عنوان : گزارش تمرین اول یادگیری ماشین

نگارنده : سحر داستانی اوغانی

شماره دانشجویی : ۹۹۱۱۲۱۰۸

دانشکده ریاضی و علوم کامپیوتر

۲-الف

این سوال هم به صورت کد حل شده و هم به صورت دستی. همانطور که در کد مشاهده می‌کنید، داده‌ها تولید شدند و مدل با تابع هزینه ذکر شده رسم شده است.

۳

ابتدا داده‌ها توسط یک نویز گوسی با مرکز ۰ و میانگین ۲ تولید شدند و در فایلی با نام data ذخیره شدند. سپس برای جلوگیری از تفاوت موجود در رنج داده‌ها (scale) و عملکرد بهتر الگوریتم و نزدیک بودن آن به normal distribution، داده‌ها توسط feature scaling و با متد MinMaxScaler، نرمال شدند. برای درک تفاوت بهتر، داده‌های نرمال شده(عکس سمت راست) و داده‌های نرمال نشده(عکس سمت چپ) در این‌جا آمده است. همانطور که مشاهده می‌کنید، scale داده‌ها بعد از نرمال شدن به بالای صفر انتقال پیدا کرده است.

	Unnamed: 0	X	T
0	0	-0.936773	-2.973252
1	1	1.500040	-2.181905
2	2	36.863630	45985.635813
3	3	74.666853	399482.033794
4	4	-60.206096	-229047.205798
...	...	...	...
95	95	-95.046266	-885629.571085
96	96	-3.566159	-76.751444
97	97	-75.928006	-454948.248417
98	98	-41.495118	-76568.334806
99	99	44.262628	80802.423524

	Unnamed: 0	X	T
0	0.000000	0.493907	0.511113
1	0.010101	0.506233	0.511113
2	0.020202	0.685114	0.534927
3	0.030303	0.876335	0.717978
4	0.040404	0.194104	0.392507
...	...	...	...
95	0.959596	0.017871	0.052509
96	0.969697	0.480607	0.511075
97	0.979798	0.114577	0.275529
98	0.989899	0.288750	0.471465
99	1.000000	0.722540	0.552956

حال برای درج مدلی روی داده‌ها نیاز داریم آن‌ها را به دو گروه آموزش و آزمون تقسیم کنیم. برای این کار ۷۰ درصد داده‌ها را با عنوان داده‌ی آزمون و ۳۰٪ آن‌ها را با عنوان داده‌ی آموزشی برچسب زدیم. شکل مقابل، فرمت x,y داده‌های آموزش و x,y داده‌های آزمون را نمایش می‌دهد. که مشخص می‌کند از میان ۱۰۰ داده‌ی تولید شده در بالا، ۳۰ تا از آن‌ها برای آموزش مدل و ۷۰ تای دیگر برای تست به کار رفته است.

```
((30,), (30,), (70,), (70,))
```

برای حل ادامه‌ی سوال‌ها نیاز است ۲ کار را فراهم کنیم:

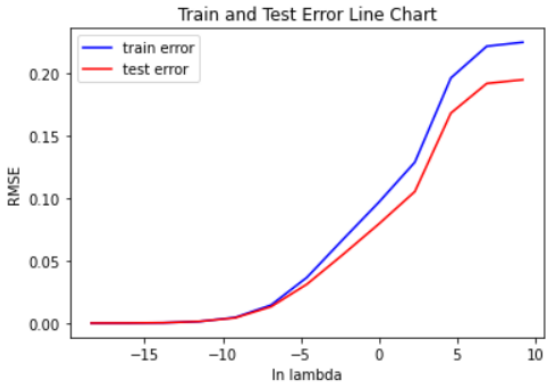
- همانطور که در صورت سوال نیز از ما خواسته شده، باید تابعی با نام fi2 را بر روی متغیر x داده‌های خود اثر دهیم و سپس ادامه‌ی الگوریتم را بر روی داده‌های transformed شده پیاده کنیم. این کار نیز توسط PolynomialFeatures انجام گرفته است.
- باید رنج آلفا را نیز از 10^-8 تا 10^4 فراهم کنیم.

۳-الف-۱

برای مدل کردن داده‌ها در این مرحله از ماژول Ridge استفاده شده است. زیرا cost function در این ماژول SSE به همراه regularization term می‌باشد. بردار w به ازای مقادیر مختلف لاندا(الفا) بدست آمده است. برای مثال، بردار w برای الفا با مقدار 1e-05 برابر است با:

```
Alpha:  1e-05
RMSE:  None
W=  [ 0.          3.03404134 -5.16964706  1.55188358  1.876638    0.49052703
 -0.38543553 -0.51434792 -0.23379231  0.09973821  0.24770773]
```

از طرفی نمودار خطای RMSE برای لانداهای مختلف(که در اینجا بر حسب ln lambda رسم شده است) به شرح مقابل است.



این نمودار برای لانداهای خیلی کوچک که به سمت صفر می‌روند، فاقد جمله‌ی منظم ساز است. در جاهایی از نمودار که فاصله‌ی میان خطای آموزش و آزمون یک حد منطقی‌ای است، overfitting رخ داده است. و در کل میزان خطای آموزش بیشتر از میزان خطای آزمون برای داده‌های تولید شده در بالا توسط نویز گوسی است.

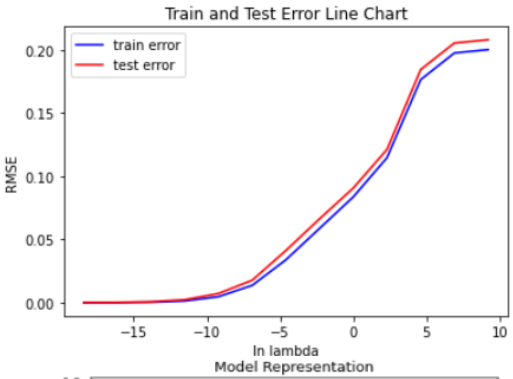
محاسبه RMSE را با اجرای ۱۰۰ بار الگوریتم بالا بر روی داده‌ها نیز انجام دادیم. مینیمم RMSE بر روی داده‌های تست زمانی رخ داده است که لاندا 1e-5 باشد.

	lambda	rmse_test	rmse_train
0	1.000000e-08	0.000058	0.000014
1	1.000000e-07	0.000203	0.000074
2	1.000000e-06	0.000715	0.000304
3	1.000000e-05	0.002310	0.001307
4	1.000000e-04	0.007228	0.004711
5	1.000000e-03	0.017732	0.013658
6	1.000000e-02	0.041244	0.033865
7	1.000000e-01	0.066314	0.058735
8	1.000000e+00	0.090635	0.083623
9	1.000000e+01	0.121158	0.114598
10	1.000000e+02	0.184332	0.176354
11	1.000000e+03	0.205326	0.197466
12	1.000000e+04	0.207899	0.200071

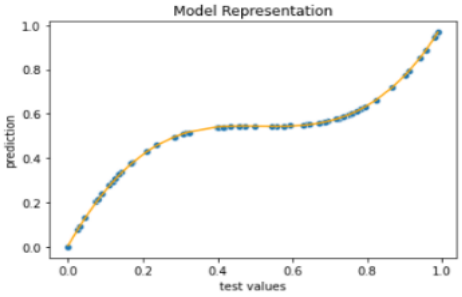
w نیز برای این مقدار به شرح زیر است:

```
Alpha:  1e-05
RMSE_test:  0.0010865639255599874
W=  [ 0.          3.05933674 -5.29306033  1.73477034  1.86017751  0.40639101
 -0.42113947 -0.49719509 -0.20617355  0.10820386  0.24759529]
```

نمودار خطای آموزش و آزمون برای ۱۰۰ بار اجرای الگوریتم به نمودار زیر تغییر پیدا کرد:



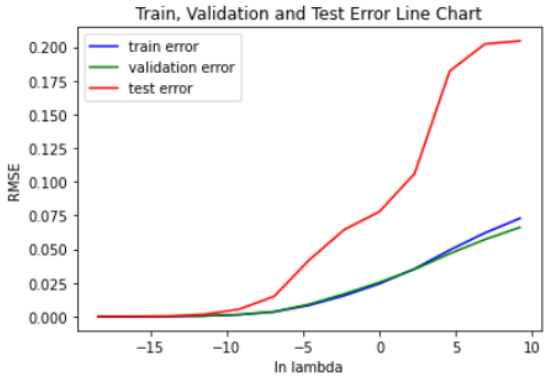
همانطور که مشاهده می‌کنید برای ۱۰۰ بار اجرای این الگوریتم، میزان خطای آزمون از خطای آموزش بیشتر شده و دو نمودار نزدیکی بیشتری به هم پیدا کردند. همین امر از overfitting جلوگیری کرده است. و میزان خطای کل نمودار کاهش پیدا کرده است. (از ۲۵ به ۲۰ به صورت تقریبی)



برای درک بهتر وضعیت مدل در بهترین لاندا (یعنی 1e-05)، به نمودار روبرو توجه کنید،مدل به خوبی بر روی داده‌ها fit شده است.

۳-الف-۲

با استفاده از ماژول Kfold مدل را تغییر داده و مدل Kfold را بر روی داده‌ها اعمال میکنیم. لازم به ذکر است عملیات قبل، از جمله‌ی تولید لاندا، اعمال تابع fi2 بر روی داده‌ها را نیز انجام می‌دهیم. سپس داده‌ها را به ۳ گروه test, train, validation تقسیم کرده و مدل Ridge را روی آن‌ها اعمال می‌کنیم. خطاهای validation,test,train را بدست آورده و آن‌ها را رسم می‌کنیم. در شکل مشخص است که با افزایش مقدار لاندا، خطای validation و train دارای افزایش می‌باشند ولی خطای تست دارای افزایش قابل تاملی است. خطای validation, train در حالت مجانبی قرار دارند. این بدان معنی است که چقدر مدلی که در نظر گرفتیم ساده‌تر از مدلی بوده است که دنبالش بودیم. در لانداهای پایین، مقدار خطاها صفر شده است، یعنی در این بازه لاندا می‌توانیم تمام پارامترهای آزاد را دقیقا مشخص کنیم.

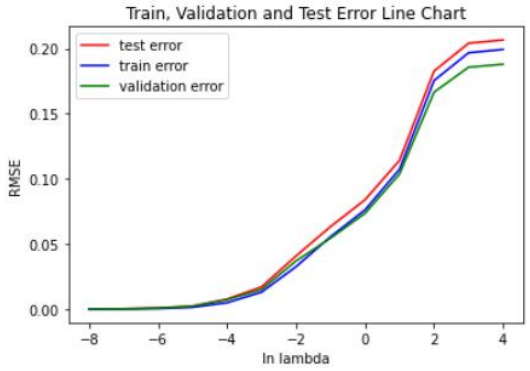


این الگوریتم را ۱۰۰ بار بر روی داده‌ها انجام دادیم و نتیجه مطابق زیر حاصل شد:

همانطور که مشخص است، این الگوریتم زمانی که مقدار لاندا برابر با 1e-08 بوده، بهترین خطا را بر روی داده‌های تست بدست آورده است.

	lambda	rmse_train	rmse_validation	rmse_test
0	1.000000e-08	0.000016	0.000044	0.000111
1	1.000000e-07	0.000078	0.000163	0.000253
2	1.000000e-06	0.000344	0.000648	0.001340
3	1.000000e-05	0.001363	0.001807	0.002616
4	1.000000e-04	0.004909	0.007151	0.010468
5	1.000000e-03	0.013166	0.015744	0.019924
6	1.000000e-02	0.034980	0.041339	0.048257
7	1.000000e-01	0.059182	0.059930	0.068661
8	1.000000e+00	0.081104	0.076624	0.089389
9	1.000000e+01	0.112333	0.103359	0.120326
10	1.000000e+02	0.177093	0.159224	0.183004
11	1.000000e+03	0.198889	0.178015	0.203160
12	1.000000e+04	0.201575	0.180314	0.205622

نمودار ۳ خطا نیز در شکل زیر آمده است: این نمودار بیانگر این است که با افزایش میزان داده‌ها میزان خطاها به هم نزدیک‌تر شدند و با افزایش لاندا میزان خطای train, validation از مقدار 0.050 به مقدار 0.20 افزایش یافته است.



میزان میانگین خطا بر روی داده‌های validation برای بهترین لاندا نیز به شرح زیر است:

```
Alpha:  1e-08
Mean_rmse_validation [0.2829484775408559, 0.22399044834884158, 0.21698283568734
605, 0.4298116498840433, 0.20962467955858827, 0.11736781583922078, 0.1168456074
9528422, 0.14216345957849008, 0.15523087162316498, 0.24340125066877513]
```

۳-ب

۵۰ مجموعه داده با اندازه‌های یکسان و مقادیر متفاوت در مرحله‌ی اول، توسط یک تابع درست شده است. ابتدا الگوریتم این قسمت را برای ۲۰ داده و سپس برای ۲۰۰ داده آزمودیم.

در هر قسمت، لاندا را تعریف کرده و تابع fi2 را بر روی داده‌ها اثر داده‌ایم و مقادیر w را برای لانداهای مختلف حساب کردیم.

برای هر کدام از مجموعه‌های تولید شده در بالا و برای هر لاندا مقدار بایاس و واریانس را با تابع bias_variance_decomp تولید کرده و میانگین خطای آزمون را به ازای مقادیر مختلف لاندا رسم کرده‌ایم.

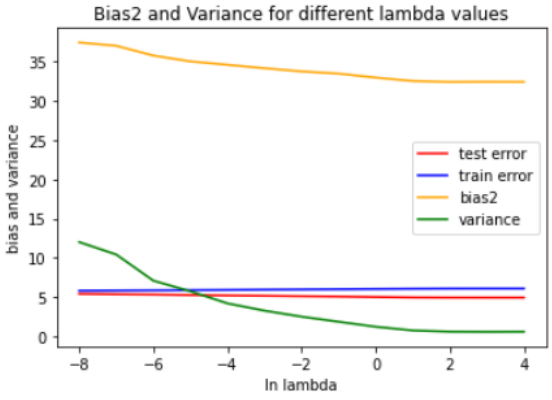
در ادامه خلاصه‌ای از مقایسه دو خروجی(برای ۲۰ داده و ۲۰۰) را بررسی می‌کنیم:

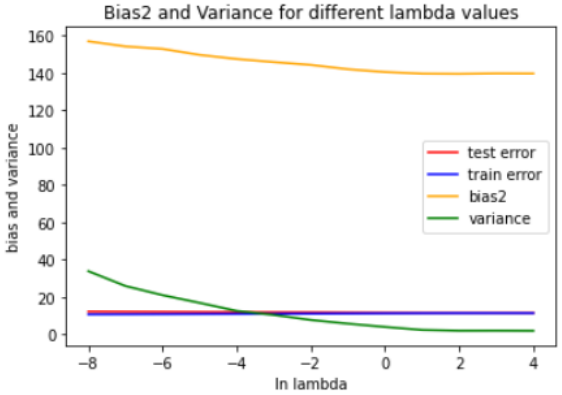
n = 20	lambda	bias2	variance	rmse_test	rmse_train
0	1.000000e-08	37.444702	11.993828	5.393186	5.795533
1	1.000000e-07	37.023856	10.404760	5.337158	5.818018
2	1.000000e-06	35.776903	7.070255	5.302005	5.844100
3	1.000000e-05	35.028639	5.718313	5.240364	5.864766
4	1.000000e-04	34.599006	4.176251	5.186493	5.888916
5	1.000000e-03	34.163871	3.246315	5.143551	5.925189
6	1.000000e-02	33.754299	2.482615	5.092806	5.954407
7	1.000000e-01	33.472050	1.833321	5.049472	5.983139
8	1.000000e+00	32.967040	1.196933	4.986316	6.013483
9	1.000000e+01	32.535975	0.723253	4.932759	6.042040
10	1.000000e+02	32.410975	0.569986	4.915896	6.062902
11	1.000000e+03	32.432401	0.551593	4.914625	6.067633
12	1.000000e+04	32.418708	0.564835	4.914523	6.068167

n = 200	lambda	bias2	variance	rmse_test	rmse_train
0	1.000000e-08	157.040252	33.820659	12.171475	10.689674
1	1.000000e-07	154.304569	25.880113	12.081493	10.737805
2	1.000000e-06	152.973850	20.975604	12.010657	10.796246
3	1.000000e-05	149.759198	16.838562	11.924933	10.864589
4	1.000000e-04	147.563911	12.525667	11.853293	10.927325
5	1.000000e-03	145.912928	10.370205	11.805111	10.984993
6	1.000000e-02	144.395791	7.669827	11.759713	11.043435
7	1.000000e-01	142.077015	5.679347	11.674375	11.129422
8	1.000000e+00	140.571847	3.925590	11.613391	11.224739
9	1.000000e+01	139.719213	2.323354	11.579959	11.300226
10	1.000000e+02	139.606307	1.917477	11.573475	11.344972
11	1.000000e+03	139.831634	1.944614	11.574397	11.354718
12	1.000000e+04	139.757604	1.872290	11.574540	11.355813

همانطور که مشخص است، مقادیر بایاس، واریانس، خطای آزمون و خطای آموزش با افزایش داده، افزایش پیدا کرده است. این بدان معناست که با ثابت نگه داشتن پیچیدگی مدل و افزایش داده‌ها می‌توان از overfitting جلوگیری کرد.

نمودارهای بایاس و واریانس :





نمودار سمت راست، نتایج بایاس و واریانس را بر روی ۲۰۰ داده و نمودار سمت چپ، بر روی ۲۰ داده نشان می‌دهد. با افزایش مقدار داده، میزان واریانس و بایاس نیز افزایش یافته است. ولی فاصله میان آن دو و فاصله‌ی آن‌ها از نمودار خطای آزمون ثابت مانده است.

در انتها نیز گزارشی از مقدار w به ازای لاندای اول یعنی 1e-08 آمده است:

```
 Alpha:  1e-08
 RMSE_test:  1.4954681076193808
 W=  [    0.          -174.86711582  1423.69011841 -4245.78107696
   2882.08750489  5789.7287718  -3845.98168043 -8921.3021034
   1186.25511    13427.64851686 -7526.49673701]
 --------------------------------------------------------------------------------
Alpha:  1e-08
RMSE_test:  6.116031831721433
W=  [     0.             82.72103301  -1213.73684948   4501.31190234
  -2713.36745896  -9556.66623061   5776.77310387  15497.06424198
  -1796.45213017 -25039.46842805  14453.96000055]
```

پاسخ تمارین تشریحی

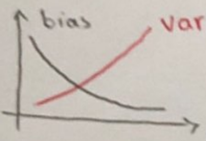
① مفهومی

الف) خطای Validation همواره کمتر از خطای آزمون است.

نادرست است. داده‌های مسئله تنها با داده‌های آموزش، آموزش می‌بینند و توسط داده‌های val و test مورد آزمون قرار می‌گیرند (البته می‌توان گفت در بخش Model-Selection، همه‌ی هایپرپارامترها را با val تعلیم می‌دهیم پس به نوعی با آن نیز آموزش می‌یابد). در error های test و val برای اینکه بتوانیم مقایسه‌ای صورت دهیم داده‌ها باید در یک order قرار گیرند. بنابراین نمی‌توان رابطه‌ای میان آن دو بیان کرد و تنها می‌توان گفت بالاتر از train error قرار دارند یا مساوی.

ب) افزایش پیچیدگی مدل در رگرسیون همیشه سبب افزایش واریانس و کاهش بایاس می‌شود.

اگر برای این جمله این شرط را بگذاریم که همه چیز را ثابت در نظر بگیریم و تنها پیچیدگی را تغییر دهیم، این جمله صحیح خواهد بود و درست می‌باشد.



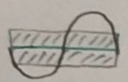
ولی اگر بخواهیم Regularization Term ای به آن اضافه کنیم، این کار می‌تواند پیچیدگی سبب زیاد را بشود و مجموعه‌ی خط‌ها را محدود کند، بنابراین var کم می‌شود و حالت استثنایی چنین مثال نیز شکل می‌گیرد. پس نادرست می‌شود:

$H_0 : f(x) = b$ مدل صفر

$H_1 : f(x) = ax + b$ مدل اول

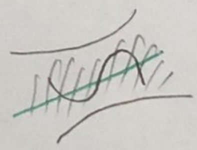
بدون Regularization term ←

H_0 :



bias = 0.5
var = 0.25

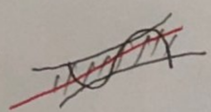
H_1 :



bias = 0.21
var = 1.69

با Regularization term ←

H_1 :

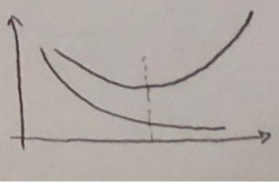


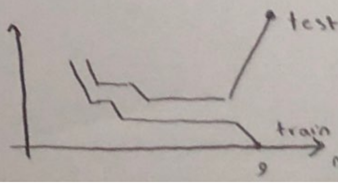
bias = 0.23
var = 0.33 → کم شده.

H_0 → bias + var = 0.75 H_1 → bias + var = 1.90 H1 + Regularization term → (0.56)

پ) افزایش پیچیدگی مدل در رگرسیون همیشه سبب کاهش خطای آموزش و افزایش خطای آزمون می‌شود.

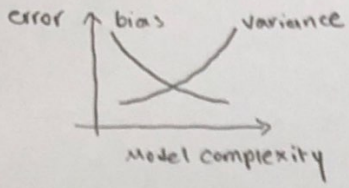
وقتی پیچیدگی مدل افزایش می‌یابد، مدل روی داده‌های train تطبیق بیشتری پیدا می‌کند. با افزایش پیچیدگی و ثابت ماندن تعداد داده‌ها، overfit رخ می‌دهد، خطا روی داده‌های train صفر شده ولی روی داده‌های test افزایش می‌یابد.





ت) در صورتی که مقدار بایاس بالا باشد و واریانس کم باشد، باید پیچیدگی مدل را کاهش داد.

نادرست است. در حالتی که مقدار بایاس بالا باشد ولی واریانس کم، مدل دچار underfitting شده و این یعنی به‌خوبی نتوانسته داده‌ها را تخمین کند. بنابراین حتماً مقدار مقابل باید پیچیدگی مدل را افزایش داد تا مقدار error کم شود و fit بهتری حاصل شود.



ث) کاهش خطای آموزش منجر به کاهش خطای آزمون می‌شود. نادرست است.

وقتی خطای آموزش کاهش می‌یابد، مدل fit بهتری بین داده‌ها می‌دهد و از طرفی خطای آزمون کاهش می‌یابد ولی این به شرطی است که خطای آموزش صفر نشود زیرا در آن صورت overfit رخ داده و خطای آزمون به صورت ناگهانی افزایش می‌یابد.

ج) در صورتی که واریانس بالا باشد، افزایش داده‌های آموزش همیشه می‌تواند سبب کاهش واریانس شود.

درست. واریانس بالا سبب overfit می‌شود، به فرض ثابت بودن پیچیدگی مدل با افزایش داده‌های آموزش می‌توان مدلی ساخت که خطای آموزش آن بالا نرفته و این بدان معنی است که overfit کمتری رخ می‌دهد پس واریانس کاهش می‌یابد.

چ) درصورتی که بایاس بالا باشد، افزایش داده‌های آموزش هیچ تضمینی به کاهش بایاس نمی‌دهد و باید پیچیدگی مدل را افزایش داد.

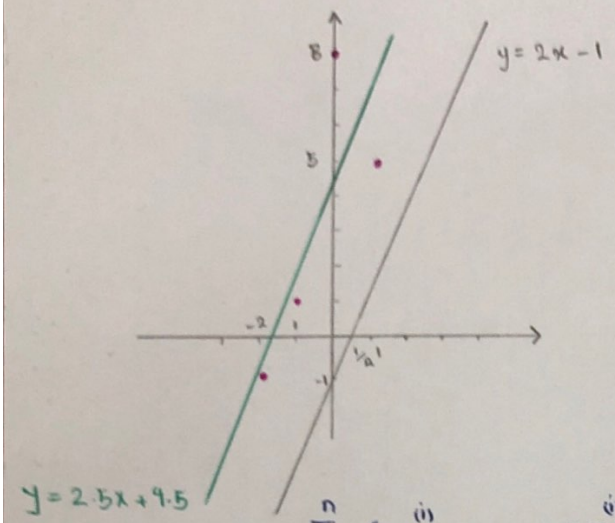
درست است. درصورتی که بایاس بالا باشد همان دچار underfit شده‌ایم پس همین پیچیدگی مدل برای تعداد این مقدار data کافی نبوده و train error بالا است.

با افزایش مقدار داده‌های آموزشی این error بالاتر نیز می‌رود. پس باید پیچیدگی مدل را افزایش دهیم تا به fit بهتری برسیم.

الف ← ۱)

$y = 2x - 1$ $\quad data = \{(-1,1), (0,8), (1,5), (-2,-1)\}$

رسم تابع و نقاط داده شده ← نمودار روبرو :



2) تابع هزینه SSE به شرح مقابل است :

$$J(w) = \sum_{i=1}^{n} \left(y^{(i)} - w_0 - w_1 x^{(i)}\right)^2$$

مدل مسأله مدلی خطی است ←

$$f(x^{(i)}; w) = w_0 + w_1 x^{(i)}$$

برای به دست آوردن خط برازش شده باید از cost-function شروع به پارامترهای آن (w_0, w_1) مشتق بگیریم و مساوی با برابر صفر قرار دهیم تا مقادیر پارامترها را به دست آوریم.

$$\begin{cases} \dfrac{\partial J(w)}{\partial w_0} = \sum_{i=1}^{n} 2\left(y^{(i)} - w_0 - w_1 x^{(i)}\right)(-1) = 0 \\ \dfrac{\partial J(w)}{\partial w_1} = \sum_{i=1}^{n} 2\left(y^{(i)} - w_0 - w_1 x^{(i)}\right)(-x^{(i)}) = 0 \end{cases}$$

$$\begin{cases} -2\sum_{i=1}^{n} y^{(i)} + 2\sum_{i=1}^{n} w_0 + 2\sum_{i=1}^{n} w_1 x^{(i)} = -2\sum_{i=1}^{n} y^{(i)} + 2w_1 \sum_{i=1}^{n} x^{(i)} + 2nw_0 = 0 \\ -2\sum_{i=1}^{n} x^{(i)} y^{(i)} + 2\sum_{i=1}^{n} w_0 x^{(i)} + 2\sum_{i=1}^{n} w_1 x^{(i)2} = -2\sum_{i=1}^{n} x^{(i)} y^{(i)} + 2w_1 \sum_{i=1}^{n} x^{(i)2} + 2w_0 \sum_{i=1}^{n} x^{(i)} = 0 \end{cases}$$

جایگذاری نقاط

$$\begin{cases} -2(1+8+5-1) + 2w_1(-1+0+1-2) + 2\times 4 w_0 = 0 \\ -2\left(\underbrace{-1\times 1}_{-1} + \underbrace{0\times 8}_{0} + \underbrace{1\times 5}_{5} + \underbrace{-2\times -1}_{2}\right) + 2w_1(1+0+1+4) + 2w_0(-1+0+1-2) = 0 \end{cases}$$

$$\begin{cases} -26 - 4w_1 + 8w_0 = 0 \\ -12 + 12w_1 - 4w_0 = 0 \end{cases} \rightarrow \begin{cases} -2w_1 + 4w_0 = 13 \\ 12w_1 - 4w_0 = 12 \end{cases} \xrightarrow{\substack{w_0 = 4.5 \\ w_1 = 2.5}} y = 2.5x + 4.5$$

ب ←

$w^* = \arg\min E_{x,y}[(y - w^T x)^2]$

(۱

$R = E_x[xx^T] \quad , \quad C = E_{xy}[xy]$

$$y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{bmatrix} , \quad X = \begin{bmatrix} 1 & x_1^{(1)} & \dots & x_d^{(1)} \\ 1 & x_1^{(2)} & \dots & x_d^{(2)} \\ \vdots & & \ddots & \\ 1 & x_1^{(n)} & \dots & x_d^{(n)} \end{bmatrix} , \quad W = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix}$$

می‌دانیم ← $\|a\| = a_1^2 + \dots + a_d^2$

$$\|y - Xw\|^2 = (y^{(1)} - w^T x^{(1)})^2 + \dots + (y^{(n)} - w^T x^{(n)})^2 = \sum_{i=1}^{n} (y^{(i)} - w^T x^{(i)})^2 = J(w)$$

حال به دنبال بهینه کردن $J(w)$ به فرم بالا هستیم

$$\nabla_w J(w) = 2X^T(y - Xw) = 0$$

$$X^T y - X^T X w = 0 \rightarrow X^T X w = X^T y \rightarrow w = (X^T X)^{-1} X^T y = X^+ y$$

در نقطه‌ای که $J(w)$ بهینه است یعنی w^* داریم پس :

$$w^* = (X^T X)^{-1} X^T y = X^+ y$$

Pesaudo inverse matrix

نکته ← معکوس ماتریس $X^T X$ وجود دارد زیرا :

$$\underset{(d+1)(N)}{X^T} \times \underset{(N)(d+1)}{X} = \underset{(d+1)(d+1)}{[\ ]} \quad \leftarrow \quad X = N\overset{d+1 \text{ (ویژگی)}}{[\quad]}$$

(N: تعداد داده)

در اکثر مسائلی که با آن روبرو هستیم، تعداد ویژگی‌ها محدود و تعداد Training data بسیار زیاد است. در این گونه مسائل کم پیش می‌آید که $N < d+1$ بشود (مخصوصاً در فضا با ابعاد پایین)

(۲ می‌دانیم $w^* = \arg\min E_{x,y}[(y - w^T x)^2]$. برای Minimize کردن w^* باید از آن نسبت به w گرادیان بگیریم.

$$\nabla_w w^* = -2x(y - w^T x) = 0$$

در نقطه‌ای که w^* بهینه است عبارت بالا برابر صفر می‌گردد یعنی

$$E_{x,y}[-2x(y - w^{*T} x)] = 0$$

$$E_{x,y}\left[(y-\hat{w}^T x)^2\right] = \underbrace{E_{x,y}\left[(y-w^{*T}x)^2\right]}_{\text{Structrul error}} + \underbrace{E_x\left[(w^{*T}x-\hat{w}^T x)^2\right]}_{\text{Approximation error}} \qquad (3$$

$$E_{x,y}\left[(y-\hat{w}^T x)^2\right] = E_{x,y}\left[(y-w^{*T}x+w^{*T}x-\hat{w}^T x)^2\right]$$

$$= E_{x,y}\left[(y-w^{*T}x)^2\right] + E_x\left[(w^{*T}x-\hat{w}^T x)^2\right] + 2\overbrace{E_{x,y}\left[(y-w^{*T}x)(w^{*T}x-\hat{w}^T x)\right]}^{A}$$

حال ادعا می‌کنیم $A=0$ است:

$$A = E_{x,y}\left[(y-w^{*T}x)(w^{*T}x-\hat{w}^T x)\right]$$

$$= E_{x,y}\left[(y-w^{*T}x)(w^*-\hat{w})^T x\right] \quad \xrightarrow{a^T b = b^T a}$$

$$= E_{x,y}\left[(y-w^{*T}x)\,x^T(w^*-\hat{w})\right]$$

$$= (w^*-\hat{w})\,E_{x,y}(y-w^{*T}x) = 0$$

$= 0$ به دلیل سوال 2

4) تعبیر هندسی رابطه بند 3 ←

True error یعنی خطای کل distribution متشکل از Structrul error (خطای ناشی از محدودیت مدل) و خطای تخمین و Approximation error (خطای ناشی از محدود بودن تعداد Training data) می‌باشد.

ضرب این عبارت بند 3 یعنی 2 عبارت $(y-w^{*T}x)$, $(w^{*T}x-\hat{w}^T x)$ برهم عمودند.

پس یعنی این دو خطا در فضای هندسی برهم عمودند.

الف) ابتدا شکل تابع هزینه پرسپترون را می‌توان به صورت $\sum_{i=1}^{n} \max(0, y^{(i)} w^T x^{(i)})$ نوشت، سپس تابع فوق را مشابه با آن را در حالت $y=1$ بر حسب $w^T x$ رسم نمایید.

نتیجه را با تابع هزینه logistic regression (در حالت $y=1$ بر حسب $x^T w$) مقایسه کنید و تفاوت این تفاوت را در عملکرد روش های مربوطه مشخص نمایید.

می‌دانیم cost function = پرسپترون به شرح زیر است:

$$J_p(w) = -\sum_{i \in M} w^T x^{(i)} y^{(i)}$$

که در این رابطه، M نشان‌دهنده نمونه هایی است که غلط دسته بندی شده اند. (misclassified)

و آن را نمونه هایی در نظر می‌گیریم که درست دسته بندی شده. (True classified)

بنابراین cost function که مساله برابر با تابع هزینه‌ی نمونه هایی که درست دسته بندی شده + تابع هزینه نمونه هایی که غلط دسته بندی شده:

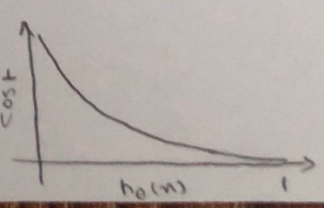
$$= \sum_{i \in T} w^T x^{(i)} y^{(i)} + -\sum_{i \in M} w^T x^{(i)} y^{(i)}$$

می‌دانیم مقدار تابع هزینه‌ی نمونه هایی که درست دسته بندی شده برابر صفر است و چون $\text{sign}(w^T x^{(i)})$ برای داده هایی که غلط دسته بندی شده صفر است، پس عبارت $-\sum w^T x^{(i)} y^{(i)}$ مثبت می‌شود و می‌توان تابع هزینه Perceptron را به صورت مقابل نوشت که اگر مقدار عبارت $-\sum w^T x^{(i)} y^{(i)}$ صفر باشد، حاصل صفر می‌شود. در نتیجه این صورت $\sum_{i=1}^{n} \max(0, y^{(i)} w^T x^{(i)})$

می‌دانیم cost function = Logistic Regression به شرح زیر است:

$$J(\theta) = -\frac{1}{m} \left[\sum_{i=1}^{m} y^{(i)} \log(h_\theta(x^{(i)})) + (1-y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right]$$

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} \log(h_\theta(x^{(i)})) \rightarrow$$



در صورتی که $y=1$ قرار دهیم:

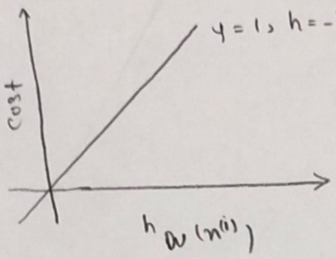
حال اگر $y=1$ را در تابع هزینه پرسپترون جایگذاری کنیم :

$$J_p(w) = -\sum w^T x^{(i)} y^{(i)}$$

به دلیل اینکه پارامترهای تابع هزینه Perceptron شامل بردار ها است پس می‌توان آن را به شرح زیر نوشت : $y=1, h=-1$

$$y=1 \rightarrow J_p(w) = -\sum w^T x^{(i)}$$

$$= -\sum_{i=1}^{n} h_w(x^{(i)})$$



تحلیل : به دلیل اینکه error در logistic regression همواره مثبت است و بالای محور x هاست، و از طرفی perceptron، خطای نمونه‌های درست ارزیابی شده را صفر در نظر می‌گیرد. ممکن است الگوریتم در بهینه محلی گیر افتاده و جواب بهینه را اعلام نکند.

ب) ۱) شکل دهید نمونه‌های کاملاً جدایی‌پذیر خطی، جدایی‌پذیر غیرخطی هستند، اما برعکس آن لزوماً درست نیست.

2) شکل دهید نمونه‌های دوبه‌دو خطی جدایی‌پذیر، لزوماً خطی جدایی‌پذیر نیستند. آیا برعکس نمونه‌های خطی جدایی‌پذیر لزوماً دوبه‌دو خطی جدایی‌پذیر هستند؟

ب) ابتدا تقاضایی از صورت مسئله بهینه‌سازی می‌آوریم و سپس آن را حل می‌کنیم:

$\mu_1 = \frac{\sum_{x^{(i)} \in C_1} x^{(i)}}{N_1} \rightarrow \mu_1' = w^T \mu_1$ ← MAP شده‌ی نقاط کلاس 1 روی بردار ← میانگین = μ_1'

$\mu_2 = \frac{\sum_{x^{(i)} \in C_2} x^{(i)}}{N_2} \rightarrow \mu_2' = w^T \mu_2$ ← MAP شده‌ی نقاط کلاس 2 روی بردار ← میانگین = μ_2'

$s_1'^2$ = واریانس MAP شده‌ی کلاس 1

s_1^2 = واریانس ضد داده‌ها در فضای اصلی است (برای داده‌های کلاس 1)

$s_1^2 = \sum_{x^{(i)} \in C_1} \| x^{(i)} - \mu_1 \|^2$ $\quad s_1'^2 = \sum_{x^{(i)} \in C_1} (w^T x^{(i)} - w^T \mu_1)^2$

همینطور برای کلاس 2.

راه اول ←

پس مسئله‌ی بیشینه‌سازی به شرح زیر می‌شود:

$$J(w) = \frac{|\mu_1' - \mu_2'|^2}{s_1'^2 + s_2'^2} \rightarrow \max J(w)$$

در واقع هدف این بود که میانگین داده‌ها در MAP شده شوند از هم فاصله بگیرند و واریانس MAP شده‌ی آن‌ها کم باشد. ← یعنی داده‌های کلاس در راستای بردار w بهتر جدا شده، دورتر از هم باشند و فشرده‌تر هم باشند.

← صورت مسئله بیشینه‌سازی:

$$|w^T\mu_1 - w^T\mu_2|^2 = w^T \underbrace{(\mu_1 - \mu_2)(\mu_1 - \mu_2)^T}_{S_B} w$$

S_B یک ماتریس $d \times d$ است، چون بردار میانگین داده‌ها یک بردار d بعدی است.

← مخرج مسئله بیشینه‌سازی:

$$\sum_{x^{(i)} \in C_1} (w^T x^{(i)} - w^T\mu_1)^2 + \sum_{x^{(i)} \in C_2} (w^T x^{(i)} - w^T\mu_2)^2$$

$$= w^T \left(\sum_{x^{(i)} \in C_1} (x^{(i)} - \mu_1)(x^{(i)} - \mu_1)^T \right) w + w^T \left(\sum_{x^{(i)} \in C_2} (x^i - \mu_2)(x^i - \mu_2)^T \right) w$$

$$= w^T \left[\sum_{x^{(i)} \in C_1} (x^i - \mu_1)(x^i - \mu_1)^T + \sum_{x^i \in C_2} (x^i - \mu_2)(x^i - \mu_2)^T \right] w = w^T \underbrace{S_w}_{S_1 + S_2} w$$

حال برای حل مسئله بهینه، از آن نسبت به w مشتق می‌گیریم و برابر صفر قرار می‌دهیم.

$$J(w) = \frac{w^T S_B w}{w^T S_w w} \rightarrow \frac{dJ(w)}{dw} = \frac{\frac{d\, w^T S_B w}{dw} \times w^T S_w w - \frac{d\, w^T S_w w}{dw} \times w^T S_B w}{(w^T S_w w)^2}$$

$$= \frac{(2 S_B w)\overbrace{w^T S_w w}^{\beta} - (2 S_w w)\overbrace{w^T S_B w}^{\alpha}}{(w^T S_w w)^2} = 0 \rightarrow \boxed{S_B w = \lambda S_w w} \rightarrow \lambda = \frac{\alpha}{\beta}$$

↓ ادامه

اگر ماتریس S_W معکوس پذیر باشد:

$$S_B w = \lambda S_W w \rightarrow S_W^{-1} S_B w = \lambda w$$

$S_B w$ را باز می‌نویسیم ←

$$S_B w = (\mu_2 - \mu_1)\underbrace{(\mu_2 - \mu_1)^T w}_{\text{عدد}} \propto (\mu_2 - \mu_1)$$

$$\rightarrow S_B w = \alpha(\mu_2 - \mu_1) = S_W w \rightarrow w \propto S_W^{-1}(\mu_2 - \mu_1)$$

$$\rightarrow \boxed{w \propto (\mu_2 - \mu_1)}$$

توضیح دلیل حذف نرم $\|w\| = 1$ ←

همچه w را بزرگتر می‌کنیم، فاصله بزرگ و بزرگتر می‌شود. پس چون می‌خواهیم جهت این بردار را بیابیم $\|w\| = w^T w = 1$ در نظر می‌گیریم.

راه دوم ← می‌دانیم رابطه‌ی تبدیل با استفاده از 2 جمله

$u_2 - u_1 = w^T(u_2 - u_1)$, $u_k = w^T u_k$

بدانبرمی‌شود با:

$$L = w^T(\mu_2 - \mu_1) + \lambda(w^T w - 1)$$

$$\nabla L = \mu_2 - \mu_1 + 2\lambda w = 0 \rightarrow 2\lambda w = \mu_1 - \mu_2 \rightarrow w = \frac{1}{2\lambda}(\mu_1 - \mu_2)$$

$$\rightarrow w = \frac{-1}{2\lambda}(\mu_2 - \mu_1) \rightarrow \boxed{w \propto \mu_2 - \mu_1}$$