Week 5 - PRACTICAL-II

Artificial Neural Network

Submitted by - Sourav Rai

Bsc.IT B

1) What is a batch size in a neural network? Describe the various batch sizes. What is the preferred batch size, and why?

   The batch size is a hyperparameter that specifies how many samples must be processed before the internal model parameters are updated. Consider a batch as a for-loop making predictions while iterating over one or more samples. The predictions are compared to the anticipated output variables at the batch's conclusion, and an error is calculated. The update algorithm is used to correct this inaccuracy, for example, by moving down the gradient of the error.

   The learning algorithm is known as batch gradient descent when all training data are combined into a single batch. Stochastic gradient descent is the name of the learning algorithm when the batch size is one sample. The learning algorithm is known as mini-batch gradient descent when the batch size is greater than one sample and less than the size of the training dataset.In the case of mini-batch gradient descent, popular batch sizes include 32, 64, and 128 samples.

   - **Batch Gradient Descent**. Batch Size = Size of Training Set
   - **Stochastic Gradient Descent**. Batch Size = 1
   - **Mini-Batch Gradient Descent**. 1 < Batch Size < Size of Training Set

2) What is the purpose of the activation function in ANN? Explain various types of activation functions used in neural networks.

   A neuron's activation status is determined by an activation function. By employing simpler mathematical procedures, it will determine whether or not the neuron's input to the network is significant during the prediction process. A node's activation function's job is to create an output from a set of input values (or a layer).

Types of Activation Functions

1. Sigmoid Function

The sigmoid function in an ANN is a non-linear AF mostly employed in feedforward neural networks. It is a smooth, differentiable real function that is defined for real input values and has positive derivatives everywhere. The sigmoid function may be found in the deep learning models' output layer and is used to forecast results that are based on probability. The sigmoid function looks like this -

$$f(x) = \left(\frac{1}{(1 + exp^{-x})} \right) - (1.4)$$

2. Hyperbolic Tangent Function (Tanh)

Another type of AF is the tanh function, also known as the hyperbolic tangent function. It has a range between -1 and 1, is smoother, and is centered on zero. The following is a representation of the tanh function's output

$$f(x_i) = \frac{exp\ (x_i)}{\sum_j exp\ (x_j)} - (1.12)$$

3. Softmax Function

Another AF type used in neural networks to calculate probability distribution from a vector of real numbers is the softmax function. This function produces an output that has a value between 0 and 1 with a probability sum of equal to 1. The following is a representation of the softmax function

$$f(x_i) = \frac{exp\ (x_i)}{\sum_j exp\ (x_j)} \quad -(1.12)$$

4. Softsign Function

The softsign function is another AF that is used in neural network computing. Although it is primarily in regression computation problems, nowadays it is also used in DL based text-to-speech applications. It is a quadratic polynomial, represented by -

$$f(x) = \left(\frac{x}{|x| + 1}\right) \quad -(1.13)$$

5. Rectified Linear Unit (ReLU) Function

The rectified linear unit (ReLU) function, one of the most well-liked AFs in DL models, is a fast-learning AF that promises to give cutting-edge performance with excellent outcomes. The ReLU function delivers significantly higher deep learning performance and generalization when compared to other AFs, such as the sigmoid and tanh functions. Gradient-descent methods are simple to optimize since the function is almost linear and retains the characteristics of linear models. Each input element is subjected to a threshold operation by the ReLU function, which sets all values below zero to zero.As a result, the ReLU is shown as:

$$f(x) = max\,(0, x) = \begin{cases} x_i, & if\ x_i \geq 0 \\ 0, & if\ x_i < 0 \end{cases} \quad -(1.14)$$

6. Exponential Linear Units (ELUs) Function

Another AF that is utilized to quicken the training of neural networks is the exponential linear units (ELUs) function (just like ReLU function). The ELU function's greatest benefit is its ability to solve the vanishing gradient problem by employing identity for positive values and enhancing the model's learning properties. ELUs have negative values that reduce computing complexity and accelerate learning by moving the mean unit activation closer to zero. The ELU is a fantastic substitute for the ReLU because it reduces bias shifts by training mean activation toward zero.

$$f(x) = \begin{pmatrix} x, & if\ x > 0 \\ \alpha\ exp(x) - 1, & if\ x \leq 0 \end{pmatrix} \quad -(1.22)$$

3) What is the Optimization process in Deep Learning? Explain the popular optimizers. Differentiate between regularization and optimization.

We must adjust the weights for each epoch during deep learning model training and reduce the loss function. An optimizer is a procedure or method that alters neural network properties like weights and learning rates. As a result, it aids in decreasing total loss and raising precision. Some of the popular optimizers are Gradient Descent, Stochastic Gradient Descent, Adagrad,Adadelta, RMSprop,Adam. The main conceptual difference is that optimization is about finding the set of parameters/weights that maximizes/minimizes some objective function (which can also include a regularization term), while regularization is about limiting the values that your parameters can take during the optimization/learning/training.

4) Discuss various loss(error) functions used in training the ANN models.

Various type of loss function are:

1. Mean Squared Error (MSE)
2. Binary Crossentropy (BCE)
3. Categorical Crossentropy (CC)
4. Sparse Categorical Crossentropy (SCC)

Mean Squared Error (MSE)

Regression tasks employ MSE loss. As implied by the name, this loss is determined by averaging the squared discrepancies between actual (target) and anticipated values.

Binary Crossentropy (BCE)

The binary classification problems employ BCE loss. When employing the BCE loss function, only one output node is required to divide the data into two categories. The output range is and the output value needs to be routed through a sigmoid activation function (0 – 1).

Categorical Crossentropy (CC)

One of the loss functions you can use when we have a multi-class classification problem is this one. There must be the same number of output nodes as classes if the CCE loss function is being used. Additionally, a softmax activation should be applied to the final layer output so that each node outputs a probability value between (0–1).

Sparse Categorical Crossentropy (SCC)

With one adjustment, this loss function is essentially identical to CCE. You do not have to hot encode the target vector when using the SCCE loss function. You simply pass 0 if the target image is a cat, otherwise 1. Basically, you just pass the class's index, regardless of the class.

5) Explain the difference between bias and variance and their impact on overfitting and underfitting.

Bias -

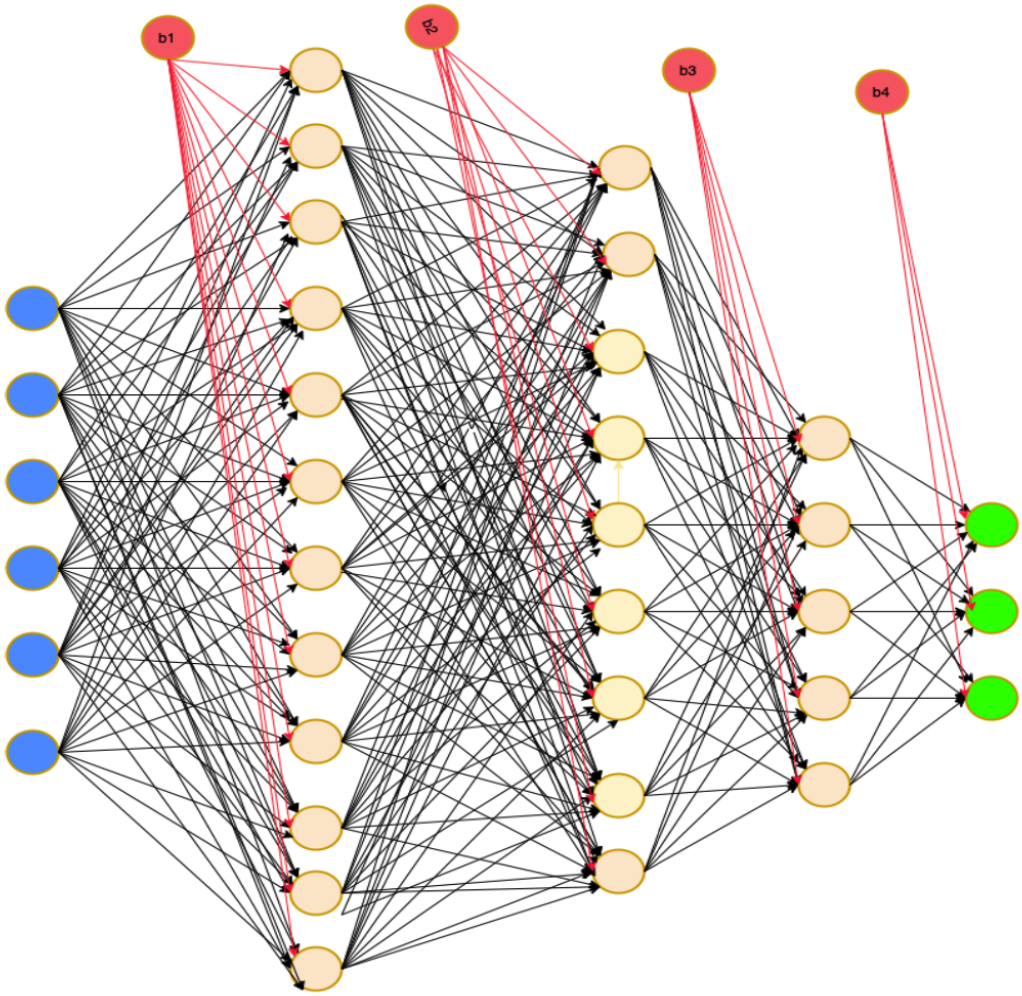
Assume that the model has been trained and that we are attempting to predict values using the input 'x train'. y predicted represents the predicted values. The error rate of y predicted and y train is known as bias. Simply put, consider bias to be the training data's error rate. When the error rate is large, we refer to it as high bias, and when it is low, we refer to it as low bias.

Variance -

Assume that the model has been trained and that we are now attempting to predict values using the input 'x test'. Once more, y predicted contains the expected values. The error rate of the y predicted and y test is known as variation. Consider variance as the testing data's error rate in plain terms. We refer to high variance when the error rate is large, and low variance when the error rate is low.

The impact of bias and variance on overfitting and underfitting is, if there is high bias then it implies underfitting, and if there is high variance then it implies overfitting.

6) What are trainable and non-trainable parameters? Calculate the trainable parameters for the Artificial Neural Network below: Explain the process.



In keras, **non-trainable** parameters (as shown in model.summary()) means the **number of weights** that are not updated during training with backpropagation.
There are mainly two types of non-trainable weights:

- The ones that you have chosen to keep constant when training. This means that keras won't update these weights during training at all.
- The ones that work like statistics in BatchNormalization layers. They're updated with mean and variance, but they're not "trained with backpropagation".

Weights are the values inside the network that perform the operations and can be adjusted to result in what we want. The backpropagation algorithm changes the weights towards a lower error at the end.

By default, all weights in a keras model are trainable. When you create layers, internally it creates its own weights and they're trainable. (The backpropagation algorithm will update these weights) When you make them untrainable, the algorithm will not update these weights anymore. This is useful, for instance, when you want a convolutional layer with a specific filter, like a Sobel filter, for instance. You don't want the training to change this operation, so these weights/filters should be kept constant.

Trainable parameters between input layer and first hidden layer: 6×12 + 12 = 84.
Trainable parameters between first and second hidden layers: 12×9+ 9 = 117.
Trainable parameters between second hidden layer and third layer: 9×5 + 5 = 50.
Trainable parameters between third hidden layer and output layer: 5×3 + 3 = 18.
Total number of trainable parameters of the neural net: 84+117+50+18 =269