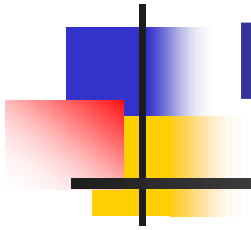
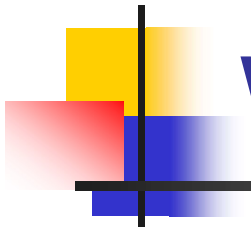


ECS509U - Probability & Matrices

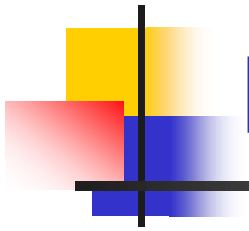
Tassos Tombros

Week 5



Week 5: Learning Objectives

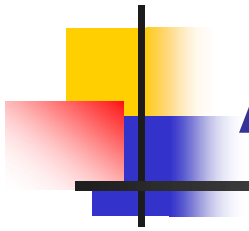
- At the end of Week 5 you should be able to appreciate computer science applications that make use of probability theory, and more specifically to:
  - understand the main principles of Bayesian classification, and to apply the same principles to different classification problems
  - grasp the main concepts behind Bayesian Networks



Remember Bayes

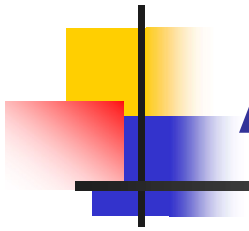
- The theorem allows us to calculate P(A|B) given P(B|A)
- P(A) is our prior belief that event A will occur
- Bayes Theorem tells us that:

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$



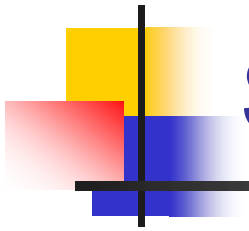
Applications of Bayes Theorem

- Bayes Theorem has found numerous applications in many fields, including Computer Science
  - Bayesian Classifiers
    - spam filtering, network traffic classification, object classification, etc.
  - Bayesian Networks
  - Bayesian Machine Learning: Bayesian Inference / Bayesian Decision Theory



Automatic classification

- What is **classification** (or else **categorisation**)?
  - We are given a number of existing classes (or categories), with some pre-classified examples
  - We are then presented with a new, unclassified, item, and are asked to make a decision about which class to fit the object into
- Applications: Numerous
  - Spam email filtering (classes: spam / not spam)
  - Web page classification, e.g. Yahoo-style directory pages (classes: Sports:Football, News:Europe:UK, etc.)
  - 'clever' feeds of on-line content (classes: relevant to the client / not relevant to the client)
  - automatic classification of recognised objects (e.g. face recognition, number plate recognition, etc.)
  - network traffic categorisation (e.g. Web, FTP, peer-to-peer, keyboard loggers, etc.)



So, how can it be done?

- Many ways of doing automatic classification, but we will talk about a **Machine Learning** (ML) approach based on **Bayesian probabilities**
- The ML approach is based on **learning the properties** (or features) of items that are **already assigned to classes** - this is called **training**
- When a new, unclassified item comes, we match its features against the learnt features of classes, and estimate the probability that it should be assigned to the classes. This is called **testing**
- **Features** (or properties): observable quantities (characteristics) that we can capture, measure and count
  - for example, the words in email messages, their subject line, the address of the sender, the time and date sent, etc.

Bayesian Spam Filtering

- In a very simple form, we would need to calculate the probabilities of assigning the class ‘spam’ (or ‘not-spam’) to an email given the set of features of the email, **P(spam|features)**
- By applying **Bayes**, we get:

$$P(spam \mid features) = \frac{P(features \mid spam)P(spam)}{P(features)}$$

- The conditional probabilities of observing the features given that the email is spam, **P(features|spam), we can estimate from the ‘training data’**
  - emails that we know whether they are spam or not - we can count all kinds of statistics for these emails (e.g. by counting how many times words occur in spam and not-spam emails, etc.)
- **P(spam) is the prior probability that an email will be spam**, we can estimate this based, again, on our training data

One step further: Naïve Bayes

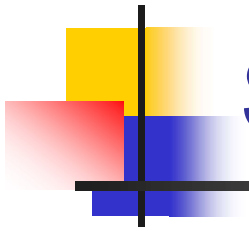
- In reality we will be using many features for our classification: $feature_1$, $feature_2$, …, $feature_i$, …
- We can estimate **P(features|spam)** if we assume that all features occur **independently** of one another given the class (e.g. given that they are spam)
  - this assumption is naïve, hence the name **Naïve Bayes**
  - The assumption allows us to calculate the probabilities as:

$$P(feature|spam) = \prod_i P(feature_i|spam) \text{ and } P(\text{features}) = \prod_i P(feature_i)$$

where P($feature_i$|spam) is the probability of observing the i-th feature amongst spam emails in the training data

and P($feature_i$) is the probability of observing $feature_i$ in the training data

- Why are we allowed to do this calculation? Because if A is **conditionally independent** of B given C, then **P(A,B|C)=P(A|C)P(B|C)**



Some specific features

- Many particular features, apart from individual words, can be used in the spam filtering application:
  - Particular phrases, e.g. “Free money”, “lottery prize”, “only £xx” etc.
  - Over-emphasised punctuation (e.g. “!!!!!”, “?????”)
  - Domain of the message sender (e.g. `.ac.uk` or `.com` or `.edu` or `.co.uk`, etc.)
  - Was message sent to a mailing list or to an individual user?
  - What time of the day was the message sent? (most spam emails are sent at night)
  - Does it contain an attachment? (most spam emails do not contain attachments)
  - etc.

So, how is the decision taken?

- We can calculate both the following probabilities:

$$P(spam \mid features) = \frac{P(features \mid spam)P(spam)}{P(features)}$$

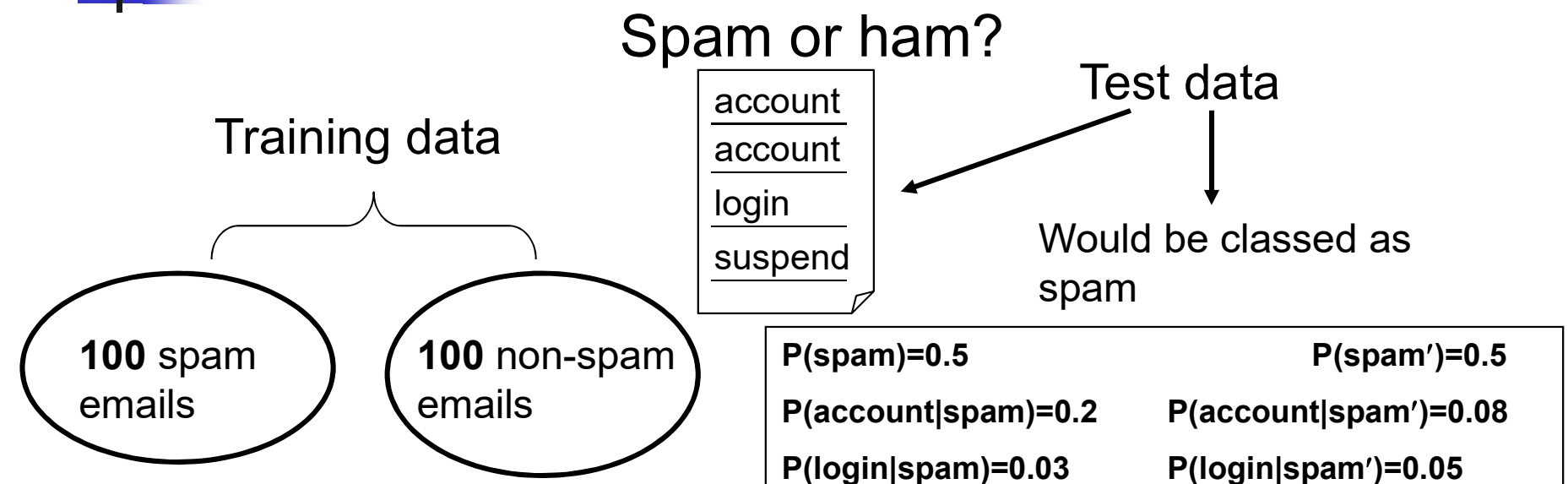
$$P(spam' \mid features) = \frac{P(features \mid spam')P(spam')}{P(features)}$$

and **we assign the email to the class with the highest probability**

so if P(spam|*features*)>P(spam′|*features*) the email will be classed as spam, and vice-versa

- Notice that **P(*features*) appears in both denominators**, so it will not affect our decision, it is the same number in both cases
  - in many implementations this is left out to save computations

Let's see an example



account: 200 times

bank: 200 times

login: 30 times

suspend: 10 times

Total words in spam set: 1,000

account: 80 times

bank: 21 times

login: 50 times

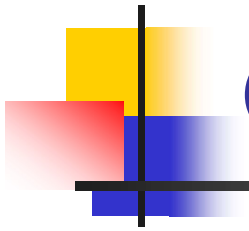
suspend: 18 times

Total words in non-spam set: 1,000

P(spam)=0.5	**P(spam′)=0.5**
P(account|spam)=0.2	**P(account|spam′)=0.08**
P(login|spam)=0.03	**P(login|spam′)=0.05**
P(suspend|spam)=0.01	**P(suspend|spam′)=0.018**

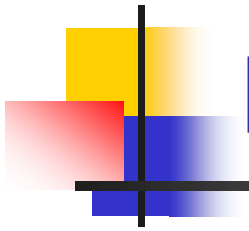
P(spam|account, account, login, suspend) = P(spam)P(account|spam)^{2}P(login|spam) P(suspend|spam)**=0.000006**

P(spam′|account, account, login, suspend) = P(spam′)P(account|spam′)^{2}P(login|spam′) P(suspend|spam′)**=0.00000288**



Google uses probabilities too…

- You all know **Google**, but have you heard of **PageRank**?
  - the value of 'importance' that Google assigns to each page on the Web
- But what is PageRank?
  - **PageRank is a probability distribution** used to represent the likelihood that a person randomly clicking on links (**random surfer**) will arrive at any particular page
  - The probability that the random surfer clicks on one link is given by the number of links on that page
  - Links by "popular" pages count more for Google
  - PageRank also involves huge operations with **matrices**
- The Information Retrieval group at EECS works on such problems (retrieval, classification, etc.)



Recommendation Systems also

- Or else **collaborative filtering** systems
- The best known example of such systems would **Amazon**
- Essentially these systems:
  - Collect data for their users (e.g. what items they browse, what items they buy, what ratings they give to items, etc.)
  - They then make recommendations to users for items that the system thinks they will 'like'
  - Very frequently, this is done by looking at what other users 'like you' have liked, bought, rated… chances are you will like the same things too

An example - Amazon

Recommended for Anastasios Tombros (If you're not Anastasios Tombros, click here.)

Narrow by Event

Page You Made

Narrow by Category

Books
DIY & Tools
DVD
Electronics & Photo
Garden & Outdoors
Kitchen & Home
Music
PC & Video Games
Software
Toys & Games
Video

Improve Your Recommendations

Update your Amazon history to improve your recommendations

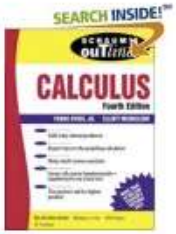
Items you own
Rated items
Not Interested

Need Help?

Visit our help area to learn more.

Recommendations for you are based on items you own and more.

view: **All** | New Releases | Coming Soon

1. 

Schaum's Outline of Calculus (Schaum's Outline)
by Frank Ayres (Jul 1, 1999)
In stock

RRP: ~~£9.99~~
Price: £5.99
46 used & new from £3.50

Add to Basket | Add to Wish List

I own it | Not interested | Rate it

Recommended because you purchased **Schaum's Outline of Probability, 2nd Edition** and more (Fix this)

2. 

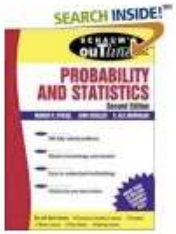
Schaum's Outline of Probability, Random Variables, and Random Processes (Schaum's Outline)
by Hwei Hsu (Nov 1, 1996)
Average Customer Review: (2)
In stock

RRP: ~~£9.99~~
Price: £5.99
28 used & new from £4.12

Add to Basket | Add to Wish List

I own it | Not interested | Rate it

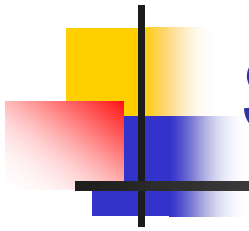
Recommended because you purchased **Schaum's Outline of Probability, 2nd Edition** and more (Fix this)

3. 

Schaum's Outline of Probability and Statistics (Schaum's Outline)
by Murray R Spiegel (April 1, 2000)
Average Customer Review: (2)
In stock

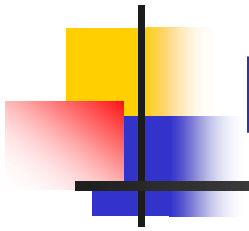
RRP: ~~£9.99~~
Price: £5.99
40 used & new from £4.00

Add to Basket | Add to Wish List



Some probabilistic flavours

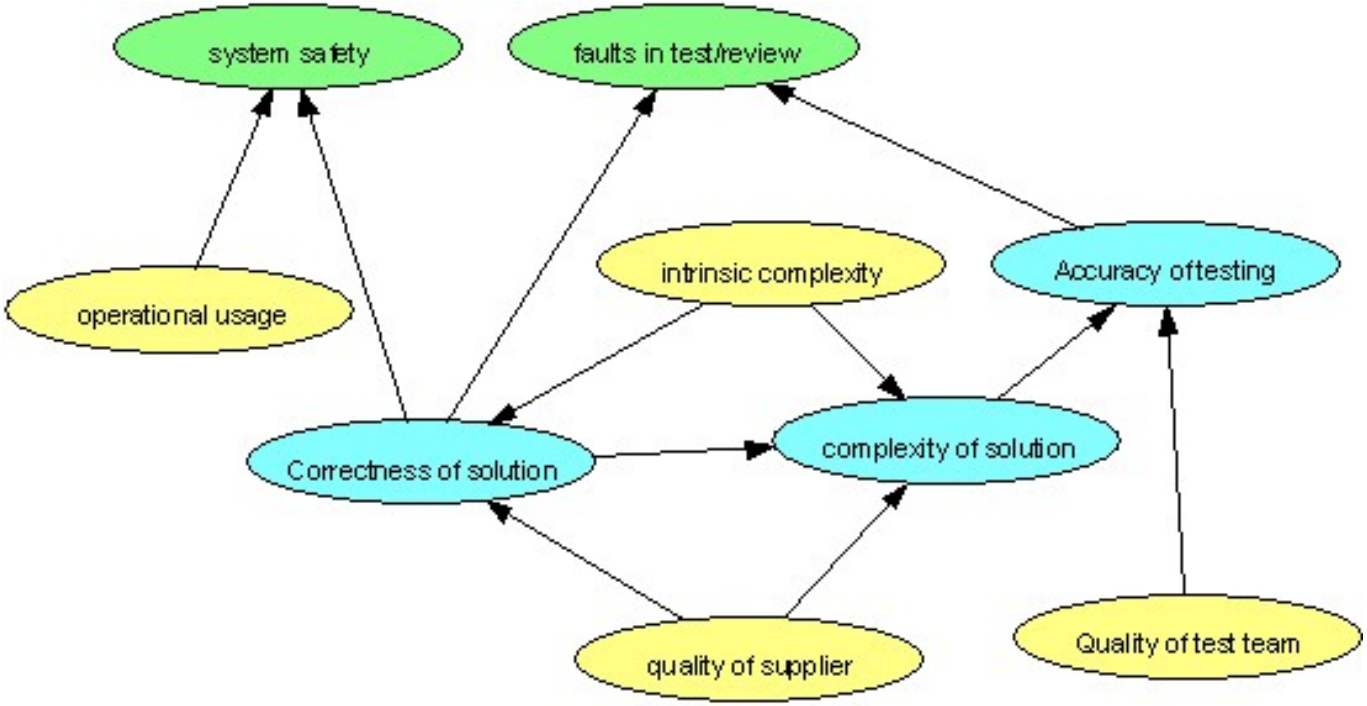
- The task in collaborative filtering is to **predict the votes** of a particular user from a database of user votes from a population of other users
- From a probabilistic point of view, the collaborative filtering task can be viewed as
  - **calculating the expected value of a vote** (for a yet unseen item), given what we know about the user
  - or, as the task of finding the **probability that a group of users should be placed in the same 'class'** based on how they have rated/viewed/bought items. Then the vote of a user can be approximated by the vote of members of his own 'class'



Bayesian Networks

- Also called **Bayesian Belief Networks (BBN)**, or **Probabilistic Inference Networks**
- A (BBN) is a directed graph, together with an associated set of probability tables
- The graph consists of nodes and arcs
  - The nodes represent variables, which can be discrete or continuous
  - The arcs represent causal/influential relationships between variables
- **Let's see these through an example**

A BBN example



"*number of faults in test/review*" is influenced by the "*correctness of the solution*" and the "*accuracy of testing*" - we model this relationship by appropriate arcs as shown

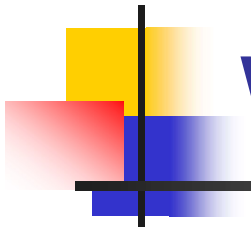
Taken from Prof. N. Fenton's notes on BBNs: http://www.eecs.qmul.ac.uk/~norman/BBNs//BBNs.htm

Nodes and probability tables

- Every node in a BBN has an associated Node Probability Table (NPT)
  - The NPTs capture the conditional probabilities of a node given the state of its parent nodes
  - If node A has parent nodes B and C then the NPT for node A expresses the probability p(A|B,C)

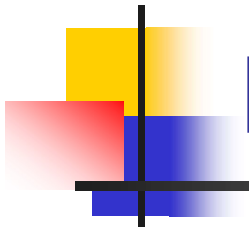
Example NPT for "*faults found in inspection/testing*" given the "*accuracy of testing*" and the "*correctness*" of the solution

Accuracy of testing	Low			Medium			High		
Correctness of solution	Low	Medium	High	Low	Medium	High	Low	Medium	High
Low	.33	.33	.3	.2	.33	.5	.1	.25	.7
Medium	.33	.33	.3	.3	.33	.3	.2	.5	.2
High	.33	.33	.4	.5	.33	.2	.7	.25	.1



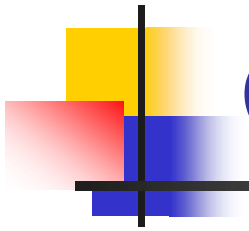
Why use BBNs?

- BBNs are a way of describing complex probabilistic reasoning
  - BBN represents the structure of the problem space in an intuitive, graphical format
- The main use of BBNs is in situations that require statistical inference
  - the user knows some evidence, e.g. events that have happened, and wishes to infer the probabilities of other events, which have not yet been observed
  - they allow us to model the uncertainty of the outcome of certain events
  - we can essentially play out different scenarios and study the outcomes



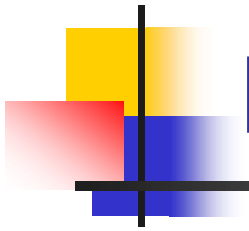
Propagation in BBNs

- Using probability theory and Bayes theorem it is possible to update the values of all the other probabilities in the BBN: **propagation**
- Bayesian analysis can be used for both **'forward'** and **'backward'** inference
  - **'Forward'** example: "*faults found in inspection/testing*" given the "*accuracy of testing*" and the "*correctness of the solution*"
  - **'Backward'** example: given a certain number for "*faults found in inspection/testing*", what are the likely values for "*accuracy of testing*" and the "*correctness of the solution*"?



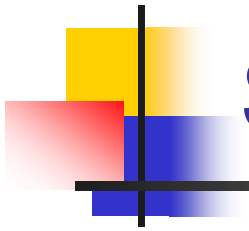
Computational challenges

- The Bayesian propagation computations, even for an example as small as the one in these slides, are very complex and cannot be calculated manually
- Given the very large number of nodes and arcs in real-life BNNs, it is challenging to find methods for the efficient calculation of ‘updated’ probabilities
  - essentially, to find efficient methods for propagating probability values along the nodes of a BBN
- A research group at EECS has developed a powerful tool (AgenaRisk) for performing fast propagations in BBNs with hundreds of nodes
  - more details next year and maybe in Software Engineering!!!



More applications of probability

- Numerous more applications
- In this lecture we focused on examples that also relate to work that happens at EECS
- Many more probabilistic applications
  - Artificial Intelligence
  - Computer Network traffic modeling
  - Theoretical computer science: Probabilistic algorithms
  - Modeling and processing of multimedia (music, speech, images, video, etc.)
  - Computer Vision



Summary of lecture

- In Week 5 we covered:
  - Applications of probability theory to computer science
    - Classification and spam filtering, esp. Naïve Bayes
    - Web search and recommendation systems
    - Bayesian networks