Practical: New York Sound Classification

Benjamin Basseri basseri@cs50.harvard.edu
Kevin Huang khuang2@college.harvard.edu
Alberto Mosconi albertomosconi@college.harvard.edu

April 9, 2021

In this practical we implement sound wave classification with various models and discuss their workings.

1 Part A: Feature Engineering

Approach: Logistic Regression with PCA as basis function PCA. In PCA, we subtract $\overline{\mathbf{x}}$ from each $\mathbf{x}_n$, then stack the centered vectors horizontally to form design matrix X. The first 500 PC's are the the eigenvectors corresponding to the 500 largest eigenvalues of covariance matrix S, where $S = X^\top X/N$. These vectors represent the directions of largest variation. Using the PC's to form matrix U gives shape 500×44100 for the amp data and 500×11136 for the Mel. Data in original space is projected into parameter space by computing $\phi(\mathbf{x}) = U^\top \mathbf{x}$

Logistic Regression. The model encodes the labels as one-hot vectors and uses regularized loss $\frac{1}{2}\sum_{n=1}^{N}(\hat{y}_n - y_n)^2 + \frac{1}{C}||\mathbf{W}||$ where $\hat{y}_n = \text{softmax}(\mathbf{W}^\top \phi(x_n))$, and $\text{softmax}_k(z) = \frac{\exp(z_k)}{\sum_j \exp(z_j)}$. A softmax vector severs as a kind of probability distribution over the possible classes; using gradient descent we seek the $\mathbf{W}$ that minimizes softmax's L2 error plus penalties proportional to $||\mathbf{W}||$

We performed logistic regression on the amplitude and flattened Mel data using the sklearn library's `LogisticRegression` class and `decomposition.PCA` for a basis function. We performed a cross-validated grid search over parameters $C \in (0.1, 1, 10)$, max iterations in $(10^3, 10^4, 10^5)$, and balanced (errors weighted by class proportion) and non-balanced class weights.

Results. Both sets' best-in-class model used $C = 0.1$, non-balanced class weights, and 100,000 training cycles. The amp model achieved 19.4% overall accuracy with class accuracies from 0% to 65.2%. Mel scored 31.3% overall accuracy and class accuracies ranging from 14.7% to 53.3% (table 1, figure 1).

Discussion The sum of the first 500 eigenvalues of $\mathbf{S}$ divided by the trace is the amount of variance described by the PCs. For the amplitude dataset the PCs capture 60.2% yet for the flattened spectrograms they capture 97.9%, suggesting better performance potential with the latter.

Also we suspect the models lack robustness against time shifts. Suppose spectrograms $\mathbf{x}_i, \mathbf{x}_j$ are identical waves apart from a time shift. Spectrograms tend to be sparse (figs 2, 3), and after flattening their covariance can vary from 0 to $||\mathbf{x}_i||^2$ as $\mathbf{x}_j$ shifts to the right (see fig 4). If this phenomenon occurs often enough, it will be difficult for PCA to return meaningful modes of variation. This could be more severe in the amp data, since shifting a wave could produce perfectly negative correlation (figs 5, 6).

Performing PCA first grants large computational advantages. Without dimension reduction, logistic regression would have to fit 44100×10 weights for the amplitude dataset and 11136×10 weights for the flattened spectrograms, orders of magnitude more than when using dimension reduction. However, projecting the data into lower dimensions *could* affect its linear separability.

2 Part B: Model Classes

Approach: Random Forest Random Forests build an ensemble of decision trees using bagging (bootstrap aggregating). The algorithm gives a decision tree a bootstrapped data sample (resampled data with replacement). At each decision node it then randomly samples *features*. The learned decisions maximally separate the data at each decision node from the available features. The prediction function is then the majority vote of the learned trees. Sampling the features helps reduce correlation between trees. We used sklearn's RandomForestClassifier with n_estimators=10.
Results The amp model achieved overall accuracy 0.193 with per-class accuracies from 0 to 0.223. The Mel model achieved 0.437 overall with per-class from 0.333 - 0.568 (table 2).
Discussion Random Forest appears to be initially overfitting the training data, evidenced by the low test accuracy and extremely high training accuracy for the amp data, and a little less extreme for the spectrogram data. Compared to logistic regression models, the random forest results seem to be doing the same on the amplitude data and a little better on the spectrogram data. Because this was completed on the raw data, the logistic regression models seems to do equally worse on the amplitude data because the features of data within a class can be completely different because a sound occurs at two different times which throws off the weighting of Logistic Regression and the original features that the decision tree splits on to be non-generalizable. The spectrogram data seems to be slightly better than the logistic regression and that may simply be because the spectrogram data has more inputs for the random forest to train its decision trees on so the model is able to segment the data through its multiple decision trees a little more accurately based on the features with less variance than a single logistic regression function.

3 Part C: Hyperparameter Tuning and Validation

Approach: Random Forest
max_depth = 10, 50, or None because the default max_depth = None is currently overfitting, so restricting tree depth may prevent branches from overly divide and classify data.
max_features = 'auto', 'sqrt', and 'log2' because restricting the of features of each trees samples introduces less correlation between trees which produces a stronger final model.
min_samples_leaf = 10, 20, 50 because tree leaves have more samples which may prevent overfitting only 1 sample per leaf, increasing the generalizability of the model.
preprocessing PCA features = 500 or raw. By having the number of features be less than the amount of data that we have, we reduce the ease of overfitting.
Results. Test accuracies ranged from 0.204 to 0.450. Best-in-class model had max depth 50, max features auto ($\sqrt{p}$ for # of parameters p), max sample (resample proportion of dataset) 0.9, and minimum 10 samples for leaf node (see Table 3.1).
Discussion. We used sklearn GridSearchCV to test all the parameters outlined with cross validation sets = 3 and reported the parameters that produced the best accuracy on the test data above. The random forest appears to continue having extremely low test accuracy after optimizing the

parameters, never reaching above 50% test accuracy. One hypothesis is that the features which is time series data is not easily separable between classes because not all sounds are being produced at the same timestamp, which can throw off fitting the model based on each timestamp as a feature. The spectrogram data was able to do a little better, but with the lack of spatial recognition, random forests did not prove to really improve that much.

Approach: CNN

penultimate layer size = 5, 10, 25, 50, 100. After convolutional layers, the CNN flattens and passes through dense layers. The output layer must be shape (10,) to match the labels. However, the penulatimate layer's width affects the number of decision boundaries the model can draw on features extracted from earlier layers. We did a hyperparameter search primarily to see the effects of this layer's width on model performance.

learning rate = 0.01, 0.001. The rate at which we take gradient descent steps after each batch.

Results. With both learning rates the accuracy seems to peak when the penultimate width is somewhere between 25-50.

Discussion. When the penultimate width is too small it may be the model isn't able to fully use features it learned in the convolutional layers. But if the width is to large it becomes more prone to overfitting (see Table 3.2). So there seems to be a sweet spot in the middle where the model has enough power to capture the patterns in the data but not so much that it overfits. Also, the learning rate 0.001 performed much better than 0.01, suggesting that larger step sizes may prevent the optimizer from finding minima.

4 Bonus, Part D: Explore some more!

4.1 CNN Implementation.

Approach. We trained a CNN on the Mel dataset using Pytorch: two convolutional layers with max pooling, then two dense layers, and a final output layer.

Results The model overall achieved 0.51 accuracy on the test data with per-class accuracy from 0.23 - 0.76 (table 4). See figure 7 for the plot of train and test accuracy over epochs. Notice how after around 125 epochs the model starts drastically overfitting.

Discussion CNN's seemed a logical choice for the 2d Mel dataset. CNN's exploit an image's 2d structure by sliding kernels over it to create activation maps. Here, we hoped a CNN would learn meaningful features in the spectrograms, more agnostic to time shifts than previous models. The CNN architecture easily outperformed logistic regression and random forests but is also subject to overfitting if trained too long (Fig 7).

We also inspected the car horns, the model's worst class at 0.23 accuracy. It predicted street music (class 9) most often, followed by car horn then siren (class 8) (see Fig 8). We suspect this is because car horns have similar frequency ranges to sirens and street music plus car horn samples are very underrepresented in the dataset: 3.5% vs. over 12% for class 8, 9. While gunshots are also underrepresented, their frequency patterns must be very different from the other classes so the network easily learned it.

4.2 Data Augmentation on Random Forests

Approach. Because our data size was not large, we tried to increase the data size through data augmentation, namely time shifting and adding noise for the amplitude data to increase the data

to 3x the current data size.
Results Adding noise to raw data at each feature with originally optimal parameters from Part C, noise_factor was taken from a Gaussian distribution * 0.5.
Accuracy score on train data: 0.594, accuracy score on test data: 0.217
Adding both noise from above and timeshift (max 50% time shift)
Accuracy score on train data: 0.487, accuracy score on test data: 0.232
Discussion We first tried to see if adding noise would help the model recognize more important features that define a class. However, with a 1% increase in accuracy, we tried to address the issue that some classes don't have the same features than others, same classes only have the same trends.
Therefore, we tried to deal with this issue with time-shifting so help produce more data that increases the chance of classes having similar features (similar amplitudes at timestamps). This led to an increase in 3% which is marginal, but better than from adding data. Overall, this leads us to conclude that the random forest isn't effective for modeling and fitting because despite the increases in test accuracy, the train accuracy is also low, so the model's accuracy is just bad. This may be caused by the lack of similarity between individual features that the random forest tries to define as its decision nodes, and the failure of random forests to recognize relationships between features that is characteristic to the noise classes we have.

5 Appendix: Figures and Tables.

Model	Overall Accuracy	Class Accuracies									
		0	1	2	3	4	5	6	7	8	9
LR Amp train	0.282	0.286	0.117	0.542	0.149	0.160	0.287	0.386	0.252	0.287	0.278
LR Amp test	0.194	0.23	0	0.652	0.057	0.049	0.140	0	0.123	0.110	0.150
LR Mel train	0.506	0.443	0.721	0.559	0.442	0.542	0.483	0.602	0.584	0.546	0.374
LR Mel test	0.313	0.210	0.538	0.532	0.188	0.330	0.348	0.533	0.203	0.487	0.147

Table 1. Overall and per-class accuracy of the Logistic Regression with PCA500 on the amplitude and spectrogram (mel) datasets

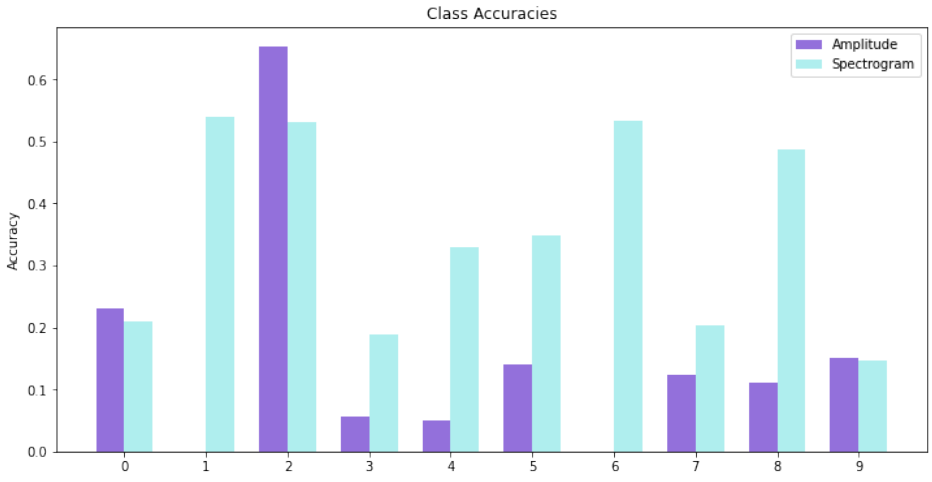

Figure 1. Class accuracies for logistic regression with PCA, amplitude and Mel data.

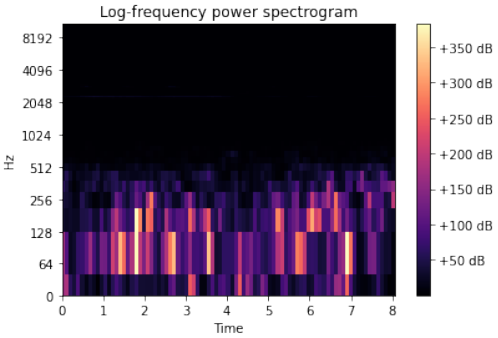

Figure 2: spectrogram as 2d array

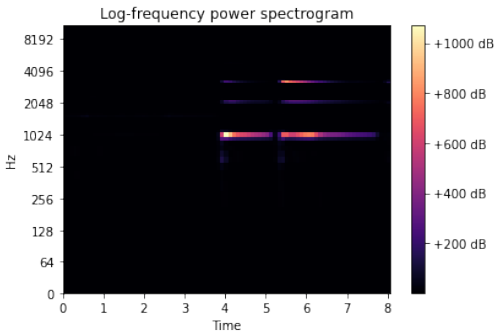

Figure 3: Spectrogram as 2d array

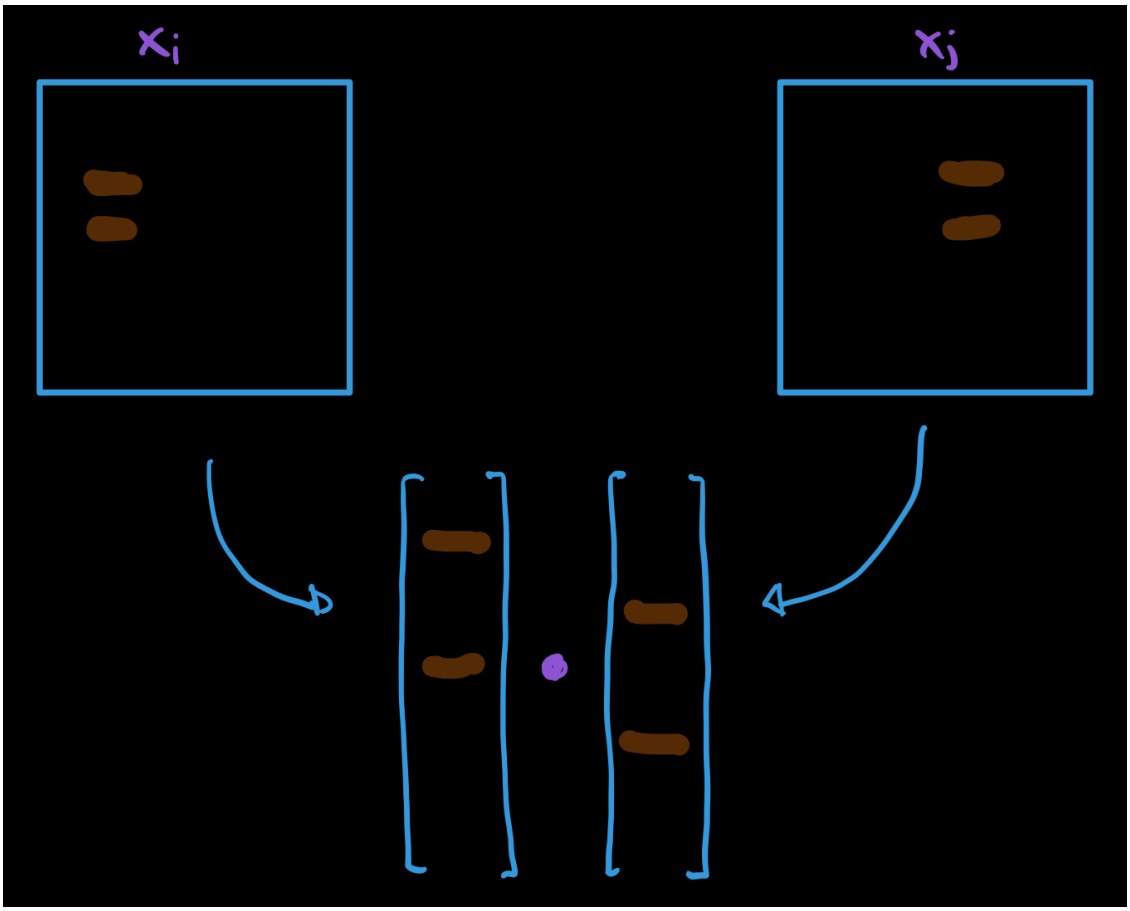

Figure 4. Two spectrograms with similar profiles except for a time shift can flatten to vectors with almost no inner product/covariance.

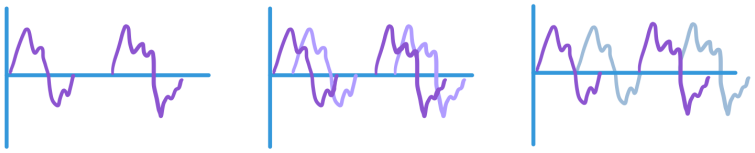

Fig 5. Sounds of the same class with similar waveforms will have drastically different covariances depending on when in the time interval they occur. In this case, as the wave shifts to the right the covariance between the two waves decreases, even though the waveforms are otherwise identical.

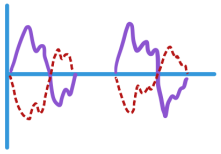

Figure 6: Phase inversion → perfect negative correlation

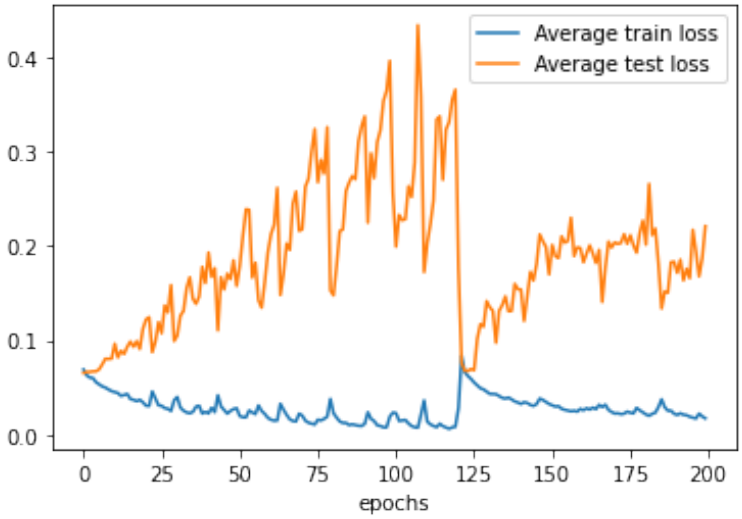

Fig 7. Plot of train and test losses over epochs for CNN model.

Raw Data vs Train, Test, and Test Class Accuracies

Data	Train	Test	0	1	2	3	4	5	6	7	8	9
Amp	0.970	0.193	0.223	0.051	0.291	0.205	0.140	0.197	0.000	0.097	0.140	0.080
Mel	0.986	0.437	0.370	0.205	0.492	0.410	0.568	0.432	0.333	0.390	0.436	0.363

Table 2: Random Forest accuracy results

Random Forest Best Parameters After Optimization and Accuracies

Data	max_depth	max_features	max_samples	min_samples_leaf	train	test
Amp Raw	10	auto	0.9	20	0.559	0.204
Amp PCA 500	10	auto	0.9	20	0.428	0.239
Mel Raw	50	auto	0.9	10	0.873	0.450
Mel PCA 500	10	auto	0.9	10	0.729	0.330

Table 3.1. Random Forest Grid Search Results

CNN Grid search results

Penultimate width	Learning Rate	Test Accuracy
5	0.01	0.41
10	0.01	0.39
25	0.01	0.43
50	0.01	0.46
100	0.01	0.40
5	0.001	0.43
10	0.001	0.53
25	0.001	0.57
50	0.001	0.50
100	0.001	0.54

Table 3.2. CNN Grid Search results

CNN Test Data Class Accuracies

Data	Overall	0	1	2	3	4	5	6	7	8	9
Train	0.93	0.92	0.90	0.91	0.94	0.96	0.95	0.88	0.100	0.94	0.91
Test	0.56	0.22	0.20	0.52	0.55	0.48	0.58	1.00	0.56	0.62	0.61

Table 4: CNN accuracy results: 120 training epochs, learning rate 0.001, 2 convolutional layers, 2 dense layers plus dense output layer

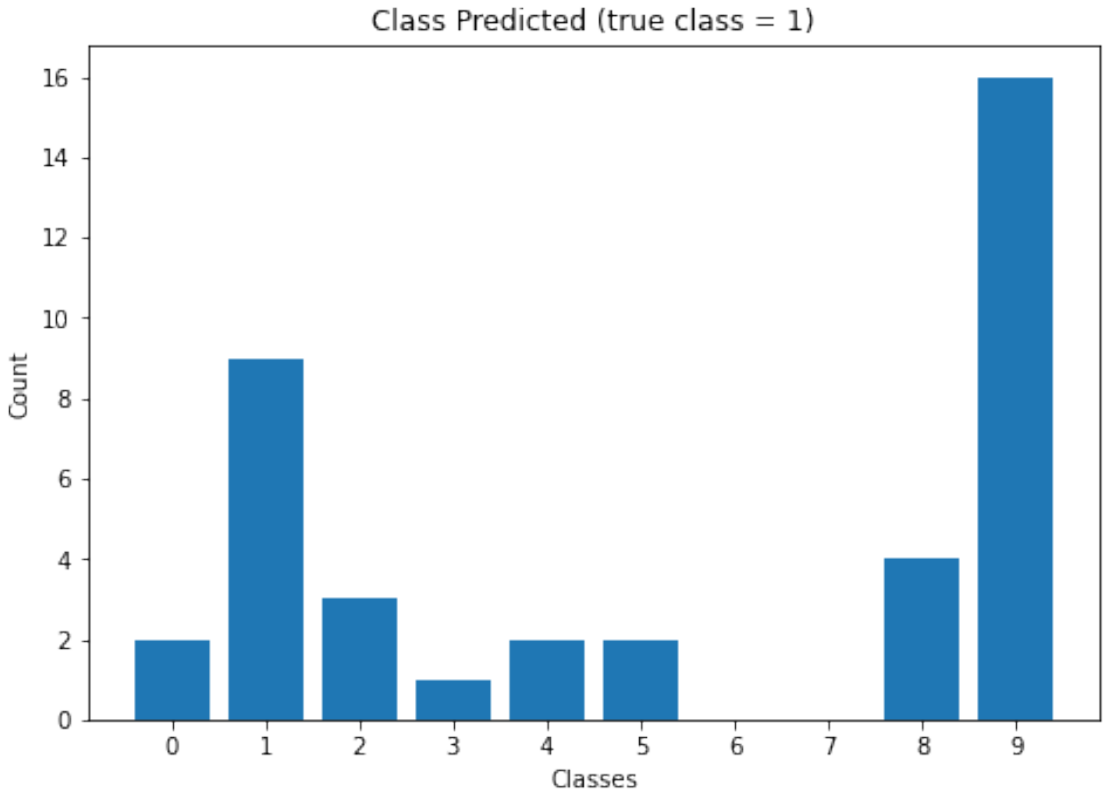

Figure 8. Spread of CNN's predicted class on class 1 (car horn) in the test set