머신러닝 실습

-기말 프로젝트

전자공학과
2022144007
김의진

데이터 통계 및 클래스 분포

총 데이터 개수: 7000개

클래스별 데이터 개수:

청사과	(0):	700개
사과	(1):	700개
바나나	(2):	700개
블랙베리	(3):	700개
오이	(4):	700개
오렌지	(5):	700개
복숭아	(6):	700개
배	(7):	700개
토마토	(8):	700개
수박	(9):	700개

Training set , Validation set 비율
→ 7:3

Training set 데이터 수: 4900개
(각 클래스별 490개)
Validation set 데이터 수: 2100개
(각 클래스별 210개)

전체 코드 흐름 요약

1. select_feature()

1) 특징 추출

2) 특징 추출 데이터 기반 클러스터링

3) 특정 목표 클래스 중심에 대한 거리 특징 추가

2. Training, Validation set 분할

1) 7 : 3 비율로 분할

3. Training set data 이용한 Two Layer Neural Network 학습

1) Hidden Layer node 수 80, Learning rate 0.01, epoch 1000, batch size 1

2) Weight 수: 1370개 (w_hidden(v): 560개, w_output(w): 810개)

4. 학습된 모델 검증

1) 학습된 weights로 Validation set을 이용한 forward propagation

2) Confusion matrix를 이용한 recall, precision, accuracy 확인

select_feature() 상세 구현

```python
def select_features(directory):
    #이미지 파일 directory


    #이미지 파일 directory 안의 파일 이름들 문자열 리스트로 ;
    file_list = os.listdir(directory)

    #특징 저장할 list
    feature_1_list = []
    feature_2_list = []
    feature_3_list = []
    feature_4_list = []

    label = []      #정답 라벨

    # file_list에 있는 값들 반복
    for name in file_list:

        #경로 설정함
        path = os.path.join(directory, name)

        #라벨 불러오기(파일명 첫 숫자)
        label.append(int(name.split('_', 1)[0]))

        #이미지 읽고 RGB(red, green, blue)값으로 변환하기
        img_GRB = cv2.imread(path)
        img_RGB = cv2.cvtColor(img_GRB, cv2.COLOR_BGR2RGB)

        #특징추출하기

        # 특징 1: 초록색 분류기
        feature_1 = No1_feature(img_RGB)

        # 특징 3: 빨간색 분류기
        feature_2 = No2_feature(img_RGB)

        # 가로 전체 밝기 분산 ==> 줄무늬같은 패턴 탐지
        feature_3 = No3_feature(img_RGB)

        # 빨간색밝기 평균에 대한 파랑, 초록 밝기 평균 비율
        feature_4 = No4_feature(img_RGB)

        #각 리스트에 각특징값 append
        feature_1_list.append(feature_1)
        feature_2_list.append(feature_2)
        feature_3_list.append(feature_3)
        feature_4_list.append(feature_4)

    #각 특징 리스트들 numpy array로 변환
    feature_1_list = np.array(feature_1_list)
    feature_2_list = np.array(feature_2_list)
    feature_3_list = np.array(feature_3_list)
    feature_4_list = np.array(feature_4_list)

    #features 한군데에 모으기
    features = np.column_stack([feature_1_list, feature_2_list, feature_3_list, feature_4_list])

    #clustering 이용
    cluster_features, m = clustering(features)

    #복숭아 군집 중심에 대한 거리 뽑는 함수
    peach_dist = extract_class_distance(features, label, cluster_features, m, target_label = 6)

    #토마토 군집 중심에 대한 거리 뽑는 함수
    tomato_dist = extract_class_distance(features, label, cluster_features, m, target_label = 8)

    #복숭아, 토마토, 사과 구분위한 특징까지 포함한 features
    features = np.column_stack([features, peach_dist, tomato_dist])

    # features 정규화
    features = standard_data(features)

    return features, label                                  #features와 label 반환
```

1. 색체, 패턴 4개 특징 추출
2. 특징 추출 데이터 클러스터링
3. 복숭아 군집, 토마토 군집에 대한 거리 특징 추가
4. 총 특징 추출 데이터 **정규화**
5. 특징 추출 데이터 features와 label 반환

clustering() 로직

```python
def clustering(data):

    K = 10
    n, c = data.shape

    rand_idx = np.random.choice(n, K, replace=False)
    m = data[rand_idx].copy()

    while(1):
        m_prev = m.copy()

        clus = np.zeros(n)

        # 한 데이터의 특징 수만큼 반복
        for i in range(n):
            d = (np.sum(((m - data[i]) ** 2), axis = 1)) ** (1 / 2)
            clus[i] = np.argmin(d)

        #군집 개수만큼 반복
        for j in range(K):
            clus_p = data[clus == j]

            #j군집에 아무것도 없을 경우 방지
            if len(clus_p) > 0:
                m[j] = np.mean(clus_p, axis = 0)

        #새로운 중심과 이전 중심과 비교(소수점이라 loop 탈출 못할 가능성 존재
        if np.allclose(m_prev, m):
            break

    return clus, m
```

1. 군집 개수 10으로 설정
→ 분류 class 10개이기 때문에 이렇게 설정

2. 군집 중심 data중 하나로 random choice

3. 각 데이터를 가장 가까운 거리(유클리드, norm 2)에 할당함

4. 각 군집에 할당된 데이터들의 평균을 구해 새로운 군집 중심으로 update

5. 이전 군집 중심과 비교하여 위치가 거의 같아지면 update 중단

6. 군집 데이터와 군집 중심 반환

extract_class_distance() 로직

```python
def extract_class_distance(data, labels, clus, m, target_label):
    labels = np.array(labels, dtype=int)
    clus = np.array(clus, dtype=int)
    max_count = 0

    #m 길이만큼 반복
    for k in range(m.shape[0]):
        cluster_labels = labels[clus == k]
        count = np.sum(cluster_labels == target_label)
        if count > max_count:
            class_clus = k
            max_count = count

    center = m[class_clus]
    d = np.log1p(np.sum(((center - data) ** 2), axis = 1)) ** (1 / 2)
    d = d.reshape(-1, 1)

    return d
```

1. Data label과 군집 데이터 numpy array로 변환, 군집 데이터 셀 count 초기화

2. 각 군집에 목표 class 개수 계산, 제일 많이 속한 군집 저장

3. 2번에서 저장한 군집 중심에 대한 데이터들의 거리(유클리드, norm 2) 계산

4. 거리 데이터 반환

분류 전/후 Confusion Matrix (validation set)

예측데이터

Recall

	0	1	2	3	4	5	6	7	8	9	10
0	201	0	0	0	0	0	0	0	0	0	1
1	0	196	0	0	0	0	2	0	0	0	0.989899
2	0	0	221	0	0	0	0	0	0	0	1
3	0	0	0	198	0	0	0	0	0	0	1
4	0	0	0	0	213	0	0	0	0	0	1
5	0	0	0	0	0	219	0	0	0	0	1
6	0	30	0	0	0	0	188	0	2	0	0.854545
7	0	0	0	0	0	0	0	231	0	0	1
8	0	1	0	0	0	0	2	0	190	0	0.984456
9	0	0	0	0	0	0	0	0	0	206	1
10	1	0.863436	1	1	1	1	0.979167	1	0.989583	1	0.982381

실제 데이터

Precision

accuracy

전체적인 recall, precision 모두 0.9가 넘고 accurac도 0.94임

하지만 class 사과(1), 복숭아(6), 토마토(8)에 대한 분류가 아쉬움

→ 이 세 class에 대한 데이터의 분포가 비슷함

Training set
MSE ≈ 0.003
Accuracy ≈ 0.983

분류 전/후 Confusion Matrix (validation set)

	0	1	2	3	4	5	6	7	8	9	10
0	198	0	0	0	0	0	0	0	0	0	1
1	0	194	0	0	0	0	10	0	0	0	0.95098
2	0	0	247	0	0	0	0	1	0	0	0.995968
3	0	0	0	237	0	0	0	0	0	0	1
4	0	0	0	0	196	0	0	0	0	0	1
5	0	0	0	0	0	211	0	0	0	0	1
6	0	7	0	0	0	0	201	0	3	0	0.952607
7	0	0	0	0	0	0	0	200	0	0	1
8	0	0	0	0	0	0	4	0	199	0	0.980296
9	0	0	0	0	0	0	0	0	0	192	1
10	1	0.965174	1	1	1	1	0.934884	0.995025	0.985149	1	0.988095

Training set
MSE ≈ 0.003
Accuracy ≈ 0.987

특징 추출 데이터 기반으로 한 클러스터링 이용 특징 추가 후
- 몇몇 class의 recall 및 precision이 감소
- Class 1의 Precision, Class 6의 Recall 눈에 띄게 증가

→ 전체적으로 데이터 분류가 균형 잡힘, accuracy 소폭 상승

정량 성능 개선

	전		후	
Precision				
-Class 1 :	0.863436	→	0.965174	0.101738 증가
-Class 6 :	0.979167	→	0.934884	0.044283 감소
-Class 7 :	1	→	0.905025	0.094975 감소
-Class 8 :	0.989583	→	0.985149	0.004434 감소
Recall				
-Class 1 :	0.989899	→	0.98098	0.008918 감소
-Class 2 :	1	→	0.995968	0.004032 감소
-Class 6 :	0.864545	→	0.952607	0.088062 증가
-Class 8 :	0.984456	→	0.980296	0.00416 감소
Accuracy				
	0.982381	→	0.988095	0.005714 증가

개선점

Clustering을 이용한 거리 기반 특징으로 극적인 효과를 보지 못함

다른 방식으로 clustering 이용한 특징 추출로 교체 or 추가
-히스토그램을 이용한 클러스터링
1. 원리
-1) 이미지 채널의 픽셀 분포를 히스토그램으로 요약함
-2) 이 데이터를 k-means clustering
-3) 각 이미지가 속한 군집과의 특징 추가
ex) 거리
2. 기대효과
-1) 세밀한 분포차이 포착 가능
-2) 위의 효과에 의한 유사 분포 클래스 미세 분리 도움