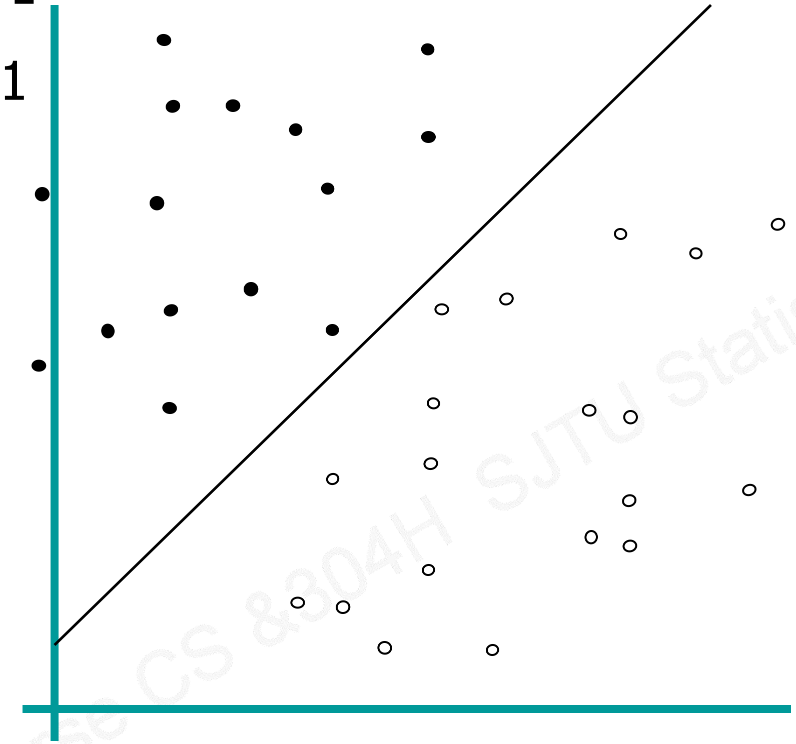
Introduction to SVMs

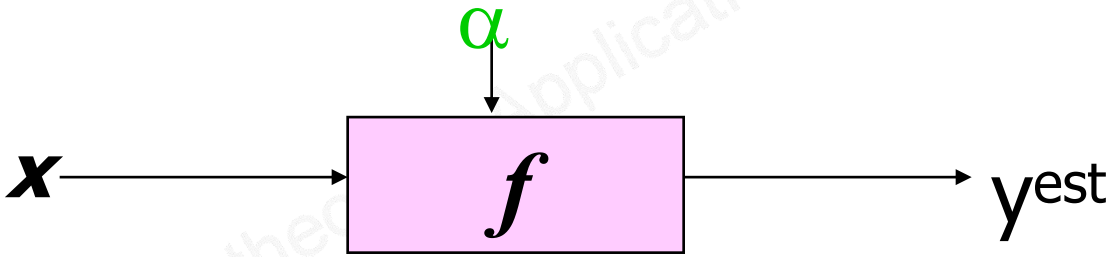
SVMs

- Geometric
  - Maximizing Margin
- Kernel Methods
  - Making nonlinear decision boundaries linear
  - Efficiently!
- Capacity
  - Structural Risk Minimization

Linear Classifiers

- • denotes +1
- ◦ denotes -1

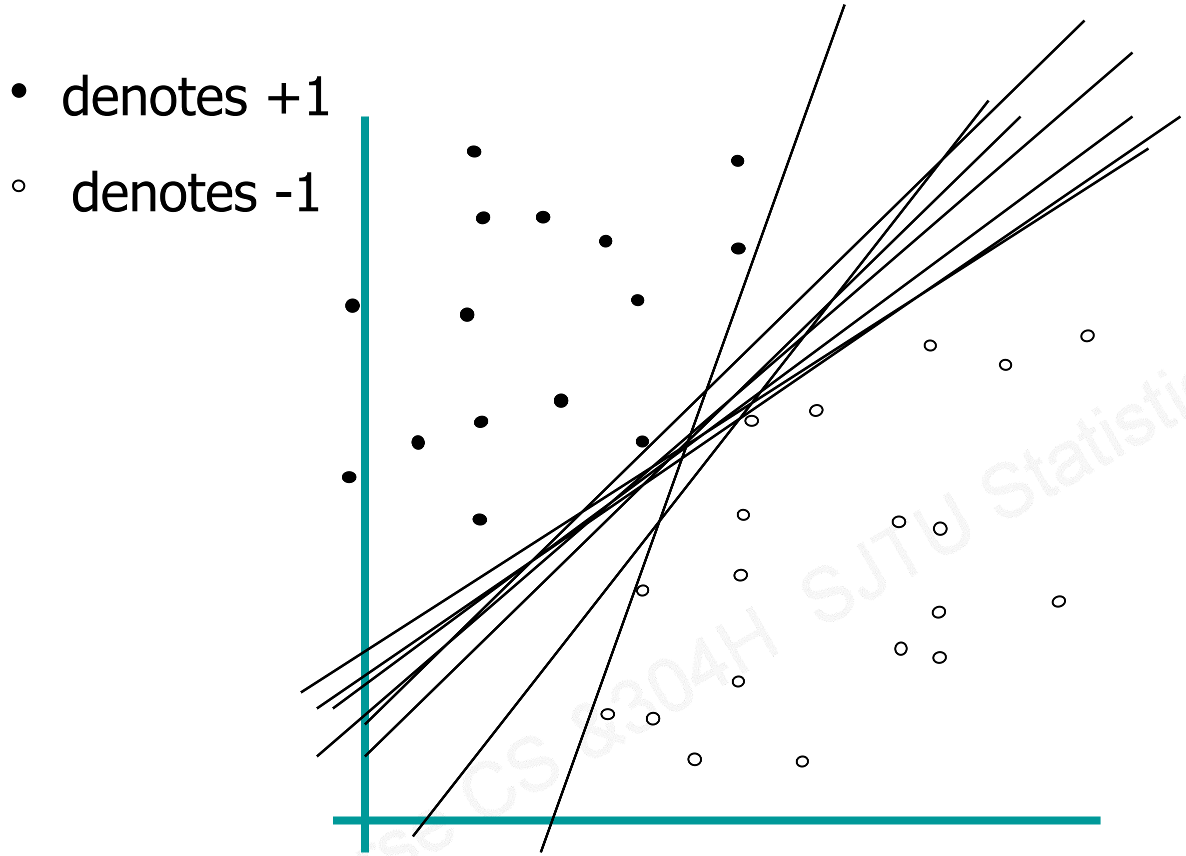


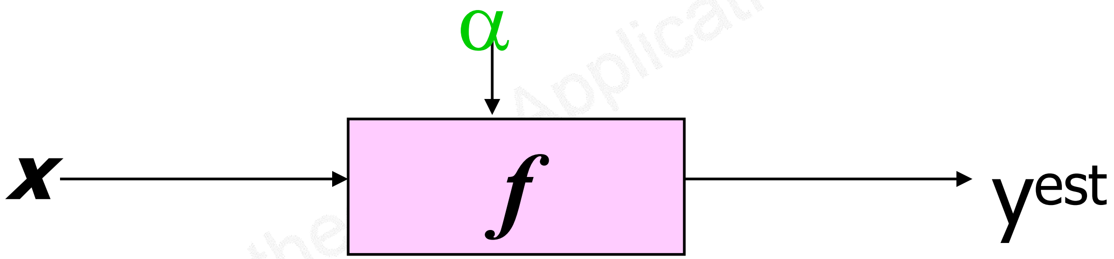


$$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} - b)$$

How would you classify this data?

Linear Classifiers



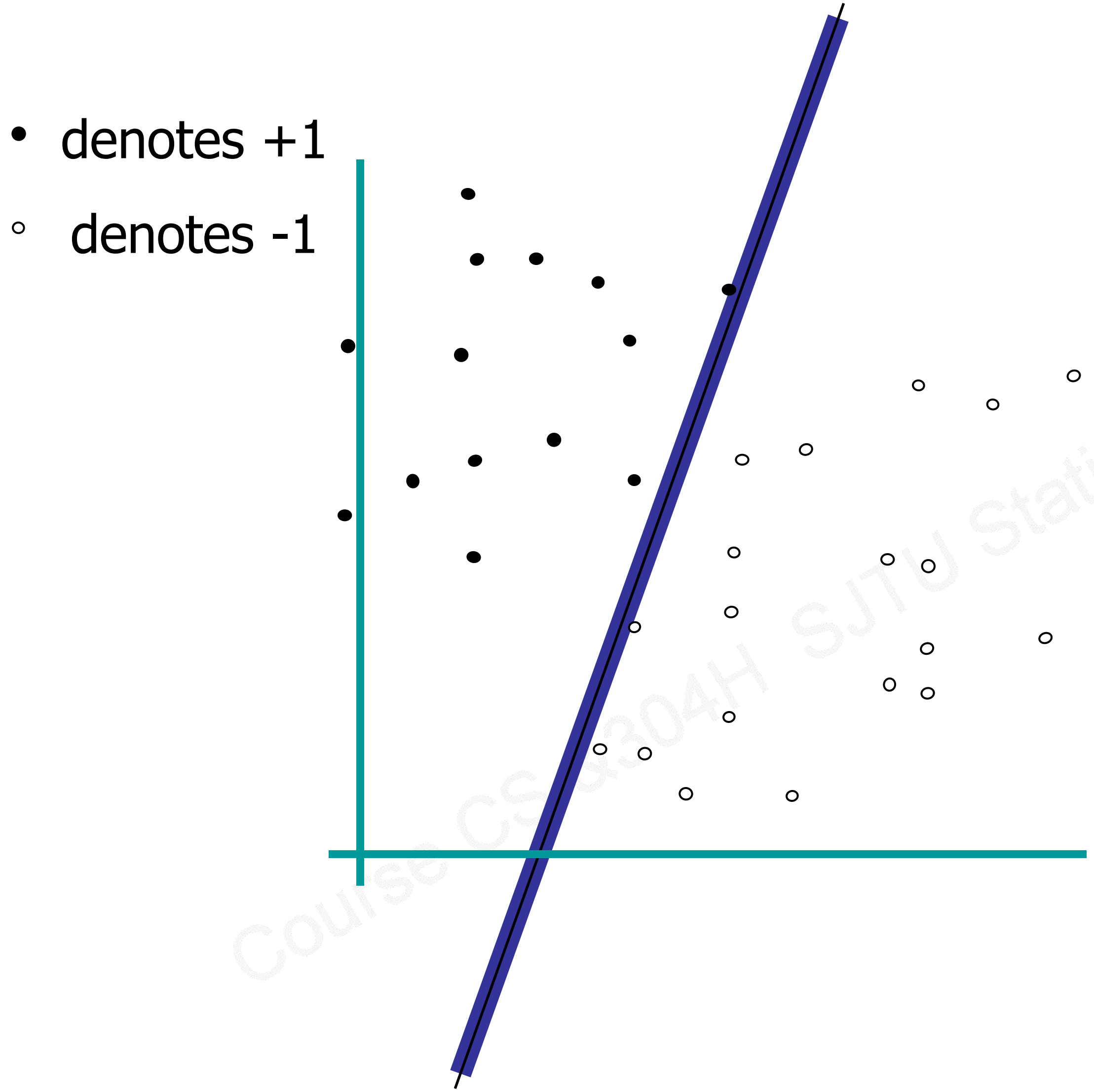


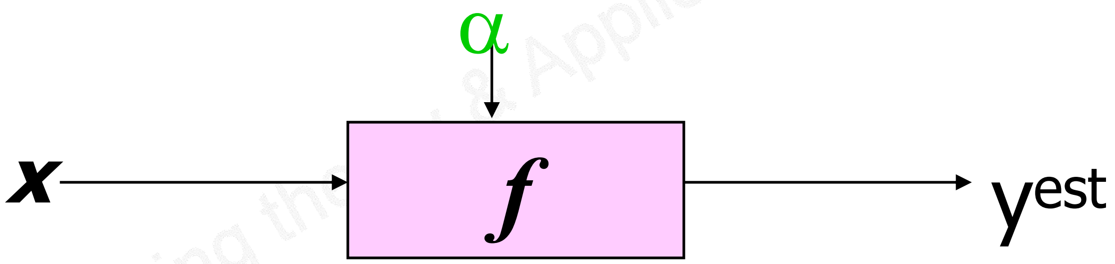
$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} - b)$

Any of these would be fine..

..but which is best?

Classifier Margin

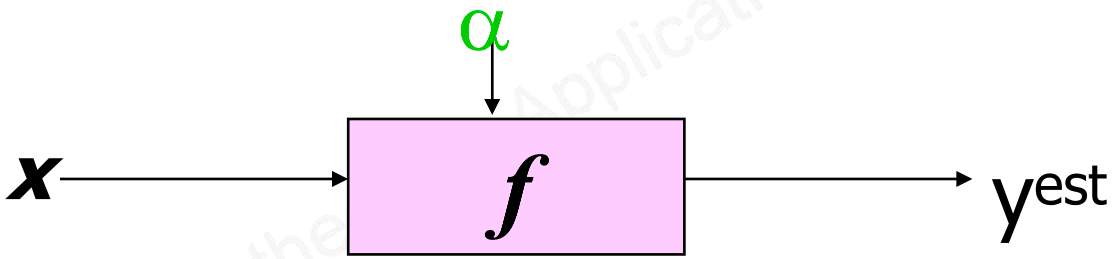




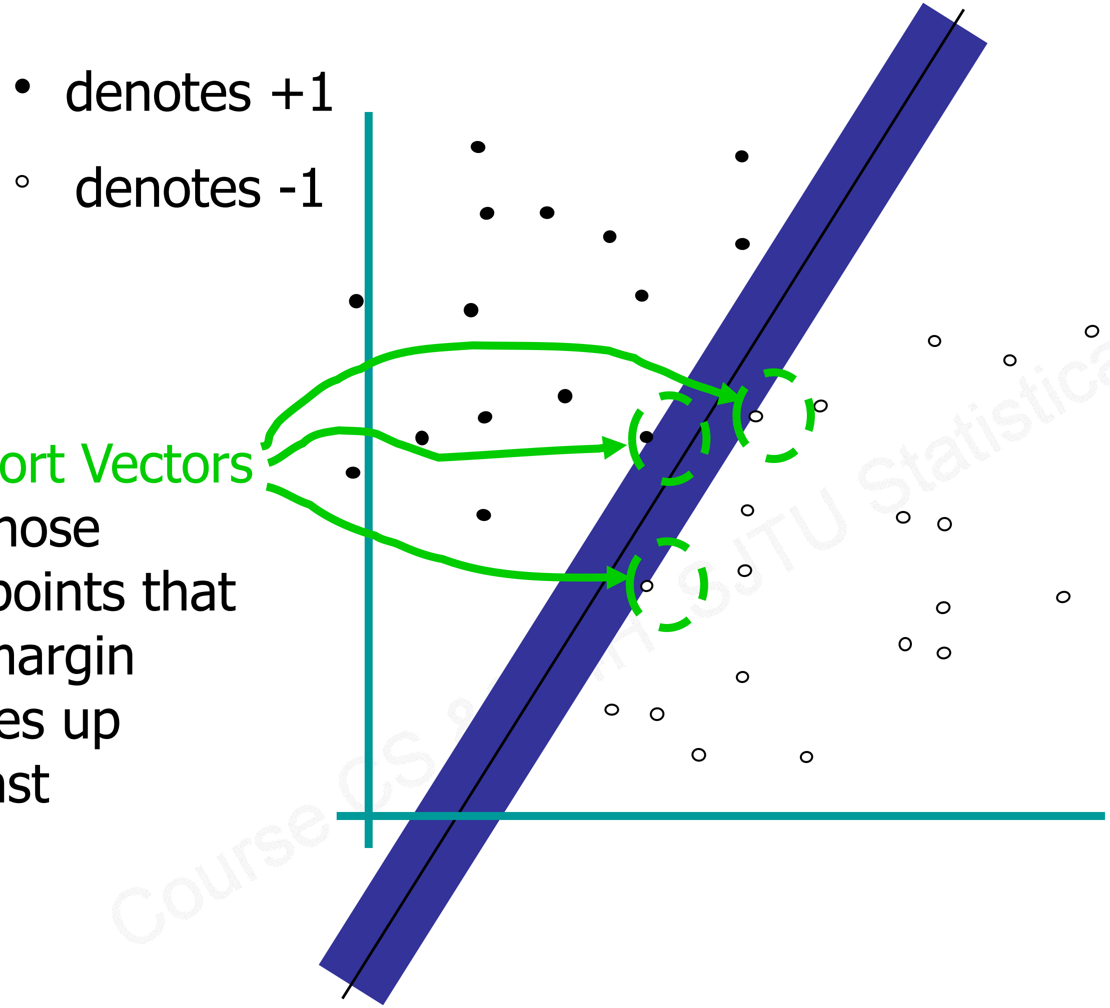
f(**x**,**w**,*b*) = *sign*(**w. x** - *b*)

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

Maximum Margin



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



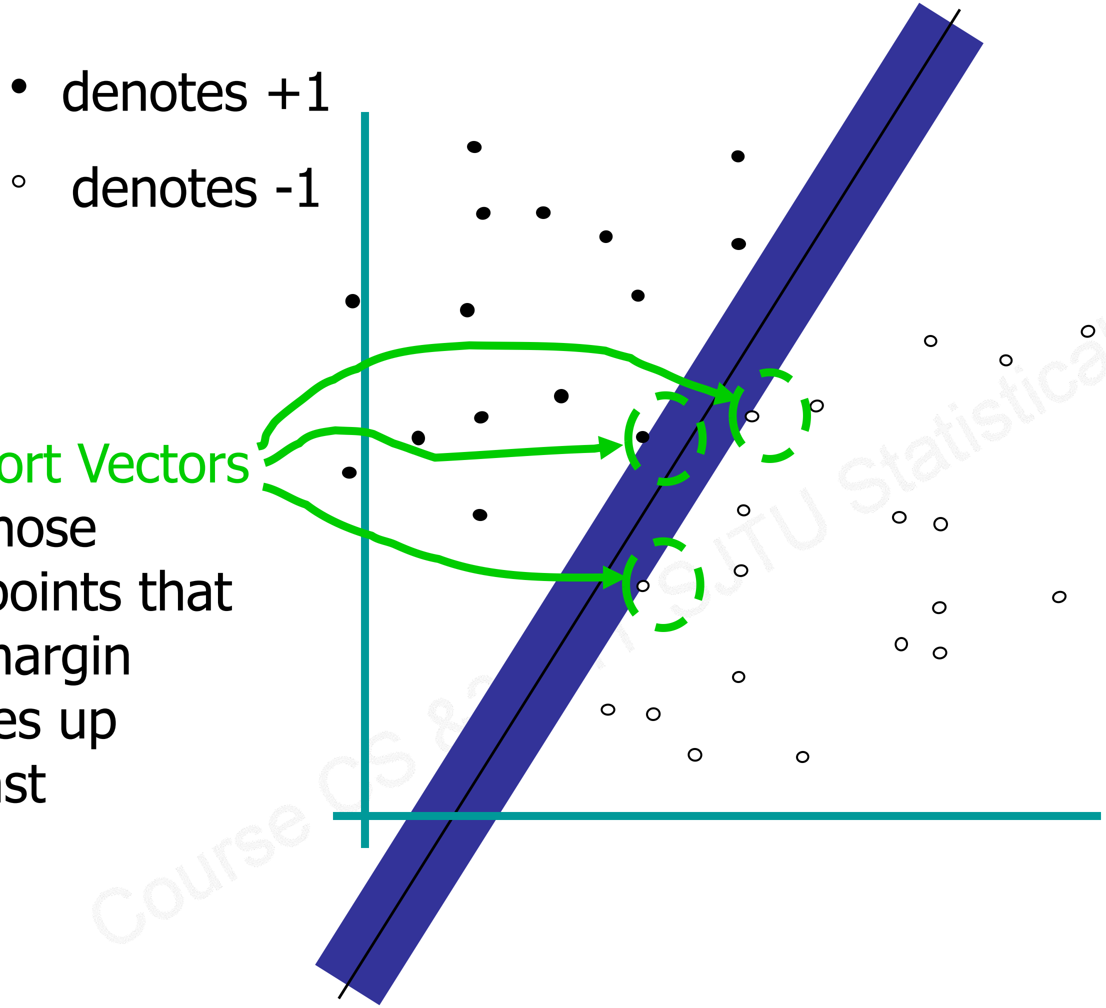
Support Vectors are those datapoints that the margin pushes up against

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

Why Maximum Margin?

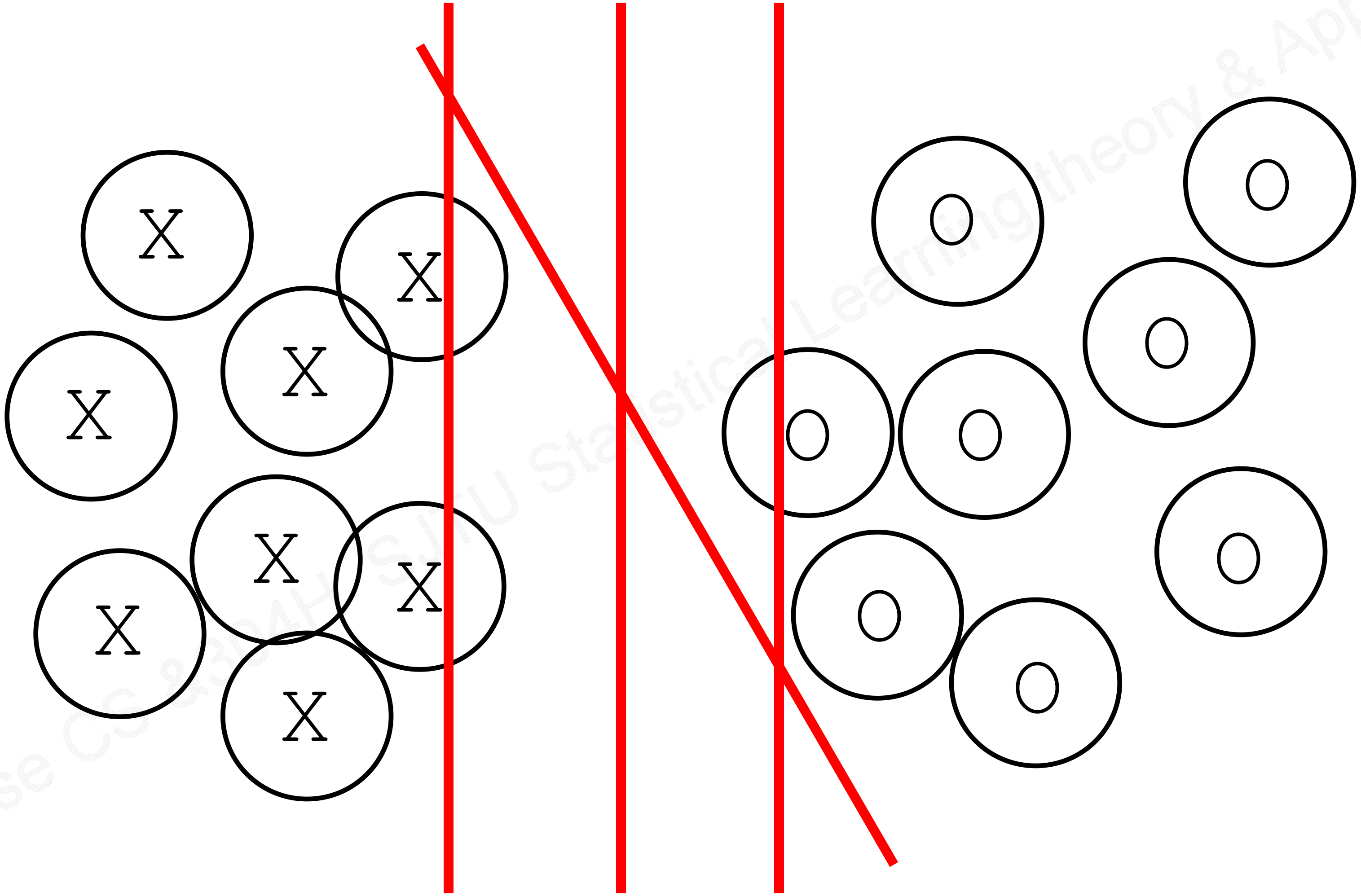


Support Vectors are those datapoints that the margin pushes up against

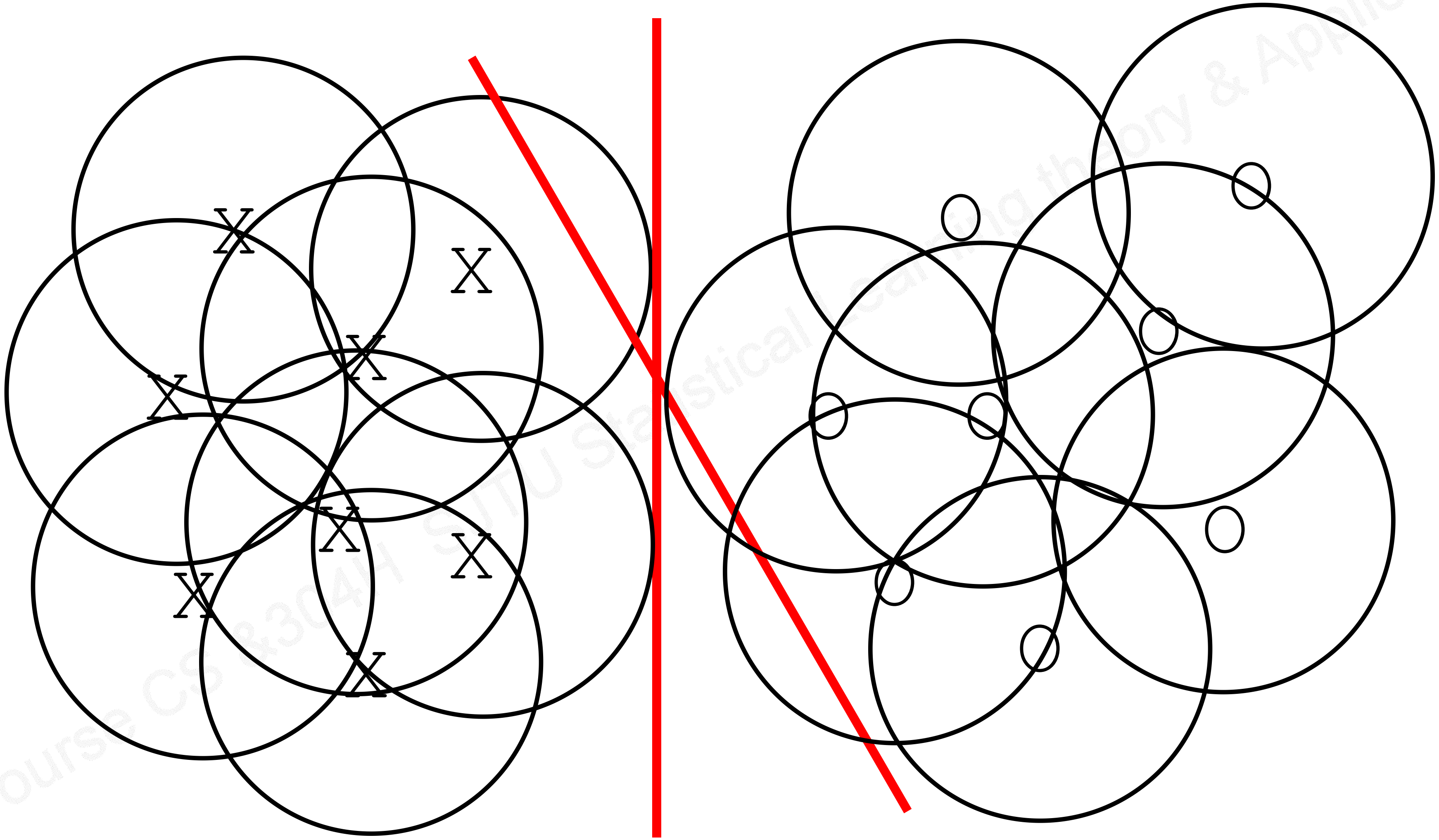
$f(\mathbf{x},\mathbf{w},b) = \mathrm{sign}(\mathbf{w}. \mathbf{x} - b)$

1. Intuitively this feels safest.
2. 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.
3. There's some theory (using VC dimension) that is related to (but not the same as) the proposition that this is a good thing.
4. Empirically it works very very well.

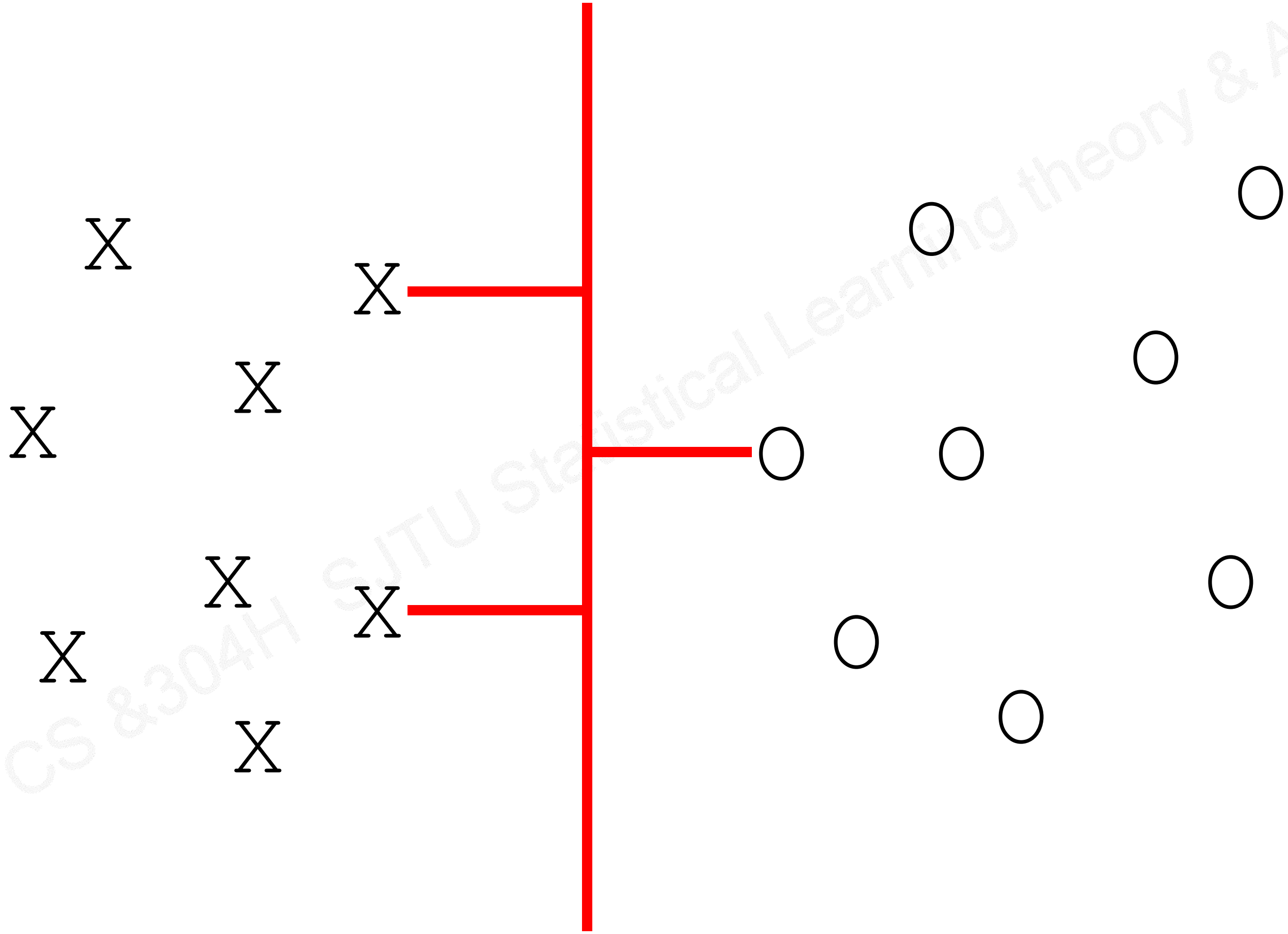
Ruling Out Some Separators



Lots of Noise

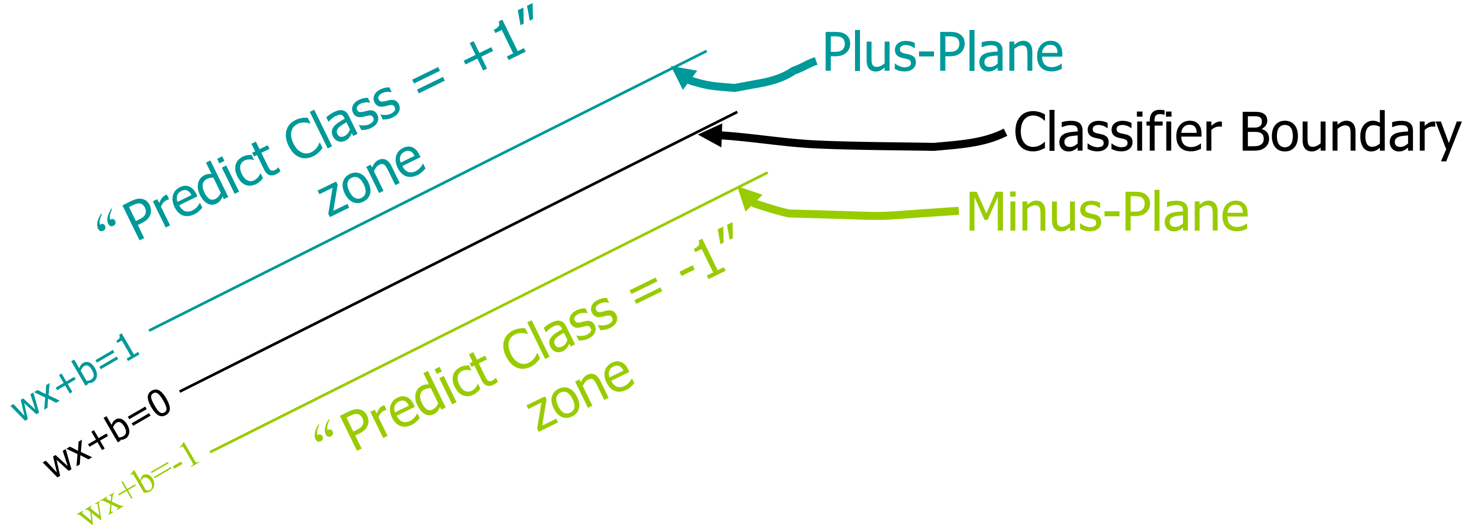


Maximizing the Margin



Specifying a line and margin

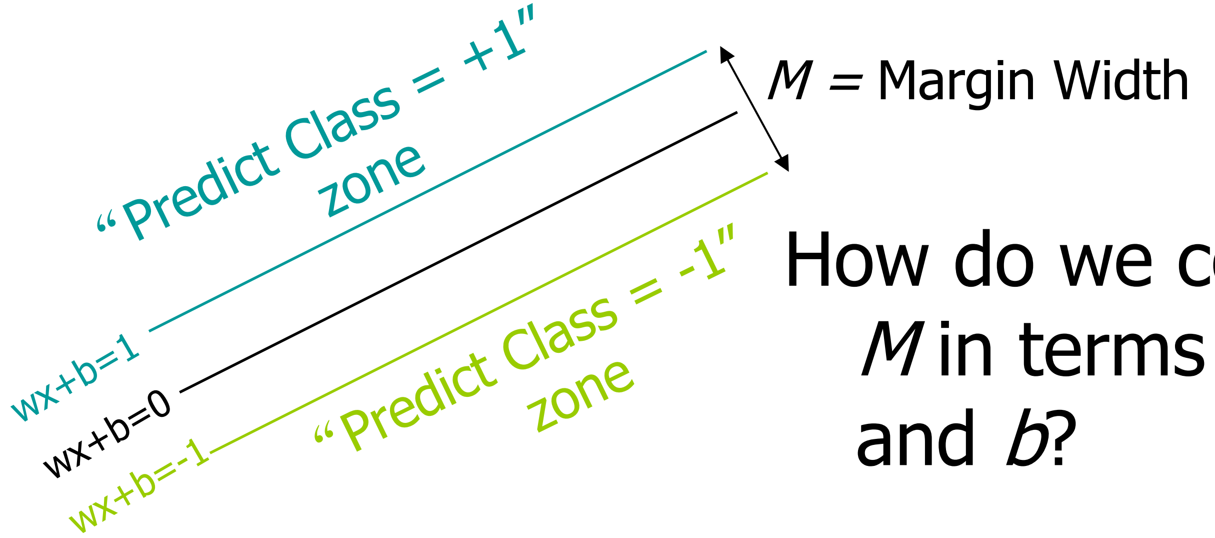
Classify as..



- Plus-plane = $\{ \mathbf{x} : \mathbf{w} \cdot \mathbf{x} + b = +1 \}$
- Minus-plane = $\{ \mathbf{x} : \mathbf{w} \cdot \mathbf{x} + b = -1 \}$

+1	if	$\mathbf{w} \cdot \mathbf{x} + b >= 1$
-1	if	$\mathbf{w} \cdot \mathbf{x} + b <= -1$
uncertain	if	$-1 < \mathbf{w} \cdot \mathbf{x} + b < 1$

Computing the margin width



How do we compute M in terms of $\mathbf{w}$ and b?

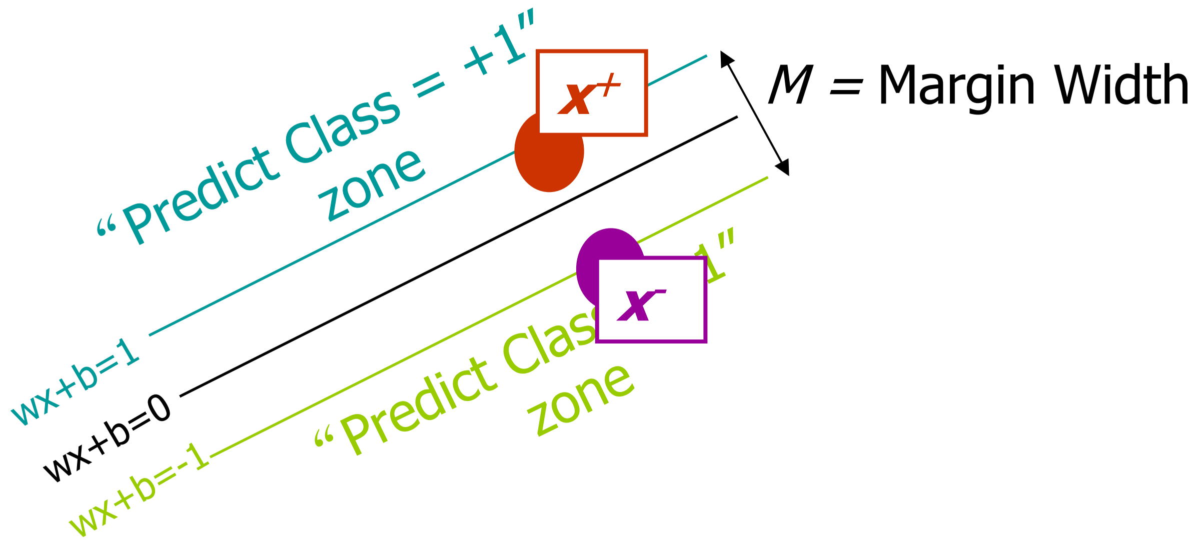
- Plus-plane = $\{ \mathbf{x} : \mathbf{w} \,.\, \mathbf{x} + b = +1 \}$
- Minus-plane = $\{ \mathbf{x} : \mathbf{w} \,.\, \mathbf{x} + b = -1 \}$

Claim: The vector **w** is perpendicular to the Plus Plane. Why?

And so of course the vector **w** is also perpendicular to the Minus Plane

Let **u** and **v** be two vectors on the Plus Plane. What is $\mathbf{w} \,.\, (\mathbf{u} - \mathbf{v})$?

Computing the margin width



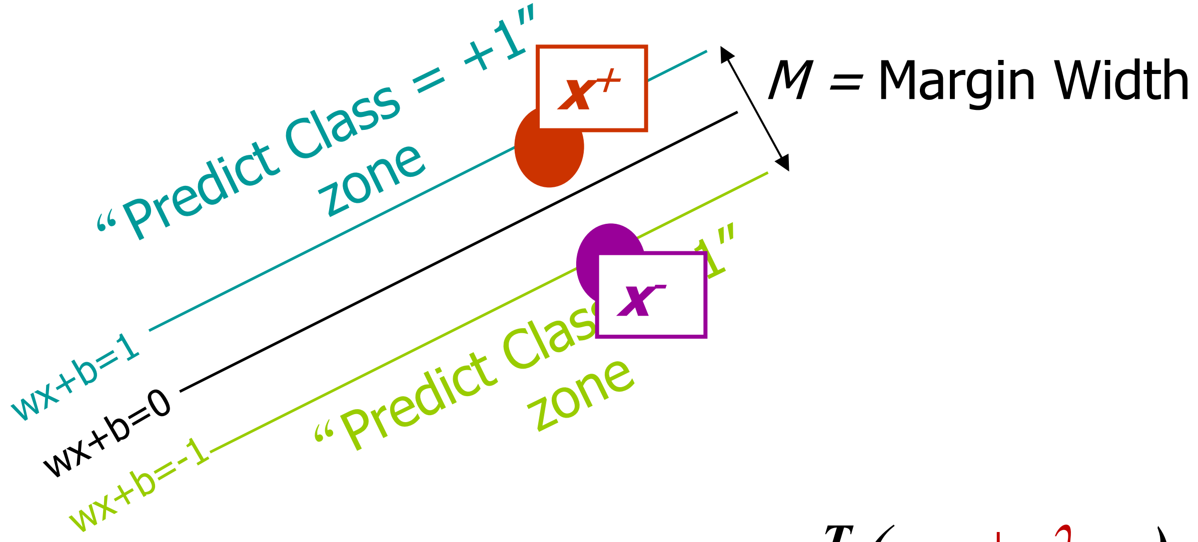
Plus-plane $=$ $\{\, \mathbf{x} : \mathbf{w} \,.\, \mathbf{x} + b = +1 \,\}$

Minus-plane $=$ $\{\, \mathbf{x} : \mathbf{w} \,.\, \mathbf{x} + b = -1 \,\}$

- $\boldsymbol{x}^{+} = \boldsymbol{x}^{-} + \lambda\, \boldsymbol{w}$ $\qquad\qquad |\boldsymbol{x}^{+} - \boldsymbol{x}^{-}| = M$

It’s now easy to get M in terms of $\boldsymbol{w}$ and b

Computing the margin width



$$\mathbf{w}^T(\mathbf{x}^- + \lambda \mathbf{w}) + b = 1$$

$$\lambda = \frac{2}{\mathbf{w}^T\mathbf{w}}$$

$$\mathbf{w}^T\mathbf{x}^- + b + \lambda \mathbf{w}^T\mathbf{w} = 1$$

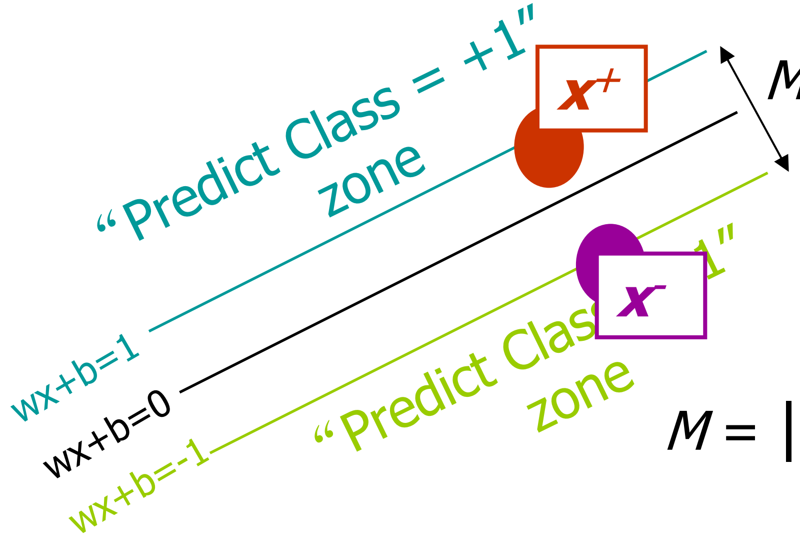
$$\rightarrow \quad -1 + \lambda \mathbf{w}^T\mathbf{w} = 1$$

What we know:

- $\mathbf{w}^T \mathbf{x}^+ + b = +1$
- $\mathbf{w}^T \mathbf{x}^- + b = -1$
- $\mathbf{x}^+ = \mathbf{x}^- + \lambda \mathbf{w}$
- $|\mathbf{x}^+ - \mathbf{x}^-| = M$

It’s now easy to get M in terms of $\mathbf{w}$ and b

Computing the margin width



$$M = \text{Margin Width} = \frac{2}{\sqrt{\mathbf{w}^T\mathbf{w}}}$$

What we know:

- $\boldsymbol{w} \,.\, \boldsymbol{x}^+ + b = +1$
- $\boldsymbol{w} \,.\, \boldsymbol{x}^- + b = -1$
- $\boldsymbol{x}^+ = \boldsymbol{x}^- + \lambda \boldsymbol{w}$
- $|\boldsymbol{x}^+ - \boldsymbol{x}^-| = M$

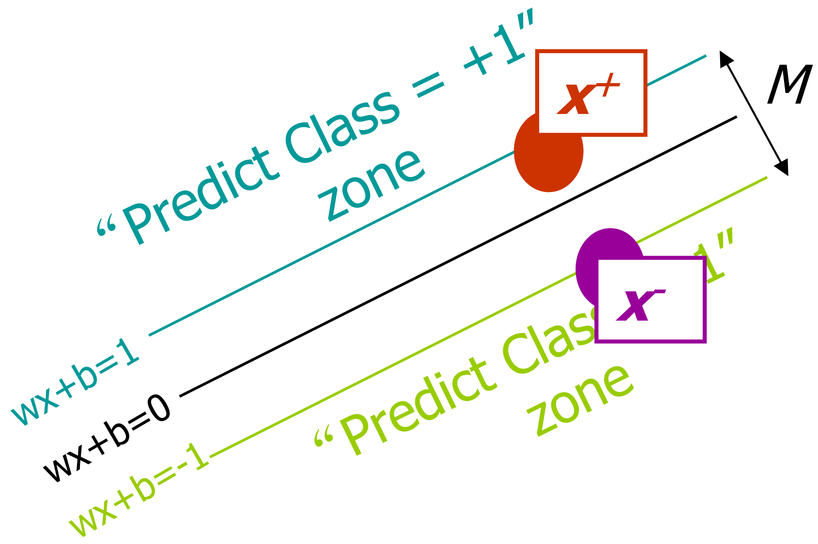
$$\lambda = \frac{2}{\mathbf{w}^T\mathbf{w}}$$

$$M = |\boldsymbol{x}^+ - \boldsymbol{x}^-| = |\lambda \boldsymbol{w}| =$$

$$= \lambda |\mathbf{w}| = \lambda \sqrt{\mathbf{w}^T\mathbf{w}}$$

$$= \frac{2\sqrt{\mathbf{w}^T\mathbf{w}}}{\mathbf{w}^T\mathbf{w}} = \frac{2}{\sqrt{\mathbf{w}^T\mathbf{w}}}$$

Learning the Maximum Margin Classifier



M = Margin Width = $\dfrac{2}{\sqrt{\mathbf{w}.\mathbf{w}}}$

Given a guess of ***w*** and *b* we can

- Compute whether all data points in the correct half-planes
- Compute the width of the margin
- to write a program to search the space of **w**′s and *b*′s to find the widest margin that matches all the datapoints. *How?*

Gradient descent? Simulated Annealing? Matrix Inversion? EM? Newton′s Method?

Quadratic Programming

Find $$\underset{\mathbf{u}}{\arg\max} \quad c + \mathbf{d}^T\mathbf{u} + \frac{\mathbf{u}^T R \mathbf{u}}{2}$$ Quadratic criterion

Subject to

$$a_{11}u_1 + a_{12}u_2 + \ldots + a_{1m}u_m \le b_1$$
$$a_{21}u_1 + a_{22}u_2 + \ldots + a_{2m}u_m \le b_2$$
$$\vdots$$
$$a_{n1}u_1 + a_{n2}u_2 + \ldots + a_{nm}u_m \le b_n$$

n additional linear <u>in</u>equality constraints

And subject to

$$a_{(n+1)1}u_1 + a_{(n+1)2}u_2 + \ldots + a_{(n+1)m}u_m = b_{(n+1)}$$
$$a_{(n+2)1}u_1 + a_{(n+2)2}u_2 + \ldots + a_{(n+2)m}u_m = b_{(n+2)}$$
$$\vdots$$
$$a_{(n+e)1}u_1 + a_{(n+e)2}u_2 + \ldots + a_{(n+e)m}u_m = b_{(n+e)}$$

e additional linear <u>e</u>quality constraints

Quadratic Programming

Find $\arg\max_{\mathbf{u}} \; c + \mathbf{d}^T\mathbf{u} + \frac{\mathbf{u}^T R\mathbf{u}}{}$ Quadratic criterion

Subject to

additional linear inequality constraints

And subject to

$\ldots = b_{(n+1)}$

$\ldots = b_{(n+2)}$

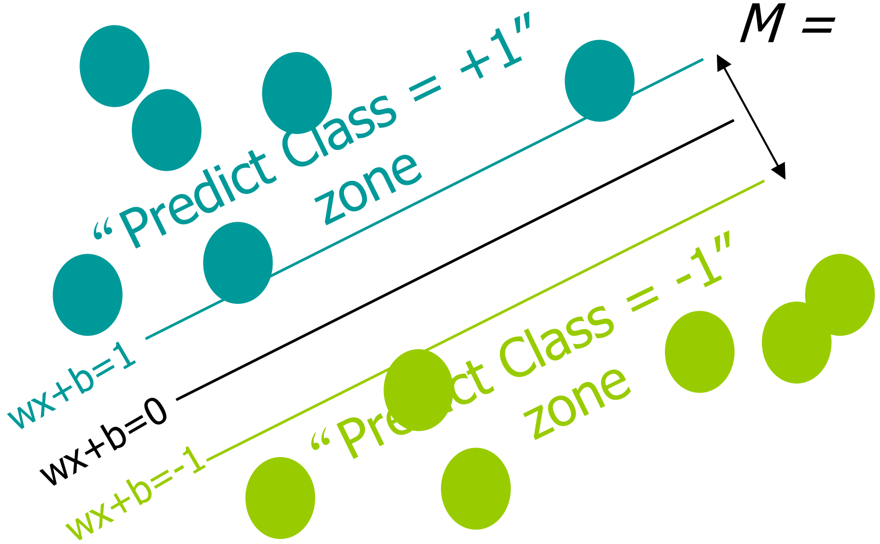
$a_{(n+e)1}u_1 + a_{(n+e)2}u_2 + \ldots + a_{(n+e)m}u_m = b_{(n+e)}$

e additional linear equality constraints

There exist algorithms for finding such constrained quadratic optima much more efficiently and reliably than gradient ascent.

(But they are very fiddly…you probably don't want to write one yourself)

Learning the Maximum Margin Classifier



$$M = \frac{2}{\sqrt{\mathbf{w}.\mathbf{w}}}$$

Given guess of $\boldsymbol{w}$, b we can

- Compute whether all data points are in the correct half-planes
- Compute the margin width

Assume R datapoints, each $(\mathbf{x}_k, y_k)$ where $\boldsymbol{y}_k = +/- \ 1$

What should our quadratic optimization criterion be?

Minimize $\boldsymbol{w}.\boldsymbol{w}$

How many constraints will we have?

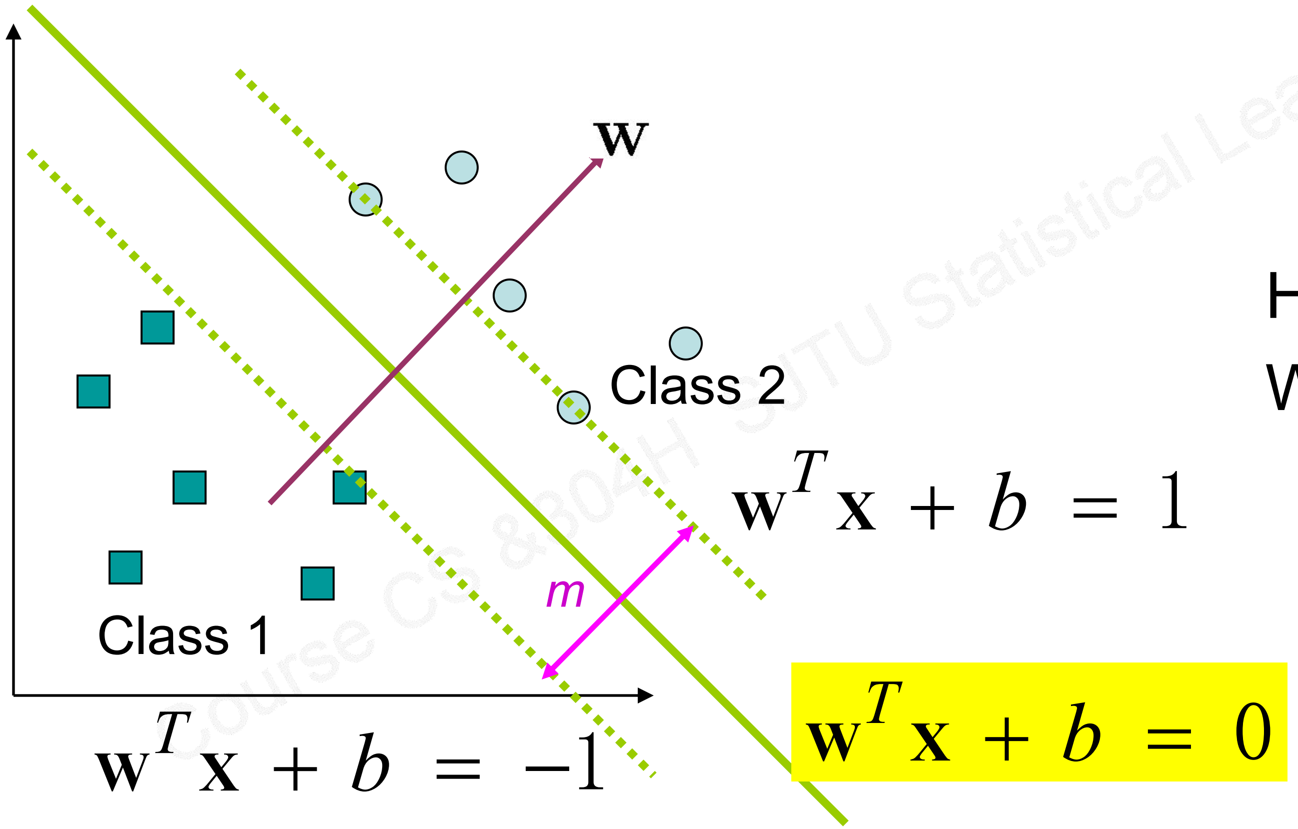
What should they be? R

$$\boldsymbol{w} \ . \ \boldsymbol{x}_k + b >= 1, \quad \text{if } y_k = 1$$

$$\boldsymbol{w} \ . \ \boldsymbol{x}_k + b <= -1, \quad \text{if } y_k = -1$$

Large-margin Decision Boundary

- The decision boundary should be as far away from the data of both classes as possible
  - We should maximize the margin, *m*
  - Distance between the origin and the line $w^t x = k$ is $k/\|\mathbf{w}\|$



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

How many constraints will we have?

What should they be? *R*

$$w \cdot x_k + b >= 1, \quad if\ y_k = 1$$

$$w \cdot x_k + b <= -1, \quad if\ y_k = -1$$

*

Finding the Decision Boundary

- Let $\{x_1, \ldots, x_n\}$ 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 $\Rightarrow$

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

- The decision boundary can be found by solving the following constrained optimization problem

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

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

- This is a constrained optimization problem. Solving it requires some new tools
  - Feel free to ignore the following several slides; what is important is the constrained optimization problem above

Back to the Original Problem

Minimize $\|\mathbf{w}\|^2 / 2$

subject to $1 - y_i(\mathbf{w}^T\mathbf{x}_i + b) \le 0,$ for $i = 1, \ldots, n$

- The Lagrangian is $\mathcal{L} = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{n} \alpha_i \left(1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)\right)$
  - Setting the gradient of L w.r.t. **w** and b to zero, we have

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

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

Illed posed problem?

*

KKT conditions

- The Karush-Kuhn-Tucker conditions,

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

  - If $\alpha_i > 0$, then $y_i(\mathbf{w}^T \mathbf{x}_i + b) = 1$, or in other word, x_i is on the boundary of the margin;
  - If $y_i(\mathbf{w}^T \mathbf{x}_i + b) \neq 1$, x_i is not on the boundary of the margin, and $\alpha_i = 0$.

*

*

The Dual Problem

- If we substitute $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$ to $\mathcal{L}$, we have

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

$$= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j + \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \alpha_i y_i \sum_{j=1}^{n} \alpha_j y_j \mathbf{x}_j^T \mathbf{x}_i - b\sum_{i=1}^{n} \alpha_i y_i$$

$$= -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j + \sum_{i=1}^{n} \alpha_i$$

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

- This is a function of α_i only

The Dual Problem

- It is known as **the dual problem**: if we know **w**, we know all α_i; if we know all α_i, we know **w**
- The objective function of the dual problem needs to **be maximized!**

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

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

Properties of α_i when we introduce the Lagrange multipliers

The result when we differentiate the original Lagrangian w.r.t. b

*

The Dual Problem

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

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

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

- $\mathbf{w}$ can be recovered by

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

*

Characteristics of the Solution

- Many of the α_i are zero
  - **w** is a linear combination of a small number of data points
- $\mathbf{x}_i$ with non-zero α_i are called support vectors (SV)
  - Let t_j (j=1, ..., s) be the indices of the s support vectors. We can write

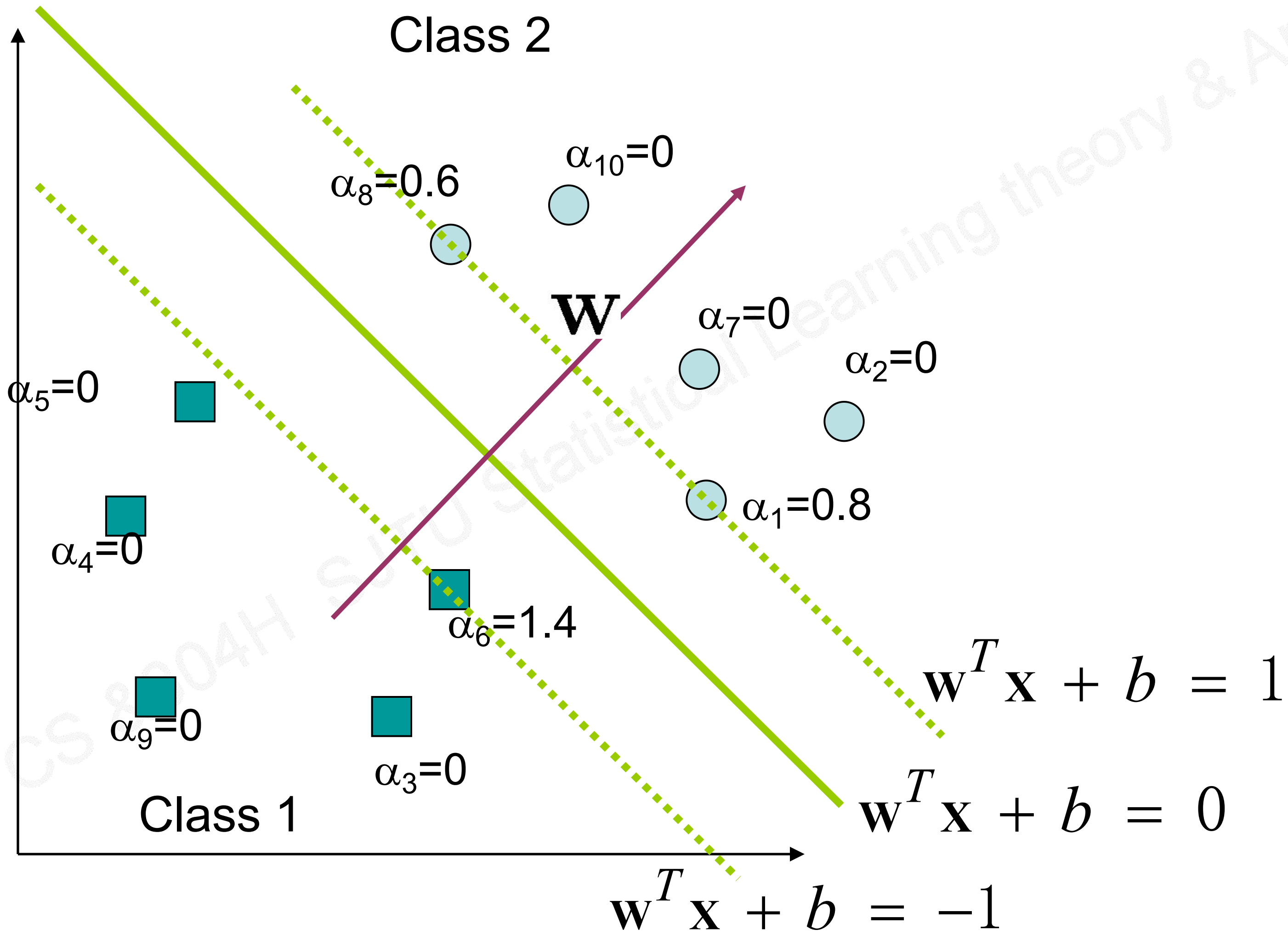
$$\mathbf{w} = \sum_{j=1}^{S} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$$

- For testing with a new data **z**
  - classify **z** as class 1 if the sum is positive, and class 2 otherwise

$$\mathbf{w}^T \mathbf{z} + b = \sum_{j=1}^{S} \alpha_{t_j} y_{t_j} (\mathbf{x}_{t_j}^T \mathbf{z}) + b$$

*

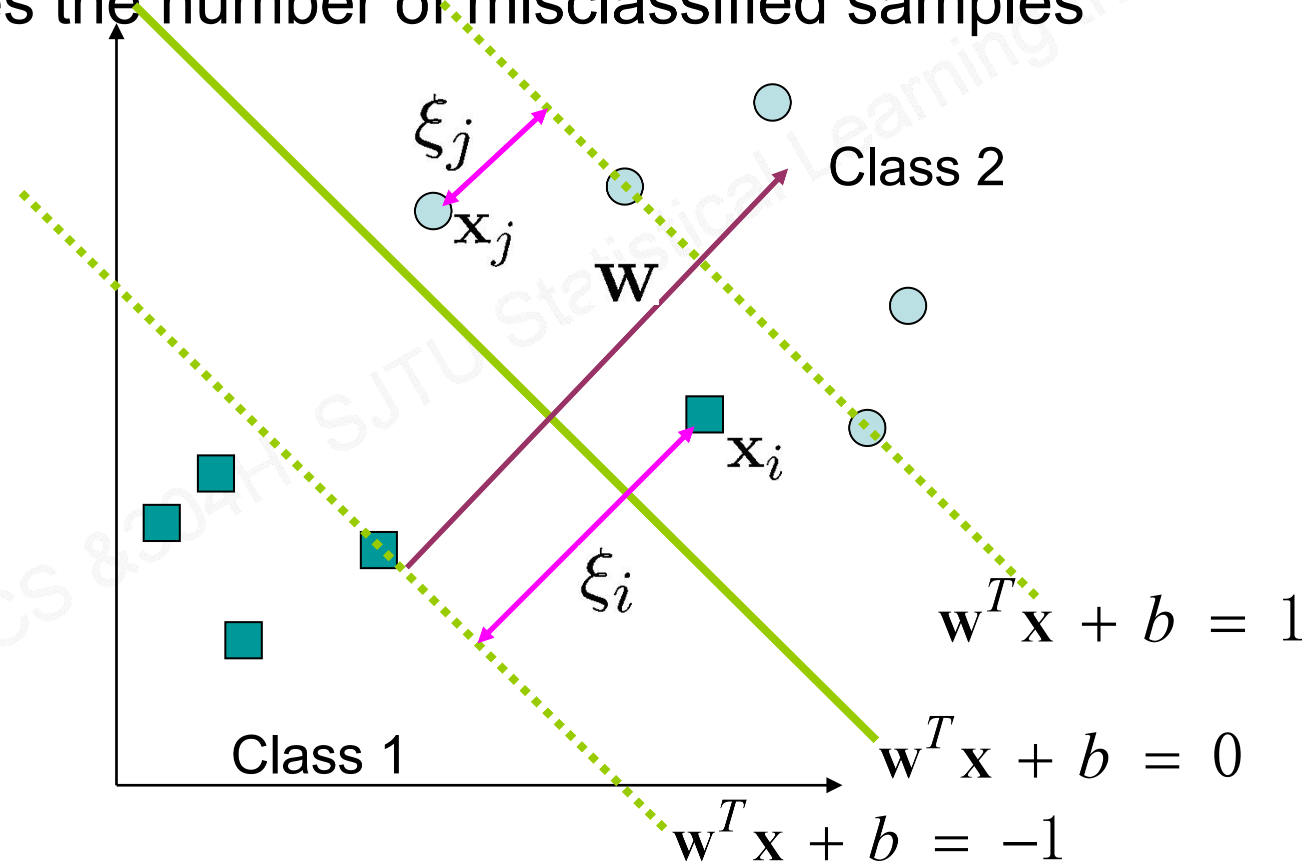
A Geometrical Interpretation



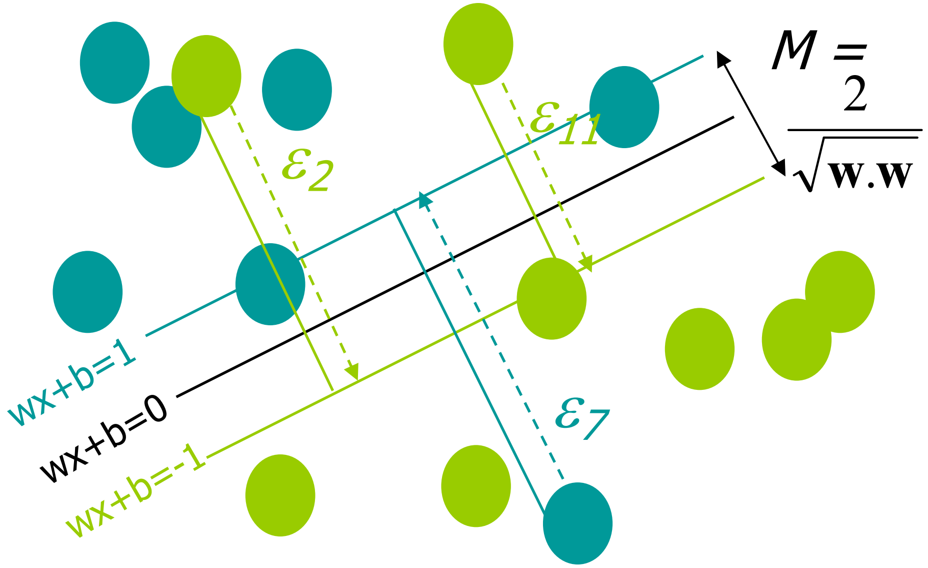
*

Non-linearly Separable Problems

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



Learning Maximum Margin with Noise



The quadratic optimization criterion:

Minimize

$$\frac{1}{2}\mathbf{w}.\mathbf{w} + C\sum_{k=1}^{R}\varepsilon_k$$

How many constraints will we have? *R*

Constraint:

$$\mathbf{w}^T\mathbf{x}_k + b \geq 1 - \varepsilon_k, \quad \text{if } y_k = 1$$

$$\mathbf{w}^T\mathbf{x}_k + b \leq -1 + \varepsilon_k, \text{ if } y_k = -1$$

Learning Maximum Margin with Noise

m = # input dimensions

Our original (noiseless data) QP had $m+1$ variables: w_1, w_2, … w_m, and b.

Our new (noisy data) QP has $m+1+R$ variables: w_1, w_2, … w_m, b, ε_k, ε_1,… ε_R

R= # samples

The quadratic optimization criterion:

Minimize

$$\frac{1}{2}\mathbf{w}.\mathbf{w} + C\sum_{k=1}^{R}\varepsilon_k$$

How many constraints will we have? 2R

Constraint:

$$y_k\left(\mathbf{w}^T\mathbf{x}_k + b\right) \geq 1-\varepsilon_k,$$

$$\varepsilon_k \geq 0, \qquad \forall \ k$$

An Equivalent Dual QP

Minimize

$$\frac{1}{2}\mathbf{w}.\mathbf{w} + C\sum_{k=1}^{R}\varepsilon_k \qquad y_k\left(\mathbf{w}^T\mathbf{x}_k + b\right) \geq 1-\varepsilon_k, \ \forall \ k$$

$$\varepsilon_k \geq 0, \quad \forall k$$

The Lagrange function:

$$L_p(w,b,\varepsilon) = \frac{1}{2}\mathbf{w}.\mathbf{w} + C\sum_{k=1}^{R}\varepsilon_k - \sum_{k=1}^{R}\mu_k\varepsilon_k -$$

$$-\sum_{k=1}^{R}\alpha_k\left[y_k\left(\mathrm{w}^T\mathbf{x}_k + b\right) - \left(1\text{-}\varepsilon_k\right)\right]$$

An Equivalent Dual QP

The Lagrange function:

$$L_p(w,b,\varepsilon) = \frac{1}{2}\mathbf{w}.\mathbf{w} + C\sum_{k=1}^{R}\varepsilon_k - \sum_{k=1}^{R}\mu_k\varepsilon_k$$

$$-\sum_{k=1}^{R}\alpha_k\left[y_k\left(\mathrm{w}^T\mathbf{x}_k + b\right) - \left(1\text{-}\varepsilon_k\right)\right]$$

- Setting the respective derivatives to zero, we get

$$\mathbf{w} = \sum_{k=1}^{R}\alpha_k y_k \mathbf{x}_k, \qquad \sum_{k=1}^{R}\alpha_k y_k = 0,$$

$$\alpha_i = C - \mu_i, \forall i, \qquad \alpha_i, \mu_i, \varepsilon_i > 0, \forall i$$

An Equivalent Dual QP

Minimize

$$\frac{1}{2}\mathbf{w}.\mathbf{w} + C\sum_{k=1}^{R}\varepsilon_k \qquad y_k\left(\mathbf{w}^T\mathbf{x}_k + b\right) \geq 1 - \varepsilon_k, \quad \forall\ k$$

$$\varepsilon_k \geq 0, \quad \forall k$$

Dual QP

Maximize $\sum_{k=1}^{R}\alpha_k - \frac{1}{2}\sum_{k=1}^{R}\sum_{l=1}^{R}\alpha_k\alpha_l Q_{kl}$ where $Q_{kl} = y_k y_l(\mathbf{x}_k.\mathbf{x}_l)$

Subject to these constraints: $0 \leq \alpha_k \leq C \quad \forall k \qquad \sum_{k=1}^{R}\alpha_k y_k = 0$

An Equivalent Dual QP

Maximize $\sum_{k=1}^{R} \alpha_k - \frac{1}{2} \sum_{k=1}^{R} \sum_{l=1}^{R} \alpha_k \alpha_l Q_{kl}$ where $Q_{kl} = y_k y_l (\mathbf{x}_k . \mathbf{x}_l)$

Subject to these constraints: $0 \leq \alpha_k \leq C \quad \forall k$ $\quad \sum_{k=1}^{R} \alpha_k y_k = 0$

Then define:

$$\mathbf{w} = \sum_{k=1}^{R} \alpha_k y_k \mathbf{x}_k$$

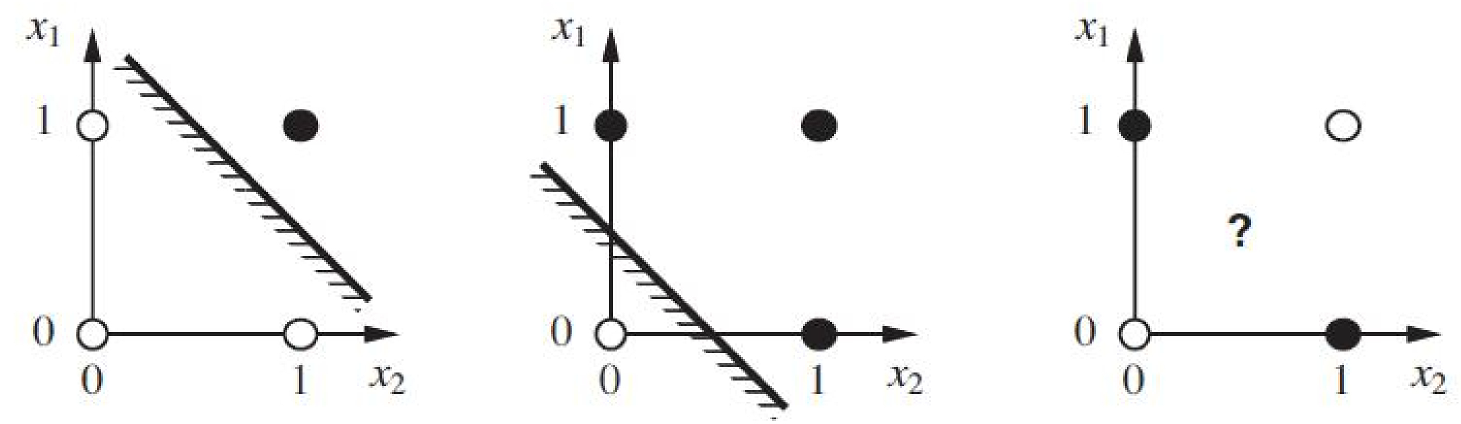
$$b = y_K (1 - \varepsilon_K) - \mathbf{x}_K . \mathbf{w}_K$$

where $K = \arg \max_k \alpha_k$

Then classify with:

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

Nonlinear Classification Problem



(a) x_1 **and** x_2 (b) x_1 **or** x_2 (c) x_1 **xor** x_2

Linear separability in threshold perceptrons. Black dots indicate a point in the input space where the value of the function is 1, and white dots indicate a point where the value is 0. The perceptron returns 1 on the region on the non-shaded side of the line. In (c), no such line exists that correctly classifies the inputs.

Example XOR problem revisited:

Let the nonlinear mapping be :

$$\varphi(\mathbf{x}) = \left(1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1 x_2, x_2^2\right)^T ; \mathbf{x} = (x_1, x_2)^T$$

And: $$\varphi(\mathbf{x}_i) = \left(1, \sqrt{2}x_{i1}, \sqrt{2}x_{i2}, x_{i1}^2, \sqrt{2}x_{i1}x_{i2}, x_{ii2}^2\right)^T$$

Therefore the feature space is in 6D with input data in 2D

$\underline{x}_1 = (-1,-1), \quad d_1 = -1$

$\underline{x}_2 = (-1, 1), \quad d_2 = 1$

$\underline{x}_3 = (1,-1), \quad d_3 = 1$

$\underline{x}_4 = (-1,-1), \quad d_4 = -1$

$$Q(a)= \Sigma a_i - \tfrac{1}{2} \Sigma \Sigma a_i a_j d_i d_j \phi(\underline{x}_i)^T \phi(\underline{x}_j)$$

$$=a_1 + a_2 + a_3 + a_4 - \tfrac{1}{2}(9 a_1 a_1 - 2a_1 a_2 - 2 a_1 a_3 + 2a_1 a_4$$

$$+9a_2 a_2 + 2a_2 a_3 - 2a_2 a_4 + 9a_3 a_3 - 2a_3 a_4 + 9 a_4 a_4)$$

To minimize Q, calculate

$$\frac{\partial Q(a)}{\partial a_i} = 0, i = 1, \ldots, 4$$

(due to optimality conditions) which gives

$$1 = 9 a_1 - a_2 - a_3 + a_4$$

$$1 = -a_1 + 9 a_2 + a_3 - a_4$$

$$1 = -a_1 + a_2 + 9 a_3 - a_4$$

$$1 = a_1 - a_2 - a_3 + 9 a_4$$

The solution of which gives the optimal values:

$$a_{0,1} = a_{0,2} = a_{0,3} = a_{0,4} = 1/8$$

$$\underline{w}_0 = \Sigma\, a_{0,i}\, d_i\, \phi(\underline{x}_i)$$
$$= 1/8[\phi(\underline{x}_1) - \phi(\underline{x}_2) - \phi(\underline{x}_3) + \phi(\underline{x}_4)]$$

$$= \frac{1}{8}\left[-\begin{pmatrix} 1 \\ 1 \\ \sqrt{2} \\ 1 \\ -\sqrt{2} \\ -\sqrt{2} \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ -\sqrt{2} \\ 1 \\ -\sqrt{2} \\ \sqrt{2} \end{pmatrix} - \begin{pmatrix} 1 \\ 1 \\ -\sqrt{2} \\ 1 \\ \sqrt{2} \\ -\sqrt{2} \end{pmatrix} - \begin{pmatrix} 1 \\ 1 \\ \sqrt{2} \\ 1 \\ \sqrt{2} \\ \sqrt{2} \end{pmatrix} \right] = \begin{pmatrix} 0 \\ 0 \\ -1/\sqrt{2} \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

where the first element of w_0 gives the bias b

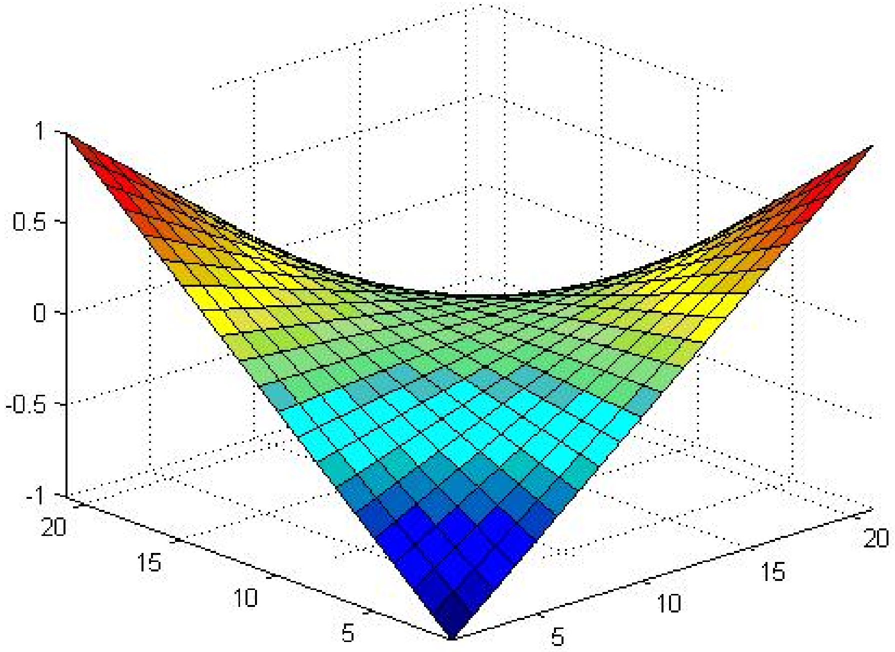
From earlier we have that the optimal hyperplane is defined by:

$$\underline{w}_0^T \phi(\underline{x}) = 0$$

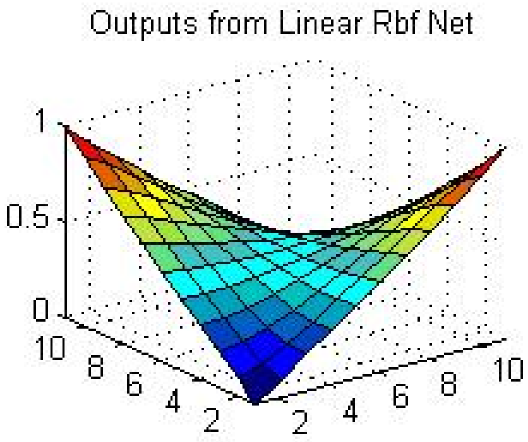
That is:

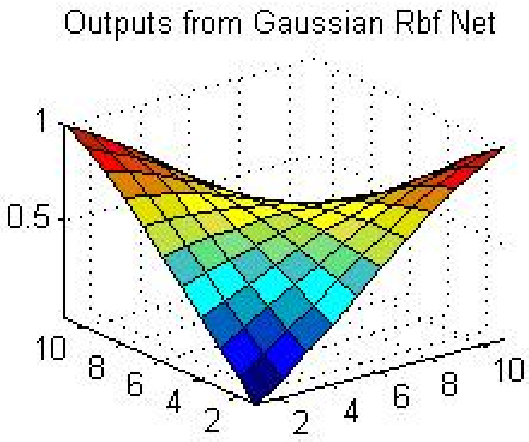
$$\underline{w}_0^T \phi(\underline{x}) = \begin{pmatrix} 0 & 0 & -1/\sqrt{2} & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \end{pmatrix} = -x_1x_2 = 0$$

which is the optimal decision boundary for the XOR problem. Furthermore we note that the solution is unique since the optimal decision boundary is unique



Output for polynomial RBF





For a non-linearly separable problem we have to first map data onto feature space so that they are linear separable

$$x_i \rightarrow \varphi(x_i) \qquad \text{sample:} \qquad \{(\varphi(x_i), y_i\}_{i=1}^{R}$$

Given the training data sample {($\underline{x}_i$,y_i), $i=1, ...,N$}, find the optimum values of the weight vector w and bias b

$$w = \sum_i a_i y_i \varphi(x_i)$$

where $a_{0,i}$ are the optimal Lagrange multipliers determined by maximizing the following objective function

$$Q(a) = \sum_{i=1}^{N} a_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} a_i a_j d_i d_j \underline{\phi}^T(\underline{x}_i)\underline{\phi}(\underline{x}_j)$$

subject to the constraints

$$\sum_{j=1}^{N} a_i y_i = 0; \quad a_i \geq 0, \forall \text{i}$$

SVM building procedure:

1. Pick a nonlinear mapping f
2. Solve for the optimal weight vector

However: how do we pick the function f?

- In practical applications, if it is not totally impossible to find f, it is very hard
- In the previous example, the function f is quite complex: How would we find it?

Answer: ***the Kernel Trick***

Notice that in the dual problem the inner product of input vectors is replaced by an inner-product kernel

$$
\begin{aligned}
Q(a) &= \sum_{i=1}^{N} a_i - \frac{1}{2}\sum_{i,j=1}^{N} a_i a_j d_i d_j \varphi_j(\underline{x})^T \varphi_j(\underline{x}_i) \\
&= \sum_{i=1}^{N} a_i - \frac{1}{2}\sum_{i,j=1}^{N} a_i a_j d_i d_j K(\underline{x_i}, \underline{x_j})
\end{aligned}
$$

How do we relate the output of the SVM to the kernel K?

Look at the equation of the boundary in the feature space and use the optimality conditions derived from the Lagrangian formulations

Hyperplane is defined by

$$\sum_{j=1}^{m1} w_j \varphi_j(\underline{x}) + b = 0$$

or

$$\sum_{j=0}^{m1} w_j \varphi_j(\underline{x}) = 0; \quad where\ \varphi_0(\underline{x}) = 1$$

$$writing:\ \underline{\varphi}(\underline{x}) = [\varphi_0(\underline{x}), \varphi_1(\underline{x}), ..., \varphi_{m1}(\underline{x})]$$

$$we\ get:\quad \underline{w}^T \underline{\varphi}(\underline{x}) = 0$$

$$from\ optimality\ conditions:\ \underline{w} = \sum_{i=1}^{N} a_i d_i \underline{\varphi}(\underline{x}_i)$$

Thus the decision boundary:

$$\sum_{i=1}^{N} a_i d_i \underline{\varphi}^T(\underline{x}_i)\underline{\varphi}(\underline{x}) = 0; \qquad or \qquad \sum_{i=1}^{N} a_i d_i K(\underline{x}, \underline{x}_i) = 0$$

Now the decision output:

$$y = \underline{w}^T \underline{\varphi}(\underline{x}) = \sum_{i=1}^{N} a_i d_i K(\underline{x}, \underline{x}_i)$$

$$where: K(\underline{x}, \underline{x}_i) = \underline{\varphi}^T(\underline{x}_i)\underline{\varphi}(\underline{x})$$

We therefore only need a suitable choice of kernel function cf: Mercer's Theorem:

Let $K(\underline{x},\underline{y})$ be a continuous symmetric kernel that defined in the closed interval [a,b]. The kernel K can be expanded in the form

$$K(\underline{x},\underline{y})=\varphi^T(\underline{y})\varphi(\underline{x})$$

provided it is positive definite. Some of the usual choices for K are:

Polynomial SVM	$(\underline{x}^T\underline{x_i}+1)^p$	p specified by user
RBF SVM	$\exp(-1/(2\sigma^2)\ \|\|\underline{x}-\underline{x_i}\|\|^2)$	σ specified by user
MLP SVM	$\tanh(s_0\,\underline{x}^T\underline{x_i}+s_1)$	

Complexity Analysis

- Why the kernel approach provides feasible solution for implementing SVM?

An Equivalent Dual QP

Maximize $\sum_{k=1}^{R}\alpha_k - \frac{1}{2}\sum_{k=1}^{R}\sum_{l=1}^{R}\alpha_k\alpha_l Q_{kl}$ where $Q_{kl} = y_k y_l(\mathbf{x}_k.\mathbf{x}_l)$

Constraints: $0 \le \alpha_k \le C \quad \forall k$ $\quad \sum_{k=1}^{R}\alpha_k y_k = 0$

Datapoints with $a_k > 0$ will be the support vectors

Then define:

$$\mathbf{w} = \sum_{k=1}^{R}\alpha_k y_k \mathbf{x}_k$$

$$b = y_K(1-\varepsilon_K) - \mathbf{x}_K.\mathbf{w}_K$$

$$\text{where} \quad K = \arg\max_k \alpha_k$$

..so this sum only needs to be over the support vectors.

Then classify with:

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

Quadratic Basis Functions

$$
\Phi(\mathbf{x}) = \begin{pmatrix} 1 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ \vdots \\ \sqrt{2}x_m \\ x_1^2 \\ x_2^2 \\ \vdots \\ x_m^2 \\ \sqrt{2}x_1x_2 \\ \sqrt{2}x_1x_3 \\ \vdots \\ \sqrt{2}x_1x_m \\ \sqrt{2}x_2x_3 \\ \vdots \\ \sqrt{2}x_1x_m \\ \vdots \\ \sqrt{2}x_{m-1}x_m \end{pmatrix}
$$

Constant Term

Linear Terms

Pure Quadratic Terms

Quadratic Cross-Terms

Number of terms (assuming m input dimensions) = (m+2)(m+1)/2

= (as near as makes no difference) $m^2/2$

You may be wondering what those $\sqrt{2}$'s are doing.

QP with basis functions

Maximize $\sum_{k=1}^{R}\alpha_k - \frac{1}{2}\sum_{k=1}^{R}\sum_{l=1}^{R}\alpha_k\alpha_l Q_{kl}$ where $Q_{kl} = y_k y_l(\mathbf{\Phi}(\mathbf{x}_k).\mathbf{\Phi}(\mathbf{x}_l))$

Constraints: $0 \le \alpha_k \le C \quad \forall k \qquad \sum_{k=1}^{R}\alpha_k y_k = 0$

Then define:

$$\mathbf{w} = \sum_{k \text{ s.t. } \alpha_k > 0} \alpha_k y_k \mathbf{\Phi}(\mathbf{x}_k)$$

$$b = y_K(1-\varepsilon_K) - \mathbf{x}_K . \mathbf{w}_K$$

$$\text{where } K = \arg\max_k \alpha_k$$

Then classify with:

f(x,w,b) = sign(w. ϕ***(x) - b)***

$$
\mathbf{\Phi}(\mathbf{a})^T \mathbf{\Phi}(\mathbf{b}) = \begin{pmatrix} 1 \\ \sqrt{2}a_1 \\ \sqrt{2}a_2 \\ \vdots \\ \sqrt{2}a_m \\ a_1^2 \\ a_2^2 \\ \vdots \\ a_m^2 \\ \sqrt{2}a_1a_2 \\ \sqrt{2}a_1a_3 \\ \vdots \\ \sqrt{2}a_1a_m \\ \sqrt{2}a_2a_3 \\ \vdots \\ \sqrt{2}a_1a_m \\ \vdots \\ \sqrt{2}a_{m-1}a_m \end{pmatrix} \bullet \begin{pmatrix} 1 \\ \sqrt{2}b_1 \\ \sqrt{2}b_2 \\ \vdots \\ \sqrt{2}b_m \\ b_1^2 \\ b_2^2 \\ \vdots \\ b_m^2 \\ \sqrt{2}b_1b_2 \\ \sqrt{2}b_1b_3 \\ \vdots \\ \sqrt{2}b_1b_m \\ \sqrt{2}b_2b_3 \\ \vdots \\ \sqrt{2}b_1b_m \\ \vdots \\ \sqrt{2}b_{m-1}b_m \end{pmatrix}
$$

$$1$$

$$+$$

$$\sum_{i=1}^{m} 2a_ib_i$$

$$+$$

$$\sum_{i=1}^{m} a_i^2 b_i^2$$

Quadratic Dot Products

$$+$$

$$\sum_{i=1}^{m} \sum_{j=i+1}^{m} 2a_ia_jb_ib_j$$

Quadratic Dot Products

$$\Phi(\mathbf{a}) \bullet \Phi(\mathbf{b}) =$$

$$1 + 2\sum_{i=1}^{m} a_i b_i + \sum_{i=1}^{m} a_i^2 b_i^2 + \sum_{i=1}^{m} \sum_{j=i+1}^{m} 2 a_i a_j b_i b_j$$

Just out of casual, innocent, interest, let's look at another function of ***a*** and ***b***:

$$(\mathbf{a}.\mathbf{b} + 1)^2$$

$$= (\mathbf{a}.\mathbf{b})^2 + 2\mathbf{a}.\mathbf{b} + 1$$

$$= \left(\sum_{i=1}^{m} a_i b_i \right)^2 + 2\sum_{i=1}^{m} a_i b_i + 1$$

$$= \sum_{i=1}^{m} \sum_{j=1}^{m} a_i b_i a_j b_j + 2\sum_{i=1}^{m} a_i b_i + 1$$

$$= \sum_{i=1}^{m} (a_i b_i)^2 + 2\sum_{i=1}^{m} \sum_{j=i+1}^{m} a_i b_i a_j b_j + 2\sum_{i=1}^{m} a_i b_i + 1$$

Quadratic Dot Products

$$\mathbf{\Phi}(\mathbf{a}) \bullet \mathbf{\Phi}(\mathbf{b}) =$$

$$1 + 2\sum_{i=1}^{m} a_i b_i + \sum_{i=1}^{m} a_i^2 b_i^2 + \sum_{i=1}^{m} \sum_{j=i+1}^{m} 2 a_i a_j b_i b_j$$

Just out of casual, innocent, interest, let's look at another function of ***a*** and ***b***:

$$K(a,b) = (\mathbf{a}.\mathbf{b}+1)^2$$

$$= (\mathbf{a}.\mathbf{b})^2 + 2\mathbf{a}.\mathbf{b} + 1$$

$$= \left(\sum_{i=1}^{m} a_i b_i\right)^2 + 2\sum_{i=1}^{m} a_i b_i + 1$$

$$= \sum_{i=1}^{m} \sum_{j=1}^{m} a_i b_i a_j b_j + 2\sum_{i=1}^{m} a_i b_i + 1$$

$$= \sum_{i=1}^{m} (a_i b_i)^2 + 2\sum_{i=1}^{m} \sum_{j=i+1}^{m} a_i b_i a_j b_j + 2\sum_{i=1}^{m} a_i b_i + 1$$

They're the same!

And this is only O(m) to compute!

QP with basis functions

Maximize $\sum_{k=1}^{R} \alpha_k - \frac{1}{2}\sum_{k=1}^{R}\sum_{l=1}^{R} \alpha_k \alpha_l Q_{kl}$ where $Q_{kl} = y_k y_l (\mathbf{\Phi}(\mathbf{x}_k).\mathbf{\Phi}(\mathbf{x}_l))$

Constraints: $0 \le \alpha_k \le C \quad \forall k \qquad \sum_{k=1}^{R} \alpha_k y_k = 0$

Then define:

$$\mathbf{w} = \sum_{k \text{ s.t. } \alpha_k > 0} \alpha_k y_k \mathbf{\Phi}(\mathbf{x}_k)$$

$$b = y_K (1 - \varepsilon_K) - \mathbf{x}_K . \mathbf{w}_K$$

$$\text{where} \quad K = \arg\max_k \alpha_k$$

We must do $R^2/2$ dot products to get this matrix ready.

Each dot product requires $m^2/2$ additions and multiplications

The whole thing costs $R^2 m^2/4$.

Higher Order Polynomials

Polynomial	$\phi(x)$	Cost to build Q_{kl} matrix traditionally	Cost $M=100$	$\phi(a).\phi(b)$	Cost to build Q_{kl} matrix efficiently	Cost if 100 inputs
Quadratic	All $m^2/2$ terms up to degree 2	$m^2 R^2 /4$	$2{,}500\ R^2$	$(a.b+1)^2$	$m\ R^2 / 2$	$50\ R^2$
Cubic	All $m^3/6$ terms up to degree 3	$m^3 R^2 /12$	$83{,}000\ R^2$	$(a.b+1)^3$	$m\ R^2 / 2$	$50\ R^2$
Quartic	All $m^4/24$ terms up to degree 4	$m^4 R^2 /48$	$1{,}960{,}000\ R^2$	$(a.b+1)^4$	$m\ R^2 / 2$	$50\ R^2$

QP with Quintic basis functions

We must do R²/2 dot products to get this matrix ready.

In 100-d, each dot product now needs 103 operations instead of 75 million

But there are still worrying things lurking away. What are they?

The use of Maximum Margin magically makes this not a problem

$\forall k$ $\qquad \sum \alpha_k y_k = 0$

•The fear of overfitting with this enormous number of terms

•The evaluation phase (doing a set of predictions on a test set) will be very expensive (why?)

Because each **w**. ϕ(**x**) (see below) needs 75 million operations. *What can be done?*

Then define:

$$\mathbf{w} = \sum_{k \text{ s.t. } \alpha_k > 0} \alpha_k y_k \boldsymbol{\Phi}(\mathbf{x}_k)$$

$$b = y_K(1 - \varepsilon_K) - \mathbf{x}_K^T \mathbf{w}_K$$

$$\text{where } K = \arg\max_k \alpha_k$$

Then classify with:

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

SVM Kernel Functions

- $K(\boldsymbol{a},\boldsymbol{b})=(\boldsymbol{a}\ .\ \boldsymbol{b}+1)^d$ is an example of an SVM Kernel Function
- Beyond polynomials there are other very high dimensional basis functions that can be made practical by finding the right Kernel Function
  - Radial-Basis-style Kernel Function:

$$K(\mathrm{a},\mathrm{b}) = exp\left(-\frac{(\mathrm{a}-\mathrm{b})^2}{2\sigma^2}\right)$$

  - Neural-net-style Kernel Function:

$$K(\mathrm{a},\mathrm{b}) = tanh(\kappa \mathrm{a}.\mathrm{b} - \delta)$$

σ, κ and δ are magic parameters that must be chosen by a model selection method such as CV or VCSRM

The End of Talk