**به نام خدا**

**دانشگاه صنعتی امیرکبیر**

**دانشکده مهندسی کامپیوتر**

# پاسخ تمرین سری اول شبکه‌های عصبی

**استاد:**

**دکتر صفابخش**

**دانشجو:**

**حلیمه رحیمی**

**شماره دانشجویی:**

**۹۹۱۳۱۰۴۳**

**بهار ۱۴۰۰**

۱- واحد عصبی پرسپترون به عنوان یک جمع کننده‌ی وزنی عمل می‌کند؛ به این صورت که برای هر یک از ویژگی‌ها یک وزن در نظر گرفته می‌شود، همچنین یک مقدار بایاس به دلیل آنکه حتما نیاز نیست مرز جداکننده‌ی دو کلاس از مبدا عبور کند تعیین می‌شود. در هر بار مقدار هدف و مقادیر ویژگی‌های داده های ورودی را برای آموزش به این واحد می‌دهیم و اگر واحد دچار خطا شود، وزن‌ها را به‌روز می‌کنیم. در صورتی که داده‌ها به صورت خطی جدایی پذیر باشند این به‌روزرسانی تا زمانی که دیگر هیچ خطایی وجود نداشته باشد ادامه می‌یابد.

به‌روزرسانی با فرمول زیر برای وزن‌ها و بایاس انجام می‌گیرد. البته می‌توان بایاس را وزن یک ویژگی ورودی همیشه ۱ دانست و همراه با وزن ها به روز کرد.

$$w_{new} = w_{old} + (t - y)x$$

$$b_{new} = b_{old} + (t - y)$$

در اینجا t مقدار هدف و y خروجی واحد پرسپترون است.

$$y = f(\sum_{i=1}^{n} w_i x_i + b)$$

این خروجی از طریق دادن مقدار محاسبه شده به یک تابع فعالسازی f حاصل می‌شود. در پرسپترون تابع فعالسازی برای مقادیر مثبت، مقدار یک و برای مقادیر منفی، مقدار صفر را برمی‌گرداند.

البته می‌توان این الگوریتم را با مقادیر هدف ۱+ و ۱- نیز اجرا کرد، در این حالت با رخداد خطا در تشخیص، مقادیر وزن‌ها و بایاس به شکل زیر به‌روز می‌شود در غیر این صورت هیچ عملی صورت نمی‌گیرد. تابع فعالسازی نیز به جای صفر، ۱- برمی‌گرداند.

$$w_{new} = w_{old} + tx$$

$$b_{new} = b_{old} + t$$

با توجه به آنچه بیان شد شرط توقف را می‌توان تغییر نکردن وزن‌ها در نظر گرفت.

واحد عصبی آدلاین سعی در کاهش مربع خطا دارد. با توجه به اینکه می‌توان مقدار کمینه را برای یک تابع از طریق برابر قرار دادن مشتق آن با صفر به دست آورد، در اینجا نیز می‌توان از آن بهره برد. از آنجایی که محاسبات برای مسائل بزرگ سنگین خواهد بود، پیشنهاد شد که به جای این کار از کاهش گرادیان استفاده شود. به این صورت که وزن ها خطا را در جهت بیشترین کاهش گرادیان سوق دهند.

$$w_{new} = w_{old} + \alpha\delta x$$

این $\delta$ به شکل $(t - I)$ به دست می‌آید که I همان مقدار $\sum_{i=1}^{n} w_i x_i + b$ بدون عبور از تابع فعالسازی است. $\alpha$ نیز درجه‌ی یادگیری است که می‌توان با تنظیم آن از کم و زیاد شدن ناگهانی مقدار بالایی از هر یک از وزن‌ها جلوگیری کرد.

در حالت Batch میانگین مطرح خواهد بود.

تفاوت پرسپترون و آدلاین در این است که تابع فعالسازی در هنگام یادگیری برای آدلاین یک ماتریس همانی است. به عبارتی مقدار همانی که به دست آمده باقی می‌ماند و صفر یا یک (یا ۱+ و ۱- با توجه به تعریف مسئله) نمی‌شود. این مسئله باعث می‌شود یادگیری حتی پس از آنکه تمامی داده‌ها به درستی تشخیص داده شدند ادامه پیدا کند و وزن ها اگر I دقیقا برابر با مقدار t نشده به روز می‌شود. شرط توقف الگوریتم را می‌توان تغییر بسیار کم وزن‌ها در نظر گرفت. همچنین آدلاین سعی در کاهش MSE دارد در حالیکه پرسپترون در پی صفر کردن خطای تشخیص است. این مسئله در برابر وجود داده‌های پرت حائز اهمیت است؛ چرا که آدلاین در پی کاهش MSE ممکن است از مرزی که کاملا داده ها را جدا می‌کند ولی مقدار MSE بیشتری دارد دوری کند. البته پرسپترون چون در پی خطای صفر برای مجموعه آموزش است قابلیت عمومیت بخشی کمتری دارد بعلاوه اینکه اگر به خطای صفر نرسد متوقف نمی‌شود.

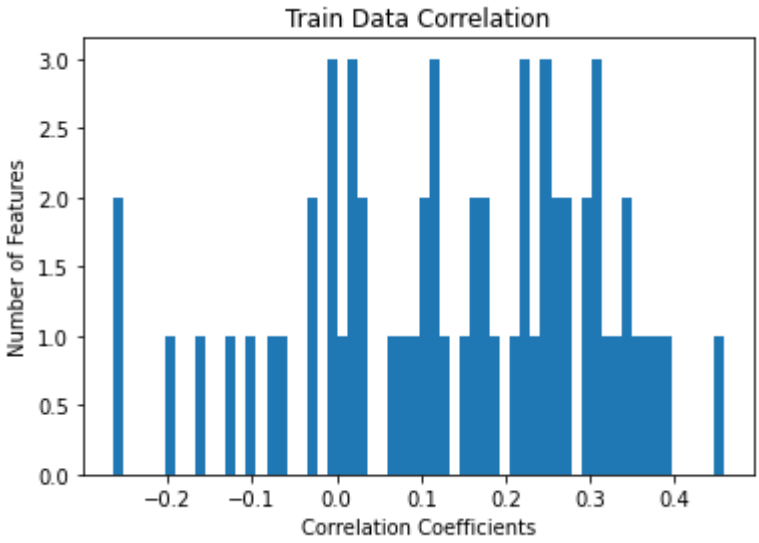
۲- در اینجا مجموعه داده را طبق خود سوال ۷۰ آموزش، ۱۰ اعتبارسنجی و باقی را آزمون قرار دادم (در متن سوالات در ابتدا نوشته شده بود ۷۰، ۲۰، ۱۰ داده ها را جدا کنیم، سپس در خود سوال نوشته شده بود ۷۰، ۱۰، ۲۰) . ترشولد را برابر با ۰/۲ در نظر گرفتم (با توجه به اینکه ضرایب تا ۰/۴ برای همه‌ی مجموعه‌ها ادامه دارد، به نظرم معقول رسید نصف آن را ترشولد قرار دهم). همانطور که مشاهده می‌شود در هر سه مجموعه ویژگی، ویژگی‌های مشترکی وجود دارد. این ویژگی‌های مشترک در تشخیص کلاس هر سه مجموعه داده کمک کننده خواهند بود (بسته به میزان همبستگی در هر مجموعه) اما آن ویژگی‌هایی که به طور مثال در مجموعه‌داده‌ی آموزش با کلاس‌ها همبستگی بیشتری دارند درحالیکه در بین ویژگی‌های منتخب مجموعه‌داده‌ی اعتبارسنجی نیستند، در صورت انتخاب به اندازه‌ای که در تشخیص کلاس‌ها برای مجموعه آموزش کمک کننده‌اند، در برابر تشخیص کلاس‌های مجموعه اعتبارسنجی یارا نخواهند بود. بنابراین با انتخاب آنها ممکن است دقت برای مجموعه آموزش بسیار بهتر از مجموعه اعتبارسنجی شود.

مجموعه ۲۸ ویژگی برای مجموعه آموزش، ۲۷ ویژگی برای مجموعه اعتبارسنجی و ۲۲ ویژگی برای مجموعه آزمون، ضریب همبستگی بالای ۰/۲ یا کمتر از ۰/۲- دارند که در تصاویر زیر مشخص شده‌اند.

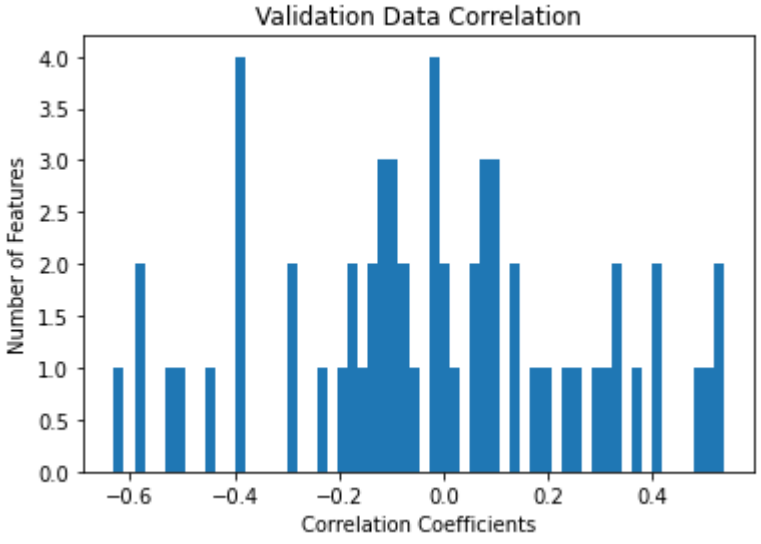
از بین این ویژگی‌ها، ویژگی‌های مشترک شامل ۹ ویژگی {x9, x10, x11, x35, x44, x45, x46, x50, x57} حتما باید انتخاب شود.

با توجه به اینکه تعداد ویژگی هایی که بین ۰,۲ و ۰,۲- هستند برای داده های آزمون بیشتر است و برای اعتبارسنجی یکی بیشتر از آموزش، به نظر می رسد ممکن است چنین چیزی به علت انتخاب داده‌هایی برای آزمون باشد که ویژگی‌ها برای تشخیص کلاس به اندازه داده‌های آموزش و اعتبارسنجی مناسب نباشند. بنابراین انتظار می‌رود در صورت نگه داشتن همه‌ی ویژگی‌ها نتایج برای آزمون به خوبی دو مجموعه‌ی دیگر نباشد و در صورت انتخاب تنها ویژگی‌های مشترک انتظار می‌رود خطا برای همه‌ی مجموعه‌ها نزدیک به یکدیگر باشد.

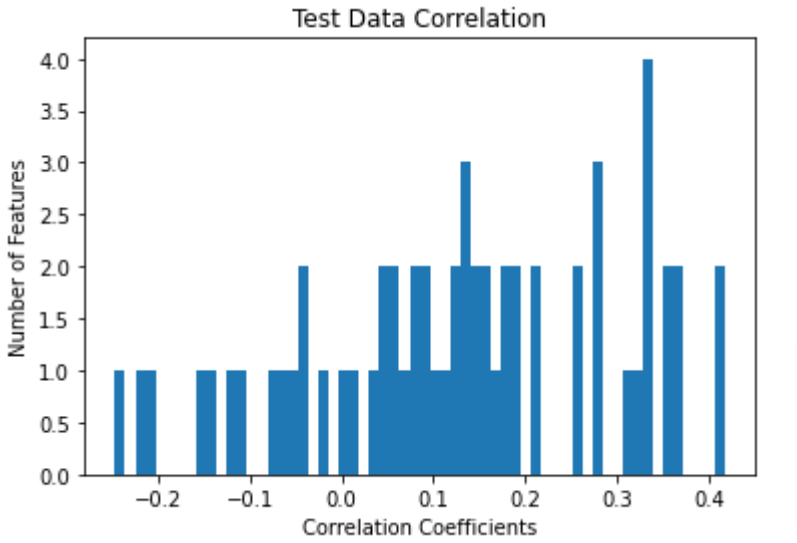
| | |
|---|---|
| x0 | 0.331836 |
| x1 | 0.273755 |
| x2 | 0.245854 |
| x3 | 0.298598 |
| x4 | 0.263876 |
| x7 | 0.221507 |
| x8 | 0.357528 |
| x9 | 0.370877 |
| x10 | 0.457613 |
| x11 | 0.396940 |
| x12 | 0.319479 |
| x19 | 0.210487 |
| x20 | 0.266403 |
| x21 | 0.251105 |
| x35 | -0.263542 |
| x36 | -0.261576 |
| x42 | 0.229579 |
| x43 | 0.262662 |
| x44 | 0.347952 |
| x45 | 0.309246 |
| x46 | 0.297362 |
| x47 | 0.338154 |
| x48 | 0.380421 |
| x49 | 0.223052 |
| x50 | 0.308593 |
| x51 | 0.311574 |
| x53 | 0.225575 |
| x57 | 0.248338 |




| | |
|---|---|
| x8 | 0.336849 |
| x9 | 0.262423 |
| x10 | 0.414491 |
| x11 | 0.535161 |
| x12 | 0.513774 |
| x13 | 0.203541 |
| x24 | 0.242858 |
| x25 | 0.401293 |
| x26 | 0.535366 |
| x27 | 0.496150 |
| x28 | 0.373224 |
| x33 | -0.280682 |
| x34 | -0.580183 |
| x35 | -0.590455 |
| x36 | -0.232405 |
| x39 | -0.505942 |
| x40 | -0.436518 |
| x44 | 0.314728 |
| x45 | 0.335659 |
| x46 | 0.287867 |
| x50 | -0.382157 |
| x53 | -0.394173 |
| x54 | -0.287575 |
| x56 | -0.521418 |
| x57 | -0.630171 |
| x58 | -0.379355 |
| x59 | -0.380423 |



| | |
|---|---|
| x9 | 0.212922 |
| x10 | 0.331473 |
| x11 | 0.318697 |
| x18 | 0.338865 |
| x19 | 0.370609 |
| x20 | 0.252709 |
| x24 | -0.246822 |
| x34 | -0.219199 |
| x35 | -0.205234 |
| x38 | 0.258698 |
| x42 | 0.330200 |
| x43 | 0.407431 |
| x44 | 0.309782 |
| x45 | 0.283060 |
| x46 | 0.333247 |
| x47 | 0.355805 |
| x48 | 0.357092 |
| x50 | 0.416928 |
| x51 | 0.278000 |
| x52 | 0.365919 |
| x57 | 0.273904 |
| x58 | 0.214245 |




۳- برای اطمینان از اینکه آموزش پرسپترون متوقف شود، تصمیم گرفتم علاوه بر شرط همگرایی، شرط حداکثر ۴۰۰۰ تکرار را نیز بگذارم. مشهود است که با ۲۳۳۶ تکرار همگرایی رخ داده و به علت آنکه در هر بار بروزرسانی مقدار نسبتا بزرگی (با توجه به مقادیر خود ویژگی‌ها) از وزن‌ها کم می‌شود، ممکن است پرسپترون درباره برخی از نمونه‌هایی که قبلا صحیح تشخیص داده بود، اشتباه کند و نمودار مشابه اینجا دچار نوسانات بسیار زیادی شود. البته به طور کلی خطا نزولی بوده و در انتها به صفر می‌رسد.

همانطور که پیش از این ذکر شد پرسپترون توجهی به MSE ندارد بنابراین احتمال آن وجود دارد که به طور مثال برای یکی از داده‌ها مقدار بسیار بزرگی را پیشبینی کند در حالیکه داده از کلاس ۱ است و پرسپترون نیز به درستی در نهایت آن را ۱ در نظر می‌گیرد. واضح است که در صورت رخداد این مسئله مقدار MSE بیشتر و بیشتر شود.

ماتریس در هم ریختگی و مقدار خطا را برای هر یک از مجموعه داده ها آورده ام. ردیف اول در ماتریس برای کلاس منفی و ردیف دوم برای کلاس مثبت می باشد.

برای پیاده سازی از $(t - y)$ برای بروزرسانی وزن‌ها استفاده کردم و کلاس‌ها را صفر و یک در نظر گرفتم. بنابراین محاسبه خروجی و فعالسازی به شکل زیر خواهد بود:

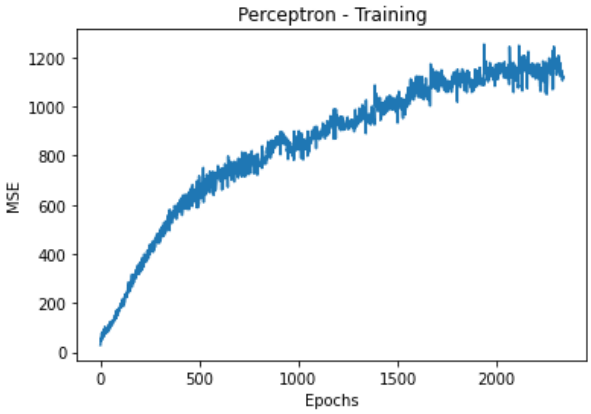
```
def predict(s, w, b):
    pred = np.dot(s, w) + b
    pred = [1 if pred>0 else 0]
    return pred
```

از دو تابع، یکی برای تکرار اجرا و بررسی همگرایی و دیگری برای بروزرسانی وزن استفاده کردم.

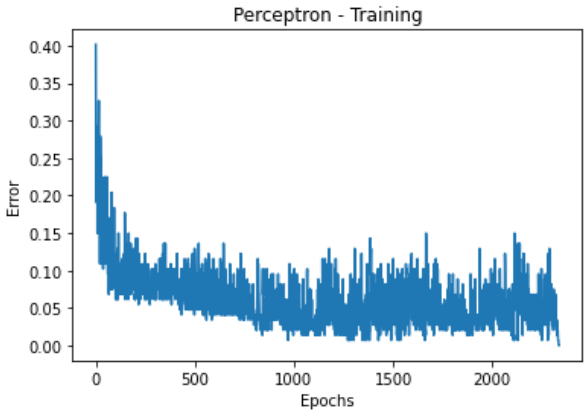
بایاس و وزن‌های اولیه را نیز برابر با صفر قرار دادم. برای دقت و ماتریس در هم ریختگی نیز از کتابخانه sklearn استفاده کردم.

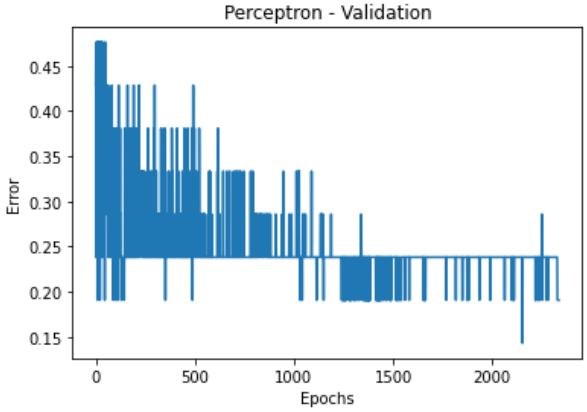
```
def training(xtrain, ytrain, weights, bias, learn_rate):
    train_pred = []
    for i in range(xtrain.shape[0]):
        s_pred = predict(xtrain.iloc[i], weights, bias)
        error = ytrain.iloc[i]-s_pred
        weights += learn_rate*error*xtrain.iloc[i]
        bias += learn_rate*error
    train_pred = np.dot(xtrain, weights)+bias
    mse = (1/xtrain.shape[0])*(sum((ytrain-train_pred)**2))
    train_pred[train_pred>0] = 1
    train_pred[train_pred<=0] = 0
    tr_error = 1-accuracy_score(ytrain, train_pred)
    tr_mat = confusion_matrix(ytrain, train_pred)
    return weights, bias, mse, tr_error, tr_mat
```

```
def train_val(xtrain, ytrain, xval, yval, learn_rate):
    mse_error = []
    train_error = []
    val_error = []
    weights = np.zeros(xtrain.shape[1])
    bias = 0
    t= True
    num_iter = 0
    while t:
        num_iter += 1
        old_weights = np.copy(weights)
        old_bias = np.copy(bias)
        weights, bias, mse, tr_error, train_mat = training(xtrain, ytrain, weights, bias, learn_rate)
        mse_error.append(mse)
        train_error.append(tr_error)
        val_pred = []
        for j in range(xval.shape[0]):
            val_pred.append(predict(xval.iloc[j], weights, bias))
        v_error = 1-accuracy_score(yval, val_pred)
        val_error.append(v_error)
        if num_iter==4000 or (all(weights==old_weights) and bias==old_bias):
            t = False
            val_mat = confusion_matrix(yval, val_pred)
    return weights, bias, mse_error, train_error, val_error, num_iter, train_mat, val_mat
```




```
Number of Iterations:  2336
Perceptron MSE Error:  1122.2908705920277
Perceptron Train Error:
 0.0
Perceptron Train Confusion Matrix:
 [[68  0]
 [ 0 79]]
Perceptron Validation Error:
 0.19047619047619047
Perceptron Validation Confusion Matrix:
 [[9 1]
 [3 8]]
Perceptron Test Error:
 0.38095238095238093
Perceptron Test Confusion Matrix:
 [[15  6]
 [10 11]]
```



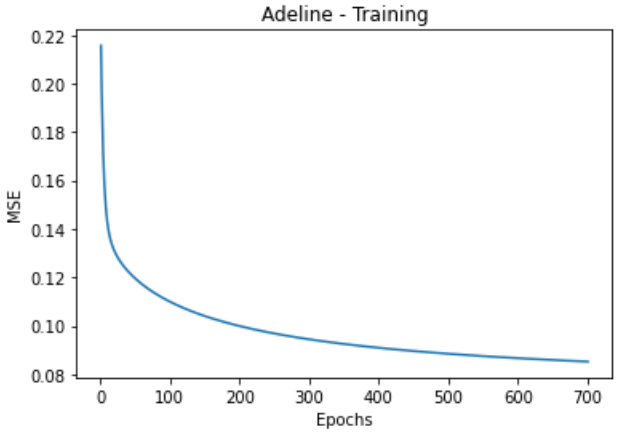




۴- در اینجا آدلاین با ۷۰۰ تکرار به همگرایی می رسد. البته شرط همگرایی را کمتر بودن بزرگترین تغییر در وزن‌ها از ۰/۰۰۱ در نظر گرفتم. نمودار میانگین مربع خطاها، نمودار خطای آموزش و اعتبارسنجی را در زیر می‌بینید. همانطور که مشهود است خطای آموزش به غیر از چند دور کوچک کاهشی بوده است و خطای اعتبارسنجی پس از کاهش دوباره افزایش یافته است. این مسئله احتمالا به این علت است که با آنکه تا چندین دور، خطای آموزش تغییری نداشته، خطای MSE باعث شده الگوریتم به بروزرسانی وزن‌ها ادامه دهد. آدلاین از آن جهت که در پی کاهش میانگین مربع خطاست، ممکن است در صورت وجود داده‌ی پرت، مرز را به

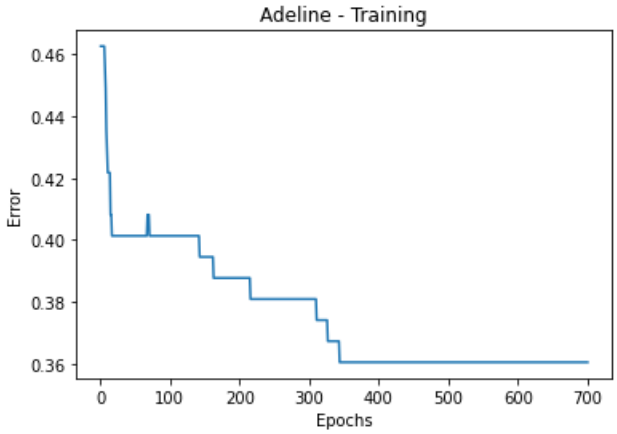
گونه‌ای تعیین کند که برخی داده‌ها به درستی تشخیص داده نشوند ولی MSE کمتر شود. مشاهده می‌کنید که در اینجا خطای آدلاین بسیار بیشتر از خطای پرسپترون شده بنابراین احتمال دارد چنین مسئله ای صحیح باشد. مقدار مرز همگرایی را کاهش دادم اما پاسخ همچنان تقریبا مشابه بود.

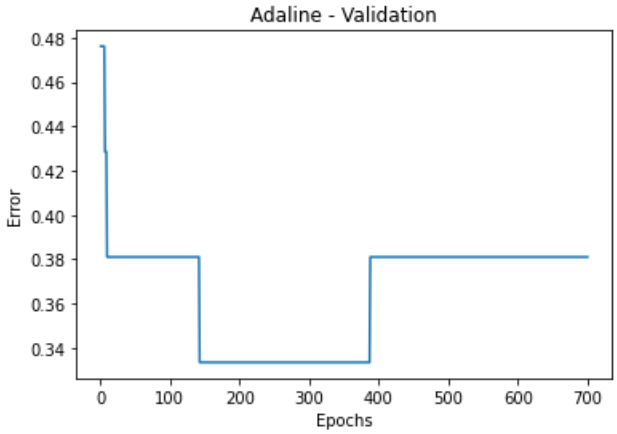
```
def training(xtrain, ytrain, weights, bias, learn_rate):
    train_pred = np.zeros(xtrain.shape[0])
    for i in range(xtrain.shape[0]):
        s_pred = np.dot(xtrain.iloc[i], weights) + bias
        error = ytrain.iloc[i]-s_pred
        weights += learn_rate*error*xtrain.iloc[i]
        bias += learn_rate*error
    train_pred = np.dot(xtrain, weights)+bias
    mse = (1/xtrain.shape[0])*(sum((ytrain-train_pred)**2))
    train_pred[train_pred>0] = 1
    train_pred[train_pred<=0] = 0
    tr_error = 1-accuracy_score(ytrain, train_pred)
    tr_mat = confusion_matrix(ytrain, train_pred)
    return weights, bias, mse, tr_error, tr_mat
```

```
def train_val(xtrain, ytrain, xval, yval, learn_rate):
    train_error = []
    mse_error = []
    val_error = []
    #weights = np.random.random_sample(size=xtrain.shape[1])
    weights = np.zeros(xtrain.shape[1])
    bias = 0
    t= True
    num_iter = 0
    while t:
        num_iter += 1
        old_weights = np.copy(weights)
        old_bias = np.copy(bias)
        weights, bias, mse, tr_error, train_mat = training(xtrain, ytrain, weights, bias, learn_rate)
        mse_error.append(mse)
        train_error.append(tr_error)
        val_pred = []
        for j in range(xval.shape[0]):
            val_pred.append(predict(xval.iloc[j], weights, bias))
        v_error = 1-accuracy_score(yval, val_pred)
        val_error.append(v_error)
        if num_iter==4000 or (np.max(abs(weights-old_weights))<=0.001):
            t = False
            val_mat = confusion_matrix(yval, val_pred)
    return weights, bias, mse_error, train_error, val_error, num_iter, train_mat, val_mat
```



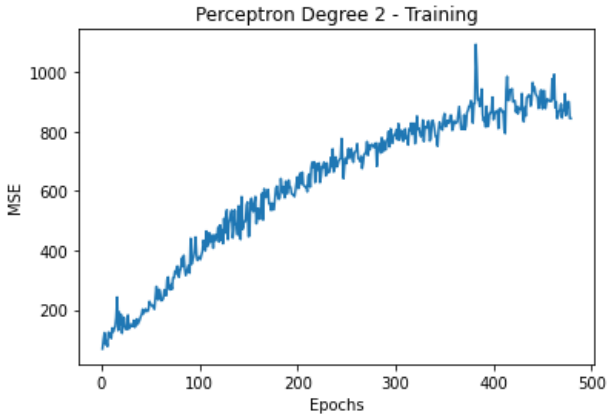

```
Number of Iteration:  700
Adaline MSE Error:
 0.08523733162303454
Adaline Train Error:
 0.3605442176870748
Adaline Train Confusion Matrix:
 [[15 53]
 [ 0 79]]
Adaline Validation Error:
 0.38095238095238093
Adaline Validation Confusion Matrix:
 [[ 2  8]
 [ 0 11]]
Adaline Test Error:
 0.47619047619047616
Adaline Test Confusion Matrix:
 [[ 1 20]
 [ 0 21]]
```



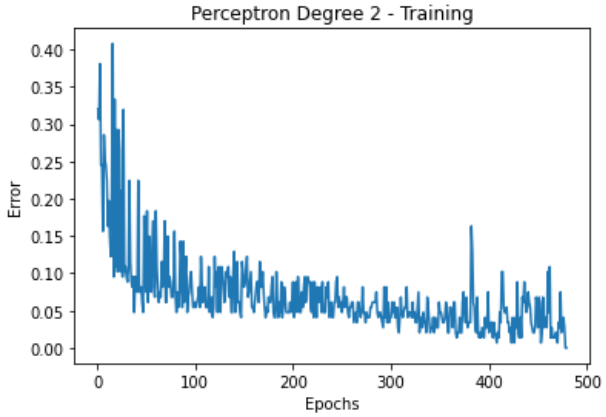



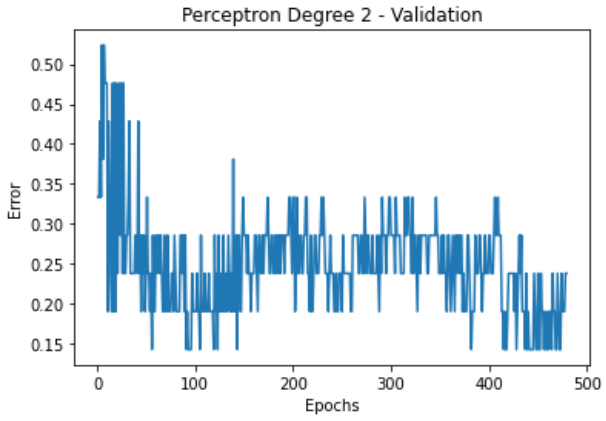

۵- برای درجه‌ی دو کافیست ترکیب درجه دومی از ورودی‌ها را به پرسپترون یا آدلاین بدهیم. این ترکیب درجه‌ی دوم در نهایت پس از آموزش معادله‌ی یک سهمی، هذلولی، دایره و... می‌تواند باشد. بعلاوه می‌توان ضرب هر دو ویژگی را نیز به عنوان ورودی اضافه کرد که موجب چرخش شکل حاصل خواهد شد و حالت کلی تر آن است. در اینجا از توان دوم مقادیر ویژگی‌ها و خود مقادیر استفاده

کردم؛ چرا که ۶۰ ویژگی در ۶۰ ویژگی بعلاوه‌ی خود آن ویژگی‌ها تعداد ۱۸۶۰ (به علت تکراری بودن برخی ضرب ها) ورودی و در نتیجه وزن را حاصل می‌داد که با تعداد نمونه‌های داده شده معقول نیست.



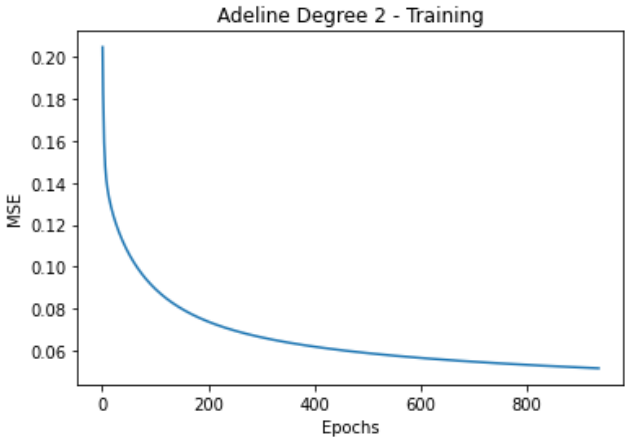

```
Number of Iterations:  479
Perceptron MSE Error:  843.6947105723683
Perceptron Train Error:
 0.0
Perceptron Train Confusion Matrix:
 [[68  0]
 [ 0 79]]
Perceptron Validation Error:
 0.23809523809523814
Perceptron Validation Confusion Matrix:
 [[9 1]
 [4 7]]
Test Error:
 0.2857142857142857
Test Confusion Matrix:
 [[16  5]
 [ 7 14]]
```




درجه دوم برای پرسپترون با سرعت بسیار بیشتری به همگرایی رسید اما پاسخ خطا برای اعتبارسنجی به خوبی قبل نشد. خطا برای آزمون نیز بهتر از قبل است. نشاندهنده‌ی این است که داده‌ها به گونه‌ای توزیع یافته‌اند که می‌توان خیلی سریعتر با یک خط درجه دوم آنها را از یکدیگر جدا کرد تا با یک خط صاف و مستقیم.

MSE و خطای آموزش برای آدلاین کمتر از حالت قبل شده اما خطای اعتبارسنجی بالاتر رفته است. خطای آزمون نیز مشابه پرسپترون کاهش یافته است. البته گاهی ممکن است به علت ادامه دادن بروزرسانی وزن، مدل در برابر برخی از داده‌هایی که قبلا به درستی تشخیص داده شده بودند، دچار اشتباه شود که در اینجا نیز چنین اتفاقی افتاده است. تعداد تکرار کمی بیشتر از حالت قبل شده که نشان می‌دهد ورودی‌های درجه دوم جدید در تشخیص کلاس‌ها اثری گذاشته اند که خود باعث شده وزن‌ها به شکل قبل بروز نشوند. قاعدتا این مسئله تعداد تکرار را کمتر یا بیشتر می‌کند و ارتباط مستقیمی وجود ندارد.




```
Number of Iteration:  935
Adaline MSE Error:
 0.051566533392987945
Adaline Train Error:
 0.34013605442176875
Adaline Train Confusion Matrix:
 [[18 50]
 [ 0 79]]
Adaline Validation Error:
 0.4285714285714286
Adaline Validation Confusion Matrix:
 [[ 1  9]
 [ 0 11]]
Test Error:
 0.4285714285714286
Test Confusion Matrix:
 [[ 4 17]
 [ 1 20]]
```