IT/PC/B/T/411

Machine Learning

Support Vector Machines and its Applications

Dr. Pawan Kumar Singh
Department of Information Technology
Jadavpur University
pawankrsingh.cse@gmail.com
+91-6291555693

Overview

- Artificial Neural Networks vs. SVM
- Intro. to Support Vector Machines (SVM)
- Properties of SVM
- Applications
  - Text Categorization
- References

ANN vs SVM

- The development of ANNs followed a heuristic path, with applications and extensive experimentation preceding theory.
- In contrast, the development of SVMs involved sound theory first, then implementation and experiments.
- A significant advantage of SVMs is that whilst ANNs can suffer from multiple local minima, the solution to an SVM is global and unique.
- Two more advantages of SVMs are that that have a simple geometric interpretation and give a sparse solution.
- Unlike ANNs, the computational complexity of SVMs does not depend on the dimensionality of the input space.
- The reason that SVMs often outperform ANNs in practice is that they deal with the biggest problem with ANNs, SVMs are less prone to overfitting.

Researchers' Opinions

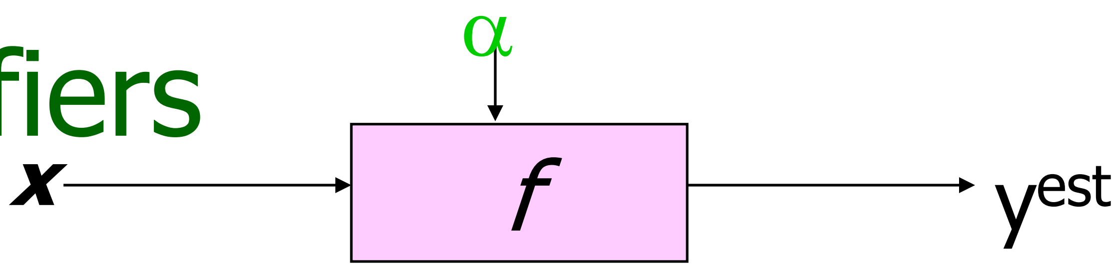
- "They differ radically from comparable approaches such as neural networks: SVM training always finds a global minimum, and their simple geometric interpretation provides fertile ground for further investigation."
  Burgess (1998)
- "Unlike conventional statistical and neural network methods, the SVM approach does not attempt to control model complexity by keeping the number of features small.
- "In contrast to neural networks SVMs automatically select their model size (by selecting the Support vectors)."
  Rychetsky (2001)

Support Vector Machine (SVM)

- A classifier derived from statistical learning theory by Vapnik, et al. in 1992
- SVM became famous when, using images as input, it gave accuracy comparable to neural-network with hand-designed features in a handwriting recognition task
- Currently, SVM is widely used in object detection & recognition, content-based image retrieval, text recognition, biometrics, speech recognition, etc.

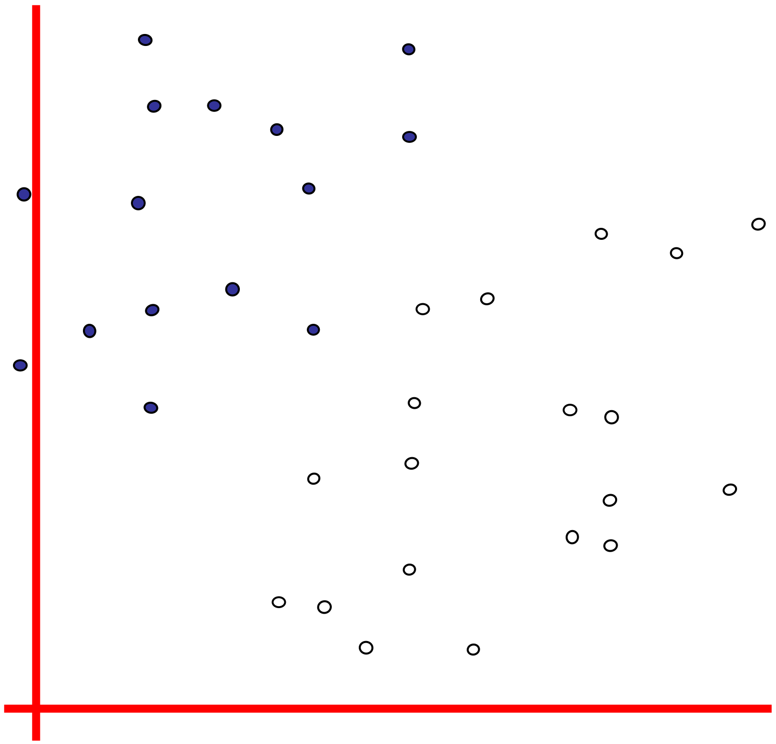
V. Vapnik

Linear Classifiers



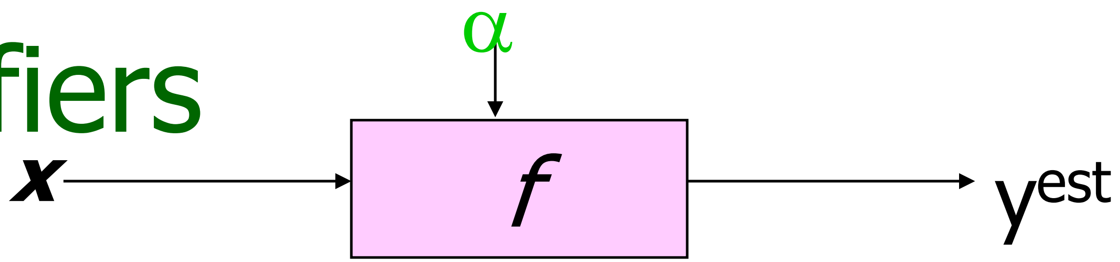
$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} - b)$

- denotes +1
- denotes -1



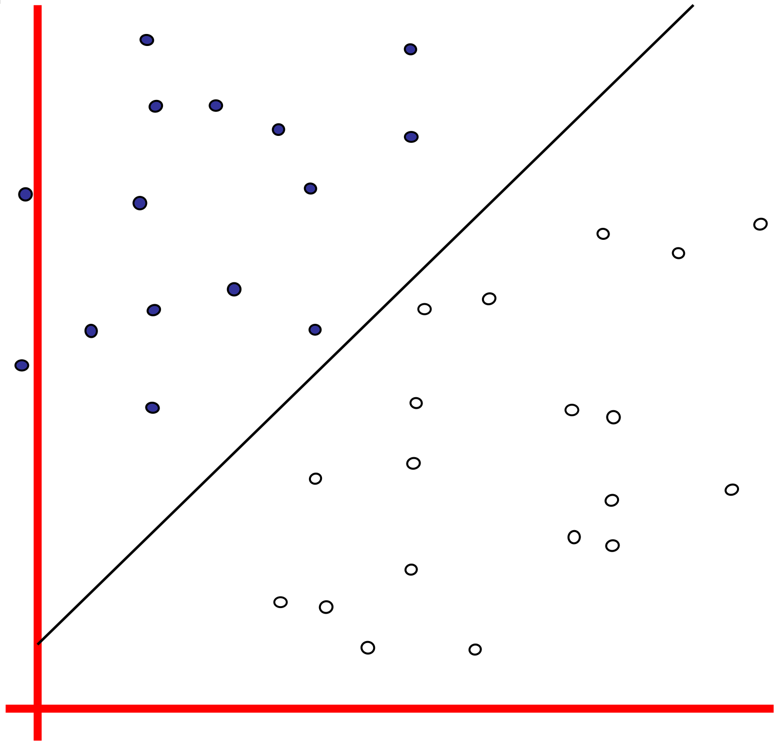
How would you classify this data?

Linear Classifiers



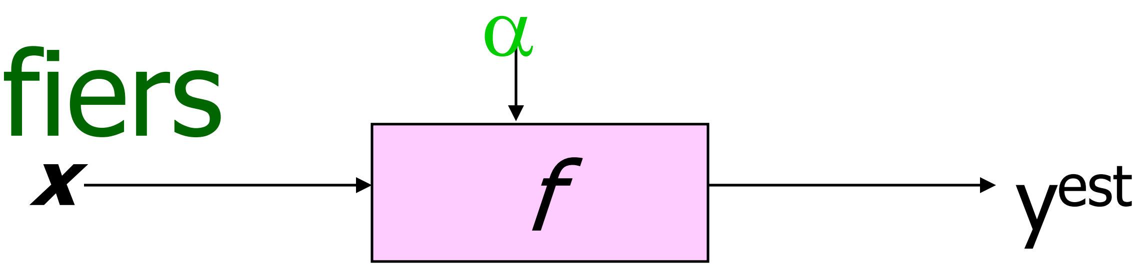
$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} - b)$

- denotes +1
- denotes -1



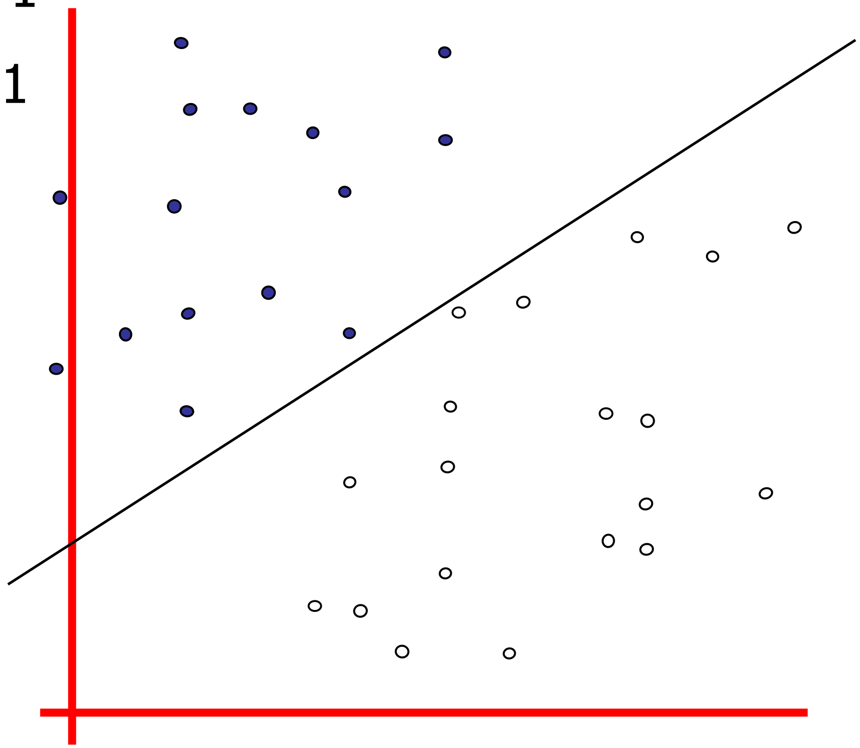
How would you classify this data?

Linear Classifiers



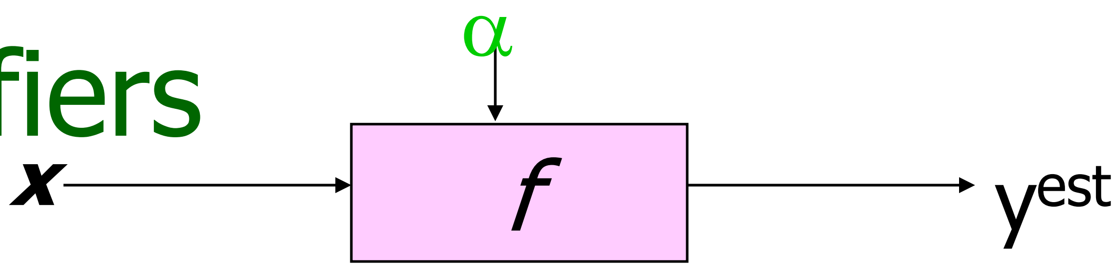
$$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} - b)$$

- denotes +1
- denotes -1



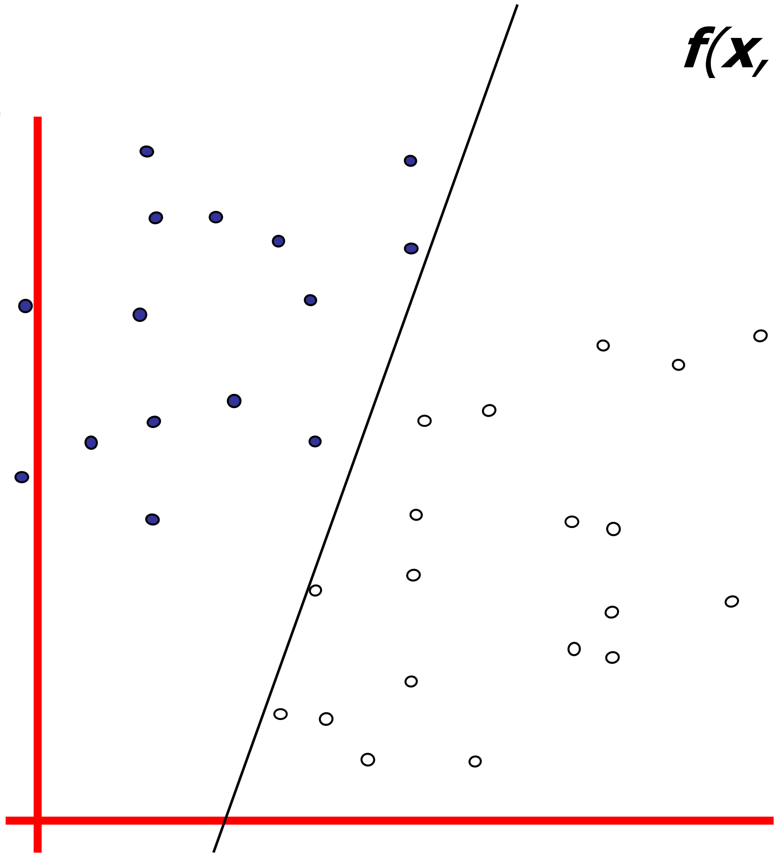
How would you classify this data?

Linear Classifiers



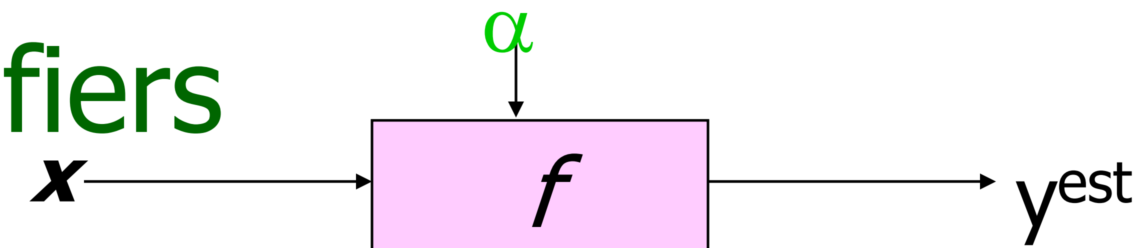
$$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} - b)$$

- denotes +1
- ◦ denotes -1



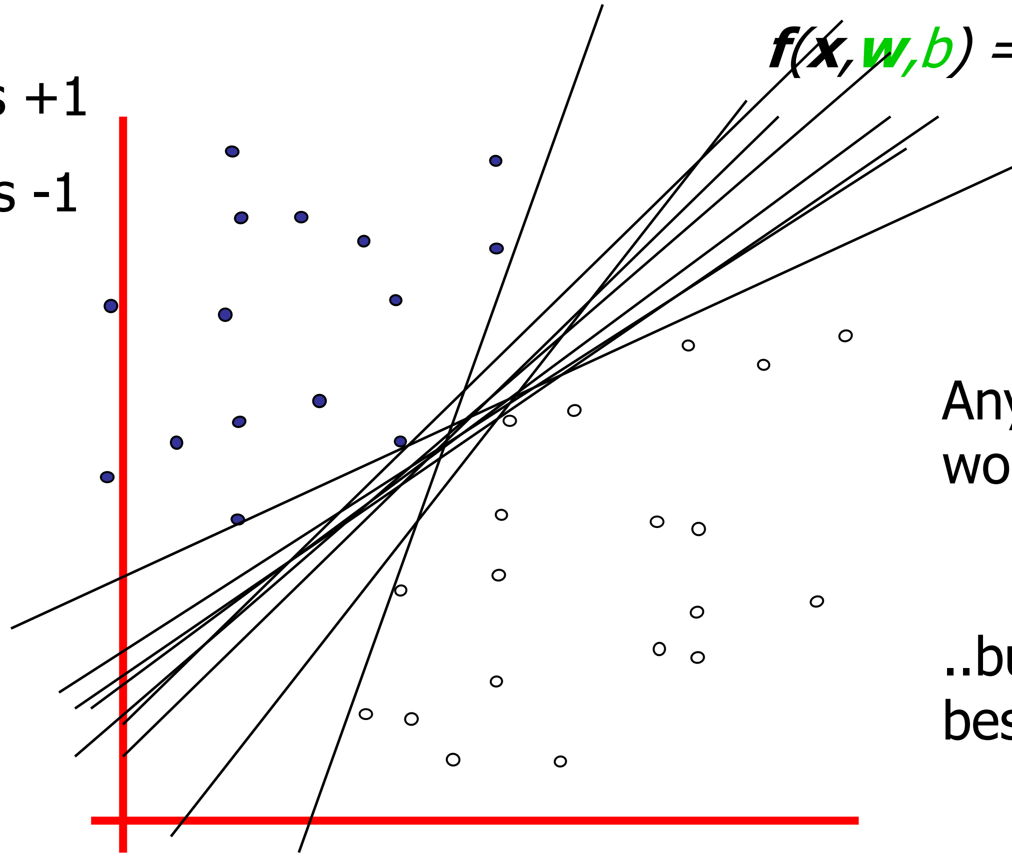
How would you classify this data?

Linear Classifiers



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

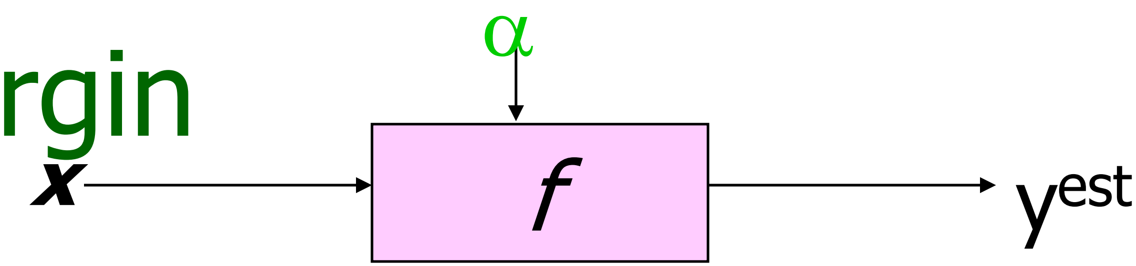
- denotes +1
- denotes -1



Any of these would be fine..

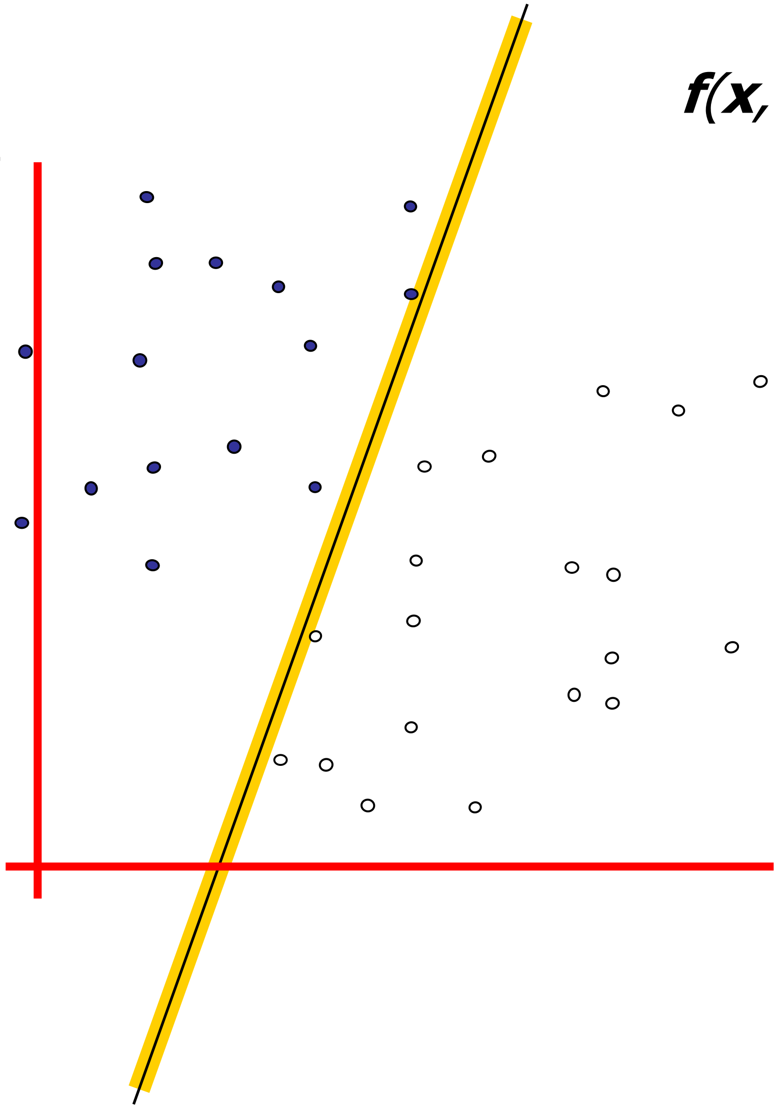
..but which is best?

Classifier Margin



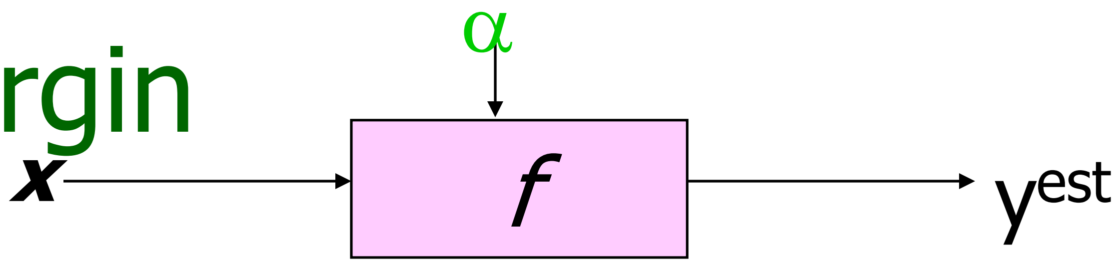
$$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} - b)$$

- denotes +1
- ○ denotes -1



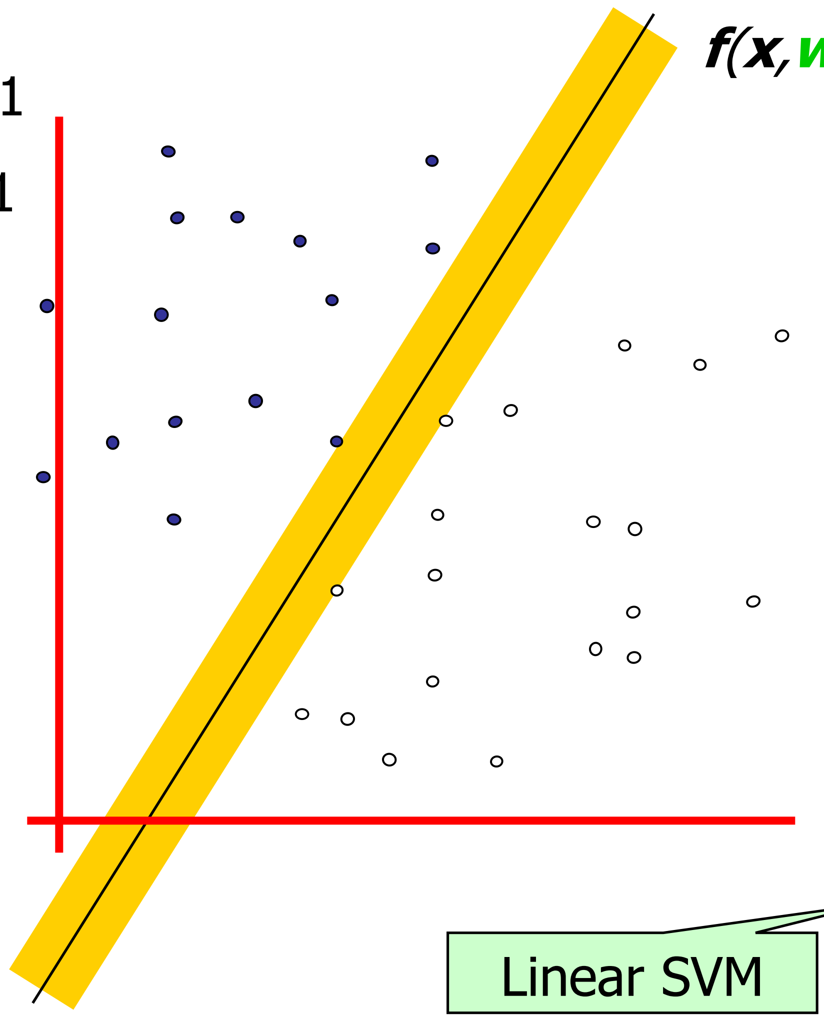
Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

Maximum Margin



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

- ● denotes +1
- ○ denotes -1

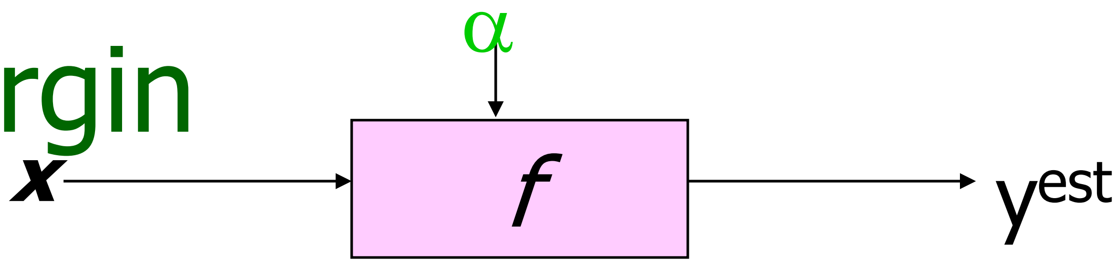


Linear SVM

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

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

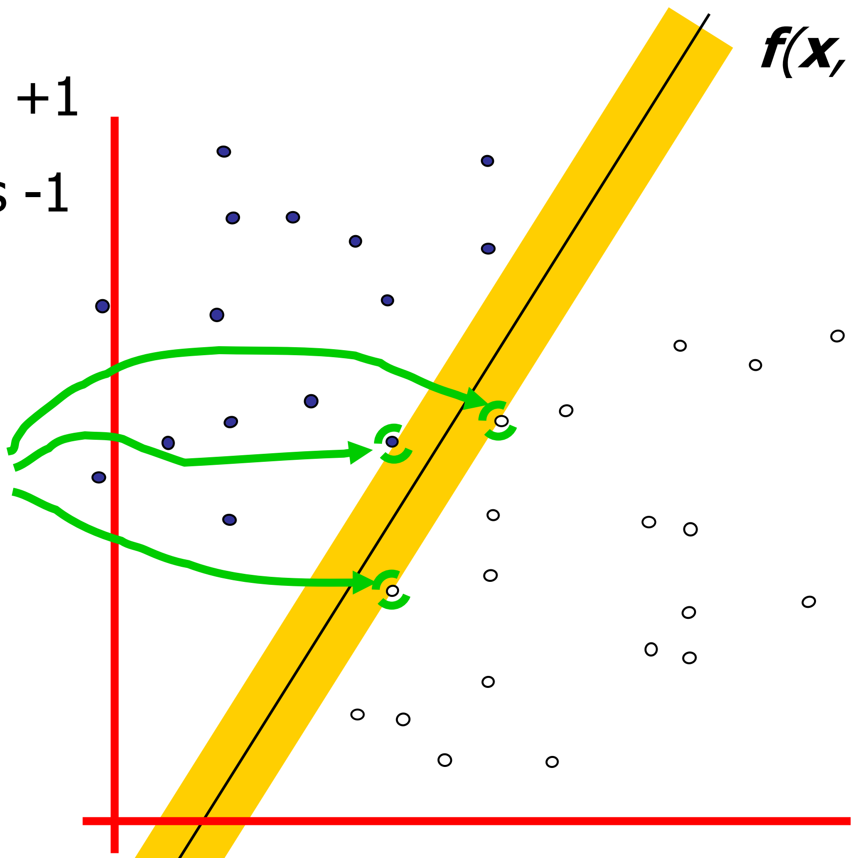
Maximum Margin



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

• denotes +1

° denotes -1



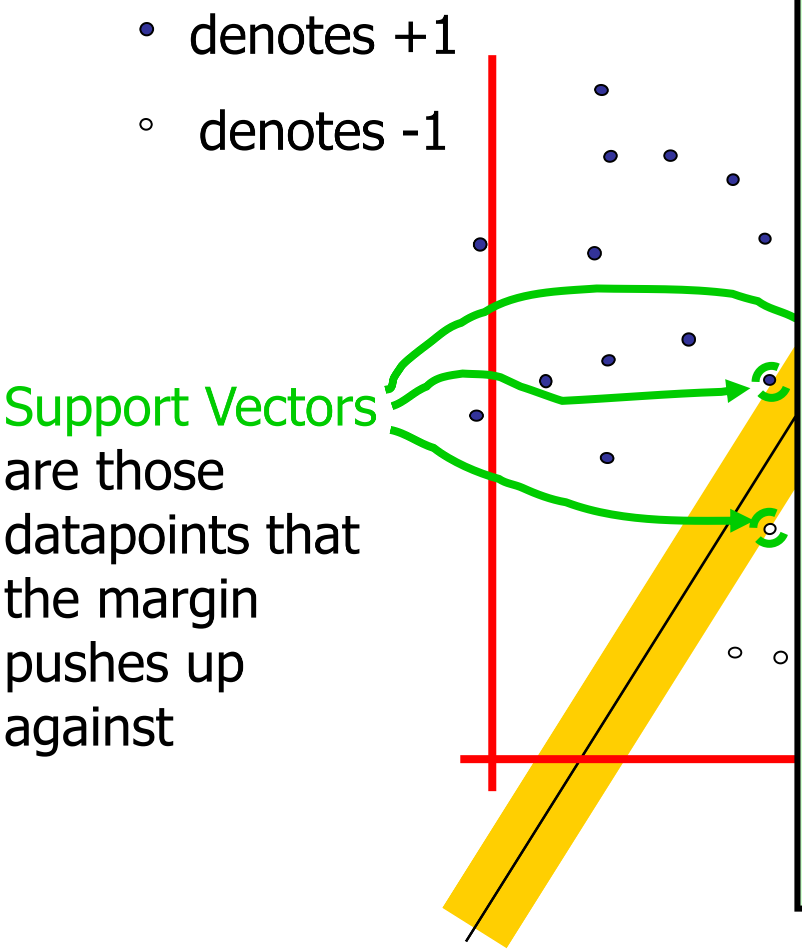
Support Vectors are those datapoints that the margin pushes up against

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

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

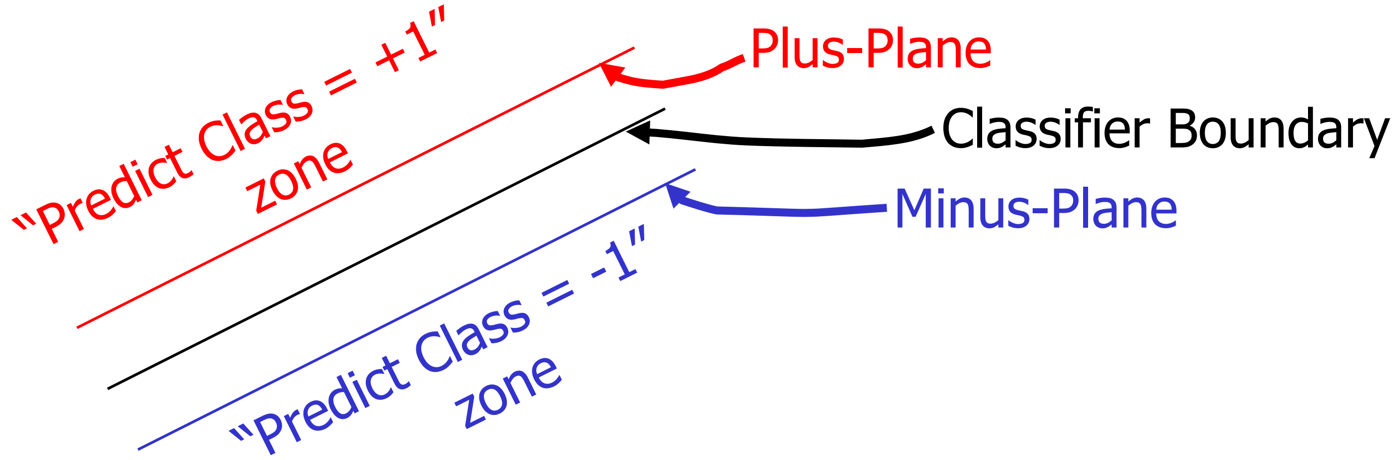
Linear SVM

Why Maximum Margin?



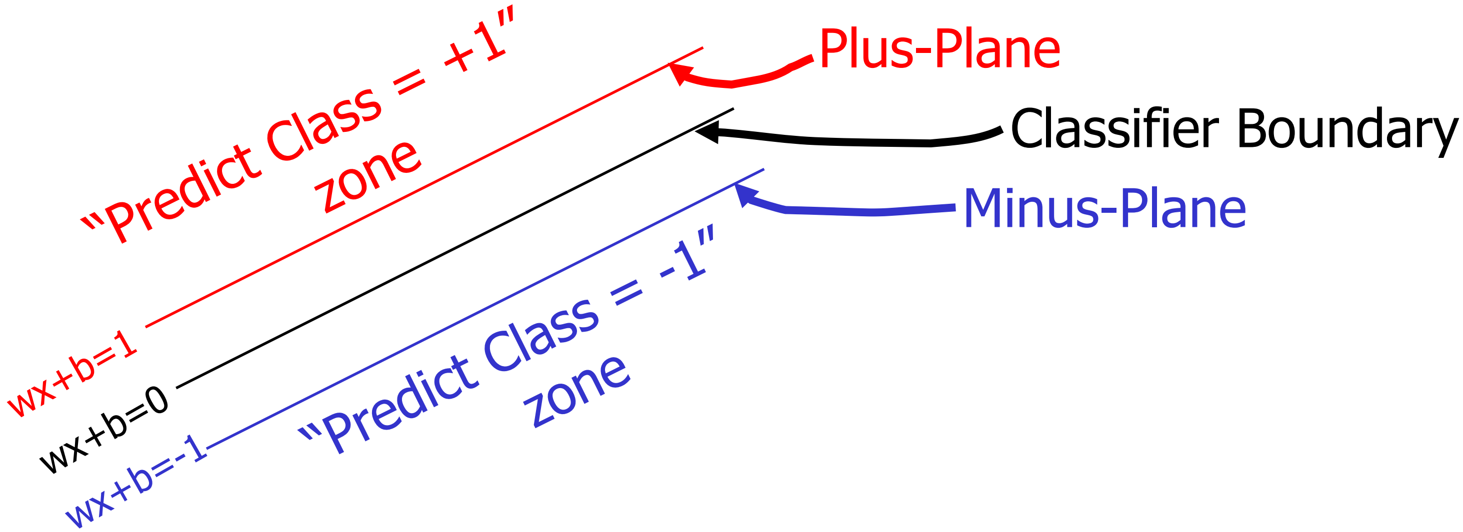
1. Intuitively this feels safest.
2. If we've made a small error in the location of the boundary (it's been jolted in its perpendicular direction) this gives us least chance of causing a misclassification.
3. LOOCV is easy since the model is immune to removal of any non-support-vector datapoints.
4. There's some theory (using VC dimension) that is related to (but not the same as) the proposition that this is a good thing.
5. Empirically it works very very well.

Specifying a line and margin



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

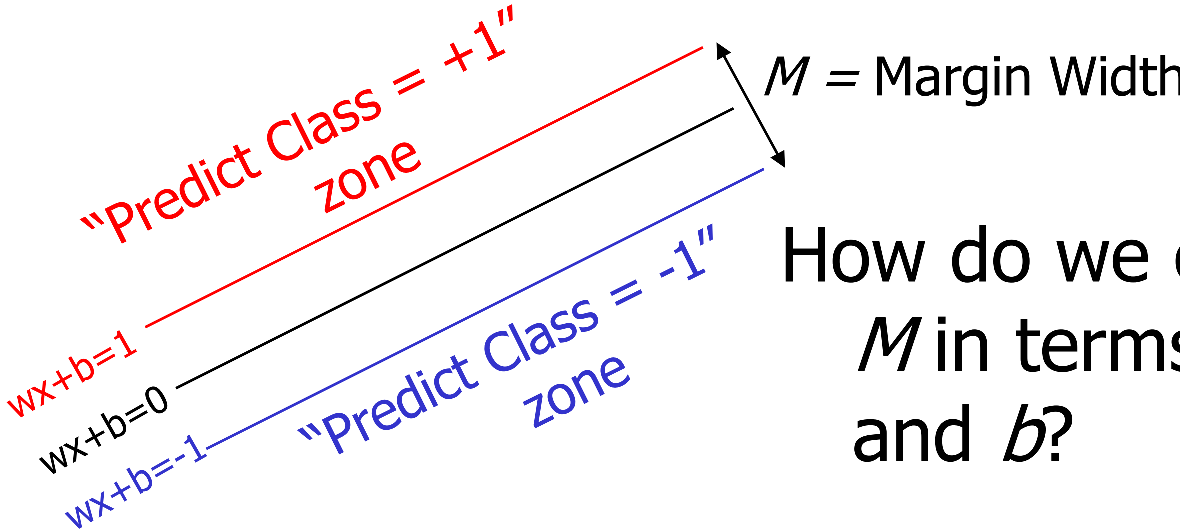
Specifying a line and margin



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

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

Computing the margin width

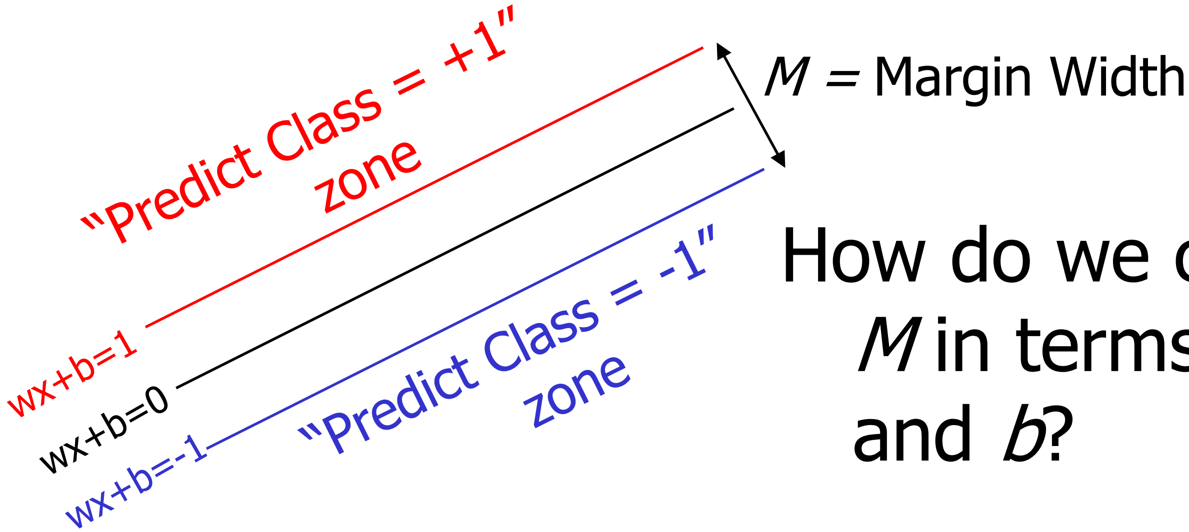


How do we compute M in terms of $\mathbf{w}$ and b?

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

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

Computing the margin width



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

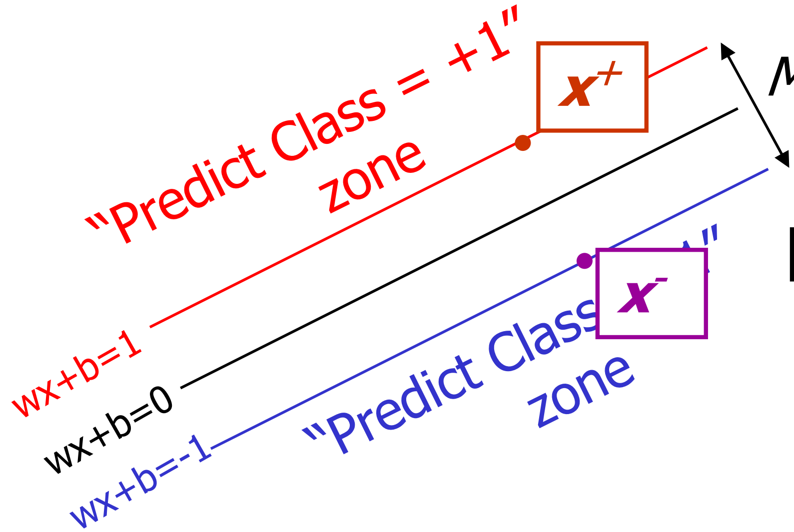
- Plus-plane = $\{ \mathbf{x} : \mathbf{w} \,.\, \mathbf{x} + b = +1 \}$
- Minus-plane = $\{ \mathbf{x} : \mathbf{w} \,.\, \mathbf{x} + b = -1 \}$

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

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

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

Computing the margin width

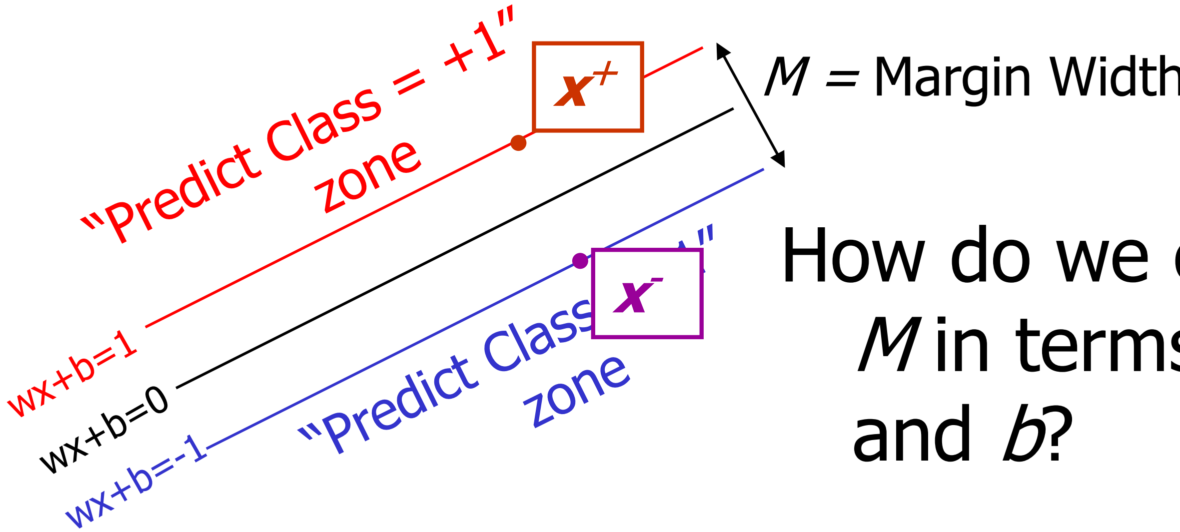


How do we compute M in terms of $\boldsymbol{w}$ and b?

- Plus-plane = $\{ \mathbf{x} : \mathbf{w} \cdot \mathbf{x} + b = +1 \}$
- Minus-plane = $\{ \mathbf{x} : \mathbf{w} \cdot \mathbf{x} + b = -1 \}$
- The vector **w** is perpendicular to the Plus Plane
- Let $\mathbf{x}^-$ be any point on the minus plane
- Let $\mathbf{x}^+$ be the closest plus-plane-point to $\mathbf{x}^-$.

Any location in R^m: not necessarily a datapoint

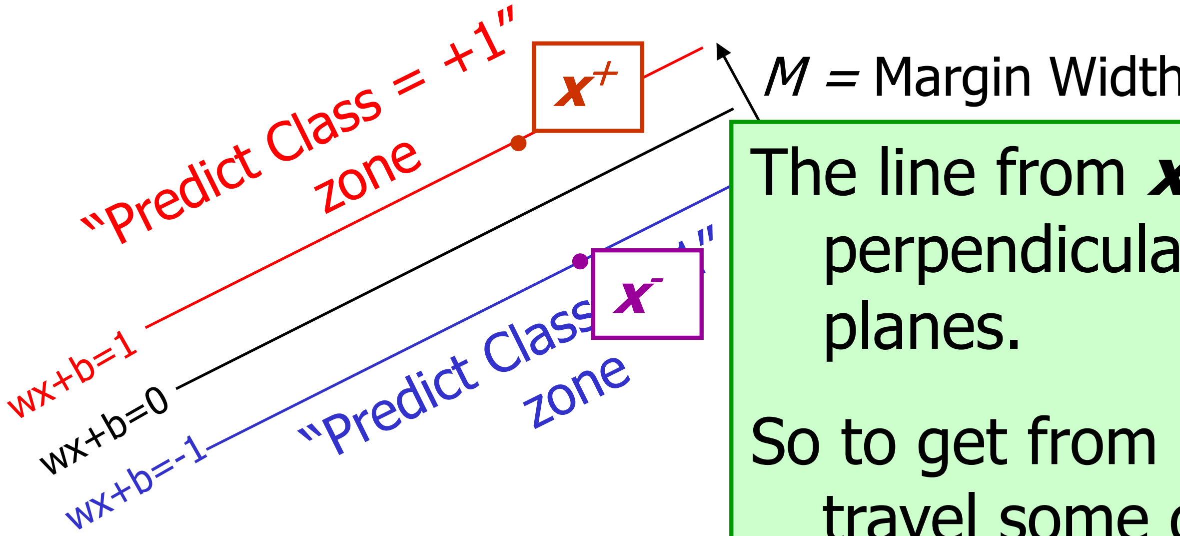
Computing the margin width



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

- Plus-plane = $\{ \boldsymbol{x} : \boldsymbol{w} \,.\, \boldsymbol{x} + b = +1 \}$
- 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 $\boldsymbol{x}^-$.
- Claim: $\boldsymbol{x}^+ = \boldsymbol{x}^- + \lambda\, \boldsymbol{w}$ for some value of λ. Why?

Computing the margin width

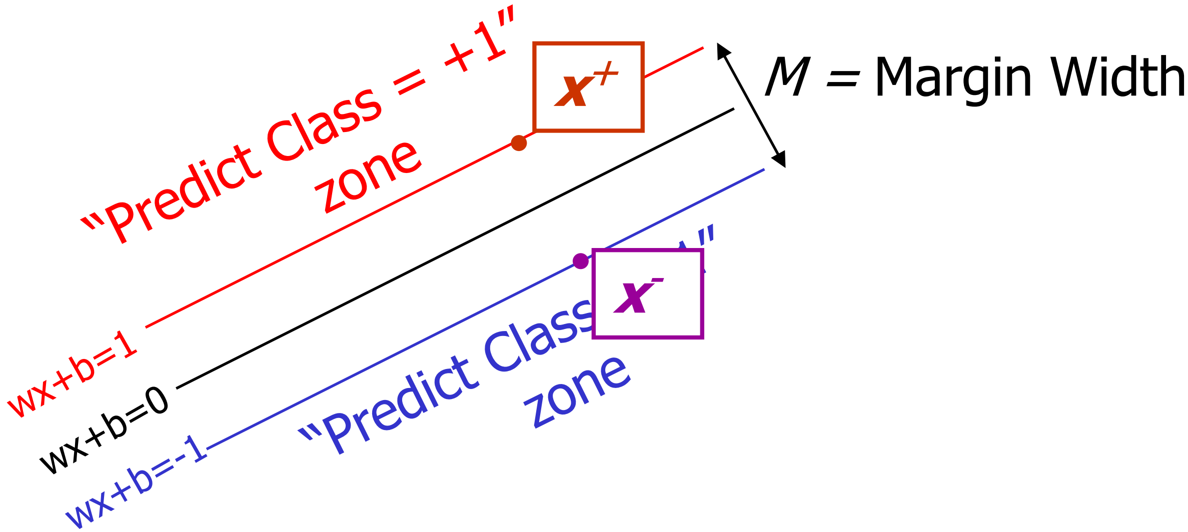


The line from $\boldsymbol{x}^-$ to $\boldsymbol{x}^+$ is perpendicular to the planes.

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

- Plus-plane = $\{ \boldsymbol{x} : \boldsymbol{w} . \boldsymbol{x} + b =$
- 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 $\boldsymbol{x}^-$.
- Claim: $\boldsymbol{x}^+ = \boldsymbol{x}^- + \lambda \boldsymbol{w}$ for some value of λ. Why?

Computing the margin width

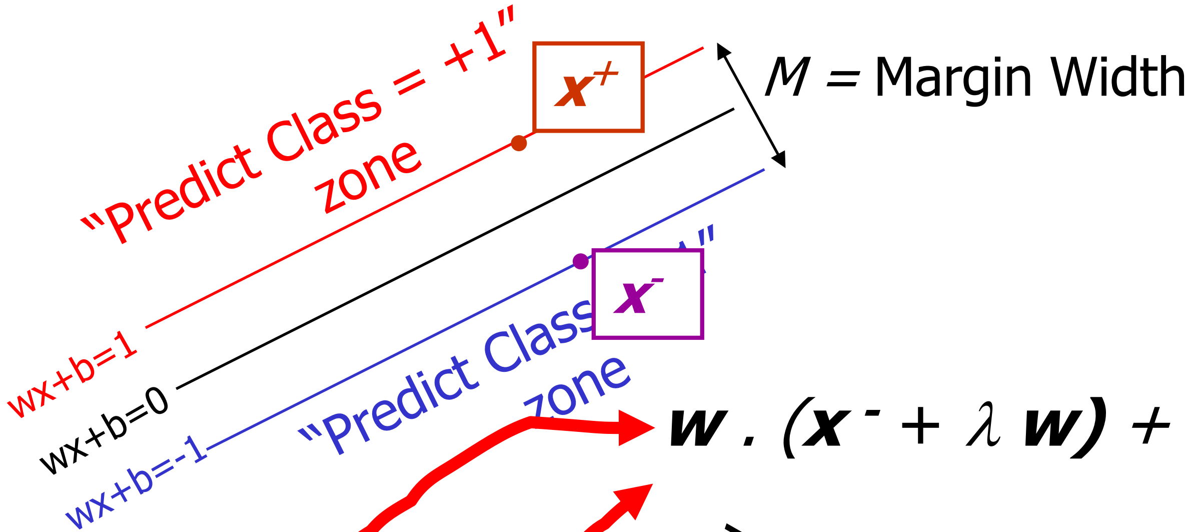


What we know:

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

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

Computing the margin width



What we know:

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

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

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

=>

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

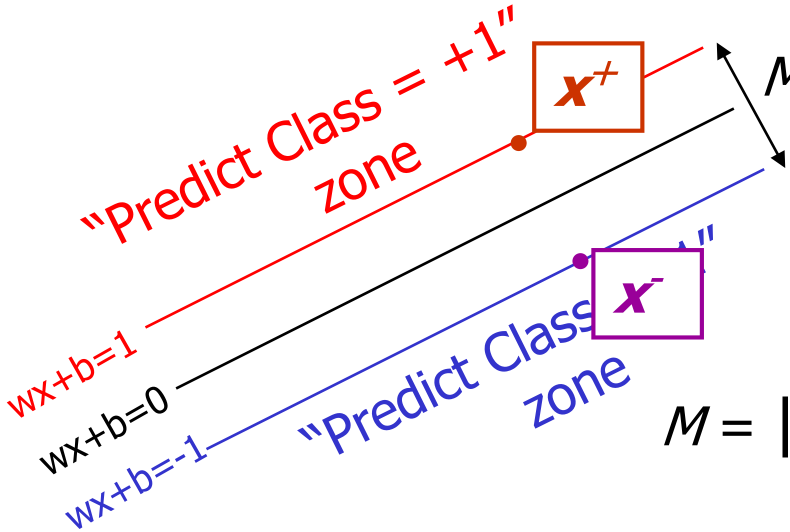
=>

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

=>

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

Computing the margin width



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

What we know:

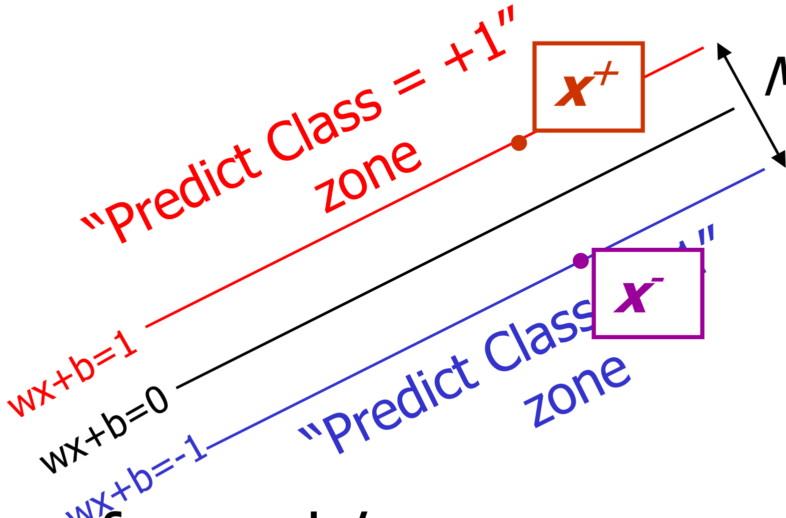
- $\boldsymbol{w} \,.\, \boldsymbol{x}^{+} + b = +1$
- $\boldsymbol{w} \,.\, \boldsymbol{x}^{-} + b = -1$
- $\boldsymbol{x}^{+} = \boldsymbol{x}^{-} + \lambda\, \boldsymbol{w}$
- $|\boldsymbol{x}^{+} - \boldsymbol{x}^{-}| = M$
- $$\lambda = \frac{2}{\mathbf{w}.\mathbf{w}}$$

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

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

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

Learning the Maximum Margin Classifier



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

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

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

So now we just need to write a program to search the space of **w**'s and *b*'s to find the widest margin that matches all the datapoints. *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 $\arg\max\limits_{\mathbf{u}} \quad c + \mathbf{d}^T\mathbf{u} + \dfrac{\mathbf{u}^T R\mathbf{u}}{2}$ ← Quadratic criterion

Subject to

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

n additional linear 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 $\arg\max_{\mathbf{u}} \quad c + \mathbf{d}^T\mathbf{u} + \dfrac{\mathbf{u}^T R\mathbf{u}}{2}$ Quadratic criterion

Subject to

...nal linear
...equality
...nstraints

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

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

And subj...

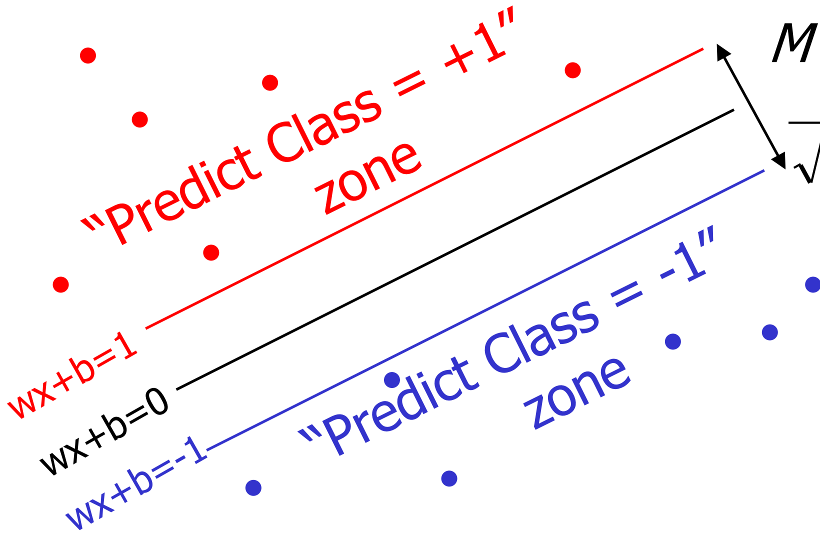
$b_{(n+1)}$

$= b_{(n+2)}$

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

e additional linear equality constraints

Learning the Maximum Margin Classifier



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

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

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

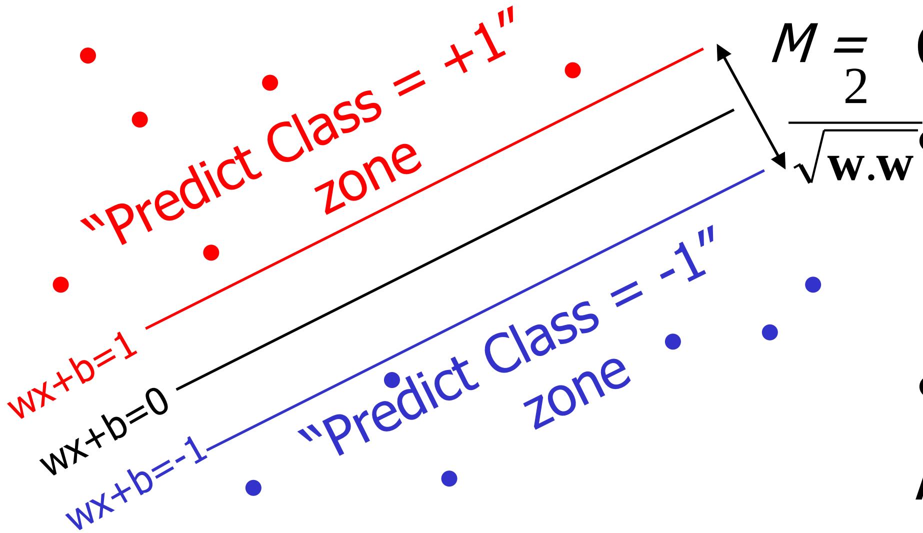
Assume R datapoints, each $(\boldsymbol{x}_k, y_k)$ where $y_k = +/- 1$

What should our quadratic optimization criterion be?

How many constraints will we have?

What should they be?

Learning the Maximum Margin Classifier



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

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

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

What should our quadratic optimization criterion be?

Minimize $\boldsymbol{w}.\boldsymbol{w}$

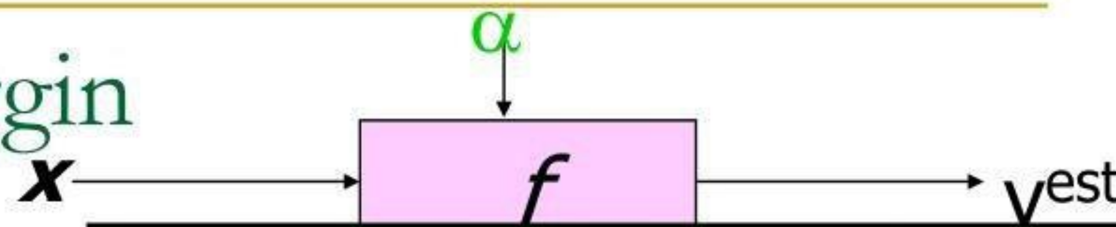
How many constraints will we have? R

What should they be?

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

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

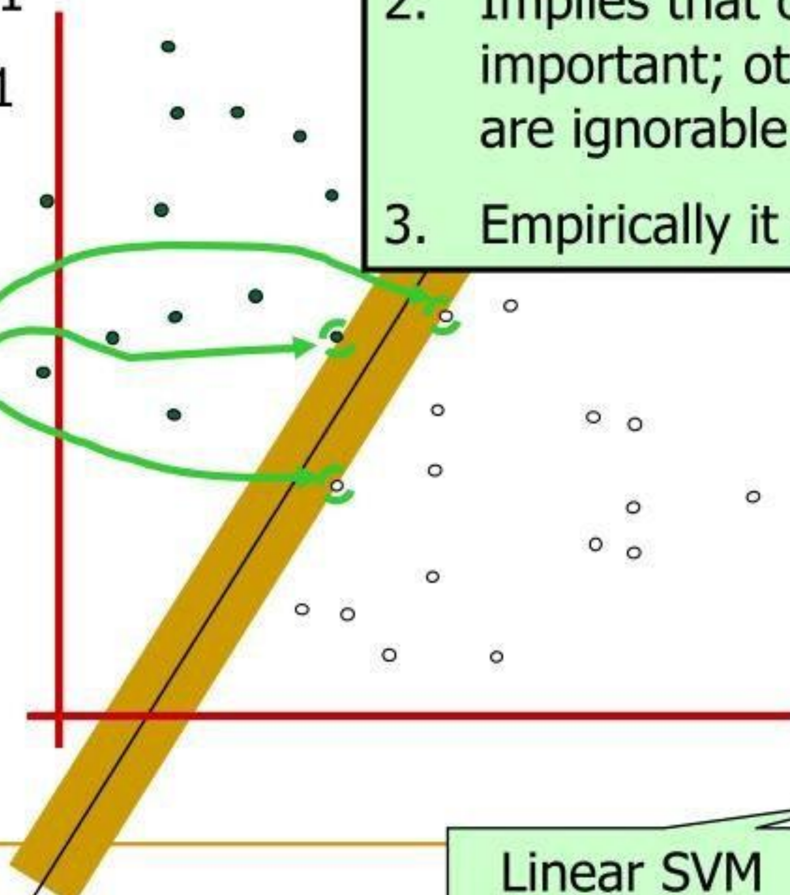
Maximum Margin



1. Maximizing the margin is good according to intuition and PAC theory
2. Implies that only support vectors are important; other training examples are ignorable.
3. Empirically it works very very well.

• denotes +1

° denotes -1



Support Vectors are those datapoints that the margin pushes up against

linear classifier with the, um, maximum margin.

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

Linear SVM

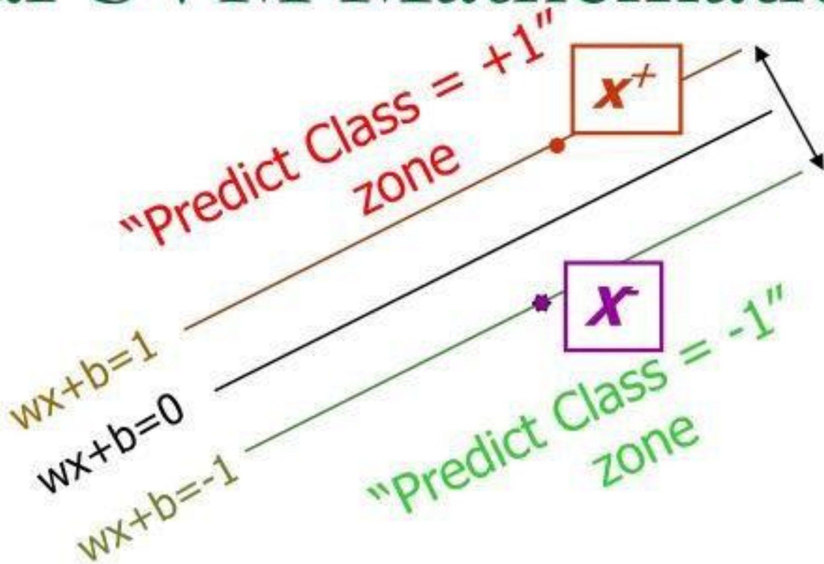
Let me digress to…what is PAC Theory?

- Two important aspects of complexity in machine learning.
-
- First, *sample complexity*: in many learning problems, training data is expensive and we should hope not to need too much of it.
- Secondly, *computational complexity*: A neural network, for example, which takes an hour to train may be of no practical use in complex financial prediction problems.
- Important that both the amount of training data required for a prescribed level of performance and the running time of the learning algorithm in learning from this data do not increase too dramatically as the `difficulty' of the learning problem increases.

Let me digress to…what is PAC Theory?

- Such issues have been formalised and investigated over the past decade within the field of `computational learning theory'.
- One popular framework for discussing such problems is the probabilistic framework which has become known as the `probably approximately correct', or PAC, model of learning.

Linear SVM Mathematically



M=Margin Width

What we know:

- ***w*** *.* ***x⁺*** *+ b = +1*
- ***w*** *.* ***x⁻*** *+ b = -1*
- ***w*** *. (*$\boldsymbol{x^+}$***-***$\boldsymbol{x^-}$*) = 2*

$$M = \frac{(x^+ - x^-) \cdot w}{|w|} = \frac{2}{|w|}$$

Linear SVM Mathematically

- Goal: **1) Correctly classify all training data**

$$wx_i + b \geq 1 \quad \textit{if } y_i = +1$$
$$wx_i + b \leq 1 \quad \textit{if } y_i = -1$$
$$y_i(wx_i + b) \geq 1 \quad \text{for all i}$$

2) Maximize the Margin $M = \frac{2}{|w|}$

same as minimize $\frac{1}{2} w^t w$

- **We can formulate a Quadratic Optimization Problem and solve for w and b**

- Minimize $\Phi(w) = \frac{1}{2} w^t w$

  subject to $y_i(wx_i + b) \geq 1 \qquad \forall i$

Solving the Optimization Problem

Find $\mathbf{w}$ and b such that
$\mathbf{\Phi}(\mathbf{w}) = ½\, \mathbf{w}^T\mathbf{w}$ is minimized;
and for all $\{(\mathbf{x_i}, y_i)\}$: $y_i(\mathbf{w}^T\mathbf{x_i} + b) \geq 1$

- **Need to optimize a *quadratic* function subject to *linear* constraints.**
- **Quadratic optimization problems are a well-known class of mathematical programming problems, and many (rather intricate) algorithms exist for solving them.**
- **The solution involves constructing a *dual problem* where a *Lagrange multiplier* α_i is associated with every constraint in the primary problem:**

Find $\alpha_1 \ldots \alpha_N$ such that
$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - ½\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x_i}^T\mathbf{x_j}$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $\alpha_i \geq 0$ for all α_i

A digression… Lagrange Multipliers

- In mathematical optimization, the method of **Lagrange multipliers** provides a strategy for finding the maxima and minima of a function subject to constraints.
- For instance, consider the optimization problem

maximize $f(x, y)$
subject to $g(x, y) = c.$

- We introduce a new variable (λ) called a Lagrange multiplier, and study the Lagrange function defined by

  $$\Lambda(x, y, \lambda) = f(x, y) + \lambda \cdot \Big(g(x, y) - c\Big).$$

  (the λ term may be either added or subtracted.)
- If (x,y) is a maximum for the original constrained problem, then there exists a λ such that (x,y,λ) is a stationary point for the Lagrange function
- (stationary points are those points where the partial derivatives of Λ are zero).

The Optimization Problem Solution

- The solution has the form:

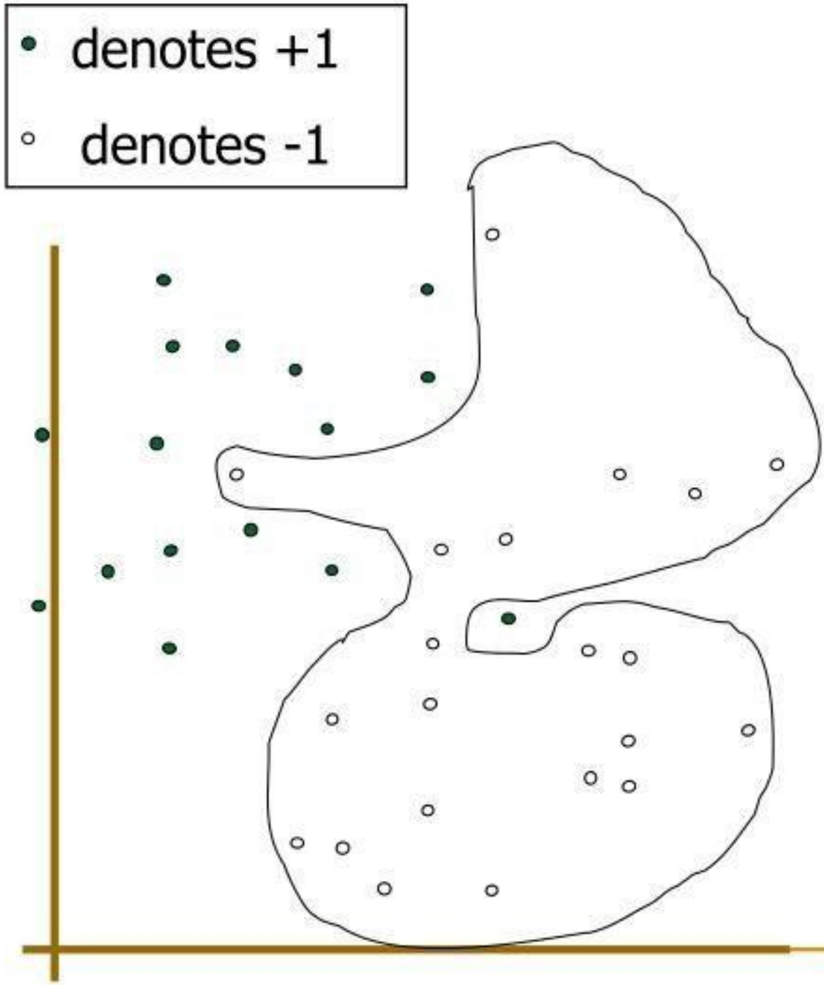
$$\mathbf{w} = \Sigma \alpha_i y_i \mathbf{x_i} \qquad b = y_k - \mathbf{w^T x_k} \text{ for any } \mathbf{x_k} \text{ such that } \alpha_k \neq 0$$

- Each non-zero α_i indicates that corresponding $\mathbf{x_i}$ is a support vector.
- Then the classifying function will have the form:

$$f(\mathbf{x}) = \Sigma \alpha_i y_i \mathbf{x_i^T x} + b$$

- Notice that it relies on an *inner product* between the test point $\mathbf{x}$ and the support vectors $\mathbf{x_i}$ – we will return to this later.
- Also keep in mind that solving the optimization problem involved computing the inner products $\mathbf{x_i^T x_j}$ between all pairs of training points.

Dataset with noise



- **Hard Margin:** **So far we require all data points be classified correctly**

  - No training error

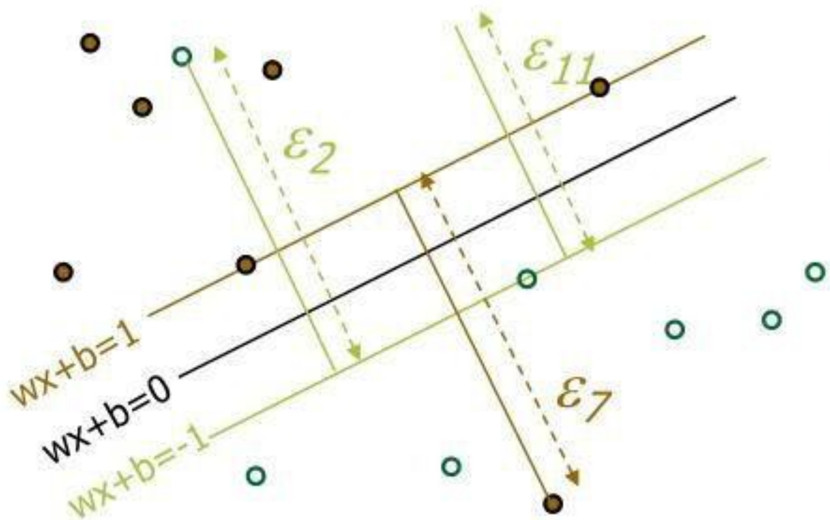
- **What if the training set is noisy?**

  - Solution 1: **use very powerful kernels**

OVERFITTING!

Soft Margin Classification

***Slack variables ξi* can be added to allow misclassification of difficult or noisy examples.**



What should our quadratic optimization criterion be?

Minimize

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

Hard Margin v.s. Soft Margin

- **The old formulation:**

Find $\mathbf{w}$ and b such that
$\mathbf{\Phi}(\mathbf{w}) = ½\, \mathbf{w}^T\mathbf{w}$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$
$y_i\,(\mathbf{w^T x_i} + \mathrm{b}) \geq 1$

- **The new formulation incorporating slack variables:**

Find $\mathbf{w}$ and b such that
$\mathbf{\Phi}(\mathbf{w}) = ½\, \mathbf{w}^T\mathbf{w} + C\Sigma\xi_i$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$
$y_i\,(\mathbf{w^T x_i} + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$ for all i

- **Parameter *C* can be viewed as a way to control overfitting.**

Linear SVMs: Overview

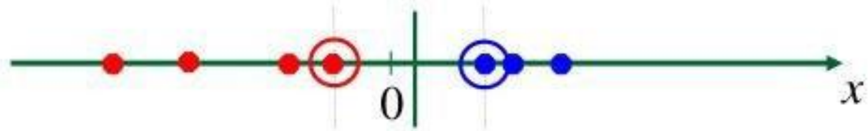
- **The classifier is a *separating hyperplane.***
- **Most "important" training points are support vectors; they define the hyperplane.**
- **Quadratic optimization algorithms can identify which training points x_i are support vectors with non-zero Lagrangian multipliers α_i.**
- **Both in the dual formulation of the problem and in the solution training points appear only inside dot products:**

Find $\alpha_1 \ldots \alpha_N$ such that
$Q(\alpha) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $0 \leq \alpha_i \leq C$ for all α_i

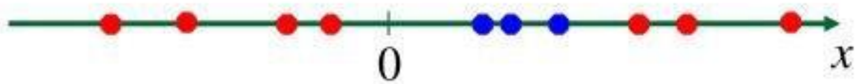
$$f(\mathbf{x}) = \Sigma\alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b$$

Non-linear SVMs

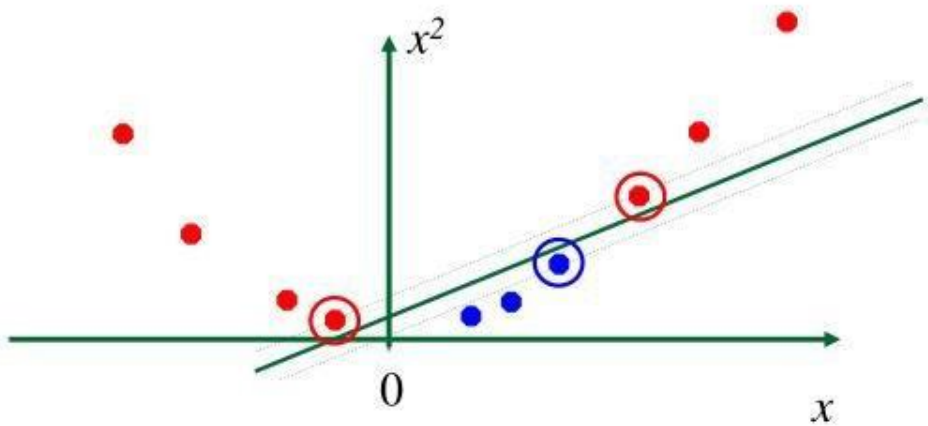
- Datasets that are linearly separable with some noise work out great:



- But what are we going to do if the dataset is just too hard?

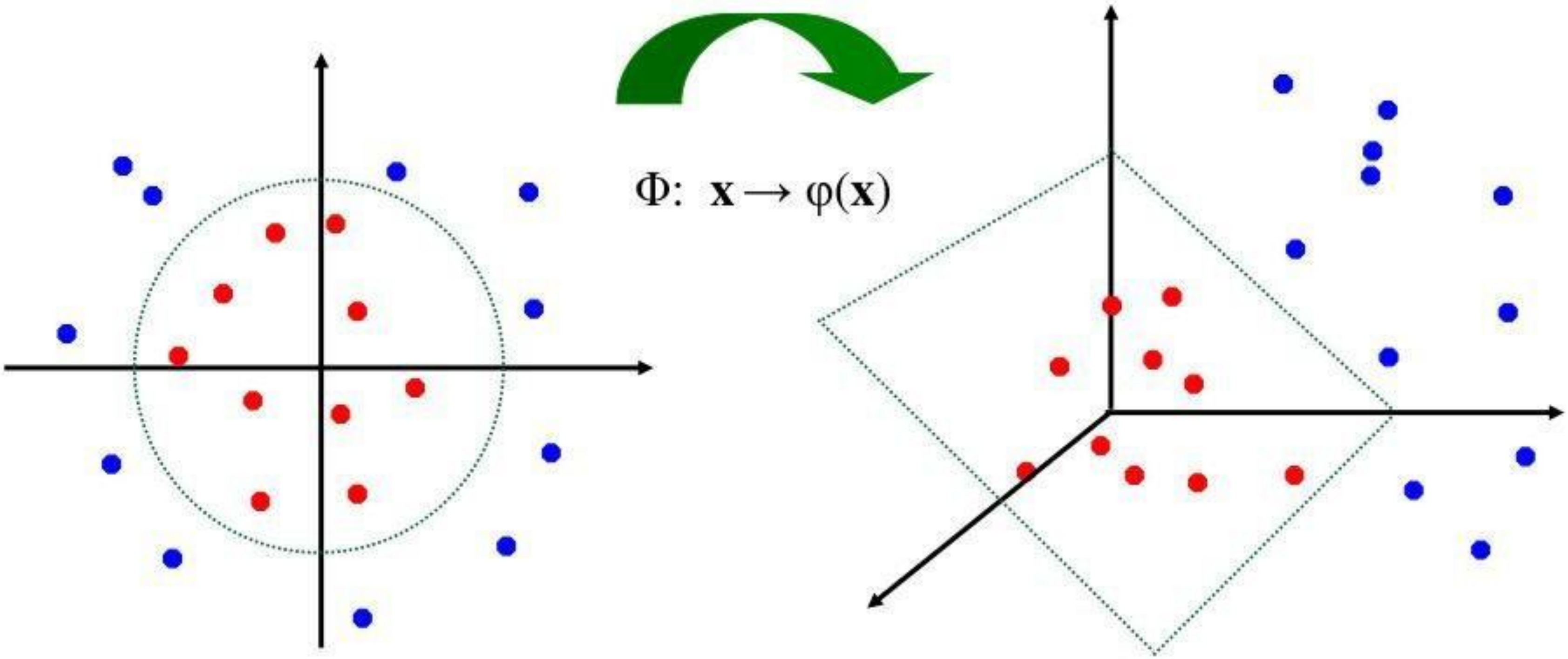


- How about… mapping data to a higher-dimensional space:



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:



The "Kernel Trick"

- **The linear classifier relies on dot product between vectors $K(\mathbf{x}_i,\mathbf{x}_j)=\mathbf{x}_i^T\mathbf{x}_j$**
- **If every data point is mapped into high-dimensional space via some transformation Φ: $\mathbf{x} \rightarrow \varphi(\mathbf{x})$, the dot product becomes:**

$$K(\mathbf{x}_i,\mathbf{x}_j)= \varphi(\mathbf{x}_i)^T\varphi(\mathbf{x}_j)$$

- **A *kernel function* is some function that corresponds to an inner product in some expanded feature space.**
- **Example:**

2-dimensional vectors $\mathbf{x}=[x_1\ \ x_2]$; let $K(\mathbf{x}_i,\mathbf{x}_j)=(1 + \mathbf{x}_i^T\mathbf{x}_j)^2$,

Need to show that $K(\mathbf{x}_i,\mathbf{x}_j)= \varphi(\mathbf{x}_i)^T\varphi(\mathbf{x}_j)$:

$$K(\mathbf{x}_i,\mathbf{x}_j)=(1 + \mathbf{x}_i^T\mathbf{x}_j)^2,$$

$$= 1+ x_{i1}^2x_{j1}^2 + 2\, x_{i1}x_{j1}\, x_{i2}x_{j2}+ x_{i2}^2x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2}$$

$$= [1\ \ x_{i1}^2\ \ \sqrt{2}\, x_{i1}x_{i2}\ \ x_{i2}^2\ \ \sqrt{2}x_{i1}\ \ \sqrt{2}x_{i2}]^T [1\ \ x_{j1}^2\ \ \sqrt{2}\, x_{j1}x_{j2}\ \ x_{j2}^2\ \ \sqrt{2}x_{j1}\ \ \sqrt{2}x_{j2}]$$

$$= \varphi(\mathbf{x}_i)^T\varphi(\mathbf{x}_j), \quad \text{where } \varphi(\mathbf{x}) = [1\ \ x_1^2\ \ \sqrt{2}\, x_1x_2\ \ x_2^2\ \ \sqrt{2}x_1\ \ \sqrt{2}x_2]$$

What Functions are Kernels?

- **For some functions $K(\mathbf{x}_i,\mathbf{x}_j)$ checking that**

  $K(\mathbf{x}_i,\mathbf{x}_j)= \varphi(\mathbf{x}_i)^T\varphi(\mathbf{x}_j)$ can be cumbersome.
- **Mercer's theorem:**

  Every semi-positive definite symmetric function is a kernel

Examples of Kernel Functions

- Linear: $K(\mathbf{x_i},\mathbf{x_j})= \mathbf{x_i}^{\mathsf{T}}\mathbf{x_j}$
- Polynomial of power p: $K(\mathbf{x_i},\mathbf{x_j})= (1+ \mathbf{x_i}^{\mathsf{T}}\mathbf{x_j})^p$
- Gaussian (radial-basis function network):

$$K(\mathbf{x_i},\mathbf{x_j}) = \exp\left(-\frac{\left\|\mathbf{x_i}-\mathbf{x_j}\right\|^2}{2\sigma^2}\right)$$

- Sigmoid: $K(\mathbf{x_i},\mathbf{x_j})= \tanh(\beta_0\mathbf{x_i}^{\mathsf{T}}\mathbf{x_j} + \beta_1)$

Non-linear SVMs Mathematically

- **Dual problem formulation:**

Find $\alpha_1 \ldots \alpha_N$ such that
$Q(\alpha) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $\alpha_i \geq 0$ for all α_i

- **The solution is:**

$$f(\mathbf{x}) = \Sigma\alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_j) + b$$

- **Optimization techniques for finding α_i's remain the same!**

Nonlinear SVM - Overview

- SVM locates a separating hyperplane in the feature space and classify points in that space
- It does not need to represent the space explicitly, simply by defining a kernel function
- The kernel function plays the role of the dot product in the feature space.

Properties of SVM

- **Flexibility in choosing a similarity function**
- **Sparseness of solution when dealing with large data sets**

  - only support vectors are used to specify the separating hyperplane
- **Ability to handle large feature spaces**

  - complexity does not depend on the dimensionality of the feature space
- **Overfitting can be controlled by soft margin approach**
- **Nice math property: a simple convex optimization problem which is guaranteed to converge to a single global solution**
- **Feature Selection**

SVM Applications

- **SVM has been used successfully in many real-world problems**

  - text (and hypertext) categorization

  - image classification – different types of sub-problems

  - bioinformatics (Protein classification,

  Cancer classification)

  - hand-written character recognition

Weakness of SVM

- **It is sensitive to noise**

  - A relatively small number of mislabeled examples can dramatically decrease the performance

- **It only considers two classes**

  - how to do multi-class classification with SVM?

  - Answer:

  1) with output arity m, learn m SVM's

  - **SVM 1 learns "Output==1" vs "Output != 1"**
  - **SVM 2 learns "Output==2" vs "Output != 2"**
  - **:**
  - **SVM m learns "Output==m" vs "Output != m"**

  2)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.

Application: Text Categorization

- Task: The classification of natural text (or hypertext) documents into a fixed number of predefined categories based on their content.

  - email filtering, web searching, sorting documents by topic, etc..

- A document can be assigned to more than one category, so this can be viewed as a series of binary classification problems, one for each category

Application : Face Expression Recognition

- Construct feature space, by use of eigenvectors or other means
- Multiple class problem, several expressions
- Use multi-class SVM

Some Issues

- **Choice of kernel**
  - Gaussian or polynomial kernel is default
  - if ineffective, more elaborate kernels are needed

- **Choice of kernel parameters**
  - e.g. σ in Gaussian kernel
  - σ is the distance between closest points with different classifications
  - In the absence of reliable criteria, applications rely on the use of a validation set or cross-validation to set such parameters.

- **Optimization criterion** – Hard margin v.s. Soft margin
  - a lengthy series of experiments in which various parameters are tested

Additional Resources

- **LibSVM**
- **An excellent tutorial on VC-dimension and Support Vector Machines:**

  C.J.C. Burges. A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery, 2(2):955-974, 1998.
- **The VC/SRM/SVM Bible:**

  Statistical Learning Theory by Vladimir Vapnik, Wiley-Interscience; 1998

  http://www.kernel-machines.org/

Reference

- **Support Vector Machine Classification of Microarray Gene Expression Data**, Michael P. S. Brown William Noble Grundy, David Lin, Nello Cristianini, Charles Sugnet, Manuel Ares, Jr., David Haussler
- www.cs.utexas.edu/users/mooney/cs391L/svm.**ppt**
- **Text categorization with Support Vector Machines:**
  learning with many relevant features

  T. Joachims, ECML - 98