hw1

September 16, 2017

1 ML - Homework 1

Gede Ria Ghosalya - 1001841
Note: Question 1 on last page

```
In [1]: %matplotlib inline

In [2]: 1+1

Out[2]: 2

In [3]: import numpy as np
        import matplotlib.pyplot as plt

In [4]: csv='https://www.dropbox.com/s/oqoyy9p849ewzt2/linear.csv?dl=1'

In [5]: data = np.genfromtxt(csv, delimiter=',')
        X = data[:,1:]
        Y = data[:,0]

In [6]: print(X)
```

```
[[-1.47752373 -0.0502532  -0.17023633  1.        ]
 [ 0.90709037  0.66451566  0.47865149  1.        ]
 [ 0.4003217   0.43267376 -0.43504849  1.        ]
 [-1.65365073  0.126796    1.00236757  1.        ]
 [-1.06691275 -0.83842795 -0.34888078  1.        ]
 [-0.68878191 -0.80913707  0.43642513  1.        ]
 [ 0.35745996  1.13429946  1.56657725  1.        ]
 [ 0.09528788 -2.06401766  1.04123572  1.        ]
 [ 1.23035184  1.61568621  1.48715984  1.        ]
 [-0.83620287  0.07849988  2.13596839  1.        ]
 [-0.43319969 -0.63613778 -0.38595494  1.        ]
 [ 0.28831115  0.24175838 -0.16395618  1.        ]
 [-2.13894535  0.97237413  0.10424867  1.        ]
 [ 0.99927052 -1.11433758 -0.45458174  1.        ]
 [ 0.49052428 -0.92271326 -0.35495995  1.        ]
 [ 0.38507689  0.08352817  0.02790376  1.        ]
 [ 0.69112273 -1.00672565 -0.88943049  1.        ]
 [-0.41774901  0.40816784  1.80858458  1.        ]
 [ 0.49400495  1.02792757 -1.00655099  1.        ]
 [ 0.58197659  0.63871748 -1.42460059  1.        ]
 [-0.01784407  0.71766039 -0.61215247  1.        ]
 [ 0.87391878 -1.75733762  0.4195343   1.        ]
 [-0.66181284  0.49298387 -0.04672385  1.        ]
 [ 1.23736049  2.21406654  1.58620271  1.        ]
 [-1.0553783   0.8954994  -1.84451136  1.        ]
 [ 2.30714568  0.25333497 -0.48961114  1.        ]
 [-0.4755072   2.056542    1.57110362  1.        ]
 [ 1.41596994 -0.92738418 -0.05858269  1.        ]
 [ 1.62513334 -1.12246556 -1.09691528  1.        ]
 [ 0.40780549  1.02729963 -0.04277794  1.        ]
 [ 1.06301487 -1.61341439  0.45159474  1.        ]
 [-0.67374261  0.74646502  0.88104198  1.        ]
 [ 0.13187129  0.31486386  1.09636616  1.        ]
 [-0.21322547  0.74141203  0.80339276  1.        ]
 [ 1.05339677  1.225563    0.16642039  1.        ]
 [ 0.15524151  0.10448045 -0.52662439  1.        ]
 [-0.69002679  0.84764888 -0.15939914  1.        ]
 [-0.45254444  0.74638293 -0.14693742  1.        ]
 [ 0.99099792 -0.99198341  0.56108373  1.        ]
 [ 0.53267167 -0.61727065  0.5028914   1.        ]
 [-1.94765626  0.15310133  0.55662237  1.        ]
 [ 1.46155517 -0.28306337  1.09531722  1.        ]
 [ 0.54346479 -1.82947935 -0.0865562   1.        ]
 [-0.31688096  0.67619353  0.30241958  1.        ]
 [-0.79405799 -0.02982322 -1.87777205  1.        ]
 [ 0.27861628  1.48764306 -1.04769245  1.        ]
 [ 0.61099245  1.00413207  0.56495587  1.        ]
 [ 0.06320248  1.48185744  0.07624095  1.        ]
 [-0.97676235  0.85265757 -0.78701903  1.        ]
 [-1.93031271 -0.60963352  0.77863057  1.        ]]
```

```
In [7]: Y
```

```
Out[7]: array([-1.13933054, -1.3895628 , -1.46427865, -0.77140228, -2.45451924,
               -2.53173137, -0.58514703, -4.53260862, -0.59382424, -1.38688906,
               -2.49527767, -1.73538943,  0.91300105, -4.14696407, -3.47921988,
               -2.00766789, -3.72876357, -1.11373905, -0.71692356, -1.33254433,
               -0.98473649, -4.77208525, -0.98948768,  0.44472789, -0.12907069,
               -2.77946764,  1.20221693, -3.91963952, -4.37192529, -0.49844168,
               -4.59804439, -0.38216805, -1.56516376, -0.7208978 , -0.7242968 ,
               -1.8829538 , -0.23740276, -0.71488811, -3.6406235 , -3.01168857,
               -0.47759712, -3.12712799, -4.75174197, -0.69416697, -1.65252122,
               -0.09836427, -0.87381787, -0.01287334, -0.06954225, -1.70276013])
```

```
In [8]: import theano

In [9]: import theano.tensor as T
        d = X.shape[1]
        n = X.shape[0]
        learn_rate = 0.5

In [10]: x = T.matrix(name='x')
         y = T.vector(name='y')
         w = theano.shared(np.zeros((d,1)),name='w')

In [11]: risk = T.sum((T.dot(x,w).T-y)**2)/(2*n)
         grad_risk = T.grad(risk, wrt=w)

In [12]: train_model = theano.function(inputs=[],
                                       outputs=risk,
                                       updates=[(w, w-learn_rate*grad_risk)],
                                       givens={x:X,y:Y})

In [13]: n_steps = 50
         for i in range(n_steps):
             print(train_model())
```

```
2.619322008585456
0.7587559422725657
0.23542812224973023
0.07939576047330808
0.0300968963645766
0.013695201975103278
0.008006092308871303
0.005970446508700004
0.005225932117259171
0.004949550127903775
0.004845924776165536
0.004806813029966767
0.004791984242871616
0.004786344307210613
0.004784194268800788
0.0047833731708215
0.004783059133878647
0.004782938874883767
0.004782892769416032
0.004782875074349396
0.004782868276122439
0.004782865661743263
0.004782864655366252
0.004782864267605405
0.004782864118060609
0.004782864060333999
```

```
0.004782864038030567
0.004782864029405719
0.004782864026067558
0.004782864024774455
0.004782864024273132
0.004782864024078615
0.004782864024003074
0.0047828640239737236
0.004782864023962309
0.004782864023957866
0.004782864023956133
0.004782864023955461
0.004782864023955196
0.0047828640239550935
0.004782864023955061
0.004782864023955039
0.004782864023955036
0.004782864023955029
0.00478286402395503
0.004782864023955029
0.004782864023955028
0.00478286402395503
0.00478286402395503
0.004782864023955033
```

1.0.1 Question 3a

```
In [14]: print(w.get_value())
```

```
[[-0.57392068]
 [ 1.35757059]
 [ 0.01527565]
 [-1.88288076]]
```

1.0.2 Question 3b

Parameter (from https://onlinecourses.science.psu.edu/stat501/node/382)

$$b = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{p-1} \end{bmatrix} = (X^{'}X)^{-1}X^{'}Y$$

```
In [15]: # B = (X'X)*X'Y
         Xt = np.transpose(X)     # X'
         Xn = np.matmul(Xt, X)    # (X'X)
         Xin = np.linalg.inv(Xn) # (X'X)*
         Xa = np.matmul(Xin, Xt) # (X'X)*X'
         B = np.matmul(Xa, Y)
         print(B)
```

```
[-0.57392068  1.35757059  0.01527565 -1.88288076]
```

w have exactly same parameter as *B*

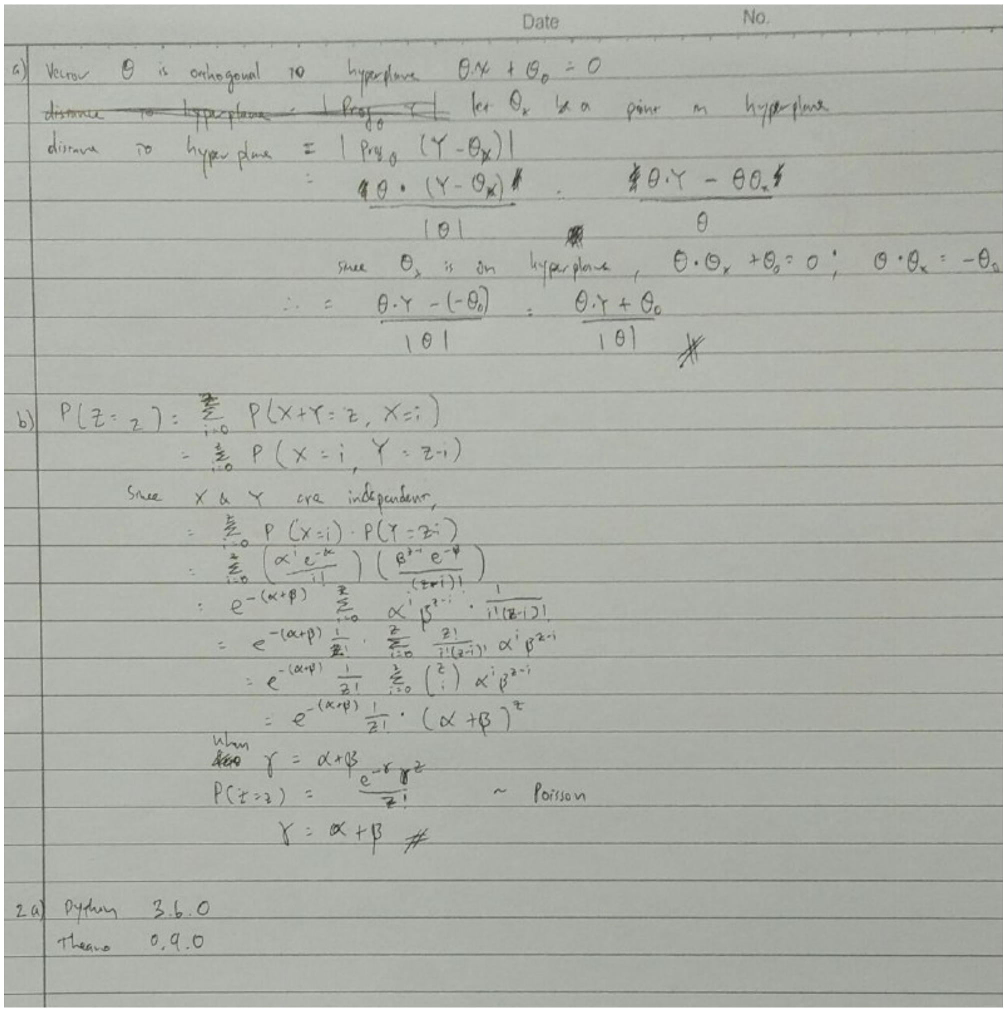
1.0.3 Question 3c

using *scipy's* SKLearn

```
In [16]: # question 3c
         # using scipy's sklearn
         from sklearn.linear_model import LinearRegression
         lrg = LinearRegression()
         lrg.fit(X, Y)
         print(lrg.coef_)
```

```
[-0.57392068  1.35757059  0.01527565  0.        ]
```

w has the first 3 coefficient to be the same as SKLearn's *parameters*, but the last one is regarded as 0 by SKLearn.



Date No.

a) Vector θ is orthogonal to hyperplane $\theta \cdot x + \theta_0 = 0$

~~distance to hyperplane = $|\mathrm{Proj}_\theta\, Y|$~~ let θ_x be a point on hyperplane

distance to hyperplane $= |\mathrm{Proj}_\theta\,(Y - \theta_x)|$

$$= \frac{\theta \cdot (Y - \theta_x)}{|\theta|} = \frac{\theta \cdot Y - \theta\theta_x}{\theta}$$

since θ_x is on hyperplane, $\theta \cdot \theta_x + \theta_0 = 0$; $\theta \cdot \theta_x = -\theta_0$

$$\therefore = \frac{\theta \cdot Y - (-\theta_0)}{|\theta|} = \frac{\theta \cdot Y + \theta_0}{|\theta|} \quad \#$$

b) $P(Z = z) = \sum_{i=0}^{z} P(X + Y = z, X = i)$

$= \sum_{i=0}^{z} P(X = i, Y = z - i)$

Since X & Y are independent,

$= \sum_{i=0}^{z} P(X = i) \cdot P(Y = z - i)$

$= \sum_{i=0}^{z} \left(\frac{\alpha^i e^{-\alpha}}{i!}\right)\left(\frac{\beta^{z-i} e^{-\beta}}{(z-i)!}\right)$

$= e^{-(\alpha+\beta)} \sum_{i=0}^{z} \alpha^i \beta^{z-i} \cdot \frac{1}{i!(z-i)!}$

$= e^{-(\alpha+\beta)} \frac{1}{z!} \cdot \sum_{i=0}^{z} \frac{z!}{i!(z-i)!} \alpha^i \beta^{z-i}$

$= e^{-(\alpha+\beta)} \frac{1}{z!} \sum_{i=0}^{z} \binom{z}{i} \alpha^i \beta^{z-i}$

$= e^{-(\alpha+\beta)} \frac{1}{z!} \cdot (\alpha + \beta)^z$

when $\gamma = \alpha + \beta$

$P(Z = z) = \frac{e^{-\gamma}\gamma^z}{z!}$ ~ Poisson

$\gamma = \alpha + \beta$ #

2 a) Python 3.6.0

Theano 0.9.0

picture