인공지능 과제 리포트

과제 제목: k-Nearest Neighbor 알고리즘을 사용한 Iris classification

학번: B511086
이름: 백성진

1. 과제 개요

본 과제는 k-Nearest Neighbor 알고리즘을 사용하여 Iris 꽃의 종을 판별하는 프로그램을 작성하는 과제입니다. 훈련용 데이터를 사용해서 환경을 설정한 후 새로운 테스트용 데이터가 어느 훈련용 데이터와 가장 근접한지 판별해내는 내용을 다룹니다.

2. 구현 환경

Windows 10, Pycharm
Python 3.8.2

3. 알고리즘에 대한 설명

본 과제에서 사용한 k-Nearest Neighbor(이하 knn)알고리즘에서 사용한 방법은 다음과 같습니다.

테스트 데이터 y와 훈련용 데이터 x의 모든 원소들의 거리를 구합니다.
이때, 거리를 구하기 위해 사용한 방법은 유클리디안 거리 계산 방법입니다.
거리를 계산한 후, 미리 준비한 딕셔너리가, 계산된 거리를 Key로 가지고 Value로 x의 인덱스를 가지도록 저장합니다.

훈련용 데이터 x와의 거리 계산을 마친 후, 기록된 딕셔너리를 Key를 피봇으로 하여 오름차순 정렬합니다. 정렬 함수를 마친 후 반환되는 값을 튜플을 원소로 가지는 리스트로 받아, 리스트의 0번째 원소가 y와 가장 짧은 유클리디안 거리를 가지도록 합니다.

리스트에는 훈련용 데이터 x의 길이만큼의 데이터가 들어있습니다. 이를 사전에 지정한 k개의 원소만 취합니다. 즉, k가 3이라면, 리스트에서 0, 1, 2 까지만 가지도록 설정합니다.

이제 k개의 가장 근접한 원소가 무엇인지 알았으므로, majority vote 방식을 사용하든, weighted majority vote 방식을 사용하든 전혀 무리가 없습니다.

우선, majority vote 방식을 사용하기 위해서는 k번 루프를 돌아야 합니다.
루프 안에서, 새로운 리스트의, 근접 리스트의 각 원소가 가지는 class번째 원소의 값을 1씩 증가 시킵니다. 결과적으로 루프를 탈출한 후, 가장 많은 투표를 받은 꽃의 종이 분류된 iris의 종이 됩니다.

weighted majority vote 방식을 위해서는 가중치 값을 구하는 수식을 우선 정해야 합니다. 본 과제에서, weight는 실제 거리의 역수의 로그 값을 가집니다. 가까운 거리를 가지는 것 일

수록 높은 가중치 값을 가지는데, 이에 로그 값을 취한다면 0과 1 사이로 정규화 되므로 파격적인 가중치를 가지는 것을 방지할 수 있습니다.
majority vote 방식과 비슷하게 루프를 돌 때, class 마다 1씩 증가시키는 방식 대신, weight를 곱한 후 그 값을 더하는 방식을 채택했습니다.

4. 데이터에 대한 설명

4.1 Input Feature

Input feature은 4차수(Sepal width/length, Petal width/length)를 가집니다.
훈련용, 그리고 테스팅 데이터에 상관 없이 동일한 Input feature를 가지며, 각각 cm단위로 저장되어 있습니다. 총 150개의 데이터가 있으며, 그 중 140개를 훈련용 데이터로, 10개를 테스트 데이터로 선정하였습니다.

4.2 Target Output

이번 테스트에서 Target이 되는 클래스는 3가지(setosa, versicolor, verginica)입니다. 모든 입력 데이터는 정확한 종을 한가지 씩 가지며 kNN 알고리즘을 통해 추정된 값과 정확한 종이 일치하는지 판별됩니다.

5. 소스코드에 대한 설명

distance_metric 함수
두 4차원 벡터의 거리를 유클리디안 거리 계산 방식으로 구한다. def distance_metric(self, a, b): dist = 0 for i in range(len(self.features[0])): sub = a[i] - b[i] dist += (sub * sub) return math.sqrt(dist)
훈련 데이터와 실제 데이터와의 거리는 유클리디안 거리 계산 방법을 적용하였습니다. feature가 4개이므로, 4개 원소 각각에 대해 뺄셈과 제곱을 해주는 방식입니다.

majority_vote 함수
일반적으로 근접한 k 개 중 가장 많은 수를 차지한 class 를 선정하는 함수. def majority_vote(self): majority_cnt = [] # class 개수의 길이를 가지는 리스트를 만든다.

```
        for i in range(len(self.iris_names)):
            majority_cnt.append(0)

        # nearest k의 이름에 해당하는 class 의 카운트를 증가시킨다.
        for i in range(self.k):
            majority_cnt[self.target[self.nearest_k[i]]] += 1

        # 가장 많은 꽃 class 가 속한 이름을 찾아 저장한다.
        self.majority_vote_value =
self.iris_names[majority_cnt.index(max(majority_cnt))]
        return self.majority_vote_value
```

majority_vote 함수 에서는 사전에 계산된 k개의 최근접 데이터를 바탕으로 각 데이터가 어떤 class에 속하는지 판별 후 1표씩 행사합니다. 그 후 가장 많이 투표된 데이터의 클래스를 반환합니다.

weighted_majority_vote 함수

```
# 가중치 계산된 majority vote 를 계산하는 함수.
    # 가중치는 거리의 역수의 로그값이며, 해당하는 class 에 계산된 가중치를 곱한것을 더하는 식으로 분류를 하였음.
    def weighted_majority_vote(self):
        majority_cnt = []

        # class 개수의 길이를 가지는 리스트를 만든다.
        for i in range(len(self.iris_names)):
            majority_cnt.append(0)

        # nearest k의 이름에 해당하는 class 의 카운트를 증가시킨다.
        for i in range(self.k):
            # 가중치는 거리의 역수로 취하고, 가중치가 너무 커지는 것을 방지하기 위해 거리의 역수의 로그값을 가중치로 한다.
            weight = math.log10(1 / self.nearest_dist[i])
            majority_cnt[self.target[self.nearest_k[i]]] += weight

        # 가장 높은 수치를 가진 꽃 class 가 속한 이름을 찾아 저장한다.
        self.weighted_majority_vote_value =
self.iris_names[majority_cnt.index(max(majority_cnt))]
        return self.weighted_majority_vote_value
```

weighted_majority_vote 함수 에서는 각 데이터가 1표씩 가지지 않고, weight 값을 가집니다. weight값은 계산된 실제 거리의 역수의 로그값입니다. 로그를 지정한 이유는, 실제 거리의 역수 값들을 0에서 1 사이로 정규화 시키는 데에 그 의의가 있는데, 거리가 매우

가까울 경우 weight값이 너무 커질 수 있기에, 그 데이터 하나에 classification이 의존하게 되는 것을 방지하기 위함입니다.

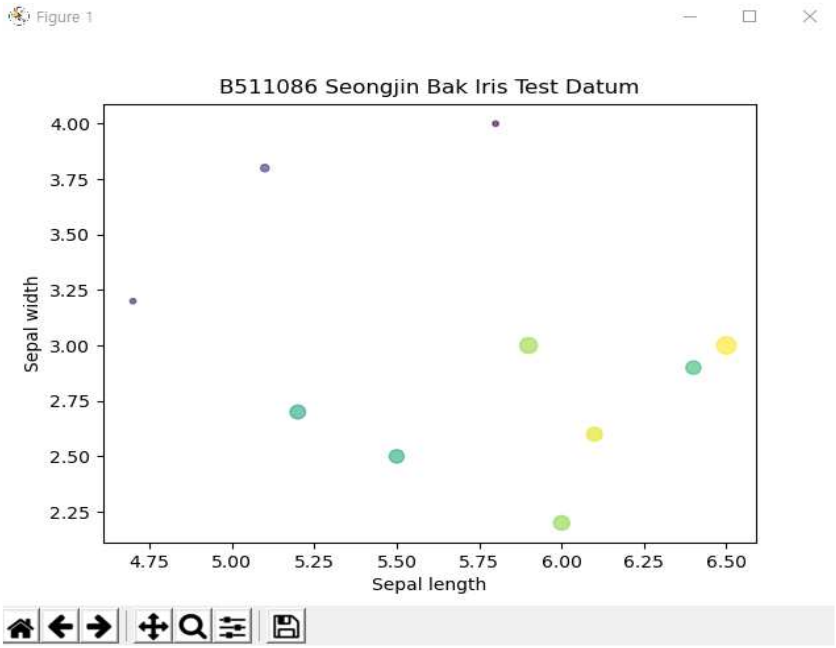
6. 결과 및 분석

6.1 데이터 도표

* 데이터들은 x 축에 sepal_length, y 축에 sepal_width 를 지정하였으며, 점 의 크기는 petal_width 의 스칼라 값 만큼 크게, 색은 petal_length 스칼라 값의 색상을 지정하였습니다.

2.1.1 트레이닝 데이터 산포도

2.1.2 테스트 데이터 산포도



6.2 테스트 결과

6.2.1 Majority Vote 방식으로 분류

6.2.1.1 K = 3 인 경우(정확도 : 90%)

```
Majority Vote,  k :  3  Test started..
Test Data:  0  Computed class:  setosa ,      True class:  setosa
Test Data:  1  Computed class:  setosa ,      True class:  setosa
Test Data:  2  Computed class:  setosa ,      True class:  setosa
Test Data:  3  Computed class:  versicolor ,  True class:  versicolor
Test Data:  4  Computed class:  versicolor ,  True class:  versicolor
Test Data:  5  Computed class:  versicolor ,  True class:  versicolor
Test Data:  6  Computed class:  virginica ,   True class:  virginica
Test Data:  7  Computed class:  versicolor ,  True class:  virginica
Test Data:  8  Computed class:  virginica ,   True class:  virginica
Test Data:  9  Computed class:  virginica ,   True class:  virginica
Majority Vote,  k :  3  Test finised..
```

6.2.1.2 K = 5 인 경우(정확도 : 90%)

```
Majority Vote,  k :  5  Test started..
Test Data:  0  Computed class:  setosa ,      True class:  setosa
Test Data:  1  Computed class:  setosa ,      True class:  setosa
Test Data:  2  Computed class:  setosa ,      True class:  setosa
```

```
Test Data:  3  Computed class:  versicolor ,   True class:  versicolor
Test Data:  4  Computed class:  versicolor ,   True class:  versicolor
Test Data:  5  Computed class:  versicolor ,   True class:  versicolor
Test Data:  6  Computed class:  virginica ,    True class:  virginica
Test Data:  7  Computed class:  versicolor ,   True class:  virginica
Test Data:  8  Computed class:  virginica ,    True class:  virginica
Test Data:  9  Computed class:  virginica ,    True class:  virginica
Majority Vote,  k :  5  Test finised..
```

6.2.1.3 K = 10 인 경우(정확도 : 100%)

```
Majority Vote,  k :  10  Test started..
Test Data:  0  Computed class:  setosa ,       True class:  setosa
Test Data:  1  Computed class:  setosa ,       True class:  setosa
Test Data:  2  Computed class:  setosa ,       True class:  setosa
Test Data:  3  Computed class:  versicolor ,   True class:  versicolor
Test Data:  4  Computed class:  versicolor ,   True class:  versicolor
Test Data:  5  Computed class:  versicolor ,   True class:  versicolor
Test Data:  6  Computed class:  virginica ,    True class:  virginica
Test Data:  7  Computed class:  virginica ,    True class:  virginica
Test Data:  8  Computed class:  virginica ,    True class:  virginica
Test Data:  9  Computed class:  virginica ,    True class:  virginica
Majority Vote,  k :  10  Test finised..
```

6.2.2 Weighted Majority Vote 방식으로 분류

6.2.2.1 K = 3 인 경우(정확도 : 90%)

```
Weighted Majority Vote,  k :  3  Test started..
Test Data:  0  Computed class:  setosa ,       True class:  setosa
Test Data:  1  Computed class:  setosa ,       True class:  setosa
Test Data:  2  Computed class:  setosa ,       True class:  setosa
Test Data:  3  Computed class:  versicolor ,   True class:  versicolor
Test Data:  4  Computed class:  versicolor ,   True class:  versicolor
Test Data:  5  Computed class:  versicolor ,   True class:  versicolor
Test Data:  6  Computed class:  virginica ,    True class:  virginica
Test Data:  7  Computed class:  versicolor ,   True class:  virginica
Test Data:  8  Computed class:  virginica ,    True class:  virginica
Test Data:  9  Computed class:  virginica ,    True class:  virginica
Weighted Majority Vote,  k :  3  Test finished..
```

6.2.2.2 K = 5 인 경우(정확도 : 90%)

```
Weighted Majority Vote,  k :  5  Test started..
Test Data:  0  Computed class:  setosa ,       True class:  setosa
Test Data:  1  Computed class:  setosa ,       True class:  setosa
Test Data:  2  Computed class:  setosa ,       True class:  setosa
Test Data:  3  Computed class:  versicolor ,   True class:  versicolor
Test Data:  4  Computed class:  versicolor ,   True class:  versicolor
Test Data:  5  Computed class:  versicolor ,   True class:  versicolor
Test Data:  6  Computed class:  virginica ,    True class:  virginica
Test Data:  7  Computed class:  versicolor ,   True class:  virginica
Test Data:  8  Computed class:  virginica ,    True class:  virginica
Test Data:  9  Computed class:  virginica ,    True class:  virginica
Weighted Majority Vote,  k :  5  Test finished..
```

6.2.2.3 K = 10 인 경우(정확도 : 100%)

```
Weighted Majority Vote,  k :  10  Test started..
Test Data:  0  Computed class:  setosa ,       True class:  setosa
Test Data:  1  Computed class:  setosa ,       True class:  setosa
Test Data:  2  Computed class:  setosa ,       True class:  setosa
Test Data:  3  Computed class:  versicolor ,   True class:  versicolor
Test Data:  4  Computed class:  versicolor ,   True class:  versicolor
Test Data:  5  Computed class:  versicolor ,   True class:  versicolor
Test Data:  6  Computed class:  virginica ,    True class:  virginica
Test Data:  7  Computed class:  virginica ,    True class:  virginica
Test Data:  8  Computed class:  virginica ,    True class:  virginica
Test Data:  9  Computed class:  virginica ,    True class:  virginica
Weighted Majority Vote,  k :  10  Test finished..
```

위 테스트 결과들로 미루어 보아, k의 값은 10 인 경우가 정확도가 가장 높았습니다. 적절한 k 값은 10으로 보이며, 이 데이터에서는 weighted majority vote와 majority vote 방식에서 결과적인 측면에서는 차이가 없었습니다. weighted majority vote와 majority vote 방식의 차이점은, 가까운 훈련 데이터 일수록 분류에 큰 영향을 끼친다는 것인데, 영향을 끼치는 정도는 weight 값에 따라 좌우됩니다. 그러므로, weight 값을 적절하게 주어야 하지만, 위 테스트에서는 그저 실제 거리의 역수의 로그값을 채택했습니다. 거리 계산 시 feature 중 중요한 feature에게는 높은 가중치를 주는 방식으로 기존의 유클리디안 거리를 개선한다면, 알고리즘이 더 높은 정확도를 보이도록 개선할 수 있을 것으로 예상됩니다.