mlp-for-regression

May 4, 2024

1 MLP for Regression

1.1 Imports

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import torch
from torch import nn, optim
import torch.nn.functional as F
from torch.utils.data import DataLoader, TensorDataset
from sklearn.metrics import r2_score
```

1.2 Load Data

[2]:
```
!cp "/content/drive/MyDrive/colab_projects/MLP for Regression/test.csv" test.csv
!cp "/content/drive/MyDrive/colab_projects/MLP for Regression/train.csv" train.
↪csv
```

[3]:
```
df_train = pd.read_csv('/content/train.csv')
df_train.head()
```

[3]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	\
0	1.4817	6	4.443645	1.134293	1397	3.350120	36.77	
1	6.9133	8	5.976471	1.026471	862	2.535294	33.68	
2	1.5536	25	4.088785	1.000000	931	4.350467	36.60	
3	1.5284	31	2.740088	1.008811	597	2.629956	34.10	
4	4.0815	21	5.166667	1.002688	1130	3.037634	37.79	

	Longitude	MedHouseVal
0	-119.84	0.720
1	-117.80	2.741
2	-120.19	0.583
3	-118.32	2.000
4	-121.23	1.179

[4]:
```
x_train = df_train.drop('MedHouseVal', axis=1)
y_train = df_train['MedHouseVal']
```

[5]:
```
df_test = pd.read_csv('/content/test.csv')
df_test.head()
```

[5]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	MedHouseVal
0	6.4114	15	7.527559	1.049869	2529	3.318898	33.60	-117.65	2.787
1	4.1843	12	6.330084	1.041783	2033	2.831476	38.62	-120.91	2.076
2	1.7411	35	5.369159	1.294393	909	4.247664	33.93	-118.23	0.967
3	3.4412	39	4.173405	1.074573	2156	1.937107	37.59	-122.37	3.538
4	7.8195	16	8.602349	1.058725	2142	3.593960	33.70	-117.98	3.905

[6]:
```
x_test = df_test.drop('MedHouseVal', axis=1)
y_test = df_test['MedHouseVal']
```

1.3 Preprocessing

1.3.1 Getting info about df

[7]:
```
df_train.shape, df_test.shape
```

[7]:
```
((16512, 9), (4128, 9))
```

[8]:
```
df_train.info
```

[8]:
```
<bound method DataFrame.info of        MedInc  HouseAge  AveRooms  AveBedrms  Population  AveOccup  Latitude  \
0      1.4817         6  4.443645   1.134293        1397  3.350120     36.77
1      6.9133         8  5.976471   1.026471         862  2.535294     33.68
2      1.5536        25  4.088785   1.000000         931  4.350467     36.60
3      1.5284        31  2.740088   1.008811         597  2.629956     34.10
4      4.0815        21  5.166667   1.002688        1130  3.037634     37.79
…         …       …         …          …           …         …         …
16507  3.0625        20  5.860000   1.112000         745  2.980000     39.59
16508  2.6133        49  5.163755   1.100437        1131  2.469432     38.11
16509  4.4958        19  5.899767   1.074592        1206  2.811189     38.92
16510  2.5750        39  3.591203   1.086675        2546  3.293661     34.16
16511  2.2478        31  5.123810   1.100000        1259  2.997619     40.80

       Longitude  MedHouseVal
0        -119.84        0.720
1        -117.80        2.741
```

```
2        -120.19        0.583
3        -118.32        2.000
4        -121.23        1.179
…          …          …
16507    -121.90        0.938
16508    -122.25        1.031
16509    -121.22        1.926
16510    -118.14        1.535
16511    -124.13        0.811

[16512 rows x 9 columns]>
```

1.4 Convert to Tensor

[9]:
```
x_train = torch.FloatTensor(x_train.values)
y_train = torch.FloatTensor(y_train.values)
```

[10]:
```
x_test = torch.FloatTensor(x_test.values)
y_test = torch.FloatTensor(y_test.values)
```

1.5 Standardization

[11]:
```
mu = x_train.mean(axis=0)
std = x_train.std(axis=0)
x_train = (x_train - mu) / std
```

[12]:
```
x_test = (x_test - mu) / std
```

1.6 Dataloader

[13]:
```
train_dataset = TensorDataset(x_train, y_train)
train_loader = DataLoader(train_dataset, batch_size = 128, shuffle = True)
```

[14]:
```
for x_batch, y_batch in train_loader:
  print(x_batch.shape, y_batch.shape)
  break
```

```
torch.Size([128, 8]) torch.Size([128])
```

[15]:
```
test_dataset = TensorDataset(x_test, y_test)
test_loader = DataLoader(test_dataset, batch_size = 258, shuffle = False)
```

1.7 Model

[16]:
```
num_feats = 8
out_feat = 1
h1 = 64
h2 = 32

model = nn.Sequential(nn.Linear(num_feats, h1),
                      nn.ReLU(),
                      nn.Linear(h1, h2),
                      nn.ReLU(),
                      nn.Linear(h2, out_feat))
```

1.8 Loss and Optimizer

[17]:
```
loss_fn = nn.MSELoss()
optimizer = optim.SGD(model.parameters(), lr=0.001)
```

1.9 Model Training

[18]:
```
num_epochs = 400

for epoch in range(num_epochs):
  loss_train = 0
  for x_batch, y_batch in train_loader:
    yp = model(x_batch)
    loss = loss_fn(yp.squeeze(), y_batch)
    loss_train += loss
    loss.backward()
    optimizer.step()
    optimizer.zero_grad()

  loss_test = 0
  for x_batch, y_batch in test_loader:
    yp = model(x_batch)
    loss_test += loss_fn(yp.squeeze(), y_batch)

  if epoch % 10 == 0:
    loss_total_train = loss_train/len(train_loader)
    loss_total_test = loss_test/len(test_loader)
    print(f'Epoch = {epoch} : Loss Train = {loss_total_train:.4}')
    print(f'Epoch = {epoch} : Loss Test = {loss_total_test:.4}')
    print()
```

```
Epoch = 0 : Loss Train = 3.829
Epoch = 0 : Loss Test = 2.482
```

```
Epoch = 10 : Loss Train = 0.6239
Epoch = 10 : Loss Test = 0.6271

Epoch = 20 : Loss Train = 0.5574
Epoch = 20 : Loss Test = 0.5661

Epoch = 30 : Loss Train = 0.5113
Epoch = 30 : Loss Test = 0.5209

Epoch = 40 : Loss Train = 0.4796
Epoch = 40 : Loss Test = 0.4892

Epoch = 50 : Loss Train = 0.4586
Epoch = 50 : Loss Test = 0.4686

Epoch = 60 : Loss Train = 0.4437
Epoch = 60 : Loss Test = 0.4538

Epoch = 70 : Loss Train = 0.4321
Epoch = 70 : Loss Test = 0.4421

Epoch = 80 : Loss Train = 0.4226
Epoch = 80 : Loss Test = 0.4324

Epoch = 90 : Loss Train = 0.4144
Epoch = 90 : Loss Test = 0.4243

Epoch = 100 : Loss Train = 0.4073
Epoch = 100 : Loss Test = 0.4174

Epoch = 110 : Loss Train = 0.4009
Epoch = 110 : Loss Test = 0.4108

Epoch = 120 : Loss Train = 0.395
Epoch = 120 : Loss Test = 0.4051

Epoch = 130 : Loss Train = 0.3896
Epoch = 130 : Loss Test = 0.3996

Epoch = 140 : Loss Train = 0.3847
Epoch = 140 : Loss Test = 0.3944

Epoch = 150 : Loss Train = 0.3801
Epoch = 150 : Loss Test = 0.3899

Epoch = 160 : Loss Train = 0.3759
Epoch = 160 : Loss Test = 0.3857
```

```
Epoch = 170 : Loss Train = 0.3719
Epoch = 170 : Loss Test = 0.3818

Epoch = 180 : Loss Train = 0.3685
Epoch = 180 : Loss Test = 0.3784

Epoch = 190 : Loss Train = 0.3651
Epoch = 190 : Loss Test = 0.3754

Epoch = 200 : Loss Train = 0.3622
Epoch = 200 : Loss Test = 0.3728

Epoch = 210 : Loss Train = 0.3595
Epoch = 210 : Loss Test = 0.3704

Epoch = 220 : Loss Train = 0.3569
Epoch = 220 : Loss Test = 0.368

Epoch = 230 : Loss Train = 0.3543
Epoch = 230 : Loss Test = 0.3657

Epoch = 240 : Loss Train = 0.352
Epoch = 240 : Loss Test = 0.3638

Epoch = 250 : Loss Train = 0.3499
Epoch = 250 : Loss Test = 0.3614

Epoch = 260 : Loss Train = 0.3477
Epoch = 260 : Loss Test = 0.3597

Epoch = 270 : Loss Train = 0.3457
Epoch = 270 : Loss Test = 0.3577

Epoch = 280 : Loss Train = 0.3438
Epoch = 280 : Loss Test = 0.356

Epoch = 290 : Loss Train = 0.342
Epoch = 290 : Loss Test = 0.3543

Epoch = 300 : Loss Train = 0.3401
Epoch = 300 : Loss Test = 0.3529

Epoch = 310 : Loss Train = 0.3386
Epoch = 310 : Loss Test = 0.3516

Epoch = 320 : Loss Train = 0.337
Epoch = 320 : Loss Test = 0.3501
```

```
Epoch = 330 : Loss Train = 0.3354
Epoch = 330 : Loss Test = 0.3489

Epoch = 340 : Loss Train = 0.3339
Epoch = 340 : Loss Test = 0.3475

Epoch = 350 : Loss Train = 0.3325
Epoch = 350 : Loss Test = 0.3462

Epoch = 360 : Loss Train = 0.3311
Epoch = 360 : Loss Test = 0.345

Epoch = 370 : Loss Train = 0.3298
Epoch = 370 : Loss Test = 0.3438

Epoch = 380 : Loss Train = 0.3284
Epoch = 380 : Loss Test = 0.3429

Epoch = 390 : Loss Train = 0.3271
Epoch = 390 : Loss Test = 0.3413
```

1.10 Evaluation

[20]:
```
yp_total = []
yt_total = []
with torch.no_grad():
  for x, y in test_loader:
    yp = model(x)
    yp_total.append(yp.squeeze())
    yt_total.append(y)
```

[21]:
```
yp_total = torch.cat(yp_total)
yt_total = torch.cat(yt_total)
```

[22]:
```
r2_score(yp_total, yt_total)
```

[22]: 0.6572521143147981