Problem Statement

With the relatively recent boom in music streaming and digital music sales, interest in music data mining has steadily increased. Audio, in the form of waveform data files, can be segmented into numerous features such as pitch, tempo and loudness, as well as contrived metrics like “danceability” and “energy”. Machine learning analysis of these features is of great interest to the industry for determining things like target market demographics and suggested songs lists.

For our project, we applied various classifiers to the problem of classifying songs by the decade in which they were released. We chose to split classes up by decade, as performing classification by discrete years unsurprisingly warranted very poor results. The dataset for the problem is Columbia’s Labrosa laboratory’s “1 million song dataset”. Of the 1 million songs, 515,345 had known years associated to them, although only a subset were used for training purposes due to time constraints. Splitting this dataset, we had 10 classes between 1920’s-2010’s, with the 2000’s being the most common choice. We used 3 classifiers detailed below: Multiclass SVM, AdaBoost and a neural network.

Baseline

If a classifier were to guess 2000’s every time, it would achieve 58.02% (299003 / 515345) accuracy. We are using this as our baseline.

SVM

Dataset: The dataset was split into training, validation and test sets. Training and validation sets together contained the first 463,715 samples while test set contained the last 51,630 samples. This split was recommended on UCI’s website to avoid producer effect which makes sure that no song from a given artist ends up being used for both training and testing.

SVC and NuSVC implement 1 vs 1 approach for multi-class classification i.e. if n is the number of classes, then n * (n - 1) / 2 models are constructed where as LinearSVC implements 1 vs rest multi-class strategy, thus training only n models. 1 vs rest strategy is usually preferred in practice, and yields similar results for significantly less runtime. Based on this information from Scikit’s website, LinearSVC was used for classification.

Parameter Tuning: 500 randomly selected samples from the first 463,715 samples were used to construct the validation set with the remaining samples in training set. Parameter tuning was done using GridSearchCV for 3-fold cross-validation. The maximum number of allowed iterations was 1 million.

Parameter tuning was done on cost values between 2^-4 and 2^8, and primal hinge loss vs dual hinge loss. The optimal accuracy was with 2^-1 and dual hinge loss at .45 (details in Appendix Table 2).

Results: The best parameters obtained after parameter tuning on validation set were used to train and test LinearSVC. The detailed classification results on test set can be found in Appendix Table 1. Precision (tp / tp + fp), recall (tp / tp + fn), f1-score (harmonic mean of precision and recall) and support (# of occurrences of corresponding class) are included for each of the 10 classes i.e. 10 decades from 1920s to 2010s. The reported averages include micro avg (averaging the total tp, fn and fp), macro avg (averaging the unweighted mean per class) and weighted avg (averaging the support-weighted mean per class) where tp = true positives, fp = false positives and fn = false negatives

Accuracy (on entire test set) was 0.559 (i.e. ~56%) with a total running time (including data loading and parameter tuning) of ~8 minutes.

Adaboost

Using scikit's implementation of Adaboost with decision stumps as the model classifier, we applied the same analysis as detailed above for SVM. Testing on 65,345 data samples, there was almost no difference in accuracy for 10, 50, 100, 300, 500 and 1000 weak classifiers on the same data split. The maximum accuracy was 59%, just slightly above baseline.

Neural Network Analysis

Multi-layer Perceptron (MLP) feature within Scikit learn package has been used to build the model, train and test over the dataset. Given a set of features X_i and target y, MLP can learn non-linear function approximator for classification and regression. While MLP classifiers have the capability to learn non-linear models in real time, due to a non-convex loss function where there exists more than one local minimum, it is of crucial importance to perform parameter tuning in these type classifiers. Using the scripts we wrote, extensive parameter tuning analyzes performed on the number of hidden layers, neurons within each layer, learning rate and regularization term. Our measurement criteria of accuracy and performance efficiency are on number of epochs, training and validation errors.

We use stochastic gradient based optimizer developed by Kingma et.al. with the optimization tolerance of 1e-4. We find a better accuracy using logistic sigmoid while the results are not super sensitive to different minimization algorithms. Figure 1 shows the contour plot of parameter tuning analysis on the number of hidden layers and neurons.

This analysis includes 40X40 calculations and as this type analysis is computationally demanding, we did the parameter tuning on a toy set of 500 data. Our observation shows that the use of large number of hidden layers decrease the model accuracy and this is expected to occur due to the vanishing gradient phenomenon. We find that 2 hidden layers with 39 neurons within each layer gives an accuracy of 72-78% which is a fair range and use this settings for our model implementation.

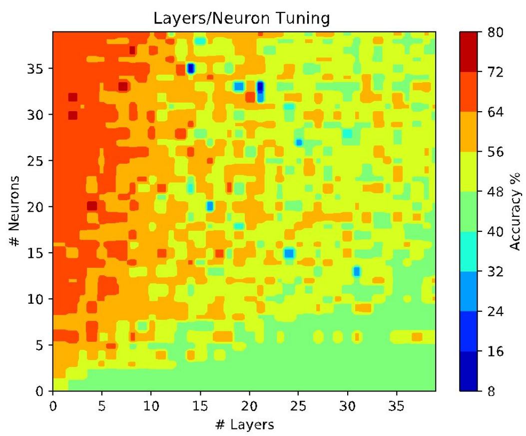

Figure 1: Contour plot of parameter tuning on the number of hidden layers and neurons within each layer

Next, we did some analysis on learning rate η and L2 regularization α parameters. Choosing the optimum learning rate is important in the minimization algorithm to avoid trapping in local metastable minimums. α term in gradient descent is added to weight update within backpropagation algorithm to ease the oscillation of weights due to the large learning rates. In fact, in downhill situation learning is accelerated by the factor of $1/(1-\alpha)$. Figure 2 and 3 shows the results after examining different learning rates in the range 0.01-0.5 and momentum term of 0-0.9. Higher η and α has shown the speed up the convergence, however the training error decreases in very large η values. We find that $\eta = 0,1 \; and \; \alpha = 0.5$ are the optimum parameters for our model with the 70-75% accuracy on the training data.

Now that we tuned the parameters of our MLP model, we implemented the cross validation algorithm to train on 75% of dataset and test on the remaining 25% unseen data. The final cross-validated accuracy of **<u>74%</u>** for prediction of the decade of each song within the dataset. This accuracy is unique on such a large dataset with more than 90 features associated with each song. More in-depth analysis can be also performed to better optimizing the model given more advanced and powerful supercomputing resources.

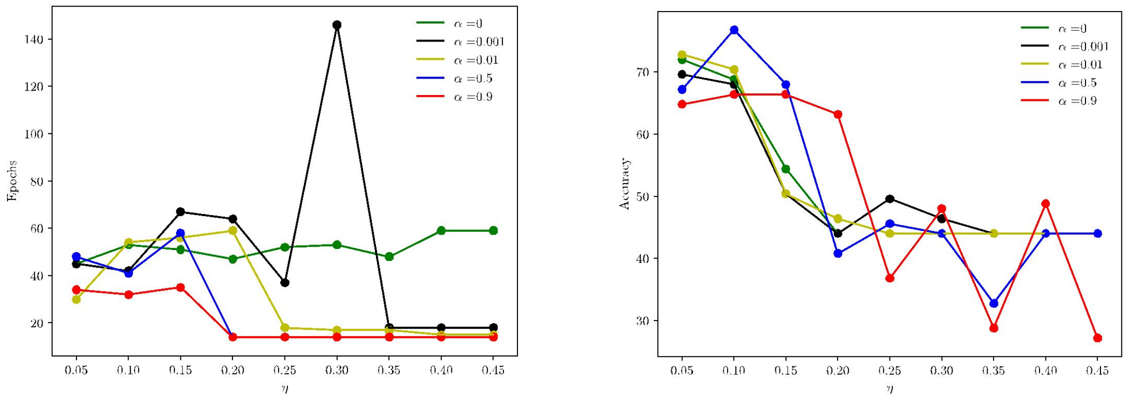

Figure 2: Parameter tuning on the learning rate and regularization parameter. Number of epochs till the convergence and the cross validation accuracy for each parameter are shown in left and right figures, respectively

Overall Conclusion

We achieved a est accuracy of about 74% for the optimized neural network, which is significantly larger than the baseline. For such a large dataset, this is a fairly significant improvement. SVM and Adaboost, however, did not perform well and were very close to the baseline of 58%.

References

- http://archive.ics.uci.edu/ml/datasets/YearPredictionMSD#
- https://scikit-learn.org/stable/modules/svm.html
- https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html#sklearn.model_selection.GridSearchCV
- https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html#sklearn.metrics.classification_report
- Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980*(2014).
- Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." *Journal of machine learning research* 12.Oct (2011): 2825-2830.

Appendix

	Precision	Recall	F1-score	Support
1920	0.02	0.65	0.05	20
1930	0.01	0.54	0.02	13
1940	0.02	0.49	0.05	61
1950	0.05	0.45	0.09	275
1960	0.16	0.35	0.22	1166
1970	0.26	0.35	0.30	2396
1980	0.36	0.44	0.40	4201
1990	0.51	0.11	0.18	12580
2000	0.73	0.81	0.77	29885
2010	0.00	0.00	0.00	1033
micro avg	0.56	0.56	0.56	51630
macro avg	0.21	0.42	0.21	51630
weighted avg	0.59	0.56	0.54	51630

Appendix Table 1: Detailed SVM classification results.

	-4	-3	-2	-1	0	1	2	3	4	5	6	7
loss=sq_hinge dual=False	.42	.42	.44	.45	.44	.44	.44	.44	.44	.44	.44	.44
loss=hinge dual=True	.38	.37	.41	.41	.42	.42	.42	.42	.42	.43	.43	.43

Appendix Table 2: Detailed SVM parameter optimization.