose kernel $\mathbf{K}(\mathbf{x}_i,\mathbf{x}_j)$?
  - $\mathbf{K}(\mathbf{x}_i,\mathbf{x}_j)$ should correspond to product $\boldsymbol{\varphi}(\boldsymbol{x}_i)^t\boldsymbol{\varphi}(\boldsymbol{x}_j)$ in a higher dimensional space
  - Mercer's condition states which kernel function can be expressed as dot product of two vectors
  - Kernel's not satisfying Mercer's condition can be sometimes used, but no geometrical interpretation

- Common choices satisfying Mercer's condition
  - Polynomial kernel $\mathbf{K}(\mathbf{x}_i,\mathbf{x}_j) = (\mathbf{x}_i^t\mathbf{x}_j + 1)^p$
  - Gaussian radial Basis kernel (data is lifted in infinite dimensions)

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

Non Linear SVM

- search for separating hyperplane in high dimension

$$\mathbf{w}\varphi(\mathbf{x}) + \mathbf{w}_0 = 0$$

- Choose $\varphi(\boldsymbol{x})$ so that the first ("0"th) dimension is the augmented dimension with feature value fixed to 1

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

- Threshold $\mathbf{w_0}$ gets folded into vector $\mathbf{w}$

$$\begin{bmatrix} \mathbf{w}_0 & \mathbf{w} \end{bmatrix} \underbrace{\begin{bmatrix} 1 \\ * \end{bmatrix}}_{\varphi(\mathbf{x})} = 0$$

Non Linear SVM

- Thus seeking hyperplane

$$\mathbf{w}\varphi(\mathbf{x}) = 0$$

- Or, equivalently, a hyperplane that goes through the origin in high dimensions
    - removes only one degree of freedom
    - but we introduced many new degrees when lifted the data in high dimension

Non Linear SVM Recepie

- Start with $\mathbf{x}_1,\ldots,\mathbf{x}_n$ in original feature space of dimension $\mathbf{d}$
- Choose kernel $\mathbf{K}(\mathbf{x}_i,\mathbf{x}_j)$
  - implicitly chooses function $\boldsymbol{\varphi}(\mathbf{x}_i)$ that takes $\mathbf{x}_i$ to a higher dimensional space
  - gives dot product in the high dimensional space
- Find largest margin linear classifier in the higher dimensional space by using quadratic programming package to solve

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

constrained to $$0 \le \alpha_i \le \beta \quad \forall \mathbf{i} \quad \textbf{and} \quad \sum_{i=1}^{n} \alpha_i \mathbf{z}_i = 0$$

Non Linear SVM Recipe

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

$$\mathbf{w} = \sum_{\mathbf{x}_i \in \mathbf{S}} \alpha_i \mathbf{z}_i \varphi(\mathbf{x}_i)$$

  - where ***S*** is the set of support vectors

$$\mathbf{S} = \{\mathbf{x}_i \mid \alpha_i \neq 0\}$$

- Linear discriminant function in the high dimensional space

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

- Non linear discriminant function in the original space:

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

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

Non Linear SVM

- Nonlinear discriminant function

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

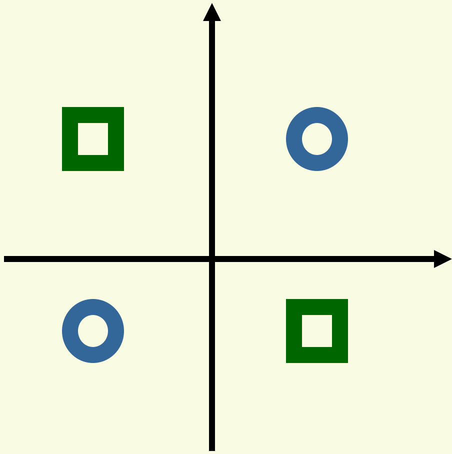
$\mathbf{g}(\mathbf{x}) = \sum$ | weight of support vector $\mathbf{x}_i$ | ∓ 1 | similarity between $\boldsymbol{x}$ and support vector $\mathbf{x}_i$

most important training samples, i.e. support vectors

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

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
  - $\mathbf{K(x_i,x_j) = (x_i{}^t x_j + 1)^2}$
  - Kernel corresponds to mapping



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

- Need to maximize $\mathbf{L_D}(\alpha) = \sum_{\mathbf{i}=1}^{4} \alpha_{\mathbf{i}} \quad \frac{1}{2}\sum_{\mathbf{i}=1}^{4}\sum_{\mathbf{j}=1}^{4} \alpha_{\mathbf{i}}\alpha_{\mathbf{j}}\mathbf{z_i z_j}\left(\mathbf{x_i^t x_j} + 1\right)^2$

  constrained to $0 \le \alpha_{\mathbf{i}} \quad \forall \mathbf{i}$ **and** $\alpha_1 + \alpha_2 - \alpha_3 - \alpha_4 = 0$

SVM Example: XOR Problem

- Rewrite $$\mathbf{L_D}(\alpha)=\sum_{\mathbf{i}=1}^{4}\alpha_{\mathbf{i}}-\frac{1}{2}\alpha^{\mathbf{t}}\mathbf{H}\alpha$$
    - where $\alpha=[\alpha_1 \quad \alpha_2 \quad \alpha_3 \quad \alpha_4]^{\mathbf{t}}$ and $\mathbf{H}=\begin{bmatrix} 9 & 1 & -1 & -1 \\ 1 & 9 & -1 & -1 \\ -1 & -1 & 9 & 1 \\ -1 & -1 & 1 & 9 \end{bmatrix}$
- Take derivative with respect to α and set it to ***0***

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

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

SVM Example: XOR Problem

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

- Weight vector **w** is:

$$\mathbf{w} = \sum_{\mathbf{i}=1}^{4} \alpha_{\mathbf{i}} \mathbf{z}_{\mathbf{i}} \varphi(\mathbf{x}_{\mathbf{i}}) = 0.25\left(\varphi(\mathbf{x}_1) + \varphi(\mathbf{x}_2) - \varphi(\mathbf{x}_3) - \varphi(\mathbf{x}_4)\right)$$

$$= \begin{bmatrix} 0 & 0 & 0 & \sqrt{2} & 0 & 0 \end{bmatrix}$$

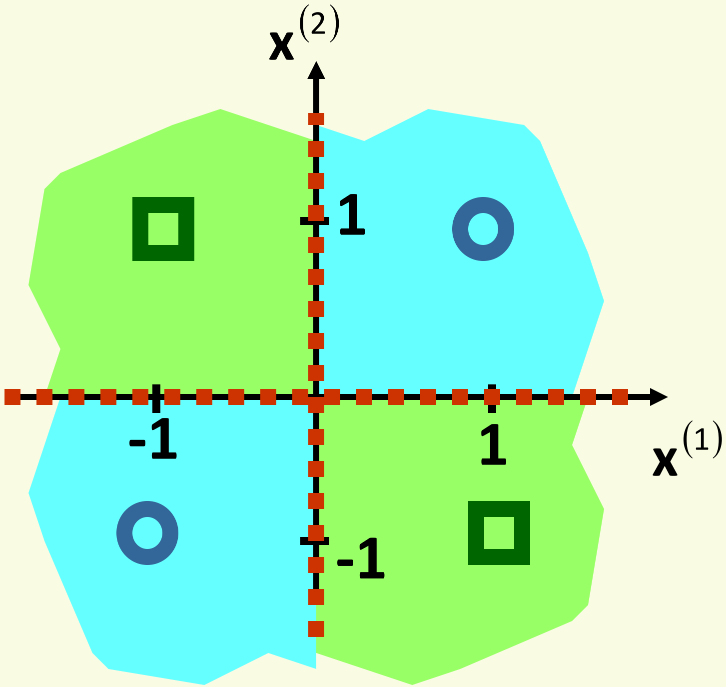
  - by plugging in $\mathbf{x_1}$ = [1,-1], $\mathbf{x_2}$ = [-1,1], $\mathbf{x_3}$ = [1,1], $\mathbf{x_4}$ = [-1,-1]

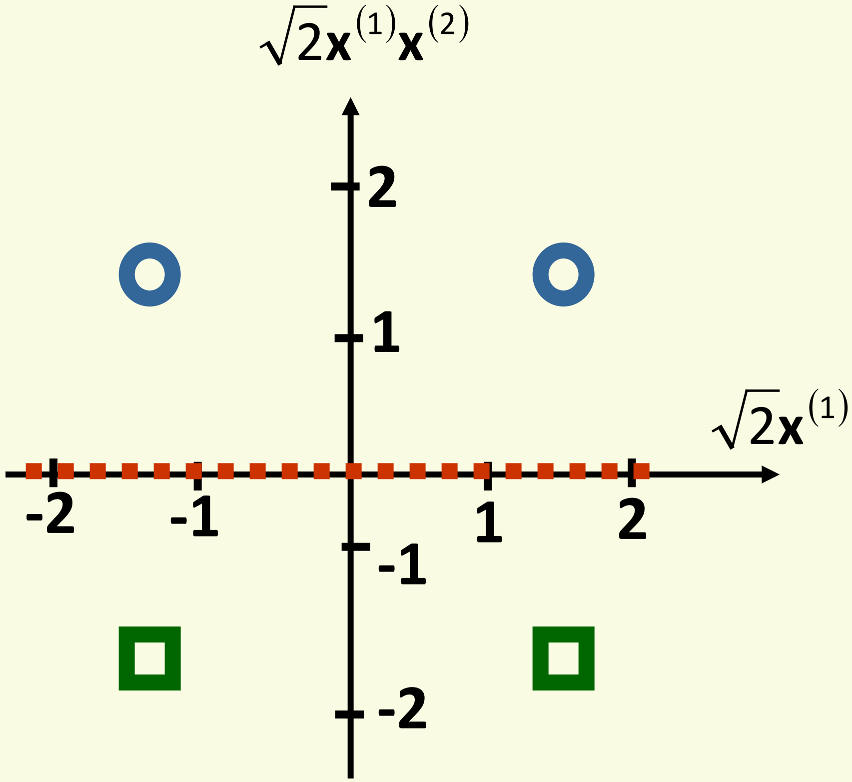
- Nonlinear discriminant function is

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

SVM Example: XOR Problem

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

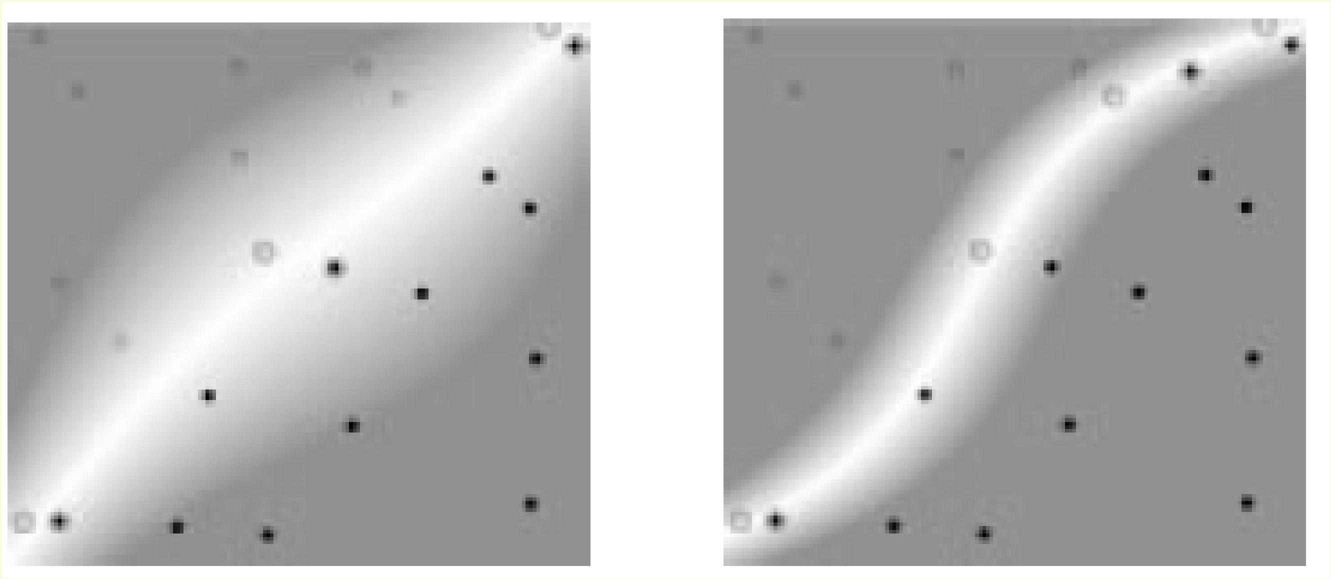




decision boundaries nonlinear

decision boundary is linear

Degree 3 Polynomial Kernel



- Left: In linearly separable case, decision boundary is roughly linear, indicating that dimensionality is controlled
- Right: nonseparable case is handled by a polynomial of degree 3

SVM as Unconstrained Minimization

- SVM formulated as constrained optimization, minimize

$$J(\mathbf{w},\xi_1,\ldots,\xi_\mathbf{n})=\frac{1}{2}\|\mathbf{w}\|^2+\beta\sum_{\mathbf{i}=1}^{\mathbf{n}}\xi_\mathbf{i}$$

  - constrained to $\begin{cases}\mathbf{z_i}\left(\mathbf{w^t x_i}+\mathbf{w}_0\right)\geq 1-\xi_\mathbf{i} & \forall\mathbf{i}\\ \xi_\mathbf{i}\geq 0 & \forall\mathbf{i}\end{cases}$

- Let us name $\mathbf{f}(\mathbf{x_i}) = \mathbf{w^t x_i}+\mathbf{w}_0$
- The constraint can be rewritten as $\begin{cases}\mathbf{z_i f}(\mathbf{x_i})\geq 1-\xi_\mathbf{i} & \forall\mathbf{i}\\ \xi_\mathbf{i}\geq 0 & \forall\mathbf{i}\end{cases}$
- Which implies $\xi_\mathbf{i}=\max(0,1-\mathbf{z_i f}(\mathbf{x_i}))$
- SVM objective can be rewritten as unconstrained optimization

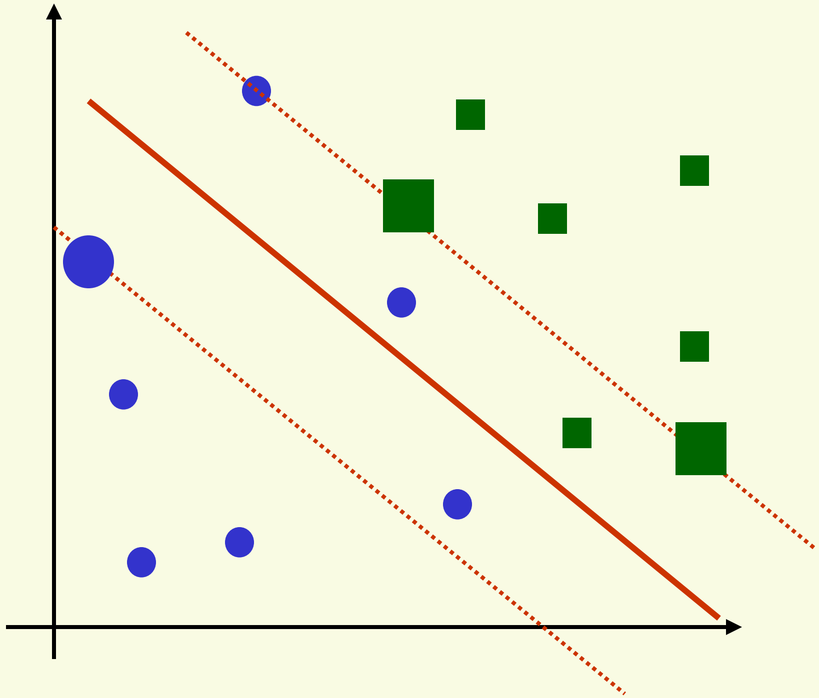
$$J(\mathbf{w},\xi_1,\ldots,\xi_\mathbf{n})=\underbrace{\frac{1}{2}\|\mathbf{w}\|^2}_{\text{weights regularization}}+\beta\underbrace{\sum_{\mathbf{i}=1}^{\mathbf{n}}\max(0,1-\mathbf{z_i f}(\mathbf{x_i}))}_{\text{loss function}}$$

SVM as Unconstrained Minimization

- SVM objective can be rewritten as unconstrained optimization

$$J(\mathbf{w}) = \underbrace{\frac{1}{2}\|\mathbf{w}\|^2}_{\text{weights regularization}} + \beta \sum_{\mathbf{i}=1}^{\mathbf{n}} \underbrace{\max\left(0, 1 - \mathbf{z}_\mathbf{i}\mathbf{f}(\mathbf{x}_\mathbf{i})\right)}_{\text{loss function}}$$

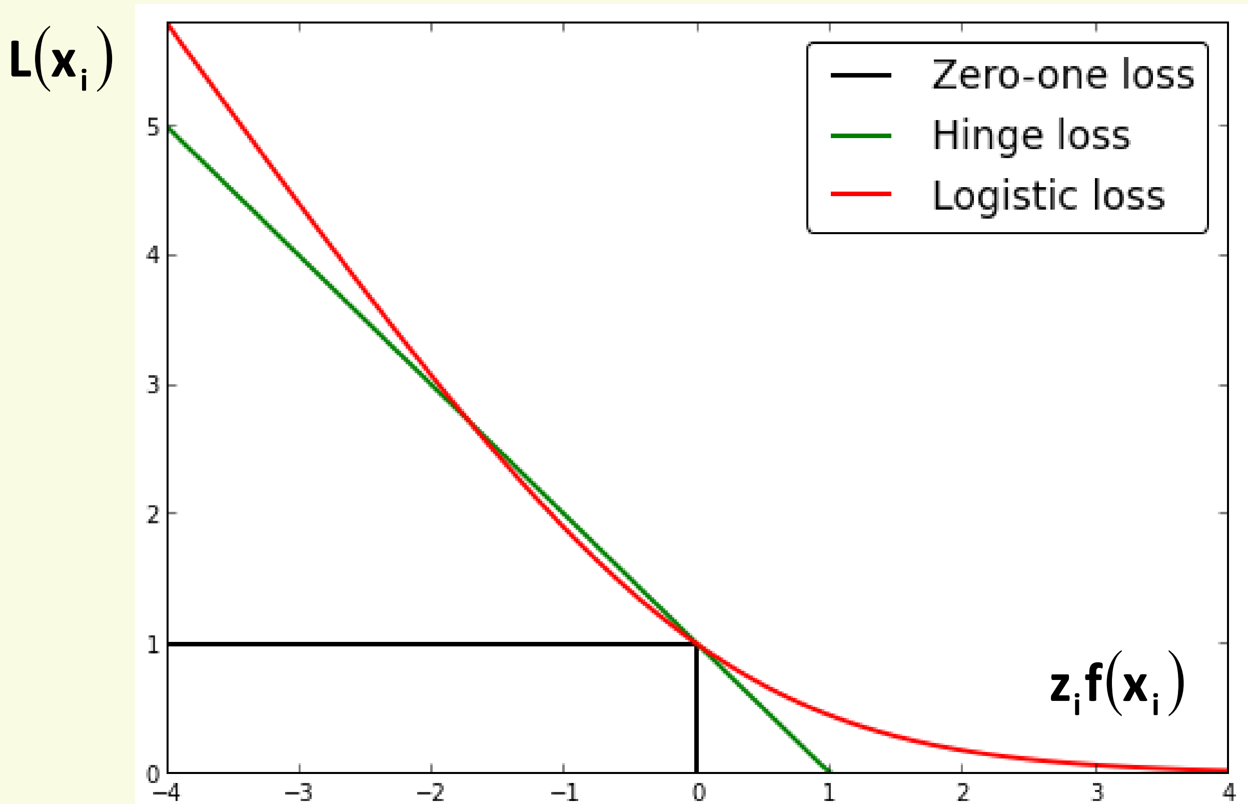
- $\mathbf{z_i}\, f(\mathbf{x_i}) > 1$: $\mathbf{x_i}$ is on the right side of the hyperplane and outside margin, no loss
- $\mathbf{z_i}\, f(\mathbf{x_i}) = 1$: $\mathbf{x_i}$ on the margin, no loss
- $\mathbf{z_i}\, f(\mathbf{x_i}) < 1$: $\mathbf{x_i}$ is inside margin, or on the wrong side of the hyperplane, contributes to loss



SVM: Hinge Loss

- SVM uses Hinge loss per sample $\mathbf{x_i}$

$$\mathbf{L_i}(\mathbf{x_i}) = \max(0, 1 - \mathbf{z_i}\mathbf{f}(\mathbf{x_i}))$$



- Hinge loss encourages classification with a margin of 1

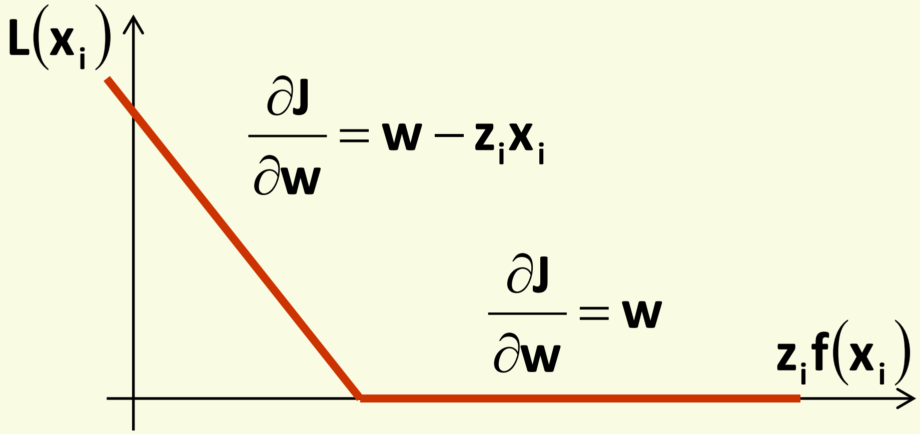
SVM: Hinge Loss

- Can optimize with gradient descent, convex function

$$J(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 + \beta \sum_{i=1}^{n} \max(0, 1 - \mathbf{z}_i \mathbf{f}(\mathbf{x}_i))$$

$$\mathbf{f}(\mathbf{x}_i) = \mathbf{w}^t \mathbf{x}_i + \mathbf{w}_0$$

- Gradient



- Gradient descent, single sample

$$\mathbf{w} = \begin{cases} \mathbf{w} - \alpha(\mathbf{w} - \beta \mathbf{z}_i \mathbf{x}_i) & \textbf{if } \mathbf{z}_i \mathbf{f}(\mathbf{x}_i) < 1 \\ \mathbf{w} - \alpha \mathbf{w} & \textbf{otherwise} \end{cases}$$

SVM Summary

- Advantages:
  - nice theory
  - good generalization properties
  - objective function has no local minima
  - can be used to find non linear discriminant functions
  - often works well in practice, even if not a lot of training data

- Disadvantages:
  - tends to be slower than other methods
  - quadratic programming is computationally expensive