Abstract

As a large e-commerce retailer of goods across the world, the Otto Group desires consistent, accurate analysis of the performance of their large array of products. Given a large dataset with a high dimensionality of features, we seek to build an algorithm that accurately classifies these products within reasonable amounts of time. The high dimensionality of the features encouraged us to explore the efficacy of multiple dimensionality reduction techniques. Comparisons of these techniques were tested with several popular machine learning algorithms namely, support vector machines (SVM), Decision Tree algorithms, and multi-layered neural networks.

Introduction

The Otto Group is one of the world’s biggest e-commerce companies existing as a conglomerate of subsidiaries in more than twenty countries across the world. Being a large company, they created a competition on Kaggle [1] with a dataset containing information of their vast array of products hoping to attract talented data scientists to construct prediction models that consistently and accurate classify their products. We decided to try our hand at the problem with the dataset. The competition included a labeled training set and an unlabeled test set. Because the provided test set lacked labels, we could not use it to test our algorithms requiring us to allocate a section of training data as our labeled test data. Additionally, they left out the descriptions of the features and labels making it more difficult to intuitively make selections about certain features and assumptions about the data.

This paper will focus on the results of using three different machine learning algorithms: support vector machines, decision trees/random forests (along with other variations), and multi-layered neural networks. Given the high dimensionality of the dataset, we also chose to implement several dimensionality reduction algorithms for comparison: principal component analysis (PCA), information gain, correlation, and variance.

Dataset

The raw training data includes 61878 samples with 93 integer valued features. There are a total of 9 different classes. We used 70% of the raw training data as a training set while the remaining 30% are used as the test set. After separating the sets, we randomized the order of the data.

Multi-Layered Neural Network

The first algorithm we used for supervised classification was the multi-layer perceptron (MLP). The MLP is commonly recognized as a feedforward artificial neural network due to its single directional flow of data between the input and output. Unlike recurrent neural networks, the data in an MLP never returns to previous layers and only flows in one direction. For purposes of implementation, the MLP was implemented using Python 3.4 with Theano (0.8.2)[2]. While Python has quite a few different packages for machine learning applications, we chose Theano for an MLP implementation due to its optimized code and increased performance. Development initially started with a single hidden layer MLP that was tested with multiple activation functions. On the top layer, we implemented a logistic softmax regression classifier. The hidden layers that fed into the top layer were comprised of either sigmoid or tanh functions. For the final comparison, we developed a double hidden layer MLP with the first hidden layer containing tanh functions while having the second hidden layer being comprised of sigmoid functions.

Softmax Regression and Parameter Tuning

Due to lack of a priori knowledge about the labels and features, we felt that a logistic regression top layer using stochastic gradient descent would work as a decent classifier since it is less prone to overfit and its overall simplicity. The output of a softmax regression classifier is a probability associated with a specific class which falls in line with the nature of the problem and the dataset. Given our training set, we chose to create a validation set of 20% of the training data which was used refining the weights. After testing with multiple learning rates, we noticed that a learning rate of 0.05 produced the most optimal results.

After implementing the softmax layer, we moved on to adding the hidden layers. When testing the different available hidden layer functions, we tuned the parameters to have 500 hidden units in a hidden layer with a learning rate of 0.005. Additionally, we added an L1 and L2 regularizers to prevent overfitting with values 0.00, and 0.0001 respectively.

Results

When the dataset was run through the softmax regression layer outside of the MLP, the accuracy ranged from ~73-76%. When it came to the single layer MLP, there was noticeable improvement in accuracy (in comparison to the softmax regression layer). Depending on the activation function, the MLP had a lower bound accuracy of ~77.5% and an upper bound of almost 80%. The double hidden layer MLP boasted in even higher accuracy breaking the 80% threshold. However, the accuracy jump was not much higher than a single hidden layer MLP.

The results of the neural network algorithms in conjunction with the dimensionality reduction algorithms were quite interesting. Information gain and variance techniques produced consistent results almost regardless of the number of features used in each algorithm. In fact, there were some cases where the classifiers were more accurate with a lower dimension than with a higher one (e.g softmax regression). Correlation did not fair as well as the other two algorithms but the accuracy lost was not very significant.