Assignment #: 1

Syed Alle Mustafa

October 5, 2022

1 Artificial Neurons

1. *Given a neuron with linear activation and weights 0.1, 0.2 and 0.3, find output for the input vector [1,1]*

   We are given three weights, $\theta_0 = 0.1, \theta_1 = 0.2 \theta_2 = 0.3$ and input vector $x_1 = 1, x_2 = 1$. By using the following formula,

   $$\hat{y} = f(\theta_1 + \theta_2 x_1 + \theta_2 x_2)$$
   $$\hat{y} = 0.1 + 0.2 * 1 + 0.3 * 1 = 0.6$$

2. *Given a linear discriminant function g(x) explain what should be its value on one side of the decision boundary, on the other side of the boundary and on the decision boundary.*

   On the one side of the decision boundary the value of discriminant function would be **g(x) greater than 0**, the value on decision boundary will be **g(x) equals to 0**, and on the other side of the decision boundary **g(x) would be less than zero**.

3. *Given the linear function $g(x) = \theta_0 + \theta_1 x_1 + \theta_2 x@$, explain meaning of coefficients $\theta_0, \theta_1, \theta_2$*

   In the linear function θ_0 is a constant weight or a bias for the neuron but in the given function, it is the distance from the origin, where $\theta_1 - \theta_n$ are called weights, which determine the influence of the feature or input on the output.

4. *Let $g(x) = 1 + 2x_1 + 3x_2$, find the normal to the decision boundary, defined by $g(x_1, x_2)$ and the distance of decision boundary from the origin.*

   The distance of the decision boundary from the origin is equals to $d = -\theta_0$, so, **the distance from the origin of the decision boundary would be -1.** where the normal vector from the decision boundary is $n = (\theta_1, \theta_2)$, so, **the normal vector would be (2,3).**

5. *In discriminant function $g(x) = \theta^T x$ where x and θ are vectors, explain where bias coefficient in θ is and what change is required in feature vector x to allow writing discriminant in this way.*

   The bias in this discriminant function is θ_0 and its the first entry in the weight's vector. And in the feature vector x the $x_0 is included which has value 1 and the first entry in feature vector.$

6. *Write the expressions for the step and logistic (sigmoid) activation functions. Explain the advantage of the sigmoid activation over step activation. What happens to the sigmoid when θ is small.*

   Following is the expression for the step function,

   $$h_0 = 1 if \theta^T x > 0$$

   $$h_0 = 0 otherwise$$

And the expression for sigmoid function is,

$$h_0(x) = 1/(1 + exp(-\theta^T x))$$

The advantage of sigmoid over step function is that the step function is very sharp and makes crisp decision which is 0 or 1. where sigmoid function gives a range between 0 and 1, If the θ is small in activation function then the sigmoid is known to be soft. where if the sigmoid becomes more like step function if the theta is large.

7. *Explain how the sigmoid is obtained using a linear function to model the log-likelihood ratio.*

   if we model log likelihood as linear discriminant, we get the following equation,

$$log((p(y = 1|x))/(p(y = 0|x))) = \theta^T x$$

   after taking exponential to remove log, and some simplifying we get,

$$p(y = 1|x) = exp(\theta^T x)/1 + exp(\theta^T x)$$

$$p(y = 1|x) = 1/(1 + exp(-\theta^T x)) = h_\theta$$

   So, using the sigmoid means using the log likelihood by modeling it as linear function, also the output of sigmoid mean the probability that the output belongs to class 1.

8. *Write the derivation of a sigmoid and use it to compute the derivative of log sigmoid*

   Derivative of sigmoid,

$$sigmoid(x) = 1/1 + exp(-\theta_T x)$$

$$dsigmoid(x)/dx = sigmoid(x)(1 - sigmoid(x))x$$

$$dsigmoid(x)/dx = h\theta(x)(1 - h\theta(x))x$$

   Now derivative of log sigmoid,

$$dlog(sigmoid())/dx = (1 - sigmoid(\theta^T x)) * x = (1 - h_\theta(x)) * x$$

9. Explain how to compute the direction used to updating parameters in gradient descent. How is the size of the update controlled.

   The direction is computed by looking locally by looking at the sharpest slope downwards, for this we subtract gradient from the previous solution into the learning rate by the gradient of the current solution. The size of update is controlled by the parameter called the learning rate which determines the step size of each update.

10. *Explain the stop condition for gradient descent. Should the condition use the loss change or parameter change.*

    We stop when difference between current gradient and previous gradient is less than the learning rate. To determine the step size i.e the learning rate we try and test different rates and choose the on with best loss without sacrificing speed of training. The condition should use the loss change because we don't know how much sensitive loss function is to the theta.

11. *Explain what happens when using a learning rate that is too large or too small.*

    If the learning rate is too small is might take alot time to get to the best loss value because our rate of change is very less, on the other side if it is too large, we might miss the optimal solution very often and it might also take many steps to get to best loss value.

12. *Explain how the empirical loss is computed/ what is the problem in using empirical loss in gradient loss.*

    Empirical loss is calculated by counting the examples that are predicted incorrectly. This loss function is a piece wise constant and we cannot minimize it using gradient descent. which is because changing the theta wont effect the loss function, which will make the gradient stuck at a specific point.

13. *Write the likelihood for a binary classifier and show how do develop it into binary cross-entropy.*

    Following is the likelihood of binary classifier,

$$\sum_{i=1}^{m} p(y = 1|x^i)^{yi} * p(y = 0|x^i)^{1-yi}$$

    To develop binary cross-entropy we take the log of likelihood of a binary classifier, and we get,

$$-\sum_{i=1}^{m} y^i log(h_\theta(x_i)) + (1 - y^i) log(1 - h_\theta(x_i))$$

14. *Write the gradient of the binary cross entropy loss function with respect to the parameter vector when using a sigmoid activation function. Write gradient descent update rule when using this loss function. Explain the meaning of the update equation you wrote.*

    Following is the gradient descent of the binary cross entropy,

$$Gradient = -\sum_{i=1}^{m} (y^i - h_\theta(x^i))x^i$$

    Its the difference between two predictions, if it is correct it has no effect, otherwise it contributes to the gradient. The gradient descent works here because h theta function here is calculated using sigmoid.

15. *Explain two strategies of multiclass classification (One against all other, One against each other), in which category its easier to discriminate the data.*

    In both cases, we have layer of neurons. In the first one, we have k classes and k neurons, k discriminant functions. Each neuron classify one class against all other, we measure the distance from all the classes and choose the best distance (probability) as the decision. Another approach is to classify one class against each other, we have k(k+1)/2 discriminant functions, to separate each class from every other, this method is easier/efficient than one against all other but we usually use the prior one.

16. *Explain the template matching interpretation of linear discriminant. Using this interpretation what is the meaning of using multiple linear discriminant function.*

    In template matching we measure the similarity between template Θ and the feature vector X, for this we take the dot product between them, if the result is high then there is a great similarity otherwise the template does not match. For multiclass we can use one hot encoding and for every class we can calculate the similarity of the feature vector.

17. *Explain the need of using softmax in output layer of a classifier.*

    Softmax function is used for the output layer, to ensure that the sum of the probabilities of classes as output are equal to 1.

18. *Write the derivative of softmax and use it to compute the derivative of log softmax.*

$$= softmax(\theta_j^T x)(learningRate_{ij} - softmax(\theta_i^T x)x$$

19. *Write likelihood for a multiclass classifier and show how to develop it into categorical cross entropy.*

    following is the likelihood of multiclass classifier,

$$= \sum_{i=1}^{m} \sum_{j=1}^{k} p(y = j|x^i)^{1|y^i=j|}$$

    this was likelihood of multiclass classifier, we take the negative log of it and get categorical cross entropy.

$$\sum_{i=1}^{m} \sum_{j=1}^{k} 1|y^i = j|logp(y = j|x^i)$$

    where $1|y^i = j|$ is the indicator function of the class.

20. *Write the gradient of the categorical cross-entropy loss function with respect to the parameter vector when using a softmax activation function. Write the gradient descent update rule when using this loss function. Explain the meaning of the update equation you wrote.*

    following is the gradient of the categorical cross entropy of kth class,

$$dL(\theta)/d\theta_j = \sum_{i=1}^{m} (h_{\theta j}(x^i) - 1(y^i = j))x^i$$

    So, the gradient descent update rule becomes,

$$theta_j = \theta_j - learningRate \sum_{i=1}^{m} (h_{\theta j}(x^i) - 1(y^i = j))x^i$$

    This equation has feature vector into the examples that were determined incorrectly and the gradient's difference. This is done for each neuron which classify one class out of the total classes.

2 Neural Networks

1. *Given a two later network where the number of hidden units is larger than the number of inputs, Explain the dimensionality increase interpretation of the hidden layer. Explain what is possible benefit for such dimension increase.*

   let suppose we have input vector with n dimensions and we map it to h units of the hidden layer, if h is bigger than n, then we are increasing the dimension and so we move to the higher dimension space than the dimensions given in the input vector. It is used when we want non linear classification. It mostly depends upon the data whether we want dimension reduction or increase.

2. *Given a two later network where the number of hidden units is smaller than the number of inputs, Explain the dimensionality decrease interpretation of the hidden layer. Explain what is possible benefit for such dimension decrease.*

   When the vector of input vector is larger than the dimension given by the hidden layer, it is said to be dimension reduction. The benefit of dimensionality reduction is that it simplifies the data.

3. *Given the update equation for the output layer in a 2 layer network. Explain why we cannot use directly the same equation for hidden layer. Assume both layers have the same activation function.*

   We cannot use the same equation because in case of input layer we know what the input feature vector is, but in case of hidden layer we are not sure what the input (z) is since it is the output of input vector which is unknown to us. Also we dont know the weights as well.

4. *Apply the chain rule to compute the gradient with respect to both hidden layer and the output layer parameters of a 2 layer regression network with a single output where the hidden layers use sigmoid activation and the output layer uses linear activation.*

   Following is the chain rule to compute the gradient with respect to both hidden layer and output regression.

   Suppose E is the loss of the final layer and v is the weight of final layer. w as the weight of input layer and z as the output of input layer, so,

   $$dE/dv = dE/d\hat{y} * d\hat{y}/dv$$

   $$dE/dw = dE/d\hat{y} * d\hat{y}/dz * dz/dw$$

   Assume we are using MSE, so applying derivatives we get,

   $$dE/dw = \sum_{i=1}^{m} (\hat{y}_i - y_i) * v * z(1-z)x$$

   $$dE/dv = \sum_{i=1}^{m} (\hat{y}_i - y_i) * z$$

5. *Apply the chain rule to compute the gradients with respect to both hidden layer and output layer parameters of a 2 layer regression network with multiple outputs where the hidden layers use sigmoid activation and the output use linear activation.*

Suppose E is the loss of the final layer and v is the weight of final layer. w as the weight of input layer and z as the output of input layer, We have k classes so j is the iteration of those classes. So,

$$dE/dv_j = dE/d\hat{y_j} * d\hat{y_j}/dv_j$$

$$dE/dw_j = dE/d\hat{y_j} * d\hat{y_j}/dz_j * dz_j/dw_j$$

Assume we are using MSE, so applying derivatives we get,

$$dE/dw_j = \sum_{j=1}^{k}\sum_{i=1}^{m}(\hat{y_{ij}} - y_{ij}) * v_j * z_j(1 - z_j)x$$

$$dE/dv_j = \sum_{i=1}^{m}(\hat{y}_i - y_i) * z_j$$

6. *Apply the chain rule to compute the gradients with respect to both hidden layer and output layer parameters of a 2 layer binary classification network with multiple outputs where the hidden layers use sigmoid activation and the output use sigmoid activation.*

$$dE/dv = dE/d\hat{y} * d\hat{y}/dv$$

$$dE/dw = dE/d\hat{y} * d\hat{y}/dz * dz/dw$$

Applying the values of derivatives,

$$dE/dv = \sum_{i=1}^{m}(\hat{y}_i - y_i) * x * \hat{y}_i(1 - \hat{y}_i) * z_i$$

$$dE/dw = \sum_{i=1}^{m}(\hat{y}_i - y_i) * x * \hat{y}_i(1 - \hat{y}_i) * v * z(1 - z) * x$$

7. *Apply the chain rule to compute the gradients with respect to both hidden layer and output layer parameters of a 2 layer multi class classification network with multiple outputs where the hidden layers use sigmoid activation and the output use softmax activation.*

$$dE/dv_j = dE/d\hat{y_j} * d\hat{y_j}/dv_j$$

$$dE/dw_j = dE/d\hat{y_j} * d\hat{y_j}/dz_j * dz_j/dw_j$$

Applying the values of derivatives,

$$dE/dvj = \sum_{j=1}^{k}\sum_{i=1}^{m}(\hat{y_j^i} - 1(y_j^i = j) * z_{ij}) * \hat{y_{ij}}(1_{ij} - \hat{y}_i) * z_{ij}$$

$$dE/dw = \sum_{j=1}^{k}\sum_{i=1}^{m}(\hat{y_j^i} - 1(y_j^i = j) * z_{ij}) * \hat{y_{ij}}(1_{ij} - \hat{y}_i) * v_{ij} * z_{ij}(1 - z_{ij})x$$

8. *Describe how the weights in the network should be initialized and explain why they are initialized in this way.*

   The weights are initialized from the values randomly drawn from uniform or gaussian distribution i.e between (0,1), (-1,1) or(-0.3,0.3). This is because of expectations of the gradient descent algorithm, which then finds the perfect weight.

3 Computational Graphs

1. *Explain the advantage of using computational graphs. Describe what is computed during the forward pass of the network and the backward pass of the network during back propagation. Explain what each node in graph needs to be able to compute and what information it needs to store and why.*

   Computing the gradient manually using the chain rule becomes very difficult when the network becomes dense and deeper because of the summation of classes. So manually we can see the weight updates of each iteration. In forward pass of the network, the activation functions are applied and the predicted value i.e $\hat{y}$ is calculated, where as in backward pass the gradient descent is done and the weights are optimized using it. On forward pass the node computes the value of function that could be activation/loss/multiplication function and store its value, where in backward pass we compute the derivative of that function in reverse order. We store the values in nodes which help later to compute the gradient upon backward pass.

2. *Given a two layer network where the hidden layer output is given by $z = f_1(W_1, x)$ and the output layer is given by $\hat{y} = f_2(W_2, z)$ express the output $\hat{y}$ as a function composition between $f_1()$ and $f_2()$. Assuming the output layer has a single node, express the L_2 loss for a batch $x^i, y^i{}_{i=1}^m$ respect to W_1 and W_2. You do not need to compute the actual derivatives of $f_1()$ and $f_2()$ as they are not specified. Repeat the question with the cross entropy loss instead of L_2.*

$$\hat{y}(x) = f_2(W2 f_1(W1, x))$$

$$L2 = L(\delta(xW_1)W_2, y)$$

$$dE/dv = dE/d\hat{y} * d\hat{y}/dv$$
$$dE/dw = dE/d\hat{y} * d\hat{y}/dz * dz/dw$$

3. *Repeat the previous question for a multiple output network.*

$$dE/dv_j = dE/d\hat{y_j} * d\hat{y_j}/dv_j$$
$$dE/dw_j = dE/d\hat{y_j} * d\hat{y_j}/dz_j * dz_j/dw_j$$

4. *Write the computation graph for a two layer regression network with 2 nodes in the hidden layer and 1 node in the output layer. Assume sigmoid activation in the hidden layer and linear activation in the output layer. Assume the feature vectors are 3-dimensional. List all the inputs of the computation*

graphs and write all the derivatives needed to compute the loss gradient with respect to network parameters when using the back-propagation pass.

$$a = W * x + b_1$$
$$z = Sigmoid(a)$$
$$s = V * z + b_2$$
$$h = Sigmoid(s)$$
$$\hat{y} = U * h + b_3$$

$$dE/dv = dE/d\hat{y} * d\hat{y}/dv$$
$$dE/dw = dE/d\hat{y} * d\hat{y}/dz * dz/dw$$
$$dE/du = dE/d\hat{y} * d\hat{y}/dh * dh/du$$

5. *Describe the back flow patterns for the following nodes: addition of a constant, addition of two inputs, multiplication by a constant, multiplication of two inputs, max of two inputs, min of two inputs.*

   (a) *Addition of a constant:* if z = x+c, then d(X) = 1 * dz

   (b) *Addition of two inputs:* if z = x + y, then dx = 1 * dz and dy = 1 * dz

   (c) *Multiplication of a constant:* if z = x*c, then dx = dz*c

   (d) *Multiplication of two inputs:* if z = x * y, then for back flow, dx = y* dz and dy = x*dz.

   (e) *Max of two inputs:* z = x if(x¿y) or y if (y¿x) dx = 1 if (x¿=y) else 0 dy = 1 if (y¿=x) else 0

   (f) *Min of two inputs:* z = x if(x¡y) or y if (y¡x) dx = 1 if (x¡=y) else 0 dy = 1 if (y¡=x) else 0

6. *Explain what derivative you expect when the input to a node is a vector and the output is a scalar. Write all the components of derivative.*

   feature vector x, weight vector w, bias b and scalar output y.

   $d\hat{y}/db = scalar$

   $d\hat{y}/dw = vector$

   $d\hat{y}/dx = vector$

7. *Explain what derivative you expect when the input to a node is a vector and the output is a vector. Write all the components of derivative.*

   feature vector x, weight vector w, bias b and vector output y.

   $d\hat{y}/db = Rank1(1+0)tensor$

   $d\hat{y}/dw = Rank2(1+1)tensor$

   $d\hat{y}/dx = Rank2(1+1)tensor$

8. *Explain what derivative you expect when the input to a node is a matrix and the output is a scalar. Write all the components of derivative.*

   feature matrix x, weight vector w, bias b and vector output y.

   $d\hat{y}/db = Rank1(1+0)tensor$

   $d\hat{y}/dw = Rank2(1+1)tensor$

   $d\hat{y}/dx = Rank2(0+2)tensor$

9. *Write the chain rule for the composition of vector valued vector functions F and G. Pay attention to the order of the Jacobian matrices you write.*

   let F be a function with y where y E $R^m -> R^p$ and G be a function with x where x E $R^n -> R^m$, if F and G are differentiable, Then the function F.G is differentiable and

$$D(F.G)(x,y) = D_F(G(X,y)D_G(x,y)$$

10. *Let F(x,y,z) = [3xy y-z]^T and G(x,y)= [x-5y xy x-y]^T. Write the composition f.g(x,y). Compute the jacobian matrix of the composition in two ways, directly and by using the chain rule.*

$$F.G = D_F(G(X,y)D_G(x,y)$$

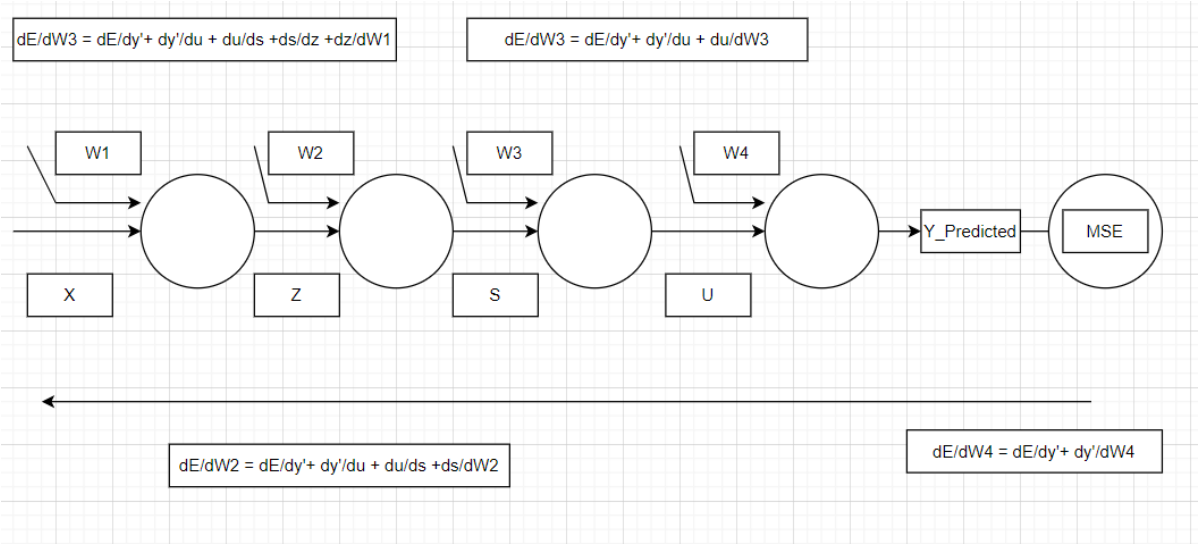
   for part 2, DG(x,y) = [[1 y 1][5 x 0]]

   DF(G(x,y)) = [[3 0] [3 1]] .[x-5y xy x-y]^T

11. *Explain the need to order nodes when traversing the computation graph in the forward and backward passes.*

    The ordering of nodes is important because of the dimensions of the output vector $\hat{y}$. if the nodes are ordered incorrectly with wrong weights our output wont be in the desired state.

12. *Draw the computation graph for a regression network with three hidden units and one output unit. Assume logistic activation at hidden layer nodes and linear activation at output. You may use logistic nodes without having to break them up into elementary computations. Assume L2 loss. Compute the values at the network for forward pass for each data point.*



x.w^T = [1 1 2][2 3 2].[0.01 0.02 0.03]^T z = [0.03 0.08]

z' = [0.03 0.08] =¿ [0.507 0.519] (after sigmoid)

z'. w2 = [0.52 0.53] [0.03 0.01 0.02]

s = [0.01521 0.00507 0.01014][0.01557 0.00519 0.01038]

s' = [0.503 0.501 0.502][0.503 0.501 0.502]

s' * w3 = [0.025080.02508][0.025080.02508] u' = [0.5062 0.5062] after sigmoid.

c. Given in the diagram.

13.

compute Jacobian matrix of given matrix.

Compute D(F.G)(2) with and without chain rule.*Vector computation graph: a. Let f(x,y) = (2x+3y)^2 compute gradient.*

compute Jacobian matrix of given matrix.

Compute D(F.G)(2) with and without chain rule.

a. [4(2x+3y) 6(2x+3y)]

b. [2x 3][2 8y]

c. F.G=DF(G(X))DG(x)

DG(x) = [1 2x]

DF(G(X)) = [2x 3][2 8y] . [x x^2]

D(F.G) = [2x^2+4x 3x^2+8x^{2}y].[1 2x]

14. Assume a two layer classification network with a single output where the hidden units and output unit use a sigmoid function. Assume a training set $x^i, y^{i^m}_{i=1}$ where $x^i ER^n$ and y^i E 0,1. Assuming L2 loss. derive the backpropagation update equations for the output weights in a similar way.

dL/dŷ =

$$dE/dv = \sum_{i=1}^{m} (\hat{y}_i - y_i) * x * \hat{y}_i(1 - \hat{y}_i) * z_i$$

$$dE/dw = \sum_{i=1}^{m} (\hat{y}_i - y_i) * x * \hat{y}_i(1 - \hat{y}_i) * v * z(1 - z) * x$$

$$w = w - (learningrate * dE/dw)$$

$$v = v - (learningrate * dE/dv)$$