CS 175, Project in Artificial Intelligence

Lecture 9: Progress Reports, Presentations, and Evaluation Methods

Padhraic Smyth
Department of Computer Science
Bren School of Information and Computer Sciences
University of California, Irvine
### Weekly Schedule

<table>
<thead>
<tr>
<th>Week</th>
<th>Monday</th>
<th>Wednesday</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feb 27</td>
<td>Lecture: Progress Reports, etc</td>
<td>Office hours (no lecture)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Progress Report due by 11pm</td>
</tr>
<tr>
<td>Mar 6</td>
<td>Project Presentations (in class)</td>
<td>Project Presentations (in class)</td>
</tr>
<tr>
<td></td>
<td>submit slides by 1pm</td>
<td>submit slides by 1pm</td>
</tr>
<tr>
<td>Mar 13</td>
<td>Short Lecture on Final Project Reports</td>
<td>No lecture or office hours</td>
</tr>
<tr>
<td>Mar 20</td>
<td>Final Report due to EEE by Tuesday March 21st 11pm</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** no final exam, just submit your final report by Tuesday night of finals week
Progress Reports

• Your proposal should be up to 4 pages long
  – Use the progress report template
  – Due to EEE by 11pm this Wednesday

• Reports will be graded and receive a weight of 20% of your overall grade.

• Proposals will primarily be graded on
  (a) clarity: are the technical ideas, experiments, etc., explained clearly?
  (b) completeness: are all sections covered?
  (c) creativity: have you added any of your own ideas to the project?
  (c) progress: how much progress has there been?

• One report per team: both students will get the same grade by default

• Tips on writing proposals (Lecture 6) are still highly relevant – see also samples of past proposals distributed via Piazza
Project Slide Presentations

• In-class, Monday and Wednesday next week

• Each student or team will make one presentation
  – Approximately 4 minutes per presentation

• Time after each presentation for questions and changeover
  – So about 6 minutes x 11 projects per day = 66 minutes total

• List and order of presentations will be announced this Wednesday (by email and on Piazza)

• Slides need to be uploaded to the EEE dropbox by 1pm the day of your presentation:
  – they will be loaded on to the classroom machine, no need to use your laptop.
  – Either PDF or Powerpoint is fine
Suggestions for Slide Content

• Slide 0: project name, student name(s)

• Slide 1: overview of project, e.g.,
  – Investigation of sentiment analysis algorithms for Twitter
  – Focus on classifying Tweets into 3 categories, ..... 
  – Using the following data set:....
  – Using the following classifiers:....
  – Evaluation methods: ....
  [Note: you could just show a figure here, e.g., an example of a tweet or a summary of the data set, and make your points using the figure]

• Slide 2:
  – Some more details on your technical approach
  – figures are good!, e.g., a block diagram of your pipeline
Suggestions for Slide Content (continued)

• Slide 3:
  – Examples of initial results

• Slide 4 (optional)
  – Challenges you have encountered

• Slide 5:
  – Plans for the remainder of the quarter
  – E.g., milestones you plan to complete
  – E.g., optional additional item you will investigate if there is time
Twitter Sentiment Analysis

Used Sentiment140 tweets DB

Preprocessing:
- Removed and replaced URLs, usernames, and hashtags with `<URL>`, `<USER>`, `<HASHTAG>`
- Omitted neutral sentiment tweets from test sets

We found that the SVM classifier had the most growth potential in accuracy.

The SVM classifier without the data preprocessed had an accuracy of 79%. After preprocessing the data, it became 4% more accurate. We believe this is because of the reduction of features, which made the classifier easier to classify the data.

We plan to further investigate and experiment with more feature sets.
It's interesting that training data with stop words predicts test data better training data without stop words. The reason maybe stop words connect sentences to form the meaning of the whole review.
Accuracy with other categories of tweets as negative examples for sarcasm.

<table>
<thead>
<tr>
<th>Category</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Happy</td>
<td>0.8594313175810974</td>
</tr>
<tr>
<td>Sad</td>
<td>0.7597116539847817</td>
</tr>
<tr>
<td>Fearful</td>
<td>0.8157366519470093</td>
</tr>
<tr>
<td>Courageous</td>
<td>0.912745545911375</td>
</tr>
<tr>
<td>Sincere</td>
<td>0.759090909090909</td>
</tr>
<tr>
<td>Relaxed</td>
<td>0.8740629685157422</td>
</tr>
<tr>
<td>Stressed</td>
<td>0.8192771084337349</td>
</tr>
</tbody>
</table>

Table Legend:
Left: Label of the tweets that are mixed in with sarcastic tweets

Right: Accuracy of the classifier with this particular mix of tweets.

Not bad for sarcastic!
Rotten Tomatoes Movie Review Classification With Machine Learning and Natural Language Processing

<table>
<thead>
<tr>
<th></th>
<th>Overall Error</th>
<th>False Positive Error</th>
<th>False Negative Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logistic Regression</td>
<td>12.38%</td>
<td>16.58%</td>
<td>9.54%</td>
</tr>
<tr>
<td>Multinomial Naive Bayes</td>
<td>13.62%</td>
<td>18.55%</td>
<td>9.59%</td>
</tr>
</tbody>
</table>

Figure: Error Rate Results - Binary Classification of Positive vs Negative Phrases

- How we conduct the binary classification: We removed phrases labeled 2, which is neutral. Then, we combined 0 and 1 into one class “Negative”, which we relabeled as 0, and 4 and 5 into one class “Positive”, which we relabeled as 1.

- We were surprised at how well the classifier was able to distinguish between positive and negative phrases. We are looking into creating a multi-level classifier that would try to classify whether a phrase is neutral or not, and then using this to determine whether that phrase is actually positive or negative.
Lessons learned along the way...

Preprocessing works:

The Logistic Regression classifier correctly classified 70.19% of the Reuters test set in a bag of words representation, and 73.61% on the lemmatized version. Lemmatizing provided a 3.42% increase in accuracy, which lead to 112 more documents being properly classified.

Cross-validate first:

After weeks of testing the Support Vector Machine classifier on only the standard train/test splits, we finally ran cross-validation and were surprised to find that the SVM was significantly less accurate than we thought. For example, it is producing 68.1% accuracy on the Ohsumed 5-category standard split, but only 59.74% in cross-validation on the same data. It is actually being outperformed by LR!

Experiment a little:

Using the Multinomial Naive Bayes classifier, an attempt was made to reduce the number of categories used to train the classifier for each document. The process produced less accuracy but this only confirmed that our baselines for each experiment were in fact the greatest lower bounds.
“aww! im so happy cuz you are happy 8D i have to go goodnight girl!! you are AMAZING! have a nice night! loveya?”

- Naïve bayes probability .93
- SVM distance 1.75

“I hate the way pills make my stomach feel like it's boiling. This is worse than being sick. Also”

- Naïve bayes probability .97
- SVM distance -2.10

The above are two tweets that both classifiers were very confident about. Both were mislabeled by the user. The first was incorrectly labeled by the user as negative and the second was incorrectly labeled as positive. The tweets sentiment however is obviously the opposite. We have found at least a few tweets with this problem
Twitter Trend Detection

- While many claim social media to be negative (even toxic), the top words are often positive!
- "love", "like", "good", "please", "best"
- Negative words and obscenities are further down.
- Twitter isn’t as bad as you think ;)}
Parsing Difficulties

- We are using the Stanford parser to generate parse trees for our PCFG. For complex sentences, the parser often finds sentences within sentences. As a result, we generate sentences with strange punctuation (periods in the middle, comma at the end, etc.).
- Also, some complex sentences are parsed as NP (noun phrase) rather than S (sentence). As a result, some of our generated sentences are just single noun phrases, and some noun phrases become complex sentences.

Bad Generated Tree Example:

```
ROOT
  ↓
  S
  ↓
  NP   VP
  ↓
  NP   SBAR ...
  ↓
  ... S
  ↓
  NP   VP ...
  ↓
  ... ...
```

- Solution: sample the generated RHS using both the symbol and its direct parent. This lets us tell the difference between a sentence/NP at the root vs. in the middle of the tree.
- Similar to using a higher-order Markov chain
- Examples of valid expansions:
  - (ROOT -> S) -> NP VP.
  - (SBAR -> S) -> NP VP
  - (ROOT -> NP) -> NP : S.
  - (S -> NP) -> DT NN
Poetry Generator - Interesting Find

- We are using a Markov generator to produce our initial sentences (sentences that have not gone through our poetry formatting yet).
- When using a Dr. Seuss data set, the following was yielded:

  Congratulations!
  Today is your day!
  Your mountain is waiting.
  So...get on your own.
  And when you're in a Lurch.
  You'll come down from the Lurch with an unpleasant bump.
  And then things start to happen, don't worry.
  Don't stew.

- It's pretty short, but this is to be expected due to the end character delimiter paring with the short sentences found in Dr. Seuss...
But What We Didn’t Expect...

- The *adverse* effect is the potential of a few large sentences causing sentences to become freakishly long more often than we want.
- For example, this is from the text Jeeves:

  Jeeves my man, you know what I subsequently learned was Madison Square Garden, where Mr. Pepper?" "Yes.

  I proposed to her at lunch one Sunday before I knew squads of chappies down Washington Square way who started the evening papers as the rest of the zest with which she had lost a bit of lunch with me to broach the subject.

  George’s uncle was in your way, Mr.

  Sturgis has offered his services to his expressed intention of remaining in the drawing-room in my autobiography.

- So, need a word count MAX *and* MIN, otherwise too much stress on poetry generator to find syllable count and delete words.
wine for cats. Because every girl
dreams of hearing her man say, 'nice
!!', for already getting

Thank you guys like having @
StephenAtHome and climbing a guy
raised $20 min: Man never heard
even realize they love you to the
photobomb with @

# WhyIMSingle tweets helped #
LateNight # FunShow http://t . I spent
the return of The Year # LateNight #
VladdyPootPoot ', b ' l . Ha !!! London
# funtimes # BlYM # FallonTonight ', b

getting old, but whee did it again! #
BurgerSummit # nomayo '', b ' Arnold @
Schwarzenegger showed me some of his other
reviews were bad.

need a little help Obama \xe2 \x9d l ( ft . Any opportunity to have no that guy
for the show about this many people
wearing a werewolf

he was arrested for stopping by the
day? Keep tweeting the man say it
was throwing shade last night at 10
minutes . co / Ux3WCDbJG2 ', b '
Thanks Jon every post - Aime with a
baby carrots
Evaluating Clustering or Topic Models
Evaluating Algorithms without Ground Truth

- Ground truth = “gold standard” (e.g., human-generated class labels)

- Evaluation of unsupervised learning is difficult
  - If reference/ground truth exists, this may be helpful, but it does not necessarily correspond with human judgement

- For some problems, like summarization, there are techniques (such as the ROUGE metric) that have been developed

- In general you will likely need to do some research and reading
  - E.g., “cluster evaluation” or “text summarization evaluation”
Finding Words that Characterize Clusters

• Say we have a clustering algorithm that produces say K=20 clusters

• To summarize these clusters for human interpretation we need to find the top words for each cluster, e.g.,
  – Find all the documents for a cluster
  – List the top 10 most common words in the cluster?
  – ...but this might not be ideal – why?
Finding Words that Characterize Clusters

• Instead we can try to find the most “discriminative” words for a cluster
  – i.e., the words that are most distinctive for this cluster versus the others

• One simple way to do this is to train a supervised classifier (such as logistic regression) to learn how to classify documents in this cluster versus all the other clusters (as one large class).

• The words with high positive weights (see Assignment 2) are the words that are most discriminative for this cluster
Clusters and their Top Words

• We can now find the top 5 or 10 (for example) most discriminative words that characterize each cluster

• E.g.,
  – Cluster 1: dog, cat, pet, animal, ....
  – Cluster 2: food, restaurant, menu, service,...
  – Cluster 3: salary, income, check, tax,....

• Can we evaluate how good these clusters in terms of human judgement?
User Studies with Word Intrusion

<table>
<thead>
<tr>
<th></th>
<th>Word 1</th>
<th>Word 2</th>
<th>Word 3</th>
<th>Word 4</th>
<th>User 1</th>
<th>User 2</th>
<th>User 3</th>
<th>User 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster 1</td>
<td>Apple</td>
<td>Orange</td>
<td>Banana</td>
<td>Melon</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cluster 2</td>
<td>Job</td>
<td>Salary</td>
<td>Office</td>
<td>Tax</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cluster 3</td>
<td>Food</td>
<td>Menu</td>
<td>Happy</td>
<td>Car</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Say we have 3 clusters and these are the 4 best words for each
## User Studies with Word Intrusion

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Word 1</th>
<th>Word 2</th>
<th>Word 3</th>
<th>Word 4</th>
<th>User 1</th>
<th>User 2</th>
<th>User 3</th>
<th>User 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster 1</td>
<td>Apple</td>
<td><strong>Cat</strong></td>
<td>Banana</td>
<td>Melon</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cluster 2</td>
<td>Job</td>
<td>Salary</td>
<td><strong>France</strong></td>
<td>Tax</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cluster 3</td>
<td><strong>Truck</strong></td>
<td>Menu</td>
<td>Happy</td>
<td>Car</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Now say we introduce one random common word, in a random position, per cluster.
## User Studies with Word Intrusion

<table>
<thead>
<tr>
<th>Cluster 1</th>
<th>Word 1</th>
<th>Word 2</th>
<th>Word 3</th>
<th>Word 4</th>
<th>User 1</th>
<th>User 2</th>
<th>User 3</th>
<th>User 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster 1</td>
<td>Apple</td>
<td><strong>Cat</strong></td>
<td>Banana</td>
<td>Melon</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Cluster 2</td>
<td>Job</td>
<td>Salary</td>
<td><strong>France</strong></td>
<td>Tax</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Cluster 3</td>
<td><strong>Happy</strong></td>
<td>Menu</td>
<td><strong>Happy</strong></td>
<td>Car</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Indicates if the user detected the true intruder word or not

We can use the user detections to indicate if
(a) the clusters generally make sense (high detection rate)
(b) if there are particular clusters that don’t make sense
User Studies in General

• Can be used to compare outputs of Algorithm A and Algorithm B

• Humans are good at indicating preferences (e.g., A is better than B or vice-versa) – not so good at providing numbers

• Ideally the study needs to be replicated across multiple users (since individual users may be biased, unreliable, etc)

• For example
  – Two summarization algorithms A and B
  – Each user is presented with output from A and output from B
    • Important that the user does not know which is which (“blind test”)
    • Original input (text) could also be provided for context
  – This can be done on paper (printouts) or with a Web form (easier for analysis)
  – Each user gets K pairs of A and B outputs and states K preferences
  – N users, K pairs -> larger values of N and K are better (e.g., N>5, K>10)
Example of Data From a User Study

Say 5 users each compared the outputs of A and B 20 times

<table>
<thead>
<tr>
<th></th>
<th>Number of Times A was preferred</th>
<th>Number of Times B was preferred</th>
<th>Difference between A and B</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 1</td>
<td>12</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>User 2</td>
<td>15</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>User 3</td>
<td>9</td>
<td>11</td>
<td>-2</td>
</tr>
<tr>
<td>User 4</td>
<td>11</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>User 5</td>
<td>12</td>
<td>8</td>
<td>4</td>
</tr>
</tbody>
</table>

Can use Wilcoxon signed test (for example) to analyze the differences statistically.
Evaluating Classification Algorithms
Reporting Classification Accuracy

- Always report the performance of a simple baseline, e.g.,
  - The majority classifier (see next slide)
  - KNN classifier with $K = 1$

- Cross-validation is useful
  - But if you have a very large amount of data a single randomly selected train/test split of the data should be fine
The Simple Majority Classifier

• Say we have K classes, and class k occurs with probability p(k), k = 1,...K

• Let k* be the most likely class, i.e., highest value of p(k)

• The “majority classifier” always predicts class k*
  – What is the accuracy of this classifier?
    Accuracy = p(k*)
  – This is a “dumb” classifier, but it's always good to report this number when reporting classification results
  – On some problems, if p(k*) = 0.99, it may be hard to beat the majority classifier
  – Or if there are more than 2 classes, p(k*) may be quite small (e.g., 0.3) and an accuracy of 0.5 (say) tells you that you are picking up predictive signal
# Illustration of 5-fold Cross-Validation

<table>
<thead>
<tr>
<th>Feature Vector</th>
<th>Class Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>d1</td>
<td>x</td>
</tr>
<tr>
<td>d2</td>
<td>x</td>
</tr>
<tr>
<td>d3</td>
<td>x</td>
</tr>
<tr>
<td>d4</td>
<td>x</td>
</tr>
<tr>
<td>d5</td>
<td>x</td>
</tr>
<tr>
<td>d6</td>
<td>x</td>
</tr>
<tr>
<td>d7</td>
<td>x</td>
</tr>
<tr>
<td>d8</td>
<td>x</td>
</tr>
<tr>
<td>d9</td>
<td>x</td>
</tr>
<tr>
<td>d10</td>
<td>x</td>
</tr>
</tbody>
</table>
### Illustration of 5-fold Cross-Validation

<table>
<thead>
<tr>
<th>Feature Vector</th>
<th>Class Label</th>
<th>Fold 1</th>
<th>Fold 2</th>
<th>Fold 3</th>
<th>Fold 4</th>
<th>Fold 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>d1</td>
<td>x</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d2</td>
<td>x</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d3</td>
<td>x</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>d4</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>d5</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>d6</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>d7</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d8</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d9</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d10</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
</tbody>
</table>

For each fold, test set in yellow, training set in green

The average accuracy across the 5 test sets is computed and referred to as the cross-validated estimate of the classifier accuracy.
Comparing Algorithms with Cross-Validation

• Important that they are evaluated on exactly the same data

• Correct approach:
  – In cross-validation, outer loop is over the train-test folds, and inner loop is over the different methods

• Incorrect approach:
  – Outer loop is over the different methods, and inner loop performs CV (with different random train/test folds for each method)

• The 2\textsuperscript{nd} approach is incorrect because it introduced additional (unnecessary) variability or noise into the process of measuring the difference between the algorithms or methods
Comments on Train/Test Numbers

Consider 2 different experiments (e.g., 2 different classifiers being fit to the same data) and we run 5-fold cross validation on each and get the following results. What might we conclude?

<table>
<thead>
<tr>
<th>Classifier 1</th>
<th>Fold</th>
<th>Train Accuracy</th>
<th>Test Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>82.0</td>
<td>71.0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>73.1</td>
<td>56.0</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>98.9</td>
<td>76.0</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>67.1</td>
<td>61.0</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>84.2</td>
<td>68.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Classifier 2</th>
<th>Fold</th>
<th>Train Accuracy</th>
<th>Test Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>74.0</td>
<td>73.8</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>73.1</td>
<td>72.0</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>75.9</td>
<td>73.7</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>74.5</td>
<td>74.0</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>73.9</td>
<td>73.2</td>
</tr>
</tbody>
</table>
Statistical Significance

• Say we compare Algorithm A versus B on 20 data sets/trials
  – A does better 12 times and B does better 8 times
  – What should we conclude?

• We can use statistical hypothesis testing to help us decide
  – Null hypothesis: the two methods are equally accurate (and any variation we see is just random noise)

• Can use different types of hypothesis tests, e.g.,
  – t-test on the difference in accuracies across the 20 tests: mean is zero or not?
  – Wilcoxon sign test: just looks at number of times A or B “won” a test

• Can use same idea for user studies: how often does the user select A over B
Going Beyond Accuracy

• To get insight into what a classifier is doing, it's often useful to dig deeper beyond accuracy numbers

• Examples:
  – Look at random examples of the documents it got right and got wrong and see if there are any obvious patterns (e.g., negation in sentiment analysis)

  – If you have a classifier that is producing probabilities, you could look at the test examples with highest predicted probability of being in class 1 but that are actually class 0 (and vice-versa)

  – With 2 or more classifiers, you could see if they tend to make errors on the same examples (look at 2 x 2 tables for 2 classifiers)
Confusion Matrices

<table>
<thead>
<tr>
<th></th>
<th>TRUE LABELS</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Class 1</td>
<td>Class 2</td>
<td>Class 3</td>
<td></td>
</tr>
<tr>
<td>PREDICTED</td>
<td>Class 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LABELS</td>
<td>Class 1</td>
<td>24</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Class 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Class 3</td>
<td></td>
<td></td>
<td>50</td>
</tr>
</tbody>
</table>
### Confusion Matrices

#### TRUE LABELS

<table>
<thead>
<tr>
<th></th>
<th>Class 1</th>
<th>Class 2</th>
<th>Class 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class 1</td>
<td>24</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>Class 2</td>
<td>5</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>Class 3</td>
<td>0</td>
<td>1</td>
<td>50</td>
</tr>
</tbody>
</table>

#### PREDICTED LABELS

Accuracy = \( \frac{24 + 5 + 50}{100} = 79\% \)
## Confusion Matrices

<table>
<thead>
<tr>
<th></th>
<th>Class 1</th>
<th>Class 2</th>
<th>Class 3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TRUE LABELS</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class 1</td>
<td>24</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>Class 2</td>
<td>5</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>Class 3</td>
<td>0</td>
<td>1</td>
<td>50</td>
</tr>
</tbody>
</table>

*PREDICTED LABELS*
## Confusion Matrices

<table>
<thead>
<tr>
<th>Predicted Labels</th>
<th>TRUE LABELS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class 1</td>
<td>Class 1</td>
</tr>
<tr>
<td>Class 1</td>
<td>24</td>
</tr>
<tr>
<td>Class 2</td>
<td>5</td>
</tr>
<tr>
<td>Class 3</td>
<td>0</td>
</tr>
</tbody>
</table>

**Actual Number per Class**

- Class 1: 29
- Class 2: 21
- Class 3: 50

**Predicted Number per Class**

- Class 1: 39
- Class 2: 10
- Class 3: 51
Analyzing Class Probabilities
Class Probabilities

- Many classifiers produce class probabilities as outputs
  - E.g., logistic regression, neural networks, naïve Bayes

- For each training or test example the classifier produces
  \[ P(c \mid x) \] where \( x \) is the input vector

- For \( K \) classes there are \( K \) class probabilities that sum to 1
# Methods for Logistic Regression Classifier in scikit-learn


<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>decision_function(X)</code></td>
<td>Predict confidence scores for samples.</td>
</tr>
<tr>
<td><code>densify()</code></td>
<td>Convert coefficient matrix to dense array format.</td>
</tr>
<tr>
<td><code>fit(X, y[, sample_weight])</code></td>
<td>Fit the model according to the given training data.</td>
</tr>
<tr>
<td><code>fit_transform(X[, y])</code></td>
<td>Fit to data, then transform it.</td>
</tr>
<tr>
<td><code>get_params([deep])</code></td>
<td>Get parameters for this estimator.</td>
</tr>
<tr>
<td><code>predict(X)</code></td>
<td>Predict class labels for samples in X.</td>
</tr>
<tr>
<td><code>predict_log_proba(X)</code></td>
<td>Log of probability estimates.</td>
</tr>
<tr>
<td><code>predict_proba(X)</code></td>
<td>Probability estimates.</td>
</tr>
<tr>
<td><code>score(X, y[, sample_weight])</code></td>
<td>Returns the mean accuracy on the given test data and labels.</td>
</tr>
<tr>
<td><code>set_params(**params)</code></td>
<td>Set the parameters of this estimator.</td>
</tr>
<tr>
<td><code>sparsify()</code></td>
<td>Convert coefficient matrix to sparse format.</td>
</tr>
<tr>
<td><code>transform(*args, **kwargs)</code></td>
<td>DEPRECATED: Support to use estimators as feature selectors will be removed in version 0.19.</td>
</tr>
</tbody>
</table>
Class Probabilities in scikit-learn


```
predict_proba(X)
```

Probability estimates.

The returned estimates for all classes are ordered by the label of classes.

For a multi_class problem, if multi_class is set to be “multinomial” the softmax function is used to find the predicted probability of each class. Else use a one-vs-rest approach, i.e calculate the probability of each class assuming it to be positive using the logistic function, and normalize these values across all the classes.

Parameters:

- `X`: array-like, shape = [n_samples, n_features]

Returns:

- `T`: array-like, shape = [n_samples, n_classes]

Returns the probability of the sample for each class in the model, where classes are ordered as they are in `self.classes_`. 
Setting up a 2-Class Problem (from 20 Newsgroups Data)

```
n_categories = 2
twenty_train, twenty_test = load_news_dataset(n_categories)
```

The names of the 2 most common labels in the data set are: ['rec.sport.hockey', 'soc.religion.christian']

Dimensions of \( X_{\text{train\_counts}} \) are \((1199, 21891)\)
Number of non-zero elements in \( X_{\text{train\_counts}} \): 157195
Percentage of non-zero elements in \( X_{\text{train\_counts}} \): 0.60
Average number of word tokens per document: 196.54
Average number of documents per word token: 10.76
Class Probabilities from a Logistic Regression Model

# fit a logistic regression model and generate the probabilities
clfLR = LogisticRegression().fit(X_train_tfidf, twenty_train.target)

# generate predictions on the test data
predicted_LR = clfLR.predict(X_test_tfidf);

# generate class probabilities on the test data
prob_test_LR = clfLR.predict_proba(X_test_tfidf);
Class Probabilities from a Logistic Regression Model

prob_test_LR

array([ [ 0.7041694, 0.2958306 ],
       [ 0.50150781, 0.49849219 ],
       [ 0.74678626, 0.25321374 ],
       ...,
       [ 0.45904319, 0.54095681 ],
       [ 0.09707758, 0.90292242 ],
       [ 0.86623216, 0.13376784 ]])
SCATTER PLOT OF CLASS PROBABILITIES: BERNOUILLI V. LOGISTIC
Using Class Probabilities

• Can look at how confident a classifier is
  – E.g., which test examples for each class is it most confident on?

• Can help with analyzing what errors a classifier is making
  – E.g., find all test examples that were incorrectly classified and look at those examples that have the highest probability of the incorrect class

• Can compare classifiers
  – E.g., using scatter plots, as with the examples on the previous slides

• Can rank the test examples and analyze rankings (next section)
Ranking-based Metrics
Metrics based on Thresholding/Ranking

- Consider binary classification

- Many classifiers produce a score indicating the likelihood of class 1
  - Class probability from logistic regression or naïve Bayes
  - Distance from boundary in SVMs

- The default threshold for this score (e.g., for class probabilities) is 0.5

- But we can vary this threshold
  - E.g., in spam email we may want to set a high-threshold for declaring spam

- We can also rank our test examples by their score and measure the quality of the ranked list (relative to some threshold)
HISTOGRAM OF CLASS 0 PROBABILITIES FOR LOGISTIC REGRESSION
Binary Classification: Ranking Metrics

- To evaluate using a ranking metric we do the following
  - take a test set with N data vectors $x$
  - compute a score for each item, say $P(C = 1 \mid x)$, using our prediction model
  - sort the N items from largest to smallest score

- This gives us 2 lists, each of length N
  - A list of predictions, with decreasing score values
  - A corresponding list of “ground truth” values, 0’s and 1’s for binary class labels

- A variety of evaluation metrics can be computed based on these 2 lists
  - Precision/recall
  - Receiver-operating characteristics
  - Lift curves
  - And so on.
Ranking Terminology

<table>
<thead>
<tr>
<th>Model's Predictions</th>
<th>True Labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>TP (True positive)</td>
</tr>
<tr>
<td>Negative</td>
<td>FP (False positive)</td>
</tr>
</tbody>
</table>

Precision = TP / (TP + FP) = ratio of correct positives predicted to total positive predicted

Recall = TP / (TP + FN) = ratio of correct positives predicted to actual number of positives

Typically will get high precision for low recall, and low precision at high recall
## Precision

Precision for class $k$

\[ \text{Precision for class } k = \frac{\text{fraction predicted as class } k \text{ that belong to class } k}{100} \]

\[ = 90\% \text{ for class 2 below} \]

<table>
<thead>
<tr>
<th></th>
<th>True Class 1</th>
<th>True Class 2</th>
<th>True Class 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted Class 1</td>
<td>80</td>
<td>60</td>
<td>10</td>
</tr>
<tr>
<td>Predicted Class 2</td>
<td>10</td>
<td>90</td>
<td>0</td>
</tr>
<tr>
<td>Predicted Class 3</td>
<td>10</td>
<td>30</td>
<td>110</td>
</tr>
</tbody>
</table>
Recall

Recall for class $k$

\[
\text{Recall for class } k = \text{fraction that belong to class } k \text{ predicted as class } k = 50\% \text{ for class 2 below}
\]

<table>
<thead>
<tr>
<th></th>
<th>True Class 1</th>
<th>True Class 2</th>
<th>True Class 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted Class 1</td>
<td>80</td>
<td>60</td>
<td>10</td>
</tr>
<tr>
<td>Predicted Class 2</td>
<td>10</td>
<td>90</td>
<td>0</td>
</tr>
<tr>
<td>Predicted Class 3</td>
<td>10</td>
<td>30</td>
<td>110</td>
</tr>
</tbody>
</table>
### Binary Classification: Simple Example of Precision and Recall

- Test set with 10 items, binary labels, 5 from each class

<table>
<thead>
<tr>
<th>Scores from the Model</th>
<th>True Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.97</td>
<td>1</td>
</tr>
<tr>
<td>0.95</td>
<td>1</td>
</tr>
<tr>
<td>0.93</td>
<td>0</td>
</tr>
<tr>
<td>0.81</td>
<td>1</td>
</tr>
<tr>
<td>0.55</td>
<td>0</td>
</tr>
<tr>
<td>0.28</td>
<td>1</td>
</tr>
<tr>
<td>0.17</td>
<td>0</td>
</tr>
<tr>
<td>0.15</td>
<td>1</td>
</tr>
<tr>
<td>0.10</td>
<td>0</td>
</tr>
<tr>
<td>0.03</td>
<td>0</td>
</tr>
</tbody>
</table>
Precision-Recall with High Threshold

- Test set with 10 items, binary labels, 5 from each class

<table>
<thead>
<tr>
<th>Scores from the Model</th>
<th>True Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.97</td>
<td>1</td>
</tr>
<tr>
<td>0.95</td>
<td>1</td>
</tr>
<tr>
<td>0.93</td>
<td>0</td>
</tr>
<tr>
<td>0.81</td>
<td>1</td>
</tr>
<tr>
<td>0.55</td>
<td>0</td>
</tr>
<tr>
<td>0.28</td>
<td>1</td>
</tr>
<tr>
<td>0.17</td>
<td>0</td>
</tr>
<tr>
<td>0.15</td>
<td>1</td>
</tr>
<tr>
<td>0.10</td>
<td>0</td>
</tr>
<tr>
<td>0.03</td>
<td>0</td>
</tr>
</tbody>
</table>

**Precision:**
Percentage above threshold that are actually class 1 = 2/2 = 100%

**Recall:**
Percentage of total number of positive examples that are above the threshold = 2/5 = 40%
# Precision-Recall with High Threshold

- Test set with 10 items, binary labels, 5 from each class

<table>
<thead>
<tr>
<th>Scores from the Model</th>
<th>True Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.97</td>
<td>1</td>
</tr>
<tr>
<td>0.95</td>
<td>1</td>
</tr>
<tr>
<td>0.93</td>
<td>0</td>
</tr>
<tr>
<td>0.81</td>
<td>1</td>
</tr>
<tr>
<td>0.55</td>
<td>0</td>
</tr>
<tr>
<td>0.28</td>
<td>1</td>
</tr>
<tr>
<td>0.17</td>
<td>0</td>
</tr>
<tr>
<td>0.15</td>
<td>1</td>
</tr>
<tr>
<td>0.10</td>
<td>0</td>
</tr>
<tr>
<td>0.03</td>
<td>0</td>
</tr>
</tbody>
</table>

**Precision:** Percentage above threshold that are actually class 1 $= \frac{5}{8} = 62.5\%$

**Recall:** Percentage of total number of positive examples that are above the threshold $= \frac{5}{5} = 100\%$
How are Precision and Recall used as Metrics?

Precision @ K is often used as a metric, where K is the top K items or the top K% of the sorted prediction list

– E.g., useful for information retrieval, for search

Precision-recall curves can be plotted by varying the threshold

High threshold (e.g., 0.95) often yields high precision, low recall

Low threshold (e.g., 0.1) often yields low precision, high recall
Computing Precision-Recall Curves

Rank our test documents by predicted probability of belonging to class 1

- Sort the ranked list
- For \( t = 1 \ldots N \) (\( N \) = total number of test documents)
  - Select top \( t \) documents on the list
  - Classify documents above as class 1, documents below as not class 1
  - Compute precision and recall number for each \( t \)

- Generates a set of pairs of (precision, recall) values we can plot as \((x,y)\)

- Perfect performance: precision = 1, recall = 1
  - Requires that all the positive examples get higher scores than all negatives
Precision-Recall Example: Reuters Data, Class Label = Crude Oil

Figure from S. Dumais (1998)
Additional Aspects of Document Classifiers
Algorithm Parameters

Many machine learning/AI algorithms have parameters that can be “tuned”

Example: classification algorithms

- Logistic regression
  - Objective function = error function + $\lambda$ * sum of weights squared
  - Here $\lambda$ is a parameter of the method that controls the amount of regularization

- Support vector machines
  - The parameter C controls regularization

Default settings

- The software you are using (e.g., from scikit-learn) may have default values for these parameters
- Note that these are not necessarily optimal
- You may want to experiment with different values via cross-validation
Sensitivity

- Accuracy on test data may be sensitive to the values of the parameters
  - E.g., with smaller data sets more regularization may give better results

- How can we figure out what values work best?

- Use a validation set (or cross-validation) from within your training data
  - Set the parameter to different values (above and below default value)
    - Initially may want to search on a log-scale to get a general idea of the scale
      - e.g., try $\lambda$ values of 0.001, 0.01, 0.1, 1, 10, 100
    - For each value, train and test a classifier
  - Look at (e.g., plot) the test accuracy as a function of the parameter value
    - Can then rerun on a finer scale to zero-in (optional)
  - This is known as grid search
Learning Curves


Score

Training Curves (Naive Bayes)

| Training score |
| Cross-validation score |

Training examples

0 200 400 600 800 1000 1200 1400 1600
Learning Curves

Feature Selection

• Can be particularly useful for Naïve Bayes models
  – Logistic regression (with regularization) and SVMs are often robust enough to not need feature selection

• Select the most common features, e.g., most common words?
  – These are not necessarily the most discriminative words
Feature Selection

• Can be particularly useful for Naïve Bayes models
  – Logistic regression (with regularization) and SVMs are often robust enough to not need feature selection

• Select the most common features, e.g., most common words?
  – These are not necessarily the most discriminative words

• Better approach: select the K most discriminative features
  – Simple method
    • Compute how well each feature does on its own, select top K
    • e.g., use the mutual information measure (see Chapter 13, Sect 5, in Manning text)

  – Greedy method:
    • Train a classifier on each feature on its own, select the feature that does best
    • Then from the rest, train D – 1 classifiers, adding each other feature one at a time
    • Pick the best
    • Continue until K features have been added
Additional Topics

• Using “document zones”
  – Weighting of terms different zones, e.g., upweighting words in titles, abstracts
  – Separate features/term-counts for different sections of a document
    • E.g., patent abstract v claims

• Features beyond term-counts or term presence
  – Using TF-IDF for term-weighting
  – Grouping terms based on prior knowledge (e.g., lists of positive and negative sentiment words)
  – Using parts-of-speech information (e.g., number of adjectives)

• Semi-supervised learning
  – Probabilistic algorithms (such as Expectation-Maximization) that can learn from both labeled and unlabeled examples
  – Still more of a research topic than a practical technique

[See Manning et al, Chapter 15, Section 15.3 for further discussion]
Advice: It's not always the about the algorithm...

- It is tempting in a predictive modeling project to spend a lot of time experimenting with the algorithms
  - E.g., different neural network architectures and learning heuristics

- But in practice which variables are used may be much more important than which algorithm

- So a useful strategy may be to rethink what the variables/attributes are. There might be another variable or two you can add to your problem that will increase classification accuracy by a large margin.
Experiments Comparing Different Classifiers

From Zhang and Oles, Text categorization based on regularized linear classification methods

Table 2. Binary classification performance on Reuters (all 118 classes).

<table>
<thead>
<tr>
<th></th>
<th>Naive Bayes</th>
<th>Lin Reg</th>
<th>Mod Least Squares</th>
<th>Logistic Reg</th>
<th>SVM</th>
<th>Mod SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision</td>
<td>77.0</td>
<td>87.1</td>
<td>89.2</td>
<td>88.0</td>
<td>89.2</td>
<td>89.4</td>
</tr>
<tr>
<td>Recall</td>
<td>76.9</td>
<td>84.9</td>
<td>85.3</td>
<td>84.9</td>
<td>84.0</td>
<td>83.7</td>
</tr>
<tr>
<td>$F_1$</td>
<td>77.0</td>
<td>86.0</td>
<td>87.2</td>
<td>86.4</td>
<td>86.5</td>
<td>86.5</td>
</tr>
<tr>
<td>BEP</td>
<td>75.8</td>
<td>86.3</td>
<td>86.9</td>
<td>86.9</td>
<td>86.5</td>
<td>86.7</td>
</tr>
</tbody>
</table>
Accuracy as a Function of Data Size

With enough data the choice of classifier may not matter much.

Scaling to very very large corpora for natural language disambiguation
Banko and Brill, Annual Meeting of the ACL, 2001
Project Slide Presentations

• In-class, Monday and Wednesday next week

• Each student or team will make one presentation
  – Approximately 4 minutes per presentation

• Time after each presentation for questions and changeover
  – So about 6 minutes x 11 projects per day = 66 minutes total

• List and order of presentations will be announced this Wednesday (by email and on Piazza)

• Slides need to be uploaded to the EEE dropbox by 1pm the day of your presentation:
  – they will be loaded on to the classroom machine, no need to use your laptop.
  – Either PDF or Powerpoint is fine
# Weekly Schedule

<table>
<thead>
<tr>
<th>Week</th>
<th>Monday</th>
<th>Wednesday</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feb 27</td>
<td>Lecture: Progress Reports, etc</td>
<td>Office hours (no lecture)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Progress Report due by 11pm</td>
</tr>
<tr>
<td>Mar 6</td>
<td>Project Presentations (in class)</td>
<td>Project Presentations (in class)</td>
</tr>
<tr>
<td></td>
<td>Submit slides by 1pm</td>
<td>Submit slides by 1pm</td>
</tr>
<tr>
<td>Mar 13</td>
<td>Short Lecture on Final Project Reports</td>
<td>No lecture or office hours</td>
</tr>
<tr>
<td>Mar 20</td>
<td></td>
<td>Final Report due to EEE by Tuesday March 21st</td>
</tr>
</tbody>
</table>

Note: no final exam, just submit your final report by Tuesday night of finals week