Dağıtılmış Sistemler için Tekrarlanan Sinir Ağları Merkezli Çevrimiçi Öğrenim Algoritmaları
Recurrent Neural Networks Based Online Learning Algorithms for Distributed Systems

Tolga Ergen, S. Onur Şahin ve S. Serdar Kozat
Elektrik ve Elektronik Mühendisliği Bölümü
İhsan Doğramacı Bilkent Üniversitesi
Ankara, Türkiye
{ergen, ssahin, kozat}@ee.bilkent.edu.tr

***Özetçe* —Bu bildiride, dağılmış ağlar içerisinde Uzun Kısa-Soluklu Bellek (UKSB) mimarisinin çevrimiçi parametre öğrenmesi çalışılmıştır. Öncelikle, bağlanım problemi için UKSB tabanlı bir yapı ortaya konulmuştur. Daha sonra, ağdaki her bir düğüm için bu yapının denklemleri durum uzay formunda sunulmuştur. Bu form kullanılarak, Dağılmış Parçacık Süzme (DPS) eğitim yöntemiyle parametre öğrenmesi yapılmaktadır. Önerilen eğitim algoritması asimptotik olarak en iyi parametre kümesine yakınsamaktadır. Sunulan algoritma bu performansı gösterirken, yalnızca etkin birinci dereceden gradyan tabanlı algoritmalara yakın bir hesaplama karmaşıklığına sahiptir. Gerçek hayat uygulamalarından alınan veri kümeleriyle yapılan deneylerde, önerilen algoritma geleneksel algoritmalara kıyasla yüksek performans artışı göstermiştir.**

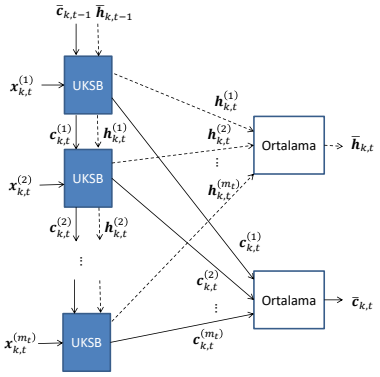
Anahtar Kelimeler*—*uzun kısa soluklu bellek ağları, dağıtılmış sistemler, çevrimiçi eğitim, ardışık bağlanım

***Abstract*—In this paper, we investigate online parameter learning for Long Short Term Memory (LSTM) architectures in distributed networks. Here, we first introduce an LSTM based structure for regression. Then, we provide the equations of this structure in a state space form for each node in our network. Using this form, we then learn the parameters via our Distributed Particle Filtering based (DPF) training method. Our training method asymptotically converges to the optimal parameter set provided that we satisfy certain trivial requirements. While achieving this performance, our training method only causes a computational load that is similar to the efficient first order gradient based training methods. Through real life experiments, we show substantial performance gains compared to the conventional methods.**

Keywords*—*long short term memory networks, distributed systems, online training, sequential regression

I. Giriş

Sinir ağları, yüksek modelleme kabiliyetleri sayesinde birçok gerçek hayat uygulamasında üstün performans göstermektedir [1]. Geçmiş bilgiyi depolamalarından ötürü, özellikle, tekrarlanan sinir ağları (TSA) zaman dizilerini modellemek amacıyla kullanılmaktadır [2]. Ancak, basit TSA yapısı, yinelenen çarpımlardan dolayı gradyanların aşırı hızda büyümesi veya küçülmesi sorunuyla karşılaşır [2]. Bu sorunları çözmek amacıyla düzenleyici yapılarla zenginleştirilmiş özgün bir tekrarlayan sinir ağı olan uzun kısa-soluklu bellek sinir ağları sunulmuştur [3]. Ancak, eklenen bu düzenleyici yapılar



Şekil 1: Sinir ağındaki düğüm k için detaylı bir şematik.

birçok parametreyi de beraberinde getirdiği için farklı öğrenim sorunlarını ortaya çıkarmıştır. Bu sorunları çözme amacıyla, bu bildiride, UKSB sinir ağlarının çevrimiçi öğrenimi doğrusal olmayan bağlanım problemi üzerinden çalışılmaktadır.

Literatürde, UKSB sinir ağları genellikle toplu olarak eğitilir. Toplu eğitimde tüm veri ulaşılabilir durumdadır ve birlikte işlenir. Ancak, büyük veri uygulamalarında, tüm verinin aynı yerde tutulması çeşitli depolama sorunlarına neden olmaktadır [4]. Ek olarak, birçok uygulamada veriler ardışık olarak elde edilmektedir ve bu durum toplu eğitimin kullanılmasını önlemektedir. Bu nedenle, bildiride ardışık eğitim kullanılmaktadır. Ardışık eğitimde, veriler sıralı olarak elde edilmekte ve UKSB sinir ağının parametreleri her elde edilen veri sonrası güncellenmektedir. Ancak, büyük veri üzerinde çalışıldığından dolayı, ardışık eğitim kullanıldığı halde depolama sorunları devam edebilmektedir. Bu soruna bir çözüm olarak sunulan dağıtılmış ağlar, tüm veriyi küçük parçalara bölerek dağıtılmış sistem içerisindeki çok sayıda düğüme paylaştırır [5]. Her bir düğüm kendisine gönderilen veri parçasını işledikten sonra, bu düğümlerde eğitilen parametreler merkez düğümde birleştirilir. Bu durumda, merkez düğüm için yüksek hesaplama gücü ve depolama alanı gereksinimi ortaya çıkmaktadır. Ayrıca, merkezleştirilmiş yöntemler, merkez düğüm üzerinde gerçekleşecek bir hataya karşı daha duyarlıdırlar. Bu bildiride, yukarıda bahsedilen sorunların çözümü amacıyla her bir düğümün yalnızca kendi komşu düğümleriyle veri paylaşımında bulunabildiği bir dağıtılmış sistem kullanılmıştır. Bu sistemde, merkez düğümü gereksinimi ortadan kaldırılmıştır.

UKSB sinir ağlarının eğitimi amacıyla genellikle olasılıksal gradyan inişi (OGİ) yönteminden faydalanılmaktadır. Bu

978-1-5386-1501-0/18/$31.00

yöntem yalnızca birinci dereceden gradyan bilgisini kullanır. Bu durum hesaplama karmaşıklığını düşük bir seviyede tutarak verimliliği arttırmasına rağmen bazı uygulamalarda yetersiz bir performans ve yakınsama sorunlarına neden olmaktadır [6]. Bu sorunları çözmek amacıyla sunulan genişletilmiş Kalman süzme (GKS) gibi ikinci dereceden gradyan bilgisini de kullanan yöntemler ise üstün bir performans sağlamasına rağmen çok yüksek hesaplama karmaşıklığına sahiptirler [6]. Bu bildiride, hem ikinci dereceden gradyan bilgisini kullanan yöntemlerin yüksek performansına hem de birinci dereceden gradyan bilgisini kullanan yöntemlerin düşük hesaplama karmaşıklığına sahip parçacık süzme (PS) tabanlı dağıtılmış çevrimiçi eğitim algoritması sunulmaktadır.

II. Problem ve Model Tanimi

Burada, K tane düğümden oluşan bir dağıtılmış ağı ele alınmaktadır. Her bir düğüm k, $\mathcal{N}_k$ ile gösterilen bir komşular kümesine sahiptir ve yalnızca bu küme içerisindeki düğümlerle veri paylaşımında bulunabilmektedir. Burada, düğüm k da komşu kümesi $\mathcal{N}_k$'nın içerisindedir, $k \in \mathcal{N}_k$. Her bir düğüm k, ardışık olarak $\{\boldsymbol{X}_{k,t}\}_{t\geq 1}$ bağlanım matrisi ve kendisine karşılık gelen $\{y_{k,t}\}_{t\geq 1}$, $y_{k,t} \in \mathbb{R}$, hedef sinyalini almaktadır. Bağlanım matrisi $\boldsymbol{X}_{k,t} = [\boldsymbol{x}_{k,t}^{(1)}\ \boldsymbol{x}_{k,t}^{(2)} \dots \boldsymbol{x}_{k,t}^{(m_t)}]$ olarak tanımlanmaktadır. Burada, girdi vektörleri $\boldsymbol{x}_{k,t}^{(l)} \in \mathbb{R}^p$, $\forall l \in \{1, 2, \dots, m_t\}$ bağlanım matrisinin sütunlarını oluşturmaktadır. $m_t \in \mathbb{Z}^+$ ise bağlanım matrisinin sütun sayısını göstermektedir ve sütun sayısı zamana bağlı olarak değişkenlik gösterebilmektedir. Burada amaç, herhangi bir t anında, bu ana kadar elde ettiğimiz girdiler ile t anındaki hedef sinyali olan $y_{k,t}$ arasında bir ilişki bularak $y_{k,t}$'yı tahmin etmektir. Her bir düğüm k için $\hat{y}_{k,t}$ ile gösterilen bu tahmin geçmiş ve şimdiki gözlemlerin bir fonksiyonu olarak düşünülebilir.

Bu bildiride, her bir düğüm k, kendisine karşılık gelen $\hat{y}_{k,t}$ tahminini yapabilmek için UKSB sinir ağı kullanmaktadır. Bağlanım matrisi $\boldsymbol{X}_{k,t}$, Şekil. 1'de görüldüğü üzere girdi olarak UKSB sinir ağına gönderilmektedir. UKSB sinir ağının iç yapısı aşağıdaki denklemler ile ifade edilmektedir:

$$\boldsymbol{z}_{k,t}^{(l)} = g(\boldsymbol{W}_k^{(z)}\boldsymbol{x}_{k,t}^{(l)} + \boldsymbol{R}_k^{(z)}\boldsymbol{h}_{k,t}^{(l-1)} + \boldsymbol{b}_k^{(z)}) \quad (1)$$

$$\boldsymbol{i}_{k,t}^{(l)} = \sigma(\boldsymbol{W}_k^{(i)}\boldsymbol{x}_{k,t}^{(l)} + \boldsymbol{R}_k^{(i)}\boldsymbol{h}_{k,t}^{(l-1)} + \boldsymbol{b}_k^{(i)}) \quad (2)$$

$$\boldsymbol{f}_{k,t}^{(l)} = \sigma(\boldsymbol{W}_k^{(f)}\boldsymbol{x}_{k,t}^{(l)} + \boldsymbol{R}_k^{(f)}\boldsymbol{h}_{k,t}^{(l-1)} + \boldsymbol{b}_k^{(f)}) \quad (3)$$

$$\boldsymbol{c}_{k,t}^{(l)} = \boldsymbol{i}_{k,t}^{(l)} \odot \boldsymbol{z}_{k,t}^{(l)} + \boldsymbol{f}_{k,t}^{(l)} \odot \boldsymbol{c}_{k,t}^{(l-1)} \quad (4)$$

$$\boldsymbol{o}_{k,t}^{(l)} = \sigma(\boldsymbol{W}_k^{(o)}\boldsymbol{x}_{k,t}^{(l)} + \boldsymbol{R}_k^{(o)}\boldsymbol{h}_{k,t}^{(l-1)} + \boldsymbol{b}_k^{(o)}) \quad (5)$$

$$\boldsymbol{h}_{k,t}^{(l)} = \boldsymbol{o}_{k,t}^{(l)} \odot g(\boldsymbol{c}_{k,t}^{(l)}). \quad (6)$$

Burada, t anındaki girdi vektörü $\boldsymbol{x}_{k,t}^{(l)} \in \mathbb{R}^p$, durum vektörü $\boldsymbol{c}_{k,t}^{(l)} \in \mathbb{R}^q$, çıktı vektörü $\boldsymbol{h}_{k,t}^{(l)} \in \mathbb{R}^q$ olarak gösterilmektedir. $\boldsymbol{z}_{k,t}^{(l)}$ blok girdi olarak adlandırılmakta olup, $\boldsymbol{i}_{k,t}^{(l)}$, $\boldsymbol{f}_{k,t}^{(l)}$ ve $\boldsymbol{o}_{k,t}^{(l)}$ sırasıyla girdi, unutma ve çıktı geçitlerini temsil etmektedir. Lineer olmayan $g(\cdot)$ fonksiyonu için hiperbolik tanjant fonksiyonu kullanılırken, $\sigma(\cdot)$ sigmoit fonksiyonunu temsil etmektedir. $g(\cdot)$ ve $\sigma(\cdot)$ vektörlere noktasal bazlı uygulanır. $\odot$ aynı boyuttaki iki vektörün birbirine denk gelen elemanlarını noktasal olarak çarparak yine aynı boyutta bir çarpım vektörü oluşturmaktadır. $\boldsymbol{W}^{(\cdot)} \in \mathbb{R}^{q\times p}$ girdi ağırlık matrisleri, $\boldsymbol{R}^{(\cdot)} \in \mathbb{R}^{q\times q}$ tekrarlanan girdi ağırlık matrisleri, $\boldsymbol{b}^{(\cdot)} \in \mathbb{R}^q$ ise yanlılık vektörleridir. Her düğüm k için hedef sinyal tahmini UKSB ağının çıktıları kullanılarak $\hat{y}_{k,t} = \boldsymbol{w}^T\bar{\boldsymbol{h}}_{k,t}$ şeklinde hesaplanmaktadır. $\boldsymbol{w}_{k,t} \in \mathbb{R}^q$ bağlanım ağırlık vektörü, $\bar{\boldsymbol{h}}_{k,t} \in \mathbb{R}^q$ ise Şekil 1'de gösterildiği üzere UKSB ağı çıktılarının ortalama metoduyla birleştirilmesiyle oluşan çıktı vektörüdür.

III. Dağıtılmış Çevrimiçi Öğrenim Algoritması

UKSB denklemleri (1)-(6) Şekil 1'de gösterilmekte olan yapı içerisinde ele alındığında, dağıtılmış sistem içerisindeki düğüm k'nın durum uzay formu aşağıdaki gibidir:

$$\bar{\boldsymbol{c}}_{k,t} = \Omega(\bar{\boldsymbol{c}}_{k,t-1}, \boldsymbol{X}_{k,t}, \bar{\boldsymbol{h}}_{k,t-1}) \quad (7)$$

$$\bar{\boldsymbol{h}}_{k,t} = \Theta(\bar{\boldsymbol{c}}_{k,t}, \boldsymbol{X}_{k,t}, \bar{\boldsymbol{h}}_{k,t-1}) \quad (8)$$

$$\boldsymbol{\alpha}_{k,t} = \boldsymbol{\alpha}_{k,t-1} \quad (9)$$

$$y_{k,t} = \boldsymbol{w}_{k,t}^T\bar{\boldsymbol{h}}_{k,t} + \varepsilon_{k,t}. \quad (10)$$

Burada, $\Omega(\cdot)$ ve $\Theta(\cdot)$, UKSB sinir ağı ve ortalama methodunun, UKSB ağının girdisi ve geçmiş durum değişkenleri üzerinde yaptığı işlemleri temsil eden fonsiyonlardır. $\boldsymbol{\alpha}_{k,t} \in \mathbb{R}^{n_\alpha}$ içerisinde $\{\boldsymbol{w}_k, \boldsymbol{W}_k^{(z)}, \boldsymbol{R}_k^{(z)}, \boldsymbol{b}_k^{(z)}, \boldsymbol{W}_k^{(i)}, \boldsymbol{R}_k^{(i)}, \boldsymbol{b}_k^{(i)}, \boldsymbol{W}_k^{(f)}, \boldsymbol{R}_k^{(f)}, \boldsymbol{b}_k^{(f)}, \boldsymbol{W}_k^{(o)}, \boldsymbol{R}_k^{(o)}, \boldsymbol{b}_k^{(o)}\}$ parametrelerini bulunduran vektördür. Bu durumda parametre vektörü $\boldsymbol{\alpha}$'nın boyutu $n_\alpha = 4q(p+q) + 5q$ olmaktadır. (9), UKSB sistem parametrelerini öğrenmek amacıyla denklemlere dahil edilmiştir. $\varepsilon_{k,t}$ tahmindeki hatayı temsil etmektedir ve bir rastgele değişken olarak modellenmektedir. [7]'te belirtilen PS algoritması üzerine varsayımlara dayanarak, (7)-(10) denklemleri aşağıdaki şekilde yazılır:

$$\boldsymbol{s}_{k,t} = \varphi(\boldsymbol{s}_{k,t-1}, \boldsymbol{X}_{k,t}) + \boldsymbol{\epsilon}_{k,t} \quad (11)$$

$$y_{k,t} = \boldsymbol{w}_{k,t}^T\bar{\boldsymbol{h}}_{k,t} + \xi_{k,t}. \quad (12)$$

Burada, $\boldsymbol{\epsilon}_{k,t}$ ve $\varepsilon_{k,t}$ sıralı olarak sistemdeki bağımsız durum ve ölçüm hatalarını temsil etmektedir. $\varphi(\cdot,\cdot)$, (7)-(10) işlemlerini temsil ederken, $\boldsymbol{s}_{k,t} = [\bar{\boldsymbol{c}}_{k,t}^T, \bar{\boldsymbol{y}}_{k,t}^T, \boldsymbol{\alpha}_{k,t}^T]^T$ olarak tanımlanır.

A. Parçacık Süzme Tabanlı Çevrimiçi Eğitim

Bu bölümde, düğüm sayısının $K = 1$ olduğu durum üzerinden, genel PS algoritması anlatılmaktadır. Burada amaç, ortalama kareli hata bakımından en iyi parametre tahmini olan $E[\boldsymbol{s}_{k,t}|y_{k,1:t}]$'yi elde etmektir. Bu amaç doğrultusunda ilk olarak durumların sonsal dağılım fonksiyonu $p(\boldsymbol{s}_{k,t}|y_{k,1:t})$ hesaplanır ve buradan koşullu ortalama tahmini elde edilir. Sonsal yoğunluk fonksiyonunu hesaplamak için [7]'de sunulmuş olan PS algoritması uygulanmaktadır.

Bu algoritmada, sonsal yoğunluk fonksiyonu $p(\boldsymbol{s}_{k,t}|y_{k,1:t})$'nin örnekleri ve bu örneklere karşılık gelen ağırlıklar, $\{\boldsymbol{s}_{k,t}^i, w_{k,t}^i\}_{i=1}^N$ kullanılmaktadır. Bu örneklere dayanarak, sonsal yoğunluk fonksiyonları aşağıdaki gibi yazılır:

$$p(\boldsymbol{s}_{k,t}|y_{k,1:t}) \approx \sum_{i=1}^{N} \omega_{k,t}^i \delta(\boldsymbol{s}_{k,t} - \boldsymbol{s}_{k,t}^i). \quad (13)$$

Sonsal yoğunluk fonksiyonunu örneklemek genel olarak zordur ve takip edilemez. Bu nedenle önem fonksiyonu olarak adlandırılan $q(\boldsymbol{s}_{k,t}|y_{k,1:t})$ örneklenir. Bu durumda örneklerin ağırlıkları aşağıdaki formül üzerinden hesaplanmaktadır:

$$w_{k,t}^i \propto \frac{p(\boldsymbol{s}_{k,t}^i|y_{k,1:t})}{q(\boldsymbol{s}_{k,t}^i|y_{k,1:t})},\ \sum_{i=1}^{N} \omega_{k,t}^i = 1. \quad (14)$$

Ardından, (14)'in çarpanlarına ayrılması ile aşağıdaki tekrar-

lanan formüle ulaşılır:

$$\omega_{k,t}^i \propto \frac{p(y_{k,t}|\boldsymbol{s}_{k,t}^i)p(\boldsymbol{s}_{k,t}^i|\boldsymbol{s}_{k,t-1}^i)}{q(\boldsymbol{s}_{k,t}^i|\boldsymbol{s}_{k,t-1}^i, y_{k,t})}\omega_{k,t-1}^i. \quad (15)$$

Bütün parçacıkların denklem (15)'e katkısının göz ardı edilemeyecek miktarda olması amacıyla, önem fonksiyonu, $\text{var}[\{w_{k,t}\}_{i=1}^N]$ en küçük değerine ulaşacak şekilde seçilir. Bu durumda, ağırlıklar için küçük değişinti koşulunu sağlaması dolayısıyla $p(\boldsymbol{s}_{k,t}^i|\boldsymbol{s}_{k,t-1}^i)$, önem fonksiyonu olarak seçilir. $p(\boldsymbol{s}_{k,t}^i|\boldsymbol{s}_{k,t-1}^i)$, denklem (15)'te $q(\boldsymbol{s}_{k,t}^i|\boldsymbol{s}_{k,t-1}^i, y_{k,t})$ yerine yazıldığında, bu denklem sadeleşerek

$$\omega_{k,t}^i \propto p(y_{k,t}|\boldsymbol{s}_{k,t}^i)\omega_{k,t-1}^i \quad (16)$$

elde edilir. (13) ve (16) birlikte değerlendirildiğinde, durum vektörünün tahmini aşağıdaki gibi hesaplanmaktadır:

$$\mathbf{E}[\boldsymbol{s}_{k,t}|y_{k,1:t}] = \int \boldsymbol{s}_{k,t}p(\boldsymbol{s}_{k,t}|y_{k,1:t})d\boldsymbol{s}_{k,t} \approx \sum_{i=1}^N \omega_{k,t}^i \boldsymbol{s}_{k,t}^i.$$

Ağırlıkların değişinti değeri, önem fonksiyonu kullanılarak küçük tutulmaya çalışıldığı halde, zamanla bazı parçacıkların etkisi artarken bazı parçacıkların etkisi azalır. Bu nedenle, yeniden örnekleme yapılarak, sisteme etkisi göz ardı edilebilecek parçacıklar elenmekte değişitinin artması önlenmektedir.

B. Dağıtılmış Parçacık Süzme Tabanlı Çevrimiçi Eğitim

Bu bölümde, birden fazla düğümden oluşabilen ve her bir düğüm k'nın kendi komşu kümesi $\mathcal{N}_k$ içerisinde bulunan düğümlerle bilgi paylaşımında bulunabildiği dağılmış sistemin çevrimiçi eğitim algoritması sunulmaktadır. Bu dağılmış sistemin eğitimi için Markov Zinciri Dağılmış Parçacık Süzme (MZDPS) algoritması [8] kullanılmaktadır. MZDPS algoritmasında, parçacıklar ağ içerisinde rastgele bir şekilde komşu kümelerinde bulunan bir düğüme hareket ederek, bu düğümün ağırlığının güncellenmesini sağlar. Dağıtılmış ağ $G = (V, E)$ ile gösterilen bir grafik olarak ele alınabilir. Burada, V, grafik G'de bulunan köşeleri, E ise kenarları temsil etmektedir. Buna ek olarak, parçacık i'nin, düğüm k'yı a adımda ziyaret etme sayısı $M^i(k, a)$ ile gösterilmektedir. Her bir parçacık belirli bir olasılığa göre komşu kümesi içerisinde bulunan düğümlerden birine ulaşmaktadır ve bu olasılıklar, komşu olasılık matrisi $\mathcal{A}$ tarafından tutulur. Bu çerçevede, toplam kenar sayısı $|E(G)|$ ve atılan toplam adım sayısı a olarak tanımladığında, her bir parçacık her bir düğüm k'yı ziyaret ettiğinde kendi ağırlığını $p(y_{k,t})^{\frac{2|E(G)|}{a\eta_k}}$ ile çarpılmaktadır. Bu durumda, toplam a adım atıldığında,parçacık i'nin düğüm k üzerinde yaptığı güncelleme aşağıdaki gibidir:

$$w_{k,t}^i = w_{k,t-1}^i \prod_{j=1}^K p(y_{j,t}|\boldsymbol{s}_{k,t}^i)^{\frac{2|E(G)|}{m\eta_j}M^i(j,m)}. \quad (17)$$

Düğüm k üzerinde, parçacıklar tarafından t anına kadar karşılaşılan örnekler $O_{k,t}$ ile gösterilirse sonsal yoğunluk fonsiyonu aşağıdaki gibi yazılır:

$$p(\boldsymbol{s}_{k,t}|O_{k,t}) \approx \sum_{i=1}^N w_{k,t}^i \delta(\boldsymbol{s}_{k,t} - \boldsymbol{s}_{k,t}^i), \quad (18)$$

Sistem parametrelerini içinde bulunduran durum vektörü tahmini aşağıdaki şekilde hesaplanır:

$$\mathbf{E}[\boldsymbol{s}_{k,t}|O_{k,t}] = \int \boldsymbol{s}_{k,t}p(\boldsymbol{s}_{k,t}|O_{k,t})d\boldsymbol{s}_{k,t} \approx \sum_{i=1}^N \omega_{k,t}^i \boldsymbol{s}_{k,t}^i. \quad (19)$$

Algorithm 1 Dağıtılmış Parçacık Süzme Algoritması Tabanlı Parametre Öğrenim Algoritması

1: $\forall j$, $p(\boldsymbol{s}_t|\{\boldsymbol{s}_{k,t-1}^i\}_{i=1}^{N(j)})$'dan $\boldsymbol{s}_{j,t}^{N(j)}$'yı örnekle
2: $\{w_{j,t}^i\}_{i=1}^{N(j)} = 1, \forall j$
3: **for** a adım **do**
4: Parçacıkları $\mathcal{A}$'ya göre hareket ettir
5: **for** $j = 1 : K$ **do**
6: $\{\boldsymbol{s}_{j,t}^i\}_{i=1}^{N(j)} = \bigcup_{l\in\mathcal{N}_j}\{\boldsymbol{s}_{l,t}^i\}_{i\in\mathcal{I}_{l\to j}}$
7: $\{w_{j,t}^i\}_{i=1}^{N(j)} = \bigcup_{l\in\mathcal{N}_j}\{w_{l,t}^i\}_{i\in\mathcal{I}_{l\to j}}$
8: $\{w_{j,t}^i\}_{i=1}^{N(j)} \leftarrow \{w_{j,t}^i\}_{i=1}^{N(j)} p(d_{j,t}|\{\boldsymbol{s}_{j,t}^i\}_{i=1}^{N(j)})^{\frac{2|E(G)|}{a\eta_j}}$
9: **end for**
10: **end for**
11: **for** $j = 1 : K$ **do**
12: Düğüm j'deki parçacıkları yeniden örnekle ve tahmin hesapla
13: **end for**

Burada sunulan yöntem ile her bir düğümde bulunan sistem parametrelerinin tahmini Algoritma 1'de açıklandığı şekilde yapılabilmektedir. Algoritma 1'de, $N(j)$ j düğümündeki parçacık sayısını temsil ederken, $\mathcal{I}_{l\to j}$ ise l düğümünden j düğümüne hareket eden parçacıkların indekslerini göstermektedir.

Teorem 1: *$\boldsymbol{s}_{k,t}$ vektörünün düğüm k için aşağıdaki eşitsizliği sağlayan bir sınırlandırılmış durum vektörü olduğunu varsayalım:*

$$0 < p_0 \leq p(y_{k,t}|\boldsymbol{s}_{k,t}) \leq ||p||_\infty < \infty \quad (20)$$

Burada, $||p||_\infty$, $p(y_{k,t}|\boldsymbol{s}_{k,t})$ yoğunluk fonksiyonunun en büyük değerini, p_0 ise bir sabiti temsil etmektedir. Bu durumda, $\boldsymbol{s}_{k,t}$ için ortalama kareli hata bakımından aşağıdaki yakınsama sonuçlarına ulaşılmaktadır:

$$\sum_{i=1}^N \omega_{k,t}^i \boldsymbol{s}_{k,t}^i \to \mathbf{E}[\boldsymbol{s}_{k,t}|\{d_{j,1:t}\}_{j=1}^K],\ N \to \infty \text{ ve } k \to \infty.$$

Teorem 1'in ispatı. (20) ve [9] kullanılarak:

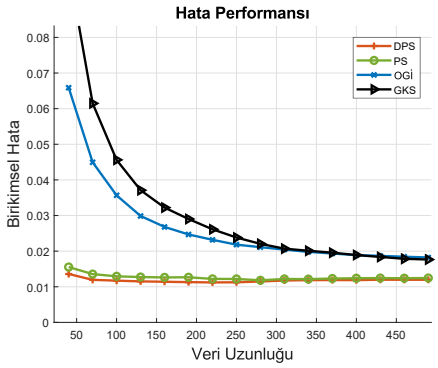
$$\mathbf{E}\big[\big(\mathbf{E}[\pi(\boldsymbol{s}_t)|\{y_{j,1:t}\}_{j=1}^K] - \sum_{i=1}^N \omega_{k,t}^i \pi(\boldsymbol{s}_{k,t}^i)\big)^2\big] \leq ||\pi||_\infty^2 \left(C_t\sqrt{U(a,\upsilon)} + \sqrt{\frac{\varsigma_t}{N}}\right)^2. \quad (21)$$

Burada, π değerleri sınırlandılmış bir fonsiyondur. υ, $\mathcal{A}$ matrisinin ikinci en büyük mutlak değerli özdeğerini, ς_t ve C_t ise zamana bağlı sabitleri temsil etmektedir. $U(a, \upsilon)$, [9]'de tanımlandığı üzere a'nın, a sonsuza giderken, sıfıra giden bir fonksiyonudur. Durum vektörü $\boldsymbol{s}_{k,t}$, sınırlandırılmış bir değer olduğu için $\pi(\boldsymbol{s}_{k,t}) = \boldsymbol{s}_{k,t}$ olarak seçilmektedir. $\pi(\boldsymbol{s}_{k,t}) = \boldsymbol{s}_{k,t}$ seçimiyle birlikte (21)'yi, N ve a sonsuza giderken değerlendirdiğimizde ispat sonuçlanmaktadır. □

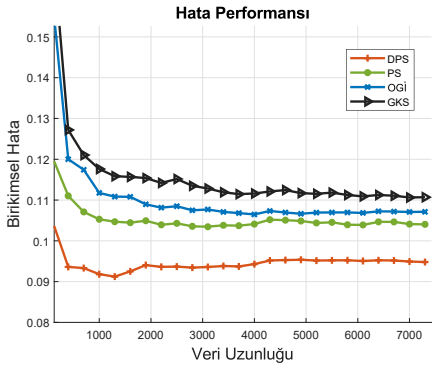
(11), (12) ve (16) denklemleri göz önünde bulundurularak, matris vektör çarpımlarından dolayı önerilen algoritma Tablo I'de belirtildiği gibi $\mathcal{O}\left(N(k)(q^2 + qp)\right)$ hesaplama karmaşıklığına sahip olmaktadır.

IV. Sayısal Örnekler

Bu bölümde, önerilen PS ve DPS eğitim algoritmalarının geleneksel yöntemler karşısındaki performansı değerlendirilmiştir. İlk olarak Hong Kong döviz kuru veri kümesi [10] kul-



Şekil 2: Yöntemlerin Hong Kong döviz kuru veri kümesi üzerinde birikimsel hata performansları.



Şekil 3: Yöntemlerin cümle veri kümesi üzerinde birikimsel hata performansları.

lanılmıştır. Bu veri kümesi 1 Amerikan doları değerine karşılık gelen Hong Kong doları miktarının günlük olarak kaydedilmesiyle oluşturulmuştur ve son iki günde elde edilmiş döviz kuru değerlerini kullanarak bir sonraki günün kur değeri tahmin edilmektedir. Bu deneyde, algoritmaların yakınsama hızları karşılaştırılmaktadır. Bu yüzden, algoritmaların parametreleri algoritmalar aynı kararlı durum hata değerine yakınsayacak şekilde ayarlanmıştır. Bu bağlamda, her bir düğüm k için parametreler şu şekilde seçilmiştir. Girdi boyutu $p = 2$ olduğu için $q = 2$ olarak seçilmiştir. Buna ek olarak, $K = 4$. PS algoritması için parçacık sayısı $N(k) = 80$ olarak belirlenmiştir. $\varepsilon_{k,t}$ ve $\boldsymbol{\epsilon}_{k,t}$ sıfır ortalamalı Gauss dağılımından $\mathrm{var}[\varepsilon_{k,t}] = 0.01$ ve $\mathrm{cov}[\boldsymbol{\epsilon}_{k,t}] = 0.0004\boldsymbol{I}$ olacak şekilde kullanılmaktadır. DPS tabanlı algoritma için, adım sayısı $a = 3$ ve komşu matrisi $\mathcal{A} = [0\ \frac{1}{2}\ 0\ \frac{1}{2}; \frac{1}{2}\ 0\ \frac{1}{2}\ 0; 0\ \frac{1}{2}\ 0 \frac{1}{2}; \frac{1}{2}\ 0\ \frac{1}{2}\ 0]$ olarak seçilmiştir. GKS algoritması için aynı gürültü istatistikleri kullanılmıştır. OGİ algoritması için öğrenme hızı $\mu = 0.1$ olarak seçilmiştir. Şekil 2'de algoritmaların tahmin performansları gösterilmektedir. Verinin lineer olmayan karakterinden dolayı GKS algoritması diğer algoritmalara göre daha yavaş bir yakınsama hızına sahiptir. Ek olarak, OGİ algoritması da yalnızca birinci dereceden gradyan bilgisini kullanması sebebiyle PS tabanlı algoritmalara göre daha yavaş bir yakınsama hızına sahiptir. Parçacık süzme tabanlı algoritmalar, son ortalama kareli hata seviyesine yüksek yakınsama hızıyla ulaşmaktadırlar. Bütün algoritmalar arasında, dağılmış ağ yapısı sayesinde DPS algoritması en yüksek yakınsama hızına sahiptir. İkinci veri kümemiz, her bir kelimenin vektör olarak tutulduğu cümle veri kümesidir [11]. Bu deneyde, algoritmaların kararlı durum hata performansları karşılaştırılmaktadır. Bu nedenle, parametreler, algoritmaların yakınsama hızları aynı olacak şekilde seçilmiştir. Burada, girdi olarak kullanılan bağlanım matrisi $\boldsymbol{X}_{k,t} \in \mathbb{R}^{2\times m_t}$ değişken uzunluktadır. Bu deneyde değişmiş olan parametreler $N(k) = 50$, $\mathrm{cov}[\boldsymbol{\epsilon}_{k,t}] = (0.025)^2\boldsymbol{I}$ ve $\mu = 0.055$'tir. Diğer parametreler için ilk deneydeki ile aynı değerler seçilmiştir. Şekil 3'te, algoritmaların etiketleme performansları gösterilmektedir. Yine GKS algoritması en yüksek kararlı durum hata değerine sahiptir. Ek olarak, OGİ algoritması parçacık süzme tabanlı algoritmalara göre daha yüksek son ortalama hata değerine sahiptir. Bu deneyde de, parçacık süzme tabanlı algoritmalar diğer algoritmalara göre üstün performans göstermişlerdir.

TABLO I: DÜĞÜM K IÇIN HESAPLAMA KARIŞIKLIĞI

Yöntem	Hesaplama Karışıklığı
OGİ	$\mathcal{O}\left(q^4 + q^2p^2\right)$
GKF	$\mathcal{O}\left(q^8 + q^4p^4\right)$
DPS	$\mathcal{O}\left(N(k)(q^2 + qp)\right)$

Düğümler arasında bilgi paylaşımı yapabilen yapısında ötürü, DPS algoritması en yüksek etiketleme performansına sahiptir.

V. SONUÇLAR

Bu bildiride, dağıtılmış bir sistemin düğümlerinde UKSB sinir ağlarının, bağlanım problemi üzerinden çevrimiçi eğitimi çalışılmıştır. İlk olarak, değişken uzunluktaki verileri işleyebilen UKSB sinir ağı tabanlı bir yapı ortaya konulmuştur. Bu yapıyı eğitebilmek amacıyla, bu yapının denklemleri durum uzay formunda yeniden yazılmıştır. Daha sonra, dağıtılmış parçacık süzme tabanlı çevrimiçi eğitim algoritmamız sunulmuştur. Bu şekilde LSTM tabanlı dağıtılmış sistemin çevrimiçi eğitimi için etkili bir algoritma elde edilmiştir. Sunulan algoritması ortalama kareli hata bakımından UKSB yapısı için en iyi parametre kümesine yakınsamayı garanti etmektedir. Ek olarak, sunulan algoritma bu üstün performansı sağlarken, birinci dereceden gradyan bilgisini kullanan yöntemlerin hesaplama karmaşıklığına sahiptir. Sayısal örneklerde, ortaya koyulan DPS tabanlı algoritmanın üstün performansını göstermiştir.

TEŞEKKÜR

Bu bildiri, Üstün Başarılı Bilim İnsanlarını Destekleme Programı kapsamında Türkiye Bilim Akademisi ve TÜBİTAK tarafından 115E917 numaralı sözleşme ile desteklenmektedir.

KAYNAKLAR

[1] D. F. Specht, "A general regression neural network," *IEEE Transactions on Neural Networks*, vol. 2, no. 6, pp. 568–576, Nov 1991.

[2] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, Mar 1994.

[3] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[4] T. Ergen and S. S. Kozat, "Online training of lstm networks in distributed systems for variable length data sequences," *IEEE Transactions on Neural Networks and Learning Systems*, 2017.

[5] K. Yuan *et al.*, "On the convergence of decentralized gradient descent," *SIAM Journal on Optimization*, vol. 26, no. 3, pp. 1835–1854, 2016.

[6] T. Ergen and S. S. Kozat, "Efficient online learning algorithms based on lstm neural networks," *IEEE transactions on neural networks and learning systems*, 2017.

[7] P. M. Djuric *et al.*, "Particle filtering," *IEEE Signal Processing Magazine*, vol. 20, no. 5, pp. 19–38, 2003.

[8] S. H. Lee and M. West, "Markov chain distributed particle filters (mcdpf)," in *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, Dec 2009, pp. 5496–5501.

[9] S. H. Lee and M. West, "Convergence of the markov chain distributed particle filter (MCDPF)," *IEEE Transactions on Signal Processing*, vol. 61, no. 4, pp. 801–812, Feb 2013.

[10] E. W. Frees, "Regression modelling with actuarial and financial applications." [Online]. Available: http://instruction.bus.wisc.edu/jfrees/jfreesbooks/Regression%20Modeling/BookWebDec2010/data.html

[11] D. Dheeru and E. Karra Taniskidou, "UCI machine learning repository," http://archive.ics.uci.edu/ml, 2017.