Predicting the Rate of Progression of the ALS Disease

Michael Black, Durga Ganesh, Neeharika Madduri

I. Introduction

ALS is a neurodegenerative disease with few known causes. The rate of progression of the disease varies significantly among patients, and it is not known which features in patients correspond to these differences. The identification of predictive factors and then the subsequent development of models can have a positive impact on a patient's diagnosis. 5,600 people are diagnosed with ALS each year and an estimated 30,000 Americans are currently living with the disease, many of whom are close to us.

Our goal is to try to predict the rate of progression of the disease in patients given a set of predictors that include everything from demographics to medical history to lab test outcomes. The features we obtained were taken during the first three months of a clinical trial, and then the rate of progression of the disease was measured during the following nine months of the trial. The progression of the disease is our response variable for this problem and is measured using the ALS Functional Rating Scale, which is a measure from 1 to 40 of a patients ability to carry out ordinary tasks such as walking, talking, etc. Our training set was taken from older trials, while our test set included all very recent subjects. The reasoning behind this was that we want try to predict future disease progression and this test set gives us the most similar subjects to test our procedures on.

We broke this problem into a regression and classification problem to get better predictive results. In the regression setting, our goal was to predict the exact rate of change of the disease, so we used various techniques which included random forests, boosted decision trees, and neural networks. In the classification setting, we first ordered and categorized our dataset into a fast progression of the disease and a slow progression of the disease. We then used Softmax regression and the SVM to classify each patient into categories. With the predictions from both settings, we were able to add a confidence to our predictions (i.e. if both methods predict similar results we were more confident). This topic was also part of a Kaggle competition for Stats 202. For this class a simple boosted decision tree was used for regression, but all the models described in this paper were used for this class.

II. Related Work

Since ALS is a disease that affects many individuals and is not understood entirely, there have been many attempts to try to predict how the disease will progress in patients. In order to obtain accurate predictions, we needed to first understand our features and how they contributed to the overall progression of the disease. One proposed method to measure the association of the features with ALS shown in [1] was to perform univariate logistic regression. With this model, the authors were able to show that the onset between diagnosis and symptoms were a very important factor for prediction while features such as past clinical history and physical activity were less so. An additional feature shown to be very predictive of shorter survival times was whether the patient began his or her disease with a bulbar onset, which is the degeneration of motor neuron cells in the bulb [2]. Additional lab result features and their descriptions are summarized in [3].

One very successful approach to this problem detailed in [4] made use of random forests to get good predictive results. The reasoning behind this was that there is a lot of variability in the data, and random forests aim to reduce the variance of the predictions by resampling the data set and only considering a subset of the features at each decision tree node. Another benefit of using sets of variables in a random forest is the notion of variable importance that comes out of constructing many trees from the sampled dataset. With an idea of the important features used for accurate predictions, the authors could augment their current models. From the previous approaches used in tackling this problem, we noticed that a very flexible and nonlinear model would be necessary to make reasonable predictions. As a result, we knew that variance would be the biggest roadblock in our way of achieving a low mean squared error. To try to relieve this problem, we looked into many statistical methods such as bootstrapping and then cross validation and hypothesis tests to compare models [5].

III. Dataset and Feature Selection

We acquired a training set of 2424 training examples and a test set with 101 examples with 858 features. The test set was acquired from more recent patients and since the goal was to predict the progression of the disease on new patients, this test set gives us the best way to test how our model will perform on patients we haven't seen before. We obtained the raw data from Prize4Life and the PRO-ACT database [3]. The initial data had many unmeasured values (NA) if the doctor didn't collect/measure a particular feature from a patient. As an initial step we threw out all features for which the fraction of training examples that contain a not recorded value was more than 25%, with the reasoning that there wasn't enough data to be useful for prediction. With the remaining features that had less than 25% of examples not recorded, we replaced the missing entries with the median of that feature.

After this simple clean up, we still had 337 features. Adding additional features that are not useful for prediction to a model causes the training error to go down, but also increases the variance of a particular method. Since test error can be decomposed into a bias and variance term, increasing the variance without decreasing the bias much will increase the overall error. Because of this, we wanted to reduce our feature set further. The most powerful method we found for doing this was regularization. Since penalizing the size of the 2-norm of the coefficients in linear regression results in small but non-zero coefficients, we decided to try penalizing the 1-norm. This

resulted in a sparse set of coefficients which is exactly what we wanted for feature selection. The L-1 regularization cost function we used is shown below in (1) below.

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}|\beta_j| \tag{1}$$

This problem can also be formulated as an optimization problem, where the coefficient estimates solve (2).

$$\underset{\beta}{\text{minimize}}\left\{\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2\right\} \quad \text{subject to} \quad \sum_{j=1}^{p}|\beta_j|\le s \tag{2}$$

The parameter, s, can be interpreted as the budget you have for the magnitude of your coefficients. This shows why the L-1 regularization technique is so effective for feature selection. If a feature is irrelevant, it is best to make it zero and not contribute to your fixed budget. Since we really want to solve the optimization problem in (3) when doing feature selection to find the best subset of features, the L-1 regularization serves as a computationally feasible alternative [6].

$$\underset{\beta}{\text{minimize}}\left\{\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2\right\} \quad \text{subject to} \quad \sum_{j=1}^{p}\boldsymbol{I}(\beta_j\neq 0)\le s \tag{3}$$

After running least squares with L-1 regularization, we got a mean squared error of 0.3795 and narrowed down our feature set to just 49 features. Since lambda is a tuning parameter in L-1 regularization, we used cross validation to find the best lambda. Figure 1 shows the mean squared error (and its standard error) for different values of lambda evaluated on our test set. The best lambda value we obtained was 0.00859.

The final technique we used to narrow down our feature set was forward and backward selection. With only 49 remaining features, these methods were computationally feasible so we were able to obtain our final dataset of 23 features which we used for all our prediction methods.

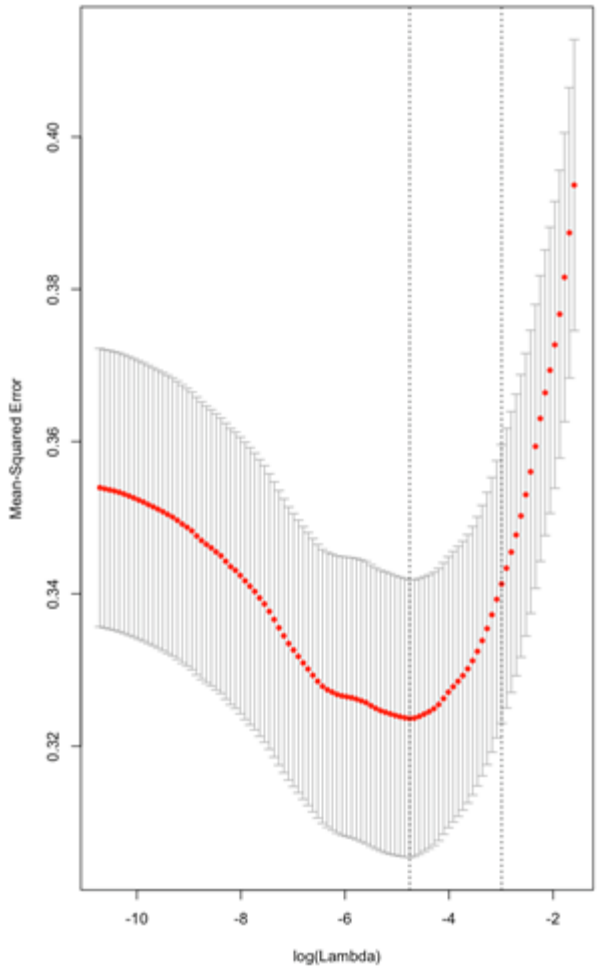

Fig. 1. CV Error for Lambda tuning in L-1 regularization

IV. Methods

We used three main regression techniques to predict the rate of progression of the ALS disease in patients. Since the relationship between the features and the response was very nonlinear, we used methods that allowed us to try to model such data.

The regression prediction method that gave us the best results was decision trees because trees can deal with data that is clustered together well whereas polynomial of different order fits cannot. At each node of a decision tree, a binary split of the data is performed so as to minimize the training RSS. This is accomplished by ordering each feature and then considering all possible splits of all features to determine the one split of a single feature that minimizes the training error, depicted in (4), where the regions R_j are the regions corresponding to the leafs of the tree after each split. This process continues at every node until there are only a handful of observations in each leaf of the tree. A piecewise constant value corresponding to the average of the observations in a given leaf is then predicted.

(4)

$$\sum_{j=1}^{J}\sum_{i\in R_j}\left(y_i-\hat{y}_{R_j}\right)^2$$

Since this is prone to overfitting, we wanted to try a couple techniques that would reduce the variance of this procedure. One such method was with a random forest. A random forest makes use of a technique called bootstrapping that amounts to resampling the dataset with replacement to obtain a series of bootstrap replicates. With each sampled dataset, we can build a decision tree and then average the final predictions in order to reduce the variance of our method. Furthermore, a random forest does not consider every variable at a split like a traditional decision tree. By only considering a subset of the features at each split, each bootstrapped decision tree will look very different, so the average of the trees will do a better job at reducing the variance. By implementing these modifications to the traditional decision tree, we were able to significantly reduce our test MSE.

An additional technique we employed was boosting in order to augment the predictions of our decision tree. Boosting is a method that in each iteration trains a decision tree on the residuals of predictions made by the previous tree. By training on the residuals, the method is able "learn" from what it could not accurately model in the last iteration, as shown in (5). By summing up shrunken sub-trees (6), we obtain a better predictive tree. The number of trees used in this approach is an important parameter since it can lead to overfitting. The rate at which the method "learns" is controlled by the shrinkage parameter (lambda) and can be found using cross validation [6]

(5)

$$\hat{f}(x)\leftarrow\hat{f}(x)+\lambda\hat{f}^b(x)$$
$$r_i\leftarrow r_i-\lambda f^b(x_i)$$

(6)

$$\hat{f}(x)=\sum_{b=1}^{B}\lambda\hat{f}^b(x)$$

Our last method for regression involved building a neural network model. Our neural network model had one hidden layer and an output layer. The neurons in the hidden layer have the sigmoid transfer function and the output neuron has a linear transfer function. The model has 23 inputs which is the number of features of the dataset. The model is shown in the Figure 2. We used 80% of the data for training, 10% for validating the model and 10% for testing the model.

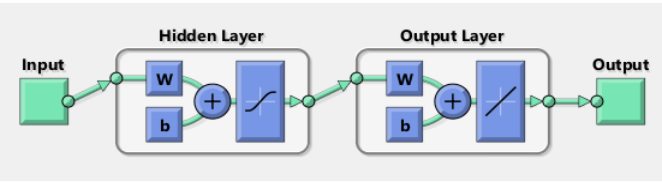

Fig. 2. Artificial Neural Network Model

Our next goal was to transform the regression problem into a classification problem. We did this by categorizing our training and test sets into three categories – fast, moderate, and slow progression of the disease.

We implemented a Softmax regression model on the data set and replaced the logistic sigmoid function with its multi-class equivalent, Softmax function given by (7)

(7)

$$s(x,W,i)=\frac{\exp\left(w_i^{-T}\phi_i\right)}{\sum_{k=1}^{K}w_k^{-T}\phi_k}$$

(8)

$$\hat{C}=argmax_i\, s(x,W,i)$$

W is a matrix of the weights for each class, with w_i being the i-th column. In order to map from the output of the Softmax function to a class label, the maximum value of the Softmax function can be used, (i.e. the estimated label is given by (8) which corresponds to maximum-a-posteriori (MAP) class labelling for the generative model that has been learnt. To define our indicator function (T) in this case, we simply found the 2nd quantile of the training targets, and split them into three classes as follows:

1. y < -0.8946, T = [1 0 0]
2. -0.8946< y < -0.4, T = [0 1 0]
3. y > -0.4, T = [0 0 1]

To optimize the feature weights we made use of the stochastic gradient descent algorithm. This calculates the log-likelihood gradient for small batch (100 data points) and then updates the current estimate of the parameters by taking a step in the direction of the calculated gradient according to (9), where $E_i(w)$ is the error function we want to minimize (i.e. the negative log-likelihood) calculated for each data point (or batch) i, at the current weight values w (for all classes).

(9)

$$w:=w-\nabla E_i(w)$$

After implementing the Softmax regression model, we realized that the classes could not be separated using a linear boundary. To get better classification results we tried two additional methods involving the SVM. Both models used just two classes for classification instead of the three used in the Softmax regression model. In the first SVM, we threw out the observations belonging to the middle class (corresponding to -0.8946 < y < -0.4) so that the two classes that were more likely to be separable and used a linear kernel. In the second SVM, we used the entire dataset (splitting the two classes down the middle) and then applied an SVM with a Gaussian kernel.

Our best classification results came from using an SVM with a Gaussian kernel (10). The Gaussian kernel maps the features into an infinite dimensional feature space, thereby increasing the likelihood that the two classes are separable [7]. Since the kernel trick allows the dot product of infinite

dimensional vectors to be computed using the original feature vectors, the SVM is still computationally feasible.

$$k(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right) \tag{10}$$

V. Results & Discussion

The results shown in this section were reported using our test set. We held out 101 observations from more recent clinical trials with which to test and compare our methods.

The random forest and boosted decision tree variants of the classical decision tree proved to do very well in prediction. The reason for this is that both of these methods aim to reduce the variance using a single tree by averaging the predictions of many trees. The random forest does this by averaging the predictions of many bootstrapped samples. The boosted decision tree accomplishes this goal by adding together many shrunken sub-trees trained using residuals.

The parameters for the random forest are the number of trees and the number of parameters to consider at each node split. The number of trees to choose is not an interesting parameter since there is a point of diminishing returns, so a sufficiently high number of 1000 was chosen to ensure that we converged to the lowest MSE. The number of parameters to consider at each node was chosen using 10-fold cross validation and was determined to be five. This corresponds to roughly the square root of the number of features.

The parameters needed for the boosted decision tree are the number of trees, the shrinkage parameter, and the interaction depth. The interaction depth is simply how many levels we want each sub-tree to have. Again, 1000 trees was chosen and the shrinkage parameter and interaction depth was chosen using 10-fold cross validation. An example of the test and training MSE for different shrinkage parameters is shown in Figure3.

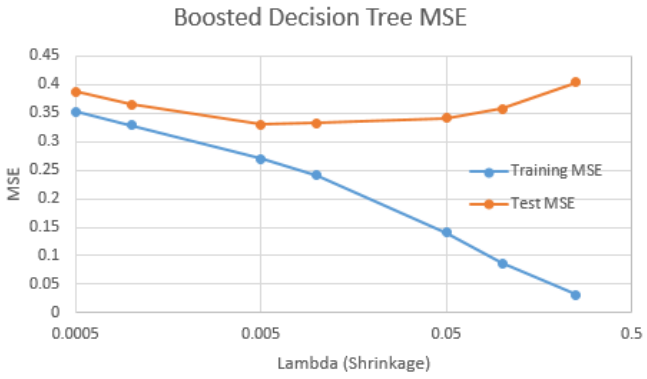

Fig. 3. Training & Test MSE for shrinkage parameter with 1000 trees

When training the neural network model, we tried three different algorithms and varied the number of neurons in the hidden layer to determine the best model. Of our original dataset, we used 80% of the data to train the model, 10% as a validation set, and 10% as a test set. The test MSE using the 10% from the training set versus the number of neurons in the hidden layer for the three different algorithms is shown in Figure 4.

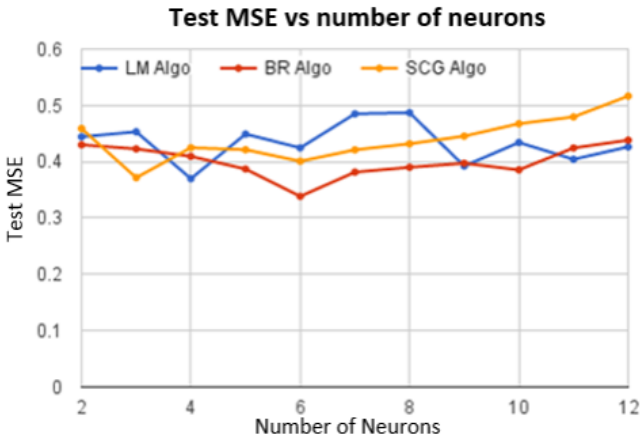

Fig. 4. Test MSE as we vary the number of neurons in the hidden layer

From this plot, we chose the Bayesian Regularization algorithm with six neurons in the hidden layer for our model. The last parameter we needed to select was the number of iterations performed during training. The training, validation, and test MSE are shown in Figure 5 for different numbers of iterations. We then trained our final model with four iterations and inserted our actual test set to get the final test MSE for this method (reported in Table 1).

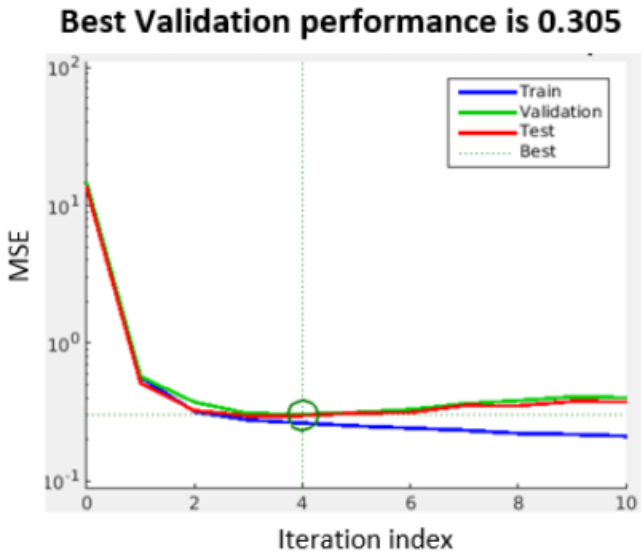

Fig. 5. MSE for train, validation, and test sets

The mean squared error on the test set is reported in Table 1 for the three regression methods detailed above. The mean squared error using a linear regression and also for just predicting the mean response (of the training set) are also shown for comparison.

TABLE I. TEST MSE FOR REGRESSION METHODS

Method	*Test MSE*
Predict Mean	0.4284
Linear Regression	0.5568
Random Forest	0.3364
Boosted Decision Tree	0.3304
Neural Network	0.3385

The accuracy of the classification methods we used are reported using confusion matrices shown in Figures 6, 7, and 8. The Softmax regression model with three classes had an accuracy of 50.7%. This showed that simple linear decision boundaries did not perform well with this dataset. Using the SVM with the linear kernel and two classes, we achieved an accuracy of 80.9%. However, in this model, we discarded the middle class in order to increase the likelihood of separability. Using all of the data, the best accuracy we achieved was 99.4% using the SVM with the Gaussian kernel.

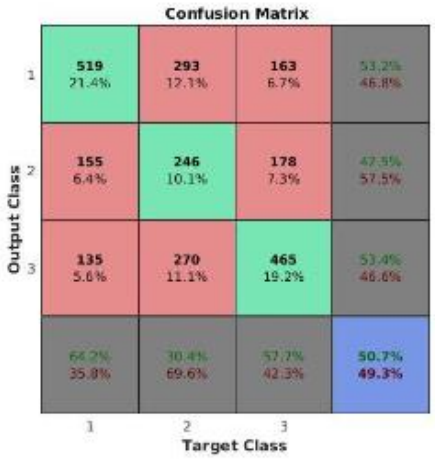

Fig. 6. Confusion Matrix for Softmax regression model

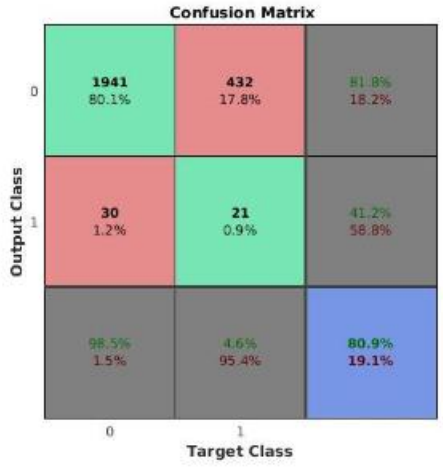

Fig. 7. Confusion Matrix for SVM with linear kernel

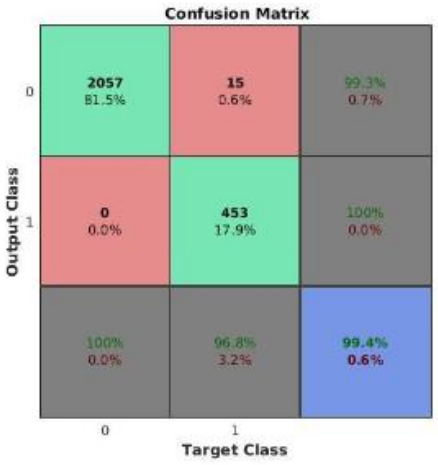

Fig. 8. Confusion Matrix for SVM with Gaussian kernel

VI. CONCLUSION

ALS is a disease that still is not well understood. With past medical data taken at clinical trials, we hope to be able to better understand the underlying factors to lead to the progression of the disease and thus diagnose it better in the future. As we continue to collect data, the algorithms will continue to perform better and learn patterns and structure that can prove to be very beneficial to doctors and the families of those affected.

In this paper, we explained our methods for regression and also classification. When performing regression, the decision trees worked the best. In order to get the best performance out of the trees, we used techniques such as bootstrapping and boosting to reduce our variance and thus improve our test MSE. In the classification setting, the SVM with a Gaussian kernel was very accurate. Using both the regression output and the classification category, we could give a confidence level to our predictions. For example if the rate of progression of the disease was predicted to be a small number by the random forest but the SVM classified it as a fast progression, we would be less confident and have to run that example through different regression methods to get a more accurate result.

For future work, we would like to continue to gather data and modify the algorithms we used to make better predictions. We explored many algorithms in the time we had, and got a very good feel for which ones worked well and why they worked well. A next step would be to ensemble all the regression methods into one and used that one to make better predictions. If one method performs well on data points in which another does not, we could use the two models together to improve our accuracy. One way to do this would be to train a global model that selects which of the three models to use depending on the test observation.

ACKNOWLEDGMENT

We would like to thank PRO-ACT and Prize4Life for the raw data, and also Ji Park, Yash Deshpande, Kenneth Jung, Lester Mackey, Kris Sankaran, and Vatsal Sharan for the derived features.

REFERENCES

[1] Qureshi MM, Hayden D, Urbinelli L, Ferrante K, Newhall K, Myers D, et al. Analysis of factors that modify susceptibility and rate of progression in amyotrophic lateral sclerosis (ALS). Amyotroph Lateral Scler. 2006;7: 173–82.

[2] Gordon PH, Cheng B, Salachas F, Pradat PF, Bruneteau G, Corcia P, et al. Progression in ALS is not linear but is curvilinear. Journal of Neurology. 2010;257:1713–7.

[3] Prize4Life, Neurological Clinical Research Institute, How to Use PRO-ACT, 2015. URL https://nctu.partners.org/ ProACT/.

[4] Torsten Hothorn & Hans H. Jung (2014) RandomForest4Life: A Random Forest for predicting ALS disease progression, Amyotrophic Lateral Sclerosis and Frontotemporal Degeneration, 15:5-6, 444-452

[5] Hothorn T, Leisch F, Zeileis A, Hornik K. The design and analysis of benchmark experiments. Journal of Computational and Graphical Statistics.2005;14:675–99.

[6] G. James, D. Witten, T. Hastie, R. Tibshirani, An Introduction to Statistical Learning with Applications in R, Springer, New York, ISBN: 978-1461471370, 2013.

[7] M. Akay, Support Vector Machines Combined with Feature Selection for Breast Cancer Diagnosis. Expert Systems with Applications. March 2009. Volume 36, Issue 2, Part 2