COS 484: Natural Language Processing

Text Classification

Fall 2019
Why classify?

- Authorship attribution
- Language detection
- News categorization

Spam detection

Sentiment analysis
Classification: The Task

• Inputs:
  • A document $d$
  • A set of classes $C = \{c_1, c_2, c_3, \ldots, c_m\}$

• Output:
  • Predicted class $c$ for document $d$

- **Movie was terrible** → Classify → Negative
- **Amazing acting** → Classify → Positive
Rule-based classification

- Combinations of features on words in document, meta-data

  IF there exists word $w$ in document $d$ such that $w$ in \{good, great, extra-ordinary, \ldots\},
  THEN output Positive

  IF email address ends in \{ithelpdesk.com, makemoney.com, spinthewheel.com, \ldots\}
  THEN output SPAM

- Can be very accurate

- Rules may be hard to define (and some even unknown to us!)

- Expensive

- Not easily generalizable
Supervised Learning: Let’s use statistics!

• Data-driven approach

• Let the machine figure out the best patterns to use

• Inputs:
  
  • Set of \( m \) classes \( C = \{c_1, c_2, \ldots, c_m\} \)

  • Set of \( n \) ‘labeled’ documents: \{\( (d_1, c_1) \), \( (d_2, c_2) \), \ldots, \( (d_n, c_n) \)\}

• Output:

  • Trained classifier, \( F : d \rightarrow c \)

KEY QUESTIONS:

a) Form of \( F \)?

b) How to learn \( F \)?
Types of supervised classifiers

- Naive Bayes
- Logistic regression
- Support vector machines
- k-nearest neighbors
Multinomial Naive Bayes

- Simple classification model making use of Bayes rule

  - Bayes Rule:

    \[
    P(c \mid d) = \frac{P(c) \cdot P(d \mid c)}{P(d)}
    \]

- Makes strong (naive) independence assumptions
Predicting a class

\[ c_{\text{MAP}} = \underset{c \in C}{\text{argmax}} \quad p(c|d) \]

\[ = \underset{c \in C}{\text{argmax}} \quad \frac{p(c) \cdot p(d|c)}{p(d)} \]

\[ = \underset{c \in C}{\text{argmax}} \quad p(c) \cdot p(d|c) \]

\( p(c) \rightarrow \text{prior probability of class } c \)

\( p(d|c) \rightarrow \text{conditional probability of generating document } d \text{ from class } c \)
How to represent $P(d \mid c)$?

- Option 1: represent the entire sequence of words
  
  - $P(w_1, w_2, w_3, \ldots, w_k \mid c)$ (too many sequences!)

- Option 2: Bag of words
  
  - Assume position of each word is irrelevant (both absolute and relative)
  
  - $P(w_1, w_2, w_3, \ldots, w_k \mid c) = P(w_1|c) P(w_2|c) \ldots P(w_k|c)$

  - Probability of each word is conditionally independent given class $c$
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!
Predicting with Naive Bayes

- We now have:

\[
C_{\text{MAP}} = \arg \max_c \quad P(d \mid c) \cdot P(c)
\]

\[
= \arg \max_c \quad P(w_1, w_2, \ldots, w_k \mid c) \cdot P(c)
\]

\[
= \arg \max_c \quad P(c) \cdot \prod_{i=1}^{k} P(w_i \mid c)
\]

(using **BOW assumption**
Naive Bayes as a generative classifier

\[ \text{\textcolor{red}{c = Science}} \]

\[ p(c) \]

\[ d_1, \ldots, d_n \]
Naive Bayes as a generative model
Naive Bayes as a generative model

\[ d_1 \]

- \( c = \text{Science} \)
- \( \Pr(c) \)
- \( P(w_1|c) \)
- \( P(w_2|c) \)
- \( P(w_3|c) \)

- \( w_1 = \text{Scientists} \)
- \( w_2 = \text{have} \)
- \( w_3 = \text{discovered} \)

\[ \ldots \]
Naive Bayes as a generative model

Generate the entire data set one document at a time
Estimating probabilities

- Maximum likelihood estimates:

\[ \hat{P}(c_j) = \frac{\text{count (class } = c_j)}{\sum_{c} \text{count (class } = c)} \]

\[ \hat{P}(w_i | c_j) = \frac{\text{count (} w_i,c_j \text{)}}{\sum_{w} \text{count (} w,c_j \text{)}} \]
Data sparsity

- What is \( \text{count('amazing', positive)} = 0? \)

- Implies \( P('amazing' \mid positive) = 0 \)

- Given a review document, \( d = "... most amazing movie ever ..." \)

\[
\hat{c}_{\text{MAP}} = \arg\max_c \hat{P}(c) \prod_{i=1}^{K} p(o_i \mid c) = \arg\max_c \hat{P}(c) \cdot 0
\]
Solution: Smoothing!

- Laplace smoothing:

\[
\hat{P}(w_i | c) = \frac{\text{count}(w_i, c) + \alpha}{\left[ \sum_{w} \text{count}(w, c) \right] + \alpha |V|}
\]

- Simple, easy to use
- Effective in practice
Overall process

- Input: Set of annotated documents \( \{(d_i, c_i)\}_{i=1}^{n} \)

A. Compute vocabulary \( V \) of all words

\[
\hat{p}(c_j) = \frac{\text{Count}(c_j)}{\text{Total # docs}}
\]

B. Calculate

\[
\hat{p}(w_i | c_j) = \frac{\text{Count}(w_i, c_j) + \alpha}{\sum_{w \in V} \left[ \text{Count}(w, c_j) + \alpha \right]}
\]

C. Calculate

D. (Prediction) Given document \( d = (w_1, w_2, \ldots, w_k) \)

\[
\hat{c}_{\text{MAP}} = \arg \max_{c} \hat{p}(c) \cdot \prod_{i=1}^{k} \hat{p}(w_i | c)
\]
Naive Bayes Example

\[ \hat{P}(c) = \frac{N_c}{N} \]
\[ \hat{P}(w \mid c) = \frac{\text{count}(w, c) + 1}{\text{count}(c) + |V|} \]

Priors:
\[ P(c) = \frac{3}{4} \]
\[ P(j) = \frac{1}{4} \]

Conditional Probabilities:
\[ P(\text{Chinese} \mid c) = \frac{5+1}{8+6} = \frac{6}{14} = \frac{3}{7} \]
\[ P(\text{Tokyo} \mid c) = \frac{0+1}{8+6} = \frac{1}{14} \]
\[ P(\text{Japan} \mid c) = \frac{0+1}{8+6} = \frac{1}{14} \]
\[ P(\text{Chinese} \mid j) = \frac{1+1}{3+6} = \frac{2}{9} \]
\[ P(\text{Tokyo} \mid j) = \frac{1+1}{3+6} = \frac{2}{9} \]
\[ P(\text{Japan} \mid j) = \frac{1+1}{3+6} = \frac{2}{9} \]

Choosing a class:
\[ P(c \mid d5) \propto \frac{3}{4} \times \left( \frac{3}{7} \right)^3 \times \frac{1}{14} \times \frac{1}{14} \approx 0.0003 \]
\[ P(j \mid d5) \propto \frac{1}{4} \times \left( \frac{2}{9} \right)^3 \times \frac{2}{9} \times \frac{2}{9} \approx 0.0001 \]
Features

- In general, Naive Bayes can use any set of features, not just words
  - URLs, email addresses, Capitalization, …
  - Domain knowledge crucial to performance

<table>
<thead>
<tr>
<th>Rank</th>
<th>Category</th>
<th>Feature</th>
<th>Rank</th>
<th>Category</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Subject</td>
<td>Number of capitalized words</td>
<td>1</td>
<td>Subject</td>
<td>Min of the compression ratio for the bz2 compressor</td>
</tr>
<tr>
<td>2</td>
<td>Subject</td>
<td>Sum of all the character lengths of words</td>
<td>2</td>
<td>Subject</td>
<td>Min of the compression ratio for the zlib compressor</td>
</tr>
<tr>
<td>3</td>
<td>Subject</td>
<td>Number of words containing letters and numbers</td>
<td>3</td>
<td>Subject</td>
<td>Min of character diversity of each word</td>
</tr>
<tr>
<td>4</td>
<td>Subject</td>
<td>Max of ratio of digit characters to all characters of each word</td>
<td>4</td>
<td>Subject</td>
<td>Min of the compression ratio for the lzw compressor</td>
</tr>
<tr>
<td>5</td>
<td>Header</td>
<td>Hour of day when email was sent</td>
<td>5</td>
<td>Subject</td>
<td>Max of the character lengths of words</td>
</tr>
</tbody>
</table>

Spam URLs Features

<table>
<thead>
<tr>
<th>Rank</th>
<th>Category</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>URL</td>
<td>The number of all URLs in an email</td>
</tr>
<tr>
<td>2</td>
<td>URL</td>
<td>The number of unique URLs in an email</td>
</tr>
<tr>
<td>3</td>
<td>Payload</td>
<td>Number of words containing letters and numbers</td>
</tr>
<tr>
<td>4</td>
<td>Payload</td>
<td>Min of the compression ratio for the bz2 compressor</td>
</tr>
<tr>
<td>5</td>
<td>Payload</td>
<td>Number of words containing only letters</td>
</tr>
<tr>
<td>1</td>
<td>Header</td>
<td>Day of week when email was sent</td>
</tr>
<tr>
<td>2</td>
<td>Payload</td>
<td>Number of characters</td>
</tr>
<tr>
<td>3</td>
<td>Payload</td>
<td>Sum of all the character lengths of words</td>
</tr>
<tr>
<td>4</td>
<td>Header</td>
<td>Minute of hour when email was sent</td>
</tr>
<tr>
<td>5</td>
<td>Header</td>
<td>Hour of day when email was sent</td>
</tr>
</tbody>
</table>
Naive Bayes and Language Models

• If features = bag of words, each class is a unigram language model!

• For class c, assigning each word: $P(\omega | c)$
  assigning sentence: $p(s | c) = \prod_{\omega \in s} P(\omega | c)$

<table>
<thead>
<tr>
<th>Class pos</th>
<th>I</th>
<th>love</th>
<th>this</th>
<th>fun</th>
<th>film</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>.05</td>
<td>0.01</td>
<td>0.1</td>
</tr>
</tbody>
</table>

$P(s \mid pos) = 0.0000005$
Naive Bayes as a language model

- Which class assigns the higher probability to $s$?

<table>
<thead>
<tr>
<th>Model pos</th>
<th>Model neg</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.2</td>
</tr>
<tr>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>0.01</td>
<td>0.001</td>
</tr>
<tr>
<td>0.05</td>
<td>0.01</td>
</tr>
<tr>
<td>0.1</td>
<td>0.005</td>
</tr>
<tr>
<td>0.1</td>
<td>0.1</td>
</tr>
</tbody>
</table>

$P(s|\text{pos}) > P(s|\text{neg})$

<table>
<thead>
<tr>
<th></th>
<th>love</th>
<th>this</th>
<th>fun</th>
<th>film</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.1</td>
<td>0.01</td>
<td>0.05</td>
<td>0.1</td>
</tr>
<tr>
<td>0.2</td>
<td>0.001</td>
<td>0.01</td>
<td>0.005</td>
<td>0.1</td>
</tr>
</tbody>
</table>
Evaluation

- Consider binary classification

- Table of predictions

<table>
<thead>
<tr>
<th>Predicted</th>
<th>Truth</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td></td>
<td>Positive</td>
<td>Negative</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>Negative</td>
<td>45</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>

- Ideally, we want:

<table>
<thead>
<tr>
<th></th>
<th>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>145</td>
<td>0</td>
</tr>
<tr>
<td>Negative</td>
<td>0</td>
<td>105</td>
</tr>
</tbody>
</table>
Evaluation Metrics

- True positive: Predicted + and actual +
- True negative: Predicted - and actual -
- False positive: Predicted + and actual -
- False negative: Predicted - and actual +

\[
\text{Accuracy} = \frac{TP + TN}{Total} = \frac{200}{250} = 80\% 
\]
# Evaluation Metrics

**Truth**

<table>
<thead>
<tr>
<th>Predicted</th>
<th>True Positive</th>
<th>True Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>100</td>
<td>5</td>
</tr>
<tr>
<td>Negative</td>
<td>45</td>
<td>100</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Predicted</th>
<th>False Positive</th>
<th>False Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>100</td>
<td>25</td>
</tr>
<tr>
<td>Negative</td>
<td>25</td>
<td>100</td>
</tr>
</tbody>
</table>

- True positive: Predicted + and actual +
- True negative: Predicted - and actual -
- False positive: Predicted + and actual -
- False negative: Predicted - and actual +

**Accuracy**

\[
\text{Accuracy} = \frac{TP + TN}{Total} = \frac{200}{250} = 80\% 
\]
Precision and Recall

- Precision: % of selected classes that are correct

\[
\text{Precision}(+) = \frac{TP}{TP + FP} \quad \text{Precision}(-) = \frac{TN}{TN + FN}
\]

- Recall: % of correct items selected

\[
\text{Recall}(+) = \frac{TP}{TP + FN} \quad \text{Recall}(-) = \frac{TN}{TN + FP}
\]
F-Score

• Combined measure

• Harmonic mean of Precision and Recall

\[ F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \]

• Or more generally,

\[ F_\beta = \frac{(1 + \beta^2) \cdot \text{Precision} \cdot \text{Recall}}{\beta^2 \cdot \text{Precision} + \text{Recall}} \]
Choosing Beta

\[ F_\beta = \frac{(1 + \beta^2) \cdot \text{Precision} \cdot \text{Recall}}{\beta^2 \cdot \text{Precision} + \text{Recall}} \]

<table>
<thead>
<tr>
<th>Truth</th>
<th>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>200</td>
<td>100</td>
</tr>
<tr>
<td>Negative</td>
<td>50</td>
<td>100</td>
</tr>
</tbody>
</table>

• Which value of Beta maximizes \( F_\beta \) for positive class?

A. \( \beta = 0.5 \)

B. \( \beta = 1 \)

C. \( \beta = 2 \)
Aggregating scores

• We have Precision, Recall, F1 for each class

• How to combine them for an overall score?
  
  • Macro-average: Compute for each class, then average

  • Micro-average: Collect predictions for all classes and jointly evaluate
Macro vs Micro average

<table>
<thead>
<tr>
<th>Class 1</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Truth: yes</td>
<td>Truth: no</td>
</tr>
<tr>
<td>Classifier: yes</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Classifier: no</td>
<td>10</td>
<td>970</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class 2</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Truth: yes</td>
<td>Truth: no</td>
</tr>
<tr>
<td>Classifier: yes</td>
<td>90</td>
<td>10</td>
</tr>
<tr>
<td>Classifier: no</td>
<td>10</td>
<td>890</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Micro Ave. Table</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Truth: yes</td>
<td>Truth: no</td>
</tr>
<tr>
<td>Classifier: yes</td>
<td>100</td>
<td>20</td>
</tr>
<tr>
<td>Classifier: no</td>
<td>20</td>
<td>1860</td>
</tr>
</tbody>
</table>

- Macroaveraged precision: \((0.5 + 0.9)/2 = 0.7\)
- Microaveraged precision: \(100/120 = 0.83\)
- Microaveraged score is dominated by score on common classes
• Choose a metric: Precision/Recall/F1

• Optimize for metric on Validation (aka Development) set

• Finally evaluate on ‘unseen’ test set

• Cross-validation:
  • Repeatedly sample several train-val splits
  • Reduces bias due to sampling errors
Advantages of Naive Bayes

- 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 we will see other classifiers that give better accuracy
Practical Naive Bayes

- Small data sizes:
  - Naive Bayes is great! (high bias)
  - Rule-based classifiers might work well too

- Medium size datasets:
  - More advanced classifiers might perform better (e.g. SVM, logistic regression)

- Large datasets:
  - Naive Bayes becomes competitive again (although most classifiers work well)
Failings of Naive Bayes (1)

• Independence assumptions are too strong

<table>
<thead>
<tr>
<th>x1</th>
<th>x2</th>
<th>Class: ( x_1, \text{XOR} x_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

• XOR problem: Naive Bayes cannot learn a decision boundary

• Both variables are jointly required to predict class
Failings of Naive Bayes (2)

- Class imbalance:
  - One or more classes have more instances than others
  - Data skew causes NB to prefer one class over the other
- Solution: Complement Naive Bayes (Rennie et al., 2003)

\[
\hat{p}(w_i | \neg c_j) = \frac{\sum_{c \neq c_j} \text{Count}(w_i, c)}{\sum_{c \neq c_j} \sum_{w} \text{Count}(w, c)}
\]
Failings of Naive Bayes (3)

- Weight magnitude errors:
  - Classes with larger weights are preferred
  - 10 documents with class=MA and “Boston” occurring once each
  - 10 documents with class=CA and “San Francisco” occurring once each
  - New document: “Boston Boston Boston San Francisco San Francisco”

\[
P(\text{class} = \text{CA} \mid \text{document}) > P(\text{class} = \text{MA} \mid \text{document})
\]
Practical text classification

- Domain knowledge is crucial to selecting good features
- Handle class imbalance by re-weighting classes
- Use log scale operations instead of multiplying probabilities
  - Since \( \log(xy) = \log(x) + \log(y) \)
    - Better to sum logs of probabilities instead of multiplying probabilities.
  - Class with highest un-normalized log probability score is still most probable.

\[
c_{NB} = \underset{c_j \in C}{\arg \max} \log P(c_j) + \sum_{i \in \text{positions}} \log P(x_i | c_j)
\]

- Model is now just max of sum of weights