L7: Support Vector Machines

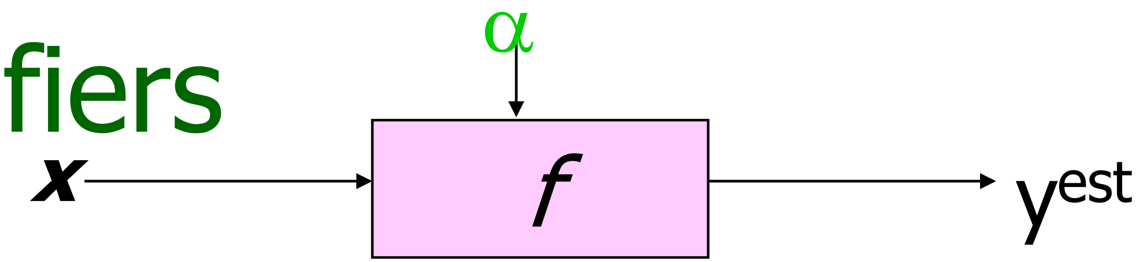
Modified from Prof. Andrew W. Moore's Lecture Notes

www.cs.cmu.edu/~awm

History

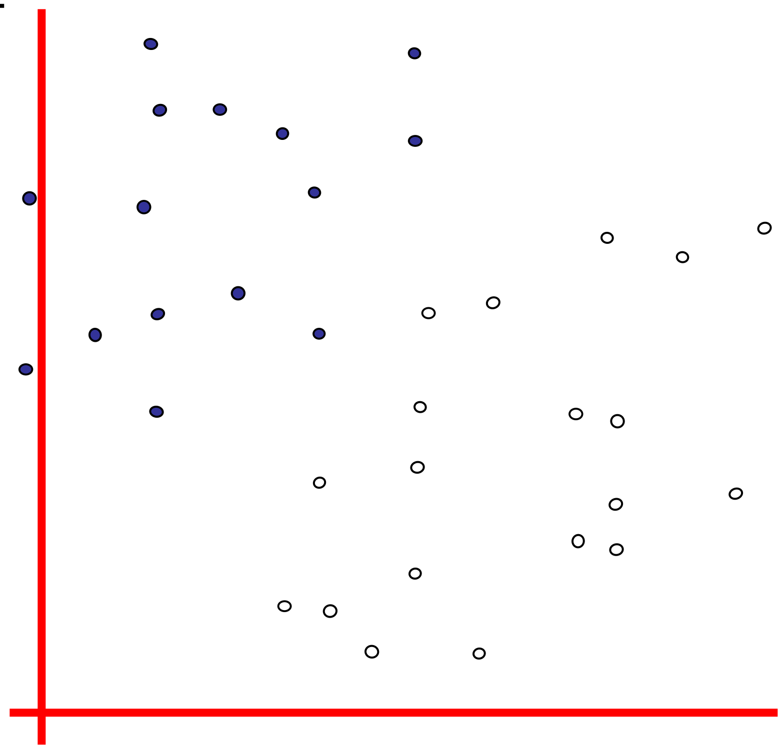
- SVM is a classifier derived from statistical learning theory by Vapnik and Chervonenkis
- SVMs introduced by Boser, Guyon, Vapnik in COLT-92
- Now an important and active field of all Machine Learning research.
- Special issues of Machine Learning Journal, and Journal of Machine Learning Research.

Linear Classifiers



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

- denotes +1
- denotes -1

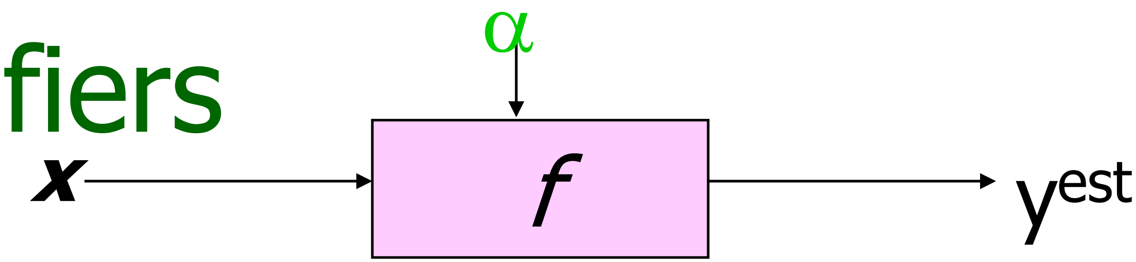


How would you classify this data?

e.g. $\boldsymbol{x} = (x_1, x_2)$, $\boldsymbol{w} = (w_1, w_2)$, $\boldsymbol{w.x} = x_1w_1 + w_2x_2$

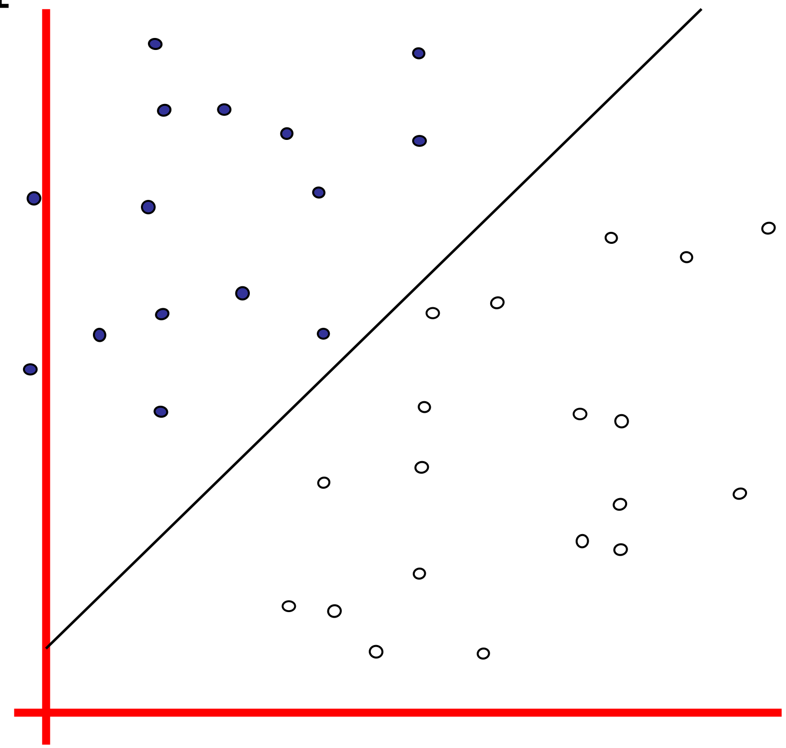
$sign(\boldsymbol{w}.\,\boldsymbol{x} + b) = +1$ *iff* $x_1w_1 + w_2x_2 - b > 0$

Linear Classifiers



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

- ● denotes +1
- ○ denotes -1

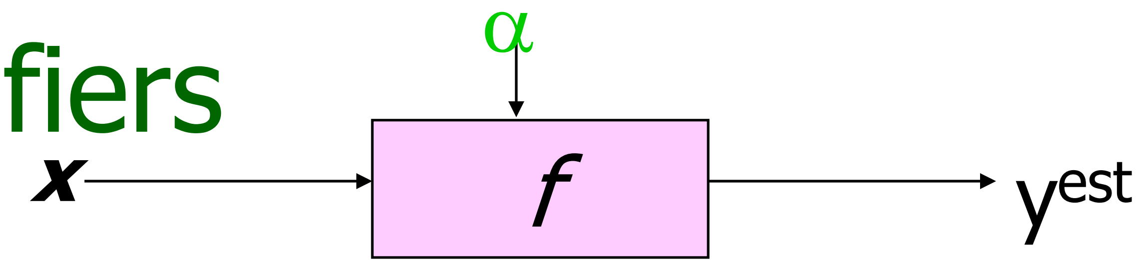


How would you classify this data?

e.g. $\boldsymbol{x} = (x_1, x_2)$, $\boldsymbol{w} = (w_1, w_2)$, $\boldsymbol{w}.\boldsymbol{x} = x_1 w_1 + w_2 x_2$

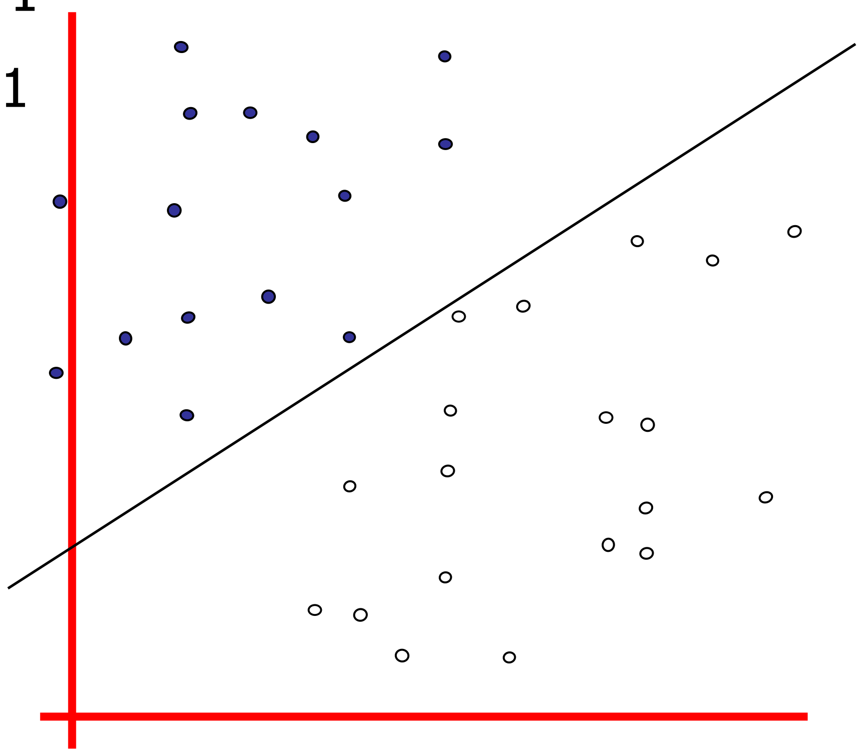
$sign(\boldsymbol{w}. \boldsymbol{x} + b) = +1$ *iff* $x_1 w_1 + w_2 x_2 - b > 0$

Linear Classifiers



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

- denotes +1
- denotes -1

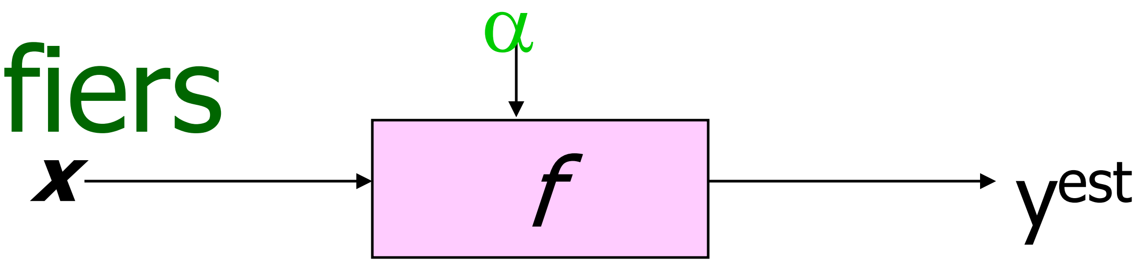


How would you classify this data?

e.g. $\mathbf{x} = (x_1, x_2)$, $\mathbf{w} = (w_1, w_2)$, $\mathbf{w.x} = x_1w_1 + w_2x_2$

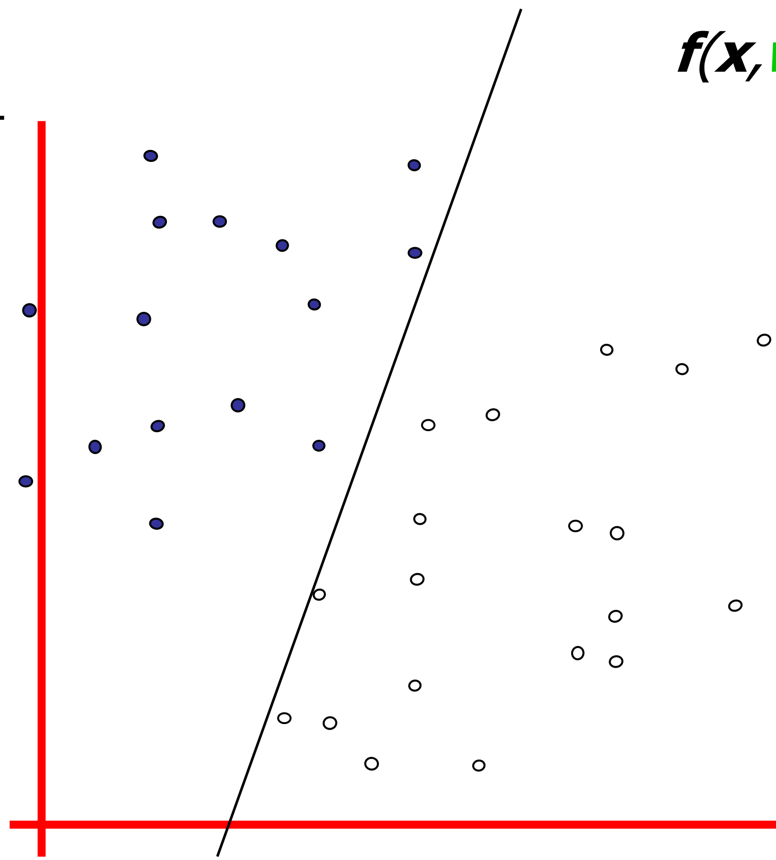
$sign(\mathbf{w}.\ \mathbf{x} + b) = +1 \quad \textit{iff} \quad x_1w_1 + w_2x_2 - b > 0$

Linear Classifiers



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

- denotes +1
- denotes -1

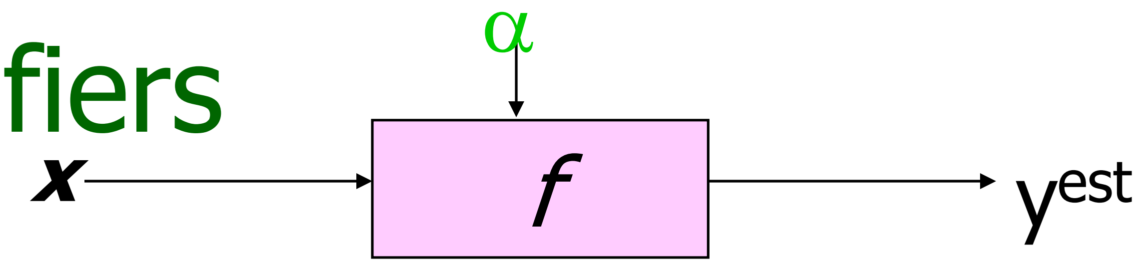


How would you classify this data?

e.g. $\mathbf{x} = (x_1, x_2)$, $\mathbf{w} = (w_1, w_2)$, $\mathbf{w.x} = x_1w_1 + w_2x_2$

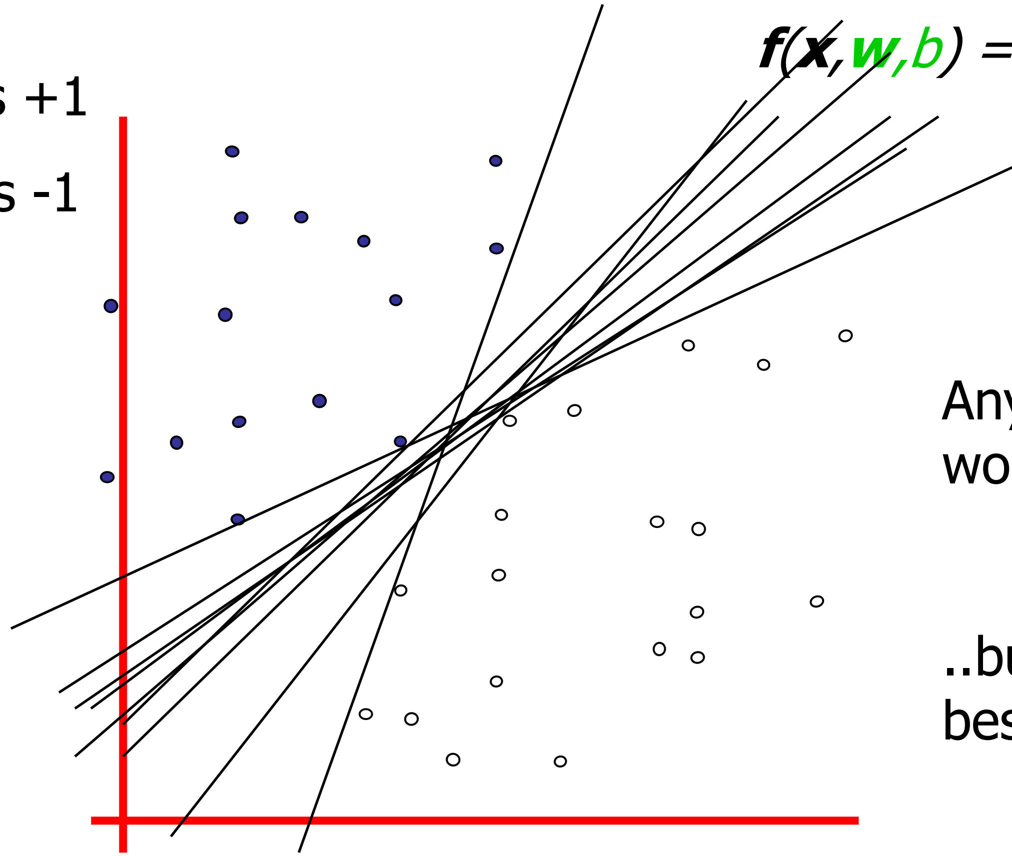
$sign(\mathbf{w}.\ \mathbf{x} + b) = +1$ *iff* $x_1w_1 + w_2x_2 - b > 0$

Linear Classifiers



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

- denotes +1
- denotes -1



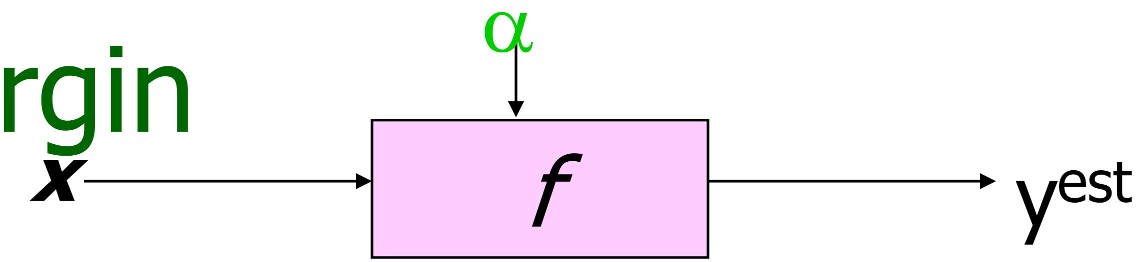
Any of these would be fine..

..but which is best?

e.g. $\boldsymbol{x} = (x_1, x_2)$, $\boldsymbol{w} = (w_1, w_2)$, $\boldsymbol{w.x} = x_1w_1 + w_2x_2$

$sign(\boldsymbol{w}.\ \boldsymbol{x} + b) = +1$ *iff* $x_1w_1 + w_2x_2 + b > 0$

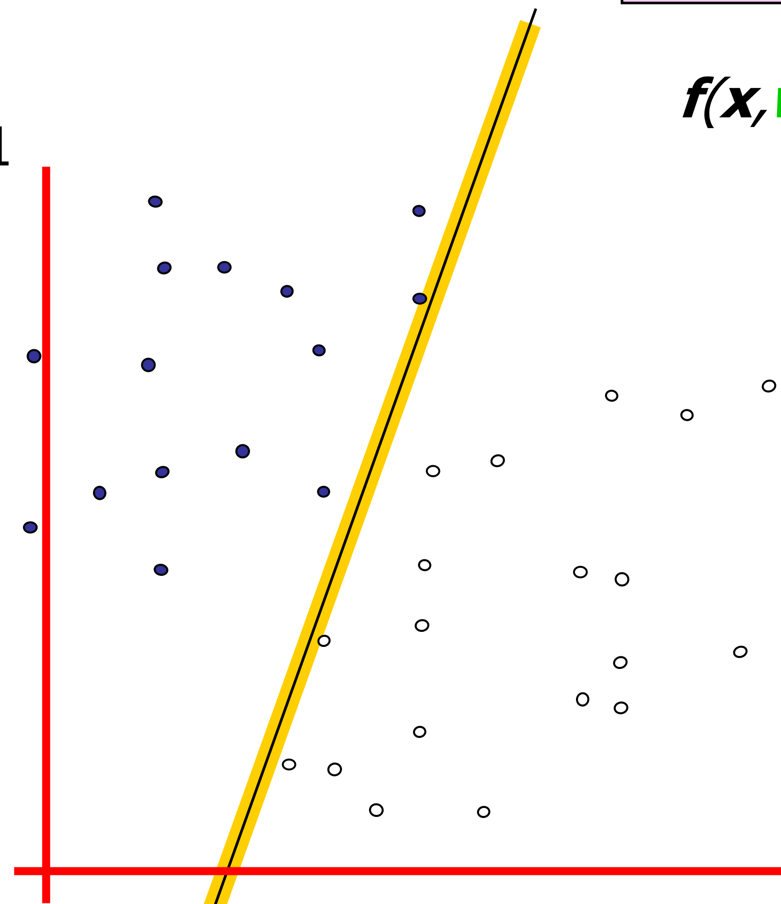
Classifier Margin



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

● denotes +1

○ denotes -1

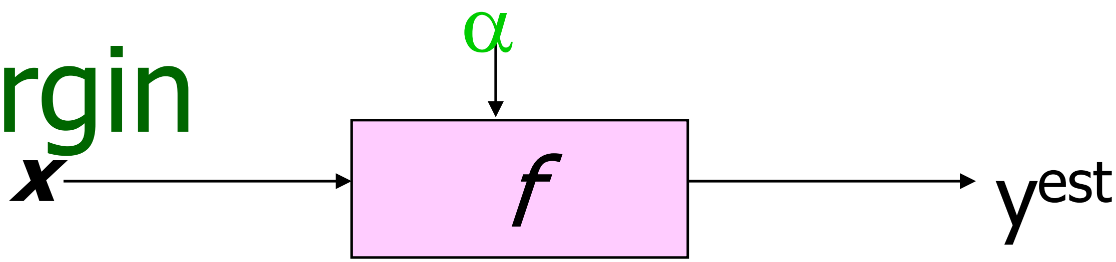


Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

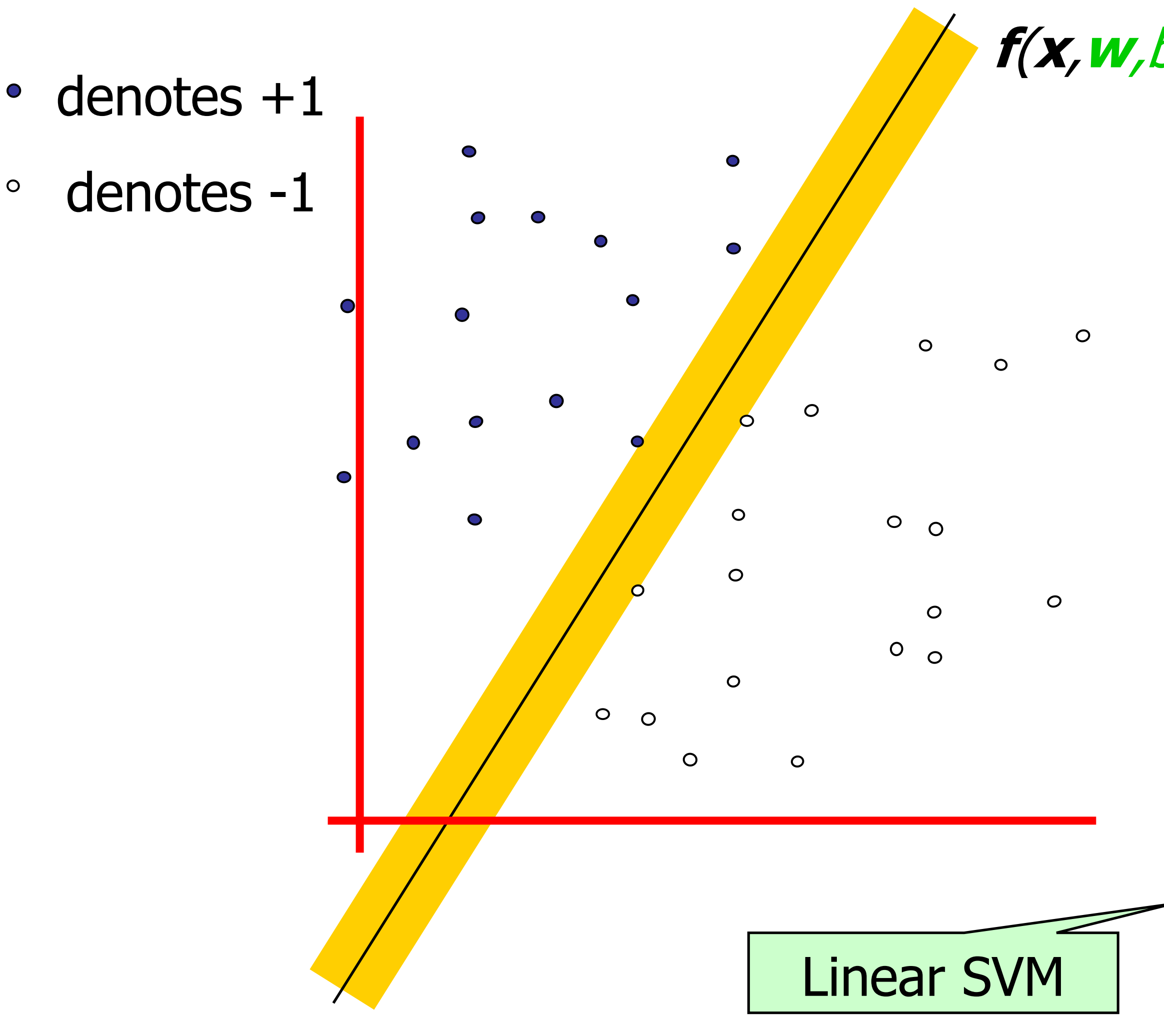
e.g. $\mathbf{x} = (x_1, x_2)$, $\mathbf{w} = (w_1, w_2)$, $\mathbf{w.x} = x_1w_1 + w_2x_2$

$sign(\mathbf{w}.\ \mathbf{x} + b) = +1 \quad iff \quad x_1w_1 + w_2x_2 - b > 0$

Maximum Margin



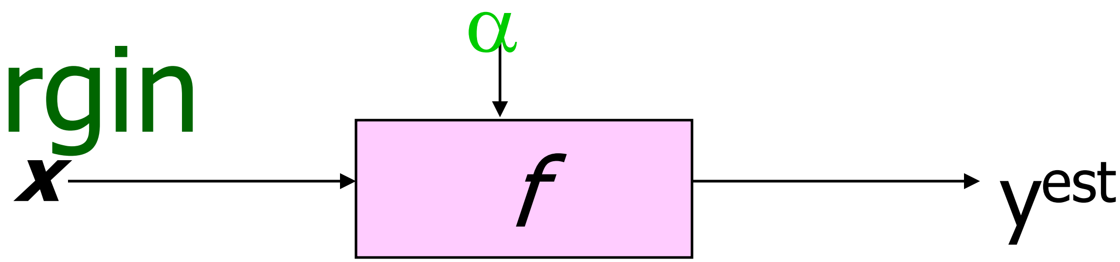
$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}\mathbf{.}\ \mathbf{x} + b)$



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

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

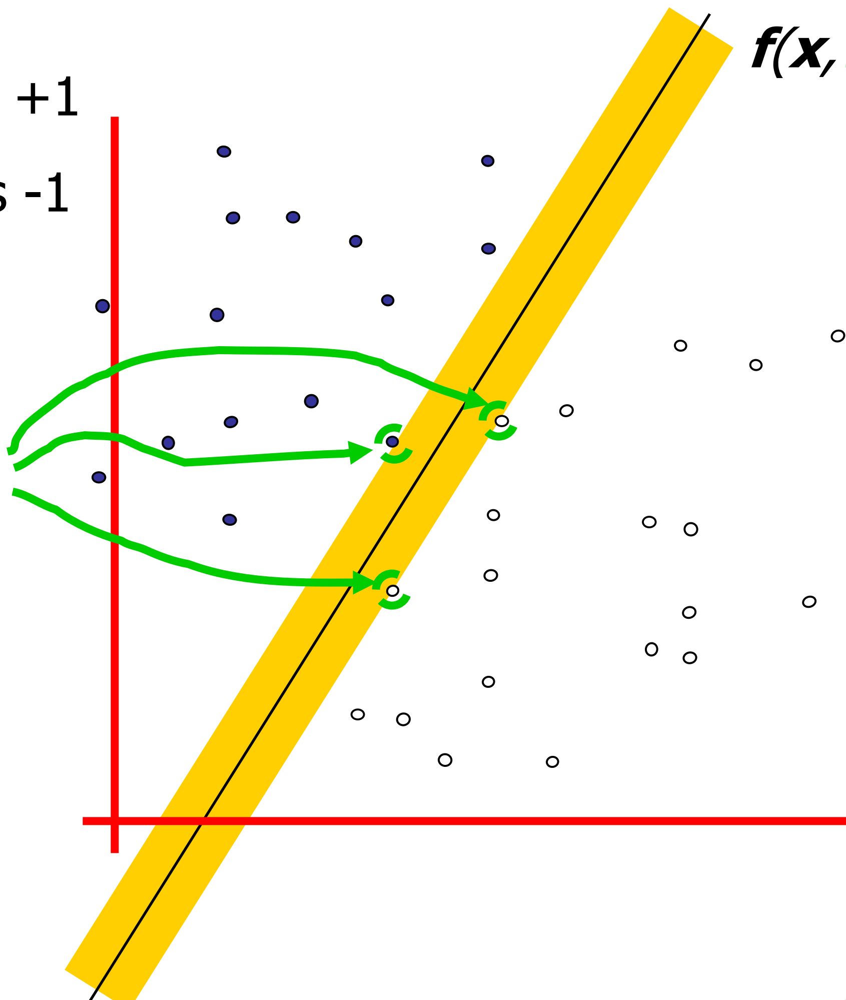
Maximum Margin



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

- 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)

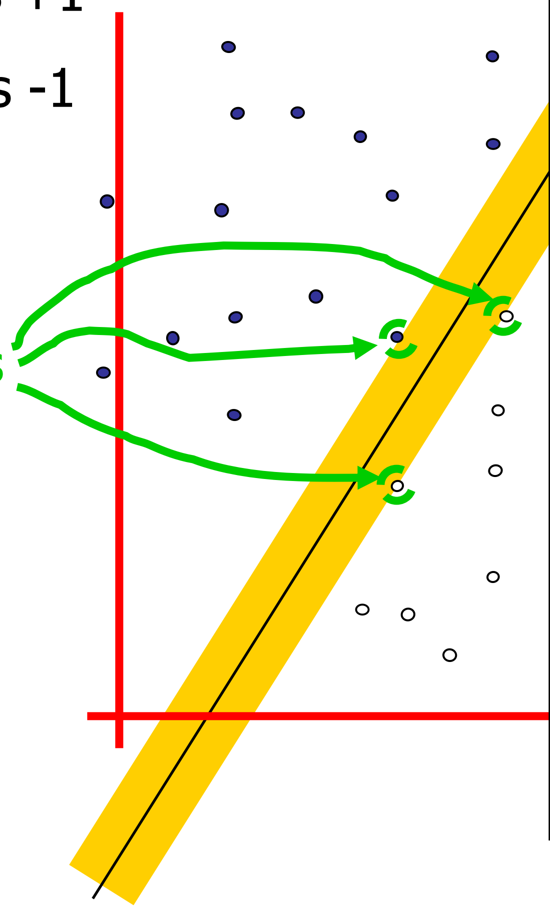
Linear SVM

Why Maximum Margin?

• denotes +1

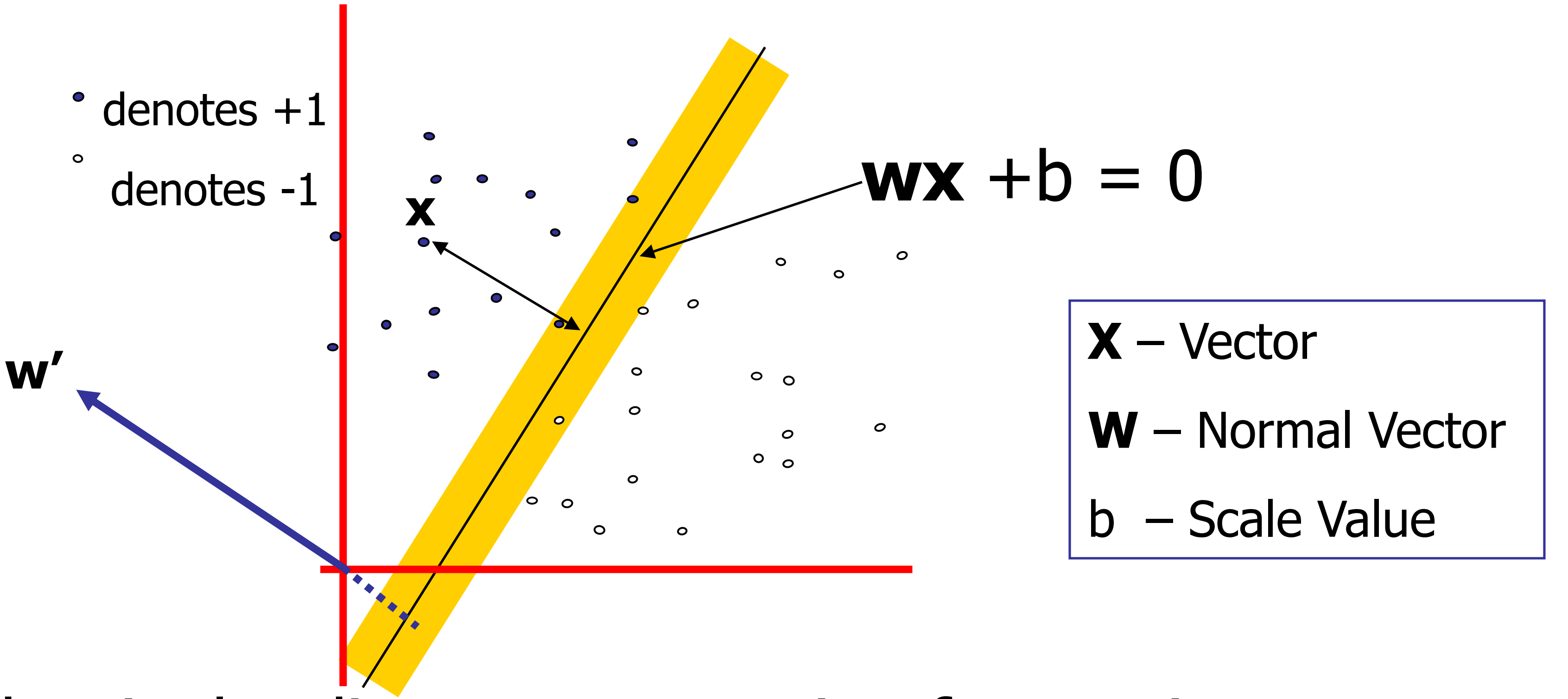
° denotes -1

Support Vectors are those datapoints that the margin pushes up against



1. **Intuitively this feels safest.**
2. **Empirically it works very well.**
3. 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.
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.

Estimate the Margin

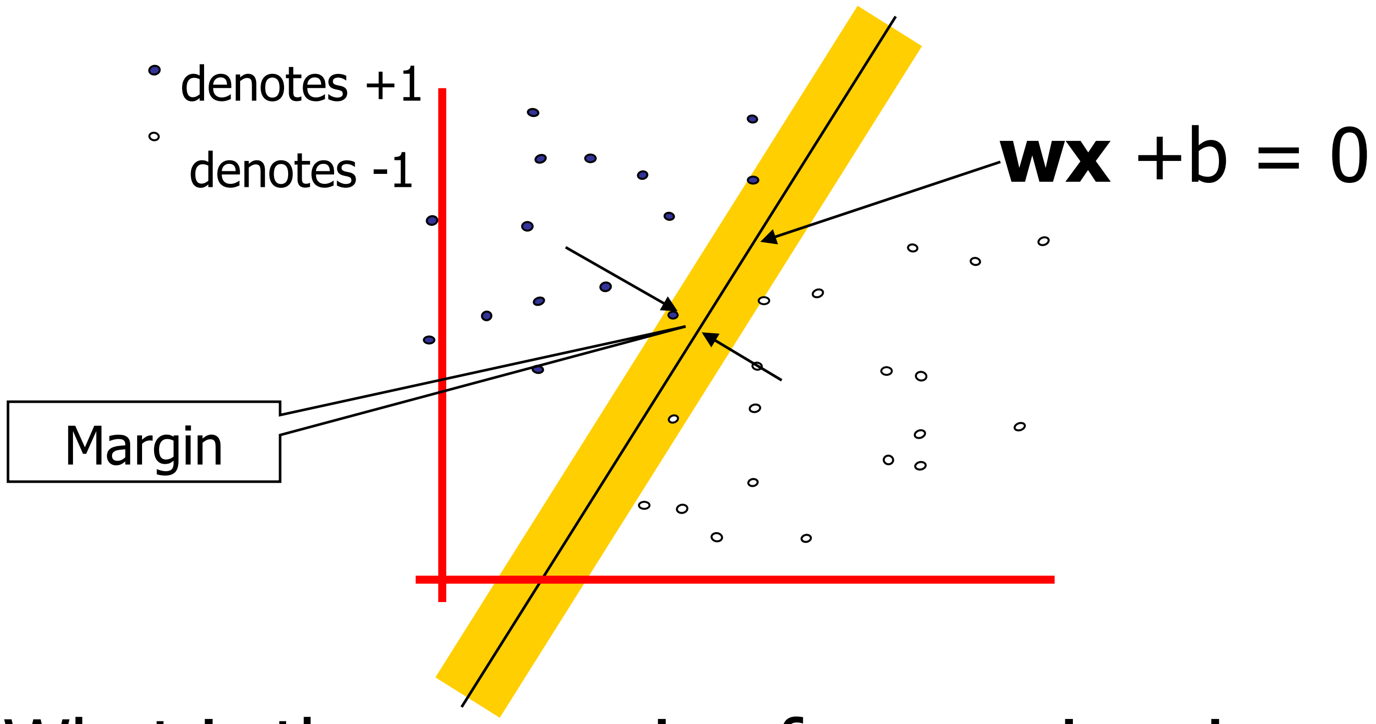


- What is the distance expression for a point **x** to a line **wx**+b= 0?

$$d(\mathbf{x}) = \frac{\left|\mathbf{x}\cdot\mathbf{w}+b\right|}{\sqrt{\left\|\mathbf{w}\right\|_2^2}} = \frac{\left|\mathbf{x}\cdot\mathbf{w}+b\right|}{\sqrt{\sum_{i=1}^{d} w_i^2}}$$

http://mathworld.wolfram.com/Point-LineDistance2-Dimensional.html

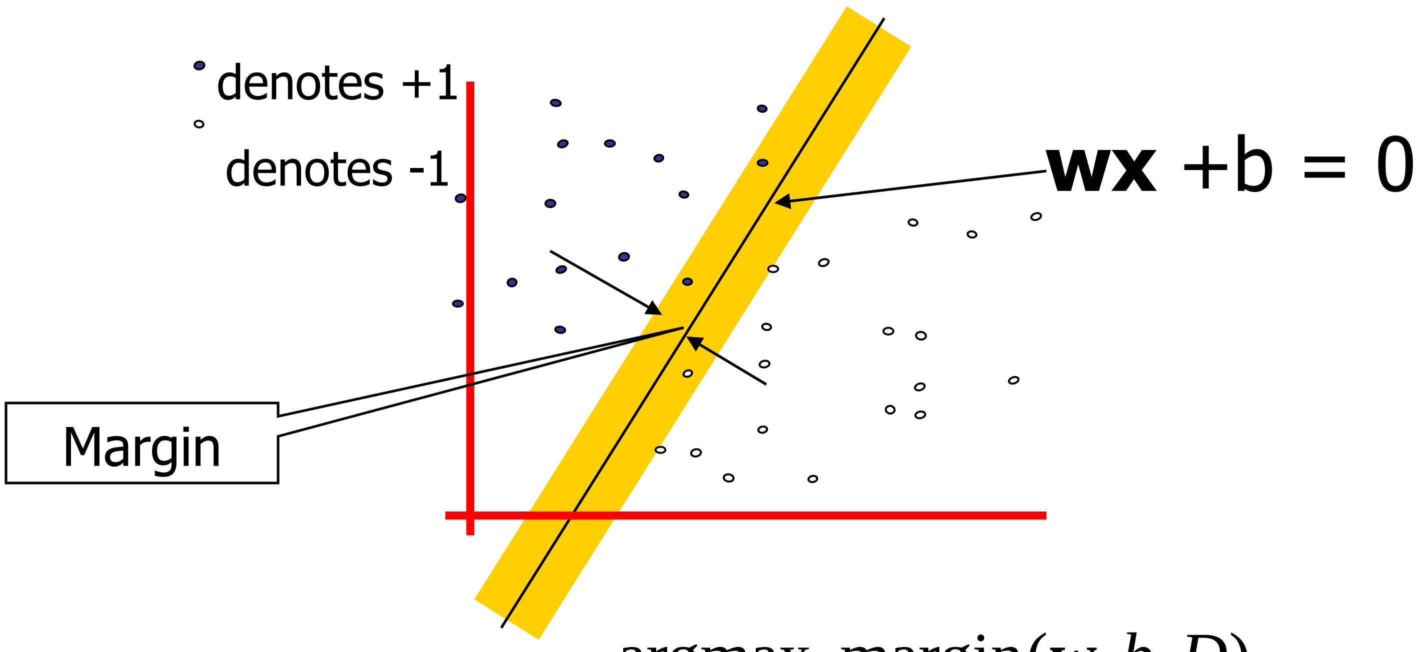
Estimate the Margin



- What is the expression for margin, given **w** and b?

$$\text{margin} \equiv \arg\min_{\mathbf{x}\in D} d(\mathbf{x}) = \arg\min_{\mathbf{x}\in D} \frac{\left|\mathbf{x}\cdot\mathbf{w}+b\right|}{\sqrt{\sum_{i=1}^{d} w_i^2}}$$

Maximize Margin

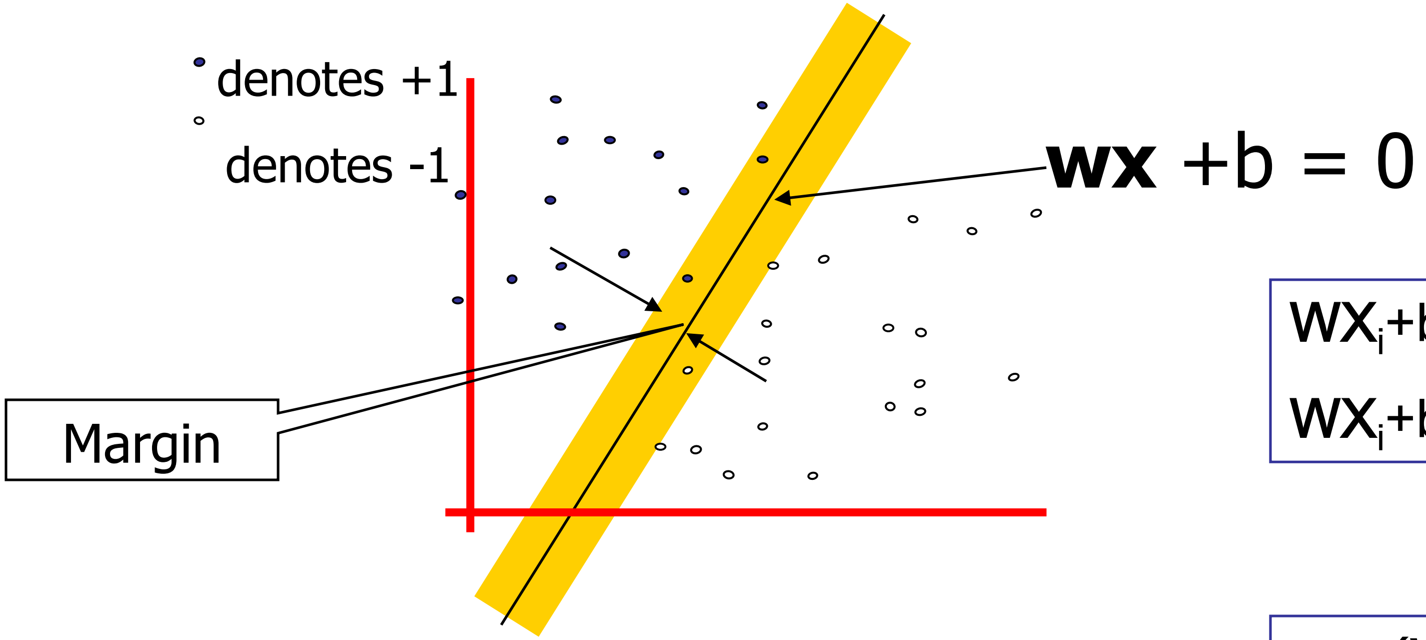


$$\underset{\mathbf{w},b}{\operatorname{argmax}}\ \operatorname{margin}(\mathbf{w},b,D)$$

$$= \underset{\mathbf{w},b}{\operatorname{argmax}}\ \underset{\mathbf{x}_i \in D}{\arg\min}\ d(\mathbf{x}_i)$$

$$= \underset{\mathbf{w},b}{\operatorname{argmax}}\ \underset{\mathbf{x}_i \in D}{\arg\min}\ \frac{\left|b + \mathbf{x}_i \cdot \mathbf{w}\right|}{\sqrt{\sum_{i=1}^{d} w_i^2}}$$

Maximize Margin



$wx_i+b \geq 0$ iff $y_i=1$

$wx_i+b \leq 0$ iff $y_i=-1$

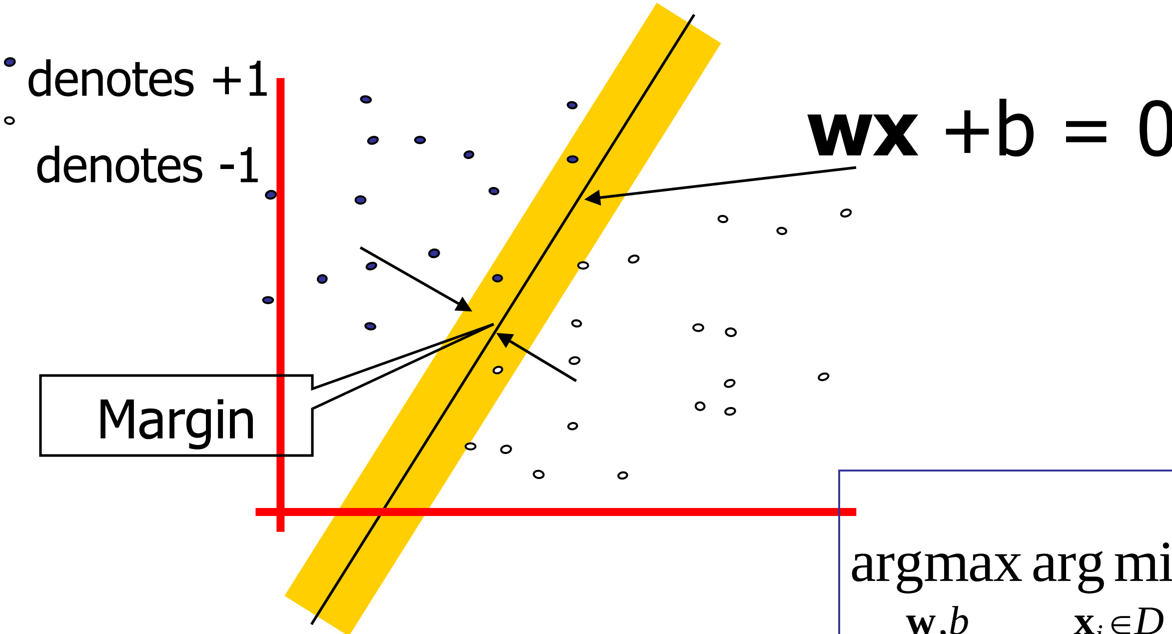
$\Downarrow$

$y_i(wx_i+b) \geq 0$

$$\underset{\mathbf{w},b}{\operatorname{argmax}} \underset{\mathbf{x}_i \in D}{\arg\min} \frac{\left|b + \mathbf{x}_i \cdot \mathbf{w}\right|}{\sqrt{\sum_{i=1}^{d} w_i^2}}$$

$$\text{subject to } \forall \mathbf{x}_i \in D : y_i\left(\mathbf{x}_i \cdot \mathbf{w} + b\right) \geq 0$$

Maximize Margin



$\mathbf{wx}_i+b \geq 0$ iff $y_i=1$

$\mathbf{wx}_i+b \leq 0$ iff $y_i=-1$

$y_i(\mathbf{WX}_i+b) \geq 0$

$$\underset{\mathbf{w},b}{\operatorname{argmax}} \; \underset{\mathbf{x}_i \in D}{\arg\min} \frac{\left|b + \mathbf{x}_i \cdot \mathbf{w}\right|}{\sqrt{\sum_{i=1}^{d} w_i^2}}$$

$$\text{subject to } \; \forall \mathbf{x}_i \in D : y_i\left(\mathbf{x}_i \cdot \mathbf{w} + b\right) \geq 0$$

Strategy:

$$\forall \mathbf{x}_i \in D: \; \left|b + \mathbf{x}_i \cdot \mathbf{w}\right| \geq 1$$

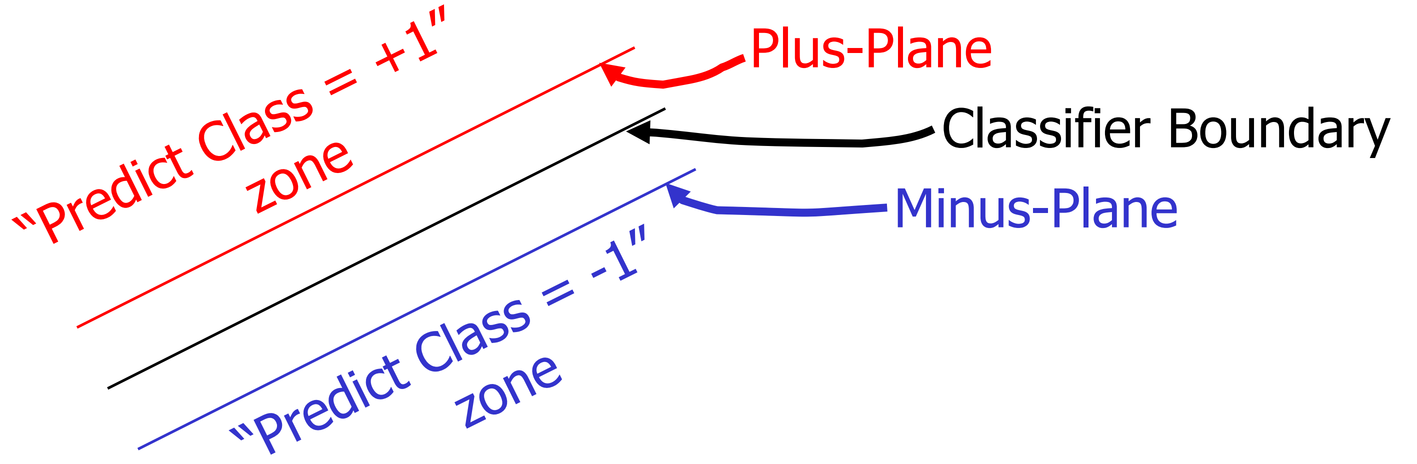
$$\underset{\mathbf{w},b}{\operatorname{argmin}} \sum_{i=1}^{d} w_i^2$$

$$\text{subject to } \; \forall \mathbf{x}_i \in D : y_i\left(\mathbf{x}_i \cdot \mathbf{w} + b\right) \geq 1$$

$\mathbf{wx} + b = 0$

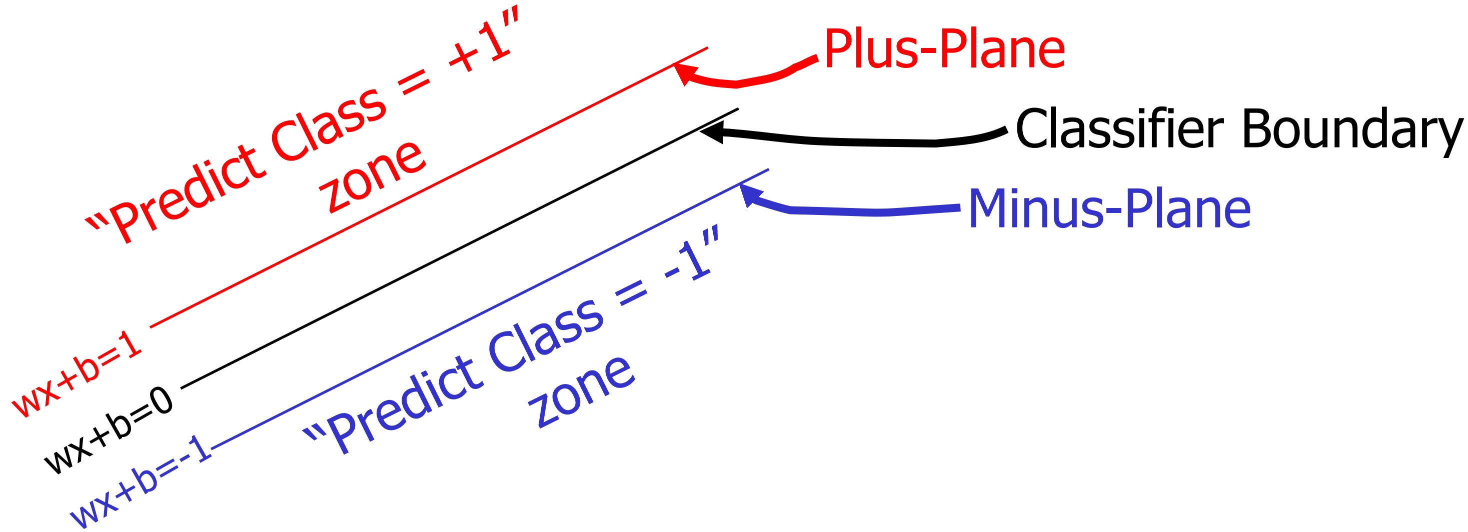
$\alpha(\mathbf{wx} + b) = 0$ where $\alpha \neq 0$

Specifying a line and margin



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

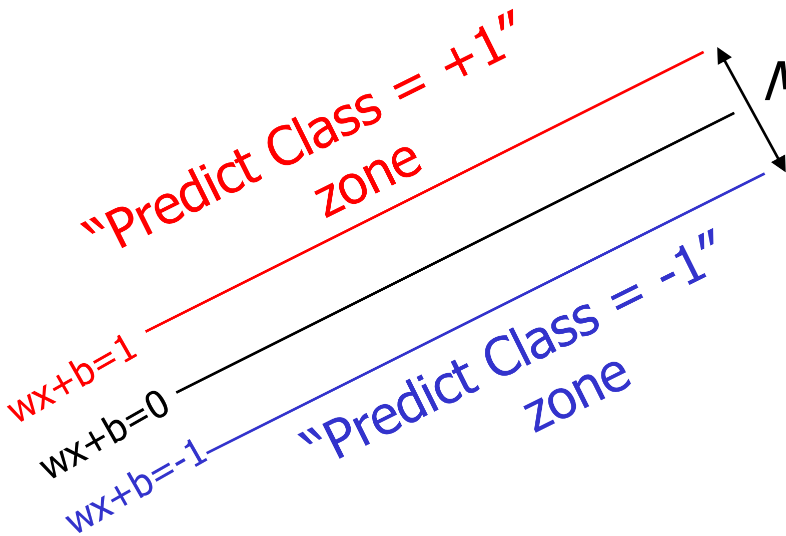
Specifying a line and margin



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

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

Computing the margin width



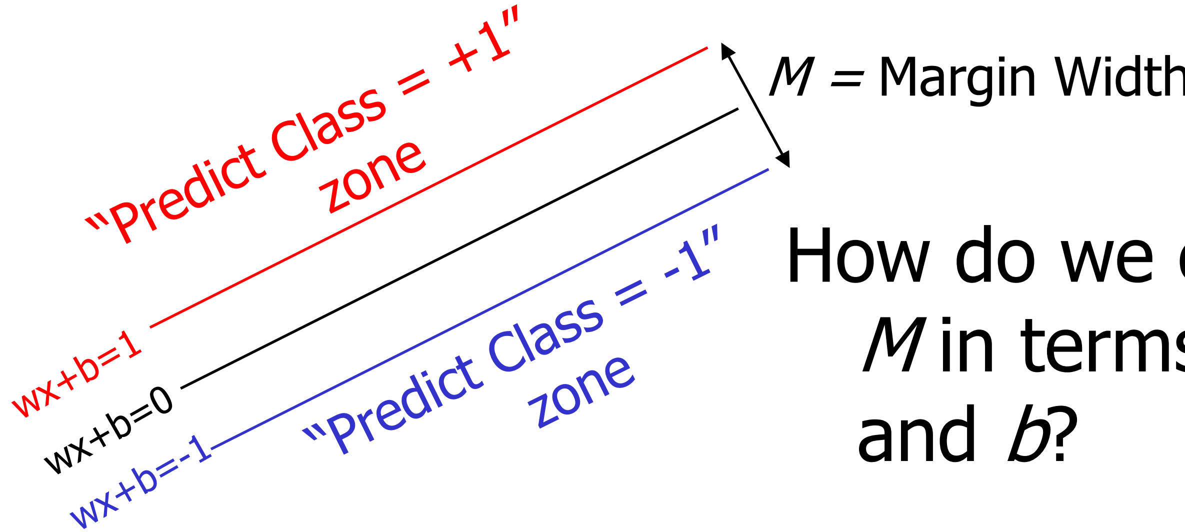
M = Margin Width

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

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

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

Computing the margin width



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

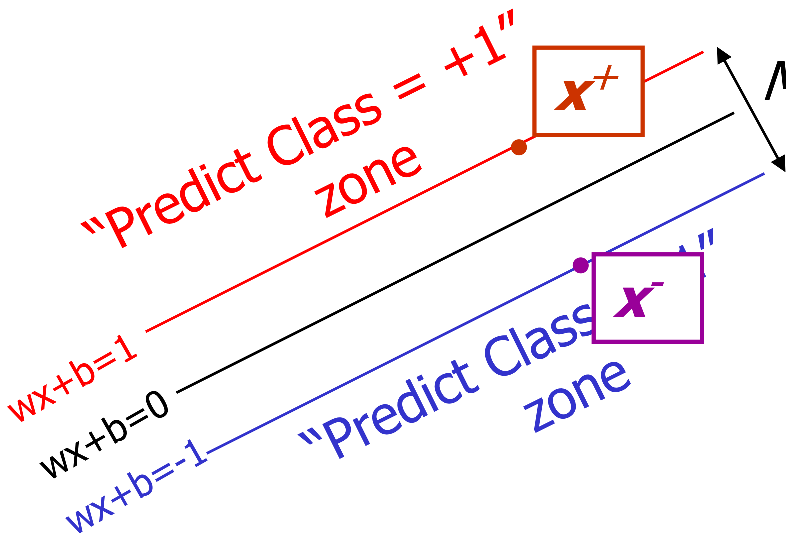
- Plus-plane = $\{ \boldsymbol{x} : \boldsymbol{w} . \boldsymbol{x} + b = +1 \}$
- Minus-plane = $\{ \boldsymbol{x} : \boldsymbol{w} . \boldsymbol{x} + b = -1 \}$

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

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

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

Computing the margin width



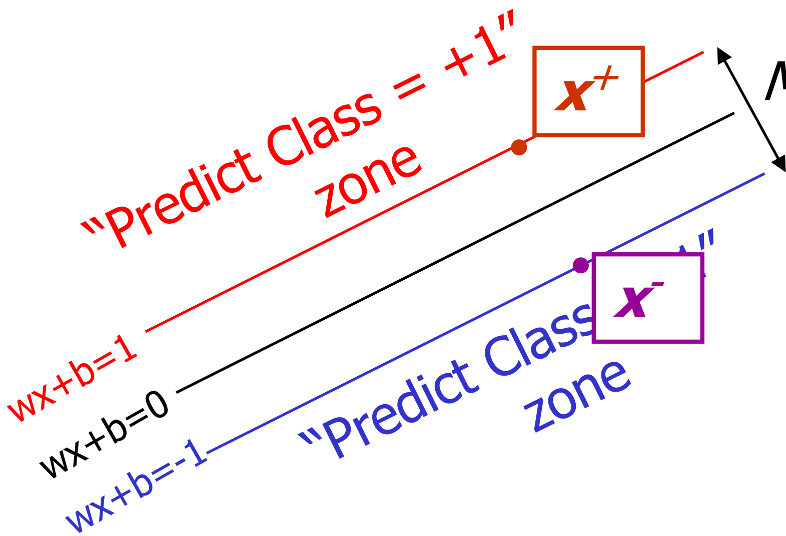
M = Margin Width

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

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

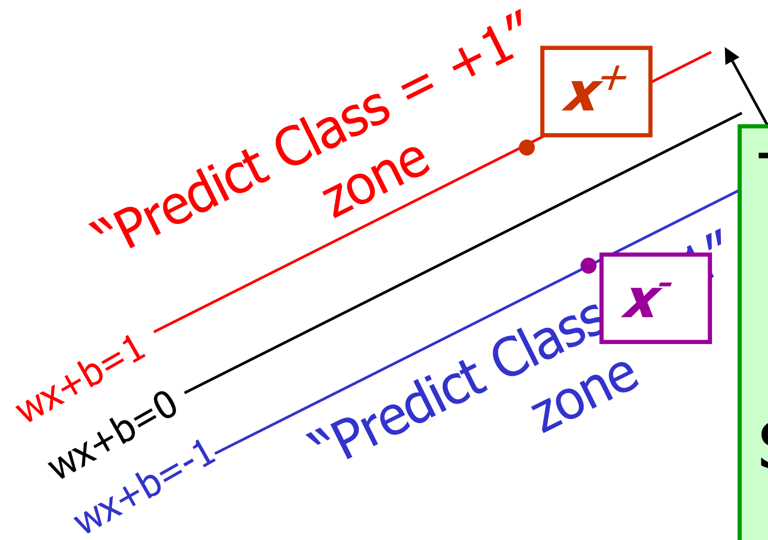
Computing the margin width



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

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

Computing the margin width

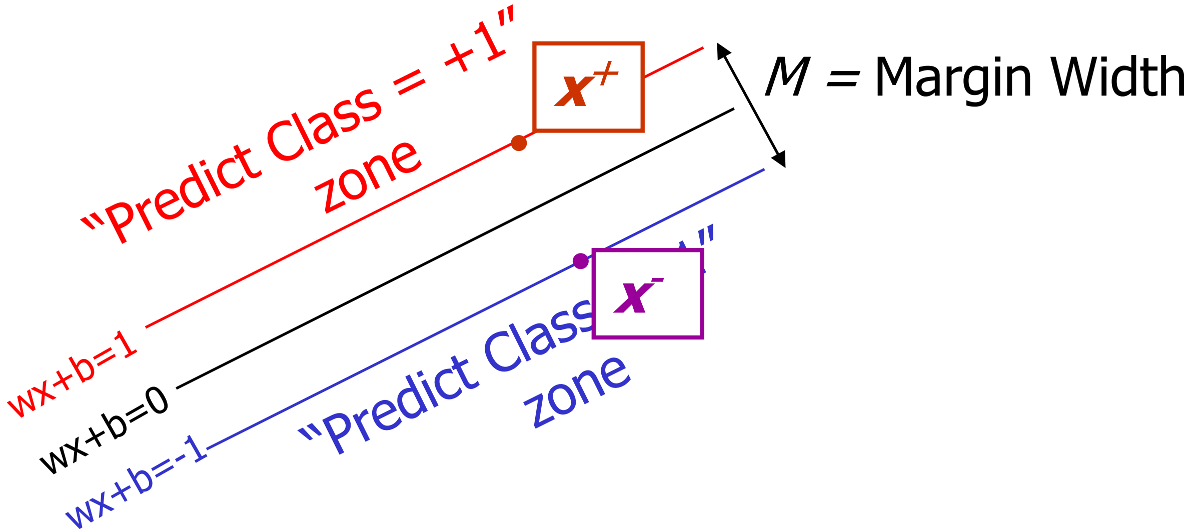


The line from $\boldsymbol{x}^-$ to $\boldsymbol{x}^+$ is perpendicular to the planes.

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

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

Computing the margin width



What we know:

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

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

Maximize Margin

- How does it come ?

$$\underset{\mathbf{w},b}{\operatorname{argmax}} \underset{\mathbf{x}_i \in D}{\arg\min} \frac{\left|b + \mathbf{x}_i \cdot \mathbf{w}\right|}{\sqrt{\sum_{i=1}^{d} w_i^2}}$$

$$\text{subject to } \forall \mathbf{x}_i \in D : y_i\left(\mathbf{x}_i \cdot \mathbf{w} + b\right) \geq 0$$

$$\forall \mathbf{x}_i \in D : \left|b + \mathbf{x}_i \cdot \mathbf{w}\right| \geq 1$$

$$\underset{\mathbf{w},b}{\operatorname{argmin}} \sum_{i=1}^{d} w_i^2$$

$$\text{subject to } \forall \mathbf{x}_i \in D : y_i\left(\mathbf{x}_i \cdot \mathbf{w} + b\right) \geq 1$$

We have

$$\arg\min \frac{|b + x_i . w|}{\sqrt{\sum_{i=1}^{d} w_i^2}} = \arg\min \frac{|b + x_i . w| \times K}{\sqrt{\sum_{i=1}^{d} w_i^2 \times K}} = \frac{1}{\sqrt{\sum_{i=1}^{d} w'^2_i}}$$

Thus,

$$\arg\max \arg\min \frac{|b + x_i . w|}{\sqrt{\sum_{i=1}^{d} w_i^2}} = \arg\max \frac{1}{\sqrt{\sum_{i=1}^{d} w'^2_i}} = \arg\min \sum_{i=1}^{d} w'^2_i$$

Maximum Margin Linear Classifier

$$\{\vec{w}^*, b^*\} = \underset{\vec{w},b}{\operatorname{argmax}} \sum_{k=1}^{d} w_k^2$$

subject to

$$y_1\left(\vec{w}\cdot\vec{x}_1 + b\right) \geq 1$$

$$y_2\left(\vec{w}\cdot\vec{x}_2 + b\right) \geq 1$$

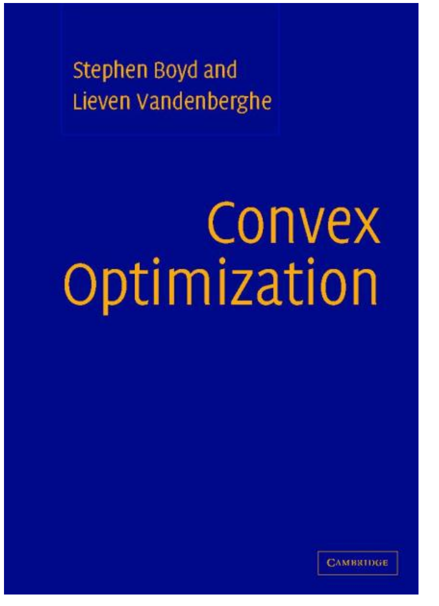
....

$$y_N\left(\vec{w}\cdot\vec{x}_N + b\right) \geq 1$$

- How to solve it?

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.
- Detail solution of Quadratic Programming
  - Convex Optimization Stephen P. Boyd
  - Online Edition, Free for Downloading



www.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf

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 <u>in</u>equality constraints

Quadratic Programming for the Linear Classifier

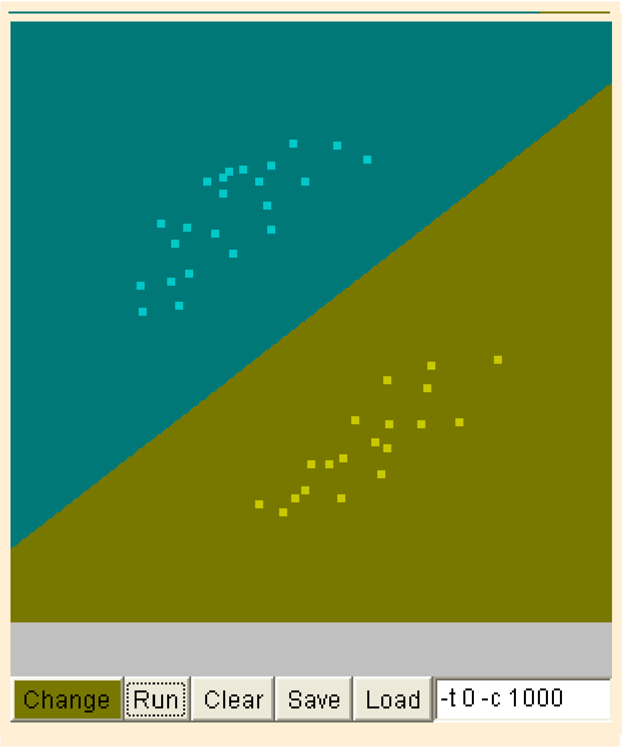
$$\{\vec{w}^*, b^*\} = \min_{\vec{w},b} \sum_i w_i^2$$

subject to $y_i\left(\vec{w}\cdot\vec{x}_i + b\right) \geq 1$ for all training data $(\vec{x}_i, y_i)$

$$\{\vec{w}^*, b^*\} = \underset{\vec{w},b}{\operatorname{argmax}} \left\{0 + \vec{0}\cdot\vec{w} - \vec{w}^T \mathbf{I_n} \vec{w}\right\}$$

$$\left.\begin{array}{l} y_1\left(\vec{w}\cdot\vec{x}_1 + b\right) \geq 1 \\ y_2\left(\vec{w}\cdot\vec{x}_2 + b\right) \geq 1 \\ \cdots\cdot \\ y_N\left(\vec{w}\cdot\vec{x}_N + b\right) \geq 1 \end{array}\right\} \text{inequality constraints}$$

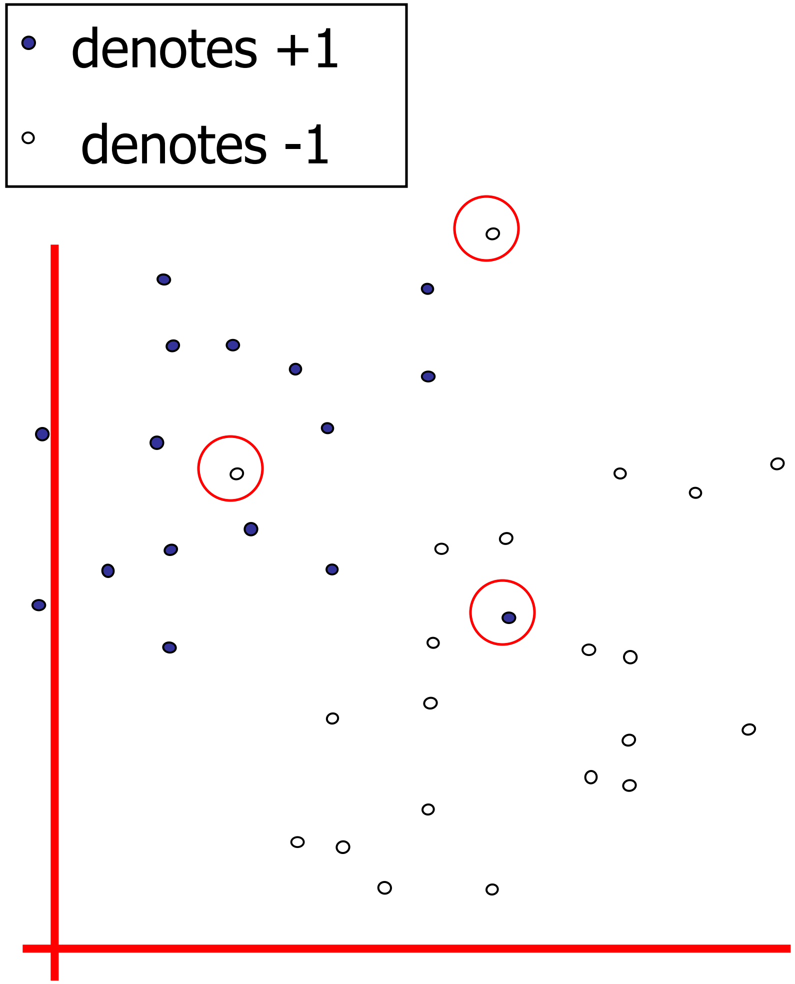
- Popular Tools - LibSVM



Uh-oh!

This is going to be a problem!

What should we do?



Uh-oh!

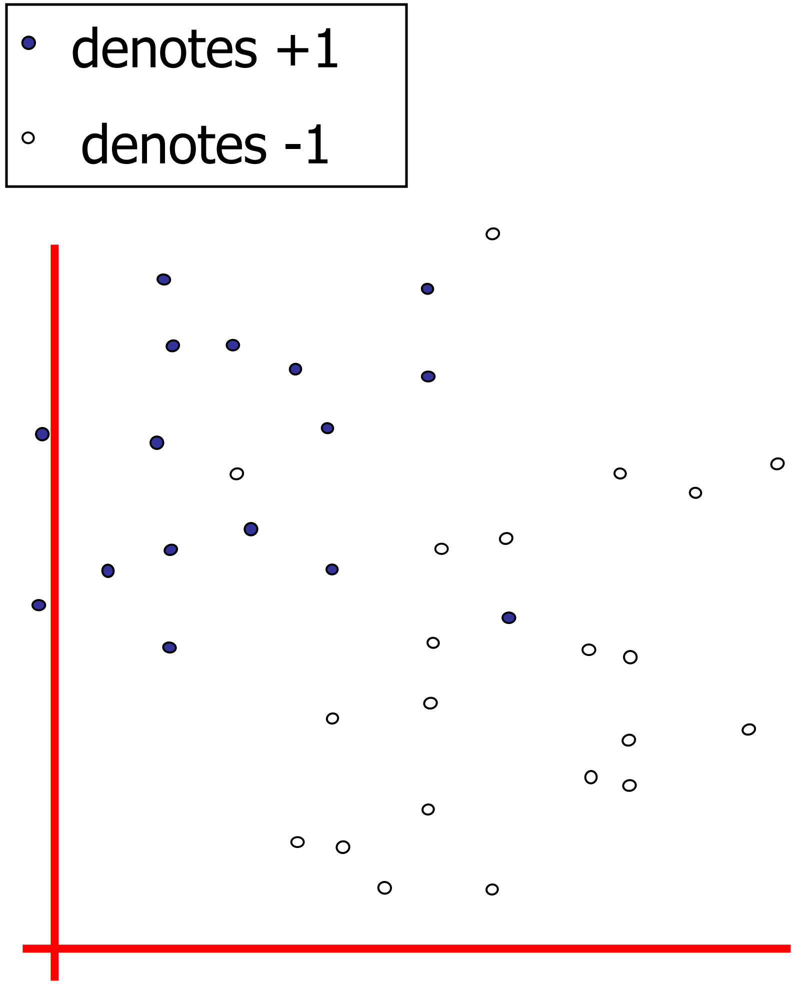
This is going to be a problem!

What should we do?

Idea 1:

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

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



Uh-oh!

This is going to be a problem!

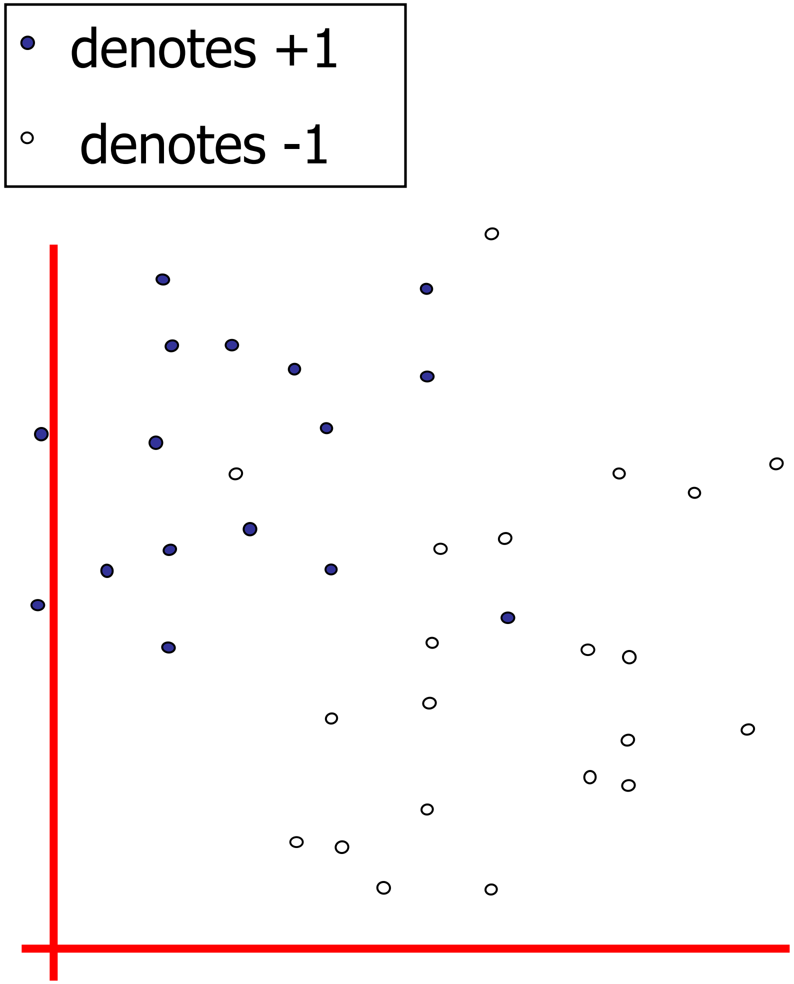
What should we do?

Idea 1.1:

Minimize

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

Tradeoff parameter



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

Uh-oh!

This is going to be a problem!

What should we do?

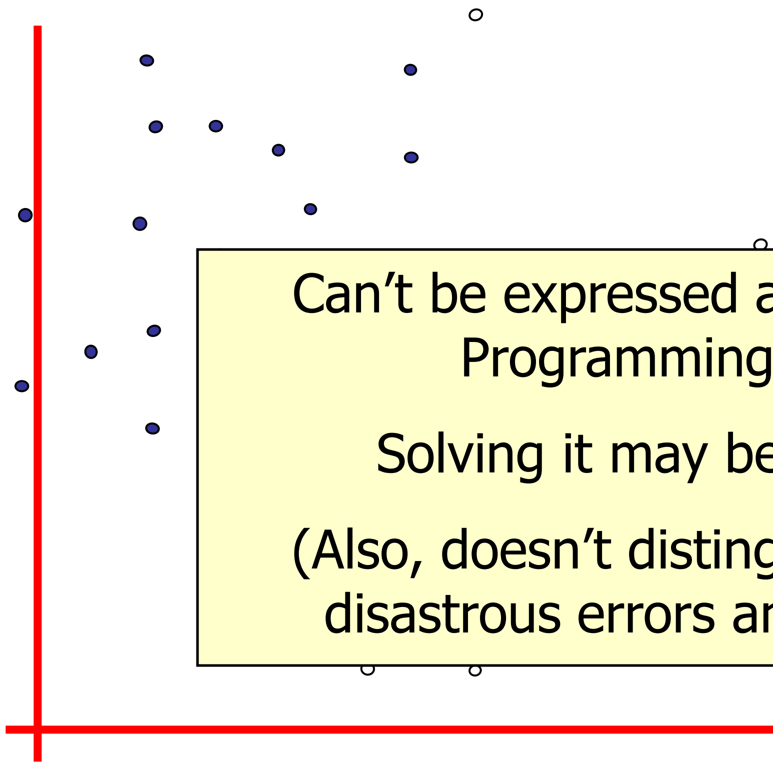
Idea 1.1:

Minimize

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

Tradeoff parameter

- denotes +1
- denotes -1



Can't be expressed as a Quadratic Programming problem.

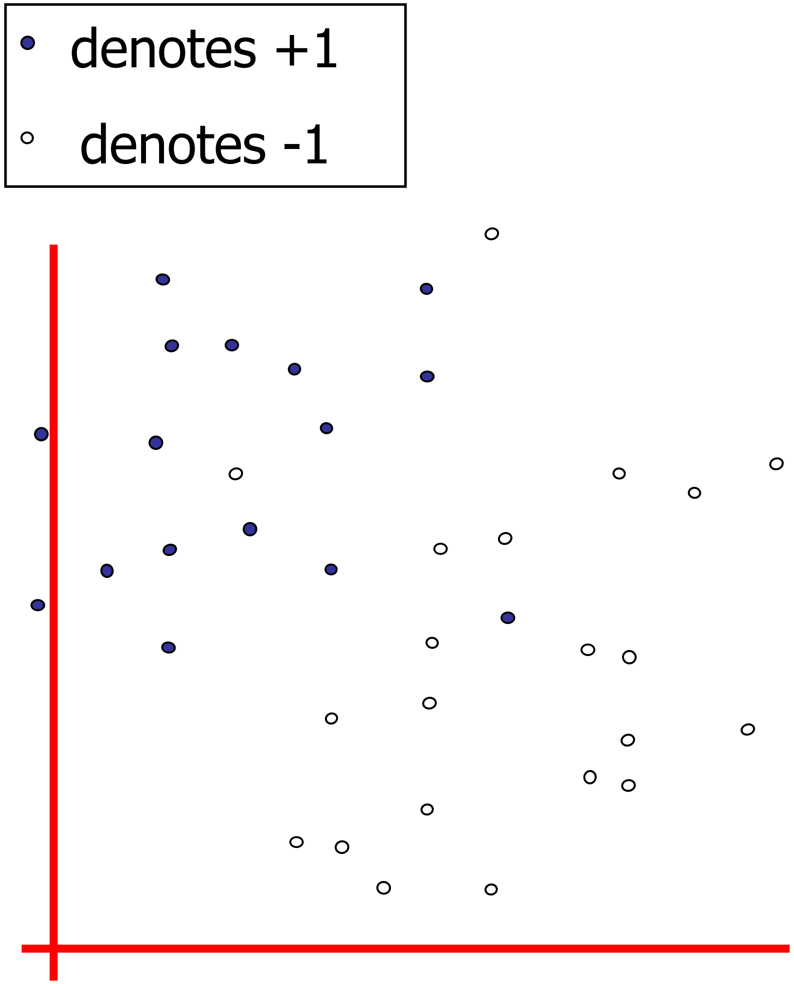
Solving it may be too slow.

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

erious
at'
ke
us reject this
you guess wh

So… any other ideas?

Uh-oh!



This is going to be a problem!

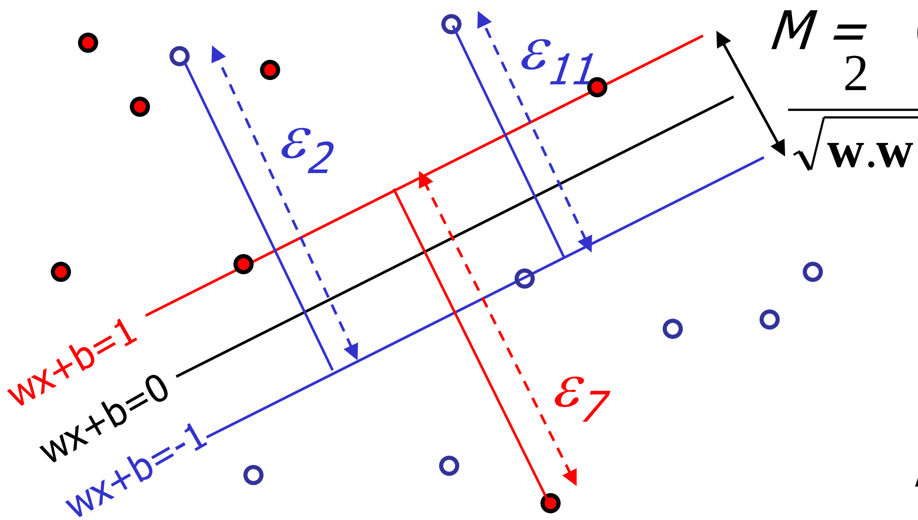
What should we do?

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

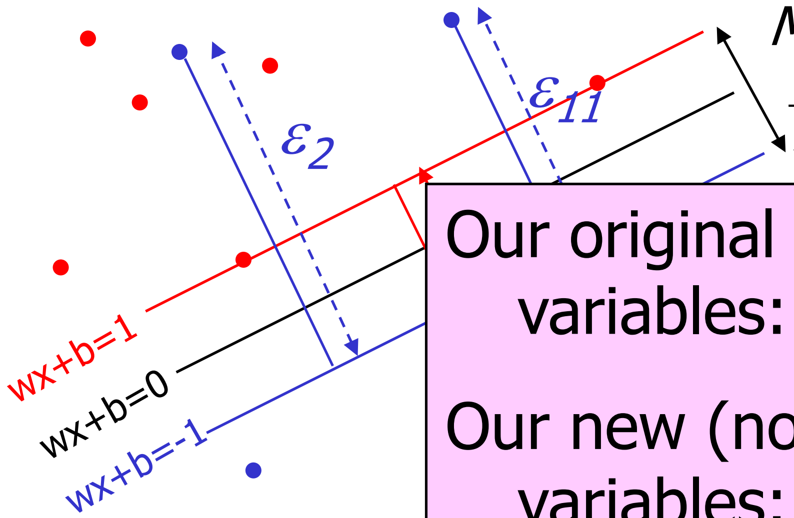
How many constraints will we have? R

What should they be?

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

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

Learning Maximum Margi … ise



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

Given gu … we can

- Compute sum o … distances

… dth

(x_k, y_k) where y_k …

m = # input dimensions

Our original (noiseless data) QP had $m+1$ variables: w_1, w_2, … w_m, and b.

Our new (noisy data) QP has $m+1+R$ variables: w_1, w_2, … w_m, b, ε_k, ε_1, … ε_R

R= # records

What should our quadratic optimization criterion be?

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

How many constraints … 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 $y_k = -1$

Support Vector Machine (SVM) for Noisy Data

$$\{\vec{w}^*, b^*\} = \min_{\vec{w},b} \sum_{i=1}^{d} w_i^2 + c\sum_{j=1}^{N} \varepsilon_j$$

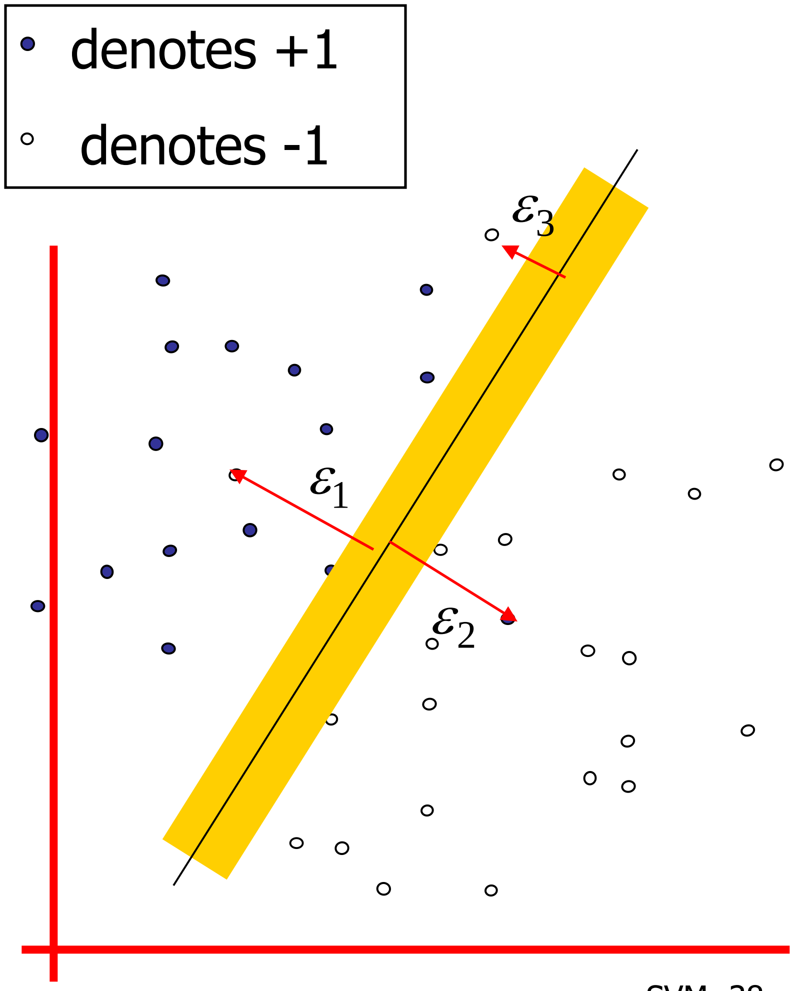
$$y_1\left(\vec{w}\cdot\vec{x}_1 + b\right) \ge 1 - \varepsilon_1$$

$$y_2\left(\vec{w}\cdot\vec{x}_2 + b\right) \ge 1 - \varepsilon_2$$

$$\ldots$$

$$y_N\left(\vec{w}\cdot\vec{x}_N + b\right) \ge 1 - \varepsilon_N$$

- Any problem with the above formulism?



Support Vector Machine (SVM) for Noisy Data

$$\{\vec{w}^*, b^*\} = \min_{\vec{w},b} \sum_{i=1}^{d} w_i^2 + c\sum_{j=1}^{N} \varepsilon_j$$

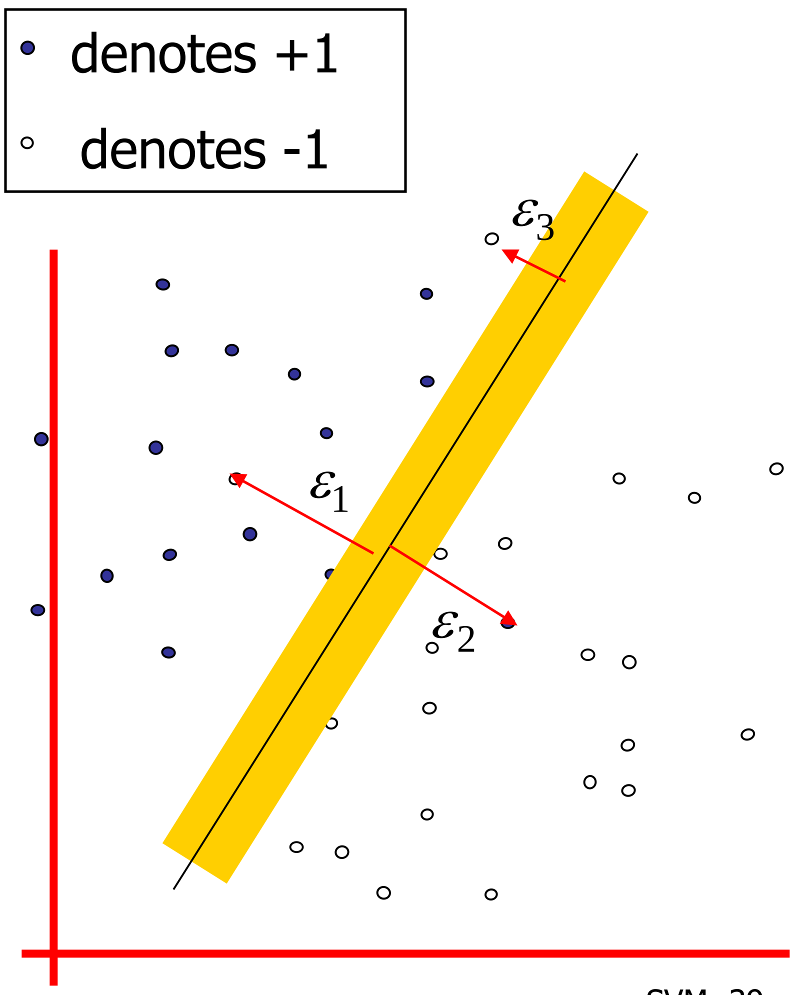
$$y_1\left(\vec{w}\cdot\vec{x}_1 + b\right) \geq 1 - \varepsilon_1, \varepsilon_1 \geq 0$$

$$y_2\left(\vec{w}\cdot\vec{x}_2 + b\right) \geq 1 - \varepsilon_2, \varepsilon_2 \geq 0$$

$$\dots$$

$$y_N\left(\vec{w}\cdot\vec{x}_N + b\right) \geq 1 - \varepsilon_N, \varepsilon_N \geq 0$$

- Balance the trade off between margin and classification errors



Support Vector Machine for Noisy Data

$$\{\vec{w}^*, b^*\} = \underset{\vec{w},b}{\operatorname{argmin}} \sum_i w_i^2 + c\sum_{j=1}^{N} \varepsilon_j$$

$$\left.\begin{array}{l}
y_1\left(\vec{w}\cdot\vec{x}_1 + b\right) \geq 1 - \varepsilon_1, \varepsilon_1 \geq 0 \\
y_2\left(\vec{w}\cdot\vec{x}_2 + b\right) \geq 1 - \varepsilon_2, \varepsilon_2 \geq 0 \\
.... \\
y_N\left(\vec{w}\cdot\vec{x}_N + b\right) \geq 1 - \varepsilon_N, \varepsilon_N \geq 0
\end{array}\right\} \text{inequality constraints}$$

How do we determine the appropriate value for c ?

Therefore, the problem of maximizing the margin is equivalent to

$$\begin{array}{ll} \text{Minimize} & J(w) = \frac{1}{2}\|w\|^2 \\ \text{Subject to} & y_i(w^T x_i + b) \geq 1 \quad \forall i \end{array}$$

– Notice that $J(w)$ is a quadratic function, which means that there exists a single global minimum and no local minima

To solve this problem, we will use classical Lagrangian optimization techniques

– We first present the Kuhn-Tucker Theorem, which provides an essential result for the interpretation of Support Vector Machines

(Kuhn-Tucker Theorem)

Given an optimization problem with convex domain $\Omega \subseteq R^N$

$$
\begin{array}{lll}
\text{Minimize} & f(z) & z \in \Omega \\
\text{Subject to} & g_i(z) \le 0 & i = 1..k \\
 & h_i(z) = 0 & i = 1..m
\end{array}
$$

- with $f \in C^1$ convex and g_i, h_i affine, necessary & sufficient conditions for a normal point z^* to be an optimum are the existence of α^*, β^* such that

$$
\begin{array}{ll}
\partial L(z^*, \alpha^*, \beta^*)/\partial z = 0 & \\
\partial L(z^*, \alpha^*, \beta^*)/\partial \beta = 0 & \\
\alpha_i^* g_i(z^*) = 0 & i = 1..k \\
g_i(z^*) \le 0 & i = 1..k \\
\alpha_i^* \ge 0 & i = 1..k
\end{array}
$$

where:

$$L(z, \alpha, \beta) = f(z) + \sum_{i=1}^{k} \alpha_i g_i(z) + \sum_{i=1}^{m} \beta_i h_i(z)$$

- $L(z, \alpha, \beta)$ is known as a *generalized Lagrangian function*
- The third condition is known as the Karush-Kuhn-Tucker (KKT) complementary condition
  - It implies that for active constraints $\alpha_i \ge 0$; and for inactive constraints $\alpha_i = 0$
  - As we will see in a minute, the KKT condition allows us to identify the training examples that define the largest margin hyperplane. These examples will be known as Support Vectors

[Cristianini and Shawe-Taylor, 2000]

The Lagrangian dual problem

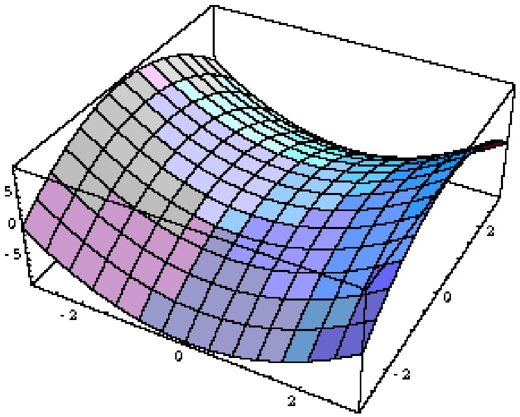
Constrained minimization of $J(w) = 1/2\|w\|^2$ is solved by introducing the Lagrangian

$$L_P(w,b,\alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{N} \alpha_i[y_i(w^T x_i + b) - 1]$$

– which yields an unconstrained optimization problem that is solved by:
  - minimizing L_P w.r.t. the primal variables w and b, <u>and</u>
  - maximizing L_P w.r.t. the dual variables $\alpha_i \geq 0$ (the Lagrange multipliers)

– Thus, the optimum is defined by a saddle point

– This is known as the <u>Lagrangian primal problem</u>



A saddle point

Solution

- To simplify the primal problem, we eliminate the primal variables (w, b) using the first Kuhn-Tucker condition $\partial J/\partial z = 0$
- Differentiating $L_P(w, b, \alpha)$ with respect to w and b, and setting to zero yields

$$\partial L_P(w,b,\alpha)/\partial w = 0 \qquad \Rightarrow w = \textstyle\sum_{i=1}^{N} \alpha_i y_i x_i$$

$$\partial L_P(w,b,\alpha)/\partial b = 0 \qquad \Rightarrow \textstyle\sum_{i=1}^{N} \alpha_i y_i = 0$$

- Expansion of L_P yields

$$L_P(w,b,\alpha) = \frac{1}{2} w^T w - \sum_{i=1}^{N} \alpha_i y_i w^T x_i - b \sum_{i=1}^{N} \alpha_i y_i + \sum_{i=1}^{N} \alpha_i$$

- Using the optimality condition $\partial J/\partial w = 0$, the first term in L_P can be expressed as

$$w^T w = w^T \sum_{i=1}^{N} \alpha_i y_i x_i = \sum_{i=1}^{N} \alpha_i y_i w^T x_i =$$

$$\sum_{i=1}^{N} \alpha_i y_i \left(\sum_{j=1}^{N} \alpha_j y_j x_j \right)^T x_i = \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

- The second term in L_P can be expressed in the same way
- The third term in L_P is zero by virtue of the optimality condition $\partial J/\partial b = 0$

[Haykin, 1999]

– Merging these expressions together we obtain

$$L_D(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

– Subject to the (simpler) constraints $\alpha_i \geq 0$ and $\sum_{i=1}^{N} \alpha_i y_i = 0$

– This is known as the **Lagrangian dual problem**

Comments

– We have transformed the problem of finding a saddle point for $L_P(w, b)$ into the easier one of maximizing $L_D(\alpha)$
  - Notice that $L_D(\alpha)$ depends on the Lagrange multipliers α, not on (w, b)

– The primal problem scales with dimensionality (w has one coefficient for each dimension), whereas the dual problem scales with the amount of training data (there is one Lagrange multiplier per example)

– Moreover, in $L_D(\alpha)$ training data appears only as dot products $x_i^T x_j$
  - As we will see in the next lecture, this property can be cleverly exploited to perform the classification in a higher (e.g., infinite) dimensional space

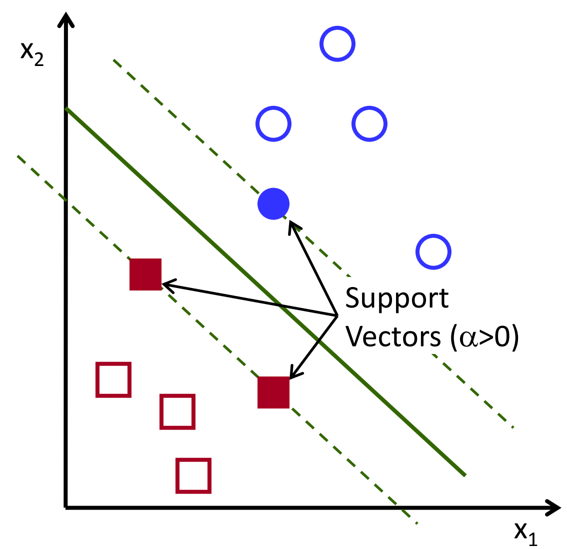
Support Vectors

The KKT complementary condition states that, for every point in the training set, the following equality must hold

$$\alpha_i[y_i(w^T x_i + b) - 1] = 0 \quad \forall i = 1..N$$

- Therefore, $\forall x$, either $\alpha_i = 0$ or $y_i(w^T x_i + b - 1) = 0$ must hold
  - Those points for which $\alpha_i > 0$ must then lie on one of the two hyperplanes that define the largest margin (the term $y_i(w^T x_i + b - 1)$ becomes zero only at these hyperplanes)
  - These points are known as the **Support Vectors**
  - All the other points must have $\alpha_i = 0$
- Note that only the SVs contribute to defining the optimal hyperplane

$$\frac{\partial J(w,b,\alpha)}{\partial w} = 0 \;\Rightarrow w = \sum_{i=1}^{N} \alpha_i y_i x_i$$



- NOTE: the bias term b is found from the KKT complementary condition on the support vectors
- Therefore, the complete dataset could be replaced by only the support vectors, and the separating hyperplane would be the same

The Dual Form of QP

Maximize $\sum_{k=1}^{R} \alpha_k - \frac{1}{2} \sum_{k=1}^{R} \sum_{l=1}^{R} \alpha_k \alpha_l Q_{kl}$ where $Q_{kl} = y_k y_l (\mathbf{x}_k \cdot \mathbf{x}_l)$

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

Then define:

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

An Equivalent QP

Maximize $\sum_{k=1}^{R} \alpha_k - \frac{1}{2} \sum_{k=1}^{R} \sum_{l=1}^{R} \alpha_k \alpha_l Q_{kl}$ where $Q_{kl} = y_k y_l (\mathbf{x}_k . \mathbf{x}_l)$

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

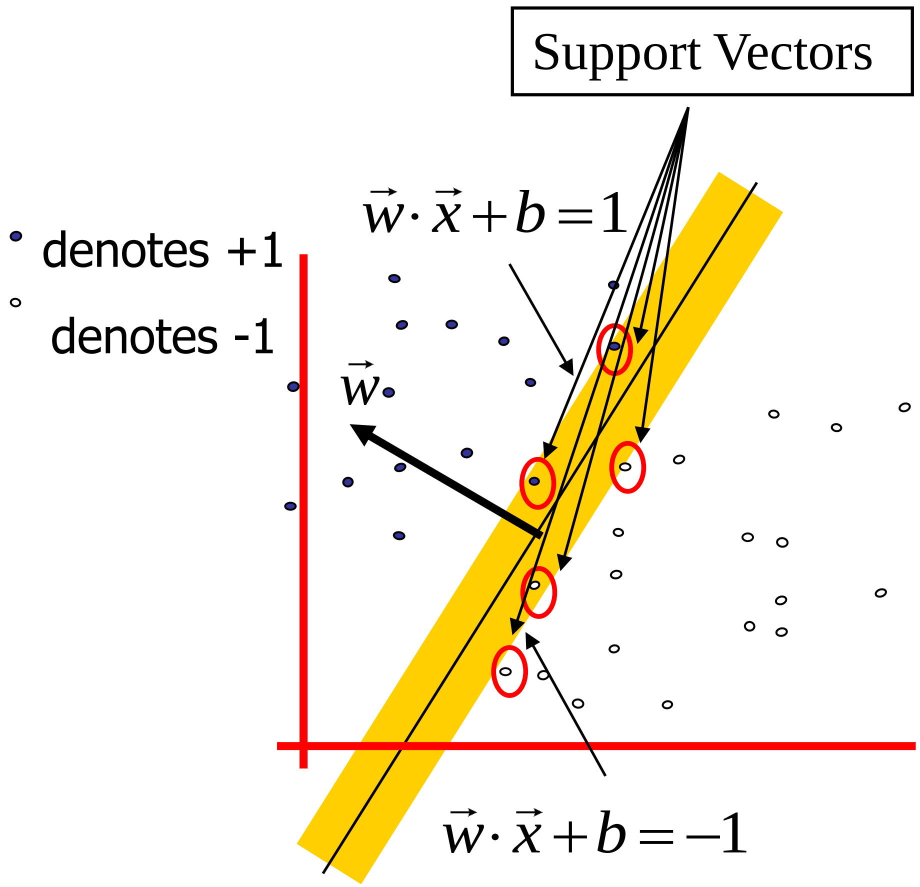
Datapoints with $\alpha_k > 0$ will be the support vectors

Then define:

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

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

Support Vectors



$$\forall i : \alpha_i \left(y_i \left(\vec{w}\cdot\vec{x}_i + b \right) - \left(1 - \varepsilon_i \right) \right) = 0$$

α_i = 0 for non-support vectors

$\alpha_i \neq$ 0 for support vectors

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

Decision boundary is determined only by those support vectors !

The Dual Form of QP

Maximize $\sum_{k=1}^{R} \alpha_k - \frac{1}{2} \sum_{k=1}^{R} \sum_{l=1}^{R} \alpha_k \alpha_l Q_{kl}$ where $Q_{kl} = y_k y_l (\mathbf{x}_k \cdot \mathbf{x}_l)$

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

Then define:

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

Then classify with:

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

How to determine *b* ?

An Equivalent QP: Determine *b*

$$\{\vec{w}^*, b^*\} = \underset{\vec{w},b}{\operatorname{argmin}} \sum_i w_i^2 + c\sum_{j=1}^{N} \varepsilon_j$$

$$y_1\left(\vec{w}\cdot\vec{x}_1 + b\right) \geq 1 - \varepsilon_1, \varepsilon_1 \geq 0$$

$$y_2\left(\vec{w}\cdot\vec{x}_2 + b\right) \geq 1 - \varepsilon_2, \varepsilon_2 \geq 0$$

....

$$y_N\left(\vec{w}\cdot\vec{x}_N + b\right) \geq 1 - \varepsilon_N, \varepsilon_N \geq 0$$

Fix **w**

$$b^* = \underset{b,\{\varepsilon_i\}_{i=1}^{N}}{\operatorname{argmin}} \sum_{j=1}^{N} \varepsilon_j$$

$$y_1\left(\vec{w}\cdot\vec{x}_1 + b\right) \geq 1 - \varepsilon_1, \varepsilon_1 \geq 0$$

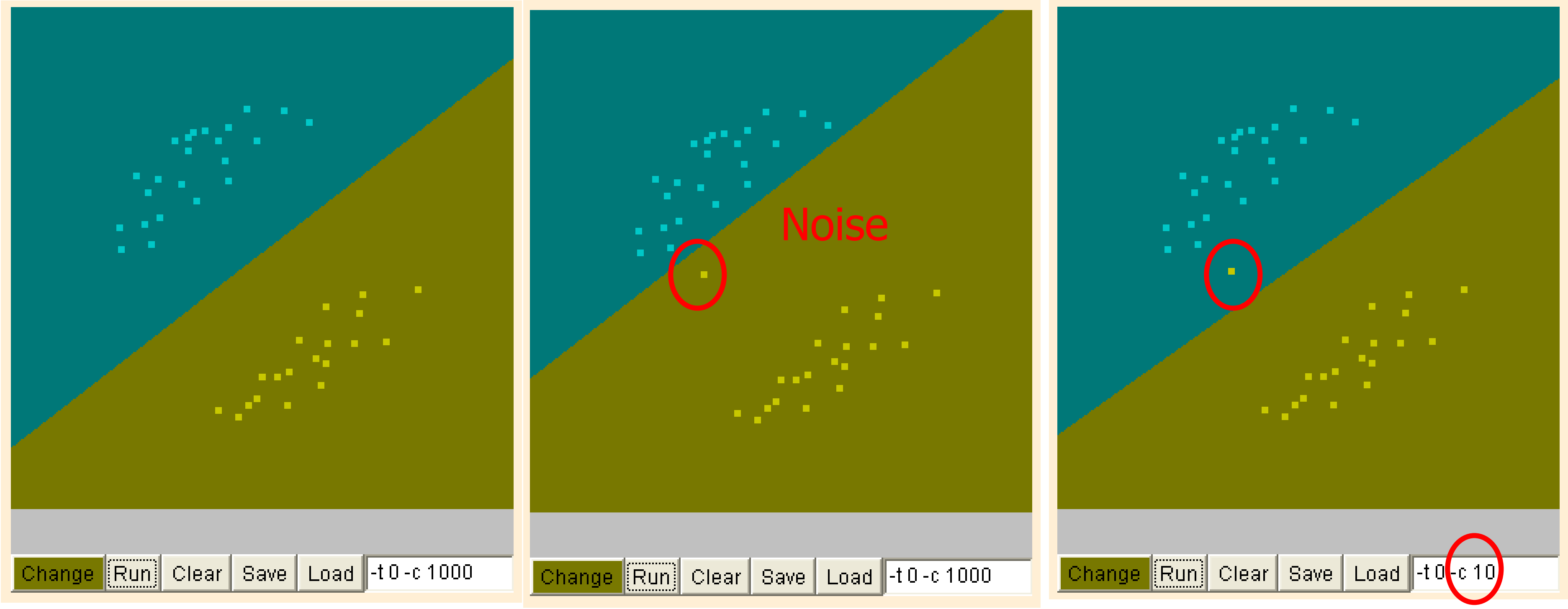
$$y_2\left(\vec{w}\cdot\vec{x}_2 + b\right) \geq 1 - \varepsilon_2, \varepsilon_2 \geq 0$$

....

$$y_N\left(\vec{w}\cdot\vec{x}_N + b\right) \geq 1 - \varepsilon_N, \varepsilon_N \geq 0$$

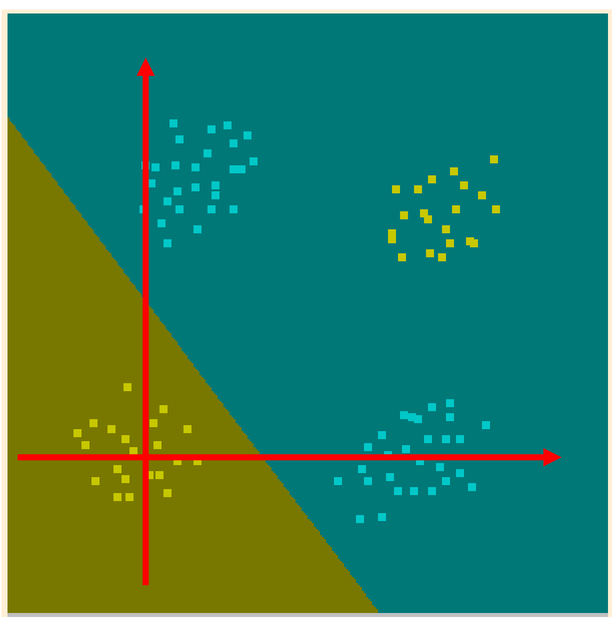
A linear programming problem !

- Parameter c is used to control the fitness



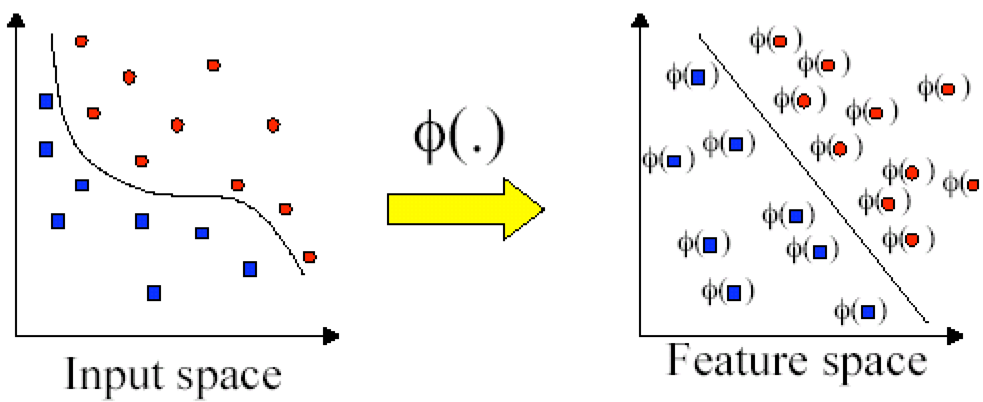
Feature Transformation ?

- The problem is non-linear
- Find some trick to transform the input
- Linear separable after Feature Transformation
- What Features should we use ?



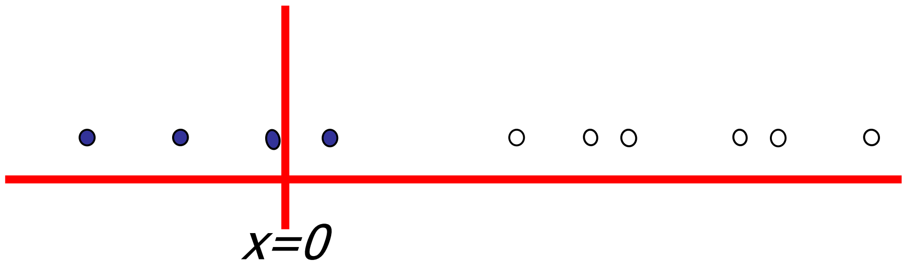
XOR Problem

Basic Idea :



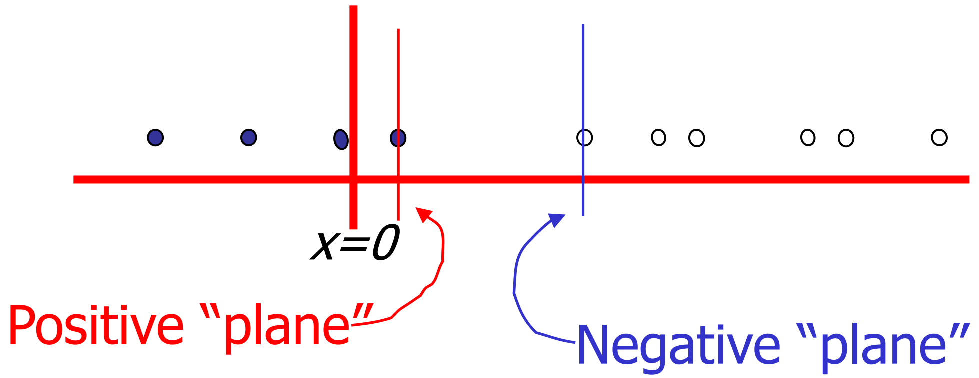
Suppose we're in 1-dimension

What would SVMs do with this data?

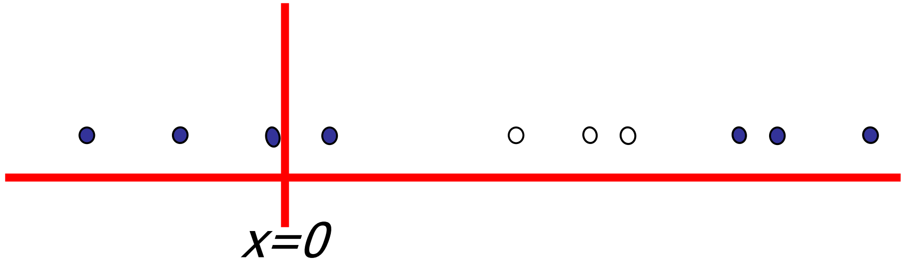


Suppose we're in 1-dimension

Not a big surprise



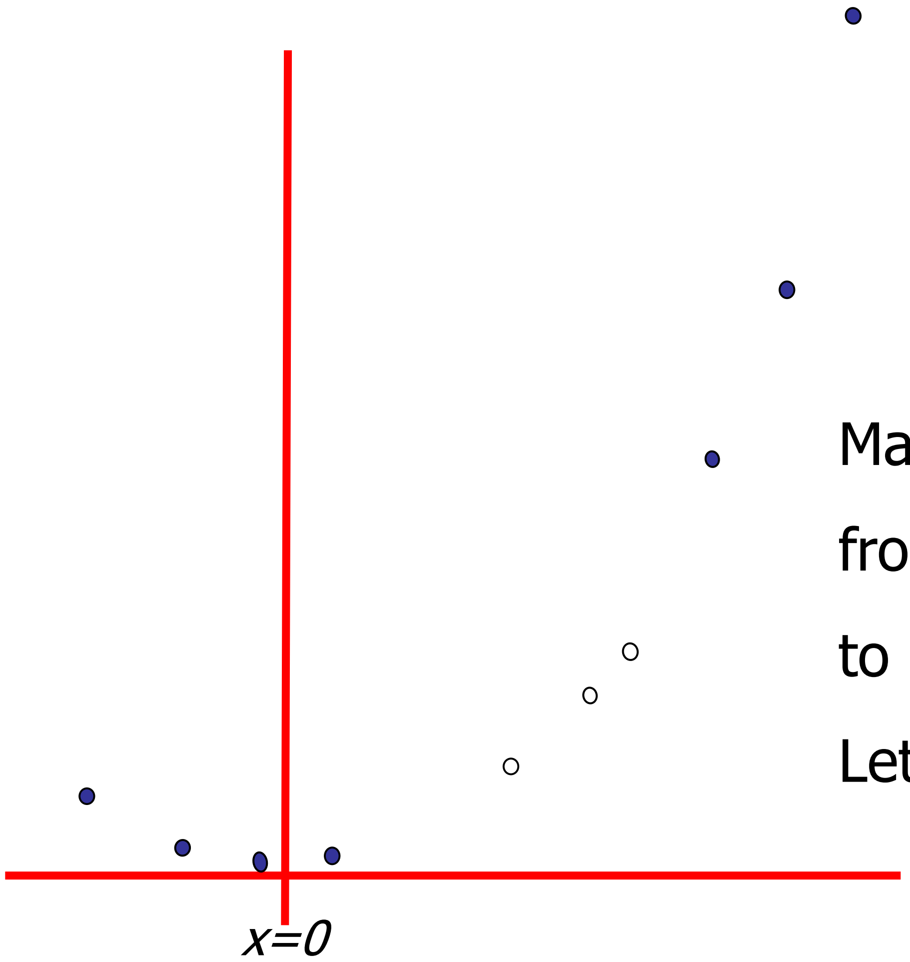
Harder 1-dimensional dataset



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

What can be done about this?

Harder 1-dimensional dataset



Map the data

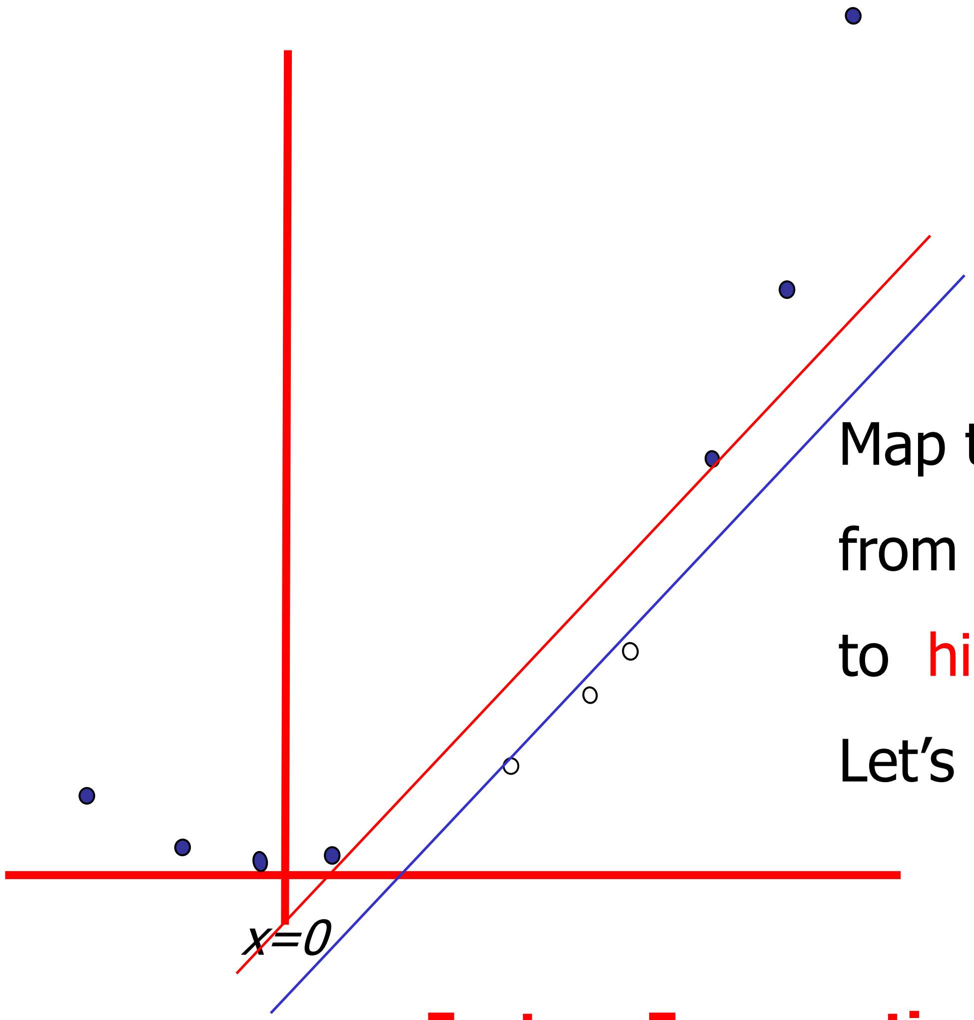
from low-dimensional space

to high-dimensional space

Let's permit them here too

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

Harder 1-dimensional dataset



Map the data

from low-dimensional space

to high-dimensional space

Let's permit them here too

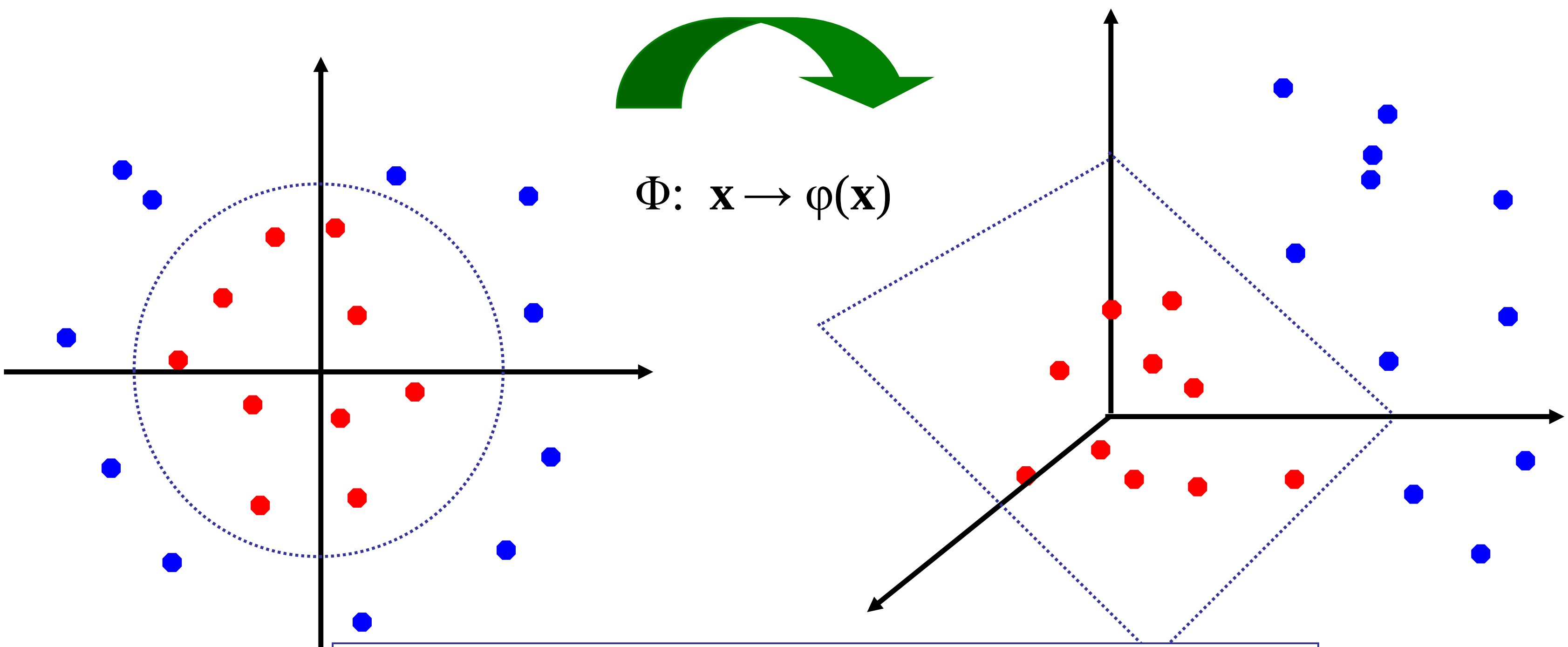
$$\mathbf{z}_k = (x_k, x_k^2)$$

Feature Enumeration

$$x_k \xrightarrow{transform} \Phi(x_k) = z_k$$

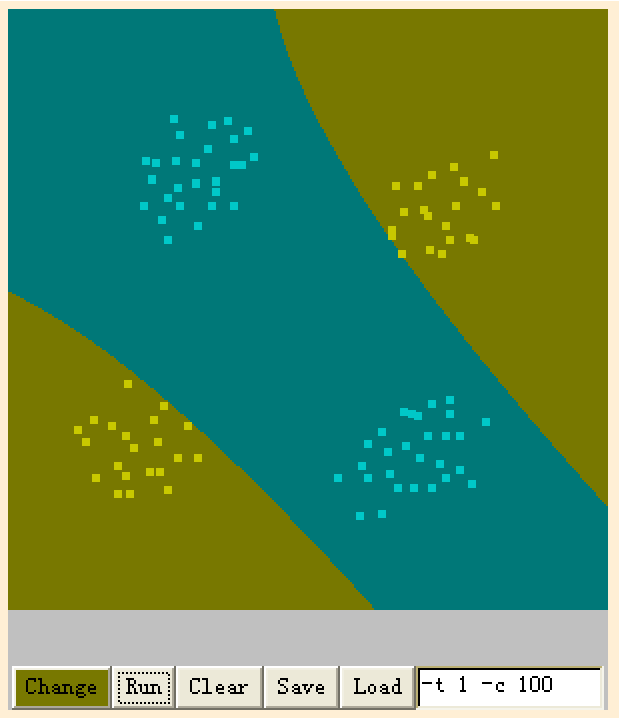
Non-linear SVMs: Feature spaces

- General idea: the original input space can always be mapped to some higher-dimensional feature space where the training set is separable:



$$\Phi : \mathbb{R}^2 \to \mathbb{R}^3$$
$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2}\, x_1 x_2, x_2^2)$$

- Polynomial features for the XOR problem



Kernel methods

Let's now see how to put together all these concepts

- Assume that our original feature vector x lives in a space R^D
- We are interested in non-linearly projecting x onto a higher dimensional implicit space $\varphi(x) \in R^{D1}$ $(D1 > D)$ where classes have a better chance of being linearly separable
  - Notice that we are not guaranteeing linear separability, we are only saying that we have a better chance because of Cover's theorem
- The separating hyperplane in R^{D1} will be defined by

$$\sum_{j=1}^{D1} w_j \varphi_j(x) + b = 0$$

- To eliminate the bias term b, let's augment the feature vector in the implicit space with a constant dimension $\varphi_0(x) = 1$
- Using vector notation, the resulting hyperplane becomes

$$w^T \varphi(x) = 0$$

- From our previous results, the optimal (maximum margin) hyperplane in the implicit space is given by

$$w = \sum_{i=1}^{N} \alpha_i y_i \varphi(x_i)$$

– Merging this optimal weight vector with the hyperplane equation

$$w^T \varphi(x) = 0$$

$$\Rightarrow \left(\sum_{i=1}^{N} \alpha_i y_i \varphi(x_i) \right)^T \varphi(x) = 0$$

$$\Rightarrow \sum_{i=1}^{N} \alpha_i y_i \varphi(x_i)^T \varphi(x) = 0$$

– and, since $\varphi^T(x_i)\varphi\left(x_j\right) = K\left(x_i, x_j\right)$, the optimal hyperplane becomes

$$\textstyle\sum_{i=1}^{N} \alpha_i y_i K(x_i, x) = 0$$

– Therefore, classification of an unknown example x is performed by computing the weighted sum of the kernel function with respect to the support vectors x_i (remember that only the support vectors have non-zero dual variables α_i)

How do we compute dual variables α_i in the implicit space?

– Very simple: we use the same optimization problem as before, and replace the dot product $\varphi^T(x_i)\varphi(x_j)$ with the kernel $K(x_i, x_j)$

– The Lagrangian dual problem for the non-linear SVM is simply

$$L_D(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K\left(x_i^T, x_j\right)$$

– subject to the constraints

$$\begin{cases} \sum_{i=1}^{N} \alpha_i y_i = 0 \\ 0 \le \alpha_i \le C \quad i = 1 \dots N \end{cases}$$

How do we select the implicit mapping $\varphi(x)$?

- As we saw in the example a few slides back, we will normally select a kernel function first, and then determine the implicit mapping $\varphi(x)$ that it corresponds to

Then, how do we select the kernel function $K(x_i, x_j)$?

- We must select a kernel for which an implicit mapping exists, this is, a kernel that can be expressed as the dot-product of two vectors

For which kernels $K(x_i, x_j)$ does there exist an implicit mapping $\varphi(x)$?

- The answer is given by Mercer's Condition

[Burges, 1998; Kaykin, 1999]

Mercer's Condition

Let $K(x, x')$ be a continuous symmetric kernel that is defined in the closed interval $a \leq x \leq b$

- The kernel can be expanded in the series:

$$K(x, x') = \sum_{i=1}^{\infty} \lambda_i \varphi_i(x) \varphi_i(x')$$

  - Strictly speaking, the space where $\varphi(x)$ resides is a Hilbert space, a "generalization" of an Euclidean space where the inner product can be any inner product, not just the scalar dot product [Burges, 1998]
  - With positive coefficients $\lambda_i > 0 \ \forall i$
- For this expansion to be valid and for it to converge absolutely and uniformly, it is necessary and sufficient that the condition

$$\int_a^b \int_a^b K(x, x') \psi(x) \psi(x') dx dx' \geq 0$$

- holds for all $\psi(\cdot)$ for which $\int_a^b \psi^2(x) dx \leq \infty$
  - The functions $\varphi_i(x)$ are called eigenfunctions of the expansion, and the numbers λ_i are the eigenvalues. The fact that all of the eigenvalues are positive means that the kernel is positive definite
- Notice that the dimensionality of the implicit space can be infinitely large
- Mercer's Condition only tells us whether a kernel is actually an inner-product kernel, but it does not tell us how to construct the functions $\varphi_i(x)$ for the expansion

[Kaykin, 1999]

Which kernels meet Mercer's condition?

- Polynomial kernels

$$K(x, x') = (x^T x' + 1)^p$$

  - The degree of the polynomial is a user-specified parameter

- Radial basis function kernels

$$K(x, x') = \exp\left(-\frac{1}{2\sigma^2}\|x - x'\|^2\right)$$

  - The width σ is a user-specified parameter, but the number of radial basis functions and their centers are determined automatically by the number of support vectors and their values

- Two-layer perceptron

$$K(x, x') = \tanh(\beta_0 x^T x' + \beta_1)$$

  - The number of hidden neurons and their weight vectors are determined automatically by the number of support vectors and their values, respectively. The H-O weights are the Lagrange multipliers α_i
  - However, this kernel will only meet Mercer's condition for certain values of β_0 and β_1

[Burges, 1998; Kaykin, 1999]

Efficiency Problem in Computing Feature

- Feature space Mapping

Preprocess the data with

$$\Phi : \mathcal{X} \to \mathcal{H}$$
$$x \mapsto \Phi(x),$$

where $\mathcal{H}$ is a dot product space, and learn the mapping from $\Phi(x)$ to y.

This use of kernel function to avoid carrying out $\Phi(x)$ explicitly is known as the kernel trick

- Example: all 2 degree Monomials

$$\Phi : \mathbb{R}^2 \to \mathbb{R}^3$$
$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2}\, x_1 x_2, x_2^2)$$

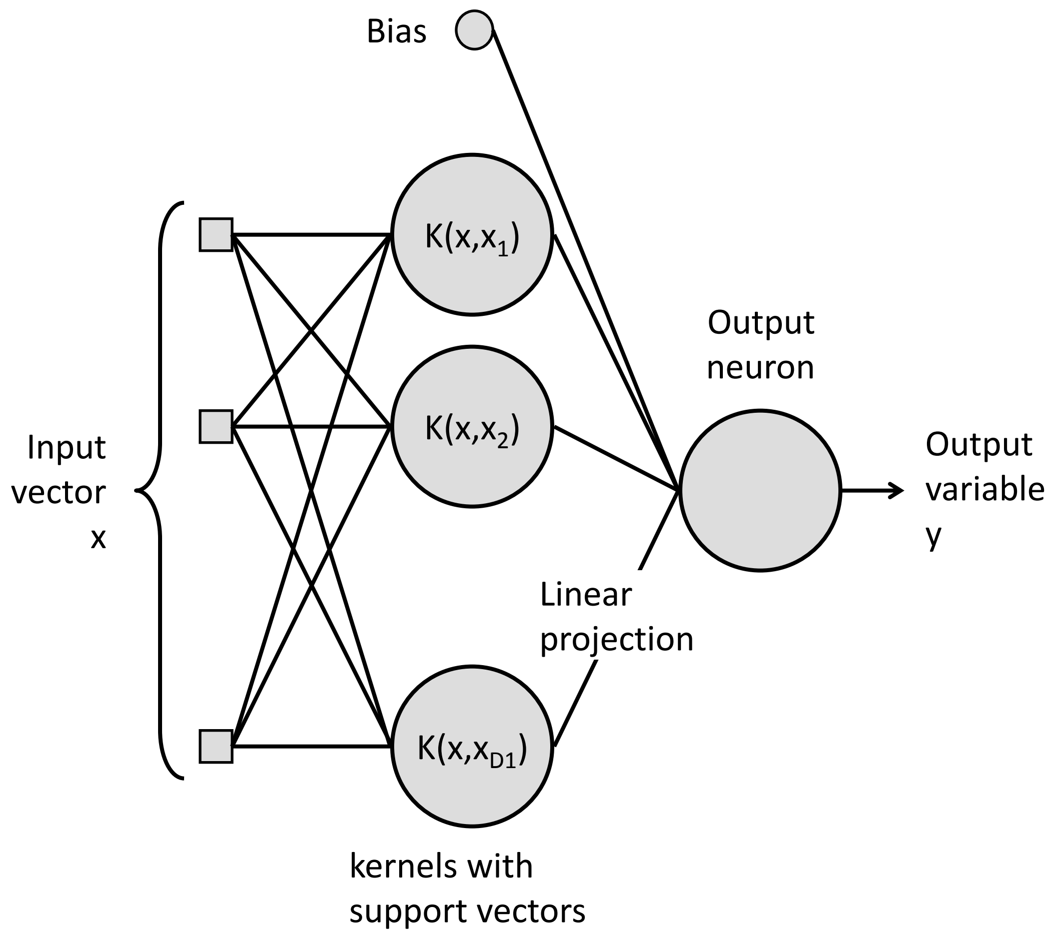
$$\langle \Phi(x), \Phi(x') \rangle = (x_1^2, \sqrt{2}\, x_1 x_2, x_2^2)(x_1'^2, \sqrt{2}\, x_1' x_2', x_2'^2)^\top$$ ← 9 Multipllication

kernel trick $= \langle x, x' \rangle^2$ ← 3 Multipllication

$$=: k(x, x')$$

$\longrightarrow$ the dot product in $\mathcal{H}$ can be computed in $\mathbb{R}^2$

Architecture of an SVM



[Kaykin, 1999]

Techniques for Constructing New Kernels.

Given valid kernels $k_1(\mathbf{x}, \mathbf{x}')$ and $k_2(\mathbf{x}, \mathbf{x}')$, the following new kernels will also be valid:

$$
\begin{aligned}
k(\mathbf{x}, \mathbf{x}') &= ck_1(\mathbf{x}, \mathbf{x}') \\
k(\mathbf{x}, \mathbf{x}') &= f(\mathbf{x})k_1(\mathbf{x}, \mathbf{x}')f(\mathbf{x}') \\
k(\mathbf{x}, \mathbf{x}') &= q\left(k_1(\mathbf{x}, \mathbf{x}')\right) \\
k(\mathbf{x}, \mathbf{x}') &= \exp\left(k_1(\mathbf{x}, \mathbf{x}')\right) \\
k(\mathbf{x}, \mathbf{x}') &= k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}') \\
k(\mathbf{x}, \mathbf{x}') &= k_1(\mathbf{x}, \mathbf{x}')k_2(\mathbf{x}, \mathbf{x}') \\
k(\mathbf{x}, \mathbf{x}') &= k_3\left(\phi(\mathbf{x}), \phi(\mathbf{x}')\right) \\
k(\mathbf{x}, \mathbf{x}') &= \mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x}' \\
k(\mathbf{x}, \mathbf{x}') &= k_a(\mathbf{x}_a, \mathbf{x}'_a) + k_b(\mathbf{x}_b, \mathbf{x}'_b) \\
k(\mathbf{x}, \mathbf{x}') &= k_a(\mathbf{x}_a, \mathbf{x}'_a)k_b(\mathbf{x}_b, \mathbf{x}'_b)
\end{aligned}
$$

where $c > 0$ is a constant, $f(\cdot)$ is any function, $q(\cdot)$ is a polynomial with nonnegative coefficients, $\phi(\mathbf{x})$ is a function from $\mathbf{x}$ to $\mathbb{R}^M$, $k_3(\cdot, \cdot)$ is a valid kernel in $\mathbb{R}^M$, $\mathbf{A}$ is a symmetric positive semidefinite matrix, $\mathbf{x}_a$ and $\mathbf{x}_b$ are variables (not necessarily disjoint) with $\mathbf{x} = (\mathbf{x}_a, \mathbf{x}_b)$, and k_a and k_b are valid kernel functions over their respective spaces.

Numerical example

To illustrate the operation of a non-linear SVM we will solve the classical XOR problem

– Dataset
  - Class 1: $x_1 = (-1,-1), \quad x_4 = (+1,+1)$
  - Class 2: $x_2 = (-1,+1), \quad x_3 = (+1,-1)$

– Kernel function
  - Polynomial of order 2: $K(x,x') = (x^T x' + 1)^2$

Solution

– The implicit mapping can be shown to be 5-dimensional

$$\varphi(x) = \begin{bmatrix} 1 & \sqrt{2}x_{i,1} & \sqrt{2}x_{i,2} & \sqrt{2}x_{i,1}x_{i,2} & x_{i,1}^2 & x_{i,2}^2 \end{bmatrix}^T$$

– To achieve linear separability, we will use $C = \infty$

– The objective function for the dual problem becomes

$$L_D(\alpha) = \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 - \frac{1}{2}\sum_{i=1}^{4}\sum_{j=1}^{4}\alpha_i\alpha_j y_i y_j k_{ij}$$

  - subject to the constraints $\begin{cases} \sum_{i=1}^{N} \alpha_i y_i = 0 \\ 0 \le \alpha_i \le C \quad i = 1 \ldots N \end{cases}$

[Cherkassky and Mulier, 1998; Haykin, 1999]

$$K(x_1, x_1) = (\begin{bmatrix} -1 \\ -1 \end{bmatrix}[-1\ -1] + 1)^2 = (2 + 1)^2 = 9$$

$$K(x_1, x_2) = (\begin{bmatrix} -1 \\ -1 \end{bmatrix}[-1\ 1] + 1)^2 = (0 + 1)^2 = 1$$

$$K(x_1, x_3) = \dots\dots$$

....

- where the inner product is represented as a 4×4 K matrix

$$K = \begin{bmatrix} 9 & 1 & 1 & 1 \\ 1 & 9 & 1 & 1 \\ 1 & 1 & 9 & 1 \\ 1 & 1 & 1 & 9 \end{bmatrix}$$

- Optimizing with respect to the Lagrange multipliers leads to the following system of equations

$$\begin{gathered} 9\alpha_1 - \alpha_2 - \alpha_3 + \alpha_4 = 1 \\ -\alpha_1 + 9\alpha_2 + \alpha_3 - \alpha_4 = 1 \\ -\alpha_1 + \alpha_2 + 9\alpha_3 - \alpha_4 = 1 \\ \alpha_1 - \alpha_2 - \alpha_3 + 9\alpha_4 = 1 \end{gathered}$$

- whose solution is $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = 0.125$

- Thus, all data points are support vectors in this case

- For this simple problem, it is worthwhile to write the decision surface in terms of the polynomial expansion

$$w = \sum_{i=1}^{4} \alpha_i y_i \varphi(x_i) = \begin{bmatrix} 0 & 0 & 0 & 1/\sqrt{2} & 0 & 0 \end{bmatrix}^T$$

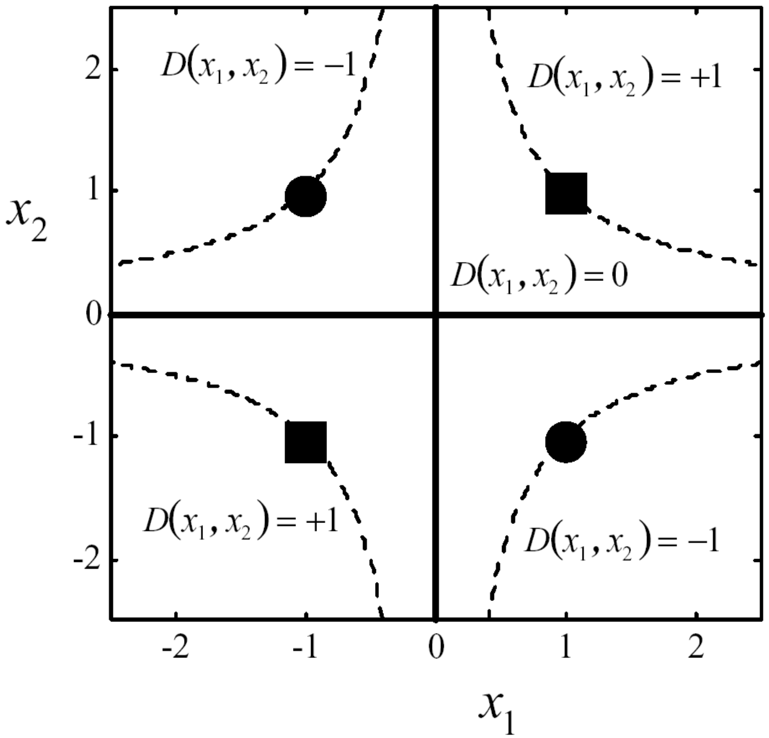
- resulting in the intuitive non-linear discriminant function

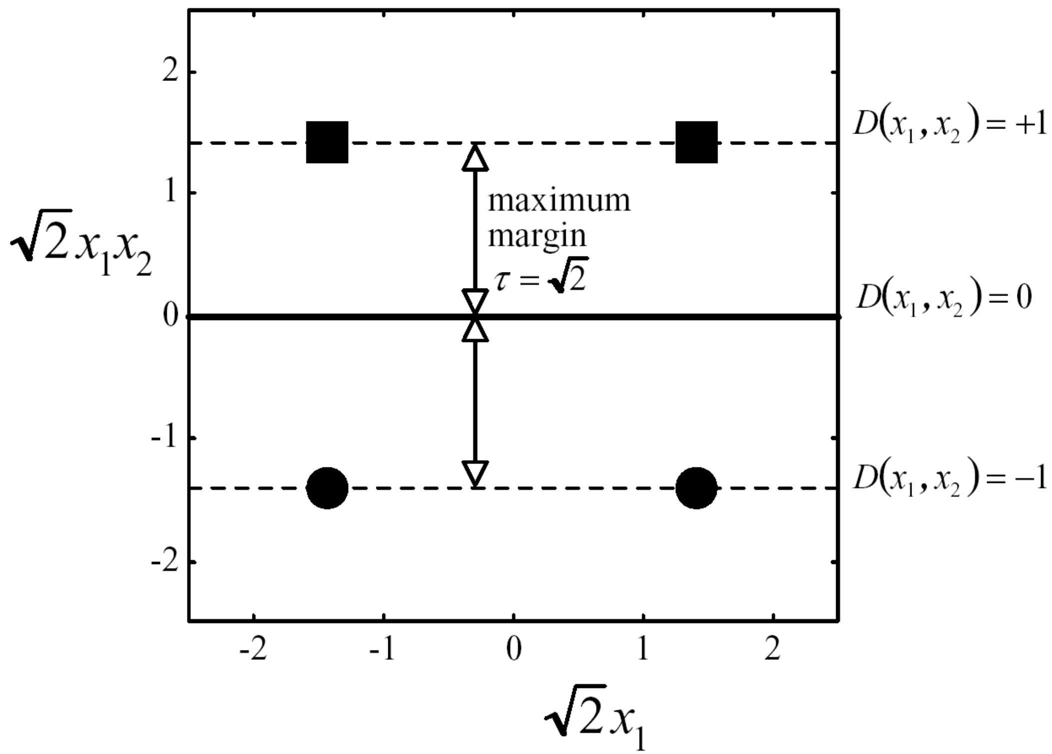
$$g(x) = \sum_{i=1}^{6} w_i \varphi_i(x) = x_1 x_2$$

- which has zero empirical error on the XOR training set

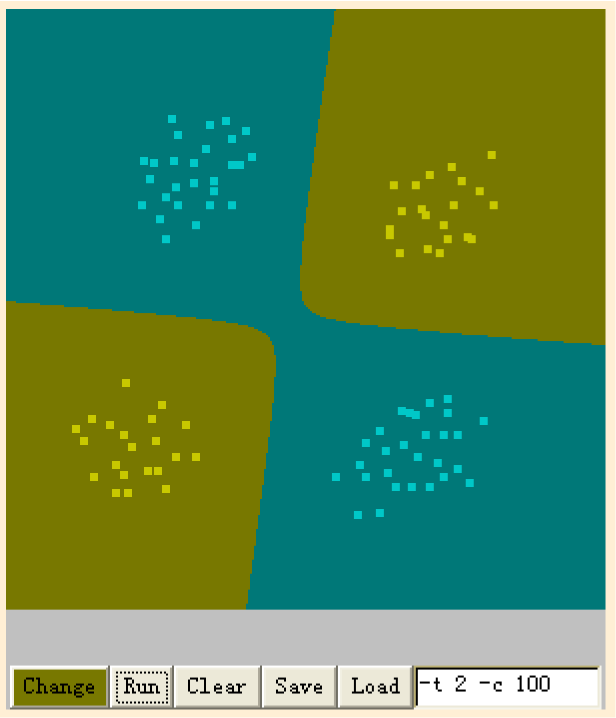
Decision function defined by the SVM

– Notice that the decision boundaries are non-linear in the original space R^2, but linear in the implicit space R^6



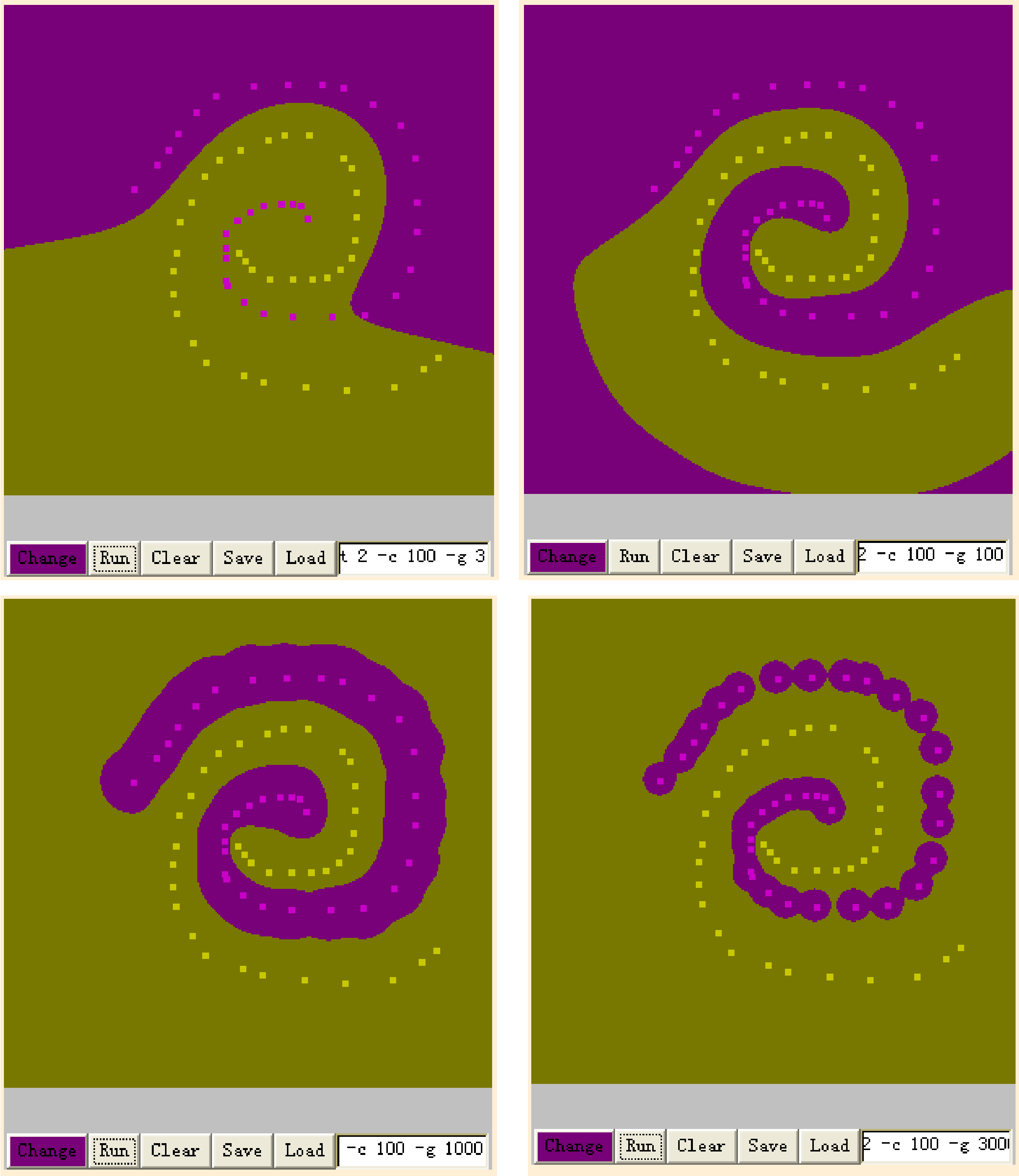


- “Radius basis functions” for the XOR problem



- Could solve complicated Non-Linear Problems
- γ and c control the complexity of decision boundary

$$\gamma = \frac{1}{\sigma^2}$$



Discussion

Advantages of SVMs

- There are no local minima, because the solution is a QP problem
- The optimal solution can be found in polynomial time
- Few model parameters to select: the penalty term C, the kernel function and parameters (e.g., spread σ in the case of RBF kernels)
- Final results are stable and repeatable (e.g., no random initial weights)
- SVM solution is sparse; it only involves the support vectors
- SVMs represent a general methodology for many PR problems: classification, regression, feature extraction, clustering, novelty detection, etc.
- SVMs rely on elegant and principled learning methods
- SVMs provide a method to control complexity independently of dimensionality
- SVMs have been shown (theoretically and empirically) to have excellent generalization capabilities

[Bennett and Campbell, 2000]

Challenges

- Do SVMs always perform best? Can they beat a hand-crafted solution for a particular problem?
- Do SVMs eliminate the model selection problem? Can the kernel functions be selected in a principled manner? SVMs still require selection of a few parameters, typically through cross-validation
- How does one incorporate domain knowledge? Currently this is performed through the selection of the kernel and the introduction of "artificial" examples
- How interpretable are the results provided by an SVM?
- What is the optimal data representation for SVM? What is the effect of feature weighting? How does an SVM handle categorical or missing features?