多项式曲线拟合

《模式识别与机器学习》$page$：10

使用 sin(x) 函数产生数据集：

```
import numpy as np

n_dots = 200                                             # 产生 200 个数据点
X = np.linspace(-2 * np.pi, 2 * np.pi, n_dots)          # X 取值范围：[-2*pi, 2*pi]
Y = np.sin(X) + 0.2 * np.random.rand(n_dots) - 0.1 # Y 加上随机噪声[-0.1, 0.1]
```

使用 sklearn.preprocessing.PolynomialFeatures 进行特征的构造：

```
from sklearn.preprocessing import PolynomialFeatures
import numpy as np

n_dots = 200
X = np.linspace(-2 * np.pi, 2 * np.pi, n_dots)
Y = np.sin(X) + 0.2 * np.random.rand(n_dots) - 0.1
```

```
X = X.reshape(-1, 1)
Y = Y.reshape(-1, 1)
```

在标准线性回归的情况下，你可能会有一类似于二维数据的模型：

$$\hat{y}(w,x) = w_0 + w_1 x_1 + w_2 x_2$$

如果我们想把数据拟合成抛物面而不是平面，可以结合二阶多项式的特征，使模型看起来像这样：

$$\hat{y}(w,x) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1 x_2 + w_4 x_1^2 + w_5 x_2^2$$

观察，这仍然还是一个线性模型，创造一个新的变量：

$$z = [x_1, x_2, x_1 x_2, x_1^2, x_2^2]$$

使用这些数据的重新标记，原问题可以写成：

$$\hat{y}(w, x) = w_0 + w_1 z_1 + w_2 z_2 + w_3 z_3 + w_4 z_4 + w_5 z_5$$

以上模型是线性模型，可以使用 sklearn.linear_model 进行求解。

PolynomialFeatures 通过构造系数来扩展一个线性回归，其有 3 个参数：

> - degree: 控制多项式的度；
> - interaction_only: 默认为 False，如果指定为 True：不会出现特征与自身结合的项，即没有平方项： x2；
> - include_bias: 默认为 True，如果指定为 True，就会出现 1 那一项。

```
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import Pipeline

def polynomial_model(degree=1):
    polynomial_features = PolynomialFeatures(degree=degree, include_bias=False)
    linear_regression = LinearRegression(normalize=True)
    pipeline = Pipeline([("polynomial_features", polynomial_features),#添加多项式特征
                         ("linear_regression", linear_regression)])
    return pipeline
```

```
from sklearn.metrics import mean_squared_error

degrees = [2, 3, 5, 10]
results = []

for d in degrees:
    model = polynomial_model(degree=d)
    model.fit(X, Y)
    train_score = model.score(X, Y)
    mse = mean_squared_error(Y, model.predict(X))
    results.append({"model": model, "degree": d, "score": train_score, "mse": mse})

for r in results:
```

```
    print("degree: {}; train score: {}; mean squared error: {}".format(r["degree"], r["score"], r["mse"]))
```

```
degree: 2; train score: 0.14623503228246737; mean squared error: 0.4271643274566476
degree: 3; train score: 0.27454911350351763; mean squared error: 0.3629649279959968
degree: 5; train score: 0.8963163750167511; mean squared error: 0.05187604037284656
degree: 10; train score: 0.9934426628788955; mean squared error: 0.003280833258749786
```

比较不同二项式阶数的拟合效果：

```
import matplotlib.pyplot as plt
from matplotlib.figure import SubplotParams

plt.figure(figsize=(12, 6), dpi=200, subplotpars=SubplotParams(hspace=0.3))
for i, r in enumerate(results):
    fig = plt.subplot(2, 2, i+1)
    plt.xlim(-8, 8)
    plt.title("LinearRegression degree={}".format(r["degree"]))
    plt.scatter(X, Y, s=5, c='b', alpha=0.5)
    plt.plot(X, r["model"].predict(X), 'r-')
```