Today’s Outline

- Brief supervised learning review
- Evaluation
- Overfitting
- Ensembles
  Learners: The more the merrier
- Co-Training
  (Semi) Supervised learning with few labeled training ex
- Clustering
  No training examples

Types of Learning

- **Supervised (inductive) learning**
  Training data includes desired outputs
- **Semi-supervised learning**
  Training data includes a few desired outputs
- **Unsupervised learning**
  Training data does not include desired outputs
- **Reinforcement learning**
  Rewards from sequence of actions

Supervised Learning

- **Inductive learning** or “Prediction”:
  Given examples of a function \( (X, F(X)) \)
  Predict function \( F(X) \) for new examples \( X \)
- **Classification**
  \( F(X) = \) Discrete
- **Regression**
  \( F(X) = \) Continuous
- **Probability estimation**
  \( F(X) = \) Probability\( (X) \):
Classifier

Hypothesis: Function for labeling examples

Label: + + Label: -

Bias

• The nice word for prejudice is “bias”.
• What kind of hypotheses will you consider?
  What is allowable range of functions you use when approximating?
• What kind of hypotheses do you prefer?
• One idea: Prefer “simplest” hypothesis that is consistent with the data

Naïve Bayes

• Probabilistic classifier:
  \[ P(C_i | \text{Example}) \]
• Bias: Assumes all features are conditionally independent given class
  \[ P(E | C_i) = P(e_1 \land e_2 \land \cdots \land e_n | C_i) = \prod_{j=1}^{n} P(e_j | C_i) \]
• Therefore, we then only need to know \( P(e_j | C_i) \) for each feature and category

Naïve Bayes for Text

• Modeled as generating a bag of words for a document in a given category
• Assumes that word order is unimportant, only cares whether a word appears in the document
• Smooth probability estimates with Laplace m-estimates assuming a uniform distribution over all words \( (p = 1/|V|) \) and \( m = |V| \)
  Equivalent to a virtual sample of seeing each word in each category exactly once.
Naïve Bayes

Probability(Seat | County) = ??
Probability(Seat | Country) = ??

Population | Seat | Language | Class
--- | --- | --- | ---
Y | Y | N | County
Y | Y | Y | County
Y | N | Y | Country
N | N | Y | Country

Naïve Bayes

Probability(Seat | County) = $2 + \frac{1}{2} + 2 = 1.0$
Probability(Seat | Country) = ??

Population | Seat | Language | Class
--- | --- | --- | ---
Y | Y | N | County
Y | Y | Y | County
Y | N | Y | Country
N | N | Y | Country

Naïve Bayes

Probability(Seat | County) = $2 + \frac{1}{2} + 2 = 0.75$
Probability(Seat | Country) = $0 + \frac{1}{2} + 2 = 0.25$

Population | Seat | Language | Class
--- | --- | --- | ---
Y | Y | N | County
Y | Y | Y | County
Y | N | Y | Country
N | N | Y | Country

Naïve Bayes

Probabilities: Important Detail!

- $P(\text{spam} | E_1 \ldots E_n) = \prod_i P(\text{spam} | E_i)$

Any more potential problems here?

- We are multiplying lots of small numbers. Danger of underflow!
  - $0.5^{57} = 7 \times 10^{-18}$
- Solution? Use logs and add!
  - $p_1 \times p_2 = e^{\log(p_1) + \log(p_2)}$
  - Always keep in log form
Multi-Class Categorization

- Pick the category with max probability
- Create many 1 vs other classifiers
  
  Classes = City, County, Country

  Classifier 1 = \{City\} \{County, Country\}
  Classifier 2 = \{County\} \{City, Country\}
  Classifier 3 = \{Country\} \{City, County\}

Multi-Class Categorization

- Use a hierarchical approach (wherever hierarchy available)

Today's Outline

- Brief supervised learning review
- Evaluation
- Overfitting
- Ensembles
  - Learners: The more the merrier
- Co-Training
  - (Semi) Supervised learning with few labeled training ex
- Clustering
  - No training examples

Experimental Evaluation

Question: How do we estimate the performance of classifier on unseen data?

- Can't just at accuracy on training data - this will yield an over optimistic estimate of performance
- Solution: Cross-validation
- Note: this is sometimes called estimating how well the classifier will generalize
Evaluation: Cross Validation

• Partition examples into k disjoint sets
• Now create k training sets
    Each set is union of all equiv classes except one
    So each set has (k-1)/k of the original training data

Cross-Validation (2)

• Leave-one-out
    Use if < 100 examples (rough estimate)
    Hold out one example, train on remaining examples

• 10-fold
    If have 100-1000's of examples

• M of N fold
    Repeat M times
    Divide data into N folds, do N fold cross-validation

Today's Outline

• Brief supervised learning review
• Evaluation
• Overfitting
• Ensembles
    Learners: The more the merrier
• Co-Training
    (Semi) Supervised learning with few labeled training ex
• Clustering
    No training examples

Overfitting Definition

• Hypothesis H is overfit when \( \exists \) H' and
    H has smaller error on training examples, but
    H has bigger error on test examples

• Causes of overfitting
    Noisy data, or
    Training set is too small
    Large number of features

• Big problem in machine learning
• One solution: Validation set
Overfitting

Accuracy

On training data
On test data

Model complexity (e.g., number of nodes in decision tree)

Validation/Tuning Set

- Split data into train and validation set
- Score each model on the tuning set, use it to pick the 'best' model

Early Stopping

Accuracy

Remember this and use it as the final classifier

Model complexity (e.g., number of nodes in decision tree)

Extra Credit Ideas

- Different types of models
  - Support Vector Machines (SVMs), widely used in web search
  - Tree-augmented naive Bayes
  - Feature construction
Support Vector Machines

Which one is best hypothesis?

Largest distance to neighboring data points

SVMs in Weka: SMO

Tree Augmented Naive Bayes (TAN)

[Friedman, Geiger & Goldszmidt 1997]

Class Node

F1 F2 F3 F N-2 F N-1 F N

Models limited set of dependencies
Guaranteed to find best structure
Runs in polynomial time

Construct Better Features

• Key to machine learning is having good features
• In industrial data mining, large effort devoted to constructing appropriate features
• Ideas??
Possible Feature Ideas

• Look at capitalization (may indicated a proper noun)

• Look for commonly occurring sequences
  • E.g. New York, New York City
  • Limit to 2-3 consecutive words
  • Keep all that meet minimum threshold (e.g. occur at least 5 or 10 times in corpus)

Today’s Outline

• Brief supervised learning review
• Evaluation
• Overfitting
• Ensembles
  Learners: The more the merrier
• Co-Training
  (Semi) Supervised learning with few labeled training ex
• Clustering
  No training examples

Ensembles of Classifiers

• Traditional approach: Use one classifier
• Alternative approach: Use lots of classifiers
• Approaches:
  • Cross-validated committees
  • Bagging
  • Boosting
  • Stacking

Voting
Ensembles of Classifiers

- Assume
  Errors are independent (suppose 30% error)
  Majority vote
- Probability that majority is wrong...
  \[ \text{Area under binomial distribution} \]
- If individual area is 0.3
  Area under curve for \( \geq 11 \) wrong is 0.026
  Order of magnitude improvement!

Constructing Ensembles

Cross-validated committees

- Partition examples into \( k \) disjoint equiv classes
- Now create \( k \) training sets
  Each set is union of all equiv classes except one
  So each set has \((k-1)/k\) of the original training data
- Now train a classifier on each set

Ensemble Construction II

Bagging

- Generate \( k \) sets of training examples
- For each set
  Draw \( m \) examples randomly (with replacement)
  From the original set of \( m \) examples
- Each training set corresponds to
  63.2% of original (+ duplicates)
- Now train classifier on each set
- Intuition: Sampling helps algorithm become more robust to noise/outliers in the data

Ensemble Creation III

Boosting

- Maintain prob distribution over set of training ex
- Create \( k \) sets of training data iteratively:
  On iteration \( i \)
    - Draw \( m \) examples randomly (like bagging)
    - But use probability distribution to bias selection
    - Train classifier number \( i \) on this training set
    - Test partial ensemble (of \( i \) classifiers) on all training exs
    - Modify distribution: increase \( P \) of each error ex
- Create harder and harder learning problems...
  "Bagging with optimized choice of examples"
**Ensemble Creation IV**

**Stacking**

- Train several base learners
- Next train meta-learner
  - Learns when base learners are right / wrong
  - Now meta learner arbitrates

Train using cross validated committees
- Meta-L inputs = base learner predictions
- Training examples = 'test set' from cross validation

---

**Today's Outline**

- Overfitting
- Ensembles
  - Learners: The more the merrier
- Co-Training
  - Supervised learning with few labeled training ex
- Clustering
  - No training examples

---

**Co-Training Motivation**

- Learning methods need labeled data
  - Lots of \( \langle x, f(x) \rangle \) pairs
  - Hard to get... (who wants to label data?)
- But unlabeled data is usually plentiful...
  - Could we use this instead??????
- Semi-supervised learning
Co-training

Suppose
• Have little labeled data + lots of unlabeled

• Each instance has two parts:
  \( x = [x_1, x_2] \)
  \( x_1, x_2 \) conditionally independent given \( f(x) \)

• Each half can be used to classify instance
  \( \exists f_1, f_2 \) such that \( f_1(x_1) \sim f_2(x_2) \sim f(x) \)

• Both \( f_1, f_2 \) are learnable
  \( f_1 \in H_1, f_2 \in H_2, \exists \) learning algorithms \( A_1, A_2 \)

Without Co-training

A Few Labeled Instances

\( \langle x_1, x_2 \rangle, f() \)

\( f_1(x_1) \sim f_2(x_2) \sim f(x) \)

A_1 learns \( f_1 \) from \( x_1 \)

A_2 learns \( f_2 \) from \( x_2 \)

Combine with ensemble? [x1, x2]

Lots of Labeled Instances

A Few Labeled Instances

\( \langle x_1, x_2 \rangle, f() \)

\( f_1(x_1) \sim f_2(x_2) \sim f(x) \)

A_1 learns \( f_1 \) from \( x_1 \)

A_2 learns \( f_2 \) from \( x_2 \)

Hypothesis

[Co-training Example]
Observations

• Can apply $A_1$ to generate as much training data as one wants
  
  If $x_1$ is conditionally independent of $x_2 / f(x)$, 
  
  then the error in the labels produced by $A_1$ 
  
  will look like random noise to $A_2$ III!

• Thus no limit to quality of the hypothesis $A_2$ 
  
  can make

Co-training

Lots of Labeled Instances $\langle x_1, x_2 \rangle, f(x)$

A_1 learns $f_1$ from $x_1$

A_2 learns $f_2$ from $x_2$

Lots of Labeled Instances

Unlabeled Instances $\langle x_1, x_2 \rangle$

Hypothesis $f_2(x_2) \sim f(x)$

It really works!

• Learning to classify web pages as course pages
  
  $x_1 =$ bag of words on a page
  
  $x_2 =$ bag of words from all anchors pointing to a page

• Naïve Bayes classifiers
  
  12 labeled pages
  
  1039 unlabeled

<table>
<thead>
<tr>
<th>Supervised training</th>
<th>UC</th>
<th>ITL</th>
<th>UI</th>
</tr>
</thead>
<tbody>
<tr>
<td>12 quitting learning</td>
<td>12%</td>
<td>12%</td>
<td>12%</td>
</tr>
</tbody>
</table>

Table 2: Error rate in percent for classifying web pages as course home pages. The top row shows errors when training on only the labeled examples. Bottom row shows errors when co-training, using both labeled and unlabeled examples.

Today's Outline

• Brief supervised learning review
• Evaluation
• Overfitting
• Ensembles
  
  Learners: The more the merrier

• Co-Training
  
  (Semi) Supervised learning with few labeled training ex

• Clustering
  
  No training examples
**Clustering Outline**

- Motivation
- Document Clustering
- Offline evaluation
- Grouper I
- Grouper II
- Evaluation of deployed systems

**Low Quality of Web Searches**

- **System perspective:**
  - small coverage of Web (<16%)
  - dead links and out of date pages
  - limited resources
- **IR perspective**
  - (relevancy of doc ~ similarity to query):
    - very short queries
    - huge database
    - novice users

**Document Clustering**

- User receives many (200 - 5000) documents from Web search engine
- Group documents in clusters by topic
- Present clusters as interface
Grouper

A document clustering interface for BullySearch

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Size</th>
<th>Shared Phrases and Sample Document Titles</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 View Results</td>
<td>37</td>
<td>Monica Lewinsky (50%), Clinton's scandal (50%), Kenneth Starr Investigation (48%), Hillary Clinton (48%)</td>
</tr>
<tr>
<td>2 View Results</td>
<td>20</td>
<td>Clinton is a positive or negative (49%), Clinton/Cure (39%), Presidential Election (37%), election of (37%)</td>
</tr>
<tr>
<td>3 View Results</td>
<td>8</td>
<td>Monica Lewinsky's (49%), documents (48%), powerful (48%), Perriello (37%), Report (37%), legal (37%), Paula (37%)</td>
</tr>
</tbody>
</table>
Desiderata

- Coherent cluster
- Speed
-Browsable clusters

Naming

Main Questions

- Is document clustering feasible for Web search engines?
- Will the use of phrases help in achieving high quality clusters?
- Can phrase-based clustering be done quickly?
1. Clustering

Group together similar items (words or documents)

Clustering Algorithms

• Hierarchical Agglomerative Clustering $O(n^2)$
• Linear-time algorithms
  K-means (Rocchio, 66)
  Single-Pass (Hill, 68)
  Fractionation (Cutting et al, 92)
  Buckshot (Cutting et al, 92)

Basic Concepts - 1

• Hierarchical vs. Flat

Basic Concepts - 2

• Hard clustering:
  Each item in only one cluster
• Soft clustering:
  Each item has a probability of membership in each cluster
• Disjunctive / Overlapping clustering:
  An item can be in more than one cluster
Basic Concepts - 3

- distance / similarity function
  (for documents)
  - dot product of vectors
  - number of common terms
  - co-citations
  - access statistics
  - share common phrases

Basic Concepts - 4

- What is “right” number of clusters?
  - apriori knowledge
  - default value: “5”
  - clusters up to 20% of collection size
  - choose best based on external criteria
    - Minimum Description Length
    - Global Quality Function
  - no good answer

K-means

- Works when we know k, the number of clusters

  Idea:
  - Randomly pick k points as the “centroids” of the k clusters
  Loop:
  - ∀ points, add to cluster w/ nearest centroid
  - Recompute the cluster centroids
  - Repeat loop (until no change)

Iterative improvement of the objective function:
  Sum of the squared distance from each point
to the centroid of its cluster

K-means Example

- For simplicity, 1-dimension objects and k=2.
  - Numerical difference is used as the distance
- Objects: 1, 2, 5, 6, 7
- K-means:
  - Randomly select 5 and 6 as centroids;
    - Two clusters (1,2,5) and (6,7); meanC1=8/3, meanC2=6.5
    - (1,2), (5,6,7); meanC1=1.5, meanC2=6
    - no change.
  Aggregate dissimilarity:
    - (sum of squares distance each point of each cluster from its cluster center—(int intra-cluster distance)
      = 0.5^2 + 0.5^2 + 1^2 + 0^2 + 1^2 = 2.5
      \[(|1-1.5|)^2\]
**K Means Example (K=2)**

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

---

**Time Complexity**

- Assume computing distance between two instances is $O(m)$ where $m$ is the dimensionality of the vectors.
- Reassigning clusters: $O(kn)$ distance computations, or $O(knm)$.
- Computing centroids: Each instance vector gets added once to some centroid: $O(nm)$.
- Assume these two steps are each done once for $I$ iterations: $O(Im)$.
- Linear in all relevant factors, assuming a fixed number of iterations, more efficient than $O(n^2)$ HAC (to come next)

---

**Vector Quantization: K-means as Compression**
Problems with K-means

- Need to know k in advance
  - Could try out several k?
- Cluster tightness increases with increasing K
  - Look for a kink in the tightness vs. K curve
- Tends to go to local minima that are sensitive to the starting centroids
  - Try out multiple starting points
- Disjoint and exhaustive
  - Doesn’t have a notion of “outliers”
    - Outlier problem can be handled by K-medoid or neighborhood-based algorithms
- Assumes clusters are spherical in vector space
  - Sensitive to coordinate changes, weighting etc.

Hierarchical Clustering

- Agglomerative
  - Bottom-up
    - Initialize: each item a cluster
    - Iterate:
      - select two most similar clusters
      - merge them
    - Halt: when have required # of clusters

- Divisive
  - Top-bottom
    - Initialize: all items one cluster
    - Iterate:
      - select a cluster (least coherent)
      - divide it into two clusters
    - Halt: when have required # of clusters

Example showing sensitivity to seeds:

Why not the minimum value?

In the above, if you start with B and E as centroids you converge to \( \{A, B, C\} \) and \( \{D, E, F\} \)
If you start with D and F you converge to \( \{A, B, D, E\} \) and \( \{C, F\} \)
HAC Similarity Measures

- Single link
- Complete link
- Group average
- Ward's method

Single Link

- Cluster similarity = similarity of two most similar members

Single Link

- $O(n^2)$
- Chaining:
  - Bottom line: simple, fast, often low quality
Complete Link

- cluster similarity = similarity of two least similar members

- worst case $O(n^3)$
- fast algo requires $O(n^2)$ space
- no chaining
- bottom line: typically much faster than $O(n^3)$, often good quality

Group Average

- cluster similarity = average similarity of all pairs

HAC Often Poor Results - Why?

- Often produces single large cluster
- Work best for:
  - spherical clusters; equal size; few outliers
- Text documents:
  - no model
  - not spherical; not equal size; overlap
- Web:
  - many outliers; lots of noise
Example: Clusters of Varied Sizes

k-means; complete-link; group-average:

single-link: chaining, but succeeds on this example

Example - Outliers

HAC:

Suffix Tree Clustering (KDD'97; SIGIR'98)

- Most clustering algorithms aren't specialized for text:
  Model document as set of words
- STC:
  document = sequence of words

STC Characteristics

- Coherent
  phrase-based overlapping clusters
- Speed and Scalability
  linear time; incremental
- Browsable clusters
  phrase-based simple cluster definition
**STC - Central Idea**

- **Identify base clusters**
  - a group of documents that share a phrase
  - use a suffix tree

- **Merge base clusters as needed**

**STC - Outline**

Three logical steps:

1. "Clean" documents
2. Use a suffix tree to identify base clusters - a group of documents that share a phrase
3. Merge base clusters to form clusters

**Step 1 - Document “Cleaning”**

- Identify sentence boundaries
- Remove
  - HTML tags,
  - JavaScript,
  - Numbers,
  - Punctuation

**Suffix Tree**

(Weiner, 73; Ukkonen, 95; Gusfield, 97)

Example - suffix tree of the string: (1) "cats eat cheese"
Example - suffix tree of the strings:
(1) "cats eat cheese",
(2) "mice eat cheese too" and
(3) "cats eat mice too"

Step 2 - Identify Base Clusters via Suffix Tree
- Build one suffix tree from all sentences of all documents
- Suffix tree node = base cluster
- Score all nodes
- Traverse tree and collect top k (500) base clusters

Step 3 - Merging Base Clusters
- Motivation: similar documents share multiple phrases
- Merge base clusters based on the overlap of their document sets
- Example (query: "salsa")
  "tabasco sauce" docs: 3, 4, 5, 6
  "hot pepper" docs: 1, 3, 5, 6
  "dance" docs: 1, 2, 7
  "latin music" docs: 1, 7, 8

Average Precision - WSR-SNIP
16% increase over k-means (not stat. sig.)
### Average Precision - WSR-DOCS

- **Graph:**
  - Y-axis: Average precision
  - X-axis: Method (random, fractionation, GAVG, Buckshot, STC, k-means)

- **Results:**
  - **45% increase over k-means (stat. sig.)**

### Grouper II

- **Dynamic Index:**
  - Non-merged based clusters

- **Multiple interfaces:**
  - List, Clusters + Dynamic Index (key phrases)

- **Hierarchical:**
  - Interactive "Zoom In" feature (similar to Scatter/Gather)

### Evaluation - Log Analysis

- **Graph:**
  - X-axis: Number of document followed
  - Y-axis: Number of clusters followed

- **Data:**
  - Group 1
  - Group 2
Northern Light

- “Custom Folders”
- 20000 predefined topics in a manually developed hierarchy
- Classify document into topics
- Display “dominant” topics in search results

Summary

- Post-retrieval clustering to address low precision of Web searches
- STC phrase-based; overlapping clusters; fast
- Offline evaluation
  Quality of STC, advantages of using phrases vs. n-grams, FS
- Deployed two systems on the Web
  Log analysis: Promising initial results

Cool Topic

- Internet allows creation of knowledge Aka structured data
- Two dominant techniques
  ML-based information extraction
  - Google scholar, product search
  - Zoominfo
  - Flipdog
- Collaborative content authoring
  - Wikipedia
  - Summitpost
  - Amazon reviews (and votes on usefulness)
- How integrate the techniques?
Integration Today

• ML first – creates a seed to attract users
• Humans act as proofreaders

Zoominfo
Zillow zestimate
dblife.cs.wisc.edu

• Surely we can do better than this!?}

Total > Sum of Parts

• Human corrections → training data → improved ML output
• Active learning to prioritize corrections
• Track author (and ML extractor) reputations
  Learn policy where ML can overwrite human
• Insert javascript code to encourage human fact checking
• Realtime-ML to create “author helper”
How Does this Fit In?