ECE 661 Homework 8

Suyash Ail
sail@purdue.edu

October 26, 2020

The task of this homework is to implement a simple image classifier using the Gram matrix for texture characterization and Support Vector Machines (SVM) for the classification.

1 Logic

1.1 Gram matrix

In order to extract the features from the image, we convolve the image with C different convolutional kernels. The resulting outputs are called feature maps.

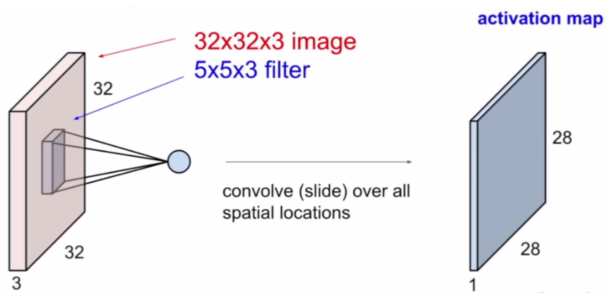

Figure 1: Convolution operation

Considering each feature map as a vector (Flatten the feature map), we take the dot product between the two vectors. The dot product us the information about the relation between them. If the dot product is high it means that the two features are closely related (similar kinds of feature) whereas if the dot product is small , it means that the features are different from each other.

In other words, the lesser the product, the lesser the two features co-occur and the greater it is, the more they occur together. This in a sense gives information about an image's style(texture).

The dot product between the features can be stored in a C x C matrix known as the Gram matrix.

$$G_{i,j} = F_i.F_j$$

where $G_{i,j}$ is the (i,j) element of the Gram matrix and F is the feature vector.

Therefore every image in the dataset can be represented as a feature in the $C^2/2$ dimensional feature space. These features can now be used to train a classifier network. In this exercise, we use the Support Vector Machine (SVM) for the classificaton task.

The best results were achieved for C = 256. Above this value there was no change in the accuracy.

2 Results

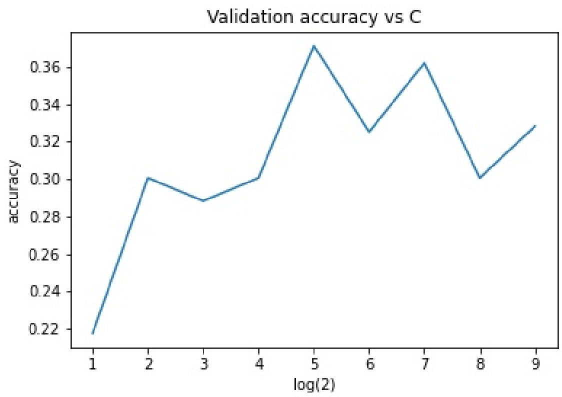

Figure 2: Validation accuracy

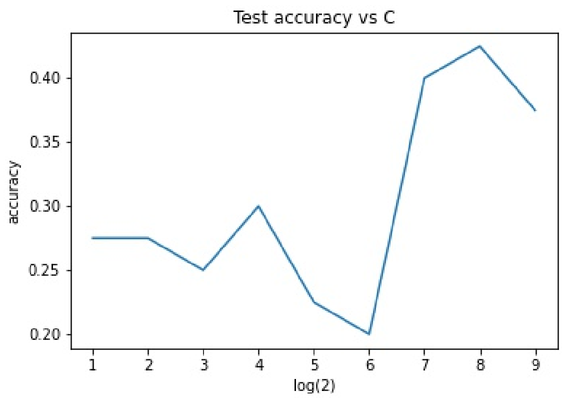

Figure 3: Test accuracy

From the two plots we can assume that the best test accuracy is achieved for C = 256. However the validation accuracy for C = 256 is lower than C=128.

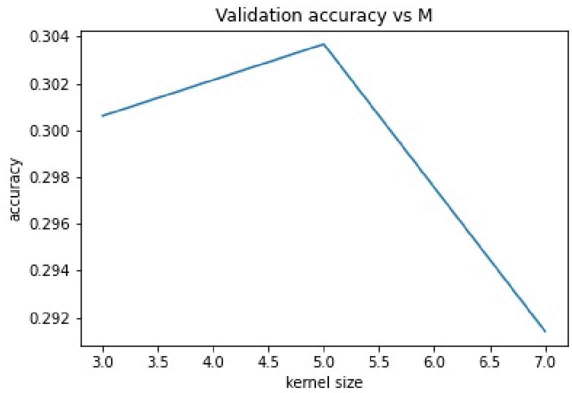

Figure 4: Validation accuracy

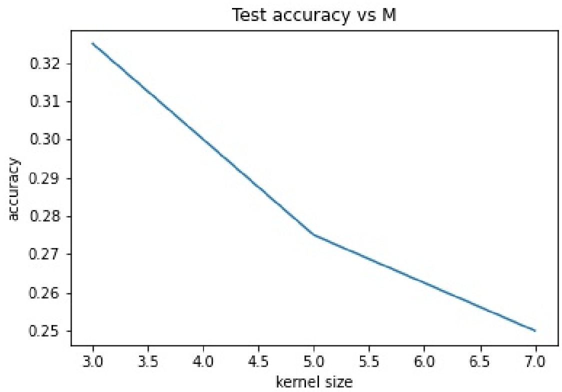

Figure 5: Test accuracy

From this we can see that the best performance is achieved for M = 3.

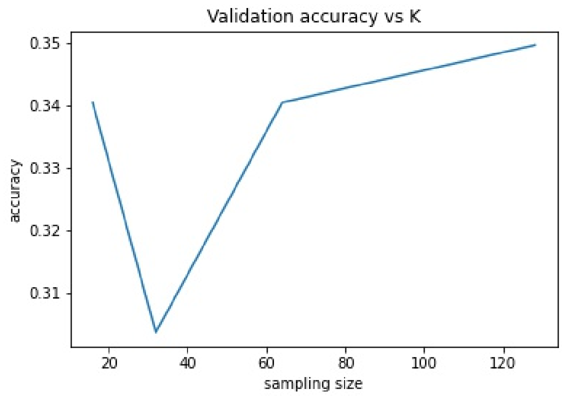

Figure 6: Validation accuracy

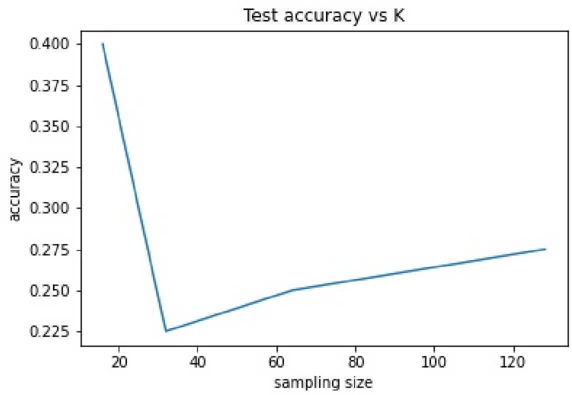

Figure 7: Test accuracy

The results obtained by K=16 and 64 are the same for validation data. The validation accuracy is very high for K=128 but performs poorly on the test data indicating overfitting. Hence k = 16 is the best choice.

2.1 Best model confusion matrix

Feature extractor parameters

1. C = 256
2. M = 3
3. K = 16

Classifier parameters:

1. C = 100
2. gamma = 'scale'
3. kernel = 'rbf'

2.1.1 Validation

$$\text{ConfMat} = \begin{bmatrix} True/Predicted & Cloudy & Rain & Shine & Sunrise \\ Cloudy & 23 & 23 & 11 & 37 \\ Rain & 0 & 42 & 0 & 18 \\ Shine & 14 & 16 & 7 & 25 \\ Sunrise & 21 & 30 & 9 & 50 \end{bmatrix}$$

2.1.2 Testing

$$\text{ConfMat} = \begin{bmatrix} True/Predicted & Cloudy & Rain & Shine & Sunrise \\ Cloudy & 3 & 2 & 0 & 5 \\ Rain & 0 & 8 & 0 & 2 \\ Shine & 0 & 2 & 3 & 5 \\ Sunrise & 1 & 4 & 1 & 4 \end{bmatrix}$$

3 Observations

1. It can be seen from the testing confusion matrix that the classifier performed really well on classifying rain images. However it did not fare well on other classes.

2. One reason might be due to grayscaling the image. Color plays an important role in differentiating between classes however most of the grayscale images have similar intensity (for shine,cloudy and sunrise). That is why most of the shine and cloudy are classified as sunrise.

3. Another reason that I think might be the center cropping of the images. Most of the images contain clouds in them. And center cropping leaves the pixels of the clouds almost always.

4. Another reason is the random initialization of the convolution kernels everytime. The kernel weights are not the optimal values and hence the features extracted may not be the correct representation of the image.

3.1 False Positives and False Negatives

3.2 Cloudy

Figure 8: False Negative (Predicted Rain)

Figure 9: False Positive (Rain but predicted Cloudy)

3.3 Rain

Figure 10: False Negative (Predicted Shine)

Figure 11: False Positive (Shine but predicted Rain)

3.4 Shine

Figure 12: False Negative (Predicted Rain)

Figure 13: False Positive (Rain but predicted Shine)

3.5 Sunrise

Figure 14: False Negative (Predicted Cloudy)

Figure 15: False Positive (Shine but predicted Sunrise)

4 Code Listings

```
# -*- coding: utf-8 -*-
"""hw8_Suyash_Ail.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1WR_TlB6XFLfP-2D4EGBtOCur2TrOlkNn
"""

# import all the necessary libraries
import numpy as np
import cv2
import matplotlib.pyplot as plt
import os
import re
import pickle
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,confusion_matrix

training_dir='/content/drive/My Drive/imagesDatabaseHW8/training/'
testing_dir = '/content/drive/My Drive/imagesDatabaseHW8/testing/'

# function to load images and get class labels
def get_images(dir):
  filenames = os.listdir(dir)
  images = []
  #class labels dictionary
  labels = {'cloudy':0,
            'rain':1,
            'shine':2,
            'sunrise':3
            }
  for filename in filenames:

    image_dir = dir + filename
    img = plt.imread(image_dir)

    classname=re.split('(\d+)',filename)[0]
    classlabel = labels[classname]
    images.append([img,classlabel])
  return images

# get the training and test images
train_images = get_images(training_dir)
test_images = get_images(testing_dir)

# function to get the gram matrix
def gram_matrix(image, C=16,M=3, K=16):
  # convert to grayscale if not already in
  if len(image.shape)>2:
    image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

  #image = cv2.resize(image,(K,K))  #rescaling didnt work out good
```

```
  h,w = image.shape
  image = image[h//2-K//2:h//2+K//2,w//2-K//2:w//2+K//2]  #center crop
  #print(image.shape)
  img_vector =[]
  # run for C times to get C channel maps
  for i in range (C):
    # get random kernel
    w = a = np.random.rand(M,M)
    w = w - np.sum(w)/M**2
    filtered_img = cv2.filter2D(image,-1,w)
    filtered_vector = filtered_img.flatten()
    img_vector.append(filtered_vector)

  gram_mat = np.zeros ((C,C))
  for i in range(len(img_vector)):
    for j in range(i+1):
    #for j in range(len(img_vector)): #trying to see if full matrix helps
   improve accuracy but no
      dot = np.dot(img_vector[i],img_vector[j])
      gram_mat[i,j] = dot

  return gram_mat

# full classification pipeline icluding finding gram vectors and
    classification using SVM

def classification(train_image_list,test_image_list,C,M,K):
  input = np.array([x[0] for x in train_image_list])
  label = [x[1] for x in train_image_list]
  #print(input[0].shape)
  #print(label)

  test_input = [x[0] for x in test_image_list]
  test_label = [x[1] for x in test_image_list]
  train_gram_list=[]
  # get gram matrix for all training images
  for img in input:
    gm = gram_matrix(img,C,M,K)
    gm = np.array(gm).flatten()
    train_gram_list.append(gm)
  #print(train_gram_list)
  test_gram_list=[]
  # get gram matrix for all testing images
  for img in test_input:
    gm = gram_matrix(img,C,M,K)
    gm = np.array(gm).flatten()
    test_gram_list.append(gm)

  ########## Start the Classification task ###############
  X_train,X_val,y_train,y_val = train_test_split(train_gram_list,label,
   test_size = 0.3,shuffle=True)
  X_test,y_test = test_gram_list, test_label

  clf = make_pipeline(StandardScaler(),SVC(C=100,gamma='scale',kernel='rbf'))
  clf.fit(X_train, y_train)
  y_pred = clf.predict(X_val)
  acc = accuracy_score(y_val,y_pred)
  y_test_pred = clf.predict(X_test)
```

```
    test_acc = accuracy_score(y_test,y_test_pred)
    return acc,test_acc

#C = [2,4,8,16,32,64,128,256,512]
M = [3,5,7]
acc_list =[]
test_acc_list =[]
for m in M:
    print("current C=",c)
    acc,test_acc = classification(train_images,test_images,256,m,16)
    acc_list.append(acc)
    test_acc_list.append(test_acc)

acc,test_acc

plt.plot(range(1,10),acc_list)
plt.xlabel('log(2)')
plt.ylabel('accuracy')
plt.title('Validation accuracy vs C')
plt.savefig("/content/drive/My Drive/imagesDatabaseHW8/val_acc.jpg")

plt.plot(range(1,10),test_acc_list)
plt.xlabel('log(2)')
plt.ylabel('accuracy')
plt.title('Test accuracy vs C')
plt.savefig("/content/drive/My Drive/imagesDatabaseHW8/test_acc.jpg")
```