Lab #5: Evaluating Classifiers

Everything Data
CompSci 290.01 Spring 2014
Announcements (Tue. Feb. 11)

• **Amazon VM** instructions available; $100 codes emailed
  – Amazon instances are optional for now
  – Don’t waste all $100 on idle instances
  – You might need some serious compute power on later on MapReduce and course project

• **Project team formation** due in one week
  – 3 is the ideal team size; talk to us if you need special arrangement
Seat assignment

Front of D106

Course staff

A  B
C
D  E
F
G  H
I
J  K
L
M  N

Back of D106
Format of this lab

• Discussion of HW #5 (15 minutes)
• Introduction to Lab #5 (10 minutes)
• Lab #5 (35 min.)
  – Team challenge: win prizes and extra credits!
• Discussion of Lab #5 (15 minutes)
HW #5, Part 1:

“Lucky” teammates

• Prob. of $A$, $B$ in the same group again?
  – 2 (lucky slots) out of 41 (remaining slots)
  – About 5%

• Prob. that no pairs got lucky?
  – Rough estimate (with incorrect independence assumptions): 3 chances to get lucky for each of the 14 groups
    • $(1 - 2/41)^{42} \approx 12\%$
  – Sample solution: slightly finer analysis + simulation ($\approx 12\%$)
Population numbers: histogram similar, but p-value small because of large # of samples

Iranian election numbers (Rezai’s): histogram dissimilar, but p-value not small because of small # of samples

Was your process of reasoning influenced by prejudice?
HW #5, Part 3:

movielens data

(A) and (B) are straightforward

(C) Just looking at who rated A and B in data won’t give you anything—**need to generalize** what you see in data:

\[
P(U \text{ female} | U \text{ rates } AB) : P(U \text{ male} | U \text{ rates } AB) \\
= ( P(U \text{ rates } A | U \text{ female}) \ P(U \text{ rates } B | U \text{ female}) \ P(U \text{ female}) ) : \\
( P(U \text{ rates } A | U \text{ male}) \ P(U \text{ rates } B | U \text{ male}) \ P(U \text{ male}) )
\]

\[\approx 0.28 : 1, \text{ which implies } P(U \text{ female} | U \text{ rates } AB) \approx 0.22\]

- Compare with prior \(P(\text{female}) : P(\text{male}) \approx 0.41 : 1\)
- Basic idea behind **Naïve Bayes Classifier**
Introducing Lab #5

**Classification** problem: Given the set of movies a user rated, and the user’s occupation, predict the user’s gender

<table>
<thead>
<tr>
<th>m1</th>
<th>m2</th>
<th>m3</th>
<th>...</th>
<th>m1682</th>
<th>o1</th>
<th>o2</th>
<th>...</th>
<th>o21</th>
<th><strong>gender</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>...</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>...</td>
<td>0</td>
<td>M</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>...</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>...</td>
<td>0</td>
<td>F</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>...</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>...</td>
<td>0</td>
<td>M</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>...</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>...</td>
<td>0</td>
<td>???</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>???</td>
</tr>
</tbody>
</table>

**Training data** to teach your classifier

**Test data** to evaluate your classifier

Accuracy = (# test records classified correctly) / (# test records)
Where is test data?

What if no test data is specified, or we don’t know the right answers?

• We can still evaluate our classifier by splitting the data given to us

Rookie mistake:
train and test using the same (whole) dataset
Lucky splits, unlucky splits

• What if a particular split gets lucky or unlucky?
• Should we tweak the heck out of our classification algorithm just for this split?

☞ Answer: *cross-validation*, a smart way to make best use of available data
$r$-fold cross-validation

- Randomly divide data into $r$ groups (say 10)
- Hold out each group for testing; train on the remaining $r-1$ groups
  - $r$ train-test runs and $r$ accuracy measurements
  - A better picture of performance
Three little classifiers

• classifyA.py: a “mystery” classifier
  – Read the code to see what it does
• classifyB.py: Naïve Bayes Classifier
  – Along the same line as HW#5, 1(C)
• classifyC.py: $k$-Nearest-Neighbor Classifier
  – Given $x$, choose the $k$ training data points closest to $x$; predict the majority class
More on the \( k \)NN classifier

\[ \Delta x \]

\( k = 1 \)

Source: Daniel B. Neil’s slides for 90-866 at CMU
Team work

1. Train-Test Runs and the Mystery of $A$
   (A) Which classifier seems to work best?
   (B) What exactly does $A$ do?

2. Tweaking $k$NN
   (A) How does $k$ affect accuracies on training vs. test data? Is big or small $k$ better for this problem?
   (B) How does $k = 500$ compare with $A$?
Team challenge

**The Evil SQL Splitters:** find a train-test split such that the classifiers are great on training data but horrible on test

- %5 extra credit if you screw up two classifiers; %10 for all three

**Redemption of Naïve Bayes:** find a train-test split such that $B$ beats $A$ and $C$ hands-down

- %5 extra credit if $B$ beats others by $2\times$; 10% if $4\times$ and $B$ has $\geq 60\%$ accuracy

**Prizes for first to get 10% and for best answers**
Discussion: Parts 1 & 2

1. Train-Test Runs and the Mystery of A
   (A) Which classifier seems to work best? A
   (B) What exactly does A do?

2. Tweaking $kNN$
   (A) How does $k$ affect accuracies on training vs. test data? Is big or small $k$ better for this problem?
   Training accuracy goes down, but test accuracy goes up; bigger $k$ seems better here
   (B) How does $k = 500$ compare with A?
   It approaches A—it basically goes by M/F ratio in a significant fraction of the training data
   Just looks at M/F ratio in training data; doesn’t even use other features
Lessons learned:

**Overfitting hurts generalization**

1NN: fits training data perfectly, but handles noise and outliers poorly and doesn’t generalize well

3NN: has a smoother “boundary,” and is less susceptible to noise and outliers

*What if we set k really, really big?*

The opposite happens: *underfitting*

Image source: Daniel B. Neil’s slides for 90-866 at CMU
Discussion: Part 3 challenge

- **The Evil SQL Splitters:** find a train-test split such that the classifiers are great on training data but horrible on test
  - Make training all males except one; test is all-female
    - B and C learn very little about females
    - A is messed up by the wrong ratio
  
- **Redemption of Naïve Bayes:** find a train-test split such that B beats A and C hands-down
  - Start with above split, but add some females (~70)
    - Not enough to sway the ratio to save A (0 accuracy)
    - Not enough to cover the space for C (0.12 accuracy)
    - B becomes better faster with more females (0.61 accuracy)
Lessons learned:

Usefulness and limitation of CV

• Smart reuses of available data
  – Paints a broader picture of accuracy
  – Allows tuning of “hyper” parameters
    • E.g., $k$ in $k$NN

• Again, don’t test on data you train with
  – If you also need to tune, split data into train-validate-test

• Still, the “real test” remains unseen
  – No amount of cross-validation will help if your data collection is flawed
Finally

- Remember to submit team.txt by midnight
- Sample solutions to Homework #5 and Lab #5 will be posted by tonight