2. Linear Regression

March 28, 2022

[1]:
```python
import pandas as pd
import numpy as np
from sklearn.metrics import mean_squared_error

from sklearn.datasets import fetch_california_housing
X, y = fetch_california_housing(return_X_y=True)


from sklearn.preprocessing import StandardScaler
sc = StandardScaler().fit(X)
X = sc.transform(X)

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2)
```

[2]:
```python
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X_train,y_train)
y_pred = lr.predict(X_test)
mean_squared_error(y_test, y_pred)
```

[2]: 0.5078383014248863

[3]:
```python
# X.shape[1] calculates the number of features in X_train
D = X_train.shape[1]
N = len(X_train)              # total number of rows in the dataset
weights = np.random.rand(D)   # randomly initialized weight vector
t = y_train                   # target vector
eta = 0.05                    # learning rate
for _ in range(100):
    y = np.dot(X_train,weights)
    error = y - t
    # calculation of OLS loss, error.T will generate transpose
    cost = np.dot(error.T,error)/(2*N)
    # w_j = w_j -  [(Σ[(y-t)(x_j)])/N]
    weights = weights - eta * np.dot(X_train.T,error)/N
```

```python
y_pred = np.dot(X_test,weights)
mean_squared_error(y_test, y_pred)
```

[3]: 4.906192186434918

```python
[ ]:
```