فاطمه توکلی_۴۰۰۱۳۱۰۱۶

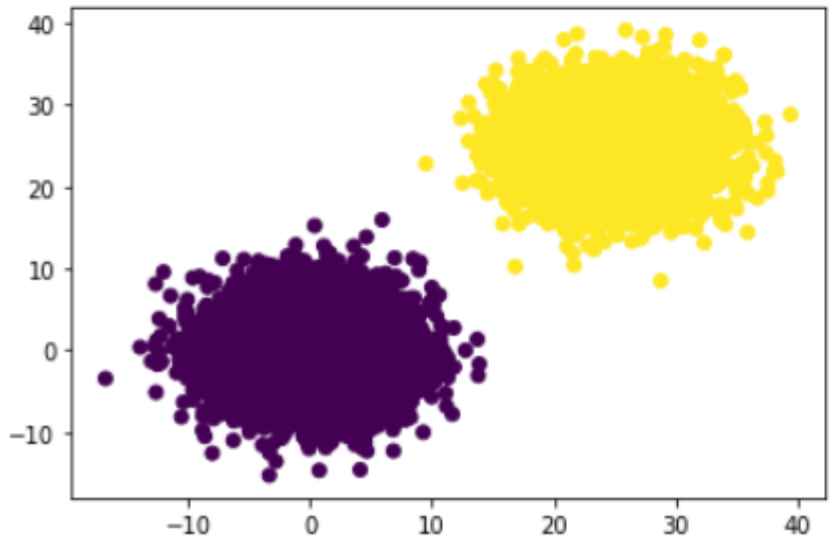
تمرین دوم شبکه عصبی

۱)

با استفاده ازrandom.normal به تولید دو بعد داده با توزیع گاوسی (میانگین متفاوت و کواریانس یکسان) می پردازیم که هرکدام شامل ۵۰۰۰ داده و بعد از پیوستن دو مجموعه در نهایت ۱۰۰۰۰ داده داریم و سپس با استفاده از df.sample به شافل کرده دیتاست میپردازیم و در نهایت داده را سم میک

```
#make 2D dataset with guassian distribution(diff mean,same cov)
x1= random.normal(loc=0, scale=4, size=(5000,2))
x2=random.normal(loc=25, scale=4, size=(5000,2))

first_dim = {'x1':x1[:,0],'x2':x1[:,1],'class':0}
f_d = pd.DataFrame(first_dim)
second_dim = {'x1':x2[:,0],'x2':x2[:,1],'class':1}
s_d = pd.DataFrame(second_dim)
ds = [f_d,s_d]
ds= pd.concat(ds)

#shuffle data
ds = ds.sample(frac=1).reset_index()
ds.drop('index',axis=1, inplace=True)

#plot data
plt.scatter(ds['x1'],ds['x2'],c= ds['class'] )
```

```
<matplotlib.collections.PathCollection at 0x7fa0f8d376d0>
```



داده را به سه مجموعه اموزش و آزمون و اعتبار سنجی با درصد گفته شده تقسیم میکنیم:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
X_train, X_validation, y_train, y_validation = train_test_split(X_train, y_train, test_size = 0.1, random_state = 0)
X_train.shape
```

```
(720, 2)
```

```
y_train.shape
```

```
(720,)
```

۲و۳) در مدل پرسپترون ورودی های (در مجموعه داده، تعداد تکرار ، نوع تابع فعال سازی ،نرخ یادگیری)گرفته میشود و دو تابع به روز رسانی وزن ها و تخمین نهایی صدا زده میشوند

به روزرسانی وزن ها ابتدا برای تمامی وزن ها مقدار ۰ در نظر گرفته و سپس به ازای تمامی داده های ازمون وزن به صورت زیر محاسبه میشود:

ابتدا تخمین زده میشود که مقدار نهایی برای داده ازمون چقدر است و سپس وزن (به جز مرحله اول)آن به صورت زیر به روزرسانی میشود:

خطا میزان اختلاف برچسب اصلی داده و مقدار تخمین زده شده است

```
error = row[-1] - prediction
```

```
weights[i + 1] = weights[i + 1] + l_rate * error * row[i]
```

در این متد پارامترهایی مثل تعداد تکرار و دقت نیز بررسی میشوند

```
#perceptron
def train_weights_perceptron(train,validate, l_rate, max_epoch, activation_func):
    weights = [0.0 for i in range(len(train[0]))]
    accuracy_train = [0]
    accuracy_validate = [0]
    flag = True
    epoch = 0
    convergence = 0
    while flag:
      p_train = list()
      for row in train:
          prediction = predict_activation([row], weights, activation_func)[0]
          p_train.append(prediction)
          error = row[-1] - prediction
          weights[0] = weights[0] + l_rate * error
          for i in range(len(row) - 1):
              weights[i + 1] = weights[i + 1] + l_rate * error * row[i]
      accuracy_train.append(accuracy_score(train[:,-1],p_train)*100)
      accuracy_validate.append(accuracy_score(validate[:,-1], predict_activation(validate,weights, activation_func))*100)

      if abs(accuracy_train[-1]-accuracy_train[-2])<3 :
        convergence+=1

      if epoch >= max_epoch :
        flag = False
      else : epoch +=1
    return weights, accuracy_train,accuracy_validate

def perceptron(train,validate, test, l_rate, max_epoch, activation_func):
    weights , accuracy_train,accuracy_validate = train_weights_perceptron(train,validate, l_rate, max_epoch, activation_func)
    predictions = predict_activation(test, weights, activation_func)
    return predictions , accuracy_train,accuracy_validate
```

با توجه به آنچه در قسمت دوم سئوال ۳ امده سه تابع فعال سازی relu,sigmoid,softmax را پیاده کرده ایم:

```
#define perceptrone with 3 diff activation function:
def predict_activation(test, weights, activation_func):
    predictions = list()
    for row in test:
      activation = weights[0]
      for i in range(len(row) - 1):
          activation += weights[i + 1] * row[i]
      temp = 0
      #relu
      if activation_func == 1:
        if activation >= 0.0 : temp = 1
        predictions.append(temp)
      #sigmoid
      elif activation_func == 2:
        temp = np.exp(-activation)/(np.exp(-activation)+1)**2
        if temp > 0.0 : temp = 1
        elif temp <= 0.0 : temp = 0
        predictions.append(temp)
      #softmax
      elif activation_func == 3:
        exps = np.exp(activation - activation.max())
        predictions.append(exps / np.sum(exps, axis=0))
    return predictions
```

تفاوت ادالاین و پرسپترون در هنگام محاسبه خطا است که در به روز رسانی وزن ها اثر میگذارد:

خطا برای هر داده آموزش به صورت زیر محاسبه میشود:

```
for i in range(len(row) - 1):
    activation += weights[i + 1] * row[i]
error = row[-1] - activation
```

و سپس وزن ها به صورت زیر به روز رسانی میشوند:

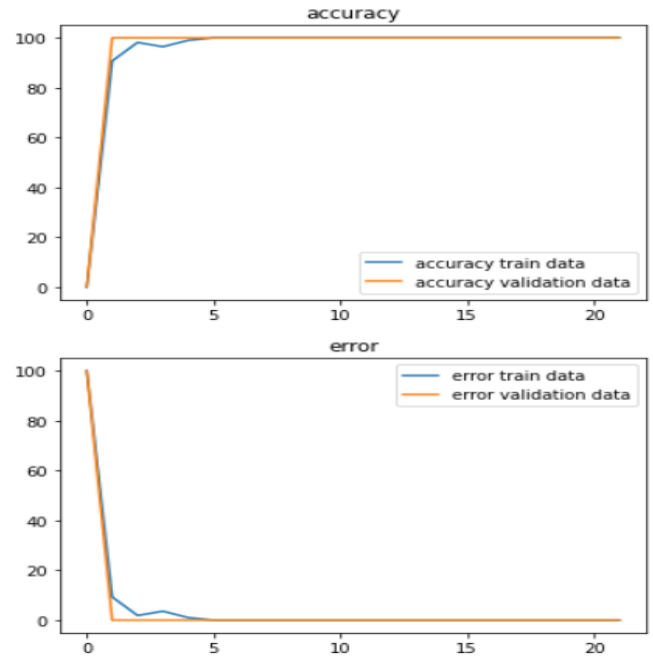
```
prediction = predict_activation([row], weights, activation_func)[0]
if (row[-1]-prediction) != 0:
  for i in range(len(row) - 1):
    weights[i + 1] = weights[i + 1] + l_rate * error * row[i]
```

پیاده سازی ادالاین:

```
def train_weights_adaline(train,validate, l_rate, max_epoch, activation_func):
    weights = [0.0 for i in range(len(train[0]))]
    accuracy_train = [0]
    accuracy_validate = [0]
    flag = True
    epoch = 0
    convergence = 0
    while flag:
      p_train = list()
      for row in train:
          activation = weights[0]
          for i in range(len(row) - 1):
              activation += weights[i + 1] * row[i]
          error = row[-1] - activation

          weights[0] = weights[0] + l_rate * error
          prediction = predict_activation([row], weights, activation_func)[0]
          if (row[-1]-prediction) != 0:
            for i in range(len(row) - 1):
              weights[i + 1] = weights[i + 1] + l_rate * error * row[i]

          p_train.append(prediction)
      accuracy_train.append(accuracy_score(train[:,-1],p_train)*100)
      accuracy_validate.append(accuracy_score(validate[:,-1], predict_activation(validate,weights, activation_func))*100)

      if abs(accuracy_train[-1]-accuracy_train[-2])<3 :
        convergence+=1

      if epoch >= max_epoch :
        flag = False
      else : epoch +=1

    return weights, accuracy_train,accuracy_validate

def adaline(train,validate, test, l_rate, max_epoch, activation_func):
    weights , accuracy_train,accuracy_validate = train_weights_adaline(train,validate, l_rate, max_epoch, activation_func)
    predictions = predict_activation(test, weights, activation_func)
    return predictions , accuracy_train,accuracy_validate
```

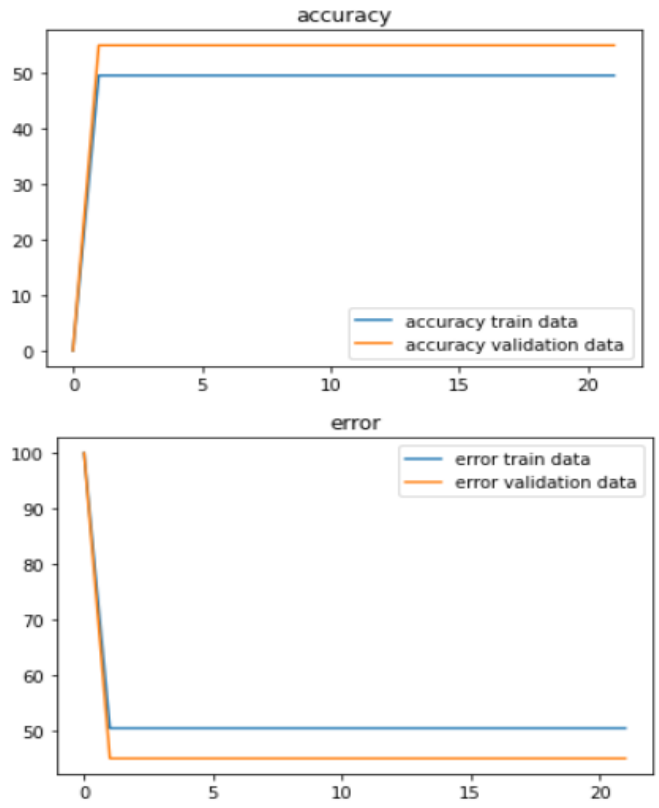
شبکه تکه لایه پرسپترون با تابع فعالسازی relu :(بهترین نتیجه را دارد)

```
Accuracy on test data = 100.0 and learninf rate is : 0.001
```



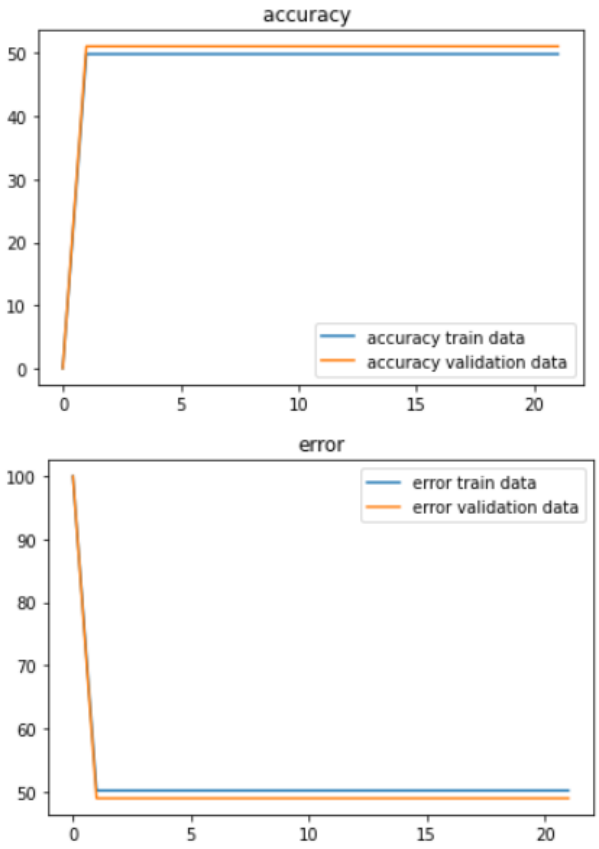
پرسپترون با تابع فعالسازی sigmoid :

```
Accuracy on test data = 49.0 and learninf rate is : 0.001
```



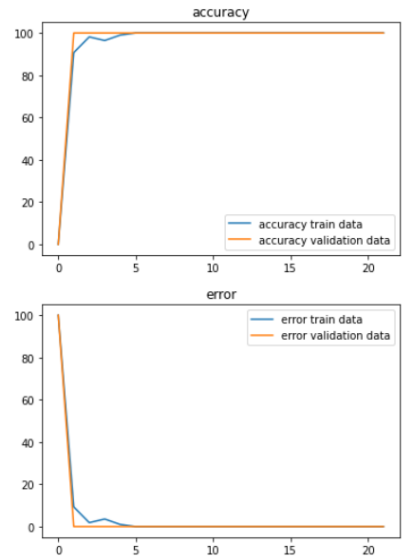
پرسپترون با تابع فعالسازی softmax :

```
Accuracy on test data = 49.85 and learninf rate is : 0.001
```

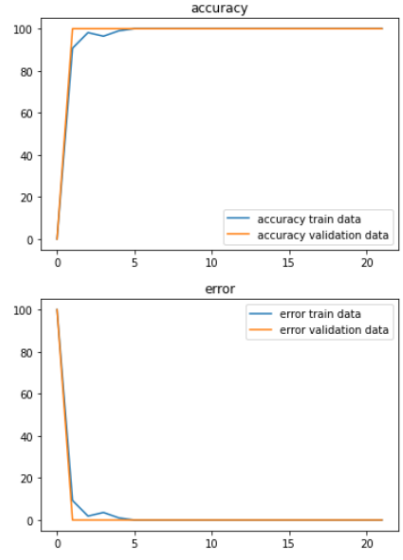


۵ نرخ یادگیری متفاوت روی پرسپترون با تابع فعالسازی relu امتحان کرده ایم:

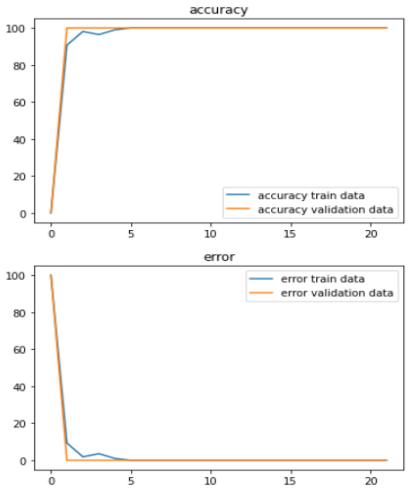
Accuracy on test data = 100.0 and learninf rate is : 0.01



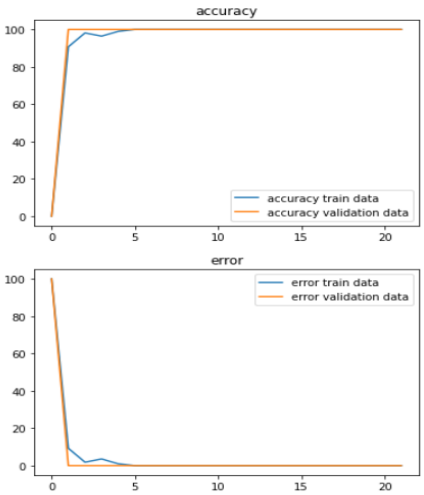
Accuracy on test data = 100.0 and learninf rate is : 0.1



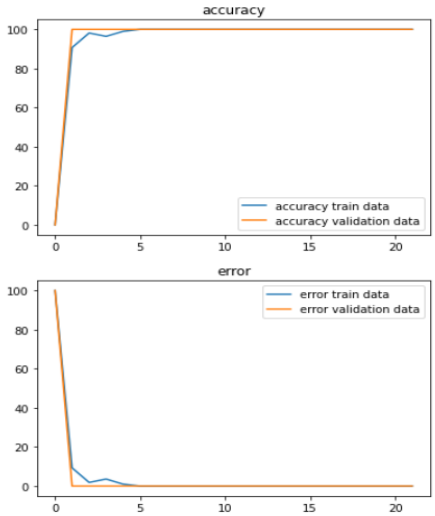
Accuracy on test data = 100.0 and learninf rate is : 0.0001



Accuracy on test data = 100.0 and learninf rate is : 0.001



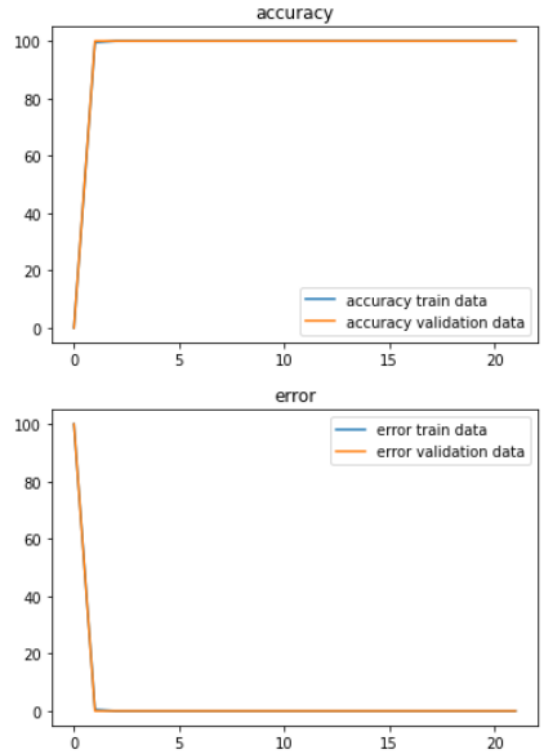
Accuracy on test data = 100.0 and learninf rate is : 1e-05



دقت نتایج به دست آمده یکسان است و تنها تفاوت در سرعت همگرایی می باشد

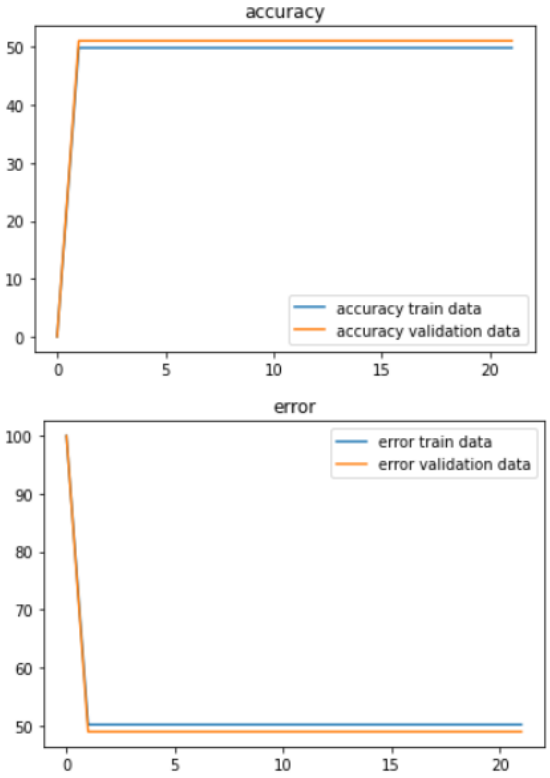
شبکه تک لایه ادالاین با تابع فعالسازی relu :

```
Accuracy on test data = 100.0 and learninf rate is : 0.02
```



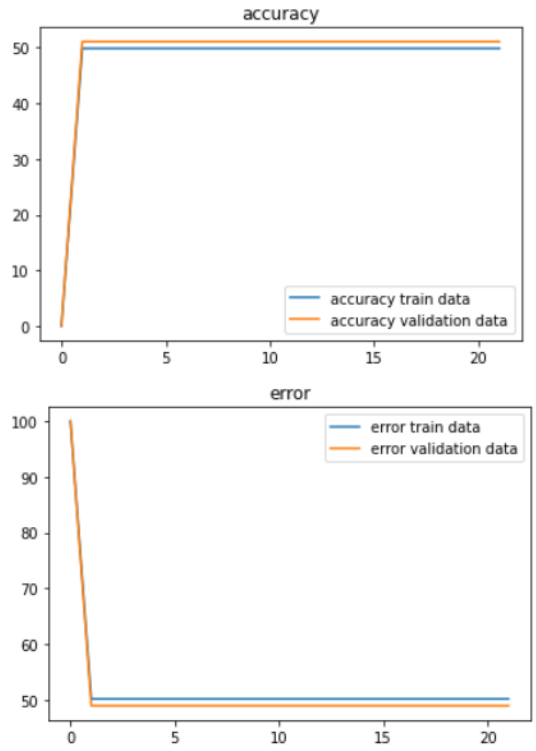
شبکه تک لایه ادالاین با تابع فعالسازی sigmoid :

```
Accuracy on test data = 49.85 and learninf rate is : 0.02
```



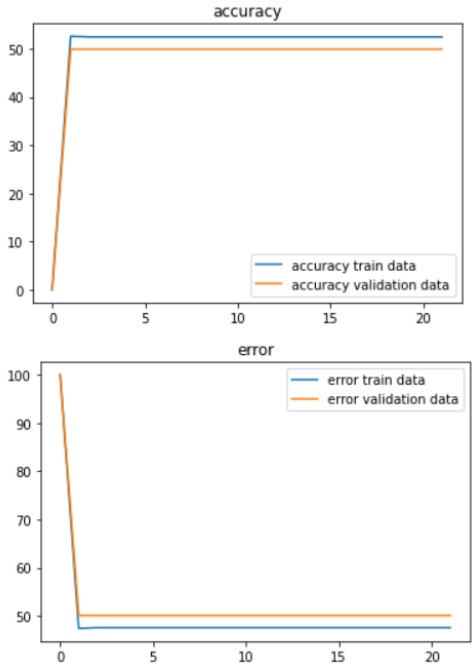
شبکه تک لایه ادالاین با تابع فعالسازی softmax :

```
Accuracy on test data = 49.85 and learninf rate is : 0.02
```

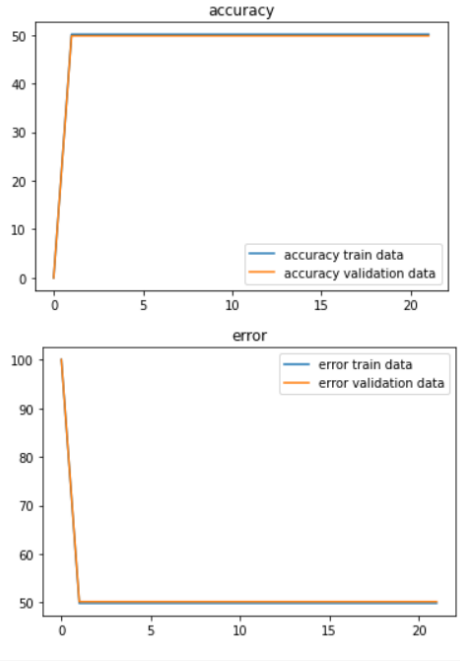


به طور کلی چون داده های ما خطی جدا پذیر بودند هردو پیاده سازی perceptron و Adaline با تابع فعالسازی های متفاوت و نرخ یادگیری متفاوت می توانند به ما دقت ۱۰۰ درصد بدهند

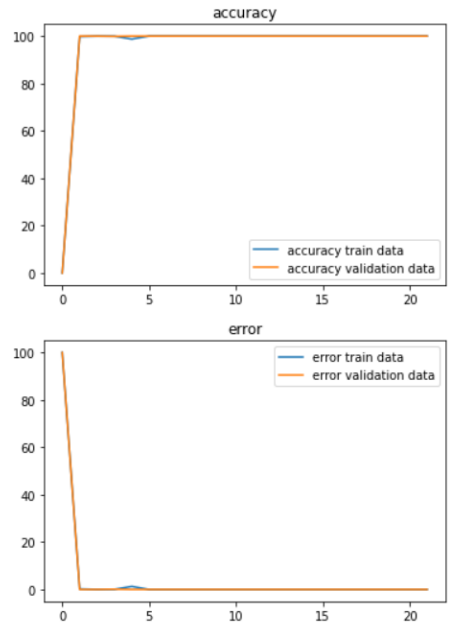
Accuracy on test data = 50.2 and learninf rate is : 0.002



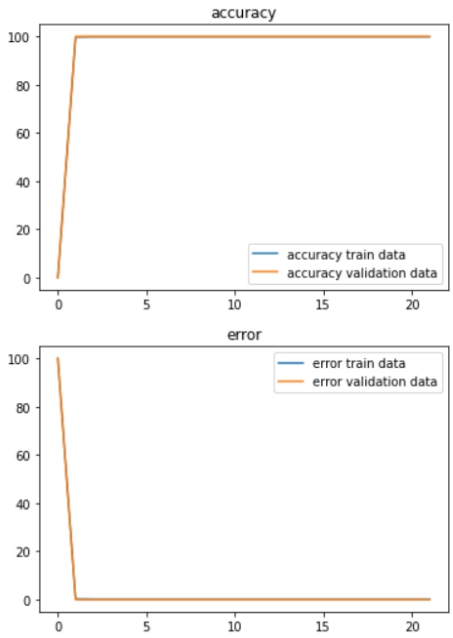
Accuracy on test data = 49.7 and learninf rate is : 0.01



Accuracy on test data = 100.0 and learning rate is : 0.01



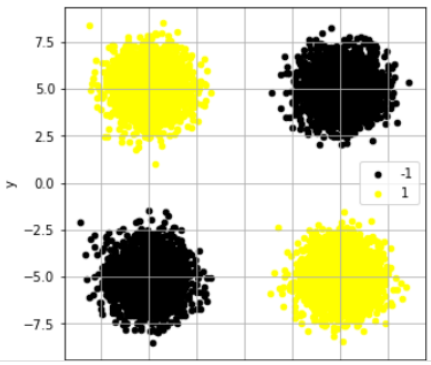
Accuracy on test data = 100.0 and learning rate is : 0.1



ادالاین با ۵ نرخ یادگیری متفاوت

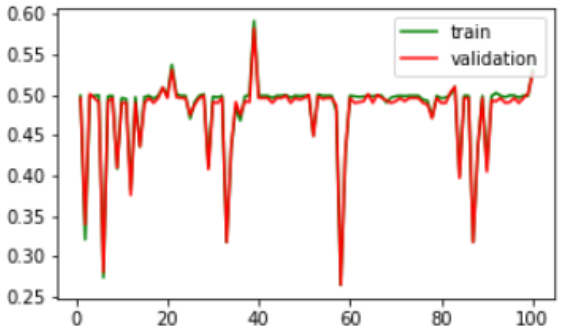
۴) ابتدا در دو جهت میانگین متقارن داده ها را به فرم نرمال گاوسی تولید میکنیم سپس ۴ حالت [-,-],[-,+],[+,-],[+,+] داده ها تقسیم و کلاس بندی میکنیم و در انتها با استفاده از برچسب داده شده به هر داده آن را رسم میکنیم:

```
X11 = np.random.multivariate_normal(mean11, cov, 2500)
X22 = np.random.multivariate_normal(mean22, cov, 2500)
X12 = np.random.multivariate_normal(mean12, cov, 2500)
X21 = np.random.multivariate_normal(mean21, cov, 2500)
df11 = pd.DataFrame(dict(x=X11[:,0], y=X11[:,1], label=[-1] * X11.shape[0]))
df22 = pd.DataFrame(dict(x=X22[:,0], y=X22[:,1], label=[-1] * X22.shape[0]))
df12 = pd.DataFrame(dict(x=X12[:,0], y=X12[:,1], label=[1] * X12.shape[0]))
df21 = pd.DataFrame(dict(x=X21[:,0], y=X21[:,1], label=[1] * X21.shape[0]))
frames = [df11, df12, df21, df22]
dataset = pd.concat(frames)
colors = {-1:'black', 1:'yellow'}
fig, ax = plt.subplots(figsize=(5, 5))
for grp, pdf in dataset.groupby(by='label'):
    pdf.plot(ax=ax, kind='scatter', x='x', y='y', label=grp, color=colors[grp])

plt.grid(True)
plt.show()
```

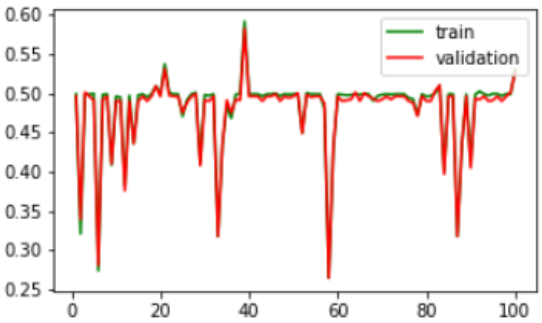


برای ۵ نرخ یادگیری متفاوت با پرسپترون:

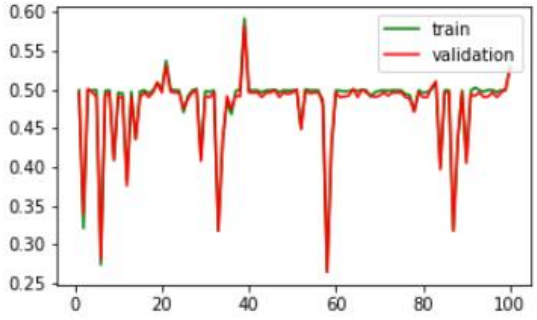
```
Converge in 100 iteration with learning-rate: 0.0001
Accuracy: 0.466
```



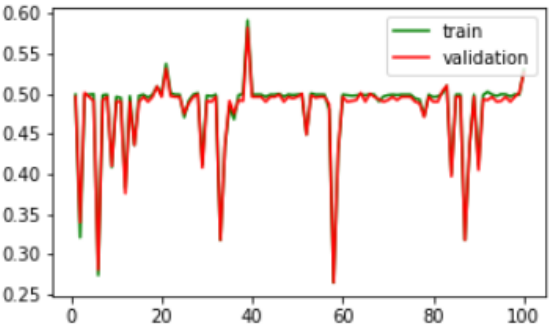

```
Converge in 100 iteration with learning-rate: 0.001
Accuracy: 0.466
```



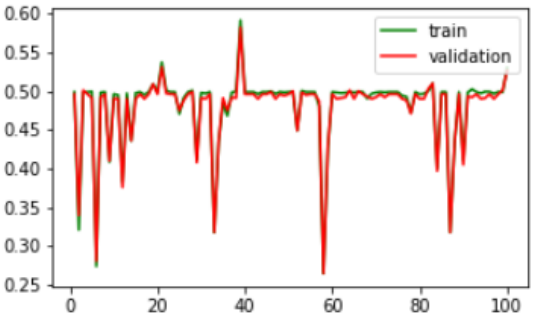
```
Converge in 100 iteration with learning-rate: 0.01
Accuracy: 0.466
```




```
Converge in 100 iteration with learning-rate: 0.1
Accuracy: 0.466
```

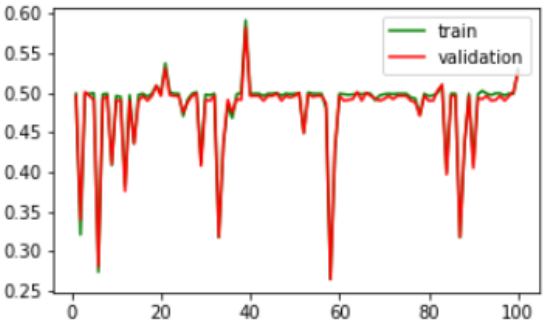



```
Converge in 100 iteration with learning-rate: 0.9
Accuracy: 0.466
```

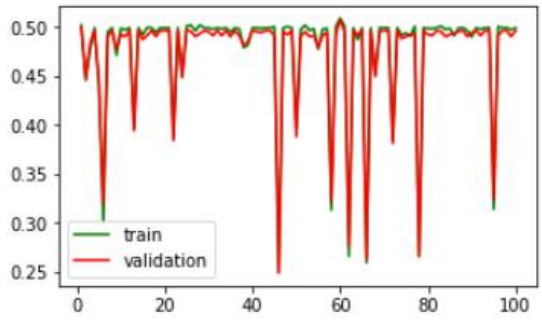


با ۳ تابع فعالسازی:

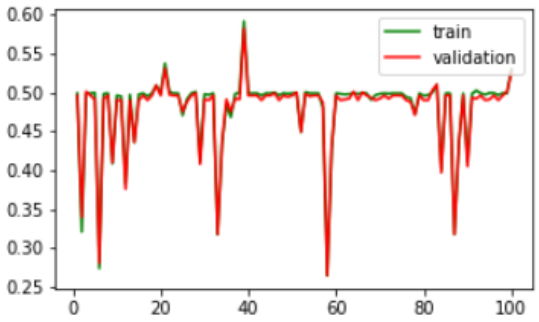
```
Converge in 100 iteration with activation function: sigmoid
Accuracy: 0.466
```



```
Converge in 100 iteration with activation function: relu
Accuracy: 0.494
```

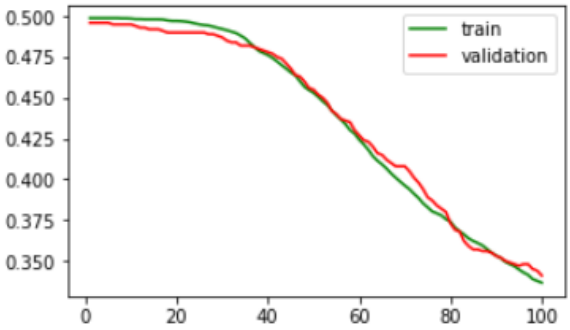



```
Converge in 100 iteration with activation function: tanh
Accuracy: 0.466
```

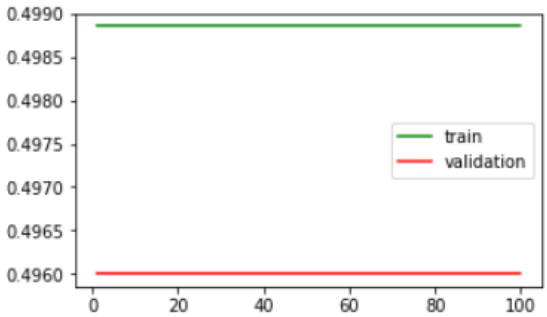


ادالاین با ۵ مقدار نرخ یادگیری متفاوت:

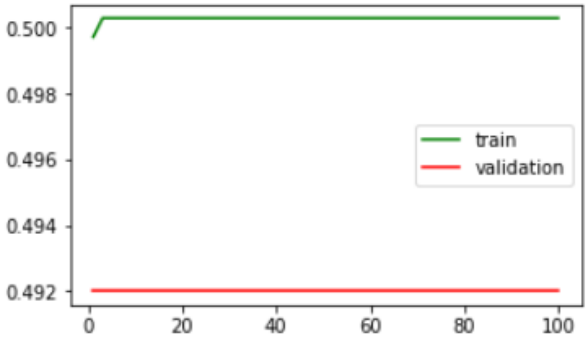
```
Converge in 100 iteration with learning-rate: 1e-06
Accuracy: 0.66
```



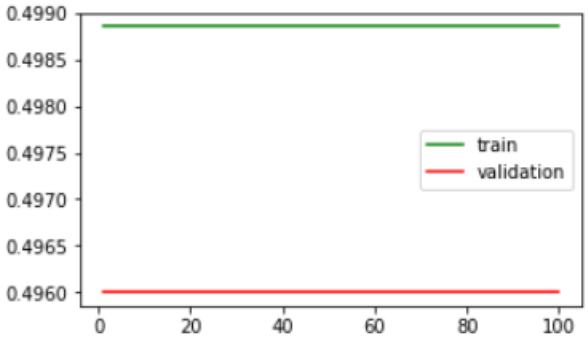

```
Converge in 100 iteration with learning-rate: 1e-05
Accuracy: 0.4945
```



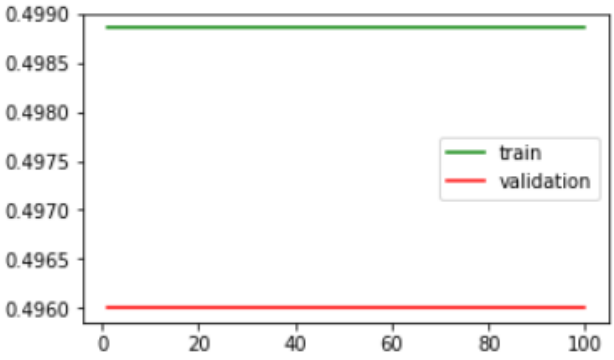
```
Converge in 100 iteration with learning-rate: 0.0001
Accuracy: 0.491
```




```
Converge in 100 iteration with learning-rate: 0.001
Accuracy: 0.494
```

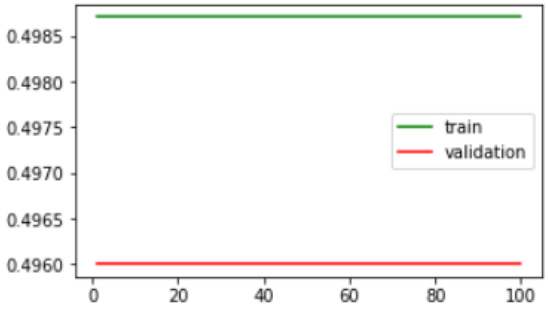



```
Converge in 100 iteration with learning-rate: 0.01
Accuracy: 0.4945
```

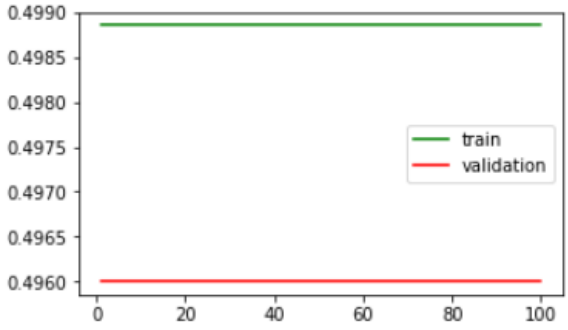


ادالاین با ۳ تابع فعالسازی متفاوت:

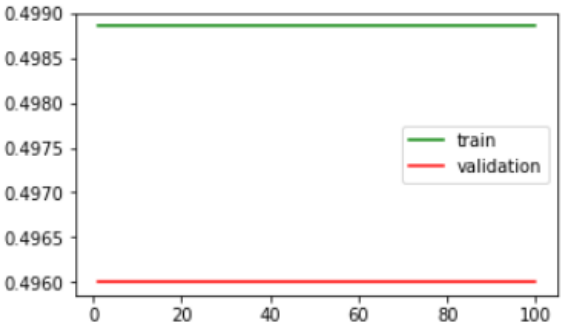
```
Converge in 100 iteration with activation function: relu
Accuracy: 0.493
```



```
Converge in 100 iteration with activation function: tanh
Accuracy: 0.4945
```




```
Converge in 100 iteration with activation function: sigmoid
Accuracy: 0.4945
```



این دو مدل توانسته برای هر دو مجموعه به خوبی تفکیک انجام دهد تنها تفاوت در میزان همگرایی و نرخ یادگیری و تابع فعالسازی است که هرکدام در نتیجه میتوانند موثر باشند.