Yapay Sinir Ağlarında Parçalı Eğitim

Mini-Batching for Artificial Neural Network Training

Mehmet Fatih Amasyalı
mfatih@ce.yildiz.edu.tr
Yıldız Teknik Üniversitesi,
Bilgisayar Mühendisliği Bölümü

Öz

Yapay sinir ağları ile büyük veri kümeleri modellenirken eğitimi paralelleştirmek için, eğitim kümesi ağa toptan yerine parçalara ayrılarak verilir. Bu sayede eğitim süresi azaltılır. Bu çalışmada parçalı eğitimin küçük veri kü[1]melerine uygulandığındaki etkisi incelenmiştir. 4 farklı eğitim algoritması ve 11 veri kümesi üzerinde yapılan testlerde, 3 eğitim algoritması için parçalı eğitimin, toptan eğitimden daha başarılı olduğunu görülmüştür.

Anahtar kelimeler: Yapay Sinir Ağları, Toptan Öğrenme, Parçalı Öğrenme, Tekil Öğrenme, Makine Öğrenmesi, Optimizasyon, Yapay Zeka

Abstract

When the large data sets are modeling with Artificial Neural Networks, the training set is divided into mini-batches to parallelize training phase. In this way, training time is reduced. In this study, the effect of the mini-batch training was investigated when it applied to small data sets. In our experiments, 4 different learning algorithms over 11 datasets were used. It is shown that the mini-batch training is more successful than the full batch training with 3 learning algorithm.

Keywords: Artificial Neural Network, Batch Learning, Mini-batch Learning, Stochastic Learning, Machine Learning, Optimization, Artificial Intelligence

1 Giriş

Yapay sinir ağları (YSA), veri modellemede başarıyla kullanılan metotlardan biridir. Başarılı sonuçlar üretebildikleri ancak bununla birlikte, lokal minimumlara takılma olasılıklarının yüksek olduğu bilinmektedir. Bu sebeple meta-parametre seçim süreci YSA'lar için büyük önem taşımaktadır [1]. YSA'ların içerdiği meta-parametrelere örnek olarak saklı katmandaki nöron sayısı, aktivasyon fonksiyonu, ağ topolojisi, örneklerin ağa verilme yöntemi ve öğrenme algoritması verilebilir. Bu parametreler seçildikten ve ağırlıkların ilk değerleri belirlendikten sonra optimal ağırlıklar lokal arama ile arandıklarından lokal minimumlara takılma olasılığı her zaman vardır.

YSA'larda eğitim örneklerinin ağa verilmesinde toplu (batch) ve tekil (online) öğrenme olmak üzere iki temel yöntem kullanılmaktadır. Toplu eğitimde eğitim kümesinin tamamı tek seferde ağa verilirken, tekil öğrenmede eğitim örnekleri birer birer ağa verilmektedir. Her iki yöntemin diğerine göre avantaj ve dezavantajları bulunmaktadır [1, 2]. Tekil eğitimin daha hızlı yakınsadığı, daha iyi lokal minimumları bulduğu, buna karşın birçok öğrenme algoritmasının sadece toplu eğitim için uygulanabildiği bilinmektedir. Toplu eğitim paralelleştirilebilirken, tekil eğitim paralelleştirilememektedir.

Literatürde, büyük veri kümelerinin YSA'larla modellenmesinde eğitim süresinin azaltılması için parçalı eğitim önerilmektedir [3, 4, 5, 6]. Buna göre

Genderim ve kabul tarihi : 29.11.2013-27.01.2015

tek bir ağın eğitimi yerine paralel birçok ağ oluşturulur, her bir ağa eğitim kümesinin bir parçası verilir. Önceden belirlenen periyotlarla, ağlardaki ağırlıklar diğer ağlardakiler kullanılarak da güncellenir. Ağlar arasındaki bu haberleşme sayesinde ağlar birbirlerinin öğrendiklerinden haberdar olur. Parçalı paralel eğitimle, toplu eğitimdeki başarı performansı azalmadan, daha kısa sürede eğitim yapılması amaçlanır.

Bu çalışmada parçalı eğitimin bu temel kullanım amacı yerine, toplu eğitime göre başarı performansı üzerinde durulmuştur. Küçük veri kümeleri üzerinde çalışılmış, tekil eğitim uygulanamayan öğrenme algoritmaları kullanılarak toplu eğitime göre performanslar incelenmiştir. Parçalı eğitimin çeşitli parametrelerinin etkisi ölçümlenmiştir.

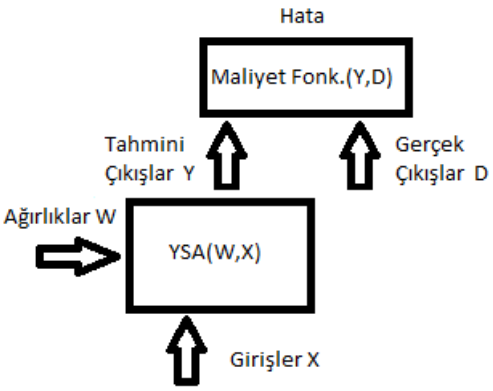
Makalenin ikinci bölümünde toplu ve tekil öğrenmenin formül ve geometrileri verilmiştir. 3. bölümde kullanılan öğrenme algoritmaları tanıtılmıştır. 4. bölümde gerçekleştirilen testlerin değişkenleri tanımlanmış ve sonuçlar sunulmuştur. Son bölümde ise sonuçlar yorumlanmış ve gelecekte yapılması planlanan çalışmalar anlatılmıştır.

2 YSA'larda Eğitim

YSA'ların eğitimi, içerdikleri nöronlar arası ağırlık değerlerinin, eğitim kümesi girişlerini çıkışlarına dönüştürecek şekilde ayarlanmasıdır.

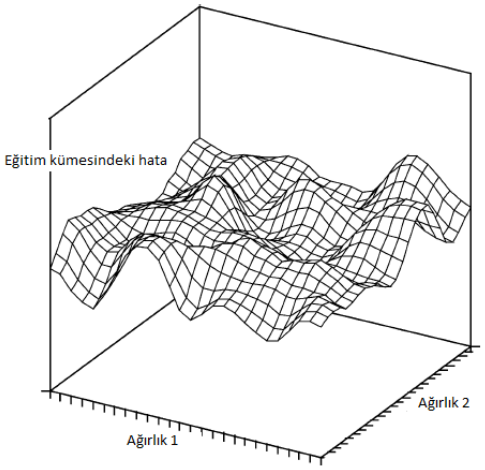
En yaygın kullanılan YSA'larının (çok katmanlı perseptron) mimarisinde, giriş katmanı, saklı katman ve çıkış katmanı yer almaktadır. Giriş ve çıkış katmanlarındaki nöron sayıları modellenmesi istenen veri kümesince belirlenir. Giriş katmanındaki nöron sayısı veri kümesinin özellik sayısı kadardır. Çıkış katmanı ise eğer veri kümesi bir sınıflandırma problemiyse sınıf sayısı kadar, regresyon problemiyse çıkış sayısı kadar nöron içerir. Ayrıca, giriş ve saklı katman, orijinden gerekli kaymaları ayarlamak için bias nöronları da içerir.

Şekil 1'de YSA'nın blok diyagramı gösterilmiştir. Maliyet fonksiyonu olarak genelde ortalama karesel hata (MSE) kullanılır.



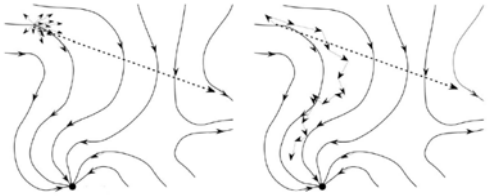
Şekil 1: YSA Blok diyagramı

Şekil 2'de iki ağırlığa sahip ütopik bir ağın eğitiminin yapıldığı uzay verilmiştir. Eğitimin başlangıcında rastgele bir (ağırlık1, ağırlık2) noktasından başlanır. Öğrenme algoritmasına ve eğitim örneklerine göre hesaplanan güncelleme vektörleriyle toplam hatanın minimum olduğu (ağırlık1, ağırlık2) noktası bulunmaya çalışılır. Ağırlık sayısının çok fazla olduğu ya da arama uzayının çözünürlüğünün çok yüksek olduğu durumlarda global arama yapılamayacağından lokal arama yapılmakta ve şekilden de görülebildiği gibi lokal minimumlara takılma problemi oluşabilmektedir.



Şekil 2: Ağırlık - Hata uzayı [2]

Toplu öğrenmede, her bir eğitim örneğinden elde edilen güncelleme vektörleri toplanarak tek seferde uygulanır. Tekil de ise bir eğitim örneği için bulunan güncelleme vektörü hemen uygulanır. Wilson ve Martinez makalelerinde, tekil eğitimin toplu eğitimden farkını Şekil 3'le açıklamıştır [2].



Şekil 3: Tekil ve Toplu eğitimin karşılaştırılması [2]

Şekil 3'te 2 boyutlu ütopik bir uzayda, solda toplu eğitim, sağda tekil eğitimin aynı başlangıç noktasından 1 iterasyon sonunda eriştikleri noktalar verilmiştir. Toplu eğitimde her bir eğitim örneği için hesaplanan güncelleme vektörleri (sol üst kısımda yer alan) toplanarak tek seferde (kesikli çizgi) uygulanmıştır. Tekil eğitimde ise güncelleme vektörleri, birbirlerine bağımlı olduklarından toplu eğitimde elde edilenlerle aynı değildir ve ağı lokal minimuma doğru götürebilmişlerdir.

Tekil öğrenmede bulunan güncelleme vektörleri birbirinden bağımsız değildir. Bir örnek için bulunan güncelleme vektörü daha önce uygulanan güncellemelere göre değişmektedir. Çünkü güncelleme vektörü ağın ağırlıklarına bağlıdır ve ağırlıklar değiştiğinde güncelleme vektörü de değişecektir. Örneklerin ağa verilme sırasına göre erişilen sonuç değişebilir [1].

Toplu öğrenmede ise güncelleme vektörleri birbirinden bağımsızdır. Güncelleme vektörleri her bir örnek için hesaplandıktan sonra toptan uygulandıklarından dolayı birbirlerini etkilememektedirler. Dolayısıyla örneklerin ağa verilme sırası da önemsizdir.

Parçalı eğitimde ağa verilecek örnek kümelerinin belirlenmesi için temelde kullanılan yöntemde, her iterasyonda önce örnekler rastgele sıralanır, ardında eşit elemanlı K adet kümeye bölünür. Her alt iterasyonda bu K kümeden birisi ağa verilir. Örneklerin rastgele sıralanması, her alt iterasyonda ağa farklı örnek birlikteliklerinin verilmesini sağlar. Şekil 4'te parçalı eğitimin algoritması verilmiştir [4].

```
For i=1: iterasyon sayısı
        K adet örnek kümesini üret
        For j=1:K
                j. örnek kümesiyle ağı eğit
```

Şekil 4: Parçalı Eğitim Algoritması

K adet örnek kümesi üretilirken, örnekler rastgele sıralanır, eşit elemanlı K adet kümeye bölünür. Üretilen kümelerin kesişimi boş kümedir.

Parçalı eğitimde, örneklerin bir kısmı birlikte ağa verildikten sonra toptan güncelleme yapılmaktadır. Parçanın içindeki örneklerin güncelleme vektörleri birbirinden bağımsız iken, parça dışındaki örnekler için hesaplanacak güncelleme vektörleri parça içindekilere bağımlıdır.

Optimizasyonda lokal minimumlardan kaçabilmek için güncellemeye bazen gürültü eklenir. Örneğin Benzetimli tavlamada (Simulated Annealing) doğru yönde olmayan güncellemelere de küçük olasılıklarla izin verilir. Tekil öğrenmede örneklerin gürültü içerebileceği ve bu sayede daha iyi lokal minimumlara erişebilme kabiliyeti olduğu ifade edilmiştir [2].

Daha iyi lokal minimumlara erişebilme avantajı olmasına rağmen birçok optimizasyon yöntemi tekil eğitime uygulanamamaktadır. Parçalı eğitim, temelde büyük veri kümeleri üzerinde çalışabilmek için önerilmiş olmasına rağmen, lokal minimumlardan kaçabilme kabiliyetine de sahip olabilir. Çünkü tüm örnekler yerine küçük bir kısmı da bir miktar gürültü içerecek ve deterministik bir güncelleme yerine rastgelelik içeren yönlendirilmiş bir güncelleme uygulayacaktır. Parçalı eğitim, aynı zamanda tekil eğitim uygulanamayan toplu eğitim algoritmalarına uygulanabilir ve bu sayede algoritmaların performansını arttırabilir.

3 Kullanılan Öğrenme Algoritmaları

YSA'ların öğrenmesi, nöronlar arası ağırlıkların uygun değerlerinin iteratif olarak belirlenmesidir. Öğrenme sürecinde her iterasyonda, mevcut ağırlık değerlerine hata hiper-düzleminde karşılık gelen

nokta hareket ettirilir (Şekil 3). Hata fonksiyonunun minimum değerinin bulunması kısıtsız bir optimizasyon problemidir. Eldeki eğitim verilerine göre bu ağırlık güncellemelerinin (hareketin) nasıl yapılacağına dair birçok yöntem geliştirilmiştir. Şekil 5'te ağırlık güncelleme algoritmalarının genel hali verilmiştir [7].

```
w(0)'ı seç.
Her iterasyonda (t):
        Adım yönünü belirle (p)
        Adım büyüklüğünü belirle (d)
        Güncelleme yap: w(t+1)=w(t)+p*d
```

Şekil 5: Genel Ağırlık Güncelleme Algoritması

Adım büyüklüğünün belirlenmesi için tüm algoritmalar aynı yöntemi kullanmaktadır. Algoritmalar arası fark, adım yönünün belirlenmesindedir.
Adım büyüklüğü belirlenirken, adım yönü belirlendikten sonra belirlenen maksimum ve minimum(>0) değerler içinde hata fonksiyonunu en az yapanı seçilir.

Adım yönü belirlenirken f hata fonksiyonunu; f(w) w ağırlık değerleri için hatayı, (w+p) ise w'nin p yönünde güncelleneceğini göstersin. Eşitlik 1'de bu güncellemenin ikinci dereceden Taylor açılımı verilmiştir.

$$f(w+p)=f(w)+p^T f'(w)+ (1/2)p^T f''(w)p \quad (1)$$

Öyle bir (w+p) değeri arıyoruz ki o noktadaki türev 0 olsun. Buna göre (1)'in p'ye göre türevi alınıp 0'a eşitlenirse Eşitlik 2 elde edilir.

$$p= - H^{-1}f'(w) \quad (2)$$

Eşitlik 2'deki p'ye literatürde Gauss-Newton adımı adı verilmektedir [8]. H ise Hessian matrisini göstermektedir.

Eşitlik 1'deki Taylor açılımında 1. derece kullanılırsa Eşitlik 3 elde edilir.

$$p= -f'(w)/| f'(w)| \quad (3)$$

Eşitlik 3'deki p'ye literatürde en dik azalma (steepest descent) adımı adı verilmektedir [8].

Literatürde 2. dereceden yaklaşımların 1. dereceye göre daha hızlı yakınsadığı gösterilmiştir [8].

Hata yüzeyinin konveks olduğu ve 2.dereceden türevinin alınabildiği durumlarda; Hessian matrisi tanımlıdır. Ancak pratikte, ağırlık yüzeyinin her noktasında bu durum mümkün olmayabilir. Eşitlik 4'te bu durum için Levenberg - Marquardt yönteminin [8] çözümü gösterilmiştir.

$$p= - (H+\lambda D)^{-1}f'(w) \quad (4)$$

Eşitlik 4'te D, Hessian matrisinin diyagonalini, λ, pozitif bir sayıyı göstermektedir. (H+λD) ifadesinde, λ 'nın büyük değerleri için (λD) baskın hale gelecek, küçük değerleri içinse (H) baskın hale gelecektir. Algoritma, (H) baskın olduğunda Gauss-Newton adımına, (λD) baskın olduğunda en dik azalma adımına benzeyecektir. Levenberg – Marquardt algoritması hata fonksiyonunun değerine göre λ değerini büyültmekte ya da küçültmektedir.

Hessian matrisinin hesaplama zamanının çok olduğu durumlar (ağırlık sayısının çok fazla olması) için Broyden – Fletcher – Goldfarb – Shanno (BFG) yöntemi [9] önerilmiştir.

Hessian matrisi direkt hesaplanmaz onun yerine yaklaşığı kullanılır. Şekil 6'da BFG algoritması verilmiştir.

```
w(0)'ı ve H(0)'ı seç.
Her iterasyonda (t):
        p= - H^-1 f'(w(t))
        Adım büyüklüğünü (d) belirle
        w(t+1)=w(t)+p*d
        s=p*d
        y=f'(w(t+1))-f'(w(t))
        H=H+((yy^T)/( y^Ts)) - ((Hss^TH)/(s^THs))
```

Şekil 6: BFG Güncelleme Algoritması

Pratikte H(0) için I*w kullanılmaktadır. Algoritmada w gibi, H de adım adım güncellenmektedir.

Ağırlık değerlerinin güncellenmesi için önerilen bir diğer yöntem Esnek Geri-yayılım (Resilient Backpropogation) (RP) algoritmasıdır [10]. Birinci dereceden türevleri kullanan bir algoritmadır. Adım boyutunun iyi belirlenemediği durumlardaki kötü

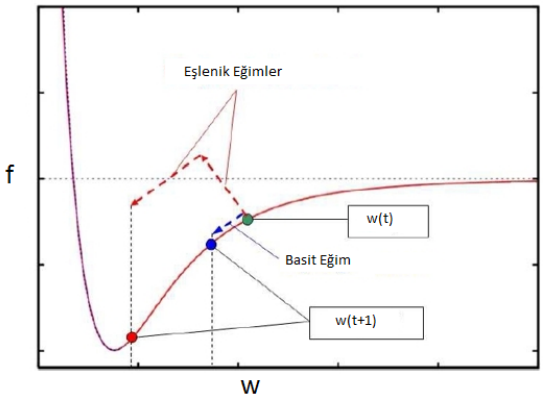
etkileri önlemek için geliştirilmiştir. Şekil 7'de RP algoritması verilmiştir.

$0<n^-<1<n^+$ olmak üzere
Her iterasyonda (t):
Her bir ağırlık için:
If $f'(w(t))*f'(w(t-1))>0$
$d(t)=d(t-1)*n^-$
else if $f'(w(t))*f'(w(t-1))<0$
$d(t)=d(t-1)*n^+$
else $d(t)=d(t-1)$
$p(t)=-sign(f'(w(t)))$
$w(t)=w(t-1)+p(t)*d(t)$

Şekil 7: RP Güncelleme Algoritması

Algoritma Şekil 7'de görüldüğü gibi güncelleme yönü olarak o noktadaki türevin işaretini, güncelleme adım boyutu için o noktadaki büyüklüğünü değil algoritmanın belirlediği iki katsayıdan birini (n^-,n^+) kullanmaktadır. Eğer türevin işareti bir önceki adıma göre değiştiyse, bir tepe atlandığını anlamakta ve adım boyutunu n^- ile çarparak azaltmakta, değişmediyse n^+ ile çarparak arttırmaktadır. Pratikte n^- için 0.5, n^+ için 1.2 değerleri kullanılmaktadır.

Ağırlık değerlerinin güncellenmesi için önerilen bir diğer yöntem Eşlenik Eğim (Conjugate Gradient) (CG) algoritmasıdır [1]. Ardışık güncelleme vektörlerinin birbirine dik olmasını amaçlar. Şekil 8'de basit eğim azaltma (gradient descent) yöntemi ile eşlenik eğim azaltma (Conjugate Gradient) birlikte verilmiştir [11].



Şekil 8: CG Güncelleme Adımları [11]

Şekil 8'de 1 boyutlu bir f fonksiyonunun minimum değerinin bulunması sürecinde 2 algoritmanın güncelleme adımları verilmiştir. Şekil 9'da CG algoritması verilmiştir.

$p(0)=-f'(w(0))$
Her iterasyonda (t):
Adım büyüklüğünü (d) belirle
$w(t+1)=w(t)+p(t)*d$
$p(t)=-f'(w(t))+b(t)p(t-1)$
$b(t)= (f'(w(t))^T f'(w(t))) / (f'(w(t-1))^T f'(w(t-1)))$

Şekil 9: CG Güncelleme Algoritması

Scaled Conjugate Gradient (SGC) algoritması [7], CG algoritmasının geliştirilmiş halidir. SGC adım büyüklüğünü (d) hesaplarken ikinci dereceden türevleri kullanır.

4 Deneysel Sonuçlar

Parçalı eğitime dair literatürde genelde büyük veri kümeleri üzerinde, veri kümesine özgü sonuçlar verilmiştir. Ancak genelleştirilebilir sonuçlara varmak için parametrelerin sistematik olarak değiştirildiği, çok sayıda veri kümesi üzerinde denenmiş bir çalışmaya rastlanmamıştır. Bu sebeple aşağıdaki sorulara cevap bulmak amacıyla çeşitli deneyler tasarlanmıştır.

1-Parçalı eğitim, küçük veri kümeleri üzerinde nasıl sonuçlar üretir?
2-Parçalı eğitimde örnek kümeleri bir kez oluşturulup bunlar mı ağa verilmelidir, yoksa her iterasyonda bu örnek kümeleri yeniden mi oluşturulmalıdır?
3-Parçalı eğitim yerine, yerine koyarak seçim yapmak nasıl sonuçlar üretir?
4-Parçaların sayısı (K) kaç olmalıdır?
5-Parçalı eğitimde, kullanılan öğrenme algoritmasının bir etkisi var mıdır?

İlk soruya cevap vermek için yapılan denemelerimizde kullanılan 11 veri kümesi Tablo 1'de verilmiştir. Veri kümeleri UCI [12] veri kümelerinden seçilmiştir.

Tablo 1: Kullanılan Veri Kümeleri

Veri kümesi	Özellik sayısı	Sınıf sayısı	Örnek sayısı
breast-cancer	39	2	286
breast-w	10	2	699
Colic	61	2	368
credit-a	43	2	690
Diabetes	9	2	768
heart-statlog	14	2	270
hepatitis	20	2	155
ionosphere	34	2	351
Labor	27	2	57
Lymph	38	2	142
Vote	17	2	435

2. soru için Şekil 4'teki algoritma ve "K adet örnek kümesini üret" adımının dış döngünün dışına konduğu algoritma denenmiştir. Bu sayede örnek kümeleri bir kez üretilmiş daha sonra aynı hep kümeler kullanılmıştır. 3. soru için "K adet örnek kümesini üret" adımında her bir küme yerine koymalı rastgele seçimle üretilmiştir. 4. Soru için 2, 3, 4 ve 10 değerleri denenmiştir. 5. Soru için RP, SCG, BFG ve LM algoritmaları denenmiştir.

Tüm denemelerde her bir eğitim örneğinin ağa 10 kez verilmesi sağlanmıştır. Tüm ağlarda saklı katmandaki nöron sayısı özellik sayısına eşit alınmıştır. Her bir veri kümesi için yapılan denemeler 10 kez tekrar edilerek ortalamaları alınmıştır. Her denemede tüm veri kümesinin rastgele seçilen yarısı eğitim, diğer yarısı test olarak kullanılmıştır. Denemeler Matlab ortamında yapılmış ve diğer tüm meta parametreler için varsayılan değerler kullanılmıştır.

Elde edilen sonuçlar Tablo 2'de yer almaktadır. Tablonun satır başlarında parçaların nasıl oluşturulduğu, K sayısı yer almaktadır. "K=10 aynı" ifadesi 10 parça oluşturulduğunu ve parçaların bir kez oluşturulduktan sonra değiştirilmeden kullanıldığı göstermektedir. "K=10 rast." ifadesi 10 parça oluşturulduğunu ve parçaların hepsi kullanıldıktan sonra 10 parçanın yeniden oluşturulduğunu göstermektedir. "BG 0.25" ifadesi, parçaların yerine koymalı seçimle orijinal örnek sayısının ¼'ü kadarının seçimiyle oluşturulduğunu göstermektedir. Sütun başlarında öğrenme algoritmaları yer almaktadır.

Tablo 2: Parçalı eğitimin toplu eğitime göre performansı

	RP	SCG	BFG	LM
K=10 rast.	**-0.1938**	-0.0093	-0.0388	0.0454
K=4 rast.	-0.1183	-0.0362	-0.0379	0.0186
K=3 rast.	-0.0422	**-0.0455**	-0.0440	-0.0048
K=2 rast.	-0.0742	-0.0149	-0.0222	-0.0021
K=10 aynı	-0.0440	-0.0188	**-0.0612**	0.0172
K=4 aynı	-0.1279	-0.0348	-0.0414	0.0032
K=3 aynı	-0.0834	**-0.0471**	**-0.0515**	0.0045
K=2 aynı	0.0075	-0.0062	-0.0312	0.0171
BG 0.10	**-0.1560**	-0.0387	-0.0401	0.0334
BG 0.25	-0.1330	-0.0441	-0.0304	0.0103
BG 0.33	-0.1030	-0.0260	-0.0332	0.0082
BG 0.50	-0.0901	-0.0157	-0.0330	0.0176

Tablo 2'deki hücre değerleri, sütun başındaki algoritmanın toplu öğrenmedeki hatasının, satır başındaki yöntemle 11 veri kümesinde ortalama yüzde kaç değiştiğini göstermektedir. Hücrelerdeki her bir değer (11 veri kümesi * 10 tekrar=) 110 değerin ortalamasıdır. Örneğin tablonun 2.satır, 2.sütununda yer alan "-0.1938" ifadesi, RP algoritmasının başarısının her seferinde yeniden oluşturulan 10 adet parça ile 11 veri kümesi üzerinde ortalama %19.38 oranında arttığını göstermektedir.

Tablo 3'te "-0.1938" değerinin nasıl bulunduğu ayrıntılı olarak gösterilmiştir.

Tablo 3'ün her hücresindeki değerler 10 denemenin ortalamasını ve standart sapmayı göstermektedir. yüzdelik farklar sütununda kalın yazılmış değerler istatistiksel olarak anlamlı farkları göstermektedir. 11 veri kümesinin 10'unda başarı artarken, 5'i üzerinde bu artış istatistiksel olarak anlamlıdır. Tablo 2'deki her bir değer Tablo 3'teki işlemle bulunmuştur.

Elde edilen sonuçların, sorduğumuz sorulara verdikleri cevaplar incelenirse:

1.soru için, parçalı eğitimin küçük veri kümelerinde kullanıldığında genelde hatayı azaltabildiği görülmüştür.

Tablo 3: Parçalı eğitimin RP algoritmasına etkileri

Veri kümesi	Toplu öğrenme RP (ort.,std)	K=10 rast. ile parçalı öğrenme RP	Yüzdelik Fark = (toplu-parçalı)/toplu
breast-cancer	0,4028 (0,134)	0,6112 (0,1554)	**-0,5174**
breast-w	0,8854 (0,0297)	0,957 (0,0095)	**-0,0809**
colic	0,5728 (0,0774)	0,5902 (0,0963)	-0,0304
credit-a	0,4951 (0,0959)	0,5014 (0,0612)	-0,0127
diabetes	0,6802 (0,0683)	0,7109 (0,0228)	-0,0451
heart-statlog	0,677 (0,0664)	0,7659 (0,0249)	**-0,1313**
hepatitis	0,7273 (0,1619)	0,7558 (0,0639)	-0,0392
ionosphere	0,4651 (0,2234)	0,7337 (0,1874)	**-0,5775**
labor	0,5714 (0,1288)	0,5714 (0,2673)	0,0000
lymph	0,5408 (0,1449)	0,6085 (0,0893)	-0,1252
vote	0,5797 (0,1391)	0,9115 (0,0387)	**-0,5724**
		ortalama	-0.1938

2.soru için, genel bir cevap verilememiş, öğrenme algoritmasına bağlı sonuçlar verilebilmiştir. RP için parçaların her seferinde yeniden oluşturulması daha iyi sonuçlar üretmiştir. SCG için anlamlı bir farkın olmadığı, BFG içinse parçaların bir kez oluşturulup aynen kullanımının daha iyi olduğu görülmüştür.

3.soru için, parçalı eğitimle, yeniden seçim (son 4 satır) arasında anlamlı bir farkın olmadığı görülmüştür.

4. sorudaki parça sayısı için genelde parça sayısının fazla olduğu durumlarda hatadaki azalmanın da fazla olduğu görülmüştür. Ancak bu durumun geçerli olmadığı sonuçlarda vardır.

5. soru için: LM (son sütun) algoritması haricindeki diğer 3 algoritma için parçalı eğitimle hatanın azaltıldığı görülmektedir.

Her bir veri kümesinde en başarılı algoritmanın hangisi olduğu Tablo 4'te verilmiştir.

Tablo 4: En başarılı algoritmalar

Veri kümesi	Algoritma	Eğitim Türü
breast-cancer	LM	K=2 aynı
breast-w	LM	K=3 aynı
colic	LM	BG 0.25
credit-a	LM	K=10 aynı
diabetes	SCG	K=3 aynı
heart-statlog	BFG	K=10 rast.
hepatitis	LM	K=4 aynı
ionosphere	LM	BG 0.50
labor	LM	K=3 rast.
lymph	SCG	K=3 rast.
vote	LM	BG 0.10

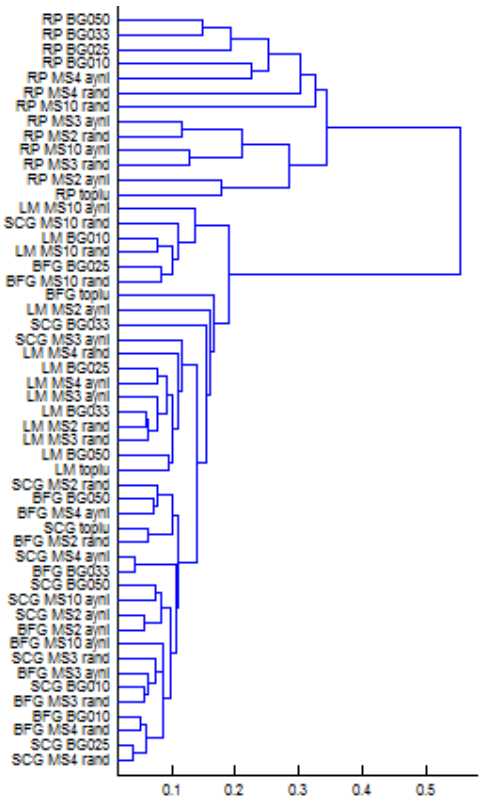
Tablo 4 incelendiğinde veri kümelerinin çoğunda en başarılı algoritmanın LM olduğu görülmektedir. Eğitim türlerinde ise parçalı eğitimin tüm veri kümelerinde daha başarılı olduğu görülmektedir. Veri kümesi karakteristikleriyle (özellik ve örnek sayısı) başarılar arasında belli bir şablon bulunmamaktadır.

Tablo 4 ve detaylı sonuçlar incelendiğinde en başarısız algoritmanın RP olduğu görülmüştür. Bunun sebebi olarak diğerleri arasında RP'nin birinci dereceden türev kullanan tek algoritma olması verilebilir. Tüm algoritmalar için iterasyon sayısı 10 seçildiğinden ve birinci derecen türev kullanan algoritmalar, ikinci dereceden kullananlara göre daha yavaş yakınsadığından bu sonuç beklenen bir sonuçtur.

Parçalı eğitimin başarısını en çok arttırdığı algoritma da RP'dir. Çok başarılı bir algoritmanın başarısını arttırmak, başarısız birininkini arttırmaktan daha zordur. RP'nin başarısı düşük olduğu için başarısını arttırmak daha kolay olmuştur.

Algoritmaların performanslarının benzerliklerini bulmak için algoritmalar 11 veri kümesi üzerlerindeki başarılarına göre kümelenmiştir. 4 algoritmanın her birinin 1 toplu 12 parçalı eğitim versiyonu olmak üzere 13 versiyonu bulunmaktadır. Buna göre 52 algoritma, 11 boyutlu bir uzayda bir

nokta olarak düşünülmüş ve benzerlikleri görebilmek amacıyla hiyerarşik olarak kümelenmiştir. Buna göre elde edilen sonuç Şekil 10'da verilmiştir.



Şekil 10: Algoritmaların benzerlikleri

Şekil 10 incelendiğinde algoritmaların kabaca 3 gruba ayrıldıkları görülmektedir. Bunlar RP grubu, LM grubu ve SCG-BFG grubudur. Algoritmaların parçalı eğitim parametrelerine göre değil, öğrenme algoritmalarına göre kümelendikleri görülmektedir. Buna göre algoritma performanslarında, öğrenme algoritmalarının, parçalı eğitim parametrelerinden daha belirleyici olduğu söylenebilir.

5 Sonuçlar

Parçalı eğitimin, toplu eğitime göre başarı performansını incelediğimiz bu çalışmada, deneyler 4 farklı öğrenme algoritması, parça sayısı ve parçaların oluşturulma yöntemleri parametrelerine göre 11 veri kümesi üzerinde gerçekleştirilmiştir.

Karşıt durumlar olmasına rağmen genel olarak görülen sonuçlara göre:

- Parçalı eğitimin toplu eğitime göre daha başarılı olduğu
- Parçalı eğitimin tüm öğrenme algoritmalarını aynı ölçüde etkilemediği
- Parça sayısının optimal değerinin ve parçaların en uygun nasıl üretileceğinin öğrenme algoritmasına bağlı olduğu
- Algoritma performansını eğitim türünden daha çok öğrenme algoritmalarının belirlediği
- Öğrenme algoritmaları arasında en başarılının LM olduğu, en başarısızın RP olduğu
- Veri kümelerinin özellik ve örnek sayısı değerleriyle öğrenme algoritmaları arasında bir ilişkinin olmadığı
- Algoritma performanslarında, öğrenme algoritmalarının, parçalı eğitim parametrelerinden daha belirleyici olduğu görülmüştür.

Gelecek çalışma olarak parçalı eğitimin diğer öğrenme algoritmalarında başarıyı arttırırken LM'de neden etkisiz olduğunun teorik açıklamasının yapılması düşünülmektedir.

6 Kaynakça

[24] **LeCun Y., Bottou, L., Orr, G. ve Muller, K.**, 1998. "Efficient BackProp", in Orr, G. and Muller K. (Eds), *Neural Networks: Tricks of the trade*, Springer.

[25] **Wilson, D.R., Martinez, T.R.,** 2003, "The general inefficiency of batch training for gradient descent learning", *Neural Networks*, 16 (2003) 1429–1451, Elsevier.

[26] **Dekel, O., Bachrach, R.G., Shamir, O., Xiao, L.,** 2012, "Optimal Distributed Online Prediction Using Mini-Batches", *Journal of Machine Learning Research*, 13 (2012) 165-202.

[27] **Gimpel, K., Das, D., Smith, N.A.,** 2010, "Distributed Asynchronous Online Learning for Natural Language Processing", *CoNLL '10 Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, 213-222.

[28] **Zhao, K., Huang, L.,** 2013, "Minibatch and Parallelization for Online Large Margin

Structured Learning", Proceedings of NAACL-HLT 2013, 370–379.

[29] **Cotter, A., Shamir, O., Srebro, N., ve Sridharan, K.,** 2011, "Better Mini-Batch Algorithms via Accelerated Gradient Methods", Advances in Neural Information Processing Systems (NIPS).

[30] **Møller, M.F., 1993,** "A scaled conjugate gradient algorithm for fast supervised learning", NEURAL NETWORKS, 6(4), 525-533.

[31] **Boyd, S., Vandenberghe, L.,** 2004, Convex Optimization, Cambridge University Press.

[32] **Fletcher, R.,** 1987, Practical methods of optimization, New York: John Wiley & Sons.

[33] **Igel, C., Hüsken, M.,** 2003, "Empirical evaluation of the improved Rprop learning algorithm", Neurocomputing, 50(C), 105-123.

[34] http://wangwzhao.googlepages.com/

[35] **Blake, C.L., Merz, C.J.,** 1998, UCI Repository of Machine Learning Databases - www.ics.uci.edu/~mlearn/MLRepository.html