Support Vector Machine

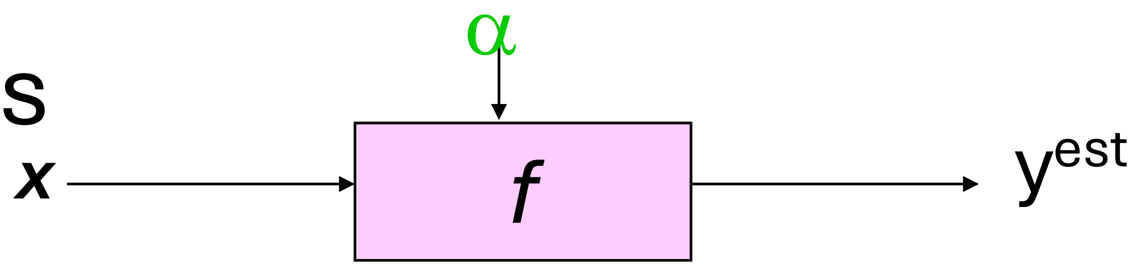
Thushari Silva, PhD

Professor in AI

Department of Computational Mathematics

University of Moratuwa

Linear Classifiers



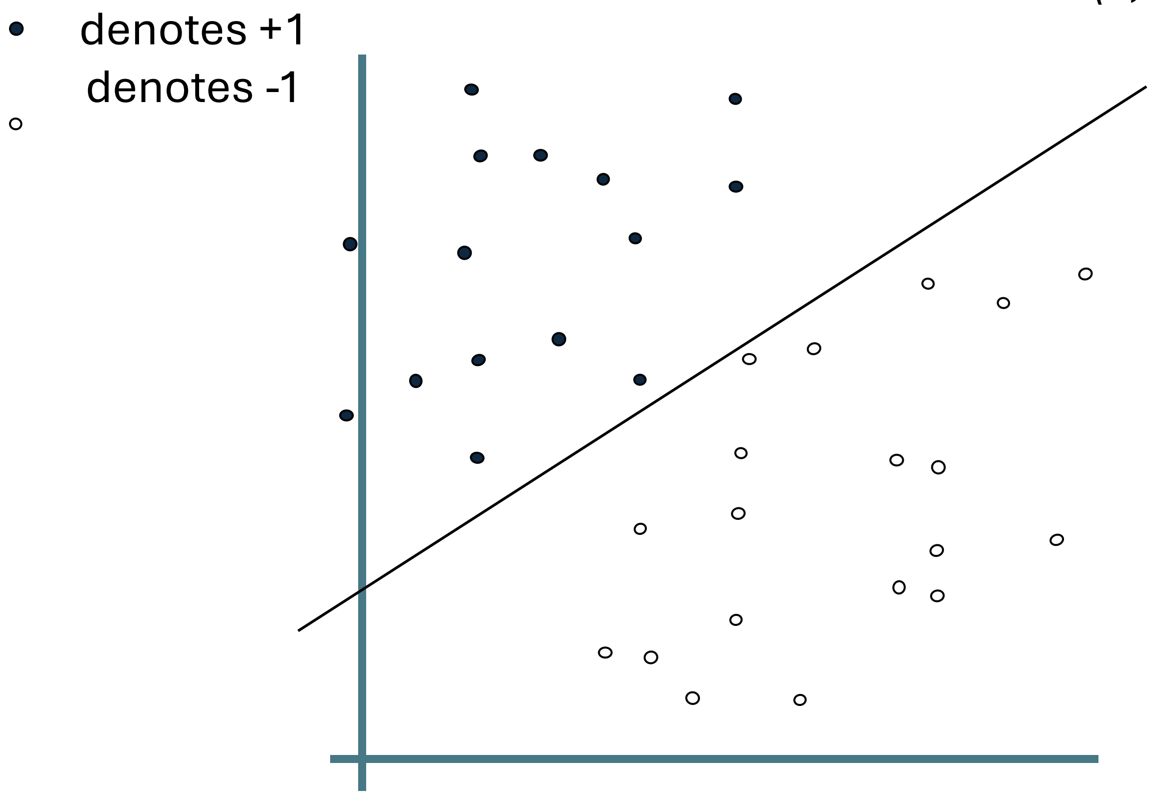
α

$\boldsymbol{x}$

f

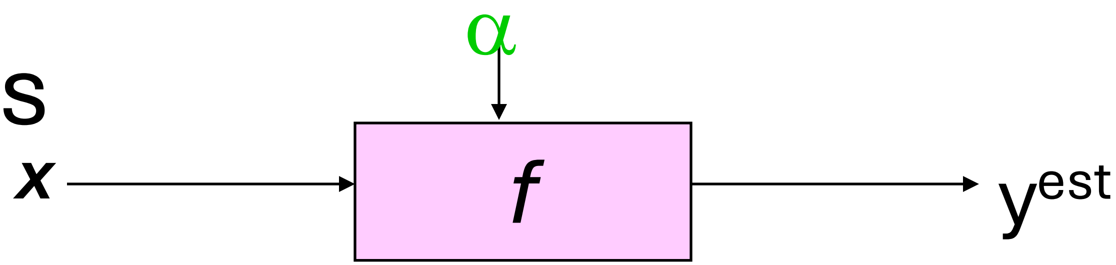
y^{est}

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

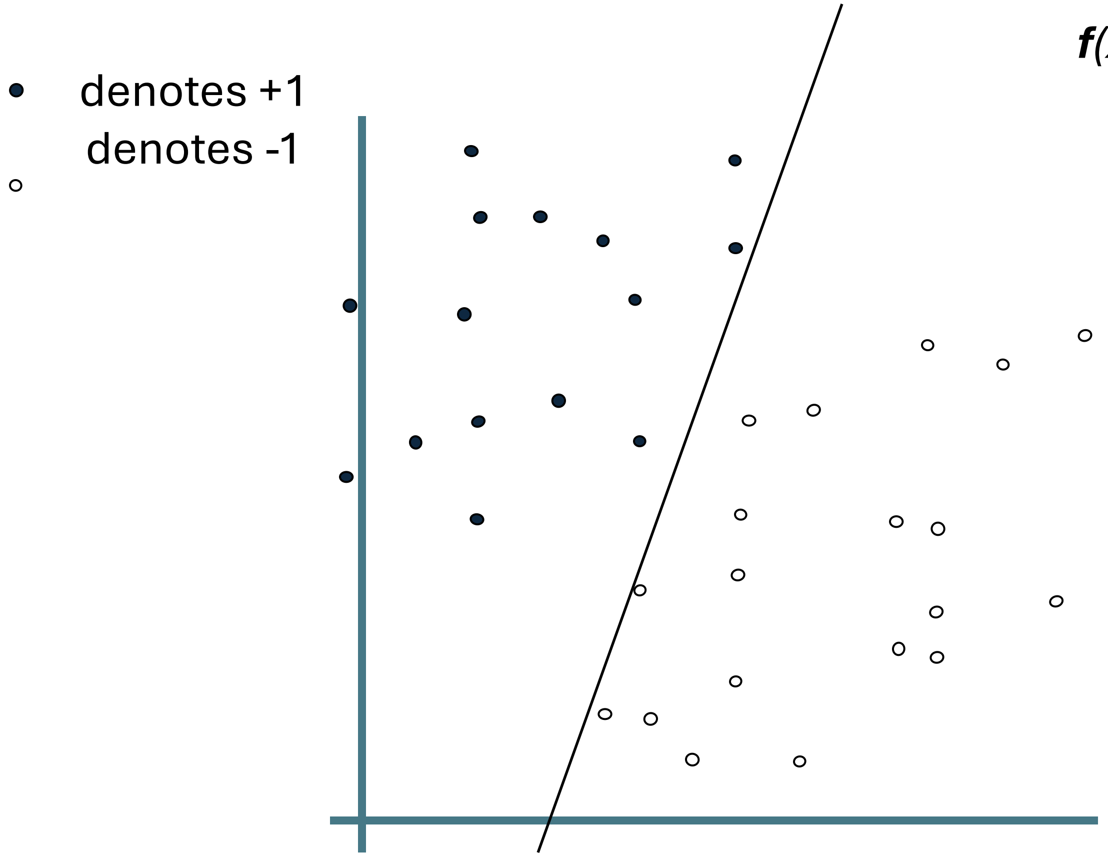


How would you classify this data?

Linear Classifiers

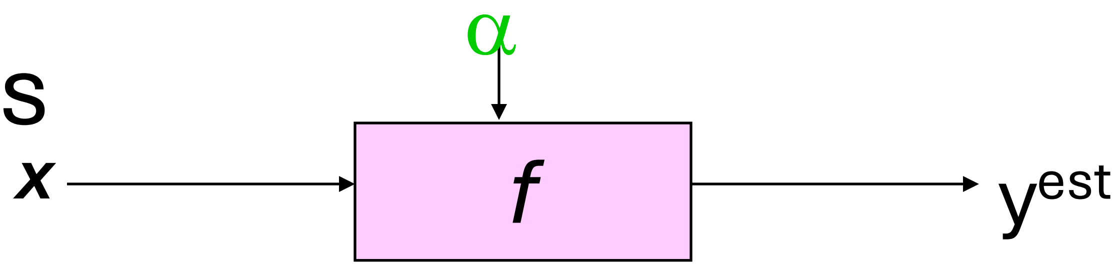


$f(x, w, b) = sign(w. x - b)$



How would you classify this data?

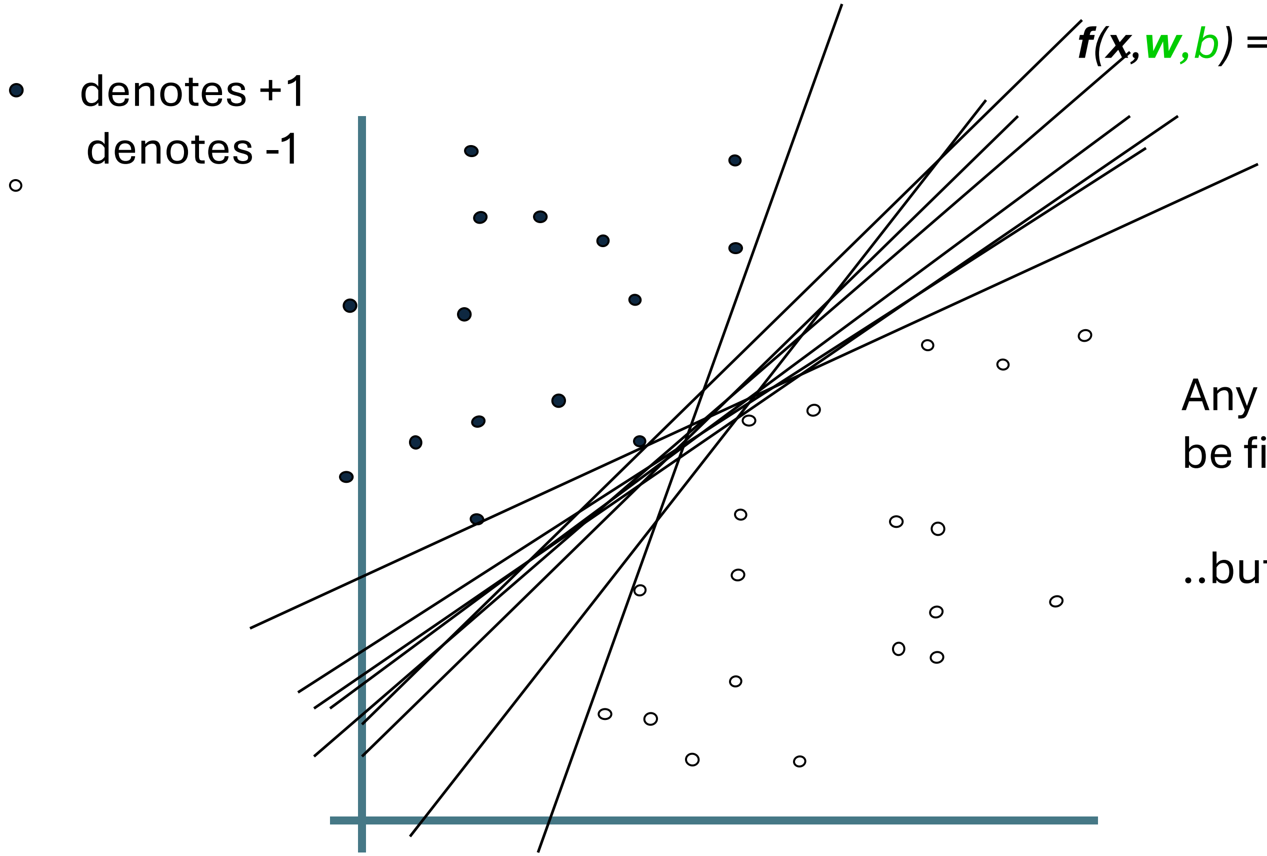
Linear Classifiers



α

$\boldsymbol{x} \rightarrow \boxed{f} \rightarrow y^{est}$

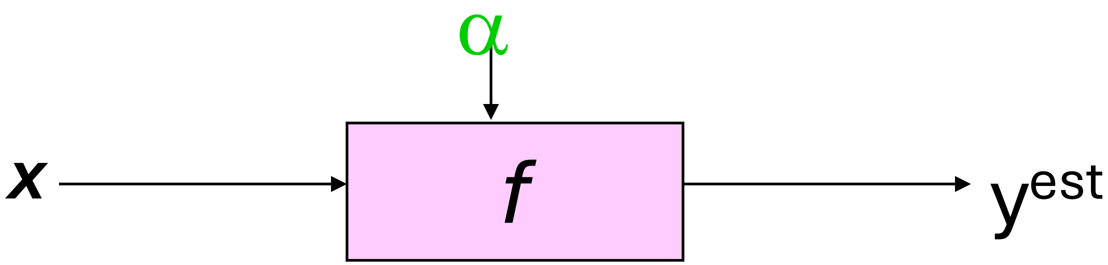
f(***x***,***w***,*b*) = *sign*(***w***. ***x*** - *b*)



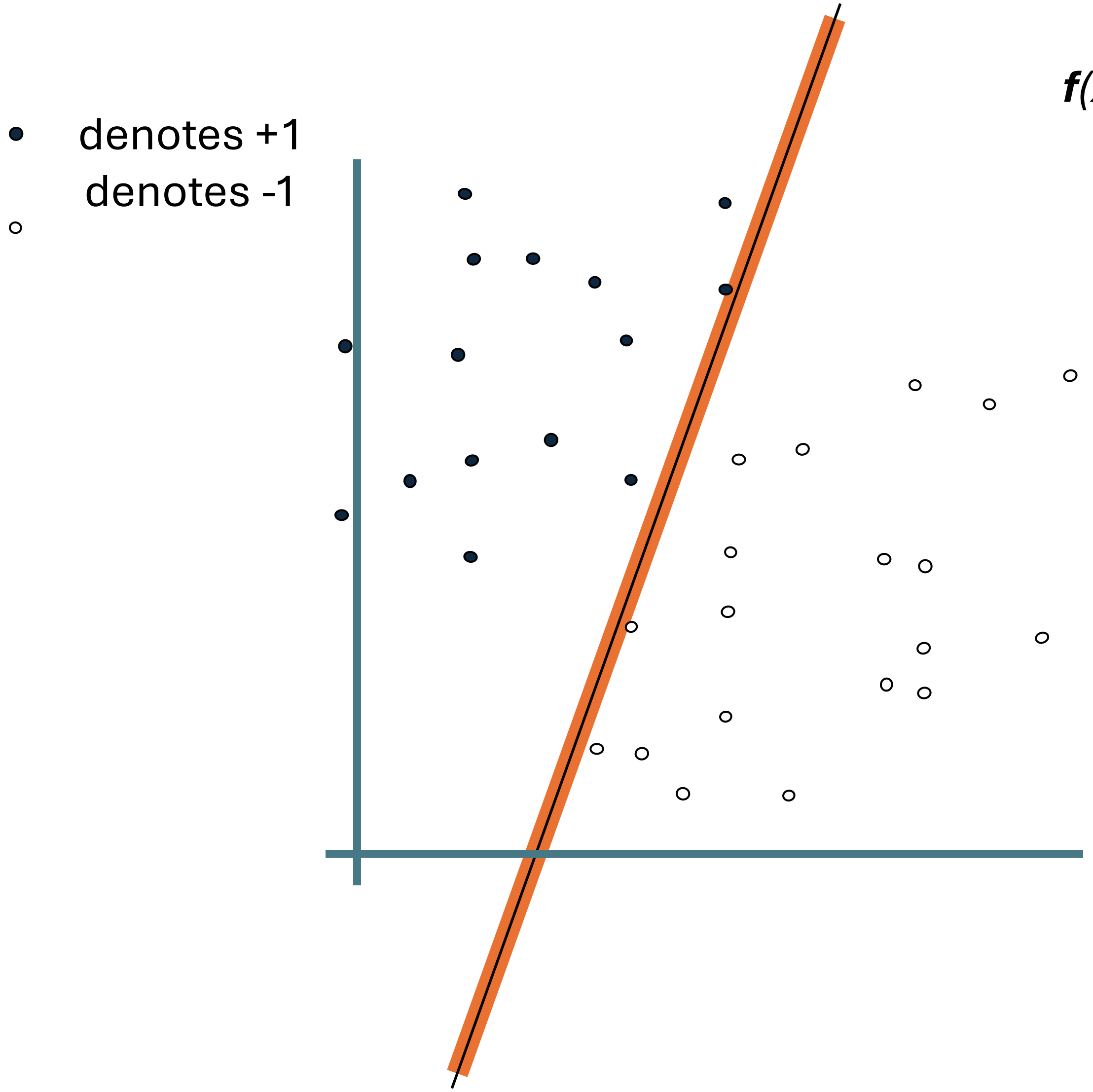
Any of these would be fine..

..but which is best?

Classifier Margin

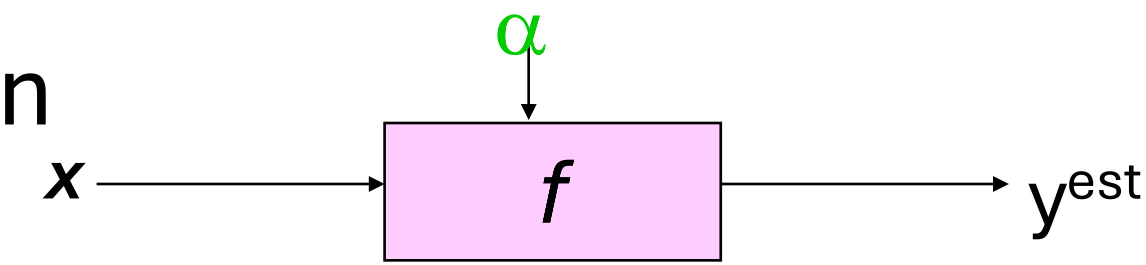


$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} - b)$

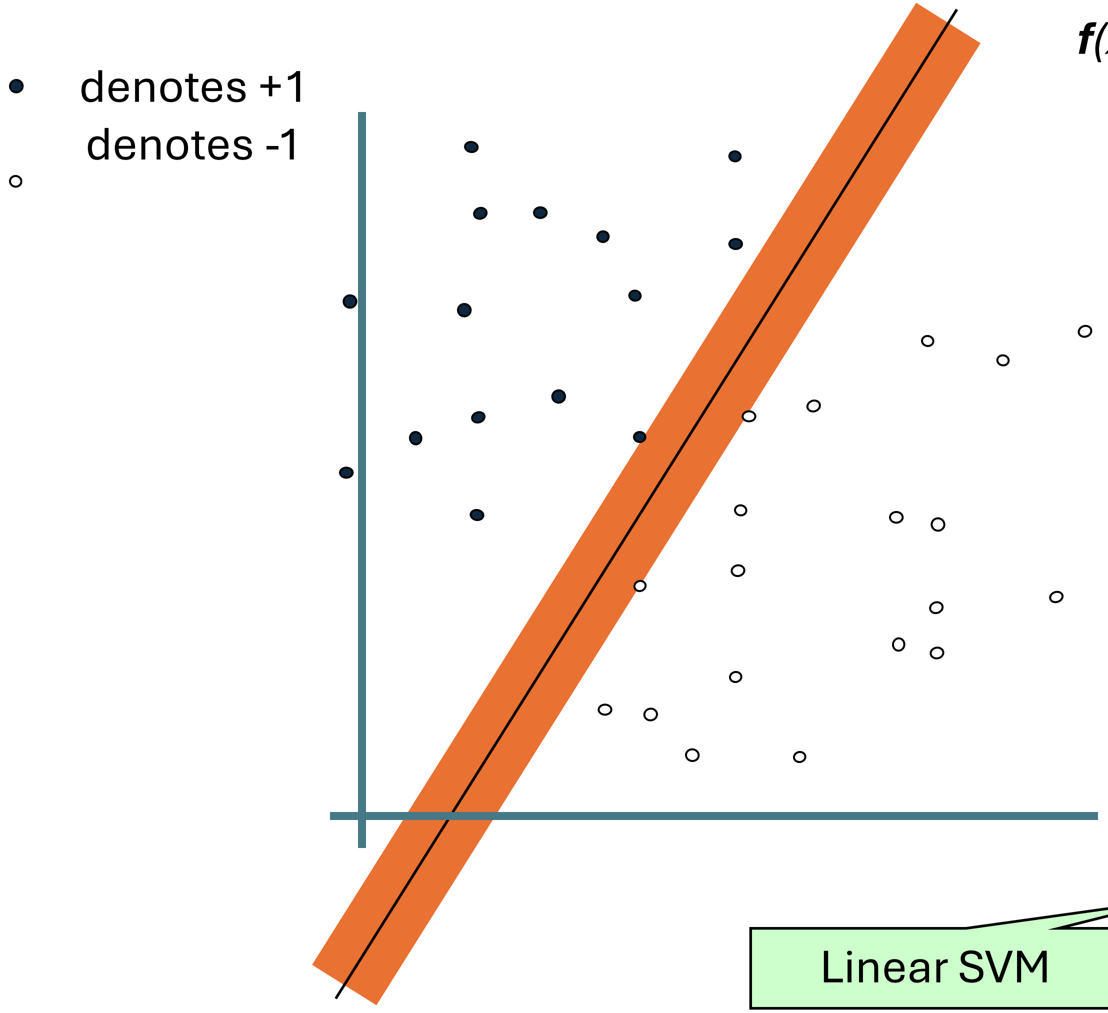


Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

Maximum Margin



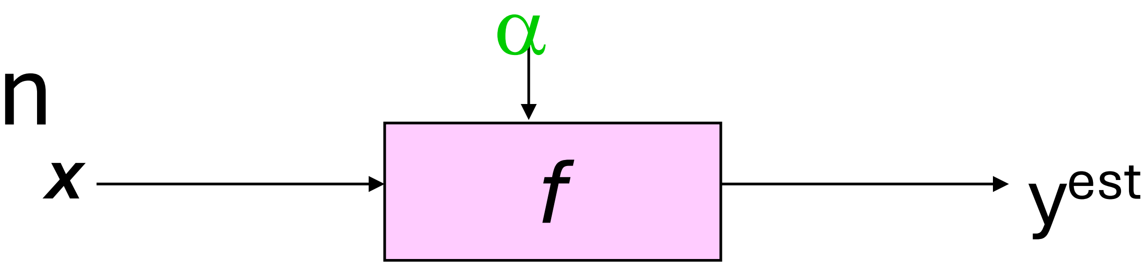
$f(x,w,b) = sign(w.\ x - b)$



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

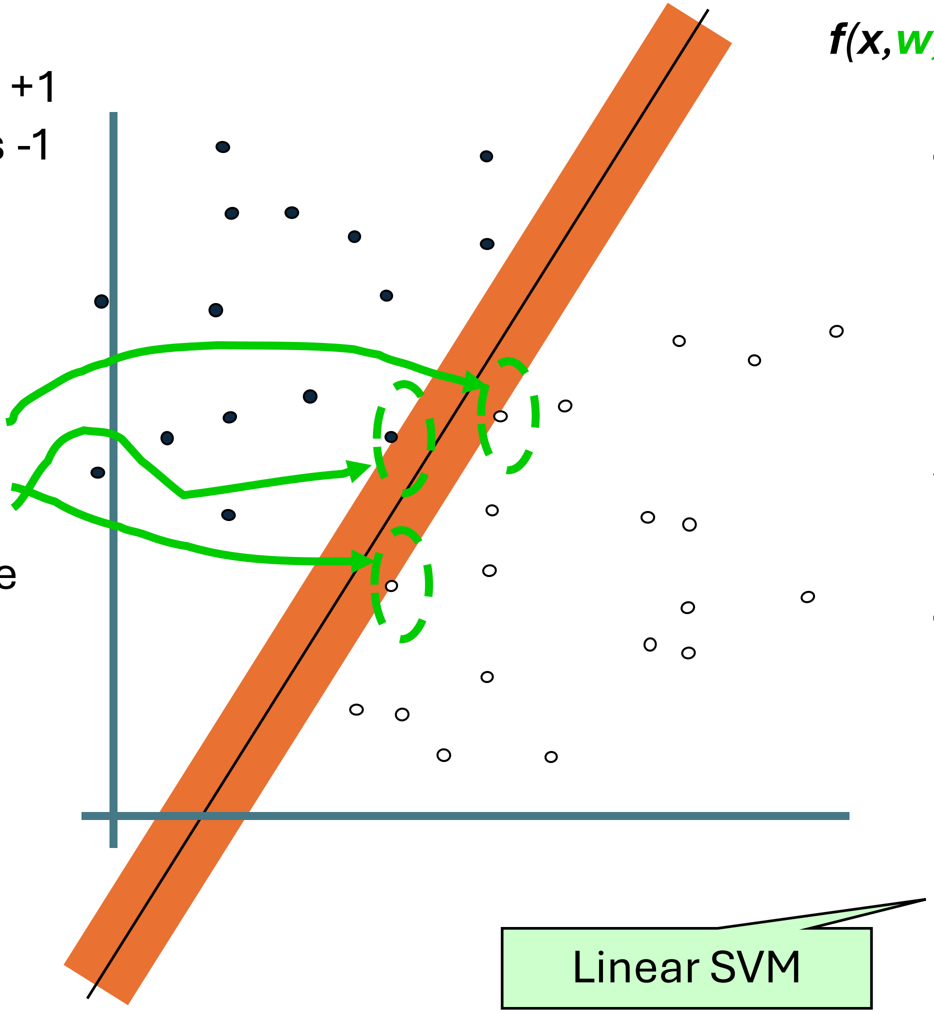
Maximum Margin



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

● denotes +1

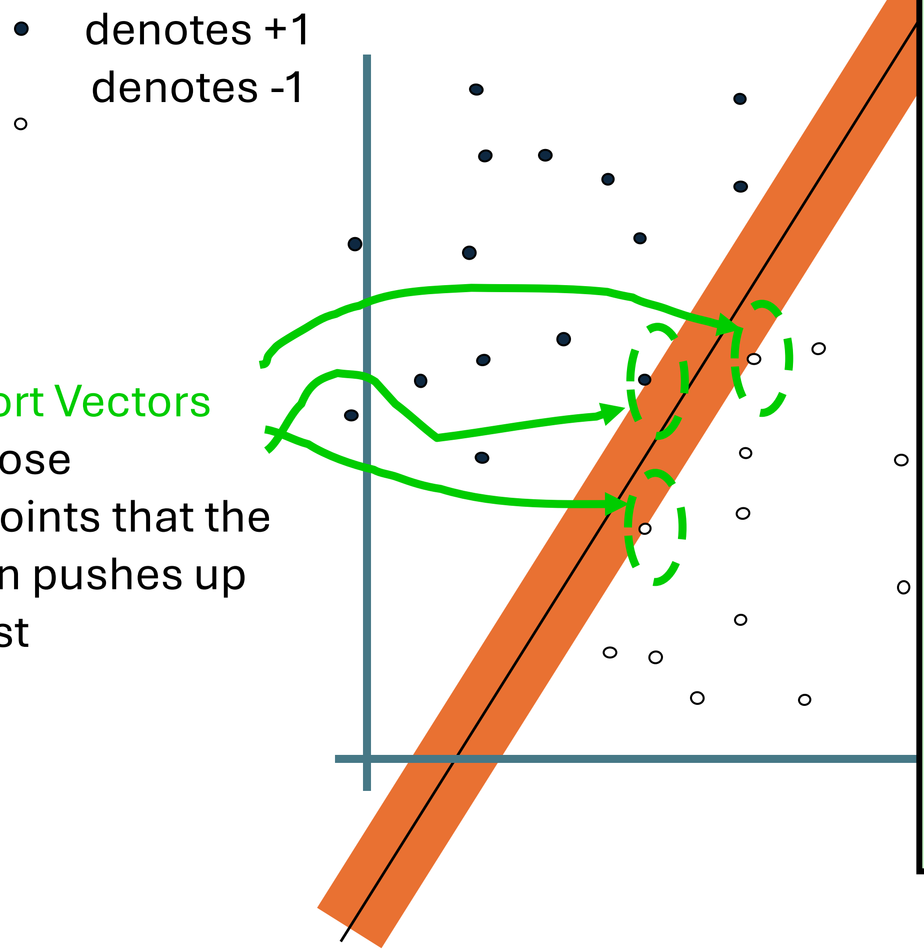
○ denotes -1



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)

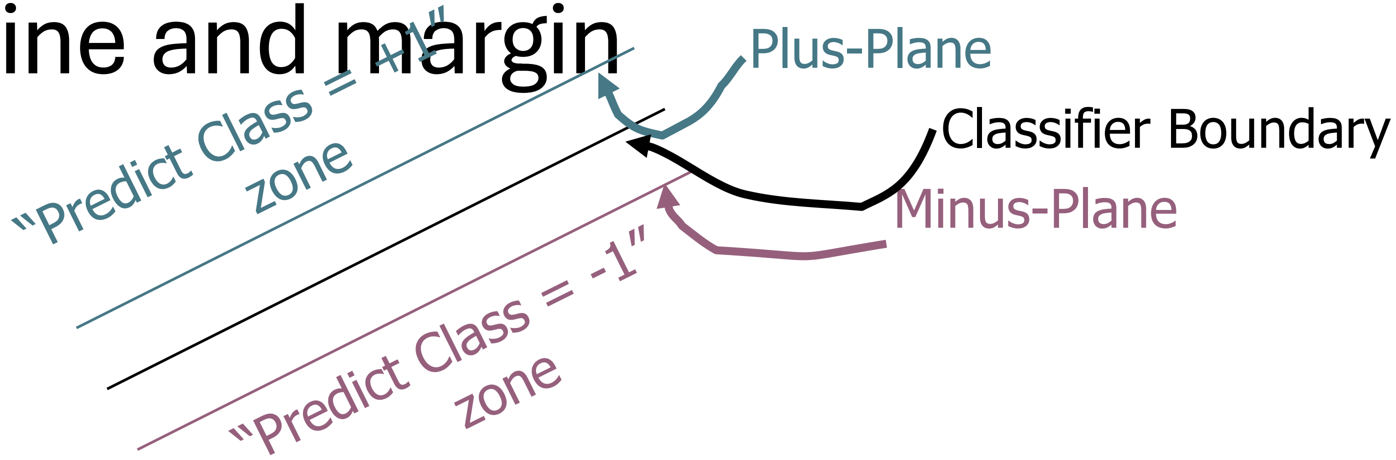
Why Maximum Margin?



Support Vectors are those datapoints that the margin pushes up against

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. LOOCV is easy since the model is immune to removal of any non-support-vector datapoints.
4. There's some theory (using VC dimension) that is related to (but not the same as) the proposition that this is a good thing.
5. Empirically it works very very well.

Specifying a line and margin



"Predict Class = +1" zone

Plus-Plane

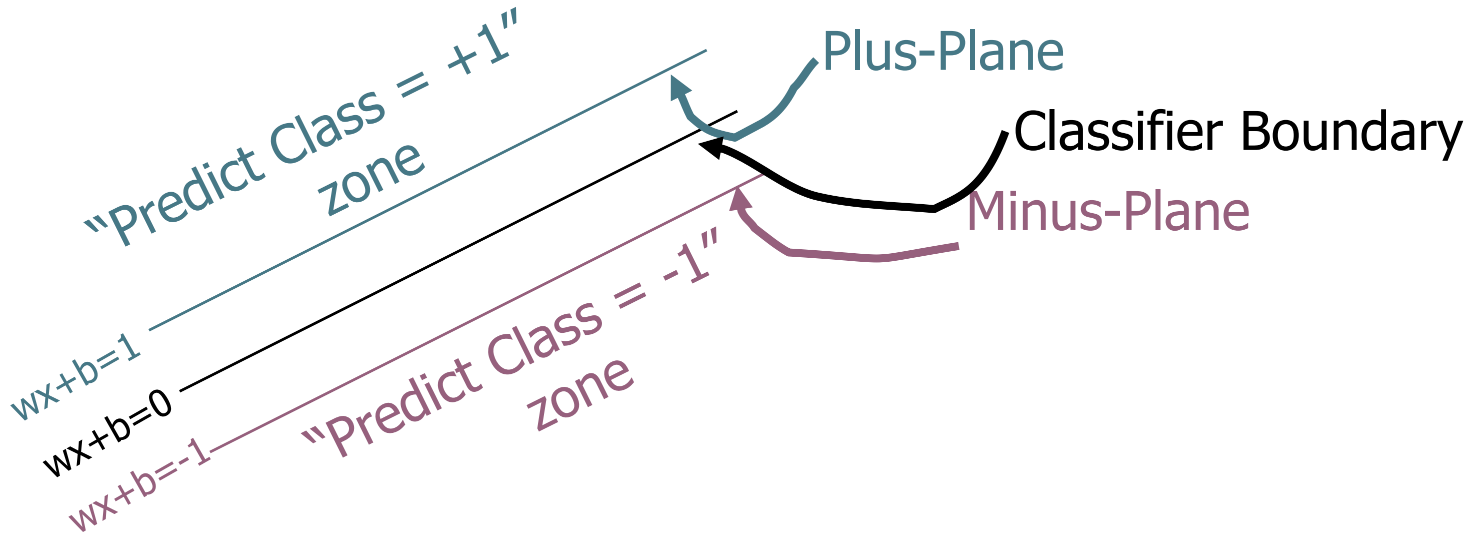
Classifier Boundary

Minus-Plane

"Predict Class = -1" zone

- How do we represent this mathematically?
- ...in *m* input dimensions?

Specifying a line and margin



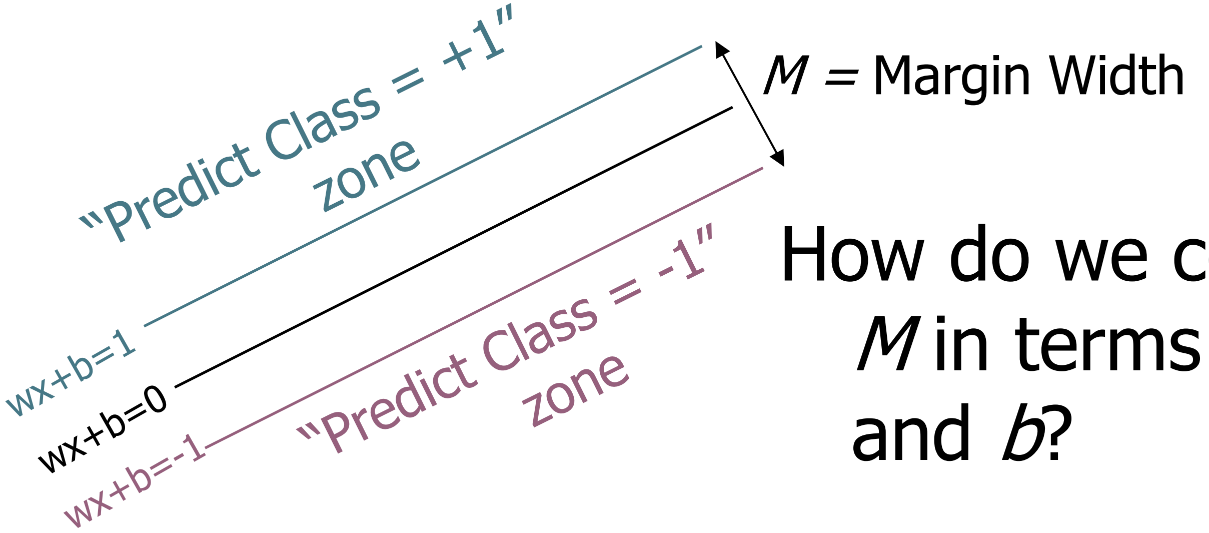
- Plus-plane = $\{ \mathbf{x} : \mathbf{w} \,.\, \mathbf{x} + b = +1 \}$
- Minus-plane = $\{ \mathbf{x} : \mathbf{w} \,.\, \mathbf{x} + b = -1 \}$

Classify as.. +1 if $\mathbf{w} \,.\, \mathbf{x} + b >= 1$

-1 if $\mathbf{w} \,.\, \mathbf{x} + b <= -1$

Universe explodes if $-1 < \mathbf{w} \,.\, \mathbf{x} + b < 1$

Computing the margin width

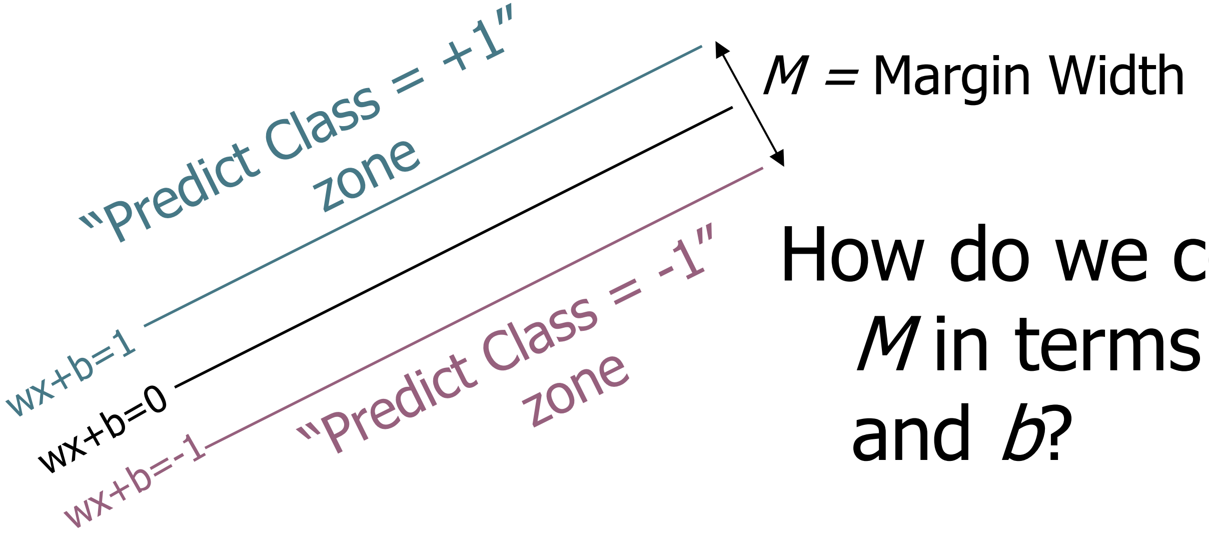


How do we compute *M* in terms of ***w*** and *b*?

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

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

Computing the margin width



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

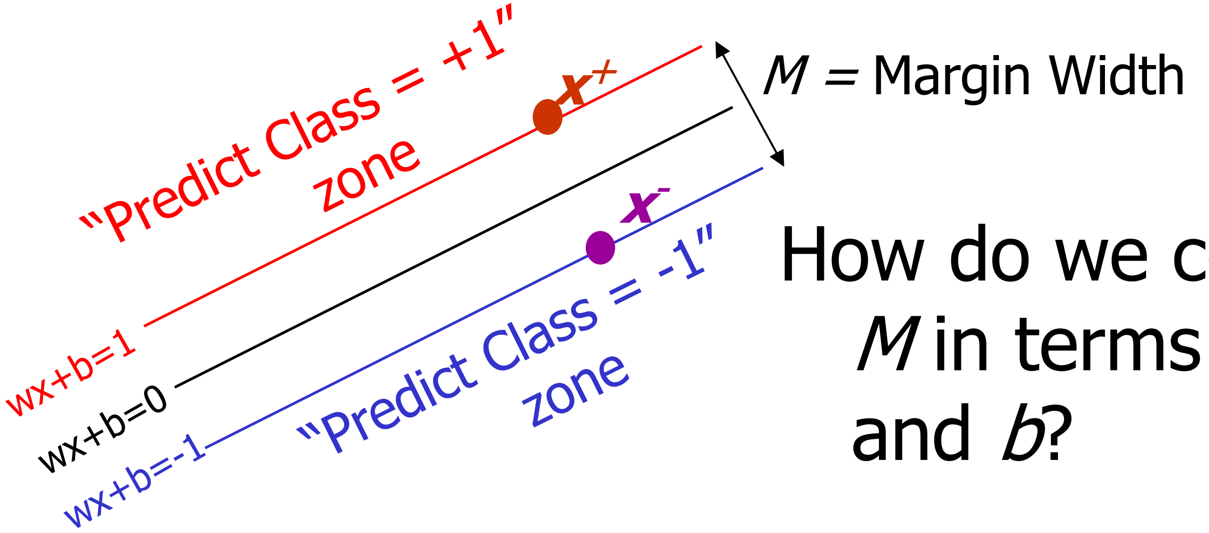
- Plus-plane = $\{ \boldsymbol{x} : \boldsymbol{w} \,.\, \boldsymbol{x} + b = +1 \}$
- Minus-plane = $\{ \boldsymbol{x} : \boldsymbol{w} \,.\, \boldsymbol{x} + b = -1 \}$

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

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

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

Computing the margin width

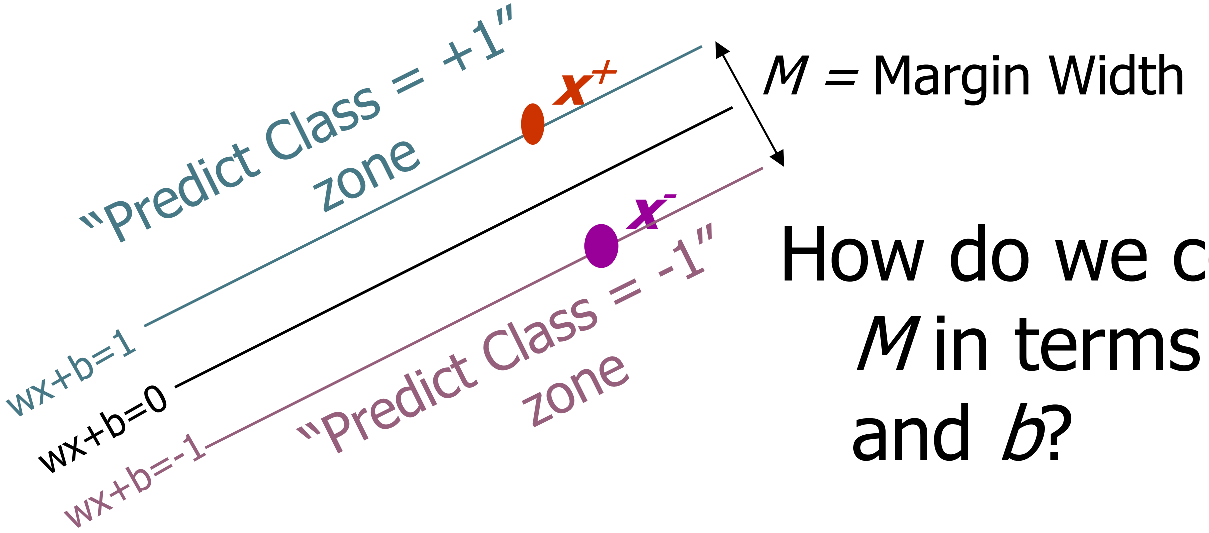


How do we compute M in terms of $\mathbf{w}$ and b?

- Plus-plane = $\{ \mathbf{x} : \mathbf{w} \cdot \mathbf{x} + b = +1 \}$
- Minus-plane = $\{ \mathbf{x} : \mathbf{w} \cdot \mathbf{x} + b = -1 \}$
- The vector **w** is perpendicular to the Plus Plane
- Let $\mathbf{x}^-$ be any point on the minus plane
- Let $\mathbf{x}^+$ be the closest plus-plane-point to $\mathbf{x}^-$.

Any location in R^m: not necessarily a datapoint

Computing the margin width

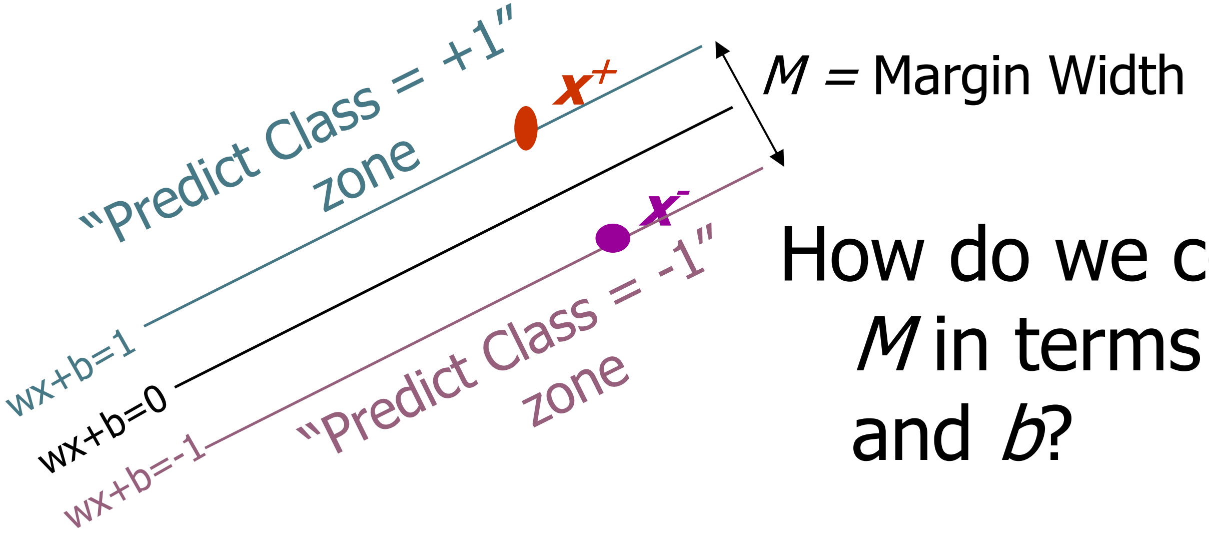


How do we compute M in terms of $\boldsymbol{w}$ and b?

- Plus-plane = $\{ \mathbf{x} : \mathbf{w} . \mathbf{x} + b = +1 \}$
- Minus-plane = $\{ \mathbf{x} : \mathbf{w} . \mathbf{x} + b = -1 \}$
- The vector **w** is perpendicular to the Plus Plane
- Let $\boldsymbol{x}^-$ be any point on the minus plane
- Let $\boldsymbol{x}^+$ be the closest plus-plane-point to $\boldsymbol{x}^-$.

Any location in R^m: not necessarily a datapoint

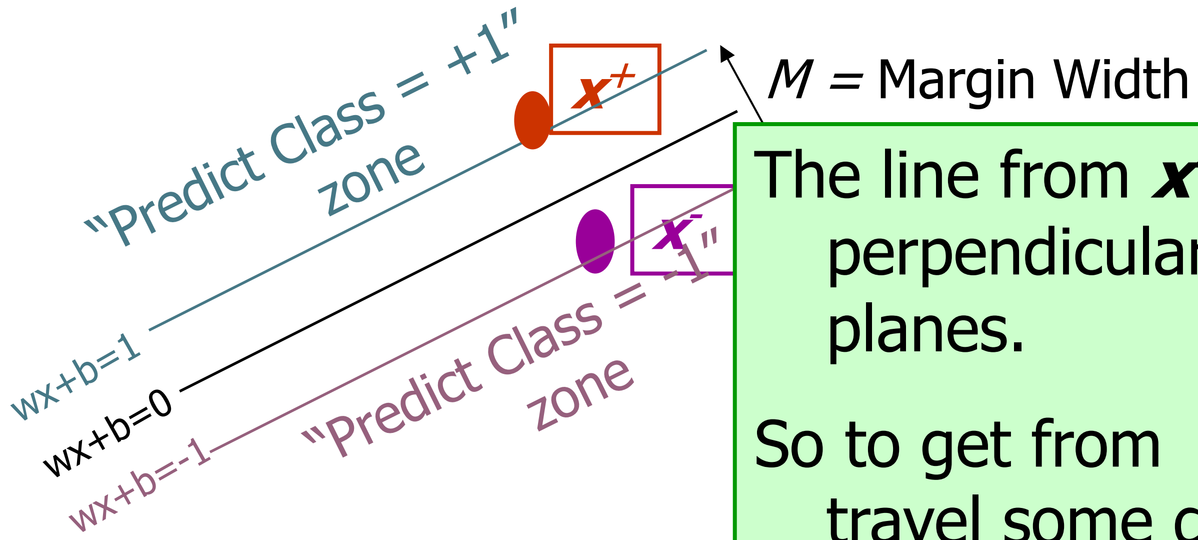
Computing the margin width



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

- Plus-plane = $\{ \mathbf{x} : \mathbf{w} \,.\, \mathbf{x} + b = +1 \}$
- Minus-plane = $\{ \mathbf{x} : \mathbf{w} \,.\, \mathbf{x} + b = -1 \}$
- The vector **w** is perpendicular to the Plus Plane
- Let $\boldsymbol{x}^-$ be any point on the minus plane
- Let $\boldsymbol{x}^+$ be the closest plus-plane-point to $\boldsymbol{x}^-$.
- Claim: $\boldsymbol{x}^+ = \boldsymbol{x}^- + \lambda \boldsymbol{w}$ for some value of λ. Why?

Computing the margin width

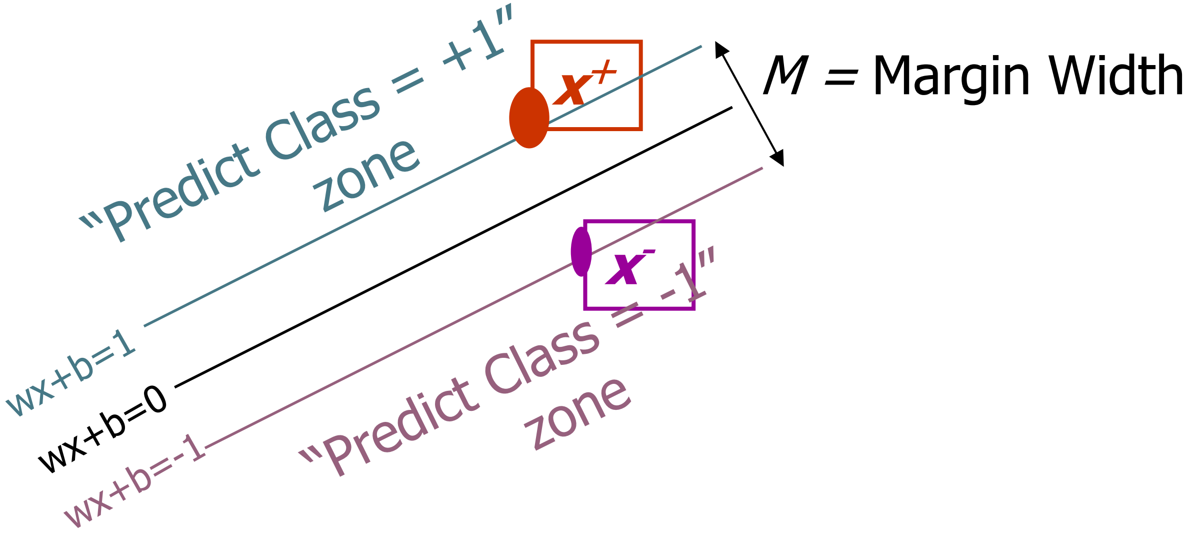


The line from $\boldsymbol{x}^-$ to $\boldsymbol{x}^+$ is perpendicular to the planes.

So to get from $\boldsymbol{x}^-$ to $\boldsymbol{x}^+$ travel some distance in direction $\boldsymbol{w}$.

- Plus-plane = $\{ \mathbf{x} : \mathbf{w} \,.\, \mathbf{x} + b = +1 \}$
- Minus-plane = $\{ \mathbf{x} : \mathbf{w} \,.\, \mathbf{x} + b = -1 \}$
- The vector **w** is perpendicular to the Plus Plane
- Let $\boldsymbol{x}^-$ be any point on the minus plane
- Let $\boldsymbol{x}^+$ be the closest plus-plane-point to $\boldsymbol{x}^-$.
- Claim: $\boldsymbol{x}^+ = \boldsymbol{x}^- + \lambda\, \boldsymbol{w}$ for some value of λ. Why?

Computing the margin width

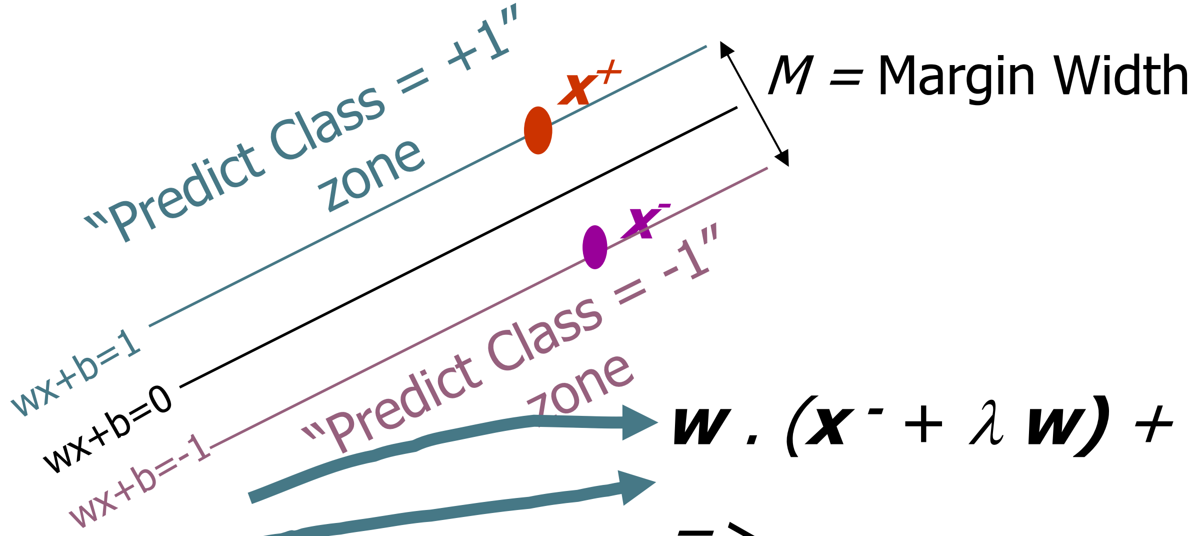


What we know:

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

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

Computing the margin width



What we know:

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

It's now easy to get *M* in terms of ***w*** and *b*

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

=>

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

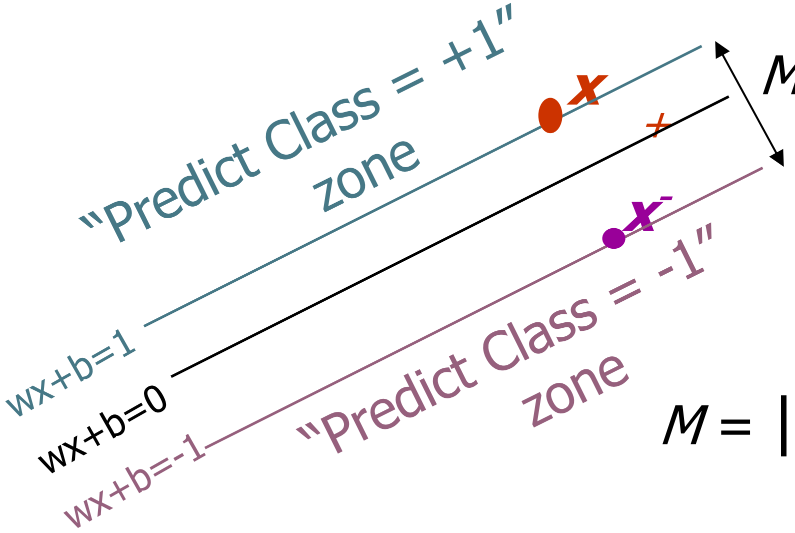
=>

$$-1 + \lambda \mathbf{w} \,.\, \mathbf{w} = 1$$

=>

$$\lambda = \frac{2}{\mathbf{w.w}}$$

Computing the margin width



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

What we know:

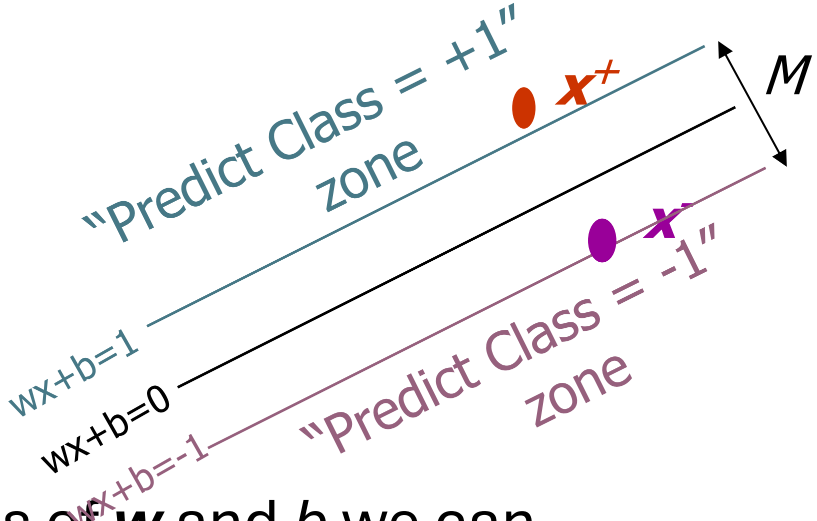
- $\boldsymbol{w} \,.\, \boldsymbol{x}^{+} + b = +1$
- $\boldsymbol{w} \,.\, \boldsymbol{x}^{-} + b = -1$
- $\boldsymbol{x}^{+} = \boldsymbol{x}^{-} + \lambda \boldsymbol{w}$
- $|\boldsymbol{x}^{+} - \boldsymbol{x}^{-}| = M$
- $$\lambda = \frac{2}{\mathbf{w.w}}$$

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

$$= \lambda |\mathbf{w}| = \lambda \sqrt{\mathbf{w.w}}$$

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

Learning the Maximum Margin Classifier



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

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

So now we just need 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?

Learning via Quadratic Programming

- QP is a well-studied class of optimization algorithms to maximize a quadratic function of some real-valued variables subject to linear constraints.

Quadratic Programming

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

Quadratic criterion

Subject to

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

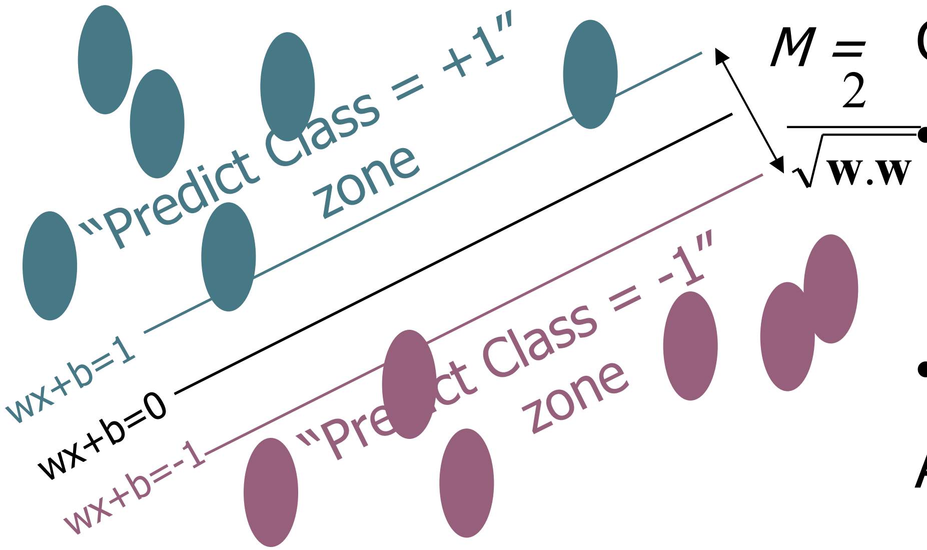
n additional linear inequality constraints

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 equality constraints

Learning the Maximum Margin Classifier


"Predict Class = +1" zone
"Predict Class = -1" zone
wx+b=1
wx+b=0
wx+b=-1
$M = \dfrac{2}{\sqrt{\mathbf{w}.\mathbf{w}}}$

Given guess of ***w*** , *b* we can

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

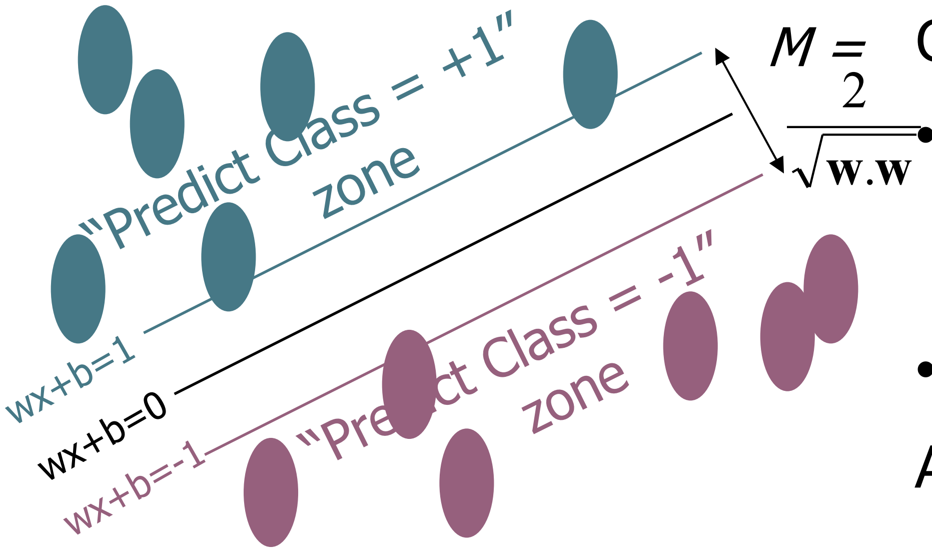
Assume *R* datapoints, each ($\mathbf{x}_k$,y_k) where y_k = +/- 1

What should our quadratic optimization criterion be?

How many constraints will we have?

What should they be?

Learning the Maximum Margin Classifier



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

Given guess of ***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 y_k = +/- *1*

What should our quadratic optimization criterion be?

Minimize ***w.w***

How many constraints will we have? *R*

What should they be?

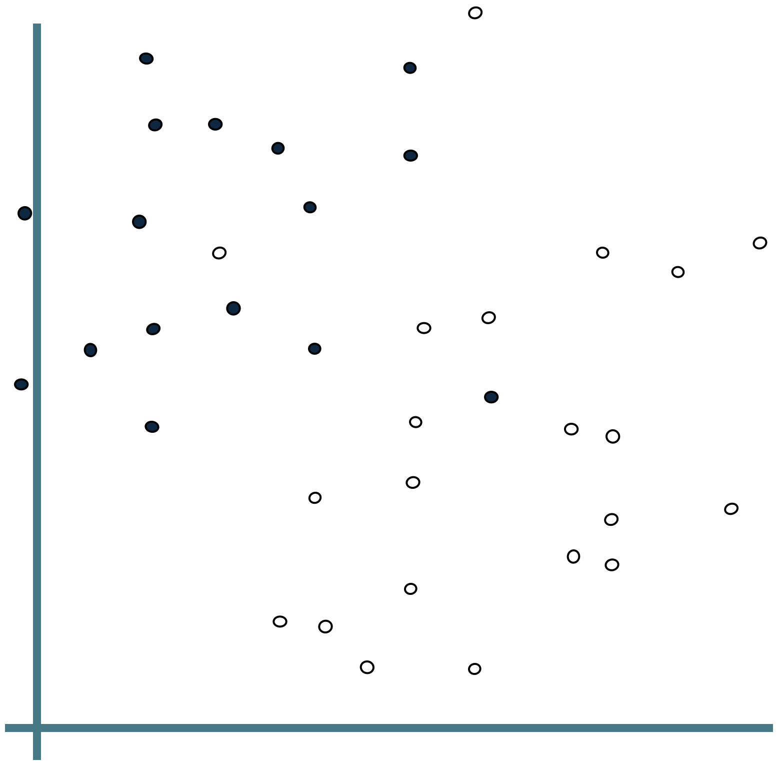
$\boldsymbol{w} \,.\, \boldsymbol{x}_k + b >= 1 \text{ if } y_k = 1$

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

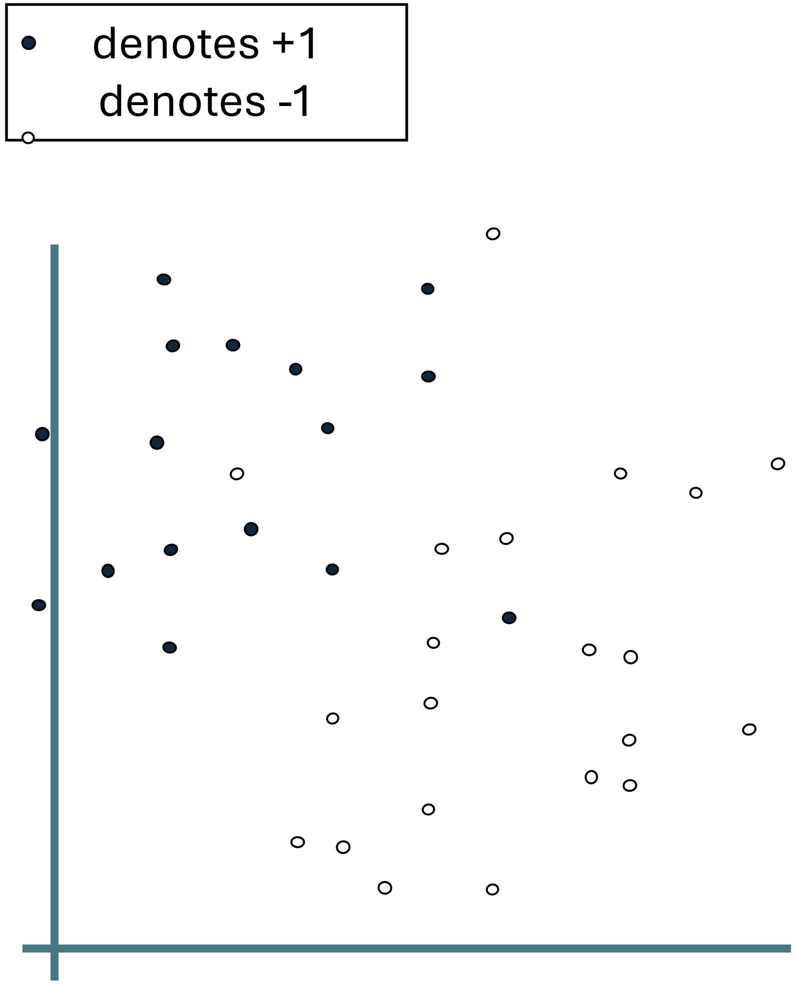
Uh-oh!

This is going to be a problem!
What should we do?

● denotes +1
○ denotes -1



Uh-oh!



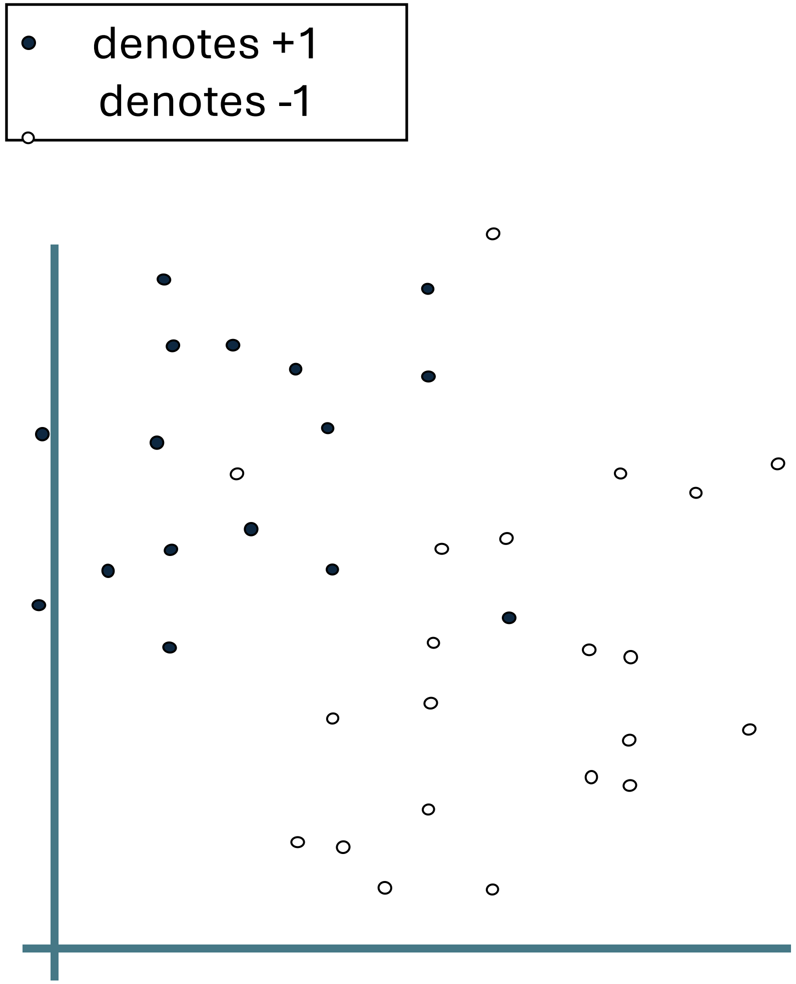
This is going to be a problem!
What should we do?

Idea 1:

Find minimum ***w.w***, while minimizing number of training set errors.

Problemette: Two things to minimize makes for an ill-defined optimization

Uh-oh!



This is going to be a problem!

What should we do?

Idea 1.1:

Minimize

$\mathbf{w}.\mathbf{w} + C\ (\#train\ errors)$

Tradeoff parameter

There's a serious practical problem that's about to make us reject this approach. Can you guess what it is?

Uh-oh!

This is going to be a problem!

What should we do?

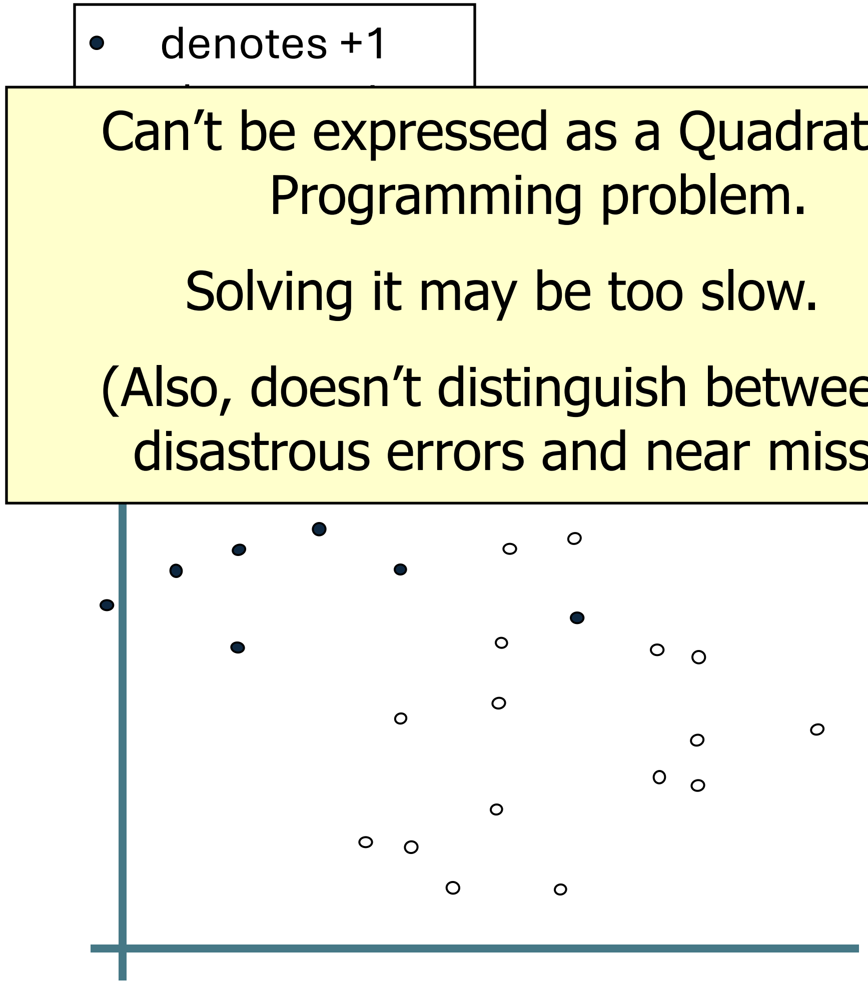
Idea 1.1:

Minimize

w.w + *C (#train errors)*

Tradeoff parameter

ere’s a serious practical blem that’s about to make us reject this approach. Can you guess what it is?



Can’t be expressed as a Quadratic Programming problem.

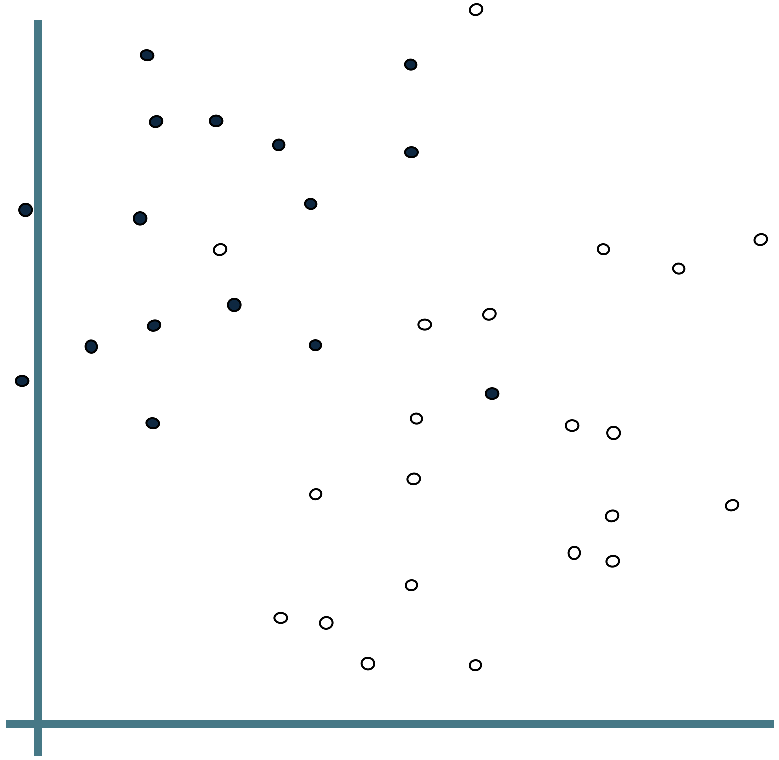
Solving it may be too slow.

(Also, doesn’t distinguish between disastrous errors and near misses)

Uh-oh!

● denotes +1

○ denotes -1



This is going to be a problem!

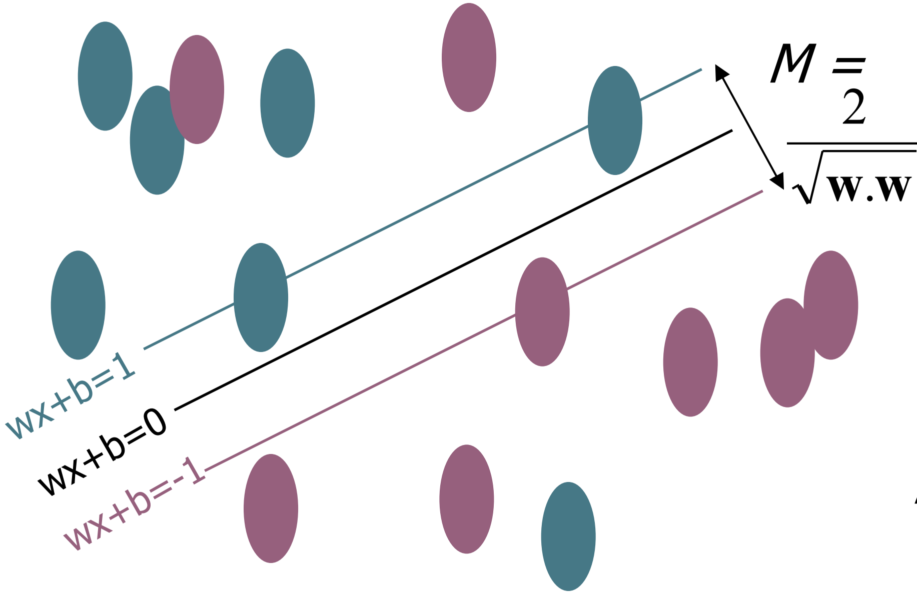
What should we do?

Idea 2.0:

Minimize

w.w + *C (distance of error points to their correct place)*

Learning Maximum Margin with Noise



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

Given guess of ***w*** , *b* we can

- Compute sum of distances of points to their correct zones
- Compute the margin width

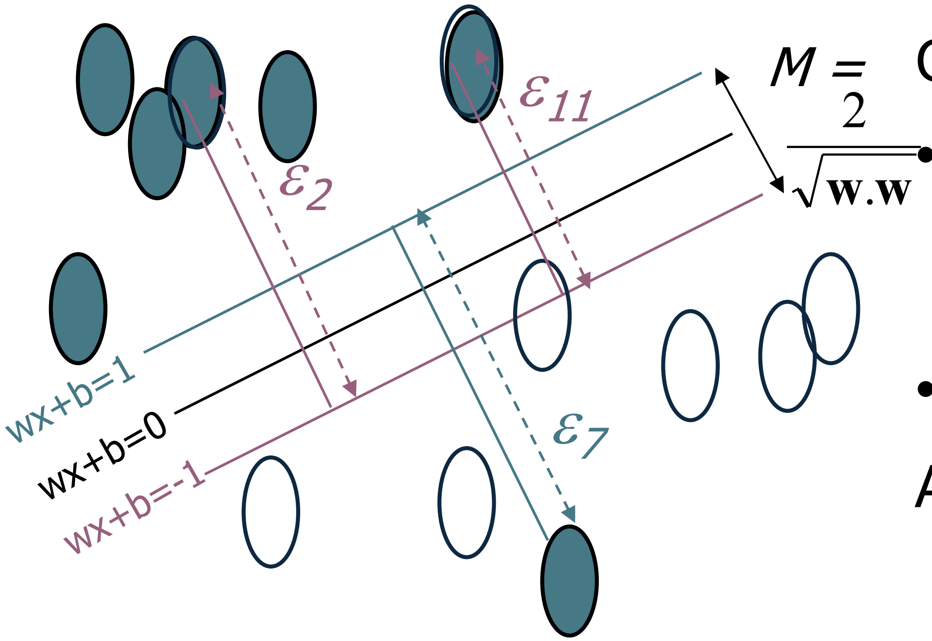
Assume *R* datapoints, each $(\mathbf{x}_k, y_k)$ where $y_k = +/- 1$

What should our quadratic optimization criterion be?

How many constraints will we have?

What should they be?

Learning Maximum Margin with Noise



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

Given guess of ***w*** , *b* we can

- Compute sum of distances of points to their correct zones
- Compute the margin width

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

What should our quadratic optimization criterion be?

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

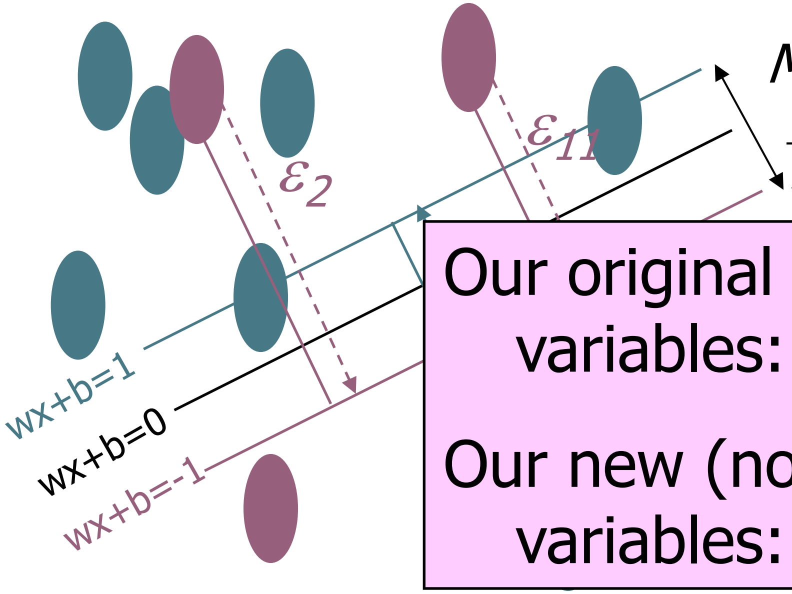
How many constraints will we have? *R*

What should they be?

$\mathbf{w} \,.\, \mathbf{x}_k + b >= 1-\varepsilon_k$ *if* $y_k = 1$

$\mathbf{w} \,.\, \mathbf{x}_k + b <= -1+\varepsilon_k$ *if* $y_k = -1$

Learning Maximum Margin with Noise



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

Given gu e can

- Compute sum of stances of points to their correct

n

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

What should our quadratic optimization criterion be?

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

How many constraints will we have? *R*

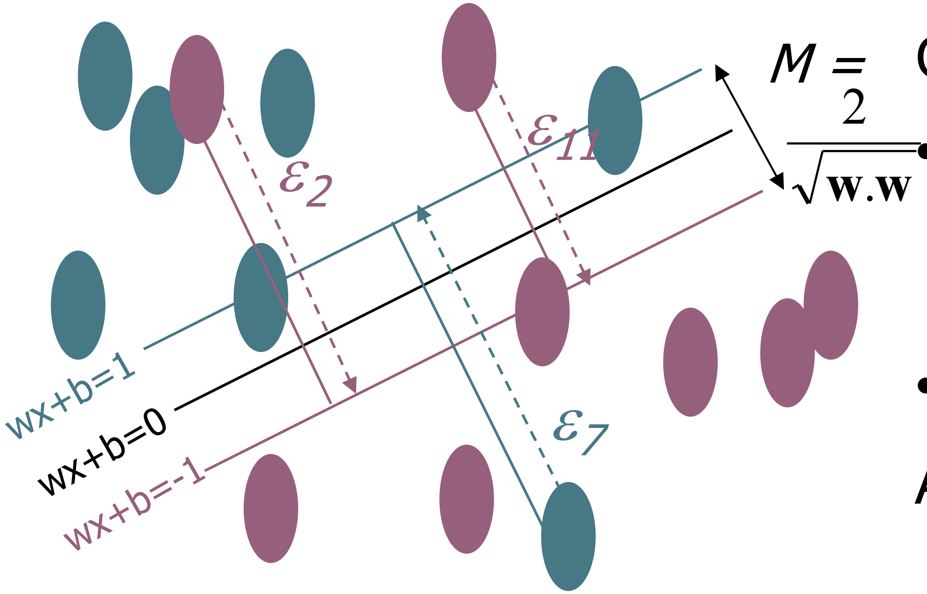
R= # records

What should they be?

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

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

Learning Maximum Margin with Noise



Given guess of ***w*** , *b* we can

- Compute sum of distances of points to their correct zones
- Compute the margin width

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

What should our quadratic optimization criterion be?

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

How many constraints will we have? *R*

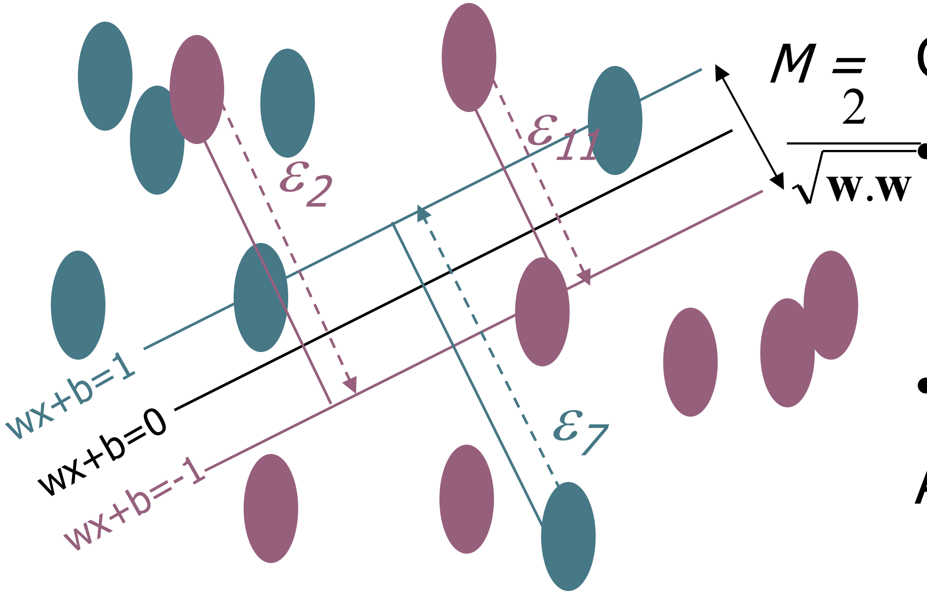
What should they be?

$\mathbf{w} \,.\, \mathbf{x}_k + b >= 1-\varepsilon_k$ *if* $y_k = 1$

$\mathbf{w} \,.\, \mathbf{x}_k + b <= -1+\varepsilon_k$ *if* $y_k = -1$

There's a bug in this QP. Can you spot it?

Learning Maximum Margin with Noise



Given guess of ***w*** , *b* we can

- Compute sum of distances of points to their correct zones
- Compute the margin width

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

What should our quadratic optimization criterion be?

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

How many constraints will we have? *2R*

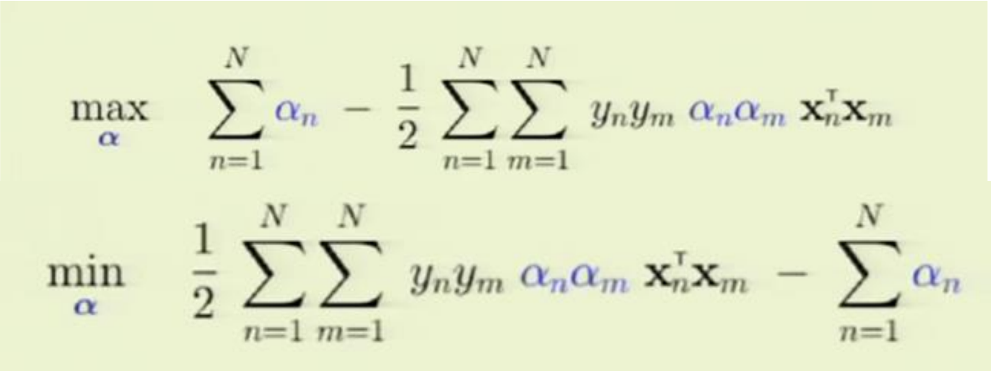
What should they be?

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

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

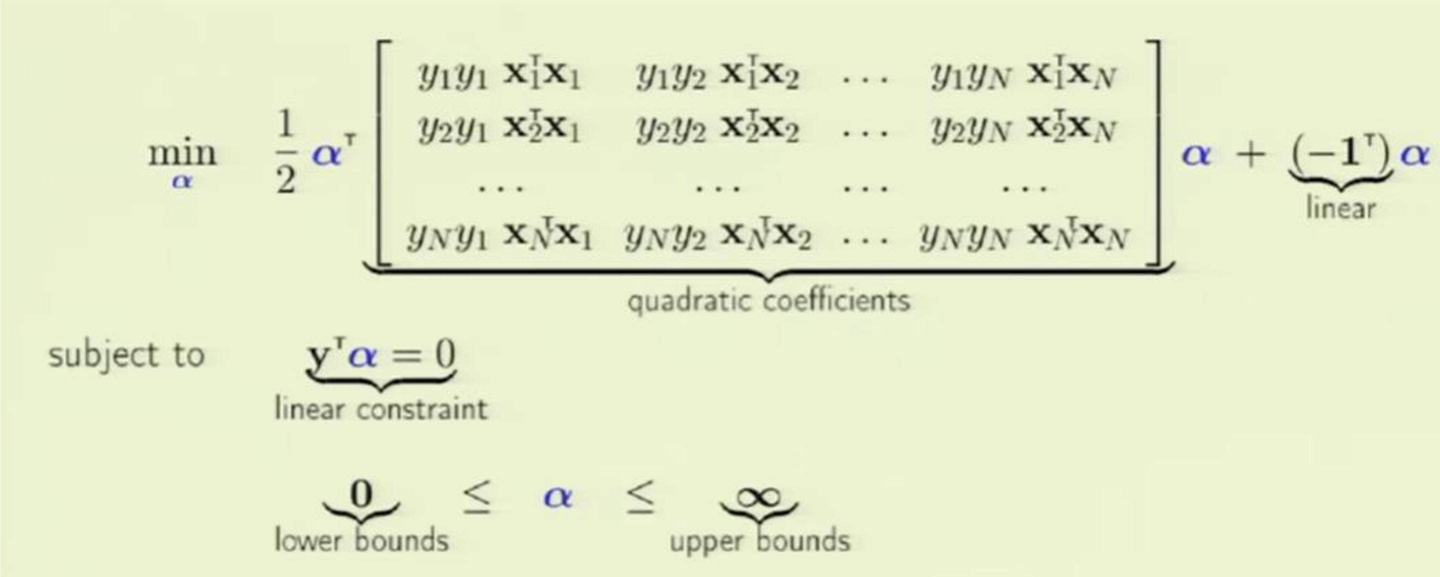
$\varepsilon_k >= 0$ for all *k*

QP Problems Nature



$$\max_{\boldsymbol{\alpha}} \quad \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} y_n y_m \, \alpha_n \alpha_m \, \mathbf{x}_n^{\intercal} \mathbf{x}_m$$

$$\min_{\boldsymbol{\alpha}} \quad \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} y_n y_m \, \alpha_n \alpha_m \, \mathbf{x}_n^{\intercal} \mathbf{x}_m - \sum_{n=1}^{N} \alpha_n$$



$$\min_{\boldsymbol{\alpha}} \quad \frac{1}{2} \boldsymbol{\alpha}^{\intercal} \underbrace{\begin{bmatrix} y_1 y_1 \, \mathbf{x}_1^{\intercal} \mathbf{x}_1 & y_1 y_2 \, \mathbf{x}_1^{\intercal} \mathbf{x}_2 & \dots & y_1 y_N \, \mathbf{x}_1^{\intercal} \mathbf{x}_N \\ y_2 y_1 \, \mathbf{x}_2^{\intercal} \mathbf{x}_1 & y_2 y_2 \, \mathbf{x}_2^{\intercal} \mathbf{x}_2 & \dots & y_2 y_N \, \mathbf{x}_2^{\intercal} \mathbf{x}_N \\ \dots & \dots & \dots & \dots \\ y_N y_1 \, \mathbf{x}_N^{\intercal} \mathbf{x}_1 & y_N y_2 \, \mathbf{x}_N^{\intercal} \mathbf{x}_2 & \dots & y_N y_N \, \mathbf{x}_N^{\intercal} \mathbf{x}_N \end{bmatrix}}_{\text{quadratic coefficients}} \boldsymbol{\alpha} + \underbrace{(-\mathbf{1}^{\intercal})}_{\text{linear}} \boldsymbol{\alpha}$$

subject to $\underbrace{\mathbf{y}^{\intercal} \boldsymbol{\alpha} = 0}_{\text{linear constraint}}$

$$\underbrace{\mathbf{0}}_{\text{lower bounds}} \leq \boldsymbol{\alpha} \leq \underbrace{\infty}_{\text{upper bounds}}$$

Solving the Optimization Problem

Find **w** and b such that

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

$\mathbf{\Phi}(\mathbf{w}) = \mathbf{w}^{\mathrm{T}}\mathbf{w}$ is minimized

and for all $(\mathbf{x}_i, y_i)$, $i=1..n$: $y_i\,(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b) \geq 1$

- Need to optimize a *quadratic* function subject to *linear* constraints.
- Quadratic optimization problems are a well-known class of mathematical programming problems for which several (non-trivial) algorithms exist.
- The solution involves constructing a *dual problem* where a *Lagrange multiplier* α_i is associated with every inequality constraint in the primal (original) problem:

Find $\alpha_1 \ldots \alpha_n$ such that

$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x}_i^{\mathrm{T}}\mathbf{x}_j$ is maximized and

(1) $\Sigma\alpha_i y_i = 0$

(2) $\alpha_i \geq 0$ for all α_i

The Optimization Problem Solution

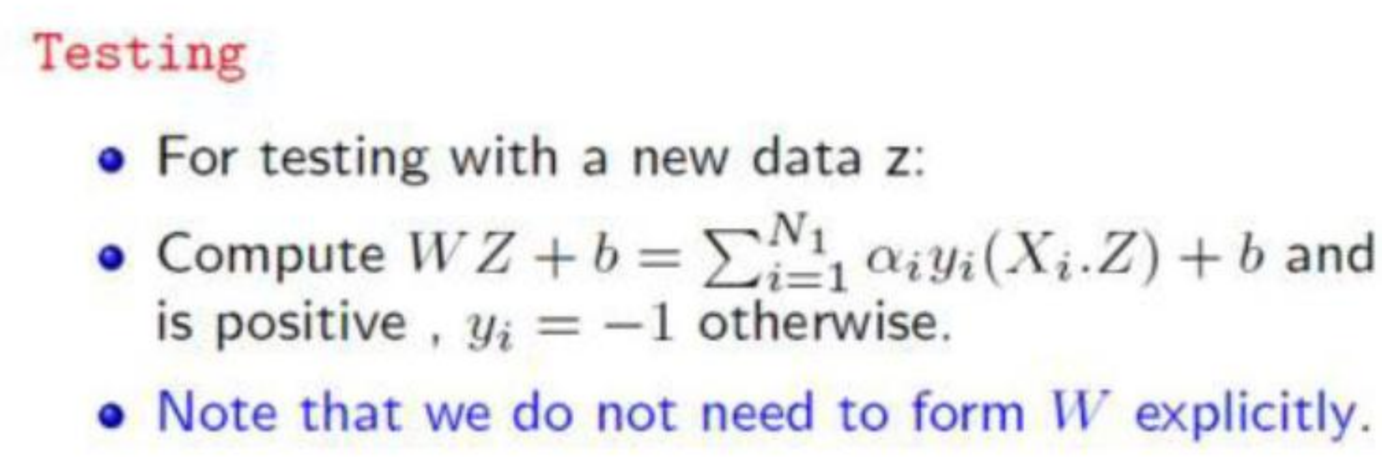
- Given a solution $\alpha_1 \ldots \alpha_n$ to the dual problem, solution to the primal is:

$$\boxed{\mathbf{w} = \Sigma\alpha_i y_i \mathbf{x}_i \qquad b = y_k - \Sigma\alpha_i y_i \mathbf{x}_i^{\mathbf{T}}\mathbf{x}_k \quad \text{for any } \alpha_k > 0}$$

- Each non-zero α_i indicates that corresponding $\mathbf{x}_i$ is a support vector.
- Then the classifying function is (note that we don't need **w** explicitly):

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

- 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 $\mathbf{x}_i^{\mathbf{T}}\mathbf{x}_j$ between all training points.



Testing

- For testing with a new data z:
- Compute $WZ + b = \sum_{i=1}^{N_1} \alpha_i y_i (X_i.Z) + b$ and classify Z as $y_i = +1$ if the sum is positive , $y_i = -1$ otherwise.
- Note that we do not need to form W explicitly.

Example

- Suppose we are given the following positively labeled data points in $\Re^2$:

$$\begin{pmatrix}3\\1\end{pmatrix}, \begin{pmatrix}3\\-1\end{pmatrix}, \begin{pmatrix}6\\1\end{pmatrix}, \begin{pmatrix}6\\-1\end{pmatrix}$$

and

- the following negatively labeled data points in $\Re^2$:

$$\begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}0\\-1\end{pmatrix}, \begin{pmatrix}-1\\0\end{pmatrix}$$

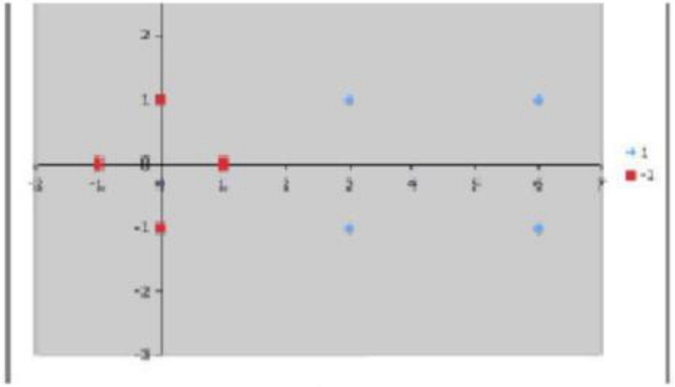


Figure 1: Sample data points in $\Re^2$. Blue diamonds are positive examples red squares are negative examples.

Lets define simple SVM that accurately discriminates the two classes. Since the data is linearly separable, we can use a linear SVM. it should be obvious that there are three support vectors:

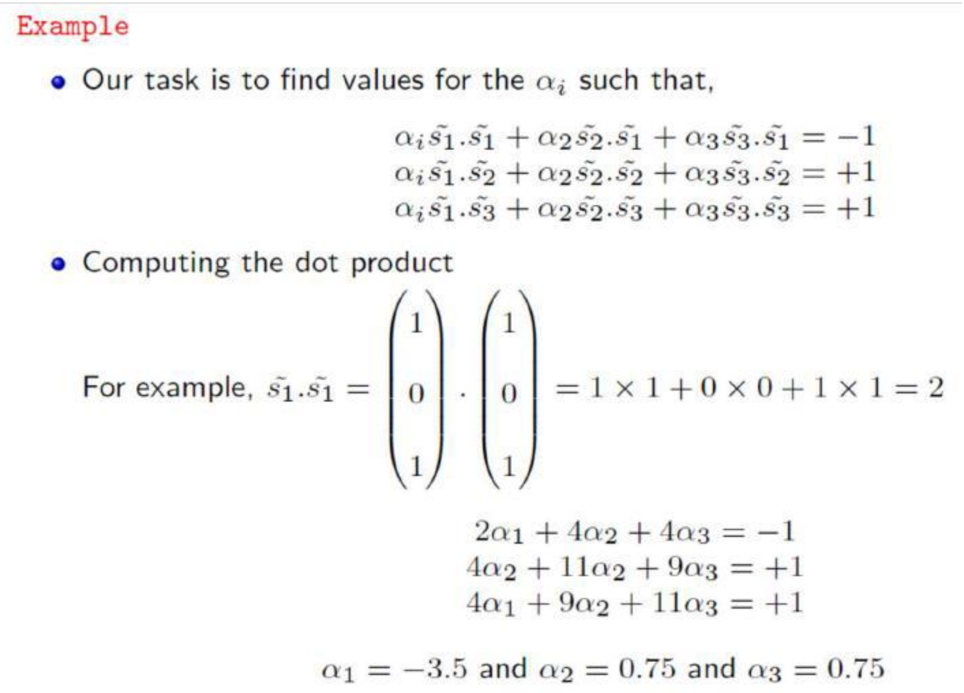
$$s_1 = \begin{pmatrix}1\\0\end{pmatrix}, s_2 = \begin{pmatrix}3\\1\end{pmatrix}, \; s_3 = \begin{pmatrix}3\\-1\end{pmatrix}$$

- We will use vectors augmented with a 1 as a bias input, and for clarity we will differentiate these with an over-tilde. So, if s1 = (10), then $\tilde{s_1} = (101)$.

$$\tilde{s_1} = \begin{pmatrix}1\\0\\1\end{pmatrix}, \tilde{s_2} = \begin{pmatrix}3\\1\\1\end{pmatrix}, \; \tilde{s_3} = \begin{pmatrix}3\\-1\\1\end{pmatrix}$$

- Our task is to find values for the α_i such that,

$$\alpha_i \tilde{s_1}.\tilde{s_1} + \alpha_2 \tilde{s_2}.\tilde{s_1} + \alpha_3 \tilde{s_3}.\tilde{s_1} = -1$$
$$\alpha_i \tilde{s_1}.\tilde{s_2} + \alpha_2 \tilde{s_2}.\tilde{s_2} + \alpha_3 \tilde{s_3}.\tilde{s_2} = +1$$
$$\alpha_i \tilde{s_1}.\tilde{s_3} + \alpha_2 \tilde{s_2}.\tilde{s_3} + \alpha_3 \tilde{s_3}.\tilde{s_3} = +1$$



Example

- Our task is to find values for the α_i such that,

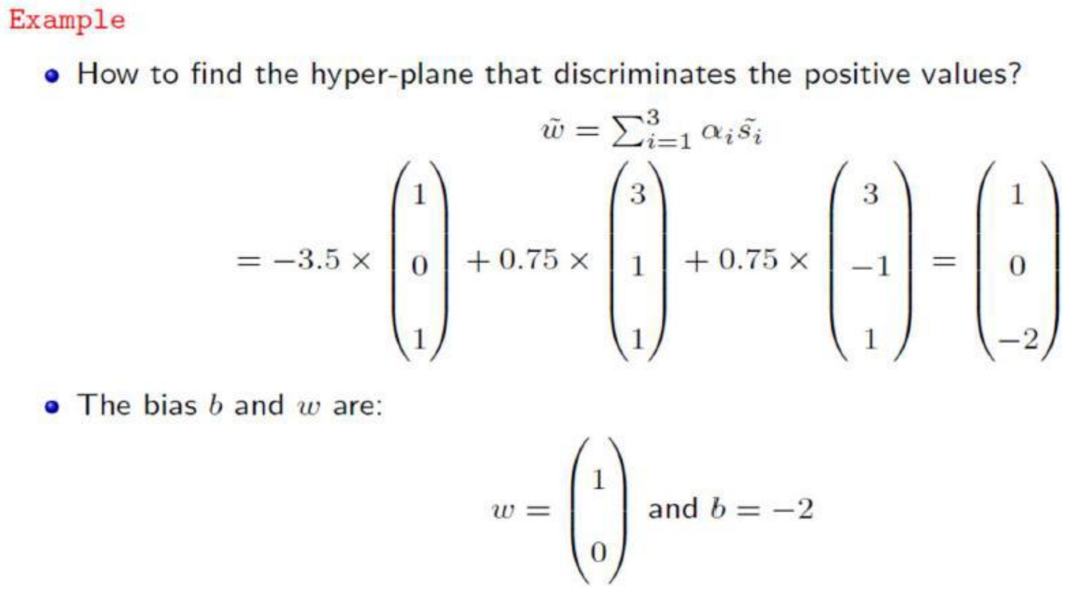
$$\begin{aligned}
\alpha_i \tilde{s}_1.\tilde{s}_1 + \alpha_2 \tilde{s}_2.\tilde{s}_1 + \alpha_3 \tilde{s}_3.\tilde{s}_1 &= -1 \\
\alpha_i \tilde{s}_1.\tilde{s}_2 + \alpha_2 \tilde{s}_2.\tilde{s}_2 + \alpha_3 \tilde{s}_3.\tilde{s}_2 &= +1 \\
\alpha_i \tilde{s}_1.\tilde{s}_3 + \alpha_2 \tilde{s}_2.\tilde{s}_3 + \alpha_3 \tilde{s}_3.\tilde{s}_3 &= +1
\end{aligned}$$

- Computing the dot product

For example, $\tilde{s}_1.\tilde{s}_1 = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} = 1 \times 1 + 0 \times 0 + 1 \times 1 = 2$

$$\begin{aligned}
2\alpha_1 + 4\alpha_2 + 4\alpha_3 &= -1 \\
4\alpha_2 + 11\alpha_2 + 9\alpha_3 &= +1 \\
4\alpha_1 + 9\alpha_2 + 11\alpha_3 &= +1
\end{aligned}$$

$\alpha_1 = -3.5$ and $\alpha_2 = 0.75$ and $\alpha_3 = 0.75$



Example

- How to find the hyper-plane that discriminates the positive values?

$$\tilde{w} = \sum_{i=1}^{3} \alpha_i \tilde{s}_i$$

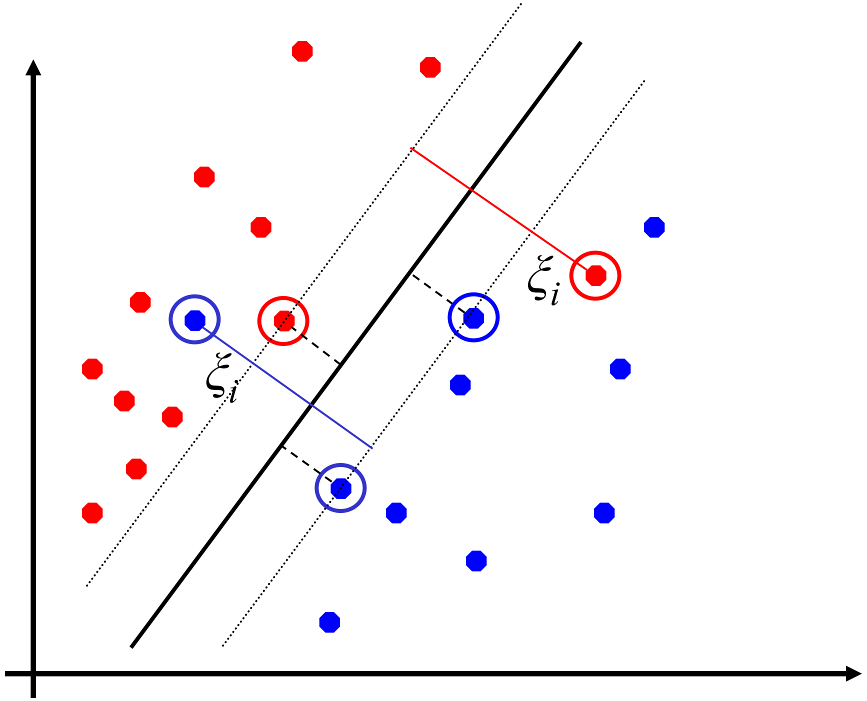
$$= -3.5 \times \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} + 0.75 \times \begin{pmatrix} 3 \\ 1 \\ 1 \end{pmatrix} + 0.75 \times \begin{pmatrix} 3 \\ -1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ -2 \end{pmatrix}$$

- The bias b and w are:

$$w = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \text{ and } b = -2$$

Soft Margin Classification

- What if the training set is not linearly separable?
- *Slack variables* ξ_i can be added to allow misclassification of difficult or noisy examples, resulting margin called *soft*.



Soft Margin Classification Mathematically

- The old formulation:

Find **w** and b such that
$\mathbf{\Phi}(\mathbf{w}) = \mathbf{w}^{\mathrm{T}}\mathbf{w}$ is minimized
and for all $(\mathbf{x}_i, y_i)$, $i=1..n$: $y_i (\mathbf{w}^{\mathbf{T}}\mathbf{x}_i + b) \geq 1$

- Modified formulation incorporates slack variables:

Find **w** and b such that
$\mathbf{\Phi}(\mathbf{w}) = \mathbf{w}^{\mathrm{T}}\mathbf{w} + C\Sigma\xi_i$ is minimized
and for all $(\mathbf{x}_i, y_i)$, $i=1..n$: $y_i (\mathbf{w}^{\mathbf{T}}\mathbf{x}_i + b) \geq 1 - \xi_{i,}$, $\xi_i \geq 0$

- Parameter C can be viewed as a way to control overfitting: it “trades off” the relative importance of maximizing the margin and fitting the training data.

Soft Margin Classification – Solution

- Dual problem is identical to separable case (would *not* be identical if the 2-norm penalty for slack variables $C\Sigma\xi_i^2$ was used in primal objective, we would need additional Lagrange multipliers for slack variables):

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

- Again, $\mathbf{x}_i$ with non-zero α_i will be support vectors.
- Solution to the dual problem is:

$\mathbf{w} = \Sigma\alpha_i y_i \mathbf{x}_i$
$b = y_k(1 - \xi_k) - \Sigma\alpha_i y_i \mathbf{x}_i^{\mathbf{T}}\mathbf{x}_k$ for any k s.t. $\alpha_k > 0$

Again, we don't need to compute **w** explicitly for classification:

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