Homework 1: Backpropagation

CSCI-GA 2572 Deep Learning

Fall 2024

1 Theory (50pt)

1.1 Two-Layer Neural Nets

1.2 Regression Task

(a) **Answer:**
1) We are using custom activation function f (slightly modified version of `ReLU`), I will first create the custom activation function.

```
class CustomReLU(nn.Module):
    def __init__(self):
        super().__init__()

    def forward(self, x):
        m = nn.ReLU()
        return 5*m(x)
```

2) I would create the model described in the above task using `nn.Sequential`,

```
model = nn.Sequential(
    nn.Linear(_, _),
    CustomReLU(), # function f
    nn.Linear(_, _),
    nn.Identity() # function g
)
```

which is mathematically equivalent to,
$\boldsymbol{z}_1 = \boldsymbol{W}^{(1)}x + \boldsymbol{b}^{(1)}$
$\boldsymbol{z}_2 = f(\boldsymbol{z}_1)\ f = 5Relu(\cdot)$
$\boldsymbol{z}_3 = \boldsymbol{W}^{(2)}\boldsymbol{z}_2 + \boldsymbol{b}^{(2)}$
$\hat{\boldsymbol{y}} = g(\boldsymbol{z}_3)\ g = Identity(\cdot)$
Here I am constructing how one forward pass of the data point through model will go.

3) Now I will setup the loss function and the optimizer. Our loss function is $\ell_{\text{MSE}}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \|\hat{\boldsymbol{y}} - \boldsymbol{y}\|^2$ which calculates distnace between $\hat{y}$ (prediction) and y (target). Similarly we are using SGD as optimizer where we updates the model parameters using individual data points.

```
criterion = nn.MSELoss()
optimizer = torch.optim.SGD(model.parameters(), lr=_, weight_decay=_)
```

4) For each data point I will repeat 4th and 5th step. I will now program a forward pass and loss computation, I will first generate the prediction of our model ($\hat{y}$). Once obtained $\hat{y}$, I will compute cost between y and $\hat{y}$ (this is our loss which tells us badness of our model w.r.to our parameters which then used for back propagation).

```
y_hat = model(x)
loss = criterion(y_hat, y)
```

5) Finally, based on the loss I will compute partial derivatives of the loss w.r.to each parameter of every layer. Taking -ve of this derivative will give steepest descent towards minima. We will multiply these derivatives with appropriate learning rate and update the parameters. We will repeat this for every data point.

```
optimizer.zero_grad() # for avoiding accumulation
of the gradients
loss.backward() # calculating all the gradients by moving
backward through network and using chain rule.

optimizer.step() #updating parameters, backpropagation complete
```

(b) **Answer:**

$$\texttt{Linear}_1 \rightarrow f \rightarrow \texttt{Linear}_2 \rightarrow g$$

1) we are giving x as input to $\texttt{Linear}_1$, thus output of $\texttt{Linear}_1 = \boldsymbol{W}^{(1)}\boldsymbol{x}+\boldsymbol{b}^{(1)}$

2) Then output of $\texttt{Linear}_1$ is given as input to f where $f(\cdot) = 5(\cdot)^+ = 5\texttt{ReLU}(\cdot)$. Output of f will be $5\texttt{ReLU}(\boldsymbol{W}^{(1)}\boldsymbol{x}+\boldsymbol{b}^{(1)})$. Which will change all the -ve elements of the input vector to 0 and scale all the +ve elements to 5 times.

3) Output of f is passed as input to the $\texttt{Linear}_2$, thus output of $\texttt{Linear}_2$ will be $\boldsymbol{W}^{(2)}\boldsymbol{m}+\boldsymbol{b}^{(2)}$ where $\texttt{m} = 5\texttt{ReLU}(\boldsymbol{W}^{(1)}\boldsymbol{x}+\boldsymbol{b}^{(1)})$. Thus output of $\texttt{Linear}_2$ is $\boldsymbol{W}^{(2)}(\mathbf{5}\texttt{ReLU}(\boldsymbol{W}^{(1)}\boldsymbol{x}+\boldsymbol{b}^{(1)}))+\boldsymbol{b}^{(2)}$.

4) Output of $\texttt{Linear}_2$ is passed as input to g and g is identity function. Thus output of g will be same as input which is $\boldsymbol{W}^{(2)}(\mathbf{5}\texttt{ReLU}(\boldsymbol{W}^{(1)}\boldsymbol{x}+\boldsymbol{b}^{(1)}))+\boldsymbol{b}^{(2)}$. This is the final output of the model ($\hat{y}$).

5) Finally, $\hat{y}$ will be used alongside y (input) in loss calculation. Loss will be $\ell_{\text{MSE}}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \|\hat{\boldsymbol{y}} - \boldsymbol{y}\|^2$.

(c) **Answer:**

$\boldsymbol{z}_1 = \boldsymbol{W}^{(1)}x + \boldsymbol{b}^{(1)}$
$\boldsymbol{z}_2 = f(\boldsymbol{z}_1)$
$\boldsymbol{z}_3 = \boldsymbol{W}^{(2)}\boldsymbol{z}_2 + \boldsymbol{b}^{(2)}$
$\hat{\boldsymbol{y}} = g(\boldsymbol{z}_3)$

- After final layer g, output is $\hat{y}$ thus first gradeint will be $\frac{\partial \ell}{\partial \hat{\boldsymbol{y}}}$ (which is given). Next gradient is $\frac{\partial g}{\partial \hat{\boldsymbol{y}}} = \frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{z_3}}$ (which is given).
- $\boldsymbol{z}_3$ is an output of $\texttt{Linear}_2$. We have to find $\frac{\partial \ell}{\partial \boldsymbol{z}_3}$. Applying chain rule $\frac{\partial \ell}{\partial \boldsymbol{z}_3} = \frac{\partial \ell}{\partial \hat{\boldsymbol{y}}} \frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{z_3}}$.
- For the $\boldsymbol{W}^{(2)}$ and $\boldsymbol{b}^{(2)}$ of $\texttt{Linear}_2$: $\frac{\partial \ell}{\partial \boldsymbol{W}^{(2)}} = \frac{\partial \ell}{\partial \boldsymbol{z}_3} \frac{\partial \boldsymbol{z}_3}{\partial \boldsymbol{W}^{(2)}}$ and $\frac{\partial \ell}{\partial \boldsymbol{b}^{(2)}} = \frac{\partial \ell}{\partial \boldsymbol{z}_3} \frac{\partial \boldsymbol{z}_3}{\partial \boldsymbol{b}^{(2)}}$. Where $\frac{\partial \boldsymbol{z}_3}{\partial \boldsymbol{W}^{(2)}} = \boldsymbol{z}_2 = \mathbf{5}\texttt{ReLU}(\boldsymbol{W}^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)})$ and $\frac{\partial \boldsymbol{z}_3}{\partial \boldsymbol{b}^{(2)}} = 1$. Thus, $\frac{\partial \ell}{\partial \boldsymbol{W}^{(2)}} = \frac{\partial \ell}{\partial \hat{\boldsymbol{y}}} \frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{z}_3} 5\texttt{ReLU}(\boldsymbol{W}^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)})$ and $\frac{\partial \ell}{\partial \boldsymbol{b}^{(2)}} = \frac{\partial \ell}{\partial \hat{\boldsymbol{y}}} \frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{z}_3}$
- $\boldsymbol{z}_2$ is an output of f and input to $\texttt{Linear}_2$. We have to find $\frac{\partial \ell}{\partial \boldsymbol{z}_2}$. Applying chain rule $\frac{\partial \ell}{\partial \boldsymbol{z_2}} = \frac{\partial \ell}{\partial \boldsymbol{z_3}} \frac{\partial \boldsymbol{z_3}}{\partial \boldsymbol{z}_2}$. We can calculate $\frac{\partial \boldsymbol{z_3}}{\partial \boldsymbol{z}_2} = \boldsymbol{W}^{(2)}$. Thus, $\frac{\partial \ell}{\partial \boldsymbol{z}_2} = \frac{\partial \ell}{\partial \hat{\boldsymbol{y}}} \frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{z_3}} \boldsymbol{W}^{(2)}$
- $\boldsymbol{z}_1$ is an output of $\texttt{Linear}_1$ and input to f. We have to find $\frac{\partial \ell}{\partial \boldsymbol{z}_1}$. Applying chain rule $\frac{\partial \ell}{\partial \boldsymbol{z}_1} = \frac{\partial \ell}{\partial \boldsymbol{z}_2} \frac{\partial \boldsymbol{z}_2}{\partial \boldsymbol{z}_1}$. Partial derivative $\frac{\partial \boldsymbol{z}_2}{\partial \boldsymbol{z}_1}$ is given. Thus, $\frac{\partial \ell}{\partial \boldsymbol{z}_1} = \frac{\partial \ell}{\partial \hat{\boldsymbol{y}}} \frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{z_3}} \boldsymbol{W}^{(2)} \frac{\partial \boldsymbol{z}_2}{\partial \boldsymbol{z}_1}$.
- For the $\boldsymbol{W}^{(1)}$ and $\boldsymbol{b}^{(1)}$ of $\texttt{Linear}_1$: $\frac{\partial \ell}{\partial \boldsymbol{W}^{(1)}} = \frac{\partial \ell}{\partial \boldsymbol{z}_1} \frac{\partial \boldsymbol{z}_1}{\partial \boldsymbol{W}^{(1)}}$ and $\frac{\partial \ell}{\partial \boldsymbol{b}^{(1)}} = \frac{\partial \ell}{\partial \boldsymbol{z}_1} \frac{\partial \boldsymbol{z}_1}{\partial \boldsymbol{b}^{(1)}}$. Where $\frac{\partial \boldsymbol{z}_1}{\partial \boldsymbol{W}^{(1)}} = x$ and $\frac{\partial \boldsymbol{z}_1}{\partial \boldsymbol{b}^{(1)}} = 1$. Thus, $\frac{\partial \ell}{\partial \boldsymbol{W}^{(1)}} = \frac{\partial \ell}{\partial \hat{\boldsymbol{y}}} \frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{z_3}} \boldsymbol{W}^{(2)} \frac{\partial \boldsymbol{z}_2}{\partial \boldsymbol{z}_1} x$ and $\frac{\partial \ell}{\partial \boldsymbol{b}^{(2)}} = \frac{\partial \ell}{\partial \hat{\boldsymbol{y}}} \frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{z_3}} \boldsymbol{W}^{(2)} \frac{\partial \boldsymbol{z}_2}{\partial \boldsymbol{z}_1}$

(d) **Answer:**

Given $\boldsymbol{x} \in \mathbb{R}^N$ as input to $\texttt{Linear}_1$, we are considering $\boldsymbol{W}^{(1)} \in \mathbb{R}^{M \times N}$. Then $\boldsymbol{b}^{(1)} \in \mathbb{R}^M$ and $\boldsymbol{z_1} \in \mathbb{R}^M$. We know $\boldsymbol{z_2} = 5\texttt{ReLU}(\boldsymbol{z_1})$ which gives us $\boldsymbol{z_2} \in \mathbb{R}^M$. Given $\hat{\boldsymbol{y}} \in \mathbb{R}^K$ and g is an identity function which takes $\boldsymbol{z_3}$ as input then $\boldsymbol{z_3} \in \mathbb{R}^K$. For $\texttt{Linear}_2$ $\boldsymbol{z_2} \in \mathbb{R}^M$ is input and we are obtaining output $\boldsymbol{z_3} \in \mathbb{R}^K$ then $\boldsymbol{W^2}$ will be $\in \mathbb{R}^{K \times M}$ and $\boldsymbol{b^2}$ will $\in \mathbb{R}^K$.

- $\boldsymbol{z_2} = f(\boldsymbol{z_1})$ where $f = 5\texttt{ReLU}(\cdot)$ Thus,

$$\boldsymbol{z_2} = f(\boldsymbol{z_1}) = \begin{cases} 5\boldsymbol{z_1} & \text{for } \boldsymbol{z_1} > 0 \\ 0 & \text{for } \boldsymbol{z_1} \leq 0 \end{cases} \tag{1}$$

and

$$\frac{\partial \boldsymbol{z_2}}{\partial \boldsymbol{z_1}} = \begin{cases} 5 & \text{for } \boldsymbol{z_1} > 0 \\ 0 & \text{for } \boldsymbol{z_1} \leq 0 \end{cases} \tag{2}$$

both $\boldsymbol{z}_2$ and $\boldsymbol{z}_1 \in \mathbb{R}^M$ so f is $\mathbb{R}^M \to \mathbb{R}^M$. Thus, $\frac{\partial \boldsymbol{z_2}}{\partial \boldsymbol{z_1}}$ will have dimensionality of $\mathbb{R}^{M\times M}$. Say $\boldsymbol{z}_1 = [z_{1_1}, z_{1_2}, z_{1_3}, \cdots, z_{1_M}]$ and $\boldsymbol{z}_2 = [z_{2_1}, z_{2_2}, z_{2_3}, \cdots, z_{2_M}]$. $\frac{\partial \boldsymbol{z_2}}{\partial \boldsymbol{z_1}}$ will look like $\begin{bmatrix} \frac{\partial z_{2_1}}{\partial z_{1_1}} & \frac{\partial z_{2_2}}{\partial z_{1_1}} & \frac{\partial z_{2_3}}{\partial z_{1_1}} & \cdot & \cdot & \frac{\partial z_{2_M}}{\partial z_{1_1}} \\ \frac{\partial z_{2_1}}{\partial z_{1_2}} & \frac{\partial z_{2_2}}{\partial z_{1_2}} & \frac{\partial z_{2_3}}{\partial z_{1_2}} & \cdot & \cdot & \frac{\partial z_{2_M}}{\partial z_{1_2}} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \frac{\partial z_{2_1}}{\partial z_{1_M}} & \frac{\partial z_{2_2}}{\partial z_{1_M}} & \frac{\partial z_{2_3}}{\partial z_{1_M}} & \cdot & \cdot & \frac{\partial z_{2_M}}{\partial z_{1_M}} \end{bmatrix}$ Note that f is an element wise independent function, thus entries like $\frac{\partial z_{2_2}}{\partial z_{1_1}}$ would be 0. Which means in the Jacobian all the elements except diagonal entries would be zero. Thus our Jacobian would look like,
$\begin{bmatrix} \frac{\partial z_{2_1}}{\partial z_{1_1}} & 0 & 0 & \cdot & \cdot & 0 \\ 0 & \frac{\partial z_{2_2}}{\partial z_{1_2}} & 0 & \cdot & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \cdot & \cdot & \frac{\partial z_{2_M}}{\partial z_{1_M}} \end{bmatrix}$. Finally from eq. (2), our $\frac{\partial \boldsymbol{z_2}}{\partial \boldsymbol{z_1}}$ would look like,
$\begin{bmatrix} 5 & 0 & 0 & \cdot & \cdot & 0 \\ 0 & 5 & 0 & \cdot & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \cdot & \cdot & 0 \end{bmatrix}$ Where diagonal entries will have values either 5 or 0 and all other entries will be zero. Dimensionality would be $\mathbb{R}^{M\times M}$.

- $\hat{\boldsymbol{y}} = g(\boldsymbol{z_3})$ where g is an identity function. Thus $\hat{\boldsymbol{y}} = \boldsymbol{z_3}$ and $\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{z_3}} = 1$. g maps from $\mathbb{R}^K \to R^K$ which means $\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{z_3}}$ will have dimensionality of $\mathbb{R}^{K\times K}$. Similar to the above answer (answer for the $\frac{\partial \boldsymbol{z_2}}{\partial \boldsymbol{z_1}}$) g is an element wise independent function. Similarly it's Jacobian would look like,
$\begin{bmatrix} \frac{\partial \hat{y}_1}{\partial z_{3_1}} & 0 & 0 & \cdot & \cdot & 0 \\ 0 & \frac{\partial \hat{y}_2}{\partial z_{3_2}} & 0 & \cdot & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \cdot & \cdot & \frac{\partial \hat{y}_K}{\partial z_{3_K}} \end{bmatrix}$. Finally, $\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{z_3}}$ would have elements like,
$\begin{bmatrix} 1 & 0 & 0 & \cdot & \cdot & 0 \\ 0 & 1 & 0 & \cdot & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \cdot & \cdot & 1 \end{bmatrix}$ where all diagonal elements will be 1 and $\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{z_3}}$ will have dimensionality of $\mathbb{R}^{K\times K}$.

- $\ell_{\text{MSE}}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \|\hat{\boldsymbol{y}} - \boldsymbol{y}\|^2$ which gives us $\frac{\partial \ell}{\partial \hat{\boldsymbol{y}}} = 2(\hat{\boldsymbol{y}} - y)$ which is difference between prediction $(\hat{\boldsymbol{y}})$ and target $(\boldsymbol{y})$. $\ell_{\text{MSE}}(\hat{\boldsymbol{y}}, \boldsymbol{y})$ is $\mathbb{R}^K \to \mathbb{R}$ and thus $\frac{\partial \ell}{\partial \hat{\boldsymbol{y}}}$ will have dimensionality of $\mathbb{R}^K$. $\frac{\partial \ell}{\partial \hat{\boldsymbol{y}}}$ would be $[\frac{\partial \ell}{\partial \hat{y_1}}, \frac{\partial \ell}{\partial \hat{y_2}}, \cdots \frac{\partial \ell}{\partial \hat{y_K}}]$. $\hat{\boldsymbol{y}} \in \mathbb{R}^K$ and $\boldsymbol{y} \in \mathbb{I}^K$. Thus $\frac{\partial \ell}{\partial \hat{\boldsymbol{y}}}$ could have elements like
$\begin{bmatrix} 2(\hat{\boldsymbol{y}}_1 - \boldsymbol{y_1}) & 2(\hat{\boldsymbol{y}}_2 - \boldsymbol{y_2}) & 2(\hat{\boldsymbol{y}}_3 - \boldsymbol{y_3}) & \cdot & \cdot & \cdot & 2(\hat{\boldsymbol{y}}_K - \boldsymbol{y_K}) \end{bmatrix}$ which $\in \mathbb{R}^K$.

1.3 Classification Task

(a) **Answer:**
Changes in (b) :

- Output of $\texttt{Linear}_1$ ($\boldsymbol{z}_1$) would be same as before which will be passed as input to f. New output of f would be $tanh(\boldsymbol{W}^{(1)}\boldsymbol{x}+\boldsymbol{b}^{(1)})$ which is $\frac{2}{1+\exp(-2(\boldsymbol{W}^{(1)}\boldsymbol{x}+\boldsymbol{b}^{(1)}))}-1$
- Output of f is given as input to $\texttt{Linear}_2$. Thus output of $\texttt{Linear}_2$ will be $\boldsymbol{W}^{(2)}\boldsymbol{m}+\boldsymbol{b}^{(2)}$ where $m=\frac{2}{1+\exp(-2(\boldsymbol{W}^{(1)}\boldsymbol{x}+\boldsymbol{b}^{(1)}))}-1$. Thus, output of $\texttt{Linear}_2$ is $\boldsymbol{W}^{(2)}(\frac{\mathbf{2}}{\mathbf{1+exp(-2(}\boldsymbol{W}^{(1)}\boldsymbol{x}+\boldsymbol{b}^{(1)}\mathbf{))}}-\mathbf{1})+\boldsymbol{b}^{(2)}$.
- Output of $\texttt{Linear}_2$ is given as input to g, output of g is $(1+\exp(-k))^{-1}$ where $k=\boldsymbol{W}^{(2)}(\frac{\mathbf{2}}{\mathbf{1+exp(-2(}\boldsymbol{W}^{(1)}\boldsymbol{x}+\boldsymbol{b}^{(1)}\mathbf{))}}-\mathbf{1})+\boldsymbol{b}^{(2)}$. Thus final output ($\hat{\boldsymbol{y}}$) is $(1+\exp(-(\boldsymbol{W}^{(2)}(\frac{\mathbf{2}}{\mathbf{1+exp(-2(}\boldsymbol{W}^{(1)}\boldsymbol{x}+\boldsymbol{b}^{(1)}\mathbf{))}}-\mathbf{1})+\boldsymbol{b}^{(2)})))^{-1}$.

Changes in (c) :
As only f and g are changing (and outputs $\boldsymbol{z_2},\boldsymbol{z_3},\hat{\boldsymbol{y}}$) and we still have to report answers in terms of $\boldsymbol{x},\boldsymbol{y},\boldsymbol{W}^{(1)},\boldsymbol{b}^{(1)},\boldsymbol{W}^{(2)},\boldsymbol{b}^{(2)},\frac{\partial \ell}{\partial \hat{\boldsymbol{y}}},\frac{\partial \boldsymbol{z}_2}{\partial \boldsymbol{z}_1},\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{z_3}}$. Values of $\frac{\partial \ell}{\partial \hat{\boldsymbol{y}}},\frac{\partial \boldsymbol{z}_2}{\partial \boldsymbol{z}_1},\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{z_3}}$ would change as f,g are changing but we can to report our answers in terms of $\frac{\partial \ell}{\partial \hat{\boldsymbol{y}}},\frac{\partial \boldsymbol{z}_2}{\partial \boldsymbol{z}_1},\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{z_3}}$. Which means we only have to change partial derivatives where $\boldsymbol{z_2},\boldsymbol{z_3}$ or $\hat{\boldsymbol{y}}$ terms are present in the answers.

- We can see only $\frac{\partial \ell}{\partial \boldsymbol{W}^{(2)}}$ has a term of $\boldsymbol{z_2}$ in it. $\frac{\partial \ell}{\partial \boldsymbol{W}^{(2)}}=\frac{\partial \ell}{\partial \hat{\boldsymbol{y}}}\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{z}_3}\boldsymbol{z_2}$. Thus $\frac{\partial \ell}{\partial \boldsymbol{W}^{(2)}}=\frac{\partial \ell}{\partial \hat{\boldsymbol{y}}}\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{z}_3}(\frac{2}{1+\exp(-2(\boldsymbol{W}^{(1)}\boldsymbol{x}+\boldsymbol{b}^{(1)}))}-1)$
- All other partial derivatives will be unchanged. Refer 1.2 c for all other partial derivatives.

Changes in (d) :
$\boldsymbol{z}_1=\boldsymbol{W}^{(1)}x+\boldsymbol{b}^{(1)}$
$\boldsymbol{z}_2=f(\boldsymbol{z}_1)=tanh(\boldsymbol{z}_1)$
$\boldsymbol{z}_3=\boldsymbol{W}^{(2)}\boldsymbol{z}_2+\boldsymbol{b}^{(2)}$
$\hat{\boldsymbol{y}}=g(\boldsymbol{z}_3)=(1+exp(-\boldsymbol{z}_3))^{-1}$

- $\boldsymbol{z_2}=f(\boldsymbol{z_1})$ where $f=\texttt{tanh}(\cdot)$. And f is $\mathbb{R}^M\rightarrow\mathbb{R}^M$, thus $\frac{\partial \boldsymbol{z}_2}{\partial \boldsymbol{z}_1}$ will have dimensionality of $\mathbb{R}^{M\times M}$. Similar to 1.2 all activation functions (f,g) are element wise independent functions.
Thus $\frac{\partial \boldsymbol{z}_2}{\partial \boldsymbol{z}_1}$ will be $\begin{bmatrix} \frac{\partial \boldsymbol{z_2}_1}{\partial \boldsymbol{z_1}_1} & 0 & 0 & \cdot & \cdot & 0 \\ 0 & \frac{\partial \boldsymbol{z_2}_2}{\partial \boldsymbol{z_1}_2} & 0 & \cdot & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \cdot & \cdot & \frac{\partial \boldsymbol{z_2}_M}{\partial \boldsymbol{z_1}_M} \end{bmatrix}$. We know that $\frac{\partial tanh(x)}{\partial x}=$ $1-(tanh(x))^2$. And $\boldsymbol{z}_2=f(\boldsymbol{z}_1)=tanh(\boldsymbol{z}_1)$ which gives us $\frac{\partial \boldsymbol{z}_2}{\partial \boldsymbol{z}_1}$ to be

$$\begin{bmatrix} 1-tanh(\boldsymbol{z}_{1_1})^2 & 0 & 0 & \cdot & \cdot & 0 \\ 0 & 1-tanh(\boldsymbol{z}_{1_2})^2 & 0 & \cdot & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \cdot & \cdot & 1-tanh(\boldsymbol{z}_{1_M})^2 \end{bmatrix}$$ Which has dimensionality of $\mathbb{R}^{M\times M}$

- $\hat{y} = g(\boldsymbol{z_3}) = (1+exp(-\boldsymbol{z_3}))^{-1}$. Function g maps from $\mathbb{R}^K \to \mathbb{R}^K$ and $\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{z_3}}$ will be $\mathbb{R}^{K\times K}$. Again function g is an element wise independent function and $\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{z_3}}$ will be $\begin{bmatrix} \frac{\partial \hat{\boldsymbol{y}}_1}{\partial \boldsymbol{z}_{3_1}} & 0 & 0 & \cdot & \cdot & 0 \\ 0 & \frac{\partial \hat{\boldsymbol{y}}_2}{\partial \boldsymbol{z}_{3_2}} & 0 & \cdot & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \cdot & \cdot & \frac{\partial \hat{\boldsymbol{y}}_K}{\partial \boldsymbol{z}_{3_K}} \end{bmatrix}$.
  We know that $\frac{\partial sigmoid(x)}{\partial x} = sigmoid(x)(1-sigmoid(x))$. And $\hat{\boldsymbol{y}} = g(\boldsymbol{z}_3) = (1+exp(-\boldsymbol{z}_3))^{-1}$. Thus $\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{z_3}}$ will be
  $$\begin{bmatrix} \hat{\boldsymbol{y}}_1(1-\hat{\boldsymbol{y}}_1) & 0 & 0 & \cdot & \cdot & 0 \\ 0 & \hat{\boldsymbol{y}}_2(1-\hat{\boldsymbol{y}}_2) & 0 & \cdot & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \cdot & \cdot & \hat{\boldsymbol{y}}_K(1-\hat{\boldsymbol{y}}_K) \end{bmatrix}$$ with dimensionality of $\mathbb{R}^{K\times K}$
- As we are not changing the loss function, $\frac{\partial \ell}{\partial \hat{\boldsymbol{y}}}$ will be same as 1.2

(b) **Answer:**

Changes in b
From 1.3 a) we are only changing the loss function so inputs and outputs will be exactly same. Specifically $\boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{z}_3, \hat{\boldsymbol{y}}$ will be same as 1.3 a)

Changes in c
From 1.3 a) we are only changing the loss function and we have to report answers in $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{W}^{(1)}, \boldsymbol{b}^{(1)}, \boldsymbol{W}^{(2)}, \boldsymbol{b}^{(2)}, \frac{\partial \ell}{\partial \hat{\boldsymbol{y}}}, \frac{\partial \boldsymbol{z}_2}{\partial \boldsymbol{z}_1}, \frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{z_3}}$. There will be no change from 1.3 a).

Changes in d
$\frac{\partial \boldsymbol{z}_2}{\partial \boldsymbol{z}_1}$ and $\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{z_3}}$ will be same as 1.3 a).
As our new loss is $\ell_{\text{BCE}}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{1}{K}\sum_{i=1}^{K} -\left[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)\right]$ then $\frac{\partial \ell}{\partial \hat{\boldsymbol{y}}}$ will change from 1.3 a). $\ell_{\text{BCE}}(\hat{\boldsymbol{y}}, \boldsymbol{y})$ maps from $\mathbb{R}^K \to \mathbb{R}$ which means $\frac{\partial \ell}{\partial \hat{\boldsymbol{y}}}$ will be $\mathbb{R}^K$. We know $\frac{\partial \ell_{\text{BCE}}(\hat{\boldsymbol{y}}, \boldsymbol{y})}{\partial \hat{y}_i} = \frac{\hat{y}_i - y_i}{\hat{y}_i(1-\hat{y}_i)}$. Thus $\frac{\partial \ell}{\partial \hat{\boldsymbol{y}}}$ will have elements $\begin{bmatrix} \frac{\hat{y}_1-y_1}{\hat{y}_1(1-\hat{y}_1)} & \frac{\hat{y}_2-y_2}{\hat{y}_2(1-\hat{y}_2)} & \frac{\hat{y}_3-y_3}{\hat{y}_3(1-\hat{y}_3)} & \cdot & \cdot & \cdot & \frac{\hat{y}_K-y_K}{\hat{y}_K(1-\hat{y}_K)} \end{bmatrix}$ which $\in \mathbb{R}^K$.

(c) **Answer:**

- Unlike Sigmoid or Tanh, which squash the input into a limited range and tend to cause gradients to vanish, ReLU is linear for positive values. And when we try to train deeper networks gradient vanishing is a prominent issue. Using Relu helps us to avoid vanishing gradients.

- ReLU outputs zero for all negative inputs, which introduces sparsity in the network. Sparsity means that many neurons are not active, leading to a more efficient network that focuses only on the most relevant features. In deeper networks, this sparsity can help reduce computation and make the model easier to optimize by removing unnecessary activations.

1.4 Conceptual Questions

(a) **Answer:**
Why the Output of Softmax Cannot Be Exactly 0 or 1:

- The softmax function involves the exponential of the inputs, e^{z_i}, which is always strictly positive for any real number z_i (since $e^{z_i} > 0$ for all z_i). This means that the numerator e^{z_i} will always be positive.
- The denominator of the softmax function is the sum of all exponentials of the input elements:
$$\sum_{j=1}^{K} e^{z_j}$$
This sum is always greater than any individual e^{z_i}, meaning that each softmax output will be a fraction where the numerator e^{z_i} is strictly positive, but the denominator is larger than any individual e^{z_i}. Therefore, each softmax output will be a fraction between 0 and 1, but strictly neither 0 nor 1.
- The softmax function normalizes the exponential values of the input vector, ensuring that the sum of all outputs is 1. Since none of the exponentials can be 0, each element of the softmax output will always be greater than 0 but less than 1, representing valid probabilities.
- Note that mathematically softmax cannot be 1 but while programming due to approximations of small values can lead to softmax to be 1.

(b) **Answer:**

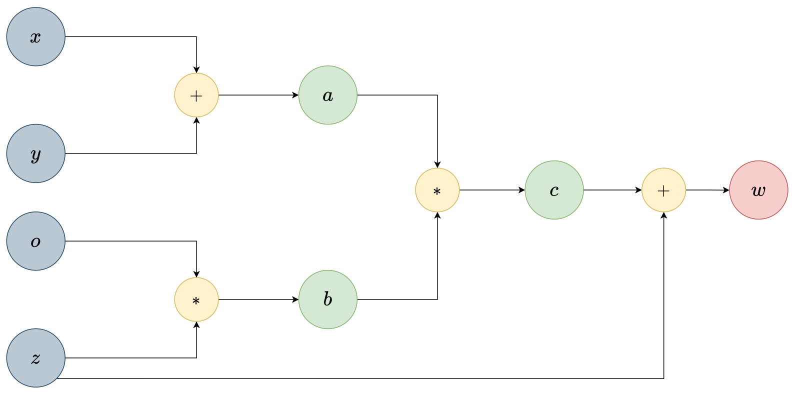
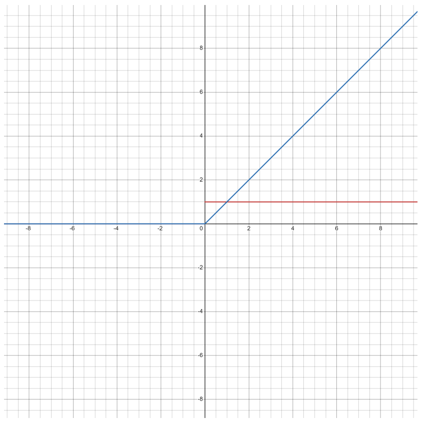
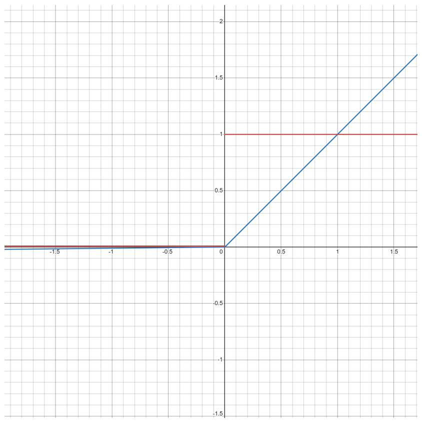


Figure 1: Computational graph of 1.4 a

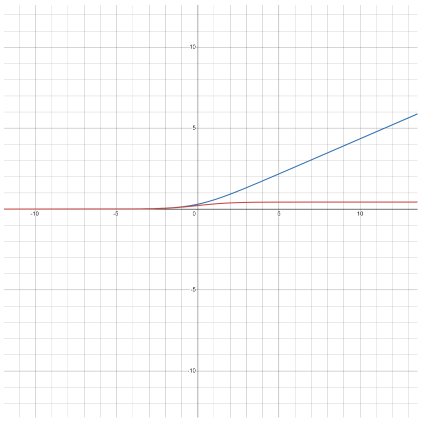
(c) **Answer:**



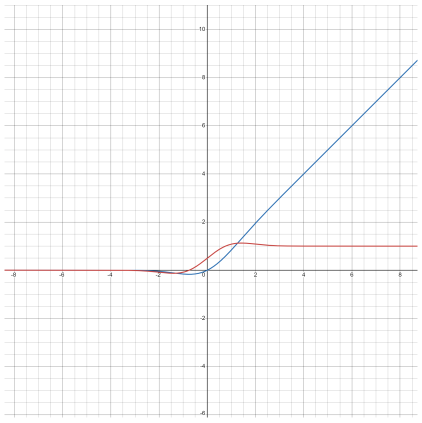
(a) Derivative of `ReLU()` is indicated by function with red colour and blue colour is `ReLU()` function



(b) Derivative of `LeakyReLU` is indicated by function with red colour and blue colour is `LeakyReLU` function



(c) Derivative of `Softplus` is indicated by function with red colour and blue colour is `Softplus` function



(d) Derivative of `GELU()` is indicated by function with red colour and blue colour is `GELU()` function

Figure 2: 1.4 c

(d) **Answer:**

`ReLU`, defined as:

$$\text{ReLU}(x) = \max(0, x)$$

It sets all negative inputs to zero and leaves positive values unchanged.

- During training, some neurons may stop updating their weights entirely. This happens when the input to the ReLU is negative, resulting in an output of zero, and thus no gradient flows through the neuron. Once a neuron gets stuck in this state, it "dies" because the gradient will always be zero, and it will no longer learn. This can lead to entire sections of the network being inactive, resulting in suboptimal training, especially in deep networks.
- The derivative of ReLU is discontinuous at zero. For $x > 0$, the derivative is 1, and for $x < 0$, the derivative is 0. This abrupt change can make optimization less smooth.
- ReLU does not have an upper bound for positive values. This can sometimes cause instability during training because very large activations could lead to exploding gradients.

How Leaky ReLU and Softplus Address Some Problems

- Leaky ReLU modifies the standard ReLU by allowing a small, non-zero gradient when the input is negative. It is defined as:

  $$\text{Leaky ReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{if } x \le 0 \end{cases}$$

  where α is a small positive constant (usually $\alpha = 0.01$).

  - By allowing a small slope (controlled by α) for negative inputs, Leaky ReLU ensures that the neurons won't "die" completely. They will still receive some gradient during backpropagation, which prevents them from becoming inactive.
- Softplus is a smooth approximation of ReLU, defined as:

  $$\text{Softplus}(x) = \log(1 + e^x)$$

  The Softplus function is differentiable everywhere and doesn't have the sharp transition that ReLU does at $x = 0$.

  - Softplus is smooth and continuously differentiable everywhere. This avoids the discontinuity at zero, which can help make optimization more stable and convergence smoother, especially in networks with delicate gradient flows.

  - Softplus doesn't have a flat zero-gradient region for negative inputs. Even for large negative values, the gradient is small but non-zero, allowing the network to continue updating the parameters.

(e) **Answer:**

- **Shear**:
  Shear transformation skews the shape, shifting one axis in a direction proportional to the other. For example square becomes parallelogram.
- **Scale**:
  Scale transformation scales all or some dimensions by a constant factor. For uniform scaling, the factor is the same for all dimensions, while for non-uniform scaling, different factors can be applied to each dimension.
- **Rotation**:
  Rotation transformation rotates a vector around an origin or a certain point, typically by a given angle in n-dimensional space
- **Reflection**:
  Rotation transformation flips vectors over a certain axis or plane, essentially creating a mirror image.
- Role of linear and non-linear transformation:
  - Linear transformations map the input data into a new space. They basically reweigh the features and highlights important features (certain weights are higher than others), but they alone are limited in their representational power. A neural network with only linear layers is effectively just a linear model, regardless of how many layers it has, because the composition of linear functions is still linear.
  - Due to caveat explained above, we use non-linear function. These functions add non linearity and helps us to create deep models. Non-linear transformations enable neural networks to learn non-linear relationships in data, which is crucial for modeling complex patterns.
  - Finally, Non-linear transformations, when combined with linear transformations, enable neural networks to approximate any continuous function.

2 Implementation (50pt)

2.1 Backpropagation (35pt)

2.2 Gradient Descent (15pt + 5pt)

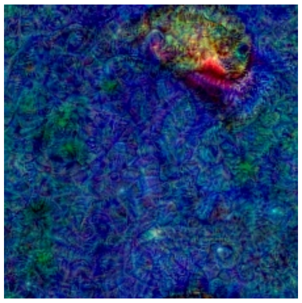


Figure 3: Generated Image for label 0

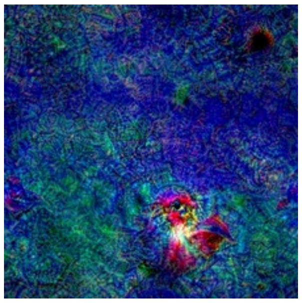


Figure 4: Generated Image for label 12

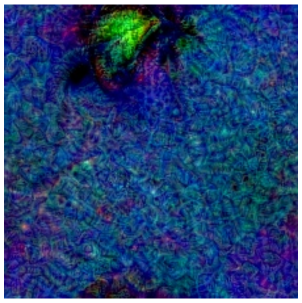


Figure 5: Generated Image for label 954

Here for extra credit I have used, techniques like blurring image after each iteration, clamping, image scaling and gradient blurring. Outputs can be seen from figs. 3 to 5