Question 1: Linear Regression

Data

In the current problem we are given m values acidity of wine such that for each i^{th} value of the acidity ($\alpha^i \varepsilon R^1$) we have $Y^i \varepsilon R^1$ which is density of wine. We define X^i such that for each i^{th} sample X^i $=< 1, \alpha^i >$ to accomodate intercept term where $X \varepsilon R^{1+1}$.

Equations used in training of model parameters: θ ($\theta \varepsilon R^{1+1}$)

Model : $h_\theta(X) = \theta^T X$

Error: $J(\theta) = \frac{1}{2} \sum_{i=1}^{m} \left(Y^i - \theta^T X^i\right)^2$

Gradient: $\bigtriangledown_\theta(J(\theta)) = -1 * \sum_{i=1}^{m} X^i(Y^i - \theta^T X^i)$

GD algorithm: $\theta^{t+1} = \theta^t - \eta \bigtriangledown_\theta (J(\theta))$

In [23]:

```
# importing necessary header files
import numpy as np
import matplotlib.pyplot as plt
import sys
sys.path.append('../')
from lib.ml import linear
```

executed in 5ms, finished 22:55:58 2018-02-12

Functions to plot model estimates

In [24]:

```
def _plot_model(self, X, Y):
    plot_eqn = 'self.theta[0,1]*X+self.theta[0,0]'
    plt.figure(1)
    plt.title("Model for learning rate %f" % (self.lr), fontsize=10, y=0.9)
    Y_predicted = eval(plot_eqn)
    plt.plot(X, Y, 'b.', label='Actual Data')
    plt.plot(X, Y_predicted, 'r-', label='model')
    plt.legend(bbox_to_anchor=(0.7, 0.2), loc=2, borderaxespad=0.)
    plt.xlabel(r'$X$')
    plt.ylabel(r'$Y$')
    plt.draw()


def _plot_error(self):
    history = np.asarray(self.history)
    t_hist = history.shape[0]
    x = (history[:, :, 0].T)[0]
    y = (history[:, :, 1].T)[0]
    z = (history[:, :, 2].T)[0]

    plt.figure(2)
    sp1 = plt.subplot(111, projection='3d')
    sp1.set_title('Gradient Descend for learning rate %f' % (self.lr))

    alpha = [(t_hist - x) / t_hist for x in range(0, t_hist)]
    sp1.plot(x, y, z, c='b', marker='')
    sp1.scatter(x, y, z, c=alpha, marker='o', s=40., cmap='gray')
    sp1.set_xlabel(r'$\theta_{0}$')
    sp1.set_ylabel(r'$\theta_{0}$')
    sp1.set_zlabel(r'$J(\theta)$')
    plt.gca().invert_xaxis()
    plt.draw()


def _plot_contours(self, X_val,Y_val,cmap='gray',mesh=False):
    history = np.asarray(self.history)
    t_hist = history.shape[0]
    x = (history[:, :, 0].T)[0]
    y = (history[:, :, 1].T)[0]

    x_temp = np.arange(min(history[:, :, 0].T[0]), max(history[:, :, 0].T[0]),
0.0025)
    y_temp = np.arange(min(history[:, :, 1].T[0]), max(history[:, :, 1].T[0]),
0.0025)

    X, Y = np.meshgrid(x_temp, y_temp)
    Z = np.ones(X.shape)
    (ith,jth) = X.shape
    for i in range(0,ith):
        for j in range(0,jth):
            Z[i,j] = np.mean(0.5*(Y_val - Y[i,j]*X_val-X[i,j])**2)
    if mesh==False:
        plt.figure(3)
        plt.title('Contours for learning rate %f' % (self.lr))
        alpha = [(t_hist - x) / t_hist for x in range(0, t_hist)]
        plt.scatter(x, y, c=alpha, marker='o', cmap='gray')
        plt.plot(x, y, c='k', marker='')
        plt.contour(X, Y, Z)
        plt.xlabel(r'$\theta_{0}$')
```

```
        plt.ylabel(r'$\theta_{1}$')
        plt.draw()
    if mesh:
        plt.figure(4)
        sp = plt.subplot(111, projection='3d')
        sp.set_title(r'Mesh grid for $J(\theta)$')
        sp.plot_surface(X, Y, Z)
        sp.set_xlabel(r'$\theta_0$')
        sp.set_ylabel(r'$\theta_1$')
        sp.set_zlabel(r'$J(\theta)$')
        plt.draw()
```

executed in 516ms, finished 22:55:59 2018-02-12

Loading training and testing data

In [25]:

```
X = (np.loadtxt(open('linearX.csv'), delimiter=",")).reshape(-1, 1)
Y = (np.loadtxt(open('linearY.csv'), delimiter=",")).reshape(-1, 1)
```

executed in 325ms, finished 22:55:59 2018-02-12

Writing equation to compute θ

Note: to see how gardient descend is working use the interactive mode by replacing

```
model._train_(lr=0.0001,b_ratio=1,iter=20000,thresh=1e-100)
```

with

```
model._train_(lr=0.0001,b_ratio=1,iter=20000,thresh=1e-100,flag=True)
```

In [26]:

```
eqn = 'np.dot(X, theta.T)/100'
error = 'np.asarray(0.5 * np.sum((Y - np.dot(X, theta.T))**2)).reshape(1,1)/100'
GD = '-1*np.dot((Y - np.dot(X, theta.T)).T,X)/100'

model = linear(X, Y, 0.8, eqn, GD, error)
theta = model._train_(
    lr=0.01, b_ratio=1, iter=20000, thresh=1e-100, flag=False)
```

executed in 6.34s, finished 22:56:06 2018-02-12

$y =$ ***np.dot(X, theta.T)/100***

$J(\theta) =$ ***np.asarray(0.5 np.sum((Y - np.dot(X, theta.T))*2)).reshape(1,1)/100***

$\nabla_\theta J(\theta) =$ ***-1*np.dot((Y - np.dot(X, theta.T)).T,X)/100***

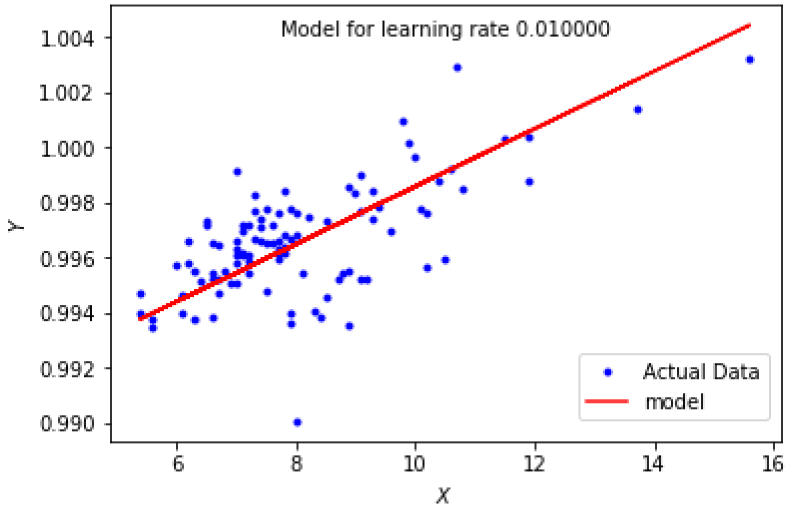
Estimated value of $\theta =$ [0.98812325 0.00104611]

Functions to plot model parameters

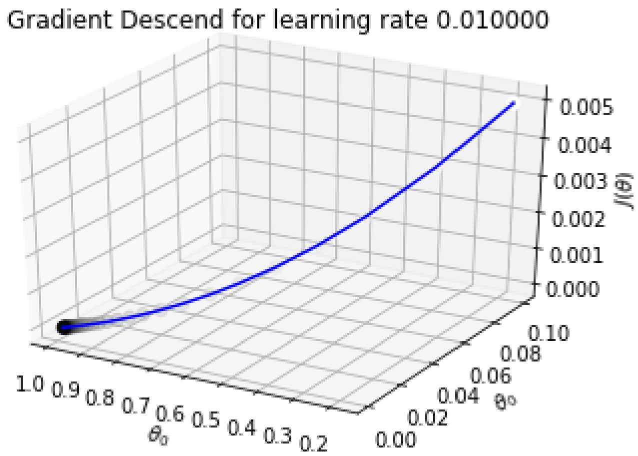
- _plot_model(model,X,Y)
- _plot_error(model)
- _plot_contours(model)

Output graphs for the model

Plotting model with data

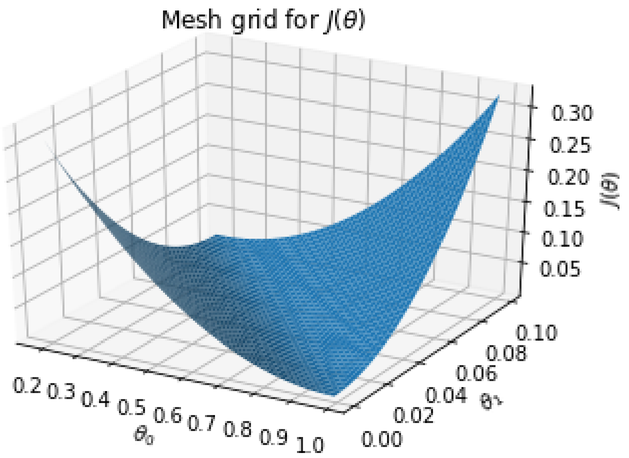


Variation of $J(\theta)$ with changing θ



(As epoch number increases shade becomes darker)

Contours for θ



(As epoch number increases shade becomes darker)

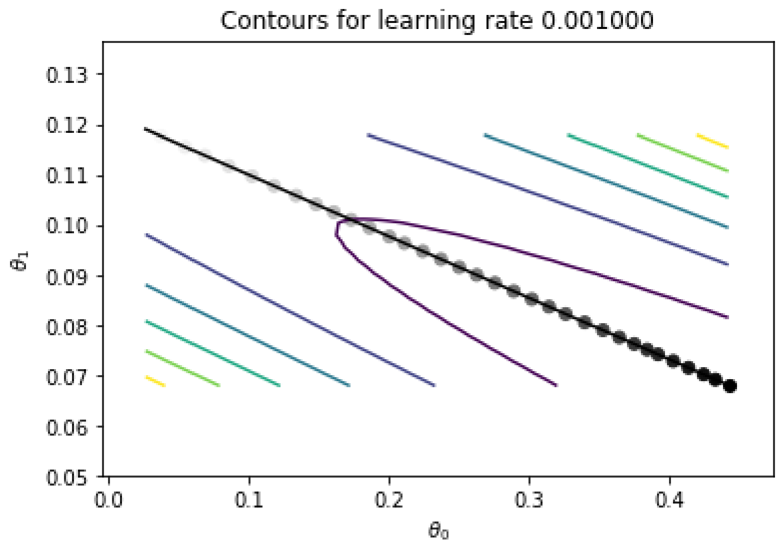
Plotting contour curve for different learning rate

In [27]:

```
theta = model._train_(lr=0.001, b_ratio=1, iter=20000, thresh=1e-100,
flag=False)
```

executed in 6.85s, finished 22:56:20 2018-02-12

Contours for θ

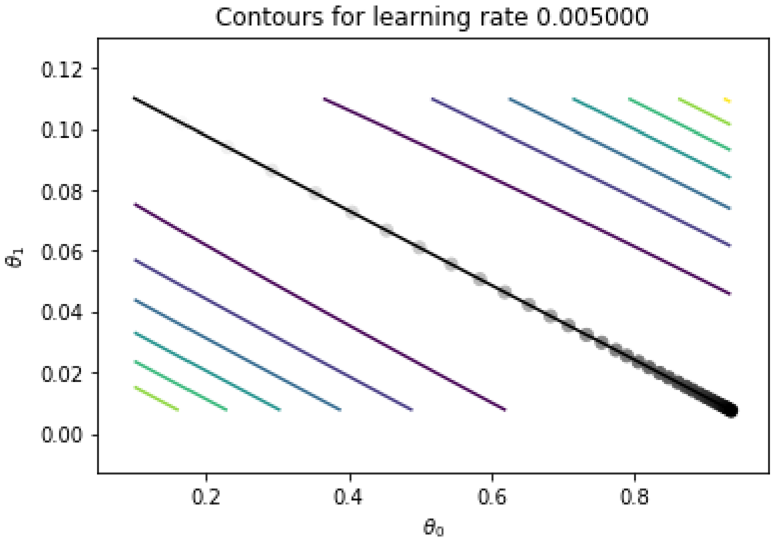


In [28]:

```
theta = model._train_(lr=0.005, b_ratio=1, iter=20000, thresh=1e-100,
flag=False)
```

executed in 6.71s, finished 22:56:29 2018-02-12

Contours for θ

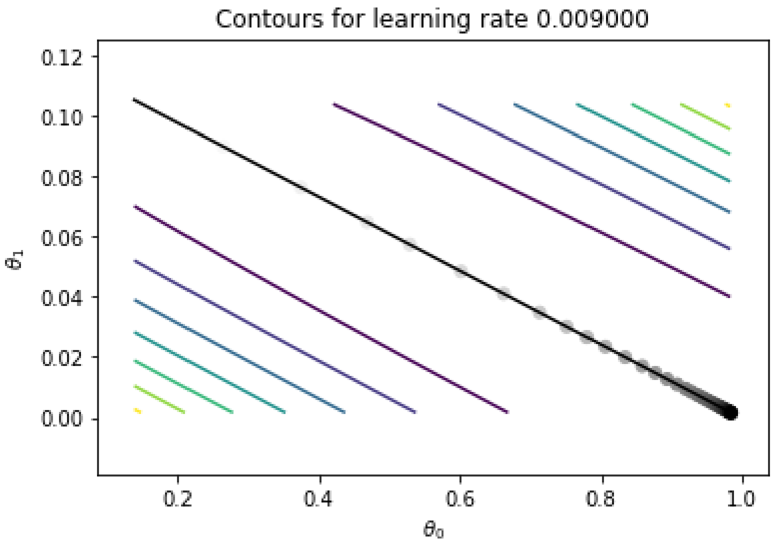


In [29]:

```
theta = model._train_(lr=0.009, b_ratio=1, iter=20000, thresh=1e-100,
flag=False)
```

executed in 6.79s, finished 22:56:38 2018-02-12

Contours for θ

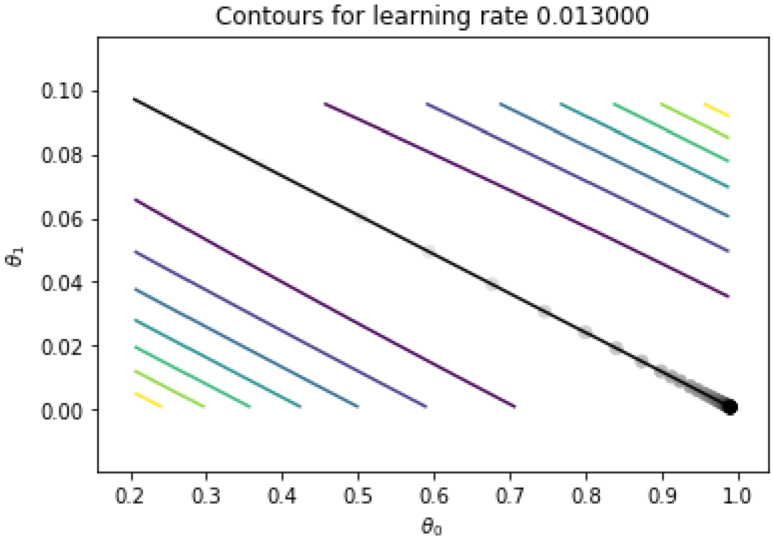


In [30]:

```
theta = model._train_(lr=0.013, b_ratio=1, iter=20000, thresh=1e-100,
flag=False)
```

executed in 6.59s, finished 22:56:47 2018-02-12

Contours for θ

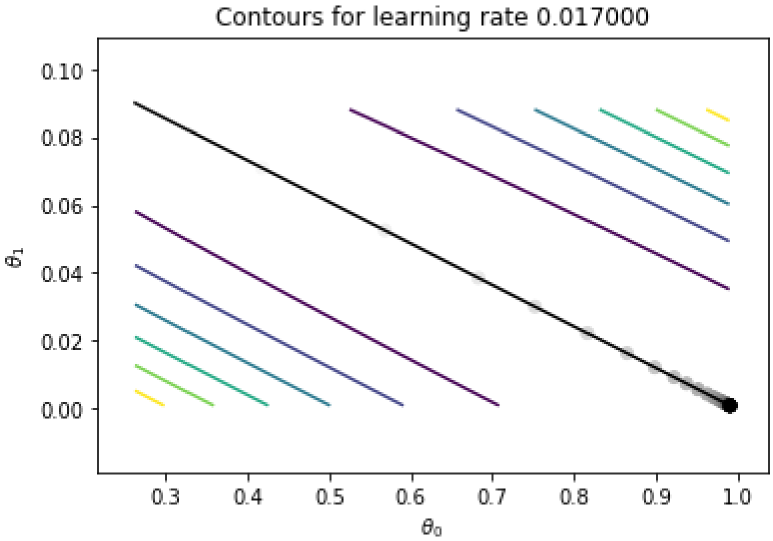


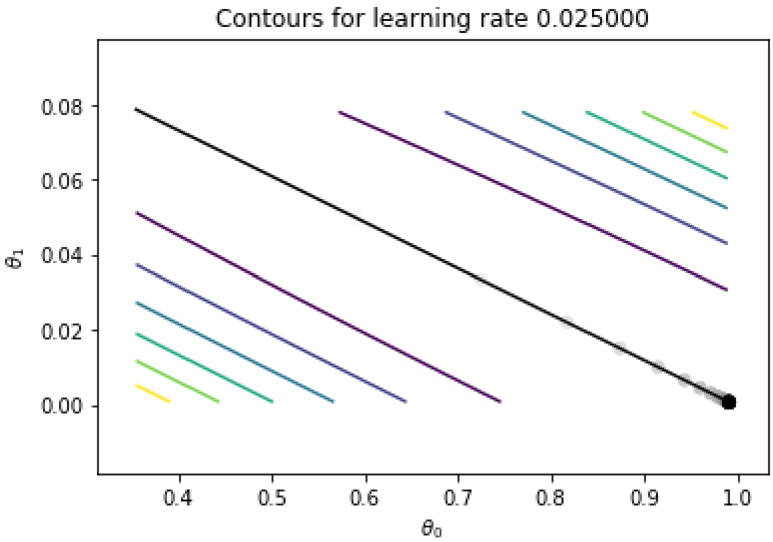
In [31]:

```
theta = model._train_(lr=0.017, b_ratio=1, iter=20000, thresh=1e-100,
flag=False)
```

executed in 6.53s, finished 22:56:56 2018-02-12

Contours for θ



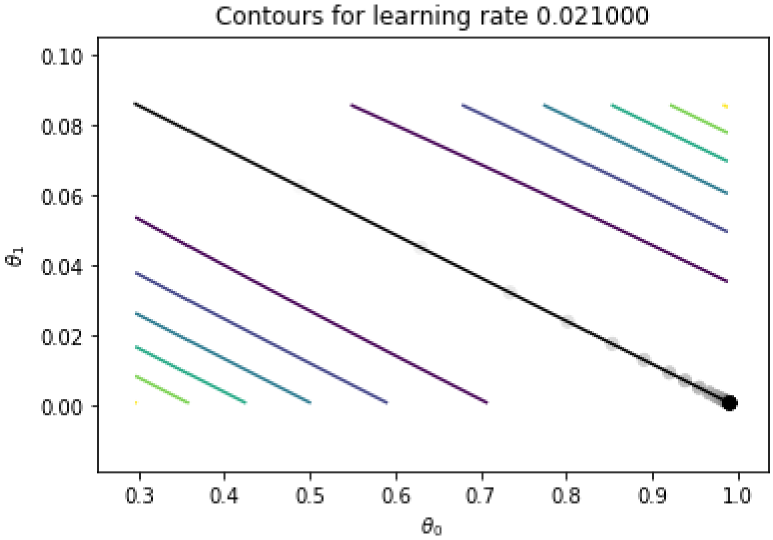


In [32]:

```
theta = model._train_(lr=0.021, b_ratio=1, iter=20000, thresh=1e-100,
flag=False)
```

executed in 6.64s, finished 22:57:05 2018-02-12

Contours for θ



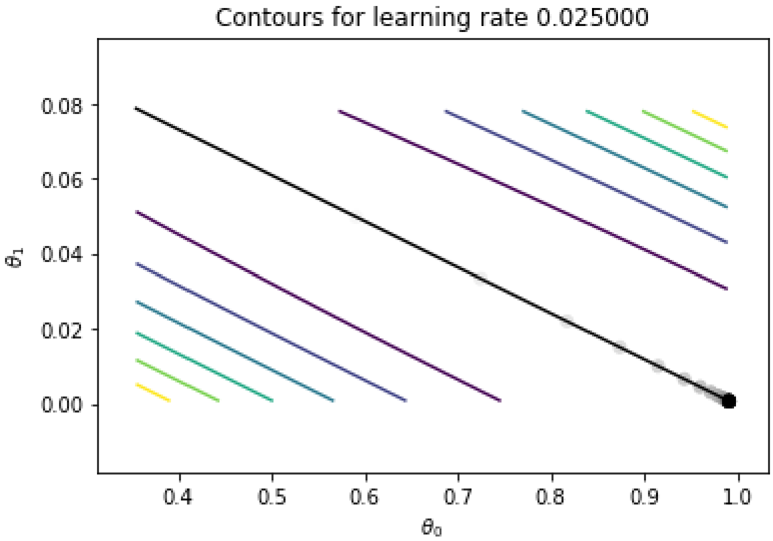
¶

In [33]:

```
theta = model._train_(lr=0.025, b_ratio=1, iter=20000, thresh=1e-100,
flag=False)
```

executed in 6.08s, finished 22:57:13 2018-02-12

Contours for θ



Notice that as the learning rate increases model converges faster towards optima