باسمه تعالی

گزارش HW2 درس هوش مصنوعی

حل مسئله رگرسیون با شبکه عصبی

ترم دوم سال تحصیلی 1399-1400

اعضای گروه : محمد حسین اصغری(9623003) ، عرفان بهرامی(9624513)، محمدجواد طاهری کیا(9631063)

«توضیحات کد keras»

در ابتدا با استفاده از کتابخانه pandas محتویات دیتای trainها را می خوانیم.

سپس متغیر های ورودی را (X) و متغیرهای خروجی (Y) را در نظر می گیریم. با توجه به اینکه keras با داده های numpy کار میکند،دیتاهای pandas را با استفاده از تابه to_numpy به دیتاهای قابل استفاده ی keras تبدیل می نماییم.حال با توجه به این که شبکه با یک تابع خطی سروکار دارد تنها یک لایه شامل 3 ورودی و 1 خروجی برای شبکه در نظر می گیریم.

```
model = keras.Sequential([Dense(1, input_shape=[3], activation = 'relu')])
```

همان طور که در شکل زیر می بینید از تابع خطا mse و از تابع بهینه ساز RMSprop استفاده کردیم.

```
model.compile(
   loss = 'mse',
   optimizer = RMSprop(0.001), #('empty')
   metrics = ['mean_absolute_error', 'mse']
)
```

با استفاده از model.fit و تنظیم کردن پارامتر های دیتاهای ورودی (x_train) و دیتاهای خروجی (y_train) و قرار دادن epochs=500 شبکه ی عصبی learning را انجام می دهد.

```
history = model.fit(
   x_train, y_train,
   batch_size=128,
   epochs = 500,
   verbose = 1,
   validation_split = 0.2
)
```

پس از این مرحله IPython.embed() را صدا می زنیم تا بتوانیم با python interpreter روی مدل learnشده کار کنیم.

```
IPython.embed()
```

«توضیحات کد sklearn»

در ابتدا همانند keras با استفاده از کتابخانه pandas محتویات دیتای trainها را می خوانیم.

بعد از مشخص کردن ورودی (X) و خروجی (Y) به sklearn اجازه می دهیم فرایند learn را انجام دهد.

پس از این عملیات prediction را انجام می دهیم.

*همان طور که در شکل زیر می بینید دیتاهایی که در صورت پروژه به ما داده شده بود یک ستون خطا داشت.

```
X1,X2,X3,Y
-0.4006259953996795,1.7800959189620613,0.007533697223139559,0
1.4140619334960287,-0.5147649695379848,-0.9381029179264804,-2.5888816069845597
-1.8260619011545498,-0.36704430344595274,0.42120287313215 83,4.280991754456522
-0.7937223198518971,-0.6218158680789405,0.5172604625433712,-3.706282371995305
-0.36216875548932886,0.7982218794965685,1.1563947281022047,-1.482889234168225
0.5573798263478918,1.6343343854875985,0.3822668143486998,-2.678954118577431
-0.4250807462269373,0.5577767432161862,-0.5927675655696607,-0.9018415471405146
-1.3331881971780009,0.12339865031709293,0.11009296774810556,-0.8151706701004001
-0.6352810480311208,0.32111171246533604,1.6240933962233135,-2.8998680124212
-1.3124119069418425,1.1374106601625602,0.46751350467696556,-3.2157672047508914
```

حال برای برطرف کردن این مشکل یک کد python با نام prepare.py نوشتیم که ضمیمه شده است و در زیر مشاهده می کنید.

```python
import sys
def prepare_data(file, out, x_size, y_size):

        input = open(file, "rb")
        content = input.read()

        lines = content.split('\n')
        lines_ = []
        for i in range(0, len(lines)):
                lines_.append(lines[i].split(','))

        str = ""
        lines_ = lines_[:-1]
        for i in range(0, len(lines_) -1):
                for j in range(0, x_size):
                        str += lines_[i][j]
                        str += ","
                for k in range(0, y_size):
                        if i != 0:
                                str += lines_[i+1][k + x_size]
                        else:
                                str +=lines_[i][k + x_size]
                        if k != y_size-1:
                                str += ","
                str += "\n"

        out = open(out, "wb")
        out.write(str)

if __name__ == "__main__":
        if len(sys.argv) < 5:
                print("[-] error specify args!")
        else:
                prepare_data(sys.argv[1], sys.argv[2], int(sys.argv[3]), int(sys.argv[4]))
```

سپس با اجرای این فایل به صورت زیر دیتاها را به شکل صحیح در می آورد.

```
┌──(v1510n㉿kali)-[~/Documents/AI/linear_regression]
└─$ python prepare.py data_2.csv data_2_out.csv 4 1
```

شکل صحیح دیتاها :

```
  GNU nano 5.3                          out_1.csv
X1,X2,X3,Y
-0.4006259953996795,1.7800959189620613,0.007533697223139559,-2.5888816069845597
1.4140619334960287,-0.5147649695379848,-0.9381029179264804,4.280991754456522
-1.8260619011545498,-0.36704430344595274,0.4212028731321583,-3.706282371995305
-0.7937223198518971,-0.6218158680789405,0.5172604625433712,-1.482889234168225
-0.36216875548932886,0.7982218794965685,1.1563947281022047,-2.678954118577431
0.5573798263478918,1.6343343854875985,0.3822668143486998,-0.9018415471405146
-0.4250807462269373,0.5577767432161862,-0.5927675655696607,-0.8151706701004001
-1.3331881971780009,0.12339865031709293,0.11009296774810556,-2.8998680124212
-0.6352810480311208,0.32111171246533604,1.6240933962233135,-3.2157672047508914
-1.3124119069418425,1.1374106601625602,0.46751350467696556,-4.229747978723211
```

و در انتها هم اجرای کد روی dataset شماره 1:

```
┌──(v1510n㉿kali)-[~/Documents/AI/linear_regression]
└─$ python3 solve_1.py
[ 2. -1. -1.]
```

و اجرای کد روی dataset شماره 2:

```
┌──(v1510n㉿kali)-[~/Documents/AI/linear_regression]
└─$ python3 solve_2.py
[ 2.  1. -1. -1.]
```

ضرایب لایه ی اول که در شبکه توسط keras ، learn شده است به صورت زیر قابل مشاهده است.

```
5.8753e-07 - val_mean_absolute_error: 6.2844e-04 - val_mse: 5.8753e-07
Epoch 500/500
63/63 [==============================] - 0s 2ms/step - loss: 1.0877e-06 - mean_absolute_error: 7.6395e-04 -
1.2539e-06 - val_mean_absolute_error: 9.1186e-04 - val_mse: 1.2539e-06
Python 3.9.2 (default, Feb 28 2021, 17:03:44)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.25.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: model.layers[0].weights
Out[1]:
[<tf.Variable 'dense/kernel:0' shape=(3, 1) dtype=float32, numpy=
 array([[ 1.9994024 ],
        [-0.99971986],
        [-0.9994609 ]], dtype=float32)>,
 <tf.Variable 'dense/bias:0' shape=(1,) dtype=float32, numpy=array([-0.00072543], dtype=float32)>]

In [2]:
```

قسمت prediction مدل که برای داده های ورودی ، خروجی مناسب را با تقریب خوبی حدس زده است.

```
In [3]: prediction
Out[3]:
array([[-2.588865 ],
       [ 4.2787714],
       [-3.7057922],
       ...,
       [ 4.690188 ],
       [-2.0849106],
       [-2.009317 ]], dtype=float32)

In [4]:
```

«تحقیقات در مورد پروژه»

تابع فعال سازی :

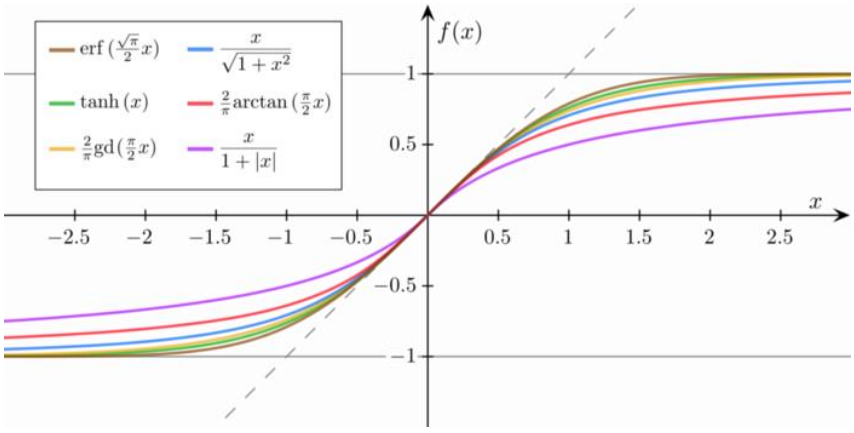
همان طور که در شکل های بالا دیدید ما در کد خود از تابع فعالسازی ReLU استفاده کردیم:

حال به بررسی انواع توابع فعالسازی در شبکه عصبی می پردازیم:

- Logistic (**Sigmoid**)
- Hyperbolic Tangent (**Tanh**)
- Rectified Linear Activation (**ReLU**)

1-تابع سیگموید

در یادگیری ماشین، برای پیاده‌سازی شبکه‌های عصبی ساده و از توابع سیگموید استفاده می‌شود. این توابع، واحدهای فعال‌سازی مقدماتی هستند. اما با توجه به ایرادت و نواقص توابع سیگموید ترجیح بر این است که از این توابع در شبکه‌های عصبی پیشرفته استفاده نشود.



تابع سیگموید و مشتق آن ساده هستند و مدت زمان ساخت مدل را کاهش می‌دهند، اما از آنجایی که بازه مشتق آن کوتاه است، در این تابع با مشکل info loss مواجه هستیم:

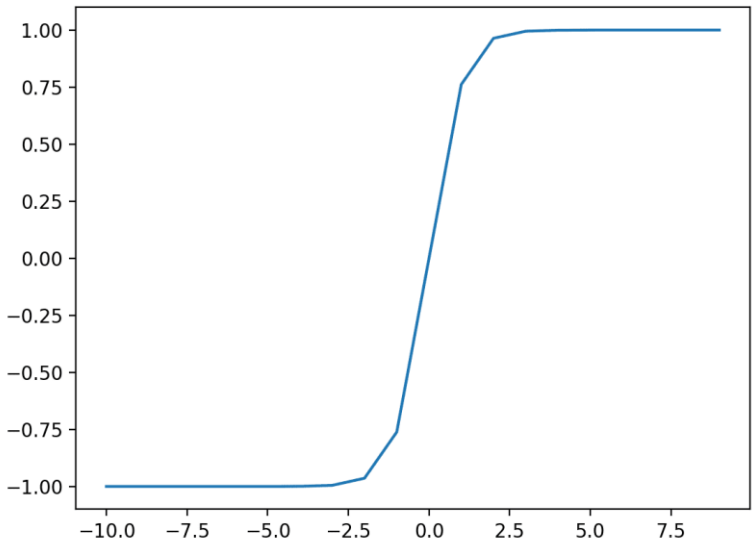
$$f'(x) = f(x)(1 - f(x))$$

لذا، هرچه شبکه عصبی ما لایه‌های بیشتری داشته باشد و یا به عبارتی عمیق‌تر باشد، در هر لایه اطلاعات بیشتری فشرده‌سازی می‌شوند و حذف می‌شوند. در نتیجه داده‌های بیشتری از بین می‌روند.

2-تابع فعال سازی تانژانت هیپربولیک

بسیار شبیه به تابع فعال سازی سیگموئید است و حتی همان شکل S را دارد.این تابع هر مقدار واقعی را به عنوان ورودی در نظر می گیرد و مقادیر خروجی را در بازه 1 تا 1 قرار می دهد.. هرچه ورودی بزرگتر باشد (مثبت تر باشد) ، مقدار خروجی به 1 نزدیکتر خواهد بود ، در حالی که هر چه ورودی کوچکتر (منفی تر) ، خروجی به 1- نزدیکتر خواهد بود. تابع فعال سازی Tanh به صورت زیر محاسبه می شود:

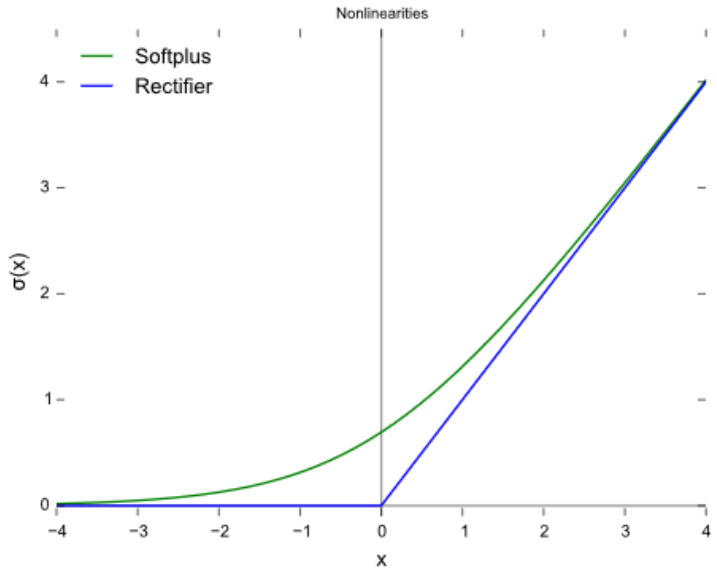
(e^x – e^-x) / (e^x + e^-x)



3- تابع ReLU

از سال 2018 محبوب ترین تابع فعالسازی شناخته می شود. در حال حاضر، بیشتر برنامه‌های برای انجام مسائل مرتبط با بینایی کامپیوتر، پردازش متن، تشخیص صوت و شبکه‌های عصبی عمیق به جای توابع فعال سازی لجستیک از تابع ReLUاستفاده می‌کنند. تابع ReLU انواع گوناگونی دارد :

Softplus (SmoothReLU)، Noisy ReLU، Leaky ReLU ، Parametric ReLU و ExponentialReLU (ELU)



در این تابع فعالسازی در صورتی‌که ورودی کمتر از ۰ باشد، تابع فعال‌سازی ReLU صفر (۰) و در غیر اینصورت مقدار خام را خروجی می‌دهد. به عبارت دیگر، اگر مقدار ورودی بیشتر از ۰ باشد، تابع ReLU همان مقدار ورودی را خروجی می‌دهد. عملکرد تابع فعال‌سازی ReLU از جهات بسیاری مشابه عملکرد نورون‌های زیستی ما است:

$$f(x) = max(x, 0)$$

ReLUیک تابع غیرخطی است و برخلاف تابع سیگموید با خطاهای پس‌انتشار مواجه نمی‌شود. علاوه بر این، اگر در شبکه‌های عصبی بزرگ‌تر به جای تابع سیگموید از تابع ReLU استفاده کنیم، سرعت مدل‌سازی بیشتر خواهد بود، به عبارت دیگر مدت زمان مدل‌سازی کاهش می‌یابد:

باورپذیری بیولوژیکی: این تابع برخلاف تابع پادتقارن tanh ، یک جانبه است.

فعال‌سازی پراکنده :برای مثال، در شبکه‌ای که به صورت تصادفی مقداردهی شده است، حدود ۵۰ درصد از واحدهای پنهان، فعال می‌شوند (و خروجی آن‌ها غیرصفر خواهد بود).

انتشار بهتر گرادیان: در این تابع برخلاف توابع فعال سازی سیگموید کمتر با مشکل محوشدگی گرادیان مواجه می‌شویم.

محاسبات اساسی: در این تابع فقط از مقایسه، جمع و ضرب استفاده می‌شود.

مقیاس‌پذیر ۰: max (0, a x) = a max (0 , x) for a ≥

توابع ReLU هم کاستی‌هایی دارند. برای مثال، میانگین این تابع صفر نیست و در صفر مشتق نمی‌شود، اما در هر جای دیگری مشتق می‌شود.

مقایسه سیگموید وReLU

مشکل دیگری که در تابع ReLU با آن مواجه هستیم، مشکل مرگ ReLU است. منظور از مرگ ReLU این است که برخی از نورون‌های ReLU می‌میرند و غیرفعال می‌شوند و برای تمامی ورودی‌ها، صفر (۰) را خروجی می‌دهند. در این حالت، هیچ گرادیانی جریان نمیابد و در صورتی‌که تعداد نورون‌های غیرفعال در شبکه عصبی زیاد باشد، عملکرد مدل تحت تأثیر قرار می‌گیرد. برای حل این مشکل می‌توانیم از تابع Leaky ReLU استفاده کنیم؛ Leaky ReLU در نمودار بالا همان قسمتی است که شیب در سمت چپ x=0 تغییر کرده است و در نتیجه باعث گسترش یا به اصطلاح نشتی بازه تابع ReLU می‌شود.

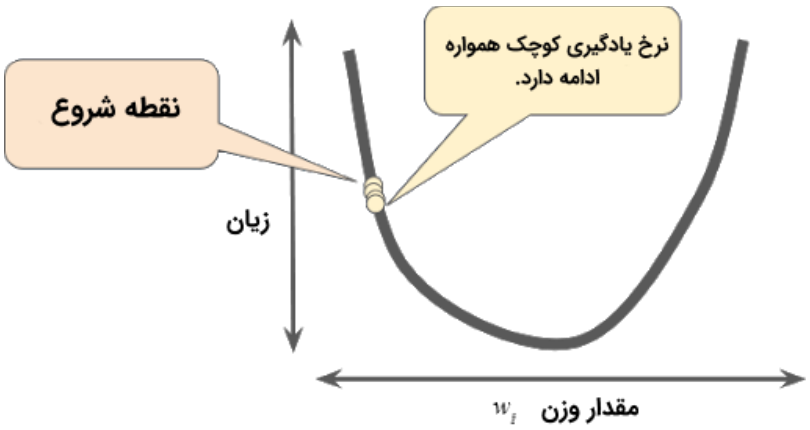
بهینه سازی :

همان طور که در شکل های ابتدای گزارش دیدید ما در کد خود از RMSprop optimizer استفاده کردیم:

حال به بررسی روش های مختلف بهینه سازی می پردازیم:

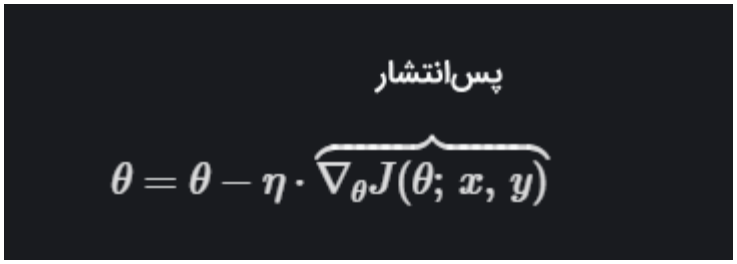
گرادیان کاهشی

محبوب‌ترین روش بهینه‌سازی محسوب می‌شود. ایده نهفته در پس این روش، به روز رسانی «تکرارشونده» (iteratively) در جهت مثبت «تابع هدف»(objective function) است. با هر به روز رسانی، این روش مدل را به سمت پیدا کردن هدف هدایت می‌کند و به تدریج به مقدار بهینه تابع هدف همگرا می‌شود.



گرادیان کاهشی تصادفی

برای حل مشکل پیچیدگی محاسباتی موجود در هر تکرار برای داده‌های کلان مقیاس در گرادیان کاهشی، معرفی شد. معادله این روش به صورت زیر است :



دریافت مقادیر و تنظیم آن‌ها به صورت بازگشتی روی پارامترهای گوناگون به منظور کاهش «تابع زیان» (loss function) ، «پس‌انتشار» (BackPropagation | BP) نامیده می‌شود. در این روش، یک نمونه برای به روز رسانی گرادیان (Theta) مورد استفاده قرار می‌گیرد؛ به جای آنکه به طور مستقیم مقدار دقیق گرادیان محاسبه شود. گرادیان تصادفی یک تخمین بدون سوگیری از گرادیان حقیقی (Real Gradient) ارائه می‌کند. این روش بهینه‌سازی زمان به روز رسانی را برای سر و کار داشتن با تعداد زیادی از نمونه‌ها کاهش می‌دهد و میزان خاصی از «افزونگی» (Redundancy) را حذف می‌کند.

روش نرخ یادگیری تطبیقی

نرخ یادگیری یکی از فراپارامترهایی (Hyperparameters) است که در بهینه‌سازی وجود دارد. نرخ یادگیری تصمیم می‌گیرد که مدل از بخش خاصی از داده‌ها پرش کند. اگر نرخ یادگیری بالا باشد، مدل ممکن است که جنبه‌های ظریف‌تر داده‌ها را از دست بدهد و در واقع، به آن‌ها توجه نداشته باشد. اگر این مقدار کم باشد، برای کاربردهای جهان واقعی مطلوب است. نرخ یادگیری تاثیر قابل توجهی روی گرادیان کاهشی تصادفی (SGD) دارد. تنظیم مقدار صحیح برای نرخ یادگیری کاری چالش‌برانگیز است. روش‌های تطبیقی ارائه شده‌اند تا این تنظیمات را به صورت خودکار انجام دهند.
انواع تطبیقی SGD به طور گسترده در شبکه های عصبیمورد استفاده قرار می گیرد. روش هایی مانند AdaDelta و RMSProp و Adam از Exponential Averaging برای فراهم کردن به روز رسانی‌های موثر و ساده کردن محاسبات استفاده می‌کنند.

- Adagrad : وزن‌ها با گرادیان بالا نرخ یادگیری پایینی خواهند داشت و بالعکس.
- RMSprop : روش Adagrad را به صورتی تنظیم می‌کند که نرخ یادگیری به طور یکنواخت رو به کاهش آن را کاهش دهد.
- Adam : شباهت زیادی به RMSProp دارد، با این تفاوت که تکانه دارد.

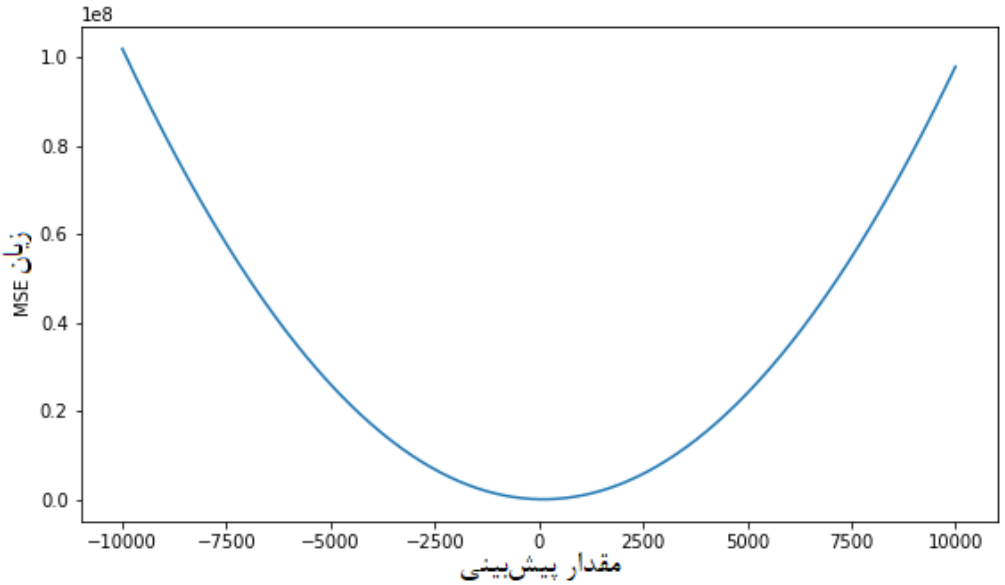
توابع خطا :

1- تابع زیان میانگین مربعات(Means Square Error)

یکی از معروف‌ترین و معمول‌ترین توابع زیان در تحلیل رگرسیونی، میانگین مربعات خطا (Means Square Error) است که به اختصار MSE نامیده می‌شود. این تابع زیان، میانگین مربعات فاصله بین مقدار پیش‌بینی و واقعی را محاسبه می‌کند. شیوه و نحوه محاسبه آن در زیر دیده می‌شود:

$$MSE = \frac{\sum(y_i - \hat{y}_i)^2}{n}$$

در مباحث آماری، معمولا به چنین تابعی، زیان L^2گفته می‌شود. با توجه به استفاده از توان ۲ در محاسبهMSE ، شکل این تابع زیان برحسب مقدارهای پیش‌بینی (یا خطا) به صورت سهمی خواهد بود. فرض کنید که مقدار واقعی برای متغیر پاسخ (y) برابر است با صفر، در نتیجه نمودار تابع زیان MSE را براساس مقدارهای پیش‌بینی در محدوده ۱۰۰۰۰- تا ۱۰۰۰۰ می‌توان به صورت زیر رسم کرد.



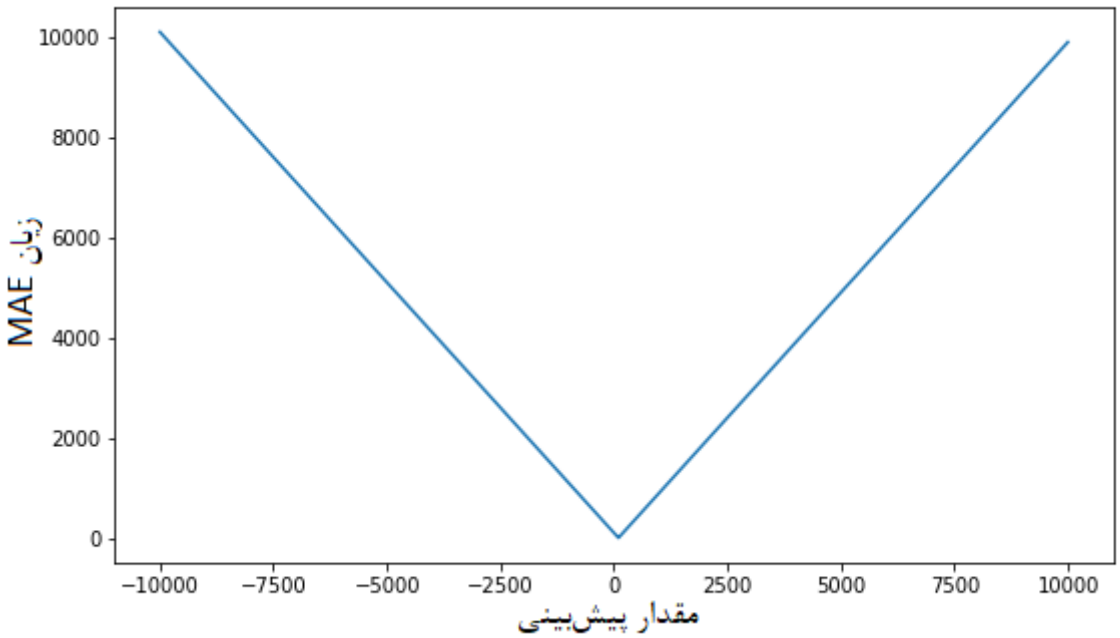
2- میانگین قدرمطلق خطا(Mean Absolute Error)

یکی دیگر از توابع زیان که خواص جالبی دارد، میانگین قدرمطلق خطا (Mean Absolute Error) است که به اختصار MAE نیز نامیده می‌شود. این تابع زیان، به مانند MSE از فاصله بین مقدار پیش‌بینی و واقعی به عنوان معیار استفاده کرده ولی جهت این تفاضل را در نظر نمی‌گیرد. بنابراین در محاسبه خطا MAE فقط میزان فاصله و نه جهت فاصله به کار می‌رود. البته گاهی در مباحث آماری، به این تابع، زیان L^1 نیز گفته می‌شود.

بنابراینMAE ، میانگین قدرمطلق تفاضل بین مقدار پیش‌بینی و واقعی را محاسبه می‌کند. شیوه بدست آوردن MAE در رابطه زیر نوشته شده است.

$$MAE = \frac{\sum |y_i - \hat{y}_i|}{n}$$

در تصویر زیر، شکل تابع زیان MAE ترسیم شده است. در این نمودار فرض بر این است که مقدار واقعی متغیر پاسخ برابر با صفر است.



مزایای استفاده از MAE

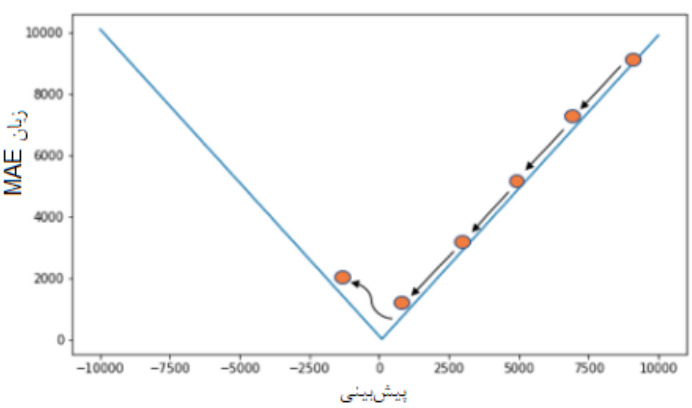
از آنجایی که در محاسبه زیان MSE از مربع خطا استفاده می‌شود، اگر میزان خطا بزرگتر از ۱ باشد، MSEبزرگ می‌شود. با وجود داده پرت خطا افزایش یافته و زیانMSE ، خطا را بسیار بزرگتر نشان می‌دهد. بنابراین به نظر می‌رسد که داده‌های پرت در محاسبه MSE وزن بیشتری نسبت به MAE دارند. بنابراین هنگامی که داده‌های پرت یا دورافتاده در مشاهدات وجود دارد، استفاده از تابع زیان MAE می‌تواند کارایی مدل را بالا ببرد.

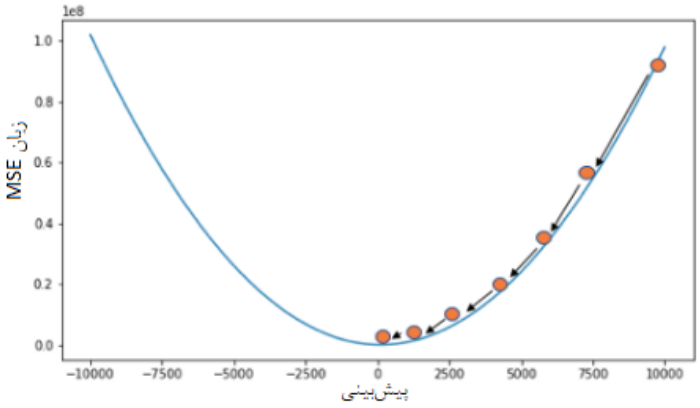
اگر بخواهیم به طوری شهودی دو تابع زیان MSE و MAE را مقایسه کنیم، می‌توان این طور تصور کرد که: «اگر بخواهیم برای همه مقدارهای متغیر مستقل، یک مقدار به عنوان متغیر پاسخ پیدا کنیم که تابع زیان MSE را کمینه کند، این نقطه میانگین مقدارهای متغیر پاسخ خواهد بود. در مقابل، مقداری که تابع زیان MAE را به ازاء همه مقدارهای متغیر مستقل کمینه می‌کند، میانه مقدارهای متغیر پاسخ است.

می‌دانیم که میانه در مقابل داده‌های پرت مقاوم است و نسبت به میانگین بسیار کمتر تحت تاثیر قرار می‌گیرد. بنابراین MAEنسبت به MSE کمتر تحت تاثیر داده‌های پرت بوده و می‌تواند برآورد بهتری برای پارامترهای مدل یا پیش‌بینی متغیر پاسخ ارائه دهد.

معایب استفاده از MAE

استفاده از تابع زیانMAE ، بخصوص در شبکه‌های عصبی، با یک مشکل بزرگ روبرو است. هنگام استفاده ازMAE ، مقدار گرادیان تابع زیان، برای مقدارهای کوچک خطا نیز بزرگ است. این وضعیت در الگوریتم‌های یادگیری ماشین مناسب نیست. برای حل این مشکل بهتر است از نرخ یادگیری پویا(Dynamic Learning Rate) استفاده شود که هنگام نزدیک شدن به نقطه کمینه، مقدارش کاهش می‌یابد. ولی گرادیان تابع زیانMSE ، زمانی که مقدار خطا، بزرگ باشد، زیاد و هنگامی که خطا کم شود، کاهش خواهد یافت. نمودارهایی که در تصاویر زیر دیده می‌شوند، این مطلب را بهتر نشان می‌دهند. در هر دو نمودار، فرض بر این است که مقدار واقعی متغیر پاسخ برابر با صفر است.





همانطور که در تصویر سمت چپ دیده می‌شود، برای مقدارهای پیش‌بینی شده با خطاهای بزرگ یا کوچک، شیب خط تابع زیان MAE ثابت است. در حالیکه در تصویر سمت راست، مقدار تابع زیانMSE ، شیب منحنی برای خطاهای بزرگ، زیاد بوده و هنگامی که مقدار پیش‌بینی به مقدار واقعی نزدیک می‌شود (کاهش خطا)، شیب منحی تابع زیان MSE نیز کاهش می‌یابد.

3- تابع زیان هوبر (Huber Loss) یا میانگین خطای قدرمطلق هموار شده(Smooth Mean Absolute Error)

تابع زیان هوبر نسبت به MSE کمتر تحت تاثیر داده‌های پرت است. همچنین، برعکس تابع زیانMAE ، مشتق‌پذیر بوده و کمینه‌سازی آن به راحتی امکان پذیر است .

یکی از مشکلات عمده، هنگام استفاده از تابع زیان MAE در یادگیری شبکه عصبی، بزرگ بودن مشتق است که ممکن است باعث شود، کمینه مقدار تابع زیان در پایان مراحل یادگیری بوسیله الگوریتم گرادیان کاهشی حاصل نشود. در عوض استفاده از تابع زیان هوبر باعث می‌شود، زمانی که میزان خطا کاهش می‌یابد، مشتق کاهش یافته و دسترسی به کمیته مقدار تابع زیان هوبر میسر شود.

بنابراین تابع زیان هوبر از مزایای هر دو تابع زیان MSE و MAE بهره برده و معایب آن‌ها را هم ندارد. به این ترتیب این تابع زیان، زمانی که داده‌های پرت وجود داشته باشد، اثر آن‌ها را در محاسبات کم می‌کند و زمانی که خطا کاهش یابد، امکان کمینه‌سازی تابع زیان هوبر به راحتی امکان پذیر است.

البته مسئله اصلی در به کاریگیری این تابع زیان، انتخاب مناسب پارامتر آن یعنی δδاست که معمولا به وسیله یک فرآیند تکراری یادگیری، تعیین می‌شود. شاید تنظیم این پارامتر به کمک روش‌های cross validation ما را به مقدار مناسب برایδ برساند.

4-تابع زیان لگاریتم کسینوس هذلولوی(Log-Cosh)

یکی دیگر از توابع زیان که بخصوص در رگرسیون به کار می‌رود، زیان کسینوس هذلولی (Logarithm of Hyperbolic Cosine)است که آن را به اختصار Log-Cosh نیز می‌گویند. برای محاسبه آن کافی است از کسینوس هذلولوی اختلاف یا خطای بین مقدار پیش‌بینی و واقعی متغیر پاسخ، لگاریتم بگیریم.

$$L_{log-cosh}(y, \hat{y}) = \sum_{i=1}^{n} \log(\cosh(y - \hat{y}))$$

ین تابع زیان، تحت تاثیر داده‌های پرت قرار نمی‌گیرد و از طرف دیگر مشتق دوم آن نیز وجود دارد. وجود مشتق دوم، از مزایای این تابع زیان است که در تابع زیان هوبر از آن محروم بودیم.

پایان

با تشکر از زحمات شما