8 Tactics to Combat Imbalanced Classes in Your Machine Learning Dataset

by **Jason Brownlee** on August 19, 2015 in **Machine Learning Process**

Has this happened to you?

You are working on your dataset. You create a classification model and get 90% accuracy immediately. “Fantastic” you think. You dive a little deeper and discover that 90% of the data belongs to one class. Damn!

This is an example of an imbalanced dataset and the frustrating results it can cause.

In this post you will discover the tactics that you can use to deliver great results on machine learning datasets with imbalanced data.

Find some balance in your machine learning.

Photo by MichaEli, some rights reserved.

Coming To Grips With Imbalanced Data

I get emails about class imbalance all the time, for example:

I have a binary classification problem and one class is present with 60:1 ratio in my training set. I used the logistic regression and the result seems to just ignores one class.

And this:

I am working on a classification model. In my dataset I have three different labels to be classified, let them be A, B and C. But in the training dataset I have A dataset with 70% volume, B with 25% and C with 5%. Most of time my results are overfit to A. Can you please suggest how can I solve this problem?

I write long lists of techniques to try and think about the best ways to get past this problem. I finally took the advice of one of my students:

Perhaps one of your upcoming blog posts could address the problem of training a model to perform against highly imbalanced data, and outline some techniques and expectations.

Frustration!

Imbalanced data can cause you a lot of frustration.

You feel very frustrated when you discovered that your data has imbalanced classes and that all of the great results you thought you were getting turn out to be a lie.

The next wave of frustration hits when the books, articles and blog posts don't seem to give you good advice about handling the imbalance in your data.

Relax, there are many options and we're going to go through them all. It is possible, you can build predictive models for imbalanced data.

What is Imbalanced Data?

Imbalanced data typically refers to a problem with classification problems where the classes are not represented equally.

For example, you may have a 2-class (binary) classification problem with 100 instances (rows). A total of 80 instances are labeled with Class-1 and the remaining 20 instances are labeled with Class-2.

This is an imbalanced dataset and the ratio of Class-1 to Class-2 instances is 80:20 or more concisely 4:1.

You can have a class imbalance problem on two-class classification problems as well as multi-class classification problems. Most techniques can be used on either.

The remaining discussions will assume a two-class classification problem because it is easier to think about and describe.

Imbalance is Common

Most classification data sets do not have exactly equal number of instances in each class, but a small difference often does not matter.

There are problems where a class imbalance is not just common, it is expected. For example, in datasets like those that characterize fraudulent transactions are imbalanced. The vast majority of the transactions will be in the “Not-Fraud” class and a very small minority will be in the “Fraud” class.

Another example is customer churn datasets, where the vast majority of customers stay with the service (the “No-Churn” class) and a small minority cancel their subscription (the “Churn” class).

When there is a modest class imbalance like 4:1 in the example above it can cause problems.

Accuracy Paradox

The accuracy paradox is the name for the exact situation in the introduction to this post. It is the case where your accuracy measures tell the story that you have excellent accuracy (such as 90%), but the accuracy is only reflecting the underlying class distribution.

It is very common, because classification accuracy is often the first measure we use when evaluating models on our classification problems.

Put it All On Red!

What is going on in our models when we train on an imbalanced dataset?

As you might have guessed, the reason we get 90% accuracy on an imbalanced data (with 90% of the instances in Class-1) is because our models look at the data and cleverly decide that the best thing to do is to always predict “Class-1” and achieve high accuracy.

This is best seen when using a simple rule based algorithm. If you print out the rule in the final model you will see that it is very likely predicting one class regardless of the data it is asked to predict.

8 Tactics To Combat Imbalanced Training Data

We now understand what class imbalance is and why it provides misleading classification accuracy.

So what are our options?

1) Can You Collect More Data?

You might think it’s silly, but collecting more data is almost always overlooked.

Can you collect more data? Take a second and think about whether you are able to gather more data on your problem.

A larger dataset might expose a different and perhaps more balanced perspective on the classes.

More examples of minor classes may be useful later when we look at resampling your dataset.

2) Try Changing Your Performance Metric

Accuracy is not the metric to use when working with an imbalanced dataset. We have seen that it is misleading.

There are metrics that have been designed to tell you a more truthful story when working with imbalanced classes.

I give more advice on selecting different performance measures in my post “Classification Accuracy is Not Enough: More Performance Measures You Can Use“.
In that post I look at an imbalanced dataset that characterizes the recurrence of breast cancer in patients.

From that post, I recommend looking at the following performance measures that can give more insight into the accuracy of the model than traditional classification accuracy:

- **Confusion Matrix**: A breakdown of predictions into a table showing correct predictions (the diagonal) and the types of incorrect predictions made (what classes incorrect predictions were assigned).
- **Precision**: A measure of a classifiers exactness.
- **Recall**: A measure of a classifiers completeness
- **F1 Score (or F-score)**: A weighted average of precision and recall.

I would also advice you to take a look at the following:

- **Kappa (or Cohen's kappa)**: Classification accuracy normalized by the imbalance of the classes in the data.
- **ROC Curves**: Like precision and recall, accuracy is divided into sensitivity and specificity and models can be chosen based on the balance thresholds of these values.

You can learn a lot more about using ROC Curves to compare classification accuracy in our post "Assessing and Comparing Classifier Performance with ROC Curves".

Still not sure? Start with kappa, it will give you a better idea of what is going on than classification accuracy.

3) Try Resampling Your Dataset

You can change the dataset that you use to build your predictive model to have more balanced data.

This change is called sampling your dataset and there are two main methods that you can use to even-up the classes:

1. You can add copies of instances from the under-represented class called over-sampling (or more formally sampling with replacement), or
2. You can delete instances from the over-represented class, called under-sampling.

These approaches are often very easy to implement and fast to run. They are an excellent starting point.

In fact, I would advise you to always try both approaches on all of your imbalanced datasets, just to see if it gives you a boost in your preferred accuracy measures.

You can learn a little more in the the Wikipedia article titled "Oversampling and undersampling in data analysis".

Some Rules of Thumb

- Consider testing under-sampling when you have an a lot data (tens- or hundreds of thousands of instances or more)
- Consider testing over-sampling when you don't have a lot of data (tens of thousands of records or less)
- Consider testing random and non-random (e.g. stratified) sampling schemes.
- Consider testing different resampled ratios (e.g. you don't have to target a 1:1 ratio in a binary classification problem, try other ratios)

4) Try Generate Synthetic Samples

A simple way to generate synthetic samples is to randomly sample the attributes from instances in the minority class.

You could sample them empirically within your dataset or you could use a method like Naive Bayes that can sample each attribute independently when run in reverse. You will have more and different data, but the non-linear relationships between the attributes may not be preserved.

There are systematic algorithms that you can use to generate synthetic samples. The most popular of such algorithms is called SMOTE or the Synthetic Minority Over-sampling Technique.

As its name suggests, SMOTE is an oversampling method. It works by creating synthetic samples from the minor class instead of creating copies. The algorithm selects two or more similar instances (using a distance measure) and perturbing an instance one attribute at a time by a random amount within the difference to the neighboring instances.

Learn more about SMOTE, see the original 2002 paper titled "SMOTE: Synthetic Minority Over-sampling Technique".

There are a number of implementations of the SMOTE algorithm, for example:

- In Python, take a look at the "UnbalancedDataset" module. It provides a number of implementations of SMOTE as well as various other resampling techniques that you could try.
- In R, the DMwR package provides an implementation of SMOTE.
- In Weka, you can use the SMOTE supervised filter.

5) Try Different Algorithms

As always, I strongly advice you to not use your favorite algorithm on every problem. You should at least be spot-checking a variety of different types of algorithms on a given problem.

For more on spot-checking algorithms, see my post "Why you should be Spot-Checking Algorithms on your Machine Learning Problems".

That being said, decision trees often perform well on imbalanced datasets. The splitting rules that look at the class variable used in the creation of the trees, can force both classes to be addressed.

If in doubt, try a few popular decision tree algorithms like C4.5, C5.0, CART, and Random Forest.

For some example R code using decision trees, see my post titled “Non-Linear Classification in R with Decision Trees“.
For an example of using CART in Python and scikit-learn, see my post titled “Get Your Hands Dirty With Scikit-Learn Now“.

6) Try Penalized Models

You can use the same algorithms but give them a different perspective on the problem.

Penalized classification imposes an additional cost on the model for making classification mistakes on the minority class during training. These penalties can bias the model to pay more attention to the minority class.

Often the handling of class penalties or weights are specialized to the learning algorithm. There are penalized versions of algorithms such as penalized-SVM and penalized-LDA.

It is also possible to have generic frameworks for penalized models. For example, Weka has a CostSensitiveClassifier that can wrap any classifier and apply a custom penalty matrix for miss classification.
Using penalization is desirable if you are locked into a specific algorithm and are unable to resample or you’re getting poor results. It provides yet another way to “balance” the classes. Setting up the penalty matrix can be complex. You will very likely have to try a variety of penalty schemes and see what works best for your problem.

7) Try a Different Perspective

There are fields of study dedicated to imbalanced datasets. They have their own algorithms, measures and terminology.

Taking a look and thinking about your problem from these perspectives can sometimes shame loose some ideas.

Two you might like to consider are **anomaly detection** and **change detection**.
Anomaly detection is the detection of rare events. This might be a machine malfunction indicated through its vibrations or a malicious activity by a program indicated by it’s sequence of system calls. The events are rare and when compared to normal operation.

This shift in thinking considers the minor class as the outliers class which might help you think of new ways to separate and classify samples.

Change detection is similar to anomaly detection except rather than looking for an anomaly it is looking for a change or difference. This might be a change in behavior of a user as observed by usage patterns or bank transactions.
Both of these shifts take a more real-time stance to the classification problem that might give you some new ways of thinking about your problem and maybe some more techniques to try.

8) Try Getting Creative

Really climb inside your problem and think about how to break it down into smaller problems that are more tractable.

For inspiration, take a look at the very creative answers on Quora in response to the question "In classification, how do you handle an unbalanced training set?"
For example:

Decompose your larger class into smaller number of other classes…

…use a One Class Classifier… (e.g. treat like outlier detection)

…resampling the unbalanced training set into not one balanced set, but several. Running an ensemble of classifiers on these sets could produce a much better result than one classifier alone

These are just a few of some interesting and creative ideas you could try.

For more ideas, check out these comments on the reddit post "Classification when 80% of my training set is of one class".

Pick a Method and Take Action

You do not need to be an algorithm wizard or a statistician to build accurate and reliable models from imbalanced datasets.

We have covered a number of techniques that you can use to model an imbalanced dataset.

Hopefully there are one or two that you can take off the shelf and apply immediately, for example changing your accuracy metric and resampling your dataset. Both are fast and will have an impact straight away.

Which method are you going to try?

A Final Word, Start Small

Remember that we cannot know which approach is going to best serve you and the dataset you are working on.

You can use some expert heuristics to pick this method or that, but in the end, the best advice I can give you is to "become the scientist" and empirically test each method and select the one that gives you the best results.

Start small and build upon what you learn.

Want More? Further Reading…

There are resources on class imbalance if you know where to look, but they are few and far between.

I've looked and the following are what I think are the cream of the crop. If you'd like to dive deeper into some of the academic literature on dealing with class imbalance, check out some of the links below.

Books

- Imbalanced Learning: Foundations, Algorithms, and Applications

Papers

- Data Mining for Imbalanced Datasets: An Overview
- Learning from Imbalanced Data
- Addressing the Curse of Imbalanced Training Sets: One-Sided Selection (PDF)
- A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data

Did you find this post useful? Still have questions?

Leave a comment and let me know about your problem and any questions you still have about handling imbalanced classes.