Using feedforward neural networks for speech measurement based parkinson's disease score prediction

Angel Astudillo Aguilar angel.astudillo@est.fib.upc.edu
Daniel García Zapata daniel.natanael.garcia@est.fib.upc.edu
Julius von Kügelgen julius.von.kugelgen@est.fib.upc.edu
Pablo Reynoso Aguirre pablo.eliseo.reynoso@est.fib.upc.edu

Abstract

In this report, we investigate the use of feedforward neural networks as a CI approach for non-linear regression on a large clinical dataset. We show that the inherent grouping structure of the data by patients suggest standardization by subject for preprocessing, and perform model selection on different network architectures and training algorithms. Our final estimate of the generalization error is comparable to other global approximation techniques from standard statistics, and inferior to a CART approach, which yields us to hypothesize that local approximation methods like CART or RBF networks might be superior for datasets like the one analysed here. Finally, we suggest further investigation of dimensionality reduction to decrease generalization error, as our brief analysis shows promising trends on this regard.

1 Problem statement and goals

The goal of this work is to predict indicator scores for Parkinson's disease (PD) as accurately as possible. In particular, we attempt to predict the total- and motor components of the unified Parkinson's disease rating scale (UPDRS) which is one of the most commonly used evaluation measures in clinical study [1]. The benchmark accuracy we wish to overcome lies at an average mean absolute test error of 5.8 points for the motor-UPDRS score, and 7.5 points for the total-UPDRS score. These values were achieved in [2], see section 2 for a more detailed description of their work. Our attempt is focused on the use of feedforward artificial neural networks, multi-layer perceptrons to be more precise, as a non-linear regression technique.

2 Previous work

Our project is based on the largest dataset in existence combining speech measurements with UPDRS scores. This data was collected in [3] using a device developed by Intel, the At-Home Testing Device (AHTD). A set of 52 subjects used the AHTD at their home to record speech measurements with the integrated microphone over a six-months period. Subsequently, a range of classical and non-classical speech signal processing algorithms was used to extract 16 different dysphonia measures which together with the age and sex of the subjects can be used as features for regression analysis. Of the 52 initial patients, 2 left the study early and 8 did not provide sufficient test data, leaving a total of 42 patients and 5875 observations in the dataset. The total and motor UPDRS scores were measured for each patient at the beginning of the study, as well as after three and six months, and intermediate values were linearly interpolated between these medically assessed scores.

Our regression analysis extends the work of [2] who did an extensive correlation analysis of the features and found that they were heavily correlated. They considered least squares (LS), iteratively re-weighted least squares (IRLS), least absolute shrinkage and selection operator (LASSO), and classification and regression trees (CART) as different regression techiques and found that the CART approach as only non-linear model performed significantly better than the linear models, with a training MAE of 4.5 for the motor and 6.0 for the total UPDRS, and a testing MAE of 5.8 for the motor and 7.5 for the total UPDRS scores. The other approaches achieved an MAE of about 6.8 for the motor and about 8.5 for the total UPDRS score with almost no difference between training and testing errors.

3 (CI) methods

All coding is done in MATLAB using built-in functions from the Statistics & Machine Learning, and the Neural Network toolboxes.

3.1 Linear mixed model as first approach

As a very first approach, we decide to investigate the grouping structure of the data using a linear mixed model, a standard statistical modelling technique developed by Fisher [4] and later refined by Henderson [5]. Mixed models, also referred to as random effects models, are often used to investigate data which contains observations from different groups and can be written in their most simple version as:

$$\mathbf{y} = X\beta + Z\nu + \epsilon \tag{1}$$

where the response, $\mathbf{y}$, is composed of the fixed effect term $\mathbf{y}$ and some normally distributed error ϵ as in ordinary regression, and an additional term $Z\nu$ which encodes the random effects. ν is a vector of random effects of length #(groups) which are assumed to be normally distributed with mean zero and covariance matrix $\sigma_\nu^2 I$, and the matrix Z encodes the group structure of the data, that is $z_{ij} = 1$ if observation i is in group j, and zero otherwise.

In our case, there is a clear grouping of the data into patients/subjects. The main motivation of having a look at the data using a mixed effects model is to check whether or not random effects from the grouping of the data are negligible or whether they should be taken into consideration. This is helpful when deciding how to normalise the data later on: normalising by subject or using global statistics for all patients combined.

3.2 Data preprocessing

The very first decision we make is to treat subjects as a whole by never splitting the observations corresponding to a given patient, as we consider this to be the most realistic and sensible approach. In the end, the prediction task should be application oriented and the real world situation would be to predict the UPDRS score with our model given the data from one new patient, whose information has not been used in training. Considering all rows as independent observations on the other hand, would firstly make it impossible to model the testing of our model with a new patient, rather than mixed observations from different sources, and secondly introduce bias since correlated data from the same patient might very likely be used both in the training and the testing phase.

Hence, we split the data into a training and a test set while maintaining patients as a whole. We decide to use 12 subjects for testing and 30 subjects for model selection and training. The splitting is done by randomly drawing 12 patients without replacement. This corresponds to roughly 28.5% for testing and 71.5% for training, which should allow for both a reasonably good training and also a narrow enough confidence interval for the estimate of the generalization error of our model.

3.2.1 Normalization

The second step we take in processing our data is to standardize it, that is to transform it to have zero mean and a standard deviation of 1. There are two ways of standardizing this dataset. The first approach is to consider the mean and standard deviation for each feature globally (on the training set), that is considering observations from all subjects when standardizing. The second approach is to standardize features by subject, that is computing the mean and standard deviation for each patient separately. The second approach is assuming a heterogeneous distribution of feature values among the groups, while the first assumes that there are no significant differences between different patients.

The results obtained in our first investigation with mixed effect models (see section 4.1) hint at the fact that inter-subject differences cannot be neglected easily and therefore we choose the second approach as our primary approach, that is centering observations by patient. This also has the advantage that we can use information from the test set in standardizing it by the patients contained in it, rather than using the somewhat unrelated mean and standard deviation from the training set as we would have to with the first approach.

However, we will also consider standardizing the data globally as a secondary approach to see whether there is any difference in the prediction quality.

3.2.2 Averaging over identical time measurements

Whereas there are about 150 observations for each patient, these actually only correspond to 25 different time points, since each measurement has been repeated six times. We therefore consider additionally averaging the data of measurements for a given patient from the same day. This has the advantage of making the data more consistent, as even though the measured features differ throughout the six measurements, the interpolated UPDRS scores are the same for all of them, so that we are attempting to predict the same response with different feature values. However, the obvious drawback is that in doing so we reduce the size of the dataset by a factor of six.

Together with the two choices from the previous section this results in four different ways of preprocessing the data which are summarized in the following table:

Dataset	**Standardization method**	**Averaging over six measurements**
D1	by subject	no
D2	by subject	yes
D3	globally	no
D4	globally	yes

Table 1: Datasets obtained through different preprocessing approaches

We treat D1 as our primary dataset, but also investigate the resulting predictions from the other three.

3.3 Cross validation

For model selection we use 10-fold cross validation [6] on the training data (30 subjects). In order to do so, we split the training set into 10 mutually exclusive validation sets consisting of three subjects each. We then use these in each fold to estimate the generalization error while using the remaining 27 subjects for training. Once the final model has been selected, we retrain it using the entire training set (including the validation data) and finally use the test set to estimate the generalization error.

3.4 Principal Component Analysis

Prior to training our neural networks, we apply principal component analysis (PCA) [7]. This has the purpose of uncorrelating the input data which in its raw form is highly correlated as shown in [2], see section 2. In order to not introduce any bias, we only apply PCA on the subset of the data used for training (for each fold), and use the extracted transformation to transform the validation set. Similarly when estimating the final error we apply PCA to the standardized training set, and transform the test set with the corresponding transformation prior to prediction.

3.5 Feedforward neural networks

As a main CI technique and regression model we use multi-layer perceptrons (MLP), a form of feedforward artificial neural networks which are built by stacking multiple perceptrons [8] on top of each other. Since we wish to predict two outputs, the motor and total UPDRS, we use two MLPs one for each output. The number of neurons in the input layer is fixed by the number of features used, 18 in our case. Finding the right number of hidden layers and the number of units within them, on the other hand, remains one of the unsolved tasks in this research area [9]. Since there is no rule or algorithm to determine the optimal number of hidden layers, we decide to test MLPs with one and two hidden layers as the size of our data set does not suggest to use more layers than that and since any function can in theory be represented with two hidden layers.

In most cases, there is no correct way to determine the best configuration for hidden units without training the several neural nets and estimating the generalization error. According to Geman, Bienenstock, and Doursat [10], if you have too few hidden units, you could get a high training error and high generalization error due to underfitting and high statistical bias. If you have too many hidden units, you may get low training error but still have high generalization error due to overfitting and high variance. "A rule of thumb is for the size of this [hidden] layer to be somewhere between the input layer size and the output layer size" [11]. According to this rule, and for the sake of exploration we train with up to 20 hidden units while using different amounts of regularization to penalise large weight values.

As activation functions we use the hyperbolic tangent for the hidden layers, and the unit function for the output layer. Since we wish to compare our results to those from [2] where the MAE is used as evaluation measure, we decide to comply with this choice and hence optimize our network with MAE as a performance function. We consider different methods for training the neural network: Levenberg-Marquardt (LM), Bayesian Regularization (BR), Conjugate Gradient with Powell/Beale Restarts (CGD) and gradient descent with momentum (GDM) as first order methods, and BFGS Quasi-Newton as a second order method.

4 Results and discussion

4.1 Mixed model results

Our first investigation of the data using a linear mixed model indicates that for both the total and motor UPDRS scores random effects from grouping are not negligible. The estimated variance of the random effects per subject is 9.8 for the total and 7.6 for the motor UPDRS, with 95% confidence intervals of [7.6, 12.6] and [5.9, 9.7] respectively.

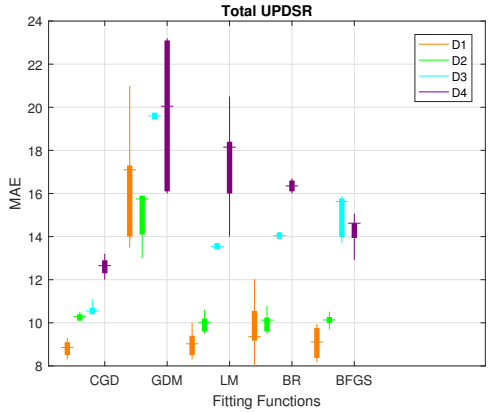
The errors on a testing set of 12 subject for the mixed effect models lie at approximately 6.5 points for the motor and 8.4 points for the total UPDRS. These results are slightly better or equal to those achieved using the LS, IRLS and Lasso methods, and worse than those of the CART results in [2]. However it has to be said that we only perform one run of the mixed linear effect model as we are not really

considering it as a final model, but use it rather as a tool to gain insights on the effect of the grouping of the data.

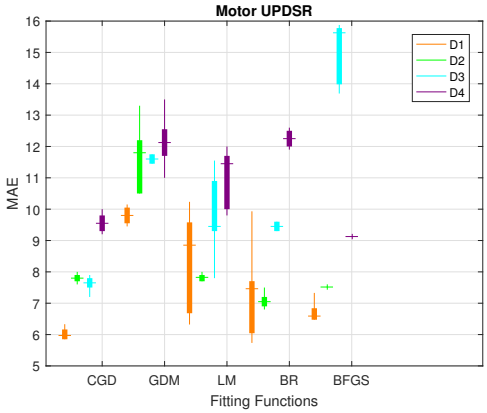
4.2 Dataset selection

We divide the process of model selection into two parts: first we investigate which of our four differently preprocessed datasets (see table 1) is the most promising, and then we tune our network architecture and training algorithm on this most promising dataset.

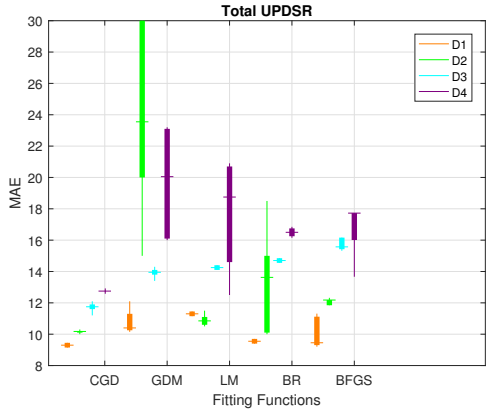
For the sake of facilitating the exploration of the different datasets, we will fix two architectures at this point: one hidden layer with twenty hidden units, and two hidden layers with twenty hidden units each. For each of the two architectures, we test the different training algorithms mentioned in section 3.5 on all four datasets using 10-fold cross validation with varying regularization and report the minimum validation errors achieved on each. The results are presented in figure 1.



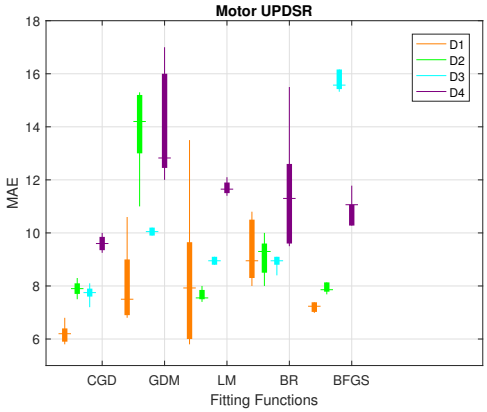
(a) total UPDRS, 1 hidden layer



(b) motor UPDRS, 1 hidden layer



(c) total UPDRS, 2 hidden layers



(d) motor UPDRS, 2 hidden layers

Figure 1: Boxplot comparison of the four datasets for different fitting functions and network architectures

It is apparent from figures 1a - 1d, that the dataset that achieves the lowest validation MAE throughout the different architectures both for the motor and total

UPDRS is D1 (orange), which is standardized by subject and not averaged over different time measurements. This coincides with our prior intuition that treating subjects as a whole is more realistic and sensible, and that averaging results in too much of reduction in size of the dataset. Consequently, we will from now on focus on dataset D1 for fine tuning. Also, we will narrow our search regarding training algorithms down to Bayesian Regularization (BR), Conjugate Gradient with Powell/Beale Restarts (CGD) and BFGS Quasi-Newton (BFGS) which achieved the lowest validation MAEs, also discarding gradient descent with momentum due to its large MAE, and Levenberg-Marquardt partially due its reduced compatibility with regularization and MAE as performance measure, and partially due to the large standard deviations shown with this method for the motor UPDRS.

It is worth mentioning that tuning Conjugate Gradient descent gives small variances, whereas Bayesian Regularization and BFGS Quasi-Newton are more susceptible to varying performance dependent on tuning the parameters as can be seen in the figure. Furthermore, we notice that both for the total and motor UPDRS, the minimum validation errors overall are achieved with only one hidden layer, as opposed to two. According to Panchal, Ganatra, Kosta and Panchal, "For many practical problems, there is no reason to use any more than one hidden layer. For nearly all problems, one hidden layer is sufficient" [12]. They mention that two hidden layers are required for modeling data with discontinuities. If this is not the case using two hidden layers rarely, if ever, improves the model. In our example this seems to be the case, as using two hidden layers does not improve performance, but even increases the error.

We will therefore focus in the following section on finding the most promising number of hidden units and the best training algorithm, for training an MLP with one hidden layer on D1.

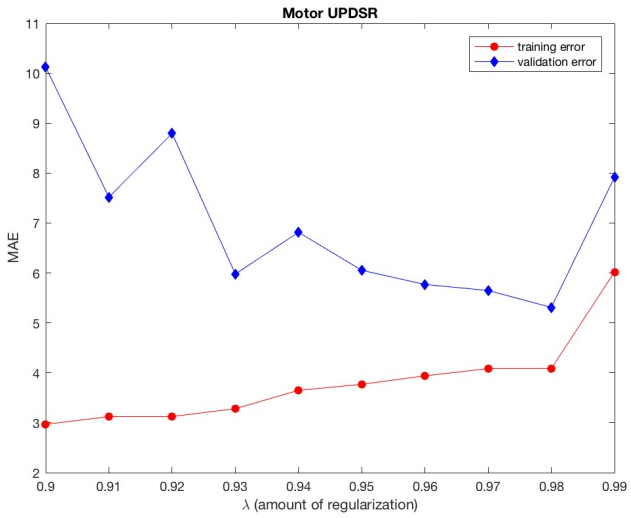


Figure 2: An example plot for finding the minimum validation error by means of regularization

4.3 Model selection

We now fine tune our 1-hidden-layer MLP on dataset D1 investigating the three different training algorithms described above while varying the number of hidden units. For each validation error estimate we again perform 10-fold cross validation with regularization. For each fold and configuration (training algorithm, number of hidden units, total/motor), we obtain a plot similar to the one shown in figure 2 from which we pick the minimum validation error and the corresponding amount of regularization, $\lambda \in [0, 1]$. Averaging these over the 10 folds yields the results presented in tables 2a-2f.

# Neurons	Val. error	λ
5	7.3162	0.96
10	7.3805	0.9
15	**7.1974**	**0.96**
20	7.3671	0.94

(a) Total UPDRS, Bayesian Regularization

# Neurons	Val. error	λ
5	5.4431	0.82
10	5.4223	0.95
15	**5.3516**	**0.92**
20	5.5354	0.96

(b) Motor UPDRS, Bayesian Regularization

# Neurons	Val. error	λ
5	7.608	0.73
10	8.4611	0.75
15	8.1981	0.93

(c) Total UPDRS, BFGS Quasi-Newton

# Neurons	Val. error	λ
5	5.41	0.85
10	5.4004	0.85
15	5.47	0.85

(d) Motor UPDRS, BFGS Quasi-Newton

# Neurons	Val. error	λ
5	8.0993	0.79
10	8.0459	0.81
15	7.64	0.84
20	8.6582	0.15

(e) Total UPDRS, Conjugate Gradient Descent

# Neurons	Val. error	λ
5	8.7884	0.1
10	8.3431	0.5
15	7.2551	0.85
20	8.2836	0.9

(f) Motor UPDRS, Conjugate Gradient Descent

Table 2: Model selection results with one hidden layer using dataset D1

Coincidentally, we come to the conclusion that both neural networks, motor and total UPDRS, should have the same architecture with one hidden layer and fifteen hidden units, differing only in the regularization parameter. Further, we find that the best training algorithm for both UPDRS scores is Bayesian Regularization. Figure 3 shows the final architecture for both neural networks.

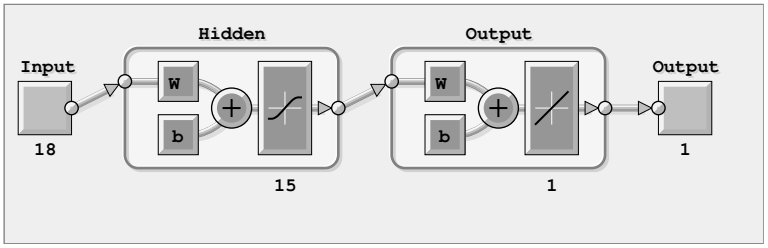


Figure 3: Neural Network Architecture

4.4 Estimating the generalization error

Once the the final network structure is decided, we retrain our MLP on the entire training set (including the validation data) and evaluate its prediction on the separate testing set. This gives us a point estimate of the generalization error of our model. Since such a point estimate has a high variance, we repeat the process of randomly drawing 12 test subjects, training our final model on the remaining 30 subjects, and evaluating it on the test set for 100 times. Finally, the so obtained more reliable estimates of the generalization error of our model are reported in table 3.

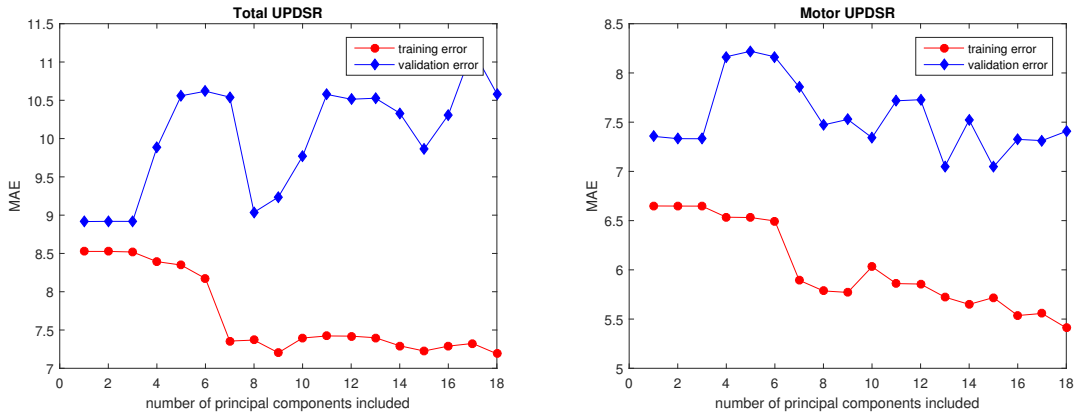
UPDRS	**Training MAE**	**Test MAE**
total	7.49 ± 0.70	9.14 ± 1.61
motor	5.61 ± 0.50	6.92 ± 1.16

Table 3: Final estimates of training and generalization errors

As can be seen from the table, the training errors achieved by our final architecture are reasonably good, but the generalization errors are significantly larger and lie above the benchmark values we set out to beat. This indicates that despite our efforts, the final model fails to generalize from the data sufficiently.

4.5 Effect of dimensionality reduction

Whereas we have so far only used PCA to decorrelate the input variables for improved network training, PCA can also be a very powerful technique to reduce the dimensionality of the input data while maintaining a large amount of the variability of the data by dropping only the least significant components. Motivated by the lack of generalization observed in the previous section, we finally conduct a brief analysis of the impact that dimensionality reduction might have on the training and validation errors.



(a) total UPDRS, $\lambda = 0.73$

(b) motor UPDRS, $\lambda = 0.85$

Figure 4: Effect of dimensionality reduction through PCA on training and validation errors: for both scores one hidden layer with 5 units was trained using the BFGS Quasi-Newton algorithm with the indicated amounts of regularization.

Figure 4 shows the results of training a one hidden layer MLP with 5 units using the BFGS Quasi-Newton algorithm with the amounts of regularization shown in tables 2c and 2d for this configuration. Reported are the average validation MAEs from 10-fold cross validation against the number of principal components of the training data used for training and evaluating the model.

As can be seen, the training error decreases as more components are included, as expected. More interestingly however, we observe that the validation error for both UPDRS scores when including all components is larger then for some smaller subsets, and that this effect is much more pronounced for the total than for the motor UPDRS. We found that only the first 10 principal components account for 98-99% of the variance of the data (depending on the split) which suggests that the number of inputs can be significantly reduced without losing much variability of the data.

5 Strengths and weaknesses

In the course of this project, we have conducted an extensive data analysis, investigating different ways of preprocessing the data. We were able to prove our initial intuition right, that the most favourable way of doing so is using the entire, non-averaged dataset and to standardize it by subject. Subsequently, we followed precisely the standard pipeline for fitting a neural network by performing model selection on a training set using 10-fold cross validation, and evaluating the final model on a separate test set. In doing so we did not only consider and learn about different training algorithms, but also the effect that the number of hidden layers and units may have on the generalization performance of the network.

Unfortunately, our final results are not very satisfying since we were not able to reach our initial goal of beating the benchmark errors set by the CART approach in [2]. Furthermore, we unfortunately did not have time to address the analysis of the effect of dimensionality reduction with the approprioate rigour. Note that in figure 4 the same architecture, training algorithm and regularization are used for different numbers of input dimensions. Much better results could be expected if we had the time to fine tune the models with selected subsets of principal components, e.g. the first 8 and 13 principal components for total and motor UPDRS, respectively.

6 Conclusions and future work

In summary, we used an MLP with one hidden layer as non-linear regression technique in an attempt to achieve better prediction accuracy for motor and total UPDRS scores than those shown in previous work. Mainly due to a lack of generalization of our method, we were not able to outperform the CART method which achieves the best results on this dataset. It might well be the case that not reducing the dimensionality of the input data from the beginning on was a mistake, and it would be very interesting to put some more effort into obtaining potentially much better performance using a dimensionality reduced set of input data to avoid overfitting.

The finding that an MLP was not even able to achieve better prediction results than the standard statistical techniques such as LASSO or Least Squares approaches used by previous authors might however also hint at the conclusion that this dataset is very hard to predict using such global approximation methods. CART on the other hand, the most successful method on this dataset so far, uses systematic splitting of the data combined with local approximations. In order to test the hypothesis that local approximators are preferable to global ones on the given dataset, it would therefore be interesting for future work to consider the use radial basis function (RBF) networks as a CI technique, which are one of the neural network variants corresponding to local, rather than global approximation techniques.

References

[1] Claudia Ramaker et al. "Systematic evaluation of rating scales for impairment and disability in Parkinson's disease". In: *Movement Disorders* 17.5 (2002), pp. 867–876.

[2] Athanasios Tsanas et al. "Accurate telemonitoring of Parkinson's disease progression by noninvasive speech tests". In: *IEEE transactions on Biomedical Engineering* 57.4 (2010), pp. 884–893.

[3] Christopher G Goetz et al. "Testing objective measures of motor impairment in early Parkinson's disease: Feasibility study of an at-home testing device". In: *Movement Disorders* 24.4 (2009), pp. 551–556.

[4] Ronald Fisher. "The Correlation between Relatives on the Supposition of Mendelian Inheritance." In: *Transactions of the Royal Society of Edinburgh* 52.2 (1918), pp. 399–433.

[5] Charles R Henderson et al. "The estimation of environmental and genetic trends from records subject to culling". In: *Biometrics* 15.2 (1959), pp. 192–218.

[6] Peter A Lachenbruch and M Ray Mickey. "Estimation of error rates in discriminant analysis". In: *Technometrics* 10.1 (1968), pp. 1–11.

[7] Karl Pearson. "LIII. On lines and planes of closest fit to systems of points in space". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11 (1901), pp. 559–572.

[8] Frank Rosenblatt. "The perceptron: A probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6 (1958), p. 386.

[9] Shuxiang Xu and Ling Chen. "A novel approach for determining the optimal number of hidden layer neurons for FNN's and its application in data mining". In: (2008).

[10] S. Geman, E. Bienenstock, and R. Doursat. "Neural Networks and the Bias/Variance Dilemma". In: *Neural Computation* (1992), pp. 4, 1–58.

[11] A. Blum. *Neural Networks in C++. An Object-Oriented Framework for Building Connectionist Systems*. Ed. by John Wiley & Sons. Vol. 1. 1. John Wiley & Sons, 1992.

[12] Gaurang Panchal et al. "Behaviour analysis of multilayer perceptronswith multiple hidden neurons and hidden layers". In: *International Journal of Computer Theory and Engineering* 3.2 (2011), p. 332.

Contents

A Linear mixed model results

The following are the results for fitting a linear mixed effects model to the UPDRS scores using all features and a random effects term conditional on the subject.

For the total UPDRS:

```
Linear mixed-effects model fit by ML

Model information:
    Number of observations            4153
    Fixed effects coefficients          19
    Random effects coefficients         30
    Covariance parameters                2

Formula:
    Linear Mixed Formula with 19 predictors.

Model fit statistics:
    AIC      BIC      LogLikelihood    Deviance
    20715    20848    -10337           20673

Fixed effects coefficients (95% CIs):
    Name                   Estimate    SE          tStat       DF      pValue         Lower         Upper
    '(Intercept)'           -2.6736      14.144    -0.18903    4134       0.85008      -30.403      25.056
    'age'                   0.38579     0.21686       1.779    4134      0.075312    -0.039367     0.81096
    'sex'                   -1.8854      3.6949    -0.51028    4134       0.60989      -9.1295      5.3586
    'Jitter___'             -32.308      80.844    -0.39963    4134       0.68945      -190.81      126.19
    'Jitter_Abs_'            3738.8      4128.5     0.90559    4134       0.36521      -4355.4       11833
    'Jitter_RAP'               4618       16217     0.28477    4134       0.77584       -27176       36412
    'Jitter_PPQ5'            38.242      71.634     0.53385    4134       0.59347       -102.2      178.68
    'Jitter_DDP'            -1529.2        5406    -0.28287    4134       0.77729       -12128      9069.4
    'Shimmer'                10.567      22.194     0.47615    4134       0.63399      -32.944      54.079
    'Shimmer_dB_'           0.71594      1.7679     0.40497    4134       0.68552      -2.7501      4.1819
    'Shimmer_APQ3'          -3215.2       16273    -0.19758    4134       0.84338       -35118       28688
    'Shimmer_APQ5'           1.9908      18.683     0.10656    4134       0.91515      -34.638      38.619
    'Shimmer_APQ11'          4.7184      8.0767      0.5842    4134       0.55912      -11.116      20.553
    'Shimmer_DDA'            1058.2      5424.2     0.19508    4134       0.84534      -9576.1       11692
    'NHR'                   -0.96353      2.2639    -0.4256    4134       0.67042      -5.4021      3.475
    'HNR'                    0.12316    0.030472      4.0418    4134    5.4001e-05     0.063422    0.18291
    'RPDE'                   3.2833     0.78411      4.1873    4134    2.8822e-05        1.746     4.8205
    'DFA'                    1.3978      1.4949     0.93505    4134       0.34982       -1.533     4.3285
    'PPE'                   0.52257      1.1388     0.45887    4134       0.64635      -1.7101     2.7553

Random effects covariance parameters (95% CIs):
Group: subject_ (30 Levels)
    Name1                Name2                Type         Estimate    Lower     Upper
    '(Intercept)'        '(Intercept)'        'std'        9.7923      7.6017    12.614

Group: Error
    Name             Estimate    Lower    Upper
    'Res Std'        2.8386      2.778    2.9005

MAEtotal =
    8.4134


SDtotal =
    5.6065
```

And for the motor UPDRS:

```
Linear mixed-effects model fit by ML

Model information:
    Number of observations            4153
    Fixed effects coefficients          19
    Random effects coefficients         30
    Covariance parameters                2

Formula:
    Linear Mixed Formula with 19 predictors.

Model fit statistics:
    AIC      BIC      LogLikelihood    Deviance
    18919    19052    -9438.6          18877

Fixed effects coefficients (95% CIs):
    Name                    Estimate    SE          tStat        DF      pValue        Lower        Upper
    '(Intercept)'            0.50056     10.939      0.04576     4134        0.9635      -20.945      21.946
    'age'                    0.23927     0.16762      1.4275     4134       0.15351    -0.089347     0.56789
    'sex'                    -0.0783      2.8559    -0.027416    4134       0.97813      -5.6775      5.5209
    'Jitter___'              -4.3032      65.142    -0.066059    4134       0.94733      -132.02      123.41
    'Jitter_Abs_'             1189.7      3326.6      0.35763     4134       0.72064      -5332.3      7711.7
    'Jitter_RAP'              7210.1       13067      0.55178     4134       0.58113       -18408       32829
    'Jitter_PPQ5'            -17.443       57.72     -0.30219     4134       0.76252      -130.61      95.721
    'Jitter_DDP'             -2385.9        4356     -0.54772     4134       0.58391       -10926      6154.2
    'Shimmer'                -6.0987      17.883     -0.34103     4134        0.7331      -41.159      28.962
    'Shimmer_dB_'             1.4807      1.4245       1.0394     4134       0.29866      -1.3121      4.2735
    'Shimmer_APQ3'               1383       13112      0.10548     4134         0.916       -24324       27090
    'Shimmer_APQ5'            15.496      15.054       1.0294     4134       0.30337      -14.018      45.011
    'Shimmer_APQ11'           3.4449       6.508      0.52933     4134       0.59661      -9.3143      16.204
    'Shimmer_DDA'            -472.62      4370.7     -0.10813     4134       0.91389      -9041.5      8096.2
    'NHR'                    -1.6471      1.8242     -0.90294     4134       0.36661      -5.2236      1.9293
    'HNR'                   0.083657    0.024553       3.4072     4134    0.00066268     0.035519     0.13179
    'RPDE'                     2.447      0.6318        3.873     4134    0.00010916       1.2083       3.6857
    'DFA'                    0.85796      1.2044      0.71235     4134       0.47629      -1.5033      3.2193
    'PPE'                    0.71213     0.91763      0.77606     4134       0.43776      -1.0869      2.5112

Random effects covariance parameters (95% CIs):
Group: subject_ (30 Levels)
    Name1               Name2               Type         Estimate    Lower     Upper
    '(Intercept)'       '(Intercept)'       'std'        7.5684      5.8752    9.7496

Group: Error
    Name             Estimate    Lower     Upper
    'Res Std'        2.2872      2.2384    2.3371

MAEmotor =
    6.4674

SDmotor =
    4.3146
```