```
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.model_selection  import train_test_split
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
%matplotlib inline
np.random.seed(100)
data_size = 100
X = np.random.randn(data_size)
y = 5 * X + 10 + np.random.randn(data_size)
fig = plt.figure()
ax = fig.add_subplot(1,1,1)
ax.scatter(X, y)
ax.set_title(u"x and y scatter plot")
ax.set_xlabel("X")
ax.set_ylabel("y")
plt.show()
```

数値計算用ライブラリ

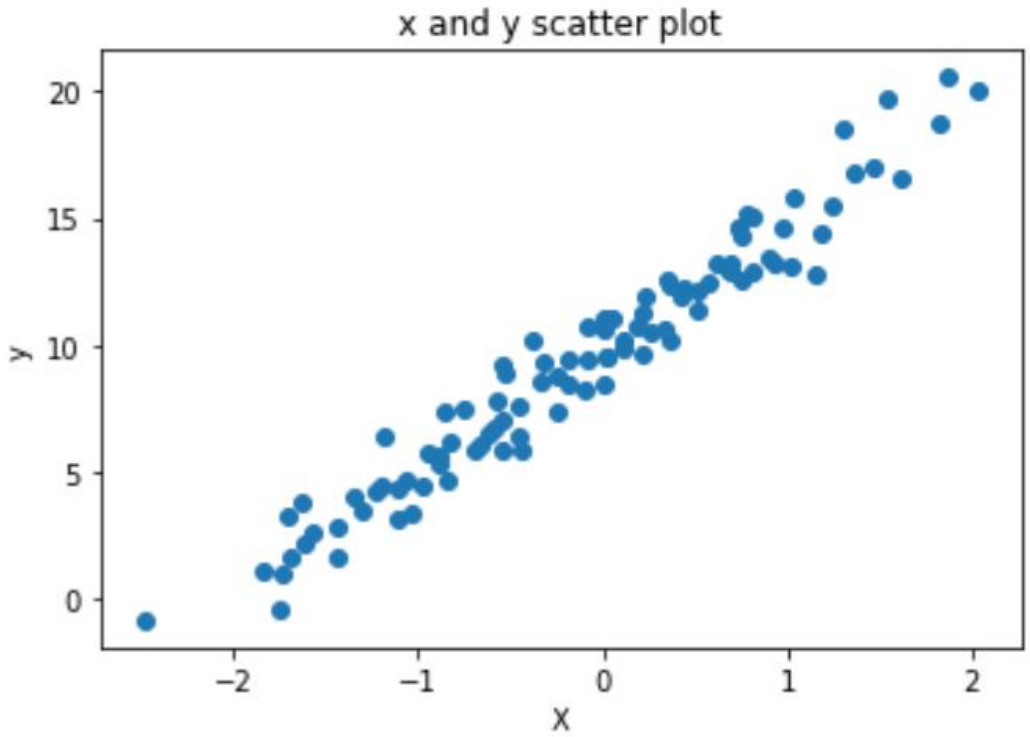
機械学習用ライブラリScikit-learn

グラフ作成用ライブラリ

いつも同じ乱数に固定

標準正規分布の乱数

誤差



```
X_train,X_test,y_train,y_test=¥
train_test_split(X,y,test_size=0.3,random_state=0)
print("X_trian :", X_train.shape)
print("X_test :", X_test.shape)
print("y_trian :", y_train.shape)
print("y_test :", y_test.shape)
lr = LinearRegression()
lr.fit(X_train.reshape(-1, 1), y_train)
print("傾き:",lr.coef_)
print("切片:",lr.intercept_)
y_pred = lr.predict(X_test.reshape(-1, 1))
print('予測結果¥n', y_test[:5])
print('答え¥n', y_pred[:5])
print('トレーニングデータの平均乗誤差:',¥
      mean_squared_error(y_train,
lr.predict(X_train.reshape(-1, 1))))
print('テストデータの平均2乗誤差:',
mean_squared_error(y_test, y_pred))
```

＼もしくは￥は、改行無視

Xとyのデータを指定した割合で分割
学習用ﾃﾞｰﾀ:ﾃｽﾄﾃﾞｰﾀ＝7:3

残差の2乗の合計が最小となる勾配と切片を算出
lr.fit（データ）

←線形回帰

傾き: [4.87800146]
切片: 9.88020924346351 1

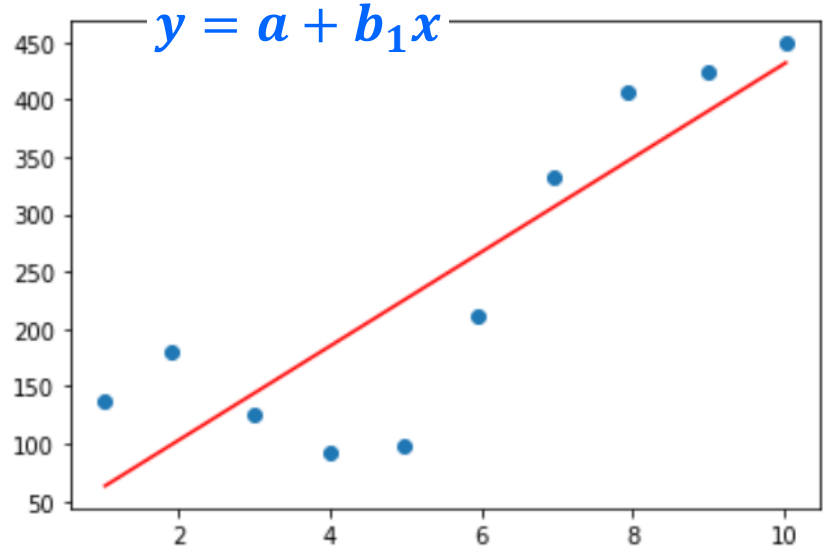
予測結果 [14.6040065 9.83994175 12.79186354 11.39130635 9.23670159]
答え [13.44602999 10.41128697 15.50471957 12.35730517 7.19382354]

トレーニングデータの平均乗誤差: 1.050530750671141
テストデータの平均2乗誤差: 1.238211079168313 6

```
[[ 1.028]
 [ 1.911]
 [ 3.002]
 [ 3.985]
 [ 4.977]
 [ 5.936]
 [ 6.957]
 [ 7.917]
 [ 9.002]
 [10.018]]
[[136.7467638 ]
 [179.3086724 ]
 [124.7939742 ]
 [ 91.608693  ]
 [ 99.06056946]
 [211.3147593 ]
 [331.87715   ]
 [407.3815    ]
 [425.1095402 ]
 [449.1164546 ]]
```

rmse : 67.44116527130275
R2 : 0.7534254734665234

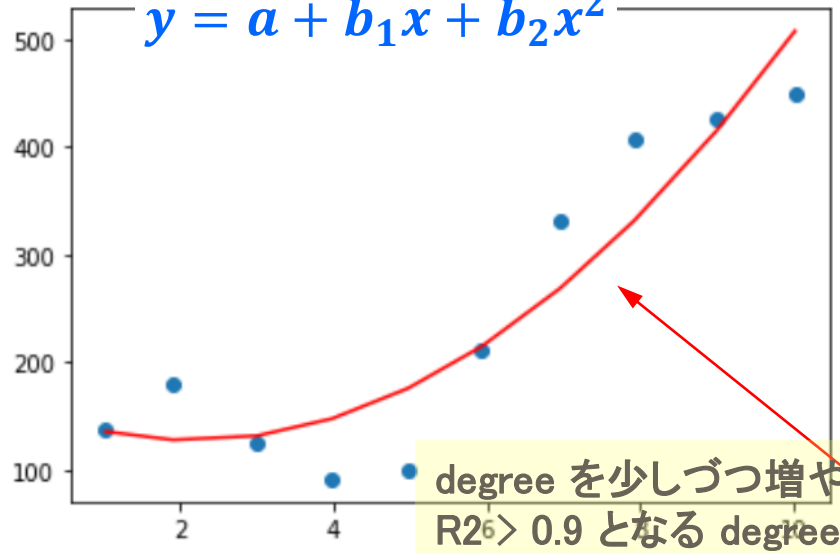
$y = a + b_1 x$



rmse : 50.06437014920699
R2 : 0.8641199837148785

degree

$y = a + b_1 x + b_2 x^2$



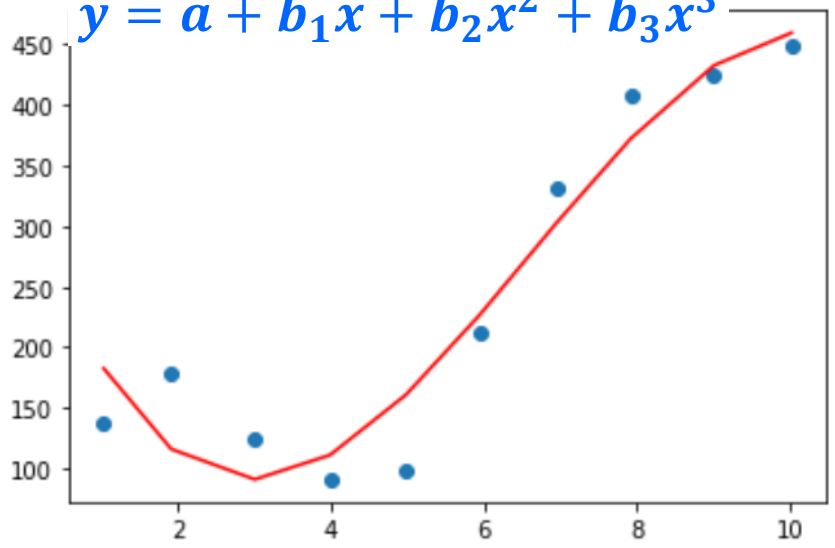
決定係数

degree	R2
1	0.753
2	0.864
3	0.925
10	1.000

degree を少しづつ増やし、R2などの評価を確認
R2> 0.9 となる degree（多項式）を採用

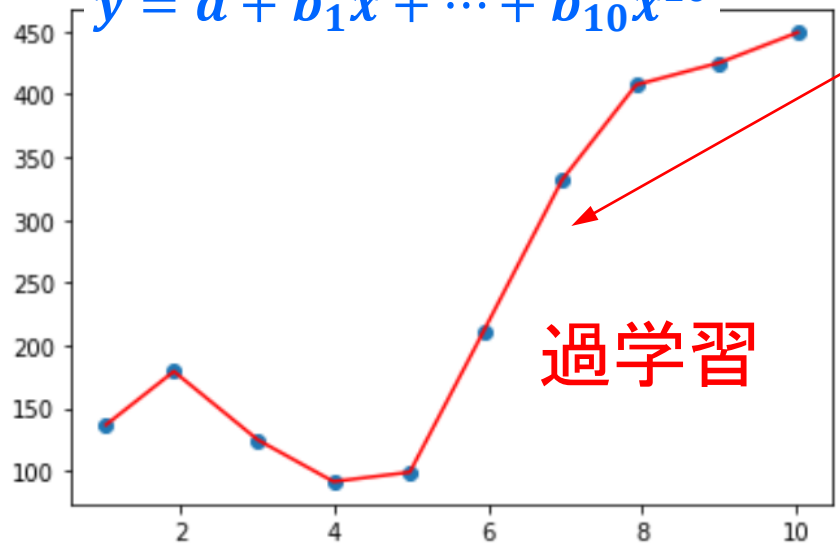
rmse : 37.231982479228975
R2 : 0.9248497551567119

$y = a + b_1 x + b_2 x^2 + b_3 x^3$



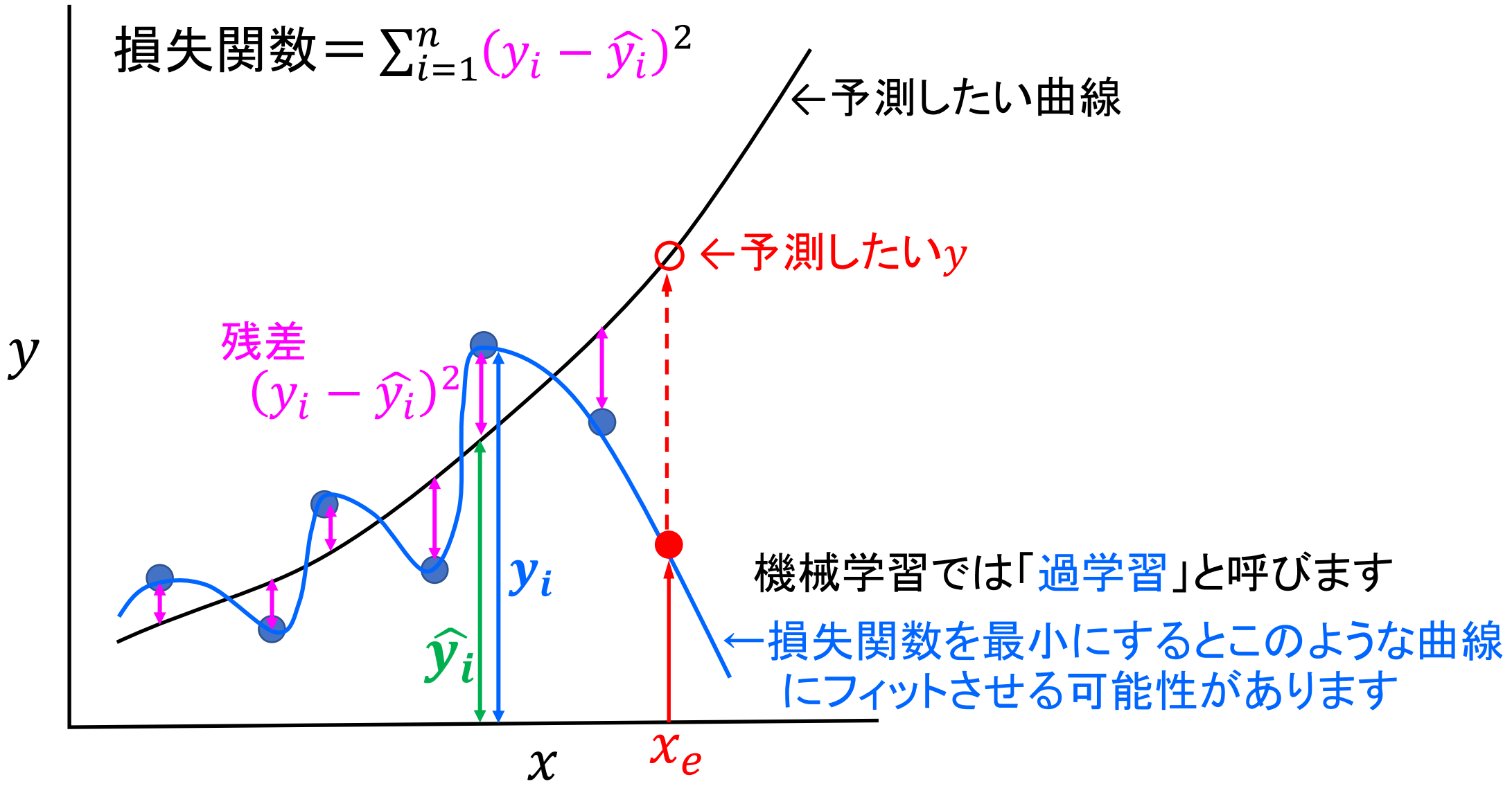
rmse : 3.635961863135608e-06
R2 : 0.9999999999999993

$y = a + b_1 x + \cdots + b_{10} x^{10}$



どちらが良さそう？

過学習


損失関数＝$\sum_{i=1}^{n}(y_i-\hat{y}_i)^2$
←予測したい曲線
←予測したいy
残差
$(y_i-\hat{y}_i)^2$
y
y_i
$\hat{y}_i$
x
x_e
機械学習では「過学習」と呼びます
←損失関数を最小にするとこのような曲線
にフィットさせる可能性があります

損失関数＝$\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$

過学習を防ぐためこの項を加える

リッジ回帰：　重みの二乗の合計を足したもの

$$\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \sum_{I=1}^{k} {\omega_i}^2$$

LASSO回帰：　重みの合計を足したもの

$$\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \sum_{i=1}^{k} |\omega_i|$$

リッジ回帰

```
from sklearn.linear_model import Ridge
rg = Ridge(alpha=0.3, random_state=1234)
rg.fit(X_train.reshape(−1, 1), y_train)
```

Alphaは罰則の重み

Ridge(alpha=0.3, copy_X=True, fit_intercept=True, max_iter=None, normalize=False, random_state=1234, solver='auto', tol=0.001)

```
print(″傾き:″,rg.coef_)
print(″切片:″,rg.intercept_)
```

傾き: [4.85673346]
切片: 9.878280306770973

```
y_pred = rg.predict(X_test.reshape(-1, 1))
print('予測結果¥n', y_test[:5])
print('答え¥n', y_pred[:5])
```

予測結果 [14.6040065 9.83994175 12.79186354 11.39130635 9.23670159]
答え [13.42855414 10.40704254 15.47826787 12.34457614 7.20360719]

```
print('トレーニングデータの平均乗誤差:',¥
    mean_squared_error(y_train,
rg.predict(X_train.reshape(-1, 1))))
print('テストデータの平均2乗誤差:',
mean_squared_error(y_test, y_pred)
```

トレーニングデータの平均乗誤差: 1.050973435013048
テストデータの平均2乗誤差: 1.231116293633471 6

LASSO回帰

```
from sklearn.linear_model import Lasso
la = Lasso(alpha=0.3, random_state=1234)
la.fit(X_train.reshape(−1, 1), y_train)
```

Ridge(alpha=0.3, copy_X=True, fit_intercept=True, max_iter=None,
normalize=False, random_state=1234, solver='auto', tol=0.001)

```
print("傾き:",la.coef_)
print("切片:",la.intercept_)
```

傾き: [4.85673346]
切片: 9.878280306770973

```
y_pred = la.predict(X_test.reshape(-1, 1))
print('予測結果¥n', y_test[:5])
print('答え¥n', y_pred[:5])
```

予測結果 [14.6040065 9.83994175 12.79186354 11.39130635 9.23670159]
答え [13.42855414 10.40704254 15.47826787 12.34457614 7.20360719]

```
print('トレーニングデータの平均2乗誤差:', ¥
    mean_squared_error(y_train,
la.predict(X_train.reshape(-1, 1))))
print('テストデータの平均2乗誤差:',
mean_squared_error(y_test, y_pred))
```

トレーニングデータの平均乗誤差: 1.050973435013048
テストデータの平均2乗誤差: 1.2311162936334716

	傾き	切片	平均2乗誤差	
			トレーニングデータ	テストデータ
線形回帰	4.878	9.880	**1.0505**	**1.2382**
リッジ回帰	4.857	9.878	**1.0510**	**1.2311**
LASSO回帰	4.857	9.878	**1.0510**	1.2311

```
fig = plt.figure()
ax = fig.add_subplot(1,1,1)
ax.scatter(X_test, y_test)
ax.set_title(u"x and y scatter plot")
ax.plot(X_test, lr.predict(X_test.reshape(-1, 1)), color = 'r',
        linestyle="-", label="LinearRegression")
ax.plot(X_test, rg.predict(X_test.reshape(-1, 1)), color = 'b',
        linestyle="--", label="Ridge")
ax.plot(X_test, la.predict(X_test.reshape(-1, 1)), color = 'g',
        linestyle=":", label="Lasso")
ax.set_xlabel("x")
ax.set_ylabel("y")
ax.legend(loc="best")
plt.show()
```