Ağaç Kullanımı ile Doğrusal Olmayan Bağlanım Probleminin Çözümü
A Tree-Based Solution to Nonlinear Regression Problem

Oğuzhan Demir[1], Mohammadreza Mohaghegh N.[1], İbrahim Delibalta[2], Süleyman S. Kozat[1]
[1]Elektrik ve Elektronik Mühendisliği Bölümü, Bilkent Üniversitesi, Ankara, Türkiye
{odemir, mohammadreza, kozat}@ee.bilkent.edu.tr
[2]Türk Telekom Labs, İstanbul, Türkiye
{ibrahim.delibalta}@turktelekom.com.tr

***Özetçe* —Bu bildiride, ardışık doğrusal olmayan bağlanım problemi için yeni bir algortitma sunulmuştur. Bu algoritmada doğrusal olmayan bağlanım modelini bulmak için parçalı doğrusal fonksiyonlar kullanılmıştır. Daha doğru ve hızlı bir yakınsama elde etmek için belirli bir şekilde oluşturulmuş parçalı doğrusal fonksiyonların ağırlıklı kombinasyonu kullanılmıştır. Bu parçalı fonksiyonlar bir "lexicographical" ağacın düğümleri tarafından temsil edilen farklı uyarlanır doğrusal fonksiyonların birleştirilmesi ile oluşturulmuştur. Bu ağaç yapısı hesaplama karmaşıklığını önemli ölçüde azaltmıştır. Yeni sunulan algoritmanın performansını göstermek için, gerçek veri kümelesi ile yapılan deneyler sunulmuştur.**

Anahtar Kelimeler*—*Parçalı doğrusal, doğrusal olmayan bağlanım, lexicographical ağaç, uyarlanır.

***Abstract*—In this paper, we offer and examine a new algorithm for sequential nonlinear regression problem. In this architecture, we use piecewise adaptive linear functions to find the nonlinear regression model sequentially. For more accurate and faster convergence, we combine a large class of piecewise linear functions. These piecewise linear functions are constructed by composing different adaptive linear functions, which are represented by the nodes of a lexicographical tree. With this tree structure, computational complexity of the algorithm is significantly reduced. To show the performance of the proposed algorithm, we present a simulation which is performed by using a well-known real data set.**

Keywords*—*Piecewise linear, nonlinear regression, lexicographical tree, adaptive.

I. GİRİŞ

Sinyal işlemede ve otomatik öğrenmede uyarlanır doğrusal bağlanım sıklıkla kullanılmaktadır [1]. Kanal kestirme, kanal denkleştirme, gürültü yok etme gibi birçok sinyal işleme probleminde uyarlanır doğrusal bağlanım yöntemleri çoğu zaman yeterli performans gösterirler [2]. Bu yöntemlerin bu kadar sık kullanılmasının nedeni doğrusal olmayan bağlanım yöntemlerine göre daha kolay tasarlanması ve analiz edilebilmesinin yanı sıra hesaplama karışıklığının da düşük olmasıdır. Doğrusal olmayan bağlanım yöntemlerinin modelleme gücü doğrusal olanlara göre fazla olmasına rağmen tasarımı zordur ve hesaplama karışıklığı fazladır [3].

Bu bildiride, yukarıda bahsedilen doğrusal olmayan bağlanımdaki problemler uyarlanır doğrusal bağlanım modelleri kullanılarak çözülmüştür. Bu çözümdeki ana yöntem bağlanım düzleminin ayrık bölgere bölünüp, her ayrık bölgede bağımsız uyarlanır doğrusal modeller oluşturmaktır. Bu yöntem parçalı uyarlanır doğrusal bağlanım olarak adlandıralabilir. Bu yöntemdeki zorluk bağlanım düzleminin kaç parçaya bölüneceğinin ve bölünmenin nereden yapılacağının bilinmemesidir. Bu zorluk birçok sayıda farklı parçalı uyarlanır doğrusal bağlanım modellerinin ağırlıklı ortalaması alınarak aşılmıştır. Bu sayede algoritmanın doğrusal olmayan bağlanım fonksiyonlarını modelleme gücü önemli ölçüde artırılmıştır.

Literatürde var olan birçok yöntemde, hesaplama karmaşıklığı daha düşük bir algoritma elde etmek için parçalı uyarlanır doğrusal modelleri oluştururken ağaç yapısı kullanılmıştır. Ağaç yapısı için literatürde sıklıkla kullanılan bağlam veya karar ağaçlarından [4] farklı olarak ağacın modelleme gücünü artırmak amacıyla lexicographical [5] ağaç kullanılmıştır. Parametresi D olan bir ağaç yapısı ile 2^D farklı parçalı uyarlanabilir doğrusal model tanımlıdır ve bu modelleri oluşturmak için yaklaşık olarak $D^2/2$ doğrusal model gereklidir. Lexicographical ağacın bu özelliği sayesinde hesaplama karışıklığı üstel seviyeden ikinci dereceden polinom seviyesine düşürülmüştür.

Bu bildiride önerilen algoritma sayesinde, hesaplama karışıklığı bakımından varolan diğer ardışık doğrusal olmayan bağlanım algoritmalarına göre daha fonksiyonel olan bir algoritma elde edilmiştir. Daha önemlisi bu algoritma çalışılması zor olduğu düşünülen doğrusal olmayan bağlanım problemlerini, üzerinde çok çalışılmış ve iyi bilinen doğrusal bağlanım problemleri ile ilişkilendirerek kolaylaştırmıştır. Ayrıca, bu sayede doğrusal bağlanım problemleri için geliştirilmiş tüm algoritmalar basit bir şekilde bu bildiride oluşturulan algoritmanın içinde kullanılabilir.

Bu bildiri problemin matematiksel tanımı, lexicographical ağaç algoritmasının oluşturulması, verimli hesaplama yönteminin tanımlanması, simulasyon ve sonuçlar kısmının verilmesi şeklinde devem edecektir.

978-1-5090-1679-2/16/$31.00

II. PROBLEM TANIMI

Bu bildiride istenen sinyal $\{d_t\}_{t\geq 1}$, $d_t \in \Re$ and bağlanım vektörleri $\{\boldsymbol{u}_t\}_{t\geq 1}$, $\boldsymbol{u}_t \in [-A, A]^m$, $A \in \Re$ ile ifade edilmektedir. Bu bildirideki temel amaç bağlanım vektörleri ile istenen sinyal arasındaki bağlantıyı veren dogrusal olmayan uyarlanır fonksiyonu bulmaktır:

$$\hat{d}_t = f_t(\boldsymbol{u}_t). \tag{1}$$

Denklem (1)'de, $\hat{d}_t$ doğrusal olmayan uyarlanabilir fonksiyonun $t \geq 1$ anındaki kestirimini göstermektedir. Bu tahmine bağlı olarak yapılan hata

$$e_t = d_t - \hat{d}_t \tag{2}$$

ile gösterilir. Bu bildiride, n anındaki en iyi doğrusal olmayan uyarlanır fonksiyonu bulurken asağıdaki optimizasyon problemini kullandık:

$$\min_{f_n(.)} \sum_{t=1}^{n} e_t^2. \tag{3}$$

(3)'teki problem çözülmesi çok zor bir problem, çünkü $f_n(.)$'i bulmak için tüm doğrusal olmayan bağlanım fonksiyonlarını düşünmemiz gerekir. Aslında derecesi n olan bir polinomla toplam hatayı 0 yapan $f_n(.)$'i bulabiliriz, ancak bu hesaplama karışıklığı bakımından çözülmesi çok zor bir problem olur. Bu nedenle (3)'de tanımalanan genel problemi çözülebilir hale getirmek için $f_n(.)$'e parçalı doğrusal fonksiyon olma kısıtlaması getirdik. Bağlanım uzayı $[-A, A]^m = \cup_{p=1}^{P} R_p$ ve $R_i \cap R_j = \emptyset, \forall i, j \; 1 \leq i \neq j \leq P$ şeklinde parçalandığında, t anındaki $f_t(.)$'yi aşağıdaki gibi ifade edebiliriz:

$$f_t(\boldsymbol{u}_t) = \boldsymbol{v}_{t,I(\boldsymbol{u}_t)}{}^T \boldsymbol{u}_t + b_{t,I(\boldsymbol{u}_t)}. \tag{4}$$

burada $I(\boldsymbol{u}_t) = p$ öyleki eğer $\boldsymbol{u}_t \in R_p$ ve $p = 1, 2, ..., P$. (4)'de verilen denklemi $\boldsymbol{u}_t = [\boldsymbol{u}_t; 1]$ and $\boldsymbol{v}_{I(\boldsymbol{u}_t)} = [\boldsymbol{v}_{I(\boldsymbol{u}_t)}; b_{I(\boldsymbol{u}_t)}]$ değişikliklerini yaparak aşağıdaki gibi ifade edebiliriz:

$$f_t(\boldsymbol{u}_t) = \boldsymbol{v}_{t,I(\boldsymbol{u}_t)}{}^T \boldsymbol{u}_t. \tag{5}$$

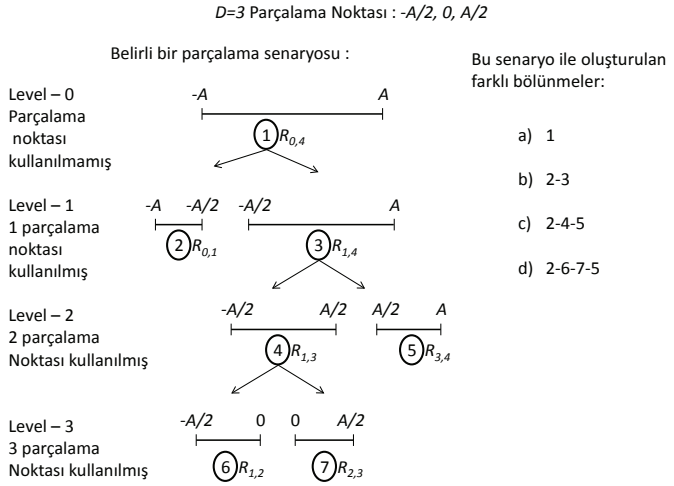
Problemi bu şekilde kolaylaştırdıktan sonra, her t anındaki bağlanım vektörü gözleminden sonra sadece bu andaki bağlanım vektörünün içinde bulunduğu ayrık bölgedeki doğrusal uyarlanır modeli güncelleyerek aradığımız parçalı doğrusal modele doğru yaklaşabiliriz. Bu bildiride doğrusal modelleri güncellemek için olasılıksal bayır inişi (SGD) [2] algoritmasını kullandık.

Eğer problemi çözmeden önce bağlanım uzayını nasıl parçalayacağımızı bilseydik, bu aşamada problemi çözmek için (5)'i ve SGD algoritmasını kullanabilirdik. Biz bu bilgiye hiçbir şekilde sahip olmadığımızı kabul ediyoruz, bu nedenle çözülmesi gereken problem aşagıdaki gibi ifade edilmelidir:

$$\begin{aligned} &\boldsymbol{v}_1^*, \boldsymbol{v}_2^*, ..., \boldsymbol{v}_P^*, R_1^*, R_2^*, ..., R_P^* \\ &\quad = \min_{R_1, R_2, ..., R_P} \min_{\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_P} \sum_{t=1}^{n} (d_t - \boldsymbol{v}_{I(\boldsymbol{u}_t)}{}^T \boldsymbol{u}_t)^2, \end{aligned} \tag{6}$$

(6)'daki problem yine çözülmesi çok zor bir problemdir. Bu nedenle bu problemi çözmek yerine, D parametreli lexicographical ağacı kullanarak 2^D tane parçalı doğrusal bağlanım modeli oluşturup, bu modellerin ağırlıklı ortalamasını alıyoruz.

Bir dahaki bölümde, lexicographical parçalama yöntemini tanıtacağız, daha sonra bu parçalama sonucunda oluşabilecek



Şekil 1. $D = 3$ ve $u_t \in [-A, A]$ durumunda lexicographical parçalama grafiği.

tüm parçalı uyarlanabilir doğrusal modelleri, hesaplama açısından nasıl etkili bir şekilde birleştirebileceğimizi göstereceğiz.

III. LEXICOGRAPHICAL AĞAÇ ALGORİTMASI

Algoritmanın son halini sunmadan önce, parametresi D olan lexicographical parçalama sonucunda nasıl 2^D parçalı doğrusal model elde edebileceğimizi, oluşan bu modelleri nasıl birleştirebileceğimizi ve son olarak bu işlemin hesaplama karışıklığını nasıl azaltacağını göstereceğiz.

Anlatım açısından kolay olması için, lexicographical parçalamayı ve algoritmanın kurulmasını 1-boyutlu bağlanım uzayı için açıklayacağız. Algoritmanın yüksek boyutlu uzaya uyarlanması basit bir şekilde yapılabilir.

A. 1-Boyutlu Uzayın Lexicographical Parçalanması

1-Boyutlu durum için bağlanım uzayı bir doğru parçası oluşturur öyleki tüm $t \geq 1$ için $\boldsymbol{u}_t = u_t \in [-A, A]$. D-parametreli lexicographical parçalama için, D adet parçalama hiperdüzlemi kullanabiliriz. 1-Boyutlu durum için, bu hiperdüzlemler $[-A, A]$ içinde bulunan birer noktadır. Bu noktaları istediğimiz şekilde seçebiliriz. Bu bildiride bu noktaları bağlanım doğru parçasını $D + 1$ eşit uzunluktaki doğru parçasına bölecek biçimde seçtik. Şekil 1'in üst kısmında bu noktaların $D = 3$ durumunda nasıl seçildiğini görebiliriz. Bu şekilden takip edilebileceği üzere şekilde verilen grafiğin her seviyesinde kullanılmamış olan herhangi bir parçalanma noktası kullanılarak bağlanım noktası olabildiğince küçük parçalara ayrılmaktadır. Şekil 1'de görülebileceği üzere bu işlem tüm kullanılmamış parçalanma noktaları için her seviyede yalnızca 1 parçalama noktası kullanılmak şartı ile yapılmaktadır. Bu sayede, tüm parçalanma senaryoları tamamlandığında, tüm parçalama noktalarının kullanıldığı ya da kullanılmadığı tüm parçalanmış bağlanım doğrusu kombinasyonları elde edilebilir. Bu nedenden dolayı, D parametreli lexicographical parçalama yöntemi ile 2^D farklı parçalı doğrusal bağlanım fonksiyonu elde edilebilir.

Buradan itibaren, lexicographical parçalanma ile oluşan doğru parçlarını $R_{i,j}$ 'ler ile adlandıracağız. Bu adlandırma,

$0 \leq i < j \leq D+1$ ve $R_{i,j} \cup R_{j,k} = R_{i,k}$ özellikleri sağlanacak şekilde yapılacaktır. Bu adlandırma sayesinde gözlemleyebileceğimiz üzere tüm lexicographical bölünmeleri oluşturabilmek için $D+1$ adet $R_{i,i+1}$ şeklinde, D adet $R_{i,i+2}$ şeklinde, ve bu şekilde devam ederek 1 adet $R_{0,D+1}$ şeklinde doğru parçasına ihtiyacımız olduğunu gözlemleyebiliriz. Bu doğru parçaları toplamda yaklaşık olarak $D^2/2$ tanedir. Daha önce belirtildiği üzere bu doğru parçaları ile 2^D farklı lexicographical bölünme oluşturulabilir. Buna ek olarak yaptığımız her gözlemde $D^2/2$ tane doğrusal bağlanım modelinin hepsini güncellememize gerek yoktur. Eğer bir bağlanım vektörü $R_{i,i+1}$ bölgesine aitse bu durumda $R_{k,i+1}$' den $R_{k,D+1}$'e kadar olan tüm bölegelerdeki doğrusal bağlanım modellerini güncellemeliyiz. Burada $0 \leq k \leq i$ olmak durumundadır. Bu nedenle eğer bir bağlanım vektörü $R_{i,i+1}$ bölgesinde ise $(D-i+1)(i+1)$ tane doğrusal bağlanım modeli güncellenmelidir. Bu gözlemler ağaç yapısının neden hesaplama karışıklığını düşürmeye yardımcı olduğunu göstermektedir.

B. Parçalı Doğrusal Bağlanım Modellerin Ardışık Ortalaması

Önceki bölümlerde anlattığımız üzere D parametreli lexicographical parçalanma ile oluşan $K = 2^D$ farklı bölünmenin her ayrık bölgesinde bağımsız bir doğrusal bağlanım modeli öğrenilerek, $K = 2^D$ parçalı doğrusal bağlanım modeli elde edebiliriz. Algoritmamızın kestirimini oluşturmak için tüm bu parçalı doğrusal modellerin ağırlıklı ortalamasını alacağız. Bu ortalamayı iyi bilinen EG [6] algoritmasını kullanarak alacağız. Eğer herhangi bir t anında K parçalı doğrusal modelin çıktılarını $\hat{d}_t^{\{k\}}$, $k = 1, 2, ..., K$ şeklinde ifade edersek. Algoritmamızın t anındaki son çıkıtısı olan $\hat{d}_t$ aşağıdaki gibi hesaplanabilir:

$$\hat{d}_t = \sum_{k=1}^{K} \hat{d}_t^{\{k\}} r_t^{\{k\}} \tag{7}$$

Kombinasyon ağırlıkalrı olan $r_t^{\{k\}}$ değerleri aşagıdaki şekilde güncellenmektedir:

$$r_{t+1}^{\{k\}} = \frac{r_t^{\{k\}} \exp(-\eta \epsilon_t \hat{d}_t^{\{k\}})}{\sum_{j=1}^{K} r_t^{\{j\}} \exp(-\eta \epsilon_t \hat{d}_t^{\{j\}})} \tag{8}$$

(8)' de $\epsilon_t = (e_t^2)' = -2e_t$ ve $\eta > 0$ sabit değerli bir öğrenim katsayısıdır. Bu ağırlıklı ortlama yöntemi ile modellerin elede edilebilcek en iyi ağırlıklı ortalamasına ulaşabiliriz. Eğer başlangıçta $r_t^{\{k\}} = 1/K$, $k = 1, 2, ..., K$ şeklinde seçersek ve istenen sinyalin tüm t anlarında $\mid \hat{d}_t \mid \leq M$ koşulunu sağladığını kabul edersek, [6]'daki Teorem 5.10, yukarıda tarif edilen ağırlıklı ortalama yöntemi kullanıldığında aşağıdaki performans üst sınırı elde etmemizi sağlar:

$$\sum_{t=1}^{n} (d_t - \hat{d}_t)^2 - \min_{r_n^{\{1\}}, ..., r_n^{\{K\}}} \sum_{t=1}^{n} (d_t - \sum_{k=1}^{K} \hat{d}_t^{\{k\}} r_t^{\{k\}})^2 \leq O(\sqrt{n}) \tag{9}$$

(9)'da verilen performans üst sınırı, algoritmamızın en iyi lexicographical parçalı dogrusal bağlanım modelinden daha iyi çalışacağını gösterir. Bu durumda bölünme noktalarının sayısı D'yi yeterince artırdığımızda, algoritmamız önceden bilinmeyen doğru bölünmenin performansına yeterince yaklaşabilir. Algoritma bu haliyle hesaplama karışıklığı bakımından yeterli değil, çünkü bu durumda hesaplama karışıklığı parçalı doğrusal olmayan bağlanım modellerinin sayısı ile doğru orantılı. Bir diğer ifade ile hesaplama karışıklığı bu durumda $O(2^D)$. Bu hesaplama karışıklığı ile algoritma büyük D değerleri için kullanışlı değil. Hesaplama karışıklığı bir dahaki bölümde özyineli bir yaklaşım kullanılarak ikinici dereceden polinom seviyesine düşürülecektir.

C. Hesaplama Karışıklığının Azaltılması

Bu bölümde algoritmamızı hesaplama karışıklığı bakımından efektif bir hale getirdik. İlk olarak, basit matematiksel işlemler sonucunda başlangıçta eşit ağırlıklı ortalama alarak, (8) aşağıdaki gibi ifade edilebilir:

$$r_t^{\{k\}} = \frac{\exp(-\eta \sum_{z=1}^{t-1} \epsilon_z \hat{d}_z^{\{k\}})}{\sum_{j=1}^{K} \exp(-\eta \sum_{z=1}^{t-1} \epsilon_z \hat{d}_z^{\{j\}})} \tag{10}$$

(10)'daki işlemi daha efektif bir şekilde yapmak için lexicographical ağacın düğümleri için değişkenler tanımlayacağız:

$$B_t{}^{\{R_{i,j}\}} = \exp(-\eta \sum_{z=1}^{t-1} \mathbf{1}_{x_t} \epsilon_z \hat{d_z}^{\{R_{i,j}\}}) \tag{11}$$

$$\mathbf{1}_{u_t} = \begin{cases} 1 & : u_t \in R_{i,j} \\ 0 & : u_t \notin R_{i,j} \end{cases}$$

Bu değişken ile (10)'un paydası arasında doğal olarak aşıdaki eşitlik ortaya çıkar:

$$\sum_{k=1}^{K} B_t{}^{\{k\}} = \sum_{k=1}^{K} \prod_{R_{i,j} \in R^{\{k\}}} B_t{}^{\{R_{i,j}\}}, \tag{12}$$

burada $R^{\{k\}}$ k'ıncı lexicographical modelin içerdiği bölgelerin kümesidir. Bu gerçeklerden yola çıkarak aşağıda verilen özyineli denklemleri oluşturduk:

$$C_t{}^{\{R_{i,j}\}} = B_t{}^{\{R_{i,j}\}} + \sum_{k=i+1}^{j-1} C_t{}^{\{R_{i,k}\}} B_t{}^{\{R_{k,j}\}} \tag{13}$$

$$\tilde{C}_t^{\{R_{i,j}\}} = \tilde{B}_t^{\{R_{i,j}\}} + \sum_{k=i+1}^{j-1} \tilde{C}_t^{\{R_{i,k}\}} \tilde{B}_t^{\{R_{k,j}\}} \tag{14}$$

burada

$$\tilde{B}_t^{\{R_{i,j}\}} = \begin{cases} B_t{}^{\{R_{i,j}\}} \hat{d}_t{}^{\{k\}} & : u_t \in R_{i,j} \subset R^{\{k\}} \\ B_t{}^{\{R_{i,j}\}} & : x_t \notin R_{i,j} \subset R^{\{k\}} \end{cases}$$

Tümevarım yöntemi ile kolayca göseterilebileceği üzere algoritmamızın son çıktısını bu özyinemli ifadeler cinsinden aşağıdaki ifade ile hesaplayabiliriz:

$$\hat{d}_t = \frac{\tilde{C}_t^{\{R_{0,D+1}\}}}{C_t{}^{\{R_{0,D+1}\}}}, \tag{15}$$

1: **Değişkenler:**
2: η : EG algoritmasının öğrenme katsayısı
3: D : Lexicographical ağacın derinliği
4: $A : \mid \{u_t\}_{t\geq 1} \mid < A$
5: **Başlangıç:**
6: $B_0^{R_{\{i,j\}}} = 1$, $C_0^{R_{\{i,j\}}}$'leri, (13)'ü kullanarak hesapla
7: **Algoritma:**
8: **for** $t = 1$ **to** n **do**
9: $V_t = \{R_{i,j} : u_t \in R_{i,j}\}$
10: **for all** $\{R_{i,j}\} \in V_t$ **do**
11: $\hat{d}_t^{\{R_{i,j}\}} = \boldsymbol{v}_{t-1,i,j}^T \boldsymbol{u}_t$
12: **end for**
13: $\tilde{C}_t^{R_{\{i,j\}}}$'leri, (14)'ü kullanarak hesapla
14: $\hat{d}_t = \frac{\tilde{C}_t^{\{R_{0,D+1}\}}}{C_t^{\{R_{0,D+1}\}}}$
15: $\epsilon_t = l'(d_t, \hat{d}_t)$
16: **for all** $\{R_{i,j}\} \in V_t$ **do**
17: $B_{t+1}^{\{R_{i,j}\}} = B_t^{\{R_{i,j}\}} \exp(-\eta \epsilon_t \hat{d}_t^{\{R_{i,j}\}})$
18: LMS algortitması [2] ile $\boldsymbol{v}_{t,i,j}$'leri güncelle [2]
19: **end for**
20: $C_{t+1}^{R_{\{i,j\}}}$'leri, (13)'ü kullanarak hesapla
21: **end for**

Tablo I. LEXİCOGRAPHİCAL AĞAÇ ALGORİTMASININ ALGORİTMİK TARİFİ.

Buna ek olarak $B_t^{\{R_{i,j}\}}$ değişkeninin ardışık güncellemesini

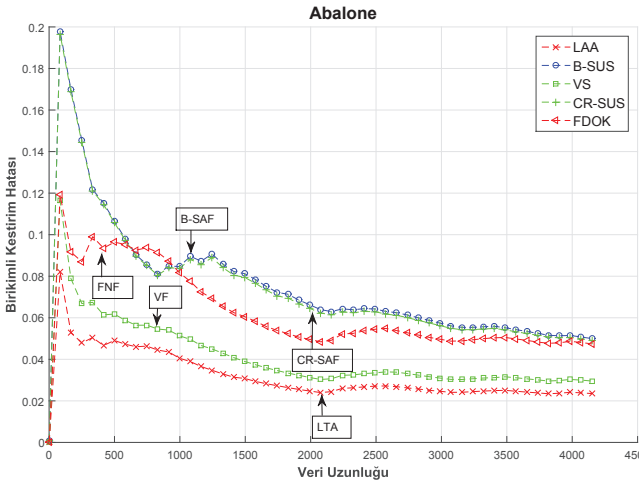
$$B_{t+1}{}^{\{R_{i,j}\}} = \begin{cases} B_t{}^{\{R_{i,j}\}} \exp(-\eta \epsilon_t \hat{d}_t{}^{\{k\}}) & : u_t \in R_{i,j} \\ B_t{}^{\{R_{i,j}\}} & : u_t \notin R_{i,j} \end{cases} \tag{16}$$

şeklinde yaparak (15)'deki hesabın ardışık olarak hesaplanmasını gerçekleştiririz. (16)'da $R^{\{i,j\}} \subset R^{\{k\}}$ koşulu sağlanmalıdır.

Daha önce belirtildiği üzere, her gözlemden sonra doğrusal bağlanım modellerinden sadece $(D-i+1)(i+1)$, $0 \leq i \leq D$ tanesini güncellememiz gerekiyordu. (16)'dan da anlaşılacağı üzere $B_{t+1}^{\{R_{i,j}\}}$ değişkenini sadece son bağlanım vektörünü içeren bölgeler için güncellememiz yeterli. Bu nedenle özyinemli hesaplama tekniği kullanıldığında, algoritmanın hesaplama karışıklığı $O(D^2)$ seviyesine düşmektedir. Bu karışıklık önceden $O(2^D)$ seviyesindeydi. Bu sayede algoritmamımızı hesaplama karışıklığı bakımından da önemli ölçüde geliştirmik olduk.

IV. DENEYLER

Bu bölümde bu bildiride önerilen Lexicographical Ağaç Algoritmasının (LAA) otomatik öğrenme literatüründe sıklıkla kullanılan gerçek data setlerinden biri olan Abalone veri kümesi [7] üzerindeki kestirim performansı sunulacaktır. Bu kestirim performansı Şekil 2'de görülebilir. Bu şekilde, buna ek olarak parametresi 3 olan LAA algoritmasının ve sıklıkla doğrusal olmayan süzgeçlemede kullanılan "VS" ikinci dereceden Volterra Süzgeciyle [8], "FDOK" üçüncü dereceden Fourier doğrusal olmayan kestirim algoritmasıyla [9] ve "SUS" Spline Uyarlanabilir Süzgeç algoritmalarıyla karşılaştırılması da verilmiştir. Şekil 2'de görüldüğü üzere bu bildiride önerilen LAA algoritması Abalone veri kümesi için literatürdeki diğer doğrusal olmayan süzgeçleme yöntemlerine göre daha iyi kestirim performansı göstermiştir. Bu simulasyonu gerçekleştirirken bağlanım uzayını bağlanım vektörlerini dik olarak kesen hiperuzaylarla parçaladık. Bu yolla bildiride anlatılan tek boyutlu uzay için parçalama yöntemi, aynı şekilde çok boyutlu bağlanım uzayları için de kullanılabilir.



Şekil 2. LAA, B-SUS, CR-SUS, VS ve FDOK algoritmalarının Abalone data kümesi üzerindeki birikimli kestirim performansları

V. SONUÇLAR

Bu bildiride sinyal işleme literatüründeki zor problemlerden biri olan doğrusal olmayan bağlanım probleminin çözümü, problemi literatürde iyi bilinen ve göreceli olarak basit olan doğrusal bağlanım problemi ile ilişkilendirilerek kolay bir hale getirilmiştir. Bu bildirideki yöntem hesaplama karmaşıklığı açısından efektif bir yöntemdir. Önerilen algoritmanın çalışması için bağlanım vektörlerinin ve istenen sinyalin büyüklüğünün sınırlı olması dışında gerekli hiç bir bilgiye ve koşula ihtiyaç yoktur. Ayrıca önerilen algoritmanın birikimli kestirim performansı diğer doğrusal olmayan süzgeçleme yöntemleri ile karşılaştırılmış ve onlardan daha iyi performans gösterebileceği gözlemlenmiştir.

KAYNAKÇA

[1] S. Kozat and A. Erdogan, "Competitive linear estimation under model uncertainties," *Signal Processing, IEEE Transactions on*, vol. 58, no. 4, pp. 2388–2393, April 2010.

[2] A. H. Sayed, *Fundamentals of Adaptive Filtering*. NJ: John Wiley & Sons, 2003.

[3] O. J. J. Michel, A. O. Hero, and A.-E. Badel, "Tree-structured nonlinear signal modeling and prediction," *IEEE Transactions on Signal Processing*, vol. 47, no. 11, pp. 3027–3041, 1999.

[4] S. S. Kozat, A. C. Singer, and G. C. Zeitler, "Universal piecewise linear prediction via context trees," *IEEE Transactions on Signal Processing*, vol. 55, no. 7, pp. 3730–3745, 2007.

[5] F. Willems, Y. Shtarkov, and T. Tjalkens, "Context weighting for general finite-context sources," *Information Theory, IEEE Transactions on*, vol. 42, no. 5, pp. 1514–1520, Sep 1996.

[6] J. Kivinen and M. K. Warmuth, "Exponentiated gradient versus gradient descent for linear predictors," *Journal of Information and Computation*, vol. 42, no. 5, pp. 1514–1520, 1996.

[7] C. Blake and C. Merz, "UCI Machine Learning Repository." [Online]. Available: http://archive.ics.uci.edu/ml/index.html

[8] V. Mathews, "Adaptive polynomial filters," *Signal Processing Magazine, IEEE*, vol. 8, no. 3, pp. 10–26, July 1991.

[9] A. Carini and G. Sicuranza, "Recursive even mirror fourier nonlinear filters and simplified structures," *Signal Processing, IEEE Transactions on*, vol. 62, no. 24, pp. 6534–6544, Dec 2014.