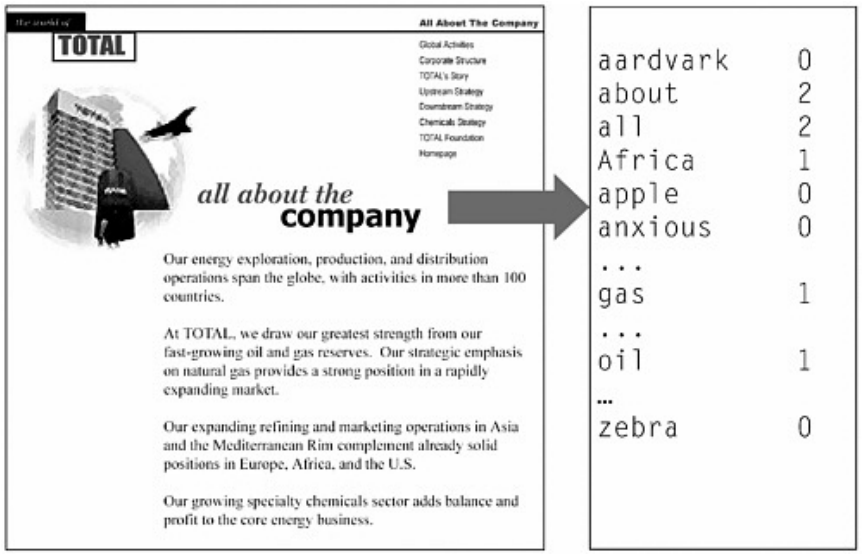
Bayesian Models
& Natural Language Processing (NLP)

Lyle Ungar

Naïve Bayes on a Bag of Words
LDA on a Bag of Words
HMMs/Deep learning on word sequences (later!)

NLP design decision 1: "Bag of Words" vs. Sequence

- Bag of words: The order of the words doesn't matter, just the count



NLP design decision 2: What is a word?

- **Tokenization**
  - "Yesterday, I didn't walk 3.14 miles to Penn Engineering."
  - *Yesterday , I didn't walk 3.14 miles to Penn Engineering .*
- **Multiword expressions/Named Entities**
  - Penn_Engineering
- **Increasingly "word part" or "byte pair" encoding**
  - Used with contextual embeddings

Naïve Bayes for Text Classification

adapted by Lyle Ungar from slides by Mitch Marcus, which were adapted from slides by Massimo Poesio, which were adapted from slides by Chris Manning :)

Example: Is this spam?

How do you know?

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
==========

Classification

- **Given**
  - A vector, *x* describing an instance
    - *Issue: how to represent text documents as vectors?*
  - A fixed set of categories: $C = \{c_1, c_2, \ldots, c_k\}$
- **Determine**
  - An optimal *classifier* $h(x)$

Examples of text categorization

- **Spam**
  - "spam" / "not spam"
- **Topics**
  - "finance" / "sports" / "asia"
- **Author**
  - "Shakespeare" / "Marlowe" / "Ben Jonson"
  - The Federalist papers author
  - Male/female
  - Native language: English/Chinese,…
- **Opinion**
  - "like" / "hate" / "neutral"
- **Emotion**
  - "angry"/"sad"/"happy"/"disgusted"/…

Conditional models

$p(Y=y|\mathbf{X}=\mathbf{x}; \mathbf{w}) \sim \exp(-||y-\mathbf{x}\cdot\mathbf{w}||^2/2\sigma^2)$ **linear regression**

$p(Y=y|\mathbf{X}=\mathbf{x}; \mathbf{w}) \sim 1/(1+\exp(-\mathbf{x}\cdot\mathbf{w}))$ **logistic regression**

- **Can be derived from the full ('generative') model**
  - $p(y|x) = p(x,y)/p(x)$
  - Requires picking a model for the distribution $p(x,y)$

Bayesian Methods

- **Use a *generative model* to approximate how data are produced**
  - Pick a category, C, with *prior probability P(C)*
  - Generate data, D, with likelihood P(D|C)
- **Estimate the MAP**
  - the *argmax* of the *posterior probability* P(C|D)

$$P(C \mid D) = \frac{P(D \mid C)P(C)}{P(D)}$$

Bayes Rule (again)

$$P(C \mid D) = \frac{P(D \mid C)P(C)}{P(D)}$$

D: document
C: category (label)

Maximum a posteriori (MAP)

$$c_{MAP} \equiv \underset{c \in C}{\operatorname{argmax}} P(c \mid D)$$

$$= \underset{c \in C}{\operatorname{argmax}} \frac{P(D \mid c)P(c)}{P(D)}$$

$$= \underset{c \in C}{\operatorname{argmax}} P(D \mid c)P(c)$$

Since *P(D)* is constant

Maximum likelihood

If all hypotheses are *a priori* equally likely, we only need to consider the P(D|c) term:

$$c_{ML} \equiv \underset{c \in C}{\operatorname{argmax}} P(D \mid c)$$

Maximum Likelihood Estimate ("MLE")

Naive Bayes Classifiers

Task: Classify a new instance $\underline{x}$ based on a tuple of attribute values $\mathbf{x} = (x_1 \ldots x_p)$ into one of the classes $c_j \in C$

$$c_{MAP} = \text{argmax}_c \, p(c|x_1, \ldots x_p)$$

$$= \text{argmax}_c \, p(x_1, \ldots x_p|c) \, p(c) \, / \, p(x_1, \ldots x_p)$$

$$= \text{argmax}_c \, p(x_1, \ldots x_p|c) \, p(c)$$

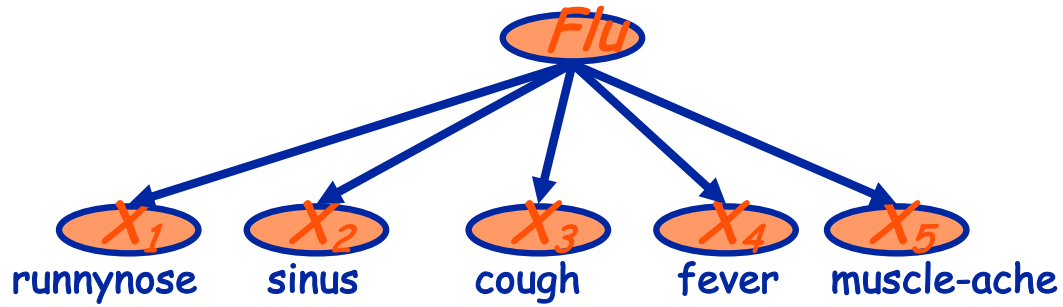
Naïve Bayes Classifier: Assumption

- $P(c_j)$
  - Estimate from the training data.
- $P(x_1,x_2,\ldots,x_p|c_j)$
  - $O(|X|^p \bullet |C|)$ parameters
  - Could only be estimated if a very, very large number of training examples was available.

Naïve Bayes assumes Conditional Independence:

- Assume that the probability of observing the conjunction of attributes is equal to the product of the individual probabilities $P(x_i|c_j)$.

The Naïve Bayes Classifier

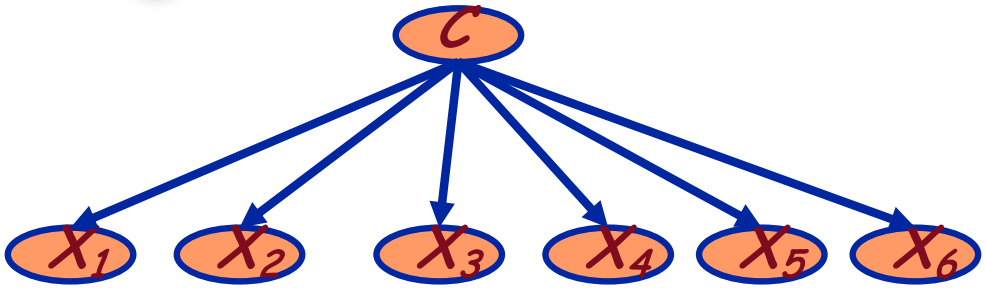


- Conditional Independence Assumption: **Features are independent of each other given the class:**

$$P(X_1,\ldots,X_5 \mid C) = P(X_1 \mid C) \bullet P(X_2 \mid C) \bullet \cdots \bullet P(X_5 \mid C)$$

- **This model is appropriate for binary variables**
  - Similar models work more generally ("Belief Networks")

Learning the Model

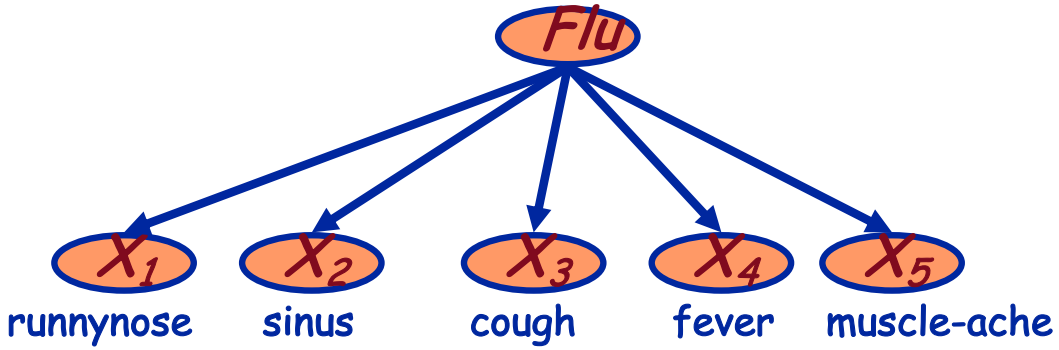


- **First attempt: maximum likelihood estimates**
  - simply use the frequencies in the data

$$\hat{P}(c_j) = \frac{N(C = c_j)}{N}$$

$$\hat{P}(x_i \mid c_j) = \frac{N(X_i = x_i, C = c_j)}{N(C = c_j)}$$

Problem with Max Likelihood



- What if we have seen no training cases where patient had no flu and muscle aches?

$$P(X_1,\ldots,X_5 \mid C) = P(X_1 \mid C) \bullet P(X_2 \mid C) \bullet \cdots \bullet P(X_5 \mid C)$$

$$\hat{P}(X_5 = t \mid C = flu) = \frac{N(X_5 = t, C = flu)}{N(C = flu)} = 0$$

- Zero probabilities cannot be conditioned away, no matter the other evidence!

$$\ell = \arg\max_c \hat{P}(c) \prod_i \hat{P}(x_i \mid c)$$

MLE Estimate

$$P(x_i|c_j) = N(X_i=true, C=c_j) / N(C=c_j)$$

$N(C=c_j)$ = # of docs in class c_j
$N(X_i=true, C=c_j)$ = # of docs in class c_j containing word x_i

MAP Estimate

- Add one document to each class with a single count of each word

$$\hat{P}(x_i \mid c_j) = \frac{N(X_i = true, C = c_j) + 1}{N(C = c_j) + v}$$

- Somewhat more subtle version

$$\hat{P}(x_i \mid c_j) = \frac{N(X_i = true, C = c_j) + mp_i}{N(C = c_j) + m}$$

overall fraction of docs containing x_i

extent of "smoothing"

$N(C=c_j)$ = # of docs in class c_j
$N(X_i=true, C=c_j)$ = # of docs in class c_j containing word x_i,
v = vocabulary size
p_i = probability that word i is present in a document, ignoring class labels

Naïve Bayes: Learning

- From training corpus, determine *Vocabulary*

- Estimate $P(c_j)$ and $P(x_k \mid c_j)$
  - For each c_j in C do

    $docs_j \leftarrow$ documents labeled with class c_j

$$P(c_j) \leftarrow \frac{|docs_j|}{|\text{total \# documents}|}$$

  - For each word x_k in *Vocabulary*

    $n_k \leftarrow$ number of occurrences of x_k in all $docs_j$

$$P(x_k \mid c_j) \leftarrow \frac{n_k + 1}{|docs_j| + |Vocabulary|}$$

"Laplace" smoothing

Naïve Bayes: Classifying

- For all words x_i in current document
- Return c_{NB}, where

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in documant} P(x_i \mid c_j)$$

What is the implicit assumption hidden in this?

Naïve Bayes for text

- **The "correct" model would have a probability for each word observed and one for each word *not* observed.**
  - Naïve Bayes for text assumes that there is no information in words that are not observed – since most words are very rare, their probability of *not* being seen is close to 1.

Naive Bayes is not so dumb

- **A good baseline for text classification**
- **Optimal if the independence assumptions hold:**
- **Very fast:**
  - Learns with one pass over the data
  - Testing linear in the number of attributes and of documents
  - Low storage requirements

Technical Detail: Underflow

- Multiplying lots of probabilities, which are between 0 and 1 by definition, can result in floating-point underflow.
- Since *log(xy)* = *log(x)* + *log(y)*, it is better to perform all computations by summing logs of probabilities rather than multiplying probabilities.
- Class with highest final un-normalized log probability score is still the most probable.

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} \log P(c_j) + \sum_{i \in positions} \log P(x_i \mid c_j)$$

More Facts About Bayes Classifiers

- **Bayes Classifiers can be built with real-valued inputs**
  - Or many other distributions
- **Bayes Classifiers don't try to be maximally discriminative**
  - They merely try to honestly model what's going on
- **Zero probabilities give stupid results**
- **Naïve Bayes is wonderfully cheap**
  - And handles 1,000,000 features cheerfully!

Naïve Bayes – MLE

word	topic	count
a	sports	0
ball	sports	1
carrot	sports	0
game	sports	2
I	sports	2
saw	sports	2
the	sports	3

P(a | sports) = 0/5

P(ball | sports) = 1/5

Assume 5 sports documents

Counts are number of documents on the sports topic containing each word

Naïve Bayes – prior (noninformative)

Word	topic	count
a	sports	0.5
ball	sports	0.5
carrot	sports	0.5
game	sports	0.5
I	sports	0.5
saw	sports	0.5
the	sports	0.5

Assume 5 sports documents

Adding a count of 0.5
beta(0.5,0.5) is a *Jeffreys* prior.

A count of 1
beta(1,1) is *Laplace smoothing.*

Pseudo-counts to be added to the observed counts

We did 0.5 here: before in the notes it was 1: either is fine

Naïve Bayes – posterior (MAP)

Word	topic	count
a	sports	0.5
ball	sports	1.5
carrot	sports	0.5
game	sports	2.5
I	sports	2.5
saw	sports	2.5
the	sports	3.5

P(a | sports) = 0.5/8.5

P(ball | sports) = 1.5/8.5

Assume 5 sports documents,

$$P(\text{word}|\text{topic}) = \frac{N(\text{word},\text{topic})+0.5}{N(\text{topic}) + 0.5\,k}$$

Pseudo count of docs on topic=sports is (5 + 0.5*7=8.5)

posterior

But words have different 'base rates'

word	topic	count	topic	count	p(word)
a	sports	0	politics	2	2/11
ball	sports	1	politics	0	1/11
carrot	sports	0	politics	0	0/11
game	sports	2	politics	1	3/11
I	sports	2	politics	5	7/11
saw	sports	2	politics	1	3/11
the	sports	3	politics	5	8/11

Assume 5 sports docs ***and 6 politics docs*** ***11 total docs***

Naïve Bayes – posterior (MAP)

$$P(word,topic) = \frac{N(word,topic) + m\,P_{word}}{N(topic) + m}$$

Arbitrarily pick m=4 as the strength of our prior

P(a | sports) = (0 + 4*(2/11))/(5 + 4) = 0.08

P(ball | sports) = (1 + 4*(1/11))/(5 + 4) = 0.15

… Our prior for p(a) is 2/11; 'a' shows up in 2 of 11 documents.
We observe 0 counts of 'a' in sports of 5 documents, and add m p(a) times or 4 * 2/11
so we 'see' 0 + 4*2/11 counts of a in 5 + 4 pseudo documents.

What you should know

- **Applications of document classification**
  - Sentiment analysis, topic prediction, email routing, author ID
- **Naïve Bayes**
  - **As MAP estimator** (uses prior for smoothing)
    - Contrast MLE
  - **For document classification**
    - Use bag of words; ignore missing words
- **Now mostly replaced with deep learning on vector embeddings**