CS498 Applied Machine Learning Homework3

Huamin Zhang && Rongzi Wang

Monday, Feburary 20, 2017

4.10

CIFAR-10 is a dataset of 32x32 images in 10 categories, collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. It is often used to evaluate machine learning algorithms. You can download this dataset from https://www.cs.toronto.edu/kriz/cifar.html.

(a)

For each category, compute the mean image and the first 20 principal components. Plot the error resulting from representing the images of each category using the first 20 principal components against the category.

Answer:

```
from sklearn.decomposition import PCA
import pickle
import numpy as np

# read data of all training data
wd = "C:\\Users\\98302\\Desktop\\cifar-10-batches-py\\data_batch_"
batch_file = ["1","2", "3", "4", "5"]
data = []
label = []
for file in batch_file:
    input_file = wd + file
    fo = open(input_file, 'rb')
    dict = pickle.load(fo)
    fo.close()
    data.append(dict['data'])
    label.append(dict['labels'])

############4.10(a)##################################################
class_data=[]
class_pca=[]
class_mean_image=[]
error = []
for i in range(0,10):
    x=[]
    y=[]
    x=np.array(x,dtype = np.uint8)
    new_dict = {'data':x,'labels':y}

    #classfier
    for j in range(0,5):
        for k in range(0,10000):
            if label[j][k] == i:
                new_dict['labels'].append(label[j][k])
                new_dict['data'] = np.concatenate((new_dict['data'],data[j][k]))
    new_dict['data'] = new_dict['data'].reshape(new_dict['data'].shape[0]/3072,3072)
    class_data.append(new_dict)

    #do pca
    pca = PCA(copy=True,n_components=20) #pca para
    new_data = pca.fit_transform(class_data[i]['data']) #transform 10000*3072 to 10000*20
    #mean image
    class_mean_image.append(pca.mean_)
    class_pca.append(new_data)

    #Constructing a low-dimensional representation
    represent_data = pca.inverse_transform(new_data)

    #calculate error
    total_dis = 0
    for image in range(5000):
        dist = np.sqrt(np.sum(np.square(represent_data[image] - class_data[i]['data'][image
        total_dis = total_dis + dist

    ave_err = total_dis/5000
    error.append(ave_err)
error
```

```
[1553.1468061927148,
 1955.6171211491858,
 1497.7538890545302,
 1714.0326230068724,
 1422.0067365299374,
 1749.0802507229187,
 1576.0713019648254,
 1811.8561263737488,
 1501.8965456651561,
 1972.6247988325288]
```

So the error resulting from representing the images of each category using the first 20 principal components against the category is (1553.15382446685, 1955.6179886516406,1497.7200032304538,1714.0308064609096, 1422.0085695761902,1749.0835367137101, 1576.0729220159319,1811.8635062055494, 1501.9083558302561,1972.6160816894055).

The barplot of the error is showed as following.

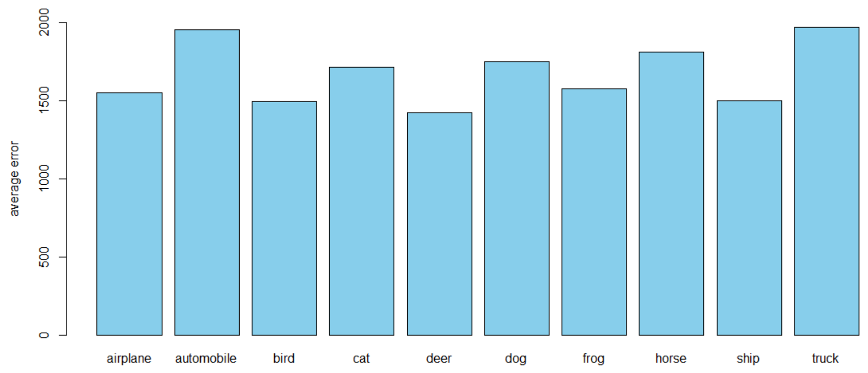

Figure 1:

(b)

Compute the distances between mean images for each pair of classes. Use principal coordinate analysis to make a 2D map of the means of each categories. For this exercise, compute distances by thinking of the images as vectors.

Answer:

```
#############4.10(b)##################################################

#create distance matrix
D = []
for i in range(10):
    for j in range(10):
        dist = np.sum(np.square(class_mean_image[i] - class_mean_image[j]))
        D.append(dist)
D = np.array(D).reshape(10,10)
#Form A,W and get the eigenvectors andeigenvalues of W
I = np.identity(10)
A = I - 0.1 * np.array([1]*100).reshape(10,10)
W = -0.5 * np.dot(np.dot(A,D),np.transpose(A))
eigval, eigvec = np.linalg.eig(W)
eigval
```

```
array([  6.11872731e+06,   1.43736291e+06,   5.23337909e+05,
         1.50913516e+05,   1.39065005e+05,  -3.15542556e-10,
         1.22613194e+04,   4.19451647e+04,   2.88933057e+04,
         3.18558605e+04])
```

```
#the top left r × r block , r =2
sort1 = eigval.argsort()[-1]
sort2 = eigval.argsort()[-2]
v1 = eigval[sort1]
v2 = eigval[sort2]
vec1 = eigvec[:,sort1]
vec2 = eigvec[:,sort2]
diag = np.array([np.sqrt(v1),0,0,np.sqrt(v2)]).reshape(2,2)
eigvec_r = np.concatenate((vec1,vec2)).reshape(2,10)
V_T = np.dot(diag,eigvec_r)
V = np.transpose(V_T)
V
```

```
array([[ 1240.98660652,  -642.90599027],
       [  124.59424171,   502.18646509],
       [ -232.62552266,  -151.03722973],
       [ -551.42585341,  -171.53378739],
       [ -770.97022443,   -68.40681696],
       [ -623.48772513,  -371.41540556],
       [ -983.32529277,   262.41028018],
       [ -260.2478262 ,   -46.83221196],
       [ 1218.75239766,   -23.36626178],
       [  837.7491987 ,   710.90095838]])
```

So the V matrix is showed as above. The 2D map of the means of each categories is showed as following and we can find animals are closest to each other.

(c)

Here is another measure of the similarity of two classes. For class A and class B, define $E(A \to B)$ to be the average error obtained by representing all the images of class A using the mean of class A and the first 20 principal components of class B. Now define the similarity between classes to be $(1/2)(E(A \to B)+E(B \to A))$. Use principal coordinate analysis to make a 2D map of the classes. Compare this map to the map in the previous exercise are they different? why?

Answer:

```
#############4.10(c)################################################

#create E matrix E(i->j) = E_ij
E = []
for i in range(10):
    for j in range(10):
```

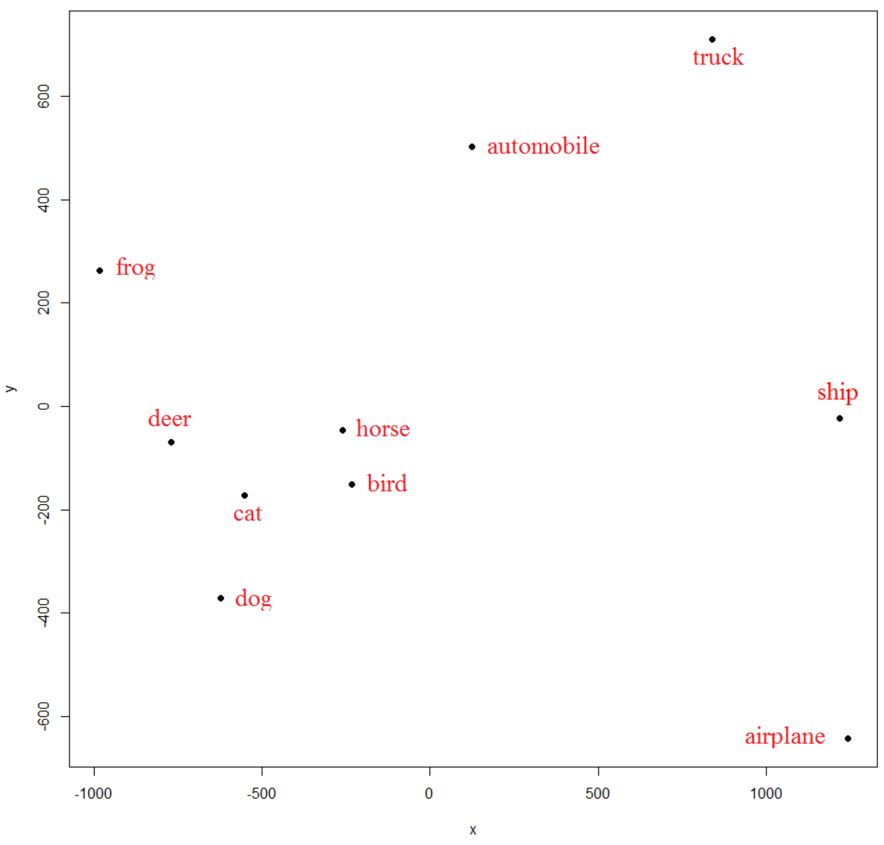
truck
automobile
frog
ship
deer
horse
bird
cat
dog
airplane
600
400
200
0
-200
-400
-600
y
-1000
-500
0
500
1000
x

Figure 2:

```
        pca_i = PCA(copy=True,n_components=20) #pca para
        data_i = pca_i.fit_transform(class_data[i]['data']) #transform 10000*3072 to 10000*
        pca_j = PCA(copy=True,n_components=20) #pca para
        data_j = pca_j.fit_transform(class_data[j]['data']) #transform 10000*3072 to 10000*
        pca_i.components_ = pca_j.components_
        represent_data_i = pca_i.inverse_transform(data_i)
        total_dis = 0
        for image in range(5000):
            dist = np.sqrt(np.sum(np.square(represent_data_i[image] - class_data[i]['data']
            total_dis = total_dis + dist
        ave_err = total_dis/5000
        E.append(ave_err)
E = np.array(E).reshape(10,10)

D2 = []
for i in range(10):
    for j in range(10):
        similarity = 0.5 * (E[i][j] + E[j][i])
        if i == j:
            similarity = 0
        D2.append(similarity)
D2 = np.array(D2).reshape(10,10)
#Form A,W and get the eigenvectors andeigenvalues of W
I2 = np.identity(10)
A2 = I - 0.1 * np.array([1]*100).reshape(10,10)
W2 = -0.5 * np.dot(np.dot(A2,D2),np.transpose(A2))
eigval2, eigvec2 = np.linalg.eig(W2)
eigval2
```

```
array([  4.45296395e+03,  -5.21294287e-14,   2.32013646e+03,
         2.13174202e+03,   1.33214552e+03,   1.76585256e+03,
         1.71689619e+03,   1.66518607e+03,   1.51750220e+03,
         1.50097262e+03])
```

```
# the top left r × r block , r =2
sort_c1 = eigval2.argsort()[-1]
sort_c2 = eigval2.argsort()[-2]
v_c1 = eigval2[sort_c1]
v_c2 = eigval2[sort_c2]
vec_c1 = eigvec2[:,sort_c1]
vec_c2 = eigvec2[:,sort_c2]
diag2 = np.array([np.sqrt(v_c1),0,0,np.sqrt(v_c2)]).reshape(2,2)
eigvec_r2 = np.concatenate((vec_c1,vec_c2)).reshape(2,10)
#compute V
V_T2 = np.dot(diag2,eigvec_r2)
V2 = np.transpose(V_T2)
V2
```

```
array([[-34.57853091,   1.99507091],
       [ 14.71842113,   3.20967007],
       [ 18.02949507, -12.5108948 ],
       [ 18.18452665, -34.06332469],
       [  8.86315487,  10.89242857],
       [ 11.19577093,  23.25709231],
       [ 14.14462948,  10.25836624],
       [ 10.53515097,   7.14165155],
       [-31.92521855,   2.70195853],
       [-29.16739964, -12.88201869]])
```

So the V matrix for another measure of the similarity of two classes is showed as above. The 2D map of the means of each categories is showed as following.

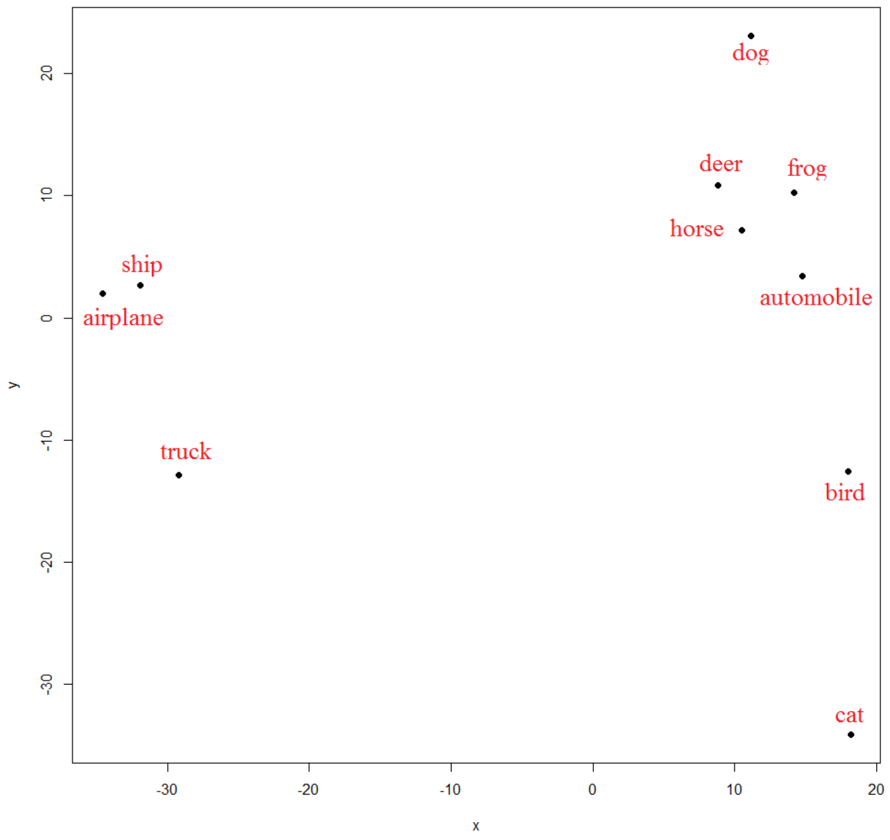

Figure 3: