Estimating House Values Using Machine Learning

- Connor Mahon
- Alexander Tran
- Chau Nguyen Ba Khanh

Dataset:

- California housing price on Kaggle:https://www.kaggle.com/datasets/camnugent/california-housing-prices.
- Training set contains 17000 samples and 9 numerical features.
- Testing set has 3000 samples

Dataset

Features:

- Longitude
- Latitude
- Median House Age
- Total Rooms
- Total Bedrooms
- Population
- Number Of Households
- Median Income

Target:

- Median House Value

Pre-processing

- Normalize dataset:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
0	0.523118	-0.152521	-0.266458	0.078248	0.115392	-0.011620	-0.004805	-0.164824	-0.289485
1	0.507182	-0.130205	-0.188027	0.131971	0.211296	-0.008424	-0.006285	-0.142314	-0.262269
2	0.498218	-0.205656	-0.227242	-0.050709	-0.056706	-0.030734	-0.063184	-0.153976	-0.250722
3	0.497222	-0.210970	-0.286066	-0.030122	-0.031411	-0.025633	-0.045259	-0.047715	-0.276083
4	0.497222	-0.218409	-0.168419	-0.031361	-0.033118	-0.022578	-0.039339	-0.135072	-0.292372
...	...	...	...	...	...	...	...	...	...
16995	-0.467918	0.526544	0.459032	-0.011247	-0.022565	-0.014647	-0.021743	-0.105273	-0.197733
16996	-0.468914	0.538233	0.145307	-0.007768	-0.001771	-0.006603	-0.005957	-0.094183	-0.264537
16997	-0.471902	0.660444	-0.227242	0.000879	-0.001305	-0.005201	-0.007437	-0.058777	-0.213815
16998	-0.471902	0.656193	-0.188027	0.000747	0.001954	-0.003688	-0.003819	-0.131300	-0.250516
16999	-0.476882	0.522293	0.459032	-0.021713	-0.037153	-0.017477	-0.038024	-0.059922	-0.232372

17000 rows × 9 columns

Model Design

Simple linear regression model:

- 3 linear layers
- Hidden layer has 64 features
- ReLU activation function
- Sigmoid function

```
class LinearRegression(nn.Module):
    def __init__(self,num_features):
        super().__init__()
        self.layer1 = torch.nn.Linear(num_features,64).to(device)
        self.layer2 = torch.nn.Linear(64,64).to(device)
        self.layer3 = torch.nn.Linear(64,1).to(device)

        self.relu = nn.ReLU().to(device)
        self.sigmoid = nn.Sigmoid().to(device)

    def forward(self,x):
        y_pred = self.layer1(x)
        y_pred = self.relu(y_pred)
        y_pred = self.layer2(y_pred)
        y_pred = self.relu(y_pred)
        y_pred = self.sigmoid(self.layer3(y_pred))
        return y_pred
```

Training Loop

- Loss function: Mean squared error
- Optimizer: Adam
- Epochs = 2500
- Batch size = 128
- Max batches = 500
- Total training time: 5749.4 seconds

Test result

Predicted housing prices with 20% error in value.

mean error accuracy: 0.20079277147555016

	Actual Median House Price	Predicted Median House Price	Difference	Percentage Error
0	184210.0	205846.3	-21636.3	0.117455
1	180917.7	317644.6	-136726.9	0.755741
2	222226.0	229423.1	-7197.1	0.032386
3	198729.5	313148.2	-114418.7	0.575751
4	178146.4	205846.3	-27699.9	0.155490
5	185323.2	205846.3	-20523.1	0.110742
6	207185.4	205846.8	1338.6	0.006461
7	220970.7	318219.9	-97249.2	0.440100
8	211354.1	205866.5	5487.6	0.025964
9	210714.6	205846.3	4868.3	0.023104
10	212349.0	308062.3	-95713.3	0.450736
11	211756.8	317003.9	-105247.1	0.497019
12	230350.3	223459.0	6891.3	0.029917
13	189397.2	205846.3	-16449.1	0.086850
14	236935.0	227945.0	8990.0	0.037943
15	212041.0	269430.9	-57389.9	0.270655
16	238143.0	251201.3	-13058.3	0.054834
17	219431.1	286862.1	-67431.0	0.307299
18	214670.2	318537.1	-103866.9	0.483844
19	210264.6	205846.3	4418.3	0.021013
20	228763.4	239233.6	-10470.2	0.045769

Practical Demonstration

- In a practical application, such as a real estate website, the pre-trained model could be loaded, and the user would enter data on the features of their district.
- The data would then be preprocessed, and given to the model, which would then output a prediction of the median house price of the area.

```
[ ]  data = pd.DataFrame(
         {'longitude' : -120,
         'latitude' : 30,
         'housing_median_age' : 30.0,
         'total_rooms' : 2000.0,
         'total_bedrooms' : 400.0,
         'population' : 1000.0,
         'households' : 400.0,
         'median_income' : 2.0000}, index=[0])

     model = LinearRegression(8)
     model = torch.load('/content/mymodel.pt')

     output = demonstrate_model(model, data)
     print('Predicted median house price: ', output)

     Predicted median house price:  236335.5
```

```
pd.read_csv('sample_data/california_housing_train.csv')

 = pd.DataFrame(
{'longitude' : np.random.uniform(df['longitude'].min(), df['longitude'].max()),
'latitude' : np.random.uniform(df['latitude'].min(), df['latitude'].max()),
'housing_median_age' : np.random.uniform(df['housing_median_age'].min(), df['housing_med
'total_rooms' : np.random.uniform(df['total_rooms'].min(), df['total_rooms'].max()),
'total_bedrooms' : np.random.uniform(df['total_bedrooms'].min(), df['total_bedrooms'].ma
'population' : np.random.uniform(df['population'].min(), df['population'].max()),
'households' : np.random.uniform(df['households'].min(), df['households'].max()),
'median_income' : np.random.uniform(df['median_income'].min(), df['median_income'].max()

l = LinearRegression(8)
l = torch.load('/content/mymodel.pt')

ut = demonstrate_model(model, data)
t('Predicted median house price: ', output)

icted median house price:  322062.9
```

Thanks for listening!