On the use of Hinge Loss as a Surrogate Fitness Function with Grammatical Evolution for Parkinson's Disease Classification

Jiajun Duan
jiajun.duan@ucdconnect.ie
Natural Computing Research And Applications Group
University College Dublin
Dublin, Ireland

Miguel Nicolau
miguel.nicolau@ucd.ie
Natural Computing Research And Applications Group
University College Dublin
Dublin, Ireland

Michael O'Neill
m.oneill@ucd.ie
Natural Computing Research And Applications Group
University College Dublin
Dublin, Ireland

Abstract

Parkinson's Disease (PD) is a progressive neurodegenerative disorder of the nervous system with a high rate of misdiagnosis. In this paper, we propose a Grammatical Evolution (GE) approach for classifying PD patients, that uses hinge loss as a surrogate fitness function. We compare our approach to standard GE using accuracy as its fitness function. Our results demonstrate that the surrogate fitness approach consistently produces models with better accuracy and reduced complexity. These results highlight the potential of using our approach as a powerful tool for developing trustworthy AI applications in the medical domain.

CCS Concepts

• **Computing methodologies** → **Genetic programming**.

Keywords

Surrogate Method, Grammatical Evolution, Parkinson's Disease, Symbolic Regression

ACM Reference Format:
Jiajun Duan, Miguel Nicolau, and Michael O'Neill. 2025. On the use of Hinge Loss as a Surrogate Fitness Function with Grammatical Evolution for Parkinson's Disease Classification. In *Genetic and Evolutionary Computation Conference (GECCO '25 Companion), July 14–18, 2025, Malaga, Spain.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3712255.3734348

1 Introduction

Parkinson's Disease (PD) is an often misdiagnosed neurodegenerative disorder of the nervous system, due to the similarity of symptoms with other neurological and movement disorders, and the subtle onset of such symptoms. In the clinical diagnosis of PD, up to 15% of patients are misdiagnosed, with the misclassification rate of non-specialists being even higher [3]. This suggests that doctors, especially non-specialists, may need other tools to assist them in clinical diagnosis.

Previous research explored different features and methods to detect PD patients. In clinical diagnosis, the first step is to detect if the patient has the presence of bradykinesia combined with either rest tremor, rigidity, or both [3]. This means that people with PD may move slightly slower than healthy people, and could have tremors or rigidity in their bodies. Other approaches for PD detection include comparing handwriting samples [4], walking gait [15], or speech analysis [16], as approximately 90% of PD patients suffer vocal impairment [24]. To build a system that helps detect PD patients, we frame the task as a binary classification problem, where Evolutionary Computation (EC) can be applied to generate interpretable models.

Genetic Programming (GP) is an EC method that can be used to generate interpretable models, and researchers show a strong interest in using it to solve real-world problems. However, when applying GP to binary classification, using accuracy as the fitness function of models provides a discrete fitness landscape, which is challenging to navigate. Our approach is to employ a grammar-based GP system, Grammatical Evolution (GE), using Hinge Loss as a surrogate fitness function. This type of approach is sometimes used with EC to enhance efficiency, improve performance or reduce computational cost [12].

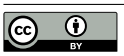
We compared both approaches (using accuracy versus using Hinge Loss) under a large set of experimental configurations, using the Gait in Parkinson's Disease dataset [8]. Our results show that using the loss function consistently produces models with better performance and/or reduced complexity. Analysis of the evolutionary process shows that this is due to the much better evolvability provided by the loss function.

The structure of this paper is as follows. In Section 2, we review the advantages of using surrogate fitness functions in GP. In Section 3, we introduce the accuracy and hinge loss fitness functions. In Section 4, we introduce our approach, including the dataset and the experimental setup. In Section 5, we analyse the results obtained, discussing the model performance, complexity, and evolvability. Finally, in the last Section, we summarise the results and outline potential directions for future research.

2 Surrogate Fitness Measures for GP

Surrogate methods can be used with GP for several reasons: reducing computational cost [12], improving performance [9], and reducing model complexity [29].

In evolutionary algorithms, the fitness function is often the most time-consuming process, especially when it involves complex data or individuals [12]. Previous studies have demonstrated that rather than calculating the fitness of each solution, we can estimate it instead, using surrogate methods, resulting in a similar performance in a fraction of the original time [13, 17, 29]. This technique is particularly useful in the case of limited computational resources or large-scale problems.



GECCO '25 Companion, July 14–18, 2025, Malaga, Spain

ACM ISBN 979-8-4007-1464-1/2025/07
https://doi.org/10.1145/3712255.3734348

Furthermore, surrogate approaches can enhance model performance by improving the efficiency of exploration of the search space. In symbolic regression, using a surrogate fitness function has been proven to yield better performance even in the presence of noise [9, 27]. This improvement is primarily due to the ability of surrogate functions to smooth fitness landscapes and enable the discovery of better-performing solutions over time. This also tends to improve the generalisation of the resulting models and may reduce overfitting.

Moreover, employing surrogate fitness functions can improve the interpretability of the models generated by GP. When a surrogate function is used, GP models tend to evolve simpler solutions compared to traditional fitness functions [29], and also tend to evolve more compact rules in a dynamic scheduling problem [25]. This tendency towards simplicity and compactness can lead to the generation of more interpretable models. In medical diagnosis or clinical diagnosis applications, transparent models are essential, which can help doctors understand why models give some suggestions, therefore, the ability to generate interpretable models is valuable.

3 Accuracy and Hinge Loss

When using evolutionary algorithms in binary classification problems, we typically define a threshold [19], and then compare the resulting model value and the threshold value. If the model value is bigger than the threshold value, then we predict the positive class, otherwise, we predict the negative class.

Accuracy (ACC) is broadly used as a fitness function in GP classification problems (e.g. [1]). ACC is measured as follows:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

where TP is a True Positive (correct positive class prediction), TN is a True Negative (correct negative class prediction), FP is a False Positive (incorrect positive class prediction), and FN is a False Negative (incorrect negative class prediction).

If the dataset is class unbalanced, as is the case with many medical datasets, then ACC may not be a reliable measure, and alternative fitness functions can be used, based on the distance of predicted value from actual value [5, 14, 22].

One such measure is the Hinge Loss (HL) function, which we use in our experiments. HL has been shown to achieve a better result than square loss in machine learning classification problems [23], and also can get a better result in autism spectrum disorder distinction than cross-entropy [6]. The HL is measured as follows:

$$L(y) = \max(0, 1 - y \times z) \tag{2}$$

where y is the actual class to predict ($y \in \{-1, 1\}$), and z is the output model value ($z \in (-\infty, \infty)$).

The output value of the HL function according to the value of z is shown in Figure 1, for the positive class case. It shows that if the incorrect label was predicted ($z <= 0$), the loss function returns a value in the range $[1, \infty)$, depending on the scale of the mis-prediction. However, if the correct label was predicted, an error in the range $[0, 1)$ can still be returned, which implies that the HL function can be used to increase the distance between the closer points to the decision boundary, as opposed to the ACC function, which merely counts correct vs. incorrect predictions.

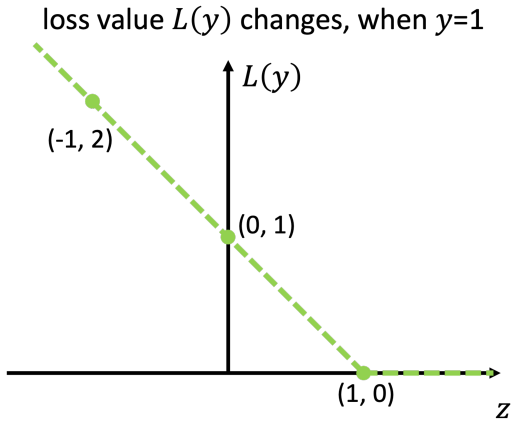


Figure 1: The loss function value changes with the predicted value for the target y equal to 1; the x-axis is the predicted value of the model z and the y-axis is the value of the loss function $L(y)$; a vertically mirrored plot can be observed when $y = -1$.

The use of the HL function can be beneficial when evaluating models for disease detection, by increasing the distance of positive and negative samples from the decision boundary, as used in Alzheimer's disease detection to construct more robust boundaries [10]. An example has been provided to discuss the difference between ACC and HL, and is shown in Figure 2. It represents a hypothetical binary classification problem, with 34 points for each class, and with two lines representing two different classification boundaries. Both lines can classify these points well (ACC is 100% for both lines), but the classification boundary of line 2 lies closer to the data points, and is both more complex, and more likely to lead to overfitting. The robustness of both lines can be assessed using HL (line 1 has an error value of 0.005, whereas line 2 has an error value of 0.059).

The use of the HL function can improve disease detection accuracy and reduce the possibility of misclassification. The trustworthiness of models can lead to further improvement by increasing the robustness of the decision boundary.

4 Experiments

4.1 Dataset

The dataset we used is Gait in Parkinson's Disease [8], in which vertical ground reaction forces were recorded from walking subjects. This pressure data was obtained from 16 sensors, with eight sensors on each foot (see their distribution in Figure 3), and the data were recorded at 100 samples per second. Alongside the data from the 16 sensors, the dataset also contains the total force under the left and right foot respectively. The dataset contains data from 166 people, 93 of whom are PD patients.

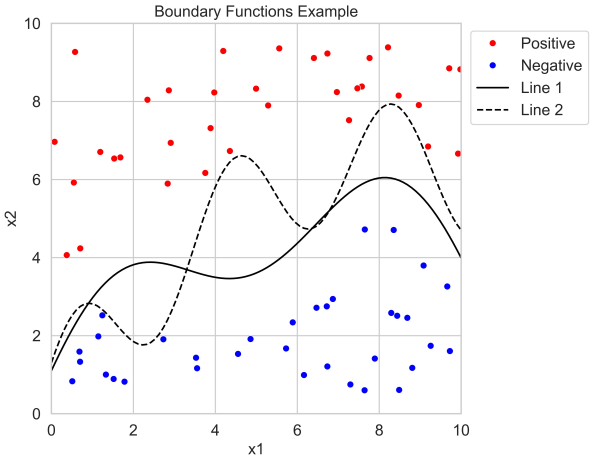


Figure 2: Comparing HL and ACC metrics in the boundary example.

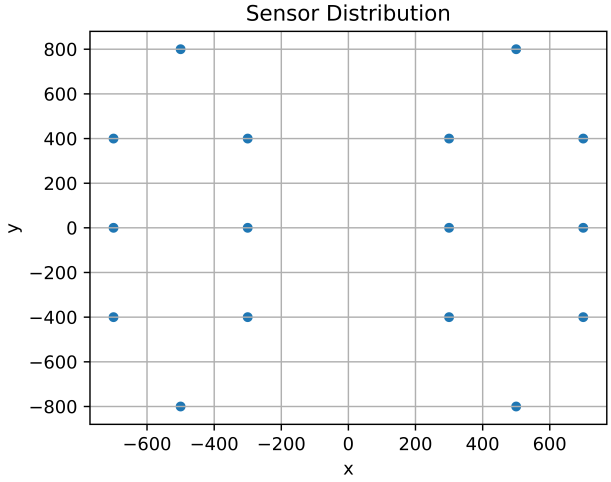


Figure 3: Sensor distribution on the left and right foot; coordinate origin is the centre of the human body.

We extracted features based on existing research [28]. For each person, we divided the pressure data by their weight to normalise it; there were three people (two with PD and one healthy) without weight data, therefore, we removed them from the data. We segmented the data into intervals of 40 seconds, and removed the first and last second, resulting in 38 seconds of data in each data item. For each item, we set a time window to extract features that contain 5 seconds of data, with 3.5 seconds of data overlapping for two adjacent windows. We extracted 45 features for each time window, and the descriptions for each feature are shown in Table 1. After this process, there were 2728 samples for PD patients and 1991 samples for healthy observations. Finally, we randomly dropped some samples from PD patients to ensure class balance.

The original dataset was split using a subject-dependent approach for several reasons. First, several prior studies working with this dataset adopted the same approach [18, 26, 28]. Second, gait pressure presents individual characteristics [21], which could be more challenging in subject-independent. Finally, the limitation samples of the original dataset.

We allocated 25% for the testing dataset (994 items), and 75% for the training dataset. For the training dataset, we split 20% for the validation dataset (596 items), and assigned the remaining data to the training dataset (2392 items). We used the training dataset to train our model, the validation dataset for model selection, and the testing dataset to test model performance.

4.2 Grammatical Evolution Setup

In this study, we used Grammatical Evolution [20], a grammar-based variant of GP, where the syntax of solutions is specified using a Backus-Naur Form grammar. We used a customised version of the PonyGE2 library [7], evolving models with subtree genetic operators (crossover and mutation) applied to the derivation trees.

We set the fitness function to HL or ACC separately, and the rest of hyperparameters are shown in Table 2. We used the maximum tree depth limit to control model performance and complexity, so we varied it from 8 to 90. We ran each maximum tree depth configuration 50 times.

4.2.1 Grammar. In our experiment, the grammar used is shown in Figure 4. In order to improve the interpretability of the resulting model, the arithmetic operations in our grammar only include addition, subtraction, multiplication, and protected division. <e> is the root node, and GE will begin at this node to select an operation structure, variable, or constant. The <v> symbol maps to the 45 features used in the PD dataset. Finally, <c> represents constants to use in the model (eight values between -1 and 1).

```
<e> ::= (<e> + <e>) | (<e> - <e>) | (<e> * <e>)
        | pdiv(<e>, <e>) | <v> | <v> | <v> | <c>
        | <c> | <c>
<v> ::= x[:, 0] | x[:, 1] | x[:, 2] | x[:, 3] | x[:, 4]
        | x[:, 5] | x[:, 6] | x[:, 7] | x[:, 8] | x[:, 9]
        | x[:, 10] | x[:, 11] | x[:, 12] | x[:, 13]
        | x[:, 14] | x[:, 15] | x[:, 16] | x[:, 17]
        | x[:, 18] | x[:, 19] | x[:, 20] | x[:, 21]
        | x[:, 22] | x[:, 23] | x[:, 24] | x[:, 25]
        | x[:, 26] | x[:, 27] | x[:, 28] | x[:, 29]
        | x[:, 30] | x[:, 31] | x[:, 32] | x[:, 33]
        | x[:, 34] | x[:, 35] | x[:, 36] | x[:, 37]
        | x[:, 38] | x[:, 39] | x[:, 40] | x[:, 41]
        | x[:, 42] | x[:, 43] | x[:, 44]
<c> ::= -1.0 | -0.1 | -0.01 | -0.001 | 0.001 | 0.01
        | 0.1 | 1.0
```

Figure 4: Grammar

Table 1: Depiction of 45 features

Feature Type	Description	Number of Features
Average Pressure	average pressure for each data in this time window	18
Coefficient of Variation	the ratio of the standard deviation to the mean for each data in this time window	18
Asymmetry Index	the asymmetry index can be represented by: $\left\|F_{left} - F_{right}\right\| / F_{left}$, where F_{left} denotes the average pressures for the left foot and F_{right} denotes the average pressures for the right foot	9

Table 2: All hyperparameters used in the PonyGE2 library

hyperparameter	value	hyperparameter	value
crossover	subtree	mutate_duplicates	True
crossover_probability	0.75	no_crossover_invalids	True
initialisation	rhh	no_mutation_invalids	True
invalid_selection	False	population_size	1000
min_init_tree_depth	3	replacement	generational
max_init_tree_depth	5	selection	tournament
max_tree_nodes	None	tournament_size	10
mutation	subtree	elite_size	10
mutation_events	1	generations	500
max_tree_depth	8-90		

4.2.2 Model Selection Strategy. Traditionally in GE, the best training model after 50 generations is typically chosen as the best model. However, given the high dimensionality of the PD dataset (45 features), a longer evolutionary process is required to discover accurate models. As a result, we extended the number of generations to 500. A downside of extending the evolutionary process for much longer is that it tends to increase the complexity of the models, or lead to overfitting. In order to control this, we used the validation dataset. At the end of each generation, we tested the best training fitness model using the validation dataset through accuracy, whether the fitness uses ACC or HL, and after 500 generations, we selected the model that obtained the highest ACC on the validation dataset. We do this to potentially increase the testing performance of the final model, as well as to reduce its complexity (if it is chosen from an earlier generation).

5 Result & Analysis

5.1 Performance

To evaluate the performance of ACC-evolved and HL-evolved models, we measure the final training and test accuracy of the best model from each run, using the ACC function. The results for all tree depths are shown in Figure 5.

The plot shows that at tree depth 8, the HL method exhibits similar training performance to the ACC-evolved model, and better testing performance. From depth 10 onwards, this is even more evident, both for training and test performance. The difference in performance of the resulting models is quite drastic. HL-evolved models always exhibit better performance than ACC-evolved models of any depth.

The accuracy of ACC-evolved models appears to hit a plateau for experimental setups with depth greater than 15, with little accuracy improvements beyond this depth limit. Similarly, for the HL-evolved models, their performance plateaus, but at a much larger depth limit (i.e., for setups with a depth limit of 40 and greater). This is probably due to the vast increase in the search space, but it could also be due to the evolvability of the fitness functions used, as discussed in Section 5.4.

We tested the statistical significance of these results for all tree depths. As the performance distribution across runs cannot be assumed normal (see Section 5.2), we used the non-parametric Brunner-Munzel test, which does not require the assumption of equal variances to measure location shift. The resulting p-values are shown in Table 3. They clearly show that, for every tree depth, the HL-evolved results are always statistically significantly better ($p < 5\%$) than those from ACC-evolved runs, both for training and test.

5.2 Consistency

Method performance consistency means that if we train models using the same experimental setup, the resulting models should have a similar performance. In the case of heuristic methods such as EC approaches, if we only change the random seed of an experiment, the resulting model should ideally perform similarly.

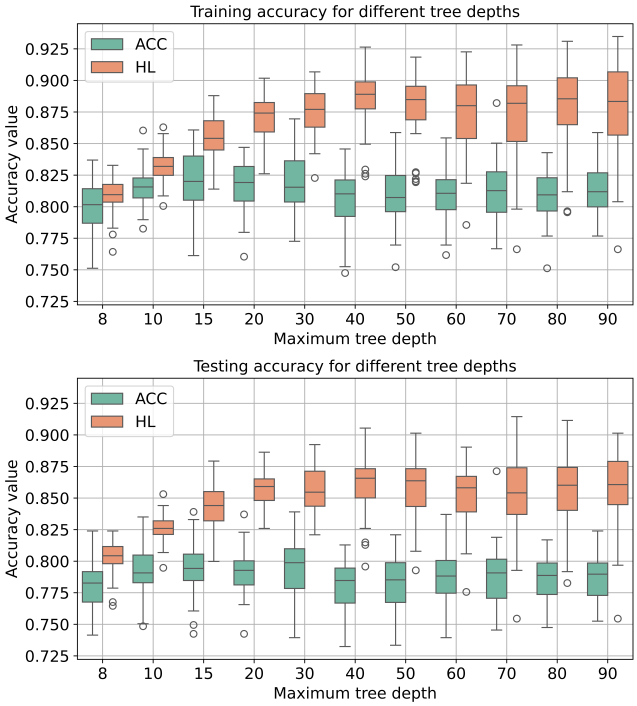


Figure 5: Performance results from different maximum tree depths, for ACC- and HL-evolved models. The y-axis shows the accuracy of the final models, as measured with the ACC function. Each tree depth shows the results from 50 independent runs.

Table 3: Brunner-Munzel Test Result

Tree Depth	Training	Testing
8	**0.007306**	**3.240e-14**
10	**1.361e-13**	**3.528e-32**
15	**2.103e-21**	**1.618e-50**
20	**2.664e-54**	**5.095e-71**
30	**9.001e-59**	**3.873e-84**
40	**5.720e-79**	**1.016e-64**
50	**7.643e-44**	**4.242e-59**
60	**4.699e-33**	**1.404e-37**
70	**2.003e-21**	**1.051e-29**
80	**3.976e-24**	**1.643e-34**
90	**6.949e-21**	**2.640e-28**

To measure this, we use the Shapiro-Wilk test to measure the normality of (train and test) model performances from the 50 different runs of each experimental setup. If the resulting model performances are approximately normally distributed, then it means that the method is reliable, and is not overly dependent on the random seed used. Otherwise, several runs are required to choose more accurate models, leading to random seeds being an actual hyperparameter for the methodology.

The run consistency measurements are shown in Table 4, for all tree depths, with p-values lower than 5% (shown in bold) indicating a strongly normal distribution. We can see that for nearly all tree depths, ACC-evolved runs are not normal distributed, whereas approximatively half of the HL-evolved runs follow a normal distribution.

Table 4: Shapiro-Wilk Test Result

	Training		Testing	
Tree Depth	ACC	HL	ACC	HL
8	0.814	**0.017**	0.636	**0.004**
10	0.111	0.951	0.835	0.610
15	0.508	0.889	0.462	0.883
20	0.053	0.486	0.716	**0.033**
30	0.546	0.341	0.581	0.407
40	0.091	**0.001**	0.079	0.293
50	0.989	**0.001**	0.295	0.070
60	0.709	0.089	0.519	0.020
70	0.349	**0.012**	**0.006**	0.311
80	0.249	**0.006**	**0.006**	0.210
90	0.500	**0.015**	0.301	**0.004**

5.3 Complexity

We estimate the complexity of models by simplifying their symbolic representation, and then counting the number of symbols in the resulting equation, we use SymPy library to achieve this function. For example, if the resulting model is $x0 - x0 + x1 + 1 + 1$, then the simplified model is $x1 + 2$, which means that its simplified size is 3.

To compare the size of the resulting models for both approaches, we plot the distribution of simplified sizes for each tree depth; this is shown in Figure 6. The plot shows that for small tree depths, the sizes of ACC- and HL-evolved models are comparable. As the tree depth increases, the complexity of the resulting models increases for both approaches, with HL-evolved models becoming much more complex for large tree depths (the median size of ACC-evolved models for tree depth 90 is 245.5, whereas that of HL-evolved models is 1603).

Given the results from Figure 5, one could claim that there is a clear performance/complexity trade-off between the ACC and HL approaches. To test this assessment, we plotted the size and performance of every single run, for both approaches, across all tree depths; this is shown in Figure 7. These results are not grouped by tree depths, meaning that two models of size 500 could come from different tree depths.

We can see that as model complexity increases, the performance of models also increases, up to an approximate size of 500 symbols, after which more complex models do not result in better performance. Much more importantly, we can clearly observe that, for similar size models (which can come from different tree depths), HL-evolved models are nearly always more accurate than ACC-evolved models. Conversely, if we focus on same model performance, then we can clearly see that HL-evolved models are less complex than ACC-evolved models. So using HL as a surrogate fitness function leads to both better performance, and more interpretable models.

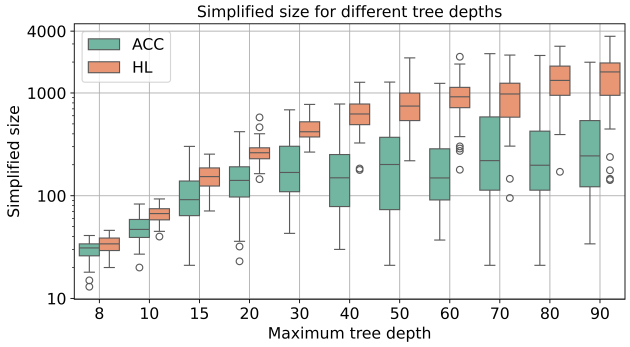


Figure 6: Simplified size distributions across 50 runs, for all tree depths. Due to the large differences across tree depths, the y-axis uses a logarithmic scale.

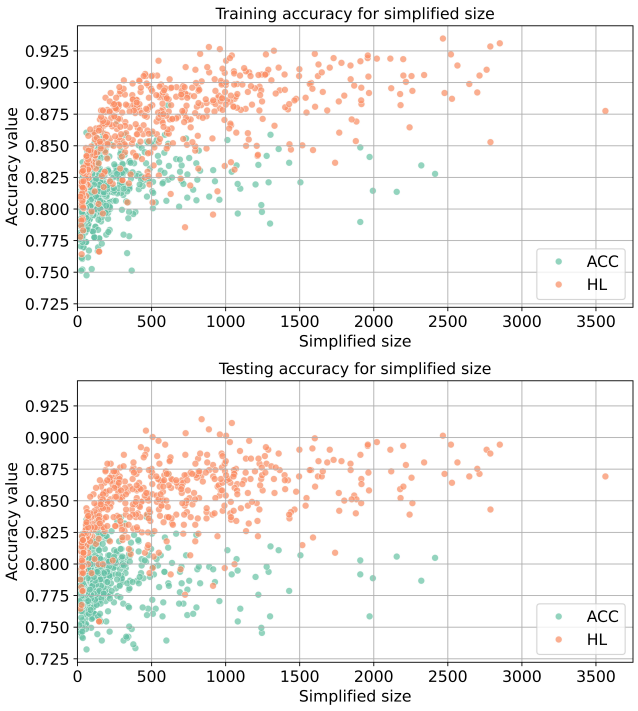


Figure 7: Model size and corresponding performance for training and testing data. There are a total of 1100 points on each plot, 550 for each approach, which are the best models from each run (50 runs total) from each depth (11 depths in total).

5.4 Evolvability

The evolvability of solutions in GP is one of the key factors that can influence the resulting model performance. In this paper, we use two simple ways to analyse evolvability. First, we assume that if more complex models are evolved, then a corresponding increase in performance should be observed. As seen in Section 5.3, this is much more the case when using HL as a fitness function, when compared to using ACC as the fitness function.

Another way to measure evolvability is to analyse the number of fitness improvements across each run, for both approaches. To do this, for each run, we counted the number of generations where there was a fitness improvement, we called this "jump count". For ACC-evolved models, this means an increase of the number of correctly-classified points; for HL-evolved models, this means a reduction of the Hinge Loss error value. This is shown in Figure 8, for all tree depths.

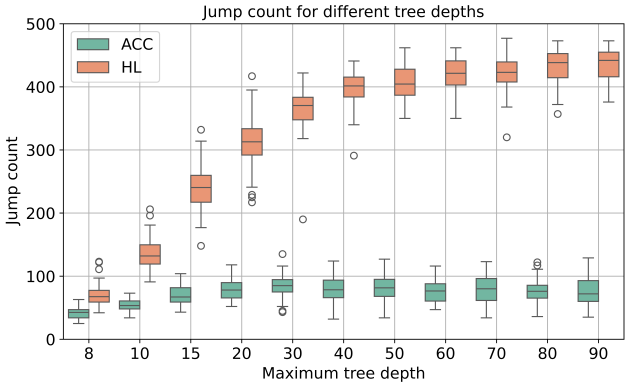


Figure 8: Number of fitness improvements (jump count) for all tree depths, for ACC- and HL-evolved models. Each tree depth shows the results from 50 independent runs.

We can clearly see that HL runs have much higher jump counts than ACC runs, across all tree depths. Also, as the tree depth increases, HL runs see an increase in number of fitness improvements, all the way to depth 40, where more than 80% of all generations create better performing models. This is in contrast to ACC runs, where less than 20% of generations create better models, with a slight increase up to tree depth 15. This suggests that the HL function can use the expanding search space more effectively than the ACC function, with the latter showing limited evolvability in this problem.

This also helps explain why the performance of HL-evolved models stop increasing after tree depth 40, and tree depth 15 for ACC-evolved models, as discussed in Section 5.1. For HL, there are probably not enough generations to continue exploring the search space, as the fitness function nearly changes at every generation; for ACC, this is likely because of the limitation of the evolvability of the fitness function.

6 Conclusion

This paper explored the use of a surrogate fitness measure to evolve models for the Parkinson's Disease patient classification problem. Our results highlight the applicability of the hinge loss function as a suitable surrogate function in GE, resulting in better performing and more interpretable models, when compared to using a standard accuracy fitness measure.

Our analysis shows that using hinge loss provides advantages at several levels. First, the resulting models are more accurate than

when using accuracy as the driving fitness function, even though in the latter case we use the same measure to train and evaluate final performance. Secondly, statistical testing showed that using hinge loss produces more consistent results across runs. Thirdly, models of similar complexity have better performance, and conversely, models with the same performance have smaller complexity. And finally, we analysed the evolvability of the hinge loss function, which helps explain the advantages mentioned.

While these results are encouraging, they relate to a single dataset, so in future work, we will apply a similar approach to other medical problems. We will also investigate the effect of our model selection approach on datasets and/or experimental setups that are more prone to over-fitting. We know there are multiple approaches to measuring evolvability, for instance, Price's theorem is used to extract the change in the distribution of fitness values [2] or nonsynonymous to synonymous substitution ratio k_a/k_s is used to measure the rate of genetic substitutions in tree-based GP [11], we will analysis the different their method and our method.

Finally, we currently apply our model selection approach at the end of the training process, meaning that there is no reduction of the training effort; we will investigate possible ways to speed up the model building with validation strategies, and how to integrate them into our experimental framework.

Acknowledgments

This publication has emanated from research conducted with the financial support of Taighde Éireann - Research Ireland under Grant number 21/FFP-A/9255.

References

[1] Hadeel Abdulrahman and Mohamed Khatib. 2020. Classification of retina diseases from OCT using genetic programming. *Int. J. Comput. Appl* 177, 45 (2020), 41–46.

[2] Lee Altenberg et al. 1994. The evolution of evolvability in genetic programming. *Advances in genetic programming* 3 (1994), 47–74.

[3] Bastiaan R Bloem, Michael S Okun, and Christine Klein. 2021. Parkinson's disease. *The Lancet* 397, 10291 (2021), 2284–2303. doi:10.1016/S0140-6736(21)00218-X

[4] F. Cavaliere, A. Della Cioppa, A. Marcelli, A. Parziale, and R. Senatore. 2020. Parkinson's Disease Diagnosis: Towards Grammar-based Explainable Artificial Intelligence. In *2020 IEEE Symposium on Computers and Communications (ISCC)*. 1–6. doi:10.1109/ISCC50000.2020.9219616

[5] Divyaansh Devarriya, Cairo Gulati, Vidhi Mansharamani, Aditi Sakalle, and Arpit Bhardwaj. 2020. Unbalanced breast cancer data classification using novel fitness functions in genetic programming. *Expert Systems with Applications* 140 (2020), 112866.

[6] Thomas Martial Epalle, Yuqing Song, Zhe Liu, and Hu Lu. 2021. Multi-atlas classification of autism spectrum disorder with hinge loss trained deep architectures: ABIDE I results. *Applied Soft Computing* 107 (2021), 107375. doi:10.1016/j.asoc.2021.107375

[7] Michael Fenton, James McDermott, David Fagan, Stefan Forstenlechner, Erik Hemberg, and Michael O'Neill. 2017. Ponyge2: Grammatical evolution in python. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 1194–1201. doi:10.1145/3067695.3082469

[8] Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. 2000. PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals. *circulation* 101, 23 (2000), e215–e220. doi:10.1161/01.CIR.101.23.e215

[9] Nathan Haut, Wolfgang Banzhaf, and Bill Punch. 2023. Correlation versus rmse loss functions in symbolic regression tasks. In *Genetic Programming Theory and Practice XIX*. Springer, 31–55.

[10] Chris Hinrichs, Vikas Singh, Guofan Xu, and Sterling Johnson. 2009. MKL for robust multi-modality AD classification. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2009: 12th International Conference, London, UK, September 20-24, 2009, Proceedings, Part II 12*. Springer, 786–794. doi:10.1007/978-3-642-04271-3_95

[11] Ting Hu and Wolfgang Banzhaf. 2009. Neutrality and variability: two sides of evolvability in linear genetic programming. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*. 963–970.

[12] Yaochu Jin. 2011. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation* 1, 2 (2011), 61–70.

[13] Ahmed Kattan, Alexandros Agapitos, Yew-Soon Ong, Ateq A Alghamedi, and Michael O'Neill. 2016. GP made faster with semantic surrogate modelling. *Information Sciences* 355 (2016), 169–185.

[14] Arvind Kumar, Nishant Sinha, and Arpit Bhardwaj. 2020. A novel fitness function in genetic programming for medical data classification. *Journal of Biomedical Informatics* 112 (2020), 103623.

[15] Stuart E Lacy, Michael A Lones, Stephen L Smith, Jane E Alty, DR Stuart Jamieson, Katherine L Possin, and Norbert Schuff. 2013. Characterisation of movement disorder in Parkinson's disease using evolutionary algorithms. In *Proceedings of the 15th annual conference companion on Genetic and evolutionary computation*. 1479–1486.

[16] Max Little, Patrick McSharry, Eric Hunter, Jennifer Spielman, and Lorraine Ramig. 2008. Suitability of dysphonia measurements for telemonitoring of Parkinson's disease. *Nature Precedings* (2008), 1–1.

[17] Ilya Loshchilov, Marc Schoenauer, and Michele Sebag. 2012. Self-adaptive surrogate-assisted covariance matrix adaptation evolution strategy *(GECCO '12)*. Association for Computing Machinery, New York, NY, USA, 321–328. doi:10.1145/2330163.2330210

[18] Laura-Ioana Mihăilă, Claudia-Georgiana Cordoş, Robert Radu Ilcşan, Paul Faragó, and Sorin Hintea. 2022. CNN-based Identification of Parkinsonian Gait using Ground Reaction Forces. In *2022 45th International Conference on Telecommunications and Signal Processing (TSP)*. 318–321. doi:10.1109/TSP55681.2022.9851260

[19] Thambo Nyathi and Nelishia Pillay. 2018. Comparison of a genetic algorithm to grammatical evolution for automated design of genetic programming classification algorithms. *Expert Systems with Applications* 104 (2018), 213–234. doi:10.1016/j.eswa.2018.03.030

[20] Michael O'Neil and Conor Ryan. 2003. *Grammatical Evolution*. Springer US, Boston, MA, 33–47. doi:10.1007/978-1-4615-0447-4_4

[21] Todd C Pataky, Tingting Mu, Kerstin Bosch, Dieter Rosenbaum, and John Y Goulermas. 2012. Gait recognition: highly unique dynamic plantar pressure patterns among 104 individuals. *Journal of The Royal Society Interface* 9, 69 (2012), 790–800.

[22] Wenbin Pei, Bing Xue, Lin Shang, and Mengjie Zhang. 2019. New Fitness Functions in Genetic Programming for Classification with High-dimensional Unbalanced Data. In *2019 IEEE Congress on Evolutionary Computation (CEC)*. 2779–2786. doi:10.1109/CEC.2019.8789974

[23] Lorenzo Rosasco, Ernesto De Vito, Andrea Caponnetto, Michele Piana, and Alessandro Verri. 2004. Are loss functions all the same? *Neural computation* 16, 5 (2004), 1063–1076. doi:10.1162/089976604773135104

[24] C Okan Sakar and Olcay Kursun. 2010. Telediagnosis of Parkinson's disease using measurements of dysphonia. *Journal of medical systems* 34 (2010), 591–599.

[25] Adilanmu Sitahong, Yiping Yuan, Ming Li, Junyan Ma, Zhiyong Ba, and Yongxin Lu. 2022. Designing dispatching rules via novel genetic programming with feature selection in dynamic job-shop scheduling. *Processes* 11, 1 (2022), 65.

[26] Dixon Vimalajeewa, Ethan McDonald, Megan Tung, and Brani Vidakovic. 2023. Parkinson's Disease Diagnosis With Gait Characteristics Extracted Using Wavelet Transforms. *IEEE Journal of Translational Engineering in Health and Medicine* 11 (2023), 271–281. doi:10.1109/JTEHM.2023.3272796

[27] Kaifeng Yang and Michael Affenzeller. 2023. Surrogate-assisted multi-objective optimization via genetic programming based symbolic regression. In *International Conference on Evolutionary Multi-Criterion Optimization*. Springer, 176–190.

[28] Huan Zhao, Ruixue Wang, Yaguo Lei, Wei-Hsin Liao, Hongmei Cao, and Junyi Cao. 2022. Severity level diagnosis of Parkinson's disease by ensemble K-nearest neighbor under imbalanced data. *Expert Systems with Applications* 189 (2022), 116113. doi:10.1016/j.eswa.2021.116113

[29] Luyao Zhu, Fangfang Zhang, Xiaodong Zhu, Ke Chen, and Mengjie Zhang. 2023. Sample-aware surrogate-assisted genetic programming for scheduling heuristics learning in dynamic flexible job shop scheduling. In *Proceedings of the Genetic and Evolutionary Computation Conference*. Association for Computing Machinery, 384–392. doi:10.1145/3583131.3590440