Support Vector Machines

Note to other teachers and users of these slides. Andrew would be delighted if you found this source material useful in giving your own lectures. Feel free to use these slides verbatim, or to modify them to fit your own needs. PowerPoint originals are available. If you make use of a significant portion of these slides in your own lecture, please include this message, or the following link to the source repository of Andrew's tutorials: http://www.cs.cmu.edu/~awm/tutorials . Comments and corrections gratefully received.

Andrew W. Moore

Professor

School of Computer Science

Carnegie Mellon University

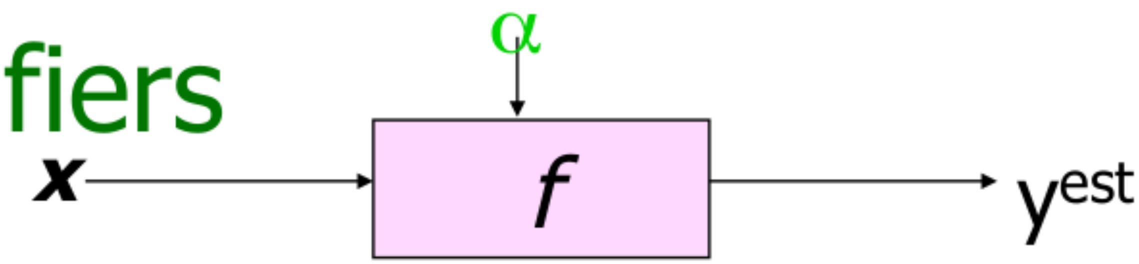
www.cs.cmu.edu/~awm

awm@cs.cmu.edu

412-268-7599

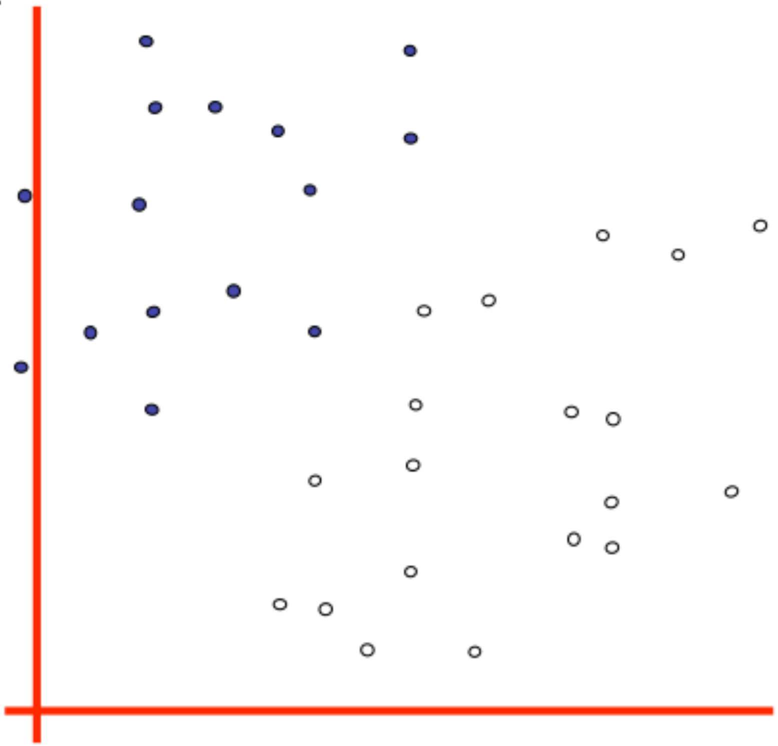
Nov 23rd, 2001

Linear Classifiers



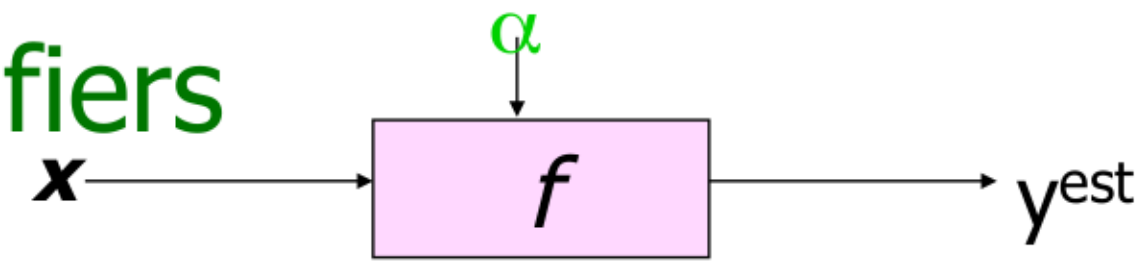
$$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} + b)$$

- • denotes +1
- ○ denotes -1



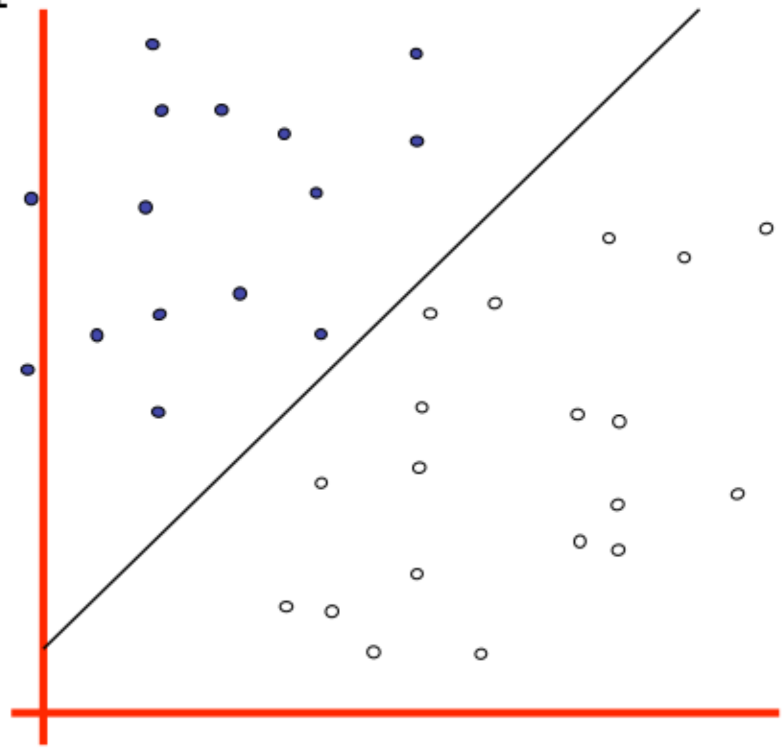
How would you classify this data?

Linear Classifiers



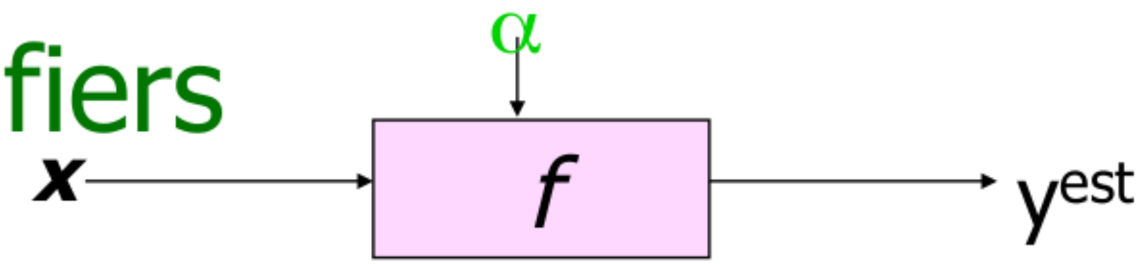
$$f(\mathbf{x}, \mathbf{w}, b) = \mathrm{sign}(\mathbf{w} \cdot \mathbf{x} + b)$$

- denotes +1
- denotes -1

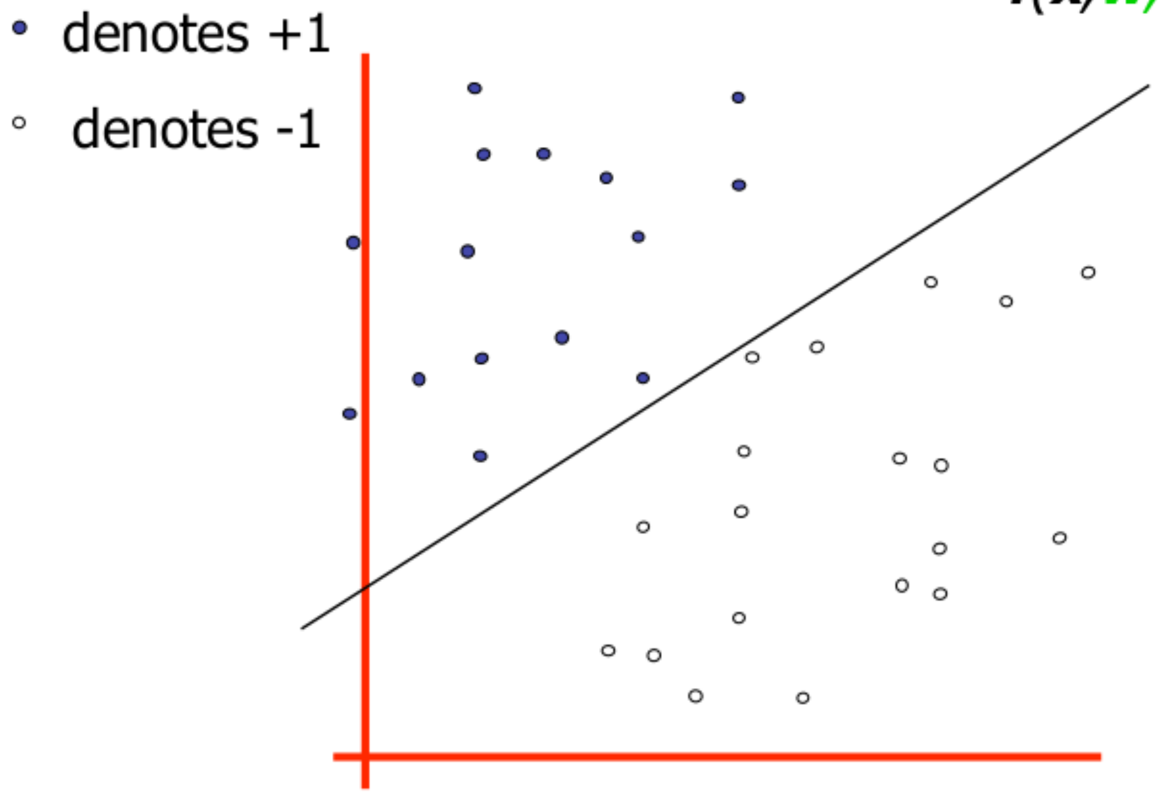


How would you classify this data?

Linear Classifiers

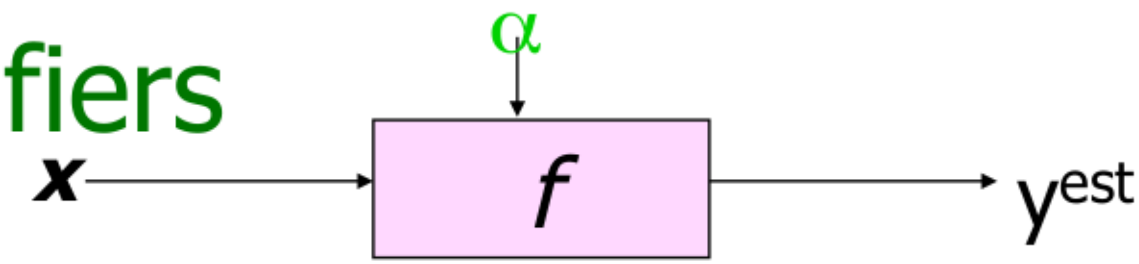


$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} + b)$



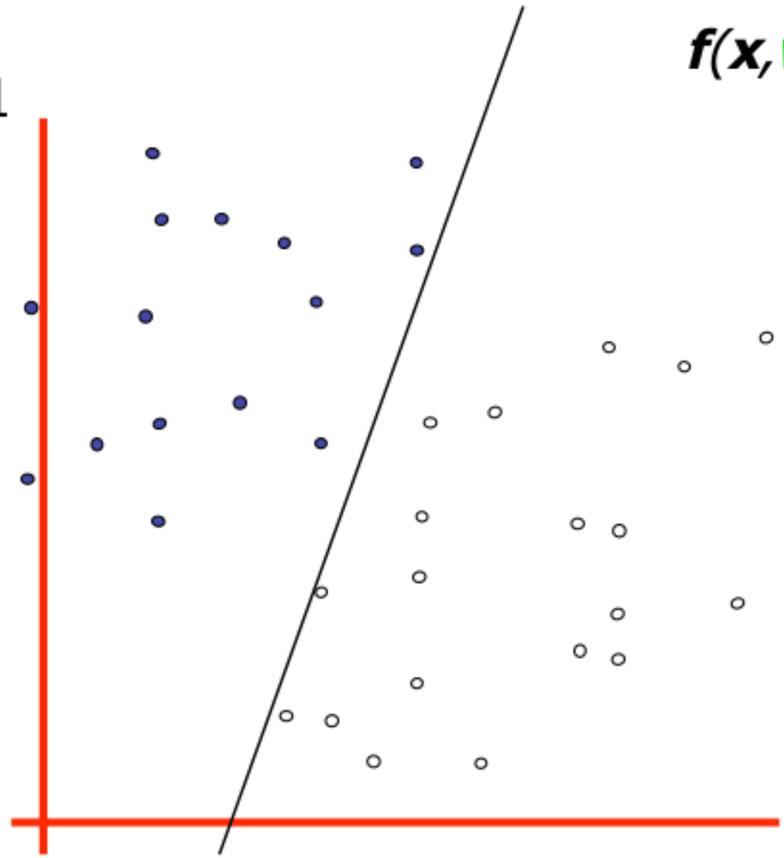
How would you classify this data?

Linear Classifiers



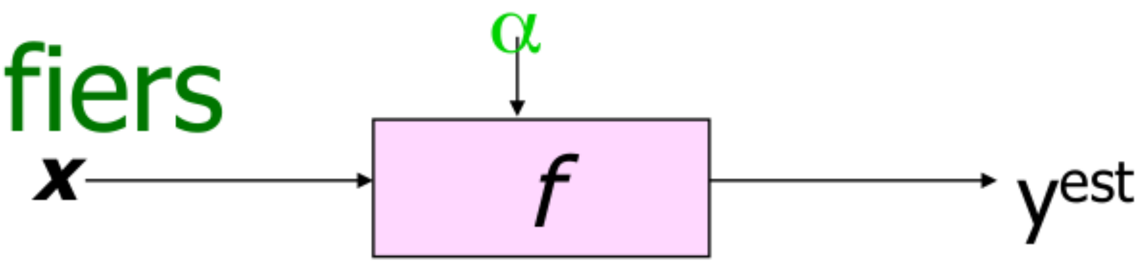
$$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}\cdot\mathbf{x} + b)$$

- • denotes +1
- ° denotes -1

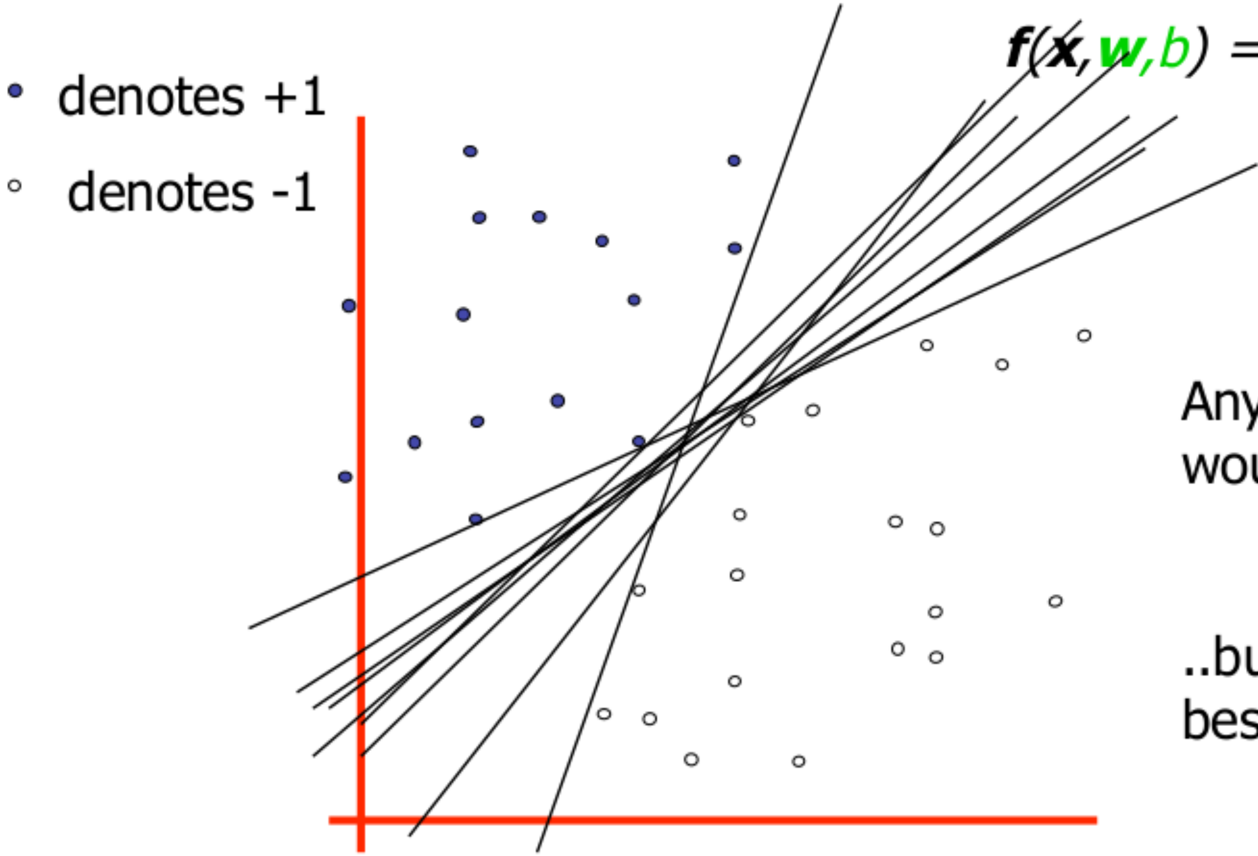


How would you classify this data?

Linear Classifiers



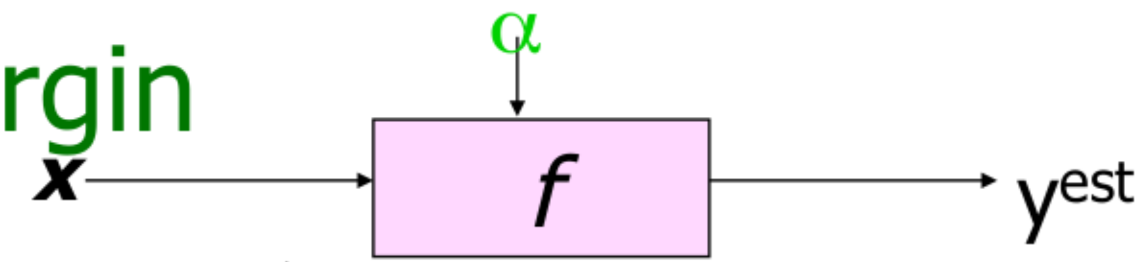
$$f(\mathbf{x}, \mathbf{w}, b) = sign(\mathbf{w} \cdot \mathbf{x} + b)$$



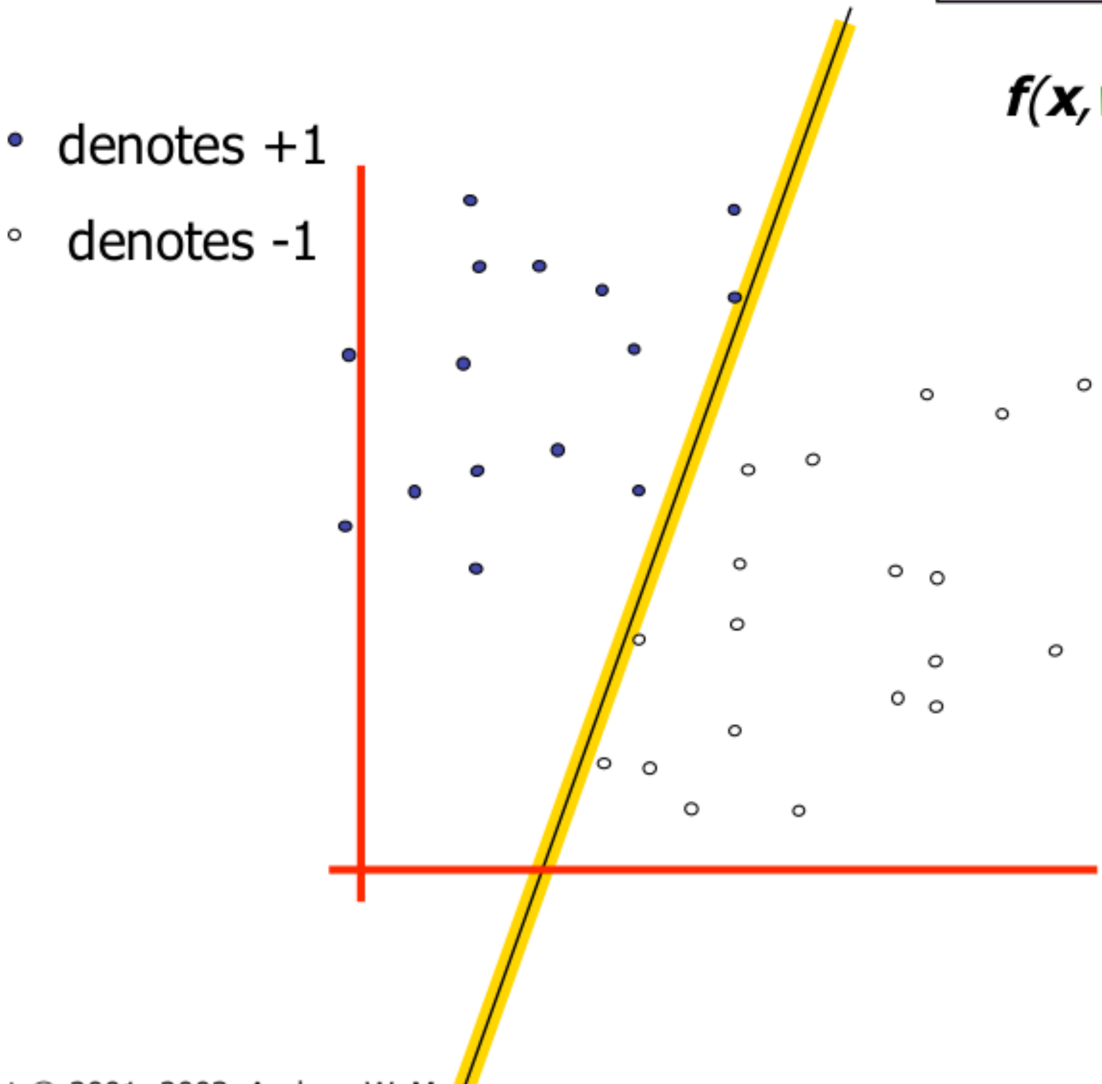
Any of these would be fine..

..but which is best?

Classifier Margin

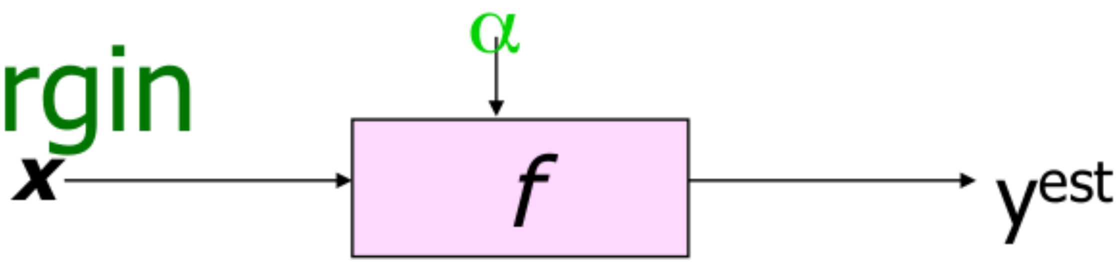


$$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} + b)$$

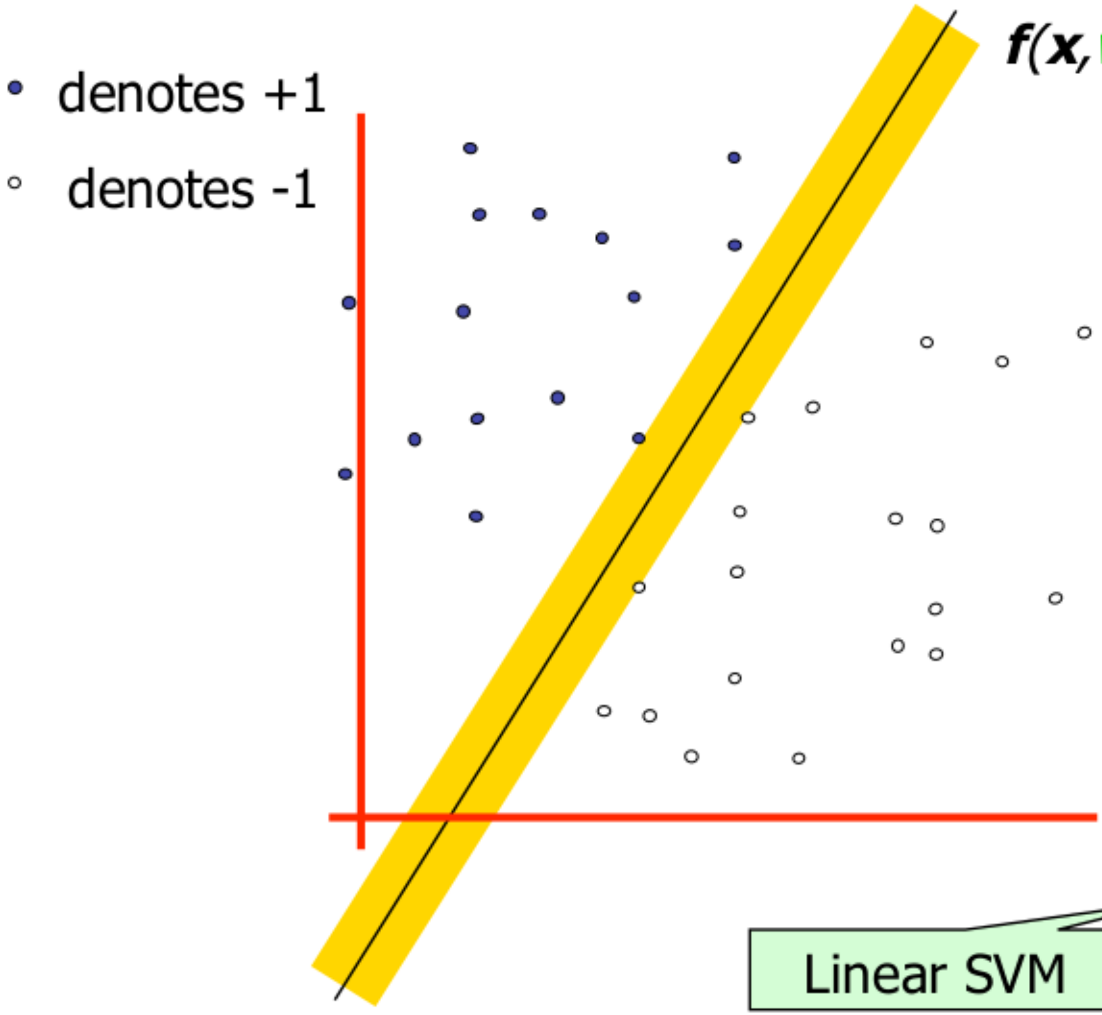


Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

Maximum Margin



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

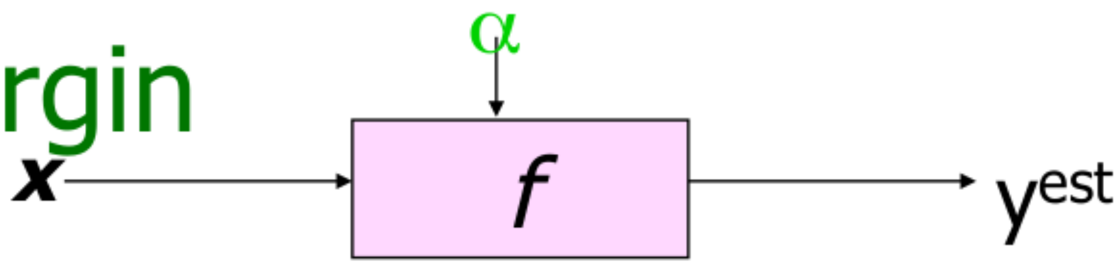


The maximum margin linear classifier is the linear classifier with the, um, maximum margin.

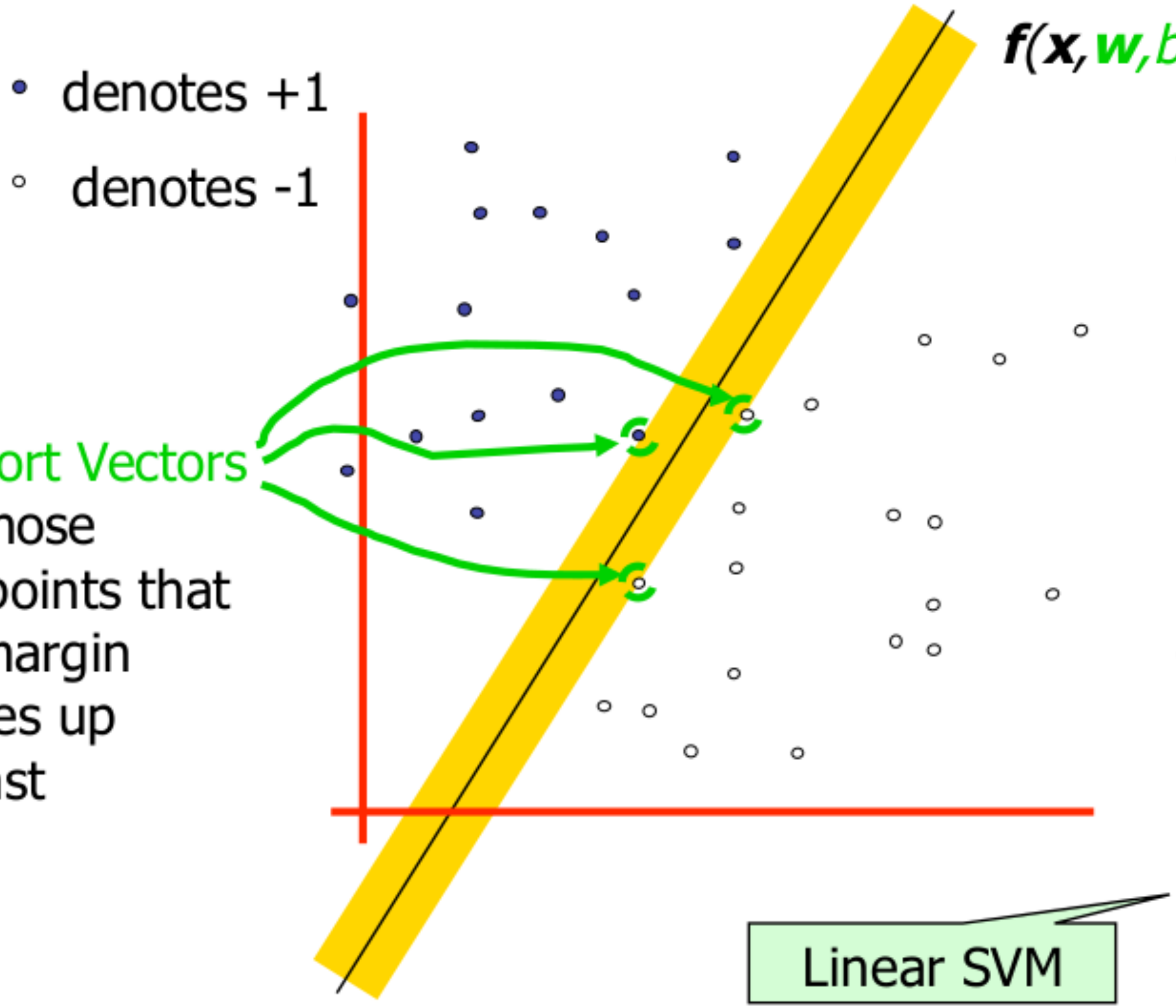
This is the simplest kind of SVM (Called an LSVM)

Linear SVM

Maximum Margin



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

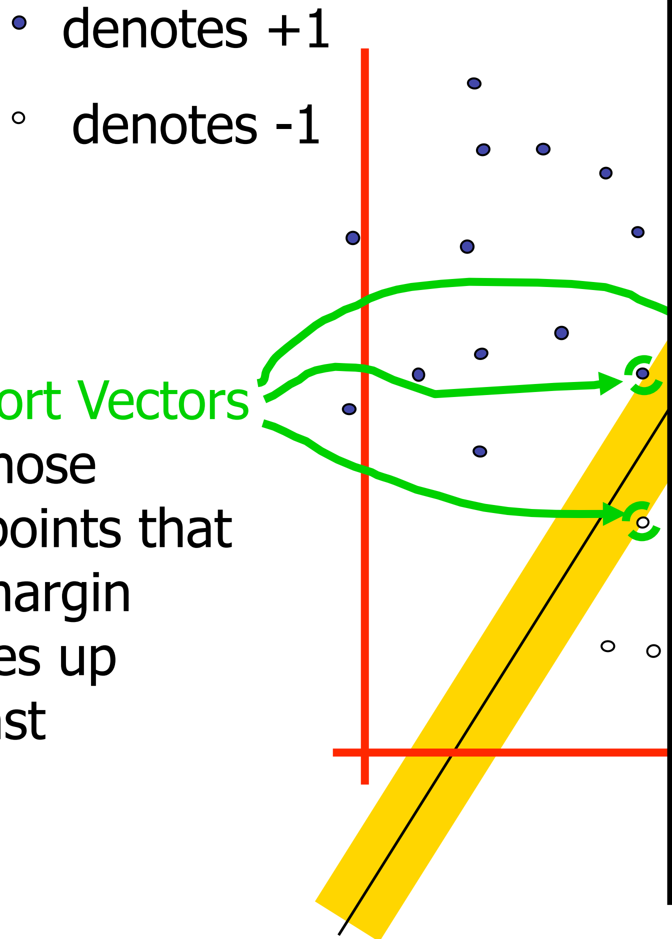


Support Vectors are those datapoints that the margin pushes up against

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

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

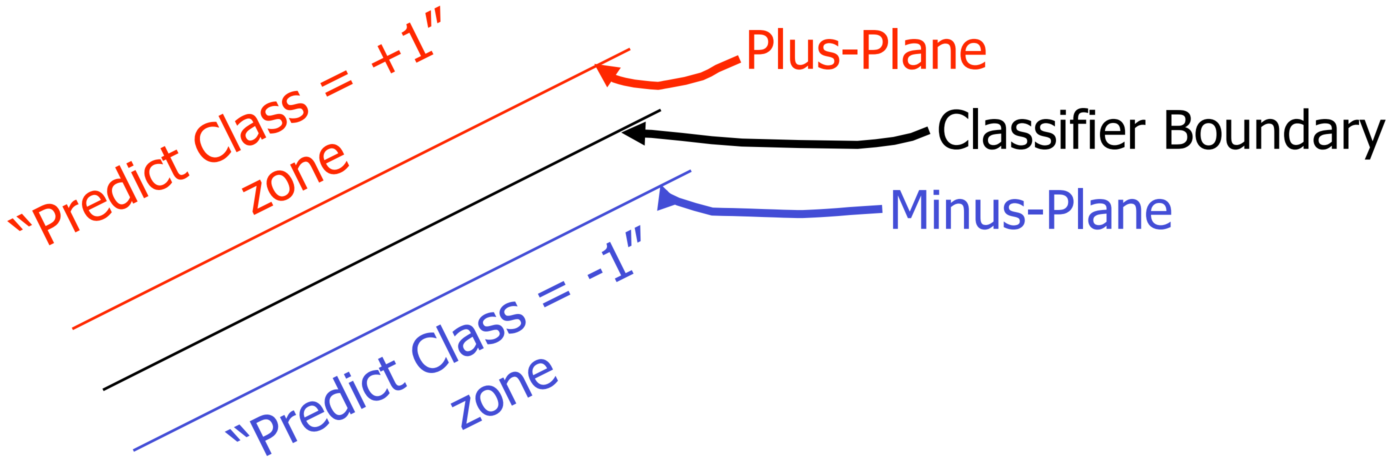
Why Maximum Margin?



Support Vectors are those datapoints that the margin pushes up against

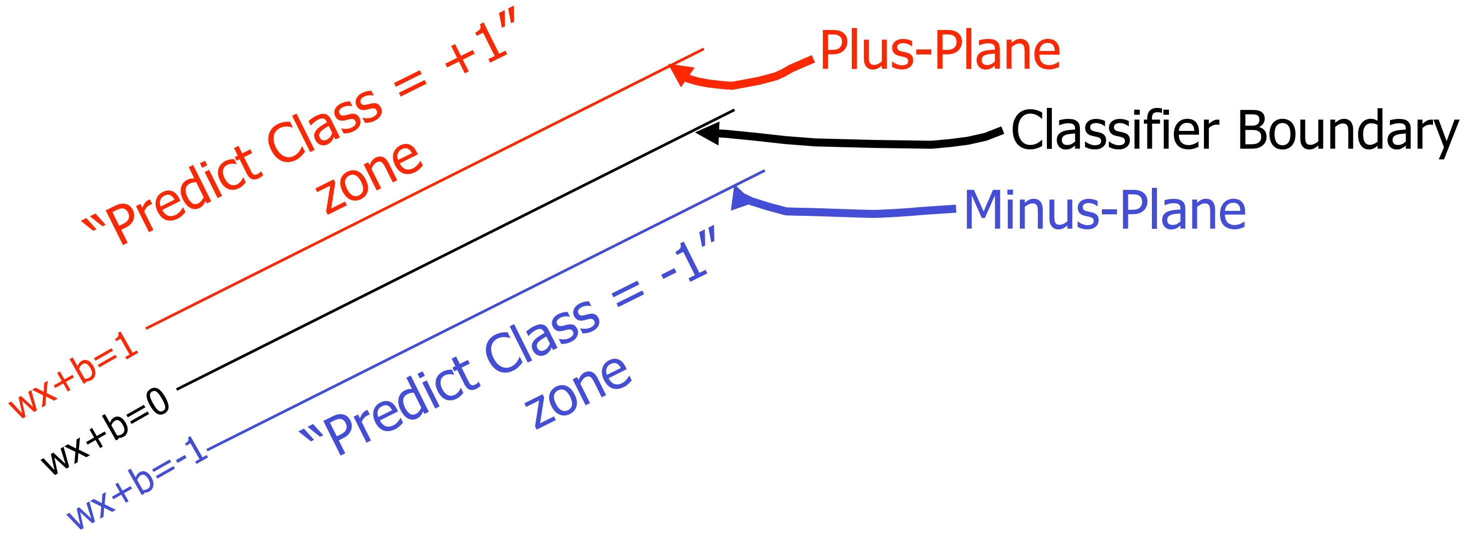
1. Intuitively this feels safest.
2. If we've made a small error in the location of the boundary (it's been jolted in its perpendicular direction) this gives us least chance of causing a misclassification.
3. LOOCV is easy since the model is immune to removal of any non-support-vector datapoints.
4. 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



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

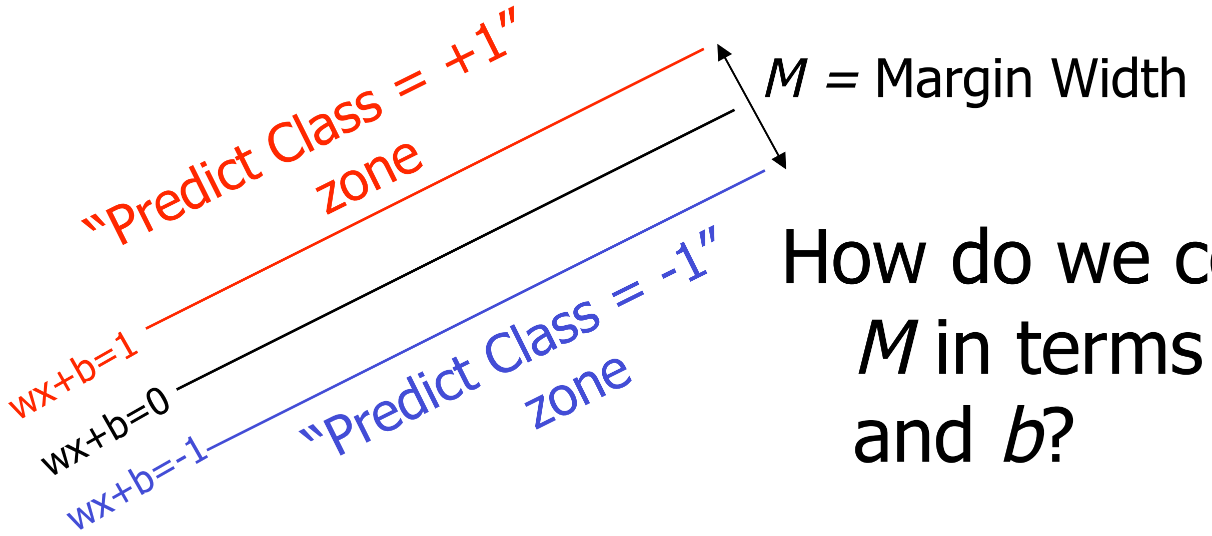
Specifying a line and margin



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

Classify as..	+1	if	$\mathbf{w} \cdot \mathbf{x} + b >= 1$
	-1	if	$\mathbf{w} \cdot \mathbf{x} + b <= -1$
	Universe explodes	if	$-1 < \mathbf{w} \cdot \mathbf{x} + b < 1$

Computing the margin width

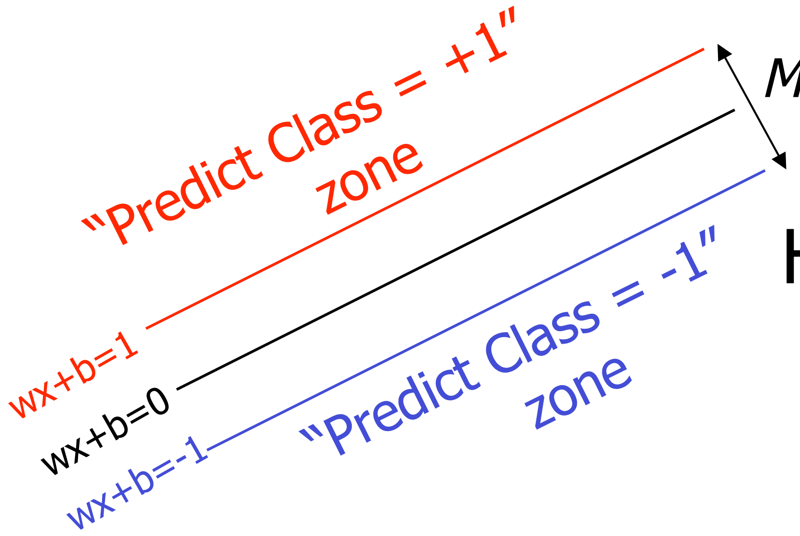


How do we compute M in terms of $\mathbf{w}$ and b?

- 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?

Computing the margin width



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

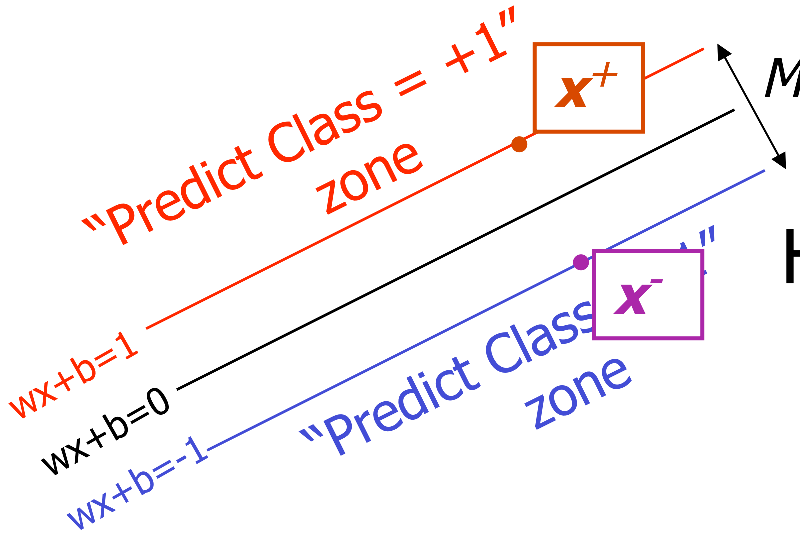
- Plus-plane = $\{ \mathbf{x} : \mathbf{w} \cdot \mathbf{x} + b = +1 \}$
- Minus-plane = $\{ \mathbf{x} : \mathbf{w} \cdot \mathbf{x} + b = -1 \}$

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

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

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

Computing the margin width

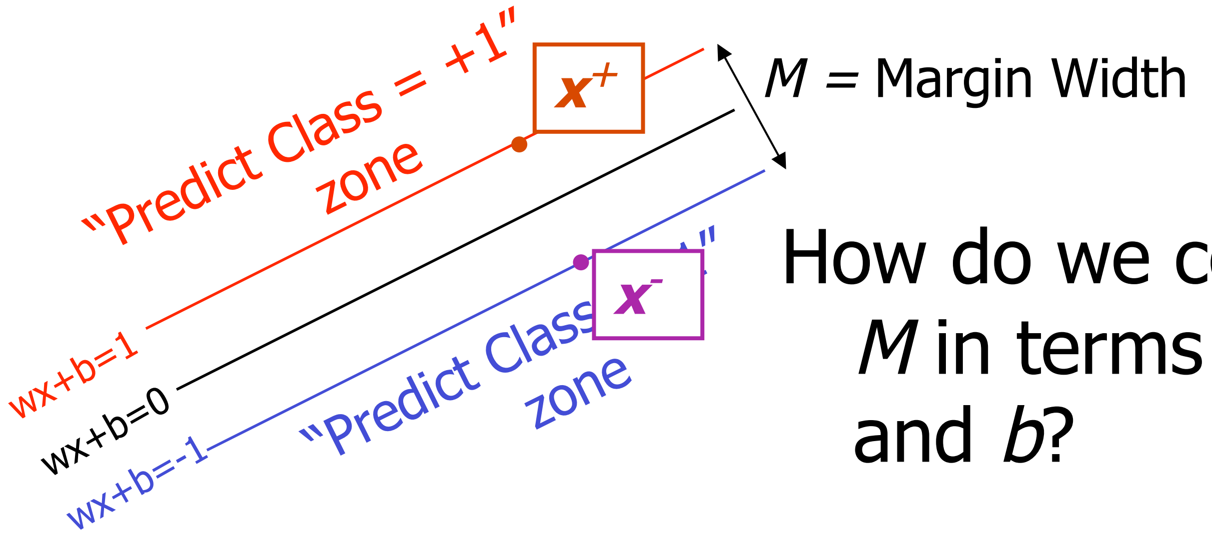


How do we compute M in terms of $\boldsymbol{w}$ and b?

- 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

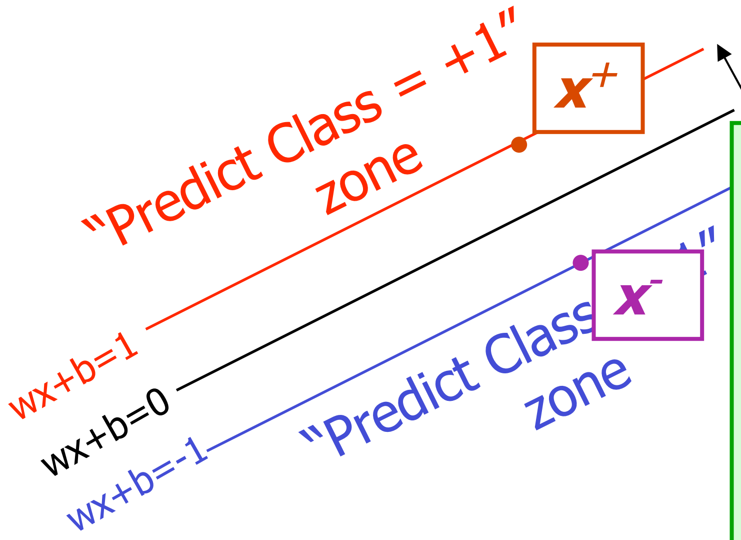
Computing the margin width



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

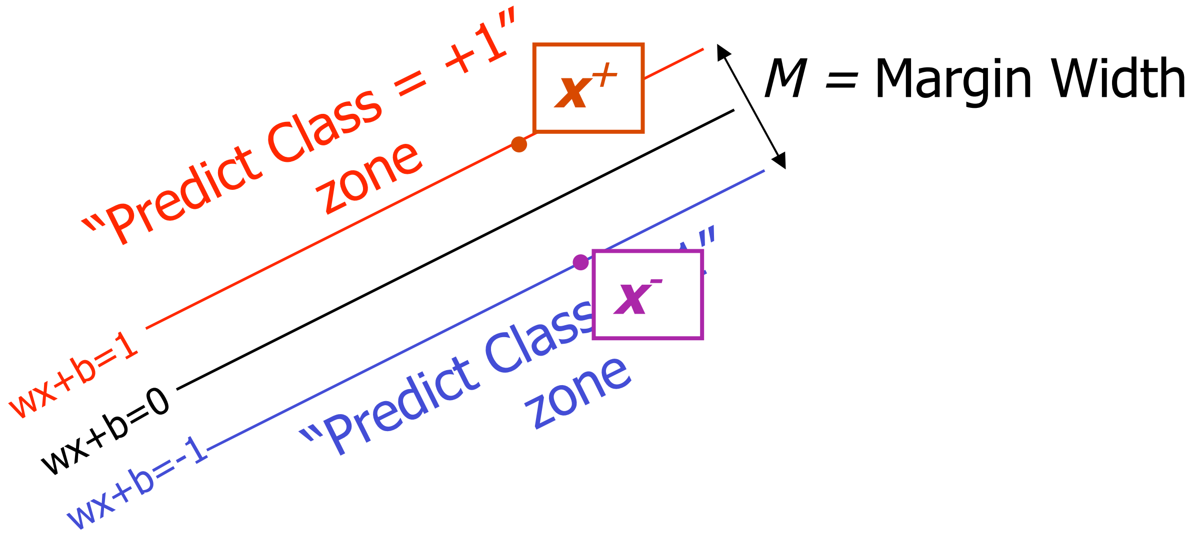


The line from $\boldsymbol{x}^-$ to $\boldsymbol{x}^+$ is perpendicular to the planes.

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

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

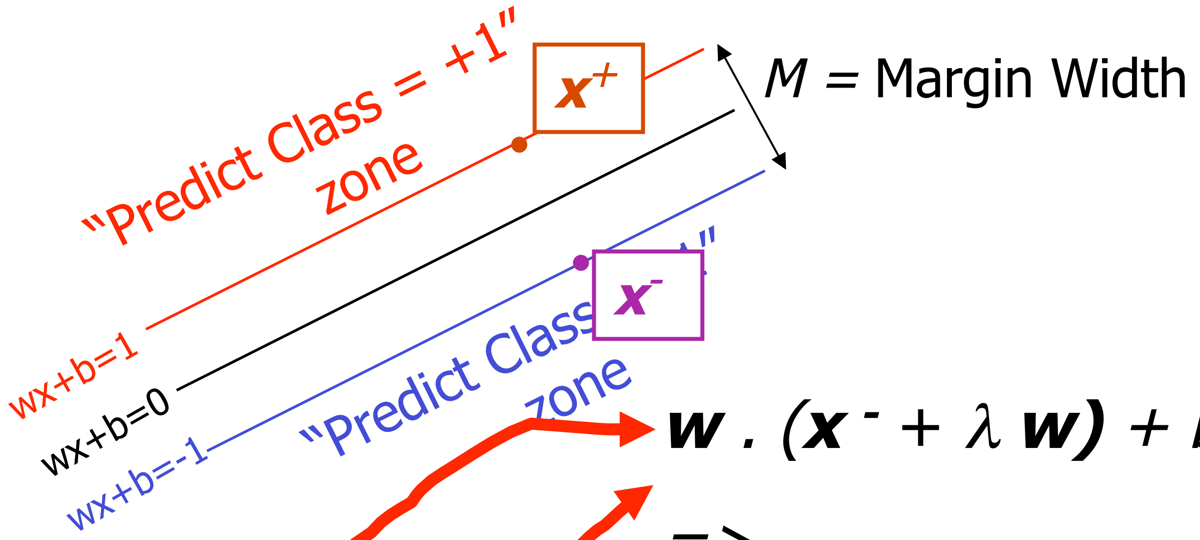


What we know:

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

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

Computing the margin width



What we know:

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

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

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

=>

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

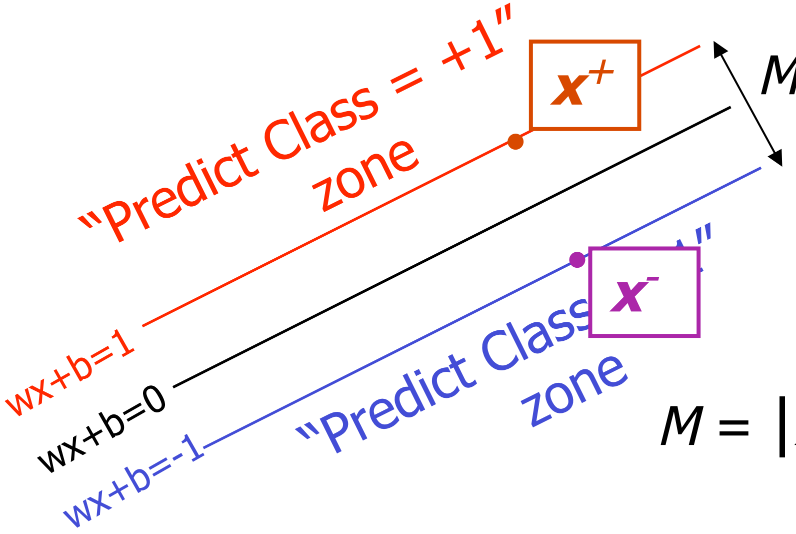
=>

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

=>

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

Computing the margin width



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

What we know:

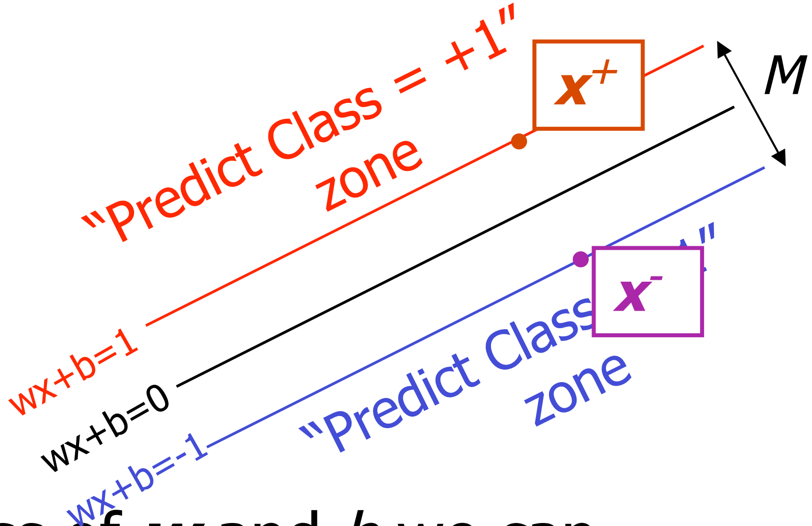
- $\boldsymbol{w} \,.\, \boldsymbol{x}^+ + b = +1$
- $\boldsymbol{w} \,.\, \boldsymbol{x}^- + b = -1$
- $\boldsymbol{x}^+ = \boldsymbol{x}^- + \lambda\, \boldsymbol{w}$
- $|\boldsymbol{x}^+ - \boldsymbol{x}^-| = M$
- $\lambda = \dfrac{2}{\mathbf{w}.\mathbf{w}}$

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

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

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

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

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 $\arg\max_{\mathbf{u}} \quad c + \mathbf{d}^T\mathbf{u} + \frac{\mathbf{u}^T R \mathbf{u}}{2}$

Quadratic criterion

Subject to

$$
\left.
\begin{aligned}
a_{11}u_1 + a_{12}u_2 + \ldots + a_{1m}u_m &\le b_1 \\
a_{21}u_1 + a_{22}u_2 + \ldots + a_{2m}u_m &\le b_2 \\
&\vdots \\
a_{n1}u_1 + a_{n2}u_2 + \ldots + a_{nm}u_m &\le b_n
\end{aligned}
\right\}
$$

n additional linear inequality constraints

And subject to

$$
\left.
\begin{aligned}
a_{(n+1)1}u_1 + a_{(n+1)2}u_2 + \ldots + a_{(n+1)m}u_m &= b_{(n+1)} \\
a_{(n+2)1}u_1 + a_{(n+2)2}u_2 + \ldots + a_{(n+2)m}u_m &= b_{(n+2)} \\
&\vdots \\
a_{(n+e)1}u_1 + a_{(n+e)2}u_2 + \ldots + a_{(n+e)m}u_m &= b_{(n+e)}
\end{aligned}
\right\}
$$

e additional linear equality constraints

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 u_1 ...

...al linear
...equality
...nstraints

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)

And subj...

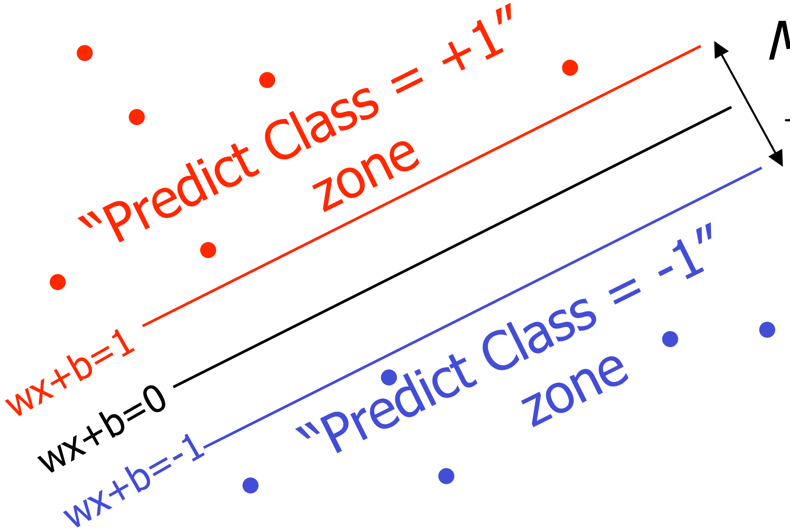
$\ldots = b_{(n+1)}$

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

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

e additional linear equality constraints

Learning the Maximum Margin Classifier



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

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

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

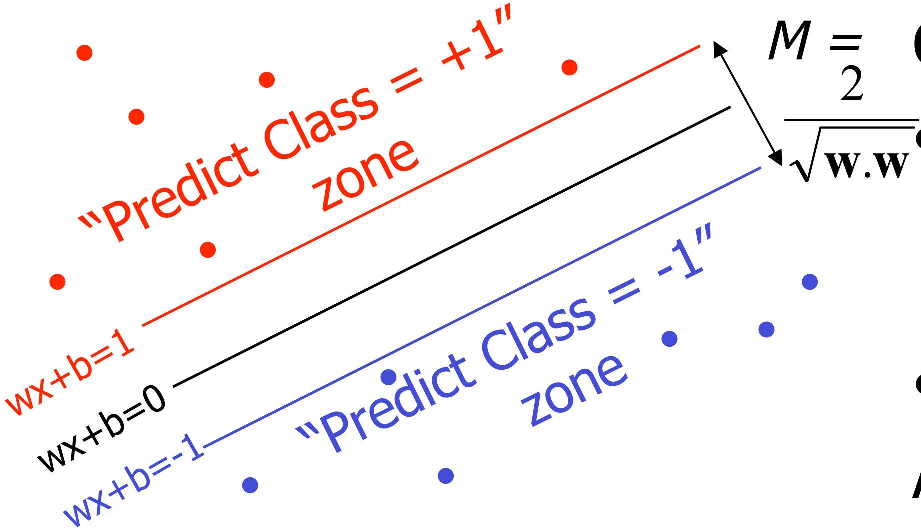
Assume R datapoints, each $(\mathbf{x}_k, y_k)$ where $y_k = +/- 1$

What should our quadratic optimization criterion be?

How many constraints will we have?

What should they be?

Learning the Maximum Margin Classifier



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?

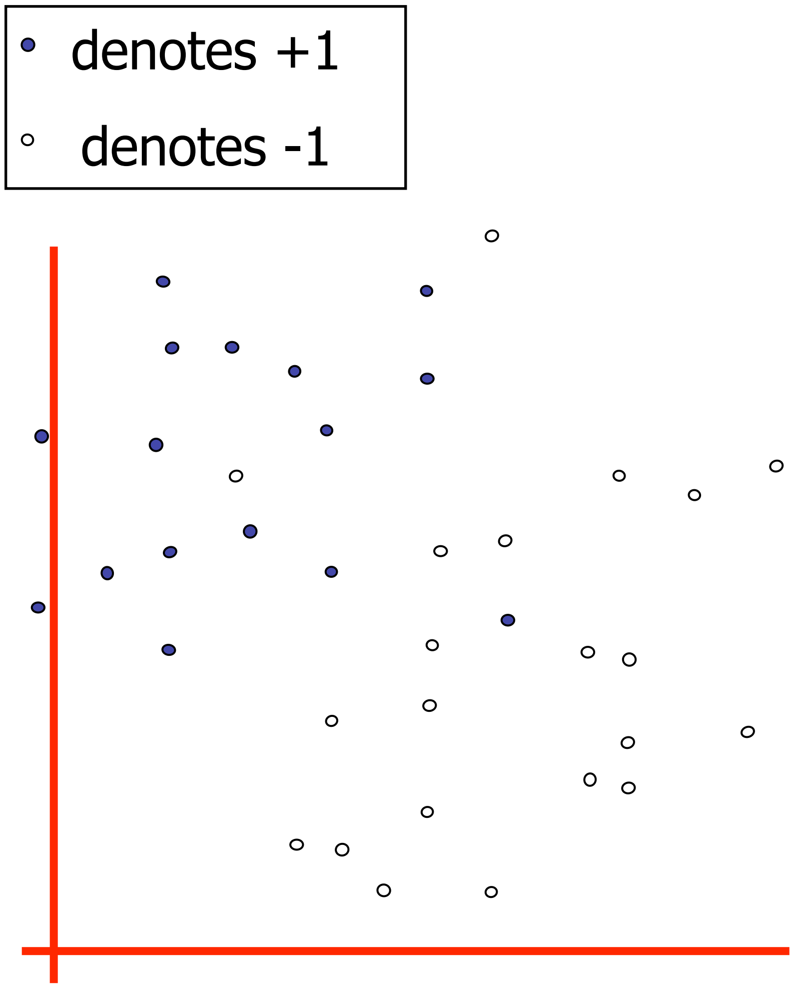
$\mathbf{w} \cdot \mathbf{x}_k + b >= 1 \text{ if } y_k = 1$

$\mathbf{w} \cdot \mathbf{x}_k + b <= -1 \text{ if } y_k = -1$

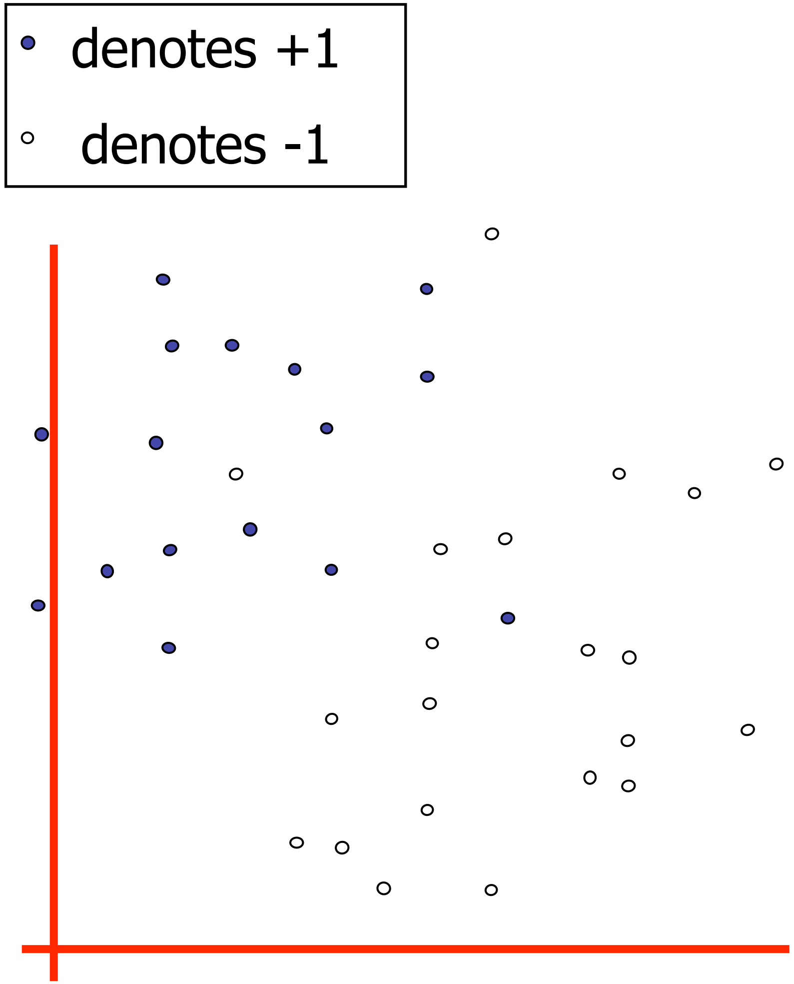
Uh-oh!

This is going to be a problem!

What should we do?



Uh-oh!



This is going to be a problem!

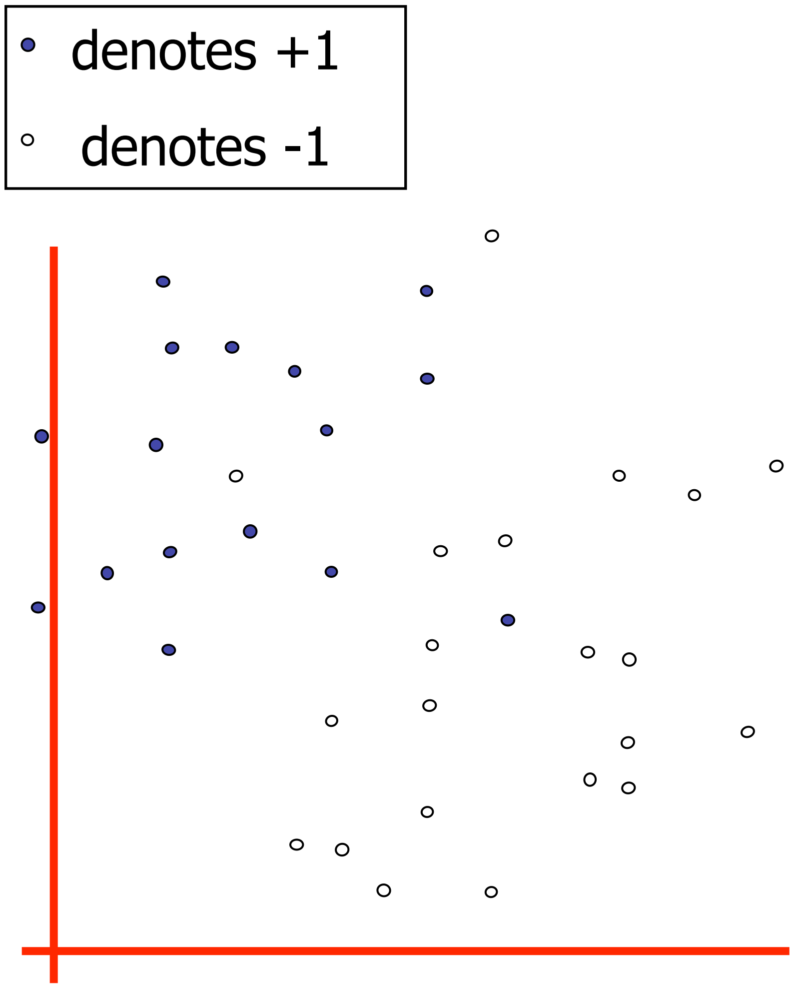
What should we do?

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

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

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?

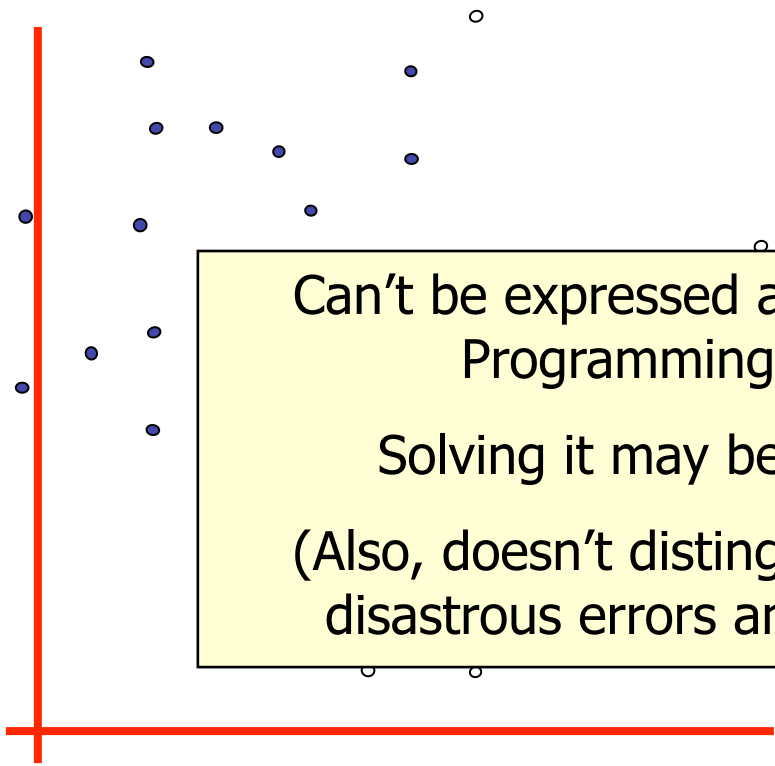
Idea 1.1:

Minimize

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

Tradeoff parameter

- denotes +1
- denotes -1



Can't be expressed as a Quadratic Programming problem.

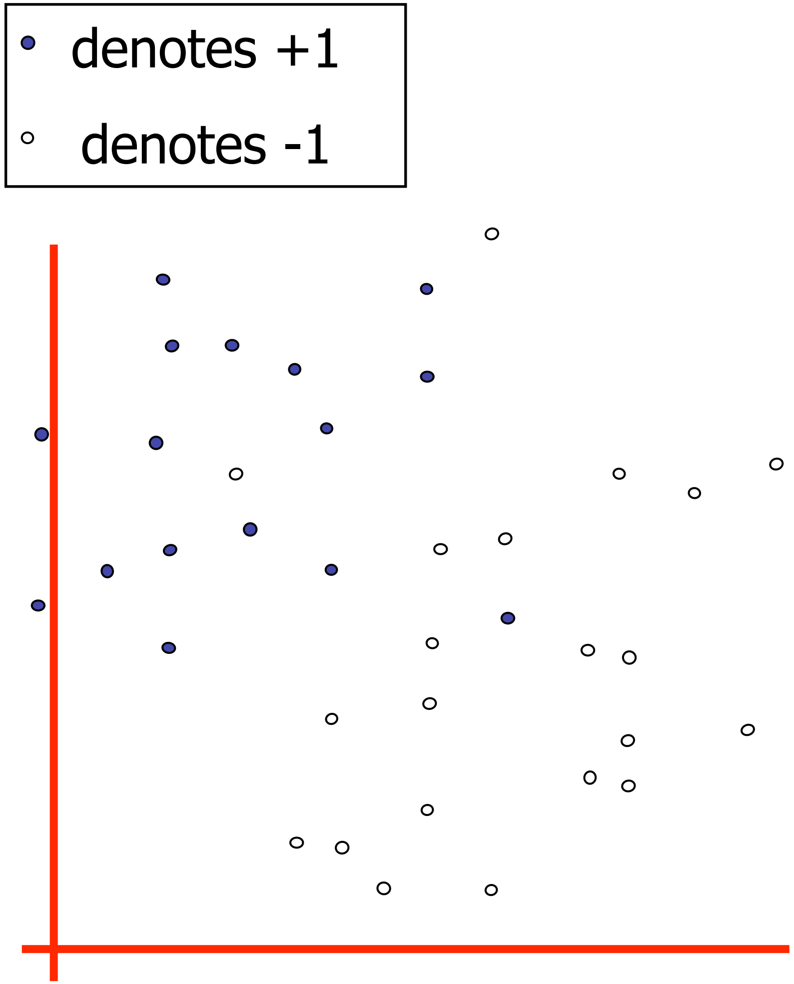
Solving it may be too slow.

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

...erious... at'... ke... us reject this... you guess wha...

So… any other ideas?

Uh-oh!



This is going to be a problem!

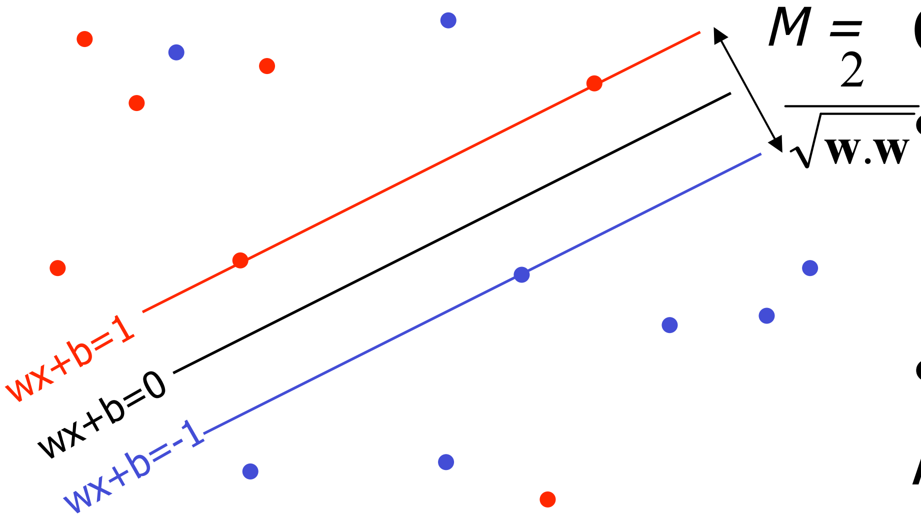
What should we do?

Idea 2.0:

Minimize

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

Learning Maximum Margin with Noise



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

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

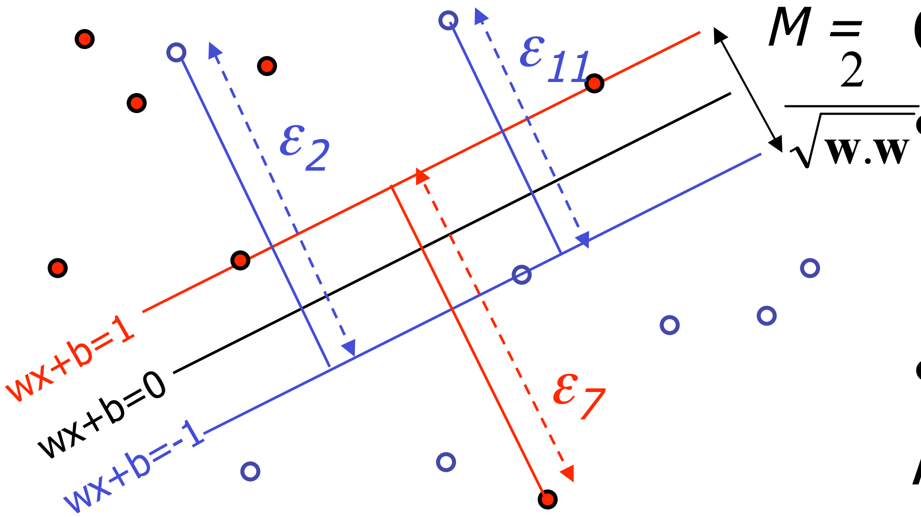
Assume R datapoints, each $(\mathbf{x}_k, y_k)$ where $y_k = +/- 1$

What should our quadratic optimization criterion be?

How many constraints will we have?

What should they be?

Learning Maximum Margin with Noise



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*

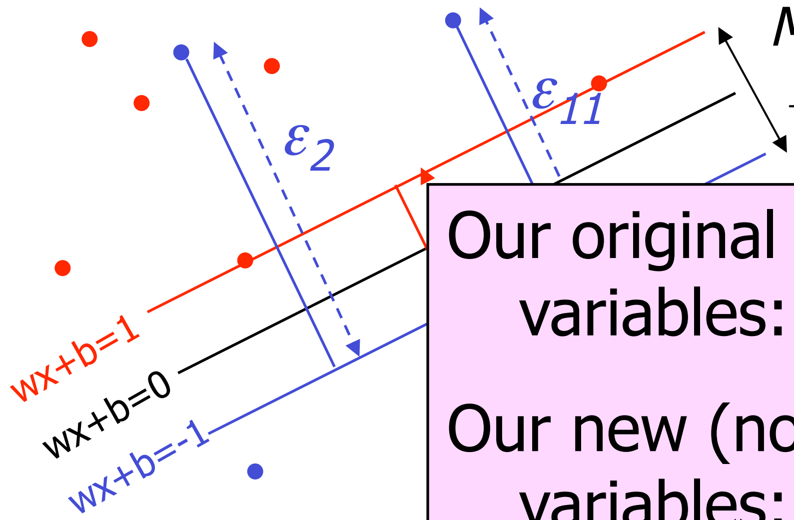
What should they be?

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

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

Learning Maximum Margin with Noise

m = # input dimensions



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

Given gu... we can

- Compute sum of distances ...

... dth

... $(\mathbf{x}_k, y_k)$ where y_k ... 1

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= # records

What should our quadratic optimization criterion be?

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

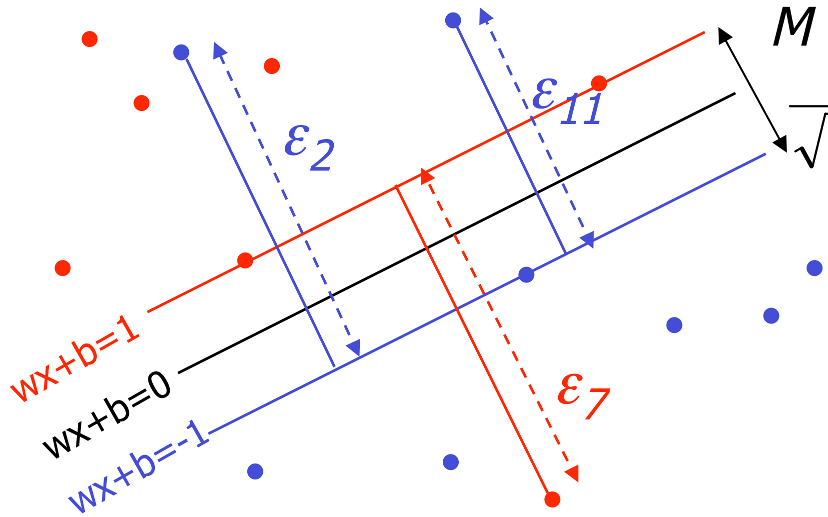
How many constraints will we have? *R*

What should they be?

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

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

Learning Maximum Margin with Noise



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

Given 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*

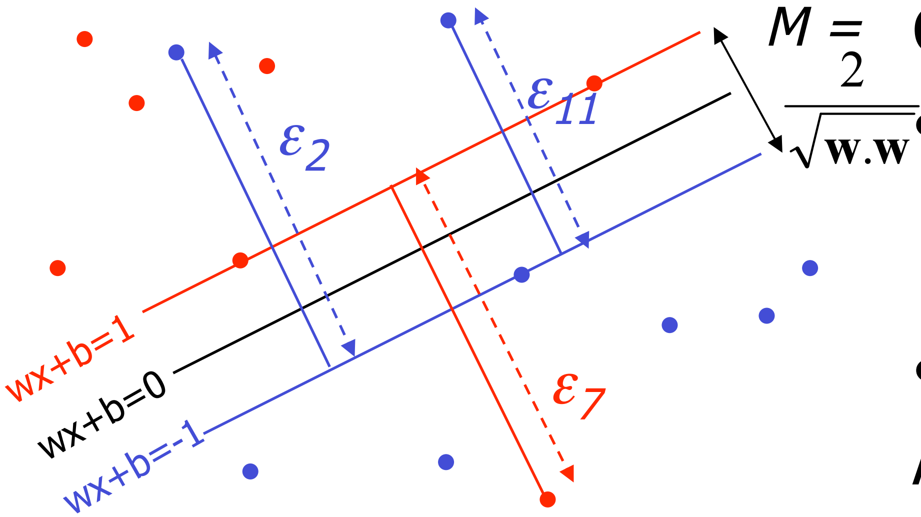
What should they be?

$\mathbf{w} \,.\, \mathbf{x}_k + b >= 1-\varepsilon_k$ *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 $\boldsymbol{w}$, b we can

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

Assume R datapoints, each $(\boldsymbol{x}_k, y_k)$ 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?

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

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

$\varepsilon_k >= 0$ for all k