Introduction to
Information Retrieval

Hinrich Schütze and Christina Lioma
Chapter 13 in IIR: Text Classification & Naive Bayes
(Slides modified by Scott Sanner)

Recap

- **Previous: Information Retrieval**
  - How to store for efficient lookup
  - The vector space model for document scoring
    - Nearest neighbor retrieval with an exemplar document
    - With new queries daily, can be hard to use parametric classifier
  - How to evaluate performance
  - Optimizations: relevance feedback + tolerant retrieval
- **Now: Text Classification**
  - Have fixed classification task and available labeled data
  - Parametric classifiers effective and efficient for this task

Outline

1. Recap
2. Text classification
3. Naive Bayes
4. Evaluation
5. Linear Classifiers

A text classification task: Email spam filtering

```
From: ‘‘’’ <takworlld@hotmail.com>
Subject: real estate is the only way... gem oalvgkay
Anyone can buy real estate with no money down
Stop paying rent TODAY !
There is no need to spend hundreds or even thousands for
similar courses
I am 22 years old and I have already purchased 6 properties
using the
methods outlined in this truly INCREDIBLE ebook.
Change your life NOW !
=================================================
Click Below to order:
http://www.wholesaledaily.com/sales/nmd.htm
=================================================
```

How would you write a program that would automatically detect and delete this type of message?

Formal definition of TC: Training

Given:

- A document space X
  - Documents are represented in this space, e.g. vectors
- A fixed set of classes C = $\{c_1, c_2, \ldots, c_J\}$
  - The classes are human-defined for the needs of an application (e.g., relevant vs. nonrelevant).
- A training set D of labeled documents with each labeled document $<d, c> \in X \times C$

Using a learning method or learning algorithm, we then wish to learn a classifier ϒ that maps documents to classes:

$$ϒ : X \rightarrow C$$

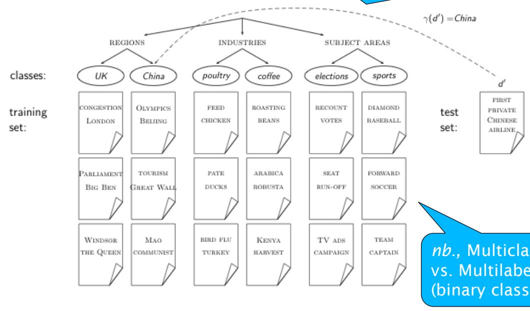
Formal definition of TC: Application/Testing

Given: a description $d \in X$ of a document (often previously unseen)

Determine: ϒ $(d) \in C$, that is, the class that is most appropriate for d

Topic classification

20 newsgroups (see ML lab)



nb., Multiclass vs. Multilabel (binary classes)

Examples of how search engines use classification

- Language identification (classes: English vs. French etc.)
- The automatic detection of spam pages (spam vs. nonspam)
- The automatic detection of sexually explicit content (sexually explicit vs. not)
- Topic-specific or *vertical* search – restrict search to a "vertical" like "related to health" (relevant to vertical vs. not)
- Standing queries (e.g., Google Alerts)
- Sentiment detection: is a movie or product review positive or negative (positive vs. negative)

Classification methods: 1. Manual

- Manual classification was used by Yahoo in the beginning of the web. Also: ODP, PubMed
- Very accurate if job is done by experts
- Consistent when the problem size and team is small
- Scaling manual classification is difficult and expensive.
- → We need automatic methods for classification.

Classification methods: 2. Rule-based

- Google Alerts uses rule-based classification.
- There are IDE-type development environments for writing very complex rules efficiently. (e.g., Verity)
- Often: Boolean combinations (as in Google Alerts)
- Accuracy is very high if a rule has been carefully refined over time by a subject expert.
- Building and maintaining rule-based classification systems is cumbersome and expensive.
  - But immediately apply to new data

A Verity topic (a complex classification rule)

```
comment line               # Beginning of art topic definition
top-level topic            art ACCRUE
topic definition modifiers     /author = "fsmith"
                               /date   = "30-Dec-01"
                               /annotation = "Topic created
                                             by fsmith"
subtopic topic             * 0.70 performing-arts ACCRUE
  evidence topic           ** 0.50 WORD
  topic definition modifier    /wordtext = ballet
  evidence topic           ** 0.50 STEM
  topic definition modifier    /wordtext = dance
  evidence topic           ** 0.50 WORD
  topic definition modifier    /wordtext = opera
  evidence topic           ** 0.30 WORD
  topic definition modifier    /wordtext = symphony
subtopic                   * 0.70 visual-arts ACCRUE
                           ** 0.50 WORD
                               /wordtext = painting
                           ** 0.50 WORD
                               /wordtext = sculpture

subtopic                   * 0.70 film ACCRUE
                           ** 0.50 STEM
                               /wordtext = film
subtopic                   ** 0.50 motion-picture PHRASE
                           *** 1.00 WORD
                               /wordtext = motion
                           *** 1.00 WORD
                               /wordtext = picture
                           ** 0.50 STEM
                               /wordtext = movie
subtopic                   * 0.50 video ACCRUE
                           ** 0.50 STEM
                               /wordtext = video
                           ** 0.50 STEM
                               /wordtext = vcr
                           # End of art topic
```

But can we learn these feature weights?

Outline

1. Recap
2. Text classification
3. Naive Bayes
4. Evaluation
5. Linear Classifiers

The Naive Bayes classifier

- The Naive Bayes classifier is a probabilistic classifier.
- We compute the probability of a document d being in a class c as follows: $P(c|d) \propto P(c) \prod_{1 \le k \le n_d} P(t_k|c)$
  - n_d is the length of the document. (number of tokens)
  - $P(t_k|c)$ is the conditional probability of term t_k occurring in a document of class c
  - $P(t_k|c)$ as a measure of how much evidence t_k contributes that c is the correct class.
  - $P(c)$ is the prior probability of c.

Maximum a posteriori class

- Goal in Naive Bayes classification is to find the "best" class.
- The best class is the most likely or maximum a posteriori (MAP) class c_{map}:

$$c_{\text{map}} = \arg\max_{c \in \mathbb{C}} \hat{P}(c|d) = \arg\max_{c \in \mathbb{C}} \hat{P}(c) \prod_{1 \le k \le n_d} \hat{P}(t_k|c)$$

Taking the log

- Multiplying lots of small probabilities can result in floating point underflow.
- Since log(xy) = log(x) + log(y), we can sum log probabilities instead of multiplying probabilities.
- Since log is a monotonic function, the class with the highest score does not change.
- So what we usually compute in practice is:

$$c_{\text{map}} = \arg\max_{c \in \mathbb{C}} [\log \hat{P}(c) + \sum_{1 \le k \le n_d} \log \hat{P}(t_k|c)]$$

Naive Bayes classifier

- Classification rule:

$$c_{\text{map}} = \arg\max_{c \in \mathbb{C}} [\log \hat{P}(c) + \sum_{1 \le k \le n_d} \log \hat{P}(t_k|c)]$$

- Simple interpretation:
  - Each conditional parameter log $\hat{P}(t_k|c)$ is a weight that indicates how good an indicator t_k is for c.
  - The prior log $\hat{P}(c)$ is a weight that indicates the relative frequency of c.
  - We select the class with the most evidence (weight).

Parameter estimation take 1: Maximum likelihood

- Estimate parameters $\hat{P}(c)$ and $\hat{P}(t_k|c)$ from train data: How?
- Prior:

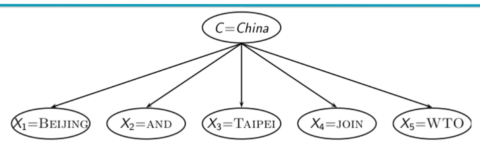
$$\hat{P}(c) = \frac{N_c}{N}$$

- N_c : number of docs in class c; N: total number of docs
- Conditional probabilities:

$$\hat{P}(t|c) = \frac{T_{ct}}{\sum_{t' \in V} T_{ct'}}$$

- T_{ct} is the number of tokens of t in training documents from class c (includes multiple occurrences)

The problem with maximum likelihood estimates: Zeros



$P(China|d) \propto P(China) \cdot P(\text{BEIJING}|China) \cdot P(\text{AND}|China) \cdot P(\text{TAIPEI}|China) \cdot P(\text{JOIN}|China) \cdot P(\text{WTO}|China)$

- If WTO never occurs in class China in the train set:

$$\hat{P}(\text{WTO}|China) = \frac{T_{China,\text{WTO}}}{\sum_{t' \in V} T_{China,t'}} = \frac{0}{\sum_{t' \in V} T_{China,t'}} = 0$$

To avoid zeros: Add-one smoothing (prior)

- Before:

$$\hat{P}(t|c) = \frac{T_{ct}}{\sum_{t' \in V} T_{ct'}}$$

- Now: Add one to each count to avoid zeros:

$$\hat{P}(t|c) = \frac{T_{ct}+1}{\sum_{t' \in V}(T_{ct'}+1)} = \frac{T_{ct}+1}{(\sum_{t' \in V} T_{ct'})+B}$$

- B is the number of different words (in this case the size of the vocabulary: $|V| = M$)

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 (offline)
- Classify test document (online)

Example: Parameter estimates

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

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

The denominators are (8 + 6) and (3 + 6) because the lengths of $text_c$ and $text_{\overline{c}}$ are 8 and 3, respectively, and because the constant B is 6 as the vocabulary consists of six terms.

Example: 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.

Linear Time complexity of Naive Bayes

mode	time complexity
training	$\Theta(\lvert\mathbb{D}\rvert L_{ave} + \lvert\mathbb{C}\rvert\lvert V\rvert)$
testing	$\Theta(L_a + \lvert\mathbb{C}\rvert M_a) = \Theta(\lvert\mathbb{C}\rvert M_a)$

- L_{ave}: average length of a training doc, L_a: length of the test doc, M_a: number of distinct terms in the test doc, $\mathbb{D}$: training set, V: vocabulary, $\mathbb{C}$: set of classes
- $\Theta(|\mathbb{D}|L_{ave})$ is the time it takes to compute all counts.
- $\Theta(|\mathbb{C}||V|)$ is the time it takes to compute the parameters from the counts.
- Generally: $|\mathbb{C}||V| < |\mathbb{D}|L_{ave}$
- Test time is also linear (in the length of the test document).
- Thus: Naive Bayes is linear in the size of the training set (training) and the test document (testing). This is optimal.

Why does Naive Bayes work?

- Naive Bayes can work well even though conditional independence assumptions are badly violated.
- Example:

	c_1	c_2	class selected
true probability $P(c\|d)$	0.6	0.4	c_1
$\hat{P}(c)\prod_{1\le k\le n_d}\hat{P}(t_k\|c)$	0.00099	0.00001	
NB estimate $\hat{P}(c\|d)$	0.99	0.01	c_1

- Double counting of evidence causes underestimation (0.01) and overestimation (0.99).
- Classification is about predicting the correct class and not about accurately estimating probabilities.
  - Correct estimation ⇒ accurate prediction.
  - But not vice versa!

Naive Bayes is not so naive

- A good dependable baseline for text classification (but not the best)
- Optimal if independence assumptions hold (never true for text, but true for some domains)
- Very fast, low storage requirements
- *More data often more important than better classifiers*

Outline

Evaluating classification

- Evaluation must be done on **test data** that are **independent of the training data**
  - Split your data into train and test sets!
- It's easy to get good performance on a test set that was available to the learner during training
  - e.g., just memorize the test set
- Measures:
  - Accuracy: not useful when class imbalance
  - Precision, recall, F_1
  - When can we use ranking metrics?

Averaging: Multiclass & Micro vs. Macro

- We now have an evaluation measure (F_1) for one class.
- Can report independently, or aggregate performance over all classes in the collection…
- Macroaveraging
  - Compute F_1 for each of the C classes
  - Average these C numbers
- Microaveraging
  - Compute TP, FP, FN for each of the C classes
  - Sum these C numbers (e.g., all TP to get aggregate TP)
  - Compute F_1 for aggregate TP, FP, FN

Naive Bayes vs. other methods

(a)

	NB	Rocchio	kNN		SVM
micro-avg-L (90 classes)	80	85	86		89
macro-avg (90 classes)	47	59	60		60

(b)

	NB	Rocchio	kNN	trees	SVM
earn	96	93	97	98	98
acq	88	65	92	90	94
money-fx	57	47	78	66	75
grain	79	68	82	85	95
crude	80	70	86	85	89
trade	64	65	77	73	76
interest	65	63	74	67	78
ship	85	49	79	74	86
wheat	70	69	77	93	92
corn	65	48	78	92	90
micro-avg (top 10)	82	65	82	88	92
micro-avg-D (118 classes)	75	62	n/a	n/a	87

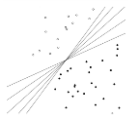
Evaluation measure: F_1 Naive Bayes does pretty well, but some methods beat it consistently (e.g., SVM).

Outline

Linear classifiers

- Definition:
  - A linear classifier computes a linear combination or weighted sum $\sum_i w_i x_i$ of the feature values.
  - Classification decision: $\sum_i w_i x_i > \theta$?
  - . . .where θ (the threshold) is a parameter.

  (First, we only consider *binary* classifiers.)



- Geometrically, this corresponds to a hyperplane separator.
  - We find this separator based on training set.
- Assumption: Classes are (mostly) linearly separable

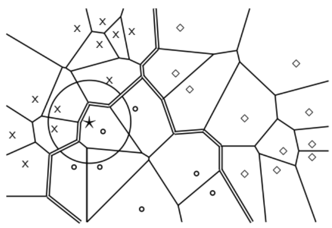
Naive Bayes as a linear classifier

Naive Bayes is a linear classifier (in log space) defined by:

$$\sum_{i=1}^{M} w_i \, f_i \; > \theta$$

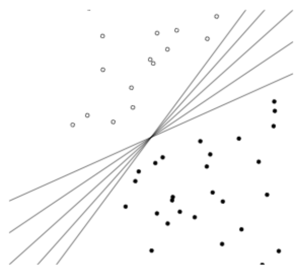
where $w_i = \log[\hat{P}(t_i|c)/\hat{P}(t_i|\bar{c})]$, f_i = is whether term t_i is present in *d*, and $\theta = -\log[\hat{P}(c)/\hat{P}(\bar{c})]$.

kNN is not a linear classifier



- Classification decision based on majority of *k* nearest neighbors.
- The decision boundaries between classes are piecewise linear . . .
- Not generally linear

Which hyperplane?



Computational Considerations

- Computationally, there are **two types** of learning algorithms.

(i) Simple learning algorithms that estimate parameters of the classifier directly from training data in one linear pass.
  - Naive Bayes, Rocchio, kNN

(ii) Iterative (discriminative) algorithms that require **optimization**
  - Support vector machines
  - Logistic regression

- Best algorithms are iterative, but need fast training
  - Up to a few million data points/features, use *scikitlearn* (see lab)
  - Beyond that (or if data streaming), need online training…