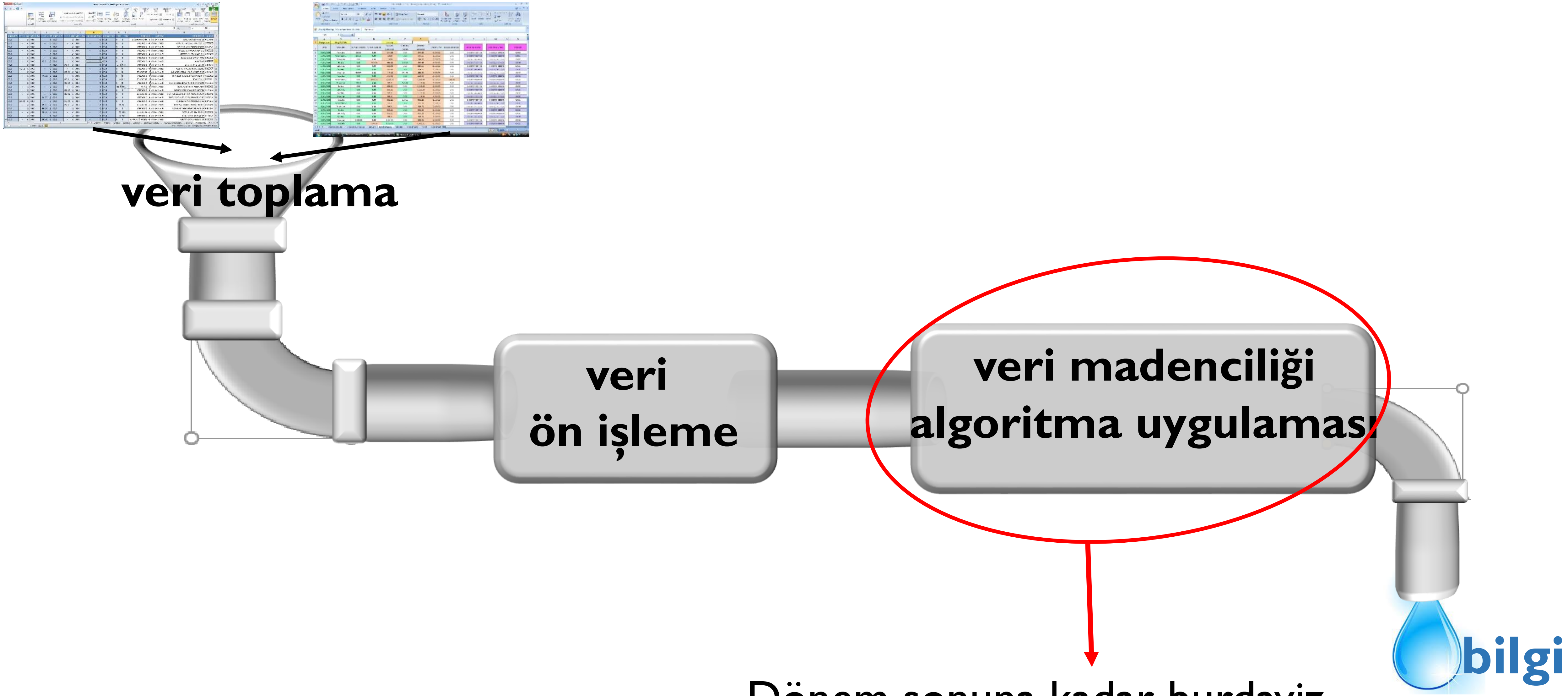
VERİ MADENCİLİĞİ

Fırat İsmailoğlu, PhD

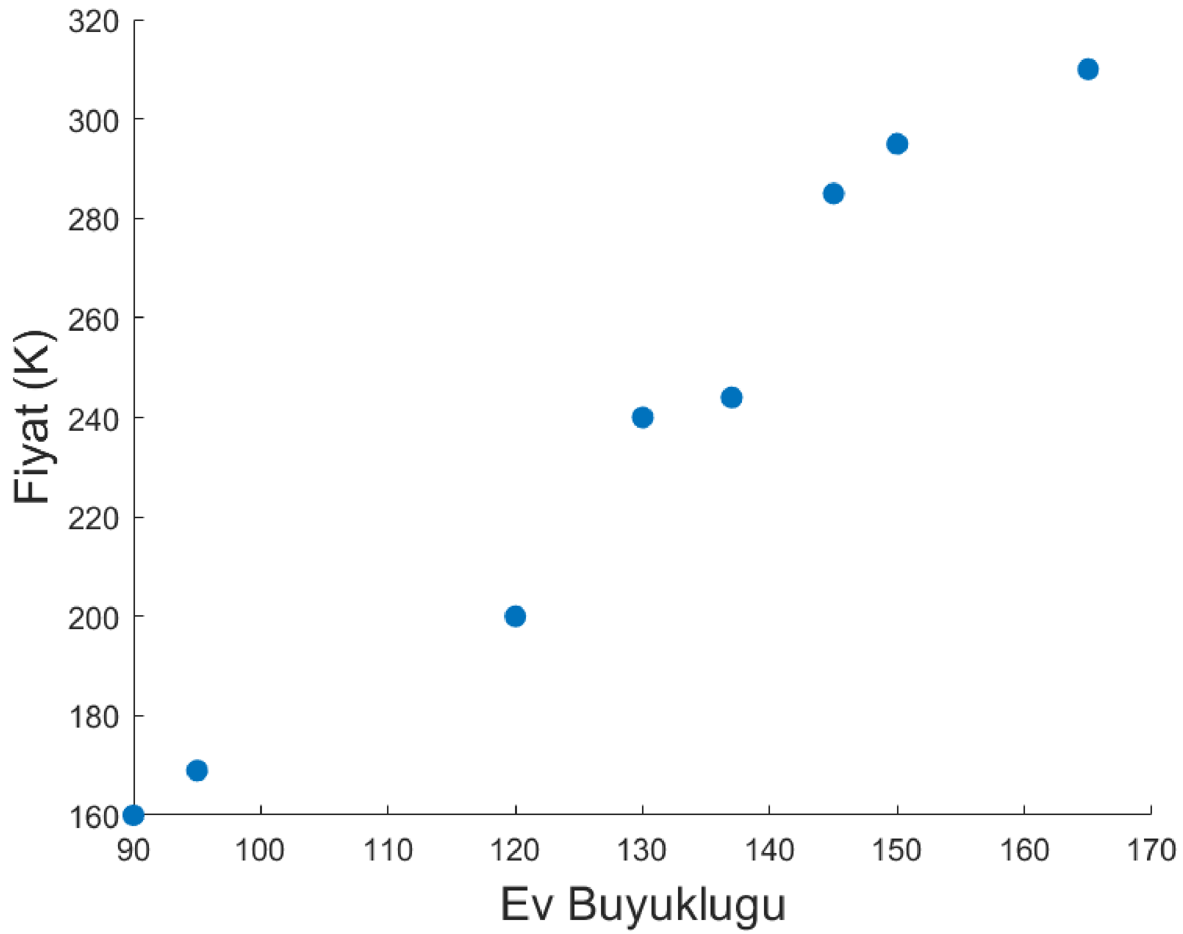
Doğrusal (Linear) Regresyon ve Gradient Descent



Dönem sonuna kadar burdayiz.

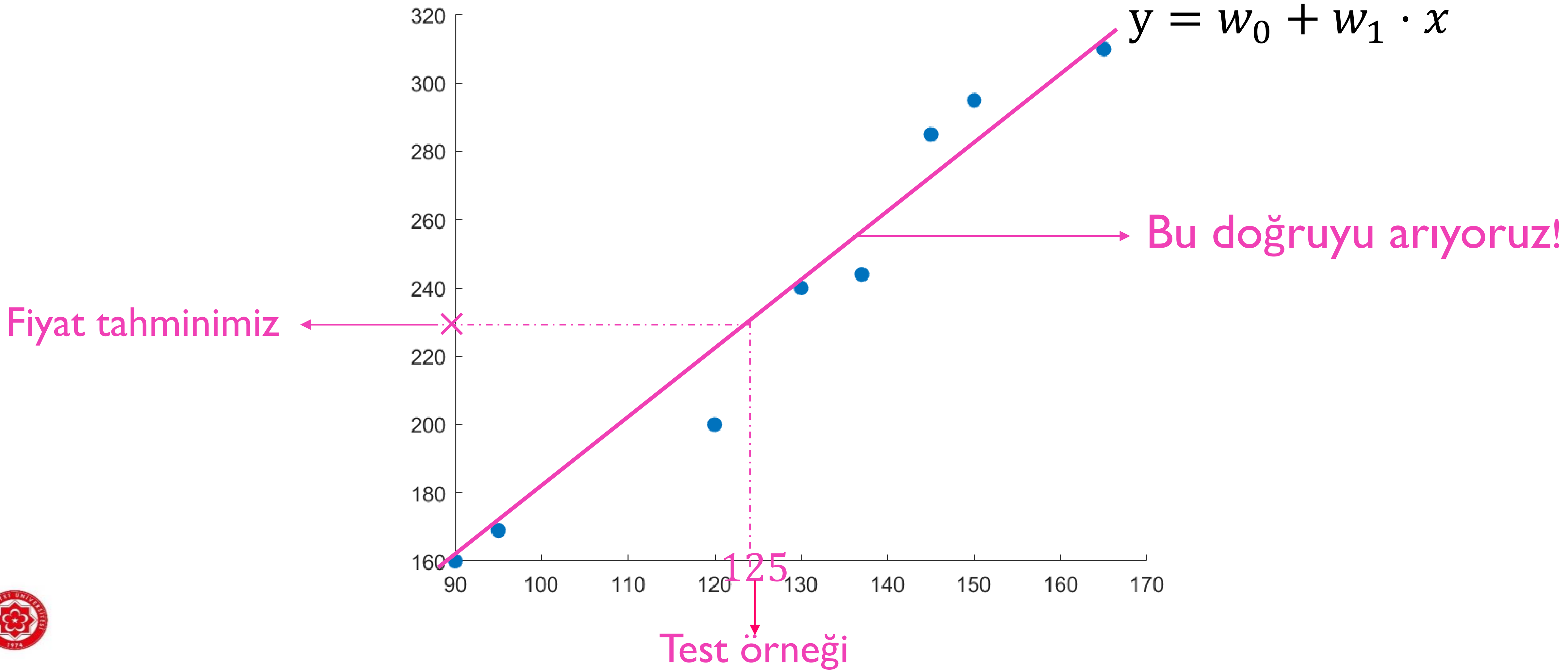
Regresyon

Bir diger veri madenciligi gorevi regresyondur. Regresyonda test örenkleri için sınıflandırmanın aksine kategorik bir değeri (sınıfı) değil, <u>sayısal bir değeri tahmin ederiz.</u>



Ev Büyüklüğü (X)	Fiyat (K) (Y)
90	160
137	244
150	295
120	200
95	169
145	285
165	310
130	240

Elimizde bu veri seti varken, ev büyüklüğü bilinen evlerin fiyatını tahmin etmek istiyoruz. Örneğin bir arkadaşımız bize 125 metrekare büyüklüğündeki bir evin fiyatının ne kadar olabileceğini sordu. Bu durumda elimizdeki veri setini (eğitim setini) kullanarak bir regresyon modeli kurariz. Bu, bir doğrudur.



Notasyon

(x^i, y^i), i. eğitim örneği olsun (yani veri matrisindeki i. satır).

x^i, i. eğitim örneğinin x değeri (ev büyüklüğü), y^i, i. eğitim örneğinin y değeri (fiyat) olsun.

Toplam eğitim seti örneği sayimiz m olsun.

Maliyet Fonksiyonu (Cost Function)

x değeri bilinen bir örneğin y değerini tahmin etmek için $f(x) = w_0 + w_1 x$ fonksiyonunu kullanıyoruz. Bu fonksiyonun eğitim setindeki örnekler için minimum hata vermesini istiyoruz.

Değeri y olan bir örneğe $f(x) = w_0 + w_1 x$ tahminini yapmakla ortaya çıkan maliyet:

$$f(x) - y$$

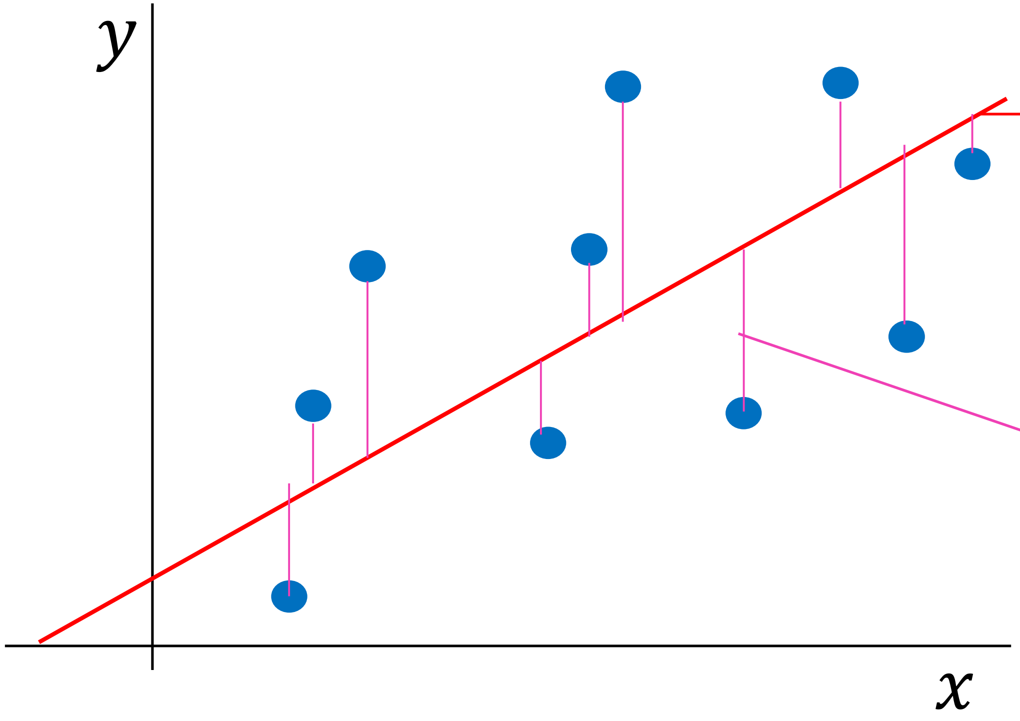
Maliyetin pozitif olmasi için bu farkın karesini aliyoruz. Eğitim setindeki tüm örnekleri için ortalama maliyet:

$$\boxed{\frac{1}{2m}\sum_{i=1}^{m}\left(f(x^i) - y^i\right)^2}$$

Maliyet fonksiyonu, M , w_0 ve w_1'in fonksiyonudur:

$$M(w_0, w_1) = \frac{1}{2n}\sum_{i=1}^{m}\left(w_0 + w_1 x^i - y^i\right)^2$$

Amacımız bu fonksiyonu minimum yapacak , w_0 ve w_1'i bulmaktır.



aradığımız regresyon doğrusu:

$y = w_0 + w_1 x$

maliyet (hata miktarı)= $w_0 + w_1 x - y$

Regresyonda çoğunlukla tek bir özellik (bağımsız değişken) olmaz. Bir çok özellik olur.

Örneğin bir evin fiyatına yalnızca evin büyüklüğü etki etmez. Bununla birlikte, oda sayısı, evin merkeze uzaklığı ve evin yaşı da evin fiyatının belirlenmesine katkıda bulunabilir.

Büyüklük ($metre^2$)	Oda Sayısı	Merkeze Uzaklık (metre)	Evin Yaşı (yıl)	Evin Fiyatı (TL-K)
140	3	3300	2	295
180	4	500	1	401
90	2	6500	14	112
120	3	5523	18	167
...	...	...	...	...
x_1	x_2	x_3	x_4	$\underbrace{\qquad}_{\text{Tahmin etmek istedigimiz alan}}^{Y}$

Birden fazla özellik varken regresyon fonksiyonumuz su şekilde olur:

$$f(x) = w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n$$

$n, (n > 1)$ burada toplam özellik sayısı.

Not: Dikkat edilirse yukarıdaki fonksiyonda her bir x_i özelliğine karşı bir w_i katsayısı (ağırlığı) vardır. Genel olarak w_i ne kadar büyükse x_i özelliği bağımlı değişken y'yi tahmin etmede o kadar büyük rol oynar.

Birden çok özellik varken maliyet fonksiyonu:

$$M(w_0, w_1, \dots, w_n) = \frac{1}{2m}\sum_{i=1}^{m}\left(f(x^i) - y^i\right)^2$$

$f\left(x^i\right) = w_0 + w_1x_1^i + w_2x_2^i + \cdots + w_nx_n^i$ ($x_j^i \rightarrow i.$ örneğin $j.$ özelliği ($i.$ satır $j.$ sütun))

Not: Maliyet fonksiyonu $w_0, w_1, \dots, w_n$'nin fonksiyonudur. Tüm bu ağırlıkların bulunması gerekir.

Maliyet Fonksiyonun Vektör Gösterimi (Kompakt Gösterim)

x^i örneğini, $x^i = \begin{pmatrix} x_1^i \\ x_2^i \\ \cdots \\ x_n^i \end{pmatrix}$ vektörü ile gösterelim. Vektöre 1 elemanı ekleyelim: $x^i = \begin{pmatrix} 1 \\ x_1^i \\ x_2^i \\ \cdots \\ x_n^i \end{pmatrix}$.

Bu durumda $(n+1)$ boyutlu x^i ile $(n+1)$ boyutlu $w^T = (w_0, w_1, w_2, \ldots, w_n)$ ağırlık vektörü aynı boyutta olur. Böylece x^i ile w^T *iç çarpımını* yapabiliriz:

$$w^T x^i = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n = f(x).$$

Maliyet fonksiyonunun yeni gösterimi:

$$\boxed{M(w_0, w_1, \ldots, w_n) = \frac{1}{2m} \sum_{i=1}^{m} \left(w^T x^i - y^i\right)^2}$$

Özellikler Arası Etkileşimin Regresyon Modelinde Dikkate Alınması

Verilen bir veri setinde bazı özellikler arasında korelasyon olabilir. Yani bir özelliğin artmasi (azalması) diğer bir özelliğin artmasını (azalmasını) etkileyebilir. Örnek olarak aşağıdaki veri setinde Büyüklük ile Oda sayısı arasında (doğrusal) korelasyon vardır. Büyüklük artarsa oda sayısı da artar. Böyle durumlarda korele olan özellikler çarpılarak yeni bir özellik oluşturulur. Daha sonra regresyona geçilir.

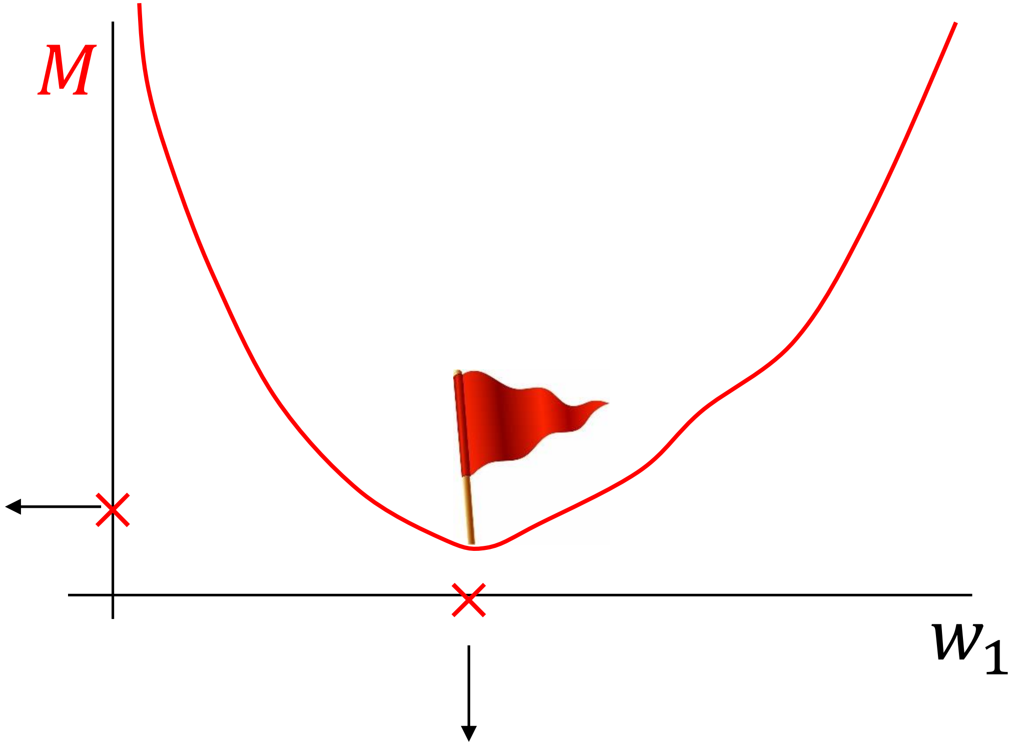
Büyüklük ($metre^2$)	Oda Sayısı	**Büyüklük x Oda S.**	Merkeze Uz. (metre)	Evin Yaşı (yıl)	Evin Fiyatı (TL-K)
140	3	**420**	3300	2	295
180	4	**720**	500	1	401
90	2	**180**	6500	14	112
120	3	**360**	5523	18	167
...	...	**...**	...	...	...

Gradient Descent (Eğimli Azalma)

Görsellik acisindan farz edelimki maliyet fonksiyonu yalnızca w_1'in fonksiyonu olsun.

Yani $M = M(w_1)$. Aşağıdaki kırmızı eğri, w_1'değerleri değişirken M'nin aldigi değerleri göstersin.

M'nin en düşük değeri



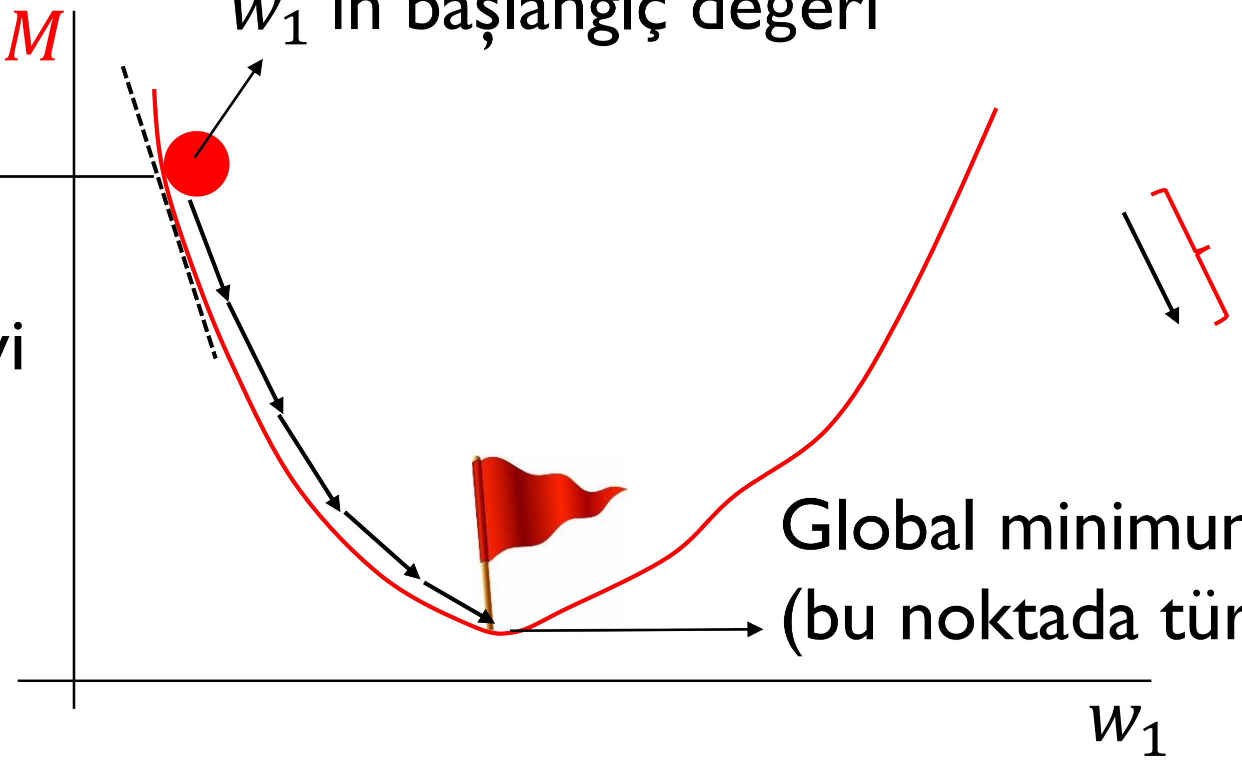
Amacımız en düşük M değerini veren bu w_1 değerini bulmak!

Gradient Descent (Eğimli Azalma)

w_1'in başlangıç değeri

Eğim (gradient)

(M'nin w_1'e göre türevi eğimi - gideceğimiz yönü verir)



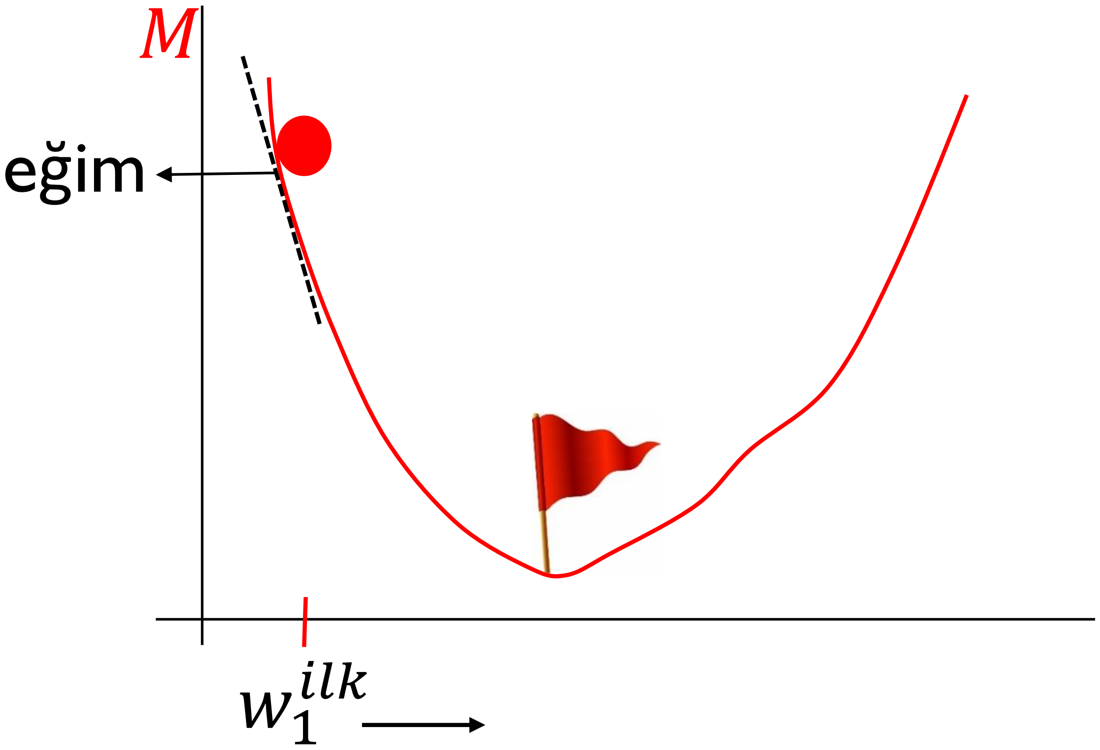
adım büyüklüğü

Global minimum nokta (bu noktada türev 0'dır)

w_1'in güncellenme kuralı

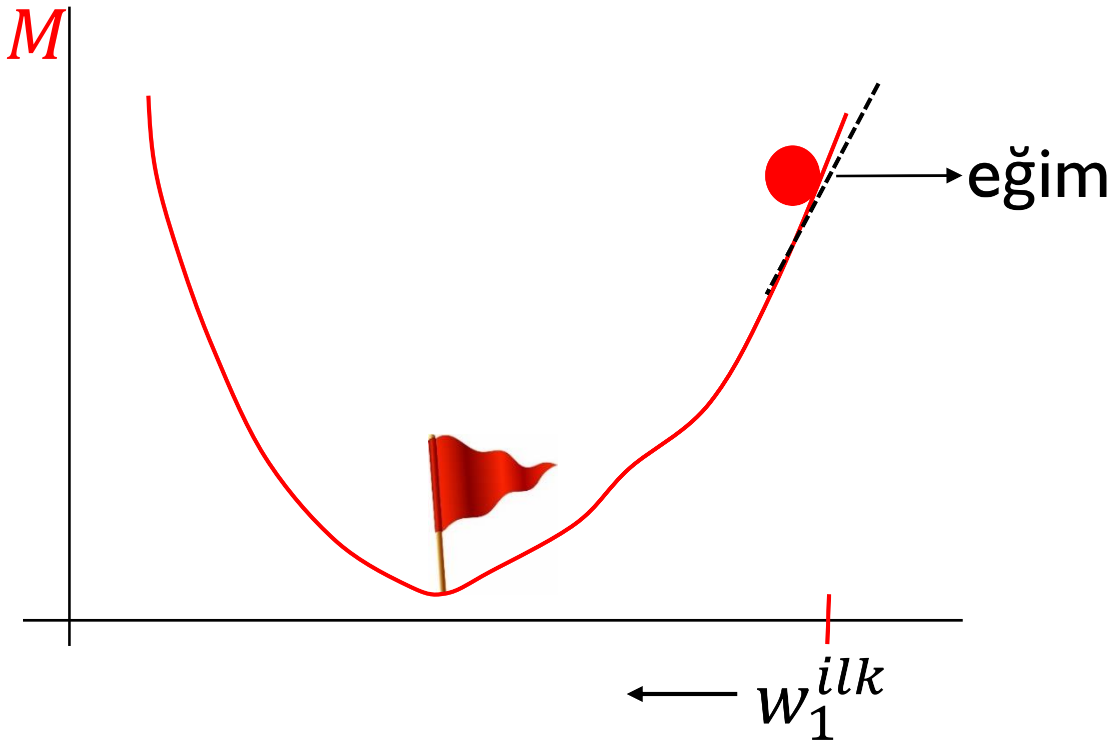
$$w_1 \leftarrow w_1 - \overbrace{\underbrace{\alpha}_{\text{öğrenme oranı}} \cdot \underbrace{\frac{dM}{dw_1}}_{\text{türev (hangi yöne gideceğimiz)}}}^{\text{adım büyüklüğü}}$$

Gradient Descent Neden Çalışır?



bu noktada eğim negatiftir, dolayısıyla $\frac{dM}{dw_1}$ türevi negatiftir.

$$\underbrace{w_1 \leftarrow w_1 - \alpha \cdot (\text{negatıf sayı})}_{w_1 \text{ artar!}}$$



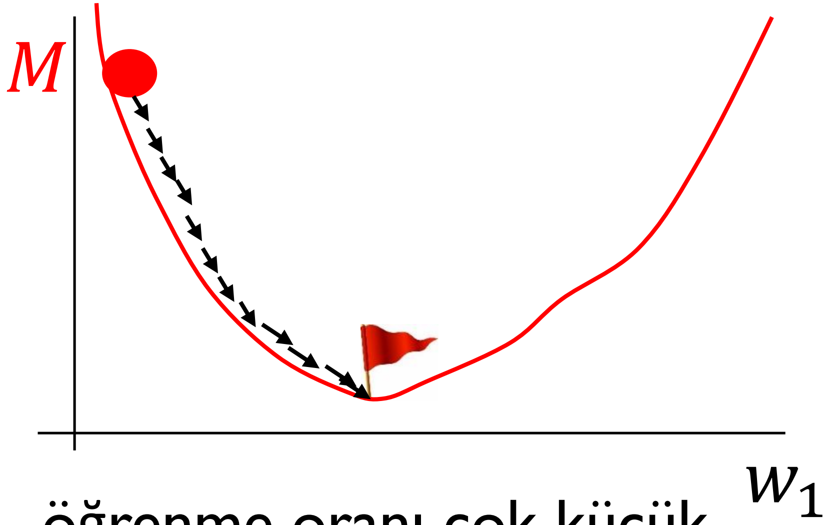
bu noktada eğim pozitiftir, dolayısıyla $\frac{dM}{dw_1}$ türevi pozitiftir.

$$\underbrace{w_1 \leftarrow w_1 - \alpha \cdot (\text{pozitif sayı})}_{w_1 \text{ azalır!}}$$

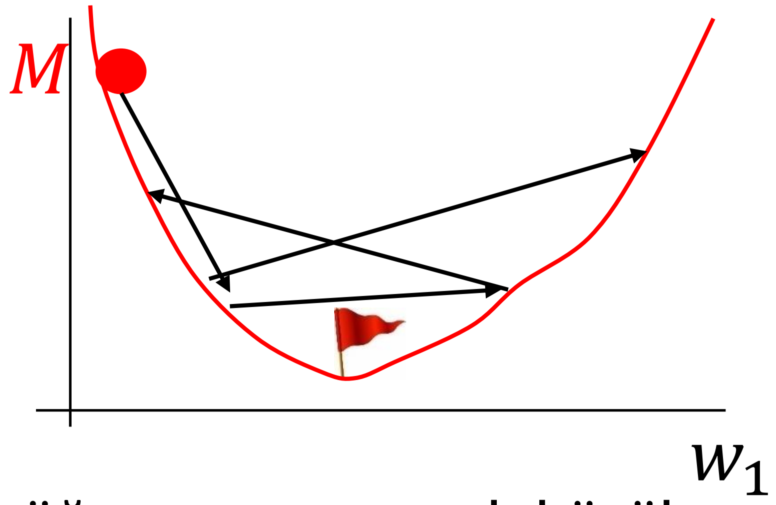
Öğrenme Oranı (Learning Rate) Nasıl Seçilmeli?

Öğrenme oranı (α) çok küçük seçilirse minimum noktaya ulaşmamız çok yavaş olur.

Öğrenme oranı çok büyük seçilirse minimum noktayı ıskalayabiliriz. Bu durumda M değeri azalacağı yerde artabilir!



öğrenme oranı çok küçük



öğrenme oranı çok büyük

Genel olarak … $0.001, 0.01, 0.1, 1,$ … değerleri denenerek biri seçilir.

Gradient Descent Algoritması (Genel)

```
Giris: w = (w_0, w_1, ..., w_n) başlangıc vektoru, M = M(w_0, w_1, ..., w_n) fonksiyonu
α öğrenme oranı, maxIter maksimum iterasyon sayısı
Cikis: optimize edilmiş w vektoru
for i=1:maxIter
    for j=0:n
        w'_j := w_j - α · ∂M/∂w_j   //her bir w_j'ye gecici deger ata
    end for
(w_0, w_1, ..., w_n) ← (w'_0, w'_1, ..., w'_n)   //w'ları aynı anda güncelle
end for
```

Giris: $w = (w_0, w_1, \ldots, w_n)$ başlangıc vektoru, $M = M(w_0, w_1, \ldots, w_n)$ fonksiyonu

α öğrenme oranı, maxIter maksimum iterasyon sayısı

Cikis: optimize edilmiş w vektoru

for i=1:maxIter

for j=0:n

$$w_j' \coloneqq w_j - \alpha \cdot \frac{\partial M}{\partial w_j}$$ //her bir w_j'ye gecici deger ata

end for

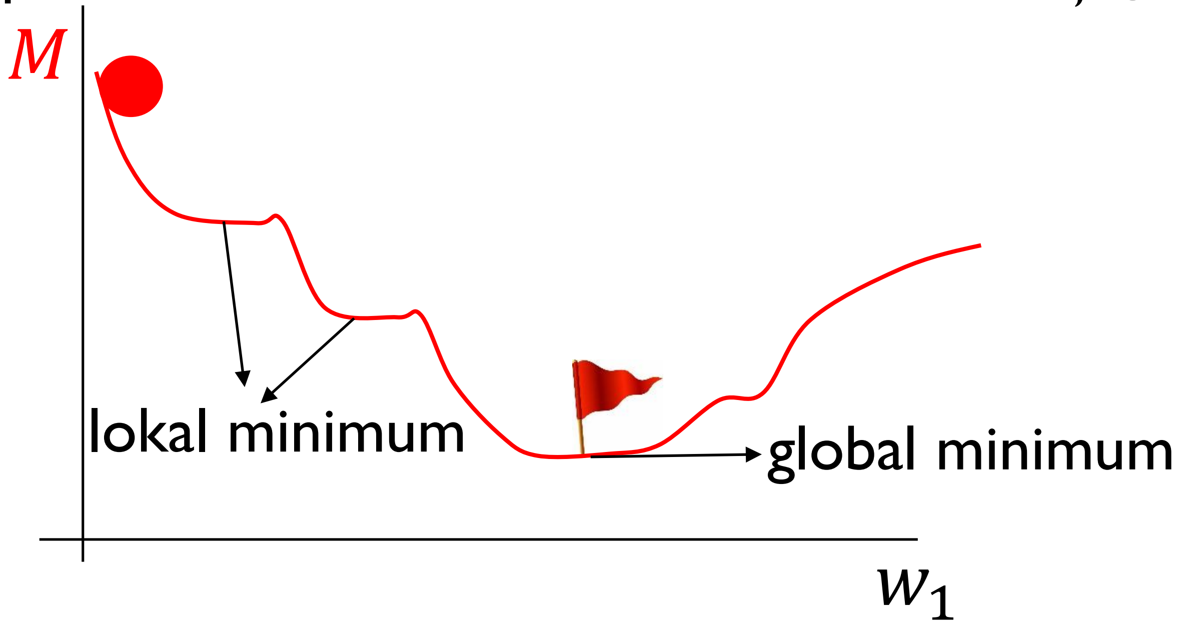
$$(w_0, w_1, \ldots, w_n) \leftarrow (w_0', w_1', \ldots, w_n')$$ //w'ları aynı anda güncelle

end for

Lokal Minimum Sorunu

Birçok kez maliyet fonksiyonunun global minimum noktalarinin yaninda *lokal minimum* noktalari da olur. Bu noktalarda da eğim 0'dır;dolayısıyla türev 0'olur. w_i'ler güncellenemez.

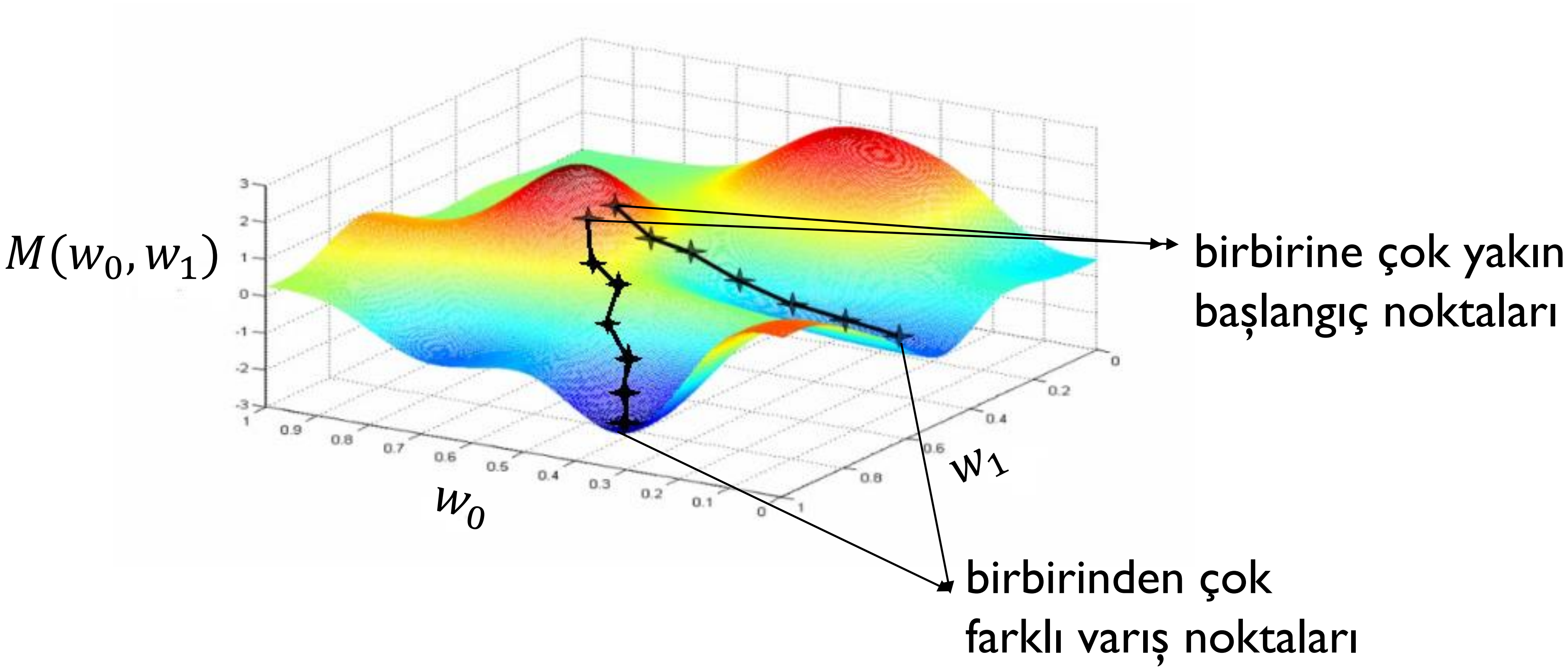
Görsel olarak, aşagidaki top lokal minimum'a takılı kalır. Daha fazla aşağı ilerlemez.



Çözüm: Her defasında farklı başlangıç değerleri vererek gradient descent algoritmasını birden fazla kez çalıştırmak. Herbir çaliştirma farklı w vektoru üretir. Bunlar içerisinde en düşük maliyeti (M'yi) veren w'yu seçmek.

Lokal Minimum Sorunu

M, bir değil iki değişkenin (w_0 ve w_1) fonksiyonu olsun. Bu durumda lokal minimum sorunu aşagidaki gibi olabilir. Farklı fakat birbirine cok yakin başlangiç noktalarindan gradient descent ile çok farkli noktalara varılabilir.



Regresyon Katsayılarının Gradient Descent Kullanılarak Bulunması

Maliyet fonskiyonu $M(w_0, w_1, \dots, w_n)$

$$M(w_0, w_1, \dots, w_n) = \frac{1}{2m}\sum_{i=1}^{m}\left(w_0 + w_1 x_1^i + w_2 x_2^i + \cdots + w_n x_n^i - y^i\right)^2$$

$w_0, w_1, \dots, w_n$'e gore kismi turevlerini alalim.

$\frac{\partial M}{\partial w_0} = \frac{1}{m}\sum_{i=1}^{m}(w_0 + w_1 x_1^i + w_2 x_2^i + \cdots + w_n x_n^i - y^i)$
$\frac{\partial M}{\partial w_j} = \frac{1}{m}\sum_{i=1}^{m}(w_0 + w_1 x_1^i + w_2 x_2^i + \cdots + w_n x_n^i - y^i) \cdot x_j^i \; (j \in \{1, 2, \dots, n\}$

Buldugumuz bu türevler gradient descent algoritmasinda w_j guncellemesinde

$$w_j' := w_j - \alpha \cdot \frac{\partial M}{\partial w_j}$$

yerine konur.

Regresyon Icin Gradient Descent Algoritması

```
Giris: başlangıc vektoru, M = M(w_0, w_1, ..., w_n) fonksiyonu
α öğrenme oranı, maxIter maksimum iterasyon sayısı
Cikis:Optimize edilmiş w vektoru
for i=1:maxIter
    w'_0 := w_0 − α · (1/m) Σ_{i=1}^{m} (w_0 + w_1 x_1^i + w_2 x_2^i + ⋯ + w_n x_n^i − y^i)
    for j=1:n
        w'_j := w_j − α · (1/m) Σ_{i=1}^{m} (w_0 + w_1 x_1^i + w_2 x_2^i + ⋯ + w_n x_n^i − y^i) · x_j^i
    end for
(w_0, w_1, ..., w_n) ← (w'_0, w'_1, ..., w'_n)  //w'ları aynı anda güncelle
end for
```

$$w_0' := w_0 - \alpha \cdot \frac{1}{m}\sum_{i=1}^{m}(w_0 + w_1 x_1^i + w_2 x_2^i + \cdots + w_n x_n^i - y^i)$$

$$w_j' := w_j - \alpha \cdot \frac{1}{m}\sum_{i=1}^{m}(w_0 + w_1 x_1^i + w_2 x_2^i + \cdots + w_n x_n^i - y^i) \cdot x_j^i$$

$$(w_0, w_1, \ldots, w_n) \leftarrow (w_0', w_1', \ldots, w_n')$$

Not: Gradient descent'e başlamadan önce özellikler genellikle *normalize* edilerek [0-1] arası değerler almaları sağlanır. Böylece gradient descent çok daha hızlı bir şekilde (daha az iterasyonla) maliyeti minimum yapacak katsayıları w_0, w_1 ... verir.

Min-max normalizasyonu: $x_j^i \leftarrow \frac{x_j^i - min}{max - min}$

ör.

Ev Büyüklüğü (x)	Fiyat (K) (y)
90	160
137	244
150	295
120	200
95	169
145	285
165	310
130	240

Normalizasyon sonrası:

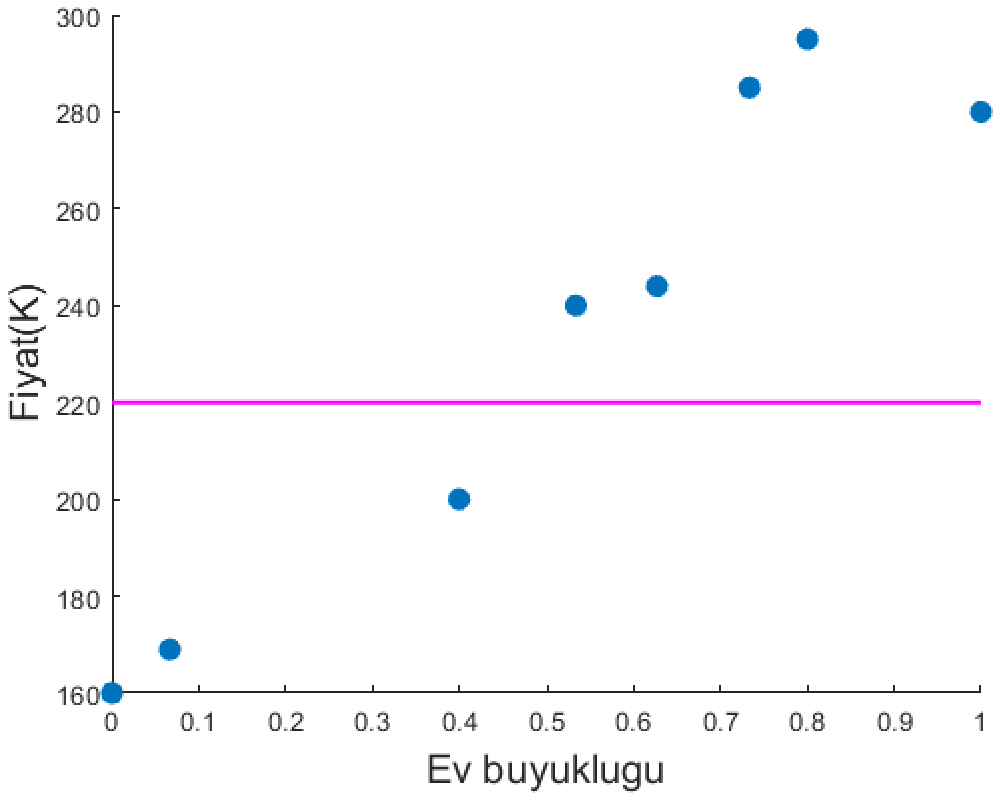
Ev Büyüklüğü (x)	Fiyat (K) (y)
0	160
0,62	244
0,8	295
0,4	200
0,06	169
0,73	285
1	310
0,53	240

Normalizayondan sonra gradient descent uygulamaya başlayalım.

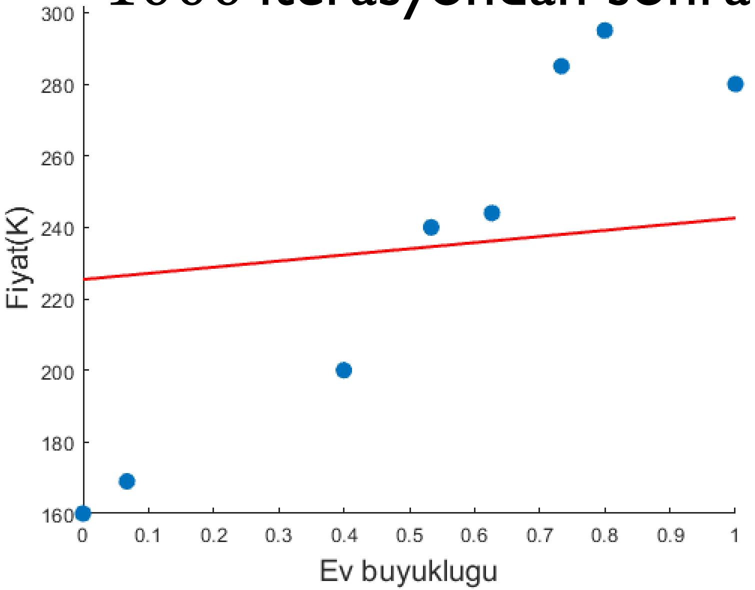
$w = (w_0, w_1)$ başlangıç vektörünü $(220{,}0)$ alalım.

öğreneme oranı $\alpha = 0.01$ olsun.

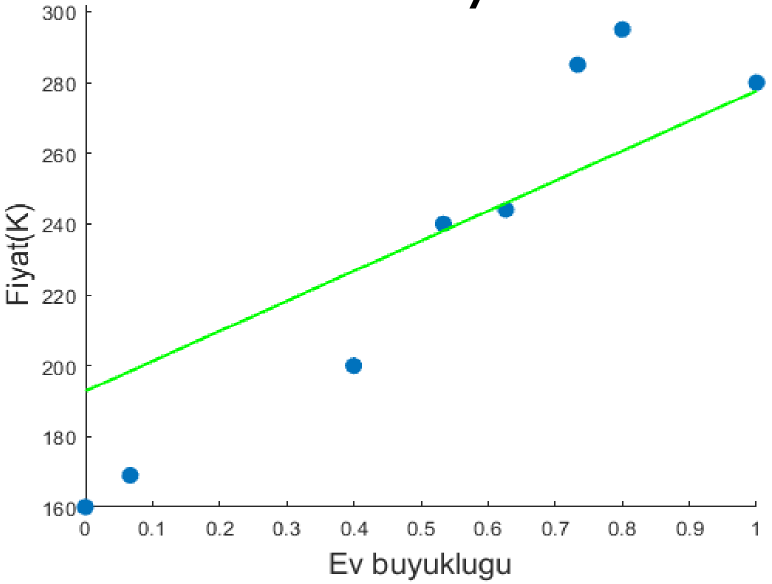
Başlangıç durumunda ($w = (220{,}0)$ için) regresyon doğrusu şöyle olur.



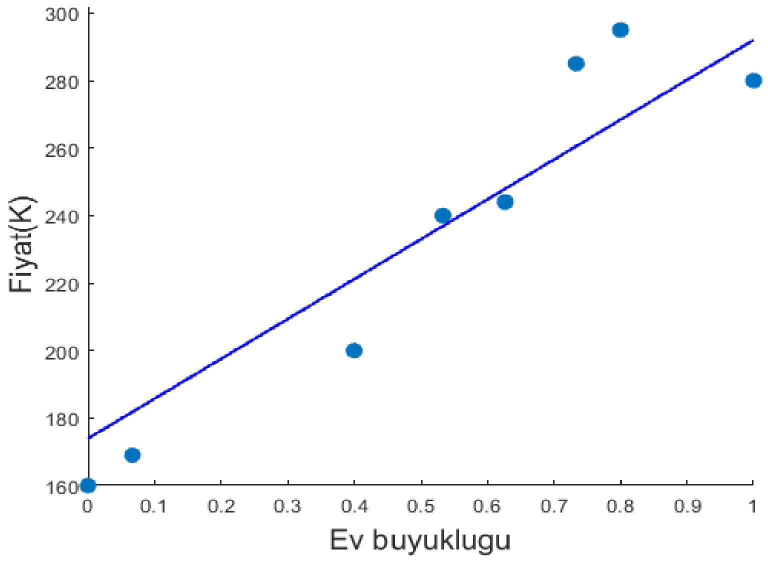
1000 iterasyondan sonra



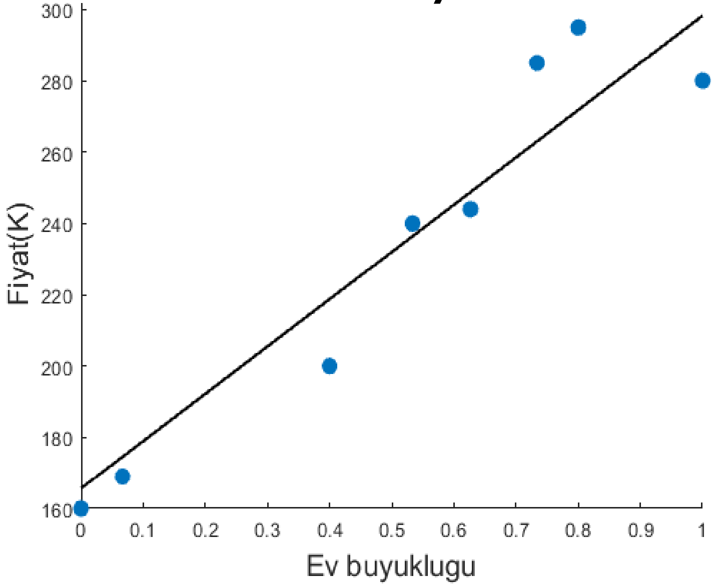
10000 iterasyondan sonra



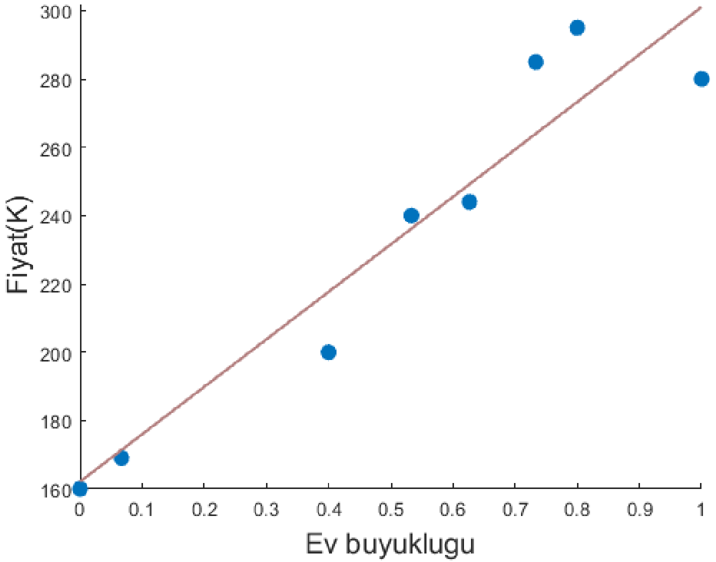
20000 iterasyondan sonra



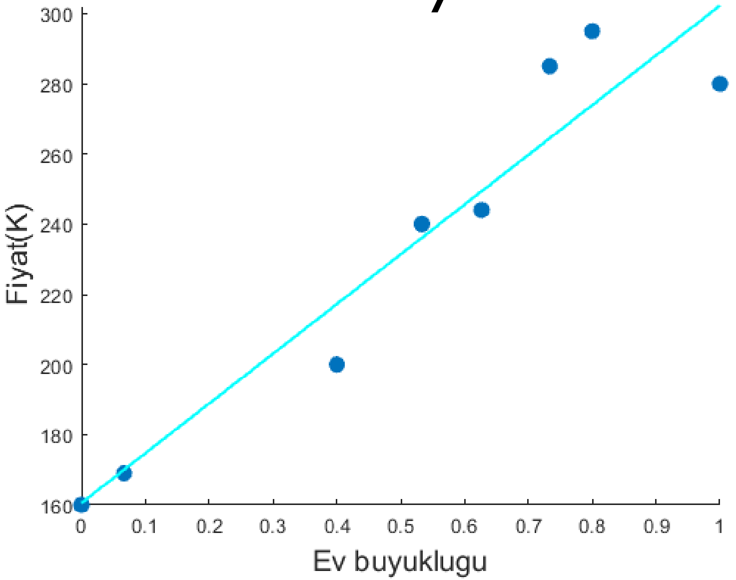
30000 iterasyondan sonra



40000 iterasyondan sonra



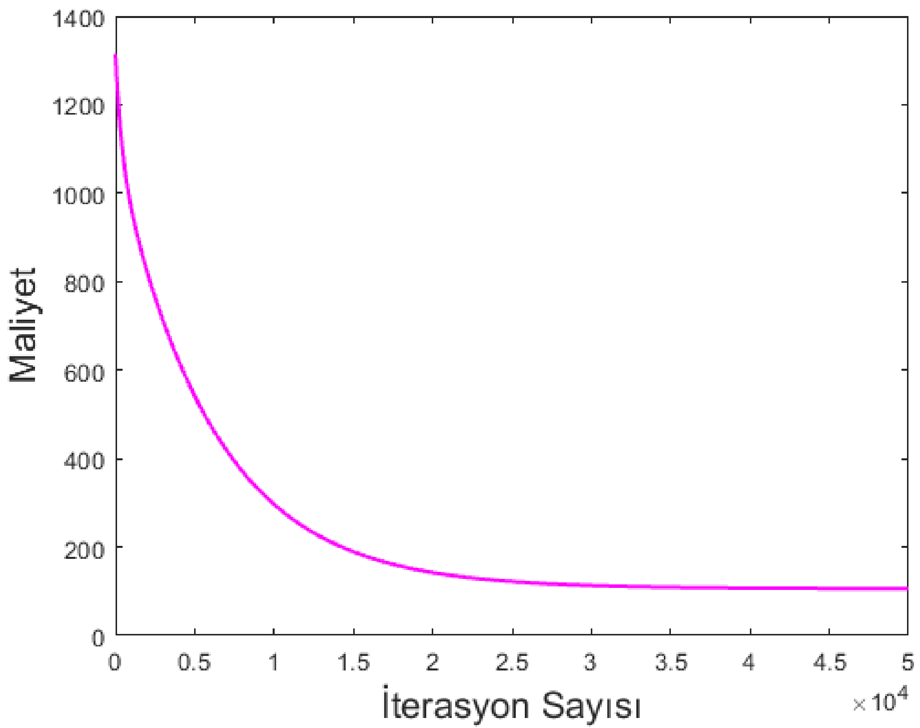
50000 iterasyondan sonra



Not: Dikkat edilirse belirli bir iterasyon sayisindan sonra (örnekte yaklaşık 20000) $w = (w_0, w_1)$ vektörü pek değişmemeye başlar. Bu, gradient descent algoritmasinin yakınsadığını, w'yu artık daha fazla iyileştırmeyeceği anlamına gelir.

Gradient Descent Algoritmasinin Çalışıp Çalışmadığı Nasıl Anlaşılır?

Bunu anlamak için gradient descentdeki her bir iterasyondaki w değeri için maliyet fonksiyonu hesaplanır. Herbir iterasyonda maliyetin azaldiği (en azından artmadigi) anlamak için maliyet-iterasyon sayısı grafiği ile çizilir.



Lineer Cebir İle Regresyon Katsayılarının Bulunması

Regresyonda $w_0, w_1, \dots, w_n$ katsayıları iterasyonsuz direkt olarak hesaplanabilir.

ör.

1 kolonu ekliyoruz.

	Ev Büyüklüğü (x)	Fiyat (K) (y)
1	0	160
1	0,62	244
1	0,8	295
1	0,4	200
1	0,06	169
1	0,73	285
1	1	310
1	0,53	240

X y

Lineer Cebir İle Regresyon Katsayılarının Bulunması

$$\underbrace{\begin{bmatrix} 1 & 0 \\ 1 & 0{,}62 \\ 1 & 0{,}8 \\ 1 & 0{,}4 \\ 1 & 0{,}06 \\ 1 & 0{,}73 \\ 1 & 1 \\ 1 & 0{,}53 \end{bmatrix}_{8\times 2}}_{X} \times \underbrace{\begin{bmatrix} w_0 \\ w_1 \end{bmatrix}_{2\times 1}}_{w} = \underbrace{\begin{bmatrix} 160 \\ 244 \\ 295 \\ 200 \\ 169 \\ 285 \\ 310 \\ 240 \end{bmatrix}_{8\times 1}}_{y}$$

Elimizdeki denklem: $X \times w = y$. Burada X ve y'yi bilirken; w'yu bilmiyoruz.

X, w ve y'yi vektor yada matris gibi degil birer reel sayi gibi dusunursek, w'yu bulmak icin denklemin her iki tarafını X'in tersi olan X^{-1} 'e bolmemiz gerekirdi. Fakat burda X matrıs oldugundan X'in tersi her zaman olmayabilir.

(Bir matrisin tersninin olmasi icin oncelikle kare olması gerekir.)

$X^T X$ matrisinin ise genelde tersi vardır.

$$X \times w = y$$

denkleminde her iki tarafi soldan X^T matrisi ile carpalim:

$$X^T X \times w = X^T y.$$

Denklemde her iki tarafi $(X^T X)^{-1}$ matrisine bölersek:

$$\boxed{w = (X^T X)^{-1} X^T y}$$

Bir onceki örnekte gradient descent ile 50000 iterasyonun sonucunda bulunan w degeri:
$w = (160.5\,, 141.7)$.

$w = (X^T X)^{-1} X^T y$ formuluyle bulunan w degeri:$w = (159.2\,, 143.9)$.

Yinede regresyon katsayilari hesaplanirken çogunluka gradient descent kullanilir. Çunku özellik sayısı fazla iken (n buyuk iken örnegin $n > 1000$) $X^T X$ matrisinin tersini hesaplamak kolay degildir. Böyle durumlarda gradient descent çok daha hızlı bir sekilde w vektorunu verir.