In [1]:

```
import tensorflow as tf
```

In [2]:

```
tf.__version__
```

Out[2]:

```
'2.0.0'
```

In [3]:

```
from tensorflow.keras import layers, optimizers, metrics, datasets, Sequential, models
```

In [4]:

```
import os
```

In [5]:

```
import matplotlib.pyplot as plt
%matplotlib inline
```

In [6]:

```
(x_train, y_train), (x_test, y_test) = datasets.boston_housing.load_data()
```

In [7]:

```
x_train.shape, y_train.shape, x_test.shape, y_test.shape
```

Out[7]:

```
((404, 13), (404,), (102, 13), (102,))
```

In [8]:

```
import numpy as np
```

归一化

In [9]:

```
mean = x_train.mean(axis=0)
std = x_train.std(axis=0)
```

In [10]:

```
x_train = (x_train - mean) / std
```

In [11]:

```
mean = x_test.mean(axis=0)
std = x_test.std(axis=0)
```

In [12]:

```
x_test = (x_test - mean) / std
```

In [13]:

```
x_train.std()
```

Out[13]:

```
0.9999999999999993
```

In [14]:

```
x_test.mean()
```

Out[14]:

```
-5.894396752157694e-16
```

In [15]:

```
def build_model():
    model=models.Sequential()
    model.add(layers.Dense(64,activation='relu', input_shape=(x_train.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折交叉验证

Type *Markdown* and LaTeX: α^2

Type *Markdown* and LaTeX: α^2

Type *Markdown* and LaTeX: α^2

Type *Markdown* and LaTeX: α^2

Type *Markdown* and LaTeX: α^2

In [16]:

```
k = 4
num_val_samples = len(x_train) // k
num_epochs = 20
all_scores = []
```

In [17]:

```
for i in range(k):
    print('processing fold #', i)
    print('range', i * num_val_samples, (i+1) * num_val_samples)
    # 第k个分区的验证集
    x_val = x_train[i * num_val_samples :  (i+1) * num_val_samples ]
    y_val = y_train[i * num_val_samples :  (i+1) * num_val_samples ]

    # 剩下的作为训练集，不要覆盖原来的x_train，否则会影响下次的训练
    rest_x_train = np.concatenate([ x_train[ : i * num_val_samples], x_train[(i+1) * num_va
    rest_y_train = np.concatenate([ y_train[ : i * num_val_samples], y_train[(i+1) * num_va

    model = build_model()
    model.fit(rest_x_train, rest_y_train, epochs=num_epochs, batch_size=1)

    val_mse, val_mae = model.evaluate(x_val, y_val, verbose=0)

    all_scores.append(val_mae)
```

```
Epoch 6/20
303/303 [==============================] - 0s 633us/sample - loss: 13.023
2 - mae: 2.4341
Epoch 7/20
303/303 [==============================] - 0s 634us/sample - loss: 11.294
1 - mae: 2.2984
Epoch 8/20
303/303 [==============================] - 0s 636us/sample - loss: 11.495
9 - mae: 2.2806
Epoch 9/20
303/303 [==============================] - 0s 631us/sample - loss: 10.506
6 - mae: 2.2321
Epoch 10/20
303/303 [==============================] - 0s 643us/sample - loss: 9.7663
- mae: 2.2016
Epoch 11/20
303/303 [==============================] - 0s 638us/sample - loss: 9.6562
- mae: 2.1480

Epoch 12/20
303/303 [==============================] - 0s 634us/sample - loss: 9.5172
```

In [18]:

```
all_scores
```

Out[18]:

```
[1.9006374, 2.4754179, 2.45792, 2.5455241]
```

In [19]:

```
np.mean(all_scores)
```

Out[19]:

```
2.3448749
```

In [20]:

```
all_mae_histories = []
for i in range(k):
    print('processing fold #', i)
    # 第k个分区的验证集
    x_val = x_train[i * num_val_samples : (i+1) * num_val_samples]
    y_val = y_train[i * num_val_samples : (i+1) * num_val_samples]
    # 剩下的作为训练集
    rest_x_train = np.concatenate([ x_train[ : i * num_val_samples], x_train[(i+1) * num_va
    rest_y_train = np.concatenate([ y_train[ : i * num_val_samples], y_train[(i+1) * num_va

    model = build_model()
    history = model.fit(rest_x_train, rest_y_train,
                        validation_data=(x_val, y_val),
                        epochs=num_epochs, batch_size=1)
#     print(history.history.keys())
    mae_history = history.history['val_mae']
    all_mae_histories.append(mae_history)
```

```
5 - mae: 2.0392 - val_loss: 16.1725 - val_mae: 2.7662
Epoch 15/20
303/303 [==============================] - 0s 813us/sample - loss: 9.6283
- mae: 2.0146 - val_loss: 15.2956 - val_mae: 2.6047
Epoch 16/20
303/303 [==============================] - 0s 809us/sample - loss: 9.8286
- mae: 2.0173 - val_loss: 17.4796 - val_mae: 2.6848
Epoch 17/20
303/303 [==============================] - 0s 809us/sample - loss: 9.6497
- mae: 1.9930 - val_loss: 16.6412 - val_mae: 2.5478
Epoch 18/20
303/303 [==============================] - 0s 809us/sample - loss: 9.1939
- mae: 1.9608 - val_loss: 16.8891 - val_mae: 2.6740
Epoch 19/20
303/303 [==============================] - 0s 829us/sample - loss: 8.7466
- mae: 1.9329 - val_loss: 15.5851 - val_mae: 2.6578
Epoch 20/20
303/303 [==============================] - 0s 813us/sample - loss: 8.5276
- mae: 1.9495 - val_loss: 16.2929 - val_mae: 2.5827
```

In [24]:

```
(np.array(all_mae_histories)).shape
```

Out[24]:

```
(4, 20)
```

计算每一轮得分的平均值

In [26]:

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

In [27]:

```python
ave_mae
```

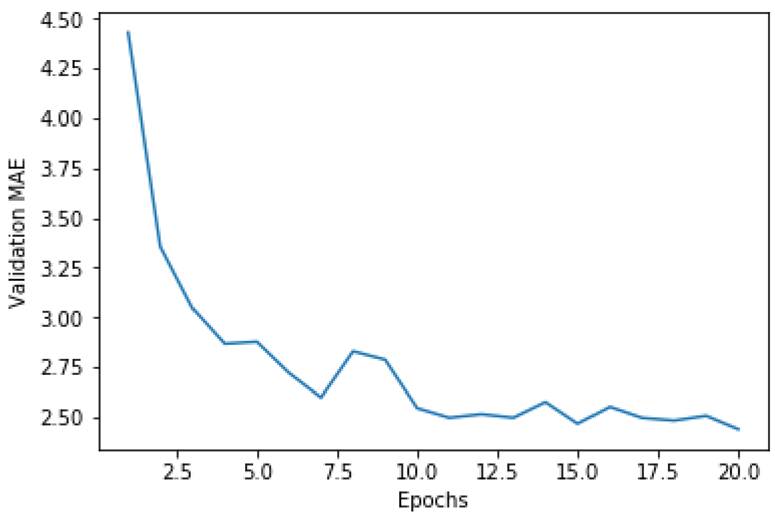
Out[27]:

```
[4.429104,
 3.3556156,
 3.0454452,
 2.8674579,
 2.8774252,
 2.7216148,
 2.5982664,
 2.8295765,
 2.7874641,
 2.5453017,
 2.4974773,
 2.515256,
 2.4976897,
 2.5760195,
 2.4678798,
 2.5520349,
 2.4970536,
 2.4837189,
 2.5073898,
 2.4400122]
```

绘图

In [28]:

```python
plt.plot(range(1, len(ave_mae) + 1), ave_mae)
plt.xlabel('Epochs')
plt.ylabel('Validation MAE')
plt.show()
```



平滑曲线，但是需要把In[20]中的num_epochs改为500

In [ ]:

```
def smooth_curve(points, factor=0.9):
    smoothed_points = []
    for point in points:
        if smoothed_points:
            previous = smoothed_points[-1]
            smoothed_points.append(previous * factor + point * (1 - factor))
        else:
            smoothed_points.append(point)
    return smoothed_points

smooth_mae_history = smooth_curve(average_mae_history[10:])

plt.plot(range(1, len(smooth_mae_history) + 1), smooth_mae_history)
plt.xlabel('Epochs')
plt.ylabel('Validation MAE')
plt.show()
```