Advanced Evaluation, Imbalanced Data

Instructor: Jessica Wu -- Harvey Mudd College

The instructor gratefully acknowledges Eric Eaton (UPenn), David Kauchak (Pomona), Tommi Jaakola (MIT) and the many others who made their course materials freely available online.

Advanced Evaluation Metrics

Learning Goals

• Describe metrics for evaluating performance
  — AUROC, precision, recall, F₁-score
ROC Curves

Obtain curve by varying threshold on confidence of instance being positive

Assume balanced classes

Identify
a) best operating point when cost of misclassifying positives and negatives is equal
b) best operating point when FP costs 10x FN
c) best operating point when FN costs 10x FP
Algorithm for Creating ROC Curve

- Sort test set predictions according to confidence that instance is positive
- Step through sorted list from high to low confidence
  - locate *threshold* between instances with opposite classes (keeping instances with same confidence value on same side of threshold)
  - compute TPR, FPR for instances above threshold
  - output (FPR, TPR) coordinate

Plotting an ROC Curve

<table>
<thead>
<tr>
<th>instance</th>
<th>confidence in positive</th>
<th>correct class</th>
<th>predicted class</th>
<th>actual class</th>
<th>TPR = TP / (TP + FN)</th>
<th>FPR = FP / (TN + FP)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>yes</td>
<td>no</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0.99</td>
<td>+</td>
<td>TP</td>
<td>FN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.98</td>
<td>+</td>
<td>FP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.72</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.70</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0.65</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0.51</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.39</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.24</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.11</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0.01</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
But wait...

Does low FPR (high specificity) indicate that most positive predictions (predictions with confidence > some threshold) are correct?

Suppose TPR = 0.9, FPR = 0.01

<table>
<thead>
<tr>
<th>fraction of instances that are positive</th>
<th>fraction of positive predictions that are correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>0.989</td>
</tr>
<tr>
<td>0.1</td>
<td>0.909</td>
</tr>
<tr>
<td>0.01</td>
<td>0.476</td>
</tr>
<tr>
<td>0.001</td>
<td>0.083</td>
</tr>
</tbody>
</table>

TPR = TP / (TP + FN)
FPR = FP / (TN + FP)
PR (positive rate) = (TP + FN) / total
NR (negative rate) = (TN + FP) / total

Fraction of positive predictions that are correct:
= TP / (TP + FP)
= TPR * PR / (TPR * PR + FPR * NR)

Confusion Matrix

Given dataset of P positive instances and N negative instances:

\[
\text{accuracy} = \frac{TP + TN}{P + N}
\]

Imagine a classifier that identifies presence of disease:

\[
\text{sensitivity} = \frac{TP}{TP + FN}
\]

(true positive rate) = probability of positive test given person has disease

\[
\text{specificity} = \frac{TN}{TN + FP}
\]

(true negative rate) = probability of negative test given person does not have disease

\[
\text{precision} = \frac{TP}{TP + FP}
\]

(positive predictive value) = probability that person has disease given positive test

\[
\text{recall} = \frac{TP}{TP + FN}
\]

(true positive rate) = probability of positive test given person has disease

Based on slide by David Page

Based on slide by Eric Eaton
Precision-Recall Curves

Plots precision vs recall as you vary threshold on confidence of instance being positive

Based on slide by David Page

[Image source: Kawaler et al., Proc of AMIA Annual Symposium, 2012]
Best Operating Point

Compromise between precision and recall
(harmonic mean)

\[ F_\beta = \frac{(1 + \beta^2) \cdot \text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}} \]

\[ = \frac{1}{\alpha \cdot \frac{1}{\text{precision}} + (1 - \alpha) \cdot \frac{1}{\text{recall}}} \quad \alpha = \frac{1}{1 + \beta^2} \]

\( F_1 \) measure most common
(\( \alpha = 0.5, \beta = 1 \), precision and recall weighted equally)

\[ F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \]

Metrics Exercise

<table>
<thead>
<tr>
<th></th>
<th>predicted class</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>yes</td>
</tr>
<tr>
<td>actual class</td>
<td></td>
</tr>
<tr>
<td>yes</td>
<td>100</td>
</tr>
<tr>
<td>no</td>
<td>10</td>
</tr>
</tbody>
</table>

sensitivity?
specificity?
precision?
recall?
\( F_1 \)-score?
Why Harmonic Mean?

- Punishes *extreme* values more

Example
- dataset: infinite examples of negative class
  one example of positive class
- classifier: (trivial) always predict positive
- then: precision = recall =
  arithmetic mean = harmonic mean ($F_1$) =

$\Rightarrow$ for a high $F_1$, need *both* high precision and recall

- Mathematically correct

  harmonic mean = reciprocal of arithmetic mean of reciprocals

  $\text{precision} = \frac{TP}{TP + FP}$ \quad $\text{recall} = \frac{TP}{TP + FN}$

  $F_1$-score takes averages over the same denominator

Comments on ROC and PR curves

*Both*
- allow predictive performance to be assessed at various levels of confidence
- assume binary classification tasks
- sometimes summarized by calculating *area under the curve* (AUROC, AUPR)

*ROC curves*
- insensitive to changes in class distribution
  (ROC curve does not change if proportion of positive and negative instances in test set are varied)
- can identify optimal classification thresholds for tasks with differential misclassification costs

*PR curves*
- show fraction of predictions that are false positives
- well-suited for tasks with lots of negative instances

Based on slide by David Page
A Word of Caution

Consider binary classifiers A, B, C

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predictions</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- Clearly A is useless since it always predicts 1
- B is slightly better than C
  - less probability mass wasted on off-diagonals
- But, here are the performance metrics

<table>
<thead>
<tr>
<th>Metric</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>0.9</td>
<td>0.9</td>
<td>0.88</td>
</tr>
<tr>
<td>Precision</td>
<td>0.9</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Recall</td>
<td>1.0</td>
<td>0.88</td>
<td>0.8667</td>
</tr>
<tr>
<td>F-score</td>
<td>0.947</td>
<td>0.941</td>
<td>0.9286</td>
</tr>
</tbody>
</table>

Imbalanced Data

Learning Goals

- Describe approaches for handling imbalanced data and the trade-offs of each
Setup

1. for 1 hour, Google collects 1M e-mails randomly
2. they pay people to label them as “phishing” or “not-phishing”
3. they give the data to you to learn to classify e-mails as phishing or not
4. you, having taken ML, try out a few of your favorite classifiers
5. you achieve an accuracy of 99.997%

Should you be happy?

Imbalanced Data

The phishing problem is what is called an **imbalanced data** problem
- occurs where there is large discrepancy between number of examples with each class label
- e.g. for our 1M example dataset, only ~30 would actually represent phishing e-mails

What is probably going on with our classifier?

- always predict not-phishing
- 99.997% accuracy

Why does the classifier learn this?
- Many classifiers are designed to optimize error/accuracy
- This tends to bias performance towards majority class
- *Anytime* there is imbalance in the data, this can happen
- It is particularly pronounced, though, when imbalance is more pronounced
Imbalanced Problem Domains

Besides phishing (and spam), what are some other imbalanced problems domains?

- Medical diagnosis
- Predicting faults/failures (e.g. hard-drive failures, mechanical failures, etc.)
- Predicting rare events (e.g. earthquakes)
- Detecting fraud (credit card transactions, internet traffic)

Black-Box Approach

Abstraction: We have generic binary classifier. How can we use it to solve our new problem?

Can we do some pre-processing/post-processing of our data to allow us to still use our binary classifiers?
**Idea 1: Subsampling**

Create a new training data set by
- including all $k$ minority-class examples
- randomly picking $k$ majority-class examples

**Idea 2: Oversampling**

Create a new training data set by
- including all $m$ majority-class examples
- including $m$ minority-class examples:
  - repeat each example fixed number of times, or
  - sample with replacement
Idea 2b: Weighted Examples

Add costs/weights to training set
- majority-class examples get weight 1
- minority-class examples get much larger weight

$99.997/0.003 = 33332$

Idea 3: Optimize Different Metric

Train classifiers that try and optimize $F_1$ or AUC or ...
come up with another learning algorithm designed specifically for imbalanced problems
Idea 1: Subsampling

Create a new training data set by
- including all $k$ minority-class examples
- randomly picking $k$ majority-class examples

Pros
- Easy to implement
- More efficient training (smaller training set)
- For some domains, can work very well

Cons
- Throwing away lots of data/information

Idea 2: Oversampling

Create a new training data set by
- including all $m$ majority-class examples
- include $m$ minority-class examples:
  - repeat each example fixed number of times, or
  - sample with replacement

Pros
- Easy to implement
- Utilizes all training data
- Tends to perform well in broader set of circumstances than subsampling

Cons
- Computationally expensive to train classifier
Idea 2b: Weighted Examples

Add costs/weights to training set
- majority-class examples get weight 1
- minority-class examples get much larger weight

Change learning algorithm to optimize weighted error

Pros
- Achieves effect of oversampling without computational cost
- Utilizes all training data
- Tends to perform well in broader set of circumstances

Cons
- Requires classifier that can deal with weights

Based on slide by David Kauchak

---

Idea 3: Optimize Different Metric

Train classifiers that try and optimize $F_1$ or AUC or ... come up with another learning algorithm designed specifically for imbalanced problems

pros/cons?
- Not all classifiers amenable to optimizing $F_1$ or AUC
- Do not want to reinvent the wheel – that said, there are a number of approaches specifically developed to handle imbalanced problems

Based on slide by David Kauchak