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: <size, color, shape>
  - size $\in$ {small, medium, large}
  - color $\in$ {red, blue, green}
  - shape $\in$ {square, circle, triangle}
- C = {positive, negative}
- D:

Example	Size	Color	Shape	Category
1	small	red	circle	positive
2	large	red	circle	positive
3	small	red	triangle	negative
4	large	blue	circle	negative

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.

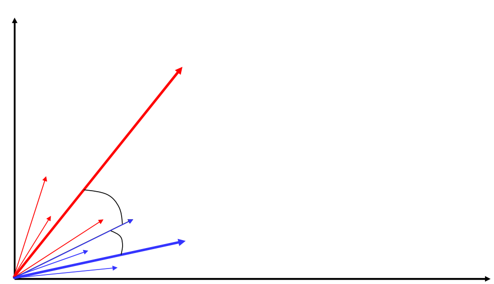
Rocchio Text Categorization Algorithm (Training)

Assume the set of categories is $\{c_1, c_2, \ldots c_n\}$
For i from 1 to n let $\mathbf{p}_i$ = <0, 0,…,0> *(init. prototype vectors)*
For each training example $\langle x, c(x)\rangle \in D$
    Let $\mathbf{d}$ 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 $\mathbf{p}_i$)
    Let $\mathbf{p}_i = \mathbf{p}_i + \mathbf{d}$

Rocchio Text Categorization Algorithm (Test)

Given test document x
Let $\mathbf{d}$ 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($\mathbf{d}$, $\mathbf{p}_i$)
    if $s > m$
        let $m = s$
        let $r = c_i$ *(update most similar class prototype)*
Return class r

Illustration of Rocchio Text Categorization



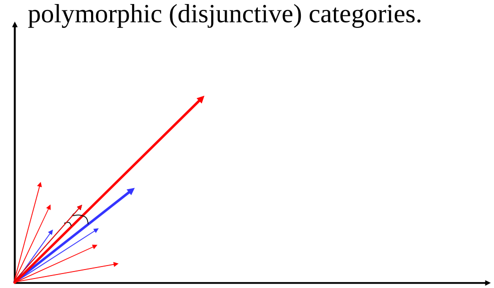
Rocchio Properties

- 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.

13

Rocchio Anomaly

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



14

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 $\mathbf{p}_i$ vectors are computed and stored during training, allowing cosSim($\mathbf{d}$, $\mathbf{p}_i$) to be computed in time proportional to the number of non-zero entries in $\mathbf{d}$ (i.e. $|V_t|$)

15

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

16

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.

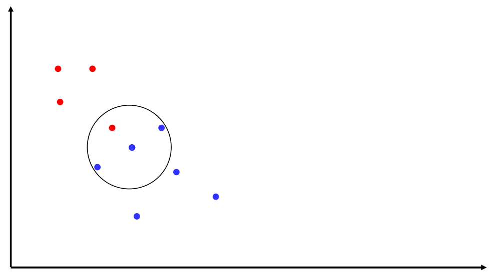
17

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.

18

3 Nearest Neighbor Illustration
(Euclidian Distance)



19

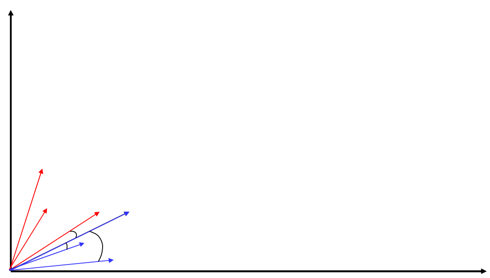
K Nearest Neighbor for Text

Training:
For each each training example <*x*, *c*(*x*)> $\in D$
Compute the corresponding TF-IDF vector, $\mathbf{d}_x$, for document *x*

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

20

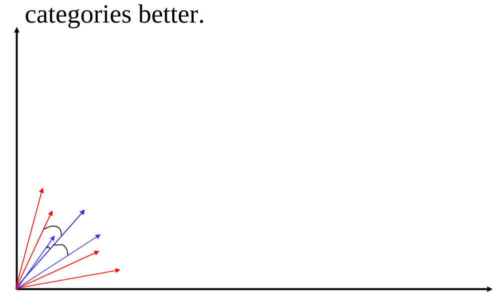
Illustration of 3 Nearest Neighbor for Text



21

3NN on Rocchio Anomaly

- Nearest Neighbor handles polymorphic categories better.



22

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 $\mathbf{d}_x$ vectors are computed and stored during training, allowing cosSim(**d**, $\mathbf{d}_x$) 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.

23

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*|

24