ML Ex: 17/18 (1)

Ex1 (a) (i) Advantages

→ GD is computationally efficient and cheap as only one observation or a mini-batch is processed at a time, and therefore requires less memory. It can converge faster for large datasets.

Disadvantages

→ Choosing the right step size is important, otherwise the gradient descent can lean into other directions, and it can take longer to achieve convergence. It does not benefit from the advantages offered by vectorized operations, which can be addressed by Batch GD.

(ii) In GD, we use first derivative to find the descent direction while in Newton, we use second derivative. In other words, we fit a parabola over our function at each step in Newton's method instead of a line in GD. This helps in much faster convergence.

(b) $\beta_0^T = [0.05 \quad 0.05 \quad 0.05]$, $\alpha = 0.6$

$$X = \begin{bmatrix} 1 & 11 & 3 \\ 1 & 14 & 4 \\ 1 & -16 & -5 \\ 1 & -18 & -6 \end{bmatrix} \; \& \; Y = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \quad X\beta_0 = \begin{bmatrix} 7.5 \\ 9.5 \\ -10 \\ -11.5 \end{bmatrix}, \quad \hat{Y} = \frac{1}{1+e^{-X\beta_0}} = \begin{bmatrix} 0.9994 \\ 0.9999 \\ 4.5\times10^{-5} \\ 1\times10^{-5} \end{bmatrix}$$

$$\beta_1 = \beta_0 + \alpha \cdot X^T(Y-\hat{Y}) = \beta_0 + 0.6\, X^T \begin{bmatrix} 6\times10^{-4} \\ 1\times10^{-4} \\ -4.5\times10^{-5} \\ -1\times10^{-5} \end{bmatrix} = \beta_0 + 0.6 \begin{bmatrix} 6.4\times10^{-4} \\ 8.9\times10^{-3} \\ 2.4\times10^{-3} \end{bmatrix} = \begin{bmatrix} 0.0504 \\ 0.0553 \\ 0.0514 \end{bmatrix}$$

$$\text{Accuracy} = 1 - \text{loss} \quad \& \quad \text{Loss} = -Y\log(\hat{Y}) - (1-Y)\log(1-\hat{Y})$$

$$= 1 - 7.55\times10^{-4} = \underline{99.92\%} \qquad = 7\times10^{-4} + 5.5\times10^{-5} = 7.55\times10^{-4}$$

(c) $$RSS = \sum_{i=1}^{N} (y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i))^2$$

$$\frac{\partial RSS}{\partial \hat{\beta}_0} = \sum_{i=1}^{N} 2(y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i))(-1) = 0$$

$\sum_{i=1}^{N} 1 = N$, so, $N\hat{\beta}_0 = \sum_{i=1}^{N} (y_i - \hat{\beta}_1 x_i)$

ML Ex: 17/18 (1)

Ex1 c. $\hat{\beta}_0 = \frac{1}{N}\sum_{i=1}^{N} y_i - \frac{1}{N}\sum_{i=1}^{N}\hat{\beta}_1 x_i = \hat{\beta}_0 = \bar{y} - \hat{\beta}_1\bar{x}$

$\frac{\partial RSS}{\partial \beta_1} \cancel{= \sum_{i=1}^{N} 2(y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i))(x_i) = 0}$

$RSS = \sum_{i=1}^{N}(y_i - (\bar{y} - \hat{\beta}_1\bar{x}) - \hat{\beta}_1 x_i)^2 = \sum_{i=1}^{N}(y_i - \bar{y} - \hat{\beta}_1(x_i - \bar{x}))^2$

$\frac{\partial RSS}{\partial \hat{\beta}_1} = \cancel{\sum_{i=1}^{N} 2(y_i - (\bar{y} - \hat{\beta}_1\bar{x}) - \hat{\beta}_1 x_i)(\bar{x} - x_i) = 0}$

$= \sum_{i=1}^{N} 2(y_i - \bar{y} - \hat{\beta}_1(x_i - \bar{x}))(-1)(x_i - \bar{x}) = 0$

$\Rightarrow -\sum_{i=1}^{N}(y_i - \bar{y})(x_i - \bar{x}) + \hat{\beta}_1\sum_{i=1}^{N}(x_i - \bar{x})^2 = 0 \Rightarrow \hat{\beta}_1 = \frac{\sum_{i=1}^{N}(y_i - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^{N}(x_i - \bar{x})^2}$

Ex 2(a) DTs can be regularized by ~~either~~ (1) minimizing the number of points per cell (2) limiting max. no. of cells (3) limiting the depth. (Slide 22). Without regularization, we can end up having a separate leaf node for each training point.

Ex 2(b)

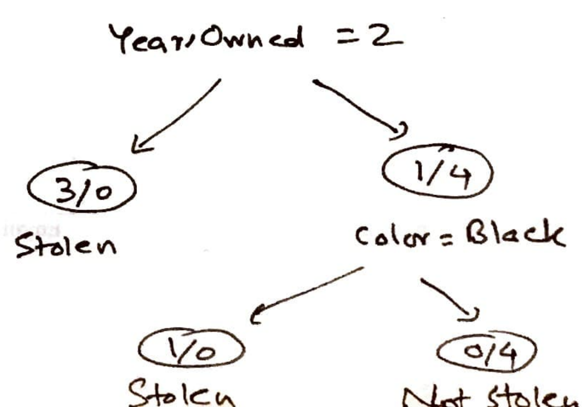
Variable	Value	Stolen Yes	Stolen No	MCR
Color	{Red}	1	3	2/8
	{BK, Bl}	3	1	
Color	{BK}	2	0	2/8
	{R, Bl}	2	4	
Color	{Bl}	1	1	4/8
	{R, BK}	3	3	
Years	{2}	3	0	1/8
	{6, 8}	1	4	
Years	{6}	1	2	3/8
	{2, 8}	3	2	
Years	{8}	0	2	2/8
	{2, 6}	4	2	

First split Years Owned = 2

Second split.

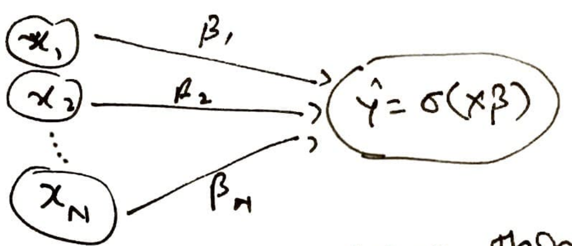
Var.	Value	Stolen Y	Stolen N	MCR
Color	{R}	0	3	1/5
	{BK, Bl}	1	1	
Color	{Bk}	1	0	0
	{Bl, R}	0	4	
Color	{Bl}	0	1	2/5
	{R, Bk}	1	3	
Years	{6}	1	2	1/5
	{8}	0	2	

2nd Split Color = Black



Ex 2c. MCR fails to recognize leaf nodes and thus may lead to longer tree depths. e.g. Splits $\begin{bmatrix} 2 & 0 \\ 2 & 4 \end{bmatrix}$ & $\begin{bmatrix} 1 & 3 \\ 3 & 1 \end{bmatrix}$ have the same MCR but first split should be preferred which MCR cannot account for.

Ex 3(a) (i) Logistic binary linear ~~classification~~ regression is given by $\hat{y} = \dfrac{1}{1+e^{-x\beta}}$, where $x = \{x_1, x_2 \dots x_N\}$



A FFNN is shown with one linear layer and sigmoid activation function. As can be seen, both are doing the same thing.

(ii) computing gradients in reverse order of computations of predictions is called backpropagation. It is used for learning a NN.

Ex 3(b) $x = 0.1$, $y = 0.1$, sigmoid, $\mu = 1$

$a_1 = x W_{12} = \begin{bmatrix} 0.01 \\ 0.05 \end{bmatrix}$, $z_1 = \sigma(a_1) = \begin{bmatrix} 0.503 \\ 0.513 \end{bmatrix}$, $a_2 = z_1^T W_{34} = 0.356$, $\hat{y} = \sigma(a_2) = 0.588$

$\ell = -y\log(\hat{y}) - (1-y)\log(1-\hat{y}) = 0.053 + 0.798 = 0.851$

$\dfrac{\partial \ell}{\partial W_{34}} = \dfrac{\partial \ell}{\partial \hat{y}} \cdot \dfrac{\partial \hat{y}}{\partial a_2} \cdot \dfrac{\partial a_2}{\partial W_{34}} = \dfrac{\hat{y}-y}{\hat{y}(1-\hat{y})} \operatorname{diag}(\hat{y}(1-\hat{y})) \cdot z_1 = +2.014 \times 0.242 \times \begin{bmatrix} 0.503 \\ 0.513 \end{bmatrix} = \begin{bmatrix} +0.245 \\ +0.25 \end{bmatrix}$

$W'_{34} = W_{34} - \mu \dfrac{\partial \ell}{\partial W_{34}} = \begin{bmatrix} 0.3 \\ 0.4 \end{bmatrix} - (1)\begin{bmatrix} +0.245 \\ +0.25 \end{bmatrix} = \cancel{\begin{bmatrix} 0.545 \\ 0.65 \end{bmatrix}} \begin{bmatrix} 0.055 \\ 0.15 \end{bmatrix}$

$\dfrac{\partial \ell}{\partial W_{12}} = \dfrac{\partial \ell}{\partial \hat{y}} \cdot \dfrac{\partial \hat{y}}{\partial a_2} \cdot \dfrac{\partial a_2}{\partial z_1} \cdot \dfrac{\partial z_1}{\partial a_1} \cdot \dfrac{\partial a_1}{\partial W_{12}} = +2.014 \times 0.242 \times \begin{bmatrix} 0.3 \\ 0.4 \end{bmatrix}^T \times \begin{bmatrix} 0.25 & 0 \\ 0 & 0.25 \end{bmatrix} \times x$

$= +0.487 \begin{bmatrix} 0.075 & 0.1 \end{bmatrix} 0.1 = \begin{bmatrix} +0.0036 \\ +0.0049 \end{bmatrix}$

$W_{12} = \begin{bmatrix} 0.1 \\ 0.5 \end{bmatrix} - \begin{bmatrix} +0.0036 \\ +0.0049 \end{bmatrix} = \cancel{\begin{bmatrix} 0.1036 \\ 0.505 \end{bmatrix}} \begin{bmatrix} 0.0964 \\ 0.4951 \end{bmatrix}$

$a_1 = x W_{12} = \cancel{\begin{bmatrix} 0.0104 \\ 0.0505 \end{bmatrix}} \begin{bmatrix} 0.0096 \\ 0.0495 \end{bmatrix}$, $z_1 = \sigma(a_1) = \begin{bmatrix} 0.5024 \\ 0.5124 \end{bmatrix}$, $a_2 = 0.1045$, $\hat{y} = \cancel{0.647}\ 0.526$

$\ell = 0.064 + 0.672 = 0.736 \downarrow$

ML Ex: 17/18 (1)

Ex 3c

	x_1	x_2
H_1	1	1
H_2	1	-1
H_3	-1	1
H_4	1	1

level = 0 for H_1 & ≥ 1 for $H_2 - H_4$

So, for (0, 0), ~~H_1 = th(0, 0) = 1~~

~~(1, 0)~~, ~~H_2 = th(2, 0)~~

(0, 1), H_3 =

	Threshold
H_1	$x = 0$
H_2	$x = 1$
H_3	$x = 1$
H_4	$x = 2$

1 vs. all ? 1 vs. lest ?

Ex 4a (i) Clustering, Dimensionality Reduction & Freq. Pattern mining.

(ii) In soft clustering, the soft partition matrix is row-stochastic and does not contain a zero column. In other words, it builds overlapping groups to which data points can belong with some membership degree. While in hard clustering, each data point can only belong to one cluster.

(iii) 'complete' means that we calculate joint probabilities of all observations and latent variables.

Ex 4b:

	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8
A_1	0	$\sqrt{25}$	$\sqrt{36}$	$\sqrt{13}$	$\sqrt{50}$	$\sqrt{52}$	$\sqrt{65}$	$\sqrt{5}$
A_4	$\sqrt{13}$	$\sqrt{18}$	$\sqrt{25}$	0	$\sqrt{13}$	$\sqrt{17}$	$\sqrt{52}$	$\sqrt{2}$
A_7	$\sqrt{65}$	$\sqrt{10}$	$\sqrt{53}$	$\sqrt{52}$	$\sqrt{45}$	$\sqrt{29}$	$\sqrt{0}$	$\sqrt{58}$

P_1: A_1

P_2: A_3 A_4 A_5 A_6 A_8

A_7

P_3: A_2

For P_1, $\Sigma = 0$, P_2, Σ = [14.23 $\underset{A_4}{14.14}$ 11.43 11.95 18.2], P_3, Σ = [3.16 $\underset{A_7}{3.16}$])

$\mu_1' = [2 \quad 10]$, $\mu_2' = [6 \quad 6]$, $\mu_3' = [3 \quad 7]$

	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8
~~A_1~~ μ_1	0	$\sqrt{25}$	$\sqrt{36}$	$\sqrt{13}$	$\sqrt{50}$	$\sqrt{52}$	$\sqrt{65}$	$\sqrt{5}$
~~A_4~~ μ_2	$\sqrt{32}$	$\sqrt{17}$	$\sqrt{16}$	$\sqrt{5}$	$\sqrt{2}$	$\sqrt{2}$	$\sqrt{41}$	$\sqrt{13}$
μ_3	$\sqrt{10}$	$\sqrt{5}$	$\sqrt{34}$	$\sqrt{5}$	$\sqrt{20}$	$\sqrt{18}$	$\sqrt{29}$	$\sqrt{5}$

$\mu_1'' = [2 \quad 10]$

$\mu_2'' = [6.5 \quad 5.25]$

$\mu_3'' = [2.5 \quad 5.5]$

P_1: A_1

P_2: A_3 A_4 A_5 A_6 ~~A_7~~

P_3: A_2 A_7 A_8