ORIE 4741 Project Final Report
Identifying Gender Based on Voice Samples

Siva Sankalp Patel (sp2337), Premdeep Sharma (ps882), Michael Hwang (mh634)
Cornell University

1. Introduction

Being able to recognize a voice is a very human thing to do. Although the exact methods are not clear, we do know for a fact that people are easily able to distinguish one person's voice from another without much effort. A task like this is much more difficult for a computer, as a computer requires prior knowledge in order to specific whose voice is whose.

Instead, we choose to give the computer an easier task – we want to see if a computer can determine the gender of a voice. By constructing models and running algorithms on a sample space, we attempt to find a method that best distinguishes between male and female.

This kind of program has the potential to be useful in voice recognition systems. Voice and speech recognition has been a topic that has been studied since the early 1900's, and has many modern-day applications. Casual, every-day uses like Siri and voce automation in homes uses these ideas. Things such as automatic translation and security use voice recognition as well. Along with all these, further research into voice recognition has the potential to advance knowledge in the field of human speech perception, a field of linguistics that has baffled many scientists.

2. Dataset Description

We used the online resource Kaggle to attain data for this project. The data set we used included 3,168 frequency related parameters from a voice sample synthesizer. These data points came from four separate resources: The Harvard-Haskins Database of Regularly-Timed Speech, Telecommunications & Signal Processing Laboratory (TSP) Speech Database at McGill University, VoxForge Speech Corpus, and Festvox CMU ARCTIC Speech Database at Carnegie Mellon University.

The data set included the following feature columns:

• meanfreq: mean frequency (in kHz) – average frequency for a particular sample
• sd: standard deviation of frequency – how much the frequency deviated from the mean throughout the experiment
• median: median frequency (in kHz) – frequency at the middle of the spectrum
• Q25: first quantile (in kHz) – frequency in the quarter of the complete frequency spectrum
• Q75: third quantile (in kHz) – frequency in the three quarter of the complete frequency spectrum

• IQR: interquantile range (in kHz) – mid-spread shows us the spread in the middle region, calculated using the difference of the upper and lower quartiles
• skew: skewness – asymmetry in the probability distribution of the vocal sample frequencies
• kurt: kurtosis – to measure the thickness of the tail end of the probability distribution of frequencies
• sp.ent: spectral entropy – measurement of difference in the distribution of spectral energy
• sfm: spectral flatness - to distinguish between a sample being a noise like (flat) compared to tone like (pointy)
• mode: mode frequency – most common frequency
• centroid: frequency centroid – multivariate mean for the multidimensional data
• meanfun: average of lowest frequency of periodic waveform measured across acoustic signal
• minfun: minimum fundamental frequency measured across acoustic signal
• maxfun: maximum fundamental frequency measured across acoustic signal
• meandom: average of most commonly occurring periodic frequency measured across acoustic signal
• mindom: minimum of dominant frequency measured across acoustic signal
• maxdom: maximum of dominant frequency measured across acoustic signal
• dfrange: range of dominant frequency measured across acoustic signal
• modindx: modulation index, calculated as the accumulated absolute difference between adjacent measurements of fundamental frequencies divided by the frequency range
• label: male or female

In this data set, there was no missing data – all 3,168 samples were accounted for. However, many points of data we had needed to be checked for corruption. The first most obvious issue would be significant outliers. The range for frequency (in the data set, meanfreq) tends to be between 0.10 and 0.30. We searched for negative frequency values (which should not exist) and frequencies over 0.30, but found none. Similar tests were done on the mean fundamental frequency (meanfun) and mean dominant frequency (meandom). The meanfun data was tested from 0 to 0.30 because of its similar distribution to the meanfreq data, whereas the meandom was tested from 0 to 3.0. No significant outliers were found.

We noticed that there was a large piece of data with value 0 for mode, but having 0 mode is a pretty likely when it comes to voice frequency since some people tend maybe take large pauses while talking, and thus make their mode for voice frequency 0. On a similar note, no blank cells were found.

There was a large variation in values of skew. The average value around 3.14 was and the median was 2.20, but many values that ranged from 10 to 30 also were in the data. And while those do seem like cases where an extreme outlier exists, we decided that this could be attributed to different voice ranges between people. We found it strange that there were no negative values for skew, but since we cannot remove data based on this observation we decided not to change the data based on this.

3. Problem Description

Our project's objective is to create a model that identifies the gender of a voice based on known voice samples. The input space for this project is some parameter from the data that describes a particular voice sample. The output space was a gender label for each sample's frequency parameter. For the sake of simplicity, only male and female were included as possible genders. We want to figure out which properties of voice are best used to determine a voice's gender, and what algorithm gives the highest confidence in its prediction.

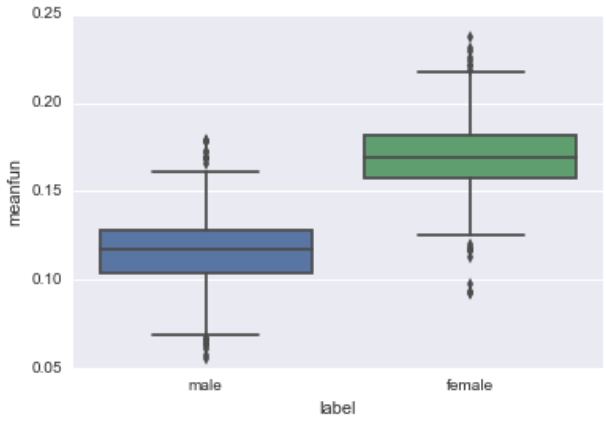
4. Exploratory Data Analysis

We first grouped the data into male and female and looked at the statistics of the features in each class. For each of the features in the data, we found a median for each gender for that feature and calculated the average of these two medians. This average is considered as the segregation point between each of the genders in the test set. For this feature, all the points in test set are compared with the segregation value and assigned appropriate label. For example, if the meanfun median for male had a value of 0.5 and meanfun median for female had a value of 0.7 then 0.6 would the segregation point. Values above 0.6 would correspond the female and values below 0.6 would correspond to male on the test data set. The success rates for all twenty features are included in the table below.

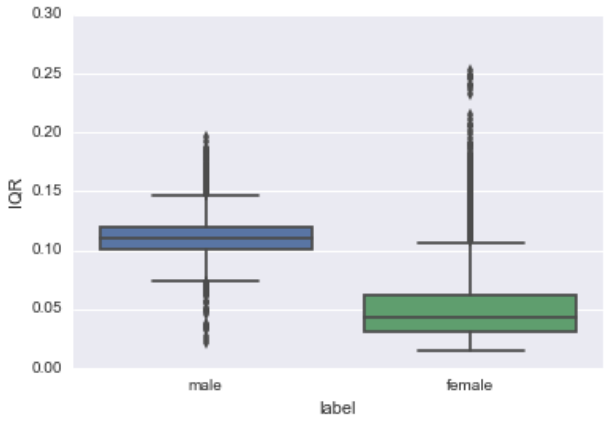
Feature	Success Rate
meanfun	0.952
IQR	0.891
Q25	0.868
sd	0.805
sp.ent	0.758
sfm	0.698
meanfreq	0.645
centroid	0.645
skew	0.631
median	0.614
mode	0.612
kurt	0.605
maxdom	0.604
dfrange	0.602
meandom	0.563
mindom	0.556
maxfun	0.553
minfun	0.545
Q75	0.532
modindx	0.503

Table 1: Individual feature threshold success rates for all features

The five features that gave the best success rates were meanfun, IQR, Q25, sd, and sp.ent.

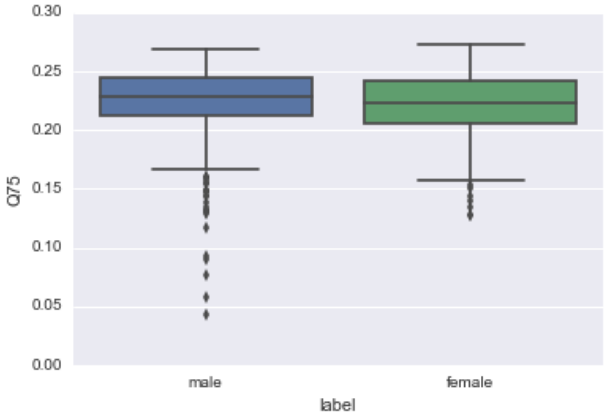


(a) meanfun (success rate - 0.952)

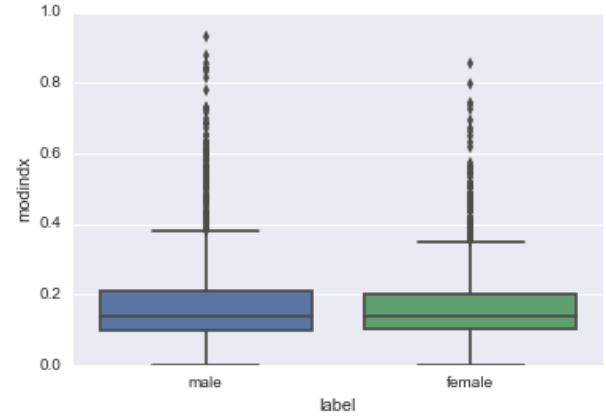


(b) IQR (success rate - 0.891)

Figure 1: Boxplot of the most discriminative features



(a) Q75 (success rate - 0.532)



(b) modindx (success rate - 0.503)

Figure 2: Boxplot of the least discriminative features

5. Algorithms

We tested the following algorithms on the data:

5.1. Simple Classifiers

a. Perceptron
The perceptron algorithm is used when the data set is potentially linearly separable and contains binary classifiers. This algorithm uses a linear predictor function and combines a set of weights with the feature vector. If the data is not linearly separable, the algorithm

never converges and thus the perceptron algorithm fails. We wanted to start out with a simple classifier to set a baseline success rate if it converges.

b. Pocket Learning
In general, if the perceptron algorithm does not converge, then there is not output and we can use a pocket learning algorithm. This model calculates the number of misclassified data points and quantifies the error for each iteration of the perceptron algorithm to find the weight vector that minimizes the misclassification error.

5.2. Support Vector Machine (SVM)

Similar to the perceptron algorithm, the SVM attempts to find an optimal hyperplane that separates the data points. Unlike the perceptron algorithm, the SVM converges to a single hyperplane that maximizes the margin between classes of data points and minimizes the no. of misclassified points.

This algorithm attempts to solve the following optimization problem:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{n}\xi_i \\ \text{s.t. } \forall i \quad & y_i(\mathbf{w}^T\mathbf{x}_i) \geqslant 1-\xi_i \\ \forall i \quad & \xi_i \geqslant 0 \end{aligned} \tag{1}$$

where ξ_i is called the slack variable and allows the input $\mathbf{x}_i$ to be closer to the hyperplane (or even on the wrong side) but results in a penalty in the objective function. The variable C controls the strictness of the SVM. We can intuitively see that as C becomes larger, the classifier tries to get all points on the right side of the classifier, whereas for small C, the classifier becomes too loose and sacrifices too many points to obtain a simpler solution.

The hyperparameter for this model is the variable C which we optimize using a validation set. Another hyperparameter which needed tuning was the gamma parameter for the RBF kernel.

5.3. Logistic Regression

In logistic regression, we estimate probabilities using a logistic function and we use this to measure a relationship between the categorical dependent variable and one or more independent variables. In the case of our project, the dependent variable refers to the gender, and the independent variable refers to one of the many quantifiers of voice frequency. This model calculates the probability of each categorical variable. Then, the quantifier with highest probability is selected.

Logistic regression models the probability of a dependent variable given the independent variables as:

$$P(y|\mathbf{x}) = \frac{1}{1+e^{-y(\mathbf{w}^T\mathbf{x})}} \tag{2}$$

The problem with this model is that having a categorical outcome variable violates the assumption of linearity in normal regression. In the case of logistic regression, a logarithmic transformation is used on the outcome variable to deal with this issue.

We use L2 regularization along with logistic loss in our model. The regularization parameter C which controls the importance of the regularization term is tuned using a validation set.

5.4. Random Forest

This algorithm creates multiple random decision trees that predict the class of a sample. Decision trees [1, 2] are non-parametric learning methods used for classification of data. In general, decision trees tend to overfit to the data on which they are trained. They are usually improved by using bagging or boosting. Bagged decision trees are called Random Forests. Each decision tree is constructed on a subsample of the data set and then the predictions are averaged. This tends to reduce the variance of the model significantly.

The hyperparameters to this model are no. of trees in the forest and maximum depth of each tree. We use a validation set to find the best values for these parameters.

5.5. Dimensionality Reduction - Principal Component Analysis (PCA)

Another method of modeling is focusing on the most important features of the data. Dimensionality reduction reduces the variables that we consider when fitting our data. This removes unnecessary data and allows us to focus on the main features of the data and avoid any noise present in our data set. In particular, PCA uses and orthogonal transformation to convert a set of observations of possibly related variables into a set of set of values of linearly uncorrelated variables called principal components. In this project, we perform PCA at the end of the all our classification models and re-performed them using the reduced dimensionality.

Although, the dimensionality of our data is quite low and PCA would be unnecessary in this context, we explored this method with the supposition that we can reduce noise. Typically, the first few principal components contain the most information and the last few components contain noise. With this idea, if we perform PCA on our data and use only the first few dimensions, we might be able to improve the accuracy of our model.

Please refer to our iPython notebook attached to the class github (`https://github.com/ORIE4741/projects/blob/master/IGBVS.md`) for the code for all the above algorithms. All the model parameters including additional graphs and plots can be found in the notebook.

All the code is written in Python and to implement the algorithms, we use the Scikit-learn package. The results can be reproduced by downloading the notebook and the data from the repository. Make sure to install the appropriate packages before running the code.

6. Model Analysis

We measure the performance of all our models with the success rate on a test set. For this reason, we split our data in training and testing set. For validation purposes, we also have a validation set. Therefore, we split our full data set in a 70:10:20 ratio where 70% is for training, 10% is for validation, and 20% for testing purposes. We define success rate as -

$$\text{success rate} = \frac{\text{\# of correct predictions}}{\text{total \# of samples}} \tag{3}$$

Here is a table summarizing the results of all the models we explored.

Algorithm	**Success Rate**
Pocket learning	0.833
SVM	0.953
Logistic Regression (LR)	0.981
Random Forest (RF)	0.981
SVM after PCA (15 dims)	0.954
LR after PCA (15 dims)	0.981
RF after PCA (15 dims)	0.951

Table 2: Summary of results

The pocket learning algorithm has a success rate of 0.833 which is the least among all the models. This is expected because the data is definitely not linearly separable. Among SVM, Logistic Regression, and Random Forest, SVM is the least performing with 0.953 which is still much higher than the pocket learning algorithm. Random Forest seems to perform well possibly due to being low-variance models. Since they are a class of ensemble models, they are very less likely to overfit and hence more likely to have a high success rate on unseen data. Logistic Regression is a probabilistic model, and works very well for noisy data and for problems where the independent variables only give a probabilistic estimate of the dependent variable. Whereas, SVM works well in high dimensions and in problems where the independent variables most certainly predict the dependent variable. So this tells us that our data is probably noisy. Performing PCA to reduce noise was not that helpful either. The success rate of SVM improved, that of LR remained the same, and that of RF decreased. We suspect this is because the scales of different features were not the same. Variance of features with higher scales will be higher. Since PCA picks the dimensions with high variance, these features will be picked up as the most significant directions. Therefore, those with low variance, but significant predictive power will be picked in the end. Although we did not try it, normalizing the dimensions before performing PCA or performing a scale invariant dimensionality reduction might help in reducing noise.

7. Conclusion

Out of the algorithms we used, logistic regression would be the best for modeling our data. With a success rate of 0.981, it comes out on top of the other algorithms that we tested. However, we probably will not use this model in production, as there is still scope for improvement. By getting better features, creating a more sophisticated model, and acquiring more data, our model can be improved significantly. The inclusion of more data points especially would be a great boon, as 3,168 is a pretty low number of samples. The project was a good proof of concept since it showed us that such models have the potential to allow us to map our data, but we feel that our results can be improved.

References

[1] CS 4786 Fall 2015: Decision Trees,
http://www.cs.cornell.edu/courses/cs4780/2015fa/web/lecturenotes/lecturenote17.html

[2] Decision Tree: Wikipedia,
https://en.wikipedia.org/wiki/Decision_tree_learning