Programming assignment 2: Linear regression

In [1]:
```python
import numpy as np

from sklearn.datasets import load_boston
from sklearn.model_selection import train_test_split
```

Your task

In this notebook code skeleton for performing linear regression is given. Your task is to complete the functions where required. You are only allowed to use built-in Python functions, as well as any `numpy` functions. No other libraries / imports are allowed.

Load and preprocess the data

I this assignment we will work with the Boston Housing Dataset. The data consists of 506 samples. Each sample represents a district in the city of Boston and has 13 features, such as crime rate or taxation level. The regression target is the median house price in the given district (in $1000's).

More details can be found here: http://lib.stat.cmu.edu/datasets/boston

In [2]:
```python
X , y = load_boston(return_X_y=True)

# Add a vector of ones to the data matrix to absorb the bias term
# (Recall slide #7 from the lecture)
X = np.hstack([np.ones([X.shape[0], 1]), X])
# From now on, D refers to the number of features in the AUGMENTED dataset
#  (i.e. including the dummy '1' feature for the absorbed bias term)

# Split into train and test
test_size = 0.2
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size)
```

Task 1: Fit standard linear regression

In [3]:
```python
def fit_least_squares(X, y):
    """Fit ordinary least squares model to the data.

    Parameters
    ----------
    X : array, shape [N, D]
        (Augmented) feature matrix.
    y : array, shape [N]
        Regression targets.

    Returns
    -------
    w : array, shape [D]
        Optimal regression coefficients (w[0] is the bias term).

    """
    # TODO
    return np.matmul(np.matmul(np.linalg.inv(np.matmul(np.transpose(X),X)),np.transpose(X)), y)
```

Task 2: Fit ridge regression

In [4]:
```python
def fit_ridge(X, y, reg_strength):
    """Fit ridge regression model to the data.

    Parameters
    ----------
    X : array, shape [N, D]
        (Augmented) feature matrix.
    y : array, shape [N]
        Regression targets.
    reg_strength : float
        L2 regularization strength (denoted by lambda in the lecture)

    Returns
    -------
    w : array, shape [D]
        Optimal regression coefficients (w[0] is the bias term).

    """
    # TODO
    return np.matmul(np.matmul(np.linalg.inv(np.matmul(np.transpose(X),X) + reg_strength*np.identity
(X.shape[1])),np.transpose(X)), y);
```

Task 3: Generate predictions for new data

In [5]:
```python
def predict_linear_model(X, w):
    """Generate predictions for the given samples.

    Parameters
    ----------
    X : array, shape [N, D]
        (Augmented) feature matrix.
    w : array, shape [D]
        Regression coefficients.

    Returns
    -------
    y_pred : array, shape [N]
        Predicted regression targets for the input data.

    """
    # TODO
    return np.matmul(X,w)
```

Task 4: Mean squared error

In [6]:
```python
def mean_squared_error(y_true, y_pred):
    """Compute mean squared error between true and predicted regression targets.

    Reference: `https://en.wikipedia.org/wiki/Mean_squared_error`

    Parameters
    ----------
    y_true : array
        True regression targets.
    y_pred : array
        Predicted regression targets.

    Returns
    -------
    mse : float
        Mean squared error.

    """
    # TODO
    err = y_true-y_pred;
    return (np.dot(err,err))/(y_true.shape[0]);
```

Compare the two models

The reference implementation produces

- MSE for Least squares $\approx$ **23.98**
- MSE for Ridge regression $\approx$ **21.05**

You results might be slightly (i.e. $\pm 1\%$) different from the reference soultion due to numerical reasons.

In [7]:
```python
# Load the data
np.random.seed(1234)
X , y = load_boston(return_X_y=True)
X = np.hstack([np.ones([X.shape[0], 1]), X])
test_size = 0.2
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size)

# Ordinary least squares regression
w_ls = fit_least_squares(X_train, y_train)
y_pred_ls = predict_linear_model(X_test, w_ls)
mse_ls = mean_squared_error(y_test, y_pred_ls)
print('MSE for Least squares = {0}'.format(mse_ls))

# Ridge regression
reg_strength = 1
w_ridge = fit_ridge(X_train, y_train, reg_strength)
y_pred_ridge = predict_linear_model(X_test, w_ridge)
mse_ridge = mean_squared_error(y_test, y_pred_ridge)
print('MSE for Ridge regression = {0}'.format(mse_ridge))
```

```
MSE for Least squares = 23.98430761177853
MSE for Ridge regression = 21.051487033771117
```

MACHINE LEARNING HOMEWORK SHEET 03
LINEAR REGRESSION

NAME: MAHALAKSHMI SABANAYAGAM

TUM ID: ge73yuw
DATE: 11.11.2018

Problem 2:

$$E_{weighted}(w) = \frac{1}{2}\sum_{i=1}^{N} t_i \left[w^T \phi(x_i) - y_i\right]^2$$

Let there be N data points and m features,

$w \rightarrow m \times 1$

$\phi(x) \rightarrow N \times m$

$y \rightarrow N \times 1$

Let T be diagonal matrix with $t_1, t_2 \ldots t_N$ as diagonal values. $T \rightarrow N \times N$

$$E_{weighted}(w) = \frac{1}{2}\sum_{i=1}^{N} t_i (\phi(x_i) w - y_i)^2$$

This can be written in matrix vector form,

$$E_{weighted}(w) = \frac{1}{2}\left[(\phi(x)w - y)^T T (\phi(x) w - y)\right]$$

$$= \frac{1}{2}\left[(w^T \phi^T(x) - y^T) T (\phi(x) w - y)\right]$$

$$= \frac{1}{2}\left[w^T \phi(x)^T T \phi(x) w - y^T T \phi(x) w - w^T \phi^T(x) T y + y^T T y\right]$$

$$(w^T \phi^T(x) T y)^T = y^T T^T \phi(x) w$$

$$= y^T T \phi(x) w$$

(since $\underline{T^T = T}$ as T is a diagonal matrix.

$$= \frac{1}{2}\left[w^T \phi^T(x) T \phi(x) w - 2 y^T T \phi(x) w + y^T T y\right]$$

to find w that minimizes $E_{weighted}(w)$, $\nabla_w E_{weighted}(w) = 0$

$$\therefore \nabla_w E_{weighted}(w) = \frac{1}{2}\left[w^T \phi^T(x) T \phi(x) + w^T \phi^T(x) T \phi(x) - 2 y^T T \phi(x)\right] = 0$$

$$\Rightarrow \cancel{2} w^T \phi^T(x) T \phi(x) - \cancel{2} y^T T \phi(x) = 0.$$

$$w^T \phi^T(x) T \phi(x) = y^T T \phi(x)$$

Taking Transpose on both the sides,

$$\phi^T(x)\, T\, \phi(x)\, W = \phi^T(x)\, T\, Y$$

$$W_{WLS} = (\phi^T(x)\, T\, \phi(x))^{-1} \phi^T(x)\, T\, Y \rightarrow (1)$$

For this W the error function is minimum.

1. weighting factor t_i in terms of variance of noise on the data:
Lets assume noise is normally distributed $N(0,\sigma^2)$

$\Rightarrow$ $y_i = \phi(x_i) W + \epsilon_i$ ϵ is noise

Substituting y in W from (1),

$$W_{WLS} = (\phi^T(x)\, T\, \phi(x))^{-1} \phi^T(x)\, T\, (\phi(x) W + \epsilon)$$

$$= (\phi^T(x)\, T\, \phi(x))^{-1} \underline{\phi^T(x)\, T\, \phi(x)} W + (\phi^T(x)\, T\, \phi(x))^{-1} \phi^T(x)\, T\, \epsilon$$

$$= I W + (\phi^T(x)\, T\, \phi(x))^{-1} \phi^T(x)\, T\, \epsilon$$

variance of W_{WLS} should be small for unbiased estimator

$$\text{var}[W_{WLS}]_j = \text{var}[W_j] + \sum_{i=1}^{N} [(\phi^T(x)\, T\, \phi(x))^{-1} \phi^T(x)\, T]_{ji} \sigma_i^2$$

$$= \sum_{i=1}^{N} [(\phi^T(x)\, T\, \phi(x))^{-1} \phi^T(x)\, T]_{ji}\, \sigma_i^2$$

For this to be small, weight T_i should be inversely proportional to σ_i^2: $T_i \propto 1/\sigma_i^2$ (given noise variance).

Thus, the weight should be large around the data points with less noise for it to be a good estimator. (unbiased)

2. data points for which there are exact copies in the dataset.
In case of biased sample where many copies of some ~~sample~~ data points exists in the sample, the weight for those samples should be ~~minimum~~ less to avoid & high bias.

If p_i is the probability of observing i^{th} sample then weight of i should be proportional to $1/p_i$. $w_i \propto 1/p_i$ so that sampling bias is minimized.

Problem 3:

The ridge regression estimates can be obtained by ordinary least squares regression on an augmented dataset.

Design matrix $\phi \in R^{N \times M}$

$y \to N \times 1$ vector

with ϕ & y, w estimate of ordinary Least Squares is

$$w = (\phi^T \phi)^{-1} \phi^T y \to (1)$$

Augmenting M additional rows with ϕ,

$$\phi_R = \begin{bmatrix} \phi \\ \sqrt{\lambda} I \end{bmatrix}_{(N+M) \times M}$$

Augmenting y with M zeros,

$$y_R = \begin{bmatrix} y \\ 0 \end{bmatrix}_{(N+M) \times 1}$$

Substituting ϕ_R & y_R in (1),

$$w = \left(\begin{bmatrix} \phi \\ \sqrt{\lambda} I \end{bmatrix}^T \begin{bmatrix} \phi \\ \sqrt{\lambda} I \end{bmatrix} \right)^{-1} \begin{bmatrix} \phi \\ \sqrt{\lambda} I \end{bmatrix}^T \begin{bmatrix} y \\ 0 \end{bmatrix}$$

$I^T = I$

$$= \left(\begin{bmatrix} \phi^T & \sqrt{\lambda} I \end{bmatrix} \begin{bmatrix} \phi \\ \sqrt{\lambda} I \end{bmatrix} \right)^{-1} \begin{bmatrix} \phi^T & \sqrt{\lambda} I \end{bmatrix} \begin{bmatrix} y \\ 0 \end{bmatrix}$$

$$= (\phi^T \phi + \lambda I^2)^{-1} \phi^T y \quad ; \quad I^2 = I$$

$$= (\phi^T \phi + \lambda I)^{-1} \phi^T y$$

$=$ ridge regression estimate.

Thus Ridge regression estimate is obtained from OLS regression estimate on augmented dataset by augmenting $\phi_{N \times M}$ with M rows of $\sqrt{\lambda} I$ and y with M zeros.

Problem 4:

Likelihood $p(y \mid \phi, w, \beta) = \prod_{i=1}^{N} \mathcal{N}(y_i \mid w^T \phi(x_i), \beta^{-1})$

prior $p(w, \beta) = \mathcal{N}(w \mid m_0, \frac{S_0}{\beta})\, \text{Gamma}(\beta \mid a_0, b_0)$

To prove:

$$p(w, \beta \mid D) = \mathcal{N}(w \mid m_N, \frac{S_N}{\beta})\, \text{Gamma}(\beta \mid a_N, b_N).$$

$$P(Y \mid \phi, w, \beta) = \prod_{i=1}^{N} \mathcal{N}(y_i \mid w^T \phi(x_i), \beta^{-1}) \qquad \sigma^2 = 1/\beta$$

(Likelihood)

$$= \prod_{i=1}^{N} \left(\frac{\sqrt{\beta}}{\sqrt{2\pi}}\right) e^{-\frac{\beta}{2}(y_i - w^T\phi(x_i))^2}$$

$$= \left(\sqrt{\frac{\beta}{2\pi}}\right)^N e^{-\frac{\beta}{2}\sum_{i=1}^{N}(y_i - w^T\phi(x_i))^2}$$

$$\propto (\sqrt{\beta})^N e^{-\frac{\beta}{2}(Y - Xw)^T(Y - Xw)}$$

$$P(w, \beta) = \mathcal{N}(w \mid m_0, \frac{S_0}{\beta})\, \text{Gamma}(\beta \mid a_0, b_0)$$

(Prior)

$$= \frac{\sqrt{\beta}}{\sqrt{2\pi}\sqrt{S_0}} e^{-\frac{(w - m_0)^2 \beta}{2S_0}} \cdot \frac{b_0^{a_0} \beta^{a_0 - 1} e^{-b_0\beta}}{\Gamma(a_0)}$$

$$\propto \frac{\sqrt{\beta}}{\sqrt{S_0}} \frac{b_0^{a_0} \beta^{a_0-1}}{\Gamma(a_0)} \cdot e^{-\beta\left[\frac{(w-m_0)^2}{2S_0} + b_0\right]}$$

$$p(Y \mid \phi, w, \beta) \cdot p(w, \beta) = (\sqrt{\beta})^{N+1} \frac{1}{\sqrt{S_0}} \frac{b_0^{a_0}\beta^{a_0-1}}{\Gamma(a_0)} e^{-\beta\left[\frac{1}{2}\left((Y - Xw)^T(Y - Xw) + \frac{(w-m_0)^2}{S_0}\right) + b_0\right]} \rightarrow (1)$$

Likelihood * Prior

$$p(w, \beta \mid D) = \frac{\sqrt{\beta}}{\sqrt{2\pi}\sqrt{S_N}} e^{-\frac{(w - m_N)^2 \beta}{2S_N}} \frac{b_N^{a_N} \beta^{a_N - 1} e^{-b_N\beta}}{\Gamma(a_N)}$$

posterior

$$\propto \frac{\sqrt{\beta}}{\sqrt{S_N}} \frac{b_N^{a_N} \beta^{a_N-1}}{\Gamma(a_N)} e^{-\beta\left(\frac{(w-m_N)^2}{2S_N} + b_N\right)} \rightarrow (2)$$

Comparing exponents in 1 and 2,

$$\frac{1}{2}\left[(Y-XW)^T(Y-XW)+\frac{(W-m_0)^2}{S_0}\right]+b_0 = \frac{(W-m_N)^2}{2S_N}+b_N \rightarrow (3)$$

Comparing the constants,

$$(\sqrt{\beta})^{N+\cancel{1}} \cdot \frac{1}{\sqrt{S_0}} \cdot \frac{b_0^{a_0}\,\beta^{a_0-1}}{\Gamma(a_0)} = \cancel{\sqrt{\beta}} \cdot \frac{1}{\sqrt{S_N}} \frac{b_N^{a_N}\,\beta^{a_N-1}}{\Gamma(a_N)}$$

$$\frac{\beta^{a_0-1+\frac{N}{2}} \cdot b_0^{a_0}}{\sqrt{S_0}\cdot\Gamma(a_0)} = \frac{\beta^{a_N-1}\, b_N^{a_N}}{\sqrt{S_N}\,\Gamma(a_N)} \rightarrow (4)$$

Comparing β exponents,

$$a_0 - \cancel{1} + \frac{N}{2} = a_N - \cancel{1}$$

$$\boxed{a_N = a_0 + \frac{N}{2}}$$

So, (4) will become

$$\frac{b_0^{a_0}}{\sqrt{S_0}\,\Gamma(a_0)} = \frac{b_N^{a_N}}{\sqrt{S_N}\,\Gamma(a_N)}$$