Support Vector Machines and Kernels

CMU machine learning group members

CSC411/2515: Machine Learning and Data Mining, Winter 2018

Michael Guerzhoy and Lisa Zhang

Supervised Learning

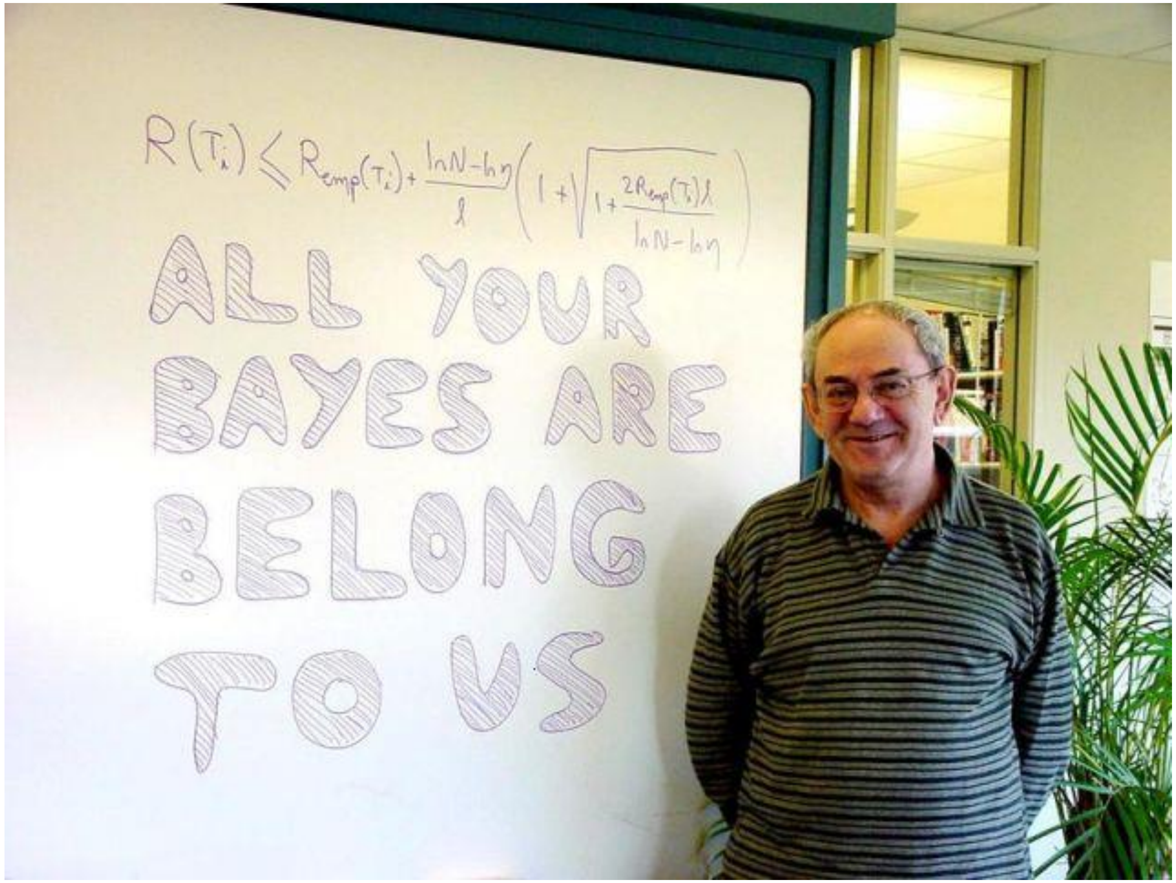
- Want to figure out the parameter of h_θ such that $h_\theta(x) = y$ for unseen data (x, y)
- Overfitting: finding a θ that corresponds to the peculiarities of the training set rather than to genuine patterns in the data
    - Definition: there exists a θ' such that
      $$TrainPerformance(\theta') < TrainPerformance(\theta)$$
      $$Performance(\theta') > Performance(\theta)$$
    - $Performance(\theta')$ is the performance of h_θ on *all possible data* (of which the training set is just a small subset)
    - Can't overfit on a training set that contains all the possible data!

Strategy so far to avoid overfitting

- Constrain the θ
  - Regularization, early stopping
- Constrain the h_θ
  - Make the neural network small
  - Don't compute too many features of the input x

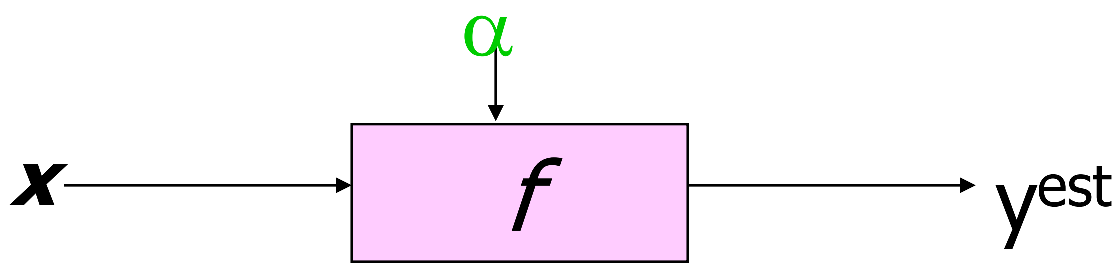
Support Vector Machines (SVMs)

- Idea: select an h_θ that separates the data with the *largest margin*
  - Hope: this will make h_θ more generalizable
- Idea: apply the *kernel trick* to be able to compute lots of features of the data, and apply SVMs to avoid overfitting
- History
  - SVMs: Vladimir Vapnik and Alexey Chervoninkis in 1963
  - Kernels and SVMs: Bernhard Boser, Isabelle Guyon, and Vladimir Vapnik in 1992
  - Soft margins: Corinna Cortes and Vladimir Vapnik, 1993
- Applied in practice to classify text, images, etc.



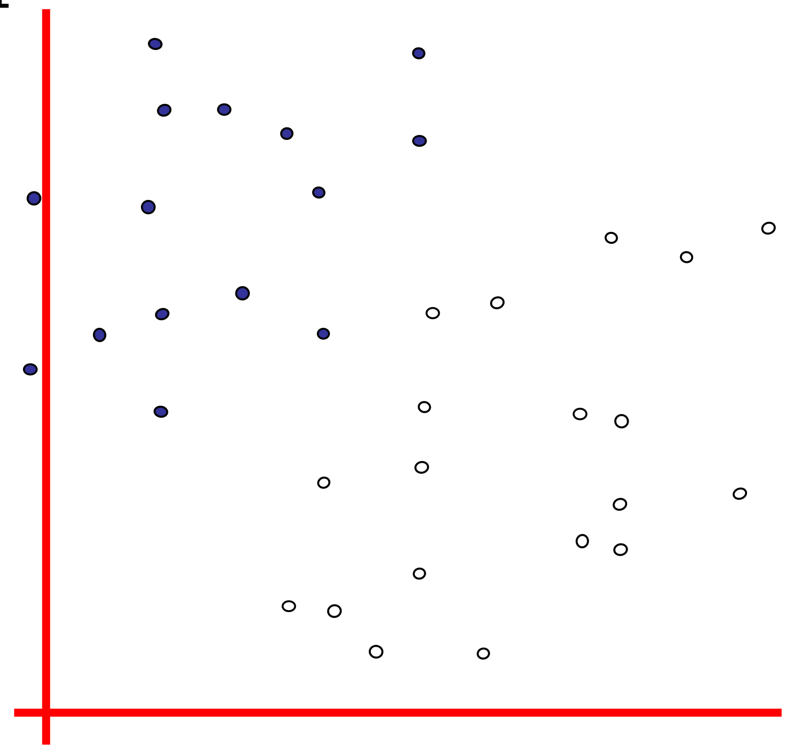
Vladimir Vapnik and his theoretical upper bound on the test error (not covered ☹)

Linear Classifiers



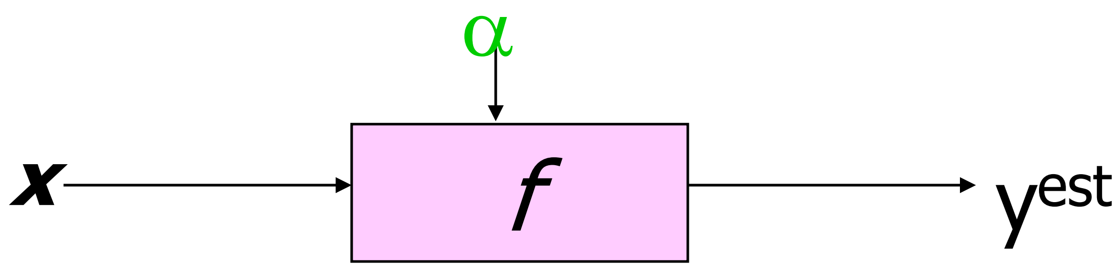
$$f(x, w, b) = sign(w^T x + b)$$

- ● denotes +1
- ○ denotes -1



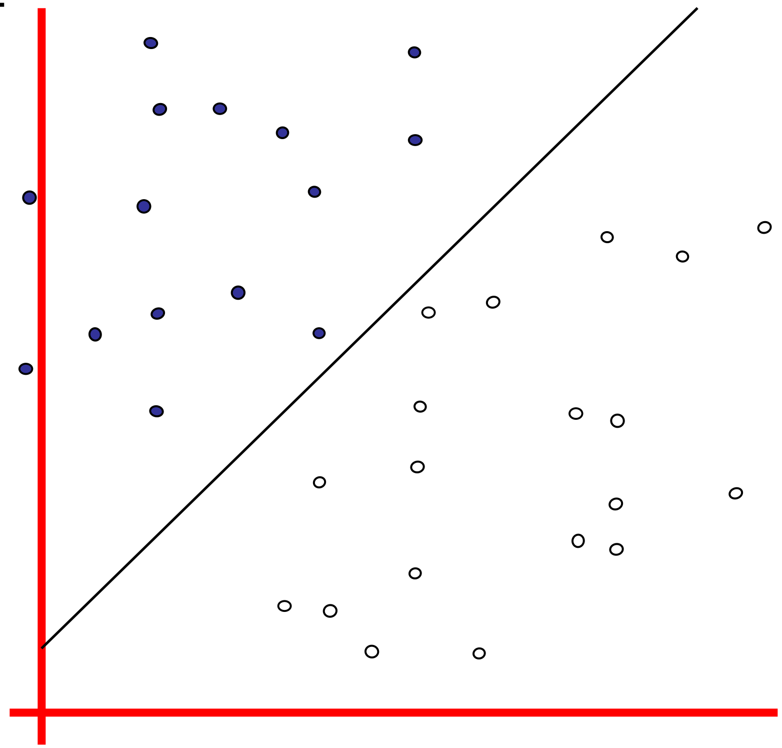
How would you classify this data?

Linear Classifiers



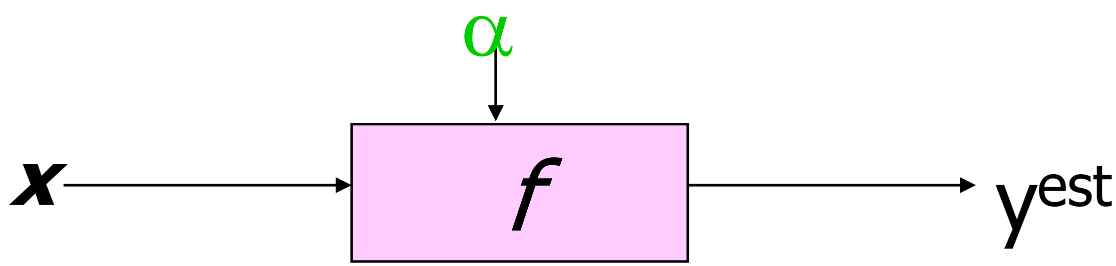
$$f(x, w, b) = sign(w^T x + b)$$

- denotes +1
- denotes -1



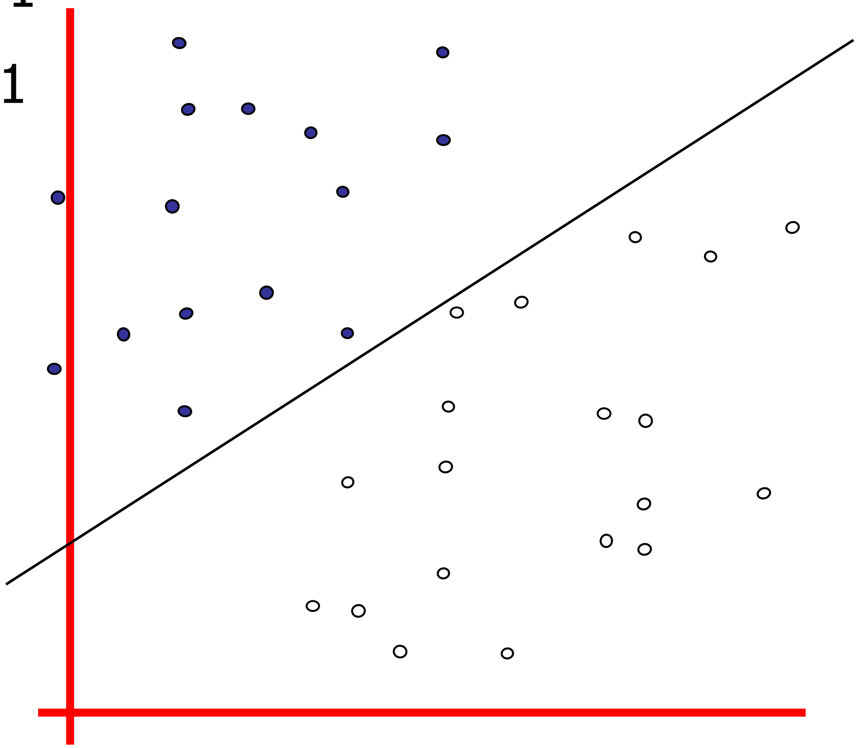
How would you classify this data?

Linear Classifiers



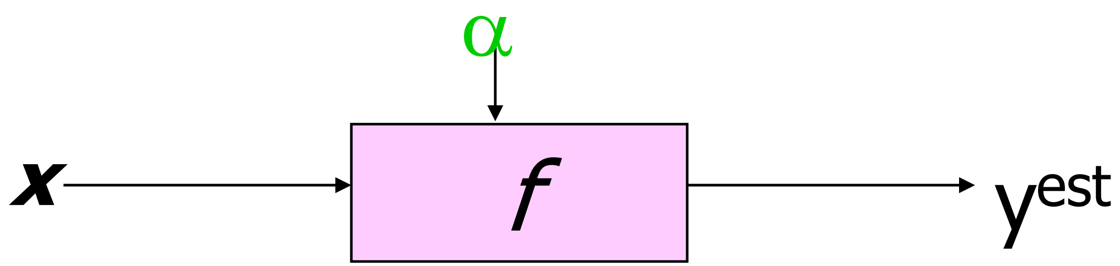
$$f(x, w, b) = sign(w^T x + b)$$

- denotes +1
- denotes -1



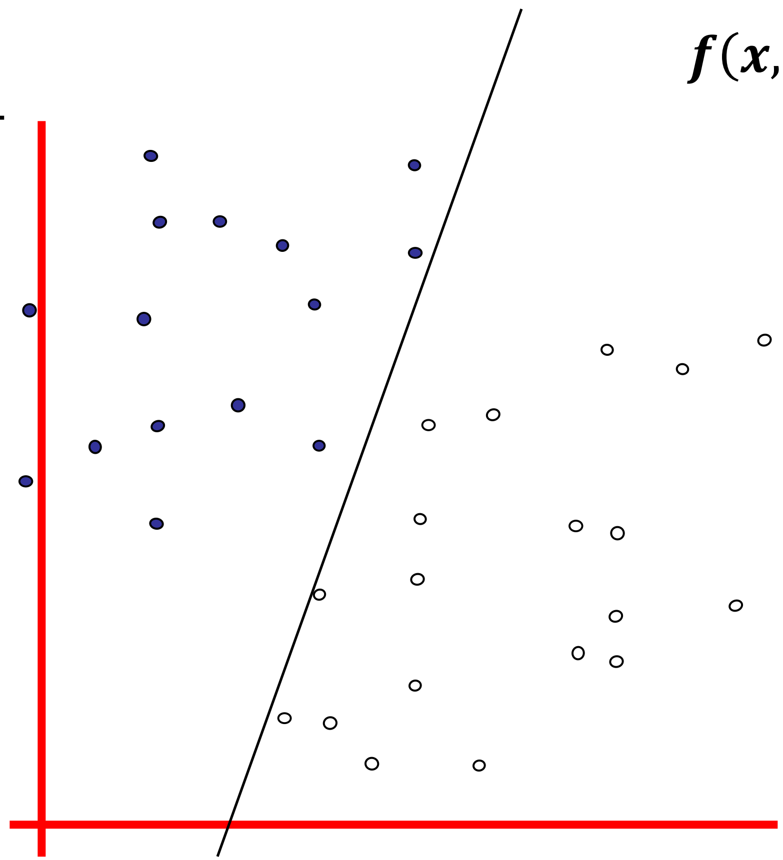
How would you classify this data?

Linear Classifiers



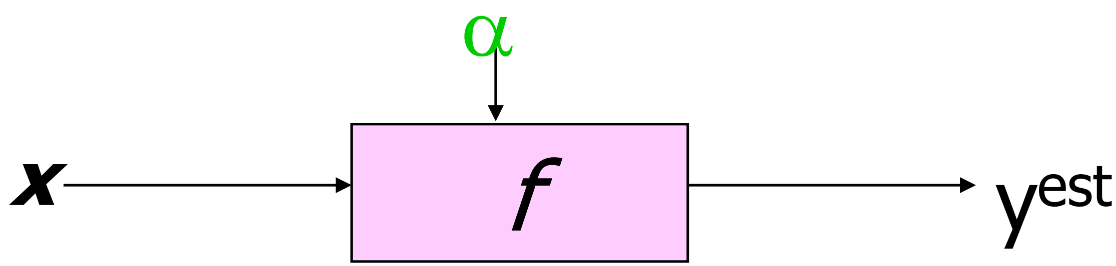
$$f(x, w, b) = sign(w^T x + b)$$

- denotes +1
- ○ denotes -1



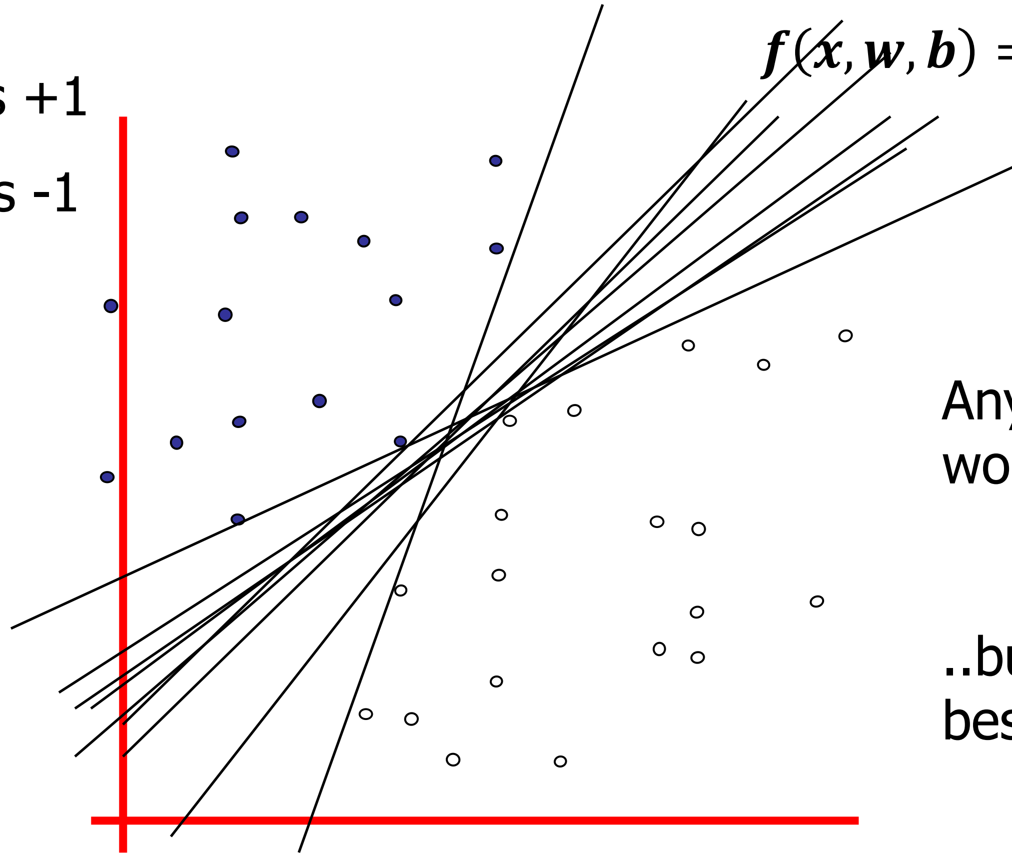
How would you classify this data?

Linear Classifiers



$$f(x, w, b) = sign(w^T x + b)$$

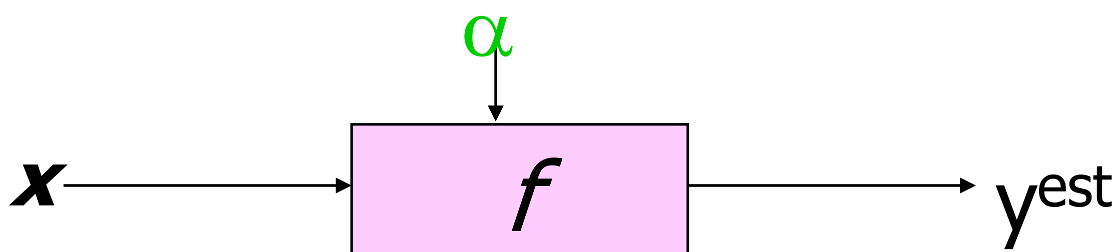
- denotes +1
- denotes -1



Any of these would be fine..

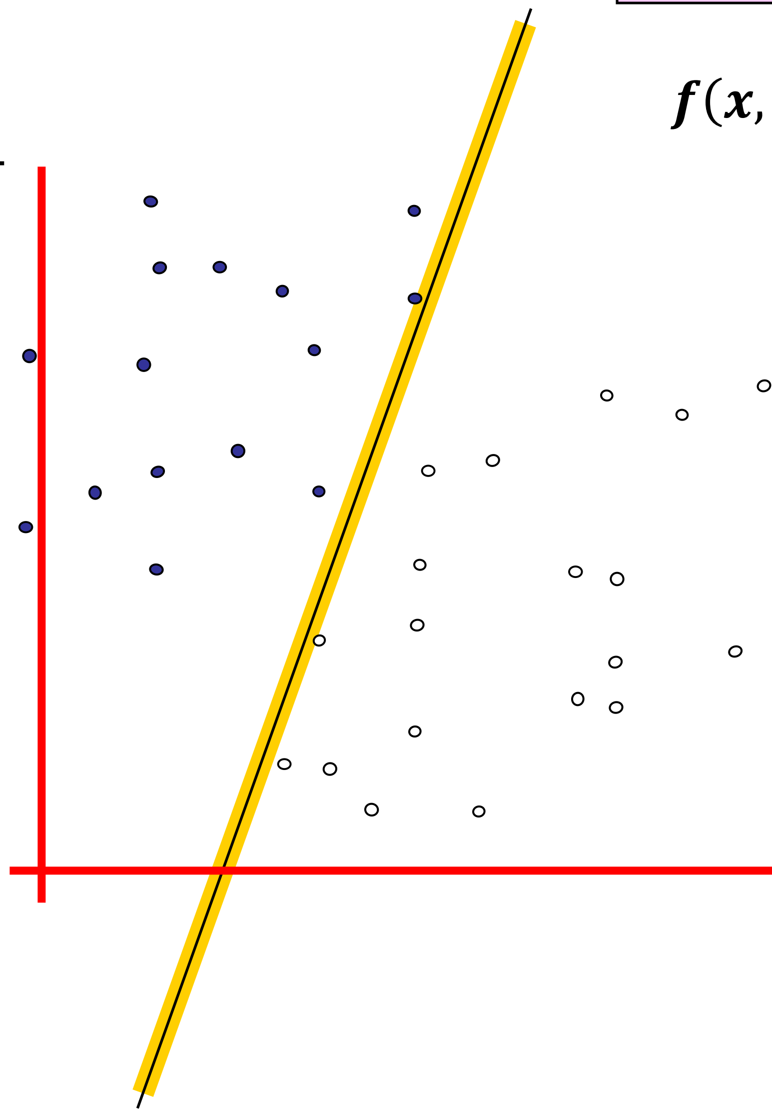
..but which is best?

Classifier Margin



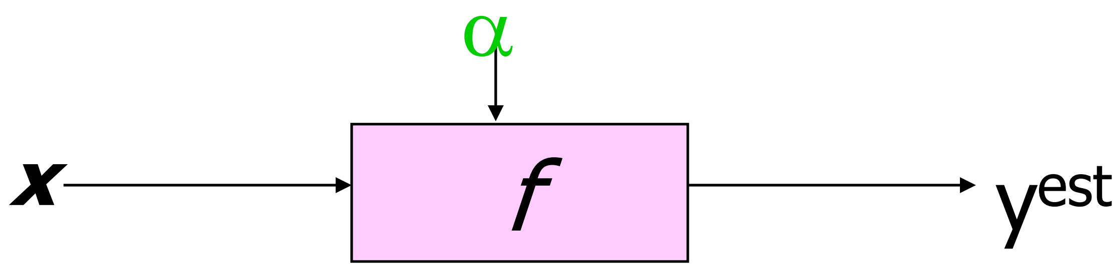
$$f(x, w, b) = sign(w^T x + b)$$

- denotes +1
- denotes -1

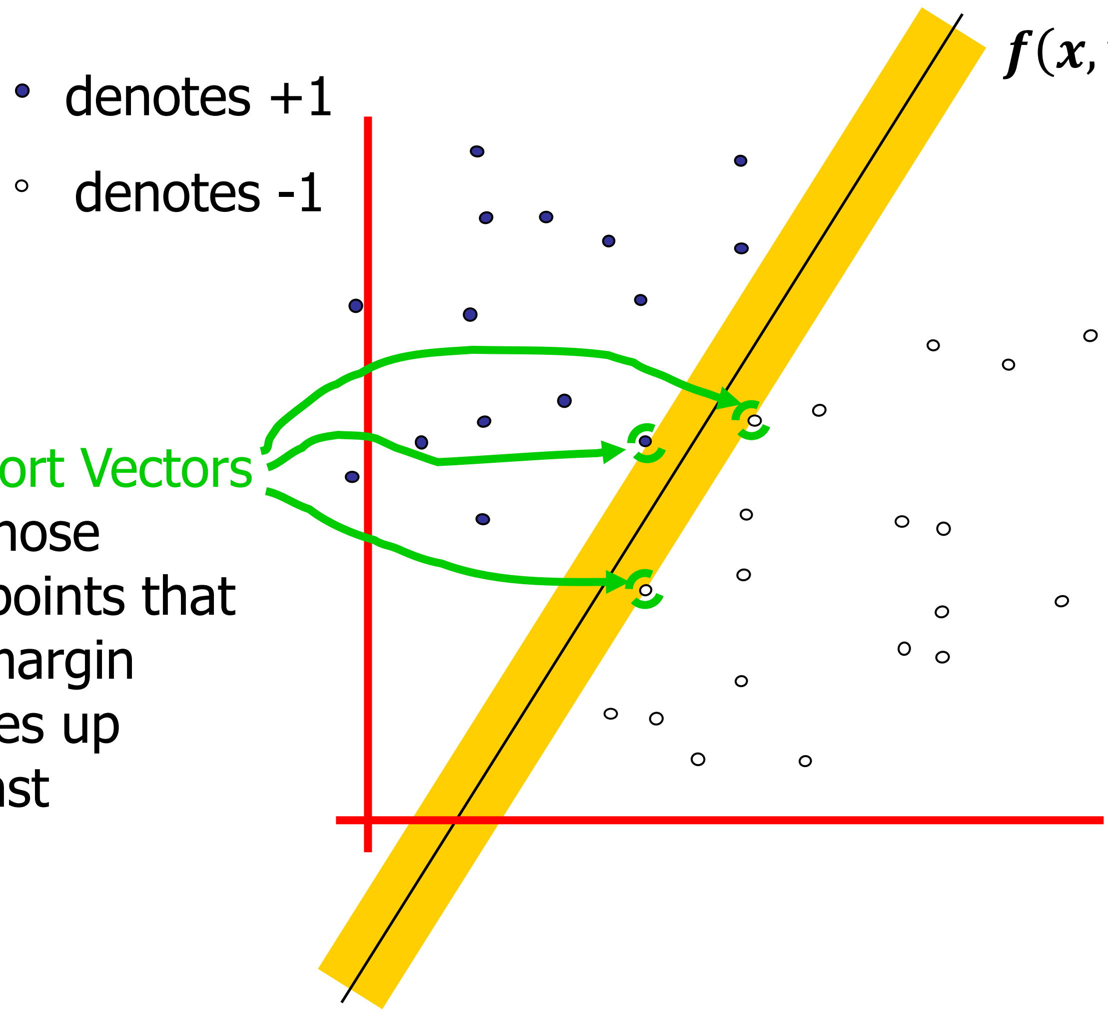


Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

Maximum Margin



$$f(x, w, b) = sign(x^T x + b)$$



Support Vectors are those datapoints that the margin pushes up against

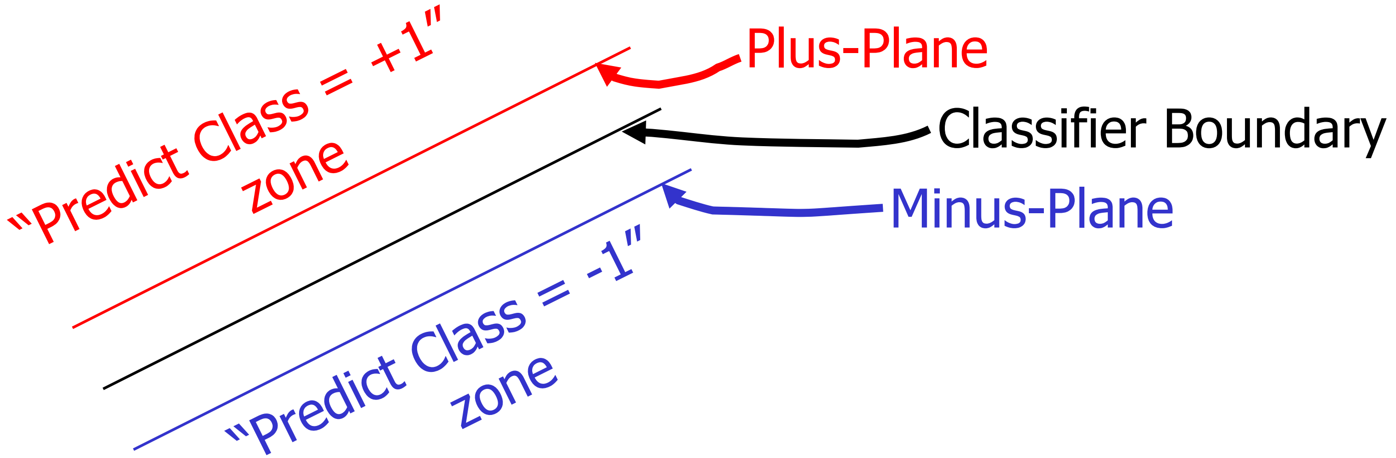
The maximum margin linear classifier is the linear classifier with the maximum margin.

Linear SVMs (LSVMs) find the decision boundary with the largest margin

Why Maximum Margin?

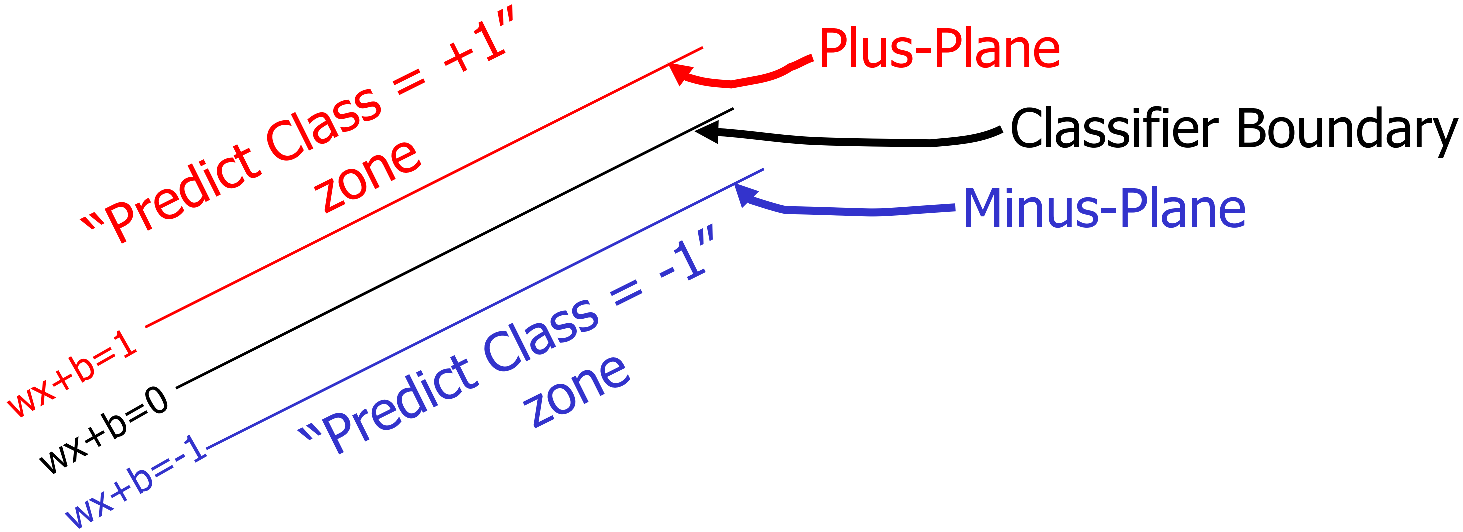
- Intuitively this feels safest.
- If we’ve made a small error in the location of the boundary (it’s been jolted in its perpendicular direction) this gives us least chance of causing a misclassification.
- Empirically it works very very well.
- It can be mathematically shown that in some settings this will have the best validation error

Specifying a line and margin



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

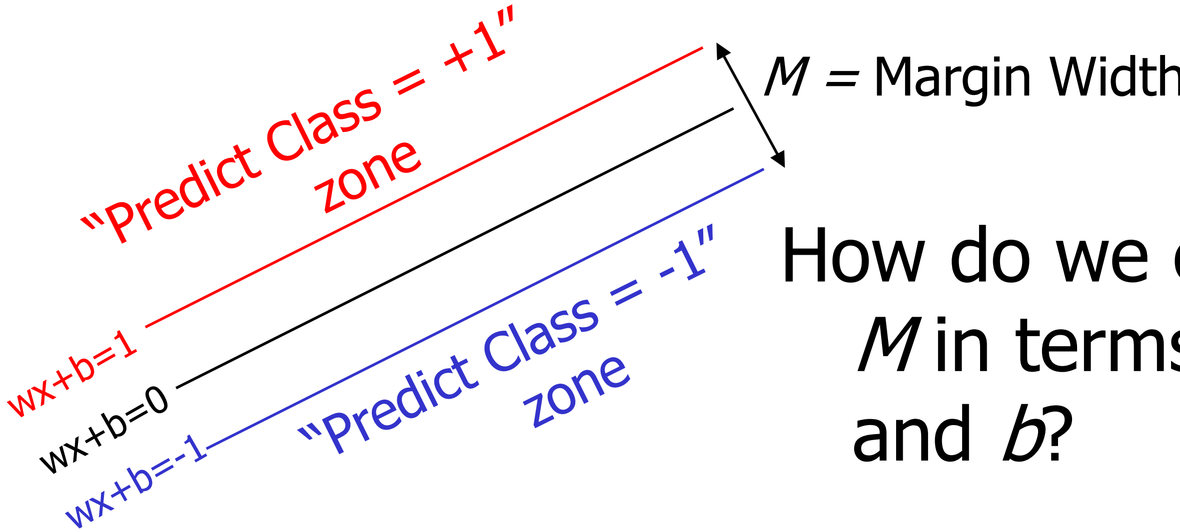
Specifying a line and margin



- Plus-plane = $\{x: w^T x + b = +1\}$
- Minus-plane = $\{x: w^T x + b = -1\}$

Classify as..	+1	if	$w^T x + b \geq 1$
	-1	if	$w^T x + b \leq -1$
	Universe explodes	if	$-1 < w^T x + b < 1$

Computing the margin width



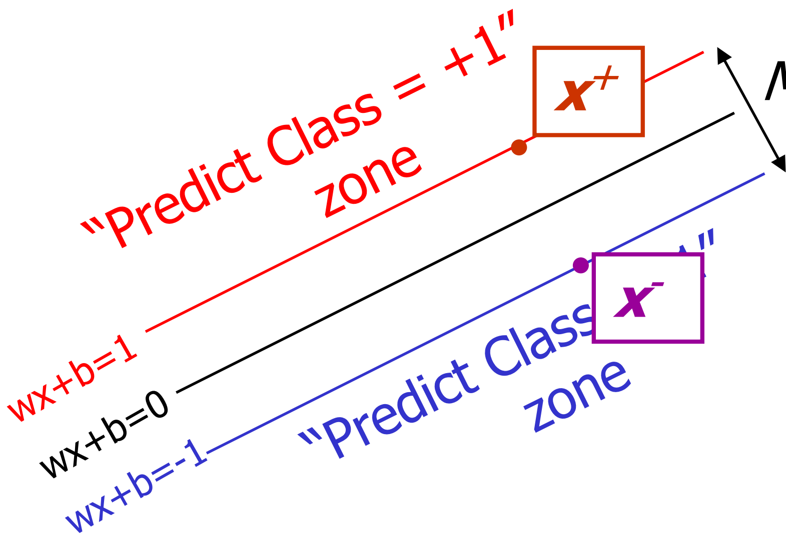
How do we compute M in terms of $\boldsymbol{w}$ and b?

- Plus-plane = $\{x\colon w^T x + b = 1\}$
- Minus-plane = $\{x\colon w^T x + b = -1\}$

The vector **w** is perpendicular to the Plus-plane

- The vector w is in general perpendicular to the plane $w^T x = 0$
  - $\{x\colon w^T x = 0\}$ is the set of vectors such that are perpendicular to w.
  - $w^T x = c$ is just $w^T x = 0$, shifted

Computing the margin width

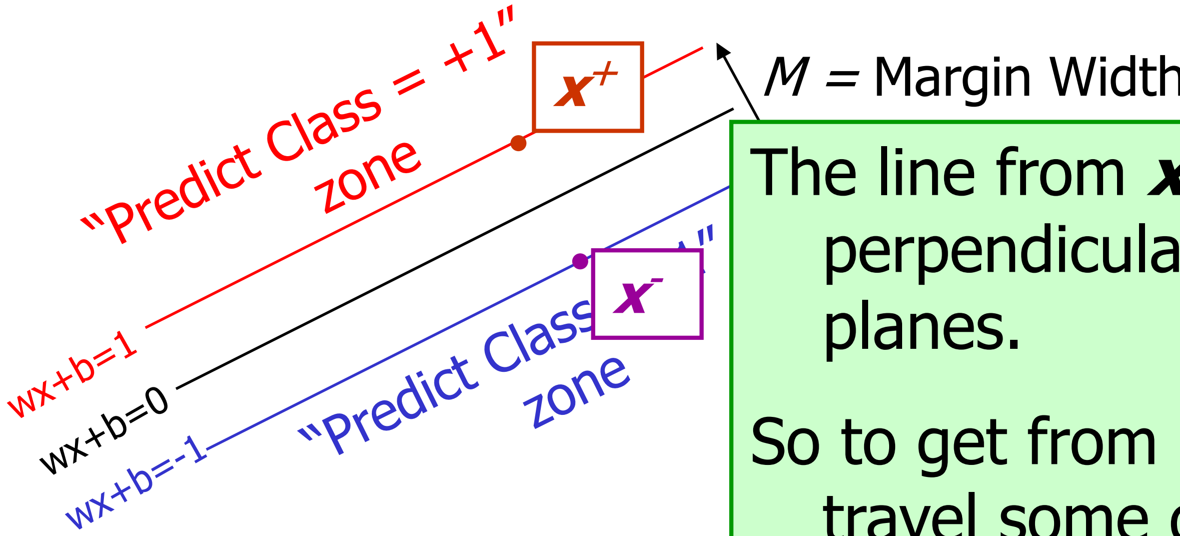


M = Margin Width

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

- Plus-plane = $\{x\colon w^T x + b = 1\}$
- Minus-plane = $\{x\colon w^T x + b = -1\}$
- 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

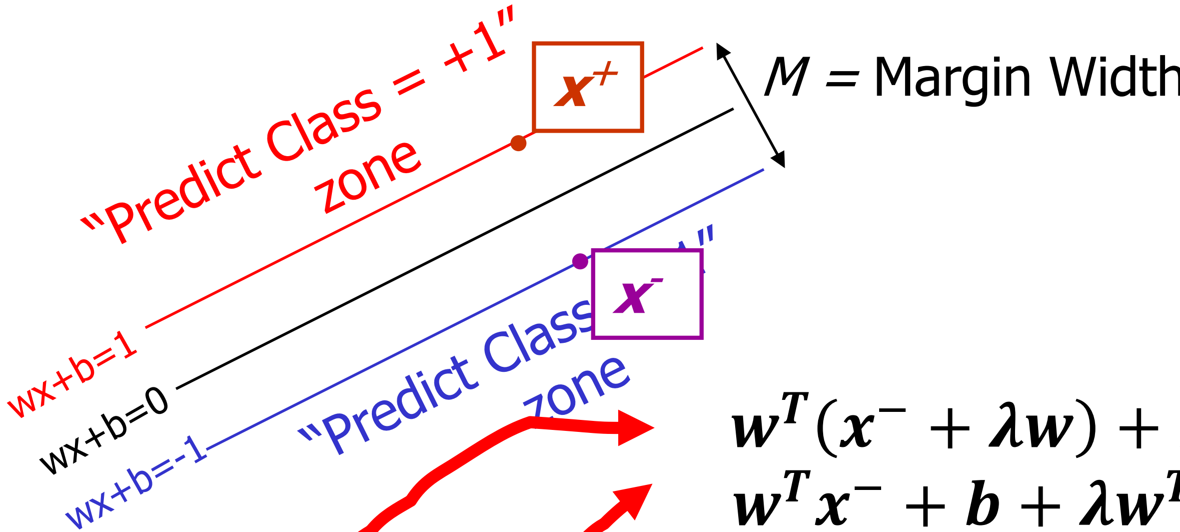


The line from x^- to x^+ is perpendicular to the planes.

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

- Plus-plane = $\{x: w^T x + b = 1\}$
- Minus-plane = $\{x: w^T x + b = -1\}$
- 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



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

What we know:

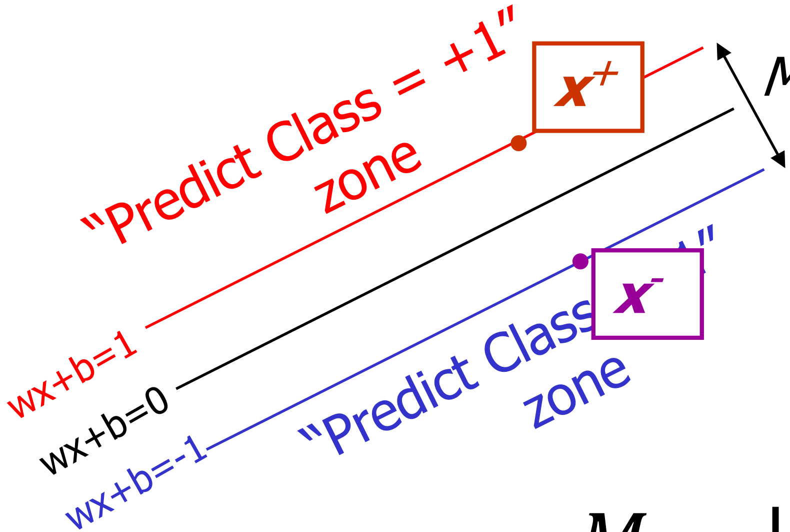
- $\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}^+ + \boldsymbol{b} = \boldsymbol{1}$
- $\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}^- + \boldsymbol{b} = -\boldsymbol{1}$
- $\boldsymbol{x}^+ = \boldsymbol{x}^- + \boldsymbol{\lambda w}$
- $|x^+ - x^-| = M$

So

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

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

Computing the margin width



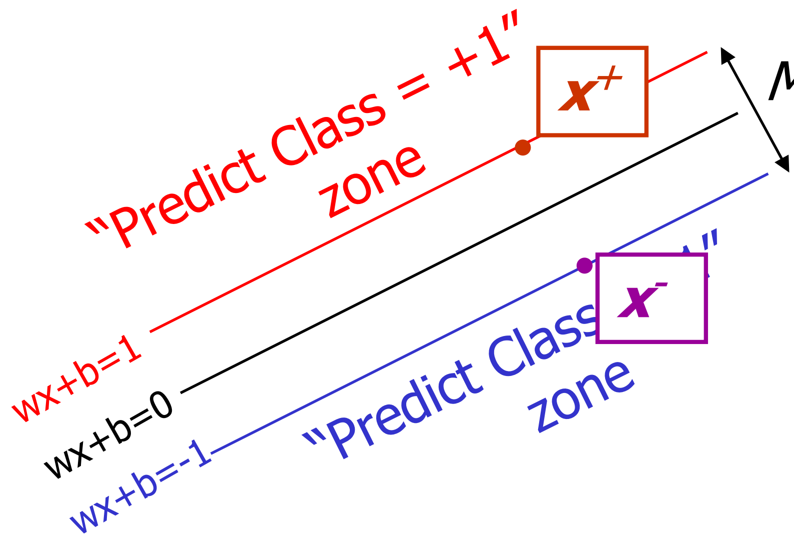
M = Margin Width = $\frac{2}{\sqrt{w^T w}}$

What we know:

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

$$M = |x^+ - x^-| = |\lambda w| = \lambda |w| = \lambda \sqrt{w^T w} = \frac{2\sqrt{w^T w}}{w^T w} = \frac{2}{\sqrt{w^T w}}$$

Learning the Maximum Margin Classifier



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

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

- Compute whether all data points are in the correct half-planes
  - Compute $sign(w^T x + b)$
- Compute the width of the margin
- We now want to find the w and b that produce the maximum margin

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

Subject to

additional linear inequality constraints

And subj…

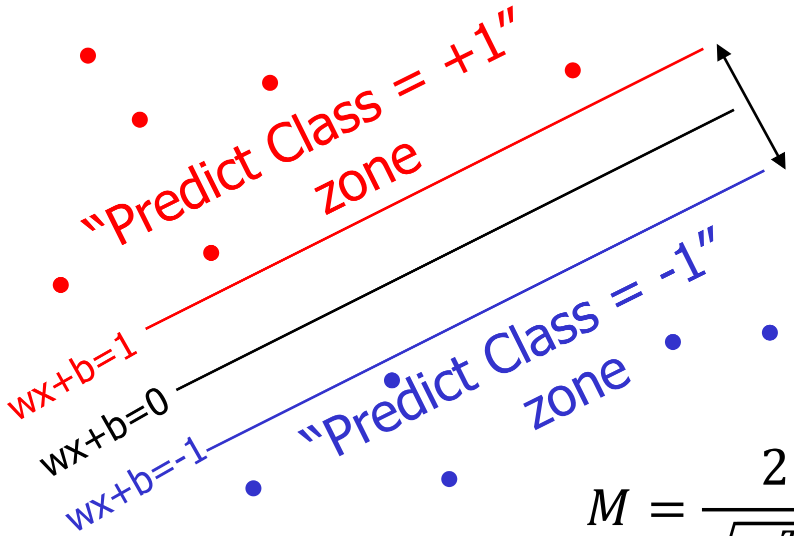
There exist algorithms for finding such constrained quadratic optima much more efficiently and reliably than gradient ascent.

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

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

e additional linear equality constraints

Learning the Maximum Margin Classifier



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

Given a 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 $\left(x^{(k)}, y^{(k)}\right)$ where $y^{(k)}$ = +/- 1

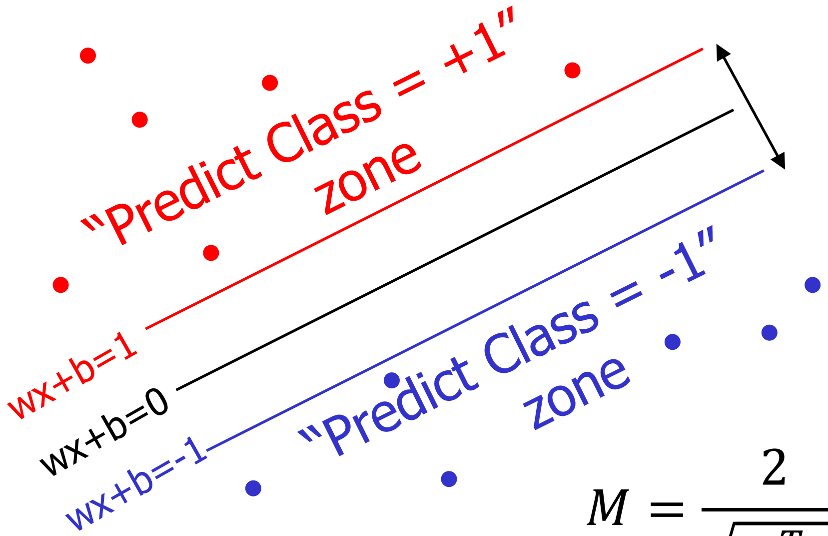
What should our quadratic optimization criterion be?

Minimize $w^T w$

How many constraints will we have?

What should they be?

Learning the Maximum Margin Classifier



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

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

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

Assume R datapoints, each $\left(x^{(k)}, y^{(k)}\right)$ where $y^{(k)}$= +/- 1

What should our quadratic optimization criterion be?

Minimize $w^T w$

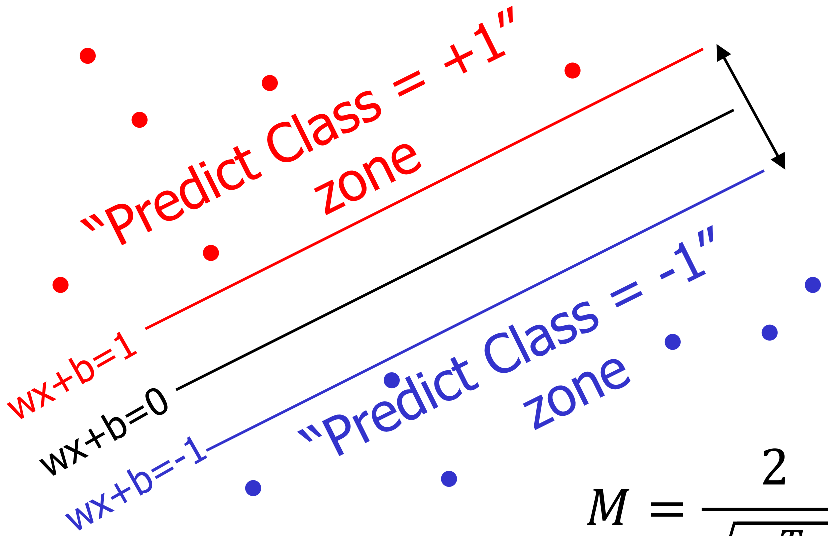
How many constraints will we have? *R*

What should they be?

$$w^T x^{(k)} + b \geq 1 \text{ if } y^{(k)} = 1$$

$$w^T x^{(k)} + b \leq -1 \text{ if } y^{(k)} = -1$$

Support Vectors



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

- x's s.t. $w^T x^{(k)} + b = 1$ lie on the line $wx + b = 1$
- x's s.t. $w^T x^{(k)} + b = -1$ lie on the line $wx + b = -1$
- Those x's define the w
  - Can ignore all other data, and the maximum-margin classifier will be the same

What should our quadratic optimization criterion be?

Minimize $w^T w$

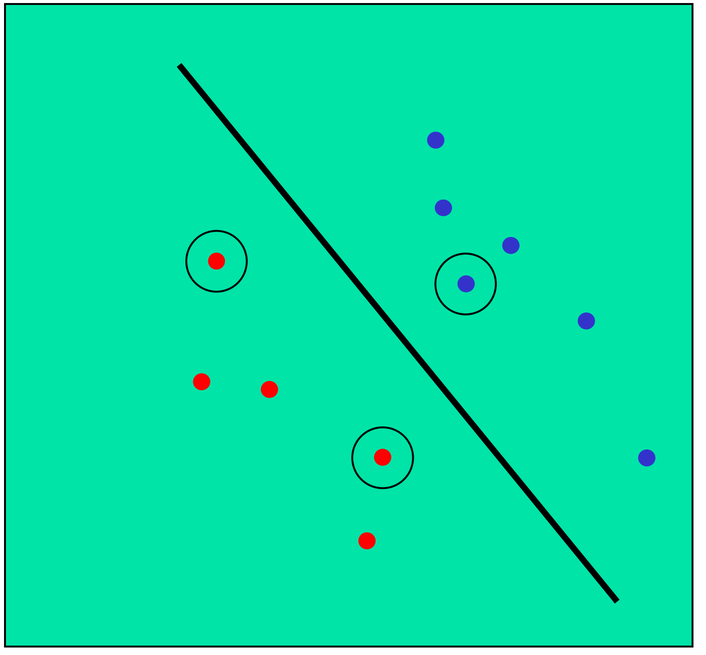
How many constraints will we have? *R*

What should they be?

$w^T x^{(k)} + b \geq 1$ if $y^{(k)} = 1$

$w^T x^{(k)} + b \leq -1$ if $y^{(k)} = -1$

Support Vectors

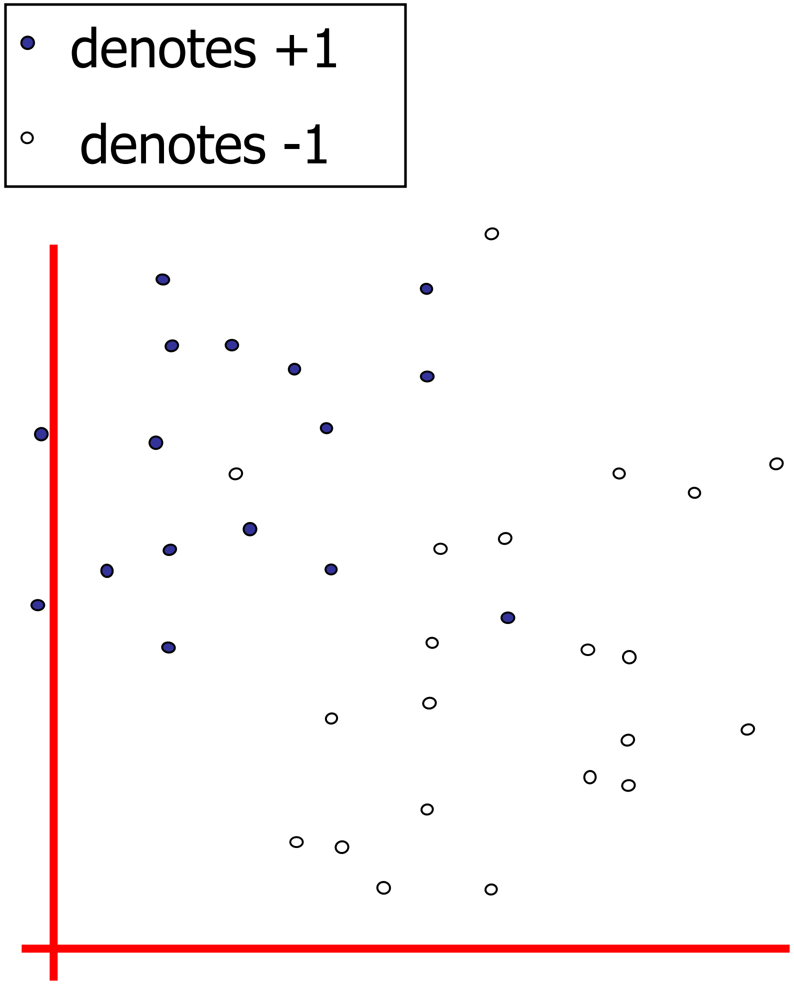


- Support vectors are the vectors that determine the separating hyperplane
- They lie on $w^T x + b = 1$ and $w^T x + b = -1$
- Other vectors in the training set don't matter for the decision boundary

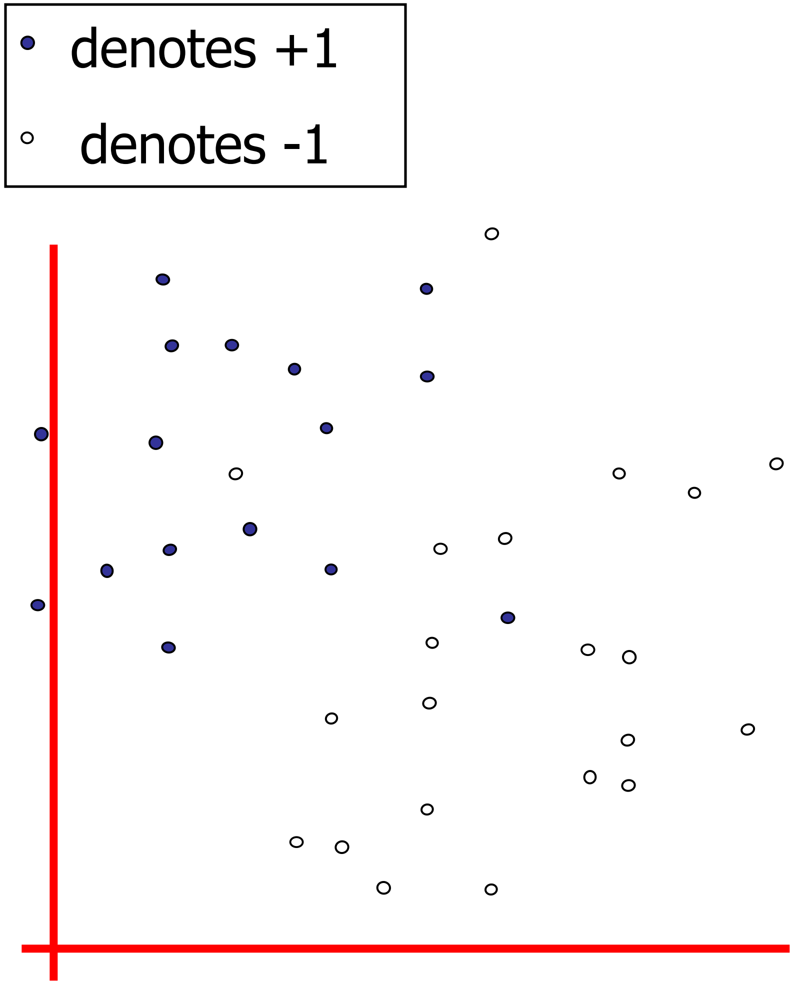
Uh-oh!

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 $w^T w$, 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!

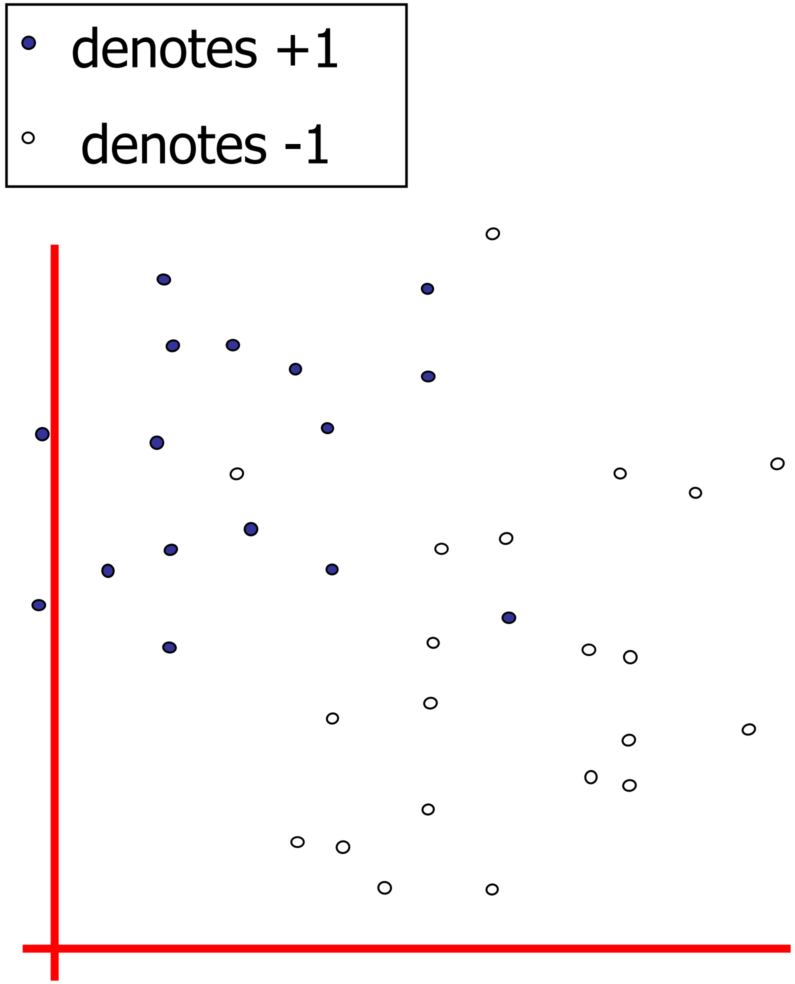
What should we do?

Idea 1.1:

Minimize

$w^T w + C$ (#train errors)

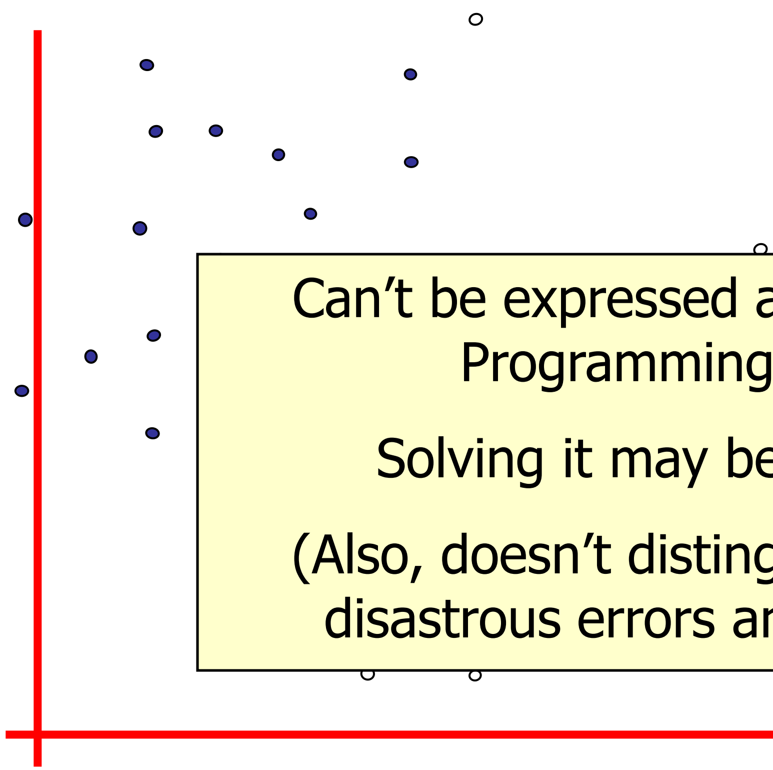
Tradeoff parameter



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

Uh-oh!

- denotes +1
- denotes -1



This is going to be a problem!

What should we do?

Idea 1.1:

Minimize

$$w^T w \quad + C \; (\#\text{train errors})$$

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)

erious

at'

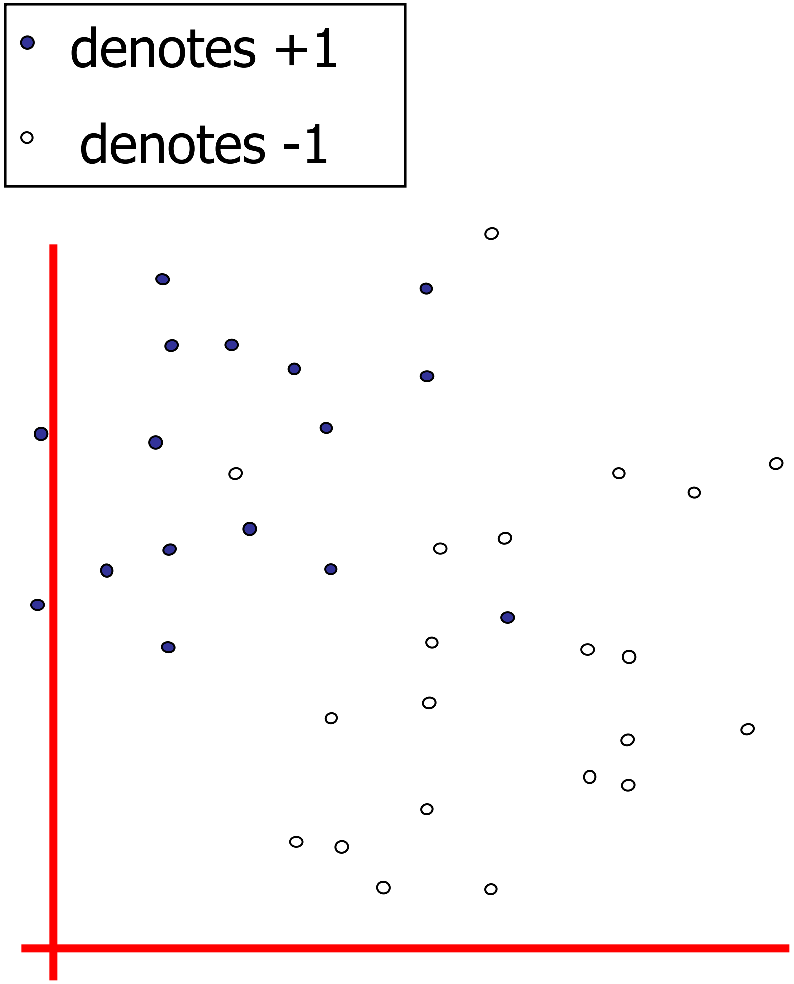
ke

us reject thi

you guess wh

So… any other ideas?

Uh-oh!



This is going to be a problem!

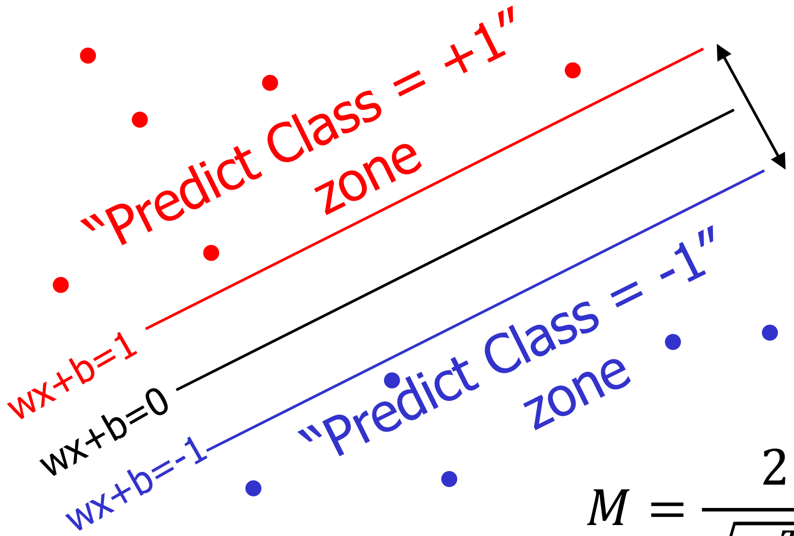
What should we do?

Idea 2.0:

Minimize

$w^T w$ + *C (distance of error points to their correct place)*

Learning the Maximum Margin with Noise



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

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

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

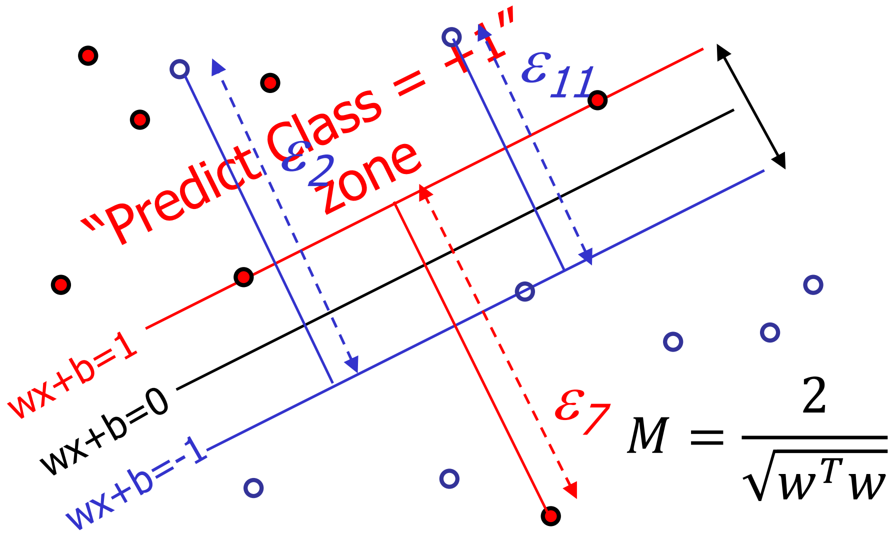
Assume *R* datapoints, each $\left(x^{(k)}, y^{(k)}\right)$ where $y^{(k)}$= +/- 1

What should our quadratic optimization criterion be?

How many constraints will we have?

What should they be?

Learning the Maximum Margin with Noise



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 $\left(x^{(k)}, y^{(k)}\right)$ where $y^{(k)}$= +/- 1

What should our quadratic optimization criterion be?

Minimize $\frac{1}{2} w^T w + C \sum_{k=1}^{R} \epsilon^{(k)}$

The $\epsilon^{(k)}$'s are called "slack variables"

The technique is called "soft margin"

How many constraints will we have? *R*

What should they be?

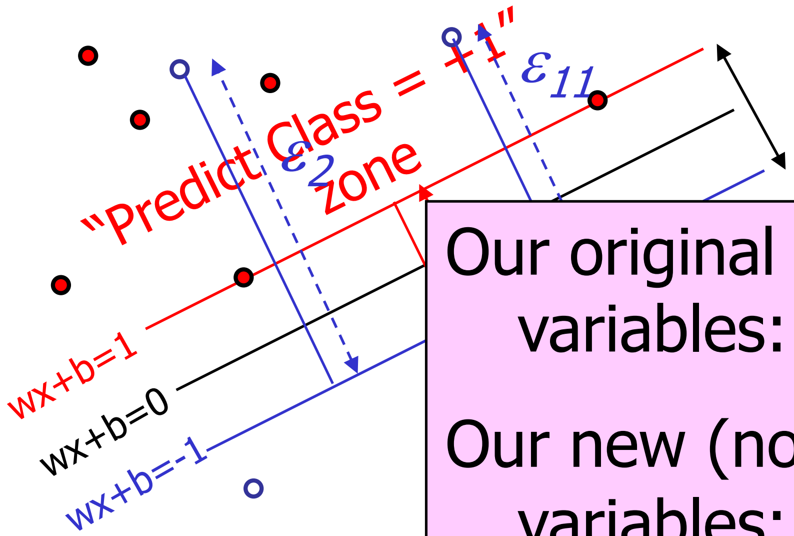
$$\boldsymbol{w^T x^{(k)} + b \geq 1 - \epsilon^{(k)}} \ \textbf{\textit{if}} \ \boldsymbol{y^{(k)} = 1}$$

$$\boldsymbol{w^T x^{(k)} + b \leq -1 + \epsilon^{(k)}} \ \textbf{\textit{if}} \ \boldsymbol{y^{(k)} = -1}$$

$$\boldsymbol{\epsilon^{(k)} \geq 0}, \textbf{\textit{for all}}\ \boldsymbol{k}$$

Learning the Maximum Margi e

m = # input dimensions



Given gu n

- Compute whethe l data points are in the correct ha f-planes

$, y^{(k)})$

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*, $,\epsilon^{(1)},, \ldots \epsilon^{(R)}$

R= # records

What should our quadratic optimization criterion be?

Minimize $\frac{1}{2} w^T w + C \sum_{k=1}^{R} \epsilon^{(k)}$

How many constrai will we have? *R*

What should they be?

$\boldsymbol{w^T x^{(k)} + b \geq 1 - \epsilon^{(k)}}$ ***if*** $\boldsymbol{y^{(k)} = 1}$

$\boldsymbol{w^T x^{(k)} + b \leq -1 + \epsilon^{(k)}}$ ***if*** $\boldsymbol{y^{(k)} = -1}$

$\boldsymbol{\epsilon^{(k)} \geq 0}$***, for all*** $\boldsymbol{k}$

Primal Formulation for SVM with slack variables

$$\min_{w,b} \frac{1}{2}\left||w|\right|^2 + C\sum_k \epsilon^{(k)}$$

s.t.:

$$w^T x^{(k)} + b \geq 1 - \epsilon^{(k)} \textit{ if } y^{(k)} = 1$$

$$w^T x^{(k)} + b \leq -1 + \epsilon^{(k)} \textit{ if } y^{(k)} = -1$$

$$\epsilon^{(k)} \geq 0, \textit{ for all } k$$

Equivalently:

$$\left(w^T x^{(k)} + b\right) y^{(k)} + \epsilon^{(k)} \geq 1$$

- Can solve using Quadratic Programming algorithms
- Know as the “primal formulation”

Solving the Primal Formulation with Gradient Descent

$$\mathbf{min}_{w,b} \frac{1}{2}\left|\left|\boldsymbol{w}\right|\right|^2 + \boldsymbol{C}\sum_{\boldsymbol{k}} \boldsymbol{\epsilon}^{(\boldsymbol{k})}$$

s.t.:

$$\boldsymbol{w}^T\boldsymbol{x}^{(k)} + \boldsymbol{b} \geq \mathbf{1} - \boldsymbol{\epsilon}^{(k)} \;\; \textbf{\textit{if}} \;\; \boldsymbol{y}^{(k)} = \mathbf{1}$$

$$\boldsymbol{w}^T\boldsymbol{x}^{(k)} + \boldsymbol{b} \leq -\mathbf{1} + \boldsymbol{\epsilon}^{(k)} \;\; \textbf{\textit{if}} \;\; \boldsymbol{y}^{(k)} = -\mathbf{1}$$

$$\epsilon^{(k)} \geq \mathbf{0}, \;\; \textbf{\textit{for all}} \; k$$

Equivalently:

$$\left(w^T x^{(k)} + b\right) y^{(k)} + \epsilon^{(k)} \geq 1$$

- Convert this to

$$\min_{w,b} C \sum_k \max\left(0, 1 - y^{(k)}\left(w^T x^{(k)} + b\right)\right) + \frac{1}{2}\left|\left|w\right|\right|^2$$

0 if $w^T x^{(k)} + b > 1 \; or < -1$ with the right $y^{(k)}$

$\epsilon^{(k)}$ otherwise

Solving the Primal Formulation with Gradient Descent

- Solve with gradient descent:

$$\min_{w,b} C \sum_k \max(0, 1 - y^{(k)}(w^T x^{(k)} + b)) + \frac{1}{2}||w||^2$$

  - max(0, t) is not differentiable at 0, but there are ways to deal with this

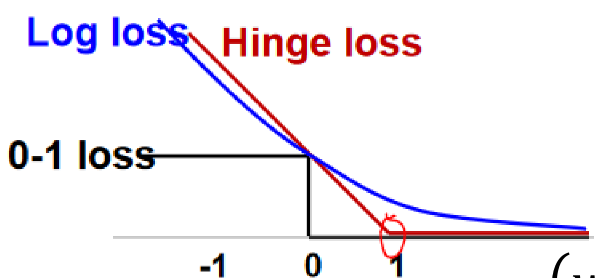
- Compare to regularized logistic regression:

$$\min_{w,b} C \sum_k (y^{(k)} \log \sigma(w^T x^{(k)} + b) + (1 - y^{(k)})(\log(1 - \sigma(w^T x^{(k)} + b)) + \lambda||w||^2$$

$$\leftrightarrow$$

$$\min_{w,b} C \sum_k \log(1 + \exp(-(w^T x^{(k)} + b) y^{(k)})) + \lambda||w||^2$$

- Hinge loss doesn't reward outputting more than 1
- Log loss rewards outputting more than 1 a little
- All the training examples matter for logistic regression, but not for linear SVM



$(w^T x^{(k)} + b) y^{(k)}$

Primal Formulation for SVM with slack variables (again)

$$\min_{w,b} \frac{1}{2}||w||^2 + C\sum_k \epsilon^{(k)}$$

s.t.:

$$w^T x^{(k)} + b \geq 1 - \epsilon^{(k)} \textit{ if } y^{(k)} = 1$$

$$w^T x^{(k)} + b \leq -1 + \epsilon^{(k)} \textit{ if } y^{(k)} = -1$$

$$\epsilon^{(k)} \geq 0\textit{, for all } k$$

Equivalently:

$$(w^T x^{(k)} + b)y^{(k)} + \epsilon^{(k)} \geq 1$$

- Can solve using Quadratic Programming algorithms
- Can solve with flavours of gradient descent
- Both would be slow for high dimensional x’s

Lagrange Multipliers

- Want to maximize $f(x)$ subject to $g_1(x) = 0, g_2(x) = 0, \ldots$
  - E.g.: $f(x) = x_1 x_2 x_3$, $g_1(x) = xy + xz + yz - 64$ $g_2(x) = x + y - 5$
- $L(x, \alpha) = f(x) - \alpha_1 g_1(x) - \ \alpha_2 g_2(x) - \cdots$
- The constrained minimum of $f(x)$ will be a local optimum of $L(x, \alpha)$

Dual Formulation for SVM (no slack variables)

- Want to minimize $\frac{1}{2}\left||w|\right|^2$ subject to $\left(w^T x^{(k)} + b\right) y^{(k)} \geq 1$
- Use Lagrange multipliers to convert this to
- $L_p = \frac{1}{2}\left||w|\right|^2 - \sum_{k=1}^{n} \alpha_k (y^{(k)}\left(w^T x^{(k)} + b\right) - 1)$
  $\min_{w,b} L_p$ subject to $\alpha_k \geq 0$, $\frac{\partial L_p}{\partial \alpha_k} = 0$ for all k
- We can show with calculus (but won't) that

$\frac{\partial L_p}{\partial w_m} = 0, \frac{\partial L_p}{\partial b_n} = 0$ for all m, n means $w = \sum_k \alpha_k y^{(k)} x^{(k)}, \sum_k \alpha_k y^{(i)} = 0$

- Substitute this in to get $\left||w|\right|^2 = \left\{\sum_i \alpha_i y^{(i)} x^{(i)}\right\}^T \left\{\sum_j \alpha_j y^{(j)} x^{(j)}\right\} =$
  $\sum_{i,j} \alpha_i \alpha_j y^{(i)} y^{(j)} (x^{(i)} \cdot x^{(j)})$

Dual Formulation for SVM

- Can show (but won't) that we can solve the dual formulation instead of the primal formulation:

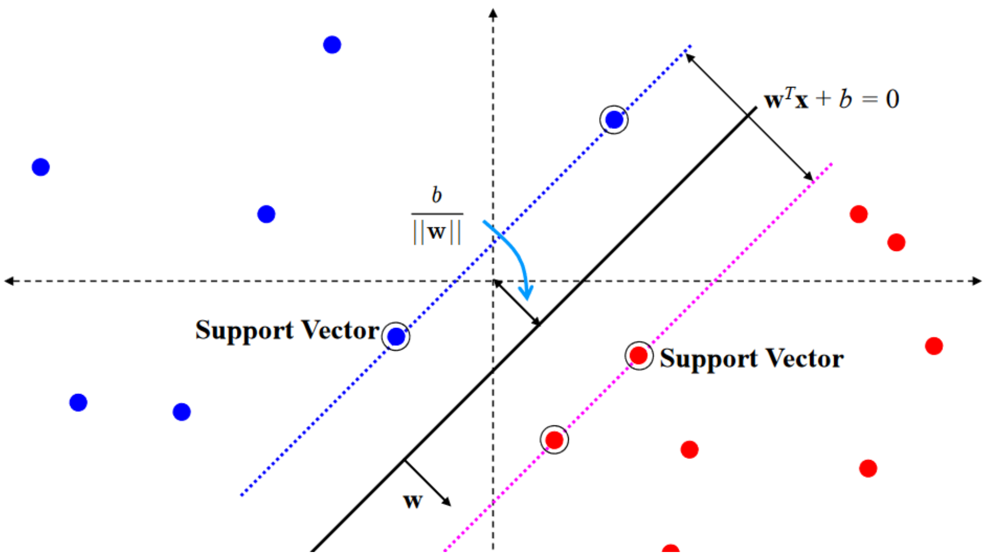
$$\max_{\alpha_k \geq 0} \sum_k \alpha_k - \frac{1}{2} \sum_{jk} \alpha_j \alpha_k y^{(j)} y^{(k)} (x^{(j)} \cdot x^{(k)})$$

subject to $\alpha_k \geq 0$ for all k, $\sum_k \alpha_k y^{(k)} = 0$

- **Representer theorem** (from last slide): $w^* = \sum_k \alpha_k y^{(k)} x^{(k)}$
  - The optimal w is a linear combination of the x's!
- We only need to compute $x^{(j)} \cdot x^{(k)}$ when optimizing and testing
  - Compute $h(x) = \sum_k \alpha_k y^{(k)} \left(x^{(k)} \cdot x\right) + b$
  - Will see why this is important soon
- When the w and x are high-dimensional, but there are few examples, it is more efficient to optimize with respect to the αs

Dual Formulation for SVM

- $h(x) = \sum_k \alpha_k y^{(k)}\left(x^{(k)} \cdot x\right) + b$
  - If $h(x) \geq 0$, predict $y = 1$
- Most $x^{(k)'}s$ don't influence the w so most $\alpha_k's$ will be zero when we solve the optimization problem
- $x^{(k)}$ s.t. $\alpha_k \neq 0$ are the support vectors



Dual Formulation for SVM

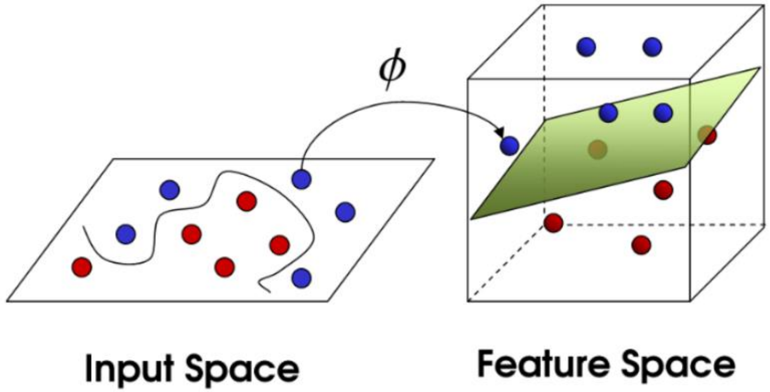
$$\max_{\alpha_k \geq 0} \sum_k \alpha_k - \frac{1}{2} \sum_{jk} \alpha_j \alpha_k y^{(j)} y^{(k)} (x^{(j)} \cdot x^{(k)})$$

subject to $0 \leq \alpha_k \leq \boldsymbol{C}$ for all k, $\sum_k \alpha_k y^{(k)} = 0$

- We are constraining α_k to be smaller than C
- Large C means a more complex model (and smaller training error)

Lifting x to higher dimensions

- We saw before that a lot of the time, data that is not linearly separable can be made separable if we compute nonlinear features of the data
  - Example: the activations of AlexNet
  - Example: compute the distance from x to every one of n examples to simulate 1-Nearest Neighbours
- Compute the features using $\phi(x)$
  - x is low-dimensional, $\phi(x)$ is high-dimensional



The Dual Formulation and the Kernel Trick

- We can sometimes get away with not having to compute ϕ by using a kernel function K
  - $K\left(x^{(i)}, x^{(j)}\right) = \phi\left(x^{(i)}\right) \cdot \phi\left(x^{(j)}\right)$
  - The K and ϕ have to correspond to each other

- Note that here, $\phi\left(x^{(i)}\right)$ is a vector.
- The $\cdot$ in $\phi\left(x^{(i)}\right) \cdot \phi\left(x^{(j)}\right)$ represents the dot product

The Dual Formulation and the Kernel Trick

$$\max_{\alpha_k \geq 0} \sum_k \alpha_k - \frac{1}{2} \sum_{jk} \alpha_j \alpha_k y^{(j)} y^{(k)} (x^{(j)} \cdot x^{(k)})$$

subject to $0 \leq \alpha_k \leq \boldsymbol{C}$ for all k, $\sum_k \alpha_k y^{(k)} = 0$

Now, let's solve the problem when x is mapped to $\phi(x)$

$$\max_{\alpha_k \geq 0} \sum_k \alpha_k - \frac{1}{2} \sum_{jk} \alpha_j \alpha_k y^{(j)} y^{(k)} (\phi(x^{(j)}) \cdot \phi(x^{(k)}))$$

subject to $0 \leq \alpha_k \leq \boldsymbol{C}$ for all k, $\sum_k \alpha_k y^{(k)} = 0$

Equivalently,

$$\boxed{K\left(x^{(i)}, x^{(j)}\right) = \phi\left(x^{(i)}\right) \cdot \phi\left(x^{(j)}\right)}$$

$$\max_{\alpha_k \geq 0} \sum_k \alpha_k - \frac{1}{2} \sum_{jk} \alpha_j \alpha_k y^{(j)} y^{(k)} (K(x^{(j)}, x^{(k)}))$$

subject to $0 \leq \alpha_k \leq \boldsymbol{C}$ for all k, $\sum_k \alpha_k y^{(k)} = 0$

The Dual Formulation and the Kernel Trick

$$\max_{\alpha_k \geq 0} \sum_k \alpha_k - \frac{1}{2} \sum_{jk} \alpha_j \alpha_k y^{(j)} y^{(k)} (K(x^{(j)}, x^{(k)}))$$

subject to $0 \leq \alpha_k \leq \boldsymbol{C}$ for all k, $\sum_k \alpha_k y^{(k)} = 0$

- We map x to a high-dimensional feature space, but only ever have to compute the kernel
  - Solves a computational problem
  - $\phi(x)$ could be infinite-dimensional

The Kernel Trick: example

- $K(a,b) = (a^T b)^3$

$$= \big((a_1, a_2)^T (b_1, b_2)\big)^3$$

$$= (a_1 b_1 + a_2 b_2)^3$$

$\phi(b)$

$$= a_1^3 b_1^3 + 3a_1^2 b_1^2 a_2 b_2 + 3a_1 b_1 a_2^2 b_2^2 + a_2^3 b_2^3$$

$$= \left(a_1^3, \sqrt{3}a_1^2 a_2, \sqrt{3}a_1 a_2^2, a_2^3\right) \cdot \left(b_1^3, \sqrt{3}b_1^2 b, \sqrt{3}b_1 b_2^2, b_2^3\right)$$

$$= \phi(a) \cdot \phi(b)$$

- Can specify K without explicitly writing down the ϕ!
- Reminder: $h(x) = \sum_k \alpha_k y^{(k)}\big(x^{(k)} \cdot x\big) + b$
- So: $h_K(x) = \sum_k \alpha_k y^{(k)}\big(\phi(x^{(k)}) \cdot \phi(x)\big) + b = \sum_k \alpha_k y^{(k)}\big(K(x^{(k)}, x)\big) + b$

Kernels

- To predict: compute $\sum_k \alpha_k y^{(k)}\left(K\left(x^{(k)}, x\right)\right) + b$
- Make sense to weight the $\alpha_k y^{(k)}$ more if $x^{(k)}$ and x are similar
- Polynomial kernel:

$$K\left(x^{(i)}, x^{(j)}\right) = \left(x^{(i)} \cdot x^{(j)} + 1\right)^d, d \geq 1$$

- Gaussian kernel

$$K\left(x^{(i)}, x^{(j)}\right) = \exp\left(-\frac{\left\|x^{(i)} - x^{(j)}\right\|^2}{2\sigma^2}\right)$$

- Sigmoid kernel

$$K\left(x^{(i)}, x^{(j)}\right) = \tanh\left(\beta\left(x^{(i)^T} x^{(j)} + a\right)\right)$$

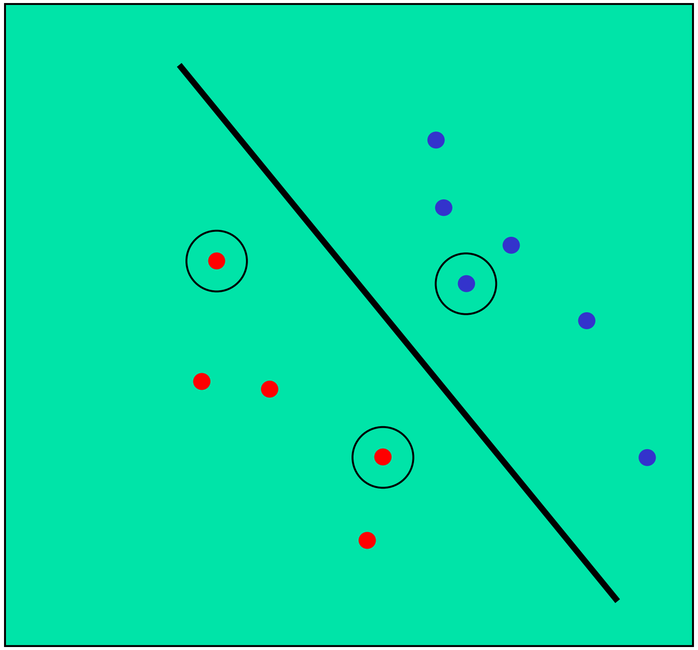
Kernels

- Mercer's Theorem: any "reasonable" kernel corresponds to some ϕ
- Polynomial kernels $\left(x^{(i)} \cdot x^{(j)} + 1\right)^d$ correspond to features spaces of size exponential in d
- The Gaussian kernel corresponds to a ϕ that maps x to an infinite-dimensional space

$$
\begin{aligned}
\exp\left(-\frac{1}{2}\|\mathbf{x}-\mathbf{x}'\|^2\right) &= \sum_{j=0}^{\infty} \frac{(\mathbf{x}^\top \mathbf{x}')^j}{j!} \exp\left(-\frac{1}{2}\|\mathbf{x}\|^2\right) \exp\left(-\frac{1}{2}\|\mathbf{x}'\|^2\right) \\
&= \sum_{j=0}^{\infty} \sum_{\sum n_i = j} \exp\left(-\frac{1}{2}\|\mathbf{x}\|^2\right) \frac{x_1^{n_1} \cdots x_k^{n_k}}{\sqrt{n_1! \cdots n_k!}} \exp\left(-\frac{1}{2}\|\mathbf{x}'\|^2\right) \frac{{x'}_1^{n_1} \cdots {x'}_k^{n_k}}{\sqrt{n_1! \cdots n_k!}}
\end{aligned}
$$

SVMs: Summary

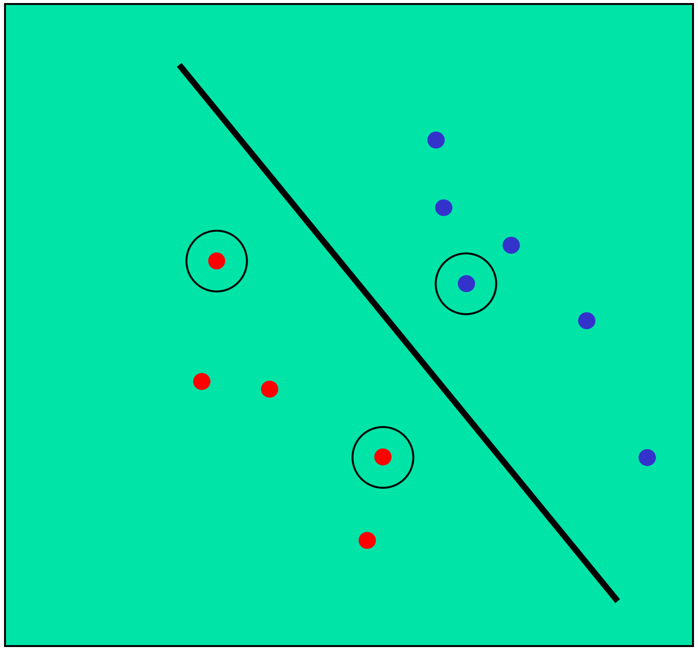
- Find the maximum-margin linear classifier
  - Possibly map the inputs to a high-dimensional space first
  - Maximum-margin classifiers try to avoid overfitting
  - Can efficiently map to very high-dimensional spaces
- The maximum-margin classifier is defined by the *support vectors*
  - Support vectors are points in the training set



- Classify using
  $\sum_k \alpha_k y^{(k)} K(x, x^{(k)}) + b \geq 0$
- Can ignore non-support vectors
- When using soft margin, select C using cross-validation
- Select kernel by thinking about the data, or by cross-validation
  - What makes two x's "close"?

SVMs: Summary

- Kernels allow for very flexible hypotheses
  - But must choose kernel parameters
- Exact optimization methods available
  - Batch algorithm – the entire training set needs to be in memory when learning
- Work well in practice
- Several good implementations available



What you need to know

- Maximum margin
- The primal formulation for SVMs
- Hinge loss vs. log-loss (Soft-margin SVMs vs. Logistic Regression)
- Getting from the Representer Theorem (slide 40) to prediction of new outputs in an SVM is we know the solution to the dual problem
- The kernel trick
- Why kernels that compute similarity make sense
- The intuition behind the prediction of new outputs
- SVMs work well in practice
- Don't need to know:
  - Anything involving Lagrange multipliers
  - Getting from the primal formulation to the dual formulation, even to the extent that we did that in class
  - How to solve QP problems