Logistic Regression

Logistik regresyon, verileri ayrık sonuçlara ayırmak için kullanılan bir yöntemdir. Örneğin; bir e-postayı spam olarak veya spam değil olarak sınıflandırmak için loğistik regresyonu kullanırız. Bir başka örnek ise bir ticaret sitesinde kullanıcıların çalıntı kredi kartı kullanıp kullanmadığını anlayan bir sistem.

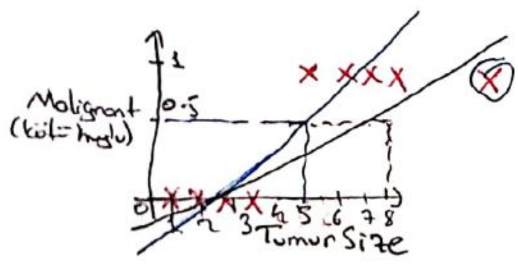
Classification

İsminden de anlaşılacağı üzere belli verileri kümelemek (sınıflandırmak) Tümörün iyi huylu olup olmadığı bir sınıflandırma örneğidir.

Basit sınıflandırmada y değişkenimiz 0 ya da 1'dir. Mesela tümör iyi huyludur veya kötü huyludur. Mail spamdır ya da değildir. Tabi gelişmiş sınıflandırma problemlerinde 0,1,2,3,4,... olabilir.

$y \in \{0,1\}$ 0: "Negative Class" 1: "Positive Class" olarakta adlandırılır.

0 ve 1'e değer atamak (spam veya değil) genelde 0 olumlu durumken 1 olumsuz durumdur. Ama bunun bir önemi yok size kalmış bir durum.



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

Threshold classifier output $h_\theta(x)$ at 0.5:

if $h_\theta(x) \geq 0.5$, predict "y=1"

if $h_\theta(x) \leq 0.5$, predict "y=0"

Yukarıda görüldüğü gibi sınıflandırma için lineer regresyon kullanabiliriz 0.5've üstü kötü huylu, altı iyi huylu diye sınıflandırabiliriz. Ancak bu yöntem iyi çalışmaz. Çünkü yukarıda iki doğru var gördüğünüz gibi mavi doğru yerleri içine alınmış çerpı yok sayılarak çizilmiş. Bu durumda 5 tümör boyutunun sol tarafı iyi huylu iken sağ tarafı kötü huyluyu gösteriyor. Sağ üst tarafa bir veri koyduğumuzda doğrumuz değişiyor ve siyah doğru oluyor. Bu durumda 8 tümör boyutunden sol tarafı iyi huylu, sağ tarafı kötü huylu oluyor. Bu da bizim asıl kötü huylu tümörlerimizin saptanamamasına neden olur. Bu niye oldu tek bir verinin modeli değiştirmesi yüzünden oldu bu yüzden lineer regresyon kullanılmamalı. Başka bir nedende y çıktılarının $0 < \overset{h_\theta(x)}{y} < 1$ arasında değer vermeme ihtimali. Çok büyük değerler verebilir ve bunu normalizasyon ile çözemeyiz.

Lojistik Regresyon Hipotez Tanımı

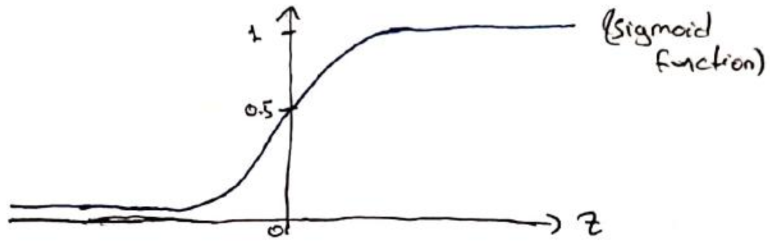
Hipotez olarak kullanacağımız fonksiyon sigmoid function veya logistic function olarak geçmektedir.

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

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

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

$z \in \mathbb{R}$



Grafiği yorumlarsak z değeri $-\infty$ yaklaştıkça sigmoid fonksiyonu 0'a yakınsıyor. z değeri $+\infty$ yaklaştıkça 1'e yakınsıyor. Bu sayede $h_\theta(x)$ çıktımızı $0 \le h_\theta(x) \le 1$ arasında tutmuş oluyoruz.

Not: Formülde görüldüğü gibi θ'ya ihtiyacımız var. Bunu bulmalıyız.

Örn:

$$x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ tumorSize \end{bmatrix}$$

$h_\theta(x) = 0.7$ $\qquad h_\theta(x) = P(y=1 \mid x;\theta)$

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

Yukarıdaki örnek belli bir tümor boyutuna göre fonksiyonumuz 0.7 sonucunu veriyor bu da bize hastamızın tümörünün %70 olasılıkla kötü huylu olduğunu gösteriyor. İyi huylu olma olasılığı ise $1-0.7=0.3$'tür. Yani %30 iyi huylu.

Desicion Boundary (Karar Sınırı)

$h_\theta(x) \ge 0,5 \rightarrow y=1$

$h_\theta(x) < 0,5 \rightarrow y=0$

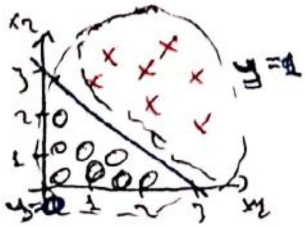
y=0 ve y=1 olanlarının nerede olduğunu ayıran çizgidir. Karar sınırı, veri kümesi ile ilgili değil hipotezin θ parametreleri ile ilgilidir.

$h_\theta(x) = g(\theta^T x) \ge 0.5$

when $\theta^T x \ge 0$

$\theta^T x \ge 0$ ise $y=1$

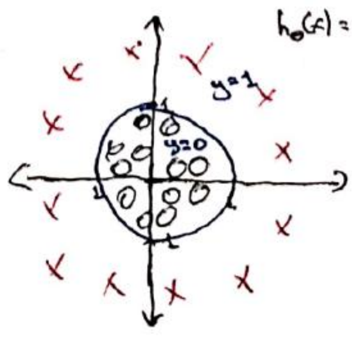
$\theta^T x < 0$ ise $y=0$



$h_\theta(x) = g(\underset{-3}{\theta_0} + \underset{1}{\theta_1} x_1 + \underset{1}{\theta_2} x_2)$

$-3 + x_1 + x_2 \ge 0$ $\qquad x_1 + x_2 = 3$

$x_1 + x_2 \ge 3$



$h_\theta(x) = g(\overset{-1}{\theta_0} + \overset{0}{\theta_1} x_1 + \overset{0}{\theta_2} x_2 + \underset{1}{\theta_3} x_1^2 + \underset{1}{\theta_4} x_2^2)$

$-1 + x_1^2 + x_2^2 \ge 0$

$x_1^2 + x_2^2 \ge 0$

$x_1^2 + x_2^2 = 0$

çember formülü

Logistic Regression Model

Cost function

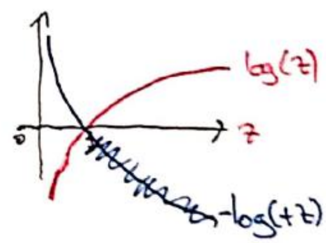
Training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots (x^{(m)}, y^{(m)})\}$

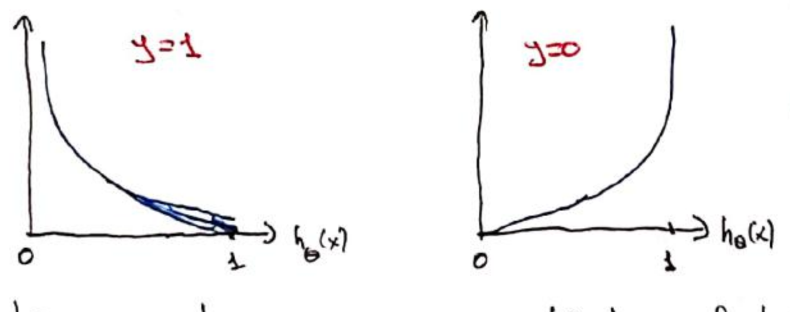
$$x \in \begin{bmatrix} x_0 \\ x_1 \\ \dots \\ x_m \end{bmatrix} \mathbb{R}^{n+1} \qquad x_0 = 1 \ , \ y \in \{0,1\}$$

Doğrusal regresyon için kullandığımız aynı maliyet işlevini kullanamayız. ($\frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}(h_\theta(x)-y)^2$) Çünkü lojistik işlev çıktılarının dışbükey bir grafik değil, birçok yerel optimuma sahip dalgalı bir grafik verir. Bu durumda yerel optimuma ulaştığımızı hiçbir zaman garanti edemez.

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)})$$

$$Cost(h_\theta(x), y) = -\log(h_\theta(x)) \quad \text{if } y=1$$
$$Cost(h_\theta(x), y) = -\log(1-h_\theta(x)) \quad \text{if } y=0$$





$Cost(h_\theta(x), y) = 0 \quad \text{if } h_\theta(x) = y$

$Cost(h_\theta(x), y) \to \infty \quad \text{if } y=0 \text{ ve } h_\theta(x) \to 1$

$Cost(h_\theta(x), y) \to \infty \quad \text{if } y=1 \text{ ve } h_\theta(x) \to 0$

Eğer doğru cevabımız $y=0$ ise hipotez fonksiyonumuz da 0 verirse maliyet fonksiyonu 0 olur. Hipotezimiz 1'e yaklaşırsa maliyet fonksiyonu sonsuza yaklaşır ve bize maliyeti oldukça fazla olur.

Eğer doğru cevabımız $y=1$ ise hipotez fonksiyonumuz 1 çıkarsa maliyet fonksiyonu 0 olur. Hipotezimiz 0'a yaklaşırsa maliyet fonksiyonu sonsuza yaklaşır.

Not Maliyet fonksiyonunun bu şekilde yazılması dışbükey olduğunu garanti eder.

Simple Cost Function ve Gradient Descent

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)})$$

$$Cost(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y=1 \\ -\log(1-h_\theta(x)) & \text{if } y=0 \end{cases}$$

Cost function'ımızı yukarıdaki gibi iki satırla ifade ediyoruz ancak gradient descent'ide uygulayabilmek için tek bir satıra indirgeyebiliriz.

$$Cost(h_\theta(x), y) = -y\log(h_\theta(x)) - (1-y)\log(1-h_\theta(x))$$

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

Lineer regresyonun maliyet fonksiyonu yukarıdaki gibidir. Ancak bu cost function yerine Principle of Maximum Likelihood Estimation (Maksimum olabilirlik tahmini ilkesi)'de kullanılabilir.

Gradient Descent

$J(\theta)$'mızı minimize etmemiz gerekiyor ve bunun içinde gradient descent kullanacağız.

$$\min_\theta J(\theta):$$

repeat{

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial\theta_j} J(\theta) \implies \theta_j := \theta_j - \frac{\alpha}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

}

$J(\theta)$ Vectorized

$$h = g(X\theta)$$

$$J(\theta) = \frac{1}{m} \cdot (-y^T \log(h) - (1-y)^T \log(1-h))$$

Gradient Descent Vectorized

$$\theta := \theta - \frac{\alpha}{m} X^T (g(X\theta) - \vec{y})$$

Advenced Optimization

Optimization Algorithms
- Gradient Descent
- Conjugate Gradient
- BFGS
- L-BFGS

• α, öğrenme oranını manuel seçmeniz gerekmez.
• Gradient descent'ten genelde daha hızlıdırlar.
• Aşırı karmaşıktır. Ne yaptıklarını anlamak oldukça güçtür.

örn. $\Theta = \begin{bmatrix} \Theta_1 \\ \Theta_2 \end{bmatrix}$

$$J(\Theta) = (\Theta_1 - 5)^2 + (\Theta_2 - 5)^2$$

$$\frac{\partial}{\partial \Theta_1} J(\Theta) = 2(\Theta_1 - 5)$$

$$\frac{\partial}{\partial \Theta_2} J(\Theta) = 2(\Theta_2 - 5)$$

```
function [jVal, gradient] = costFunction(theta)
jVal = (theta(1) -5)^2 + (theta(2)-5)^2;
gradient = zeros(2,1);
gradient(1) = 2*(theta(1)-5);
gradient(2) = 2*(theta(2)-5);
```

```
options = optimset('GradObj', 'on', 'MaxIter', 100);
initalTheta = zeros(2,1);
[optTheta, functionVal, exitFlag] = fminunc(@costFunction, initalTheta, options);
```

Not: Yukarıdaki optimizasyon algoritmalarını hazır kütüphanelerden kullanın. fminunc'ta Octave'de örnek verilebilecek optimizasyon fonksiyonudur. Daha büyük problemler üzerinde çalışırken gradient descent'ten daha hızlı olan bu algoritmaları kullanabilirsiniz.

Logistic Regression with Multi-class Classification

Emaillerinizi iş, arkadaş, aile, hobi olarak 4 sınıfa ayırmak isteyebilirsiniz veya tıkalı bir burunla hastaneye giden biri hasta olmayabilir, soğukalgınlığı, grip olabilir. Bu iki örnekte multi-class sınıflandırmaya örnektir.

$y \in \{0, 1, \ldots, n\}$

$h_\Theta^{(0)}(x) = P(y=0 \mid x;\theta)$

$h_\Theta^{(1)}(x) = P(y=1 \mid x;\theta)$

$\ldots$

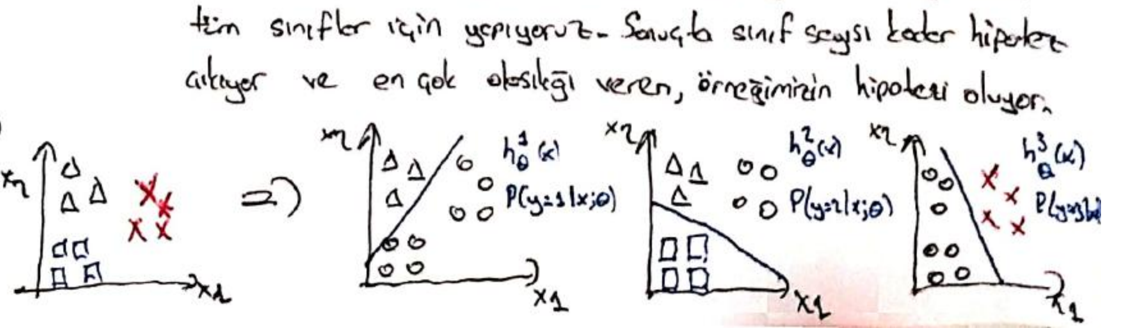
$h_\Theta^{(n)}(x) = P(y=n \mid x;\theta)$

prediction $= \max h_\Theta^{(i)}(x)$

One vs All (one vs rest)

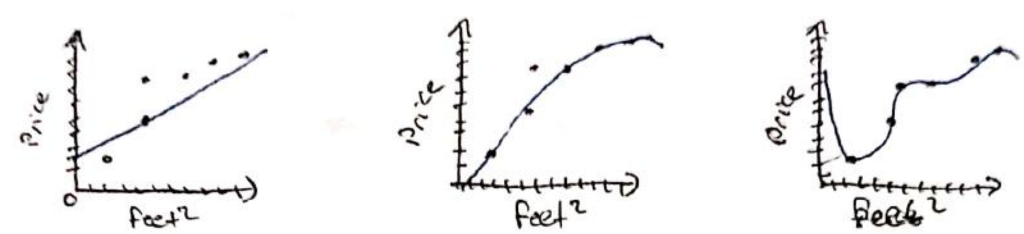
Bu yöntem tüm sınıfların içinden birini seçip diğerlerini bir gibi düşünüp logistic regresyonu uyguluyoruz ve bu yöntemi tüm sınıflar için yapıyoruz. Sonuçta sınıf sayısı kadar hipotez çıkıyor ve en çok olasılığı veren, örneğimizin hipotezi oluyor.

class 1: Δ
class 2: □
class 3: ×



Regularization (Düzenlileştirme)

Overfitting Problemi



Ev fiyat tahmini üzerinden devam edelim. En soldaki şekilde $y = \theta_0 + \theta_1 x_1$ modelinde verilerin düz bir çizgi üzerinde olmadığını ve uyumun (fit) çok iyi olmadığını görüyoruz.
Orta şekilde $y = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2$ modeli verilerimize dahi iyi uyan bir model sergiliyor. Ancak daha fazla özellik eklemizle beraber verilere daha iyi uyum gösteren model çıkmasına karşın yeni örnekleri iyi bir tahminde bulunma olasılığı düşecektir.

En soldaki şekilde model verileri iyi bir şekilde yakalayamıyor buna underfitting deniyor. En sağdaki şekil ise verilere çok fazla uyum sağlamış buna da overfitting deniyor.

Underfitting veya yüksek bias, hipotez fonksiyonumuz ile verilerin zayıf eşleşmesidir. Genellikle çok basit veya çok az özellik kullanan bir işlevden kaynaklanır.

Overfitting veya yüksek varyans, mevcut verilere (training set) çok iyi uyan ancak iyi bir genelleştirme yapamayan hipotez işlevinden kaynaklanır. Genellikle verilerle ilgisi olmayan çok sayıda gereksiz eğri ve açı oluşturan karmaşık bir fonksiyondan kaynaklanır.

Örnek olarak lineer regresyon örneği verdik ama logistic regresyonda da meydana gelir.

Overfittin özelliğini giderebilmek için iki ana yol vardır:

1. Özellik sayısını azaltın
   - Saklanacak özellikleri manuel seç.
   - Bir model seçim algoritması kullan.
2. Regularization (Düzenleme)
   - Tüm özellikleri koruyun, ancak parametrelerin (θ_j) büyüklüğünü azaltın.
   - Düzenleme, biraz kullanışlı özelliklere sahip olduğumuzda iyi çalışır.

Regularization - Cost Function

Eğer fonksiyonumuzda overfitting varsa, neden olan parametrelerin maliyetlerini arttırarak daha küçük değer almasını sağlayabiliriz.

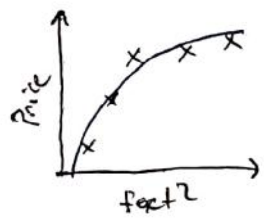
Diyelim ki aşağıdaki fonksiyonu karesel yapmak istiyoruz:

$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

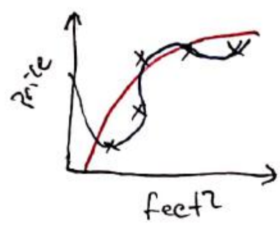
$\theta_3 x_3$ ve $\theta_4 x^4$ parametrelerinin etkisini kaldırmamız gerekecektir. Bu özelliklerden kurtulmadan ya da hipotezimizin biçimini değiştirmeden, cost function'ımızı değiştirebiliriz.

$$\min_\theta \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + 1000\,\theta_3^2 + 1000\,\theta_4^2$$

Maliyet fonksiyonumuzun sonuna maliyetini arttırmak için iki terim ekledik. Şimdi maliyet fonksiyonun sıfıra yaklaşması için θ_3 ve θ_4 sıfıra yakın olması gerekecek.



$\theta_0 + \theta_1 x + \theta_2 x^2$



$\theta_0 + \theta_1 x + \theta_2 x^2 + \underbrace{\theta_3 x^3 + \theta_4 x^4}_{\text{regularized}}$

Çok fazla özelliği olan bir problemde hangi özelliği overfittinge neden olduğunu anlamayabiliriz. Bunun için tüm parametreleri düzenleyen genel bir formül kullanırız. Overfitting olmasa dahi parametreleri regularize ederek küçültmek modeli daha pürüzsüz hale getiririz.

$$\min_\theta \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} \theta_j^2$$

Not: θ_0 regularize edilmez. Eğer regularization parametresi (λ) çok büyük seçilirse hipotez θ_0'a yakınsayan bir fonksiyona dönüşür. λ seçimine dikkat edilmelidir.

Regularization - Lineer Regression

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right]$$

$$\Rightarrow \frac{1}{2m} 2\lambda\theta_j = \frac{\lambda}{m}\theta_j$$

Gradient Descent

Cost function'ımız değiştiğine göre tekrar kısmi türev alırız ve aşağıdaki sonuç çıkar:

$$\theta_0 := \theta_0 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_0^{(i)}$$

$$\theta_j := \theta_j - \alpha\left[\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} + \frac{\lambda}{m}\theta_j\right] \quad j = (1,2,\ldots,n)$$

$$\theta_j := \theta_j\left(1 - \alpha\frac{\lambda}{m}\right) - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

Bu formülde $1-\alpha\frac{\lambda}{m}$ 1'den biraz küçük bir şeyi çıkacaktır. Çünkü α öğrenme oranımız küçük eğitim veri sayımız büyük olduğu için buradan çok küçük bir pozitif sayı çıkacaktır. Bunu da 1'den çıkardığımızda 0,80-0,99 arasında bir değer elde ederiz ve bu sayede θ güncellemelerinde θ'mızı daha küçük bir şekilde güncellemiş oluruz.

Normal Equation

$$X = \begin{bmatrix}(x^{(1)})^T \\ \vdots \\ (x^{(m)})^T\end{bmatrix} \qquad y = \begin{bmatrix}y^{(1)} \\ \vdots \\ y^{(m)}\end{bmatrix}$$

$m \times (n+1)$ $\qquad$ $m \times 1$

$$\theta = (X^TX)^{-1}X^Ty$$

$$\theta = \Big(X^TX + \lambda\underbrace{\begin{bmatrix}0 & & 0 \\ & 1 & \\ 0 & & \ddots \; 1\end{bmatrix}}_{(n+1)\times(n+1)}\Big)^{-1}X^Ty \Rightarrow \text{Update regularization formül}$$

$(n+1) \times (n+1)$ boyutlarındadır.

Not: Regularization normal equation'daki non-inventibility (tersine çevrilemezlik) sorununu da ortadan kaldırır. Tek şart λ'nın 0'dan büyük olmasıdır.

Not: Düzenleme yaparken θ_0'ı düzenlemediğimizden θ_0'ı formülde ayrı hesaplıyoruz.

Regularization - Logistic Regression

$$J(\theta) = -\left[\frac{1}{m}\sum_{i=1}^{m} y^{(i)}\log(h_\theta(x^{(i)})) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

Gradient Descent

repeat {

$$\theta_0 := \theta_0 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_0^{(i)}$$

$$\theta_j := \theta_j - \alpha\left[\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} + \frac{\lambda}{m}\theta_j\right] \qquad j = (1,2,3,\dots,n)$$

Aslında gradient descent lineer regresyondaki ile aynı yazım tarzında tek farkı hipotezlerinin farklı olmasıdır.

Not Logistic regresyonun doğru çalışıp çalışmadığını anlamak için $J(\theta)$ güncelleme geçmişini ve iterasyon sayısını içeren grafik çizmek ve bunun sonucunda azalan bir grafik görmek gerekir.

Regularization - Advanced Optimization Algorithm (fminunc)

function [jVal, gradient] = costFunction(theta)

jVal = [code to compute $J(\theta)$] $\Rightarrow J(\theta) = \left[\frac{-1}{m}\sum_{i=1}^{m} y^{(i)}\log(h_\theta(x^{(i)})) + (1-y)\log(1-h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$

gradient (1) = [code to compute $\frac{\partial}{\partial\theta_0}J(\theta)$] $\Rightarrow \left[\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_0^{(i)}\right]$

⋮

gradient(n+1) = [code to compute $\frac{\partial}{\partial\theta_n}J(\theta)$] $\Rightarrow \left[\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_n^{(i)}\right] + \frac{\lambda}{m}\theta_n$

Not Regularization her zaman eğitim setinde en iyi sonucu verir diyemeyiz.

Not Yeni bir özellik eklemek modeli daha iyi bir hale getirebilir.