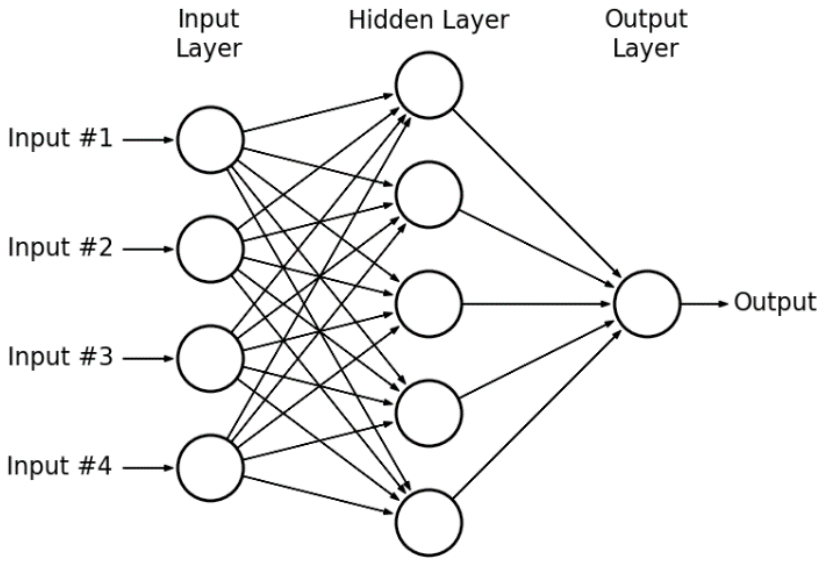
SAPIENZA
UNIVERSITÀ DI ROMA

AML Homework 1

Deep Neural Network and Backpropagation



Debodeep Banerjee
Trina Sahoo
Paolo Falcone
Dario Baraghini

October 2021

Question 2

a)

The loss function is defined by,
$j(\theta, x_i, y_{i\,i=1}^{N}) = \frac{1}{N}\sum -log[\frac{exp(z_i^{(3)})}{\sum exp(z_i^{(3)})_j}]$
where, $\frac{exp(z_i^{(3)})}{\sum exp(z_i^{(3)})_j} = \psi(z_i^{(3)})$
our task is to take the Loss gradient w.r.t to $z_i^{(3)}$
first we'll work on the

$$-log(softmax)$$

.
$-\log(\frac{e^{(z_i^{(3)})_{yi}}}{\sum_j e^{z_j}}) = -\log(e^{(z_i^{(3)})_{yi}}) + \log(\sum_j e^{(z_i^{(3)})_j}) = -(z_i^{(3)})_j + \log(\sum_j (e^{(z_i^{(3)})_j})$

now let's start calculating the gradient
$\nabla_{z_i}(-(z_i^{(3)})_j + \log(\sum_j e^{(z_i^{(3)})_j})) = \nabla_{z_i}\log(\sum_j e^{(z_i^{(3)})_j}) - \nabla_{z_i}(z_i^{(3)})_{yi} =$

$= \frac{1}{\sum_j e^{(z_i^{(3)})_j}}\nabla_{z_i}\sum_j e^{(z_i^{(3)})_j} - \nabla_{z_i}(z_i^{(3)})_{yi} =$

(note that the derivative of all the elements of the sum where $j \neq i$ are 0)
$= \frac{e^{(z_i^{(3)})}}{\sum_j e^{(z_i^{(3)})_j}} - \nabla_{z_i}(z_i^{(3)})_{yi} =$

$$\psi(z_i^{(3)})) - \delta_i$$

where $(\delta_i)_j = 1$ if $j = y_i, 0$ otherwise

Using the results above we get:
$\nabla_{z_i}\frac{1}{N}\sum_k \psi((z_i^{(3)})) - \delta_i =$

$$\frac{1}{N}(\psi((z_i^{(3)})) - \delta_i)$$

b)

Now our job is to compute the partial derivative of the loss w.r.t $W^{(2)}$. In order to do so we apply the chain rule:
$\frac{\delta J}{\delta W^{(2)}} = \sum_{i=1}^{N}\frac{\delta J}{\delta z_i^{(3)}}\cdot\frac{\delta z_i^{(3)}}{\delta W^{(2)}}$

The first term is the Derivative of the Loss function that we have calculated in the point 2.a. The second term is:
$\frac{\delta}{\delta W^{(2)}} z_i^{(3)}$. Since $z_i^{(3)} = W^{(2)}a^{(2)} + b(2)$ we can write:
$\frac{\delta}{\delta W^{(2)}} W^{(2)}a^{(2)} + b(2) = a^{(2)}$

putting everything together:

$$\sum_{i=1}^{N}\frac{1}{N}(\psi((z_i^{(3)})) - \delta_i)a_i^{(2)^T}$$

.

Now let's do the same thing but let's also take into account the Regularization term.
The loss function in this case is:

$$\frac{1}{N}\sum -log[\frac{exp(z_i^{(3)})}{\sum exp(z_i^{(3)})_j}] + \lambda(||W^{(1)}||_2^2 + ||W^{(2)}||_2^2)$$

let's work on the Regularization term.

$\frac{\delta}{\delta W^{(2)}}\lambda(||W^{(1)}||_2^2 + ||W^{(2)}||_2^2) = 2\lambda W^{(2)}$
We now get:

$$\sum_{i=1}^{N}\frac{1}{N}(\psi((z_i^{(3)})) - \delta_i)a_i^{(2)^T} + 2\lambda W^{(2)}$$

c)

The task is to derive the expression for regularization loss with respect to $W^{(1)}, b^{(1)}, b^{(2)}$.
Using chain rule the derivative of $W^{(1)}$ is obtained with respect to the loss J. Later the regularization term will also be added. So, according to the chain rule,
$\frac{\delta J}{\delta W^{(1)}} = \frac{\delta J}{\delta z_i^{(2)}}.\frac{\delta z_i^{(2)}}{\delta W^{(1)}}$
Now consider, $\frac{\delta J}{\delta z_i^{(2)}} = \frac{\delta J}{\delta a_i^{(2)}} \odot \frac{\delta a_i^{(2)}}{\delta z_i^{(2)}}$
$\frac{\delta J}{\delta z_i^{(2)}} = (\frac{\delta J}{\delta z_i^{(3)}}.\frac{\delta z_i^{(3)}}{\delta a_i^{(2)}}) \odot \frac{\delta a_i^{(2)}}{\delta z_i^{(2)}}$
where $\odot$ is the element wise multiplication.
Obtaining all the derivatives:

$$\frac{\delta a_i^{(2)}}{\delta z_i^{(2)}} = \begin{cases} 1 & \text{if } z_i^{(3)} > 0 \\ 0 & \text{otherwise} \end{cases}$$

From the equation $z^{(3)} = W^{(2)}a^{(2)} + b^{(2)}$, the derivative is: $\frac{\delta z_i^{(3)}}{\delta a_i^{(2)}}) = W^{(2)}$
From the equation $z^{(2)} = W^{(1)}a^{(1)} + b^{(1)}$, the derivative is: $\frac{\delta z_i^{(2)}}{\delta W^{(1)}} = a^{(1)}$
Combining all the derivative, we get,
$\frac{\delta J}{\delta W^{(1)}} = \frac{1}{N}\{[(W^{(2)})^T(\psi(z_i^{(3)} - \Delta)] \odot \frac{\delta a_i^{(2)}}{\delta z_i^{(2)}}\}.(a^{(2)})^T$
Now, we have to obtain the derivative of $W^{(1)}$ with respect to regularization term, $\frac{\delta\lambda(||W^{(1)}||_2^2 + ||W^{(2)}||_2^2)}{\delta W^{(1)}} = 2\lambda W^{(1)}$
So, finally we have as the regularization loss with respect to $W^{(1)}$,
$\frac{\delta J}{\delta W^{(1)}} = \frac{1}{N}\{[(W^{(2)})^T(\psi(z_i^{(3)} - \Delta)] \odot \frac{\delta a_i^{(2)}}{\delta z_i^{(2)}}\}.(a^{(2)})^T + 2\lambda W^{(1)}$

Using chain rule the derivative of $b^{(1)}$ is obtained with respect to the loss J. So according to the chain rule,
$\frac{\delta J}{\delta b^{(1)}} = \frac{\delta J}{\delta z_i^{(2)}}.\frac{\delta z_i^{(2)}}{\delta b^{(1)}}$
$\frac{\delta J}{\delta b^{(1)}} = (\frac{\delta J}{\delta z_i^{(3)}}.\frac{\delta z_i^{(3)}}{\delta a_i^{(2)}}) \odot \frac{\delta a_i^{(2)}}{\delta z_i^{(2)}}.\frac{\delta z_i^{(2)}}{\delta b^{(1)}}$
From the equation $z^{(2)} = W^{(1)}a^{(1)} + b^{(1)}$, the derivative is: $\frac{\delta z_i^{(2)}}{\delta b^{(1)}} = I$

where I represents matrix of ones.
So, finally we have as the regularization loss with respect to $b^{(1)}$,
$\frac{\delta J}{\delta b^{(1)}} = \frac{1}{N}\{[(W^{(2)})^T(\psi(z_i^{(3)} - \Delta)] \odot \frac{\delta a_i^{(2)}}{\delta z_i^{(2)}}\}.I$

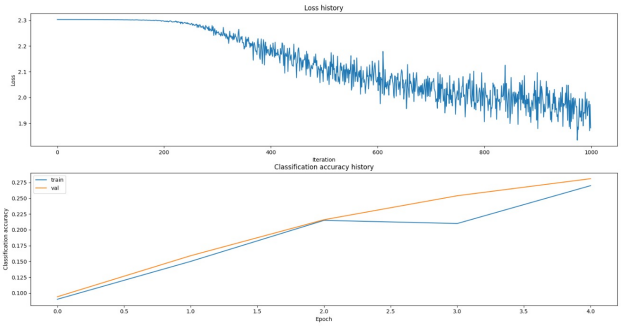
Using chain rule the derivative of $b^{(2)}$ is obtained with respect to the loss J. So according to the chain rule,
$\frac{\delta J}{\delta b^{(2)}} = \frac{\delta J}{\delta z_i^{(3)}} . \frac{\delta z_i^{(3)}}{\delta b^{(2)}}$
From the equation $z^{(3)} = W^{(2)}a^{(2)} + b^{(2)}$, the derivative is: $\frac{\delta z_i^{(3)}}{\delta b_i^{(2)}}) = I$
So, finally we have as the regularization loss with respect to $b^{(2)}$,
$\frac{\delta J}{\delta b^{(2)}} = \frac{1}{N}(\psi(z_i^{(3)}) - \Delta).I$

Question 3

b)

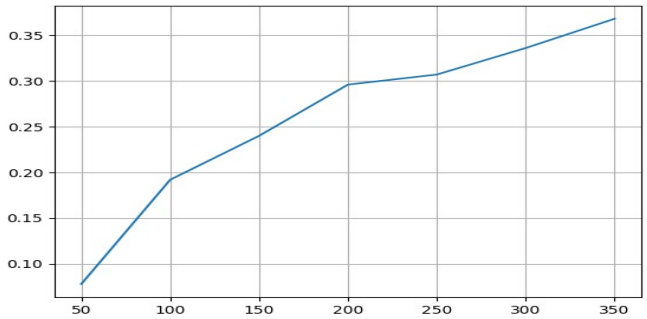
The first plot explains the training loss:

With the default hyper-parameters we obtained the loss curve and classification accuracy shown below:



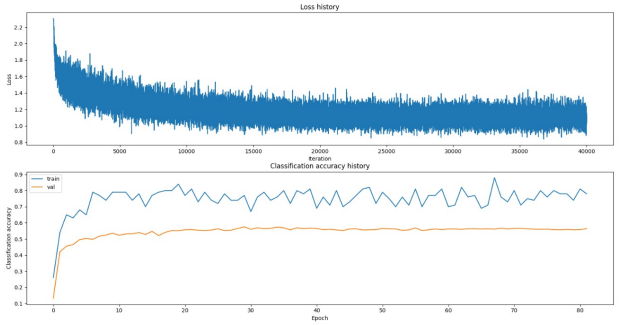
It is evident that the loss is too high and at the same time the accuracy is also poor which is around 29%. So we need to tune the given hyper parameter, that is, hidden size, input size, learning rate and number of classes. At first we have checked the best size of the hidden layer and then obtain the validation accuracy keeping the other parameters same except changing the learning rate from $1e{-}4$ to $1e{-}3$.

The picture below shows the graph of the hidden size:



We consider the size of the hidden layer as 350. The selection of the hyper-parameter gives us 57.5% accuracy on the test set. The validation accuracy is obtained as 58.6%.

Next we obtain the loss curve and classification accuracy curve as before but increasing the iteration to 30000, hidden size as 350, learning rate as $1e-3$ and learning rate decay as 0.92. The figure is provided below:



In this graph we can observe that the loss has reduced considerably and at the same time the classification accuracy has also increased. So, the accuracy on the test set is obtained as 57.5%.

Question 4

b), c)

The final validation accuracy we achieved with the first two-layer network using pytorch is 51.4%.
The next step is to experiment different networks with different parameters and see how they perform. Take note that we will report for each network the best configuration we got. In our case the best Configuration for all networks was achieved using Linear layers and Relu Activation functions.

# Layers	Neurons in each Layer	Dropout	Batch Normalization	Validation Accuracy
2	2000/700	Yes	Yes	55.10%
3	700/400/100	Yes	Yes	55%
4	700/400/100/300	No	Yes	54.5%
5	700/400/100/300/200	No	Yes	52.8%

Table 1: Configurations

Our best performing configuration is the following:

```
self.flatten=torch.nn.Flatten()
self.linear1 = torch.nn.Linear(input_size, 2000)
self.activation1 = torch.nn.ReLU()
self.drop=torch.nn.Dropout(p=0.4)
self.linear2 = torch.nn.Linear(2000,700)
self.batch=torch.nn.BatchNorm1d(700)
self.activation2 = torch.nn.ReLU()
self.out = torch.nn.Linear(700,num_classes)
```

with batch size of 256 and 100 epoches.
The validation accuracy is 55.10% and the test accuracy is 59.00%

One important thing to notice is that increasing the number of layers not always leads to better performances as we can see in Table 1 . Batch Normalization and Dropout often lead to better performances regardless of the number of layers but keep in mind that the latter can lead to errors if we have a small number of neurons or if it's just before the output layer.
Also notice that the best performing number of Neurons is $< inputsize + outputsize$ like many heuristics suggest.