Machine learning homework - 1

<Naive Bayesian Classifier Design>

컴퓨터정보공학부

2020202013

고가연

I. 과제 제목 및 목적

본 과제는 주어진 4개의 feature(Petal width, Petal length, Sepal width, Sepal length) 데이터들을 이용해 붓꽃(Iris)의 3가지 종류(Setosa, Versicolor, Virginica)로 분류하는 naïve Bayesian classifier를 설계하는 것이다. Python으로 6개의 함수(feature_normalization, get_normal_parameter, get_prior_probability, Gaussian_PDF, Gaussian_Log_PDF, Gaussian_NB)를 작성하여 Classifier의 구조를 이해하고 설계하는 것을 목적으로 한다.

II. 소스코드에 대한 설명

KGY.py 파일에서 6개의 Python 함수를 작성했고 아래와 같이 순서대로 설명하겠다. main_app.py에서 아래의 작성한 함수들을 호출하면서 Classifier가 동작한다.

1. feature_normalization

   -> iris.csv 파일로 받은 4개의 feature 데이터를 정규화(normalization)하는 단계이다. 데이터를 정규화하는 식은 아래와 같다. $\bar{x}$는 정규화된 데이터를 말하고, x는 원래 데이터, mu는 해당 feature의 평균(mean)이고, σ는 해당 feature의 표준편차(standard deviation)이다.

   $$\bar{x} = \frac{x - mu}{\sigma} - 1$$

   -> 해당 함수는 인자로 data를 받고 이는 iris.csv 파일에서 추출한 150x4인 2차원 배열의 feature 데이터이다. 각 열은 4개의 feature를 나타내고 각 feature에는 총 150개의 데이터가 주어진다. 그리고 mu와 std는 4개 feature의 1차원 배열이고 순서대로 각 feature의 평균과 표준편차를 계산해서 저장했다. 이때 numpy.mean()과 numpy.std()를 사용했고 각 함수의 두번째 인자를 0로 함으로써 data에서 각 열에 따라 산술 평균과 표준편차를 계산한다.

   -> 그리고나서 이중 for문을 통해 2차원 배열인 data에서 요소를 1개씩 불러오면서 python 코드로 작성한 위의 정규화 식을 통해 각 요소(데이터 값)를 정규화하여 normal_feature 2차원 배열에 순서대로 저장하도록 소스코드를 작성했다. 결과적으로 해당 함수는 normal_feature 배열을 반환한다.

2. get_normal_parameter

-> 해당 함수에서는 train data의 label과 feature에 대해 각각 평균과 표준편차 2차원 배열을 계산하는 동작을 하도록 작성했다. 즉, train data의 평균 배열은 3x4 크기로, 각 label 3개에 따라 feature 4개에 대한 평균값을 각각 계산해 저장한다. 마찬가지로 표준편차 배열도 동일한 크기로, 총 12개의 표준편차 값을 계산해 저장한다.

-> se_0array 리스트와 같이 각각 3개의 label과 4개의 feature에 해당하는 데이터 값들을 별도로 저장해주기 위해서 총 12개의 리스트를 선언해주었다. 예를 들어, se_0array의 label은 Setosa이고 feature는 0번 feature(가장 처음 feature)를 말한다. 그리고나서, 이중 for문을 사용해 해당 함수 첫 번째 인자로 받은 data의 한 요소씩 가져오면서 맨 처음 열이라면 feature0라고 판단하고 두 번째 열이라면 feature1, 세 번째 열은 feature2, 네 번째 열은 feature3이라고 판단하도록 작성했다. 이렇게 feature를 알고나서, 해당 함수의 두 번째 인자로 받은 label이라는 1차원 배열에서 각 요소를 가져와서 label 값이 0이면 Setosa이고, 1이면 Versicolor이고, 2이면 Virginica라고 판단하도록 만들었다. 따라서, 해당 데이터의 feature와 label을 알아낸 후, 앞서서 만들었던 해당하는 리스트에 저장하도록 했다.

-> 각 label과 feature에 해당하는 데이터 값을 저장한 12개의 리스트를 가지고 numpy.mean()과 numpy.std() 함수를 사용해 평균과 표준편차를 계산할 수 있었다. 그리고 3*4 크기의 2차원 배열인 mu과 sigma에 순서대로 저장하도록 소스코드를 작성했다. 결과적으로 해당 함수는 mu과 sigma를 반환한다.

3. get_prior_probability

-> 해당 함수에서는 prior probability를 계산한다. 해당 함수의 첫 번째 인자로 받은 label 1차원 배열의 요소를 반복문을 통해 하나씩 가져오면서 label0(Setosa)의 개수와 label1(Versicolor)의 개수, label2(Virginica)의 개수를 셀 수 있다. 그리고나서 처음에 data_point 변수로 전체 label의 개수를 저장해 두었기 때문에 각 label의 개수 나누기 data_point를 하면 각 label의 prior probability를 계산할 수 있다. 즉, 각 label이 나타날 확률을 말한다. 결과적으로 각 label에 해당하는 prior probability를

prior 1차원 배열에 순서대로 저장한 후 반환한다.

4. Gaussian_PDF

   -> 해당 함수는 인자로 x(데이터 값), mu(평균), sigma(표준편차)를 받아서 Python으로 작성한 가우시안 정규분포 수식에 각각 넣어서 계산한다. 결과적으로 나온 pdf 값(likelihood)을 반환한다.

   -> 아래의 수식은 가우시안 분포 수식이다. 이는 데이터와 평균과 분산만 알면 계산할 수 있다. 해당 식을 Python 코드로 작성했다. 이때 제곱근을 계산하기 위해서 numpy.sqrt() 함수를 사용했고, π(=3.14..) 값을 계산하기 위해서 numpy.pi를 작성했다. 또한, numpy.exp() 함수를 통해 지수함수인 exponential 함수를 구현할 수 있었다.

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$$

5. Gaussian_Log_PDF

   -> 해당 함수는 위의 Gaussian_PDF 함수에서 구현한 가우시안 분포 수식에서 로그함수를 취한 수식을 구현하는 것이다. 마찬가지로 데이터 값 x와 평균 mu, 표준편차 sigma를 인자로 받고 Python으로 구현한 수식에 넣어서 likelihood를 계산하는 동작이다.

   -> 기존의 가우시안 분포 수식에서 로그를 취하면 아래와 같다. 자연로그(밑이 e)를 취하면 exponential과 함께 사라지고 더 계산하기 편해진다. 추가적으로 numpy.log() 함수를 사용해 자연로그를 구현하였다.

$$p(x) = \log\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} = \log\frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2$$

6. Gaussian_NB

   -> 해당 함수는 Naïve Bayesian 이론을 사용해서 각 class(label)에 대한 feature vector의 posterior 확률 값을 추정한다. Naïve Bayesian theorem은 아래와 같다. 아래의 수식에서 $p(C_k|x)$ 는 posterior 값을 말하고 $p(C_k)$는 prior probability이고 $p(x|C_k)$는 likelihood 값이다. 앞서 작성한 함수를 통해 prior와 likelihood 값을 가져올 수 있다.

$$p(C_k|x) = \frac{p(C_k)p(x|C_k)}{p(x)}$$

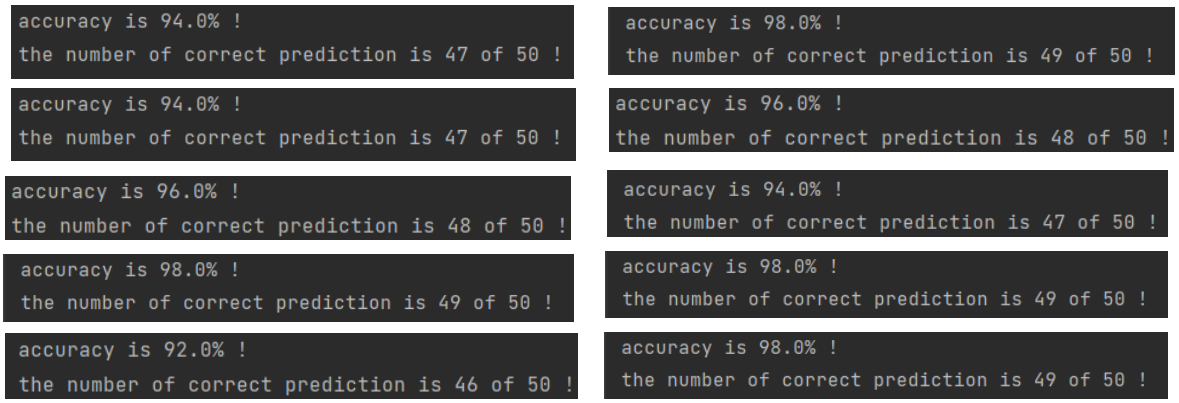
-> 해당 함수의 인자는 mu(평균), sigma(표준편차), prior, data이다. 그리고 data는 50*4 크기의 2차원 배열이다. 별도로 함수 내에서 선언된 likelihood와 posterior은 0으로 채워진 50*3 크기의 2차원 배열이다. 따라서, data의 한 행에 있는 4개의 feature 값들을 가지고 먼저 class0이라고 가정하고 4개의 data 값을 하나씩 가져와서 Gaussian_PDF 함수를 호출해 likelihood를 계산했다. 이때 우리는 모든 사건을 독립이라고 간주했기 때문에 data의 한 행에 있는 4개의 값을 다 돌 때까지 계산한 likelihood를 누적해서 곱했다. 4개의 feature 값을 모두 곱했으면 likelihood 배열에 순서대로 해당하는 class0인 첫 번째 열에 data와 동일한 행 index로 저장했다. 동시에 posterior 배열에는 계산된 likelihood 값과 해당 함수의 인자로 받은 prior에서 class0 확률을 곱한 다음 numpy.log() 함수를 사용해 로그를 취해서 계산하여 저장했다.

-> 위에서 설명한 3중 for문을 통해 class0에 해당하는 50개 데이터들의 (Gaussian_PDF 함수를 호출해) likelihood 값을 1열에 모두 계산해 저장하고 동시에 posterior의 1열도 계산해 저장한다. 그리고나서 class1에 해당하는 50개 데이터들의 likelihood 값과 posterior를 각 배열의 2열에 전부 계산해 저장하고 마지막으로 class2에 해당하는 50개의 데이터들의 likelihood 값과 posterior은 각 배열의 3열에 모두 저장하는 동작을 수행한다. 결과적으로 전부 값이 새로 채워진 posterior 2차원 배열을 반환하는 함수이다.

위의 6개의 함수 외에도 split_data, classifier, accuracy 함수가 존재한다. 먼저, split_data 함수의 동작을 설명하자면, 전체 data를 train data와 test data로 분할하는 함수이다. 세 번째 인자로 split_factor=100을 받고 train data와 train label은 처음부터 100까지 슬라이싱하고 나머지 100부터 마지막까지의 데이터는 test data와 test label로 반환한다. 그리고 classifier 함수는 posterior 값을 가지고 MLE(Maximum Likelihood Estimation)를 적용해 예측 값을 추정하는 동작을 수행한다. 따라서 prediction을 반환한다. 마지막으로 accuracy 함수는 파라미터로 prediction과 test_label을 받아서 예측한 값과 정답 값이 일치하는 경우를 count 해서 hit_num(정답을 맞춘 개수)을 계산하고 정답을 맞춘 확률(정확도; accuracy)과 hit_num을 반환하여 classifier의 정확도 결과를 출력할 수 있게 해준다.

III. 실행결과

다음은 해당 소스코드를 10번 실행한 결과이다. 아래와 같이 붓꽃을 분류하는 classifier의 정확도는 모두 90% 이상으로 나오는 것을 확인할 수 있다.



IV. 참고문헌

https://dhpark1212.tistory.com/entry/Likelihood%EA%B0%80%EB%8A%A5%EB%8F%84

https://www.delftstack.com/ko/api/numpy/python-numpy-mean/