CSCI 5417
Information Retrieval Systems

Jim Martin

Lecture 14
10/11/2011

Today 10/11

- Finish up classification
- Clustering
  - Flat clustering
  - Hierarchical clustering

SVM Summary

- Support vector machines (SVM)
  - Choose hyperplane based on support vectors
    - Support vector = "critical" point close to decision boundary
  - Degree-1 SVMs are just fancy linear classifiers.
  - Best performing text classifier
    - But there are cheaper methods that perform about as well as SVM, such as logistic regression (MaxEnt)
  - Partly popular due to availability of SVMlight
    - SVMlight is accurate and fast – and free (for research)
    - Also libSVM, tinySVM, Weka

The Real World

P. Jackson and I. Moulinier: *Natural Language Processing for Online Applications*

- "There is no question concerning the commercial value of being able to classify documents automatically by content. There are myriad potential applications of such a capability for corporate Intranets, government departments, and Internet publishers"
- "Understanding the data is one of the keys to successful categorization, yet this is an area in which most categorization tool vendors are extremely weak. Many of the 'one size fits all' tools on the market have not been tested on a wide range of content types."

The Real World

- Gee, I'm building a text classifier for real, now!
- What should I do?

- How much training data do you have?
  - None
  - Very little
  - Quite a lot
  - A huge amount and its growing

Manually written rules

- No training data, but adequate domain expertise go with hand-written rules
  - If (wheat or grain) and not (whole or bread) then
    - Categorize as grain
  - In practice, rules get a lot bigger than this
  - Can also be phrased using tf or tf.idf weights
- With careful crafting (human tuning on development data) performance is high:
  - *Construe*: 94% recall, 84% precision over 675 categories (Hayes and Weinstein 1990)
- Amount of work required is huge
  - Estimate 2 days per class … plus ongoing maintenance

Very little data?

- If you're just doing supervised classification, you should stick to something with high bias
  - There are theoretical results that naïve Bayes should do well in such circumstances (Ng and Jordan 2002 NIPS)
- An interesting research approach is to explore semi-supervised training methods
  - Bootstrapping, EM over unlabeled documents, …
- The practical answer is to get more labeled data as soon as you can
  - How can you insert yourself into a process where humans will be willing to label data for you

A reasonable amount of data?

- Perfect, use an SVM
- But if you are using a supervised ML approach, you should probably be prepared with the "hybrid" solution
  - Users like to hack, and management likes to be able to implement quick fixes immediately
  - Hackers like regular expressions

A huge amount of data?

- This is great in theory for doing accurate classification…
- But it could easily mean that expensive methods like SVMs (training time) or kNN (testing time) are quite impractical
- Naïve Bayes can come back into its own again!
  - Or other methods with linear training/test complexity like regularized logistic regression

How many categories?

- A few (well separated ones)?
  - Easy!
- A zillion closely related ones?
  - Library of Congress classifications, MeSH terms, Reuters...
  - Quickly gets difficult!
  - Evaluation is tricky

How can one tweak performance?

- Aim to exploit any domain-specific useful features that give special meanings or that zone the data
  - an author byline, mail headers, titles, zones in texts.
- Aim to collapse things that would be treated as different but shouldn't be.
  - Part numbers, chemical formulas, gene/ protein names, dates, etc.

Do "hacks" help?

- You bet!
- You can get a lot of value by differentially weighting contributions from different document zones:
  - Upweighting title words helps (Cohen & Singer 1996)
    - Doubling the weighting on the title words is a good rule of thumb
  - Upweighting the first sentence of each paragraph helps (Murata, 1999)
  - Upweighting sentences that contain title words helps (Ko *et al,* 2002)

Measuring Classification Figures of Merit

- Not just accuracy; in the real world, there are economic measures:
  - Your choices are:
    - Do no classification
      - That has a cost (hard to compute)
    - Do it all manually
      - Has an easy to compute cost if doing it like that now
    - Do it all with an automatic classifier
      - Mistakes have a cost
    - Do it with a combination of automatic classification and manual review of uncertain/difficult/"new" cases
  - Commonly the last method is most cost efficient and is adopted

A common problem: Concept Drift

- Categories change over time
- Example: "president of the united states"
  - 1999: clinton is great feature
  - 2002: clinton is bad feature
- One measure of a text classification system is how well it protects against concept drift.
  - Can favor simpler models like Naïve Bayes
- Feature selection: can be bad in protecting against concept drift

The Concept Drift Problem

- Things change
- Example: "president of the united states"
  - 1999: clinton is great feature
  - 2010: clinton is bad feature
- One measure of a text classification system is how well it protects against concept drift.
  - Can favor simpler models like Naïve Bayes

What is Clustering?

- Clustering: the process of grouping a set of objects into classes of similar objects
  - It is the most common form of *unsupervised learning*
    - Unsupervised learning = learning from raw data, as opposed to supervised data where a classification of examples is given
  - A common and important task that finds many applications in IR and other places

Clustering in IR

- Whole corpus analysis/navigation
  - Google News
- For better navigation of search results
  - Effective "user recall" will be higher
- For speeding up vector space retrieval
  - Faster search

For improving ad hoc search

- *Cluster hypothesis* - Documents with similar text are related with respect to the information needs satisfied by the documents
  - Independent of any particular query
- Therefore, to improve search recall:
  - Cluster docs in corpus *a priori*
  - When a query matches a doc *D*, also return other docs in the cluster containing *D*

For better navigation of search results

- For grouping search results thematically
  - Yippy

Issues for clustering

- Representation for clustering
  - Document representation
    - Vector space? Normalization?
  - Similarity/distance metric
    - Cosine
    - Euclidean distance
- How many clusters?
  - Fixed *a priori*?
  - Completely data driven?
    - Avoid "trivial" clusters - too large or small
      - In an application, if a cluster's too large, then for navigation purposes you've wasted an extra user click without whittling down the set of documents much.

Clustering Algorithms

- Partitional algorithms
  - Usually start with a random (partial) partitioning
  - Refine it iteratively
    - *K*-means clustering
    - Model based clustering
- Hierarchical algorithms
  - Bottom-up, agglomerative
  - Top-down, divisive

Partitioning Algorithms

- Partitioning method: Construct a partition of *n* documents into a set of *K* clusters
- Given: a set of documents and the number *K*
- Find: a partition of *K* clusters that optimizes the chosen partitioning criterion
  - Globally optimal: exhaustively enumerate all partitions
  - Effective heuristic methods: *K*-means and *K*-medoids algorithms

K-Means

- Assumes documents are real-valued vectors.
- Clusters based on *centroids* (aka the *center of gravity* or mean) of points in a cluster, *c*:

$$\mu(c) = \frac{1}{|c|}\sum_{\vec{x}\in c}\vec{x}$$

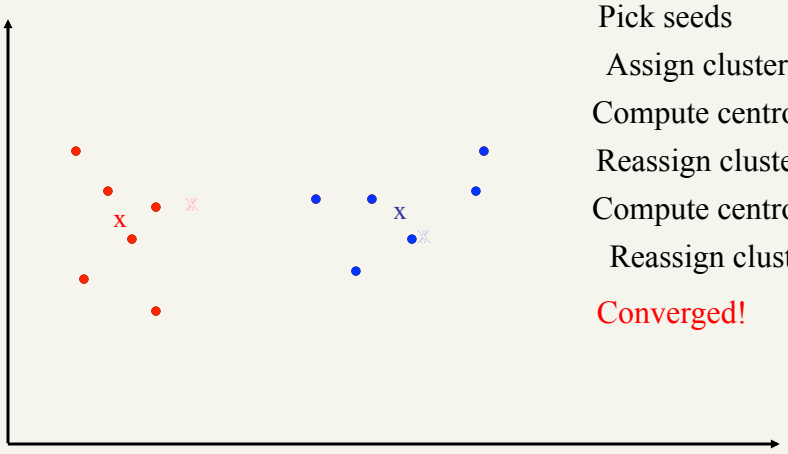
- Iterative reassignment of instances to clusters is based on distance to the current cluster centroids.
  - (Or one can equivalently phrase it in terms of similarities)

K-Means Algorithm

Select K random docs $\{s_1, s_2, \ldots s_K\}$ as seeds for initial clusters c_k
Until stopping criterion:
  For each doc d_i:
    Assign d_i to the cluster c_j
    such that $dist(x_i, s_j)$ is minimal.

  For each cluster c_j
    s_j = centroid(c_j)

K Means Example (*K*=2)



Pick seeds
Assign clusters
Compute centroids
Reassign clusters
Compute centroids
Reassign clusters
Converged!

Termination conditions

- Several possibilities
  - A fixed number of iterations
  - Doc partition unchanged
  - Centroid positions don't change

Efficiency: Medoid As Cluster Representative

- The centroid does not have to be a document (typically won't be)
- Medoid: A cluster representative that is one of the documents
- For example: the document closest to the centroid
- One reason this is useful
  - Consider the representation of a large cluster (>1000 documents)
  - The centroid of this cluster will be a dense vector
  - The medoid of this cluster will be a sparse vector

Evaluation of clustering

- Perhaps the most substantive issue in data mining in general:
  - how do you measure goodness?
- Most measures focus on computational efficiency
  - Time and space
- For application of clustering to search:
  - Measure retrieval effectiveness

Approaches to evaluating

- Anecdotal
- User inspection
- Ground "truth" comparison
  - Cluster retrieval
- Purely quantitative measures
  - Probability of generating clusters found
  - Average distance between cluster members
- Utility (in vivo)

Anecdotal evaluation

- Probably the commonest (and surely the easiest)
  - "I wrote this clustering algorithm and look what it found!"
- No benchmarks, no comparison possible
- Any clustering algorithm will pick up the easy stuff like partition by languages
- Generally, unclear scientific or practical value.

User inspection

- Induce a set of clusters or a navigation tree
- Have subject matter experts evaluate the results and score them
  - some degree of subjectivity
- Often combined with search results clustering
- Not clear how reproducible across tests
- Expensive / time-consuming

Ground truth comparison

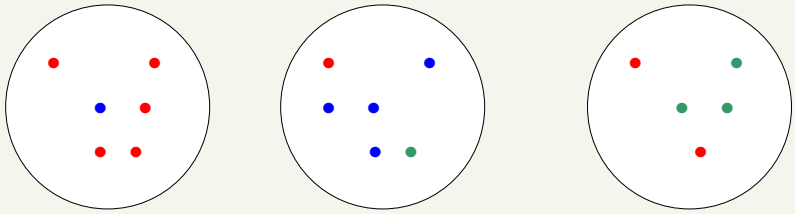
- Take a union of docs from a taxonomy & cluster
  - Yahoo!, ODP, newspaper sections …
- Compare clustering results to original taxonomy
  - e.g., 80% of the clusters found map "cleanly" to taxonomy nodes
  - How exactly would we measure this?
- But is that the "right" answer?
  - There can be several equally right answers
- For the docs given, the static prior taxonomy may be incomplete/wrong in places
  - the clustering algorithm may have gotten right things not in the static taxonomy

External Evaluation of Cluster Quality

- Simple measure: purity, the ratio between the dominant class in the cluster and the size of cluster

$$Purity(\omega_i) = \frac{1}{n_i} \max_j (n_{ij}) \quad j \in C$$

Purity example



Cluster I Cluster II Cluster III

Cluster I: Purity = 1/6 (max(5, 1, 0)) = 5/6

Cluster II: Purity = 1/6 (max(1, 4, 1)) = 4/6

Cluster III: Purity = 1/5 (max(2, 0, 3)) = 3/5

Utility viewpoint

- Anything - including clustering - is only as good as the utility it provides
- For clustering: net economic gain produced by an approach (vs. another approach)
- Strive for a concrete optimization problem
- Example
  - Recommendation systems

Misc. Clustering Topics

- Clustering terms
- Clustering people
- Feature selection
- Labeling clusters

Term vs. document space

- So far, we *clustered documents* based on their similarities in term space
- For some applications, e.g., topic analysis for inducing navigation structures, you can "dualize":
  - Use docs as axes
  - Represent (some) terms as vectors
  - *Cluster terms*, *not* docs

Feature selection

- Which terms to use as axes for vector space?
- Large body of (ongoing) research
- IDF is a form of feature selection
  - Can exaggerate noise e.g., mis-spellings
- Better is to use highest weight *mid-frequency* words – the most discriminating terms
- Pseudo-linguistic heuristics, e.g.,
  - drop stop-words
  - stemming/lemmatization
  - use only nouns/noun phrases
- Good clustering should "figure out" some of these

Clustering people

- Take documents (pages) containing mentions of ambiguous names and partition the documents into bins with identical referents.
  - SemEval competition
    - Web People Search Task

Labeling clusters

- After clustering algorithm finds clusters - how can they be useful to the end user?
- Need pithy label for each cluster
  - In search results, say "Somali" or "Pittsburgh Pirates" in the ***pirates*** example.

How to Label Clusters

- Show titles of typical documents
  - Titles are easy to scan
  - Authors create them for quick scanning!
  - But you can only show a few titles which may not fully represent cluster
- Show words/phrases prominent in cluster
  - More likely to fully represent cluster
  - Use distinguishing words/phrases
    - Differential labeling
  - But harder to scan

Labeling

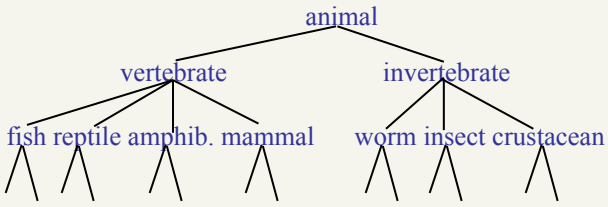
- Common heuristics - list 5-10 most frequent terms in the centroid vector.
  - Drop stop-words; stem.
- Differential labeling by frequent terms
  - Within a collection "Computers", clusters all have the word ***computer*** as frequent term.
  - Discriminant analysis of centroids.
- Perhaps better: distinctive noun phrases
  - Requires NP chunking

Hierarchical Clustering

- These approaches are based on the notion of a partition
  - Each item (document) goes into 1 and only 1 cluster
- What if you want more structure than that? Two options
  - Soft clusters (sort of this and sort of that)
  - Hierarchical clusters

Hierarchical Clustering

- Build a tree-based hierarchical taxonomy (*dendrogram*) from a set of unlabeled examples.



Hierarchical Clustering algorithms

- **Agglomerative (bottom-up):**
  - Start with each document being a single cluster.
  - Eventually all documents belong to the same cluster.
- **Divisive (top-down):**
  - Start with all documents belong to the same cluster.
  - Eventually each node forms a cluster on its own.
- Does not require the number of clusters ***k*** to be known in advance
  - But it does need a cutoff or threshold parameter condition

Hierarchical -> Partition

- Run the algorithm to completion
  - Take a slice across the tree at some level
    - Produces a partition
- Or insert an early stopping condition into either top-down or bottom-up

Hierarchical Agglomerative Clustering (HAC)

- Assumes a similarity function for determining the similarity of two instances and two clusters.
- Starts with all instances in separate clusters and then repeatedly joins the two clusters that are most similar until there is only one cluster.
- The history of merging forms a binary tree or hierarchy.

Hierarchical Clustering

- Key problem: as you build clusters, how do you represent each cluster, to tell which pair of clusters is closest?

"Closest pair" in Clustering

- Many variants to defining closest pair of clusters
  - Single-link
    - Similarity of the most cosine-similar (single-link)
  - Complete-link
    - Similarity of the "furthest" points, the least cosine-similar
  - "Center of gravity"
    - Clusters whose centroids (centers of gravity) are the most cosine-similar
  - Average-link
    - Average cosine between all pairs of elements

Single Link Agglomerative Clustering

- Use maximum similarity of pairs:

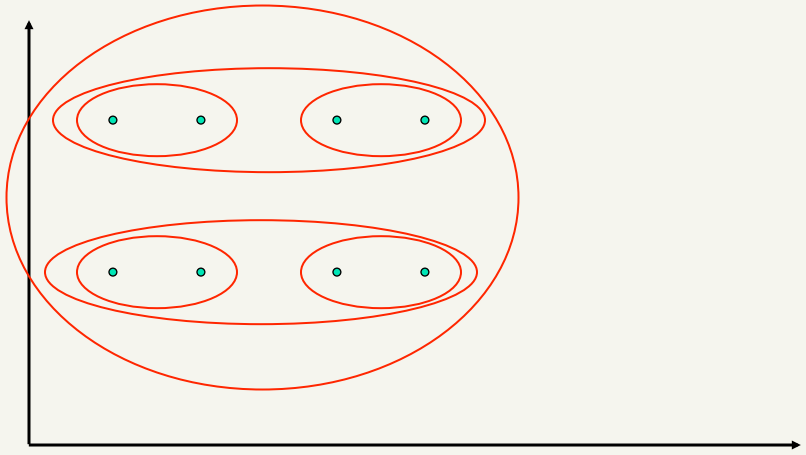
$$sim(c_i, c_j) = \max_{x \in c_i, y \in c_j} sim(x, y)$$

- Can result in "straggly" (long and thin) clusters due to chaining effect.

- After merging c_i and c_j, the similarity of the resulting cluster to another cluster, c_k, is:

$$sim((c_i \cup c_j), c_k) = \max(sim(c_i, c_k), sim(c_j, c_k))$$

Single Link Example



Complete Link Agglomerative Clustering

- Use minimum similarity of pairs:

$$sim(c_i, c_j) = \min_{x \in c_i, y \in c_j} sim(x, y)$$

- Makes "tighter," spherical clusters that are typically preferable.
- After merging c_i and c_j, the similarity of the resulting cluster to another cluster, c_k, is:

$$sim((c_i \cup c_j), c_k) = \min(sim(c_i, c_k), sim(c_j, c_k))$$

Complete Link Example