High-Level Computer Vision
Summer Semester 2019

Assignment 2: Deep Neural Networks and Back propagation

Group Members:

- **Chirag Bhuvaneshwara - 2571703 - chiragbhuvaneshwar@gmail.com**
- **Jyotsna Singh - 2576744 - s8jysing@stud.uni-saarland.de**
- **Jyothsna Shashikumar Sastry - 2571729 - s8jysast@stud.uni-saarland.de**

Q1)

Difference between your scores and correct scores:
2.91734116586e-08
Difference between your loss and correct loss:
1.79412040779e-13

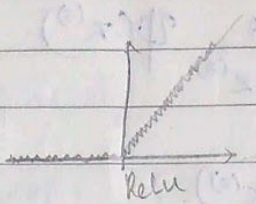
Q2)

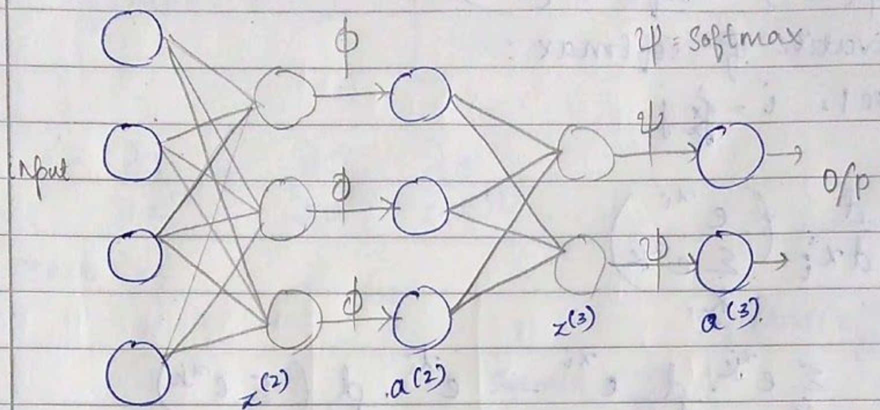
W2 max relative error: 3.440708e-09
b2 max relative error: 3.865070e-11
W1 max relative error: 3.561318e-09
b1 max relative error: 1.555470e-09
Final training loss: 0.015543351766

Group :
Chirag Bhuvaneshwara - 2571703
Jyothsna Sastry - 2571729
Jyothsna Singh - 2576744

2) Network Architecture

input image





$$a^{(1)} = x$$
$$z^{(2)} = W^{(1)} a^{(1)} + b^{(1)}$$
$$a^{(2)} = \phi(z^{(2)})$$
$$z^{(3)} = a^{(2)} . W^{(2)} + b^{(2)}$$
$$f_\theta(x) : a^{(3)} = \psi(z^{(3)})$$

$$J(\theta) = \frac{1}{N} \sum_{i=1}^{N} -\log \left[\frac{e^{z^{(3)}_{y_i}}}{\sum_j e^{z^{(3)}_j}} \right]$$

(a) $$\frac{\partial J}{\partial z^{(3)}} = \frac{\partial J}{\partial a^{(3)}} \cdot \frac{\partial a^{(3)}}{\partial z^{(3)}}$$

consider $$\frac{\partial J}{\partial a^{(3)}} \cdot \frac{\partial a^{(3)}}{\partial z^{(3)}} = \frac{\partial}{\partial a^{(3)}} \left[\frac{1}{N} \sum_{i=1}^{N} -\log a^{(3)} \right]$$

$$= \frac{-1}{N} * \sum_{i=1}^{N} \frac{1}{a^{(3)}} \cdot \frac{\partial a^{(3)}}{\partial z^{(3)}}$$

$$= \frac{-1}{N} \sum_{i=1}^{N} \frac{1}{a^{(3)}} \cdot \frac{\partial a^{(3)}}{\partial z^{(3)}}$$

Consider $\dfrac{\partial a^{(3)}}{\partial (z^{(3)})} = \dfrac{\partial}{\partial z^{(3)}} \psi(z^{(3)})$

$\psi(z^{(3)}) = \text{Softmax}(z^{(3)})$

Derivative of softmax:

case 1: $i = j$

$$\frac{d}{dx_i}\left(\frac{e^{x_i}}{\sum_k e^{x_k}}\right)$$

$$= \frac{\sum_k e^{x_k} \cdot \frac{d}{dx_i} e^{x_i} - e^{x_i} \cdot \frac{d}{dx_i}\left(\sum_k e^{x_k}\right)}{\left(\sum_k e^{x_k}\right)^2} \quad \text{--- (1)}$$

$$= \frac{\sum e^{x_j} e^{x_i} - e^{x_i} e^{x_i}}{\left(\sum_k e^{x_k}\right)^2}$$

$$= \frac{e^{x_i}}{\sum e^{x_k}} \left(\frac{\sum e^{x_j} - e^{x_i}}{\sum e^{x_k}}\right)$$

$$= S_i(1 - S_i)$$

case 2: $i \neq j$

substituting in (1)

$$= \frac{0 - e^{x_i} \cdot e^{x_j}}{\left(\sum_k e^{x_k}\right)^2}$$

$$= -\frac{e^{x_i}}{\sum_k e^{x_k}} \cdot \frac{e^{x_j}}{\sum_k e^{x_k}}$$

$$= -S_i S_j$$

$$\text{Softmax}(z^{(3)}) = \begin{cases} s_i(1-s_i) & \text{for } i=j \\ -s_i s_j & \text{for } i \neq j \end{cases}$$

$$\therefore \frac{\partial J}{\partial z^{(3)}} = \frac{\partial J}{\partial a^{(3)}} \cdot \frac{\partial a^{(3)}}{\partial z^{(3)}}$$

case $i=j$

$$= \frac{-1}{N} \cdot \sum_{i=1}^{N} \frac{1}{\cancel{\text{softmax}(z_{y_i}^{(3)})}} \, \cancel{\text{softmax}(z_{y_i}^{(3)})}\left(1 - \text{soft}(z_{y_i}^{(3)})\right)$$

$$= \frac{1}{N}\left[\text{softmax}(z^{(3)}) - 1\right]$$

case $i \neq j$

$$= \frac{-1}{N} \sum_{i=1}^{N} \frac{1}{\cancel{\text{softmax}(z_{y_i}^{(3)})}} \cdot -\cancel{\text{softmax}\, z_{y_i}^{(3)}}\ \text{soft}\, z_{y_j}^{(3)}$$

$$= \frac{-1}{N} \text{softmax}(z^{(3)})$$

$$\therefore \frac{\partial J}{\partial z^{(3)}} = \frac{1}{N}\left(\text{softmax}(z^{(3)}) - \Delta\right)$$

where $\Delta_{ij} = \begin{cases} 1 & \text{if } y_i = j \\ 0 & \text{otherwise} \end{cases}$

b) $$\frac{\partial J}{\partial W^{(2)}} = \underbrace{\frac{\partial J}{\partial a^{(3)}} \cdot \frac{\partial a^{(3)}}{\partial z^{(3)}}}_{\text{from (a)}} \cdot \frac{\partial (z^{(3)})}{\partial W^{(2)}}$$

$$= \frac{1}{N}\left(\psi(z^{(3)}) - \Delta\right) \cdot \frac{\partial}{\partial W^{(2)}}\left[a^{(2)}W^{(2)} + b^{(2)}\right]$$

$$= \frac{1}{N}\left(\psi(z^{(3)}) - \Delta\right) \cdot a^{(2)T}$$

ii) eq: 13

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N} -\log\left[\frac{e^{z^{(3)}_{y_i}}}{\sum_j e^{z^{(3)}_j}}\right] + \lambda\left(\|W^{(1)}\|_2^2 + \|W^{(2)}\|_2^2\right)$$

$$\frac{\partial J}{\partial W^{(2)}} = \underbrace{\frac{\partial}{\partial W^{(2)}} \frac{1}{N}\sum_{i=1}^{N} -\log\left[\frac{e^{z^{(3)}_{y_i}}}{\sum_j e^{z^{(3)}_j}}\right]}_{\text{from (b)}} + \frac{\partial}{\partial W^{(2)}} \lambda\left(W^{(1)T}W^{(1)} + W^{(2)T}W^{(2)}\right)$$

$$= \frac{\partial J}{\partial W^{(2)}} = \frac{1}{N}\left(\psi(z^{(3)}) - \Delta\right) a^{(2)T} + 2\lambda W^{(2)}$$

(c) Expressions for regularized loss w.r.t $W^{(1)}$, $b^{(1)}$, $b^{(2)}$

$$\frac{\partial J}{\partial b^{(2)}} = \underbrace{\frac{\partial J}{\partial a^{(3)}} \cdot \frac{\partial a^{(3)}}{\partial z^{(3)}}} \cdot \frac{\partial z^{(3)}}{\partial b^{(2)}}$$

$$= \frac{1}{N} \cdot \left(\psi(z^{(3)}) - \Delta\right) \cdot \frac{\partial}{\partial b^{(2)}}\left(a^{(2)}W^{(2)} + b^{(2)}\right)$$

1

$$\frac{\partial J}{\partial w^{(1)}} = \underbrace{\frac{\partial J}{\partial a^{(3)}} \cdot \frac{\partial a^{(3)}}{\partial z^{(3)}}} \cdot \frac{\partial z^{(3)}}{\partial a^{(2)}} \cdot \frac{\partial a^{(2)}}{\partial z^{(2)}} \cdot \frac{\partial z^{(2)}}{\partial w^{(1)}}$$

$$= \frac{1}{N}\left(\psi(z^{(3)}) - \Delta\right) \cdot \frac{\partial}{\partial a^{(2)}}\left[a^{(2)} w^{(2)} + b^{(2)}\right] \cdot \frac{\partial}{\partial z^{(2)}} \mathrm{ReLU}(z^{(2)})$$

$$\cdot \frac{\partial}{\partial w^{(1)}} a^{(1)} w^{(1)} + b^{(1)}$$

$$+ \frac{\partial}{\partial w^{(1)}}\left(\lambda \|w^{(1)}\|_2^2 + \|w^{(2)}\|_2^2\right)$$

$$= \frac{1}{N}\left(\psi(z^{(3)}) - \Delta\right) \cdot w^{(2)} \cdot \phi'(z^{(2)}) \cdot a^{(1)}$$

$$+ 2\lambda w^{(1)}$$

$$\frac{\partial J}{\partial b^{(1)}} = \frac{\partial J}{\partial a^{(3)}} \cdot \frac{\partial a^{(3)}}{\partial z^{(3)}} \cdot \frac{\partial z^{(3)}}{\partial a^{(2)}} \cdot \frac{\partial a^{(2)}}{\partial z^{(2)}} \cdot \frac{\partial z^{(2)}}{\partial b^{(1)}}$$

$$= \frac{1}{N}\left(\psi(z^{(3)}) - \Delta\right) \cdot w^{(2)} \cdot \phi'(z^{(2)}) \cdot (1) + 0$$

Q3)
b) Hyperparameter Tuning:

The important parameters that impacted our best_model's performance are:

```
hidden_size = 100
num_iters=2000
learning_rate=1e-3
reg=0.5
```

Previously, the loss vs iteration curve was linear and we fixed this by changing the learning rate from 1e-4 to 1e-3 which resulted in a more or less exponentially decreasing curve.

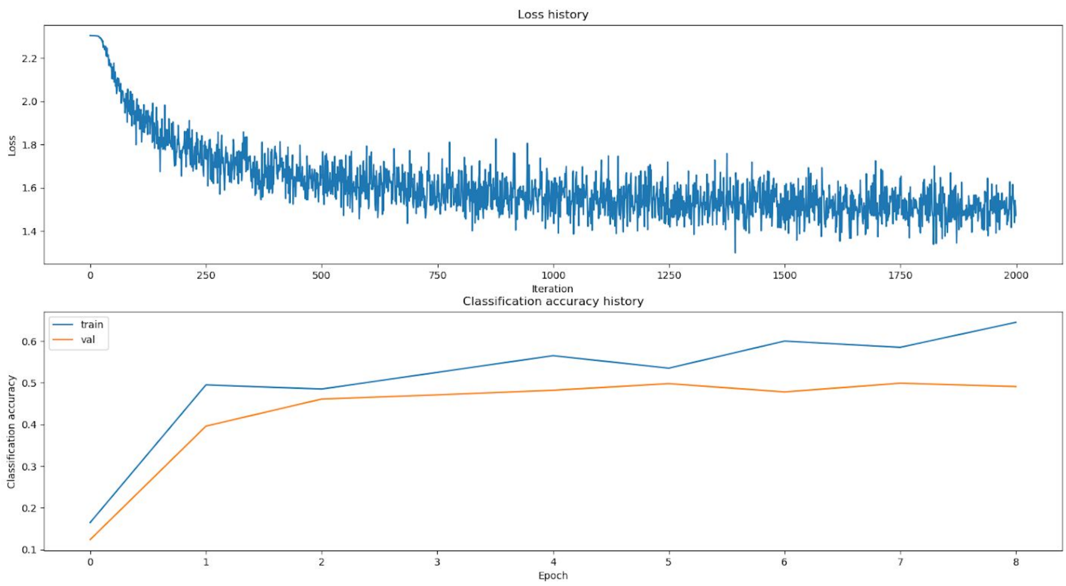
Increasing the hidden_size from 50 to 100 increased the capacity of the model. But just this would lead to overfitting and result in very low Test accuracy. To combat this, we increased the regulatriastion from reg=0.25 to reg=0.5.

Finally, we increased the number of iterations num_iters from 1000 to 2000 which improved the performance. This is because we are using mini batch gradient descent which performs better when we increase the number of passes through the dataset.

The final performance we got:

Validation accuracy: 0.496
Test accuracy: 0.51

Plots:



Q4)

b) For the given 2 Layers:

Validataion accuracy is: 51.6 %

Accuracy of the network on the 1000 test images: 50.7 %

c)

3 Layers	MultiLayerPerceptron(
	  (layers): Sequential(
	    (0): Linear(in_features=3072, out_features=50, bias=True)
	    (1): ReLU()
	    (2): Linear(in_features=50, out_features=40, bias=True)
	    (3): ReLU()
	    (4): Linear(in_features=40, out_features=10, bias=True)
	  )
	)
		Validataion accuracy is: 52.5 %
		
		Accuracy of the network on the 1000 test images: 49.7 %

4 Layers	MultiLayerPerceptron(
	  (layers): Sequential(
	    (0): Linear(in_features=3072, out_features=50, bias=True)
	    (1): ReLU()
	    (2): Linear(in_features=50, out_features=40, bias=True)
	    (3): ReLU()
	    (4): Linear(in_features=40, out_features=30, bias=True)
	    (5): ReLU()
	    (6): Linear(in_features=30, out_features=10, bias=True)
	  )
	)
		Validataion accuracy is: 7.9 %
		
		Accuracy of the network on the 1000 test images: 10.0 %

5 Layers	MultiLayerPerceptron(
	  (layers): Sequential(
	    (0): Linear(in_features=3072, out_features=50, bias=True)
	    (1): ReLU()
	    (2): Linear(in_features=50, out_features=40, bias=True)
	    (3): ReLU()
	    (4): Linear(in_features=40, out_features=30, bias=True)
	    (5): ReLU()
	    (6): Linear(in_features=30, out_features=25, bias=True)
	    (7): ReLU()
	    (8): Linear(in_features=25, out_features=10,
		Validataion accuracy is: 7.8 %
		
		Accuracy of the network on the 1000 test images: 9.0 %

	bias=True)))	

6 Layers	MultiLayerPerceptron(
	  (layers): Sequential(
	    (0): Linear(in_features=3072, out_features=50, bias=True)
	    (1): ReLU()
	    (2): Linear(in_features=50, out_features=40, bias=True)
	    (3): ReLU()
	    (4): Linear(in_features=40, out_features=30, bias=True)
	    (5): ReLU()
	    (6): Linear(in_features=30, out_features=25, bias=True)
	    (7): ReLU()
	    (8): Linear(in_features=25, out_features=20, bias=True)
	    (9): ReLU()
	    (10): Linear(in_features=20, out_features=10, bias=True)
	  )
	)
		Validataion accuracy is: 7.9 %
		
		Accuracy of the network on the 1000 test images: 10.0 %