Lojistik Regresyon

```
from pandas import *
df = read_csv("testSet.txt",sep='\t',names=['x','y','labels'],header=None)
df['intercept']=1.0
data = df[['intercept','x','y']]
labels = df['labels']
print data[:10]
print labels[:10]
```

```
intercept         x          y
0          1 -0.017612  14.053064
1          1 -1.395634   4.662541
2          1 -0.752157   6.538620
3          1 -1.322371   7.152853
4          1  0.423363  11.054677
5          1  0.406704   7.067335
6          1  0.667394  12.741452
7          1 -2.460150   6.866805
8          1  0.569411   9.548755
9          1 -0.026632  10.427743
0    0
1    1
2    0
3    0
4    0
5    1
6    0
7    1
8    0
9    0
Name: labels
```

Sigmoid fonksiyonu

$$\frac{e^x}{1+e^x}$$

Fonksiyon oyle hazirlanmis ki, ne kadar buyuk olursa olsun ne zaman bir x degeri gecersek, bolendeki deger her zaman bolunenden 1 daha fazla olacaktir bu da fonksiyonun sonucunun 1'den her zaman kucuk olmasini garantiler. Cok kucuk x degerleri icin bolum sonucu biraz daha buyuk olacaktir tabii, vs. Daha temiz bir ifade icin bolen ve boluneni \$ e^{-x}\$ ile carpalim,

$$\frac{e^x e^{-x}}{e^{-x}+e^x e^{-x}}$$

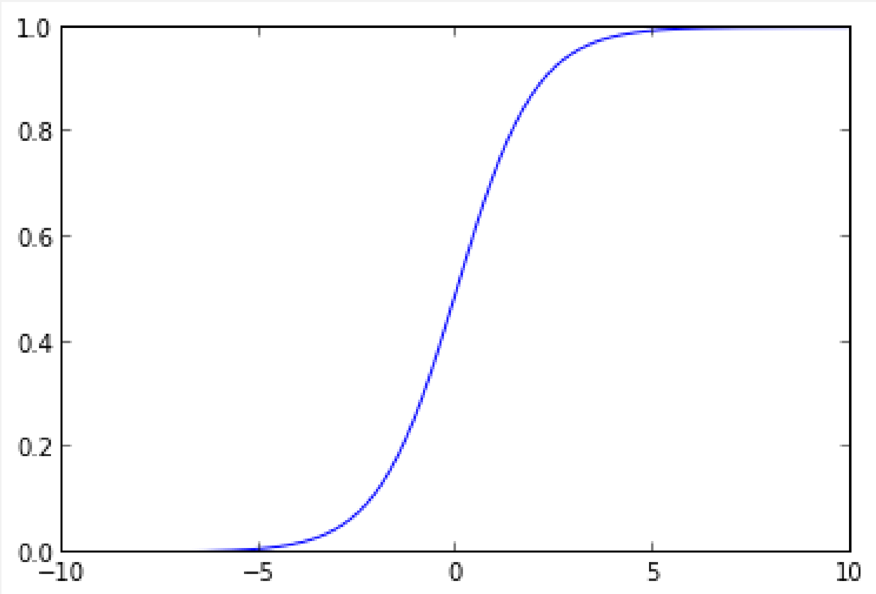
$$g(x) = \frac{1}{1+e^{-x}}$$

Sigmoid fonksiyonun "-sonsuzluk ile +sonsuzluk arasindaki degerleri 0 ve 1 arasina esledigi (map) / indirgedigi" sozu de litaraturde mevcuttur.

```
def sigmoid(arr):
    return 1.0/(1+exp(-arr))

x = np.array(arange(-10.0, 10.0, 0.1))
plot(x,sigmoid(x))
```

```
[<matplotlib.lines.Line2D at 0xadfea6c>]
```



Peki ustteki fonksiyon bir olasilik fonksiyonu olabilir mi?

```
import sympy
x = sympy.Symbol('x')
sympy.integrate('1/(1+exp(-x))')
```

```
x + log(1 + exp(-x))
```

Daha temizlemek icin

$x + \ln(1 + e^{-x})$

x ifadesi ayni zamanda suna esittir $x = ln(e^x)$. Bu ifade bize kolaylik saglayacak boylece,

$\ln e^x + \ln(1 + e^{-x})$

diyebiliriz. Dogal log'un (ln) carpimlari toplamlara donusturdugunu biliyoruz, bunu tersinden uygulayalim,

$\ln(e^x \cdot 1 + e^x e^{-x})$

$\ln(e^x + 1) = \ln(1 + e^x)$

```python
print log (1+exp(-inf))
print log(1+exp(inf))
```

```
0.0
inf
```

Demek ki fonksiyon bir olasilik dagilimi olamaz, cunku egri altindaki alan sonsuz buyuklugunde. Aslinda bu fonksiyonun kumulatif dagilim fonksiyonu (cumulative distribution function -CDF-) ozellikleri vardir, yani kendisi degil ama turevi bir olasilik fonksiyonu olarak kullanilabilir. Bu durumda g'nin 0 ile 1 arasinda olmasi da dagilim altindaki alanin en fazla 1 olabilmesi durumunu ortaya cikarir ki bu CDF tanimina uygundur.

Simdi elimizde olabilecek k tane degisken ve bu degiskenlerin bilinmeyen katsayilari icin 0 ve 1'e eslenecek bir regresyon olusturalim. Diyelim ki katsayilar $\theta_0, .., \theta_k$. Bu katsayilari degiskenler ile carpip toplayarak $h(x)$'e verelim, ve verideki etiketlere gore (0/1) cikip cikmayacagi katsayilara bagli olacak $h(x)$ sonucu ile eldeki veriler arasinda bir baglanti olusturmaya ugrasalim. Bu modele gore eger θ'yi ne kadar iyi secersek, eldeki veriye o kadar yaklasmis olacagiz.

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

"Veriye olabildigince yaklasmak icin en iyi α'yi bulmak" sozu bize maksimum olurluk (maximum likelihood) hesabini hatirlatmali. Bu hesaba gore icinde bilinmeyen α'yi barindiran formulun uzerinden tum verinin sonuclarinin teker teker birbiri ile carpimi olabildigince buyuk olmalidir. Bu ifadeyi maksimize edecek α veriye en uygun α olacaktir.

Simdi olasiliklari dusunelim

$$P(y = 1|x; \theta) = h_\theta(x)$$

$$P(y = 0|x; \theta) = 1 - h_\theta(x)$$

Not: Olasilik degerleri (buyuk $P(\cdot)$ ile), CDF fonksiyonlari (dagilim olmasa da) olurluk hesabinda kullanilabilir. Bu arada $P(X < x)$ gibi alansal hesaplar CDF uzerinden gerceklestirilebiliyor.

Hepsi bir arada olacak sekilde yanyana koyarsak,

$$p(y|x; \theta) = (h_\theta(x))^y (1 - h_\theta(x))^{1-y}$$

Olurluk icin tum veri noktalarini teker teker bu fonksiyona gecip sonuclarini carpacagiz (ve verilerin birinden bagimsiz olarak uretildigini farzediyoruz), eger m tane veri noktasi var ise

$$L(\theta) = \prod_{i=1}^{m} (h_\theta(x^i))^{y^i} (1 - h_\theta(x^i))^{1-y^i}$$

Eger log'unu alirsak carpimlar toplama donusur, isimiz daha rahatlasir,

$$l(\theta) = \log L(\theta)$$

$$= \sum_{i=1}^{m} y^i \log((h_\theta(x^i))) + (1 - y^i)\log((1 - h_\theta(x^i)))$$

Daha fazla ilerlemeden once bir esitlik ve bir turev gostermemiz gerekiyor. Once esitlik

$$1 - g(z) = g(-z)$$

$$1 - \frac{1}{1+e^{-z}} = \frac{1+e^{-z}-1}{1+e^{-z}}$$

$$\frac{e^{-z}}{1+e^{-z}} = \frac{1}{1+e^{z}}$$

Hakikaten son esitligin sag tarafina bakarsak, $g(-z)$'yi elde ettigimizi goruyoruz. Simdi tureve gelelim,

$$g'(z) = \frac{d}{dz}\frac{1}{1+e^{-z}} = \frac{1}{(1+e^{-z})^2}(e^{-z})$$

e^{-z} turevinden bir eksi isareti gelecegini beklemis olabilirsiniz, fakat hatirlayacagimiz uzere

$$\frac{d}{dx}\frac{1}{1+x} = \frac{-1}{(1+x)^2}$$

Yani eksiler birbirini yoketti. Simdi iki ustteki denklemin sag tarafini acalim

$$= \frac{1}{1+e^{-z}}\frac{e^{-z}}{1+e^{-z}}$$

$$= \frac{1}{1+e^{-z}}\frac{1}{1+e^{z}}$$

Carpimda iki bolum var, bolumler $g(z)$ ve $g(-z)$ olarak temsil edilebilir, ya da $g(z)$ ve $1 - g(z)$,

$$= g(z)(1 - g(z))$$

Artik olurluk denklemine donebiliriz. Olurlugu nasil maksimize ederiz? Gradyan inisi (gradient descent) kullanilabilir. Eger olurluk $l(\theta)$'nin en maksimal oldugu noktadaki θ'yi bulmak istiyorsak (dikkat sadece olurlugun en maksimal noktasini aramiyoruz, o noktadaki θ'yi ariyoruz), o zaman bir θ ile baslariz, ve adim adim θ'yi maksimal olana dogru yaklastiririz. Formul

$$\theta_{yeni} = \theta_{eski} + \alpha \nabla_\theta l(\theta)$$

Ustteki formul niye isler? Cunku gradyan $\nabla_\theta l(\theta)$, yani $l(\theta)$'nin gradyani her zaman fonksiyon artisinin en fazla oldugu yonu gosterir. Demek ki o yone adim atmak, yani $l(\theta)$'a verilen θ'yi o yonde degistirmek (degisim tabii ki θ bazinda, θ'nin degisimi), bizi fonksiyonun bir sonraki maksimum noktasina yaklastiracaktir. Sabit α bir tek sayi sadece, atilan adimin (hangi yonde olursa olsun) olcegini azaltip / arttirabilmek icin disaridan eklenir. Adim yonu vektor, bu sabit bir tek sayi. Carpimlari vektoru azaltir ya da cogaltir.

Simdi $\nabla_\theta l(\theta)$ turetmemiz gerekiyor.

Eger tek bir $\frac{\partial l(\theta)}{\partial \theta_j}$'yi hesaplarsak ve bunu her j icin yaparsak, bu sonuclari bir vektorde ustuste koyunca $\nabla_\theta l(\theta)$'yi elde ederiz.

$$\frac{\partial l(\theta)}{\partial \theta_j} = y \frac{\frac{\partial}{\partial \theta_j} g(\theta^T x)}{g(\theta^T x)} - (1-y) \frac{\frac{\partial}{\partial \theta_j} g(\theta^T x)}{1 - g(\theta^T x)}$$

$$= \left(y \frac{1}{g(\theta^T x)} - (1-y) \frac{1}{1-g(\theta^T x)} \right) \frac{\partial}{\partial \theta_j} g(\theta^T x)$$

Simdi en sagdaki kismi acalim,

$$\frac{\partial}{\partial \theta_j} g(\theta^T x) = g'(\theta^T x) \frac{\partial}{\partial \theta_j} \theta^T x = g'(\theta^T x) x_j$$

$\frac{\partial}{\partial \theta_j} \theta^T x$ nasil x_j haline geldi? Cunku tum θ vektorunun kismi turevini aliyoruz fakat o kismi turev sadece tek bir θ_j icin, o zaman vektordeki diger tum ogeler sifir olacaktir, sadece θ_j 1 olacak, ona tekabul eden x ogesi, yani x_j ayakta kalabilecek, diger x ogelerinin hepsi sifirla carpilmis olacak.

Turevin kendisinden de kurtulabiliriz simdi, daha once gosterdigimiz esitligi devreye sokalim,

$$= g(\theta^T x)(1 - g(\theta^T x)) x_j$$

Bu son formulu 3 ustteki formulun sag tarafina geri koyarsak, ve basitlestirirsek,

$$\big(y(1 - g(\theta^T x)) - (1-y) g(\theta^T x) \big) x_j$$

Carpimi daha temiz gormek icin sadece y, g harflerini kullanirsak,

$$\big(y(1-g) - (1-y)g \big) x_j = (y - yg - g + yg) x_j = (y - g) x_j$$

yani

$$= (y - g(\theta^T x)) x_j$$

$$= (y - h_\theta(x)) x_j$$

Iste $\nabla_\theta l(\theta)$ icin ne kullanacagimizi bulduk. O zaman, ve her i veri noktasi icin

$$\theta_{yeni} = \theta_{eski} + \alpha (y^i - h_\theta(x^i)) x_j^i$$

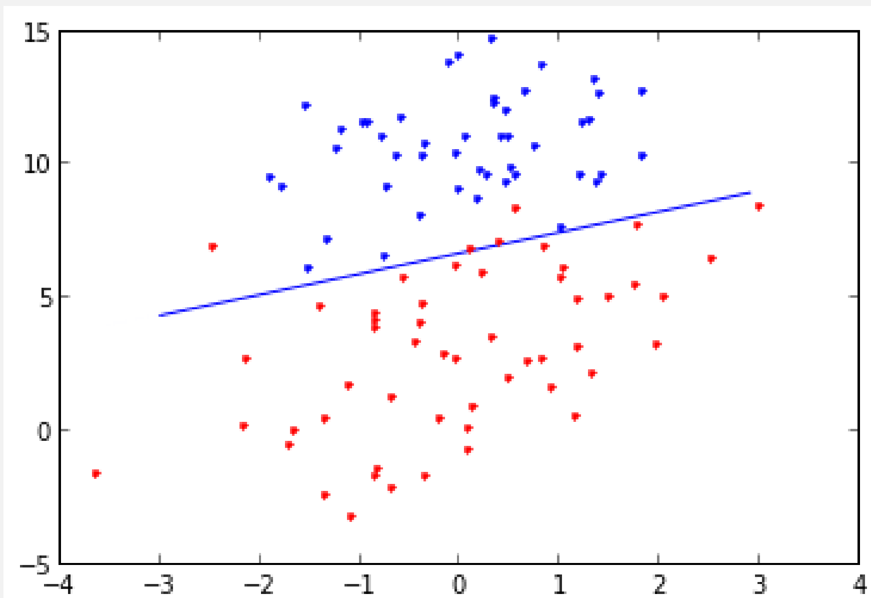
```
def grad_ascent(data_mat, label_mat):
    m,n = data_mat.shape
    label_mat=label_mat.reshape((m,1))
    alpha = 0.001
    iter = 500
    theta = ones((n,1))
    for k in range(iter):
        h = sigmoid(dot(data_mat,theta))
        error = label_mat - h
        theta = theta + alpha * dot(data_mat.T,error)
    return theta

theta = np.array(grad_ascent(array(data),array(labels).T ))
theta.T
```

```
array([[ 4.12414349,  0.48007329, -0.6168482 ]])
```

```
def plot_theta(theta):
     x = np.array(arange(-3.0, 3.0, 0.1))
     y = np.array((-theta[0]-theta[1]*x)/theta[2])
     plt.plot(x, y)
     plt.hold(True)
     class0 = data[labels==0]
     class1 = data[labels==1]
     plt.plot(class0['x'],class0['y'],'b.')
     plt.hold(True)
     plt.plot(class1['x'],class1['y'],'r.')
     plt.hold(True)

plot_theta(theta)
```



Ustteki kod bir dongu icinde belli bir x noktasindan baslayarak gradyan inisi yapti ve optimal θ degerlerini, yani regresyon agirliklarini (weights) hesapladi. Sonra bu agirliklari bir ayrac olarak ustte grafikledi. Ayracin oldukca iyi degerler buldugu belli oluyor.

Rasgele Gradyan Inisi (Stochastic Gradient Descent)

```
def stoc_grad_ascent0(data_mat, label_mat):
    m,n = data_mat.shape
    print m,n
    label_mat=label_mat.reshape((m,1))
    alpha = 0.01
    theta = ones((n,1))
```

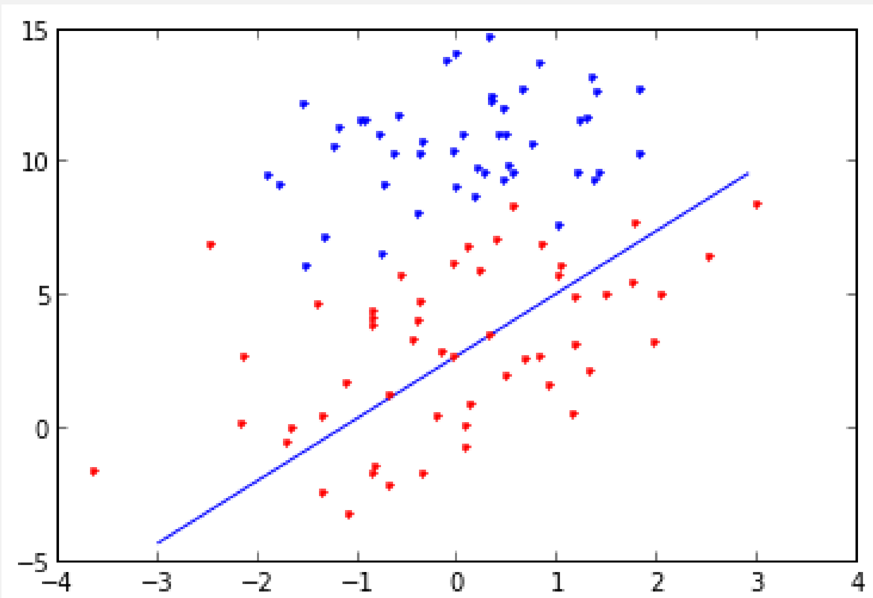
```
    for i in range(m):
        h = sigmoid(sum(dot(data_mat[i],theta)))
        error = label_mat[i] - h
        theta = theta + alpha * data_mat[i].reshape((n,1)) * error
        theta = theta.reshape((n,1))
    return theta

theta = np.array(stoc_grad_ascent0(array(data),array(labels).T ))
theta.T
```

```
100 3

array([[ 1.01702007,  0.85914348, -0.36579921]])
```

```
plot_theta(theta)
```




```
def stoc_grad_ascent1(data_mat, label_mat):
    m,n = data_mat.shape
    iter = 150
    label_mat=label_mat.reshape((m,1))
    alpha = 0.01
    theta = ones((n,1))
    for j in range(iter):
        data_index = range(m)
        for i in range(m):
            alpha = 4/(1.0+j+i)+0.0001
            rand_index = int(random.uniform(0,len(data_index)))
            h = sigmoid(sum(dot(data_mat[rand_index],theta)))
            error = label_mat[rand_index] - h
```

```
            theta = theta + alpha * data_mat[rand_index].reshape((n,1)) * error
            theta = theta.reshape((n,1))
    return theta

theta = np.array(stoc_grad_ascent1(array(data),array(labels).T ))
theta.T
```

```
array([[ 14.36959353,   1.24940793,  -1.84689592]])
```

```
plot_theta(theta)
```