CSCI 5922 -Neural Networks and Deep Learning
Problem Set - 2 Solutions

Gowri Shankar Raju Kurapati
Student ID 110568555

September 19, 2022

1 Model Parameters versus Hyperparameters

- **Model Parameters** define the configuration of the model and they are internal to the model. These parameters are essential for making the predictions which are learned via training. These parameters are dependent on a dataset directly.
- **Hyperparameters** are the parameters that are clearly and explicitly specified and are used for optimizing the model. These parameters are set manually by the engineer as these are independent of the dataset. The quality of the model is enhanced by fine-tuning the hyperparameters.
- **Difference** While hyperparameters are manually defined, employed in the optimization of the model, and aid in estimating model parameters, model parameters are intrinsic to the model and are calculated from data automatically.

2 Dataset Splits

- A given model is analyzed using the validation set, although this is done frequently. This knowledge is utilized by machine learning engineers to adjust the model hyperparameters. As a result, even if the model occasionally encounters this data, it never ”Learns” from it. We adjust higher-level hyperparameters by using validation set findings. In this respect, a model is indirectly impacted by the validation set.
- The purpose of the validation dataset is to act as a bridge between the training dataset and the test dataset. While the training dataset is used to find the model parameters for a particular set of hyperparameters, we do not want to adjust the model on test data because the very purpose of the test split is lost if we did. That is why we need a separate dataset to analyze how well the model is performing on unseen data during training so that we can tune the hyperparameters better but at the same time we want another dataset to FINALLY test on. That is the main motivation to split the training dataset into the actual training datasets and validation datasets.

3 Overfitting versus Underfitting

- *figure-(b)* represents the Under-fitting learning curve of the model. As we can observe, the training loss saturated at a very high loss, and validation loss shows a similar trend with validation loss slightly more than training loss as the training progresses. The accuracy is less as the model is underfitting the training data, thereby having a high loss on both training data and validation data.

- *figure-(a)* represents the Over-fitting learning curve of the model. As we can observe, as the training loss decreases quickly, the validation loss is still high and gets saturated with a very high loss compared to the very low loss of training. As the model is trained over a large training set, it is trying to overfit the points and thus, performing worse on the unseen validation set.

4 Optimization

- **Batch Gradient Descent**: In batch gradient descent, each step is chosen after taking into account all the training data. By using mean gradient, we update our parameters by averaging the gradients of all the training samples. Therefore, it is just one gradient descent step in one epoch.
- **Stochastic Gradient Descent**: In Stochastic Gradient Descent (SGD), we consider just one example at a time to take a single step in the gradient space. So for every epoch and for every data point in that epoch, we find the gradient step and move in the loss space.
- **Mini-Batch Gradient Descent**: Neither we use all the dataset all at once nor we use the single example at a time. We use a batch of a fixed number of training examples which is less than the actual dataset and call it a mini-batch. For every epoch,we divide the dataset to mini batch and use each minibatch to make one gradient step towards the minimum in the loss space.

  If our dataset is very huge, Batch Gradient Descent would have to iterate over all the dataset to just take a single step which is not efficient and that is where stochastic gradient comes to the picture to update for every single point, but this too is inefficient, as at each step, one training datapoint is deciding the direction of the step and there may be very high chances of fluctuations in gradient. Mini-batch gradient descent gets the best of both worlds by decreasing the cons of SGD and BGD, managing the step efficiently with fewer fluctuations.

5 Model Size

4 Layer neural network with input layer(i), 3 hidden Layers(h1 -> h2 -> h3) and one output layer(o) of 100 nodes.
The input is an image of the size 64 X 64.

5.1 Fully Connected Layers

Each hidden layer has 5 nodes

- Weight Parameters, Bias Parameters
  i -> h1 = (64 * 64) * 5 , 5
  h1 -> h2 = (5 * 5) , 5
  h2 -> h3 = (5 * 5) , 5
  h3 -> o = (5 * 100), 100
  Total = **21030**, **115**

  There are **21,030 Weight parameters** and **115 bias parameters**

5.2 Convolution Neural networks

In

- Weight Parameters, Bias Parameters
  i -> h1 = (3*3*1)*3 , 3
  h1 -> h2 = (3*3*3) * 3 ,3
  h2 -> h3 = (3*3*3) * 3 ,3
  h3 -> o = (58 * 58 *3)* 100,100
  Total = **1,009,389**, **109**

  There are **1,009,389 Weight parameters** and **109 bias parameters**

6 Convolution Neural Networks

CNN's are mostly used for image processing and that is why the advantages are discussed in terms of dealing with image data.

6.1 Advantages

- **Spatial Information** : Because fully connected layers are "completely linked," meaning that all output neurons are coupled to all input neurons, fully connected layers typically result in loss of spatial information. If you are working in a vast space of possibilities, this type of design cannot be employed for segmentation. But on the other hand, CNN's preserve spatial information by applying filters on the images and thus extracting useful features from the input.
- **Input Size**: You can use the network to process images of practically any size even if your network doesn't have any completely linked layers. However, a fully connected layer expects the input to be of fixed size which is the number of nodes of the input layer.

6.2 Stride = 1, No Padding

Applying the filter on input data

- The given input data is

1	1	0	2
4	0	8	1
6	4	2	3
8	7	4	2

- The filter is

0	0.5	0
0.5	1	0
0	0	0

First Step is to do one-one multiplication of highlighted sub matrix of input with the filter.

- Highlighted Input Matrix:

1	**1**	**0**	2
4	**0**	**8**	1
6	**4**	**2**	3
8	7	4	2

- The multiplication is follows :

1*0.5 + 0 * 1 + 4 * 0.5 = 2.5	TBD
TBD	TBD

Second Step is to do one-one multiplication of highlighted sub matrix of input with the filter.

- Highlighted Input Matrix:

1	**1**	**0**	**2**
4	**0**	**8**	**1**
6	**4**	**2**	**3**
8	7	4	2

- The multiplication is follows :

2.5	0 * 0.5 + 8 * 1 + 1 * 0.5 = 8
TBD	TBD

Third Step is to do one-one multiplication of highlighted sub matrix of input with the filter.

- Highlighted Input Matrix:

1	1	0	2
4	**0**	**8**	1
6	**4**	**2**	3
8	**7**	**4**	2

- The multiplication is follows :

2.5	8
0 * 0.5 + 4 * 1 + 6 * 0.5 = 7	TBD

Final Step is to do one-one multiplication of highlighted sub matrix of input with the filter.

- Highlighted Input Matrix:

1	1	0	2
4	**0**	**8**	**1**
6	**4**	**2**	**3**
8	**7**	**4**	**2**

- The multiplication is follows :

2.5	8
7	0.5 * 8 + 1 * 2 + 0.5 * 4 = 8

The Output is :

2.5	8
7	8

6.3 Stride = 3 x 3, Zero Padding

Applying the filter on padded input data

- Input Data After Padding

0	0	0	0	0	0
0	1	1	0	2	0
0	4	0	8	1	0
0	6	4	2	3	0
0	8	7	4	2	0
0	0	0	0	0	0

- The filter is

0	0.5	0
0.5	1	0
0	0	0

The First Step is to do one-one multiplication of highlighted sub matrix of input with the filter.

- Highlighted Input Matrix:

0	**0**	**0**	0	0	0
0	**1**	**1**	0	2	0
0	**4**	**0**	8	1	0
0	6	4	2	3	0
0	8	7	4	2	0
0	0	0	0	0	0

- The multiplication is follows :

0.5 * 0 + 1 * 1 + 0.5 * 0 = 1	TBD
TBD	TBD

The Second Step is to do one-one multiplication of highlighted sub matrix of input with the filter and stride 3 row wise.

- Highlighted Input Matrix:

0	0	0	**0**	**0**	**0**
0	1	1	**0**	**2**	**0**
0	4	0	**8**	**1**	**0**
0	6	4	2	3	0
0	8	7	4	2	0
0	0	0	0	0	0

- The multiplication is follows :

1	0.5 * 0 + 1 * 2 + 0.5 * 0 = 2
TBD	TBD

The Third Step is to do one-one multiplication of highlighted sub matrix of input with the filter and stride 3 column wise.

- Highlighted Input Matrix:

0	0	0	0	0	0
0	1	1	0	2	0
0	4	0	8	1	0
0	**6**	**4**	2	3	0
0	**8**	**7**	4	2	0
0	**0**	**0**	0	0	0

- The multiplication is follows :

1	2
0.5 * 6 + 1 * 8 + 0.5 * 0 = 11	TBD

The Final Step is to do one-one multiplication of highlighted sub matrix of input with the filter and stride 3 row wise.

- Highlighted Input Matrix:

0	0	0	0	0	0
0	1	1	0	2	0
0	4	0	8	1	0
0	6	4	**2**	**3**	**0**
0	8	7	**4**	**2**	**0**
0	0	0	**0**	**0**	**0**

- The multiplication is follows :

1	2
11	0.5 * 3 + 1 * 2 + 0.5 * 4 = 5.5

The Output is :

1	2
11	5.5