Issues in Datamining

CSE 454

A Learning Problem

Learning occurs when

PREJUDICE meets DATA!

• The nice word for prejudice is “bias”.
• What kind of hypotheses will you consider?
  – What is allowable range of functions you use when approximating?
  – E.g. conjunctions
• What kind of hypotheses do you prefer?

Learning for Text Categorization

• Manual development of text categorization functions is difficult.
• Learning Algorithms:
  – Bayesian (naive)
  – Neural network
  – Relevance Feedback (Rocchio)
  – Rule based (C4.5, Ripper, Slipper)
  – Nearest Neighbor (case based)
  – Support Vector Machines (SVM)

Bayesian Categorization

• Let set of categories be \( \{c_1, c_2, \ldots, c_n\} \)
• Let \( E \) be description of an instance.
• Determine category of \( E \) by determining for each \( c_i \)
  \[ P(c_i \mid E) = \frac{P(c_i)P(E \mid c_i)}{P(E)} \]
• \( P(E) \) can be determined since categories are complete and disjoint.
  \[ \sum_{c_i} P(c_i \mid E) = \sum_{c_i} \frac{P(c_i)P(E \mid c_i)}{P(E)} = 1 \]
  \[ P(E) = \sum_{c_i} P(c_i)P(E \mid c_i) \]
Naïve Bayesian Categorization

- Too many possible instances (exponential in $m$) to estimate all $P(E | c_i)$

- If we assume features of an instance are independent given the category ($c_i$) (conditionally independent),

\[ P(E | c_i) = P(e_1 \land e_2 \land \cdots \land e_m | c_i) = \prod_{j=1}^{m} P(e_j | c_i) \]

- Therefore, we then only need to know $P(e_j | c_i)$ for each feature and category.

Evidence is Easy?

\[ P(c_i | E) = \frac{\# E \land c_i + m p}{\# E + \# c_i + m} \]

- Or…. Are their problems?

Smooth with a Prior

\[ P(c_i | E) = \frac{\# E \land c_i + m p}{\# E + \# c_i + m} \]

$p = $ prior probability

$m = $ weight

Note that if $m = 10$, it means “I’ve seen 10 samples that make me believe $P(X_i | S) = p”$.

Hence, $m$ is referred to as the equivalent sample size

Probabilities: Important Detail!

- $P($spam$ | E_1 \ldots E_n) = \prod_{i} P($spam$ | E_i)$

Any more potential problems here?

- We are multiplying lots of small numbers

  Danger of underflow!

  \[ 0.5^{57} = 7 \times 10^{-18} \]

- Solution? Use logs and add!

  \[ p_1 \times p_2 = e^{\text{log}(p_1) + \text{log}(p_2)} \]

  - Always keep in log form

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,
  - $c$ is the number of correctly classified test instances.

- Results can vary based on sampling error due to different training and test sets.

  - Bummer… what should we do?

  - Average results over multiple training and test sets (splits of the overall data) for the best results.

  - Bummer… that means we need lots of labeled data…

Outline

- Evaluation of learning algorithms
- Co-training
- Focussed crawling

Evaluating Categorization
**N-Fold Cross-Validation**

- Ideally: test, training sets are independent on each trial.
  - But this would require too much labeled data.
- Cool idea:
  - Partition data into \( N \) equal-sized disjoint segments.
  - Run \( N \) trials, each time hold back a different segment for testing
  - Train on the remaining \( N-1 \) segments.
- This way, at least test-sets are independent.
- Report average classification accuracy over the \( N \) trials.
- Typically, \( N = 10 \).

Also nice to report standard deviation of averages.

**Cross Validation**

- Partition examples into \( k \) disjoint equiv classes
- Now create \( k \) training sets
  - Each set is union of all equiv classes *except one*
  - So each set has \((k-1)/k\) of the original training data

**Cross Validation**

- Partition examples into \( k \) disjoint equiv classes
- Now create \( k \) training sets
  - Each set is union of all equiv classes *except one*
  - So each set has \((k-1)/k\) of the original training data

**Learning Curves**

- In practice, labeled data is usually rare and expensive.
  - So…would like to know how performance varies with the number of training instances.
- *Learning curves* plot classification accuracy on independent test data (Y axis) versus number of training examples (X axis).

**N-Fold Learning Curves**

- Want learning curves averaged over multiple trials.
- Use \( N \)-fold cross validation to generate \( N \) full training and test sets.
- For each trial,
  - Train on increasing fractions of the training set,
  - Measure accuracy on test data for each point on the desired learning curve.
Co-Training Motivation

- Learning methods need labeled data
  - Lots of \(<x, f(x)>\) pairs
  - Hard to get… (who wants to label data?)
- But unlabeled data is usually plentiful…
  - Could we use this instead??????

Co-training

- Suppose each instance has two parts:
  \(x = [x_1, x_2]\)
  \(x_1, x_2\) conditionally independent given \(f(x)\)
- Suppose each half can be used to classify instance
  \(\exists f_1, f_2\) such that \(f_1(x_1) = f_2(x_2) = f(x)\)
- Suppose \(f_1, f_2\) are learnable
  \(f_1 \in H_1, f_2 \in H_2, \exists\) learning algorithms \(A_1, A_2\)

Unlabeled Instances \(\Rightarrow [x_1, x_2]\)
Labeled Instances \(\Rightarrow [x_1, x_2], f_1(x_1)\)
Hypothesis \(\Rightarrow f_2\)

Observations

- Can apply \(A_1\) to generate as much training data as one wants
  - If \(x_1\) is conditionally independent of \(x_2 / f(x)\),
  - then the error in the labels produced by \(A_1\)
    - will look like random noise to \(A_2\) !!!!
- Thus no limit to quality of the hypothesis \(A_2\) can make

It really works!

- Learning to classify web pages as course pages
  - \(x_1\) = bag of words on a page
  - \(x_2\) = bag of words from all anchors pointing to a page
- Naive Bayes classifiers
  - 12 labeled pages
  - 1039 unlabeled

<table>
<thead>
<tr>
<th>Hypothesis</th>
<th>Hypothesis</th>
<th>Hypothesis</th>
<th>Hypothesis</th>
<th>Hypothesis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Backlink</td>
<td>Backlink</td>
<td>Backlink</td>
<td>Backlink</td>
<td>Backlink</td>
</tr>
</tbody>
</table>

Focussed Crawling

- Cho paper
  - Looks at heuristics for managing URL queue
  - Aim1: completeness
  - Aim2: just topic pages
- Prioritize if word in anchor / URL
- Heuristics:
  - Pagerank
  - #backlinks

Figure 1. Basic similarity-based metrics, \(f(y) = BR(y)\), topic in computer.
Modified Algorithm

- Page is hot if:
  - Contains keyword in title, or
  - Contains 10 instances of keyword in body, or
  - Distance(page, hot-page) < 3

Results

More Results

Reinforcement Learning

Ever Feel Like Pavlov’s Poor Dog?

How is learning to act possible when…

- Actions have non-deterministic effects
  - Which are initially unknown
- Rewards / punishments are infrequent
  - Often at the end of long sequences of actions
- Learner must decide what actions to take
- World is large and complex

Applications to the Web

- Focused Crawling
  - Limited resources
    - Fetch most important pages first
  - Topic specific search engines
    - Only want pages which are relevant to topic
  - Minimize stale pages
    - Efficient re-fetch to keep index timely
    - How track the rate of change for pages?
Information Extraction

“The Truth Is Out There”

Papers: Scaling Question Answering for the Web (WWW '01)
Web-Scale Information Extraction in KnowItAll (WWW '04)

Information Goals

Finding Topics
Where can I find pages about skiing?

Finding Answers
Who killed Lincoln? “John Wilkes Booth”

Mulder

- Question Answering System
  - User asks Natural Language question:
    “Who killed Lincoln?”
  - Mulder answers: “John Wilkes Booth”
- KB = Web/Search Engines
- Domain-independent
- Fully automated

Mulder versus…

<table>
<thead>
<tr>
<th></th>
<th>Web Coverage</th>
<th>Direct Answers</th>
<th>Automated</th>
<th>Ease of use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mulder</td>
<td>Wide</td>
<td>Yes</td>
<td>Yes</td>
<td>Easy</td>
</tr>
<tr>
<td>Directories</td>
<td>Narrow</td>
<td>No</td>
<td>No</td>
<td>Easy</td>
</tr>
<tr>
<td>Search Engines</td>
<td>Wide</td>
<td>No</td>
<td>Yes</td>
<td>Difficult</td>
</tr>
<tr>
<td>AskJeeves</td>
<td>Narrow</td>
<td>No</td>
<td>No</td>
<td>Easy</td>
</tr>
</tbody>
</table>

Mulder is 99% confident the answer is John Wilkes Booth
The following are possible answers, in order of confidence:

1. John Wilkes Booth (99%)  
   **statement template**
   …John Wilkes Booth shot Lincoln with a pistol. Why? Because John Wilkes Booth killed Lincoln because he was a threat to the peace...

   **Unnatural**
   John Wilkes Booth killed Lincoln at the presidential box at Washington's Ford Theatre during a performance of "Our American Cousin."

2. Mary Todd (1%)
   **statement template**
   …Mary Todd killed Lincoln...in the middle of a conversation with her husband...

   **Unnatural**
   THE ONLY MAN WHO SHOT ABRAHAM LINCOLN IS A WOMAN’S ENTRANCED!
Challenges

- Web: Huge
  - Difficult to pinpoint facts
- Noise
  - “Conspiracy theorists believes that Mary Todd killed Lincoln”
- False claims
  - “John Glenn is the first American in space”

Architecture

Natural Language Parsing

Question Classification

Rule-based system

- Question Words (who, when)
- Question subject (what height)
  - LinkParser [Sleator et al., 91]
    - Recovers relationships among words in a sentence
  - WordNet [Miller 90]
    - Semantic network: relationships between words
    - Subtyping: height – magnitude – number

Query Formulation
Query Formulation

- Transformation examples
  - Grammatical
    “Lincoln was killed by person” “person killed Lincoln”
  - Query Expansion: “person murdered Lincoln” “person assassinated Lincoln”…
  - Verb Conversion: “When did Lincoln die?” “Lincoln died in/on date”

Answer Extraction

John Wilkes Booth killed Lincoln in the presidential box at Washington’s Ford Theater during a performance of “Our American Cousin.”

Answer Candidates

- John Wilkes Booth
- presidential box
- Lincoln

Answer Selection

- Clustering – grouping phrases with common words
  - Reduces answer variations
  - Reduces noise – Assume the truth prevails over others

Empirical Evaluations

- Test Suite
  - NIST’s TREC-8 (The 8th Text REtrieval Conference)
  - ~200 questions
  - Not guaranteed to find answers on the web

- What experiments would you run?
  - Contributions of each Mulder module
  - Mulder VS. Google VS. AskJeeves ???
Experimental Methodology

- Idea: In order to answer \( n \) questions, how much user effort has to be exerted
- Implementation:
  - A question is answered if
    - the answer phrases are found in the result pages returned by the service, or
    - they are found in the web pages pointed to by the results.
  - Bias in favor of Mulder’s opponents

Comparison Results

Contributions of Modules

- Compare Mulder with stripped down variants.

<table>
<thead>
<tr>
<th>System</th>
<th>Total effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mulder</td>
<td>1.0</td>
</tr>
<tr>
<td>No Answer Selection</td>
<td>2.3</td>
</tr>
<tr>
<td>No Query Formulation</td>
<td>3.0</td>
</tr>
<tr>
<td>No Answer Extraction</td>
<td>1.8</td>
</tr>
<tr>
<td>Nothing but Google</td>
<td>6.6</td>
</tr>
</tbody>
</table>

KnowItAll

- Mulder on Steroids.
- Instead of answering one question --- collect millions and millions of facts.
- How can we do this?

KnowItAll Architecture

- Extraction engine: rules for extracting information from text.
- Assessor: uses PMI-IR to assess probability that extractions are correct.
- Rule Learner: automatically learn new extraction rules.
KnowItAll System Architecture

Example Extraction Rule

A rule template for `instanceOf(Class1):`

```
NP1 "such as" NP2
& head(NP1)= label(Class1) & properNoun(head(NP2))
=>
instanceOf(Class1, head(NP2))
```

Example: High quality laptops such as the Thinkpad T-40.

Yields: `instanceOf(laptops, Thinkpad T-40)`.

Web-scale Validation of Facts

Probability of fact \( \phi \), given evidence \( f_1, f_2, f_n \) using Bayes rule with independence assumption.

\[
P(\phi | f_1, f_2, f_n) = \frac{P(f_1, f_2, f_n | \phi) P(\phi)}{P(f_1, f_2, f_n)}
\]

\[
P(\phi) = \prod_{i=1}^n P(f_i | \phi)
\]

Pointwise mutual information of instance \( I \) with discriminator phrase \( D \), based on search engine hit counts

\[
\text{pmiScore}(I, D) = \frac{\log(1 + \text{Hits}_I^D)}{\log 1 + \text{Hits}_I^D}
\]

Features for Web-scale Validation

Features are based on PMI score thresholds.

Find threshold that best separates positive from negative training instances (maximize entropy).

\[
f_\tau = \text{pmiScore} \geq \tau
\]

Estimate the probability of score over threshold, given that the instance is positive (or negative). Probability is the proportion of a holdout set \( H \) with score over threshold, with \( m \)-smoothing.

\[
P(\text{pmiScore} \geq \tau | \phi) = \frac{\text{Hits}_I^D | \phi}{|H| + m}
\]

Web pages retrieved versus time

Unique facts versus web pages retrieved
Conclusion

You ain’t seen nuffin’ yet!

KnowItAll System Performance

- Longest run to date: 5 days
  - Substantially longer runs are possible
- Performance of current prototype:
  - 3 web pages examined per second
  - 0.9 sentences containing extractions found per second
  - Web-scale validation assigns probability > .80 to nearly half of the extractions. These are facts.
  - KnowItAll achieve precision of 95% for facts.