Final 2024 - 2025

Ex 1:

a) Perceptron update rule:

Let $\hat{y} = \begin{cases} 1 & \text{if } w_1x_1 + w_2x_2 + w_3x_3 + b > 0 \\ 0 & \text{if } w_1x_1 + w_2x_2 + w_3x_3 + b \leq 0 \end{cases}$

if $y \neq \hat{y}$, update the weights and biases:

- $w_i = w_i + \eta y x_i$
- $b = b + \eta y$

b) $w = (0,0,0)$, $b = 0$, $\eta = 1$

- Epoch 1:

1) $(0,0,0)$, $y = 0$:

$\hat{y} = (0.0 + 0.0 + 0.0 + 0) = 0$ No update

2) $(0,1,1)$, $y = 0$:

$\hat{y} = (0.0 + 0.1 + 0.1 + 0) = 0$ No update

3) $(1,0,1)$, $y = 0$:

$\hat{y} = (0.1 + 0.0 + 0.1 + 0) = 0$ No update

4) $(1,1,1)$, $y = 1$:

$\hat{y} = (0.1 + 0.1 + 0.1 + 0) = 0$ misclassified, update:

$w = (0,0,0) + (1)(1)(1,1,1) = (1,1,1)$

$b = 0 + (1)(1) = 1$

- Epoch 2:

1) $(0,0,0)$, $y = 0$:

$\hat{y} = (1.0 + 1.0 + 1.0 + 1) = 1$ misclassified, update:

$w = (1,1,1) + (1)(0)(0,0,0) = (1,1,1)$

$b = 1 + (1)(0) = 1$

2) $(0,1,1)$, $y = 0$:

$\hat{y} = (1.0 + 1.1 + 1.1 + 1) = 3$ misclassified, update:

$w = (1,1,1) + (1)(0)(0,1,1) = (1,1,1)$

$b = 1 + (1)(0) = 1$

3) $(1,0,1)$, $y = 0$:

$\hat{y} = (1.1 + 1.0 + 1.1 + 1) = 3$ misclassified, update

$w = (1,1,1) + (1)(0)(1,0,1) = (1,1,1)$

$b = 1 + (1)(0) = 1$

4) $(1,1,1)$, $y = 1$:

$\hat{y} = (1.1 + 1.1 + 1.1 + 1) = 4$ correct classification.

c) Increasing dimensionality makes it easier for the perceptron model to find a linear separator.

A dataset that is not linearly separable in n dimensions might become separable when mapped to a higher-dimensional space $m > n$.

Ex 2:

$$L(w,b) = \frac{1}{2}\sum_{i=1}^{2} (\hat{y}_i - y_i)^2 + \lambda \|w\|_2^2$$

where $\hat{y}_i = w^T x_i + b$

- $x_1 = (2,1)$, $y_1 = 4$
- $x_2 = (1,3)$, $y_2 = 7$
- $w = (0,0)$, $b = 0$
- $\eta = 0.1$, $\lambda = 1$

a) . Gradient w.r.t. w:

$$\frac{dL(w,b)}{dw} = \frac{d}{dw}\frac{1}{2}\sum (\hat{y}_i - y_i)^2 + \frac{d}{dw}\lambda \|w\|_2^2$$

$$= \frac{1}{2}\sum \frac{d}{dw}(\hat{y}_i - y_i)^2 + \lambda \frac{d}{dw}\|w\|_2^2$$

$$= \frac{1}{2}\sum 2 x_i (\hat{y}_i - y_i) + 2\lambda w$$

$$= \sum x_i (\hat{y}_i - y_i) + 2\lambda w$$

. Gradient w.r.t. b:

$$\frac{dL(w,b)}{db} = \frac{d}{db}\frac{1}{2}\sum (\hat{y}_i - y_i)^2 + \frac{d}{db}\lambda \|w\|_2^2$$

$$= \sum (\hat{y}_i - y_i) + 0 \quad \text{with} \quad \frac{d\hat{y}}{db} = 1$$

$$= \sum (\hat{y}_i - y_i)$$

b) One batch gradient descent update:

- Predictions: $\hat{y}_i = w^T x_i + b$

. $\hat{y}_1 = [0, 0] \begin{bmatrix} 2 \\ 1 \end{bmatrix} + 0 = 0$

. $\hat{y}_2 = [0, 0] \begin{bmatrix} 1 \\ 3 \end{bmatrix} + 0 = 0$

- Errors: $e_i = \hat{y}_i - y_i$

. $e_1 = \hat{y}_1 - y_1 = 0 - 4 = -4$

. $e_2 = \hat{y}_2 - y_2 = 0 - 7 = -7$

- Gradients:

. $\frac{dl}{dw} = \sum x_i (\hat{y}_i - y_i) + 2\lambda w$

$= [(2, 1)(-4)] + [(1, 3)(-7)] + [2(1)(0, 0)]$

$= (-8, -4) + (-7, -21) + (0, 0)$

$= (-15, -25)$

. $\frac{dL}{db} = \sum (\hat{y}_i - y_i)$

$= e_1 + e_2 = -4 - 7 = -11$

- Gradient Descent update:

. $w = w - \eta \frac{dl}{dw} = (0, 0) - (0.1)(-15, -25) = (1.5, 2.5)$

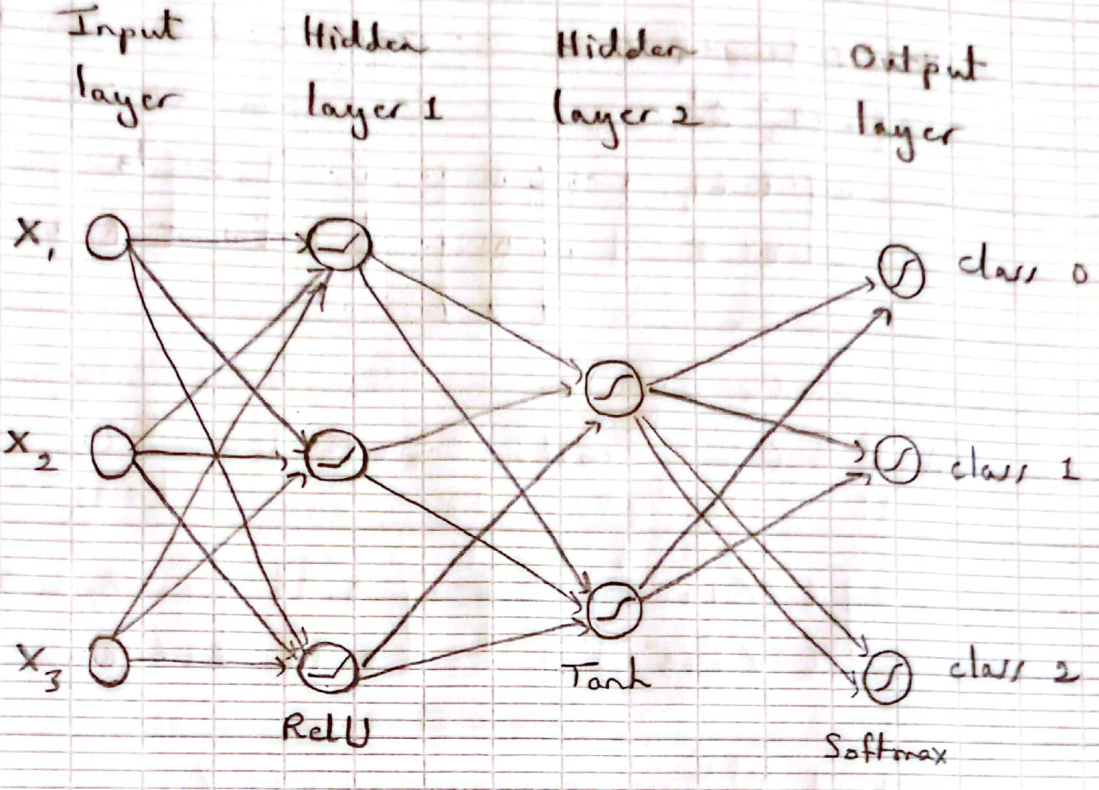
. $b = b - \eta \frac{dl}{db} = 0 - (0.1)(-11) = 1.1$

c) Tuning the regularization parameter λ is essential to control the trade-off btw the model's ability to minimize the generalization error and its structural complexity.

- When λ is low, the model prioritizes minimizing the sum of squared errors, allowing weights to grow freely to capture high-frequency patterns, and thus leading to overfitting.
- When λ is high, it penalizes large weights heavily. This simplifies the model by shrinking coefficients towards zero, and thus leads to underfitting.

Ex 3:

a)



Title: 3-layer neural network architecture

b) . Hidden layer 1:

$$Z_1 = W_1^T X + b_1$$

$$= \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix}$$

$$A_1 = G_1(Z_1) \quad \text{with} \quad G_1 = \text{ReLU} = \max(0, Z_i)$$

$$= \begin{bmatrix} \max(0,2) \\ \max(0,0) \\ \max(0,0) \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix} \begin{matrix} \text{output of neuron 0} \\ \text{neuron 1} \\ \text{neuron 2} \end{matrix}$$

- Hidden layer 2:

$$Z_2 = W_2^T A^1 + b_2$$

$$= \begin{bmatrix} 1 & -1 & 1 \\ 0 & 2 & -1 \end{bmatrix} \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

$$A_2 = G_2(Z_2) \quad \text{with } G_2 = \tanh = \frac{2}{1+e^{-2z_i}} - 1$$

$$= \begin{bmatrix} (2/1+e^{-4})-1 \\ (2/1+e^{-2})-1 \end{bmatrix} = \begin{bmatrix} 0.96 \\ 0.76 \end{bmatrix} \begin{matrix} \text{neuron 0 output} \\ \text{neuron 1 output} \end{matrix}$$

- Output layer:

$$Z_3 = W_3^T A^2 + b_3$$

$$= \begin{bmatrix} 1 & 0 \\ -1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0.96 \\ 0.76 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.96 \\ -0.2 \\ 0.76 \end{bmatrix}$$

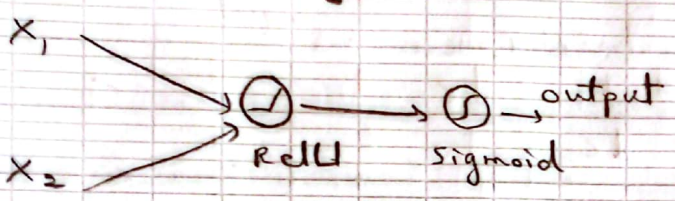
$$A_3 = G_3(Z_3) \quad \text{with } G_3 = \text{softmax} = \frac{e^{z_i}}{\sum e^{z_i}}$$

$$= \begin{bmatrix} e^{0.96}/(e^{0.96}+e^{-0.2}+e^{0.76}) \\ e^{-0.2}/(e^{0.96}+e^{-0.2}+e^{0.76}) \\ e^{0.76}/(e^{0.96}+e^{-0.2}+e^{0.76}) \end{bmatrix}$$

$$= \begin{bmatrix} 0.47 \\ 0.15 \\ 0.38 \end{bmatrix} \begin{matrix} \text{probability for class 0} \\ \text{prob for class 1} \\ \text{prob for class 2} \end{matrix}$$

c) Sigmoid is preferred for binary classification because it outputs a single probability btw 0 and 1 representing the likelihood of the positive class. Since there are only two class, the prob. of the other class is 1-p. Softmax is preferred for multi-class because it converts multiple output scores into a prob distribution across all classes, allowing the model to choose the most likely class among k mutually exclusive classes.

Ex 4:



a) . Hidden layer output h:

$$z_1 = w_1^T x + b$$

$$= [0.1, 0.2] \begin{bmatrix} 1 \\ 2 \end{bmatrix} + 0.3 = 0.8$$

$$h = G_1(z_1) \quad \text{with} \quad G_1 = \text{ReLU} = \max(0, z_1)$$

$$= \max(0, 0.8) = 0.8$$

b) . Predicted output $\hat{y}$:

$$\hat{y} = G_2(z_2) \quad \text{with} \quad G_2 = \frac{1}{1 + e^{-(vh+b)}}$$

$$= \frac{1}{1 + e^{-z_2}}$$

with $z_2 = vh + b = (0.5)(0.8) - 0.1 = 0.3$

$\hat{y} = \frac{1}{1 + e^{-0.3}} = 0.574$ The model predicts that there is a 57.4% chance of this sample to belong to class 1.

c) Computing loss using MSE:

$$L = \frac{1}{2n} \sum (y_i - \hat{y}_i)^2 \quad \text{with } n=1$$

$$= \frac{1}{2} (y_i - \hat{y}_i)^2$$

$$= \frac{1}{2} (1 - 0.574)^2$$

$$= 0.09$$

d) Using the chain rule:

$$\cdot \frac{dL}{dv} = \frac{dL}{d\hat{y}} \cdot \frac{d\hat{y}}{dz_2} \cdot \frac{dz_2}{dv}$$

with $\frac{dL}{d\hat{y}} = \left[\frac{1}{2} (y_i - \hat{y}_i)^2\right]' = -(y_i - \hat{y}_i) = -0.426$

$$\frac{d\hat{y}}{dz_2} = G_2'(z_2) = G_2(z_2)(1 - G_2(z_2))$$

$$= \hat{y}(1 - \hat{y}) = 0.244$$

$$\frac{dz_2}{dv} = (vh + b)' = h = 0.8$$

then $\frac{dL}{dv} = (-0.426)(0.24)(0.8) = -0.0834$

$$\cdot \frac{dL}{db_2} = \frac{dL}{d\hat{y}} \cdot \frac{d\hat{y}}{dz_2} \cdot \frac{dz_2}{db} = (-0.426)(0.24)(1) = -0.104$$

e) Using chain rule:

$$\frac{dL}{dw} = \frac{dL}{d\hat{y}} \cdot \frac{d\hat{y}}{dz_2} \cdot \frac{dz_2}{dh} \cdot \frac{dh}{dz_1} \cdot \frac{dz_1}{dw}$$

with $\frac{dz_2}{dh} = v = 0.5$; $\frac{dh}{dz_1} = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{else} \end{cases}$

$$\frac{dz_1}{dw} = x$$

- $\frac{dl}{dw} = (-0.426)(0.24)(0.5)(1)(1,2)$

$= -0.0522(1,2) = (-0.0521, -0.1042)$

- $\frac{dl}{db_1} = \frac{dl}{d\hat{y}} \cdot \frac{d\hat{y}}{dz_2} \cdot \frac{dz_2}{dL} \cdot \frac{dL}{dz_1} \cdot \frac{dh}{db}$

$= (-0.426)(0.24)(1)(0.5)(1) = -0.0521$

f) Using gradient descent update rule:

- output layer:

$v = v - \eta \frac{dl}{dv} = 0.5 - (0.01)(-0.0834) = 0.500834$

$b_2 = b_2 - \eta \frac{dl}{db_2} = -0.1 - (0.01)(-0.104) = -0.09896$

- Hidden layer:

$w = w - \eta \frac{dl}{dw} = (0.1, 0.2) - (0.01)(-0.0521, -0.1042)$

$= (0.100521, 0.201042)$

$b_1 = b_1 - \eta \frac{dl}{db_1} = 0.3 - (0.01)(-0.0521) = 0.300521$

g) If the hidden layer activation was tanh instead of ReLU, then the gradients would change to:

$$\cdot \frac{dl}{dv} = \frac{dl}{d\hat{y}} \cdot \frac{d\hat{y}}{dz_2} \cdot \frac{dz_2}{dv}$$

$$= -(y-\hat{y})^2 \cdot \hat{y}(1-\hat{y}) \cdot h \quad \text{(no change)}$$

$$\cdot \frac{dl}{dw} = \frac{dl}{d\hat{y}} \cdot \frac{d\hat{y}}{dz_2} \cdot \frac{dz_2}{dh} \cdot \frac{dh}{dz_1} \cdot \frac{dz_1}{dw}$$

with $\frac{dh}{dz_1} = G_1'(z_1)$ with $G_1 = \tanh$

$$= 1 - [G_1(z_1)]^2$$

$$= 1 - h^2$$

$$\frac{dl}{dw} = -(y-\hat{y})^2 \cdot \hat{y}(1-\hat{y}) \cdot h \cdot (1-h^2) \cdot x$$

. The gradient $\frac{dh}{dz_1}$ is $1-\tanh^2$ for tanh fct instead of 1 for ReLU fct.
Since $\tanh^2 \leq 1$, then the gradient will be smaller, leading to slower convergence compared to ReLU.