Amir Hossein Eyvazkhani
1747696
Ex5

Task 1)

ⓐ In Newton's method, we use the tylor expansion (to the 2nd order) to approximate any function at a certain point. After that, we use the Second order and first order gradient to find the minimum of that function. Since we are using a quadratic approximation, our approximate tend to be more exact than other techniques like gradient descent. Moreover, this is an affine algorithm, which means we get the same results in various Coordinates.

ⓑ Newton's step : $\Delta x = -\nabla^2 f(x)^{-1} \nabla f(x)$

$f_1 = x^3 - 2x - 5$

$$\left.\begin{array}{l}\nabla f = 3x^2 - 2 \\ \nabla^2 f = 6x\end{array}\right\} \rightarrow \text{step} = \frac{-3x^2 + 2}{6x} \Rightarrow$$

iter				
0	$x=8$	step $=-3.95$	$x=-10$	step $=-0.49$
1	$x=4.05$	step $=-1.94$	$x=-10.49$	step $=5.21$
2	$x=2.11$	step $=-0.89$	$x=-5.28$	step $=2.57$
3	$x=1.22$	step $=-0.33$	$x=-2.71$	Step $=1.23$
4	$x=0.89$		$x=-1.48$	

$\nabla f = 0 \Rightarrow x = \pm 0.812 \rightarrow$ min

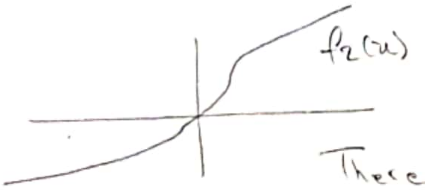
The method is getting close to the minimum here.

$f_2(x) = 3x^{1/3}$

$$\left.\begin{array}{l}\nabla f = x^{-2/3} \\ \nabla^2 f = -\frac{2}{3} x^{-5/3}\end{array}\right\} \rightarrow \text{Step} = \frac{3}{2} x$$

iter				
0	$x=-0.5$	step $=-0.75$	$x=1$	step $=1.5$
1	$x=-1.25$	Step $=-1.87$	$x=2.5$	step $=3.75$
2	$x=-3.12$	step $=-4.68$	$x=6.25$	step $=9.37$
3	$x=-7.80$	Step $=-11.7$	$x=15.62$	step $=23.43$
4	$x=-19.5$		$x=39.05$	

$\nabla f = 0 \Rightarrow \frac{1}{\sqrt[3]{x^2}} = 0$



There is no minimum. Hence newton's method would go to optimum which ① is ∞. Hence it never converges.

ⓒ The newton step can over shoot when :

i) The second order taylor expansion is not a good approximation of the function.

ii) Hessian matrix is not positive definite.

iii) near saddle points.

iv) near minimum with high curves.

v) non-convex function like f2 in part b).

Task 2)

ⓐ $\nabla_\beta L = 2\sum (x_i\beta - y_i)x_i$

$\nabla^2_\beta L = 2\sum x_i \cdot x_i > 0$

We typically don't use newton method for linear regression for ML. Since we can see that in $\nabla^2_\beta L$, there is a $\sum$, so computing the Hessian is computationaly expensive or large datasets. This is the main reason we use gradient descent (mini-batch version) instead.

ⓑ $$L_\beta = -\sum_{i=1}^{m} y_i \log(\sigma(x_i\beta)) + (1-y_i)\log(1-\sigma(x_i\beta))$$

$$= -\sum_{i=1}^{m} y_i\left(\log e^{x_i\beta} - \log(1+e^{x_i\beta})\right) + (1-y_i)\left(\log \frac{1+e^{x_i\beta}}{1+e^{x_i\beta}} - \frac{e^{x_i\beta}}{1+e^{x_i\beta}}\right)$$

$$= -\sum_{i=1}^{m} y_i x_i\beta - y_i \log(1+e^{x_i\beta}) - \log(1+e^{x_i\beta}) + y_i \log(1+e^{x_i\beta})$$

$$= -\sum_{i=1}^{m} y_i x_i \beta - \log(1+e^{x_i\beta})$$

$$\nabla_\beta L = -\sum_{i=1}^{m} y_i x_i - \frac{1}{1+e^{x_i\beta}}\left(e^{x_i\beta}\right)(x_i) = -\sum_{i=1}^{m} x_i\left(y_i - \frac{e^{x_i\beta}}{1+e^{x_i\beta}}\right)$$

$$= -\sum_{i=1}^{m} x_i\left(y_i - f_\beta(x_i)\right)$$

$$\nabla_\beta^2 L = -\sum_{i=1}^{m} -x_i\left[\frac{d}{d\beta}\,\frac{e^{x_i\beta}}{1+e^{x_i\beta}}\right] = \sum_{i=1}^{m} x_i\left[\frac{x_i e^{x_i\beta}(1+e^{x_i\beta}) - (x_i e^{x_i\beta})(e^{x_i\beta})}{(1+e^{x_i\beta})^2}\right] = \sum_{i=1}^{m} x_i\left[\frac{x_i \cdot e^{x_i\beta}}{(1+e^{x_i\beta})^2}\right]$$

The same reason as the part a) can be mentioned here. Since there is a $\sum$ in ∇^2, it's computationally expensive for large dataset. However, for small ones we can say that it is useful, since it can go to min faster than gradient descent.