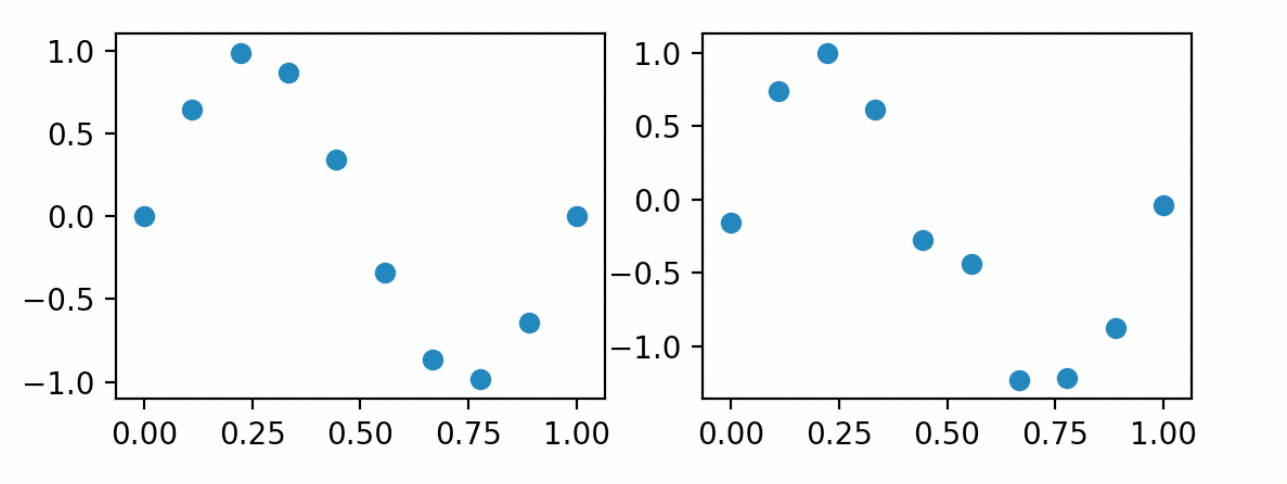
Homework 1.1

1.生成正弦序列

2.使用噪声函数对正弦序列加噪

按照课本中的说明，在0-1区间均匀选择了十个点，求它们的sin(2*pi*x)从而生成了正弦序列，并且添加了高斯噪声，通过实验选取了均值为0，标准差为0.2的高斯噪声。生成了如下图的正弦序列点以及带噪声的正弦序列点：



主要代码如下：

```
x = np.linspace(0., 1., size)
s = np.sin(2*np.pi*x)
```

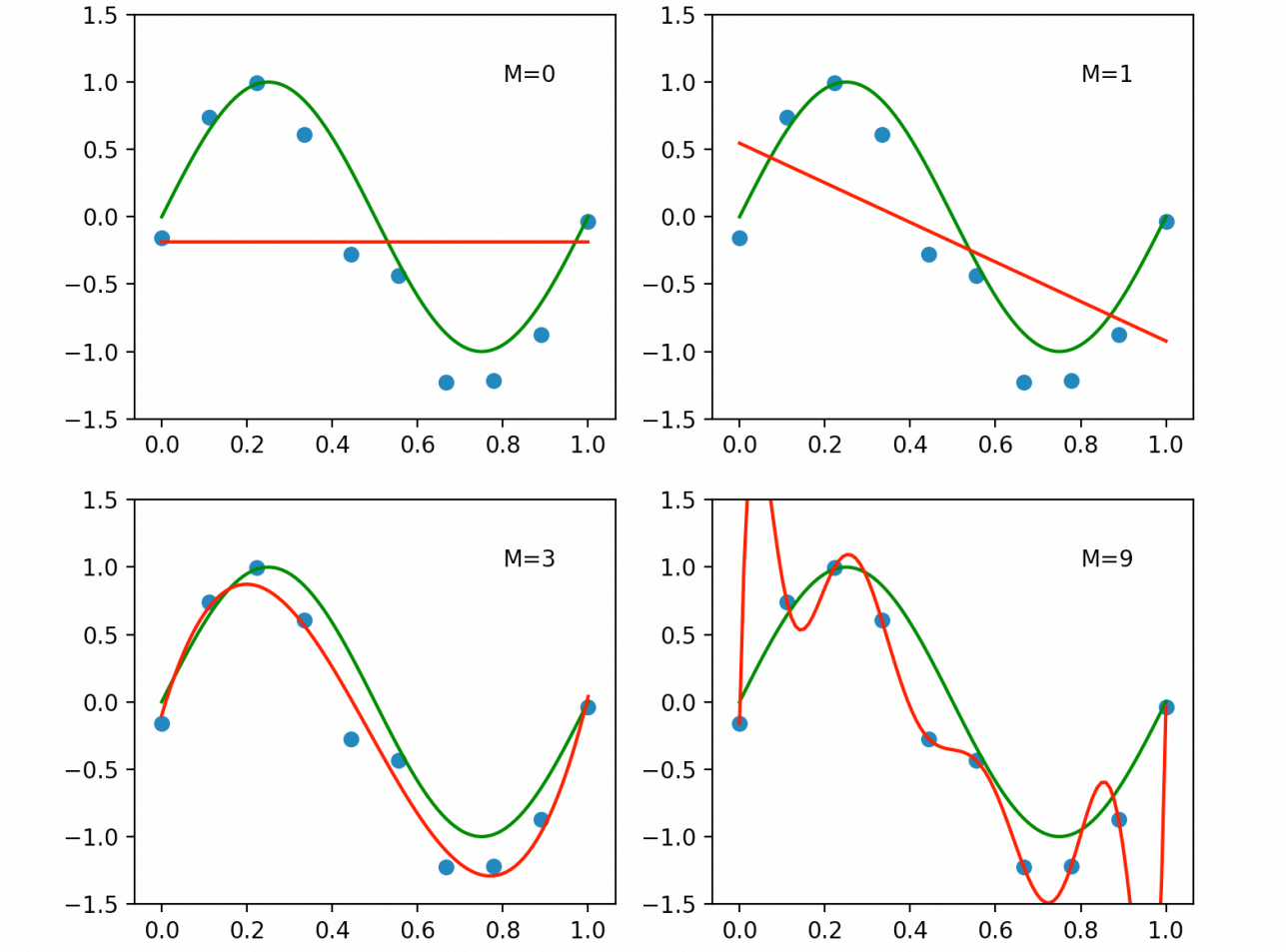
```
mu = 0
sigma = 0.2
w = np.random.normal(mu, sigma, size)
s = s + w
```

3.使用多项式回归模型对序列进行拟合，并分析过拟合和欠拟合情况

参照书上的图1.4，1.6和1.7，以及公式1-2和1-7。我用*sklearn*进行了用不同阶数以及不同正则项的多项式模型去拟合不同数量级的训练数据的实验。

首先是分别用阶数为0，1，3，9的多项式去拟合生成的序列点，可以看下面这张拟合的生成图，蓝色的点是需要拟合的序列点，即训练数据，绿色的线是100个测试数据构成的正弦曲线，红色的线则是多项式拟合曲线。在阶数M为0和1时多项式回归模型没有很好地去拟合训练数据，出现了**欠拟合**的情况。而阶数为3时，它对于训练数据拟合情况不错，对于绿色的测试数据构成的曲线也很接近。而阶数为9时，它对训练数据可以说是十分拟合的（红线直接穿过了每个点），但是却没有很好地表

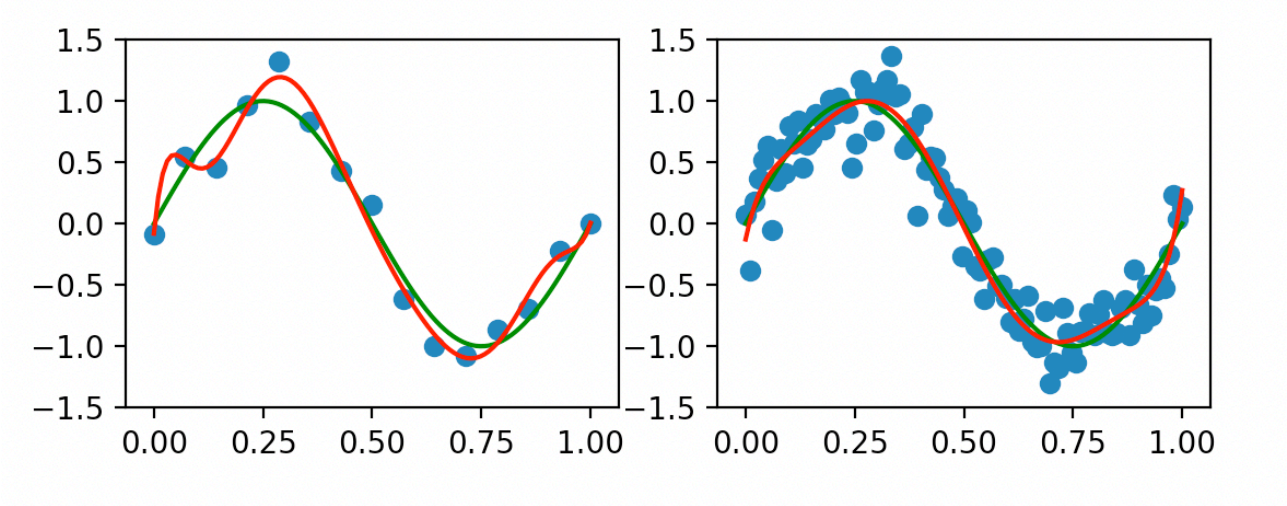
示测试数据的绿色曲线，这就是出现了**过拟合**的情况。因此要适当的选择合适阶数的多项式模型去拟合数据。



上面不同阶数多项式的拟合实验代码如下：

```
# Figure 1.4 Plots of polynomials having various orders M
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import make_pipeline
x_test = np.linspace(0, 1, 100)
y_test = np.sin(2*np.pi*x_test)
X_train = x[:, np.newaxis]
X_test = x_test[:, np.newaxis]
plt.figure(2)
for i, degree in enumerate([0,1,3,9]):
    plt.subplot(2, 2, i + 1)
    model = make_pipeline(PolynomialFeatures(degree), LinearRegression())
    model.fit(X_train, s)
    predict = model.predict(X_test)
    plt.scatter(x, s)
    plt.plot(x_test, y_test, c="g")
    plt.plot(x_test, predict, c="r")
    plt.ylim(-1.5, 1.5)
    plt.annotate("M={}".format(degree), xy=(0.8, 1))
plt.show()
```

之后是当训练数据从15增加到100时，数据在阶数为9的多项式拟合上的变化，如下图所示，当数据量为15时，曲线会出现过拟合的情况，而当数据量增加到100时，过拟合的情况被改善了。这说明了训练数据量的增加会减轻过拟合的问题。



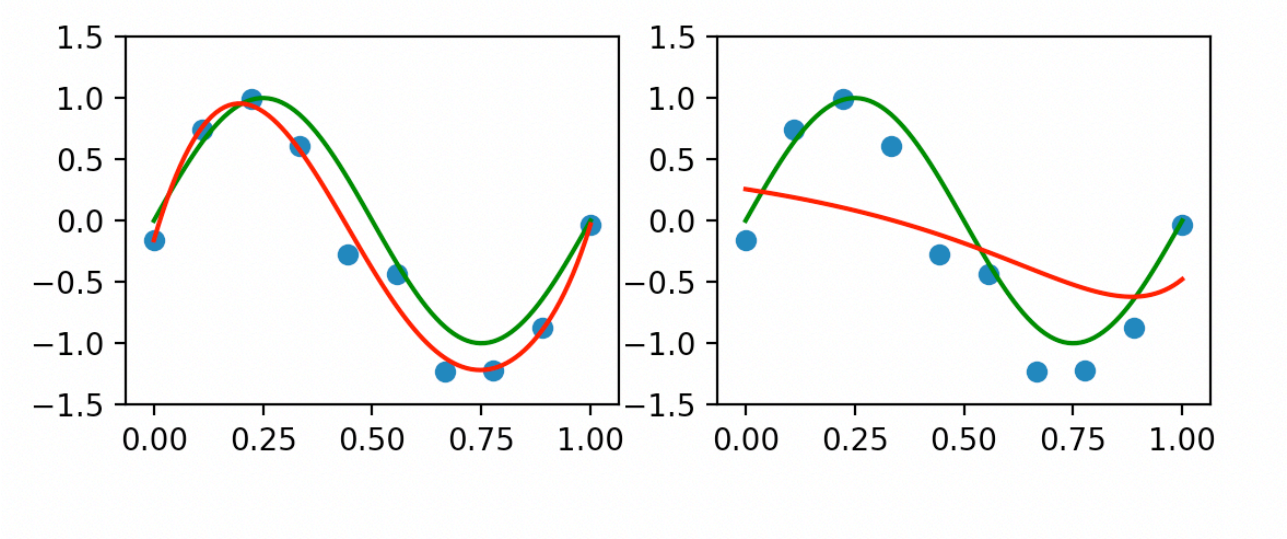
上面分析数据量增加对于拟合的情况的代码如下：

```
# Figure 1.6 using the M = 9 polynomial for N = 15 data points (left plot) and N = 100 data points
def generate_data(x, size):
    return np.sin(2*np.pi*x) + np.random.normal(mu, sigma, size)

x_train_A = np.linspace(0, 1, 15)
x_train_B = np.linspace(0, 1, 100)
y_train_A = generate_data(x_train_A, 15)
y_train_B = generate_data(x_train_B, 100)
X_train_A = x_train_A[:, np.newaxis]
X_train_B = x_train_B[:, np.newaxis]
plt.figure(3)
plt.subplot(221)
model = make_pipeline(PolynomialFeatures(9), LinearRegression())
model.fit(X_train_A, y_train_A)
predict = model.predict(X_test)
plt.scatter(x_train_A, y_train_A)
plt.plot(x_test, y_test, c="g")
plt.plot(x_test, predict, c="r")
plt.ylim(-1.5, 1.5)
plt.subplot(222)
model.fit(X_train_B, y_train_B)
predict = model.predict(X_test)
plt.scatter(x_train_B, y_train_B)
plt.plot(x_test, y_test, c="g")
plt.plot(x_test, predict, c="r")
plt.ylim(-1.5, 1.5)
plt.show()
```

最后是**正则项对于阶数为9的多项式模型拟合数据量为10的训练数据的情况**，可以看下图，左侧为添加了ln(lambda)=-18的正则项的拟合情况，右侧为添加了ln(lambda)=0的正则项的拟合情况，而在前面有一张图为无正则项的阶数为9的多项式模型拟合情况。在没有正则项时，9阶多项式模型拟合这些数据出现了过拟合的情况，而加上了ln(lambda)=-18的正则项后，如左图，过拟合的情况被得到了

抑制，然而当正则项变得过于大后（ln(lambda)=0），就会得到欠拟合的情况。因此要适当选择正则系数lambda，要即能够解决过拟合的情况而又不会导致欠拟合。



正则项对于多项式拟合的影响的实验代码如下：

```
# Figure 1.7 Plots of M = 9 polynomials for two values of the regularization parameter \
# λ corresponding to ln λ = -18 and ln λ = 0.
import math
from sklearn.linear_model import Ridge
plt.figure(4)
plt.subplot(221)
model = make_pipeline(PolynomialFeatures(9), Ridge(solver='lsqr',alpha=math.exp(-18)))
model.fit(X_train, s)
predict = model.predict(X_test)
plt.scatter(x, s)
plt.plot(x_test, y_test, c="g")
plt.plot(x_test, predict, c="r")
plt.ylim(-1.5, 1.5)
plt.subplot(222)
model = make_pipeline(PolynomialFeatures(9), Ridge(solver='lsqr',alpha=math.exp(0)))
model.fit(X_train, s)
predict = model.predict(X_test)
plt.scatter(x, s)
plt.plot(x_test, y_test, c="g")
plt.plot(x_test, predict, c="r")
plt.ylim(-1.5, 1.5)
plt.show()
```