OPTIMIZATION METHODS

HOMEWORK 5

Name : SWETANJAL DUTTA

ROLL No:- 20171077

1) Let $f(x) = \frac{1}{x}$

Now, $f'(x) = -\frac{1}{x^2}$ and $f''(x) = \frac{2}{x^3}$

$\forall x > 0$, $f''(x) > 0 \Rightarrow f(\cdot)$ is convex in the interval $(0, \infty)$.

Note that the result stated in the problem only holds for $x \in \mathbb{R}^+ - \{1\}$

By Jensen's Inequality,

$$f\left(\frac{1\cdot a + 1\cdot b + 1\cdot c}{1+1+1}\right) \leq \frac{1\cdot f(a) + 1\cdot f(b) + 1\cdot f(c)}{1+1+1}$$

Now, let $a = x-1$, $b = x$, $c = x+1$:

$$f\left[\frac{x-1+x+x+1}{3}\right] \leq \frac{f(x-1)+f(x)+f(x+1)}{3}$$

$$\Rightarrow f(x) \leq \left\{\frac{1}{x-1} + \frac{1}{x} + \frac{1}{x+1}\right\} \cdot \frac{1}{3}$$

$$\Rightarrow \frac{3}{x} \leq \frac{1}{x-1} + \frac{1}{x} + \frac{1}{x+1}$$

As x & $x+1$ are never equal for $x \in \mathbb{R}$, the equality condition does not hold.

$$\therefore \frac{1}{x-1} + \frac{1}{x} + \frac{1}{x+1} > \frac{3}{x}$$

Note, we could have proved this result by simple manipulation as follows:

$$0 > -1$$

$$\Rightarrow x^2 > x^2 - 1 \quad \forall\, x \in \mathbb{R}$$

$$\Rightarrow \frac{1}{x^2-1} > \frac{1}{x^2}$$

$$\Rightarrow \frac{x}{x^2-1} > \frac{1}{x} \quad \text{assuming } x > 0$$

$$\Rightarrow \frac{2x}{x^2-1} > \frac{2}{x} \quad \Rightarrow \frac{(x-1)+(x+1)}{(x-1)(x+1)} > \frac{2}{x}$$

$$\Rightarrow \frac{1}{x-1} + \frac{1}{x+1} > \frac{2}{x}$$

$$\Rightarrow \frac{1}{x-1} + \frac{1}{x} + \frac{1}{1+x} > \frac{3}{x}$$

Clearly, the above is a strict inequality.

2) $f: [a,b] \longrightarrow \mathbb{R}$ is convex & $x \in (a,b)$ if $\omega_1 = \dfrac{b-x}{b-a}$

& $\omega_2 = \dfrac{x-a}{b-a}$; $\omega_1, \omega_2 \geq 0$

and $\omega_1 + \omega_2 = 1$.

$\therefore$ By Jensen's Inequality,

$$f(\omega_1 a + \omega_2 b) \leq \omega_1 f(a) + \omega_2 f(b)$$

$$\Rightarrow f\left(\frac{(b-x)a + (x-a)b}{b-a}\right) \leq \frac{(b-x)f(a) + (x-a)f(b)}{b-a}$$

$$\Rightarrow f\left(\frac{ab - ax + bx - ab}{b-a}\right) \leq \frac{bf(a) - xf(a) + xf(b) - af(b)}{b-a}.$$

$$\Rightarrow f\left(\frac{(b-a)x}{b-a}\right) \leq \frac{x(f(b)-f(a)) + bf(a) - af(b)}{b-a}$$

$$\Rightarrow f(x)(b-a) \leq x(f(b)-f(a)) + bf(a) - af(b) \quad \text{— (I)}$$

$$\Rightarrow f(x)(b-a) - f(a)\cdot b \leq x(f(b)-f(a)) - af(b)$$

Adding $af(a)$ on both sides,

$$f(x)(b-a) - f(a)(b-a) \leq x(f(b)-f(a)) - a(f(b)-f(a))$$

$$\Rightarrow \{f(x)-f(a)\}(b-a) \leq \{f(b)-f(a)\}(x-a)$$

$$\Rightarrow \frac{f(x)-f(a)}{x-a} \leq \frac{f(b)-f(a)}{b-a} \quad \text{— (II)}$$

From (I),

~~$f(x)(b-a) + af(b) \leq x(f(b)-f(a)) - b(f(b)-f(a))$~~

$$f(x)(b-a) + af(b) \leq x(f(b)-f(a)) + bf(a)$$

Subtracting $bf(b)$ from both sides,

$$f(x)(b-a) - f(b)(b-a) \leq x(f(b)-f(a)) - b(f(b)-f(a))$$

$$\Rightarrow (f(x)-f(b))(b-a) \leq (x-b)(f(b)-f(a))$$

Considering the negated inequality,

$$(f(b)-f(x))(b-a) \geq (b-x)(f(b)-f(a))$$

$$\Rightarrow \frac{f(b)-f(x)}{b-x} \geq \frac{f(b)-f(a)}{b-a} \quad \text{— (III)}$$

From (II) and (III),

$$\frac{f(x)-f(a)}{x-a} \leq \frac{f(b)-f(a)}{b-a} \leq \frac{f(b)-f(x)}{b-x}$$

Hence, proved.

3) Given : $x, y, z \in \mathbb{R}^+$ & $x+y+z=1$

Let $\log\left(1+\frac{1}{x}\right) \equiv f(x) \quad \forall\, x \in \mathbb{R} - [-1, 0]$

Now, $f'(x) = \frac{1}{1+\frac{1}{x}} \cdot -\frac{1}{x^2} = \frac{x}{x+1} \cdot \frac{-1}{x^2}$

$$= -\frac{1}{x(x+1)} = \frac{x-(x+1)}{x(x+1)}$$

$$= \frac{1}{x+1} - \frac{1}{x}$$

$$f''(x) = -\frac{1}{(x+1)^2} + \frac{1}{x^2} = \frac{1}{x^2} - \frac{1}{(x+1)^2}$$

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

Clearly; $f''(x) > 0 \quad \forall\, x > -\frac{1}{2}$

$\Rightarrow f(x)$ is convex $\forall\, x > 0$ as $f(x)$ is not defined in the interval $\left[-\frac{1}{2}, 0\right]$

Using Jensen's Inequality, we get

$$f\left(\frac{x+y+z}{3}\right) \leq \frac{1}{3} f(x) + \frac{1}{3} f(y) + \frac{1}{3} f(z)$$

$$\Rightarrow f\left(\tfrac{1}{3}\right) \leq \frac{1}{3}\left\{\log\left(1+\frac{1}{x}\right) + \log\left(1+\frac{1}{y}\right) + \log\left(1+\frac{1}{z}\right)\right\}$$

$$\Rightarrow \log\left(1+\frac{\frac{4}{3}}{\frac{1}{3}}\right) \leq \frac{1}{3}\left\{\log\left(\left(1+\frac{1}{x}\right)\left(1+\frac{1}{y}\right)\left(1+\frac{1}{z}\right)\right)\right\}$$

$$\Rightarrow 3\log 4 \leq \log\left\{\left(1+\frac{1}{x}\right)\left(1+\frac{1}{y}\right)\left(1+\frac{1}{z}\right)\right\}$$

$$\Rightarrow \log\left\{\left(1+\frac{1}{x}\right)\left(1+\frac{1}{y}\right)\left(1+\frac{1}{z}\right)\right\} \geq \log 4^3$$

$$\Rightarrow \left(1+\frac{1}{x}\right)\left(1+\frac{1}{y}\right)\left(1+\frac{1}{z}\right) \geq 64 \quad \text{--- (I)}$$

[as $g(x) = e^x$ is a strictly increasing function]

Equality holds in (I) for

$$x = y = z = \frac{1}{3}$$

4) $f(x) = x^3 - 2x - 5$

Goal: Find roots of the equation $f(x) = 0$ using the secant method.

Given: $x_0 = 2$, $x_1 = 3$

We know: the recurrence relation for the secant method is given by:

$$x_n = \frac{x_{n-2} f(x_{n-1}) - x_{n-1} f(x_{n-2})}{f(x_{n-1}) - f(x_{n-2})}$$

$$\therefore x_2 = \frac{x_0 f(x_1) - x_1 f(x_0)}{f(x_1) - f(x_0)}$$

$$= \frac{2f(3) - 3f(2)}{f(3) - f(2)}$$

Now, $f(3) = 3^3 - 2 \times 3 - 5 = 16$

& $f(2) = 2^3 - 2 \times 2 - 5 = -1$

$$\therefore x_2 = \frac{2 \times 16 - 3(-1)}{16 - (-1)} = \frac{35}{17} \approx 2.05882$$

And:

$$x_3 = \frac{x_1 f(x_2) - x_2 f(x_1)}{f(x_2) - f(x_1)}$$

$$= \frac{3 f\left(\frac{35}{17}\right) - \frac{35}{17} f(3)}{f\left(\frac{35}{17}\right) - f(3)}$$

$$\cdot f\left(\frac{35}{17}\right) = \left(\frac{35}{17}\right)^3 - 2 \times \frac{35}{17} - 5$$

$$\simeq -0.39079$$

$$\therefore x_3 = \frac{3 \times (-0.39079) - \frac{35}{17} \times 16}{-(0.39079) - 16}$$

$$\simeq 2.0812636598$$

This concludes ~~in~~ two iterations.

5) $g(x) = xe^x - 1$ such that $x \in [0, 1]$

Now, $g(0) = -1$ and $g(1) = e - 1 > 0$

Let $a = 0$ & $b = 1$ be our initial values.

Iteration 1

$a = 0$; $b = 1$

$$mid = \frac{0 + 1}{2} = 0.5$$

$$g(0.5) = \frac{1}{2} e^{\frac{1}{2}} - 1 \approx -0.175639 < 0$$

$\therefore a = mid$

Iteration 2

$a = \frac{1}{2}$, $b = 1$

$$mid = \frac{\frac{1}{2} + 1}{2} = 0.75$$

$$g(0.75) = \frac{3}{4} e^{\frac{3}{4}} - 1 \approx 0.58775$$

$\therefore b = mid.$

6) Newton's Method Update Rules :-

$$x_{n+1} = x_n - \frac{f(x_n, y_n)}{\frac{\partial f(x_n, y_n)}{\partial x}}$$

$$y_{n+1} = y_n - \frac{f(x_n, y_n)}{\frac{\partial f(x_n, y_n)}{\partial y}}$$

$$f(x, y) = x^2 - y^2 - 3$$

$$\frac{\partial f}{\partial x} = 2x \qquad \frac{\partial f}{\partial y} = -2y$$

Iteration 1

$$x_1 = 2.55 + \frac{3}{5.1} = 3.14$$

$$y_1 = 2.55 - \frac{3}{5.1} = 1.96$$

Iteration 2

$$x_2 = 3.14 - \frac{3.018}{6.28} = 2.66$$

$$y_2 = 1.96 + \frac{3.018}{3.92} = 2.73$$

$$f(x,y) = x^2 + y^2 - 13$$

$$\frac{\partial f}{\partial x} = 2x \qquad \frac{\partial f}{\partial y} = 2y.$$

Iteration 1

$$x_1 = 2.55 - \frac{0}{5.1} = 2.55$$

$$y_1 = 2.55 - \frac{0}{5.1} = 2.55$$

Iteration 2

$$x_2 = 2.55 - \frac{0}{5.1} = 2.55$$

$$y_2 = 2.55 - \frac{0}{5.1} = 2.55$$

7) $g(x) = \dfrac{\cos(x) + 3}{2}$

Now, $g(x) = x$

$\Rightarrow \dfrac{\cos(x) + 3}{2} = x \quad \Rightarrow \cos x + 3 = 2x$

$\Rightarrow 2x - \cos(x) - 3 = 0.$

We run the fixed point iteration scheme with $x_0 = \dfrac{\pi}{2}$.

Iteration 1:

$x_0 = \dfrac{\pi}{2}, \quad x_1 = g(x_0) = g\left(\dfrac{\pi}{2}\right) = \dfrac{\cos\frac{\pi}{2} + 3}{2}$

$= \dfrac{3}{2} = 1.5$

Iteration 2:

$x_1 = \dfrac{3}{2}, \quad x_2 = g(x_1) = g\left(\dfrac{3}{2}\right)$

$= \dfrac{\cos\left(\frac{3}{2}\right) + 3}{2}$

$= 1.5353686.$

8) In gradient ascent, we go in the direction of gradient to reach the peak in the function curve.

Let $f(x)$ be the function which we are trying to maximize; x being the sequence of parameters or values used in the computation of f.

Algorithm:-

1. Initialize x to a suitable value.
2. Initialize prev_x to a value sufficiently different from x.
3. While L2 norm of $(x - x\_prev) > \theta$, do.

a) let $A \leftarrow x + \eta \frac{\partial f(x)}{\partial x}$

where $\frac{\partial f(x)}{\partial x} = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \\ \vdots \\ \frac{\partial f(x)}{\partial x_d} \end{bmatrix}$

assuming X is a vector of dimensionality d. This is done to allow simultaneous updates.

b) prev_x = x

c) x ← A

4. Output f^* as $f(x)$

5. Output x^* as x

Note: η is the learning rate

θ = threshold norm for convergence.

let $f(x) \equiv 6 - (x_1^2(x_1^2 - 16) + x_2^2(x_2^2 - 9))$

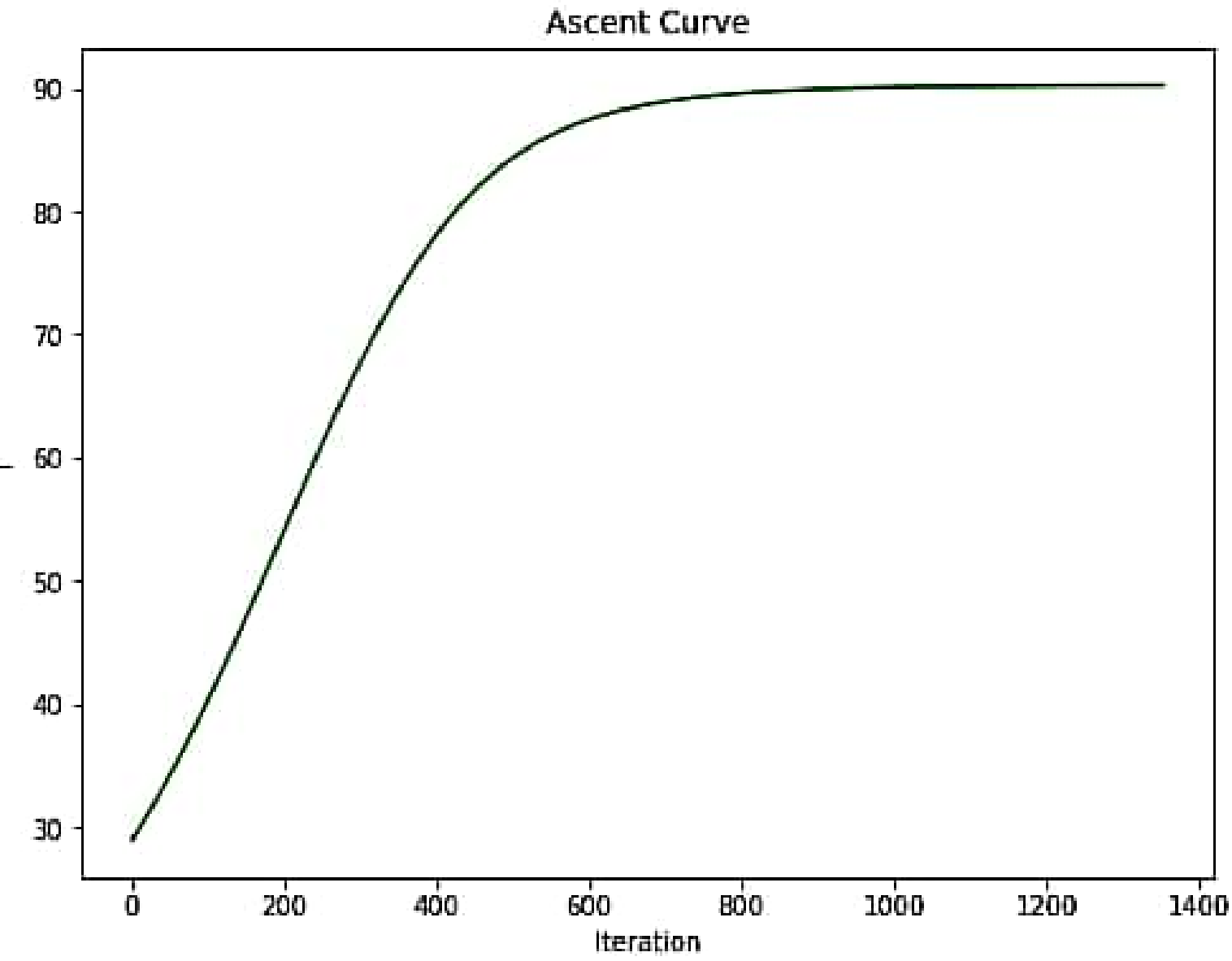
$\Rightarrow f(x) \equiv 6 - x_1^4 + 16x_1^2 - x_2^4 + 9x_2^2$

$$\therefore \frac{\partial f(x)}{\partial x} = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \end{bmatrix} = \begin{bmatrix} -4x_1^3 + 32x_1 \\ -4x_2^3 + 18x_2 \end{bmatrix}$$

where $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

Note: Initializing X to $[0\ 0]^T$ does not lead anywhere since it is a stationary point for G.D. updates. With $x_0 = [1\ 1]^T$ $\theta = 10^{-8}$, $\eta = 10^{-4}$ it takes approx. 1400 iterations to reach optima

of $f^* \cong 90.23$ at $x^* = \begin{bmatrix} 2.826 \\ 2.093 \end{bmatrix}$


Ascent Curve
90
80
70
60
50
40
30
0
200
400
600
800
1000
1200
1400
Iteration

9) The main reasons people use batch gradient ~~is~~ descent is:

→ the entire dataset is too large to fit in memory and ~~he~~ therefore use a subset of this dataset.

→ training on the entire dataset is much slower compared to using a mini batch from this dataset..

Suppose, we are training a model f to compute some kind of function value on samples x_is & we have a set of labels y_i for every x_i. (keeping the task unspecified and proceeding in a fairly generic manner)

Let $\mathcal{L}(f(x_i), y_i)$ give the loss for a single sample. If we have a total of N datapoints available for training; our total loss can be computed as follows:

$$L = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(f(x_i), y_i)$$

We use the gradient of L w.r.t w (note that w can be a vector of reals i.e $w \in \mathbb{R}^d$) to run gradient descent for finding the optimum w and in turn the best possible function f. w here is a parameter used to compute f & in that, f is computed as follows:

$$f(x) = w^T x$$

In normal gradient descent, the update rule is as follows:

$$w^{(t+1)} \leftarrow w^{(t)} - \eta \nabla_w L \quad \text{where,}$$

$$\nabla_w L = \frac{\partial L}{\partial w} \quad \text{and } \eta \text{ is the learning rate.}$$

Computing L & $\nabla_w L$ can be computationally infeasible when N is very large ($> 10^6$). Intuitively, using a smaller set of samples which can effectively capture the attributes of the loss L seems like a clever way of doing it. Hence, in every iteration, a batch of samples of size k is chosen

denoted by $B_k = \{x_1^b, x_2^b \cdots x_k^b\}$ and use batch loss given by:

$$L^B = \frac{1}{k} \sum_{i=1}^{k} \mathcal{L}(f(x_i^b), y_i^b) \text{ and}$$

$\nabla_w L^B$ instead. This reduces the computation cost drastically.

Now, the batch needs to represent the entire training dataset samples. Heavily skewed batches focussed on only one type of data will drive the learning in a different direction than what is desired. This is why in practice, we choose the samples in a batch stochastically.

Batch gradient due to it's reduced size leads to faster training if schemes like batch normalization is adopted. This induces a sort of regularization feature aiding in faster convergence.

Usually, forward pass is done on the entire batch parallely. GPUs are specially suited for this task.

Batch sizes are usually powers of 2.

Therefore, batch gradient descent has the following advantages over normal gradient descent over the entire training set:

→ Brings down the computation cost several folds.

→ Induced normalization criteria.

→ Faster convergence.

→ Parallel ~~comp~~ computation for entire batch in forward pass.

Since, we choose the batch stochastically this algorithm is ~~reffered to~~ referred to as stochastic batch gradient descent.

10) If we are using gradient descent to optimize a parameter, say w where the function to be modelled say f uses the specific value of w to compute a forward pass on a sample from the dataset; then we can ~~cos~~ consider the loss function purely as a function of the model parameters themselves, provided the

Sample collection for forward & ~~backprog~~ backpropagation passes at each possible iteration have been taken care of.

If $\mathcal{L}$ is the loss function dependent on w; $\nabla_w \mathcal{L}$ represents the gradient in $\mathcal{L}$ w.r.t w & thus, $\nabla_w \mathcal{L} = \frac{\partial}{\partial w} \mathcal{L}$. Therefore, our gradient descent update at time stamp $t+1$:

$$w^{(t+1)} \leftarrow w^{(t)} - \eta \nabla_w \mathcal{L}\left(w^{(t)}\right)$$

$$\Rightarrow w^{(t+1)} - w^{(t)} = -\eta \nabla_w \mathcal{L}$$

$$= \text{the change in weight}$$

$$= s(t)$$

Now, we can use the first order Taylor Series approximation to obtain:

$$\mathcal{L}\left(w^{(t+1)}\right) = \mathcal{L}\left(w^{(t)}\right) + s^T \nabla_w \mathcal{L}\left(w^{(t)}\right)$$

Now as $s = w^{(t+1)} - w^{(t)}$

$$= -\eta \nabla_w \mathcal{L}\left(w^{(t)}\right);$$

we have

$$\mathcal{L}(w^{(t+1)}) = \mathcal{L}(w^{(t)}) - \eta \nabla_w \mathcal{L}(w^{(t)})^T \cdot \nabla_w \mathcal{L}(w^{(t)})$$

$$\Rightarrow \mathcal{L}(w^{(t+1)}) - \mathcal{L}(w^{(t)}) = -\eta \cdot \Theta$$

where Θ is of the form $a^T a$ and is clearly non negative

∴ The change in loss with each iteration of GD is negative indicating that the loss decreases. When $\nabla_w \mathcal{L}$ is 0, there is no change in objective and the iteration stops.

The convergence follows immediately from the fact that loss $\mathcal{L}$ is bounded below and cannot decrease indefinitely.

The following assumptions were made:

i) A linear approximation for $\mathcal{L}$; ignoring higher order terms.

ii) Optimizing a function that is bounded below.

(i) may not be perfectly valid across situations. For (ii), say we are trying to optimize something unbounded such as minimizing a function; and

the learning rate η is too high, we may overshoot. Thus, we can see that η is an important metric.

If η is too small; we need too many iterations & for too large η we may diverge away from the optima. We can use the 2nd order approximation (which works well in case of losses with squared errors) and compute the optimal learning rate as:

$$\eta = \frac{\| \nabla L \|^2}{\nabla L^T H \nabla L}$$, where H is the Hessian matrix.

Note that computation of H may be difficult and expensive. In practice we try different values of η and use the one which is not too large or too small for training purposes.