

# Support Vector Machines

Wee Hyong Tok

May 17 2017

# Support vector machines

Similar to linear regression they aim to find a hyperplane that linearly separates data points belong to different classes

In addition SVMs aim to find the hyperplane that is least likely to over fit the training data

- By design, similar to pruning in Decision Trees, SVMs attempt to regulate the hypothesis space to ensure good performance on validation set...
- Fast in the nonlinear case
  - Use a mathematical trick to avoid creating “pseudoattributes”
  - The nonlinear space is created implicitly

# SVM, in a nutshell...

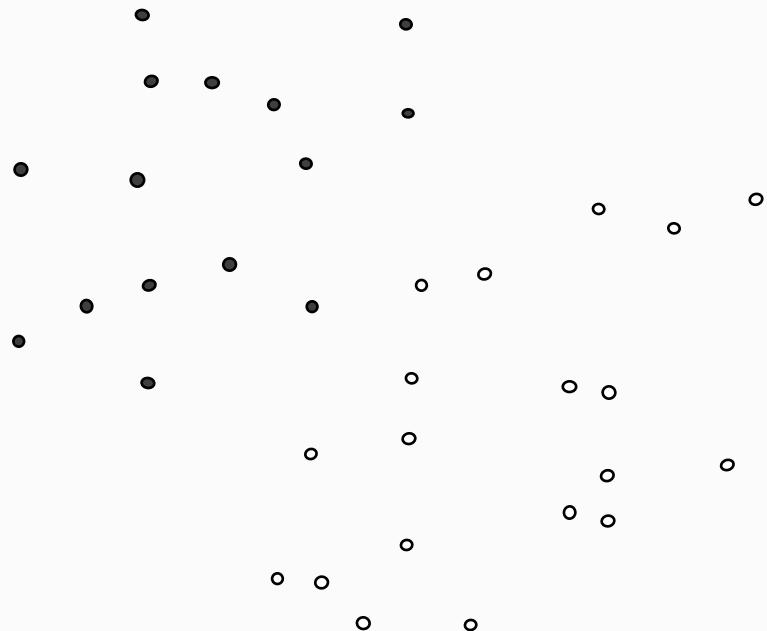
- SVM views the input data as two sets of vectors in an n-dimensional space. It constructs a separating hyperplane in that space, one which maximizes the margin between the two data sets.
- To calculate the margin, two parallel hyperplanes are constructed, one on each side of the separating hyperplane.
- A good separation is achieved by the hyperplane that has the largest distance to the neighboring data points of both classes.
- The vectors (points) that constrain the width of the margin are the support vectors.

# Linear Classifiers

Estimation:



- denotes +1
- denotes -1

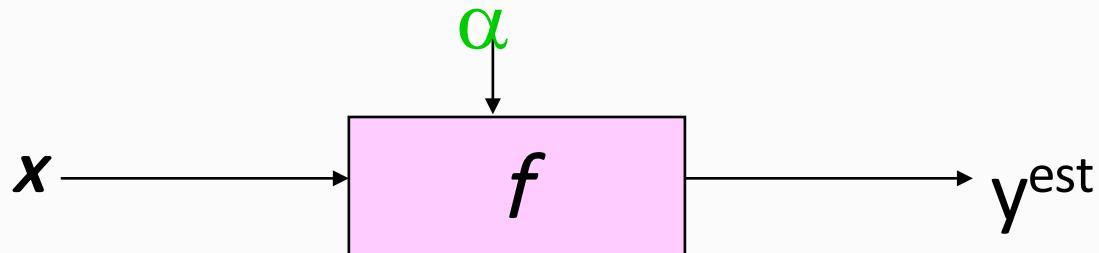


$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \mathbf{x} - b)$$

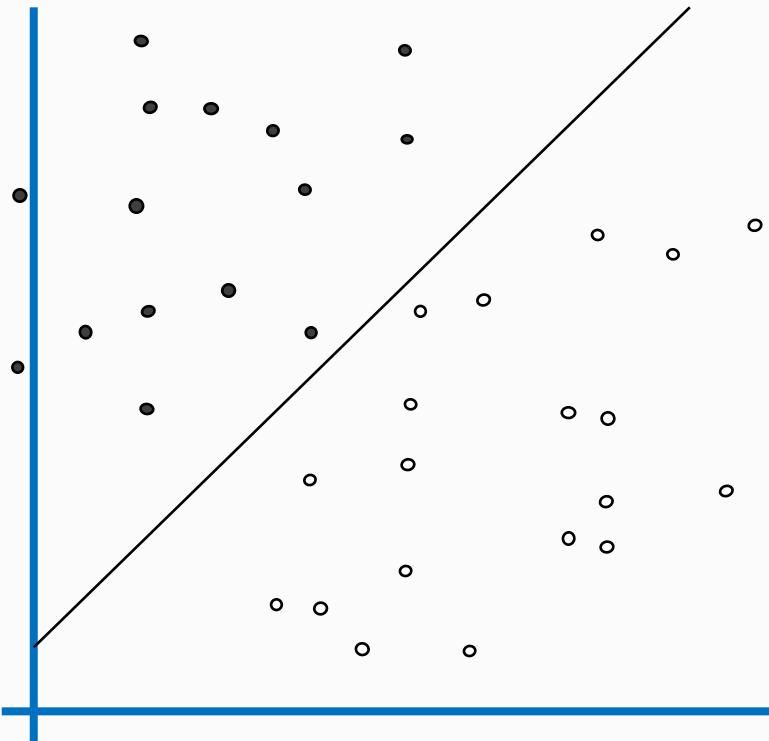
w: weight vector  
x: data vector

How would you  
classify this data?

# Linear Classifiers



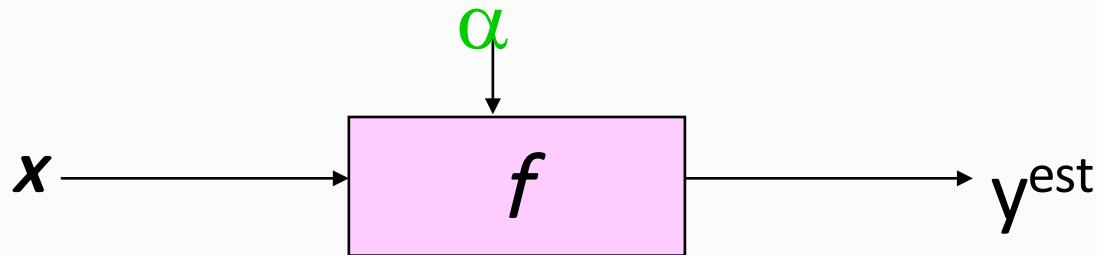
- denotes +1
- denotes -1



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

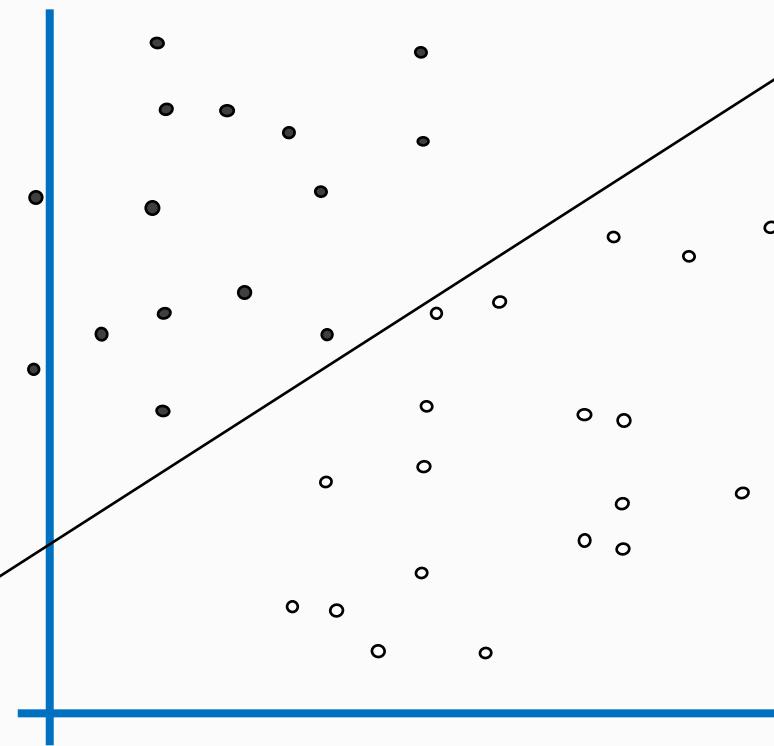
How would you  
classify this data?

# Linear Classifiers



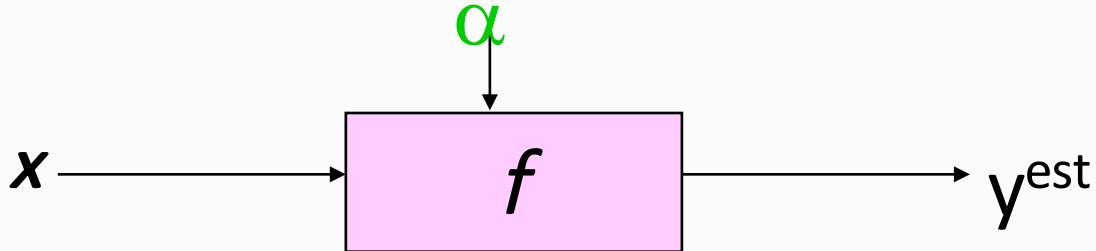
- denotes +1
- denotes -1

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

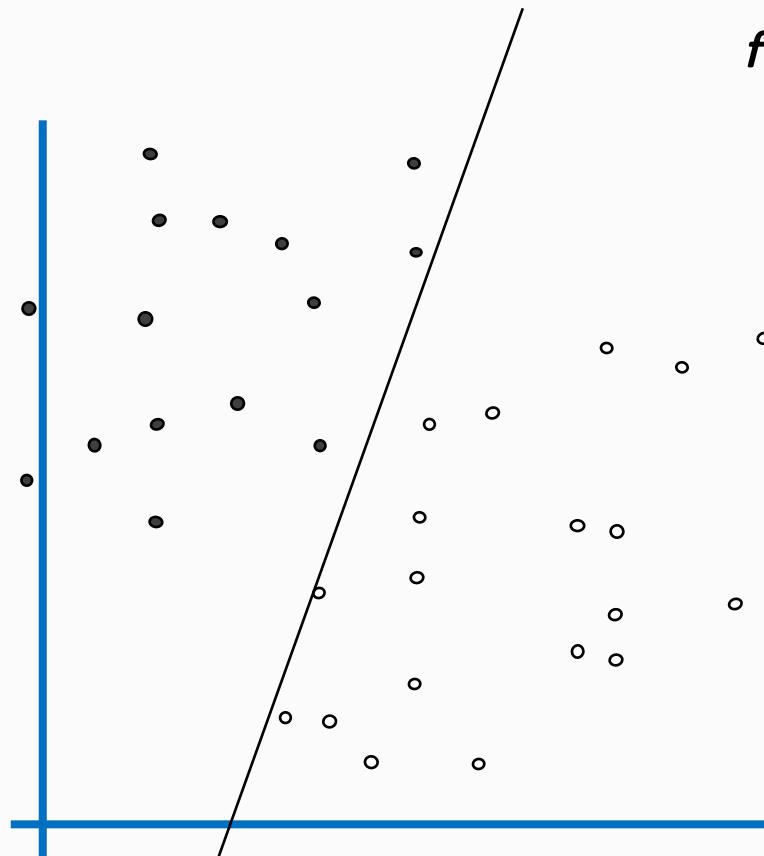


How would you  
classify this data?

# Linear Classifiers



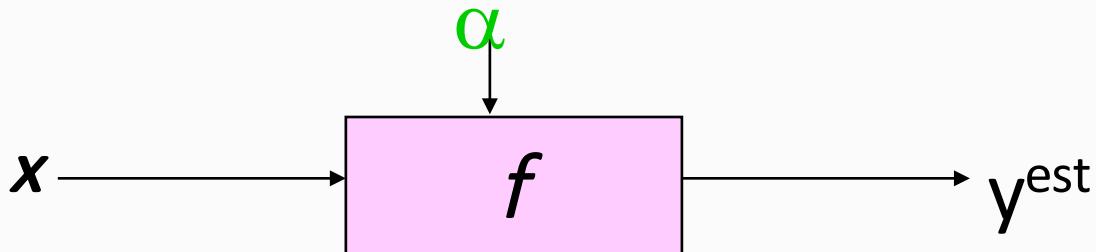
- denotes +1
- denotes -1



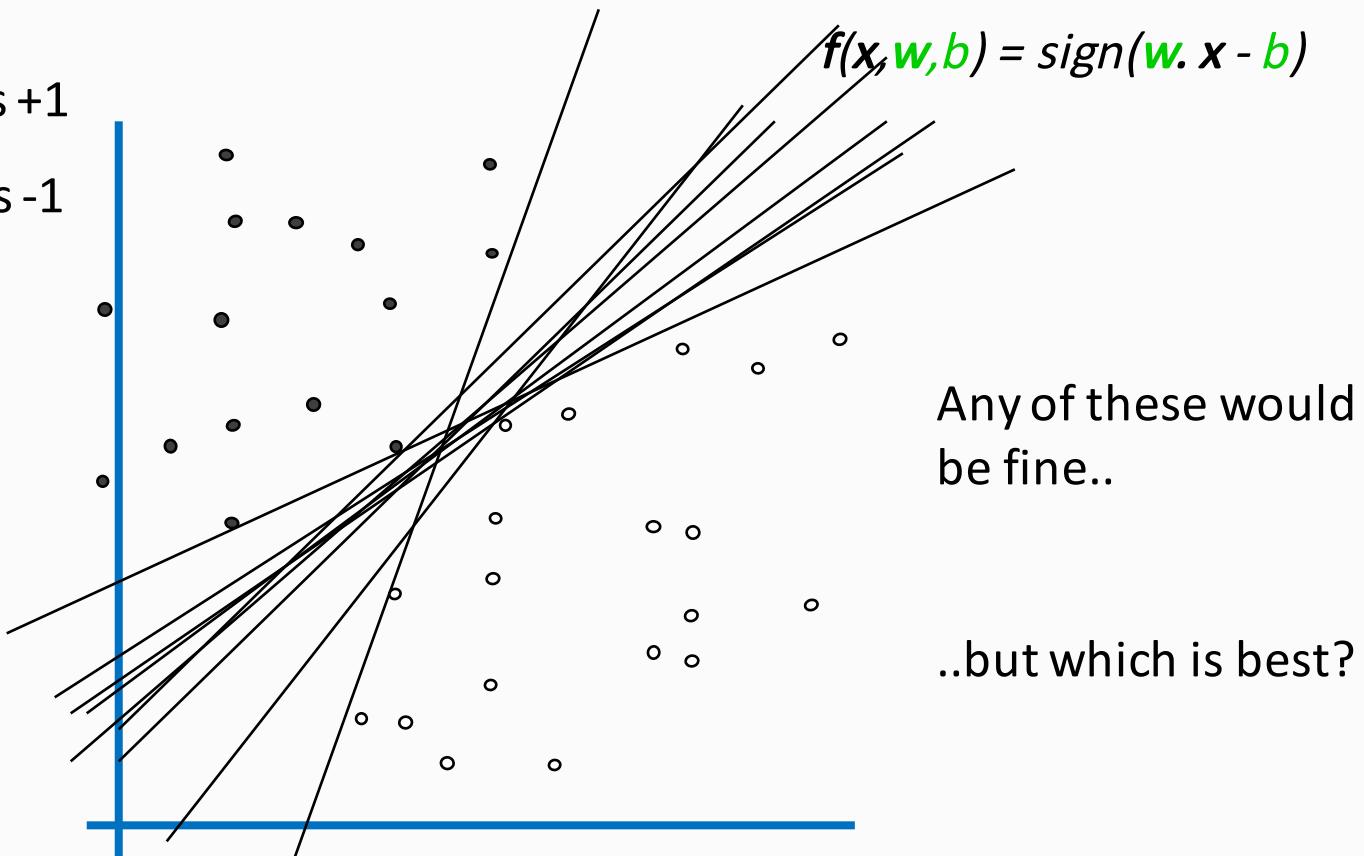
$$f(x, w, b) = \text{sign}(w \cdot x - b)$$

How would you  
classify this data?

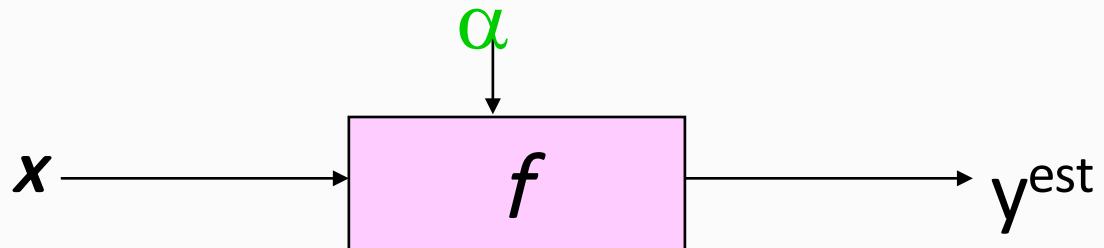
# Linear Classifiers



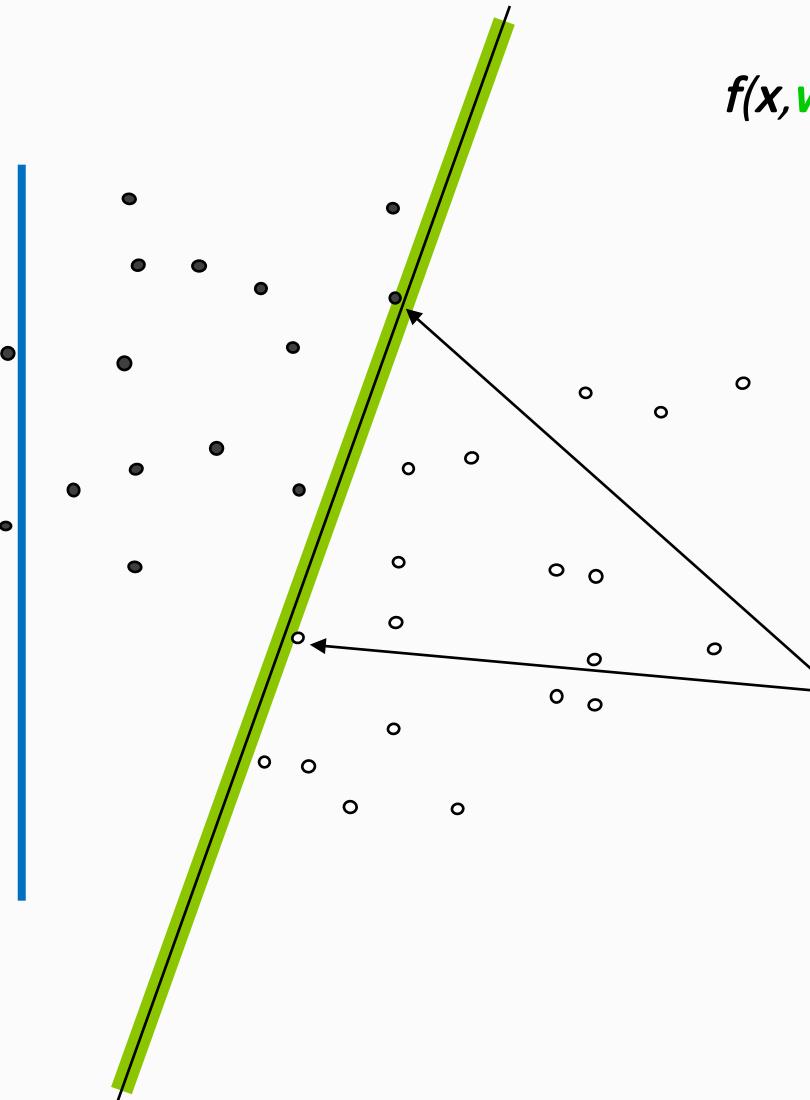
- denotes +1
- denotes -1



# Classifier Margin



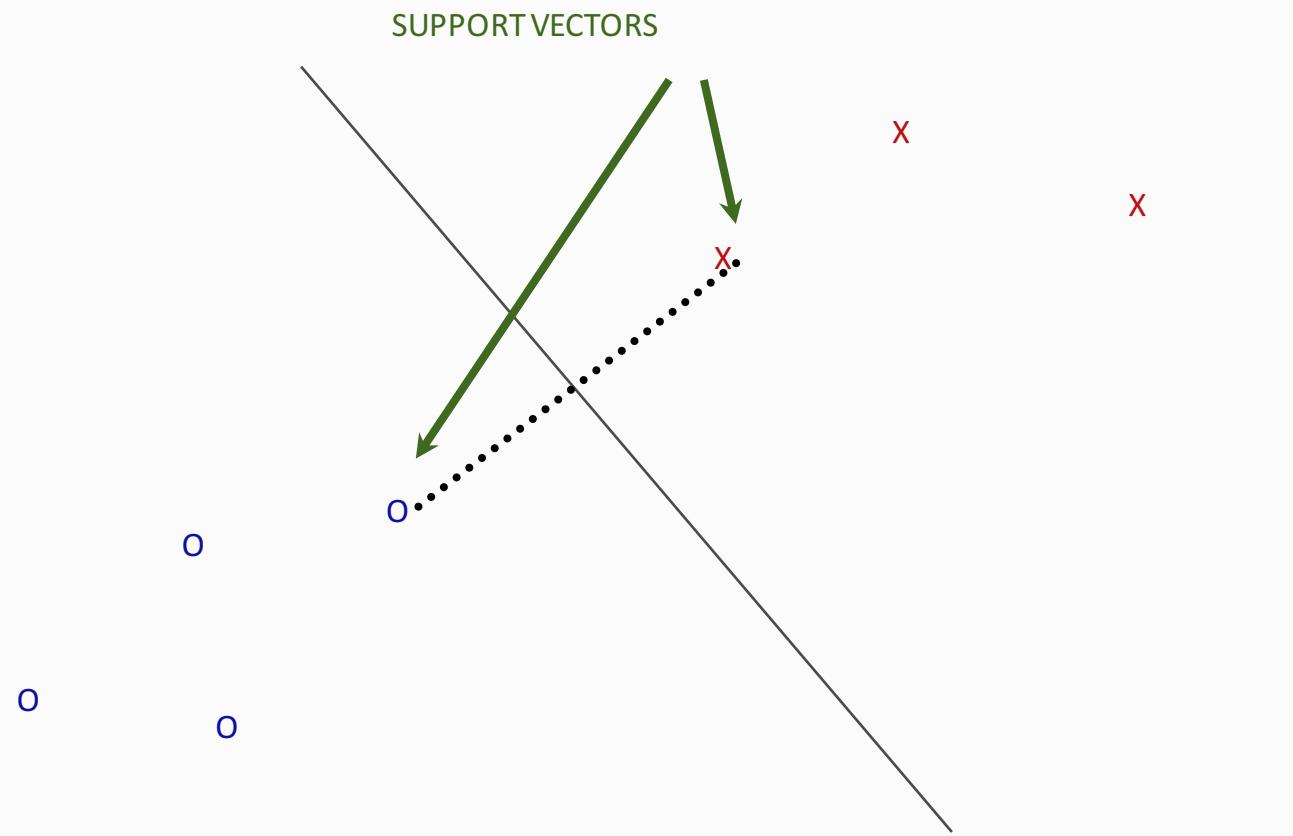
- denotes +1
- denotes -1



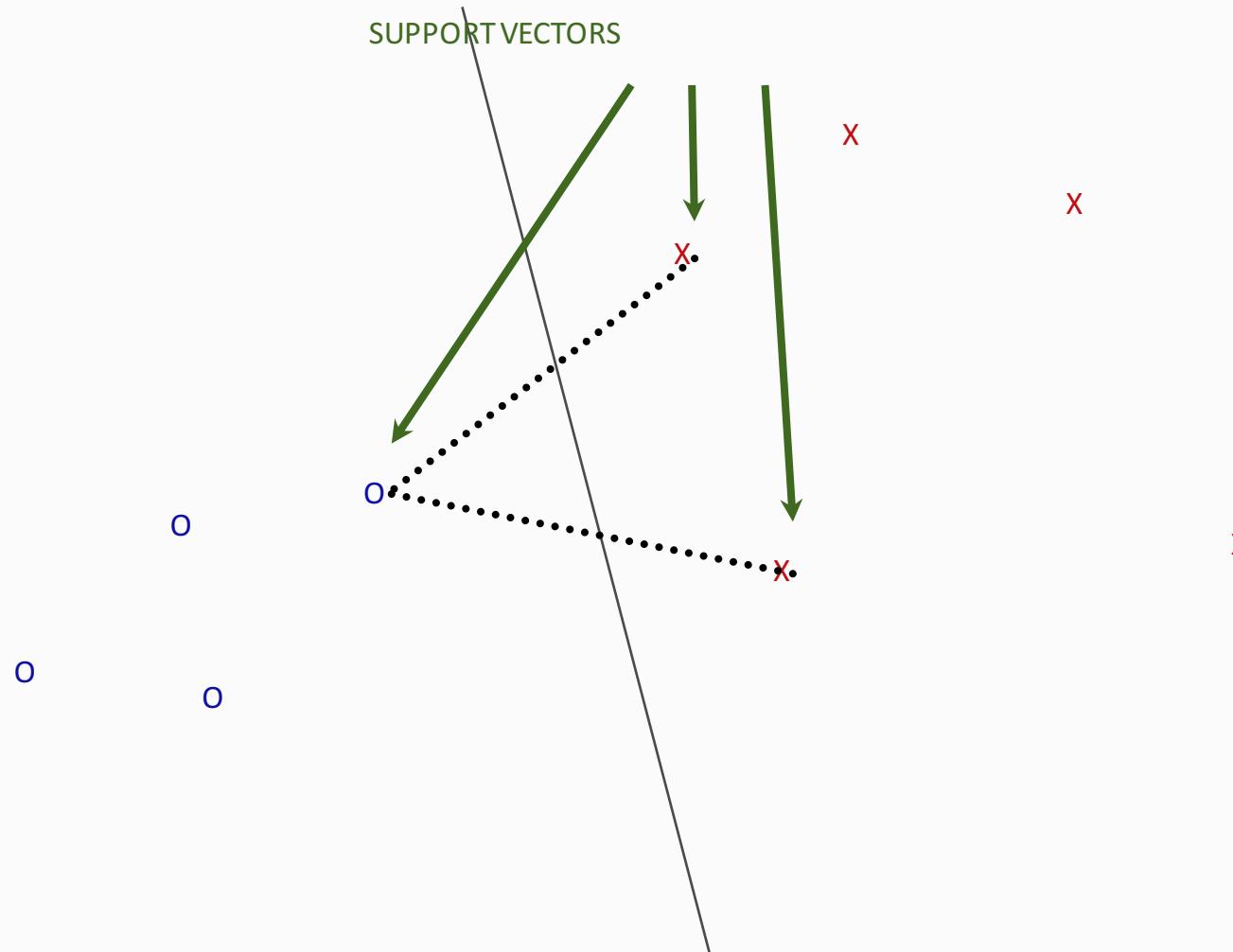
$$f(x, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \mathbf{x} - b)$$

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

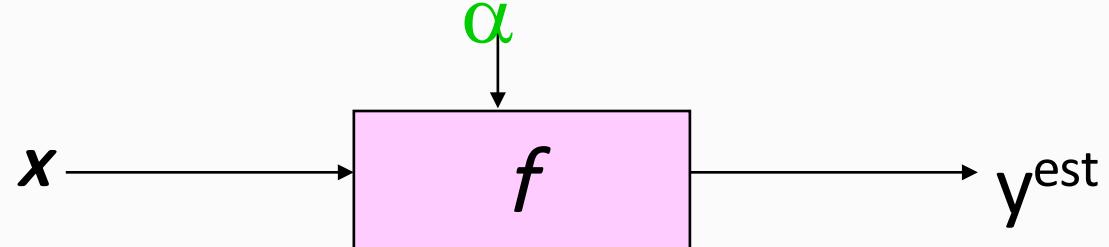
# Geometric Intuition



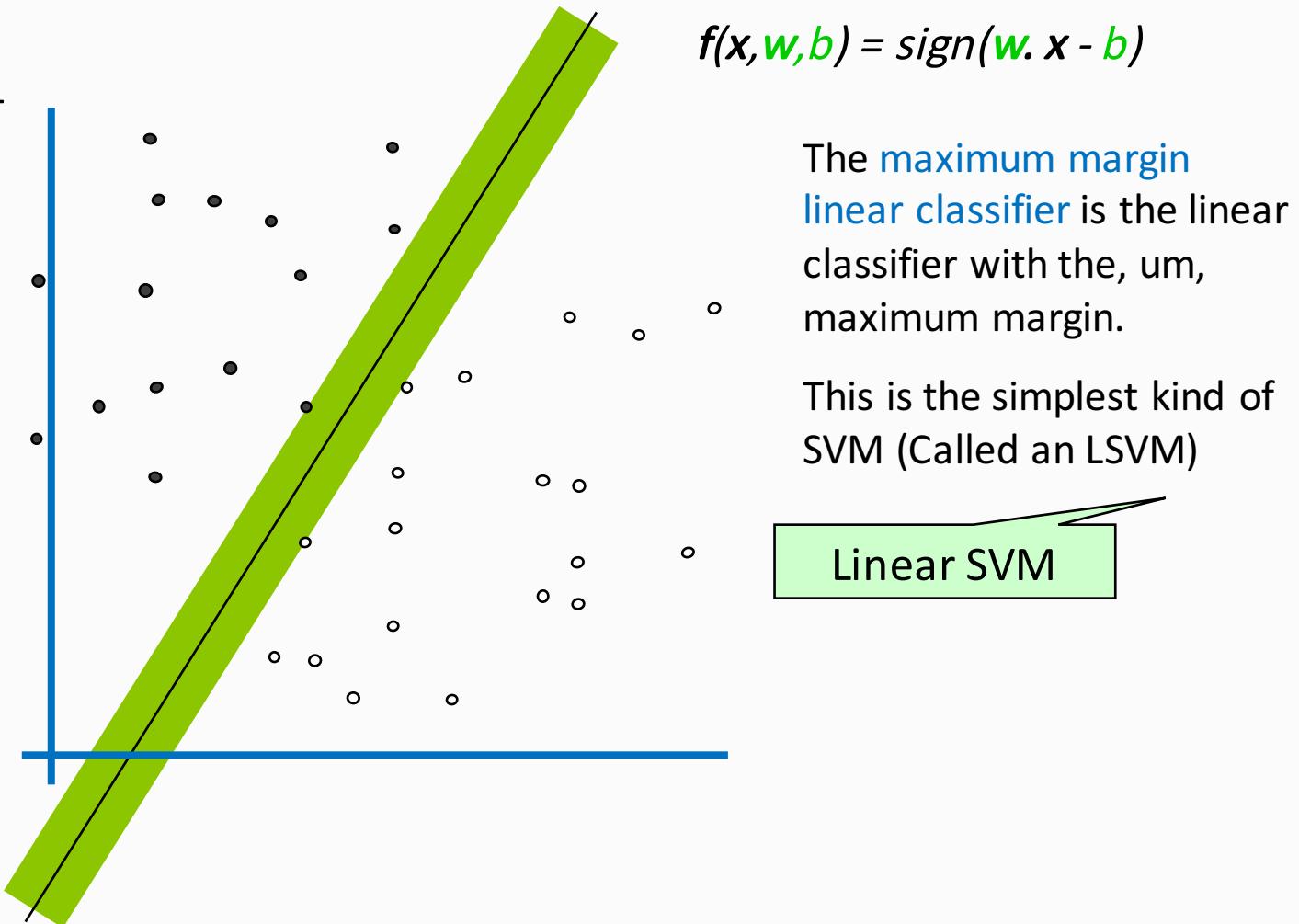
# Geometric Intuition



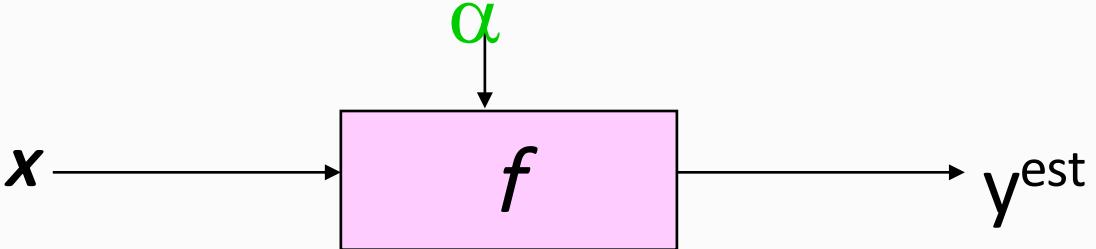
# Maximum Margin



- denotes +1
- denotes -1

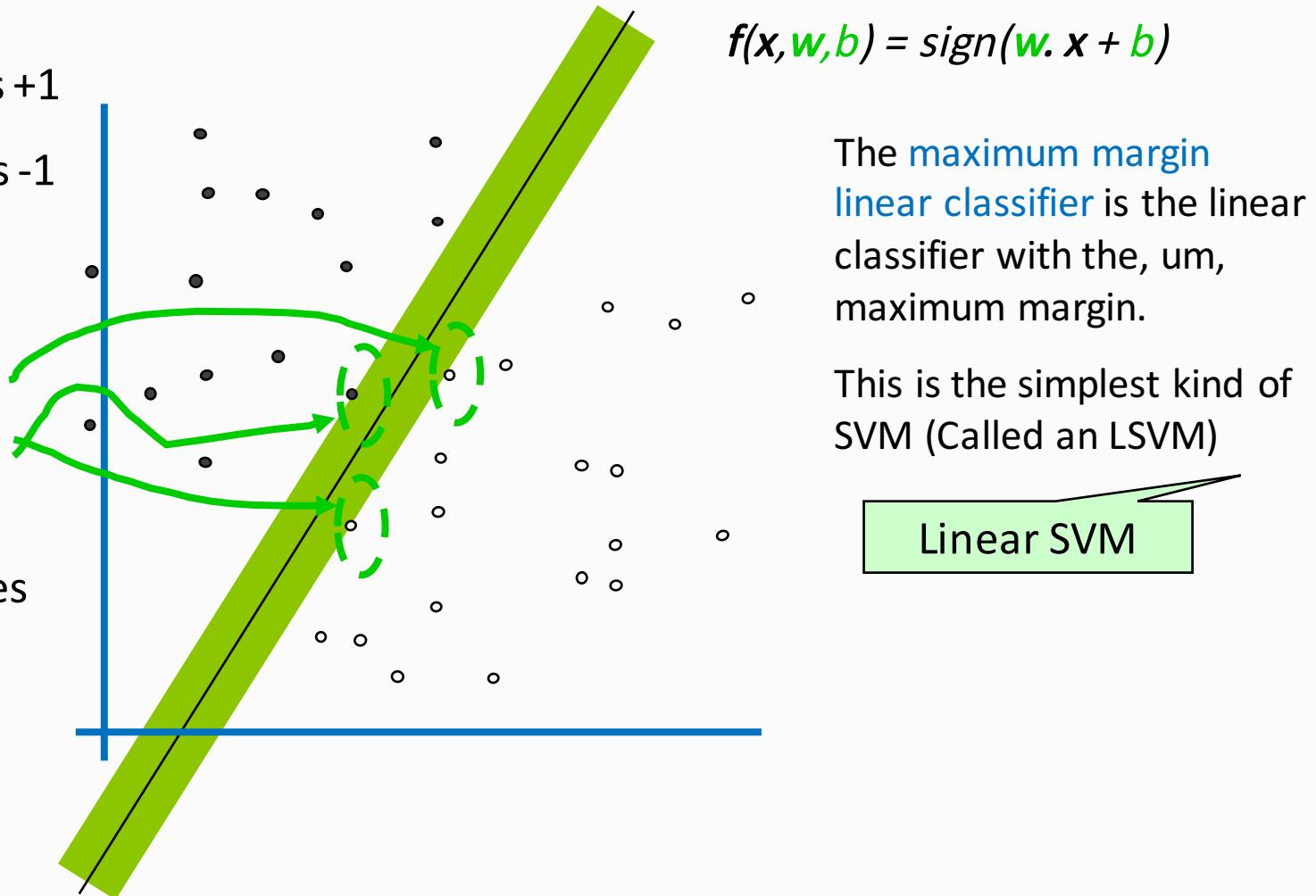


# Maximum Margin



- denotes +1
- denotes -1

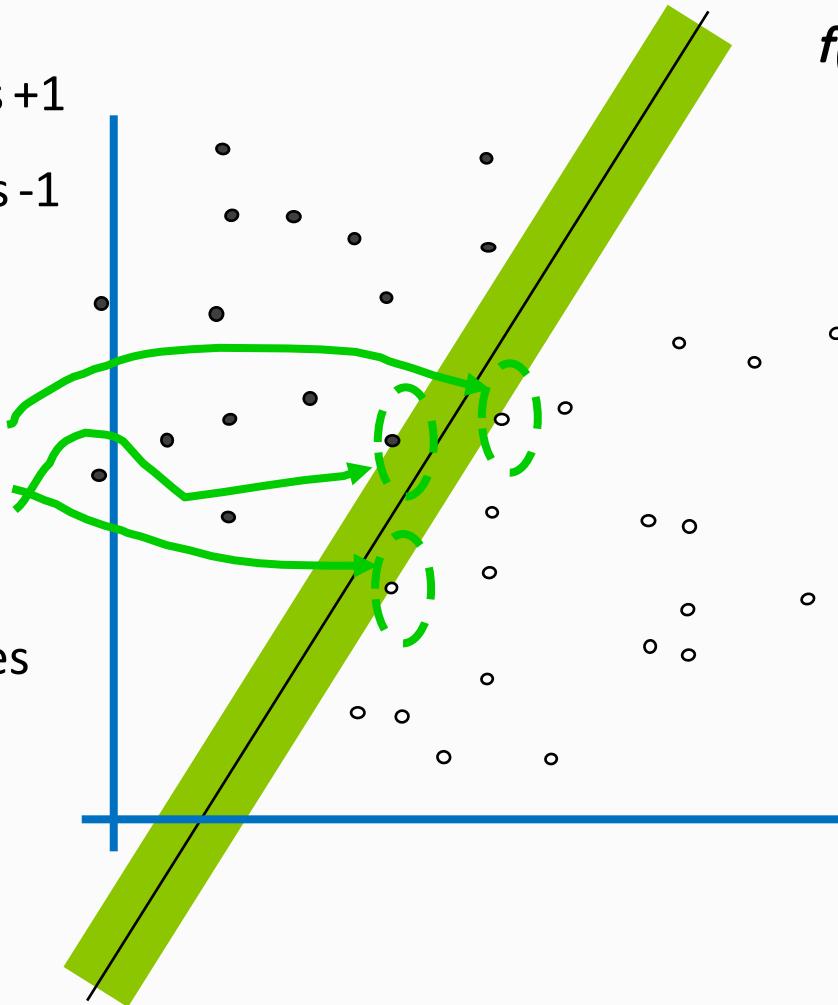
Support Vectors  
are those  
datapoints that  
the margin pushes  
up against



# Why Maximum Margin?

- denotes +1
- denotes -1

Support Vectors  
are those  
datapoints that  
the margin pushes  
up against



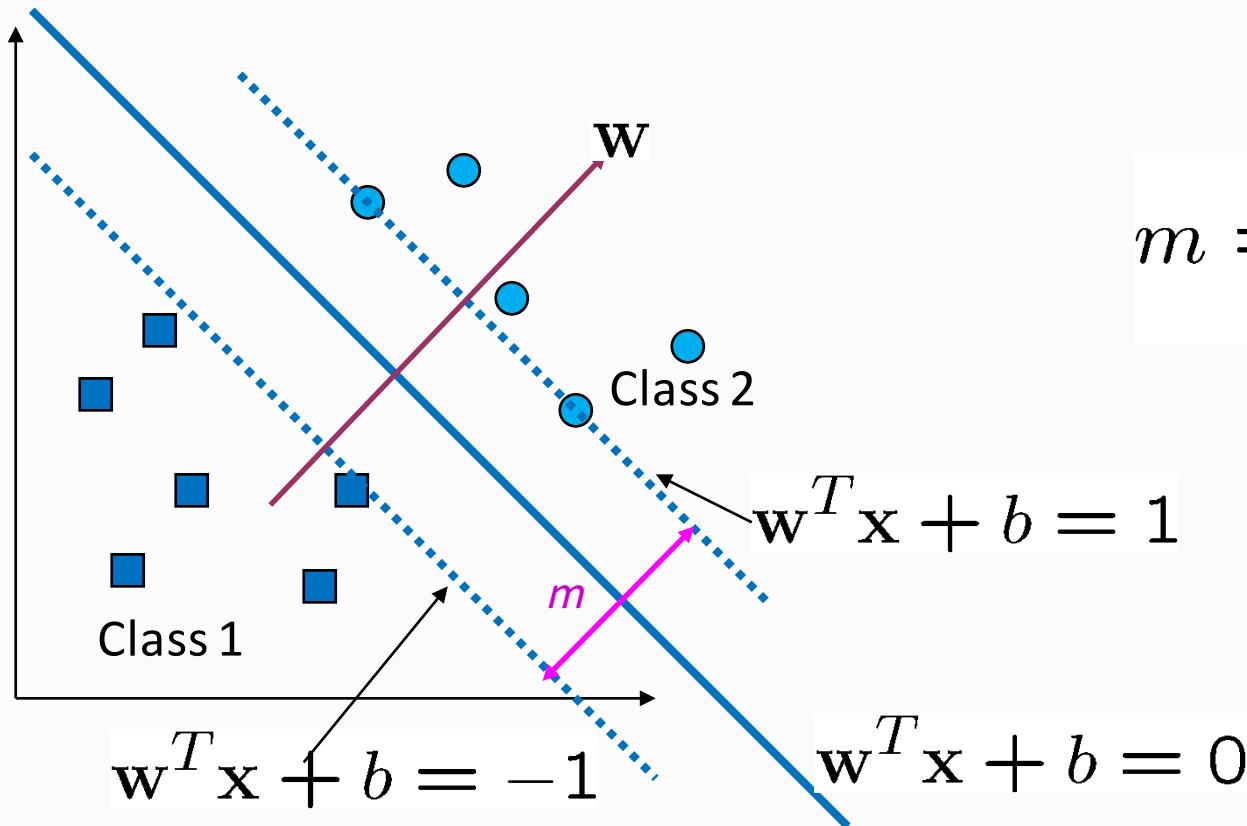
$$f(x, w, b) = \text{sign}(w \cdot x - b)$$

1. Intuitively this feels safest.
2. If we've made a small error in the location of the boundary this gives us least chance of causing a misclassification.
3. LOOCV (leave one out cross validation) is easy since the model is immune to removal of any nonsupport-vector data points.
5. Empirically it works **very** well.

# Large-Margin Decision Boundary

The decision boundary should be as far away from the data of both classes as possible

- We should maximize the margin,  $m$
- Distance between the origin and the line  $\mathbf{w}^T \mathbf{x} = -b$  is  $b / \| \mathbf{w} \|$

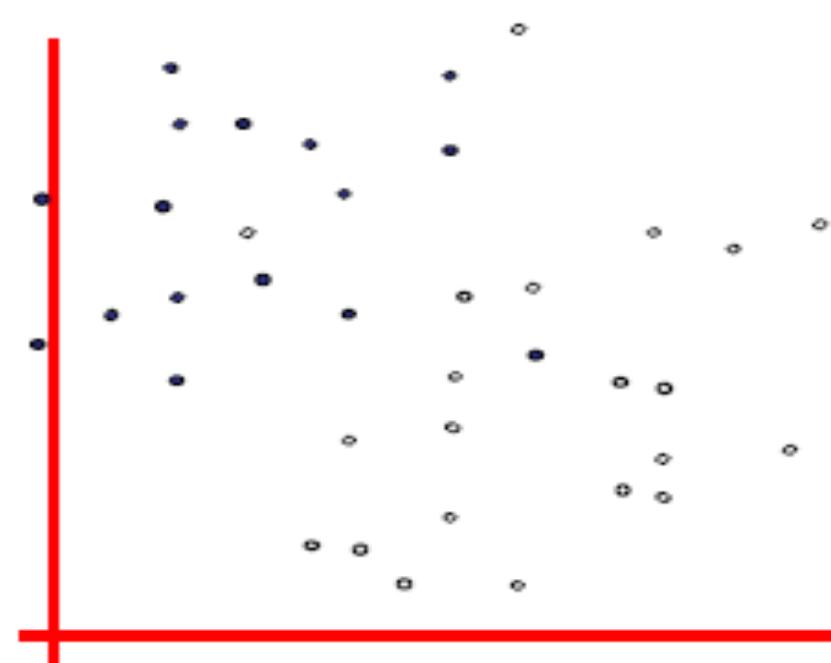


$$m = \frac{2}{\| \mathbf{w} \|}$$

# Uh-oh!

## This is going to be a problem!

- denotes +1
- denotes -1



### Slack Variables

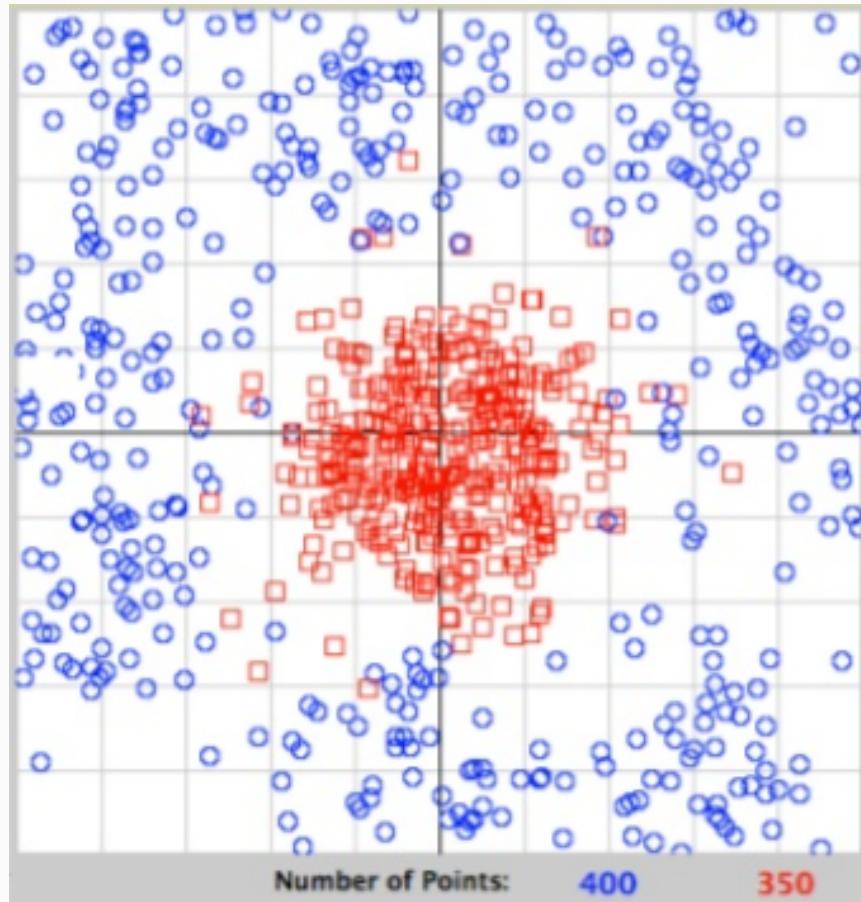
- SVMs are made robust by adding ‘slack variables’ that allow training error to be non-zero;
- One for each data point. Slack variable == 0 for correctly classified points

# Noise

- Have assumed that the data is separable (in original or transformed space)
- Can apply SVMs to noisy data by introducing a “noise” parameter  $C$
- $C$  bounds the influence of any one training instance on the decision boundary
- Still a quadratic optimization problem
- Have to determine  $C$  by experimentation

# The Kernel Trick

What if the data looks like that below?...

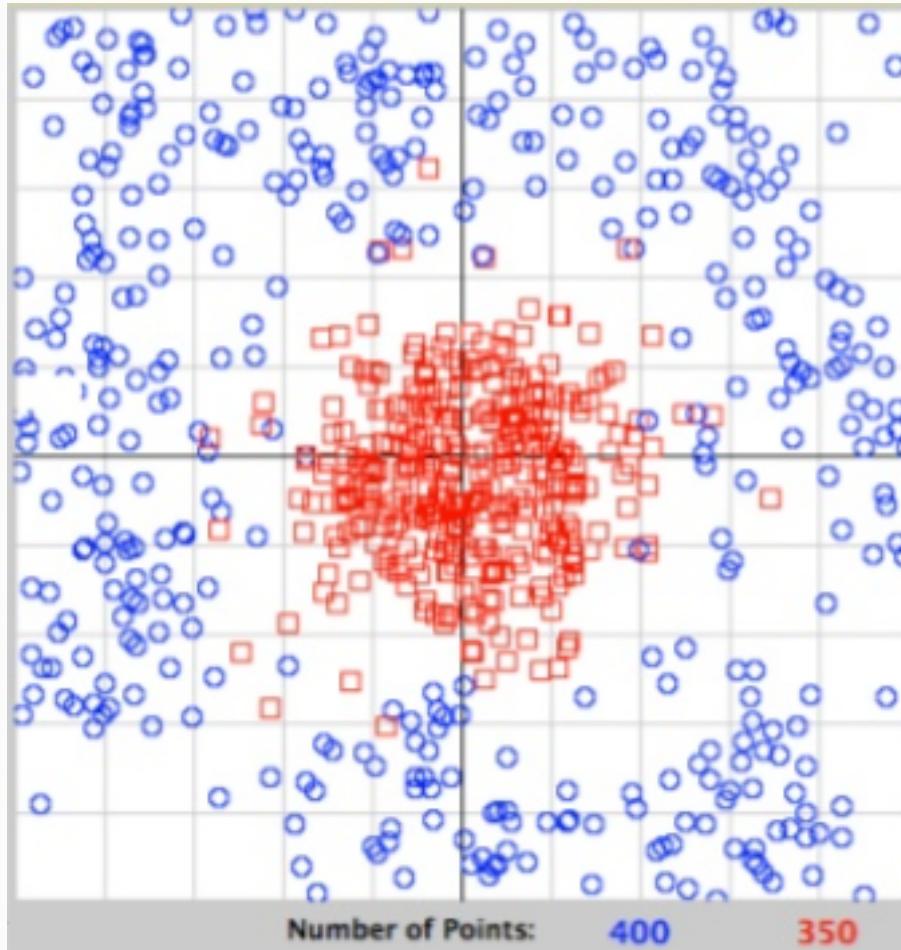


The simplest way to divide two groups is with a straight line, flat plane or an N-dimensional hyperplane. But what if the points are separated by a nonlinear region?

Rather than fitting nonlinear curves to the data, SVM handles this by using a ***kernel function*** to map the data into a different space where a hyperplane can be used to do the separation.

# The Kernel Trick

What if the data looks like that below?...



The kernel trick allows you to use SVMs with non-linear separators

Different kernels

- Polynomial
- Gaussian
- Exponential
- ...

# The Kernel Trick

## Everything you need to know in one slide...

Sometimes it improves your classifier if you map (i.e. transform) your data into a different feature space.

- Example 1: you have nonlinear data but want to use a linear classifier
- Example 2: your original data aren't numbers – for example they could be sequences

Kernel methods map your original data into a different space so that you can use linear classifiers.

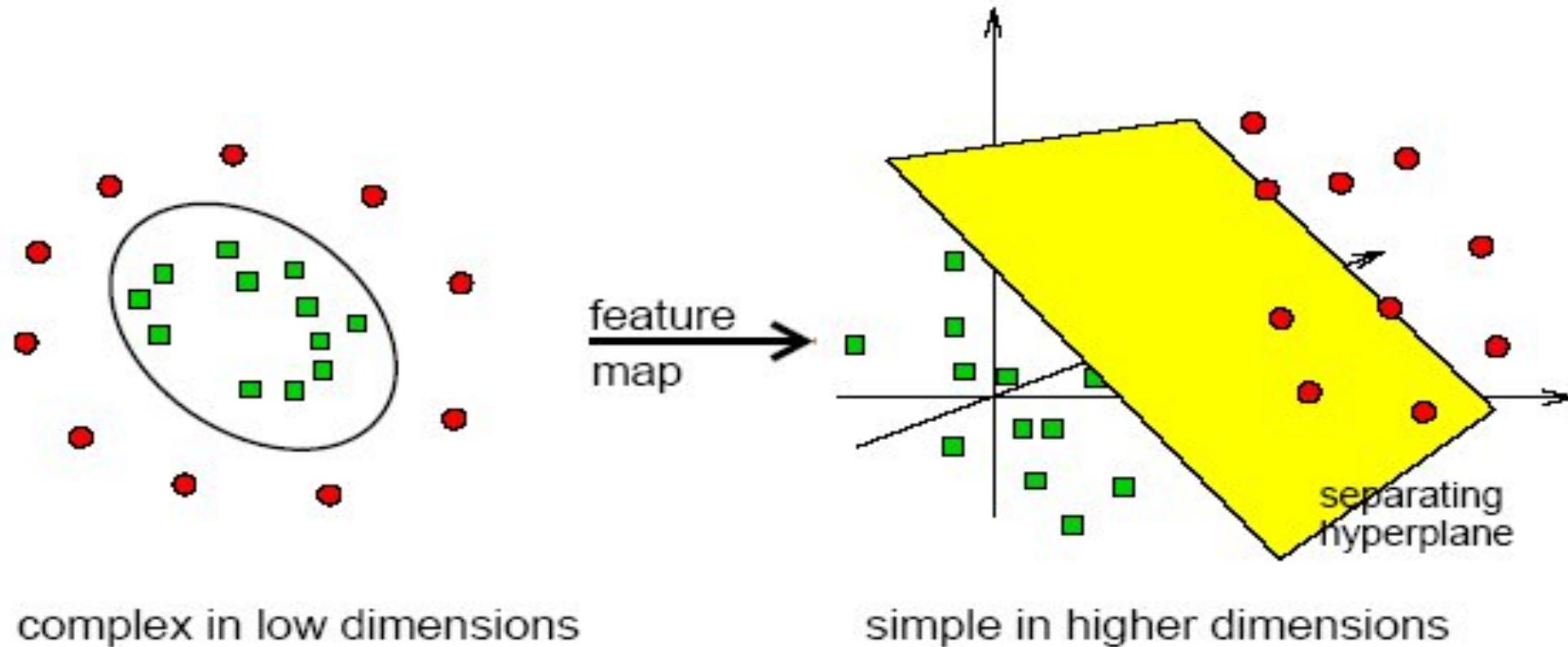
This mapping can often substantially increases the number of features to consider.

- This can be problematic as your number of dimensions grows.

The “Kernel Trick” addresses this by putting a cap on the feature explosion so that the complexity of your classifier increases only linearly with the size of your original data.

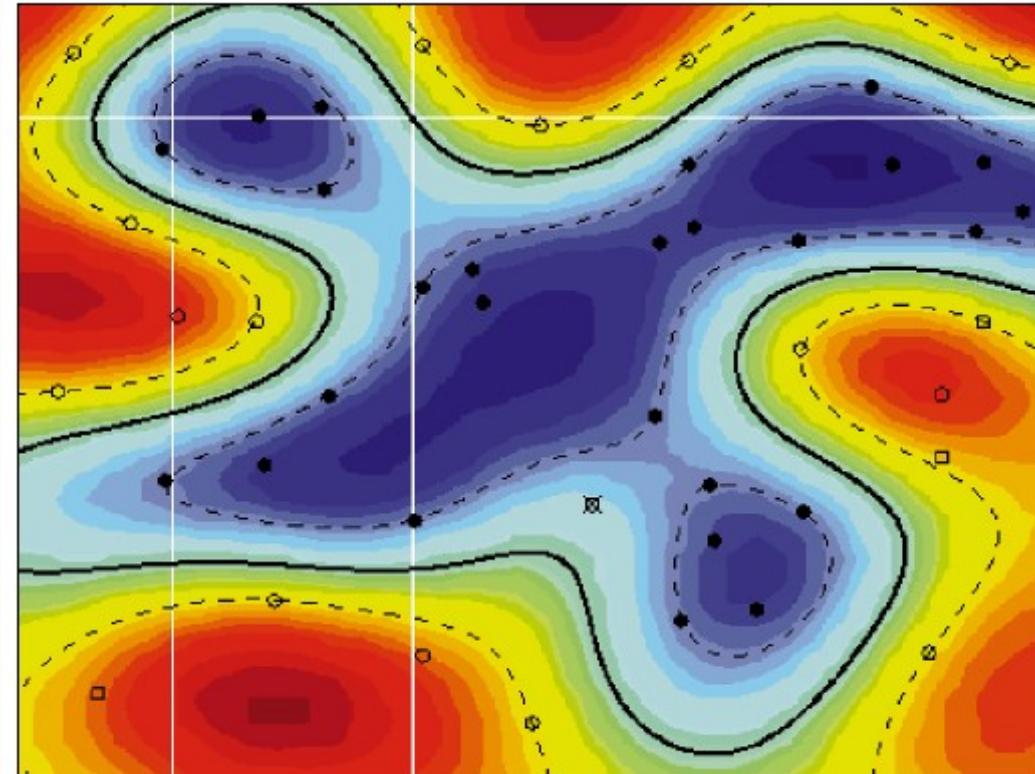
# Nonlinear Support Vector Machines

Separation may be easier in higher dimensions



# Nonlinear Support Vector Machines

The kernel function transforms the data into a higher dimensional space to make it possible to perform the separation.



# Choosing the Kernel Function

Probably the most tricky part of using SVM

*RBF is a good first option...*

Depends on your data—try several.

- *Kernels have even been developed for nonnumeric data like sequences, structures, and trees/graphs.*

May help to use a combination of several kernels.

Don't touch your evaluation data while you're trying out different kernels and parameters.

- Use cross-validation for this if you're short on data



Type of Kernel	Inner product kernel $K(\vec{x}, \vec{x}_i), i = 1, 2, \dots, N$	Comments
Polynomial Kernel	$K(\vec{x}, \vec{x}_i) = (\vec{x}^T \vec{x}_i + \theta)^d$	Power $p$ and threshold $\theta$ is specified a priori by the user
Gaussian Kernel	$K(\vec{x}, \vec{x}_i) = e^{-\frac{1}{2\sigma^2} \ \vec{x} - \vec{x}_i\ ^2}$	Width $\sigma^2$ is specified a priori by the user
Sigmoid Kernel	$K(\vec{x}, \vec{x}_i) = \tanh(\eta \vec{x}^T \vec{x}_i + \theta)$	Mercer's Theorem is satisfied only for some values of $\eta$ and $\theta$
Kernels for Sets	$K(\chi, \chi') = \sum_{i=1}^{N_\chi} \sum_{j=1}^{N_{\chi'}} k(x_i, x'_j)$	Where $k(x_i, x'_j)$ is a kernel on elements in the sets $\chi, \chi'$
Spectrum Kernel for strings	count number of substrings in common	It is a kernel, since it is a dot product between vectors of indicators of all the substrings.

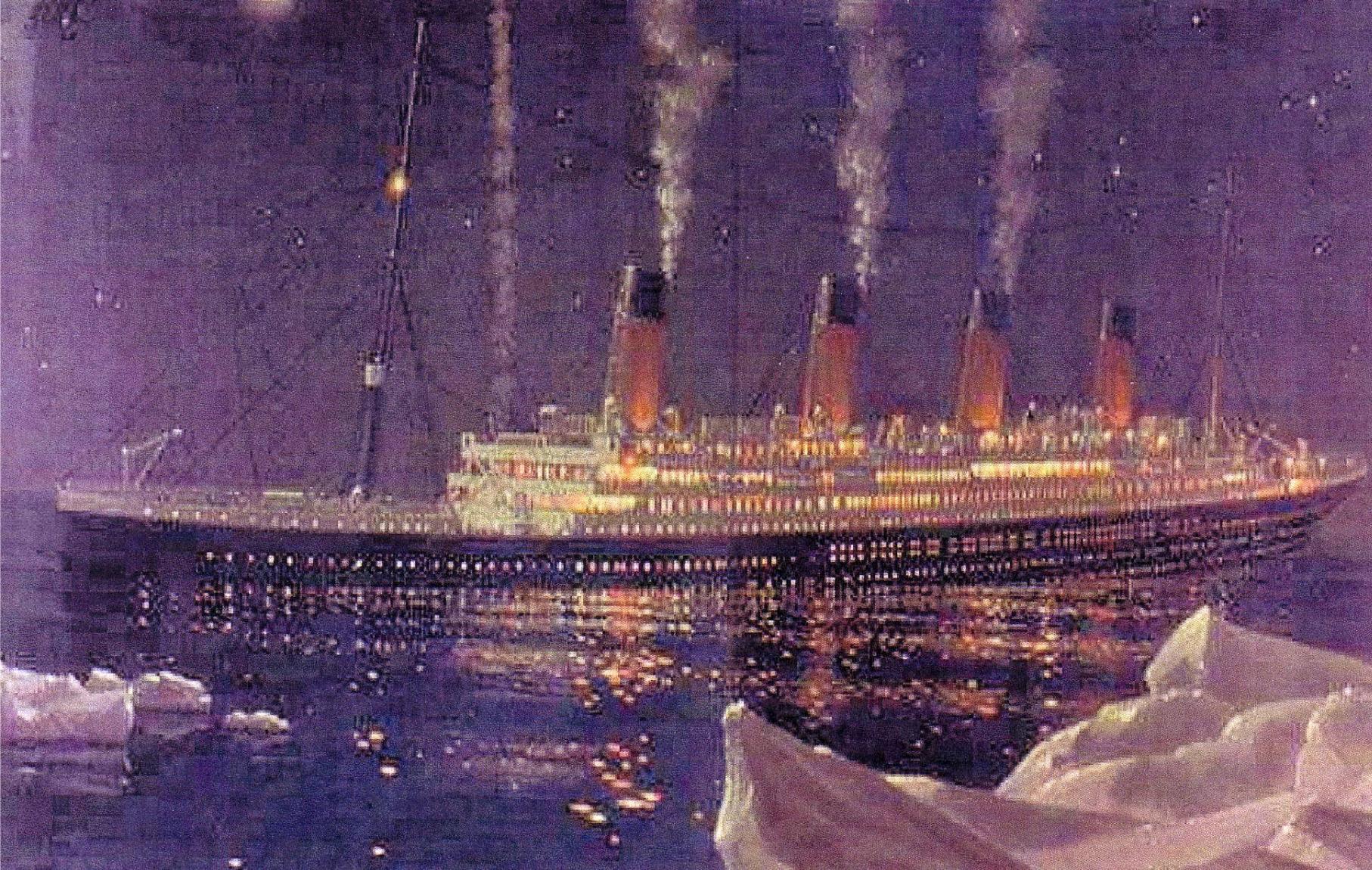
Table 1: Summary of Inner-Product Kernels [Hay98]

*Complexity of the optimization problem remains only dependent on the dimensionality of the input space and not of the feature space!*

# Example

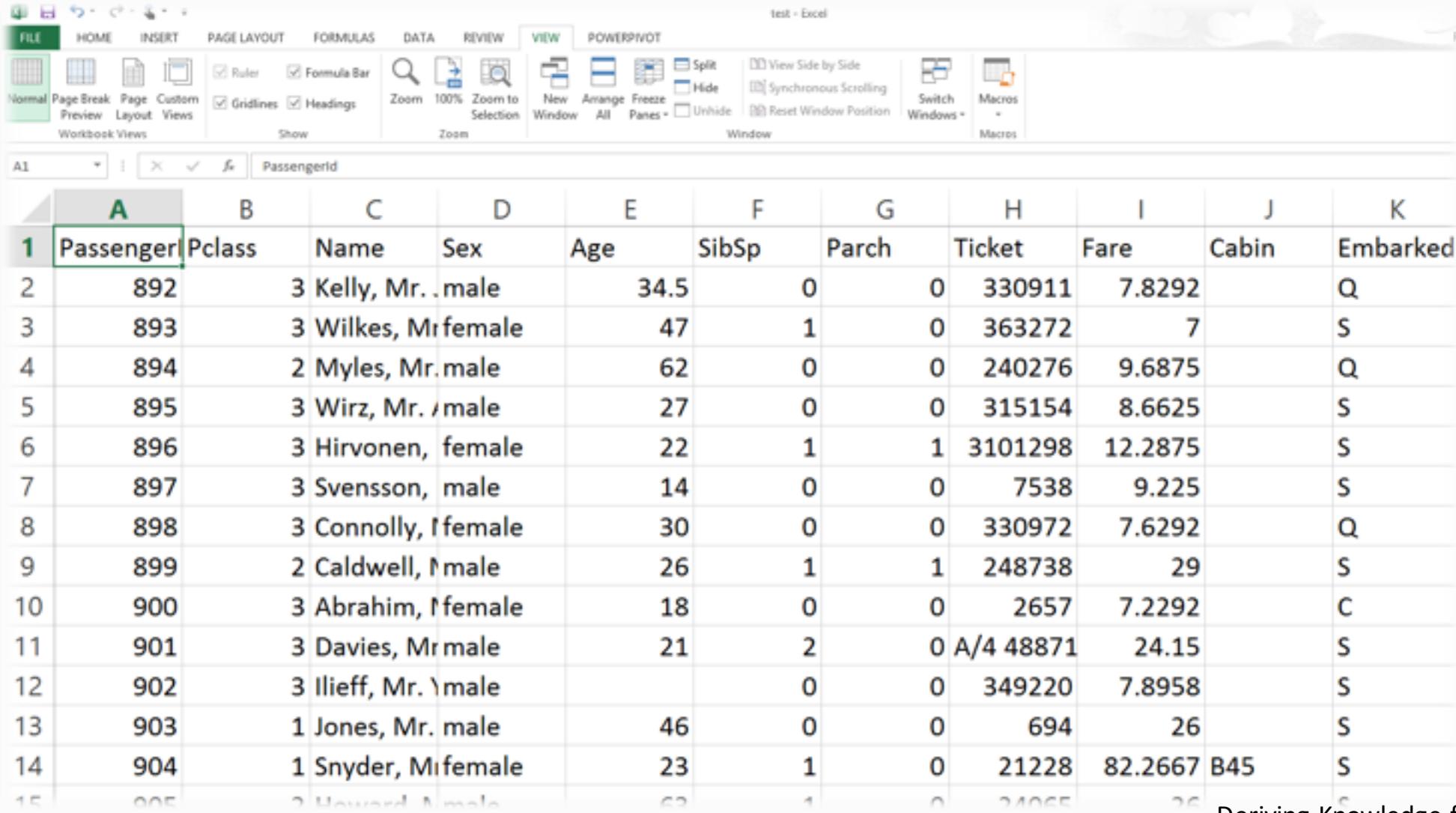


# The Tragedy of the Titanic



# First Steps...

- Look at the data, exploratory data analysis

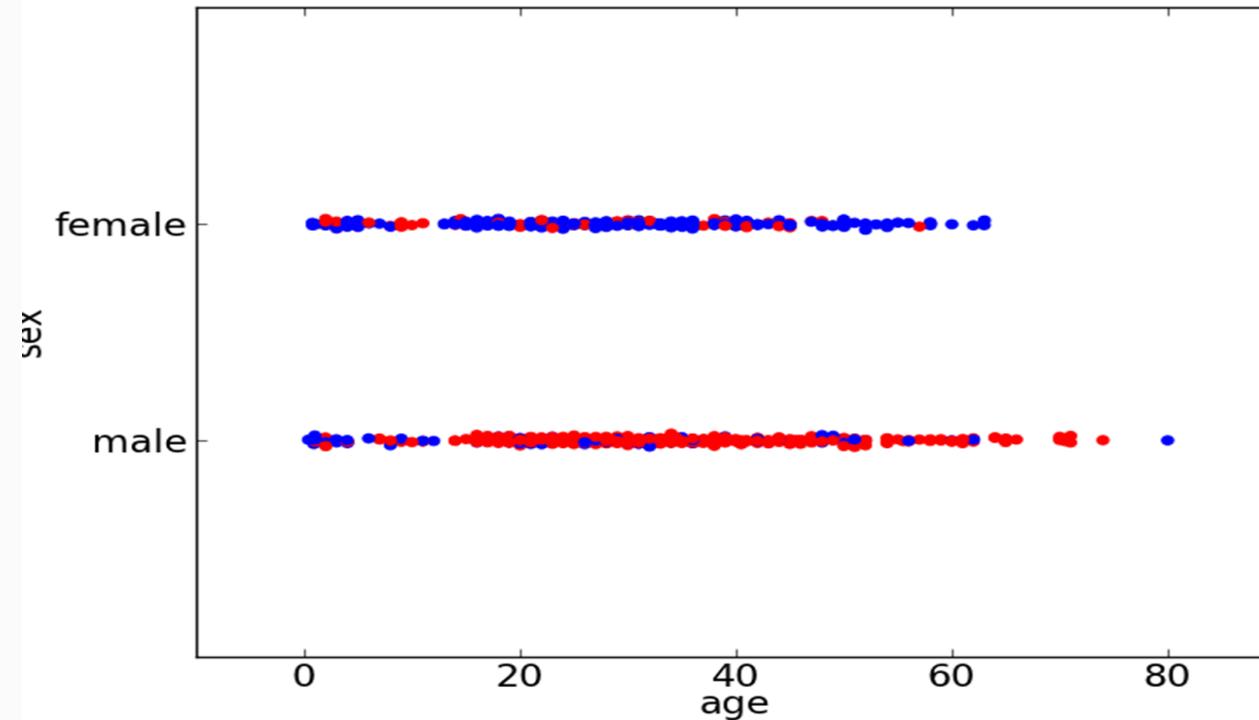


The screenshot shows a Microsoft Excel spreadsheet titled "test - Excel". The ribbon menu is visible at the top, with the "VIEW" tab selected. The main area displays a dataset of 15 passengers from the Titanic. The columns are labeled A through K, and the rows are numbered 1 through 15. The first row contains column headers: PassengerId, Pclass, Name, Sex, Age, SibSp, Parch, Ticket, Fare, Cabin, and Embarked. The data includes various passenger details such as name, age, sex, ticket number, fare, cabin, and embarkation point. Row 15 is partially visible at the bottom.

	A	B	C	D	E	F	G	H	I	J	K
1	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
2	892	3	Kelly, Mr.	male	34.5	0	0	330911	7.8292	Q	
3	893	3	Wilkes, Mr.	female	47	1	0	363272	7	S	
4	894	2	Myles, Mr.	male	62	0	0	240276	9.6875	Q	
5	895	3	Wirz, Mr.	male	27	0	0	315154	8.6625	S	
6	896	3	Hirvonen,	female	22	1	1	3101298	12.2875	S	
7	897	3	Svensson,	male	14	0	0	7538	9.225	S	
8	898	3	Connolly, I	female	30	0	0	330972	7.6292	Q	
9	899	2	Caldwell, M	male	26	1	1	248738	29	S	
10	900	3	Abrahim, I	female	18	0	0	2657	7.2292	C	
11	901	3	Davies, Mr.	male	21	2	0	A/4 48871	24.15	S	
12	902	3	Ilieff, Mr.	male		0	0	349220	7.8958	S	
13	903	1	Jones, Mr.	male	46	0	0	694	26	S	
14	904	1	Snyder, M	female	23	1	0	21228	82.2667	B45	S
15	905	2	Maurice, A	male	62	1	0	24065	26	S	

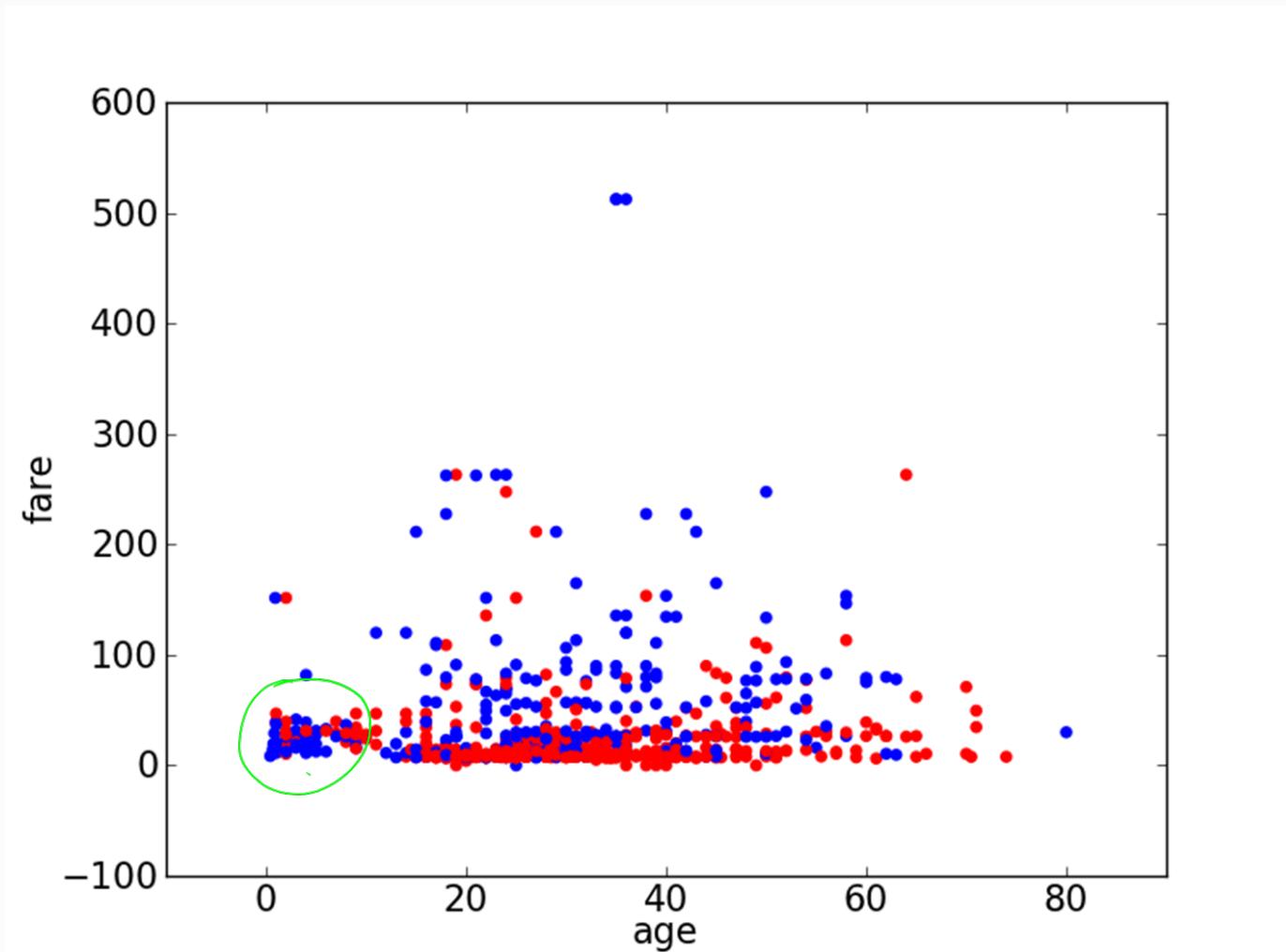
# First Steps...

- Look at the data;
- Follow your intuition, “*women and children first...*”
  - *What percentage of Females survived, percentage of males survived?...*



# First Steps...

- Other features good predictors?



# A very naïve classifier

pclass	sex	age	sibsp	parch	fare	cabin	embarked
1	female	35	1	0	53.1	C123	S

Does the new data point  $x^*$  *exactly* match a previous point  $x_i$ ?

If so, assign it to the same class as  $x_i$ .

Otherwise, just guess.

*This is the “rote” classifier*

# A minor improvement

pclass	sex	age	sibsp	parch	fare	cabin	embarked
1	female	35	1	0	53.1	C123	S

Does the new data point  $x^*$  match a set of previous points  $x_i$  on some specific attribute?

If so, take a vote to determine class.

Example: If most females survived, then assume every female survives

But there are lots of possible rules like this.

And an attribute can have more than two values.

If most people under 4 years old survive, then assume everyone under 4 survives

If most people with 1 sibling survive, then assume everyone with 1 sibling survives

How do we choose?



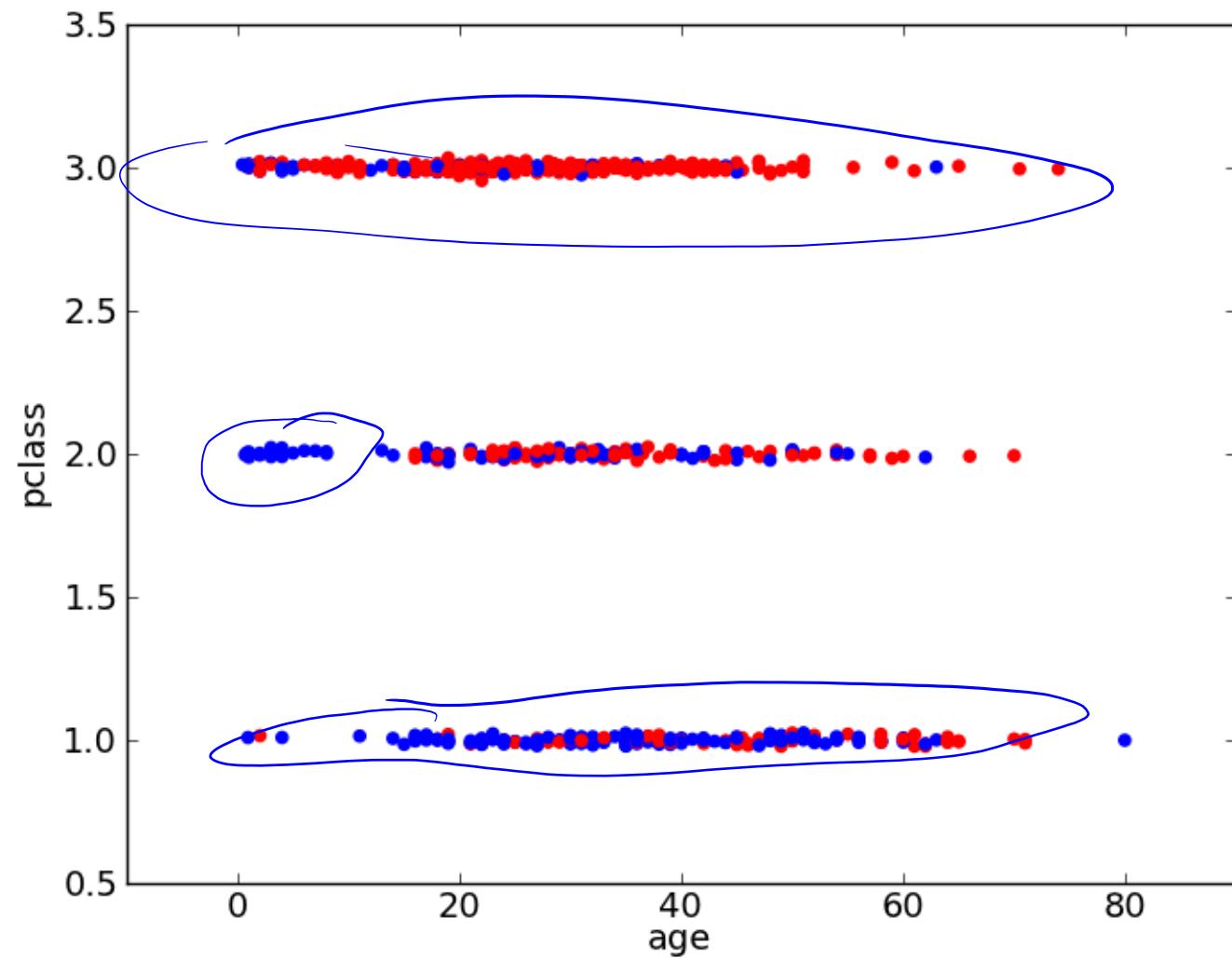
**IF sex='female' THEN survive=yes  
ELSE IF sex='male' THEN survive = no**

**confusion matrix**

		<-- classified as	
		no	yes
no	468	109	
yes	81	233	

$$\frac{(468 + 233)}{(468+109+81+233)} = 79\% \text{ correct (and 21\% incorrect)}$$

**Not bad!**



```
IF pclass='1' THEN survive=yes  
ELSE IF pclass='2' THEN survive=yes  
ELSE IF pclass='3' THEN survive=no
```

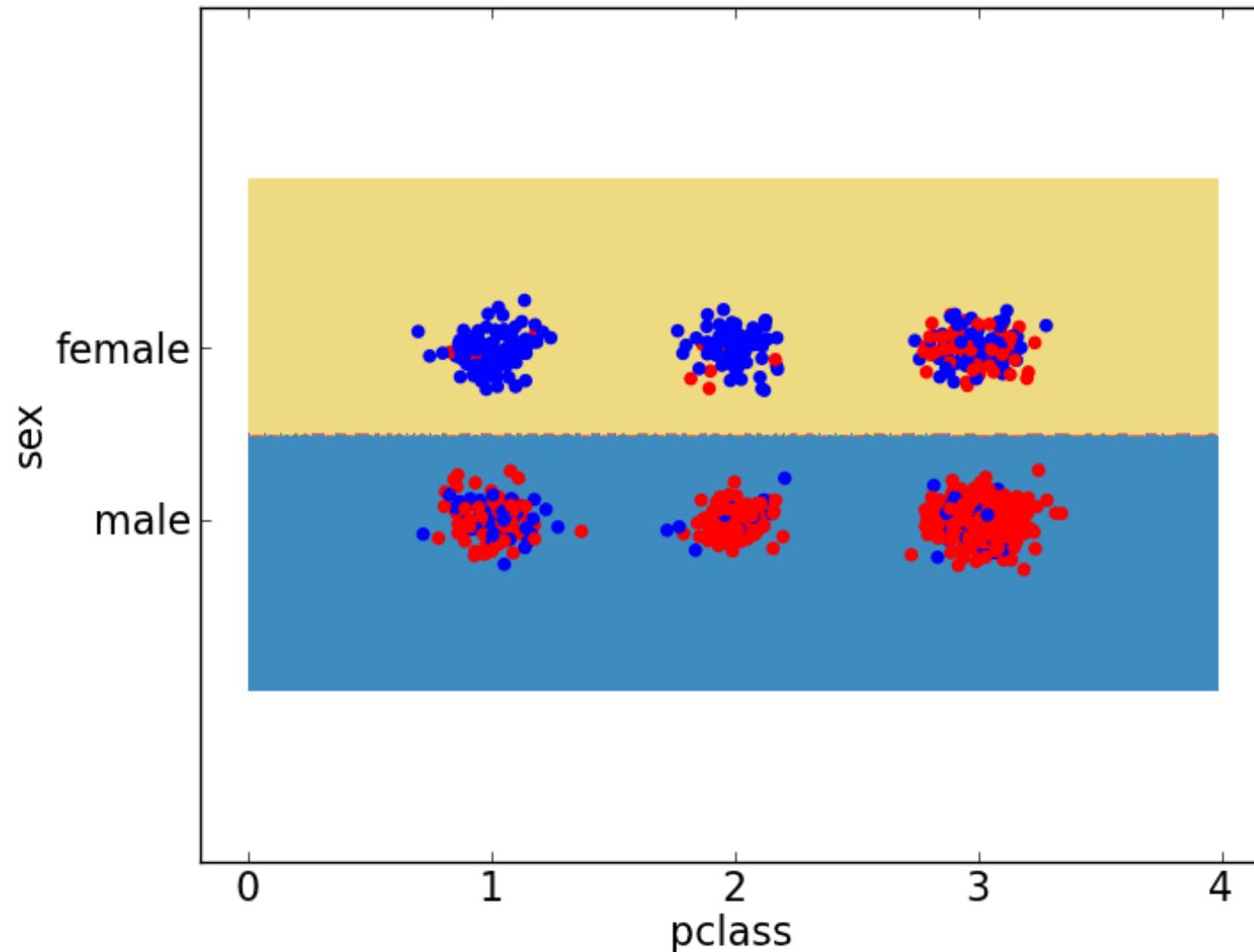
#### confusion matrix

no	yes	<-- classified as
372	119	no
177	223	yes

$$(372 + 223) / (372+119+223+177) = 67\% \text{ correct (and 33\% incorrect)}$$

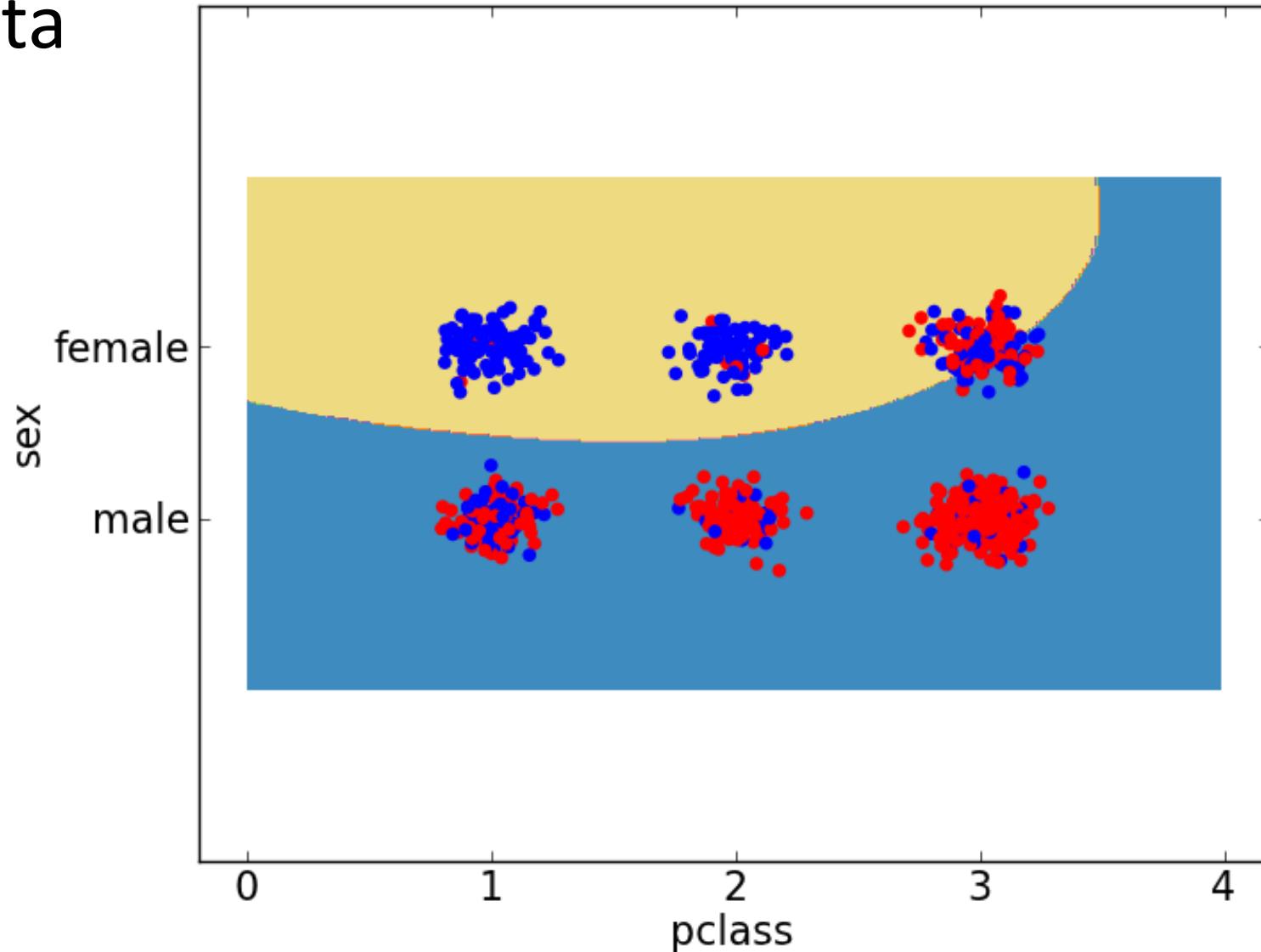
a little worse

# Support Vector Machine Model, Titanic Data, Linear Kernel



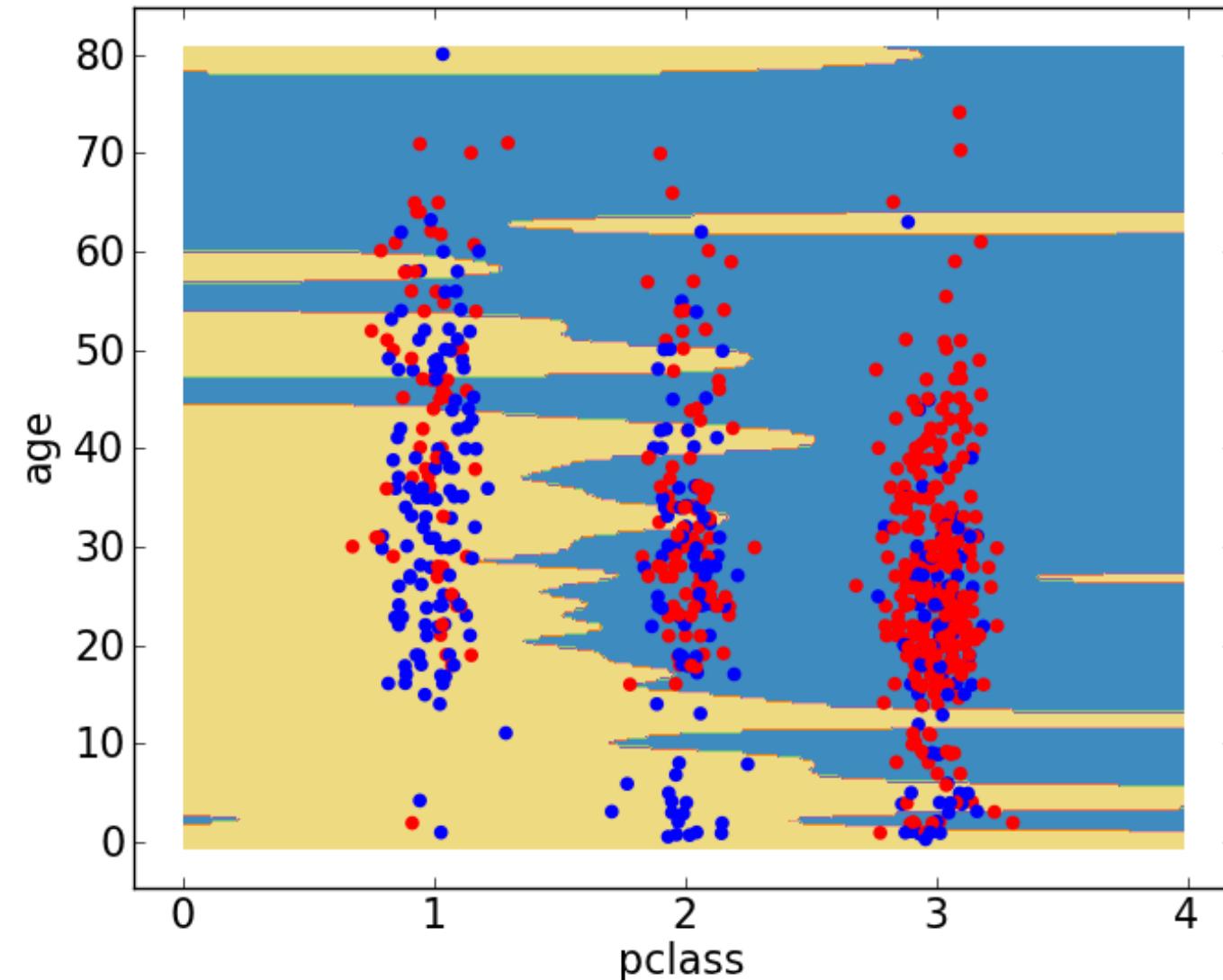
# Support Vector Machine Model, RBF Kernel

## Titanic Data



# Support Vector Machine Model, RBF Kernel

## Titanic Data

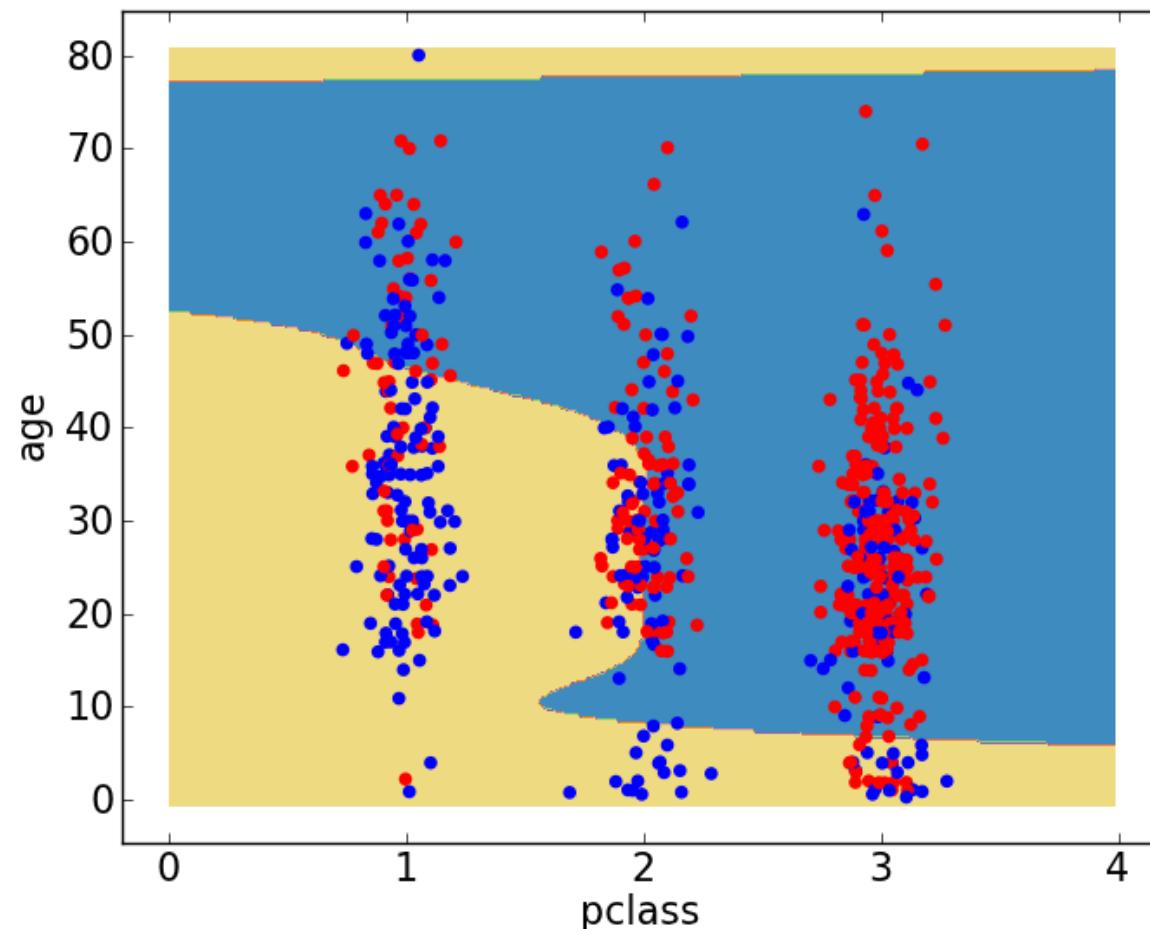


*overfitting?*

# Support Vector Machine Model, RBF Kernel

## Titanic Data

*A gamma, parameter that controls/balances model complexity against accuracy*



# Sparse Data

- SVM algorithms speed up dramatically if the data is sparse (i.e. many values are 0)
- Why? Because they compute lots and lots of dot products
- Sparse data compute dot products very efficiently
- Iterate only over nonzero values
- SVMs can process sparse datasets with 10,000s of attributes



# Doing multi-class classification

- SVMs can only handle two-class outputs (i.e. a categorical output variable with arity 2).
- What can be done?

# Doing multi-class classification

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

# What you need to know...

- First, try logistic regression. Easy, fast, stable. No ‘tuning’ parameters.
- Equivalently, you can first try linear SVMs, but you need to tune ‘C’
- If results are “good enough”, stop
- Else, try SVMs with Gaussian kernels (RBF)

Need to tune bandwidth, C – by using validation data...



# Summary: Steps for Classification

- SVMs require vector of real numbers
  - Categorical variables → numeric data {R,G,B} → {0,0,1},...{1,0,0}
  - Scaling to the range [-1, +1] or [0,1]
- Select the kernel function to use
  - RBF is a reasonable first choice, two parameters  $C$  and  $\gamma$
  - Grid search to identify best values for parameters
  - Use  $v$ -fold cross validation to ensure good performance on test data
- Unseen data can be classified using support vectors



# Conclusion

- SVMs balance between correctness and generalization
  - Decision boundaries
  - Margins
  - Support vector
- Two key concepts of SVM: maximize the margin and the kernel trick
- Many SVM implementations are available on the web for you to try on your data set!

# Data Science

Deriving Knowledge from Data at Scale

*That's all for tonight...*

