Project 4

Ziyang Wang
73086246

December 15, 2018

Basic Principles

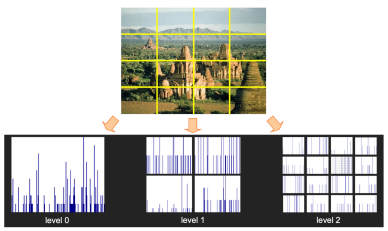
- Bag of words based image representation.
- Spatial pyramid matching based imagerepresentation.
- SVM classification

Implementation Details

- Extract SIFT features from all training data.
- Using kmeans to reduce the number of features to ”visual words”.
- Quantize SIFT features using visual vocabulary.
- Represent images by frequencies of ”visual words”.
- Using SVM classifier to train data by their histogram and categories.

Spatial Pyramid

Partition the image into sub-regions and computing histograms of local features found inside each sub-region when representing. Sum the histograms of different weight as training data.



Lazebnik,Schmid & Ponce(CVPR 2006)

Result

Because of my computer's performance, it costs a lot to train a model on the whole CalTech-256 dataset, I choose 3 categories for training at the beginning. The minimum number of images in a category is 80, so I choose 80 images in all categories for trainning and testing.

After training with 15,30,45,60 each category as training sets respectively, I found that the larger the trainingset, the greater the accuracy.

```
.*
optimization finished, #iter = 53
nu = 0.783438
obj = -12.515455, rho = -0.244168
nSV = 30, nBSV = 8
.*
optimization finished, #iter = 43
nu = 0.977545
obj = -14.903317, rho = -0.028142
nSV = 30, nBSV = 13
.*
optimization finished, #iter = 49
nu = 0.841677
obj = -13.750910, rho = 0.197993
nSV = 30, nBSV = 10
Total nSV = 45
Accuracy = 100% (45/45) (classification)
Accuracy = 56.4103% (110/195) (classification)
```

15/80 images for training

```
.*
optimization finished, #iter = 110
nu = 0.829017
obj = -26.613075, rho = -0.196607
nSV = 59, nBSV = 22
.*
optimization finished, #iter = 92
nu = 0.946044
obj = -29.139254, rho = -0.064254
nSV = 60, nBSV = 24
.*
optimization finished, #iter = 107
nu = 0.889590
obj = -28.471802, rho = 0.134300
nSV = 60, nBSV = 24
Total nSV = 90
Accuracy = 100% (90/90) (classification)
Accuracy = 57.3333% (86/150) (classification)
```

30/80 images for training

```
>> proj4
.*
optimization finished, #iter = 161
nu = 0.806943
obj = -39.135982, rho = -0.220517
nSV = 89, nBSV = 30
.*
optimization finished, #iter = 136
nu = 0.943707
obj = -43.476766, rho = -0.066462
nSV = 90, nBSV = 35
.*
optimization finished, #iter = 168
nu = 0.865412
obj = -41.635166, rho = 0.157044
nSV = 90, nBSV = 34
Total nSV = 135
Accuracy = 100% (135/135) (classification)
Accuracy = 59.0476% (62/105) (classification)
```

45/80 images for training

```
.*
optimization finished, #iter = 217
nu = 0.806209
obj = -52.547804, rho = -0.227358
nSV = 119, nBSV = 40
.*
optimization finished, #iter = 182
nu = 0.943343
obj = -58.168094, rho = -0.068121
nSV = 120, nBSV = 41
.*
optimization finished, #iter = 227
nu = 0.849293
obj = -54.464482, rho = 0.165159
nSV = 120, nBSV = 41
Total nSV = 180
Accuracy = 100% (180/180) (classification)
Accuracy = 63.3333% (38/60) (classification)
```

60/80 images for training

Meanwhile, the higher the vocabulary, the higher the accuracy. However, increasing the vocabulary will greatly increase the computing time.

```
.*
optimization finished, #iter = 185
nu = 0.982599
obj = -59.370890, rho = -0.017461
nSV = 120, nBSV = 51
*
optimization finished, #iter = 75
nu = 0.999823
obj = -59.994209, rho = -0.000156
nSV = 120, nBSV = 34
.*
optimization finished, #iter = 180
nu = 0.983276
obj = -59.410644, rho = 0.017252
nSV = 120, nBSV = 52
Total nSV = 180
Accuracy = 100% (180/180) (classification)
Accuracy = 35% (21/60) (classification)
```

k=512

```
.*
optimization finished, #iter = 212
nu = 0.906319
obj = -56.374463, rho = -0.100161
nSV = 120, nBSV = 45
.*
optimization finished, #iter = 172
nu = 0.992340
obj = -59.808448, rho = -0.008475
nSV = 120, nBSV = 56
.*
optimization finished, #iter = 208
nu = 0.918103
obj = -57.065911, rho = 0.088560
nSV = 120, nBSV = 47
Total nSV = 180
Accuracy = 100% (180/180) (classification)
Accuracy = 45% (27/60) (classification)
```

k=1024

```
.*
optimization finished, #iter = 217
nu = 0.806209
obj = -52.547804, rho = -0.227358
nSV = 119, nBSV = 40
.*
optimization finished, #iter = 182
nu = 0.943343
obj = -58.168094, rho = -0.068121
nSV = 120, nBSV = 41
.*
optimization finished, #iter = 227
nu = 0.849293
obj = -54.464482, rho = 0.165159
nSV = 120, nBSV = 41
Total nSV = 180
Accuracy = 100% (180/180) (classification)
Accuracy = 63.3333% (38/60) (classification)
```

k=2048

After computing spatial pyramids representation of the dataset, the accuracy has not only not increased, but has declined.

```
.*
optimization finished, #iter = 200
nu = 0.888667
obj = -56.324612, rho = 0.122971
nSV = 120, nBSV = 46
Total nSV = 180
Accuracy = 100% (180/180) (classification)
Accuracy = 53.3333% (32/60) (classification)
```

The accuracy of classifying 3 categories after computing spatial pyramids, comparing to 63.3333% with single level features.

Finally, I chose to run 256 categories'classification with a set of parameters with the highest accuracy in the test before.

```
*
optimization finished, #iter = 30
nu = 1.000000
obj = -29.999986, rho = 0.000001
nSV = 60, nBSV = 32
*
optimization finished, #iter = 30
nu = 1.000000
obj = -29.999990, rho = 0.000001
nSV = 60, nBSV = 38
*
optimization finished, #iter = 30
nu = 1.000000
obj = -30.000000, rho = 0.000000
nSV = 60, nBSV = 58
Total nSV = 7710
Accuracy = 100% (7710/7710) (classification)
Accuracy = 2.25681% (58/2570) (classification)
```

The accuracy is just 2.25681%. After changing test class to 30% of whole dataset, and changing vl_dsift to vl_sift, the accuracy increased to 10.2685% for 256 categories.

Since the insufficient performance of my computer, I didn't do further training and parameter adjustment.

```
*.*
optimization finished, #iter = 311
nu = 0.222222
obj = -48.672914, rho = 0.915153
nSV = 192, nBSV = 34
*.*
optimization finished, #iter = 330
nu = 0.228571
obj = -45.440633, rho = 0.754173
nSV = 206, nBSV = 32
Total nSV = 9131
Accuracy = 63.8272% (5867/9192) (classification)
Accuracy = 10.2685% (2199/21415) (classification)
```

Code

```
clear all;
run('vlfeat-0.9.21/toolbox/vl_setup');
categories = dir('256_ObjectCategories');
categories={categories.name};
categories0=categories(3:length(categories));
categories=categories0;
imgSets=[];
for i =1:length(categories)
    categories{i}=categories{i}(5:length(categories{i}));
    imgSets=[imgSets,imageSet(fullfile('256
        _ObjectCategories',categories0{i}))];
end
imgSets=partition(imgSets,80);
[train,test]=partition(imgSets,60);
trainnum=0;
testnum=0;
for i=1:length(categories)
    trainnum=trainnum+train(i).Count;
    testnum=testnum+test(i).Count;
end
trainpath=cell(trainnum,1);
testpath=cell(testnum,1);
trainlabel=cell(trainnum,1);
testlabel=cell(testnum,1);
traincount=1;
testcount=1;
for i=1:length(categories)
    for j=1:train(i).Count
        trainpath{traincount}=train(i).ImageLocation{j};
        trainlabel{traincount}=categories{i};
        traincount=traincount+1;
    end

    for j=1:test(i).Count
```

```
34          testpath{testcount}=test(i).ImageLocation{j};
35          testlabel{testcount}=categories{i};
36          testcount=testcount+1;
37      end
38  end
39
40  k=2048;
41  features=[];
42  featurenum=zeros(1,trainnum);
43  for i=1:trainnum
44      image=imread(trainpath{i});
45      if max(size(image,1),size(image,2))>512
46          if size(image,1)>size(image,2)
47              image=imresize(image,[512 NaN]);
48          else
49              image=imresize(image,[NaN 512]);
50          end
51      end
52
53      if size(image,3)>1
54          image=rgb2gray(image);
55      end
56      [f,d]=vl_dsift(single(image),'step',8,'size',16);
57      features=[features,d];
58      featurenum(i)=size(f,2);
59  end
60
61  [dic,index]=vl_kmeans(double(features),k);
```

Bag of Feature

```
1   h=zeros(trainnum,k);
2   count=1;
3   for i=1:trainnum
4       for j=1:featurenum(i)
5           h(i,index(count))=h(i,index(count))+1;
6           count=count+1;
7       end
8   end
9
10
```

```
forest = vl_kdtreebuild(dic);
testhist=zeros(testnum,k);
for i=1:testnum
    image=imread(testpath{i});
    if max(size(image,1),size(image,2))>512
        if size(image,1)>size(image,2)
            image=imresize(image,[512 NaN]);
        else
            image=imresize(image,[NaN 512]);
        end
    end
    if size(image,3)>1
        image=rgb2gray(image);
    end
    [f,d]=vl_dsift(single(image),'step',8,'size',16);
    [testindex,testdist]=vl_kdtreequery(forest,dic,double
        (d));
    for j=1:length(testindex)
        testhist(i,testindex(j))=testhist(i,testindex(j))
            +1;
    end
end
```

Spatial Pyramid

```
h=zeros(trainnum,k);
for i=1:trainnum
    image=imread(trainpath{i});
    if max(size(image,1),size(image,2))>512
        if size(image,1)>size(image,2)
            image=imresize(image,[512 NaN]);
        else
            image=imresize(image,[NaN 512]);
        end
    end
    if size(image,3)>1
        image=rgb2gray(image);
    end
    height=size(image,1);
    width=size(image,2);
    for j=0:2
```

```
17          temp_width=floor(width/(2^j));
18          temp_height=floor(height/(2^j));
19          for m=1:2^j
20              for l=1:2^j
21                  imgtemp=image((l-1)*temp_height+1:l*
                        temp_height,(m-1)*temp_width+1:m*
                        temp_width);
22                  [f,d]=vl_dsift(single(imgtemp),'step',8,'
                        size',16);
23                  trainindex=vl_kdtreequery(forest,dic,
                        double(d));
24                  for n=1:length(trainindex)
25                      h(i, trainindex(n))=h(i,trainindex(n)
                            )+1/2^j;
26                  end
27              end
28          end
29      end
30  end
31
32  testhist=zeros(testnum,k);
33  for i=1:testnum
34      image=imread(testpath{i});
35      if max(size(image,1),size(image,2))>512
36          if size(image,1)>size(image,2)
37              image=imresize(image,[512 NaN]);
38          else
39              image=imresize(image,[NaN 512]);
40          end
41      end
42      if size(image,3)>1
43          image=rgb2gray(image);
44      end
45      height=size(image,1);
46      width=size(image,2);
47      for j=0:2
48          temp_width=floor(width/(2^j));
49          temp_height=floor(height/(2^j));
50          for m=1:2^j
51              for l=1:2^j
52                  imgtemp=image((l-1)*temp_height+1:l*
                        temp_height,(m-1)*temp_width+1:m*
```

```
                    temp_width);
53                  [f,d]=vl_dsift(single(imgtemp),'step',8,'
                        size',16);
54                  testindex=vl_kdtreequery(forest,dic,
                        double(d));
55                  for n=1:length(testindex)
56                      testhist(i, testindex(n))= testhist(i
                            , testindex(n))+1/2^j;
57                  end
58              end
59          end
60      end
61  end
```

```
1   Y=zeros(trainnum,1);
2   for i=1:length(categories)
3       for j=1:trainnum
4           if strcmp(trainlabel(j),categories(i))
5               Y(j)=i;
6           end
7       end
8   end
9
10  testY=zeros(testnum,1);
11  for i=1:length(categories)
12      for j=1:testnum
13          if strcmp(testlabel(j),categories(i))
14              testY(j)=i;
15          end
16      end
17  end
18  addpath('/Applications/MATLAB_R2018a.app/toolbox/libsvm
        -3.23/matlab');
19
20  model=libsvmtrain(Y,h);
21
22  [predict,accuracy,prob]=libsvmpredict(Y,h,model);
23  [predict,accuracy,prob]=libsvmpredict(testY,testhist,
        model);
```