Introduction to Neural Networks
Johns Hopkins University — Engineering for Professionals Program
605.447/625.438
Fall Semester 2024
Classification Programming Assignment
Documentation

Landon Jones
Aurora Meng
Mohamed Affan Dhankwala
Yuliia Kolesnyk

Executive Summary

The primary objective of this project is to develop, implement, and evaluate two neural network models to address a critical classification problem in targeted advertising decision-making. Advertising campaigns often rely on accurate predictions to determine which consumer groups will respond positively to promotional efforts, maximizing revenue and ensuring efficient use of resources. This task involves predicting whether an advertising campaign for a given household will be worthwhile (coded as 1) or not (coded as 0), based on two key metrics: Size of Wallet (SOW) and Local Affluence Code (LAC). These metrics are derived from socio-economic factors, including disposable income, household composition, neighborhood affluence, and housing market charecteristics.

To tackle this challenge, the project implements two neural network architectures: a simple perceptron model and a feed-forward backpropagation (FFBP) network. The perceptron model serves as a baseline, leveraging basic linear separability with a sigmoid activation function. In contrast, the FFBP network introduces a hidden layer to capture complex, non-linear relationships between the input features (SOW and LAC) and the output classification (TACA). Both models undergo rigorous training using structured datasets, allowing the networks to minimize errors through iterative weight and bias adjustments.

The training dataset comprises odd-numbered rows of the provided data, while even-numbered rows are reserved for testing. The training process incorporates online learning techniques, repeatedly cycling through input-output pairs to refine the network's performance. A critical aspect of the project is determaining the optimal thresholds for mapping the output of the sigmoid activation function to binary classes. This involves experimentation and analysis of the network's receiver operating characteristics (ROC), including sensitivity, specificity, and predictive probabilities.

Through this approach, the project aims to demonstrate the effectiveness of neural networks as decision-support tools in the domain of targeted advertising. By comparing the performance of the perceptron and FFBP models, the study provides insights into their computational efficiency, classification accuracy, and adaptability. The ultimate goal is to enable advertisers to make data-driven decisions that maximize the return on investment, ensuring targeted campaigns are directed toward the most profitable consumer groups.

Problem Description, Network Design, and Its Solution

Problem Description
Advertising firms seek to maximize revenue by identifying profitable consumer groups to target for campaigns. The problem is framed as a binary classification task:

Inputs:

- Size of Wallet (SOW): Represents a household's disposable income (range: [0, 3]).
- Local Affluence Code (LAC): Reflects neighborhood affluence and socio-economic conditions (range: [0, 3]).

Output:

- Targeted Advertising Campaign Assignment (TACA): Binary outcome (0 or 1) indicating whether an advertising campaign is profitable.

The challenge is to train a neural network to classify households into these categories accurately using historical data provided.

Network Design
Two neural network models were implemented:
1. Single Perceptron Model:

- A fundamental unit of neural networks that calculates outputs using weighted inputs and a bias, followed by activation.
- Activation function: Sigmoid.
- Training: Adjusts weights and biases using gradient descent to minimize error.

2. Feed-Forward Backpropagation (FFBP) Network:

- A multi-layer neural network with:
    - Input Layer: Takes SOW and LAC as inputs.
    - Hidden Layer: Composed of multiple perceptrons that process inputs non-linearly.
    - Output Layer: Produces the TACA value.

- Training:
    - Implements backpropagation to adjust weights and biases iteratively using the perceptron delta rule and chain rule of differentiation.
    - Optimized over multiple cycles to minimize error.

How the Design Addresses the Problem
1) Accuracy and Generalization:

- The FFBP network leverages hidden layers to model complex non-linear relationships between inputs (SOW and LAC) and the output (TACA), improving classification performance compared to the perceptron model.

2) Optimization:

- Weight updates during training ensure minimal error, achieving convergence to optimal values.

3) Threshold Logic:

- By testing various thresholds, the models achieve better ROCs, fine-tuning for real-world conditions.

4) Data Usage:

- Training is conducted using odd-numbered data points, while even-numbered data points are reserved for testing. This ensures robust evaluation.

5) Adaptability:

- The models can be retrained with new datasets, making them flexible tools for varying advertising datasets and scenarios.

Computational Performance

Both of our neural network designs were trained on and tested upon the same set of data. However, the performance between the two can be compared. Performance is determined by the proportion of correctly predicted class labels. Performance can also stem from the code complexity and runtime complexity of our model. A model that is simple to understand can be easily reproduced while a model that is simple to run can complete faster than a more complex one.

When comparing the performance of the two models with respect to the accuracy of the two models, it is critical to remember that the simple perceptron consists of some input nodes projecting to a single output node. In our case, the two input values of LAC and SOW are passed in with respective weights for both input values and we return an output which modifies the original weights based on its delta error. There are only two variables, the edge weights, that can be modified. On the other hand, a neural network with one hidden layer with two hidden nodes contains six edges. This allows for a much higher level of tuning that can be applied to the model. For this reason, the accuracy of our simple perceptron is much lower than that of the hidden layer equipped one. This is supported within our code where we create a simple perceptron that is trained for 1,000 iterations and then accurately predicts 70% of the test data versus an undertrained hidden layer equipped neural network that was trained for 30 iterations and still accurately predicted 80% of the test data.

If one was to create a model that could be easily understood, the simple perceptron is a more easily understood model—although not by much. The simple perceptron deals with a direct connection of all input nodes to the output while the other model deals with hidden layers. Since we were tasked with creating a hidden layer equipped model with the same specifications as the module 7 model, we were not required to tune the number of hidden layers or the number of nodes per hidden layer. However, if we had to find the optimal values out, our model would perform better but the difficulty to find these nodes would require multiple model generation, training, testing and evaluating. For simple perceptrons, there isn't much to tune as there isn't much going on and thus, they are simpler to code and reproduce.

Finally, the testing runtime for both of these models, although not identical, is close enough to be considered identical. However, training a simple perceptron requires us to tune the values of two edges while the training sequence for the other model requires six. Even though this may not seem like much of a difference, the training iterations required to train six edges is much more than that to train two. Therefore, the overall runtime, training plus testing, of a simple perceptron is quicker than that of a hidden layer equipped perceptron.

Further Performance Improvement

Focusing on the metric of accuracy, we must identify ways that we can boost accuracy of our models. In our situation, predicting the correct label more often is a sufficient goal. For other models, the goal may be to improve precision or recall, based on the field of the model.

Due to lack of complexity on the simple perceptron, increasing the iterations from 30 to 1,000 had minimal impact on the model's performance. This implies that we either did not train enough or that we were overfitting the model. Since our original data set has only about 10 datapoints, both our models are severely underfit and therefore we predict a higher accuracy if our training set consisted of more data points. Pre-processing to remove any noisy values that are abnormal could have also improved our accuracy. Finally, we had no metric of stratification. Since we were limited to which data to utilize for training and testing, we could not ensure a properly stratified class label. Our training data consisted of 5 ones and 5 zeros for TACA. This implies a 50/50 split ratio among all data. However, our total dataset contains 8 ones and 12 zeroes. If we had properly matched the total dataset's class label ratio with that of our training data, we would expect a higher performance value.

All of the previous points can be drawn upon for inspiration on how to improve a neural network with hidden layer's accuracy. Iteration count, however, did play a large role in this model. Our model had an almost 100% accuracy rate when we trained it upon 1200 iterations versus an 80% accuracy rate when it was trained upon 30 iterations. We could have also increased the number of hidden layers and hidden layer nodes within these hidden layers. This, however, would be an overkill tactic since our original data is limited to 10 datapoints. If we were dealing with thousands of data points, increasing the number of hidden layers would have substantial increase in accuracy of the model. In our situation, the additional hidden layer may add a level of complexity that hinders the model and damages our accuracy.

Summary and Conclusion

To conclude, comparison between the two models on a decisive note would miss out on crucial tradeoffs. The simple performance is quicker and easier to understand, code and reproduce but lacks accuracy. A neural network with hidden layers is more accurate but at the cost of runtime and code complexity. Both models have apparent strengths and weaknesses and can be applied within their respective fields. The accuracy of both models can also be improved by a couple of shared factors such as increasing the number of data points in the original dataset and stratifying the class label within the training dataset to match the overall class label proportion. The hidden layer equipped model could also be trained on more iterations for better performance. Additional hidden layers and more nodes per hidden nodes is an option—although it must be tuned properly to ensure no overfitting or divergence.