1 Joby George

2 HW 1

3 DS GA 1008

3.1 Due 9/29/22

4 Section 1

4.1 1.1

You are given the following neural net architecture:

$$Linear_1 \rightarrow f \rightarrow Linear_2 \rightarrow g \tag{1}$$

where $Linear_i(x) = W^{(i)}x + b^{(i)}$ is the i-th affine transformation, and *f,g* are element-wise nonlinear activation functions. when an input x $\in R^n$ is fed to the network, $\hat{y} \in R^k$ is obtaied as the output

4.2 1.2

We would like to perform the regression task. we choose f(.) = 5ReLU(.) and g to be the identity function. To train the network, we chose MSE loss function:

$$l_{MSE}(\hat{y}, y) = ||\hat{y} - y||^2 \tag{2}$$

where y is the target output.

4.2.1 A

Name and mathematically describe the 5 programming steps you would take to train this model with PyTorch using SGD on a single batch of data.

4.2.2 Answer:

```
    1: You would complete the forward pass: output = model(inp
ut)
    2: You would calculate the training loss with respect to t
he ground truth: J = loss(output, label)
    3: You would zero out existing gradients so they are not a
ccumulated: model.zero_grad()
    4: You would run the backwards pass to calculate the gradi
ent: J.backward()
    5: You would update the model weights: optimiser.step()
```

4.2.3 B

For a single datapoint(x,y),write down all inputs and outputs for forward pass of each layer. You can only use variable x, y, $W^{(1)}$, $b^{(1)}$, $W^{(2)}$, $b^{(2)}$ in your answer. (note that $Linear_i(x) = W^{(i)}x + b^{(i)}$)

4.2.4 Answer

The overall set of steps can be defined procedeurally as:

$$Input\ layer : x \in R^n \rightarrow Linear_1 \rightarrow f \rightarrow Linear_2 \rightarrow g \tag{3}$$

$$z_1 = Linear_1 = W^{(1)}x + b^{(1)} \tag{4}$$

$$z_2 = f(z_1) = 5ReLu(z_1) \tag{5}$$

$$z_3 = Linear_2(z_2) = W^{(2)}z_2 + b^{(2)} \tag{6}$$

$$\hat{y} = g(z_3) = I_k z_3 \tag{7}$$

$$L(\hat{y}, y) = \frac{1}{N}||\hat{y} - y||^2 \tag{8}$$

4.2.5 C

Write down the gradients calculated from the backward pass. You can only use the following variables:x, y, $W^{(1)}$, $b^{(1)}$, $W^{(2)}$, $b^{(2)}$, $\frac{\partial l}{\partial \hat{\mathbf{y}}}$, $\frac{\partial z_3}{\partial z_1}$, $\frac{\partial \hat{y}}{\partial z_3}$ in your answer.

4.2.6 Answer:

The derivatives in the backpropogation are the change in our loss function with respect to our weights, represented by: $\frac{\partial l}{W^{(k)}}$and $\frac{\partial l}{b^{(k)}}$

$$\frac{\partial l}{W_2} = \frac{\partial l}{\partial \hat{y}} * \frac{\partial \hat{y}}{z_3} * \frac{\partial z_3}{\partial W_2} \tag{9}$$

$$\frac{\partial l}{b_2} = \frac{\partial l}{\partial \hat{y}} * \frac{\partial \hat{y}}{z_3} * \frac{\partial z_3}{\partial b_2} \tag{10}$$

$$\frac{\partial l}{W_1} = \frac{\partial l}{\partial \hat{y}} * \frac{\partial \hat{y}}{z_3} * \frac{\partial z_3}{\partial z_2} * \frac{\partial z_2}{\partial z_1} * \frac{\partial z_1}{\partial W_1} \tag{11}$$

$$\frac{\partial l}{b_1} = \frac{\partial l}{\partial \hat{y}} * \frac{\partial \hat{y}}{z_3} * \frac{\partial z_3}{\partial z_2} * \frac{\partial z_2}{\partial z_1} * \frac{\partial z_1}{\partial b_1} \tag{12}$$

4.2.7 D

Show us the elements of:

$$1.D.\ A: \frac{\partial l}{\partial \hat{\mathbf{y}}} \tag{13}$$

$$1.D.\ B: \frac{\partial \hat{y}}{\partial z_3} \tag{14}$$

$$1.D.\ C: \frac{\partial z_3}{\partial z_2} \tag{15}$$

(be careful about the dimensionality).

4.2.8 1.D.A

$$L(\hat{y}, y) = \frac{1}{N}||\hat{y} - y||^2 \tag{16}$$

$$\frac{\partial l}{\partial \hat{\mathbf{y}}} = \frac{1}{N} < \hat{y} - y >^T < \hat{y} - y > \frac{d}{\partial \hat{\mathbf{y}}} \tag{17}$$

$$\frac{\partial l}{\partial \hat{\mathbf{y}}} = \frac{1}{N}(\hat{y}^2 - 2\hat{y}y + y^2 \frac{d}{\partial \hat{\mathbf{y}}}) \tag{18}$$

$$\frac{\partial l}{\partial \hat{\mathbf{y}}} = \frac{2}{N}(\hat{y} - y) \tag{19}$$

This partial has a dimension of R^k (the number of classes)

4.2.9 1.D.B

$$\hat{y} = I_k z_3 \tag{20}$$
$$\frac{\partial \hat{y}}{\partial z_3} = I_k z_3 \frac{d}{\partial z_3} \tag{21}$$
$$\frac{\partial \hat{y}}{\partial z_3} = I_k \tag{22}$$

This partial has a dimension of R^{kxk}.

4.2.10 1.D.C

$$z_3 = W^{(2)} z_2 + b^{(2)} \tag{23}$$
$$\frac{\partial z_3}{\partial z_2} = W^{(2)} z_2 + b^{(2)} \frac{d}{\partial z_2} \tag{24}$$
$$\frac{\partial z_3}{\partial z_2} = W^{(2)T} \tag{25}$$

This partial has a dimension of $R^{(kxd)}$where d is the number of columns in W_2 This is equivalent to a matrix with dimmensions

$$\begin{matrix} \frac{\partial z_{31}}{\partial z_{21}} & .. & \frac{\partial z_{31}}{\partial z_{2K}} \\ \frac{\partial z_{32}}{\partial z_{21}} & .. & \frac{\partial z_{32}}{\partial z_{2K}} \\ \frac{\partial z_{3N}}{\partial z_{21}} & .. & \frac{\partial z_{31}}{\partial z_{2K}} \end{matrix}$$

4.3 1.3

We would like to perform multi-class classification task, so we set f = tanh and g = σ the logistic sigmoid function:

$$\sigma(z) = (1 + exp(-x))^{-1} \tag{26}$$

4.3.1 A

If you want to train this network, what do you need to change in the equations of:

1.2.B
1.2.C
1.2.D

assuming we are using the same MSE loss function

4.3.2 Answer for 1.2.B

While the network's computational diagram remains the same for the feedforward functions, the activation functions have changed giving us:

$$Input\ layer : x \in R^n \rightarrow Linear_1 \rightarrow f \rightarrow Linear_2 \rightarrow g \tag{27}$$

$$z_1 = Linear_1 = W^{(1)}x + b^{(1)} \tag{28}$$

$$z_2 = f(z_1) = tanh(z_1) \tag{29}$$

$$z_3 = Linear_2(z_2) = W^{(2)}z_2 + b^{(2)} \tag{30}$$

$$\hat{y} = g(z_3) = \sigma(z_3) \tag{31}$$

$$L(\hat{y}, y) = ||\hat{y} - y||^2 \tag{32}$$

4.3.3 Answer for 1.2.C

The backpropogatiaon algorithm finds how changes in our inputs and parameters impacts our loss function. Because the computational graph is still the same the partial derivatives calculated during the backpropagation algorithm remain the same, namely:

$$\frac{\partial l}{W_2} = \frac{\partial l}{\partial \hat{y}} * \frac{\partial \hat{y}}{z_3} * \frac{\partial z_3}{\partial W_2} \tag{33}$$

$$\frac{\partial l}{b_2} = \frac{\partial l}{\partial \hat{y}} * \frac{\partial \hat{y}}{z_3} * \frac{\partial z_3}{\partial b_2} \tag{34}$$

$$\frac{\partial l}{W_1} = \frac{\partial l}{\partial \hat{y}} * \frac{\partial \hat{y}}{z_3} * \frac{\partial z_3}{\partial z_2} * \frac{\partial z_2}{\partial z_1} * \frac{\partial z_1}{\partial W_1} \tag{35}$$

$$\frac{\partial l}{b_1} = \frac{\partial l}{\partial \hat{y}} * \frac{\partial \hat{y}}{z_3} * \frac{\partial z_3}{\partial z_2} * \frac{\partial z_2}{\partial z_1} * \frac{\partial z_1}{\partial b_1} \tag{36}$$

4.3.4 Answer for 1.2.D

Since we have changed the activation functions, we have some new gradients, however, because the loss function and linear layers remain the same $\frac{\partial l}{\partial \hat{\mathbf{y}}}$ and $\frac{\partial z_3}{\partial z_2}$ are the same as 1.2.D.

$$L(\hat{y}, y) = \frac{1}{N}||\hat{y} - y||^2 \tag{37}$$

$$\frac{\partial l}{\partial \hat{\mathbf{y}}} = \frac{1}{N} < \hat{y} - y >^T < \hat{y} - y > \frac{d}{\partial \hat{\mathbf{y}}} \tag{38}$$

$$\frac{\partial l}{\partial \hat{\mathbf{y}}} = \frac{1}{N}(\hat{y}^2 - 2\hat{y}y + y^2 \frac{d}{\partial \hat{\mathbf{y}}}) \tag{39}$$

$$\frac{\partial l}{\partial \hat{\mathbf{y}}} = \frac{2}{N}(\hat{y} - y) \tag{40}$$

$$z_3 = W^{(2)} z_2 + b^{(2)} \tag{41}$$

$$\frac{\partial z_3}{\partial z_2} = W^{(2)} z_2 + b^{(2)} \frac{d}{\partial z_2} \tag{42}$$

$$\frac{\partial z_3}{\partial z_2} = W^{(2)T} \tag{43}$$

$\frac{\partial \hat{y}}{\partial z_3}$ changes, and the new partial is claculated below:

$$\frac{\partial \hat{y}}{\partial z_3} = \sigma(z_3) \frac{d}{\partial z_3} \tag{44}$$

$$\frac{\partial \hat{y}}{\partial z_3} = (1 + e^{-z_3})^{-2} * \frac{d}{\partial z_3}(1 + e^{-z_3}) \tag{45}$$

$$\frac{\partial \hat{y}}{\partial z_3} = \frac{e^{-z_3}}{(1 + e^{-z_3})^{-2}} \tag{46}$$

Re-expressing this we get

$$\frac{\partial \hat{y}}{\partial z_3} = \frac{1}{1 + e^{-z_3}} * \frac{e^{-z_3}}{1 + e^{-z_3}} \tag{47}$$

Using a clever trick, we add and subtract one to the second fraction giving us

$$\frac{\partial \hat{y}}{\partial z_3} = \frac{1}{1 + e^{-z_3}} * \frac{(e^{-z_3} + 1) - 1}{1 + e^{-z_3}} \tag{48}$$

Now we can re-express the second fraction as:

$$\frac{\partial \hat{y}}{\partial z_3} = \frac{1}{1 + e^{-z_3}} * [1 - \frac{1}{1 + e^{-z_3}}] \tag{49}$$

$$\frac{\partial \hat{y}}{\partial z_3} = \sigma(z_3)(1 - \sigma(z_3)) \tag{50}$$

It is important to note that this matrix is a **diagonal** matrix R^{kxk} that follows this pattern:

$$\frac{\partial \hat{y}}{\partial z_3} = \begin{cases} \sigma(z_3)(1 - \sigma(z_3)) & \text{if } i == j \\ 0 \text{ if } i != j \end{cases}$$

4.3.5 B

Now you think you can do a better job by using a Binary Cross Entropy (BCE) loss function

$$l_{BCE}(\hat{y}, y) = \frac{1}{K}\sum_{i=1}^{K}(-y_i log(\hat{y_i})) + [(1-y_i)log(1-\hat{y_i})] \tag{51}$$

What do you need to change in the equations of (b), (c) and (d)

4.3.6 Answer 1.2.B

We would change the last layer to take into account the new loss function

$$Input\ layer : x \in R^n \rightarrow Linear_1 \rightarrow f \rightarrow Linear_2 \rightarrow g \tag{52}$$

$$z_1 = Linear_1 = W^{(1)}x + b^{(1)} \tag{53}$$

$$z_2 = f(z_1) = tanh(z_1) \tag{54}$$

$$z_3 = Linear_2(z_2) = W^{(2)}z_2 + b^{(2)} \tag{55}$$

$$\hat{y} = g(z_3) = \sigma(z_3) \tag{56}$$

$$L(\hat{y}, y) = \frac{1}{K}\sum_{i=1}^{K}(-y_i log(\hat{y_i})) + [(1-y_i)log(1-\hat{y_i})] \tag{57}$$

4.3.7 Answer 1.2.C

The overall backpropogation algorithim computes the same partials, with respect to the same variables as previously. The difference is that $\frac{\partial l}{\partial \hat{y}}$ will now have changed, which will be shown in the next part.

4.3.8 Answer 1.2.D

$$\frac{\partial L}{\partial \hat{y}} = \frac{1}{K}\sum_{i=1}^{K}(-y_i log(\hat{y_i})) + [(1-y_i)log(1-\hat{y_i})]\frac{\partial d}{\hat{y}} \tag{58}$$

$$\frac{\partial L}{\partial \hat{y}} = \frac{1}{K} * \frac{\partial d}{\hat{y}}(-ylog(\hat{y})) - \frac{\partial d}{\hat{y}}[(1-y)log(1-\hat{y})] \tag{59}$$

Applying log derivative rules we get:

$$\frac{\partial L}{\partial \hat{y}} = \frac{1}{K} * \frac{-y}{\hat{y}} + \frac{(1-y)}{(1-\hat{y})} \tag{60}$$

4.3.9 C

Things are getting better. You realize that not all intermediate hidden activations need to be binary (or soft version of binary). You decide to use f (·) = (·)+ but keep g as σz_3. Explain why this choice of f can be beneficial for training a (deeper) network.

4.3.10 Answer

The choice of ReLu as an activation function in comparison with the tanh function will help training a deeper network because the gradient will persist through the backpropogation algorithm. Since the derivative of ReLu(x) is:

$$\nabla ReLu(z) = \begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{if } z \leq 0 \end{cases}$$

In comparison: $\nabla tanh(z) = \begin{cases} 0+ & \text{if } z \leq -5 \text{ or } z \geq 5 \\ (0, 1] & \text{if } z \in [-5, 5] \end{cases}$

Thus our gradient will be close to 0 if our input to the activation function takes a large positive or negative value

4.4 1.4

4.4.1 A

Why is softmax actually softargmax?

4.4.2 Answer:

Goodfellow, Bengio and Courville explain this succintly in [page 183 of Deep Learning (https://www.deeplearningbook.org/contents/mlp.html)](https://www.deeplearningbook.org/contents/mlp.html).

The function is more similar to the arg max function than the max function. The term “soft” derives from the fact that the softmax function is continuous and differentiable as the arg max function is represented as a one-hot vector, and is not continuous nor differentiable. The softmax function thus provides a “softened” version of the arg max.

To get the more literal definition of softmax, we would perform:

$$softmax(z) = softargmax(z)^T z \tag{61}$$

This would give us a differentiable way of calculating the maximum value of our input array.

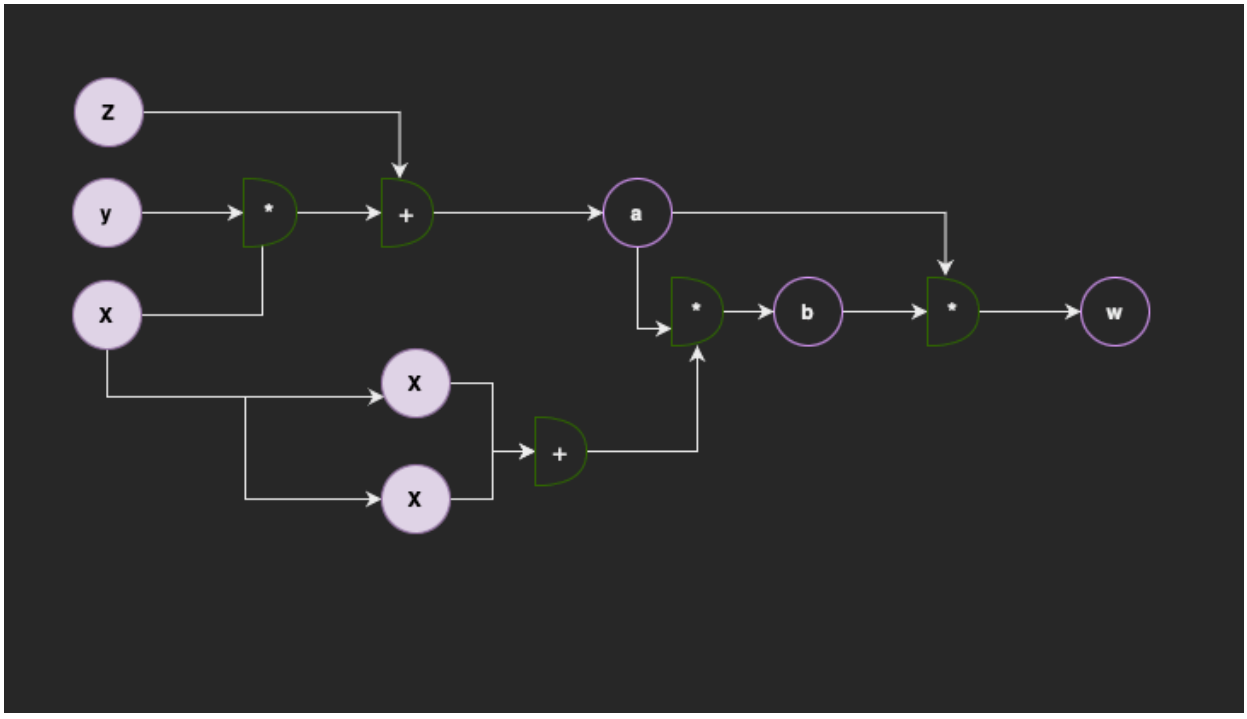
4.4.3 B

Draw the computational graph defined by this function, with inputs $x, y, z \in \mathcal{R}$ and output $w \in \mathcal{R}$

You make use symbols x,y,z,o, and operators $*$,+ in your solution. Be sure to use the correct shape for symbols and operators as shown in class.

$$a = x * y + z \tag{62}$$
$$b = (x + x) * a \tag{63}$$
$$w = a * b \tag{64}$$

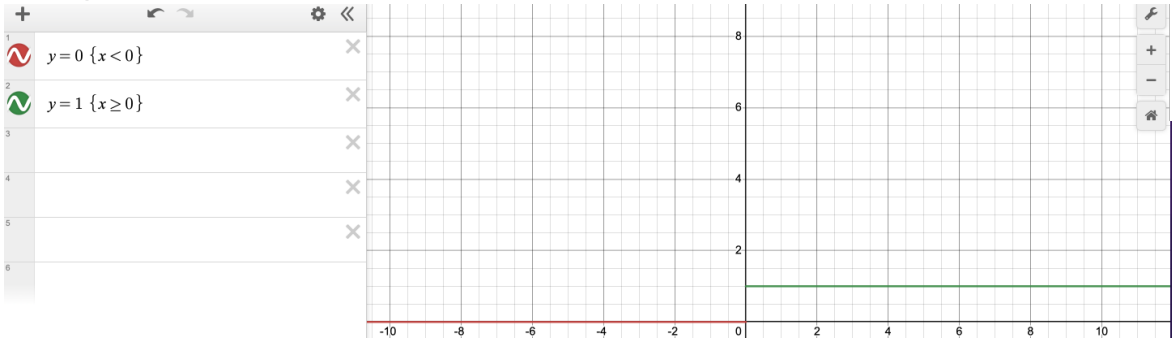
4.4.4 Answer



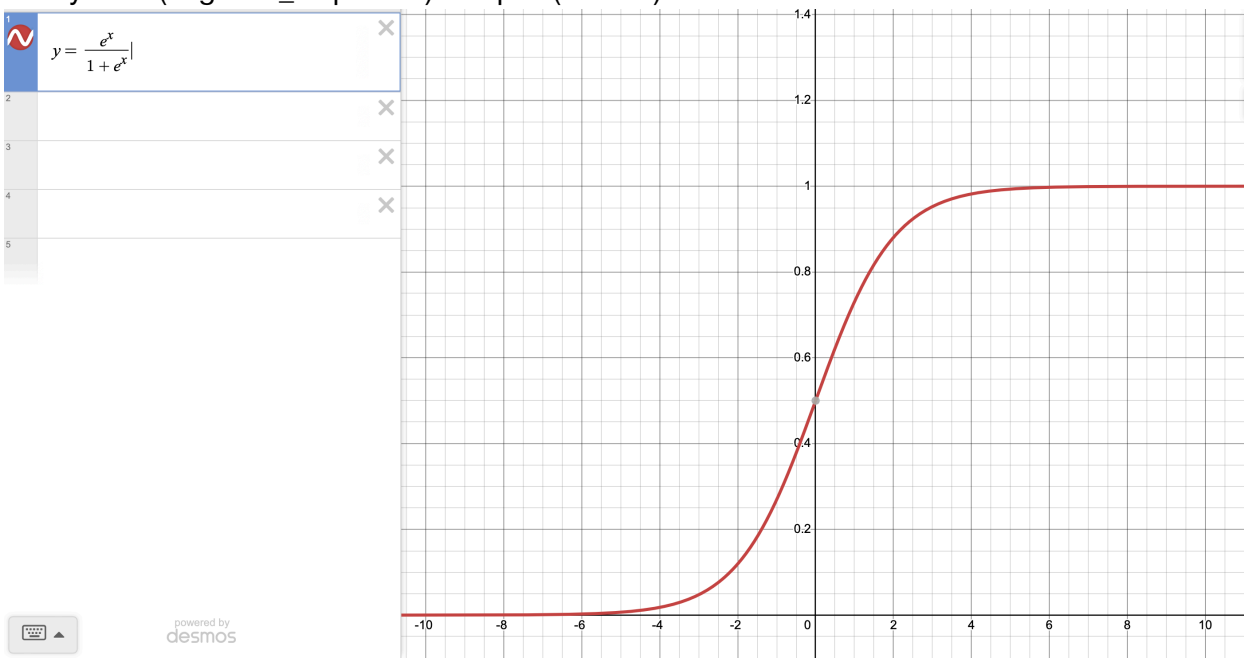
4.4.5 C

Draw the graph of the derivative for the following functions:

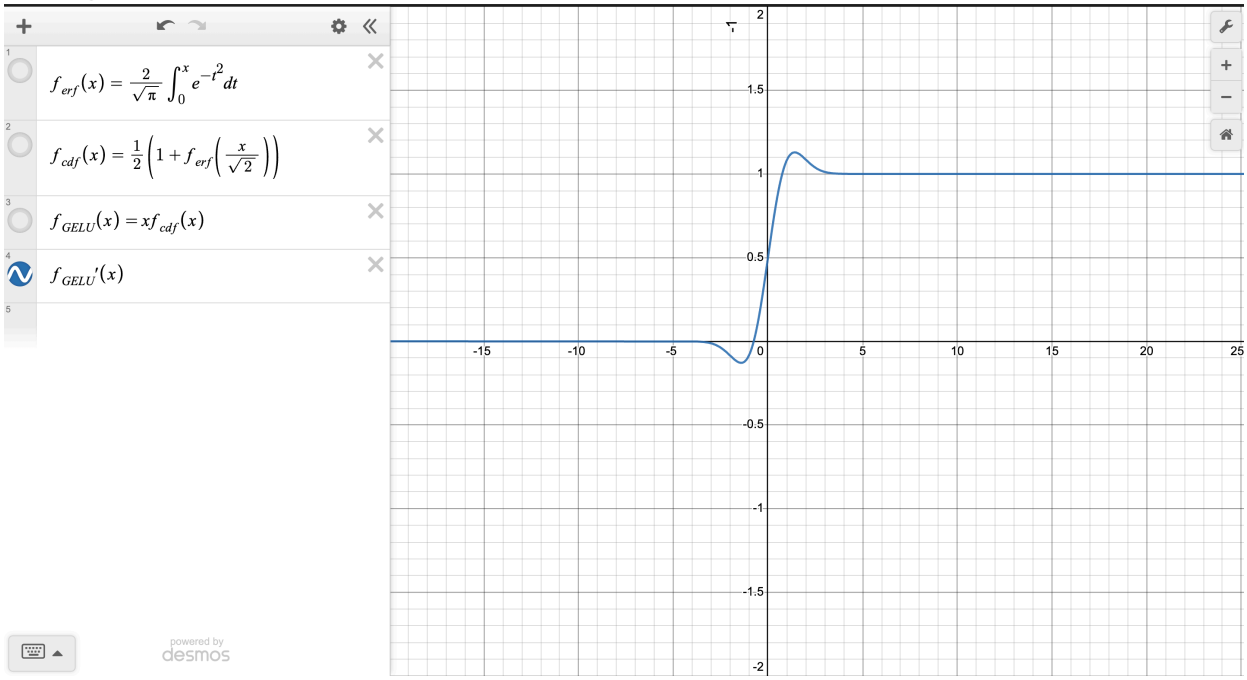
ReLU()



LeakyReLU(negative_slope=.01) Softplus(beta=1)



GELU()



4.4.6 Answer

4.4.7 d

What are 4 different types of linear transformations? What are the roles of linear transfomration and non linear transformation in a neural network.

4.4.8 Answer:

Linear Transformations can be:

1. Rotation (orthogonal matrix w/determinant one)
2. Reflection (matrix w/negative determinant)
3. Scaling (multiply by a diagonal matrix)
4. Shearing (multiply by a non-diagonal matrix)
5. Projection (mapping a dimmension to 0)

Linear transformations followed by non linear transformations serve to take our inputs and "rotate and squash" them. Linear transformations often map our input to a higher dimmensional reprsentation where this hidden state is then "squashed" by our nonlinear functions (ReLU, tan(h), etc).

4.5 1.4 e

Answer:

$$\underset{\theta}{\text{argmin}} L(F_\theta(D)Y) \tag{65}$$

Type *Markdown* and LaTeX: α^2