Building a classifier over imbalanced data

Sources
https://svds.com/learning-imbalanced-classes/
http://www.cs.pomona.edu/~dkauchak/classes/f13/cs451-f13/
Imbalanced classes

• Most classification methods assume classes are reasonably balanced
Imbalanced classes

- In reality it is quite common to have a very popular class and a rare (yet interesting) one
Imbalanced classes

• Examples:
  - About 2% of **credit card** accounts are defrauded per year\(^1\).
    (Most fraud detection domains are heavily imbalanced.)
  - **Medical screening** for a condition is usually performed on a large population of people without the condition, to detect a small minority with it (e.g., HIV prevalence in the USA is \(~0.4\%\)).
  - **Disk drive failures** are approximately \(~1\%\) per year.
  - The **conversion rates** of online ads has been estimated to lie between 10-3 to 10-6.
  - **Factory production defect** rates typically run about 0.1\%.  

The phishing problem is what is called an **imbalanced data** problem.

This occurs where there is a large discrepancy between the number of examples with each class label.

E.g. for our 1M example dataset only about 30 would actually represent phishing e-mails.

What is probably going on with our classifier?
Imbalanced data

Why does the classifier learn this?
Imbalanced data: current classifiers

How will our current classifiers do on this problem?
Imbalanced data: current classifiers

All will do fine if the data can be easily separated/distinguished

Decision trees:
• explicitly minimizes training error
• when pruning pick “majority” label at leaves
• tend to do very poor at imbalanced problems

k-NN:
• even for small k, majority class will tend to overwhelm the vote

perceptron:
• can be reasonable since only updates when a mistake is made
• can take a long time to learn
Handling imbalanced data

• Possible alternatives
  – Do nothing and hope to be lucky
  – Balance the training set in some way:
    • Oversample the minority class
    • Undersample the majority class
    • Synthesize new minority classes
  – Throw away minority examples and switch to an anomaly detection framework
  – At the algorithm level:
    • Adjust the class weight (misclassification costs)
    • Adjust the decision threshold
    • Modify an existing algorithm to be more sensitive to rare classes
  – Construct an entirely new algorithm to perform well on imbalanced data
Balancing the dataset

- Oversampling the minority class
Balancing the dataset

• Undersampling the majority class
Balancing the dataset

- Undersampling the majority class
- Bayesian argument (Wallace et al., ICDM 2011)
Balancing the dataset

- Undersampling the majority class
- Bayesian argument (Wallace et al., ICDM 2011)
Balancing the dataset

- Smart undersampling
  - Remove some majority class points
  - Neighbor-based approaches, e.g. *Tomek links*
    - Remove majority points having as NN a minority point
Balancing the dataset

- Smart oversampling
  - Add some minority class points
  - E.g. SMOTE (Synthetic Minority Oversampling Technique)
    - Add points through interpolation

Adjusting class weights

- Example from Python scikit-learn
  - Some classifiers have a “class_weight” parameter
Related topic: evaluating classifiers on imbalanced data

- When classes are **slightly** imbalanced, no balancing is needed.
- Yet, take that into consideration when evaluating performances.

E.g.: Assume the test set contains 100 records

**Positive cases** = 75, **Negative cases** = 25

- Is a classifier with 70% accuracy good?
- No, the trivial classifier (always **positive**) reaches 75%

**Positive cases** = 50, **Negative cases** = 50

- Is a classifier with 70% accuracy good?
- At least much better than the trivial classifier.

**Take-home message**

- Accuracy scores should be compared against some baseline classifier, e.g. Majority class classifier or a simple-yet-not-trivial one.
Similar situation: multiclass problems

- Assume N classes

- If classes are perfectly balanced, a trivial classifier (e.g. majority) will yield $A_{\text{trivial}} \sim \frac{100}{N} \%$ accuracy
  - N=2 → $A_{\text{trivial}} \sim 50\%$
  - N=4 → $A_{\text{trivial}} \sim 25\%$

- Goodness of accuracy of a model should be compared against $A_{\text{trivial}}$
  - If N=5, an accuracy of 40% would look large
Again on evaluation: scoring/ranking vs. classifying

- Two slightly different objectives
  - Classifying = assigning a record to a class
  - Scoring/ranking = assigning **probabilities** of belonging to a class

- Several classification methods compute scores, and then assign class
  - Score $p > 50\% \rightarrow \text{class} = Y$
  - Otherwise $\rightarrow \text{class} = N$

- E.g.: decision trees have $p = \#\text{positive}/\#\text{negative cases over each leaf}$
Again on evaluation: scoring/ranking vs. classifying

- What if we generalize the schema into:
  - Score $p > X\% \rightarrow \text{class} = Y$
  - Otherwise $\rightarrow \text{class} = N$
- For each $X$ (in $[0-100]$) we get a different set of predictions
  - The confusion matrix changes
  - All indicators derived from it change
    - Accuracy
    - TPR
    - TNR
    - ...

Again on evaluation: scoring/ranking vs. classifying

- Deeper insights on our model can be obtained looking at how performances change with $X$
  - ROC curve: plots TPR vs. FPR
Again on evaluation: scoring/ranking vs. classifying

- Deeper insights on our model can be obtained looking at how performances change with X
  - Precision vs. recall
Again on evaluation: scoring/ranking vs. classifying

- Deeper insights on our model can be obtained looking at how performances change with $X$
  - Lift chart: % of positive cases vs. % of dataset classified as $Y$

Notice: “Lift chart” is a rather general term, often used to identify also other kinds of plots. Don’t get confused!
Again on evaluation: Application example

- From Lift chart we can easily derive an “economical value” plot, e.g. in target marketing
  - Question: Given our predictive model, how many customers should we target to maximize income?

- Simple economical model
  - Profit = UnitB*MaxR*Lift(X) - UnitCost*N*X/100
    - $\text{UnitB} = \text{unit benefit}$, $\text{UnitCost} = \text{unit postal cost}$
    - $N = \text{total customers}$, $\text{MaxR} = \text{expected potential respondents in all population (N)}$
    - $\text{Lift}(X) = \text{lift chart value for X, in [0,..,1]}$
Again on evaluation: Application example

- From Lift chart we can easily derive an “economical value” plot, e.g. in target marketing
  - Question: Given our predictive model, how many customers should we target to maximize income?

<table>
<thead>
<tr>
<th>UnitB</th>
<th>N</th>
<th>MaxR</th>
<th>UnitCost</th>
</tr>
</thead>
<tbody>
<tr>
<td>6€</td>
<td>30000</td>
<td>10500</td>
<td>2.30€</td>
</tr>
</tbody>
</table>

Optimal X = 40%