1.2.

a)

1. Generate a prediction

$\sim y$ = model(x)

2. Compute the loss

$L(w, x, y) = f(x, y) = C(\sim y, y) = ||y - \sim y||^2$

3. Set the gradient parameters to zero

optimizer.zero_grad()

4. Compute and accumulate the gradient parameters

L.backward()

5. Step in towards the negative of the gradient parameters

optimizer.step()

b)

$x \to \text{Linear}_1 \to s_1 \to 3\text{ReLU}(.) \to a_1 \to \text{Linear}_2 \to s_2 \to g \to \sim y$

where,

$s_1 = W_1^T x + b_1$,

$a_1 = (3\text{ReLU})(W_1^T x + b_1)$

$s_2 = (W_2^T)\{3\text{ReLU}(W_1^T x + b_1)\} + b_2$, and

$\sim y = (W_2^T)\{3\text{ReLU}(W_1^T x + b_1)\} + b_2$

c)

$\partial C/ \partial \mathbf{W_1} = (x)(\partial C/ \partial \mathbf{\sim y})(\partial \sim y/ \partial \mathbf{s_2})(W_2^T)(\partial \mathbf{a_1}/ \partial \mathbf{s_1})$

$\partial C/ \partial \mathbf{b_1} = (\partial C/ \partial \mathbf{\sim y})(\partial \mathbf{\sim y}/ \partial \mathbf{s_2})(W_2^T)(\partial \mathbf{a_1}/ \partial \mathbf{s_1})$

$\partial C/ \partial \mathbf{W_2} = (3\text{ReLU}(W_1^T x + b_1))(\partial C/ \partial \mathbf{\sim y})(\partial \mathbf{\sim y}/ \partial \mathbf{s_2})$

$\partial C/ \partial \mathbf{b_2} = (\partial C/ \partial \mathbf{\sim y})(\partial \sim y/ \partial \mathbf{s_2})$

Moreover,

$\partial \mathbf{s_2}/ \partial \mathbf{W_2} = a_1$ (Used above)

∂ $\mathbf{s_1}$/ ∂ $\mathbf{W_1}$ = x

∂ $\mathbf{s_1}$/ ∂ $\mathbf{b_1}$ = 1

∂ $\mathbf{s_2}$/ ∂ $\mathbf{b_2}$ = 1

And (∂ **~y**/ ∂ $\mathbf{s_2}$) = 1

d)

L1 is the output size of the first linear layer

∂ $\mathbf{a_1}$/ ∂ $\mathbf{s_1}$ = AS_{ij} where AS_{ij} = 3 when i = j and and S_{ij}>0

0 otherwise

∂ **~y**/∂ $\mathbf{s_2}$ = I_{KxK} where I is the identity matrix

∂ C/ ∂ **~y** = $2[y^{\sim} - y]_{1xK}$

1.3.

a.

x -> $Linear_1$ -> s_1 -> tanh(.) -> a_1 -> $Linear_2$ -> s_2 -> 1/(1 + exp(-x)) -> ~y

where,

$s_1 = W_1^T x + b_1$,

$a_1 = \tanh(W_1^T x + b_1)$ (Changed)

$s_2 = W_2^T\{\tanh(W_1^T x + b_1)\} + b_2$, and (Changed)

$\sim y = 1/(1+ \exp(-W_2^T\{\tanh(W_1^T x + b_1)\} - b_2))$ (Changed)

ii.

The value of (∂ **~y**/ ∂ $\mathbf{s_2}$) has changed to `sigmoid(s₂) * (1 - sigmoid(s₂))`

And a_1 is $\tanh(W_1^T x + b_1)$

$\partial C/ \partial \mathbf{W_1} = (x) (\partial C/ \partial \mathbf{\sim y}) (\partial \sim y/ \partial \mathbf{s_2}) (W_2^T) (\partial \mathbf{a_1}/ \partial \mathbf{s_1})$

$\partial C/ \partial \mathbf{b_1} = (\partial C/ \partial \mathbf{\sim y}) (\partial \mathbf{\sim y}/ \partial \mathbf{s_2}) (W_2^T) (\partial \mathbf{a_1}/ \partial \mathbf{s_1})$

$\partial C/ \partial \mathbf{W_2} = (\tanh(W_1^T x + b_1)) (\partial C/ \partial \mathbf{\sim y}) (\partial \mathbf{\sim y}/ \partial \mathbf{s_2})$

$\partial C/ \partial \mathbf{b_2} = (\partial C/ \partial \mathbf{\sim y}) (\partial \sim y/ \partial \mathbf{s_2})$

Moreover,

$\partial \mathbf{s_2}/ \partial \mathbf{W_2} = a_1$ (Used above)

$\partial \mathbf{s_1}/ \partial \mathbf{W_1} = x$

$\partial \mathbf{s_1}/ \partial \mathbf{b_1} = 1$

$\partial \mathbf{s_2}/ \partial \mathbf{b_2} = 1$

The value of $(\partial \mathbf{\sim y}/ \partial \mathbf{s_2})$ has changed to `sigmoid(s₂) * (1 - sigmoid(s₂))`

iii.

L1 is the output size of the first linear layer

$\partial \mathbf{a_1}/ \partial \mathbf{s_1} = 1- \tanh^2 (s_{ij})$ where i and j go to L1xL1

$\partial \mathbf{\sim y}/\partial \mathbf{s_2} = \sim y*(1-\sim y)$ of dimension K x K

$\partial C/ \partial \mathbf{\sim y} = 2[y\sim - y]_{1xK}$

b)

The value of $\partial C/ \partial \mathbf{\sim y}$ is `-1*y/~y + (1-y)/(1 - ~y)` where

`~y = 1/(1 + exp (-W₂*tanh(W₁ᵀx + b₁) – b₂))`

Hence, the expressions for s_1, a_1, s_2 and ~y do not change over the previous part but the substitution of $\partial C/\partial \mathbf{\sim y}$ would change in the values

$\partial C/ \partial \mathbf{W_1} = (x) (\partial C/ \partial \mathbf{\sim y}) (\partial \sim y/ \partial \mathbf{s_2}) (W_2^T) (\partial \mathbf{a_1}/ \partial \mathbf{s_1})$

$\partial C/ \partial \mathbf{b_1} = (\partial C/ \partial \mathbf{\sim y}) (\partial \mathbf{\sim y}/ \partial \mathbf{s_2}) (W_2^T) (\partial \mathbf{a_1}/ \partial \mathbf{s_1})$

$\partial C/ \partial \mathbf{W_2} = (3\mathrm{ReLU}(W_1^T x + b_1)) (\partial C/ \partial \mathbf{\sim y}) (\partial \mathbf{\sim y}/ \partial \mathbf{s_2})$

$\partial C/ \partial \mathbf{b_2} = (\partial C/ \partial \mathbf{\sim y}) (\partial \sim y/ \partial \mathbf{s_2})$

c)

$\partial C / \partial \tilde{\mathbf{y}} = [(1-y)/(1-\tilde{y}) - y/\tilde{y}]_{1xK}$

Rest everything is same as the part b

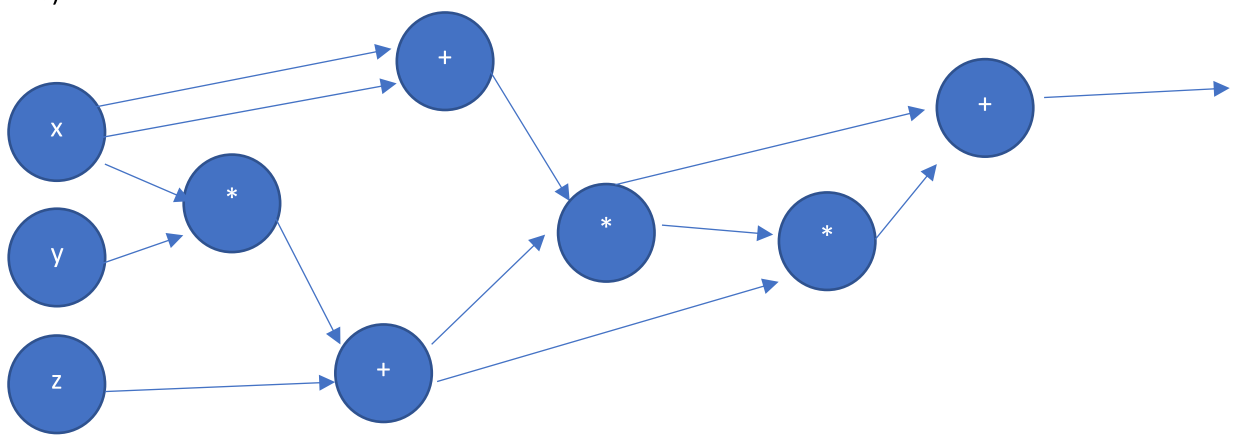
1.4.

Because softmax implemented in most machine learning libraries actually gives the one-hot encoded vector with 1 at the index with the maximum argument, while softmax mathematically gives

$\text{Softmax}_b(e) = (1/b)*\log \sum \exp(b*e)$ (= max(e) as b tends to infinity) and

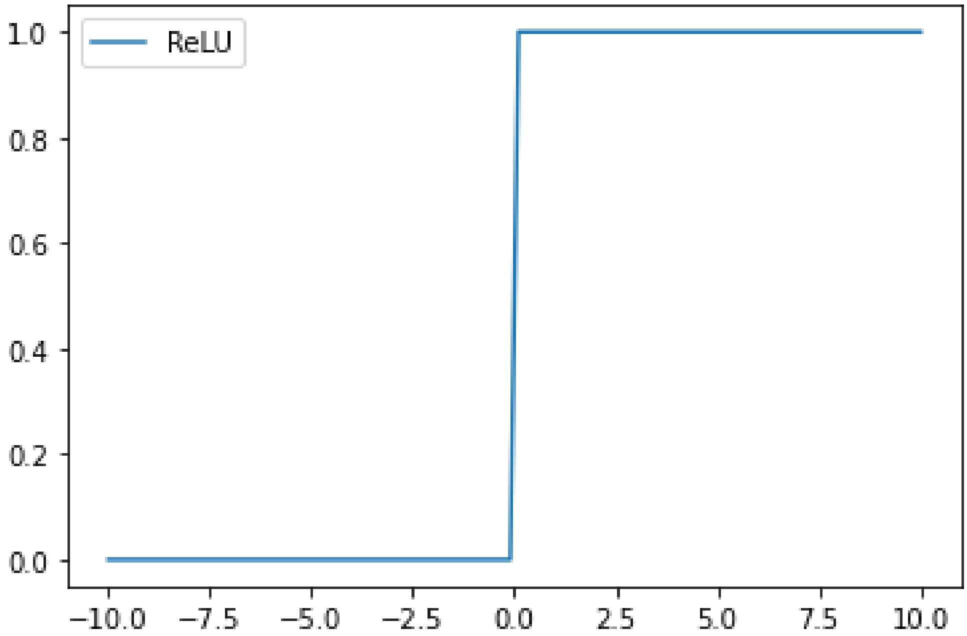
$\text{Softargmax}_b(e) = \exp(be)/\sum \exp(b*e)$ actually mathematically corresponds to the maximum argument (as b tends to infinity)

b)

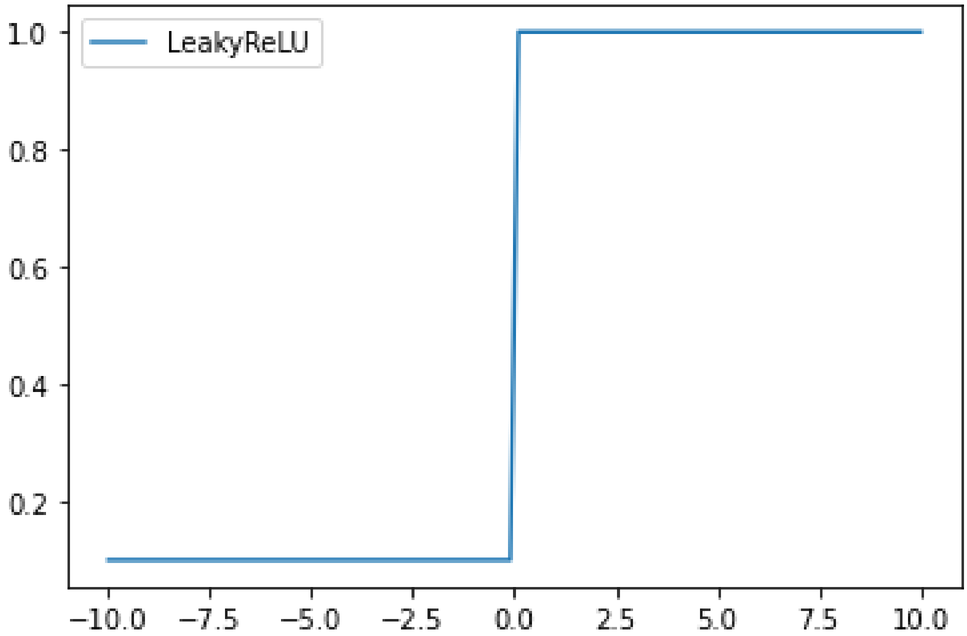


1.4.

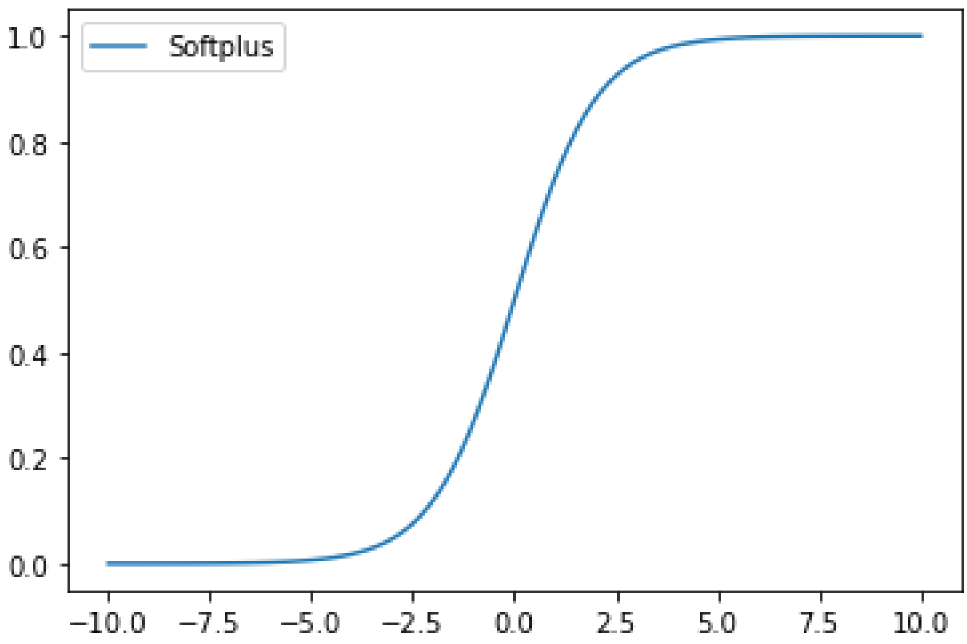
c)

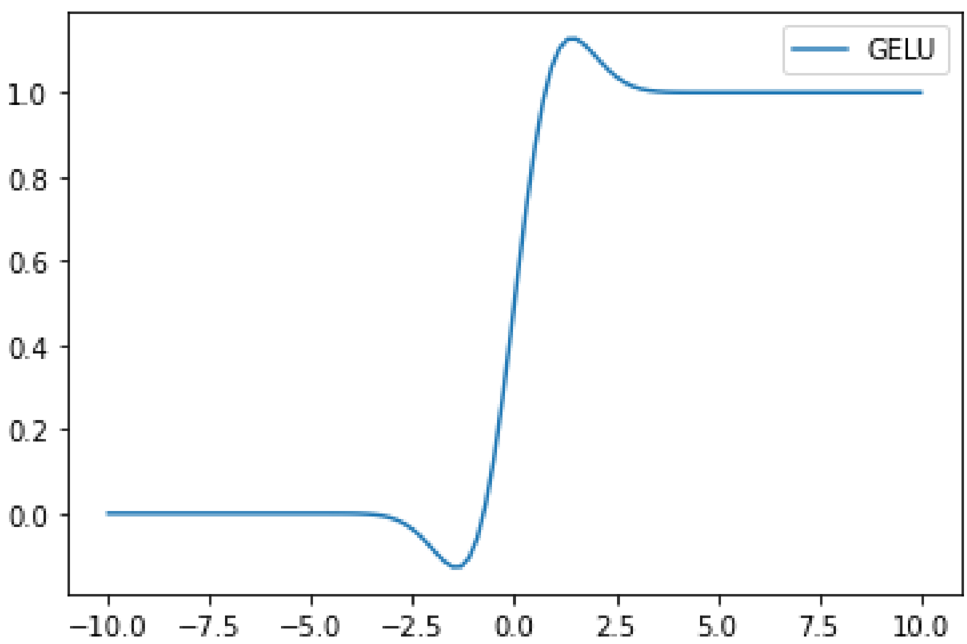


ReLU
1.0
0.8
0.6
0.4
0.2
0.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0



LeakyReLU
1.0
0.8
0.6
0.4
0.2
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0





d)

Jacobian of f = ∂ f/ ∂ x = W1

Jacobian of g = ∂ g/ ∂ x = W2

ii.

Jacobian of h = ∂ h/ ∂ x = W1 + W2

If W1 = W2

h = ∂ h/ ∂ x = 2 (W2)

e)

Jacobian of f = ∂ f/ ∂ x = W1

Jacobian of g = ∂ g/ ∂ x = W2

Jacobian of h = ∂ h/ ∂ x = (W2) (W1)

If W1 = W2

h = ∂ h/ ∂ x = $(W_1)^2 = (W_2)^2$