18) Predicting House Prices with Keras

Vitor Kamada

August 2019

Reference

Chollet, F. (2018). **Deep Learning with Python**. Ch 3.

https://github.com/fchollet/deep-learning-with-python-notebooks/blob/master/3.7-predicting-house-prices.ipynb

The Boston Housing Price Dataset

```
import keras
```

```
from keras.datasets import boston_housing
(train_data, train_targets), (test_data,
  test_targets) =  boston_housing.load_data()
```

```
train_data.shape
```

```
(404, 13)
```

```
test_data.shape
```

```
(102, 13)
```

13 Features

1. Per capita crime rate.
2. Proportion of residential land zoned for lots over 25,000 square feet.
3. Proportion of non-retail business acres per town.
4. Charles River dummy variable (= 1 if tract bounds river; 0 otherwise).
5. Nitric oxides concentration (parts per 10 million).
6. Average number of rooms per dwelling.
7. Proportion of owner-occupied units built prior to 1940.
8. Weighted distances to five Boston employment centres.
9. Index of accessibility to radial highways.
10. Full-value property-tax rate per $10,000.
11. Pupil-teacher ratio by town.
12. 1000 * (Bk - 0.63) ** 2 where Bk is the proportion of Black people by town.
13. % lower status of the population.

Normalizing the Data

```
train_targets
```

```
array([ 15.2,  42.3,  50. ,
        14.4,  12.1,  17.9,
        24.1,  27.5,  10.9,
        34.7,  16.6,  17.5,
        13.9,  13.1,  20.4,
```

$$z = \frac{X - \mu}{\sigma}$$

```
mean = train_data.mean(axis=0)
train_data -= mean
std = train_data.std(axis=0)
train_data /= std
```

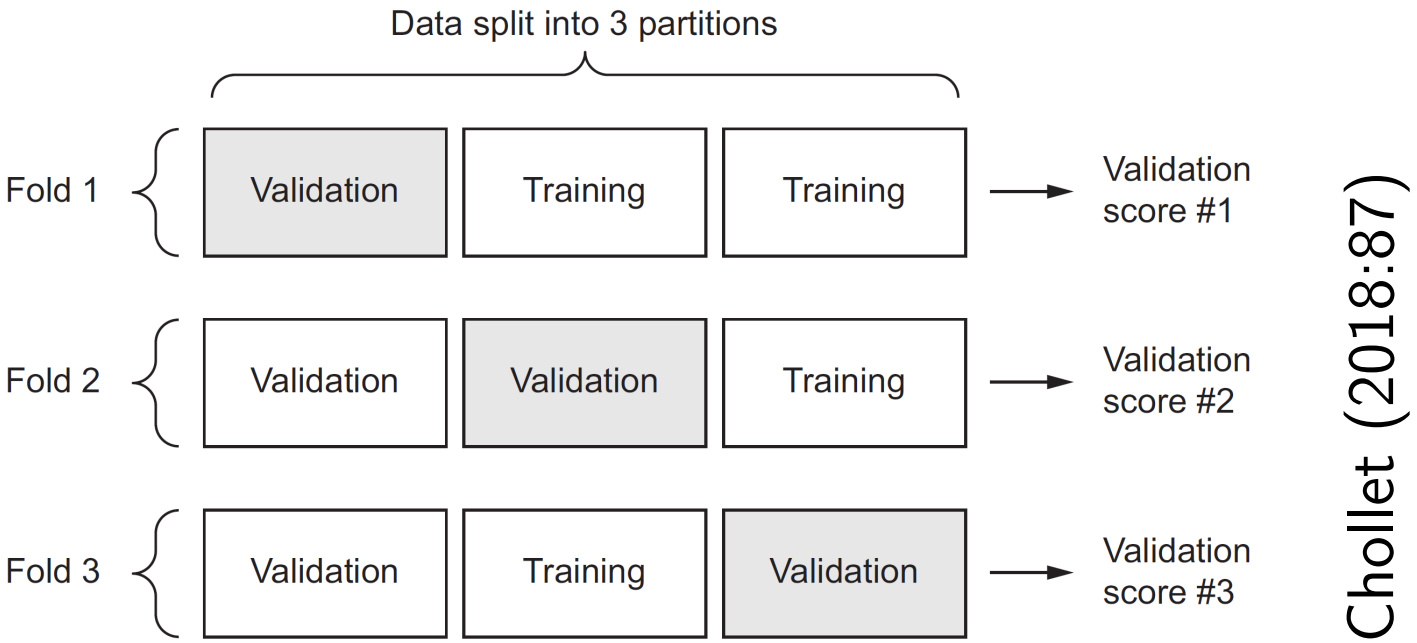
```
test_data -= mean
test_data /= std
```

Building the Network

```
from keras import models
from keras import layers
```

```
def build_model():
    # Because we will need to instantiate
    # the same model multiple times,
    # we use a function to construct it.
    model = models.Sequential()
    model.add(layers.Dense(64,activation='relu',
        input_shape=(train_data.shape[1],)))
    model.add(layers.Dense(64,activation='relu'))
    model.add(layers.Dense(1))
    model.compile(optimizer='rmsprop',
                  loss='mse', metrics=['mae'])
    return model
```

K-fold Validation



Chollet (2018:87)

```
import numpy as np

k = 4
num_val_samples = len(train_data) // k
```

```
num_epochs = 500
all_mae_histories = []
```

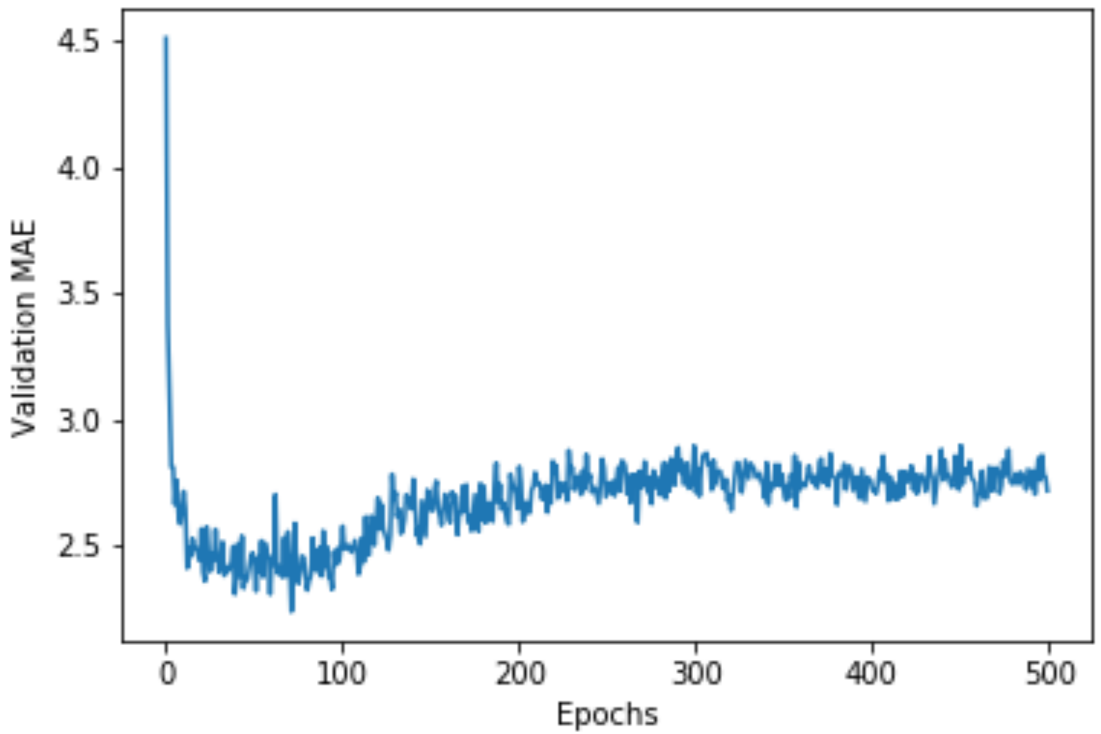
k=4, epochs = 500

```
for i in range(k):
  print('processing fold #', i)
  # Prepare the validation data: data from partition # k
  val_data = train_data[i *num_val_samples: (i + 1) * num_val_samples]
  val_targets = train_targets[i*num_val_samples:(i+1)*num_val_samples]
  # Prepare the training data: data from all other partitions
  partial_train_data = np.concatenate(
        [train_data[:i * num_val_samples],
         train_data[(i + 1) * num_val_samples:]],
        axis=0)
  partial_train_targets = np.concatenate(
        [train_targets[:i * num_val_samples],
         train_targets[(i + 1) * num_val_samples:]],
        axis=0)
  # Build the Keras model (already compiled)
  model = build_model()
  # Train the model (in silent mode, verbose=0)
  history = model.fit(partial_train_data, partial_train_targets,
                      validation_data=(val_data, val_targets),
                      epochs=num_epochs, batch_size=1, verbose=0)
  mae_history = history.history['val_mean_absolute_error']
  all_mae_histories.append(mae_history)
```

Avoid Overfitting

```
average_mae_history = [
   np.mean([x[i] for x in all_mae_histories]
           ) for i in range(num_epochs)]
```

```
import matplotlib.pyplot as plt
plt.plot(range(1, len(average_mae_history) + 1),
         average_mae_history)
plt.xlabel('Epochs')
plt.ylabel('Validation MAE')
plt.show()
```



Performance on Test Data

```
# Get a fresh, compiled model.
model = build_model()
# Train it on the entirety of the data.
model.fit(train_data, train_targets,
          epochs=80, batch_size=16, verbose=0)
test_mse_score, test_mae_score = model.evaluate(test_data,
                                                test_targets)
```

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i|$$

```
test_mae_score
```

```
2.553248433505787 7
```