線形回帰

課題1

In [6]:
```python
# Setup
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

try:
    train_data1 = pd.read_csv('week6/data/temp_ene.csv')
except FileNotFoundError:
    train_data1 = pd.read_csv('./data/temp_ene.csv')

x = train_data1['temp'].values.reshape(-1, 1)
y = train_data1['ene'].values.reshape(-1, 1)
```

線形回帰モデルによる予測$\hat{y}$は、特徴量ベクトル$\mathbf{x}$とパラメータベクトルθの内積と考えることができる。

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n = \boldsymbol{\theta} \cdot \boldsymbol{x}$$

バイアス項θ_0を用いるために特徴量ベクトルに$x_0 = 1$を追加する必要がある事に留意する。
これはscikit-learnのadd_dummy_featureを使えば実装できる。

In [7]:
```python
from sklearn.preprocessing import add_dummy_feature
x_original = x.copy()
x = add_dummy_feature(x, value=1)
```

インスタンス数をmとすると、特徴量を格納したサイズ$m \times (n+1)$の行列$\mathbf{X}$とターゲット値を格納したベクトル$\mathbf{y}$を用いて平均二乗誤差(Mean Square Error)は

$$MSE(\mathbf{X}, h_\theta) = \frac{1}{m} \sum_{i=1}^{m} \left(\boldsymbol{\theta}^\intercal \boldsymbol{x}^{(i)} - y^{(i)} \right)^2 = \frac{1}{m} \|\mathbf{X}\boldsymbol{\theta} - \mathbf{y}\|^2$$

と表せる。
損失関数 $MSE(\boldsymbol{\theta})$ の $\boldsymbol{\theta}$ による勾配（偏微分）は

$$\frac{\partial MSE(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \propto \mathbf{X}^\top (\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$$

となるため、損失関数が最小となる最適なパラメータは

$$\boldsymbol{\theta} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

と計算できる。これはpythonの行列演算により実装できる。

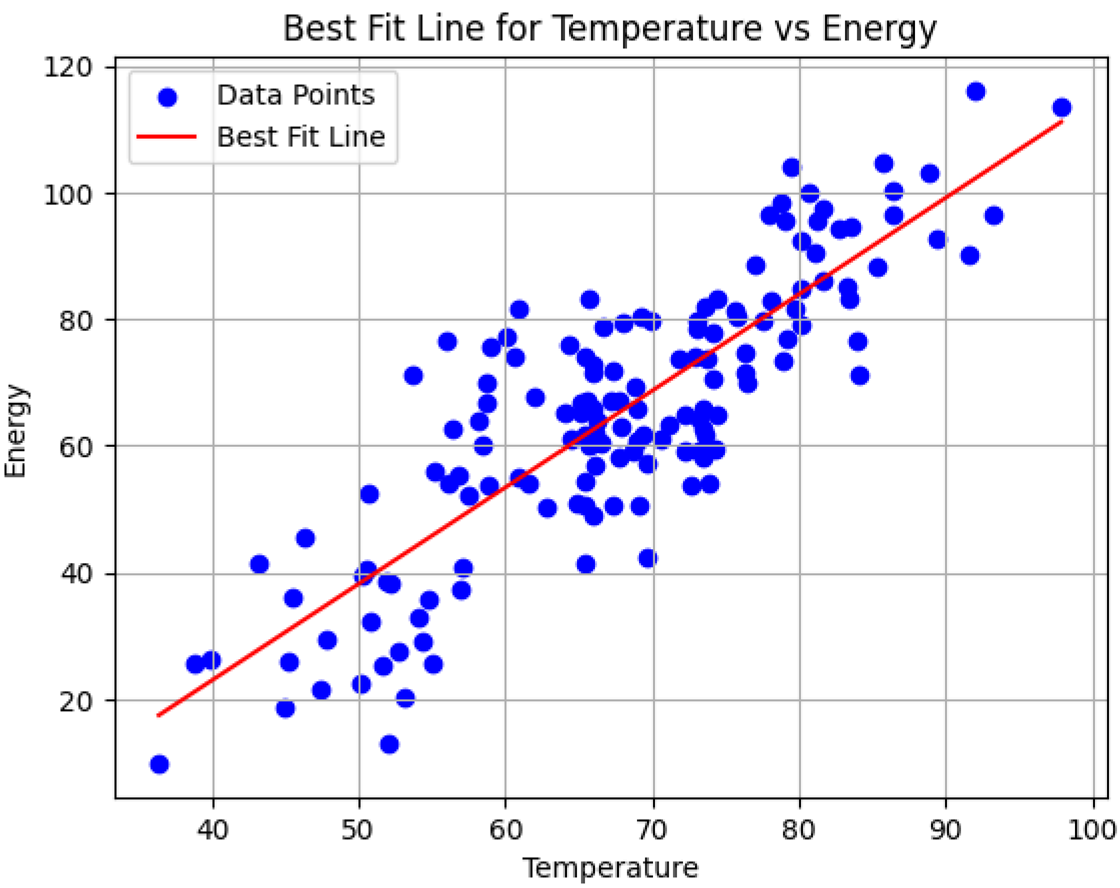
In [8]:
```python
best_theta = np.linalg.inv(x.T @ x) @ x.T @ y
print(f"Best theta: {best_theta}")
y_pred = x @ best_theta
```

```
Best theta: [[-37.89032132]
 [  1.52262384]]
```

In [9]:
```
# 上の値がScikit-Learnの結果と一致することを確認
from sklearn.linear_model import LinearRegression
model = LinearRegression(fit_intercept=False)
model.fit(x, y)
print(f"Scikit-learn theta: {model.coef_.T}")
```

```
Scikit-learn theta: [[-37.89032132]
 [  1.52262384]]
```

In [10]:
```
# モデルの図示
plt.scatter(x_original, y, label='Data Points', color='blue')
plt.plot(x_original, y_pred, label='Best Fit Line', color='red')
plt.xlabel('Temperature')
plt.ylabel('Energy')
plt.title('Best Fit Line for Temperature vs Energy')
plt.legend()
plt.grid()
plt.show()
```



課題2

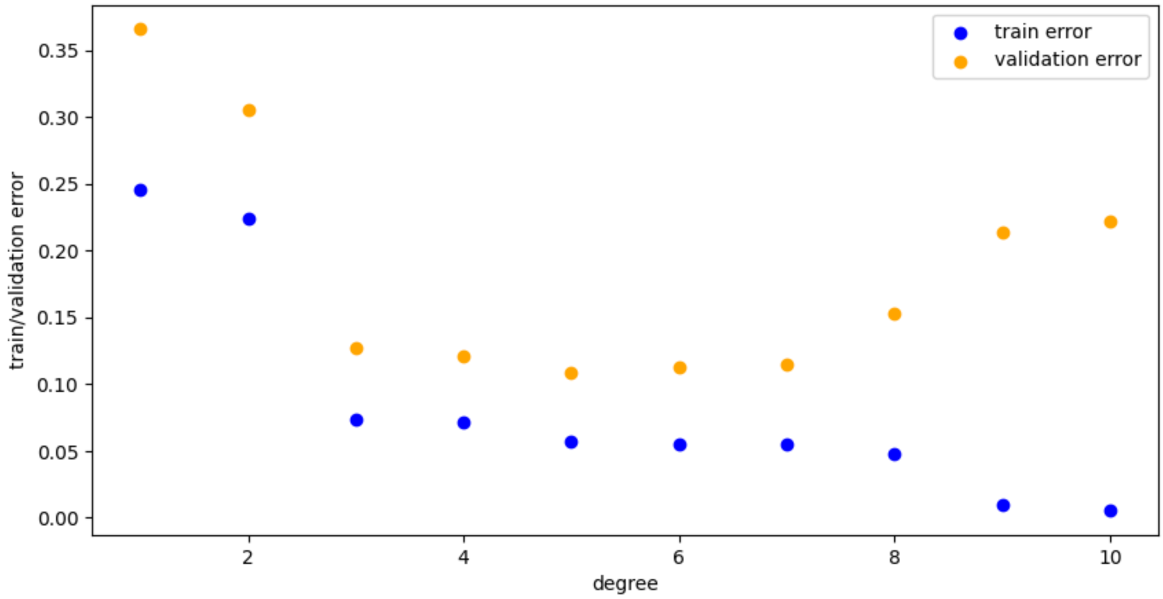
多項式回帰は、各特徴量の累乗を新しい特徴量として追加し、線形モデルを学習させる事で実装できる。すなわち先ほどのベクトルの要素で言うとx_2が$(x_1)^2$、x_3が$(x_1)^3$になる。パラメータの最適化には先ほどと同様に正規方程式を用いる。
高次特徴量の追加には、Scikit-LearnのPolynomialFeaturesを使う。
以下のコードはn=1~10の範囲でn次元多項式回帰を行い、学習セットと検証セットそれぞれでの誤差を出力するものである。

In [16]:
```python
try:
    train_data2 = pd.read_csv('week6/data/train_data.csv')
    valid_data2 = pd.read_csv('week6/data/validation_data.csv')
except FileNotFoundError:
    train_data2 = pd.read_csv('./data/train_data.csv')
    valid_data2 = pd.read_csv('./data/validation_data.csv')
x2 = train_data2['x'].values.reshape(-1, 1)
y2 = train_data2['y'].values.reshape(-1,1)
y2_valid = valid_data2['y_valid'].values.reshape(-1, 1)
train_error = []
valid_error = []
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error
for n in range(1,11):
    poly=PolynomialFeatures(degree=n, include_bias=True) # degree=nはn次多項式回
    X_poly = poly.fit_transform(x2) # Xをn次多項式回帰用に変換
    theta = np.linalg.inv(X_poly.T @ X_poly) @ X_poly.T @ y2 # 正規方程式によるパ
    y_pred = X_poly @ theta # トレーニングデータに対する予測値

    train_error.append(mean_squared_error(y_pred, y2)) # トレーニングデータの二乗
    valid_error.append(mean_squared_error(y_pred, y2_valid)) # 検証データの二乗誤

    print(f'Polynomial degree: {n}') # 多項式の次数
    print(f'train data: S={mean_squared_error(y_pred, y2)}') # yの予測値とトレーニ
    print('validation data: S=',mean_squared_error(y_pred, y2_valid)) # yの予測値
```

```
Polynomial degree: 1
train data: S=0.24567619345803174
validation data: S= 0.3658868811838186
Polynomial degree: 2
train data: S=0.2241201590057087
validation data: S= 0.30549917163805496
Polynomial degree: 3
train data: S=0.0730323538972319
validation data: S= 0.12670139024713864
Polynomial degree: 4
train data: S=0.07114111380244
validation data: S= 0.12123535510270965
Polynomial degree: 5
train data: S=0.05670371301779766
validation data: S= 0.10884713952670572
Polynomial degree: 6
train data: S=0.05537847193738201
validation data: S= 0.11273426710523424
Polynomial degree: 7
train data: S=0.054855234446740925
validation data: S= 0.1151495101392312
Polynomial degree: 8
train data: S=0.047962076139622514
validation data: S= 0.15251750970262468
Polynomial degree: 9
train data: S=0.009776351967132967
validation data: S= 0.21312912859016375
Polynomial degree: 10
train data: S=0.005185354749008788
validation data: S= 0.22156625135835728
```

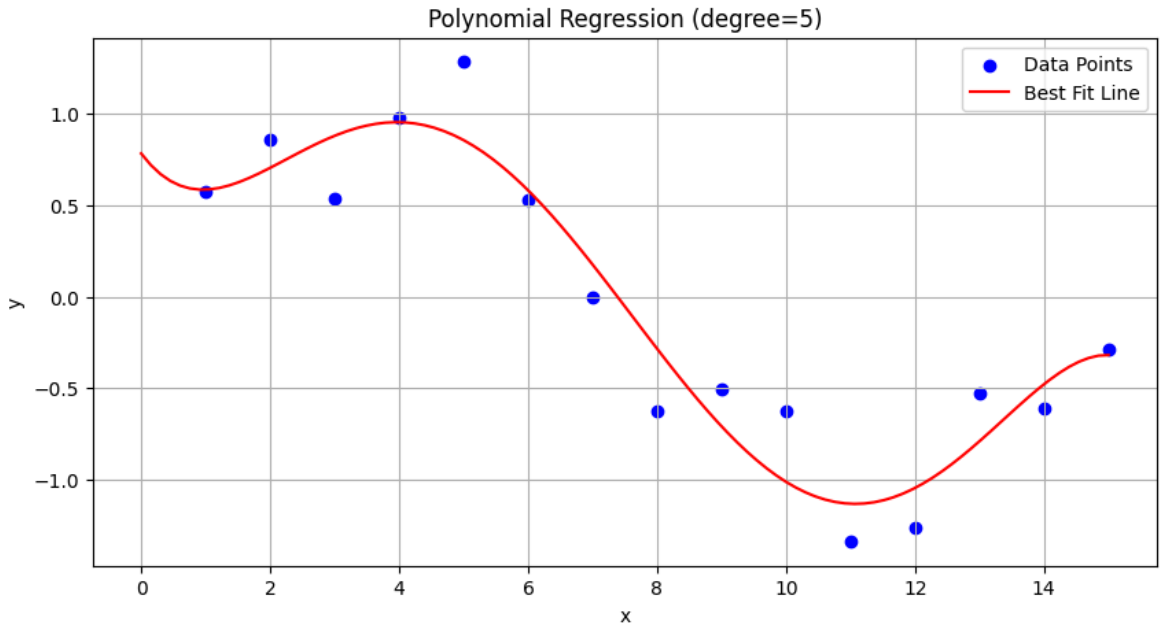
In [17]:
```python
# Plot Data and Regression line
x = range(1, 11) # 多項式の次数
```

```
plt.figure(figsize=(10, 5))
plt.scatter(x, train_error, color='blue', label='train error') # トレーニングデー
plt.scatter(x, valid_error, color='orange', label='validation error') # 検証デー

plt.legend()
plt.xlabel('degree'); plt.ylabel('train/validation error')
plt.show()
```



各次数での誤差をプロットしてみると、n=8から過剰適合を始めてることが分かる。
以下は検証セットでの誤差が最小であった5次のモデルの図示である。

In [19]:
```
poly=PolynomialFeatures(degree=5, include_bias=True) # degree=nはn次多項式回帰
X_poly = poly.fit_transform(x2) # Xをn次多項式回帰用に変換
theta = np.linalg.inv(X_poly.T @ X_poly) @ X_poly.T @ y2 # 正規方程式によるパラメ
x = np.linspace(0,15,100).reshape(-1, 1) # xの範囲を-1から1に設定
x_poly = poly.fit_transform(x) # xをn次多項式回帰用に変換
y_pred = x_poly @ theta # トレーニングデータに対する予測値
# Plot Data and Regression Line
plt.figure(figsize=(10, 5))
plt.scatter(x2, y2, color='blue', label='Data Points') # トレーニングデータをプロ
plt.plot(x, y_pred, color='red', label='Best Fit Line') # 回帰直線をプロット(赤)
plt.xlabel('x'); plt.ylabel('y')
plt.title('Polynomial Regression (degree=5)')
plt.legend()
plt.grid()
plt.show()
```



課題3

リッジ回帰では、損失関数にパラメータベクトルのl_2ノルムにハイパーパラメータαをかけたものを加える。

$$J(\boldsymbol{\theta}) = MSE(\boldsymbol{\theta}) + \frac{\alpha}{m}\sum_{i=1}^{n}\theta_i^2$$

リッジ回帰の閉形式解:

$$\hat{\theta} = (\mathbf{X}^\top\mathbf{X} + \alpha\mathbf{A})^{-1}\mathbf{X}^\top\mathbf{y}$$

ここで$\mathbf{A}$は単位行列。以下のコードは最適なパラメータを$\alpha = 0, 0.1, 1$の場合でそれぞれ正規方程式によって求めるものである。

In [24]:
```
try:
    file=pd.read_csv('./data/data_for_ridge.csv') #データの読み込み
except FileNotFoundError:
    file=pd.read_csv('week6/data/data_for_ridge.csv')

X = np.array(file['x'], dtype=float).reshape(-1,1)
y = np.array(file['y'], dtype=float)

poly = PolynomialFeatures(degree=3, include_bias=True) # degree=nはn次多項式回帰
X_poly = poly.fit_transform(X) # Xをn次多項式回帰用に変換
# Ridge Regression
alpha = [0,0.1,1]
I = np.eye(X_poly.shape[1]) # 単位行列
theta = []
for i in range(3):
    theta.append(np.linalg.inv(X_poly.T @ X_poly + alpha[i] * I) @ X_poly.T @ y)

# Plot Data and Regression Line(グラフ作成)
x0 = np.linspace(1, 7).reshape(-1, 1) # 回帰曲線の描画用
color_list = ['red', 'orange', 'green'] # 色のリスト
plt.scatter(X, y, color='blue', label='data') # 散布図にデータをプロット(青)
for i in range(3):
```

```
    y_pred = poly.transform(x0) @ theta[i] # 回帰曲線の予測値
    plt.plot(x0, y_pred, label=f'alpha={alpha[i]}', color=color_list[i]) # 回帰曲
plt.legend()
plt.xlabel('x'); plt.ylabel('y')
plt.show()
```