**MLP**

15종류의 폐기물들의 feature들을 담은 train.pkl, test.pkl 파일과 그에 따른 label을 담고 있는 Ytrain.pkl, Ytest.pkl 파일을 사용해 train accuracy와 test accuracy를 측정했다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | N = 50 | N = 75 | N = 100 | N = 150 | N = 300 |
| Train accuracy | 0.856 | 0.856 | 0.855 | 0.856 | 0.856 |
| Test accuracy | 0.856 | 0.856 | 0.853 | 0.856 | 0.856 |

1. N (hidden layer size)의 변화에 따른 accuracy는 아래 표와 같다.  
   hidden layer의 크기가 증가함에 따라 몇 초에서 몇 십 초까지 실행시간이 증가했지만, 정확도에는 유의미한 변화가 보이지 않았으며, 때문에 이후 다른 parameter들의 영향을 측정할 때, hidden layer size는 75로 고정하였다.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Identity | Logistic | relu |
| Train accuracy | 0.833 | 0.856 | 0.856 |
| Test accuracy | 0.829 | 0.856 | 0.857 |

1. Activation function을 relu, identity, logistic 3가지 값으로 설정하고 모델을 테스트한 결과는 아래 표와 같다.  
     
     
     
     
   Identity에서만 유난히 accuracy가 떨어지는 것을 확인할 수 있다.  
   training/test 데이터셋으로 폐기물에 대한 feature 4가지가 0부터 1사이의 값으로 제공되는데, Identity는 나머지 두 activation function과 다르게 f(x) = x를 리턴하기 때문에, 각 feature들에 대한 값의 차이가 상대적으로 적게 반영되었을 것으로 추측된다.

**SVM**MLP와 마찬가지로 15 종류의 폐기물 정보가 담긴 dataset, label을 활용하여 모델을 학습시키고, 여러 parameter들의 값을 바꿔가며 테스트했다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Linear | Poly | Rbf | sigmoid |
| Train accuracy | 0.835 | 0.856 | 0.856 | 0.785 |
| Test accuracy | 0.852 | 0.856 | 0.856 | 0.783 |

1. Kernel function으로 linear, poly, rbf, sigmoid 4가지 값을 부여하며 테스트했다.  
   feature의 개수가 5가지이고, 각 폐기물 class의 feature들이 linear하게 존재할 가능성이 적다고 생각했기 때문에 linear가 적합하지 않을 것이라고 예상했으며, 결과는 아래 표와 같다.  
   linear함수가 poly, rbf 함수보다 약간 accuracy가 떨어지긴 했지만, 예상외로 sigmoid함수를 적용했을 때 accuracy가 크게 떨어지는 것을 볼 수 있었다.
2. 실수형 C parameter를 부여할 수 있는데, 이 값이 커질수록 training 과정에서 오류를 허용하지 않는 마진을 구하게 되고(hard margin), 이 값이 작을수록 outlier들을 허용하는 마진(soft margin)을 구하게 된다.  
   C parameter의 값을 바꿔가며 테스트하다 보면, 일정 수준 이상의 값에서 train accuracy는 높아지고, test accuracy는 비교적 낮아지는 현상(overfitting)이 발생할 것이라고 예상했다. 결과는 아래 표와 같다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | C = 0.005 | C = 0.01 | \*C = 1 | C = 10 | C = 100 | C = 1000 |
| Train accuracy | 0.812 | 0.825 | 0.856 | 0.85617 | 0.85628 | 0.85657 |
| Test accuracy | 0.808 | 0.823 | 0.856 | 0.85598 | 0.85684 | 0.85684 |

C값이 작아질수록 accuracy가 감소하는 모습을 보였지만, C값이 증가할 때는 예상했던 overfitting 현상이 나타나지 않았다.  
train accuracy와 test accuracy 둘 다 증가하는 추세를 보이고, C=1000 이상부터는 두 accuracy가 수렴하여 더 이상 변화를 보이지 않았다.

**Comparison**

MLP와 SVM 사이에서 accuracy에 대해 큰 차이가 보이진 않았으며, 어떠한 방식이 더 좋다고 말할 수는 없을 것 같다.  
모델이 예측한 값이 label과 다를 때를 출력해봤을 때, Label 2를 1로, Label 1을 2로 잘못 예상하는 경우가 다른 경우들에 비해 압도적으로 많았다. 이는 MLP, SVM 두 모델에서 동일하게 나타났다.  
dataset의 feature들에서 대략적인 이유를 찾을 수 있었는데, 먼저 Label 1과 2는 각각 가구류/의자, 고철류/고철이다. 이 2개의 사물이 5개의 feature에 대해 가지는 값을 예상해보면 겹치는 부분이 상당 수 생길 수 있다. (훼손 정도, 불투명, 딱딱함 등)  
다른 class임에도 여러 feature들에서 비슷한 값을 가지게 되고, 그에 따라 많은 misprediction이 발생했다고 추측된다.

추가로, dataset이 줄어들었을 때의 성능을 보기 위해 label 14, 15번에 대해서만 training시키고 test accuracy를 측정해봤다. 1, 2번 Label같은 경우 겹치는 feature가 많아 정확한 classification이 힘들다는 것을 앞에서 살펴봤기 때문에, 14번 15번 Label을 선택했다.  
테스트해 본 결과, MLP와 SVM 둘 다 98.9%의 정확도를 보였다. 5가지의 feature가 작은 수의 class를 판별하기에는 충분하지만, 15개의 class를 판별하기에는 다소 부족할 수 있다고 생각한다. 앞에서 본 1, 2번 Label 같은 경우도 재질, 무게 등 다른 feature를 이용할 수 있다면 misprediction을 꽤 줄일 수 있었을 것이다.

**Environment**

* Python 3.9.0
* scikit-learn 1.0.2