NEAT Based Neural Network Architecture for California Housing Price Problem

Refik Can Öztaş
Computer Engineering
Hacettepe University
Ankara, Turkey
refikoztas@hacettepe.edu.tr

***Abstract*—California housing prices is a problem that has been tried to be solved by various methods. In order to achieve regression, although there has been various solutions offered, Artificial Neural Networks (ANNs) models seem to be working well on this problem. This paper suggests Genetic Algorithm (GA) based NeuroEvolution of Augmenting Topologies (NEAT) methods to find optimal Artificial Neural Network (ANN) in order to make this regression.**

Keywords — artificial neural networks; california-housing-price; neat; neuroevolution of augmenting topologies;

I. INTRODUCTION

In recent years, the demand for housing has been steadily increasing, leading to an increasing demand for property development. As a result, housing prices have been subject to fluctuations. One approach to understanding these fluctuations is to utilize machine learning techniques, specifically regression algorithms. The California Housing Dataset provides an excellent platform for conducting such investigations.

This paper suggests a NEAT based dynamic house price prediction model, that can be implemented with web scraping real-life data. Although, this research mostly focuses on NEAT based neural network generation and its performance evaluation, as a future work, there can be web-scraping for feeding dataset, or real-life inflation, geolocation and etc. data can be added easily to NEAT model, which change the final neural network model.

This dataset, comprising 20,640 samples and 9 attributes, serves as a comprehensive benchmark for evaluating the performance of regression algorithms in predicting housing prices. The attributes included in the dataset are the population of the region, the proportion of housing units built since 1960, the average housing value for each region, the average income for each region, the physical characteristics of each region, the age distribution of housing in the region, the latitude, and the longitude.

II. NEUROEVOLUTION OF AUGMENTING TOPOLOGIES FRAMEWORK

The field of neuroevolution and the evolution of neural network topologies has witnessed significant contributions from various research endeavors. In this section, I provide a comprehensive overview of the related work that has laid the groundwork for the development and understanding of the NEAT (NeuroEvolution of Augmenting Topologies) framework.

- Evolutionary Algorithms for Neural Networks

Early work in evolving neural network architectures using genetic algorithms set the stage for NEAT's development. Holland's seminal work on genetic algorithms [1] and Goldberg's exploration of genetic algorithms in optimization [2] paved the way for the incorporation of evolutionary principles in the realm of neural networks.

- Genetic Algorithms in Neural Network Optimization

Research efforts focusing on genetic algorithms for neural network optimization have been influential. Sareni and Krähenbühl's work on evolving neural network structures through genetic algorithms [3] demonstrated the potential of combining evolutionary algorithms with neural networks, inspiring the simultaneous evolution of structure and weights in NEAT.

- Neuroevolution Frameworks

The landscape of neuroevolution frameworks has expanded over the years. Notable frameworks include SharpNEAT [4], which shares commonalities with NEAT but is .NET-based, and ESP (Evolving Structure Preserving

Networks) [5], which focuses on evolving neural network structures. NEAT distinguishes itself by combining these principles in a Python-based framework with a user-friendly interface.

- Evolution of Complex Adaptive Systems

The concept of evolving complex adaptive systems has been explored in the broader context of artificial life and evolutionary computation. Holland's work on complex adaptive systems [6] has influenced the conceptualization of NEAT, treating neural networks as dynamic entities that adapt to environmental demands through structural innovations.

- Continuous Neuroevolution

NEAT has been extended and adapted to address challenges in continuous action and parameter spaces. Koutník et al.[7] proposed a continuous version of NEAT, showcasing the framework's adaptability to different problem domains. This extension broadens NEAT's applicability, allowing it to excel in scenarios with continuous and dynamic variables.

- Comparative Neuroevolution Studies

Comparative studies evaluating the performance of neuroevolution algorithms have played a crucial role in understanding their strengths and weaknesses. Stanley et al.'s comprehensive study comparing neuroevolution methods [8] positioned NEAT as a leading algorithm, highlighting its ability to discover effective neural network architectures across diverse tasks and problem domains.

In summary, NEAT builds upon a rich foundation of evolutionary algorithms, genetic algorithms, and neuroevolution frameworks. Its unique contribution lies in the simultaneous evolution of structure and weights, providing a powerful and flexible platform for automated design and adaptation of neural networks. The subsequent sections of this paper delve into the specific features and contributions of NEAT, demonstrating its efficacy through experimental results and applications.

III. MAIN IDEA OF PAPER

This paper focuses on the California Housing Price dataset, and as a future work it proposes dynamically generated neural networks with live feed data via NEAT. With more resources and time, problems can be widened into much more complex systems. If the proposed dynamically generated neural networks model is suitable, other problems such as complex stock exchange, currency, crypto-market regression problems can be also implemented with other live feeds. However, I believe California Housing Price is a good start for that kind of structure.

This paper does not implement real life data that will add to the NEAT in order to feed the Neural Network structure. It only implements the California Housing Dataset neural network structure.

IV. IMPORTANCE OF SOLUTION

NEAT is mostly used in neural networks that play games such as Flappy Bird, Tetris etc. Game inputs are dynamic, and it needs to be played in order to learn and find optimal neural network solutions. Most time-series based data such as stock exchange prices, currency rate over time, etc are similar to games. Dynamic, self-learning, auto generated neural network structure can be a solution of this kind of problem with live data just like game inputs. Based on the proposed architecture, much more complex models can be crafted.

In other NEAT applications, such as games, NEAT topology learns to play the game with trial and error. Game inputs can be the location of the character in 3D space, some time-based metrics such as weight and height of the object, and other dynamic inputs from the player or game itself. With a just like reinforcement-learning based decision making system, optimal neural network solutions can be crafted with punishment for the error and reward for the success.

Based on that approach, we can think of inputs of the California Housing dataset as game inputs. As mentioned in the future work direction, scraped dynamic prices from websites, economical metrics, and live-feed from social media like twitter can be dynamic input for the topology which will ultimately find an optimal neural network for solution.

V. MODEL ARCHITECTURE

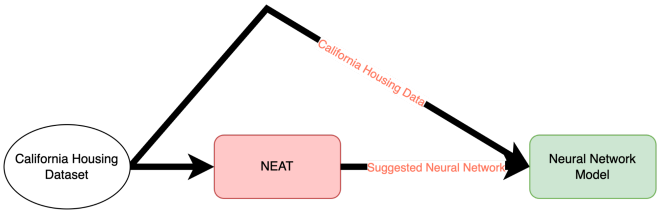

Fig. 1: NEAT based model architecture

The recommended architecture suggests the use of a dynamic neural network technique. During the first stage, an ideal neural network model that fits the dataset's properties is found using the NEAT (NeuroEvolution of Augmenting Topologies) algorithm. In order to improve performance, NEAT iteratively improves the network architecture by optimizing its parameters and structure.

A secondary neural network is created based on this revised design once the best model has been determined using NEAT. The California Housing dataset, with its variety of real-world housing-related variables, is then used to train this recently developed model. The neural network is adjusted during training so that it can adjust to the unique patterns and complexities found in the housing dataset.

The architecture can be summarized as follows: first, an optimal neural network structure is dynamically designed by the evolutionary NEAT algorithm; second, a neural network is instantiated based on this optimized architecture and further trained using the California Housing dataset for improved performance on real-world housing prediction tasks. This method makes use of dynamic neural network construction and evolutionary algorithms to ensure adaptation to the features of the dataset.

VI. METHODOLOGY

I. Preprocessing

We include min-max normalization as a crucial preprocessing step in this paper. In the field of data preprocessing, normalization is essential because it allows features in a dataset to be scaled and standardized at the same time. Normalization's principal goal is to level the playing field by making all feature scales uniform. By doing this, the possibility of some features overshadowing others just because of size disparities is reduced, guaranteeing an impartial and accurate portrayal of the data.

This study uses min-max normalization, a popular method for data normalization. Using this method, a feature's values are scaled and transformed to fall into a predetermined range, usually between 0 and 1. To accomplish this, take each data point and subtract the feature's minimum value. Then divide the result by the feature's range, which is the difference between the maximum and minimum values. By bringing all feature values into a uniform range, this normalization technique makes the learning process for developing new models more efficient and reliable.

My goal is to improve the overall performance of my neural network models by applying min-max normalization as a preprocessing step. This method guarantees that the models are trained on feature-scale data that is standardized, which improves training convergence and the model's capacity to generalize to new data. Within the framework of my suggested architecture, this preprocessing stage acts as a cornerstone, laying the groundwork for the later use of the NEAT algorithm and the dynamic creation of neural network architectures that are customized to the particular features of the California Housing dataset.

II. NEAT

The preprocessed input data is fed into the NEAT (NeuroEvolution of Augmenting Topologies) network when the preprocessing step is finished. NEAT uses a set of hyperparameters to direct the evolutionary process of developing neural network architectures. 150 as a population size in the configuration was selected.

As a default activation function Rectified Linear Unit (ReLU), a default activation function that is well-known for introducing non-linearity into the network's learning process, is used. A crucial factor affecting population variety is the mutation rate, which is chosen at 0.3 to permit a limited degree of structural alterations in the developing neural networks.

NEAT generates neural networks with 4 hidden layers by default, given the input size of 8 features. The network's ability to adjust to various input dimensions guided this decision, guaranteeing a well-balanced and efficient design for processing the California Housing dataset. The probability of adding new connections to the developing neural networks is determined by the connection add rate, which is fixed at 0.5. This makes it easier for the neural networks to identify

complex correlations in the dataset by promoting the investigation of different connectivity patterns.

Furthermore, NEAT includes a node add rate of 0.2, which affects the likelihood of adding new nodes (neurons) to the designs that are evolving. This dynamic modification improves the network's ability to model the underlying patterns in the housing dataset by enabling it to modify its structure by adding complexity as needed. The NEAT algorithm's combination of these hyperparameters guides the formation of neural network designs that are well-suited to the features of the California Housing dataset by ensuring a balance between exploration and exploitation during the evolutionary process. The goal is to use neuroevolution to find the best neural network architectures for training on the particular housing-related characteristics later on.

III. Neural Network based on NEAT

After executing the NEAT algorithm multiple times, it consistently converged towards a neural network topology characterized by a single hidden layer with Rectified Linear Unit (ReLU) activation. The discovered architectures typically comprised between 30 and 33 neurons in the hidden layer. Opting for a balanced approach, a specific configuration with 32 hidden-layer neurons was selected for the current iteration.

However, to enhance the adaptability and flexibility of the model, there is a plan to transition towards a fully dynamic architecture in future iterations. In this envisioned improvement, the neural network will possess the capability to dynamically adjust the number of hidden-layer neurons based on the intricacies and patterns inherent in the California Housing dataset. This dynamic adaptation aligns with the overarching goal of creating a model that can efficiently capture and represent the varying complexities within the dataset, optimizing its performance across different scenarios.

The transition to a completely dynamic hidden layer is a progressive improvement over the existing architecture. By taking into account the many subtleties and variances found in housing-related data, this method enables the model to automatically modify its complexity so that it more closely matches the underlying patterns. The goal is to create a more resilient and adaptable neural network that can enhance and adapt to different data examples and settings throughout time. This dynamic flexibility emphasizes the architecture's evolutionary character even more, which is consistent with NEAT's guiding principles and its ability to iteratively find the best neural network architectures.

IV. Results

I created training-testing and validation data. After training with training data, mean absolute percentage error is 61.7829

The absolute percentage error (APE) is a measure of the difference between the actual value and forecast value of a data point, expressed as a percentage of the actual value.

Since neural network structure is relatively-optimal for the problem, I tried to find how epoch size affects the absolute percentage error, and tried to find if there is a relation.

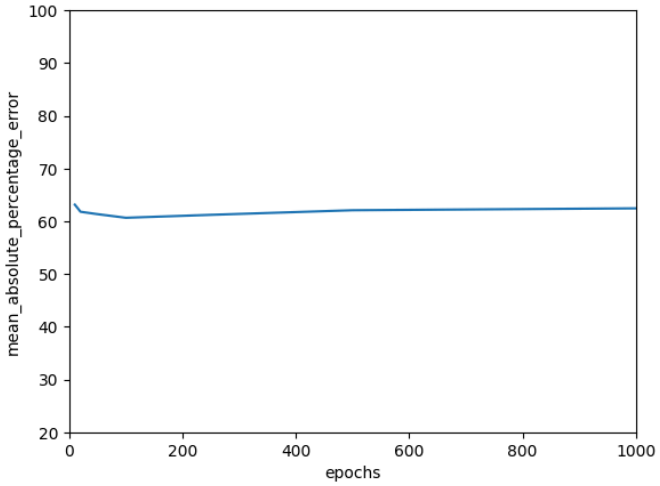

Fig. 2: epoch - error rate ratio

As we can see in Figure 2, as the epoch size increases the error percentage does not change. Since NEAT tries to find an optimal neural network, the generated neural network structure does not require a lot of epoch in order to train.

VII. DISCUSSION

The NEAT structure's intrinsic randomness has presented difficulties during testing, making it difficult to find the best solution over many runs. Another issue that surfaced was variances in runtime, with some cases needing a long time to converge. Although the NEAT structure primarily employed default values, it is recognized that optimizing these parameters may result in better outcomes and more stable performance.

In this work, an unusual method of data preparation was used. Rather than following the traditional division of training and test data, the dataset experienced a special procedure. Training and test data, which are normally separated by default, were combined at first. A fully-randomized split was then used to produce a fresh configuration of training and test data. This divergence from traditional approaches adds experimentation while also admitting the possibility of more improvements or different approaches that could improve model performance.

Despite these difficulties and atypical preprocessing measures, the outcomes suggest that the dynamically produced neural network architecture demonstrates near-optimality. Interestingly, the mean absolute % error is resistant to changes in epoch size, indicating that the architecture may generalize without any trouble. This robustness may be ascribed to the neural network's flexibility and dynamic character, demonstrating the possibility of time-saving advantages in the development of neural networks suited for dynamic problem domains.

It is important to note that although the study offers insights into the possibilities of dynamically formed network architectures, a live-data fed version of the architecture is not implemented. The study, however, establishes the foundation for such a system by providing brief descriptions and recommending directions for further research into the integration of real-time data inputs.

VIII. FUTURE WORK ARCHITECTURE

Although this paper uses only California Housing data, as a future work, real life data from several housing price websites via web scraping, some economical metrics (for non U.S. country dollar rate) can feed the NEAT structure. With that live feed, other neural networks can be formed, in order to find optimal results for the problem.

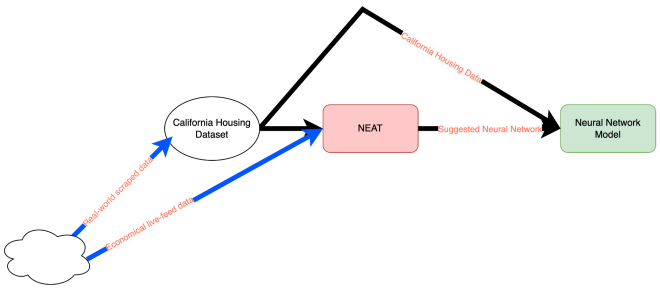

Fig. 3: NEAT based model architecture as future work suggestion

IX. CONCLUSION

In conclusion, this paper introduces a novel approach to dynamic house price prediction through a NEAT-based regression model applied to the California Housing Dataset. The primary focus lies in the generation and evaluation of neural networks using the NEAT algorithm, showcasing its effectiveness in evolving architectures for predicting housing prices. The dataset, comprising 20,640 samples with 9 attributes, serves as a robust benchmark for assessing regression algorithm performance.

Through meticulous preprocessing techniques, including min-max normalization, the study aims to enhance training convergence and model generalization. The NEAT network is configured with a population size of 150, demonstrating the algorithm's capacity to evolve neural network structures with a single hidden layer, ReLU activation, and 30-33 neurons. The mean absolute percentage error (APE) is employed as a performance metric, revealing that the NEAT-generated structures require fewer epochs for training, underscoring their efficiency.

While acknowledging challenges associated with the intrinsic randomness and runtime variances during testing, the paper highlights the near-optimality and generalizability of the dynamically created neural network structures. The study identifies potential avenues for future research, suggesting the incorporation of real-life data through web scraping from housing price websites. Additionally, fine-tuning parameters for dynamic problem-solving scenarios is emphasized as a crucial aspect of further exploration.

In summary, this research contributes to the evolving landscape of dynamic prediction models, showcasing the promise of NEAT-generated neural networks in housing price prediction. The presented methodology, results, and identified areas for improvement pave the way for future advancements in the realm of dynamic neural network architectures for real-world applications.

REFERENCES

[1] Holland, J. H. (1975). Adaptation in Natural and Artificial Systems. University of Michigan Press.

[2] Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley.

[3] Sareni, B., & Krähenbühl, L. (1998). Fitness sharing and niching methods revisited. IEEE Transactions on Evolutionary Computation, 2(3), 97-106.

[4] SharpNEAT. (n.d.). Retrieved from https://www.sharpneat.com/

[5] ESP: Evolving Structure Preserving Networks. (n.d.). Retrieved from https://esp-nn.github.io/

[6] Holland, J. H. (2014). Complexity: A Very Short Introduction. Oxford University Press.

[7] Koutník, J., Gomez, F., & Schmidhuber, J. (2010). Evolving Neural Networks in Compressed Weight Space. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO).

[8] Stanley, K. O., Clune, J., Lehman, J., & Miikkulainen, R. (2009). Compositional Pattern Producing Networks: A Novel Abstraction of Development. Genetic Programming and Evolvable Machines, 11(2), 131-162.