Practical 3: Classifying Sounds

Meriton Ibrahimi
meritonibrahimi@college.harvard.edu
Kaggle: @meriton

April 13, 2019

In this practical, our objective was to classify sounds into one of ten categories: air conditioner, car horn, children playing, dog bark, drilling, engine idling, gun shot, jackhammer, siren, and street music. Below, we provide an overview and discussion of our approach to feature engineering and modelling in addition to our results.

Feature Engineering

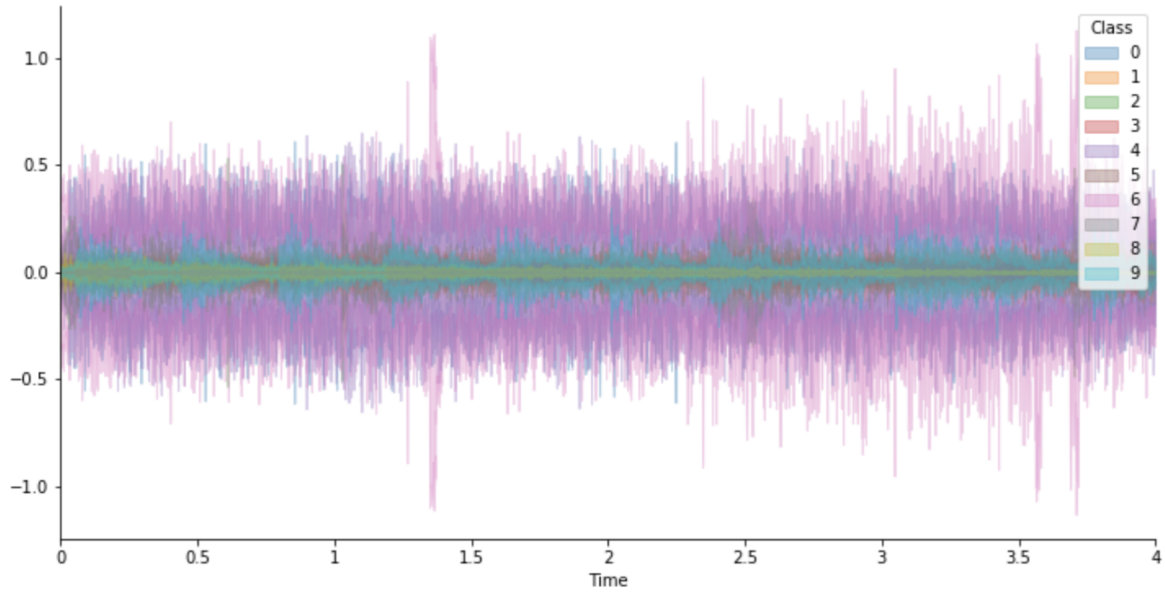
Building a model with the capability to classify audio requires sufficient power in the form of features in order to distinguish between classes; below we list and discuss the features we included in ours model in order to provide this power.

Zero crossing rate: the number of times that a signal crosses the horizontal axis
Chromagram: 12-element feature indicating how much energy of each pitch class, (C, C#, D, D#, ..., B), is present in the signal
Chroma Variant: 12-element chroma variant - Chroma Energy Normalized
MFCCs: 40-element Mel-frequency cepstral coefficients
Root-mean-square value: effective value of the total waveform
Spectral centroid: indicates the "center of mass" of the waveform
Spectral bandwidth, contrast, flatness, and rolloff: 1-,1-,7-,1-element characterists of the spectrum of the waveform
Polynomial fit: 4-element feature of the coefficients from an 3rd-order polynomial to the columns of the spectrogram
Max: 0.9 times the maximum amplitude
Min: 0.9 times the minimum amplitude
Count: the number of times the amplitude exceeds 0.85 times the range

The first several features are those defined and characterized in the Librosa API; these features are common for audio classification. The last three features are those we chose to add as we found them to be useful. In looking at the waveplots of the various classes, we see that certain classes are distinguishable by the the range of the maximum and minimum values; further, some classes are characterized by the number of times that they reach a peak.

Models

Waveplots of the 10 Classes



Logistic Regression

We started our model building by fitting a logistic regression (LR) model. A LR model is a simple linear predictor with a logit link function. We fit a LR model and optimized for regularization strength and penalty type. Doing so, we achieved our best cross validation results with an L_1 penalty and a regularization strength of 0.1. The accuracy achieved using this model on our left-out (stratified) validation set was 73.81%. Adjusting for the weights of the audio classes - as some classes are very infrequent - we fit a weighted LR model, optimizing in the same fashion, and achieved a validation score of 71.43%. Here, we saw that this discriminative model did not benefit from an adjustment for the class weights. Thus, we decided to move onto generative models.

Discriminant Analysis

For our generative model approach, we started with Linear Discriminant Analysis (LDA), which looks to classify an observation into the class which maximizes a discriminator function derived from Bayes Rule which is equivalent to the product of the likelihood of the observation and the prior probability of a particular class. We fit an LDA model and achieved 71.42% accuracy on the validation set. Seeing that this generative model does not outperform the discriminative model, we moved onto Quadratic Discriminant Analysis (QDA), postulating that the classes have different variance priors and that the classes have a non-linear, but quadratic separable boundary. With QDA, we achieved an accuracy of 80.03%. From here, we decided to move onto more non-linear models as the data appears to have significant non-linearity.

Random Forest

A Random Forest (RF) is an ensembled set of Decision Trees (DTs). RFs look to achieve models with simultaneously low bias and low variance, doing so by reducing the variance in low bias, high variance DTs. RFs reduce model variability in individual DTs in two manners. By bagging together

individual DTs from bootstrap aggregated data, RFs reduce variance by averaging. Further, RFs reduce model variability by adding another state of randomness in how each tree is built. Because DT models run via a greedy algorithm, even when individual DTs are built on boostrapped data, the DTs will likely still be very similar, splitting at each node in a similar manner. By randomly limiting and choosing from the pool of features at each splitting node, each tree in the RFs is (ideally) slightly different from any other tree. With many, different trees, RFs are able to provide low bias and low variance.

A RF has many hyperparameters that we need to optimize over e.g. how deep to build each tree, what percentage of features to choose from at each split, how many trees to bag together. Optimizing over several combinations of hyperparameters, we achieve a model a validation accuracy of 85.06%. Curious to push the model further, we refit a RF with the same optimized parameters but increased the number of trees to be run to 2000. With this latter model, we achieved a validation accuracy of 85.50%.

eXtreme Gradient Boosting
A RF model uses uses an ensemble of trees to build a low bias, low variance model by reducing the variance of individual low bias, high variance base models. In Boosting, we take the reverse approach. By using an ensemble of high bias, low variance individual tress built consecutively on top of the previous tree's error. In eXtreme Gradient Boosting, we follow the same approach but add a number of additioanl improvement over the traditional gradient boosting algorithms. Namely, XGBoost has built in tree pruning to tackle the greedy algorithmic nature of trees, is faster, and has built-in regularization. This means that XGBoost is incredibly flexible, but this also means that we have a number of hyperparameters to tune. We need to cross validate for the number of trees to use, L_1 and L_2 regularization strength, learning rate, and maximum depth of trees. In a plot below, we show code on how we cross validate over a grid of parameters.

Artificial Neural Network
Continuing the dive into non-linear models, we built an artificial neural network (ANN) in the hopes of capturing any non-linear patterns in our engineered features. We built a wide, deep neural network with the aim of approximating the best model through the universal approximation theorem. Many different architectures were trained and the model with the best validation accuracy had a score of 83.33%.

Convolution
After switching between between feature engineering and modeling, we figured we may have reached a limit to the number of quality features we could build without expert knowledge in the field; thus, we decided to let the data speak for itself, so we built Convolution Neural Networks (CNNs) to perform feature engineering and modeling all in one for us. We build CNNs with one- and two-dimensional convolutions by solely using the data and by transforming the data into a spectrogram, respectively. The one-dimensional CNN reached a 35.25% accuracy, and the two-dimensional CNN received a validation score of 84.55%.

Semi-supervised learning
In order to make the most of our available data, we looked to make use of a semi-supervised learn-

Parameter Grid Search Example for XGBoost

```
# XGBoost
n_class = 10; lrn_rates = [0.1,0.2,0.3,0.4]; lambdas = [0.01, 0.1, 1, 10]
max_depths = [3,4,5,6]; n_estimators = [100,500,1000]
xgb_param_grid = dict(learning_rate= lrn_rates, reg_lambda= lambdas,reg_alpha= lambdas,
                      max_depth= max_depths, n_estimators= n_estimators)
xgb_grid = GridSearchCV(estimator=XGBClassifier(objective= 'multi:softmax', num_class= n_class),
                        param_grid=xgb_param_grid, cv=3).fit(X_lbl_train, y_lbl_train)
```

Results

Model	**Test Accuracy %**
Baselines	
Logistic Regression Baseline	16.39
Random Forest Baseline	23.90
Model	**Validation Accuracy %**
Custom Features	
Logistic Regression	73.81
Logistic Regression W/ Weights	71.43
Linear Discriminant Analysis	71.43
Quadratic Discriminant Analysis	80.03
Random Forest	85.06
Random Forest (2,000 trees)	85.50
XGBoost	85.71
Artificial Neural Network	83.33
Convolution	
CNN 1-d	35.25
CNN 2-d	84.55

ing technique. We chose our best models, as determined by the validation score, re-trained the models on all the labelled data, predicted labels for the unlabeled training data, re-trained on all the training data, and predicted labels for the test data. Doing this, however, saw no improvement in our Kaggle submission score; in fact, most of the time, we saw our score decrease.

Conclusion

Our best performing models on Kaggle fell in line with our validation set. The Random Forest trained on our engineered features, cross validated for hyperparameters, and trained with 2000 trees came in second with a public score of 52.63% and a private score of 54.41%. Edging out the latter model was the cross validated XGBoost, which achieved a public score of 55.37% and a private score of 60.92%.

Using machine learning for audio classification is an art, one still being developed. Feature engineering is still being defined and finding enough labeled data to train on can be difficult.