Hafta 02 - Çok Değişkenli Regresyon

BGM 565 - Siber Güvenlik için Makine Öğrenme Yöntemleri

Bilgi Güvenliği Mühendisliği
Yüksek Lisans Programı

Dr. Ferhat Özgür Çatak
ozgur.catak@tubitak.gov.tr

İstanbul Şehir Üniversitesi
2018 - Bahar

İçindekiler

İçindekiler

Çok Değişkenli Doğrusal Regresyon I
Multivariate Linear Regression

Tek Değişkenli Doğrusal Regresyon

$$h(x) = w_0 + w_1 x \tag{1}$$

Çok Değişkenli Doğrusal Regresyon

$$\begin{aligned} h(\mathbf{x}) &= w_0 + w_1 x_1 + \cdots + w_n x_n \\ y &= w_0 + w_1 x_1 + \cdots + w_n x_n \end{aligned} \tag{2}$$

y	Bağımlı değişken
$x_1, \cdots, x_n$	Bağımsız değişkenler
w_0	Sabit
$w_1, \cdots, w_n$	Katsayı

Çok Değişkenli Doğrusal Regresyon II
Multivariate Linear Regression

Yeni Değişken

x_0	x_1	x_2	x_3	x_4	y
1.00	0.54	0.17	0.93	0.58	3.74
1.00	0.85	0.35	0.84	0.45	4.55
1.00	0.97	0.74	0.44	0.30	5.24
1.00	0.62	0.68	0.67	0.98	5.92
1.00	0.59	0.88	0.09	0.89	5.75
1.00	0.66	0.83	0.92	0.82	6.43
1.00	0.64	0.04	0.82	0.84	3.91
1.00	0.85	0.83	0.95	0.07	5.31
1.00	0.74	0.16	0.71	0.57	3.89
1.00	0.32	0.33	0.13	0.59	3.02

Eski hipotez: $h(x) = w_0 + w_1 x$
Yeni hipotez: $h(\mathbf{x}) = w_0 x_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4$
Örnek: $h(\mathbf{x}) = 0.2 + 2x_1 + 0.1x_2 + 0.5x_3 + 1.2x_4$

Çok Değişkenli Doğrusal Regresyon III

Multivariate Linear Regression

Hipotez: $h(\mathbf{x}) = \underbrace{w_0 x_0}_{x_0=1} + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$

$$\mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1}, \quad \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{bmatrix} \in \mathbb{R}^{n+1} \tag{3}$$

$$\underbrace{\begin{bmatrix} w_0 & w_1 & \cdots & w_n \end{bmatrix}}_{\mathbf{w}^T} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$$

$$\begin{aligned} h(\mathbf{x}) &= w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n \\ &= \mathbf{w}^T \mathbf{x} \end{aligned} \tag{4}$$

Çok Değişkenli Doğrusal Regresyon IV

Multivariate Linear Regression

Hipotez: $h(\mathbf{x}) = w_0 x_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$

Maliyet Fonksiyonu: $C(w_0, w_1, \cdots, w_n) = \frac{1}{2m} \sum_{i=1}^{m} \left(h(\mathbf{x^{(i)}}) - y^{(i)} \right)^2$

Tekrarla {

$w_j = w_j - \alpha \frac{\partial}{\partial w_j} C(w_0, w_1, \cdots, w_n)$

}

Yeni Algoritma (Gradient Descent)

$$w_0 = w_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h(\mathbf{x}^{(i)}) - y^{(i)} \right) x_0^{(i)}$$

$$w_1 = w_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h(\mathbf{x}^{(i)}) - y^{(i)} \right) x_1^{(i)}$$

$$w_2 = w_2 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h(\mathbf{x}^{(i)}) - y^{(i)} \right) x_2^{(i)}$$

$$\cdots$$

$$w_n = w_n - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h(\mathbf{x}^{(i)}) - y^{(i)} \right) x_n^{(i)}$$

Çok Değişkenli Doğrusal Regresyon V
Multivariate Linear Regression

Genel Gösterim

tekrarla {

$$w_j = w_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h(\mathbf{x}^{(i)}) - y^{(i)} \right) \cdot x_j^{(i)}$$

}

Gradient Descent - Matris Gösterimi

Gradient descent kuralı:

$$\mathbf{w} = \mathbf{w} - \alpha \nabla C(\mathbf{w}) \tag{5}$$

$\nabla C(\mathbf{w})$ ifadesi kolon vektörü şeklinde gösterilebilir.

$$\nabla C(\mathbf{w}) = \begin{bmatrix} \frac{\partial C(w)}{\partial w_0} \\ \frac{\partial C(w)}{\partial w_1} \\ \vdots \\ \frac{\partial C(w)}{\partial w_n} \end{bmatrix}$$

$$\begin{aligned} \frac{\partial C(w)}{\partial w_j} &= \frac{1}{m} \sum_{i=1}^{m} \left(h(\mathbf{x}^{(i)}) - y^{(i)} \right) \cdot \mathbf{x}_j^{(i)} \\ &= \frac{1}{m} \sum_{i=1}^{m} \mathbf{x}_j^{(i)} \cdot \left(h(\mathbf{x}^{(i)}) - y^{(i)} \right) \qquad (6) \\ &= \frac{1}{m} \mathbf{x}_j^T (X\mathbf{w} - \mathbf{y}) \end{aligned}$$

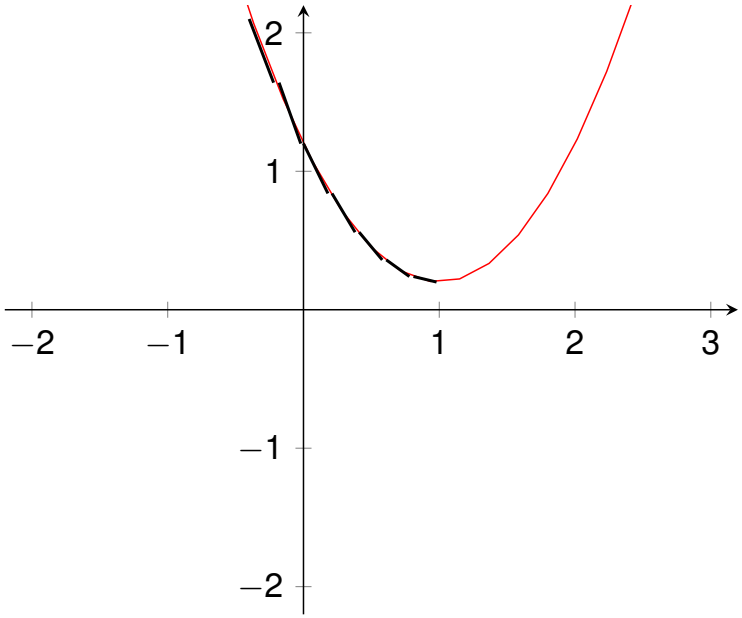
$$\nabla C(\mathbf{w}) = \frac{1}{m} X^T (X\mathbf{w} - \mathbf{y}) \tag{7}$$

Gradient descent kuralının matris gösterimi:

$$\mathbf{w} = \mathbf{w} - \frac{\alpha}{m} X^T (X\mathbf{w} - \mathbf{y}) \tag{8}$$

Normal Denklem I
Normal Equation

Gradient Descent



Normal Equations

- Gradient descent, *C*'yi minimize etmek için çözümlerden biri
- Alternatif *Yinelemeli olmayan (non-iterative)* yöntem: **Normal denklem (Normal equation)**

$$\mathbf{w} = \left(X^T X\right)^{-1} X^T y \tag{9}$$

Normal Denklem II
Normal Equation

Doğrusal Regresyon İçin Normal Denklemin Türetilmesi

Hipotez fonksiyonu: $h(\mathbf{x}) = w_0 x_0 + w_1 x_1 + \cdots + w_n x_n$

Maliyet fonksiyonu: $C(\mathbf{w}) = \frac{1}{2m} \sum_{i=1}^{m} \left(h(\mathbf{x}^i) - y^i \right)^2$

Hipotez: $h(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$

Maliyet:

$$\begin{aligned} h(\mathbf{w}) &= \frac{1}{2m} (X\mathbf{w} - y)^T (X\mathbf{w} - y) \\ &= \left((X\mathbf{w})^T - y^T \right) (X\mathbf{w} - y) \\ &= (X\mathbf{w})^T X\mathbf{w} - (X\mathbf{w})^T y - y^T (X\mathbf{w}) + y^T y \\ &= \mathbf{w}^T X^T X\mathbf{w} - 2(X\mathbf{w})^T y + y^T y \end{aligned} \tag{10}$$

$$\begin{aligned} \frac{\partial C}{\partial \mathbf{w}} &= 2X^T X\mathbf{w} - 2X^T y = 0 \\ X^T X\mathbf{w} &= X^T y \end{aligned} \tag{11}$$

Eşitliğin her iki tarafı $(X^T X)^{-1}$ ile çarpılırsa

$$\mathbf{w} = (X^T X)^{-1} X^T y \tag{12}$$

Normal Denklem III
Normal Equation

x_1	x_2	x_3	x_4	y
0,54	0,17	0,93	0,58	3,74
0,85	0,35	0,84	0,45	4,55
0,97	0,74	0,44	0,30	5,24
0,62	0,68	0,67	0,98	5,92
0,59	0,88	0,09	0,89	5,75
0,66	0,83	0,92	0,82	6,43
0,64	0,04	0,82	0,84	3,91
0,85	0,83	0,95	0,07	5,31
0,74	0,16	0,71	0,57	3,89
0,32	0,33	0,13	0,59	3,02

$$X = \begin{bmatrix} 1,00 & 0,54 & 0,17 & 0,93 & 0,58 \\ 1,00 & 0,85 & 0,35 & 0,84 & 0,45 \\ 1,00 & 0,97 & 0,74 & 0,44 & 0,30 \\ 1,00 & 0,62 & 0,68 & 0,67 & 0,98 \\ 1,00 & 0,59 & 0,88 & 0,09 & 0,89 \\ 1,00 & 0,66 & 0,83 & 0,92 & 0,82 \\ 1,00 & 0,64 & 0,04 & 0,82 & 0,84 \\ 1,00 & 0,85 & 0,83 & 0,95 & 0,07 \\ 1,00 & 0,74 & 0,16 & 0,71 & 0,57 \\ 1,00 & 0,32 & 0,33 & 0,13 & 0,59 \end{bmatrix} \in \mathbb{R}^{10\times 5} \quad \mathbf{y} = \begin{bmatrix} 3,74 \\ 4,55 \\ 5,24 \\ 5,92 \\ 5,75 \\ 6,43 \\ 3,91 \\ 5,31 \\ 3,89 \\ 3,02 \end{bmatrix} \in \mathbb{R}^{10} \tag{13}$$

$$\mathbf{w} = \left(X^T X\right)^{-1} X^T \mathbf{y} \Rightarrow \mathbf{w} = [0.12490622, 1.9516536, 2.98882317, 0.97638019, 1.96358802]$$

$$h(\mathbf{x}) = 0.12490622 + 1.9516536 \cdot x_1 + 2.98882317 \cdot x_2 + 0.97638019 \cdot x_3 + 1.96358802 \cdot x_4$$

Normal Denklem IV
Normal Equation

Gradient Descent

- α değeri seçilmesi gerekli
- Yineleme sayısı oldukça fazla
- Yüksek boyutlu veri kümeleri için oldukça uygun. (Kolon sayısı yüksek)

Normal Equation

- Parametrik değil
- Yineleme yok
- Yüksek boyutlu veri kümeleri için uygun değil, $\left(X^T X\right)^{-1}$ karmaşıklık $O(n^3)$

Normal Denklem V
Normal Equation

```
import pandas as pd
import numpy as np

# veri kumesini oku
verikumesi = pd.read_csv("ds2.txt",delimiter="\t")
verikumesi.insert(loc=0, column='x0', value=1)

X = verikumesi.iloc[:,:-1].values
y = verikumesi.iloc[:,X.shape[1]].values

# Normal equation
tmp = np.linalg.inv(np.matmul(X.T,X))
w = np.dot(np.matmul(tmp,X.T),y)

print(w)
# [2.06239085 2.99213354 0.98455834 2.02928992]

y_pred = np.matmul(X,w.T)
df = pd.DataFrame({"y":y,"y_pred":y_pred})
print(df)
```

İçindekiler

Regularized Linear Regression I

Regularization

- Model karmaşıklığının azaltılması için kullanılır.
- Aşırı öğrenme (ezberleme, overfitting) probleminin çözümünde kullanılır.
  - Oluşturulan modelin eğitim veri kümesine oldukça uyumlu fakat yeni örneklerde hatalı sonuçlar vermesi
- **Çözüm**: oluşturulan hipotezin kullanacağı bazı ağırlıkların etkisinin azaltılması
- $h(\mathbf{x}) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3$. Örnek olarak w_2 ve w_3 etkisini azaltmak istiyorsak (0'a yaklaşmaları), C'ye bir bileşen eklenebilir.

Regularization

Hata karelerinin toplamı + λ * model karmaşıklık cezası

Regularized Linear Regression II

$$C(w_0, w_1, \cdots, w_n) = \frac{1}{2m} \sum_{i=1}^{m} \left(h(\mathbf{x^i}) - y^{(i)} \right) \tag{14}$$

$$C(w_0, w_1, \cdots, w_n) = \frac{1}{2m} \left(\sum_{i=1}^{m} \left(h(\mathbf{x^i}) - y^{(i)} \right) + \lambda \sum_{j=1}^{n} w_j^2 \right) \tag{15}$$

λ: Regularization parametresi

Regularized Gradient Descent

tekrarla {

$$
\begin{aligned}
w_0 &= w_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h(\mathbf{x^{(i)}}) - y^{(i)} \right)^2 x_0^{(i)} \\
w_j &= w_j - \alpha \left[\left(\frac{1}{m} \sum_{i=1}^{m} \left(h(\mathbf{x^{(i)}}) - y^{(i)} \right)^2 x_0^{(i)} \right) + \frac{\lambda}{m} w_j \right]
\end{aligned}
\tag{16}
$$

}

Regularization Yaklaşımları I

Regularization Yaklaşımları

- L2-Regularization (**Ridge**)

$$C(w_0, w_1, \cdots, w_n) = \frac{1}{2m}\left(\sum_{i=1}^{m}\left(h(\mathbf{x^i}) - y^{(i)}\right)^2 + \lambda||\mathbf{w}||_2\right)$$

- L1-Regularization (**Lasso**)

$$C(w_0, w_1, \cdots, w_n) = \frac{1}{2m}\left(\sum_{i=1}^{m}\left(h(\mathbf{x^i}) - y^{(i)}\right)^2 + \lambda||\mathbf{w}||\right)$$

Norm - Başlangıç noktasına (Orjin) olan uzaklık

- Mutlak değer norm (Absolute Norm):
  $||\mathbf{w}|| = \sum_{i=1}^{N} |w_i|$
- Öklid Normu (Euclidean Norm):
  $||\mathbf{w}||_2 = \sqrt{\left[\sum_{i=1}^{N} |w_k|^2\right]}$
- Genel Vektör Normu (General Vector Norm):
  $||\mathbf{w}||_p = \left[\sum_{i=1}^{N} |w_k|^p\right]^{\frac{1}{p}}$
- $w = [-2, 3, -1] \Rightarrow$
  $||\mathbf{w}||_2 = 3.7417, ||\mathbf{w}||_1 = 6$

Regularization Yaklaşımları II

$L1$ vs $L2$

- Örnek model 1: $y = 1 \times x_1 + 1 \times x_2$ bu durumda
  - $L1 = (1 + 1) \times \lambda = 2 \times \lambda$
  - $L2 = (1^2 + 1^2) \times \lambda = 2 \times \lambda$
- Örnek model 2: $y = 2 \times x_1 + 0 \times x_2$ bu durumda
  - $L1 = (2 + 0) \times \lambda = 2 \times \lambda$
  - $L2 = (2^2 + 0^2) \times \lambda = 4 \times \lambda$
- $L1$ regularization uygulandığı zaman özellik katsayılarından bazıları daha fazla 0 olmaya başlar.
  - $L1$ ceza yöntemi veri kümelerinden **nitelik seçimi** için daha uygundur (Sparse solution).
-

Python I

Scikit-learn Regresyon

```
class sklearn.linear_model.SGDRegressor(loss='squared_loss', penalty='l2',
alpha=0.0001, l1_ratio=0.15, fit_intercept=True, max_iter=None, tol=None,
shuffle=True, verbose=0, epsilon=0.1, random_state=None,
learning_rate='invscaling', eta0=0.01, power_t=0.25, warm_start=False,
average=False, n_iter=None)
```

- **loss**: squared_loss
- **penalty**='l2'
- **alpha**(regularization term)
- **max_iter**
- **tol**
- **learning_rate**

Python II

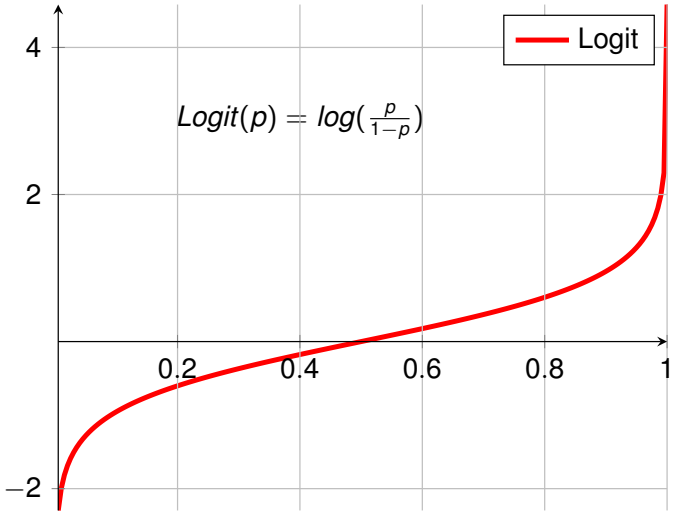
```
import numpy as np
import pandas as pd
from sklearn.linear_model import SGDRegressor

# veri kumesini oku
verikumesi = pd.read_csv("ds2.txt",delimiter="\t")

X = verikumesi.iloc[:,:-1].values
y = verikumesi.iloc[:,X.shape[1]].values

# modeli tanimla
clf = SGDRegressor(penalty='none', verbose=1, max_iter=100000)
# modeli egit
clf.fit(X, y)

print(clf.intercept_, clf.coef_)
# Grad. Dc.: [ 0.14157558  1.91993045  2.99348001  0.98027489  1.94982636]
# Norm. Eq.: [ 0.12490622  2.06239085  2.99213354  0.98455834  2.02928992]
```

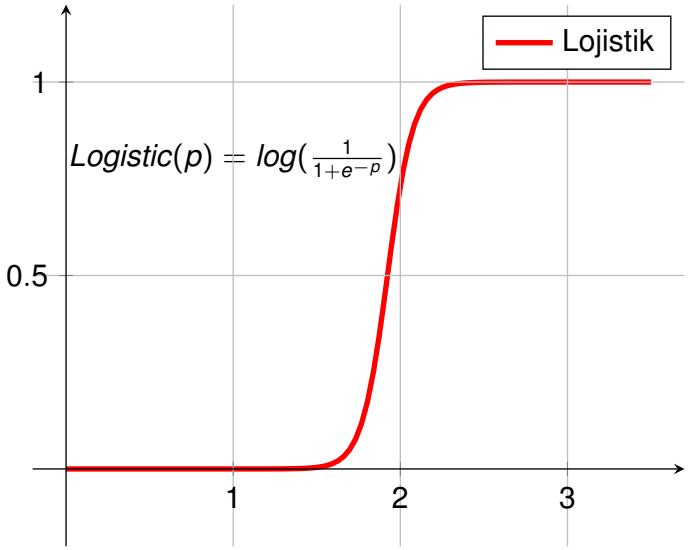
İçindekiler

Lojistik Regresyon I

Logistic Regression, Logit Regression

- Model $P(y = 1|\mathbf{x})$: doğrusal fonksiyon?
  - **Problem**: Olasılık $P(y = 1|\mathbf{x})$ doğrusal model olamaz. $P(y = 1|\mathbf{x})$ 0 ve 1, $[0, 1]$ aralığında olmalıdır.
  - **x**'in değişiminin sonuçları, olasılık aralığında $[0, 1]$ sabit olmalıdır.
  - Eğer $P(y = 1|\mathbf{x})$ sonucu +1 veya 0'a yakınsa, **x** değişiminin y etkisi fazla olmalıdır.
- **Çözüm**: **Logit transformation**





Lojistik Regresyon II

Logistic Regression, Logit Regression

Hatırlatma: Euler sayısı (e)

- $e = 2.71828182845904523536028747135 27$
- Matematik, mühendislikte sık kullanılan sabit

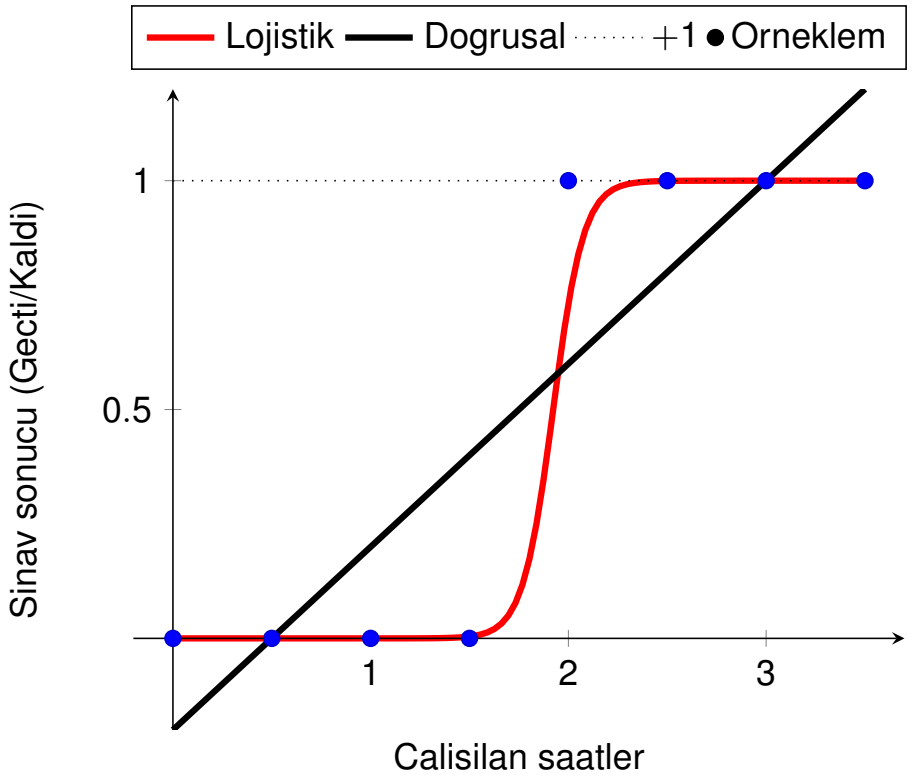
$$e = \lim_{n \to \infty} \left(1 + \frac{1}{n}\right)^n \tag{17}$$

n	$(1 + 1/n)^n$
1	2,00000
2	2,25000
5	2,48832
10	2,59374
100	2,70481
1.000	2,71692
10.000	2,71815
100.000	2,71827

Lojistik Regresyon III

Logistic Regression, Logit Regression

- Bağımlı değişkenin kategorik olduğu regresyon modelidir.
  - Ürün satın alındı/alınmadı
  - E-posta cevabı alındı/alınmadı
  - Hastalık var/yok



Lojistik Regresyon IV

Logistic Regression, Logit Regression

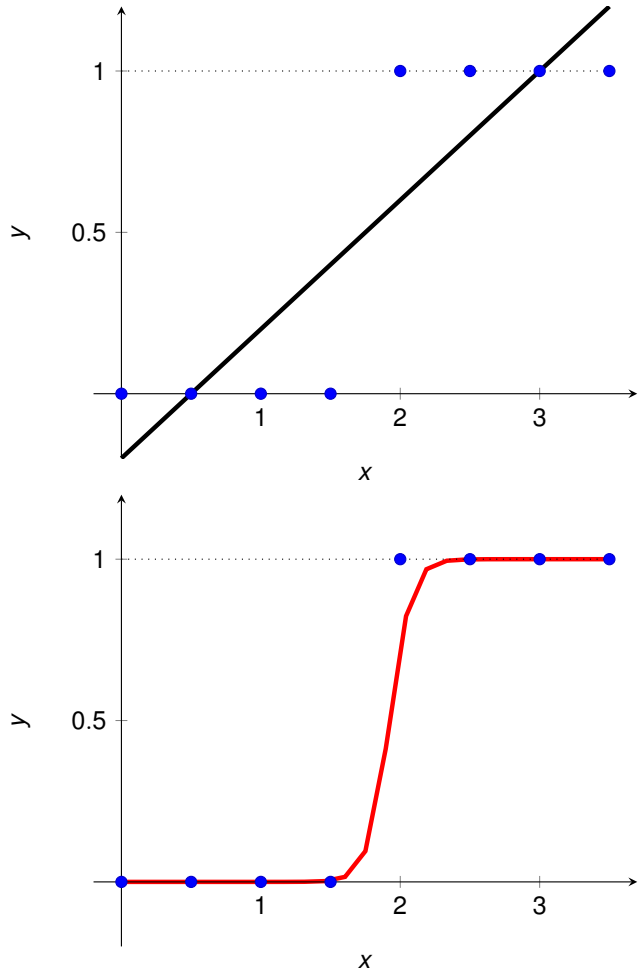
Doğrusal Regresyon:

$$h(\mathbf{x}) = w_0 + w_1 x_1 + \cdots + w_n x_n \quad (18)$$

Sigmoid fonksiyonu:

$$p = \frac{1}{1 + e^{-h(\mathbf{x})}} \quad (19)$$

$$ln(\frac{p}{1-p}) = w_0 + w_1 x_1 + \cdots + w_n x_n \quad (20)$$



Lojistik Regresyon V

Logistic Regression, Logit Regression

- Bağımlı değişken $y \in \{0, 1\}$,
  - Negatif sınıf etiketine sahip olan örnekler için 0, pozitif sınıf etiketine sahip örnekler için 1 gösterilecektir.
- Oluşturulacak olan sınıflandırma modeli şu şartı yerine getirmelidir: $0 \leq h(\mathbf{x}) \leq 1$

- Ayrık 0-1 sınıflandırmasını elde etmek için

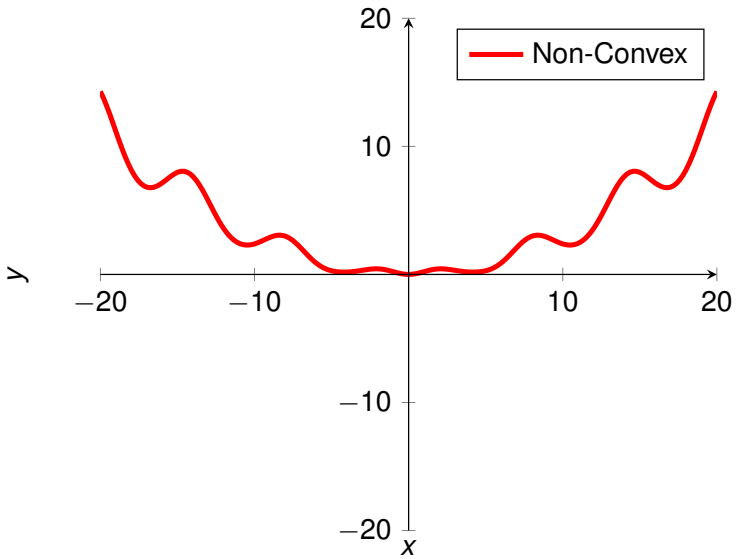
$$h(\mathbf{x}) \geq 0.5 \rightarrow y = 1$$

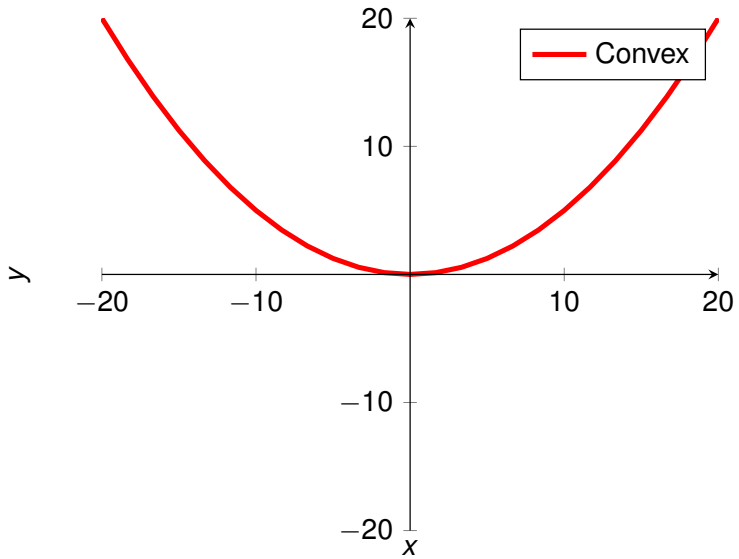
$$h(\mathbf{x}) < 0.5 \rightarrow y = 0$$

Maliyet Fonksiyonu I
Cost Function

Maliyet Fonksiyonu

- Doğrusal regresyon için kullanılan çözüm, lojistik regresyon için uyumlu olmayacaktır.
- Dış bükey (Convex) fonksiyon olmaması sebebiyle birden fazla lokal minimum noktası bulunmaktadır. Bu nedenle hatalı sonuçlara neden olabilmektedir.





Maliyet Fonksiyonu II
Cost Function

Dogrusal regresyon maliyet fonksiyonu:

$$C(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}\left(h(x^{(i)}) - y^{(i)}\right)^2 \tag{21}$$

Bu maliyet fonksiyonu içerisinden $\frac{1}{2}\left(h(x^{(i)}) - y^{(i)}\right)^2$ değiştirilsin

$$\begin{aligned} &\frac{1}{2}\left(h(x^{(i)}) - y^{(i)}\right)^2 \Rightarrow Loss(h(\mathbf{x}), y) \\ &C(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m} Loss(h(\mathbf{x}), y) \end{aligned} \tag{22}$$

Maliyet Fonksiyonu III
Cost Function

$Cost(h(\mathbf{x}), y)$ ifadesinin konveks olması için:

$$Loss(h(\mathbf{x}), y) = \begin{cases} -\log(h(\mathbf{x})), & \text{if } y = 1. \\ -\log(1 - h(\mathbf{x})), & \text{if } y = 0. \end{cases} \tag{23}$$

- Özellikler
  - Eğer $h(\mathbf{x}) = y$, $Loss(h(\mathbf{x}), y) = 0$
  - Eğer $y = 0$ ve $h(\mathbf{x}) \rightarrow 1$ ise $Loss(h(\mathbf{x}), y) \rightarrow \infty$
  - Eğer $y = 1$ ve $h(\mathbf{x}) \rightarrow 0$ ise $Loss(h(\mathbf{x}), y) \rightarrow \infty$

Gradient descent'e daha elverişli bir biçimde maliyeti yeniden yazabiliriz:

$$Loss(h(\mathbf{x}), y) = -y \log\left(h(\mathbf{x})\right) - (1 - y) \log\left(1 - h(\mathbf{x})\right) \tag{24}$$

Bu durumda tüm maliyet fonksiyonumuz:

$$C(\mathbf{w}) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log\left(h(\mathbf{x}^{(i)})\right) + (1 - y^{(i)}) \log\left(1 - h(\mathbf{x}^{(i)})\right)\right] \tag{25}$$

Maliyet Fonksiyonu IV
Cost Function

Konveks

- Lojistik regresyon için diğer maliyet fonksiyonlarını kullanabilir
- Ancak bu maksimum olasılık tahmini (maximum likelihood estimation) ilkesinden türetilir ve konveks olma özelliğine sahiptir
- Bu nedenle bu temel olarak lojistik regresyon için kullandığı bir maliyet fonksiyonudur.

Gradient Descent

$minJ(\mathbf{w})$ hesaplamak için
tekrarla $\{$
$w_j = w_j - \frac{\alpha}{m} \sum_{i=1}^{m} \left(h(\mathbf{x}^{(i)}) - y^i \right) \cdot x_j^i$
$\}$
Doğrusal regresyon için kullanılan gradient descent algoritması ile aynıdır.
Fakat $h(\mathbf{x})$ artık doğrusal değildir $h(\mathbf{x}) = \frac{1}{1+e^{\mathbf{w}^T \mathbf{x}}}$.

Python I

Scikit-learn Lojistik Regresyon

```
class sklearn.linear_model.LogisticRegression
```

- **penalty**='l2'
- **C**: Regularization strength
- **max_iter**
- **tol**

Python II

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression

# veri kumesini oku
verikumesi = pd.read_csv("ds_logreg.txt",delimiter="\t")

X = verikumesi.iloc[:,:-1].values
y = verikumesi.iloc[:,X.shape[1]].values

# modeli tanimla
clf = LogisticRegression(verbose=1)
# modeli egit
clf.fit(X, y)

print(clf.intercept_, clf.coef_)
```

İçindekiler

KDDCUP'99 Veri Kumesi I

KDDCUP'99 Veri Kümesi

- 1999 yılında bir konferansta (The Fifth International Conference on Knowledge Discovery and Data Mining - KDD) yapılan, Bilgi Çıkarımı ve Veri Madenciliği Araçları Yarışmasında (International Knowledge Discovery and Data Mining Tools Competition) kullanılan veri kümesi.
- Amaç: "kötü" bağlantıları saldırılar ve "iyi" bağlantıları normal olarak ayırt edebilen tahmin modeli olan bir IDS (Intrusion Detection System) oluşturmaktı.
- Lab ortamında gerçekleştirilmiş ve birçok saldırının simüle edilmiş halinin kayıt altına alınmasıyla oluşturulmuştur.
- İçerdiği saldırılar 4 ana kategoriye ayırlmaktadır:
  - **DOS**: denial-of-service, örn. syn flood;
  - **R2L**: unauthorized access from a remote machine, örn. guessing password;
  - **U2R**: unauthorized access to local superuser (root) privileges, örn., various "buffer overflow" attacks;
  - **Probing**: surveillance and other probing, örn., port scanning.

KDDCUP'99 Veri Kumesi II

Table: TCP bağlantılarının temel özellikleri.

Feature Name	Description	Type
duration	length (number of seconds) of the connection	continuous
protocol_type	type of the protocol, e.g. tcp, udp, etc.	discrete
service	network service on the destination, e.g., http, telnet, etc.	discrete
src_bytes	number of data bytes from source to destination	continuous
dst_bytes	number of data bytes from destination to source	continuous
flag	normal or error status of the connection	discrete
land	1 if connection is from/to the same host/port; 0 otherwise	discrete
wrong_fragment	number of "wrong" fragments	continuous
urgent	number of urgent packets	continuous

KDDCUP'99 Veri Kumesi III

Table: Alan bilgisi ile önerilen bir bağlantı içindeki içerik özellikleri.

Feature Name	Description	Type
hot	number of "hot" indicators	continuous
num_failed_logins	number of failed login attempts	continuous
logged_in	1 if successfully logged in; 0 otherwise	discrete
num_compromised	number of "compromised" conditions	continuous
root_shell	1 if root shell is obtained; 0 otherwise	discrete
su_attempted	1 if "su root" command attempted; 0 otherwise	discrete
num_root	number of "root" accesses	continuous
num_file_creations	number of file creation operations	continuous
num_shells	number of shell prompts	continuous
num_access_files	number of operations on access control files	continuous
num_outbound_cmds	number of outbound commands in an ftp session	continuous
is_hot_login	1 if the login belongs to the "hot" list; 0 otherwise	discrete
is_guest_login	1 if the login is a "guest"login; 0 otherwise	discrete

KDDCUP'99 Veri Kumesi IV

Table: İki saniyelik bir zaman aralığı kullanılarak hesaplanan trafik özellikleri.

Feature Name	Description	Type
count	number of connections to the same host as the current connection in the past two seconds	continuous
serror_rate	% of connections that have "SYN" errors	continuous
rerror_rate	% of connections that have "REJ" errors	continuous
same_srv_rate	% of connections to the same service	continuous
diff_srv_rate	% of connections to different services	continuous
srv_count	number of connections to the same service as the current connection in the past two seconds	continuous
srv_serror_rate	% of connections that have "SYN" errors	continuous
srv_rerror_rate	% of connections that have "REJ" errors	continuous
srv_diff_host_rate	% of connections to different hosts	continuous

KDDCUP'99 Veri Kumesi V

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split

# veri kumesini oku
kolon_adlari = ['duration','protocol_type','service','flag','src_bytes','dst_bytes','land','w
'hot','num_failed_logins','logged_in','num_compromised','root_shell','su_attempted','num_root
'num_shells','num_access_files','num_outbound_cmds','is_host_login','is_guest_login','count',
'serror_rate','srv_serror_rate','rerror_rate','srv_rerror_rate','same_srv_rate','diff_srv_rate
'dst_host_count','dst_host_srv_count','dst_host_same_srv_rate','dst_host_diff_srv_rate','dst_h
'dst_host_srv_diff_host_rate','dst_host_serror_rate','dst_host_srv_serror_rate','dst_host_rer
'dst_host_srv_rerror_rate','label']

verikumesi = pd.read_csv("kddcup99.tar.gz",compression="gzip", names=kolon_adlari,
low_memory=False, skiprows=1)

# ilgili kolonlari sec
secilecek_kolonlar = ['duration','src_bytes','dst_bytes','wrong_fragment','urgent','hot','num_
'num_compromised','root_shell','su_attempted','num_root','num_file_creations','num_shells',
'num_access_files','num_outbound_cmds','count','srv_count','serror_rate','srv_serror_rate',
'rerror_rate','srv_rerror_rate','same_srv_rate','diff_srv_rate','srv_diff_host_rate','dst_host
'dst_host_srv_count','dst_host_same_srv_rate','dst_host_diff_srv_rate','dst_host_same_src_port
'dst_host_srv_diff_host_rate','dst_host_serror_rate','dst_host_srv_serror_rate','dst_host_rer
'dst_host_srv_rerror_rate']
X = verikumesi[secilecek_kolonlar].as_matrix()
y = verikumesi['label'].apply(lambda d:0 if d == 'normal.' else 1).as_matrix()
```

KDDCUP'99 Veri Kumesi VI

```
# Egitim ve test veri kumeleri olustur
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33)

# modeli tanimla
clf = LogisticRegression(verbose=0)
# modeli egit
clf.fit(X_train, y_train)

# confusion matrix
y_hat = clf.predict(X_test)
cm = confusion_matrix(y_test,y_hat)

print(cm)
```