Exercise 4.2

November 15, 2017

0.0.1 Exercise H4.2 - Assignment 4

Group PeterPan

Team members:

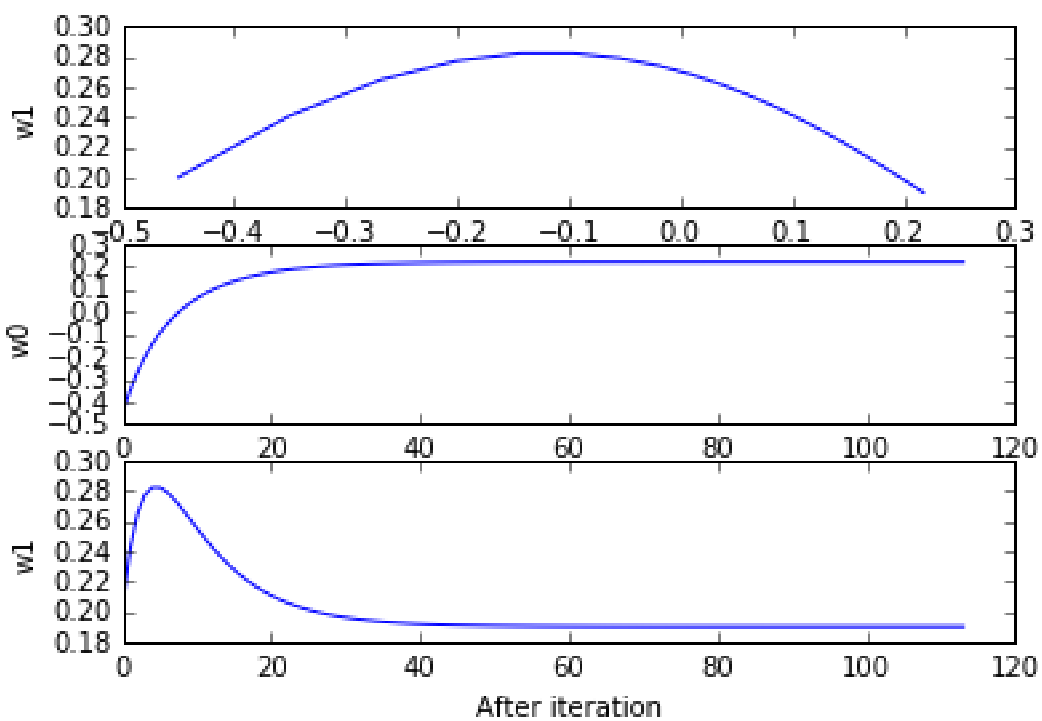

```
1. Dmitrij Feller dmitrij.feller@campus.tu-berlin.de
2. Ajla Dzajic ajla.dzajic@campus.tu-berlin.de
3. Jan Kliewer jankliewer1@googlemail.com
4. Onat Tanriöver onat.tanriover@yahoo.com
5. Seema Narasimha Swamy seema.n.swamy@campus.tu-berlin.de
6. Christopher Vahldieck vahldieckc@gmail.com
```


Gradient Descent with constant learning rate

In [18]:
```
import numpy as np
import numpy.linalg as lng
import matplotlib.pyplot as plt
import numpy.random as rnd
%matplotlib inline

X = np.array([[1, 1, 1], [-1, 0.3, 2]], dtype=float)
y = np.array([[-0.1, 0.5, 0.5]], dtype=float)
w = np.array([[-0.45], [0.2]], dtype=float)
all_weights = []
H = np.matmul(X, np.transpose(X))
b = np.matmul(-X, np.transpose(y))
learning_rate = 0.05
all_weights.append(w)


for i in range(0, 3000):
    gradient = np.matmul(H, w) + b
    if lng.norm(gradient) <= 1.0E-6:
        print('Algorithm converged in', i, 'steps')
        break
    w = w - learning_rate * gradient
    all_weights.append(w)
```

```
print('Weights found with constant learning rate', w)
w_0 = [all_weights[i][0, 0] for i in range(0, len(all_weights))]
w_1 = [all_weights[i][1, 0] for i in range(0, len(all_weights))]
x_axis = [i for i in range(0, len(all_weights))]

plt.figure()
plt.subplots_adjust(left=0.125, right=0.9, bottom=0.1, top=0.9, wspace=0.2
plt.subplot(311)
plt.plot(w_0, w_1)
plt.xlabel('w0')
plt.ylabel('w1')
plt.subplot(312)
plt.plot(x_axis, w_0)
plt.xlabel('After iteration')
plt.ylabel('w0')
plt.subplot(313)
plt.plot(x_axis, w_1)
plt.xlabel('After iteration')
plt.ylabel('w1')
plt.show()
```

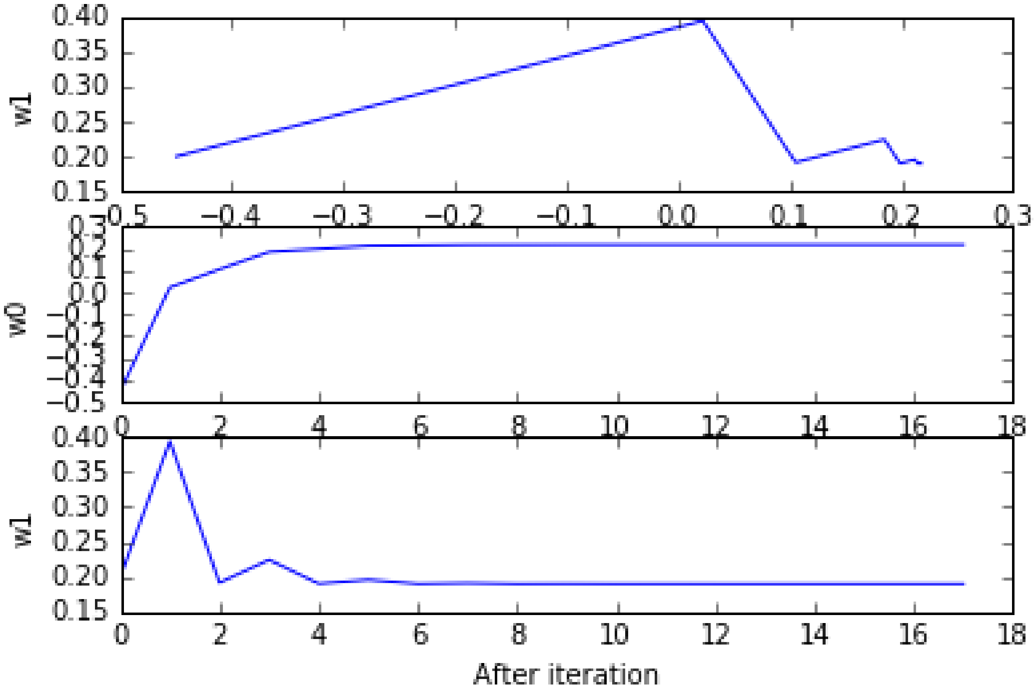
```
('Algorithm converged in', 113, 'steps')
('Weights found with constant learning rate', array([[ 0.21767271],
       [ 0.18998543]]))
```



Line search

```
In [19]: all_weights = []
         w = np.array([[-0.45], [0.2]], dtype=float)

         all_weights.append(w)

         for i in range(0, 3000):
             gradient = np.matmul(H, w) + b
             if lng.norm(gradient) <= 1.0E-6:
                 print('Algorithm converged in', i, 'steps')
                 break
             learning_rate = np.matmul(np.transpose(gradient), gradient)[0, 0] / np
             w = w - learning_rate * gradient
             all_weights.append(w)
         print('Weights found with line search method', w)
         w_0 = [all_weights[i][0, 0] for i in range(0, len(all_weights))]
         w_1 = [all_weights[i][1, 0] for i in range(0, len(all_weights))]
         x_axis = [i for i in range(0, len(all_weights))]


         plt.figure()
         plt.subplot(311)
         plt.plot(w_0, w_1)
         plt.xlabel('w0')
         plt.ylabel('w1')
         plt.subplot(312)
         plt.plot(x_axis, w_0)
         plt.xlabel('After iteration')
         plt.ylabel('w0')
         plt.subplot(313)
         plt.plot(x_axis, w_1)
         plt.xlabel('After iteration')
         plt.ylabel('w1')
         plt.show()
```

```
('Algorithm converged in', 17, 'steps')
('Weights found with line search method', array([[ 0.21767292],
       [ 0.18998541]]))
```



Conjugate gradient

```
In [20]: all_weights = []
         w = np.array([[-0.45], [0.2]], dtype=float)
         all_weights.append(w)
         old_gradient = 0.0
         gradient = np.matmul(H, w) + b
         d = -gradient
         for i in range(0, 3000):
             if lng.norm(gradient) <= 1.0E-6:
                 print('Algorithm converged in', i, 'steps')
                 break
             step_size = - np.matmul(np.transpose(d), gradient)[0, 0] / np.matmul(n
             w = w + step_size * d
             all_weights.append(w)
             old_gradient = gradient
             gradient = np.matmul(H, w) + b
             momentum = -np.matmul(np.transpose(gradient), gradient)[0, 0] / np.mat
             d = gradient + momentum * d
         print('Weights found with conjugate gradient', w)
         w_0 = [all_weights[i][0, 0] for i in range(0, len(all_weights))]
         w_1 = [all_weights[i][1, 0] for i in range(0, len(all_weights))]
         x_axis = [i for i in range(0, len(all_weights))]

         plt.figure()
```

```python
plt.subplot(311)
plt.plot(w_0, w_1)
plt.xlabel('w0')
plt.ylabel('w1')
plt.subplot(312)
plt.plot(x_axis, w_0)
plt.xlabel('After iteration')
plt.ylabel('w0')
plt.subplot(313)
plt.plot(x_axis, w_1)
plt.xlabel('After iteration')
plt.ylabel('w1')
plt.show()
```

```
('Algorithm converged in', 2, 'steps')
('Weights found with conjugate gradient', array([[ 0.21767305],
       [ 0.18998527]]))
```