# Naïve Bayes for Text Classification

**Phạm Quang Nhật Minh**
Aimesoft JSC
minhpham0902@gmail.com

January 13, 2024

# Lecture Contents



- The Task of Text Classification
- The Naïve Bayes Text Classifier
- Naïve Bayes: Learning
- Sentiment and Binary Naïve Bayes
- Accuracy, Precision, Recall, and F measure

# Is this spam?



**Subject:** **Important notice!**
**From:** Stanford University <newsforum@stanford.edu>
**Date:** October 28, 2011 12:34:16 PM PDT
**To:** undisclosed-recipients:;

---

Greats News!

You can now access the latest news by using the link below to login to Stanford University News Forum.

http://www.123contactform.com/contact-form-StanfordNew1-236335.html

Click on the above link to login for more information about this new exciting forum. You can also copy the above link to your browser bar and login for more information
about the new services.

# Positive or negative movie review?



- \+ *...zany characters and richly applied satire, and some great plot twists*
- – *It was pathetic. The worst part about it was the boxing scenes...*
- \+ *...awesome caramel sauce and sweet toasty almonds. I love this place!*
- – *...awful pizza and ridiculously overpriced...*

# Positive or negative movie review?



- \+ *...zany characters and* ***richly*** *applied satire, and some* ***great*** *plot twists*
- − *It was* ***pathetic****. The* ***worst*** *part about it was the boxing scenes...*
- \+ *...****awesome*** *caramel sauce and sweet toasty almonds. I* ***love*** *this place!*
- − *...****awful*** *pizza and* ***ridiculously*** *overpriced...*

# Text Classification Tasks



- Sentiment analysis
- Spam detection
- Language identification
- Assignining categories to news articles
- …

# Text Classification: definition



- *Input*:
  - a document $d$
  - a fixed set of classes $C = \{c_1, c_2, \ldots, c_J\}$

- *Output*:
  - a predicted class $c \in C$

# Classification Methods: Hand-coded rules



- Rules based on combination of words and other features
  - spam: black-list-address OR (“dollars” AND “have been selected”)
- Accuracy can be high
  - If rules carefully refined by expert

But building and maintaining these rules is expensive

# Classification Methods: Supervised Machine Learning



*Input*:

- a document *d*
- a fixed set of classes $C = \{c_1, c_2, \ldots, c_J\}$
- A training set of *m* hand-labeled documents $D = \{(d_1, c_1), \ldots, (d_m, c_m)\}$

*Output*:

- A learned classifier $\gamma: d \rightarrow c$

# Classification Methods: Supervised Machine Learning



Any kinds of classifier

- Naïve Bayes
- Logistic regression
- Neural networks
- k-Nearest Neighbors
- …

# Lecture Contents



- The Task of Text Classification
- The Naïve Bayes Text Classifier
- Naïve Bayes: Learning
- Sentiment and Binary Naïve Bayes
- Accuracy, Precision, Recall, and F measure

# Naïve Bayes Intuition

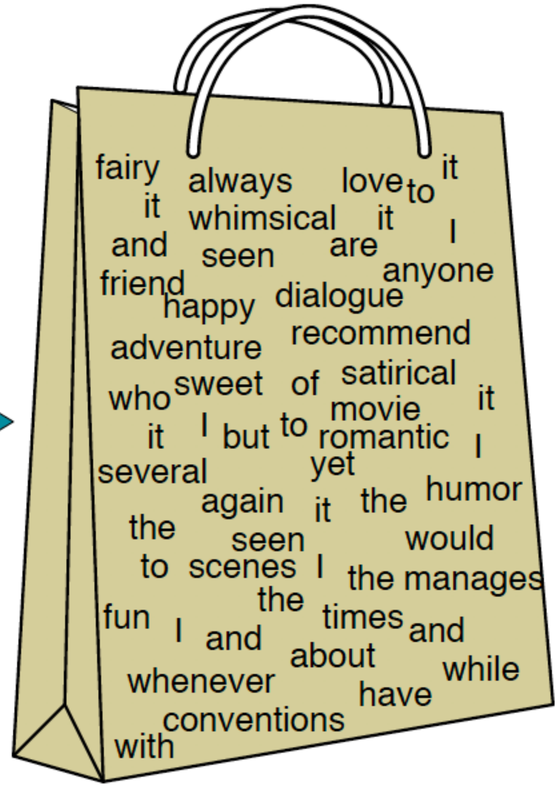


- Simple ("naïve") classification method based on Bayes rule
- Relies on very simple representation of document
  - Bag of words

# The bag of words representation



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 bag of words representation



$\gamma($

| | |
|---|---|
| seen | 2 |
| sweet | 1 |
| whimsical | 1 |
| recommend | 1 |
| happy | 1 |
| ... | ... |

$)=c$

# Bayes' Rule Applied to Documents and Classes



- For a document *d* and a class *c*

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)}$$

# Naïve Bayes Classifier (1)



- The classifier returns the class $\hat{c}$ which has the maximum posterior probability (MAP) given the document

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

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

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

Bayes Rule

Drop *P(x)* because *P(x)* is the same for all classes

# Naïve Bayes Classifier (2)



- Document *d* is represented as features $(x_1, \dots, x_n)$

"Likelihood" "Prior"

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

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

# Multinomial Naïve Bayes Independence Assumptions



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

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

$$P(x_1, x_2, \ldots, x_n | c) = P(x_1 | c) P(x_2 | c) \ldots P(x_n | c)$$

# Multinomial Naïve Bayes Classifier



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

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{i=1}^{n} P(x_i | c)$$

# Applying Naïve Bayes Classifiers to Text Classification



positions ← all word positions in test documents

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{i \in positions} P(w_i | c)$$

# Problems with multiplying lots of probs



$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{i \in positions} P(w_i | c)$$

Multiplying lots of probabilities can result in floating-point underflow!

.0006 * .0007 * .0009 * .01 * .5 * .000008….

Idea: Use logs, because log(ab) = log(a) + log(b)

We'll sum logs of probabilities instead of multiplying probabilities!

# Calculating in log space



Instead of this:

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{i \in positions} P(w_i|c)$$

Use:

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} \log P(c) + \sum_{i \in positions} \log P(w_i|c)$$

Notes:

1) Taking log doesn't change the ranking of classes!

- The class with highest probability also has highest log probability!

2) It's a linear model:

- Just a max of a sum of weights: a linear function of the inputs
- So naive bayes is a linear classifier

# Lecture Contents



- The Task of Text Classification
- The Naïve Bayes Text Classifier
- Naïve Bayes: Learning
- Sentiment and Binary Naïve Bayes
- Accuracy, Precision, Recall, and F measure

# Learning the Multinomial Naive Bayes Model



Maximum likelihood estimation (MLE)

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

$N_c$ is the number of documents in class *c* and *N* is the total number of documents

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

count(w, c) is the count of the number of word *w* occurs in documents of class *c* in the training data

# Parameter Estimation



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

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

Create mega-document for topic $j$ by concatenating all docs in this topic

- Use frequency of w in mega-document

# Problem with Maximum Likelihood



- MLE estimate gets zero for a term-class combination that did not occur in the training data.
- E.g., what if we have seen no training documents with the word *fantastic*

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

# Laplace (add-1) smoothing for Naïve Bayes



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

# Multinomial Naïve Bayes: 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 $=c_j$

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 $Text_j$

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

# Unknown words



- What about unknown words
  - that appear in our test data
  - but not in our training data or vocabulary?
- We ignore them
  - Remove them from the test document!
  - Pretend they weren't there!
  - Don't include any probability for them at all!
- Why don't we build an unknown word model?
  - It doesn't help: knowing which class has more unknown words is not generally helpful!

# Stop words



Some systems ignore stop words

- Stop words: very frequent words like the and a.
    - Sort the vocabulary by word frequency in training set
    - Call the top 10 or 50 words the stopword list.
    - Remove all stop words from both training and test sets
        - As if they were never there!

But removing stop words doesn't usually help

- So in practice most NB algorithms use all words and don't use stopword lists

# Lecture Contents



- The Task of Text Classification
- The Naïve Bayes Text Classifier
- Naïve Bayes: Learning
- Sentiment and Binary Naïve Bayes
- Accuracy, Precision, Recall, and F measure

# Let's do a worked sentiment example!

| | Cat | Documents |
|---|---|---|
| Training | - | just plain boring |
| | - | entirely predictable and lacks energy |
| | - | no surprises and very few laughs |
| | + | very powerful |
| | + | the most fun film of the summer |
| Test | ? | predictable with no fun |

# A worked sentiment example with add-1 smoothing

| | Cat | Documents |
|---|---|---|
| Training | - | just plain boring |
| | - | entirely predictable and lacks energy |
| | - | no surprises and very few laughs |
| | + | very powerful |
| | + | the most fun film of the summer |
| Test | ? | predictable ~~with~~ no fun |

1. Prior from training:

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

P(-) = 3/5
P(+) = 2/5

2. Drop "with"

3. Likelihoods from training:

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

$$P(\text{“predictable”}|-) = \frac{1+1}{14+20} \qquad P(\text{“predictable”}|+) = \frac{0+1}{9+20}$$

$$P(\text{“no”}|-) = \frac{1+1}{14+20} \qquad P(\text{“no”}|+) = \frac{0+1}{9+20}$$

$$P(\text{“fun”}|-) = \frac{0+1}{14+20} \qquad P(\text{“fun”}|+) = \frac{1+1}{9+20}$$

4. Scoring the test set:

$$\boxed{P(-)P(S|-) = \frac{3}{5} \times \frac{2 \times 2 \times 1}{34^3} = 6.1 \times 10^{-5}}$$

$$P(+)P(S|+) = \frac{2}{5} \times \frac{1 \times 1 \times 2}{29^3} = 3.2 \times 10^{-5}$$

# Optimizing for sentiment analysis

For tasks like sentiment, word **occurrence** seems to be more important than word **frequency**.

- The occurrence of the word *fantastic* tells us a lot
- The fact that it occurs 5 times may not tell us much more.

**Binary multinominal naive bayes**, or **binary NB**

Clip our word counts at 1

Note: this is different than Bernoulli naive bayes; see the textbook at the end of the chapter.

# Binary Multinomial Naive Bayes on a test document *d*



- First remove all duplicate words from *d*
- Then compute NB using the same equation:

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

# Binary multinominal naive Bayes

**Four original documents:**

- – it was pathetic the worst part was the boxing scenes
- – no plot twists or great scenes
- + and satire and great plot twists
- + great scenes great film

# Binary multinominal naive Bayes

**Four original documents:**

- − it was pathetic the worst part was the boxing scenes
- − no plot twists or great scenes
- + and satire and great plot twists
- + great scenes great film

| | NB Counts + | NB Counts − |
|---|---|---|
| and | 2 | 0 |
| boxing | 0 | 1 |
| film | 1 | 0 |
| great | 3 | 1 |
| it | 0 | 1 |
| no | 0 | 1 |
| or | 0 | 1 |
| part | 0 | 1 |
| pathetic | 0 | 1 |
| plot | 1 | 1 |
| satire | 1 | 0 |
| scenes | 1 | 2 |
| the | 0 | 2 |
| twists | 1 | 1 |
| was | 0 | 2 |
| worst | 0 | 1 |

# Binary multinominal naive Bayes

**Four original documents:**

- − it was pathetic the worst part was the boxing scenes
- − no plot twists or great scenes
- + and satire and great plot twists
- + great scenes great film

**After per-document binarization:**

- − it was pathetic the worst part boxing scenes
- − no plot twists or great scenes
- + and satire great plot twists
- + great scenes film

| | NB Counts + | NB Counts − |
|---|---|---|
| and | 2 | 0 |
| boxing | 0 | 1 |
| film | 1 | 0 |
| great | 3 | 1 |
| it | 0 | 1 |
| no | 0 | 1 |
| or | 0 | 1 |
| part | 0 | 1 |
| pathetic | 0 | 1 |
| plot | 1 | 1 |
| satire | 1 | 0 |
| scenes | 1 | 2 |
| the | 0 | 2 |
| twists | 1 | 1 |
| was | 0 | 2 |
| worst | 0 | 1 |

# Binary multinominal naive Bayes

**Four original documents:**

– it was pathetic the worst part was the boxing scenes
– no plot twists or great scenes
+ and satire and great plot twists
+ great scenes great film

**After per-document binarization:**

– it was pathetic the worst part boxing scenes
– no plot twists or great scenes
+ and satire great plot twists
+ great scenes film

| | NB Counts + | NB Counts – | Binary Counts + | Binary Counts – |
|---|---|---|---|---|
| and | 2 | 0 | 1 | 0 |
| boxing | 0 | 1 | 0 | 1 |
| film | 1 | 0 | 1 | 0 |
| great | 3 | 1 | 2 | 1 |
| it | 0 | 1 | 0 | 1 |
| no | 0 | 1 | 0 | 1 |
| or | 0 | 1 | 0 | 1 |
| part | 0 | 1 | 0 | 1 |
| pathetic | 0 | 1 | 0 | 1 |
| plot | 1 | 1 | 1 | 1 |
| satire | 1 | 0 | 1 | 0 |
| scenes | 1 | 2 | 1 | 2 |
| the | 0 | 2 | 0 | 1 |
| twists | 1 | 1 | 1 | 1 |
| was | 0 | 2 | 0 | 1 |
| worst | 0 | 1 | 0 | 1 |

Counts can still be 2! Binarization is within-doc!

# Lecture Contents



- The Task of Text Classification
- The Naïve Bayes Text Classifier
- Naïve Bayes: Learning
- Sentiment and Binary Naïve Bayes
- Accuracy, Precision, Recall, and F measure

# Evaluation



- Let's consider just binary text classification tasks
- Imagine you're the CEO of Delicious Pie Company
- You want to know what people are saying about your pies
- So you build a "Delicious Pie" tweet detector
  - Positive class: tweets about Delicious Pie Co
  - Negative class: all other tweets

# The 2-by-2 confusion matrix



*gold standard labels*

| *system output labels* | | gold positive | gold negative |
|---|---|---|---|
| | system positive | **true positive** | **false positive** |
| | system negative | **false negative** | **true negative** |

$$\textbf{precision} = \frac{tp}{tp+fp}$$

$$\textbf{recall} = \frac{tp}{tp+fn}$$

$$\textbf{accuracy} = \frac{tp+tn}{tp+fp+tn+fn}$$

# Evaluation: Accuracy



- Why don't we use **accuracy** as our metric?
- Imagine we saw 1 million tweets
  - 100 of them talked about Delicious Pie Co.
  - 999,900 talked about something else
- We could build a dumb classifier that just labels every tweet "not about pie"
  - It would get 99.99% accuracy!!! Wow!!!!
  - But useless! Doesn't return the comments we are looking for!
  - That's why we use **precision** and **recall** instead

# Evaluation: Precision



% of items the system detected (i.e., items the system labeled as positive) that are in fact positive (according to the human gold labels)

$$\textbf{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

# Evaluation: Recall



% of items actually present in the input that were correctly identified by the system.

$$\textbf{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

# Why Precision and Recall



Our dumb pie-classifier

Just label nothing as "about pie"

Accuracy=99.99%

but

Recall = 0

(it doesn't get any of the 100 Pie tweets)

Precision and recall, unlike accuracy, emphasize true positives:

finding the things that we are supposed to be looking for.

# A combined measure: F



- F measure: a single number that combines P and R:

$$F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

- We almost always use balanced $F_1$ (i.e., $\beta = 1$)

$$\mathrm{F}_1 = \frac{2PR}{P + R}$$

# Why harmonic means?



- Classifier1: P:0.53, R:0.36
- Classifier2: P:0.01, R:0.99

| Harmonic | Average |
| --- | --- |
| 0.429 | 0.445 |
| 0.019 | 0.500 |

# Confusion Matrix for 3-class classification



*gold labels*

| *system output* | | urgent | normal | spam |
|---|---|---|---|---|
| | urgent | 8 | 10 | 1 |
| | normal | 5 | 60 | 50 |
| | spam | 3 | 30 | 200 |

$$\textbf{precision}_u = \frac{8}{8+10+1}$$

$$\textbf{precision}_n = \frac{60}{5+60+50}$$

$$\textbf{precision}_s = \frac{200}{3+30+200}$$

$$\textbf{recall}_u = \frac{8}{8+5+3} \qquad \textbf{recall}_n = \frac{60}{10+60+30} \qquad \textbf{recall}_s = \frac{200}{1+50+200}$$

# How to combine P/R from 3 classes to get one metric

Macroaveraging:

compute the performance for each class, and then average over classes

Microaveraging:

collect decisions for all classes into one confusion matrix

compute precision and recall from that table.

# Macroaveraging and Microaveraging



**Class 1: Urgent**

| | true urgent | true not |
|---|---|---|
| system urgent | 8 | 11 |
| system not | 8 | 340 |

$$\text{precision} = \frac{8}{8+11} = .42$$

**Class 2: Normal**

| | true normal | true not |
|---|---|---|
| system normal | 60 | 55 |
| system not | 40 | 212 |

$$\text{precision} = \frac{60}{60+55} = .52$$

**Class 3: Spam**

| | true spam | true not |
|---|---|---|
| system spam | 200 | 33 |
| system not | 51 | 83 |

$$\text{precision} = \frac{200}{200+33} = .86$$

**Pooled**

| | true yes | true no |
|---|---|---|
| system yes | 268 | 99 |
| system no | 99 | 635 |

$$\text{microaverage precision} = \frac{268}{268+99} = \mathbf{.73}$$

$$\text{macroaverage precision} = \frac{.42+.52+.86}{3} = \mathbf{.60}$$

# Development Test Sets and Cross-validation



| Training set | Development Test Set | Test Set |
| --- | --- | --- |

- Metric: P/R/F1 or Accuracy
- Unseen test set
  - avoid overfitting ("tuning to the test set")
  - more conservative estimate of performance
- Cross-validation over multiple splits
  - *k*-fold cross validation or multiple train/test splits

# K-fold cross validation



- Break up data into 10 folds
  - (Equal positive and negative inside each fold?)
- For each fold
  - Choose the fold as a temporary test set
  - Train on 9 folds, compute performance on the test fold
- Report average performance of the 10 runs