10-601 Midway Report

Jacob Buckman
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
`jacobbuckman@cmu.edu`

John Corbett
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
`jtcorbett@cmu.edu`

1 Introduction

The goal of this project is to devise a machine learning classifier that performs well on the CIFAR-100 dataset. This dataset contains $50,000$ labeled images divided into 100 classes and 20 super-classes. To classify these images we will try three different machine learning approaches: a Gaussian Naïve Bayes classifier (GNB), a Neural Network (NN), and a Random Forest (RF). We will then withhold a subset of the training data, train each of the classifiers on the remaining data, and see how it scores on the withheld data. This process is known as cross-validation. We aim to achieve better than baseline accuracy[1] with at least one of the aforementioned approaches.

2 Methods

Since the images that we are classifying are only 32×32 pixels, take each pixel to a feature for training the classifiers for a total of 3072 features. Before running the classifiers on the image data, it may be useful to run feature extraction which will train the classifiers on the features of the images rather than the raw image data. We will have a preprocessing step where feature extraction occurs, then train the classifiers on the preprocessed data rather than the raw data. In addition to training the classifiers on the raw data (RGB pixel values of the image), we will also train it on the HSV pixel values of the image, the vectorized HOG descriptor of the image obtained using the `VLFeat` library, the regional average pixel color and number of edges and corners detected.

Given that the CIFAR-100 dataset contains a large about of images and these classifiers take a long time to train and to run, when determining the efficiency of the classifiers, we will operate on the CIFAR-10 dataset under the assumption that success in classifying this dataset will directly translate to success in classifying the CIFAR-100 dataset.

Gaussian Naïve Bayes

We choose to use a GNB classifier over a regular Naïve Bayes classifier because even though the data points are discrete, they can take on a wide range of values in a continuous fashion which lends itself more to the Gaussian version of the classifier. The GNB classifier was trained with and without feature detection and tested with cross-validation withholding 5% and 10% of the dataset. We use a maximum likelihood estimation when training GNB, so there is no fabricated data and the resulting model will maximize the likelihood of the training data occurring.

GNB relies on the assumption that each of the features are independent of each other. Given that this is clearly a faulty assumption, we do not expect that this classifier will perform as well as the others, but it given that it runs very quickly, we can use this classifier to quickly test the efficiency of other parts of our model such as feature extraction. Our implementation of GNB does *not* assume that the variance is independent of the training samples or labels.

[1] We will take baseline accuracy to be 48%.

Neural Network

One approach that was very popular in the literature of image classification was neural networks. We implemented a basic neural network framework with logistic nodes, trained by backpropagation with stochastic gradient descent. The implementation allowed for a flexible number of hidden layers, each of which is fully connected to the next hidden layer. The output layer of nodes is 10 nodes in size, each node representing one category. For example, an image in class 2 would be trained on the output node weights of [0 1 0 0 0 0 0 0 0].

The network is trained in batches. After each batch of training points have been calculated, the average of their gradients is added to the weights. After all samples of training data have been placed into a batch, and the network has been updated with all of these weights, the network is evaluated. This is done by randomly sampling a number of training data points from the overal set, and calculating the network's accuracy on these points. The training continues until this accuracy passes a certain threshold, or a maximum number of epochs pass.

3 Results

As expected, the Neural Network greatly outperformed the GNB in terms of accuracy, but the GNB performed much faster in both the training and the classifying steps. Interestingly, all classifiers performed better without feature detection.

Gaussian Naïve Bayes

Without feature extraction, GNB was able to correctly classify just over 20% of samples. This is about 10% better than random guessing, but is still significantly below baseline accuracy. With feature detection (HSV and vhog), this accuracy drops to about 14%, only slightly better than random guessing.

Neural Network

Due to time and computation constrains, we were unable to fully evaluate the optimal combination of parameters for the neural network. However, the best combination of features we were able to find was input of raw data (no feature extraction), a network with the following set of layers: 500-250-100. The learning rate was sent to .5, the local-maximum threshold .01, the batch size 500, and the evaluation set size 100. It was trained for 20 epochs on a data set of 10000 points. This gave an accuracy of 24% on the development set. This is still below the baseline, but this number will likely increase as further optimizations are applied.

4 Conclusion

It is too early to draw any conclusions, but our initial results look promising. We hope to have a more fully developed set of algorithms by the final project deadline.

5 Contributions

Jacob implemented the Neural Network and designed a testing script that allowed for easy comparison between the different algorithms. John implemented the Gaussian Naïve Bayes Classifier as well as the Random Forest and the HSV feature extractor. Both team members worked equally on this writeup.