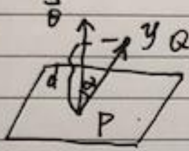
Li Xingxuan 1002189

1. Linear Algebra and Probability Review

(a)



$$d = |\vec{PQ}| \cdot \cos\alpha = \frac{|\vec{\theta}| \cdot |\vec{PQ}| \cdot \cos\alpha}{|\vec{\theta}|} = \frac{\vec{\theta} \cdot \vec{PQ}}{|\vec{\theta}|}$$

$$= \frac{(y - y_P)\vec{\theta}}{|\vec{\theta}|} = \frac{y\vec{\theta} - y_P\vec{\theta}}{|\vec{\theta}|}$$

$$= \frac{y\vec{\theta} + \theta_0}{|\vec{\theta}|}$$ (y_P is on the hyperplane, so $\vec{\theta} \cdot y_P + \theta_0 = 0$)

#

(b) $P(X=x) = \frac{a^x e^{-a}}{x!}$, $P(Y=y) = \frac{\beta^y \cdot e^{-\beta}}{y!}$, for all $x, y > 0$

$$P(Z = X+Y) = \sum P(x) \cdot P(z-x) = \sum \frac{\alpha^x e^{-\alpha}}{x!} \cdot \frac{\beta^{z-x} \cdot e^{-\beta}}{(z-x)!}$$

$$= e^{-\alpha-\beta} \sum \frac{\alpha^x \beta^{z-x}}{x!\,(z-x)!} = e^{-\alpha-\beta} \sum \frac{C_z^x \cdot \alpha^x \cdot \beta^{z-x}}{z!}$$

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

$Z = X+Y$ is also Poision, and the rate $\gamma = \alpha+\beta$

2. Python and Theano

(a) python : 3.6.5

theano : 1.0.2+2.gc449c86

(a) Original Code Result:

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

(b) Eexact solution:

```
In [13]: XTX = np.dot(X.T, X)
         inverseXTX = np.linalg.inv(XTX)
         XXX = np.dot(inverseXTX, X.T)
         theta = np.dot(XXX, Y)
         print(theta)

         [-0.57392068  1.35757059  0.01527565 -1.88288076]
```

(c) Sklearn:

```
In [1]: import pandas as pd
        df = pd.read_csv('linear.csv', header=None)
        X = df.drop(0, axis=1)
        y = df[[0]]
```

```
In [2]: from sklearn.linear_model import LinearRegression
        regression_model = LinearRegression()
        regression_model.fit(X, y)

Out[2]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

```
In [3]: for idx, col_name in enumerate(X.columns):
            print("The coefficient for {} is {}".format(col_name, regression_model.coef_[0][idx]))

        intercept = regression_model.intercept_[0]
        print("The intercept for our model is {}".format(intercept))

        The coefficient for 1 is -0.5739206829247852
        The coefficient for 2 is 1.3575705905387387
        The coefficient for 3 is 0.01527565450724058
        The coefficient for 4 is 0.0
        The intercept for our model is -1.882880764360151
```

(d) SGD

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
import random
data = np.genfromtxt('linear.csv',delimiter=',')
X = data[:,1:]
```

```
Y = data[:,0]
# print(type(X))
```

In [42]:

```
def mini_batch(X, y):
    loc = random.randint(1, 44)
    result_X = X[loc:loc + 5, :]
    result_Y = Y[loc:loc + 5]
    return result_X, result_Y

test_X, test_y = mini_batch(X, Y)
# print(test_X, test_y)
```

In [20]:

```
import theano
import theano.tensor as T

x = T.matrix(name='x')      # feature matrix
y = T.vector(name='y')      # responese vector
w = theano.shared(np.zeros((4,1)),name='w')      # model parameteres

risk = T.sum((T.dot(x,w).T - y)**2)/10      # empirical risk
grad_risk = T.grad(risk, wrt=w)      # gradient of the risk
```

In [41]:

```
n_steps = 1000
lost_list = []
w_dict = {}
for i in range(n_steps):
    for_x, for_y = mini_batch(X, Y)
    train_model = theano.function(inputs=[],
                                  outputs=risk,
                                  updates=[(w, w-(1/(i+1))*grad_risk)],
                                  givens={x:for_x, y:for_y})

#     print(train_model())
#     print(w.get_value())
    lost_list.append(train_model().item(0))
    w_dict[lost_list[-1]] = w.get_value()

print(w.get_value())
print(lost_list[-1])

lost_list.sort()
print(w_dict[lost_list[0]])
print(lost_list[0])
```

```
[[-0.57992765]
 [ 1.35436854]
 [ 0.01884709]
 [-1.88086824]]
```

```
0.006610305112215585
[[-0.5673264 ]
 [ 1.34984663]
 [ 0.017143  ]
 [-1.87757628]]
0.0007559927287628303
```