lect6 Gradient Descent

September 20, 2016

In [2]:
```
# The last line is just for Jupyter.  It tells matplotlib where to plot.
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
%matplotlib notebook
```

1 We will use gradient descent to solve linear least squares

Minimizing

$$\|Ax - b\|_2^2,$$

and showing off the use of LaTex inside Markdown.

In [3]:
```
A = np.random.randn(6,3)
A
```

Out[3]:
```
array([[ 1.15060602, -0.01218118, -0.04796248],
       [ 1.15437098, -0.39000259,  2.12394918],
       [-0.5050806 ,  0.595549  , -0.28412123],
       [-1.26502479, -0.02941371, -0.9850352 ],
       [ 0.46446842,  0.9712223 ,  2.88669233],
       [-0.99269205,  0.34621931, -3.24982113]])
```

In [4]:
```
b = np.random.randn(6,1)
b
```

Out[4]:
```
array([[-0.22250668],
       [-0.32637213],
       [-0.09209012],
       [ 1.67930288],
       [-1.31278255],
       [ 0.17752924]])
```

In [5]:
```
x = np.zeros((3,1))
x
```

Out[5]:
```
array([[ 0.],
       [ 0.],
       [ 0.]])
```

In [6]:
```
# This will produce an error.
# * does not multiply matrices
A*x
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)

<ipython-input-6-41169bdeaf36> in <module>()
  1 # This will produce an error.
  2 # * does not multiply matrices
----> 3 A*x


ValueError: operands could not be broadcast together with shapes (6,3) (3,1
```

In [7]:
```
# A.dot(x) is the result of multiplying A by the vector x
A.dot(x)
```

Out[7]:
```
array([[ 0.],
       [ 0.],
       [ 0.],
       [ 0.],
       [ 0.],
       [ 0.]])
```

In [8]:
```
# It also works for matrices
M1 = np.array([[1, 1],[0, 1]])
print "M1: \n", M1
M2 = np.array([[1, 0],[0, 2]])
print "M2: \n" , M2
print "M1*M2: \n", M1.dot(M2)
```

```
M1:
[[1 1]
 [0 1]]
M2:
[[1 0]
 [0 2]]
M1*M2:
[[1 2]
 [0 2]]
```

In [ ]:

In [9]:
```
Atrans = A.transpose()
Atrans
```

```
Out[9]: array([[ 1.15060602,  1.15437098, -0.5050806 , -1.26502479,  0.46446842,
                -0.99269205],
               [-0.01218118, -0.39000259,  0.595549  , -0.02941371,  0.9712223 ,
                 0.34621931],
               [-0.04796248,  2.12394918, -0.28412123, -0.9850352 ,  2.88669233,
                -3.24982113]])

In [10]: # we now define the function we want to minimize
         def f(x):
             return sum((A.dot(x) - b)**2)
         f(x)

Out[10]: array([ 4.73948141])
```

Recall that the gradient is

$$2A^T(Ax - b)$$

```
In [11]: # this is a funtion that returns its gradient
         def grad_f(x):
             return 2*Atrans.dot(A.dot(x)-b)

         grad_f(x)

Out[11]: array([[  6.99319359],
                [  2.37556394],
                [ 13.3541423 ]])

In [12]: r = np.random.randn(3,1)*1e-6
         r

Out[12]: array([[  2.03599178e-08],
                [  5.45307095e-07],
                [ -8.73490800e-08]])

In [13]: f(x+r) - f(x)

Out[13]: array([  2.71321219e-07])

In [14]: # to multiply g by r, we could first turn it into a row vector by taking
         grad_f(x).transpose().dot(r)

Out[14]: array([[  2.71320673e-07]])

In [15]: # or, we could turn them into one-dimensional arrays by slicing
         grad_f(x)[:,0].dot(r[:,0])

Out[15]: 2.7132067308217276e-07

In [16]: r[:,0]
```

```
Out[16]: array([  2.03599178e-08,   5.45307095e-07,  -8.73490800e-08])

In [18]: r[:,0].shape

Out[18]: (3,)

In [19]: r.shape

Out[19]: (3, 1)

In [23]: # play with this line to see more examples
         r = np.random.randn(3,1)*1e-3
         [f(x+r) - f(x), np.dot(grad_f(x)[:,0],r[:,0])]

Out[23]: [array([ 0.00880828]), 0.0087974122017961051]

In [24]: def grad_descent(x0, f, g, tol=10**(-6)):

             f0 = f(x0)  # initial value
             g0 = g(x0)  # initial gradient

             step_size = 0.1  # initially

             while ((np.linalg.norm(g0) > tol) & (step_size > tol)):

                 xnew = x0-step_size*g0
                 fnew = f(xnew)

                 if (fnew < f0):
                     f0 = fnew   # the best function value
                     x0 = xnew   # the corresponding vector
                     g0 = g(x0)  # and, its gradient
                     print f0    # this will be a pain in a long run
                 else:
                     step_size = step_size/2

             return x0

In [25]: x1 = grad_descent(np.zeros((3,1)), f, grad_f)

[ 2.81413791]
[ 2.39362162]
[ 2.20839309]
[ 2.07103017]
[ 1.95259173]
[ 1.84750624]
[ 1.75378369]
[ 1.6700981]
[ 1.59534154]
```

```
[ 1.5285429]
[ 1.46884254]
[ 1.41547726]
[ 1.36776837]
[ 1.32511171]
[ 1.28696891]
[ 1.2528599]
[ 1.22235635]
[ 1.19507588]
[ 1.17067703]
[ 1.14885476]
[ 1.1293365]
[ 1.11187864]
[ 1.09626345]
[ 1.08229625]
[ 1.06980298]
[ 1.05862802]
[ 1.04863218]
[ 1.03969101]
[ 1.03169319]
[ 1.02453916]
[ 1.01813989]
[ 1.01241573]
[ 1.00729545]
[ 1.00271534]
[ 0.99861841]
[ 0.99495367]
[ 0.99167554]
[ 0.98874323]
[ 0.98612025]
[ 0.98377398]
[ 0.98167522]
[ 0.97979785]
[ 0.97811854]
[ 0.97661637]
[ 0.97527267]
[ 0.97407072]
[ 0.97299556]
[ 0.97203382]
[ 0.97117353]
[ 0.970404]
[ 0.96971564]
[ 0.9690999]
[ 0.96854911]
[ 0.96805643]
[ 0.96761572]
[ 0.9672215]
[ 0.96686887]
```

```
[ 0.96655343]
[ 0.96627127]
[ 0.96601888]
[ 0.96579311]
[ 0.96559116]
[ 0.96541051]
[ 0.96524891]
[ 0.96510437]
[ 0.96497507]
[ 0.96485941]
[ 0.96475596]
[ 0.96466341]
[ 0.96458063]
[ 0.96450658]
[ 0.96444035]
[ 0.9643811]
[ 0.9643281]
[ 0.96428069]
[ 0.96423828]
[ 0.96420035]
[ 0.96416641]
[ 0.96413606]
[ 0.96410891]
[ 0.96408462]
[ 0.9640629]
[ 0.96404347]
[ 0.96402608]
[ 0.96401053]
[ 0.96399663]
[ 0.96398418]
[ 0.96397305]
[ 0.9639631]
[ 0.96395419]
[ 0.96394623]
[ 0.9639391]
[ 0.96393273]
[ 0.96392703]
[ 0.96392193]
[ 0.96391737]
[ 0.96391329]
[ 0.96390964]
[ 0.96390637]
[ 0.96390345]
[ 0.96390084]
[ 0.9638985]
[ 0.96389641]
[ 0.96389454]
[ 0.96389287]
```

```
[ 0.96389137]
[ 0.96389003]
[ 0.96388884]
[ 0.96388776]
[ 0.96388681]
[ 0.96388595]
[ 0.96388518]
[ 0.9638845]
[ 0.96388388]
[ 0.96388334]
[ 0.96388285]
[ 0.96388241]
[ 0.96388201]
[ 0.96388166]
[ 0.96388135]
[ 0.96388107]
[ 0.96388082]
[ 0.96388059]
[ 0.96388039]
[ 0.96388021]
[ 0.96388005]
[ 0.9638799]
[ 0.96387978]
[ 0.96387966]
[ 0.96387956]
[ 0.96387947]
[ 0.96387938]
[ 0.96387931]
[ 0.96387924]
[ 0.96387918]
[ 0.96387913]
[ 0.96387908]
[ 0.96387904]
[ 0.963879]
[ 0.96387897]
[ 0.96387894]
[ 0.96387891]
[ 0.96387889]
[ 0.96387887]
[ 0.96387885]
[ 0.96387883]
[ 0.96387882]
[ 0.9638788]
[ 0.96387879]
[ 0.96387878]
[ 0.96387877]
[ 0.96387876]
[ 0.96387875]
```

```
[ 0.96387874]
[ 0.96387874]
[ 0.96387873]
[ 0.96387873]
[ 0.96387872]
[ 0.96387872]
[ 0.96387871]
[ 0.96387871]
[ 0.96387871]
[ 0.96387871]
[ 0.9638787]
[ 0.9638787]
[ 0.9638787]
[ 0.9638787]
[ 0.9638787]
[ 0.9638787]
[ 0.96387869]
[ 0.96387869]
[ 0.96387869]
[ 0.96387869]
[ 0.96387869]
[ 0.96387869]
[ 0.96387869]
[ 0.96387869]
[ 0.96387869]
[ 0.96387869]
[ 0.96387869]
[ 0.96387869]
[ 0.96387869]
[ 0.96387869]
[ 0.96387869]
[ 0.96387869]
[ 0.96387869]
[ 0.96387869]
[ 0.96387869]
[ 0.96387869]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
```

```
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
```

```
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
[ 0.96387868]
```

```
In [26]: # the best vector
         x1

Out[26]: array([[-0.74353994],
                [-1.05497996],
                [ 0.01158344]])

In [27]: # the fuction value there
         f(x1)

Out[27]: array([ 0.96387868])

In [28]: # the gradient at x1
         grad_f(x1)

Out[28]: array([[  4.57893717e-07],
                [  8.60847059e-07],
                [ -1.89797915e-07]])

In [29]: # the norm of the gradient
         np.linalg.norm(grad_f(x1))

Out[29]: 9.9335168155578283e-07
```

1.1 in larger dimensions

We now make a new problem. A and b were global variables and f and grad_f know about them. So, we do not need to redefine f and grad_f. But, we do need to redefine Atrans, which is used inside f.

```
In [30]: np.random.seed(1)
         n = 10
         A = np.random.randn(n,n)
         Atrans = A.transpose()
         b = np.zeros((n,1))

         x0 = np.random.randn(n,1)
         f(x0)

Out[30]: array([ 40.34491096])

In [31]: x1 = grad_descent(x0,f,grad_f,tol=10**(-3))

[ 32.58916415]
[ 2.56896543]
[ 1.49271742]
[ 1.22090968]
[ 1.02169664]
[ 0.86228249]
[ 0.73319711]
[ 0.62808727]
[ 0.54216454]
[ 0.47169504]
[ 0.41371931]
[ 0.36587093]
[ 0.32624811]
[ 0.29331768]
[ 0.26584105]
[ 0.24281607]
[ 0.22343082]
[ 0.20702679]
[ 0.19306928]
[ 0.18112365]
[ 0.1708362]
[ 0.16191877]
[ 0.15413631]
[ 0.14729683]
[ 0.14124331]
[ 0.13584712]
[ 0.13100274]
[ 0.12662343]
[ 0.12263779]
[ 0.11898689]
[ 0.11562201]
[ 0.11250279]
[ 0.10959565]
[ 0.10687264]
[ 0.10431041]
```

```
[ 0.10188936]
[ 0.09959303]
[ 0.09740751]
[ 0.09532102]
[ 0.09332354]
[ 0.09140651]
[ 0.08956261]
[ 0.08778549]
[ 0.0860697]
[ 0.08441048]
[ 0.08280367]
[ 0.08124565]
[ 0.07973318]
[ 0.07826344]
[ 0.0768339]
[ 0.07544228]
[ 0.07408657]
[ 0.07276494]
[ 0.07147574]
[ 0.07021746]
[ 0.06898872]
[ 0.06778829]
[ 0.06661499]
[ 0.06546778]
[ 0.06434566]
[ 0.06324773]
[ 0.06217314]
[ 0.06112109]
[ 0.06009085]
[ 0.05908171]
[ 0.05809304]
[ 0.0571242]
[ 0.05617464]
[ 0.05524378]
[ 0.05433111]
[ 0.05343615]
[ 0.0525584]
[ 0.05169744]
[ 0.05085283]
[ 0.05002415]
[ 0.04921103]
[ 0.04841307]
[ 0.04762992]
[ 0.04686124]
[ 0.04610669]
[ 0.04536594]
[ 0.04463869]
[ 0.04392464]
```

```
[ 0.04322349]
[ 0.04253497]
[ 0.04185881]
[ 0.04119474]
[ 0.04054251]
[ 0.03990186]
[ 0.03927257]
[ 0.03865439]
[ 0.0380471]
[ 0.03745048]
[ 0.03686431]
[ 0.03628839]
[ 0.03572251]
[ 0.03516646]
[ 0.03462007]
[ 0.03408314]
[ 0.03355548]
[ 0.03303691]
[ 0.03252726]
[ 0.03202636]
[ 0.03153404]
[ 0.03105014]
[ 0.03057449]
[ 0.03010694]
[ 0.02964733]
[ 0.02919552]
[ 0.02875136]
[ 0.0283147]
[ 0.02788541]
[ 0.02746334]
[ 0.02704837]
[ 0.02664036]
[ 0.02623919]
[ 0.02584472]
[ 0.02545683]
[ 0.0250754]
[ 0.02470032]
[ 0.02433146]
[ 0.02396872]
[ 0.02361198]
[ 0.02326113]
[ 0.02291607]
[ 0.02257668]
[ 0.02224288]
[ 0.02191455]
[ 0.02159159]
[ 0.02127392]
[ 0.02096144]
```

```
[ 0.02065404]
[ 0.02035165]
[ 0.02005416]
[ 0.01976151]
[ 0.01947358]
[ 0.01919032]
[ 0.01891162]
[ 0.01863741]
[ 0.01836761]
[ 0.01810214]
[ 0.01784093]
[ 0.01758389]
[ 0.01733096]
[ 0.01708207]
[ 0.01683714]
[ 0.0165961]
[ 0.01635888]
[ 0.01612543]
[ 0.01589566]
[ 0.01566952]
[ 0.01544695]
[ 0.01522788]
[ 0.01501225]
[ 0.01480001]
[ 0.01459109]
[ 0.01438543]
[ 0.01418298]
[ 0.01398369]
[ 0.0137875]
[ 0.01359435]
[ 0.0134042]
[ 0.01321699]
[ 0.01303267]
[ 0.0128512]
[ 0.01267252]
[ 0.01249659]
[ 0.01232336]
[ 0.01215279]
[ 0.01198482]
[ 0.01181942]
[ 0.01165654]
[ 0.01149614]
[ 0.01133818]
[ 0.01118261]
[ 0.0110294]
[ 0.01087851]
[ 0.01072989]
[ 0.01058352]
```

```
[ 0.01043934]
[ 0.01029734]
[ 0.01015746]
[ 0.01001968]
[ 0.00988396]
[ 0.00975026]
[ 0.00961855]
[ 0.00948881]
[ 0.00936099]
[ 0.00923506]
[ 0.009111]
[ 0.00898877]
[ 0.00886835]
[ 0.0087497]
[ 0.00863279]
[ 0.0085176]
[ 0.0084041]
[ 0.00829226]
[ 0.00818206]
[ 0.00807346]
[ 0.00796644]
[ 0.00786098]
[ 0.00775705]
[ 0.00765463]
[ 0.00755368]
[ 0.0074542]
[ 0.00735615]
[ 0.00725952]
[ 0.00716427]
[ 0.00707039]
[ 0.00697786]
[ 0.00688665]
[ 0.00679674]
[ 0.00670812]
[ 0.00662076]
[ 0.00653464]
[ 0.00644974]
[ 0.00636605]
[ 0.00628354]
[ 0.0062022]
[ 0.006122]
[ 0.00604294]
[ 0.00596499]
[ 0.00588813]
[ 0.00581235]
[ 0.00573764]
[ 0.00566396]
[ 0.00559132]
```

```
[ 0.00551969]
[ 0.00544905]
[ 0.0053794]
[ 0.00531071]
[ 0.00524298]
[ 0.00517618]
[ 0.00511031]
[ 0.00504534]
[ 0.00498127]
[ 0.00491808]
[ 0.00485576]
[ 0.0047943]
[ 0.00473367]
[ 0.00467388]
[ 0.0046149]
[ 0.00455672]
[ 0.00449934]
[ 0.00444274]
[ 0.00438691]
[ 0.00433183]
[ 0.0042775]
[ 0.00422391]
[ 0.00417103]
[ 0.00411888]
[ 0.00406742]
[ 0.00401665]
[ 0.00396657]
[ 0.00391716]
[ 0.00386841]
[ 0.00382032]
[ 0.00377286]
[ 0.00372604]
[ 0.00367984]
[ 0.00363426]
[ 0.00358928]
[ 0.0035449]
[ 0.00350111]
[ 0.00345789]
[ 0.00341525]
[ 0.00337317]
[ 0.00333165]
[ 0.00329067]
[ 0.00325023]
[ 0.00321032]
[ 0.00317094]
[ 0.00313207]
[ 0.00309371]
[ 0.00305585]
```

```
[ 0.00301849]
[ 0.00298161]
[ 0.00294522]
[ 0.00290929]
[ 0.00287383]
[ 0.00283884]
[ 0.00280429]
[ 0.0027702]
[ 0.00273654]
[ 0.00270332]
[ 0.00267053]
[ 0.00263816]
[ 0.0026062]
[ 0.00257466]
[ 0.00254352]
[ 0.00251278]
[ 0.00248244]
[ 0.00245248]
[ 0.0024229]
[ 0.00239371]
[ 0.00236488]
[ 0.00233643]
[ 0.00230833]
[ 0.00228059]
[ 0.00225321]
[ 0.00222617]
[ 0.00219947]
[ 0.00217311]
[ 0.00214709]
[ 0.00212139]
[ 0.00209602]
[ 0.00207097]
[ 0.00204623]
[ 0.0020218]
[ 0.00199768]
[ 0.00197387]
[ 0.00195035]
[ 0.00192713]
[ 0.00190419]
[ 0.00188155]
[ 0.00185919]
[ 0.0018371]
[ 0.00181529]
[ 0.00179376]
[ 0.00177249]
[ 0.00175148]
[ 0.00173074]
[ 0.00171025]
```

```
[ 0.00169002]
[ 0.00167004]
[ 0.0016503]
[ 0.00163081]
[ 0.00161156]
[ 0.00159255]
[ 0.00157377]
[ 0.00155523]
[ 0.00153691]
[ 0.00151882]
[ 0.00150095]
[ 0.00148329]
[ 0.00146586]
[ 0.00144864]
[ 0.00143163]
[ 0.00141483]
[ 0.00139823]
[ 0.00138183]
[ 0.00136564]
[ 0.00134964]
[ 0.00133384]
[ 0.00131823]
[ 0.00130281]
[ 0.00128758]
[ 0.00127254]
[ 0.00125767]
[ 0.00124299]
[ 0.00122848]
[ 0.00121415]
[ 0.00119999]
[ 0.00118601]
[ 0.00117219]
[ 0.00115854]
[ 0.00114506]
[ 0.00113173]
[ 0.00111857]
[ 0.00110557]
[ 0.00109272]
[ 0.00108003]
[ 0.00106749]
[ 0.0010551]
[ 0.00104286]
[ 0.00103077]
[ 0.00101882]
[ 0.00100701]
[ 0.00099534]
[ 0.00098382]
[ 0.00097243]
```

```
[ 0.00096118]
[ 0.00095006]
[ 0.00093908]
[ 0.00092822]
[ 0.0009175]
[ 0.0009069]
[ 0.00089643]
[ 0.00088609]
[ 0.00087586]
[ 0.00086576]
[ 0.00085578]
[ 0.00084592]
[ 0.00083617]
[ 0.00082654]
[ 0.00081702]
[ 0.00080762]
[ 0.00079832]
[ 0.00078914]
[ 0.00078007]
[ 0.0007711]
[ 0.00076224]
[ 0.00075348]
[ 0.00074482]
[ 0.00073627]
[ 0.00072782]
[ 0.00071947]
[ 0.00071122]
[ 0.00070306]
[ 0.000695]
[ 0.00068703]
[ 0.00067916]
[ 0.00067138]
[ 0.00066369]
[ 0.00065609]
[ 0.00064858]
[ 0.00064116]
[ 0.00063383]
[ 0.00062658]
[ 0.00061941]
[ 0.00061233]
[ 0.00060534]
[ 0.00059842]
[ 0.00059158]
[ 0.00058483]
[ 0.00057815]
[ 0.00057155]
[ 0.00056503]
[ 0.00055859]
```

```
[ 0.00055221]
[ 0.00054592]
[ 0.00053969]
[ 0.00053354]
[ 0.00052746]
[ 0.00052145]
[ 0.00051552]
[ 0.00050965]
[ 0.00050384]
[ 0.00049811]
[ 0.00049244]
[ 0.00048684]
[ 0.0004813]
[ 0.00047583]
[ 0.00047042]
[ 0.00046507]
[ 0.00045978]
[ 0.00045456]
[ 0.00044939]
[ 0.00044429]
[ 0.00043924]
[ 0.00043426]
[ 0.00042933]
[ 0.00042445]
[ 0.00041964]
[ 0.00041488]
[ 0.00041017]
[ 0.00040552]
[ 0.00040092]
[ 0.00039638]
[ 0.00039188]
[ 0.00038744]
[ 0.00038305]
[ 0.00037871]
[ 0.00037442]
[ 0.00037018]
[ 0.00036599]
[ 0.00036185]
[ 0.00035775]
[ 0.0003537]
[ 0.0003497]
[ 0.00034574]
[ 0.00034183]
[ 0.00033796]
[ 0.00033414]
[ 0.00033036]
[ 0.00032663]
[ 0.00032294]
```

```
[ 0.00031929]
[ 0.00031568]
[ 0.00031211]
[ 0.00030859]
[ 0.0003051]
[ 0.00030166]
[ 0.00029825]
[ 0.00029488]
[ 0.00029155]
[ 0.00028826]
[ 0.00028501]
[ 0.00028179]
[ 0.00027861]
[ 0.00027547]
[ 0.00027236]
[ 0.00026929]
[ 0.00026625]
[ 0.00026325]
[ 0.00026028]
[ 0.00025735]
[ 0.00025445]
[ 0.00025158]
[ 0.00024875]
[ 0.00024594]
[ 0.00024317]
[ 0.00024043]
[ 0.00023773]
[ 0.00023505]
[ 0.0002324]
[ 0.00022978]
[ 0.0002272]
[ 0.00022464]
[ 0.00022211]
[ 0.00021961]
[ 0.00021714]
[ 0.0002147]
[ 0.00021228]
[ 0.00020989]
[ 0.00020753]
[ 0.0002052]
[ 0.00020289]
[ 0.00020061]
[ 0.00019835]
[ 0.00019612]
[ 0.00019392]
[ 0.00019174]
[ 0.00018958]
[ 0.00018745]
```

```
[ 0.00018535]
[ 0.00018326]
[ 0.0001812]
[ 0.00017917]
[ 0.00017715]
[ 0.00017516]
[ 0.0001732]
[ 0.00017125]
[ 0.00016933]
[ 0.00016743]
[ 0.00016555]
[ 0.00016369]
[ 0.00016185]
[ 0.00016003]
[ 0.00015824]
[ 0.00015646]
[ 0.00015471]
[ 0.00015297]
[ 0.00015125]
[ 0.00014956]
[ 0.00014788]
[ 0.00014622]
[ 0.00014458]
[ 0.00014296]
[ 0.00014135]
[ 0.00013977]
[ 0.0001382]
[ 0.00013665]
[ 0.00013512]
[ 0.00013361]
[ 0.00013211]
[ 0.00013063]
[ 0.00012916]
[ 0.00012771]
[ 0.00012628]
[ 0.00012487]
[ 0.00012347]
[ 0.00012208]
[ 0.00012072]
[ 0.00011936]
[ 0.00011803]
[ 0.0001167]
[ 0.0001154]
[ 0.0001141]
[ 0.00011283]
[ 0.00011156]
[ 0.00011031]
[ 0.00010908]
```

```
[ 0.00010786]
[ 0.00010665]
[ 0.00010545]
[ 0.00010427]
[ 0.00010311]
[ 0.00010195]
[ 0.00010081]
[  9.96822679e-05]
[  9.85664686e-05]
[  9.74632166e-05]
[  9.63723697e-05]
[  9.52937872e-05]
[  9.42273301e-05]
[  9.31728607e-05]
[  9.21302434e-05]
[  9.10993438e-05]
[  9.00800291e-05]
[  8.90721681e-05]
[  8.80756311e-05]
[  8.70902897e-05]
[  8.61160174e-05]
[  8.51526886e-05]
[  8.42001797e-05]
[  8.32583680e-05]
[  8.23271327e-05]
[  8.14063539e-05]
[  8.04959136e-05]
[  7.95956946e-05]
[  7.87055815e-05]
[  7.78254599e-05]
[  7.69552170e-05]
[  7.60947411e-05]
[  7.52439217e-05]
[  7.44026499e-05]
[  7.35708176e-05]
[  7.27483183e-05]
[  7.19350466e-05]
[  7.11308982e-05]
[  7.03357701e-05]
[  6.95495606e-05]
[  6.87721689e-05]
[  6.80034954e-05]
[  6.72434419e-05]
[  6.64919110e-05]
[  6.57488067e-05]
[  6.50140337e-05]
[  6.42874982e-05]
[  6.35691072e-05]
```

```
[  6.28587690e-05]
[  6.21563927e-05]
[  6.14618885e-05]
[  6.07751678e-05]
[  6.00961427e-05]
[  5.94247267e-05]
[  5.87608338e-05]
[  5.81043794e-05]
[  5.74552797e-05]
[  5.68134518e-05]
[  5.61788138e-05]
[  5.55512848e-05]
[  5.49307847e-05]
[  5.43172343e-05]
[  5.37105555e-05]
[  5.31106708e-05]
[  5.25175038e-05]
[  5.19309790e-05]
[  5.13510214e-05]
[  5.07775574e-05]
[  5.02105137e-05]
[  4.96498182e-05]
[  4.90953995e-05]
[  4.85471870e-05]
[  4.80051109e-05]
[  4.74691021e-05]
[  4.69390925e-05]
[  4.64150145e-05]
[  4.58968017e-05]
[  4.53843879e-05]
[  4.48777080e-05]
[  4.43766976e-05]
[  4.38812930e-05]
[  4.33914312e-05]
[  4.29070499e-05]
[  4.24280875e-05]
[  4.19544831e-05]
[  4.14861767e-05]
[  4.10231086e-05]
[  4.05652201e-05]
[  4.01124529e-05]
[  3.96647497e-05]
[  3.92220534e-05]
[  3.87843080e-05]
[  3.83514578e-05]
[  3.79234478e-05]
[  3.75002239e-05]
[  3.70817321e-05]
```

```
[  3.66679195e-05]
[  3.62587336e-05]
[  3.58541223e-05]
[  3.54540345e-05]
[  3.50584193e-05]
[  3.46672265e-05]
[  3.42804067e-05]
[  3.38979106e-05]
[  3.35196899e-05]
[  3.31456965e-05]
[  3.27758831e-05]
[  3.24102029e-05]
[  3.20486093e-05]
[  3.16910568e-05]
[  3.13374998e-05]
[  3.09878937e-05]
[  3.06421942e-05]
[  3.03003574e-05]
[  2.99623402e-05]
[  2.96280996e-05]
[  2.92975933e-05]
[  2.89707796e-05]
[  2.86476170e-05]
[  2.83280647e-05]
[  2.80120821e-05]
[  2.76996293e-05]
[  2.73906668e-05]
[  2.70851554e-05]
[  2.67830565e-05]
[  2.64843319e-05]
[  2.61889437e-05]
[  2.58968547e-05]
[  2.56080279e-05]
[  2.53224267e-05]
[  2.50400151e-05]
[  2.47607572e-05]
[  2.44846179e-05]
[  2.42115622e-05]
[  2.39415556e-05]
[  2.36745639e-05]
[  2.34105534e-05]
[  2.31494908e-05]
[  2.28913430e-05]
[  2.26360774e-05]
[  2.23836618e-05]
[  2.21340642e-05]
[  2.18872532e-05]
[  2.16431975e-05]
```

```
[  2.14018664e-05]
[  2.11632293e-05]
[  2.09272562e-05]
[  2.06939171e-05]
[  2.04631827e-05]
[  2.02350238e-05]
[  2.00094115e-05]
[  1.97863175e-05]
[  1.95657135e-05]
[  1.93475718e-05]
[  1.91318646e-05]
[  1.89185649e-05]
[  1.87076457e-05]
[  1.84990804e-05]
[  1.82928427e-05]
[  1.80889065e-05]
[  1.78872462e-05]
[  1.76878361e-05]
[  1.74906513e-05]
[  1.72956669e-05]
[  1.71028581e-05]
[  1.69122008e-05]
[  1.67236708e-05]
[  1.65372444e-05]
[  1.63528981e-05]
[  1.61706086e-05]
[  1.59903530e-05]
[  1.58121084e-05]
[  1.56358525e-05]
[  1.54615629e-05]
[  1.52892178e-05]
[  1.51187954e-05]
[  1.49502742e-05]
[  1.47836330e-05]
[  1.46188508e-05]
[  1.44559068e-05]
[  1.42947804e-05]
[  1.41354513e-05]
[  1.39778996e-05]
[  1.38221053e-05]
[  1.36680487e-05]
[  1.35157105e-05]
[  1.33650715e-05]
[  1.32161127e-05]
[  1.30688154e-05]
[  1.29231609e-05]
[  1.27791309e-05]
[  1.26367073e-05]
```

```
[  1.24958722e-05]
[  1.23566077e-05]
[  1.22188964e-05]
[  1.20827210e-05]
[  1.19480642e-05]
[  1.18149091e-05]
[  1.16832389e-05]
[  1.15530371e-05]
[  1.14242872e-05]
[  1.12969731e-05]
[  1.11710788e-05]
[  1.10465882e-05]
[  1.09234859e-05]
[  1.08017563e-05]
[  1.06813841e-05]
[  1.05623540e-05]
[  1.04446512e-05]
[  1.03282608e-05]
[  1.02131682e-05]
[  1.00993589e-05]
[  9.98681850e-06]
[  9.87553290e-06]
[  9.76548810e-06]
[  9.65667023e-06]
[  9.54906560e-06]
[  9.44266067e-06]
[  9.33744206e-06]
[  9.23339652e-06]
[  9.13051095e-06]
[  9.02877242e-06]
[  8.92816812e-06]
[  8.82868540e-06]
[  8.73031174e-06]
[  8.63303476e-06]
[  8.53684223e-06]
[  8.44172204e-06]
[  8.34766223e-06]
[  8.25465097e-06]
[  8.16267656e-06]
[  8.07172742e-06]
[  7.98179212e-06]
[  7.89285935e-06]
[  7.80491792e-06]
[  7.71795678e-06]
[  7.63196498e-06]
[  7.54693170e-06]
[  7.46284627e-06]
[  7.37969810e-06]
```

```
[  7.29747673e-06]
[  7.21617183e-06]
[  7.13577318e-06]
[  7.05627066e-06]
[  6.97765428e-06]
[  6.89991415e-06]
[  6.82304050e-06]
[  6.74702367e-06]
[  6.67185409e-06]
[  6.59752231e-06]
[  6.52401900e-06]
[  6.45133491e-06]
[  6.37946090e-06]
[  6.30838795e-06]
[  6.23810710e-06]
[  6.16860954e-06]
[  6.09988651e-06]
[  6.03192940e-06]
[  5.96472965e-06]
[  5.89827881e-06]
[  5.83256854e-06]
[  5.76759058e-06]
[  5.70333676e-06]
[  5.63979900e-06]
[  5.57696932e-06]
[  5.51483982e-06]
[  5.45340270e-06]
[  5.39265023e-06]
[  5.33257478e-06]
[  5.27316880e-06]
[  5.21442483e-06]
[  5.15633548e-06]
[  5.09889346e-06]
[  5.04209154e-06]
[  4.98592259e-06]
[  4.93037956e-06]
[  4.87545545e-06]
[  4.82114338e-06]
[  4.76743652e-06]
[  4.71432812e-06]
[  4.66181150e-06]
[  4.60988008e-06]
[  4.55852732e-06]
[  4.50774677e-06]
[  4.45753206e-06]
[  4.40787688e-06]
[  4.35877499e-06]
[  4.31022021e-06]
```

```
[  4.26220646e-06]
[  4.21472770e-06]
[  4.16777796e-06]
[  4.12135135e-06]
[  4.07544204e-06]
[  4.03004426e-06]
[  3.98515230e-06]
[  3.94076054e-06]
[  3.89686338e-06]
[  3.85345533e-06]
[  3.81053092e-06]
[  3.76808477e-06]
[  3.72611155e-06]
[  3.68460598e-06]
[  3.64356285e-06]
[  3.60297700e-06]
[  3.56284335e-06]
[  3.52315684e-06]
[  3.48391250e-06]
[  3.44510540e-06]
[  3.40673066e-06]
[  3.36878346e-06]
[  3.33125905e-06]
[  3.29415269e-06]
[  3.25745975e-06]
[  3.22117560e-06]
[  3.18529570e-06]
[  3.14981554e-06]
[  3.11473066e-06]
[  3.08003665e-06]
[  3.04572916e-06]
[  3.01180389e-06]
[  2.97825657e-06]
[  2.94508300e-06]
[  2.91227899e-06]
[  2.87984045e-06]
[  2.84776329e-06]
[  2.81604348e-06]
[  2.78467705e-06]
[  2.75366006e-06]
[  2.72298860e-06]
[  2.69265884e-06]
[  2.66266696e-06]
[  2.63300920e-06]
[  2.60368184e-06]
[  2.57468118e-06]
[  2.54600360e-06]
[  2.51764549e-06]
```

```
[  2.48960329e-06]
[  2.46187348e-06]
[  2.43445259e-06]
[  2.40733716e-06]
[  2.38052379e-06]
[  2.35400912e-06]
[  2.32778983e-06]
[  2.30186261e-06]
[  2.27622421e-06]
[  2.25087142e-06]
[  2.22580105e-06]
```

```
In [32]: f(x1)
```

```
Out[32]: array([  2.22580105e-06])
```

That took a lot of iterations, and it was only 10 dimensional! So, we will limit the number of iterations of gradient descent, and stop all the printing.

```
In [33]: def grad_descent(x0, f, g, tol=10**(-6), max_iters = 10**5):

             f0 = f(x0)
             g0 = g(x0)

             step_size = 0.1
             iters = 0

             while ((np.linalg.norm(g0) > tol) & (step_size > tol) & (iters < max_i
                 xnew = x0-step_size*g0
                 fnew = f(xnew)
                 iters += 1

                 if (fnew < f0):
                     f0 = fnew
                     x0 = xnew
                     g0 = g(x0)
                 else:
                     step_size = step_size/2

             print "function value:", f0
             print "number iterations:", iters

             return x0
```

With this protection in place, we can handle larger matrices

```
In [34]: np.random.seed(1)
         n = 100
```

```
A = np.random.randn(n,n)
Atrans = A.transpose()
b = np.zeros((n,1))

x0 = np.random.randn(n,1)
f(x0)
```

```
Out[34]: array([ 9992.15931358])
```

```
In [35]: x1 = grad_descent(x0,f,grad_f,tol=10**(-3),max_iters=10)
```

```
function value: [ 194.18072624]
number iterations: 10
```

```
In [36]: x1 = grad_descent(x0,f,grad_f,tol=10**(-3),max_iters=100)
         x1 = grad_descent(x0,f,grad_f,tol=10**(-3),max_iters=1000)
         x1 = grad_descent(x0,f,grad_f,tol=10**(-3),max_iters=10000)
         x1 = grad_descent(x0,f,grad_f,tol=10**(-3),max_iters=100000)
```

```
function value: [ 4.4088504]
number iterations: 100
function value: [ 0.12314598]
number iterations: 1000
function value: [ 0.00682233]
number iterations: 10000
function value: [  5.84378556e-05]
number iterations: 100000
```

To see how long that took, we can time it.

```
In [37]: %time x1 = grad_descent(x0,f,grad_f,tol=10**(-3),max_iters=10000)
```

```
function value: [ 0.00682233]
number iterations: 10000
CPU times: user 2.25 s, sys: 36.2 ms, total: 2.28 s
Wall time: 1.21 s
```

1.2 Computing it directly using matrix inversion

We use the following formula from lecture 2:

$$x = (A^T A)^{-1} A^T b$$

```
In [38]: x = np.linalg.inv(Atrans.dot(A)).dot(Atrans.dot(b))
         f(x)
```

```
Out[38]: array([ 0.])
```

2 Things to be careful of when using vectors

Some of these lines might surprise you until you understand what's going on.

```
In [3]: x = np.zeros((3,1))

In [4]: y = x
        y[0] = 1
        x

Out[4]: array([[ 1.],
               [ 0.],
               [ 0.]])

In [5]: y -= np.random.randn(3,1)
        y

Out[5]: array([[ 2.13917673],
               [ 0.43729632],
               [ 0.33470413]])

In [6]: x

Out[6]: array([[ 2.13917673],
               [ 0.43729632],
               [ 0.33470413]])

In [7]: y = y + 4
        y

Out[7]: array([[ 6.13917673],
               [ 4.43729632],
               [ 4.33470413]])

In [8]: x

Out[8]: array([[ 2.13917673],
               [ 0.43729632],
               [ 0.33470413]])

In [ ]:
```