IPCV
Gupta, Kush
FTDL - Take Home exam part- 3

Q1: Argue what family of deep neural networks would not be adequate to solve a classification problem involving 10000 available patterns with 20-coordinate input vectors and two target classes in which all patterns have a non-overlapping dimension, i.e., at least one of the input coordinates has a distinct minimum value for each class in the training set.

Ans:-> For the given classification problem, 10000 patterns with 20-coordinate input vectors and two target classes in which all patterns have a non-overlapping dimension. We can observe that the problem is linearly separable. Hence, any simple architecture like a simple neural network with no hidden layers or a simple multi-layer perceptron (MLP) model with any type of activation function can also be used to solve this problem. However, the convolutional neural network (CNNs) or Recurrent neural networks (RNNs) would probably fail because they utilize local patterns.

==========

Q2 a) Calculate the error in the last layer of a feed-forward neural network with a cross entropy cost function and a bipolar sigmoid transfer function f.

Ans:->

We know, cross entropy loss function.

$$C(y(x), t) = -\sum_{j=1}^{n_k} [t_j \log y_j + (1-t_j) \log(1-y_j)]$$

where, y is the model output, t is the expected output. Here $n = 2$.

∴ the gradient will be calculated as

$$\nabla_y C(y(x), t)_j = \frac{y_j - t_j}{y_j(1-y_j)}$$

⇒ the error on the last layer:-

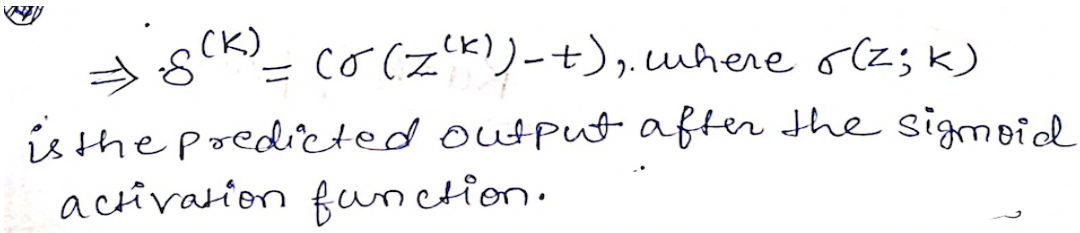
$$\delta_j^{(K)} = \frac{f(z_j) - t_j}{f(z_j)(1-f(z_j))} f'(z_j), \text{ where}$$

$$f(z_j) = \sigma(z_j) = \frac{1}{1+e^{-z_j}}$$

$$\Rightarrow f'(z_j) = \sigma(z_j)(1-\sigma(z_j))$$

b) Express it only in terms of F and the target values in the most simplified way so that it has minimal computational cost in a raw implementation.

Ans-> The simplified error function can be written as below.



c) Discuss how this error contributes to the vanishing gradient problem in the first layers.

Ans-> We can observe the vanishing gradient problem when the gradient of the loss function approaches zero and the weights (W) of the network do not vary significantly during the training of the model. When using the sigmoid activation function at the output layer of the network, we can observe that a huge change in the inputs will only cause a very small change in the outputs causing the derivatives to become smaller.

Q3. a) Specify a parallelization strategy that could be implemented for an N-neuron neural network if no parallelization libraries, TPUs or GPUs are available for such implementation but a multicore system is ready with a core number larger than the number of neurons in the network.

Ans-> For the implementation of an N-neuron neural network where no parallelization libraries, TPU or GPU are available. In that case, we can use the data parallelization method. In this method, the whole dataset is splitted into batches. All of these batches are sent to multiple processor cores. Each of the cores is loaded with a full replica of the model and runs its batch of training examples through the model. After which, the processor cores communicate with each other and share the results for backpropagation and to update the weights. In this case, the problem with respect to speed arises because of the communication between the processors. If a large number of processors are used, then they will spend more time communicating with each other and have to wait for each other if one of them is slower than the others.

b) Specify another parallelization strategy if just the opposite situation occurs: TPUs, GPUs, and parallelization libraries are available but the number of processors is considerable less than N. Mention speed bottleneck sources in both cases.

Ans-> When we have TPUs, GPUs, and parallelization libraries available, but the number of processors is considerably less than N. In this case we can use model parallelization where the neural network is divided and distributed among different processor cores of (GPU, TPU), not the training data. All the cores process the same training data. When the model has a high number of parameters, and multiple branches, this method works well. During the training process, when a part of the model is trained its outcome is synchronized with the next layer in another processor. The bottleneck with respect to speed, in this case, could be the time taken to transfer the data between the GPUs/TPUs.

Q4: a) What is the gradient of the cost function in hinge loss?
Ans->

We know, formula for hinge loss

$$l_{hinge} = \max(0, 1 - y x \cdot w)$$

We can calculate it's gradient by taking partial derivative w.r.t the weights, w_i

$$\frac{\partial l}{\partial w_i} = \begin{cases} -y \cdot x_i, & \text{if } (x \cdot w) < 1. \\ 0, & \text{otherwise} \end{cases}$$

b) How hinge loss will perform as a cost function for a deep feedforward neural network?
Ans-> The hinge loss is rarely used as a cost function for deep-feed forward neural networks because of the vanishing gradient problems, which can degrade the network performance. However, it can perform better instead of cross-entropy for some specific cases.

c) Will there be saturation problems?
Ans-> Yes, there will be saturation problems, as the derivative could be zero in many cases. (It can be observed in the equation in part a).

Q5: Reasonably propose and justify your design for:
(a) a network type and architecture,
Ans-> For the problem of Image Classification, the input to the network will be an image gray(2D) or color (3D). To process such data, we can build a convolutional neural network with max-pooling layers. I'll use kernels of typical sizes (3x3), and (5x5) for the convolutional layers followed by a max-pooling of 2x2. After that a network of fully connected hidden layers to get the output. The initial model would be with 3 convolutional
layers, however, the number of convolutional layers and hidden layers could be changed if the results are not as expected. In case, the model gets overfitted, we can use batch normalization and dropout layers in the network, to improve the results.

(b) activation functions,
Ans-> For the hidden layers I will use 'relu' as the activation function, and for the output, layer softmax can be used since it is a multiclass classification task.

(c) initialization strategy,
Ans-> weight initialization by normal distribution is almost always good.

(d) learning rule,
Ans-> In my opinion, the backpropagation strategy would be an appropriate choice for this task.

(e) a subdivision of training and test sets,
Ans-> As the provided dataset is large in my opinion, the distribution could be: 70% of the data as the training set, and the remaining 30% could be split into the validation set and the test set. Further, 20% of the remaining data could be used for the validation, and the remaining 10% of data can be used as the test set. Moreover, the data distribution should be stratified with respect to the classes so that each class could have an equal representation in the dataset.

(f) optimization options
Ans-> Initially, I would train the model with an Adam optimizer with a default learning rate of 0.001, as Adam works very well in most of the cases. Later, we can modify the learning rate to have a better model accuracy or we can change the optimizer function to SGD, or RMSprop, if required.

(g) an associated method to validate the learning in a classification problem that uses 100,000,000 patterns, each of which has 64,000 components (well-characterized images) in the input vectors and 4 components in the target output.
Ans-> In order, to validate the accuracy of the model we can calculate the validation accuracy and validation loss on the stratified validation set after every epoch.

(h) Indicate how would you deal with the problem of several highly unbalanced classes in the training.
Ans-> For the unbalanced classes scenario, we can compute the class weight by using the compute_class_weight() function and then use the class weights during the training process to balance out the classes which have a lesser weight by using the inverse ratio. I.e a higher weight is assigned to classes with a lower number of representatives and a lower weight will be assigned to classes with a higher number of representatives.