مهسا امینی به شماره دانشجویی ۹۸۱۷۸۲۳

سوال یک

الف)

خط آبی مربوط به روش Batch GD خط سبز مربوط به SGD وخط قرمز برای روش Mini-batch GD است. در روش SGD محاسبات ما نویزی میشوند چون مدل را N بار تکرار و اپتیمایز میکنیم و در هر بار با حالت شک و تردید به جلو می رویم و نسبت به اپتیمایز خودمان اطمینانی نداریم و ممکن است خیلی از نمونه های ما داده ی پرت حساب شوند و ما مدلمان را روی همین داده های پرت بروزرسانی میکنیم پس به صورت نویزی به جلو میرویم اما در روش batch GD روش و رویکرد بر مبنای بررسی تمامی خطاها در همه ی داده ها است و زمانی مدل بروزرسانی و اپتیمایز میشود که بر روی تمامی داده های train انجام شود. و ما فقط داده ها را یکبار اما به محاسبات سنگین اپتیمایز کرده ایم پس نویز کمی داریم. اگر بخواهیم یک روش متعادل تر داشته باشیم به سمت Mini-batch GD میرویم که در این روش هر دفعه یک batch از داده ها را آموزش میدهیم که نویزی شدن مسیر نسبت به GD خیلی کمتر میشود.

ب)

در شکل a ما داده ها را scale نکرده ایم اما در شکل b داده ها را scale کرده ایم. وقتی داده ها را scale میکنیم جهت گرادیان عمود میشود که سریع تر و ساده تر است. در شکل a حالت زیکزاکی داریم چون هر دفعه ممکن است مینیموم را رد کنیم و باید برگردیم که این رد کردن و برگشتن باعث وجود اعوجاج میشود.

اگر مسئله ای داشته باشیم که چندین متغیر یا ویژگی دارد و متغیر ها در مقیاس مشابهی نسبت بهم باشند در این حالت گرادیان کاهشی با سرعت بیشتری به همگرایی میرسد و اگر در مقیاس مشابهی نباشند نمودار کانتور شکل بلند و بیضی خواهد داشت که گرادیان کاهشی برای پیدا کردن مینیموم باید زمان زیادی صرف کند.

سوال دو

الف) اگر داده های پرت زیاد باشند روش MSE چون توان دو وجود دارد، خطا را خیلی زیاد میکند . و به عبارتی MSE خطاهای بزرگ را به میزان بیشتری نسبت به MAE مجازات میکند. و قدر مطلق نسبت به داده های پرت مقاوم تر است و MSE در مواجهه با نقاط outlier در داده های ورودی عملکرد خوبی ندارد . با این حال اگر می خواهید مدلی را آموزش دهید که بر کاهش خطاهای پرت بزرگ تمرکز دارد، MSE انتخاب بهتری است، در حالی که اگر این مهم نیست و تفسیر پذیری بیشتری را ترجیح می دهید، MAE بهتر است.

ب) چون در این نقطه MAE مشتق پذیر نیست از MSE استفاده میکنیم.

ج) زمانی که ویژگی های داده ها در مقیاس های مختلف است و نرمالسازی انجام نشده است بهتر است از MAE استفاده شوند چرا که این توان دو رساندن باعث میشود scale داده بدتر شود مثلا اگر داشته باشیم:

$0.1<x1<0.2$

$10<x2<100$

این داده ها پس این که در فورمول توان دویی برای آن ها محاسبه شود فاصله به شکل خیلی زیادی افزایش میابد.

سوال سه

عبارت سوم برای ستون C میباشد که این عبارت به سمت صفر کردن میرود و همان طور که می بینیم در ستون C تعداد صفر ها و اعداد نزدیک به صفر زیاد تر است که همان روش lasso می باشد و یک ویژگی جالب خطای رگرسیون lasso این است که می تواند ضرایب یک مدل را به صفر برساند و به طور موثر تعداد ویژگی های مدل را کاهش می دهد به خاطر همین گاهی از آن به

عنوان feature selection نیز استفاده میشود. عبارت دوم مربوط به ستون B است که این عبارت همان ridge است که بر خلاف lasso به سمت صفر کردن نمی رود و فقط به سمت کوچک کردن می رود و لازم نیست حتما به صفر خیلی خیلی نزدیک باشد چرا که به توان دو میرسد. و ستون A هم مربوط به عبارت اول است. که

سوال چهار

$$0 \le x_i \le 1$$

$$p(D \mid \theta) = \prod_{i=1}^{N} p(x_i \mid \theta)$$

$$p(D \mid \theta) = \prod_{i=1}^{N} \sqrt{\theta} x_i^{\sqrt{\theta}-1}$$

$$\ln(p(D \mid \theta)) = \sum_{i=1}^{N} \ln p(x_i \mid \theta)$$

$$\ln p(D \mid \theta) = \sum_{i=1}^{N} \ln \sqrt{\theta} x_i^{\sqrt{\theta}-1}$$

$$N \ln \sqrt{\theta} + \sum_{i=1}^{N} (\sqrt{\theta} - 1) \ln(x_i)$$

حال نسبت به تتا مشتق جزئی میگیریم و مقدار آن را برابر صفر میگذاریم:

$$\frac{N}{2\theta} + \frac{1}{2\sqrt{\theta}} \sum_{i=1}^{N} \ln(x_i) = 0$$

$$\frac{1}{2\sqrt{\theta}} \sum_{i=1}^{N} \ln(x_i) = -\frac{N}{2\theta}$$

$$\sqrt{\theta} = u$$

$$\frac{N}{2u^2} + \frac{1}{2u} \sum_{i=1}^{N} \ln(x_i) = 0$$

$$u = -\frac{N}{\sum_{i=1}^{N} \ln(x_i)}$$

$$\theta = \frac{N^2}{\sum_{i=1}^{N} (\ln(x_i))^2}$$

سوال پنج

$$\mu_{MAP} = \arg\max\ p(\mu \mid D) = \arg\max\ p(D \mid \mu)\, p(\mu)$$
$$\mu_{MAP} = \arg\max(\log(\, p(D \mid \mu)) + \log\ p(\mu))$$

برای محاسبه ی تتا از عبارت مستق جزئی بر حسب تتا میگیریم و آن را برابر صفر قرار میدهیم:

$$0 = \frac{\partial}{\partial \mu}(\log(\, p(D \mid \mu)) + \log\ p(\mu))$$

ابتدا مشتق جزئی $\log\ p(\mu)$ را حساب می کنیم:

$$\frac{\partial}{\partial \mu}\log(\, p(\mu)) = \frac{\partial}{\partial \mu}\log(\frac{1}{\sqrt{2\pi}\sigma_\mu} e^{-\frac{(\mu-\mu_0)^2}{2\sigma_\mu}})$$
$$= \frac{\partial}{\partial \mu}(\log\frac{1}{\sqrt{2\pi}\sigma_\mu} - \frac{(\mu-\mu_0)^2}{2\sigma_\mu})$$
$$= -\frac{(\mu-\mu_0)}{\sigma_\mu} = \frac{\mu_0-\mu}{\sigma_\mu}$$

حال ابتدا $p(D \mid \mu)$ را حساب میکنیم.

$$\arg\max\ p(D \mid \mu)$$
$$p(x \mid \mu) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(X-\mu)^2}{2\sigma^2}}$$
$$p(D \mid \mu) = \prod_{i=1}^{N} p(x_i \mid \mu) = (\frac{1}{\sqrt{2\pi\sigma^2}})^n e^{-\frac{\sum_{i=1}^{N}(x_i-\mu)^2}{2\sigma^2}}$$
$$\log(\, p(D \mid \mu)) = -\frac{N}{2}\log(2\pi\sigma^2) - \frac{\sum_{i=1}^{N}(x_i-\mu)^2}{2\sigma^2}$$

حال از آن مشتق جزئی میگیریم :

$$\frac{\partial}{\partial \mu}(\log(\, p(D \mid \mu))) = \frac{\sum_{i=1}^{N} x_i - N\mu}{\sigma^2}$$

مقدار مشتق بدست آمده را با مشتق $\log(p(\mu))$ که در قبل حساب کردیم جمع میزنیم و مقدار آن را برابر صفر میگذاریم:

$$\frac{\sum_{i=1}^{N} x_i - N\mu}{\sigma^2} + \frac{\mu_0 - \mu}{\sigma_\mu} = 0$$

مقدار مورد نظر را از عبارت بالا حساب میکنیم که پس از ساده سازی برابر است با:

$$\mu = \frac{\sum x_i + \sigma^2 \frac{\mu_0}{\sigma_\mu}}{N + \sigma^2} \sigma_\mu$$

سوال شش

1

```
In [21]: x = 35.38
         y = 2955.53
         a=0
         b=0
         c=0
         y_pred = (0*x*x)+(0*x)+0
         a_g = (x**2)*(y_pred-y)
         b_g =(x)*(y_pred-y)
         c_g =(y_pred-y)

In [22]: alfa =0.1
         a= a-a_g*alfa
         b=b-b_g*alfa
         c=c-c_g*alfa

In [23]: a
Out[23]: 369956.8126532001

In [24]: b
Out[24]: 10456.665140000003

In [25]: c
Out[25]: 295.55300000000005
```

2

In [26]:
```
x = 0.29
y = 2.28
y_pred = (a*x*x)+(b*x)+c
a_g = (x**2)*(y_pred-y)
b_g =(x)*(y_pred-y)
c_g =(y_pred-y)
```

In [27]:
```
alfa =0.1
a= a-a_g*alfa
b=b-b_g*alfa
c=c-c_g*alfa
```

In [28]:
```
a
```

Out[28]: 369667.18004225

In [29]:
```
b
```

Out[29]: 9457.931998792714

In [30]:
```
c
```

Out[30]: -3148.354383473413

3

In [31]:
```
x = 11.74
y = 334.32
y_pred = (a*x*x)+(b*x)+c
a_g = (x**2)*(y_pred-y)
b_g =(x)*(y_pred-y)
c_g =(y_pred-y)
```

In [32]:
```
alfa =0.1
a= a-a_g*alfa
b=b-b_g*alfa
c=c-c_g*alfa
```

In [33]:
```
a
```

Out[33]: -703349027.3966404

In [34]:
```
b
```

Out[34]: -59932509.23807639

In [35]:
```
c
```

Out[35]: -5108937.721510831

4

```
In [36]: x = 39.45
         y = 3670.48
         y_pred = (a*x*x)+(b*x)+c
         a_g = (x**2)*(y_pred-y)
         b_g =(x)*(y_pred-y)
         c_g =(y_pred-y)

In [37]: alfa =0.1
         a= a-a_g*alfa
         b=b-b_g*alfa
         c=c-c_g*alfa

In [38]: a
Out[38]: 170724638148354.97

In [39]: b
Out[39]: 4327578635231.759

In [40]: c
Out[40]: 109694221043.03886
```

5

```
In [41]: x = 26.85
         y = 1709.09
         y_pred = (a*x*x)+(b*x)+c
         a_g = (x**2)*(y_pred-y)
         b_g =(x)*(y_pred-y)
         c_g =(y_pred-y)

In [42]: alfa =0.1
         a= a-a_g*alfa
         b=b-b_g*alfa
         c=c-c_g*alfa

In [43]: a
Out[43]: -8.881272808831941e+18

In [44]: b
Out[44]: -3.3077569228989696e+17

In [45]: c
Out[45]: -1.231944411838723e+16
```

6

```
In [46]: x = 3.98
         y = 13.08
         y_pred = (a*x*x)+(b*x)+c
         a_g = (x**2)*(y_pred-y)
         b_g =(x)*(y_pred-y)
         c_g =(y_pred-y)

In [47]: alfa =0.1
         a= a-a_g*alfa
         b=b-b_g*alfa
         c=c-c_g*alfa

In [48]: a

Out[48]: 2.160709729327067e+20

In [49]: b

Out[49]: 5.618988906689066e+19

In [50]: c

Out[50]: 1.418885260592698e+19
```

7

```
In [51]: x = 19.05
         y = 864.44
         y_pred = (a*x*x)+(b*x)+c
         a_g = (x**2)*(y_pred-y)
         b_g =(x)*(y_pred-y)
         c_g =(y_pred-y)

In [52]: alfa =0.1
         a= a-a_g*alfa
         b=b-b_g*alfa
         c=c-c_g*alfa

In [53]: a

Out[53]: -2.884760910861468e+24

In [54]: b

Out[54]: -1.513861713620828e+23

In [55]: c

Out[55]: -7.935541396798255e+21
```

8

```
In [56]: x = 15.32
         y = 560.30
         y_pred = (a*x*x)+(b*x)+c
         a_g = (x**2)*(y_pred-y)
         b_g =(x)*(y_pred-y)
         c_g =(y_pred-y)

In [57]: alfa =0.1
         a= a-a_g*alfa
         b=b-b_g*alfa
         c=c-c_g*alfa

In [58]: a
Out[58]: 1.5942502468620654e+28

In [59]: b
Out[59]: 1.040670234555238e+27

In [60]: c
Out[60]: 6.7930812547806874e+25
```

9

```
In [61]: x = 30.51
         y = 2202.93
         y_pred = (a*x*x)+(b*x)+c
         a_g = (x**2)*(y_pred-y)
         b_g =(x)*(y_pred-y)
         c_g =(y_pred-y)

In [62]: alfa =0.1
         a= a-a_g*alfa
         b=b-b_g*alfa
         c=c-c_g*alfa

In [63]: a
Out[63]: -1.3843646178581084e+33

In [64]: b
Out[64]: -4.537360896465816e+31

In [65]: c
Out[65]: -1.487137891373382e+30
```