**VR Siddhartha Engineering College**

**M.Tech DataScience Second Semester External Exam August 2023**

**19ITDS2014B -Deep Learning**

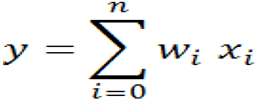
Answer one question from each unit Max. Marks :60

1. a. Summarize the process of neuron 9M

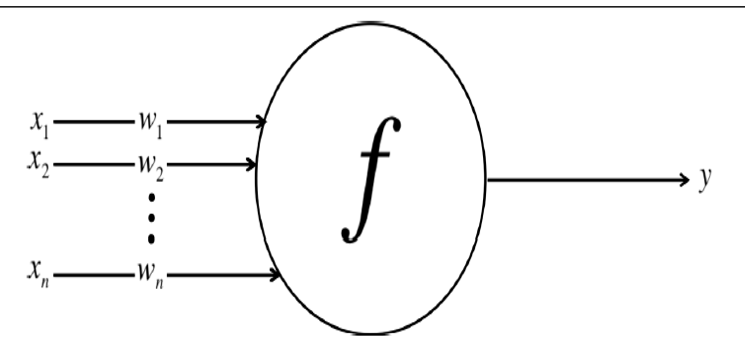
Explanation of biological neuron and structure with diagram : 4M

Artificial neuron structure with diagram: 5M

At its core, the neuron is optimized to receive information from other neurons, pro‐ cess this information in a unique way, and send its result to other cells. The neuron receives its inputs along antennae-like structures called dendrites. Each of these incoming connections is dynamically strengthened or weakened based on how often it is used (this is how we learn new concepts!), and it’s the strength of each connection that determines the contribution of the input to the neuron’s output. After being weighted by the strength of their respective connections, the inputs are summed together in the cell body. This sum is then transformed into a new signal that’s propagated along the cell’s axon and sent off to other neurons.

* The functional understanding of the neurons in our brain is translated into an artificial model . Artificial neuron takes in some number of inputs, *x1, x2, . . . , xn, each of which is multiplied by a specific* weight, *w1,w2, . . . ,wn. These weighted inputs are, summed together to* produce the *logit of the neuron. the logit also includes a bias, which is a constant. The logit is then passed through a function f to produce the output y = f (z) .*
* *This output can be transmitted to other neurons .*

A diagram of a cell

Description automatically generated

b. Discuss linear neurons and their limitations.6M

What islinear neuron -2M , Limitations-4M

A linear neuron is a neuron that use a linear function in the form of *f (z)*= *ax*+b as its activation function.

* The artificial neuron takes in some number of inputs, *x1, x2, . . . ,xn, each of which is multiplied by a specific* weight, *w1,w2, . . . ,wn.*

. Then we can re-express the outputof the neuron as

A black and white image of symbols

Description automatically generated with medium confidence

Where w0 is the bias term.

A neural network with perceptron acts as a linear neuron. It uses step function as its activation function.

A diagram of a function

Description automatically generated

Linear neurons are easy to compute with, but they run into serious limitations. The following are the disadvantages of linear neurons:

* The function’s derivative is a constant. That means there is constant gradient descent occurring since there is no relation to the value of z.
* A linear model is not really learning as it does not improve upon the error term, which is the whole point of the neural network.
* Since the activation is linear, nesting in 2 or N number of hidden layers with the same function will have no real effect. It can be shown that any feed-forward neural network consisting of only linearneurons can be expressed as a network with no hidden layers. This is problematic because, hidden layers are what enable us to learn importantfeatures from the input data.
* In other words, in order to learn complex relationships, we need to use neurons that employ some sort of nonlinearity.

**(OR)**

2.a. Describe feed forward neural networks and different activation functions that can be used in neurons. 8M

Feed forward neural network - 4M

Activation functions description with mathematical expression -4M

A diagram of a neural network

Description automatically generated

* Input Layer: The bottom layer of the network pulls in the input data.
* Output Layer: The top layer of neurons (output nodes) computes our final answer.
* Hidden layer: The middle layer of neurons in between input and output layers are called the *hidden layers.*
* Connections only traverse from a lower layer to a higher layer. There are no connections between neurons in the same layer, and there are no connections that transmit data from a higher layer to a lower layer. These neural networks are called feed-forward networks.
* Depending on the complexity of the problem, number of hidden layers may be more than 1. A neuron takes in some number of inputs, x1, x2, . . . ,xn, each of which is multiplied by a specific weight, w1,w2, . . . ,wn. They are summed together to produce the logit .logit also includes a bias, which is a constant. The logit is then passed through a function f to produce the output y = f (z).This output can be transmitted to other neurons .The output is calculated at each neuron in hidden and output layers using the following formula

A black and white image of symbols

Description automatically generated with medium confidence

Activation functions: Step, sigmoid, relu, tanh

b. Explain a layer which uses probability distribution and gives us a better idea of how confident we are in our predictions.7M

Discussion of output layer with multiple outputs- 2M, Discussion of Softmax – 5M

Softmax output layer:

Oftentimes, we want our output vector to be a probability distribution over a set of mutually exclusive labels. For example, if we want to build a neural network to recognize handwritten digits from the MNIST dataset. Each label (0 through 9) is mutually exclusive, but it’s unlikely that we will be able to recognize digits with 100% confidence. Using a probability distribution gives us a better idea of how confident we are in our predictions. As a result, the desired output vector is of the form below,

A math equations and formulas

Description automatically generated with medium confidence

This is achieved by using a special output layer called a somax layer. Unlike in other kinds of layers, the output of a neuron in a softmax layer depends on the outputs of all the other neurons in its layer. This is because we require the sum of all the outputs to be equal to 1. Letting zi be the logit of the i th softmax neuron, we can achieve this normalization by setting its output to:

A mathematical equation with numbers and letters

Description automatically generated with medium confidence

A strong prediction would have a single entry in the vector close to 1, while the remaining entries were close to 0. A weak prediction would have multiple possible labels that are more or less equally likely.

**UNIT-II**

3.a. Discuss how to minimize the squared error over all the training examples by minimizing the problem. 8M

Error function -1M, Gradient descent discussion -2M, Back propagation- 5M

The error function E for a neural network model will be given by



The squared error is zero when our model makes a perfectly correct prediction on every training example. Moreover, the closer E is to 0, the better our model is. As a result, our goal will be to select the parameter vector θ (the values for all the weights in our model) such that E is as close to 0 as possible.

* If we visualize how we might minimize the squared error over all of the training examples by simplifying the problem with two weights, *w1 and w2, then we can imagine a three-dimensional*  space where the horizontal dimensions correspond to the weights *w1 and w2, and the* vertical dimension corresponds to the value of the error function *E. i*f we consider the errors over all possible weights, we get a surface in this three-dimensional space in the shape of a quadratic bowl. The direction of the steepest descent is always perpendicular to the contours. This direction is expressed as a vector known as the *gradient.*

A diagram of a cone with lines and stars

Description automatically generated

Gradient Descent with sigmoidal neurons –In sigmoidal neuron, output of the neuron is computed from its inputs as:

A math equation with black and white text

Description automatically generated with medium confidence

* Derivative of logit Z wrt inputs and weights:

A mathematical equation with numbers

Description automatically generated with medium confidence

* Derivative of Output wrt logit is:

A mathematical equation with numbers and symbols

Description automatically generated

A mathematical equations on a white background

Description automatically generated

A white paper with black text

Description automatically generated

b. With a neat diagram explain detailed workflow for training and evaluating a deep learning model. 7M

Diagram - 3M, Explanation - 4M

A diagram of a program

Description automatically generated

**(OR)**

Discuss various techniques that have been proposed to prevent overfitting in deep neural networks during training process. 15M

Regularization – Desription -2M, L1-2M,L2-2M – (Total: 6M), Max norm- 2M,

Dropout-diagram -1M, explanation -3M -(Total: 4M), inverse dropout- 2M

1. Regularization:Regularization modifies the objective function that we minimize by adding additional terms that penalize large weights. In other words, we change the objective function so that it becomes Error + λ f (θ) , where f θ grows larger as the components of θ grow larger, and λ is the regularization strength.

a.L2 regularization. It can be implemented by augmenting the error function with the squared magnitude of all weights in the neural network. In other words, for every weight w in the neural network, we add 1/ 2λw2 to the error function.

b. L1 regularization. Here, we add the term λ| w| for every weight w in the neural network

2. Max norm constraints have a similar goal of attempting to restrict θ from becoming too large, but they do this more directly.Max norm constraints enforce an absolute upper bound on the magnitude of the incoming weight vector for every neuron and use projected gradient descent to enforce the constraint. In other words, any time a gradient descent step moves the incoming weight vector such that ||w ||> c, we project the vector back onto the ball (centered at the origin) with radius c.

3. Dropout While training, dropout is implemented by only keeping a neuron active with some probability p (a hyperparameter), or setting it to zero otherwise.

A diagram of a network

Description automatically generated

**UNIT-III**

~~5.a. Explain the following word2vec frameworks 8M~~

~~i. skipgram model , ii. noise contrastive estimation~~

~~Skipgram -4M, NCE- 4M~~

~~i.~~ **~~The Skip-Gram model~~** ~~takes the target word as an input, and then attempts to predict one of the words in the context. For example, in the sentence “the boy went to the bank.” If we broke this sentence down into a sequence of (context, target) pairs, we would obtain [([the, went], boy), ([boy, to], went), ([went, the], to), ([to, bank], the)]. Taking this a step further, we have to split each (context, target) pair into (input, output) pairs where the input is the target and the output is one of the words from the context. From the first pair ([the, went], boy), we would generate the two pairs (boy, the) and (boy, went). We continue to apply this operation to every (context, target) pair to build our dataset.~~

~~we replace each word with its unique index~~ *~~i~~* ~~∈0, 1, . . . ,~~ *~~V~~* ~~− 1 corresponding to its index in the vocabulary.~~

~~The structure of the encoder is surprisingly simple. It is essentially a lookup table with~~ *~~V~~* ~~rows, where the~~ *~~ith~~* ~~row is the embedding corresponding to the~~ *~~ith~~* ~~vocabulary word. All the encoder has to do is take the index of the input word and output the appropriate row in the lookup table.~~

~~The decoder is slightly trickier because we make some modifications for performance. The naive way to construct the decoder would be to attempt to reconstruct the one-hot encoding vector for the output, which we could implement with a runof-the-mill feed-forward layer coupled with a softmax. The only concern is that it’s inefficient because we have to produce a probability distribution over the whole vocabulary space.~~

~~To reduce the number of parameters, decoder known as noise-contrastive estimation (NCE) is implemented.~~

~~ii.~~**~~The NCE strategy~~** ~~uses the lookup table to find the embedding for the output, as well as embeddings for random selections from the vocabulary that are not in the context of the input. We then employ a binary logistic regression model that, one at a time, takes the input embedding and the embedding of the output or random selection, and then outputs a value between 0 to 1 corresponding to the probability that the comparison embedding represents a vocabulary word present in the input’s context.We then take the sum of the probabilities corresponding to the noncontext comparisons and subtract the probability corresponding to the context comparison.~~

b. Explain a method that is aggressively used to reduce dimensionality of feature maps and sharpen the located features. 7M

Max pooling – explanation -4M, Diagram -3M

To aggressively reduce dimensionality of feature maps and sharpen the located features, we sometimes insert a max pooling layer after a convolutional layer. The essential idea behind max pooling is to break up each feature map into equally sized tiles. Then we create a condensed feature map. Specifically, we create a cell for each tile, compute the maximum value in the tile, and propagate this maximum value into the corresponding cell of the condensed feature map. One interesting property of max pooling is that it is locally invariant. This means that even if the inputs shift around a little bit, the output of the max pooling layer stays constant.

A screenshot of a crossword puzzle

Description automatically generated

**(OR)**

6.a. Illustrate a situation in which PCA fails to optimally transform the data for dimensionality reduction. 8M

Explanation of case study: 6M , Diagram -2M

PCA fails to capture important relationships that are piecewise linear or nonlinear.

A diagram of a diagram

Description automatically generated

It highlights the shortcomings of an approach like PCA in capturing important relationships in complex datasets.The example shows data points selected at random from two concentric circles. Here, there is no linear direction that contains more information here than another (we have equal variance in all directions). Instead, as a human being, we notice that information is being encoded in a nonlinear way, in terms of how far points are from the origin. With this information in mind, we notice that the polar transformation (expressing points as their distance from the origin, as the new horizontal axis, and their angle bearing from the original x-axis, as the new vertical axis) does just the trick.

b. Describe an autoencoder in tensorflow model. 7M

Autoencoder concept- 2M , Diagram -3M, Decoder code explanation - 2M

Figure shows the experimental setup for dimensionality reduction of the MNIST dataset.the twodimensional embedding is now treated as the input, and the network attempts to reconstruct the original image. Because we are essentially applying an inverse operaion, we architect the decoder network so that the autoencoder has the shape of an hourglass. The output of the decoder network is a 784-dimensional vector that can be reconstructed into a 28 × 28 image.

A diagram of a flowchart

Description automatically generated

A white text with black text

Description automatically generated

A white background with black text

Description automatically generated

**UNIT IV**

7.a. Formulate the gradient in a Recurrent neural network with generalized back propagation algorithm 8M

RNN description with weights and diagram -4M, Back propagation derivation -4M

Considering an example of a recurrent network that maps an input sequence to an output sequence of the same length:

A diagram of a diagram

Description automatically generated

Here we assume the hyperbolic tangent activation function. Also, the figure does not specify exactly what form the output and loss function take. Here we assume that the output is discrete, as if the RNN is used to predict words or characters. . Forward propagation begins with a specification of the initial state h (0). Then, for each time step from t = 1 to t = τ , we apply the following update equations:

A screenshot of a computer

Description automatically generated

A black text on a white background

Description automatically generated

where the parameters are the bias vectors b and c along with the weight matrices U, V and W, respectively, for input-to-hidden, hidden-to-output and hidden to-hidden connections. The total loss for a given sequence of x values paired with a sequence of y values would then be just the sum of the losses over all the time steps. For example, if L (t) is the negative log-likelihood of y (t) given x (1) , . . . , x (t) , then

A white background with black text

Description automatically generated

where pmodel y (t) | {x (1) , . . . , x (t)} is given by reading the entry for y (t) from the model’s output vector yˆ (t)

The nodes of our computational graph include the parameters U, V , W, b and c as well as the sequence of nodes indexed by t for x (t) , h (t) , o (t) and L (t) . For each node N we need to compute the gradient ∇NL recursively, based on the gradient computed at nodes that follow it in the graph. We start the recursion with the nodes immediately preceding the final loss:

In this derivation we assume that the outputs o (t) are used as the argument to the softmax function to obtain the vector yˆ of probabilities over the output. We also assume that the loss is the negative log-likelihood of the true target y (t) given the input so far. The gradient ∇o(t)L on the outputs at time step t, for all i, t, is as follows:

A black and white image of a mathematical equation

Description automatically generated

We work our way backward, starting from the end of the sequence. At the final time step τ , h (τ) only has o (τ) as a descendent, so its gradient is simple:

 We can then iterate backward in time to back-propagate gradients through time, from t = τ − 1 down to t = 1, noting that h (t) (for t < τ ) has as descendents both o (t) and h (t+1). Its gradient is thus given by

A close-up of math equations

Description automatically generated

where diag (1 – (h (t+1)))2 ) indicates the diagonal matrix containing the elements 1 − (hi(t+1) )2. Once the gradients on the internal nodes of the computational graph are obtained, we can obtain the gradients on the parameter nodes.

the gradient on the remaining parameters is given by

A screenshot of a computer screen

Description automatically generated

A group of math equations

Description automatically generated

b. Describe how an RNN could respond to a directed graphical model over a sequence of random variables y(t) with no inputs x. 7M

Description with mathematical expressions -5M, Diagram -2M

* As a simple example, let us consider the case where the RNN models only a sequence of scalar random variables Y = *{*y(1)*, . . . ,* y(*τ*)*}*, with no additional inputs x.
* The input at time step *t* is simply the output at time step *t −*1. The RNN then defines a directed graphical model over the y variables.
* We parametrize the joint distribution of these observations using the chain rule for conditional probabilities:

A black and white math equation

Description automatically generated

* Hence the negative log-likelihood of a set of values *{y*(1)*, . . . , y*(*τ*)*}* according to such a model is A black and white math equation

  Description automatically generated WhereA black and white text

  Description automatically generated with medium confidence

Fully connected graphical model for a sequence *y*(1)*, y* (2)*, . . . , y*(*t*)*, . . .*:

* One way to interpret an RNN as a graphical model is to view the RNN as defining a graphical model whose structure is the complete graph, able to represent direct dependencies between any pair of y values.

A diagram of a graph

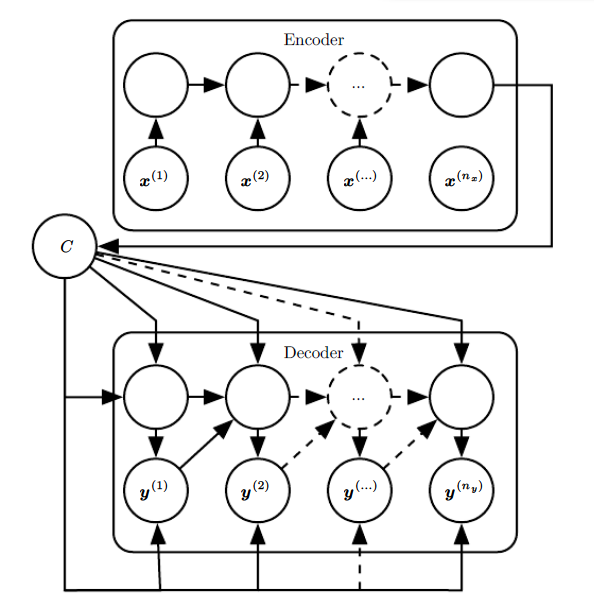
Description automatically generated

**(OR)**

8.a. Discuss how an RNN can be trained to map an input sequence to an output sequence which is not necessarily of the same length in application such as speech recognition. 9M

Description of autoencoder architecture -6M, Diagram -3M

* In many applications, such as speech recognition, machine translation or question answering, RNN can be trained to map an input sequence to an output sequence which is not necessarily of the same length. This architecture is called encoder-decoder or sequence-to-sequence RNN architecture. We call the input to the RNN the “context.” We want to produce a representation of this context, *C* . The context *C* might be a vector or sequence of vectors that summarize the input sequence *X* = (*x*(1)*, . . . , x*(*nx*)). an encoder or reader or input RNN processes the input sequence. The encoder emits the context *C* , usually as a simple function of its final hidden state. A decoder or writer or output RNN is conditioned on that fixed-length vector to generate the output sequence*Y* = (*y*(1) *, . . . , y*(*ny* ))



b. How recursive neural networks represent another generalization of recurrent network with a different kind of computational graph. 6M

Description -4M , Diagram -2M

Recursive neural networks represent yet another generalization of recurrent networks, with a different kind of computational graph, which is structured as a deep tree, rather than the chain-like structure of RNNs. Recursive networks have been successfully applied to processing *data structures* as input to neural nets in natural language processing and computer vision. One clear advantage of recursive nets over recurrent nets is that for a sequenceof the same length *τ*, the depth (measured as the number of compositions ofnonlinear operations) can be drastically reduced from *τ* to *O*(log *τ* ), which mighthelp deal with long-term dependencies.

A diagram of a tree

Description automatically generated

Dr.T.Anuradha Dr.T.Anuradha Dr.M.Suneetha

Course Coordinator Programme Coordinator HOD::IT