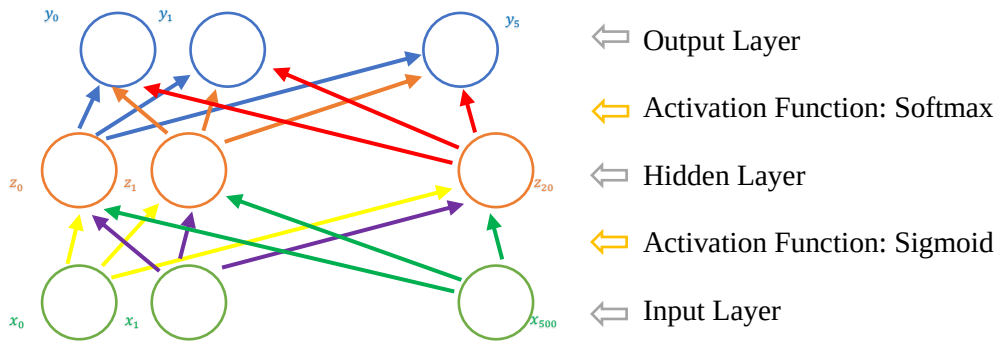
ENGR 421

Homework 03: Multilayer Perceptron for Multiclass Discrimination

Deadline: October 31, 2019, 11:59 PM

Sitare Arslantürk, 57677

In this assignment, a multiclass discrimination is implemented via multilayer perceptron.



The input layer, hidden layer and output layer consists of 500, 20 and 5 nodes, respectively. The multilayer perceptron for multiclass discrimination is constructed by the sigmoid activation function for twenty nodes in the hidden layer and the softmax activation function for five nodes in the output layer.

The given formulas are applied in the algorithm step by step following the following recipe:

1- W, V and eta are initialized
2- The gradients of ΔW, ΔV, $\Delta W0$, $\Delta V0$ are calculated
3- W, V, W0, V0 are updated
4- Go to step 2, if there is any change in the parameters until the max_iteration is reached, STOP

The following functions are implemented in the code:

Let's give $A = (V_c^T * z_i + V_0)$ where V_c^T is the weight parameter between output and hidden layer and V_0 represents the bias for the first node, z_i.

$$\hat{y}_{ic} = softmax(A) = \frac{\exp(A)}{\sum_{i=1}^{K} \exp(A)}$$

Let's give $B = \left(W_h^T * x_i + W_0\right)$ where W_h^T is the weight parameter between input and hidden layer and W_0 represents the bias for the first node, x_i.

$$z_{ih} = sigmoid(B) = \frac{1}{1 + \exp(-B)}$$

Thus, the error function for i is given by

$$Error_i = -\sum_{c=1}^{K} y_{ic} * \log(\hat{y}_{ic})$$

The derivative of the error function is taken with respect to V, V_0, W and W_0 in order to find the gradient descent. The finalized update rules are the following:

$$\Delta V_{ch} = eta * \sum_{i=1}^{N} (y_{ic} - \hat{y}_{ic}) * z_{ih}$$

$$\Delta V_0 = eta * (y_{0c} - \hat{y}_{0c})$$

$$\Delta W_{hd} = eta * \sum_{i=1}^{N} \left[\sum_{c=1}^{K} (y_{ic} - \hat{y}_{ic}) * V_{ch} \right] * z_{ih} * (1 - z_{ih}) * x_{id}$$

$$\Delta W_0 = eta * \left[\sum_{c=1}^{K} (y_{0c} - \hat{y}_{0c}) * V_{ch} \right] * z_{0h} * (1 - z_{0h})$$

The confusion matrix is printed with the help of sklearn metrics library and to plot the graph, I used matplotlib. The following are the pseudocodes of the fit, forward and prediction functions of the Perceptron class. The gradient descent is calculated via the backpropagation steps mentioned above.

```
def fit(X, Y, W, W0, V, V0):
      costs = []
      for i in max_iteration:
            output, hidden = forward(X, Y, W, W0, V, V0)
            c = cost(T, output)
            costs.append(c)
            V += ΔV
            W += ΔW
            V0 += ΔV0
            W0 += ΔW0

def forward(X, Y, W, W0, V, V0):
      Z = sigmoid(X.dot(V)+W0)
      Y = softmax(Z.dot(V)+V0)
      return Y, Z

def predict(X, W, W0, V, V0):
      return argmax(forward(X, Y, W, W0, V, V0)[0])
```