CS276A
Text Retrieval and Mining

Lecture 16

[Borrows slides from Ray Mooney and Barbara Rosario]

## Recap of the last lecture

- Linear Algebra
- SVD
- Latent Semantic Analysis

Okay, today's lecture doesn't very directly follow on from these topics…

- We're returning to text classification
- But we will continue a focus on a vector-space representation of texts

## Text Classification

- Today:
  - Introduction to Text Classification
    - K Nearest Neighbors
    - Decision boundaries
    - Vector space classification using centroids
    - Decision Trees (briefly)
- Next week (last week of classes!)
  - More text classification
    - Support Vector Machines
    - Text-specific issues in classification

## 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., "contains adult language" :"doesn't"*

## 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:
    $C = \{c_1, c_2, \ldots, c_n\}$
- Determine:
  - The category of $x$: $c(x) \in C$, where $c(x)$ is a *categorization function* whose domain is $X$ and whose range is $C$.
    - We want to know how to build categorization functions ("classifiers").

## Recall: Vector Space Representation

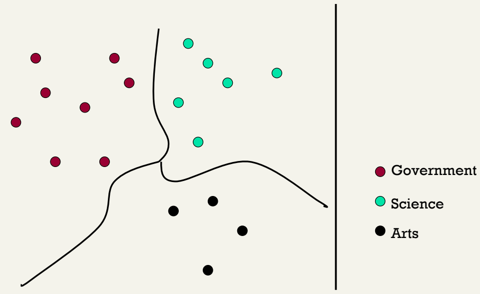
- Each document is a vector, one component for each term (= word).
- Normalize to unit length.
- High-dimensional vector space:
  - Terms are axes
  - 10,000+ dimensions, or even 100,000+
  - Docs are vectors in this space

Exercise: think how this compares with probabilistic representations (multinomial and multivariate Bernoulli)

## Classification Using Vector Spaces

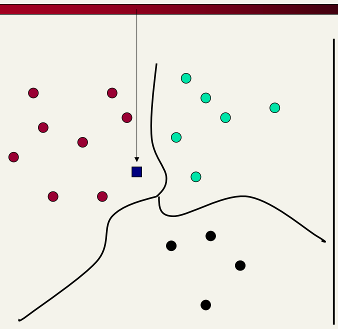
- Each training doc a point (vector) labeled by its topic (= class)
- Hypothesis: docs of the same class form a contiguous region of space
- We define surfaces to delineate classes in space

## Classes in a Vector Space




## Test Document = Government

Similarity hypothesis true in general?

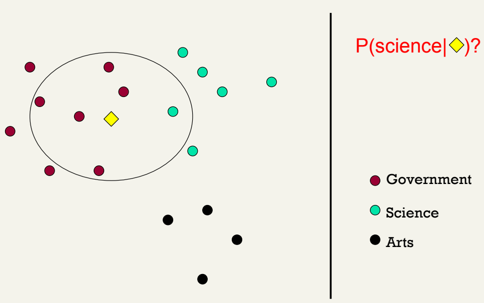



## k Nearest Neighbor Classification

- To classify document *d* into class c
- Define *k*-neighborhood N as *k* nearest neighbors of *d*
- Count number of documents i in N that belong to c
- Estimate P(c|*d*) as i/k
- Choose as class $\text{argmax}_c$ P(c|*d*) [ = majority class]

## Example: k=6 (6NN)

P(science|◇)?




## 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 learning
  - Memory-based learning
  - Lazy learning

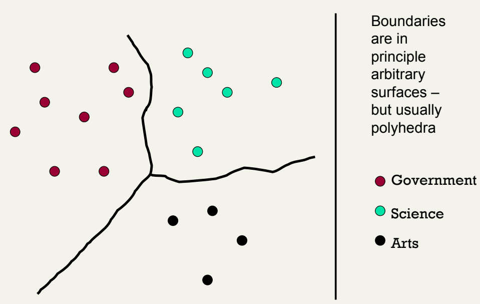
## kNN Is Close to Optimal

- Cover and Hart 1967
- Asymptotically, the error rate of 1-nearest-neighbor classification is less than twice the Bayes rate [error rate of classifier knowing model that generated data]
- In particular, asymptotic error rate is 0 if Bayes rate is 0.
- Assume: query point coincides with a training point.
- Both query point and training point contribute error → 2 times Bayes rate

## K Nearest-Neighbor

- Using only the closest example to determine the 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.

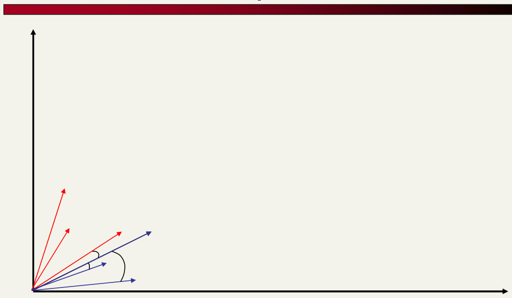
## kNN decision boundaries




## 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.

## Illustration of 3 Nearest Neighbor for Text Vector Space



## Nearest Neighbor with Inverted Index

- Naively finding nearest neighbors requires a linear search through |*D*| documents in collection
- But 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 vector space 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.
  - Typically $B << |D|$

## kNN: Discussion

- No feature selection necessary
- Scales well with large number of classes
  - Don't need to train *n* classifiers for *n* classes
- Classes can influence each other
  - Small changes to one class can have ripple effect
- Scores can be hard to convert to probabilities
- No training necessary
  - Actually: perhaps not true. (Data editing, etc.)

## kNN vs. Naive Bayes

- Bias/Variance tradeoff
  - Variance ≈ Capacity
- kNN has high variance and low bias.
  - Infinite memory
- NB has low variance and high bias.
  - Decision surface has to be linear (hyperplane – see later)
- Consider: Is an object a tree? (Burges)
  - Too much capacity/variance, low bias
    - Botanist who memorizes
    - Will always say "no" to new object (e.g., # leaves)
  - Not enough capacity/variance, high bias
    - Lazy botanist
    - Says "yes" if the object is green
  - Want the middle ground

## Binary Classification

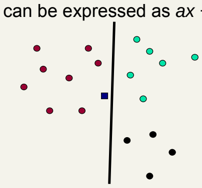
- Consider 2 class problems
  - Deciding between two classes, perhaps, government and non-government

    [one-versus-rest classification]
- How do we define (and find) the separating surface?
- How do we test which region a test doc is in?

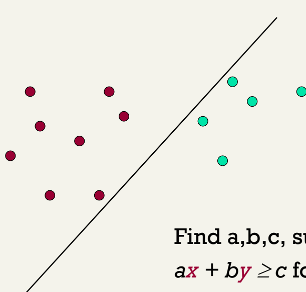
## Separation by Hyperplanes

- A strong high-bias assumption is *linear separability*:
  - in 2 dimensions, can separate classes by a line
  - in higher dimensions, need hyperplanes
- Can find separating hyperplane by *linear programming*

  (or can iteratively fit solution via perceptron):
  - separator can be expressed as $ax + by = c$



## Linear programming / Perceptron

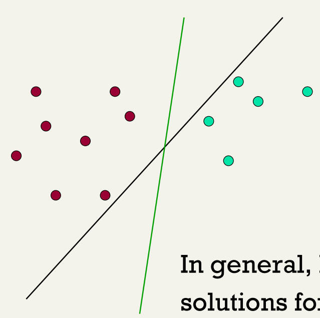


Find a,b,c, such that

$ax + by \geq c$ for red points

$ax + by \leq c$ for green points.

## Which Hyperplane?

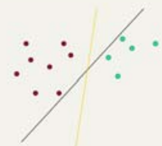


In general, lots of possibl

solutions for $a,b,c$.

## Which Hyperplane?

- Lots of possible solutions for *a,b,c.*
- Some methods find a separating hyperplane, but not the optimal one [according to some criterion of expected goodness]
  - E.g., perceptron
- Most methods find an optimal separating hyperplane
- Which points should influence optimality?
  - All points
    - Linear regression
    - Naïve Bayes
  - Only "difficult points" close to decision boundary
    - Support vector machines



## Linear classifier: Example

- Class: "interest" (as in interest rate)
- Example features of a linear classifier
- | $w_i$ $t_i$ | $w_i$ $t_i$ |
  |---|---|
  | • 0.70 prime | • -0.71 dlrs |
  | • 0.67 rate | • -0.35 world |
  | • 0.63 interest | • -0.33 sees |
  | • 0.60 rates | • -0.25 year |
  | • 0.46 discount | • -0.24 group |
  | • 0.43 bundesbank | • -0.24 dlr |
- To classify, find dot product of feature vector and weights

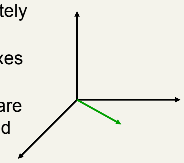
## Linear Classifiers

- Many common text classifiers are linear classifiers
  - Naïve Bayes
  - Perceptron
  - Rocchio
  - Logistic regression
  - Support vector machines (with linear kernel)
  - Linear regression
  - (Simple) perceptron neural networks
- Despite this similarity, large performance differences
  - For separable problems, there is an infinite number of separating hyperplanes. Which one do you choose?
  - What to do for non-separable problems?
  - Different training methods pick different hyperplanes
- Classifiers more powerful than linear often don't perform better. Why?

Exercise: show Naive Bayes is linear in log space

## High Dimensional Data

- Pictures like the one at right are absolutely misleading!
- Documents are zero along almost all axes
- Most document pairs are very far apart (i.e., not strictly orthogonal, but only share very common words and a few scattered others)
- In classification terms: virtually all document sets are separable, for most any classification
- This is part of why linear classifiers are quite successful in this domain



## Aside: Author identification

- Federalist papers
  - 77 short essays written in 1787-1788 by Hamilton, Jay and Madison to persuade NY to ratify the US Constitution; published under a pseudonym
  - The authorships of 12 papers was in dispute
  - In 1964 Mosteller and Wallace* solved the problem
  - They identified 70 *function* words as good candidates for authorship analysis
  - Using statistical inference they concluded the author was Madison

*Mosteller, Frederick and Wallace, David L. 1964. Inference and Disputed Authorship: The Federalist.

29

## Function words for Author Identification

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | *a* | 15 | *do* | 29 | *is* | 43 | *or* | 57 | *this* |
| 2 | *all* | 16 | *down* | 30 | *it* | 44 | *our* | 58 | *to* |
| 3 | *also* | 17 | *even* | 31 | *its* | 45 | *shall* | 59 | *up* |
| 4 | *an* | 18 | *every* | 32 | *may* | 46 | *should* | 60 | *upon* |
| 5 | *and* | 19 | *for* | 33 | *more* | 47 | *so* | 61 | *was* |
| 6 | *any* | 20 | *from* | 34 | *must* | 48 | *some* | 62 | *were* |
| 7 | *are* | 21 | *had* | 35 | *my* | 49 | *such* | 63 | *what* |
| 8 | *as* | 22 | *has* | 36 | *no* | 50 | *than* | 64 | *when* |
| 9 | *at* | 23 | *have* | 37 | *not* | 51 | *that* | 65 | *which* |
| 10 | *be* | 24 | *her* | 38 | *now* | 52 | *the* | 66 | *who* |
| 11 | *been* | 25 | *his* | 39 | *of* | 53 | *their* | 67 | *will* |
| 12 | *but* | 26 | *if* | 40 | *on* | 54 | *then* | 68 | *with* |
| 13 | *by* | 27 | *in* | 41 | *one* | 55 | *there* | 69 | *would* |
| 14 | *can* | 28 | *into* | 42 | *only* | 56 | *things* | 70 | *your* |

Table 1: Function Words and Their Code Numbers

## Function Words for Author Identification

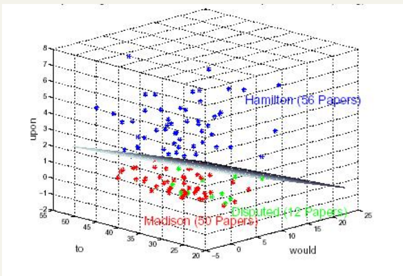
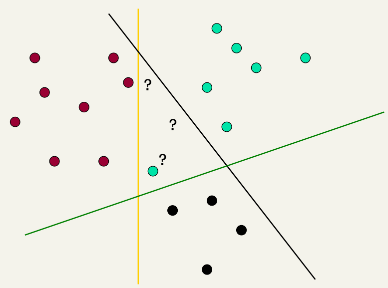



Figure 1: Obtained Hyperplane in 3 dimensions

## More Than Two Classes

- Any-of or multivalue classification
  - Classes are independent of each other.
  - A document can belong to 0, 1, or >1 classes.
  - Decompose into *n* binary problems
  - Quite common for documents
- One-of or multinomial or polytomous classification
  - Classes are mutually exclusive.
  - Each document belongs to exactly one class
  - E.g., digit recognition is polytomous classification
    - Digits are mutually exclusive

## Composing Surfaces: Issues




## Set of Binary Classifiers: Any of

- Build a separator between each class and its complementary set (docs from all other classes).
- Given test doc, evaluate it for membership in each class.
- Apply decision criterion of classifiers independently
- Done

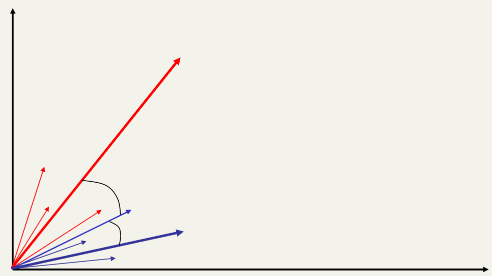
## Set of Binary Classifiers: One of

- Build a separator between each class and its complementary set (docs from all other classes).
- Given test doc, evaluate it for membership in each class.
- Assign document to class with:
  - maximum score
  - maximum confidence
  - maximum probability
- Why different from multiclass/any of classification?

## Using Relevance Feedback (Rocchio)

- Relevance feedback methods can be adapted for text categorization.
- Use standard TF/IDF weighted vectors to represent text documents
- For each category, compute a *prototype* vector by summing the vectors of the training documents in the category.
  - Prototype = centroid of members of class
- Assign test documents to the category with the closest prototype vector based on cosine similarity.

## Illustration of Rocchio Text Categorization



## Rocchio Properties

- 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.
- Does not guarantee classifications are consistent with the given training data.

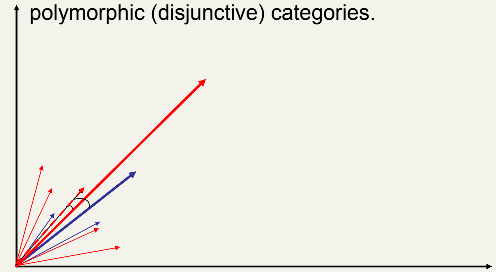
Why not?

## 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 **centroid** vectors are computed and stored during training, allowing cosSim(**d**, **c**$_i$) to be computed in time proportional to the number of non-zero entries in **d** (i.e. $|V_t|$)

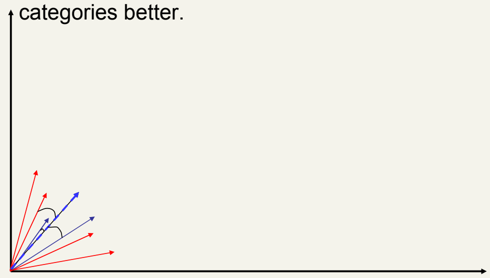
## Rocchio Anomaly

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



## 3 Nearest Neighbor Comparison

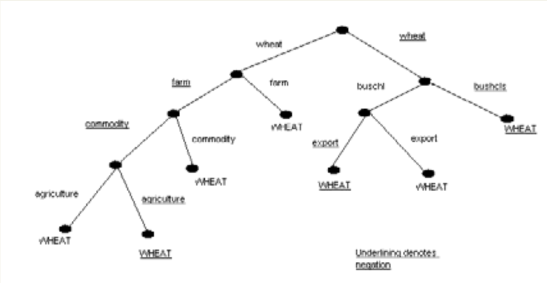
- Nearest Neighbor tends to handle polymorphic categories better.



## Decision Tree Classification

- Tree with internal nodes labeled by terms
- Branches are labeled by tests on the weight that the term has
- Leaves are labeled by categories
- Classifier categorizes document by descending tree following tests to leaf
- The label of the leaf node is then assigned to the document
- Most decision trees are binary trees (never disadvantageous; may require extra internal nodes)
- DT make good use of a few high-leverage features

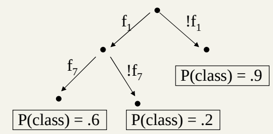
## Decision Tree Categorization: Example




Geometric interpretation of DT?

## Decision Tree Learning

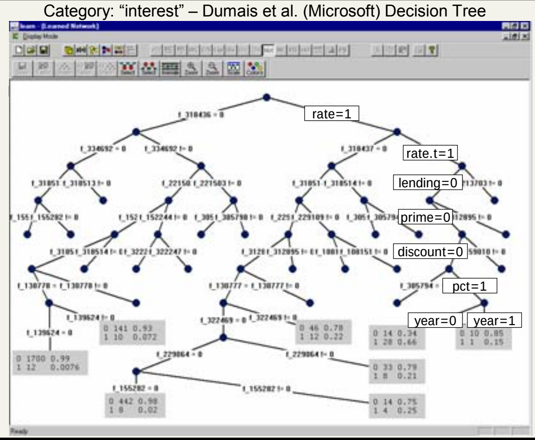
- Learn a sequence of tests on features, typically using top-down, greedy search
  - At each stage choose the unused feature with highest Information Gain (feature/class MI)
- Binary (yes/no) or continuous decisions




## Decision Tree Learning

- Fully grown trees tend to have decision rules that are overly specific and are therefore unable to categorize documents well
  - Therefore, pruning or early stopping methods for Decision Trees are normally a standard part of classification packages
- Use of small number of features is potentially bad in text cat, but in practice decision trees do well for some text classification tasks
- Decision trees are very easily interpreted by humans – much more easily than probabilistic methods like Naive Bayes
- Decision Trees are normally regarded as a symbolic machine learning algorithm, though they can be used probabilistically

Category: "interest" – Dumais et al. (Microsoft) Decision Tree




## Summary: Representation of Text Categorization Attributes

- Representations of text are usually very high dimensional (one feature for each word)
- High-bias algorithms that prevent overfitting in high-dimensional space generally work best
- For most text categorization tasks, there are many relevant features and many irrelevant ones
- Methods that combine evidence from many or all features (e.g. naive Bayes, kNN, neural-nets) generally 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

## References


- Fabrizio Sebastiani. Machine Learning in Automated Text Categorization. *ACM Computing Surveys*, 34(1):1-47, 2002.
- Tom Mitchell, Machine Learning. McGraw-Hill, 1997.
- Yiming Yang & Xin Liu, A re-examination of text categorization methods. *Proceedings of SIGIR*, 1999.
- Evaluating and Optimizing Autonomous Text Classification Systems (1995) David Lewis. Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval
- Foundations of Statistical Natural Language Processing. Chapter 16. MIT Press. Manning and Schütze.
- Trevor Hastie, Robert Tibshirani and Jerome Friedman, *Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer-Verlag, New York.