Programming Assignment 4: A Comparative Analysis of Neural Networks

Wanyi(Julia) Dai wdai17@jh.edu
Department of AMS
Johns Hopkins University
Baltimore, MD 21218, USA

Editor:

Abstract

This paper provides a comprehensive overview of Programming Assignment 4 from the course "Introduction to Machine Learning." The primary objective of this project is to implement a simple linear network (for regression data), logistic regression (for classification data), a traditional feedforward neural network with two hidden layers, and an autoencoder-based network. Upon project completion, we will apply k×2 cross-validation to all the datasets with the models, investigating the best hyperparameters and conducting a statistical analysis of the performance of each dataset in different neural networks.

Keywords: Programming Assignment 4, Backpropagation, Feedforward Neural Networks, Logistic Regression, Autoencoder-based Network, 5×2 Cross Validation, hyperparameter tuning

1. Introduction

With its remarkable capacity to uncover meaningful patterns within data, machine learning is an indispensable tool in the contemporary landscape of data-driven decision-making. Among many machine learning algorithms available, neural networks emerge as a fundamental genre of models with wide-ranging applications across diverse domains. This paper aims to present a comprehensive comparative analysis of the performance of these four essential machine-learning models. Our focus lies in evaluating their effectiveness in both regression and classification tasks, employing a variety of datasets. By meticulously examining their performance across diverse problem domains, we aim to illuminate the strengths and weaknesses inherent in each model. Additionally, we explore the selection of optimal hyperparameters tailored to each dataset, providing valuable insights that can inform future applications across real-world problems.

2. Problem Statement

This paper addresses this need by investigating the practical application of three significant neural network algorithms: simple linear regression for regression tasks and logistic regression for classification, a traditional feedforward neural network with two hidden layers, and a feedforward neural network with an autoencoded hidden layer. If applicable, we will apply the algorithms on diverse datasets and assess their ability to improve predictions through

hyperparameter tuning on learning rate and number of hidden neurons.

We expect that simple linear networks (including logistic regression for binary classification) are well-suited for regression or classification tasks when the relationships between input features and the target variable are predominantly linear. However, complex classification tasks with nonlinear decision boundaries may not perform as well as more complex models. Traditional feedforward neural networks with two hidden layers can capture both linear and nonlinear relationships within data. The right architecture and hyperparameter tuning can outperform simple linear networks on classification and regression tasks when the underlying relationships are nonlinear or when higher-order interactions between features are essential. Autoencoder-based networks are primarily designed for unsupervised learning and feature representation tasks. They may not be the first choice for traditional regression problems, as they focus on learning compact feature representations rather than directly predicting continuous target variables.

In general, we expect that the performance of these neural network architectures on regression and classification data depends on the nature of the data and the relationships between features and target variables. Simple linear networks are suitable for linear problems. Traditional feedforward networks excel at capturing both linear and nonlinear patterns, while autoencoders are more commonly used for unsupervised learning and feature extraction tasks rather than direct regression or classification.

3. Datasets

Before conducting experiments or applying any algorithm to the dataset, conducting a thorough preliminary analysis of each dataset is essential. This analysis should unveil the dataset's inherent characteristics, as they can significantly influence the success or failure of the algorithm in question. Understanding these dataset attributes is foundational to making informed decisions and achieving optimal results in the subsequent data analysis or modeling processes.

3.1 Classification Data

Classification datasets hold a critical role in the study of machine learning, offering a fertile ground for understanding the intricate interplay under the algorithm of ID3. These datasets challenge us to categorize data points into discrete classes or categories, a task that decision trees excel at. Hence, we expect fewer complications in the decision tree generation, and the decision tree should perform well on the following data:

- **Breast Cancer Wisconsin:** This breast cancer data set was obtained from the University of Wisconsin, aiming at distinguishing between benign and malignant breast tumors
  - Row number: 699
  - Features: [sample code, clump thickness, uniformity of cell size, uniformity of cell shape, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli, motoses]. Each feature is constructed by integers ranging from 1 to 10.

  - Target range: [2, 4]

- **Car:** The data is on evaluations of car acceptability based on price, comfort, and technical specifications.
  - Row number: 1728
  - Features: [buying, maint, doors, persons, lug boot, safety]. The features are all categorical variable, that we need to address in the preprocessing step.
  - Target range: [unacc, acc, vgood, good]

- **House Votes:** This data set includes votes for each U.S. House of Representatives Congressmen on the 16 key votes identified by the Congressional Quarterly Almanac
  - Row number: 435
  - Features: [handicapped, water project cost sharing, adoption of the budget resolution, physician fee freeze, El Salvador aid, religious groups in schools, anti-satellite test ban, aid to Nicaraguan contras, mx missile, immigration, synfuels corporation cutback, education spending, superfund right to sue, crime, duty-free exports, export administration act South Africa]. The features are all categorical and need to be one-hot encoded. This would results in a large and possibly sparse matrix that might not yield better results than the others.
  - Target range: [republican, democrat]

3.2 Regression Data

Regression datasets serve as a pivotal element in machine learning, allowing us to explore the intricate connection between data characteristics and the application of neural networks. While classification data categorizes information into discrete classes, regression data embarks on predicting continuous numerical values. If there are linear relationships between the feature and the target, we would expect regression datasets to yield better results.

- **Abalone:** Predicting the age of abalone from physical measurements.
  - Row number: 4177. There are a lot of data points in this dataset. With most of the features being numeric, we could ended up with a large matrix that takes longer time to perform the gradient descent.
  - Features: [sex, length, diameter, height, whole weight, shucked weight, viscera weight, shell weight]
  - Target range: [1 to 29]

- Machine: The authors evaluated the estimated relative performance values using a linear regression method.
  - Row number: 209
  - Features: [vendor, model, MYCT, MMIN, MMAX, CACH, CHMIN, CHMAX, ERP]. The dataset machine has complicated features, primarily numerical, but

the feature of model and vendor are categorical with many categories. This could result in a large and sparse matrix. A brief analysis on these variables should help us determine what should we do with them later.

– Target range: [6 to 1150]. The massive range could make it harder to tune the gradients.

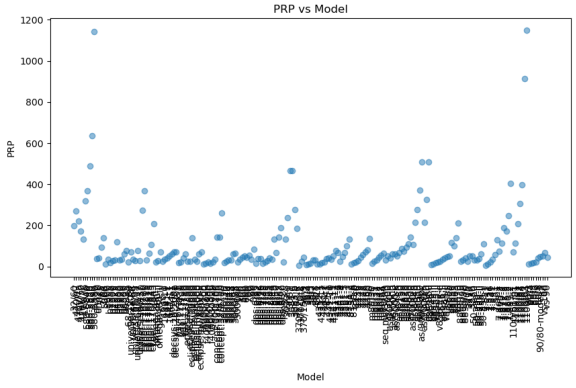

Figure 1: Figure 2: PRP vs. Model

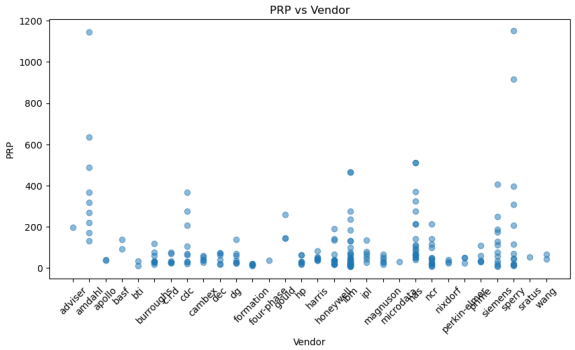

Figure 2: Figure 3: PRP vs. Vendor

– As shown in the Figure 2, the models are distributed fairly randomly and there are only one or two values a model type. Figure 3 presents a much better image, and we could use ANOVA to calculate the correlation between vendor and PRP, which results in p-value = 1.9e-5. Thus, we will keep vendor and one-hot encode it during the preprocessing step.

- Forest Fires: This dataset uses meteorological and other data to predict the burned area of forest fires in the northeast region of Portugal.

  – Row number: 517

  – Features: [X, Y, month, day, FFMC, DMC, DC, ISI, temp, R.H., wind, rain, area]

  – Target range: [0 to 1090.84]. The data from the target variable is very skewed (Figure 1) that we might need to transform it to adjust the influence of the zero values.

4. Preprocessing Steps

For us to experiment, several preprocessing steps exist on the datasets. After properly loading and stroring the datasets, we first must handle the missing data before giving a dataset to a machine learning algorithm. Most of our datasets are complete, except the breast cancer Wisconsin data, which the missing value is credited with the column mean. Note that house vote data also has "?", but it is denoted as a category of neither "yes" nor "no" instead of missing data.

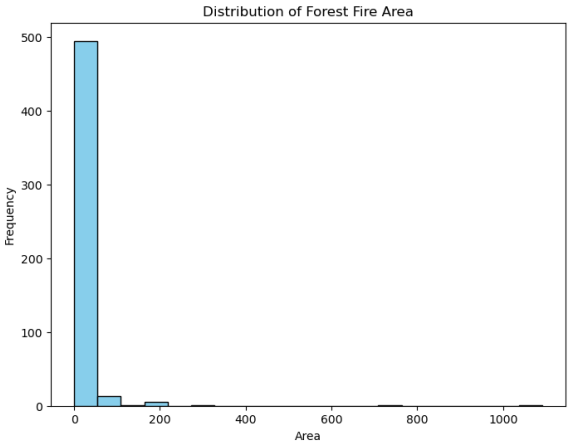

Figure 3: Figure 1: Distribution of Forest Fire Area

4.1 Handling Categorical Data

Categorical data consists of discrete, distinct categories or labels representing different groups or classes in the dataset. They are often described as strings in the dataset, which could be hard to use in computation. We need to convert the categorical data into numbers while preserving the information. There are two types of categorical data in our datasets.
Handling Ordinal Data: Ordinal data is data with order, such as [low, med, high]. We could match this data to an integer series to replace the string type. Modifications are made for the following datasets and columns.

- Car: replacing the following columns with integer series 1, 2, 3, ...: [buying, maint, doors, persons, lug_boot, safety, class].

Handling Nominal Data: Nominal data is categorical data without order. We apply one-hot encoding to cast this data type into higher dimensional binary space. Modifications are made for the following datasets and columns.

- Abalone: sex
- House votes: all columns
- Forest Fire: month, day. They have cyclic natural that are misleading if we treat them as ordinal data. It would be hard to deal with cyclic data in neural networks, and for simplicity, we will be treating them as nominal categorical data.
- Machine: vendor, model. As our analysis shows, we will remove model to avoid high dimensional sparse data. Vendor is kepted and one-hot encoded as it has a correlation with the target value proven by the ANOVA test.

5. Experimental Approach

There are generally two steps in our experiment. First, build the neural network models we will be using on our datasets. Then, apply the k×2 cross-validation to tune the hyperparameters. All the mathematical functions are shown in Appendix 1.

5.1 Simple Linear Network for Regression Data

Simple linear regression serves as the cornerstone of predictive modeling. Its elegance lies in its simplicity—a single linear function mapping input features to a continuous target variable. We will assess how well this straightforward model performs on regression datasets, examining its ability to capture linear relationships in the data. The algorithm follows the following step:

1. Initialize weights and bias as zeros.
2. For the number of iterations, do the following:
    - Calculate the current prediction.
    - Use the current prediction to calculate the gradients. (See formula in appendix)
    - Update the weight and bias using their corresponding gradients.

Assumptions made for the algorithm:
1. The regression output is calculated by $y = wX + b$
2. The classification output is calculated by feeding into the softmax layer.

5.2 Traditional Feedforward Neural Network with Two Hidden Layers

Feedforward neural networks, with their capacity to model complex nonlinear relationships, are the workhorses of deep learning. By implementing a traditional feedforward neural network with two hidden layers, we aim to explore the potential improvements in performance over linear models for both regression and classification problems. We follow the below steps to implement the algorithm:

1. Initialize the weight and bias for each layer.
2. Forwardpropagate each layer: Implement the forward pass through the network to calculate predictions for the entire training dataset. Compute the weighted sum and apply the activation function for each neuron in each layer. Adjust the weights and biases in the direction that minimizes the loss.
3. Calculate loss: Compute the loss by comparing the predicted values with the actual target values using the chosen loss function.
4. Backpropagation each layer (backward): Perform backpropagation to calculate gradients of the loss with respect to network weights and biases. Compute gradients layer by layer, starting from the output layer and moving backward through the hidden layers. Use the chain rule to calculate gradients efficiently.

Assumptions made for the algorithm:
1. The activation function is defined as logistic function for our case.
2. The loss function is defined as mean sqaured error (MSE) for regression data and cross-entropy loss for classfication data.

5.3 Autoencoder-Based Network

Autoencoders, a class of neural networks used for unsupervised learning and feature representation, offer a unique perspective on feature extraction and data reconstruction. In this study, we will examine the utility of autoencoders in capturing latent representations of data and their potential advantages in both regression and classification scenarios. The autoencoder network has the same algorithm as the traditional feedforward neural network, but the forwardpropagate and backwardpropagate steps involves encoding and decoding. The encoded value is then passed on to the next layer.

5.4 K×2 Cross Validation

We will use k×2 cross-validation with k = 5 for hyperparameter tuning on each dataset's neural network models. The 5x2 cross-validation includes the following steps *(Quoting from the Project 1 description)*:

1. Divide data into two parts: 80% will be used for training, and 20% will be used for tuning, pruning, or other types of "validation." Stratify so that the class distributions are the same in the partitions if the dataset type is classification.

2. Do the following five times: Divide the 80% into two equally sized partitions. Construct the neural network model of our choice using a candidate set of parameters. Each model will be trained with one of the halves and tested on the held-out 20%.

3. Average the results of these ten experiments for each parameter setting and pick the parameter settings with the best performance.

4. Do the following five times: Divide the 80%. again into two equally sized partitions (stratified). Train a model using the tuned hyperparameters on the first half and test on the second half. Train a model using the tuned hyperparameters in the second half and test in the first.

5. Average the results of these ten experiments and report the average and the chosen hyperparameter setting.

(End of Quote)
Assumptions made for the algorithm:
1. The mean square error denotes the regression performance.
2. The accuracy denotes the classification performance.
3. To determine our choice of parameters (learning rate for simple linear network, learning rate and number of hidden neurons for the others), the regression datasets select the parameters with minimum mean square error; the classification datasets select the parameters with the maximum accuracy.

5.5 Choice for Hyperparameter Tuning

Number of iterations: 1000. We set number of iterations the same for all the models for comparison purposes. It is also big enough to have the result converge, without running for unnecessary time.
Learning rate: Generally, we will start with a range of values that covers from zero decimals (10^0) to three decimals (10^-3): [0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1]. However, there are cases where these rate do not produce valid output, and we will try different values to find a calculable one. Then, our hyperparameter tuning will build around that value.
Number of Hidden Neurons: This parameter is only tuned for the neural networks with hidden layers (traditional feedforward neural network with backpropagation and autoencoded based neural network). Setting the value too high might harm the result with curse of dimensionality, so we chose to start with values: [1, 2, 3]

6. Results

To compare the performance and best hyperparameters chosen for each model, tables will be utilized to summarize our results from running the 5x2 cross-validation. We will not record the steps of the 5x2 cross-validation but compare the results directly. The attributes of our result tables are as follows:

1. Dataset Type: whether the dataset is regression or classification, we will combine datasets of the same type for easy comparison.

2. Dataset: name of the dataset.

3. Number of Rows: number of rows in the dataset / number of instance of the data.

4. Best Hyperparameters: the best hyperparameter (combination if there is more than one tuning value) will be used to calculate the average performance. In our case we have learning rate and number of hidden neurons (as neuron num in the table).

5. Performance: the average performance of the ten folds using the best hyperparameter. Note we have mean square error (MSE) for regression data and accuracy (A) for classification data.

The result tables for the three neural networks are shown as in Table 1, 2, and 3:

Table 1: Table 1. Result table of simple linear network / logistic regression

Dataset Type	Dataset	num of rows	Best Hyperparameters	Performance
Regression	Abalone	4177	learning rate = 0.5	MSE = 1.5992
Regression	Forest Fire	517	learning rate = 1e-06	MSE = 1.1305
Regression	Machine	209	learning rate = 1e-09	MSE = 51.2149
Classification	Breast Cancer	699	learning rate = 1	A = 0.9480
Classification	Car	1728	learning rate = 0.5	A = 0.8132
Classification	House Votes	435	learning rate = 0.1	A = 0.9441

Table 2: Table 2. Result table of two-layer traditional feedforward neural network with backpropagation

Dataset Type	Dataset	num of rows	Best Hyperparameters	Performance
Regression	Abalone	4177	learning rate = 1, neuron num = 3	MSE = 8.9759
Regression	Forest Fire	517	learning rate = 1e-07 neuron num = 2	MSE = 1.1623
Regression	Machine	209	learning rate = 1e-09, neuron num = 3	MSE = 106.4037
Classification	Breast Cancer	699	learning rate = 0.01, neuron num = 1	A = 0.3441
Classification	Car	1728	learning rate = 0.01, neuron num = 1	A = 0.7004
Classification	House Votes	435	learning rate = 0.01, neuron num = 1	A = 0.6138

Table 3: Table 3. Result table of two-layer autoencoder based feedforward neural network with backpropagation

Dataset Type	Dataset	num of rows	Best Hyperparameters	Performance
Regression	Abalone	4177	learning rate = 0.1, neuron num = 3	MSE = 8.9172
Regression	Forest Fire	517	learning rate = 1e-07, neuron num = 1	MSE = 1.0892
Regression	Machine	209	learning rate = 5e-10, neuron num = 1	MSE = 105.8119
Classification	Breast Cancer	699	learning rate = 0.01, neuron num = 1	A = 0.3438
Classification	Car	1728	learning rate = 0.01, neuron num = 1	A = 0.7102
Classification	House Votes	435	learning rate = 0.01, neuron num = 1	A = 0.6141

7. Discussion of the Behavior

Based on the provided result tables for three different types of neural networks (simple linear network/logistic regression, two-layer traditional feedforward neural network with backpropagation, and two-layer autoencoder-based feedforward neural network with backpropagation), we can observe several important points about their behavior.

Firstly, the simple linear network performs well on classification datasets (A ¿ 0.8), which achieves the highest accuracy (A = 0.9480) on the "Breast Cancer" dataset. It performs

well on abalone and forest fires data with MSE ¡ 2, indicating the strong linear relationship within these datasets. However, it has relatively high MSE values on machine datasets (MSE = 51.2149). The machine also has the lowest learning rate of 1e-09 to generate the best performance, while breast cancer has the highest learning rate that generates the best performance (learning rate = 1). Comparing the best hyperparameters between classification and regression, we could see that regression datasets tend to have a lower learning rate than classification data.

Table 2 shows that the traditional feedforward neural network generally performs poorly on regression datasets, with higher MSE values than the simple linear model. The MSE for forest fires remains similar to the simple linear model (+0.03), while the MSE of abalone is almost six times the simple linear model. In classification, it achieves moderate accuracy (A ¿ 0.3) but is outperformed by the simple linear network in terms of classification accuracy. The closest accuracy compared to the linear model is the car, with -11% accuracy using the two-layer traditional feedforward neural network. Regarding hyperparameter tuning, we could see that the best number of hidden neurons of classification data is always 1, while this parameter tends to be two and 3s in regression data.

The autoencoder-based network shows performance similar to the traditional feedforward neural network, with high MSE values on regression datasets. Interestingly, forest fires yield the best MSE with this model, with MSE = 1.0892, indicating that an autoencoder-based network might be the best model for forest fires. For classification, we yield the similar results, as well as the similar hyperparameter tuning. Thus, a feedforward neural network might not perform well in classification data.

In general, the choice of learning rate and the number of hidden neurons (neuron num) appear to impact the model performance significantly. The simple linear network performs better classification, particularly on the "Breast Cancer" dataset. The traditional feedforward and autoencoder-based networks show similar behavior and are less effective in classification tasks. Further hyperparameter tuning or alternative architectures may be explored to improve regression performance, especially for datasets like machines.

8. Conclusions

In conclusion, our study underscores the importance of selecting the exemplary neural network architecture for specific tasks. While the simple linear network proved effective in classification, there is room for improvement in regression scenarios. Researchers and practitioners should consider dataset characteristics, hyperparameter tuning, and the nature of the problem when choosing an appropriate neural network. Future research efforts should focus on refining existing models and exploring novel approaches to address the challenges of regression tasks. These insights contribute to the ongoing dialogue surrounding neural network selection and optimization, paving the way for improved machine-learning model performance in classification and regression applications.

Appendix A.

.1 Activation Function

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{1}$$

.2 Gradient Calculation Formula

$$\nabla w = \frac{1}{n} \cdot (X^\top (y_{pred} - y_{true})) \tag{2}$$

$$\nabla b = \frac{1}{n} \cdot \Sigma (y_{pred} - y_{true}) \tag{3}$$

.3 Gradient Calculation Formula for Back Probagation

Let the forward step be calculated by:

$$Z = W \cdot A_{prev} + b \tag{4}$$

$$A = activationFunction(Z) \tag{5}$$

Then the backward gradients are calculated by:

$$\frac{dC}{dA} = lossFunction \tag{6}$$

$$\frac{dC}{dA} = d\_activationFunction(Z) \cdot \frac{dC}{dA} \tag{7}$$

$$\frac{dC}{dW} = \frac{dC}{dZ} \cdot A_{prev}^\top \tag{8}$$

$$\frac{dC}{db} = \frac{dC}{dZ} \tag{9}$$

$$\frac{dC}{dA_{prev}} = W^\top \cdot \frac{dC}{dZ} \tag{10}$$

References