HCMUTE

ARTIFICIAL NEURAL NETWORK

in Python LANGUAGE

Chapter 2.3: Loss & Accuracy

2.3. Loss & Accuracy

- **Role of Loss function (cost function):**
  - Allow the estimation of the Network’s error, i.e. how wrong the model is.
  - Ideally, a perfect model has a loss function equal to 0.
  - Generally we have 2 principal types of loss function: MSE (mean squared error loss) for regression applications & Categorical Cross Entropy Loss for classification / identification

2.3. Loss & Accuracy

Categorical Cross Entropy Loss:

Categorical Cross Entropy Loss is used with Softmax output. The formula for the determination of the function is as follow:

<u>CCE Loss function:</u>

$$L = \sum_{i} y_i \log\left(\hat{y}_i\right)$$

y: actual distribution
ŷ: predicted distribution

<u>Example:</u>

$$y = [1 \quad 0 \quad 0]$$
$$\hat{y}(\text{softmax}) = [0{,}7 \quad 0{,}2 \quad 0{,}1]$$

$$L = 1.\log(0{,}7) + 0.\log(0{,}2) + 0.\log(0{,}1) = 0{,}36$$

2.3. Loss & Accuracy

- **Categorical Cross Entropy Loss:**

```
# Class Cross Entropy Loss

import numpy as np
import math

# Loss Class
class Loss:
    # Calculate the data and regulazation losses given
    # the output and the ground truth values
    def calculate(self,output,y):
        #Calculate sample losses:
        sample_losses = self.forward(output,y)
        #Calculate mean loss:
        data_loss = np.mean(sample_losses)
        #Return loss:
        return data_loss
```

```
class Loss_CategoricalCrossentropy(Loss):
    # Forward Pass
    def forward(self, y_pred, y_true):
        samples = len(y_pred)
        y_pred_clipped = np.clip(y_pred,1e-7,1-1e-7)  #1e-7: avoid division by 0
        # Probabilities for target values only if categorical labels
        if len(y_true.shape) == 1: # The label array is 1D [0 0 ... 1 ... 0 0]
            correct_confidence = y_pred_clipped[range(samples), y_true]
        elif len(y_true.shape) == 2:
            # The label array is 2D [[0 0 ... 1 ... 0 0] [0 0 ... 1 ... 0 0] …]
            correct_confidence = np.sum( y_pred_clipped*y_true, axis = 1)
        # Losses calculation
        negative_log_likelihoods = -np.log(correct_confidence)
        return negative_log_likelihoods
```

2.3. Loss & Accuracy

- **Categorical Cross Entropy Loss:**

Code example:

```
#Example Loss Function

import numpy as np
import matplotlib.pyplot as plt
import Loss as loss # type: ignore
# Test the class now:
softmax_output = np.array([[0.7, 0.1, 0.2],
                                       [0.1, 0.5, 0.4],
                                       [0.02, 0.9, 0.08]])
#class_target = np.array([0, 1, 1])
class_target = np.array([[1, 0, 0],
                         [0, 1, 0],
                        [0, 1, 0]])
loss_function = loss.Loss_CategoricalCrossentropy()
loss_CCE = loss_function.calculate(softmax_output,class_target)
print(loss_CCE)
```

2.3. Loss & Accuracy

- **Mean Square Error Loss:**

Mean Square Error Loss is used generally for outputs of regression applications (or prediction applications). The formula for the determination of the function is as follow:

MSE Loss function:

$$L = \frac{1}{N_{out}} \sum_{i} \left(\hat{y}_i - y_i \right)^2$$

y: actual distribution
ŷ: predicted distribution

2.3. Loss & Accuracy

- **Accuracy:**

For the classification problem, one simple way to compute the accuracy is to compare the softmax output vector (using **argmax** function) with the label vector. An example of code is as follow:

```
import numpy as np
    # 3 samples outputs
    softmax_outputs = np.array([[0.7, 0.2, 0.1],
                                [0.5, 0.1, 0.4],
                                [0.02, 0.9, 0.08]])
    #Target (Ground truth labels) for the 3 samples
    class_targets = np.array([0, 1, 1])
    #Calculate max values indices along 2nd the samples
    predictions = np.argmax(softmax_outputs, axis = 1)
    print(predictions)

    if (len(class_targets.shape) == 2):
    class_targets = np.argmax(class_targets,axis=1)
    # Calculate the accuracy (True = 1, False = 0)
    accuracy = np.mean(predictions == class_targets)
    print("Accuracy: ", accuracy)
```

Artificial Neural Network

END OF CHAPTER 2 – Part 3