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

Öğrenme, Optimizasyon ve İhtiyaç Duyulan Temel Matematiksel Temeller II:

Metrik Uzayları ve Makine Öğrenmesi İçin Önemli Metrikler:

Metrik uzayı üzerinde metrik tanımlanabilen kümeler ile oluşturulur. Metrik ise bu kümenin iki elemanı arasındaki uzaklığı ifade etmek için kullanılan bir fonksiyondur.

Metrik uzayları (X, d) sıralı ikilisi ile tanımlanır. Burada X bir kümedir ve d ise metrik uzayı üzerinde tanımlanmış bir metriktir. Şöyle ifade edilir.

$$d : X \times X \rightarrow R^{+} \cup \{0\}$$

Makine öğrenmesi için X kümesi verilen ait olduğu ve istenen çözümün elde edileceği arama uzayını ifade etsin. Bu arama uzayı üstünde, özellik(feature) olarak da ifade edebileceğimiz iki veri noktası $k_1, k_2 \in X$ olsun. Bu iki noktanın birbirine benzerliği, iki nokta arasındaki mesafeyi ifade eden uzaklık metriği $d(k_1, k_2)$ ile ifade edilebilir. Bu uzaklık ölçüm fonksiyonu, kartezyen çarpımı kümesinden seçilen her (k_1, k_2) ikilisi için bir sıfır dahil pozitif reel sayı dönmelidir. ($d : X \times X \rightarrow R^{+} \cup \{0\}$ tanımında olduğu gibi.)

Bir d fonksiyonu metrik ise şu özellikleri sağlamak zorundadır.

$x, y, z \in X$ noktalar olsun.

* $d(x, y) \geq 0$ (Negatif değer almaz)

* $d(x, x) = 0$ dolayısı ile $d(x, y) = 0 \Rightarrow x = y$ (Uzaklık sıfır ise çözüm noktaları aynıdır.)

(Dolayısı ile $d(x, y) > 0$ ise $x \neq y$'dir. $d(x, y)$ değerine bağlı olarak çözüm noktaları birbirinden uzaklaşır. Bu durum makine öğrenmesi açısından metrik uzayında ölçülen uzaklığın iki veri noktanın benzerliğini ifade etmesini sağlar. Uzaklık azaldıkça veriler birbirine daha fazla benzer.)

* $d(x, y) = d(y, x)$ (Metrik(Uzaklık) fonksiyonu simetriktir. Sıralamanın değişmesi sonucu değiştirmez.)

* $d(x, z) \leq d(x, y) + d(y, z)$ (Metrik fonksiyonu üçgen kuralını sağlar.)

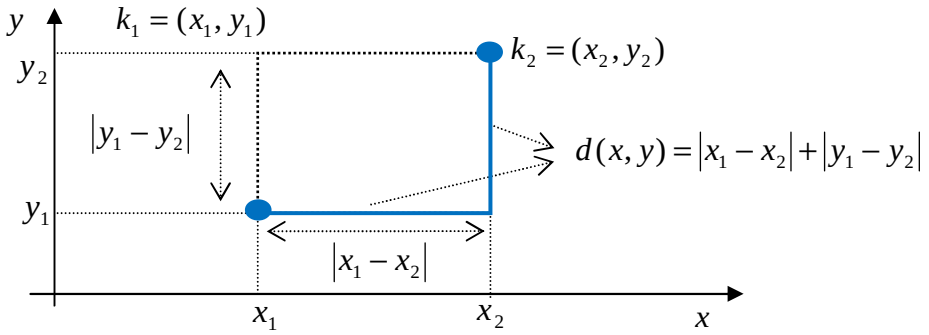
Bilgisayar bilimlerinde metrik(uzaklık) fonksiyonu metrik uzayına göre verilerinin benzerlikleri veya farklılıklarını ölçmek için kullanılır.

Bilgisayar Bilimlerinde Kullanılan Başlıca Uzaklık Metrikleri,

1- **Mutlak Fark Metriği (Absolute Difference)**: p boyutlu metrik uzayında $x=(x_1,x_2,x_3,..,x_p)$ ve $y=(y_1,y_2,y_3,..,y_p)$ için $d(x,y)=|x-y|=\sum_i^p |x_i-y_i|$ olarak ifade edilir.

Metrik uzayının her hangi bir noktasının orijine (sıfır noktasına) uzaklığı norm olarak adlandırılır. Dolayısı ile noktalardan biri orijin noktası alınır ($y=(0,0,0,..,0)$) ise $d(x,y)=|x-y|=|x|=\sum_i^p |x_i|=\|x\|_1$ => $\|x\|_1=|x|=\sum_i^p |x_i|$ (Mutlak değer normu olarak adlandırılır. Bu L1-norm olarak da bilinir.)

Örnek: 2 boyutlu (R^2) kartezyen koordinat sisteminde tanımlanan $k_1=(x_1,y_1)$ ve $k_2=(x_2,y_2)$ noktaları için, $d(k_1,k_2)=|x_1-x_2|+|y_1-y_2|$ ile ifade edilen mutlak fark metriği (city-block metric (Manhattan distance)) olarak anılır. Aşağıda mavi ile gösterilen uzunluklar toplamı mutlak fark metriğini verir.



Matlab ve Phyton kodu aşağıda görüldüğü gibi yazılabilir. (Phyton kodu sonucu web'de online Phyton derleyici sitelerinde kod kopyalanarak görülebilir.)

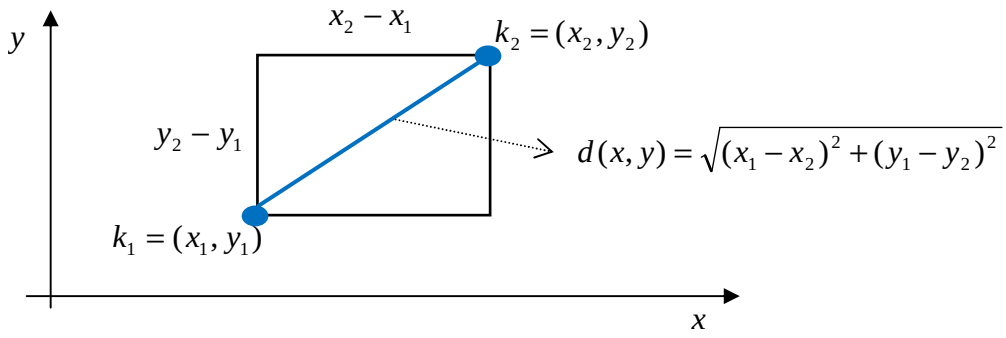
```
% x ve y vektörleri tanımlansın
k1=[1 2];
k2=[3 4];
% d(x,y) mutlak fark metriği (Manhattan distance)
d=sum(abs(k1-k2));
```

```
import numpy as np
V= np.array([1, 2])
Y= np.array([3, 4])
d=sum(np.abs(V-Y))
print('V=',V);print('Y=',Y);print('d=',d);
```

2- Euclid Uzaklığı (Euclid Distance) Metriği: p boyutlu metrik uzayı içinde tanımlanan $x=(x_1,x_2,x_3,..,x_p)$ ve $y=(y_1,y_2,y_3,..,y_p)$ noktaları için $d(x,y)=\|x-y\|=\sqrt{\sum_i^p (x_i-y_i)^2}$ olarak ifade edilir.

$y=(0,0,0,..,0)$ ise $d(x,y)=\|x-y\|=\|x\|=\sqrt{\sum_i^p x_i^{\,2}}=\|x\|_2$ => $\|x\|_2=\|x\|=\sqrt{\sum_i^p x_i^{\,2}}$ (Euclid normu olarak adlandırılır. L2-norm olarak da bilinir. Eğer x bir vektör ifade ediyor ise Euclid normu vektörün büyüklüğünü ifade eder.)

Örnek: 2 boyutlu R^2 kartezyen koordinat sisteminde tanımlanan $k_1=(x_1,y_1)$ ve $k_2=(x_2,y_2)$ noktaları için, Euclid uzaklığı $d(k_1,k_2)=\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}$ ile ifade edilir. Bu noktalar arasındaki doğrusal uzaklığı verir. Aşağıda mavi renkli uzunluk Euclid uzaklığıdır. (Hipotenüs uzunluğunun Pisagor teorimine göre yazılabileceğini görünüz.)



Matlab kodlaması şöyle yapılabilir.

```
% x ve y vektörleri tanımlansın
x=[1 2];
y=[3 4];
% d(x,y) Euclid distance
d=sqrt(sum((x-y).^2));
```

```
import numpy as np
V= np.array([1, 2])
Y= np.array([3, 4])
d= np.sqrt(sum(np.square(V-Y)))
print('V=',V);print('Y=',Y);print('d=',d);
```

3- Hamming Uzaklığı (Euclid Distance): p boyutlu binary (ikili) metrik uzayı içinde tanımlanan $x=(x_1,x_2,x_3,..,x_p)$ ve $y=(y_1,y_2,y_3,..,y_p)$ binary kodları için $(x_i,y_i\in\{0,1\})$ noktaları için $d(x,y)=\sum_i^p z_i$,

$$z_i=\begin{cases}0 & x_i=y_i\\ 1 & x_i\neq y_i\end{cases}$$

olarak ifade edilir. Hamming uzaklığı, ikili sayı formatında verilen x ve y kodları arasındaki farklı bitlerin sayısını verir.

Örnek: x=1000111 ve y=1100110 noktaları arasındaki Hamming uzaklığını bulunuz.

Farklı bit sayısı Hamming uzaklığıdır. x=1**0**00111 ve y=11001**1**0 farklı bitler iki adettir. $d(x,y)=2$.

Örnek: $k_1=(1,2)$ ve $k_2=(3,4)$ noktaları verilsin. Mutlak fark uzaklığını ve Euclid uzaklıklarını hesaplayınız. Sonuçlara göre $k_1=(1,2)$ ve $k_2=(3,4)$ arasındaki mesafeyi hangisi daha büyük ölçer tartışınız.

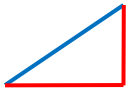
Mutlak fark uzaklığını

$$d(k_1,k_2)=|k_1-k_2|=\sum_i^p|x_i-y_i|=|x_1-y_1|+|x_2-y_2|=|1-3|+|2-4|=2+2=4$$

Euclid uzaklığı

$d(k_1,k_2)=\|k_1-k_2\|=\sqrt{(x_1-y_1)^2+(x_2-y_2)^2}=\sqrt{(1-3)^2+(2-4)^2}=\sqrt{2^2+2^2}=2\sqrt{2}$ ile ifade edilir.

Burada $4>2\sqrt{2}$ olduğu için mutlak fark uzaklığının daha büyük ölçtüğü görülür. Bu durum aslında metrik uzayının üçgen kuralı ($d(x,z)\le d(x,y)+d(y,z)$) ile doğrudan ilişkilidir.



Vektör Normları:

Vektör uzayında tanımlı bir $x=(x_1,x_2,x_3,..,x_p)$ vektörünün orijine olan uzaklığı ifade eder. Metriklerin bütün özelliklerini miras alır.

Bazı temel normlar:

$x=(x_1,x_2,x_3,..,x_p)$ bir vektör olmak üzere

L0-norm: Bir vektördeki sıfırdan farklı eleman sayısıdır. $\|x\|_0 = \sum_i^n h_i$, $h_i = \begin{cases} 0 & x_i = 0 \\ 1 & x_i \neq 0_i \end{cases}$

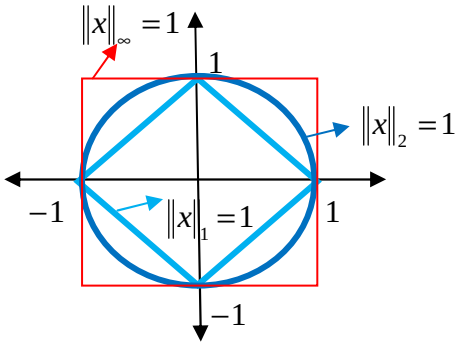
L1-norm (Mutlak değer normu) : $\|x\|_1 = \sum_i^n |x_i|$

L2-norm (Euclid normu) : $\|x\|_2 = \sqrt{\sum_i^n x_i^{\,2}}$

Lp-norm (Euclid normu) : $\|x\|_p = \left(\sum_i^n x_i^{\,p} \right)^{\frac{1}{p}}$

L∞-norm (Euclid normu) : $\|x\|_\infty = \max\{|x_1|, |x_2|, ..., |x_n|\}$

Normların geometrik yorumunu görmek için $\|x\|_1 = 1$, $\|x\|_2 = 1$ ve $\|x\|_\infty = 1$ denklemlerini çözüm kümelerini aşağıdaki grafikte inceleyelim.



Örnek: $x = (1,1)$ vektörü için normları hesaplayınız.

$\|x\|_0 = 2$ (vektörde sıfırdan farklı bileşen sayısı)

$\|x\|_1 = \sum_i^2 |x_i| = |x_1| + |x_2| = |1| + |1| = 2$

$\|x\|_2 = \sqrt{\sum_i^2 x_i^{\,2}} = \sqrt{1^2 + 1^2} = \sqrt{2}$

$\|x\|_3 = \left(\sum_i^2 x_i^{\,3} \right)^{\frac{1}{3}} = \left(1^3 + 1^3\right)^{\frac{1}{3}} = (2)^{\frac{1}{3}}$

$\|x\|_\infty = \max\{|1|, |1|\} = 1$

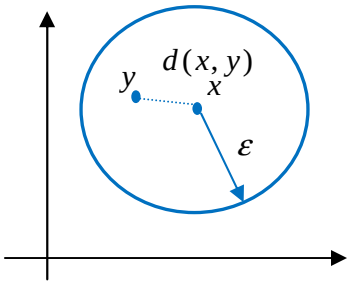
Komşuluk Kavramı ve Yakınsama:

Arama uzayı üzerinde bir çözüm noktasına bir uzaklık metriğine göre yakın kabul edilen çözümler kümesi bu noktanın komşularıdır.

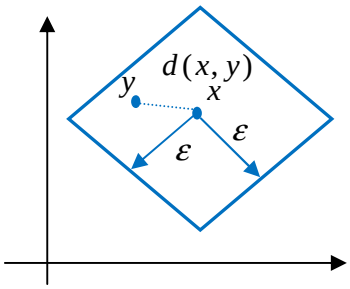
$$N_\varepsilon = \{y \in X \mid d(x,y) < \varepsilon\}$$

ifadesi bir $x \in X$ noktasının ε komşuluğunu ifade eder ve arama metriği $d(x,y) < \varepsilon$ sağlayan bütün $y \in X$ noktalarının kümesini ifade edilir. Diğer bir ifade ile $x \in X$ notasına uzaklığı $d(x,y)$ metriğine göre ε'dan küçük kaldığı bölgeyi ve içindeki bütün $y \in X$ noktalarını ifade eder.

Örnek: Aşağıda R^2 de tanımlı bir x çözüm noktasının ε komşuluğu, Euclid uzaklığı metriğine göre bir dairesel bölgenin içindeki alan ve mutlak fark metriğine göre bir karesel komşuluk alanı tanımlar.



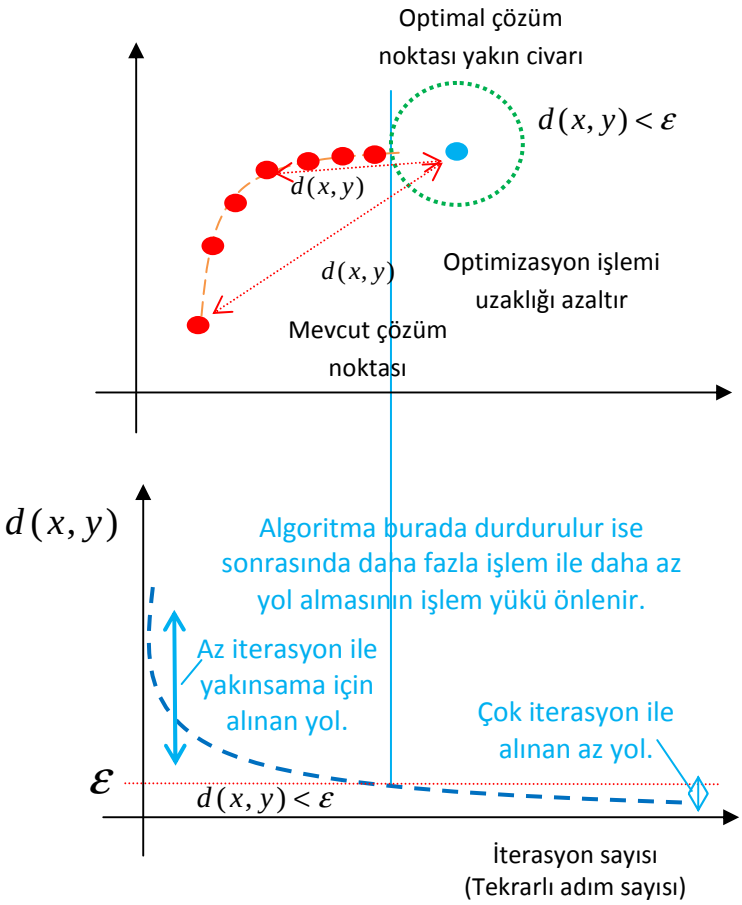
Euclid mesafesine göre 2D komşuluk bir dairesel bölge tanımlar.



Mutlak fark metriğine göre 2D komşuluk bir dairesel bölge tanımlar.

Yakınsama(Convergence): Makine öğrenmesinde yada nümerik hesaplamada bir çözüm noktasının belirlenmiş bir $\varepsilon > 0$ komşuluğuna ulaşılması $N_\varepsilon = \{y \in X \mid d(x,y) < \varepsilon\}$ yakınsama (Convergence) olarak isimlendirilir. Yakınsama, bir çok nümerik arama algoritmasında durdurma koşulu olarak kullanılır.

Eğer y çözümü, x aranan noktanın yakın civarına ulaşma durumu (yakınsama durumu) $d(x,y) < \varepsilon$ ise test edilir ve koşul sağlanır ise algoritma durdurulabilir. Nümerik yada metasezgisel optimizasyon kullanılması halinde bazen $d(x,y) = 0$ durumuna çok zor ulaşılabilir veya hiç bir zaman ulaşılamayabilir. Bu durum gereksiz yüksek işlem maliyetlerine yol açar. Bunun yerine optimal çözüme bir uzaklık metriğine göre yeterince yakın duruma ulaşmak yani ε komşuluğuna yaklaşılması ($d(x,y) < \varepsilon$) yeterli olur ve bu aşamada algoritma durdurulabilir. Böylece gereksiz işlem maliyetlerinden algoritma kurtulur. Dolayısı ile bazı durumlarda sadece optimal çözümün yakın civarına gelinmesi diğer bir ifade ile çözümün yakınsaması yeterli olabilir ve algoritma bu aşamada durdurulursa çözümün işlem maliyeti oldukça düşürebilir.



Örnek: y çözümünün 0 noktasının mutlak fark metriğine göre 0.01 komşuluğuna ulaştığı zaman programı durduran bir durdurma kriterini matematiksel olarak elde ediniz ve Matlab kodu ve Python kodunu yazınız.

$d(0,y) < 0.01 \Rightarrow |0-y| < 0.01$ (Mutlak fark metriği kullanılırsa)$\Rightarrow |y| < 0.01$ elde edilir.

```
% Matlab
% Rekürsif çözüm döngüsü içinde durdurma
if abs(y)<0.01
fprintf('Yeterince optimal değer bulundu');
break;
end
```

```
# Phyton
# Rekürsif çözüm döngüsü içinde durdurma
if abs(y) < 0.01:
  break
print('Yeterince optimal değer bulundu')
```

Optimizasyon Kavramı ve Temel Bilgiler

Matematiğin uygulamalı bilimlerde en çok ihtiyaç duyulan konusu optimizasyondur. Optimizasyon işlemine matematik alanında “programlama” adı verilir. Örneğin, lineer fonksiyonların optimizasyonu konusu "lineer programlama" başlığı altında incelenir. Basitçe ifade edilmesi gerekirse bir çözüm kümesi veya çözüm uzayı içinde en iyi (en uygun) çözümü arama ve bulma çabası optimizasyon olarak adlandırılabilir.

Optimizasyon, belirlenmiş kıstaslara göre en uygun değeri bulma problemini çözer. Optimizasyon algoritmaları Türkçede "en iyileme" yöntemleri olarak da isimlendirilir.

Örnek: Birikimlerimizi maksimize etmek isteriz, bu nedenle hayatımızda bazı düzenlemeler yaparız. Mesela, gereksiz gördüğümüz harcamaları kısmaya çalışırız ve giderlerimizi minimize ederiz. Diğer bir çözüm gelirlerimizi artırmaya çabalarız yani gelirlerimizi maksimize ederiz. Bu süreç bireysel birikimlerimizi artırmak için bireysel ekonomimizi "optimize etme" (iyileme) çabası olarak görülebilir. Bu türden ekonomik optimizasyon herkesin günlük hayatında doğal olarak yaptığı süreçtir.

Optimizasyon problemi, mühendislikte uygulamaları açısından matematiksel olarak tanımlanmış bir amaç fonksiyonun (Objective function), maksimum veya minimum noktasını bulma problemine dönüşür.

* Hesapsal zekada, doğru çözüme oluşabilmek için problem bir optimizasyon problemi olarak ifade edilir ve bu problemin matematiksel çözümü optimal yani en iyi çözümü bize sunar. Bu nedenle optimizasyon hesapsal zekanın temel bir matematiksel aracıdır.

$E = \|X_d - X_t\|_2$ eğer $E \to 0$ giderse çözüm tahmini doğru çözüme gider . $X_t \to X_d$

O halde $\min_{X_t} E$ işlemi hesapsal olarak doğru çözüme yakınsama eğilimi gösterir. Burada doğru çözüme yakınsama eğilimi uygulama içinde zeka olarak karşılık bulur.

Amaç fonksiyonu (Objective function): Her optimizasyon problemi optimizasyon sürecinde amacımıza ne kadar yakın olduğumuzu ifade eden bir ölçüte yada metriğe ihtiyaç duyar. Bir değerleme ölçütü olarak sonucun amaca uygunluğu bir amaç fonksiyonu ile matematiksel olarak ifade edilir.

Örneğin, amaç fonksiyonu, mühendislikte uygulamalarının doğasına bağlı olarak bir maliyet fonksiyonu (Cost function), bir hata fonksiyonu (Error function) veya bir kayıp fonksiyonu (loss function) olarak ifade edilir. Bu durumda optimizasyon işlemi maliyet, hata veya kayıp fonksiyonunun minimum noktasını bulma işidir ve bir minimizasyon problemini ifade ederler. Bir doğruluk fonksiyonu ise optimizasyon bir maksimizasyon problemi ifade eder.

Bir fonksiyonun maksimum veya minimum noktalarına ekstremum noktaları (kritik noktaları) denir. Türevlenebilir (türevi her noktada alınabilir) ve sürekli bir $f(x)$ fonksiyonu için 4 farklı ekstremum noktası türü tanımlayabiliriz.

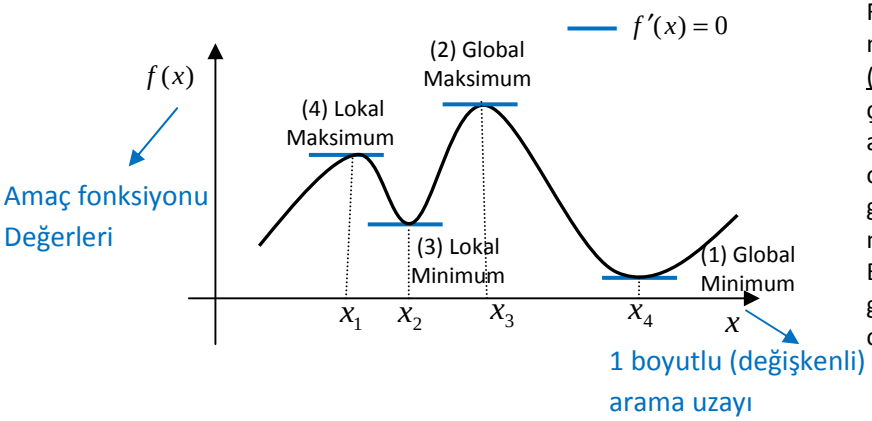
(i) **Global Minimum**: Fonksiyon değeri en küçük olan ektremum noktası global minimumdur. (Minimumların en küçüğü) Burası fonksiyonun en düşük değerli çukur(dip) noktasıdır.

(ii) **Global Maksimum**: Fonksiyon değeri en büyük olan ektremum noktası global maksimumdur. (Maksimumların en büyüğü) Burası fonksiyonun en yüksek değerli tepe(pik) noktasıdır.

(iii) **Lokal Minimum**: Fonksiyon değeri, global minimumdan daha büyük olan çukur noktalarıdır. Değerce, global minimumdan daha yüksek ancak çukur karakteristiğine sahiptir.

(iv) **Lokal Maksimum**: Fonksiyon değeri, global maksimumdan daha küçük olan tepe noktalarıdır. Değerce, global maksimumdan daha küçük tepe noktalarıdır.

x arama uzayı üzerinde tanımlı bir $f(x)$ amaç fonksiyonu aşağıda görülmektedir. Optimizasyon problemi çözümü olan ekstremum notaları $x_1, x_2, x_3, x_4, \in x$, arama uzayından gelir.



Fonksiyonun tanım kümesi çözüm noktalarının arandığı arama uzayını (search space) oluşturur. Optimal çözümler bu kümeden gelir. Yandaki amaç fonksiyonu, kar fonksiyonu olsa idi optimizasyon maksimizasyon gerçekleştirmeli idi. Bu aralık için global maksimum x_3 bir optimal çözüm olur. Eğer amaç fonksiyonu maliyet olsa idi global minimum yani x_4 optimal çözüm olurdu.

Optimizasyon Probleminin Matematiksel Olarak İfade Edilmesi:

Optimizasyon problemi genelde iki bileşen ile tanımlanır:

(i) Maksimize ve minimize edilecek bir amaç fonksiyonu,

(ii) Uygulamanın doğasının gereği olarak ortaya çıkan amaç fonksiyonu optimize edilirken anlamlı sonuçlar alabilmek için sağlanması gereken bazı sınırlamalar (kıstaslar- constraints) olabilir. Yazılan kısıt fonksiyonları,

* Matematiksel olarak arama uzayını sınırlayan ifadelerdir.

* Optimizasyon sürecinde kesinlikle uyulması gereken sınırları ifade eder,

* Genelde uygulamanın doğası gereği anlamlı(uygulanabilir) sonuçlar elde edilmesi için optimizasyon işlemi kısıtlanır,

Kıstaslar optimizasyon çözümlerinin gerçek dünya için daha geçerli ve anlamlı olması için kullanılır. Diğer taraftan arama uzayının sınırlanması bazı optimizasyon yöntemlerinin anlamlı çözümlere daha kolay ve çabuk ulaşmasını sağlayabilir.

Örneğin, gerçek sistem parametreleri sonlu bir değere sahip olabilir veya pratikte sınırlı bir aralık içinde değer alabilirler. Örneğin, karasal iklim olan bir bölgede hava sıcaklığı -10 ile 40 aralığında değişebilir. Optimizasyonun sonucunun gerçek sistemde uygulanabilir bir sonuç verebilmesi için arama uzayının sistem parametrelerinin değişim aralığına göre sınırlanması gerekir. Aksi halde, matematiksel olarak bulunan optimal çözümler, gerçek sistem için anlamlı veya uygulanabilir olmayabilir. Örneğin, yaşam maliyetlerinizin optimizasyonunda matematiksel olarak en optimal çözüm sıfır maliyettir. Ancak, bu gerçek hayatta olanaksız ve anlamsız bir çözümdür. Hayatımızı devam ettirebilmek için minimal gereksinimler (beslenme, barınma vs) karşılanmalıdır. Bunlar bir kıstas(sınırlayıcı) olarak optimizasyon probleminde tanımlanmalıdır.

(a) Minimizasyon problemi için,

$$\min f(x)$$
$$\text{S.t. } g_i(x) > 0, h_i(x) = 0, ..$$

(Burada S.t.; Subject to kısaltması "bağlı olmak üzere" anlamında)

(b) Maksimizasyon problemi için

$$\max f(x)$$
$$\text{S.t. } g_i(x) > 0, h_i(x) = 0, ..$$

Bu problemlerin optimal çözümleri genelde değişken üzerinde * ile (x^*) gösterilir.

Bir Parametreli Fonksiyonların Ekstremum Noktaları ve Kısıtsız Optimizasyonu

Sürekli ve türevlenebilir bir $f(x)$ fonksiyonun

(i) Ektrememum noktaları, $\frac{df(x_i)}{dx}=0$ denklemi çözülerek bulunur. Eğimi sıfır olan noktalar tepeler (maximum) veya çukurlar (minimum) olabilir. Hangisinin olduğunu anlamak için ikinci türev testi uygulanır.

(ii) Ektremum noktları ikinci türevde kullanılır. (İkinci türev testi)

(a) $\frac{d^2 f(x_i)}{dx^2}>0$ ise x_i noktası bir nokta bir lokal minimumdur.

(b) $\frac{d^2 f(x_i)}{dx^2}<0$ ise x_i noktası bir nokta bir lokal maksimumdur.

(c) $\frac{d^2 f(x_i)}{dx^2}=0$ ise bu test x_i noktası hakkında bir şey söylemez. Test bir sonuç vermez.

Örnek: Bir makine öğrenmesi modelinin eğitimi için öğrenme hatası fonksiyonu $E(w)=\frac{1}{2}w^2+2w+1$ olarak ifade edilen bir sistemin hatasını minimum yapan optimal w değerini ve hatanın minimum değerini hesaplayınız.

Optimizasyon problemi şöyle ifade edilir.

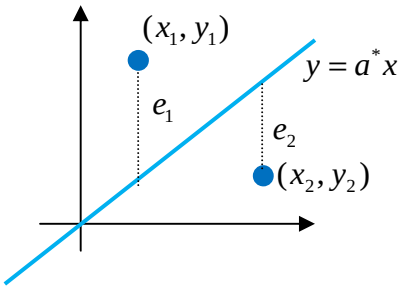
$\min E(w)$

$\frac{dE(w)}{dw}=w+2=0$ => $w=-2$ ektremum (kritik) noktadır. Buranın bir minimum yada maksimum olup olmadığını bilmiyoruz. İkinci türev testi ile bir minimum olup olmadığını görelim.

$\frac{d^2E(w)}{dw^2}|_{w=-2}=1>0$ olduğu için $w^*=-2$ bir minimum noktadır.

Minimum noktasında hatanın değeri $E(w=-2)=\frac{1}{2}(-2)^2+2(-2)+1=-1$

Örnek: $y = ax$ fonksiyonu yardımı ile T={ (1,4) , (3,2) } veri noktaları kümesinden en az karesel hata ile geçmesi için a nın değerini belirleyiniz?



T={ (1,4) , (3,2) } verileri için $y = ax$ modeli için ortalama karesel hatası yazılırsa (T kümesindeki her veri için istenen değerden $y = ax$ modelinin verdiği değerin farkı ile her veri için hata bulunur. Veri hatalarının karelerinin toplamı ile karesel hata elde edilir.

$e_1 = (4 - a.1)$,

$e_2 = (2 - a.3)$

$E(a) = \frac{1}{2}e_1^2 + \frac{1}{2}e_2^2 = \frac{1}{2}(4-a)^2 + \frac{1}{2}(2-a3)^2$ elde edilir. (1/2 çarpanı türevden gelen 2 çarpanı ile sadeleşir. Böylece 2 ile çarpım işlemini kaldırır.)

Bu fonksiyonu minimum yapan a değeri arıyoruz. O halde optimizasyon probleminin standart ifadesi

$\min E(a)$

şimdi bu tek değişkenli optimizasyon problemini çözmek için birinci türevi sıfıra eşitleyerek kritik noktaları bulalım.

$$\frac{dE(a)}{da} = -1(4-a) - 3(2-a3) = 0 \Rightarrow a = \frac{4+6}{1+9} = \frac{10}{10} = 1$$

Tek bir ekstremum noktasıdır. Karesel hatanın konveks bir fonksiyon olduğunu biliyoruz. Dolayısı ile elde edilen tek ektremum noktası minimum nokta olmalıdır. Yinede, bunu teyit etmek gerekiyor. Bunun için ikinci türevini hesaplayalım.

$\frac{d^2E(a)}{da^2} = -1+9 = 8 > 0$ dolayısı ile bir minimum noktadır.

O halde $\min E(a)$ probleminin optimal çözümü $a^* = 1$ dir.

Not: Pratikte öğrenen modelin ayarlanabilir parametre sayısı makine öğrenmesinde oldukça yüksektir. Analitik optimizasyon yöntemleri ile binlerce parametresi olan bir optimizasyon problemi çözmek zorlaşır. (Makine öğrenmesinin destek vektör makineleri (Support Vector Machine) analitik çözüm konusuna odaklanmış ve çözüm yöntemleri önermişti.)

Ancak, çok sayıda parametre içeren optimizasyon problemleri nümerik yöntemler ile bilgisayar ortamında daha kolaylıkla çözülür. Gelecek bölümde bu amaca dönük olarak makine öğrenmesinde en yaygın kullanılan nümerik optimizasyon yöntemi gradyan iniş yöntemi tanıtılacaktır.

Regularization (Düzleştirme) Terimi ile Ezberlemenin (Aşırı Öğrenmenin) Önlenmesi:

Makine öğrenmesinde norm kavramı, Regularization (düzleştirme) adı verilen süreçte kullanılır. Kayıp fonksiyonuna eklenen ayaklanabilir(öğrenme) katsayıların (ağırlık katsayıları) büyüklüğünü optimizasyon sürecinde kontrol etmek için tercih edilir. Yaygın olarak iki formu vardır.

L1-norm kullanan L1 regularization ve L2-norm kullanan L2 regularization işlemidir. Yaygın olarak aşırı öğrenme (overfitting) önlemek için kayıp fonksiyonuna eklenir.

L1 regularization (L1 düzleştirme):

Varsayalım $T=\{(u_1,d_1),\ (u_2,d_2),\ ..,\ (u_n,d_n)\}$ eğitim kümesi için $y = w_2u^2 + w_1u + w_0$ polinomsal regresyon problemi çözülmek istetiyor. Burada modelin optimize edilen parametreleri vektör formunda $W = [w_2 \quad w_1 \quad w_0]$.

Bu durumda karesel hata

$E = \frac{1}{2}e_1^{\,2} + \frac{1}{2}e_2^{\,2} + .. + \frac{1}{2}e_n^{\,2} = \frac{1}{2}\sum_{i=1}^{n} e_i^{\,2} = \frac{1}{2}\sum_{i=1}^{n}(d_i - (w_2u^2 + w_1u + w_0))^2$ olarak yazılabilir.

Burada $e_i = d_i - y_i = d_i - (w_2u^2 + w_1u + w_0)$ veri başına hatayı ifade eder.

L1 regularization için karesel hata kayıp fonksiyonu

$$E = \frac{1}{2}\sum_{i=1}^{n}(e_i)^2 + \lambda\|W\|_1$$

$$= \frac{1}{2}\sum_{i=1}^{n}(d_i - y_i)^2 + \lambda\|W\|_1$$

$$= \frac{1}{2}\sum_{i=1}^{n}(d_i - (w_2 u_i^{\ 2} + w_1 u_i + w_0))^2 + \lambda\|[w_2 \quad w_1 \quad w_0]\|_1$$

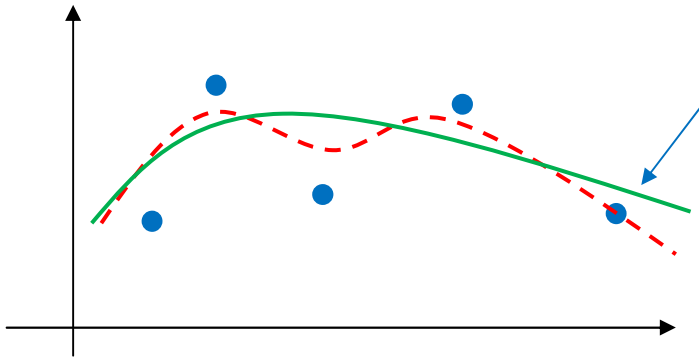
$$= \frac{1}{2}\sum_{i=1}^{n}(d_i - (w_2 u_i^{\ 2} + w_1 u_i + w_0))^2 + \lambda\sum_{i=0}^{2}|w_i|$$

$$= \frac{1}{2}\sum_{i=1}^{n}(d_i - (w_2 u_i^{\ 2} + w_1 u_i + w_0))^2 + \lambda(|w_2| + |w_1| + |w_0|)$$

Karesel hatayı minimize eder. Modelin eğitimi için gereklidir.

Düzleştirme terimini minimize eder. Modelin genelleme sağlayabilmesi için kullanılır. (Ezberlemeyi önler)

olarak yazılabilir. Burada Bu kayıp fonksiyonu minimize edildiği zaman hem karesel hata toplamı ($\frac{1}{2}\sum_{i=1}^{n}(e_i)^2$) hem de L1 regularization terimi ($|w_2| + |w_1| + |w_0|$) birlikte minimize olur. Böylece, karesel hata minimizasyonu ile karesel hatası az olan model elde edilir hem de modelin katsayıları mümkün olduğunca küçük değerler alması sağlanır. Böylece, eğitim sonucunda elde edilen modelin karakteristiğinin daha düz olması sağlanır ve bu aşırı öğrenmeyi (overfitting) önleyebilir. Aşağıdaki şekil düzleştirmenin etkisini göstermek için çizilmiştir. Yeşil kesikli çizgi ile görülen regresyon modeli düzleştirmeye uygulanmayan modeli gösteriyor. Bütün veri noktalarından geçme yani öğrenme eğiliminde. Burada düzleştirme uygulandığı zaman model ayarlanabilir parametrelerin değerleri azalacağı için model daha az değişim gösterebilir. Düzleştirme durumunda yeşil renkli modele dönüşebilir. Böylece modelin genelleme özelliği artırılabilir



Düzleştirme uygulanırsa model yeşil renkle ifade edilen model dönüşebilir ve aşırı öğrenmeye bir çözüm olabilir.

Burada λ düzleştirme katsayısı optimizasyonda ($|w_2| + |w_1| + |w_0|$) teriminin minimizasyonun önem derecesini belirler. λ yüksek değerli ise optimizasyon ($|w_2| + |w_1| + |w_0|$) teriminin minimizasyonuna ($\frac{1}{2}\sum_{i=1}^{n}(e_i)^2$) teriminden daha fazla önem verir. Çünkü büyük λ değeri

$(|w_2|+|w_1|+|w_0|)$ terimi ile çarpılarak E hata değerini çok yükseltir. Optimizasyon algoritması E azaltabilmek için değeri yüksek olan terime daha fazla önem verir. Bu durumda eğer λ uygun belirlenmez (çok yüksek değer verilirse) ise yetersiz öğrenme (under-fitting) problemlerine yol açabilir. Çünkü, optimizasyon $(\frac{1}{2}\sum_{i=1}^{n}(e_i)^2)$ teriminin miniminizasyonuna daha az önem verir. Buda modelin verileri temsil performansını azaltabilir.

L2 regularization (L2 düzleştirme):

L2 regularization için karesel hata kayıp fonksiyonu

$$E=\frac{1}{2}\sum_{i=1}^{n}(e_i)^2+\lambda\|W\|_2$$

$$=\frac{1}{2}\sum_{i=1}^{n}(d_i-y_i)^2+\lambda\|W\|_2$$

$$=\frac{1}{2}\sum_{i=1}^{n}(d_i-(w_2{u_i}^2+w_1u_i+w_0))^2+\lambda\|[w_2 \quad w_1 \quad w_0]\|_2$$

$$=\frac{1}{2}\sum_{i=1}^{n}(d_i-(w_2{u_i}^2+w_1u_i+w_0))^2+\lambda\sqrt{\sum_{i=0}^{2}{w_i}^2}$$

$$=\frac{1}{2}\sum_{i=1}^{n}(d_i-(w_2{u_i}^2+w_1u_i+w_0))^2+\lambda\sqrt{{w_2}^2+{w_1}^2+{w_0}^2}$$

Karesel hatayı minimize eder. Modelin eğitimi için gereklidir.

Düzleştirme terimini minimize eder. Modelin genelleme sağlayabilmesi için kullanılır. (Ezberlemeyi önler)

olarak yazılabilir. Burada Bu kayıp fonksiyonu minimize edildiği zaman hem karesel hata toplamı $(\frac{1}{2}\sum_{i=1}^{n}(e_i)^2)$ hem de L2 regulazation terimi $\sqrt{{w_2}^2+{w_1}^2+{w_0}^2}$ birlikte minimize olur. Böylece, veriye uyum sağlanırken model katsayılarının mümkün olduğunca küçük değerlere ayarlanması sağlanır. Böylece, modelin karakteristiğinin daha düz olması sağlanır ve bu aşırı öğrenmeyi (overfiti) önleyebilir. Burada λ katsayısı optimizasyonda L2 regularization terimi $\sqrt{{w_2}^2+{w_1}^2+{w_0}^2}$ teriminin önem derecesini belirler. Eğer λ uygun belirlenmez ve çok yüksek seçilirse ise yetersiz öğrenme (under-fitting) problemlerine yol açabilir.

L1 regularization (Lasso Regression'da kullanılır) ile L2 regularization (Ridge regression'da kullanılır) etkileri arasında fark olduğu belirtilir.

* Yapılan deneysel çalışmalarda, L1 regularization önemsiz ayarlanabilir parametreleri (w_i) sıfıra yaklaştırma eğiliminde olduğu gözlemlenmiştir. Diğer bir ifade ile eğer w_i parametresi sıfıra yakın değer alırsa modelde etkisi (önemi) azalır. Bu durumda L1 regularization bu parametreyi sıfırlama eğiliminde olduğu kabul edilebilir ve modeli küçültmek (model karmaşıklığını azaltmak) için tercih edilir. Örneğin $y = w_2u^2 + w_1u + w_0$ modelde $w_2 \to 0$ giderse aslında model lineer bir model yaklaşır. $y = w_2u^2 + w_1u + w_0 \to w_1u + w_0$ bu model karmaşıklığını düşürebilir.

* L2 regularization ise karesel terimlerden dolayı ($\sqrt{{w_2}^2 + {w_1}^2 + {w_0}^2}$) çok yüksek w_i değerlerini öncelikle azaltma eğiliminde olacaktır. Genelde ağırlıkları 1'den küçük değerlerine getirme eğilimi gösterir.

Örnek: T={ (1,2) , (2,5) } eğitim kümesi için $y = w_2u^2 + w_1u + w_0$ modeli için

a) $w_2 = 1, w_1 = 2, w_0 = 5$ katsayıları için düzleştirme katsayısı $\lambda = 0.5$ için L2-düzleştirme uygulayan hata fonksiyonu değerini hesaplayınız.

$$E = \frac{1}{2}\sum_{i=1}^{n}(e_i)^2 + \lambda\|W\|_2 = \frac{1}{2}\sum_{i=1}^{n}(d_i - y_i)^2 + \lambda\|W\|_2 \quad =>$$

$$E = \frac{1}{2}\sum_{i=1}^{n}(d_i - (w_2{u_i}^2 + w_1u_i + w_0))^2 + \lambda\sqrt{{w_2}^2 + {w_1}^2 + {w_0}^2} \quad =>$$

$$E = \frac{1}{2}((d_1 - (w_2{u_1}^2 + w_1u_1 + w_0))^2 + (d_2 - (w_2{u_2}^2 + w_1u_2 + w_0))^2) + \lambda\sqrt{{w_2}^2 + {w_1}^2 + {w_0}^2}$$

$$E = \frac{1}{2}((2 - (1.1^2 + 2.1 + 5))^2 + (5 - (1.2^2 + 2.2 + 5))^2) + 0.5\sqrt{1^2 + 3^2 + 5^2}$$

$$E = \frac{1}{2}((2 - (1.1^2 + 2.1 + 5))^2 + (5 - (1.2^2 + 2.2 + 5))^2) + 0.5\sqrt{1^2 + 2^2 + 5^2} = 52.73$$

b) $w_2 = 1, w_1 = 1, w_0 = 2$ katsayıları için düzleştirme katsayısı $\lambda = 0.5$ için L2-düzleştirme uygulayan hata fonksiyonu değerini hesaplayınız. (a) şıkkındaki sonuç ile kıyaslayınız. Optimizasyon sürecinde minimizasyon gerçekleşmesi için hangi şıktaki ağırlıklar tercih edilir?

$$E = \frac{1}{2}((2 - (1.1^2 + 1.1 + 2))^2 + (5 - (1.2^2 + 1.2 + 2))^2) + 0.5\sqrt{1^2 + 1^2 + 2^2} = 15.72$$

Daha küçük hata değeri verdiği için optimizasyon sürecinin $w_2 = 1, w_1 = 1, w_0 = 2$ katsayılarını seçmesi beklenebilir.