یادگیری عمیق

تمرین اول

استاد درس: دکتر محمدی

مهسا موفق بهروزی

زمستان ۱۴۰۱

سوال یک)

الف: ابتدا باید از فرمول softmax مقدار پیش‌بینی ($\hat{y}$) را برای تک تک نمونه‌ها محاسبه کرد. سپس در فرمول cross entropy جای‌گذاری کرده و مقدار loss را به دست می‌آوریم.

از آن‌جایی که در cross entropy مقدار y در $\log \hat{y}$ ضرب خواهد شد و مقدار y برای ۳ نمونه صفر است، برای محاسبه‌ی تابع ضرر، کافی‌ست فقط مقدار softmax را برای نمونه اول محاسبه کنیم. اما از آن‌جایی که برای محاسبه‌ی گرادیان به تمامی موارد نیاز داریم، محاسبات به‌طور کامل و با استفاده از کد پایتون انجام شده است.

$$\hat{y}_i = softmax(o_i) = \frac{\exp\,(o_i)}{\sum_{k=1}^{q} \exp\,(o_i)}$$

```
import numpy as np
np.set_printoptions(suppress=True)
```

```
def softmax(x):
  e_x = np.exp(x - np.max(x))
  return e_x / e_x.sum()
```

```
x = [4, 6, -10, -20]
softmax(x)
```

```
array([0.11920291, 0.88079699, 0.0000001 , 0.        ])
```

$$\hat{y}_1 = softmax(4) = \frac{\exp\,(4)}{\exp(4)+\exp(6)+\exp(-10)+\exp\,(-20)} = 0.11$$

$$\hat{y}_2 = softmax(6) = \frac{\exp\,(6)}{\exp(4)+\exp(6)+\exp(-10)+\exp\,(-20)} = 0.88$$

$$\hat{y}_1 = softmax(-10) = \frac{\exp\,(-10)}{\exp(4)+\exp(6)+\exp(-10)+\exp\,(-20)} = 0.0000001$$

$$\hat{y}_1 = softmax(-20) = \frac{\exp\,(-20)}{\exp(4)+\exp(6)+\exp(-10)+\exp\,(-20)} = 0.$$

$$cross\ entropy\ (y\ ,\hat{y}_i) = -\sum_{i=1}^{q} y_i\ \log \hat{y}_i$$

```
-(ln(math.exp(4))- ln(math.exp(4) + math.exp(6) + math.exp(-10) + math.exp(-20)))
```

```
2.1269281101681203
```

$cross\ entropy\ (y\ ,\hat{y}_i) = -\ \log \hat{y}_1 = 2.12$

$$\partial_{o_i} l(y, \hat{y}) = \frac{\exp\ (o_i)}{\sum_{k=1}^{q} \exp\ (o_i)} - y_i = softmax(o_i) - y_i$$

$\partial_{o_1} l(y, \hat{y}_1) = softmax(o_1) - y_1 = 0.11 - 1 = -0.89$

$\partial_{o_2} l(y, \hat{y}_2) = softmax(o_2) - y_2 = 0.88 - 0 = 0.88$

$\partial_{o_3} l(y, \hat{y}_3) = softmax(o_3) - y_3 = 0.0000001 - 0 = 0.0000001$

$\partial_{o_4} l(y, \hat{y}_4) = softmax(o_4) - y_4 = 0. - 0 = 0.$

ب: این‌بار به جای جای‌گذاری output ها در تابع softmax، از تابع sigmoid استفاده می‌کنیم.

$$\text{sigmoid}(o_i) = \frac{1}{1 + \ exp^{-o_i}}$$

```
def sigmoid(x):
  return 1/(1 + np.exp(-x))
```

```
x = np.array([4, 6, -10, -20])
sigmoid(x)
```

```
array([0.98201379, 0.99752738, 0.0000454 , 0.        ])
```

$\text{sigmoid}(4) = \frac{1}{1+ exp^{-4}} = 0.98$

$\text{sigmoid}(6) = \frac{1}{1+ exp^{-6}} = 0.99$

$\text{sigmoid}(-10) = \frac{1}{1+ exp^{10}} = 0.0000454$

$\text{sigmoid}(-20) = \frac{1}{1+ exp^{20}} = 0.$

$$binary\ cross\ entropy\ (y\ , \hat{y}) = -\sum_{i=1}^{n} y_i\ \log \hat{y}_i + (1 - y_i\ )\ \log(1 - \hat{y}_i)$$

با جای‌گذاری مقادیر Y = [1 , 0, 0, 0] و ساده کردن فرمول بالا به رابطه‌ی زیر خواهیم رسید:

$$\log \hat{y}_1 + \log(1 - \hat{y}_2) + \log(1 - \hat{y}_3) + \log(1 - \hat{y}_4)$$

```
ln(0.98201379) + ln(1 - 0.99752738) + ln(1 -0.0000454) + ln(1 - 0.        )
-6.020672290759547
```

و با جای‌گذاری مقادیر پیش‌بینی که محاسبه کرده‌ایم و با استفاده از کد پایتون، جواب نهایی برابر با 6.02- خواهد بود.

برای محاسبه‌ی گرادیان:

$$l = -y\ \log \sigma(o) - (1-y)\ \log(1 - \sigma(o))$$

$$\text{If y = 0: } l = -\ \log(1 - \sigma(o))$$

$$\frac{dl}{do} = \sigma(o)$$

$$\text{If y = 1: } l = -\log \sigma(o)$$

$$\frac{dl}{do} = \sigma(o) - 1$$

که در حالت کلی می‌توان مشتق loss نسبت به o را به‌صورت: $\sigma(o) - y$ نوشت.

```
x = np.array([4, 6, -10, -20])
y = np.array([1, 0, 0, 0])
sigmoid(x) - y
```

```
array([-0.01798621,  0.99752738,  0.0000454 ,  0.        ])
```

$\sigma(o_1) - y_1$ = -0.017

$\sigma(o_2) - y_2$ = 0.99

$\sigma(o_3) - y_3$ = 0.0000454

$\sigma(o_3) - y_4$ = 0.

سوال دو)

فرمول تابع ضرر hinge بر اساس مرجع ارائه شده در سوال به‌صورت زیر می‌باشد:

$$l(y) = \max(0, 1 - t.y)$$

که t ، output و y پیش‌بینی مدل است.

محور افقی نمودار رسم شده همان $t.y$ می‌باشد. با در نظر گرفتن $z = t.y$ ، تابع به شکل $\max(0, 1 - z)$ درآمده که با رسم تابع، نمودار به رنگ آبی را خواهیم داشت.

فرمول تابع ضرر cross entropy به‌صورت زیر می‌باشد:

$$-\sum_{i=1}^{n} y_i \log \hat{y}_i$$

با در نظر گرفتن y=1، محور افقی نمودار به $\log \hat{y}$ تبدیل می‌شود. با در نظر گرفتن $\hat{y} = sigmoid(o)$ در نقطه‌ی 2- داریم:

```python
def sig(x):
  return 1 / (1 + np.exp(-x))
```

```python
-np.log2(sig(-2))
```

```
3.068508493859523
```

با توجه به مقدار به دست آمده، ۲ نمودار بنفش و نارنجی رد شده و برای رد کردن تابع آبی رنگ نیز کافی‌ست مقدار تابع در نقطه‌ای مانند ۲ نیز محاسبه شود.

مقدار تابع CE در نقطه‌ی ۲ با فرضیات گفته شده و طبق کد بالا برابر با ۰.۱۸ است، درحالیکه در نمودار آبی رنگ برابر با ۰ است. پس نمودار آبی نیز رد شده و نمودار سبز، نمودار CE می‌باشد.

```python
def sig(x):
  return 1 / (1 + np.exp(-x))
```

```python
-np.log2(sig(2))
```

```
0.1831184120815962
```

سوال سه)

الف: ابتدا مدل رگرسیون خطی را با استفاده از ماژول nn می‌سازیم. برای این‌کار لازم داریم سایز ورودی‌ها و خروجی‌ها را در nn.Linear مشخص کنیم. از آن‌جایی که با دیتاست MNIST که حاوی تصاویر ۲۸*۲۸ پیکسل می‌باشد کار می‌کنیم، اندازه ورودی ۲۸*۲۸ خواهد بود و مساله ۱۰ کلاسه است، پس به ۱۰ خروجی نیاز داریم.

در بخش Prediction از notebook ابتدا تابع ضرر و بهینه‌ساز مساله را تعریف میکنیم. در این مساله، تابع ضرر، که معمولا با نام criterion تعریف می‌شود، Cross entropy می‌باشد که برای مسائل دسته‌بندی مناسب است. همچنین از optimizerِ Adam استفاده شده که با توجه به آزمون و خطا بین SGD و Adam، انتخاب شده است.

هر حلقه‌ی training شامل سه مرحله می‌باشد:

یک) داده‌های ورودی وارد مدل شده، و مدل یک پیش‌بینی انجام می‌دهد و سپس بر اساس پیش‌بینی مدل و مقدار برچسب واقعی، مقدار خطا محاسبه می‌شود.

این کار در قطعه کد زیر انجام گرفته است:

```
x, y = data
output = model(x.view(-1, 28*28))
loss = criterion(output, y)
```

استفاده از view به این منظور است که در هر سطر از ماتریس x یک پیکسل قرار گیرد.

دو) با استفاده از مقدار loss محاسبه شده و optimizer، وزن‌ها در شبکه آپدیت می‌شوند؛ که در قطعه کد زیر این کار صورت گرفته است:

```
loss.backward()
optimizer.step()
```

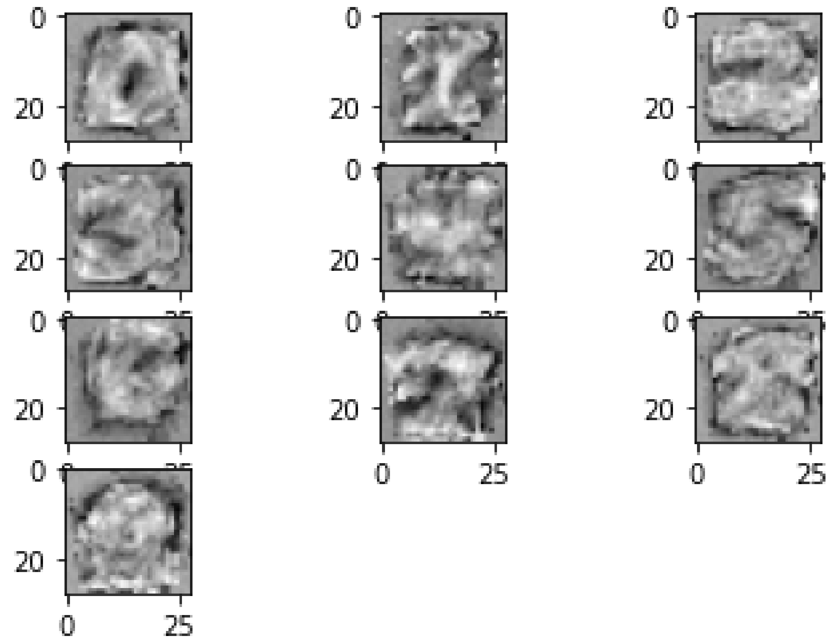
سه) در نهایت پیش از epoch بعدی، گرادیان‌ها صفر می‌شوند:

```
optimizer.zero_grad()
```

پس از اتمام مرحله‌ی training، نمودارهای خطا و دقت ترسیم شده است که بخشی از آن به‌صورت آماده در notebook تمرین بود و کافی بود دیتای مورد نیاز به آن‌ها اضافه شود:

```
iters.append(epoch)
iters_sub.append(epoch)
losses.append(loss.item())
val_acc.append(get_accuracy(model,mnist_val))
train_acc.append(get_accuracy(model,mnist_train))
```

ب: وزن‌های مربوط به ۱۰ کلاس در شکل زیر قابل مشاهده است:



نکته قابل مشاهده این است که شبکه لبه‌یابی کرده و یک ماسک از پترن کلی نمایش هر کاراکتر ایجاد کرده است.

سوال چهار)

الف: پاسخ این بخش به‌صورت دستی نوشته شده و عکس آن در ادامه قرار داده می‌شود.

سؤال ۴ - قسمت الف)

$$\hat{y} = \frac{d}{s} w_0 + t w_1 + weather\ w_2 + b + \varepsilon$$

$$\varepsilon \sim \mathcal{N}(\mu = 0, \sigma^2)$$

اگر میانگین خطا ۰ باشد، میانگین y که برابر با $y = \hat{y} + \varepsilon$ است، به اندازه $\hat{y}$ شیفت یافته.

$$P(y \mid x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(y - \hat{y})^2\right)$$

$$P(y \mid X) = \prod_{i=1}^{n} P(y^i \mid x^i)$$

$$\log P(y \mid X) = \sum_{i=1}^{n} \log P(y^i \mid x^i)$$

$$= \sum_{i=1}^{n} -\frac{1}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}(y - \hat{y})^2$$

میتوان عبارت بالا را max کرد یا منفی آن را min کرد.

$$-\log P(y \mid X) \xrightarrow{\text{با ثابت در نظر گرفتن } \sigma^2} \min \sum_{i=1}^{n} (y - \hat{y})^2$$

برای یافتن مقدار بهینه تابع، میتوان مشتق گرفت و مساوی صفر قرار داد.

$$\min \sum_{i=1}^{n} \underbrace{\left(y - \left(\frac{d_i}{s_i} w_0 + t_i w_1 + weather_i\ w_2 + b + \varepsilon\right)\right)^2}_{f}$$

مساوی ۰ قرار میدهیم پس میتوان در نظر نگرفت

$$\frac{\partial \sum_{i=1}^{n} f}{\partial w_0} = 0 = \sum_{i=1}^{n} -2 f \left(\frac{d}{s}\right) = -2\left(n\bar{y} - n\left(\overline{\left(\frac{d}{s}\right)^2} w_0 + \overline{\left(\frac{d}{s} t\right)} w_1 + \overline{\left(\frac{d}{s} weather\right)} w_2 + \bar{b} + \varepsilon\right)\right)$$

$$w_0^* = \frac{n\bar{y} - n\overline{\left(\frac{d}{s} t\right)} w_1 - n\overline{\left(\frac{d}{s} weather\right)} w_2 - n\bar{b} - n\varepsilon}{n\overline{\left(\frac{d}{s}\right)^2}}$$

$$\frac{\partial \sum_{i=1}^{n} f}{\partial b} = 0 = \sum_{i=1}^{n} -2 f(1) = -2\left(n\bar{y} - \left(n\overline{\left(\frac{d}{s}\right)} w_0 + n\bar{t} w_1 + n\overline{weather}\ w_2 + n\bar{b} + \varepsilon\right)\right)$$

$$b^* = \frac{n\bar{y} - n\overline{\left(\frac{d}{s}\right)} w_0 - n\bar{t} w_1 - n\overline{weather}\ w_2 - \varepsilon}{n}$$

ب: پس از وارد کردن کتابخانه‌های مورد نیاز، لازم است تا تغییراتی در دیتا به وجود بیاید. از آنجایی که ۴ ستون در دیتاست موجود است اما در سوال از تقسیم دو ستون آن بر یکدیگر به عنوان یک فیچر جدید استفاده شده، در کدهای زیر، تقسیم ستون distance بر speed به دست آمده و به دیتاست اضافه شده و دو ستون گفته شده حذف شده‌اند.
سپس ستون Time_arrival به عنوان برچسب و باقی ستون‌ها به عنوان فیچرها در نظر گرفته شده است. و ۱۵۰۰ دیتای اول از هرکدام به عنوان داده‌های train و ما بقی به عنوان داده‌ی تست در نظر گرفته شده است.

```
df['DPS'] = df['distance']/df['speed']
del df["distance"]
del df["speed"]

y= df['Time_Arrival']
x = df.loc[:, df.columns != 'Time_Arrival']

x_train = x.iloc[:1500]
y_train = y.iloc[:1500]

x_test = x.iloc[1500:]
y_test = y.iloc[1500:]
```

در ادامه، با استفاده از تابع آماده Linear regression ، مدل با داده‌های train، آموزش داده شده و پیش‌بینی آن بر روی داده‌های تست به دست آمده است.

```
reg = LinearRegression().fit(x_train, y_train)
res = reg.predict(np.array( x_test ))
```

مقدار خطا برابر با تفاوت پیش‌بینی مدل با مقدار واقعی برچسب‌ها می‌باشد، بنابراین تفاوت این دو با دو روش MAE و MSE به دست آمده است.

```python
print("errors from linear regression model")
mae = mean_absolute_error(res, y_test.values)
mse = mean_squared_error(res, y_test.values)
print("mae => " + str(mae))
print("mse => " + str(mse))
```

در ادامه، مشابه سوال ۲ یک مدل رگرسیون خطی طراحی شده که توضیح آن مشابه سوال قبلی‌ست و سپس میزان خطا با دو روش MSE و MAE به دست آمده است.

ج: در این بخش، یک مدل رگرسیون خطی که در هر دور آموزش با ۳۰۰ داده‌ی بر خورده‌ی تصادفی آموزش می‌بیند را در یک حلقه‌ی ۱۰۰ تایی آموزش داده‌ایم.

برای هر ویژگی، نمودار وزن‌ها در ۱۰۰ بار آموزش و میانگین و واریانس این ۱۰۰ عدد محاسبه شده است. هرچه واریانس بیشتر باشد، مدل به قطعیتی در مورد وزن فیچر نرسیده است و هرچه میانگین بیشتر باشد یعنی آن فیچر تاثیر زیادی در پیش‌بینی مدل داشته و هرچه به صفر نزدیک‌تر باشد یعنی تاثیر کمتری در پیش‌بینی مدل داشته است.

که در این مساله به ترتیب، فیچر weather condition ، سپس نود distance به رویspeed و در آخر نود traffic تاثیر بیشتری در پیش‌بینی مدل داشته**اند.**

همچنین بایاس فقط برای نود آخر تعریف شده است.

سوال پنج)

الف: احتمال برابر با $\frac{644}{4238}$ خواهد بود.

ب: پاسخ این بخش به‌صورت دستی نوشته شده و عکس آن در ادامه قرار داده می‌شود.

سؤال 5 - ب :

$$P(y_i|P) = P^{y_i}(1-P)^{1-y_i}$$

$$\text{Likelihood}(P) = \prod_{i=1}^{n} P(y_i|P)$$

$$= P^{y_1}(1-P)^{1-y_1} \times P^{y_2}(1-P)^{1-y_2} \times \cdots \times P^{y_n}(1-P)^{1-y_n}$$

$$= P^{\sum_{i=1}^{n} y_i}(1-P)^{n-\sum_{i=1}^{n} y_i}$$

میتوان به جای max کردن likelihood ، log آن را max کرد :

$$\log(\text{Likelihood}(P)) = \left(\sum_{i=1}^{n} y_i\right)\log P + \left(n-\sum_{i=1}^{n} y_i\right)\log(1-P)$$

برای به دست آوردن نقطه بهینه ، مشتق گرفته و مساوی صفر قرار میدهیم :

$$\frac{\partial \log(\text{Likelihood}(P))}{\partial P} = \frac{\sum y_i}{P} - \frac{n-\sum y_i}{1-P} = 0$$

ضرب در

$P(1-P)$

$$: \left(\sum y_i\right)(1-P) - \left(n-\sum y_i\right)P = 0$$

$$\sum y_i - P\sum y_i - nP + P\sum y_i = 0 \rightarrow \sum y_i - nP = 0$$

$$\boxed{\hat{P} = \frac{\sum_{i=1}^{n} y_i}{n}}$$

ج: برای استفاده از دیتاست این سوال، مجموعه‌ی داده‌ها در گوگل درایو قرار داده شده و دو خط زیر برای استفاده از درایو در گوگل کولب و همچنین خواندن فایل مربوطه (heat_disease.csv) از گوگل درایو می‌باشد.

```
drive.mount('/content/drive')
df = pd.read_csv('/content/drive/My Drive/DLAssignments/01/heart_disease.csv')
```

سپس داده‌ها به دو قسمت x و y تقسیم شده‌اند که y همان برچسب‌های ما بر اساس ستون TenYearCHD و x مابقی ستون‌هاست که به عنوان فیچر استفاده خواهند شد.

باتوجه به دیتاست موجود، برخی مقادیر not assigned بودند که مقدار میانگین سایر فیچرها را برای آن فیلد در نظر می‌گیریم:

```
y= df['TenYearCHD']
x = df.loc[:, df.columns != 'TenYearCHD']
x.fillna(x.mean(), inplace=True)
y.fillna(y.mean(), inplace=True)
```

داده‌ها به دو بخش test و train تقسیم شده‌اند که ۴۰۰۰ نمونه‌ی اول به عنوان داده‌ی آموزشی و مابقی به عنوان داده‌ی تست استفاده شده‌اند.

```
x_train = x.iloc[:4000]
y_train = y.iloc[:4000]
x_test = x.iloc[4000:]
y_test = y.iloc[4000:]
```

با استفاده از توابع آماده‌ی موجود در sklearn، یک تابع logistic regression ایجاد کرده و با استفاده از داده‌های train آن را آموزش می‌دهیم.

```
logisticRegr = LogisticRegression()
logisticRegr.fit(x_train, y_train)
```

پیش‌بینی مدل را بر روی داده‌های Test یافته و در ادامه نتایج نمایش داده شده است.

```
predictions = logisticRegr.predict(x_test)
r = logisticRegr.score(x_test, y_test)
print(predictions)
print(y_test)
print(r)
```

از linear regression در مسائل رگرسیون، یعنی پیش‌بینی یک مقدار پیوسته استفاده می‌شود. که حاصل جمع ضرب ویژگی‌ها در وزن هرکدام به اضافه یک بایاس می‌باشد.

اما logistic regression در مسائل classification کاربرد دارد، که همان تابع رگرسیون خطی‌ست که روی آن یک تابع sigmoid اعمال شده است.