Neural Net Classification Algorithm

Alexander Heckett

September 5, 2015

Abstract

This paper provides a relatively in-depth look at the algorithm used behind my neural net classifier. It does not cover the background corresponding to, the progression of ideas that led to, or any results coming from the program implemented.

Contents

1 The pre-computation

1.1 The Neural Net

1.1.1 Simulated Data

1.1.2 Training Scheme

1.2 Boosting

1.2.1 BrownBoost

1.2.2 LogitBoost

2 Per-dataset computation

2.1 Evaluation and Error

Having constructed our monster classifier, we now brace to use it. We have already laid out the general principle behind how it is to be used. For some starting position, look forward some fixed number of points into the future, where each point is separated by a constant number of frames, find the error bar at these points, and pass them into the neural net. We will thus generate a sequence of classifications, starting at every frame in the dataset except for the small slice of the set which would require looking at frames past the end.

It is now our task to turn these individual classifications into one final conclusion. However, this is not trivial. In an ideal world in which the master classifier would produce zero error, we would simply look at every classification, and if a star comes up with an exoplanet, record our identification. However, due to the many times we call our classifier, many misclassifications will occur. With this method, practically every star will come through as having an exoplanet, since at least one frame will be incorrectly identified as a false positive. Thus, we must contrive some algorithm to weed out the error.

Before proceeding to eliminate the error, we must define some expectation of how our error is going to behave. This model follows from two very basic propositions which I am now going to take as axioms:

1. The likelihood of one star to be misclassified is not dependent on another star's likihood.

2. The stars are all treated the same way by the master classifier.

These two propositions form the basis of my theory. Due to these two axioms, the probability that a truly exoplanet free star comes out incorrectly positive is some constant number - say α. The probability that a star with an exoplanet comes out incorrectly negative is some other constant number - say β.

2.2 The Decoder

2.3 Finding Epsilon

Up until now, I have assumed we somehow knew ϵ. However, there is no obvious way to predict this from theory. We thus must resort to using the dataset to find the most probable value of it. This is similar in concept to finding σ in my Dynamic Binning algorithm, just harder to implement.

The program has been tasked with the problem of, given the classifications in an interval, find the best ϵ to work with. Let's start by working this problem in reverse: if we knew ϵ, what would be the distribution of classifications? At this point, we must define some more terminology before we can proceed.