Neural Network Training and Accuracy Evaluation for Iris and MNIST Datasets

Course Instructor : Dr. Khaled Mostafa El Sayed
TAs : Eng. Marwa Monier & Eng. Gamal Zayed

Nada Ismail, 202-001-387

1St January, 2024

A Problem definition and motivation

The task at hand is applying Support Vector Machines with different kernels and regularization to classify the six datasets. SVM is a powerful machine learning algorithm used for both classification and regression. In classification, it works by finding the hyperplane that best separates different classes in the feature space. The problem is to explore the performance of SVM with various configurations and assess its ability to accurately classify instances in the Aggregation dataset. Different kernels (linear, polynomial, RBF) have different ways in handling different types of data. By applying SVM with various kernels, we aim to understand how each kernel copes with the unique features of the each dataset. This information is crucial for choosing the most appropriate kernel for datasets with similar complexities. =

B Approach and methodology

B.1 Generating weights

B.1.1 Parameters:

The function takes in three parameters: "num layers" (the total number of layers in the neural network including input and output layers), "input size" (the number of nodes in the input layer), and "num nodes" (the number of nodes in the hidden layers).

B.1.2 Initialization:

VIt initializes an empty list called "weight matrices" to store the weight matrices for each layer of the neural network.

B.1.3 Initial Layer:

It creates the weight matrix for the initial layer by generating a 2D list of zeros with dimensions "num nodes" (rows) by "input size" (columns). This matrix represents the connections between the input layer and the first hidden layer.

B.1.4 Intermediate Layers:

for the intermediate layers. It initializes an intermediate layer matrix by generating a 2D list of zeros with dimensions "num nodes" by "num nodes". This process is repeated for "num layers - 2" times, where each intermediate layer matrix represents the connections between consecutive hidden layers in the neural network.

B.1.5 Return:

The function returns the list of weight matrices representing the connections between nodes in the neural network layers.

B.2 Neural Network Structure Initialization:

B.2.1 Parameters:

The function takes in three parameters: "num neurons hidden" (number of neurons in the hidden layer), "num neurons output" (number of neurons in the output layer), and "num input samples" (number of input samples).

B.2.2 Initialization:

It initializes three 2D arrays: "input nodes", "hidden nodes", and "output nodes". "input nodes" represents the input layer of the neural network and is initialized as a 2D array of zeros with dimensions "num input samples" by 3, indicating the three input features for each input sample. "hidden nodes" represents the hidden layer of the neural network and is initialized as a 2D array of zeros with dimensions "num neurons hidden" by 3, indicating the three features for each neuron in the hidden layer. "output nodes" represents the output layer of the neural network and is initialized as a 2D array of zeros with dimensions "num neurons output" by 3, indicating the three features for each neuron in the output layer.

B.2.3 Return:

The function returns a list containing the initialized input, hidden, and output nodes, encapsulated in a nested list structure.

B.3 Neural Network Forward Propagation

B.3.1 Parameters:

The function takes in two parameters: "weights" (representing the weight matrices for each layer) and "nodes" (representing the nodes in each layer of the neural network).

B.3.2 Process

The function iterates through each layer of the neural network. Within each layer, it iterates through the nodes and calculates the total weighted sum for each node by performing a summation of the products of the input values and corresponding weights connecting to the node. The total weighted sum is then passed through an activation function (here assumed to be sigmoid activation) to produce the output value for the node. The updated node values, including the calculated sum and the output after activation, are stored in the "nodes" data structure.

B.3.3 Return:

The function returns the updated "nodes" data structure after applying the forward propagation process.

B.4 Backpropagation Algorithm for Neural Network Training

B.4.1 Parameters:

The function takes in four parameters: "input nodes" (representing the nodes in each layer of the neural network), "weights" (containing the weight matrices for each layer), "target" (the target output), and "learning rate" (the rate at which the weights are updated during training).

B.4.2 Backpropagation Steps

The function iterates through each layer of the network in reverse order, starting from the output layer and moving towards the input layer. For each node in a layer, it calculates the error signal (delta) based on the difference between the target output and the actual output, multiplied by the derivative of the activation function. It then updates the weights connecting the current layer to the previous layer based on the calculated error signal and the input from the previous layer, scaled by the learning rate. This process continues iteratively through each layer, updating the weights based on the error signals propagated backward through the network.

B.4.3 Return:

The function returns the updated "weights" matrix after applying the backpropagation algorithm.

B.5 Iris Dataset Preprocessing

B.6 Preprocessing and Subsetting of the MNIST Dataset

B.6.1 Concatenation and Shuffling:

B.6.2 Subsetting

To select the first 1000 samples from the shuffled dataset

B.7 Neural Network Training and Accuracy Evaluation

B.7.1 Initialization

initializing the input data, target data, learning rate, the number of layers, and an empty list "y pred list" to store the predicted outputs.

B.7.2 Weight Initialization

initializing the weights for the neural network using the "generate weights" function based on the specified number of layers, input data size, and target data size.

B.7.3 Forward Propagation and Backpropagation

It then iterates through each sample in the input data, performing forward propagation using the initialized weights and input samples. After each forward pass, the backpropagation algorithm is utilized to update the weights based on the calculated error signal and learning rate.

B.7.4 Prediction Generation

The predicted output values are retrieved from the final layer of the neural network and stored in the "y pred list."

B.7.5 Accuracy Evaluation

The predicted output is rounded and stored in a DataFrame. Subsequently, the accuracy of the predictions is calculated using the "accuracy score" function, comparing the predicted output with the actual target data.

B.7.6 Print Accuracy Score

Finally, the accuracy score is printed as a percentage to evaluate the performance of the neural network on the Iris dataset.