Learning from Imbalanced Data

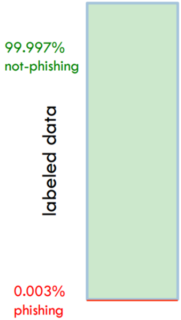
Piyush Rai

Machine Learning (CS771A)

Nov 5, 2016

Learning from Imbalanced Data

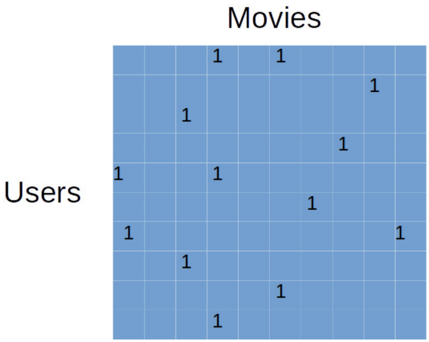
- Consider binary classification
- Often the classes are highly imbalanced



- Should I feel happy if my classifier gets 99.997% classification accuracy on test data ?

Learning from Imbalanced Data

- Other problems can also exhibit imbalance (e.g., binary matrix completion)



Binary Matrix Completion
0.001 % 1s in the matrix

- Should I feel happy if my matrix completion model gets 99.999% matrix completion accuracy (or MAE close to 0) on the test entries?

True Definition of Imbalance Data?

- Debatable..
- Scenario 1: 100,000 negative and 1000 positive examples
- Scenario 2: 10,000 negative and 10 positive examples
- Scenario 3: 1000 negative and 1 positive example
- Usually, imbalance is characterized by absolute rather than relative rarity
    - Finding needles in a haystack..

Minimizing Loss

- Any model to minimize the loss, e.g.,

$$\text{Classification:} \quad \hat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} \sum_{n=1}^{N} \ell(y_n, \boldsymbol{w}^\top \boldsymbol{x}_n)$$

$$\text{Matrix Completion:} \quad (\hat{\mathbf{U}}, \hat{\mathbf{V}}) = \arg\min_{\mathbf{U},\mathbf{V}} ||\mathbf{X} - \mathbf{U}\mathbf{V}^\top||^2$$

  .. will usually get a high accuracy

- However, it will be highly biased towards predicting the majority class
  - Thus accuracy alone can't be trusted as the evaluation measure if we care more about predicting minority class (say positive) correctly

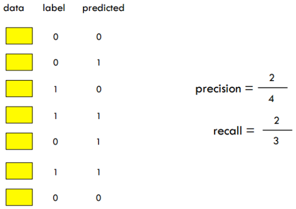
Better Evaluation Measures

- Precision: What fraction of positive predictions is truly positive

$$P = \frac{\#\ \text{example correctly predicted as positive}}{\#\ \text{examples predicted as positive}}$$

- Recall: What fraction of total positives are predicted as positives

$$R = \frac{\#\ \text{example correctly predicted as positive}}{\#\ \text{total positive examples in the test set}}$$



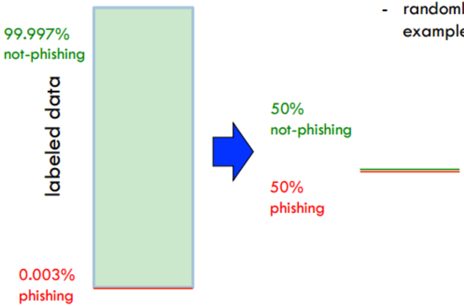
- Often there is a trade-off between precision and recall. Also these can be combined to yield other measures such as F1 score, AUC score, etc.

Dealing with Class Imbalance

- Modifying the training data (the class distribution)
  - Undersampling the majority class
  - Oversampling the minority class
  - Reweighting the examples
- Modifying the learning model
  - Use loss functions customized to handle class imbalance
- Reweighting can be also seen as a way to modify the loss function

Modifying the Training Data

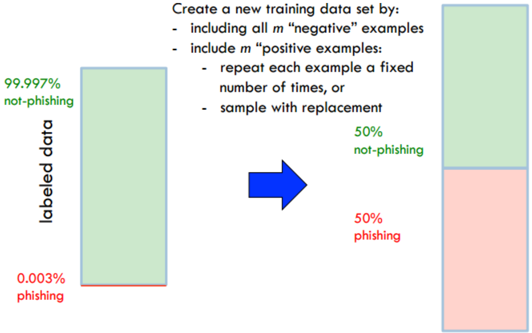
Undersampling



Create a new training data set by:
- including all *k* “positive” examples
- randomly picking *k* “negative” examples

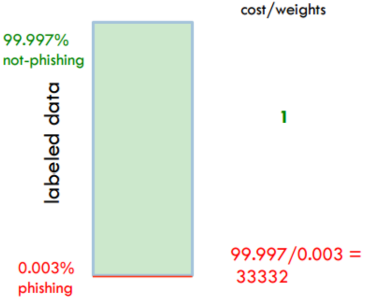
- Throws away a lot of data/information. But efficient to train

Oversampling



- From the loss function's perspective, the repeated examples simply contribute multiple times to the loss function
- Oversampling usually tends to perform undersampling because we are using more data to train the model
- Some oversampling methods (SMOTE) are based on creating synthetic examples from the minority class

Reweighting Examples



Add costs/weights to the training set

"negative" examples get weight 1

"positive" examples get a much larger weight

change learning algorithm to optimize weighted training error

- Similar effect as oversampling but is more efficient (because there is no multiplicity of examples)
- Also requires a classifier that can learn with weighted examples

Modifying the Loss Function

Loss Functions Customized for Imbalanced Data

- Traditional loss functions have the form: $\sum_{n=1}^{N} \ell(y_n, f(\boldsymbol{x}_n))$
- Such loss functions look at positive and negative examples individually, so the majority class tends to overwhelm the minority class
- Reweighting the loss function differently for different classes can be one way to handle class imbalance, e.g., $\sum_{n=1}^{N} C_{y_n} \ell(y_n, f(\boldsymbol{x}_n))$
- Alternatively, we can use loss functions that look at pairs of examples (a positive example $\boldsymbol{x}_n^+$ and a negative example $\boldsymbol{x}_m^-$). For example:

$$\ell(f(\boldsymbol{x}_n^+), f(\boldsymbol{x}_m^-)) = \begin{cases} 0, & \text{if } f(\boldsymbol{x}_n^+) > f(\boldsymbol{x}_m^-) \\ 1, & \text{otherwise} \end{cases}$$

- These are called "pairwise" loss functions
- Why is it a good loss function for imbalanced data?

Pairwise Loss Functions

- Using pairs with one +ve and one -ve doesn't let one class overwhelm other

$$\sum_{n=1}^{N_+} \sum_{m=1}^{N_-} \ell(f(\boldsymbol{x}_n^+), f(\boldsymbol{x}_m^-)) + \lambda R(f)$$

- The pairwise loss function only cares about the difference between scores of a pair of positive and negative examples (which is actually a good thing!)
  - Minimizing the above loss w.r.t. f give us an f that tends to give positive examples a higher score than the negative examples, which is similar in spirit to maximizing the AUC (Area Under the ROC Curve) score
  - AUC (intuitively): The probability that a randomly chosen pos. example will have a higher score than a randomly chosen neg. example
  - Empirical AUC of f on a training set with N_+ and N_- pos. and neg. ex.

$$\text{AUC}(f) = \frac{1}{N_+ N_-} \sum_{n=1}^{N_+} \sum_{m=1}^{N_-} \mathbb{1}(f(\boldsymbol{x}_n^+) > f(\boldsymbol{x}_m^-))$$

- Note: Commonly used pairwise loss functions act as a proxy of the (negative) AUC score (or of closely related measures such as F1 score)

Pairwise Loss Functions

- A proxy based on hinge-loss like pairwise loss function for a linear model

$$\ell(\boldsymbol{w}, \boldsymbol{x}_n^+, \boldsymbol{x}_m^-) = \max\{0, 1 - (\boldsymbol{w}^\top \boldsymbol{x}_n^+ - \boldsymbol{w}^\top \boldsymbol{x}_m^-)\} = \max\{0, 1 - \boldsymbol{w}^\top (\boldsymbol{x}_n^+ - \boldsymbol{x}_m^-)\}$$

- It basically says that the difference between scorees of positive and negative examples should be at least 1 (which is like a "margin")
- The overall objective will have the form

$$\frac{||\boldsymbol{w}||^2}{2} + \sum_{n=1}^{N_+} \sum_{m=1}^{N_-} \ell(\boldsymbol{w}, \boldsymbol{x}_n^+, \boldsymbol{x}_m^-)$$

- Convex objective (if using the hinge loss). Can be efficiently optimized using stochastic optimization (see "Online AUC Maximization", Zhao et al, 2011)
- Note: Similar ideas can be used for solving binary matrix factorization and matrix completion problems as well†
  - E.g., if matrix entry $X_{nm} = 1$ and $X_{nm'} = -1$ then loss=0 (or "small") if $\boldsymbol{u}_n^\top \boldsymbol{v}_m > \boldsymbol{u}_n^\top \boldsymbol{v}_{m'}$. E.g., $\ell(\mathbf{U}, \mathbf{V}, n, m, m') = -\log \sigma(\boldsymbol{u}_n^\top \boldsymbol{v}_m - \boldsymbol{u}_n^\top \boldsymbol{v}_{m'})$

† If interested, see: "BPR: Bayesian Personalized Ranking from Implicit Feedback" (Rendle et al, 2009)

Summary

- Imbalanced data needs to be handled with care
- Classification accuracies can be very misleading for such data
  - Should look at measures such as precision, recall, or other variants that are robust to class imbalance
- Sampling heuristics work reasonably on many data sets
- More principled approaches are based on modifying the loss function
  - Instead of minimizing the classication error, optimize w.r.t. other metrics such as precision, recall, F1 score, AUC, etc.
- Another way to look at this problem could be as an anomaly detection problem (minority class is anomaly) or density estimation problem