Introduction to NLP

513.

Naïve Bayes

Generative vs. Discriminative Models

Generative

- Learn a model of the joint probability p(d, c)
- Use Bayes' Rule to calculate p(c|d)
- Build a model of each class; given example, return the model most likely to have generated that example
- Examples:
  - Naïve Bayes
  - HMM
  - Probabilistic CFG

Discriminative

- Model posterior probability p(c|d) directly
- Class is a function of document vector
- Find the exact function that minimizes classification errors on the training data
- Examples:
  - Logistic regression
  - Neural Networks (NNs)
  - Support Vector Machines (SVMs)
  - Decision Trees

Assumptions of Discriminative Classifiers

- Data examples (documents) are represented as vectors of features (words, phrases, ngrams, etc)
- Looking for a function that maps each vector into a class
- This function can be found by minimizing the errors on the training data (plus other various criteria)
- Different classifiers vary on what the function looks like, and how they find the function

Discriminative vs. Generative

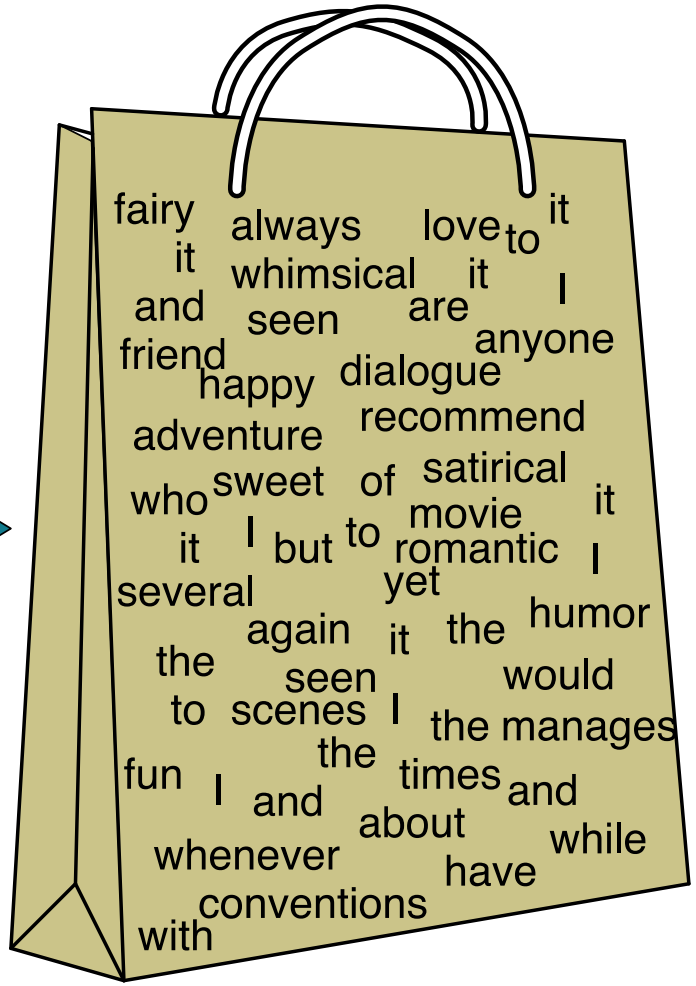
- Discriminative classifiers are generally more effective, since they directly optimize the classification accuracy. But
  - They are all sensitive to the choice of features, and so far these features are extracted heuristically
  - Also, overfitting can happen if data is sparse
- Generative classifiers are the “opposite”
  - They directly model text, an unnecessarily harder problem than classification
  - They can easily exploit unlabeled data

Naïve Bayes Intuition

- Simple (“naïve”) classification method based on Bayes rule
- Relies on very simple representation of document
  - Bag of words

Bag of Words

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!



it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
…	…

Remember Bayes' Rule?

Bayes' Rule: $$P(c|d) = \frac{P(d|c)P(c)}{P(d)}$$

- *d* is the document (represented as a list of features of a document, x_1, …, x_n)
- *c* is a class (e.g., "not spam")

Naïve Bayes Classifier

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

MAP is "maximum a posteriori" = most likely class

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

Bayes Rule

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

Dropping the denominator

Naïve Bayes Classifier

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

$$= \underset{c \in C}{\operatorname{argmax}} P(x_1, x_2, \ldots, x_n \mid c) P(c)$$

Document d represented as features x1..xn

But where will we get these probabilitites?

Naïve Bayes Classifier

- Naïve Bayesian classifier

$$P(d \in C | F_1, F_2, .. F_k) = \frac{P(F_1, F_2, .. F_k | d \in C) P(d \in C)}{P(F_1, F_2, .. F_k)}$$

- Assuming statistical independence

$$P(d \in C | F_1, F_2, .. F_k) = \frac{\prod_{j=1}^{k} P(F_j | d \in C) P(d \in C)}{\prod_{j=1}^{k} P(F_j)}$$

- Features = words (or phrases) typically

Training Naïve Bayes

function TRAIN NAIVE BAYES(D, C) **returns** log $P(c)$ and log $P(w|c)$

for each class $c \in C$ # Calculate $P(c)$ terms
  N_{doc} = number of documents in D
  N_c = number of documents from D in class c
  $logprior[c] \leftarrow \log \frac{N_c}{N_{doc}}$
  $V \leftarrow$ vocabulary of D
  $bigdoc[c] \leftarrow$ **append**(d) **for** d $\in$ D **with** class c
  for each word w in V # Calculate $P(w|c)$ terms
    $count(w,c) \leftarrow$ # of occurrences of w in $bigdoc[c]$
    $loglikelihood[w,c] \leftarrow \log \frac{count(w,c) + 1}{\sum_{w' \text{ in } V} (count(w',c) + 1)}$
return $logprior$, $loglikelihood$, V

function TEST NAIVE BAYES($testdoc$, $logprior$, $loglikelihood$, C, V) **returns** best c

for each class $c \in C$
  $sum[c] \leftarrow logprior[c]$
  for each position i in $testdoc$
    $word \leftarrow testdoc[i]$
    if $word \in V$
      $sum[c] \leftarrow sum[c] + loglikelihood[word,c]$
return $\text{argmax}_c\ sum[c]$

Figure 4.2 The naive Bayes algorithm, using add-1 smoothing. To use add-α smoothing instead, change the $+1$ to $+\alpha$ for loglikelihood counts in training.

Learning the Parameters

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

$$\hat{P}(c_j) = \frac{doccount(C = c_j)}{N_{doc}}$$

$$\hat{P}(w_i \mid c_j) = \frac{count(w_i, c_j)}{\sum_{w \in V} count(w, c_j)}$$

Parameter Estimation

$$\hat{P}(w_i \mid c_j) = \frac{count(w_i, c_j)}{\sum_{w \in V} count(w, c_j)}$$

fraction of times word w_i appears among all words in documents of topic c_j

- Create mega-document for topic *j* by concatenating all docs in this topic
  - Use frequency of *w* in mega-document

Problem with Maximum Likelihood

- What if we have seen no training documents with the word ***fantastic*** and classified in the topic **positive** (***thumbs-up)***?

$$\hat{P}(\text{"fantastic"}|\text{positive}) = \frac{count(\text{"fantastic"}, \text{positive})}{\sum_{w \in V} count(w, \text{positive})} = 0$$

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

$$c_{MAP} = \text{argmax}_c \hat{P}(c) \prod \hat{P}(x_i | c)$$

Laplace Smoothing

$$\hat{P}(w_i|c) = \frac{count(w_i, c)}{\sum_{w \in V} (count(w, c))}$$

$$\hat{P}(w_i|c) = \frac{count(w_i, c) + 1}{\sum_{w \in V} (count(w, c) + 1)}$$

$$= \frac{count(w_i, c) + 1}{\left(\sum_{w \in V} count(w, c)\right) + |V|}$$

Multinomial Naïve Bayes Independence Assumptions

$$P(x_1, x_2, \ldots, x_n \mid c)$$

- Bag of Words assumption
  - Assume position doesn't matter
- Conditional Independence
  - Assume the feature probabilities $P(x_i \mid c)$ are independent given the class c.

$$P(x_1, \ldots, x_n \mid c) = P(x_1 \mid c) \bullet P(x_2 \mid c) \bullet P(x_3 \mid c) \bullet \ldots \bullet P(x_n \mid c)$$

[Jurafsky and Martin]

Multinomial Naïve Bayes

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}} P(x_1, x_2, \ldots, x_n \mid c) P(c)$$

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} P(c_j) \prod_{x \in X} P(x \mid c)$$

This is why it's naïve!

[Jurafsky and Martin]

Multinomial NB - Learning

- From training corpus, extract *Vocabulary*

- Calculate $P(c_j)$ terms
  - For each c_j in C do

    $docs_j \leftarrow$ all docs with class $=c_j$

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

- Calculate $P(w_k \mid c_j)$ terms
  - $Text_j \leftarrow$ single doc containing all $docs_j$
  - For each word w_k in *Vocabulary*

    $n_k \leftarrow$ # of occurrences of w_k in $Text_j$

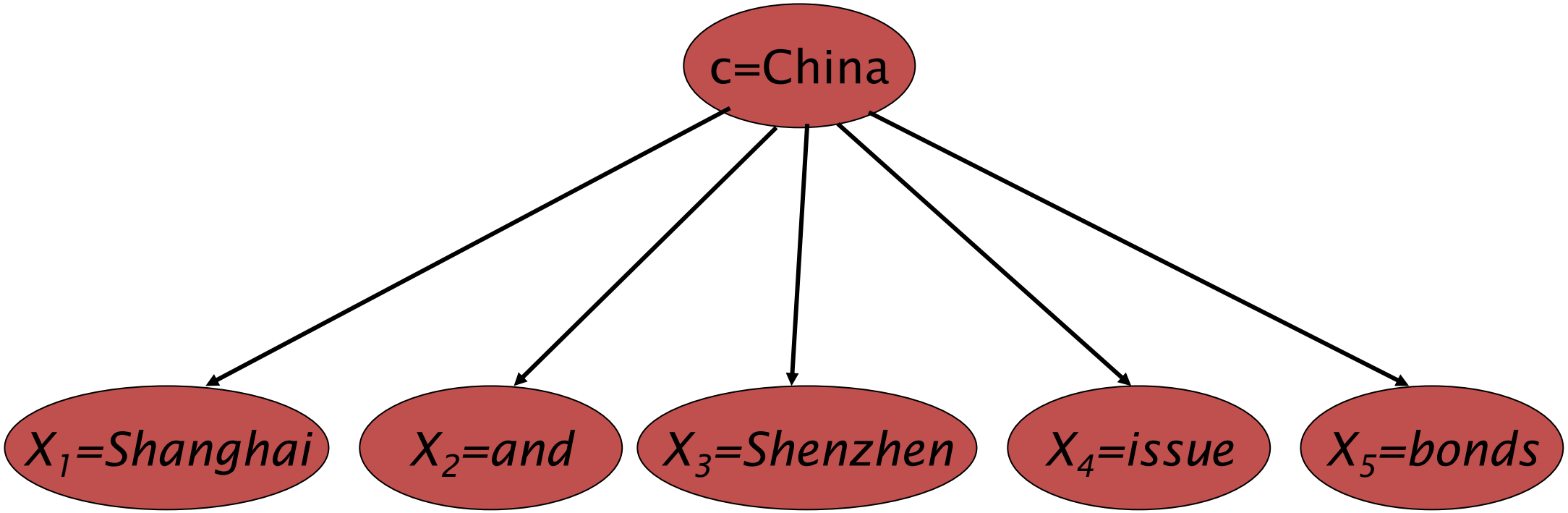
$$P(w_k \mid c_j) \leftarrow \frac{n_k + \alpha}{n + \alpha |Vocabulary|}$$

Multinomial NB Classifier for Text

positions $\leftarrow$ all word positions in test document

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

Generative Model for Multinomial NB



Example

- Features =
  [I hate love this book]
- Training
  - I hate this book = [1 1 0 1 1]
  - Love this book = [0 0 1 1 1]
- What is P(Y|X)?
- Prior p(Y)
- Testing
  - hate book = [0 1 0 0 1]
- Different conditions
  - a = 0 (no smoothing)
  - a = 1 (smoothing)

$$P(Y) = [1/2 \quad 1/2]$$

a = 0 (no smoothing)

$$M = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

$$P(X|Y) = \begin{bmatrix} 1/4 & 1/4 & 0 & 1/4 & 1/4 \\ 0 & 0 & 1/3 & 1/3 & 1/3 \end{bmatrix}$$

$$P(Y|X) \propto [1/2 \times 1/4 \times 1/4 \quad 1/2 \times 0 \times 1/3]$$
$$= [1 \quad 0]$$

a = 1 (smoothing)

$$M = \begin{bmatrix} 2 & 2 & 1 & 2 & 2 \\ 1 & 1 & 2 & 2 & 2 \end{bmatrix}$$

$$P(X|Y) = \begin{bmatrix} 2/9 & 2/9 & 1/9 & 2/9 & 2/9 \\ 1/8 & 1/8 & 2/8 & 2/8 & 2/8 \end{bmatrix}$$

$$P(Y|X) \propto [1/2 \times 2/9 \times 2/9 \quad 1/2 \times 1/8 \times 2/8]$$
$$= [0.613 \quad 0.387]$$

Another Example

this movie was great! would watch again	+
I liked it well enough for an action flick	+
I expected a great film and left happy	+
brilliant directing and stunning visuals	+
that film was awful, I'll never watch again	—
I didn't really like that movie	—
dry and a bit distasteful, it misses the mark	—
great potential but ended up being a flop	—

$$P(+) = \frac{1}{2}$$

$$P(-) = \frac{1}{2}$$

$$P(\text{great}|+) = \frac{1}{2}$$

$$P(\text{great}|-) = \frac{1}{4}$$

it was great ⟶ $P(y|x) \propto \begin{bmatrix} P(+)P(\text{great}|+) \\ P(-)P(\text{great}|-) \end{bmatrix} = \begin{bmatrix} 1/4 \\ 1/8 \end{bmatrix} = \begin{bmatrix} 2/3 \\ 1/3 \end{bmatrix}$

[Greg Durrett]

Summary (1)

- Data point $x = (x_1, ..., x_n)$, label $y \in \{0, 1\}$
- Formulate a probabilistic model that places a distribution $P(x, y)$
- Compute $P(y|x)$, predict $\text{argmax}_y P(y|x)$ to classify

$$P(y|x) = \frac{P(y)P(x|y)}{P(x)}$$

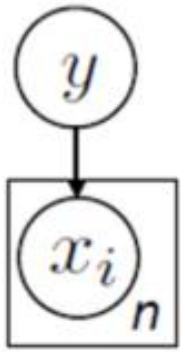
Bayes' Rule

$P(x)$ ← constant: irrelevant for finding the max

$$\propto P(y)P(x|y)$$

"Naive" assumption:

$$= P(y)\prod_{i=1}^{n} P(x_i|y)$$



linear model!

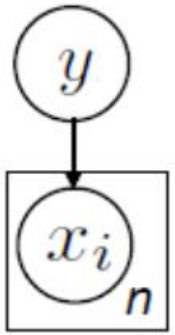
$$\text{argmax}_y P(y|x) = \text{argmax}_y \log P(y|x) = \text{argmax}_y \left[\log P(y) + \sum_{i=1}^{n} \log P(x_i|y)\right]$$

[Greg Durrett]

Summary (2)

- Model

$$P(x, y) = P(y) \prod_{i=1}^{n} P(x_i|y)$$



- Inference

$$\text{argmax}_y \log P(y|x) = \text{argmax}_y \left[\log P(y) + \sum_{i=1}^{n} \log P(x_i|y) \right]$$

- Alternatively: $\log P(y = +|x) - \log P(y = -|x) > 0$

$$\Leftrightarrow \log \frac{P(y = +|x)}{P(y = -|x)} + \sum_{i=1}^{n} \log \frac{P(x_i|y = +)}{P(x_i|y = -)} > 0$$

- Learning: maximize $P(x, y)$ by reading counts off the data

[Greg Durrett]

Not So Naïve after all!

- Very fast, low storage requirements
- Robust to irrelevant features
- Irrelevant features cancel each other without affecting results
- Very good in domains with many equally important features
  - Decision trees suffer from *fragmentation* in such cases – especially if little data
- Optimal if the independence assumptions hold
  - If assumed independence is correct, then it is the Bayes Optimal Classifier for problem
- A good, dependable baseline for text classification
  - But other classifiers give better accuracy

[Jurafsky and Martin]

Naïve Bayes is a Generative Model

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} \overbrace{P(d|c)}^{\text{likelihood}} \overbrace{P(c)}^{\text{prior}}$$

Scoping

```
didnt like this movie , but I
```

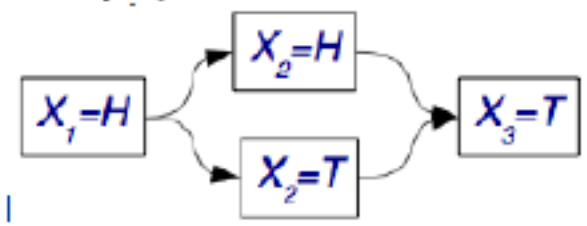
becomes

```
didnt NOT_like NOT_this NOT_movie , but I
```

Missing features

One of the key strengths of Bayesian approaches is that they can naturally handle missing data

- What happens if we don't have value of some feature $x_k^{(i)}$
  - e.g., applicants credit history unknown
  - e.g., some medical tests not performed
- How to compute $\Pr(x_1, x_2, \ldots x_{j-1}, ?, x_{j+1} \ldots, x_d | y)$?
  - e.g., three coin tosses $\mathrm{E} = \{H, ?, T\}$
  - $\Rightarrow \Pr(E) = \Pr(\{H, H, T\}) + \Pr(\{H, T, T\})$



- More generally

$$\Pr(x_1, x_2, \ldots x_{j-1}, ?, x_{j+1} \ldots, x_d | y) = \sum_{z_j} \Pr(x_1, x_2, \ldots x_{j-1}, z_j, x_{j+1} \ldots, x_d | y)$$

[Karl Stratos]

Missing features in naive Bayes

$$\Pr(x_1, x_2, \ldots x_{j-1}, ?, x_{j+1} \ldots, x_d | y)$$

$$= \sum_{z_j} \Pr(x_1, x_2, \ldots x_{j-1}, z_j, x_{j+1} \ldots, x_d | y)$$

$$= \sum_{z_j} \left[\Pr(z_j | y) \prod_{k \neq j} \Pr(x_k | y) \right]$$

$$= \prod_{k \neq j} \Pr(x_k | y) \sum_{z_j} \Pr(z_j | y)$$

$$= \prod_{k \neq j} \Pr(x_k | y)$$

- Simply ignore the missing values and compute likelihood based only observed features
- no need to fill-in or explicitly model missing values

[Karl Stratos]