Developing A Classifier For Parkinson's Disease

Edward Shiang

Contents

1 Background **1**

1.1 Introduction 1

1.2 Variable Selection & Manipulation 2

2 Results **4**

2.1 Tuning Parameters 4

2.2 Final Model 4

3 Conclusion **5**

3.1 Strengths 5

3.2 Weaknesses 5

1 Background

1.1 Introduction

The objective of this project was to design, implement, and refine a statistical model that could predict if a patient had Parkinson's disease as accurately as possible upon being provided a specific pool of 310 initial predictor options.

In terms of the predictability vs. interpretability trade-off, the primary goal of this project fell into prioritizing accurate classification over drawing insight from the model's structure. Therefore, it was decided that despite handicapped interpretability, complicated models that produced more correct classifications were favored over simpler ones that did not.

A neural network is a design for a function: it takes input and spits out output after executing a series of computational steps (Miles Chen 102B Lecture Notes 4-1). Inspired by the neural structure of the human brain, the input passes through layer(s) of nodes. While a neural network may be intricate and lack a simple interpretation, through tuning its features via cross validation, a neural network can produce incredibly accurate results. Because of this project's direct emphasis on accuracy as stated earlier, a neural network was selected and coded as the classification model.

1.2 Variable Selection & Manipulation

The "metric" behind variable selection was the magnitude of a potential predictor's correlation with respect to the category response variable, which represents the strength of a linear relationship between the two variables. Through variable selection, it was crucial to filter out insignificant information without carelessly erasing information valuable for predicting Parkinson's disease.

If the correlation value's magnitude of a potential variable and category fell below 0.2, interpretation-wise, that meant that if a simple linear regression line was fitted to predict Parkinson's disease exclusively from that variable, less than 4% of the variation in the Parkinson's category would be explained by it. Ultimately, a cut-off of 0.2 in correlation with category seemingly represented a balance of eliminating unrelated variables while remaining wary to not recklessly discard information. Thus, only 89 variables in the dataset that had a magnitude of 0.2 or higher (at least a weak to moderate correlation value) remained in consideration for the final model: the others lying below this threshold were neglected.

The motivation for variable transformation stemmed from the relatively high dimensionality of data that existed: with a vast number of possible predictor variables still in consideration (89) relative to number of training observations (88) as well as numerous instances of multicollinearity, it appeared that principal component analysis would effectively achieve dimension reduction while preserving essential information.

The criterion behind deciding the groups to separately perform principal component analysis lied with differentiating the broad dysphonia measure categories commonly used to predict Parkinson's disease. In the article provided, primarily three were discussed: Jitter/Shimmer/Other, SNR, and MFCC. They measure different sources: for example, the Jitter/Shimmer/Other group measures target vocal fold vibration patterns, while the MFCCs focus on articulator placement. In addition, variables within a general metric group may share similarities, and some groups may be more closely linked to Parkinson's than others.

Ultimately, variables already meeting the "0.2 correlation threshold" that were part of one of these categories were retained and respectively partitioned into four groups: these were `DYS1` (Jitter/Shimmer/Other: 26 variables), `DYS2` (signal-to-noise ratio: 10 variables), `MFCC` (Mel-Frequency Cepstral Coefficient: 4 variables), and `IMF` (Intrinsic Mode Function: 4 variables). and principal component analysis was performed on each of them. The groups are shown here:

DYS1
Jitter->pitch_PQ5_classical_Schoentgen
Jitter->pitch_percent
Jitter->F0_abs_dif
Jitter->F0_PQ5_classical_Schoentgen
Jitter->F0_dif_percent
Jitter->F0_TKEO_mean
Jitter->pitch_abs
Jitter->pitch_FM
Jitter->F0_FM
Shimmer->Ampl_TKEO_prc25
Jitter->pitch_PQ5_generalised_Schoentgen
Shimmer->Ampl_abs0th_perturb
Jitter->F0_TKEO_prc5
Jitter->F0_PQ5_generalised_Schoentgen
Jitter->pitch_TKEO_prc25
Shimmer->Ampl_AM
Jitter->pitch_TKEO_prc95
Jitter->pitch_TKEO_prc5
Shimmer->Ampl_TKEO_prc95
Shimmer->Ampl_TKEO_prc75
Shimmer->Ampl_PQ3_generalised_Schoentgen
Shimmer->Ampl_PQ11_generalised_Schoentgen

DYS1
Shimmer->Ampl_PQ5_generalised_Schoentgen
Jitter->F0_TKEO_prc25
Jitter->F0_TKEO_std
Jitter->F0_TKEO_prc95

	DYS2
name1	DFA
name2	HNR->HNR_dB_Praat_std
name3	VFER->SNR_SEO
name4	GNE->std
name5	VFER->SNR_TKEO1
name6	VFER->NSR_TKEO1
name7	PPE
name8	VFER->NSR_SEO
name9	VFER->SNR_TKEO
name10	GNE->mean

MFCC
MFCC_2nd coef
MFCC_1st coef
MFCC_0th coef
MFCC_3rd coef

IMF
IMF->NSR_SEO
IMF->NSR_entropy
IMF->SNR_TKEO
IMF->SNR_entropy

Before conducting principal component analysis, the training data was centered and scaled for each group. Principal component analysis was then executed for the training data in the form of singular value decomposition. The final number of principal components was decided based on loose adherence to the Kaiser's Rule, which was to keep the components with eigenvalues that were approximately no less than 1 (as they contained less information than a single variable did).

In the end, 11 principal components were ascertained for the training data: 5 for `DYS1`, 3 for `DYS2`, 2 for `MFCC`, and 1 for `IMF`. These components, along with the variable `Log energy` (which did not fit into any of these groups but had a strong correlation with category of 0.446), comprised of the final set of inputs into the neural network (final set of predictors).

2 Results

2.1 Tuning Parameters

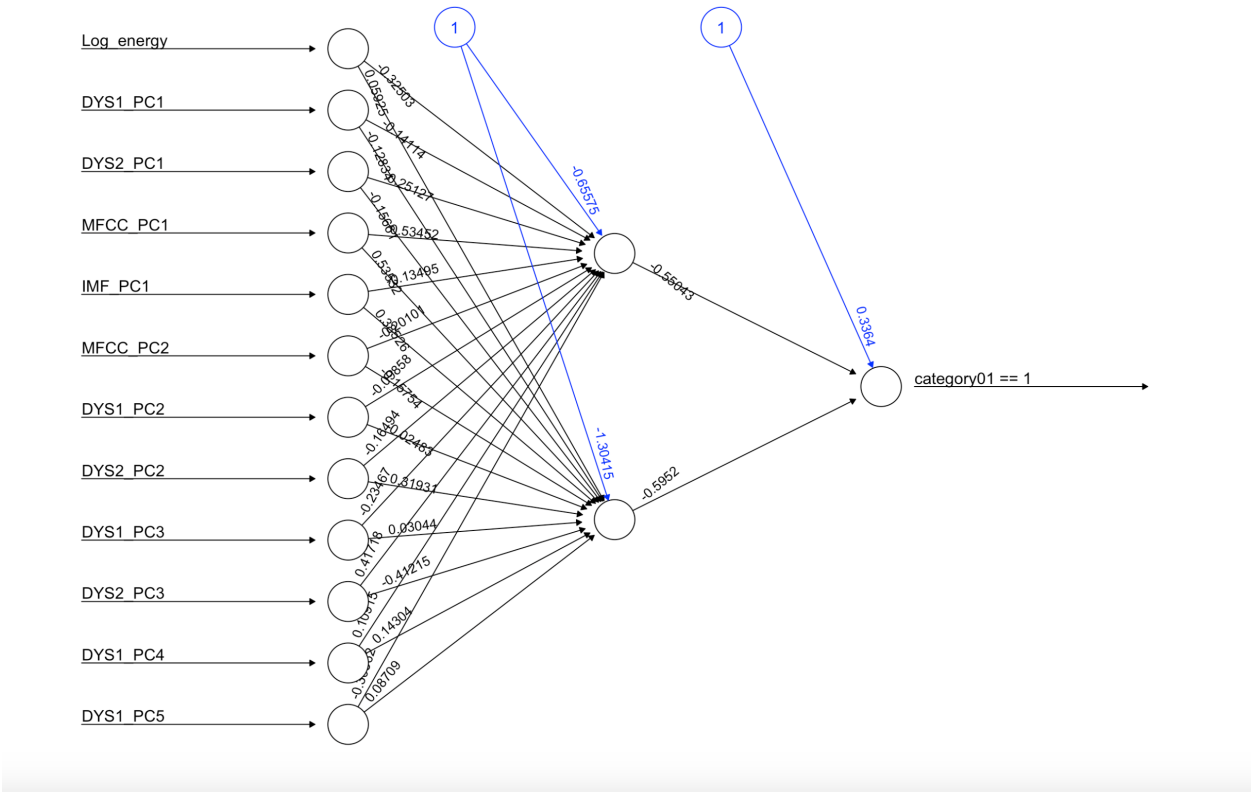
Weight initialization was done through Xavier initialization: setting the weight values to those from a distribution of mean 0 and variance of 1 / # of inputs. For ascertaining the parameters of the neural network function (such as # of hidden nodes, algorithm, activation function, etc.), a k-fold cross validation function of k = 4 was coded and repeatedly executed to gauge a rough estimate of how the neural network is performing through checking an outputted table. The following table was returned after executing the final set of parameters chosen besides # of hidden nodes (which was assumed to be best if between 1 and 12, the range of # of inputs):

I	meanTest	meanTrain	medianTest	medianTrain	testSD
1	0.239	0.182	0.227	0.152	0.136
2	0.091	0.110	0.091	0.106	0.083
3	0.125	0.095	0.114	0.098	0.114
4	0.227	0.110	0.227	0.076	0.083
5	0.193	0.068	0.205	0.061	0.078
6	0.159	0.057	0.159	0.061	0.131
7	0.159	0.076	0.182	0.053	0.136
8	0.216	0.038	0.250	0.038	0.155
9	0.136	0.068	0.136	0.045	0.083
10	0.159	0.049	0.182	0.038	0.120
11	0.205	0.098	0.182	0.114	0.045
12	0.261	0.125	0.273	0.136	0.068

In order from left to right, the columns are I (# of hidden nodes in the neural network), meanTest (mean test error rate), meanTrain (mean train error rate), medianTest (median test error rate), medianTrain (median train error rate), and testSD (standard deviation of the error rates). For these tables, the median and mean values help provide a sense of the distribution skewness, while the pairwise test and train comparisons can show a glimpse into the amount of overfitting that potentially happens. Examining the table above, it appears that I = 2 is the best number of hidden nodes among the 12 (lowest meanTest, lowest difference between meanTest and meanTrain, lowest testSD).

2.2 Final Model

```
finalModel <- neuralnet(category01 == 1 ~., hidden = 2, data = trainData,
                        linear.output= F, learningrate = 0.01,
                        threshold = 2.5, algorithm = 'rprop+',
                        startweights = startWeights, act.fct = 'tanh')
plot(finalModel)
```



The diagram above illustrates the final model generated: there are 12 inputs (Log energy + all the principal components based on dysphonia measure type), which undergo a series of calculations (weight terms indicated as the black numeric values, bias terms as the blue ones) shown in the hidden and output layers. Other features of the neural network decided after cross-validation are mentioned here. The threshold, which essentially is a measure of when to stop iterations, is 2.5. The type of algorithm used is resilient backpropagation with weight tracking. Again, the starting weights are determined by the Xavier initialization technique. The activation function used is tangent hyperbolicus.

3 Conclusion

3.1 Strengths

The primary strength of this model is its prediction power. According to the k-fold cross-validation results, this model had a mean prediction error rate of 9% with a standard deviation of around 8%. For the test data on Kaggle, this model had a 5-6% error rate, which was the 2nd lowest value in the class leaderboard.

A corresponding strength lies with the success of achieving dimension reduction through utilizing knowledge on dysphonia measures for Parkinson's without resulting in a model that fails to predict accurately. The input size that the final neural network takes in is less than 4% of the total number of predictors: despite this, the model is capable of having an estimated predictive power of over 90%.

3.2 Weaknesses

The main flaw of this model is its limited interpretability, which is a fundamental characteristic of neural networks. Explanations and insight of the model's structure are simply not nearly as digestible as that of logistic and multiple linear regression models.

In addition, a potential flaw lies with how the train error was slightly less than the test error: this might suggest a small amount of overfitting to the training data.

Future areas for further study include examining how simpler, more interpretable models fare with perhaps a subset of the predictors, trying different preliminary correlation thresholds other than 0.2, doing principal component analysis on rationale not based on dysphonic measure type, and achieving a model that generates consistently comparable test and train error rates.