solution04

November 10, 2020

Exercise Sheet 4 **Gradient methods for parameter optimization**

[1]:
```
import matplotlib.pyplot as plt
import numpy as np
mm = np.matmul # matrix multiplication
```

4.2 Comparison of gradient descent methods

[2]:
```
# INITIATION

X = np.array([[1,-1],[1,0.3],[1,2]]).T # input and bias
Y = np.array([-0.1,0.5,0.5]) # output
W = np.array([-.45,0.2]) # weights
iE = 0.001 # acceptable weight change for end of loop
```

[3]:
```
# FUNCTIONS

def _Hess_(X): # Hessian
    return mm(X,X.T)

def _g_(X,Y,W): # Gradient
    H = _Hess_(X)
    b = -mm(X,Y.T)
    return mm(H,W)+b

def _LRg_(X,Y,W): # dynamic Learning rate based on Conjugate Gradient
    g = _g_(X,Y,W)
    H = _Hess_(X)
    return mm(g.T,g)/mm(g.T,mm(H,g))

def _LRd_(X,Y,W,d): # dynamic Learning rate based on direction
    g = _g_(X,Y,W)
    H = _Hess_(X)
    return mm(d.T,g)/mm(d.T,mm(H,d))

def _Beta_(g0,g1): # momentum factor
    return mm(g1.T,g1)/mm(g0.T,g0)
```

```
def _plot_(Wt):
    plt.figure(figsize=(7,7))
    plt.subplot(2,1,1)
    plt.scatter(Wt[0][0],Wt[1][0],color='b',label='Starting weight')
    for i in range(1,len(Wt[0])-1):
        if i == 1:
            plt.scatter(Wt[0][i],Wt[1][i],color='r',label='Proceeding␣
↪weight(s)')
        else:
            plt.scatter(Wt[0][i],Wt[1][i],color='r')
    plt.scatter(Wt[0][-1],Wt[1][-1],color='g',label='Ending weight')
    plt.xlabel('w0')
    plt.ylabel('w1')
    plt.legend()
    plt.subplot(2,1,2)
    plt.plot(np.arange(len(Wt[0])),Wt[0])
    plt.plot(np.arange(len(Wt[1])),Wt[1])
    plt.xlabel('iteration')
    plt.ylabel('W')
    plt.legend(('bias weight','input weight'))
    plt.title('')
    plt.show()
```

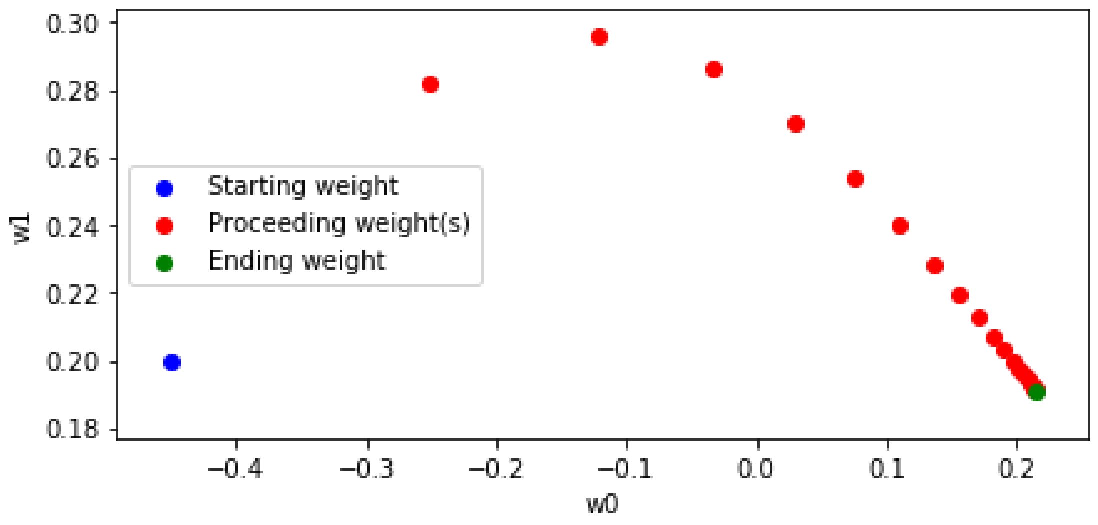
(a)

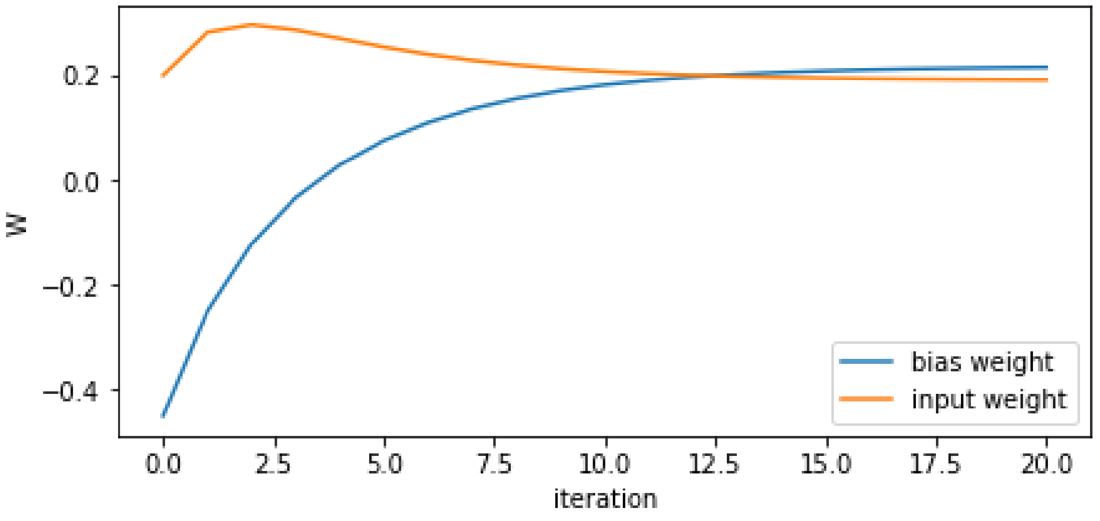
[4]:
```
# Gradient Descent

Wi = np.array([-.45,0.2])
iLR = 0.1
iEpochMax = 100
iEpoch = 0
Wt =[[],[]]
Wt[0].append(Wi[0])
Wt[1].append(Wi[1])
while iEpoch < iEpochMax:
    g = _g_(X,Y,Wi)
    Wi = Wi - iLR*g
    Wt[0].append(Wi[0])
    Wt[1].append(Wi[1])
    if np.sum(np.abs(iLR*_g_(X,Y,Wi))) <= iE:
        print('Based on acceptable weight change, '
              'the process converged after %d epochs '
              'to the final weight:\n%s'%(iEpoch+1, Wi))
        break
    iEpoch += 1
_plot_(Wt)
```

```
Based on acceptable weight change, the process converged after 20 epochs to the
```

```
final weight:
[ 0.21527225  0.19113568]
```



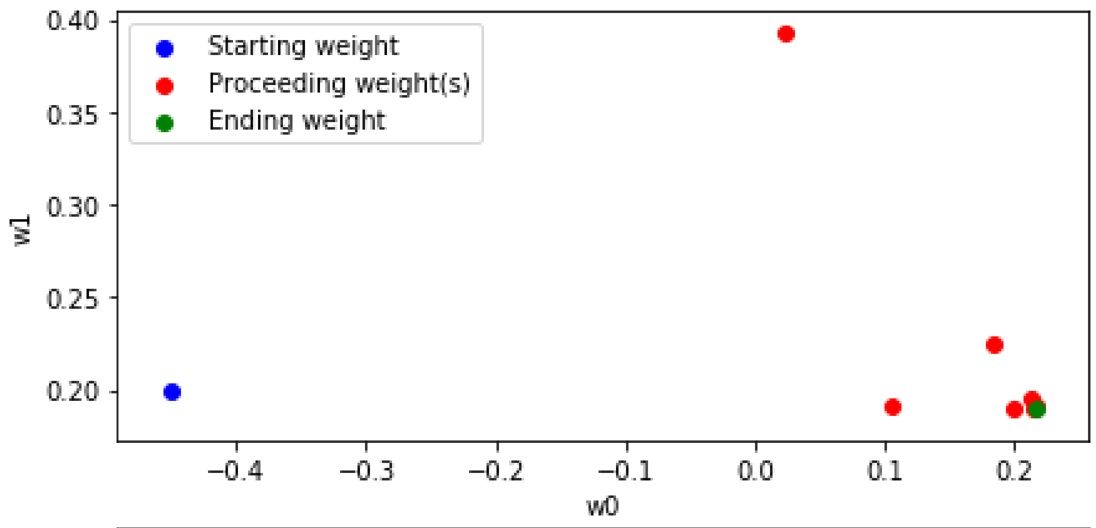


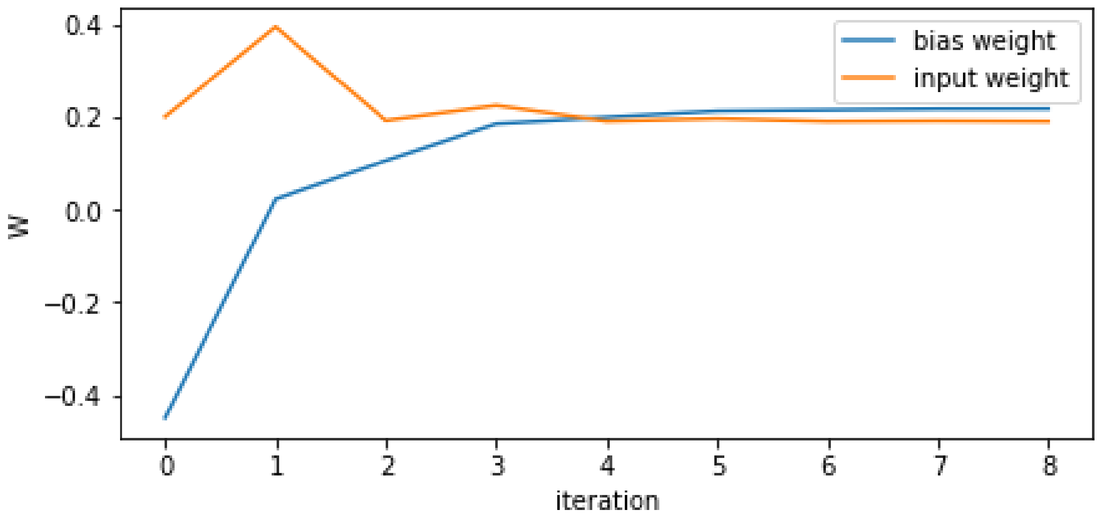
(b)

[5]:
```
# LINE SEARCH

Wi = np.array([-.45,0.2])
iEpochMax = 100
iEpoch = 0
Wt =[[],[]]
Wt[0].append(Wi[0])
Wt[1].append(Wi[1])
while iEpoch < iEpochMax:
    g = _g_(X,Y,Wi)
    iLR = _LRg_(X,Y,Wi)
    Wi = Wi - iLR*g
    Wt[0].append(Wi[0])
    Wt[1].append(Wi[1])
    if np.sum(np.abs(_LRg_(X,Y,Wi)*_g_(X,Y,Wi))) <= iE:
        print('Based on acceptable weight change, '
              'the process converged after %d epochs '
              'to the final weight:\n%s'%(iEpoch+1, Wi))
        break
    iEpoch += 1
_plot_(Wt)
```

```
Based on acceptable weight change, the process converged after 8 epochs to the
final weight:
[ 0.21713403  0.18999336]
```



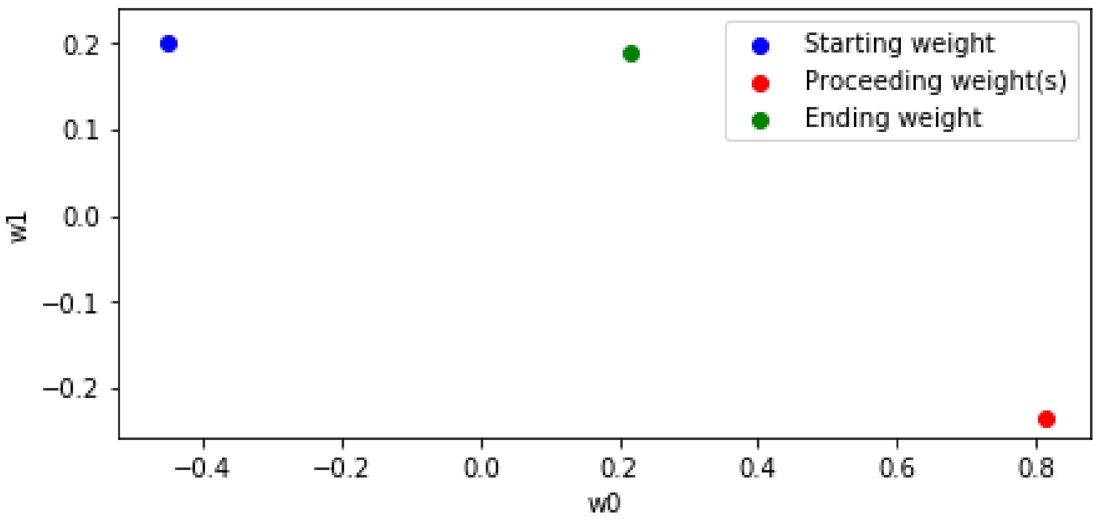


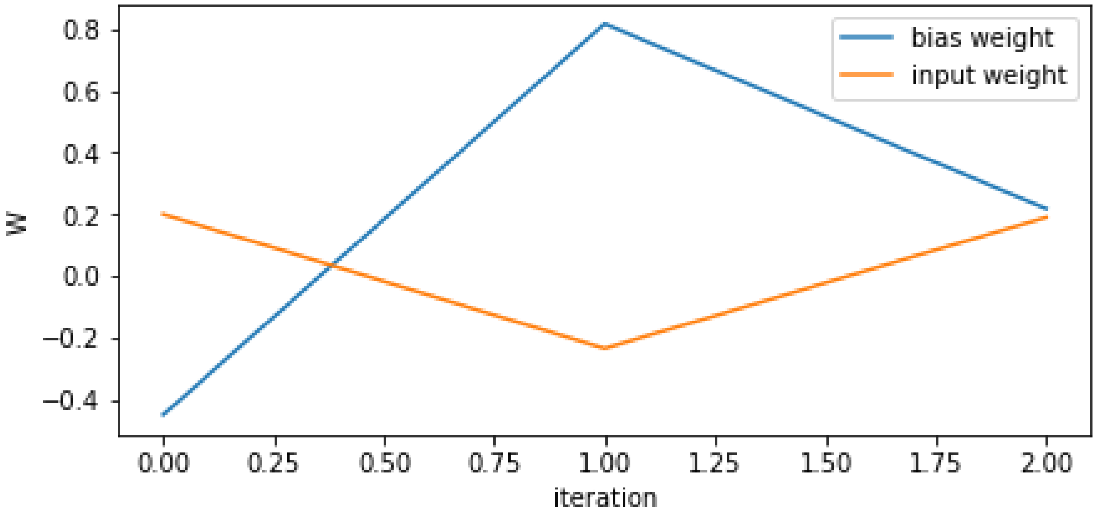
(c)

```
[6]: # Conjugate Gradient

Wi = np.array([-.45,0.2])
iEpochMax = 10
iEpoch = 0
Wt =[[],[]]
Wt[0].append(Wi[0])
Wt[1].append(Wi[1])
Wi = -_g_(X,Y,Wi)
di = -_g_(X,Y,Wi)
while iEpoch < iEpochMax:
    g0 = _g_(X,Y,Wi)
    iLR = _LRd_(X,Y,Wi,di)
    Wi = Wi - iLR*di
    g1 = _g_(X,Y,Wi)
    iBeta = _Beta_(g0,g1)
    di = g1-iBeta*di
    Wt[0].append(Wi[0])
    Wt[1].append(Wi[1])
    if np.sum(np.abs(_LRd_(X,Y,Wi,di)*di)) <= iE:
        print('Based on acceptable weight change, '
              'the process converged after %d epochs '
              'to the final weight:\n%s'%(iEpoch+1, Wi))
        break
    iEpoch += 1
_plot_(Wt)
```

```
Based on acceptable weight change, the process converged after 2 epochs to the
final weight:
[ 0.21767305  0.18998527]
```





Finally, it is obvious (based on the presented plots/results) that Conjugate Gradient converges faster than Line Search method and Line Search converges better than Gradient Descent (considering the same criteria for convergence). From the other hand, the necessary computation for Conjugate Gradient is more demanding (complex) than Line Search and Gradient Descent has the simplest implementation. Also, it seems that in the Line Search, the convergence is not a steady process where in Gradient Descent, the weights converge slowly but steadily.