1 Toy Example of Image Retrieval(1)

1.1 Bag of Visual Words Representation

I am going to demonstrate how to represent an image using histogram of its constituent visual words. But before that, I would like to do some review in order to fix the terminology that I are going to use through out this article.

To the computer, an image is just an array of pixels(an interesting article about the common misunderstanding that a pixel is a little square, written by Alvy Ray Smith, can be found here.) But this low level representation corresponds poorly to our perceived 'meaning' of the images - there is a *semantic gap* between our perception and the representation. Instead of talking about the pixel of an image, we would like to talk about the colors, textures, shapes, and other visual features. Along this line of effort, the computer vision(CV) community has created many ways to describe and quantify different visual features, which we call *visual descriptors*.

I am going to use the **Scale-Invariant Feature Transform (SIFT)** descriptor developed by David Rowe. For our purpose here, we only have to know that the SIFT algorithm detects various interesting points in an image, and encodes the local visual feature around the points as vectors. At this stage of work, we have transformed the raw pixel represention of an image into a collection of vectors that describe the local visual features of some interesting points in the image.

Given all the descriptors of a collection of images, those that describe similar visual features can be grouped together. The resultant groups can be thought as typical local features. In practice, we will calculate some kind of 'average value' of all the descriptors within a group and use it to represent the group. These groups or 'average values' are called **visual words** because they constitue an image similar to the words constitue a text. At the previous stage, we have transform an image into a collection of descriptors; at this stage, we have assigned each descriptors a group. As a result, we can count how many times a typical visual feature, represented by the average value of the group, occurs in any given image. In another word, we have constructed a histogram of typical visual features for every images. In the machine learning literature, the grouping of the descriptors is called **clustering**; the assignment of descriptors to their corresponding group, **vector quantization**; and the histogram that represents the image, **bag of visual words representation** or **vector space representation**.

I have included the Python code that implements the ideas above here. You can execute them inside IPython notebook to verify the result.

```
#loading required libraries

import os
import sift
from sklearn.preprocessing import normalize
from sklearn import cluster
import cPickle
import scipy.sparse as sp
```

I have selected 10 pictures from each of the 10 first categories of Caltech101 datasets. They are inside the folder images/Training_Images. Here are 3 examples from the training dataset.

(you might not be able to see the images if you are using the pdf version of this file)

Next, I calculate the SIFT descriptor for each of the images and store them in the same folder

```
# the conversion takes some time, so you can skip this step
# i have included the .sift file in this repository
impath = '../images/Training_Images/'
feapath = '../images/sift_file/'

imlist = os.listdir(impath)[1:]
sift_list = map(lambda x: x.replace('jpg','sift'),imlist)

for i in range(len(imlist)):
    sift.process_image(os.path.join(impath,imlist[i]),os.path.join(feapath,sift_list[i]))
```

To prepare for clustering, I stack the descriptor together and normalize it.

```
# v is the list of all sift descriptor
v = [sift.read_features_from_file(os.path.join(feapath,sift_list[i]))[1] for i in arange
    (100)]

# we stack all the descriptors in the list v together
# the result is a matrix
feature_vector = vstack(v)

# we normalize the matrix
n_feature_vector = normalize(feature_vector)

#save the matrix for further use
save('n_feature_vector',n_feature_vector)
```

Now we perform the clustering using k-mean clustering algorithm implemented in the scikit library.

```
k = cluster.KMeans(k=100)
k.fit(n_feature_vector[::10])
```

```
KMeans(copy_x=True, init='k-means++', k=100, max_iter=300, n_init=10,
    n_jobs=1, precompute_distances=True,
    random_state=<mtrand.RandomState object at 0x24e830>, tol=0.0001,
    verbose=0)
```

Save the result to disk since it is computationaly intensive to train the learner.

```
with open('learner.pkl','wb') as f:
    cPickle.dump(k,f)
```

Now we can represent the images using the 100 visual words.

```
with open('learner.pkl','rb') as f:
    k = cPickle.load(f)
```

```
# we normalize the sift descriptors so that
# they are comparable to our learning result.
```

```
n_n = map(normalize,v)

# we assign the corresponding visual word to
# each of the descriptors
q = [k.predict(x) for x in n_n]
```

Anticipating the result is sparse array, I write some helper function to calculate the norm and cosine distance.

```
# count the number of typical features,
# and form the bag of visual words representation
def vector_sparray(vec):
    array = sp.dok_matrix((100,1),dtype=float32)
    for v in vec:
        if (v,0) in array:
            array[v,0] = array[v,0] + 1
        else:
            array[v,0] = 1

    return array

def snorm(svector):
    return sqrt(svector.T.dot(svector)[0,0])

def cosine_distance(v1,v2):
    return 1 - (v1.T.dot(v2))[0,0]/(snorm(v1)*snorm(v2))
```

```
# the bag of visual words representation for all images
Histogram = map(vector_sparray,q)

# the representation for the first image
h1 = Histogram[0]
```

```
#calculate the cosine distance from h1 to every other images
distance = map(lambda x: cosine_distance(h1,x),Histogram)
```

```
# return the top 10 results that are nearest to the first image
d=dict(enumerate(distance))
sorted(d,key=d.get)[:10]
```

```
[0, 8, 1, 3, 6, 2, 5, 91, 55, 90]
```

The first 10 images in the dataset is similar, and our top 10 result contains 7 of them.

1.1.1 TODO : More data analysis to justify the representation