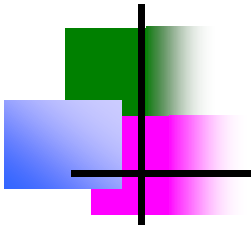


Machine Learning and Computational Intelligence
Lecture 12

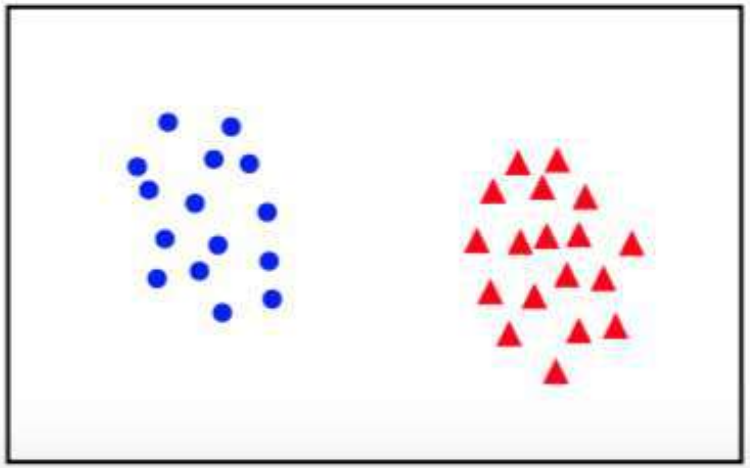
Sanjeeb Prasad Panday, PhD
Associate Professor
Dept. of Electronics and Computer Engineering
Director (ICTC)
IOE, TU

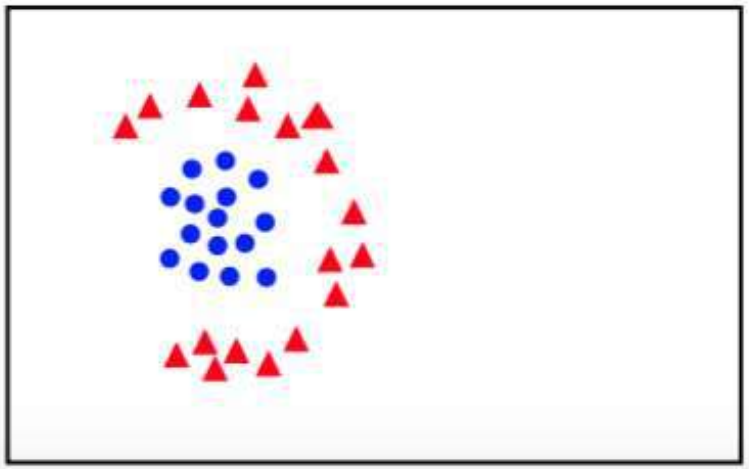
Binary Classification

Given training data $(\mathbf{x}_i, y_i)$ for $i = 1 \ldots N$, with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$, learn a classifier $f(\mathbf{x})$ such that

$$f(\mathbf{x}_i) \begin{cases} \geq 0 & y_i = +1 \\ < 0 & y_i = -1 \end{cases}$$

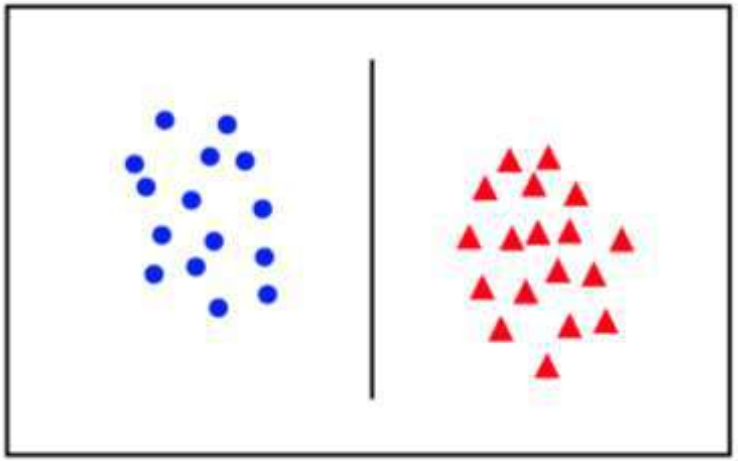
i.e. $y_i f(\mathbf{x}_i) > 0$ for a correct classification.

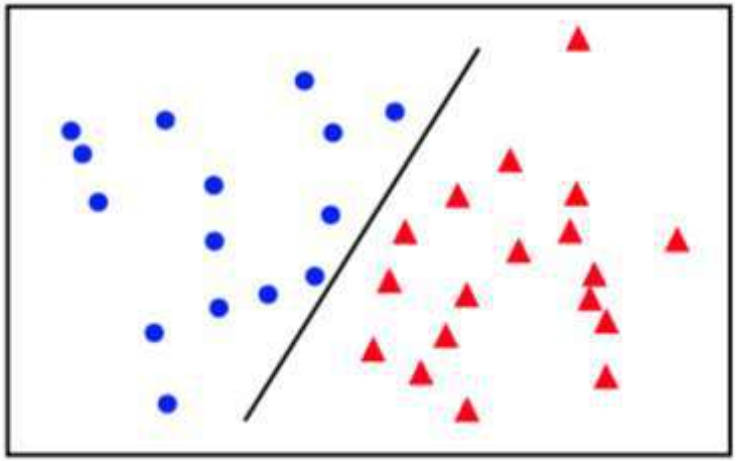




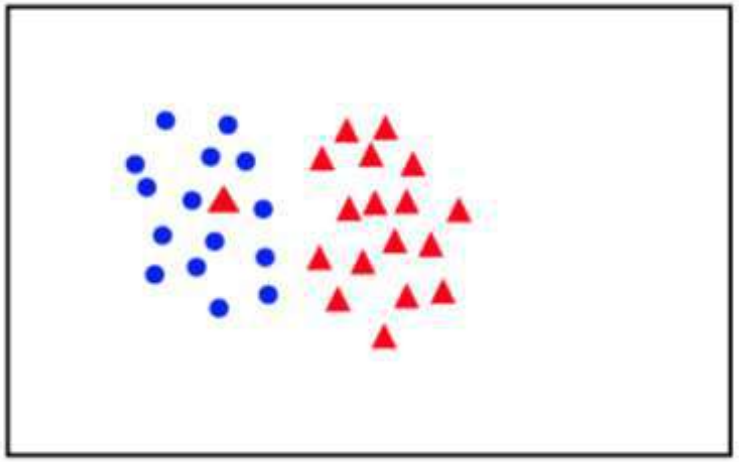
Linear separability

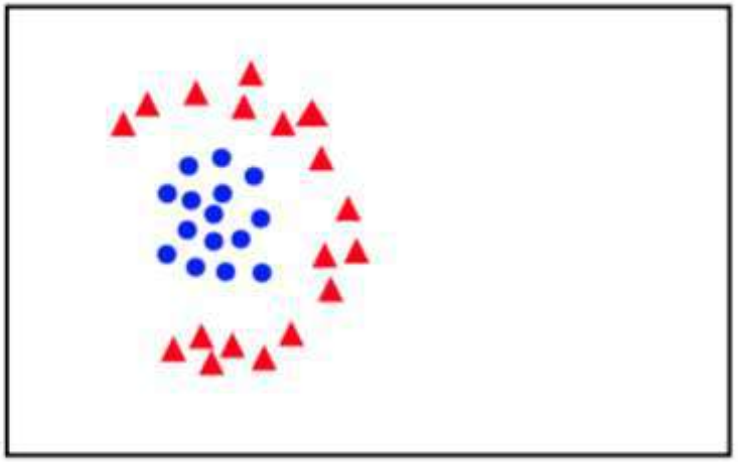
linearly separable





not linearly separable

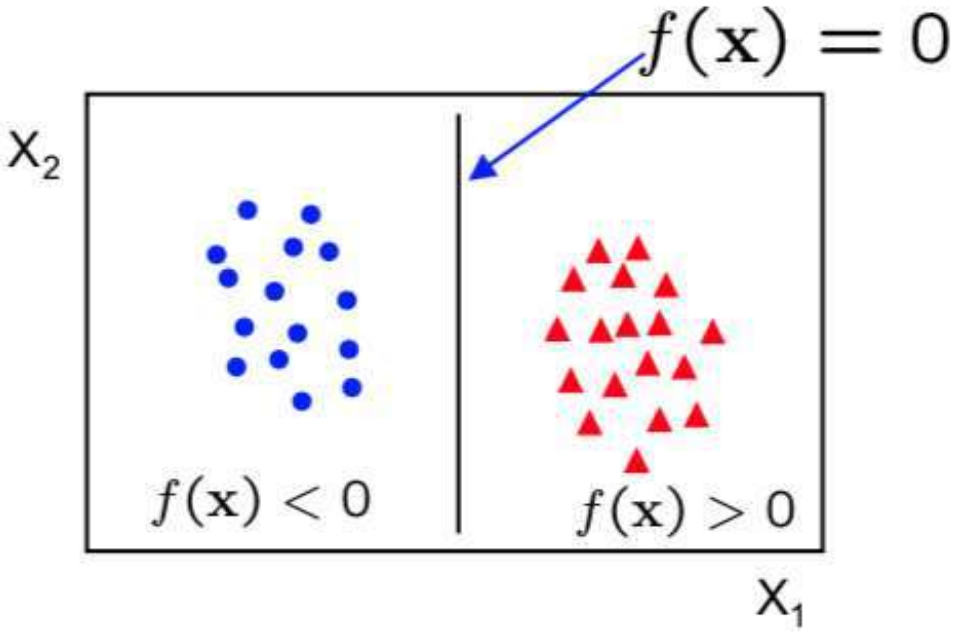




Linear classifiers

A linear classifier has the form

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$$

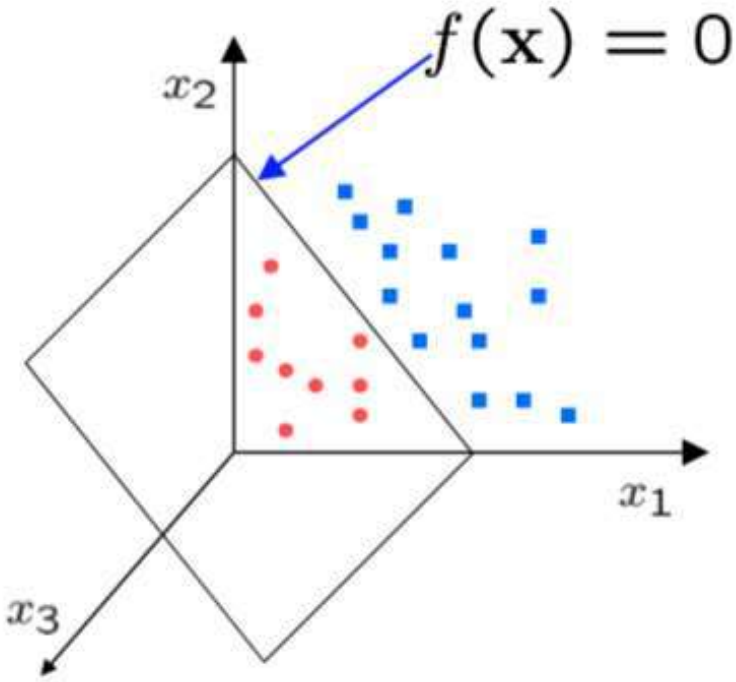


- in 2D the discriminant is a line
- $\mathbf{w}$ is the normal to the line, and b the bias
- $\mathbf{w}$ is known as the weight vector

Linear classifiers

A linear classifier has the form

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$$



- in 3D the discriminant is a plane, and in nD it is a hyperplane

For a K-NN classifier it was necessary to `carry' the training data

For a linear classifier, the training data is used to learn **w** and then discarded

Only **w** is needed for classifying new data

The Perceptron Classifier

Given linearly separable data $\mathbf{x}_i$ labelled into two categories $y_i = \{-1, 1\}$, find a weight vector $\mathbf{w}$ such that the discriminant function

$$f(\mathbf{x}_i) = \mathbf{w}^\top \mathbf{x}_i + b$$

separates the categories for i = 1, .., N

- how can we find this separating hyperplane ?

The Perceptron Algorithm

Write classifier as $f(\mathbf{x}_i) = \tilde{\mathbf{w}}^\top \tilde{\mathbf{x}}_i + w_0 = \mathbf{w}^\top \mathbf{x}_i$

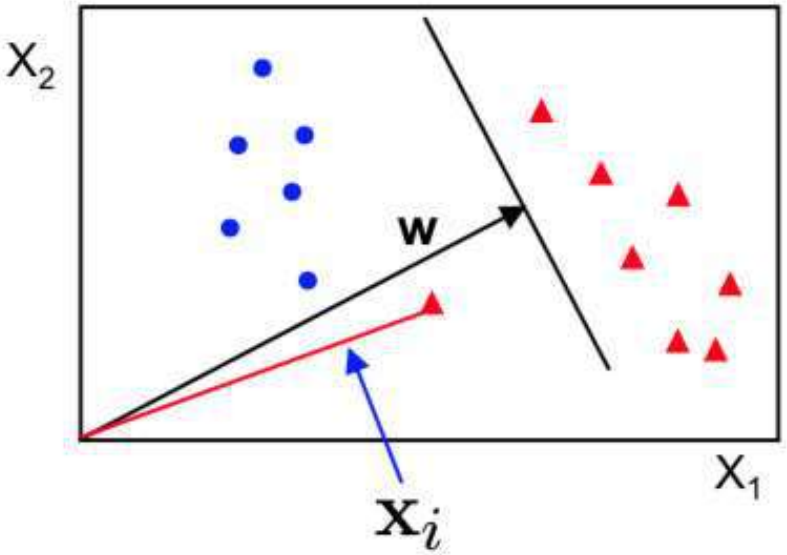
$$\text{where } \mathbf{w} = (\tilde{\mathbf{w}}, w_0), \mathbf{x}_i = (\tilde{\mathbf{x}}_i, 1)$$

- Initialize $\mathbf{w} = 0$
- Cycle though the data points $\{ \mathbf{x}_i, y_i \}$
  - if $\mathbf{x}_i$ is misclassified then $\mathbf{w} \leftarrow \mathbf{w} + \alpha \, \text{sign}(f(\mathbf{x}_i)) \, \mathbf{x}_i$
- Until all the data is correctly classified

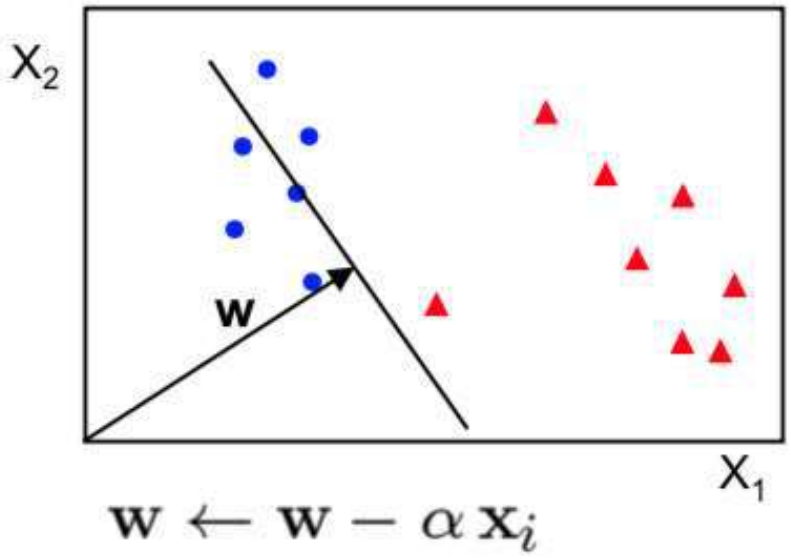
For example in 2D

- Initialize **w** = 0
- Cycle though the data points { $\mathbf{x}_i$, y_i }
  - if $\mathbf{x}_i$ is misclassified then $\mathbf{w} \leftarrow \mathbf{w} + \alpha \, \text{sign}(f(\mathbf{x}_i)) \, \mathbf{x}_i$
- Until all the data is correctly classified

before update

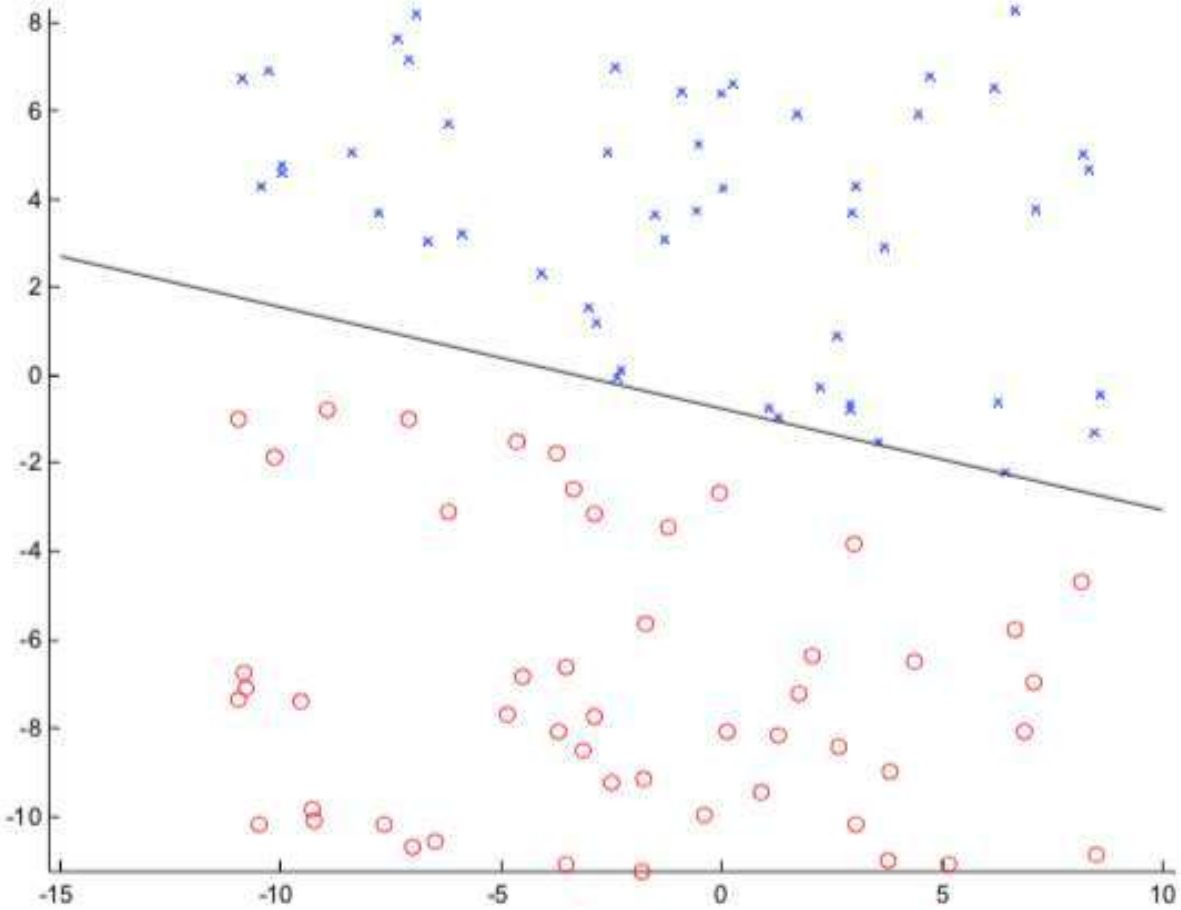


after update



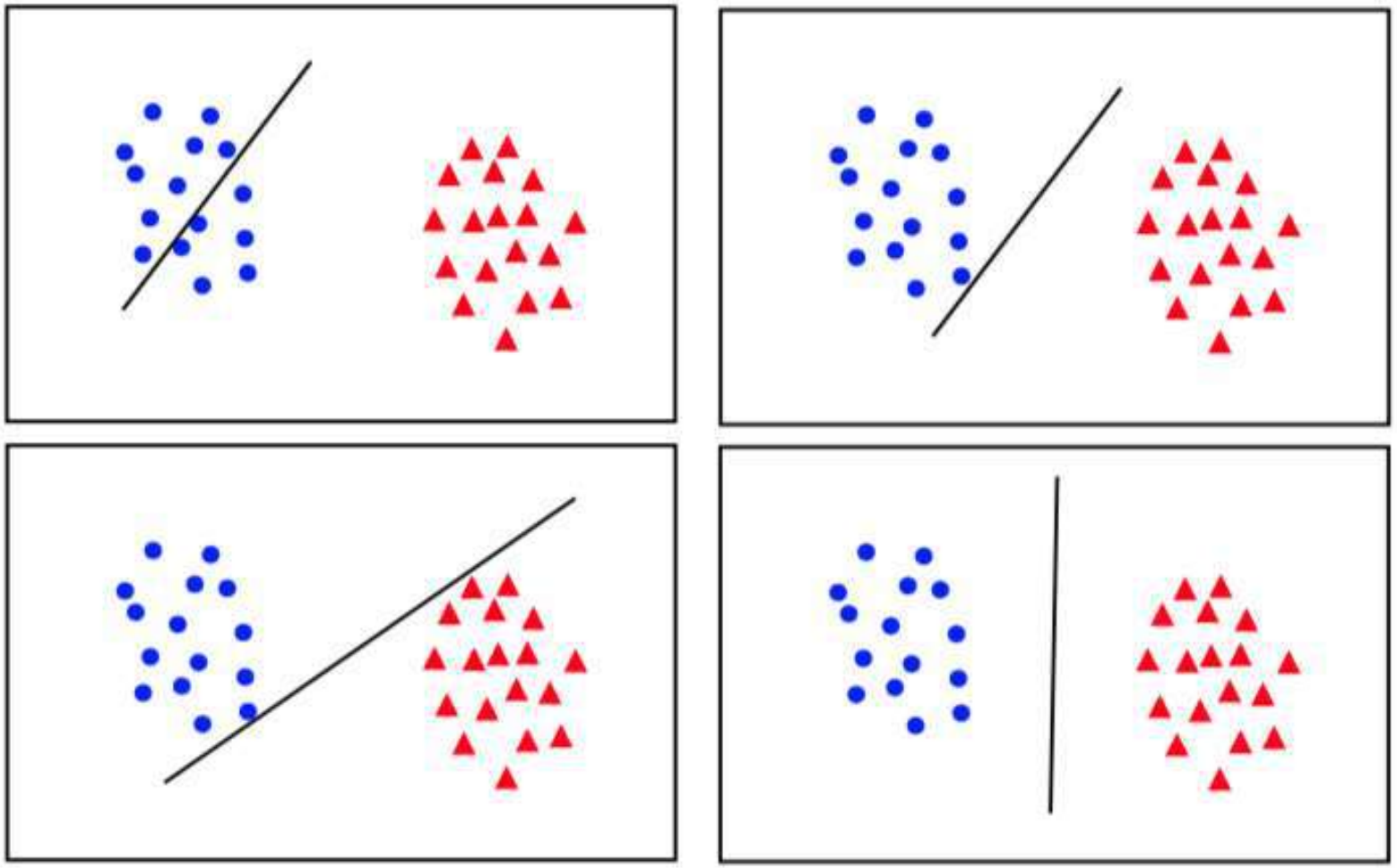
NB after convergence $\mathbf{w} = \sum_i^N \alpha_i \mathbf{x}_i$

Perceptron example



- if the data is linearly separable, then the algorithm will converge
- convergence can be slow …
- separating line close to training data
- we would prefer a larger margin for generalization

What is the best w?

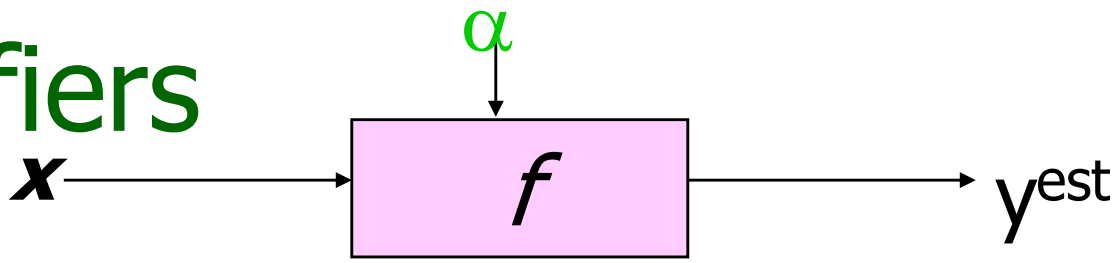


- maximum margin solution: most stable under perturbations of the inputs

Support Vector Machines (SVM)

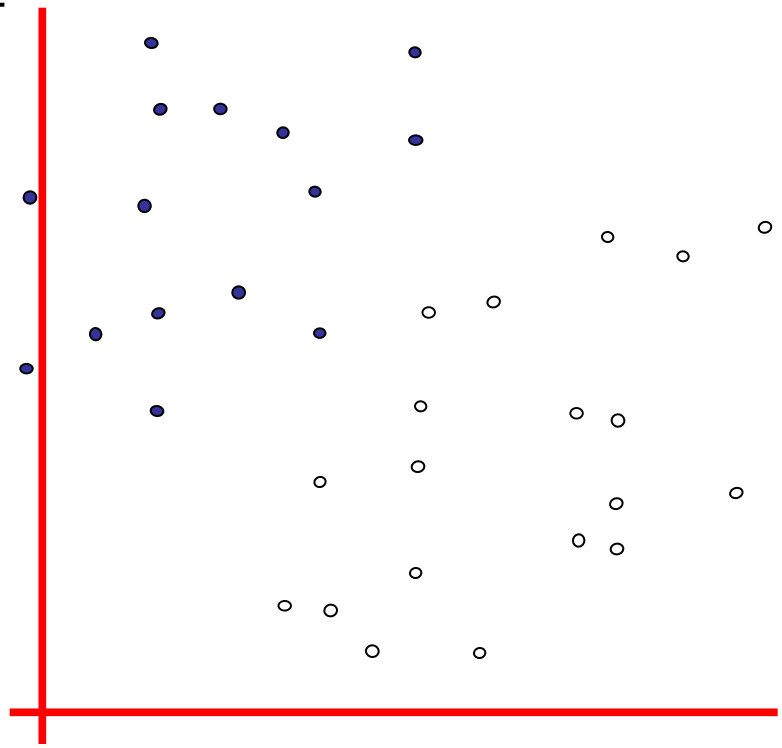
- → Supervised learning methods for classification and regression
- →they can represent non-linear functions and they have an efficient training algorithm
- → derived from statistical learning theory by Vapnik and Chervonenkis (COLT-92)
- → SVM got into mainstream because of their exceptional performance in Handwritten Digit Recognition
  - 1.1% error rate which was comparable to a very carefully constructed (and complex) ANN

Linear Classifiers



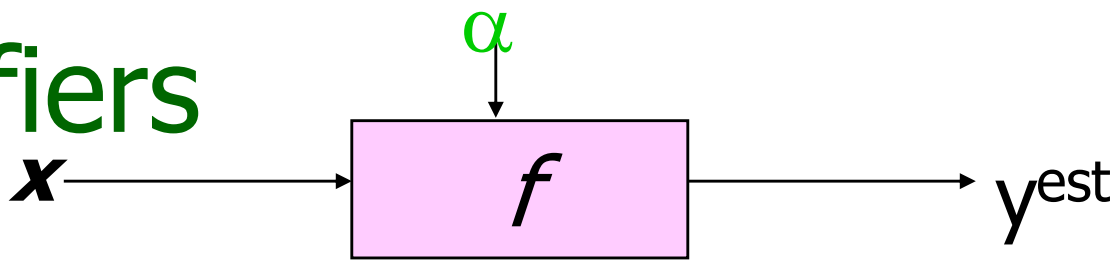
$f(\mathbf{x}, \mathbf{w}, b) = sign(\mathbf{w}.\ \mathbf{x} - b)$

- denotes +1
- denotes -1



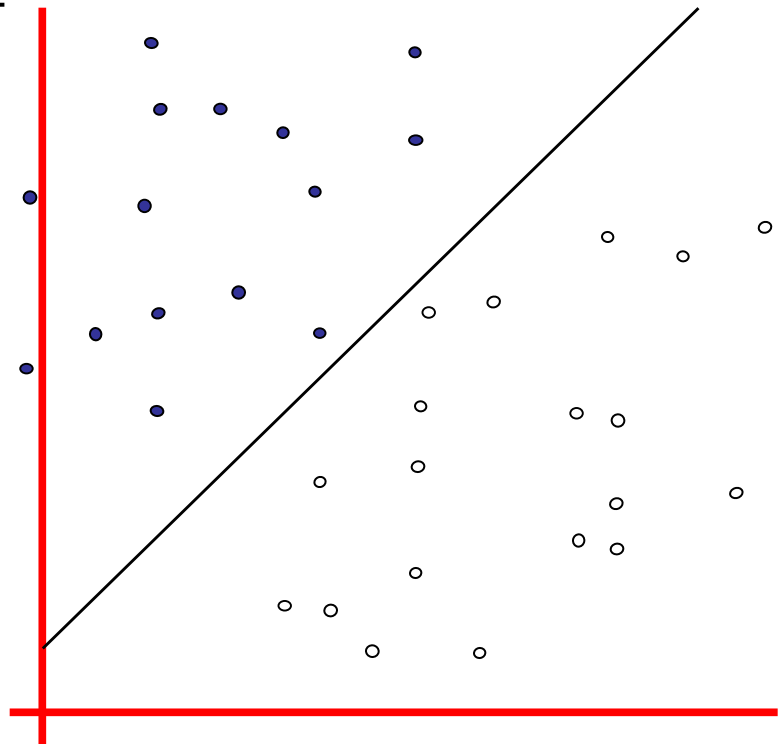
How would you classify this data?

Linear Classifiers



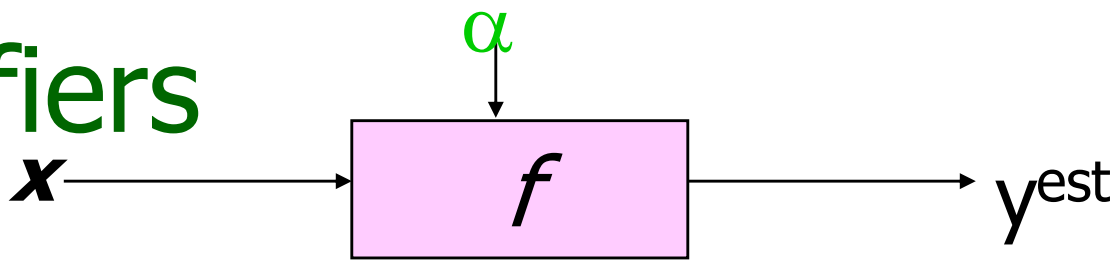
$f(\mathbf{x}, \mathbf{w}, b) = sign(\mathbf{w}.\ \mathbf{x} - b)$

- denotes +1
- denotes -1



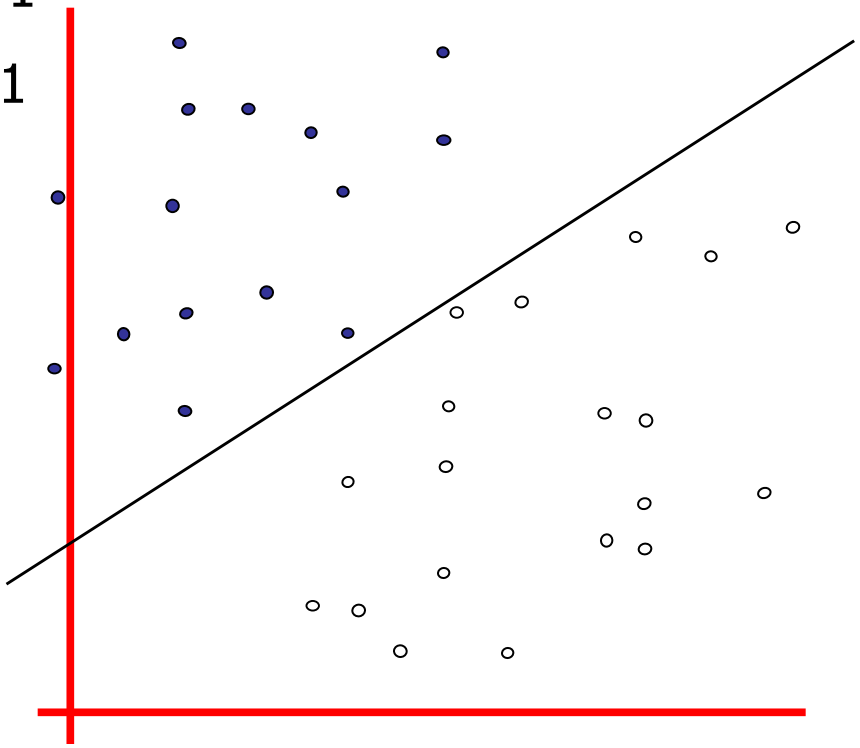
How would you classify this data?

Linear Classifiers



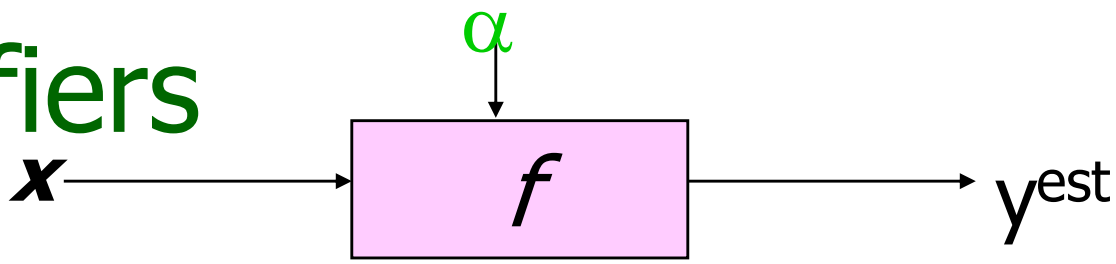
$f(\mathbf{x}, \mathbf{w}, b) = sign(\mathbf{w}.\ \mathbf{x} - b)$

- denotes +1
- denotes -1

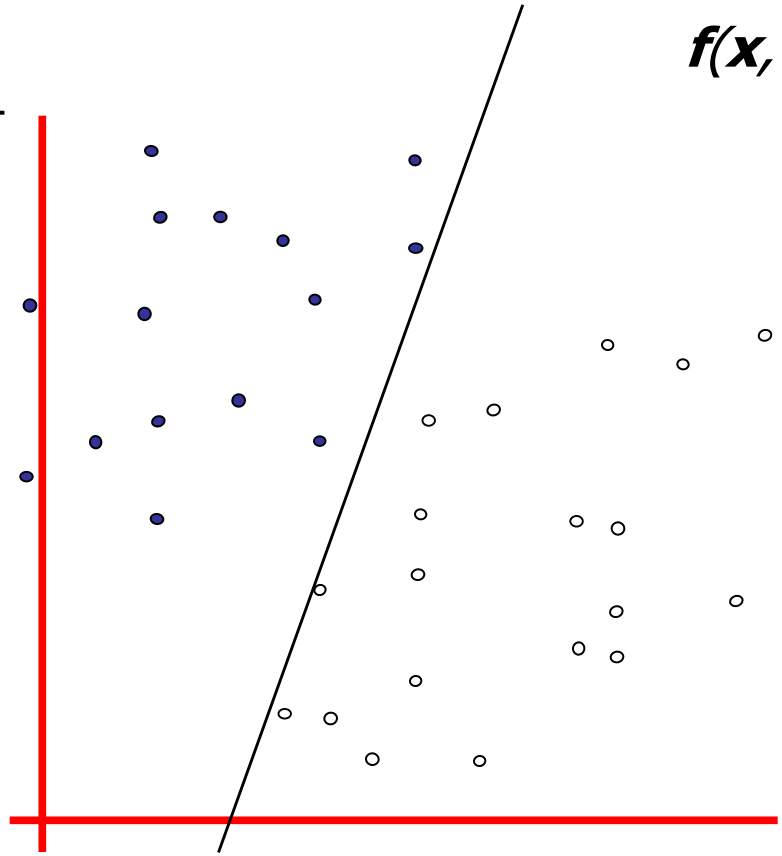


How would you classify this data?

Linear Classifiers



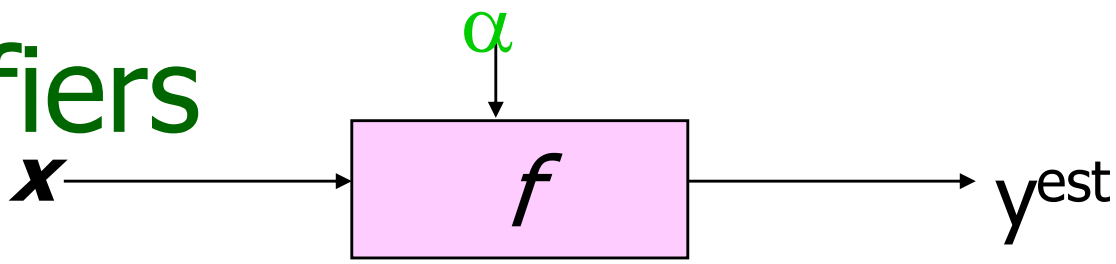
- denotes +1
- denotes -1



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

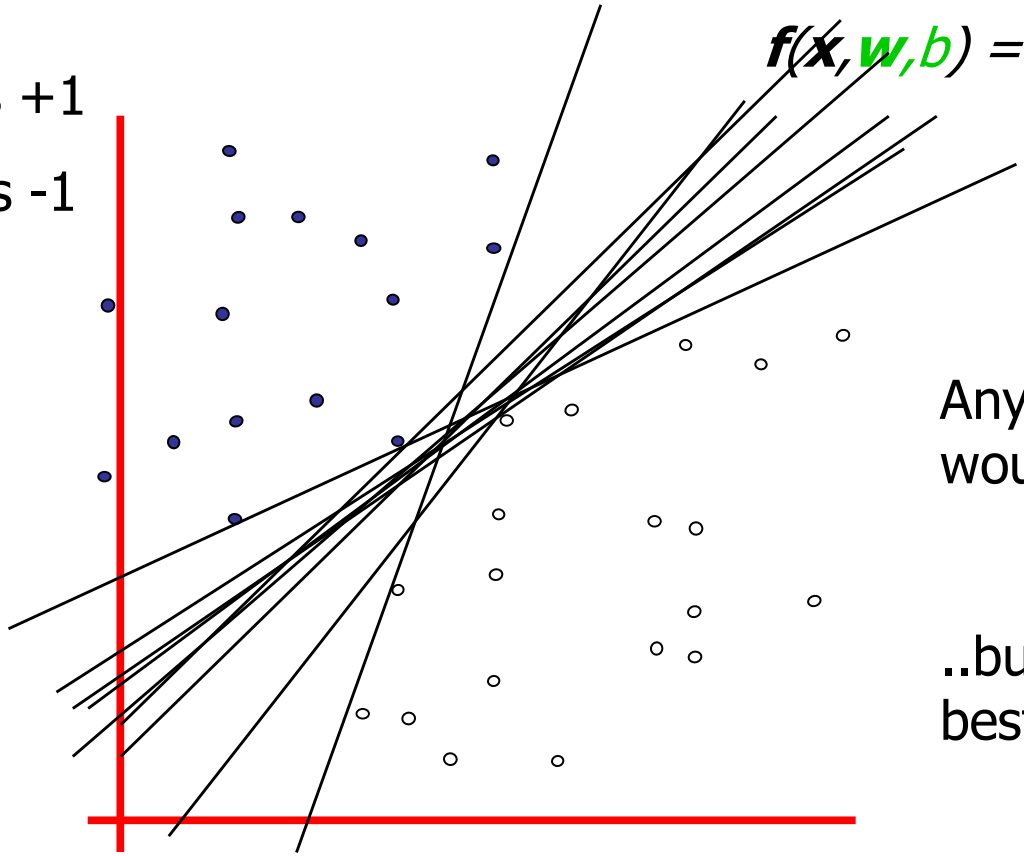
How would you classify this data?

Linear Classifiers



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

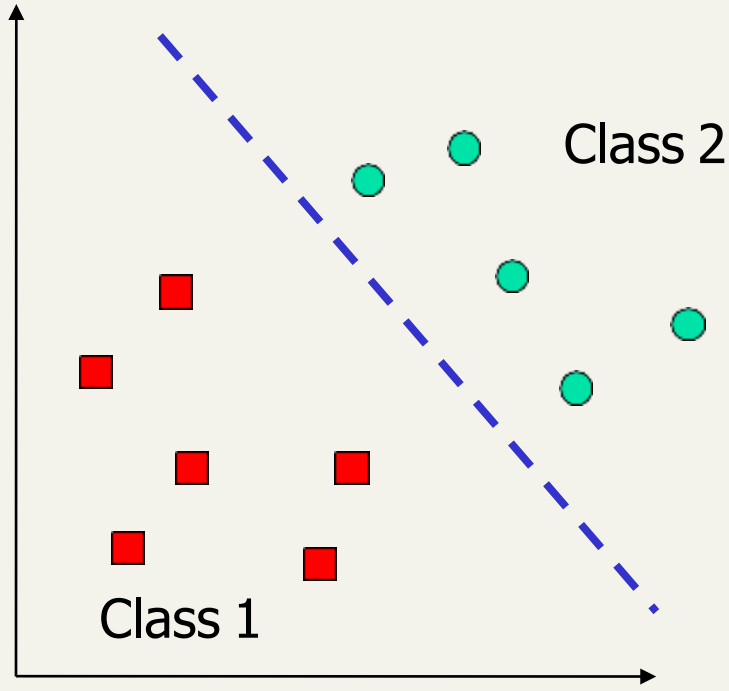
- denotes +1
- denotes -1



Any of these would be fine..

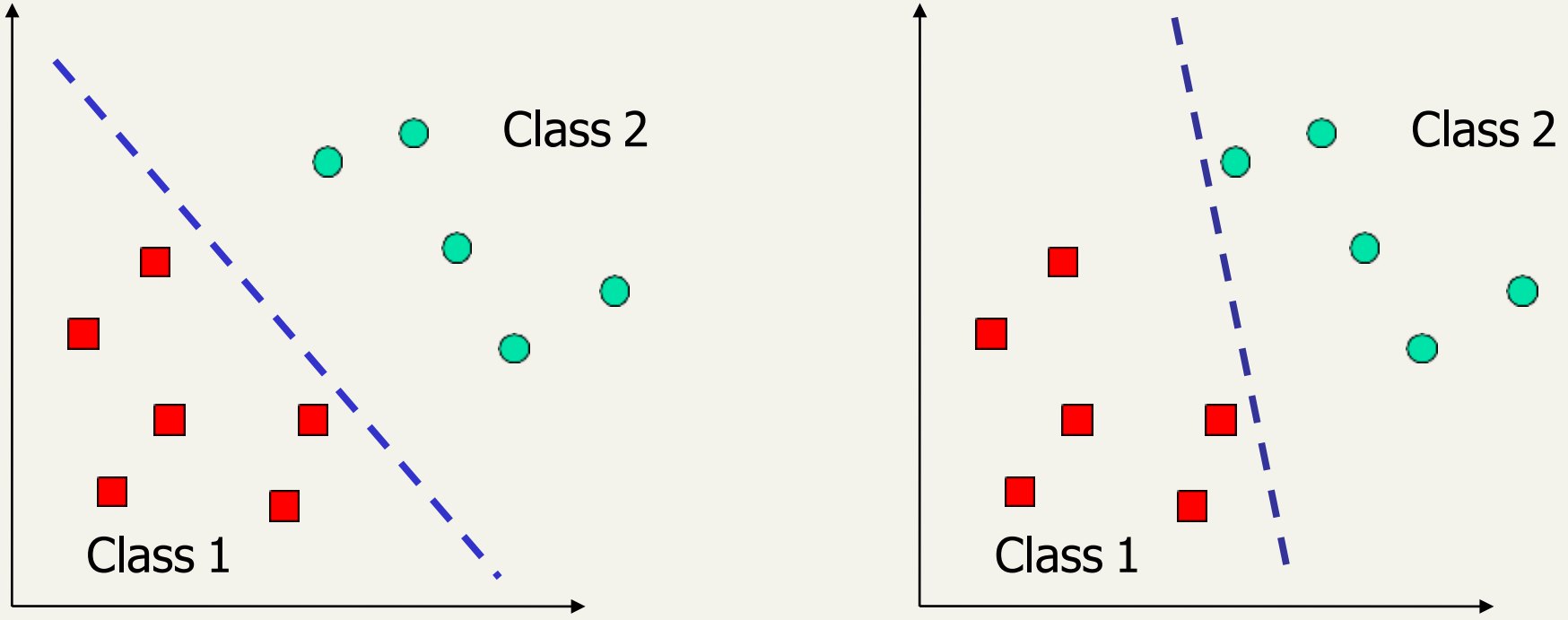
..but which is best?

Two Class Problem: Linear Separable Case



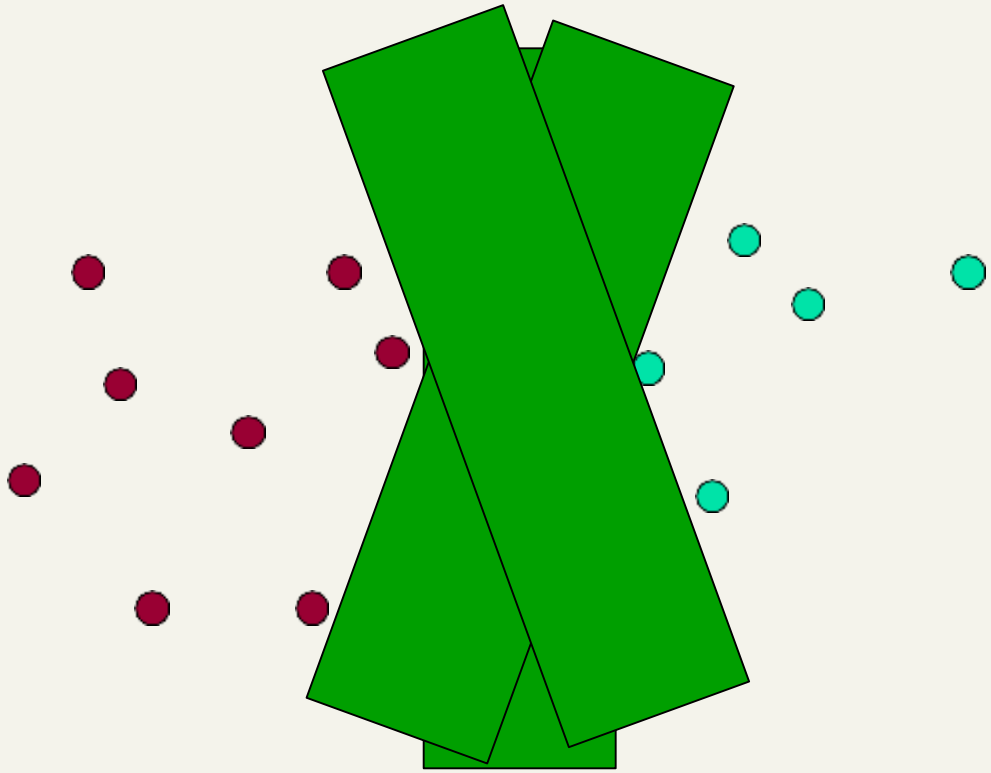
- Many decision boundaries can separate these two classes
- Which one should we choose?

Example of Bad Decision Boundaries

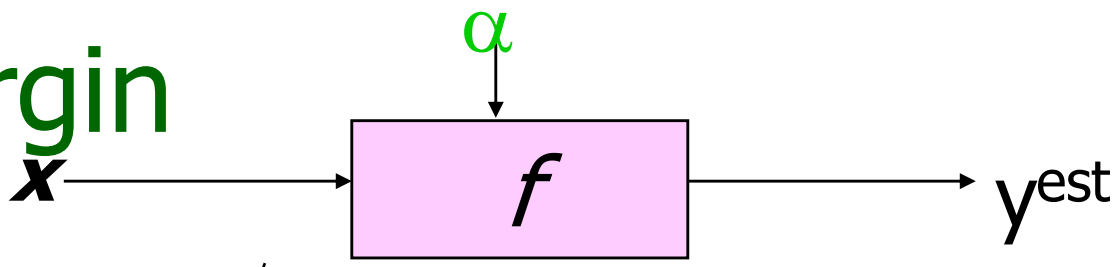


Another intuition

- If you have to place a fat separator between classes, you have less choices, and so the capacity of the model has been decreased

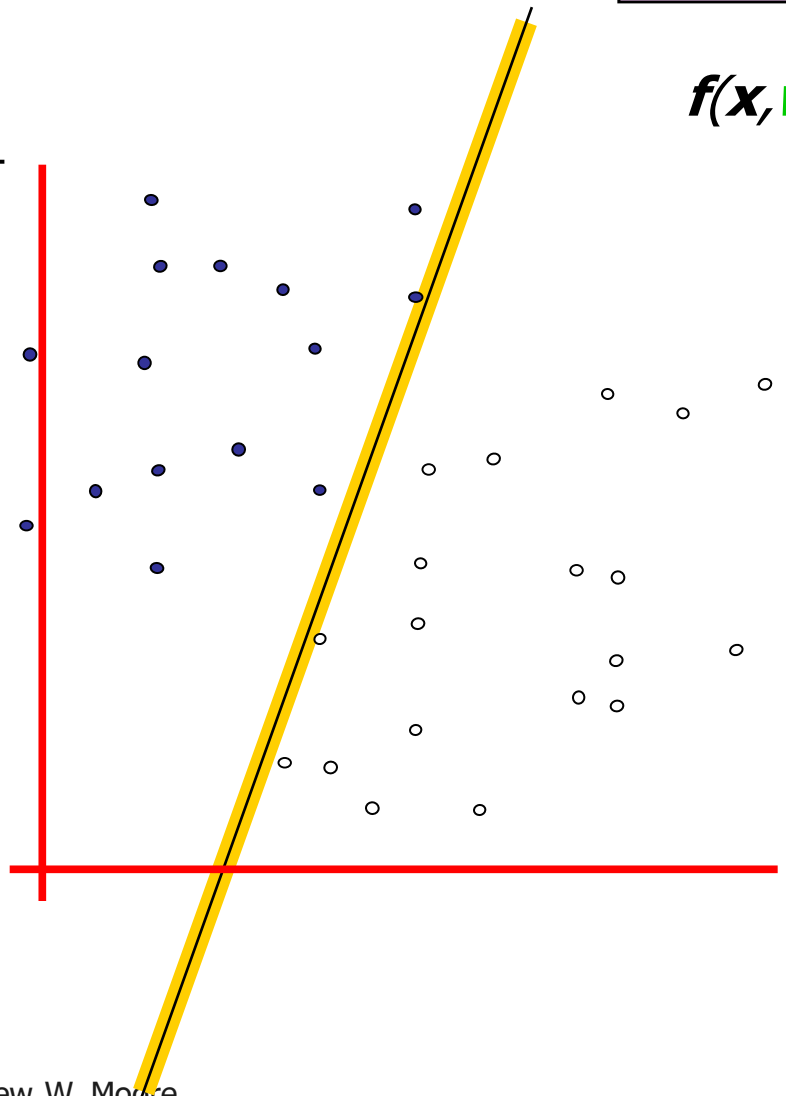


Classifier Margin



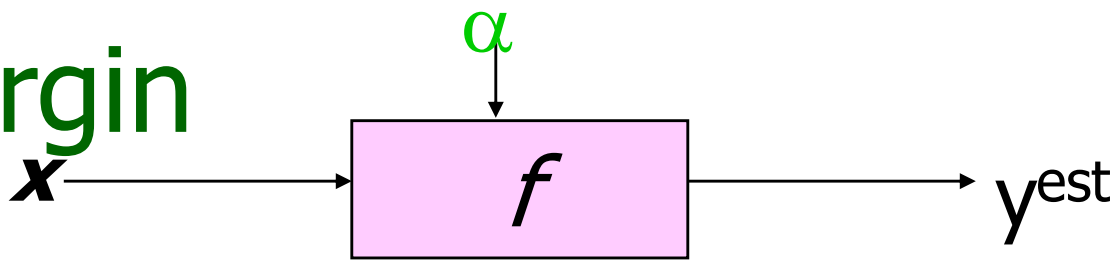
$f(\mathbf{x}, \mathbf{w}, b) = sign(\mathbf{w}.\ \mathbf{x} - b)$

- denotes +1
- denotes -1



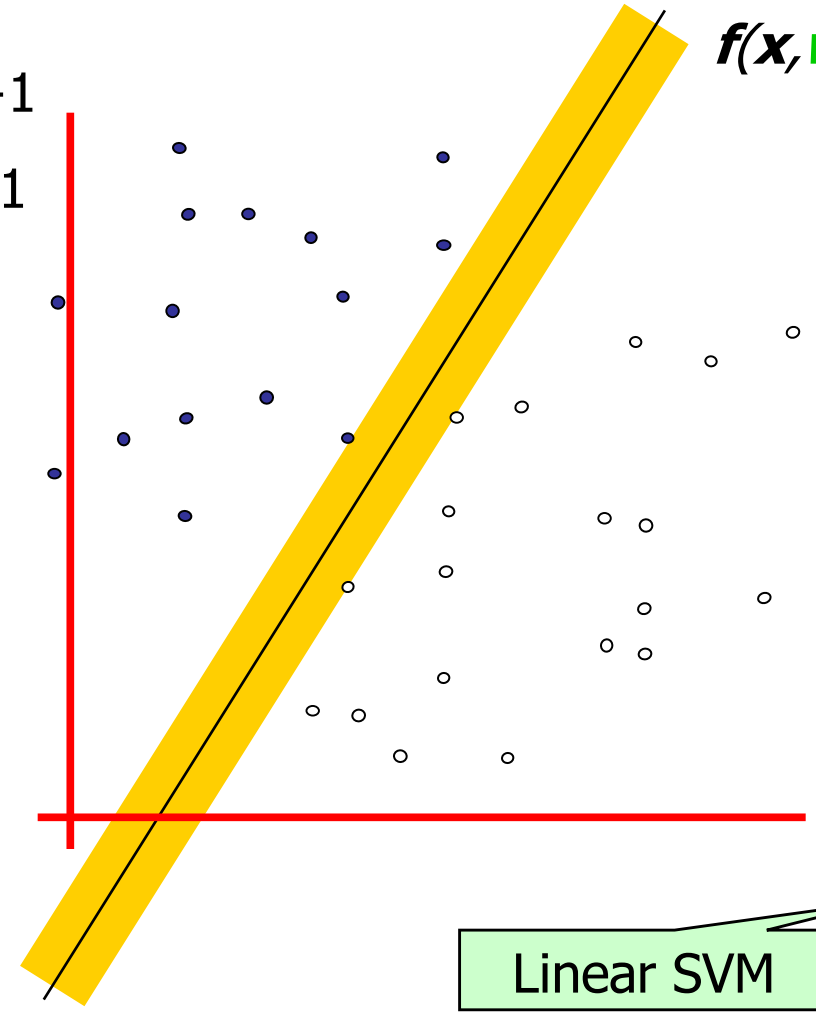
Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

Maximum Margin



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

- denotes +1
- denotes -1

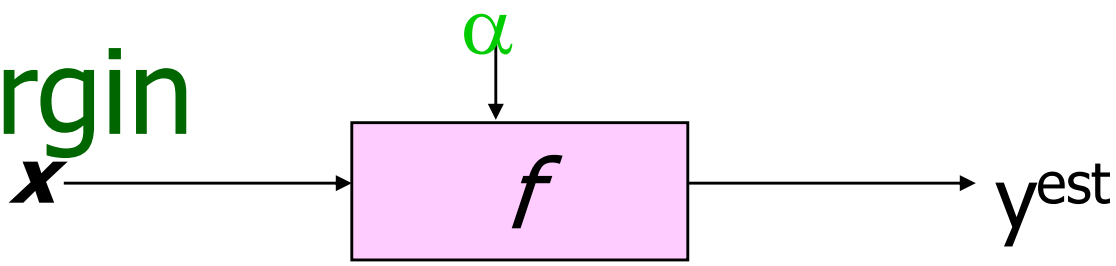


The maximum margin linear classifier is the linear classifier with the, um, maximum margin.

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

Linear SVM

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

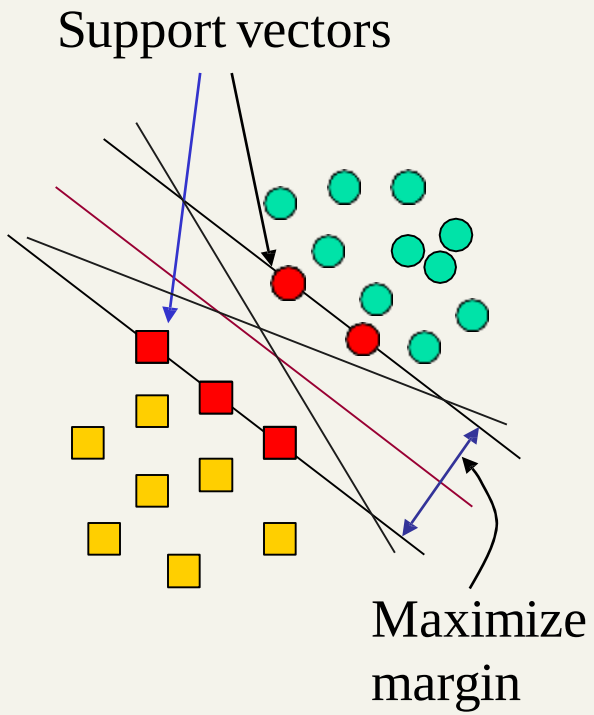
The maximum margin linear classifier is the linear classifier with the, um, maximum margin.

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

Linear SVM

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*.
- *Quadratic programming* problem



Formalization

- w: weight coefficients
- x_i: data point i
- y_i: class result of data point i (+1 or -1)
- Classifier is: $f(x_i) = \text{sign}(w^T x_i + b)$
- Functional margin of x_i is: $y_i (w^T x_i + b)$
  - We can increase this margin by scaling w, b …

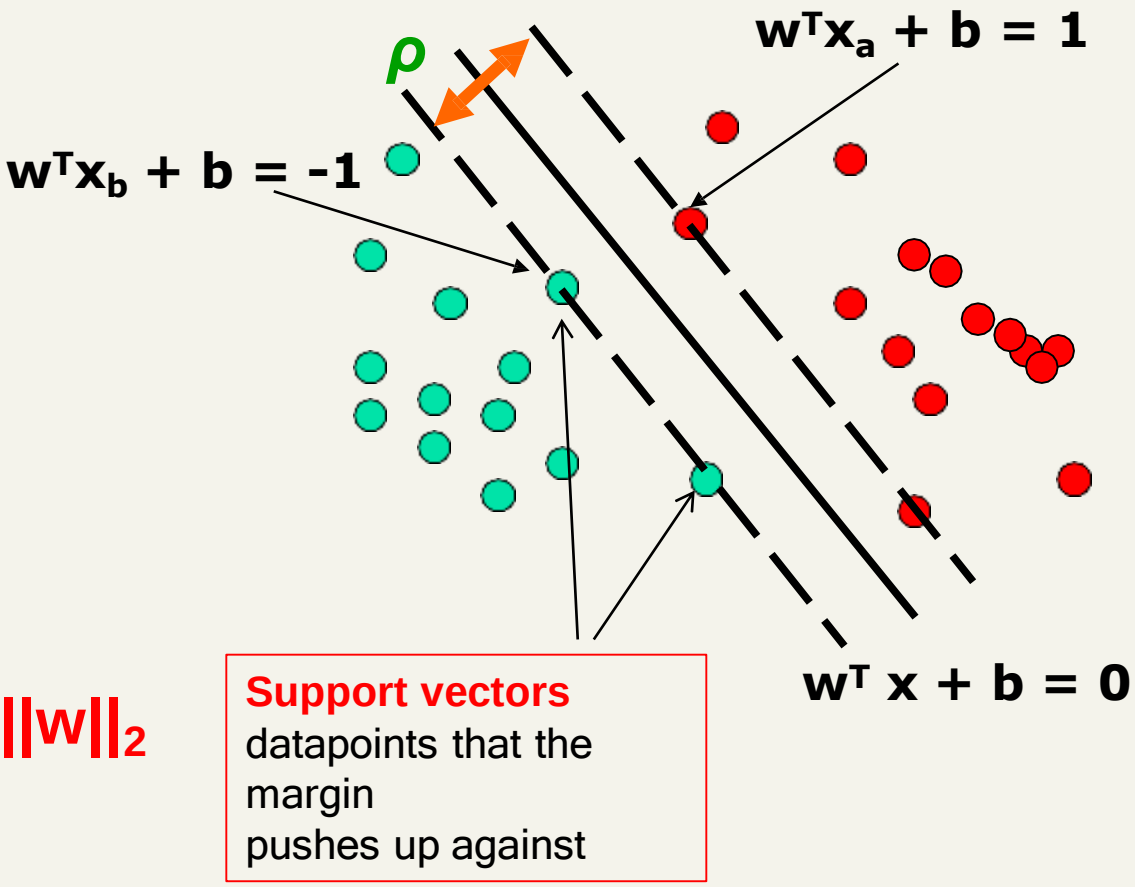
Linear Support Vector Machine (SVM)

- **Hyperplane**

$w^T x + b = 0$

$w^T x + b = 1$

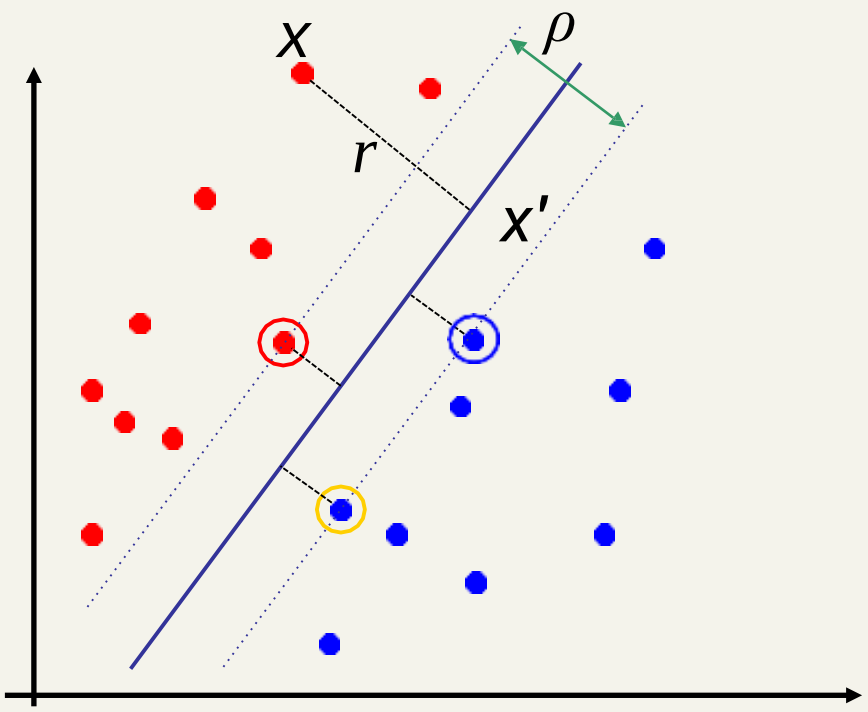
$w^T x + b = -1$



- $\rho = ||x_a - x_b||_2 = 2/||w||_2$

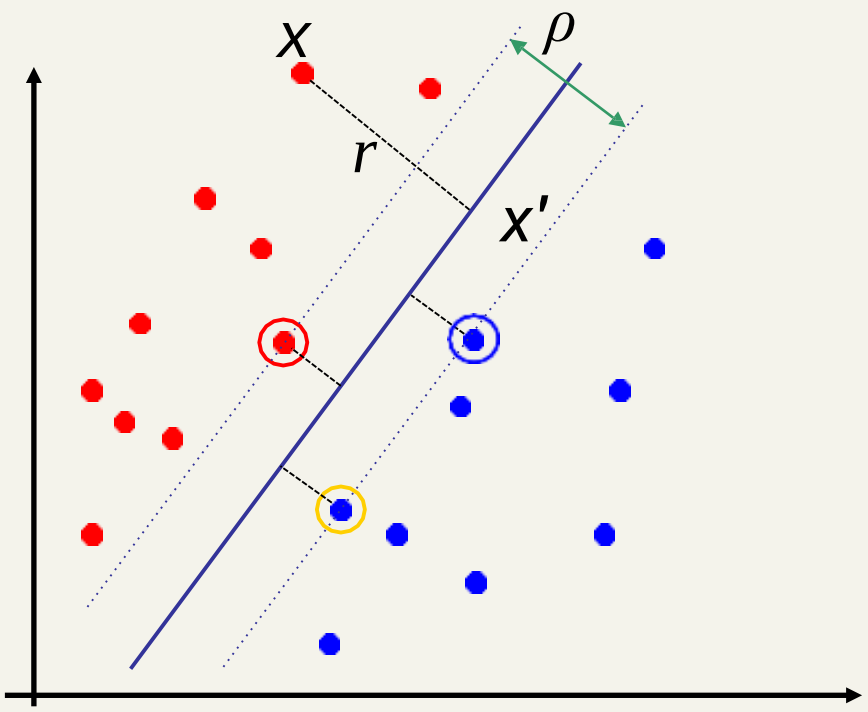
Geometric View: Margin of a point

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



Geometric View of Margin

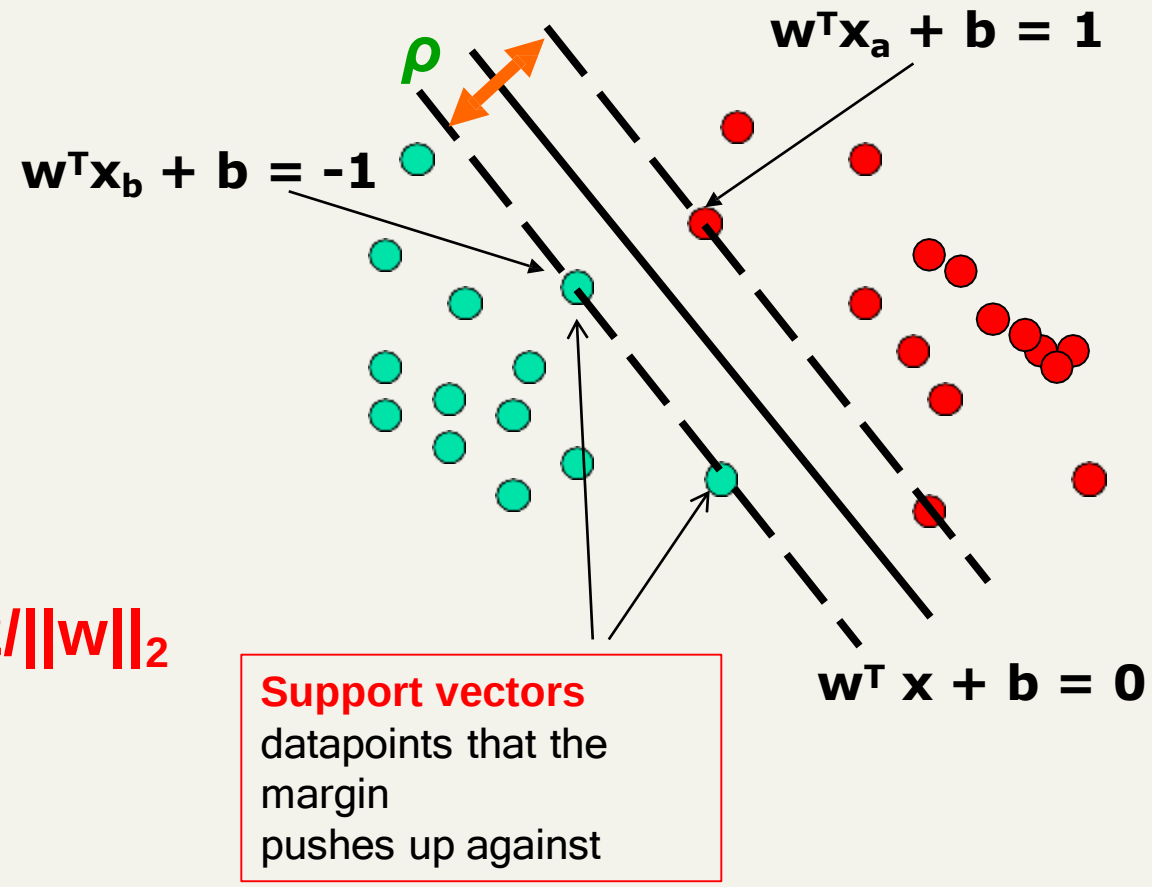
- Distance to the separator is $r = y\dfrac{\mathbf{w}^T\mathbf{x}+b}{\|\mathbf{w}\|}$
- Let X be in line wTx+b=z. Thus (wTx+b) –(wTx'+b)=z-0 then |w| |x-x'|= |z| = y(wTx+b) thus |w| r = y(wTx+b).



Linear Support Vector Machine (SVM)

- **Hyperplane**
  $w^T x + b = 0$

- This implies:
  $w^T(x_a - x_b) = 2$
  $\rho = ||x_a - x_b||_2 = 2/||w||_2$



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 $\{(\mathbf{x_i}, y_i)\}$

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

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

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

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

- The margin of dataset is:

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

The Optimization Problem

- Let $\{x_1, ..., x_n\}$ be our data set and let $y_i \in \{1,-1\}$ be the class label of x_i
- The decision boundary should classify all points correctly $\Rightarrow$
- A constrained optimization problem

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

- $||\mathbf{w}||^2 = \mathbf{w}^T\mathbf{w}$

$$\text{subject to } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \qquad \forall i$$

Lagrangian of Original Problem

Minimize $\frac{1}{2}||\mathbf{w}||^2$ subject to $1-y_i(\mathbf{w}^T\mathbf{x}_i+b) \leq 0$ for $i = 1,$

- The Lagrangian is

Lagrangian multipliers

$$\mathcal{L} = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{n} \alpha_i \left(1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)\right)$$

  - Note that $||\mathbf{w}||^2 = \mathbf{w}^T\mathbf{w}$
- Setting the gradient of $\mathcal{L}$ w.r.t. **w** and b to zero,

$$\mathbf{w} + \sum_{i=1}^{n} \alpha_i(-y_i)\mathbf{x}_i = \mathbf{0} \quad \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

$\alpha_i \geq 0$

The Dual Optimization Problem

- We can transform the problem to its dual

Dot product of X

$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

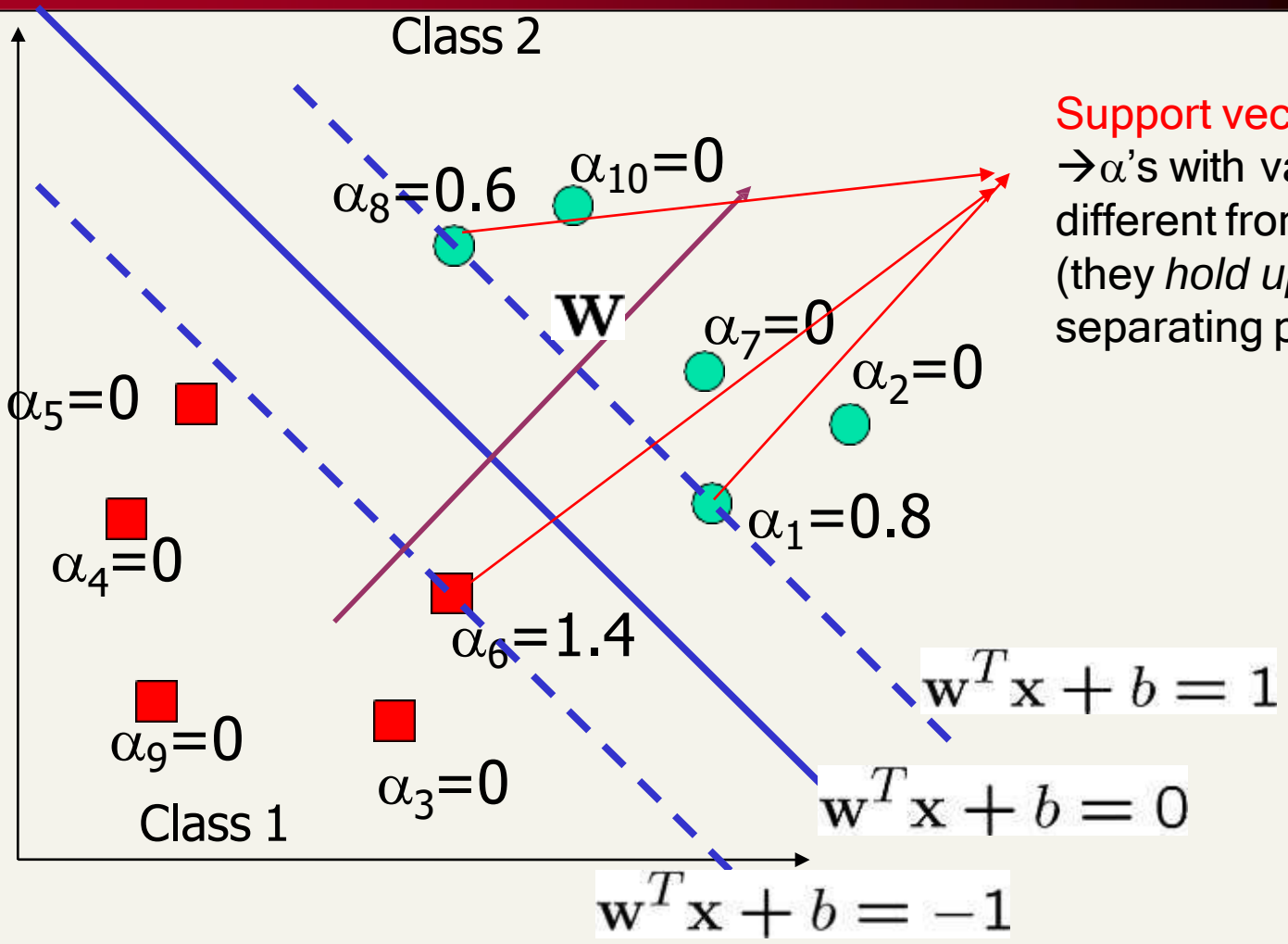
α's → New variables
(Lagrangian multipliers)

- This is a convex quadratic programming (QP) problem
  - Global maximum of α_i can always be found

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

→well established tools for solving this optimization problem (e.g. cplex)

A Geometrical Interpretation



Support vectors
→α's with values different from zero (they *hold up* the separating plane)!

The Optimization Problem Solution

- The solution has the form:

$$\mathbf{w} = \sum \alpha_i y_i \mathbf{x_i} \qquad b = y_k - \mathbf{w}^T \mathbf{x_k} \text{ for any } \mathbf{x_k} \text{ such that } \alpha_k \neq 0$$

- Each non-zero α_i indicates that corresponding $\mathbf{x_i}$ is a support vector.
- Then the classifying function will have the form:

$$f(\mathbf{x}) = \sum \alpha_i y_i \mathbf{x_i}^T \mathbf{x} + b$$

- Notice that it relies on an *inner product* between the test point **x** and the support vectors $\mathbf{x_i}$ – we will return to this later.
- Also keep in mind that solving the optimization problem involved computing the inner products $\mathbf{x_i}^T \mathbf{x_j}$ between all pairs of training points.

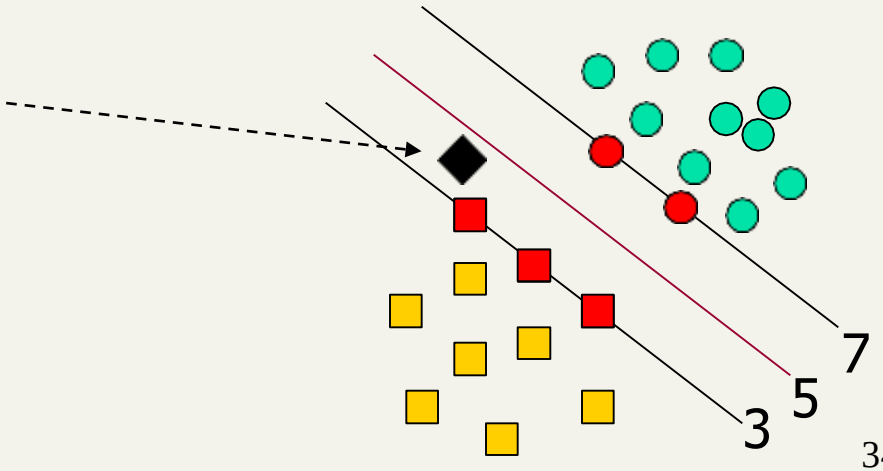
Classification with SVMs

- Given a new point (x_1,x_2), we can score its projection onto the hyperplane normal:
  - In 2 dims: score = $w_1x_1+w_2x_2+b$.
    - I.e., compute score: $wx + b = \Sigma\alpha_i y_i \mathbf{x_i}^T\mathbf{x} + b$
  - Set confidence threshold t.

Score > t: yes

Score < -t: no

Else: don't know



- Given a two-class linearly separable dataset of N points of the form
- $(\vec{x}_1, y_1), (\vec{x}_2, y_2) \ldots \ldots \ldots (\vec{x}_N, y_N)$
- where the y_i's are either $+1$ or -1

	X1	X2	Class
x_1	2	2	-1
x_2	4	5	+1
x_3	7	4	+1

- Find $\vec{\alpha} = (\alpha_1, \alpha_2, \ldots\ldots, \alpha_N)$ which maximizes

$$\phi(\vec{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{N} \alpha_i \alpha_j y_i y_j (\vec{x}_i \cdot \vec{x}_j)$$

- Subject to,

$$\sum_{i=1}^{N} \alpha_i y_i = 0$$

$$\text{for } i = 1, 2, \ldots, N$$

- $-\alpha_1 + \alpha_2 + \alpha_3 = 0$

X1	X2	Class
2	2	-1
4	5	+1
7	4	+1

- Compute,

$$\vec{w} = \sum_{i=1}^{N} \alpha_i y_i \vec{x}_i$$

- Compute

$$b = \frac{1}{2}\left(\min_{i:y_i=+1}(\vec{w}\cdot\vec{x}_i) + \max_{i:y_i=-1}(\vec{w}\cdot\vec{x}_i)\right)$$

- The SVM classifier function is given by

$$f(\vec{x}) = \vec{w}\cdot\vec{x} - b$$

X1	X2	Class
2	2	-1
4	5	+1
7	4	+1

- Using the SVM algorithm, find the hyperplane with maximum margin for the following data.

X1	X2	Class
2	1	+1
4	3	-1

- $N = 2$
- $\vec{x}_1 = (2, 1)$
- $\vec{x}_2 = (4, 3)$
- $y_1 = +1$
- $y_2 = -1$
- $\vec{\alpha} = (\alpha_1, \alpha_2)$
- subject to the conditions
- $\alpha_1 - \alpha_2 = 0$ *means* $\alpha_1 = \alpha_2$
- $\alpha_1 > 0, \alpha_2 > 0$

$+\alpha_1$

$-\alpha_2$

$$f(\vec{x}) = \vec{w} \cdot \vec{x} - b$$

X1	X2	Class
2	1	+1
4	3	-1

$$\phi(\vec{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{N} \alpha_i \alpha_j y_i y_j (\vec{x}_i \cdot \vec{x}_j)$$

$$= (\alpha_1 + \alpha_2) - \frac{1}{2}[\alpha_1 \alpha_1 y_1 y_1 (\vec{x}_1 \cdot \vec{x}_1) + \alpha_1 \alpha_2 y_1 y_2 (\vec{x}_1 \cdot \vec{x}_2) +$$

$$\alpha_2 \alpha_1 y_2 y_1 (\vec{x}_2 \cdot \vec{x}_1) + \alpha_2 \alpha_2 y_2 y_2 (\vec{x}_2 \cdot \vec{x}_2)]$$

$$= (\alpha_1 + \alpha_2) -$$

$$\frac{1}{2}[\alpha_1^2 (+1)(+1)(2 \times 2 + 1 \times 1) + \alpha_1 \alpha_2 (+1)(-1)(2 \times 4 + 1 \times 3) +$$

$$\alpha_2 \alpha_1 (-1)(+1)(4 \times 2 + 3 \times 1) + \alpha_2^2 (-1)(-1)(4 \times 4 + 3 \times 3)]$$

$$= (\alpha_1 + \alpha_2) - \frac{1}{2}[5\alpha_1^2 - 22\alpha_1 \alpha_2 + 25\alpha_2^2]$$

X1	X2	Class
2	1	+1
4	3	-1

$N = 2$

$\vec{x}_1 = (2, 1)$

$\vec{x}_2 = (4, 3)$

$y_1 = +1$

$y_2 = -1$

$\vec{\alpha} = (\alpha_1, \alpha_2)$

$\alpha_1 = \alpha_2$

$\alpha_1, \alpha_2 > 0$

- Find values of α_1 and α_2 which maximizes

$$\phi(\vec{\alpha}) = (\alpha_1 + \alpha_2) - \frac{1}{2}\left[5\alpha_1^2 - 22\alpha_1\alpha_2 + 25\alpha_2^2\right]$$

$$\phi(\vec{\alpha}) = 2\alpha_1 - 4\alpha_1^2$$

- For Ø to be maximum we must have

$$\frac{d\phi}{d\alpha_1} = 2 - 8\alpha_1 = 0$$

$$\alpha_1 = \frac{1}{4}$$

$$\alpha_2 = \frac{1}{4}$$

X1	X2	Class
2	1	+1
4	3	-1

$N = 2$

$\vec{x}_1 = (2, 1)$

$\vec{x}_2 = (4, 3)$

$y_1 = +1$

$y_2 = -1$

$\vec{\alpha} = (\alpha_1, \alpha_2)$

$\alpha_1 = \alpha_2$

$\alpha_1, \alpha_2 > 0$

$$\vec{w} = \sum_{i=1}^{N} \alpha_i y_i \vec{x}_i$$

$$= \alpha_1 y_1 \vec{x}_1 + \alpha_2 y_2 \vec{x}_2$$

$$= \frac{1}{4}(+1)(2,1) + \frac{1}{4}(-1)(4,3)$$

$$= \frac{1}{4}(-2,-2)$$

$$= (-\tfrac{1}{2}, -\tfrac{1}{2})$$

$$\alpha_1 = \frac{1}{4}$$

$$\alpha_2 = \frac{1}{4}$$

X1	X2	Class
2	1	+1
4	3	-1

$$N = 2$$
$$\vec{x}_1 = (2,1)$$
$$\vec{x}_2 = (4,3)$$
$$y_1 = +1$$
$$y_2 = -1$$
$$\vec{\alpha} = (\alpha_1, \alpha_2)$$
$$\alpha_1 = \alpha_2$$
$$\alpha_1, \alpha_2 > 0$$

$$b = \frac{1}{2}\left(\min_{i:y_i=+1}(\vec{w}\cdot\vec{x}_i) + \max_{i:y_i=-1}(\vec{w}\cdot\vec{x}_i)\right)$$

$$= \frac{1}{2}\left((\vec{w}\cdot\vec{x}_1) + (\vec{w}\cdot\vec{x}_2)\right)$$

$$= \frac{1}{2}\left((-\tfrac{1}{2}\times 2 - \tfrac{1}{2}\times 1) + (-\tfrac{1}{2}\times 4 - \tfrac{1}{2}\times 3)\right)$$

$$= \frac{1}{2}\left(-\frac{10}{2}\right) = -\frac{5}{2}$$

$$\alpha_1 = \frac{1}{4}$$

$$\alpha_2 = \frac{1}{4}$$

$$\vec{w} = (-\tfrac{1}{2}, -\tfrac{1}{2})$$

X1	X2	Class
2	1	+1
4	3	-1

$N = 2$

$\vec{x}_1 = (2, 1)$

$\vec{x}_2 = (4, 3)$

$y_1 = +1$

$y_2 = -1$

$\vec{\alpha} = (\alpha_1, \alpha_2)$

$\alpha_1 = \alpha_2$

$\alpha_1, \alpha_2 > 0$

- The SVM classifier function is given by

$$f(\vec{x}) = \vec{w} \cdot \vec{x} - b$$

$$= (-\tfrac{1}{2}, -\tfrac{1}{2}) \cdot (x_1, x_2) - (-\tfrac{5}{2})$$

$$= -\frac{1}{2}x_1 - \frac{1}{2}x_2 + \frac{5}{2}$$

$$= -\frac{1}{2}(x_1 + x_2 - 5)$$

X1	X2	Class
2	1	+1
4	3	-1

$$\alpha_1 = \frac{1}{4}$$

$$\alpha_2 = \frac{1}{4}$$

$$\vec{w} = (-\tfrac{1}{2}, -\tfrac{1}{2})$$

$$b = -\frac{5}{2}$$

$N = 2$

$\vec{x}_1 = (2, 1)$

$\vec{x}_2 = (4, 3)$

$y_1 = +1$

$y_2 = -1$

$\vec{\alpha} = (\alpha_1, \alpha_2)$

$\alpha_1 = \alpha_2$

$\alpha_1, \alpha_2 > 0$

- The equation of the maximal margin hyperplane is

$$f(\vec{x}) = 0$$

$$f(\vec{x}) = -\frac{1}{2}(x_1 + x_2 - 5)$$

$$-\frac{1}{2}(x_1 + x_2 - 5) = 0$$

$$x_1 + x_2 - 5 = 0$$

$$\alpha_1 = \frac{1}{4}$$

$$\alpha_2 = \frac{1}{4}$$

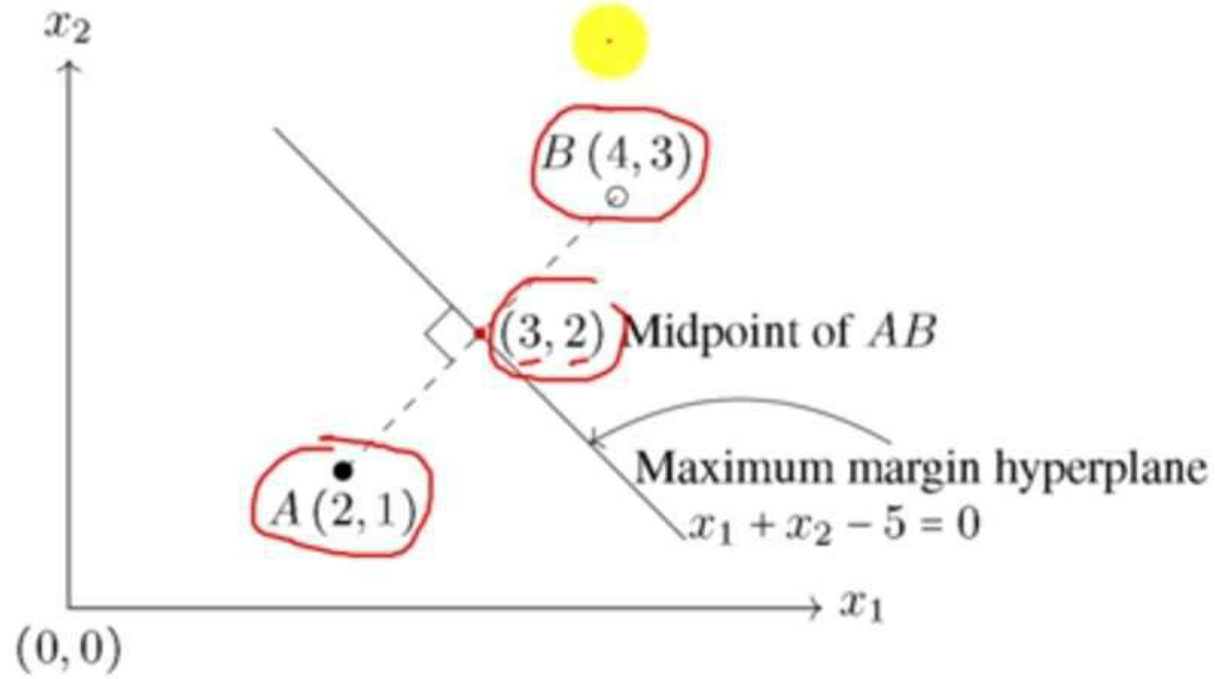
$$\vec{w} = (-\tfrac{1}{2}, -\tfrac{1}{2})$$

$$b = -\frac{5}{2}$$

X1	X2	Class
2	1	+1
4	3	-1

$$N = 2$$
$$\vec{x}_1 = (2, 1)$$
$$\vec{x}_2 = (4, 3)$$
$$y_1 = +1$$
$$y_2 = -1$$
$$\vec{\alpha} = (\alpha_1, \alpha_2)$$
$$\alpha_1 = \alpha_2$$
$$\alpha_1, \alpha_2 > 0$$

- The equation $x_1 + x_2 - 5 = 0$ is the perpendicular bisector of the line segment joining the points (2, 1) and (4, 3)

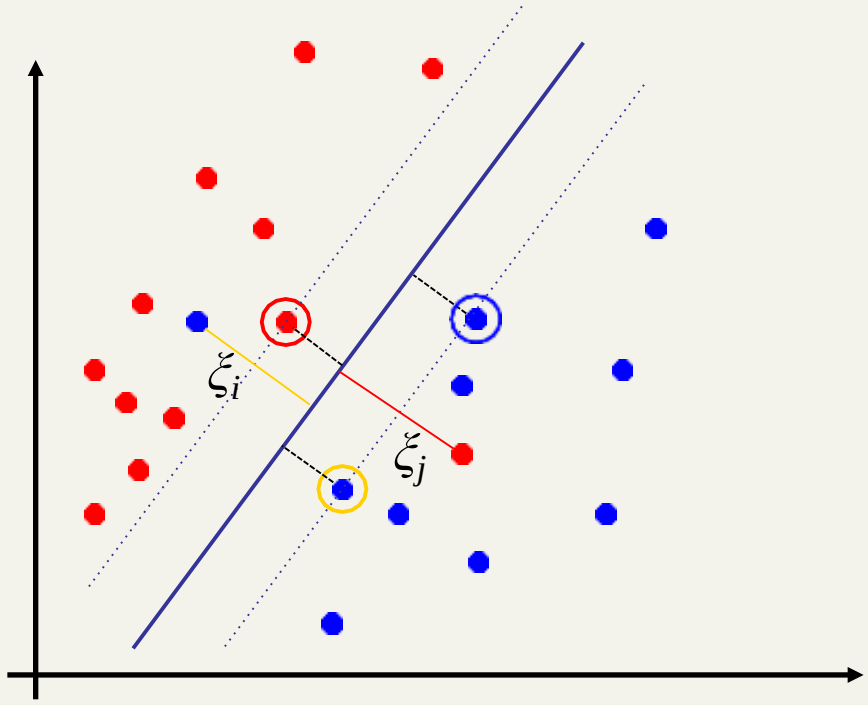


X1	X2	Class
2	1	+1
4	3	-1

$$N = 2$$
$$\vec{x}_1 = (2,1)$$
$$\vec{x}_2 = (4,3)$$
$$y_1 = +1$$
$$y_2 = -1$$
$$\vec{\alpha} = (\alpha_1, \alpha_2)$$
$$\alpha_1 = \alpha_2$$
$$\alpha_1, \alpha_2 > 0$$

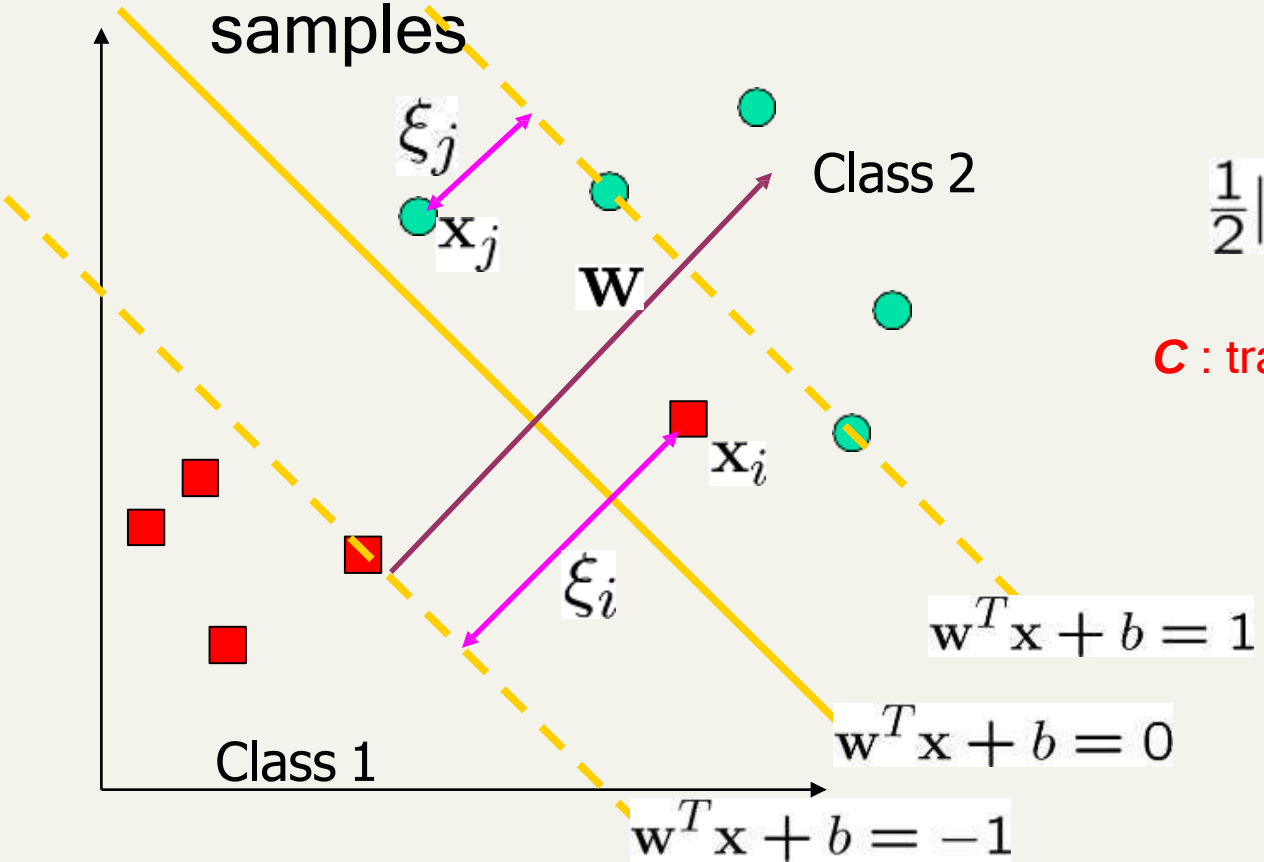
Soft Margin Classification

- If the training set is not linearly separable, *slack variables* ξ_i can be added to allow misclassification of difficult or noisy examples.
- Allow some errors
  - Let some points be moved to where they belong, at a cost
- Still, try to minimize training set errors, and to place hyperplane "far" from each class (large margin)



Soft margin

- We allow "error" ξ_i in classification; it is based on the output of the discriminant function $\mathbf{w}^T\mathbf{x}+b$
- ξ_i approximates the number of misclassified samples



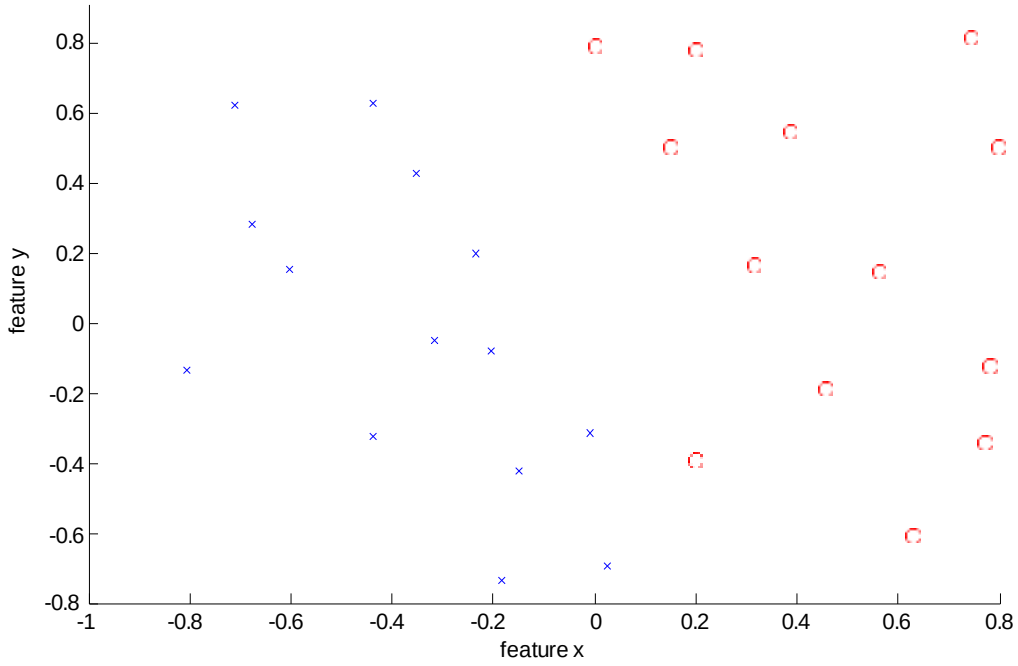
New objective function:

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

C : tradeoff parameter between error and margin; chosen by the user; large C means a higher penalty to errors

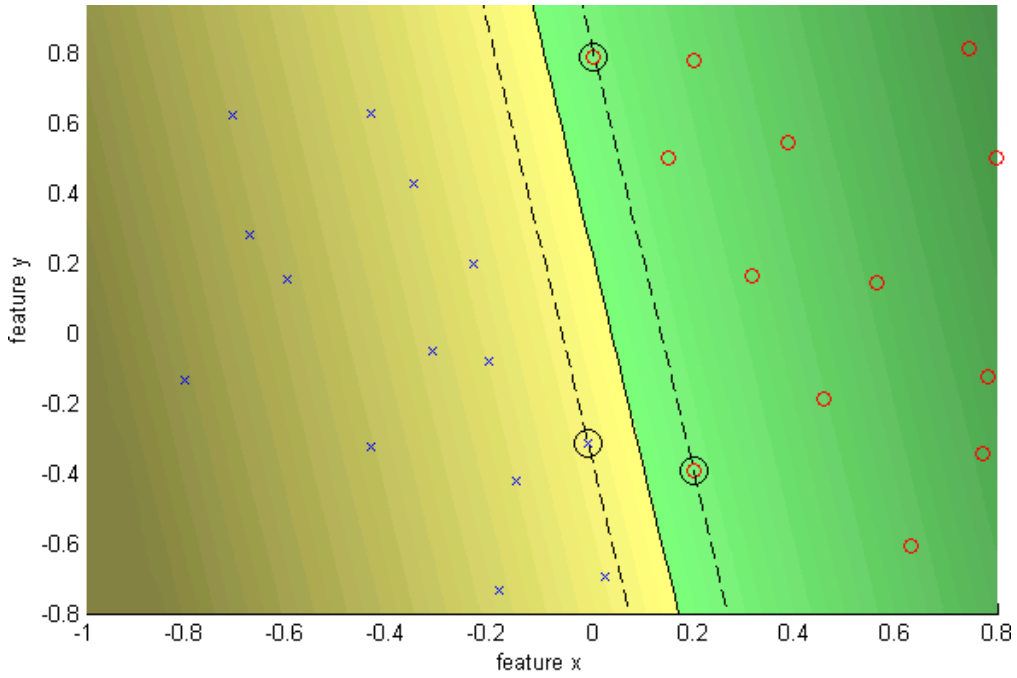
“Soft” margin solution

- Every constraint can be satisfied if ξ_i is sufficiently large
- C is a regularization parameter:
  - small C allows constraints to be easily ignored → large margin
  - large C makes constraints hard to ignore → narrow margin
  - $C = \infty$ enforces all constraints: hard margin
- This is still a quadratic optimization problem and there is a unique minimum. Note, there is only one parameter, C.



- data is linearly separable
- but only with a narrow margin

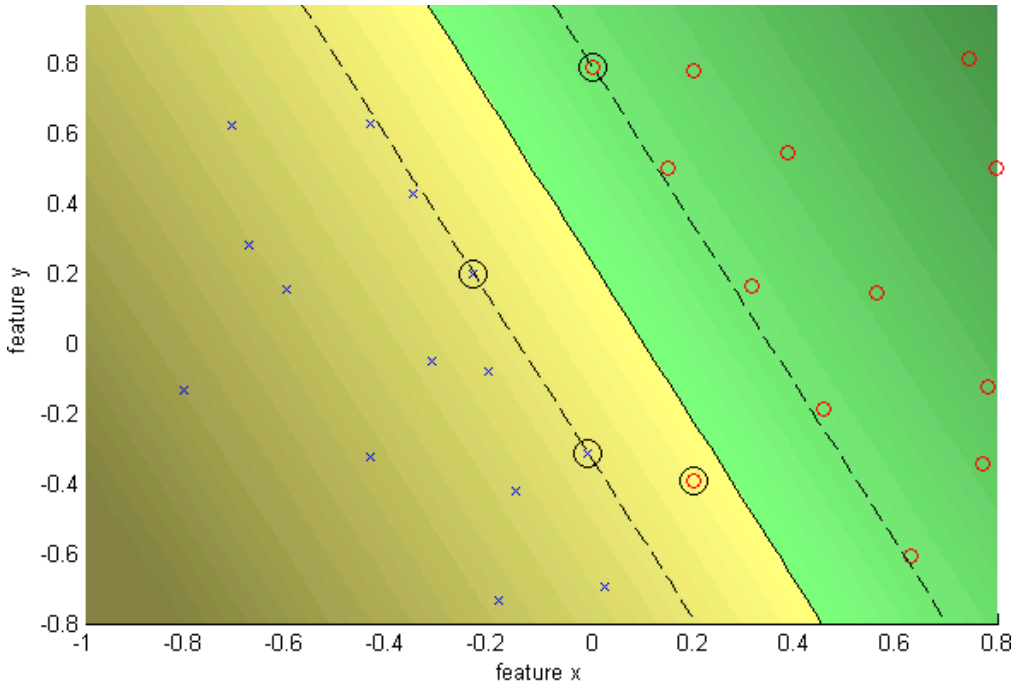
C = Infinity hard margin



Comment Window

```
SVM (L1) by Sequential Minimal Optimizer
Kernel: linear (-), C: Inf
Kernel evaluations: 971
Number of Support Vectors: 3
Margin: 0.0966
Training error: 0.00%
```

C = 10 soft margin



Comment Window

```
SVM (L1) by Sequential Minimal Optimizer
Kernel: linear (-), C: 10.0000
Kernel evaluations: 2645
Number of Support Vectors: 4
Margin: 0.2265
Training error: 3.70%
```

Soft Margin Classification Mathematically

- The old formulation:

Find $\mathbf{w}$ and b such that
$\mathbf{\Phi(w)} = \frac{1}{2}\,\mathbf{w}^T\mathbf{w}$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$
$y_i\,(\mathbf{w}^T\mathbf{x_i} + b) \geq 1$

- The new formulation incorporating slack variables:

Find $\mathbf{w}$ and b such that
$\mathbf{\Phi(w)} = \frac{1}{2}\,\mathbf{w}^T\mathbf{w} + C\sum\xi_i$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$
$y_i\,(\mathbf{w}^T\mathbf{x_i} + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$ for all i

- Parameter C can be viewed as a way to control overfitting – a regularization term

The Optimization Problem

$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

- **w** is also recovered as $\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$
- The only difference with the linear separable case is that there is an upper bound *C* on α_i
- Once again, a QP solver can be used to find α_i efficiently!!!

Soft Margin Classification – Solution

- The dual problem for soft margin classification:

Find $\alpha_1 \ldots \alpha_N$ such that
$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x_i}^T\mathbf{x_j}$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $0 \leq \alpha_i \leq C$ for all α_i

- Neither slack variables ξ_i nor their Lagrange multipliers appear in the dual problem!
- Again, $\mathbf{x_i}$ with non-zero α_i will be support vectors.
- Solution to the dual problem is:

$\mathbf{w} = \Sigma\alpha_i y_i \mathbf{x_i}$
$b = y_k(1 - \xi_k) - \mathbf{w}^T\mathbf{x_k}$ where $k = \underset{k}{\text{argmax}}\ \alpha_k$

But **w** not needed explicitly for classification!

$f(\mathbf{x}) = \Sigma\alpha_i y_i \mathbf{x_i}^T\mathbf{x} + b$

Linear SVMs: Summary

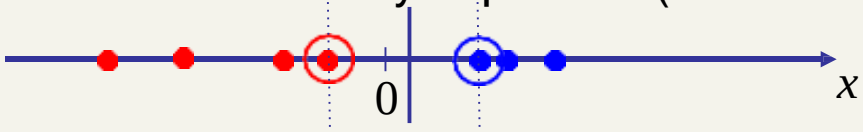
- The classifier is a *separating hyperplane.*
- Most "important" training points are support vectors; they define the hyperplane.
- Quadratic optimization algorithms can identify which training points $\mathbf{x}_i$ are support vectors with non-zero Lagrangian multipliers α_i.
- Both in the dual formulation of the problem and in the solution training points appear only inside inner products:

Find $\alpha_1 \ldots \alpha_N$ such that
$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $0 \le \alpha_i \le C$ for all α_i

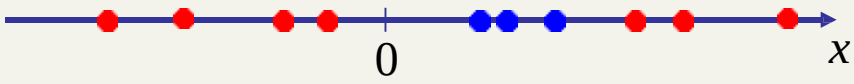
$$f(\mathbf{x}) = \Sigma\alpha_i y_i \mathbf{x}_i^T\mathbf{x} + b$$

Non-linear SVMs

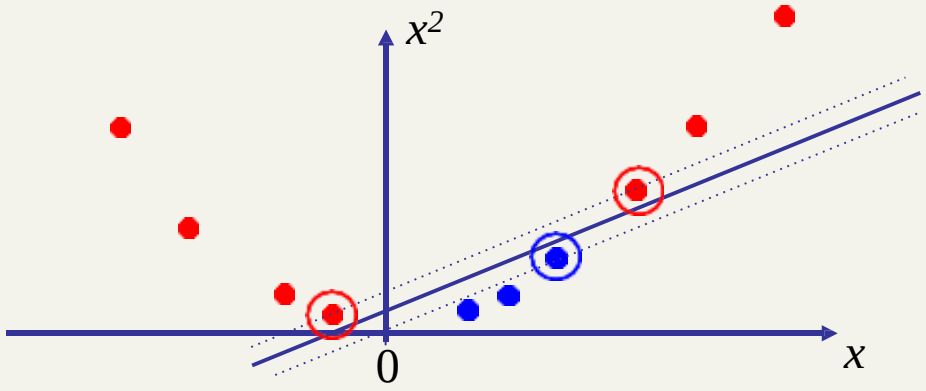
- Datasets that are linearly separable (with some noise) work out great:



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

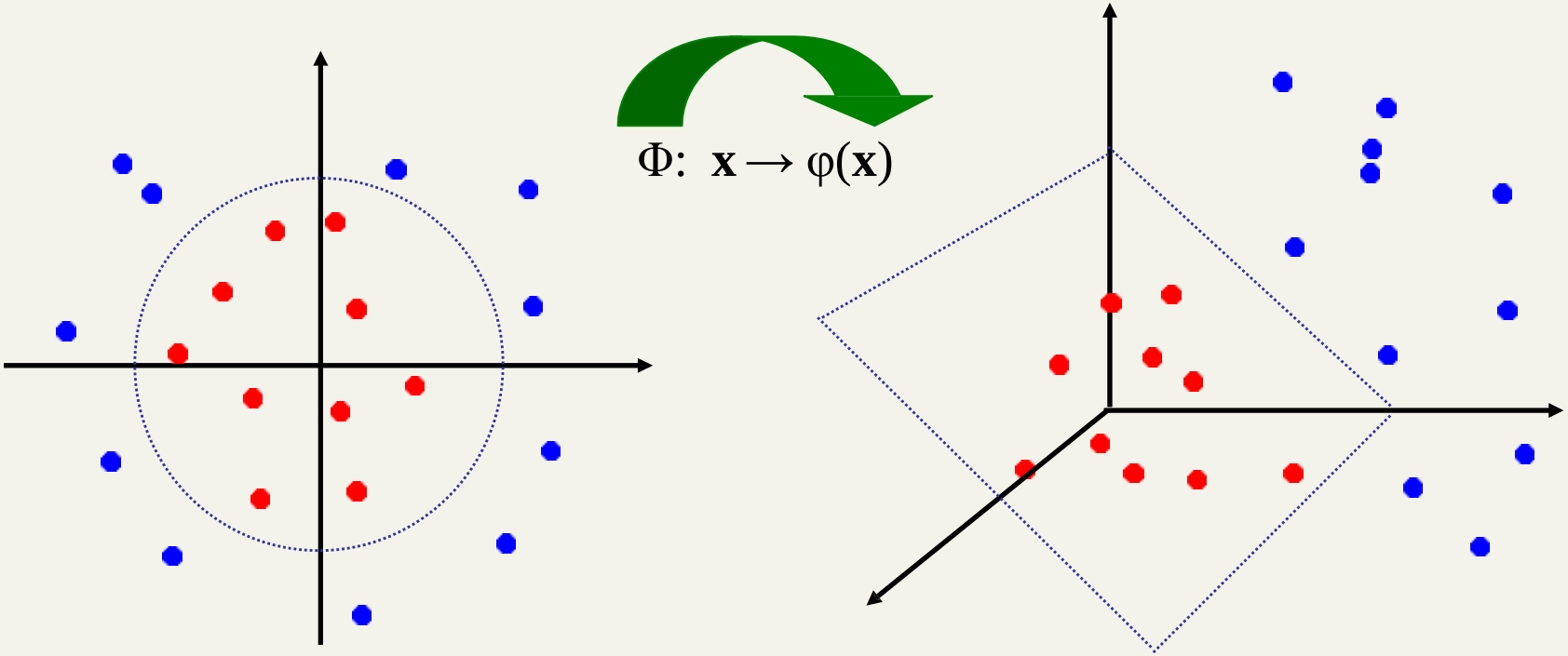


- How about … mapping data to a higher-dimensional space:



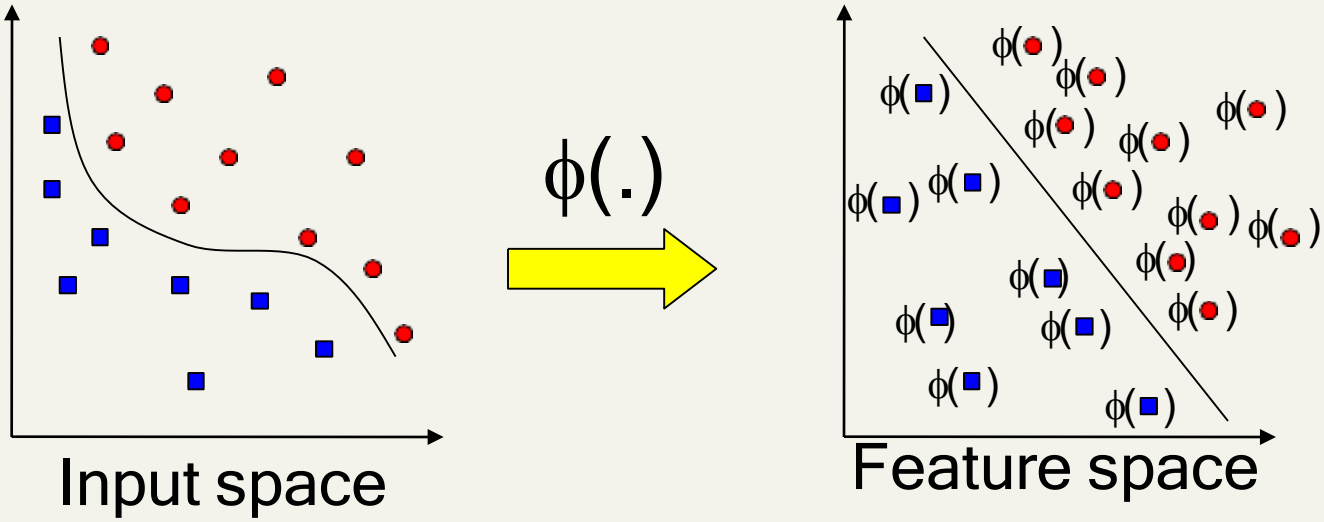
Non-linear SVMs: Feature spaces

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



Transformation to Feature Space

- "Kernel tricks"
  - Make non-separable problem separable.
  - Map data into better representational space



Modification Due to Kernel Function

- Change all inner products to kernel functions
- For training,

Original

$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

With kernel function

$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

$$K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$$

Example Transformation

- Consider the following transformation

$$\phi(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

$$\phi(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}) = (1, \sqrt{2}y_1, \sqrt{2}y_2, y_1^2, y_2^2, \sqrt{2}y_1y_2)$$

$$\begin{aligned} \langle \phi(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}), \phi(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}) \rangle &= (1 + x_1y_1 + x_2y_2)^2 \\ &= K(\mathbf{x}, \mathbf{y}) \end{aligned}$$

- Define the kernel function K (**x**,**y**) as

$$K(\mathbf{x}, \mathbf{y}) = (1 + x_1y_1 + x_2y_2)^2$$

- The inner product $\phi(.)\phi(.)$ can be computed by K without going through the map $\phi(.)$ explicitly!!!

Choosing a Kernel Function

- Active research on kernel function choices for different applications
- Examples:
  - Polynomial kernel with degree *d*

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

  - Radial basis function (RBF) kernel

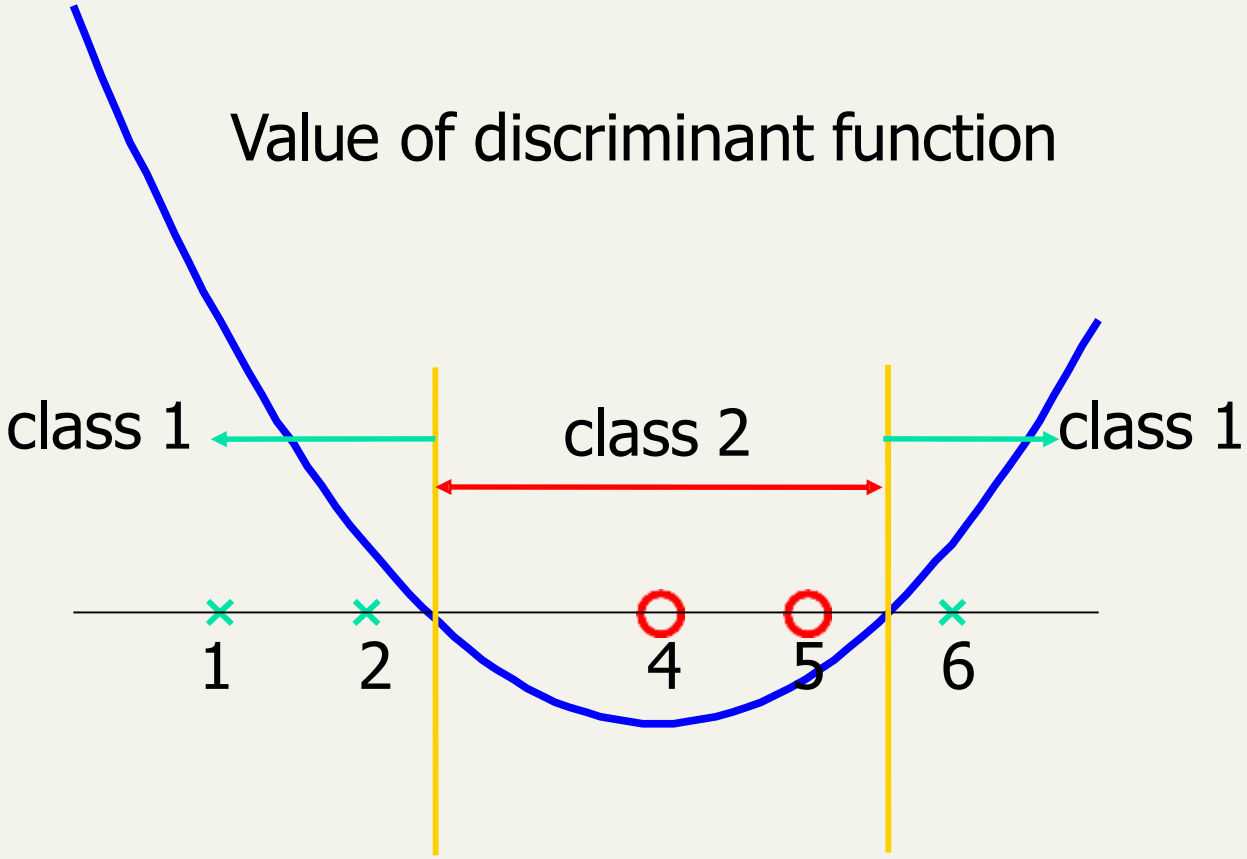
$$k(\mathbf{x_i}, \mathbf{x_j}) = \exp(-\gamma\|\mathbf{x_i} - \mathbf{x_j}\|^2)$$

  or sometime

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

    - Closely related to radial basis function neural networks
- In practice, a low degree polynomial kernel or RBF kernel is a good initial try

Example: 5 1D data points



Example

- 5 1D data points
  - x_1=1, x_2=2, x_3=4, x_4=5, x_5=6, with 1, 2, 6 as class 1 and 4, 5 as class 2 $\Rightarrow$ y_1=1, y_2=1, y_3=-1, y_4=-1, y_5=1
- We use the polynomial kernel of degree 2
  - $K(x,y) = (xy+1)^2$
  - C is set to 100
- We first find α_i (i=1, … , 5) by

$$\text{max.} \sum_{i=1}^{5} \alpha_i - \frac{1}{2} \sum_{i=1}^{5} \sum_{i=1}^{5} \alpha_i \alpha_j y_i y_j (x_i x_j + 1)^2$$

$$\text{subject to } 100 \geq \alpha_i \geq 0, \sum_{i=1}^{5} \alpha_i y_i = 0$$

Example

- By using a QP solver, we get

  α_1=0, α_2=2.5, α_3=0, α_4=7.333, α_5=4.833

  - Verify (at home) that the constraints are indeed satisfied
  - The support vectors are {x_2=2, x_4=5, x_5=6}
- The discriminant function is

$$f(y) = 2.5(1)(2y+1)^2 + 7.333(-1)(5y+1)^2 + 4.833(1)(6y+1)^2 + b$$
$$= 0.6667x^2 - 5.333x + b$$

$$y_i(\mathbf{w}^T\phi(z) + b) = 1$$

- *b* is recovered by solving f(2)=1 or by f(5)=-1 or by f(6)=1, as x_2, x_4, x_5 lie on f(y) and all give b=9 with

$$f(y) = 0.6667x^2 - 5.333x + 9$$

Software

- A list of SVM implementation can be found at http://www.kernel-machines.org/software.html
- Some implementation (such as LIBSVM) can handle multi-class classification
- SVMLight is among one of the earliest implementation of SVM
- Several Matlab toolboxes for SVM are also available

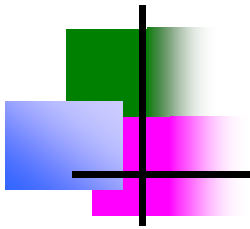
Kernel Trick in Support Vector Machine

- For example, one mapping function $\emptyset: R^2 \rightarrow R^3$ used to transform a 2D data to 3D data is given as follows:

$$\emptyset(x, y) = (x^2, \sqrt{2}xy, y^2)$$

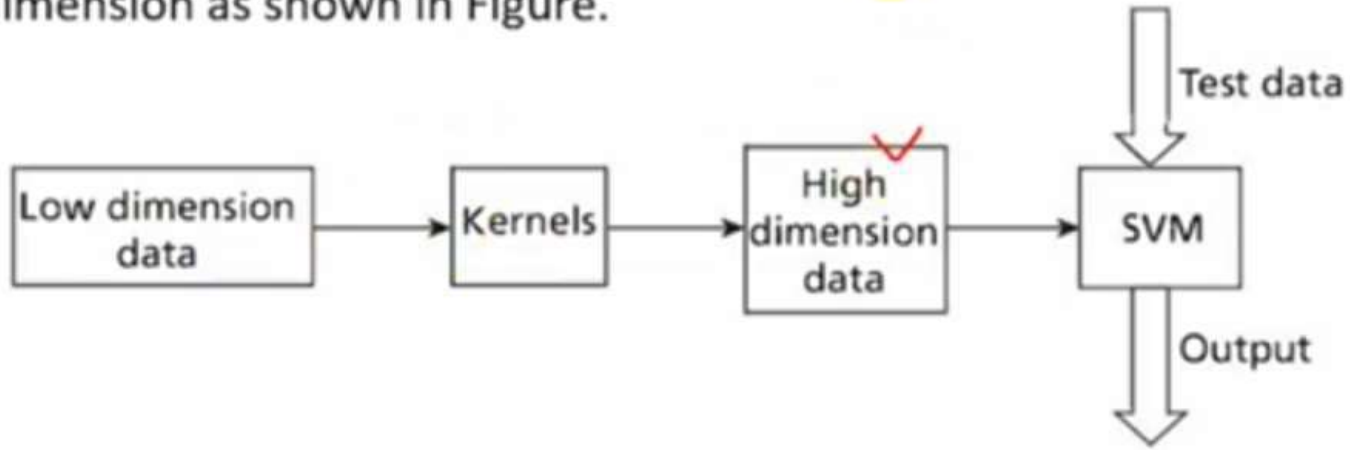
- Consider a point (2, 3) in 2D space, if you apply above mapping function we can convert it into 3D space and it looks like this,
- Here $x = 2\ and\ y = 3$,
- Hence point in 3D space is:

$$(2^2, \sqrt{2} * 2 * 3, 3^2) = (4, 6\sqrt{2}, 9)$$



- While mapping functions play an important role, there are many disadvantages, as mapping involves more computations and learning costs.
- Also, the disadvantages of transformations are that there is no generalized thumb rule available describing what transformations should be applied and if the data is large, mapping process takes huge amount of time.
- In real applications, there might be many features in the data and applying transformations that involve many polynomial combinations of these features will lead to extremely high and impractical computational costs.
- In this context, only kernels are useful.
- Kernels are used to compute the value without transforming the data.

- **What is a Kernel?**
- Kernels are a set of functions used to transform data from lower dimension to higher dimension and to manipulate data using dot product at higher dimensions.
- The use of kernels is to apply transformation to data and perform classification at the higher dimension as shown in Figure.



- **Kernel Trick for 2nd degree Polynomial Mapping** $\phi(x, y) = (x^2, \sqrt{2}xy, y^2)$

$$\phi(\mathbf{a})^T \cdot \phi(\mathbf{b}) = \begin{pmatrix} a_1^2 \\ \sqrt{2}\, a_1 a_2 \\ a_2^2 \end{pmatrix}^T \cdot \begin{pmatrix} b_1^2 \\ \sqrt{2}\, b_1 b_2 \\ b_2^2 \end{pmatrix} = a_1^2 b_1^2 + 2a_1 b_1 a_2 b_2 + a_2^2 b_2^2$$

$$= (a_1 b_1 + a_2 b_2)^2 = \left(\begin{pmatrix} a_1 \\ a_2 \end{pmatrix}^T \cdot \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \right)^2 = \left(\mathbf{a}^T \cdot \mathbf{b} \right)^2$$

Evaluation: Reuters News Data Set

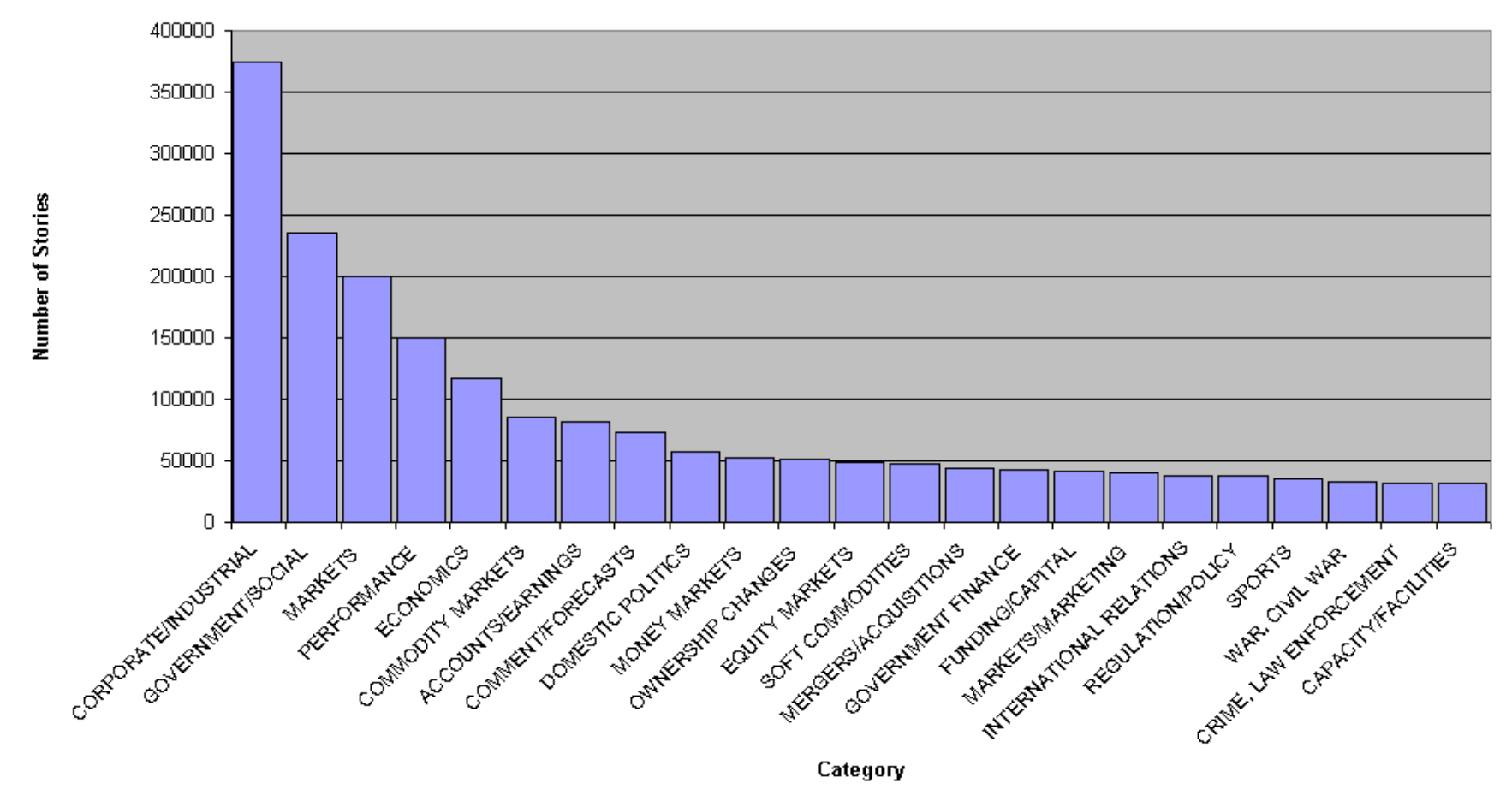
- Most (over)used data set
- 21578 documents
- 9603 training, 3299 test articles (ModApte split)
- 118 categories
  - An article can be in more than one category
  - Learn 118 binary category distinctions
- Average document: about 90 types, 200 tokens
- Average number of classes assigned
  - 1.24 for docs with at least one category
- Only about 10 out of 118 categories are large

Common categories
(#train, #test)

- Earn (2877, 1087)
- Acquisitions (1650, 179)
- Money-fx (538, 179)
- Grain (433, 149)
- Crude (389, 189)
- Trade (369,119)
- Interest (347, 131)
- Ship (197, 89)
- Wheat (212, 71)
- Corn (182, 56)

New Reuters: RCV1: 810,000 docs

- Top topics in Reuters RCV1



Dumais et al. 1998: Reuters - Accuracy

	Rocchio	NBayes	Trees	LinearSVM	
earn	92.9%	95.9%	97.8%	98.2%	
acq	64.7%	87.8%	89.7%	92.8%	
money-fx	46.7%	56.6%	66.2%	74.0%	
grain	67.5%	78.8%	85.0%	92.4%	
crude	70.1%	79.5%	85.0%	88.3%	
trade	65.1%	63.9%	72.5%	73.5%	
interest	63.4%	64.9%	67.1%	76.3%	
ship	49.2%	85.4%	74.2%	78.0%	
wheat	68.9%	69.7%	92.5%	89.7%	
corn	48.2%	65.3%	91.8%	91.1%	
Avg Top 10	64.6%	81.5%	88.4%	91.4%	
Avg All Cat	61.7%	75.2%	na	86.4%	

Recall: % labeled in category among those stories that are really in category

Precision: % really in category among those stories labeled in category

Break Even: (Recall + Precision) / 2

Results for Kernels (Joachims 1998)

					SVM (poly) degree d =					SVM (rbf) width γ =			
	Bayes	Rocchio	C4.5	k-NN	1	2	3	4	5	0.6	0.8	1.0	1.2
earn	95.9	96.1	96.1	97.3	98.2	98.4	**98.5**	98.4	98.3	**98.5**	98.5	98.4	98.3
acq	91.5	92.1	85.3	92.0	92.6	94.6	**95.2**	95.2	95.3	95.0	95.3	95.3	**95.4**
money-fx	62.9	67.6	69.4	78.2	66.9	72.5	75.4	74.9	**76.2**	74.0	75.4	**76.3**	75.9
grain	72.5	79.5	89.1	82.2	91.3	93.1	**92.4**	91.3	89.9	**93.1**	91.9	91.9	90.6
crude	81.0	81.5	75.5	85.7	86.0	87.3	88.6	**88.9**	87.8	**88.9**	89.0	88.9	88.2
trade	50.0	77.4	59.2	77.4	69.2	75.5	76.6	77.3	**77.1**	76.9	78.0	**77.8**	76.8
interest	58.0	72.5	49.1	74.0	69.8	63.3	67.9	73.1	**76.2**	74.4	75.0	**76.2**	76.1
ship	78.7	83.1	80.9	79.2	82.0	85.4	86.0	**86.5**	86.0	**85.4**	86.5	87.6	87.1
wheat	60.6	79.4	85.5	76.6	83.1	84.5	85.2	**85.9**	83.8	**85.2**	85.9	85.9	85.9
corn	47.3	62.2	87.7	77.9	86.0	86.5	85.3	**85.7**	83.9	**85.1**	85.7	85.7	84.5
microavg.	**72.0**	**79.9**	**79.4**	**82.3**	84.2	85.1	85.9	86.2	85.9	86.4	86.5	86.3	86.2
					combined: **86.0**					combined: **86.4**			

Micro- vs. Macro-Averaging

- If we have more than one class, how do we combine multiple performance measures into one quantity?
- Macroaveraging: Compute performance for each class, then average.
- Microaveraging: Collect decisions for all classes, compute contingency table, evaluate.

Micro- vs. Macro-Averaging: Example

Class 1

	Truth: yes	Truth: no
Classifier: yes	10	10
Classifier: no	10	970

Class 2

	Truth: yes	Truth: no
Classifier: yes	90	10
Classifier: no	10	890

Micro.Av. Table

	Truth: yes	Truth: no
Classifier: yes	100	20
Classifier: no	20	1860

- Macroaveraged precision: (0.5 + 0.9)/2 = 0.7
- Microaveraged precision: 100/120 = .83
- Why this difference?

The Real World

- Gee, I'm building a text classifier for real, now!
- What should I do?

- How much training data do you have?
  - None
  - Very little
  - Quite a lot
  - A huge amount and its growing

Manually written rules

- No training data, adequate editorial staff?
- Never forget the hand-written rules solution!
  - If (wheat or grain) then categorize as grain
- In practice, rules get a lot bigger than this
  - Can also be phrased using tf or tf.idf weights
- With careful crafting (human tuning on development data) performance is high:
  - 94% recall, 84% precision over 675 categories (Hayes and Weinstein 1990)
- Amount of work required is huge
  - Estimate 2 days per class … plus maintenance

Very little data?

- If you're just doing supervised classification, you should stick to something high bias
  - There are theoretical results that Naïve Bayes should do well in such circumstances (Ng and Jordan 2002 NIPS)
- The interesting theoretical answer is to explore semi-supervised training methods:
  - Bootstrapping, EM over unlabeled documents, …
- The practical answer is to get more labeled data as soon as you can
  - How can you insert yourself into a process where humans will be willing to label data for you??

A reasonable amount of data?

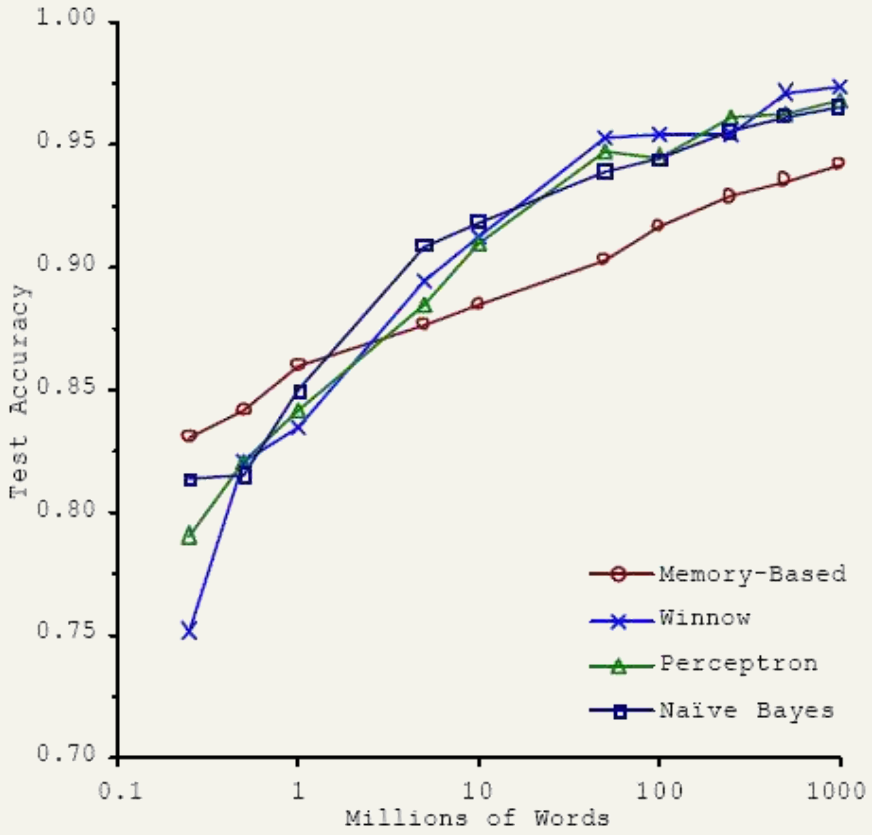
- Good with SVM
- But if you are using an SVM/NB etc., you should probably be prepared with the “hybrid” solution where there is a boolean overlay
  - Or else to use user-interpretable Boolean-like models like decision trees
  - Users like to hack, and management likes to be able to implement quick fixes immediately

A huge amount of data?

- This is great in theory for doing accurate classification…
- But it could easily mean that expensive methods like SVMs (train time) or kNN (test time) are quite impractical
- Naïve Bayes can come back into its own again!
  - Or other advanced methods with linear training/test complexity like regularized logistic regression (though much more expensive to train)

A huge amount of data?

- With enough data the choice of classifier may not matter much, and the best choice may be unclear
  - Learning curve experiment: Brill and Banko on context-sensitive spelling correction



How many categories?

- A few (well separated ones)?
  - Easy!
- A zillion closely related ones?
  - Think: Yahoo! Directory, Library of Congress classification, legal applications
  - Quickly gets difficult!
    - Classifier combination is always a useful technique
      - Voting, bagging, or boosting multiple classifiers
    - Much literature on hierarchical classification
      - Mileage fairly unclear
    - May need a hybrid automatic/manual solution

Can data “hacking”/debugging work?

- Yes!
- Aim to exploit any domain-specific useful features that give special meanings
- Aim to collapse things that would be treated as different but shouldn’t be.
  - E.g., part numbers, chemical formulas

Text Summarization techniques in text classification

- Text Summarization: Process of extracting key pieces from text, normally by features on sentences reflecting position and content
- Much of this work can be used to suggest weightings for terms in text categorization
  - See: Kolcz, Prabakarmurthi, and Kolita, CIKM 2001: Summarization as feature selection for text categorization
  - Categorizing purely with title,
  - Categorizing with first paragraph only
  - Categorizing with paragraph with most keywords
  - Categorizing with first and last paragraphs, etc.

Does data hacking/debugging help?

- Yes!
- Application: Document summary (snippet)
- Weighting contributions from different document zones:
  - Upweighting title words helps (Cohen & Singer 1996)
    - Doubling the weighting on the title words is a good rule of thumb
  - Upweighting the first sentence of each paragraph helps (Murata, 1999)
  - Upweighting sentences that contain title words helps (Ko *et al,* 2002)

Does stemming/lowercasing/… help?

- As always it's hard to tell
- The role of tools like stemming is slightly different for TextCat vs. IR:
  - For IR, you may want to collapse forms of the credit card/credit cards, since all of those documents will be relevant to a query for *credit card*
    - *Error happens when doing aggressively.*
    - *Avoid when there is enough data.*
  - For TextCat, with sufficient training data, stemming *does no good.* It only helps in compensating for data sparseness (which can be severe in TextCat applications). *Overly aggressive stemming can easily degrade performance.*

Measuring Classification Figures of Merit

- Not just accuracy; in the real world, there are economic measures:
  - Your choices are:
    - Do no classification
    - Do it manually
    - Do it all with an automatic classifier
      - Mistakes have a cost
    - Do it with a combination of automatic classification and manual review of uncertain/difficult/“new” cases
  - Commonly the last method is most cost efficient and is adopted

A common problem: Concept Drift

- Categories change over time
- Example: “president of the united states”
  - 1999: clinton is great feature
  - 2002: clinton is bad feature
- One measure of a text classification system is how well it protects against concept drift.
  - Can favor simpler models like Naïve Bayes
- Feature selection: can be bad in protecting against concept drift

Summary

- Support vector machines (SVM)
  - Choose hyperplane based on support vectors
    - Support vector = “critical” point close to decision boundary
  - (Degree-1) SVMs are linear classifiers.
  - Kernels: powerful and elegant way to define similarity metric
  - Perhaps best performing text classifier
    - But there are other methods that perform about as well as SVM, such as regularized logistic regression (Zhang & Oles 2001)
  - Partly popular due to availability of SVMlight
    - SVMlight is accurate and fast – and free (for research)
  - Now lots of software: libsvm, TinySVM, … .
- Comparative evaluation of methods
- Real world: exploit domain specific structure!

Resources

- A Tutorial on Support Vector Machines for Pattern Recognition (1998) Christopher J. C. Burges
- S. T. Dumais, Using SVMs for text categorization, IEEE Intelligent Systems, 13(4), Jul/Aug 1998
- S. T. Dumais, J. Platt, D. Heckerman and M. Sahami. 1998. Inductive learning algorithms and representations for text categorization. *CIKM '98*, pp. 148-155.
- A re-examination of text categorization methods (1999) Yiming Yang, Xin Liu 22nd Annual International SIGIR
- Tong Zhang, Frank J. Oles: Text Categorization Based on Regularized Linear Classification Methods. Information Retrieval 4(1): 5-31 (2001)
- Trevor Hastie, Robert Tibshirani and Jerome Friedman, "Elements of Statistical Learning: Data Mining, Inference and Prediction" Springer-Verlag, New York.
- 'Classic' Reuters data set: http://www.daviddlewis.com /resources /testcollections/reuters21578/
- T. Joachims, *Learning to Classify Text using Support Vector Machines*. Kluwer, 2002.
- Fan Li, Yiming Yang: A Loss Function Analysis for Classification Methods in Text Categorization. ICML 2003: 472-479.

Thank you for your attention