Text Categorization

Categorization (review)

• Given:
  – A description of an instance, \( x \in X \), where \( X \) is the instance language or instance space.
  – A fixed set of categories: \( C = \{ c_1, c_2, \ldots, c_n \} \)

• Determine:
  – The category of \( x \): \( c(x) \in C \), where \( c(x) \) is a categorization function whose domain is \( X \) and whose range is \( C \).

Learning for Categorization

• A training example is an instance \( x \in X \), paired with its correct category \( c(x) \):
  \( <x, c(x)> \) for an unknown categorization function, \( c \).

• Given a set of training examples, \( D \).

• Find a hypothesized categorization function, \( h(x) \), such that:
  \[ \forall \; <x, c(x)> \in D : h(x) = c(x) \]

  Consistency

Sample Category Learning Problem

• Instance language: \( \langle \text{size, color, shape} \rangle \)
  – size \( \in \{ \text{small, medium, large} \} \)
  – color \( \in \{ \text{red, blue, green} \} \)
  – shape \( \in \{ \text{square, circle, triangle} \} \)

• \( C = \{ \text{positive, negative} \} \)

<table>
<thead>
<tr>
<th>Example</th>
<th>Size</th>
<th>Color</th>
<th>Shape</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>small</td>
<td>red</td>
<td>circle</td>
<td>positive</td>
</tr>
<tr>
<td>2</td>
<td>large</td>
<td>red</td>
<td>circle</td>
<td>positive</td>
</tr>
<tr>
<td>3</td>
<td>small</td>
<td>red</td>
<td>triangle</td>
<td>negative</td>
</tr>
<tr>
<td>4</td>
<td>large</td>
<td>blue</td>
<td>circle</td>
<td>negative</td>
</tr>
</tbody>
</table>

General Learning Issues

• Many hypotheses are usually consistent with the training data.

• Bias
  – Any criteria other than consistency with the training data that is used to select a hypothesis.

• Classification accuracy (% of instances classified correctly).
  – Measured on independent test data.

• Training time (efficiency of training algorithm).

• Testing time (efficiency of subsequent classification).

Generalization

• Hypotheses must generalize to correctly classify instances not in the training data.

• Simply memorizing training examples is a consistent hypothesis that does not generalize.

• Occam’s razor:
  – Finding a simple hypothesis helps ensure generalization.
### Text Categorization

- Assigning documents to a fixed set of categories, e.g.
  - Web pages
    - Categories in search (see microsoft.com)
    - Yahoo-like classification
  - Newsgroup Messages
    - Recommending
    - Spam filtering
  - News articles
    - Personalized newspaper
  - Email messages
    - Routing
    - Prioritizing
    - Folderizing
    - Spam filtering

### Learning for Text Categorization

- Hard to construct text categorization functions.
- Learning Algorithms:
  - Bayesian (naïve)
  - Neural network
  - Relevance Feedback (Rocchio)
  - Rule based (C4.5, Ripper, Slipper)
  - Nearest Neighbor (case based)
  - Support Vector Machines (SVM)

### Using Relevance Feedback (Rocchio)

- Use standard TF/IDF weighted vectors to represent text documents (normalized by maximum term frequency).
- For each category, compute a prototype vector by summing the vectors of the training documents in the category.
- Assign test documents to the category with the closest prototype vector based on cosine similarity.

### Rocchio Text Categorization Algorithm (Training)

Assume the set of categories is \( \{c_1, c_2, \ldots, c_n\} \)

For \( i \) from 1 to \( n \) let \( p_i = <0, 0, \ldots, 0> \) (init. prototype vectors)

For each training example \( <x, c(x)> \in D \)

Let \( d \) be the frequency normalized TF/IDF term vector for doc \( x \)

Let \( i = j \) such that \( (c_j = c(x)) \)

(sum all the document vectors in \( c_i \) to get \( p_i \))

Let \( p_i = p_i + d \)

### Rocchio Text Categorization Algorithm (Test)

Given test document \( x \)

Let \( d \) be the TF/IDF weighted term vector for \( x \)

Let \( m = -2 \) (init. maximum cosSim)

For \( i \) from 1 to \( n \):

(compute similarity to prototype vector)

Let \( s = \cosSim(d, p_i) \)

if \( s > m \)

let \( m = s \)

let \( r = c_j \) (update most similar class prototype)

Return class \( r \)

### Illustration of Rocchio Text Categorization
Rocchio Properties

- Does not guarantee a consistent hypothesis.
- Forms a simple generalization of the examples in each class (a prototype).
- Prototype vector does not need to be averaged or otherwise normalized for length since cosine similarity is insensitive to vector length.
- Classification is based on similarity to class prototypes.

Rocchio Anomaly

- Prototype models have problems with polymorphic (disjunctive) categories.

Rocchio Time Complexity

- **Note**: The time to add two sparse vectors is proportional to minimum number of non-zero entries in the two vectors.
- **Training Time**: $O(|D| (L_d + |V_d|)) = O(|D| L_d)$ where $L_d$ is the average length of a document in $D$ and $V_d$ is the average vocabulary size for a document in $D$.
- **Test Time**: $O(L_t + |C| |V_t|)$ where $L_t$ is the average length of a test document and $|V_t|$ is the average vocabulary size for a test document.
  - Assumes lengths of $p$ vectors are computed and stored during training, allowing $\text{cosSim}(d_i, p_j)$ to be computed in time proportional to the number of non-zero entries in $d_i$ (i.e. $|V_t|$)

Nearest-Neighbor Learning Algorithm

- Learning is just storing the representations of the training examples in $D$.
- Testing instance $x$:
  - Compute similarity between $x$ and all examples in $D$.
  - Assign $x$ the category of the most similar example in $D$.
- Does not explicitly compute a generalization or category prototypes.
- Also called:
  - Case-based
  - Memory-based
  - Lazy learning

K Nearest-Neighbor

- Using only the closest example to determine categorization is subject to errors due to:
  - A single atypical example.
  - Noise (i.e. error) in the category label of a single training example.
- More robust alternative is to find the $k$ most-similar examples and return the majority category of these $k$ examples.
- Value of $k$ is typically odd to avoid ties, 3 and 5 are most common.

Similarity Metrics

- Nearest neighbor method depends on a similarity (or distance) metric.
- Simplest for continuous $m$-dimensional instance space is *Euclidian distance*.
- Simplest for $m$-dimensional binary instance space is *Hamming distance* (number of feature values that differ).
- For text, cosine similarity of TF-IDF weighted vectors is typically most effective.
Nearest Neighbor Illustration (Euclidian Distance)

K Nearest Neighbor for Text

Training:
For each each training example \(<x, \text{c}(x)\> \in D
Compute the corresponding TF-IDF vector, \(d_x\), for document \(x\)

Test instance \(y\):
Compute TF-IDF vector \(d\) for document \(y\)
For each \(<x, \text{c}(x)\> \in D
Let \(s_x = \text{cosSim}(d, d_x)\)
Sort examples, \(x\), in \(D\) by decreasing value of \(s_x\)
Let \(N\) be the first \(k\) examples in \(D\).
Return the majority class of examples in \(N\)

Illustration of 3 Nearest Neighbor for Text

3NN on Rocchio Anomaly

• Nearest Neighbor handles polymorphic categories better.

Nearest Neighbor Time Complexity

• **Training Time:** \(O(|D|L_d)\) to compose TF-IDF vectors.
• **Testing Time:** \(O(L_t + |D||V_t|)\) to compare to all training vectors.
  - Assumes lengths of \(d\) vectors are computed and stored during training, allowing \(\text{cosSim}(d, d_x)\) to be computed in time proportional to the number of non-zero entries in \(d\) (i.e. \(|V_t|\))
• Testing time can be high for large training sets.

Nearest Neighbor with Inverted Index

• Determining \(k\) nearest neighbors is the same as determining the \(k\) best retrievals using the test document as a query to a database of training documents.
• Use standard VSR inverted index methods to find the \(k\) nearest neighbors.
• **Testing Time:** \(O(B|V_t|)\)
  where \(B\) is the average number of training documents in which a test-document word appears.
• Therefore, overall classification is \(O(L_t + B|V_t|)\)
  – Typically \(B << |D|\)