Deep Learning Homework 1

312511052 李元中

Brief intro of my code

My Python code provided entails the implementation of a neural network for regression analysis on the two dataset. The following is a succinct overview of the key steps and functions employed in this code:

Activation Functions: Several activation functions are defined, including sigmoid, relu, linear, tanh, and softmax. These activation functions are critical for both the forward and backward propagation within the neural network layers.

Neural Network Initialization: The 'init_layers' function is responsible for initializing the weights and biases of the neural network. The architecture of the network is explicitly specified through the 'nn_architecture' list.

Forward Propagation: The 'forward_propagation' function computes the forward pass of the neural network, encompassing the application of activation functions, and caches intermediate values for subsequent use in backpropagation.

Backward Propagation: The 'backward_propagation' function calculates the gradients of the network's parameters concerning the loss using the backpropagation technique.

Loss Functions: Two distinct loss functions are defined, namely Mean Squared Error (MSE) and Cross-Entropy. These functions play a pivotal role in quantifying the error during the training process.

Parameter Update: The 'update' function is responsible for updating the network's parameters, which include weights and biases. These updates are based on the gradients and a specified learning rate.

Training: The training of the neural network is accomplished using the 'train' function. This function involves the initialization of the network, execution of forward and backward propagation over a specified number of epochs, and the recording of loss values at each epoch.

Model Training: The 'train' function is employed to train the neural network on a selection of features and target values, with a specific architecture and loss function.

1) Regression

(a) Implement the nn for regression by using the energy efficiency dataset.

Data Handling and Preprocessing: The dataset is initially loaded from a CSV file, namely 'energy_efficiency_data.csv,' leveraging the Pandas library. To ensure data randomness, it is shuffled, and subsequently, a division into training and testing sets is performed. Moreover, feature selection is conducted using Spearman correlation to identify the most pertinent features for the regression task.

Evaluation: Post training, the model is utilized to make predictions on both the training and testing datasets. Subsequently, the Root Mean Square (RMS) error is computed for both sets, and the results are printed.

Visualization: To facilitate a visual assessment of the model's performance, the code includes visualization components. This comprises a learning curve illustrating the progression of loss values across epochs, as well as regression result visualizations that compare actual labels to predicted labels for both training and testing data.
In summary, this code exemplifies the process of training a neural network for regression, with a particular emphasis on predicting the heating load. The inclusion of visualizations enhances the ability to gauge the model's performance.

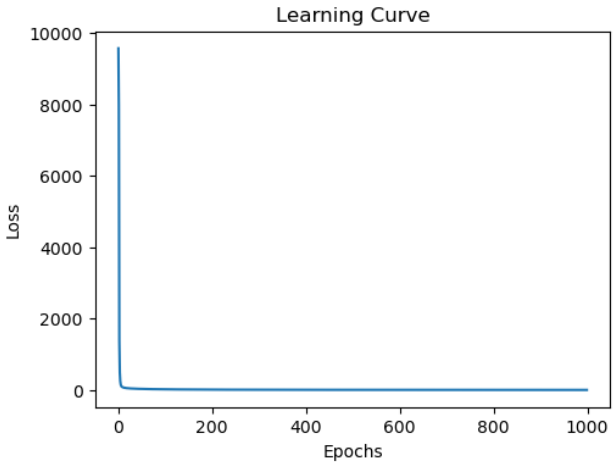
(b) Show result

(1) network architecture

```
nn_architecture = [
    {'input_size': X_train.shape[1], 'output_size': 64, 'activation': relu},
    {'input_size': 64, 'output_size': 32, 'activation': relu},
    {'input_size': 32, 'output_size': 1, 'activation': linear}  # Linear activation for regression
]
```

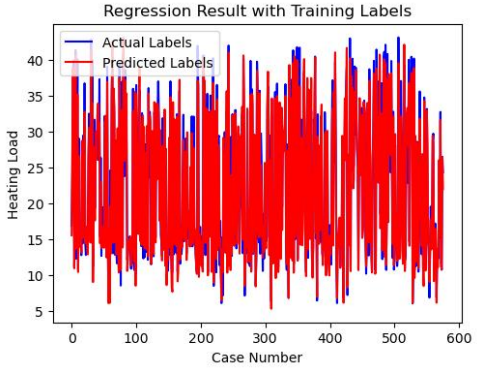
As shown in figure, the neural network architecture comprises three layers. The input layer dynamically adjusts its size to match the number of input features in the dataset. Following this, two hidden layers are introduced. The first hidden layer consists of 64 neurons and employs the rectified linear unit (ReLU) activation function, which introduces non-linearity. The second hidden layer, comprising 32 neurons, continues to utilize the ReLU activation function. Finally, the output layer, designed for regression tasks, possesses a single neuron with a linear activation function, facilitating the direct prediction of continuous values, specifically for forecasting the heating load based on the provided input features.

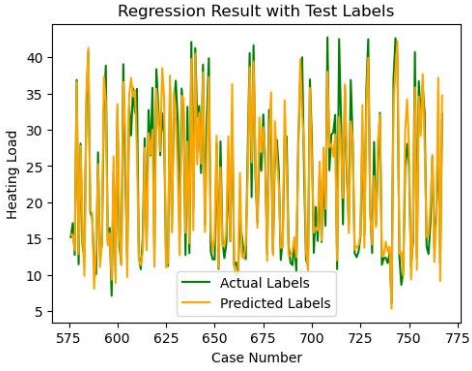
Result 1 (All 16 features are selected) :



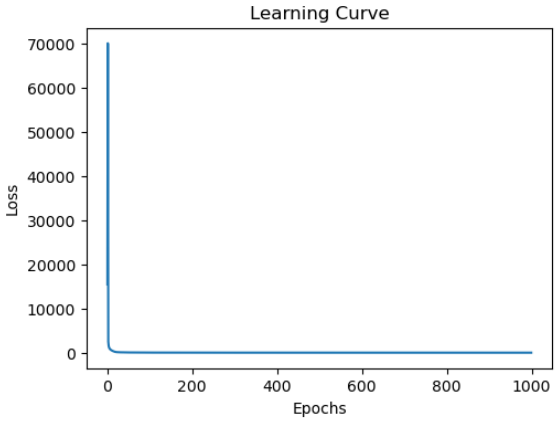
Training RMS Error:
2.253396146525804

Testing RMS Error:
3.0113293248843798



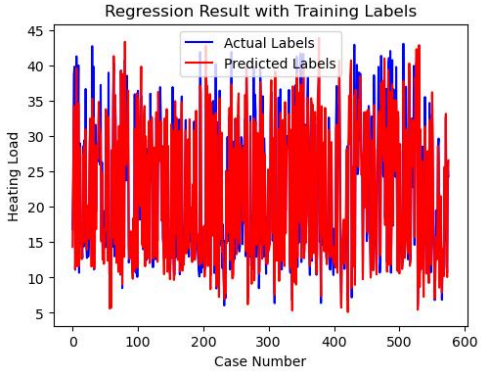


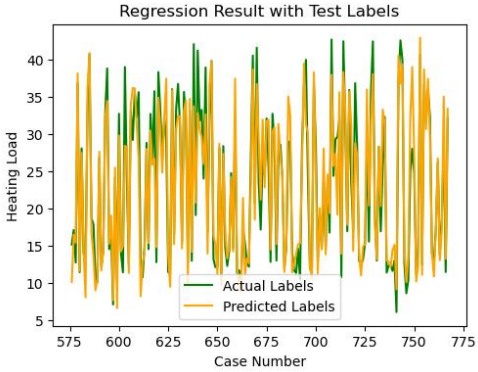
Result 2 (12 are selected) :



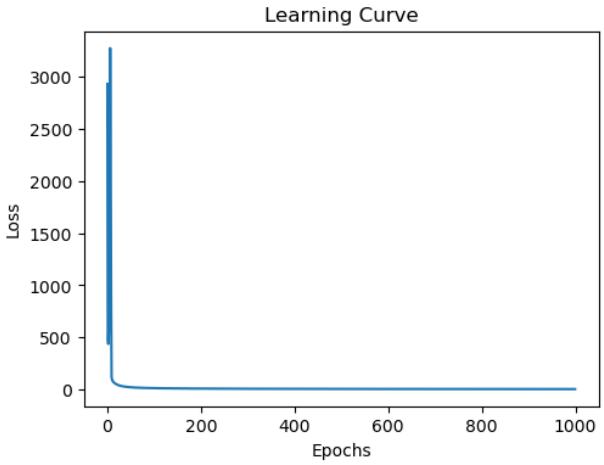
Training RMS Error:
1.0038239238762356

Testing RMS Error:
0.9648298211921631



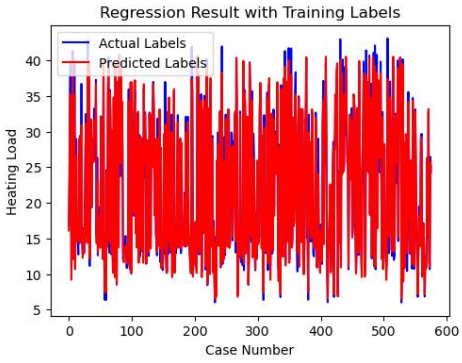


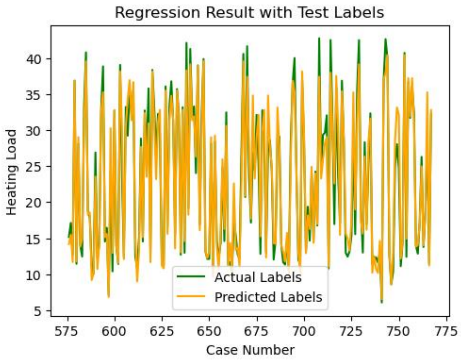
Result 3 (8 are selected) :



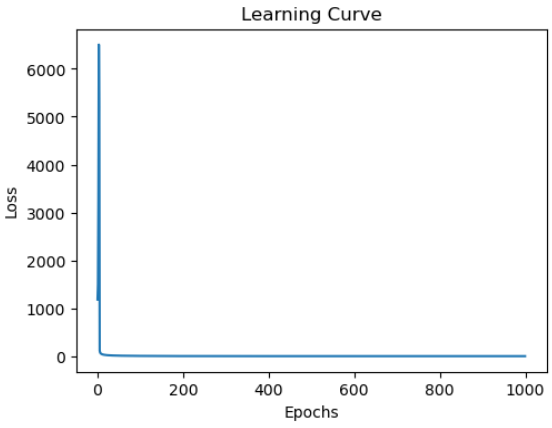
Training RMS Error:
1.8115861336174046

Testing RMS Error:
2.4586820378213363



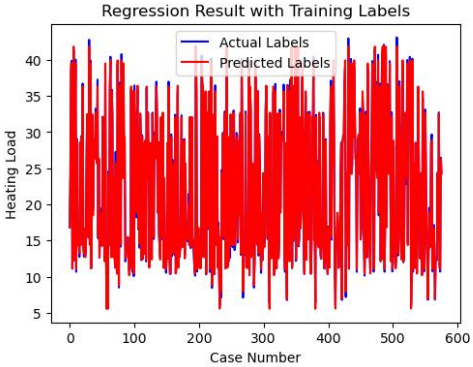


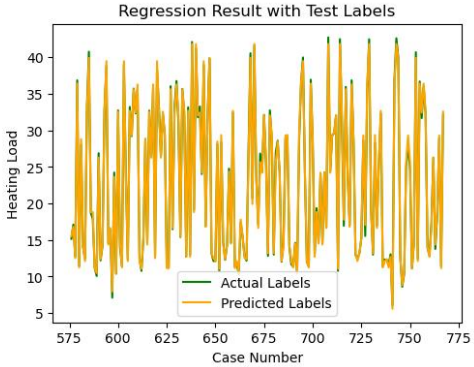
Result 4 (4 are selected) :



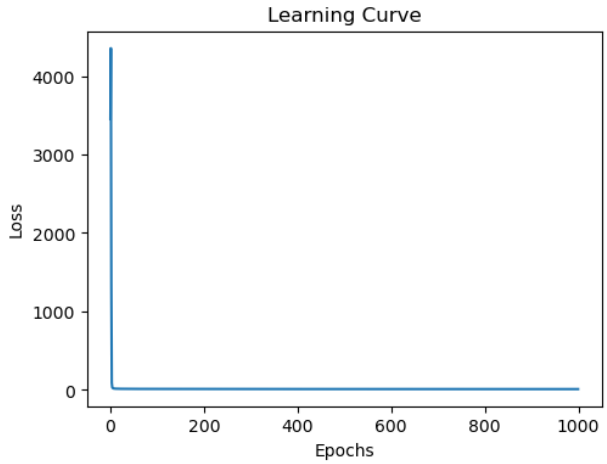
Training RMS Error:
0.5143609538063602

Testing RMS Error:
0.46098387601570007



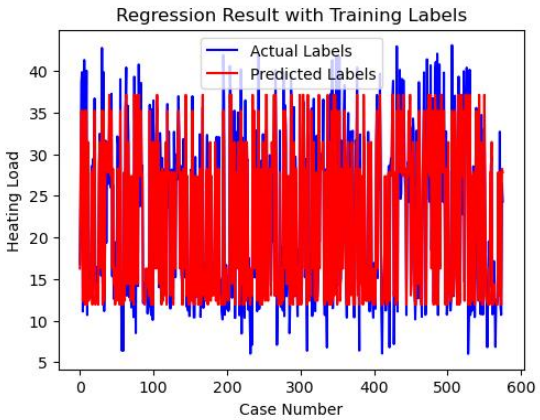


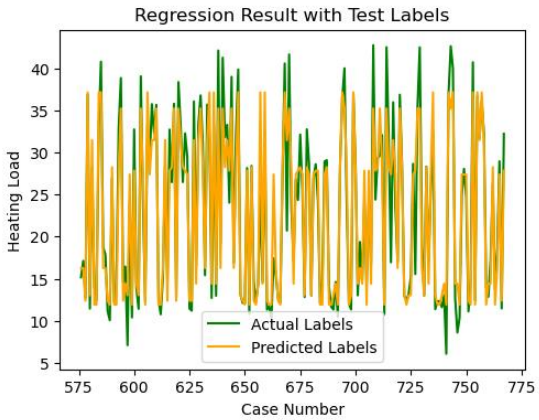
Result 5 (2 are selected) :



Training RMS Error:
3.2815202482584565

Testing RMS Error:
3.2812078773172706





(c) Feature selection

The selection process commences by calculating Spearman correlation coefficients between each individual feature and the target variable, which, in this context, pertains to the Heating Load. Spearman correlation is a non-parametric metric that assesses the monotonic association between two variables, rendering it robust to non-linear relationships. Subsequently, the features are arranged in descending order based on their absolute correlation with the target variable. By limiting the selection to the top four features displaying the most substantial Spearman correlations, the model emphasizes those features that exhibit the strongest monotonic relationships with the target variable.

The efficacy of this feature selection methodology lies in its capacity to discern and prioritize features that exert the most considerable influence on predicting the Heating Load, while mitigating the impact of less informative features. This judicious reduction in feature dimensionality not only minimizes noise but also enhances the model's generalization capability, resulting in the achievement of

the lowest RMS error. I've shown the result in the last paragraph, when the feature number is set to 4, we successfully minimize the error, underscoring the efficacy of this feature selection technique. Spearman correlation-based feature selection, therefore, emerges as a discerning and powerful technique for augmenting the predictive accuracy of the neural network, primarily by accentuating the significance of the most pertinent input features, thereby refining its capacity to accurately predict heating loads. In scenarios characterized by an abundance of features, this approach contributes to a more streamlined and efficient model development process.

2) Classification

(a) Implement the nn for binary classification by using the Ionosphere dataset

Data Preprocessing: The code begins by loading the Ionosphere dataset from a CSV file and performs necessary data preprocessing. It encodes the class labels ('g' and 'b') into binary values (1 and 0), making it suitable for binary classification. The data is then divided into training and testing sets, with an 80% - 20% split.

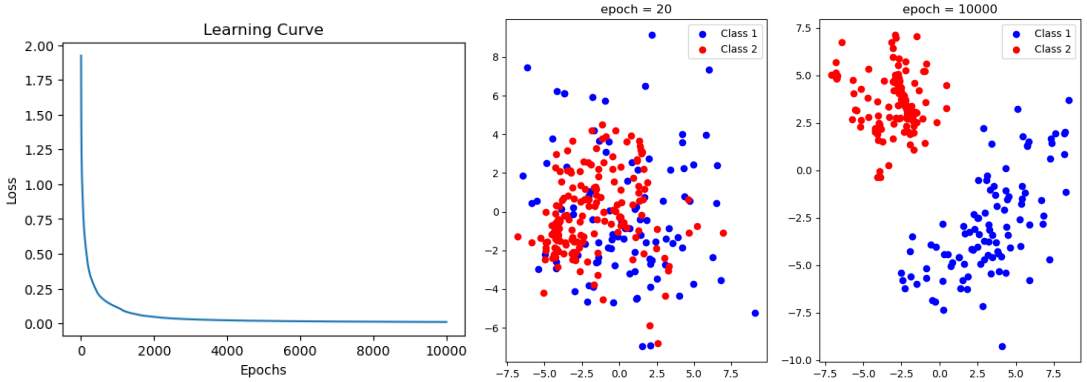
Model Training and Evaluation: The code proceeds to train the neural network using the training data. It collects loss history over 10,000 epochs, employing the cross-entropy loss function and a learning rate of 0.01. After training, the model is used to make predictions on both the training and testing datasets. The predictions are thresholded to binary values (0 or 1) with a threshold of 0.5. The code then calculates and prints the training and testing accuracy.

Neural Network Architecture (Initial): The neural network architecture is defined with four layers. The input layer dynamically adapts to the number of features in the dataset. Three hidden layers follow with 64, 32, and 16 neurons, respectively, both using the hyperbolic tangent (tanh) activation function. The output layer contains two neurons, employing the softmax activation function, which is typical for classification tasks.

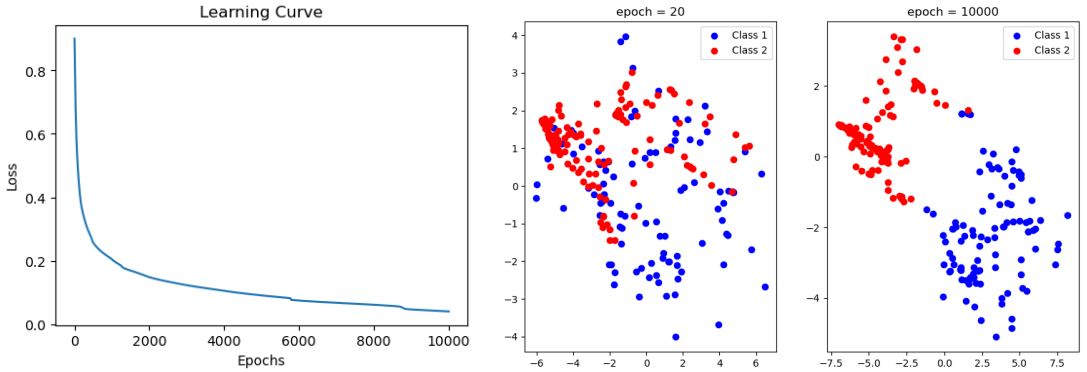
Visualization: The code includes two visualizations. The first shows the learning curve, displaying the loss over epochs, which can help assess the training progress. The second visualization comprises scatter plots of the data points in the second hidden layer (Z2) based on their class labels ('g' or 'b') from the training data. This visualization can offer insights into the model's ability to separate and classify the data.

(b) Show result

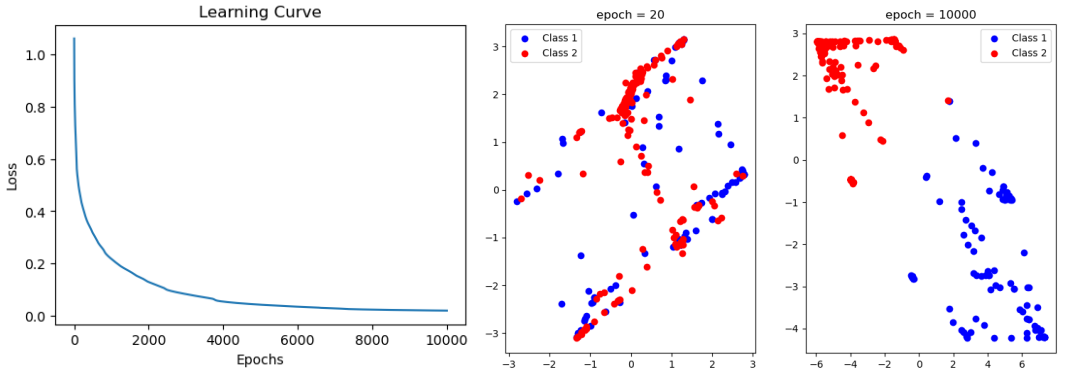
Result 1 (NN layers: 64, 32, 16, 2) :



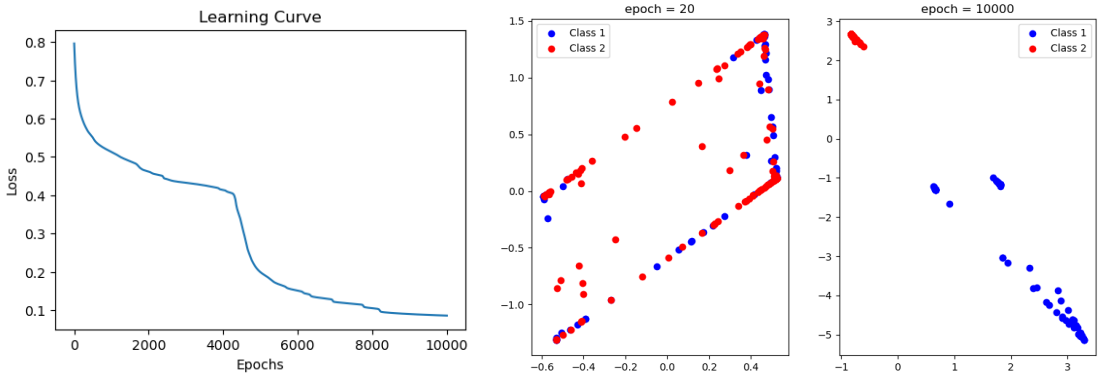
Result 2 (NN layers: 64, 32, 8, 2) :



Result 3 (NN layers: 64, 32, 4, 2) :



Result 4 (NN layers: 64, 32, 2, 2) :



(c) different numbers of nodes in the layer before the output layer

In comparing the results of different neuron counts in the layer preceding the output layer and observing the apparent squeezing of data points when reducing the neuron count to 2, it becomes evident that neural network architecture significantly influences data representation. The reduction in neuron count to 2, essentially constraining the network's expressive capacity, prompts the network to adopt simpler, linear decision boundaries. This behavior aligns with expectations, given that fewer neurons in the layer limit the network's ability to capture complex, non-linear data patterns. The result is a preference for a linear decision boundary, leading to data points clustering along a line with a slope of -1 in a two-dimensional feature space. This underscores the pivotal role of model complexity in shaping data representations and the need to strike the right balance between simplicity and capacity.