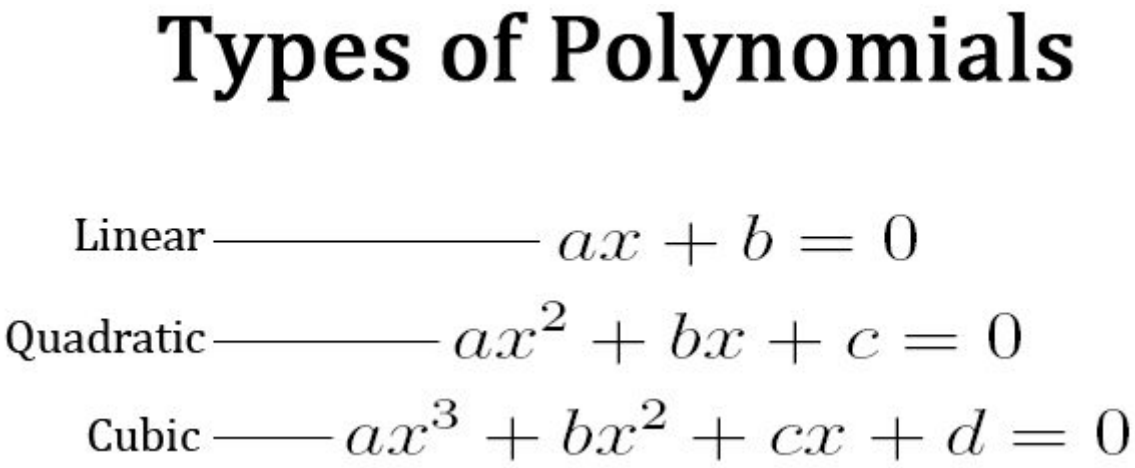
Validation Curves using Polynomial Regression



In [1]:

```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import make_pipeline
def PolynomialRegression(degree=2, **kwargs):
  return make_pipeline(PolynomialFeatures(degree), LinearRegression(**kwargs))
    #เรียงลำดับการทำงาน โดยทำ Polynomial ต่อด้วย Linear Regression
    #**kwargs คือ Parameter ที่ใช้แทน Parameter แบบระบุชื่อ กี่ตัวก็ได้ มี Type เป็น Dictionary
```

In [2]:

```
import numpy as np
rng = np.random.RandomState(1)
X = rng.rand(4, 1)
X
```

Out[2]:

```
array([[4.17022005e-01],
       [7.20324493e-01],
       [1.14374817e-04],
       [3.02332573e-01]])
```

In [3]:

```
X.ravel()
```

Out[3]:

```
array([4.17022005e-01, 7.20324493e-01, 1.14374817e-04, 3.02332573e-01])
```

In [4]:

```
import numpy as np
def make_data(N, err=1.0, rseed=1):
  # randomly sample the data
  rng = np.random.RandomState(rseed) #สร้างตัวเลขสุ่มโดยมีการกระจายค่าความน่าจะเป็น และกำหนด seed เ
  X = rng.rand(N, 1) ** 2 #ทำการสุ่มค่า x จำนวน N ตัว เป็น feature vector (n,1)
  y = 10 - 1. / (X.ravel() + 0.1) #numpy.ravel => row-major order
  if err > 0:
    y += err * rng.randn(N)
  return X, y
X, y = make_data(40)
```

In [5]:

```
X
```

Out[5]:

```
array([[1.73907352e-01],
       [5.18867376e-01],
       [1.30815988e-08],
       [9.14049845e-02],
       [2.15372915e-02],
       [8.52641608e-03],
       [3.46928663e-02],
       [1.19412216e-01],
       [1.57424429e-01],
       [2.90323473e-01],
       [1.75724041e-01],
       [4.69525764e-01],
       [4.18007224e-02],
       [7.71090232e-01],
       [7.50080261e-04],
       [4.49526682e-01],
       [1.74143298e-01],
       [3.12134324e-01],
       [1.97084925e-02],
       [3.92442000e-02],
       [6.41191864e-01],
       [9.37530479e-01],
       [9.82347155e-02],
       [4.79310604e-01],
       [7.68057946e-01],
       [8.00321082e-01],
       [7.23251789e-03],
       [1.52527609e-03],
       [2.88423714e-02],
       [7.71134256e-01],
       [9.67209972e-03],
       [1.77331632e-01],
       [9.17552352e-01],
       [2.84265221e-01],
       [4.78693941e-01],
       [9.95501134e-02],
       [4.71283524e-01],
       [6.96600012e-01],
       [3.34461088e-04],
       [5.62716493e-01]])
```

In [6]:

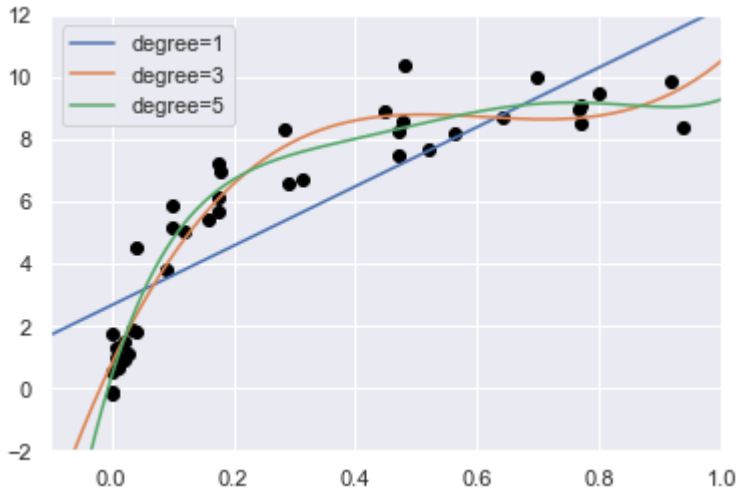
```
y
```

Out[6]:

```
array([ 7.24998644,  7.70041705, -0.12288892,  3.83970623,  1.50418461,
        1.31600899,  1.88404109,  5.0456151 ,  5.42819209,  6.59281674,
        5.70193919,  8.23148858,  1.8305394 ,  9.086429  ,  1.73425177,
        8.92229631,  6.16043712,  6.68597765,  0.8992155 ,  4.51082693,
        8.70162943,  8.39917725,  5.14639037, 10.37406543,  8.96816213,
        9.50648826,  0.9746409 , -0.20201375,  1.09605993,  8.50272859,
        0.67301646,  6.98083184,  9.856233  ,  8.32873282,  8.55755817,
        5.87386864,  7.49515774,  9.997533  ,  0.54626444,  8.1929663 ])
```

In [7]:

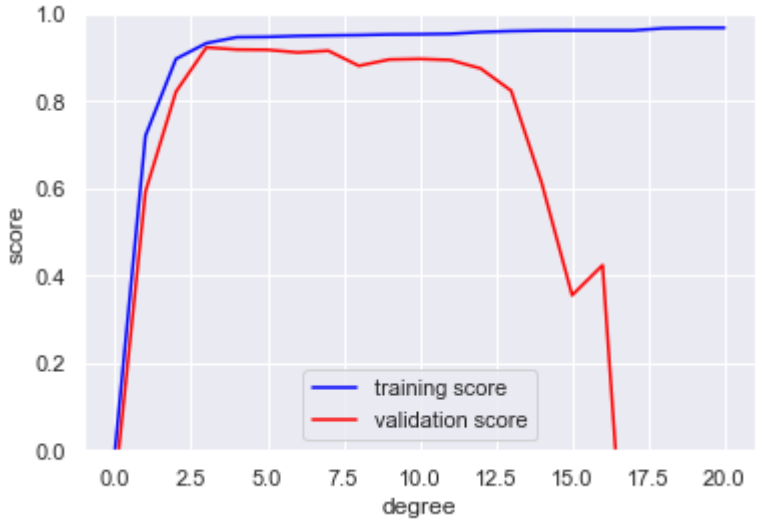
```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn; seaborn.set() # plot formatting
X_test = np.linspace(-0.1, 1.1, 500)[:, None] #สร้างอาร์เรย์ที่เป็นตัวเลขเรียงกัน โดยเริ่มต้นที่ -0.1 และสิ้
plt.scatter(X.ravel(), y, color='black')
axis = plt.axis()
for degree in [1, 3, 5]:
  y_test = PolynomialRegression(degree).fit(X, y).predict(X_test)
  plt.plot(X_test.ravel(), y_test, label='degree={0}'.format(degree))

plt.xlim(-0.1, 1.0)
plt.ylim(-2, 12)
plt.legend(loc='best');
```



Validation Curves

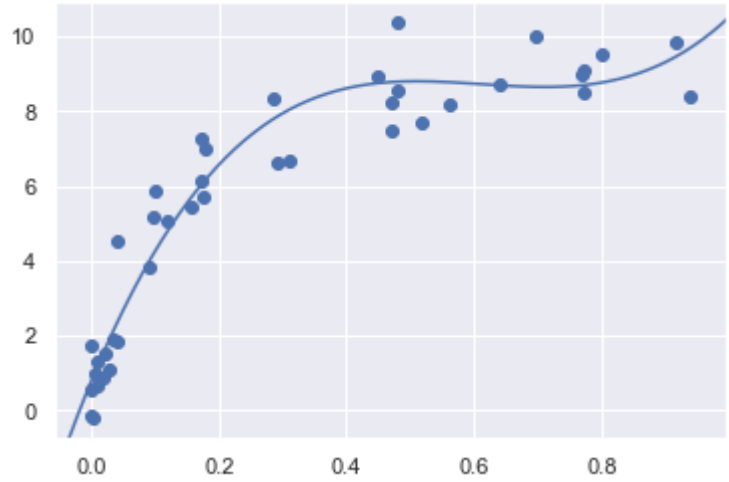
In [8]:

```
#visualizing the validation curve
#https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.validation_curve

from sklearn.model_selection import validation_curve
degree = np.arange(0, 21)

train_score, val_score = validation_curve(PolynomialRegression(), X, y,'polynomialfeatures_

plt.plot(degree, np.median(train_score, 1), color='blue', label='training score')
plt.plot(degree, np.median(val_score, 1), color='red', label='validation score')

plt.legend(loc='best')
plt.ylim(0, 1)
plt.xlabel('degree')
plt.ylabel('score');
```



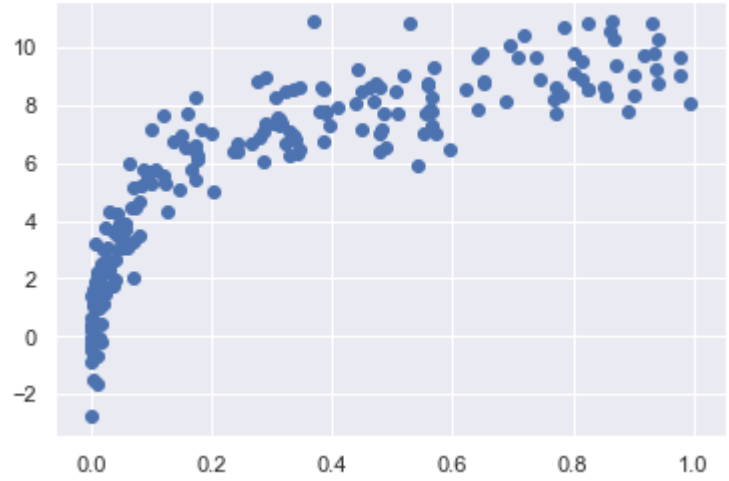
In [9]:

```
#จากการ plot graph จุดที่ดีที่สุดน่าจะอยู่ที่ประมาณ 3
#plot graph

plt.scatter(X.ravel(), y)
lim = plt.axis()
y_test = PolynomialRegression(3).fit(X, y).predict(X_test)
plt.plot(X_test.ravel(), y_test);
plt.axis(lim);
```



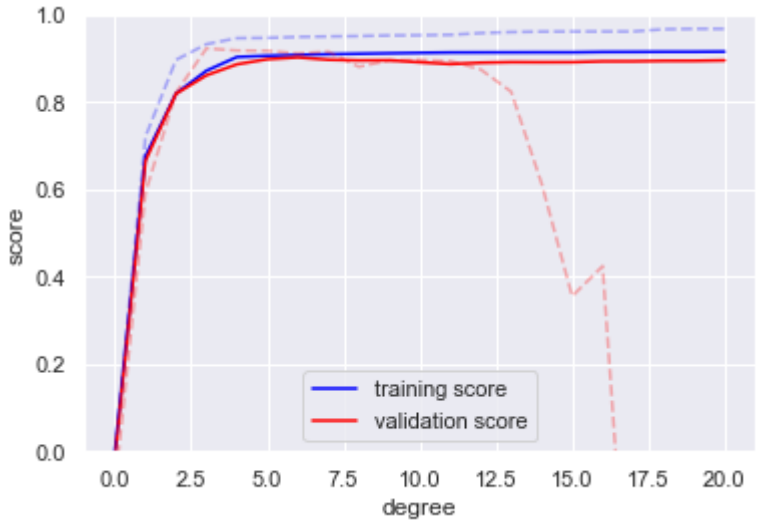
In [10]:

```
X2, y2 = make_data(200) #สร้างข้อมูลใหม่จำนวน 200 ตัว
plt.scatter(X2.ravel(), y2);
```



In [11]:

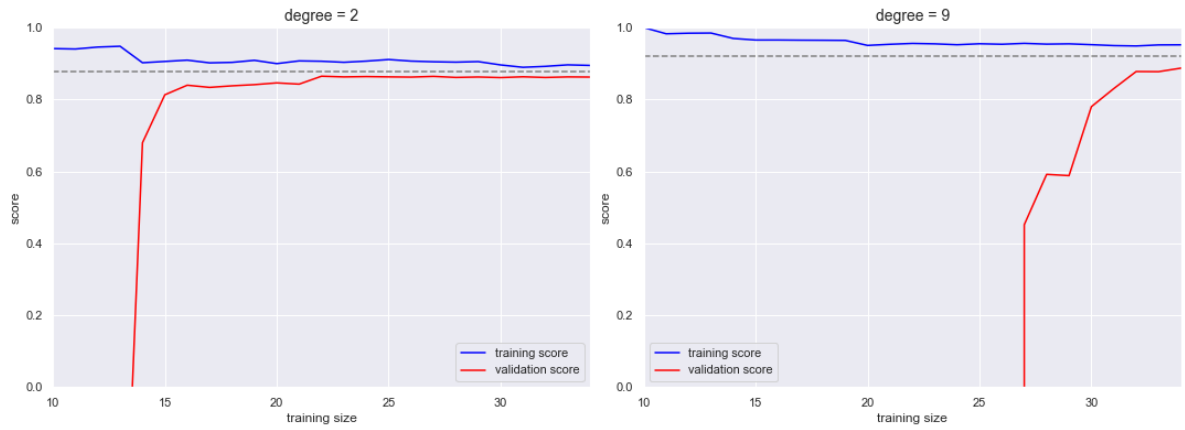
```
#สร้าง validation curves ของข้อมูล X2,y2  ที่สร้างขึ้นมา
degree = np.arange(21) #สร้างตัวเลขตั้งแต่ 0 ถึง 21
train_score2, val_score2 = validation_curve(PolynomialRegression(), X2, y2,'polynomialfeatu

plt.plot(degree, np.median(train_score2, 1), color='blue', label='training score')
plt.plot(degree, np.median(val_score2, 1), color='red', label='validation score')

#คำสั่งเดียวกันกับคำสั่ง visualizing the validation curve ด้านบน
plt.plot(degree, np.median(train_score, 1), color='blue', alpha=0.3,linestyle='dashed')
plt.plot(degree, np.median(val_score, 1), color='red', alpha=0.3,linestyle='dashed')

plt.legend(loc='lower center')
plt.ylim(0, 1)

plt.xlabel('degree')
plt.ylabel('score');
```



Learning Curves

In [12]:

```
from sklearn.model_selection import learning_curve

fig, ax = plt.subplots(1, 2, figsize=(16, 6))
fig.subplots_adjust(left=0.0625, right=0.95, wspace=0.1)

for i, degree in enumerate([2, 9]):
    N, train_lc, val_lc = learning_curve(PolynomialRegression(degree),
                                         X, y, cv=7,train_sizes=np.linspace(0.3, 1, 25))


    ax[i].plot(N, np.mean(train_lc, 1), color='blue', label='training score')
    ax[i].plot(N, np.mean(val_lc, 1), color='red', label='validation score')
    ax[i].hlines(np.mean([train_lc[-1], val_lc[-1]]), N[0], N[-1], color='gray',linestyle='

    ax[i].set_ylim(0, 1)
    ax[i].set_xlim(N[0], N[-1])
    ax[i].set_xlabel('training size')
    ax[i].set_ylabel('score')
    ax[i].set_title('degree = {0}'.format(degree), size=14)
    ax[i].legend(loc='best')
```



พิจารณาเส้นประสีเทา degree = 2

- เมื่อ training และ validation score เข้าใกล้กันแล้ว ต่อให้เพิ่มจำนวน training data ก็ไม่มีผลที่จะทำให้ learning curve ดีขึ้น

degree = 9

- เมื่อเพิ่มความซับซ้อนของโมเดล ทำให้ได้ค่า learning curve เพิ่มขึ้นเมื่อเทียบกับ degree = 2
- แต่ให้สังเกตุด้วยว่าที่จำนวน training size น้อยๆ ส่งผลให้มี variance ที่สูง นั่นคือ หากข้อมูลเรามีน้อยเกินไปก็จะทำให้เกิด overfit ได้

ดังนั้น การ plot learning curve ของ model ที่เราเลือกใช้กับ dataset จะช่วยให้เราตัดสินใจเลือกใช้ model ได้ดีขึ้น

GridSearchCV

In [13]:

```
from sklearn.model_selection import GridSearchCV
param_grid = {'polynomialfeatures__degree': np.arange(21),
'linearregression__fit_intercept': [True, False],
'linearregression__normalize': [True, False]}
grid = GridSearchCV(PolynomialRegression(), param_grid, cv=7) #หา best parameter สำหรับโมเดล
```

In [14]:

```
grid.fit(X, y);
```

In [15]:

```
grid.best_params_ #แสดงค่า parameter ที่ดีที่สุด
```

Out[15]:

```
{'linearregression__fit_intercept': False,
 'linearregression__normalize': True,
 'polynomialfeatures__degree': 4}
```

In [16]:

```
model = grid.best_estimator_
plt.scatter(X.ravel(), y)
lim = plt.axis()
y_test = model.fit(X, y).predict(X_test)
plt.plot(X_test.ravel(), y_test);
plt.axis(lim);
```