ASSIGNMENT 1 - Regression (normal equation)

Matthew Dunne

Use Linear equation normal equation to predict water temperature T_degC
1) Only use 'Salnty', 'STheta' for predictors
2) Remove NaN / NA values from dataset (prior to building train/test sets).
3) Solve for rmse, variance explained, and r-squared.

In [1]:

```
import numpy as np
import os
import pandas as pd
from sklearn.model_selection import train_test_split
```

In [2]:

```
data=pd.read_csv('bottle.csv')
#use only 'Salnty', 'STheta' for predictors and T_degC as predictor
data=data[['T_degC','Salnty', 'STheta']]
#remove NaN/NA values
data=data.dropna()
X=data[['Salnty', 'STheta']]
Y=data['T_degC']
#split into training and test
X_train, X_test, y_train, y_test = train_test_split(X, Y, random_state=0)
```

```
C:\Users\mjdun\Anaconda\lib\site-packages\IPython\core\interactiveshell.py:2698: DtypeWarning:
Columns (47,73) have mixed types. Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

In [3]:

```
#remove NaN/NA values
data=data.dropna()
#split into input/output
X=data[['Salnty', 'STheta']]
Y=data['T_degC']
#split into training and test
X_train, X_test, y_train, y_test = train_test_split(X, Y, random_state=0)
```

In [4]:

```
#count the number of rows in the train and test data for the next step
train_length=len(X_train.index)
test_length=len(X_test.index)
# add x0 = 1 to each instance for both train and test predictors
train_b = np.c_[np.ones((train_length, 1)), X_train]
test_b = np.c_[np.ones((test_length, 1)), X_test]
```

In [5]:

```
#use the normal equation
theta_best = np.linalg.inv(train_b.T.dot(train_b)).dot(train_b.T).dot(y_train)
theta_best
```

Out[5]:

```
array([ 99.08024365,  -0.60203564,  -2.62812934])
```

In [6]:

```
#predicted values
y_predict = test_b.dot(theta_best)
```

```
y_predict[0:5]
```

Out[6]:

```
array([  9.04428417,  11.86422131,   9.53611653,   6.44390949,   7.32103302])
```

How well did the model work?

Root Mean Squared Error

In [7]:

```
from sklearn.metrics import mean_squared_error
from math import sqrt
rms = sqrt(mean_squared_error(y_test, y_predict))
rms
```

Out[7]:

```
1.7917771619463705
```

Variance Explained

In [8]:

```
from sklearn.metrics import explained_variance_score
explained_variance_score(y_test, y_predict)
```

Out[8]:

```
0.81943219874066209
```

R-Squared

In [9]:

```
from sklearn.metrics import r2_score
r2_score(y_test, y_predict)
```

Out[9]:

```
0.81943070540441232
```