Practice: Texture classification

Long Bin Jin

Artificial Inteligence
& Computer Vision
Laboratory

Texture Classification

1. Feature extraction

2. Bayesian classifier

Feature extraction



- Data 구성
  - brick, grass, ground 세 개 클래스 사용
- X_train: (300, 11)
  - 한 이미지에서 랜덤하게 10개 작은 이미지 patch를 crop
  - 특징: GLCM(dissimilarity, correlation)(2)
    Laws' texture (TEM)(9)
- Y_train: (300)
  - Label encoding
  - brick:0, grass:1, ground:2

```
texture_date
|----train
|       |----brick
|       |       brick1.jpg
|       |       brick2.jpg
|       |       ... ...
|       |       brick10.jpg
|       |----grass
|       |       grass1.jpg
|       |       grass2.jpg
|       |       ... ...
|       |       grass10.jpg
|       |----ground
|       |       ground1.jpg
|       |       ground2.jpg
|       |       ... ...
|       |       ground10.jpg
|       |----water
|       |       water1.jpg
|       |       water2.jpg
|       |       ... ...
|       |       water10.jpg
|       |----wood
|               wood1.jpg
|               wood2.jpg
|               ... ...
|               wood10.jpg
|----test
        |----brick
        |       brick1.jpg
        |       brick2.jpg
        |       ... ...
        |       brick50.jpg
        |----grass
        |       grass 1.jpg
        |       grass 2.jpg
        |       ... ...
        |       grass 50.jpg
        |----ground
        |       ground 1.jpg
        |       ground2.jpg
        |       ... ...
        |       ground50.jpg
        |----water
        |       water1.jpg
        |       water2.jpg
        |       ... ...
        |       water50.jpg
        |----wood
                wood1.jpg
                wood2.jpg
                ... ...
                wood50.jpg
```

Feature extraction



- Data 구성
  - brick, grass, ground 세 개 클래스 사용
- X_test: (150, 11)
  - 특징: GLCM(dissimilarity, correlation)(2)

    Laws' texture (TEM)(9)
- Y_test: (150)
  - Label encoding
  - brick:0, grass:1, ground:2

```
texture_date
|----train                          |----test
|       |----brick                        |----brick
|       |       brick1.jpg                |       brick1.jpg
|       |       brick2.jpg                |       brick2.jpg
|       |       ... ...                   |       ... ...
|       |       brick10.jpg               |       brick50.jpg
|       |----grass                        |----grass
|       |       grass1.jpg                |       grass 1.jpg
|       |       grass2.jpg                |       grass 2.jpg
|       |       ... ...                   |       ... ...
|       |       grass10.jpg               |       grass 50.jpg
|       |----ground                       |----ground
|       |       ground1.jpg               |       ground 1.jpg
|       |       ground2.jpg               |       ground2.jpg
|       |       ... ...                   |       ... ...
|       |       ground10.jpg              |       ground50.jpg
|       |----water                        |----water
|       |       water1.jpg                |       water1.jpg
|       |       water2.jpg                |       water2.jpg
|       |       ... ...                   |       ... ...
|       |       water10.jpg               |       water50.jpg
|       |----wood                         |----wood
|               wood1.jpg                        wood1.jpg
|               wood2.jpg                        wood2.jpg
|               ... ...                          ... ...
|               wood10.jpg                       wood50.jpg
```

Bayesian classifier

$$g_i(X) = -\frac{1}{2}(X-\mu_i)^t\Sigma_i^{-1}(X-\mu_i) - \frac{1}{2}\ln|\Sigma_i| + \ln P(\omega_i)$$

- X: 테스트 데이터
- μ_i: i번째 클래스의 평균 벡터
- Σ_i: i번째 클래스의 공분산 행렬
- $P(\omega_i)$: i번째 클래스의 사전확률

- Y_pred = $argmax_i(g_i(X))$, for i in all classes

공분산 행렬 계산 (예시)

$$\begin{bmatrix}\sigma_{11} & \sigma_{12}\\ \sigma_{21} & \sigma_{22}\end{bmatrix} = \begin{bmatrix}cov(X,X) & cov(X,Y)\\ cov(Y,X) & cov(Y,Y)\end{bmatrix} = \begin{bmatrix}\frac{1}{n}\sum(x_i - m_x)(x_i - m_x) & \frac{1}{n}\sum(x_i - m_x)(y_i - m_y)\\ \frac{1}{n}\sum(y_i - m_y)(x_i - m_x) & \frac{1}{n}\sum(y_i - m_y)(y_i - m_y)\end{bmatrix}$$

$data$: [[1,2], [3,4], [5,4], [3,6]]

$X = [1,3,5,3]\ \ m_x = 3$

$Y = [2,4,4,6]\ \ m_y = 4$

$n = 4$

$$\frac{1}{n}\sum(x_i - m_x)(x_i - m_x) = 2$$
$$\frac{1}{n}\sum(x_i - m_x)(y_i - m_y) = 1$$
$$\frac{1}{n}\sum(y_i - m_y)(x_i - m_x) = 1$$
$$\frac{1}{n}\sum(y_i - m_y)(y_i - m_y) = 2$$

$$\begin{bmatrix}\sigma_{11} & \sigma_{12}\\ \sigma_{21} & \sigma_{22}\end{bmatrix} = \begin{bmatrix}2 & 1\\ 1 & 2\end{bmatrix}$$

Bayesian classifier

- 가상환경 활성화 및 모듈 설치

- numpy==1.19.3
  matplotlib
  scikit-image
  scipy
  opencv-python==3.4.2.16
  opencv-contrib-python==3.4.2.16
  sklearn

```
(CV) D:\>pip install sklearn
```

필요한 모듈 import
(Bayesian.py)


```
1   from sklearn.metrics import accuracy_score, confusion_matrix    # 정확도 계산, confusion matrix 계산 함수
2   from skimage.feature import greycomatrix, greycoprops
3   import matplotlib.pyplot as plt
4   from scipy import signal as sg
5   import itertools                                                # confusion matrix 시각화 함수에서 사용
6   import numpy as np
7   import cv2
8   import os
9
```

Laws' texture 계산 함수


```python
# === laws texture 계산 함수 ===
def laws_texture(gray_image):
    (rows, cols) = gray_image.shape[:2]
    smooth_kernel = (1/25)*np.ones((5,5))                                  # smoothing filter 만들기
    gray_smooth = sg.convolve(gray_image ,smooth_kernel,"same")            # 흑백이미지 smoothing하기
    gray_processed = np.abs(gray_image - gray_smooth)                      # 원본이미지에서 smoothing된 이미지 빼기

    filter_vectors = np.array([[ 1,  4,  6,  4, 1],                       # L5
                               [-1, -2,  0,  2, 1],                       # E5
                               [-1,  0,  2,  0, 1],                       # S5
                               [ 1, -4,  6, -4, 1]])                      # R5
    filters = []                                                           # 16(4x4)개 filter를 저장할 filters
    for i in range(4):
        for j in range(4):
            filters.append(np.matmul(filter_vectors[i][:].reshape(5,1), # 매트릭스 곱하기 연산을 통해 filter값 계산
                                     filter_vectors[j][:].reshape(1,5)))

    conv_maps = np.zeros((rows, cols,16))                                  # 계산된 convolution 결과를 저장할 conv_maps
    for i in range(len(filters)):
        conv_maps[:, :, i] = sg.convolve(gray_processed,                   # 전처리된 이미지에 16개 필터 적용
                                         filters[i],'same')

    # === 9+1개 중요한 texture map 계산 ===
    texture_maps = list()
    texture_maps.append((conv_maps[:, :, 1]+conv_maps[:, :, 4])//2)     # L5E5 / E5L5
    texture_maps.append((conv_maps[:, :, 2]+conv_maps[:, :, 8])//2)     # L5S5 / S5L5
    texture_maps.append((conv_maps[:, :, 3]+conv_maps[:, :, 12])//2)    # L5R5 / R5L5
    texture_maps.append((conv_maps[:, :, 7]+conv_maps[:, :, 13])//2)    # E5R5 / R5E5
    texture_maps.append((conv_maps[:, :, 6]+conv_maps[:, :, 9])//2)     # E5S5 / S5E5
    texture_maps.append((conv_maps[:, :, 11]+conv_maps[:, :, 14])//2)   # S5R5 / R5S5
    texture_maps.append(conv_maps[:, :, 10])                            # S5S5
    texture_maps.append(conv_maps[:, :, 5])                             # E5E5
    texture_maps.append(conv_maps[:, :, 15])                            # R5R5
    texture_maps.append(conv_maps[:, :, 0])                             # L5L5 (use to norm TEM)

    # === Law's texture energy 계산 ===
    TEM = list()
    for i in range(9):
        TEM.append(np.abs(texture_maps[i]).sum() /                      # TEM계산 및 L5L5 값으로 정규화
                   np.abs(texture_maps[9]).sum())
    return TEM                                                          # 9차원의 TEM feature 추출: list
```

Train 이미지에서 특징 추출


```python
# === 이미지 패치에서 특징 추출 ===
train_dir = './texture_data/train'                                  # train data 경로
test_dir = './texture_data/test'                                    # test data 경로
classes = ['brick', 'grass', 'ground']                              # 클래스 이름

X_train = []                                                        # train 데이터를 저장 할 list
Y_train = []                                                        # train 라벨을 저장 할 list

PATCH_SIZE = 30                                                     # 이미지 패치 사이즈
np.random.seed(1234)
for idx, texture_name in enumerate(classes):                        # 각 class 마다
    image_dir = os.path.join(train_dir, texture_name)               # class image가 있는 경로
    for image_name in os.listdir(image_dir):                        # 경로에 있는 모든 이미지에 대해
        image = cv2.imread(os.path.join(image_dir, image_name))     # 이미지 불러오기
        image_s = cv2.resize(image, (100, 100), interpolation=cv2.INTER_LINEAR)         # 이미지를 100x100으로 축소

        for _ in range(10):                                         # 이미지에서 random하게 10개 패치 자르기
            h = np.random.randint(100-PATCH_SIZE)                   # 램덤하게 자를 위치 선정
            w = np.random.randint(100-PATCH_SIZE)

            image_p = image_s[h:h+PATCH_SIZE, w:w+PATCH_SIZE]       # 이미지 패치 자르기
            image_p_gray = cv2.cvtColor(image_p, cv2.COLOR_BGR2GRAY)    # 이미지를 흑백으로 변환
            #image_hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV))    # 이미지를 HSV로 변환
            glcm = greycomatrix(image_p_gray, distances=[1], angles=[0], levels=256,     # GLCM co-occurrence 계산
                                symmetric=False, normed=True)
            X_train.append([greycoprops(glcm, 'dissimilarity')[0, 0],
                            greycoprops(glcm, 'correlation')[0, 0]]     # GLCM dissimilarity, correlation 특징 추가 (2차원)
                            + laws_texture(image_p_gray))               # laws texture 특징 추가 (9차원)
            Y_train.append(idx)                                         # 라벨 추가

X_train = np.array(X_train)                                         # list를 numpy array로 변경
Y_train = np.array(Y_train)
print('train data:  ', X_train.shape)                               # (500, 11)
print('train label: ', Y_train.shape)                               # (500)
```

Test 이미지에서 특징 추출


```
88  X_test = []                                                    # test 데이터를 저장 할 list
89  Y_test = []                                                    # test 라벨을 저장 할 list
90
91  for idx, texture_name in enumerate(classes):                   # 각 class 마다
92      image_dir = os.path.join(test_dir, texture_name)           # class image가 있는 경로
93      for image_name in os.listdir(image_dir):                   # 경로에 있는 모든 이미지에 대해
94          image = cv2.imread(os.path.join(image_dir, image_name))# 흑백으로 불러오기
95          image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
96          glcm = greycomatrix(image_gray, distances=[6], angles=[0], levels=256,          # GLCM co-occurrence 계산
97                              symmetric=False, normed=True)
98          X_test.append([greycoprops(glcm, 'dissimilarity')[0, 0],
99                         greycoprops(glcm, 'correlation')[0, 0]]  # GLCM dissimilarity, correlation 특징 추가 (2차원)
100                        + laws_texture(image_gray))              # laws texture 특징 추가 (9차원)
101         Y_test.append(idx)                                      # 라벨 추가
102
103 X_test = np.array(X_test)                                      # list를 numpy array로 변경
104 Y_test = np.array(Y_test)
105 print('test data:  ', X_test.shape)                            # (250, 11)
106 print('test label: ', Y_test.shape)                            # (250)
```

Bayesian classifier

$$g_i(X) = -\frac{1}{2}(X-\mu_i)^t\Sigma_i^{-1}(X-\mu_i) - \frac{1}{2}\ln|\Sigma_i| + \ln P(\omega_i)$$

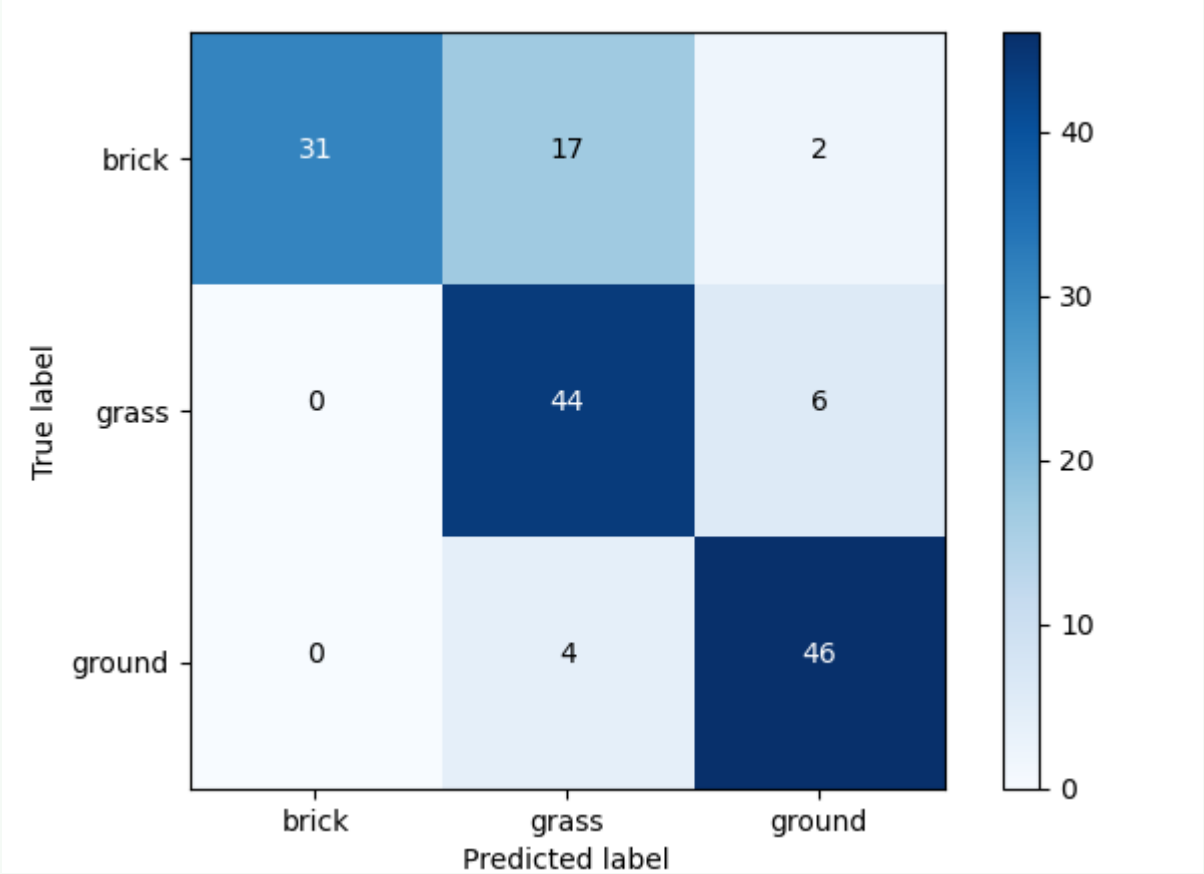
```
priors = []
covariances = []
means = []
# === Bayesian classifier ===
for i in range(len(classes)):                                   # 각 클래스 마다
    X = X_train[Y_train == i]                                   # i번째 클래스 데이터를 X에 저장
    priors.append((len(X) / len(X_train)))                      # priors에 사전확률 저장
    means.append(np.mean(X, axis=0))                            # means에 평균값 저장
    covariances.append(np.cov(np.transpose(X), bias=True))      # covariances에 공분산 값 저장

# === likelihood 계산 함수 ===
def likelihood(x, prior, mean, cov):
    return -0.5 * np.linalg.multi_dot([np.transpose(x-mean), np.linalg.inv(cov), (x-mean)]) - 0.5 * np.log(np.linalg.det(cov)) + np.log(prior)

Y_pred = []                                                     # 예측 데이터를 저장 할 list
for i in range(len(X_test)):                                    # 각 테스트 데이터에 대해
    likelihoods = []                                            # 모든 클래스에 대한 likelihood를 저장할 list
    for j in range(len(classes)):
        likelihoods.append(likelihood(X_test[i], priors[j], means[j], covariances[j]))  # 모든 클래스의 likeelihood 저장
    Y_pred.append(likelihoods.index(max(likelihoods)))          # 그중 likelihood가 제일 큰 값을 정답으로 Y_pred에 추가
acc = accuracy_score(Y_test, Y_pred)                            # test라벨과 예측라벨을 비교하여 정확도 계산
print('accuracy: ', acc)
```

Confusion matrix visualization


```python
# === confusion matrix 시각화 ===
def plot_confusion_matrix(cm, target_names=None, labels=True):
    accuracy = np.trace(cm) / float(np.sum(cm))

    cmap = plt.get_cmap('Blues')

    plt.figure(figsize=(6, 4))
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.colorbar()
    thresh = cm.max() / 2

    if target_names is not None:
        tick_marks = np.arange(len(target_names))
        plt.xticks(tick_marks, target_names)
        plt.yticks(tick_marks, target_names)

    if labels:
        for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
            plt.text(j, i, "{:,}".format(cm[i, j]),
                     horizontalalignment="center",
                     color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.show()

plot_confusion_matrix(confusion_matrix(Y_test, Y_pred), target_names=classes)   # confusion matrix 시각화
```

Accuracy

- accuracy: 0.8066666666666666



성능 향상을 위해 조절 할 수 있는 변수

- Train 이미지 patch 크기와 개수
- GLCM 특징
  - i와 j의 위치 관계
  - ‘contrast’, ‘dissimilarity’, ‘homogeneity’, ‘energy’, ‘correlation’, ‘ASM’
- Laws texture 특징
  - 16개 특징 (혹은 더 적은 특징)
- Color 특징
  - B, G, R 값