

Support Vector Machines

• Slides from:

• **Andrew W. Moore, CMU**

• (<http://www.cs.cmu.edu/~awm/tutorials>)

• **Nello Cristianini, BIOwulf Technologies**

• (<http://www.support-vector.net/tutorial.html>)

Overview

- Linear Classifier
- Margin
- Quadratic Programming
- Kernels

A Little History

- SVMs introduced in COLT-92 by Boser, Guyon, Vapnik. Greatly developed ever since.
- Initially popularized in the Neural Information Processing (NIPS) community, now an important and active field of all Machine Learning research.
- Special issues of Machine Learning Journal, and Journal of Machine Learning Research.
- Kernel Machines: large class of learning algorithms, SVMs a particular instance.

Very Informal Reasoning

- The class of kernel methods implicitly defines the class of possible patterns by introducing a notion of similarity between data
- Example: similarity between documents
 - By length
 - By topic
 - By language ...
- Choice of similarity -> Choice of relevant features

More formal reasoning

- Kernel methods exploit information about the inner products between data items
- Many standard algorithms can be rewritten so that they only require inner products between data (inputs)
- Kernel functions = inner products in some feature space (potentially very complex)
- If kernel given, no need to specify what features of the data are being used

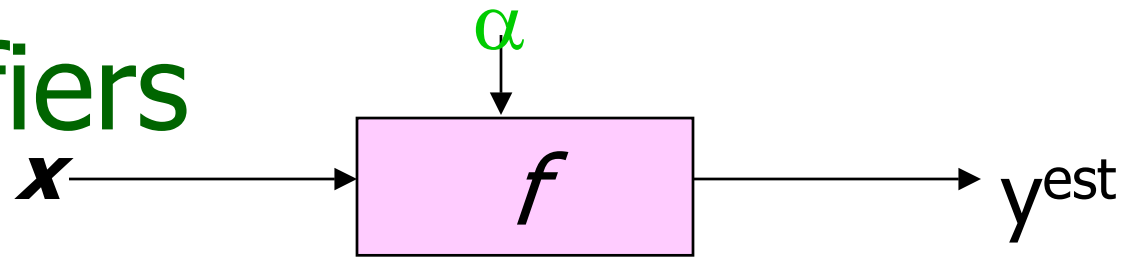
Modularity

- Any kernel-based learning algorithm composed of two modules:
 - A general purpose learning machine
 - A problem specific kernel function
- Any K-B algorithm can be fitted with any kernel
- Kernels themselves can be constructed in a modular way

Linear Learning Machines

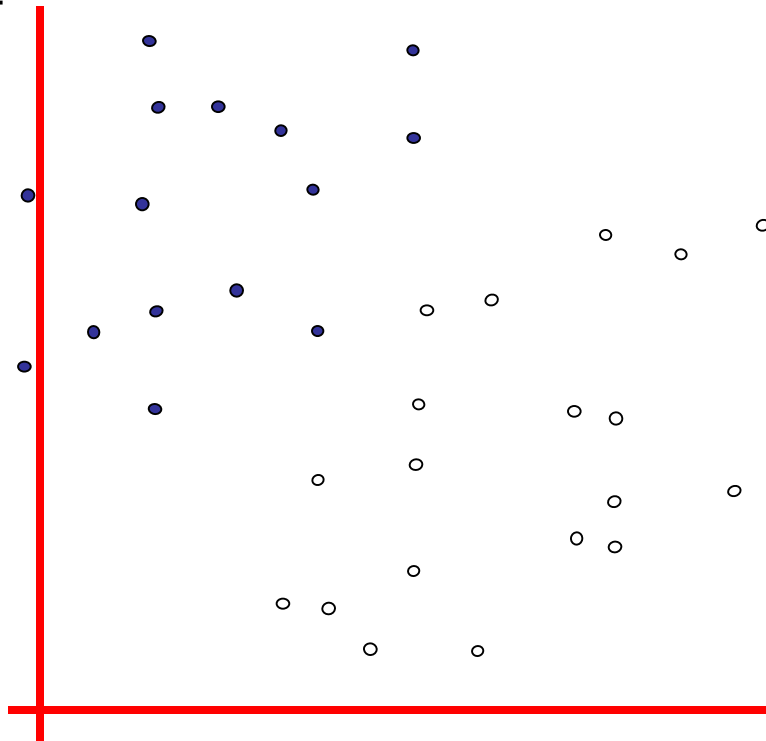
- Simplest case: classification. Decision function is a hyperplane in input space
- The Perceptron Algorithm (Rosenblatt, 57)
- Useful to analyze the Perceptron algorithm, before looking at SVMs and Kernel Methods in general

Linear Classifiers



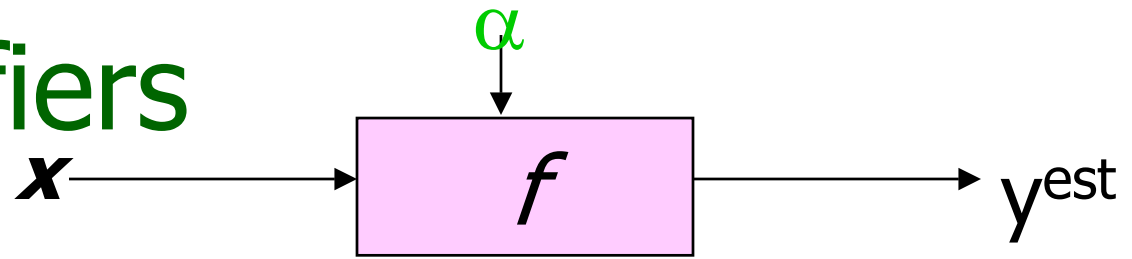
$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \mathbf{x} - b)$$

- denotes +1
- denotes -1

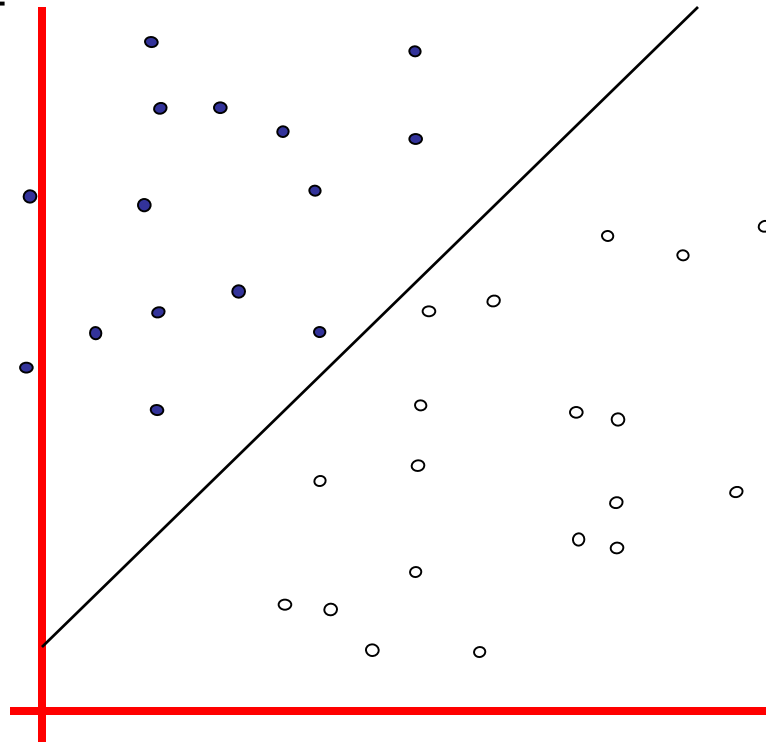


How would you classify this data?

Linear Classifiers



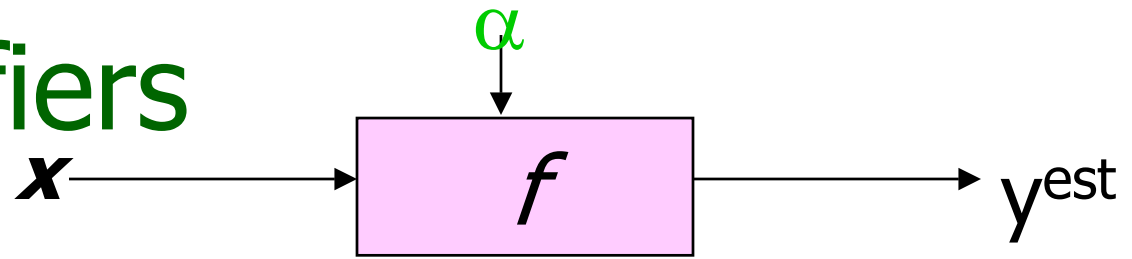
- denotes +1
- denotes -1



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

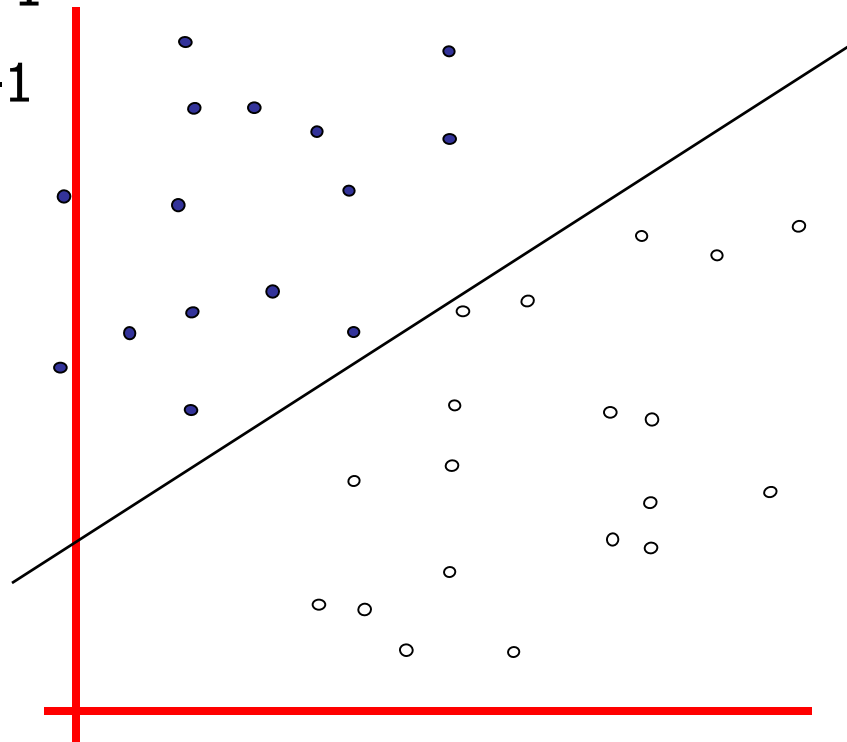
How would you classify this data?

Linear Classifiers



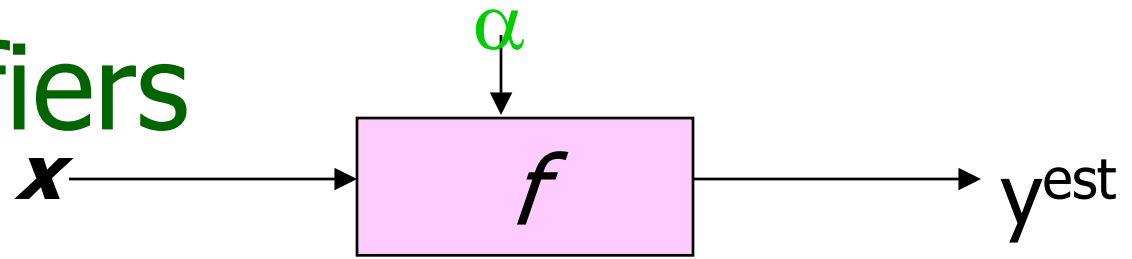
$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \mathbf{x} - b)$$

- denotes +1
- denotes -1

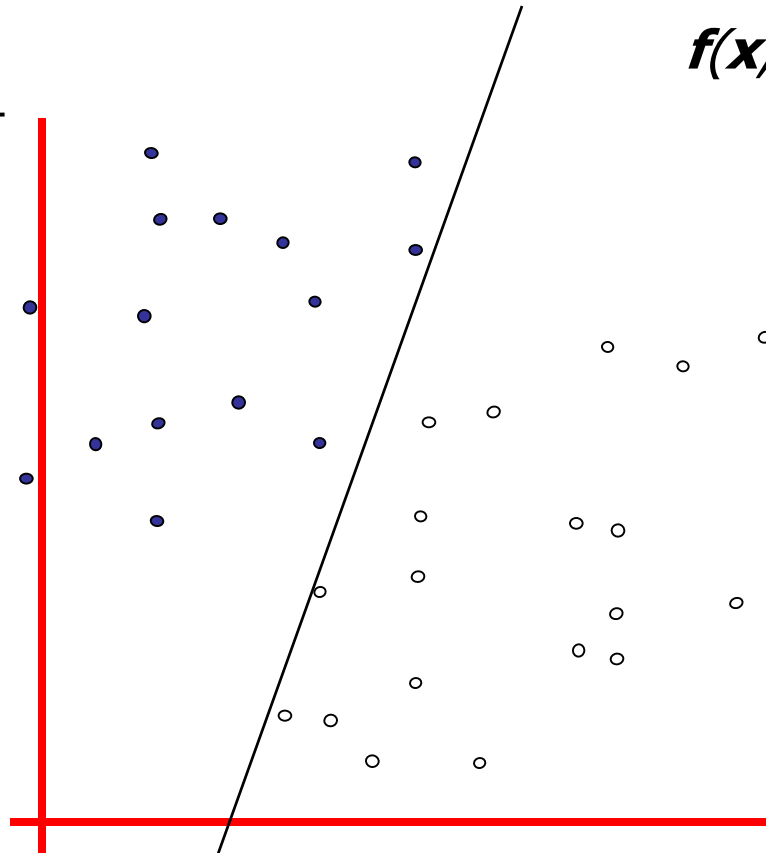


How would you classify this data?

Linear Classifiers



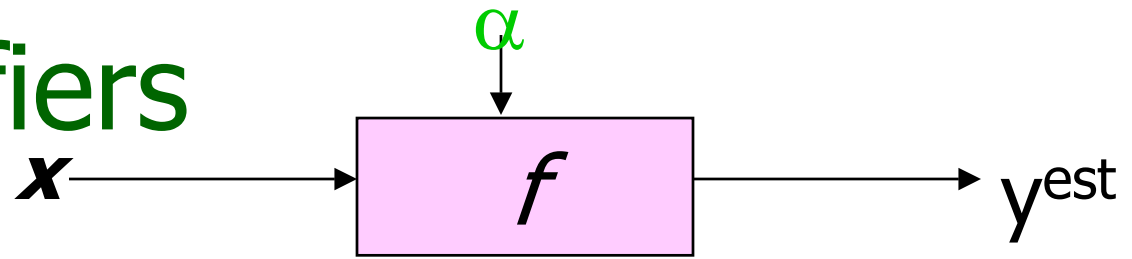
- denotes +1
- denotes -1



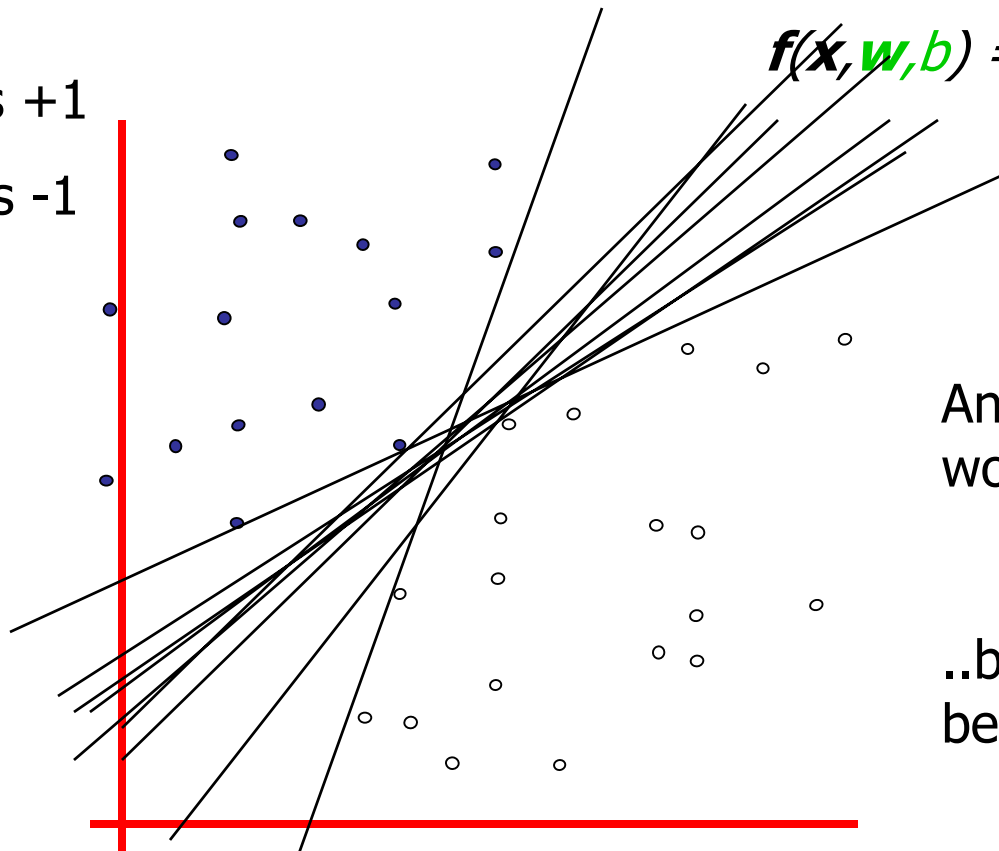
$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \mathbf{x} - b)$$

How would you classify this data?

Linear Classifiers



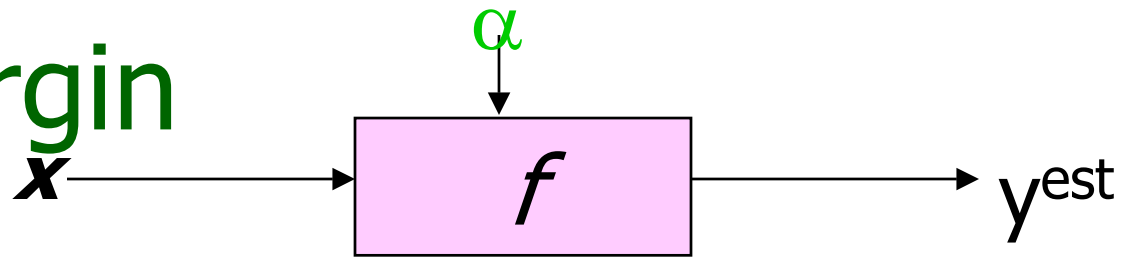
- denotes +1
- denotes -1



Any of these
would be fine..

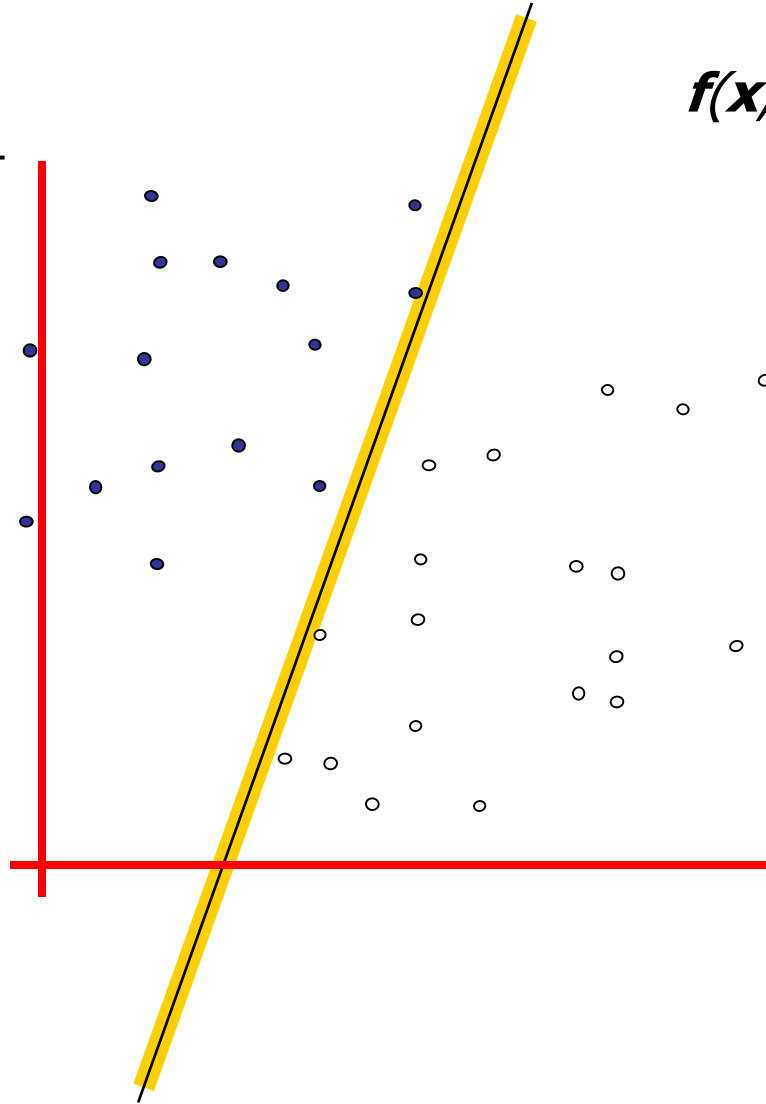
..but which is
best?

Classifier Margin



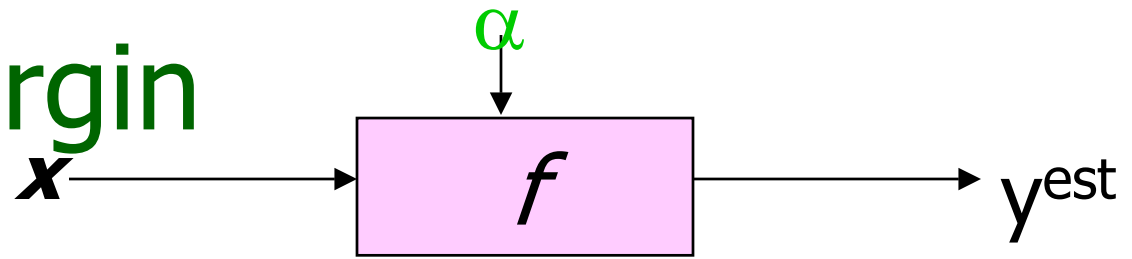
$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \mathbf{x} - b)$$

- denotes +1
- denotes -1

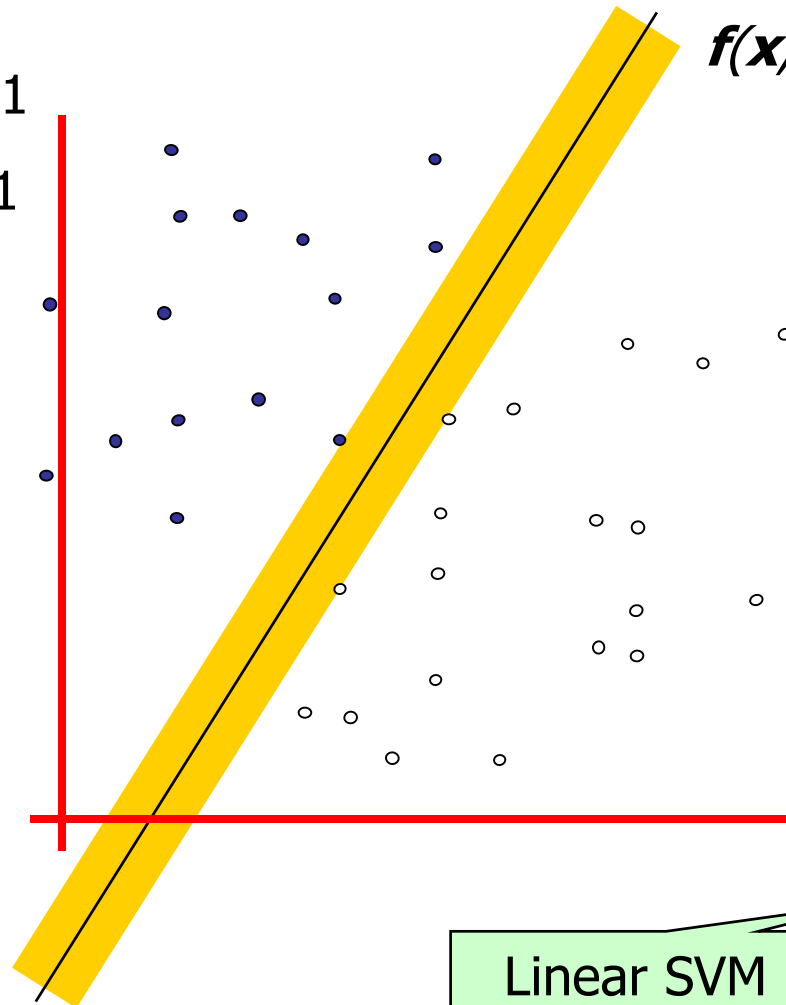


Define the **margin** of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

Maximum Margin



- denotes +1
- denotes -1



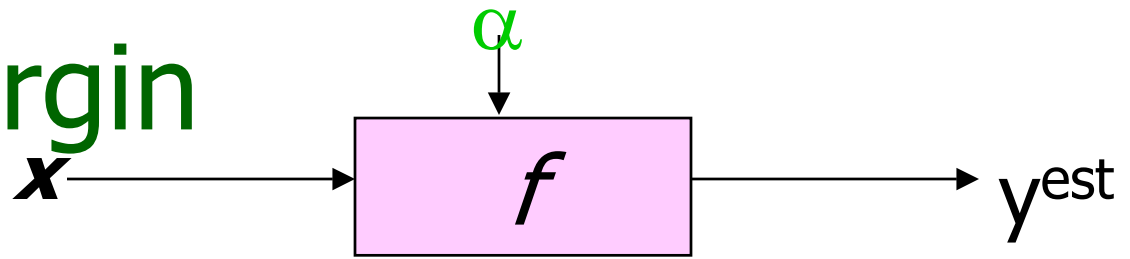
$$f(x, w, b) = \text{sign}(w \cdot x - b)$$

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

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

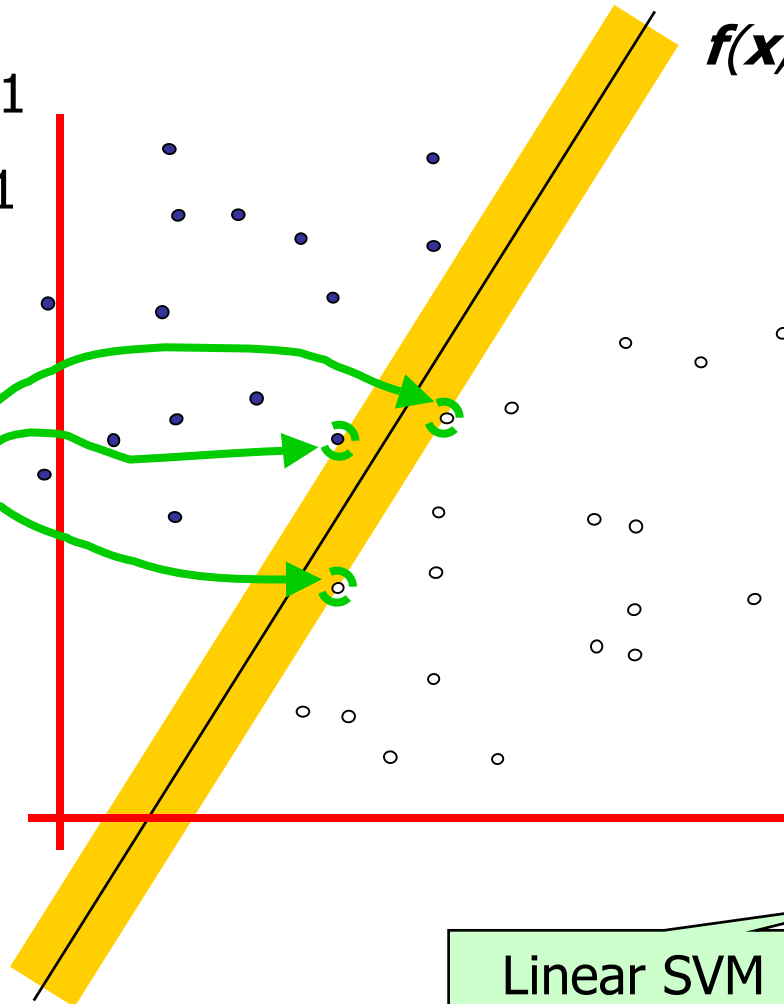
Linear SVM

Maximum Margin



- denotes +1
- denotes -1

Support Vectors
are those
datapoints that
the margin
pushes up
against



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

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

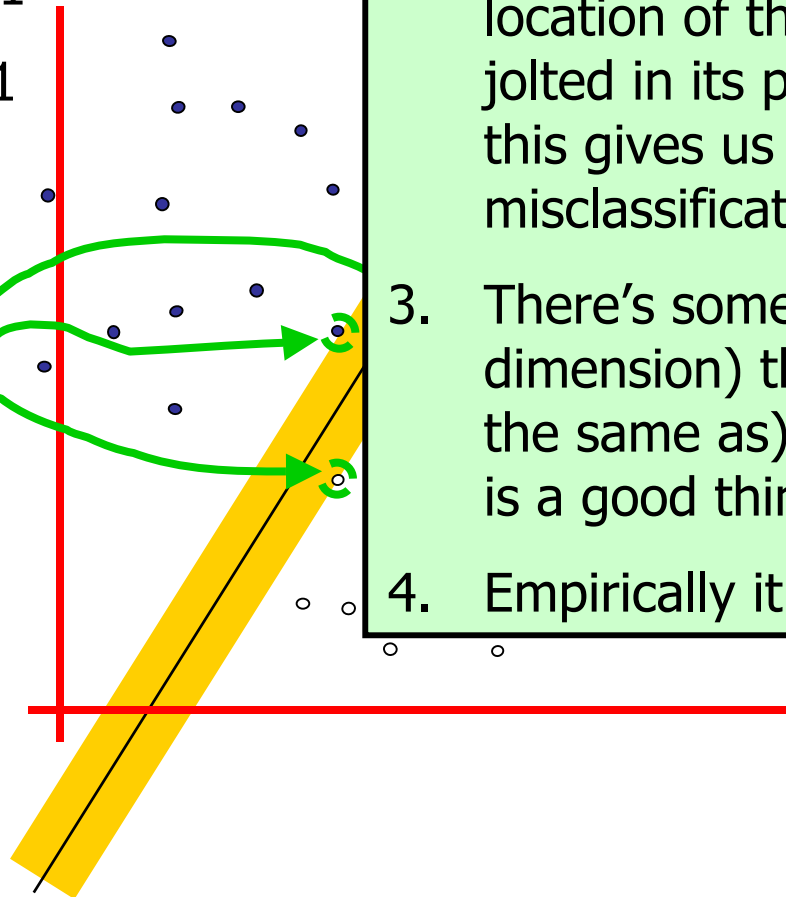
This is the simplest kind of SVM (Called an LSVM)

Linear SVM

Why Maximum Margin?

- denotes +1
- denotes -1

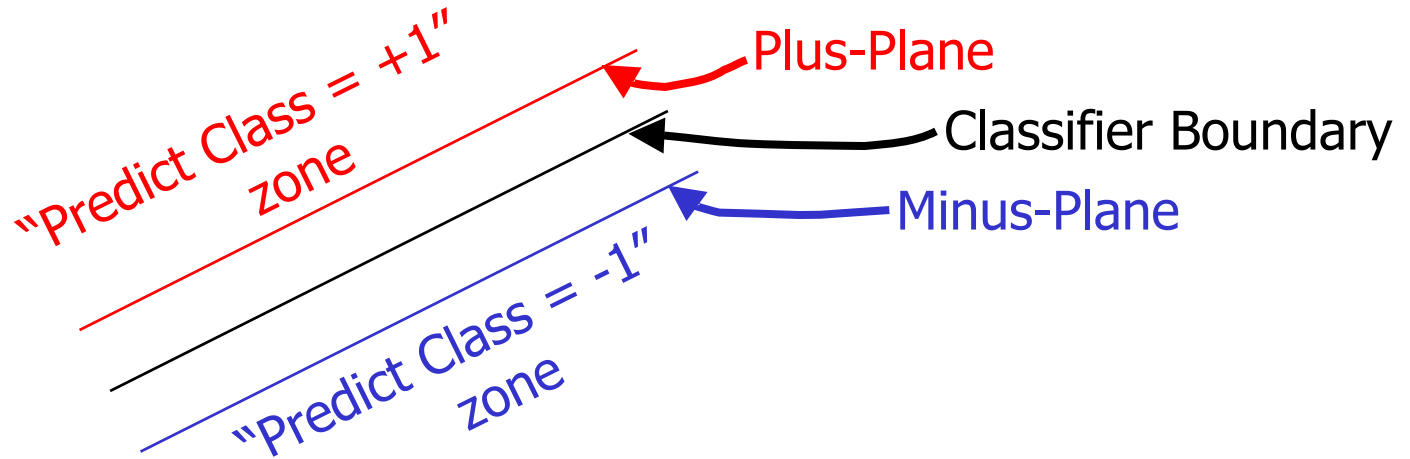
Support Vectors
are those
datapoints that
the margin
pushes up
against



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. 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.

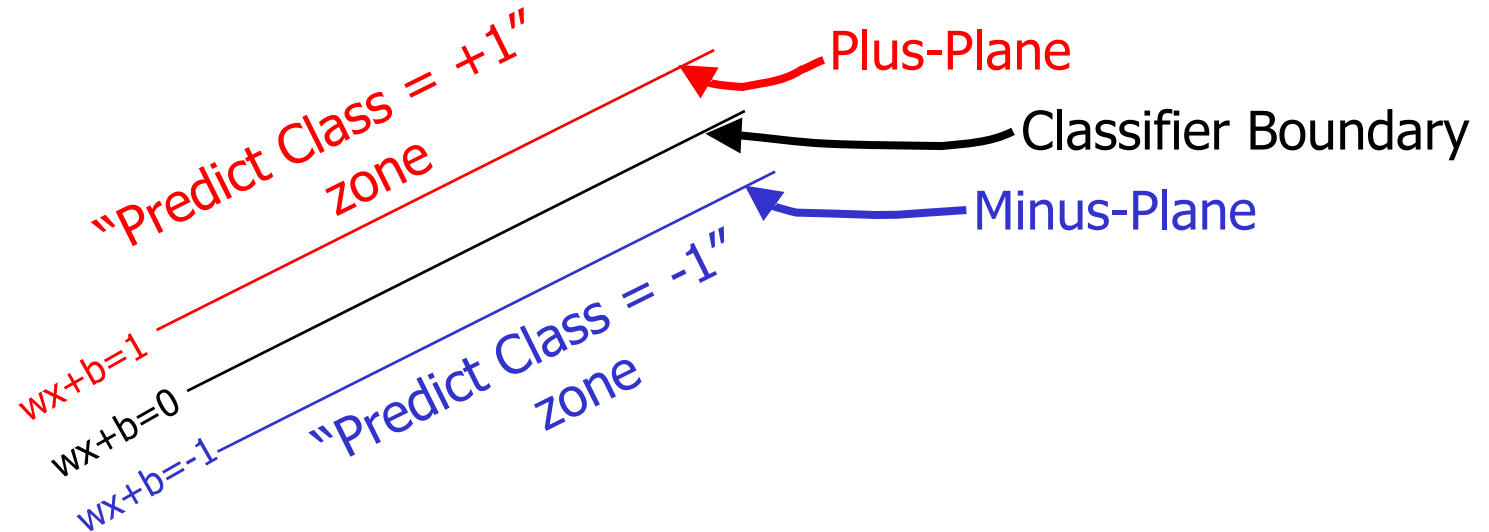
simplest kind of
SVM (Called an
LSVM)

Specifying a line and margin



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

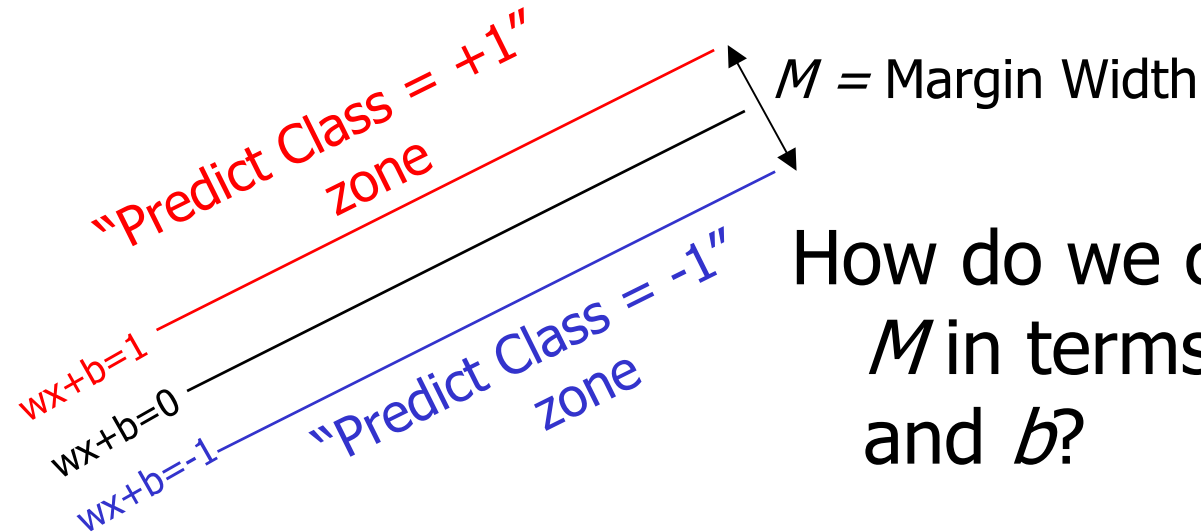
Specifying a line and margin



- 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 $\mathbf{w} \cdot \mathbf{x} + b \geq 1$
 -1 if $\mathbf{w} \cdot \mathbf{x} + b \leq -1$

Computing the margin width

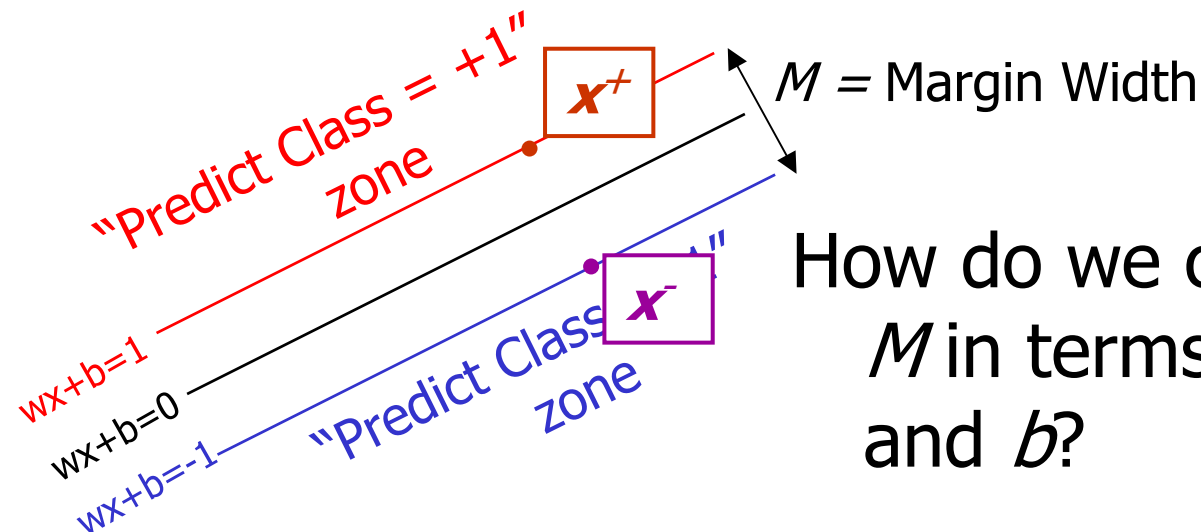


How do we compute M in terms of \mathbf{w} and b ?

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

The vector \mathbf{w} is perpendicular to the Plus Plane.

Computing the margin width

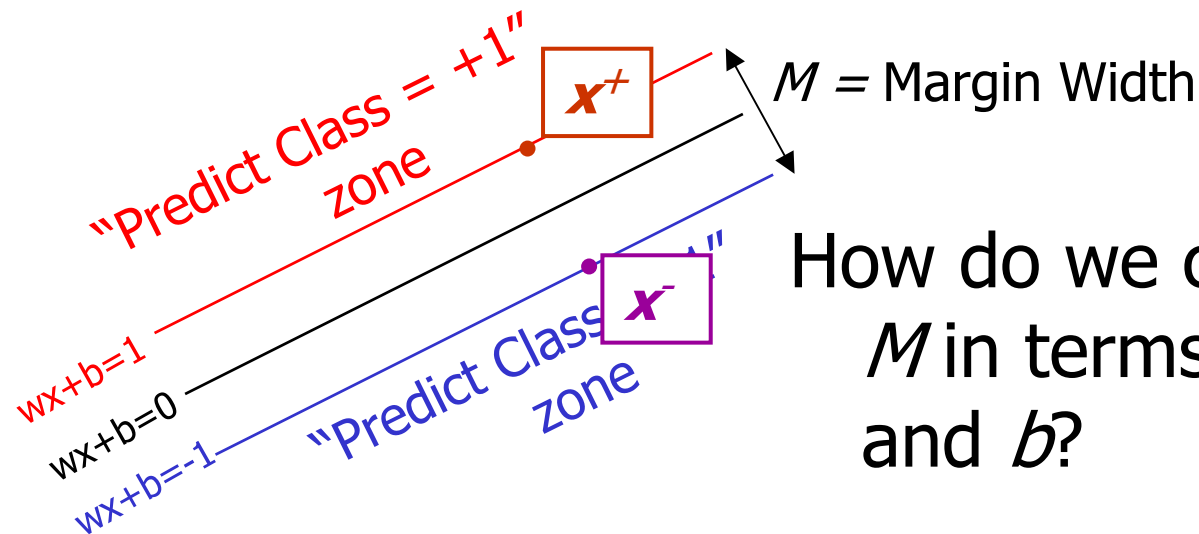


How do we compute M in terms of \mathbf{w} and b ?

- Plus-plane = $\{ \mathbf{x} : \mathbf{w} \cdot \mathbf{x} + b = +1 \}$
- Minus-plane = $\{ \mathbf{x} : \mathbf{w} \cdot \mathbf{x} + b = -1 \}$
- The vector \mathbf{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 \mathbb{R}^m : not necessarily a datapoint

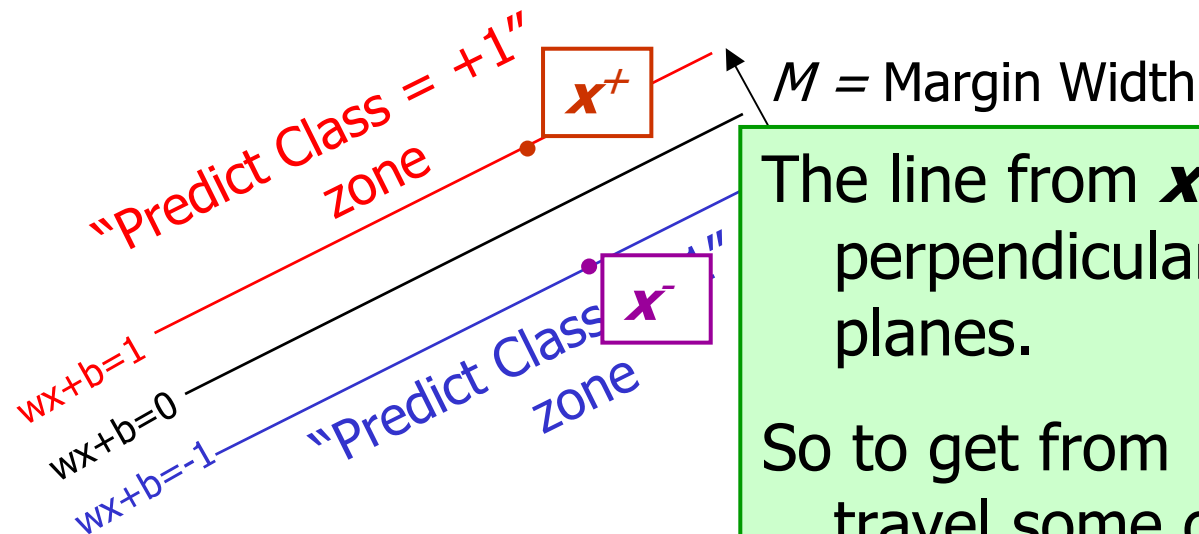
Computing the margin width



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 \mathbf{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}^- .
- **Claim:** $\mathbf{x}^+ = \mathbf{x}^- + \lambda \mathbf{w}$ for some value of λ . **Why?**

Computing the margin width

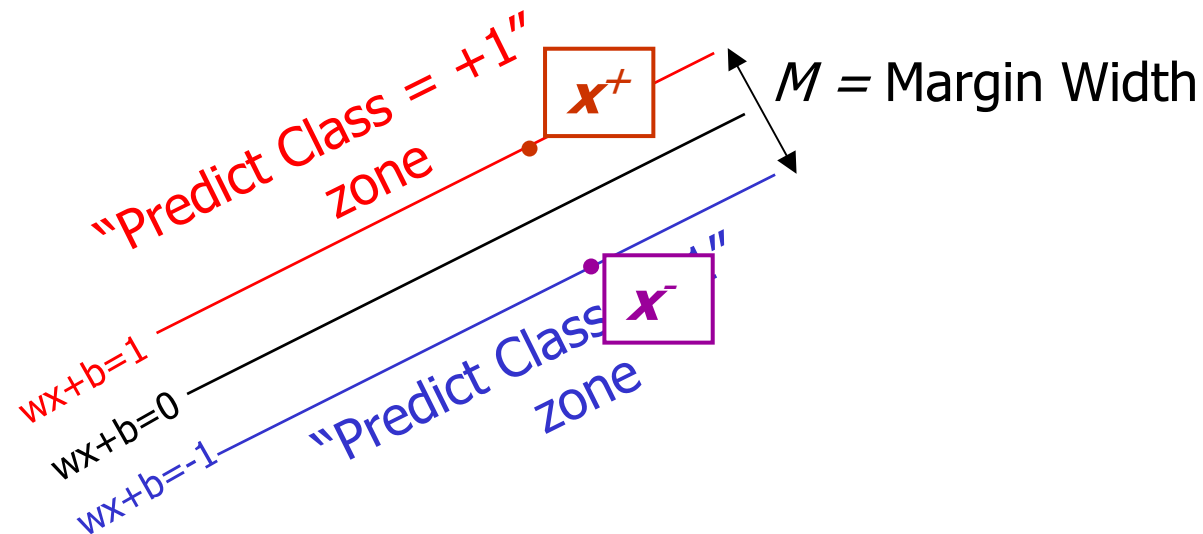


The line from x^- to x^+ is perpendicular to the planes.

So to get from x^- to x^+ travel some distance in direction w .

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

Computing the margin width

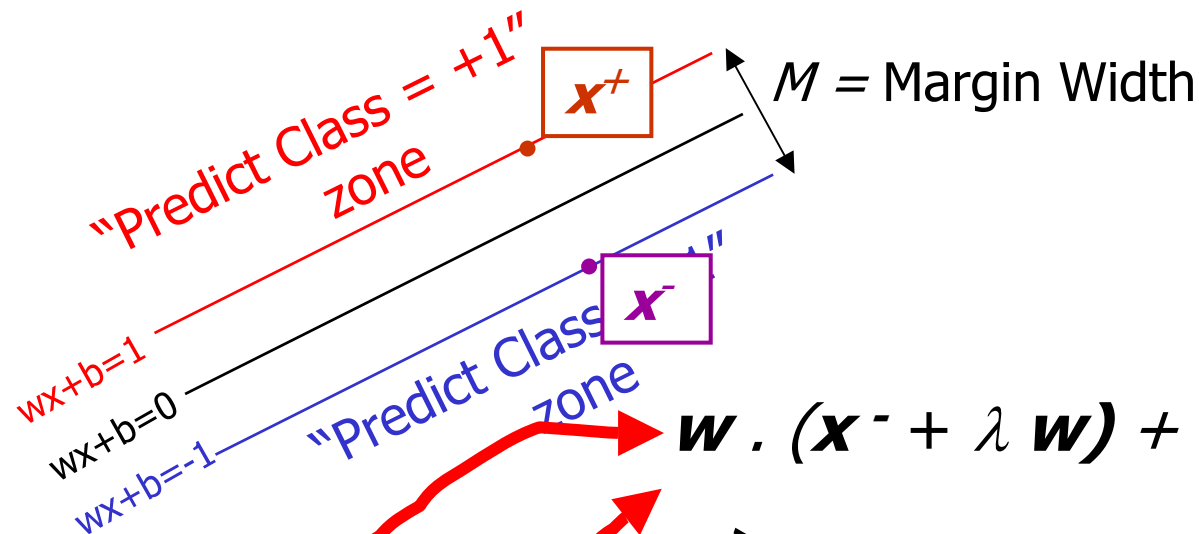


What we know:

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

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

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

\Rightarrow

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

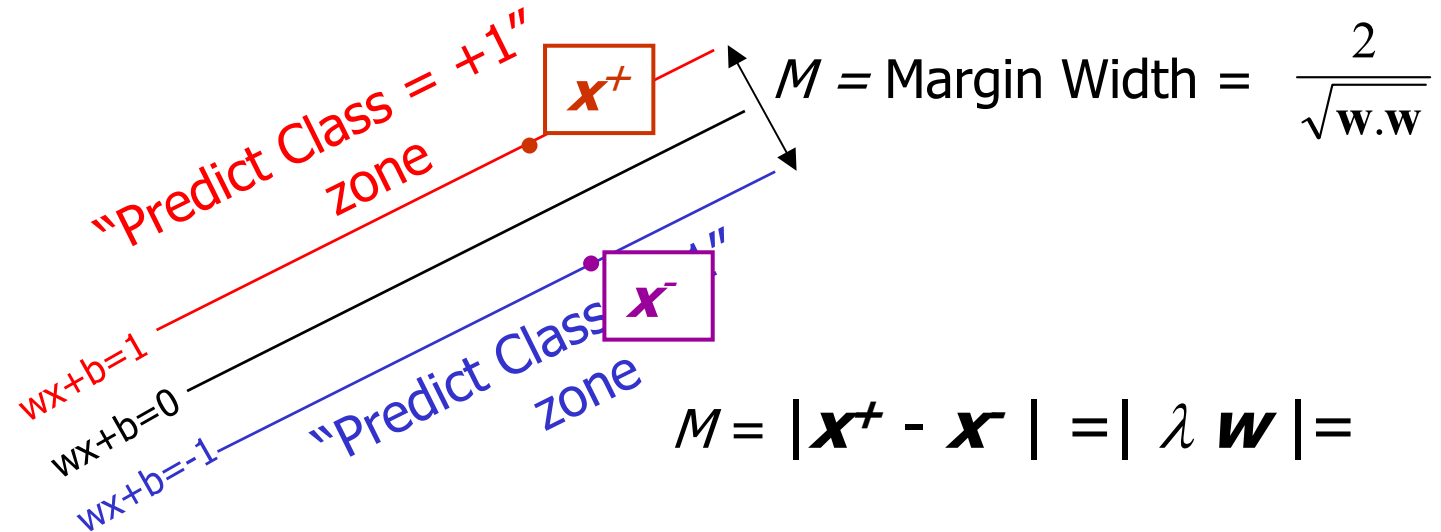
\Rightarrow

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

\Rightarrow

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

Computing the margin width



$$M = |x^+ - x^-| = |\lambda w| =$$

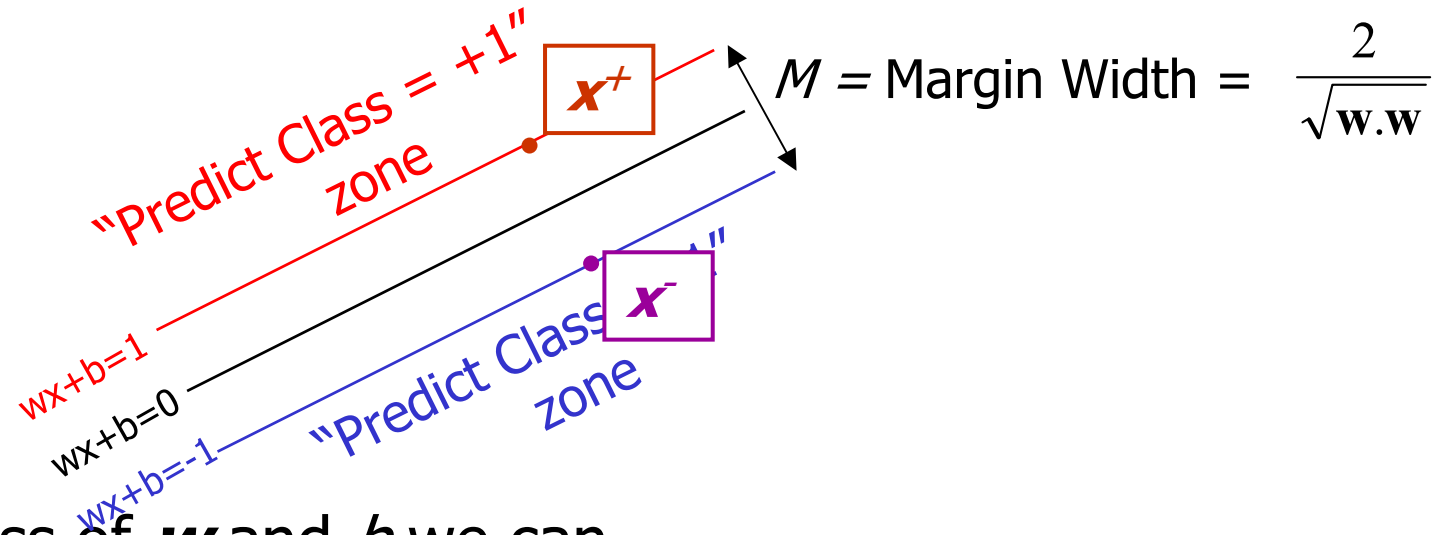
$$= \lambda |w| = \lambda \sqrt{w \cdot w}$$

$$= \frac{2\sqrt{w \cdot w}}{w \cdot w} = \frac{2}{\sqrt{w \cdot w}}$$

What we know:

- $w \cdot x^+ + b = +1$
- $w \cdot x^- + b = -1$
- $x^+ = x^- + \lambda w$
- $|x^+ - x^-| = M$
- $\lambda = \frac{2}{w \cdot w}$

Learning the Maximum Margin Classifier



Given a guess of \mathbf{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 \mathbf{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 $\arg \max_{\mathbf{u}} c + \mathbf{d}^T \mathbf{u} + \frac{\mathbf{u}^T R \mathbf{u}}{2}$  Quadratic criterion


Subject to

$$\begin{aligned} a_{11}u_1 + a_{12}u_2 + \dots + a_{1m}u_m &\leq b_1 \\ a_{21}u_1 + a_{22}u_2 + \dots + a_{2m}u_m &\leq b_2 \\ &\vdots \\ a_{n1}u_1 + a_{n2}u_2 + \dots + a_{nm}u_m &\leq b_n \end{aligned}$$

 n additional linear inequality constraints

And subject to

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

 e additional linear equality constraints

Quadratic Programming

Find $\arg \max_{\mathbf{u}} c + \mathbf{d}^T \mathbf{u} + \frac{\mathbf{u}^T R \mathbf{u}}{2}$ ← Quadratic criterion

Subject to

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

And subject to

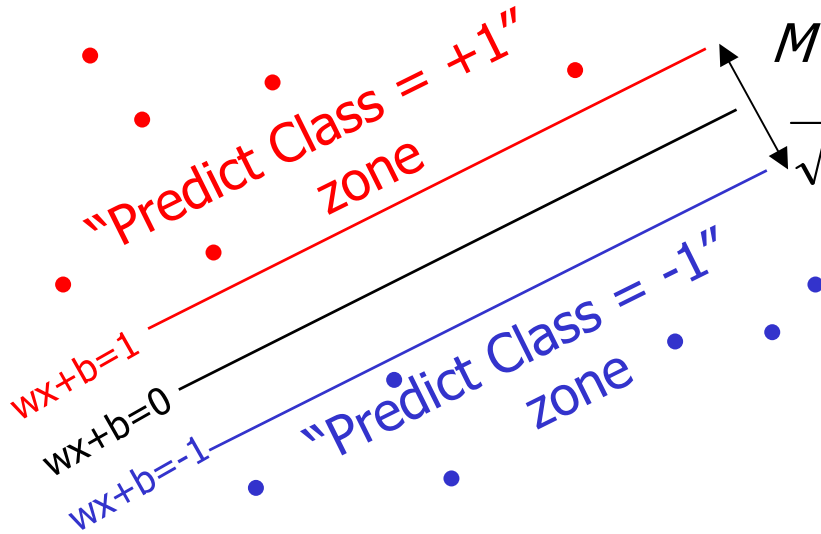
(But they are very fiddly...you probably don't want to write one yourself)

additional linear equality constraints

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

additional linear equality constraints

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

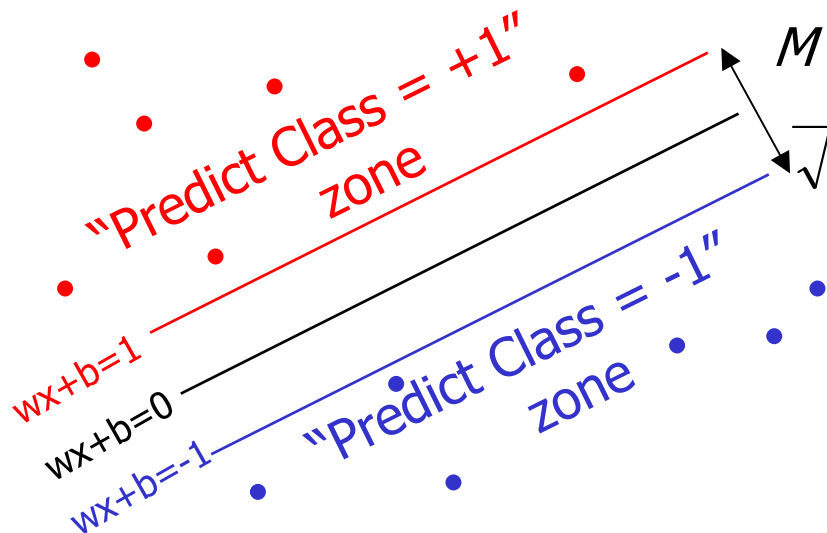
Assume R datapoints, each (\mathbf{x}_k, y_k) where $y_k = +/- 1$

What should our quadratic optimization criterion be?

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 $\mathbf{w} \cdot \mathbf{w}$

How many constraints will we have? R

What should they be?

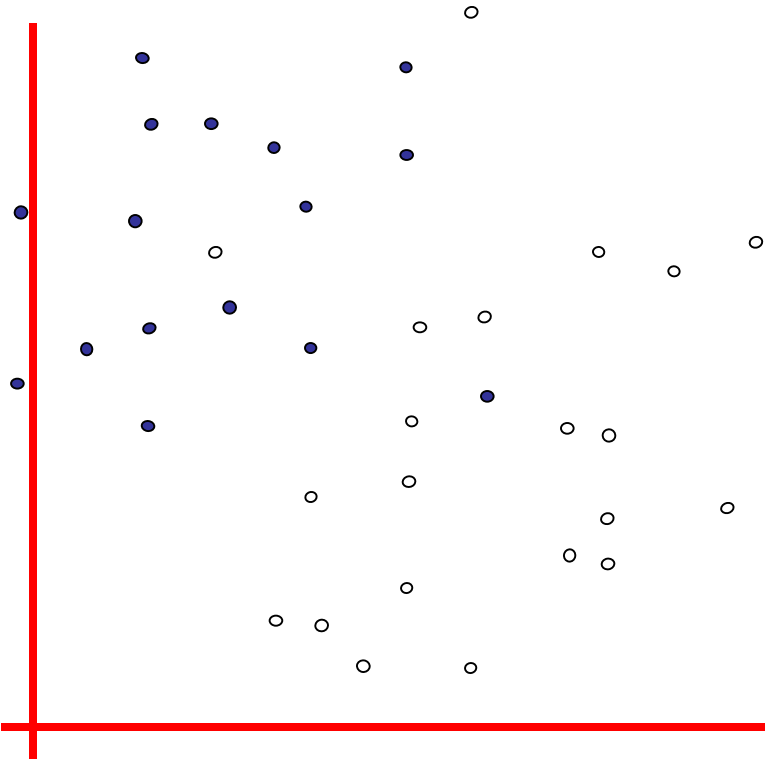
$$\mathbf{w} \cdot \mathbf{x}_k + b \geq 1 \text{ if } y_k = 1$$

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

Uh-oh!

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

- denotes +1
- denotes -1

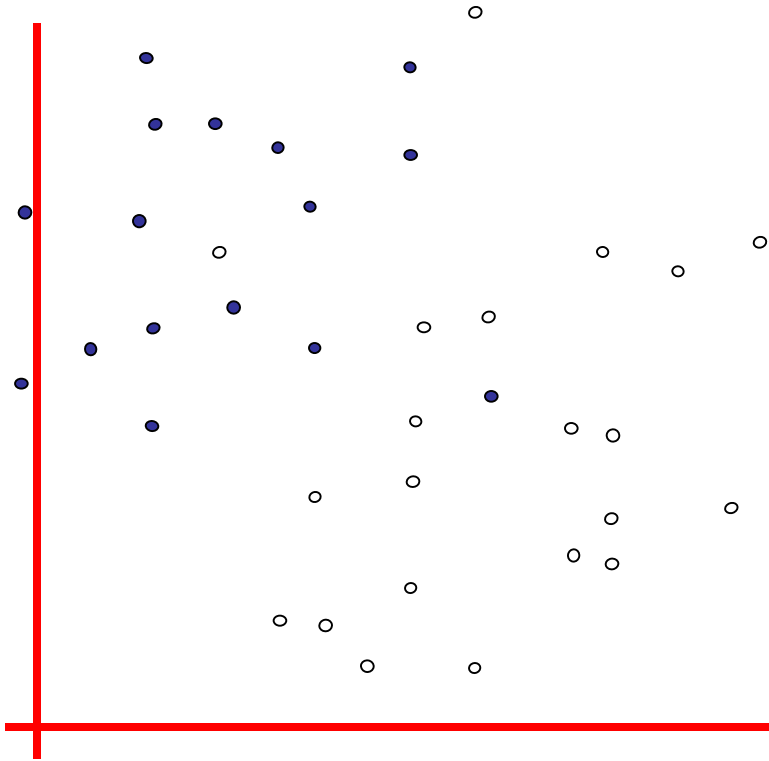


Uh-oh!

This is going to be a problem!

What should we do?

- denotes +1
- denotes -1



Idea 1:

Find minimum $\|w, w\|$, while minimizing number of training set errors.

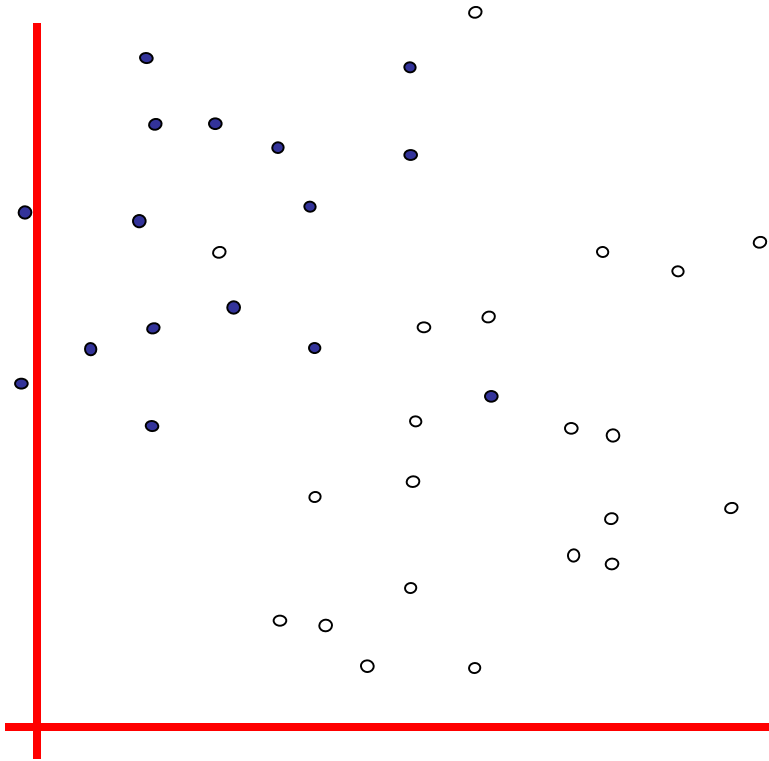
Problem: Two things to minimize makes for an ill-defined optimization

Uh-oh!

This is going to be a problem!

What should we do?

- denotes +1
- denotes -1



Idea 1.1:

Minimize

$w \cdot w + C (\#train\ errors)$

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?

- denotes +1
- denotes -1

Idea 1.1:

Minimize

$w \cdot w + C (\#train\ errors)$

Tradeoff parameter

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?

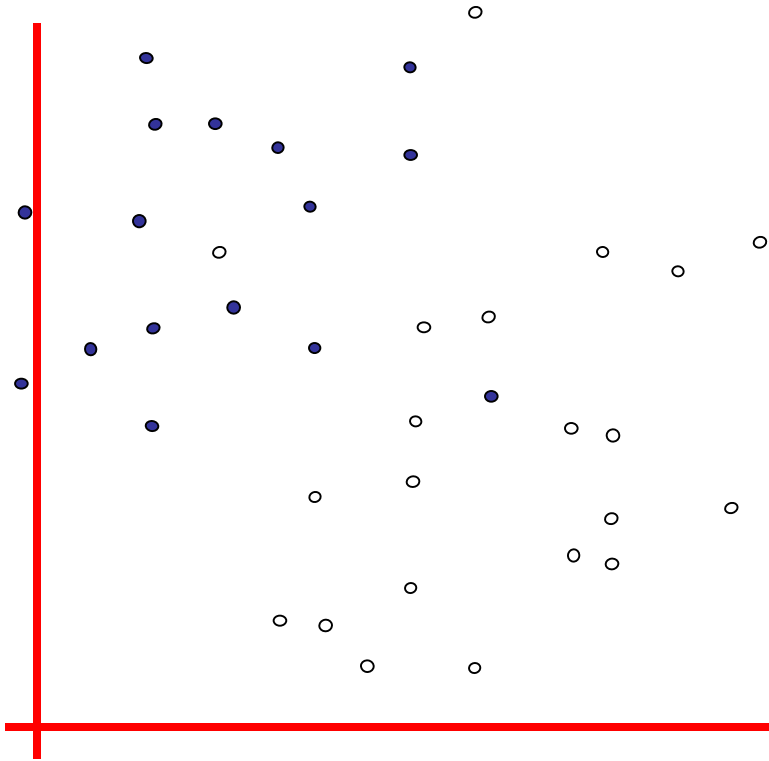
as reject this
you guess what?

Uh-oh!

This is going to be a problem!

What should we do?

- denotes +1
- denotes -1

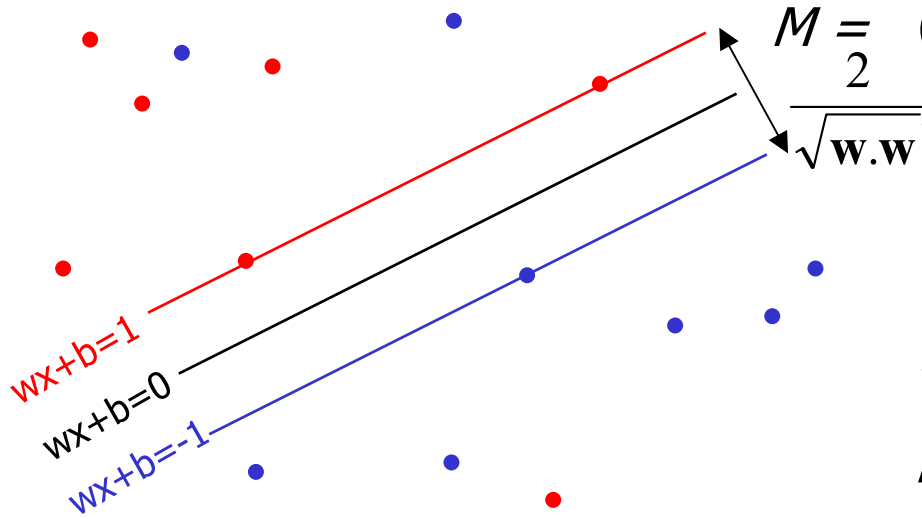


Idea 2.0:

Minimize

$W \cdot W + C$ (distance of error points to their correct place)

Learning Maximum Margin with Noise



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

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

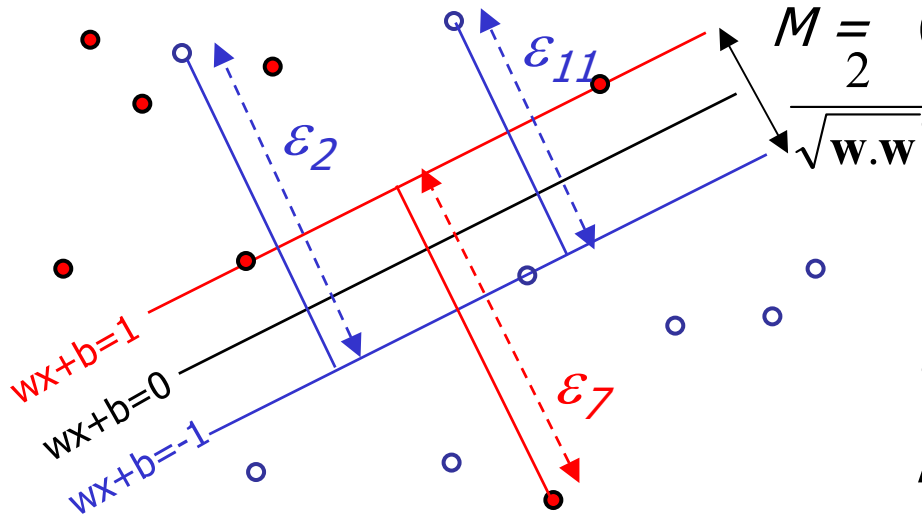
Assume R datapoints, each (\mathbf{x}_k, y_k) where $y_k = +/- 1$

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 \mathbf{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?

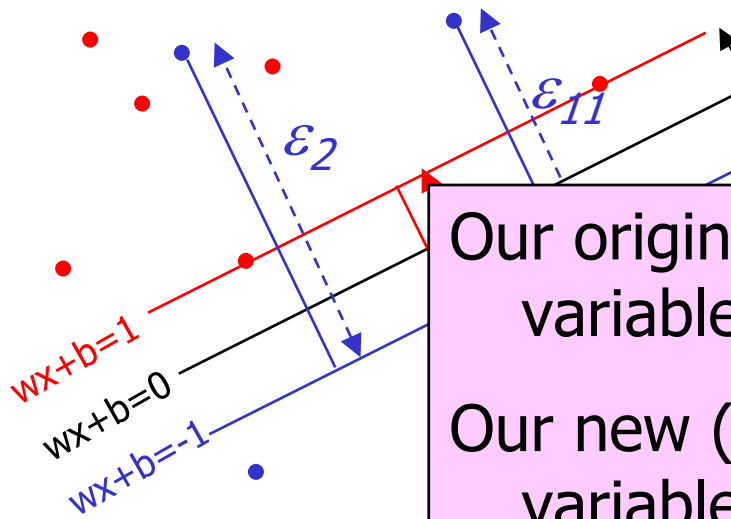
Minimize $\frac{1}{2} \mathbf{w} \cdot \mathbf{w} + C \sum_{k=1}^R \varepsilon_k$

How many constraints will we have? R

What should they be?

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

Learning Maximum Margin with Noise



$m = \#$ input dimensions

Given \mathbf{g} we can
 • Compute sum of distances

Our original (noiseless data) QP had $m+1$ variables: w_1, w_2, \dots, w_m and b .
 Our new (noisy data) QP has $m+1+R$ variables: $w_1, w_2, \dots, w_m, b, \epsilon_k, \epsilon_1, \dots, \epsilon_R$

What should our quadratic optimization criterion be?

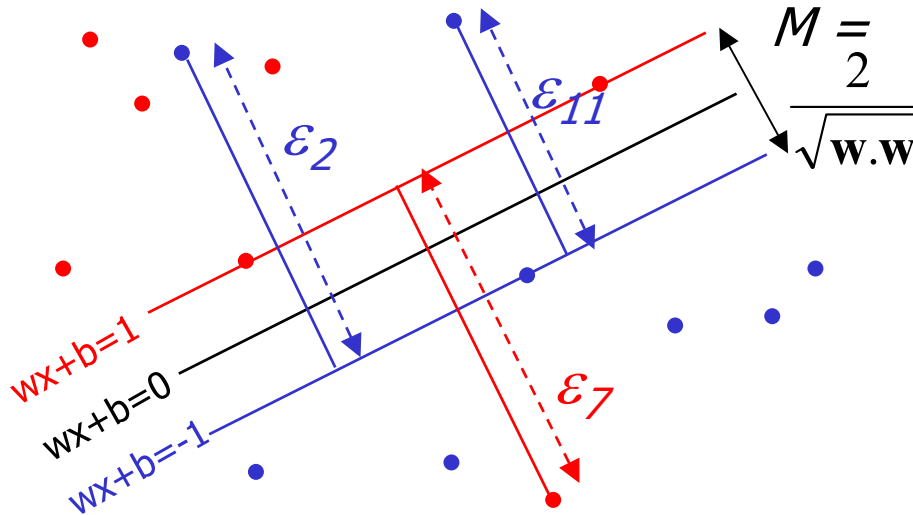
How many constraints do we have? R

$R = \#$ records

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

What should they be?
 $\mathbf{w} \cdot \mathbf{x}_k + b \geq 1 - \epsilon_k$ if $y_k = 1$
 $\mathbf{w} \cdot \mathbf{x}_k + b \leq -1 + \epsilon_k$ if $y_k = -1$

Learning Maximum Margin with Noise



Given guess of \mathbf{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

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

How many constraints will we have? R

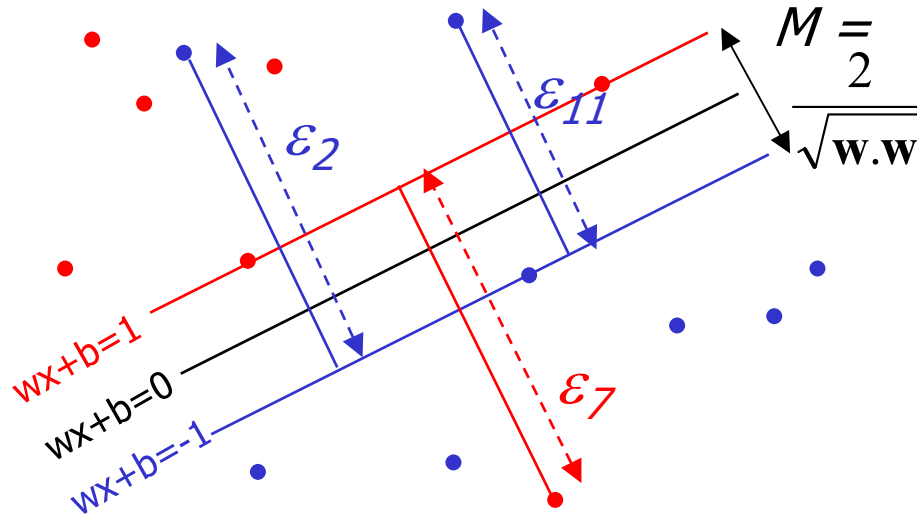
What should they be?

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

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

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

Learning Maximum Margin with Noise



Given guess of \mathbf{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
$$\frac{1}{2} \mathbf{w} \cdot \mathbf{w} + C \sum_{k=1}^R \varepsilon_k$$

How many constraints will we have? $2R$

What should they be?

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

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

$$\varepsilon_k \geq 0 \text{ for all } k$$

An Equivalent QP

$$\text{Maximize } \sum_{k=1}^R \alpha_k + \sum_{k=1}^R \sum_{l=1}^R \alpha_k \alpha_l Q_{kl} \quad \text{where } Q_{kl} = y_k y_l (\mathbf{x}_k \cdot \mathbf{x}_l)$$

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

Then define:

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

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

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

Then classify with:

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

An Equivalent QP

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

Subject to these constraints:

$$0 \leq \alpha_k \leq C \quad \forall k$$

$$\sum_{k=1}^R \alpha_k y_k = 0$$

Then define:

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

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

where $K = \arg \max_k \alpha_k$

Datapoints with $\alpha_k > 0$ will be the support vectors

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

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

An Equivalent QP

Maximize $\sum_{k=1}^R \alpha_k$

Subject to

constraints

Why did I tell you about this equivalent QP?

- It's a formulation that QP packages can optimize more quickly
- Because of further jaw-dropping developments you're about to learn.

Then

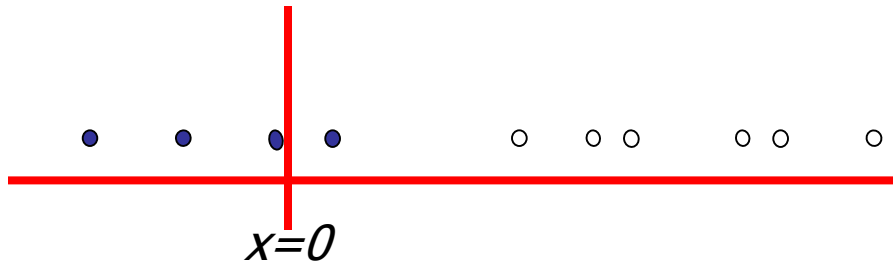
$\mathbf{w} = \sum_{k=1}^R \alpha_k \mathbf{x}_k$

$b = y_K (1 - \alpha_K)$

where $K = \arg \max_k \alpha_k$

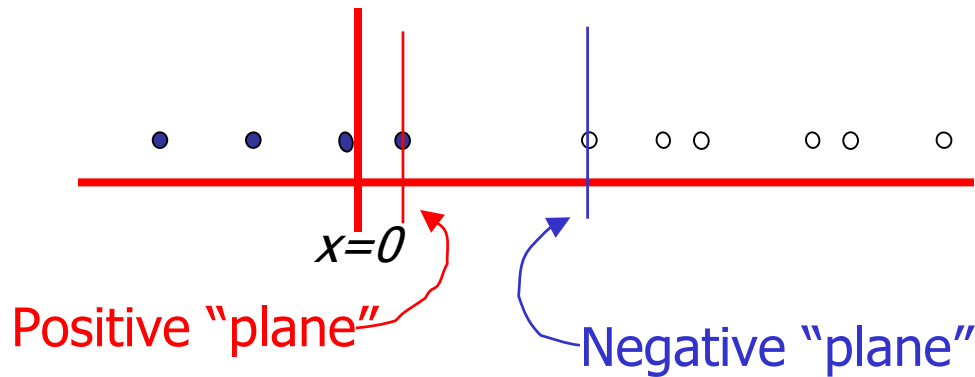
Suppose we're in 1-dimension

What would
SVMs do with
this data?



Suppose we're in 1-dimension

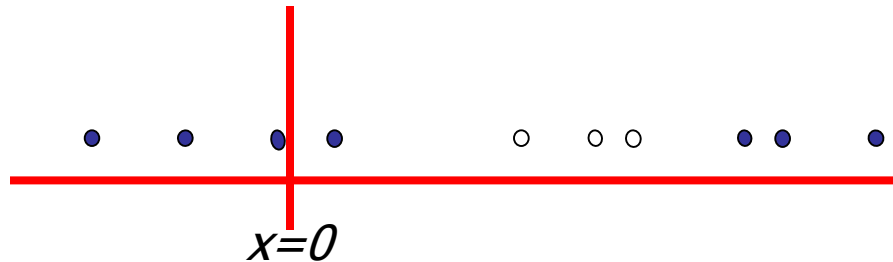
Not a big surprise



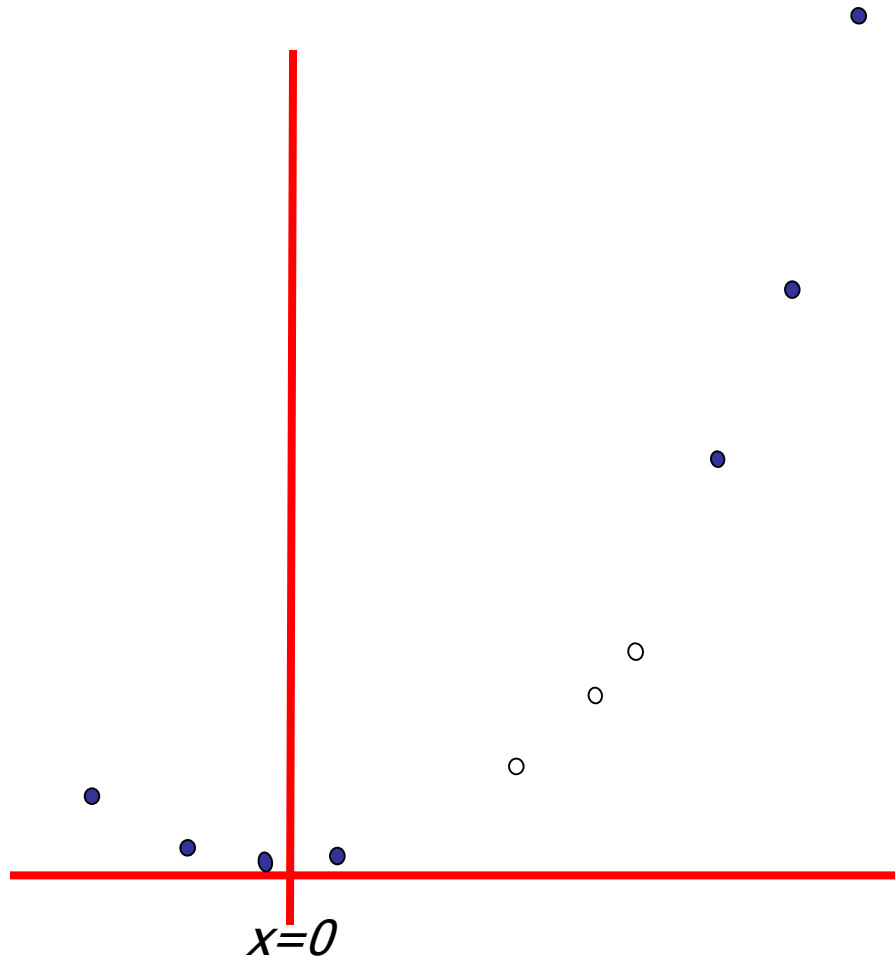
Harder 1-dimensional dataset

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

What can be done about this?



Harder 1-dimensional dataset

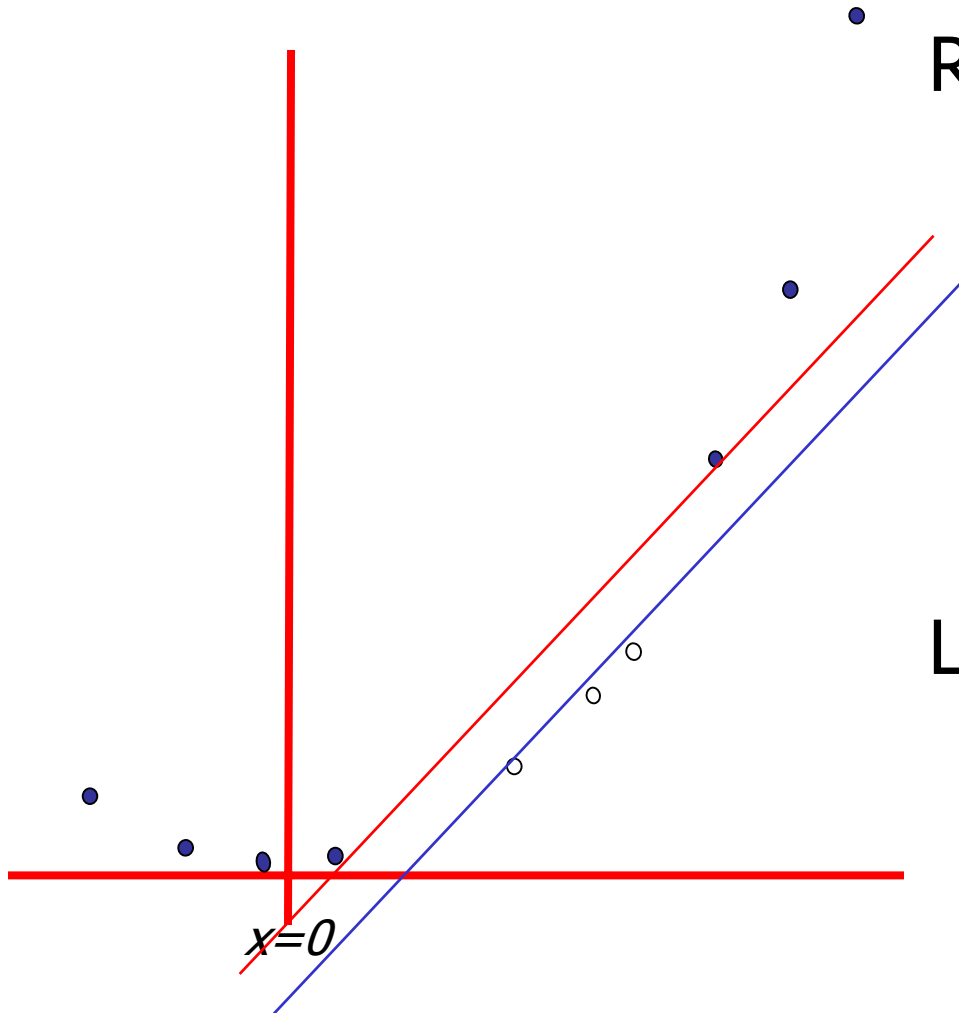


Remember how permitting non-linear basis functions made linear regression so much nicer?

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

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

Problems with Feature Space

- Working in high dimensional feature spaces solves the problem of expressing complex functions
- BUT:
 - There is a computational problem (working with very large vectors)
 - And a generalization theory problem (curse of dimensionality)

Implicit Mapping to Feature Space

- We will introduce Kernels:
 - Solve the computational problem of working with many dimensions
 - Can make it possible to use infinite dimensions
 - efficiently in time / space
 - Other advantages, both practical and conceptual

Common SVM basis functions

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

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

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

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

This is sensible.

Is that the end of the story?

No...there's one more trick!

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) = $(m+2)$ -choose-2

$$= (m+2)(m+1)/2$$

$$= (\text{as near as makes no difference}) m^2/2$$

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

- You should be happy that they do no harm
- You'll find out why they're there soon.

QP with basis functions

$$\text{Maximize } \sum_{k=1}^R \alpha_k + \sum_{k=1}^R \sum_{l=1}^R \alpha_k \alpha_l Q_{kl} \quad \text{where } Q_{kl} = y_k y_l (\Phi(\mathbf{x}_k) \cdot \Phi(\mathbf{x}_l))$$

Subject to these constraints:

$$0 \leq \alpha_k \leq C \quad \forall k$$

$$\sum_{k=1}^R \alpha_k y_k = 0$$

Then define:

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

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

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

Then classify with:

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

QP with basis functions

$$\text{Maximize } \sum_{k=1}^R \alpha_k + \sum_{k=1}^R \sum_{l=1}^R \alpha_k \alpha_l Q_{kl} \quad \text{where } Q_{kl} = y_k y_l (\Phi(\mathbf{x}_k) \cdot \Phi(\mathbf{x}_l))$$

Subject to these constraints:

$$0 \leq \alpha_k \leq 1$$

Then define:

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

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

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

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 $R^2 m^2 / 4$.
Yeeks!

...or does it?

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

Quadratic Dot Products

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

$$\left(\begin{array}{c} 1 \\ \sqrt{2}a_1 \\ \sqrt{2}a_2 \\ \vdots \\ \sqrt{2}a_m \\ a_1^2 \\ a_2^2 \\ \vdots \\ a_m^2 \\ \sqrt{2}a_1a_2 \\ \sqrt{2}a_1a_3 \\ \vdots \\ \sqrt{2}a_1a_m \\ \sqrt{2}a_2a_3 \\ \vdots \\ \sqrt{2}a_1a_m \\ \vdots \\ \sqrt{2}a_{m-1}a_m \end{array} \right) \bullet \left(\begin{array}{c} 1 \\ \sqrt{2}b_1 \\ \sqrt{2}b_2 \\ \vdots \\ \sqrt{2}b_m \\ b_1^2 \\ b_2^2 \\ \vdots \\ b_m^2 \\ \sqrt{2}b_1b_2 \\ \sqrt{2}b_1b_3 \\ \vdots \\ \sqrt{2}b_1b_m \\ \sqrt{2}b_2b_3 \\ \vdots \\ \sqrt{2}b_1b_m \\ \vdots \\ \sqrt{2}b_{m-1}b_m \end{array} \right)$$

$$\left. \begin{array}{l} 1 \\ + \\ \sum_{i=1}^m 2a_i b_i \\ + \\ \sum_{i=1}^m a_i^2 b_i^2 \\ + \\ \sum_{i=1}^m \sum_{j=i+1}^m 2a_i a_j b_i b_j \end{array} \right\}$$

Quadratic Dot Products

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

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

Just out of casual, innocent, interest, let's look at another function of \mathbf{a} and \mathbf{b} :

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

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

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

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

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

Quadratic Dot Products

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

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

Just out of casual, innocent, interest, let's look at another function of \mathbf{a} and \mathbf{b} :

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

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

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

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

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

They're the same!

And this is only $O(m)$ to compute!

QP with Quadratic basis functions

$$\text{Maximize } \sum_{k=1}^R \alpha_k + \sum_{k=1}^R \sum_{l=1}^R \alpha_k \alpha_l Q_{kl} \quad \text{where } Q_{kl} = y_k y_l (\Phi(\mathbf{x}_k) \cdot \Phi(\mathbf{x}_l))$$

Subject to these constraints:

$$0 \leq \alpha_k \leq 1$$

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

Each dot product now only requires m additions and multiplications

Then define:

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

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

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

Then classify with:

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

Higher Order Polynomials

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

QP with Quintic basis functions

We must do $R^2/2$ 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?

constraints.

$$Q_{kl} = y_k y_l (\Phi(\mathbf{x}_k) \cdot \Phi(\mathbf{x}_l))$$

$$\forall k \quad \sum_{k=1}^R \alpha_k y_k = 0$$

Then define:

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

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

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

Then classify with:

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

QP with Quintic basis functions

We must do $R^2/2$ 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?

$$Q_{kl} = y_k y_l (\Phi(\mathbf{x}_k) \cdot \Phi(\mathbf{x}_l))$$

$$\forall k \quad \sum_{k=1}^R \alpha_k y_k = 0$$

constraints.

- 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?*)

Then define:

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

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

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

Then classify with:

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

QP with Quintic basis functions

We must do $R^2/2$ 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?

constraints.

$$Q_{ll} = v_l v_l (\Phi(\mathbf{x}_l) \cdot \Phi(\mathbf{x}_l))$$

The use of Maximum Margin magically makes this not a problem

$$\forall k, \alpha_k y_k = 0$$

- 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?)

Then define:

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

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

where $K = \arg \max_k \alpha_k$

Because each $\mathbf{w} \cdot \phi(\mathbf{x})$ (see below) needs 75 million operations. *What can be done?*

Then classify with:

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

QP with Quintic basis functions

We must do $R^2/2$ 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?

$$Q_{ll} = y_l y_l (\Phi(\mathbf{x}_l) \cdot \Phi(\mathbf{x}_l))$$

The use of Maximum Margin magically makes this not a problem

$$\forall k, \alpha_k y_k = 0$$

constraints.

- The fear of overfitting with this enormous number of terms

Then define:

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

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

Because each $\mathbf{w} \cdot \phi(\mathbf{x})$ (see below) needs 75 million operations. What can be done?

$$\begin{aligned} \mathbf{w} \cdot \Phi(\mathbf{x}) &= \sum_{k \text{ s.t. } \alpha_k > 0} \alpha_k y_k \Phi(\mathbf{x}_k) \cdot \Phi(\mathbf{x}) \\ &= \sum_{k \text{ s.t. } \alpha_k > 0} \alpha_k y_k (\mathbf{x}_k \cdot \mathbf{x} + 1)^5 \end{aligned}$$

Then classify with:

Only Sm operations ($S = \#$ support vectors)

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

QP with Quintic basis functions

We must do $R^2/2$ 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?

$$Q_{ll} = y_l y_l (\Phi(\mathbf{x}_l) \cdot \Phi(\mathbf{x}_l))$$

The use of Maximum Margin magically makes this not a problem

$$\forall k, \alpha_k y_k = 0$$

constraints.

- The fear of overfitting with this enormous number of terms

Then define:

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

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

Because each $\mathbf{w} \cdot \phi(\mathbf{x})$ (see below) needs 75 million operations. What can be done?

$$\begin{aligned} \mathbf{w} \cdot \Phi(\mathbf{x}) &= \sum_{k \text{ s.t. } \alpha_k > 0} \alpha_k y_k \Phi(\mathbf{x}_k) \cdot \Phi(\mathbf{x}) \\ &= \sum_{k \text{ s.t. } \alpha_k > 0} \alpha_k y_k (\mathbf{x}_k \cdot \mathbf{x} + 1)^5 \end{aligned}$$

Then classify with:

Only Sm operations ($S = \#$ support vectors)

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

QP with Quintic basis functions

$$\text{Maximize } \sum_{k=1}^R \alpha_k + \sum_{k=1}^R \sum_{l=1}^R \alpha_k \alpha_l Q_{kl} \quad \text{wh}$$

$$\text{Subject to these constraints: } 0 \leq \alpha_k \leq C$$

Then define:

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

$$\begin{aligned} \mathbf{w} \cdot \Phi(\mathbf{x}) &= \sum_{k \text{ s.t. } \alpha_k > 0} \alpha_k y_k \Phi(\mathbf{x}_k) \cdot \Phi(\mathbf{x}) \\ &= \sum_{k \text{ s.t. } \alpha_k > 0} \alpha_k y_k (\mathbf{x}_k \cdot \mathbf{x} + 1)^5 \end{aligned}$$

Only S m operations ($S = \#$ support vectors)

Andrew's opinion of why SVMs don't overfit as much as you'd think:

No matter what the basis function, there are really only up to R parameters: $\alpha_1, \alpha_2 \dots \alpha_R$, and usually most are set to zero by the Maximum Margin.

Asking for small $\mathbf{w} \cdot \mathbf{w}$ is like "weight decay" in Neural Nets and like Ridge Regression parameters in Linear regression and like the use of Priors in Bayesian Regression---all designed to smooth the function and reduce overfitting.

Then classify with:

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

SVM Kernel Functions

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

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

σ , κ and δ are magic parameters that must be chosen by a model selection method

- Neural-net-style Kernel Function:

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

Doing multi-class classification

- SVMs can only handle two-class outputs (i.e. a categorical output variable with arity 2).
- What can be done?
- Answer: with output arity N , learn N SVM's
 - SVM 1 learns "Output==1" vs "Output != 1"
 - SVM 2 learns "Output==2" vs "Output != 2"
 - :
 - SVM N learns "Output== N " vs "Output != N "
- Then to predict the output for a new input, just predict with each SVM and find out which one puts the prediction the furthest into the positive region.

What You Should Know

- Linear SVMs
- The definition of a maximum margin classifier
- What QP can do for you (but, for this class, you don't need to know how it does it)
- How Maximum Margin can be turned into a QP problem
- How we deal with noisy (non-separable) data
- How we permit non-linear boundaries
- How SVM Kernel functions permit us to pretend we're working with ultra-high-dimensional basis-function terms