# Lojistik Regresyon

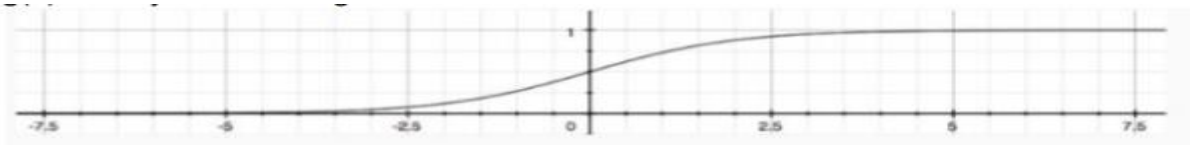
Sınıflandırma problemine, y'nin kesikli olduğu gerçeğini göz ardı ederek yaklaşabiliriz ve x değerini tahmin etmeye çalışmak için doğrusal regresyon algoritmasını kullanabiliriz. Mantıksal olarak, hipotezimizin değerinin 1'den büyük veya 0'dan küçük değerler alması mantıklı değildir, çünkü sonuçlarımızın {0, 1} kümesinden seçiyoruz. Bunu düzeltmek için, hipotezlerimiz için fonksiyonu değiştirelim. Fonksiyonumuz "Lojistik Fonksiyon" olarak da adlandırılan "Sigmoid Fonksiyon" unu kullanır:

$$h_\theta(x) = g(\theta^T x)$$

$$z = \theta^T x$$

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

g(z) fonksiyonuna baktığımızda



Burada gösterilen fonksion, herhangi bir gerçek sayıyı (0, 1) aralığına eşler ve bu da rastgele değerli bir fonksiyonun sınıflandırma için daha uygun bir fonksiyona dönüştürülmesini sağlamaktadır. g(z) nin z → ∞ olduğunda 1'e eğilimli olduğuna ve g(z) z → -∞ olduğunda 0'a doğru eğilim gösterdiğini görebiliyoruz. Ayrıca, g(z) ve dolayısıyla da h(x), daima 0 ile 1 arasında sınırlandırılmıştır. Peki lojistik regresyon verildiğinde parametrelerimizi, teta değerlerini nasıl bulacağız? Başlamadan önce sigmoid fonksiyonunun türevini hatırlayalım.

$$\begin{aligned} g'(z) &= \frac{d}{dz}\frac{1}{1+e^{-z}} \\ &= \frac{1}{(1+e^{-z})^2}(e^{-z}) \\ &= \frac{1}{(1+e^{-z})} \cdot \left(1 - \frac{1}{(1+e^{-z})}\right) \\ &= g(z)(1-g(z)). \end{aligned}$$

Aşağıdaki gibi bir varsayımda bulunabiliriz:

$$\begin{aligned} P(y = 1 \mid x; \theta) &= h_\theta(x) \\ P(y = 0 \mid x; \theta) &= 1 - h_\theta(x) \end{aligned}$$

veya bunu şöyle de ifade edebiliriz:

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

m tane eğitim örneğimizin bağımsız olarak üretildiğini varsayarsak, parametrelerin olasılığını aşağıdaki gibi yazabiliriz:

$$\begin{aligned} L(\theta) &= p(\vec{y} \mid X; \theta) \\ &= \prod_{i=1}^{m} p(y^{(i)} \mid x^{(i)}; \theta) \\ &= \prod_{i=1}^{m} \left(h_\theta(x^{(i)})\right)^{y^{(i)}} \left(1 - h_\theta(x^{(i)})\right)^{1-y^{(i)}} \end{aligned}$$

log ihtimalini maksimize etmek daha kolaydır. Bu nedenle şöyle yazabiliriz:

$$\begin{aligned} \ell(\theta) &= \log L(\theta) \\ &= \sum_{i=1}^{m} y^{(i)} \log h(x^{(i)}) + (1 - y^{(i)}) \log(1 - h(x^{(i)})) \end{aligned}$$

Olasılığı nasıl maksimize edeceğiz? Doğrusal regresyon örneğimizdeki gibi dereceli azalma/yükselme kullanacağız. Vektörel gösterimle yazılmış olan güncellemelerimiz bu nedenle θ: = θ + α∇θℓ (θ) ile verilecektir. Sadece bir eğitim örneği (x, y) ile çalışarak başlayalım ve stokastik artan kuralını üretmek için türevlerini alalım: Yukarıda, g'(z) = g(z) (1 - g (z)) gerçeğini kullandık. Bu nedenle bize stokasti artan kuralını üretmek için türevlerini alalım:

$$\begin{aligned}\frac{\partial}{\partial\theta_j}\ell(\theta) &= \left(y\frac{1}{g(\theta^T x)} - (1-y)\frac{1}{1-g(\theta^T x)}\right)\frac{\partial}{\partial\theta_j}g(\theta^T x)\\ &= \left(y\frac{1}{g(\theta^T x)} - (1-y)\frac{1}{1-g(\theta^T x)}\right)g(\theta^T x)(1-g(\theta^T x)\frac{\partial}{\partial\theta_j}\theta^T x\\ &= \left(y(1-g(\theta^T x)) - (1-y)g(\theta^T x)\right)x_j\\ &= \left(y - h_\theta(x)\right)x_j\end{aligned}$$

Yukarıda, g'(z) = g(z) (1 - g (z)) gerçeğini kullandık. Bu nedenle bize stokastik dereceli yükselme kuralını veriyor:

$$\theta_j := \theta_j + \alpha\left(y^{(i)} - h_\theta(x^{(i)})\right)x_j^{(i)}$$

Lojistik regresyon, bağımlı değişkenin kategorik bir değişken olduğu regresyon problemi gibidir. Doğrusal sınıflandırma problemlerinde yaygın bir biçimde kullanılır. Regresyon denilmesine rağmen burada bir sınıflandırma söz konusudur. Lojistik regresyon, bir sonucu belirleyen bir veya daha fazla bağımsız değişken bulunan bir veri kümesini analiz etmek için kullanılan istatistiksel bir yöntemdir. Sonuç, ikili bir değişkenle ölçülür (yalnızca iki olası sonuç vardır). Lojistik regresyonda, bağımlı değişken ikili veya ikili, yani yalnızca 1 (DOĞRU, başarı, hamile vb.) Veya 0 (YANLIŞ, hata, gebe olmayan vb.) Olarak kodlanmış verileri içeriyor. Lojistik regresyonun amacı, iki yönlü karakteristiği (bağımlı değişken = yanıt veya sonuç değişkeni) ile ilgili bir dizi bağımsız (öngörücü veya açıklayıcı) değişken arasındaki ilişkiyi tanımlamak için en uygun (henüz biyolojik olarak makul) modeli bulmaktır. Lojistik regresyon, ilgi karakteristiklerinin varlığının olasılığını logit dönüşümünü tahmin etmek için bir formülün katsayılarını (ve standart hatalarını ve önem seviyelerini) üretir:

$$logit(p) = b_0 + b_1X_1 + b_2X_2 + b_3X_3 + \ldots + b_kX_k$$

Burada p, karakteristik özelliğinin var olma olasılığıdır.

$$odds = \frac{p}{1-p} = \frac{probability\ of\ presence\ of\ characteristic}{probability\ of\ absence\ of\ characteristic}$$

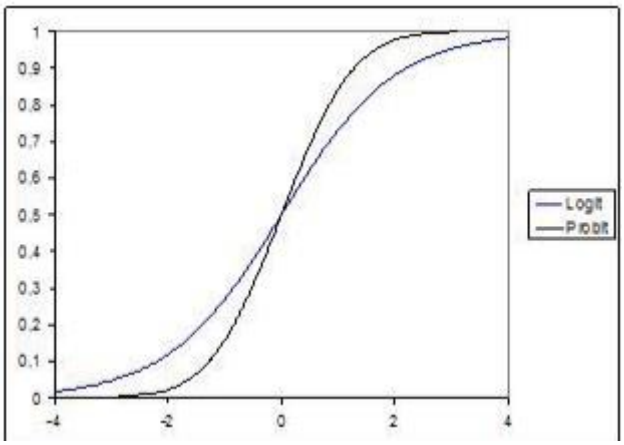
ve

$$logit(p) = \ln\left(\frac{p}{1-p}\right)$$

Karekök hataların toplamını en aza indirgeyen parametreleri seçmek yerine (sıradan regresyon gibi), lojistik regresyonda tahmin, örnek değerlerin gözlem olasılığını en yükseğe çıkaran parametreleri seçer. Bu nedenle parametrelerin belirlenmesinde dereceli yükselme kullanılabilir. Lojistik regresyon, bağımlı değişken ikili (binary) olduğunda yürütülecek uygun regresyon analizidir. Tüm regresyon analizlerinde olduğu gibi, lojistik regresyon da bir tahmini analizdir. Logistik regresyon, veriyi tanımlamak ve bir bağımlı ikili değişken ile bir veya daha fazla nominal, sıra arası, aralık veya oran seviyesinde bağımsız değişkenler arasındaki ilişkiyi açıklamak için kullanılır. Lojistik Regresyonun inceleyebileceği sorun türü: Akciğer kanseri olma ihtimali (evet ya da hayır), kilo ve günde içilen her paket sigara için nasıl değişir? veya Vücut ağırlığı kalori alımı, yağ alımı ve katılımcı yaşı kalp krizi üzerine etkiye sahip mi (evet veya hayır)? gibidir.

Lojistik regresyon, bağımlı değişkenin stokastik bir olay olduğunu varsayar. Örneğin, böcek ilacı öldürme oranını analiz edersek, sonuç öldürür veya öldürmez olur. En dirençli hatalardan biri bile ancak bu iki durumdan biri olabilir; lojistik regresyon böceklerin öldürülme ihtimalini düşünür. Eğer böcek öldürme ihtimali> 0.5 ise ölü kabul edilir, eğer <0.5 ise canlı olarak kabul edilir.

Sonuç değişkeni – 0 ve 1 olarak kodlanmalıdır – tüm öngörüler Covariates kutusuna girilirken (Kategorik değişkenler uygun şekilde modeli kodlanmalıdır) Bağımlı etiketli ilk kutuya yerleştirilir. Bazen lojistik regresyon için logit modeli yerine bir probit modeli kullanılır. Aşağıdaki grafik, farklı değerler (-4,4) için bir logit ve probit modelinin farkını göstermektedir. Her iki model de lojistik regresyonda yaygın olarak kullanılmaktadır ve çoğu durumda bir model her iki fonksiyonla donatılmıştır ve daha iyi uyuma sahip fonksiyon seçilmiştir. Bununla birlikte, probit, günlük dağılımı varsayılarak logit olayının olasılığının normal dağılımını varsaymaktadır. Böylece logit ve probit arasındaki fark genellikle küçük örneklerde görülür.



Lojistik regresyon analizinin merkezinde, bir olayın log oranını tahmin eden görev bulunur.

$$\text{probit}(\pi(x)) = \beta_0 + \beta x$$

Probit modeli:

Şimdi bir lojistik regresyon örneği ile devam edelim. Burada veri seti olarak, AndrewNG'nin Coursera'da kullandığı veri setini kullanacağız. Örneğimizin bu bölümünde, bir öğrencinin bir üniversiteye kabul edilip edilmeyeceğini tahmin etmek için bir lojistik regresyon modeli oluşturacağız.

Bir üniversite bölümünün yöneticisi olduğunuzu ve her başvuru sahibinin iki sınava ilişkin sonuçlarına dayanarak kabul edilme şansını belirlemek istediğinizi varsayalım. Lojistik regresyon için bir eğitim seti olarak kullanabileceğiniz geçmişte başvuranların verilerine sahipsiniz. Her bir eğitim örneği için, başvuranın iki sınava ve kabul kararına da sahipsiniz. Amacımız, başvuranların bu iki sınavın puanlarını temel alarak başvuru olasılığını tahmin eden bir sınıflandırma modeli oluşturmaktır.

calistir.py

```
# coding=utf-8
""" Main
Calistir """

import sys

reload(sys)
sys.setdefaultencoding('utf-8')

import pandas as pd
import numpy as np

from ciz import dagilim_grafigi, hipotez_grafigi
from maliyeti_hesapla import maliyeti_hesapla
from dereceli_fonksiyon import dereceli_fonksiyon
from sigmoid import sigmoid

from scipy import optimize

u""" 1. Veri Setinin Yuklenmesi """
print '1. Veri Seti Yukleniyor ...'
# Veri Setini Txt Dosyasindan Okuma
ex1data1 = pd.read_csv('ex2data1.txt', delimiter=',', header=None,
names=['Exam1', 'Exam2', 'Admission'])

# orjinal veri setini bozmamak icin kopya olusturalim.
veri_seti = ex1data1.copy()

u""" 2. Dagilim Grafigi"""
print '2. Dagilim Grafigi Gosteriliyor ...'
dagilim_grafigi(veri_seti)

# Tahmin Etmeye Calisacagimiz Verileri Okuma
y = veri_seti['Admission'].as_matrix()
del veri_seti['Admission']

# Toplam Girdi Sayisi
m, n = veri_seti.shape

# Ongorucu Veriler, ilk iki degisken: exam1 ve exam2
X_veri = veri_seti.as_matrix()

u""" 3. Kesme Degerinin Eklenmesi"""
print '3. Kesme Degeri Ekleniyor ...'
X = np.ones(shape=(m, (len(X_veri[0]) + 1)))
X[:, 1:(len(X_veri[0]) + 1)] = X_veri
```

```
u""" 4. Tetanin ilk Degeri(initial) Ayarlaniyor ..."""
print '4. Tetanin ilk Degeri(initial) Ayarlaniyor ...'
teta = np.zeros(n + 1)

maliyet = maliyeti_hesapla(teta, X, y)
gradyan = dereceli_fonksiyon(teta, X, y)

print ' Initial tetaya gore maliyet: ', maliyet
print ' Initial tetaya gore gradyan: ', gradyan

u""" 5. scipy ile Optimizasyon Yapiliyor ..."""
print '5. scipy ile Optimizasyon Yapiliyor ...'

result = optimize.minimize(
    lambda teta: maliyeti_hesapla(teta, X, y),
    teta,
    method='Nelder-Mead',
    options={
        'maxiter': 400}
)

teta = result.x
maliyet = result.fun
print ' Maliyet: ', maliyet
print ' Teta :', teta

u""" 6. Hipotez Grafigi Gosteriliyor ..."""
print '6. Hipotez Grafigi Gosteriliyor'
# grafik cizilirken kesme degerini ekledigimiz bolumu cikartarak
cizdiriyoruz
hipotez_grafigi(X[:, 1:], y, teta)

u""" 7. Tahminler Yapalim ..."""
print '7. Tahminler Yapalim ...'
print ' 45 ve 85 notlarini alan bir ogrencinin kabul edilme
olasiligi', sigmoid(np.array([1, 45, 85]).dot(teta))
print ' 50 ve 50 notlarini alan bir ogrencinin kabul edilme
olasiligi', sigmoid(np.array([1, 50, 50]).dot(teta))
print ' 65 ve 70 notlarini alan bir ogrencinin kabul edilme
olasiligi', sigmoid(np.array([1, 65, 70]).dot(teta))

u""" 8. Modelin Basarisi ..."""
print '8. Modelin Basarisi ...'
tahminler = sigmoid(X.dot(teta)) >= 0.5
basari = 100 * np.mean(tahminler == y)
print ' Egitimin Basarisi : ', basari, '%'
```

dereceli_fonksiyon.py

```
# coding=utf-8
""" Gradient Descent
 Dereceli Fonksiyon """

from sigmoid import sigmoid
import numpy as np


def dereceli_fonksiyon(theta, X, y):
    """

    :param theta:
    :param X:
    :param y:
    :return: gradyan
    """
    m = X.shape[0]
    h = sigmoid(X.dot(theta))
    cost = sum(-y * np.log(h) - (1.0 - y) * np.log(1.0 - h))
    grad = X.T.dot(h - y)
    return grad / m
```

maliyeti_hesapla.py

```
# coding=utf-8
""" Compute Cost Function
 Maliyet Fonksiyonunu Hesapla"""

import numpy as np
from sigmoid import sigmoid


def maliyeti_hesapla(theta, X, y):
    """

    :param theta:
    :param X:
    :param y:
    :return: maliyet
    """
    m = X.shape[0]
    h = sigmoid(X.dot(theta))
    cost = sum(-y * np.log(h) - (1.0 - y) * np.log(1.0 - h))
    return cost / m
```

sigmoid.py

```
# coding=utf-8
""" Sigmoid
Sigmoid """

import numpy as np


def sigmoid(z):
    """

    :param z: veri
    :return: sigmoid fonksiyonu

    >> sigmoid(0) == 0.5
    """
    return 1.0 / (1.0 + np.exp(-z))
```

ciz.py

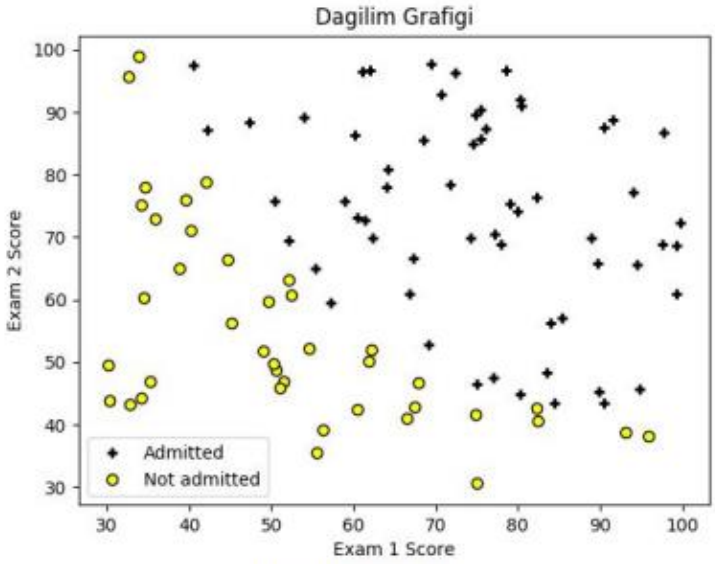
```
# coding=utf-8
""" Plotting Data
Veri Setinin Grafiginin Cizilmesi"""

import matplotlib.pyplot as plt
import numpy as np


def dagilim_grafigi(veri_seti):
    """

    :param veri_seti: Dagilim Grafiginin Yapilacagi Veri Seti
    """

    pos_veri = veri_seti[veri_seti['Admission'] == 1]
    neg_veri = veri_seti[veri_seti['Admission'] == 0]

    plt.figure()

    plt.plot(pos_veri['Exam1'], pos_veri['Exam2'], '+',
markeredgecolor='black', markeredgewidth=2)
    plt.plot(neg_veri['Exam1'], neg_veri['Exam2'], 'o',
markeredgecolor='black', markerfacecolor='yellow')
    plt.xlabel('Exam 1 Score')
    plt.ylabel('Exam 2 Score')
```

```
    plt.title('Dagilim Grafigi')
    plt.legend(['Admitted', 'Not admitted'])
    plt.show()


def hipotez_grafigi(X, y, teta):
    """

    :param X: Ongorucu Degiskenler, Sinav Notlari
    :param y: Hedef Degisken, Kabul Edilip Edilmedigi
    :param teta: Parametreler
    """

    pos_index = y.nonzero()[0]
    neg_index = (y == 0).nonzero()[0]

    plt.plot(X[pos_index, 0], X[pos_index, 1], '+',
markeredgecolor='black', markeredgewidth=2)
    plt.plot(X[neg_index, 0], X[neg_index, 1], 'o',
markeredgecolor='black', markerfacecolor='yellow')
    plt.xlabel('Exam 1 score')
    plt.ylabel('Exam 2 score')

    x_hipotez = np.array([X.min() - 2, X.max() + 2])
    y_hipotez = (-teta[0] - teta[1] * x_hipotez) / teta[2]

    plt.plot(x_hipotez, y_hipotez, color='blue')
    plt.legend(['Admitted', 'Not admitted'])
    plt.show()
```

Programın Çıktısı:

```
1. Veri Seti Yukleniyor ...
2. Dagilim Grafigi Gosteriliyor ...
3. Kesme Degeri Ekleniyor ...
4. Tetanin ilk Degeri(initial) Ayarlaniyor ...
 Initial tetaya gore maliyet:  0.69314718056
 Initial tetaya gore gradyan:  [ -0.1        -12.00921659
-11.26284221]
5. scipy ile Optimizasyon Yapiliyor ...
 Maliyet:  0.20349770159
 Teta : [-25.16130062   0.20623142   0.20147143]
6. Hipotez Grafigi Gosteriliyor
7. Tahminler Yapalim ...
 45 ve 85 notlarini alan bir ogrencinin kabul edilme olasiligi
0.776291590411
 50 ve 50 notlarini alan bir ogrencinin kabul edilme olasiligi
```

```
0.00835787437461
 65 ve 70 notlarini alan bir ogrencinin kabul edilme olasiligi
0.91267489481
8. Modelin Basarisi ...
 Egitimin Basarisi :  89.0 %
```




Hipotez Grafiği :