Class Imbalance

Anhthy Ngo

December 23, 2020

Contents

1 Overview

When modeling discrete classes, the relative frequencies of classes can have a significant impact on model performance. An imbalance occurs when one or more classes have very low proportions in the training data as compared to the other classes. Imbalanced classes pose challenges for predictive modeling as most algorithms used for classification were designed around the assumption of balanced classes. This results in models that have poor predictive performance, specifically for the minority class. This is a problem because typically, the minority class is more important to classify correctly (e.g., spam detection).

We see class imbalance in many domains such as:

1. **Food Safety:** The USDA Food Safety and Inspective Services (FSIS) tests food products for food-bourne illnesses. Food products at processing establishments that test negative outweigh the positive cases of food-bourne illnesses such as Salmonella or E. coli.
2. **Online advertising** An advertisement is presented to a viewer which creates an impression. The click through rate is the number of times an ad was clicked on divided by the total number of impressions and tends to be very low (reported rate less than 2.4%).

2 Problems with Class Imbalance

1. The minority class is harder to predict because there are fewer examples, by definition. This means it is more challenging for a model to learn the characteristics of examples from this class, and to differentiate examples from this class from the majority class (or classes).
2. Metrics like accuracy can be misleading. The abundance of examples from the majority class (or classes) can swamp the minority class. Thus, the model may naively focus on learning the characteristics of the abundant observations only (garnering high accuracy), neglecting the examples from the minority class that is, in fact, of more interest and whose predictions are more valuable.
3. Models often achieve good specificity (since the majority negative class is predicted correctly most of the time) but have poor sensitivity (recall).

3 Remedies

Threshold Moving The decision for converting a predicted probability is governed by a parameter referred to as the ”threshold”. The default value for the threshold is 0.5 for normalized predicted probabilities or scores in the range between 0 or 1. The most straightforward approach is to use the ROC curve since it calculates the sensitivity and specificity across a continuum of cutoffs. Using this curve, an appropriate balance between sensitivity and specificity can be determined. Several techniques exist for determining a new cutoff:

1. If there is a particular target that must be met for the sensitivity or specificity, this point can be found on the ROC curve and the corresponding cutoff can be determined. (e.g. we might care more about precision/sensitivity).
2. Find the point on the ROC curve that is closest (i.e., the shortest distance) to the perfect model (with 100% sensitivity and 100 % specificity), which is associated with the upper left corner of the ROC plot.

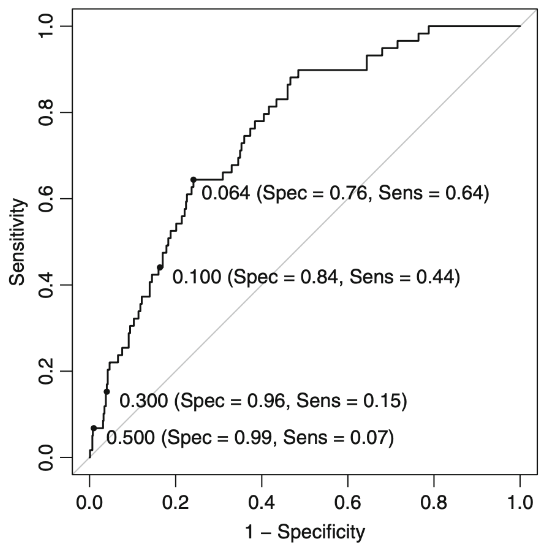

Figure 1: The random forest ROC curve for predicting the classes using the test set. Several possible probability cutoffs are used, including the threshold geometrically closest to the perfect model (0.064).

Resampling Methods Two approaches that can help attenuate the effects of the imbalance during model training.

1. **Up-sampling** is any technique that simulates or imputes additional data points to improve balance across classes (sampled with replacement from data set).

   - prone to overfitting due to repeated observations
   - no information loss

2. **Down-sampling** refers to any technique that reduces the number of samples to improve the balance across classes. We can down-sample by (1) randomly sampling the majority classes so that all classes have approximately the same size or (2) take a bootstrap sample across all cases such that the classes are balanced in the bootstrap set.
   - discarded observations could have important information
   - may lead to bias

4 SMOTE

SMOTE works by finding the k-nearest neighbor for each minority class and draws a line between the neighbor and the original point. It generates synthetic observations by linear interpolation of the minority class (randomly selecting a point on the line) and repeat until reaching the expected number of samples.

Algorithm:

1. Let $\mathbf{A}$ be the minority class set. For each $x \in \mathbf{A}$, the k-nearest neighbors of x are obtained by calculating the Euclidian distance between x and every other sample in $\mathbf{A}$
2. The sampling rate n is set according to the imbalanced proportion. For each $x \in \mathbf{A}$, n examples (i.e., $x_1, x_2, \ldots, x_n$) are randomly selected from its k-nearest neighbors, and they construct the set $\mathbf{A}_1$
3. For each example $x_k \in A_1$ $(k = 1, 2, \ldots, n)$, the following formula is used to generate a new example:
   $$x' = x + rand(0,1) * \mid x - x_k \mid$$
   in which $rand(0,1)$ represents the random number between 0 and 1

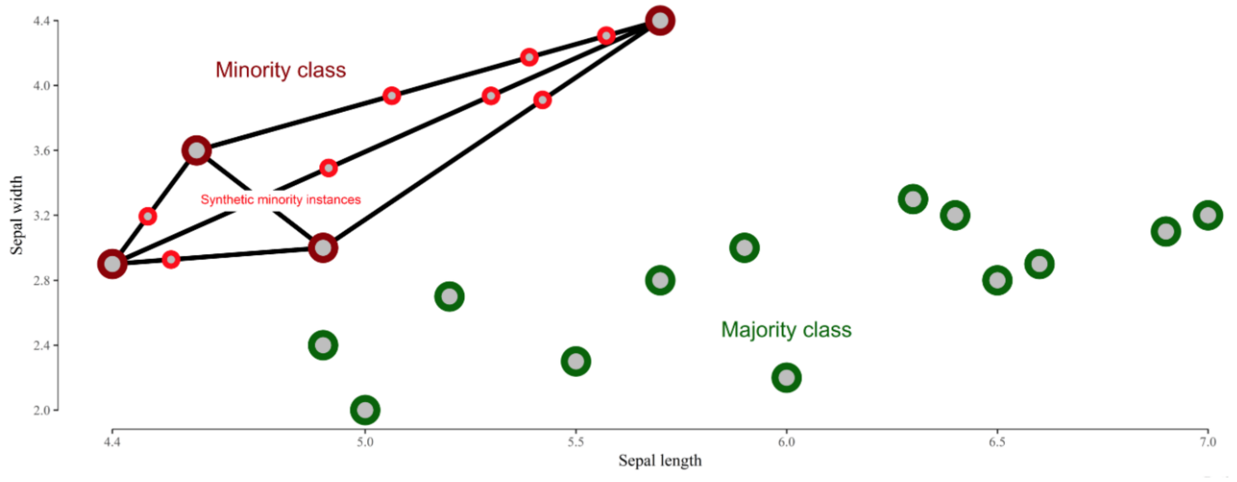

Figure 2: Larger maroon dots represent the original minority class, while larger green dots represent majority class. SMOTE generates synthetic observations from the minority class (smaller red dots) depending on specified sampling rate n.

4.1 Pros and Cons

Pros:

1. **Reduces Overfitting:** Instead of replicating already existing observations (random upsampling), SMOTE generates synthetic observations, thus alleviating overfitting.
2. **No Information Loss:** There is no down-sampling with SMOTE, so there is no loss of information.

Cons:

1. **Increased Noise:** SMOTE uses the KNN algorithm to generate synthetic samples, however, it does not consider that neighboring examples can be from other classes. This can increase the overlapping of classes and can introduce additional noise.
2. **Not Practical for High-Dimensional Data:** If variable selection is not performed than the KNN classification is counter intuitively biased towards the minority class, so SMOTE for KNN without variable selection should not be used in practice.

5 Managing Class Imbalance in Practice

1. **Do not evaluate with accuracy:** Accuracy assumes a naive 0.50 threshold to decide between classes, and this is usually wrong when the classes are imbalanced.

2. **Choose the right threshold:** When you generate probability estimates, do not naively use the 0.50 decision threshold to separate classes. Plot performance curves to decide what threshold to use.

3. **Use Stratified K-Fold [1]:** When the distribution of the label is severely skewed, it is likely one or more folds will have few or no examples from the minority class when using k-fold cross validation. Thus, we need to use stratified k-fold to enforce the class distribution of each split of the data to match the distribution in the complete training data set.

[1] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html

References

[1] Max Kuhn and Kjell Johnson. *Applied Predictive Modeling*. New York, 2013: Springer.

[2] N.V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. *SMOTE: Synthetic Minority Over-sampling Technique*. J. Artif. Intell. Res. 2002, 16, 321-357.

[3] Richard Kunert. SMOTE explained for noobs - Synthetic Minority Over-sampling. Nov 6, 2017.
https://rikunert.com/SMOTE_explained

[4] R. Blagus and L. Lusa. Evaluation of SMOTE for High-Dimensional Class-Imbalanced Microarray Data. 2012 11th International Conference on Machine Learning and Applications. Vol 2. 89-94.
https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-14-106