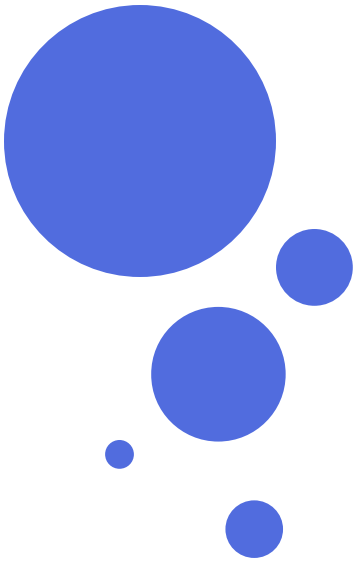
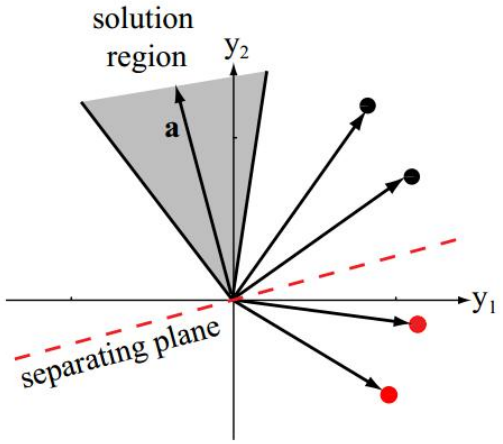
Rensselaer

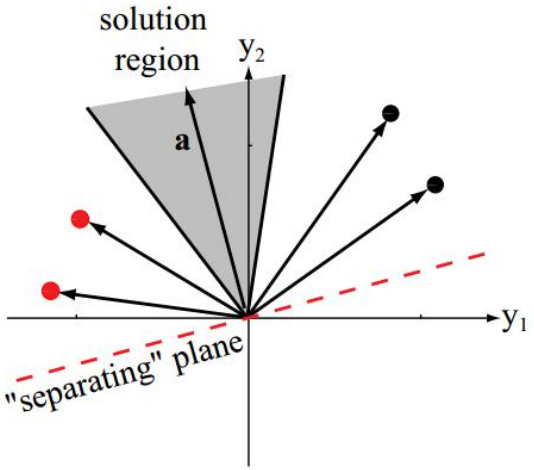


Lecture 11: Support Vector Machine

Dr. Chengjiang Long
Computer Vision Researcher at Kitware Inc.
Adjunct Professor at RPI.
Email: **longc3@rpi.edu**

Recap Previous Lecture





$a^t y_i > 0$ for all samples y_i
solve system of linear inequalities

$a^t y_i = b_i$ for all samples y_i
solve system of linear equations

$$J_{HK}(a,b) = \|Ya - b\|^2$$

$$\nabla_a J_{HK} = 2Y^t(Ya - b) = 0$$

$$\nabla_b J_{HK} = -2(Ya - b) = 0$$

Alternate the two steps below until convergence:

① Fix **b** and minimize $J_{HK}(\mathbf{a},\mathbf{b})$ with respect to **a**

② Fix **a** and minimize $J_{HK}(\mathbf{a},\mathbf{b})$ with respect to **b**

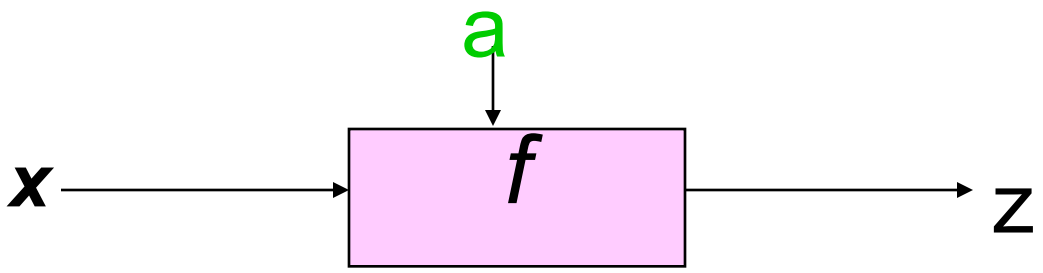
Outline

- Why Maximum Margin？
- Support Vector Machine
- Kernel and Kerneled SVM

Outline

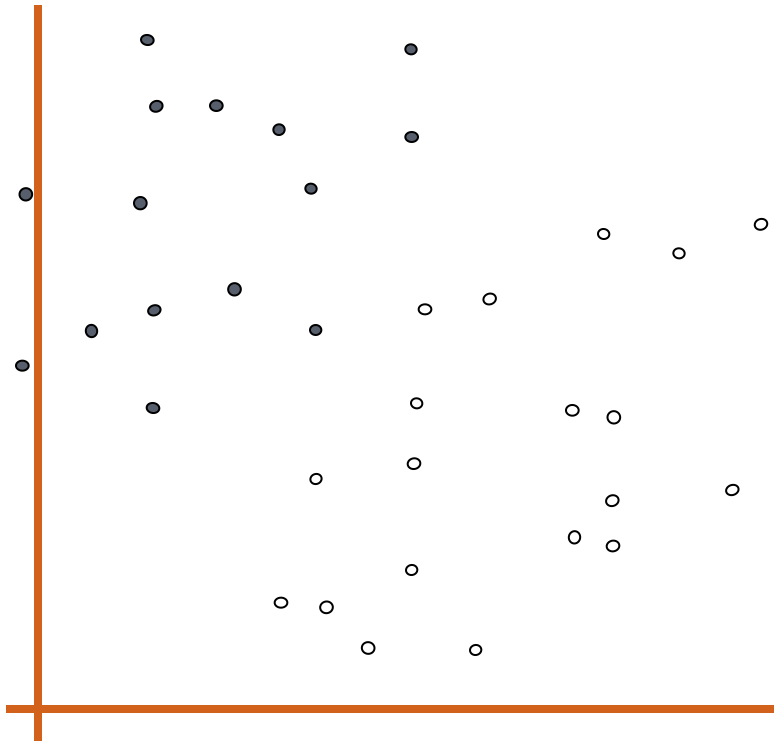
- **Why Maximum Margin？**
- Support Vector Machine
- Kernel and Kerneled SVM

Linear Classifiers



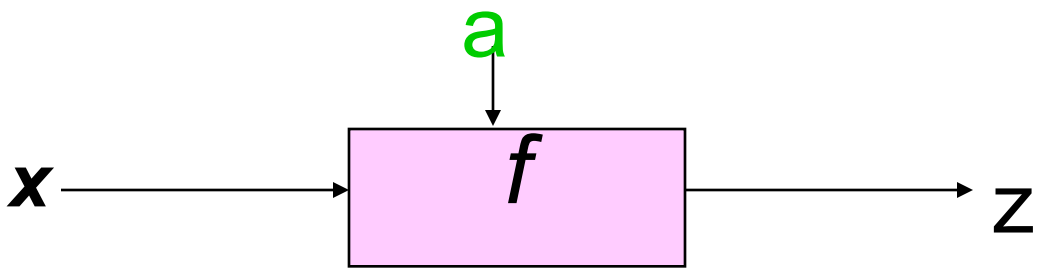
$$f(\mathbf{x}, \mathbf{w}, b) = sign(\mathbf{w} \cdot \mathbf{x} + b)$$

- denotes +1
- denotes -1



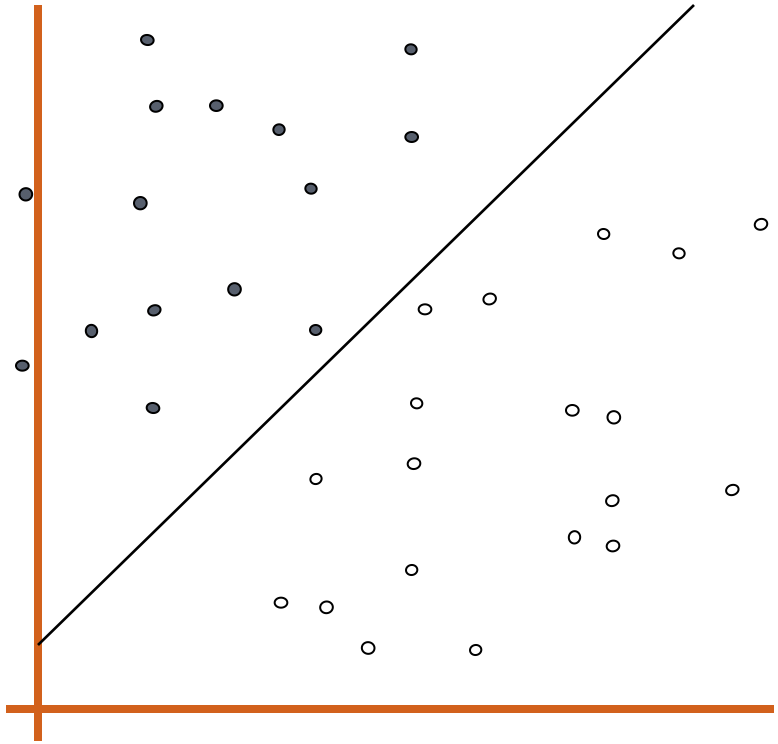
How would you classify this data?

Linear Classifiers



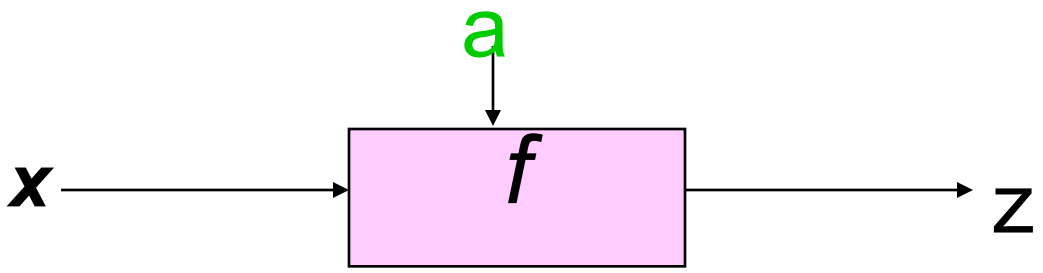
$f(\mathbf{x}, \mathbf{w}, b) = sign(\mathbf{w} \cdot \mathbf{x} + b)$

- denotes +1
- denotes -1



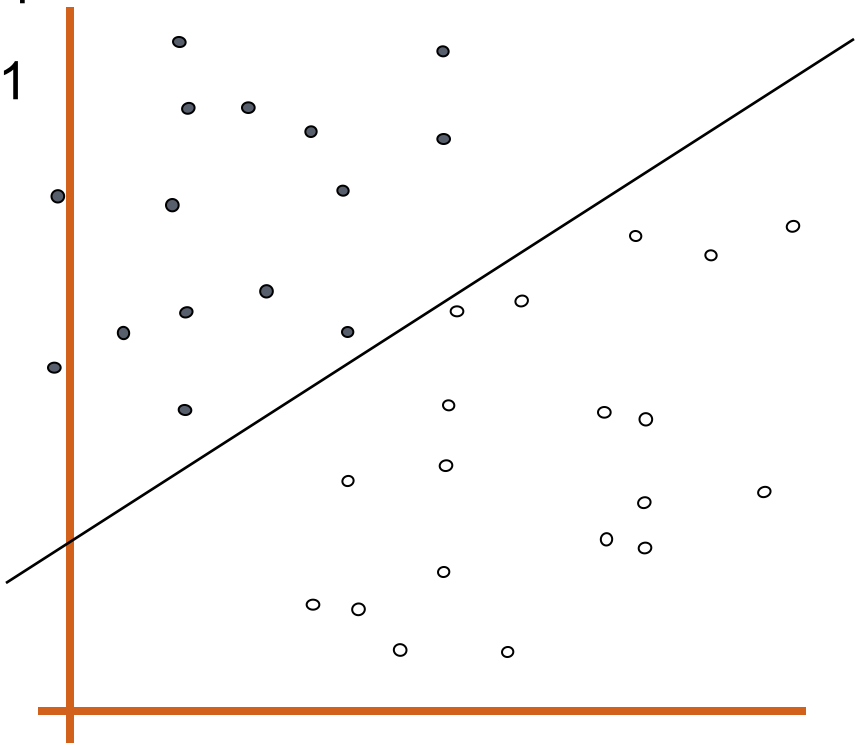
How would you classify this data?

Linear Classifiers



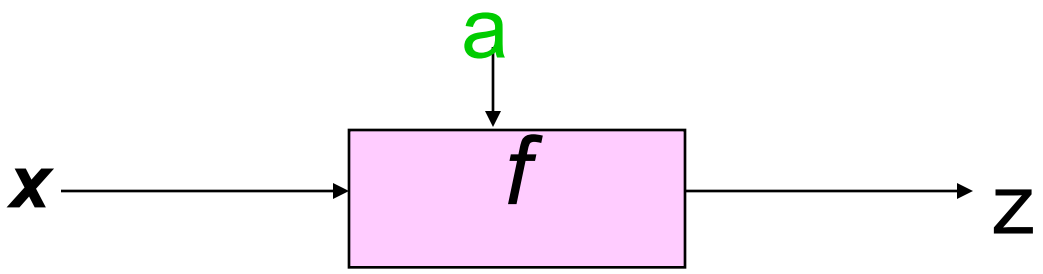
$f(\mathbf{x}, \mathbf{w}, b) = sign(\mathbf{w} \cdot \mathbf{x} + b)$

- denotes +1
- denotes -1



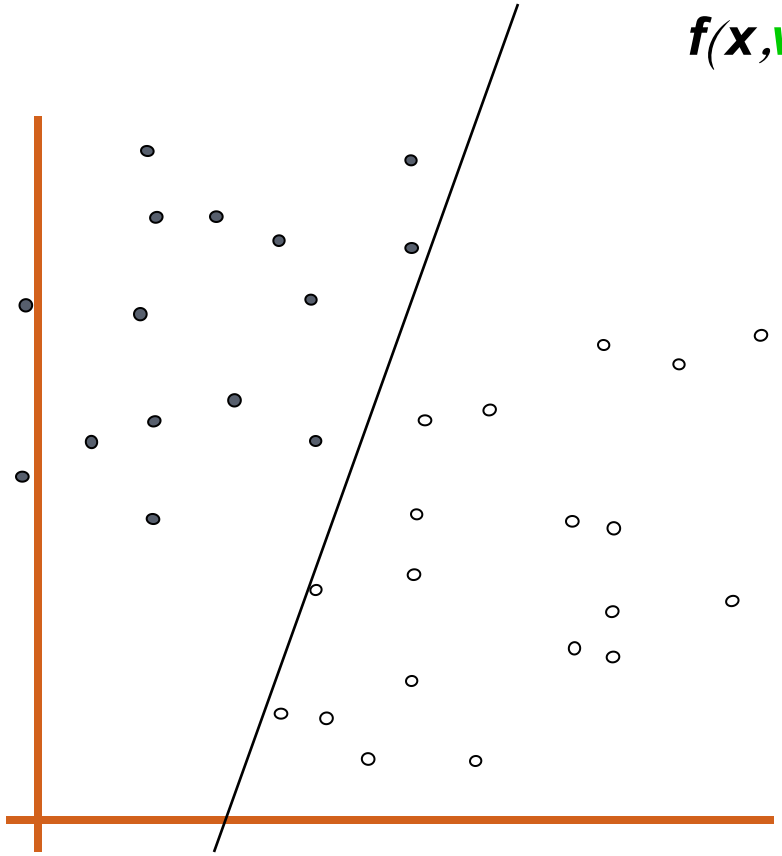
How would you classify this data?

Linear Classifiers



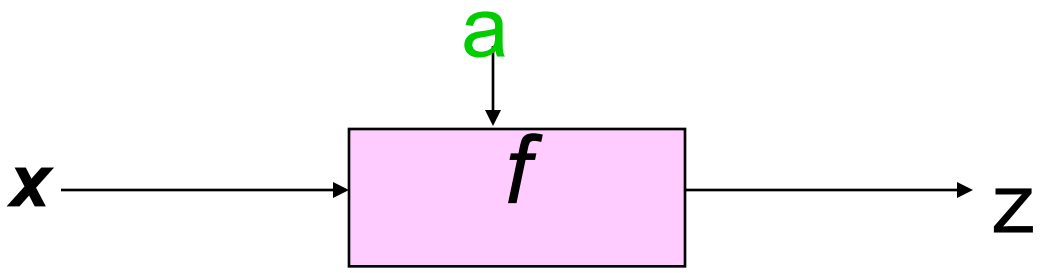
$f(\mathbf{x}, \mathbf{w}, b) = sign(\mathbf{w} \cdot \mathbf{x} + b)$

- denotes +1
- denotes -1



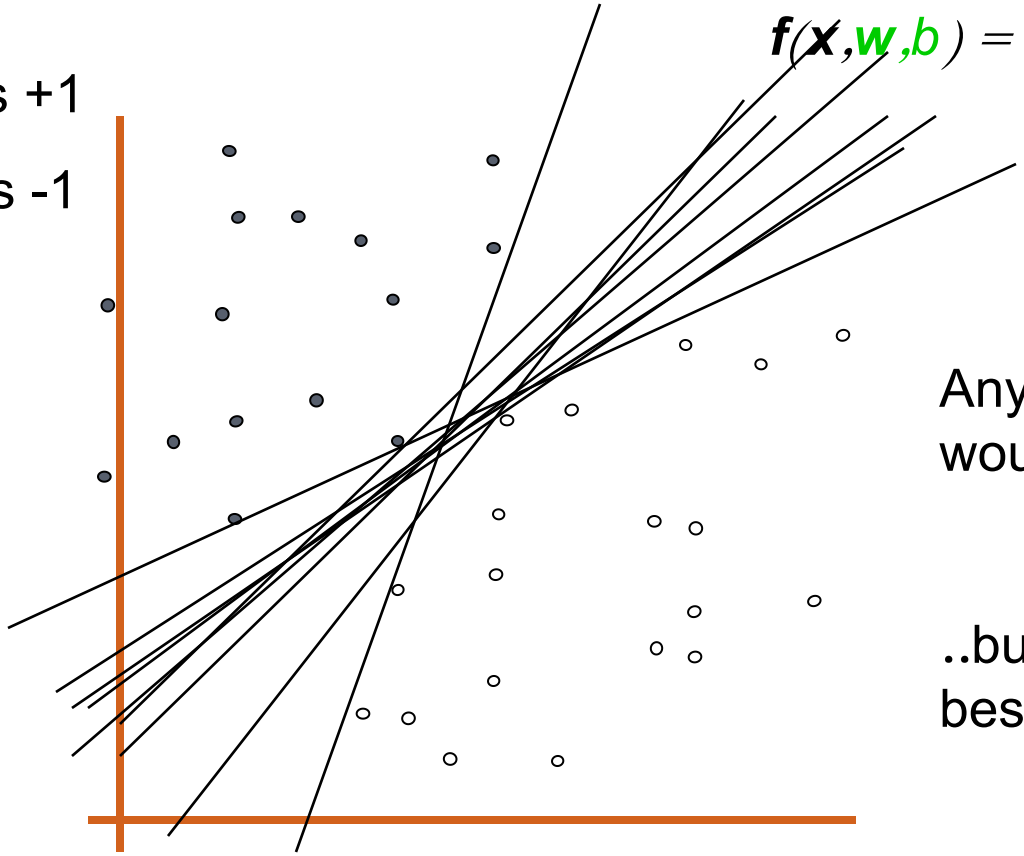
How would you classify this data?

Linear Classifiers



$f(\mathbf{x}, \mathbf{w}, b) = sign(\mathbf{w}.\ \mathbf{x} + b)$

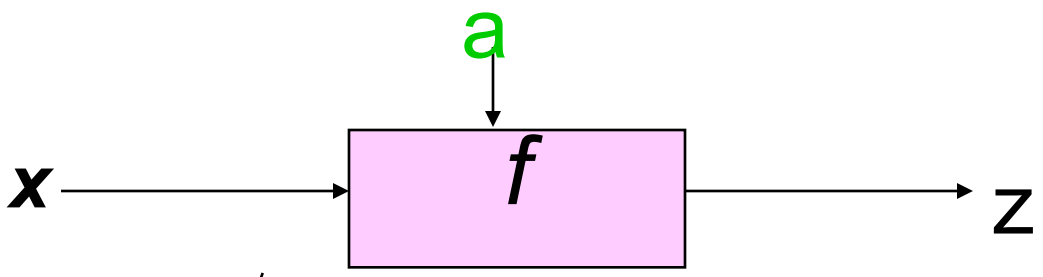
- denotes +1
- denotes -1



Any of these would be fine..

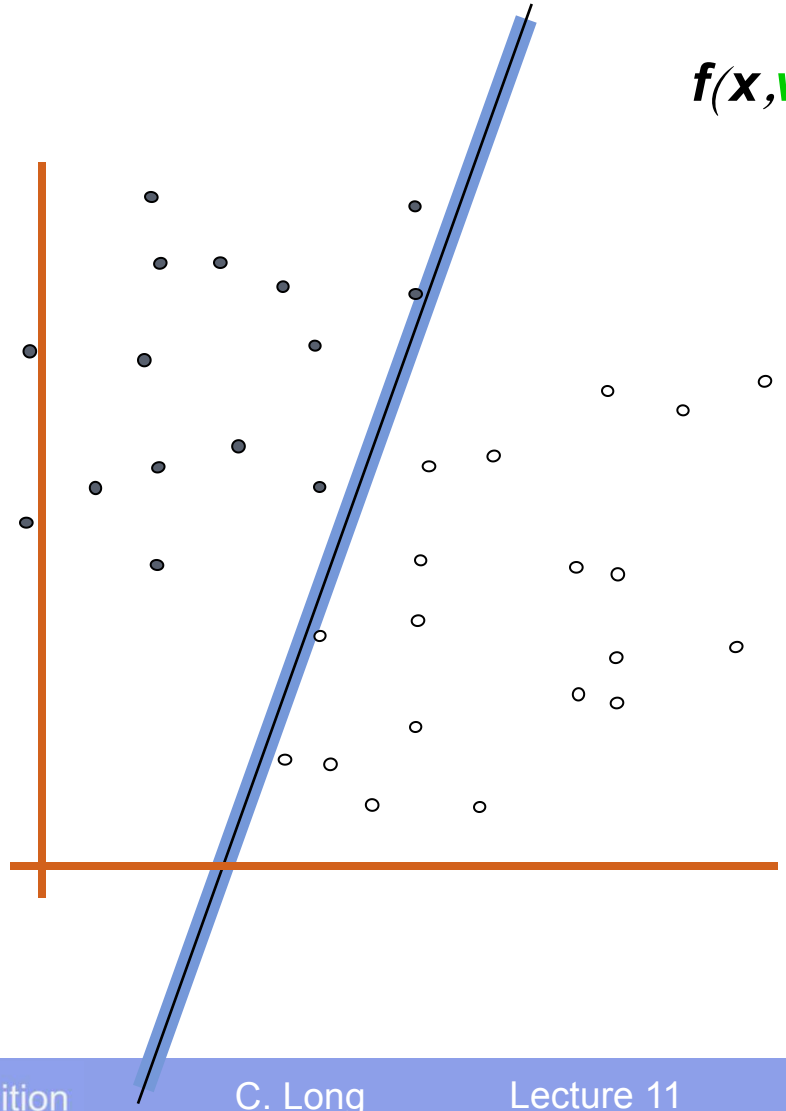
..but which is best?

Classifier Margin



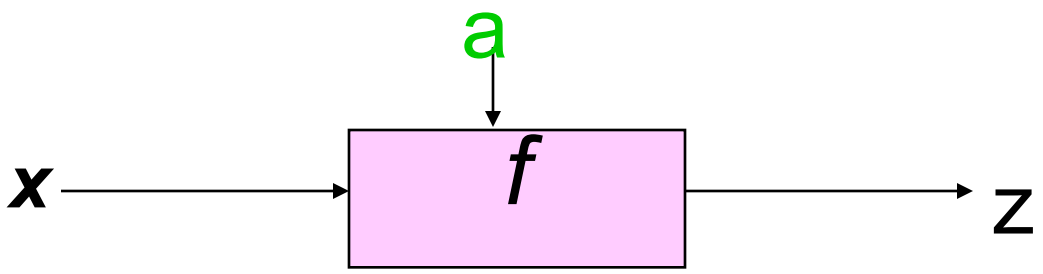
$f(\boldsymbol{x},\boldsymbol{w},b) = sign(\boldsymbol{w}.\boldsymbol{x} + b)$

- denotes +1
- denotes -1



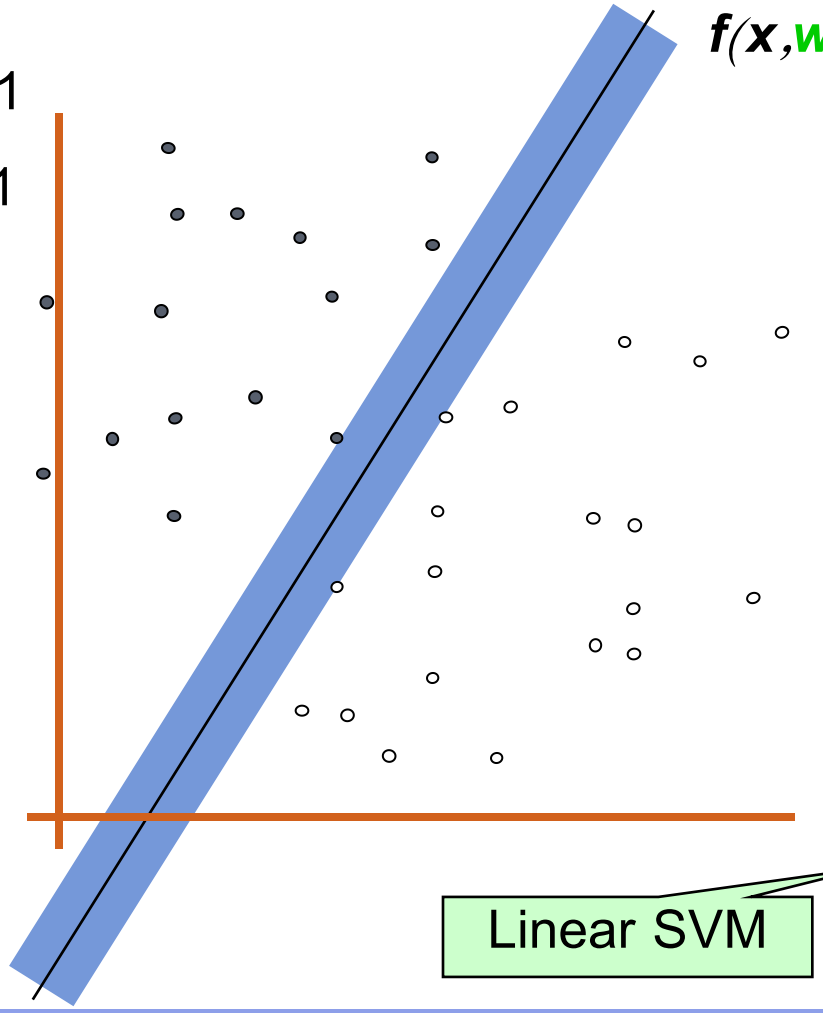
Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

Maximum Margin



$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\mathbf{x} + b)$

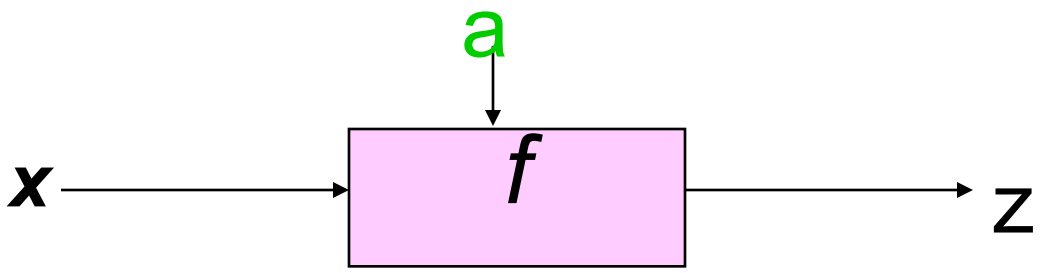
- denotes +1
- denotes -1



The maximum margin linear classifier is the linear classifier with the, um, maximum margin.

This is the simplest kind of SVM (Called an LSVM)

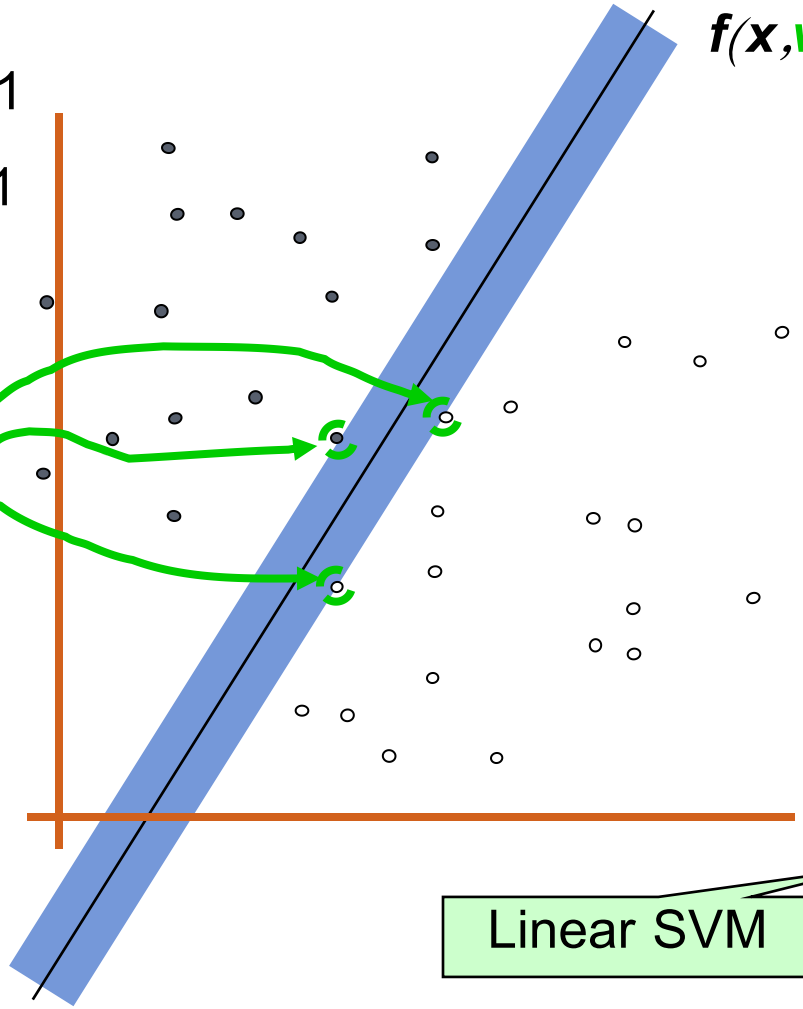
Maximum Margin



$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\mathbf{x} + b)$

- denotes +1
- denotes -1

Support Vectors are those datapoints that the margin pushes up against

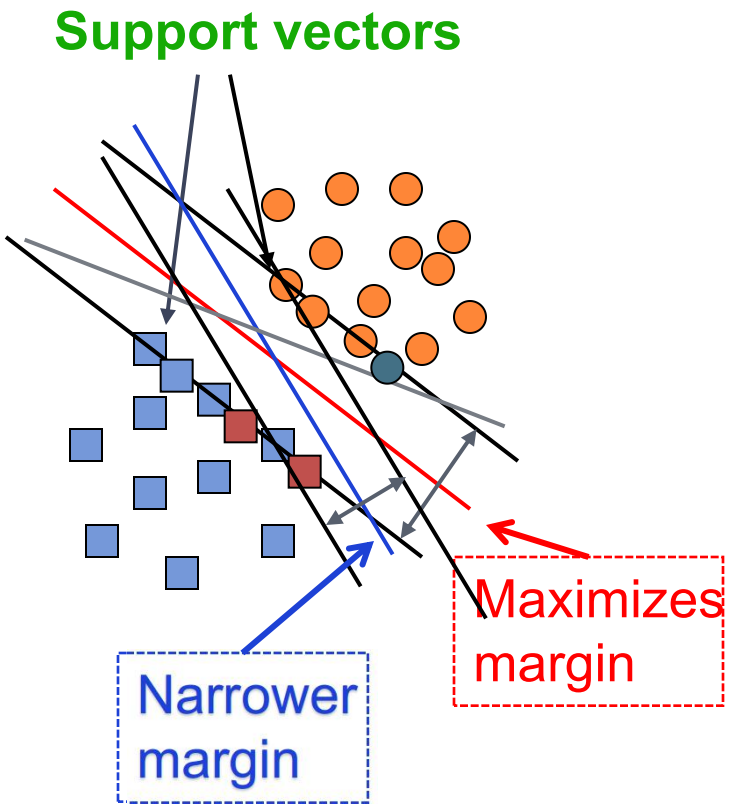


The maximum margin linear classifier is the linear classifier with the, um, maximum margin.

This is the simplest kind of SVM (Called an LSVM)

Support Vector Machine (SVM)

- SVMs maximize the *margin* around the separating hyperplane.
  - A.k.a. large margin classifiers
- The decision function is fully specified by a subset of training samples, *the support vectors*.
- Solving SVMs is a *quadratic programming* problem
- Seen by many as the most successful current text classification method*



Why Maximum Margin?

- Intuitively this feels safest.
- If we've made a small error in the location of the boundary (it's been jolted in its perpendicular direction) this gives us least chance of causing a misclassification.
- Leave-one-out cross-validation (LOOCV) is easy since the model is immune to removal of any non-support-vector datapoints.
- There's some theory (using VC dimension) that is related to (but not the same as) the proposition that this is a good thing.
- Empirically it works very very well.

Outline

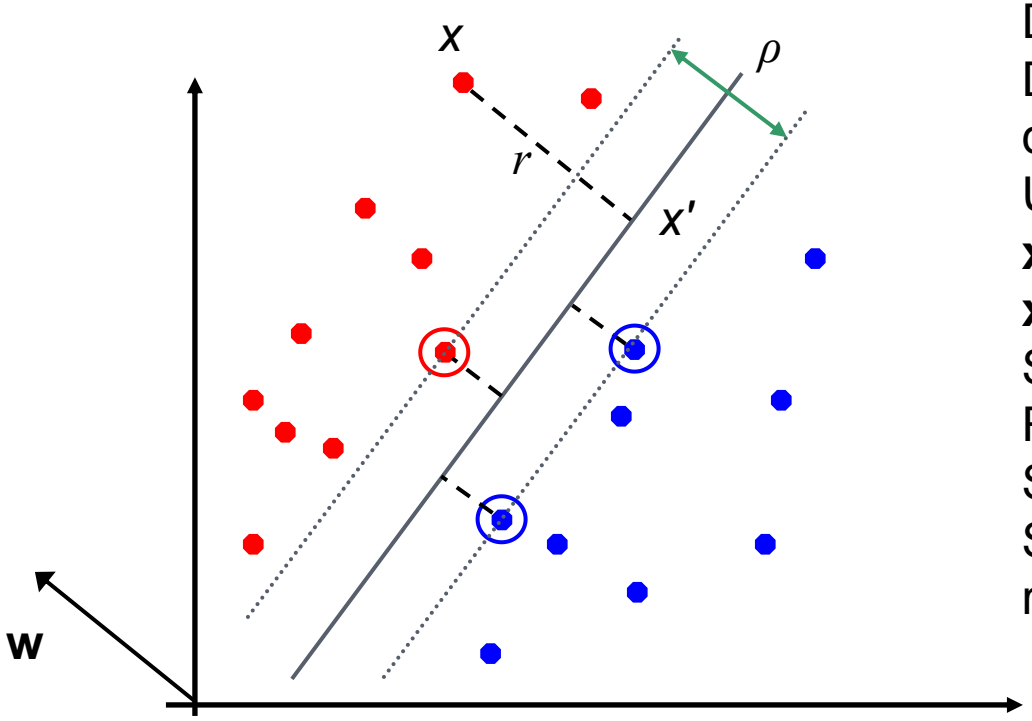
- Why Maximum Margin？
- **Support Vector Machine**
- Kernel and Kerneled SVM

Maximum Margin: Formalization

- **w**: decision hyperplane normal vector
- $\mathbf{x}_i$: data point *i*
- z_i: class of data point *i* $(+1$ or **-**$1)$
- Classifier is: $f(\mathbf{x}_i) = \text{sign}(\mathbf{w}^T\mathbf{x}_i + b)$
- Functional margin of $\mathbf{x}_i$ is: $z_i(\mathbf{w}^T\mathbf{x}_i + b)$
  - But note that we can increase this margin simply by scaling **w**, **b**….
- Functional margin of dataset is twice the minimum functional margin for any point
  - The factor of 2 comes from measuring the whole width of the margin

Geometric Margin

- Distance from example to the separator is $r = z\dfrac{\mathbf{w}^T\mathbf{x}+b}{\|\mathbf{w}\|}$
- Examples closest to the hyperplane are ***support vectors***.
- ***Margin*** ρ of the separator is the width of separation between support vectors of classes.



Derivation of finding *r:*
Dotted line **x**' −**x** is perpendicular to decision boundary so parallel to **w**.
Unit vector is **w**/|**w**|, so line is r**w**/|**w**|.
x' = **x** – zr**w**/|**w**|.
x' satisfies $\mathbf{w}^T\mathbf{x}' + b = 0$.
So $\mathbf{w}^T(\mathbf{x} - zr\mathbf{w}/|\mathbf{w}|) + b = 0$.
Recall that |**w**| = sqrt($\mathbf{w}^T\mathbf{w}$).
So $\mathbf{w}^T\mathbf{x} - zr|\mathbf{w}| + b = 0$
So, solving for r gives:
$r = z(\mathbf{w}^T\mathbf{x} + b)/|\mathbf{w}|$

Linear SVM Mathematically

- Assume that all data is at least distance 1 from the hyperplane, then the following two constraints follow for a training set $\{(\mathbf{x_i}, z_i)\}$

$$\mathbf{w^T x_i} + b \geq 1 \quad \text{if } z_i = 1$$

$$\mathbf{w^T x_i} + b \leq -1 \quad \text{if } z_i = -1$$

- For support vectors, the inequality becomes an equality
- Then, since each example's distance from the hyperplane is

$$r = z\frac{\mathbf{w}^T\mathbf{x} + b}{\|\mathbf{w}\|}$$

- The margin is:

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

Linear Support Vector Machine (SVM)

- **Hyperplane**

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

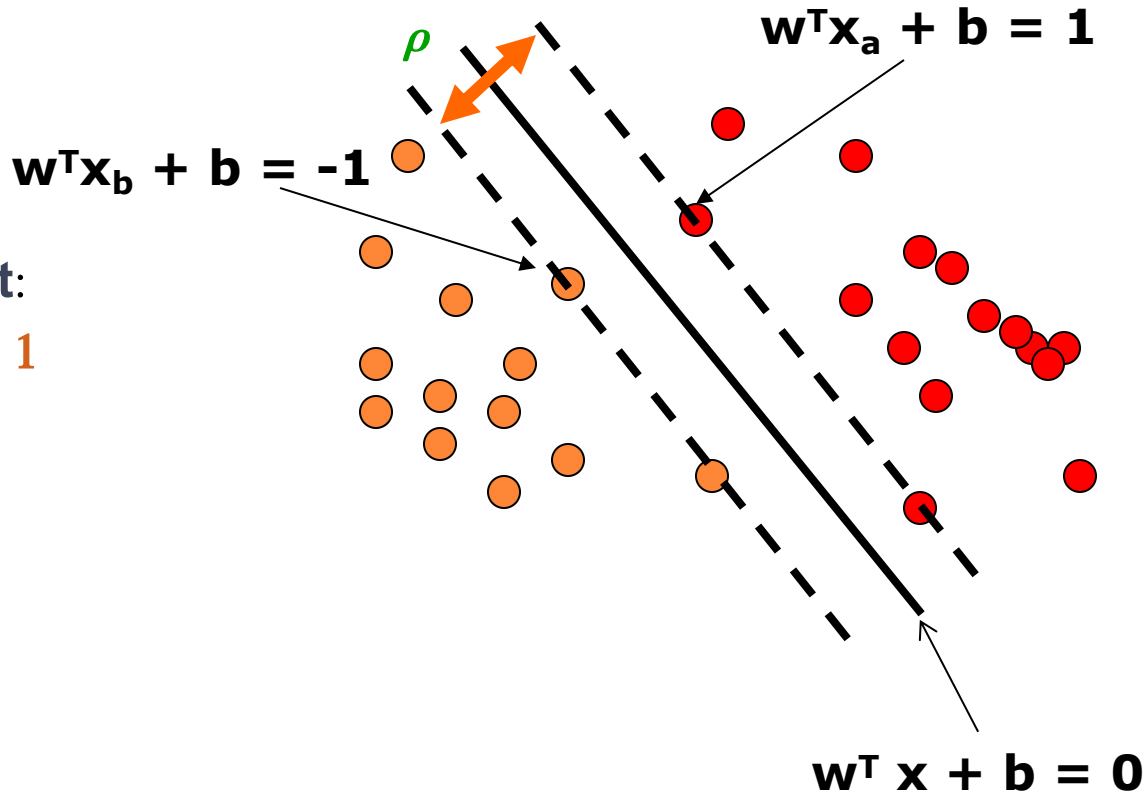
- **Extra scale constraint**:

  $\min_{i=1,\ldots,n} |\mathbf{w}^T \mathbf{x}_i + b| = 1$

- This implies:

  $\mathbf{w}^T(\mathbf{x}_a - \mathbf{x}_b) = 2$

  $\rho = ||\mathbf{x}_a - \mathbf{x}_b|| = 2/||\mathbf{w}||$



SVM: Optimal Hyperplane

- Maximize margin $\rho = ||x_a - x_b|| = 2/||w||$
- Subject to constraints $\begin{cases} w^t x_i + w_0 \geq 1 & \text{if } x_i \text{ is positive example} \\ w^t x_i + w_0 \leq -1 & \text{if } x_i \text{ is negative example} \end{cases}$
- Let $\begin{cases} z_i = 1 & \text{if } x_i \text{ is positive example} \\ z_i = -1 & \text{if } x_i \text{ is negative example} \end{cases}$
- Can convert our problem to minimize

$$\text{minimize } J(w) = \frac{1}{2}\|w\|^2$$
$$\text{constrained to } z_i(w^t x_i + w_0) \geq 1 \quad \forall i$$

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

SVM: solution, Lagrange multipliers

$$\begin{aligned} &\text{minimize } J(w) = \frac{1}{2}\|w\|^2 \\ &\text{constrained to } z_i(w^t x_i + w_o) \geq 1 \quad \forall i \end{aligned}$$

- Let start by representing the constraints as losses

$$\max_{\alpha>0} \alpha(1 - z_i(\mathbf{w}^t\mathbf{x}_i + w_0)) = \begin{cases} 0, & z_i(\mathbf{w}^t\mathbf{x}_i + w_0) - 1 \geq 0 \\ \infty, & \text{otherwise} \end{cases}$$

and rewrite the minimization problem in terms of these

$$\min_{\mathbf{w}} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{n} \max_{\alpha_i>0} \alpha_i(1 - z_i(\mathbf{w}^t\mathbf{x}_i + w_0)) \right\}$$
$$= \min_{\mathbf{w}} \max_{\alpha_i>0} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{n} \alpha_i(1 - z_i(\mathbf{w}^t\mathbf{x}_i + w_0)) \right\}$$

SVM solution cont'd

- We can then swap "max" and "min":

$$\min_{\mathbf{w}} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{n} \max_{\alpha_i>0} \alpha_i(1 - z_i(\mathbf{w}^t\mathbf{x}_i + w_0)) \right\}$$

$$= \min_{\mathbf{w}} \max_{\alpha_i>0} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{n} \alpha_i(1 - z_i(\mathbf{w}^t\mathbf{x}_i + w_0)) \right\}$$

$$= \max_{\alpha_i>0} \min_{\mathbf{w}} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{n} \alpha_i(1 - z_i(\mathbf{w}^t\mathbf{x}_i + w_0)) \right\}$$

- As a result we have to be able to minimize $J(\mathbf{w}; \boldsymbol{\alpha})$ with respect to parameters w for any fixed setting of the Lagrange multipliers $\alpha_i \geq 0$

SVM solution cont'd

- We can then swap 'max' and 'min':

$$\min_{\mathbf{w}} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{n} \max_{\alpha_i>0} \alpha_i(1 - z_i(\mathbf{w}^t\mathbf{x}_i + w_0)) \right\}$$

$$= \min_{\mathbf{w}} \max_{\alpha_i>0} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{n} \alpha_i(1 - z_i(\mathbf{w}^t\mathbf{x}_i + w_0)) \right\}$$

$$= \max_{\alpha_i>0} \min_{\mathbf{w}} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{n} \alpha_i(1 - z_i(\mathbf{w}^t\mathbf{x}_i + w_0)) \right\}$$

- We can find the optimal w as a function of $\{\alpha_i\}$ by setting the derivatives to zero:

$$\frac{\partial}{\partial \mathbf{w}} J(\mathbf{w}, \alpha) = \mathbf{w} - \sum_{i=1}^{n} \alpha_i z_i \mathbf{x}_i = \mathbf{0}$$

$$\frac{\partial}{\partial w_0} J(\mathbf{w}, \alpha) = -\sum_{i=1}^{n} \alpha_i z_i = 0$$

SVM solution cont'd

- Back into the objective and get (after som algebra):

$$\max_{\substack{\alpha_i \geq 0 \\ \sum_i \alpha_i z_i = 0}} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{n} \alpha_i (1 - z_i(\mathbf{w}^t\mathbf{x} + w_0)) \right\}$$

$$= \max_{\substack{\alpha_i \geq 0 \\ \sum_i \alpha_i z_i = 0}} \left\{ \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j z_i z_j \mathbf{x}_i^t \mathbf{x}_j \right\}$$

SVM: Optimal Hyperplane

- Use Kuhn-Tucker Theorem (KTT) condition to convert our problem to:

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

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

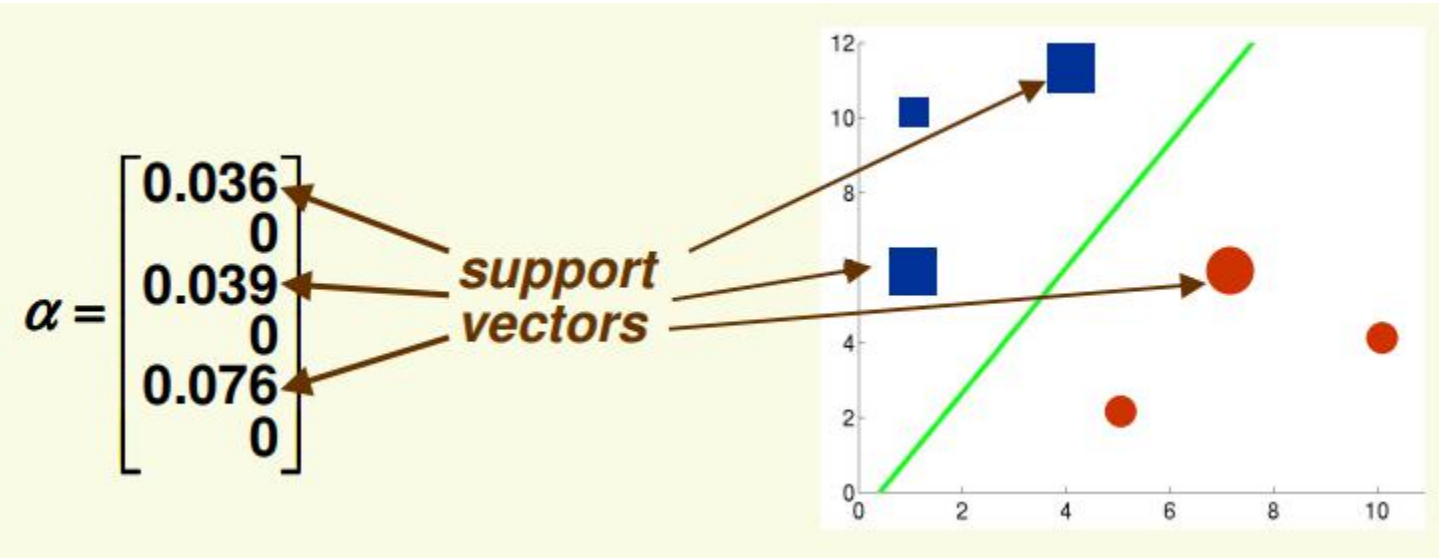
- $a = \{a_1, \ldots, a_n\}$ are new variables, one for each sample
- Optimized by quadratic programming

SVM: Optimal Hyperplane

- After finding the optimal $a = \{a_1, \ldots, a_n\}$
- Final discriminant function:

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

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



SVM: Optimal Hyperplane

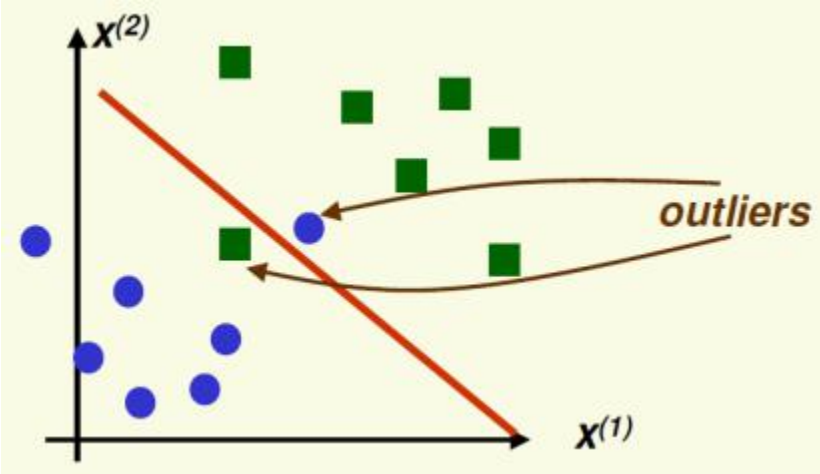
$$\text{maximize} \quad L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j z_i z_j \mathbf{x}_i^t \mathbf{x}_j$$

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

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

SVM: Non-Separable Case

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



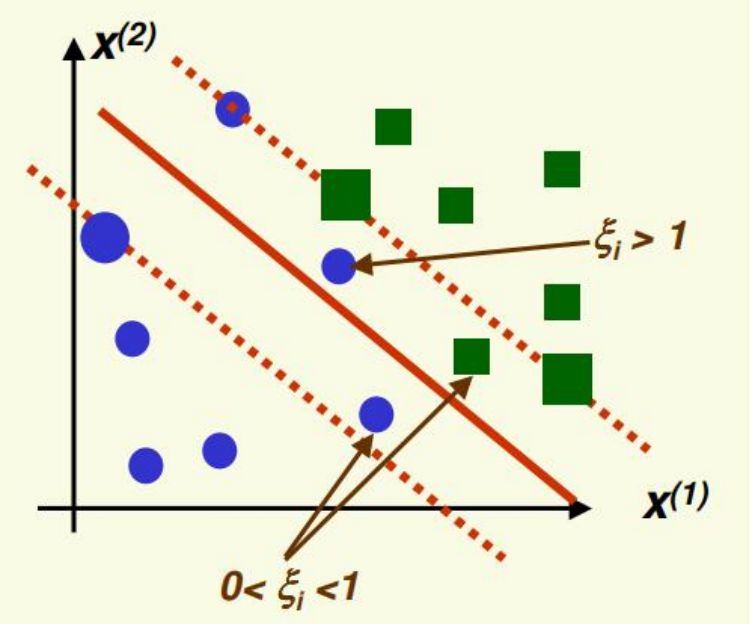
- Can apply SVM in non linearly separable case
- Data should be "almost" linearly separable for good performance

SVM: Non-Separable Case

- Use slack variables $\xi_1, \ldots, \xi_n$ (one for each sample)
- Change constraints:

$$z_i(w^t x_i + w_0) \geq 1 \quad \forall i \quad \Rightarrow \quad z_i(w^t x_i + w_0) \geq 1 - \xi_i \quad \forall i$$

- ξ_i is a measure of deviation from the ideal for x_i
  - $\xi_i \geq 1$: x_i is on the wrong side of the separating hyperplane
  - $0 < \xi_i < 1$: x_i is on the right side of separating hyperplane but within the region of maximum margin
  - $\xi_i < 0$: is the ideal case for x_i



SVM: Non-Separable Case

- We would like to minimize

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

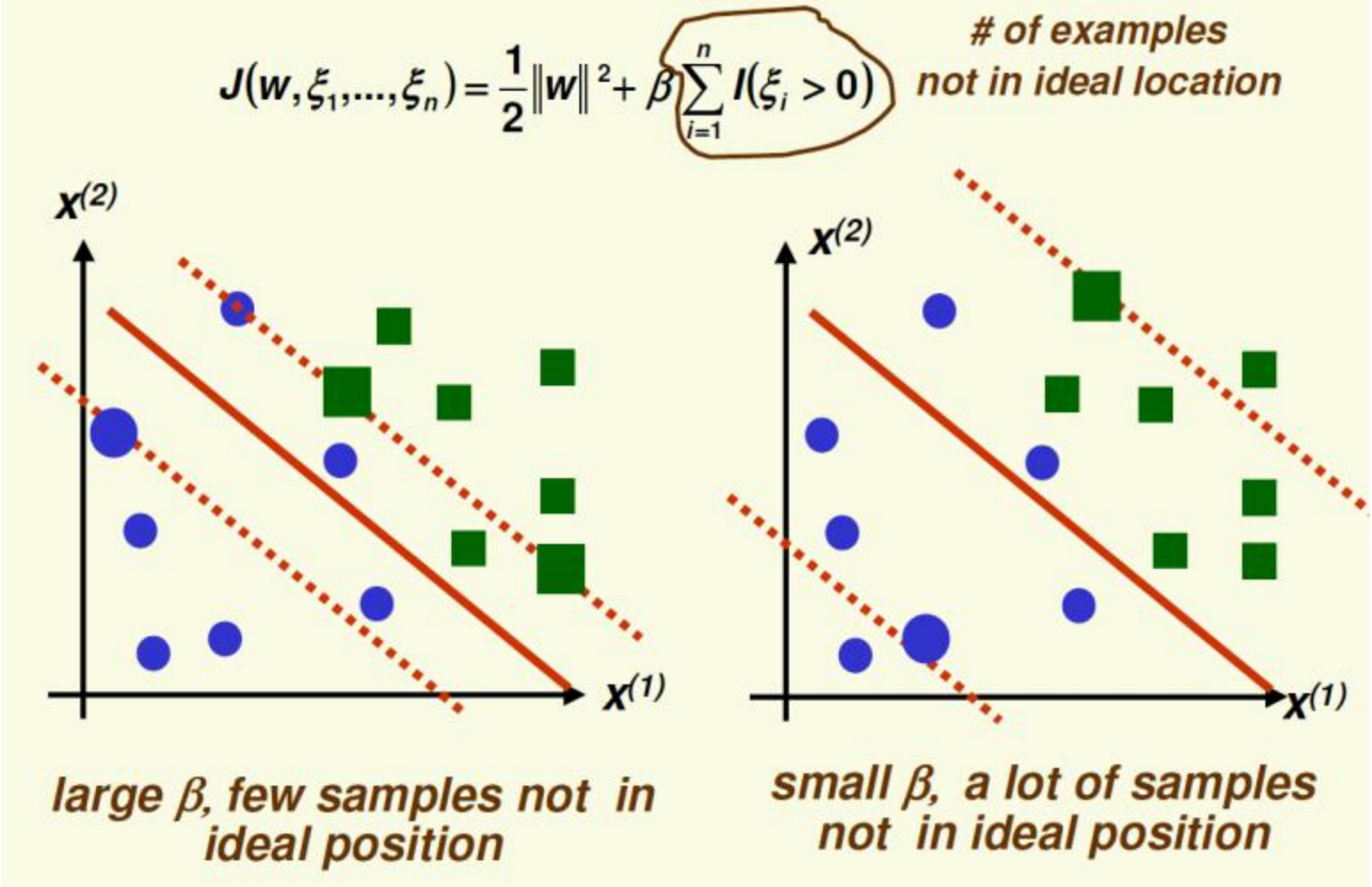
# of samples not in ideal location

Where $I(\xi_i>0)=\begin{cases}1 & \text{if } \xi_i>0\\ 0 & \text{if } \xi_i\le 0\end{cases}$

- Constrained to $z_i(w^t x_i+w_0)\ge 1-\xi_i$ and $\xi_i\ge 0 \quad \forall i$

- β is a constant which measures relative weight of the first and second terms
  - if β is small, we allow a lot of samples not in ideal position
  - if β is large, we want to have very few samples not in ideal positon

SVM: Non-Separable Case



SVM: Non-Separable Case

- Unfortunately this minimization problem is NP–hard due to the discontinuity of I(ξ_i)

- Instead, we minimize

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

a measure of # of misclassified examples

subject to

$$\begin{cases} z_i\left(w^t x_i + w_0\right)\geq 1-\xi_i & \forall i \\ \xi_i \geq 0 & \forall i \end{cases}$$

SVM: Non-Separable Case

- Use Kuhn–Tucker Theorem (KTT) conditon to convert to:

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

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

- w and w_0 is computed using:

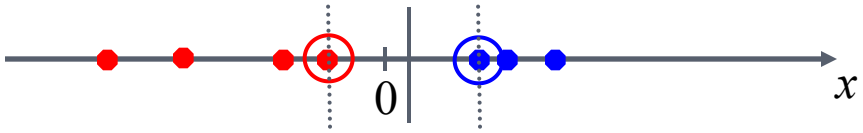
$$w = \sum_{i=1}^{n} \alpha_i z_i x_i \quad \Rightarrow \quad \alpha_i \left[z_i \left(w^t x_i + w_0 \right) - 1 \right] = 0$$

- Remember that:

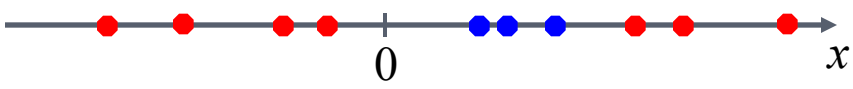
$$g(x) = \left(\sum_{x_i \in S} \alpha_i z_i x_i \right)^t x + w_0$$

Non-linear SVMs

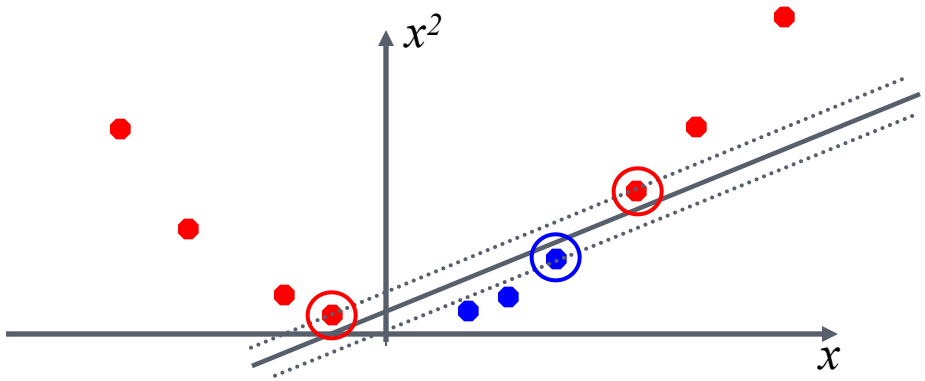
- Datasets that are linearly separable (with some noise) work out great:



- But what are we going to do if the dataset is just too hard?

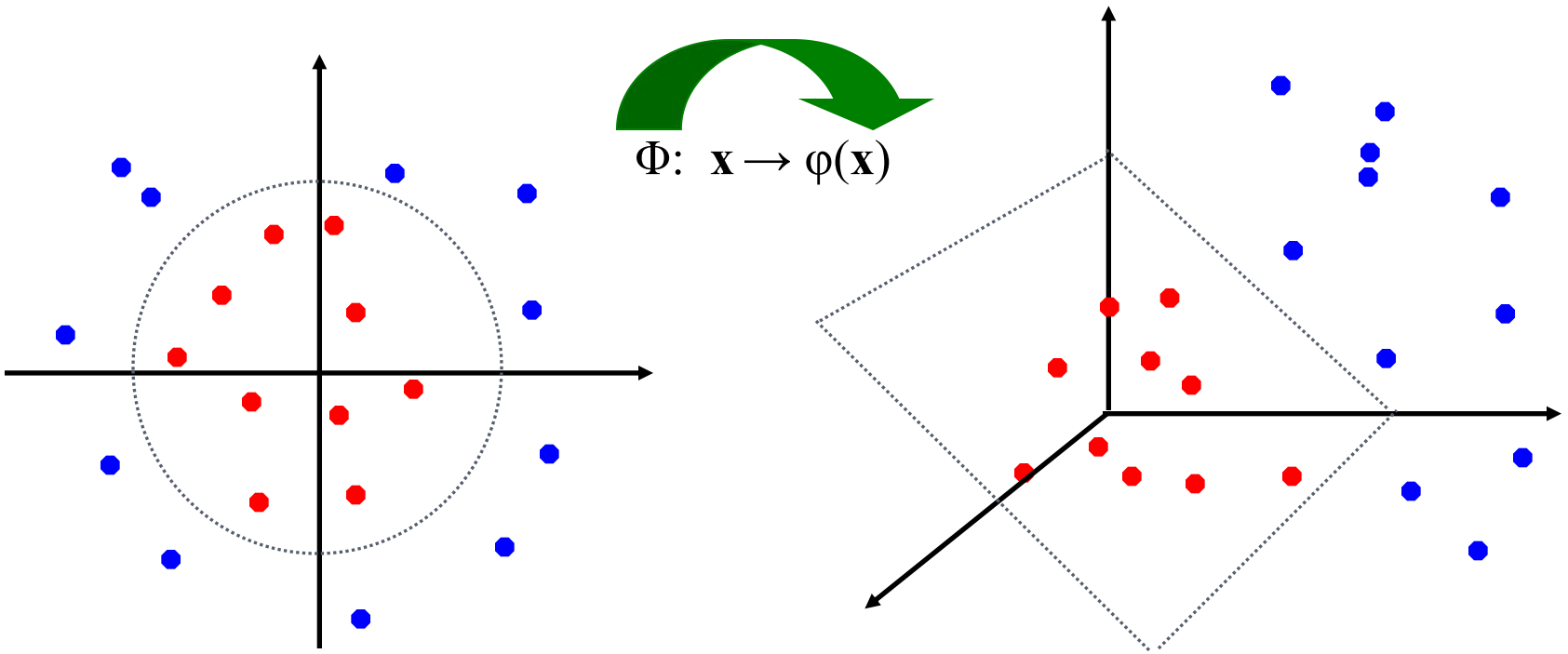


- How about mapping data to a higher dimensional space:



Non-linear SVMs: Feature spaces

- General idea: the original feature space can always be mapped to some higher dimensional feature space where the training set is separable:



Outline

- Why Maximum Margin？
- Support Vector Machine
- **Kernel and Kerneled SVM**

Kernels

- SVM optimization:

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

- Note this optimization depends on samples x_i only through the dot product $x_i^t x_j$

- If we lift x_i to high dimension using $\phi(\mathbf{x_i})$, we need to compute high dimensional product $\phi(\mathbf{x_i})^T \phi(\mathbf{x_j})$

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

$$K(x_i, x_j)$$

- Idea: find kernel function K(xi,xj) s.t. $K(x_i, x_j) = \varphi(x_i)^t \varphi(x_j)$

Kernel Trick

- Then we only need to compute $K(x_i, x_j)$ instead of $\phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$

- "**kernel trick**": do not need to perform operations in high dimensional space explicitly

Example

- 2-dimensional vectors $\mathbf{x}=[x_1, x_2]$.
- Let $K(\mathbf{x_i},\mathbf{x_j})=(1 + \mathbf{x_i}^T\mathbf{x_j})^2$,
- Need to show that $K(\mathbf{x_i},\mathbf{x_j})= \phi(\mathbf{x_i})^T \phi(\mathbf{x_j})$:

$$K(\mathbf{x_i},\mathbf{x_j})=(1 + \mathbf{x_i}^T\mathbf{x_j})^2$$

$$= 1+ x_{i1}^2x_{j1}^2 + 2\, x_{i1}x_{j1}\, x_{i2}x_{j2}+ x_{i2}^2x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2}$$

$$= [1,\ x_{i1}^2, \sqrt{2}x_{i1}x_{i2},\ x_{i2}^2, \sqrt{2}x_{i1}, \sqrt{2}x_{i2}]^T [1,\ x_{j1}^2, \sqrt{2}x_{j1}x_{j2},\ x_{j2}^2,\ \sqrt{2}x_{j1},\ \sqrt{2}x_{j2}]$$

Choice of Kernel

- How to choose kernel function $K(x_i, x_j)$?
  - $K(x_i, x_j)$ should correspond to $\phi(\mathbf{x_i})^T \phi(\mathbf{x_j})$ in a higher dimensional space
  - Mercer's condition tells us which kernel function can be expressed as dot product of two vectors
  - If K and K' are kernels aK+bK' is a kernel

- Intuitively: Kernel should measure the similarity between x_i and x_j
  - As inner product measures similarity of unit vectors
  - May be problem specific

Choice of Kernel

- Some common choices:
  - Polynomial kernel

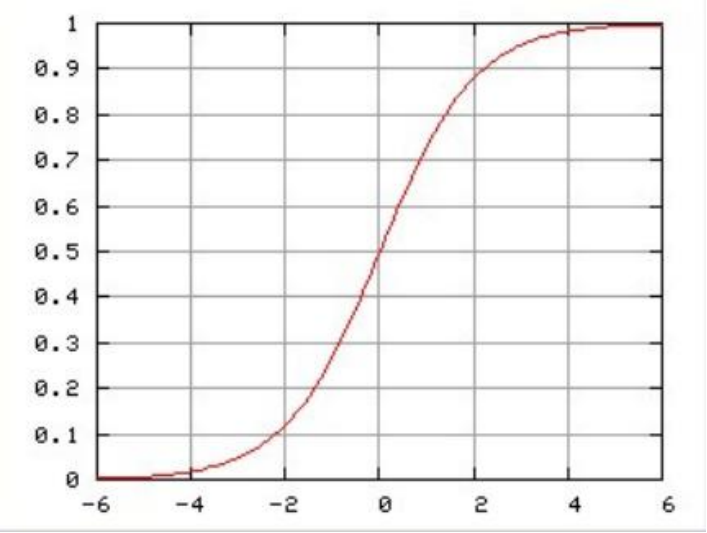
$$K(x_i, x_j) = (x_i^t x_j + 1)^p$$

  - Gaussian radial Basis kernel

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

  - Hyperbolic tangent (sigmoid) kernel

$$K(x_i, x_j) = \tanh(k\, x_i^t x_j + c)$$



- The mappings $\phi(x_i)$ never have to be computed!!

Intersection Kernel

- Feature vectors are histograms

$$K(x_i, x_j) = \sum_{k=1}^{n} \min(x_{ik}, x_{jk})$$

- When $K(x_i, x_j)$ is small, x_i and x_j are dissimilar
- When $K(x_i, x_j)$ is large, x_i and x_j are similar
- The mapping $\phi(x)$ does not exist

More Additive Kernels

- χ^2 kernel $K_{\chi^2} = \sum_{k=1}^{n} \frac{2x_k y_k}{x_k + y_k}$
- Hellinger's kernel $K_H = \sum_{k=1}^{n} \sqrt{x_k y_k}$
- Designed for feature vectors that are histograms
  - Can be used for other feature vectors
- Offer very large speed ups

The Kernel Matrix

- a.k.a 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)

- Contains all necessary information for the learning algorithm
- Fuses information about the data and the kernel (similarity measure)

Bad Kernels

- The kernel matrix is mostly diagonal
  – All points are orthogonal to each other
- Bad similarity measure
- Too many irrelevant features in high dimensional space
- We need problem specific knowledge to choose appropriate kernel

Nonlinear SVM Step-by-Step

- Start with data $x_1,\ldots,x_n$ which live in feature space of dimension d
- Choose kernel $K(x_i,x_j)$ or function $\phi(x_i)$ which lifts sample x_i to a higher dimensional space
- Find the maximum margin linear discriminant function in the higher dimensional space by using quadratic programming package to solve:

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

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

Nonlinear SVM Step-by-Step

- Weight vector w in the high dimensional space:

$$\boldsymbol{w} = \sum_{x_i \in S} \alpha_i z_i \varphi(\boldsymbol{x}_i)$$

  – where S is the set of support vectors

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

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

- Non linear discriminant function in the original space:

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

- decide class 1 if $g(x) \geq 0$, otherwise decide class 2

Nonlinear SVM

- Nonlinear discriminant function

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

$$g(\boldsymbol{x}) = \sum$$

weight of support vector $\boldsymbol{x}_i$

∓ 1

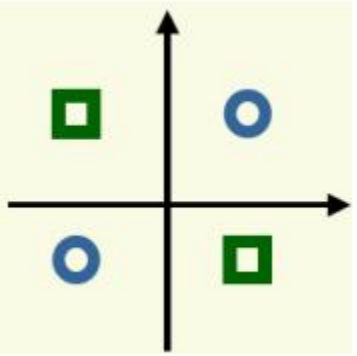
"inverse distance" from $\boldsymbol{x}$ to support vector $\boldsymbol{x}_i$

most important training samples, i.e. support vectors

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

SVM Example: XOR Problem

- Class 1: $x1 = [1,1]$, $x2 = [1,1]$
- Class 2: $x3 = [1,1]$, $x4 = [1,1]$
- Use polynomial kernel of degree 2:



$$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^t \mathbf{x}_j + 1)^2$$

  - This kernel corresponds to the mapping

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

- Need to maximize

$$L_D(\alpha) = \sum_{i=1}^{4} \alpha_i - \frac{1}{2}\sum_{i=1}^{4}\sum_{j=1}^{4} \alpha_i \alpha_i z_i z_j (\mathbf{x}_i^t \mathbf{x}_j + 1)^2$$

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

SVM Example: XOR Problem

- After some manipulation …
- The solution is $a1 = a2 = a3 = a4 = 0.25$
  - satisfies the constraints

$$\forall i,\ 0 \le \alpha_i \text{ and } \alpha_1 + \alpha_2 - \alpha_3 - \alpha_4 = 0$$

- **All samples are support vectors**

SVM Example: XOR Problem

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

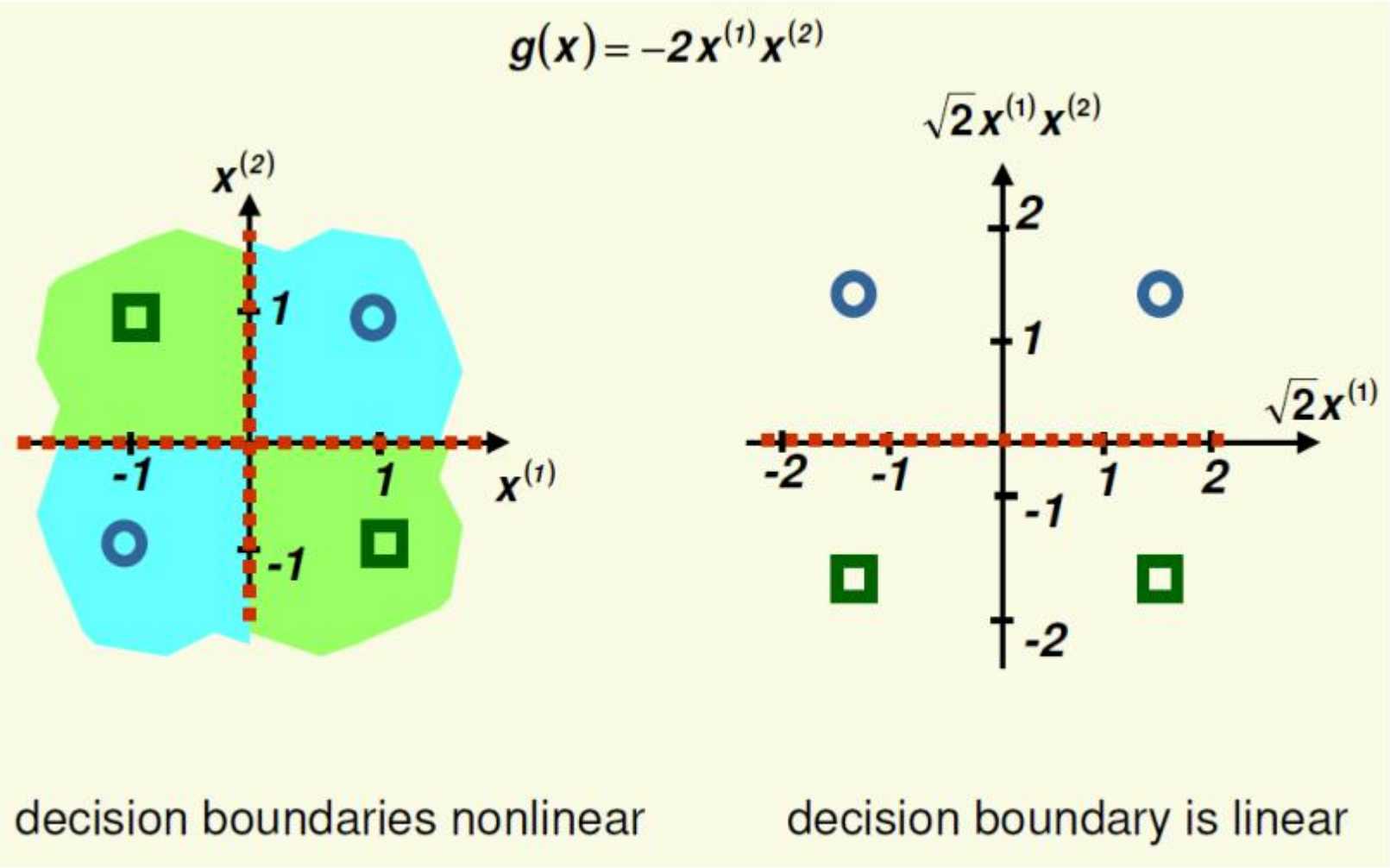
- The weight vector w is:

$$w = \sum_{i=1}^{4} \alpha_i z_i \varphi(x_i) = 0.25(\varphi(x_1) + \varphi(x_2) - \varphi(x_3) - \varphi(x_4))$$
$$= \begin{bmatrix} 0 & 0 & 0 & -\sqrt{2} & 0 & 0 \end{bmatrix}$$

- Thus the nonlinear discriminant function is:

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

SVM Example: XOR Problem



SVMs for image classification

- Pick an image representation (in our case, bag of features)
- Pick a kernel function for that representation
- Compute the matrix of kernel values between every pair of training examples
- Feed the kernel matrix into your favorite SVM solver to obtain support vectors and weights
- At test time: compute kernel values for your test example and each support vector, and combine them with the learned weights to get the value of the decision function

What about multi-class SVMs?

- Unfortunately, there is no "definitive" multi−class SVM formulation
- In practice, we have to obtain a multi−class SVM by combining multiple two/ class SVMs
- One vs. others
  - Traning: learn an SVM for each class vs. the others
  - Testing: apply each SVM to test example and assign to it the class of the SVM that returns the highest decision value
- One vs. one
  - Training: learn an SVM for each pair of classes
  - Testing: each learned SVM "votes" for a class to assign to the test example

Application Example: Digital Recognition

- MINST Dataset
  - All images were centered in a 28×28 image. This database contains $60{,}000$ images for training and $10{,}000$ images for testing.

$$y_i(w \cdot x_i + b) \geq 1 - \varepsilon_i, \quad \varepsilon_i \geq 0, \quad 1 \leq i \leq n$$

$$\min \ \frac{1}{2}||w||^2 + C\sum_{i=1}^{n} \varepsilon_i$$

$$K(x_i, x_j) = \exp(-\gamma||x_i - x_j||^2)$$

x-axis

y-axis

The strategy for multi-class classification with binary classifiers is one-against-one, 10-fold cross validation.

y = x

y = -x

[Tuba,E., Tuba,M., Simian,D. Handwritten Digit Recognition by Support Vector Machine Optimized by Bat Algorithm, In Proc. WSCG 2016.]

Application Example: Digital Recognition

	0	1	2	3	4	5	6	7	8	9
0	99	0	0	0	0	0	0	0	1	0
1	0	99	0	0	0	0	0	0	1	0
2	0	0	97	1	1	0	0	0	1	0
3	0	0	3	89	0	1	0	1	6	0
4	0	0	0	0	98	1	0	0	0	1
5	0	0	1	7	0	91	0	0	1	0
6	0	0	0	0	0	0	100	0	0	0
7	0	0	0	0	0	1	0	93	0	6
8	0	0	1	0	0	1	2	1	95	0
9	1	0	0	0	2	0	0	1	1	95

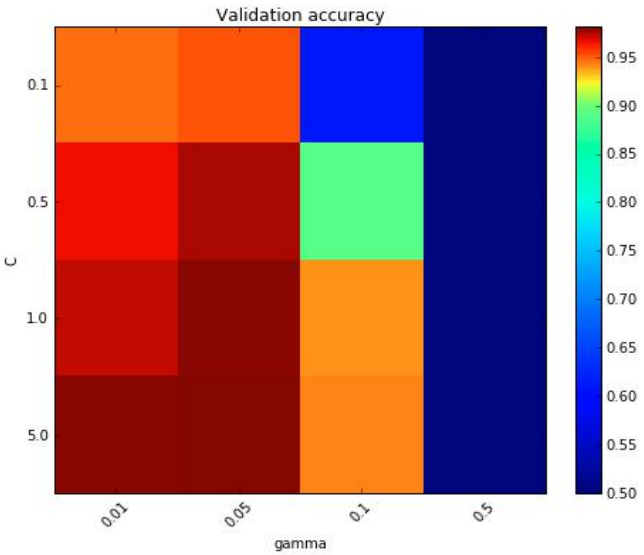
Table 1: Accuracy of classification for our proposed method (%)

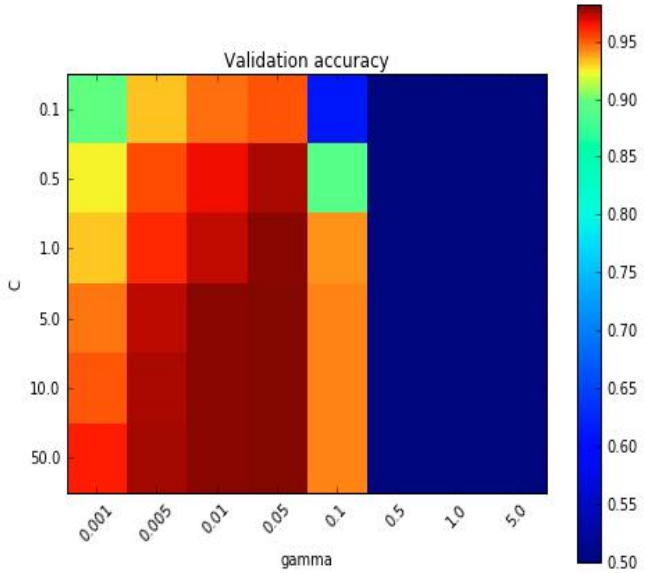
Digit	MLNN	SVM-BAT
0	86.45	99.00
1	94.39	99.00
2	88.73	97.00
3	77.02	89.00
4	76.12	98.00
5	84.10	91.00
6	78.81	100.00
7	77.12	93.00
8	79.03	95.00
9	49.64	95.00
Global	79.14	**95.60**

MLNN: Kessab, B. E., Daoui, C., Bouikhalene, B. Fakir, M. and Moro, K. Extraction method of handwritten digit recognition tested on the MNIST database. IJAST, 2013.

SVM-BAT: Tuba,E., Tuba,M., Simian,D. Handwritten Digit Recognition by Support Vector Machine Optimized by Bat Algorithm, In Proc. WSCG 2016.

Application Example: Digital Recognition





Method	Accuracy	Comments
Random forest	0.937	
Simple one-layer neural network	0.926	
Simple 2 layer convolutional network	0.981	
SVM RBF	0.9852	C=5, gamma=0.05
Linear SVM + Nystroem kernel approximation		
Linear SVM + Fourier kernel approximation		

Best parameters:

- C = 5
- gamma = 0.05
- accuracy = 0.9852

- Code available at Github.
- URL: https://github.com/ksopyla/svm_mnist_digit_classification

SVMs: Pros and cons

- Pros
  - Many publicly available SVM packages: http://www.kernel-machines.org/software
  - Kernel-based framework is very powerful, flexible
  - SVMs work very well in practice, even with very small training sample sizes

- Cons
  - No "direct" multi-class SVM, must combine two-class SVMs
  - Computation, memory
    - During training time, must compute matrix of kernel values for every pair of examples
    - Learning can take a very long time for large-scale problems

LIBSVM

- http://www.csie.ntu.edu.tw/~cjlin/libsvm/

- Available Examples with datasets :
  - http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/
  - http://archive.ics.uci.edu/ml/datasets.html

Q & A