Categorization/Classification

- Given:
  - A description of an instance, \( x \in X \), where \( X \) is the instance language or instance space.
  - Issue: how to represent text documents.
  - A fixed set of categories:
    \[ C = \{ c_1, c_2, ..., c_n \} \]
- Determine:
  - The category of \( x \): \( c(x) \in C \), where \( c(x) \) is a categorization function whose domain is \( X \) and whose range is \( C \).
  - We want to know how to build categorization functions (“classifiers”).

Text Categorization Examples

Assign labels to each document or web-page:
- Labels are most often topics such as Yahoo-categories
  e.g., “finance,” “sports,” “news-world-asia-business”
- Labels may be genres
  e.g., “editorials,” “movie-reviews,” “news”
- Labels may be opinion
  e.g., “like,” “hate,” “neutral”
- Labels may be domain-specific binary
  e.g., “interesting-to-me” “not-interesting-to-me”
  e.g., “spam” “not-spam”
  e.g., “is a toner cartridge ad” “isn’t”

Document Classification

<table>
<thead>
<tr>
<th>Class</th>
<th>Testing Data</th>
<th>Training Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>ML</td>
<td>(AI)</td>
<td></td>
</tr>
<tr>
<td>Texting</td>
<td>Programming</td>
<td>Garb. Coll.</td>
</tr>
<tr>
<td>Semantics</td>
<td>(HCI)</td>
<td></td>
</tr>
<tr>
<td>ML</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Texting</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Semantics</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Garb. Coll.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multimedia</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GUI</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(Note: in real life there is often a hierarchy, not present in the above problem statement; and you get papers on ML approaches to Garb. Coll.)

Methods (1)

- Manual classification
  - Used by Yahoo!, Looksmart, about.com, ODP, Medline
  - very accurate when job is done by experts
  - consistent when the problem size and team is small
  - difficult and expensive to scale

- Automatic document classification
  - Hand-coded rule-based systems
    - Used by CS dept’s spam filter, Reuters, CIA, Verity, ...
    - E.g., assign category if document contains a given boolean combination of words
  - Commercial systems have complex query languages (everything in IR query languages + accumulators)
Methods (2)

- Accuracy is often very high if a query has been carefully refined over time by a subject expert
- Building and maintaining these queries is expensive
- Supervised learning of document-label assignment function
  - Many new systems rely on machine learning (Autonomy, Kana, MSN, Verity, ...)
  - k-Nearest Neighbors (simple, powerful)
  - Naive Bayes (simple, common method)
  - Support-vector machines (new, more powerful)
  - ... plus many other methods
  - No free lunch: requires hand-classified training data
  - But can be built (and refined) by amateurs

Text Categorization: attributes

- Representations of text are very high dimensional (one feature for each word).
- High-bias algorithms that prevent overfitting in high-dimensional space are best.
- For most text categorization tasks, there are many irrelevant and many relevant features.
- Methods that combine evidence from many or all features (e.g. naive Bayes, kNN, neural-nets) tend to work better than ones that try to isolate just a few relevant features (standard decision-tree or rule induction)*

*Although one can compensate by using many rules

Bayesian Methods

- Our focus today
- Learning and classification methods based on probability theory.
- Bayes theorem plays a critical role in probabilistic learning and classification.
- Build a generative model that approximates how data is produced
- Uses prior probability of each category given no information about an item.
- Categorization produces a posterior probability distribution over the possible categories given a description of an item.

Bayes’ Rule

\[ P(C, X) = P(C \mid X)P(X) = P(X \mid C)P(C) \]

\[ P(C \mid X) = \frac{P(X \mid C)P(C)}{P(X)} \]

Maximum a posteriori Hypothesis

\[ h_{MAP} = \arg \max_{h \in H} P(h \mid D) \]

\[ h_{MAP} = \arg \max_{h \in H} \frac{P(D \mid h)P(h)}{P(D)} \]

\[ h_{MAP} = \arg \max_{h \in H} P(D \mid h)P(h) \]

Maximum likelihood Hypothesis

If all hypotheses are a priori equally likely, we only need to consider the \( P(D \mid h) \) term:

\[ h_{ML} \equiv \arg \max_{h \in H} P(D \mid h) \]
Naive Bayes Classifiers

Task: Classify a new instance based on a tuple of attribute values $(x_1, x_2, \ldots, x_n)$

\[ c_{MAP} = \arg\max_{c \in C} P(c | x_1, x_2, \ldots, x_n) \]

\[ c_{MAP} = \arg\max_{c \in C} P(x_1, x_2, \ldots, x_n | c)P(c) \]

\[ c_{MAP} = \arg\max_{c \in C} P(x_1, x_2, \ldots, x_n | c) \]

Naive Bayes Classifier: Assumptions

- $P(C)$
  - Can be estimated from the frequency of classes in the training examples.
- $P(x_1, x_2, \ldots, x_n | c_j)$
  - $O(|X| \times |C|)$
  - Could only be estimated if a very, very large number of training examples was available.

Conditional Independence Assumption:

\[ \Rightarrow \text{Assume that the probability of observing the conjunction of attributes is equal to the product of the individual probabilities.} \]

The Naïve Bayes Classifier

- Conditional Independence Assumption:
  - features are independent of each other given the class:
  \[ P(X_1, \ldots, X_n | C) = P(X_1 | C) \cdot P(X_2 | C) \cdot \ldots \cdot P(X_n | C) \]

Learning the Model

- Common practice: maximum likelihood
  - simply use the frequencies in the data
  \[ \hat{P}(c_j) = \frac{N(C = c_j)}{N} \]
  \[ \hat{P}(x_i | c_j) = \frac{N(X_i = x_i, C = c_j)}{N(C = c_j)} \]

Problem with Max Likelihood

- What if we have seen no training cases where patient had no flu and muscle aches?
  \[ \hat{P}(X_i = t | C = nf) = \frac{N(X_i = t, C = nf)}{N(C = nf)} = 0 \]

- Zero probabilities cannot be conditioned away, no matter the other evidence!
  \[ \ell = \arg\max_{c \in C} \hat{P}(c) \prod \hat{P}(x_i | c) \]

Smoothing to Avoid Overfitting

- Somewhat more subtle version
  \[ \hat{P}(x_i | c_j) = \frac{N(X_i = x_i, C = c_j) + m_{i,j}}{N(C = c_j) + m} \]
  \[ \text{extreme of ”smoothing”} \]

- overall fraction in data where $X = x_{i,k}$
Using Naive Bayes Classifiers to Classify Text: Basic method

- Attributes are text positions, values are words.
  \[ c_{NB} = \arg\max_{c \in C} P(c) \prod_{j} P(x_j | c) \]

- Naive Bayes assumption is clearly violated.
- Example?
- Still too many possibilities
- Assume that classification is independent of the positions of the words
- Use same parameters for each position

Text Classification Algorithms: Learning

- From training corpus, extract Vocabulary
- Calculate required \( P(c) \) and \( P(x_j | c) \) terms
  - For each \( c \in C \) do
    - \( docs_c \) - subset of documents for which the target class is \( c \)
    - \( P(c) = \frac{|docs_c|}{\text{total # documents}} \)
  - \( Text_c \) - single document containing all \( docs_c \)
  - for each word \( x_j \) in Vocabulary
    - \( n_{j,c} \) - number of occurrences of \( x_j \) in \( Text_c \)
    - \( P(x_j | c) = \frac{n_{j,c} + 1}{\sum_{c \in C} n_{j,c} + |Vocabulary|} \)

Text Classification Algorithms: Classifying

- \( \text{positions} \leftarrow \text{all word positions in current document which contain tokens found in Vocabulary} \)
- Return \( c_{NB} \) where
  \[ c_{NB} = \arg\max_{c \in C} P(c) \prod_{i \in \text{positions}} P(x_i | c) \]

Naive Bayes Time Complexity

- Training Time: \( O(|D|L_{avg} + |C||V|) \)  where \( L_{avg} \) is the average length of a document in \( D \).
  - Assumes \( V \) and all \( D \), \( n_c \) and \( n_{j,c} \) pre-computed in \( O(|D|L_{avg}) \) time during one pass through all of the data.
  - Generally just \( O(|D|L_{avg}) \) since usually \( |C||V| < |D|L_{avg} \)
- Test Time: \( O(C L_{avg}) \)  where \( L_{avg} \) is the average length of a test document.
  - Very efficient overall, linearly proportional to the time needed to just read in all the data.

Underflow Prevention

- Multiplying lots of probabilities, which are between 0 and 1 by definition, can result in floating-point underflow.
- Since \( \log(xy) = \log(x) + \log(y) \), it is better to perform all computations by summing logs of probabilities rather than multiplying probabilities.
- Class with highest final un-normalized log probability score is still the most probable.

Naïve Bayes Posterior Probabilities

- Classification results of naïve Bayes (the class with maximum posterior probability) are usually fairly accurate.
- However, due to the inadequacy of the conditional independence assumption, the actual posterior-probability numerical estimates are not.
  - Output probabilities are generally very close to 0 or 1.
Two Models

- Model 1: Multivariate binomial
  - One feature $X_w$ for each word in dictionary
  - $X_w$ = true in document if $w$ appears in $d$
  - Naive Bayes assumption:
    - Given the document’s topic, appearance of one word in document tells us nothing about chances that another word appears

Two Models

- Model 2: Multinomial
  - One feature $X_i$ for each word pos in document
    - feature’s values are all words in dictionary
  - Value of $X_i$ is the word in position $i$
  - Naive Bayes assumption:
    - Given the document’s topic, word in one position in document tells us nothing about value of words in other positions
  - Second assumption:
    - word appearance does not depend on position
    $P(X_i = w | c) = P(X_j = w | c)$
    for all positions $i,j$, word $w$, and class $c$

Parameter estimation

- Binomial model:
  $\hat{P}(X_w = t | c_j) = \text{fraction of documents of topic } c_j \text{ in which word } w \text{ appears}$

- Multinomial model:
  $\hat{P}(X_i = w | c_j) = \text{fraction of times in which word } w \text{ appears across all documents of topic } c_j$
  - creating a mega-document for topic $c_j$ by concatenating all documents in this topic
  - use frequency of $w$ in mega-document

Feature selection via Mutual Information

- We might not want to use all words, but just reliable, good discriminators
- In training set, choose $k$ words which best discriminate the categories.
- One way is in terms of Mutual Information:
  $I(w, c) = \sum_{e_u \in (0,1)} \sum_{e_v \in (0,1)} p(e_u, e_v) \log \frac{p(e_u, e_v)}{p(e_u)p(e_v)}$
  - For each word $w$ and each category $c$

Feature selection via MI (contd.)

- For each category we build a list of $k$ most discriminating terms.
- For example (on 20 Newsgroups):
  - sci.electronics: circuit, voltage, amp, ground, copy, battery, electronics, cooling, ...
  - rec.autos: car, cars, engine, ford, dealer, mustang, oil, collision, autos, tires, toyota, ...
  - Greedy: does not account for correlations between terms
  - In general feature selection is necessary for binomial NB, but not for multinomial NB

Evaluating Categorization

- Evaluation must be done on test data that are independent of the training data (usually a disjoint set of instances).
- Classification accuracy: $c/n$ where $n$ is the total number of test instances and $c$ is the number of test instances correctly classified by the system.
- Results can vary based on sampling error due to different training and test sets.
- Average results over multiple training and test sets (splits of the overall data) for the best results.
Example: AutoYahoo!
- Classify 13,589 Yahoo! webpages in “Science” subtree into 95 different topics (hierarchy depth 2)

Example: WebKB (CMU)
- Classify webpages from CS departments into:
  - student, faculty, course, project

WebKB Experiment
- Train on ~5,000 hand-labeled web pages
  - Cornell, Washington, U.Texas, Wisconsin
- Crawl and classify a new site (CMU)
- Results:

<table>
<thead>
<tr>
<th></th>
<th>Student</th>
<th>Faculty</th>
<th>Person</th>
<th>Project</th>
<th>Course</th>
<th>Departmt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extracted</td>
<td>160</td>
<td>65</td>
<td>246</td>
<td>39</td>
<td>28</td>
<td>1</td>
</tr>
<tr>
<td>Correct</td>
<td>130</td>
<td>28</td>
<td>194</td>
<td>72</td>
<td>25</td>
<td>1</td>
</tr>
<tr>
<td>Accuracy</td>
<td>72%</td>
<td>42%</td>
<td>79%</td>
<td>73%</td>
<td>89%</td>
<td>100%</td>
</tr>
</tbody>
</table>

NB Model Comparison

Sample Learning Curve
(Yahoo Science Data)
Importance of Conditional Independence

Assume a domain with 20 binary (true/false) attributes $A_1, \ldots, A_{20}$ and two classes $c_1$ and $c_2$.

Goal: for any case $A = A_1, \ldots, A_{20}$ estimate $P(A, c_i)$.

A) No independence assumptions:
- The training database will not be so large.
- Huge Memory requirements / Processing time.
- Error prone (small sample error).

B) Strongest conditional independence assumptions (all attributes independent given the class) - Naive Bayes:
- $P(A, c_i) = P(A_1, c_i) \cdots P(A_{20}, c_i)$
- Computation of $2^{10} \times 2^2 = 80$ parameters.
- Space and time efficient.
- Robust estimations.
- What if the conditional independence assumptions do not hold?!

C) More relaxed independence assumptions

Tradeoff between A) and B)

Conditions for Optimality of Naive Bayes

Fact
Sometimes NB performs well even if the Conditional Independence assumptions are badly violated.

Questions
WHY? And WHEN?
Classification is about predicting the correct class label and NOT about accurately estimating probabilities.

Answer
Assume two classes $c_i$ and $c_j$. A new case $A$ arrives.

$P(A, c_i)>P(A, c_j)$

NB will classify $A$ to $c_i$.

 Besides the big error in estimating the probabilities the classification is still correct.

Correct estimation => accurate prediction but NOT accurate-prediction => Correct-estimation

Naive Bayes is Not So Naive

- Naive Bayes: first and second place in KDD-CUP 97 competition, among 16 (then) state of the art algorithms.
- Goal: Financial services industry direct mail response prediction model. Predict if the recipient of mail will actually respond to the advertisement - 750,000 records.
- Robust to Irrelevant Features
- Irrelevant Features cancel each other without affecting results
- Instead Decision Trees & Nearest Neighbor methods can heavily suffer from this.
- Very good in Domains with many equally important features
- Decision Trees suffer from fragmentation in such cases – especially if little data
- A good dependable baseline for text classification (but not the best!)
- Optimal if the Independence Assumptions hold: if assumed independence is correct, then it is the Bayes Optimal Classifier for problem
- Very fast: Learning with one pass over the data; testing linear in the number of attributes, and document collection size
- Low storage requirements
- Handles Missing Values

Interpretability of Naive Bayes

(From R.Kohavi, Silicon Graphics Minedel Evidence Visualizer)

Naive Bayes Drawbacks

- Doesn’t do higher order interactions
- Typical example: Chess end games
  - Each move completely changes the context for the next move
  - $C_{4.5} = 99.5\%$ accuracy : NB = 87% accuracy.
- What if you have BOTH high order interactions AND few training data?
- Doesn’t model features that do not equally contribute to distinguishing the classes.
  - If few features ONLY mostly determine the class, additional features usually decrease the accuracy.
- Because NB gives same weight to all features.

Final example: Text classification vs. information extraction
Naive integration of IE & TC

- Use conventional classification algorithms to classify substrings of document as "to be extracted" or not.

- This has been tried, often with limited success [Califf, Freitag]
- But in some domains this naive technique is remarkably effective.

Kushmerick: CoA Results

<table>
<thead>
<tr>
<th></th>
<th>Words</th>
<th>Pluses</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P</td>
<td>R</td>
</tr>
<tr>
<td>Message classification</td>
<td>.96</td>
<td>.66</td>
</tr>
<tr>
<td>Address classification</td>
<td>.96</td>
<td>.62</td>
</tr>
<tr>
<td>Overall accuracy</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

36 CoA messages
86 addresses
55 old, 31 new
5720 non-CoA

‘Change of Address’ email

From: Robert Kushnicky <Robert@halcyon.com>
Subject: Email update
Hi all - I’m moving jobs and wanted to stay in touch with everyone so...
My new email address is: Robert@halcyon.com
Hope all is well :) 

Modify address book, etc.

Resources