1. GRADIENT DESCENT

(i) We are given $f_a(x) = ax^2$ for $a > 0$

Let $a = 2$, we consider 2-step sizes
$\eta_1 = \frac{1}{2}$, $\eta_2 = 2$
observation at starting gradient descent at
$x_0 = 1$?

Soln.

Since we already know the value of $a = 2$, then we proceed as below

$f(x) = 2x^2$ - this is the function we gonna be using to determined the optimal solution with given step sizes

=> We first find the derivative of the above function

$$\Rightarrow \frac{df(x)}{dx} = 4x \quad \text{———(ii)}$$

=> Then at $x_0 = 1$, Eqn (ii) will result to

$$f'(x) = \left.\frac{df(x)}{dx} = 4x\right|_{x=1}$$

$$= f'(1) = \frac{df(1)}{dx} = 4(1) = 4$$

=> One observation to note is that the derivative is positive, i.e $= 4$

=> So with this we already know that the optimal value solution is getting bigger / larger thus we want to go backwards

=> In this aspect we use the below function, that

$$x_{i+1} = x_i - \eta f'(x) \quad \text{—— (iii)}$$

=> So we need to find the result of x given the 2-step sizes as follow

$$x_{i+1} = x_i - \eta f'(x) \Big|_{x=1} \quad \text{and} \quad \eta = \tfrac{1}{2}$$

=> $x_{i+1} = 1 - \frac{1}{2} f'(x)$ but $f'(x) = 4$

$= x_{i+1} = 1 - \frac{1}{2}(4) = -1$

=> We use -1 as the new x value to feed forward to the eqn (iii) as below

=> First we compute the derivative value $f'(x)$ at the new x value i.e. -1

=> $f'(x) = 4x \Big|_{x=-1} = 4(-1) = -4$

=> The the resulting observation of the x new value will be given as

$$x_{i+1} = -1 - 2(-4) = 7$$

=> So starting gradient decent will result to a sudden increase and the diversion of family of functions $f(x)$ which is contributed by a resulting increase in x values starting $x_0 = 1$

Determine set of all step sizes $\eta > 0$ for which the gradient descent will fail to converge to the minimum.

Soln

=> So inorder to find the interval or set of all step sizes, we need to consider the following

(i) First we need to find the value of η at which the gradient descent will converge

(ii) And then using that η-value we determine the x_0 value to confirm that it is actually starting at $x_0 = 0$

(iii) Then to determine the set of all step sizes, we have to take opposite interval starting but not including the η-value

i.e

=> Since we are guaranteed of a positive derivative from $f'(x) = 2x^2$, we know the correct function to use is given by

$$x_{i+1} = x_i - \eta f'(x)$$

=> Now let $x_0 = 1$ from (i) and assuming η is not given we have.

$$f'(x) = 4x \Big|_{x=1} = 4$$

$\Rightarrow x_{i+1} = 1 - \eta(4) = 1 - 4\eta$

$\Rightarrow$ Now we use this value as an new x-value and substituting it in $f(x) = 2x^2$ we have

$$f(1-4\eta) = 2(1-4\eta)^2 = 2[1-8\eta+16\eta^2]$$

$$2 - 16\eta + 32\eta^2 = g(\eta)$$

$$\Rightarrow g(\eta) = 32\eta^2 - 16\eta + 2$$

$\Rightarrow$ So to get $g_{minimum}$, we set $\frac{dg}{d\eta} = 0$

$$\Rightarrow \frac{dg(\eta)}{d(\eta)} = 64\eta - 16 = 0$$

$$\Rightarrow 64\eta = 16 \quad \Rightarrow \eta^* = 0.25$$

$\Rightarrow$ lets determine x_{i+1}

$$\Rightarrow x_{i+1} = 1 - 4\eta \Big|_{\eta=0.25} = 1 - 1 = 0$$

$$\Rightarrow x_0 \neq 0$$

$\Rightarrow$ So the step-size of η includes all values other than $\eta = 0.25$ and $\eta > 0$

$\Rightarrow$ open interval from 0.25 to $+\infty$

Set of of all $\eta = \{\eta \in \mathbb{R} / 0.25 < \eta \leq \infty\}$

$\mathbb{R}$ is Real no. System

(iii) Let $\varepsilon > 0$, after how many steps, i do we have $|x_i - x^*| < \varepsilon$

Soln

In other words we are to prove the number of iterations (i) need to achieve convergence of the gradient descent function

=> So suppose the function given $f_a(x) : \mathbb{R}^n \to \mathbb{R}$ is convex and differentiable, and that its gradient is Lipschitz continuous with constant $L > 0$, i.e

for any x, if we run gradient descent for i iterations with step size η, then we yield a solution $f^{(i)}$ which statisties the below function

$$f(x^{(i)}) - f(x^*) \leq \frac{\|x^{(0)} - x^*\|_2^2}{2ti}$$

The rate of convergence of this function is the $O(1/i)$

=> So with this we can easily determine the i iterations needed for convergence

=> We consider the bound of $f(x^{(i)}) - f(x^*) \leq \varepsilon$ which implies that $|x_i - x^*| < \varepsilon$

=> Thus with linear convergence rate shown above it can only be achieved using only

$O(\log(1/\varepsilon))$ number of i iterations

2. Simple Neural Nets

(i) Soln

So in this sense the multiclass logistic regression model is more powerful and able to represent more complex relationships between inputs and output

=> One important key takeaway when determining efficiency of a model is that we need to consider the number of hidden-units which should in most cases lesser than the twice the size of the input layer

=> In our case with 25 hidden units and the number of classes in the input layer being only 10, this may lead to the model being overfitted and with the fact that no regularization on that model => simply means that this model will work only efficiently on the training data but will fail to make correct predictions on the output of the actual / test data

=> Thus the number of hidden units should be kept between the size of the input layer and the size of the output layer which is not the case with the 25-hidden-units model thus making it more prone to overfitting and hence less powerfull than the multi class logistic regression

=> The multi class logistic regression can deal with arbitrarily number of classes hence outputting values for each class with no overfitting

=> Meaning that the model works better on both training and testing data with high accuracy

with 4 hidden units, the model will be a bit powerful than the multiclass logistic regression in the sense that, the number of hidden units is kept in between the size of the input layer and the size of the output layer

=> Also during training this model will out-perform multiclass_logistic regression because of the lesser number of hidden units used in the hidden layer making it pretty fast. and powerful

=> We also need to mention that the chosen hidden ~~lay~~ units makes the model not being underfitted => implying that it is going to work efficiently on both the training and testing data with high accuracy and fast speed

=> Thus model with 4 hidden units is better in this case than multiclass logistic regression - due to the fact it takes lesser time to train the model and will obtains high accuracy with the actual datagen

(iii) Convex optimisation problem if we use cross entropy loss function on the output layer?

Soln

=> To answer this problem we need to find the 2nd order derivative of the below function w.r.t β (cross entropy function)

=> thus we have.

$$h = \frac{1}{N} \sum_{i=1}^{N} \left[-y_i \log(\sigma_i) - (1-y_i) \log(1-\sigma_i) \right]$$

=> so we need to find the 2nd order derivative of this function, so if it is greater than or equal to zero, then we can make conclusions

$$\partial^2 h / \partial \beta^2 = ?$$

=> let $h_i = -y_i \log(\sigma_i) - (1-y_i) \log(1-\sigma_i)$

$$\Rightarrow \partial h_i / \partial \beta_K = -y_i \frac{\partial \log(\sigma_i)}{\partial \beta_K} - (1-y_i) \frac{\partial \log(1-\sigma_i)}{\partial \beta_K}$$

$$= -y_i \frac{\partial \sigma_i}{\partial \beta_K} \frac{1}{\sigma_i} + (1-y_i) \frac{\partial \sigma_i}{\partial \beta_K} \frac{1}{1-\sigma_i}$$

with $\frac{\partial \sigma_i}{\partial \beta_K} = \sigma(Lm) \cdot (1 - \sigma(Lm)) \frac{Lm}{\partial \beta_K}$

=> This simplifies into

$$= \left(-y_i (1-\sigma_i) + (1-y_i)\sigma_i\right) \frac{\partial LM}{\partial \beta_K}$$

$$= (\sigma_i - y_i) \,\partial LM / \partial \beta_K$$

=> i.e $-y_i + \cancel{y_i \cdot \sigma_i} + \sigma_i - \cancel{y_i \cdot \sigma_i} \;\partial LM / \partial \beta_K$ =

=> Now on to the 2nd order derivative,

$$\partial^2 h_i / \partial \beta_K^2 = \partial \left[\sigma_i - y_i\right] \frac{\partial LM}{\partial \beta_K} \Big/ \partial \beta_K$$

$$\Rightarrow \partial^2 h_i / \partial \beta_K^2 = \frac{\partial \sigma_i}{\partial \beta_K} \,\partial LM / \partial \beta_K$$

with $\partial \sigma_i / \partial \beta_K = \sigma(LM) \cdot (1 - \sigma(LM)) \,\partial LM / \partial \beta_K$

Therefore $\sigma_i (1-\sigma_i) \left(\frac{\partial LM}{\partial \beta_K}\right)^2$ and $\partial^2 LM / \partial \beta_K^2 = 0$

$$\underset{\geqslant 0}{\Downarrow} \qquad \underset{\geqslant 0}{\Downarrow} \quad \text{with } \sigma_i \in [0,1]$$

Thus we have that $\partial^2 h / \partial \beta_K^2 \geqslant 0$

=> Thus we can conclude that with cross entropy loss function on the output layer => We get a convex optimisation problem

☒

3. Digit Classification Using MLPs

(i) To reduce the size of the network and increase efficiency
=> suggestions
(i) Using binary Encoding instead of one-hot-Encoding
(ii) Reducing number of neurons in the middle layer
What to think about this suggestion?

Soln

So to answer the above question we gonna implement the exact neural network ~~one~~ as ~~described above and~~ Compared it also described above with binary encoding, lesser neurons in the middle layer and compared its performance with a NN with one-hot-encoding

=>
So now based on the experiment conducted on the implemented model with binary encoding and lesser neurons => this does not actually reduce the size and efficiency increase
=> By evaluating the two models performance
(i) Model using binary encoding had a accuracy of ~~98~~ 87%
(ii) Model using one_hot encoding had an accuracy of 96%

=> From these results it is clear that these suggestions actually do not result into a more efficient model
=> Thus a model with one-hot-encoding is much more improved and high efficient.

See Code attached [Question 3B]

=> So from the result obtained from the run models we obtained a 96% accuracy on the model using one_hot_encoding while and 92% accuracy on the model using binary encoding and an additional layer which is relatively close

See attached code for verification

What happens when the NN above is trained without adding the last layer

=> From the experimental observations is that training this model directly we are actually getting a lesser efficient model with a lower accuracy of 87%

See attached code [Question 3C]