Math 7243 Machine Learning - Homework 2

For programming questions, you can only use numpy library. You should not use any build in function from Scikit-learn or StatsModels libraries.

Problem 1 Loss Functions:

Let X be the data matrix and θ be the parameter vector.

a) In Lecture 2, we showed that the residual sum of square can be written

$$RSS(\theta) = (Y - X\theta)^T(Y - X\theta)$$

Find a **critical point** for $RSS(\theta)$ by calculate $\frac{\partial}{\partial\theta}RSS(\theta) = 0$.

b) **Ridge regression** changes the loss function to add in a term penalizing the θ if they get to large: For any positive number λ, the Ridge loss function

$$\text{Ridge}_\lambda(\theta) = (Y - X\theta)^T(Y - X\theta) + \lambda^2\theta^T\theta$$

Find an expression for the location of the critical point of $\text{Ridge}_\lambda(\theta)$.

Problem 2 - Computing Linear Regression:

Consider the points

$x^{(i)}$	1.2	3.2	5.1	3.5	2.6
$y^{(i)}$	7.8	1.2	6.4	2.6	8.1

a). Fit a linear function to this dataset when the loss is RSS. You may use a computer to solve the matrix equation but you should report the best fit function.

b). Fit a linear function to this dataset when the loss is the Ridge Loss from Problem 1.b) with $\lambda = 1$ and with $\lambda = 10$. What specifically explains the difference in values between the three fits.

Problem 3 - Gradient Decent and Newton's method

Consider solving the problem of locally weighted linear regression using gradient descent and Newton's method. Given data $\{\vec{x}^{(i)}, y^{(i)}\}$ for $i = 1, 2, ..., n$ and and a query point $\vec{x}$, we choose a parameter vector θ to minimize the loss function

$$J(\vec{\theta}; \vec{x}) = \sum_{i=1}^{n} w^{(i)}(\vec{\theta}^T\vec{x} - y^{(i)})^2$$

Here the weight function is $w^{(i)} = \exp\left(-\frac{\|\vec{x}^{(i)} - \vec{x}\|^2}{2\tau^2}\right)$ where τ is a hyper-parameter that must be tuned. Note that whenever we receive a new query point $\vec{x}$, we must solve the entire problem again with these new weights $w^{(i)}$.

(a) Given a data point $\vec{x}$, derive the gradient of $J(\vec{\theta}; \vec{x})$ with respect to $\vec{\theta}$.

(b) Given a data point $\vec{x}$, derive the Hessian of $J(\vec{\theta}; \vec{x})$ with respect to $\vec{\theta}$.

(c) Given a data point $\vec{x}$, write the update formula for gradient descent. Use the symbol η for an arbitrary step size.

(d) Given a data point $\vec{x}$, write the update formula for Newtons method.

Problem 4 - (Stochastic) Gradient Decent

(1) The data file $\{\vec{x}^{(i)}, y^{(i)}\}$ for $i = 1, 2, ..., n$ is drawn (with noise) from

$$f(x) = \beta_0 + \beta_1 \sin(x) + \beta_2 \cos(x)$$

Can you solve the parameters use the least squares method? Find a closed formula and explain the matrices clearly in your formula.

(2) The data file $\{\vec{x}^{(i)}, y^{(i)}\}$ for $i = 1, 2, ..., n = 10$ is drawn (with noise) from the function:

$$g(x) = \beta_0 + \sin(\beta_1 x) + \cos(\beta_2 x)$$

$x^{(i)}$	0	2	4	6	8	10	12	14	16	18
$y^{(i)}$	2.85	1.5	0.49	1.57	1.9	0.6	0.38	2.33	1.65	0.3

Use gradient decent(GD) or stochastic gradient decent (SGD) to fit the data to the function $g(x)$ by minimizing the RSS loss

$$RSS = \sum_{i=1}^{n} (y^{(i)} - g(x^{(i)}))^2$$

Turn in any associated computations, your learning rate, and the parameters.