میانترم یادگیری ماشین

سید محمد حسینی

۹۸۲۱۲۵۳

سوال ۱:
طبقه‌بند بیز به دلیل اینکه توزیع احتمالاتی رگرسورهای مسئله را ایجاد می‌کند و با استفاده از آنها یک مرز تصمیم‌گیری ایجاد می‌کند بهترین طبقه‌بند ممکن خواهد بود اما باید در نظر داشت که بدست آوردن توزیع احتمالاتی برای یک مسئله با تعداد زیاد ویژگی‌های ورودی می‌تواند دشوار باشد. (درست)
طبقه‌بند بیز به صورت ذاتی مستعد overfit شدن می‌باشد زیرا اگر در هنگام تقسیم بندی دیتاست به زیربخش‌های Train, Test و Validation توزیع این داده‌ها بهم بریزد، رویکرد بیز یک طبقه‌بند با دقت پایین روی زیربخش‌های Test و Validation ایجاد می‌کند. در نتیجه رویکرد بیز بسیار مبتنی بر مجموعه داده آموزش می‌باشد و اگر یک مشکل خاص در این مجموعه داده و تقسیم بندی آن وجود داشته باشد می‌تواند منجر به overfit شدن بشود پس به صورت کلی این جمله نادرست است. (نادرست)
Information Gain به عنوان یک معیار برای انتخاب ویژگی مناسب در درخت تصمیم شناخته می‌شود و

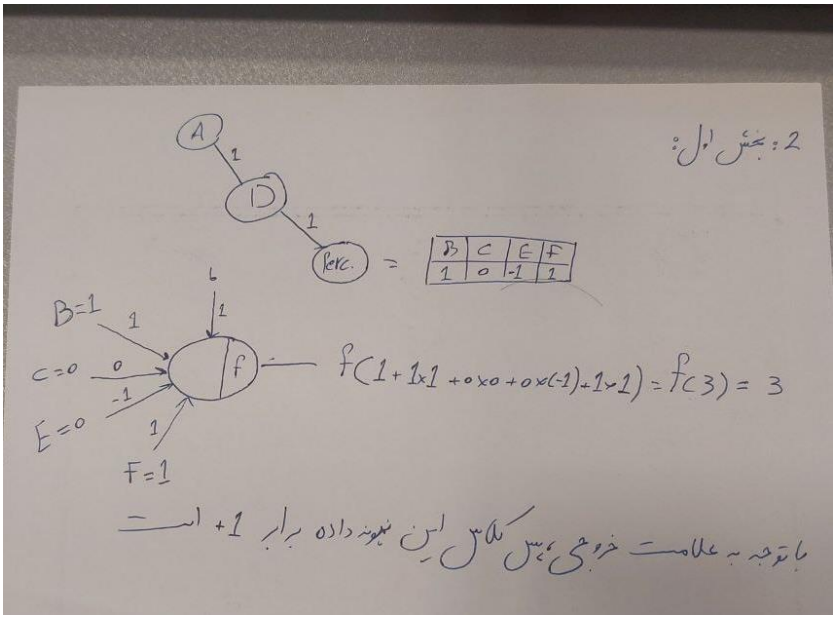
عملکرد این معیار به تعداد حالات ویژگی بستگی ندارد، به بیان دیگر اگر این معیار نشان دهد که یک ویژگی موجب کاهش entropy، بیشتر از دیگر ویژگی‌ها خواهد شد، آن ویژگی انتخاب می‌شود. (نادرست)
این تحلیل درست می‌باشد، به فرمول زیر که نشان دهنده رابطه ریاضی دو لایه است توجه کنید:

$$p(u) = f_2(f_1(uw_1 + b_1)w_2 + b_2)$$

حال اگر توابع f به تابع خطی تغییر حالت بدهند داریم:

$$p(u) = uw_1w_2 + b_1w_2 + b_2 \rightarrow p(u) = uW + B$$

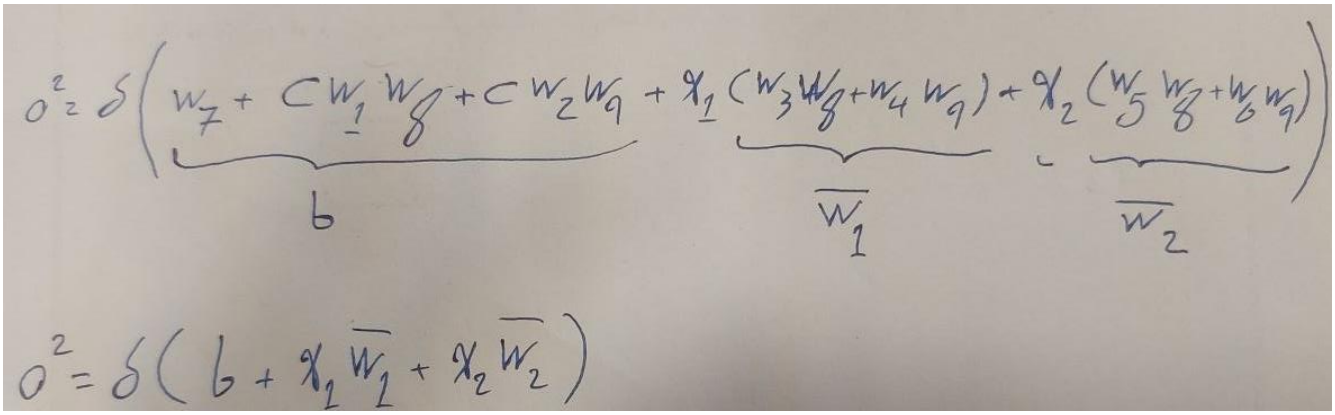
(درست)
سوال ۲:



مرز تصمیم هواره خطی نخواهد بود زیرا درخت تصمیم در گام نخست یک روش است که می‌تواند منجر به ایجاد یک طبقه‌بند با مرز تصمیم غیرخطی شود که این مسئله روی لایه پرسپترون هم تاثیر خواهد گذاشت. به بیان دیگر عملکرد سلسله مراتبی این مدل منجر به این خواهد شد تا مرز تصمیم خطی نباشد.

در ادامه به منظور مقایسه درخت تصمیم با درخت پرسپترون به ازای مقادیر کوچک عمق، باید به این مسئله توجه داشت که اگر مقدار عمق این دو درخت با یکدیگر برابر باشد، می‌توان نتیجه گرفت که عملکرد

خروجی از هر دوی درخت‌ها یکسان است اما با اضافه کردن یک لایه پرسپترون به انتهای درخت تصمیم، یک طبقه‌بند ثانویه به صورت سلسله‌مراتبی ایجاد خواهیم کرد که می‌تواند منجر به افزایش عملکرد فرایند طبقه‌بندی شود زیرا یک درخت تصمیم با لایه‌های کم با احتمال پایینی دچار overfitting شده است و اضافه



کردن یک طبقه‌بند دیگر به انتهای آن می‌تواند منجر به افزایش دقت شود.

سوال۳:

$$\begin{cases} o^2 = \sigma(W_7 + o_1^1 W_8 + o_2^1 W_9) \\ o_1^1 = c(W_1 + x_1 W_3 + x_2 W_5) \\ o_2^1 = c(W_2 + x_1 W_4 + x_2 W_6) \end{cases}$$

$$\rightarrow P(Y = 1|X, W) = o^2$$
$$= \sigma(W_7 + c(W_1 + x_1 W_3 + x_2 W_5)W_8 + c(W_2 + x_1 W_4 + x_2 W_6)W_9)$$

امکان ایجاد یک شبکه عصبی بدون لایه پنهان وجود دارد که به شکل زیر قابل انجام است:

سوال۴:

در این قسمت با ضریب ۰/۲ دیتا به دو قسمت train و val تقسیم شده است تا بتوانیم هنگام آموزش از overfitشدن جلوگیری کنیم.

با استفاده از کد زیر ویژگی‌ها استخراج خواهد شد:

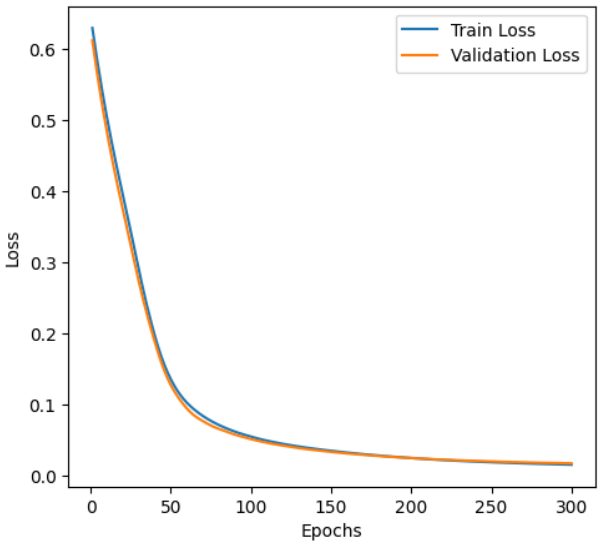
```
# Standard Deviation
features.append(np.std(data, axis=1))

# Peak
features.append(np.max(np.abs(data), axis=1))

# Crest Factor (peak divided by RMS)
rms = np.sqrt(np.mean(np.square(data), axis=1))
features.append(features[1] / rms)

# Clearance Factor (peak divided by the mean of the square root of the absolute values)
features.append(features[1] / np.mean(np.sqrt(np.abs(data)), axis=1))

# Peak to Peak
features.append(np.ptp(data, axis=1))

# Shape Factor (RMS divided by the mean of the absolute values)
features.append(rms / np.mean(np.abs(data), axis=1))

# Impact Factor (peak divided by mean)
features.append(features[1] / np.mean(data, axis=1))

# Square Mean Root (the square root of the mean of the squares)
features.append(rms)

# Mean
features.append(np.mean(data, axis=1))
```

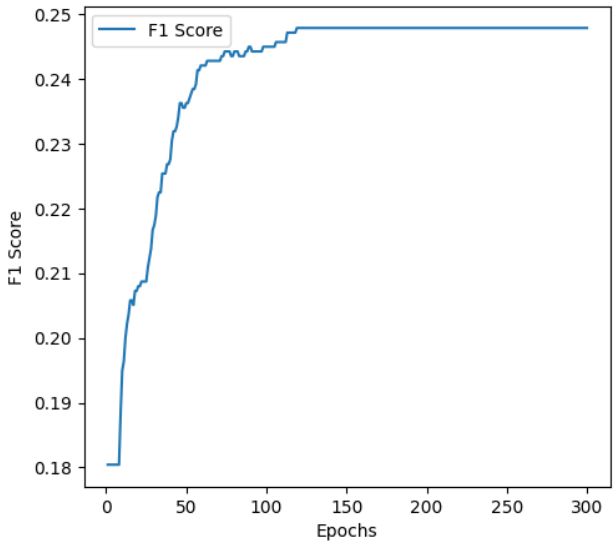
```python
class MLP(nn.Module):
    def __init__(self, input_dim, output_dim):
        super(MLP, self).__init__()
        self.hidden1 = nn.Linear(input_dim, 32)
        self.hidden2 = nn.Linear(32, 64)
        self.output = nn.Linear(64, output_dim)
        self.relu = nn.ReLU()
        self.softmax = nn.Softmax(dim=1)

    def forward(self, x):
        x = self.relu(self.hidden1(x))
        x = self.relu(self.hidden2(x))
        x = self.output(x)
        x = self.softmax(x)
        return x
```

مدل یک MLP ساده است که متشکل از ۲ لایه پنهان با ابعداد ۳۲ و ۶۴ می‌باشد که تابع فعالسازی آن ReLU است و لایه آخر به علت چند کلاسه بودن از softmax استفاده می‌کند.

```python
optimizer = optim.Adam(model.parameters(), lr=0.001)
criterion = nn.BCELoss()
scheduler = optim.lr_scheduler.ReduceLROnPlateau(optimizer, 'min', patience=3)
```

در ادامه از بهینه ساز Adam با گام آموزش ۰/۰۰۱ و تابع هزینه BCE استفاده شده. همانطور که مشخص است از بهینه‌ساز گام آموزش نیز استفاده شده است. در ادامه این فرایند از early_stop استفاده شده است.



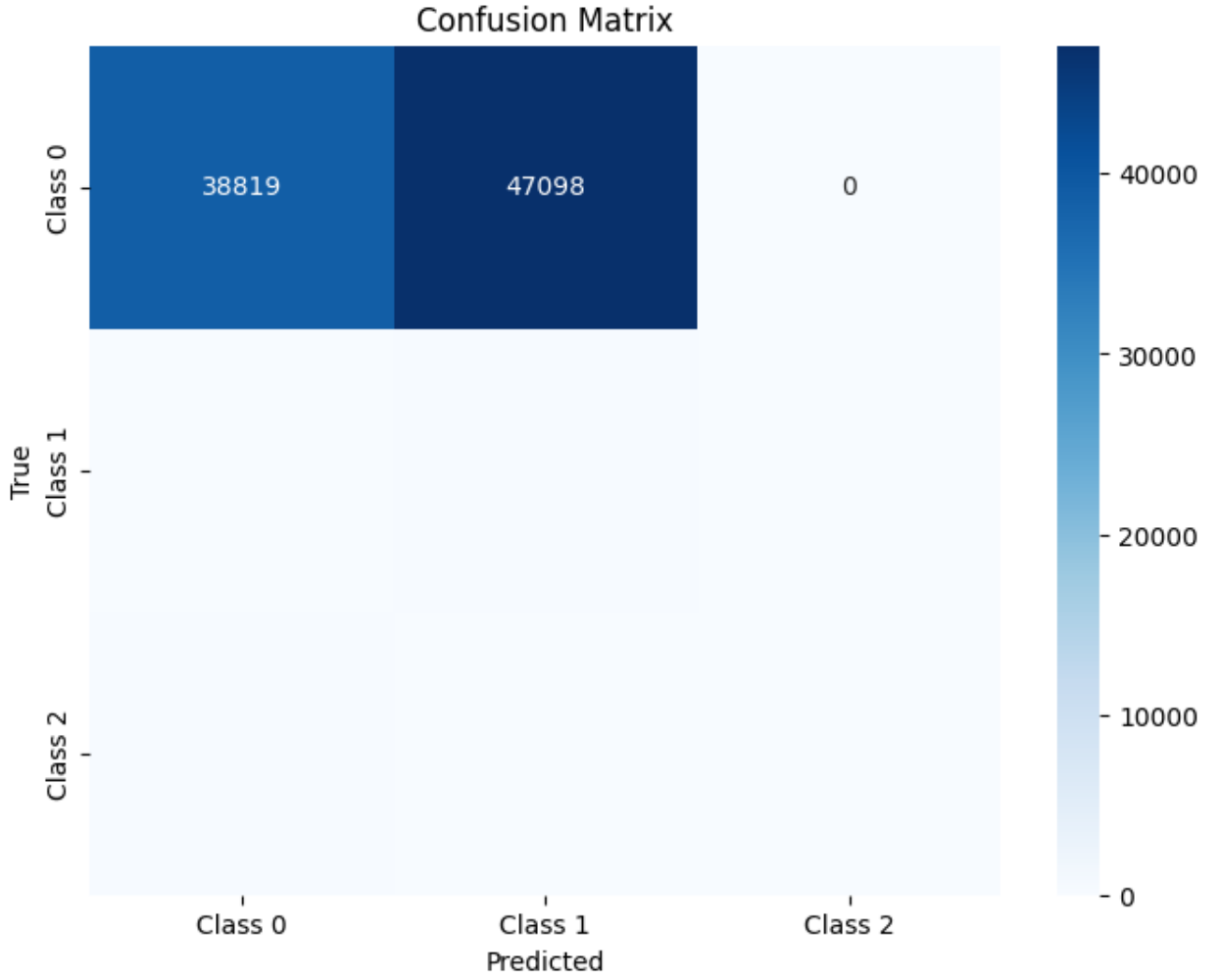


نمودار فوق مربوط به فرایند آموزش می‌باشد. همانطور که مشخص است نمودار تابع هزینه با روند مناسبی کاهشی بوده و early stop نشده است. از سوی دیگر مشخص است که نتیجه f-1 score مناسب نیست که این مسئله بیشتر از جهت imbalance بودن دیتا بوده است که به همین دلیل دیتاهای سالم را کم می‌کنیم تا فرایند آموزش بهتر انجام پذیرد.

Classification Report:

	precision	recall	f1-score	support
Class 0	0.99	0.45	0.62	85917
Class 1	0.01	1.00	0.01	276
Class 2	0.00	0.00	0.00	207
accuracy			0.45	86400
macro avg	0.33	0.48	0.21	86400

weighted avg	0.99	0.45	0.62	8640



گزارش فوق نشان می‌دهد که عملکرد مدل روی دیتاست تست مناسب نیست و کلاس ۰ که همانکلاس سالم است را با f-1 score ۶۱ درصد پیدا کرده است که به معنی عملکرد ضعیف مدل در این کلاس است اما درمورد کلاس‌های خطا نتایج بدتر نیز شده است و تقریبا به جز تعداد اندکی از آنها، باقی موارد را پیدا نکرده است.

به منظور رفع این مشکل باید از روش پنجره استفاده کنیم تا با در نظر گرفتن یک بازه از دیتا بتوانیم نتایج را محاسبه کنیم زیرا نتایج نشان می‌دهد که این رویکرد مناسب نبوده است.