Homework 3

Qianlang Chen

Problem 1

```
import numpy

# read in the data for this problem
x = numpy.genfromtxt("data/x.csv")
y = numpy.genfromtxt("data/y.csv")
print(len(x), len(y))
```

```
## 100 100
```

Part (a)

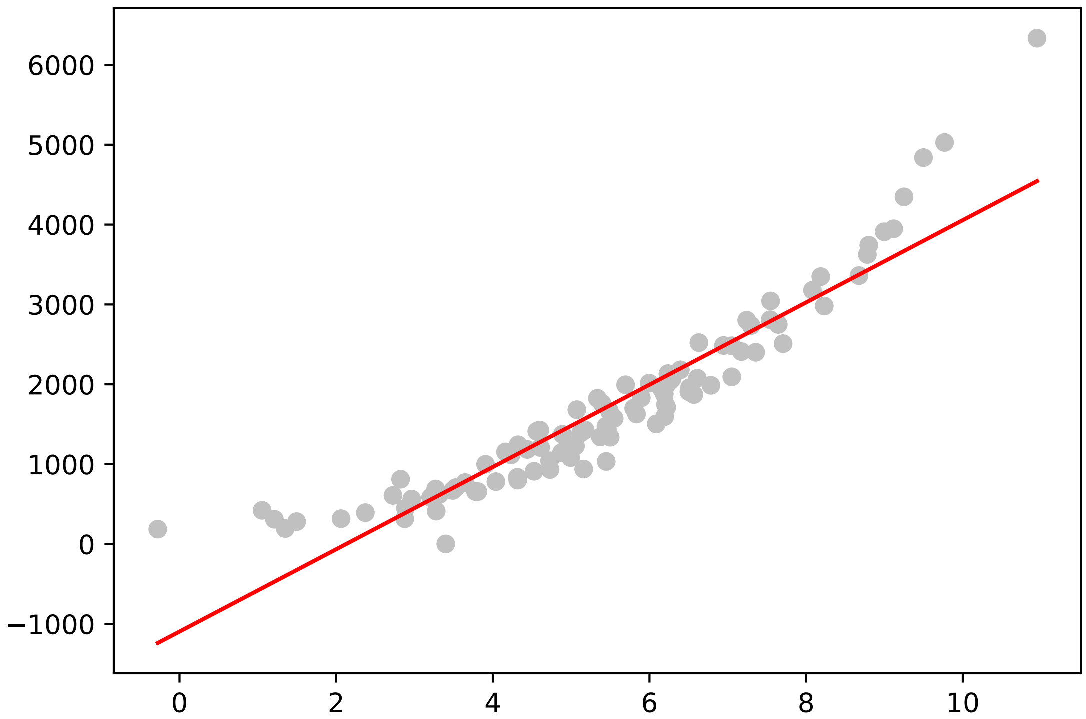
With the help of Python, let's find out the parameters of the regression line:

```
# line equation: y_hat = a*x + b
(a, b) = numpy.polyfit(x, y, 1)
print(a, b)
```

```
## 515.1222963490035 -1095.6026286820393
```

```
# plot the data and the regression line
from matplotlib import pyplot
pyplot.scatter(x, y, c="#C0C0C0")

line_x = numpy.linspace(min(x), max(x), 1729)
line_y = a * line_x + b
pyplot.plot(line_x, line_y, "r-", zorder=1729)
```



According to the output, the best-fit line for this data is approximately

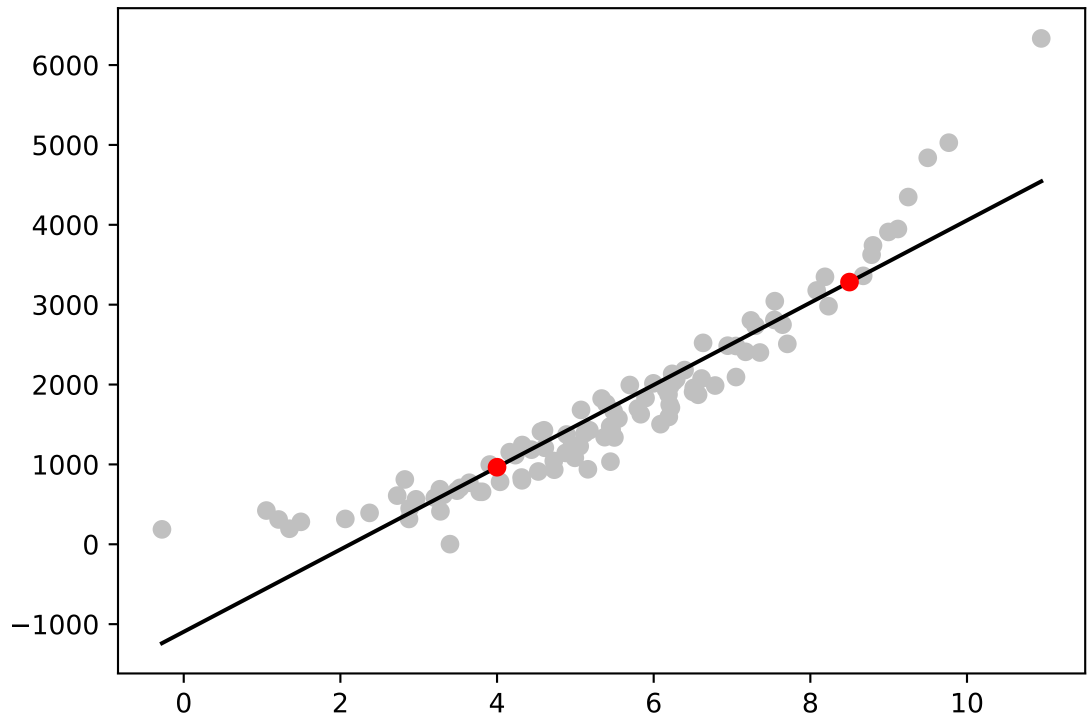
$$\hat{y} = M(x) = 515.1x - 1096$$

Using this model, we can then predict the values of y for $x = 4$ and $x = 8.5$:

```
# calculate and plot the predictions
pred_x = numpy.array([4, 8.5])
pred_y = a * pred_x + b
print(pred_x, pred_y)
```

```
## [4.  8.5] [ 964.88655671 3282.93689028]
```

```
pyplot.scatter(x, y, c="#C0C0C0")
pyplot.plot(line_x, line_y, "k-")
pyplot.scatter(pred_x, pred_y, c="r", zorder=1729)
```



The output indicates that our predictions for inputs 4 and 8.5 are about 964.9 and 3283, respectively, and as seen from the plot, they do fit the data arguably nicely.

Part (b)

```
# split to get the training data and perform linear regression on it
train_len = int(.80 * len(x))
train_x = x[:train_len]
train_y = y[:train_len]

(train_a, train_b) = numpy.polyfit(train_x, train_y, 1)
print(train_len, train_a, train_b)
```

```
## 80 492.8393809405474 -991.1378015106925
```

The best-fit model for the training data is approximately

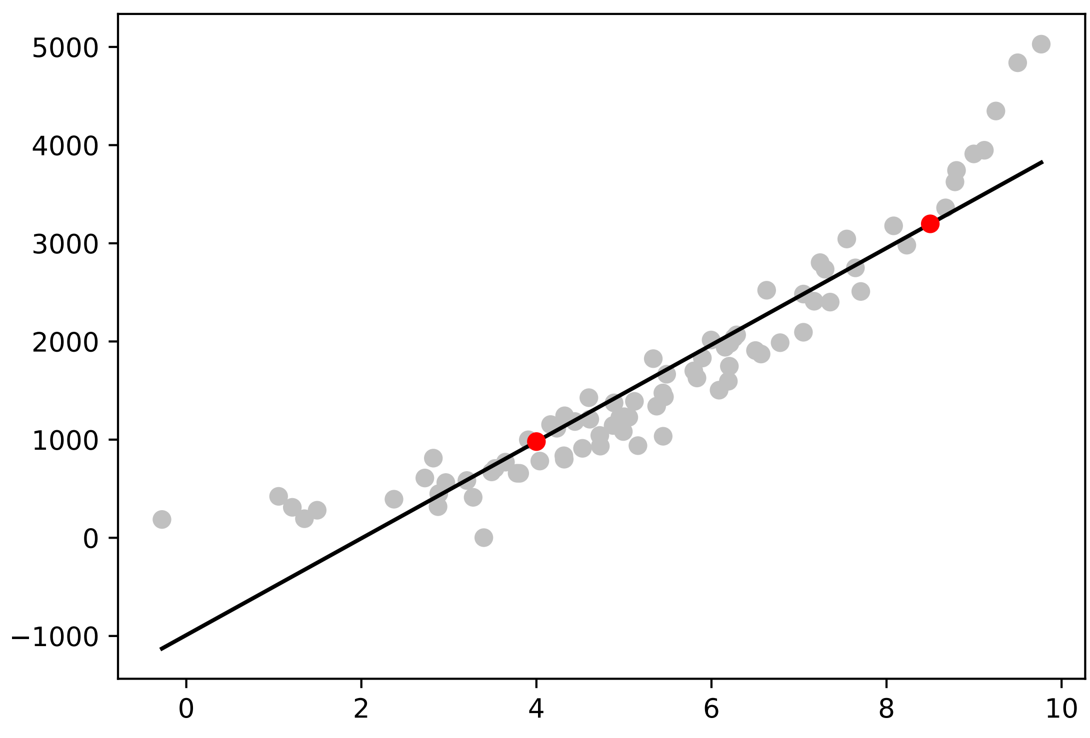
$$\hat{y} = M(x) = 492.8x - 991.1$$

```
# calculate and plot the predictions
pred_x = numpy.array([4, 8.5])
pred_y = train_a * pred_x + train_b
print(pred_x, pred_y)
```

```
## [4.  8.5] [ 980.21972225 3197.99693648]
```

```
line_x = numpy.linspace(min(train_x), max(train_x), 1729)
line_y = train_a * line_x + train_b

pyplot.scatter(train_x, train_y, c="#C0C0C0")
pyplot.plot(line_x, line_y, "k-")
pyplot.scatter(pred_x, pred_y, c="r", zorder=1729)
```



Our predictions this time were about 980.2 and 3198, which ain't too bad either.

Part (c)

```
# split to get the testing data
test_x = x[train_len:]
test_data_y = y[train_len:]

# run the tests and calculate the residuals
test_full_y = a * test_x + b
test_train_y = train_a * test_x + train_b

res_full = test_full_y - test_data_y
res_train = test_train_y - test_data_y
print("Residual of the testing data using model built from the full data:",
  res_full,
  "L2-norm of this residual vector: "+str(numpy.linalg.norm(res_full, 2)),
  "",
  "Residual of the testing data using model built from the training data:",
  res_train,
  "L2-norm of this residual vector: "+str(numpy.linalg.norm(res_train, 2)),
  sep="\n")
```

```
## Residual of the testing data using model built from the full data:
## [  215.89690803   -17.36418609   -20.12249288  -100.18110103
##     -6.71460716  -350.63745562   234.75647375 -1789.87014063
##    144.22127685   -78.39281636  -227.1982354     17.23207118
##    399.06602318    -4.66010069  -165.00458314   300.50641249
##   -156.03643754  -157.72442192   397.35006545   189.12760159]
## L2-norm of this residual vector: 2009.387840861227
##
## Residual of the testing data using model built from the training data:
## [  182.45148582   -51.85145805   -83.69546569   -68.59127319
##     23.87084202  -292.13940793   191.89664509 -1929.35492764
##    133.28875434   -94.12670145  -305.15792262   -20.78470372
##    381.01044539   -54.94514931  -173.56777438   259.8903937
##   -178.47588917  -154.88998906   363.21463011   169.93289968]
## L2-norm of this residual vector: 2115.5423038301
```

```
# calculate the residuals for the models built on the full data and the
#   training data
train_full_y = a * x + b
train_train_y = train_a * train_x + train_b

res_train_full = train_full_y - y
res_train_train = train_train_y - train_y
print("L2-norm of the residual vector of the model built from the full data:",
  numpy.linalg.norm(res_train_full, 2),
  "",
  "L2-norm of the residual vector of the model built from the training data:",
  numpy.linalg.norm(res_train_train, 2),
  sep="\n")
```

```
## L2-norm of the residual vector of the model built from the full data:
## 4071.9265456415037
##
## L2-norm of the residual vector of the model built from the training data:
## 3513.93821745281
```

Part (d)

```
# create tilde-x for a degree-3 polynomial regression
tilde_x = numpy.matrix([[x[i]**j for j in range(4)] for i in range(train_len)])
print("First three row of tilde-X:", tilde_x[:3], sep="\n")
```

```
## First three row of tilde-X:
## [[  1.           3.64641412  13.29633592  48.483947  ]
##  [  1.           9.76756879  95.40540001 931.8788072 ]
##  [  1.           2.72543008   7.42796911  20.24441042]]
```

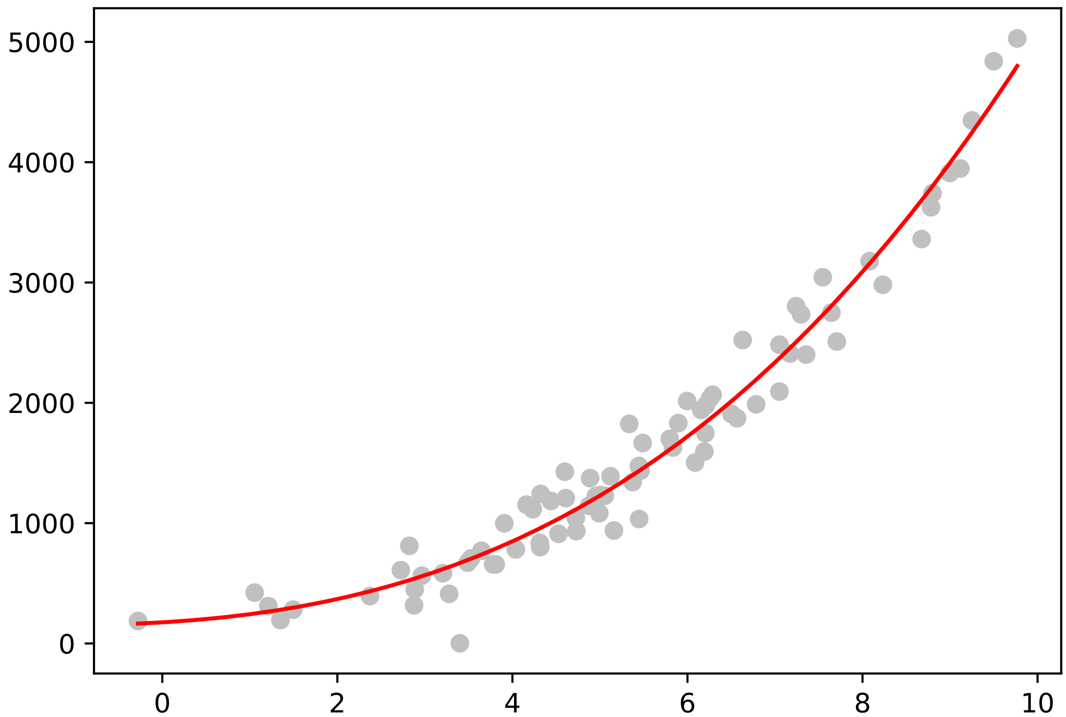
```
# perform the polynomial regression and calculate the best-bit curve
coeffs = (tilde_x.T * tilde_x).I * tilde_x.T * numpy.matrix([train_y]).T
poly = numpy.poly1d(numpy.flip(numpy.squeeze(numpy.asarray(coeffs))))

print(poly)
```

```
##         3          2
## 2.209 x + 22.49 x + 42.99 x + 175.4
```

```
# plot the best fit line
line_x = numpy.linspace(min(train_x), max(train_x), 1729)
line_y = poly(line_x)

pyplot.scatter(train_x, train_y, c="#C0C0C0")
pyplot.plot(line_x, line_y, "r-")
```



The best-fit degree-3 polynomial model for the training data is

$$\hat{y} = M_3(x) = 2.209x^3 + 22.49x^2 + 42.99x + 175.4$$

```
# calculate the residuals
res_test = poly(test_x) - test_data_y
res_train = poly(train_x) - train_y
```

```python
print("L2-norm of the residual vector of the testing data:",
  numpy.linalg.norm(res_test, 2),
  "L2-norm of the residual vector of the training data:",
  numpy.linalg.norm(res_train, 2),
  sep="\n")
```

```
## L2-norm of the residual vector of the testing data:
## 942.2002113309505
## L2-norm of the residual vector of the training data:
## 1831.546668230139
```

Problem 2

Part (a)

Part (b)

Part (c)

Part (d)

Problem 3

Part (a)

Part (b)

Part (c)

Part (d)