Applications of Evolutionary Computation to Regression with Multilayer Perceptron Artificial Neural Networks

Donald Dunagan (dgd45125@uga.edu) and Andrew Stout for the course CSCI 6560, 11 December 2018

**Abstract**

While there exist many machine learning algorithms capable of performing regression, that is, the calculation of real-valued output based on input, such as linear regression, support vector machines, and regression trees, this project is interested in performing regression with multilayer perceptron artificial neural networks developed with the aid of evolutionary computation. Specifically, the first experiment involves using a genetic algorithm to tune the hyperparameters for a Scikit-learn multilayer perceptron while the second involves using a genetic algorithm and then an evolution strategy to set the node weights of fixed architecture multilayer perceptrons. The dataset to be analyzed is The Parkinson’s Telemonitoring Dataset, a dataset from the University of California Irvine Machine Learning Repository. The dataset was derived from speech samples taken from patients with Parkinson’s Disease (PD) with each observation containing 16 measurements of disordered speech, 2 measurements of PD severity, and information about the patient. The objective is to predict PD severity based on the measures of disordered speech found in voice samples.

**1. Introduction**

Through neurological connection, information is received from the outside world and it is through these same connections that muscles are controlled in the body. Neurodegenerative diseases and disorders such as Alzheimer’s Disease, epilepsy, and Parkinson’s Disease often reduce a patient’s capacity, through the degradation of neurological pathways, for both.

After Alzheimer’s Disease, PD is the most common neurodegenerative disorder [1] with estimates of more than one million people being affected just in North America [2]. Additionally, twenty percent of people with PD end up never being diagnosed [3]. With age as the most dominant risk-factor for the onset of PD and the fact that after the age of fifty, the risk of onset increases greatly [4], the number of people with PD is expected to increase as the worldwide population ages [5]. Current drug treatments are incapable of curing or reversing the effects of PD [6]. However, some therapies such as Lee Silverman Voice Treatment LOUD have the ability to increase a patient’s quality of life. In the case of LSVT LOUD, this is through improvements in facial expression, speech, and breathing and swallowing ability [7].

As part of their treatment, PD patients undergo subjective physical and vocal tests administered by a clinician with the Unified Parkinson’s Disease Rating Scale (UPDRS) being used as a metric to measure the advancement of PD symptoms. The UPDRS is scaled from 0-176 with 176 being total disability. An overall UPDRS value is composed of three subsections: (1) Mentation, Behavior, and Mood, (2) Activities of daily living, and (3) Motor. While symptoms of PD can include tremor, muscular stiffness, and cognitive difficulty, this paper is most concerned with the vocal impairments associated with PD. Specific traits found in the speech of advanced PD patients include dysphonia (difficulty speaking) and hypophonia (quiet speech volume) [8]. Speech difficulty falls under the Motor subsection of the UPDRS evaluation with Motor being scored from 0-108 and 108 signifying significant impairment.

A new medical technology, telemonitoring, is a possible option for PD patients which can not only reduce inconvenient clinic visits, but offer a potentially more accurate assessment of the patient’s speech than that of a clinician [9]. A task which can be easily integrated into a telemonitoring system and has been shown to be effective in PD symptom monitoring is a sustained vowel phonation, which can be analyzed by signal processing algorithms for PD features [10]. While previous studies [10], [11] have demonstrated that the speech of people with PD can accurately be distinguished from that of healthy individuals, Tsanas *et al.* [9] extend this idea to assigning a UPDRS value to a PD patient based on the severity of their voice symptoms. This is done by applying speech signal processing algorithms to a voice sample to extract dysphonia features which can then be used as features for regression to map speech signal properties to a UPDRS estimation.

This paper investigates the use of multilayer perceptron artificial neural networks developed with the aid of evolutionary computation for regression and compares them to the results presented by Tsanas *et al.* [9] as well as others. Following sections will discuss the data used and its origin, the results of prior work on this dataset, the methods used in this paper, the results of these methods, and concluding remarks.

**2. Data and Previous Work**

The dataset used for the analysis in this paper, The Parkinson’s Telemonitoring Dataset [9], comes out of the work done by Goetz *et al.* [12]. In collaboration with Intel Corporation and supervised by ten medical centers in the United States, 52 patients with PD underwent a six-month study in which they used Intel Corporation’s At-Home Testing Device (AHTD) to record their PD symptoms. One facet of each AHTD session was sustained vowel phonations in which the patient would produce a given sound at a prescribed intensity for as long as possible.

To these phonations, Tsanas *et al.* [9] applied signal processing algorithms and extracted 16 dysphonia measures. Data from 10 of the study participants was dropped, 8 due to inadequate test data and 2 who left the study early, resulting in 5875 instances from 42 patients. Each instance is a speech recording of a sustained vowel phonation of the phoneme /a/, “ahh.” The dataset contains 22 attributes: subject number, age, gender, time from recruitment for the study, motor UPDRS, total UPDRS, and 16 extracted dysphonia measures.

During the six-month trial period, clinical evaluations of UPDRS were taken initially as a baseline, at three months into the study, and at six months at the conclusion. The voice samples, however, were taken on a weekly basis. Because of the lack of weekly UPDRS values, weekly intermediary UPDRS values were interpolated linearly, passing through a patient’s initial, three month, and final UPDRS evaluations. A linear progression of PD symptoms is well supported [13], particularly over a time span of a year or less [14]. The mean baseline total UPDRS was: 26.39 ± 10.80 points, at three months: 29.36 ± 1.82 points, and at six months: 29.57 ± 11.92 points.

Tsanas *et al.* [9] used three linear models and one nonlinear model to predict UPDRS values based on dysphonia measures. The linear models included classical least squares, iteratively re-weighted least squares, to reduce the influence of any large, outlying values, and least absolute shrinkage and selection operator (LASSO). The LASSO was used not only in accordance with the general principle of parsimony, but also in anticipation of the curse of dimensionality due to the sixteen input features. It has been shown that dysphonia measures are highly correlated [10], in which LASSO helps due to its capacity to effectively reduce the coefficients of some features towards zero. The nonlinear model used was a classification and regression tree (CART), specifically, a pruned binary regression tree. The testing mean absolute error (MAE) for total UPDRS following 1,000 repetitions of out of sample testing were 8.5 points for least squares, 8.47 ± 0.27 points for iteratively re-weighted least squares, and 8.6 points for the LASSO. The CART model performed better than the linear models with a testing MAE of 7.52 ± 0.25 points for total UPDRS after 1,000 testing repetitions. These results are extended in Tsanas *et al.* [15] when least squares is applied to a LASSO selected subset of the 13 classical measures and the log transformations of those 13 measures. After 100 repetitions of out of sample testing, the testing MAE was 8.38 ± 0.23 points for total UPDRS. Incorporating the log transformed features into the least squares regression model reduced error compared to the model which did not. However, even with the improvement, least squares did not perform as well as the CART model in Tsanas *et al.* [9].

Additional regression techniques were applied by Eskidere *et al.* [16]: support vector machines (SVM), least squares support vector machines (LS-SVM), multilayer perceptron neural networks (MLP), and general regression neural networks. The testing MAE for total UPDRS following 100 repetitions of out of sample testing was 7.02 ± 0.18 points for SVM, 6.18 ± 0.16 points for LS-SVM, 7.19 ± 0.22 points for MLP, and 8.03 ± 0.19 points for the general regression neural network, respectively.

While an additional study, Hlavica *et al.* [17], has performed regression on this dataset, they only targeted motor UPDRS and did not develop models targeting total UPDRS. While the previously mentioned studies developed models for motor UPDRS as well, this paper only investigates models targeting total UPDRS due to time constraints.

**3. Methods**

The first experiment involved evolving the hyperparameters of a Scikit-learn [18] MLP. In order to get a baseline total UPDRS MAE for the MLP machine learning scheme, a default MLP was used. The default hyperparameters for Scikit-learn MLPs are: a single hidden layer of size 100, the rectified linear unit function (ReLU) [19] as the activation function for the hidden layer, Adam, the stochastic gradient-based optimizer proposed by Kingma and Ba [20], an L2 penalty of 0.0001, a batch size of 200, a constant learning rate, an initial learning rate of 0.001, an exponential decay rate for estimates of the first moment vector of 0.9 and of the second moment vector 0.999, and a maximum number of iterations of 200. After training on 80% of the total available data (4,700 observations), the MLP made predictions for the withheld 20% (1,175 observations) testing data. The resulting MAE between predicted and actual was 8.88 points.

With all of the above hyperparameters plus some others that are not applicable to the default model, the number of hyperparameter combinations is limitless. The idea behind this experiment is to use the principle of evolution, specifically through a genetic algorithm, to find an optimal combination of parameters that results in the development of an accurate model. This can be compared to the brute trial and error evaluation of hypermeter combinations with cross validation or a cross validated grid search of hyperparameters.

Looking at the available hyperparameters, we decided on 5 that would be compatible in all of their different possible combinations. The hyperparameters tuned by the genetic algorithm in this experiment where hidden layer sizes for two hidden layers which were restricted to 100 nodes per layer, the activation function for the hidden layers which was either the identity function, the logistic sigmoid function, the hyperbolic tan function, or the ReLU function, the weight optimization solver which was either the limited-memory Broyden-Fletcher-Goldfarb-Shanno algorithm [21], stochastic gradient descent, or Adam, and the L2 penalty parameter.

To make such a selection of hyperparameters work with a genetic algorithm took some creative thinking, but the implementation follows. Each individual consisted of 5 real valued numbers, one for each hyperparameter to be tuned. For the non-numeric hyperparameters, the value would be constrained to [0,N-1] where N is the number of hyperparameter options. During evaluation, the value would be rounded to the nearest whole number and then correlated to one of the N-1 potential options. The hidden layer sizes were rounded to the nearest whole number and constrained to [1,100] and the L2 penalty parameter was left as is.

The genetic algorithm used was steady state and implemented with the DEAP Python package [22]. The population size was 10. Crossover was one-point crossover with probability 0.7. Mutation was Gaussian mutation with mu 0 and sigma 1 with probability 0.2 for any given value in an individual. Selection was handled by tournament selection of size 3. Fitness was evaluated by training and evaluating a Scikit-learn MLP with the phenotype hyperparameters on the training data. The algorithm was run 30 times with each run being 10 generations long.

The second experiment involved setting MLP weights with a genetic algorithm and an evolution strategy, two dialects of evolutionary computing. The idea behind this experiment was to remove the necessity of hyperparameters to be passed to the Scikit-learn package for model development and to use these two flavors of evolutionary computing to optimize a string of real-valued numbers which would become the weights for the nodes in the MLP. In an attempt to reduce complexity (potentially at the cost of accuracy) and to focus on implementation, the architecture for the MLP was fixed beforehand with the MLP being restricted to a single hidden layer.

The first step in this experiment was to establish an MLP architecture. To do this, it was decided that default Scikit-learn MLPs would be developed differing only in the size of their single hidden layer. The default hyperparameters have been covered previously in the paper. We wanted to maximize the potential accuracy of the MLPs without unnecessarily increasing complexity and set a lower bound of 8 hidden nodes and an upper bound of 128 hidden nodes. We began by evaluating the 5-fold cross validated training MAE of a default MLP with 8 hidden nodes which had an MAE of 8.67 points and did not converge on 4 of the 5 folds. Knowing that we needed a more complex architecture, we went to the other extreme and tested an MLP with 128 hidden nodes with the same method which has a training MAE of 8.53 points and converged for all 5 folds. This was more accurate, than the MLP with 8 hidden nodes, but could the accuracy be improved and/or the complexity reduced? The next MLP to be evaluated had 64 hidden nodes and had a training MAE of 8.51 points. To ensure that this was the sweet spot for our MLP architecture, the final model to be evaluated had 32 hidden layers and a training MAE of 8.56 points. In summary, the most accurate default MLP had 64 hidden nodes in its single layer. This would be our architecture moving forward to developing node weights.

In order to utilize evolutionary computation to optimize the node weights, we needed to define what an individual would be and how its fitness would be evaluated. For both algorithms, an individual had 1088 real-valued weights as with 16 input nodes, 64 nodes in the hidden layer, and a single output node, we needed one weight matrix of 16 by 64 and a second weight matrix of 64 by 1. The process by which the UPDRS values and an individual’s fitness were calculated follows. The 4700 x 16 training data were multiplied by the 16 by 64 matrix and 1 was added to each value in the resulting matrix simulating a bias weight. To this, the ReLU activation function was applied. Then, the result was multiplied by the 64 by 1 weight matrix with a linear activation function to create 4700 predicted UPDRS values. From these values, the training MAE was calculated and returned as the candidate solution’s fitness.

The genetic algorithm used was a steady state genetic algorithm and was implemented with the DEAP package in Python. Each initial individual consisted of a string of 1088 real-valued numbers drawn from the standard normal distribution. The algorithm had a population size of 544. This was done due to having such a high dimensionality and in the interest of computational feasibility. Crossover was one-point crossover which occurred with probability 0.7 and the mutation operator used was Gaussian mutation with mu 0 and sigma 1 which occurred with probability 1/1088 for each value in each individual. Selection was handled by tournament selection of size 50. The algorithm was run 30 times with each run being 40 generations long.

The evolution strategy was also implemented using the DEAP package. The algorithm had a population size of 544 and each individual consisted of 2176 real-valued numbers: 1088 weights and 1088 corresponding strategy values. Mu and lambda were both set to 544. The initial individual weights were random uniformly drawn from the range [-3,3] and each strategy value was random uniformly drawn from the range [0.01,3]. Crossover consisted of two-point crossover on both the individuals and their strategy with probability 0.6. Mutation was log normal strategy mutation. The algorithm was run 60 times with each run being 40 generations long.

**4. Results**

The Scikit-learn MLP models with genetic algorithm tuned hyperparameters resulted in a testing MAE of 8.43 ± 0.11 points. The MLP models with genetic algorithm optimized nodes weights had a testing MAE of 8.48 ± 0.06 points. The MLP models with evolution strategy optimized node weights had a testing MAE of 8.42 ± 0.06 points.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Applied Frameworks and Respective MAEs** | | | | | |
|  |  |  |  |  |  | **Total UPDRS** | |
| **Tsanas et al. 2010** | | |  |  |  |  |
| Least Squares | |  |  |  |  | 8.5 |
| Iteratively Reweighted Lease Squares | | | | |  | 8.47 ± 0.27 | |
| Least Absolute Shrinkage and Selection Operator (LASSO) | | | | | | 8.6 |
| Classification and Regression Tree (CART) | | | |  |  | 7.50 ± 0.25 | |
|  |  |  |  |  |  |  |
| **Tsanas et al. 2010** | |  |  |  |  |  |
| Least Squares with log transformed features | | | | |  | 8.38 ± 0.23 | |
|  |  |  |  |  |  |  |
| **Eskidere et al. 2012** | |  |  |  |  |  |
| Support Vector Machine (SVM) | | |  |  |  | 7.02 ± 0.18 | |
| Least Squares Support Vector Machine (LS-SVM) | | | | |  | 6.18 ± 0.16 | |
| Multilayer Perceptron Neural Network (MLP) | | | | |  | 7.19 ± 0.22 | |
| General Regression Neural Network | | | | |  | 8.03 ± 0.19 | |
|  |  |  |  |  |  |  |
| **Dunagan and Stout Project** | | |  |  |  |  |
| GA Hyperparameter Tuned MLP | | |  |  |  | 8.43 ± 0.11 |
| GA Weight Optimized MLP | | |  |  |  | 8.48 ± 0.06 |
| ES Weight Optimized MLP | | |  |  |  | 8.42 ± 0.06 |

**5. Discussion**

Frankly, the accuracy and predictive capacity of these models is not good. The MLPs which had their node weights optimized with evolutionary computing and the genetic algorithm hyperparameter tuned MLPs performed about as well as the linear models presented by Tsanas *et al.* in [9] and worse than all of the more complex models presented by Eskidere *et al.* in [16]. We will, however, still try and provide some interesting remarks.

The two MLPs which had their node weights optimized via a genetic algorithm and an evolution strategy had similar accuracies. We believe that this is because this is the maximal capability of the fixed architecture that we decided to use for the second experiment. An MLP with a single hidden layer just does not have the capacity for greater accuracy with this dataset. It is reassuring in regard to our implementations to see that the two different approaches to the problem arrived at models with the similar accuracies.

Interestingly, accuracy between the genetic algorithm hyperparameter tuned MLPs and the MLPs which had their node weights optimized via a genetic algorithm and an evolution strategy were quite similar even though the genetic algorithm hyperparameter tuned MLPs had a second hidden layer. We believe that this is due to the two-hidden layer MLPs not being fully optimized (even if they were optimized with the constraints that we enforced). This is supported by the accuracy of the MLPs developed by Eskidere *et al.* [16] which is significantly greater.

Certainly, the primary goal of this paper was to apply different evolutionary computation methods to a nontrivial dataset which has already been analyzed in the literature to investigate what could be accomplished. Even though the results were not good, the authors certainly learned about the application of and gained experience in the implementation of these two different evolutionary computation dialects. A future project which could bring about better results would be to take the same methodology from the second experiment and apply it to a much more complex MLP. This, we believe, could create a model with much better predictive ability.

References

[1] M. C. de Rijk *et al.*, “Prevalence of Parkinson’s disease in Europe: A collaborative study of population-based cohorts. Neurologic Diseases in the Elderly Research Group.,” *Neurology*, vol. 54, no. 11 Suppl 5, pp. S21-3, 2000.

[2] A. E. Lang and A. M. Lozano, “Parkinson’s disease. First of two parts.,” *N. Engl. J. Med.*, vol. 339, no. 15, pp. 1044–1053, 1998.

[3] A. Schrag, “How valid is the clinical diagnosis of Parkinson’s disease in the community?,” *J. Neurol. Neurosurg. Psychiatry*, vol. 73, no. 5, pp. 529–534, 2002.

[4] A. Elbaz *et al.*, “Risk tables for parkinsonism and Parkinson’s disease,” *J. Clin. Epidemiol.*, vol. 55, no. 1, pp. 25–31, 2002.

[5] S. K. Van Den Eeden, “Incidence of Parkinson’s Disease: Variation by Age, Gender, and Race/Ethnicity,” *Am. J. Epidemiol.*, vol. 157, no. >11, pp. 1015–1022, 2003.

[6] N. Singh, V. Pillay, and Y. E. Choonara, “Advances in the treatment of Parkinson’s disease,” *Prog. Neurobiol.*, vol. 81, no. 1, pp. 29–44, 2007.

[7] S. Sapir, J. L. Spielman, L. O. Ramig, B. H. Story, and C. Fox, “Effects of Intensive Voice Treatment (the Lee Silverman Voice Treatment [LSVT]) on Vowel Articulation in Dysarthric Individuals With Idiopathic Parkinson Disease: Acoustic and Perceptual Findings,” *J. Speech Lang. Hear. Res.*, 2007.

[8] A. K. Ho, R. Iansek, C. Marigliani, J. L. Bradshaw, and S. Gates, “Speech Impairment in a Large Sample of Patients with Parkinson’s Disease,” *Behav. Neurol.*, vol. 11, no. 3, pp. 131–137, 1999.

[9] A. Tsanas, M. A. Little, P. E. McSharry, and L. O. Ramig, “Accurate Telemonitoring of Parkinson’s Disease Progression by Noninvasive Speech Tests,” *IEEE Trans. Biomed. Eng.*, vol. 57, no. 4, pp. 884–893, Apr. 2010.

[10] M. A. Little, P. E. McSharry, E. J. Hunter, J. Spielman, and L. O. Ramig, “Suitability of dysphonia measurements for telemonitoring of Parkinson’s disease,” *IEEE Trans. Biomed. Eng.*, vol. 56, no. 4, pp. 1015–1022, 2009.

[11] B. Harel, M. Cannizzaro, and P. J. Snyder, “Variability in fundamental frequency during speech in prodromal and incipient Parkinson’s disease: A longitudinal case study,” *Brain Cogn.*, vol. 56, no. 1, pp. 24–29, 2004.

[12] C. G. Goetz *et al.*, “TESTING OBJECTIVE MEASURES OF MOTOR IMPAIRMENT IN EARLY PARKINSON’S DISEASE: FEASIBILITY STUDY OF AN AT-HOME TESTING DEVICE,” *Mov Disord*, vol. 24, no. 4, pp. 551–556, 2009.

[13] P. L. S. Chan and N. H. G. Holford, “Drug Treatment Effects on Disease Progression,” *Annu. Rev. Pharmacol. Toxicol.*, vol. 41, no. 1, pp. 625–659, 2001.

[14] W. M. M Schüpbach *et al.*, “Segmental progression of early untreated Parkinson’s disease: a novel approach to clinical rating,” 2009.

[15] A. Tsanas, M. A. Little, and O. Patrick E. McSharry Lorraine Ramig, “Enhanced Classical Dysphonia measures and sparse Regression for Tele Monitoring of Parkinson Disease Progression,” *Int. Conf. Acoust. Speech Signal Process.*, no. March, pp. 594–597, 2010.

[16] Ö. Eskidere, F. Ertaş, and C. Hanilçi, “A comparison of regression methods for remote tracking of Parkinson’s disease progression,” *Expert Syst. Appl.*, vol. 39, no. 5, pp. 5523–5528, 2012.

[17] J. Hlavica, M. Prauzek, T. Peterek, and P. Musilek, “Assessment of Parkinson’s disease progression using neural network and ANFIS models,” *Neural Netw. World*, vol. 26, no. 2, pp. 111–128, 2016.

[18] F. Pedregosa *et al.*, “Scikit-learn: Machine Learning in Python,” *J. Mach. Learn. Res.*, 2012.

[19] R. H. R. Hahnioser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas, and H. S. Seung, “Digital selection and analogue amplification coexist in a cortex- inspired silicon circuit,” *Nature*, vol. 405, no. 6789, pp. 947–951, Jun. 2000.

[20] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” Dec. 2014.

[21] J. Nocedal, “Updating quasi-Newton matrices with limited storage,” *Math. Comput.*, vol. 35, no. 151, pp. 773–773, 1980.

[22] F.-A. Fortin, U. Marc-André Gardner, M. Parizeau, and C. Gagné, “DEAP: Evolutionary Algorithms Made Easy François-Michel De Rainville,” 2012.