A4 Report

2017313260 이재민

1. Environment

Python version 3.7.4

Scikit-learn version 0.23.2

2. Code

```
N = 1000
cluster = 1000
```

N is number of pictures, cluster is number of clusters in k-mean

```
sift_set = np.ndarray([N], dtype=np.object)
```

sift_set store sift vectors each picture.

```
vector_num = []
```

vector_num store number of sift vectors each picture.

```
for i in range(N):
    sift_set[i] = np.fromfile("./sift/sift10" + str(i).zfill(4),
                              dtype=np.uint8).astype(np.int16).reshape(-1, 128)
    vector_num.append(sift_set[i].shape[0])
```

Store sift vectors and store number of vectors each file.

sift_set reshape for dimension 128

```
features = np.vstack(sift_set).astype(np.double)
kmeans = MiniBatchKMeans(n_clusters=cluster, batch_size=200, verbose=10).fit(features)
```

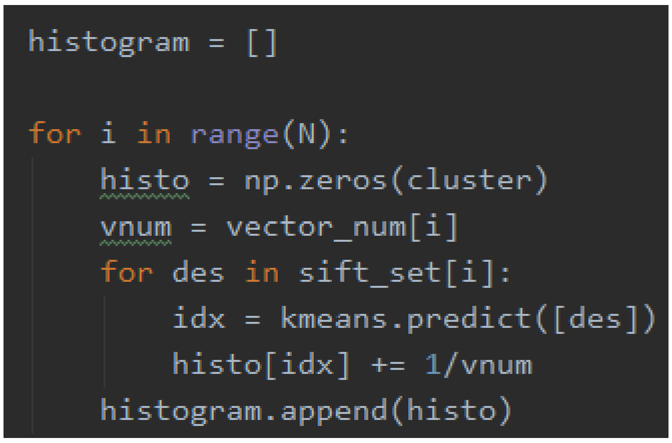
features is np.ndarray, which can fit MiniBatchMeand function, made from sift_set. It has all vectors.

MiniBatchMeans

- n_cluster is number of clusters.

- batch_size is size of the mini batches. It makes calculate faster.

- verbose is logging rate. (optional)



```python
histogram = []

for i in range(N):
    histo = np.zeros(cluster)
    vnum = vector_num[i]
    for des in sift_set[i]:
        idx = kmeans.predict([des])
        histo[idx] += 1/vnum
    histogram.append(histo)
```

histogram has all histogram each picture.

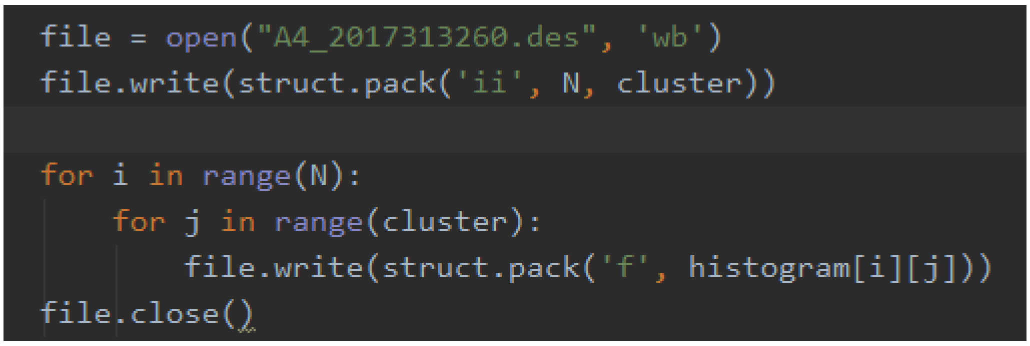
histo is histogram for one picture.

vnum is number of vectors for one picture.

(cluster == D)

idx is cluster index each vector.

Not Just histo[idx] += 1. It has to divided by vnum. Because each picture has another number of vectors. So, check for distribution, it must be normalize. This is necessary to increase accuracy.



```python
file = open("A4_2017313260.des", 'wb')
file.write(struct.pack('ii', N, cluster))

for i in range(N):
    for j in range(cluster):
        file.write(struct.pack('f', histogram[i][j]))
file.close()
```

Create a .des file according to the form.

3. Result

Accuracy value selected after 3 times running code.

N = 1000, cluster = 1000

Not normalize -> accuracy = 2.54000

Normalize -> accuracy = 3.40800

Increasing batch_size reduces KMean time. However, the difference is not significant because the predict function takes longer than the fitting time. (Increasing batch_size does not have a significant effect on accuracy because of the large number of data. I tested for 20~500)

Cluster = 800 -> Accuracy = 3.364000

Cluster = 500 -> Accuracy = 3.206000

Although there were slight variations depending on the number of clusters, the change was greater depending on the clustering efficiency of the MiniBatchKMeans function. Of course, number of clusters is important too.