CS 181 Practical

Moses Mayer, Haneul Shin, Max Guo
mosesmayer, haneulshin, mguo @ college.harvard.edu

March 26, 2021

Part A: Feature Engineering

Approach. We performed PCA using the PCA function from `sklearn` to select the top 500 principal components for the Raw Amp and Mel data sets. These components represent the eigenvectors corresponding to the 500 largest eigenvalues, and are the 500 vectors that capture the most variance in the data. Specifically, this is done by finding the eigenvalues of the matrix $\frac{1}{n}X^TX$, where n is the number of observations and X is the matrix of observations.

We then performed Logistic Regression using `LogisticRegression` from `sklearn`. The model we trained predicts output probabilities for each data point by taking a linear combination of the 500 principal components (by applying PCA to a raw data point) using the learned weights, which yields a vector of length 10, and then applies the softmax function to transform this vector into a probability vector of length 10, where the i^{th} component represents the probability the data point is in class i per the model. The model then uses these probabilities to calculate the loss. By differentiating this loss function and performing gradient descent, it optimizes our weight parameters.

Results. The detailed overall and per-class testing classification accuracies are shown in **Table 1** in the Appendix, but we give a few summary values below.

Accuracies	Training	Testing
Raw Amplitude Data	40.91%	19.16%
Mel Spectrogram Data	70.27%	31.04%

Discussion. The Amp and Mel overall testing accuracies were 19.16% and 31.04% respectively. Although both accuracies were relatively low, the Mel accuracy was higher than that of Amp. We hypothesize that this is the case because Amp only measures the raw amplitude, whereas Mel measures the spectrum of frequencies, which may provide more detailed information to allow the model to distinguish between different features of sounds. Furthermore, as seen from **Figures 1** and **2**, PCA does far better at capturing the variance within the Mel Spectrogram data as opposed to the Raw Amplitude data, as 500 components explain nearly all of the variance in the Mel Spectogram data, whereas 500 components only explains $\approx 60\%$ of the Raw Amplitude data.

Specifically, in **Table 1**, we see that Amp performed poorly on all the classes, which is likely because the sounds in each class have varying amplitudes, so it is difficult to distinguish them based on that feature. Mel had relatively high accuracies on classes such as car horn (58.97%), children playing (58.97%), gun shot (56.67%), and siren (47.03%). This is likely because each of these sounds are associated with a particular set of frequencies. On the other hand, Mel performed poorly on air conditioner (19.00%), dog barking (15.72%), and street music (16.00%). This may be because these sounds have more variance in their frequencies, e.g. there are many different types of street music.

It may be better to perform PCA before the logistic regression because the number of features in any data point in the amplitude data is extremely large (44100) and may result in issues of both computational run-time and, perhaps more importantly, a higher variance within the weights due to the larger number of weights involved. Without regularization, this can easily lead to overfitting and an inability to generalize the model. PCA reduces the dimensionality of the data significantly while capturing most of the variance within the data, which leads to more feasible run-times and fewer weights, reducing the variance within the model. This can reduce overfitting. However, this may lead to increased overall bias due to the data loss from the PCA compression.

Part B: Model Classes

Approach. We implemented a KNN model with $K = 10$ using KNeighborsClassifier from sklearn. The algorithm is specified in detail in the pseudocode in **Algorithm 1** in the Appendix. We then fit this model on both the Amp and Mel datasets. We calculated the per-class testing classifications via `sklearn`'s `classification_report`, which gives the recall ($\frac{TP}{TP+FN}$) for each class.

Results. The detailed overall and per-class testing classification accuracies are shown in **Table 2** in the Appendix. Below, we compare these with those from Logistic Regression from Part A.

Accuracy	PCA + Logistic Reg.		PCA + KNN	
	Training	Testing	Training	Testing
Raw Amplitude Data	40.91 %	19.16 %	30.29 %	16.61%
Mel Spectrogram Data	70.27 %	31.04 %	58.06 %	25.72 %

Discussion. First, both models (KNN and Logistic Regression) do better on the training than testing dataset for both Amp and Mel. This may suggest that both models are overfitting. However, since we used $K = 10$ (a somewhat larger value of K) for KNN to ensure this is not likely, this suggests the compression method for PCA is very data-dependent. In other words, fitting a PCA model on different data (e.g. the testing datasets) will likely result in different principal components.

As in Logistic Regression, the Mel data yields a better accuracy than the Raw data, which is likely due to the same reason of Mel providing more distinguishing features of the sound data than Mel (see the Part A Discussion section for details). Since we used PCA before applying KNN, the lower variance explained in the Amp data case affects the accuracy of the resulting KNN model as well.

We observe from **Table 2** that for Raw Amp, Classes 1 (car horn) and 7 (jackhammer) are not being predicted by KNN at all. This implies that the classes of the nearest neighbors of PCA-compressed Raw Amp data may not be a good indicator of the class of a certain data point, especially since Class 7 represents 11.85% of the data, which is a large proportion. We thus hypothesize that the Class 7 data points are quite spread out, and their nearest neighbors within the principal component-defined space are all from different classes.

This hypothesis is supported by comparison with Class 8. By **Tables 1** and **2** for Mel accuracy for Classes 7 and 8, note that KNN performs well for 7 (66.53% accuracy) and not so well for 8 (26.27% accuracy), while Logistic Regression does worse on 7 (26.27%) than on 8 (47.03%). This suggests that the data may be distributed or spread out differently for the different classes. For example, if for all data points in a certain class there are small clusters spread throughout different areas, then KNN may perform well whereas Logistic Regression would perform poorly. However, if the same points lied on the same hyperplane that the Logistic Regression could transform into the same class, then KNN might do poorly if each point isn't surrounded by neighbors of the same class. In other words, the data isn't very well organized to suit either KNN or Logistic Regression.

Part C: Hyperparameter Tuning and Validation

Approach. We used KNN and Random Forests using RandomForestClassifer and KNeighborsClassifier from sklearn for our two models. We used `n_estimators = 100` trees for the latter.

We explained how KNN works in Section 2.1 and in **Algorithm 1** of the Appendix. We now explain how the random forest classifier works. Recall that a decision tree is a model such that for each data point, the output is determined by a sequence of comparisons based on the features against threshold values, which form a tree structure. A random forest is an ensemble of independent decision trees that is an extension of bootstrap aggregation. It first trains each tree on a separate bootstrapped sample of observations from the training set, and for each tree node, randomly selects some subset of features and picking the one that acts as the optimal predictor at that split. To see the technical algorithm, refer to **Algorithm 2** of the Appendix.

We performed our hyperparameter search using sklearn's GridSearch, using 5-fold cross validation to verify the best selection of parameters. For each choice of parameter, GridSearchCV trains 5 versions of that model and takes the average of their performance on their respective untouched portions of the training dataset (standard cross-validation) to find the model's score with that parameter. We choose the hyperparameter that yields the highest score for our final model.

For KNN, we performed a hyperparameter search over the value of $K \in \{1, 5, 10, 25, 35, 50\}$, i.e. the number of nearest neighbors. For Random Forests, we performed a hyperparameter search over the max-tree depth max_depth $\in \{3, 6, 9, 12, 18, 24\}$.

Results. For KNN, the accuracy for each hyperparameter value is shown in **Table 3** in the Appendix. The value $K = 1$ had the highest accuracy for both Raw Amp and Mel, with values of 21.65% and 37.42% respectively. Generally, as K increased, the accuracy decreased for both datasets. The overall and per-class accuracies for 1NN for both datasets are shown in **Table 5**.

For the random forest, the accuracy for each hyperparameter value is shown in **Table 4** in the Appendix. The value max_depth = 24 had the highest accuracy for Raw Amp with accuracy 22.49%, whereas max_depth = 18 had the highest accuracy of 41.13% for Mel. As the max depth increased, barring a few exceptions, the general trend was that accuracy increased for both datasets. The overall and per-class accuracies for max_depth = 24 (Amp) and 18 (Mel) are in **Table 6**.

Discussion. For KNN, by **Table 3**, the best choice for K was $K = 1$. However, the $1NN$ model does not perform vey well (accuracies of only 21.65% and 37.42% for Amp and Mel); it is simply the best among all the choices. This supports the hypothesis from Part B that the PCA-compressed data does not have well-formed single clusters of data, but instead that each class has data points scattered in the domain, with perhaps smaller clusters forming that allow for the $1NN$ model to make better predictions than the others since it only uses the closest data point as a predictor.

For random forests, by **Table 4**, deep trees had better accuracies for both datasets. Deep trees allow the models to fit the data better, but with the tradeoff that there may be a higher variance resulting from the model overfitting. Since deep trees fit the data better, this suggests that, from a generative perspective, the data is generated by a complex model which requires more degrees of freedom from a decision tree model to fit well. However, since the difference between the different model accuracies is only a few percentage points, this statement is only true to a slight extent.

An important note about our search is that we performed model selection via cross validation to obtain better metrics for our model performance. Thus, the accuracies in **Table 3** and **Table 4** represent the average on the performance of the remaining portion of the *training* dataset, and not the entirety of the training set. The cross validation ensures that our scores in determining the hyperparameters are similar to the test scores. Indeed, for example, our overall testing accuracy of 40.05% for a random forest max depth of 18 is similar to the score during cross validation, which is 41.13% (and this phenomena can be generally seen in the other models and datasets).

Bonus, Part D: Explore some more!

Approach. We used `PyTorch` Neural networks and focused our application to the PCA Compressed version of the Amp data (due to computation time and variance reasons; see Part A). We stayed within the framework of a feedforward Neural Network, and we varied the number of layers and number of nodes as our hyperparameters to minimize the loss. After tuning these values, we ended up testing two models: one with 4 hidden layers and one with 3. This involved setting up the `PyTorch` dataloaders as needed and creating the models following the structure of an `nn.Module`.

A feedforward model with one hidden layer (for illustration purposes) can be represented as follows:

$$\mathbf{z} = \text{softmax}(\mathbf{W}_2\text{ReLU}(\mathbf{W}_1(\mathbf{x}) + \mathbf{b}_1) + \mathbf{b}_2)$$

where $\mathbf{W}_i$'s are weights, and $\mathbf{b}_i$'s are bias vectors. The initial $\mathbf{W}_1$ transforms the input data into the right dimensions for the hidden layer $\mathbf{W}_2$, which outputs a vector of size 10, which is transformed by the softmax layer into probabilities. We hypothesized that the nonlinearity of ReLU would allow this model to capture more complex patterns than that of Logistic Regression on PCA. Thus, we expected the performance of models like this with additional layers to surpass that of Part A.

Results. The results showed that with 300 hidden layer nodes, the 4-layer network had an overall testing accuracy of 22.67% whereas the 3-layer network had an overall testing accuracy of 19.34%. With the 200 hidden layer nodes, the 4-layer network had 22.03% testing accuracy whereas the 3-layer network had 23.39% testing accuracy. The full data is available in **Table 7** and **Table 8**.

Discussion. We first picked the number of hidden layers since this determines the overall structure of the network. By tuning these values, we found that the accuracy tended to decrease as the number of layers increased, which may result from the increased complexity since it can cause overfitting to the training data. Thus, we first decided to focus on 3 and 4-layer NNs. Next, we decided to tune the number of nodes since this is the next key restriction on the structure of the network. Increasing the number of nodes initially increased the accuracy, but then decreased the accuracy for more than 300 nodes. Per **Table 8**, while the 4-layer models have similar accuracies for 200 and 300 nodes, the 3-layer model does better (23.39% vs. 19.34%) with 200 nodes. This again suggests that overfitting occurs when the number of nodes increases: a large number of nodes can fit to any individual data set better, but may generalize less well.

From the overall testing accuracy, our hypothesis that neural networks has the capacity to perform better than Logistic Regression was confirmed. In particular, the Logistic Regression had a testing accuracy of 19.16%, whereas our best Neural Network performance (with 200 Nodes and 3 layers) was 23.39%, which is significantly higher. Future work may include more nuanced tuning of the nodes, as well as other hyperparameters such as batch size, learning rate, and even going beyond simple feed-forward classification networks.

Appendix

Table 1: Testing Accuracies using PCA + Logistic Regression

Model	Raw Amp. (%)	Mel Spec. (%)
Overall Acc	19.16	31.04
Class 0 Acc	22.67	19.00
Class 1 Acc	0.00	58.97
Class 2 Acc	58.19	51.51
Class 3 Acc	6.55	15.72
Class 4 Acc	9.09	32.95
Class 5 Acc	14.02	32.95
Class 6 Acc	6.67	56.67
Class 7 Acc	11.02	26.27
Class 8 Acc	11.86	47.03
Class 9 Acc	15.67	16.00

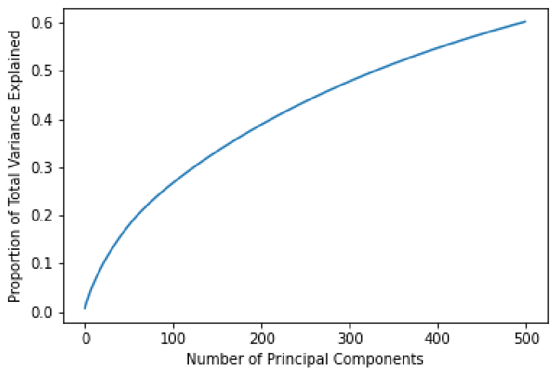

Figure 1: PCA Variance Explained for Raw Amplitudes

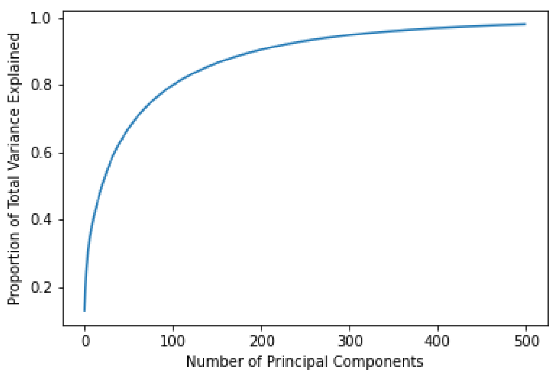

Figure 2: PCA Variance Explained for Mel Spectrogram

Algorithm 1 Implementation of K-Nearest-Neighbors

```
1:  procedure KNN($X$, $d$)
2:      Predictions = []
3:      for data point $x \in X$ do
4:          Find $k$ closest data points to $x$ within $X \setminus \{x\}$ with respect to
5:          the Minkowski metric with $p = 2$, or the L2 norm (the default for sklearn):
```

$$D(x, y) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$$

```
6:          add mode of the classifications of nearest neighbors to Predictions
7:      end for
8:      return Predictions
9:  end procedure
10:
11: Perform PCA on training data set $X$ with 500 principal components to obtain $X_{PCA}$
12: KNN ($X_{PCA}$, 500)
```

Algorithm 2 Implementation of Random Forest (Source: The Elements of Statistical Learning)

```
    procedure RANDOM-FOREST($X$, max-depth)
 2:     for b do = 1 to $B$ (number of bootstrap samples)
            Draw a bootstramp sample of size $N$ from the training data.
 4:         Grow a random forest tree to the bootstrapped data by repeating the steps
            for each terminal node in the tree, until maximum depth is reached:
 6:             Select $m$ variables at random from the $p$ variables (features).
                Pick the best variable/split-point among the $m$, based on entropy.
 8:             Split the node into two daughter nodes.
            end for
10:     end for
        Output the ensemble of trees $\{T_b\}_{b=1}^{B}$.
12: end procedure

14: Perform PCA on training data set $X$ with 500 principal components to obtain $X_{PCA}$
    random-forest($X_{PCA}$, max-depth)
```

Table 2: Testing Accuracies using KNN

Model	Raw Amp. (%)	Mel Spec. (%)
Overall Acc	16.61	25.72
Class 0 Acc	38.33	10.33
Class 1 Acc	0.00	33.33
Class 2 Acc	10.03	27.76
Class 3 Acc	16.16	12.23
Class 4 Acc	4.92	29.92
Class 5 Acc	12.12	33.33
Class 6 Acc	6.67	33.33
Class 7 Acc	0.00	66.53
Class 8 Acc	54.24	26.27
Class 9 Acc	2.67	4.67

Table 3: KNN Accuracies by K value

Model	Raw Amp. (%)	Mel Spec. (%)
1	21.65	37.42
5	19.93	34.02
10	19.36	32.99
25	17.90	28.27
35	17.16	26.92
50	16.19	25.55

Table 4: Random Forest Accuracies by max_depth value

Model	Raw Amp. (%)	Mel Spec. (%)
3	19.25	29.01
6	20.82	33.89
9	21.56	37.96
12	21.95	39.78
18	21.73	41.13
24	22.49	39.96

Table 5: Testing Accuracies using 1NN

Model	Raw Amp. (%)	Mel Spec. (%)
Overall Acc	19.80	28.58
Class 0 Acc	28.33	16.33
Class 1 Acc	0.00	43.59
Class 2 Acc	14.05	25.42
Class 3 Acc	25.33	24.02
Class 4 Acc	7.20	31.82
Class 5 Acc	31.82	31.44
Class 6 Acc	13.33	36.67
Class 7 Acc	2.12	58.05
Class 8 Acc	42.80	33.47
Class 9 Acc	12.33	12.33

Table 6: Testing Accuracies using max_depth = 24 (Amp), 18 (Mel)

Model	Raw Amp. (%)	Mel Spec. (%)
Overall Acc	25.35	40.05
Class 0 Acc	30.00	19.67
Class 1 Acc	0.00	17.95
Class 2 Acc	61.54	55.52
Class 3 Acc	6.11	24.02
Class 4 Acc	8.33	31.06
Class 5 Acc	40.91	39.39
Class 6 Acc	3.33	6.67
Class 7 Acc	19.49	55.08
Class 8 Acc	19.92	64.41
Class 9 Acc	15.00	41.00

Table 7: Testing Accuracies using Neural Networks, 300 Nodes per Layer

Model	4-layer NN (%)	3-layer NN (%)
Overall Acc	22.67	19.34
Class 0 Acc	26.00	42.67
Class 1 Acc	0.00	5.13
Class 2 Acc	68.56	0.00
Class 3 Acc	18.34	30.57
Class 4 Acc	10.61	20.83
Class 5 Acc	20.46	26.52
Class 6 Acc	0.00	0.00
Class 7 Acc	11.44	6.36
Class 8 Acc	8.48	14.41
Class 9 Acc	14.67	17.00

Table 8: Testing Accuracies using Neural Networks, 200 Nodes per Layer

Model	4-layer NN (%)	3-layer NN (%)
Overall Acc	22.03	23.39
Class 0 Acc	44.33	39.00
Class 1 Acc	0.00	0.00
Class 2 Acc	41.14	57.19
Class 3 Acc	18.35	15.72
Class 4 Acc	9.85	13.26
Class 5 Acc	26.89	25.76
Class 6 Acc	0.00	0.00
Class 7 Acc	0.00	6.36
Class 8 Acc	7.63	9.75
Class 9 Acc	23.67	16.33