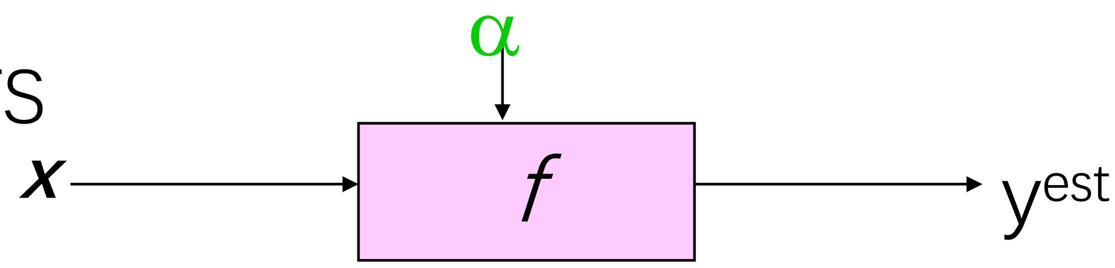
2019
怪兽
学堂

Support Vector Machines

虾米 2019-5

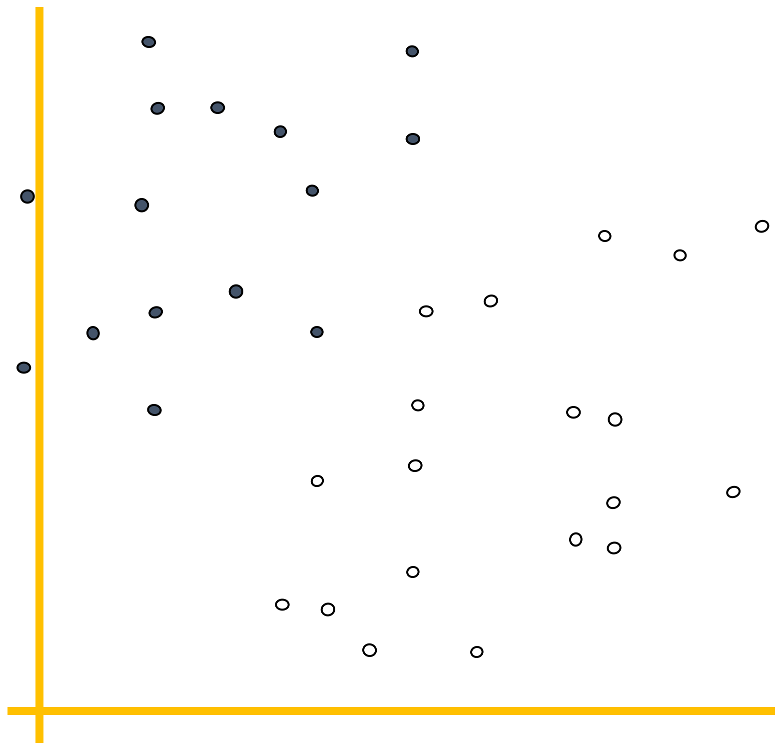
Linear Classifiers



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

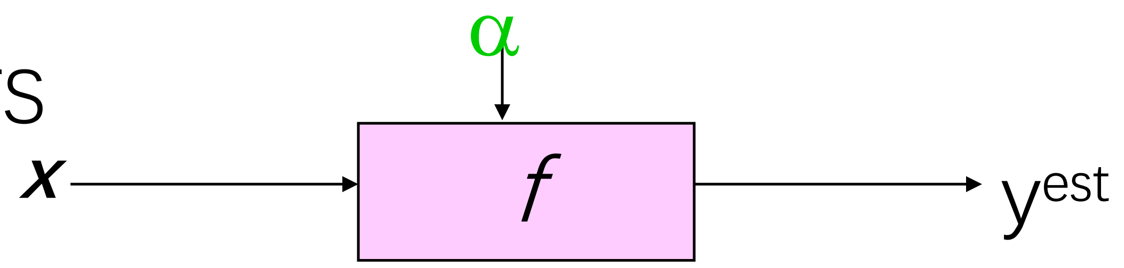
• denotes +1

○ denotes -1

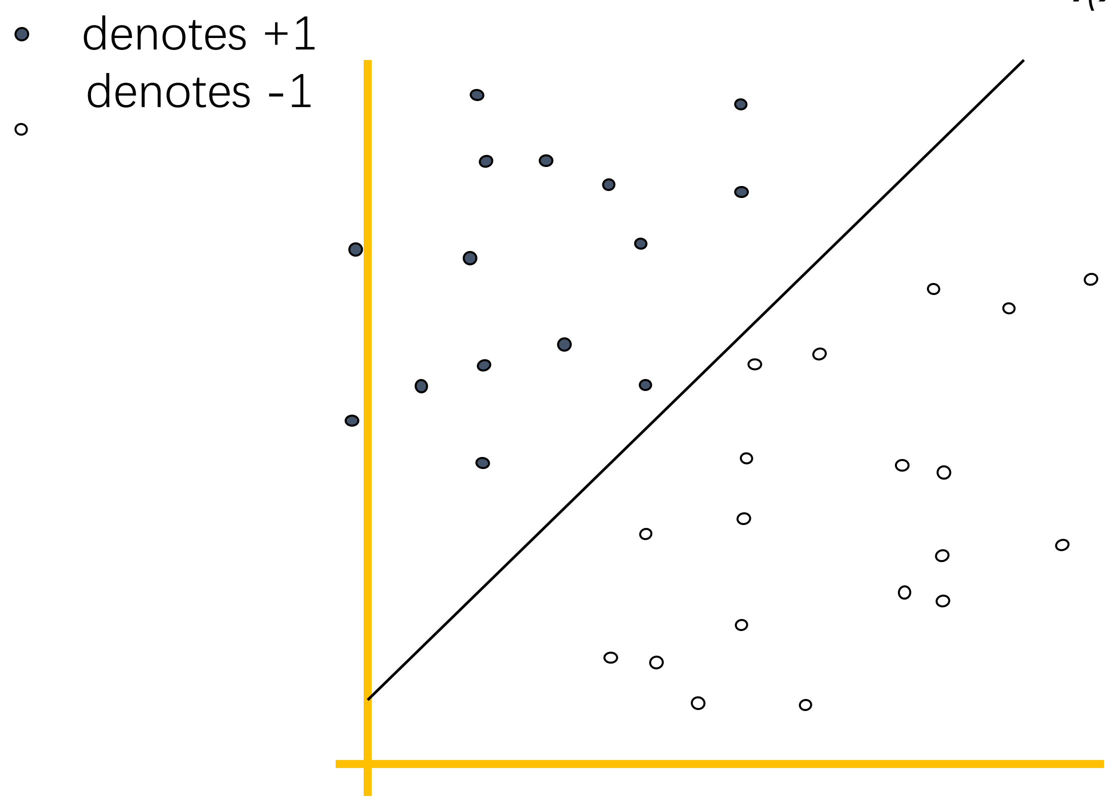


How would you classify this data?

Linear Classifiers

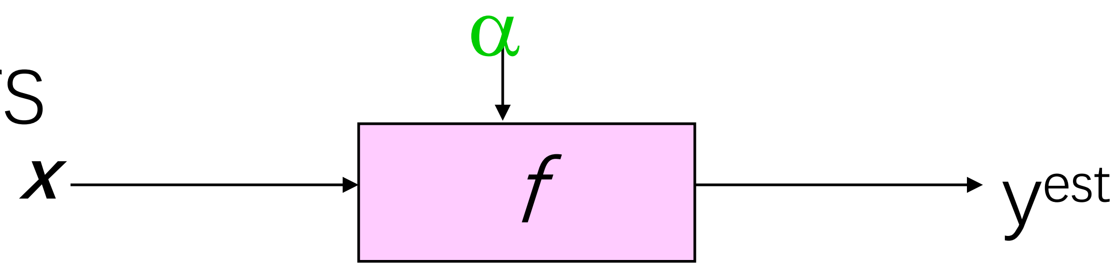


$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w} \cdot \mathbf{x} - b)$

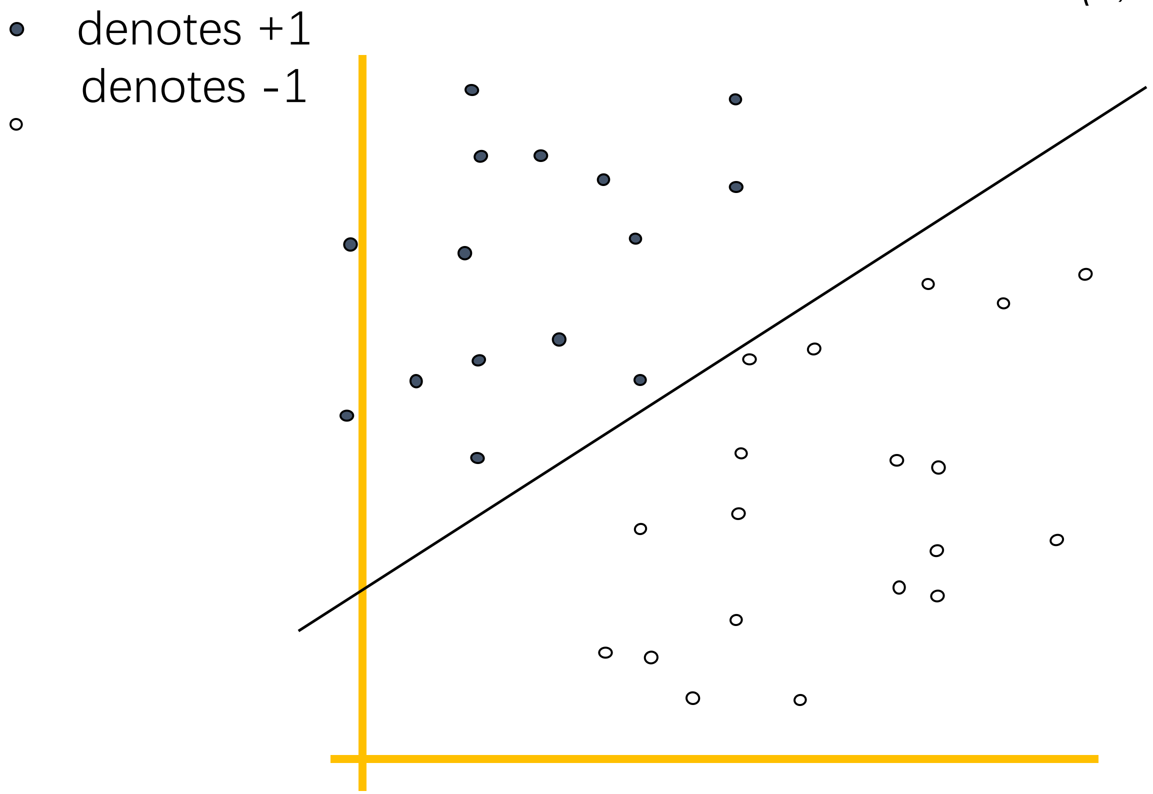


How would you classify this data?

Linear Classifiers

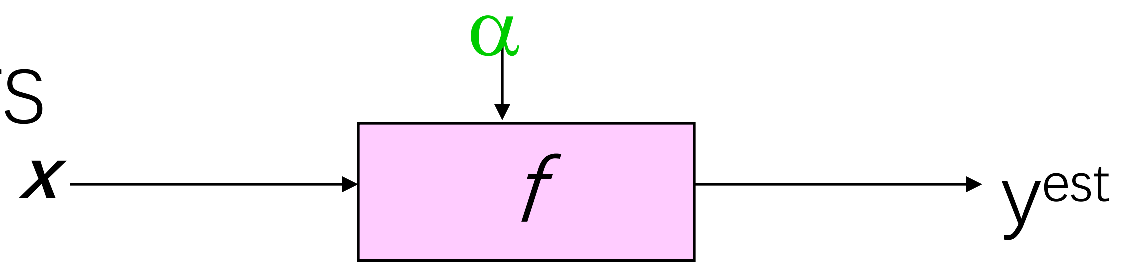


$f(x,w,b) = sign(w. x - b)$

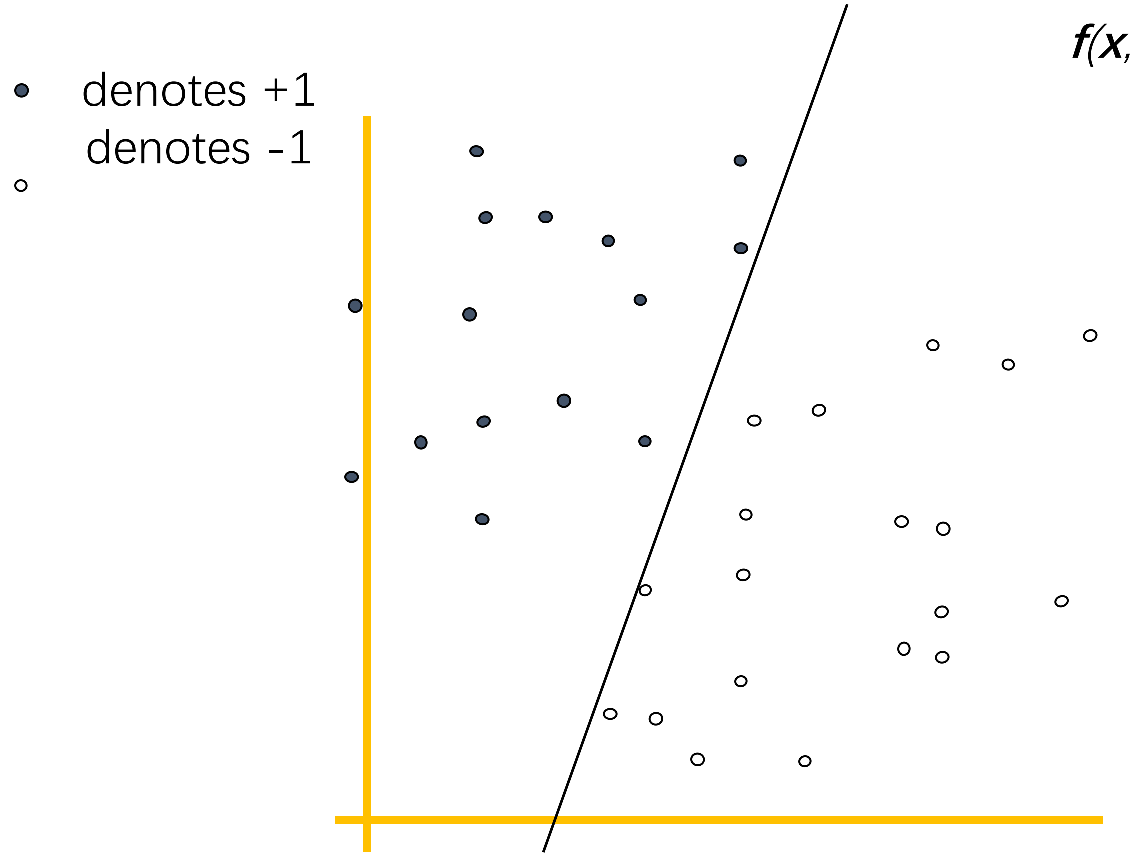


How would you classify this data?

Linear Classifiers

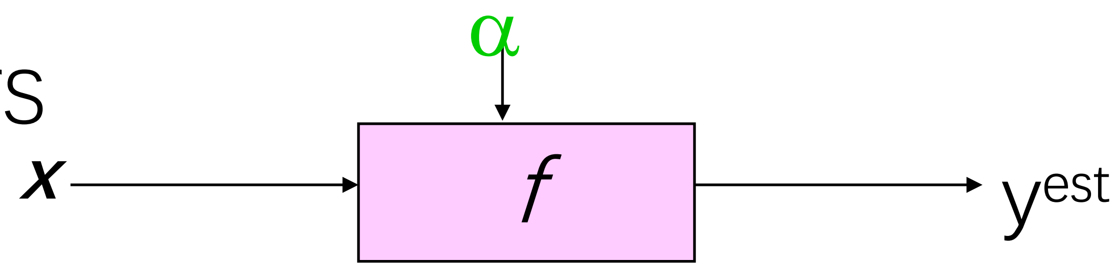


$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} - b)$

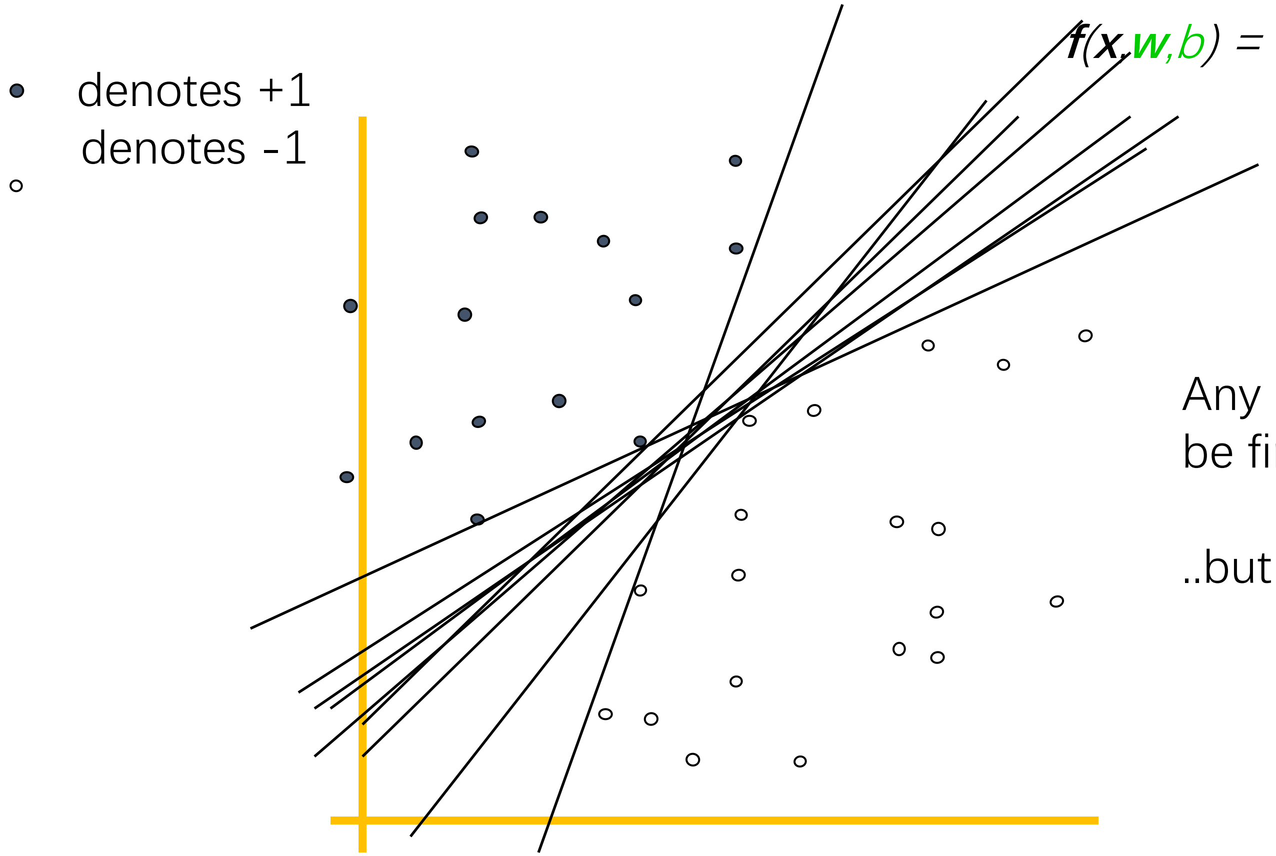


How would you classify this data?

Linear Classifiers



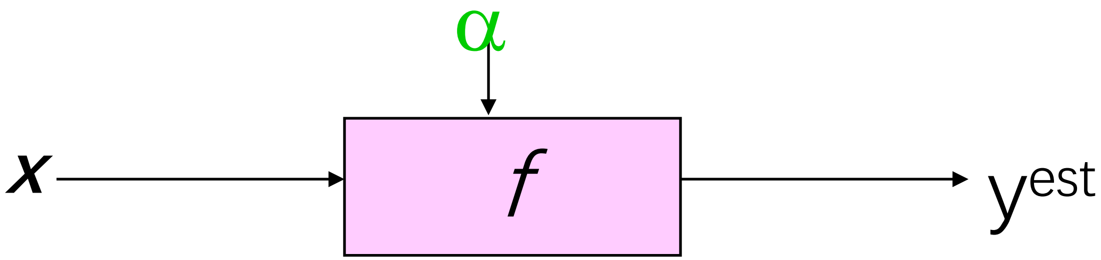
$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} - b)$



Any of these would be fine..

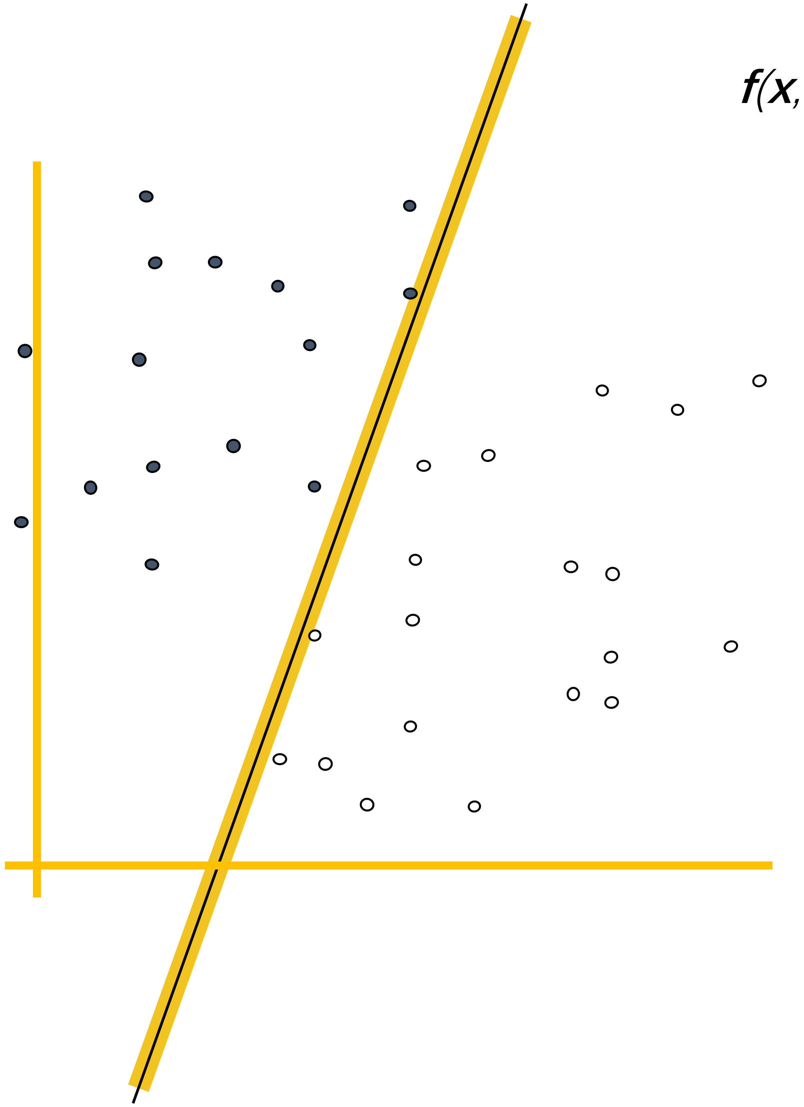
..but which is best?

Classifier Margin



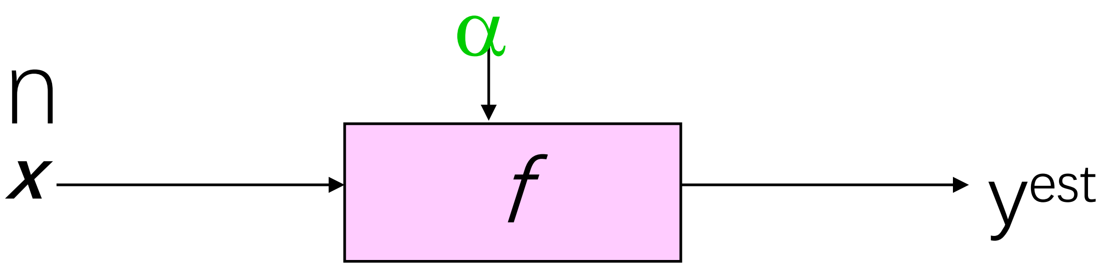
$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} - b)$

- denotes +1
- denotes -1

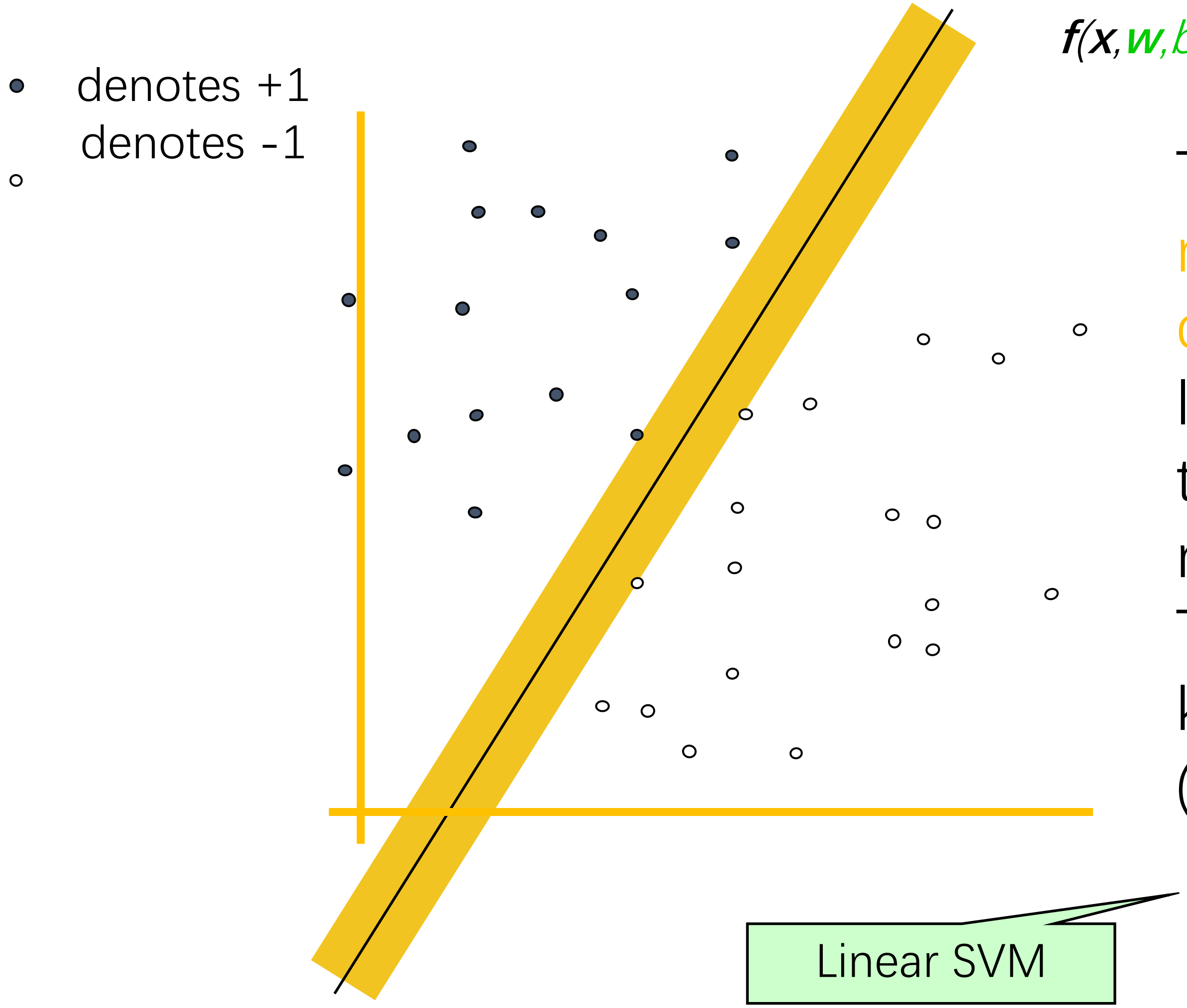


Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

Maximum Margin

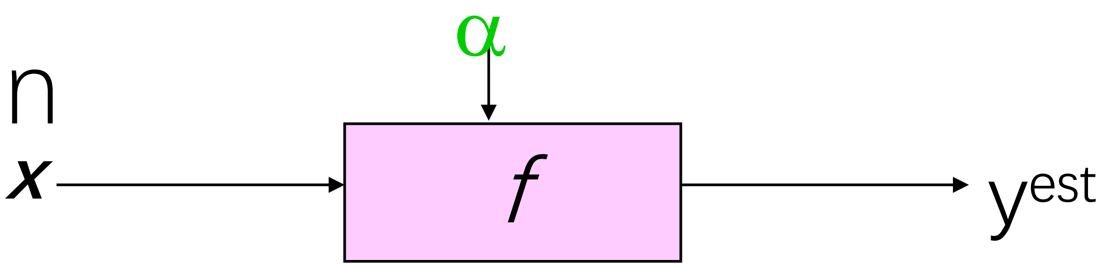


$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} - b)$



The maximum margin linear classifier is the linear classifier with the, um, maximum margin.
This is the simplest kind of SVM (Called an LSVM)

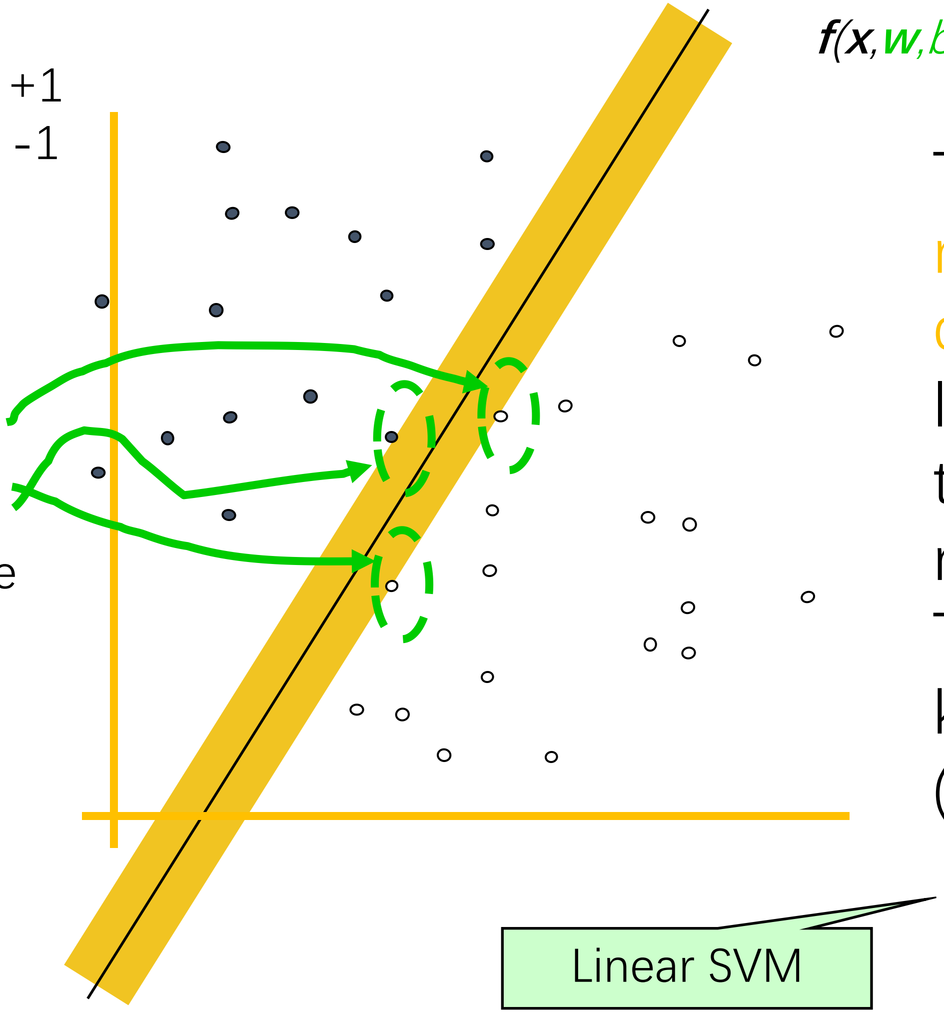
Maximum Margin



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

● denotes +1

○ denotes -1

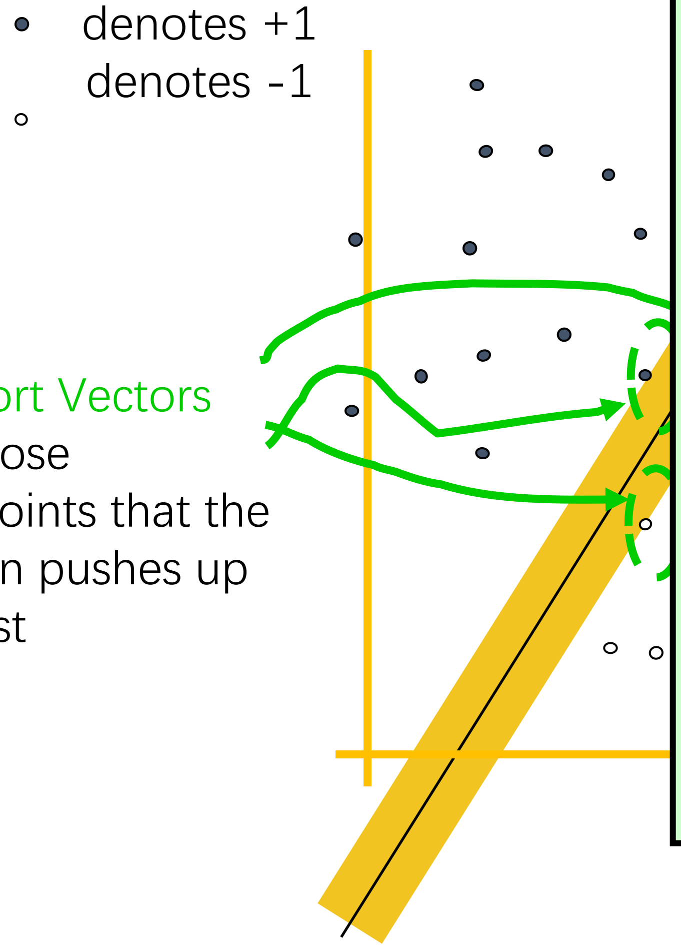


Support Vectors are those datapoints that the margin pushes up against

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

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

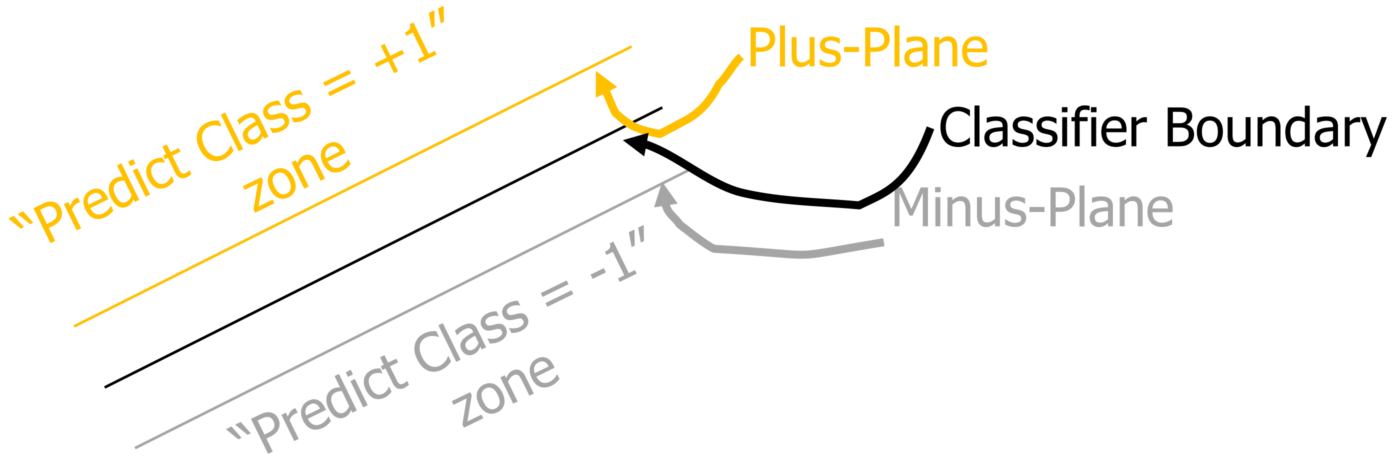
Why Maximum Margin?



Support Vectors are those datapoints that the margin pushes up against

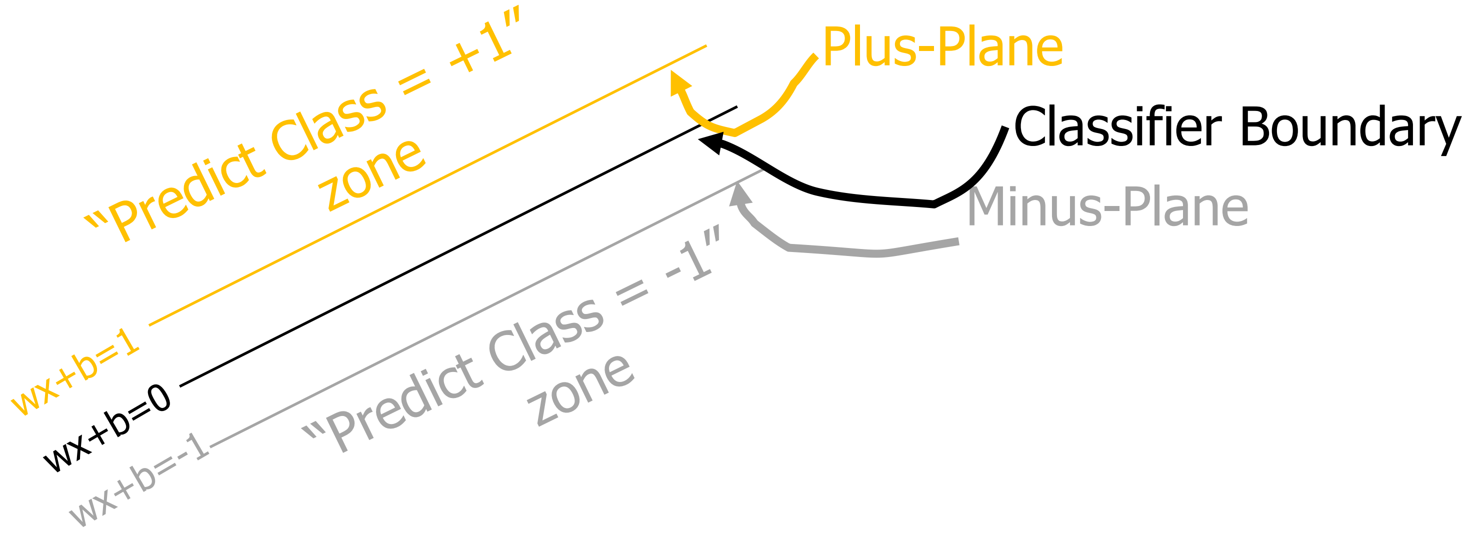
1. Intuitively this feels safest.
2. If we've made a small error in the location of the boundary (it's been jolted in its perpendicular direction) this gives us least chance of causing a misclassification.
3. Model is immune to removal of any non-support-vector datapoints.
4. There's some theory (using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

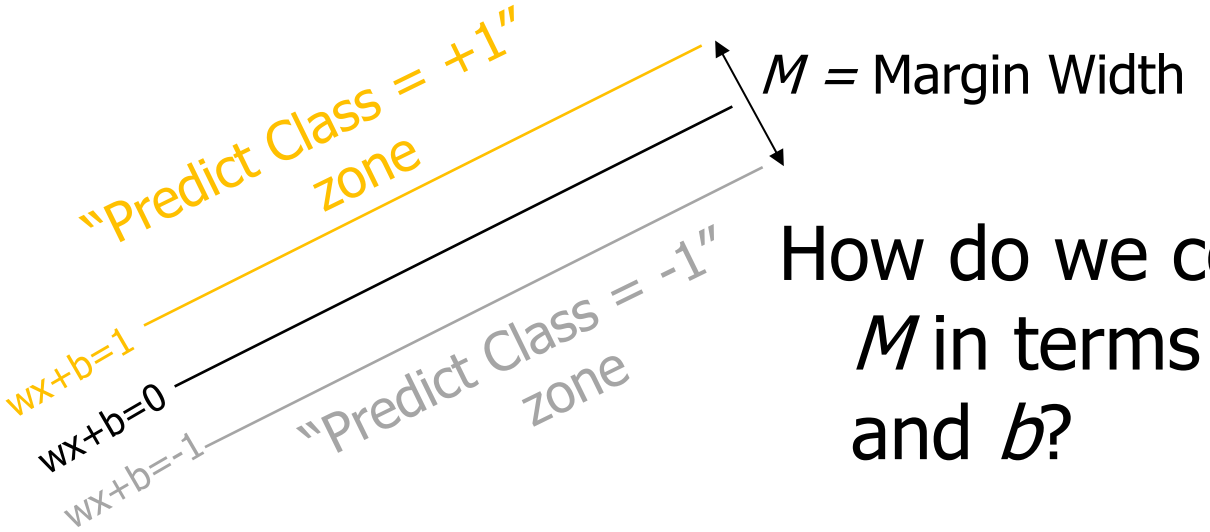
Specifying a line and margin



- 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



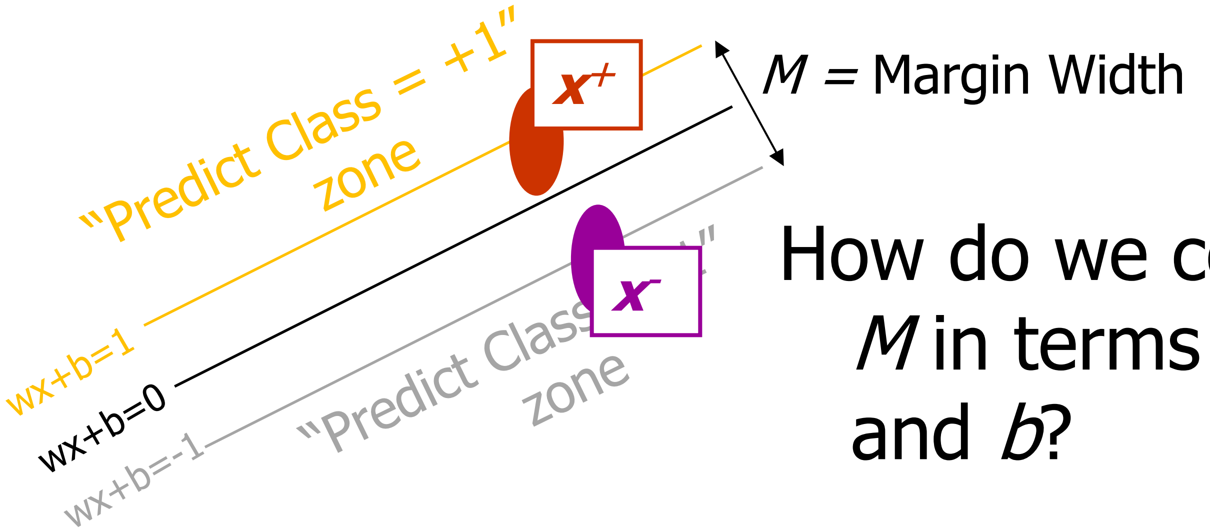
How do we compute M in terms of $\mathbf{w}$ and b?

- 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.

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

Computing the margin width

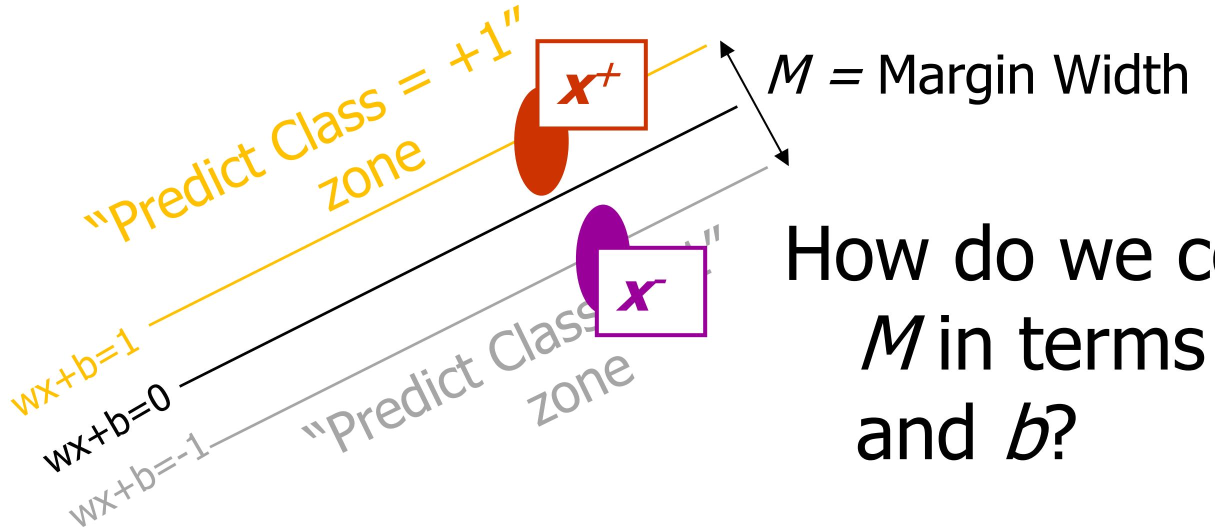


How do we compute M in terms of $\boldsymbol{w}$ and b?

- 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

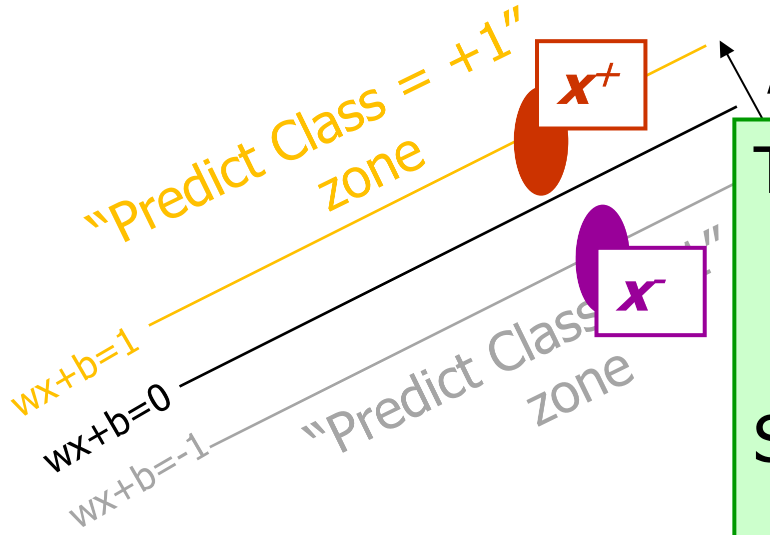
Computing the margin width



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.

Computing the margin width

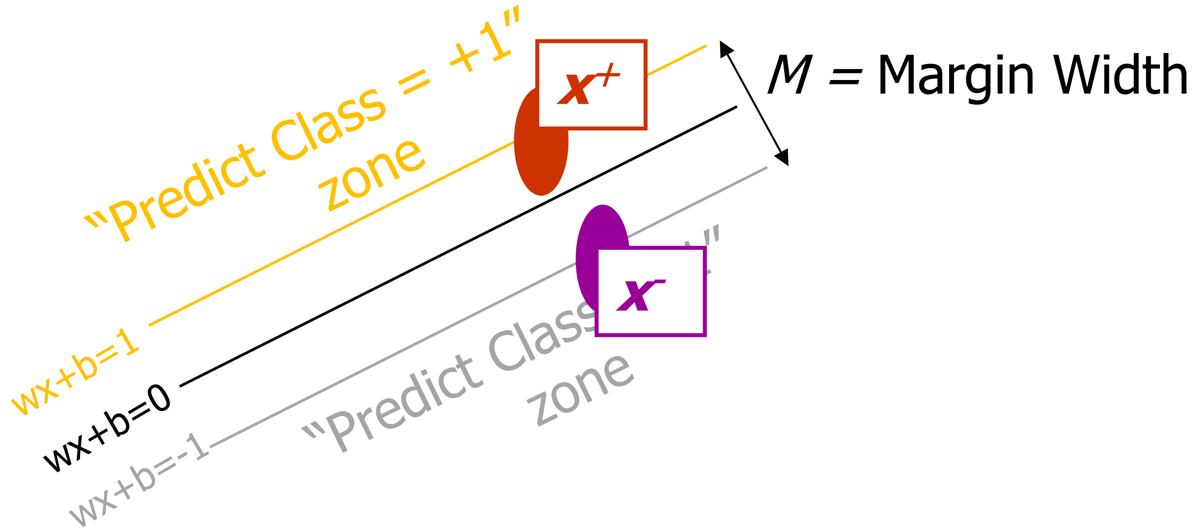


The line from $\boldsymbol{x}^-$ to $\boldsymbol{x}^+$ is perpendicular to the planes.

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.

Computing the margin width

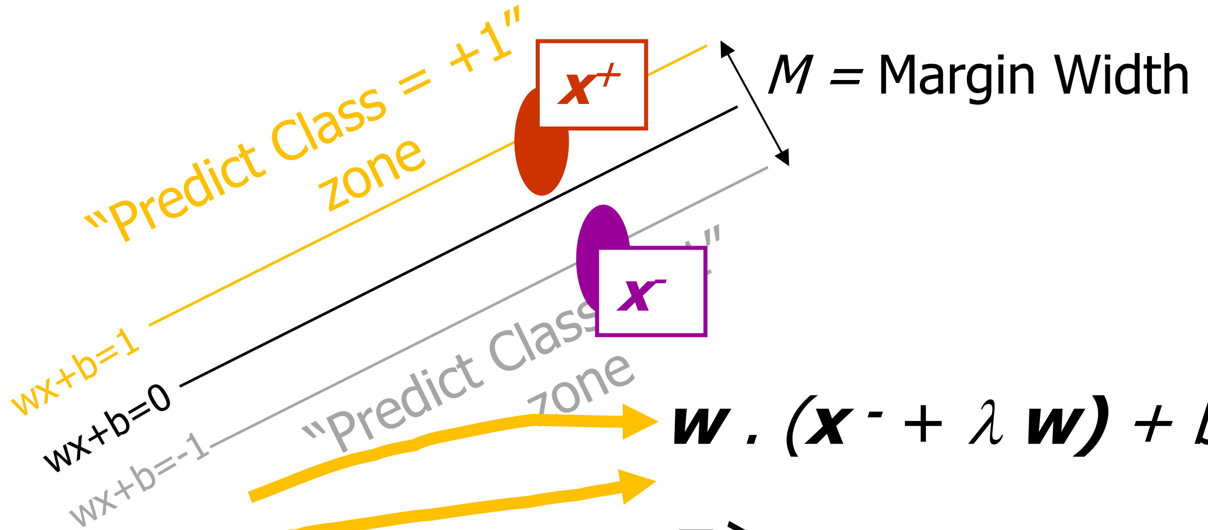


What we know:

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

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

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

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

=>

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

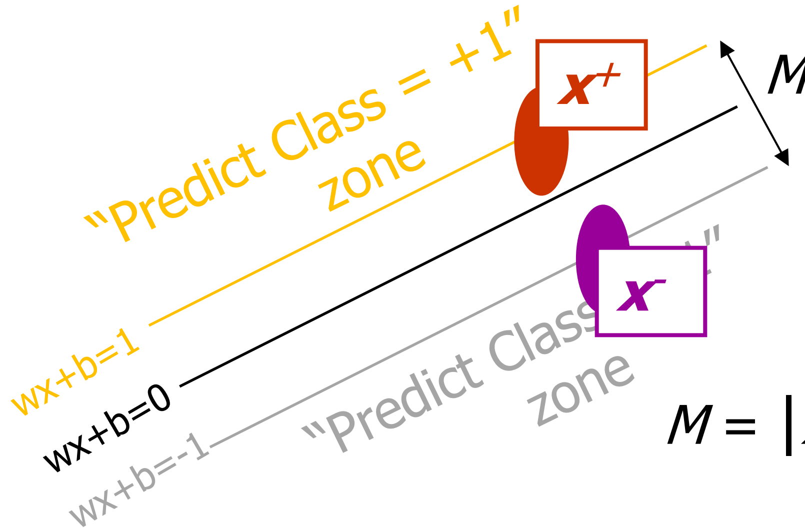
=>

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

=>

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

Computing the margin width



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

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

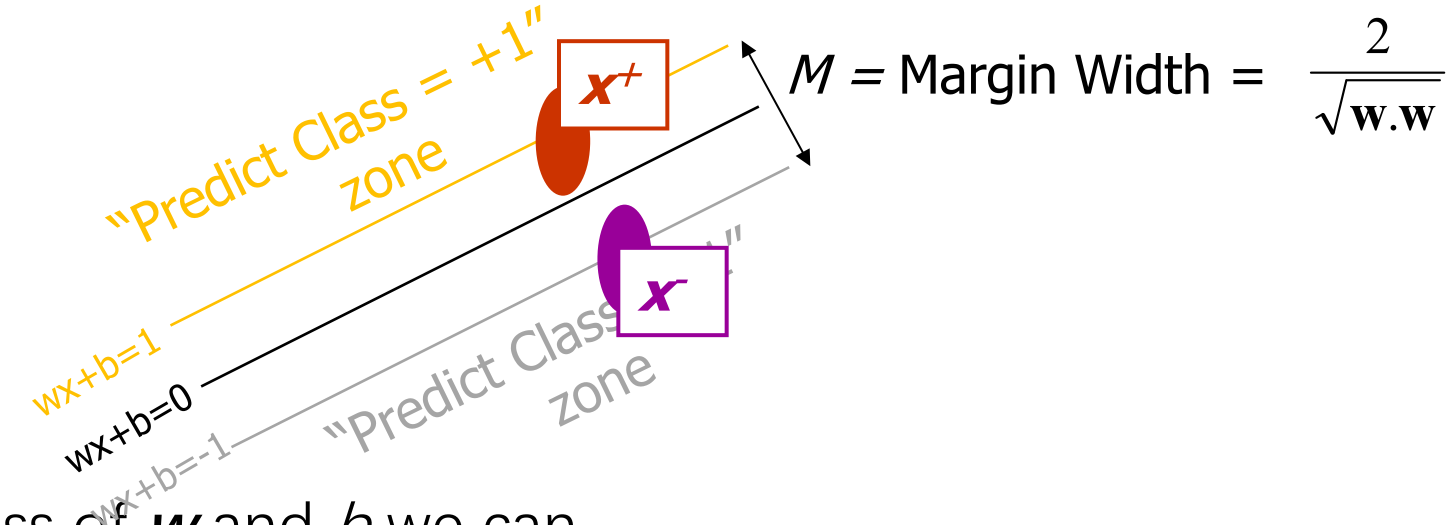
$$= \lambda\, |\mathbf{w}| = \lambda \sqrt{\mathbf{w}.\mathbf{w}}$$

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

What we know:

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

Learning the Maximum Margin Classifier



Given a guess of $\boldsymbol{w}$ 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.

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

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

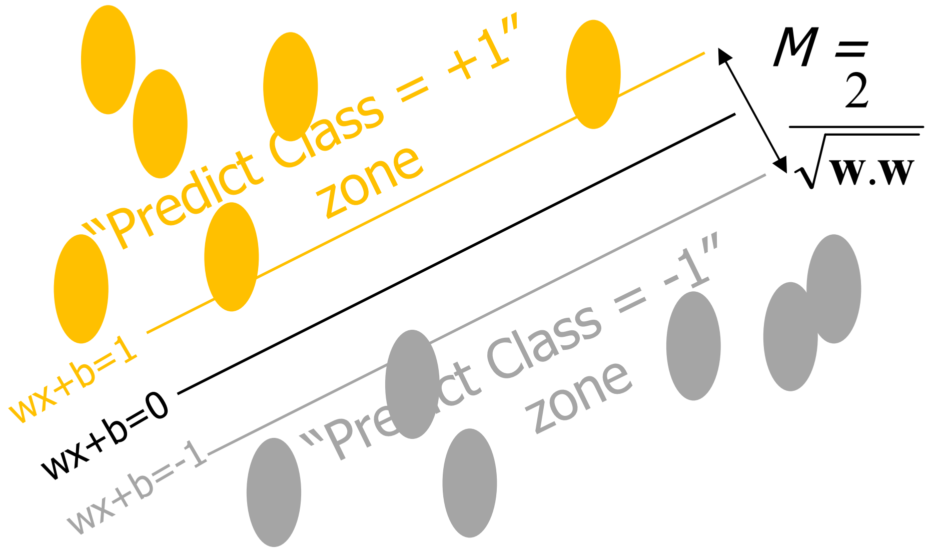
n additional linear <u>in</u>equality constraints

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 <u>e</u>quality constraints

Learning the Maximum Margin Classifier



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

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

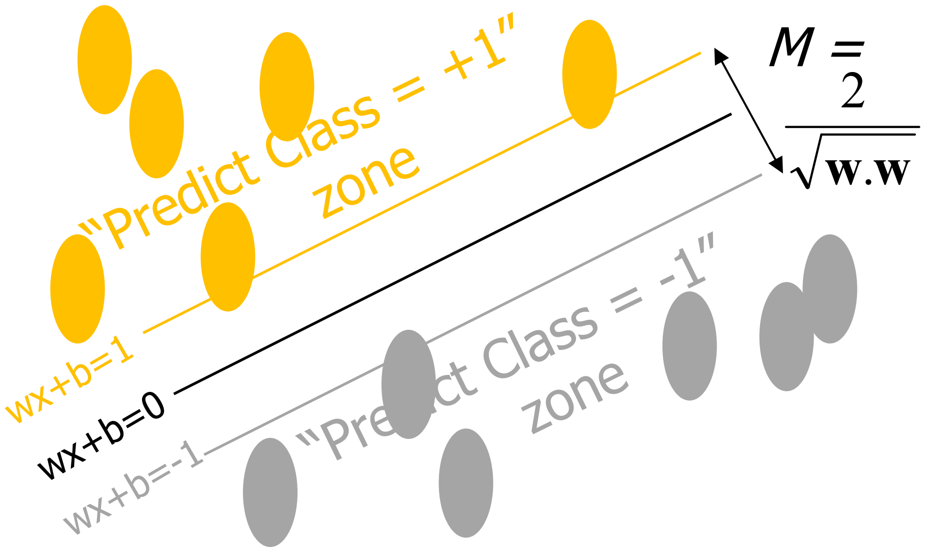
Assume R datapoints, each $(\boldsymbol{x}_k, y_k)$ where $y_k = +/- 1$

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 $\mathbf{w}$, 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?

Minimize $\mathbf{w}.\mathbf{w}$

How many constraints will we have? *R*

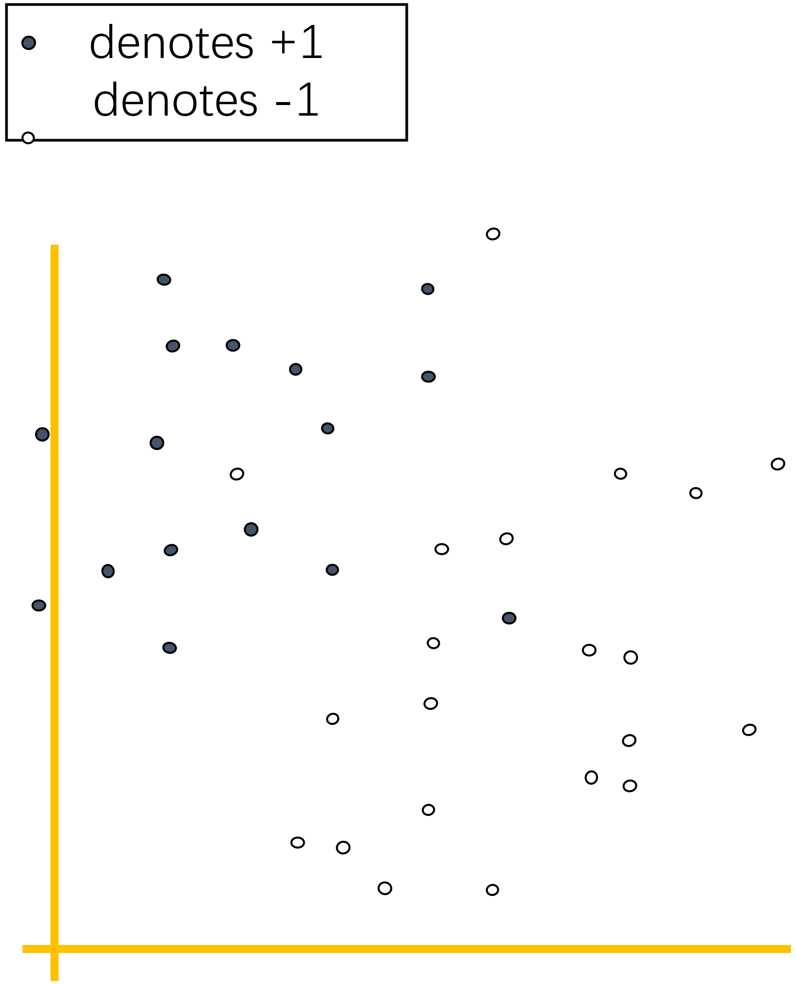
What should they be?

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

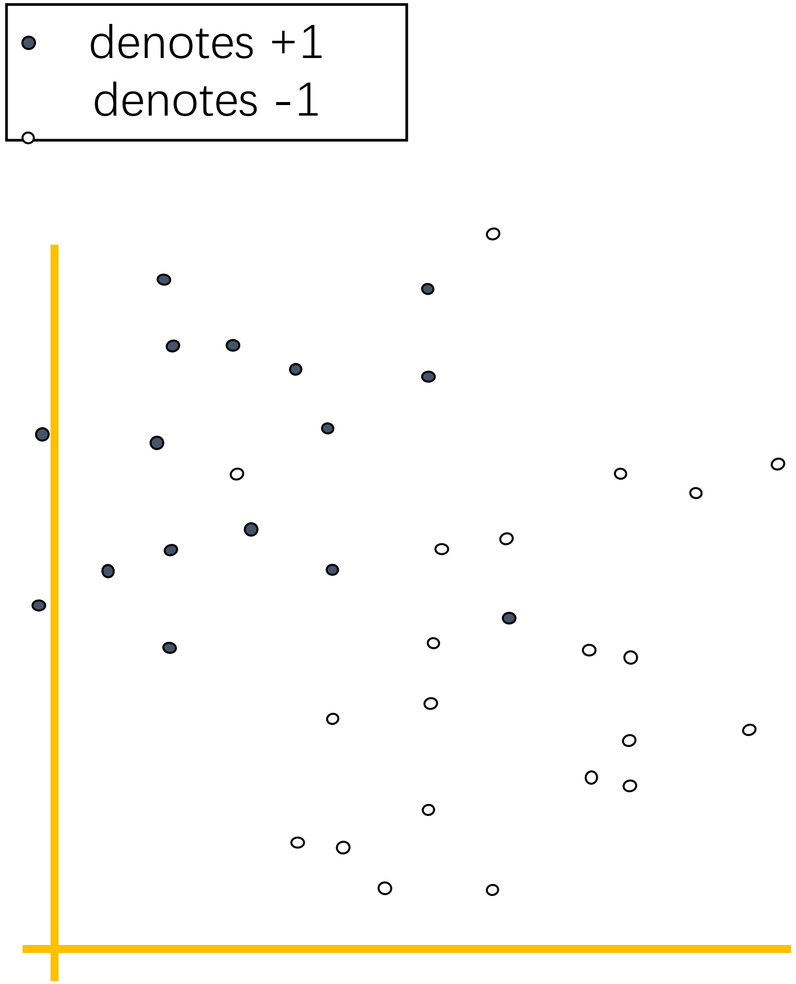
$\mathbf{w} \,.\, \mathbf{x}_k + b <= -1 \text{ if } y_k = -1$

Uh-oh!

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



Uh-oh!



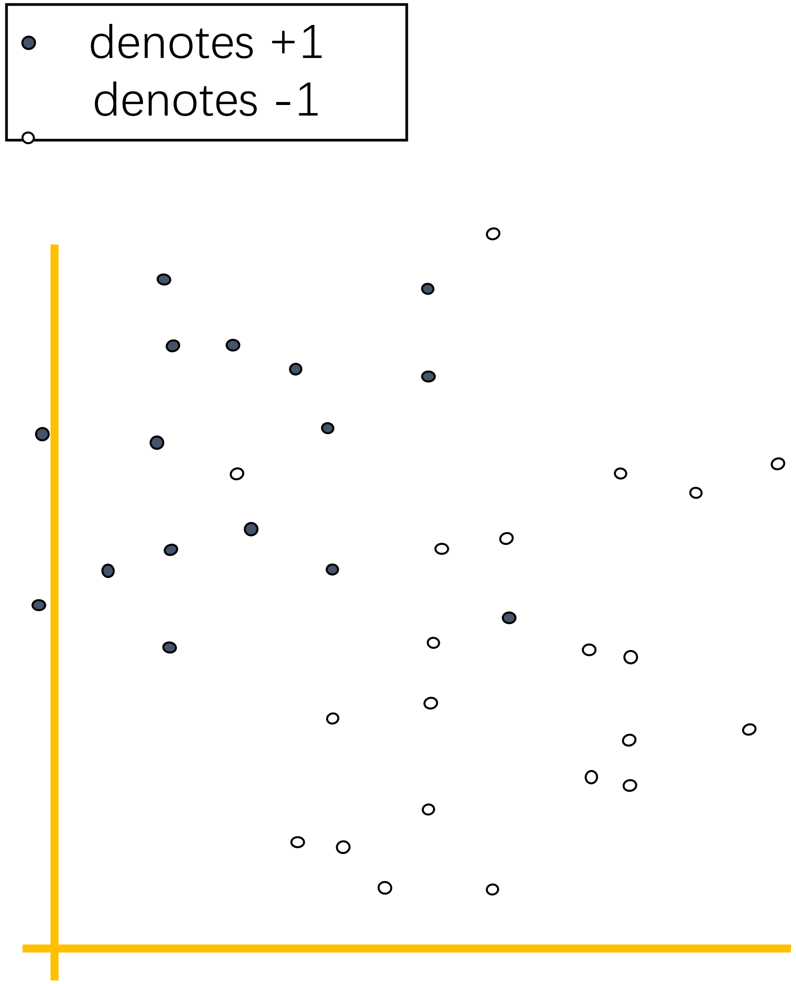
This is going to be a problem!
What should we do?

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

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

Tradeoff parameter

There's a serious practical problem that's about to make us reject this approach.

Uh-oh!

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

Idea 1.1:

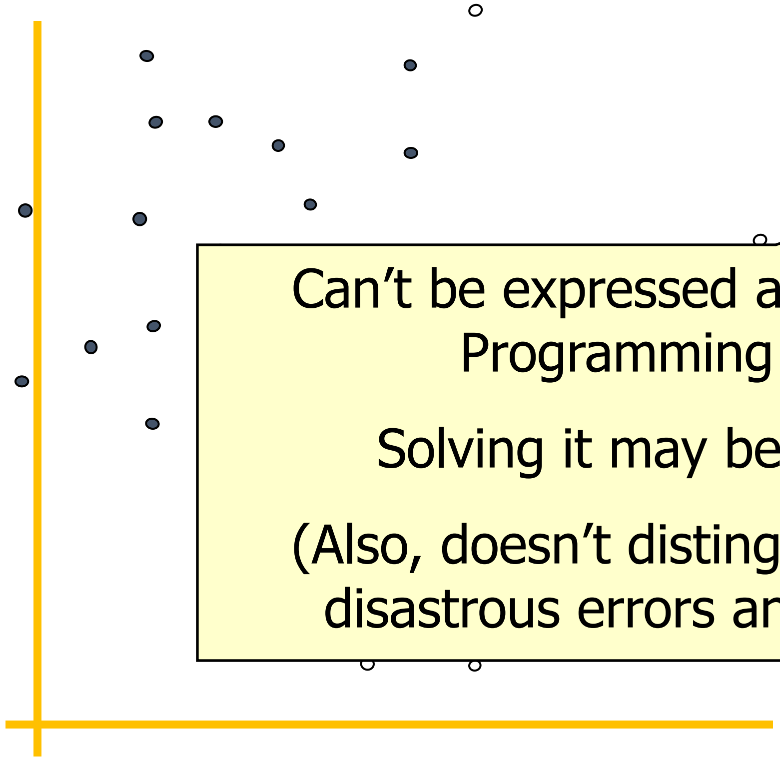
Minimize

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

Tradeoff parameter

There's a serious practical
m that's about to make
s approach. Can
what it is?

denotes +1
denotes -1



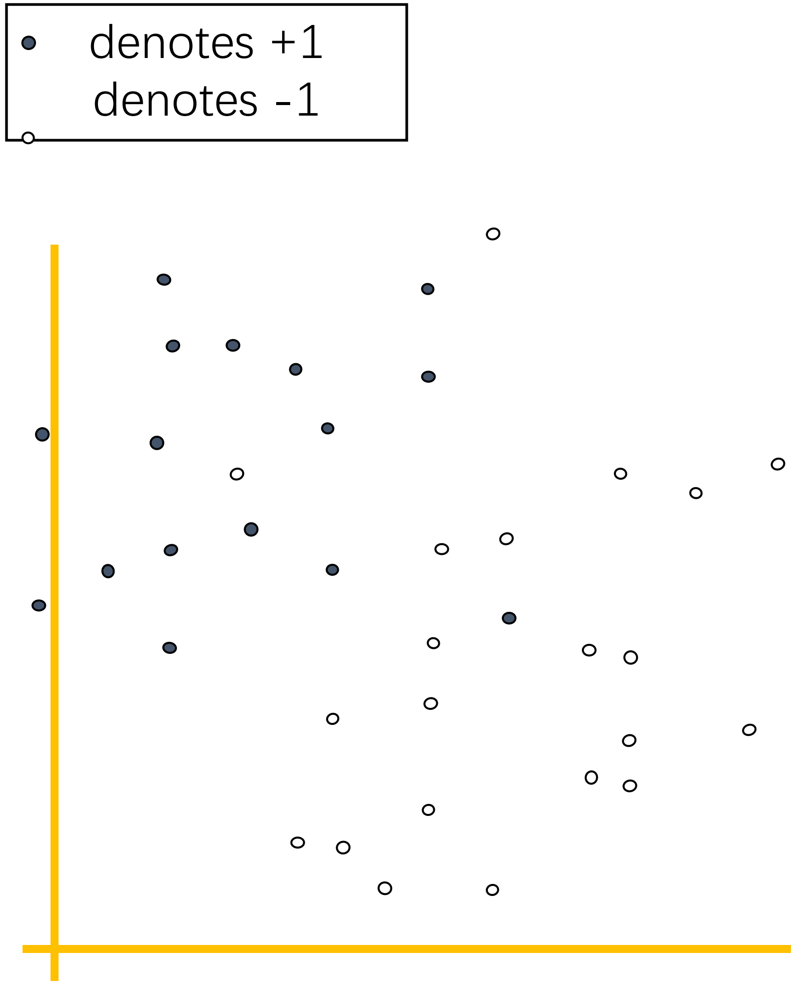
Can't be expressed as a Quadratic Programming problem.

Solving it may be too slow.

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

So… any other ideas?

Uh-oh!



This is going to be a problem!

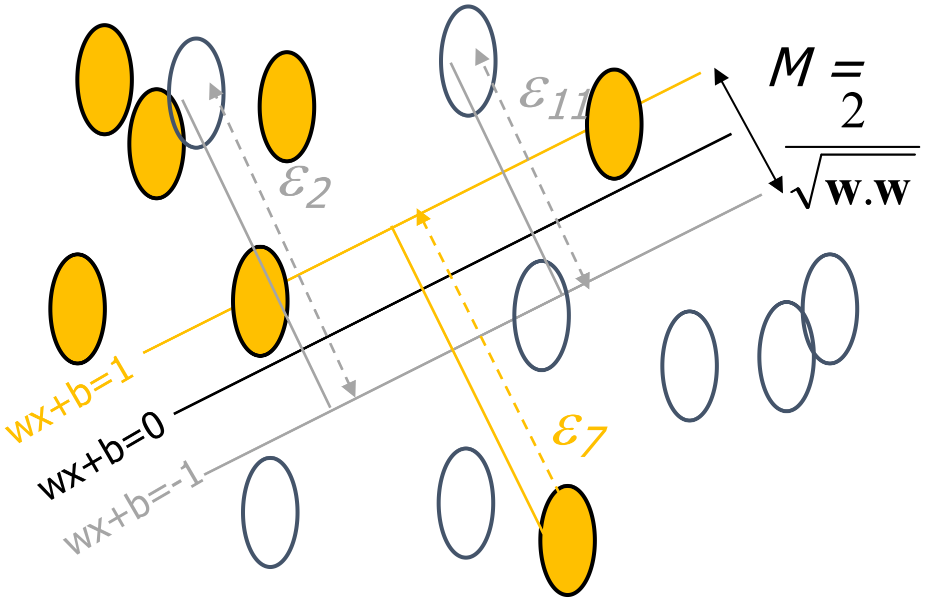
What should we do?

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 ($\boldsymbol{x}_k$,y_k) where y_k = +/- 1

What should our quadratic optimization criterion be?

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

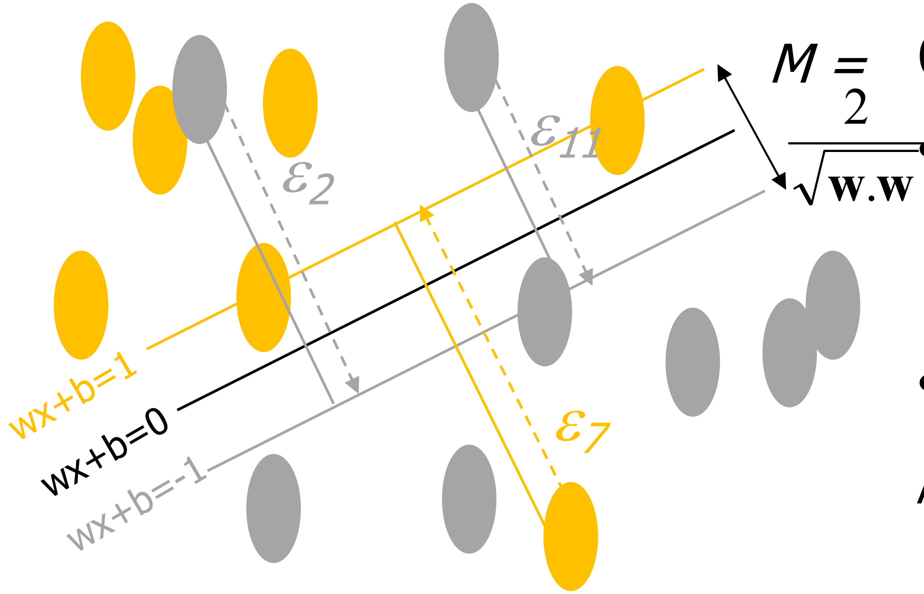
How many constraints will we have? *R*

What should they be?

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

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

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?

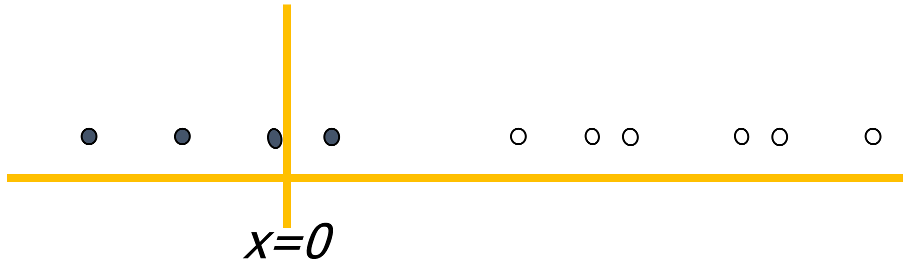
$\boldsymbol{w} \,.\, \boldsymbol{x}_k + b >= 1\text{-}\varepsilon_k \text{ if } y_k = 1$

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

$\varepsilon_k >= 0$ for all k

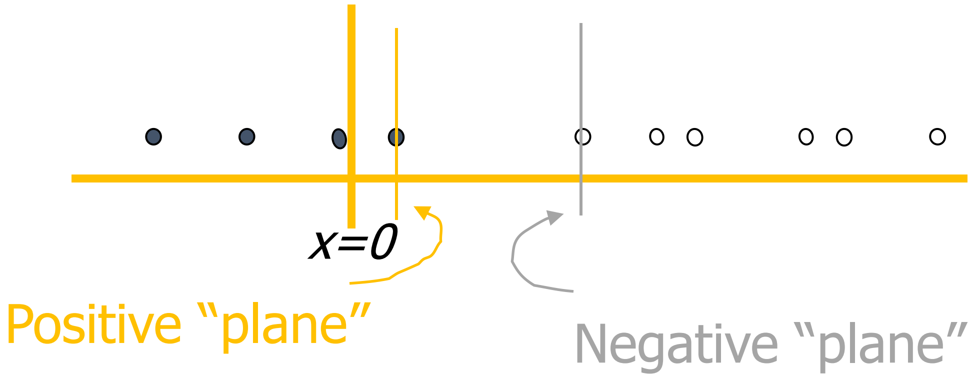
Suppose we're in 1-dimension

What would SVMs do with this data?

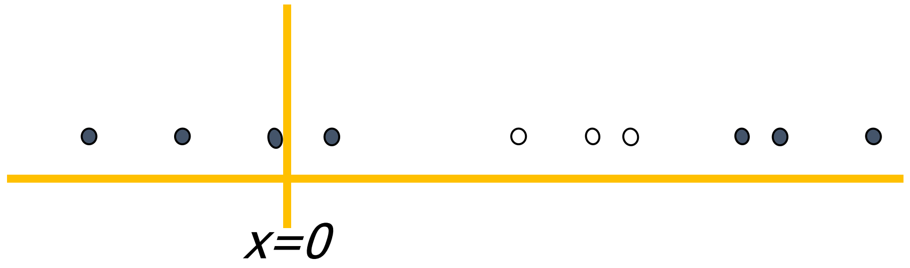


Suppose we're in 1-dimension

Not a big surprise



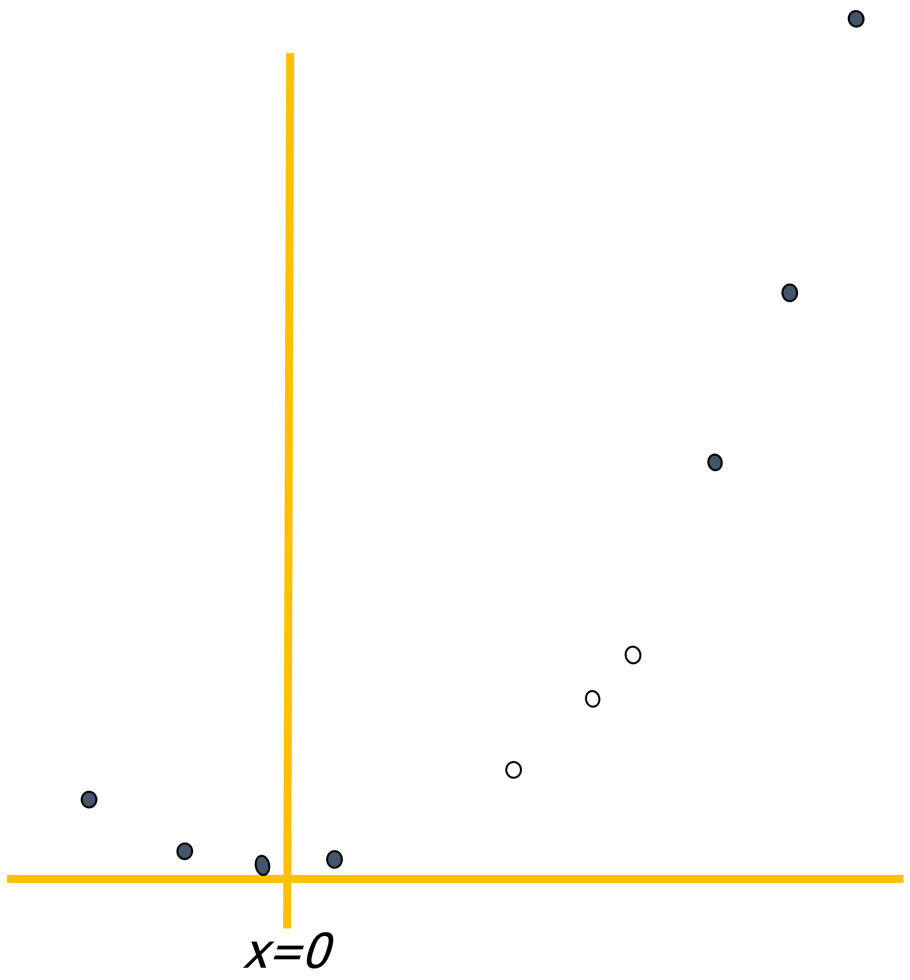
Harder 1-dimensional dataset



That's wiped the smirk off SVM's face.

What can be done about this?

Harder 1-dimensional dataset

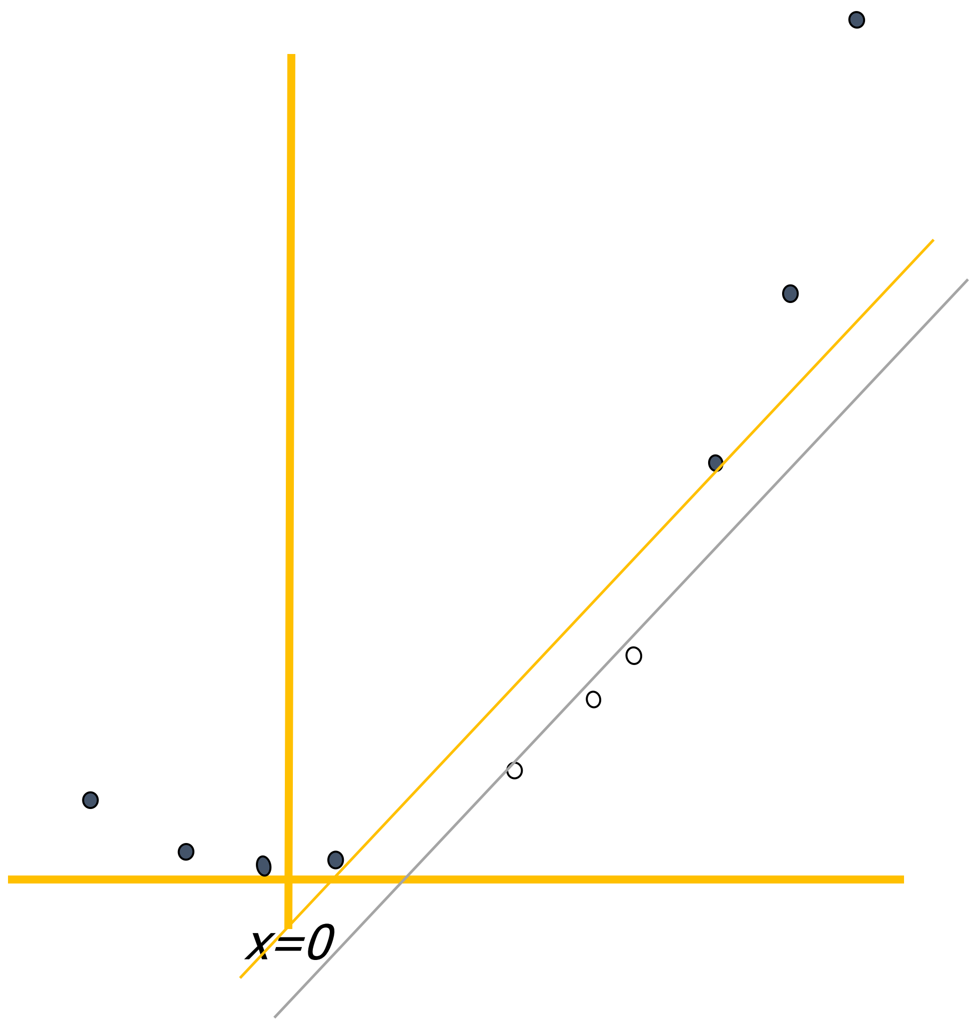


Remember how permitting non-linear basis functions made linear regression so much nicer?

Let's permit them here too

$$\mathbf{z}_k = (x_k, x_k^2)$$

Harder 1-dimensional dataset



Remember how permitting non-linear basis functions made linear regression so much nicer?

Let’s permit them here too

$$\mathbf{z}_k = (x_k, x_k^2)$$

Common SVM basis functions

$\boldsymbol{z}_k$ = (polynomial terms of $\boldsymbol{x}_k$ of degree 1 to q)

$\boldsymbol{z}_k$ = (radial basis functions of $\boldsymbol{x}_k$)

$$\mathbf{z}_k[j] = \varphi_j(\mathbf{x}_k) = \mathrm{KernelFn}\left(\frac{|\mathbf{x}_k - \mathbf{c}_j|}{\mathrm{KW}}\right)$$

$\boldsymbol{z}_k$ = (sigmoid functions of $\boldsymbol{x}_k$)

Quadratic Basis Functions

$$
\boldsymbol{\Phi}(\mathbf{x}) = \left(\begin{array}{c}
1 \\
\sqrt{2}x_1 \\
\sqrt{2}x_2 \\
\vdots \\
\sqrt{2}x_m \\
x_1^2 \\
x_2^2 \\
\vdots \\
x_m^2 \\
\sqrt{2}x_1x_2 \\
\sqrt{2}x_1x_3 \\
\vdots \\
\sqrt{2}x_1x_m \\
\sqrt{2}x_2x_3 \\
\vdots \\
\sqrt{2}x_1x_m \\
\vdots \\
\sqrt{2}x_{m-1}x_m
\end{array}\right)
\begin{array}{l}
\left.\vphantom{\begin{array}{c}1\end{array}}\right\}\text{Constant Term} \\
\left.\vphantom{\begin{array}{c}\sqrt{2}x_1\\\sqrt{2}x_2\\\vdots\\\sqrt{2}x_m\end{array}}\right\}\text{Linear Terms} \\
\left.\vphantom{\begin{array}{c}x_1^2\\x_2^2\\\vdots\\x_m^2\end{array}}\right\}\text{Pure Quadratic Terms} \\
\left.\vphantom{\begin{array}{c}\sqrt{2}x_1x_2\\\sqrt{2}x_1x_3\\\vdots\\\sqrt{2}x_1x_m\\\sqrt{2}x_2x_3\\\vdots\\\sqrt{2}x_1x_m\\\vdots\\\sqrt{2}x_{m-1}x_m\end{array}}\right\}\text{Quadratic Cross-Terms}
\end{array}
$$

SVM Kernel Functions

- $K(\boldsymbol{a},\boldsymbol{b})=(\boldsymbol{a} \cdot \boldsymbol{b} +1)^d$ 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:

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

  - Neural-net-style Kernel Function:

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

2019

怪兽学堂

THANKS