LogoType

Introduction to Big Data Analytics

2015313754

TaeHyung Gil

Index

1. DataSet Description
2. Feature Data Kernel SVM
3. Raw Image Data Kernel SVM
4. Conclusion

1. DataSet & Goal Description

Celelb A :https://www.kaggle.com/jessicali9530/celeba-dataset

This dataset consists of two kinds of data.

01 Features of images (Extracted by human)

There are many attributes describe each image using Boolean value. For example, there are features such as Big_Nose, Bangs …

image_id	5_o_Clock_Shad...	Arched_Eyebrows	Attractive	Bags_Under_Eyes
202599 unique values	-1 1	-1 1	-1 1	-1 1
000001.jpg	-1	1	1	-1
000002.jpg	-1	-1	-1	1
000003.jpg	-1	-1	-1	-1
000004.jpg	-1	-1	1	-1
000005.jpg	-1	1	1	-1
000006.jpg	-1	1	1	-1

02 Raw Image

Images of men and women

1. DataSet & Goal Description

Goal

Gender Classification

My project goal is to classify each data as Male of Female

By Kernel SVM Algorithm

Using

1. Features of images (only)
2. Raw Images (only)

And Compare their results with the other

2. Feature Data Kernel SVM

(This experiment uses only Feature data)

01 I made SVM classification objects with given linear and RBF kernel

01 Linear kernel

```
clf = svm.SVC(kernel='linear',verbose=True,gamma='scale')
```

Python code for making SVM object (Sklearn library)

$$k(x_1, x_2) = x_1^T x_2$$

kernel function of ‘linear’ object.

02 RBF kernel

```
clf = svm.SVC(kernel='rbf',verbose=True,gamma='scale')
```

$$k(x_1, x_2) = \exp\left(-\gamma ||x_1 - x_2||^2\right)$$

kernel function of ‘rbf’ object.

2. Feature Data Kernel SVM

02 I customized SVM's kernel function

I changed SVM's kernel function so that it can calculate the vector's distance.
Hamming Distance and Cosine Distance are known as good metrics for
Boolean Encoded vectors' distance.

(I made following kernels so that they calculate values inversely proportional to their distances)

01 Hamming Distance kernel

```
def get_Hamming_Dist(x1,x2):
    ret = np.zeros(shape=(len(x1), len(x2)), dtype=np.float)
    for idx1, _x1 in enumerate(x1):
        for idx2, _x2 in enumerate(x2):
            ret[idx1][idx2]=np.sum(_x1==_x2)
    return ret
```

02 Cosine Distance kernel

```
def ret_Cosine_Dist(x1,x2):
    ret = np.zeros(shape=(len(x1), len(x2)), dtype=np.float)
    for idx1, _x1 in enumerate(x1):
        for idx2, _x2 in enumerate(x2):
            ret[idx1][idx2]=distance.cosine(_x1,_x2)
    return 1-ret
```

I Used scipy.spatial.cosine function

2. Feature Data Kernel SVM

02 I customized SVM’s kernel function

03 Interpolation Hamming Distance & Cosine Distance

```
def interpolation_HAM_COS(prac=0.5):
    def ret_interpolation(x1,x2):
        return prac*get_Hamming_Dist(x1,x2)+(1-prac)*ret_Cosine_Dist(x1,x2)
    return ret_interpolation
```

This kernel return interpolated value between Hamming Distance and Cosine Distance Using given ratio(prac value).

Below are python codes making SVM objects

```
clf = svm.SVC(kernel=get_Hamming_Dist)

clf = svm.SVC(kernel=ret_Cosine_Dist)

clf = svm.SVC(kernel=interpolation_HAM_COS(prac=0.5))
```

2. Feature Data Kernel SVM

Comparing Results (1,400 Training Data & 600 Test Data)

01

01 Linear kernel

ACC: 0.9217
F1 score: 0.9069
Precision: 0.8808
Recall: 0.9347

Confusion
TP : 229 FP : 31
FN : 16 TN : 324

02 RBF kernel

ACC: 0.9133
F1score: 0.9008
Precision: 0.8459
Recall: 0.9633

Confusion
TP : 236 FP : 43
FN : 9 TN : 312

02

01 Hamming Dist

ACC: 0.9150
F1score: 0.9006
Precision: 0.8619
Recall: 0.9429

Confusion
TP : 231 FP : 37
FN : 14 TN : 318

02 Cosine Dist

ACC: 0.9067
F1score: 0.8943
Precision: 0.8316
Recall: 0.9673

Confusion
TP : 237 FP : 48
FN : 8 TN : 307

03 InterPolation

ACC: 0.9250
F1score: 0.9112
Precision: 0.8817
Recall: 0.9429

Confusion
TP : 231 FP : 31
FN : 14 TN : 324

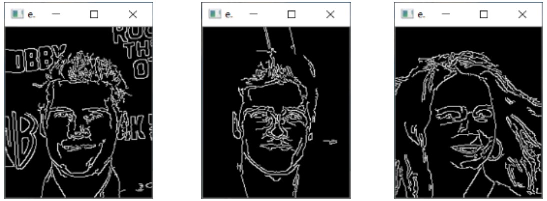
Conclusion

InterPolation Kernel Showed Best ACC & F1

3. Raw Image Data Kernel SVM

01 Preprocessing

01 Edge Detection



```
img = cv2.imread('./img_files/img_align_celeba/'+img_idx+'.jpg',cv2.IMREAD_GRAYSCALE)
img = cv2.Canny(img, 50, 200)
```

Read Image as gray scale
& Detect Edge
Using OpenCV Lib.

02 PCA

Reduced Image Dimension from 38,804 to 1,000
PCA object was fitted using 20,000 Images

```
pca = PCA(n_components=1000)
pca.fit(train_img)
```

3. Raw Image Data Kernel SVM

02 Kernel SVM Classification Result (1,400 Training Data & 600 Test Data)

01 Linear kernel

ACC: 0.5850
F1score: nan
Precision: nan
Recall: 0.0000

Confusion
TP : 0 FP : 0
FN : 166 TN : 234

02 RBF kernel

ACC: 0.5850
F1score: nan
Precision: nan
Recall: 0.0000

Confusion
TP : 0 FP : 0
FN : 166 TN : 234

4. Conclusion

1. **Raw Image Data Kernel SVM Model almost didn’t get fitted to the data.**
   I guess this is because the raw image data is too confusing to classify gender.

2. **When the data is encoded as Boolean values,**
   Cosine Distance, Hamming Distance Calculation can work as SVM Kernel function.

Thank you