Probabilistic and Bayesian Analytics

Chenghua Lin

Chenghua.Lin@abdn.ac.uk

Road Map

- Probabilistic and Bayesian Analytics
- Classification
  - Naïve Bayes Classifier
  - Support vector machines (SVM)
  - Decision Trees
- Association Rule Mining
- Feature Engineering
- Data visualization
- Case study
- Data Mining Issues

Today’s Lecture

- Probabilistic and Bayesian Analytics
- Classification
  - Naïve Bayes Classifier
  - Support vector machines (SVM)
  - Decision Trees
- Association Rule Mining
- Feature Selection
- Visualization I and II
- Case study
- Data Mining Issues

Probability

- The world is a very uncertain place
- Probability: a mathematical framework for reasoning about uncertainty
  - How likely next Monday is going to rain?
  - You have a headache. What's the chance you have got flu?
  - Many other similar examples …

What we’re going to do

- We will review the fundamentals of probability.
- It’s really going to be worth it
- We will keep mathematics to the minimum

Discrete Random Variables

- A is a **Boolean-valued random variable** if A denotes an event, and there is some degree of uncertainty as to whether A occurs.
- Examples
  - A = The UK PM in 2023 will be female
  - A = You wake up tomorrow with a headache
  - A = You have Ebola…

The Axioms of Probability

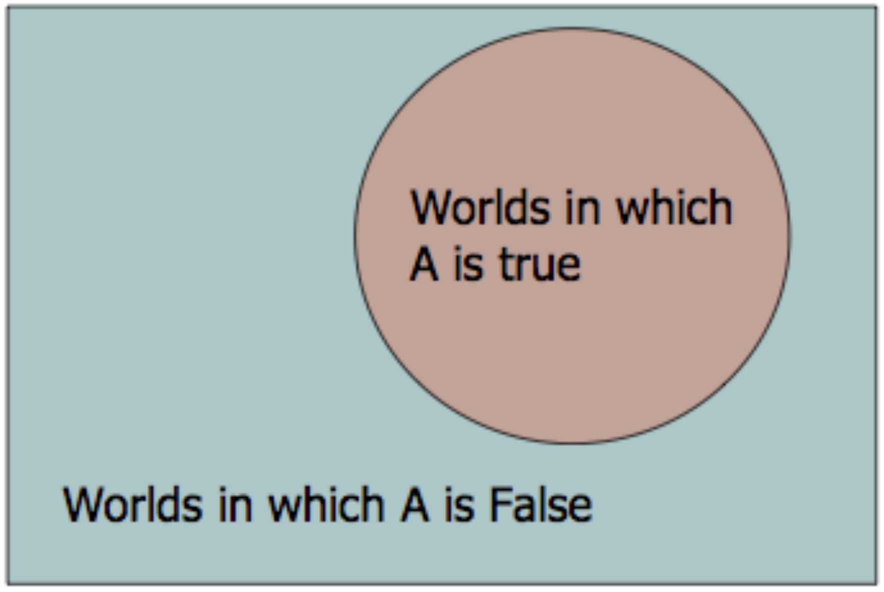
- $0 <= P(A) <= 1$
- $P(\Omega) = 1$

Probabilities

- We write P(A) as "**the fraction of possible worlds in which A is true**"

Sample space of all possible worlds **(Ω)**

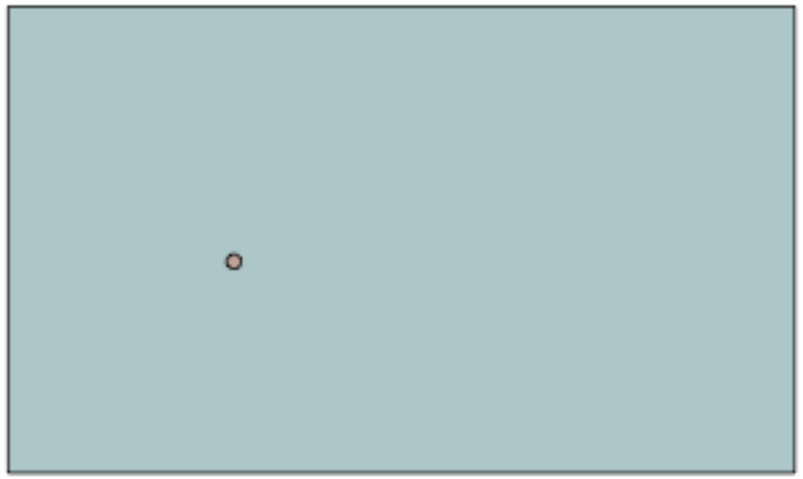
Its area is 1



P(A) = Area of reddish oval

Interpreting the axioms

- 0<=P(A)<=1
- P(Ω) = 1

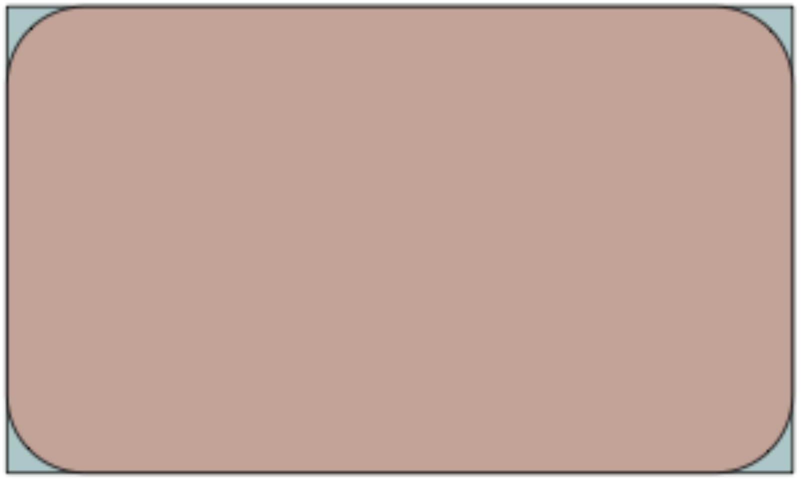


The area of A can't get any smaller than 0

And a zero area would mean no world could ever have A true

Interpreting the axioms

- 0<=P(A)<=1
- P(Ω) = 1



The area of A can't get any bigger than 1

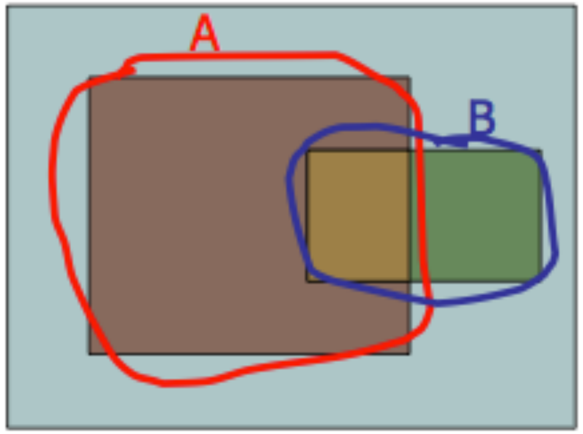
And an area of 1 would mean all worlds will have A true

The Axioms of Probability

- $0 <= P(A) <= 1$
- $P(\Omega) = 1$
- P(A or B)=P(A)+P(B)-P(A and B)

Interpreting the axioms

- $0 <= P(A) <= 1$
- $P(\Omega) = 1$
- P(A or B)=P(A)+P(B)-P(A and B)



Multivalued Random Variables

- Suppose A can take on more than 2 values
- A is a *random variable with arity k* if it can take on exactly one value out of {v1,v2, .. vk}
  - i.e., the elements of the value space has to be **mutually exclusive**
- Thus... $P(A=v_i \wedge A=v_j)=0 \text{ if } i \neq j$

$$P(A=v_1 \vee A=v_2 \vee A=v_k)=1$$

An easy fact about Multivalued Random Variables

- Using the axioms of probability
  - 0<=P(A)<=1
  - P(Ω) = 1
  - P(A or B)=P(A)+P(B)-P(A and B)
- And assuming that A obeys

$$P(A = v_i \wedge A = v_j) = 0 \text{ if } i \neq j$$
$$P(A = v_1 \vee A = v_2 \vee A = v_k) = 1$$

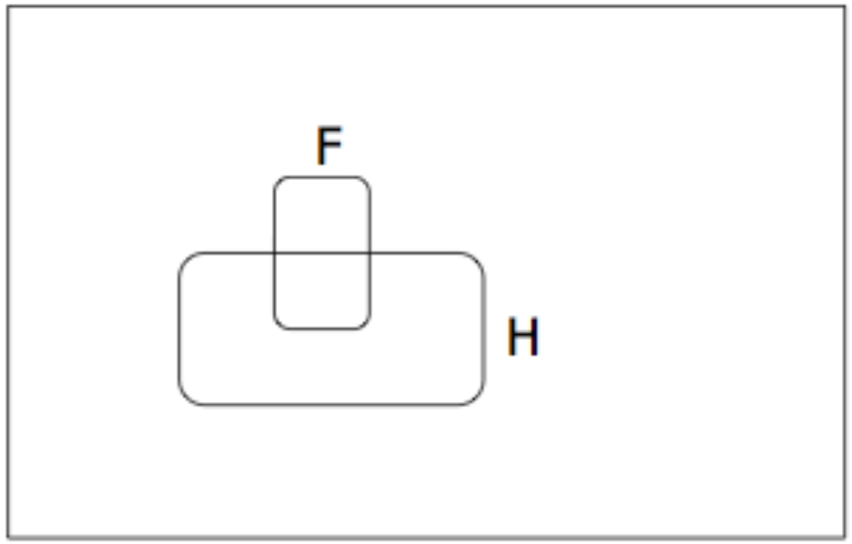
- It is easy to prove that

$$P(A = v_1 \vee A = v_2 \vee A = v_i) = \sum_{j=1}^{i} P(A = v_j)$$

- And thus we can prove $\sum_{j=1}^{k} P(A = v_j) = 1$

Conditional Probability

- P(A|B) = The probability of an event (A), given that another (B) has already occurred.

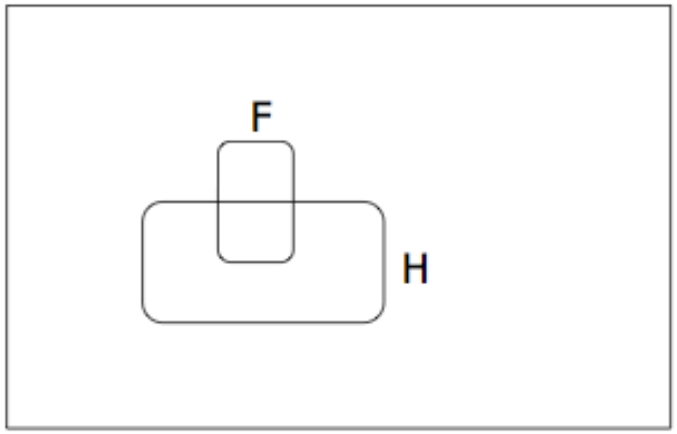


H = "Have a headache"
F = "Coming down with Flu"

P(H) = 1/10
P(F) = 1/40
P(H|F) = 1/2

"Headaches are rare and flu is rarer, but if you're coming down with 'flu there's a 50-50 chance you'll have a headache."

Conditional Probability



H = “Have a headache”
F = “Coming down with Flu”
P(H) = 1/10
P(F) = 1/40
P(H|F) = 1/2

P(H|F) = Fraction of flu-inflicted worlds in which you have a headache

$$= \frac{\text{\#worlds with flu and headache}}{\text{\#worlds with flu}}$$

$$= \frac{\text{Area of “H and F” region}}{\text{Area of “F” region}}$$

$$= \frac{P(H \wedge F)}{P(F)}$$

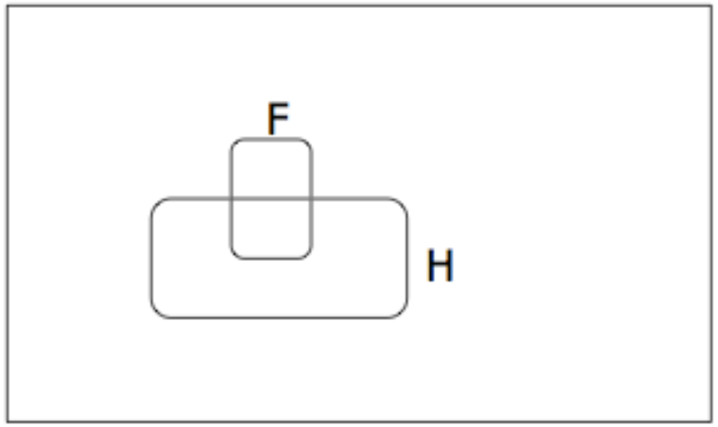
Definition of Conditional Probability

$$P(A|B) = \frac{P(A \wedge B)}{P(B)}$$

- Corollary: The Chain Rule

$P(A \wedge B) = P(A|B)\, P(B)$

Probabilistic Inference



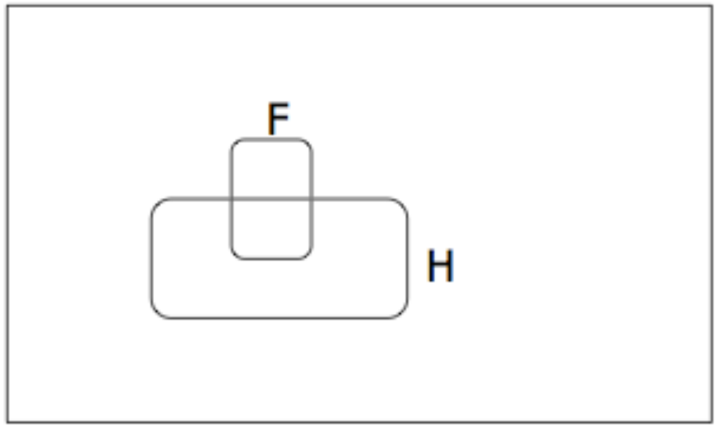
H = “Have a headache”
F = “Coming down with Flu”

P(H) = 1/10
P(F) = 1/40
P(H|F) = 1/2

One day you wake up with a headache. You think: “Drat! 50% of flus are associated with headaches so I must have a 50-50 chance of coming down with flu”

Is this reasoning good?

Probabilistic Inference



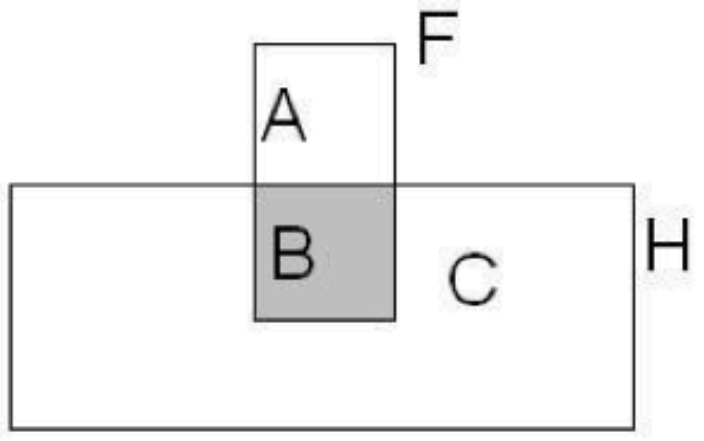
H = “Have a headache”
F = “Coming down with Flu”

P(H) = 1/10
P(F) = 1/40
P(H|F) = 1/2

P(F ^ H) = ...

P(F|H) = ...

Another way to understand the intuition



Let's say we have P(F), P(H), and P(H|F), like in the example in class.

Areawise, P(F) = A + B, P(H) = B + C,

Also, $P(H|F) = \frac{B}{A + B}$

Thus, to get the opposite conditional probability, ie, P(F|H), we need to figure out $\frac{B}{B + C}$

Since we know B / (A+B), we can get B / (B+C) by multiplying by (A+B) and dividing by (B+C). But since we already calculated, A+B = P(F), and B+C = P(H), so we are actually multiplying by P(F) and dividing by P(H). Which is Bayes Rule:

$$P(F|H) = P(H|F) * \frac{P(F)}{P(H)}$$

What we just did...

$$P(B|A) = \frac{P(A \wedge B)}{P(A)} = \frac{P(A|B)\,P(B)}{P(A)}$$

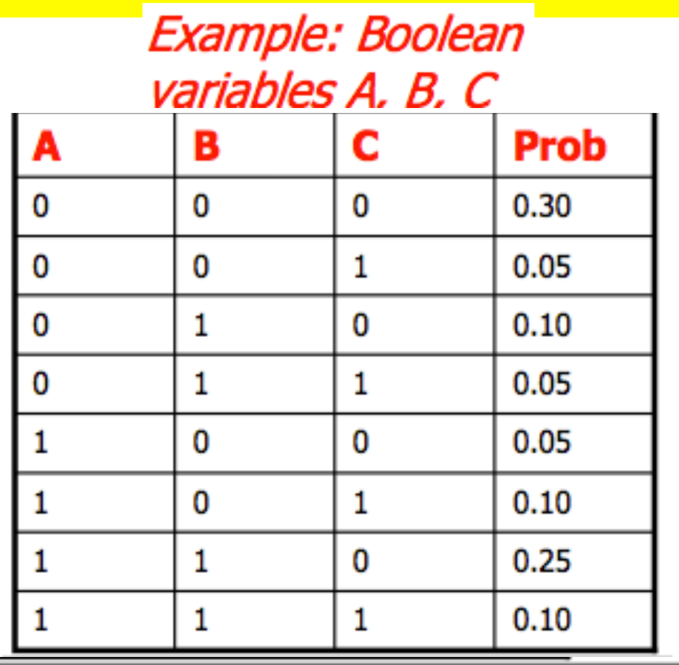
This is the FAMOUS **Bayes Rule**

Bayes, Thomas (1763) An essay towards solving a problem in the doctrine of chances. Philosophical Transactions of the Royal Society of London, **53:370-418**

The Joint Distribution

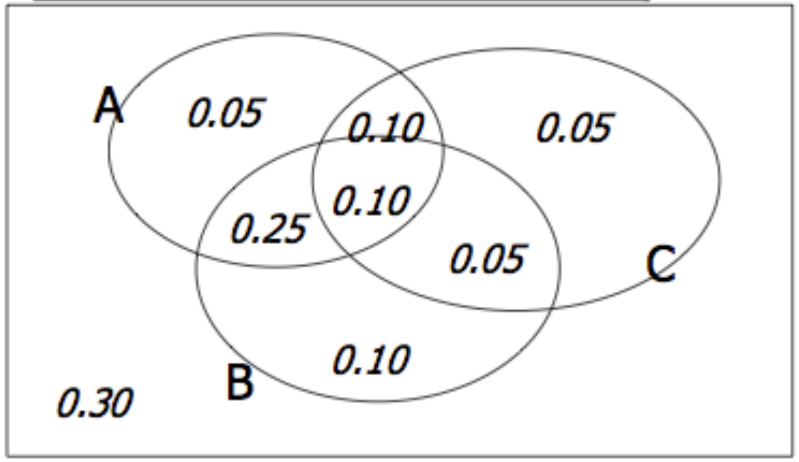
Joint distribution: the probability of two or more events occurring together.

1. Make a truth table listing all combinations of values of your variables (if there are M Boolean variables then the table will have 2M rows).
2. For each combination of values, say how probable it is.
3. If you subscribe to the axioms of probability, those numbers must sum to 1.



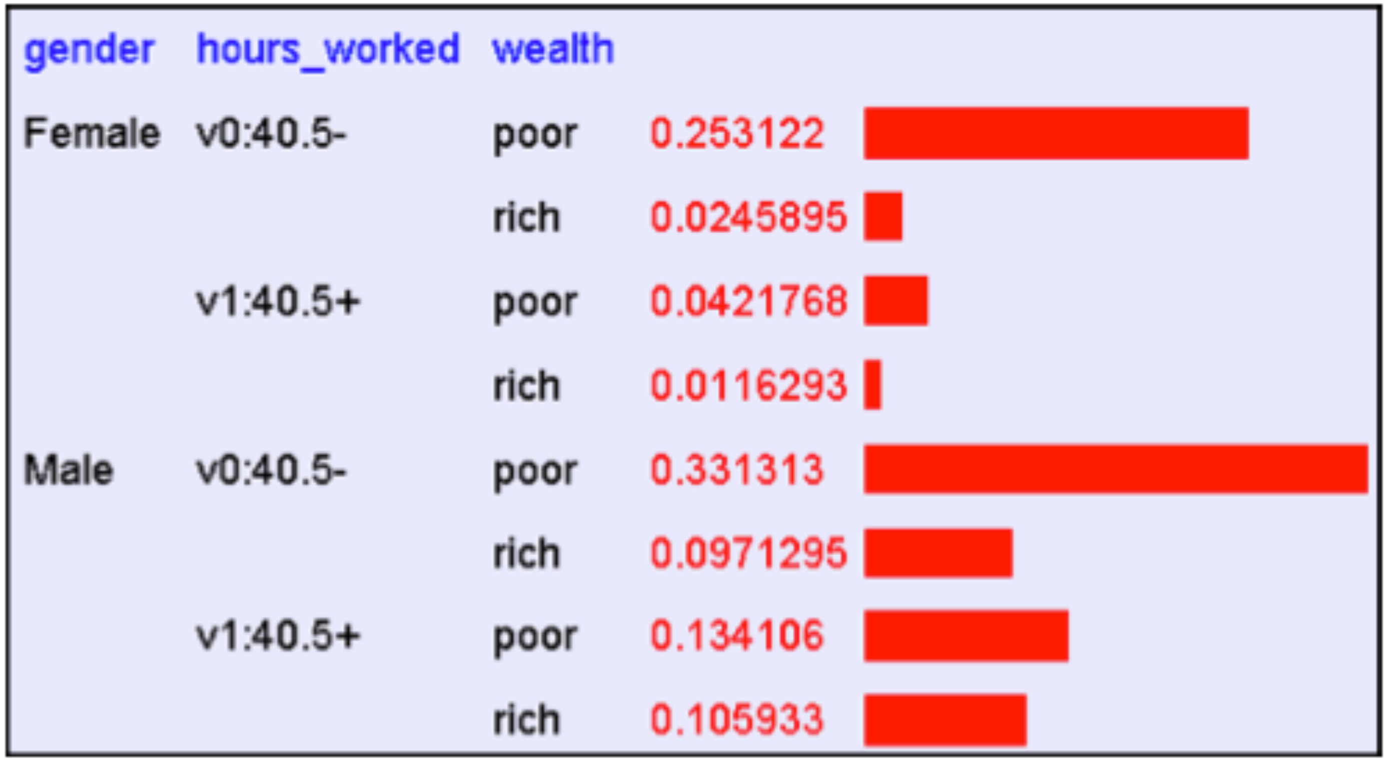
Example: Boolean variables A, B, C

A	B	C	Prob
0	0	0	0.30
0	0	1	0.05
0	1	0	0.10
0	1	1	0.05
1	0	0	0.05
1	0	1	0.10
1	1	0	0.25
1	1	1	0.10



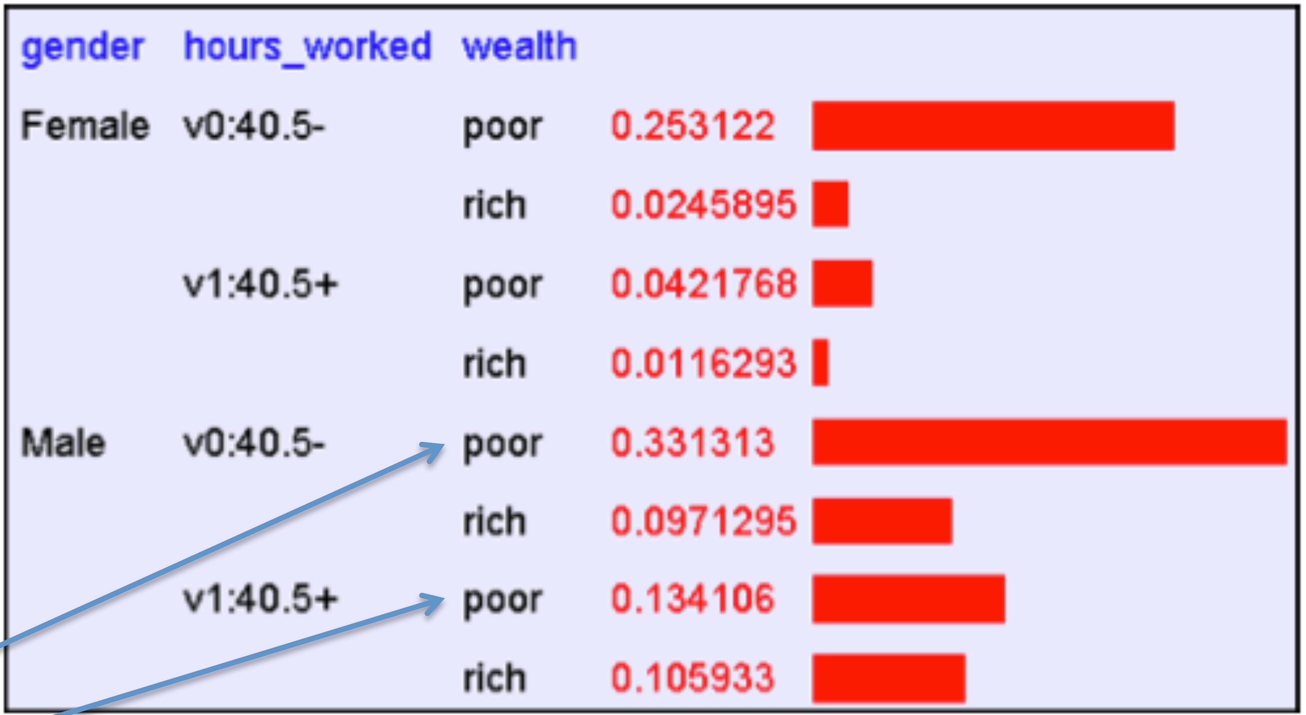
Using the Joint

- Once you have the joint distribution you can ask for the probability of any logical expression involving your attribute



gender	hours_worked	wealth		
Female	v0:40.5-	poor	0.253122	
		rich	0.0245895	
	v1:40.5+	poor	0.0421768	
		rich	0.0116293	
Male	v0:40.5-	poor	0.331313	
		rich	0.0971295	
	v1:40.5+	poor	0.134106	
		rich	0.105933	

Using the Joint



gender	hours_worked	wealth		
Female	v0:40.5-	poor	0.253122	🟥
		rich	0.0245895	🟥
	v1:40.5+	poor	0.0421768	🟥
		rich	0.0116293	🟥
Male	v0:40.5-	poor	0.331313	🟥
		rich	0.0971295	🟥
	v1:40.5+	poor	0.134106	🟥
		rich	0.105933	🟥

P(Poor Male) = 0.4654

P(Poor) = 0.7604

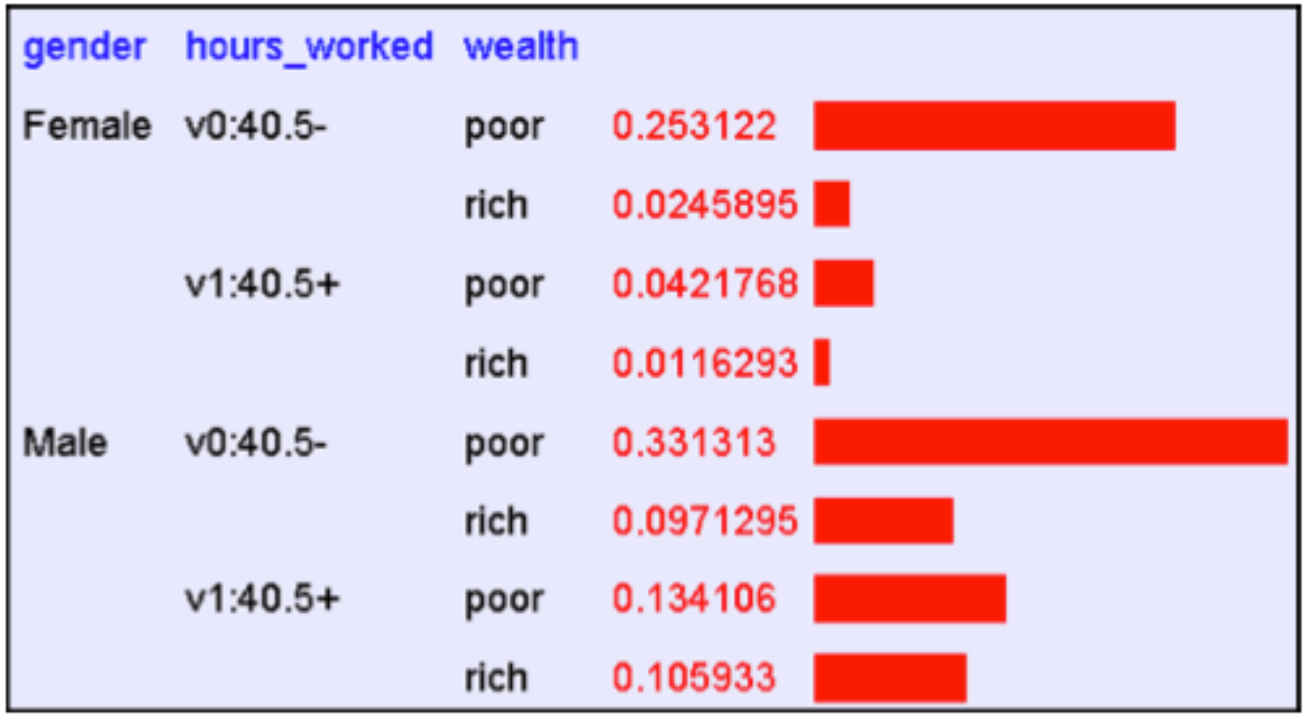
$$P(E) = \sum_{\text{rows matching } E} P(\text{row})$$

Inference with the Joint

P(Poor Male) = 0.4654

P(Poor) = 0.7604

P(Male | Poor)
= 0.4654 / 0.7604
= 0.612



gender	hours_worked	wealth		
Female	v0:40.5-	poor	0.253122	🟥
		rich	0.0245895	🟥
	v1:40.5+	poor	0.0421768	🟥
		rich	0.0116293	🟥
Male	v0:40.5-	poor	0.331313	🟥
		rich	0.0971295	🟥
	v1:40.5+	poor	0.134106	🟥
		rich	0.105933	🟥

$$P(E_1 \mid E_2) = \frac{P(E_1 \wedge E_2)}{P(E_2)} = \frac{\sum_{\text{rows matching } E_1 \text{ and } E_2} P(\text{row})}{\sum_{\text{rows matching } E_2} P(\text{row})}$$

What you should know

- The Axioms of Probability
- Conditional probability
- Joint probability
- Bayes rule

Naïve Bayesian Classifier

Chenghua Lin
Chenghua.Lin@abdn.ac.uk

What you should know

- Why it is called naïve?
  - The independence assumption
- How to build a naïve Bayes classifier
  - What happen in training time
  - What happen in testing time
- How to deal with unseen features
  - Smoothing

Overview

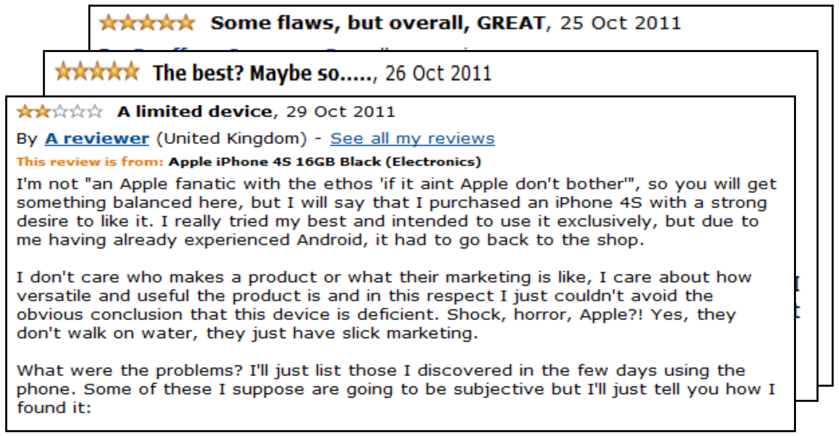
- We have learnt -- Bayes rule
- Today we learn about:
  - How to turn Bayes rule into a classifier, i.e., a naïve Bayes classifier
  - A supervised probabilistic model of the observed data
  - Can be used to predict the class label of new/ unseen data

Supervised Classification

- **Supervised learning**: the machine learning task of inferring a function from labeled training data

- Given:
  - **Target**: a fixed set of **classes**: Y = {y_1, y_2,…, y_n}, e.g. {sports, politics, …, music}
  - **Training data**: a collection of data objects *X with known classes Y, i.e.* (X, Y) = {(x_1,y_1), (x_2, y_2)… ($x_{n,}$,y_n)}. E.g {(doc1, sports), (doc2, sports), (doc3, music) …}.
  - **Testing data**: a description of an unseen instance, D_{new} *e.g. a new document **without** class label information*
- Goal:
  - Predict the category/class of D_{new}: *y*(*x*)∈*Y,* where *y*(*x*) is a *classification function, aka trained model***,** whose domain is *X* and whose range is *Y*.

Supervised Classifier



Some flaws, but overall, GREAT, 25 Oct 2011

The best? Maybe so....., 26 Oct 2011

A limited device, 29 Oct 2011

By A reviewer (United Kingdom) - See all my reviews

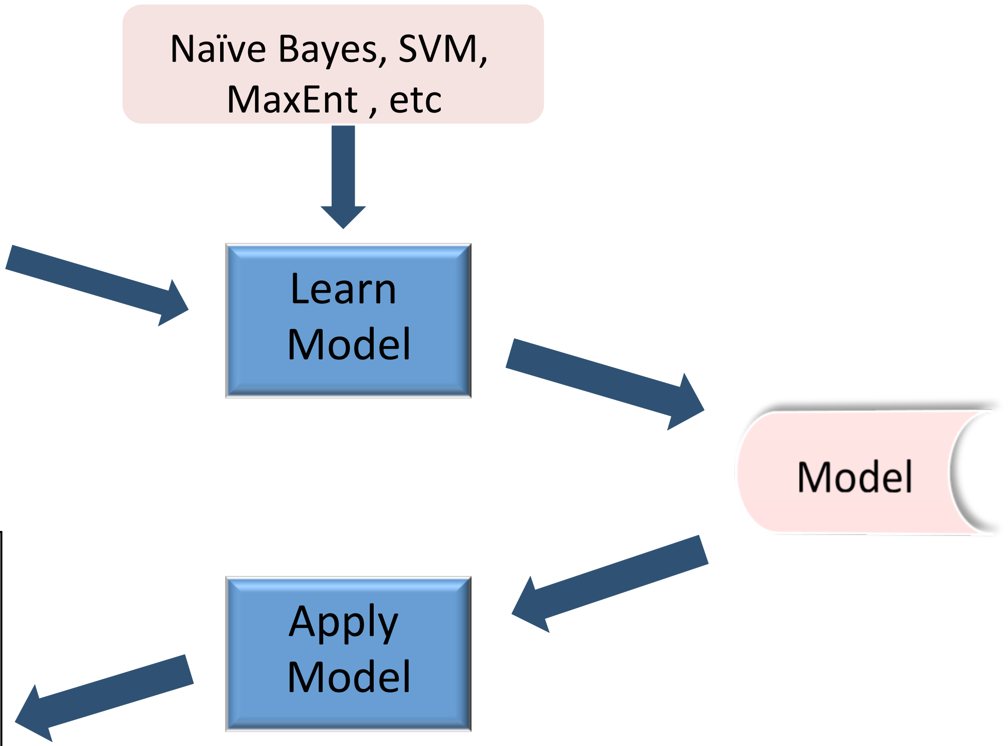
This review is from: Apple iPhone 4S 16GB Black (Electronics)

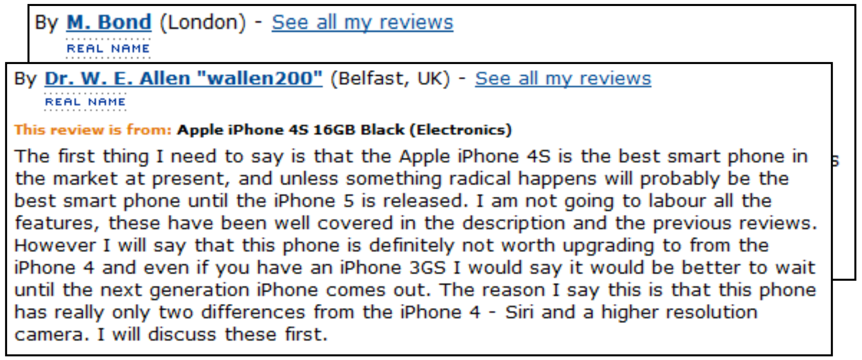
I'm not "an Apple fanatic with the ethos 'if it aint Apple don't bother'", so you will get something balanced here, but I will say that I purchased an iPhone 4S with a strong desire to like it. I really tried my best and intended to use it exclusively, but due to me having already experienced Android, it had to go back to the shop.

I don't care who makes a product or what their marketing is like, I care about how versatile and useful the product is and in this respect I just couldn't avoid the obvious conclusion that this device is deficient. Shock, horror, Apple?! Yes, they don't walk on water, they just have slick marketing.

What were the problems? I'll just list those I discovered in the few days using the phone. Some of these I suppose are going to be subjective but I'll just tell you how I found it:

Training set





By M. Bond (London) - See all my reviews

REAL NAME

By Dr. W. E. Allen "wallen200" (Belfast, UK) - See all my reviews

REAL NAME

This review is from: Apple iPhone 4S 16GB Black (Electronics)

The first thing I need to say is that the Apple iPhone 4S is the best smart phone in the market at present, and unless something radical happens will probably be the best smart phone until the iPhone 5 is released. I am not going to labour all the features, these have been well covered in the description and the previous reviews. However I will say that this phone is definitely not worth upgrading to from the iPhone 4 and even if you have an iPhone 3GS I would say it would be better to wait until the next generation iPhone comes out. The reason I say this is that this phone has really only two differences from the iPhone 4 - Siri and a higher resolution camera. I will discuss these first.

Test set

- Rely on syntactic or co-occurrence patterns in large text corpora

What can you do with classification?

Applications:

- Topic classification
  - Given a news article, predict the topic of the article, e.g., *finance* vs. *sports*
- Spam Classification
  - Given an email, predict whether it is spam or not
- Medical Diagnosis
  - Given a list of symptoms, predict whether a patient has cancer or not
- Weather
  - Based on temperature, humidity, etc… predict if it will rain tomorrow

More Text Classification Examples:

Many search engine functionalities use classification

Assign labels to each document or web-page:

- Labels are most often topics such as Yahoo-categories

  e.g., "finance," "sports," "news>world>asia>business"
- Labels may be genres

  e.g., "editorials" "movie-reviews" "news“
- Labels may be opinion on a person/product

  e.g., “like”, “hate”, “neutral”
- Labels may be domain-specific

  e.g., "interesting-to-me" : "not-interesting-to-me”

  e.g., “contains adult language” : “doesn’t”

  e.g., language identification: English, French, Chinese, …

  e.g., search vertical: about Linux versus not

  e.g., “link spam” : “not link spam”

Supervised Learning for Classification

- Many commercial systems (partly) rely on classification techniques (MSN, Verity, Enkata, Yahoo!, …)
  - Naive Bayes (simple, common method)
  - Decision trees (intuitive, powerful)
  - Support-vector machines (new, more powerful)
  - plus many other methods …
- No free lunch: requires hand-classified training data
- But data can be built up (and refined) by amateurs, e.g., Amazon Mechanical Turk
- Note that many commercial systems use a mixture of methods

The Bayes Rule

$$\boxed{\underbrace{p(Y \mid X_1,\ldots,X_n)}_{\text{Posterior}} = \frac{\overbrace{P(X_1,\ldots,X_n \mid Y)}^{\text{Likelihood}}\,\overbrace{P(Y)}^{\text{Prior}}}{\underbrace{P(X_1,\ldots,X_n)}_{\text{Normalization Constant}}}}$$

$P(Y)$:	Prior belief (probability of hypothesis Y before seeing any data)
$P(X_1,\ldots,X_n \mid Y)$:	Likelihood (probability of the data if the hypothesis Y is true)
$P(X_1,\ldots,X_n)$:	Data evidence (marginal probability of data)
$P(Y \mid X_1,\ldots,X_n)$:	Posterior (probability of hypothesis Y after having seen the data)

Bayes Classifiers

Task: Given a **trained Bayes classifier**, predict a new instance *D* based on a tuple of attribute values into one of the classes $y_j \in Y$

P('sports' |"The football match of the year ….")

$$D = \langle x_1, x_2, \ldots, x_n \rangle$$

Apply Bayes rule!

$$y_D = \operatorname*{argmax}_{y_j \in Y} P(y_j \mid x_1, x_2, \ldots, x_n)$$

$$= \operatorname*{argmax}_{y_j \in Y} \frac{P(x_1, x_2, \ldots, x_n \mid y_j) P(y_j)}{P(x_1, x_2, \ldots, x_n)}$$

$$\propto \operatorname*{argmax}_{y_j \in Y} P(x_1, x_2, \ldots, x_n \mid y_j) P(y_j)$$

Can be learned from Training data!

argmax: return the argument value for which the probability expression takes the maximum value

Bayes Classifiers

- $P(y_j)$
  - The probability of class label y_j, e.g. Prob(politics)
  - Can be estimated from the frequency of classes in the training examples.
- $P(x_1,x_2,\ldots,x_n|y_j)$
  - The probability of generating observed instances/data given a class label y_j .
  - For instance, given a class label '*sports*', what is the probability of observing document *d*,

    e.g. Prob("The football match of the year ….”| 'sports')
  - Could only be estimated if a very, very **large** number of training examples was available
  - Why??

Issues with the Bayes Model

- The issue with **explicitly modeling** $P(x_1,x_2,\ldots,x_n|y_j)$
  - Usually way too many parameters parameters
  - We' ll run out of space
  - We' ll run out of time, because …

$O(|X|^n \bullet |Y|)$

$$
\begin{aligned}
& P(x_1,x_2,\ldots,x_n \mid y_j) \\
&= P(x_1 \mid y_j)P(x_2,\ldots,x_n \mid y_j,x_1) \\
&= P(x_1 \mid y_j)P(x_2 \mid y_j,x_1)P(x_3,\ldots,x_n \mid y_j,x_1,x_2) \\
&= P(x_1 \mid y_j)P(x_2 \mid y_j,x_1)\ldots.P(x_n \mid y_j,x_1,x_2,\ldots,x_n)
\end{aligned}
$$

The Independence Assumption

- Assume A and B are Boolean Random variables. Then

“A and B are independent”

if and only if

$$P(A|B) = P(A)$$

“A and B are independent” is often notated as

$$A \perp B$$

Naïve Bayes Model

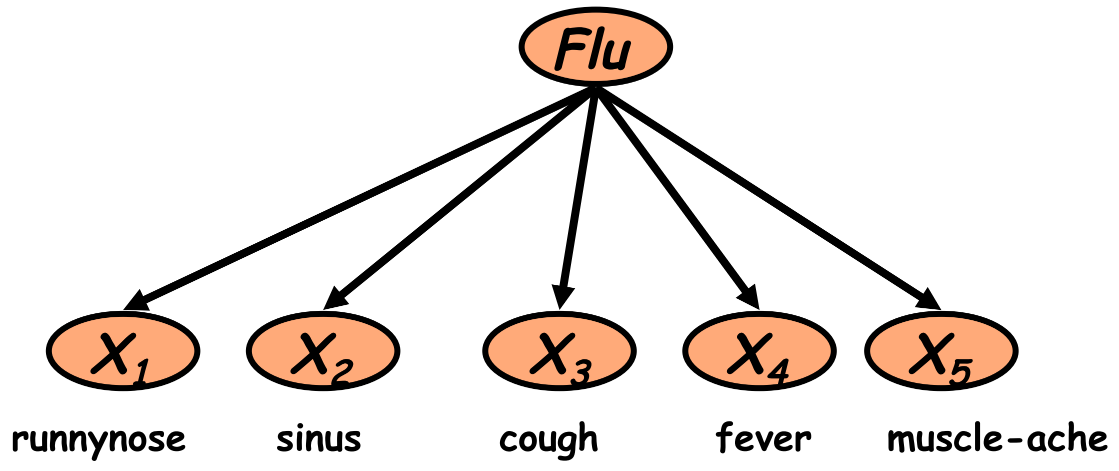
- The problem with **explicitly modeling** $P(X_1,\ldots,X_n|Y)$ is that there are usually way too many parameters:

$$
\begin{aligned}
&P(x_1, x_2, \ldots, x_n \mid y_j) \\
&= P(x_1 \mid y_j) P(x_2, \ldots, x_n \mid y_j, x_1) \\
&= P(x_1 \mid y_j) P(x_2 \mid y_j, x_1) P(x_3, \ldots, x_n \mid y_j, x_1, x_2) \\
&= P(x_1 \mid y_j) P(x_2 \mid y_j, x_1) \ldots . P(x_n \mid y_j, x_1, x_2, \ldots, x_n)
\end{aligned}
$$

- **Solution**: assume that all features are independent **given the class label Y**, yielding the <u>naïve Bayes version</u>

$$P(x_1, x_2, \ldots, x_n \mid y_j) = \prod_{i=1}^{n} P(x_i \mid y_j)$$

The Independence Assumption



- Features (term presence) are *independent* of each other given the class:

$$P(X_1,\ldots,X_5 \mid Y) = P(X_1 \mid Y) \bullet P(X_2 \mid Y) \bullet \cdots \bullet P(X_5 \mid Y)$$

Why is this useful?

- # of parameters for modeling $P(X_1,\ldots,X_n|Y)$, i.e. **Bayes version**:
  - **$2(2^n-1)$**
- # of parameters for modeling $P(X_1|Y),\ldots,P(X_n|Y)$, i.e., **naïve Bayes version**
  - **$2n$**

NB Model Parameters

- For the Naive Bayes classifier, we need to “learn” two functions:
  - the likelihood
  - the prior

$$P(Y|X_1,\ldots,X_n) = \frac{\overbrace{P(X_1,\ldots,X_n|Y)}^{\text{Likelihood}}\,\overbrace{P(Y)}^{\text{Prior}}}{\underbrace{P(X_1,\ldots,X_n)}_{\text{Normalization Constant}}}$$

How to estimate these parameters??? We will see later on

Multinomial Naïve Bayes Training

Learning Algorithm for Text Classification

- From training corpus, extract *Vocabulary*
- Calculate $P(y_j)$ *each* y_j *in* Y

$$P(y_j) = \frac{|docs_j|}{|\text{total \# documents}|}$$

- Calculate $P(x_k \mid y_j)$ terms
  - for each word x_k in <u>vocabulary</u>
  - n_k : number of occurrences of x_k in a subset of documents for which the target class is y_j
  - n : total number of word tokens in a subset of documents for which the target class is y_j

$$P(x_k \mid y_j) = \frac{n_k}{n}$$

Smoothing

Note: the vocabulary is derived from the entire training corpus for all possible labels.

- This means that some words may only appear in some particular classes → $n_k = 0$

$$P(x_k \mid y_j) = \frac{n_k}{n}$$

- Calculate $P(x_k \mid y_j)$ terms with **add one smoothing**
  - n_k : number of occurrences of x_k in a subset of documents for which the target class is y_j
  - n : total number of word tokens in a subset of documents for which the target class is y_j

$$P(x_k \mid y_j) = \frac{n_k + 1}{n + |Vocabulary|}$$

Naïve Bayes: Classifying

- For all word positions in the testing document *d* which contain tokens found in *Vocabulary*
- Return y_d, where

$$yd = \operatorname*{argmax}_{y_j \in Y} P(y_j) \prod_{i \in positions} P(x_i \mid y_j)$$

Exercise

	docID	words in document	in $c = China$?
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

- Estimate parameters of Naive Bayes classifier
- Classify test document

$$P(y_j) = \frac{|docs_j|}{|\text{total \# documents}|} \qquad P(x_k \mid y_j) = \frac{n_k + 1}{n + |Vocabulary|}$$

Example: Training Phase

Model parameter estimation

Priors: $\hat{P}(c) = 3/4$ and $\hat{P}(\overline{c}) = 1/4$ Conditional probabilities:

$$\begin{aligned}
\hat{P}(\text{CHINESE}|c) &= (5+1)/(8+6) = 6/14 = 3/7 \\
\hat{P}(\text{TOKYO}|c) = \hat{P}(\text{JAPAN}|c) &= (0+1)/(8+6) = 1/14 \\
\hat{P}(\text{CHINESE}|\overline{c}) &= (1+1)/(3+6) = 2/9 \\
\hat{P}(\text{TOKYO}|\overline{c}) = \hat{P}(\text{JAPAN}|\overline{c}) &= (1+1)/(3+6) = 2/9
\end{aligned}$$

Example: Testing Phase

Classification

$$\begin{aligned}\hat{P}(c|d_5) &\propto 3/4 \cdot (3/7)^3 \cdot 1/14 \cdot 1/14 \approx 0.0003\\ \hat{P}(\overline{c}|d_5) &\propto 1/4 \cdot (2/9)^3 \cdot 2/9 \cdot 2/9 \approx 0.0001\end{aligned}$$

Thus, the classifier assigns the test document to *c* = *China*. The reason for this classification decision is that the **three** occurrences of the positive indicator CHINESE in d_5 outweigh the occurrences of the **two** negative indicators JAPAN and TOKYO.

What you should know

- Why it is called naïve?
  - The independence assumption
- How to build a naïve Bayes classifier
  - What happen in training time
  - What happen in testing time
- How to deal with unseen features in training example
  - Smoothing

Conclusions

- Naïve Bayes is:
  - Really easy to implement and often works well
  - Often a good first thing to try
- Actually, the Naïve Bayes assumption is almost never true
- Still… Naïve Bayes often performs surprisingly well even when its assumptions do not hold

Support Vector Machine

Chenghua Lin
Chenghua.Lin@abdn.ac.uk

Outline

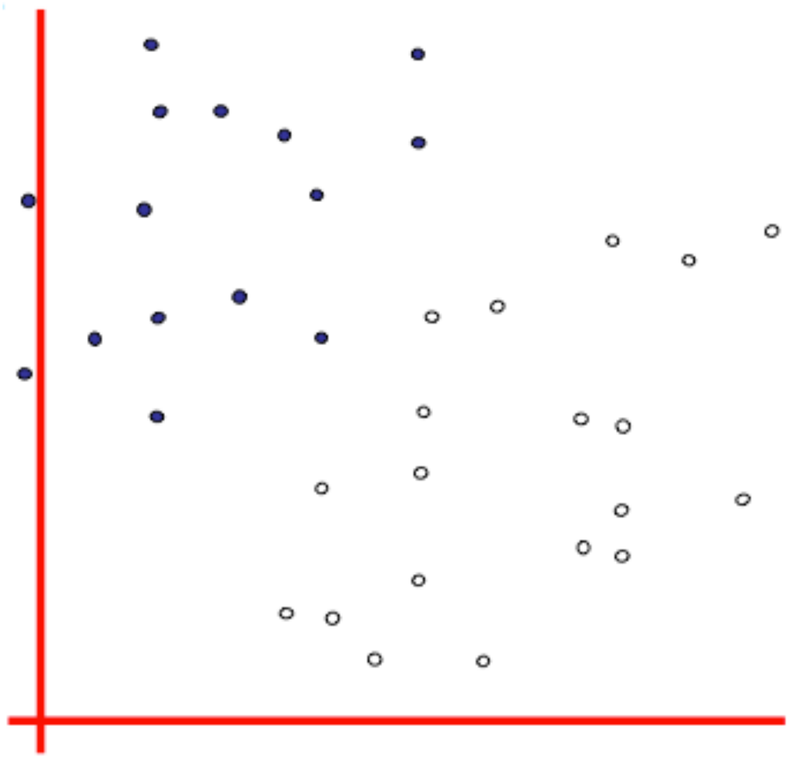
- Probabilistic and Bayesian Analytics
- Classification
  - Naïve Bayes Classifier
  - Support vector machines (SVM)
  - Decision Trees
- Association Rule Mining
- Feature Selection
- Visualization I and II
- Case study
- Data Mining Issues

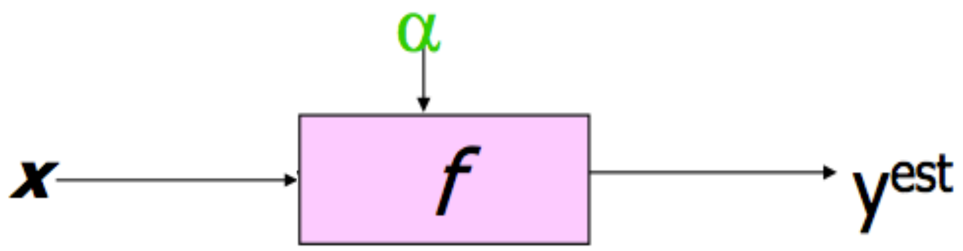
What You Should Know

- Linear SVMs
- The definition of a maximum margin classifier
- What QP (Quadratic Programming) can do for you
  - for this class, you don’t need to know how it does it
- How we deal with noisy data, i.e. misclassified data
- How we permit non-linear boundaries
- How SVM Kernel functions permit us to pretend we’re working with ultra-high-dimensional basis- function terms

Linear Classifiers

- ● denotes +1
- ○ denotes -1





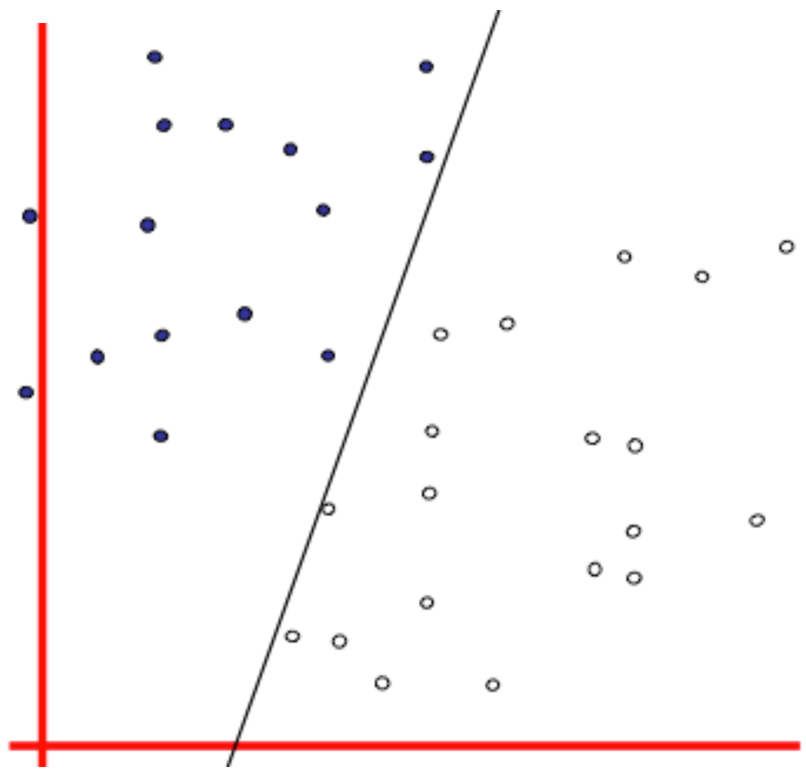
$f(\boldsymbol{x}, \boldsymbol{w}, b) = sign(\boldsymbol{w} \cdot \boldsymbol{x} - b)$

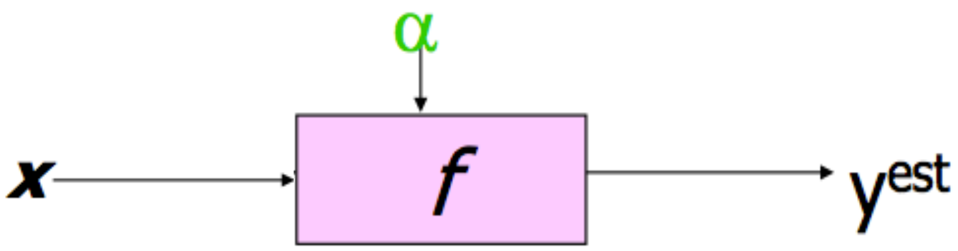
How would you classify this data?

Linear Classifiers

● denotes +1

○ denotes -1

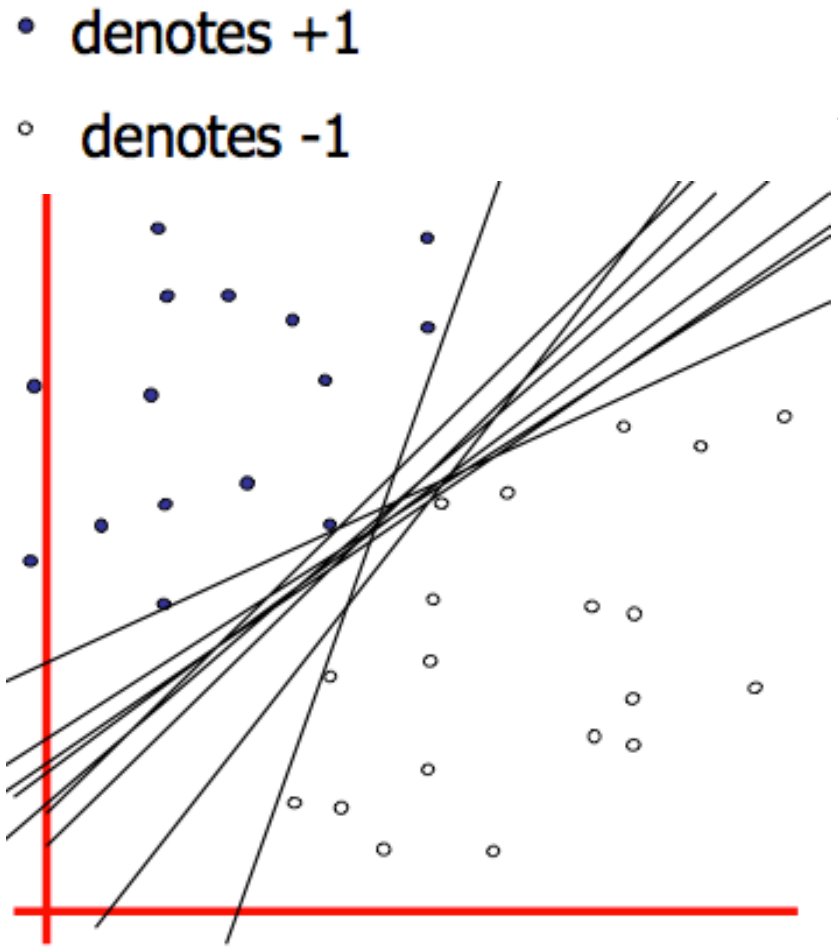


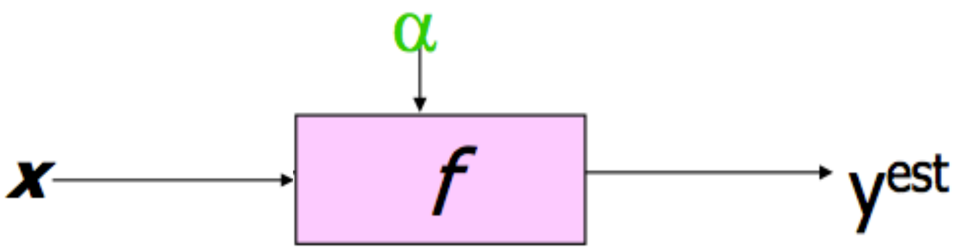


$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}\cdot\mathbf{x} - b)$

How would you classify this data?

Linear Classifiers



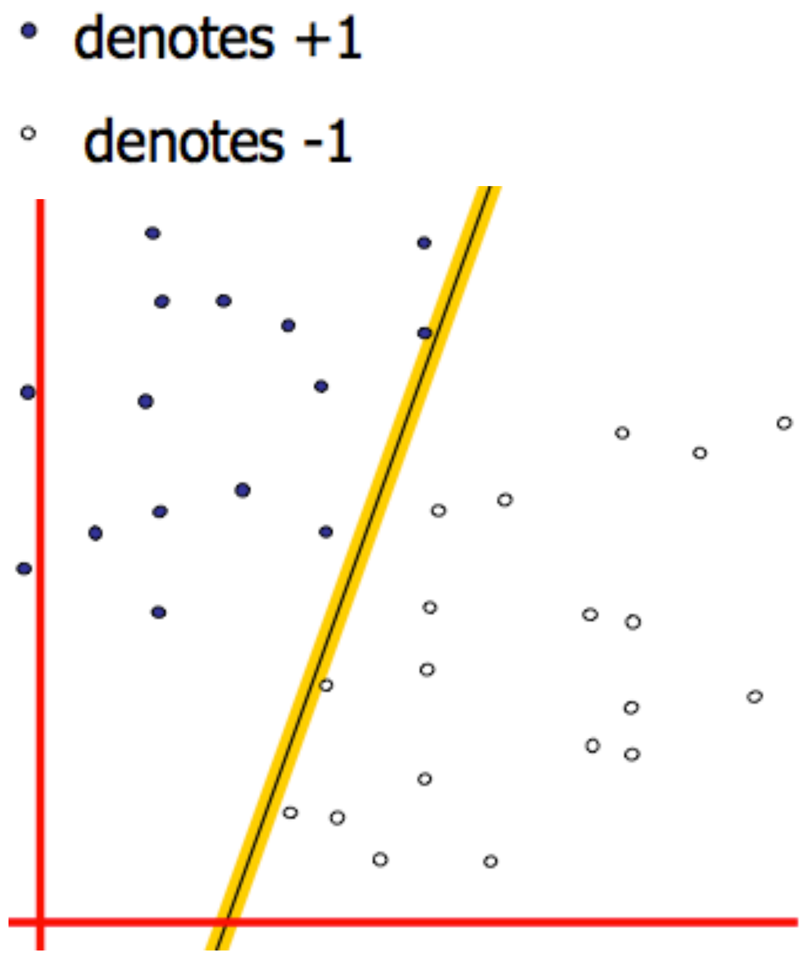


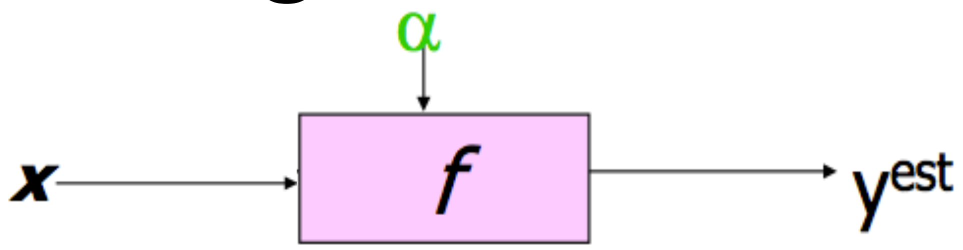
$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} - b)$

Any of these would be fine …

… but which is best?

Classifier Margin

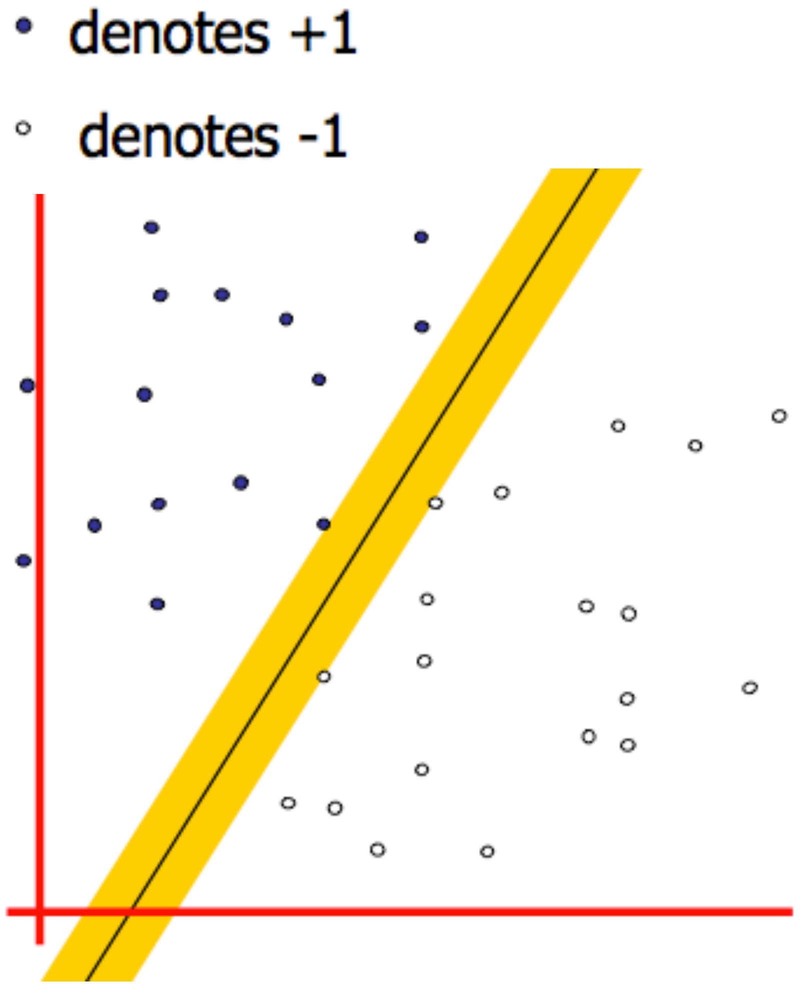


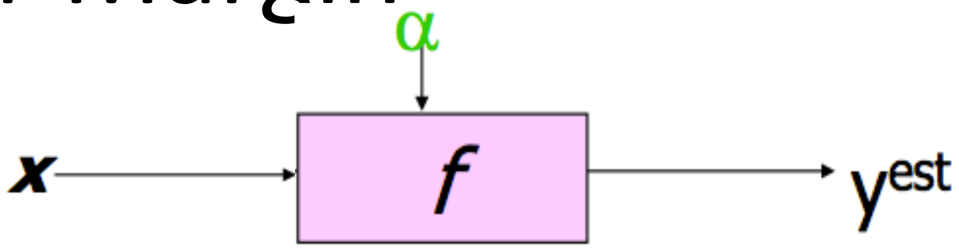


$f(\mathbf{x},\mathbf{w},b) = \mathrm{sign}(\mathbf{w} \cdot \mathbf{x} - b)$

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

Classifier Margin



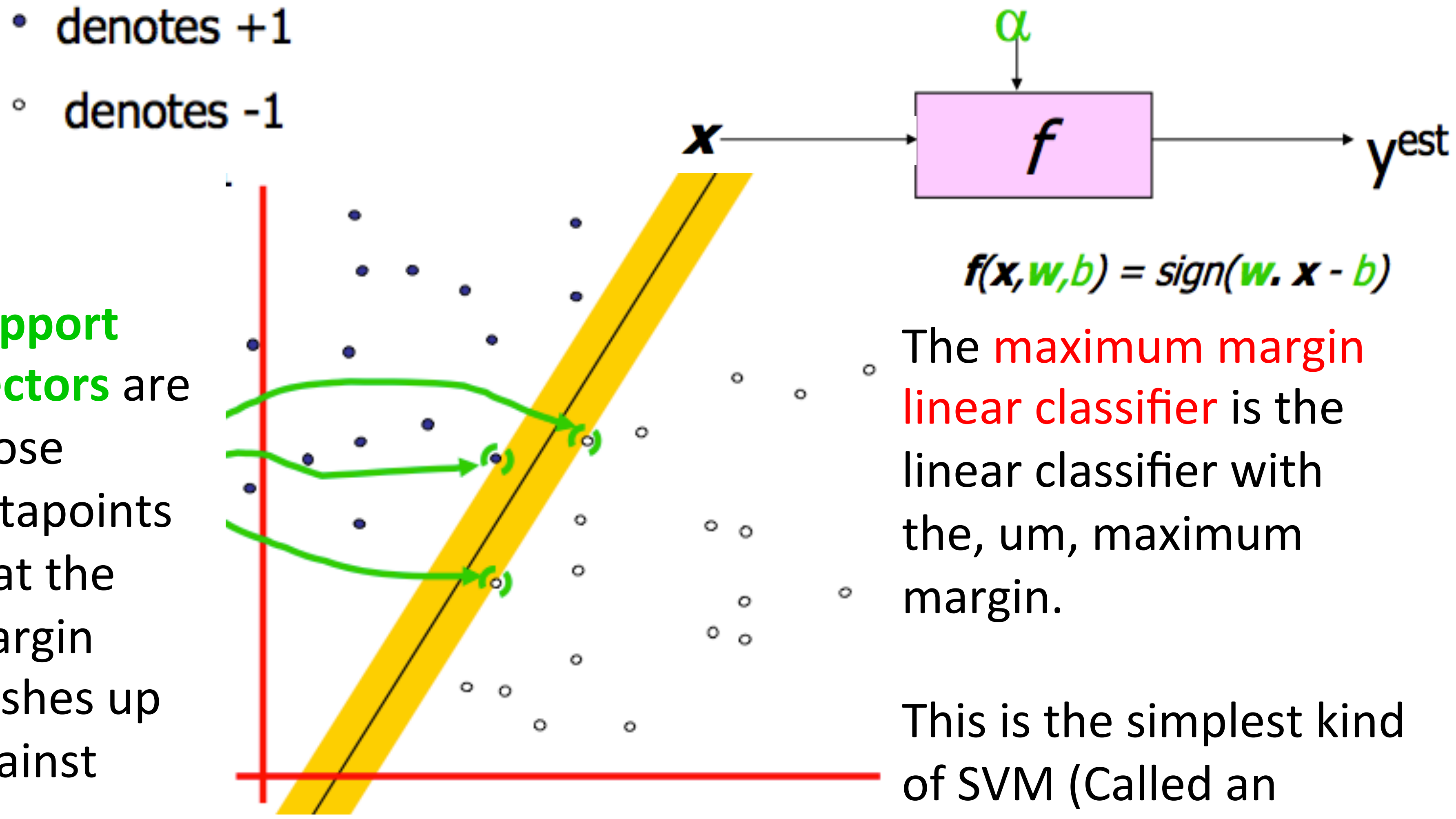


$f(x,w,b) = sign(w. x - b)$

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

This is the simplest kind of SVM (Called an LSVM, i.e. Linear SVM)

Maximum Margin



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

Support Vectors are those datapoints that the margin pushes up against

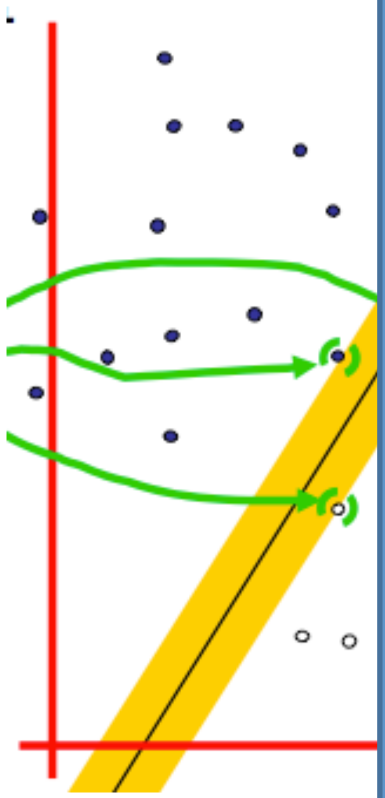
The maximum margin linear classifier is the linear classifier with the, um, maximum margin.

This is the simplest kind of SVM (Called an LSVM, i.e. Linear SVM)

Why Maximum Margin

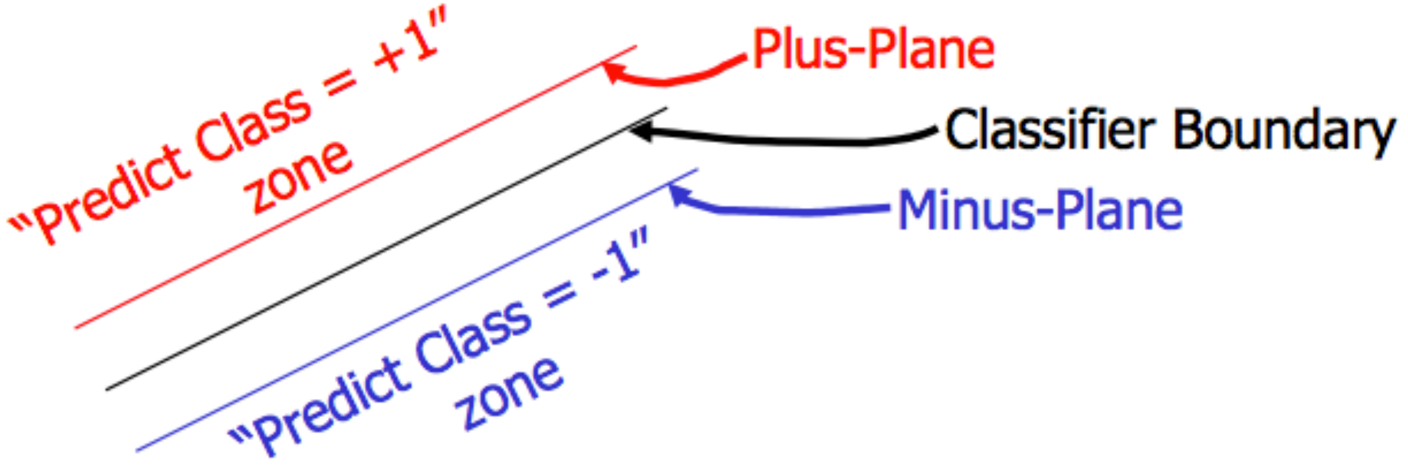
- ● denotes +1
- ○ denotes -1

Support Vectors are those datapoints that the margin pushes up against



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

Specifying a line and margin



- Plus-plane = {**x**:**w**.**x**+b=+1}
- Minus-plane= {**x**:**w**.**x**+b=-1}

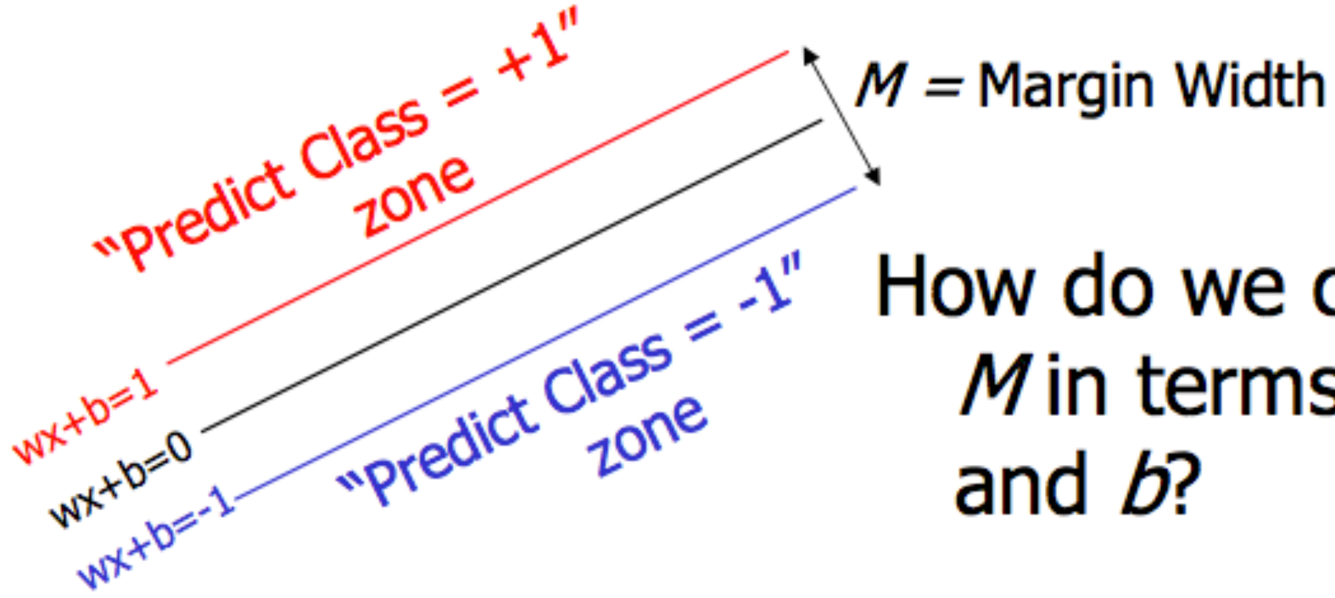
Classify as..

+1 if **w**.**x**+b>=1

-1 if **w**.**x**+b<=-1

Universe Explodes if -1 < **w** . **x** + b < 1

Compute the margin width

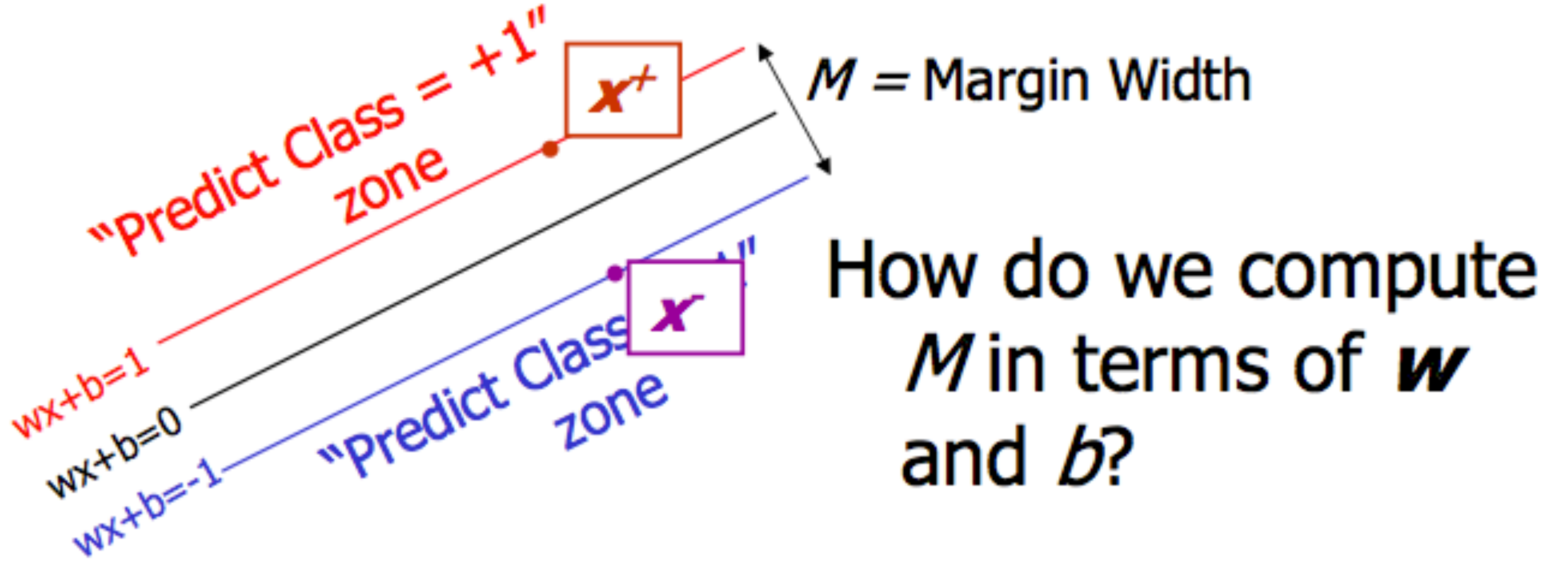


How do we compute *M* in terms of ***w*** and *b*?

- Plus-plane = {**x**:**w**.**x**+b=+1}
- Minus-plane= {**x**:**w**.**x**+b=-1}

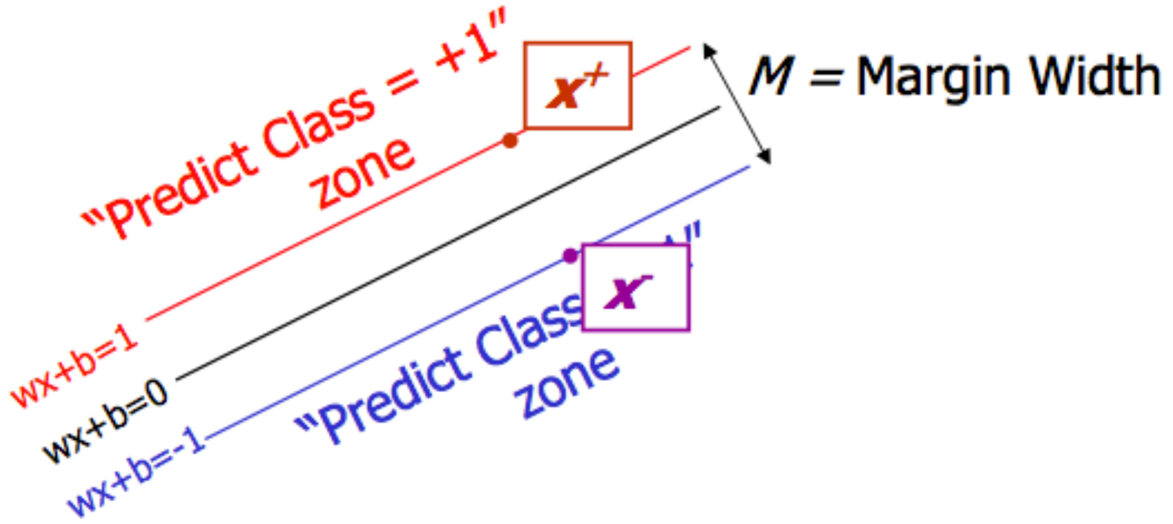
Claim: The vector **w** is perpendicular to the Plus Plane. Why?

Computing the margin width


"Predict Class = +1" zone
x^+
M = Margin Width
"Predict Class = -1" zone
x^-
wx+b=1
wx+b=0
wx+b=-1
How do we compute M in terms of **w** and b?

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

Computing the margin width

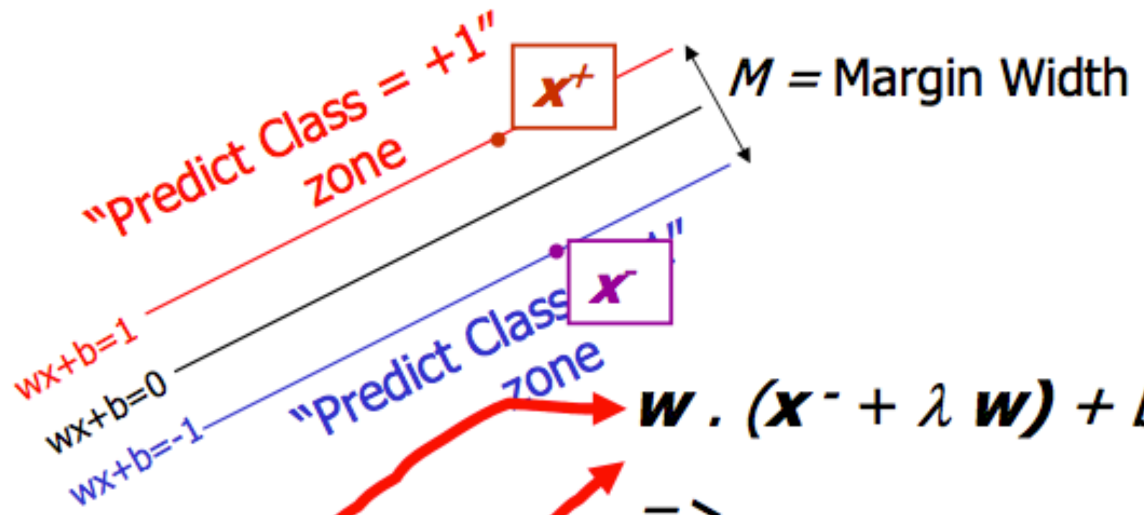


What we know:

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

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

Computing the margin width



What we know:

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

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

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

=>

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

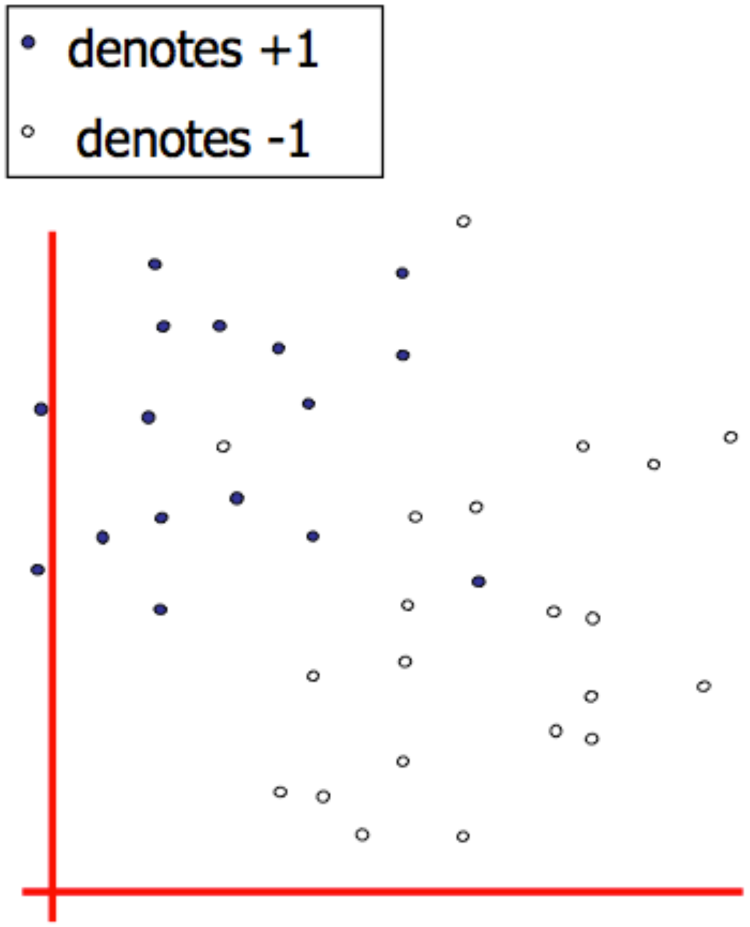
=>

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

=>

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

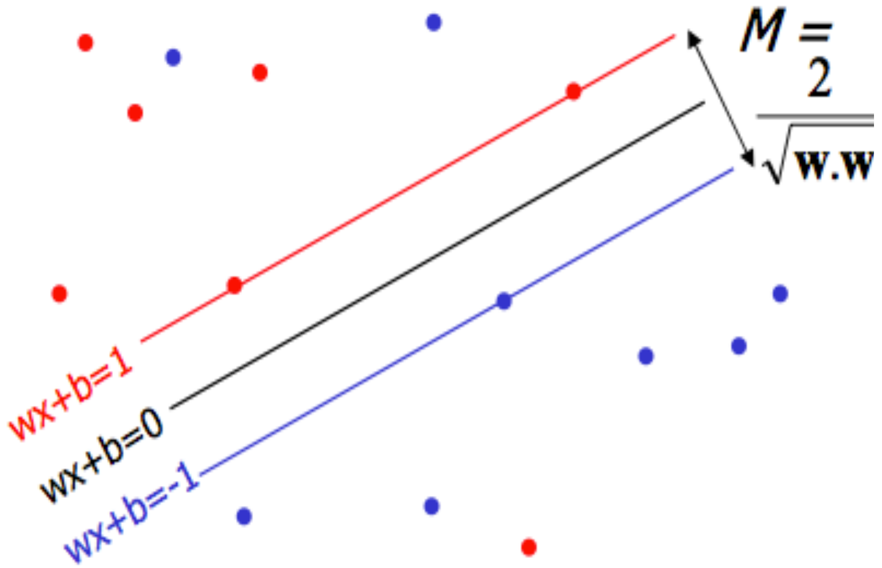
Noise, Uh-oh!



This is going to be a problem!
What should we do?

Minimize: **w.w** + D
where D is the distance of error points to their correct place

Learning Maximum Margin with Noise

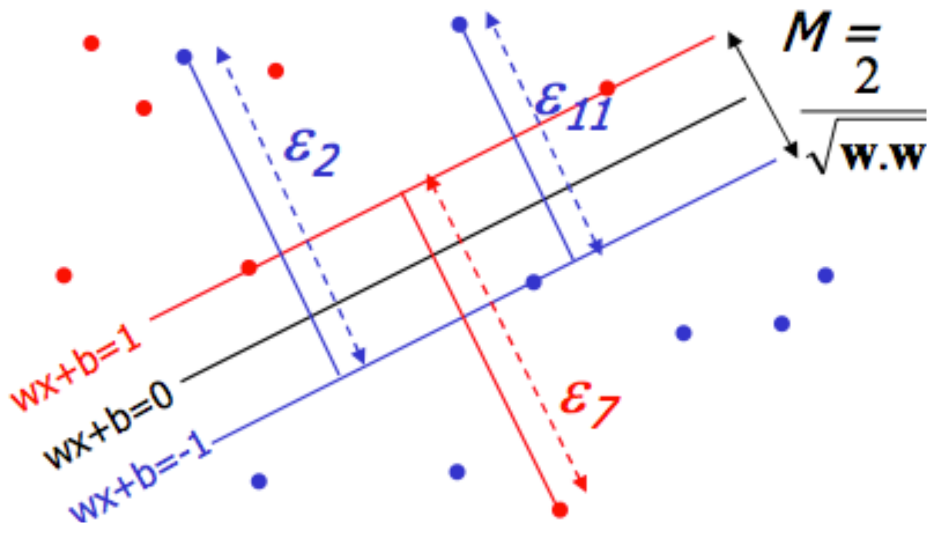


Given guess of W, b, we can

- Compute sum of distances of points to their correct zones
- Compute the margin width. Assume R datapoints, each ($\mathbf{x}_k$,y_k) where y_k = +/- 1

- How many constraints will we have?

Learning Maximum Margin with Noise



What should our quadratic optimization criterion be?

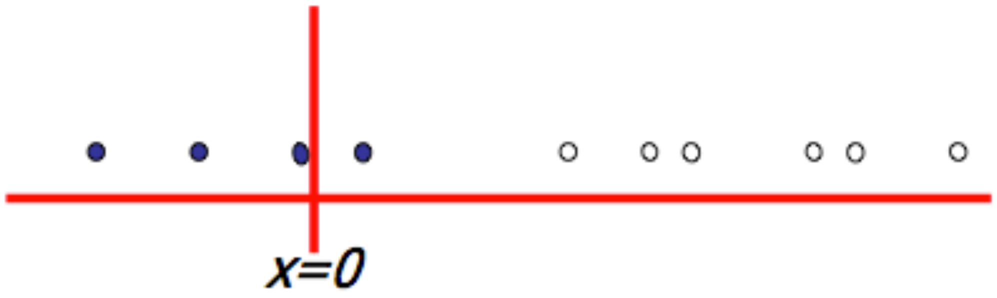
Minimize $\frac{1}{2}\mathbf{w.w} + C\sum_{k=1}^{R}\varepsilon_k$

Given guess of W, b can

- Compute sum of distances of points to their correct zones
- Compute the margin width. Assume R datapoints, each ($\mathbf{x}_k$,y_k) where y_k = +/- 1
- ε: the slack variable
- C: control the trade-off between the slack variable and the margin.

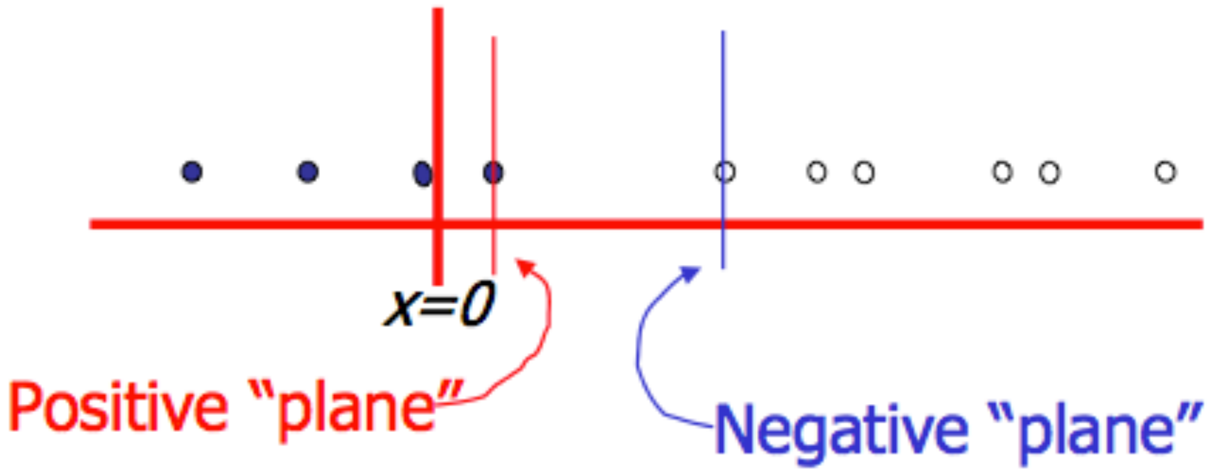
Suppose we’re in 1-dimension

What would SVMs do with this data?



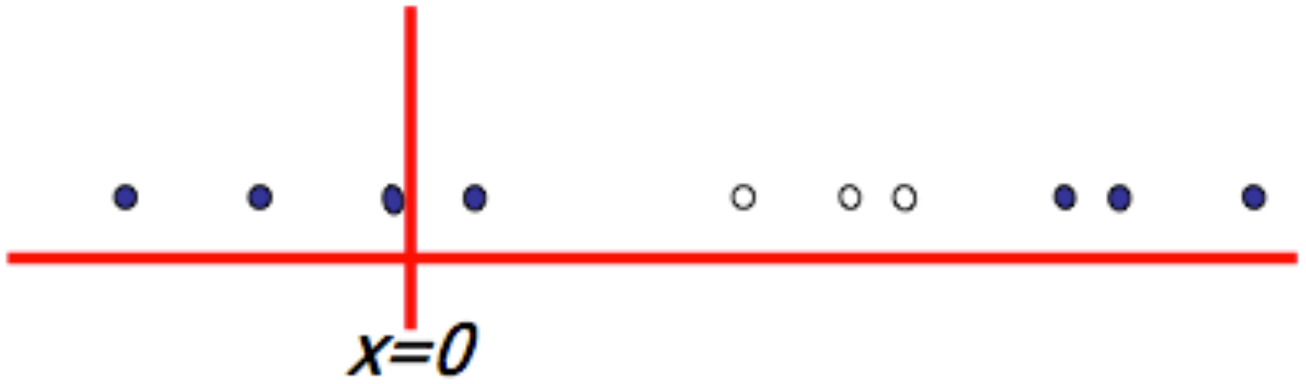
Suppose we're in 1-dimension

Not a big surprise

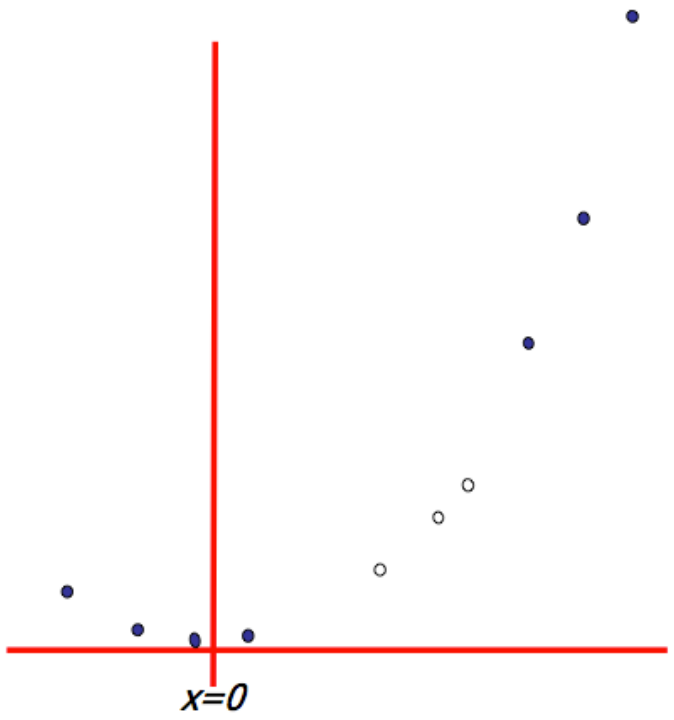


Harder 1-dimensional dataset

What can be done about this? i.e. linear not separable

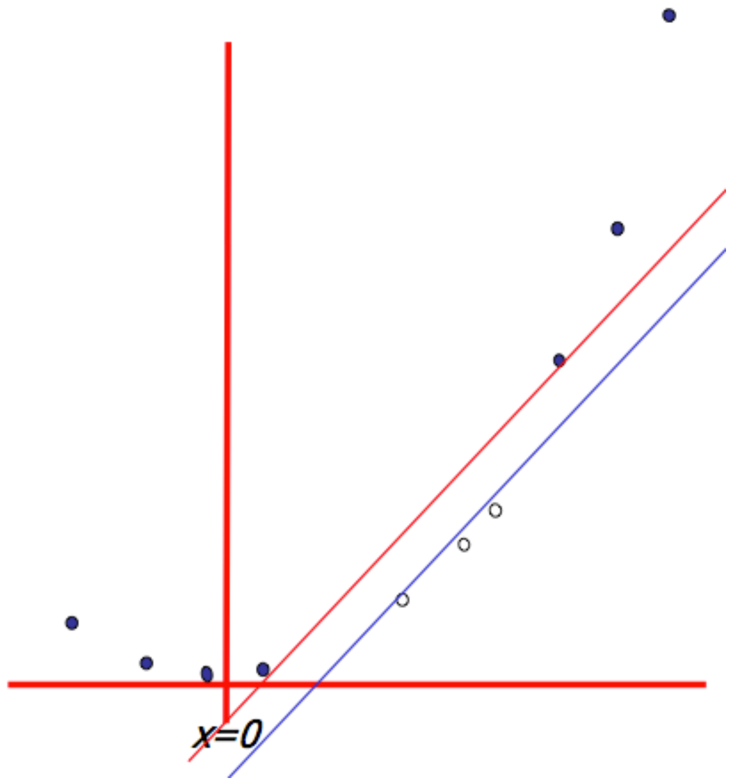


Harder 1-dimensional dataset



- Non-linear basis functions to rescue!
- To project original datapoints to higher dimensions within which datapoints are separable
- $\mathbf{z}_k = (x_k, x_k^2)$

Harder 1-dimensional dataset



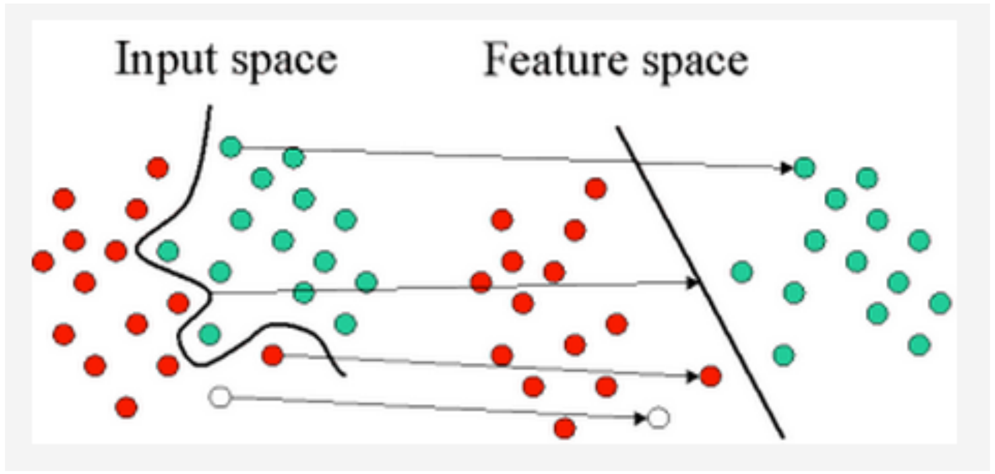
- Non-linear basis functions to rescue!
- To project original datapoints to higher dimensions within which datapoints are separable
- $\mathbf{z}_k = (x_k, x_k^2)$

Common SVM Kernel functions

Kernel trick: SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

Kernel functions:

- polynomial functions
- radial basis functions
- sigmoid functions



Summary

- The definition of a maximum margin classifier
- How Maximum Margin can be turned into a QP problem
- How we deal with noisy data, i.e. misclassified data
  - slack variable
- How we permit non-linear boundaries
  - SVM Kernel functions permit us to pretend we're working with ultra-high-dimensional basis-function terms
  - And in the new feature space, datapoints are linearly separable

Readings

- 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. http://citeseer .nj.nec.com/burges98tutorial.html
- The VC/SRM/SVM Bible:
  - Statistical Learning Theory by Vladimir Vapnik, Wiley- Interscience; 1998

Classifier’s Performance Metrics

Chenghua Lin
Department of Computing Science
University of Aberdeen

Outline

- Accuracy
- Recall, precision and F-measure
- ROC curve
- Cross validation

Metrics for Evaluating Classifier’s Performance

- Error rate of a classifier measures its overall performance
  - Error rate = proportion of errors = number of misclassifications/total number of test instances
- Error rate does not discriminate between different types of errors
- A binary classifier (yes and no) makes two kinds of errors
  - Calling an instance of ‘no’ as an instance of ‘yes’
    - False positives
  - Calling an instance of ‘yes’ as an instance of ‘no’
    - False negatives
- In practice false positives and false negatives have different associated costs
  - Cost of lending to a defaulter is larger than lost-business cost of refusing loan to a non-defaulter
  - Cost of failing to detect fire is larger than the cost of a false alarm

Confusion Matrix

- The four possible outcomes of a binary classifier are usually shown in a confusion matrix
- A number of performance metrics defined using these counts

		Predicted Class	
		Positive'	negative'
Actual Class	positive	TP	FN
	negative	FP	TN

Confusion Matrix

Predicted Class

		Positive'	negative'	
Actual Class	positive	TP	FN	
	negative	FP	TN	

- True Positives (TP)
  - # of correct predictions that an instance is positive
- True Negatives (TN)
  - # of correct predictions that an instance is negative
- False Positives (FP)
  - # of incorrect predictions that an instance is positive
- False Negatives (FN)
  - # of incorrect of predictions that an instance negative

Accuracy

		Predicted Class	
		Positive’	negative’
Actual Class	positive	TP	FN
	negative	FP	TN

- **Accuracy of positive class**: the proportions of positive class instances have been correctly predicted
  - Acc_pos = TP / (TP + FN)
- **Accuracy of negative class**: the proportions of negative class instances have been correctly predicted
  - Acc_pos = TN / (FP + TN)
- **Overall accuracy**: the proportion of the total number of predictions that were correct
  - Acc = (TP + TN) / (TP + FP + FN + TN)

Confusion Matrix: example1

	Spam (Predicted)	Non-Spam (Predicted)	Accuracy
Spam (Actual)	27	6	81.81
Non-Spam (Actual)	10	57	85.07
Overall Accuracy			84

The spam dataset:

- Contains 100 instances
- 33 instances are spam
- 67 instances are non-spam

Accuracy:

- Acc(spam) = 27/(27 + 6) = 81.81%
- Acc(non-spam) =57/(10 + 57) = 85.07%
- Overall_acc = (27 + 57) / (27+6+10+57) = 84%

Confusion Matrix: example2

	Spam (Predicted)	Non-Spam (Predicted)	Accuracy
Spam (Actual)	0	10	??
Non-Spam (Actual)	0	990	??
Overall Accuracy			??

The spam dataset:

- 10 patterns are spam
- 990 pattern are non-spam

Accuracy:

- Acc(spam) = 0/10 = 0%
- Acc(non-spam) =990/990 = 100%
- Overall_acc = (0+990)/(0+10+0+990) = 99%

Issues with accuracy

- The confusion matrix tells us how the classifier is behaving for individual classes.
- Accuracy
  - Work well for (more or less) balanced dataset (e.g., 100 positive and 100 negative data instances)
  - Cannot capture true classifier performance when dataset is highly unbalanced.

Beyond accuracy…

- Recall
  - Aka True Positive rate (TP), or sensitivity
  - Recall = TP/(TP+FN)
- Precision
  - the proportion of the predicted positive instance that were correct (positive predictive value)
  - Precision = TP/(TP + FP)
- F-measure
  - Aka F_1-score, is the harmonic mean of precision and recall
  - Suitable for cases where one of the classes is rare
  - F_1=2x(recall x precision) / (recall + precision)

Confusion Matrix: example3

	Positive (Predicted)	Negative (Predicted)
Positive (Actual)	100	50
Negative (Actual)	150	9700

The dataset:

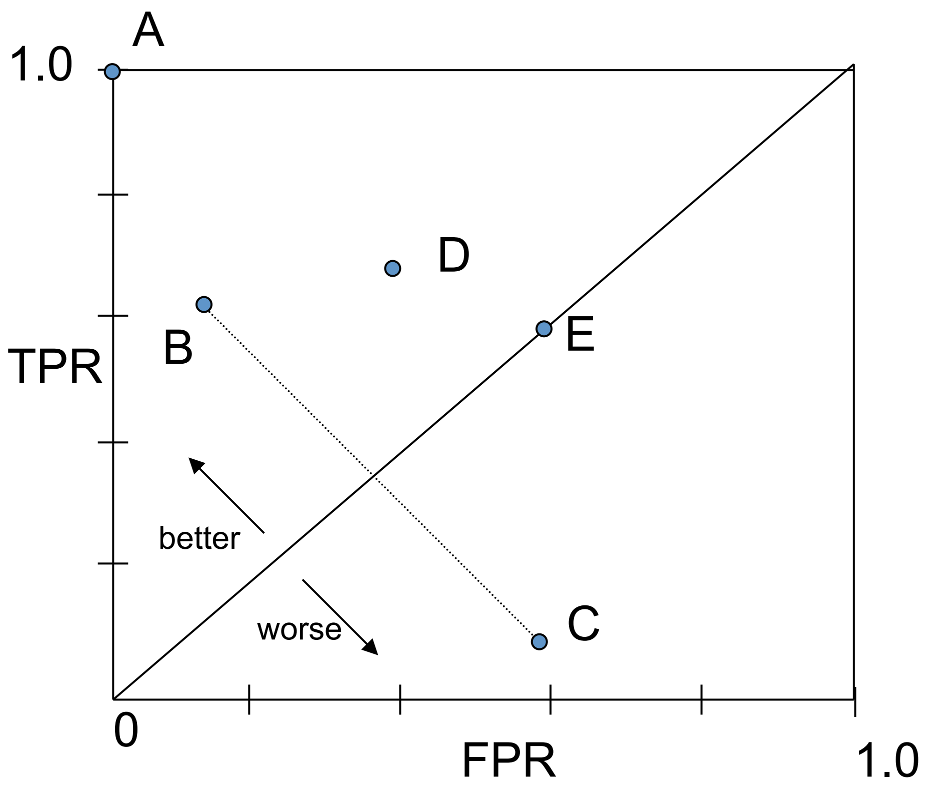
- 150 positive class instances
- 9850 negative class instances

Accuracy:

- Overall_acc = (100+9700)/(100+50+150+9700) = 0.98
- Recall = 100/(100+50) = 0.667
- Precision = 100/(100+150) = 0.4
- F1 = 2 x (0.667x0.4)/(0.667+0.4) = 0.5

ROC - Receiver Operating Characteristic

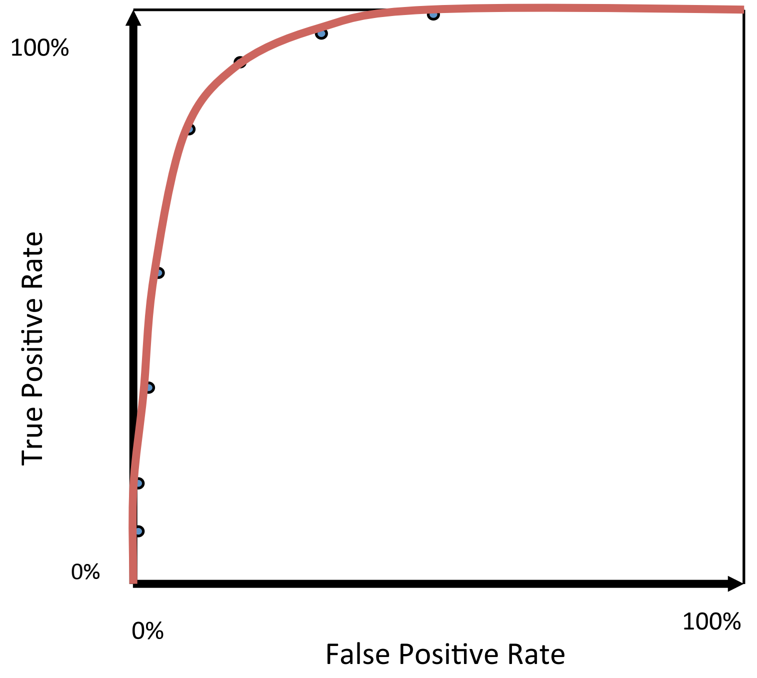
- Particularly a plot of TPR on y-axis against FPR on x axis is known as ROC
- A, B, C, D and E are five classifiers with different TPR and FPR values
- A is the ideal classifier because it has TPR = 1.0 and FPR = 0
- E is on the diagonal which stands for random guess
- C performs worse than random guess
  - But inverse of C which is B is better than D
- Classifiers should aim to be in the northwest



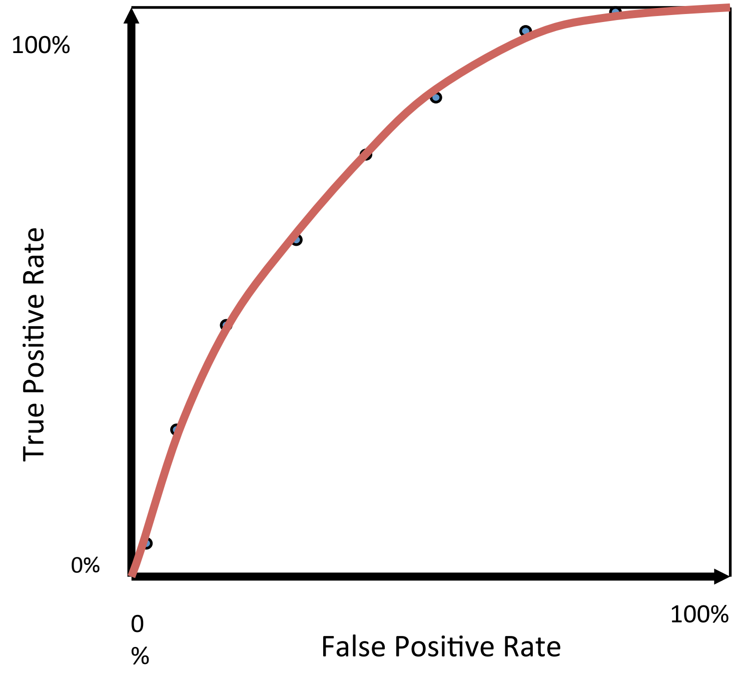
TPR = TP/(TP+FN)
FPR= FP/(FP+TN)

ROC curve comparison

A good test:



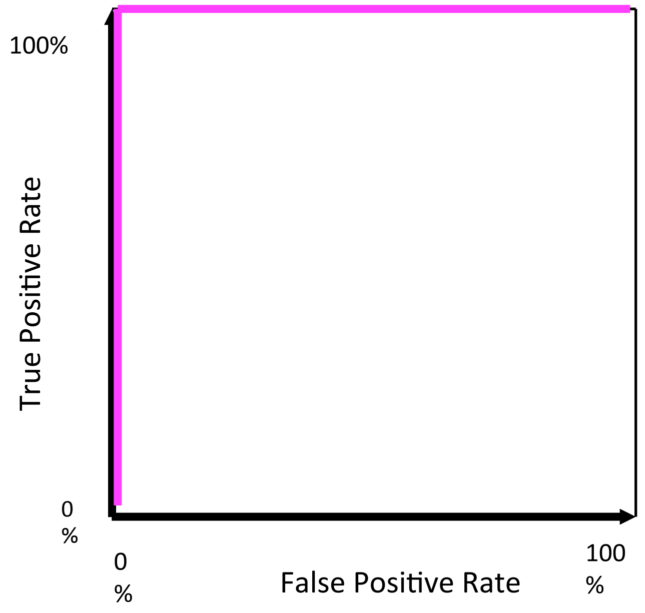
A poor test:



AUC: area under the curve

Summary

Best Test:



The distributions
don' t overlap at all

Testing Classifier

- Testing the classifier on training data is not useful
  - Performance figures from such testing will be optimistic
  - Because the classifier is trained from the very data
- Ideally, a new data set called ‘test set’ needs to be used for testing
  - If test set is large performance figures will be more realistic
  - Creating test set needs experts’ time and therefore creating large test sets is expensive
  - After testing, test set is combined with training data to produce a new classifier
  - Sometimes, a third data set called ‘validation data’ used for fine tuning a classifier or to select a classifier among many
- In practice several strategies used to make up for lack of test data
  - Holdout procedure – a certain proportion of training data is held as test data and remaining used for training
  - Cross-validation
  - Leave-one-out cross-validation

Testing Classifier 2

- Cross Validation
  - Partition the data into a fixed number of folds
  - Use data from each of the partitions for testing while using the remaining for training
  - Every instance is used for testing once
  - 10-fold cross-validation is standard, particularly repeating it 10 times
- Leave-one-out
  - Is n-fold cross-validation, where n is the data size
  - One instance is held for testing while using the remaining for training
  - Results from single instance tests are averaged to obtain the final test result
  - Maximum utilization of data for training
  - No sampling of data for testing, each instance is systematically used for testing
  - High costs involved because classifier is trained n times
  - Hard to ensure representative data for training

Summary

What you should know

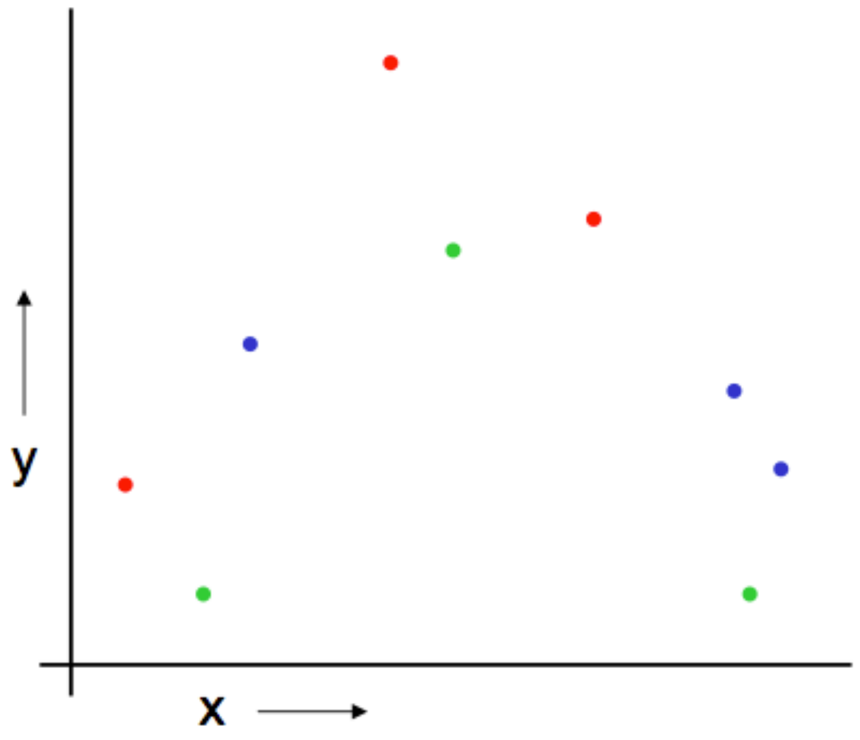
- Accuracy
- Precision, recall, F-measure
- ROC curve
- when to use accuracy or F-measure
- Why you need test data
- Cross validation

Cross-validation for model selection

Outline

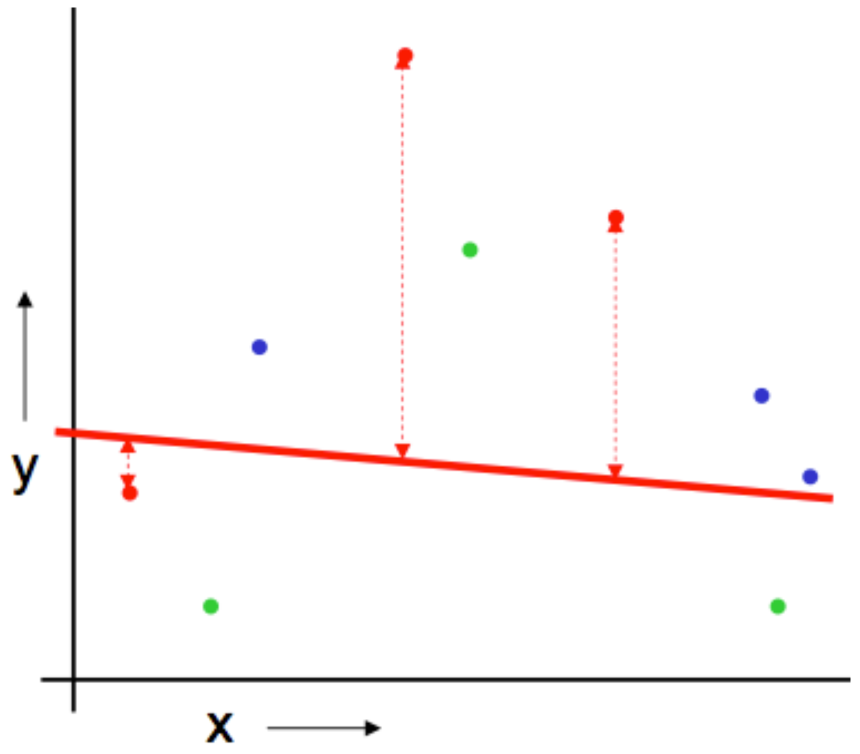
- Test-set cross-validation
- Leave-one-out cross-validation
- k-fold cross-validation

K-fold Cross validation



Randomly break the dataset into k partitions (in our example we'll have k=3 partitions colored Red Green and Blue)

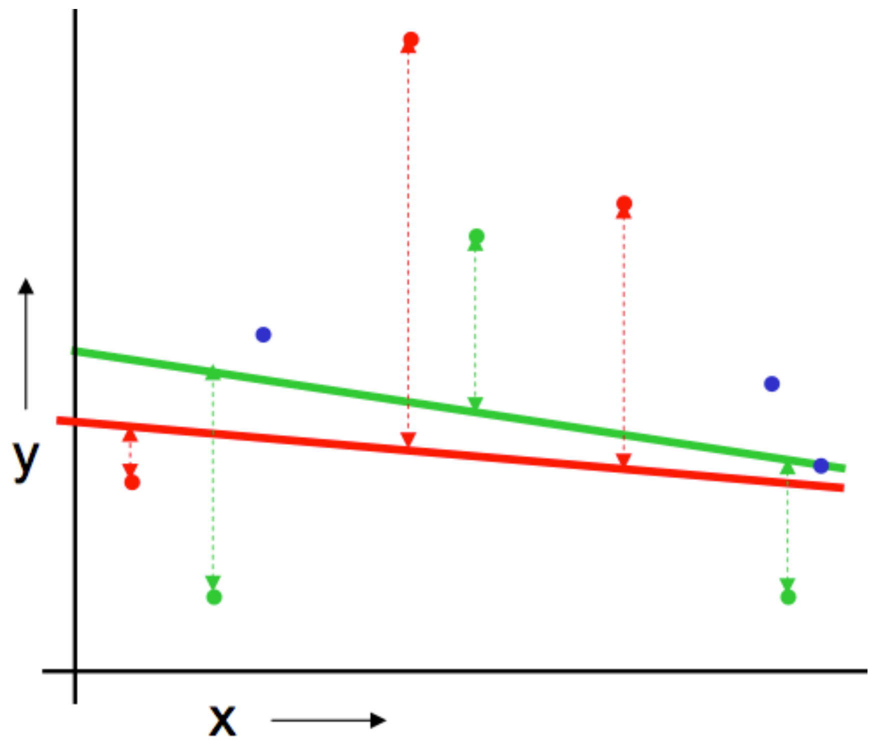
K-fold Cross validation



Randomly break the dataset into k partitions (in our example we'll have k=3 partitions colored Red Green and Blue)

For the red partition: Train on all the points not in the red partition. Find the test-set sum of errors on the red points.

K-fold Cross validation

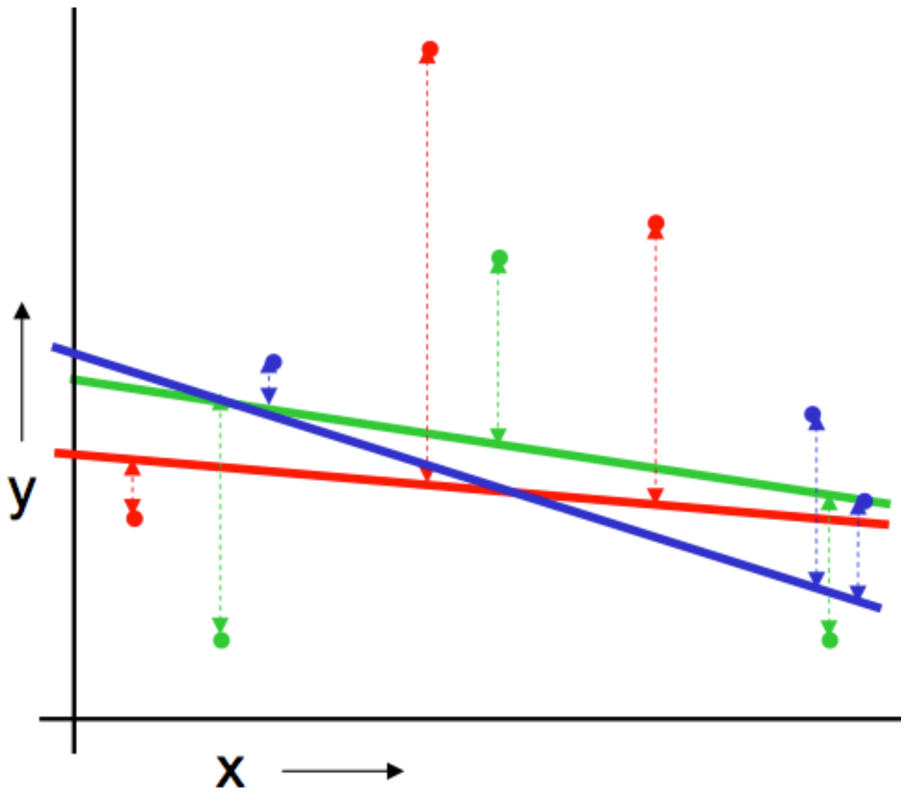


Randomly break the dataset into k partitions (in our example we'll have k=3 partitions colored Red Green and Blue)

For the red partition: Train on all the points not in the red partition. Find the test-set sum of errors on the red points.

For the green partition: Train on all the points not in the green partition. Find the test-set sum of errors on the green points.

K-fold Cross validation



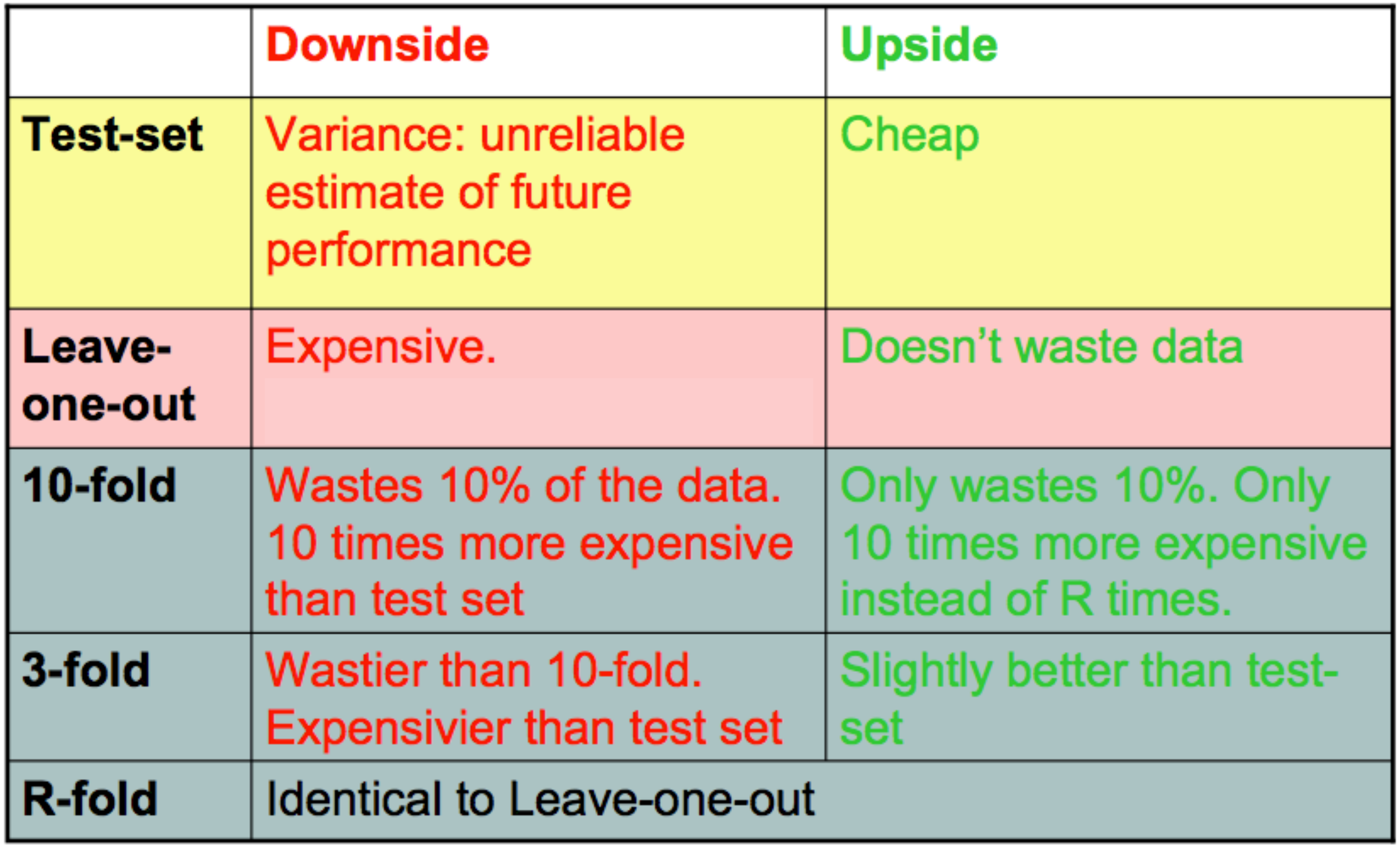
Randomly break the dataset into k partitions (in our example we'll have k=3 partitions colored Red Green and Blue)

For the red partition: Train on all the points not in the red partition. Find the test-set sum of errors on the red points.

For the green partition: Train on all the points not in the green partition. Find the test-set sum of errors on the green points.

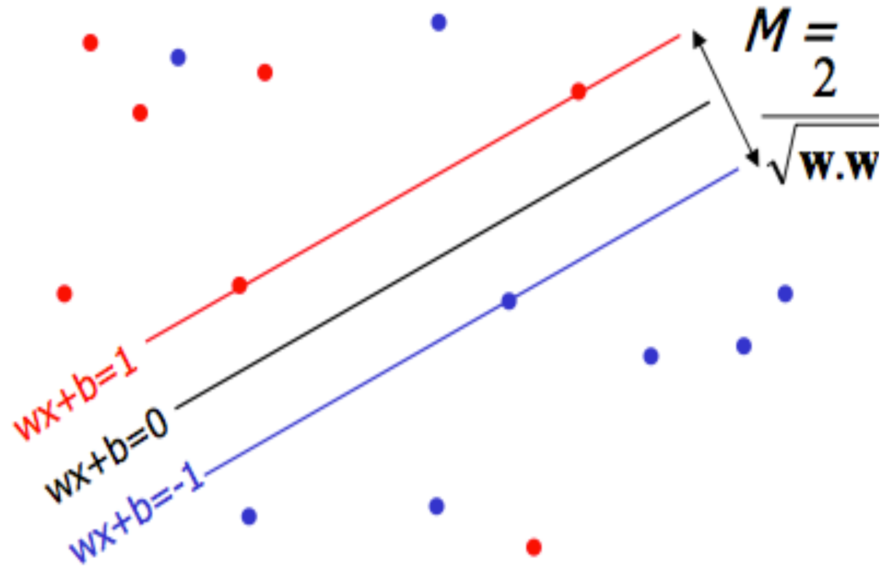
For the blue partition: Train on all the points not in the blue partition. Find the test-set sum of errors on the blue points.

Which kind of validation



	Downside	Upside
Test-set	Variance: unreliable estimate of future performance	Cheap
Leave-one-out	Expensive.	Doesn't waste data
10-fold	Wastes 10% of the data. 10 times more expensive than test set	Only wastes 10%. Only 10 times more expensive instead of R times.
3-fold	Wastier than 10-fold. Expensivier than test set	Slightly better than test-set
R-fold	Identical to Leave-one-out	

Learning Maximum Margin with Noise

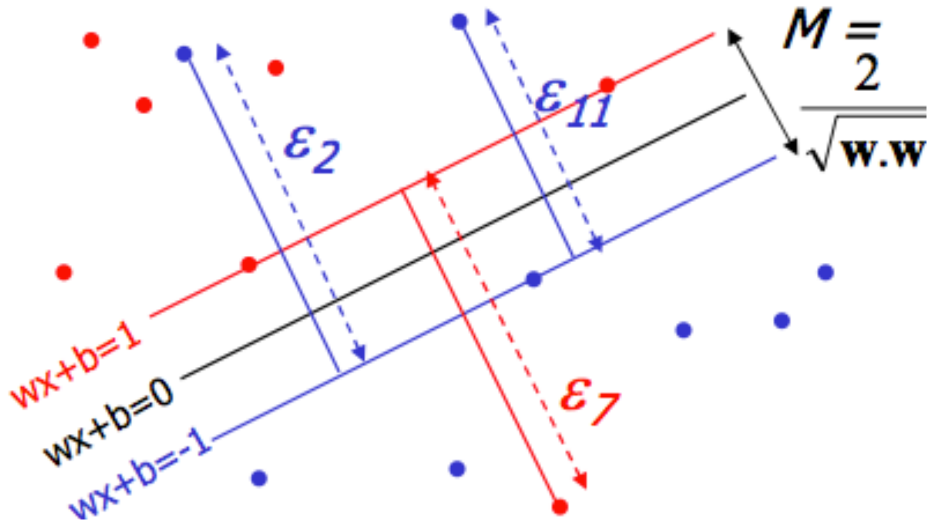


Given guess of W, b, we can

- Compute sum of distances of points to their correct zones
- Compute the margin width. Assume R datapoints, each ($\mathbf{x}_k$,y_k) where y_k = +/- 1

- How many constraints will we have?

Learning Maximum Margin with Noise



What should our quadratic optimization criterion be?

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

Given guess of W, b can

- Compute sum of distances of points to their correct zones
- Compute the margin width. Assume R datapoints, each ($\mathbf{x}_k$,y_k) where y_k = +/- 1
- ε: the slack variable
- C: control the trade-off between the slack variable and the margin.

Decision Tree

Chenghua Lin
Computing Science
University of Aberdeen

The Supervised Classification Task

- Input: Collection of instances with a set of attributes x and a special nominal attribute Y called class attribute
- Output: A model that accurately predicts y from x on previously unseen instances
  - Previously unseen instances are called test set
- Usually input collection is divided into
  - Training set for building the required model
  - Test set for evaluating the model built

Example Data

Class Attribute

Outlook	**Temperature**	**Humidity**	**Windy**	**Play**
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

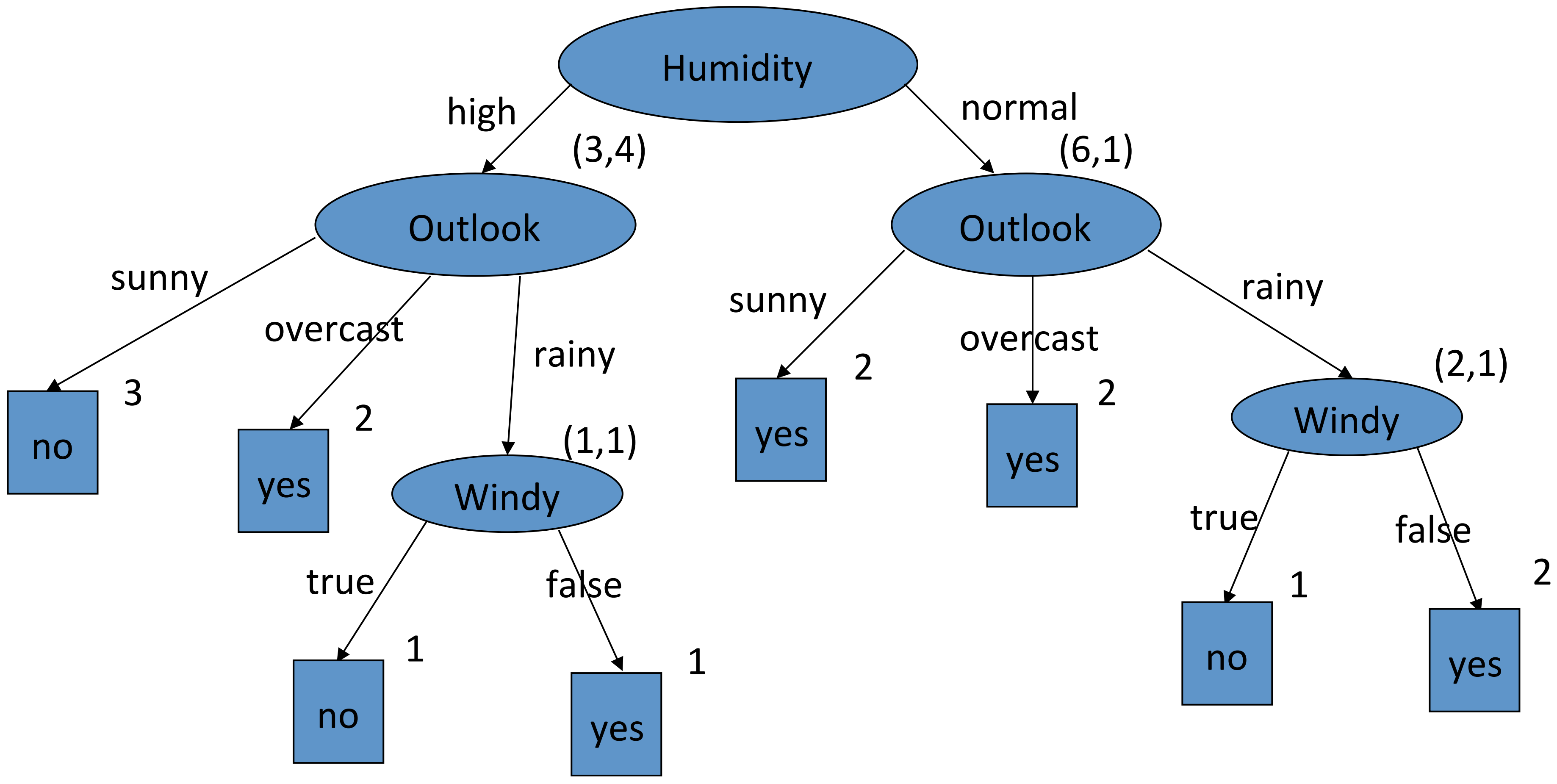
Several Techniques

- Nearest neighbour methods
- Naïve Bayes and Bayesian networks
- Discriminant analysis approach, e.g. SVM
- Decision tree based methods
  - Study in this lecture

Decision Tree Construction

- Recursive procedure
  - Select an attribute to place at the root node
  - Make one branch for each possible value of the selected attribute
  - For each branch repeat the above two steps recursively
    - Using only those instances that actually reach the branch
  - Stop developing a branch if it has instances belonging to the same class
- Several decision trees are possible
  - Based on the order of the selection of the attributes

Example Decision Tree 1



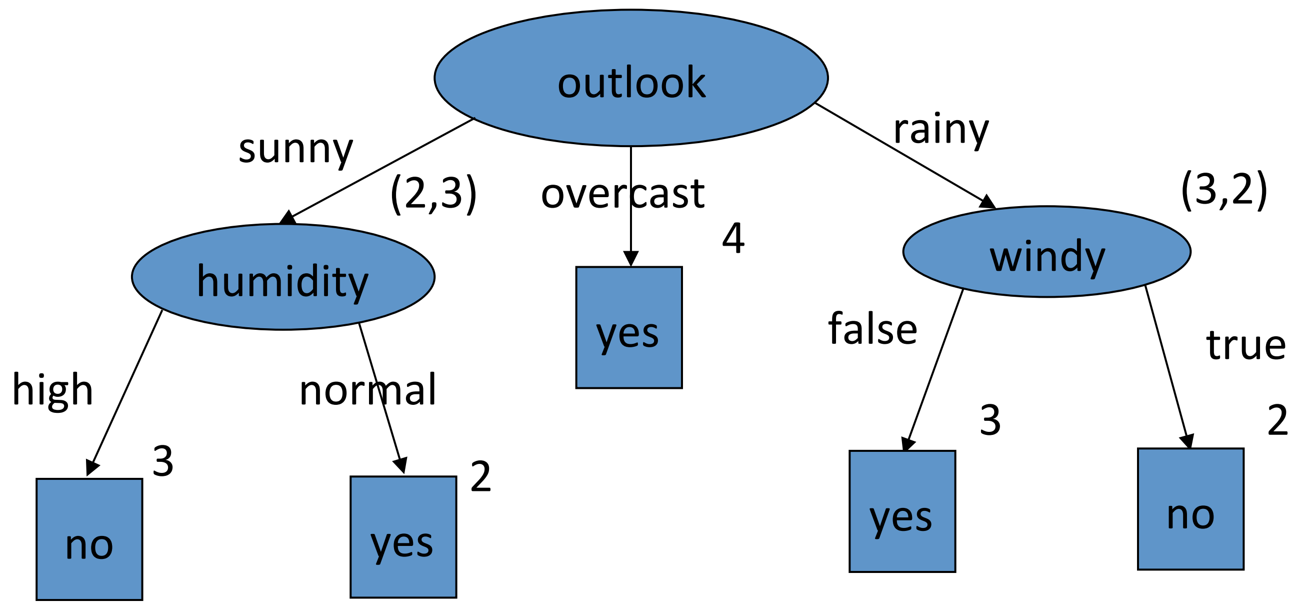
Observations:
Outlook and windy repeated in this tree
Windy tested only when outlook is rainy

Actions:

Start the tree with Outlook

Test windy when outlook is rainy

Example Decision Tree 2



Occam's Razor

- Principle stated by William of Ockham
  - "Other things being equal, simple theories are preferable to complex ones"
  - Informally,
  - "Keep it simple, stupid!!"
- This has been the guiding principle for developing scientific theories
- Applied to our two decision trees describing the weather data
  - Decision tree 2 is preferable to decision tree 1
- Small decision trees are better
  - Attribute ordering makes the difference

Attribute Selection for Splitting

- In our case, outlook is a better attribute at the root than others
  - Because when outlook is at the root, one of its branches (overcast) immediately leads to a ‘pure’ daughter node which terminates further tree growth
  - Pure nodes have the same class label for all the instances reaching that node
- We need a generic function that helps to rank order attributes
  - The function should reward attributes that produce ‘pure’ daughter nodes

Entropy

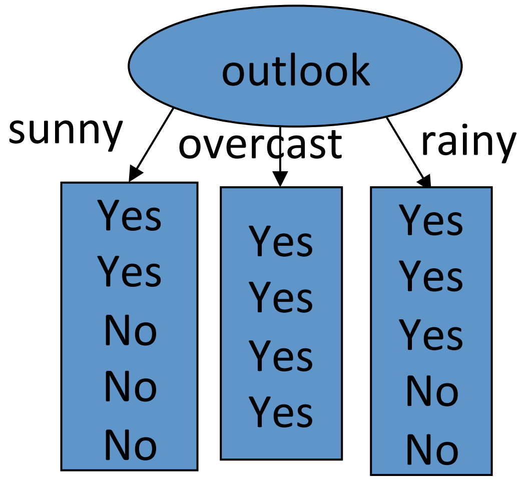
- Entropy is a measure of purity expressed in bits
- For a node in the tree, it represents the expected amount of information that would be needed to specify the class of a new instance that reached the node
- **Entropy(p_1,p_2,…,p_n) = -$p_1$logp_1 – $p_2$logp_2 … - $p_n$logp_n**
- Logarithms are computed for base 2 because we want the entropy measure in bits
- **p1, p2, … pn are fractions that sum up to 1**
- Because logarithms of fractions are negative, minus signs are used in the above formula to keep the entropy measure positive
- For the weather data
  - p1 = fraction of instances with play is true = 9/14
  - p2 = fraction of instances with play is false = 5/14
  - entropy(9/14,5/14) = -9/14log9/14 – 5/14log5/14

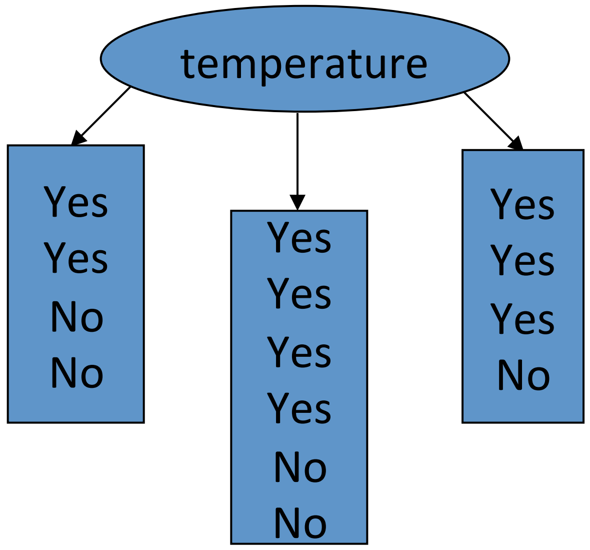
  = -9/14(log9 – log14) -5/14(log5 – log14)

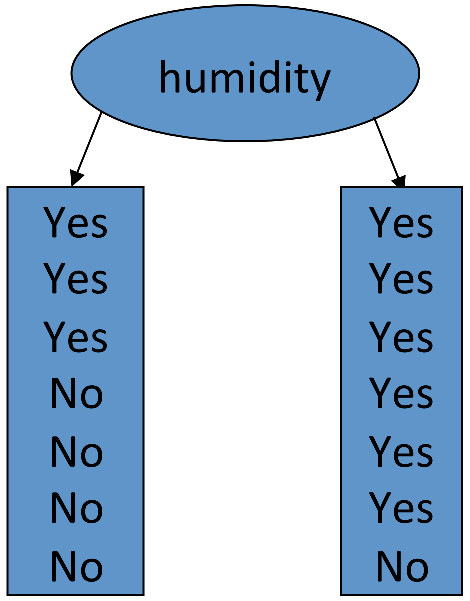
  = -9/14log9 + 9/14log14 – 5/14log5 +5/14log14

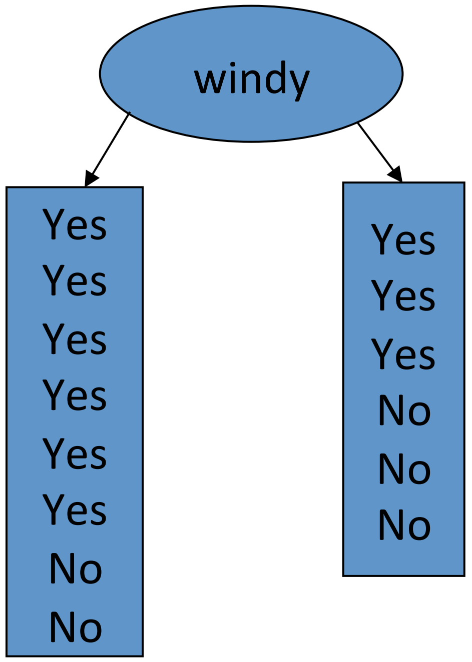
  =-9/14log9 -5/14log5 +14/14log14 = (-9log9 -5log5 +14log14)/14 = 0.940 bits (fractions of bits allowed!!)

Tree stumps for the weather data









Entropy for the outlook stump

- Count the numbers of yes and no classes at the leaf nodes
  - [2,3], [4,0] and [3,2]
- Compute the entropy for each branch of the stump
  - Entropy(2/5,3/5) = 0.971 bits
  - Entropy(4/4,0/4) = 0.0 bits
  - Entropy(3/5,2/5) = 0.971 bits
- Compute the entropy for the whole stump
  - Entropy([2,3],[4,0],[3,2]) = 5/14* Entropy(2/5,3/5) + 4/14* Entropy(4/4,0/4) + 5/14* Entropy(3/5,2/5) =5/14*0.971+4/14*0+5/14*0.971
    =0.693 bits
- Represents the information needed in bits to specify the class for a new instance using this stump

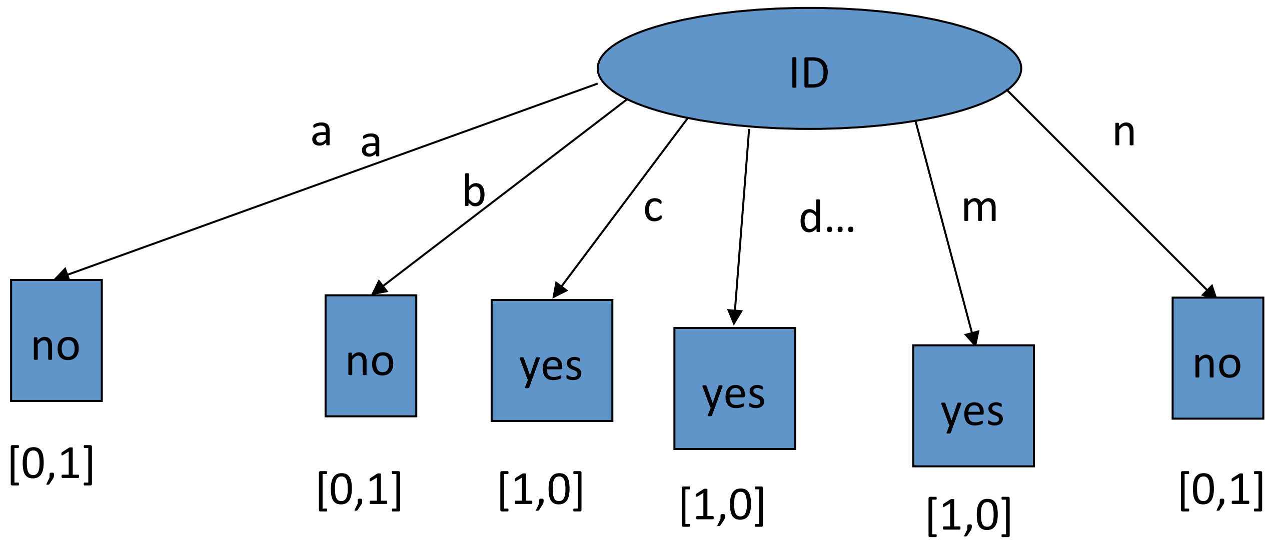
Information Gain

- When the attribute outlook was not considered, the weather data set has an entropy of 0.940 bits (as computed on slide 10)
- Entropy for the outlook stub is 0.693 bits (as computed on the previous slide)
- We made a saving of (0.940 – 0.693) bits on the information needed to specify the class of a new instance using this stump
  - This is the informational value of creating the outlook node
  - Gain(outlook) = (0.940 – 0.693) bits = 0.247 bits
- Similar computations for other stumps give us
  - Gain(temperature) = 0.029 bits
  - Gain(humidity) = 0.152 bits
  - Gain(windy)=0.048
- Because information gain is maximum for outlook, it should be selected for the root node
- Continuing with the above procedure builds the example decision tree 2
- ID3 algorithm uses information gain with the recursive procedure described earlier

Weather Data with ID

ID	Outlook	Temperature	Humidity	Windy	Play
a	sunny	hot	high	false	no
b	sunny	hot	high	true	no
c	overcast	hot	high	false	yes
d	rainy	mild	high	false	yes
e	rainy	cool	normal	false	yes
f	rainy	cool	normal	true	no
g	overcast	cool	normal	true	yes
h	sunny	mild	high	false	no
i	sunny	cool	normal	false	yes
j	rainy	mild	normal	false	yes
k	sunny	mild	normal	true	yes
l	overcast	mild	high	true	yes
m	overcast	hot	normal	false	yes
n	rainy	mild	high	true	no

Tree Stump for the ID attribute



Entropy([0,1],[0,1],[1,0],[1,0] …..[1,0],[0,1]) =
1/14*Entropy(0/1,1/1)+1/14*Entropy(0/1,1/1) +
1/14*Entropy(1/1,0/1)+ 1/14*Entropy(1/1,0/1)+ …+
1/14*Entropy(1/1,0/1)+1/14*Entropy(0/1,1/1) =
1/14*0 + 1/14*0 + 1/14*0 …1/14*0 = 0

Gain(ID) = 0.940 – 0 = 9.940 bits

Highly branching attributes

- Weather data with ID has different id values for each instance
  - Knowing the id value is enough to predict the class
  - Therefore entropy for the stump with this attribute would be zero
    - Gain is high (0.940) and therefore this attribute will be selected first for tree construction
  - The tree constructed would be useless
    - Cannot predict new instances
    - Tells nothing about structure of the decision
- The above situation arises <u>whenever an attribute leads to high branching</u>
- A split always results in entropy reduction
  - Because a split reduces the number of classes
- Information gain does not account for this <u>intrinsic information of a split</u>

Gain Ratio

- Gain ratio is a measure used to adjust for the intrinsic information of a split
- Intrinsic information of a split is computed using the number and size of the daughter nodes produced by the split
  - Without taking the class information into account
- Intrinsic Information for the ID stump
  - Entropy([1,1,1,…,1])=14*(-1/14log1/14)=3.807 bits
- Gain Ratio = Information Gain/Intrinsic Information
- Gain ratio for the ID stump = 0.940/3.807=0.247
- Gain ratios for other stumps are
  - GainRatio(outlook) = 0.157, GainRatio(temperature)=0.019
  - GainRatio(humidity) = 0.152, GainRatio(windy) = 0.049
- In this case, ID still wins but not by a huge margin
- Additional measures used to guard against such useless attributes

Gain Ratio 2

- Gain ratio might overcompensate for intrinsic information
  - Because humidity (0.152) splits the data only into two branches, its GainRatio is nearly equal to that of outlook (0.157)
- Possible to fix this by choosing an attribute
  - with high gain ratio and
  - has InfoGain equal to the average of the InfoGains for all the attributes
- C4.5 (j4.8 in Weka) uses GainRatio and is an improvement over ID3

Summary so far

- Attribute selection for splitting achieved using measures of ‘purity’
  - Information Gain
  - Gain Ratio
  - Etc.
- Issues related to decision tree construction
  - Numerical Attributes
  - Missing values
  - Overfitting and Pruning