BURAK ÖMÜR

2016400186

İçindekiler

QUESTION 1

1- Investigation of Poor Performance

There may be few reasons for poor performance because of fixed hyperparameters, kernel and classifier. We know that svm is regularizes itself, so reason is not the regularization. We have not much reasons left.

- Reason can be imbalanced data.
- Reason can non-binary labels.
- Reason can be the unnormalized data.
- Reason can feature/sample ratio.

We can try and check if any of those above is the main reason.

2- Conclusion & Implementation

Problem is not the non-binary classification. By checking labels we can see that. Also, by counting the 0 and 1 labels we can clearly see that data imbalance also not the problem. Feature number is 50 and sample number is 1600, so feature/sample ratio is not the problem.

We are left with one last reason. By checking dataset, we can see that each of 50 features have different range of values. In example, we can think it as comparing apple fruit with apple company, so their max and min values are different from each other. By max-min rescaling, we can achieve some improvement in accuracy. This adjusts them between 0 and 1 and, then we check if it resolves the problem. After implementation, accuracy has improved from 0.59 to 1, therefore we can say that we have found the main problem.

Also, to make a point, normalization is a must for linear models like SVM in this case.

$$0.59 \rightarrow 1$$

- I have implemented a class structure to normalize dataset. Also, i have used this class for 2nd question also.

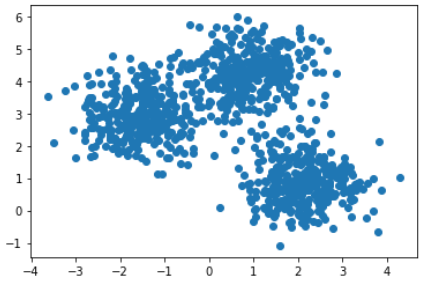
3- Code

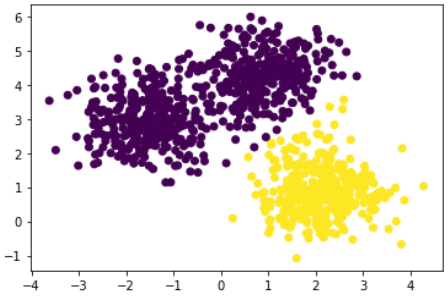
```
import numpy as np
import numpy as np
from sklearn import svm

train_data = np.load('train_data1.npy')
train_label = np.load('train_label1.npy')
test_data = np.load('test_data1.npy')
test_label = np.load('test_label1.npy')

class SVM:

    feature_max = []  # max values of features
    feature_min = []  # min values of features
    clf = None        # model

    def __init__(self, train_data, train_label):
        self.feature_max = []  # init
        self.feature_min = []   # init
        self.max_min_features(train_data)   # calculate min-max values and assing
        self.clf = svm.SVC(gamma=0.001, C=100.)   # construct model
        self.clf.fit(self.normalize(train_data), train_label)  # train

    def max_min_features(self, data):
        (row, column) = np.shape(data)
        for j in range(0, column):   # for each of features
            mx, mn = np.max(data[:,j]), np.min(data[:,j])  # calculate min and max
            self.feature_max.append(mx)  # assign
            self.feature_min.append(mn)  # assign

    def normalize(self,data):
        (row, column) = np.shape(data)
        new = np.zeros((row, column))
        for j in range(0, column):   # for each feature
            mx, mn = self.feature_max[j], self.feature_min[j] # get min and max
            for i in range(0, row):  # for each point
                new[i,j] = (data[i,j] - mn) / (mx-mn) # max-min scale
        return new  # return scaled

    def test(self, test_data, test_label):
        y_pred = self.clf.predict(self.normalize(test_data))  # predict with normalization
        correct_prediction = np.equal(y_pred, test_label)    # check for accuracy
        return np.mean(correct_prediction.astype(np.float32))  #calculate


svMachine = SVM(train_data, train_label)
accuracy = svMachine.test(test_data, test_label)

print(accuracy)
```

QUESTION 2

1- Propose a solution to obtain labels

We first try to see visually because there are 2 feature which allows us to visualize dataset. By scattering dataset, we can clearly see that there is a clustering in dataset. We see 3 globular shaped clusters but in order to simplify things we choose binary clustering. Therefore, labels can be obtained using an unsupervised clustering technique. By applying, kmeans-clustering we can label them as below.





```
import numpy as np
data = np.load('data2.npy')
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans as KM

labels = KM(n_clusters=2, random_state=0).fit(data).labels_ # fit data into 2 clusters and get true labels

plt.scatter(data[:,0], data[:,1], c=labels)  # scatter
plt.show()
```

2- Classification with SVM

In order to do classification, we construct test & train data and get their corresponding labels from part 1 of this question. The ratio of train to test is 4 that corresponds to 800/200. Then, Using SVM-classifier as in question 1, we train the classifier with train data and its labels, then do test on test data. After obtaining the predicted classifications, we compare them with the actual obtained labels. This gives us 0.995 accuracy.

$$Accuracy: 0.995$$

```
stacked = np.column_stack((labels.reshape((1000, 1)), data)) # stack labels with data
np.random.seed(1)        #random seed
np.random.shuffle(stacked)  # shuffle stacked matrix
test_data, train_data = stacked[:200, 1:], stacked[200:, 1:]  # get data
test_label, train_label = stacked[:200, 0], stacked[200:, 0]  # get labels

svMachine = SVM(train_data, train_label)     # use prewritten class to train
accuracy = svMachine.test(test_data,test_label)  # test

print(accuracy)
```

QUESTION 3

In my opinion diabetes is mostly related with age of person. Therefore, “age” attribute would give the lowest entropy in given attributes because, if we assume unbiased dataset, age will most likely help us most splitting tree into braches. In other words, homogenity of “age” splitted data would be high. We would choose most least entropy values attribute as root.

No, i do not think that using “patient ID” as attribute is meaningful. Because, it is assumed to be unique for each of patients. It wont help to construct tree and gain information from dataset. It will always have high nonhomogenioity and high entropy value.

QUESTION 4

Below i have showed how we find principal components.

- PCA focuses on most variative features. To find principal components, we first normalize dataset by making $\lambda = 0$ and $\sigma^2 = 1$ to simplify things. Then, we calculate covariance matrix as shown below.

$$S = \frac{1}{N}(X^T.X)\,, \qquad X\ is\ data\ matrix, \qquad N\ is\ number\ of\ rows$$

  - We want to solve:

$$\max(VAR[Z_1]) = \max(a_1^T.S.a_1)$$

$$st.\ a_1^T.a_1 = 1$$

  - This gives: $S.a_1 = \lambda * a_1\ ,\ where\ a_1 is\ eigenvector\ of\ i^{th} feature$
  - We look for most largest eigenvalued eigenvectors to get PC's.

$$PC_i = S.a_i, where\ i\ is\ feature\ number$$

1- Imputing New Values

If we impute mean values for features, we decrease the variance in that feature, which decrease the chance to be a PC. In more detail, we look for the features that retain most of the variance to find principal components. If we apply mean imputation it direcly reduces variance in that feature.

$$var(X) = s^2 = \sum \frac{(x_i - x_{bar})^2}{n-1}$$

$For\ imputed\ values\ nominator\ will\ be\ 0, which\ clearly\ shows\ variance\ will\ decrease\ why$

2- Deleting Missing Valued Rows

If we delete missing valued customers from dataset, we lose too much information which we dont want. Deleting 200 customers can cause the variances of all of the features change drastically. If variances change, covariances change, then principal components may change.

3- Conclusion

Either way, missing features or attributes are not good in any way. Deleting those who missing is the easiest way, but not the good way. Imputing values is a good approach, but mean imputation technique may change the distribution of data which is also not good. However, we can easily say that applying some kind of imputation is always better than deleting those who missing.

4- Result

We can say that covariance matrix changes and this changes resulting PC's. In other words, change in variances (σ^2) leads to change in covariance matrix (S) that results in changing eigenvalues (λ) and eigenvectors (a_i). This makes PC's to change without doubt. Therefore, in both cases, resulting principal components will be different.

QUESTION 5

1- Compute the optimal w and b

Matrices: $x = \begin{bmatrix} 1 & 3 \\ 0 & 0 \end{bmatrix}_{2x2}, y = \begin{bmatrix} -1 \\ 1 \end{bmatrix}_{2x1}, w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}_{2x1}$

- Applying the condition: $y_i * (w^T.x_i + b) \geq 1, for\ i\ in\ [1, n]$

1) $-1 * (w_1 + b) \geq 1$
2) $1 * (3 * w_1 + b) \geq 1$

$$Deduction \rightarrow\ 1) + 2) = 2 * w_1 \geq 2 = w_1 \geq 1$$

$$\min\left(\frac{1}{2} * (w^T.w)\right) = \min\left(\frac{1}{2}(w_1^2 + w_2^2)\right) = \frac{1}{2}$$

$because$ of the square terms in paranthesis: $\min(w_1^2) = 1\ \&\ \min(w_2^2) = 0$

$$we\ found\ that: w_1 = 1, w_2 = 0$$

- By using expressions 1) and 2) from above:
  1) $-1 * (1 + b) \geq 1$
  2) $1 * (3 + b) \geq 1$

$$Deducion \rightarrow b = -2$$

$$Result \rightarrow w^* = \begin{bmatrix} 1 \\ 0 \end{bmatrix}_{2x1}, b^* = -2$$

2- Compute the optimal α

Matrices: $x = \begin{bmatrix} 1 & 3 \\ 0 & 0 \end{bmatrix}_{2x2}, y = \begin{bmatrix} -1 \\ 1 \end{bmatrix}_{2x1}, w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}_{2x1}, \alpha = \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix}_{2x1}$

- Applying the condition: $\sum_{n=1}^{2} y_n * \alpha_n = 0, \alpha_n > 0$

1) $-1 * \alpha_1 + 1 * \alpha_2 = 2,$

$$Deduction \rightarrow\ \alpha_1 = \alpha_2$$

$$\max\left(\sum_{n=1}^{2} \alpha_n - \frac{1}{2}\sum_{n=1}^{2}\sum_{m=1}^{2} y_n * y_m * \alpha_n * \alpha_m * x_n^T.x_m\right)$$

- By calculating the above expression we can obtain:

$$\max\left(\alpha_1 + \alpha_2 - \frac{1}{2}(\alpha_1^2 - 6 * \alpha_1 * \alpha_2 + 9 * \alpha_2^2)\right)$$

- By merging the above result with 1):

$$\max\left(2 * \alpha_1 - \tfrac{1}{2}(4 * \alpha_1^2)\right) = \max(2 * \alpha_1 - 2 * \alpha_1^2) = \max\left(2 * \alpha_1(1 - \alpha_1)\right)$$

- To calculate α that gives maximal value, we take derivation and find α's.

$$\frac{d(2 * \alpha_1 - 2 * \alpha_1^2)}{d(\alpha_1)} = 2 - 4 * \alpha_1 = 0 \rightarrow\ \alpha_1 = \alpha_2 = \frac{1}{2}$$

$$Result \rightarrow\ \alpha = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}_{2x1}$$

3- Compute the optimal w based on the optimal α

Calculating $w^* \; from \; \alpha$:

$$w^* = \sum_{n=1}^{2} y_n * \alpha_n^* * x_n = -1 * \frac{1}{2} * \begin{bmatrix} 1 \\ 0 \end{bmatrix}_{2x1} + 1 * \frac{1}{2} * \begin{bmatrix} 3 \\ 0 \end{bmatrix}_{2x1} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}_{2x1}$$

Which the resulting optimal weight vector is equal to the resut in the 1'th part of this solution. Which shows we did it correct.