머신러닝

학번: 2016722074

학과: 컴퓨터정보공학과

이름: 김영태

과제 제목 및 목적

Naïve Bayesian Classifier을 직접 디자인하고 flower dataset을 이용해 직접 데이터를 분류해 보는 실습을 진행한다. 해당과제를 진행하면서 전체적인 classifier의 구조를 이해하고 classifier의 동작 과정을 이해할 수 있다. 해당 과제의 데이터셋에는 총 4개의 특성이 있다. 4개의 특성이 있는 데이터셋을 사용해서 데이터들의 Likelihood와 Prior를 구하고 이 값들을 베이즈 정리를 사용해서 Posterior 값을 구한다.

소스코드 설명

```
def feature_normalization(data): # 10 points
    # parameter
    feature_num = data.shape[1]
    data_point = data.shape[0]

    # you should get this parameter correctly
    normal_feature = np.zeros([data_point, feature_num])
    mu = np.zeros([feature_num])
    std = np.zeros([feature_num])

    # your code here
    mu = np.mean(data, axis=0)  #Find mean value in each feature
    std = np.std(data, axis=0)  #Find variane value in each feature

        # Caculate normal_features
    for i in range(0, data_point):
        for j in range(0, feature_num):
            normal_feature[i][j] = (data[i][j] - mu[j]) / std[j]

    print(normal_feature)

    # end

    return normal_feature
```

먼저 feature_normalization 함수이다.

데이터를 인자로 전달받고 이 데이터의 각 feature마다 분산과 평균을 구한다. 또 이때 구한 평균과 분산으로 이 데이터의 normalization을 진행한다. 이때 이 값은 데이터에서 평균을 빼고 이 값을 분산으로 나눈 값이다. 최종적으로 원본데이터를 변환한 normal_feature을 리턴한다.

```python
def get_normal_parameter(data, label, label_num): # 20 points
    # parameter
    feature_num = data.shape[1]

    # you should get this parameter correctly
    mu = np.zeros([label_num,feature_num])
    sigma = np.zeros([label_num,feature_num])

    # your code here

        # 3-dimensional data
    labeled_data = np.zeros([data.shape[0], label_num, feature_num])

    for i in range(0, data.shape[0]):
        for k in range(0, feature_num):
            _label = label[i]
            labeled_data[i][_label][k] = data[i][k] #data labeling

    mu = np.mean(labeled_data, axis=0)      # mean value in labeled data
    sigma = np.std(labeled_data, axis=0)    # std value in labeld data

    # end

    return mu, sigma
```

다음은 get_normal_parameter 함수이다. 함수의 인자로 데이터와 레이블값, 레이블의 개수를 전달받는다. 레이블된 데이터를 저장해 함수 내에서 사용하기 위해 3차원 배열 데이터 labeled_data를 선언하고 0으로 초기화했다. 이 배열은 함수의 인자로 전달받은 label의값에 해당하는 값에 따라 데이터를 레이블링해 새로운 데이터(레이블링된)가 만들어진다. 이제 이 배열을 data_point 축을 따라 평균과 분산 값을 구하고 이 두 값을 반환했다.

```python
def get_prior_probability(label, label_num): # 10 points
    # parameter
    data_point = label.shape[0]

    # you should get this parameter correctly
    prior = np.zeros([label_num])

    # your code here

    for i in range(0, data_point):  # total data num (X num)
        idx = label[i]
        prior[idx] += 1;            # counting x in each class

    prior = prior/data_point        # probability in each class(status)


    # end
    return prior
```

다음은 Prior를 구하는 함수이다. 이 함수는 레이블 값을 전달받아 사전지식, 즉 prior를 계산한다. 여기서 prior란 각 클래스에 속할 확률을 의미하는데, 이 과제를 예시로 들면 총 3개의 클래스가 있으므로 이 prior배열의 길이는 3이고 이 배열의 모든 값을 더하면 1이된다. prior의 각 클래스에

해당하는 값은 각 클래스에 데이터가 속할 확률을 의미한다. 따라서 코드를 보면 천체 레이블의 길이를 구하고, 각 클래스마다 몇 개의 데이터가 있는지 숫자를 세서 확률을 구했다.

```
def Gaussian_PDF(x, mu, sigma): # 10 points
    # calculate a probability (PDF) using given parameters
    # you should get this parameter correctly
    pdf = 0

    # your code here

    var = float(sigma)**2        # variance
    denom = (2*np.pi*var)**.5    # square root (2pi*variance)
    num = np.exp(-(float(x)-float(mu))**2/(2*var)) # exponential
    pdf = num/denom              # get pdf value
    # end

    return pdf

def Gaussian_Log_PDF(x, mu, sigma): # 10 points
    # calculate a probability (PDF) using given parameters
    # you should get this parameter correctly
    log_pdf = 0

    # your code here

    var = float(sigma)**2
    denom = (2*np.pi*var)**.5
    num = np.exp(-(float(x)-float(mu))**2/(2*var))
    log_pdf = np.log(num/denom) # Log gaussian value
    # end

    return log_pdf
```

다음은 pdf의 값을 구하는 함수이다. 첫번째 함수는 가우시안 함수인데 이 함수는 평균과 분산만 있다면 pdf를 구할 수 있다. 따라서 pdf를 구하기 위한 평균과 표준편차값을 함수의 인자로 전달 받는다. 또 데이터의 Likelihood값을 구하는 것이 목적이기 때문에 데이터 x역시 주어진다. 가우시안 함수는 e의 –(x-평균)^(2/2*분산)을 루트 2*pi*분산 으로 나눈 값 이기 때문에 numpy내장함수를 사용해서 코드를 구현했다. 밑의 가우시안 로그 pdf의 경우는 앞의 가우시안 pdf의 값에 로그를 취한것이기 때문에 numpy 내장함수를 사용해서 로그를 취한 뒤 값을 리턴했다.

```python
def Gaussian_NB(mu, sigma, prior, data): # 40 points
    # parameter
    data_point = data.shape[0]
    label_num = mu.shape[0]

    # you should get this parameter correctly
    likelihood = np.zeros([data_point, label_num])
    posterior = np.zeros([data_point, label_num])
    ## evidence can be ommitted because it is a constant

    # your code here
        ## Function Gaussian_PDF or Gaussian_Log_PDF should be used in this section


    for i in range(0, label_num):           # To find P(X|w1), P(X|w2), ... , P(X|wn)
        for j in range(0, data_point):      # To find P(X1|w) + P(X2|w) +...+ P(Xn|w)
            likelihood[j][i] = 0            # Set likelihood zero
            for k in range(0, data.shape[1]):   # Add likelihood because of log function
                likelihood[j][i] += Gaussian_Log_PDF(data[j][k], mu[i][k], sigma[i][k])
                # likelihood[j][i] :  P(X1|wi) + P(X2|wi) +...+ P(Xj|wi)
                # if Gaussian_PDF ? : P(X1|wi) * P(X2|wi) *...* P(Xj|wi)

    for i in range(0, label_num):
        for j in range(0, data_point):
            posterior[j][i] = (prior[i]) + likelihood[j][i]
            # posterior : P(wi|X) = P(wi) + P(X1|wi) + P(X2|wi) +...+ P(Xj|wi)

    # end
    return posterior
```

마지막으로 지금까지 구한 Likelihood와 Prior값을 이용해서 Posterior 값을 구하고 각 데이터 포인트에서 어떤 레이블, 즉 어떤 클래스에서의 값이 제일 높은 지 비교하여 어떤 클래스에 속할지 예측한다.

코드의 구현을 보면, 각 레이블은 각 클래스를 의미한다. 각 데이터 포인터에서의 레이블마다의 Likelihood를 구한다. 이때 코드에서 사용한 함수는 Gaussian Log PDF이므로 각 데이터 포인트에서의 pdf값은 로그를 취했으므로 곱셈이 아니라 덧셈을 수행한다. 각 레이블과 데이터 포인트에서의 Likelihood를 구했으므로 이 값에 Prior를 추가해 Posterior값을 계산한다. 계산에 로그 사용했기 때문에 Prior값을 곱하지 않고 더한다. 각 데이터 포인트마다 각 클래스에 속할 확률을 구했으므로 최종적으로 이 Posterior의 값을 비교하여 어떤 클래스에 속할지 결정하게 된다.

실행결과

```
In [181]: runcell(0, 'C:/Users/김영태/Desktop/4-1/머신러닝/Homework_01/Homework_01/main_app.py')
Mean: [5.84333333 3.05733333 3.758      1.19933333]
normalled_Mean: [-4.28176013e-16  2.10202226e-16 -2.78665979e-16 -2.70154269e-17]
accuracy is 72.0% !
the number of correct prediction is 36 of 50 !

In [182]: runcell(0, 'C:/Users/김영태/Desktop/4-1/머신러닝/Homework_01/Homework_01/main_app.py')
Mean: [5.84333333 3.05733333 3.758      1.19933333]
normalled_Mean: [-4.10042370e-16 -7.81597009e-16 -2.37957802e-16 -2.36847579e-17]
accuracy is 54.0% !
the number of correct prediction is 27 of 50 !

In [183]: runcell(0, 'C:/Users/김영태/Desktop/4-1/머신러닝/Homework_01/Homework_01/main_app.py')
Mean: [5.84333333 3.05733333 3.758      1.19933333]
normalled_Mean: [-4.48021250e-16  2.07241631e-16 -3.40468394e-16  2.75705384e-16]
accuracy is 60.0% !
the number of correct prediction is 30 of 50 !

In [184]: runcell(0, 'C:/Users/김영태/Desktop/4-1/머신러닝/Homework_01/Homework_01/main_app.py')
Mean: [5.84333333 3.05733333 3.758      1.19933333]
normalled_Mean: [-4.33357054e-16 -7.95659834e-16 -2.42768768e-16  2.61272485e-16]
accuracy is 88.0% !
the number of correct prediction is 44 of 50 !

In [185]: runcell(0, 'C:/Users/김영태/Desktop/4-1/머신러닝/Homework_01/Homework_01/main_app.py')
Mean: [5.84333333 3.05733333 3.758      1.19933333]
normalled_Mean: [-4.39648318e-16 -7.79376563e-16 -2.73855013e-16  2.61272485e-16]
accuracy is 56.00000000000001% !
the number of correct prediction is 28 of 50 !

In [186]: runcell(0, 'C:/Users/김영태/Desktop/4-1/머신러닝/Homework_01/Homework_01/main_app.py')
Mean: [5.84333333 3.05733333 3.758      1.19933333]
normalled_Mean: [-4.23365047e-16 -8.05281767e-16 -2.49800181e-16 -9.99200722e-18]
accuracy is 68.0% !
the number of correct prediction is 34 of 50 !

In [187]: runcell(0, 'C:/Users/김영태/Desktop/4-1/머신러닝/Homework_01/Homework_01/main_app.py')
Mean: [5.84333333 3.05733333 3.758      1.19933333]
normalled_Mean: [-4.44089210e-16 -7.77156117e-16  2.82736797e-16 -9.62193288e-17]
accuracy is 70.0% !
the number of correct prediction is 35 of 50 !
```

```
In [188]: runcell(0, 'C:/Users/김영태/Desktop/4-1/머신러닝/Homework_01/Homework_01/main_app.py')
Mean: [5.84333333 3.05733333 3.758      1.19933333]
normalled_Mean: [ 5.31611792e-16 -8.01951098e-16 -2.72744790e-16 -7.29046453e-17]
accuracy is 62.0% !
the number of correct prediction is 31 of 50 !

In [189]: runcell(0, 'C:/Users/김영태/Desktop/4-1/머신러닝/Homework_01/Homework_01/main_app.py')
Mean: [5.84333333 3.05733333 3.758      1.19933333]
normalled_Mean: [-4.75360492e-16 -8.20084741e-16 -3.19744231e-16 -3.33066907e-16]
accuracy is 86.0% !
the number of correct prediction is 43 of 50 !

In [190]: runcell(0, 'C:/Users/김영태/Desktop/4-1/머신러닝/Homework_01/Homework_01/main_app.py')
Mean: [5.84333333 3.05733333 3.758      1.19933333]
normalled_Mean: [-4.21144601e-16 -8.03801470e-16  2.02800739e-16 -1.08061708e-16]
accuracy is 56.00000000000001% !
the number of correct prediction is 28 of 50 !

In [191]: runcell(0, 'C:/Users/김영태/Desktop/4-1/머신러닝/Homework_01/Homework_01/main_app.py')
Mean: [5.84333333 3.05733333 3.758      1.19933333]
normalled_Mean: [-4.25585493e-16 -8.09722659e-16 -3.07161703e-16 -3.17523785e-16]
accuracy is 52.0% !
the number of correct prediction is 26 of 50 !

In [192]: runcell(0, 'C:/Users/김영태/Desktop/4-1/머신러닝/Homework_01/Homework_01/main_app.py')
Mean: [5.84333333 3.05733333 3.758      1.19933333]
normalled_Mean: [-4.19664303e-16 -7.86037901e-16  2.70894418e-16  2.76815607e-16]
accuracy is 70.0% !
the number of correct prediction is 35 of 50 !
```

참고문헌

https://numpy.org/doc/stable/index.html

https://ko.wikipedia.org/wiki/%EB%82%98%EC%9D%B4%EB%B8%8C_%EB%B2%A0%EC%9D%B4%EC%A6%88_%EB%B6%84%EB%A5%98

http://www.cs.ucr.edu/~eamonn/CE/Bayesian%20Classification%20withInsect_examples.pdf