Deep Learning Fundamentals - Assessment 1 - Designing and Optimizing a Single Layer Perceptron to make Predictions with Higher Accuracy

Mrinank Sivakumar
University of Adelaide
Adelaide SA 5005
mrinank.sivakumar@student.adelaide.edu.au

Abstract

This experiment involved designing a custom Perceptron (also interpreted as a dense layer neural network) and incorporating several optimization methods to maximize its prediction accuracy. For the purposes of this experiment, the Perceptron was trained and tested to predict diabetes using the PIMA Indians Diabetes Database (raw dataset in CSV format), sourced from Kaggle.

1. Introduction

Single Layer Perceptrons (SLPs) are fundamental to artificial neural networks, representing one of the simplest models for supervised learning. However, SLPs often have limited predictive power compared to more complex Neural Networks like Multi-Layer Perceptrons (MLPs). This is due to their single dense layer, which restricts their ability to learn complex input-output relationships. Despite these limitations, understanding SLPs is crucial as they provide the foundation for more advanced neural networks [1].

The goal of this experiment is to design, implement, and optimize an SLP to improve its predictive accuracy compared to the traditional SLP. This experiment uses the PIMA Indians Diabetes Database, sourced from Kaggle [2], which is widely used for testing classification models [3]. The original Perceptron, proposed by Rosenblatt in 1958, used a simple Perceptron Update Rule for binary classification [4] as follows:

$$\mathbf{w} \leftarrow \mathbf{w} + \eta \cdot y \cdot \mathbf{x} \tag{1}$$

where $\mathbf{w}$ is the weight vector, η is the learning rate, y is the true label, and $\mathbf{x}$ is the feature vector. In this experiment, the basic SLP is enhanced by incorporating alternative activation functions, regularization, and advanced optimization techniques such as Adam and SGD.

Traditionally, SLPs use the Binary Step activation function, which creates a hard decision boundary, making it challenging to train models with overlapping data. To address this, flexible activation functions were added such as Sigmoid, Tanh, and ReLU, which introduce non-linearity and make the model differentiable. Additionally, regularization techniques like L1, L2, and Elastic Net were used to reduce overfitting and improve generalization [5]. Advanced optimization techniques such as Adam and SGD were implemented to enhance the training process. Hyperparameter tuning was also conducted pre-training to find the optimal learning rate, activation function, and regularization parameters to maximize accuracy.

Preprocessing was an essential first step to ensure effective model training, given that a raw dataset with potential erroneous entries was used. This included data cleaning, splitting the dataset, imputation of missing values, scaling the data, and further splitting into training, validation, and testing sets.

This experiment demonstrated how modifying a basic SLP with advanced activation functions, optimization algorithms, and regularization techniques could significantly enhance its performance with a prediction task.

2. Description of Method

The methodology for designing, implementing, and optimizing an SLP involved a series of systematic steps aiming to enhance the traditional SLP and improve predictive performance. To start, the dataset was first loaded and examined using the pandas library [6].

2.1. Data Cleaning and Visualization

The initial phase of the experiment involved meticulous data cleaning and visualizing. The PIMA Indians Diabetes dataset, contained large amounts of erroneous values, which was confirmed using the 'describe()' function. Then, the following steps were taken as part of the data cleaning process:

2.1.1 Identification of Erroneous Entries

The dataset was scrutinized to identify impossible or extreme values. For instance, '0' values in features such as Glucose, BMI, and Insulin were replaced with missing values (NaN) for subsequent imputation [7]. Blood Pressure values below 40 mm Hg [8] and Skin Thickness values below 10 mm [9] were also considered unrealistic and flagged as missing values.

2.1.2 Imputation Strategies

Two approaches for handling missing values were considered:

MEAN IMPUTATION: Missing values were replaced with the mean of each respective feature. Although this method was simple, it failed to capture the underlying distribution of the data. The effects of this imputation method were visualized as shown in Figure 1 and Figure 2.[1]

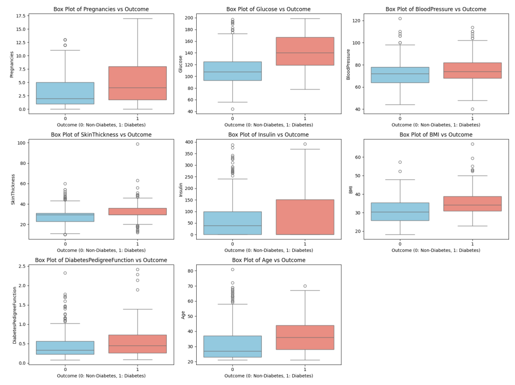

Figure 1. Box plot for each feature, stratified by the outcome variable and examining the value distributions for the two classes, thereby facilitating identification of outliers and effects of Mean imputation.

K-NEAREST NEIGHBOURS (KNN) IMPUTATION: This technique imputed missing values by leveraging the weighted average of the nearest neighbouring values, thereby considering feature similarities. KNN imputation was able to yield more accurate estimates compared to mean imputation. The effects of this imputation method were visualized as shown in Figure 3 and Figure 4.

2.1.3 Class Distribution Visualization

The distribution of diabetes and non-diabetes cases within the dataset were visualized using a bar plot as shown in Figure 5. This was crucial for understanding the class imbalance, which could influence the model's performance. The bar graph revealed a higher prevalence of non-diabetes cases, solidifiying the need to address class imbalance during training.

[1] As a note, I tried to organize the figures under their respective sections. However, I could not figure out why Overleaf places them randomly.

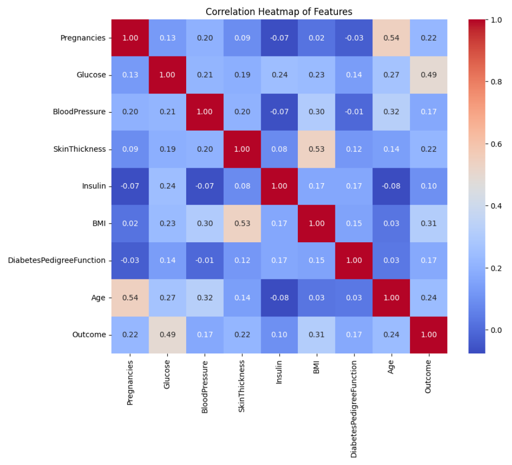

Figure 2. Heatmap of feature correlations elucidating relationships between variables after Mean imputation.

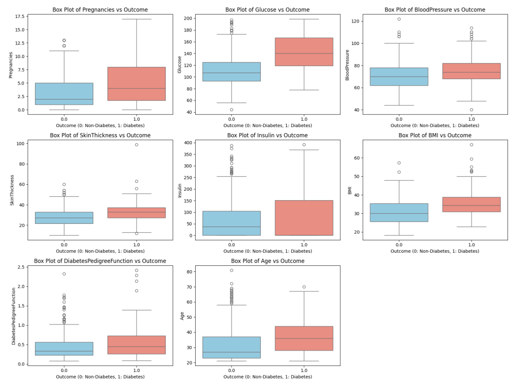

Figure 3. Box plot for each feature, stratified by the outcome variable and examining the value distributions for the two classes, thereby facilitating identification of outliers and effects of KNN imputation.

2.2. Data Splitting and Preprocessing

Following the data cleaning and visualization steps, the dataset was split into training, validation, and test sets to ensure robust model evaluation. The dataset was divided into features and the target variable ('Outcome') and subsequently split into training, validation, and test subsets. More specifically, 20% of the dataset was reserved for testing. The remainder once again had 20% split off to create a validation set, with the remainder becoming the training set. Stratified sampling was employed to maintain consistent class distributions across subsets.

Subsequent to splitting, the KNN Imputation and RobustScaler techniques from sklearn [10] were applied to preprocess the datasets. KNN Imputation was used to fill in

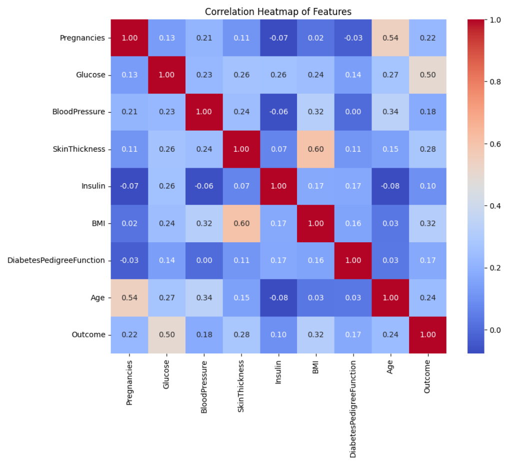

Figure 4. Heatmap of feature correlations elucidating relationships between variables after KNN imputation.

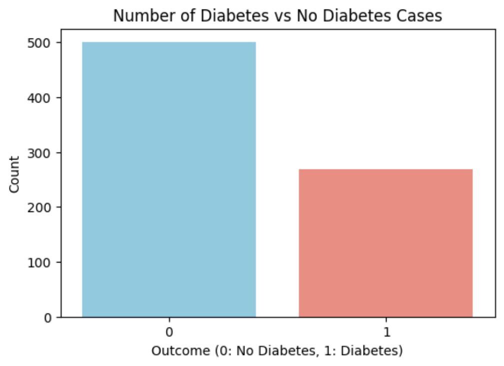

Figure 5. Class Distribution Bar Graph

missing values, while the RobustScaler was utilized to normalize feature values based on the median and interquartile range, mitigating the influence of outliers.

2.3. Model Definition

Three primary models were defined to facilitate performance comparisons: a dummy classifier, a Simple Single-Layer Perceptron (SSLP), and an Optimized Single-Layer Perceptron (OSLP).

DUMMY CLASSIFIER: A dummy classifier, from sklearn, using a stratified strategy was implemented as a baseline model. This classifier made random predictions while preserving the original class distribution, providing a benchmark for evaluating more sophisticated models.

SIMPLE SINGLE-LAYER PERCEPTRON (SSLP): An SSLP, was implemented using TensorFlow [11], and comprised of a single dense layer with a linear activation function (Binary Step Activation). The model was trained using the Adam optimizer and a binary cross-entropy loss function. This basic architecture served as the foundational model for comparison.

OPTIMIZED SINGLE-LAYER PERCEPTRON (OSLP): The OSLP was designed to incorporate various optimization techniques, including advanced activation functions (e.g., Sigmoid, Tanh, ReLU), regularization methods (L1, L2, Elastic Net), and optimizers (Adam, SGD). These enhancements allowed the model to capture more complex relationships in the data and improve generalization.

2.4. Hyperparameter Tuning

A comprehensive hyperparameter tuning process was conducted to identify the optimal configuration for the OSLP model. The hyperparameter space included multiple learning rates (0.001, 0.01), activation functions (Sigmoid, Tanh, ReLU), regularization methods (L1, L2, Elastic Net), and optimizers. A custom grid search style approach was employed, using itertools, to evaluate different combinations, with each configuration trained and validated using the training and validation datasets. Key metrics, such as validation accuracy and loss, were used to determine the best-performing configuration, with the optimal setup chosen based on the highest validation accuracy achieved during the grid search.

2.5. Training and Evaluation

Upon selecting the optimal hyperparameters, both the OSLP and SSLP models were trained on the training dataset and evaluated on the validation dataset. Metrics such as loss and accuracy were monitored during training to ensure effective learning, while mitigating signs of overfitting. The dummy classifier was also trained as a baseline for comparison.

The models were evaluated using metrics such as accuracy, balanced accuracy (more effective for class imbalance), and classification reports, providing insight into their ability to distinguish between diabetic and non-diabetic cases. Training and validation accuracy curves were plotted over epochs to visualize learning progress, and correlation heatmaps were plotted to visualize true vs predicted labels.

2.6. Final Training and Testing

After the initial training and validation, the OSLP and SSLP models were re-trained using the combined training and validation datasets to maximize the data available for learning. These final models were subsequently evaluated on the test dataset, which was held out to provide an unbiased assessment of model performance.

The final evaluation on the test dataset, once again, involved calculating metrics such as accuracy, balanced accuracy, and generating detailed classification reports. Confusion matrices were also used again, to visualize the models' performance in distinguishing between diabetic and non-diabetic cases.

The results indicated that the OSLP consistently outperformed the SSLP and the dummy classifier in terms of accuracy and generalization, highlighting the benefits of incorporating advanced optimization techniques into a simple neural network model.

3. Experimental Analysis

This experiment evaluated the performance of an optimized single-layer perceptron compared to a traditional single-layer perceptron to assess the effectiveness of enhancements made to the traditional SLP model.

3.1. Experimental Setup

The PIMA Indians Diabetes dataset was used for this experiment, and was split into training, validation, and testing subsets for unbiased evaluation. The SSLP, OSLP, and a baseline dummy classifier were trained on the training set, validated, retrained on a combined training set, and finally evaluated on the test set.

The objectives were as follows:

BASELINE COMPARISON: Establishing a baseline using a dummy classifier to ensure SSLP and OSLP predictions were performing better than 50/50 random guessing.

MODEL TRAINING AND VALIDATION: Analyzing the training and validation accuracy and focusing them on improving the impact of activation functions, regularization, and optimization techniques in the OSLP.

TESTING PERFORMANCE: Evaluating and maximising the model's performance on the test set using various metrics including accuracy, precision, recall, F1 score, and confusion matrices.

HYPERPARAMETER INFLUENCE: Examining how different hyperparameters affected the OSLP's learning through validation and loss.

3.2. Results and Observations

4. Conclusion

Optimizing the Single-Layer Perceptron (SLP) with advanced activation functions, regularization, and optimization algorithms led to significant performance improvements. The Optimized Single-Layer Perceptron (OSLP) consistently outperformed both the Simple Single-Layer Perceptron (SSLP) and the dummy classifier. Thoughtful model design and hyperparameter tuning were crucial for achieving strong predictive accuracy, especially with complex datasets like the PIMA Indians Diabetes dataset. Additional visualizations are available in the Jupyter notebook on the GitHub page, if interested.

4.1. Discussion

The SSLP lacked the capacity for non-linear modeling, limiting its effectiveness. Whereas, the OSLP's improvements highlight the importance of non-linearity and regularization in neural networks. The Adam optimizer facilitated faster convergence, though properly tuned SGD also showed good results. These optimizations made the OSLP a suitable model for predicting diabetes onset. Future research could stray away from single-layer solutions and explore deeper architectures or ensemble methods to further improve performance. Additionally, utilizing PyTorch in future implementations could significantly enhance the performance and efficiency of the code, despite potentially reducing readability.

5. References

[1] I. Goodfellow, Y. Bengio, and A. Courville. Deep Learning. MIT Press, 2016. See https://www.deeplearningbook.org/

[2] Kaggle. PIMA Indians Diabetes Database. See https://www.kaggle.com/uciml/pima-indians-diabetes-database

[3] J. W. Smith, J. E. Everhart, W. C. Dickson, W. C. Knowler, and R. S. Johannes. Application of the ADAP Learning Algorithm for Predicting the Onset of Diabetes Mellitus. In Proceedings of the Annual Symposium on Computer Application in Medical Care, pages 261-265, 1988.

[4] F. Rosenblatt. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. Psychological Review, 65(6):386-408, 1958.

[5] A. Y. Ng. Feature selection, L1 vs. L2 regularization, and rotational invariance. In Proceedings of the twenty-first international conference on Machine learning, page 78, 2004.

[6] Pandas Development Team. pandas documentation, 2021. See https://pandas.pydata.org/pandas-docs/stable/

[7] Z. Zhang and Y. Dong. Imputing missing data in a large clinical dataset: a comparison of five strategies. Journal of Statistical Computation and Simulation, 81(12):1581-1597, 2011. See https://doi.org/10.1080/00949655.2010.529356

[8] A. C. Guyton and J. E. Hall. Textbook of Medical Physiology (13th ed.). Elsevier, 2015.

[9] V. H. Heyward and D. R. Wagner. Applied Body Composition Assessment (2nd ed.). Human Kinetics, 2004.

[10] Scikit-learn. Scikit-learn: Machine Learning in Python. See https://scikit-learn.org/stable/

[11] TensorFlow. TensorFlow: An end-to-end open source machine learning platform. See https://www.tensorflow.org/

[12] BrownAssassin. DLF - A1 - Perceptron Optimization. See https://github.com/BrownAssassin/DLF—A1—Perceptron-Optimization