Evaluation and applications of classifiers

Micha Elsner

January 21, 2014
Where we are now

Introduced the Naive Bayes classifier...

- Choose between labels for an object given features

Remember, parts of a CL project:

- Problem design
  - How do you pick features? Some later lecture...
- Model construction
  - Just covered
- Estimation
  - Covered for NB; next lecture for ME
- Inference
  - Covered
- Evaluation
  - How do you score your results?
Scoring results

We built this great classifier!

- How well does it work?

Obvious answer 1:

**Accuracy**

Fraction of correctly labeled test objects:

\[ \text{acc} = \frac{\#(\text{proposed} = \text{correct})}{n} \]

Obvious answer 2 (probabilistic classifiers):

**Log-likelihood of the test labels**

How surprised are we by the test data?

\[ \mathcal{L} = \sum_{\text{test label } t, \text{features } F} \log P(t|F) \]
Accuracy vs goodness of fit

- Stereotypically, CL tends to care about outcomes
  - Are predictions significantly better?
- Rather than either goodness-of-fit (likelihood)
- Or significance of individual features
- Decisions are model-independent
  - Goodness of fit not always comparable across models
- Performance matters
  - For engineering, downstream research tasks
- Large number of “insignificant” features can be useful
  - Context words in LM
Issues with accuracy

Consider a medical diagnosis problem:

▶ Ohio Department of Health estimates 4 cases of TB per 100000 people in Franklin Co.

TB Test one

▶ If you have TB: yes
▶ If you don’t: P(yes) = .01, else no

TB Test two

▶ Always: No

<table>
<thead>
<tr>
<th>Test one</th>
<th>Test two</th>
</tr>
</thead>
<tbody>
<tr>
<td>999996 no TB</td>
<td>0 wrong</td>
</tr>
<tr>
<td>about 100 wrong</td>
<td>4 with TB</td>
</tr>
<tr>
<td>4 wrong</td>
<td>0 wrong</td>
</tr>
<tr>
<td>acc</td>
<td>99.99%</td>
</tr>
</tbody>
</table>

98%
Class skew

Accuracy has problems with very skewed proportions of classes

- Comp ling is full of these problems
- ...a consequence of Zipf’s law (again!)

NN  13166
IN  9857
NNP 9410
DT  8165
NNS 6047
JJ  5834

... ... 
LS  13
FW  4
UH  3
SYM 1
The most popular CL way to deal with this:

**Precision of class** $t$

$$ prec(t) = \frac{\#(proposed = true = t)}{\#(proposed = t)} $$

When you claim to detect $t$, are you right?

**Recall of class** $t$

$$ rec(t) = \frac{\#(proposed = true = t)}{\#(true = t)} $$

How many of the $t$ did you get?
F-score

\[ F1(t) = \frac{2 \times prec(t) \times rec(t)}{prec(t) + rec(t)} \]

The harmonic mean of precision and recall

- Between the two, but closer to the lower one
Back to our TB example:

<table>
<thead>
<tr>
<th></th>
<th>999996 no TB</th>
<th>4 with TB</th>
<th>acc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test one</td>
<td>about 100</td>
<td>0</td>
<td>98%</td>
</tr>
<tr>
<td></td>
<td>wrong</td>
<td>wrong</td>
<td></td>
</tr>
<tr>
<td>Test two</td>
<td>0</td>
<td>4</td>
<td>99.99%</td>
</tr>
<tr>
<td></td>
<td>wrong</td>
<td>wrong</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Proposed with TB</th>
<th>Detected</th>
<th>Prec</th>
<th>Rec</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test one</td>
<td>100</td>
<td>4</td>
<td>4</td>
<td>.076</td>
</tr>
<tr>
<td>Test two</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Precision and recall for multiclass classifiers

Two basic ideas:

- **Macro-averaging**
  - Compute $p$, $r$, $f$ for each class (noun, verb, prep)
  - Then average the f-scores
  - Equal weight for each category
    - Performance on “noun” equally weighted vs “foreign word”)

- **Micro-averaging**
  - Compute global $p$, $r$ over all test items
  - Compute $f$ from these
  - Equal weight for each item
    - Performance on “noun” matters most
  - With exactly one label per item, equal to accuracy
The precision/recall tradeoff

$p$ and $r$ are complementary:

- You can raise $p(t)$ by saying “no” more often
  - But more likely to miss true instances of $t$
- And $r(t)$ by saying “yes” more often
  - But more likely to falsely detect $t$

Possible to alter this tradeoff by tuning the decision threshold (probability at which you detect $t$)

- For a binary classifier, threshold usually $.5$
  - If $P(t) > .5$ answer $t$, else not
  - But if we want to detect more $t$, move this down...
    - If $P(t) > .25$ answer $t$...
- In general case, tune prior $P(T)$ or bias features in ME

This kind of tuning doesn’t improve model of the data
- But can matter for performance
ROC curves

Visualize classifier’s performance over entire range of tradeoffs

- Could plot \( p \) vs \( r \)
- Instead conventional to plot sensitivity and false positive rate
- True positive rate (\( = \) recall) \( \frac{\#(\text{proposed} = \text{truth} = t)}{\text{truth} = t} \)
- False positive rate: \( \frac{\#(\text{proposed} = t, \text{truth} \neq t)}{\text{truth} \neq t} \)
- (There are some arguments that this kind of measurement is better than precision/recall)
  - Renders scores more comparable across datasets with different class ratios
- In practice, you should use what people in your field use
The ROC curve

Sometimes summarized as Area Under the Curve (AUC) number

Confusion matrices

Rather than just ask “how good”, often worth digging deeper:

▶ What does the classifier do wrong?

<table>
<thead>
<tr>
<th>true↓ / proposed→</th>
<th>NN</th>
<th>JJ</th>
<th>VB</th>
</tr>
</thead>
<tbody>
<tr>
<td>NN</td>
<td>12206</td>
<td>379</td>
<td>170</td>
</tr>
<tr>
<td>JJ</td>
<td>211</td>
<td>5289</td>
<td>16</td>
</tr>
<tr>
<td>VB</td>
<td>428</td>
<td>44</td>
<td>1825</td>
</tr>
</tbody>
</table>

Diagonal entries indicate correct decisions

▶ Off diagonals show mistakes
As a heat map

(Produced with pylab.pcolor)