Deniz ALKISLAR / 21502930

Question 1

We first need to define the formulas for forward propagation. By propagating forward, our model generates a prediction about our classes.

$\alpha^L = \varrho(W^L \alpha^{L-1} + b^L)$ is the formula for computing the activations. α^L is the activation of layer L, ϱ is the activation function, W^L is the weights of layer L, α^{L-1} is the activations of layer L – 1 and b^L is the bias for layer L.

By using the formula, we can create our neural network as follows:

Input layer(X, N) => $\alpha^1 = \varrho(X + b^1)$
First hidden layer(N, N') => $\alpha^2 = \varrho(W^2\alpha^1 + b^2)$
Second hidden layer(N', N'') => $\alpha^3 = \varrho(W^3\alpha^2 + b^3)$
Output layer(N'', 2) => $\alpha^4 = \varrho^*(W^4\alpha^3 + b^4)$

$\varrho(x)$ = ReLU = max(0, x)
$\varrho^*(x)$ = Sigmoid = $1 / (1 + e^{-x})$

The last step is backward propagation. This step is used for recalculating the weights. The model evaluates its predictions based on a cost function. Then by calculating partial derivative of the cost function with respect to weights, we basically find how much a change in weights would affect our predictions.

As a cost function binary cross entropy(BCE) would be a great choice since we have two classes.

$$BCE = -\frac{1}{N}\sum_{i=0}^{N} y_i \cdot log(\hat{y}_i) + (1 - y_i) \cdot log(1 - \hat{y}_i)$$

Where y is the actual label and y(hat) is the predicted label.

For backward propagation we define

$z^L = \sum W^L a^{(L-1)} + b^L$ to make calculations easier.

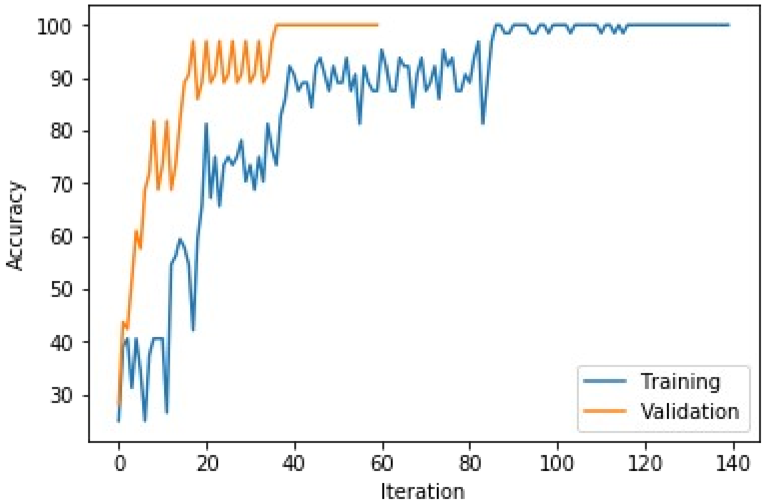
The last step is to find partial derivative of the cost function with respect to weights. Here we use chain rule:

$$\frac{dBCE}{dW^L} = \frac{dBCE}{da^L}\frac{da^L}{dz^L}\frac{dz^L}{dW^L}$$

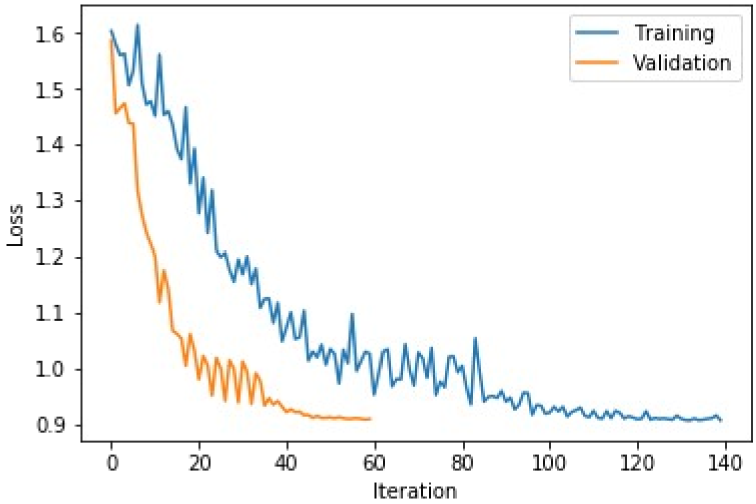
If we take respective derivatives in the formula, we get:

$$\frac{dBCE}{dW^L} = \left(\frac{dBCE}{da^{(L+1)}} * W^{(L+1)}\right)\left(\varrho'(z^L)\right)\left(a^{(L-1)}\right)$$

Question 2



Graph1. Accuracy plot for training and validation iterations



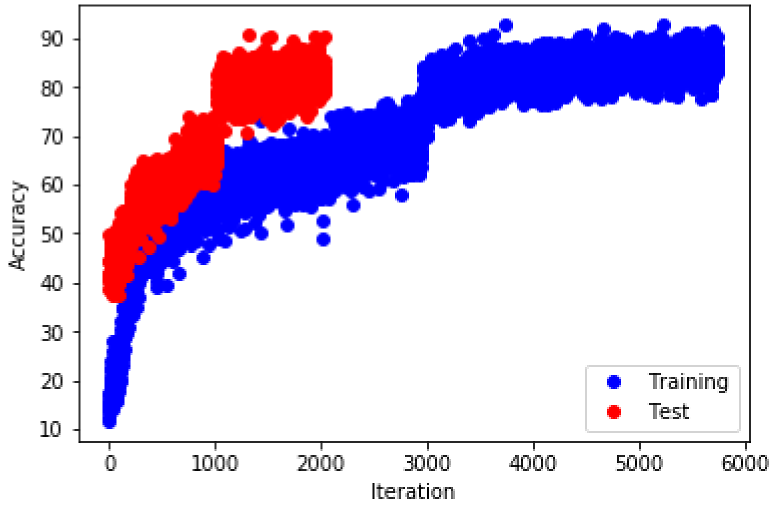
Graph2. Loss plot for training and validation iterations

Confusion matrix for the test set:

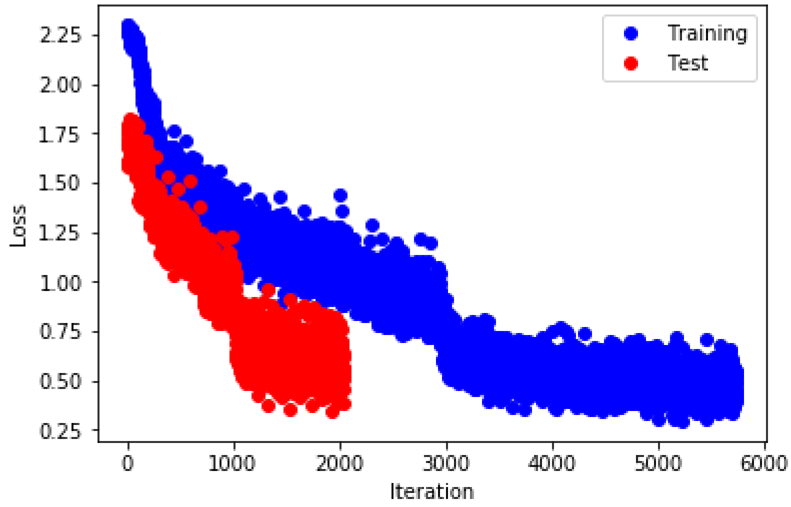
```
[[37.  0.  0.  0.  0.]
 [ 0. 43.  0.  0.  0.]
 [ 0.  0. 92.  0.  0.]
 [ 0.  0.  0. 40.  0.]
 [ 0.  1.  0.  0. 27.]]
```

Accuracy: 100%

Question 3



Graph3. Accuracy plot for training and validation iterations of MLP



Graph4. Loss plot for training and validation iterations of MLP

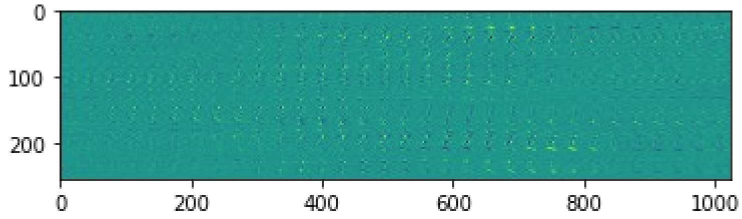


Image1. Weight visualization for MLP