Assignment 1, Deep Learning Fundamentals

Possakorn Kittipipatthanapong
The University of Adelaide
a1873765@adelaide.edu.au

September 29, 2023

Abstract

This assignment explores the application of Artificial Neural Networks (ANN), specifically perceptron algorithms, to predict diabetes using the Pima Indians Diabetes Database. The study encompasses the optimization of learning rates, selection of optimization algorithms, and adjustment of model complexity to enhance predictive accuracy and mitigate overfitting. Results suggest a multi-layer perceptron with one hidden layer and 32 nodes as optimal, achieving 74.5% accuracy without overfitting.

1 Introduction

the complicated essence of many medical cases, traditional techniques including regression, discrimination, and other standard analyses often find it challenging to operate with constrained data. The inherent functional relations remain unknown, encompassing complex interactions and relationships across multiple variables.[10]

Neural network models, first introduced in the late 1950s as a subset of learning algorithms, have been progressively refined and have garnered increasing attention in recent times.[8] These algorithms are proficient in deciphering patterns within datasets by utilizing training data, offering a sophisticated solution for investigating intricate medical conditions where conventional strategies might be inadequate due to the sophisticated and diverse nature of variable relationships.[9] Hence, approaches based on neural networks, such as the Perceptron algorithm, are well-equipped to manage such challenging circumstances efficiently.

1.1 Material and data sources

This report is focused on the exploration of diabetes prediction utilizing data from the Pima Indians Diabetes Database available on Kaggle[3], originally sourced from the National Institute of Diabetes and Digestive and Kidney Diseases, and is also implemented following the preprocessing dataset accessible from the CSIE website[2].

The dataset under examination is composed of 768 instances and pertains solely to female patients of Pima Indian descent who are 21 years of age or older near Phoenix, Arizona. It encompasses eight features: Number of pregnancies, Plasma Glucose Concentration, Diastolic Blood Pressure (mm Hg), Triceps Skin Fold Thickness (mm), 2-Hour Serum Insulin $(\mu U/ml), BodyMassIndex(Weightinkg/(Heightin\text{m}^2)$, Diabetes Pedigree Function, and Age (years).

1.2 Method & Algorithm

This examination will initially employ the fundamental neural network algorithm known as Perceptron. Subsequently, advanced refinement will be carried out through potential hyperparameter tuning, incorporating alterations in learning rate, optimizer selection, and the quantity of neural network nodes, aiming to optimize and enhance the overall performance and efficacy of the model outcome. There exists an alternative implementation utilizing Artificial Intelligence through the application of Artificial Neural Networks (ANN) for the prediction of diabetes, achieving a Validation Accuracy of approximately 91.4%.[1]

2 Method Description

Exploring the principles of perceptron algorithms opens avenues to numerous associated methodologies designed for model implementation and refinement. These methodologies aim at achieving efficient outcomes by optimizing various elements such as the learning rate, selection of optimization algorithms, and the complexity of neural networks. Through meticulous application of these methodologies, it is possible to significantly enhance the model's capability to learn and generalize from the input data, providing a robust framework for solving complex problems in the domain.

2.1 Learning rate

In the context of Artificial Neural Networks (ANN), the learning rate serves as a pivotal hyperparameter during the training phase, playing a crucial role in the rule that updates the weights. Essentially, it governs the magnitude of the steps taken to attain a (local) minimum. A high learning rate may expedite the model's convergence but risks surpassing the minimum, potentially leading to divergence and consequently, a failure in learning. Conversely, a low learning rate can slow the learning process, potentially causing the model to become ensnared in local minima and prolonging the training period. Determining the optimal learning rate is imperative for the development of efficient models, impacting convergence, the caliber of the model developed, the duration of the training, and the potential for overfitting or underfitting.[4]

2.2 Optimization algorithms

Optimization algorithms in perceptron implementation are essential, serving a pivotal role in the training of deep learning models. They minimize the loss function and adjust the weights of the model, allowing it to make more precise predictions. Various optimization algorithms are employed in the implementation of perceptrons, including Stochastic Gradient Descent (SGD), Mini-Batch Gradient Descent, and Adaptive Moment Estimation (Adam)[5]. Each optimization algorithm has its own rate of convergence, thus selecting the appropriate algorithm is crucial. Faster convergence aids in saving computational time and resources and significantly influences the quality of the developed model.

2.3 Model Complexity

In Artificial Neural Networks (ANN), model complexity is typically characterized by the network's architecture, entailing the number of layers, the number of nodes within each layer, and the linkages among the nodes. The complexity of a model substantially influences its ability to learn from the dataset, generalize to unseen data, and its vulnerability to overfitting or underfitting.

The number of hidden layers within a network dictates the network's depth. Deeper networks have the capability to model more intricate functions and learn hierarchical representations. The number of nodes in each layer, on the other hand, determines the breadth of the network, allowing the network to learn more nuanced representations. Employing intricate models, characterized by a higher number of layers and nodes, enhances the model's learning capacity, allowing them to model sophisticated functions and patterns. However, it also elevates the risk of overfitting, which can lead to poor generalization to unfamiliar, unseen data.[6]

3 Method Implementation

The execution of the project begins with exploratory data analysis and preprocessing of data, followed by the implementation of the foundational Perceptron algorithm (single-layer perceptron). Further refinement and development of the base model are conducted through hyperparameter tuning, which includes adjustments to the learning rate, selection of an optimizer, model complexity, and the number of nodes in the neural network.

3.1 Exploratory Data Analysis

In following methodical procedures to explore the correlations between each variable and the results, the exploratory data analysis phase did not reveal any significant relationships or patterns. Nevertheless, the initial dataset displayed instances of outliers and issues related to non-scaling, as substantiated by the findings outlined in Section 4: Code.

3.2 Preprocessing data

Each of the predictive features has undergone manipulation through centering and scaling techniques. These techniques are instrumental in enhancing the quality of learning, accelerating convergence, preventing the occurrence of vanishing/exploding gradients, and subsequently, optimizing the overall performance of the model. This transformation was executed utilizing the **StandardScaler()** function available in the **sklearn** library.[9]

Finally, the data is separated into training and testing subsets utilizing the **train_test_split()** function from the **sklearn** library. Stratification is employed to maintain the proportionality of the outcomes, and the test size is designated as 0.25. Subsequent to this partitioning, the data types are converted to PyTorch tensors to facilitate ensuing analyses.

3.3 Based Perceptron Algorithm

The foundational model was executed using the **PyTorch** library, employing a single-layer perceptron, which undertook computations over the summation of input vector elements corresponding to their weights. Subsequently, the outputs, after being processed through the activation function and devoid of any hidden layers, had their performance computed. This layer is characterized by a fully connected layer, denoted as **torch.nn.Linear**, coupled with a Sigmoid activation function, represented as **torch.nn.Sigmoid**.

3.4 Hyperparameter Selection

Subsequent to establishing the base model, there is a necessity to refine it to achieve the optimal single-layer perceptron by selecting appropriate hyperparameters, such as varying the optimizer and learning rate. Nonetheless, the loss function has been predetermined as Binary Cross Entropy (**torch.nn.BCELoss()**), serving as a suitable optimizer for binary problems, and the number of epochs has been set to 1000 to depict a distinct trend in model performance.

Once the practical base model is ascertained, enhancements will be made by augmenting the model's complexity. This augmentation includes the addition of more layers, which are anticipated to more effectively discern patterns and relationships within each instance, and an increment in the number of neurons in each layer, projected to yield superior outcomes.

4 Experimental Analysis

In adherence to the methodology delineated in the preceding section, this implementation aims to achieve two objectives, as outlined in Part A and Part B. Part A necessitates the establishment of an appropriate foundational model, incorporating the tuning of the optimizer and learning rate. Concurrently, Part B is dedicated to the further refinement and development of this model. In this segment, the aim is to explore the repercussions of augmenting model complexity on its performance, achieved by the incorporation of more profound neural layers and an increased number of neural networks. This augmentation aims to capture the underlying relationships between each instance more effectively, thereby yielding enhanced quality of outcomes.

4.1 Part A: Based model - optimizer tuning

The base model was formulated utilizing a variety of optimization algorithms, encompassing Adam, Stochastic Gradient Descent, and RMSprop algorithm. Post-completion of 1000 epochs, the model employing the Stochastic Gradient Descent algorithm exhibited superior testing accuracy and demonstrated a minimal tendency for overfitting, as depicted in the accompanying Figure 1 and loss shown in Figure 2.

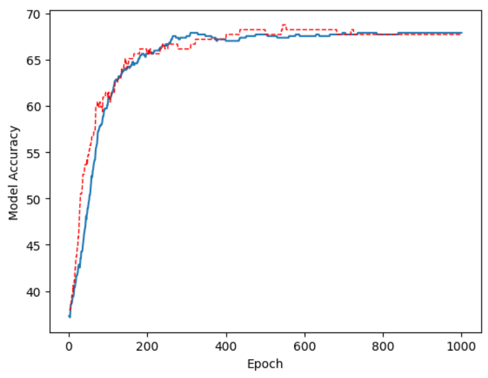

Figure 1: plot of epochs and model accuracy(train & test) on the base model implemented using SGD optimization

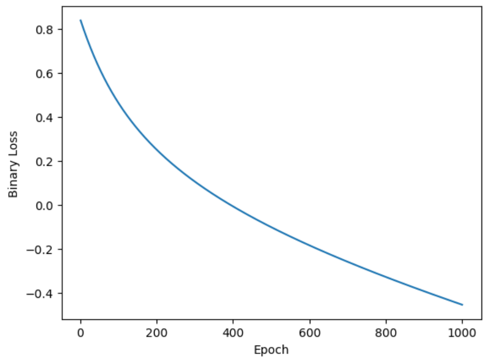

Figure 2: plot of epochs and binary loss on the base model implemented using SGD optimization

4.2 Part A: Based model - learning rate tuning

Subsequently, the base model was also experimented with, incorporating diverse learning rates such as 0.1, 0.05, 0.01, 0.001, and 0.0001. The variance in learning rates yielded approximated test accuracies of 65.1, 65.6, and 67.7 for learning rates 0.1, 0.05, and 0.01 respectively. It was observed that there was a declining trend post a learning rate of 0.001, with an accuracy of 60.9, and a substantial decrease to around 39.6 was noted at a learning rate of 0.0001. Hence, an optimal learning rate is discerned to lie within the range of 0.1 to 0.01. The data corroborating these findings can be found in referenced code.

4.3 Part B: Developed model - model complexity tuning

Upon achieving the optimal base model, we adapted a more intricate structure of the neural network to embrace the data's complexity by incorporating one and two hidden layers respectively. Additionally, we altered the number of neural networks to ensure a superior fit with the data's abstract structure, augmenting nodes to 16, 32, and 64. When compared, multilayer perceptrons exhibited enhanced results over single-layer perceptrons; however, the model with two hidden layers demonstrated high training accuracy but lower testing accuracy, indicating an overfitting issue as shown in Figure 3. Conversely, employing one hidden layer, depicted in Figure 4, yielded high accuracy in both training and testing datasets without overfitting concerns. Ultimately, amongst the chosen configurations, 32 nodes rendered the most favorable results, balancing high accuracy with minimal overfitting.

5 Code

The analysis was conducted using **Python**, leveraging the **PyTorch** framework, and **the results with specific detail** have been made publicly available through **GitHub** at the following URL: `https://github.com/possakorn/UoA_DL_2023_3_PK`

6 Conclusion

Employing perceptron algorithms can elucidate concealed patterns in the medical domain, enhancing the

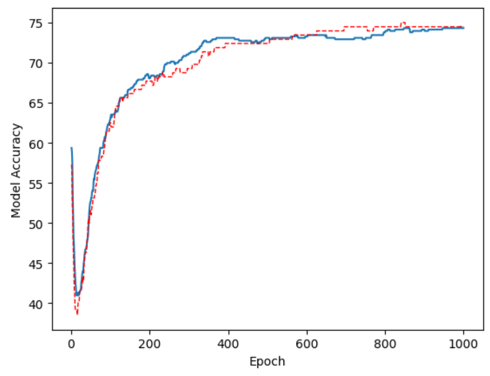

Figure 3: plot of epochs and model accuracy(train & test) on the developed model implemented using multi-layer perceptron - 1 layer and 32 nodes. At 1000 epochs, train acc equal to 74.3% and test acc equal to 74.5%

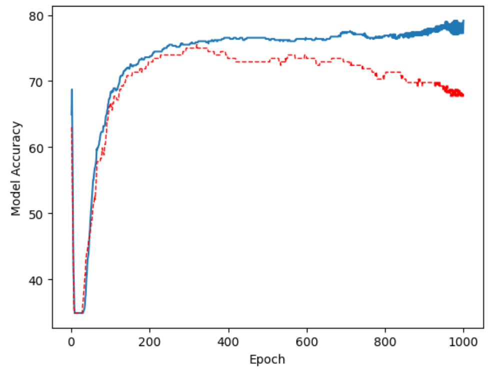

Figure 4: plot of epochs and model accuracy(train & test) on the developed model implemented using multi-layer perceptron - 2 layers and 32 nodes. At 1000 epochs, train acc equal to 79.2% and test acc equal to 68.2%

predictive accuracy for diabetes. Additionally, integrating more complexity into the model can augment its performance, but caution is essential to avoid overfitting on the test data. Based on the outcomes of the empirical analysis, a multi-layer perceptron with one hidden layer and 32 nodes is the most optimal under the tested conditions, yielding the highest accuracy without overfitting, achieving 74.5% accuracy at 1000 epochs compared to the 65.6% of the base model. Ultimately, while models with additional hidden layers did exhibit enhanced accuracy, they presented issues of overfitting. Therefore, future analyses could benefit from incorporating regularization techniques such as dropout, batch normalization, and weight initialization to mitigate the discrepancy between training and testing and to stabilize performance.[7]

References

[1] Deep learning prediction. https://www.kaggle.com/code/risenattarach/deep-learning-prediction-val-acc-91-45. Accessed: 2023-09-26. 2

[2] Libsvm data: Classification (binary class). https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html. Accessed: 2023-09-26. 1

[3] Pima indians diabetes database. https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database. Accessed: 2023-09-26. 1

[4] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. *CoRR*, abs/1206.5533, 2012. 2

[5] Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning, 2018. 2

[6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org. 2

[7] Jeremy Howard Sylvain Gugger. *Deep Learning for Coders with fastai PyTorch*. Oreilly, 2021. 5

[8] Frank. Rosenblatt. The perceptron, a theory of statistical separability in cognitive systems. *Cornell Aeronautical Laboratory, Inc. Report no. VG-1196-G-1*, 10:261–62, 1958. 1

[9] D. Rumelhart, G. Hinton, and James Mcclelland. A general framework for parallel distributed processing. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, 1, 01 1986. 1, 3

[10] Jack Smith, J. Everhart, W. Dickson, W. Knowler, and Richard Johannes. Using the adap learning algorithm to forcast the onset of diabetes mellitus. *Proceedings - Annual Symposium on Computer Applications in Medical Care*, 10, 11 1988. 1