**Image Classification & Security Applications**

Nicholas Lytal

991834259

1. INTRODUCTION
2. DATA MANAGEMENT
3. CLASSIFICATION METHODS
4. RESULTS
5. CONCLUSIONS
6. FUTURE IMPROVEMENTS

**1. Introduction**

Many people are familiar with the existence of CAPTCHAs—miniature Turing tests designed to distinguish humans and computers. Most frequently, if someone wants to access a sensitive webpage they will receive a picture of some distorted letters and numbers, and required to enter them correctly to proceed. Ideally, this protects against computers gaining unwanted access by presenting a challenge only a human can solve, but advances in image identification algorithms have become increasingly effective at identifying such images automatically.

An alternative to CAPTCHA exists, known as Asirra. With this system, those who wish to access a webpage receive pictures of dogs and cats instead of letters and numbers, and prompted to identify ten such animals in a row. While more complex than the CAPTCHA system, it may still be possible to build an algorithm that can distinguish cats from dogs a reasonable portion of the time and answer accordingly. If such an algorithm could exist, it may compromise the effectiveness of Asirra, requiring security programmers to explore another alternative.

Our goal in this project is to construct three different types of classifiers that can distinguish cats from dogs based on features present in each image. We can use both linear regression and random forests to construct a classifier that gives a better than average chance of identifying a picture correctly.

**2. Data Management**

Before we can construct our classifiers, we must first convert the data into a usable form. Our initial task is to standardize all the available images so we can use them to construct a classification matrix. We first convert each image in our collection to a 250x250 jpeg. Once the images are standardized, we can use the “jpeg” and “grDevices” packages in R to convert a given image’s data into HSV (hue, saturation, and value) values that will form our color features. An image thus converted becomes a 250x250x3 matrix array, with one 250x250 matrix of values ranging from 0 to 1 corresponding to each feature (H, S, or V).

Using these results, we can split each picture into a number of horizontal and vertical strips—in our case, five strips of size 50 pixels each. This gives us 25 separate cells per picture, which we further divide by breaking up each HSV component into eight separate blocks that cover the entire range of values. We then establish a binary vector that indicates whether any number of the pixels in a given cell fall into the HSV divisions established. The resulting vector consists of 8^3 HSV level combinations per cell, forming a vector of length 12800. Once all pictures are converted into this form, the resulting 25000x18000 matrix will represent the color features for our entire data set. We use this matrix to construct our classifiers.

**3. Classification Methods**

With our pictures converted, we proceed to sample training and test data from the pictures, with an equal number of cat and dog pictures for each set. We then use the training set to build our classifiers before applying them to the test data to measure their effectiveness at classifying unknown images. Of the many methods available to us, we will focus on linear regression and random forests.

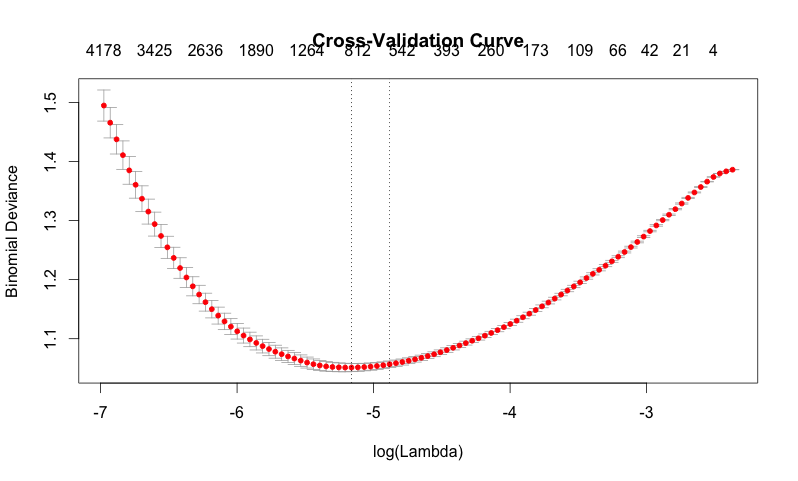
Generalized linear regression considers our data as a series of observations in p-dimensional space (where p is 12800 in this case). With regression, we can construct the p-dimensional plane that most effectively separates dog observations from cat ones. However, since p is effectively larger than our training data set of 12500 images, our sample size is insufficient to use regression as is. Instead we must use penalized regression to fit our general linear model by penalized maximum likelihood. The R package “glmnet” allows us to efficiently use sparse matrices like our color features matrix, which have only a few non-zero entries among many zeroes, to perform the necessary regression. The formula[[1]](#footnote-1) it uses for regression is:

where l represents the negative log-likelihood, and the formula above is tested over a large range of penalty terms λ. Once an ideal λ is clear, we then predict on the test data using our chosen λ to most effectively identify the pictures in question.

Random forests are based on a simpler ensemble method known as “bagging” or bootstrap aggregating. Bagging involves the construction of many decision trees, each of which attempts to classify data into separate groups based on the value of certain variables in a data set. By determining the most important and influential variables of the many available, such decision trees attempt to make the best divisions among the data, and we can average the findings of the resulting trees to obtain a classifier for our data. Random Forests expand on this method by using a random selection of the existing variables at each tree’s decision splits to avoid overfitting to the training data. By using a different selection of variables for each tree, including ones that would not normally be chosen by the bagging process, we can get a more complete picture of the data, and over time we can acquire a more effective classifier.

**4. Results**

When designing our fit for penalized regression, we use cross-validation to obtain the ideal λ on which to predict our test data and we get the following results:



The initial fit suggests we should use a tuning parameter of about λ = 0.00573, or log(λ) = -5.16. By applying this tuning parameter to the prediction, we then predict the test data with the following accuracies:

ACTUAL

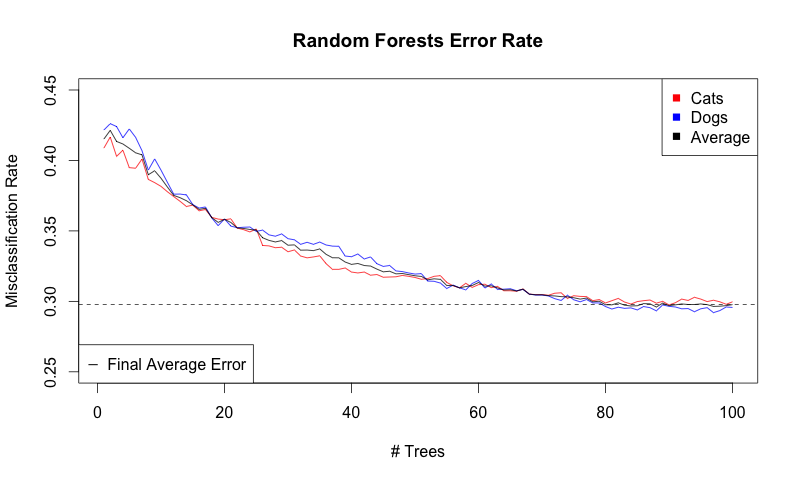
cats dogs

PREDICTED cats 4801 1449

dogs 1821 4429

This works out to about a 73.8% chance of identifying pictures correctly.

When using random forests, we can get progressively more accurate results (to a certain extent) the more trees we consider, and we can observe the prediction accuracy for the first 100 trees constructed:



After 100 trees, we achieve about a 70% chance of correctly identifying a single picture. We could construct additional trees beyond 100, but the gains are likely to be minimal and would not be worth the additional time. Just 100 trees take over an hour to construct with default settings, and even after that length of time, the end result is less effect than using penalized linear regression, which takes only about 18 minutes in comparison.

**5. Conclusions**

Taking our penalized regression classifier as our best result, we obtain about a 73.8% accuracy rate at identifying a cat or dog picture correctly. Though this is not particularly reliable, it is a significant improvement over random chance, and over 100 times more likely to succeed at completing a test run of twelve consecutive pictures. Though this is still only a 2.6% chance of success for such a run, even this percentage may well be large enough to reconsider implementing the Asirra system if a relatively straightforward classifier can yield such a result. With a few improvements, the success rate of such algorithms could be a credible threat to the security of the system.

**6. Future Improvements**

The classification matrix used consisted of only one set of parameters. With considerably more time and a revised, more efficient algorithm, we could compare the effectiveness of classification matrices with a different number of cells or HSV component divisions. Theoretically, a larger number of cells would allow for a more accurate representation of the color features within, though the size of the matrix would be considerably greater. Naturally, a larger number of pictures for our training data would provide a better basis for the classifiers, though alone they will be limited by the methods used.

Color features are only one aspect of the image. We could in addition consider texture features, which depict actual shapes in small portions of the images rather than just coloration. By constructing a large collection of reasonably disparate texture cells, we could compare each of the training images to the collection and record a vector of “distances” between the pixels of the image and the collection’s tiles in terms of RGB values.

Other classifiers may offer a more effective means of identifying pictures correctly. For instance, support vector machines were the classifier of choice mentioned by the original Kaggle project, and could offer improvements over both penalized linear regression and random forests. There may well be other methods even more effective at classification if time constraints are not an issue, such as deep neural networks, that would be worth pursuing as well.

1. Formula taken from the Glmnet Vignette at http://www.stanford.edu/~hastie/glmnet/glmnet\_alpha.html [↑](#footnote-ref-1)