Categorization

Important task for both humans and machines

- object identification
- face recognition
- spoken word recognition

Text categorization

- Spam detection (spam/not-spam)
- Sentiment analysis (pos/neg attitude, 1-5 star review)
- Author attribution (male/female, specific author, healthy/mental illness)
- Topic classification (sport/finance/travel/etc)

Formalizing the task

Given document \( d \) and set of categories \( C \), we want to assign it to category

\[
\hat{c} = \arg\max_{c \in C} P(c|d) = \arg\max_{c \in C} P(d|c)P(c)
\]
How to approximate $P(d|c)$?

- Each document $d$ is represented by features $f_1, f_2, \ldots, f_n$ (words in doc).
- In fact, we only care about the feature counts:

<table>
<thead>
<tr>
<th>doc 1</th>
<th>the</th>
<th>your</th>
<th>model</th>
<th>cash</th>
<th>Viagra</th>
<th>class</th>
<th>account</th>
<th>orderz</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>25</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>17</td>
<td>5</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- A model where we care only about (unigram) word counts is called a bag-of-words model.

How to approximate $P(d|c)$?

- Represent $d$ using its feature values:

$$P(d|c) = P(f_1, f_2, \ldots, f_n|c)$$

- Naive Bayes assumption: features are conditionally independent given class.

$$P(d|c) \approx P(f_1|c)P(f_2|c)\ldots P(f_n|c)$$

Full model

- Given document with features $f_1, f_2, \ldots, f_n$ and set of categories $C$, choose

$$\hat{c} = \arg\max_{c \in C} P(c) \prod_{i=1}^{n} P(f_i|c)$$

- This is called a Naive Bayes classifier (classification = categorization)

Learning the class priors

- $P(c)$ normally estimated with MLE:

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

- $N_c = \text{the number of training documents in class } c$
- $N = \text{the total number of training documents}$
Learning the class priors: example

- Given training documents with correct labels:

<table>
<thead>
<tr>
<th></th>
<th>the</th>
<th>your</th>
<th>model</th>
<th>cash</th>
<th>Viagra</th>
<th>class</th>
<th>account</th>
<th>orderz</th>
<th>spam?</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc 1</td>
<td>12</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>doc 2</td>
<td>10</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>+</td>
</tr>
<tr>
<td>doc 3</td>
<td>25</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>doc 4</td>
<td>14</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>+</td>
</tr>
<tr>
<td>doc 5</td>
<td>17</td>
<td>5</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>+</td>
</tr>
</tbody>
</table>

- $\hat{P}(\text{spam}) = \frac{3}{5}$

Sharon Goldwater  ANLP Lecture 10  6 October 2014

Learning the feature probabilities

- $P(f_i|c)$ normally estimated with simple smoothing:

$$\hat{P}(f_i|c) = \frac{\text{count}(f_i, c) + \alpha}{\sum_{f \in F} (\text{count}(f_i, c) + \alpha)}$$

- $\text{count}(f_i, c)$ = the number of times $f_i$ occurs in class $c$
- $F$ = the set of possible features

Sharon Goldwater  ANLP Lecture 10  6 October 2014

Classifying a test document: example

- Test document $d$: get your cash and your orderz

$$P(+|d) \propto \hat{P}(f_i|+) \prod_{i=1}^{n} P(f_i|+)$$

$$= P(+) \cdot \frac{\alpha}{(70 + \alpha F)} \cdot \frac{11 + \alpha}{(70 + \alpha F)} \cdot \frac{7 + \alpha}{(70 + \alpha F)}$$

$$= \frac{\alpha}{(70 + \alpha F)} \cdot \frac{11 + \alpha}{(70 + \alpha F)} \cdot \frac{2 + \alpha}{(70 + \alpha F)}$$

Sharon Goldwater  ANLP Lecture 10  6 October 2014
Classifying a test document: example

- Test document $d$:
  
  get your cash and your orderz

- Do the same for $P(\neg|d)$

- Choose the one with the larger value

---

Alternative feature values and feature sets

- Use only **binary** values for $f_i$: did this word occur in $d$ or not?

- Use only a subset of the vocabulary for $F$
  
  - Ignore **stopwords** (function words and others with little content)
  
  - Choose a small task-relevant set (e.g., using a sentiment lexicon)

- Use more complex features (bigrams, syntactic features, morphological features, ...)

---

Advantages of Naive Bayes

- Very easy to implement

- Very fast to train and test

- Doesn’t require as much training data and some other methods

- Usually works reasonably well

- This should be your baseline method for any classification task

---

Problems with Naive Bayes

- Naive Bayes assumption is naive!

- Consider categories **Travel**, **Finance**, **Sport**.

- Are the following features independent given the category?

  beach, sun, ski, snow, pitch, palm, football, relax, ocean
Problems with Naive Bayes

- Naive Bayes assumption is naive!
- Consider categories **Travel**, **Finance**, **Sport**.
- Are the following features independent given the category?
  
  beach, sun, ski, snow, pitch, palm, football, relax, ocean
- No! They might be closer if we defined finer-grained categories (beach vacations vs. ski vacations), but we don’t usually want to.

Non-independent features

- Features are not usually independent given the class
- Adding multiple feature types (e.g., words and morphemes) often leads to even stronger correlations between features
- Accuracy of classifier can sometimes still be ok, but it will be highly overconfident in its decisions.
  
  - Ex: NB sees 5 features that all point to class 1, treats them as five independent sources of evidence.
  - Like asking 5 friends for an opinion when some got theirs from each other.

How to evaluate performance?

- Important question for any NLP task
- **Intrinsic** evaluation: design a measure inherent to the task
  
  - Language modeling: perplexity
  - POS tagging: accuracy (% of tags correct)
  - Categorization: F-score (coming up next)

How to evaluate performance?

- Important question for any NLP task
- **Intrinsic** evaluation: design a measure inherent to the task
  
  - Language modeling: perplexity
  - POS tagging: accuracy (% of tags correct)
  - Categorization: F-score (coming up next)

- **Extrinsic** evaluation: measure effects on a downstream task
  
  - Language modeling: does it improve my ASR/MT system?
  - POS tagging: does it improve my parser/IR system?
  - Categorization: does it reduce user search time in an IR setting?
Intrinsic evaluation for categorization

- Categorization as detection: document about sport or not?
- Classes may be very unbalanced.
- Can get 95% accuracy by always choosing “not”; but this isn’t useful.
- Need a better measure.

Two measures

- Assume we have a **gold standard**: correct labels for test set
- We also have a system for detecting the items of interest (docs about sport)

\[
\text{Precision} = \frac{\text{# items system got right}}{\text{# items system detected}}
\]

\[
\text{Recall} = \frac{\text{# items system got right}}{\text{# items system should have detected}}
\]

Example of precision and recall

<table>
<thead>
<tr>
<th>Gold standard</th>
<th>Item in class?</th>
<th>+</th>
<th>+</th>
<th>-</th>
<th>-</th>
<th>+</th>
<th>-</th>
<th>-</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td>System output</td>
<td></td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

- # correct = 1
- # detected = 2
- # in GS = 3

Precision-Recall curves

- If system has a tunable parameter to vary the precision/recall:

Figure from: [http://ivrgwww.epfl.ch/supplementary_material/RK_CVPR09/](http://ivrgwww.epfl.ch/supplementary_material/RK_CVPR09/)
F-measure

- Can also combine precision and recall into a single F-measure:
  \[ F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R} \]

- Normally we just set \( \beta = 1 \) to get \( F_1 \):
  \[ F_1 = \frac{2PR}{P + R} \]

- \( F_1 \) is the harmonic mean of \( P \) and \( R \): similar to arithmetic mean when \( P \) and \( R \) are close, but penalizes large differences between \( P \) and \( R \).