In lectures 1 and 2, we discussed about the concepts of back-propagation and stochastic gradient descent. In this homework, you will be implementing a two-layered neural network using those concepts. This will help you in getting a basic understanding of neural network.

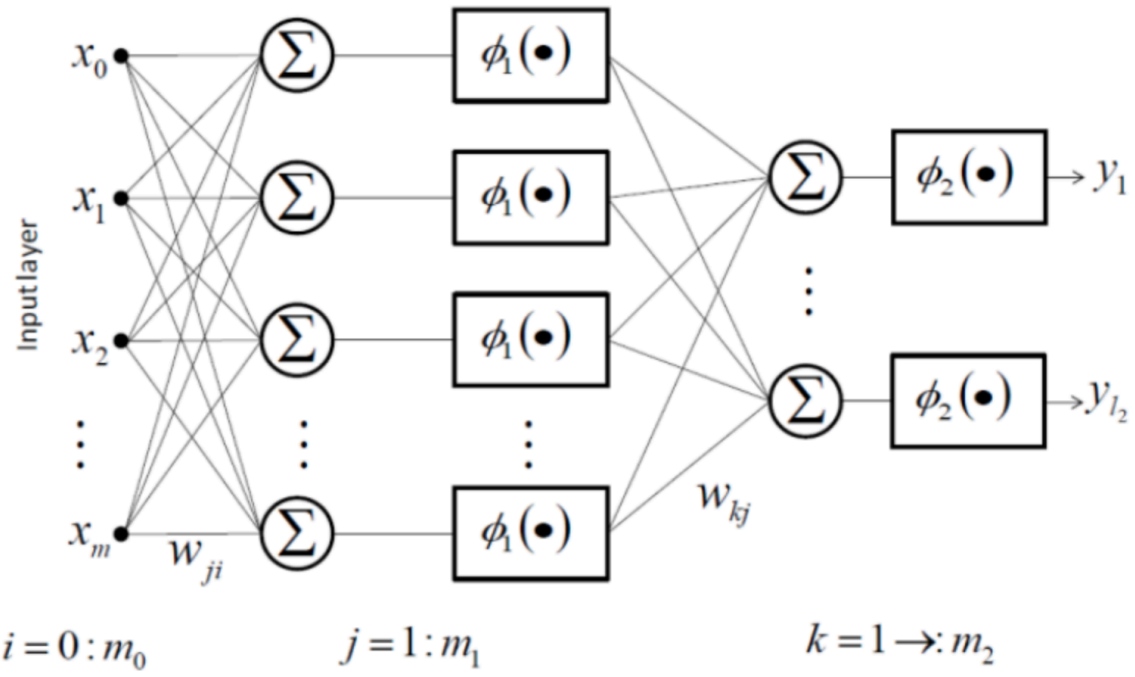


*Figure 1: two-layered neural network*

In figure 1, a two-layered neural network is shown. For this homework, you are given MNIST dataset consisting of hand written digits as images with shape (28,28). Since each image should be given as input to neural network, we convert to the shape (1,784). Therefore, $m_0 = 783$, $m_1$ can be any value (say 30) and finally $m_2$ will be 10 since we are dealing with numbers 0 through 9. The data has 60000 training images and 10000 test images. Assume $\phi_1$ as 'Rectified linear unit (ReLu)' and $\phi_2$ as 'softmax' whose equations are given by:

$$\phi_1(x) = x^+ = \max(0, x)$$

$$and$$

$$\phi_2(x_j) = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}} \; for\; j = 1, \ldots, k \; where\; k \; is \; number \; of \; classes$$

In general, the loss/error function that is accompanied with softmax is cross entropy function which is given by:

$$loss\ (J\ ) = \sum_{k=1}^{K} \frac{t(k)\log\big(y(k)\big)}{N} where\ t\ is\ binary\ value\ and\ N\ is\ number\ of\ samples$$

You will learn about these functions in detail later in the course. For now, you need to understand that ReLu removes all negative values and softmax gives probabilities for each class as output.

Examples for ReLu, softmax and loss are given below:

ReLu:

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.24254239 | -0.0093585 | 0.06251788 | | 0.24254239 | 0 | 0.06251788 |
| 0.29174967 | 0.00640839 | 0.0088702 | | 0.29174967 | 0.00640839 | 0.06251788 |
| 0.21299534 | 0.01685097 | -0.01915576 | max $(0, x)$ | 0.21299534 | 0.01685097 | 0 |
| 0.28040822 | 0.08560704 | -0.20277768 | ----------> | 0.28040822 | 0.08560704 | 0 |
| 0.20900069 | 0.05125724 | -0.23243211 | | 0.20900069 | 0.05125724 | 0 |

*Table 1: ReLu conversion*

Softmax**:**
If there are 10 classes (0-9) whose output-layer inputs have following values:

```
[-1.70250359, -4.2325084, -4.32629959, 5.89011416, -5.86517858,
10.57400549, -3.038615, -8.27508649, 6.23970273, 2.72117404]
```

Then the output of softmax output $\phi_2$ will be:

```
[4.55628258e-06, 3.62947261e-07, 3.30453631e-07, 9.03738205e-03,
7.09224737e-08, 9.77756708e-01, 1.19769115e-06, 6.37050950e-09,
1.28193803e-02, 3.80005111e-04]
```

You can check that the summation of all values in output is 1.

Loss ( $J$ ):
Consider a batch size ($b$) of 3 and the softmax output $\emptyset_2(x)$ be:

```
[[ 4.55628258e-06,   3.62947261e-07,   3.30453631e-07,
    9.03738205e-03,   7.09224737e-08,   9.77756708e-01,
    1.19769115e-06,   6.37050950e-09,   1.28193803e-02,
    3.80005111e-04],
 [  1.02072520e-03,   2.07443268e-02,   1.36165460e-04,
    1.31895168e-01,   7.33937145e-04,   2.33243538e-01,
    2.03292669e-03,   3.39410794e-04,   5.27287683e-01,
    8.25661187e-02],
 [  5.19327780e-02,   1.89228826e-03,   1.90638660e-02,
    2.89810025e-01,   4.21878509e-02,   4.39552262e-01,
    1.67685711e-02,   5.51353408e-04,   1.38094898e-01,
    1.46107555e-04]]
```

and corresponding targets be `[6, 4, 7]`
Then loss ( $J$ ) will be calculated as:
`sum(log(1.19769115e-06,7.33937145e,04+5.51353408e-04))/3 = 9.45`

The back propagation requires finding derivatives of each layer and traversing backwards. For your convenience, we are providing the softmax cross-entropy derivatives which you can use directly.

$$\frac{d\,J}{d\,x} = \frac{d\,J}{d\,\emptyset_2(x)} * \frac{d\,\emptyset_2(x)}{d\,x}$$

$$\frac{d\,J}{d\,x} = \begin{cases} (\emptyset_2(x)_l - 1)\ for\ l = t \\ (\emptyset_2(x)_l)\ for\ l \neq t \end{cases}$$

To get better understanding, consider $\emptyset_2(x)$ which we used for loss. The 6th value of 1st array,4th value of 2nd array and 7th value of 3rd array will be subtracted by 1. Then the entire array will be divided by 3 to get the average of three samples. Therefore the $\frac{d\,J}{d\,x}$ values will be:

```
[[  1.51876086e-06,   1.20982420e-07,   1.10151210e-07,
    3.01246068e-03,   2.36408246e-08,   3.25918903e-01,
   -3.33332934e-01,   2.12350317e-09,   4.27312677e-03,
    1.26668370e-04],
 [  3.40241732e-04,   6.91477560e-03,   4.53884867e-05,
    4.39650558e-02,  -3.33088688e-01,   7.77478461e-02,
    6.77642230e-04,   1.13136931e-04,   1.75762561e-01,
    2.75220396e-02],
 [  1.73109260e-02,   6.30762753e-04,   6.35462201e-03,
    9.66033417e-02,   1.40626170e-02,   1.46517421e-01,
    5.58952372e-03,  -3.33149549e-01,   4.60316327e-02,
    4.87025185e-05]]
```

Lastly the derivative of ReLu is given as: $\emptyset_1^{|}(x) = \begin{cases} 0\ if\ x \leq 0 \\ 1\ if\ x > 0 \end{cases}$

The rest of derivatives can easily be computed using chain rule.