Support Vector Machines

Natasha Balac, Ph.D.
Paul Rodriguez, Ph.D.
Predictive Analytics Center of Excellence
San Diego Supercomputer Center
University of California, San Diego

Support Vector Machines (SVM)

1. **Overview**
2. Linear SVM
   - Finding Support Vectors
3. Non-linear SVM
   - Kernel Trick
4. Real Applications
5. Conclusion

Support Vector Machines (SVM) Overview

- **Supervised Learning Algorithm**
  - Classification/Regression algorithm
  - Blend of linear modeling + instance based learning

- **SVM**
  - Uses a small number of critical boundary instances (support vectors) from each class
  - Goal: To build the linear discriminant function that separates classes with maximum margins

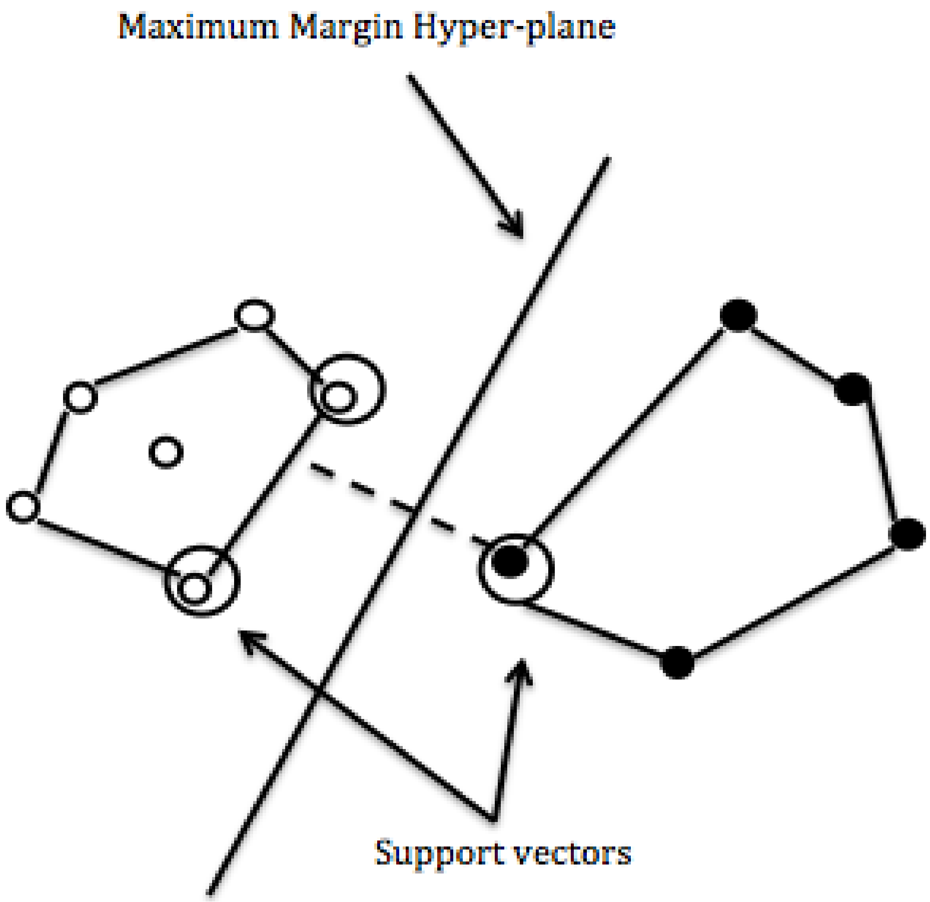
Support Vector Machines Properties

- **Define non-linear functions for SVM**
  - Overcome the limitations of linear boundaries
  - Include extra nonlinear terms in the calculations
  - Form quadratic, cubic, higher-order decision boundaries

- **Resilient to over fitting**
  - The maximum margin hyper plane

- **They are fast in the nonlinear case**
  - Employ a clever mathematical trick to avoid the creation of “pseudo-attributes”
  - Nonlinear space is created implicitly

The Maximum Margin Hyper-plane

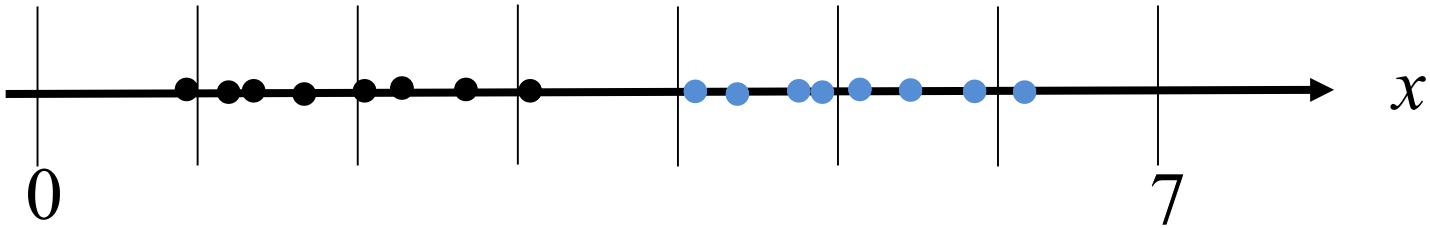


Support Vector Machines (SVM)

Outlines

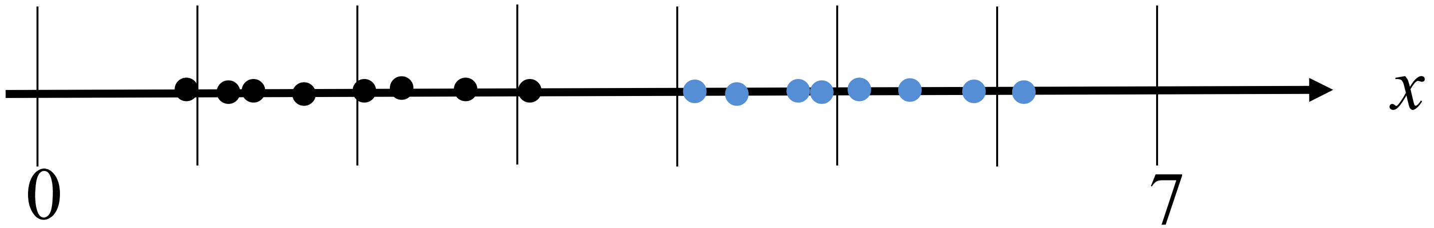
1. Overview
2. **Linear SVM**
    - Finding Support Vectors
3. Non-linear SVM
    - Kernel Trick
4. Real Applications
5. Conclusion

Recall decision threshold:



Given the above points in 1 dimension, 2 classes (black and blue dots), what value of x separates the classes? Write this as a decision rule:

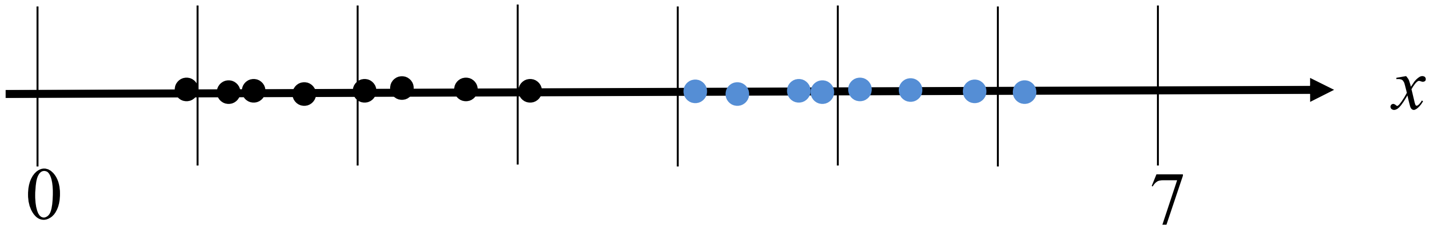
Recall decision threshold:



Given the above points in 1 dimension, 2 classes (black and blue dots), what value of x separates the classes? Write this as a decision rule:

Output = Blue if $x >= 3.5$
Output = Black if $x < 3.5$

Recall decision threshold:



decision rule:

Output = Blue if $x >= 3.5$

Output = Black if $x < 3.5$

Write this as a simple step function where Class Blue=1, Class Black = -1:

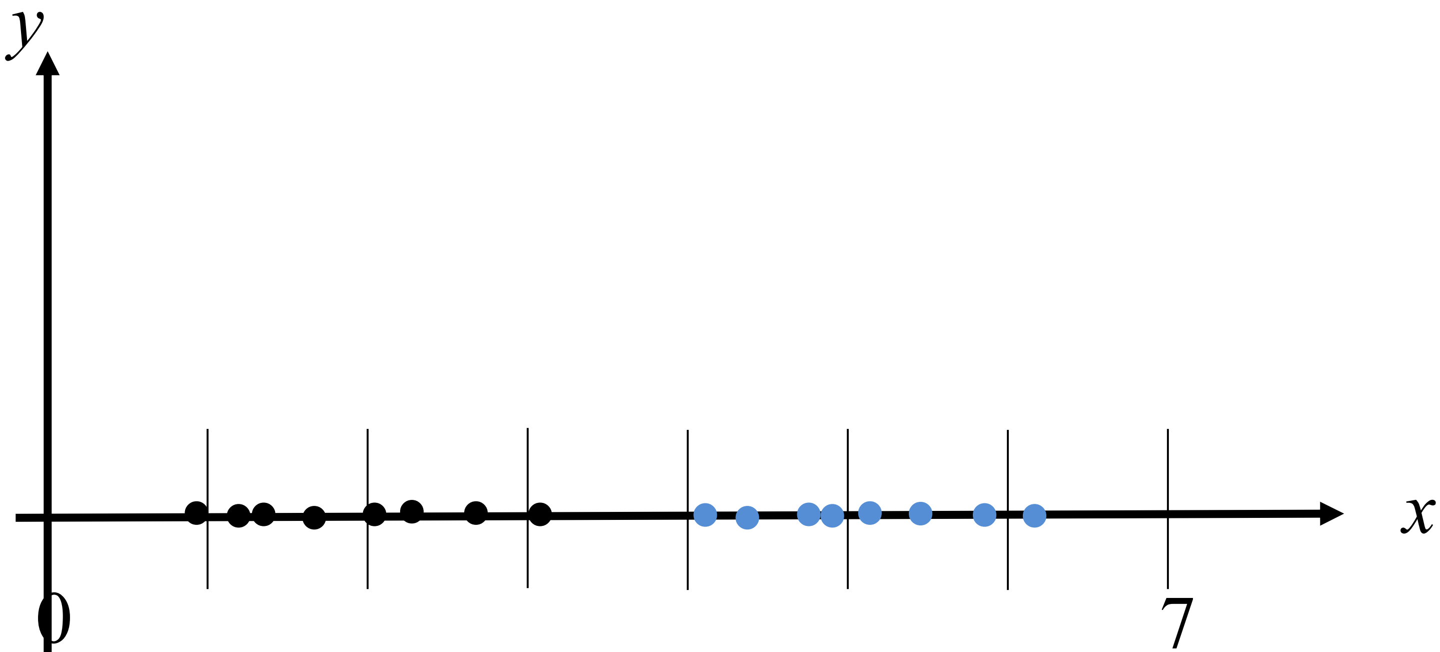
$y = \{ 1 \text{ if } x >= 3.5; \quad -1 \text{ if } x < 3.5$

Recall decision threshold:

draw step function : y={ 1 if $x >= 3.5$; -1 if $x<3.5$

AKA: sign(x-3.5) defined as x-3.5<0 is neg, x-3.5>=0 is pos

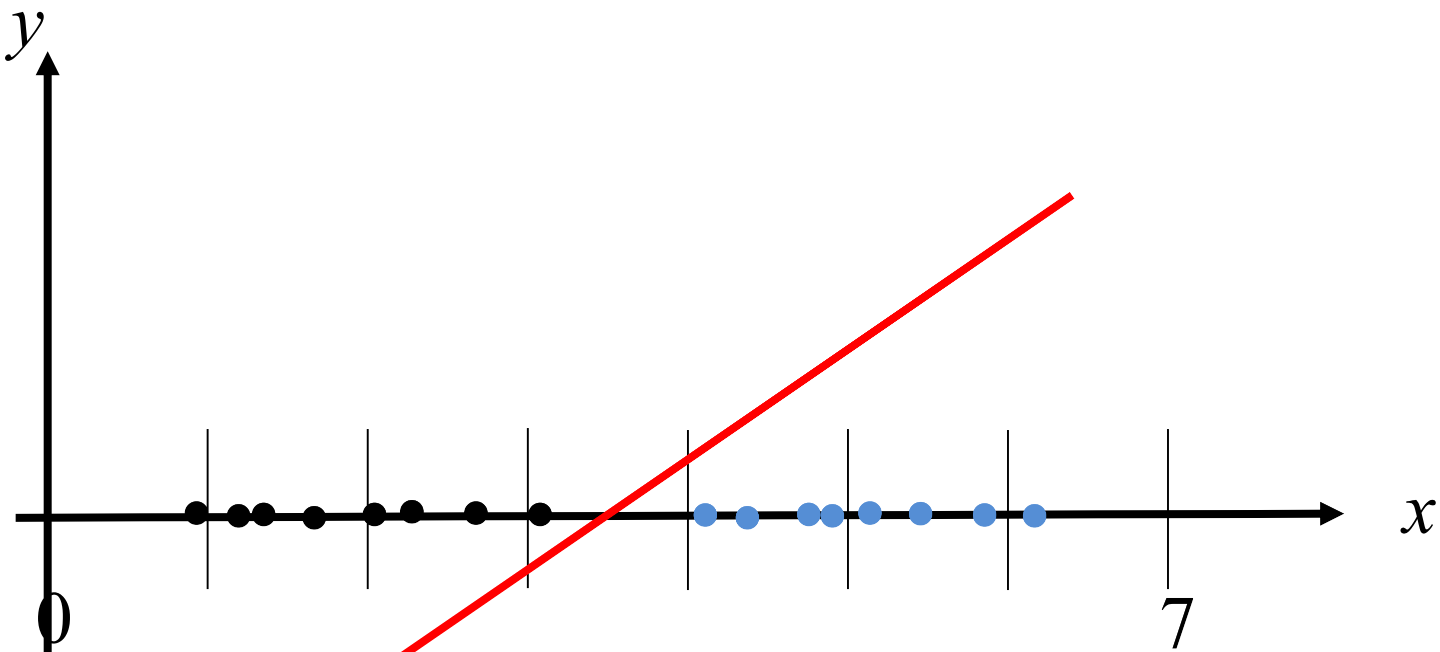
Recall decision threshold:



Draw linear function : $y = w * x + b$,

let w=1 and find b so that $w * x + b = 0$ when x=3.5

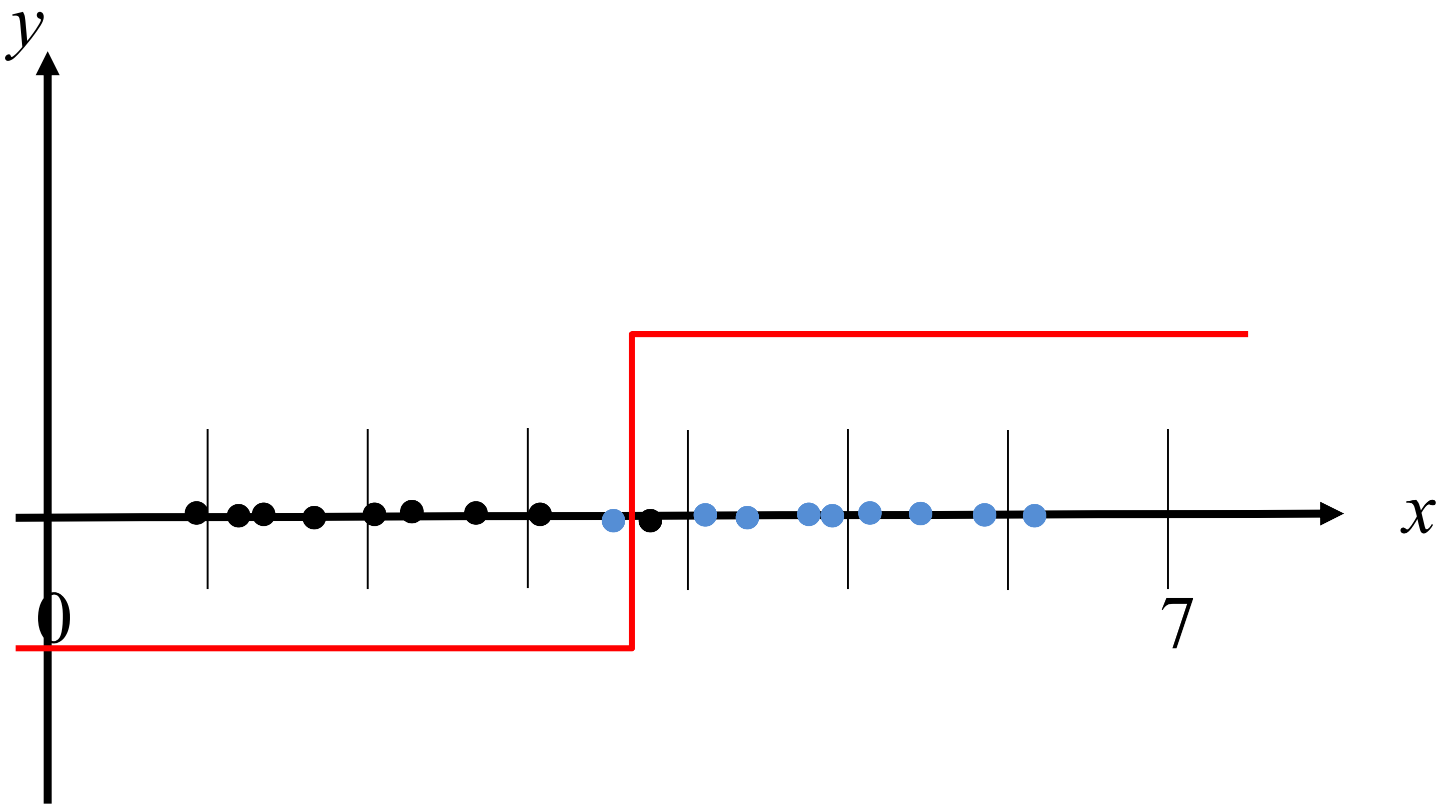
Recall decision threshold:



Draw linear function : y=w*x+b,

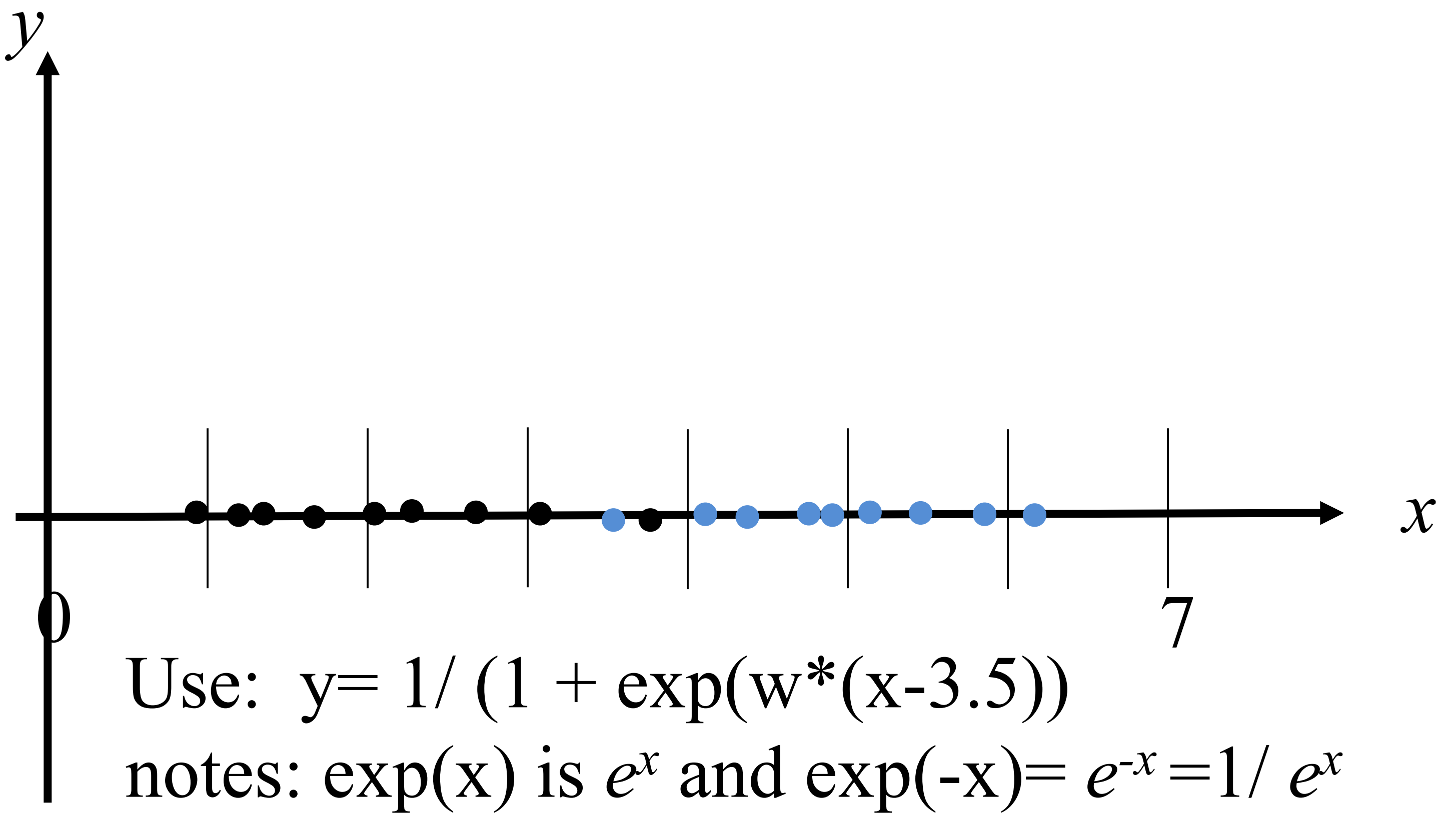
let w=1, b=-3.5, so that y<0 for black,y>0 for blue, and the line crosses the x-axis at x=3.5

Recall decision threshold:



What's if there are points that misclassify?

Sigmoidal (soft) decision threshold:



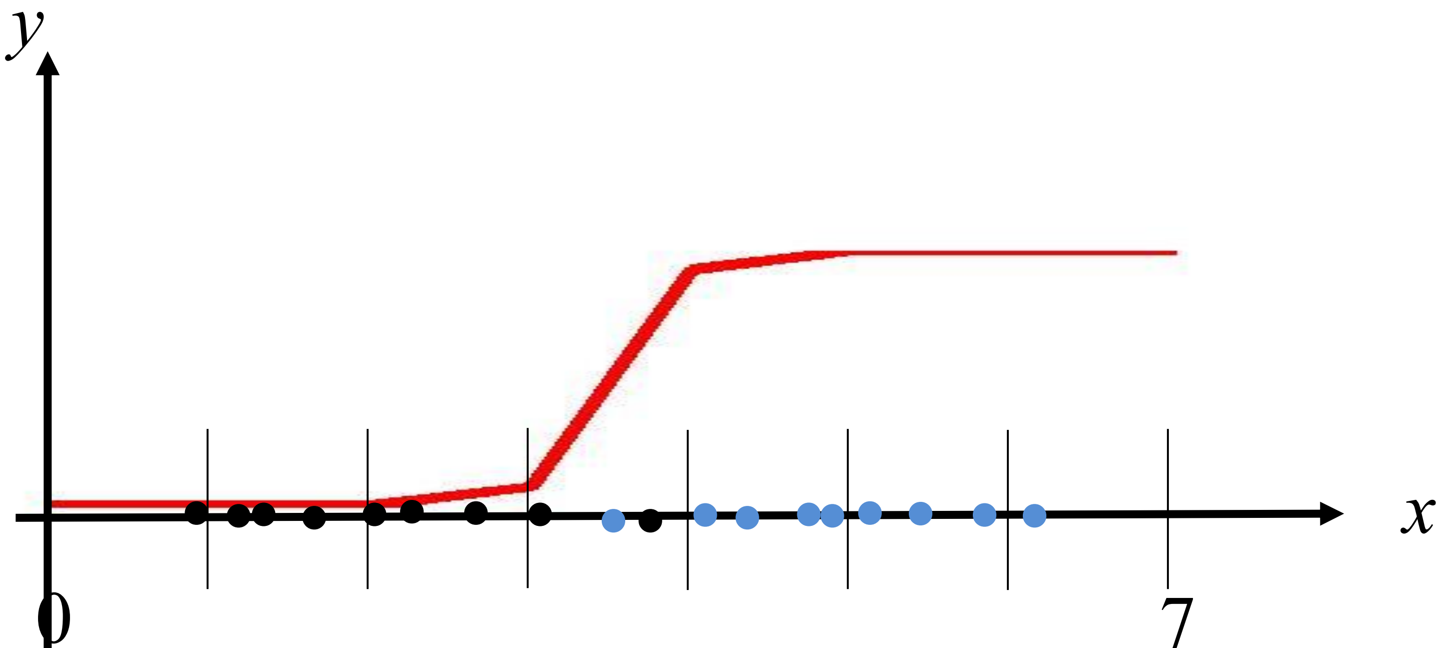
Use: y= 1/ (1 + exp(w*(x-3.5)))

notes: exp(x) is e^x and exp(-x)= e^{-x} =1/ e^x

let w=5 or w=-5 and try a few *x* values

assume class values are 0,1

Sigmoidal (soft) decision threshold:



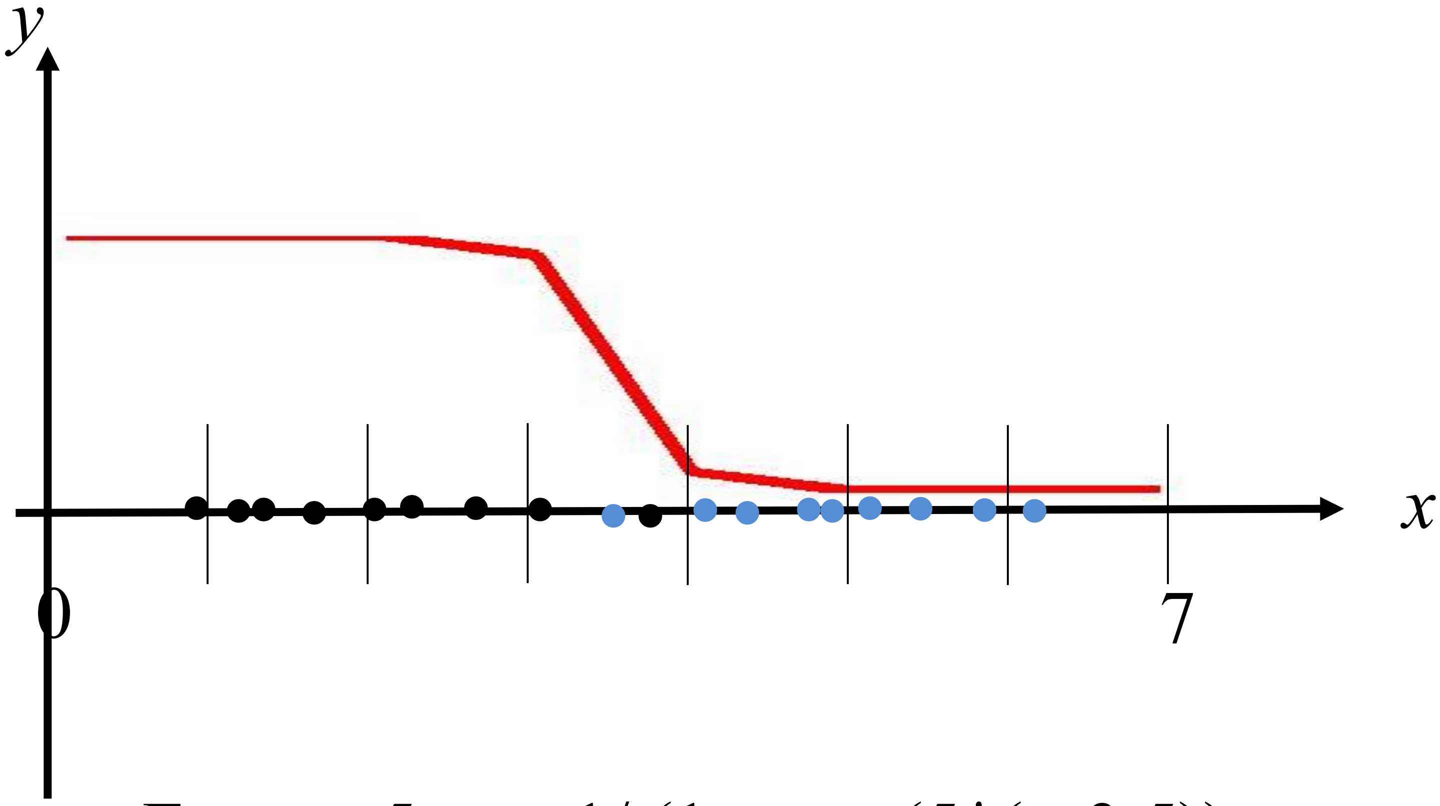
Try: y= 1/ (1 + exp(-5*(x-3.5))

Alternatively use 1/1+exp(*wx*+*b*)

for w= -5   b= 5*3.5

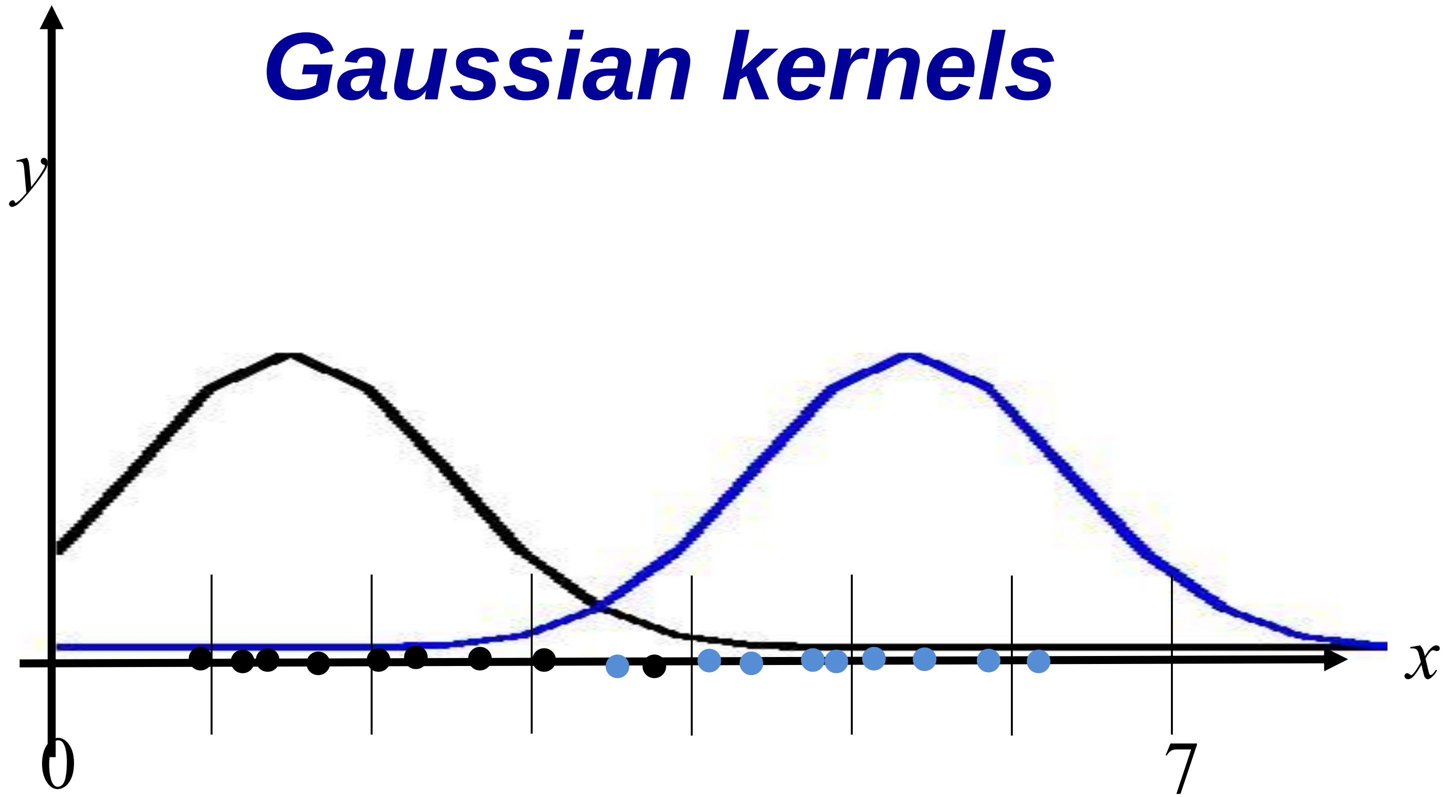
Curve is like the probability of blue as x=0…7

Sigmoidal (soft) decision threshold:



For w=5, y= 1/ (1 + exp(5*(x-3.5))

Gaussian kernels

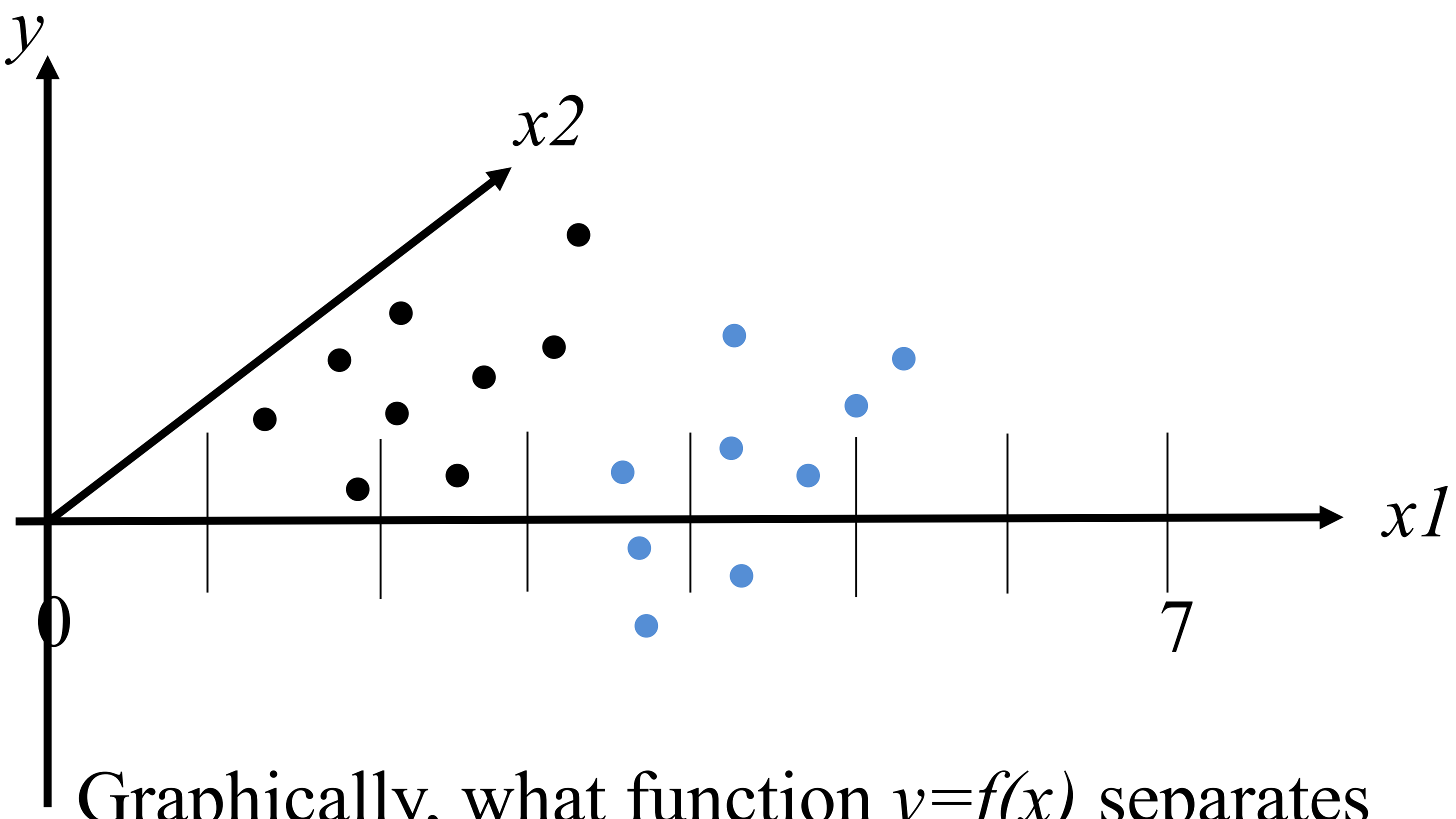


Try: y= exp(-1*((x-1.5).^2)/2) AND exp(-1*((x-5.5).^2)/2)
to model class distributions,

What is decision rule here?

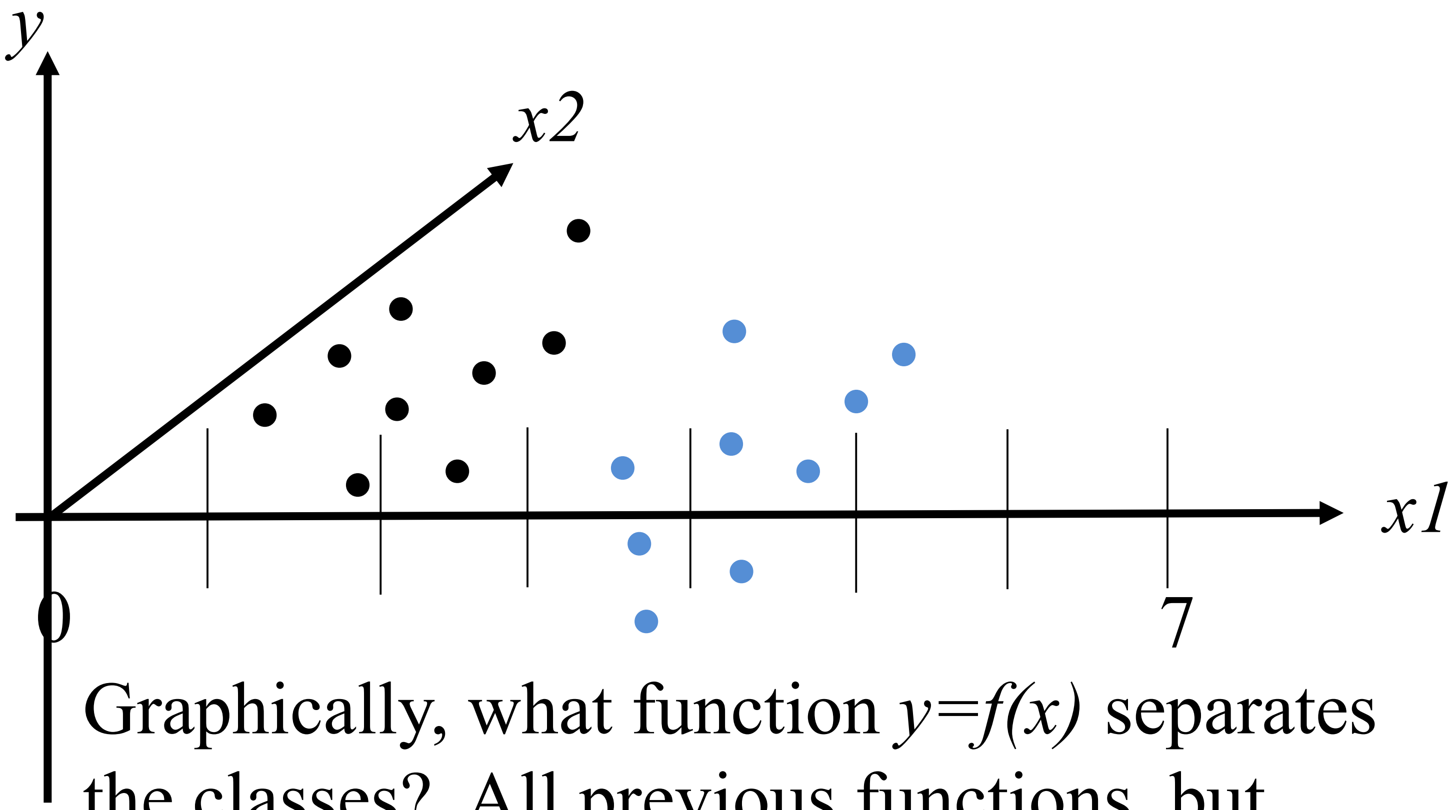
What is likelihood of being in classes wrt these curves?

In 2 dimensions:



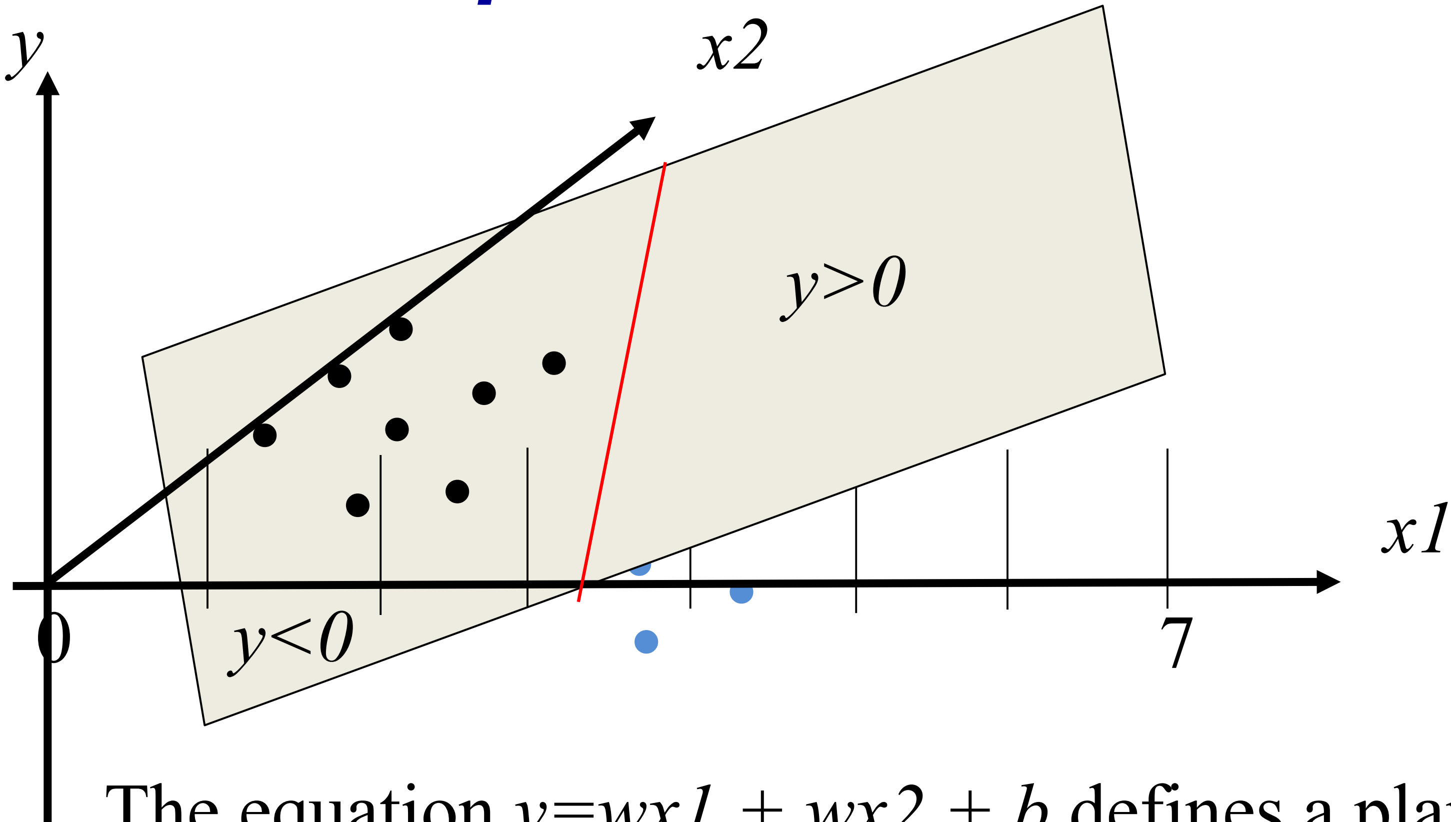
Graphically, what function $y=f(x)$ separates the classes?

In two input dimensions:



Graphically, what function *y=f(x)* separates the classes? All previous functions, but now they define hypersurfaces

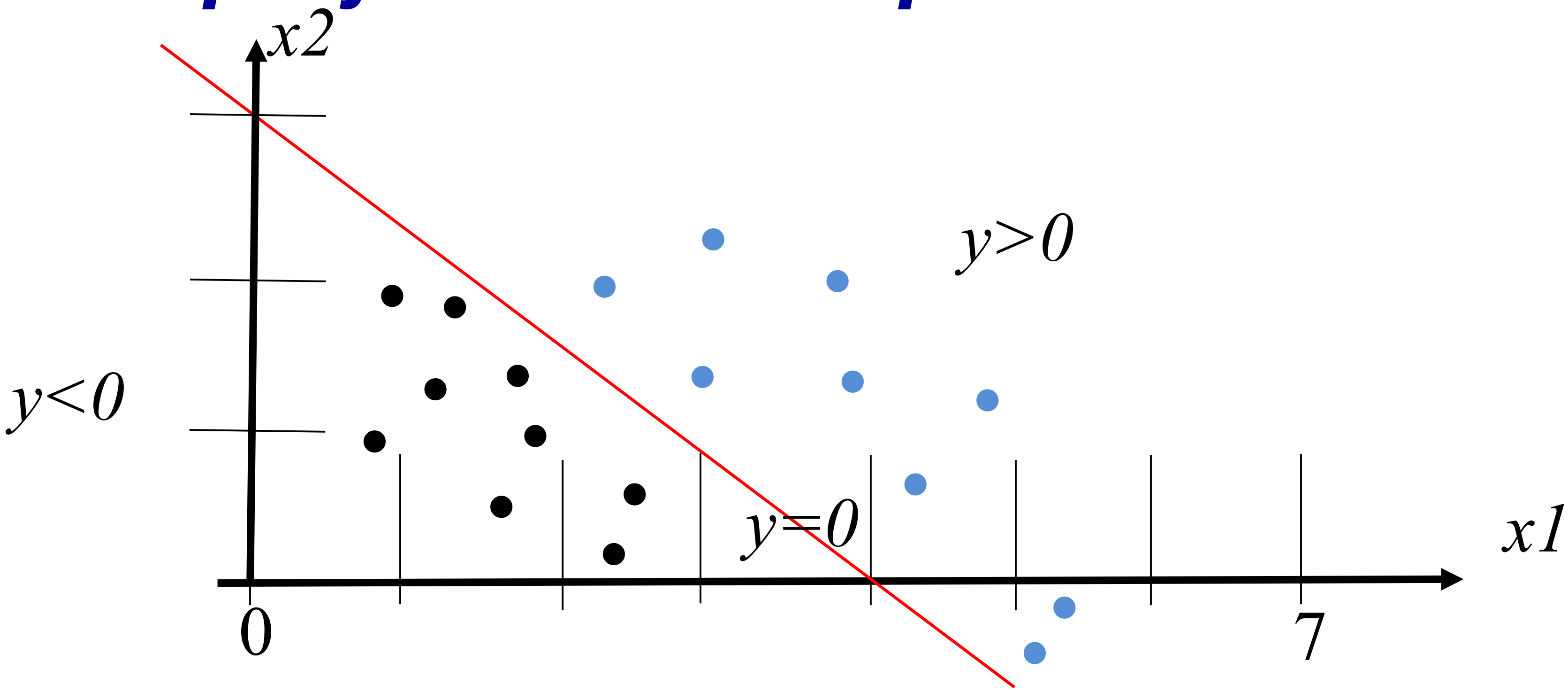
In two input dimensions:



The equation $y = wx1 + wx2 + b$ defines a plane in 3D (hyperplane)

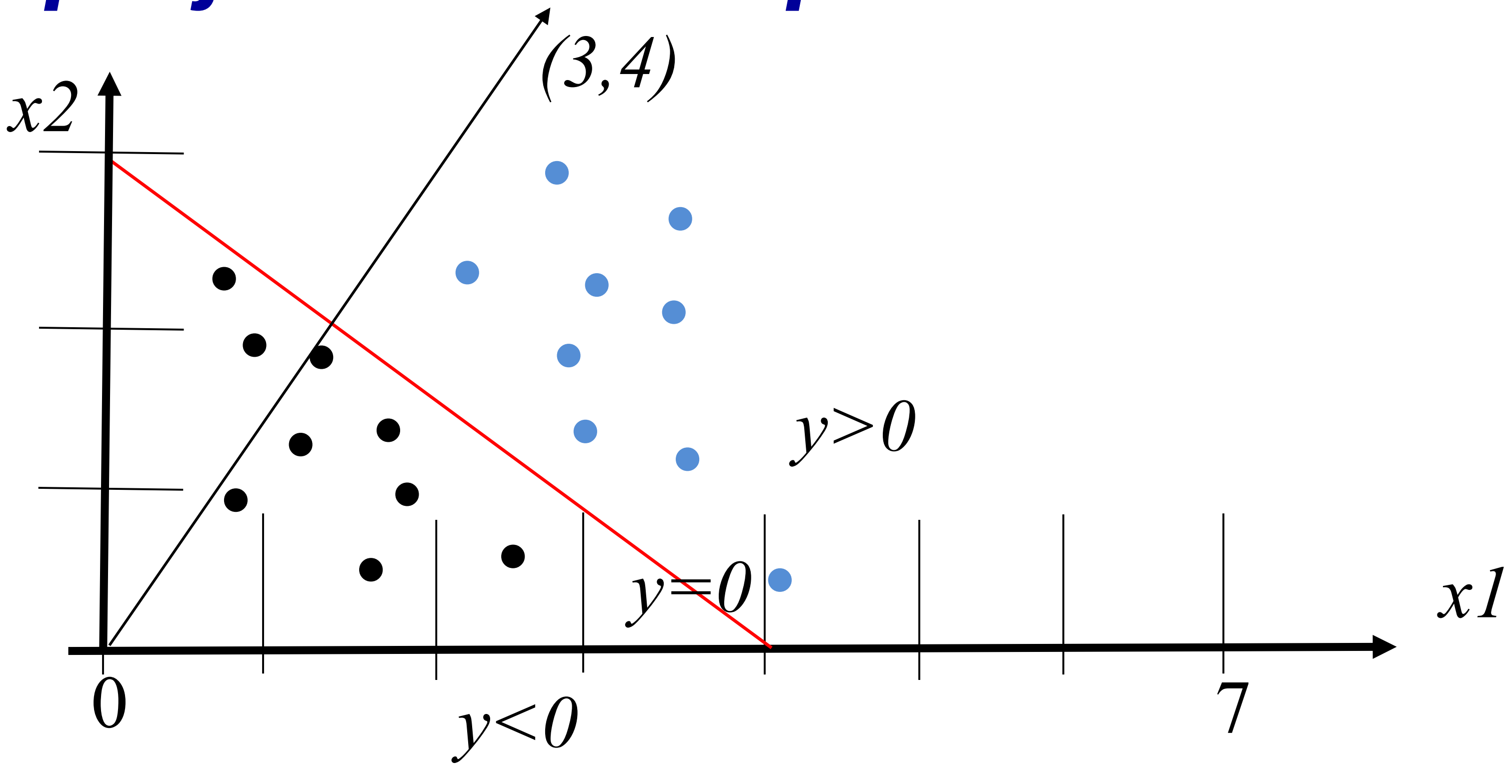
When y=0 it crosses the x1,x2 plane in a line.

project to two input dimensions



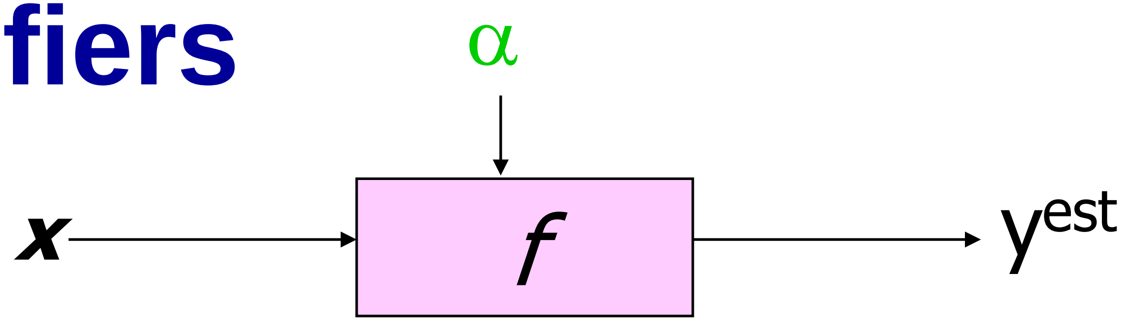
Note: This separating line in 2D is $x2=(-3/4)*x1+3$.
The equation for the output is $y= 4*x2 + 3*x1 – 12$
Is it unique?

project to two input dimensions



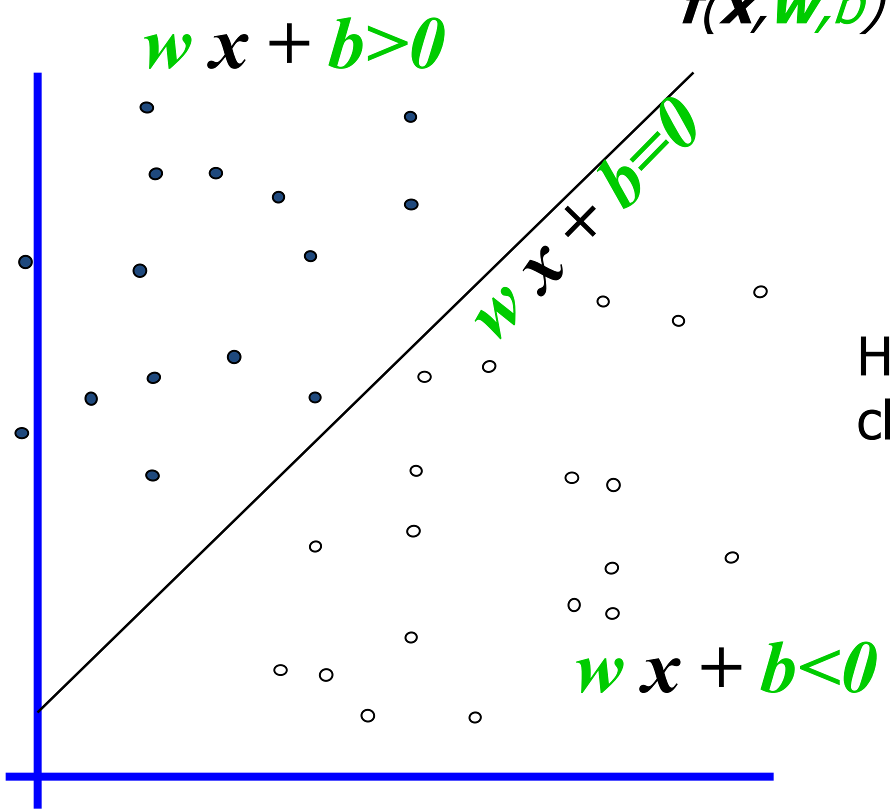
Note: The weights (w=*w1,w2)=(3,4) give* a vector in x1,x2 space.
What do you notice about it?

Linear Classifiers



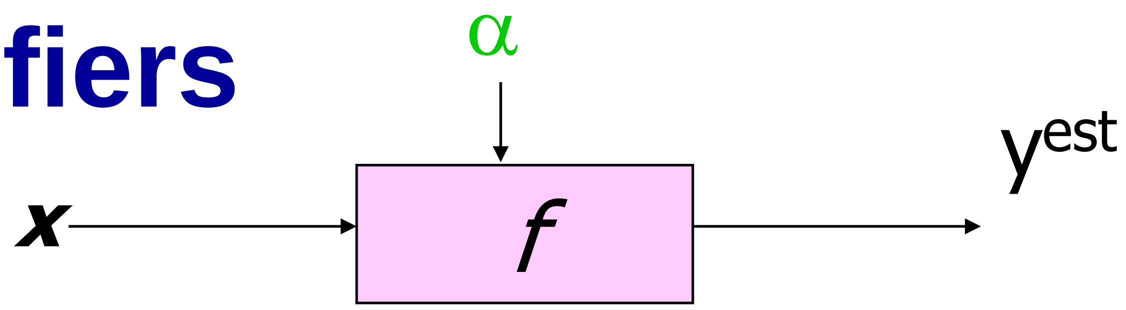
- • denotes +1
- ∘ denotes -1

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



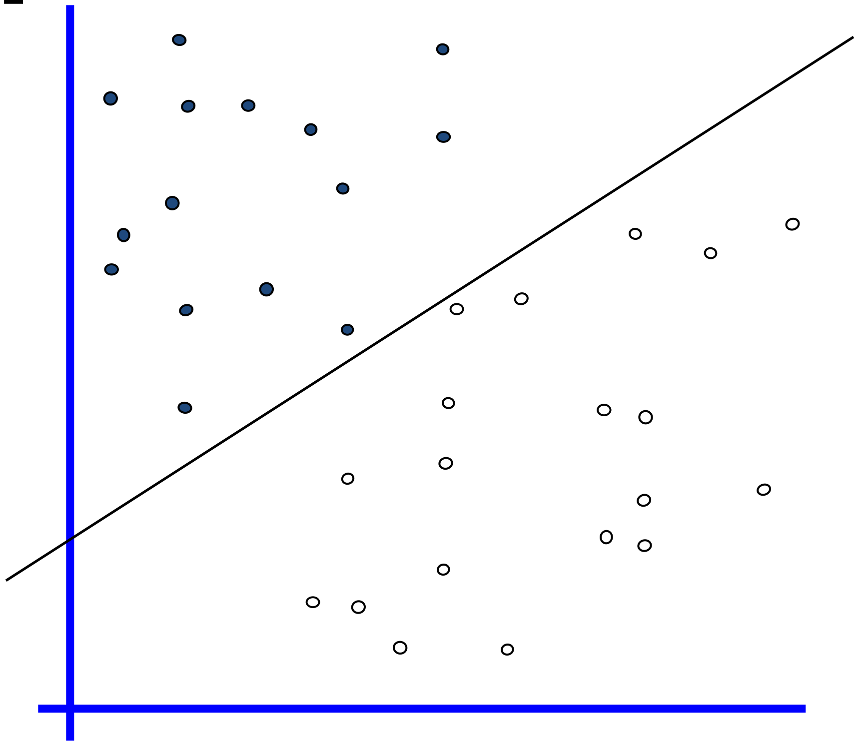
How would you classify this data?

Linear Classifiers



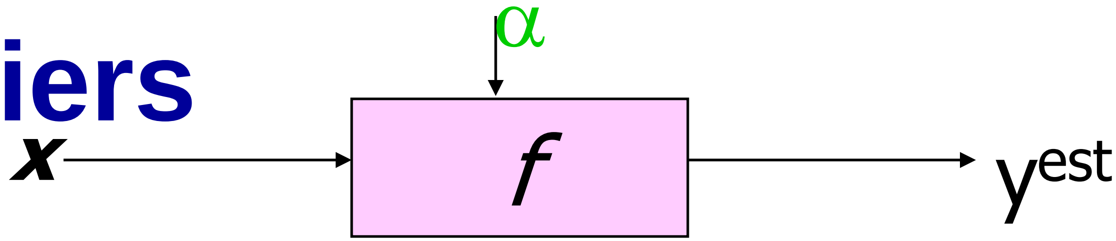
- denotes +1
- denotes -1

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



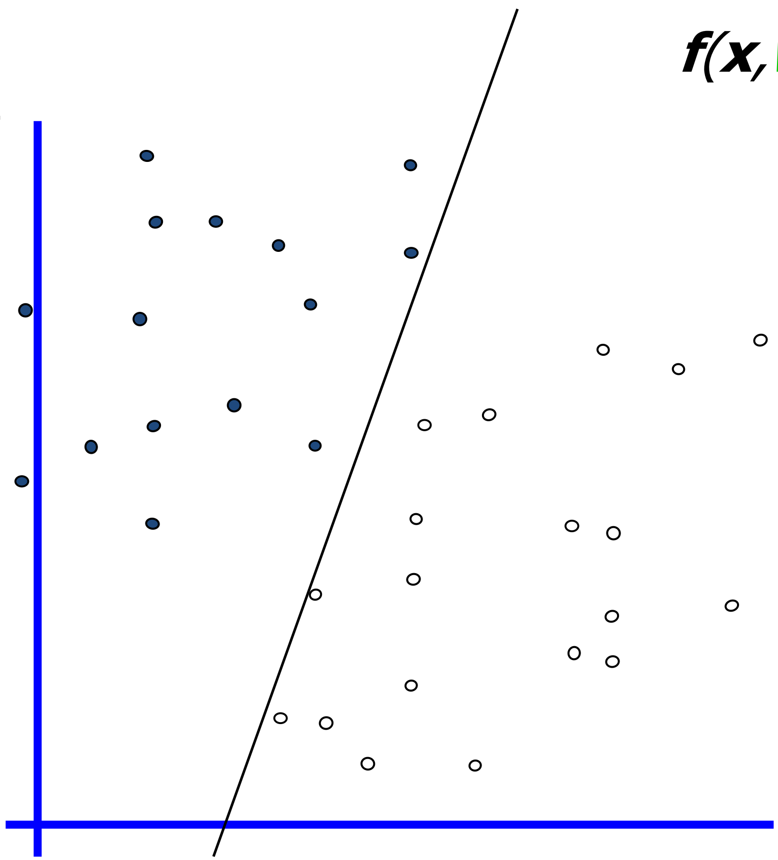
How would you classify this data?

Linear Classifiers



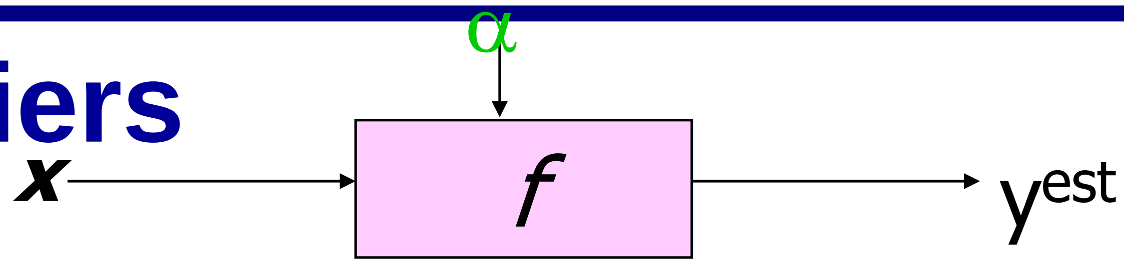
$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}\,\mathbf{x} + b)$

- • denotes +1
- ○ denotes -1



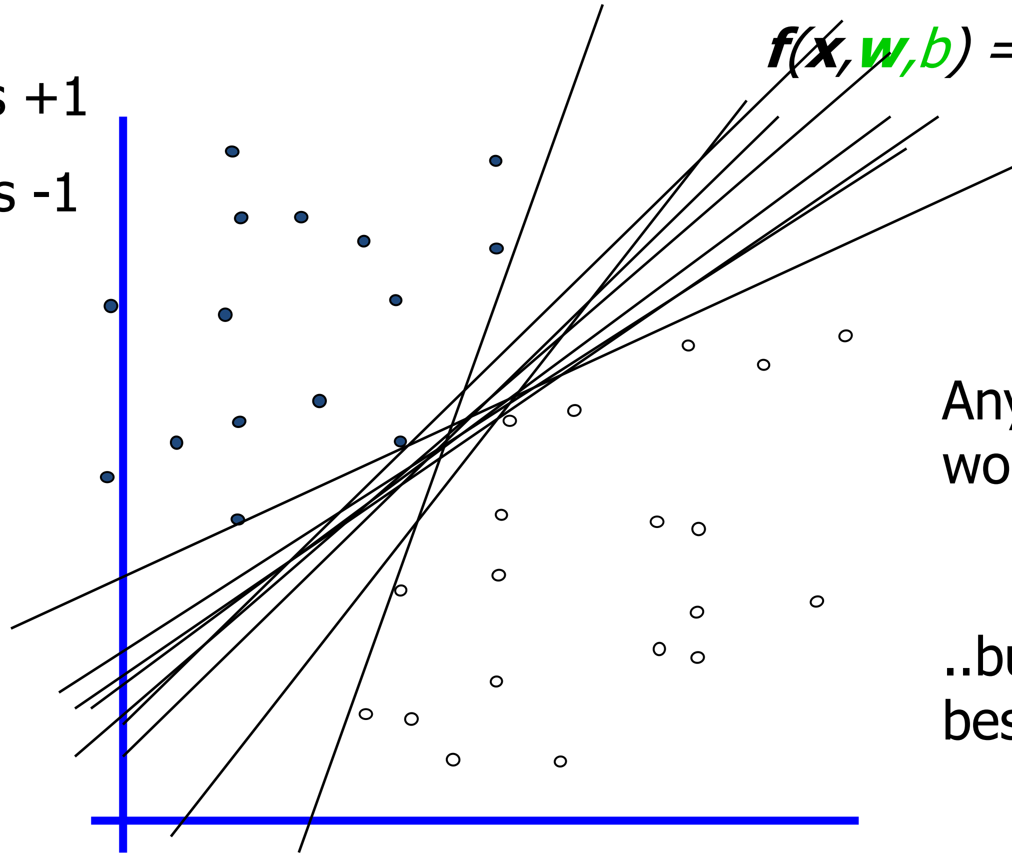
How would you classify this data?

Linear Classifiers



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

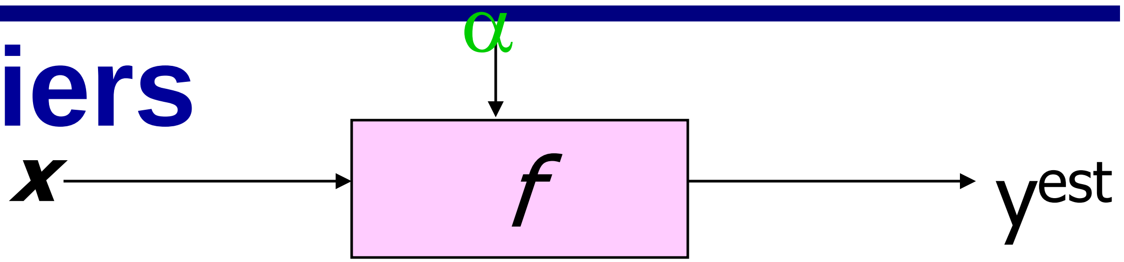
- denotes +1
- denotes -1



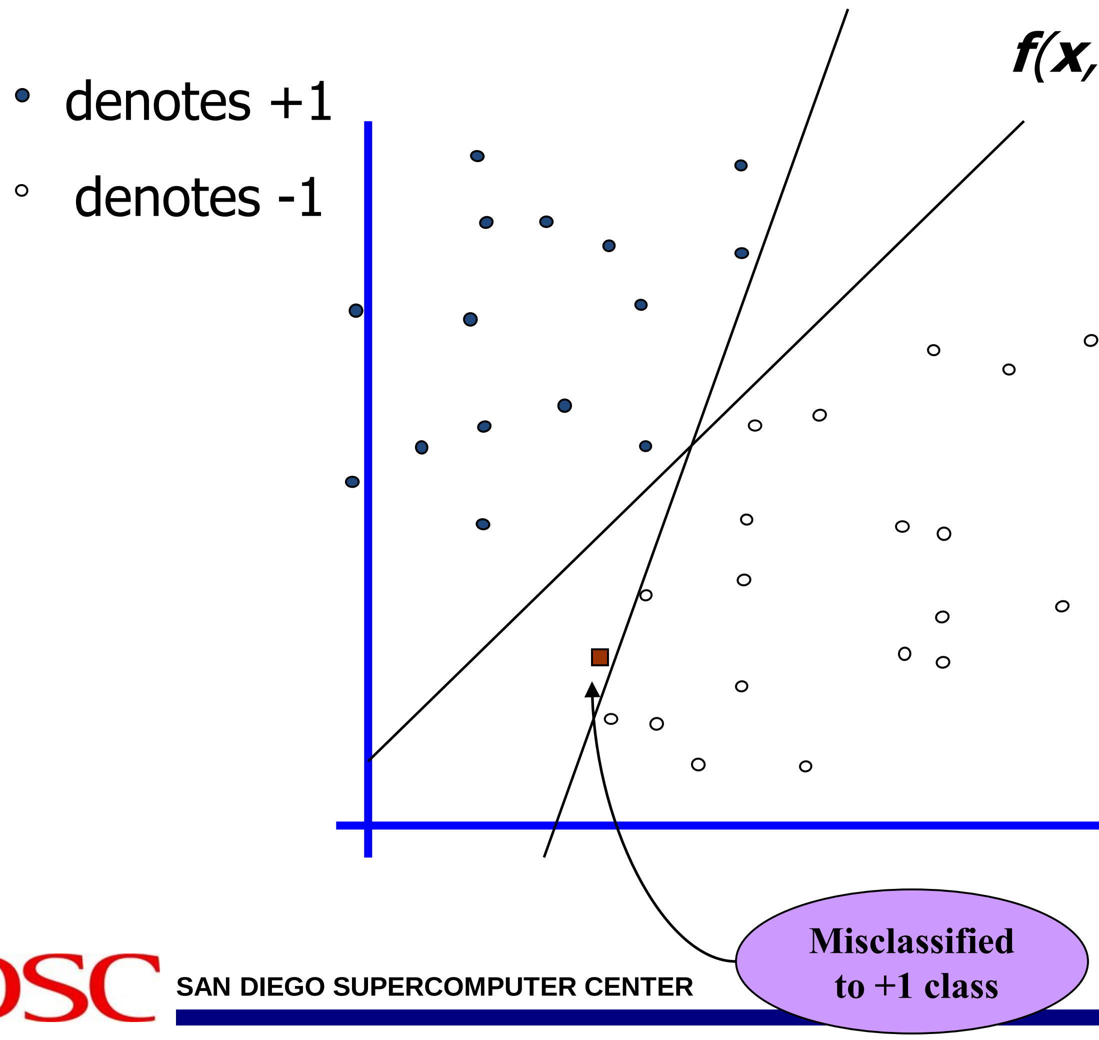
Any of these would be fine..

..but which is best?

Linear Classifiers

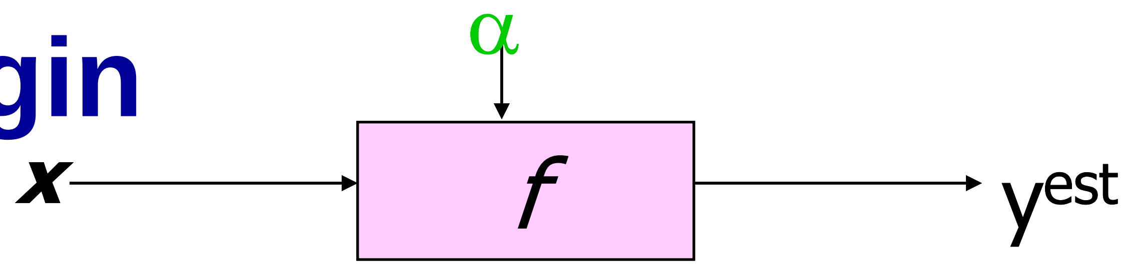


$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}\ \mathbf{x} + b)$

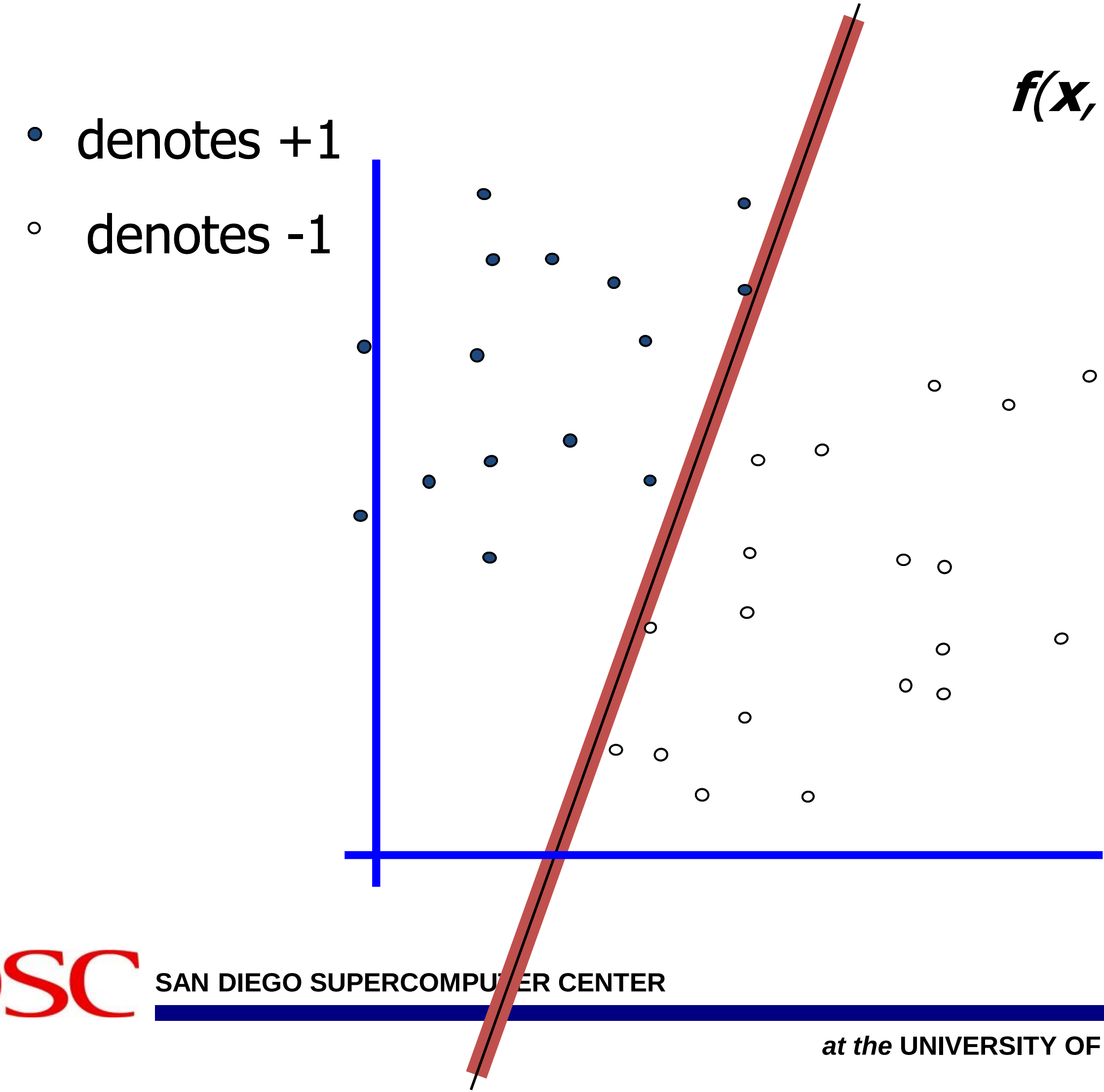


How would you classify this data?

Classifier Margin

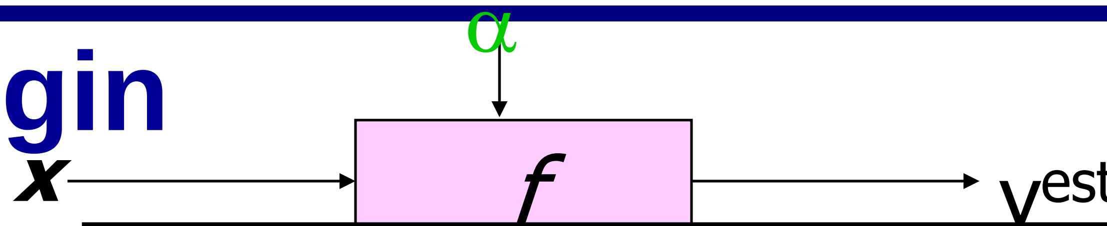


$$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}\,\mathbf{x} + b)$$



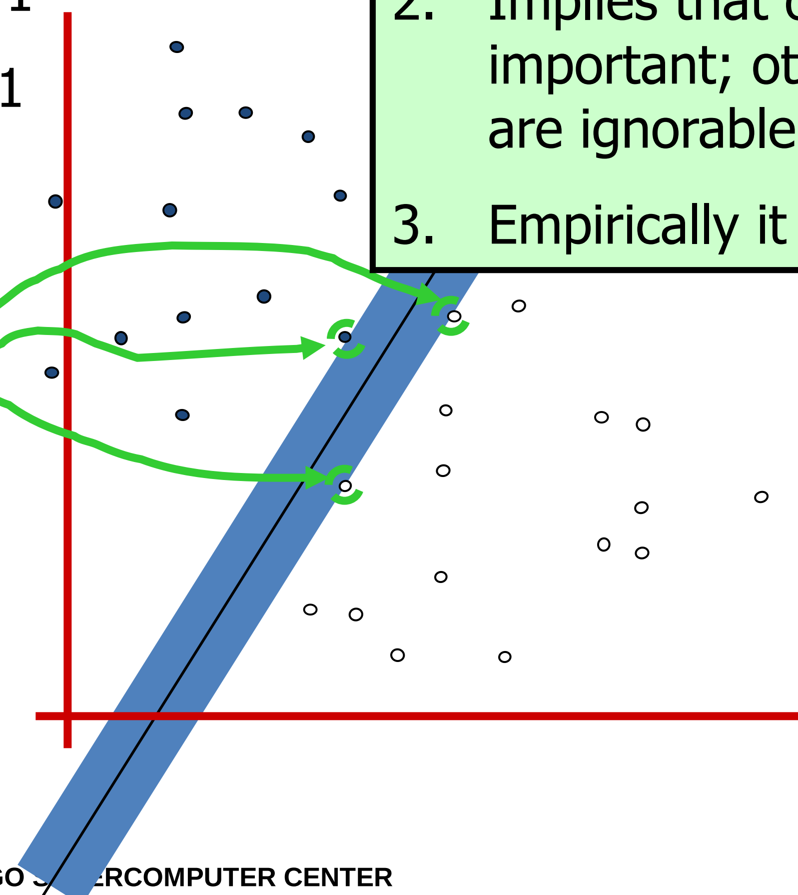
Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

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.

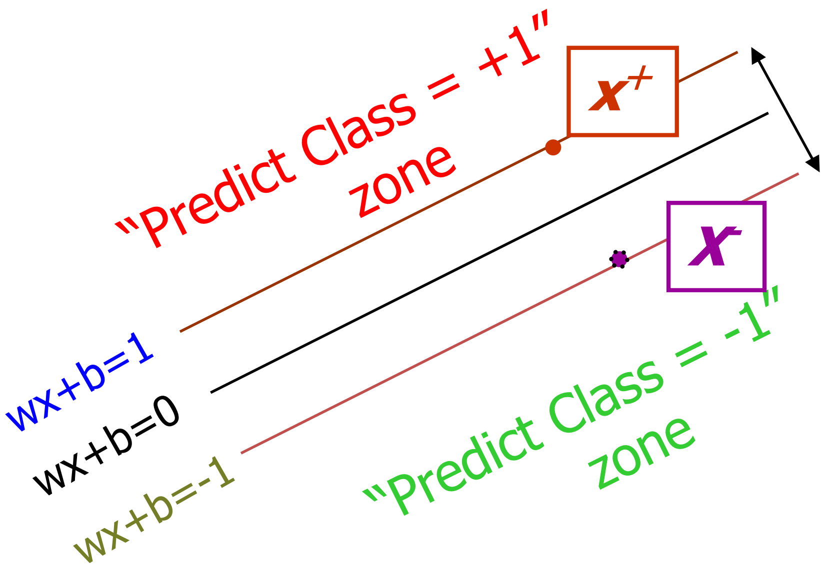
- denotes +1
- denotes -1



Support Vectors are those datapoints that the margin pushes up against

The maximum margin linear classifier is the simplest kind of SVM (Called an LSVM)

Linear SVM Mathematically



M=Margin Width

Given:

- ***w . x⁺ + b = +1***
- ***w . x⁻ + b = -1***
- ***w . (x⁺-x⁻)= 2***

$$M = \left| x^{+} - x^{-} \right| = \frac{2}{|w|}$$

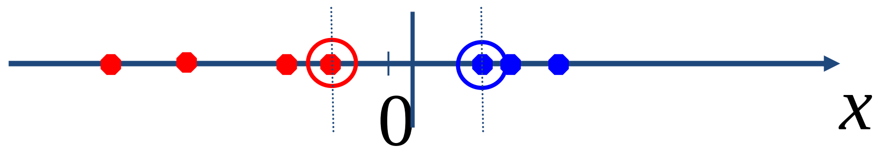
Length of Vector

To get max M, take min $|w|$ that works, which leads to

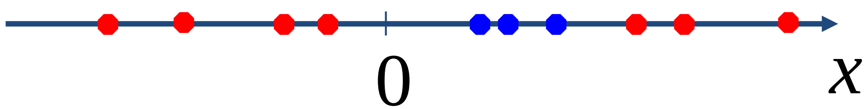
$$w = f(y=1 \text{ or } -1,\ x_s = \text{support vector set}) = \sum_s y^* x_s$$

Non-linear SVMs

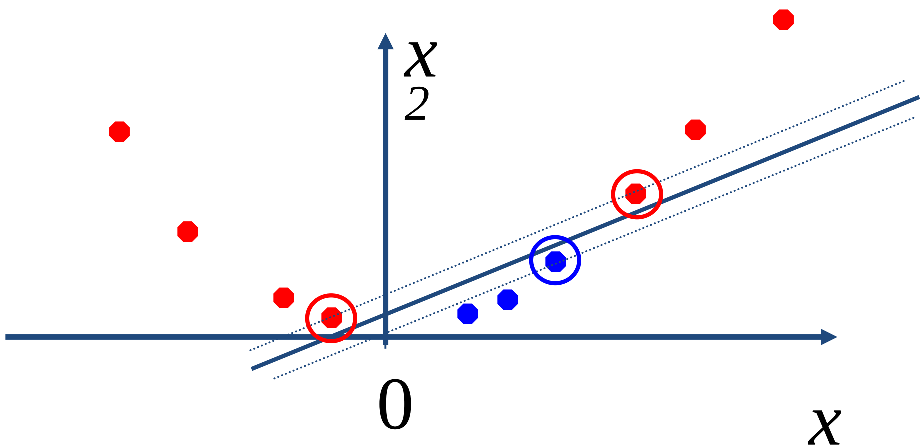
- Datasets that are linearly separable with some noise work out great:



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

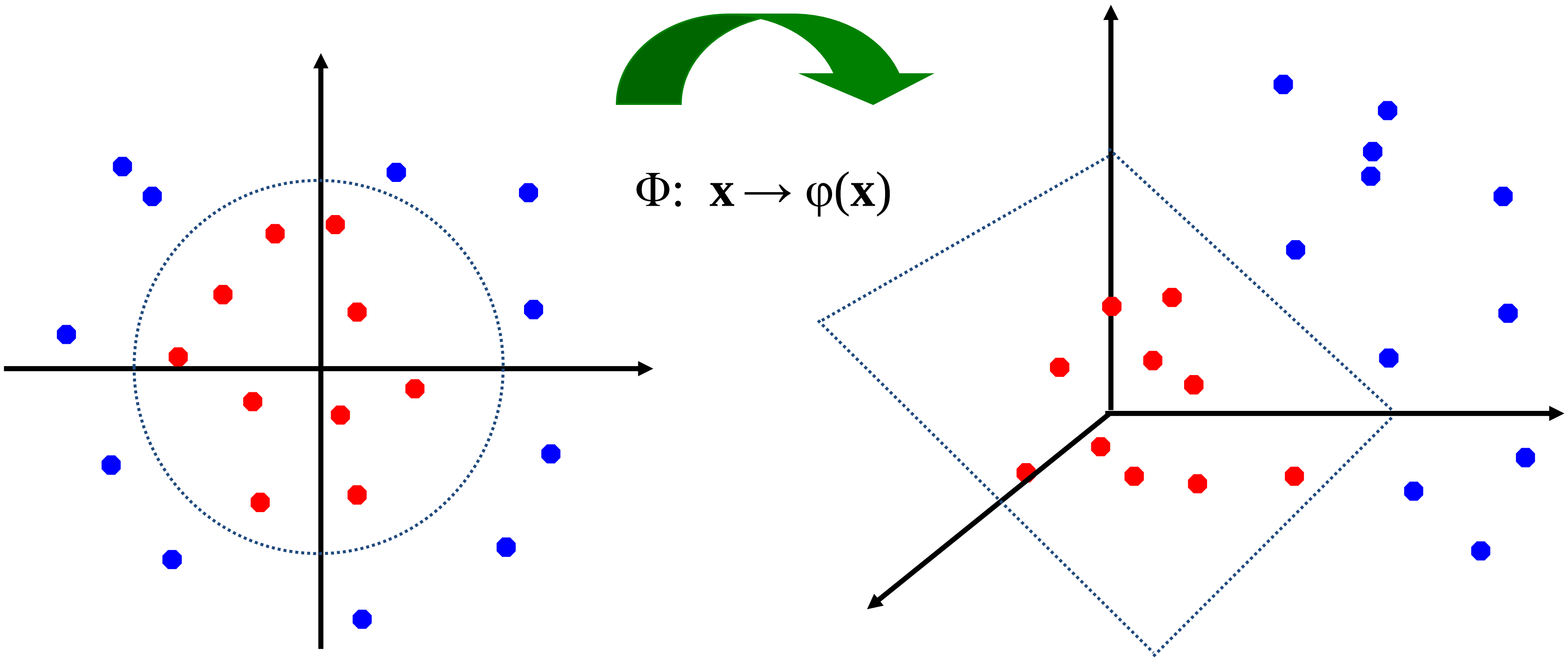


- How about… mapping data to a higher-dimensional space:



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"

- The kernel trick is a way of mapping observations from a general set S into an inner product space V, without having to compute the mapping explicitly.
- 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: $\mathrm{x} \longrightarrow \varphi(\mathrm{x})$, the dot product becomes:

$$K(\mathbf{x_i}, \mathbf{x_j}) = f(\mathbf{x_i})^T f(\mathbf{x}_j)$$

- Think of a kernel function as giving some kind of measure of similarity or covariance in the high dimensional space

The Kernel Trick

- The direct way: transform and then get dot product

Compute the Mapping
$\Phi: \mathbf{x} \rightarrow \varphi(\mathbf{x})$
→
Calculate the inner Product
$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$
→
Linear Mapping

- The trick: write equations so that $\boldsymbol{\varphi}(\mathbf{x})$ is always multiplied by another $\boldsymbol{\varphi}(\mathbf{x})$, then just take Kernel (*w* never computed)

$$w*x+b \Rightarrow$$
$$(\sum y*x_s)*x+b$$
$$=\sum y*K(x_s,x)+b$$

→
Calculate the inner Product
$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$
→
Linear Mapping

Examples of Kernel Functions

- Linear: $K(\mathbf{x_i}, \mathbf{x_j}) = \mathbf{x_i}^T \mathbf{x_j}$

- Polynomial of power *p*: $K(\mathbf{x_i}, \mathbf{x_j}) = (1 + \mathbf{x_i}^T \mathbf{x_j})^P$

- Gaussian (radial-basis function network):

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

- Sigmoid: $K(\mathbf{x_i}, \mathbf{x_j}) = \tanh(b_0 \mathbf{x_i}^T \mathbf{x_j} + b_1)$

Note: these have parameters to choose!

Nonlinear SVMs

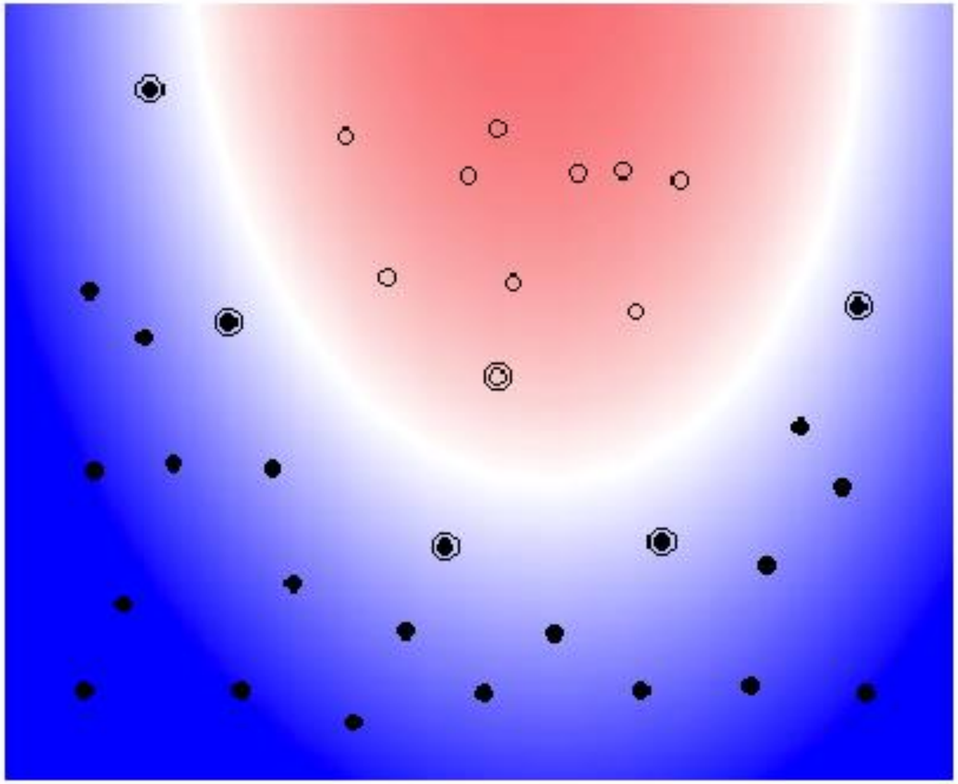
- **Over-fitting is unlikely to occur because maximum margin hyper-plane is stable**
  - There are usually few support vectors relative to the size of the training set
- **Computation time still an issue**
  - Every time an instance is classified it's dot product with all support vectors must be calculated

Nonlinear Kernel (I)

Example: SVM with Polynomial of Degree 2

Kernel: $K(\vec{x}_i, \vec{x}_j) = [\vec{x}_i \cdot \vec{x}_j + 1]^2$

plot by Bell SVM applet

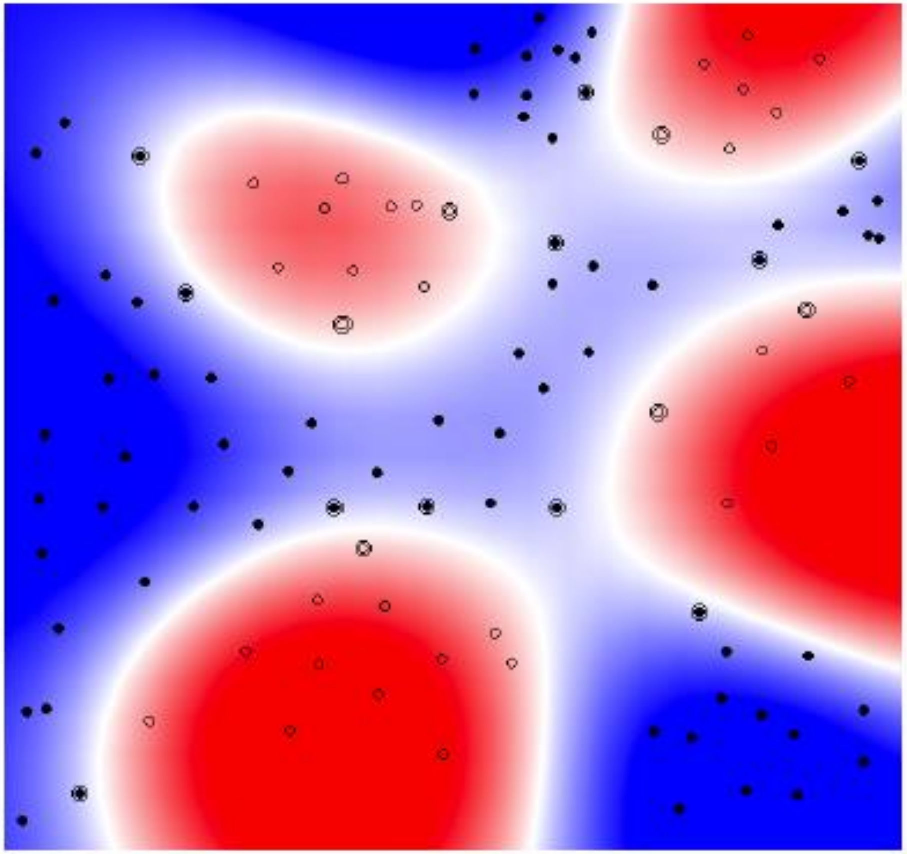


Nonlinear Kernel (II)

Example: SVM with RBF-Kernel

Kernel: $K(\vec{x}_i, \vec{x}_j) = \exp(-|\vec{x}_i - \vec{x}_j|^2 / \sigma^2)$

plot by Bell SVM applet



SVM Applications

- **SVM has been used successfully in many real-world problems**
  - Text (and hypertext) categorization
  - Image classification
  - Bioinformatics (Protein classification, Cancer classification)
  - Hand-written character recognition

SVM 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**

Representation of Text

IR's vector space model (aka bag-of-words representation)

- A doc is represented by a vector indexed by a pre-fixed set or dictionary of terms
- Values of an entry can be binary or weights

$$\phi_i(x) = \frac{\mathrm{tf}_i \log(\mathrm{idf}_i)}{\kappa},$$

- Normalization, stop words, word stems
- Doc x => φ(x)

Text Categorization using SVM

- **The distance between two documents is $\varphi(x)\cdot\varphi(z)$**
  - $K(x,z) = \varphi(x)\cdot\varphi(z)$ is a valid kernel, SVM can be used with $K(x,z)$ for discrimination

- **Why SVM?**
  - High dimensional input space
  - Few irrelevant features (dense concept)
  - Sparse document vectors (sparse instances)
  - Text categorization problems are linearly separable

Conclusion

SVM Strengths

- Kernel trick mitigates complexity/capacity for high dimensional data
- Finding the weights is a quadratic programming problem - guaranteed to find a minimum of the error surface
- Can obtain good generalization performance due to maximum margin in non-linear space, even w/ small training sets

- **Weakness of SVM**
  - It is sensitive to noise
    - A relatively small number of mislabeled examples can dramatically decrease the performance
  - Multiclass SVMs work pairwise (one class vs the rest)

Summary

- The SVM was proposed in the 70’s
- It become popular in 90’s
- Kernel methods and SVM extensions still a topic of research

More information:

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

AN INTRODUCTION TO SUPPORT VECTOR MACHINES (and other kernel-based learning methods). N. Cristianini and J. Shawe-Taylor, Cambridge University Press. 2000. ISBN: 0 521 78019 5