Project

- Shared task: ‘chunk parsing’
- Description

Probabilistic models

- Suppose we have a representation of an instance as feature vector $x$ and we want to predict its class $c$
  
  - If we have a way of modeling $P(c|x)$, *Bayes Decision Rule* says our predicted $\hat{c}$ should be:
    
    $$\hat{c} = \arg\max_{c \in \mathcal{C}} P(c|x)$$
  
  - This minimizes the expected error:
    
    $$P(error|x) = 1 - P(\hat{c}|x)$$
    $$P(error) = \sum_x P(error|x) P(x)$$

- There are two ways of applying the Bayes decision rule
  
  - A *discriminative* (aka *diagnostic*) method directly models $P(c|x)$
  
  - More commonly, a *generative* (aka *sampling*) method is used, which models the joint distribution $P(x,c)$ and uses Bayes rule:
    
    $$\hat{c} = \arg\max_{c \in \mathcal{C}} P(c|x)$$
    $$= \arg\max_{c \in \mathcal{C}} \frac{P(x|c) P(c)}{P(x)}$$
    $$= \arg\max_{c \in \mathcal{C}} P(x|c) P(c)$$
    $$= \arg\max_{c \in \mathcal{C}} P(x,c)$$

Baseline classifier

- We often compute a ‘baseline’ for a classification task by simply assigning the most frequent class to each instance:
  
  $$\hat{c} = \arg\max_{c \in \mathcal{C}} P(c)$$

- Here we assume that $P(c|x) = P(c)$, i.e., $X$ and $C$ are independent

- The extra error a baseline classifier makes is:
  
  $$\sum_x P(x) [P(x,c) - P(x) P(c)]$$
Bayes Optimal Classifiers

- Call a particular model $h$, chosen from the hypothesis space $H$.

- The maximum likelihood hypothesis selects:
  \[ \hat{c} = \arg\max_{c \in C, h \in H} P(c|x, h) P(d|h) \]

- The maximum a posteriori hypothesis selects:
  \[ \hat{c} = \arg\max_{c \in C, h \in H} P(c|x, h) P(d|h) P(h) \]

- Both of these commit us to choosing one $h$, which may or may not wind up being the best choice

Bayes Optimal Classifiers

- The Bayes Optimal Classifier selects:
  \[ \hat{c} = \arg\max_{c \in C} \sum_{h \in H} P(c|x, h) P(d|h) P(h) \]

- We remove the dependence on a particular $h$ by averaging over all possible $h$s

- This is almost always impossible to apply in practice, but it can be used to establish a lower bound on the error rate

- We can also sometimes approximate it, e.g., by randomly drawing $h$ from the posterior distribution $P(d|h) P(h)$
Naive Bayes classifiers

- To apply a generative Bayesian classifier, we need $P(x, c)$
- We can break this down into two parts: the class prior $P(c)$, and a likelihood $P(x|c)$
- The class priors are easy to estimate from training data:
  \[ \hat{P}(c) = \frac{\text{# of instances in class } c}{\text{# of instances}} \]
- This won’t work for $P(x|c)$, since any particular feature vector $x$ is unlikely to turn up in the training data:
  \[ \hat{P}(x|c) = \frac{\text{# of instances of } x \text{ in } c}{\text{# of instances in } c} \approx 0 \]
  \[ \text{# of instances of } x \text{ in } c \]
- Both $\hat{P}(c)$ and $\hat{P}(x|c)$ can be estimated using whatever tricks we have available

Naive Bayes classifiers

- The naive Bayes classifier selects the class $\hat{c}$ such that:
  \[ \hat{c} = \arg\max_{c \in C} P(c) \prod_i P(x_i|c) \]
- Naive Bayes classifiers have been used primarily for classifying texts (Maron 1961)
- We treat a text as a set or bag of words, an unordered collection of all the words that appear in the text
- “We treat a text as a set or bag of words” \(\equiv\) \{ a, a, as, bag, of, or, set, text, treat, we, words \}
- Ignoring word order in the feature representation removes the most obvious syntactic dependencies between words
  \[ P(\text{the}) P(\text{book}) \neq P(\text{the book}) \]
- There are still semantic dependencies:
  \[ P(\text{tackle}) P(\text{touchdown}) \neq P(\text{tackle, touchdown}) \]
- And, multiple occurrences of words are probably not independent
Text classification

- Text classification can be useful for information retrieval and natural language processing tasks
  - indexing
  - message routing
  - summarization

- Text classification also plays a role in linguistic research
  - authorship identification
  - genre studies
  - forensic linguistics
  - sociolinguistics

- A combination of the two makes the WWW available as a resource for research

Feature selection

- A straight bag-of-words model leads to positing a very large number of features
- Some of those features will not be relevant for the task (stop words)
- Many of the features will appear relevant, but won’t be: we can’t avoid the Curse of Dimensionality
- So, we want to select a subset of features which appear promising, usually by mutual information information gain

Multivariate Bernoulli event model

- If we represent a document as a set of words, then each feature $x_i$ is a Bernoulli variable, where:
  \[
P(x_i|c_j) = P(x_i = 1|c_j) (1 - P(x_i = 1|c_j))^{1-x_i}
  \]

- If there are $v$ words in the vocabulary, a document is constructed by flipping $v$ coins

- Call $p_{ij} = P(x_i = 1|c_j)$. Substituting this in, we get:
  \[
P(c_j|x) = \frac{P(c_j) \prod_i P(x_i|c_j)}{P(x)} 
  \]
  \[
  = \frac{P(c_j) \prod_i p_{ij}^{x_i} (1-p_{ij})^{1-x_i}}{P(x)}
  \]

- And taking the log gives us:
  \[
  \log P(c_j|x) = \log P(c_j) + \sum_i x_i \log p_{ij} + \sum_i (1-x_i) \log (1-p_{ij}) - \log P(x)
  \]
  \[
  = \log P(c_j) + \sum_i x_i \log p_{ij} + \sum_i \log (1-p_{ij}) - \sum_i x_i \log (1-p_{ij}) - \log P(x)
  \]
  \[
  = \log P(c_j) + \sum_i x_i \log \frac{p_{ij}}{1-p_{ij}} + \sum_i \log (1-p_{ij}) - \log P(x)
  \]

- Suppose we only have two classes. Then $P(c_1|x) = 1 - P(c_2|x)$, and the posterior log odds are:
  \[
  \log \frac{P(c_1|x)}{1-P(c_1|x)} = \sum_i x_i \log \frac{p_{1i}(1-p_{2i})}{(1-p_{1i})p_{2i}} + \sum_i \log \frac{1-p_{1i}}{1-p_{2i}} + \log \frac{P(c_1)}{1-P(c_1)}
  \]
Multivariate Bernoulli event model

- Under this binary independence model, the parameters $p_{ij}$ can be estimated via:
  \[ \hat{p}_{ij} = \frac{\# \text{ of documents containing } x_i \text{ in } c_j}{\# \text{ of documents in class } c_j} \]

- Note that this doesn't take into account the length of the document
- It also doesn't take into account the number of times a word appears in a document

Multinomial event model

- If instead we represent a document as a bag of words, then we can model a document as a sequence of random draws from a multinomial distribution

  - The probability of picking word $w_i$ if the document class is $c_j$ once is $P(w_i|c_j)$
  - The probability of picking word $w_i$ $x_i$ times in a row is $P(w_i|c_j)^{x_i}$
  - The probability of drawing a collection of words in that order is:
    \[ \prod_i P(w_i|c_i)^{x_i} \]

  - This underestimates $P(x|c_j)$, since lots of ordered sequences correspond to the same bag of words

  - How many different ways are there to draw word $w_1$ $x_1$ times, word $w_2$ $x_2$, and so on?

  - We can use the multinomial coefficient:
    \[ \binom{n}{n_1, n_2, \ldots} = \frac{n!}{n_1!(n-n_1)!n_2!(n-n_1-n_2)! \cdots} \]

  - To be completely correct, we also need to think about the probability of finding a document of a particular length:
    \[ P(x|c_j) = P(N|c_j)(\sum_i x_i)! \prod_i \frac{P(w_i|c_j)^{x_i}}{x_i!} \]
    but in practice this can be hard to do.

Multinomial event model

- So, if we draw $N = \sum_i x_i$ words, we have:
  \[ P(x|c_j) = \binom{N}{x_1, x_2, \ldots} \prod_i P(w_i|c_j)^{x_i} \]
  \[ = N! \prod_i \frac{P(w_i|c_j)^{x_i}}{x_i!} \]
**Multinomial event model**

- The parameters of the multinomial model are the individual word probabilities $P(w_i|c_j)$

- Since these are the parameters of a multinomial distribution, we need to maintain:
  $$\sum_i P(w_i|c_j) = 1$$

- We can estimate those from training data as:
  $$\hat{P}(w_i|c_j) = \frac{\text{# of times } w_i \text{ occurs in documents in } c_j}{\text{# of words in documents in class } c_j}$$

- As always, smoothing is important

**Text classification**

- The multinomial model takes word frequencies and document length into account, but treats multiple occurrences of a word as independent events

- McCallum and Nigam (1998) compare the two event models

- Multinominal model almost always outperforms multivariate Bernoulli model, by 25% or so

- The multinominal model handles large vocabulary sizes much better

- It’s easier to see how to add non-text features and to account for limited inter-dependencies using a multivariate Bernoulli model

**Naive Bayes classifiers**

- Despite its obvious limitations, naive Bayes text classifiers work quite well

- Lewis and Ringuette 1994 ‘breakeven point’ for naive Bayes very close to decision trees

- In other work, naive Bayes scores close to, but consistently worse than, more sophisticated methods

- Since naive Bayes is pretty good, and it’s easy to implement, it is very widely used

**Naive Bayes classifiers**

- Paul Graham wrote an article on naive Bayes classifiers for filtering junk mail, which has become a standard method

Free CableTV! No more pay! % RND_SYB

requisite silt administer orphanage teach hypothalamus diatomic conflict atlas moser cofactor electret coffin diversionary solicitous becalm absent satiable blurb mackerel sibilant tehran delivery germicidal barometer falmouth capricorn
Naive Bayes classifiers

- Maron (1961):

  It is feasible to have a computing machine read a document and to decide automatically the subject category to which the item in question belongs. No real intellectual breakthroughs are required before a machine will be able to index rather well. Just as in the case of machine translation of natural language, the road is gradual but, by and large, straightforward.