

Büyük Veriler İçin Sıralı Öğrenme

Doç. Dr. Süleyman Serdar Kozat

Elektrik ve Elektronik Bölümü

Bilkent Üniversitesi

kozat@ee.bilkent.edu.tr

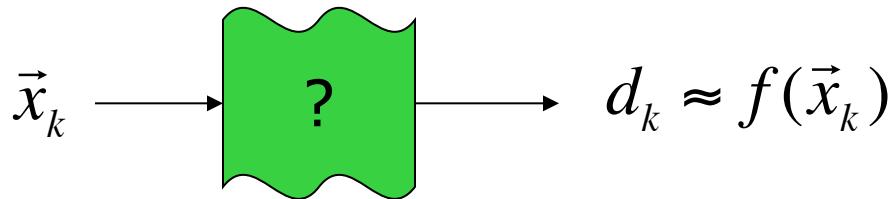
<http://www.ee.bilkent.edu.tr/~kozat>



ÖZET

- Öğrenmeye giriş
- Çalıştığımız çerçevenin modellediği uygulamalar
- Öğrenme türleri
- Sıralı (Online) öğrenme
- En çok kullanılan sıralı öğrenme teknikleri
- Uzmanların karışıntıları yaklaşımları
- Tek kollu haydut yaklaşımları
- Büyük verilerden sıralı öğrenmenin teorik alt yapısı
- Pişmanlığa dayalı öğrenme
- Sıralı portföy yönetimi

Öğrenme nedir?



k: zaman, verinin indeksi, örnek indeksi...

x_k : sistemin girdisi, input vektörü, öznitelik verktörü, girdi sinyali...

d_k : istenen veri, etiket, ulaşılacak sistem çıktısı

$$d_k \approx f(\vec{x}_k)$$

İstenen çıktı veya etiket ile elimizdeki girdi vektörü arasında ‘varsayılan’ ilişkiyi öğrenme bizim bugünkü problemimiz. Ancak:

- Böyle bir ilişki varsa bu nasıl öğrenilir?
- Öğrenme ne demek? İyi öğrendim ne zaman denilebilir?
- Öğrendiğinize nasıl inanabilirim?
- Çünkü çalışıyor ne demek?
- Öğrenmeniz dış etkenlere karşı dayanıklı mı?

Gerçek hayattan örnekler

■ Sınıflandırma

- **Spam süzgeçleme:**

email subject: "Learn 3 languages just in 7 days!" Spam mı değil mi?

k^{th} email $\longrightarrow \vec{x}_k = \text{Kelimelerden öznitelik vektörü} \longrightarrow d_k \in \{0,1\}$
Spam ya da değil

- **Yüz tanıma:**

k^{th} vesikalık $\longrightarrow \vec{x}_k = \text{Yüz yapısından öznitelik vektörü} \longrightarrow d_k \in \{0,1\}$
Kadın ya da erkek

- **İnternette kişiye özel reklam seçimi**

k^{th} kullanıcı $\longrightarrow \vec{x}_k = \text{Kullanım verilerinden, kişisel bilgilerden çıkarılan öznitelik vektörü} \longrightarrow d_k \in \{1,2,\dots,m\}$
m tane reklamdan kullanıcıya en uygunu

Örnekler

- **Düzensizlik takibi:**

k^{th} log dosyası $\longrightarrow \vec{x}_k =$ Portlara
bağlılarından
öznitelik vektörü $\longrightarrow d_k \in \{0,1\}$
Düzensizlik var/yok

■ Regresyon

- **Kestirim/Tahmin:**

$$\{x(1), x(2), \dots, x(t), \dots\}$$

IBM hissesinin borsadaki fiyatı,
t: bugün

$$x(t+1)$$

Yarınki, t+1, fiyat nedir?

t gününe kadar
geçmiş zamanındaki
fiyatlar

$$\vec{x}_k = \begin{bmatrix} x(t) \\ \vdots \\ x(t-d) \end{bmatrix} \longrightarrow d_k = x(t+1)$$
$$d_k \in R$$

Örnekler

- **Portfoy seçimi:**

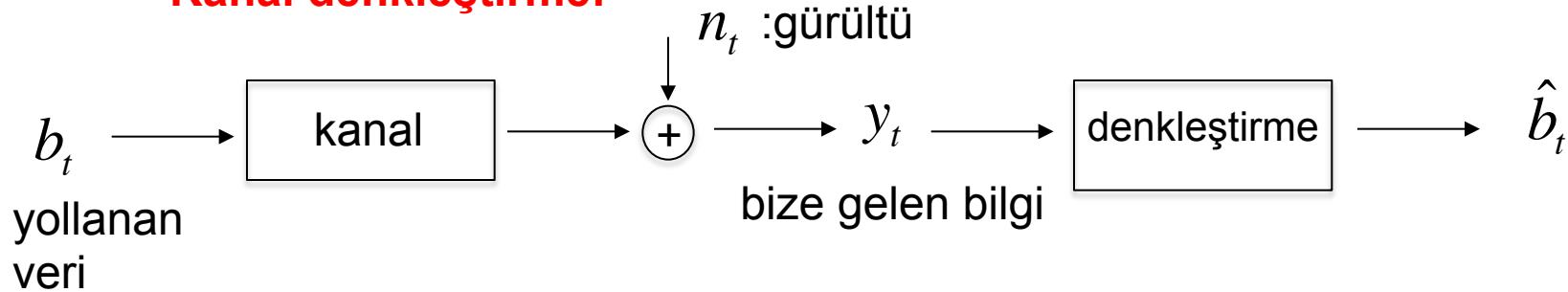
Paramı bugün hangi hisseye yatırayım: IBM, Microsoft, Apple?

1 Türk Lira paramın: %20'u IBM, %30'si Microsoft, %50'si Google

Portföy vektörü: $\vec{b}_t = \begin{bmatrix} 0.2 & 0.3 & 0.5 \end{bmatrix}$

t gününe kadar şirketler hakkındaki bilgiler $\vec{x}_k = \begin{bmatrix} \text{Öznitelik vektörü} \end{bmatrix}$ $d_k = \vec{b}_t$
 $d_k \in Simplex$

- **Kanal denkleştirme:**



$$\{y_t\} \rightarrow \vec{x}_k = \begin{bmatrix} y(t+d) \\ \vdots \\ y(t-d) \end{bmatrix} \rightarrow d_k = b_t$$

bize gelen bilgi

‘Öğrenme’ Türleri

▪ Etiketli Öğrenme (Supervised Learning)

Hem gözlem vektörleri hem de bunlara karşılık gelen etiketler verilir. $\{(\vec{x}_k, d_k)\}$

- En tercih edilen öğrenme yöntemidir.
- Gerçek hayatı etiketleme yapmak son derece masraflı ve zaman gerektiren bir ugrastır.
- Etikelemenin son derece hatasız olması önemlidir.

▪ Etiketsiz Öğrenme (Unsupervised Learning)

 $\{(\vec{x}_k, .)\}$

Gözlem vektörlerine karşılık gelen istenen d_k 'ler verilmez.

- Özellikle son yıllarda büyük miktarda verinin toplanabileceği uygulamalar sonucunda önemi artmıştır.
- Genellikle topaklama ismi ile de bilinir, e.g., PCA, SVD, etc.
- Yeni bir araştırma yönü: Derin öğrenmede autoencoders, etc.

‘Öğrenme’ Türleri

■ **Yarı Etiketli Öğrenme (Semi Supervised Learning)**

Bazı gözlem vektörleri ve bunlara karşılık gelen etiketler verilir.

- Bir kısım etiketin verildiği durumlardaki öğrenmedir. Genelde ses tanıma uygulamalarında genişçe kullanılır.

■ **Aktif Öğrenme (Active Learning)**

Bazı gözlem vektörlerine karşılık gelen etiketlerin d_k 'lerin aktif istediği durumdur.

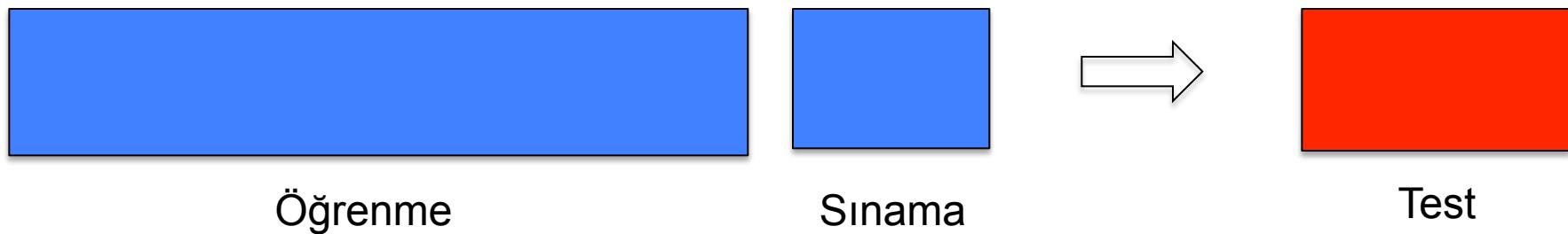
- Büyük miktarlardaki etiketsiz veri tabanlarında bazı kriterlere göre etiketlerin istenmesi durumudur.
- Özellikle büyük veri uygulamalarında önem kazanmaktadır.
- Yarı etiketli öğrenmeden farkı etiketlerin belli bir kriter'e göre sadece belirlenen öznitelik vektörleri için istenmesi durumudur.
- Son derece popüler bir yaklaşımındır.

‘Öğrenme’ Türleri

■ Toplu Öğrenme (Batch Learning)

Öznitelik vektörleri ile istenen veriler toplu olarak öğrenme algoritmasına verilir:

$$\{(\vec{x}_1, d_1), (\vec{x}_2, d_2), \dots, (\vec{x}_N, d_N)\}$$



- En çok kullanılan, üzerinde en çok çalışılmış geleneksel öğrenme çerçevesidir.
- Toplanan veriden sistem öğrendikten sonra gerçek hayatı doğrudan uygulanır.
- Genellikle sistem parametreleri uygulamaya geçildikten sonra değiştirilmez.
- VC dimension, SVM, Deep Learning, Gradient Descent etc.
- Bir önceki tüm durumlar için geçerlidir: etiketli ve etiketsiz öğrenme, yarı ve aktif etiketli öğrenme gibi.

‘Öğrenme’ Türleri

▪ Sıralı Öğrenme (Online Learning)

Öznitelik vektörleri ve (varsayımsa) etiketler kademeli olarak gelir/işlenir

$$(\vec{x}_1, d_1) \rightarrow \text{öğren} \rightarrow (\vec{x}_2, d_2) \rightarrow \text{öğren} \rightarrow (\vec{x}_3, d_3) \rightarrow \text{öğren}$$



Büyük Veri	Sıralı öğrenme
Yüksek hızla gelen veri	Sadece yeni gelen veri için güncelleme
Değişkenlik (kalite, istatistiksel yapı, amaç gibi)	Sürekli yeni gelen veriye uyum
Yüksek hacim	Toplu öğrenmenin aksine herhangi bir depolamaya ihtiyaç duymadan anlık işleme

‘Öğrenme’ Türleri

■ İstatiksel çerçeve

- Genellikle öznitelik vektörleri ve etiketlerin bilinmeyen bir dağılımdan i.i.d. olarak elde edildiği varsayıılır:

$$(\vec{x}_t, d_t) \in P(\vec{x}_t, d_t)$$

$$P(\vec{x}_t, d_t): \text{bilinmiyor}$$

- VC dimensions, MSE kestirim.

■ Belirlenimci çerçeve

- Öznitelik vektörleri ve etiketleri üzerinde herhangi istatiksel varsayıım yok.
- Kaotik sinyaller.
- İstatistiksel olarak modellenemeyen yüksek derecede düzensizlik içeren veriler.
- Gerçek hayat verileri (?)
- Bu kapsamda aklımızda tutalım: Performans nasıl tanımlanacak?

‘Öğrenme’ Türleri

▪ Modele dayalı ya da parametrik öğrenme

- İstenen veri ile öznitelik vektörleri arasındaki ilişkinin belli parametreleri olan bir modelden geldiği varsayılar:

$$d_k \approx f(\vec{x}_k, \vec{\theta})$$

- İlişki kurma problemi bu sistem parametrelerini öğrenme olarak kabul edilir.
- En kullanılan parametrik modeller:
 - Linear modellerde sistem parametreleri doğrusal katsayılardır: $f(\vec{x}_k, \vec{\theta}) = \vec{\theta}^T \vec{x}_k$
 - Sinir ağlarında sistem parametreleri, doğrusal katsayılar, sinir düğümleri sayısı, doğrusal olmayan eşikleme fonksiyonlarıdır: $f(\vec{x}_k, \vec{\theta})$
 - Aynı şekilde CNN ve RNN'lerde kademe sayısı, kurulum düzeni gibi.
 - GMM'lerde Gaus dağılımlarının ortama vektörleri ve korelasyon matrislerinin tümüdür:

$$f(\vec{x}_k, \vec{\theta}) = \sum_{k=1}^m w_k N(\vec{\mu}_k, \Sigma_k, \vec{x}_k) \quad \vec{\theta} = \{w_k, \vec{\mu}_k, \Sigma_k\}_{k=1}^m$$

‘Öğrenme’ Türleri

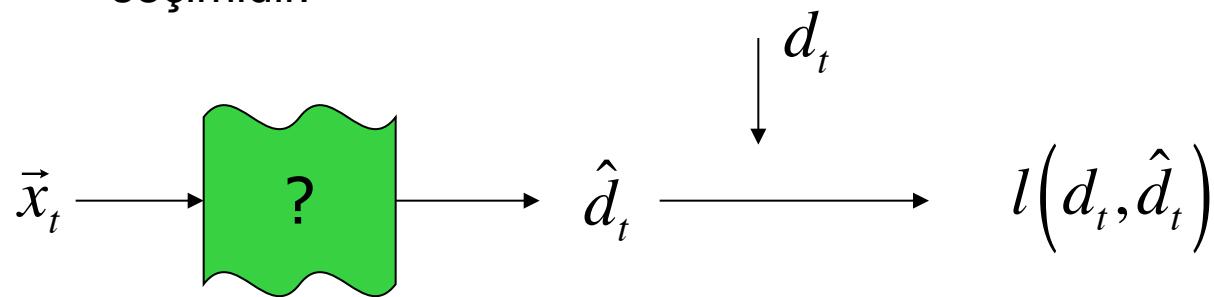
▪ **Modelsiz ya da parametrik olmayan öğrenme**

- İstenen veri ile öznitelik vektörleri arasındaki ilişkinin doğrudan veriden herhangi bir modelleme (?) yapılmadan öğrenildiği durumdur.
- Gerçek hayat uygulamalarında modellemenin sağlıklı yapılamayacağının düşünüldüğü, kaotik sistemler gibi, kullanılır.
- Parzen penceresine dayalı istatistiksel dağılım öğrenme yaklaşımları
- Genelde iyi çalışması için yüksek miktarlarda veriye ihtiyaç vardır.

Gerekli araçlar

- **Hata fonksiyonu:**

- Öğrenme konusunda en önemli konulardan biri hata fonksiyonunun seçimidir.



- Karesel hata: Pratikte en çok kullanılan hata çünkü $l(d_t, \hat{d}_t) = (d_t - \hat{d}_t)^2$ kullanımı kolay.
- Mutlak hata: $l(d_t, \hat{d}_t) = |d_t - \hat{d}_t|$
- Log hata: $l(d_t, \hat{d}_t) = (1 - \hat{d}_t) \ln \frac{1 - \hat{d}_t}{1 - d_t} + \hat{d}_t \ln \frac{\hat{d}_t}{d_t}$
- Sınıflandırma hatası: $l(d_t, \hat{d}_t) = 1_{d_t \neq \hat{d}_t}$

Etiketli Öğrenme İle Karesel Hata Altında Doğrusal Regresyon

1) $(\vec{x}_t, d_t) \in P(\vec{x}_t, d_t)$

2) Karesel hata altında yapabileceğim en iyi kestirim

$$f^* = E[d_t | \vec{x}_t]$$

3) Modele dayalı öğrenme yapalım, doğrusal öğrenme: $\hat{d}_t = \vec{w}^T \vec{x}_t$

4) En iyi doğrusal model ($P(.,.)$ bilinmediği için hesaplanamaz).

$$\vec{w}^* = \underset{\vec{w} \in F}{\operatorname{argmin}} E[(d_t - \vec{w}^T \vec{x}_t)^2] = R^{-1} \vec{p} \quad R = E[\vec{x}_t \vec{x}_t^T] \quad \vec{p} = E[\vec{x}_t d_t]$$

5) Gözlem yapıp veri topladım:

$$\{(\vec{x}_1, d_1), (\vec{x}_2, d_2), \dots, (\vec{x}_N, d_N)\}$$

6) Gözlemlerime uyan en iyi doğrusal model:

$$\vec{w}_N = \underset{\vec{w} \in F}{\operatorname{argmin}} \sum_{t=1}^N (d_t - \vec{w}^T \vec{x}_t)^2 = \left(\sum_{t=1}^N \vec{x}_t \vec{x}_t^T \right)^{-1} \left(\sum_{t=1}^N \vec{x}_t d_t \right)$$

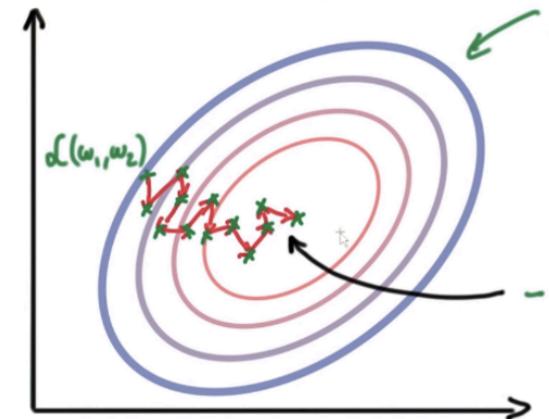
Stokastik Meyil Yönünde İniş Algoritması

1) Hata fonksiyonu: $J(\vec{w}) = E[(d_t - \vec{w}^T \vec{x}_t)^2]$

2) Aşamalı minimale gidiş:

$$\vec{w}_{k+1} = \vec{w}_k - \mu \nabla J(\vec{w}) = \vec{w}_k + \mu E[(d_t - \vec{w}^T \vec{x}_t) \vec{x}_t]$$

öğrenme hızı ya da atılan adımın boyut parametresi



3) Ama istatistiksel dağılım bilinmiyor: $E[.]$ bilgisi ve meyil bilgisi yok.

4) O zaman gerçek meyil yerine ortalama meyili verecek kestirim kullanalım:

$(d_t - \vec{w}^T \vec{x}_t) \vec{x}_t$ gerçek meyilin $\nabla J(\vec{w})$ tutarlı bir kestirimidir.

5) SGD algoritması: $\tilde{w}_{t+1} = \tilde{w}_t + \mu e_t \vec{x}_t \quad e_t = d_t - \tilde{w}^T \vec{x}_t$

6) N iterasyon sonucunda, for $t=1, \dots, N \rightarrow \tilde{w}_N$

7) \tilde{w}_N mi seçilmeli \vec{w}_N mi?

Örnek Sıralı Algoritmalar

- 1) Daha iyi meyil kestirimi, bir pencerede eğim kestirimi, düzensizliği daha az bir kestirim, :

$$\tilde{w}_{k+1} = \tilde{w}_k + \mu \sum_{t=t_k}^{t_{k+1}} (d_t - \tilde{w}_k^T \vec{x}_t) \vec{x}_t / (t_{k+1} - t_k + 1)$$

- 2) Eğim bilgisini de kullanalım (ikinci derece sıralı algoritmalar):

$$\vec{w}_{k+1} = \vec{w}_k - \mu H(\vec{w}_k) \nabla J(\vec{w}_k) \quad H(\vec{w}_k) : \text{Hessian}$$

Stokastik:

$$\vec{w}_{t+1} = \vec{w}_t + \mu R_t^{-1} e_t \vec{x}_t \quad R_t = \sum_{r=1}^{t-1} \vec{x}_r \vec{x}_r^T$$

Tüm iterasyonlar yapılmınca $t=1, \dots, N$ $\Rightarrow \vec{w}_N$

Örnek Sıralı Algoritmalar

Ancak sıralı bilgiyi kullanırsanız:

$$R_t = R_{t-1} + \vec{x}_t \vec{x}_t^T$$

ortaya sinyal işleme literatürde en çok bilinen ve kullanılan RLS algoritması ya da Kalman algoritması çıkar:

$$\vec{w}_{t+1} = \vec{w}_t + e_t \vec{g}_t$$

$$\vec{g}_t = P_{t-1} \vec{x}_t \left\{ \mu^{-1} + \vec{x}_t^T P_{t-1} \vec{x}_t \right\}^{-1}$$

$$P_t = \mu^{-1} P_{t-1} - \vec{g}_t \vec{g}_t^T \mu P_{t-1}$$

- RLS algoritması istatistiksel koşullarda özdeğer dağılımını düzelttiği için SGD yaklaşımılarına göre daha hızlı yakınsama yapar.
- RLS algoritmasının Online Learning literatüründeki ismi Online Newton Step algoritmasıdır

Örnek Sıralı Algoritmalar

3) Bregman Farkı'na (Divergence) dayalı sıralı algoritma çalışma yaklaşımları:

- Diyelim ki elimize şu ana kadar gelen belli sayıda veriden kestirim yapmış olalım:

$$\{(\vec{x}_1, d_1), (\vec{x}_2, d_2), \dots, (\vec{x}_t, d_t)\} \rightarrow \vec{w}_t$$

- Yeni veri elimize gelsin:

$$(\vec{x}_{t+1}, d_{t+1})$$

- Yeni gelen bilgi ile \vec{w}_t 'yi nasıl değiştirelim?

$$\vec{w}_{t+1} = \arg \min_{\vec{w}} \left\{ D(\vec{w}, \vec{w}_t) + \mu l(d_{t+1}, \vec{w}^T \vec{x}_{t+1}) \right\}$$

Yeni seçilen sistem parametreleri eski sistem parametrelerinden çok farklı olmamalı ancak yeni gelen verileri de modelleyebilmeli.

Örnek Sıralı Algoritmalar

$$\vec{w}_{t+1} = \operatorname{argmin}_{\vec{w}} \left\{ D(\vec{w}, \vec{w}_t) + \mu l(d_{t+1}, \vec{w}^T \vec{x}_{t+1}) \right\}$$

$D(\vec{w}, \vec{w}_t)$: Eski ve yeni arasındaki farkın cezası. Bregman Farkı

$l(d_{t+1}, \vec{w}^T \vec{x}_{t+1})$: Yeni gelen veriye uyum

μ : Uzlaşma parametresi

Bir sonraki parametreyi bulmak için $D(\vec{w}, \vec{w}_t)$ ve $l(d_{t+1}, \vec{w}^T \vec{x}_{t+1})$ içbükey fonksiyonlar seçilir ise:

$$\nabla_{\vec{w}} \left\{ D(\vec{w}, \vec{w}_t) + \mu l(d_{t+1}, \vec{w}^T \vec{x}_{t+1}) \right\} = \vec{0}$$

Örnek Sıralı Algoritmalar

a) Bregman Farkı'nı Euclid'e dayalı fark, hatayı da karesel alırsak:

$$\left. \begin{array}{l} D(\vec{w}, \vec{w}_t) = \|\vec{w} - \vec{w}_t\|_I^2 \\ l(d_{t+1}, \vec{w}^T \vec{x}_{t+1}) = (d_{t+1} - \vec{w}^T \vec{x}_{t+1})^2 \end{array} \right\} \quad \vec{w}_{t+1} = \vec{w}_t + \mu \frac{\vec{e}_t \vec{x}_t}{1 + \vec{x}^T \vec{x}_t}$$

Normalized SGD ya da
NLMS algoritması

b) Bregman Farkı'nı ağırlıklı Euclid'e dayalı fark, hatayı da karesel alırsak:

$$\left. \begin{array}{l} D(\vec{w}, \vec{w}_t) = \|\vec{w} - \vec{w}_t\|_{I - \vec{x}_t \vec{x}_t^T}^2 \\ l(d_{t+1}, \vec{w}^T \vec{x}_{t+1}) = (d_{t+1} - \vec{w}^T \vec{x}_{t+1})^2 \end{array} \right\} \quad \vec{w}_{t+1} = \vec{w}_t + \mu \vec{e}_t \vec{x}_t$$

SGD ya da LMS algoritması

Örnek Sıralı Algoritmalar

c) Bregman Farkı'nı ağırlıklı Euclid'e dayalı fark, hatayı da 4. kuvvet alırsak:

$$\left. \begin{aligned} D(\vec{w}, \vec{w}_t) &= \|\vec{w} - \vec{w}_t\|_{I - \vec{x}_t \vec{x}_t^T}^2 \\ l(d_{t+1}, \vec{w}^T \vec{x}_{t+1}) &= (d_{t+1} - \vec{w}^T \vec{x}_{t+1})^4 \end{aligned} \right\} \quad \begin{aligned} \vec{w}_{t+1} &= \vec{w}_t + \mu e_t^3 \vec{x}_t \\ &\text{LMF algoritması} \end{aligned}$$

d) Bregman Farkı'nı ağırlıklı Euclid'e dayalı fark, hatayı da karesel alırsak:

$$\left. \begin{aligned} D(\vec{w}, \vec{w}_t) &= \|\vec{w} - \vec{w}_t\|_{R_t}^2 \\ l(d_{t+1}, \vec{w}^T \vec{x}_{t+1}) &= (d_{t+1} - \vec{w}^T \vec{x}_{t+1})^2 \\ R_t &= \sum_{k=1}^t \vec{x}_k \vec{x}_k^T \end{aligned} \right\} \quad \begin{aligned} \vec{w}_{t+1} &= \vec{w}_t + \mu R_t^{-1} e_t \vec{x}_t \\ &\text{Online Newton Step ya da} \\ &\text{RLS algoritması} \end{aligned}$$

Örnek Sıralı Algoritmalar

e) Bregman Farkı'nı ağırlıklı KL divergence, hatayı da karesel alırsak:

$$\left. \begin{aligned} D(\vec{w}, \vec{w}_t) &= \sum_{i=1}^d w_i \ln \left(\frac{w_i}{w_{i,t}} \right) \\ l(d_{t+1}, \vec{w}^T \vec{x}_{t+1}) &= (d_{t+1} - \vec{w}^T \vec{x}_{t+1})^2 \end{aligned} \right\} \begin{aligned} w_{i,t+1} &= w_{i,t} \exp(-e_t x_{i,t}) \\ &\text{Multiplicative Update} \\ &\text{algoritması} \end{aligned}$$

f) Bregman Farkı'nı ağırlıklı KL divergence, hatayı da karesel alırsak:

$$\left. \begin{aligned} D(\vec{w}, \vec{w}_t) &= \sum_{i=1}^d w_i \ln \left(\frac{w_i}{w_{i,t}} \right) \\ l(d_{t+1}, \vec{w}^T \vec{x}_{t+1}) &= (d_{t+1} - \vec{w}^T \vec{x}_{t+1})^2 \end{aligned} \right\} \begin{aligned} w_{i,t+1} &= w_{i,t} \frac{\exp(-e_t x_{i,t})}{\sum_{i=1}^d \exp(-e_t x_{i,t})} \\ \vec{w}^T \vec{w} &= 1 \\ &\text{Normalized Multiplicative Update} \\ &\text{algoritması} \end{aligned}$$

Örnek Sıralı Algoritmalar

Benim uzun süredir ispat etmeye çalıştığım bir öngörü: Tüm sıralı (online algoritmalar)

$$\vec{w}_{t+1} = \operatorname{argmin}_{\vec{w}} \left\{ D(\vec{w}, \vec{w}_t) + \mu l(d_{t+1}, \vec{w}^T \vec{x}_{t+1}) \right\}$$

ya da

$$\vec{w}_{t+1} = \operatorname{argmin}_{\vec{w}} \left\{ D(\vec{w}, \vec{w}_t) \right\}$$

s.t. $l(d_t, \vec{w}^T \vec{x}_t)$ satisfies some constraint

şeklinde yazılabilir. Bu sayede algoritma seçimi hata fonksiyonu seçimine ya da divergence seçimine, algoritma karşılaştırma da bu iki fonksiyonun farklarına göre yapılabilir.

Örnek Sıralı Algoritmalar

$$\vec{w}_{t+1} = \operatorname{argmin}_{\vec{w}} \left\{ \|\vec{w} - \vec{w}_t\|^2 \right\}$$

$$\text{s.t. } |d_t - \vec{w}^T \vec{x}_t| \leq \lambda$$

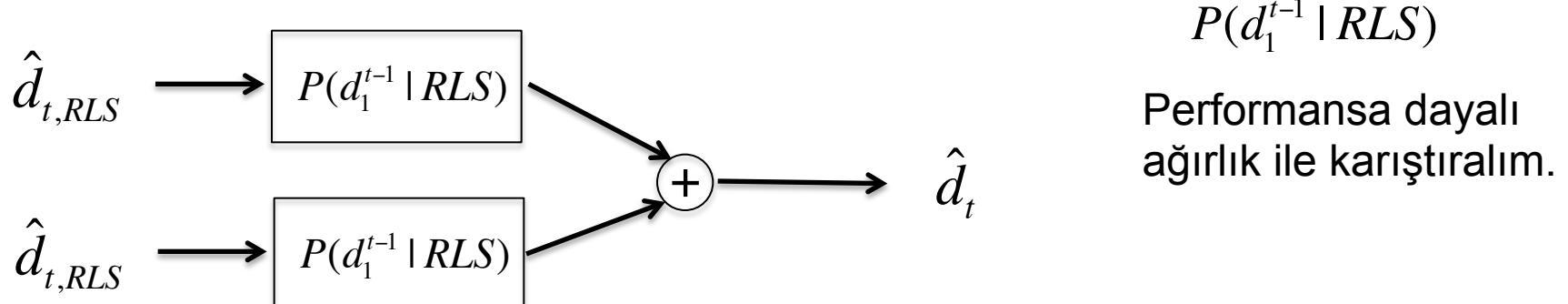
- Küme üyesi (set membership) sıralı öğrenme algoritmasını veriyor.
- Bu algoritmaya da Online Learning literatüründe Passive Aggressive algoritması deniyor.

Uzmanların karışımı yaklaşımı yaklaşımları

- Birçok sıralı (online) algoritma var, bunlardan hangisi tercih edilmeli?

$$\hat{d}_{t,RLS} = \vec{w}_{t,RLS}^T \vec{x}_t \quad \Rightarrow \quad \left\{ (\vec{x}_1, d_1), (\vec{x}_2, d_2), \dots \right\} \quad \left[\begin{array}{l} \sum_{t=1}^N (d_t - \hat{d}_{t,RLS})^2 \\ \sum_{t=1}^N (d_t - \hat{d}_{t,LMS})^2 \end{array} \right]$$
$$\hat{d}_{t,LMS} = \vec{w}_{t,LMS}^T \vec{x}_t \quad \Rightarrow \quad$$

- Geleceği bilmeden hangi algoritmanın en iyi olduğu bilinemez.
- O zaman ne yapmalı?
- Uzmanların karışımı yaklaşımı yaklaşımları



Uzmanların karışımı yaklaşımı yaklaşımları

- Tercih yapmadan karıştıralım:

m tane uzman: $\{\hat{d}_{t,1}, \dots, \hat{d}_{t,m}\}$

$$\hat{d}_t = \sum_{k=1}^m \alpha_{t,k} \hat{d}_{t,k}$$

$$\alpha_{t,k} = \frac{\sum_{j=1}^{t-1} \exp(-\mu l'(d_j, \hat{d}_{j,k}) \hat{d}_{j,k})}{\sum_{r=1}^m \sum_{j=1}^{t-1} \exp(-\mu l'(d_j, \hat{d}_{j,m}) \hat{d}_{j,m})}$$

- Ortaya çıkan uzmanların karışımı algoritması aynı veriye uygulanırsa:

$$\hat{d}_t \quad \Rightarrow \quad \{(\vec{x}_1, d_1), (\vec{x}_2, d_2), \dots\}$$

- Performansı herhangi bir varsayımsız en iyi algoritma kadar iyidir:

$$\frac{\sum_{t=1}^N l(d_t, \hat{d}_t) - \min_{k \in \{1, \dots, m\}} \sum_{t=1}^N l(d_t, \hat{d}_{t,m})}{N} \leq O\left(\frac{\ln m}{N}\right)$$

Uzmanların karışımı yaklaşımı yaklaşımları

- Tercih yapmadan karıştıralım:

m tane uzman: $\{\hat{d}_{t,1}, \dots, \hat{d}_{t,m}\}$

$$\hat{d}_t = \hat{d}_{t,k} \text{ w.p. } \alpha_{t,k}$$

$$\alpha_{t,k} = \frac{\sum_{j=1}^{t-1} \exp(-\mu l(d_j, \hat{d}_{j,k}))}{\sum_{r=1}^m \sum_{j=1}^{t-1} \exp(-\mu l(d_j, \hat{d}_{j,m}))}$$

- Ortaya çıkan uzmanların karışımı algoritması aynı veriye uygulanırsa:

$$\hat{d}_t \quad \Rightarrow \quad \{(\vec{x}_1, d_1), (\vec{x}_2, d_2), \dots\}$$

- Performansı herhangi bir varsayımsız en iyi algoritma kadar iyidir:

$$\frac{E[\sum_{t=1}^N l(d_t, \hat{d}_t)] - \min_{k \in \{1, \dots, m\}} \sum_{t=1}^N l(d_t, \hat{d}_{t,m})}{N} \leq O\left(\frac{\ln m}{N}\right)$$

Tek kollu haydut algoritmaları

- Gerçek hayat uygulamalarında birden fazla algoritma olsa bile hepsinin performansı bazı uygulamalarda aynı anda ölçülemez.
- Örnek olarak online reklam (Facebook) seçimini ele alalım.
- Her kullanıcı için 'm' tane reklamdan biri seçilecek ya da reklam seçim algoritmasından biri kullanılacak:

reklam kümesi: $\{\hat{d}_{t,1}, \dots, \hat{d}_{t,m}\}$

- Sadece bir reklam ya da algoritma seçiliip kullanıcıya gösterilecek, mesela k^{th} reklam seçilmiş olsun.
- Kullanıcı reklamı beğenip tıklarsa kayıp yok tıklamazsa kayıp var:

$$l(d_t, \hat{d}_t) = 1_{d_t \neq \hat{d}_t} \quad \begin{aligned} d_t &: \text{kullanıcının o an seveceği reklam} \\ \hat{d}_t &: \text{bizim kullanıcı için seçtiğimiz reklam} \end{aligned}$$

- Ancak k^{th} reklam ya da algoritma dışında diğerlerinin beğenip beğenilmeyeceği yani kayıp fonksiyonu belli değil. O zaman en iyi reklam nasıl seçilecek?

Tek kollu haydut algoritmaları

- Exp3 algoritması: $\gamma \in [0,1]$ $\alpha_{1,k} = 1/m$

1) Her reklam ya da algoritma için olasılık atanır: $p_{t,k} = (1 - \gamma) \frac{\alpha_{t,k}}{\sum_{i=1}^m \alpha_{t,i}} + \frac{\gamma}{m}$

2) Atanan olasılık ile i_t algoritması seçilir.

3) Seçilen reklam/algoritma için kayıp gözlemlenir: $l(i_t)$

4) Bu kayıp seçme olasılığı ile düzelttilir/orantılanır: $\tilde{l}(i_t) = \frac{l(i_t)}{p_{t,i_t}}$

5) Seçilen reklam/algoritma için ağırlık hesaplanır: $\alpha_{t+1,i_t} = \alpha_{t,i_t} \exp(-\gamma \tilde{l}(i_t) / m)$

6) Tüm diğer reklam/algoritmalar için: $\alpha_{t+1,k} = \alpha_{t,k}$

Tek kollu haydut algoritmaları

Exp3 algoritmasını herhangi bir kişiye uygularsak:

$$Exp3 \implies \{(\vec{x}_1, d_1), (\vec{x}_2, d_2), \dots\} \quad \vec{x}_t = [\hat{d}_{t,1}, \dots, \hat{d}_{t,m}]^T$$

tüm tercih ve reklam dizileri için, varsayımsız olarak,

$$E \left[\sum_{t=1}^N l(d_t, \hat{d}_{t,Exp3}) \right] - \min_k \sum_{t=1}^N l(d_t, \hat{d}_{t,k}) \leq (1-e)\gamma H_{\min} + \frac{m \ln m}{\gamma}$$

$$H_{\min} = \min_k \sum_{t=1}^N l(d_t, \hat{d}_{t,k})$$

performans sınırına ulaşılır.

Yani en iyi reklam seçimi geçmişe bakılarak öğrenilebilir.

Öğrenmenin temelleri

1) $(\vec{x}_t, d_t) \in P(\vec{x}_t, d_t)$

2) Verilen bir hata fonksiyonu için en iyi algoritma

$$f^* = \arg \min_f E[l(d_t, f(\vec{x}_t))] \quad H(f^*) = E[l(d_t, f^*(\vec{x}_t))]$$

3) Öğrenme için bir model kümesi seçtim F , örneğin CNN's, RNN's gibi

$$f_F^* = \arg \min_{f \in F} E[l(d_t, f(\vec{x}_t))] \quad H(f_F^*) = E[l(d_t, f_F^*(\vec{x}_t))] \geq H(f^*)$$

4) Veri topladım

$$\{(\vec{x}_1, d_1), (\vec{x}_2, d_2), \dots, (\vec{x}_T, d_T)\}$$

5) Toplanan veri için model sınıfımdaki en iyi algoritmayı buldum

$$f_N = \arg \min_{f \in F} \sum_{t=1}^N l(d_t, f(\vec{x}_t)) \quad H(f_N) = E[l(d_t, f_N(\vec{x}_t))] \geq H(f^*)$$

Öğrenmenin temelleri

- 5) Bu üç hata oranı arasındaki ilişki nedir?

$$H(f_N) \geq H(f_F^*) \geq H(f^*)$$

$$H(f_N) - H(f^*) = H(f_F^*) - H(f^*) + H(f_N) - H(f_F^*)$$
$$\underbrace{}_{\text{Err}_{app}}$$
$$\underbrace{}_{\text{Err}_{est}}$$

Yakınsama hatası Kestirim hatası
(Bias) (Variance)

- 6) Gerçek hayatta gözlemlediğim verileri kullanarak model sınıflımda en iyi algoritmayı seçiyorum, örnek CNN'i veriyi kullanarak eğitiyorum.

$$\left\{ (\vec{x}_1, d_1), (\vec{x}_2, d_2), \dots, (\vec{x}_T, d_T) \right\} \rightarrow \tilde{f}_N \rightarrow f_N$$

SGD, GD,
Newton gibi

$$H(\tilde{f}_N) = E \left[l(d_t, \tilde{f}_N(\vec{x}_t)) \right]$$

Öğrenmenin temelleri

$$H(\tilde{f}_N) - H(f^*) = \underbrace{H(\tilde{f}_N) - H(f_N)}_{\text{Err}} + \underbrace{H(f_N) - H(f_F^*)}_{\text{Err}_{opt}} + \underbrace{H(f_F^*) - H(f^*)}_{\text{Err}_{est}} + \underbrace{H(f_F^*) - H(f^*)}_{\text{Err}_{app}}$$

Optimizasyon hatası Kestirim hatası Yakınsama hatası

- Belirli bir bütçe altında, zaman, işlem yükü gibi, bu üç hata değeri arasındaki ilişki bize hangi algoritmayı seçmemiz gerektiğini söyler.
- Eğer çok büyük miktarlarda veriniz var ise çok karışık ve güçlü modelleri yine çok güçlü öğrenme algoritmaları ile eğitmek yerine çok daha az karışık modelleri çok veri ile eğitmek daha iyi olabilir.
- Bu öğrenme literatürünü tamamen değiştirmiş bir yaklaşımdır. Genelde az veriden, çok güçlü model ve öğrenme algoritmaları ile öğrenme yapıılırken, artık çok veriden çok daha basit algoritmalar kullanarak çok daha güçlü modeller üretilmektedir.

Öğrenmenin temelleri

İyi optimizasyon algoritması \neq İyi öğrenme algoritması

$$Err = Err_{opt} + Err_{est} + Err_{app}$$

N

- Geleneksel, az veri var. Öğrenmeyi sınırlayan veri miktarı.
- Ancak az veri olduğu için f_N zaten f^* yakın değil.
- O yüzden çok iyi optimize etsenizde, Err_{opt} küçük olsa da toplam Err küçük olmuyor



- Büyük miktarda veri var. Err_{est} çok düşük ancak bu kadar veriyi iyi bir optimizasyon algoritmasının işlemesi çok zor.
- Yani Err_{opt} düşürelemiyor bu nedenle tüm veri işlenmiyor.
- İyi bir optimizasyon algoritması yerine, daha az ancak çok veri process eden yaklaşımlar genellemeyi en aza indirmek için daha iyi olabilir.

Öğrenmenin temelleri

- SGD algoritması:

$$\vec{w}_{t+1} = \vec{w}_t + \mu e_t \vec{x}_t$$

- GD algoritması:

$$\tilde{\vec{w}}_{k+1} = \tilde{\vec{w}}_k + \mu \sum_{t=1}^N (d_t - \tilde{\vec{w}}_k^T \vec{x}_t) \vec{x}_t$$

- Online Newton Step algoritması:

$$\vec{w}_{t+1} = \vec{w}_t + \mu R_t^{-1} e_t \vec{x}_t \quad R_t = \sum_{r=1}^{t-1} \vec{x}_r \vec{x}_r^T$$

- Bu algoritmaların üçü de N uzunlığında ve n boyutunda bir veriye uygulansın.

$$\{(\vec{x}_1, d_1), (\vec{x}_2, d_2), \dots, (\vec{x}_N, d_N)\}$$

Öğrenmenin temelleri

Algoritma	İşlem yükü	r seviyesinde hataya ulaşmak için gereken iterasyon	r seviyesinde hataya ulaşmak için gereken zaman	Err < H(f*) + e olmasına kadar geçen zaman
SGD	O(n)	O(1/r)	O(n/r)	O(n/e)
GD	O(Nn)	O(log 1/r)	O(N n log 1/r)	O(n ² log(1/e)/e)
ONS	O(n ²)	O(log log 1/r)	O(n ² log log 1/r)	O(n ² log ² (1/e) log(1/e)/e)

$$Err \leq Err_{opt} + r$$

- Az veriden iyi optimizasyon ile çok bilgi çıkarmaya çalışmak yerine, çok veriden azar azar bilgi çıkarıp çok daha iyi öğrenmek mümkün!

Pişmalığa (Regret) dayalı öğrenme

- Büyük veriler hayli karışık, düzensiz ve çoğu zaman istatistiksel bir modele uymuyor.
- Verilerin hızlı, etkin ve depolamaya ihtiyaç duymadan işlenmesi gerekiyor
- Herhangi istatistiksel bir model yoksa performans nasıl ölçülecek ya da algoritmalar nasıl karşılaştırılacak.
- Bir cevap: Pişmanlık analizi
- Bir örnek, doğrusal kestirim.
- Sürekli elinize veri geliyor:

$$\{x(1), x(2), x(3), \dots\} \quad \Rightarrow \quad \{(\vec{x}_1, d_1), (\vec{x}_2, d_2), \dots\}$$

- Tüm verilerinizde olsaydı yani geleceği görebilseydim:

$$\{(\vec{x}_1, d_1), (\vec{x}_2, d_2), \dots, (\vec{x}_N, d_N)\}$$

$$\vec{w}_N = \underset{\vec{w} \in F}{\operatorname{argmin}} \sum_{t=1}^N l(d_t, \vec{w}^T \vec{x}_t)$$

$$Err_N^* = \sum_{t=1}^N l(d_t, \vec{w}_N^T \vec{x}_t)$$

Pişmalığa (Regret) dayalı öğrenme

- Ancak veriler büyük veri uygulamalarında kademeli olarak teker teker geliyor:

$$\{(\vec{x}_1, d_1), (\vec{x}_2, d_2), \dots\}$$

- Her t anında elimdeki veriler ile model üretiyorum:

$$\{(\vec{x}_1, d_1), \dots, (\vec{x}_{t-1}, d_{t-1})\} \rightarrow \tilde{w}_t \rightarrow \sum_{t=1}^N l(d_t, \tilde{w}_t^T \vec{x}_t)$$

- Sıralı algoritmamın performansı geleceği gören algoritma ile aynı olabilir mi?

$$\sum_{t=1}^N l(d_t, \tilde{w}_t^T \vec{x}_t) \approx \operatorname{argmin}_{\vec{w} \in F} \sum_{t=1}^N l(d_t, \vec{w}^T \vec{x}_t)$$

- Yakınsamayı matematiksel olarak tanımlarsak, en iyi algoritmayı seçemediğimz için yaşadığımız pişmanlık:

$$\sum_{t=1}^N l(d_t, \tilde{w}_t^T \vec{x}_t) - \operatorname{argmin}_{\vec{w} \in F} \sum_{t=1}^N l(d_t, \vec{w}^T \vec{x}_t) \leq o(N)$$

Yarışmacı yaklaşım

1) Diyelim ki her farklı ve sabit doğrusal model bir uzman olsun:

$$\hat{d}_{t,v} = \vec{v}^T \vec{x}_t \quad \Rightarrow \quad \sum_{t=1}^N l(d_t, v^T \vec{x}_t)$$

2) Tüm 'v'ler arasında bir tanesi en iyi ama hangisi olduğu bilinmiyor:

$$\vec{v}^* = \operatorname{argmin}_{\vec{w} \in F} \sum_{t=1}^N l(d_t, \vec{w}^T \vec{x}_t)$$

3) En iyi aramak yerine herkesi performanslarına bakıp birlestirelim

$$P(\vec{v}, t) = \frac{\exp\left(-\mu \sum_{k=1}^{t-1} l(d_t, \vec{v}^T \vec{x}_t)\right)}{\int_{\vec{z}} \exp\left(-\mu \sum_{k=1}^{t-1} l(d_t, \vec{z}^T \vec{x}_t)\right) d\vec{z}}$$

$$\tilde{v}_t = \int_{\vec{z}} P(\vec{v}, t) \vec{v} d\vec{v}$$

Yarışmacı yaklaşım

4) Ortaya çıkan algoritma herhangi bir veri dizisi için

$$\sum_{t=1}^N l(d_t, \tilde{v}_t^T \vec{x}_t) - \operatorname{argmin}_{\vec{w} \in F} \sum_{t=1}^N l(d_t, \vec{w}^T \vec{x}_t) \leq O(\ln N)$$

performansını hiçbir varsayıma gerek kalmadan sağlar.

5) Yukarıdaki performans da minimax optimaldır. Yani bu performanstan daha iyi performans veren algoritma yoktur.

6) Peki bu algoritma nedir? Karesel hata için:

$$\vec{v}_t = \left(\sum_{k=1}^{t-1} \vec{x}_k \vec{x}_k^T + \sigma I \right)^{-1} \left(\sum_{k=1}^{t-1} \vec{x}_k d_k \right)$$

7) Pişmanlığı en aza indiren filtre tekrarlamalı en küçük karesel hata (Recursive Least Squares) (RLS) algoritmasının çok az değiştirilmiş halidir.

Yarışmacı yaklaşım

8) Aynı şekilde SGD algoritması için

$$\tilde{v}_{t+1} = \tilde{v}_t - \mu \nabla l(d_t, \tilde{v}_t^T \vec{x}_t)$$

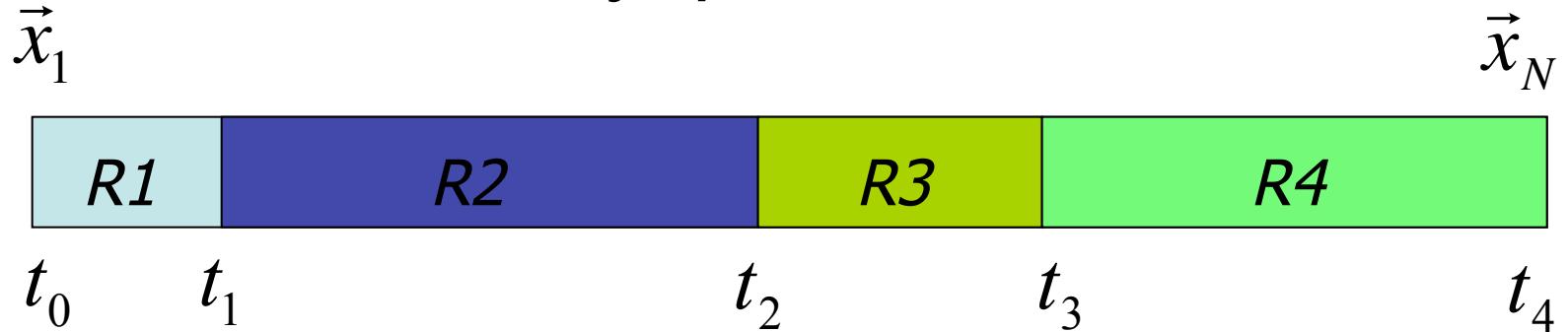
benzer performans sınırları çıkarılabilir:

$$\sum_{t=1}^N l(d_t, \tilde{v}_t^T \vec{x}_t) - \operatorname{argmin}_{\vec{w} \in F} \sum_{t=1}^N l(d_t, \vec{w}^T \vec{x}_t) \leq O(\ln N)$$

9) Herhangi bir istatistiksel varsayımlının olmadığı durumlarda SGD algoritması da belli koşullar altında pişmaliği en aza indirir.

10) Yani birinci derece algoritmalar ikinci derece algoritmalar kadar etkilidir.

Durağan olmayan durumlarda ne yapılmalı?



- Tüm veriler size verilsin:

$$\{(\vec{x}_1, d_1), (\vec{x}_2, d_2), \dots, (\vec{x}_N, d_N)\}$$

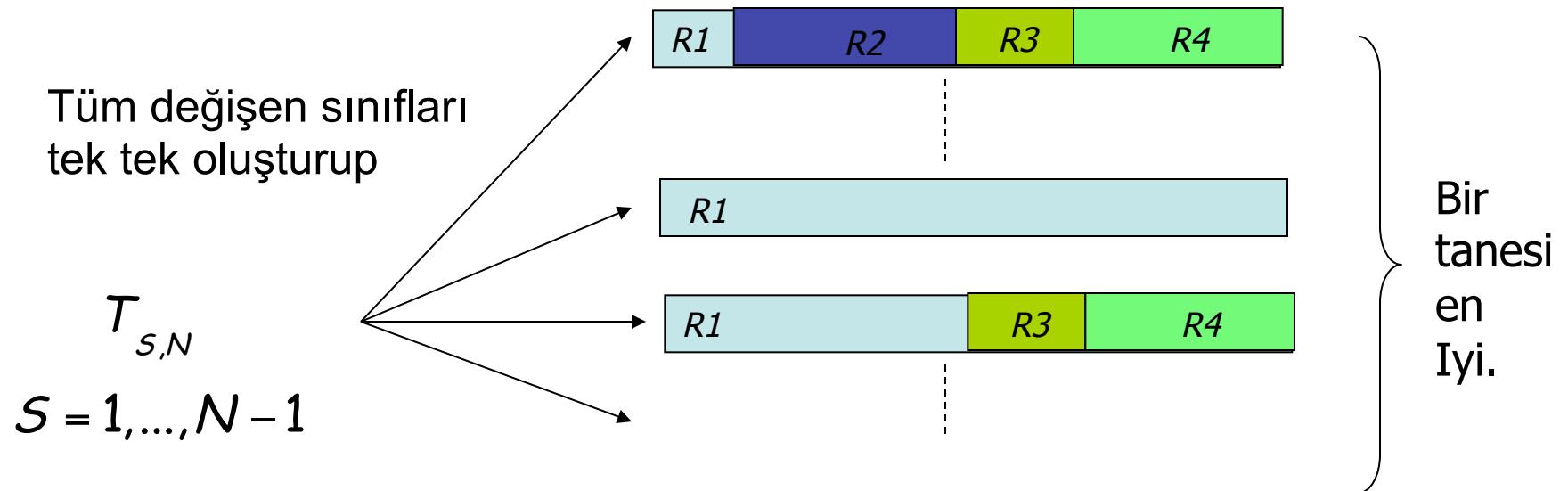
- Bu verileri istediginiz kadar parçaaya bölün, diyelim ki S tane parçaaya, ve her parçaaya da en iyi doğrusal modeli kullanalım:

$$\min_{t_1, t_2, \dots, t_S} \min_{\vec{v}_k \in F} \sum_{k=1}^S \sum_{t=t_k+1}^{t_{k+1}} l(d_t, \vec{v}_k^T \vec{x}_t)$$

- Ve bize veriler hala birer birer verilsin. Geleceği bilmeden bu performansa ulaşabilir miyiz?

Durağan olmayan durumlarda ne yapılmalı?

- 1) Yapılması gerekenler yine aynı. Bu sefer yarışığınız değişen sınıf :



- 2) Herbirini paralel olarak çalıştırıp, en sonunda performansa göre birleştiririrsek ortaya pişmanlığı en aza indirecek bir algoritma çıkar:

$$P(T_{S,N}, t) = \frac{W(T_{S,N}) \exp\left(-\mu \sum_{k=1}^{t-1} l(d_t, T_{S,N})\right)}{\sum_{T_{S,N}} W(T_{S,N}) \exp\left(-\mu \sum_{k=1}^{t-1} l(d_t, T_{S,N})\right)}$$

Durağan olmayan durumlarda ne yapılmalı?

3) Ortaya çıkan sıralı kestirim:

$$\tilde{v}_t = \sum_{T_{S,N}} P(T_{S,N}, t) \tilde{v}_{t,T_{S,N}}$$

minimax olarak en iyidir

$$l(d_t, \vec{v}_t^T \vec{x}_t) - \min_{t_1, t_2, \dots, t_S} \min_{\vec{v}_k \in F} \sum_{k=1}^S \sum_{t=t_k+1}^{t_{k+1}} l(d_t, \vec{v}_k^T \vec{x}_t) \leq O(S \ln N)$$

Yani anahtarlamalı sistemler için daha iyi bir pişmanlık performansı veren bir sistem bulmak mümkün değildir.

İki Borsa Senedinin Hikayesi

- İki borsa senedi olsun, IBM ve Microsoft: kapanış fiyatı $p_j[t]$, $j=1,2$.
- Her bir lira için yapacağım kazanç $y_j[t] = p_j[t]/p_j[t-1]$, $j=1,2$,
 $Y[t] = [y_1[t] \ y_2[t]]$ buna fiyat oran vektörü deniyor.
- Portföy vektörü hangi hisse senedine ne kadar para yatırığınızı gösteriyor.
- $B[t]$ is the portfolio, $B[t] = [b_1[t] \ b_2[t]]$, $b_1[t]+b_2[t]=1$, $b_j[t]>=0$.
- m değişik uzman olsun: $B_1[t], B_2[t], \dots, B_m[t]$. Her bir uzman n gün sonra:

$$\prod_{t=1}^n B_i[t]^T Y[t].$$

İki Borsa Senedinin Hikayesi

- Sıralı olarak n gün sonrası yaptığınız kazanç:

$$\left. \begin{array}{l} B_1[t] \rightarrow \text{kazanç} : \prod_{t=1}^n B_1[t]^T Y[t] \\ \vdots \\ B_m[t] \rightarrow \text{kazanç} : \prod_{t=1}^n B_m[t]^T Y[t] \end{array} \right\} \text{En iyi kazanç: } \max_i \prod_{t=1}^n B_i^T[t] Y[t]$$

- Borsanın her gerçekleşmesi için $\{Y[1], Y[2], \dots, Y[n]\}$ bu uzmanlardan bir tanesi en iyi.
- Ama en iyi bilinmiyor çünkü geleceği bilmiyoruz.

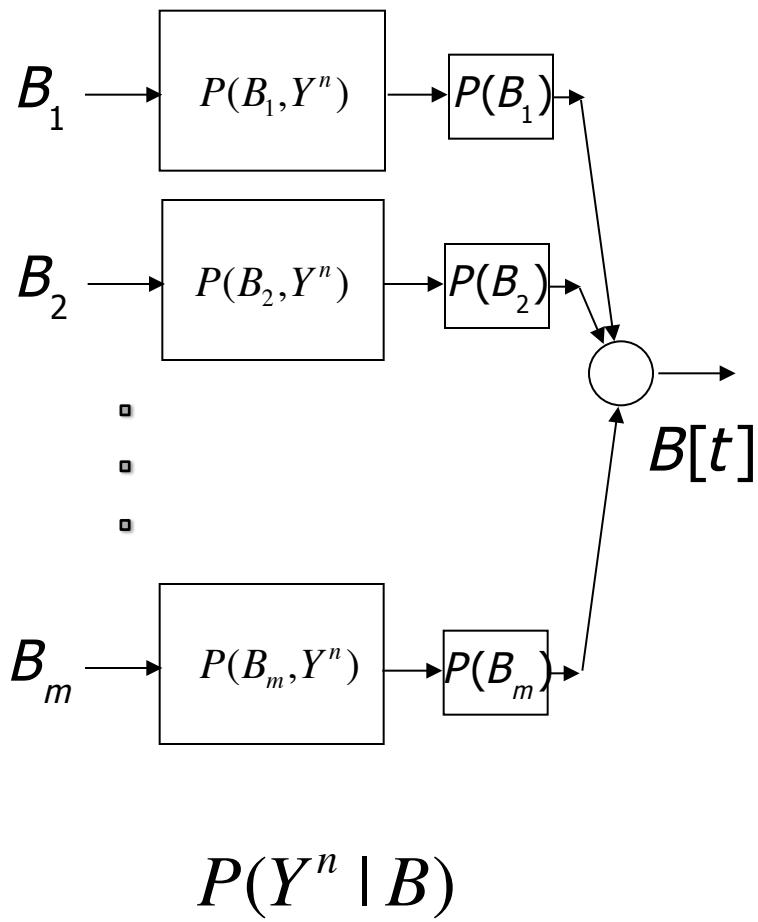
Geleceği yani $X[t]$ 'yi bilmeyen sıralı bir algoritma $B[t]$ ve performansı:

$$\prod_{t=1}^n B^T[t] Y[t] \approx \max \prod_{t=1}^n B_i^T[t] Y[t],$$

herhangi bir borsa gerçekleşmesi için $\{X[1], X[2], \dots, X[n]\}$, mümkün mü?

Sıralı Öğrenme

Varsayımsız Bayeş Karışımı Uzmanların Karışımı



- ❑ $P(B_i, Y^n)$ B_i yatırım algoritmasının $\{Y[1], Y[2], \dots, Y[n]\}$ üzerindeki performansı.
- ❑ $P(B_i, Y^n)$ Bayes öğrenme teorisinde kullanılan şartlı olasılık $P(Y^n|B_i)$ diye düşünülebilir.
- ❑ $P(B_i)$ B_i stratejisine duyduğumuz güven
- ❑ $P(B_i)$ Bayes öğrenme teorisindeki öncül model istatistiği.

$$\sum_{i=1}^m P(B_i)P(B_i, Y^n) \geq \max_k P(B_k)P(B_k, Y^n)$$

- ❑ Öyle bir $B[t]$ bulalım ki $P(B[t], Y^n)$ en az yukarıdaki Bayeşçi karışım kadar iyi olsun. Bunu yaparsak işimiz bitti.

Sıralı Yaklaşım

- $P(X^n) = \sum_{i=1}^m P(B_i)P(B_i, X^n) \geq \max_k P(B_k)P(B_k, X^n).$
- Öyle bir $B(t)$ bulalım ki her X^n için $P(B[t], X^n)$ kadar büyük olsun:
- Yani $P(B[t], X^n) \geq P(X^n) \geq \max_k P(B_k)P(B_k, X^n).$
- Eğer bu şartları sağlayacak $P(B_i, X^n)$, $P(B_i)$ ve $B[t]$, bulabilirsek işimiz bitti. Zengin olduk.
- Sıralı algoritma literatüründe yukarıdakini sağlayan $B[t]$ 'ye 'substitution function' deniyor.

$$B[t] = \sum_{i=1}^m \mu_i[t] B_i[t], \quad \sup_{X^n} \frac{\max_{B_k} \prod_{t=1}^n B_k^T Y[t]}{\prod_{t=1}^n B[t]^T Y[t]} \rightarrow \ln(m).$$

Sıralı Yaklaşım

$$B[t] = \sum_{i=1}^m \mu_i[t] B_i[t], \quad \mu_i[t] = \frac{P(B_k) \prod_{i=1}^{t-1} B_k^T Y[i]}{\sum_{k=1}^m P(B_k) \prod_{i=1}^{t-1} B_k^T Y[i]}.$$

- Bu algoritma herhangi bir fiyat oran dizisine uygulanırsa getireceği kazanç en az kümedeki o dizi için en iyi yatırım kadar iyi olur:

$$\sup_{Y^n} \frac{\max_{B_k} \prod_{t=1}^n B_k^T Y[t]}{\prod_{t=1}^n B[t]^T Y[t]} \rightarrow \ln(m).$$

Daha iyi portföy seçimi

- AMAÇ: Geleceği bilen en iyi portföy seçimi ile yarışmak, uzmanlar ile yarışmak değil:

$$B^* = \arg \max_B \prod_{t=1}^n B^T Y[t]$$

- Yukarıda verilen iki borsa kağıdının olduğu durumda en iyi yatırım stratejisidir. Literatürde bu stratejiye ‘Constant Rebalanced Portfolio’ denilmektedir.
- Biz öyle bir sıralı portföy seçimi algoritması arıyoruz ki:

$$B[t] \rightarrow \{Y[1], Y[2], \dots\}$$
$$\left. \right\} \quad \begin{aligned} & \max_B \prod_{t=1}^n B^T Y[t] \\ & \sup_{Y^n} \frac{\prod_{t=1}^n B[t]^T Y[t]}{\prod_{t=1}^n B[t]^T Y[t]} \rightarrow 1 \end{aligned}$$

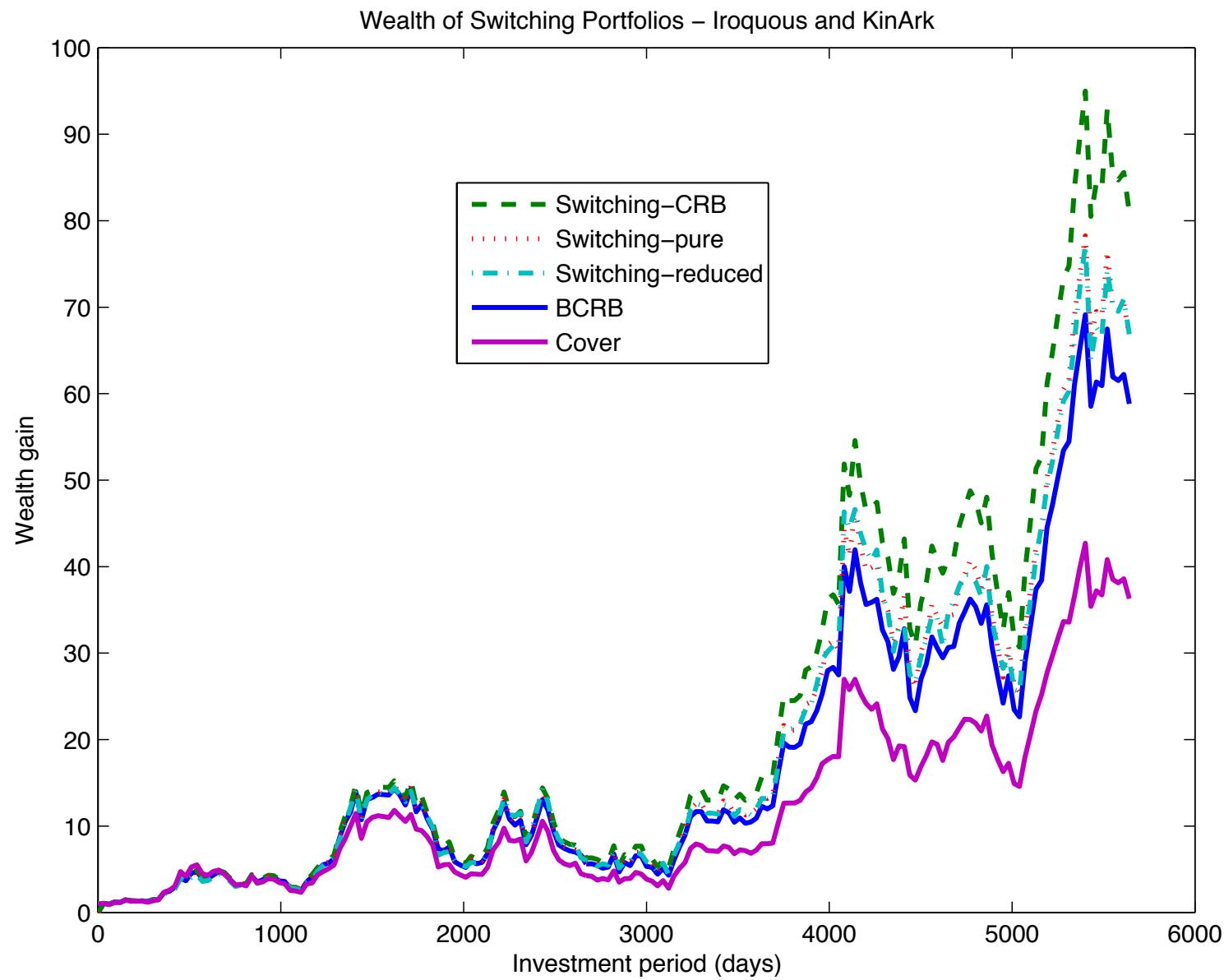
Constant Rebalanced Portfolios

- Her $B \in [0,1]^2$ yatırım stratejisi için.
- Her B için, $P(B, X^{t-1}) = \prod_{j=1}^{t-1} B^T X[j]$, seçelim, yani B 'ye karşılık gelen gerçek kazanç.
- Her B için, öncü istatistiksel dağılım olarak Dirichlet dağılımını seçelim ya da uniform dağılım $P(B)=1$.
- Bayesçi karışım:
$$\int_B \left(\prod_{j=1}^t B^T X[j] \right) \mu(B)$$

$$\int_B B \left(\prod_{j=1}^{t-1} B^T X[j] \right) \mu(B)$$
- Substitution function:
$$B[t] = \frac{\int_B \left(\prod_{j=1}^{t-1} B^T X[j] \right) \mu(B)}{\int_B \left(\prod_{j=1}^t B^T X[j] \right) \mu(B)}.$$
- Biraz matematik:

$$\sup_{X^n} \ln \frac{W(B^*, X^n)}{W(B[t], X^n)} \leq (m-1) \ln(n).$$

iki borsa senedi



Gerçek hayat

