Text Classification:
Vector space classification

SEG5680
Recall: Vector Space Representation

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

- How can we do classification in this space?
  - Recall that Naïve Bayes classification does not make use of the term weight.
Classification Using Vector Spaces

- As before, the training set is a set of documents, each labeled with its class (e.g., topic)
- In vector space classification, this set corresponds to a labeled set of points (or, equivalently, vectors) in the vector space
- Premise 1: Documents in the same class form a contiguous region of space
- Premise 2: Documents from different classes don’t overlap (much)
- We define surfaces to delineate classes in the space
Documents in a Vector Space

- Government
- Science
- Arts
Test Document of what class?

- Government
- Science
- Arts
Test Document = Government

The main strategy is how to find good separators
Using Rocchio for text classification

- Relevance feedback methods can be adapted for text categorization
  - As noted before, relevance feedback can be viewed as 2-class classification
    - Relevant vs. nonrelevant documents
- Use standard TF/IDF weighted vectors to represent text documents
- For training documents in 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.
Definition of centroid

\[ \tilde{\mu}(c) = \frac{1}{|D_c|} \sum_{d \in D_c} \vec{v}(d) \]

where \( D_c \) is the set of all documents that belong to class \( c \) and \( \vec{v}(d) \) is the vector space representation of \( d \).

- **Note that centroid will in general not be a unit vector even when the inputs are unit vectors.**
Illustration of Rocchio Text Categorization

Note: Centroid vectors are illustrated by directions only.
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.
Rocchio Anomaly

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

- Rocchio forms a simple representation for each class: the centroid/prototype
- Classification is based on similarity to / distance from the prototype/centroid
- It does not guarantee that classifications are consistent with the given training data

- It is little used outside text classification, but has been used quite effectively for text classification
- Again, cheap to train and test documents
k Nearest Neighbor Classification

- kNN = k Nearest Neighbor

- 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|♦)?

- Government
- Science
- Arts
Nearest-Neighbor Learning Algorithm

- Learning is just storing the representations of the training examples in $D$.
- Testing instance $x$ (under 1NN):
  - 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
- Rationale of kNN: contiguity hypothesis
k Nearest Neighbor

- Using only the closest example (1NN) to determine the class is subject to errors due to:
  - A single atypical example.
  - Noise (i.e., an 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

Boundaries are in principle arbitrary surfaces – but usually polyhedra

Government
Science
Arts

kNN gives locally defined decision boundaries between classes – far away points do not influence each classification decision (unlike in Naïve Bayes, Rocchio, etc.)
Similarity Metrics

- Nearest neighbor method depends on a similarity (or distance) metric.
- For text, cosine similarity of tf.idf weighted vectors is typically most effective.
Illustration of 3 Nearest Neighbor for Text Vector Space
3 Nearest Neighbor Comparison

- Nearest Neighbor tends to handle polymorphic categories better.
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.)
- May be more expensive at test time
Linear Classifiers
Linear Classifiers

- 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 decide which region a test doc is in?
A strong high-bias assumption is *linear separability*: in 2 dimensions, can separate classes by a line; in higher dimensions, need hyperplanes; separator can be expressed as $ax + by = c$.
Which Hyperplane?

In general, lots of possible solutions for $a, b, c$. 
Which Hyperplane?

- Lots of possible solutions for $a, b, c$.
- Some methods find an optimal separating hyperplane
  [according to some criterion of expected goodness]
- Which points should influence optimality?
  - All points
    - Linear regression
    - Naïve Bayes
  - Only “difficult points” close to decision boundary
    - Support vector machines
High-Dimensional Linear Classifier

- For general linear classifiers, assign the document \( d \) with \( m \) features \( d=(d_1,\ldots,d_M) \) to one class if:
  \[
  \left( \sum_{i=1}^{M} w_i d_i \right) - \theta > 0
  \]
  Otherwise, assign to the other class.

Linear classifier: Example

- Class: “interest” (as in interest rate)
- Example features of a linear classifier
  - \( w_i \quad t_i \)
    - 0.70 prime
    - 0.67 rate
    - 0.63 interest
    - 0.60 rates
    - 0.46 discount
    - 0.43 bundesbank
  - \( w_i \quad t_i \)
    - -0.71 dlr
    - -0.35 world
    - -0.33 sees
    - -0.25 year
    - -0.24 group
    - -0.24 dlr

- To classify, find dot product of feature vector and weights
Two-class Rocchio as a linear classifier

- Rocchio is a linear classifier because:

\[
\sum_{i=1}^{M} w_i d_i \left( \sum_{i=1}^{M} w_i d_i \right) - \theta > 0
\]

\[
\vec{w} = \vec{\mu}(c_1) - \vec{\mu}(c_2)
\]

\[
\theta = 0.5 \times (|\vec{\mu}(c_1)|^2 - |\vec{\mu}(c_2)|^2)
\]
Linear Classifiers

- Many common text classifiers are linear classifiers
  - Naïve Bayes
  - Perceptron
  - Rocchio
  - Logistic regression
  - Support vector machines (with linear kernel)
  - Linear regression
- Despite this similarity, noticeable performance differences
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
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
- Though maybe you could do better by considering dependencies between categories
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
Which classifier do I use for a given text classification problem?

- How much training data is available?
- How simple/complex is the problem? (linear vs. nonlinear decision boundary)
- How noisy is the problem?
- How stable is the problem over time?
  - For an unstable problem, it’s better to use a simple and robust classifier.
References

- *IIR 14*
- Open Calais: Automatic Semantic Tagging
  - Free (but they can keep your data), provided by Thompson/Reuters
- Weka: A data mining software package that includes an implementation of many ML algorithms