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
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

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

On training data
On test data
On validation data

Remember this and use it as the final classifier

Accuracy

0.9
0.8
0.7
0.6
Support Vector Machines

Which one is best hypothesis?
Support Vector Machines

Largest distance to neighboring data points

SVMs in Weka: SMO
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 indicate 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)
Properties of Text

- Word frequencies - skewed distribution
- `The’ and `of’ account for 10% of all words
- Six most common words account for 40%

Zipf’s Law:

Rank * probability = c
Eg, c = 0.1

From [Croft, Metzler & Strohman 2010]
Associate Press Corpus `AP89`

From [Croft, Metzler & Strohman 2010]
Middle Ground

- Very common words $\rightarrow$ bad features
- Language-based **stop list**: words that bear little meaning
  20-500 words
  http://www.dcs.gla.ac.uk/idom/ir_resources/linguistic_utils/stop_words
- Subject-dependent stop lists
- Very rare words *also* bad features
  Drop words appearing less than k times / corpus
Stop lists

• **Language-based stop list:**
  words that bear little meaning
  20-500 words
  http://www.dcs.gla.ac.uk/idom/ir_resources/linguistic_utils/stop_words

• **Subject-dependent stop lists**
Stemming

- Are there different index terms?
  
  retrieve, retrieving, retrieval, retrieved, retrieves...

- Stemming algorithm:
  
  (retrieve, retrieving, retrieval, retrieved, retrieves) $\Rightarrow$ retriev

  Strips prefixes of suffixes (-s, -ed, -ly, -ness)

  Morphological stemming
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
Ensembles of Classifiers

- Traditional approach: Use one classifier
- Alternative approach: Use lots of classifiers

Approaches:
- Cross-validated committees
- Bagging
- Boosting
- Stacking
Ensembles of Classifiers

• Assume
  Errors are independent (suppose 30% error)
  Majority vote

• Probability that majority is wrong...
  = 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 \textit{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

• Brief supervised learning review
• Evaluation
• Overfitting
• Ensembles
  Learners: The more the merrier
• Co-Training
  (Semi) Supervised learning with few labeled training ex
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 *doesn’t* include desired outputs

• **Reinforcement learning**
  Rewards from sequence of actions
Co-Training  Motivation

• Learning methods need labeled data
  Lots of $<x, f(x)>$ 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:
  \[ x = [x_1, x_2] \]
  \[ x_1, x_2 \text{ conditionally independent given } f(x) \]

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

• Both \( f_1, f_2 \) are learnable
  \[ f_1 \in H_1, \quad f_2 \in H_2, \quad \exists \text{ learning algorithms } A_1, A_2 \]
Co-training Example

Prof. Domingos
Students: Parag,…
Projects: SRL, Data mining
I teach a class on data mining

CSE 546: Data Mining
Course Description:…
Topics:…
Homework: …

Jesse
Research: SRL

© Daniel S. Weld
Without Co-training

A Few Labeled Instances

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

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

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

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

[\( x_1, x_2 \)]

Bad!! Not using Unlabeled Instances!

Combine with ensemble?
Co-training

A Few Labeled Instances

<[x_1, x_2], f()> 

A_1 learns f_1 from x_1
A_2 learns f_2 from x_2

Unlabeled Instances

[x_1, x_2]

Lots of Labeled Instances

<[x_1, x_2], f_1(x_1)> 

Hypothesis

f_1(x_1) \sim f_2(x_2) \sim f(x)
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 $A_2$ !!!

- Thus *no limit* to quality of the hypothesis $A_2$ can make
Co-training

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

A_1 learns \( f_1 \) from \( x_1 \)
A_2 learns \( f_2 \) from \( x_2 \)

Lots of Labeled Instances

\( [<x_1, x_2], f() > \)

Unlabeled Instances

\( [x_1, x_2] \)

Lots of Labeled Instances

\( [<x_1, x_2], f_1(x_1)> \)

Hypothesis

\( f_{f_2} \)
It really works!

• Learning to classify web pages as course pages
  \[ x_1 = \text{bag of words on a page} \]
  \[ x_2 = \text{bag of words from all anchors pointing to a page} \]

• Naïve Bayes classifiers
  12 labeled pages
  1039 unlabeled

<table>
<thead>
<tr>
<th></th>
<th>Page-based classifier</th>
<th>Hyperlink-based classifier</th>
<th>Combined classifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supervised training</td>
<td>12.9</td>
<td>12.4</td>
<td>11.1</td>
</tr>
<tr>
<td>Co-training</td>
<td>6.2</td>
<td>11.6</td>
<td>5.0</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.
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 *doesn’t* include desired outputs