Carnegie Mellon University
Language
Technologies
Institute

11-411/11-611 Natural Language Processing

Naive Bayes Classifiers

David R. Mortensen (after Kemal Oflazer)

September 27, 2022

Language Technologies Institute

Learning Objectives

By the end of this lecture, students should be able to do the following:

- Explain what is meant by *classification* and recognize a classification task when they see one in the wild
- Evaluate a classifier
- Reproduce the mathematical basis for a Naïve Bayes (NB) classifier
- Implement an NB classifier that uses bag-of-words feature to classify text documents (from scratch)
- Know the difference between generative and discriminative classifiers

Text Classification

- We have a set of documents (news items, emails, product reviews, movie reviews, books, ...)
- Classify this set of documents into a small set *classes*.
- Applications:
  - Topic of a news article (classic example: finance, politics, sports, ...)
  - Sentiment of a movie or product review (good, bad, neutral)
  - Email into spam or not or into a category (business, personal, bills, ...)
  - Reading level (K-12) of an article or essay
  - Author of a document (Shakespeare, James Joyce, ...)
  - Genre of a document (report, editorial, advertisement, blog, ...)
  - Language identification

Notation and Setting

- We have a set of n documents (texts) $d_i \in \mathcal{V}^+$, where $\mathcal{V}$ is the vocabulary of the corpus.
  - We assume the texts are segmented already.
- We have set $\mathcal{L}$ of labels, ℓ_i
- Human experts annotate documents with labels and give us $\{(d_1, \ell_1), (d_2, \ell_2), \cdots, (d_n, \ell_n)\}$
- We learn a *classifier* `classify` : $\mathcal{V}^+ \to \mathcal{L}$ with this labeled training data.
- Afterwards, we use `classify` to classify new documents into their classes.

We Can Evaluate a Classifier Using Accuracy

Accuracy is our first shot.

- Accuracy:

$$\mathrm{A}(\texttt{classify}) = \sum_{\substack{d \in \mathcal{V}^+, \ell \in \mathcal{L}, \\ \texttt{classify}(d) = \ell}} p(x, \ell) \tag{1}$$

  where p is the true distribution over data.
- Error is $1 - A$.
- Accuracy is estimated using a test set $\{(\overline{d}_1, \overline{\ell}_1), (\overline{d}_2, \overline{\ell}_2), \cdots, (\overline{d}_m, \overline{\ell}_m)\}$

$$\hat{\mathrm{A}}(\texttt{classify}) = \frac{1}{m} \sum_{i=1}^{m} \{\texttt{classify}(\overline{d}_i) = \overline{\ell}_i\} \tag{2}$$

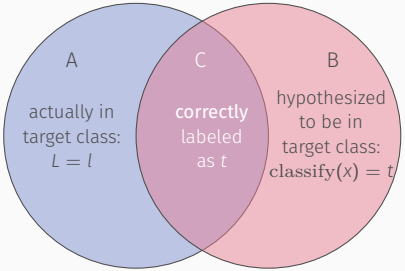
- Higher-order functions are our friends!

Issues with Using Test Set Accuracy

- Suppose that 99% of the messages you get are spam and 1% are not (we're being realistic here).
- Suppose, too, that you have a spam filter that **always** classifies messages as spam.
  1. You would get lots of work done, because you wouldn't have to answer email.
  2. The email classifier would have $\hat{A} \approx 0.99$.
  3. Everybody would be happy, except for your boss.
- You must take other things into account:
  - Relative importance of classes or cost of error types.
  - Variance due to the test data.

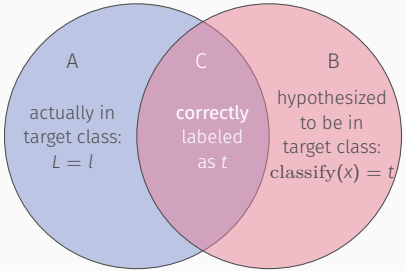
Evaluation in the Two-class case

- Suppose we have one of the classes $t \in \mathcal{L}$ as the target class.
- We would like to identify documents with label t in the test data.
  - Like information retrieval
- We get



- Precision $\hat{P} = \frac{C}{B}$ (percentage of documents `classify` correctly labeled as t)
- Recall $\hat{R} = \frac{C}{A}$ (percentage of actual t labeled documents correctly labeled as t)
- $F_1 = 2\frac{\hat{P} \cdot \hat{R}}{\hat{P} + \hat{R}}$

A Different View – Contingency Tables



	$L = t$	$L \neq t$	
$\mathbf{classify}(X) = t$	C (true positives)	$B \backslash C$ (false positives)	B
$\mathbf{classify}(X) \neq t$	$A \backslash C$ (false negatives)	(true negatives)	
	A		

Evaluation with > 2 Classes

- **Macroaveraged precision and recall**: let each class be the target and report the average $\hat{P}$ and $\hat{R}$ across all classes.
- **Microaveraged precision and recall**: pool all one-vs.-rest decisions into a single contingency table, calculate $\hat{P}$ and $\hat{R}$ from that.

How I Remember Precision and Recall

I find contingency tables to be the easiest way to remember how to calculate and reason about precision and recall. Precision is the number of true positives over the total number of positives:

$$\hat{P} = \frac{|TP|}{|TP \cup FP|}$$

Recall is the number of true positives over the total number of relevant elements in the reference set (the true positives and the false negatives):

$$\hat{R} = \frac{|TP|}{|TP \cup FN|}$$

Mnemonic: TP is always in on top and bottom. Precision adds FP to the denominator while recall adds FN.

Cross-validation

- Remember that $\hat{A}$, $\hat{P}$, $\hat{R}$, and $\hat{F_1}$ are all estimates of the classifier's quality under the true data distribution.
  - Estimates are noisy!
- K-fold cross validation
  - Partition the training data into K nonverlapping "folds", $\boldsymbol{d}^1, \boldsymbol{d}^2, \ldots, \boldsymbol{d}^K$,
  - For $i \in \{1, \ldots, K\}$
    - Train on $\boldsymbol{d}_{1:n} \backslash \boldsymbol{d}^i$, using $\boldsymbol{d}^i$ as development data
    - Estimate quality on the $\boldsymbol{d}^i$ development set as $\hat{\mathrm{A}}^i$
  - Report average accuracy as $\hat{\mathrm{A}} = \frac{1}{K} \sum_{i=1}^{K} \hat{\mathrm{A}}^i$ and perhaps also the standard deviation.

Features in Text Classification

- Running example $\boldsymbol{d}$ = “The spirit is willing but the flesh is weak”
- *Feature random variables*
- For $j \in \{1, \ldots, d\}$ F_j is a discrete random variable taking values in $\mathcal{F}_j$
- Most of the time these can be frequencies of words or n-grams (including character n-grams) in a text.
  - $f_{f-spirit} = 1, f_{f-is} = 2, f_{f-the-flesh} = 1, \ldots$
- They can be boolean “exists” features.
  - $f_{e-spirit} = 1, f_{e-is} = 1, f_{f-strong} = 0, \ldots$

Spam Detection

- A training set of email messages (marked *Spam* or *Not-Spam*)
- A set of features for each message (considered as a bag of words)
  - For each word: Number of occurrences
  - Whether phrases such as “Nigerian Prince”, “email quota full”, “won ONE HUNDRED MILLION DOLLARS” are in the message
  - Whether it is from someone you know
  - Whether it is a reply to your message
  - Whether it is from your domain (e.g., `cmu.edu`)

Movie Ratings

- A training set of movie reviews (with star ratings 1 - 5)
- A set of features for each message (considered as a bag of words)
  - For each word: Number of occurrences
  - Whether phrases such as *Excellent*, *sucks*, *blockbuster*, *biggest*, *Star Wars*, *Disney*, *Adam Sandler*, ...are in the review

Probabilistic Classification

- Documents are preprocessed: each document $\boldsymbol{d}$ is mapped to a d-dimensional feature vector $\boldsymbol{f}$.
- Classification rule

(3)

Probabilistic Classification

- Documents are preprocessed: each document $\boldsymbol{d}$ is mapped to a d-dimensional feature vector $\boldsymbol{f}$.
- Classification rule

$$\text{classify}(\boldsymbol{f}) \quad = \quad \underset{\ell \in \mathcal{L}}{\text{argmax}}\, p(\ell \mid \boldsymbol{f}) \tag{3}$$

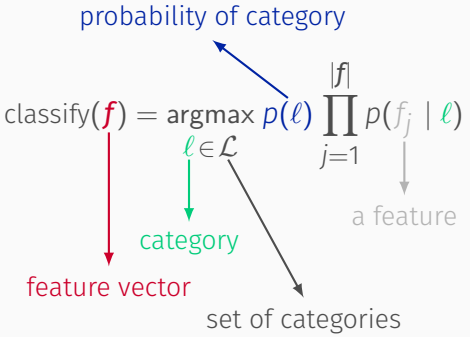
Probabilistic Classification

- Documents are preprocessed: each document $\boldsymbol{d}$ is mapped to a d-dimensional feature vector $\boldsymbol{f}$.
- Classification rule

$$\begin{aligned} \text{classify}(\boldsymbol{f}) &= \operatorname*{argmax}_{\ell \in \mathcal{L}} p(\ell \mid \boldsymbol{f}) \\ &= \operatorname*{argmax}_{\ell \in \mathcal{L}} \frac{p(\ell, \boldsymbol{f})}{p(\boldsymbol{f})} \\ &= \operatorname*{argmax}_{\ell \in \mathcal{L}} p(\ell, \boldsymbol{f}) \text{(Why?)} \end{aligned} \tag{3}$$

Overview

- The feature vector $\boldsymbol{f}$ consists of the features that have been extracted from the document
- Each element of this vector, f_j, corresponds to a feature such as the presence of a particular word or ngram, the length of the document, and so on.
- There is a set of categories or labels, $\mathcal{L}$ (like 1 star, 2 stars,..., 5 stars).
- Each of these labels, ℓ, corresponds to a category like “five stars” or “negative polarity.”
- π_ℓ is the “base” probability of ℓ, disregarding any of the features in $\boldsymbol{f}$



But what about $p(f_j \mid \ell)$? How do we calculate it? You’ll find out soon.

Three kinds of Naive Bayes

How you calculate $p(f_j \mid \ell)$ varies depending on what kind of classifier you are developing. There are three of these:

- **Gaussian Naïve Bayes** Considers a feature vector of continuous variables
- **Multinomial Naïve Bayes** Considers a feature vector representing frequencies
- **Bernoulli Naïve Bayes** Considers a vector of binary features

For sentiment analysis, language ID, and other document classification tasks, Multinomial Naïve Bayes is generally best.

Getting to (Multinomial) Naive Bayes

$$p(\ell|\mathbf{f}) = \frac{p(\ell)p(\mathbf{f}|\ell)}{p(\mathbf{f})}$$

We are only interested in the numerator, which is equivalent to:

$$p(\ell, f_i, \ldots, f_n) \tag{4}$$

Via the chain rule, we get:

$$p(\ell, f_1, \ldots, f_n) = p(f_1|f_2, \ldots, f_n, \ell)p(f_2|f_3, \ldots, f_n, \ell), \ldots, p(f_{n-1}|f_n, \ell)p(f_n|\ell)p(\ell) \tag{5}$$

Assuming conditional independence, we get:

$$p(f_i|f_{i+1}, \ldots, f_n, \ell) = p(f_i|\ell) \tag{6}$$

Getting to Naive Bayes

We can thus express the joint model as:

$$\begin{aligned} p(\ell, f_1, \ldots, f_n) &\propto p(\ell, f_1, \ldots, f_n) \\ &\propto p(\ell)p(f_1|\ell)p(f_2|\ell)p(f_3|\ell)\ldots \\ &\propto p(\ell)\prod_{i=1}^{n} p(f_i|\ell) \end{aligned}$$

Naive Bayes Classifier

$$p(L = \ell, F_1 = f_1, \ldots, F_d = f_d) = p(\ell) \prod_{j=1}^{d} p(F_j = f_j \mid \ell) \tag{7}$$

$$= \pi_\ell \prod_{j=1}^{d} \theta_{f_j|j,\ell}$$

- Parameters: π_ℓ is the class or label prior (equivalent to $p(\ell)$, but fancier, because it's Greek).
  - The probability that a document belongs to class ℓ – without considering any of its features.
  - They can be computed directly from the training data $\{(f_1, \ell_1), (f_2, \ell_2), \cdots, (f_n, \ell_n)\}$. These sum to 1.
- For each feature function j and label ℓ, a distribution over values $\theta_{*|j,\ell}$
  - These sum to 1 for every (j, ℓ) pair.

Generative vs Discriminative Classifier

- Naive Bayes is known as a Generative Classifier.
- Generative Classifiers build a model of each class.
- Given an observation (document), they return the class most likely have generated that observation.

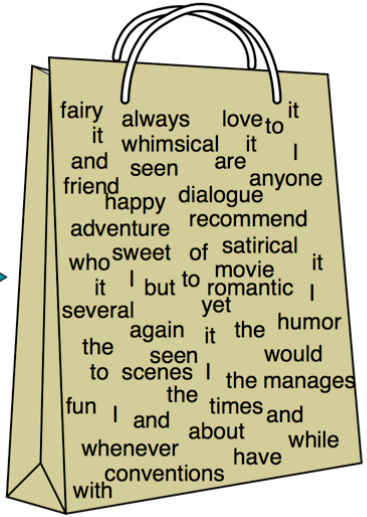
- A Discriminative Classifier instead learns what features from the input are useful to discriminate between possible classes.

Bayesians Against Discrimination

The Most Basic Naive Bayes Classifier

(From Jurafsky's slides).

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
…	…

The Most Basic Naive Bayes Classifier for Documents

- Features are just words f_j in $\boldsymbol{d}$
- **Naive Assumption**: Word positions do not matter—"bag of words".
- **Conditional Independence**: Feature probabilities $p(f_i \mid \ell)$ are independent given the class ℓ.
  - $p(\boldsymbol{d} \mid \ell) = \prod_{j=1}^{|\boldsymbol{d}|} p(d_j \mid \ell)$
  - The probability that a word in a sports document is "soccer" is estimated as $p(\text{soccer} \mid \text{sports})$ by counting "soccer" in all sports documents.
- So

$$\text{classify}(\boldsymbol{d}) = \text{argmax}\, \ell \in \mathcal{L} \pi_\ell \prod_{j=1}^{|\boldsymbol{d}|} p(d_j \mid \ell)$$

- Smoothing is very important as any new document may have unseen words.

We Must Compute Probabilities for Some Words not in the Training Data

Naive approach to estimating $\hat{P}(\ell)$ and $\hat{P}(w_i|\ell)$:

$$\hat{P}(\ell) = \frac{N_\ell}{N_{\text{total}}}$$

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

where N_{total} is the total number of documents, N_ℓ is the number of documents with label ℓ, V is the vocabulary of the corpus, and w_i is a word in that vocabulary. But what happens if no training document classified as NEGATIVE contains the word *terrible*?

$$\hat{P}(\text{terrible}|\text{NEGATIVE}) = \frac{\text{count}(\text{terrible}, \text{NEGATIVE})}{\sum_{w \in V} \text{count}(w, \text{NEGATIVE})} = 0$$

Zero probabilities cannot be conditioned away, no matter what other evidence there may be!

When We Count, We Must Smooth

A simple solution is to use Laplace, or add-K smoothing:

- Add α (usually 1, thus the term "add-one smoothing") to the numerator (so that the co-occurrence count of the class and the word $\geq K$)
- Add α for each word in the vocabulary to the denominator

$$\hat{P}(w_i|\ell) = \frac{\text{count}(w_i, \ell) + 1}{\sum_{w \in V}(\text{count}(w, \ell) + 1)} \tag{8}$$

$$= \frac{\text{count}(w_i, \ell) + 1}{\left(\sum_{w \in V} \text{count}(w, \ell)\right) + |V|} \tag{9}$$

Laplace smoothing is not good for language modeling, but it's easy and it works well for classification.

The Most Basic Naive Bayes Classifier

$$\text{classify}(d) = \text{argmax}\, \ell \in \mathcal{L} p(\ell) \prod_{i=1}^{|d|} p(w_i \mid \ell)$$

$$\Downarrow$$

$$\text{classify}(d) = \text{argmax}\, \ell \in \mathcal{L} \left(\log p(\ell) + \sum_{i=1}^{|d|} \log p(w_i \mid \ell) \right)$$

- All computations are done in **log** space to avoid underflow and increase speed.
- Class prediction is based on a linear combination of the inputs.
- Hence Naive Bayes is considered as a *linear classifier*.

A Blueprint for NB

To classify a document d:

$$\text{classify}(d) = \text{argmax}\, c \in C \left(\log\, p(c) + \sum_{i=1}^{|d|} \log\, p(w_i \mid c) \right)$$

Training:

$\mathcal{D} \leftarrow$ all documents
for each $c \in C$ **do**
  $\mathcal{D}_c \leftarrow$ docs with class c
  $p(c) \leftarrow \frac{|\mathcal{D}_c|}{|\mathcal{D}|}$

Training:

$T_c \leftarrow \text{concatenate}(\mathcal{D}_c)$
for each $w_i \in V$ **do**
  $n_i \leftarrow \text{occurrences}(w_i, T_c)$
  $p(w_i|c) \leftarrow \frac{n_i + \alpha}{|T_c| + \alpha|V|}$

At inference time, compute smoothed probabilities for words that were not in the training set.

Algorithm for Training Multinomial NB

1: **function** TRAINNB(D: list of $\langle \text{label}, \text{string} \rangle$, n: integer, k: real)
2: **for each** $\ell, d \in D$ **do**
3: $G \leftarrow \text{extract}(d)$
4: $V \leftarrow V \cup G$
5: $T[\ell] \leftarrow \text{extend}(G, T[\ell])$
6: $\mathcal{L} \leftarrow \mathcal{L} \cup \{\ell\}$
7: **for each** $\ell, D_\ell \in T$ **do**
8: **for each** $d \in D_\ell$ **do**
9: **for each** $w \in d$ **do**
10: $F[w][\ell] \leftarrow F[w][\ell] + 1$
11: $\pi[\ell] \leftarrow log \frac{|D_\ell|}{|D|}$
12: **for each** $w \in V$ **do**
13: $L[w][\ell] \leftarrow log \frac{F[w][\ell]+k}{\sum_{w' \in V}(F[w'][\ell]+k)}$
14: **return** $\langle \pi, L, F, \mathcal{L}, V \rangle$

Algorithm for Inference Using a Naive Bayes Classifier

1: **function** TESTNB(d, π, L, F, $\mathcal{L}$, V)
2: **for each** $\ell \in \mathcal{L}$ **do**
3: $s[\ell] \leftarrow \pi[\ell]$
4: **for each** $w \in V$ **do**
5: **if** L is undefined **then**
6: $L[w][\ell] \leftarrow log \frac{F[w][\ell]+k}{\sum_{w' \in V}(F[w'][\ell]+k)}$
7: $s[\ell] \leftarrow s[\ell] + \mathcal{L}[w, \ell]$
8: **return** $\text{argmax}_\ell \, s[\ell]$

An Example

	ℓ	d
Training	$-$	just plain boring
	$-$	entirely predictable and lacks energy
	$-$	no surprises and few laughs
	$+$	very powerful
	$+$	the most fun film of the summer
Test	?	predictable with not fun

- $|V| = 20$
- Add 1 Laplace smoothing

An Example

	ℓ	d
Training	$-$	just plain boring
	$-$	entirely predictable and lacks energy
	$-$	no surprises and few laughs
	$+$	very powerful
	$+$	the most fun film of the summer
Test	?	predictable with not fun

- $|V| = 20$
- Add 1 Laplace smoothing

$$\pi_- = p(-) = \frac{3}{5}$$

$$\pi_+ = p(+) = \frac{2}{5}$$

$$N_- = 13$$

$$N_+ = 9$$

An Example

	ℓ	d
Training	$-$	just plain boring
	$-$	entirely predictable and lacks energy
	$-$	no surprises and few laughs
	$+$	very powerful
	$+$	the most fun film of the summer
Test	?	predictable with not fun

- $|V| = 20$
- Add 1 Laplace smoothing

$$\pi_- = p(-) = \frac{3}{5}$$

$$\pi_+ = p(+) = \frac{2}{5}$$

$$N_- = 13$$

$$N_+ = 9$$

$$p(\text{“predictable”} \mid -) = \frac{1+1}{13+20} \quad p(\text{“predictable”} \mid +) = \frac{0+1}{9+20}$$

An Example

	ℓ	d
Training	$-$	just plain boring
	$-$	entirely predictable and lacks energy
	$-$	no surprises and few laughs
	$+$	very powerful
	$+$	the most fun film of the summer
Test	?	predictable with not fun

- $|V| = 20$
- Add 1 Laplace smoothing

$$\pi_- = p(-) = \frac{3}{5}$$

$$\pi_+ = p(+) = \frac{2}{5}$$

$$N_- = 13$$

$$N_+ = 9$$

$$p(\text{“predictable”} \mid -) = \frac{1+1}{13+20} \quad p(\text{“predictable”} \mid +) = \frac{0+1}{9+20}$$

$$p(\text{“no”} \mid -) = \frac{1+1}{13+20} \quad p(\text{“no”} \mid +) = \frac{0+1}{9+20}$$

An Example

	ℓ	d
Training	$-$	just plain boring
	$-$	entirely predictable and lacks energy
	$-$	no surprises and few laughs
	$+$	very powerful
	$+$	the most fun film of the summer
Test	?	predictable with not fun

- $|V| = 20$
- Add 1 Laplace smoothing

$$\pi_- = p(-) = \frac{3}{5}$$

$$\pi_+ = p(+) = \frac{2}{5}$$

$$N_- = 13$$

$$N_+ = 9$$

$$p(\text{“predictable”} \mid -) = \frac{1+1}{13+20} \quad p(\text{“predictable”} \mid +) = \frac{0+1}{9+20}$$

$$p(\text{“no”} \mid -) = \frac{1+1}{13+20} \quad p(\text{“no”} \mid +) = \frac{0+1}{9+20}$$

$$p(\text{“fun”} \mid -) = \frac{0+1}{13+20} \quad p(\text{“fun”} \mid +) = \frac{1+1}{9+20}$$

An Example

	ℓ	d
Training	$-$	just plain boring
	$-$	entirely predictable and lacks energy
	$-$	no surprises and few laughs
	$+$	very powerful
	$+$	the most fun film of the summer
Test	?	predictable with not fun

- $|V| = 20$
- Add 1 Laplace smoothing

$$\pi_- = p(-) = \frac{3}{5}$$

$$\pi_+ = p(+) = \frac{2}{5}$$

$$N_- = 13$$

$$N_+ = 9$$

$$p(\text{“predictable”} \mid -) = \frac{1+1}{13+20} \quad p(\text{“predictable”} \mid +) = \frac{0+1}{9+20}$$

$$p(\text{“no”} \mid -) = \frac{1+1}{13+20} \quad p(\text{“no”} \mid +) = \frac{0+1}{9+20}$$

$$p(\text{“fun”} \mid -) = \frac{0+1}{13+20} \quad p(\text{“fun”} \mid +) = \frac{1+1}{9+20}$$

$$p(+)p(\boldsymbol{s} \mid +) = \frac{2}{5} \times \frac{1 \times 1 \times 2}{29^3} = 3.2 \times 10^{-5}$$

An Example

	ℓ	d
Training	$-$	just plain boring
	$-$	entirely predictable and lacks energy
	$-$	no surprises and few laughs
	$+$	very powerful
	$+$	the most fun film of the summer
Test	?	predictable with not fun

- $|V| = 20$
- Add 1 Laplace smoothing

$$\pi_- = p(-) = \frac{3}{5}$$

$$\pi_+ = p(+) = \frac{2}{5}$$

$$N_- = 13$$

$$N_+ = 9$$

$$p(\text{“predictable”} \mid -) = \frac{1+1}{13+20} \quad p(\text{“predictable”} \mid +) = \frac{0+1}{9+20}$$

$$p(\text{“no”} \mid -) = \frac{1+1}{13+20} \quad p(\text{“no”} \mid +) = \frac{0+1}{9+20}$$

$$p(\text{“fun”} \mid -) = \frac{0+1}{13+20} \quad p(\text{“fun”} \mid +) = \frac{1+1}{9+20}$$

$$p(+)p(\mathbf{s} \mid +) = \frac{2}{5} \times \frac{1 \times 1 \times 2}{29^3} = 3.2 \times 10^{-5}$$

$$p(-)p(\mathbf{s} \mid -) = \frac{3}{5} \times \frac{2 \times 2 \times 1}{33^3} = 6.7 \times 10^{-5}$$

Other Optimizations for Sentiment Analysis

- Ignore unknown words in the test.
- Ignore **stop words** like *the*, *a*, *with*, etc.
  - Remove most frequent 10-100 words from the training and test documents.
- Count vs existence of words: Binarized features.
- Negation Handling `didnt like this movie , but I` $\rightarrow$ `didnt NOT_like NOT_this NOT_movie , but I`