Einführung in die Computerlinguistik
Text Classification and Naive Bayes

Alexander Fraser and Robert Zangenfeind

Center for Information and Language Processing

2020-01-13

Dieses Foliensatz wurde von Prof. Dr. Hinrich Schütze erstellt.

Fehler und Mängel sind ausschließlich meine Verantwortung.

Outline

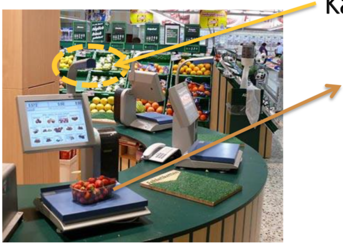
Outline

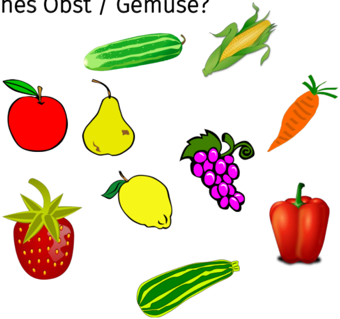
1. Text classification
2. Naive Bayes
3. NB theory
4. Evaluation of TC

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
=================================================
```

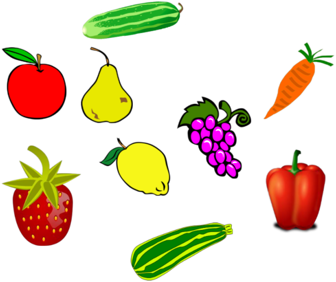
Mustererkennung (pattern recognition)





Was sind mögliche Erkennungsmerkmale?

Mustererkennung (pattern recognition)



Merkmale/Attribute:

- Farbe
- Größe
- Form
- …

Beispiele:
Attribute/Werte + richtige Klasse

↓

Algorithmus
(Maschinelles Lernen)

neues
Objekt → **Classifier** → *Zitrone (80%)*
Birne (20%)

Formal definition of TC: Training

Formal definition of TC: Training

Given:

- A document space $\mathbb{X}$

Formal definition of TC: Training

Given:

- A document space $\mathbb{X}$
  - Documents are represented in this space – typically some type of high-dimensional space.

Formal definition of TC: Training

Given:

- A document space $\mathbb{X}$
  - Documents are represented in this space – typically some type of high-dimensional space.
- A fixed set of classes $\mathbb{C} = \{c_1, c_2, \ldots, c_J\}$

Formal definition of TC: Training

Given:

- A document space $\mathbb{X}$
  - Documents are represented in this space – typically some type of high-dimensional space.
- A fixed set of classes $\mathbb{C} = \{c_1, c_2, \ldots, c_J\}$
  - The classes are human-defined for the needs of an application (e.g., spam vs. nonspam).

Formal definition of TC: Training

Given:

- A document space $\mathbb{X}$
  - Documents are represented in this space – typically some type of high-dimensional space.
- A fixed set of classes $\mathbb{C} = \{c_1, c_2, \ldots, c_J\}$
  - The classes are human-defined for the needs of an application (e.g., spam vs. nonspam).
- A training set $\mathbb{D}$ of labeled documents. Each labeled document $\langle d, c \rangle \in \mathbb{X} \times \mathbb{C}$

Formal definition of TC: Training

Given:

- A document space $\mathbb{X}$
  - Documents are represented in this space – typically some type of high-dimensional space.
- A fixed set of classes $\mathbb{C} = \{c_1, c_2, \ldots, c_J\}$
  - The classes are human-defined for the needs of an application (e.g., spam vs. nonspam).
- A training set $\mathbb{D}$ of labeled documents. Each labeled document $\langle d, c \rangle \in \mathbb{X} \times \mathbb{C}$

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

$$\gamma : \mathbb{X} \to \mathbb{C}$$

We can view sentences also as documents – so "document" refers to any piece of text we want to classify.

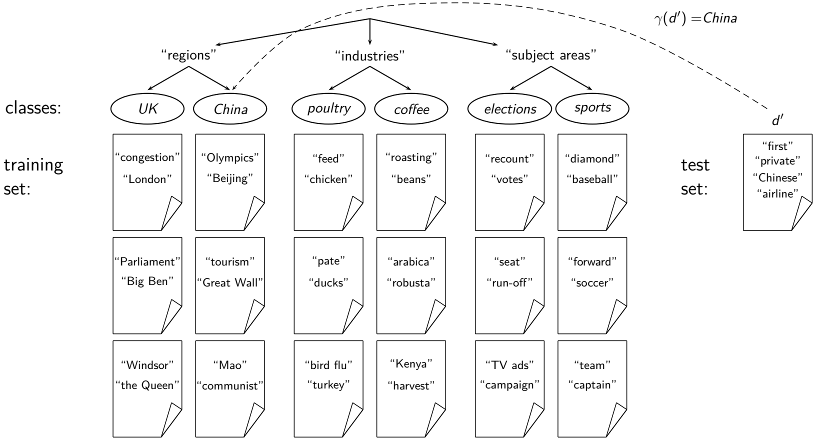
Formal definition of TC: Application/Testing

Formal definition of TC: Application/Testing

Given: a description $d \in \mathbb{X}$ of a document

Determine: $\gamma(d) \in \mathbb{C}$, that is,
determine the class that is most appropriate for d

Topic classification



Applications of text classification

- Language identification
  (classes: English vs French vs ...)
- The automatic detection of spam pages
  (spam vs nonspam)
- Sentiment analysis:
  Is a movie or product review positive or negative
  (positive vs negative)
- Topic-specific or *vertical* search:
  Restrict search to a "vertical" like "related to health"
  (classes: relevant to vertical vs not)

Classification methods: 1. Manual

Classification methods: 1. Manual

- Manual classification was used by Yahoo in the beginning of the web. Also: ODP, PubMed

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

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

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.

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.
- $\rightarrow$ We need automatic methods for classification.

Classification methods: 2. Rule-based

Classification methods: 2. Rule-based

- E.g., Google Alerts is rule-based classification.

Classification methods: 2. Rule-based

- E.g., Google Alerts is rule-based classification.
  - Google Alerts allows the definition of Google queries which are tracked in both News and Web.

Classification methods: 2. Rule-based

- E.g., Google Alerts is rule-based classification.
  - Google Alerts allows the definition of Google queries which are tracked in both News and Web.
- There are IDE-type development enviroments for writing very complex rules efficiently. (e.g., Verity)

Classification methods: 2. Rule-based

- E.g., Google Alerts is rule-based classification.
  - Google Alerts allows the definition of Google queries which are tracked in both News and Web.
- There are IDE-type development enviroments for writing very complex rules efficiently. (e.g., Verity)
- Often: Boolean combinations (as in Google Alerts)

Classification methods: 2. Rule-based

- E.g., Google Alerts is rule-based classification.
  - Google Alerts allows the definition of Google queries which are tracked in both News and Web.
- There are IDE-type development enviroments 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.

Classification methods: 2. Rule-based

- E.g., Google Alerts is rule-based classification.
  - Google Alerts allows the definition of Google queries which are tracked in both News and Web.
- There are IDE-type development enviroments 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.

A Verity topic (a complex classification rule)

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"`
subtopictopic	`* 0.70 performing-arts ACCRUE`
evidencetopic	`** 0.50 WORD`
topicdefinition modifier	`/wordtext = ballet`
evidencetopic	`** 0.50 STEM`
topicdefinition modifier	`/wordtext = dance`
evidencetopic	`** 0.50 WORD`
topicdefinition modifier	`/wordtext = opera`
evidencetopic	`** 0.30 WORD`
topicdefinition 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`

Classification methods: 3. Statistical/Probabilistic

Classification methods: 3. Statistical/Probabilistic

- This was our definition of the classification problem: Text classification as a learning problem

Classification methods: 3. Statistical/Probabilistic

- This was our definition of the classification problem: Text classification as a learning problem
- (i) Supervised learning of a the classification function γ and (ii) application of γ to classifying new documents

Classification methods: 3. Statistical/Probabilistic

- This was our definition of the classification problem:
  Text classification as a learning problem
- (i) Supervised learning of a the classification function γ and
  (ii) application of γ to classifying new documents
- We will look at one method for doing this:
  Naive Bayes

Classification methods: 3. Statistical/Probabilistic

- This was our definition of the classification problem:
  Text classification as a learning problem
- (i) Supervised learning of a the classification function γ and
  (ii) application of γ to classifying new documents
- We will look at one method for doing this:
  Naive Bayes
- No free lunch: requires hand-classified training data

Classification methods: 3. Statistical/Probabilistic

- This was our definition of the classification problem:
  Text classification as a learning problem
- (i) Supervised learning of a the classification function γ and
  (ii) application of γ to classifying new documents
- We will look at one method for doing this:
  Naive Bayes
- No free lunch: requires hand-classified training data
- But this manual classification can be done by non-experts.

Outline

1. Text classification
2. Naive Bayes
3. NB theory
4. Evaluation of TC

The Naive Bayes classifier

- The Naive Bayes classifier is a probabilistic classifier.

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:

$$P(c|d) \propto P(c) \prod_{1 \leq k \leq n_d} P(t_k|c)$$

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:

$$P(c|d) \propto P(c) \prod_{1 \leq k \leq n_d} P(t_k|c)$$

- n_d is the length of the document (the number of tokens), k an index to the kth token t_k.

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:

$$P(c|d) \propto P(c) \prod_{1 \leq k \leq n_d} P(t_k|c)$$

- n_d is the length of the document (the number of tokens), k an index to the kth token t_k.
- $P(t_k|c)$ is the conditional probability = bedingte Wahrscheinlichkeit
  of term t_k occurring in a document of class c

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:

$$P(c|d) \propto P(c) \prod_{1 \leq k \leq n_d} P(t_k|c)$$

- n_d is the length of the document (the number of tokens), k an index to the kth token t_k.
- $P(t_k|c)$ is the conditional probability = bedingte Wahrscheinlichkeit
  of term t_k occurring in a document of class c
- $P(t_k|c)$ is a measure of how much evidence t_k contributes that c is the correct class.

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:

$$P(c|d) \propto P(c) \prod_{1 \leq k \leq n_d} P(t_k|c)$$

- n_d is the length of the document (the number of tokens), k an index to the kth token t_k.
- $P(t_k|c)$ is the conditional probability = bedingte Wahrscheinlichkeit
  of term t_k occurring in a document of class c
- $P(t_k|c)$ is a measure of how much evidence t_k contributes that c is the correct class.
- $P(c)$ is the prior probability of c.

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:

$$P(c|d) \propto P(c) \prod_{1 \leq k \leq n_d} P(t_k|c)$$

- n_d is the length of the document (the number of tokens), k an index to the kth token t_k.
- $P(t_k|c)$ is the conditional probability = bedingte Wahrscheinlichkeit

  of term t_k occurring in a document of class c
- $P(t_k|c)$ is a measure of how much evidence t_k contributes that c is the correct class.
- $P(c)$ is the prior probability of c.
- If a document's terms do not provide clear evidence for one class vs. another, we choose the c with highest $P(c)$.

Maximum a posteriori class

- Goal in Naive Bayes classification:
  Find the “best” class

Maximum a posteriori class

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

$$c_{\text{map}} = \text{argmax}_{c \in \mathbb{C}} \hat{P}(c|d) = \text{argmax}_{c \in \mathbb{C}} \ \hat{P}(c) \prod_{1 \leq k \leq n_d} \hat{P}(t_k|c)$$

Taking the log

Taking the log

- Multiplying lots of small probabilities can result in floating point underflow.

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.

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.

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:

$$c_{\text{map}} = \text{argmax}_{c \in \mathbb{C}} \, [\log \hat{P}(c) + \sum_{1 \leq k \leq n_d} \log \hat{P}(t_k|c)]$$

Naive Bayes classifier

Naive Bayes classifier

- Classification rule:

$$c_{\text{map}} = \text{argmax}_{c \in \mathbb{C}} \left[\log \hat{P}(c) + \sum_{1 \leq k \leq n_d} \log \hat{P}(t_k|c)\right]$$

Naive Bayes classifier

- Classification rule:

$$c_{\text{map}} = \text{argmax}_{c \in \mathbb{C}} \, [\log \hat{P}(c) + \sum_{1 \leq k \leq n_d} \log \hat{P}(t_k|c)]$$

- Simple interpretation:

Naive Bayes classifier

- Classification rule:

$$c_{\text{map}} = \operatorname{argmax}_{c \in \mathbb{C}} \left[\log \hat{P}(c) + \sum_{1 \leq k \leq n_d} \log \hat{P}(t_k|c)\right]$$

- 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.

Naive Bayes classifier

- Classification rule:

$$c_{\text{map}} = \text{argmax}_{c \in \mathbb{C}} \, [\log \hat{P}(c) + \sum_{1 \leq k \leq n_d} \log \hat{P}(t_k|c)]$$

- 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.

Naive Bayes classifier

- Classification rule:

$$c_{\text{map}} = \text{argmax}_{c \in \mathbb{C}} \left[\log \hat{P}(c) + \sum_{1 \leq k \leq n_d} \log \hat{P}(t_k|c)\right]$$

- 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.
  - The sum of log prior and term weights is then a measure of how much evidence there is for the document being in the class.

Naive Bayes classifier

- Classification rule:

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

- 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.
  - The sum of log prior and term weights is then a measure of how much evidence there is for the document being in the class.
  - We select the class with the most evidence.

Parameter estimation take 1: Maximum likelihood

Parameter estimation take 1: Maximum likelihood

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

Parameter estimation take 1: Maximum likelihood

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

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

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

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:

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

Parameter estimation take 1: Maximum likelihood

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

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

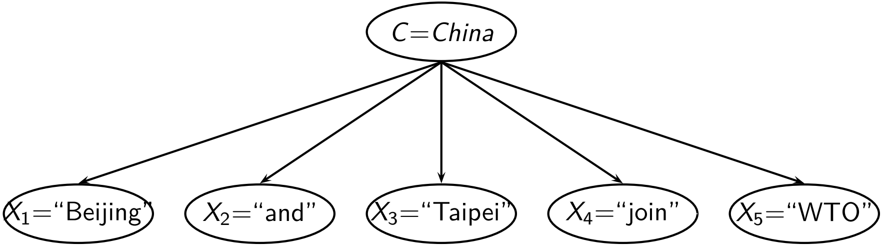
- N_c: number of docs in class c; N: total number of docs
- Conditional probabilities:

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

- 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

The problem with maximum likelihood estimates: Zeros

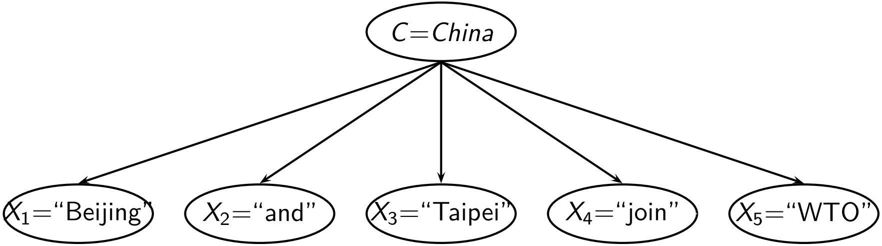


$$
\begin{aligned}
P(China|d) \;\propto\; & P(China) \cdot P(\text{“Beijing”}|China) \cdot P(\text{“and”}|China) \\
& \cdot P(\text{“Taipei”}|China) \cdot P(\text{“join”}|China) \cdot P(\text{“WTO”}|China)
\end{aligned}
$$

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

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

The problem with maximum likelihood estimates: Zeros



$$P(China|d) \propto P(China) \cdot P(\text{“Beijing”}|China) \cdot P(\text{“and”}|China) \cdot P(\text{“Taipei”}|China) \cdot P(\text{“join”}|China) \cdot \color{red}{P(\text{“WTO”}|China)}$$

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

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

The problem with maximum likelihood estimates: Zeros (cont)

- If there are no occurrences of "WTO" in documents in class China, we get a zero estimate:

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

The problem with maximum likelihood estimates: Zeros (cont)

- If there are no occurrences of "WTO" in documents in class China, we get a zero estimate:

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

- $\rightarrow$ We will get $P(China|d) = 0$ for any document that contains WTO!

To avoid zeros: Add-one smoothing

To avoid zeros: Add-one smoothing

- Before:

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

To avoid zeros: Add-one smoothing

- Before:

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

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

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

To avoid zeros: Add-one smoothing

- 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 bins – in this case the number of different words or the size of the vocabulary $|V| = M$

Naive Bayes: Summary

Naive Bayes: Summary

- Estimate parameters from the training corpus using add-one smoothing

Naive Bayes: Summary

- Estimate parameters from the training corpus using add-one smoothing
- For a new document, for each class, compute sum of (i) log of prior and (ii) logs of conditional probabilities of the terms

Naive Bayes: Summary

- Estimate parameters from the training corpus using add-one smoothing
- For a new document, for each class, compute sum of (i) log of prior and (ii) logs of conditional probabilities of the terms
- Assign the document to the class with the largest score

Naive Bayes: Training

Naive Bayes: Training

$\text{TrainMultinomialNB}(\mathbb{C}, \mathbb{D})$

$V \leftarrow \text{ExtractVocabulary}(\mathbb{D})$

$N \leftarrow \text{CountDocs}(\mathbb{D})$

for each $c \in \mathbb{C}$

do $N_c \leftarrow \text{CountDocsInClass}(\mathbb{D}, c)$

    $prior[c] \leftarrow N_c/N$

    $text_c \leftarrow \text{ConcatenateTextOfAllDocsInClass}(\mathbb{D}, c)$

    for each $t \in V$

    do $T_{ct} \leftarrow \text{CountTokensOfTerm}(text_c, t)$

    for each $t \in V$

    do $condprob[t][c] \leftarrow \frac{T_{ct}+1}{\sum_{t'}(T_{ct'}+1)}$

return $V, prior, condprob$

Naive Bayes: Testing

Naive Bayes: Testing

$\textsc{ApplyMultinomialNB}(\mathbb{C}, V, \mathit{prior}, \mathit{condprob}, d)$

$W \leftarrow \textsc{ExtractTokensFromDoc}(V, d)$

for each $c \in \mathbb{C}$

do $\mathit{score}[c] \leftarrow \log \mathit{prior}[c]$

    for each $t \in W$

    do $\mathit{score}[c] += \log \mathit{condprob}[t][c]$

return $\mathrm{argmax}_{c \in \mathbb{C}} \mathit{score}[c]$

Exercise: Estimate parameters, classify test set

	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	?

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

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

(B is the number of bins – in this case the number of different words or the size of the vocabulary $|V| = M$)

$$c_{\text{map}} = \text{argmax}_{c \in \mathbb{C}} \, [\hat{P}(c) \cdot \prod_{1 \leq k \leq n_d} \hat{P}(t_k|c)]$$

Example: Parameter estimates

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

Example: Classification

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

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".

UNK – unknown words

UNK

An UNK is a word that occurs in the test set,
but did not occur in the training set.

- Option 1: Simply ignore UNKs
- Option 2: Add UNK to the training vocabulary
  - All counts $T_{c\text{UNK}}$ are zero
    (since UNK does not occur in training set).
  - All words in the test set that did not occur in the training set are replaced by "UNK".

Outline

Naive Bayes: Analysis

Naive Bayes: Analysis

- Now we want to gain a better understanding of the properties of Naive Bayes.

Naive Bayes: Analysis

- Now we want to gain a better understanding of the properties of Naive Bayes.
- We will formally derive the classification rule ...

Naive Bayes: Analysis

- Now we want to gain a better understanding of the properties of Naive Bayes.
- We will formally derive the classification rule ...
- ...and make our assumptions explicit.

Derivation of Naive Bayes rule

Derivation of Naive Bayes rule

We want to find the class that is most likely given the document:

$$c_{\mathrm{map}} = \mathrm{argmax}_{c \in \mathbb{C}} \; P(c|d)$$

Apply Bayes rule $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$:

$$c_{\mathrm{map}} = \mathrm{argmax}_{c \in \mathbb{C}} \; \frac{P(d|c)P(c)}{P(d)}$$

Drop denominator since $P(d)$ is the same for all classes:

$$c_{\mathrm{map}} = \mathrm{argmax}_{c \in \mathbb{C}} \; P(d|c)P(c)$$

Too many parameters / sparseness

$$\begin{aligned} c_{\text{map}} &= \text{argmax}_{c \in \mathbb{C}} \; P(d|c)P(c) \\ &= \text{argmax}_{c \in \mathbb{C}} P(\langle t_1, \ldots, t_k, \ldots, t_{n_d} \rangle | c) P(c) \end{aligned}$$

Too many parameters / sparseness

$$
\begin{aligned}
c_{\text{map}} &= \text{argmax}_{c \in \mathbb{C}}\ P(d|c)P(c) \\
&= \text{argmax}_{c \in \mathbb{C}} P(\langle t_1, \ldots, t_k, \ldots, t_{n_d} \rangle | c) P(c)
\end{aligned}
$$

- There are too many parameters $P(\langle t_1, \ldots, t_k, \ldots, t_{n_d} \rangle | c)$, one for each unique combination of a class and a sequence of words.

Too many parameters / sparseness

$$
\begin{aligned}
c_{\text{map}} &= \text{argmax}_{c \in \mathbb{C}}\ P(d|c)P(c) \\
&= \text{argmax}_{c \in \mathbb{C}} P(\langle t_1, \ldots, t_k, \ldots, t_{n_d} \rangle | c) P(c)
\end{aligned}
$$

- There are too many parameters $P(\langle t_1, \ldots, t_k, \ldots, t_{n_d} \rangle | c)$, one for each unique combination of a class and a sequence of words.
- We would need a very, very large number of training examples to estimate that many parameters.

Too many parameters / sparseness

$$
\begin{aligned}
c_{\text{map}} &= \text{argmax}_{c \in \mathbb{C}}\ P(d|c)P(c) \\
&= \text{argmax}_{c \in \mathbb{C}} P(\langle t_1, \ldots, t_k, \ldots, t_{n_d} \rangle | c) P(c)
\end{aligned}
$$

- There are too many parameters $P(\langle t_1, \ldots, t_k, \ldots, t_{n_d} \rangle | c)$, one for each unique combination of a class and a sequence of words.
- We would need a very, very large number of training examples to estimate that many parameters.
- This is the problem of data sparseness.

Bag of words model

To reduce the number of parameters to a manageable size, we make the Naive Bayes conditional independence (bedingte Unabhängigkeit) assumption:

$$P(d|c) = P(\langle t_1, \ldots, t_{n_d} \rangle | c) = \prod_{1 \leq k \leq n_d} P(X_k = t_k | c)$$

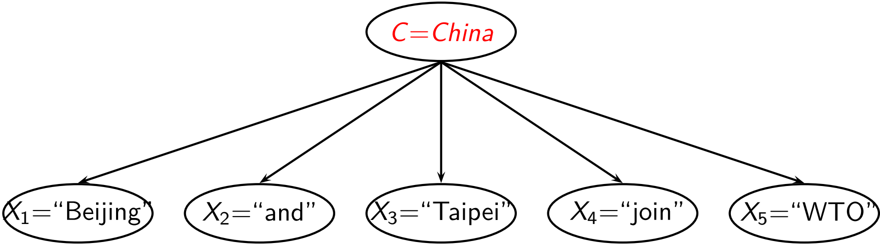
We assume that the probability of observing the conjunction of attributes is equal to the product of the individual probabilities $P(X_k = t_k|c)$.

Recall from earlier the estimates for these conditional probabilities: $\hat{P}(t|c) = \frac{T_{ct}+1}{(\sum_{t' \in V} T_{ct'})+B}$

This can be referred to as a bag of words model.

Generative model

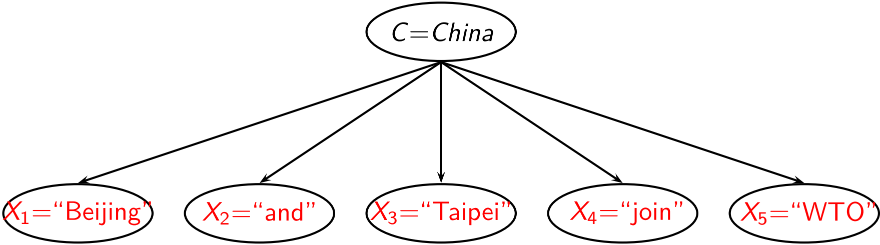
Generative model



$P(c|d) \propto P(c) \prod_{1 \leq k \leq n_d} P(t_k|c)$

- Generate a class with probability $P(c)$

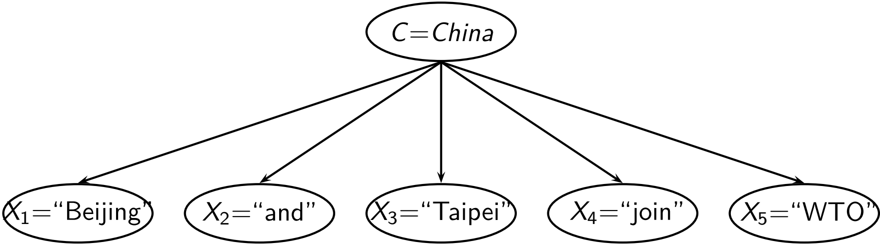
Generative model



$P(c|d) \propto P(c) \prod_{1 \leq k \leq n_d} P(t_k|c)$

- Generate a class with probability $P(c)$
- Generate each of the words (in their respective positions), conditional on the class, but independent of each other, with probability $P(t_k|c)$

Generative model



$P(c|d) \propto P(c) \prod_{1 \leq k \leq n_d} P(t_k|c)$

- Generate a class with probability $P(c)$
- Generate each of the words (in their respective positions), conditional on the class, but independent of each other, with probability $P(t_k|c)$
- To classify docs, we "reengineer" this process and find the class that is most likely to have generated the doc.

Naive Bayes is not so naive

Naive Bayes is not so naive

- Naive Bayes has won some bakeoffs (e.g., KDD-CUP 97)

Naive Bayes is not so naive

- Naive Bayes has won some bakeoffs (e.g., KDD-CUP 97)
- More robust to nonrelevant features than some more complex learning methods

Naive Bayes is not so naive

- Naive Bayes has won some bakeoffs (e.g., KDD-CUP 97)
- More robust to nonrelevant features than some more complex learning methods
- More robust to concept drift (changing of definition of class over time) than some more complex learning methods

Naive Bayes is not so naive

- Naive Bayes has won some bakeoffs (e.g., KDD-CUP 97)
- More robust to nonrelevant features than some more complex learning methods
- More robust to concept drift (changing of definition of class over time) than some more complex learning methods
- Better than methods like decision trees when we have many equally important features

Naive Bayes is not so naive

- Naive Bayes has won some bakeoffs (e.g., KDD-CUP 97)
- More robust to nonrelevant features than some more complex learning methods
- More robust to concept drift (changing of definition of class over time) than some more complex learning methods
- Better than methods like decision trees when we have many equally important features
- A good dependable baseline for text classification (but not the best)

Naive Bayes is not so naive

- Naive Bayes has won some bakeoffs (e.g., KDD-CUP 97)
- More robust to nonrelevant features than some more complex learning methods
- More robust to concept drift (changing of definition of class over time) than some more complex learning methods
- Better than methods like decision trees when we have many equally important features
- 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)

Naive Bayes is not so naive

- Naive Bayes has won some bakeoffs (e.g., KDD-CUP 97)
- More robust to nonrelevant features than some more complex learning methods
- More robust to concept drift (changing of definition of class over time) than some more complex learning methods
- Better than methods like decision trees when we have many equally important features
- 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

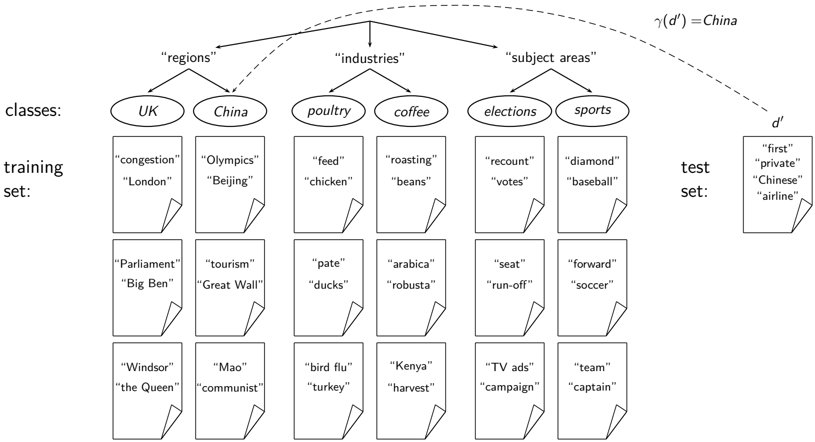
Naive Bayes is not so naive

- Naive Bayes has won some bakeoffs (e.g., KDD-CUP 97)
- More robust to nonrelevant features than some more complex learning methods
- More robust to concept drift (changing of definition of class over time) than some more complex learning methods
- Better than methods like decision trees when we have many equally important features
- 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

Outline

1. Text classification
2. Naive Bayes
3. NB theory
4. Evaluation of TC

Evaluation on Reuters



Example: The Reuters collection

symbol	statistic	value
N	documents	800,000
L	avg. # word tokens per document	200
M	word types	400,000

Example: The Reuters collection

symbol	statistic	value
N	documents	800,000
L	avg. # word tokens per document	200
M	word types	400,000

type of class	number	examples
region	366	UK, China
industry	870	poultry, coffee
subject area	126	elections, sports

A Reuters document

A Reuters document

REUTERS

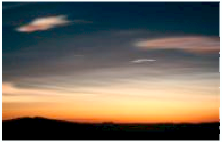
You are here: Home > News > Science > **Article**

Go to a Section: U.S. International Business Markets Politics Entertainment Technology Sports Oddly Enoug

Extreme conditions create rare Antarctic clouds

Tue Aug 1, 2006 3:20am ET

Email This Article | Print This Article | Reprints



[-] Text [+]

SYDNEY (Reuters) - Rare, mother-of-pearl colored clouds caused by extreme weather conditions above Antarctica are a possible indication of global warming, Australian scientists said on Tuesday.

Known as nacreous clouds, the spectacular formations showing delicate wisps of colors were photographed in the sky over an Australian

Evaluating classification

Evaluating classification

- Evaluation must be done on test data that are independent of the training data, i.e., training and test sets are disjoint.

Evaluating classification

- Evaluation must be done on test data that are independent of the training data, i.e., training and test sets are disjoint.
- 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).

Evaluating classification

- Evaluation must be done on test data that are independent of the training data, i.e., training and test sets are disjoint.
- 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: Precision, recall, F_1, classification accuracy

Precision P and recall R

	in the class	not in the class
predicted to be in the class	true positives (TP)	false positives (FP)
predicted to not be in the class	false negatives (FN)	true negatives (TN)

TP, FP, FN, TN are counts of documents. The sum of these four counts is the total number of documents.

$$\begin{aligned} \text{precision:} \quad P &= TP/(TP + FP) \\ \text{recall:} \quad R &= TP/(TP + FN) \end{aligned}$$

Precision/recall tradeoff

Precision/recall tradeoff

Precision/recall tradeoff

- You can easily increase recall by returning more results.

Precision/recall tradeoff

- You can easily increase recall by returning more results.
- Recall is a non-decreasing function of the number of results returned.

Precision/recall tradeoff

- You can easily increase recall by returning more results.
- Recall is a non-decreasing function of the number of results returned.
- A system that returns everything has 100% recall!

Precision/recall tradeoff

- You can easily increase recall by returning more results.
- Recall is a non-decreasing function of the number of results returned.
- A system that returns everything has 100% recall!
- The converse is also true (usually): It's easy to get high precision for very low recall.

Precision/recall tradeoff

- You can easily increase recall by returning more results.
- Recall is a non-decreasing function of the number of results returned.
- A system that returns everything has 100% recall!
- The converse is also true (usually): It's easy to get high precision for very low recall.
- In most application scenarios, we need both good precision and good recall.

Precision/recall tradeoff

- You can easily increase recall by returning more results.
- Recall is a non-decreasing function of the number of results returned.
- A system that returns everything has 100% recall!
- The converse is also true (usually): It's easy to get high precision for very low recall.
- In most application scenarios, we need both good precision and good recall.
- So we need to find a good precision-recall tradeoff.

A combined measure: F_1

- F_1 allows us to trade off precision against recall.

A combined measure: F_1

- F_1 allows us to trade off precision against recall.
- $$F_1 = \frac{1}{\frac{1}{2}\frac{1}{P} + \frac{1}{2}\frac{1}{R}} = \frac{2PR}{P+R}$$

A combined measure: F_1

- F_1 allows us to trade off precision against recall.
- $$F_1 = \frac{1}{\frac{1}{2}\frac{1}{P} + \frac{1}{2}\frac{1}{R}} = \frac{2PR}{P+R}$$
- This is the harmonic mean of P and R: $\frac{1}{F} = \frac{1}{2}(\frac{1}{P} + \frac{1}{R})$

A combined measure: F_1

- F_1 allows us to trade off precision against recall.
- $$F_1 = \frac{1}{\frac{1}{2}\frac{1}{P} + \frac{1}{2}\frac{1}{R}} = \frac{2PR}{P+R}$$
- This is the harmonic mean of P and R: $\frac{1}{F} = \frac{1}{2}(\frac{1}{P} + \frac{1}{R})$
- The harmonic mean is a kind of "soft" minimum.

Accuracy

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

F_1 scores for 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

F_1 scores for 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

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

Confusion matrix for Reuters-21578

true class:	assigned class: *money-fx*	*trade*	*interest*	*wheat*	*corn*	*grain*
money-fx	95	0	10	0	0	0
trade	1	1	90	0	1	0
interest	13	0	0	0	0	0
wheat	0	0	1	34	3	7
corn	1	0	2	13	26	5
grain	0	0	2	14	5	10

Example: 14 documents from *grain* were incorrectly assigned to *wheat*.

Exercise

Compute precision, recall and F_1:

	in class	not in class
predicted to be in class	TP: 18	FP: 2
predicted not to be in class	FN: 82	TN: 1,000,000,000

$$\begin{aligned} \text{precision:} \quad P &= TP/(TP+FP) \\ \text{recall:} \quad R &= TP/(TP+FN) \\ F_1 &= \frac{2PR}{P+R} \end{aligned}$$

Besonders klausurrelevant

- What is text classification?
  (or: What is sentence classification?)
- Naive Bayes classification rule
- Estimation of Naive Bayes priors and conditionals
- Theory: Bag of words model
  - Maximum likelihood
  - Add-one = Laplace
- Precision, recall, F_1
- Precision-recall tradeoff
- Confusion matrix