Data Mining
Lecture 7: Document Filtering
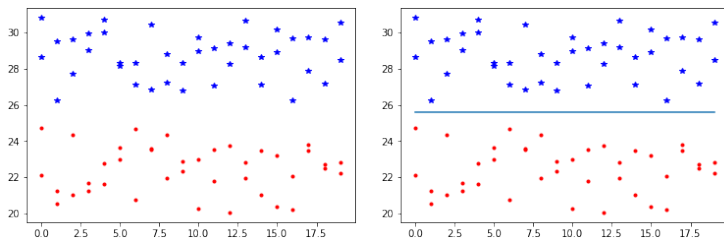
Jo Houghton

ECS Southampton

March 4, 2019

Document Filtering - Introduction

Document filtering can be accomplished by a range of approaches.

► Supervised Machine Learning - Classification
► Naïve Bayes
► Fisher's Method
► Feature Engineering

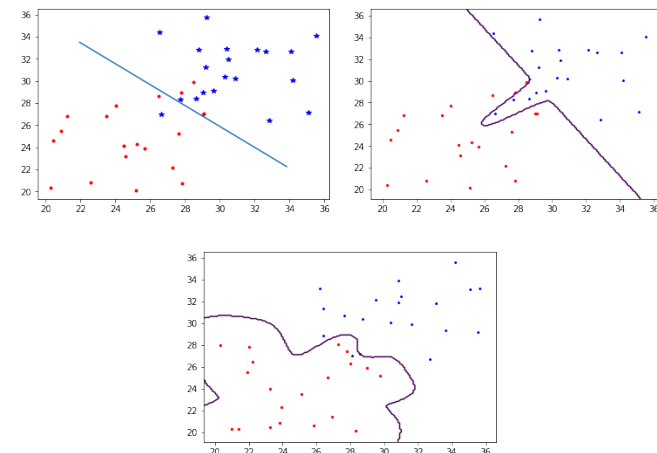## Document Filtering - Classification

Linear classifiers:

Separate different groups using a boundary, usually a hyperplane through featurespace, minimising error.

Different algorithms will chose the boundary dependent on different constraints.

This used a simple distance to mean classifier, that the Bayesian Decision Boundary simplifies to when the data is isotropic

---

Document Filtering - Classification

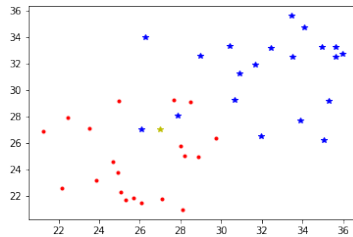Non linear binary classifiers can work when the data is not linearly separable

but you do need to watch out for overfitting

Document Filtering - Classification

Multiclass classifiers: KNN
K-Nearest Neighbours: Assigns class based on majority class of closest K neighbours in featurespace

- ▶ K = 1? blue star
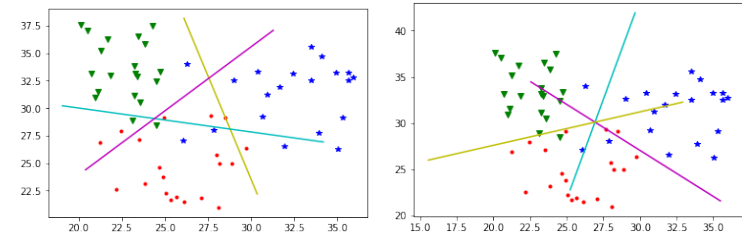- ▶ K = 3? blue star
- ▶ K = 5? red dot
- ▶ K > 5? red dot

Document Filtering - Classification

Linear classifiers are binary classifiers.
How do you do multiclass classification using linear classifiers?

- ▶ One vs All - one classifier per class
- ▶ One vs One - K(K-1)/2 classifiers

One vs. All , One vs. One

Document Filtering - Spam Filters

Problem statement: Spam Filter
Need to classify if email is *spam* or *ham*
Early attempts used rules:
i.e. it is spam if it has:

- ▶ Too many capitals
- ▶ The word viagra or penis
- ▶ Garish colors in embedded html
- ▶ Email only has an embedded picture

Problems..?

- ▶ Some people like to use uppercase
- ▶ Spammers learn rules and get round them
- ▶ No personalisation
- ▶ What if the user wants to buy viagra?

Solution: use Machine Learning to build personalised spam filters

Document Filtering - Naïve Bayes

Start with *bag of words*

- ▶ simple tokenisation: split on non-letters
- ▶ simple pre-processing: convert all to lower case
- ▶ count only presence or absence of term

Make a table counting the number of times a term has appeared for each category

	money	viagra	tea	you	dear	...	Total
Spam	15	8	0	19	15	...	30
Ham	2	0	15	20	12	...	70

Count up the total number of documents

Document Filtering - Naïve Bayes

Now work out the conditional probability, i.e. the probability of a feature given a category:

If n_c is the number of documents in a category, and n_{fc} is the number of documents with feature f in category c..

$$p(f|c) = \frac{n_{fc}}{n_c}$$

	money	viagra	tea	you	dear	...	Total
Spam	15	8	0	19	15	...	30
Ham	2	0	15	20	12	...	70

Using the data above:
calculate $p(\text{``viagra''}|Spam) = 8/30 \approx 0.27$
calculate $p(\text{``tea''}|Ham) = 15/70 \approx 0.21..$ but..
calculate $p(\text{``tea''}|Spam) = 0/30 = 0$!.. does this make sense?

Document Filtering - Naïve Bayes

Smoothing probability estimates

Should "tea" *never* be expected to appear in Spam documents?

Need a better way to estimate the conditional probability that accounts for infrequently seen features (sample size too small)

We introduce an *assumed* probability

- ▶ where there is little evidence
- ▶ can be based on some evidence

Produce a weighted estimate for the conditional probability based on the assumed and the raw computed probability

---

Document Filtering - Naïve Bayes

Using Bayesian Statistics, assuming that random variables are drawn from probability distributions rather than point samples

In this case, Spam/Ham classification for a feature f is binomial with a *beta* distributed **prior**

$$p_w(f|c) = \frac{weight \times assumed + count \times p_{raw}(f|c)}{count + weight}$$

where *count* is the number of times feature f occurs across all categories
Note: You can use Naïve Bayes with a Bayesian approach or with a frequentist (MLE) approach.

Document Filtering - Naïve Bayes

However the conditional probability of the whole document is required

To do this we assume that all features are independent of each other.

This is what makes it Naïve Bayes

In this case the *features* are words, and as certain words are very likely to appear together this assumption is false. However in practice, it doesn't matter.

It will still work even if incorrect!

Document Filtering - Naïve Bayes

The Naïve assumption allows the conditional probability to be expressed as the product of all the conditional feature probabilities

$$p(d|c) = \prod_{f \in d} p(f|c)$$

This is the **likelihood** of the document given a category.

When implementing, the product of a lot of very small numbers could lead to *floating point underflow*, so we take the logs and sum instead.

$$\log(p(c|d)) \propto \log(p(c)) + \sum_{f \in d} \log(p(f|c))$$

Document Filtering - Naïve Bayes

We use Bayes' Theorem to calculate the probability of a class (i.e. Spam or Ham) given a document, the **posterior**
Bayes' Theorem:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \quad \therefore p(c|d) = \frac{p(d|c) \times p(c)}{p(d)}$$

$p(c)$ is the probability that a document is in category c, the **prior** of c.
The **prior** can be calculated:

- ▶ Empirically, using the total No. docs in c divided by the total number
- ▶ or assuming all classes to be equally probably

$p(d)$ is constant, so therefore irrelevant, as same for all categories

Document Filtering - Naïve Bayes

posterior $\propto$ likelihood $\times$ prior

$$\therefore p(c|d) \propto p(d|c) \times p(c)$$

$p(c|d)$ can be calculated for all categories using this formula
The most likely category is thus assigned to the document, i.e. with the largest $p(c|d)$. This is **maximum a posteriori**, MAP, and assumes the categories are equal.

In the case of Spam/Ham this isn't true, the cost of misclassifying a good email as Spam is much higher than misclassifying a Spam email as Ham.

Document Filtering - Naïve Bayes

Could require a higher ratio to classify..

$$if \quad \frac{p(Spam|d)}{p(Ham|d)} > 3 \quad then \quad Spam$$

Document Filtering - Fisher's Method

Naïve Bayes: uses feature likelihoods to compute whole document probability

Fisher's Method

- ▶ calculates probability of each category for each feature $p(c|f)$
- ▶ tests to see if combined probabilities are more or less likely than a random

This assumes independence of features

Document Filtering - Fisher's Method

To calculate category probabilities $p(c|f)$ for given feature: Can use Bayes' Extended form, but need $P(c)$ for all c

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{\sum_j P(B|A_j)P(A_j)}$$

$P(c)$ can be estimated from the data, or from an unbiased estimate, all $P(c)$ equally likely.

Document Filtering - Fisher's Method

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{\sum_j P(B|A_j)P(A_j)}$$

For a Spam/Ham classifier, there is no *a priori* reason to assume one or the other.

$$P(c) = P(Ham) = P(Spam) = 0.5$$

$$\therefore P(c|f) = \frac{P(f|c)}{(P(f|Ham) + P(f|Spam))}$$

Document Filtering - Fisher's Method

Fisher's method combines k probabilities ("p-values") $P(c = C|f_k)$ from each test in to a single test statistic, X^2

$$X_{2k}^2 \sim -2 \sum \ln(p(c = C|f_i))$$

if the p-values are independent, this X^2 statistic will follow a *chi-squared* distribution in $2k$ degrees of freedom

So we can calculate a combined p-value

$$p_C = K^{-1}(-2 \sum_{i=1}^{k} \ln(p(c = C|f_i)), 2k) = K^{-1}(-2 \ln(\prod_{i=1}^{k} p(c = C|f_i)), 2k)$$

where K^{-1} is the inverse chi-squared function

Document Filtering - Fisher's Method

Making classifications

$$I = \frac{1 + p_{Spam} - p_{Ham}}{2}$$

I tends to 1 if document is Spam, and 0 if it is Ham.

Document Filtering - Improving Text Features

Feature Engineering:

In email, there are *Fields*

- ▶ sender; to; cc;
- ▶ title; subject; body

These can be used as separate features

Document Filtering - Improving Text Features

Spam emails may contain lots of capitals, so a new feature can be engineered to measure this

e.g. if more than 30% of words are uppercase, then record a virtual 'uppercase' feature..

	money	viagra	tea	you	dear	...	v. upper	Total
Spam	15	8	0	19	15	...	5	30
Ham	2	0	15	20	12	...	0	70

Document Filtering - Improving Text Features

N-Grams

Bag of words loses the order and context the words are in, so instead of looking at ust individual workds, we can use pairs, or triplets (n=2, 3) of words as a base feature.
Generally these are called $n - grams$

"The quick brown fox jumped over the lazy dog"
becomes:

'The quick" "quick brown" "brown fox" "fox jumped" "jumped over" "over the" "the lazy" "lazy dog"

Document Filtering - Improving Text Features

N-Grams
Advantages?
- ▶ captures context
- ▶ gets names
- ▶ more sense of the sentence

Disadvantages?
- ▶ feature explosion!

This makes orders of magnitude more features.

Document Filtering - Improving Text Features

Can we do better?
Natural Language Processing
- ▶ POS - Part of Speech tagging
  `"The quick brown fox jumped over the lazy dog"`

  `[('The', 'DT'), ('quick', 'JJ'), ('brown', 'NN'),`
  `('fox', 'NN'), ('jumped', 'VBD'), ('over', 'IN'),`
  `('the', 'DT'), ('lazy', 'JJ'), ('dog', 'NN')]`

  " The quick brown fox jumped over the lazy dog "
  ipynb demo

Document Filtering - Improving Text Features

- ▶ NE - Named entity extraction
  `"but unfortunately for MSC Bellissima, the`
  `British weather intervened."`
    but/CC
    unfortunately/RB
    for/IN
    (ORGANIZATION MSC/NNP Bellissima/NNP)
    the/DT
    (GPE British/JJ)
    weather/NN
    intervened/VBD

Document Filtering - Improving Text Features

Using these tools we can get $\approx 91\%$ accuracy for POS tagging

State of the art is close to 97% accurate.

Natural Language Processing is **hard**!

More accurate, robust NLP is shallow
i.e. superficial analysis, not really understanding sentence

Document Filtering - Summary

Learning models to categorise data is a core part of data mining

- Many supervised machine learning techniques can be used
- For Spam/Ham, probabilistic approaches work well as:
  - easy to implement
  - interpretable
  - computationally efficient
  - online
  - .. but assuming independence of features (naïve) can be problematic

Choice of features is key