Problem set 3 202201839 _ Nael Haidar

Prob 1

1) False, back propagation is used to calculate the gradients in a neural network for updating weights.

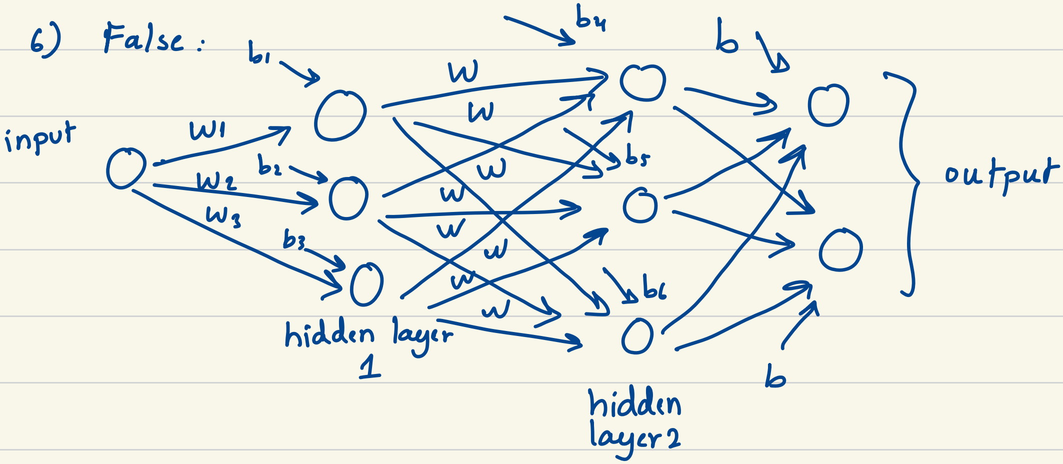
2) True, a neuron with a step activation function can implement logical operations.

3) True, Softmax makes sure that outputs sum to 1 and are between 0 and 1.

4) False, a deep network is a neural network with many hidden layers which create the depth.

5) True, autoencoders reconstruct their inputs using input data as the target.

6) False:

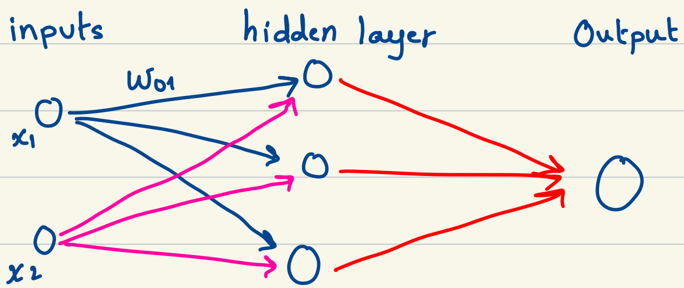


Total parameters = 26.

7) True, it's also more efficient in computation
8) True, l_1 and l_2 regularization prevent large weights.
9) True, with more neurons, the parameters increase so the estimation error increases.
10) True, it makes differentiation and back propagation automatic.

Problem 2

$$\sigma'(x) = \sigma(x)(1 - \sigma(x)).$$



$$L(y,\hat{y}) = \frac{1}{m}\sum_{i=1}^{m}\left(\hat{y}^{(i)} - y^{(i)}\right)^2$$

1) $a_1 = \sigma\left(W_{01}^{(1)} + W_{11}^{(1)}x_1^{(i)} + W_{12}x_2^{(i)}\right)$

$\hat{y}^{(i)} = \sigma\left(W_0^{(2)} + \sum_{j=1}^{3} W_j^{(2)} a_j\right)$

→ Derivative of Loss with respect to $\hat{y}$:

$$\frac{\partial \mathcal{L}}{\partial \hat{y}^{(i)}} = 2\left(\hat{y}^{(i)} - y^{(i)}\right)$$

→ Backpropagation for W_{12}:

$$\frac{\partial \mathcal{L}}{\partial w_{12}^{(1)}} = \frac{\partial \mathcal{L}}{\partial \hat{y}^{(i)}} \cdot \frac{\partial \hat{y}^{(i)}}{\partial a_1} \cdot \frac{\partial a_1}{\partial w_{12}^{(1)}}$$

$$= 2\left(\hat{y}^{(i)} - y^{(i)}\right) \cdot w_1^{(2)} \cdot \hat{y}^{(i)}\left(1 - \hat{y}^{(i)}\right) \cdot x_2^{(i)} \cdot a_1(1 - a_1)$$

→ Gradient descent update rule:

$$w_{12}^{(1)} = w_{12}^{(1)}(\text{old}) - \eta \cdot \left(\quad \right)$$

2) Step function $f(x) = \begin{cases} 1 & ; \quad x \geq 0 \\ 0 & ; \quad x < 0 \end{cases}$

- The output neuron $\hat{y}$ will output 1 if any of the hidden layer neurons output 1.

→ For perfect classification

for a_1: Bias $w_{01}^{(1)} = 0.5$; $w_{11}^{(1)} = 1$; $w_{21}^{(1)} = -1$

a_2: $w_{02}^{(1)} = 0.5$; $w_{12}^{(1)} = 1$; $w_{22}^{(1)} = 1$

a_3: $w_{03}^{(1)} = -1.5$; $w_{13}^{(1)} = 1$; $w_{23}^{(1)} = 1$

$\hat{y}$: $w_0^{(2)} = -0.5$; $w_1^{(2)} = 1$; $w_2^{(2)} = 1$; $w_3^{(2)} = 1$.

3) No, there isn't a set of weights that will make the loss zero since finding an exact set of weights requires an optimization process like gradient descent, and the linear activation function restrict the network's ability to find a perfect seperation.

Problem 3| 1) Yes, it can if the inputs are concatenated time steps. The input dimensions have to be $k \times d$. (k time steps with each of dimension d).

Additionally, there should be multiple hidden layers with non-linear activations.

2) $$\frac{dx}{dt} \approx (x_i - x_{i-1}) / (t_i - t_{i-1})$$

This means $x_i = x_{i-1} + g(x_{i-1}) \cdot (t_i - t_{i-1})$

- Comparison:

DE model	NN model
mathimatical representation	Data driven approach
Requires known equations	Can learn complex non linear relationships.
Analytical and interpretable	doesn't require prior knowledge.

3) - Input: 5 previous time steps.
- Convolution operation:
  - Apply 1D convolutional filter across time series
  - Filter slides across input, extracting local features.
  - Convolution detects short term trends, recurring patterns, and local correlations.

→ Small filters work for immediate correlations and micro-trends.
Large filters work for macro-trends and long-range dependencies.

→ After the convolution:

- Pooling layer: reduces feature map dimensionality
- Flatten layer: convert feature map to 1D vector.
- Fully connected layers: learn complex relationships.
- Output layer: predict next time step.

Prob 4) on github.