Comparing Machine Learning Approaches for Classifying songs

***Abstract*—We taught a deep multilayered perceptron to categorize thousands of songs to ten different genres. Training was done with 4363 songs that each were transformed to 246 features. The deep neural network learned to give songs a probability the song is part of each class. We could then choose the most likely class for a song. This is useful as a way to group music into categories that can be later used for recommendation or discovery. We tried 5 different machine learning approaches to classifying the data. Finally we managed to tune MLP Classifier to accuracy of 65.45%.**

***Keywords*—multi-layer perceptron, scikit-learn, neural network. gaussian naive bayes, k-nearest neighbors, linear support machine**

I. Introduction

We learned how to use neural networks with classification task and tested how they compete against other machine learning methods. To achieve this, we familiarized ourselves with 5 different classification methods: gaussian naive bayes, k-nearest neighbors, linear support vector machine, C-support vector machine and multi-layer perceptron (MLP) -classifiers. Out of these we focused on tuning the MLP classifier to its best performance. Tuning the classifiers required us to familiarize with regularization (L2 regularization), various neural network activation functions as well as different methods for optimizing weights in neural network (e.g. SGD and Adam).

A. Data analysis

The data consisted of 264 features and 10 classes. Features were normalized by removing the mean and scaling to unit variance before testing.

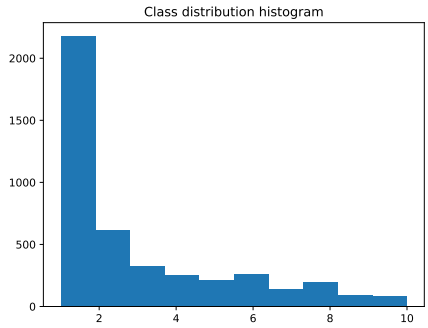

Fig. 1. Class distribution histogram

As can be seen from Fig. 2 the most of the of the training data belonged to pop rock class.

II. Methods and experiments

Initially we approached the problem by comparing how multiple classifiers worked with our data. Since we used scikit learn python library for our data analysis it was easy to test out how different classifiers performed. We tried following classifiers: gaussian naive bayes, k-nearest neighbors, linear support vector machine, C-support vector machine and multi-layer perceptron neural network -classifier. Validation of methods was done by splitting the shuffled training data into 80% part for training and fitting classifiers and 20% for validating. In the following subsection each classifier is discussed in more detail and subsequent subsection the procedures in tuning multi-layer perceptron neural network -classifier are discussed in length.

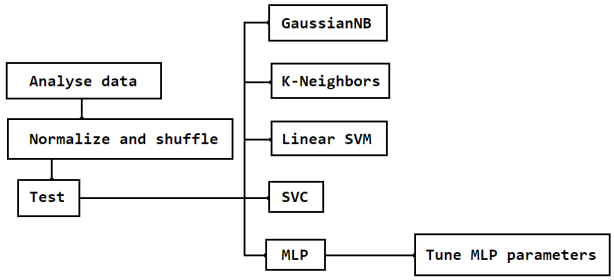

Fig. 2. Block diagram of our workflow

A. Comparison of Classifiers

Gaussian naive bayes classifier uses bayes theorem to obtain maximum posteriori probability that is then used for classifying data into appropriate classes. In this method it is assumed that continuous values in the data associated to each class fit Gaussian distribution. Fitting the data and classifying with it gave training accuracy score of 30% and testing accuracy score of 29%. These values were not particularly good so we did not investigate gaussian naive bayes classifier further.

K-nearest neighbor vote classifier uses a non-parametric method (type of instance-based learning) for classifying data into classes based on k nearest neighbors in feature space. We used value 9 for neighbors as this gives a good balance between noise suppression and class boundary distinction. Method yielded training accuracy of 65% and testing accuracy of 59%. This result was relatively good but did not as good as other classifiers that we tried.

The basic idea behind support vector classifier is maximizing the margins between classes. We tested linear support vector classifier with hinge loss function L2 regularization $\alpha = 0.0001$ and stochastic gradient descent of learning rate

$\eta = \frac{1}{\alpha*(t+t_0)}$ where t_0 was chosen by heuristic proposed by Leon Bottou. This was the default configuration of SGD classifier offered by scikit learn library. It performed relatively well with training accuracy of 70% and testing accuracy of 64%.

We also tested classification with C-Vector Classifier with penalty parameter $C = 1$, radial basis function kernel with parameters $\gamma = \frac{1}{n_{features}}$ and tolerance of stopping criterion of value 1e-6. This was also the default configuration of SVC classifier offered by scikit learn library. This resulted in training accuracy of 76% and testing accuracy of 65%.

Finally we tested the multi-layer perceptron neural network -classifier with a single hidden layer of 8 perceptrons size. Rectified linear unit activation function ($f(x) = max(0, x)$) was used. Weights were optimized with stochastic gradient based optimizer called Adam proposed by Kingma, Diedric and Jimmy Ba. initial learning rate was set to 0.001, L2 regularizer parameter was set to $\alpha = 0.0001$ and tolerance for optimization was set to 1e-10. This setup yielded training accuracy of 78% and testing accuracy of 64%.

Although the support vector machine classifiers equally well or better than the MLP classifier at the testing score, MLP classifier had better testing accuracy. Capacity of MLP classifier was relatively easy to increase by increasing number of perceptrons in hidden layer and number of hidden layers to the point where training accuracy reached 100% but testing accuracy lowered to around 60% (clear case of overfitting). This meant that there was a lot of potential for this classifier if tuned well. Following subsection discusses different approaches our group took for tuning the MLP classifier.

B. Tuning of Multi-layer Perceptron Classifier

Common approach to tuning neural network is to first increase capacity of neural network so that the training error is as small as possible and then to start regularization until testing error is as good as possible. We tried it at first by increasing the number of perceptrons in the hidden layer up to 24. This was the point where accuracy of the training error reached 100% but accuracy was as low as 59%. Next we started increasing the L2 regularization term α until the testing error was as high as possible. With α value of $\alpha = 0.72$ training accuracy of 87% and testing accuracy of 65.63% was achieved.

We also tried increasing the capacity of the neural network up to 1000 perceptrons in hidden layer and had the training stop when validation error of based on 10% of training data was not increasing anymore. This method gave the better result than L2 regularization approach above with training accuracy of 79% and testing accuracy of 67.23%. Of course because our validation data was quite small the this method did not perform as well in the kaggle competition (kaggle accuracy test yielded accuracy of 0.654). Increasing hidden layer size to 10000 perceptrons did not improve the result.

Some testing was also done on making the neural network deeper. We tried regularization and early stopping approach with 2 hidden layers of 100 perceptrons each as well as 4 layers with 100 perceptrons each. These two approaches did not improve the results. Typically early hidden layers have more neurons than the deeper layers so also a setup of 3 layers with first layer having 132 perceptrons, second layer having 66 and third layer having 33 perceptrons (all nicely divisible by the 264 features we were analyzing). This setup gave training accuracy of 70% and testing accuracy of 65.86% when run with L2 regularization term $\alpha = 6.0$.

After testing with all these setups we decided that best results were achieved with one hidden layer of 1000 perceptron and early stopping. This was the method we used when returning our result for kaggle evaluation.

III. Results

Our results were measured using accuracy and LogLoss (results acquired from Kaggle):
$Accuracy$: 0.65464
$LogLoss$: 0.17756

These scores were acquired using MLP classifier. To compare against other classification methods we used provided test data. After training different models with training data we compared test score (locally) and chose the best model for the Kaggle competition. MLP classifier achieved the best result and we were able to submit its classification results to Kaggle. After two submissions we got the results shown previously.

From confusion matrix in Fig. 3 we can see the data was mostly centered around pop rock since most of the labels belonged to this class. Labels with the highest misclassification rates are definitely Blues, Reggae and International as can be seen in the figure.

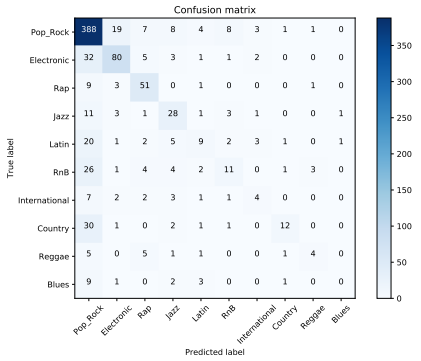

Fig. 3. Confusion matrix

IV. Conclusion

The results were surprisingly good considering how hard it is to tell some classes apart. Completely random guesses would yield accuracy of 1/10 and our neural network surpassed that more than six times.

References

[1] http://scikit-learn.org/stable/index.html

[2] http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html

[3] https://en.wikipedia.org/wiki/Naive_Bayes_classifier

[4] http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

[5] http://scikit-learn.org/stable/modules/neighbors.html#classification

[6] http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html

[7] http://scikit-learn.org/stable/modules/sgd.html#sgd

[8] http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC

[9] http://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

[10] http://www.faqs.org/faqs/ai-faq/neural-nets/part1/preamble.html