Search Engines

WS 2009 / 2010

Lecture 11, Thursday January 21st, 2010
(Text Classification with Naïve Bayes)

Prof. Dr. Hannah Bast
Chair of Algorithms and Data Structures
Department of Computer Science
University of Freiburg

UNI FREIBURG

Overview of Today‘s Lecture

- Learn how to do text classification
  - for example, for a given paper title, decide whether the paper is from a theory conference or from a search engine conference
  - we will learn the simplest of all methods: Naive Bayes
  - also some mathematical foundations
- But before
  - another nice demo of what a method like latent semantic indexing can achieve and how it works ...

Demo for LSI, PLSI, etc.

- Recall the intuition of the matrices U and V
  - columns of U are the "concepts"
  - columns of V are the mix of concepts per document

web	1	0	0	0.5	0
internet	1	1	0	0.5	0
surfing	1	1	1	1	0
beach	0	0	1	0.5	1

≈

1	0
1	0
1	1
0	1

1	1	0	0.5	0
0	0	1	0.5	1

Here is a nice tool showing this for real collections

Text Classification

- Consider the following paper titles

A nearly optimal oracle for avoiding failed vertices and edges	STOC
On iterative intelligent medical search	SIGIR
Guilt by association as a search principle	SIGIR
List decoding tensor products and interleaved codes	STOC
On dynamic range reporting in one dimension	STOC
Probabilistic Latent Semantic Indexing	SIGIR

- We want to tell from the titles alone
  - which one of these are STOC papers (the top theory conference)
  - and which ones are SIGIR papers (the top search conference)
  - Idea: use the invididual terms to predict whether STOC or SIGIR
    - e.g. "search" makes SIGIR more likely, "vertices" speaks for STOC

How to make a formal algorithm from this idea?

"Naive Bayes" Classification

- Three basic steps
  - STEP 1: decide on certain features and represent each record wrt to these features
    - we will take the words as features
    - other possible features → later slide
  - STEP 2: for each feature "learn" the likeliness / probability of that feature for each class
    - for example Pr(SIGIR | search) = 0.8
  - STEP 3: from these learned probabilities, compute the likeliness / probability of each class for a new record, e.g.
    - Pr(SIGIR | Document Expansion for Speech Retrieval) = 0.7
    - Pr(STOC | Document Expansion for Speech Retrieval) = 0.3

How do we get "Probabilitites" ?

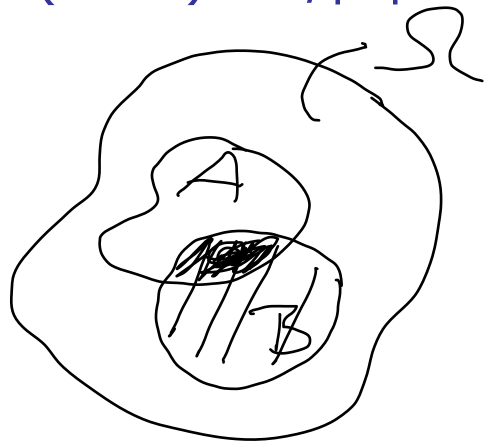
- We assume the following random process
  - for generating a single record / document with m words
  - pick class c with probability p_c, where $\Sigma_c\, p_c = 1$
  - pick the i-th word as w with probability p_{wc}, where $\Sigma_w\, p_{wc} = 1$
  - we make the following strong assumption
    - each word chosen independently of the other words
    - very unrealistic indeed why?
    - hence the "Naive" in Naive Bayes
- However unrealistic ...
  - now we have well-defined probabilities to compute with

Crash Course: Conditional Probabilities

- Bayes Theorem
  - let A and B be events in a probability space Ω
  - denote by Pr(A | B) the probability of A n B in the space B
  - then Pr(A | B) := Pr(A n B) / Pr (B)
  - and Pr(A | B) · Pr(B) = Pr (B | A) · Pr(A)
- For a good intuition, assume Ω is a finite set
  - from which we pick a random element X with Pr(X = x) = 1/|Ω|

$$Pr(A) = \frac{|A|}{|\Omega|}$$

$$Pr(A \mid B) = \frac{|A \cap B|}{|B|} = \frac{|A \cap B| \setminus |\Omega|}{|B| \setminus |\Omega|} = \frac{Pr(A \cap B)}{Pr(B)}$$



Naive Bayes — Now Formally

- For a new document D we want to compute
  - $\Pr(C = c \mid W_1 = w_1 \cap \ldots \cap W_m = w_m)$ for each class c

    where w_i is the i-th word of D
  - and then pick that class for which this probability is largest

    $\mathrm{argmax}_c \Pr(C = c \mid W_1 = w_1 \cap \ldots \cap W_m = w_m)$
  - by our independence assumptions + Bayes this is equal to

    $\mathrm{argmax}_c \Pr(C = c) \cdot \Pi_{i=1,\ldots,m} \Pr(W_i = w_i \mid C = c)$
  - proof on next slide ...

Proof that ...

argmax$_c$ Pr(C = c | W$_1$ = w$_1$ ∩ ... ∩ W$_m$ = w$_m$)

= argmax$_c$ Pr(C = c) · Π$_{i=1,...,m}$ Pr(W$_i$ = w$_i$ | C = c)

$$\Pr(C=c \mid W_1=w_1 \wedge \ldots \wedge W_m=w_m)$$

$$= \frac{\Pr(C=c)}{\underbrace{\Pr(W_1=w_1 \wedge \ldots \wedge W_m=w_m)}_{=:Q}} \cdot \underbrace{\Pr(W_1=w_1 \wedge \ldots \wedge W_m=w_m \mid C=c)}_{= \prod_{i=1}^{m} \Pr(W_i=w_i \mid C=c)}$$

Note: Q indep. of c

Note: here we need the independence assumption!

$$\xrightarrow[\text{the } Q]{\text{drop}} \Pr(C=c) \cdot \prod_{i=1}^{m} \Pr(W_i=w_i \mid C=c)$$

Learning our Priors from a Test Set

- We need the following prior probabilities
  - $\Pr(C = c)$ (the likeliness of each class)
  - $\Pr(W = w \mid C = c)$ (the likeliness of each word for each class)
  - we estimate these from a test set for which we already know the classes
- The following looks very natural
  - let T be our test set, and T_c the set of documents from class c
  - then $\Pr(C = c) := |Tc| / |T|$ note that $\Sigma_c |T_c| = T$
  - let n_{wc} = #occurrences of word w in documents from T_c
  - let n_c = #occurrences of all words in documents from T_c
  - then $\Pr(W = w \mid C = c) := n_{wc} / n_c$ note that $\Sigma_c n_{wc} = n_c$

Why is this a good choice for our priors?

Maximum Likelihood Estimation (MLE)

- Sequence of coin flips HHTTTTTTHTTTTTHTTHHT
  - say 5 times H and 15 times T
  - which Pr(H) and Pr(T) are the most likely?
  - looks like Prob(H) = ¼ and Pr(T) = ¾

Let S be the sequence we observed.
Let $h = \#H$ in S, and $t = \#T$
Let $p = Pr(H)$ and $q = Pr(T)$. Note: we don't know p and q

$$Pr(S) = p^h \cdot q^t \qquad \operatorname{argmax}_{p,q} \cancel{p^h \cdot q^t}\ L$$

$$L := \log Pr(S) = h \cdot \log p + t \cdot \log q \quad q = 1-p$$

$$\frac{\partial L}{\partial p} = \frac{h}{p} - \frac{t}{1-p} = 0 \Rightarrow (1-p) \cdot h = p \cdot t$$

$$h = p \cdot (h+t)$$

$$p = \frac{h}{h+t} \Rightarrow q = \frac{t}{h+t}$$

Maximum Likelihood Estimation (MLE)

- Sequence of coin flips HHTTTTTTHTTTTTHTTHHT
  - say 5 times H and 15 times T
  - which Pr(H) and Pr(T) are the most likely?
  - looks like Prob(H) = ¼ and Pr(T) = ¾

$$L = h \cdot \log p + t \cdot \log q \quad , \quad p + q = 1$$

$$\hat{L} = h \cdot \log p + t \cdot \log q + \lambda (1 - p - q)$$

$$\frac{\partial \hat{L}}{\partial \lambda} = 1 - p - q = 0$$

$$\frac{\partial \hat{L}}{\partial p} = \frac{h}{p} - \lambda = 0 \Rightarrow \lambda = \frac{h}{p} \Rightarrow p = \lambda \cdot h$$

$$\frac{\partial \hat{L}}{\partial q} = \frac{t}{q} - \lambda = 0 \Rightarrow \lambda = \frac{t}{q} \Rightarrow q = \lambda \cdot t$$

$$p + q = 1 \Rightarrow \lambda = \frac{1}{h+t}$$

$$\Rightarrow \frac{h}{p} = \frac{1}{h+t} \Rightarrow p = \frac{h}{h+t} \qquad q = \frac{t}{h+t}$$

Quality Evaluation

- How do we measure how good our classification is?
  - for each class c we do the following
  - let D_c = #documents from class c (ground truth)
  - let D'_c = #documents classified as c
  - then, as usual (note that these are per class)
    - precision $P := |D'_c \cap D_c| / |D'_c|$
    - recall $R := |D'_c \cap D_c| / |D_c|$
    - F-measure $F := 2 \cdot P \cdot R / (P + R)$
  - note that if $D_c = D'_c$ then $P = R = F = 100\%$ and only then

Feature Design and Selection

- Feature Design
  - in our example, we picked each word as feature
  - other example: pick all 3-grams
  - and / or additionally consider word positions
  - and / or additionally consider part of speech (POS) tags
- Feature Selection
  - just picking all words is easy
  - but some words are not very predictive, like new
  - considering them adds unnecessary noise to our decision
  - many methods to pick only predictive features
  - one of the simplest one: pick only frequent words

References

- LSI / PLSI demo
  - automatic Windows installer with tool + demo collections

    http://www.mpi-inf.mpg.de/~dfischer/alwis-1.1.0-full.exe
- Naïve Bayes
  - The Wikipedia article is quite good

    http://en.wikipedia.org/wiki/Naive_Bayes_classifier
  - The definitive book on the whole subject of learning

    Elements of Statistical Learning, Springer 2009