Makine Öğrenmesi Ders Notları
Hazırlayan: Barış Baykant Alagöz
İnönü Üniversitesi, Bilgisayar Mühendisliği, 2023-2024 Bahar Dönemi Lisans Dersi
Sorularınız ve düzeltme önerileriniz için e-mail: baykant.alagoz@inonu.edu.tr

Gradyan İniş Yöntemi

Gradyan iniş yöntemi lineer(doğrusal) olmayan optimizasyon yöntemidir ve gradyan iniş yönteminin nümerik çözümü yapay sinir ağları gibi bir çok karmaşık ve lineer olmayan optimizasyon yöntemlerinde başarı ile uygulanmış ve bir çok makine öğrenmesi ve modelleme yönteminin matematiksel altyapısını oluşturmuştur.

İşlem maliyeti düşük ve rekürsif (öz yinelemli) çözüme imkan sağlaması dolayısı ile gradyan iniş yöntemleri iteratif çözüm gerektiren yüksek karmaşıklıktaki makine öğrenmesi algoritmalarında tercih edilen bir nümerik optimizasyon yöntemine dönüşmüştür.

Gradyan İniş Yönteminin Temel Mekanizması:

Sürekli ve türevlenebilir bir $f(x)$ fonksiyonun bir noktadaki türevi, fonksiyonun o noktadaki eğimidir. Eğim fonksiyonu $m(x)=\frac{df(x)}{dx}$ olarak ifade edilir. Optimizasyonda ihtiyaç duyulan ekstremum noktaları (tepe yada çukur noktaları), fonksiyonun eğiminin sıfır olduğu noktalarda elde edilir. Tek değişkenli bir fonksiyonun eğimini sıfır yapan nokta, fonksiyon türevi sıfıra eşitlenerek bulunur.

$$\frac{df(x)}{dx}=0$$

denkleminin çözümü olan x_i değerleri ekstremum(dönüm noktaları - tepe ve çukurlar) noktalarıdır. Bu noktalara(dönüm noktalarına), $f(x)$ fonksiyonu sürekli ve türevlenebilir olması durumunda, fonksiyonun her noktadaki eğimini takip ederek ulaşabiliriz. Bu bize bir noktadan başlayıp iteratif (rekürsif, özyinelemeli) olarak bir ekstramum noktasına nümerik olarak yaklaşmamızı sağlayabilir. Dolayısı ile, sürekli ve türevlenebilir bir $f(x)$ fonksiyonun $x=x_0$ noktasından başlanması durumunda lokal minimum noktasına yakınsayan öz yinelemeli çözümü,

$$x_{n+1}=x_n-\eta\frac{df(x)}{dx}|_{x=x_n}$$

ile ifade edilir. Bu iteratif çözüm, aslında $f(x)$ fonksiyon eğimini fonksiyon değerinin azalan yönünde izleyen bir nümerik çözüm sağlar. Bu ifadeye gradyan iniş çözümü denir ve rekürsif olarak çözülerek bir başlangıç noktasından minimum noktasına ulaşılır. Aynı ifadenin programlama için uygun olan ifadesi şöyle de yazılır.

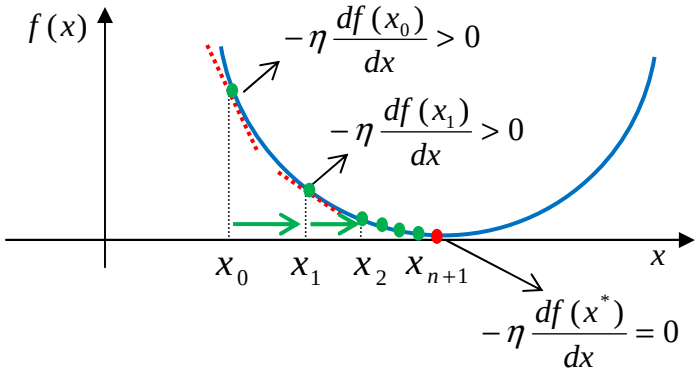
$$x_{i+1}\leftarrow x_i-\eta f'(x_i)$$

Burada her bir iterasyonda (hesaplama adımında) x'deki değişim yani güncelleme miktarı,

$$\Delta x_i = x_{i+1} - x_i \quad => \quad \Delta x_i = -\eta \frac{df(x_i)}{dx}$$

ifade edilir ve η öğrenme katsayısı sabit olduğu kabul edilirse $f(x)$'in birinci türevi ($\frac{df(x_i)}{dx}$) yani $f(x)$ fonksiyonun eğimi ile belirlenir. Şimdi bu güncelleme sürecini inceleyelim.

Bir $x = x_0$ noktasından başlanırsa ve rekürsif çözüm yapılırsa,



Optimal nokta dip noktasıdır ve burada fonksiyonun eğim sıfır olur. Dolayısı burası fonksiyonun bir çukur noktası, minimum yapan optimal çözüm x^* noktasıdır

$x_1 = x_0 - \eta \frac{df(x_0)}{dx}$ (Güncelleme miktarı $\Delta x_o = -\eta \frac{df(x_0)}{dx} > 0$ pozitif, çözüm noktası sağa doğru güncellenir.)

$x_2 = x_1 - \eta \frac{df(x_1)}{dx}$ (Güncelleme miktarı $\Delta x_1 = -\eta \frac{df(x_1)}{dx} > 0$ pozitif, çözüm noktası sağa doğru güncellenir.)

$x_3 = x_2 - \eta \frac{df(x_2)}{dx}$ (Güncelleme miktarı $\Delta x_2 = -\eta \frac{df(x_2)}{dx} > 0$ pozitif, çözüm noktası sağa doğru güncellenir.)

…

..

$x_{n+1} = x_n - \eta \frac{df(x_n)}{dx}$ (Güncelleme miktarı $\Delta x_n = -\eta \frac{df(x_n)}{dx} > 0$ pozitif, ancak eğim azaldığı için çözüm noktası sağa doğru daha küçük adımla güncellenir.)

Eğimin yüksek olduğu noktalarda güncelleme miktarı yani x'deki değişim yüksektir. Yukarıdaki şekilde görüleceği üzere x_i iterative çözümü dip noktasına yaklaştıkça eğim azalacaktır ve Δx_i adımları küçülecektir.

Eğer, sürekli ve $f(x)$ şekilde verildiği gibi bir konveks fonksiyon ve η öğrenme katsayısı yeterince küçük ise

$$-\eta\frac{df(x_0)}{dx} > -\eta\frac{df(x_1)}{dx} > -\eta\frac{df(x_2)}{dx} > -\eta\frac{df(x_3)}{dx} > \ldots > -\eta\frac{df(x_{n+1})}{dx} \ldots \to 0$$

Dolayısı ile güncelleme miktarları $|\Delta x_o| > |\Delta x_1| > |\Delta x_2| > \ldots > |\Delta x_n| \ldots \to 0$

olması nedeni ile x_n'deki adımlar sürekli küçülür. Buna yakınsama (convergence) adı verilir. (Yakınsama konusunu, metrikler ve komşuluk konusunda tartışmıştık.) Eğer minimum noktaya ulaşılırsa türevi $\left|\frac{df(x)}{dx}|_{x^*}\right| = 0$ olur ve güncelleme miktarı

$\Delta x = 0$ olur. Algoritma burada kendiliğinden durur. Ancak, adımlar çok küçülürse algoritma $\left|\frac{df(x)}{dx}|_{x^*}\right| = 0$ ulaşması çok fazla zaman alır. Bu durumda, durdurma kriteri ile uygulanarak minimum noktanın yakın bir komşuluğunda gradyan iniş algoritması durdurulabilir.

Not: Makine öğrenmesinde bir öğrenme hata fonksiyonun minimum noktasına yakınsayan çözümler aranır ve bu işlem eğitim olarak adlandırılır.

Dikkat: Nümerik çözümler optimal noktaya yakınsamayı sağlar. Optimal noktayı bulmayı garanti etmez. Bu nedenle bir komşuluk ile ifade edilen durdurma kriteri kullanılabilir.

Örnek: min $f(x) = (x-1)^2 + (x-2)^2$ fonksiyonu gradyan iniş yöntemi ile iteratif çözümünü elde ediniz. $x_0 = 0.1$ başlangıç noktasından $\eta = 0.01$ öğrenme katsayısı için rekürsive çözümü bir kaç defa çözünüz. Matlab ile bilgisayar uygulamasını yapınız.

$f(x) = (x-1)^2 + (x-2)^2$ => $\frac{df(x)}{dx} = 2(x-1) + 2(x-2) = 4x - 6$

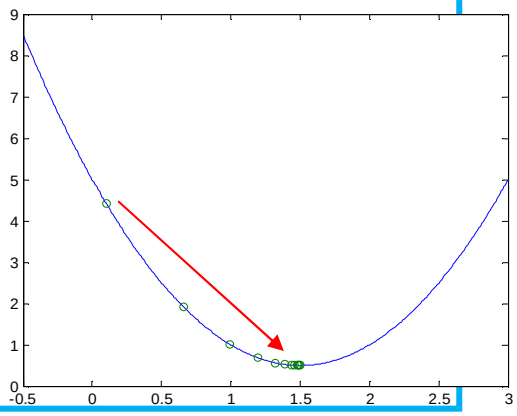
$x_{n+1} = x_n - \eta\frac{df(x_n)}{dx}$ => $x_{n+1} = x_n - 0.01.(4x_n - 6)$

$x_0 = 0.1$ alınırsa $x_0 = 0.1$ => $x_1 = x_0 - 0.01.(4x_0 - 6) = 0.1 - 0.01(4.0.1 - 6) = 0.156$

$x_1 = 0.156$ => $x_2 = x_1 - 0.01.(4x_1 - 6) = 0.156 - 0.01(4.0.156 - 6) = 0.209$,

$x_3 = 0.261$, $x_4 = 0.310 \ldots, x_{99} = 1.474, x_{100} = 1.475$

```
% E=(x-1)^2+(x-2)^2
% min E gradyan iniş ile çözümü
clear all;
xa=-0.5:0.01:3;
ft=(xa-1).^2+(xa-2).^2;
n=0.1; %0.1; % öğrenme katsayısı n=0.55 kararsız. n=0.49
%salınım
x=0.1; % Başlangıç çözümü
for i=1:20
xh(i)=x;
% E'nin birinci türevi E'=4*xa-6;
xs=x-n*(4*x-6);
x=xs;
% Yakın komşuluğa göre durdurma kriteri
Dx=abs(xh(i)-xs);
if Dx<0.0001
fprintf('%d. itersyonda Dx= %f . Durdurma kriteri etkin \n',i,Dx);
break;
end
end
% Sonuçların raporlanması
fprintf('Toplam iterasyon sayısı %d \n',i);
fprintf('x değerleri %f \n',xh);
fmin=(xs-1).^2+(xs-2).^2;
fprintf('Ulaşılan min E %f \n',fmin);
% Grafik olarak çizimler
figure(1)
plot(xa,ft,xh,(xh-1).^2+(xh-2).^2,'o')
xlabel('x')
ylabel('f')
grid
fi=(xh-1).^2+(xh-2).^2;
% Grafik olarak çizimler
figure(2)
plot(1:length(fi),fi,'o-')
xlabel('iterasyon')
ylabel('f')
grid
```



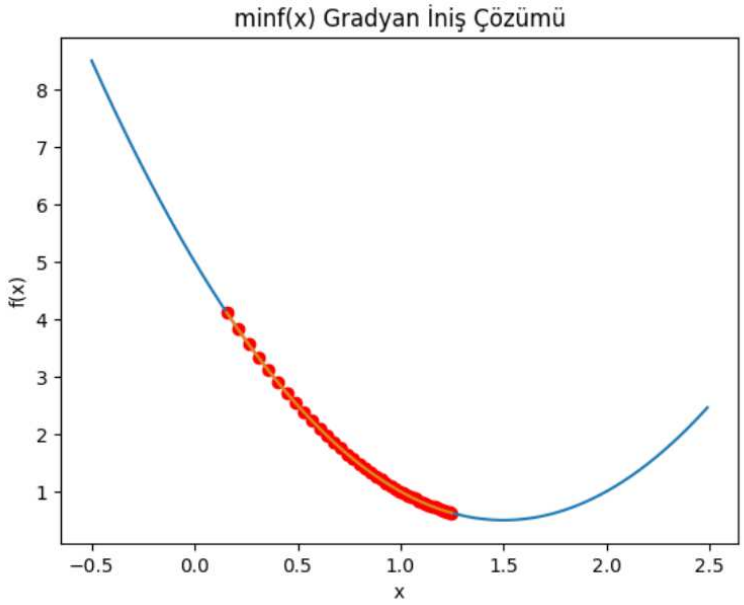
Python ile kodlama örneği

```
-----------------------------------------------------
# Gradyan iniş optimizasyonu için örnek bir kodlama
# minf(x) in gradyan iniş ile çözümü
import numpy as np
import matplotlib.pyplot as plt

# Optimize edilen f(x) fonksiyonu
def fx(x):

  # Optimize edilecek fonksiyonu burada tanımla
  f = (x-1)**2+(x-2)**2
  return f

# Optimize edilen f(x) fonksiyonun türevi
def Diff_fx(x):

  # Optimize edilecek fonksiyonun türevini burada tanımla
  f = 4*x-6
  return f

def main():

  # Başlangıç parametreleri
```

```
  x = 3.5 # 0.1 başlangıç noktası solda ve 3.5 başlangıç sağda
  ogrenme_katsayisi = 0.1 # 0.1 salınımsız yakınsama, 0.4 salınımlı yakınsama, 0.6 k
ararsızlık
  iterasyon_sayisi = 100
  DurdurmaEsik=0.0001 # Durma kriteri için eşik değeri
  costs = []
  iterasyon = []
  x_deger = []
  costs.append(fx(x))
  x_deger.append(x)
  iterasyon.append(0)

  # Estimation of optimal parameters
  for i in range(iterasyon_sayisi):
    # Fonksiyonun mevcut değeri
    y = fx(x)
    # Fonksiyonun türevinde x değerini kullan
    Turev_fx = Diff_fx(x)
    # Gradyan iniş güncelleme. (x değeri güncellenir)
    x = x - (ogrenme_katsayisi * Turev_fx )

    # Fonksiyonun yeni değeri
    y_yeni = fx(x)



    # Durdurma kriteri: Fonksiyonun yeni değeri, mevcut değerinin belli bir
    # komsuluğunda kalıyor ise itersayonu durdur. Değişim küçük ise durur.
    if abs(y_yeni-y)<=DurdurmaEsik:
      break

    costs.append(y_yeni)
    iterasyon.append(i)
    x_deger.append(x)
    # Her iterasyonda sonuçlar ekrana basılır
    print(f"Iteration: {i+1}, x={x}, f(x)= {y_yeni}")

  fx_ciz = []
  x_ciz = []
  for j in range(400):
    x_ciz.append(-0.5+j/100)
    fx_ciz.append(fx(-0.5+j/100))

  # Sonuçların grafik çizimi
  plt.figure()
  plt.plot(x_ciz, fx_ciz)
  plt.plot(x_deger, costs)
  #plt.plot(iterasyon, costs)
  plt.scatter(x_deger, costs, marker='o', color='red')
  plt.title("minf(x) Gradyan İniş Çözümü")
  plt.ylabel("f(x)")
  plt.xlabel("x")
  plt.show()

  plt.figure()
  plt.plot(iterasyon, costs)
  plt.scatter(iterasyon, costs, marker='o', color='red')
  plt.title("Her iterasyonda F(x) değeri")
  plt.ylabel("f(x)")
  plt.xlabel("iterasyon")
  plt.show()

if __name__=="__main__":
  main()
```

Elde edilen sonuçlar

```
Iteration: 1, x=0.156, f(x)= 4.112672
Iteration: 2, x=0.20976, f(x)= 3.8294385152
Iteration: 3, x=0.2613696, f(x)= 3.5684105356083196
Iteration: 4, x=0.310914816, f(x)= 3.3278471496166278
Iteration: 5, x=0.35847822336, f(x)= 3.106143933086684
Iteration: 6, x=0.40413909442560003, f(x)= 2.901822248732688
Iteration: 7, x=0.44797353064857603, f(x)= 2.7135193844320455
Iteration: 8, x=0.490054589422633, f(x)= 2.539979464692572
Iteration: 9, x=0.5304524058457277, f(x)= 2.380045074660675
Iteration: 10, x=0.5692343096118986, f(x)= 2.2326495408072784
Iteration: 11, x=0.6064649372274227, f(x)= 2.0968098168079874
Iteration: 12, x=0.6422063397383257, f(x)= 1.971619927170241
Iteration: 13, x=0.6765180861487927, f(x)= 1.8562449248800945
Iteration: 14, x=0.709457362702841, f(x)= 1.7499153227694948
Iteration: 15, x=0.7410790681947274, f(x)= 1.651921961464367
Iteration: 16, x=0.7714359054669383, f(x)= 1.5616112796855606
Iteration: 17, x=0.8005784692482608, f(x)= 1.4783809553582117
Iteration: 18, x=0.8285553304783304, f(x)= 1.4016758884581284
Iteration: 19, x=0.8554131172591972, f(x)= 1.3309844988030108
Iteration: 20, x=0.8811965925688293, f(x)= 1.2658353140968552
Iteration: 21, x=0.9059487288660761, f(x)= 1.2057938254716616
Iteration: 22, x=0.9297107797114331, f(x)= 1.150459589554683
Iteration: 23, x=0.9525223485229758, f(x)= 1.0994635577335958
Iteration: 24, x=0.9744214545820568, f(x)= 1.0524656148072817
Iteration: 25, x=0.9954445963987745, f(x)= 1.0091523106063915
Iteration: 26, x=1.0156268125428234, f(x)= 0.9692347694548502
Iteration: 27, x=1.0350017400411105, f(x)= 0.9324467635295901
Iteration: 28, x=1.053601670439466, f(x)= 0.8985429372688702
Iteration: 29, x=1.0714576036218872, f(x)= 0.867297170986991
Iteration: 30, x=1.0885992994770117, f(x)= 0.8385010727816109
Iteration: 31, x=1.1050553274979313, f(x)= 0.8119625886755326
Iteration: 32, x=1.120853114398014, f(x)= 0.7875047217233708
Iteration: 33, x=1.1360189898220936, f(x)= 0.7649643515402584
Iteration: 34, x=1.1505782302292098, f(x)= 0.7441911463795021
Iteration: 35, x=1.1645551010200414, f(x)= 0.7250465605033491
Iteration: 36, x=1.1779728969792398, f(x)= 0.7074029101598867
Iteration: 37, x=1.1908539811000702, f(x)= 0.6911425220033514
Iteration: 38, x=1.2032198218560675, f(x)= 0.6761569482782888
Iteration: 39, x=1.2150910289818249, f(x)= 0.6623462435332708
Iteration: 40, x=1.226487387822552, f(x)= 0.6496182980402623
Iteration: 41, x=1.2374278923096498, f(x)= 0.6378882234739058
Iteration: 42, x=1.2479307766172638, f(x)= 0.6270777867535515
```



Örnek: min $f(x) = x^4 - 6x^3 - 6x^2 + 6x + 1$ fonksiyonu gradyan iniş yöntemi ile iteratif çözümünü elde ediniz. $x_0 = 0.1$ bağlangıç noktasından $\eta = 0.01$ öğrenme katsayısı için rekürsive çözümü bir kaç defa çözünüz. Matlab ile bilgisayar uygulamasını yapınız.

$$f(x) = x^4 - 6x^3 - 6x^2 + 6x + 1$$

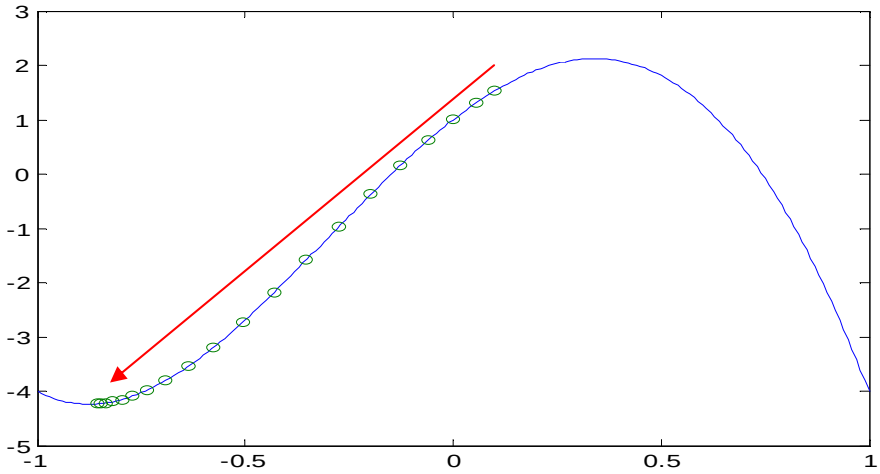
$$\frac{df(x)}{dx} = 4x^3 - 18x^2 - 12x + 6$$

$$x_{n+1} = x_n - \eta \frac{df(x_n)}{dx} \Rightarrow\ x_{n+1} = x_n - 0.01(4x_n^{\ 3} - 18x_n^{\ 2} - 12x_n + 6)$$ elde edilir.

$x_0 = 0.1, x_1 = 0.0538, x_2 = 0.007, x_3 = -0.0592 \ldots$

Çözüm bilgisayarda uygulanırsa,

```
% y=x^4-6*x^3-6*x^2+6*x+1 denkleminin min y gradyan
desken ile çözümü
clear all;
xa=-1:0.01:1;
y=xa.^4-6*xa.^3-6*xa.^2+6*xa+1;

n=0.01; % öğrenme katsayısı
x=0.1; % Başlangıç çözümü
for i=1:20
    xh(i)=x;
    xs=x-n*(4*x^3-18*x^2-12*x+6);
    x=xs;
end
disp(xh)
figure(1)
plot(xa,y,xh,xh.^4-6*xh.^3-6*xh.^2+6*xh+1,'o')
```



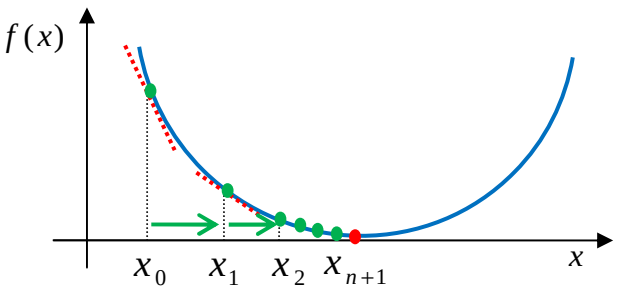
Öğrenme Katsayısının Önemi:

Öğrenme katsayısının etkisini görmek için x_i değişkenin güncelleme miktarını

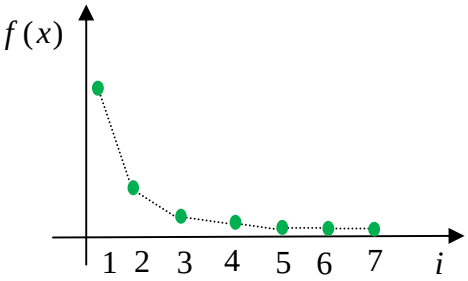
$$\Delta x_i = -\eta \frac{df(x_i)}{dx}$$

göz önünde tutalım. Burada η öğrenme katsayısı ne kadar büyük seçilirse güncelleme miktarı orantılı olarak büyüyecektir. Bu optimizasyonun minimum noktaya daha az adımda yaklaşmasını sağlayabilir diğer bir ifade ile optimizasyon hızını artırabilir. Ancak, öğrenme katsayısını aşırı büyük seçmek çözüme yakınsama yerine ıraksamaya da neden olabilir. Bu duruma çözümün kararsız olması adı verilir. Aşağıdaki şekillerde olası durumlar temsili olarak gösterilmiştir.

1. Salınımsız Yakınsama: Öğrenme katsayısı yeterince küçük ise minimum noktaya sürekli azalan adımlar ile yaklaşılır.

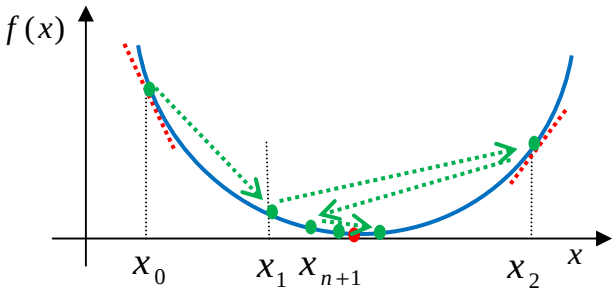


Çözüm noktaları her iterasyonda fonksiyon üzerinde azalan adımlar ile minimum noktaya (kırmızı nokta) yaklaşır.

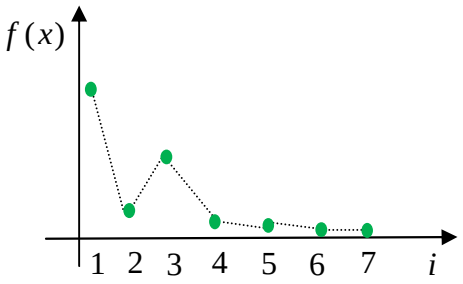


Her iterasyonda (hesaplama adımında) fonksiyon değerleri çizilirse fonksiyon değerlerinin sürekli azaldığı ve asimptotik olarak yakınsadığı görülür.

2. Salınımlı Yakınsama: Öğrenme katsayısı, salınımsız yakınsamaya imkân verecek kadar küçük değilse minimum noktaya salınımlı azalan adımlar ile yaklaşılır.

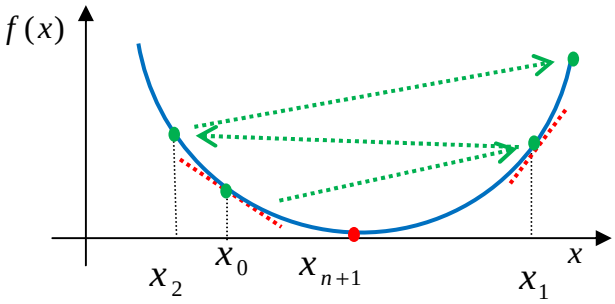


Çözüm noktaları yeterince küçük olmadığı için mimimum noktayı aşıp diğer tarafa düşebilir. Ancak eğimi takip ederek tekrar döner . Burada salınımlı hareketini küçülterek minimum noktaya yaklaşır. Çözüme salınımlı yakınsar.

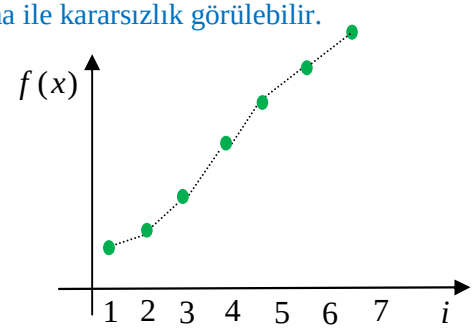


Her iterasyonda (hesaplama adımında) fonksiyon değerleri çizilirse fonksiyon değerlerinin minimum nokta etrafında salınımlar nedeni ile dalgalandığı görülebilir.

3. Kararsızlık durumu (Iraksama): Öğrenme katsayısı, yakınsamaya izin vermeyecek derecede büyük seçilirse minimum noktadan sürekli uzaklaşma ile kararsızlık görülebilir.



Çözüm noktaları yeterince küçük olmadığı için mimimum noktayı aşıp diğer tarafa düşebilir. Ancak eğimi takip ederek tekrar döner . Ancak öğrenme adımı çok büyük olursa minimumdan daha uzak bir noktaya düşür.Bu değeri artıran salınımlı hareketi öğrenme katsayısı çok büyük olduğunda görülebilir.



Her iterasyonda (hesaplama adımında) fonksiyon değerleri çizilirse fonksiyon değerlerinin minimum nokta etrafında artan salınımlar nedeni ile büyüyebilir

Gradyan iniş çözümünün kararlık ve yakınsaklık koşulları uygulamalı matematik alanında çalışılmış ve belirlenmiştir. Makine öğrenmesinin temel bir konusu olmadığı için bu derste kararlılık konusu detaylı olarak incelenmeyecektir. Bu ders için olası durumlar ve etkilerinin grafiksel olarak anlaşılması yeterlidir. Uygulama açısından öğrenme katsayına bağlı durumların ve etkilerinin bilinmesi yeterli olacaktır.