**Machine Learning Assignment**

조 병 웅

2019312570

원전공 사학과/복수전공 인공지능 융합

테이블이나 그림 추가 가능합니다.

각 섹션은 절대 변경하지 마세요. 단, 추가적인 섹션이 필요하다면 추가하셔도 됩니다.

1. **Problem Definition**

해결하고자 하는 문제는, 네트워크 트래픽 정보를 담고 있는 4가지 타입의 데이터 셋들이 주어졌을 때, 이들을 전처리하여 어떤 데이터가 어떤 타입에 속하는 지를 예측하는 모델을 만드는 것이다. 4가지 타입에는 interactive, bulk, web, video가 있으며, 구현할 수 있는 모델의 종류는, Naive bayes, SVM, Logistic Regression 세가지가 있다. 필자는 이번 문제에서 해결을 수행할 모델로 다중 분류 로지스틱 회귀 모델을 채택하였다.

1. **Data**
   1. **Data Description**

input data :

해당 데이터들은 train 디렉터리 밑에 각각 데이터 형식에 따라 수십개의 csv로 분할되어 있다. 각 파일 데이터의 형식은 먼저 첫번째 파일 데이터가 어떤 데이터인지를 알려주는 이름이 나와있다. 그리고 시간을 나타내는 time,. 프로토콜을 나타내는 proto, 데이터의 크기를 나타내는, data\_len,, 그리고 데이터의 출발 아이피와 출발포트, 도착 아이피와 도착포드를 뜻하는 ip\_src, ip\_dst, src\_port, dst\_port 등 총 7개의 클래스로 구성되어있다. time 클래스는 부동소수점 방식으로 변환되어있으며, ip들은 IPv4 형식으로 나타나져 있다. 대체로 각 데이터 파일들은 큰 차이를 보이지 않지만, data\_len 클래스가 시간과 IP에 따라 들어오는 형태가 각 데이터 별로 유의미한 분석을 이끌어 낼 수 있어 해당하는 특징을 통해 분석을 시도하리라 예측이 가능하다.

전처리 후 data :

전처리후 데이터들은 총 11가지의 클래스를 가진다. 해당 클래스들은 11개의 크기로 windowing된 data\_len의 개수들인데, 즉, 기준부터 t초까지 데이터를 처리하도록 만들었다. 또한 해당 클래스들을 그대로 정수 형태로 사용하는 것이 아닌, 큰 데이터가 모델링에 영향을 최소하기 위해 standardscaler를 사용해 표준화되어있는 형태를 띈다.

또한, 해당하는 데이터들은 한가지 파일에서 전처리되고 concat 되기 때문에, 전처리된 파일들을 합치기전 해당 파일들의 라벨링을 할 수 있다. 때문에 concat을 처리하기전 해당 데이터 프레임에 원핫인코딩된 4개 클래스의 라벨링된 정답들을 붙여 나중에 y데이터로 사용하기 위해 열을 붙인다. 해당하는 클래스는 각각 4가지의 데이터 타입들을 나타내고, 해당 데이터 타입은 1, 나머지는 0을 표시하는 원핫 인코딩방식으로 표현된다.

test data :

test data 형태는 전처리된 형태와 labeling이 안된 것 빼고는 동일하다. 하지만, train 데이터에도 standardscaler를 처리하였기 때문에 test data에도 동일하게 처리를 해준다.

output data :

본래 정답을 4개의 열(클래스)로 원핫인코딩해놓았기 때문에, 모델링 과정에는 y햇 데이터와 y데이터는 4개의 요소를 가진 정답들의 합으로 구성된다. 하지만 우리가 submission해야하는 csv파일은 인덱스와 0,1,2,3 정답만을 가진 파일이기 때문에, prediction된 정답을 가지고 다시 전처리를 통해 0,1,2,3 라벨링된 한가지 요소만을 가진 정답 파일로 바꾼다.

* 1. **Data Pre-processing**

1. 파일 불러오기

oslistdir()함수를 통해 파일 리스트를 불러오고, 각 파일을 read\_csv()함수를 통해 불러온다.

2. 시간 변화

str화된 time클래스의 값을 time클래스의 strftime()함수를 통해 형식을 바꾼다.

3. 작성한 함수(count\_ip\_dst)를 통해 ip\_dst값을 측정한다.

groubby로 묶인 데이터 프레임을 순회하면서, 동일한 시간 동일한 ip들을 딕셔너리 형식에 키로 찾는다. 그리고 해당하는 키가 찾아질때마다 값을 증가시켜 증가된 값과 40을 비교하여 제거해야할 데이터들을 파악한다.

4. 작성한 함수(discard\_ip\_dst)를 통해 ip\_dst값들이 40이하인 값을 삭제한다.

딕셔너리를 통해 값을 파악하고, 만약 40이하라면 drop()메서드를 사용해 값들을 제거한다. 이때 복사에 주의한다

5. 작성한 함수(counting\_data\_len)를 통해 data\_len 개수를 구한다.

grouipby를 통해 count하고, 카운트된 값을 시리즈 형식 temp변수에 저장한다. 이때, temp를 순회하면서 11개의 데이터 들을 추출하여 to\_csv\_df 데이터프레임에 각 행으로 저장한다.

다 완료된 데이터프레임에, 원핫 인코딩된 결과를 저장하기 위해 y햇 데이터 클래스 4개를 만들고 거기에 파일의 데이터 타입에 맞게 저장한다.

1. **Analysis**

모델에 활용될 피처는 data\_len의 개수들을 windowing을 통해 11개의 클래스를 주어진 형태이다. 본래 data\_len개수를 모아논 변수에서 data\_len개수의 값들은 특정 시간대 별로 나열이 되어있었는데, 이를 windowing 하여 특정 데이터 타입이 t초 안에서 시간대의 변화별로 나열된 데이터들이 피처로 활용되었다고 이해하였다.

1. **Model**
   1. **Classifier**

Multi classes Logistiv Regression을 사용하였다. 해당 알고리즘은 기존에 4번째 제출했던 로지스틱 회귀 과제와 비슷하지만, 멀티 클래스 분류라는 것에서 차이점을 가진다. input 데이터와 정답이 주어지고, 이를 weight와 bias와 연산하여 나온 y햇을 정답과 연산하여 오차를 통해 다시금 weights와 bias를 수정한다는 아이디어는 동일하다. 하지만, 멀티클래스이기 때문에 2가지 주요한 차이점을 가진다. 먼저, 각 클래스로 분류하기 위해 각 클래스마다 연산되는 weights가 존재한다. 즉, 이번 과제에서는 0,1,2,3 총 4가지로 분류하기 때문에 weights도, bias도 총 4개가 필요한 것이다. 또한 기존에 sigmoid함수를 통해 이진분류를 진행했던 것과 달리, 멀티클래스에서는 softmax를 구현해야한다. 구해진 결과를 softmax를 통과하여 확률을 구하고, 가장 높은 확률이 측정된 클래스를 정답 클래스로 도출하는 것이다.

* 1. **Design Consideration**

기존의 큰 틀은 과제속에서 구현했던 로지스틱 회귀를 유지하면서도, 멀티클래스를 위해 바뀌어어야할 고려사항은 다음과 같다. 먼저 weights를 초기화하는 수행에서 weights들을 클래스 개수에 맞게 각각 구해놓는다. fwpass 부분에서 train\_x와 연산되는 weight들을 각각 분류해야할 클래스마다 연산을 수행한다. 즉, z1, z2, z3, z4 연산을 따로 수행한 뒤 이것들을 행렬화하여 저장한다. (나중에 연산이 용이하도록) 그리고 해당 행렬을 soft max함수를 통과하여 각 확률값들을 구해놓는다. 이후 이를 원핫인코딩된 y 데이터에 뺄셈 연산을 수행하여 err값을 만든다. bwpass에서는 w 그래디언트를 구하는 과정들을 앞에서 fwpass에서 수행한것과 동일하게 각 클래스 분류 개수에 맞게 구한다. 그후 나온 행렬을 가지고 prediction을 수행한다.

1. **Experiments**
   1. **Settings**

Train test 비율은 디폴드 값인 0.25로 넣었다. 나머지 데이터들은 sklearn 모듈을 모델링 부분에서 사용할 수 없었기 때문에, sklearn의 파라미터 수행을 볼 수가 없어서 test데이터가 변화하는 부분을 관하면서 값을 결정하였다. lr와 epoch는 에포크를 줄인다면 러닝 레이트 비율을 높여보는 방식으로 측정을 하였고, 때문에 가장 정확도가 높게 나왔던 lr : 0.001과 500 = max\_iter로 구하였다. lambda 값은 0.01로 결정되었다. 정규화는 2가지 중 가장 높게나온 L2 정규화로 구하였다. 학습 프로세스는 미니배치 등 다양한 프로세스를 고려하였으나, 아쉽게도 시간 상의 이유로 구현하지 못했다.

* 1. **Performance Metrics**

평가 지표는 기존의 로지스틱 회귀 결과값 평가 방식과 동일하게 prediction된 데이터와 정답 데이터를 ==연산하고, True값의 mean()을 측정하는 방식으로 구하였다.

* 1. **Results**

정답 데이터가 원핫 인코딩 되어있기 때문에, 0,1,0,0이런 식으로 구해진다. 표현상의 문제로 행렬 전체의 평균이 나오지 않고 각 열의 평균들이 구해졌는데, 첫번째 열인 interactive의 결과 수행 값이 0.7 정도이고, 나머지 수행 값들이 0.9 정도로 결과가 나왔다. 이렇게 결과가 나온 이유는 데이터 결과의 비중이 비교적 0으로 측정된 결과가 높은 편이고, 때문에 interactive 데이터 타입을 잘 파악했냐 안 파악했냐가 수행의 측정도를 높이는 요소였던 것 같다. 물론 w의 비중이 interactive 타입 쪽으로 편향했을 가능성도 무시할 수는 없다.

1. **Discussion and Limitation**

세부적인 디테일, 배치 사이즈 등과 같은 부분을 시간의 부족으로 구현 못한 부분이 있어 최종적인 결과물이 0.75정도로 비교적 높지 않았던 것 같다. 때문에 보다 정확도 있는 모델을 구현하지 못해 아쉬었다. 그리고 iter 사이즈가 비교적 작은 500이기 때문에, overfitting의 가능성도 배제 할 수 없어 더 아쉬움이 남았다. 해당 클래스 분류의 한계일지도 모르는 문제이긴 하지만, 비교적 구현이 쉽고 빠르기 때문에 멀티클래스 로지스틱 회귀를 통한 트래픽 데이터 분류는 완성도만 높일 수 있다면 여러가지로 활용방안이 높을 것으로 기대된다.

1. **References**

-캐글의 수행속도가 너무 늦어 코랩에서 작업하였다. 코랩에서 작업할때 api 코드들을 해당 사이트에서 얻어왔다.

“테디노트”, github.io, 2019년 12월 7일 수정, 2023년 4월 28일 접속, <https://teddylee777.github.io/kaggle/Kaggle-API-%EC%82%AC%EC%9A%A9%EB%B2%95/>.

다중 로지스틱 회귀 레퍼런스

‘perconsi’, tistory.blog, 2022년 5월 7일 작성, 2023년 4월 28ㅇ리 접속, https://perconsi.tistory.com/83.