Homework 2 CMPUT 466

Zhi Han

September 2019

I was having issues with the provided template, so I wrote the assignment on a blank template.

1 Question 1

1.1 a)

Maximize

$$p(\theta|X) \propto p(X|\theta)p(\theta) \tag{1}$$

$$\theta_{MAP} = \text{argmax} \sum_i \log p(X_i|\theta) + \log p(\theta) \tag{2}$$

We are told $p(\theta)$ is the normal distribution.

$$p(\theta) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(\theta-\mu)^2}{2\sigma^2}} \tag{3}$$

$$\log p(\theta) = \log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) + -\frac{(\theta-\mu)^2}{2\sigma^2} \tag{4}$$

$$\log p(X_i|\theta) = \sum_i \frac{-(X_i-\theta)^2}{2\sigma_0^2} \tag{5}$$

Take the derivatives

$$\frac{\partial}{\partial\theta} \log p(\theta) = \frac{\partial}{\partial\theta} \frac{-(\theta-\mu)^2}{2\sigma^2} = \frac{-(\theta-\mu)}{\sigma^2} \tag{6}$$

$$\frac{\partial}{\partial\theta} \sum_i \log p(X_i|\theta) = \frac{\partial}{\partial\theta} \sum_i \frac{-(X_i-\theta)^2}{2\sigma_0^2} = \sum_i \frac{(X_i-\theta)}{\sigma^2} \tag{7}$$

Set the derivative to zero

$$\sum_i \frac{(X_i-\theta)}{\sigma_0^2} + \frac{-(\theta-\mu)}{\sigma^2} = 0 \tag{8}$$

$$\implies \sum_i \left(\frac{X_i}{\sigma_0^2}\right) - \frac{n\theta}{\sigma_0^2} + \frac{-(\theta-\mu)}{\sigma^2} = 0 \tag{9}$$

$$\implies \sum_i \left(\frac{X_i}{\sigma_0^2}\right) - \frac{n\theta}{\sigma_0^2} = \frac{(\theta-\mu)}{\sigma^2} \tag{10}$$

$$\implies \sum_i \left(\frac{X_i}{\sigma_0^2}\right) + \frac{\mu}{\sigma^2} = \frac{\theta}{\sigma^2} + \frac{n\theta}{\sigma_0^2} \tag{11}$$

$$\implies \sum_i \left(\frac{X_i}{\sigma_0^2}\right) + \frac{\mu}{\sigma^2} = \theta\left(\frac{1}{\sigma^2} + \frac{n}{\sigma_0^2}\right) \tag{12}$$

$$\implies \boxed{\frac{\sum_i \left(\frac{X_i}{\sigma_0^2}\right) + \frac{\mu}{\sigma^2}}{\left(\frac{1}{\sigma^2} + \frac{n}{\sigma_0^2}\right)} = \theta} \tag{13}$$

1.2 b)

First note that this problem is identical to a), except that our prior is now

$$p(\theta) = \frac{1}{2b}\exp\left(\frac{-|\theta - \mu|}{b}\right) \tag{14}$$

Then,

$$\log p(\theta) = \log\left(\frac{1}{2b}\right) + \left(\frac{-|\theta - \mu|}{b}\right) \tag{15}$$

The first part is still the same:

$$\theta_{MAP} = \text{argmax} \sum_i \log p(X_i|\theta) + \log p(\theta) \tag{16}$$

$$\frac{\partial}{\partial\theta}\sum_i \log p(X_i|\theta) = \frac{\partial}{\partial\theta}\sum_i \frac{-(X_i - \theta)^2}{2\sigma_0^2} = \sum_i \frac{(X_i - \theta)}{\sigma_0^2} \tag{17}$$

But now,

$$\frac{\partial}{\partial\theta}\log p(\theta) = \frac{\partial}{\partial\theta}\left(\log\left(\frac{1}{2b}\right) + \left(\frac{-|\theta - \mu|}{b}\right)\right) \tag{18}$$

is not differentiable at zero. Now we run into a problem. The argmax operator is not linear, so we can't just optimize each of the terms. We must use an iterative approach. I would do the following. We need to maximize:

$$\text{argmax}_\theta \log\left(\frac{1}{2b}\right) + \left(\frac{-|\theta - \mu|}{b}\right) + \sum_i \frac{-(X_i - \theta)^2}{2\sigma_0^2} \tag{19}$$

But each random variable X_i has some data x_{ij}, so we just minimize w.r.t to the data, that is x_j in the domain of $p(x)$.

$$\text{argmax}_\theta \left(\frac{-|\theta - \mu|}{b}\right) + \sum_{i=1}^{d}\sum_{j=1}^{n} \frac{-(x_{ij} - \theta)^2}{2\sigma_0^2} \tag{20}$$

This is the equation I would maximize. If the data is small then I would guess trial values of theta and pick the largest, but if the data is big then I would use gradient descent. (Algorithm 4), or just call scipy.optimize.minimize.

1.3 c)

Note that $\boldsymbol{\sigma}$ is diagonal. Maximize

$$p(\boldsymbol{\theta}|\mathbf{X}) \propto p(\mathbf{X}|\boldsymbol{\theta})p(\boldsymbol{\theta}) \tag{21}$$

$$\boldsymbol{\theta}_{MAP} = \text{argmax} \sum_i \log p(\mathbf{X}_i|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) \tag{22}$$

We are told $p(\boldsymbol{\theta})$ is the normal distribution.

$$p(\boldsymbol{\theta}) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(\boldsymbol{\theta}-\boldsymbol{\mu})^2}{2\sigma^2}} \tag{23}$$

$$\log p(\boldsymbol{\theta}) = \log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) + -\frac{(\boldsymbol{\theta}-\boldsymbol{\mu})^2}{2\sigma^2} \tag{24}$$

$$\log p(\mathbf{X}_i|\boldsymbol{\theta}) = \sum_i \frac{-(\mathbf{X}_i-\boldsymbol{\theta})^2}{2\sigma_0^2} \tag{25}$$

Take the derivatives

$$\frac{\partial}{\partial\boldsymbol{\theta}} \log p(\boldsymbol{\theta}) = \frac{\partial}{\partial\boldsymbol{\theta}} \frac{-(\boldsymbol{\theta}-\boldsymbol{\mu})^2}{2\sigma^2} = \frac{-(\boldsymbol{\theta}-\boldsymbol{\mu})}{\sigma^2} \tag{26}$$

$$\frac{\partial}{\partial\boldsymbol{\theta}} \sum_i \log p(\mathbf{X}_i|\boldsymbol{\theta}) = \frac{\partial}{\partial\boldsymbol{\theta}} \sum_i \frac{-(\mathbf{X}_i-\boldsymbol{\theta})^2}{2\sigma_0^2} = \sum_i \frac{(\mathbf{X}_i-\boldsymbol{\theta})}{\sigma_0^2} \tag{27}$$

Set the derivative to zero

$$\sum_i \frac{(\mathbf{X}_i-\boldsymbol{\theta})}{\sigma_0^2} + \frac{-(\boldsymbol{\theta}-\boldsymbol{\mu})}{\sigma^2} = 0 \tag{28}$$

Note that $\boldsymbol{\sigma}_0 = 1$, and $\boldsymbol{\mu} = 0$ so

$$\implies \sum_i (\mathbf{X}_i) = \frac{\boldsymbol{\theta}}{\sigma^2} + n\boldsymbol{\theta} \tag{29}$$

$$\implies \sum_i \mathbf{X}_i = \boldsymbol{\theta}\left(\frac{1}{\sigma^2} + n\right) \tag{30}$$

$$\implies \boxed{\frac{\sum_i \mathbf{X}_i}{\left(\frac{1}{\sigma^2}+n\right)} = \boldsymbol{\theta}} \tag{31}$$

2 Question 2

2.1 a)

As I increased the number of features, I found that the average error in FSLinearRegression decreased.

The default file has [1, 2, 3, 4, 5] features selected with an error of approximately 20-22.

As I increased to about half the features, the error decreased to about 14.

Then, as I increased to the entire dataset, the error was about 11.

Hence, I conclude that **increasing the number of features decreases the error up to a maximum limit.** In our case the limit is about 11 or so.

This behaviour is to be expected since our model is not a very complicated model, it is linear regression. If the true function we are trying to approximate is not linear, then our model is underfitting the data and the only solution is to choose a more complicated model.

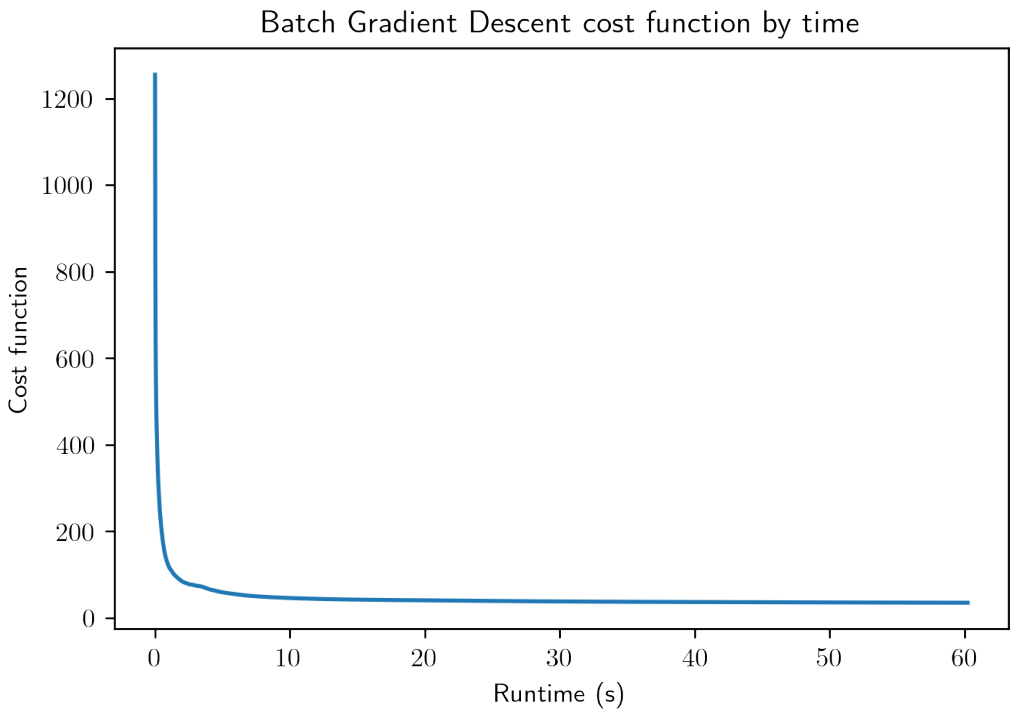
2.2 b)

The standard error of the mean is given by

$$\sigma_\mu = \frac{\sigma}{\sqrt{n}} \tag{32}$$

This is implemented as:

```
sample_error = np.std(errors['FSLinearRegression'], axis=1)/np.sqrt(numruns)
sample_error
```



2.3 c) Ridge Regression

Ridge Regression is implemented.

The difference between a) this is that if we select all the features in part a), then $X^T X$ is not invertible. Part a) also is overfitting and hence the error might be higher due to having no overfitting.

2.4 d) Lasso Regression

Lasso is implemented. The average error using all training data, and $\lambda = 0.01$:

$$\text{Average Error over 5 runs} = 8.710357609801331 \pm 0.05402196105293677 \tag{33}$$

2.5 e) Stochastic Gradient Descent

Stochastic Gradient Descent is implemented. The average error using all training data, stepsize $\eta = 0.01$ and 1000 epochs:

$$\text{Average Error over 5 runs} = 8.607170557855877 \pm 0.07382256706412164 \tag{34}$$

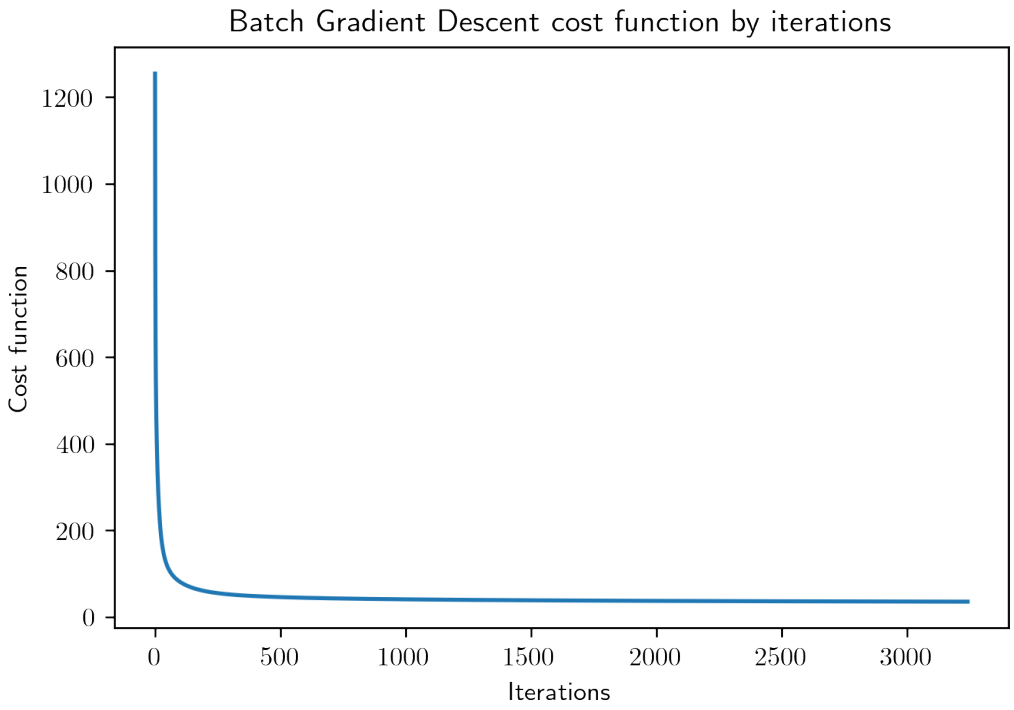
Although this is somewhat bad performance, I suspect that this is due to the stepsize. Perhaps if the stepsize is set lower, the algorithm will do better.

2.6 f) Batch Gradient Descent

Batch Gradient Descent is implemented. Using a stepsize of $\eta = 0.01$.

$$\text{Average Error over 5 runs} = 8.788215455749896 \pm 0.09061808451385017 \tag{35}$$

Batch Gradient descent processed 3243 iterations over the training set, compared to 1000 iterations by stochastic gradient descent but performed significantly better than stochastic gradient descent.


Batch Gradient Descent cost function by iterations
Cost function
1200
1000
800
600
400
200
0
0
500
1000
1500
2000
2500
3000
Iterations