Middle East Technical University Department of Computer Engineering

CENG 462

Artificial Intelligence

Fall '2022-2023

Homework 6

Due date: 21 January 2023, Saturday, 23:55

1 Objectives

This assignment aims to familiarize you with the implementation of the backpropagation algorithm and enable you to gain hands-on experience in classification problems with this algorithm.

2 Problem Definition

Multilayer-Perceptrons (feed-forward neural networks) are universal function approximators, that is to say, they can approximate any given function ($y = f(x)$, functions to be approximated have to comply with certain requirements, e.g continuous, bounded functions) arbitrarily closely provided that MLP has sufficiently many nodes and layers. In addition, they are able to do feature engineering (they find the most important features automatically for the problem) at their first layer and provide generalization capabilities (e.g they can interpolate their results for the new unseen samples).

Basically, we want a neural network to learn any information that we provide to it and make guesses for the never-provided information. Hopefully, it yields correct results for this unknown information. To assess whether the network learns or not, we can easily check the output of the network and the ground truth for the information. So basically we want it to output values/guesses as closely as possible to the ground truth. We can define a function that measures the closeness between the truth and the predicted values by the neural network. If they are close, the function can generate lower values otherwise higher values. So we can cast this neural network learning problem as a function optimization problem. With this respective, we alter the network weights in such a way that a particular function is minimized. Here minimization corresponds to bringing the network outputs as close as to the actual truth. From the calculus, we simply know that to minimize a function we should take steps in the reverse direction of the gradient vector. The backpropagation algorithm basically applies this logic, it is the direct application of the gradient-descent method for minimizing particular functions.

3 Implementation

For the implementation of the backpropagation learning algorithm 1. Here the source code for the classification. The codes already read datasets and provide a very basic implementation of the stochastic gradient descent method on the problems. You are expected to fill the algorithmic gaps in the source codes to complete the backpropagation algorithm.

function BACK-PROP-LEARNING(*examples*, *network*) **returns** a neural network
  inputs: *examples*, a set of examples, each with input vector **x** and output vector **y**
    network, a multilayer network with L layers, weights $w_{i,j}$, activation function g
  local variables: Δ, a vector of errors, indexed by network node

  repeat
    for each weight $w_{i,j}$ **in** *network* **do**
      $w_{i,j} \leftarrow$ a small random number
    for each example $(\mathbf{x}, \mathbf{y})$ **in** *examples* **do**
      /* *Propagate the inputs forward to compute the outputs* */
      for each node i in the input layer **do**
        $a_i \leftarrow x_i$
      for $\ell = 2$ **to** L **do**
        for each node j in layer ℓ **do**
          $in_j \leftarrow \sum_i w_{i,j}\, a_i$
          $a_j \leftarrow g(in_j)$
      /* *Propagate deltas backward from output layer to input layer* */
      for each node j in the output layer **do**
        $\Delta[j] \leftarrow g'(in_j) \times (y_j - a_j)$
      for $\ell = L - 1$ **to** 1 **do**
        for each node i in layer ℓ **do**
          $\Delta[i] \leftarrow g'(in_i) \sum_j w_{i,j}\, \Delta[j]$
      /* *Update every weight in network using deltas* */
      for each weight $w_{i,j}$ **in** *network* **do**
        $w_{i,j} \leftarrow w_{i,j} + \alpha \times a_i \times \Delta[j]$
  until some stopping criterion is satisfied
  return *network*

Figure 18.24 The back-propagation algorithm for learning in multilayer networks.

Figure 1: The back-propagation algorithm for learning in multilayer networks.

```
class Network:
    Create Directed Acyclic Network of given number layers.



def BackPropagationLearner(X,y, net, learning_rate, epochs):



    # initialize each weight with the values min_value=-0.5, max_value=0.5,

    for epoch in range(epochs):
    # Iterate over each example

      # Activate input layer

      # Forward pass

      # Error for the MSE cost function

      # The activation function used is sigmoid function

      # Backward pass

```

```
        #  Update weights



    return net



def NeuralNetLearner(X,y, hidden_layer_sizes=None, learning_rate=0.01, epochs
    =100):
    """
    Layered feed-forward network.
    hidden_layer_sizes: List of number of hidden units per hidden layer if None
     set 3
    learning_rate: Learning rate of gradient descent
    epochs: Number of passes over the dataset
    activation:sigmoid
  """



    # construct a raw network and call BackPropagationLearner

    def predict(example):
        # activate input layer


        # forward pass


        # find the max node from output nodes
        return prediction

    return predict

from sklearn import datasets

iris = datasets.load_iris()
X = iris.data
y = iris.target

nNL = NeuralNetLearner(X,y)
print(nNL([4.6, 3.1, 1.5, 0.2])) #0
print(nNL([6.5, 3. , 5.2, 2. ])) #2
```

4 Specifications

- You are expected to complete the missing parts for the backpropagation algorithm in the source codes.
- You may utilize the `numpy` package as well as other numerical/scientific calculation libraries.
- You may represent all weights as numpy arrays and perform vector/matrix operations such as addition, multiplication, etc. Or you may keep each weight in a separate class variable and perform operations individually on them.

- Commenting is crucial for understanding your implementation and decisions made during the process. Your implementation is going to be inspected manually.

5 Regulations

1. **Late Submission:** No late submission is allowed. Since we have a strict policy on submissions of homework to be able to attend the final exam, please pay close attention to the deadlines.
2. **Cheating: We have zero-tolerance policy for cheating**. People involved in cheating will be punished according to university regulations.
3. **Discussion:** You must follow ODTUClass for discussions and possible updates on a daily basis. If you think that your question concerns everyone, please ask them on ODTUClass.
4. **Evaluation:** Your assignment is going to be graded manually.

6 Submission

Submission will be done via the ODTUClass system. You should upload completed versions of **ClassificationMLP.py**.

References

[1] `https://archive.ics.uci.edu/ml/datasets/Iris`

[2] Announcements Page

[3] Discussions Page