기계학습 텀프로젝트 5

19011494 조국희

05. 비디오를 이용한 행동 분류

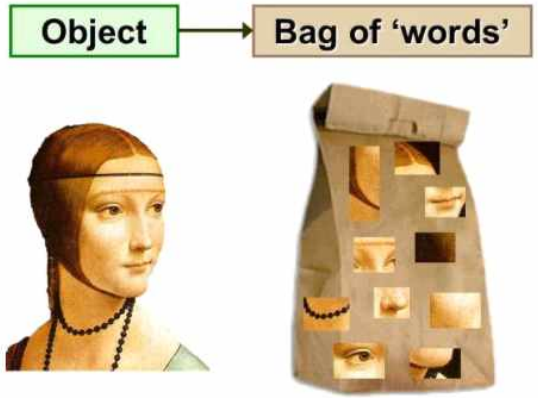
- **프로젝트 목적**

  3D 비디오 데이터를 Handcrafted Feature로 기술하는 법을 알 수 있음

- **비디오**
  - ✓ 비디오는 여러 개의 프레임으로 구성
  - ✓ 비디오 내 단일 프레임의 feature를 여러가지 방식로 기술
  - ✓ BoWV 방법, VLAD 방법 적용

05. 비디오를 이용한 행동 분류

- **Bag of Word Visual**
  1) 모든 영상의 특징점 검출
  2) 모든 특징점들의 Codebook 계산 (중요한/대표적인 특징들을 선정)
  3) 모든 영상과 Codebook을 비교해서 histogram (빈도수) 계산
  4) 분류기를 사용해 학습 및 예측



SIFT

영상 내에서 환경 변화에 강인한 부분,
즉 엣지부분을 찾아 특징점으로 추출

DenseSIFT

영상 내 일정 간격의 위치를 특징점으로
선정하고 이에 대해 기술

05. 비디오를 이용한 행동 분류

- **DenseSift**

```
if dense:
    img = data[i]
    step_size = 8
    # 전체이미지에 대한 keypoint 추출
    kp = [cv2.KeyPoint(x, y, step_size) for x in range(0, img.shape[0], step_size) for y in range(0, img.shape[1], step_size)]
    # SIFT 객체 생성
    sift = cv2.SIFT_create()
    # keypoint에 해당하는 특징점 추출
    _, desc = sift.compute(img, kp)
```

- **특징점들의 대표 특징점 선정**

```
def clustering(train_desc, test_desc=None, n_clusters=200):

    kmeans = MiniBatchKMeans(n_clusters = n_clusters, random_state = 0).fit(train_desc)
    clusters = kmeans.cluster_centers_    ← 클러스터 중심점
    ##########################################
    train_pred = kmeans.predict(train_desc)
    if test_desc is not None:
        test_pred = kmeans.predict(test_desc)
    else:
        test_pred = None
    return train_pred, test_pred, clusters, kmeans
```

05. 비디오를 이용한 행동 분류

- **BOW**

alloc : 한 프레임의 특징점(visual word)들이 대표 특징점(codebook)에 할당된 위치

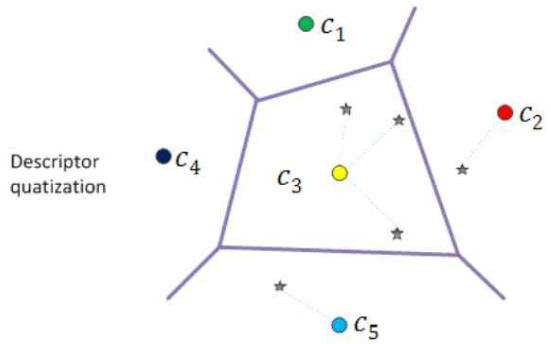
n_cluster : 대표 특징점(codebook)의 수

```
def BoW(alloc, n_cluster):

    V, _ = np.histogram(alloc, bins = range(n_cluster + 1), normed = True)
    V = V.flatten()

    return V
```

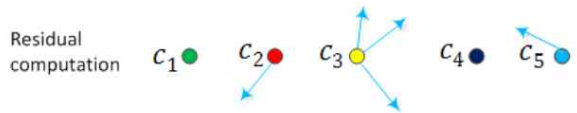
모든 영상과 Codebook을 비교해서 histogram (빈도수) 계산

05. 비디오를 이용한 행동 분류

- **VLAD**



각 특징점들을 대표 특징점에 거리순 할당



벡터차 계산

동일한 대표 특징점에 할당된 벡터차 합산

$$V(k) = \sum_{i=1}^{N} a_k(\mathbf{x}_i)(x_i - c_k)$$

X의 대표 특징점이 중심 클러스터(c)와 같은 군집일때만 1, 아닐때는 0

Cluster center

```
def VLAD(X, alloc, centers):
    # centers --> codebook
    m,d = X.shape # d가 descriptor 개수
    k = centers.shape[0] # cluster center 개수

    # VLAD feature를 담기 위한 변수
    V=np.zeros([k,d])

    # VLAD에서 V[i]는 Xi의 모든 descriptor에 대해 해당하는 cluster center와의 차이를 계산해 더한 값
    for i in range(k):
        if np.sum(alloc==i)>0: # 대표특징점으로 할당된 특징점이 중심 클러스터와 일치하는게 한개 이상이라도 있으면
            # 그 중심 클러스터로 할당된 X들의 descriptor 값들에서 실제 중심 클러스터의 descriptor 값들 차를 계산해 합함
            V[i] = np.sum(X[alloc==i] - centers[i], axis = 0)
```

05. 비디오를 이용한 행동 분류

- **비디오의 모든 프레임에서 얻은 feature를 평균내어 비디오 feature 생성 및 분류**

```
X_mean = []
test_mean = []
y_mean = []
# train_global_id는 0-2019까지 있으므로, 각 id에 해당하는 train_global_desc값들의 평균을 구해 X_mean, test_mean 구함
for i in range(2020):                                    train_global_desc 에는 5frame씩 2020개의 비디오가 존재
    X_mean.append(np.mean(train_global_desc[train_global_id==i], axis = 0))
for i in range(505):                                     test_global_desc 에는 5frame씩 505개의 비디오가 존재
    test_mean.append(np.mean(test_global_desc[test_global_id==i], axis = 0))

# 같은 라벨이 중복되므로 5개씩 중복되므로 5씩 건너뛰며 y_mean에 라벨값 저장
for i in range(0, 10100, 5):
    y_mean.append(train_global_label[i])

# list 형식에서 numpy array 형식으로 변환
X_mean = np.array(X_mean)
test_mean = np.array(test_mean)
y_mean = np.array(y_mean)
```

+ SVC에 대한 GridSearchCV를 이용해 최적화된 파라미터를 찾아 예측

05. 비디오를 이용한 행동 분류

- **모든 프레임으로 예측 진행 후 예측치 중 가장 많은 빈도를 나타낸 행동을 선정**

```
X_v = train_global_desc
test_v = test_global_desc
y_v = train_global_label

# 모든 데이터에 대해 SVC로 학습 및 예측을 한 후
params = {
    'C' : [3, 2, 4, 5],
    'kernel' : ['rbf']

}                    GridSearchCV를 이용해 최적화된 파라미터를 찾아 예측
clf = GridSearchCV(SVC(random_state = 0), params, cv=3)
clf.fit(X_v, y_v)
svm_predict = svc.predict(test_v)

# 각 비디오의 5개 Frame의 예측값들 중 가장 빈도가 높게 나온 값을 구해 svm_finalpredict에 추가해줌
svm_finalpredict=[]
for i in range(0, 2521, 5):        mode함수를 사용해 최빈값을 반환(2525->505 결과값 얻음)
    svm_finalpredict.append(mode(svm_predict[i:i+5])[0][0])

svm_finalpredict = np.array(svm_finalpredict)
svm_predict = svm_finalpredict
```

05. 비디오를 이용한 행동 분류

- **프레임 feature에서 대표되는 feature를 선정 후 BoW or VLAD 방식으로 비디오 feature를 기술**

```
train_global_alloc, test_global_alloc, global_codebook, global_kmeans = clustering(train_global_desc, test_global_desc, args_global_cluster)
```

비디오의 프레임에서 대표되는 feature 선정하기 위해 clustering

```
# Train 비디오 내 프레임 별로 기술된 이미지 feature를 기반으로 한번 더 기술하여(한번 더 BoW 혹은 VLAD)
# 각 비디오에 대한 비디오 feature 기술
# Empty Module 7과 관련있으며, 5,6과는 무관
train_video_desc = []
train_video_desc_key = []
for vid_name in train_vid_names_u:
    cind = np.where(vid_name==train_vid_names)[0]
    if args_aggr=="bow":
        hist_desc = BoW(train_global_alloc[cind], args_global_cluster)
    elif args_aggr=="vlad":
        hist_desc = VLAD(train_global_desc[cind], train_global_alloc[cind], global_codebook)
    else:
        import pdb; pdb.set_trace()

    train_video_desc.append(hist_desc)
    train_video_desc_key.append(vid_name)
train_video_desc = np.asarray(train_video_desc)
train_video_desc_key = np.asarray(train_video_desc_key)
```

각 비디오에 대한 특징적 feature 추출

```
# 분류를 위해, 행동 분류에 대한 각 train 비디오 별 label 가공
train_video_id = np.array([int(i.split("/")[-1].split(".")[0]) for i in train_video_desc_key])
train_video_label = []
for fid in train_video_id:
    cind = np.where(train_csv_arr[:, 0]==fid)[0]
    clsname = train_csv_arr[cind, 1]
    cinfo_ind = np.where(classinfo_arr[:, 1] == clsname)[0]
    train_video_label.append(classinfo_arr[cinfo_ind, 0].astype("int"))
train_video_label = np.asarray(train_video_label).ravel()
```

label 가공

+ vlad일 경우 PCA 해주고 모델학습을 통한 예측 진행

05. 비디오를 이용한 행동 분류

▪ 성능 향상을 위한 시도

- ✓ 해상도 높임
- ✓ Clustering 하는 군집 개수 늘림
- ✓ PCA 차원 수 조절

베이스라인	성능
VLAD	0.28118
BoW	0.20990

모델	성능
SVM + BoW+average	0.22178
SVM + BoW+vote	0.21584
SVM + BoW+BoW	0.09108
SVM + VLAD+average	0.29504
SVM + VLAD+vote	0.27722
SVM + VLAD+VLAD(PCA1200)	0.21782
SVM + VLAD+average (cluster250, PCA200)	0.30099
SVM + VLAD+average (해상도(156,156), cluster250, PCA200)	0.32277