**HW to Chapter 4 & 5 “Neural Network with one hidden layer”**

**Non-programming Assignment:**

**1.What is the Hadamard matrix product?**

The Hadamard product of two matrices is the element-wise multiplication of the corresponding components of the matrices. To compute the Hadamard product, both matrices must be of the same size, and the resultant matrix will also be of the same size​.

**2.Describe matrix multiplication.**

Matrix multiplication, or the dot product, is not elementwise. Instead, the number of columns in the first matrix must be equal to the number of rows in the second matrix. The product of a matrix AAA of size N×MN \times MN×M and a matrix BBB of size M×KM \times KM×K will be a matrix of size N×KN \times KN×K. The resulting elements are computed as the sum of the products of the corresponding elements in the row of the first matrix and the column of the second matrix​.

**3.What is transpose matrix and vector?**

A transpose of a matrix AAA of size N×MN \times MN×M is a matrix ATA^TAT of size M×NM \times NM×N, where each element is flipped across its diagonal. For example, the element at position (i,j)(i, j)(i,j) in matrix AAA is placed at position (j,i)(j, i)(j,i) in ATA^TAT. Similarly, a transpose of a vector (which is a 1-dimensional matrix) swaps its orientation from column to row or vice versa​.

**4.Describe the training set batch.**

In neural network training, a batch refers to a subset of the training data processed at a single time during one forward and backward pass. It helps to improve computational efficiency and stability in gradient descent by averaging over multiple training samples. A batch can consist of one or more samples from the training set, and this helps in estimating the gradient more accurately compared to using a single sample.

**5.Describe the entropy-based loss (cost or error) function and explain why it is used for training neural networks.**

The entropy-based loss function, specifically the cross-entropy or log-loss, is used as a cost function in training neural networks for classification. It measures the difference between the predicted probability distribution (from the neural network) and the true distribution (actual labels). The cross-entropy function is convex, which ensures that the gradient descent algorithm can converge to a global minimum, making it efficient for optimization. It is widely used because it effectively penalizes incorrect classifications with greater weights when the confidence in the prediction is high​.

**6.Describe the neural network supervised training process.**

Supervised training involves feeding a neural network input data with corresponding correct outputs (labels) and allowing it to adjust its weights and biases through a process called backpropagation. The training steps include:

**Forward Propagation**: Compute the output of the network by passing inputs through the layers of the neural network.

**Loss Calculation**: Calculate the difference (error) between the predicted outputs and the actual outputs using a loss function (e.g., cross-entropy).

**Backward Propagation**: Use the error calculated to update the weights and biases of the network by computing the gradients (derivatives) of the loss function with respect to each parameter.

**Optimization**: Apply gradient descent or another optimization technique to minimize the loss by updating the parameters in the direction that decreases the loss. This process is repeated over several epochs until the network's performance improves and reaches a satisfactory level of accuracy.

**7.Describe in detail forward propagation and backpropagation.**

**Forward Propagation**:

In forward propagation, the input data is fed through the neural network layer by layer.

Each layer performs a weighted sum of inputs, adds a bias, and then applies an activation function to generate the output, which is passed to the next layer.

This continues until the output layer generates the network's final prediction.

Mathematically, it can be represented as:

Aggregated signal: Z=WX+bZ = WX + bZ=WX+b

Activation: A=f(Z)A = f(Z)A=f(Z), where fff is the activation function (e.g., ReLU, sigmoid).

**Backpropagation**:

Backpropagation is used to compute the gradient of the loss function with respect to each weight and bias in the network.

It begins by calculating the error at the output layer and then propagates this error backward through the network.

The gradients of the loss with respect to the parameters are computed layer by layer using the chain rule of calculus.

These gradients are then used to update the weights and biases via gradient descent:

For weights: W:=W−rδWW := W - r \delta WW:=W−rδW

For biases: b:=b−rδbb := b - r \delta bb:=b−rδb

This process is repeated iteratively to minimize the loss function, effectively training the network to produce better predictions.