Uyarlanır Uzay Bölümleme ile Çevrimiçi Sınıflandırma

Online Adaptive Hierarchical Space Partitioning Classifier

O. Fatih Kılıç¹, N. Denizcan Vanlı² Hüseyin Özkan¹, İbrahim Delibalta³, Suleyman S. Kozat¹

¹Elektrik ve Elektronik Mühendisliği Bölümü, İhsan Doğramacı Bilkent Üniversitesi, Ankara, Türkiye
{kiloc, huseyin, kozat}@ee.bilkent.edu.tr

²Elektrik ve Elektronik Mühendisliği Bölümü, Massachusetts Teknik Üniversitesi, Cambirdge, MA
denizcan@mit.edu

³Türk Telekom Labs, İstanbul, Türkiye
ibrahim.delibalta@turktelekom.com.tr

Özetçe —Bu bildiride, denetimli öğrenim için görgül sıfır-bir kayıp altında yüksek performans gösteren çevrimiçi bir algoritma sunuyoruz. Sunulan bu yöntem uyarlanır olarak öznelik uzayını hiyerarşik bir şekilde parçalara ayırıp etkili bir sınıflandırma algoritması oluşturmak için bu basit modeller için güçlü bir birleşim oluşturmaktadır. Burada ortaya koyduğumuz yöntem hesaplama karmaşıklığı açısından oldukça ölçeklendirilebilir düzeydedir. örn. öznelik uzayının boyutları ile doğrusal olarak ölçeklenebilir. Yaygın olarak kullanılan veri kümeleri üzerinde yaptığımız deneylerde sunulan algoritmanın diğer gelişmiş tekniklere göre daha üstün performans gösterdiğini de bildiriyoruz.

Anahtar Kelimeler—çevrimiçi öğrenme, sınıflandırma, uyarlanır ağaçlar, etkili hesaplama.

Abstract—We introduce an on-line classification algorithm based on the hierarchical partitioning of the feature space which provides a powerful performance under the defined empirical loss. The algorithm adaptively partitions the feature space and at each region trains a different classifier. As a final classification result algorithm adaptively combines the outputs of these basic models which enables it to create a linear piecewise classifier model that can work well under highly non-linear complex data. The introduced algorithm also have scalable computational complexity that scales linearly with dimension of the feature space, depth of the partitioning and number of processed data. Through experiments we show that the introduced algorithm outperforms the state-of-the-art ensemble techniques over various well-known machine learning data sets.

Keywords—on-line learning, classification, adaptive trees, computational efficiency.

I. GİRİŞ

Günümüzde bilgi teknolojilerindeki gelişmeler nedeniyle genellikle yapılanmamış karmaşık formda hızlı bir şekilde akan verileri işlememiz gerekmektedir [1], [2]. Bu nedenle, biz bu bildiride özellikle büyük ihtimale ilintili olan rastgele veri akımları için özgün ve etkili bir çevrimiçi sınıflandırma algoritması sunuyoruz.

Algoritmamız karmaşık (fazlasıyla doğrusal olmayan) sınıflandırma sınırlarını yakınsamak için parçalı doğrusal fonksiyonları ve yakınsama problemlerini azaltmak için de

bölgesel düzenleri kullanmaktadır. Özellikle de her bölümünde sıralı olarak eğittiğimiz basit doğrusal sınıflandırıcılar kullanan hiyerarşik bir yapı kullanmaktadır. Bu özellik sayesinde her bir parça bizim temel sınıflandırıcı olarak adlandırdığımız farklı bir doğrusal olmayan sınıflandırma modeline karşılık gelmektedir ve bizim sistemimizde bu modellerin tamamı temel sınıflandırıcıların rekabet sınıfını oluşturmaktadır. Tanımladığımız bu rekabet sınıfını biz daha sonra bölüm parametreleri (bölge ayırıcılar) üzerine parametrize edip olasılıksal bayır inişi yöntemi ile optimize etmekteyiz. Bu optimizasyon sayesinde gelen veri akışı doğrultusunda rekabet sınıfı bölümlendirme yapısını değiştirerek sürekli bir şekilde kendini geliştirmektedir.

Sunduğumuz çevrimiçi sınıflandırma algoritması her anda temel sınıflandırıcıların çıktılarını birleştirerek nihai sınıflandırma çıktısını üretir. Tanımlanan bu birleştirme ile sunduğumuz algoritma veri üzerinde hiçbir istatistiki varsayıma dayanmadan asimptotik olarak en iyi temel sınıflandırıcının performansına ulaşmaktadır. Sunulan algoritmanın hesaplama karmaşıklığı bütün veriler için verinin boyutuna ve kullanılan hiyerarşik modelin derinliğine bağlı olarak doğrusal bir şekilde değişmektedir ve biz hiyerarşik yapıdaki doğrusal modellerin birleşimi için sınırlı sayıda bir birleşim kullandığımız için algoritma iyi bir genelleme yapabilir ve veri üzerine aşırı uyum sağlamaz (ya da sınırlı bir aşırı uyum yapar) [3], [4].

II. PROBLEM TANIMI

Öznelik vektörlerinin çevrimiçi gözlemlendiği ve bunlara karşılık gelen etiketlere de çevrimiçi karar verilen bir ortamda ikili çevrimiçi sınıflandırma yapıyoruz. ¹ Burada amacımızı vektör $\mathbf{x}_t \in \mathbb{R}^p$ ve $y_t \in \{-1, 1\}$ olduğu $f_t(\mathbf{x}_t)$ biçiminde bir çevrimiçi sınıflandırıcı oluşturmaktır. Bu sınıflandırıcı için gözleme dayalı kayıp

$$L_T(f_t) \triangleq \sum_{t=1}^T \mathbb{1}_{\{f_t(\mathbf{x}_t) \neq y_t\}}, \quad (1)$$

olarak tanımlanmıştır ve herhangi T uzunluğundaki bir dizi için en az rekabet sınıfı $\mathcal{S}(\phi)$ 'deki en iyi temel sınıflandırıcı olan $C(\phi)$ kadar gözlemsel kaybı olabilir. Temel

¹Kullanılan bütün vektörler sütun vektörüdür ve kalın küçük harflerle ifade edilmişlerdir. Bildiri boyunca zaman indisleri altsimges ile belirtilmiştir.

sınıflandırıcıların bir kümesi olan ve parametreye bağlı olan rekabet sınıfı $\mathcal{S}(\phi)$, ϕ üzerinden optimize edilebilir. Burada ϕ parametresi temel sınıflandırıcılar için bir parametre değildir, fakat rekabet sınıfını doğrudan optimize edebilir. Bu doğrultuda, sınıflandırıcımız f_t kendini sürekli olarak geliştiren en iyi rekabetçiye karşı yarışmaktadır.

Sınıflandırıcı f_t 'nin temel sınıflandırıcı $f_t^{(C)}$ 'ye karşı göreceli performansını ölçmek için

$$R_T(f_t; f_t^{(C)}) \triangleq \frac{1}{T} [L_T(f_t) - L_T(f_t^{(C)})] \quad (2)$$

şeklinde herhangi T uzunluğundaki dizin için tanımlanan zarar fonksiyonunu kullanacağız. Amacımız tanımlanan bu zarar fonksiyonunu optimizasyon parametresi ϕ ve $\mathcal{S}(\phi)$ üzerindeki sınıflandırıcı ağırlıklarının uyarlanırlar olarak seçme bağlamında minimize edecek iki yönlü optimizasyon sistemini kurmaktır.

III. REKABET SINIFININ OLUŞTURULMASI

ϕ parametresi ile optimize edilebilecek temel sınıflandırıcılar kümesini oluşturmak için öznitelik uzayını ϕ parametre vektörüne göre hiyerarşik olarak bölümlendiriyoruz. Bu işlemi detaylandırarak olursak, öncelikle öznitelik uzayını ϕ 'nin bir fonksiyonu olan ayırıcı bir fonksiyon ile ikiye bölüyoruz. Daha sonra oluşan bu bölümleri başka ayırıcı fonksiyonlar kullanarak tekrar ikiye bölerek ilerliyoruz ve bu şekilde sistemimiz için bölümlendirme ağacı gibi tamamlanmış bir hiyerarşik model elde ediyoruz. Bu doğrultuda ağacın her alt düğümü için düğümün temsil ettiği bölgeyi ikiye ayıran bir ayırıcı fonksiyon atamış oluyoruz. Ayrıca tanımlanan her düğüm için basit bir bölgesel sınıflandırıcı da oluşturuyoruz, örn. algılayıcı gibi doğrusal ve çevrimiçi bir sınıflandırıcı. Bu yapıyı örneklendirecek olursak, derinliği 2 olan, $f_{t,n}$ 'nin bir bölgesel sınıflandırıcısı ve $s_{t,n}$ 'nin düğüm n için bir ayırıcı fonksiyonu temsil ettiği bir ağaç yapısı düşünüldüğünde, ana düğümü (ya da düğüm λ) bütün öznitelik uzayını temsil edecek şekilde ve ayırıcı fonksiyon $s_{t,\lambda}$ 'yi de bu bölgeyi ikiye ayıracak ve düğüm 0 ile düğüm 1 oluşturacak fonksiyon şeklinde tanımlayabiliriz. Benzer şekilde, bu alt düğümler de $s_{t,0}$ ve $s_{t,1}$ ayırıcı fonksiyonları da ikiye ayrılarak sırasıyla 00, 01 ile 10, 11 alt düğümlerini oluşturacaktır.

Şunu söylememiz gerekir ki bölgesel sınıflandırıcıların ve ayırıcı fonksiyonların seçimi bütünüyle tercihe bağlıdır. Fakat biz bu bildiride düğümlerdeki sınıflandırıcılar olarak algılayıcıları ve ayırıcı fonksiyonlar olarak da hiperdüzlemleri kullanacağız. Detaylandırarak olursak, ayırıcı fonksiyon $s_{t,n}$, sigmoid fonksiyon $s_{t,n}(x_t) = (1 + \exp(\phi_{t,n}^T x_t))^{-1}$ şeklinde $\phi_{t,n}$ parametresinin bir fonksiyonudur. Burada $\phi_{t,n}$ düğümün ayırıcı hiperdüzlemine dik olan doğrunun açısını temsil etmektedir. Bu doğrultuda temel sınıflandırıcılar kümesinin parametrisasyonu parametre vektörü $\phi = \{\phi_{t,n}\}$ ile yapılmaktadır. Tanımlanan ayırıcı fonksiyonlara göre de x_t vektörü ana düğümden yaparak düğümlere belirli bir dalı izleyerek yol alır. Öyle ki, eğer düğüm n 'de $\phi_{t,n}^T x_t \leq 0$ olursa düğüm 1'e doğru, aksi takdirde düğüm 0'a doğru yol alır. Bu sırada geline her düğümden de bölge sınıflandırıcısı $f_{t,n}$ tarafından sınıflandırılır.

Üst üste gelmeyen bölgelerin birleşimini alarak farklı temel sınıflandırıcılar oluşturabiliriz. Örneğin, düğüm 0 ve 1 birlikte bir temel sınıflandırıcı oluşturabilir. Benzer şekilde, düğüm 00, 01 ve 1 farklı bir temel sınıflandırıcı oluşturabilir. Bu doğrultuda oluşturulan temel sınıflandırıcı $C \in \mathcal{S}(\phi)$, x_t oluşumunu bölgesel sınıflandırıcı $f_{t,n}(x_t)$ çıktısını kullanarak sınıflandırabilir. Burada n , x_t oluşumunu içeren C 'yi oluşturan alt ağacın bir yaprak düğümüdür. D derinliğinde bir ağacı oluşturan

yaklaşık 1.5^{2^D} alt ağaç olduğu için ve x_t sınıflandırmak için D derinliğindeki ağacın herhangi bir alt ağacı kullanılabildiği için her alt ağacı bir temel sınıflandırıcı olarak değerlendirmiyoruz [5]. Şunu da belirtmemiz gerekir ki her ayırıcı fonksiyon hassas bir şekilde öznitelik uzayını bölümlendirdiği için elde edilen temel sınıflandırıcılar fazlasıyla doğrusal olmayan modellerdir.

IV. UYARLANIR UZAY BÖLÜMLEME İLE ÇEVİRİMİÇİ SINIFLANDIRMA ALGORİTMASI (UUCS)

Şimdi yukarıda tanımladığımız öznitelik uzayı bölümlendirme sistemine göre $\mathcal{S}(\phi)$ 'deki bütün temel sınıflandırıcıların çıktısını birleştirerek nihai sınıflandırıcı olan f_t 'yi oluşturabiliriz. Bu doğrultuda veri uzunluğu T sonsuza giderken eşitlik (2)'de tanımlanan zarar sıfıra gidecek ve sınıflandırıcı f_t en iyi temel sınıflandırıcının performansına ulaşmış olacaktır. Sunduğumuz algoritma temel sınıflandırıcıların ağırlıklı birleşimini oluştururken aynı zamanda öznitelik uzayının bölümlendirilmesini ϕ 'yi güncelleyerek uyarlayıp sınıflandırma hatasını minimize edecektir. Aşağıda algoritmanın yapısı ile birlikte algoritmanın beklenen hatası için üst sınır belirleyen teoriyi sunuyoruz.

Teorem 1: $\{x_t\}_{t \geq 1}$ ve $\{y_t\}_{t \geq 1}$ 'nin rastgele ve gerçek bir dizinin sırasıyla öznitelik vektörü ve onun etiketi olduğunu kabul edecek olursak, Alg.1'deki algoritma bu herhangi T uzunluğundaki veri dizinine uygulandığında $O(Dp)$ hesaplama karmaşıklığı ile

$$\max_{C \in \mathcal{S}(\phi)} E [R_T(f_t; f_t^{(C)})] \leq O\left(\frac{2^D}{T}\right), \quad (3)$$

hata üst sınırını verir. Burada p öznitelik vektörlerinin boyutunu temsil eder ve beklenti operatörü de rastgeleleştirme parametresine göre çalışmaktadır.

Algoritmanın Yapısı

Gösterim: Düğümleri etkili bir şekilde belirtebilmek için burada tanımlanan gösterimi kullanacağız. Ağacın her bir düğümü ikili dizgilerle etiketlenmiştir. Örneğin $m_i = \{0, 1\}$ şeklinde ikili bir harf olarak ve ağacının derinliğini de d ile tanımlayacak olursak, düğüm n 'nin etiketi $n = m_1 \dots m_d$ şeklinde olacaktır. Bu şekilde düğüm n 'nin sağ ve sol alt düğümleri sırasıyla $n0$ ve $n1$ şeklinde tanımlanabilir. Ayrıca ana düğüm ya da boş dizgiyi λ şeklinde ifade ediyoruz ve de bütün $i = 1, \dots, d'$ için $d' \leq d$ ve $m'_i = m_i$ olacak şekilde düğüm $n' = m'_1 \dots m'_{d'}$ 'yi düğüm $n = m_1 \dots m_d$ 'nin örnek düğümü olarak tanımlıyoruz.

Temel Sınıflandırıcıların Oluşturulması: x_t oluşumunun düğüm n ile temsil edilen bir bölüme düştüğünü kabul edecek olursak, bu x_t oluşumunun ayrıca $n_0, \dots, n_D$ düğümlerine de düştüğünü gösterir. Burada $n_D = n'$ 'yi ve $n_0 = \lambda$ 'yi temsil etmektedir. Genelliği kaybetmeden, düğüm n_d 'nin temel sınıflandırıcı C 'yi oluşturan alt ağacın bir yaprak düğümü olduğunu kabul edecek olursak $f_t^{(C)}(x_t) = f_{t,n_d}(x_t)$ şeklinde daha önceki çalışmalarda da yapıldığı gibi belirleyebiliriz [6], [7]. Temel sınıflandırıcının oluşumu doğru olarak sınıflandırdığı kabul edilen bu tarz çalışmalarda her oluşum doğrudan bir düğüme atanır. Fakat, biz bu bildiride öznitelik uzayının bölümlendirilmesi işleme daha başlanmadan belirlendiği için atanan düğümdeki sınıflandırıcının oluşumu

doğru sınıflandırmayabileceğini de kabul ediyoruz. Bu sebeple de uyarlanırlar bir şekilde özneliteklilik uzayını yeniden yapılandırılabilir için her gelen oluşumu düğümlere belirli ağırlıklar (ya da olasılıklar) vererek atıyoruz. Bunun için de düğüm n_d ile düğüm λ arasında oluşan yolun ağırlıklarını değerlendirmek için 'güvenilirlik oranı' diye adlandırdığımız bir parametre tanımlıyoruz. Bu güvenilirlik oranı

$$c_{t,n_d}(\mathbf{x}_t) \triangleq \prod_{i=0}^{d-1} s_{t,n_i,m_{i+1}}(\mathbf{x}_t), \quad (4)$$

şeklinde tanımlanmıştır. Burada $s_{t,n_i,m_{i+1}}(\cdot)$ elemanı m_{i+1} yönündeki n_i düğüme karşılık gelen bölümlendirme fonksiyonunun değerini temsil etmektedir. örn.

$$s_{t,n_i,m_{i+1}}(\mathbf{x}_t) \triangleq \begin{cases} s_{t,n_i}(\mathbf{x}_t) & , \text{ eğer } m_{i+1} = 0 \\ 1 - s_{t,n_i}(\mathbf{x}_t) & , \text{ eğer } m_{i+1} = 1 \end{cases}$$

Burada basit bir şekilde özneliteklilik vektörü bölge sınırlarına yakın olduğu zaman güvenilirlik oranının düşük olacağını öngörebiliriz (örn. yaklaşık $(0.5)^d$). Bu sebeple bu özneliteklilik vektörünü başka bir düğümden sınıflandırıcı ile (mesela kardeş düğümün sınıflandırıcısı ile) sınıflandırmayı düşünebiliriz. Bunu baz alarak düğüm n_d 'nin sınıflandırma çıktısının $c_{t,n_d}(\mathbf{x}_t)$ olasılıkla doğru olduğunu kabul edip ve sınıflandırıcı f_{t,n_d} 'yi bir hata payı ile birlikte tanımladığımızda bunu bütünleyen etiketi de $-f_{t,n_d}(\mathbf{x}_t)$ şeklinde gösterebilir ve çıktısının da $1 - c_{t,n_d}(\mathbf{x}_t)$ olasılıkla doğru olacağını tanımlayabiliriz. Yani n_d düğümünün nihai çıktısı $\{c_{t,n_d}(\mathbf{x}_t), 1 - c_{t,n_d}(\mathbf{x}_t)\}$ olasılıklarıyla sırasıyla $\{f_{t,n_d}(\mathbf{x}_t), -f_{t,n_d}(\mathbf{x}_t)\}$ etiketlerinden biri olacaktır. Düğüm için sınıflandırıcıyı $f_{t,n_d}(\mathbf{x}_t)$ şeklinde tekrar tanımlayacak olursak, temel sınıflandırıcının çıktısını da $f_t^{(C)}(\mathbf{x}_t) = f_{t,n_d}(\mathbf{x}_t)$ şeklinde ifade edebiliriz. Bu yöntem sayesinde bize etkili bir şekilde özneliteklilik uzayını bölümlendirmeyi öğrenmemizde yardımcı olacak temel sınıflandırıcıların serbestlik derecesini ciddi şekilde arttırmış olduk.

Temel Sınıflandırıcıların Doğrudan Birleşimi: Tüm temel sınıflandırıcıları oluşturanın ardından en iyi temel sınıflandırıcının performansına yaklaşmak için uzman birleşimi yaklaşımını kullanacağız. Bu yöntemi sunmadan, öncelikle bazı gösterimler tanımlayacağız. t anında, sınıflandırıcı f_t için anlık gözleme dayalı zararın beklentisini $\ell_t(f_t) \triangleq E[\mathbb{1}_{\{f_t(\mathbf{x}_t) \neq y_t\}}]$ şeklinde, beklentinin sadece sınıflandırıcı f_t 'nin rastgeleleştirme parametrelerine göre alınmak üzere tanımlayacak olursak, T uzunluğundaki bir dizgi için bu sınıflandırıcının gözleme dayalı toplam zararının beklentisi $L_T(f_t) = \sum_{t=1}^T \ell_t(f_t)$ şeklinde olacaktır.

Ayrıca t anında düğüm n 'nin etkili olan bölgesini de $\mathcal{R}_{t,n_d} \triangleq \{\mathbf{x} : P_{t,n_d}(\mathbf{x}) \geq (0.5)^d\}$ şeklinde tanımlayacak olursak, tanımlanan temel sınıflandırıcı yapısına göre n_d düğümü $\mathbf{x}_t$ oluşumunu sadece $\mathbf{x}_t \in \mathcal{R}_{t,n_d}$ olursa sınıflandıracaktır. Bu sebeple veri akışı sırasında herhangi bir n düğümü için zamanla biriken gözleme dayalı zarar beklentisi

$$L_{T,n} \triangleq \sum_{t \leq T: \mathbf{x}_t \in \mathcal{R}_{t,n}} \ell_t(f_{t,n}). \quad (5)$$

şeklinde ifade edilebilir. Aynı şekilde temel sınıflandırıcı $C \in \mathcal{S}(\phi)$ için de zamanla biriken gözleme dayalı zarar beklentisi $L_T^{(C)} \triangleq \sum_{n \in \mathcal{L}(C)} L_{T,n}$ şeklinde bulunur. Burada $\mathcal{L}(C)$, C 'yi oluşturan alt ağacın yaprak düğümlerinin kümesini temsil etmektedir. Tanımlanan bu gösterimlerle birlikte, uzman birleşimi yaklaşımının doğrudan uygulanışı şu

Algoritma 1 Çevrimiçi Uyarlanırlar Uzay Bölümlendirme Sınıflandırıcısı (UÜCS)

- 1: **for** $t \geq 1$ **do**
 - 2: $\mathbf{x}_t$ 'yi ana düğümden yaprak düğümlere ilerlet ve gidilen $n_0, \dots, n_D$ düğümlerini elde et.
 - 3: Bütün $d \in 0, \dots, D$ için (4)'ü kullanarak $c_{t,n_d}(\mathbf{x}_t)$ hesapla.
 - 4: Bütün $d \in 0, \dots, D$ için (9)'u kullanarak $w_{t,n_d}(\mathbf{x}_t)$ hesapla.
 - 5: $f_{t,n_d}(\mathbf{x}_t)$ bulmak için $\{1, -1\}$ sınıflandırma çıktısından birini sırasıyla $c_{t,n_d}(\mathbf{x}_t)$ ve $1 - c_{t,n_d}(\mathbf{x}_t)$ olasılıklarıyla belirle.
 - 6: Düğüm çıktıları $f_{t,n_d}(\mathbf{x}_t)$ 'ni $w_{t,n_0}, \dots, w_{t,n_D}$ ağırlıklarıyla birleştir ve bu birleşime göre nihai çıktıyı rastgele belirle.
 - 7: Ziyaret edilen düğümlerde bölgesel sınıflandırıcıları (algılayıcıları) güncelle [8].
 - 8: $\ell_t(f_t) \leftarrow \mathbb{1}_{\{f_t(\mathbf{x}_t) \neq y_t\}}$
 - 9: Bütün $d \in 0, \dots, D$ için (5)'i kullanarak L_{t,n_d} güncelle.
 - 10: Bütün $d \in 0, \dots, D$ için M_{t+1,n_d} güncellemek için (7)'deki özyinelemeyi uygula.
 - 11: (10)'u kullanarak ayırıcı parametre ϕ 'leri güncelle.
 - 12: **end for**
-

şekilde olacaktır. Algoritmamızın nihai sınıflandırma çıktısı $f_t(\mathbf{x}_t) = \sum_{C \in \mathcal{S}(\phi)} w_t^{(C)} f_t^{(C)}$ şeklinde olacaktır. Burada $w_t^{(C)} = 2^{-J(C)} \exp(-b L_{t-1}^{(C)}) / Z_{t-1}$ şeklindedir ve $b \geq 0$ algoritmanın öğrenme oranını temsil ederken, $J(C) \leq 2|\mathcal{L}(C)| - 1$ sınıflandırıcı C 'yi kodlamak için gereken bit sayısını ($\sum_{C \in \mathcal{S}(\phi)} J(C) = 1$ koşulunu sağlamaktadır) ve $Z_t = \sum_{C \in \mathcal{S}(\phi)} 2^{-J(C)} \exp(-b L_t^{(C)})$ düzgeleme faktörünü temsil etmektedir. Şunu da belirtmeliyiz ki her ne kadar sınıflandırıcının nihai çıktısı $f_t(\mathbf{x}_t) \in [-1, 1]$ şeklinde belirlense de, nihai çıktı $\{(1 + f_t(\mathbf{x}_t))/2, 1 - f_t(\mathbf{x}_t)/2\}$ olasılıklarla sırasıyla $\{1, -1\}$ değerlerine aynı zarar beklentisi sağlanacak şekilde atanabilir.

Etkili Birleştirme Yöntemi: Yukarıdaki birleştirme algoritması kullanarak eşitlik (3)'teki teorik üst sınır sağlansa da, bu algoritma $|\mathcal{S}(\phi)| \approx 1.5^{2^D}$ olduğu için $O(1.5^{2^D} p)$ kadar hesaplama karmaşıklığı gerektirecektir. Fakat $\mathbf{x}_t \in \mathcal{R}_{t,n_D}$ için geçerli olan olası bütün sınıflandırma kararları kümesi $\{f_t^{(C)}(\mathbf{x}_t)\}_{C \in \mathcal{S}(\phi)} = \{f_{t,n_d}(\mathbf{x}_t)\}_{0 \leq d \leq D}$ nicelik olarak $O(D)$ kadar küçüktür. Yani, $\mathbf{x}_t$ oluşumundan $f_t(\mathbf{x}_t)$ 'yi üretmek için $\mathcal{S}(\phi)$ 'teki bütün temel sınıflandırıcıları hesaplamak gereksizdir. Aslında $f_t(\mathbf{x}_t)$ hesaplamak için gereken hesaplama karmaşıklığı $O(1.5^{2^D} p)$ değerinden $O(Dp)$ değerine f_{t,n_d} 'ler üzerinde aynı birleşim kullanılarak sadece ağırlıkların değişimi ile düşürülebilir. Yeni ağırlık atamaları basitçe

$$w_{t,n_d} = \sum_{C \in \mathcal{S}(\phi) : f_t^{(C)}(\mathbf{x}_t) = f_{t,n_d}(\mathbf{x}_t)} w_t^{(C)}. \quad (6)$$

şeklinde bulunabilir.

Eşitlik (6)'yı hesap karmaşıklığı $O(Dp)$ olacak şekilde hesaplamak için evrensel kodlama sistemini göz önünde bulundurarak ve herhangi bir n düğümü için

$$M_{t,n} \triangleq \begin{cases} \exp(-b L_{t,n}) & , \text{ eğer } n \text{'in derinliği } D \\ \frac{1}{2} [M_{t,n_0} M_{t,n_1} + \exp(-b L_{t,n})] & , \text{ aksi takdirde} \end{cases} \quad (7)$$

Tablo I: Veriler üzerinde ortalama hata oranları. İlk satır düzelenmiş veriler için sonuçlar örn., her özellik doğrusal olarak $[-1, 1]$ aralığına eşleştirilmiş. İkinci satır $\mathbf{x}_t \leftarrow \frac{\mathbf{x}_t}{\max(\|\mathbf{x}_t\|, 1)}$ şeklinde kesilmiş veriler için sonuçlar.

Bu kısımda, algoritmamızın (UUCS) gözleme dayalı performansını [10]'de sunulan diğer modern birleştirme teknikleri çevrimiçi AdaBoost (CAB) ve GradientBoost (CGB) algoritmaları ile karşılaştıracaktır. Algoritmamız (UUCS) için öğrenme oranını $\eta = 0.05$ olarak ve ağaç derinliğini de 4 olarak belirledik. Karşılaştıracığımız algoritmalar için zayıf öğrenicilerde ve bizim algoritmamız için bölgesel sınıflandırıcılarda algılayıcı algoritmasını kullandık

Veriler (Büyüklik/Boyut)	CAB	CGB	UUCS
Kalp (270/13)	0.2396 0.2400	0.2328 0.2314	0.2009 0.2083
Göğüs Kanseri (683/10)	0.0544 0.0538	0.0571 0.0533	0.0465 0.0458
Diyabet (768/8)	0.3243 0.3258	0.3349 0.3335	0.2575 0.2728

tanımlayarak elde edebiliriz [9]. Burada $M_{t,\lambda} = Z_t$ olduğunu da belirtmeliyiz. Bu bağlamda, (7)'deki özylenelemeyi kullanarak birleşim ağırlıkları $w_t^{(C)}$ için paydaları elde ederiz. Pay değerlerini de etkili bir şekilde hesaplamak için ara parametreler şu şekilde tanımlanmıştır. n_d' düğüm n_d 'nin kardeş düğümü olacak şekilde belirtirsek, $\forall d \in \{0, \dots, D\}$ ve $\mathbf{x}_t \in \mathcal{R}_{t,n_D}$ olacak şekilde

$$\kappa_{t,n_d} \triangleq \begin{cases} \frac{1}{2} & , \text{ eğer } d = 0 \\ \frac{1}{2} M_{t-1,n_d'} \kappa_{t,n_{d-1}} & , \text{ eğer } 0 < d < D \\ M_{t-1,n_d'} \kappa_{t,n_{d-1}} & , \text{ eğer } d = D \end{cases} \quad (8)$$

ifadesini özylenelemeli olarak tanımlayabiliriz. Eşitlik (7) ve (8)'de tanımlanan ara parametreleri kullanarak

$$w_{t,n_d} = \frac{\kappa_{t,n_d} \exp(-b L_{t,n_d})}{M_{t,\lambda}}. \quad (9)$$

olduğunu gösterebiliriz. Bu şekilde algoritmanın nihai çıktısını hesaplama karmaşıklığı $O(D)$ olacak şekilde $f_t(\mathbf{x}_t) = \sum_{d=0}^D w_{t,n_d} f_{t,n_d}(\mathbf{x}_t)$ olarak hesaplarız.

Uzay Bölümlemesinin Öğrenilmesi: Yapılanlara ek olarak, nihai sınıflandırma hatasını azaltmak için yukarıda tanımlanan algoritmanın nihai çıktısını kullanarak ağacın bölüm sınırlarını güncelliyoruz. Bu bağlamda, ϕ' 'yi güncellemek için olasılıksal bayır inişi algoritmasını, $\forall d \in \{0, \dots, D-1\}$

$$\phi_{t+1,n_d} = \phi_{t,n_d} - (-1)^{m_{d+1}} \eta (x_t - f_t(\mathbf{x}_t)) \pi_{t,n_d} s_{t,n_d,m_{d+1}}(\mathbf{x}_t) \mathbf{x}_t. \quad (10)$$

şeklinde kullanıyoruz. Burada η algoritmanın öğrenme oranını belirtmekte, m_{d+1}' ise ikili sistemde m_{d+1} elemanının tamamlayıcı harfini temsil etmektedir. Ayrıca

$$\pi_{t,n_d} \triangleq \begin{cases} f_{t,n_d}(\mathbf{x}_t) & , \text{ eğer } d = D-1 \\ \pi_{t,n_{d+1}} + f_{t,n_d}(\mathbf{x}_t) & , \text{ eğer } d < D-1 \end{cases}$$

ifadesi ise eşitlik (10)'deki güncellenmenin gerçekleştirilmesi için gerekli hesaplama karmaşıklığı $O(p)$ olan ara parametredir. Bu şekilde toplam algoritmanın toplam hesaplama karmaşıklığı $O(Dp)$ olmuştur.

Böylelikle sunduğumuz algoritma (UUCS) tamamlanmış olup, algoritmanın yalancı kodu Algoritma 1'de görülebilir.

V. PERFORMANS DEĞERLENDİRMESİ

[8]. Bu algoritmaları [10]'de sunulan bazı makine öğrenmesi verileri üzerinde test ettik. Her algoritmaya veriler sırasıyla verilerek tüm veri için hata oranları hesaplandı. Bu işlemi verilerin 100 adet rastgele permütasyonu için deneyerek ortalama hata oranlarını Tablo 1'deki gibi gösterdik.

Burada da görüleceği üzere, burada sunduğumuz algoritma modern birleştirme algoritmalarına göre daha üstün performans sergilemektedir. Bunun sebebi ise sunulan algoritma fazlasıyla doğrusal olmayan veriler üzerinde diğerlerine göre daha iyi sonuçlar elde etmesidir.

VI. SONUÇ

Bu bildiride büyük veri uygulamaları ve hesaplama karmaşıklığı açısından oldukça etkili olan bir çevrimiçi denetimli öğrenme algoritması sunduk. Sunulan bu yöntem yerel bölgelerinde tanımlanan basit doğrusal sınıflandırıcıların çıktılarını birleştirerek nihai kararını vermektedir. Burada kullandığımız yaklaşım aynı anda hem bölümlendirme yapısını hem de bu bölümlere karşılık gelen doğrusal modelleri optimize etmektedir. Ortaya çıkan bu oldukça dinamik hiyerarşik yapı sayesinde verilen herhangi bir uzunluktaki veri akışı için biriken hatayı minimize edebiliyoruz. Algoritmanın modern tekniklere olan bu konudaki üstünlüğünü birçok veri kümesi üzerinde deneyerek de gösterdik.

KAYNAKÇA

- [1] O. Bousquet and L. Bottou, "The tradeoffs of large scale learning," in *Advances in Neural Information Processing Systems*, 2008, pp. 161–168.
- [2] T. Mohamadpoor and B. Pfister, "A boosting framework on grounds of online learning," in *Advances in Neural Information Processing Systems*, 2014, pp. 2267–2275.
- [3] Joseph Wang and Venkatesh Saligrama, "Local supervised learning through space partitioning," in *Advances in Neural Information Processing Systems (NIPS)*, pp. 91–99, 2012.
- [4] N. D. Vanli and S. S. Kozat, "A comprehensive approach to universal piecewise nonlinear regression based on trees," *IEEE Transactions on Signal Processing*, vol. 62, no. 20, pp. 5471–5486, Oct 2014.
- [5] A. V. Aho and N. J. A. Sloane, "Some doubly exponential sequences," *Fibonacci Quarterly*, vol. 11, pp. 429–437, 1970.
- [6] 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.
- [7] Wei-Yin Loh, "Classification and regression trees," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 1, no. 1, pp. 14–23, 2011.
- [8] Yoav Freund and Robert E Schapire, "Large margin classification using the perceptron algorithm," *Machine learning*, vol. 37, no. 3, pp. 277–296, 1999.
- [9] F. M. J. Willems, Y. M. Shtarkov, and T. J. Tjalkens, "The context-tree weighting method: basic properties," *IEEE Transactions on Information Theory*, vol. 41, no. 3, pp. 653–664, May 1995.
- [10] Shang-Tse Chen, Hsuan-Tien Lin, and Chi-Jen Lu, "An online boosting algorithm with theoretical justifications," *International Conference on Machine Learning*, 2012.
- [11] Qinxun Bai, Henry Lam, and Stan Sclaroff, "A bayesian framework for online classifier ensemble," in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 2014, pp. 1584–1592, JMLR Workshop and Conference Proceedings.
- [12] N. C. Oza and S. Russell, "Online bagging and boosting," in *Artificial Intelligence and Statistics*, 2001, pp. 105–112.
- [13] Yoav Freund and Robert E Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in *Computational learning theory*. Springer, 1995, pp. 23–37.
- [14] Christian Leistner, Amir Saffari, Peter M Roth, and Horst Bischof, "On robustness of on-line boosting-a competitive study," in *IEEE 12th International Conference on Computer Vision Workshops (ICCV Workshops)*, 2009, pp. 1362–1369.