Classifying Fish with Machine Learning

Catherine Hastings

Introduction

We aimed to use machine learning to optimize the classification of species of fish based upon photographs submitted to Kaggle, as described here [1]. Having an algorithm which could accurately classify fish from photographs or video footage would be a great help in keeping track of fishing quotas and therefore could improve sustainability within the fishing industry. The problem with classifying fish from photographs manually is the sheer quantity of data produced. That is why this task is a perfect application for machine learning; a successful algorithm could drastically reduce the human effort, and the large amount of data required for machine learning is readily available.

Kaggle provided two sets of photographs: a training set, split into different classes and a test set with unlabelled photographs. There were six different species of fish specified, as well a class for any other fish and another for photographs with no fish in at all, giving eight classes in total. The photographs were of varying quality; the fish were in different locations within the photographs, of inconsistent sizes and the photographs were taken in very varied light levels. There were also different numbers of photographs for each class, though all the photographs had the same dimensions.

Creating the Classifier

Throughout, our classifiers were created using Mathematica. We also reduced the dimensions of the photographs from 1280 by 720 to 256 by 144 pixels on importing.

A First Attempt

Our first step was to go through the process of creating a very basic classifier. This meant that we could ensure that all the data was being imported correctly, that the training data was appropriately organized and labelled for the Mathematica `Classify` function, that the classifier created could operate on the test data and that the results were formatted so that they could then be submitted to Kaggle. Consequently, the accuracy of the first classifier was not of high priority. Of greater importance was that each step could be performed quickly, so that we could then move on to optimizing the classifier.

Our first classifier was trained upon a training set of reduced size. This was to reduce the import time of the photographs. It also ensured that the training sets for each class were of equal size. In this first step, we did not alter the settings of the Mathematica `Classify` function, and therefore the logistic regression classification method was chosen as a default. This classifier was tested upon

all the test data. The results were uploaded to Kaggle on 12/12/2016 at 14:22:35 and the classifier achieved a score of 1.84094.

Using All the Data

We next tried training our classifier on the whole of the training set. This meant that the training sets for each class were no longer of equal size, but greatly increased the total size of the training set (from 536 to 3777 photographs). For this second classifier, we likewise made no alteration to the Mathematica `Classify` function. However, based upon the new larger training set, Mathematica instead implemented the nearest neighbours classification method. Again the classifier was tested upon all the test data, and was uploaded to Kaggle on 14/12/2016 at 12:57:28, achieving a score of 1.50857.

Testing Classifier Methods

In order to compare the different classification methods, we split the training data in two, giving 80% for training the classifier and then testing it on the remaining 20%. It would have been better to repeat this process, testing the data on each mutually exclusive fifth. However, it took approximately 15 minutes to train each classifier and then another 4 minutes to classify the data. Performing a 5-fold cross validation for 7 different classification methods would have taken too much time. Cross-validation reduces impact of randomly choosing a heavily skewed training or test set, by averaging over multipe sets.

We took the most probable of the classifier's predictions for each photograph of the test data, and compared this against the correct answer, giving us a binary "correct" or "incorrect" response. We plotted the results (Fig. 1) as percentage success rates.

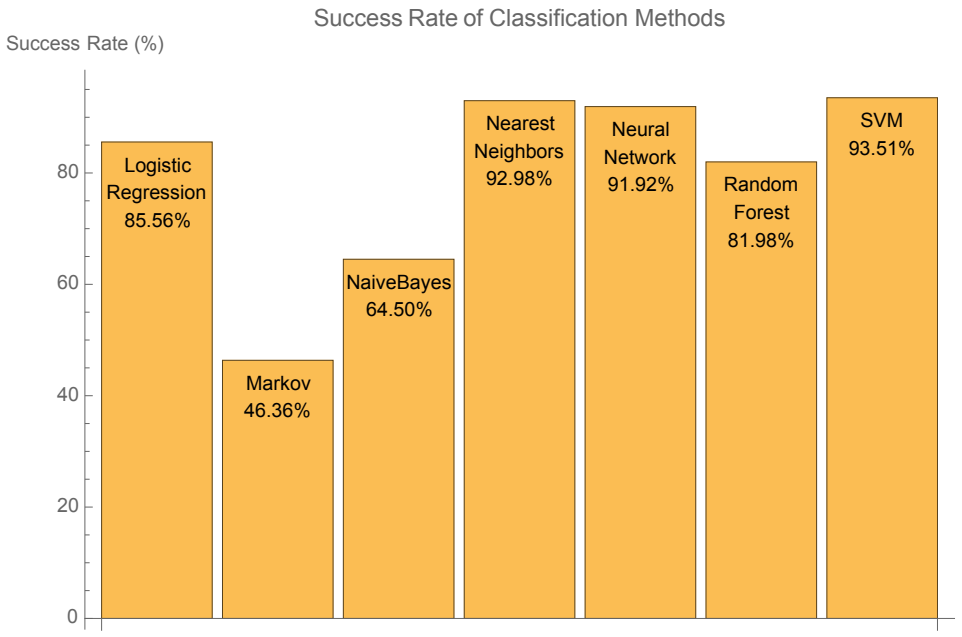

Figure 1: A bar chart comparing the success rates of different Mathematica classifier methods.

Based on this data, we created a classifier which was trained on all the training data, but which implemented the method `"SupportVectorMachine"`. We tested it upon all the test data and uploaded it to Kaggle on 16/12/2016 at 10:01:41. However, the classifier only achieved a score of

2.07043. This implies that the classifier was overtrained, and that there must be some significant difference between the training data set and the test set.

According to our tests, the second best performing algorithm is the nearest neighbours method. We already implemented this method during stage two of optimizing our classifier, and it received a score of 1.50857. Therefore, we trained additional classifiers using the methods `"NeuralNetwork"`, `"LogisticRegression"` and `"RandomForest"` and they achieved Kaggle scores of 1.94603, 1.72735 and 1.41165 respectively.

These results did not correlate at all well with the performance data we produced for each of the classifiers. This could have be an artefact of overtraining, meaning that the classifiers were making prediction on some feature of the photographs unrelated to the fish, so that when presented with a completely new set of photographs they under-perform.

Conclusions

We succeeded in creating an algorithm for classfying different species of fish using Mathematica's in-built `Classify` function. A submission to Kaggle with uniform probabilities for all classes and all photographs receives a score of 2.07944. This means that all the classifiers we created performed better than random chance.

We observed that increasing the size of the training set had a positive impact on the performance of the classifier. Though we cannot say that this would be the case in every circumstance. If the training set and the test set are significantly different, then increasing the size of the training set will lead to overtraining.

We also found that the method of classification had a significant impact on the success of the classifier. We found that classifying with a random forests algorithm gave best results. However, we also found that testing the binary success rate within the training data did not prove a good predictor of classification method performance. This could be due to overtraining. It is also possible that the performance metric used makes a difference. Kaggle implements a log-loss scoring system based upon all of the probabilities submitted, as opposed to a binary method. We could check this hypothesis by implementing Kaggle's scoring method alongside the binary one to see whether it gives a more accurate predictor of performance for the test data set.

References

[1] kaggle.com. The nature conservancy fisheries monitoring - problem introduction. `https://www.kaggle.com/c/the-nature-conservancy-fisheries-monitoring`. Accessed 2016-12-14.