Text Categorization (continued)

CSE 454

Course Overview

Immediate Organization

- Tues 11/1
  - Learning overview
  - Text categorization (Rocchio, nearest neighbor)
- Thurs 11/3
  - Text categorization (naïve Bayes); evaluation; topics
- Tues 11/8
  - Information extraction (HMMs)
- Thurs 11/10
  - KnowItAll (overview, rule learning, statistical model)
- Tues 11/15
  - KnowItAll (speedup, relational learning, opinion mining)

Review: Checkers as ML

- Task T:
  - Playing checkers
- Performance Measure P:
  - Percent of games won against opponents
- Experience E:
  - Playing practice games against itself
- Target Function
  - \( \hat{V}(b) = a + bx_1 + cx_2 + dx_3 + ex_4 + fx_5 + gx_6 \)

Approximating the Target Function

- Profound Formulation:
  - Can express any type of inductive learning as approximating a function
- E.g., Checkers
  - V: boards -> evaluation
- E.g., Handwriting recognition
  - V: image -> word
- E.g., Mushrooms
  - V: mushroom-attributes -> \{E, P\}

Supervised Learning

- Inductive learning or “Prediction”:
  - Given examples of a function \((X, F(X))\)
  - Predict function \(F(X)\) for new examples \(X\)
- Classification (“Categorization”)
  - \(F(X) = \text{Discrete}\)
- Regression
  - \(F(X) = \text{Continuous}\)
- Probability estimation
  - \(F(X) = \text{Probability}(X)\)
Learning ~ 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?
  - What kind of hypotheses do you prefer?

Two Strategies for ML

- Restriction bias: use prior knowledge to specify a restricted hypothesis space.
  - Rocchio
  - Naïve Bayes
- Preference bias: use a broad hypothesis space, but impose an ordering on the hypotheses.
  - General Bayesian learning

Rocchio Anomaly

- Prototype models ~ very strong bias
- Can’t represent polymorphic categories.

Bayesian Methods

- Learning and classification methods based on probability theory.
  - Bayes theorem plays a critical role in probabilistic learning and classification.
  - 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.
Axioms of Probability Theory

- All probabilities between 0 and 1
  \[0 \leq P(A) \leq 1\]
- True proposition has probability 1, false has probability 0.
  \[P(\text{true}) = 1 \quad P(\text{false}) = 0.\]
- The probability of disjunction is:
  \[P(A \lor B) = P(A) + P(B) - P(A \land B)\]

Conditional Probability

- \(P(A \mid B)\) is the probability of \(A\) given \(B\)
- Assumes:
  - \(B\) is all and only information known.
- Defined by:
  \[P(A \mid B) = \frac{P(A \land B)}{P(B)}\]

Independence

- \(A\) and \(B\) are independent iff:
  \[P(A \mid B) = P(A) \quad P(B \mid A) = P(B)\]
- Therefore, if \(A\) and \(B\) are independent:
  \[P(A \land B) = P(A)P(B)\]

Bayes Theorem

\[
P(H \mid E) = \frac{P(E \mid H)P(H)}{P(E)}
\]

Simple proof from definition of conditional probability:

\[
P(H \mid E) = \frac{P(H \land E)}{P(E)} \quad \text{(Def. cond. prob.)}
\]
\[
P(E \mid H) = \frac{P(H \land E)}{P(H)} \quad \text{(Def. cond. prob.)}
\]
\[
P(H \land E) = P(E \mid H)P(H) \quad \text{(Mult both sides of 2 by P(H))}
\]

QED: \[P(H \mid E) = \frac{P(E \mid H)P(H)}{P(E)} \quad \text{(Substitute 3 in 1.)}\]

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.
  \[
  \sum_{i=1}^{n} P(c_i \mid E) = \sum_{i=1}^{n} P(c_i)P(E \mid c_i) = 1
  \]
  \[P(E) = \sum_{i=1}^{n} P(c_i)P(E \mid c_i)\]
Bayesian Categorization (cont.)

- Need to know:
  - Priors: \( P(c_i) \)
  - Conditionals: \( P(E | c_i) \)
- \( P(c_i) \) are easily estimated from data.
  - If \( n_i \) of the examples in \( D \) are in \( c_i \), then \( P(c_i) = n_i / |D| \)
- Assume instance is a conjunction of binary features:
  \( E = e_1 \land e_2 \land \cdots \land e_m \)
- Too many possible instances (exponential in \( m \)) to estimate all \( P(E | c_i) \)

Naïve Bayesian Motivation

- 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.

Naïve Bayesian Categorization

- 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.

Naïve Bayes Example

- \( C = \{ \text{allergy, cold, well} \} \)
- \( e_1 = \text{sneeze} \); \( e_2 = \text{cough} \); \( e_3 = \text{fever} \)
- \( E = \{ \text{sneeze, cough, ¬fever} \} \)

<table>
<thead>
<tr>
<th></th>
<th>Well</th>
<th>Cold</th>
<th>Allergy</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P(c_i) )</td>
<td>0.9</td>
<td>0.05</td>
<td>0.05</td>
</tr>
<tr>
<td>( P(\text{sneeze}</td>
<td>c_i) )</td>
<td>0.1</td>
<td>0.9</td>
</tr>
<tr>
<td>( P(\text{cough}</td>
<td>c_i) )</td>
<td>0.1</td>
<td>0.8</td>
</tr>
<tr>
<td>( P(\text{fever}</td>
<td>c_i) )</td>
<td>0.05</td>
<td>0.05</td>
</tr>
</tbody>
</table>

Evidence is Easy?

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

*Or…. Are their problems?*
Smooth with a Prior

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

\( p = \) 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

Estimating Probabilities

- Normally, probabilities are estimated based on observed frequencies in the training data.
- If \( D \) contains \( n_i \) examples in category \( c_i \), and \( n_{ij} \) of these \( n_i \) examples contains feature \( e_j \), then:
  \[ P(e_j | c_i) = \frac{n_{ij}}{n_i} \]
- However, estimating such probabilities from small training sets is error-prone.
- If due only to chance, a rare feature, \( e_k \), is always false in the training data, \( \forall c_i: P(e_k | c_i) = 0 \).
- If \( e_k \) then occurs in a test example, \( E \), the result is that \( \forall c_i: P(E | c_i) = 0 \) and \( \forall c_i: P(c_i | E) = 0 \)

Smoothing

- To account for estimation from small samples, probability estimates are adjusted or smoothed.
- Laplace smoothing using an \( m \)-estimate assumes that each feature is given a prior probability, \( p \), that is assumed to have been previously observed in a “virtual” sample of size \( m \).
  \[ P(e_j | c_i) = \frac{n_{ij} + mp}{n_i + m} = \frac{(n_{ij} + 1)}{(n_i + 2)} \]
- For binary features, \( p \) is simply assumed to be 0.5.

Naïve Bayes for Text

- Modeled as generating a bag of words for a document in a given category by repeatedly sampling with replacement from a vocabulary \( V = \{w_1, w_2, \ldots, w_m\} \) based on the probabilities \( P(w_j | c_i) \).
- Smooth probability estimates with Laplace \( m \)-estimates assuming a uniform distribution over all words (\( p = 1/|V| \)) and \( m = |V| \)
  - Equivalent to a virtual sample of seeing each word in each category exactly once.

Text Naïve Bayes Algorithm (Train)

Let \( V \) be the vocabulary of all words in the documents in \( D \)
For each category \( c_i \in C \)
  - Let \( D_i \) be the subset of documents in \( D \) in category \( c_i \)
  - \( P(c_i) = |D_i| / |D| \)
Let \( T_i \) be the concatenation of all the documents in \( D_i \)
Let \( n_i \) be the total number of word occurrences in \( T_i \)
For each word \( w_j \in V \)
  - Let \( n_{ij} \) be the number of occurrences of \( w_j \) in \( T_i \)
  - \( P(w_j | c_i) = (n_{ij} + 1) / (n_i + |V|) \)

Text Naïve Bayes Algorithm (Test)

Given a test document \( X \)
Let \( n \) be the number of word occurrences in \( X \)
Return the category:
\[ \arg\max_{c_i \in C} P(c_i) \prod_{i=1}^{n} P(a_i | c_i) \]
where \( a_i \) is the word occurring the \( i \)th position in \( X \)
Naïve Bayes Time Complexity

- **Training Time:** $O(|D|L_d + |C||V|)$
  - where $L_d$ is the average length of a document in $D$.
  - Assumes $V$ and all $D_i$, $n_i$, and $n_{ij}$ pre-computed in $O(|D|L_d)$ time during one pass through all of the data.
  - Generally just $O(|D|L_d)$ since usually $|C||V| < |D|L_d$
- **Test Time:** $O(|C| L_t)$
  - where $L_t$ is the average length of a test document.
  - Very efficient overall, linearly proportional to the time needed to just read in all the data.
  - Similar to Rocchio time complexity.

Easy to Implement

- But…
- If you do… it probably won’t work…

Probabilities: Important Detail!

- $P(\text{spam} \mid E_1 \ldots E_n) = \prod_i P(\text{spam} \mid E_i)$

  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!
  - $p_1 * p_2 = e^{\log(p_1) + \log(p_2)}$
  - Always keep in log form

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
  - I.e. the class with maximum posterior probability…
  - Usually fairly accurate (?)!!?)
- However, due to the inadequacy of the conditional independence assumption…
  - Actual posterior-probability estimates not accurate.
  - Output probabilities generally very close to 0 or 1.

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…
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

Learning Curves

- In practice, labeled data is usually rare and expensive.
- 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 the test data
  - for each point on the desired learning curve.
Sample Learning Curve
(Yahoo Science Data)