ex1-bayes-histograms

October 7, 2018

1 Practical work 3

Author: Romain Claret ## Exercice 1 ### a. Bayes Histograms

a) Loading training data

```
In [1]: import pandas as pd
        col_id = ['x1', 'x2', 'y']
        data_train = pd.read_csv('ex1-data-train.csv', names=col_id)
        data_train.head(3)
```

```
Out[1]:          x1         x2  y
        0  34.623660  78.024693  0
        1  30.286711  43.894998  0
        2  35.847409  72.902198  0
```

b) Compute P(C0) and P(C1)

```
In [2]: x = data_train[col_id[:2]]
        y = data_train[col_id[-1:]]
        N = len(y)

        neg = y[y['y'] == 0].index
        pos = y[y['y'] == 1].index

        neg = x.loc[neg][col_id[0]],x.loc[neg][col_id[1]]
        pos = x.loc[pos][col_id[0]],x.loc[pos][col_id[1]]

        class_neg = neg[0]+neg[1]
        class_pos = pos[0]+pos[1]

        p_c0 = len(class_neg)/N
        p_c1 = len(class_pos)/N

        print("p_c0:",p_c0)
        print("p_c1:",p_c1)
```

```
p_c0: 0.4
p_c1: 0.6
```

c) Compute histograms for x1 and x2

```
In [3]: import matplotlib.pyplot as plt
        import numpy as np

        fig, ((fig_11, fig_12),(fig_21, fig_22)) = plt.subplots(2, 2, figsize=(15,15))

        histValues, edgeValues = np.histogram(pos[0], bins='auto')
        fig_11.bar(edgeValues[:-1], histValues)
        fig_11.set_title('x1 pass')
        fig_11.set_xlabel("Score")
        fig_11.set_ylabel("Students")

        histValues, edgeValues = np.histogram(neg[0], bins='auto')
        fig_12.bar(edgeValues[:-1], histValues)
        fig_12.set_title('x1 fail')
        fig_12.set_xlabel("Score")
        fig_12.set_ylabel("Students")

        histValues, edgeValues = np.histogram(pos[1], bins='auto')
        fig_21.bar(edgeValues[:-1], histValues)
        fig_21.set_title('x2 pass')
        fig_21.set_xlabel("Score")
        fig_21.set_ylabel("Students")

        histValues, edgeValues = np.histogram(neg[1], bins='auto')
        fig_22.bar(edgeValues[:-1], histValues)
        fig_22.set_title('x2 fail')
        fig_22.set_xlabel("Score")
        fig_22.set_ylabel("Students")

        plt.show()
```

```
<Figure size 1500x1500 with 4 Axes>
```

d) compute likelihoods ### Question to TA: I really don't get why len(edgeValues) is len(histValues)-1?

```
In [4]: def likelihoodHist(x, histValues, edgeValues):
            total_histValues = np.sum(histValues)

            for i in range(len(histValues)):
                if edgeValues[i] == x:
                    return histValues[i]/total_histValues
```

```
            if edgeValues[i] > x:
                return histValues[i-1]/total_histValues
        return 0

    #histValues, edgeValues = np.histogram(pos[0], bins='auto')
    #for i in range(N):
    #    print(likelihoodHist(x['x1'][i], histValues, edgeValues))
```

e) implement bayes rule

```
In [5]: data_test = pd.read_csv('ex1-data-test.csv', names=col_id)
        data_test_x = data_train[col_id[:2]]
        data_test_y = data_train[col_id[-1:]]
        data_test_N = len(data_test_y)

        neg_0_histValues, neg_0_edgeValues = np.histogram(neg[0], bins='auto')
        pos_0_histValues, pos_0_edgeValues = np.histogram(pos[0], bins='auto')
        neg_1_histValues, neg_1_edgeValues = np.histogram(neg[1], bins='auto')
        pos_1_histValues, pos_1_edgeValues = np.histogram(pos[1], bins='auto')

        result_x1 = []
        for x in data_test_x['x1']:
            result_x1.append(np.argmax([likelihoodHist(x, neg_0_histValues, neg_0_edgeValues) *
                                        likelihoodHist(x, pos_0_histValues, pos_0_edgeValues) *

        result_x2 = []
        for x in data_test_x['x1']:
            result_x2.append(np.argmax([likelihoodHist(x, neg_1_histValues, neg_1_edgeValues) *
                                        likelihoodHist(x, pos_1_histValues, pos_1_edgeValues) *

        result_x1x2 = []
        for i in range (len(data_test_x)):
            result_x1x2.append(np.argmax([likelihoodHist(data_test_x['x1'][i], neg_0_histValues
                                          likelihoodHist(data_test_x['x2'][i], neg_1_histValues
                                          likelihoodHist(data_test_x['x1'][i], pos_0_histValues
                                          likelihoodHist(data_test_x['x2'][i], pos_1_histValues

        print("x1 accuracy: ", sum(1 for x in (data_test_y['y'] == result_x1) if x)/data_test_
        print("x2 accuracy: ", sum(1 for x in (data_test_y['y'] == result_x2) if x)/data_test_
        print("x1x2 accuracy: ", sum(1 for x in (data_test_y['y'] == result_x1x2) if x)/data_te

x1 accuracy:  0.71
x2 accuracy:  0.54
x1x2 accuracy:  0.64
```

- The result is scandalous.... I don't get it.. how is it even possible.. Logically the x1x2 should be the best.

1.0.1 b. Bayes Gaussian Distribution

```
In [6]: def likelihoodGauss(x, mean, var):
            return (1.0 / (var * np.sqrt(2 * np.pi)) * np.exp( - np.square(x - mean) / (2 * np

        #print(likelihoodGauss(10, np.mean(neg[0]), np.var(neg[0])))
```

```
In [7]: neg_0_mean = np.mean(neg[0])
        neg_0_var = np.var(neg[0])

        neg_1_mean = np.mean(neg[1])
        neg_1_var = np.var(neg[1])

        pos_0_mean = np.mean(pos[0])
        pos_0_var = np.var(pos[0])

        pos_1_mean = np.mean(pos[1])
        pos_1_var = np.var(pos[1])
```

```
In [8]: result_x1_gauss = []
        for x in data_test_x['x1']:
            result_x1_gauss.append(np.argmax([likelihoodGauss(x, neg_0_mean, neg_0_var) * p_c0
                                              likelihoodGauss(x, pos_0_mean, pos_0_var) * p_c1]))
        #print(result_x1_gauss)

        result_x2_gauss = []
        for x in data_test_x['x1']:
            result_x2_gauss.append(np.argmax([likelihoodGauss(x, neg_1_mean, neg_1_var) * p_c0
                                              likelihoodGauss(x, pos_1_mean, pos_1_var) * p_c1]))
        #print(result_x2_gauss)

        result_x1x2_gauss = []
        for i in range (len(data_test_x)):
            result_x1x2_gauss.append(np.argmax([likelihoodGauss(data_test_x['x1'][i], neg_0_mea
                                                likelihoodGauss(data_test_x['x2'][i], neg_1_mean, neg
                                                likelihoodGauss(data_test_x['x1'][i], pos_0_mean, pos
                                                likelihoodGauss(data_test_x['x2'][i], pos_1_mean, pos
        #print(result_x1x2_gauss)

        print("x1 accuracy: ", sum(1 for x in (data_test_y['y'] == result_x1_gauss) if x)/data_
        print("x2 accuracy: ", sum(1 for x in (data_test_y['y'] == result_x2_gauss) if x)/data_
        print("x1x2 accuracy: ", sum(1 for x in (data_test_y['y'] == result_x1x2_gauss) if x)/d
```

```
x1 accuracy:  0.6
x2 accuracy:  0.6
x1x2 accuracy:  0.6
```

- The result is still scandalous....