Problem Set 2
CSCI 5922: Neural Networks and Deep Learning

Rajeev R Menon (110581437)

November 3, 2022

Problem 1: Model Parameters vs. Hyperparameters

Model parameters are properties of the neural network that are learnt after training on a given dataset. These parameters vary with each neural network according to the objective of the model and the type of dataset used. For example, weights and biases which are determined after training of a neural network are considered as model parameters.

Hyperparameters are properties of the learning algorithm that controls the value of ultimate model parameters. Instead of learning it after training, hyperparameters are supplied to the model before training. Common hyperparameters in neural networks include, number of hidden nodes, number of hidden layers, activation function, learning rate etc.

Problem 2: Dataset Splits

For training and testing purposes of any neural network models, the dataset is usually split into three parts, training, testing and validation sets. Training set is the actual set of data used to train the model and determine various parameters. After training, the model is tested using the values from test dataset to evaluate the performance. The validation set is a split of the dataset, separate from the training data, used to validate the model during training.

The validation process using the validation dataset, helps in giving information on adjusting the hyperparameters. Unlike during training, the parameters are not updated during validation against the validation set. Validation set ensures that the model is not overfitting to the training dataset. This helps in generalizing the model and helps in making accurate classifications on data that it was not trained on.

Problem 3: Overfitting versus Underfitting

(a)

Graph (b) in Figure 1 is representative of underfitting in a model. The learning curve shows high training error, which subsequently results in high validation error. This causes an increase in the value of the biases resulting in an underfitting model. The curve flattens relatively quickly indicating an unrepresentative dataset as well. The training dataset does not provide sufficient information to learn the problem.

(b)

Graph (a) of figure 1 is representative of an overfitting model. The training error is shown to decrease and flatten towards the end, but the validation error remains high. The model over fits to the training model, resulting in lower error values for the training data and higher error for any data from outside the training dataset, in this case the validation dataset. This is the result of training even after reaching an optimal fit.

Problem 4: Optimization

The process of repeatedly adjusting the input parameters of a function by some multiple of the negative gradient is called gradient descent. It converges the function towards a local minimum of the cost function. In neural networks, it is used to minimise the value of loss during training. There are a few varieties of gradient descent used in deep learning.

Batch gradient descent considers the whole of training data in a single step. It calculates the average loss and gradient of each training sample. The mean value is then used to adjust the parameters of the model. It has only one step of gradient descent per epoch. Batch gradient descent is generally suitable for models with relatively smooth descent to local minima of the cost function.

Stochastic gradient descent considers one random sample of data as a single step. Instead of taking carefully calculated descent to the local minima, stochastic gradient descent results in quick random steps towards the local minima. This is applicable in the case of large datasets where it will help in converging relatively faster around the local minima.

Mini batch gradient descent is a combination of both batch gradient descent and stochastic gradient descent. The dataset is split into random mini batches of training samples. The mean gradient descent of each of these batches are used to adjust the parameters of the model. This helps in faster computation of the gradient descent at the same time retaining the advantages of both batch and stochastic gradient descent.

Problem 5: Model Size

Part a

Input = 64 * 64 matrix
No. of layers = 4
No. of hidden layers = 3
No. of nodes in output layer = 100
No. of nodes per hidden layer = 5

- Number of model parameters between input layer and hidden layer 1:
  Number of weights = 64 * 64 * 5 = 20480
  Number of bias terms = 5
- Number of model parameters between hidden layers 1 and 2:
  Number of weights = 5 * 5 = 25
  Number of bias terms = 5
- Number of model parameters between hidden layers 2 and 3:
  Number of weights = 5 * 5 = 25
  Number of bias terms = 5
- Number of model parameters between hidden layer 3 and output layer:
  Number of weights = 5 * 100 = 500
  Number of bias terms = 100

Total number of weights = 20480 + 25 + 25 + 500 = 21030
Total number of biases = 5 + 5 + 5 + 100 = 115
Total number of model parameters = = 21030 + 115 = 21145

Part b

Input = 64x64 matrix
Number of layers = 4
Number of hidden layers = 3
Number of nodes in the output layer = 100
Filter size for convolutional layer between the hidden layers = 3x3

- Number of model parameters between input layer and hidden layer 1:
  Number of 3 x 3 filters per layer: 3
  Convoluted image size: 62 x 62
  Weights = 3 * 3 * 3 = 27
  Bias terms = 3

- Number of model parameters between hidden layers 1 and 2:
  Number of 3 x 3 filter: 3
  Number of channels of image: 3
  Image size: 60 x 60
  Number of weights = 3 * 3 * 3 * 3 = 81
  Number of bias terms = 3

- Number of model parameters between Hidden layers 2 and 3:
  Number of 3 x 3 filter: 3
  Number of channels of image: 3
  Image size: 58 x 58
  Number of weights = 3 * 3 * 3 * 3 = 81
  Number of bias terms = 3

- Number of model parameters between Hidden layers 3 and 4:
  Both layers are fully connected.
  The input of the hidden layer 3: 58x58x3
  Number of nodes in output layer: 100
  Number of weights = 58 * 58 * 3 * 100 = 1009200
  Number of bias terms = 100

Total number of weights = 27 + 81 + 81 + 1009200 = 1009389
Total number of biases = 3 + 3 + 3 + 100 = 109
Total number of model parameters = = 1009389 + 109 = 1009498

Problem 6: Convolution Neural Networks

Part a

Convolutional layers in a neural network has various advantages over a fully connected neural network. Convolutional neural networks automatically identifies features in problems without the need for human intervention on its own. With the help of features like parameter sharing, pooling procedures etc, CNNs are computationally efficient which enables it to run on any devices. It also do not have dense connections which provides it more flexibility in learning high dimensional data like visual and audio data.

Part b

Input data:

$$\begin{bmatrix} 1 & 1 & 0 & 2 \\ 4 & 0 & 8 & 1 \\ 6 & 4 & 2 & 3 \\ 8 & 7 & 4 & 2 \end{bmatrix}$$

Filter:

$$\begin{bmatrix} 0 & 0.5 & 0 \\ 0.5 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Output:

$$\begin{aligned} Output_{0,0} &= Input_{0,0} * Filter_{0,0} + Input_{0,1} * Filter_{0,1} + Input_{0,2} * Filter_{0,2} \\ &+ Input_{1,0} * Filter_{1,0} + Input_{1,1} * Filter_{1,1} + Input_{1,2} * Filter_{1,2} \\ &+ Input_{2,0} * Filter_{2,0} + Input_{2,1} * Filter_{2,1} + Input_{2,2} * Filter_{2,2} \\ &= 1 * 0 + 1 * 0.5 + 0 * 0 + 4 * 0.5 + 1 * 0 + 2 * 0 + 6 * 0 + 4 * 0 + 2 * 0 \\ &= 0.5 + 2 \\ &= 2.5 \end{aligned}$$

$$\begin{aligned} Output_{0,1} &= Input_{0,1} * Filter_{0,0} + Input_{0,2} * Filter_{0,1} + Input_{0,3} * Filter_{0,2} \\ &+ Input_{1,1} * Filter_{1,0} + Input_{1,2} * Filter_{1,1} + Input_{1,3} * Filter_{1,2} \\ &+ Input_{2,1} * Filter_{2,0} + Input_{2,2} * Filter_{2,1} + Input_{2,3} * Filter_{2,2} \\ &= 1 * 0 + 0 * 0.5 + 2 * 0 + 0 * 0.5 + 8 * 1 + 0 * 0 + 4 * 0 + 2 * 0 + 3 * 0 \\ &= 8 \end{aligned}$$

$$\begin{aligned} Output_{1,0} &= Input_{1,0} * Filter_{0,0} + Input_{1,1} * Filter_{0,1} + Input_{1,2} * Filter_{0,2} \\ &+ Input_{2,0} * Filter_{1,0} + Input_{2,1} * Filter_{1,1} + Input_{2,2} * Filter_{1,2} \\ &+ Input_{3,0} * Filter_{2,0} + Input_{3,1} * Filter_{2,1} + Input_{3,2} * Filter_{2,2} \\ &= 4 * 0 + 0 * 0.5 + 8 * 0 + 6 * 0.5 + 4 * 1 + 2 * 0 + 8 * 0 + 7 * 0 + 4 * 0 \\ &= 3 + 4 \\ &= 7 \end{aligned}$$

$$\begin{aligned} Output_{1,1} &= Input_{0,0} * Filter_{0,0} + Input_{0,1} * Filter_{0,1} + Input_{0,2} * Filter_{0,2} \\ &+ Input_{1,0} * Filter_{1,0} + Input_{1,1} * Filter_{1,1} + Input_{1,2} * Filter_{1,2} \\ &+ Input_{2,0} * Filter_{2,0} + Input_{2,1} * Filter_{2,1} + Input_{2,2} * Filter_{2,2} \\ &= 0 * 0 + 8 * 0.5 + 1 * 0 + 4 * 0.5 + 2 * 1 + 3 * 0 + 7 * 0 + 4 * 0 + 2 * 0 \\ &= 4 + 2 + 2 \\ &= 8 \end{aligned}$$

Output:

$$\begin{bmatrix} 2.5 & 8 \\ 7 & 8 \end{bmatrix}$$

Part c

Input with zero padding layer:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 2 & 0 \\ 0 & 4 & 0 & 8 & 1 & 0 \\ 0 & 6 & 4 & 2 & 3 & 0 \\ 0 & 8 & 7 & 4 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Output:

$$\begin{aligned} Output_{0,0} &= Input_{0,0} * Filter_{0,0} + Input_{0,1} * Filter_{0,1} + Input_{0,2} * Filter_{0,2} \\ &+ Input_{1,0} * Filter_{1,0} + Input_{1,1} * Filter_{1,1} + Input_{1,2} * Filter_{1,2} \\ &+ Input_{2,0} * Filter_{2,0} + Input_{2,1} * Filter_{2,1} + Input_{2,2} * Filter_{2,2} \\ &= 0 * 0 + 0 * 0.5 + 0 * 0 + 0 * 0.5 + 1 * 0 + 1 * 0 + 0 * 0 + 4 * 0 + 0 * 0 \\ &= 1 \end{aligned}$$

$$\begin{aligned} Output_{0,1} &= Input_{0,3} * Filter_{0,0} + Input_{0,4} * Filter_{0,1} + Input_{0,5} * Filter_{0,2} \\ &+ Input_{2,0} * Filter_{1,3} + Input_{1,4} * Filter_{1,1} + Input_{1,5} * Filter_{1,2} \\ &+ Input_{2,3} * Filter_{2,0} + Input_{2,4} * Filter_{2,1} + Input_{2,5} * Filter_{2,2} \\ &= 0 * 0 + 0 * 0.5 + 0 * 0 + 0 * 0.5 + 2 * 1 + 0 * 0 + 8 * 0 + 1 * 0 + 0 * 0 \\ &= 2 \end{aligned}$$

$$\begin{aligned} Output_{1,0} &= Input_{3,0} * Filter_{0,0} + Input_{3,1} * Filter_{0,1} + Input_{3,2} * Filter_{0,2} \\ &+ Input_{4,1} * Filter_{1,0} + Input_{4,1} * Filter_{1,1} + Input_{4,2} * Filter_{1,2} \\ &+ Input_{5,0} * Filter_{2,0} + Input_{5,1} * Filter_{2,1} + Input_{5,2} * Filter_{2,2} \\ &= 0 * 0 + 6 * 0.5 + 4 * 0 + 0 * 0.5 + 8 * 1 + 7 * 0 + 0 * 0 + 0 * 0 + 0 * 0 \\ &= 3 + 8 \\ &= 11 \end{aligned}$$

$$\begin{aligned} Output_{0,1} &= Input_{3,3} * Filter_{0,0} + Input_{3,4} * Filter_{0,1} + Input_{3,5} * Filter_{0,2} \\ &+ Input_{4,3} * Filter_{1,3} + Input_{4,4} * Filter_{1,1} + Input_{4,5} * Filter_{1,2} \\ &+ Input_{5,3} * Filter_{2,0} + Input_{5,4} * Filter_{2,1} + Input_{5,5} * Filter_{2,2} \\ &= 2 * 0 + 3 * 0.5 + 0 * 0 + 4 * 0.5 + 2 * 1 + 0 * 0 + 0 * 0 + 0 * 0 + 0 * 0 \\ &= 1.5 + 2 + 2 \\ &= 5.5 \end{aligned}$$

Output:

$$\begin{bmatrix} 1 & 2 \\ 11 & 5.5 \end{bmatrix}$$