Indian Institute of Technology Dharwad

CS214: Artificial Intelligence Laboratory

Lab 7: Neural Network

You are given the Wisconsin Diagnostic Breast Cancer (WDBC) dataset. The dataset features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe the characteristics of the cell nuclei present in the image. A few of the images can be found at http://www.cs.wisc.edu/ street/images/. The separating plane (separating plane in the 3-D space of Worst Area, Worst Smoothness and Mean Texture) was obtained using Multisurface Method-Tree (MSM-T) [1], a classification method which uses linear programming to construct a decision tree. Relevant features were selected using an exhaustive search in the space of 1-4 features and 1-3 separating planes. The actual linear program used to obtain the separating plane in the 3-dimensional space is that described in [2].

The dataset details

- Number of attributes: 31 (Diagnosis, 30 real-valued input features)
- Attribute information
  1. Diagnosis (1 = malignant, 0 = benign) (Attribute 1)
  2. Ten real-valued features are computed for each cell nucleus: (Attribute 2-31)
     (a) radius (mean of distances from center to points on the perimeter)
     (b) texture (standard deviation of gray-scale values)
     (c) perimeter
     (d) area
     (e) smoothness (local variation in radius lengths)
     (f) compactness ($perimeter^2/area - 1.0$)
     (g) concavity (severity of concave portions of the contour)
     (h) concave points (number of concave portions of the contour)
     (i) symmetry
     (j) fractal dimension ("coastline approximation" - 1)

  The mean, standard error, and "worst" or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features. For instance, Attribute 2 is Mean Radius, Attribute 12 is Radius SE, Attribute 22 is Worst Radius.

We are providing you with the prepared datasets (train, validation, and test datasets) in the csv files.

1. Standardized Dataset
   - train data: ***WDBC_Scaled_Train.csv***
   - validation data: ***WDBC_Scaled_Validation.csv***
   - test data: ***WDBC_Scaled_Test.csv***

2. PCA transformed dataset for ten components ($l = 10$)
   - train data: ***WDBC_PCA10_Train.csv***
   - validation data: ***WDBC_PCA10_Validation.csv***
   - test data: ***WDBC_PCA10_Test.csv***

Write a Python program to complete the following tasks.

Part A: Single and Two hidden layer Neural network architecture for Standardized Dataset data

Perform the neural network classification and performance metric evaluation as listed below.

1. Load train, validation, and test data from ***WDBC_Scaled_Train.csv***, ***WDBC_Scaled_Validation.csv***, and ***WDBC_Scaled_Test.csv***, respectively.
2. Implement a neural network for binary classification using `PyTorch` with the following specifications:
   - **Input layer**: 30 neurons (corresponding to the real-valued input features).
   - All neurons after the input layer use the sigmoid activation function.
   - **Hidden layers**: Experiment with different architectures:
     (a) **Single hidden layer** architectures:
       - **Architecture 1**: Input layer → 10 neurons → Output layer
       - **Architecture 2**: Input layer → 100 neurons → Output layer
       - **Architecture 3**: Input layer → 64 neurons → Output layer
       - **Architecture 4**: Input layer → 512 neurons → Output layer

     (b) **Two hidden layer** architectures:
       - **Architecture 1**: Input layer → 64 neurons → 16 neurons → Output layer
       - **Architecture 2**: Input layer → 128 neurons → 32 neurons → Output layer
       - **Architecture 3**: Input layer → 256 neurons → 64 neurons → Output layer
       - **Architecture 4**: Input layer → 512 neurons → 128 neurons → Output layer
   - **Output layer**: 1 neuron (sigmoid activation function).
3. Train the neural network using:
   - **Optimizer**: Stochastic Gradient Descent (SGD)
   - **Loss function**: Sum of Squared Error (SSE)
4. Evaluate and compare the performance metrics (listed below) on validation data and test data.
   - Confusion matrix
   - Accuracy, Precision, Recall, and F1-score
5. Plot a graph for epoch vs loss for training dataset for all architecture.

Part B: Single and Two hidden layer Neural network architecture for PCA transformed dataset for ten components (l = 10) data

Perform the neural network classification and performance metric evaluation as listed below.

1. Load train, validation, and test data from ***WDBC_PCA10_Train.csv***, ***WDBC_PCA10_Validation.csv***, and ***WDBC_PCA10_Test.csv***, respectively.

2. Implement a neural network for binary classification using `PyTorch` with the following specifications:

   - **Input layer**: 30 neurons (corresponding to the real-valued input features).
   - All neurons after the input layer use the sigmoid activation function.
   - **Hidden layers**: Experiment with different architectures:
     (a) **Single hidden layer** architectures:
       - **Architecture 1**: Input layer → 10 neurons → Output layer
       - **Architecture 2**: Input layer → 100 neurons → Output layer
       - **Architecture 3**: Input layer → 64 neurons → Output layer
       - **Architecture 4**: Input layer → 512 neurons → Output layer

     (b) **Two hidden layer** architectures:
       - **Architecture 1**: Input layer → 64 neurons → 16 neurons → Output layer
       - **Architecture 2**: Input layer → 128 neurons → 32 neurons → Output layer
       - **Architecture 3**: Input layer → 256 neurons → 64 neurons → Output layer
       - **Architecture 4**: Input layer → 512 neurons → 128 neurons → Output layer
   - **Output layer**: 1 neuron (sigmoid activation function).

3. Train the neural network using:

   - **Optimizer**: Stochastic Gradient Descent (SGD)
   - **Loss function**: Sum of Squared Error (SSE)

4. Evaluate and compare the performance metrics (listed below) on validation data and test data.

   - Confusion matrix
   - Accuracy, Precision, Recall, and F1-score

5. Plot a graph for epoch vs loss for training dataset for all architecture.

Part C: Comparision and Summarization of accuracy for all datasets and classification algorithms on validation and test data.

In the end, summarize the accuracy in the form of table and a bar chart with proper labeling. The sample code is also shown below.

```
# Define the table data
data = {
    "Dataset": [ "Standardized", "PCA (l=10)"],
    "Single hidden layer model 1": [ 100, 100],
    "Single hiddne layer model 2": [ 100, 100],
    "Single hidden layer model 3": [ 100, 100],
    "Single hidden layer model 4": [ 100, 100],
    "Two hidden layer model 1": [ 100, 100],
    "Two hiddne layer model 2": [ 100, 100],
    "Two hidden layer model 3": [ 100, 100],
    "Two hidden layer model 4": [ 100, 100],


}
```

Note

1. Create a separate Jupyter Notebook for each part.
2. For Neural Network, use the below code from *PyTorch*:

```
import torch.nn as nn
import torch.optim as optim

class NeuralNet(nn.Module):
    def __init__(self, input_size, hidden1, hidden2):
        super().__init__()
        self.layers = nn.Sequential(
            nn.Linear(input_size, hidden1),
            nn.Sigmoid(),
            nn.Linear(hidden1, hidden2),
            nn.Sigmoid(),
            nn.Linear(hidden2, 1),
            nn.Sigmoid()
        )

    def forward(self, x):
        return self.layers(x)
```

```
model = NeuralNet(input_size=30, hidden1=128, hidden2=32)
criterion = nn.MSELoss()  # SSE Loss
optimizer = optim.SGD(model.parameters(), lr=0.01)  # SGD Optimizer
```

Please upload the completed Jupyter Notebook. Copy all Jupyter Notebook and data files in a folder and rename the folder as ***your_rollnumber_Lab7***, then create the zip file with the name ***your_rollnumber_Lab7.zip***. Finally, upload the zip file on the Moodle for evaluation.

References

[1] K. P. Bennett, "Decision Tree Construction Via Linear Programming." Proceedings of the 4th Midwest Artificial Intelligence and Cognitive Science Society, pp. 97-101, 1992

[2] K. P. Bennett and O. L. Mangasarian: "Robust Linear Programming Discrimination of Two Linearly Inseparable Sets", Optimization Methods and Software 1, 1992, 23-34