Chapter 7 Part 5 요약 (Page 318-327) 100620 강인승

Using a Neural Network

- 뉴럴넷을 이용하여 tabula 데이터를 학습시킬 수 있음
  - 데이터를 불러오고, 카테코리 타입으로 반영하여, 데이터 항목을 배열한다
  - Dep variable 를 log 를 취한다 (loss 를 mean squared log loss 하기 위해)
  - 날짜에 관한 내용을 expand 하여 col 에 추가한다

```
df_nn = pd.read_csv(path/'TrainAndValid.csv', low_memory=False)
df_nn['ProductSize'] = df_nn['ProductSize'].astype('category')
df_nn['ProductSize'].cat.set_categories(sizes, ordered=True, inplace=True)
df_nn[dep_var] = np.log(df_nn[dep_var])
df_nn = add_datepart(df_nn, 'saledate')
```

- 뉴럴넷에서는 카테고리컬 데이터를 임베딩을 통해 해결함
  - 임베딩을 할 때 전체 col 안의 카테고리의 개수를 주의할 것 (Max_card argument 사용)
  - Max card 값보다 높은 개수의 카테고리가 있으면 continuous variable 로 취급

```
cont_nn,cat_nn = cont_cat_split(df_nn_final, max_card=9000, dep_var=dep_var)
```

  - 결과값은 range 밖의 값도 예측이 되어야하기에 (extrapolation) cont variable 로 지정

```
cont_nn.append('saleElapsed')
cat_nn.remove('saleElapsed')
```

  - Categorical 항목들을 봤을 때 5000 개의 항목이 2 개가 있음 (model 과 관련)
  - 이런 경우는 redundancy 가 있을 수 있음 (연산처리에 효율적이지 않을 수 있음)

```
df_nn_final[cat_nn].nunique()
```

```
YearMade                73
ProductSize              6
Coupler_System           2
fiProductClassDesc      74
ModelID               5281
Hydraulics_Flow          3
fiSecondaryDesc        177
fiModelDesc           5059
ProductGroup             6
Enclosure                6
fiModelDescriptor      140
Drive_System             4
Hydraulics              12
Tire_Size               17
dtype: int64
```

  - 5000 개 category 를 가지는 하나 항목을 지웠을 때의 효과를 체크하고 문제 없으면 하나 category 를 지운다

- Random forest 와 (데이터의 순서만 의미가 있음) 다르게 뉴럴넷은 데이터의 normalization 이 필요
  - Tabular 데이터는 GPU RAM 을 많이 사용 안하기에 batch size 가 커도 상관없음
  - Regression 모델에서는 예측값의 range 를 잡아주는것이 좋음

```
procs_nn = [Categorify, FillMissing, Normalize]
to_nn = TabularPandas(df_nn_final, procs_nn, cat_nn, cont_nn,
                      splits=splits, y_names=dep_var)
```

```
dls = to_nn.dataloaders(1024)
```

```
y = to_nn.train.y
y.min(),y.max()
```

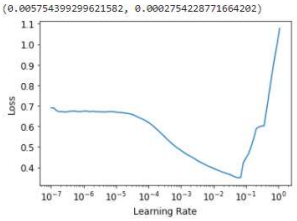
```
(8.465899897028686, 11.863582336583399)
```

- 일반적인 방법으로 뉴럴넷을 학습시킴 (tabular learner)
  - 기본적으로 모델을 불러오면 2 레이어의 200, 100 activation (데이터셋 크기가 크기때문에 500, 250 을 사용)

```
learn = tabular_learner(dls, y_range=(8,12), layers=[500,250],
                        n_out=1, loss_func=F.mse_loss)
```

```
learn.lr_find()
```

```
(0.005754399299621582, 0.0002754228771664202)
```



```
1 learn.fit_one_cycle(5, 1e-2)
```

epoch	train_loss	valid_loss	time
0	0.069705	0.062389	00:11
1	0.056253	0.058489	00:11
2	0.048385	0.052256	00:11
3	0.043400	0.050743	00:11
4	0.040358	0.050986	00:11

```
1 preds,targs = learn.get_preds()
2 r_mse(preds,targs)
```

```
0.2258
```

Ensembling

- Random forest 의 결과가 좋은 이유는 각각의 tree 의 결과값이 다른 tree 와 상관되지 않기 때문
    - Tree 개수가 충분히 있으면 모든 나무의 결과값의 error 의 평균값이 0 에 가까워질 수 있음
- 현재까지 학습한 random forest 와 뉴럴넷의 결과 예측값을 평균낼 수도 있음

```
1 rf_preds = m.predict(valid_xs_time)
2 ens_preds = (to_np(preds.squeeze()) + rf_preds) /2
```

```
1 r_mse(ens_preds,valid_y)
```

```
0.22291
```

Boosting

- Random forest 같이 많은 나무를 엮어서 평균을 내는건 bagging
- Boosting 의 방법
    - 작은 사이즈의 모델이 underfit 되게 데이터를 학습한다
    - 학습된 모델로 결과값을 예측한다
    - 결과값과 예측값의 차를 구한다 (residual)
    - 첫번째 단계로 돌아가 새로운 작은 사이즈의 모델이 앞의 residual 을 학습하게 한다
    - 위의 단계를 계속 순차적으로 반복하면 stopping criterion 에 도달할 때까지 모델의 개수를 늘린다
- Prediction 을 하기 위해 각 tree 에서 예측된 값들을 합한다.
    - Gradient boosting machines (GBMs), gradient boosted decision trees (GBDTs)
    - 가장 인기있는 모델은 XGBoost
- Random forest 와 달리 (나무가 각각 independent 함) boosting ensemble 기법은 (다음 나무가 전 나무에 dependent) overfit 할 수 있음

Combining Embeddings with Other Methods

- 아래의 table 은 다른 알고리즘을 raw col 로 학습했을 때와 categorical embedding 을 한 데이터를 학습했을 때의 차이를 보여줌

method	MAPE	MAPE (with EE)
KNN	0.290	0.116
random forest	0.158	0.108
gradient boosted trees	0.152	0.115
neural network	0.101	0.093