SETSCI conference proceedings
SETSCI Conference Proceedings
6 (1), 27-32, 2023

Cognitive Models and Artificial Intelligence Conference
26-28 October 2023, Ankara, Türkiye

https://doi.org/10.36287/setsci.6.1.013
2687-5527/

BMYZ 2023

İleri Beslemeli Yapay Sinir Ağları için Donanım Uygulamasına Yatkın Çevrimiçi ELM Tabanlı Eğitim Algoritması

Önder Polat[1*], Sema Koç Kayhan [1]

[1]*Elektrik-Elektronik Mühendisliği, Gaziantep Üniversitesi, Gaziantep, Türkiye*
[*]*(onderp@gantep.edu.tr)*

Özet – Bu bildiride, ileri beslemeli tek gizli katmanlı yapay sinir ağları için donanım uygulamasına uyumlu olan ve seyrek düzenlileştirilmiş bir aşırı öğrenme makinası (ELM) algoritması sunulmaktadır. Önerilen algoritma, düzenlileştirme için genelleştirilmiş minimaks konkav (GMC) ceza fonksiyonuna dayanmakta ve matris tersi alma veya norm hesaplaması gibi donanım mimarisi için karmaşıklığı arttıracak matematiksel işlemleri içermemektedir. Önerilen algoritma, çevrimiçi sıralı ELM (OS-ELM) algoritmasına benzer olarak, çevrimiçi öğrenmeyi destekleyecek şekilde değiştirilmiştir. Önerilen algoritma, UCI makine öğrenmesi veri deposundan alınan veri kümeleri ile test edilmiş ve genel olarak daha yüksek seyreklik seviyelerine ulaşmıştır. OS-ELM algoritmasına göre genellikle daha iyi sonuçlar vermiştir. Ek olarak, düzenlileştirme ile artan seyreklik seviyeleri sayesinde, önerilen algoritma yapay sinir ağından gereksiz değişkenleri elemekte ve test verileri için tahmin doğruluğunu artırmaktadır.

Anahtar Kelimeler – Yapay Sinir Ağı, Çevrimiçi makine öğrenmesi, ELM, Donanım uygulaması

I. GİRİŞ

Tek gizli katmanlı ileri beslemeli ağlar (SLFN'ler), kompakt yapıları ve iyi tahmin yetenekleri nedeniyle literatürde geniş bir şekilde incelenmiştir. İleri beslemeli yapay sinir ağları (FNN'ler) genellikle geriye yayılım (BP) yöntemleri kullanılarak eğitilmektedir. Ancak, bu yöntemler eğitim için gereken geriye yayılım adımı nedeniyle daha yavaş olma eğilimindedir. Aşırı Öğrenme Makinası (ELM), analitik olarak SLFN eğitimi için kullanılan çok hızlı bir algoritmadır [1], [2]. ELM algoritmasının hızlı eğitim özelliği, ağırlık ve sapma değerlerinin rastgele başlatılmasından kaynaklanır ve çıkış katmanı için analitik olarak çözülerek, gerekli ağırlık ve sapma değerleri bulunur. ELM algoritması ile bu rastgele atanan değerlerin geriye yayılım gibi tekrarlayan işlemlere ihtiyaç duyulmadan ağı eğitmek için kullanılabileceği kanıtlanmıştır [3]. ELM literatürdeki benzer öğrenme algoritmaları ile de kıyaslanabilir bir performans sergilemektedir [3].

Pratik uygulamalara bakıldığında, yapay sinir ağına ekstra yük getiren ve performansını düşüren gereksiz değişkenler içeren veri setleri görülmektedir. Bu yapay sinir ağı için gereksiz karmaşıklığa neden olabilmekte ve eğitilen ağda aşırı öğrenme oluşabilmektedir. Ek olarak, genelleme performansını kötüleştirebileceğinden ve tahmin hızını düşüreceğinden, eğitilen model aşırı derecede karmaşık da olmamalıdır. Bu sorunu aşmak için literatürde birçok çalışma önerilmiştir. Bu yöntemler genel olarak 2 ana gruba ayrılabilir: 1) Bilgi Kriterleri (IC) temelli budama,ve 2) Düzenlileştirme temelli yöntemler. Budama temelli yöntemler, önceden belirlenmiş belirli bir IC'ye dayanarak gizli katman nöronlarının bazılarını kaldırmaktadır. Düzenlileştirme temelli yöntemler, zarar fonksiyonuna bir ceza fonksiyonu tanıtarak bu problemi bir optimizasyon problemi olarak formüle etmeye ve optimizasyon yöntemlerini kullanarak çözmeye dayanır. Bunlar, ağırlıkları normalleştirerek ve aşırı öğrenmeden kaçınmak için ağırlıkları düzenlileştirerek çıkış parametrelerini düzenleyen bir dizi tekniktir. Ceza fonksiyonun türüne bağlı olarak çıkış ağırlık parametrelerinin çoğu sıfır değerini alabilir ve seyrek bir yapıya ulaşabilir.

Geleneksel ELM, eğitim süreci başlamadan önce tüm eğitim verilerinin mevcut olduğunu varsaymaktadır. Ancak, bazı pratik uygulamalarda eğitim verileri sıralı olarak gelebilmektedir. Bu durumda tüm veriler önceden toplu olarak hazır bulunmayabilir. Bu durum, yapay sinir ağını her yeni veri örneği için ek kaynaklar (örneğin bellek, hesaplama gücü vb.) kullanmadan eğiten çevrimiçi makine öğrenme algoritmalarını gerektirmektedir. Ayrıca, bu algoritmaların bu dinamik eğitim sürecini etkili bir şekilde yönetmesi ve ilgili uygulamanın gerçek zamanlı gereksinimlerini karşılaması gerekmektedir. Bu tür sistemlerin avantajı; kaynak kullanımının eğitim başlamadan önce bilinmesi ve sabit olması, ayrıca sisteme yeni bir veri örneği geldiğinde, ağı eğitmeye devam ederken sınıflandırma sonuçlarını da sunabilmesidir. Böyle bir sistemin hesaplama gereksinimi yüksek olacağından, özelleştirilmiş donanım geliştirilmesi ve paralel hesaplama yapılması gibi konular bir gereksinim haline gelmektedir.

Bu nedenle, bu bildiri kapsamında; donanım uygulamasına da yatkın olan bir çevrimiçi makine öğrenme algoritması geliştirilmesi hedeflenmiştir. Bu amaçla, ELM algoritması tabanlı bir seyrek çevrimiçi öğrenme algoritması önerilmiştir. Önerilen algoritma, konveks olmayan bir ceza fonksiyonu ile optimizasyon yaparak, çıkış katmanındaki ağırlıkları düzenlileştirmekte ve seyrekleştirmektedir. Ayrıca algoritma modifiye edilerek sıralı verilerle çevrimiçi öğrenmeye uygun hale getirilmiştir. Önerilen algoritma, yalnız gizli katman matrisini ve etiket vektörünü kullanmakta ve donanım

uygulamasına çok uygun olmayan matris tersi alma işlemi gibi işlemler içermemektedir. Bu özellikleriyle de donanım uygulamasına yatkın bir algoritmadır. Önerilen algoritmanın test verileri için doğruluk oranları da incelenmiş ve klasik ELM algoritmalarına göre yeterli seviyede sonuç verdiği görülmüştür.

Bildirinin geri kalanı aşağıdaki şekilde düzenlenmiştir: Bölüm 2'de mevcut literatürde bulunan önemli çalışmalar açıklanmıştır. Bölüm 3'te, önerilen seyrek ELM algoritması sunulmuştur. Bölüm 4'te önerilen algoritmanın çeşitli veri kümeleri üzerindeki performansı değerlendirilmiştir. Bildiri, Bölüm 5'te sonuçlanmaktadır.

II. MEVCUT LİTERATÜR

Bu bölümde, literatürde mevcut önemli çalışmalar sunulacaktır. ELM algoritmasının regresyon ve sınıflandırma için önerilen çeşitli uzantıları bulunmaktadır. Bu bildiri kapsamında, ELM ağının düzenlileştirilmesi ve budanması ile ilgili çalışmalar ana odak noktası olmuştur. ELM algoritması gizli katmanı rastgele oluşturduğu için geleneksel sinir ağlarına göre daha fazla gizli nöron gerektirmektedir. Artan gizli katman nöron sayısı performansı kısıtlamakta, aşırı öğrenmeye sebebiyet vermekte ve sınıflandırma süresi uzamaktadır. Bu nedenle, ağın karmaşıklığını azaltma konusu mevcut literatürde büyük ilgi çekmektedir.

A. Budama temelli yöntemler

[4]'te sunulan çalışma, optimize edilmiş budama ELM (OP-ELM) olarak adlandırılmıştır. OP-ELM, mevcut gizli nöronları en küçük açı regresyonu (LARS) algoritması temel alarak sıralamaktadır. LARS, l_1 düzenlileştirmesi yapan bir algoritmadır [5]. Sıralama işleminden sonra nöronlar çapraz doğrulama (CV) ile budanmaktadır. Özellikle büyük veri kümeleri için çapraz doğrulama (CV) süreci zaman alıcı olduğundan, [4]'ün yazarları, her veri örneği için CV olarak tanımlanabilecek LOO-CV'nin kapalı form bir sürümünü kullanmakta ve buna tahmin karelerin toplamı (PRESS) demektelerdir. Her nöronu LOO-CV sürecine dahil ederek, en düşük hataya sahip sayıyı seçerek bunu en iyi gizli nöron sayısı olarak kullanmaktalardır. Sunulan yöntem küçük veri kümeleri için etkilidir, ancak LOO-CV adımı nedeniyle büyük veri kümeleri için eğitim süresi çok yavaştır. Başka bir budama tabanlı ELM de [6]'da önerilmiştir. Burada gizli nöronlar üç farklı IC'ye dayalı olarak budanmaktadır; Ki-kare, bilgi kazancı (IG) ve Akaike Bilgi Kriteri (AIC). İlk olarak, gizli katman matrisi H rastgele ağırlıklar ve eğitim verileri ile oluşturularak, temel ELM yapısı elde edilmektedir. Daha sonra, her gizli düğümün istatistiksel ilişkisi IC'lere dayalı olarak hesaplanmaktadır. Kullanıcı tarafından verilen bir ilişki eşiği, ağdaki önemli nöronları tanımlamak için kullanılmaktadır. Son adımda, aynı zamanda minimum AIC değerine sahip nöronları bulmak için doğrulama kullanılmaktadır. Son olarak, seçilen nöronlar ağı son kez eğitmek için kullanılmaktadır. P-ELM test verileri için iyi isabet oranlarına ulaşmaktadır, ancak diğer ELM uzantılarına göre ağın eğitim süresi hala yavaştır. [7]'deki çalışmada yine PRESS formülü kullanılmıştır. Yazarlar LOO-CV hesaplaması için sıralı bir yöntem önermişlerdir. Her gizli nöron ağa sırayla eklenmekte ve LOO-CV formülü sırayla güncellenmektedir. En iyi nöron sayısı, yazarlar tarafından önerilen bir erken durdurma kriterini kullanarak belirlenir. Veri kümeleri için daha iyi olan verimli LOO-CV tabanlı ELM (ELOO-ELM), OP-ELM ve P-ELM algoritmalarından daha iyi sonuç vermektedir, ancak büyük veri kümeleri için hala yavaştır.

Yukarıda bahsedilen ve temelde CV tabanlı olan bu yöntemler, hem norm hesaplamaları için karmaşık işlemler içermeleri hem de CV nedeniyle bütün veri setinin baştan hazır olmasını istediklerinden, çevrimiçi uygulamalara ve donanım uygulamasına yatkın değildir.

B. Düzenlileştirmeye dayalı yöntemler

Seyrek yapay sinir ağı elde etmek için, optimizasyonda kullanılan farklı normlar arasında, en yaygın uygulanan seyrek düzenlileştirme yöntemleri şunlardır: l_0-norm ile seyrek düzenleme, l_1-norm ile seyrek düzenleme, l_2-norm ile seyrek düzenleme ve karışık norm minimizasyonu ($l_{2,1}$). Bir lineer denklemler sistemi $(y = Ax)$, optimizasyon kullanılarak seyrek bir şekilde çözülebilir. Genellikle, düzenlileştirilmiş en küçük kareler maliyet fonksiyonu, şu şekilde bir ceza fonksiyonu ile minimize edilmektedir:

$$\arg\min_x f(x) = \|y - Ax\|_2^2 + \lambda\psi(x) \quad \lambda > 0 \tag{1}$$

l_1-norm, genellikle makine öğrenimi sorunlarını çözmek için düzenlileştirme yapmak için yaygın olarak kullanılmaktadır. l_1-norm, genellikle aşağıdaki optimizasyon problemi tanımlanarak kullanılmaktadır:

$$\arg\min_x f(x) = \|y - Ax\|_2^2 + \lambda\|x\|_1 \quad \lambda > 0 \tag{2}$$

Başka bir olası yaklaşım ise Tikhonov düzenlileştirmesi (veya ridge regresyon) olarak adlandırılan yöntemi kullanmaktır. Bu yöntemde, aşağıdaki formülasyon kullanılarak l_2-norm minimizasyonu yapılmaktadır:

$$\arg\min_x f(x) = \|y - Ax\|_2^2 + \lambda\|x\|_2^2 \quad \lambda > 0 \tag{3}$$

l_2-norm minimizasyonu, katsayıları küçültür ve ağın karmaşıklığını azaltır, ancak seyreklik özellikleri, l_1-norm minimizasyonuna göre zayıftır. l_1-norma göre daha iyi sonuçlar vermektedir. l_2-norm, verideki aykırı değerlere karşı hassastır, çünkü ridge regresyon, artık değişkeninde biriken hataların karesini almaktadır. Öte yandan, l_1-norm, bu hata değerlerinin sadece mutlak değerini alır, bu nedenle bu tür aykırı değerlere karşı dayanıklıdır.

[8]'de sunulan çalışmada, OP-ELM algoritmasına dayanan l_2-norm temelli bir düzenleme önerilmiştir. OP-ELM'deki l_1-norm sıralama adımının yanı sıra, Tikhonov düzenlemeli OP-ELM (TROP-ELM) algoritması, l_2-norm kullanarak çıkış ağırlıklarını düzenlemekte ve hızlı matris hesaplaması için PRESS istatistiğinin değiştirilmiş bir versiyonunu kullanmaktadır. Yazarlar, regresyon veri kümeleri için karşılaştırmalı sonuçlarını sunmuştur. Hata oranları OP-ELM'den daha iyidir, ancak eğitim süreleri OP-ELM'den daha yavaştır. Ayrıca, sonuçlar yalnızca regresyon durumu için sunulmuş olup sınıflandırma görevi için herhangi bir bilgi verilmemiştir.

Literatürde çevrimiçi çalışan ve düzenlileştirme kullanan birçok algoritma da önerilmiştir[9]–[12]. Düzenlileştirme kullanan birçok çalışma olmasına karşın, çevrimiçi olanlar da dahil, donanım uygulamasına yatkın değillerdir. Matris tersi işlemi, norm hesaplaması, karekök vb. işlemler içeren bu algoritmalar, olası bir donanım tasarımı sürecinde donanımın karmaşıklığını oldukça arttıracak ve hesaplama performansını düşürecektir. Bu nedenle, donanıma yatkın algoritmalara ihtiyaç duyulmaktadır.

III. Önerilen Çevrimiçi ELM Algoritması

ELM algoritmasında, $T = H\beta$ şeklindeki sistemde çıkış ağırlık vektörü β, gizli katman matrisi H'nin yalancı-ters matrisi ile etiket matrisi T birlikte kullanılarak şu şekilde hesaplanmaktadır:

$$\beta = (H^T H)^{-1} H^T T \tag{4}$$

Gizli katman matrisi H, aktivasyon fonksiyonu $G(\mathrm{w}, \mathrm{x}, \mathrm{b})$ kullanılarak şu şekilde tanımlanabilir:

$$H(x) = \begin{bmatrix} G(w_1 \cdot x_1 + b_1) & \dots & G(w_L \cdot x_1 + b_L) \\ . & & . \\ . & & . \\ . & & . \\ G(w_1 \cdot x_N + b_1) & \dots & G(w_L \cdot x_N + b_L) \end{bmatrix}_{NxL} \tag{5}$$

ELM'deki gizli katman nöronları, eğitim verileri yapay sinir ağına sunulmadan önce rastgele oluşturulmaktadır. Temel ELM algoritması, eğitim sürecinden önce tüm eğitim verilerinin hemen kullanılabilir olduğunu varsayar. Bazı pratik uygulamalarda ise eğitim verileri sıralı olarak sunulmaktadır ve önceden mevcut değildir. Ayrıca, sıralı olarak gelecek verinin örnek sayısı olarak büyüklüğü bilinememektedir. Bu nedenle, yapay sinir ağı mimarisi hem algoritma kısmında teorik olarak hem de donanım kısmında pratik olarak bu koşulu desteklemelidir. Bu amaçla, ELM algoritmasının çevrimiçi sürümü olan OS-ELM, [16]' da önerilmiştir. OS-ELM, eğitim verilerini birer birer veya gruplar halinde öğrenebilmektedir ve öğrenme işleminden sonra verileri atmaktadır. OS-ELM algoritmasının, sisteme kaç eğitim örneğinin gireceğini önceden bilmesine gerek yoktur. OS-ELM algoritması, orijinal ELM algoritmasına dayanmaktadır ve onun özyinelemeli hesaplamayı destekleyen versiyonudur. Özetlemek gerekirse, OS-ELM sıralı şekilde gelen her veri grubu için aşağıdaki işlemleri yapar:

$$K_0 = H_0^T H_0 \tag{6}$$

$$\beta_0 = (H_0^T H_0)^{-1} H_0^T T_0 \tag{7}$$

$$K_{k+1} = K_k + H_{k+1}^T H_{k+1} \tag{8}$$

$$\beta_{k+1} = \beta_k + K_{k+1}^{-1} H_{k+1}^T (T_{k+1} - H_{k+1}\beta_k) \tag{9}$$

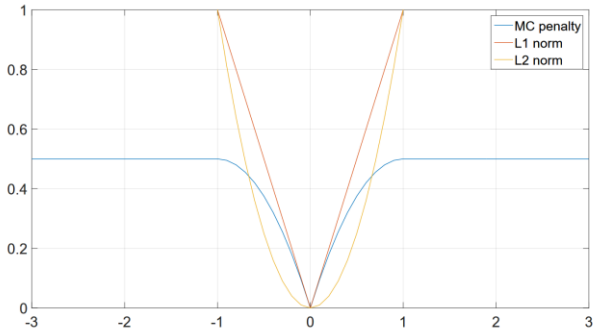
A. Önerilen seyrek düzenlileştirilmiş çevrimiçi ELM algoritması

l_1-norm kullanılarak $T = H\beta$ doğrusal sistemi için seyrek yaklaşık bir çözüm bulunabilir. l_1-norm, en seyrek çözümü sağladığı için genellikle düzenlileştirme için ilk tercih edilen seçenektir. Ancak, l_1-norm yüksek genlikli bileşenleri daha fazla baskıladığı için bu bileşenlerin sisteme katkılarını sınırlayabilmektedir [13]. l_1-norm'un sınırlamalarını aşmanın yollarından biri, düzenlileştirme için konveks olmayan ceza fonksiyonlarını kullanmaktır [14]. Bu tür bir fonksiyon, minimaks konkav (MC) ceza fonksiyonudur[15]:

$$\psi(x) = \begin{cases} |x| - \frac{1}{2}x^2, & |x| \le 1 \\ \frac{1}{2}, & |x| \ge 1 \end{cases} \tag{10}$$

l_1-norm, l_2-norm ve MC ceza fonksiyonlarını Şekil 1'de gösterilmektedir. MC ceza fonksiyonunun genelleştirilmiş hali olan (GMC), Huber fonksiyonu kullanılarak [16] nolu çalışmada önerilmiştir. b ölçeklendirme parametresi olmak üzere, ölçekli Huber fonksiyonu aşağıdaki gibidir:

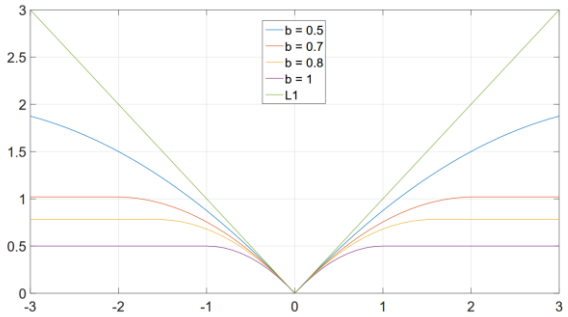
$$s_b(x) = \begin{cases} |x| - \frac{1}{2b^2}, & |x| \ge 1/b^2 \\ \frac{1}{2}b^2x^2, & |x| \le 1/b^2 \end{cases} \tag{11}$$



Şekil 1: MC, L1 ve L2 ceza fonksiyonları

GMC ise şu şekilde tanımlanmaktadır:

$$\psi_B(x) = \|x\|_1 - s_B(x) \tag{12}$$



Şekil 2: Farklı ölçek değerleri için GMC ceza fonksiyonu

Bu nedenle, GMC, x'in mutlak değerinden (l_1-norm) genelleştirilmiş Huber fonksiyonunu çıkartarak elde edilir. GMC ceza fonksiyonu, l_1-norm'un konveks özelliğini korurken seyrekliği güçlü bir şekilde teşvik etmektedir. GMC

ceza fonksiyonunun ölçekli versiyonu Şekil 2'de görselleştirilmiştir. Şekil 2'de GMC ceza fonksiyonunun küçük değerleri büyük değerlerden daha az baskıladığı gözlenmektedir, ancak bu baskılama l_1-norm kadar şiddetli değildir.

GMC ceza fonksiyonuyla ELM algoritması için optimizasyon problemi şu şekilde ifade edilebilir:

$$\arg\min_{\beta} g(\beta) = \|T - H\beta\|_2^2 + \lambda\psi_B(\beta) \quad (13)$$

Burada λ, düzenleme parametresidir. g(β) maliyet fonksiyonu olmak üzere, bu optimizasyon problemi ileri-geri yer değiştirme algoritması (FBS) kullanılarak çözülebilmektedir. FBS kullanarak çözebilmek için, optimizasyon problemi eyer noktası problemi olarak şu şekilde tanımlanmaktadır[16], [17]:

$$\arg\min_{x} g(\beta,\sigma) = \|T - H\beta\|_2^2 + \lambda\|\beta\|_1 - \lambda\|\sigma\|_1 - \frac{\gamma}{2}\|H(\beta-\sigma)\| \quad (14)$$

Algoritma 1 FBS Algoritması

Girdi:

ρ: $max\{1, \gamma/(1-\gamma)\}\|A^TA\|_2$
μ: $0 \le \mu \le 2/\rho$
γ: $0 \le \gamma \le 1$
λ: $\lambda > 0$
K: Maksimum yineleme sayısı.
A: $A \in R^{M\times N}$
y: $y \in R^M$

Çıktı:

x: $x \in R^N$

x ve v için başlangıç değerlerini sıfır yap.
for $i = 1$ to K **do**
  $w_i = x_i - \mu A^T(A(x_i + \gamma(v_i - x_i)) - y)$
  $u_i = v_i - \mu\gamma A^T A(v_i - x_i)$
  $x_{i+1} = soft(w_i, \mu\lambda)$
  $v_{i+1} = soft(u_i, \mu\lambda)$
end for

FBS algoritması yalnızca yumuşak eşikleme operatörünü ve A^TA ve A^Ty matrislerinin çarpımını içermektedir. Parametreler λ ve γ kontrol edilerek, (14) denklemi içindeki optimizasyon problemi için seyrek bir çözüme ulaşılabilir. GMC ceza fonksiyonunu kullanan FBS algoritması, ELM algoritması için çıkış ağırlık vektörü β'yı düzenlileştirmek için kullanılabilir. Yalnızca matrisler A, A^T ve y'nin çarpımını içerdiğinden ve matris tersi işlemi gerektirmediğinden, donanım uygulaması için oldukça uygun iyi bir algoritmadır.

FBS algoritmasını çevrimiçi öğrenmeyi destekleyecek şekilde modifiye etmek için, matrisler H^TH ve H^TT'nin özyinelemeli formülleri türetilmiştir. Başlangıç veri örnekleriyle, gizli katman matrisinin H_0 ve hedef matrisinin T_0 olduğu varsayılırsa, yeni bir eğitim veri örneği (x_1, t_1) geldiğinde, yeni gizli katman matrisi H_1 şu şekilde tanımlanır:

$$H_1 = \begin{bmatrix} H_0 \\ h_1 \end{bmatrix} \quad (15)$$

$$C = [H_0^T h_1^T] \begin{bmatrix} H_0 \\ h_1 \end{bmatrix} = [H_0^T H_0 + h_1^T h_1] \quad (16)$$

$$D = [H_0^T h_1^T] \begin{bmatrix} T_0 \\ t_1 \end{bmatrix} = [H_0^T T_0 + h_1^T t_1] \quad (17)$$

Yeni bir veri örneği (x_i, t_i) geldiğinde, matrisler C_1 ve D_1 için özyinelemeli formüller aşağıdaki gibi tanımlanabilir:

$$C_1 = C_0 + h_1^T h_1$$
$$D_1 = D_0 + h_i^T t_1$$

Bu özyinelemeli formülleri kullanarak önerilen, seyrek düzenlileştirilmiş çevrimiçi ELM algoritması (SDÇ-ELM), Algoritma 2'de verilmiştir.

Algoritma 2 SDÇ-ELM Algoritması

Girdi:

Parametreler: γ, λ, $0 \le \mu \le 2/\rho$, and L.
T_0: N_0xm boyutlu başlangıç etiket matrisi.
H_0: N_0xL boyutlu başlangıç gizli katman matrisi.
R: Maksimum yineleme sayısı.

Çıktı:

β: Lxm boyutlu çıkış ağırlık matrisi

β ve σ için başlangıç değerlerini sıfır yap.
Başlangıç Aşaması:
H_0 matrisini başlangıç veri örneği sayısı N_0 için hesapla.
for $i = 1$ to R **do**
  $\delta_i = \beta_{i-1}^{(0)} - \mu H_0^T(H_0(\beta_{i-1}^{(0)} + \gamma(\sigma_{i-1} - \beta_{i-1}^{(0)})) - T_0)$
  $\eta_i = \sigma_{i-1} - \mu\gamma H_0^{T_0} H_0(\sigma_{i-1} - \beta_{i-1}^{(0)})$
  $\beta_i^{(0)} = soft(\delta_i, \mu\lambda)$
  $\sigma_i = soft(\eta_i, \mu\lambda)$
end for
Sıralı Öğrenme Aşaması:
for $k = 1, 2, 3, \ldots$ **do**
  $C_k = C_{k-1} + h_k^T h_k$
  $D_k = D_{k-1} + h_k^T t_k$
  $\delta_k = \beta_{k-1} - \mu(C_k(\beta_{k-1} + \gamma(\sigma_{k-1} - \beta_{k-1})) - D_k)$
  $\eta_k = \sigma_{k-1} - \mu\gamma C_k(\sigma_{k-1} - \beta_{k-1})$
  $\beta_k = soft(\delta_k, \mu\lambda)$
  $\sigma_k = soft(\eta_k, \mu\lambda)$
end for
return β_k

IV. Deneysel Sonuçlar

Önerilen algoritmanın performansı, literatürde genellikle kıyaslama amacıyla kullanılan veri kümeleri ile test edilmiştir. Sınıflandırma problemleri için doğruluk oranı ve regresyon problemleri için kök ortalama kare hata (RMSE) ölçütleri baz alınmış ve UCI makine öğrenmesi veri deposundan sekiz veri seti kullanılmıştır[18]. Kullanılan veri kümeleri Tablo 2'de verilmiştir. Bu çalışmada amaç donanım mimarisine uygun bir algoritma önermek olduğundan, önerilen algoritma klasik ELM algoritmaları ile karşılaştırılarak yeterli performans verip vermediği ve seyreklik özelliği incelenmiştir.

Deneylerde tüm veri kümeleri için maksimum gizli nöron sayısı 100 olarak ayarlanmıştır. SDÇ-ELM için düzenlileştirme parametreleri manuel olarak ayarlanmış ve en

iyi parametreler seçilmiştir. OS-ELM ve SDÇ-ELM için sıralı eğitim adımında her yineleme için örnek kümesi büyüklüğü veri setinin yüzde 10'u olarak alınmıştır. Her veri kümesi için

Tablo 1. ELM, OS-ELM ve SDÇ-ELM için RMSE, doğruluk oranı ve eğitim süresi karşılaştırması

Veri Seti	Algoritma	Eğitim Süresi (s)	RMSE/Doğruluk Oranı	Nöron Sayısı
Breast Cancer	ELM	0.0040	0.9412	100
	OS-ELM	0.0085	0.9344	100
	SDÇ-ELM	0.0423	0.9373	74
Diabetes	ELM	0.0044	0.7359	100
	OS-ELM	0.0082	0.5379	100
	SDÇ-ELM	0.0443	0.7407	87
Iris	ELM	0.0029	0.5911	100
	OS-ELM	0.0056	0.4822	100
	SDÇ-ELM	0.0306	0.9244	40
Segments	ELM	0.0082	0.8960	100
	OS-ELM	0.0154	0.7831	100
	SDÇ-ELM	0.0872	0.8955	81
Abalone	ELM	0.0125	0.2658	100
	OS-ELM	0.0363	0.1006	100
	SDÇ-ELM	0.2466	0.1344	100
Body-fat	ELM	0.0023	0.0053	100
	OS-ELM	0.0054	0.0039	100
	SDÇ-ELM	0.2209	0.0046	45
Auto P.	ELM	0.0023	0.0529	100
	OS-ELM	0.0056	0.0104	100
	SDÇ-ELM	0.1347	0.0448	60
Boston H.	ELM	0.0039	0.0360	100
	OS-ELM	0.0075	0.0201	100
	SDÇ-ELM	0.1076	0.0289	64

50 deneme yapılmış ve 5 katlı CV için ortalama değerler rapor edilmiştir. Çevrimiçi algoritmalar için başlangıç örnek boyutu 100 örnek olarak ayarlanmıştır. Her denemede başlangıç ağırlık değerleri rastgele olarak belirlendiğinden, sabit bir tohum değeri belirlenerek her algoritma için aynı değerlerin kullanılması sağlanmıştır. Tablo 1'de eğitim süresi, test seti için tahmin performansı (doğruluk oranı ve RMSE) ve kullanılan nöron sayısı (seyreklik) sonuçları sunulmuştur. Tablo 1'den görüleceği üzere, önerilen SDÇ-ELM algoritması OS-ELM'ye göre genellikle daha iyi doğruluk oranlarına ulaşmıştır. Özelikle sınıflandırma veri setleri için daha yüksek sonuçlar sergilemiştir. Genel olarak değerlendirildiğinde ELM ve OS-ELM'ye yakın veya daha iyi performans vermiştir. ELM ve OS-ELM'nin performansı bazı veri setlerinde düşerken, önerilen algoritma her veri seti için stabil sonuçlar sunmuştur. Bunun en önemli nedeni düzenlileştirme yapma kabiliyetine sahip olmasıdır. SDÇ-ELM'nin eğitim süreleri, düzenlileştirme adımları nedeniyle daha uzun sürse de donanım mimarisine uygun bir algoritma olduğu için, olası bir donanım uygulamasında klasik ELM algoritmalarına göre daha hızlı çalışacaktır. Böyle bir donanım mimarisi matris ters alma işlemi gibi işlemler de içermeyeceğinden daha sade bir mimari elde edilecektir. Seyreklik bakımından incelendiğinde ise, SDÇ-ELM'nin daha seyrek yapıda bir yapay sinir ağı oluşturduğu görülmektedir.

V. SONUÇ

Sonuç olarak, önerilen çevrimiçi ELM algoritması donanım uygulamaları için uygun ve seyrek düzenlileştirme kabiliyetine sahip bir algoritmadır. Yapılan deneyler, önerilen algoritmanın test veri kümeleri üzerinde daha yüksek seyreklik seviyeleri ve genellikle daha iyi performans sağladığını göstermektedir. Seyreklik seviyelerinin artırılması, yapay sinir ağlarının gereksiz değişkenlerden arındırılmasına ve test verileri için tahmin doğruluğunun artırılmasına olanak tanımaktadır. Gelecekte, önerilen algoritmanın daha fazla veri kümesi ile ayrıntılı performans analizleri yapılabilir. Ayrıca, önerilen algoritmanın programlanabilir donanımlar üzerinde donanım mimarisi olarak gerçekleştirilmesi de muhtemel çalışmalardandır.

Tablo 2.Çalışmada kullanılan veri setleri

Veri Seti	Öznitelikler	Örnek Sayısı	Tür
Breast Cancer	10	683	İkili Sınıflandırma
Abalone	8	4177	Regresyon
Segments	20	3000	Çoklu Sınıflandırma
Iris	5	150	Çoklu Sınıflandırma
Diabetes	9	768	İkili Sınıflandırma
Body Fat	15	252	Regresyon
Auto Price	16	160	Regresyon
Boston Housing	14	506	Regresyon

KAYNAKLAR

[1] G. B. Huang, H. Zhou, X. Ding, ve R. Zhang, "Extreme Learning Machine for Regression and Multiclass Classification", *IEEE Trans. Syst. Man Cybern. Part B Cybern.*, c. 42, sy 2, ss. 513-529, Nis. 2012, doi: 10.1109/TSMCB.2011.2168604.

[2] G.-B. Huang, D. H. Wang, ve Y. Lan, "Extreme learning machines: a survey", *Int. J. Mach. Learn. Cybern.*, c. 2, sy 2, ss. 107-122, Haz. 2011, doi: 10.1007/s13042-011-0019-y.

[3] G.-B. Huang, Q.-Y. Zhu, ve C.-K. Siew, "Extreme learning machine: Theory and applications", *Neurocomputing*, c. 70, sy 1, ss. 489-501, Ara. 2006, doi: 10.1016/j.neucom.2005.12.126.

[4] Yoan Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, ve A. Lendasse, "OP-ELM: Optimally Pruned Extreme Learning Machine", *IEEE Trans. Neural Netw.*, c. 21, sy 1, ss. 158-162, Oca. 2010, doi: 10.1109/TNN.2009.2036259.

[5] B. Efron, T. Hastie, I. Johnstone, ve R. Tibshirani, "Least angle regression", *Ann. Stat.*, c. 32, sy 2, ss. 407-499, Nis. 2004, doi: 10.1214/009053604000000067.

[6] H.-J. Rong, Y.-S. Ong, A.-H. Tan, ve Z. Zhu, "A fast pruned-extreme learning machine for classification problem", *Neurocomputing*, c. 72, sy 1-3, ss. 359-366, Ara. 2008, doi: 10.1016/j.neucom.2008.01.005.

[7] Z. Shao, M. J. Er, ve N. Wang, "An Efficient Leave-One-Out Cross-Validation-Based Extreme Learning Machine (ELOO-ELM) With Minimal User Intervention", *IEEE Trans. Cybern.*, c. 46, sy 8, ss. 1939-1951, Ağu. 2016, doi: 10.1109/TCYB.2015.2458177.

[8] Y. Miche, M. van Heeswijk, P. Bas, O. Simula, ve A. Lendasse, "TROP-ELM: A double-regularized ELM using LARS and Tikhonov regularization", *Neurocomputing*, c. 74, sy 16, ss. 2413-2421, Eyl. 2011, doi: 10.1016/j.neucom.2010.12.042.

[9] Ö. Polat ve S. K. Kayhan, "GPU-accelerated and mixed norm regularized online extreme learning machine", *Concurr. Comput. Pract. Exp.*, c. 34, sy 15, s. e6967, 2022, doi: 10.1002/cpe.6967.

[10] H. T. Huynh ve Y. Won, "Regularized online sequential learning algorithm for single-hidden layer feedforward neural networks", *Pattern Recognit. Lett.*, c. 32, sy 14, ss. 1930-1935, Eki. 2011, doi: 10.1016/j.patrec.2011.07.016.

[11] Z. Shao ve M. J. Er, "An online sequential learning algorithm for regularized Extreme Learning Machine", *Neurocomputing*, c. 173, ss. 778-788, Oca. 2016, doi: 10.1016/j.neucom.2015.08.029.

[12] T. Song, D. Li, Z. Liu, ve W. Yang, "Online ADMM-Based Extreme Learning Machine for Sparse Supervised Learning", *IEEE Access*, c. 7, ss. 64533-64544, 2019, doi: 10.1109/ACCESS.2019.2915970.

[13] C.-H. Zhang ve J. Huang, "The sparsity and bias of the Lasso selection in high-dimensional linear regression", *Ann. Stat.*, c. 36, sy 4, ss. 1567-1594, Ağu. 2008, doi: 10.1214/07-AOS520.

[14] "A Survey on Nonconvex Regularization-Based Sparse and Low-Rank Recovery in Signal Processing, Statistics, and Machine Learning | IEEE Journals & Magazine | IEEE Xplore". Erişim: 11 Ekim 2023. [Çevrimiçi]. Erişim adresi: https://ieeexplore.ieee.org/document/8531588

[15] C.-H. Zhang, "Nearly unbiased variable selection under minimax concave penalty", *Ann. Stat.*, c. 38, sy 2, ss. 894-942, Nis. 2010, doi: 10.1214/09-AOS729.

[16] I. Selesnick, "Sparse Regularization via Convex Analysis", *IEEE Trans. Signal Process.*, c. 65, sy 17, ss. 4481-4494, Eyl. 2017, doi: 10.1109/TSP.2017.2711501.

[17] H. H. Bauschke ve P. L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. içinde CMS Books in Mathematics. New York, NY: Springer, 2011. doi: 10.1007/978-1-4419-9467-7.

[18] "Citation - UCI Machine Learning Repository". Erişim: 11 Ekim 2023. [Çevrimiçi]. Erişim adresi: https://archive-beta.ics.uci.edu/citation