T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YAPAY SİNİR AĞLARININ EĞİTİMİNDE ÖRNEKLERİN ZORLUK SEVİYESİNE GÖRE SIRALANMASININ ETKİSİNİN İNCELENMESİ

MELİKE NUR MERMER

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
BİLGİSAYAR MÜHENDİSLİĞİ PROGRAMI

DANIŞMAN
DOÇ. DR. MEHMET FATİH AMASYALI

İSTANBUL, 2018

T.C.

YILDIZ TEKNİK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

YAPAY SİNİR AĞLARININ EĞİTİMİNDE ÖRNEKLERİN ZORLUK SEVİYESİNE GÖRE SIRALANMASININ ETKİSİNİN İNCELENMESİ

Melike Nur MERMER tarafından hazırlanan tez çalışması 28.03.2018 tarihinde aşağıdaki jüri tarafından Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Tez Danışmanı

Doç. Dr. Mehmet Fatih AMASYALI

Yıldız Teknik Üniversitesi

Jüri Üyeleri

Doç. Dr. Mehmet Fatih AMASYALI

Yıldız Teknik Üniversitesi _____

Doç. Dr. Sırma YAVUZ

Yıldız Teknik Üniversitesi _____

Dr. Öğr. Üyesi Tolga ENSARİ

İstanbul Üniversitesi _____

ÖNSÖZ

Tez çalışmam öncesinde ve süresince ufkumu açan düşünceleri ve yönlendirmeleri ile benim için her zaman bir ilham kaynağı olan değerli danışmanım Doç. Dr. Mehmet Fatih Amasyalı'ya sonsuz teşekkürlerimi sunuyorum.

Bu süreçte beni hep anlayışla karşılayan, motive eden ve desteklerini her daim yanımda hissettiğim kıymetli aileme ve arkadaşlarıma teşekkür ediyorum.

Çalışmamı ilkokul günlerimden bugüne kadar tüm öğrencilik hayatımda bana güzel örnek olan öğrenmeyi ve öğretmeyi sevdiren bütün hocalarıma ithaf ediyorum.

Şubat, 2018

Melike Nur MERMER

İÇİNDEKİLER

Sayfa

SİMGE LİSTESİ

$\nabla_x f(x)$	f(x) fonksiyonunun gradyanı
g	Gradyan
ϵ	Öğrenme katsayısı
θ	Hedef fonksiyonunun parametreleri
α	Momentum parametresi
ϑ	Başlangıç hızı
δ	Küçük bir sabit değer
r	Gradyan birikim değişkeni
ρ	Çürüme oranı
N	Toplam eğitim örneği sayısı
$\ell(\theta)$	Kayıp fonksiyonu
E[ℓ(θ)]	ℓ(θ)'nın beklenen değeri
K[ℓ(θ)]	ℓ(θ)'nın karmaşıklığı
$\mathrm{PDF}(\ell(\theta_B))$	θ_B noktasındaki olasılık yoğunluk fonksiyonu
Pr	Olasılık

KISALTMA LİSTESİ

ark.	Arkadaşları
ACL	Anti Curriculum Learning (Ters Planlı Öğrenme)
CL	Curriculum Learning (Planlı Öğrenme)
CNN	Convolutional Neural Network (Evrişimsel Sinir Ağı)
CV	Cross Validation (Çapraz doğrulama)
K-NN	K-Nearest Neighbour (K-en yakın komşu)
LSTM	Long Short-Term Memory (Uzun Kısa-Süreli Bellek)
MSE	Mean Squared Error (Ortalama karesel hata)
PDF	Probability Density Function (Olasılık yoğunluk fonksiyonu)
RNN	Recurrent Neural Network (Özyinelemeli Sinir Ağı)
ROGS	Random Ordered Growing Sets (Rastgele Düzenli Büyüyen Kümeler)
SGD	Stochastic Gradient Descent (Stokastik Gradyan Düşümü)
SGDM	Stochastic Gradient Descent with Momentum (Momentumlu Stokastik Gradyan Düşümü)
SPL	Self Paced Learning (Kendini Planlayan Öğrenme)
SPLD	Self Paced Learning with Diversity (Çeşitlilik içeren Kendini Planlayan Öğrenme)
SPLI	Self Paced Learning Inversed (Kendini Tersten Planlayan Öğrenme)
SVM	Support Vector Machine (Destek Vektör Makinesi)
YSA	Yapay Sinir Ağı

ŞEKİL LİSTESİ

Sayfa

ÇİZELGE LİSTESİ

Sayfa

ÖZET

YAPAY SİNİR AĞLARININ EĞİTİMİNDE ÖRNEKLERİN ZORLUK SEVİYESİNE GÖRE SIRALANMASININ ETKİSİNİN İNCELENMESİ

Melike Nur MERMER

Bilgisayar Mühendisliği Anabilim Dalı

Yüksek Lisans Tezi

Tez Danışmanı: Doç. Dr. Mehmet Fatih AMASYALI

Makine öğrenmesinde popüler konulardan olan Planlı Öğrenme (Curriculum Learning) ve Kendini Planlayan Öğrenme (Self paced Learning) eğitim örneklerinin zorluk seviyelerine göre sıralanarak öğreniciye verilmesini önermektedir. Bu konulardaki çalışmalar küçük bir eğitim kümesi ile başlayıp zorluk seviyesine göre yeni örnekler ekleyerek devam etmenin öğrenme performansını geliştirdiğini göstermektedir.

Literatürdeki çalışmalarda örneklerin kolaydan zora sıralanmasının yanı sıra zordan kolaya sıralanması ile de daha iyi performans elde edildiği görülmektedir. Bu tez çalışmasının 3. Bölümünde Planlı Öğrenme ve ters versiyonunun birçok uygulama alanına adapte edilmesi için zorluk seviyelerinin otomatik olarak belirlendiği bir yöntem üzerinde çalışılmıştır. Örnekler bu yöntem ile kolaydan zora ve zordan kolaya sıralanarak öğreniciye verilmiş ve elde edilen sonuçlar klasik eğitim metodu ile karşılaştırılmıştır. Karşılaştırmalar sonucunda uygulama alanlarının önemli bir kısmında sıralama yapılan yöntemlerin istatistiksel açıdan anlamlı ve daha başarılı bir performans gösterdiği görülmüştür.

Örneklerin kolaydan zora sıralanması ile daha iyi bir performans görülmesi beklenen bir durumken zordan kolaya sıralanması ile de iyi sonuçlar elde etmek şaşırtıcı bir durumdur. Bu nedenle Planlı Öğrenme ve ters versiyonunun her ikisinin de başarılı olmasının altında yatan sebepler üzerine araştırmalar yapılmıştır. Çalışmanın 4.

Bölümünde sıralama yapılan yöntemlerin daha başarılı olmasının aslında sıralama yapıyor olmalarından değil eğitim kümesinin aşamadan aşamaya büyüyor olmasından kaynaklandığı öne sürülmüştür. Bu nedenle örneklerin anlamlı bir sıralama ile değil de rastgele düzenli büyüyen kümeler halinde verilmesinin de öğrenme performansını artıracağı düşünülmüştür. Önerilen yöntemin teorik alt yapısı açıklanmış ve uygulama bölümünde önceki yöntemlerle karşılaştırılmıştır. Karşılaştırmalar sonucunda rastgele düzenli büyüyen kümeler yönteminin tüm örneklerin tek aşamada verildiği klasik eğitim metodundan daha başarılı Planlı ve Kendini Planlayan Öğrenme yöntemleri ile yakın bir performans gösterdiği görülmüştür.

Teorik ve deneysel çalışmalar neticesinde Planlı Öğrenme ve ters versiyonun ortak özelliği olan büyüyen kümelerle eğitim yönteminin optimizasyon sırasında daha iyi bir yerel minimum bulmayı sağlayan bir özelliğinin olduğu ve bu sebeple tek aşamada gerçekleştirilen eğitimden daha düşük hata oranlarının elde edebildiği sonucuna varılmıştır.

Anahtar Kelimeler: Yapay Sinir Ağları, Optimizasyon, Planlı Öğrenme, Kendini Planlayan Öğrenme, Büyüyen Kümelerle Eğitim.

ABSTRACT

A STUDY ABOUT THE EFFECTIVENESS OF ORDERING THE SAMPLES BY DIFFICULTY LEVELS IN THE TRAINING OF ARTIFICIAL NEURAL NETWORKS

Melike Nur MERMER

Department of Computer Engineering

MSc. Thesis

Advisor: Assoc. Prof. Dr. Mehmet Fatih AMASYALI

Curriculum Learning and Self-paced Learning are popular topics in machine learning which suggests to put the training samples in an order related with their difficulty levels. Studies about these topics show that starting with a small training set and continue to add new samples according to the difficulty levels improves the performance of the learner.

It is seen that the learner has better performance with hard-to-easy ordering as well as easy-to-hard. Chapter 3 is about a method to automatically determine the difficulty levels of the samples to see the performance of the Curriculum and its reverse version on many application areas. Samples have been ordered from easy-to-hard and hard-to-easy with the proposed method and compared with classical training and each other. According to the results, methods which have ordered the training samples obtained statistically significant lower error rates.

While it is expected to get better results by ordering the samples from easy-to-hard it is surprising to get also better results when ordering the samples from hard-to-easy. Because of this, the underlying reasons for the success of both the Curriculum Learning and its reverse version have been researched. It is suggested in Chapter 4 that the success of these methods with ordering are not due to the fact that they have ordering but the training set is growing from stage to stage. For this reason, it is thought that giving the samples in randomly growing groups rather than with a meaningful order will

increase the learning performance. The theoretical perspective of the proposed method is explained and the proposed method is compared with previous methods in the experiments. As a result of the comparisons, it is seen that the random ordered growing sets method performed better than the classical method which all the samples were given in one step and closely with the successful Curriculum and Self-Paced Learning methods.

As a result of the theoretical and experimental studies, it has been concluded that training with growing sets method, which is a common feature of Curriculum Learning and its reverse version, has a feature that allows to find a better local minimum during optimization and therefore can achieve lower error rates than training in a single stage.

Keywords: Artificial Neural Networks, Optimization, Curriculum Learning, Self-Paced Learning, Training with Growing Sets.

BÖLÜM 1

GİRİŞ

1.1 Literatür Özeti

Makine öğrenmesi yöntemleri sınıfları bilinen eğitim örnekleri ile öğrenme fonksiyonlarını ayarlayarak sınıfları bilinmeyen test örneklerini en başarılı şekilde tahmin etmeye çalışırlar. Eğitim sırasında eğitim örneklerinin veya aynı dağılımdan bir başka küme olan validasyon örneklerinin sınıflarının doğru tahmin edilme oranı ile öğrenme derecesini ayarlarlar. Sınıfı bilinen örneklerde minimum hata yapılan noktada eğitim sona erer ve elde edilen parametreler kullanılarak test örnekleri tahmin edilir.

Yapay Sinir Ağları çok sayıda parametre içermesinden dolayı parametre uzayında minimum hata yapılan noktanın nerede olduğu çoğu zaman kestirilememektedir. Bu gibi durumlarda hatanın belirli bir sınırın altına düşmesi dağılımın iyi öğrenilmiş olmasını göstermek için yeterli olarak kabul edilir ve bunu sağlayan bir yerel minimum (local minimum) noktasında eğitim sona erer [1]. Sözlükte [2] en uygun duruma getirme karşılığına gelen optimizasyon, öğrenme fonksiyonun en uygun durumunu bulmaya çalışır. Yapay Sinir Ağları gibi çok parametreli yöntemlerin öğrenme fonksiyonlarında birçok yerel minimum ve eyer noktası (saddle point) bulunduğu için dışbükey olmayan optimizasyon (non-convex optimization) uygulanır. Gradyan düşümü algoritması [3] öğrenme fonksiyonun türevini kullanarak hatanın çevresine göre en düşük değerde olduğu noktayı bulur ancak bu nokta kendi çevresine göre düşük olmasına rağmen fonksiyonun geneline göre hatanın minimum olmadığı bir nokta olabilir. Bu nedenle optimizasyonu daha da iyileştirmek için momentum ilavesi [4,5] ve adaptif öğrenme katsayılı algoritmalar [6-8] önerilmiştir. Bunlar dışında öğrenme fonksiyonunun ikinci

türevini kullanan daha gelişmiş algoritmalar da mevcuttur. Bu algoritmaların eğitim sürecinde farklı stratejilerin [9,10] kullanılması da optimizasyon sağlamaktadır. Bu stratejilerden biri olan Planlı Öğrenme (Curriculum Learning) [11] eğitim örneklerinin zorluk seviyesine göre anlamlı bir sıralama ile öğreniciye verilmesini önermektedir. Konu ile ilgili eski çalışmalardan [12-14] yola çıkılarak ortaya atılan makinelerin öğrenme sürecini insanların öğrenme sürecine benzer şekilde düzenleme fikri kısa zamanda popüler olmuş ve konu hakkında birçok araştırma yapılmıştır. Ancak Planlı Öğrenmenin ön koşulu olan örneklerin zorluk seviyesinin bilinmesi bu stratejinin uygulanabilirliğini oldukça kısıtlamaktadır. Eğitim sırasında öğrenilecek örnekleri öğrenicinin kendisinin belirlediği Kendini Planlayan Öğrenme [15] yöntemi bu amaçla literatürde çok kullanılan yöntemlerden biri olmuştur.

Literatürde örneklerin kolaydan zora dışında farklı şekilde planlanması ile de başarılı sonuçlar elde edildiği görülmüştür [16-19]. Bunun yanı sıra Planlı Öğrenmenin matematiksel açıklamasını arayan birkaç çalışma [24,25] da mevcuttur. Öğrenicinin emin olmadığı zor örneklere öncelik verildiğinde iyi sonuçlar elde edilebildiği de görülmektedir [26,27]. Bu durum Planlı öğrenme ve ters versiyonunun ortak bir özelliği nedeniyle her iki stratejinin de iyi olduğu fikrini akla getirmektedir.

1.2 Tezin Amacı

Yapay Sinir Ağlarının optimizasyonu için geliştirilen stratejilerden olan Planlı Öğrenme ve ters versiyonunun birçok alanda uygulanabilmesini sağlamak için zorluk seviyesi belirleme aşamasını otomatik hale getiren bir yöntem geliştirilmek istenmiştir. Bu sayede Planlı ve Ters Planlı öğrenme yaklaşımları birçok uygulama alanında denenip klasik yöntemle karşılaştırılacak ve genel geçer bir optimizasyon sağlayıp sağlamadığı anlaşılabilecektir.

Planlı ve Ters Planlı Öğrenme yaklaşımlarının birçok alanda klasik yöntemden daha iyi olduğunun anlaşılması üzerine hem kolaydan zora hem de zordan kolaya sıralama ile eğitilen ağların ortak bir özelliğinin bu iyileşmeyi sağladığı düşünülmüştür. Araştırmalar sonucunda bu yöntemlerin ortak özelliğinin örneklerin anlamlı bir sıralama ile verilmesi değil de büyüyen kümeler halinde verilmesi olduğu ve bu durumun iyileştirme sağlıyor olabileceği fark edilmiştir. Bunun üzerine örneklere zorluk seviyelerine göre anlamlı bir

sıralama verilmeksizin sadece Rastgele Düzenli Büyüyen Kümeler halinde eğitimin gerçekleştirilmesi ile de iyi sonuçlar elde edilebileceği öne sürülebilir. Önerilen yöntem Planlı ve Kendini Planlayan Öğrenme yöntemleri ve ters versiyonları ile karşılaştırılarak örneklere anlamlı bir sıralama vermenin gerekli olup olmadığı görülecektir. Dahası büyüyen kümeler yöntemi teorik açıdan da ele alınacak ve nasıl bir optimizasyon sağladığı matematiksel olarak açıklanacaktır.

Tez çalışması şu şekilde organize edilmiştir: 2. Bölümde Yapay Sinir Ağlarının optimizasyonu hakkında temel bilgiler verilecek ve çok sık kullanılan optimizasyon algoritmalarına yer verilecektir. 3. Bölümde Planlı ve Ters Planlı Öğrenme stratejilerinin birçok alana uygulanıp klasik yöntemle karşılaştırıldığı bir çalışma açıklanacak ve elde edilen sonuçlar yorumlanarak yöntemlerin neden ve nasıl bu sonuçları elde ettiği incelenecektir. 4. Bölüm Rastgele Düzenli Büyüyen Kümeler yönteminin çıkış noktasından başlamakta ve bu yöntemin teorik açıklaması ile devam etmektedir. Bu bölümün uygulama kısmında büyüyen kümeler yöntemi klasik, Planlı, Ters Planlı, Kendini Planlayan ve Kendini Tersten Planlayan Öğrenme yöntemleri ile birçok uygulama alanında karşılaştırılacak ve örneklerin anlamlı şekilde sıralanmasından kaynaklanan bir optimizasyonun olup olmadığı anlaşılacaktır. Rastgele Düzenli Büyüyen Kümeler yöntemi kullanılarak Planlı Öğrenme ve onun başka versiyonlarına yakın bir performans elde edilirse zorluk seviyesi belirleme aşamasını ortadan kaldıran yeni bir optimizasyon yöntemi ortaya çıkmış olacaktır.

1.3 Hipotez

Tez çalışması Planlı Öğrenme yaklaşımını birçok alanda uygulanabilir hale getirme amacıyla başlamış ancak bu yaklaşımın ters versiyonunun da oldukça başarılı olduğu görüldükten sonra her iki versiyonun iyileştirmeyi sağlayan ortak özelliğinin eğitimin büyüyen kümelerle gerçekleştirilmesi olduğu öne sürülmüştür. Çalışmanın başından itibaren yürütülen hipotezler aşağıdaki maddelerde sırası ile verilmektedir.

- Planlı Öğrenme ve ters versiyonu olan Ters Planlı Öğrenme birçok uygulama alanında klasik yöntemden daha iyi sonuçlar elde eder.

- Planlı ve Ters Planlı Öğrenmenin ikisinin de iyi olmasının sebebi aşamalar sırasında parametrelerin değişiminde gerçekleşen ortak bir özelliktir.
- Planlı ve Ters Planlı Öğrenmenin ortak özelliği her aşamanın başında yeni örneklerin eklenmesi ile parametre uzayında büyük bir sıçrama gerçekleşmesidir ve aslında örneklerin anlamlı bir sıralama ile ağa verilmesi ile ilgili değildir.
- Zorluk seviyesi ile ilgili bir sıralama olmaksızın rastgele seçilen örneklerle eğitim kümesini büyütmek bir optimizasyon sağlar.
- Rastgele düzenli büyüyen kümeler yönteminin neden optimizasyon sağladığı ile ilgili matematiksel bir açıklaması vardır ve ispat edilebilir.

BÖLÜM 2

YAPAY SİNİR AĞLARININ OPTİMİZASYONU

Makine öğrenmesi algoritmaları bilinen girdi ve çıktılar ile çok sayıda parametreyi optimize ederek çıktısı bilinmeyen girdiler için en iyi tahmini yapmaya çalışır. Bu algoritmalardan Yapay Sinir Ağları ise birçok girdinin birbirine göre çıktıyı etkileyebildiği bir yöntemdir ve dışbükey olmayan bir optimizasyon fonksiyonuna sahiptir. Bu bölümde Yapay Sinir Ağlarının hedef fonksiyonunun optimizasyonunda kullanılan terimler ve optimizasyon algoritmaları açıklanacaktır. Verilen algoritmalar Goodfellow ve ark. yazmış olduğu Derin Öğrenme kitabının [1] 8. Bölümünde yer almaktadır.

2.1 Optimizasyon Nedir?

Sözlükteki anlamı en uygun duruma getirme [2] olan optimizasyon bir sistemi en verimli hale getirmek için kaynakların en uygun şekilde kullanılmasıdır. Matematiksel bir terim olarak her alanda yeri olan optimizasyon, makine öğrenmesi algoritmalarında da mevcut girdilerle modeli en uygun duruma getirmeyi amaçlamaktadır. Bu işlem genellikle hata miktarının en aza düşürülmesi ile gerçekleştirilir. Makine öğrenmesinde optimizasyon, f(x) fonksiyonunu minimize eden x değerini bulmaktır. Minimize edilmek istenen f(x) fonksiyonuna hedef fonksiyonu (objective function) denir. Bu çalışmada hata fonksiyonu (error function), kayıp fonksiyonu (loss function) ve maliyet fonksiyonu (cost function) terimleri de aynı amaçla kullanılmıştır. f(x) fonksiyonunu minimize eden x değerini x* ile gösterirsek bu fonksiyonun optimizasyonu Denklem 2.1'deki gibi ifade edilir [1].

$$x^* = argmin\, f(x) \tag{2.1}$$

Destek Vektör Makineleri (Support Vector Machines, SVM) gibi makine öğrenmesi metotlarının parametreleri tek bir minimum noktasına sahip dışbükey fonksiyonlar üzerinde optimize edilir. Dışbükey optimizasyona (convex optimization) sahip algoritmalar için mevcut parametrelerin en uygun değerlerini bulmak kolaydır. Ancak Yapay Sinir Ağları gibi iteratif algoritmaların optimizasyonu dışbükey değildir. Yapay Sinir Ağlarında optimize edilmek istenen ağırlıklar ve hedef fonksiyonu her iterasyonda güncellenmektedir. Bu nedenle birden fazla minimum noktasının bulunduğu bir fonksiyon üreten Yapay Sinir Ağları dışbükey olmayan optimizasyon (non-convex optimization) yöntemleri ile en uygun duruma getirilir. Bu yöntemlerden en temeli parametrelerin türevlerinin kullanıldığı gradyan tabanlı optimizasyondur.

2.2 Gradyan Tabanlı Optimizasyon

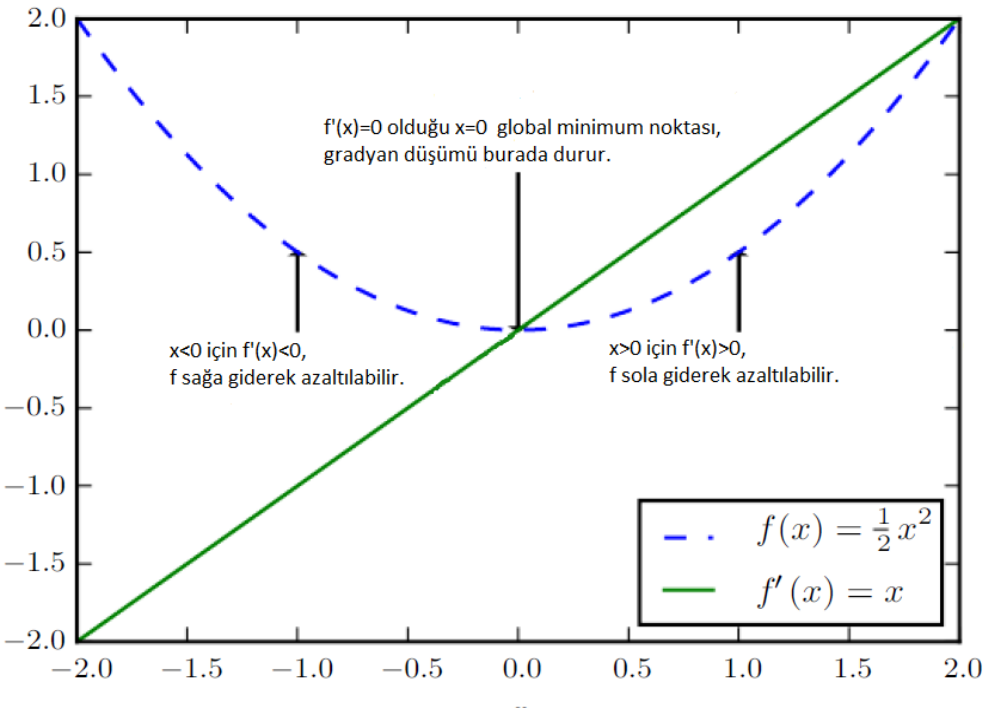
x ve y gerçel sayılar olmak üzere y=f(x) şeklinde bir fonksiyon olduğunu varsayalım. Bu fonksiyonun türevi f'(x) veya $\frac{dy}{dx}$ şeklinde gösterilir. Türev yani f'(x), f(x) fonksiyonunun x noktasındaki eğimini vermektedir. Başka bir deyişle girişteki küçük bir değişimin ilgili çıkışa nasıl yansıtılacağını belirtir [1].

$$f(x+\epsilon) \approx f(x) + \epsilon f'(x). \tag{2.2}$$

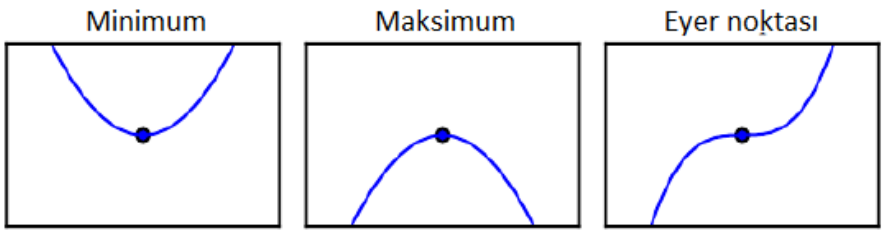
Türev burada bir fonksiyonu minimize etmek için kullanılabilir çünkü y'de küçük bir geliştirme yapmak için x'in nasıl değiştirileceği hakkında bilgi vermektedir. Böylece türevin işaretinin tersi yönünde x'i küçük adımlarla hareket ettirerek f(x) 'i minimize etmek mümkündür. Bu tekniğe gradyan düşümü (gradient descent) [3] adı verilir. Şekil 2.1'de bir fonksiyonu minimuma götürmek için gradyan düşümü algoritmasında türevinin nasıl kullanıldığını gösteren basit bir örnek verilmiştir.

f'(x)=0 olduğu noktalarda hedef fonksiyonunun türevi hangi yöne harekete devam edileceğine dair bir bilgi verememektedir. Bu noktalara kritik nokta (critical point) denir. f(x) 'in tüm komşularından daha düşük olduğu x noktasına ise yerel minimum (local minimum) noktası denir. Benzer şekilde f(x) 'in tüm komşularından daha yüksek olduğu x noktasına ise yerel maksimum (local maximum) noktası denir. Bazı kritik noktalar ise ne minimum ne de maksimum noktadır. Kendisinden hem daha yüksek hem de daha

düşük komşuya sahip kritik noktalara eyer noktası (saddle point) denir [1]. Bahsedilen kritik noktalar Şekil 2.2'de gösterilmiştir.



Şekil 2.1 Gradyan düşümü [1]

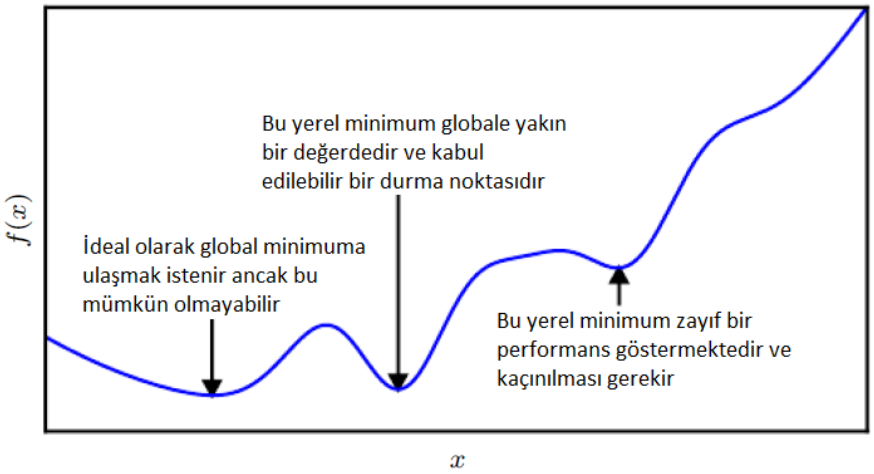


Şekil 2.2 Kritik noktalar [1]

f(x)'in mutlak en düşük değerini aldığı nokta global minimum noktasıdır. Yapay Sinir Ağlarının hata fonksiyonları optimal olmayan birden çok yerel minimum noktası ve eyer noktası içerir. Bu fonksiyonlar üzerinde gerçekleştirilen optimizasyonun amacı gerektiği kadar minimum olan bir noktayı bulmaktır. Şekil 2.3 'te yaklaşık minimizasyon (approximate minimization) örneği verilmiştir [1].

Çok girdili fonksiyonlarda kısmi türev kuralları geçerli olmaktadır. Kısmi türev $\frac{\partial}{\partial x_i} f(x)$ sadece x_i değişkenine bağlı olarak f 'in x noktasında nasıl değiştiğini ölçer. Gradyan ise

tüm kısmi türevleri içeren vektör olarak ifade edilir ve $\nabla_x f(x)$ ile gösterilir. Çok boyutlarda kritik noktalar gradyanın tüm elemanlarının sıfır olduğu noktalardır [1].



Şekil 2.3 Yaklaşık minimizasyon [1]

2.3 Optimizasyon ve Öğrenme

Çoğu makine öğrenmesi yönteminin performansı, test kümesine göre tanımlanan bir performans ölçütüne göre hesaplanır. Optimizasyonun makine öğrenmesinde kullanılması ise farklı bir maliyet fonksiyonu olan J(θ)'nın minimize edilmesi ile dolaylı olarak P'nin en uygun duruma getirilmesidir. Burada sadece kendi başına J'nin minimize edilmek istendiği genel optimizasyondan daha farklı bir durum söz konusudur. Maliyet fonksiyonu, eğitim kümesi üzerindeki ortalama hata miktarı olacak şekilde Denklem 2.3'deki gibi yazılabilir [1].

$$J(\theta) = E_{(x,y)\sim \dot{p}_{veri}} L(f(x;\theta), y) \tag{2.3}$$

L, burada her örneğin kayıp fonksiyonudur. f(x;θ), x girdisine karşılık tahmin edilen çıktı ve $\dot{p}_{veri}$ denemedeki dağılımdır. Etiketli öğrenmede (supervised learning) y hedef etiketidir. Denklem 2.3 eğitim kümesine göre hedef fonksiyonunu tanımlamaktadır. Ancak genellikle eğitim kümesinden alınan alt dağılımlar üzerinde hedef fonksiyonu optimize edilir [1].

$$J^*(\theta) = E_{(x,y)\sim p_{veri}} L(f(x;\theta), y) \tag{2.4}$$

Denklem 2.4'te p_{veri} eğitim kümesinden alınan alt dağılımı ifade eder.

2.4 Temel Optimizasyon Algoritmaları

Bölüm 2.2.'de gradyan düşümü ile ilgili bilgi verilmiştir. Bu bölümde ise en çok kullanılan optimizasyon algoritması olan Stokastik Gradyan Düşümü ve onun üzerine geliştirilen yöntemlerden bahsedilecektir.

2.4.1 Stokastik Gradyan Düşümü

Eğitim kümesinden alınan m elemanlı küçük bir grubun (mini-batch) ortalama gradyanı ile tahmini gradyanı elde etmek mümkündür [1]. Algoritma 2.1'de Stokastik Gradyan Düşümü (Stochastic Gradient Descent, SGD) sırasında izlenen yol verilmiştir.

Algoritma 2.1 Stokastik Gradyan Düşümü (SGD)

1: Öğrenme katsayısı ϵ

2: Başlangıç parametresi θ

3: **döngü** durma koşulu sağlanmadı **ise**

4: Eğitim kümesinden m örnekli küçük bir grup $\{x^{(1)}, x^{(2)},., x^{(m)}\}$ ve ilgili etiketler $y^{(i)}$ alınır.

5: Gradyan tahmini hesaplanır: $\hat{g} \leftarrow \frac{1}{m} \nabla_\theta \sum_i L(f(x^{(i)}; \theta), y^{(i)})$

6: Parametre güncellenir: $\theta \leftarrow \theta - \epsilon \hat{g}$

7: **döngü bitti**

2.4.2 Momentumlu Stokastik Gradyan Düşümü

Momentum algoritması önceki gradyanların hareketlerinin ortalamasını üstel fonksiyonla ayarlayarak bir yön bulur ve o yönde hareket etmeye devam eder [4]. Momentumlu Stokastik Gradyan Düşümü Algoritması (Stochastic Gradient Descent with Momentum, SGDM) Algoritma 2.2'de verilmiştir.

Algoritma 2.2 Momentumlu Stokastik Gradyan Düşümü (SGDM)

1: Öğrenme katsayısı ϵ, momentum parametresi α.

2: Başlangıç parametresi θ, başlangıç hızı ϑ.

3: **döngü** durma koşulu sağlanmadı **ise**

4: Eğitim kümesinden m örnekli küçük bir grup $\{x^{(1)}, x^{(2)},., x^{(m)}\}$ ve ilgili etiketler $y^{(i)}$ alınır.

5: Gradyan tahmini hesaplanır: $g \leftarrow \frac{1}{m}\nabla_\theta \sum_i L(f(x^{(i)}; \theta), y^{(i)})$

6: Hız güncellemesi hesaplanır: $\vartheta \leftarrow \alpha\vartheta - \epsilon g$

7: Parametre güncellenir: $\theta \leftarrow \theta + \vartheta$

8: **döngü bitti**

2.4.3 Nesterov Momentumlu Stokastik Gradyan Düşümü

Nesterov momentum ile standart momentum arasındaki fark gradyanın hesaplandığı yerdir. Nesterov momentumlu yöntemde geçerli hız uygulandıktan sonra gradyan hesaplanır [5]. Nesterov Momentumlu SGD Algoritmasında takip edilen adımlar Algoritma 2.3'te verilmiştir.

Algoritma 2.3 Nesterov Momentumlu Stokastik Gradyan Düşümü

1: Öğrenme katsayısı ϵ, momentum parametresi α.

2: Başlangıç parametresi θ, başlangıç hızı ϑ.

3: **döngü** durma koşulu sağlanmadı **ise**

4: Eğitim kümesinden m örnekli küçük bir grup $\{x^{(1)}, x^{(2)},., x^{(m)}\}$ ve ilgili etiketler $y^{(i)}$ alınır.

5: Parametre geçici olarak güncellenir: $\theta^* \leftarrow \theta + \alpha\vartheta$

5: Gradyan hesaplanır (geçici noktadaki): $g \leftarrow \frac{1}{m}\nabla_{\theta^*} \sum_i L(f(x^{(i)}; \theta^*), y^{(i)})$

6: Hız güncellemesi hesaplanır: $\vartheta \leftarrow \alpha\vartheta - \epsilon g$

7: Parametre güncellenir: $\theta \leftarrow \theta + \vartheta$

8: **döngü bitti**

2.5 Adaptif Öğrenme Katsayılı Algoritmalar

Yapay Sinir Ağlarında öğrenme katsayısı, modelin performansında çok önemli bir etkiye sahip olduğu için ayarlanması en zor hiperparametrelerdendir. Maliyet fonksiyonu parametre uzayındaki bazı yönlere karşı yüksek hassasiyete sahipken bazı yönlere karşı duyarsız olabilmektedir. Hassasiyet doğrultularının eksene hizalı olduğu düşünülürse her

parametre için ayrı bir öğrenme katsayısı belirlemek ve öğrenme sürecinde bu katsayıları otomatik olarak uyarlamak mantıklıdır [1].

Bu bölümde model parametrelerinin öğrenme katsayılarının öğrenme süreci boyunca uyarlandığı bazı yöntemler açıklanacaktır.

2.5.1 AdaGrad

Algoritma 2.4'te verilen AdaGrad algoritması [6] model parametrelerinin tümünü bireysel şekilde tüm parametrelerin eski değerlerinin karelerinin toplamının karekökü ile ters orantılı olarak ölçeklendirerek öğrenme katsayısını uyarlar.

Algoritma 2.4 AdaGrad Algoritması

1: Global öğrenme katsayısı ϵ

2: Başlangıç parametresi θ

3: δ, sayısal tutarlılık için 10^{-7} gibi küçük bir sabit değer

4: Gradyan birikim değişkeni r=0 olarak başlatılır.

5: **döngü** durma koşulu sağlanmadı **ise**

6: Eğitim kümesinden m örnekli küçük bir grup $\{x^{(1)}, x^{(2)},., x^{(m)}\}$ ve ilgili etiketler $y^{(i)}$ alınır.

7: Gradyan hesaplanır: $g \leftarrow \frac{1}{m}\nabla_\theta \sum_i L(f(x^{(i)};\theta), y^{(i)})$

8: Gradyanların kareleri toplanır: $r \leftarrow r + g \odot g$

6: Güncelleme hesaplanır: $\Delta\theta \leftarrow -\frac{\epsilon}{\delta+\sqrt{r}} \odot g$ (Bölme ve karekök işlemleri eleman başına uygulanır.)

7: Parametre güncellenir: $\theta \leftarrow \theta + \Delta\theta$

8: **döngü bitti**

2.5.2 RMSProp

RMSProp algoritmasında [7] dışbükey olmayan durumda daha iyi performans göstermesi için AdaGrad algoritması üzerinde gradyan birikimini üstel olarak ağırlıklandırılmış hareket ortalamasına dönüştürecek şekilde bir değişiklik yapılmıştır. RMSProp Algoritması, Algoritma 2.5'te verilmiştir.

Algoritma 2.5 RMSProp Algoritması

1: Global öğrenme katsayısı ϵ, çürüme oranı ρ.

2: Başlangıç parametresi θ

3: δ, küçük sayılara bölme işleminde sayısal tutarlılık için 10^{-6} gibi küçük bir sabit değer.

4: Birikim değişkenleri r=0 olarak başlatılır

5: **döngü** durma koşulu sağlanmadı **ise**

6: Eğitim kümesinden m örnekli küçük bir grup $\{x^{(1)}, x^{(2)},., x^{(m)}\}$ ve ilgili etiketler $y^{(i)}$ alınır.

7: Gradyan hesaplanır: g $\leftarrow \frac{1}{m} \nabla_\theta \sum_i L(f(x^{(i)}; \theta), y^{(i)})$

8: Gradyanların kareleri toplanır: $r \leftarrow \rho r + (1-\rho)$ g⊙g

6: Güncellemeyi hesapla: $\Delta\theta = -\frac{\epsilon}{\sqrt{\delta+r}} \odot g$. ($\frac{1}{\sqrt{\delta+r}}$ eleman başına)

7: Parametre güncellenir: $\theta \leftarrow \theta + \Delta\theta$

8: **döngü bitti**

2.5.3 Adam

RMSProp ve momentumun bazı küçük farklılıklarla birleştirildiği Adam algoritması [8], Algoritma 2.6'da verilmiştir.

Algoritma 2.6 Adam Algoritması

1: Adım miktarı ϵ (varsayılan 0,001 önerilir)

2: Moment tahmini için [0,1) aralığında üstel çürüme oranları ρ_1 ve ρ_2 (varsayılan değerler sırasıyla 0,9 ve 0,999)

3: δ, sayısal tutarlılık için kullanılan 10^{-8} gibi küçük bir sabit değer

4: Başlangıç parametresi θ

5: 1. ve 2. moment değişkenleri s=0, r=0 olarak başlatılır.

6: Zaman adımı t=0 olarak başlatılır.

7: **döngü** durma koşulu sağlanmadı **ise**

8: Eğitim kümesinden m örnekli küçük bir grup $\{x^{(1)}, x^{(2)},., x^{(m)}\}$ ve ilgili etiketler $y^{(i)}$ alınır.

9: Gradyan hesaplanır: g $\leftarrow \frac{1}{m} \nabla_\theta \sum_i L(f(x^{(i)}; \theta), y^{(i)})$

10: $t \leftarrow t + 1$

11: Ağırlıklandırılmış 1. moment tahmini güncellenir: $s \leftarrow \rho_1 \mathrm{s} + (1-\rho_1)\mathrm{g}$

12: Ağırlıklandırılmış 2. moment tahmini güncellenir: $r \leftarrow \rho_2 \mathrm{r} + (1-\rho_r)\ \mathrm{g} \odot \mathrm{g}$

13: 1. momentteki sapma düzeltilir: $\acute{\mathrm{s}} \leftarrow \frac{s}{1-\rho_1^t}$

14: 2. momentteki sapma düzeltilir: $\acute{\mathrm{r}} \leftarrow \frac{r}{1-\rho_2^t}$

15: Güncelleme hesaplanır: $\Delta\theta = -\epsilon \frac{\acute{s}}{\sqrt{\acute{r}}+\delta} \odot g$ (işlemler her eleman başına yapılır.)

16: Parametre güncellenir: $\theta \leftarrow \theta + \Delta\theta$

17: **döngü bitti**

2.6 Yaklaşık İkinci Derece Yöntemler

İkinci derece yöntemler, birinci derece yöntemlerden farklı olarak optimizasyonu geliştirmek için ikinci türevleri kullanırlar. Yapay sinir ağlarının optimizasyonunda en çok kullanılan ikinci derece optimizasyon yöntemleri bu bölümde açıklanacaktır.

2.6.1 Newton Yöntemi

Newton yöntemi J(θ) maliyet fonksiyonunun θ_0 yakınlarında ikinci dereceden Taylor serisi açılımında yüksek dereceli türevlerin ihmal edilmesine dayanan bir optimizasyon yöntemidir [1]. Bu yöntemin uygulanması sırasında izlenen yol Algoritma 2.7'de verilmiştir.

Algoritma 2.7 Newton Yöntemi

1: Başlangıç parametresi θ

2: m örnekli eğitim kümesi alınır.

3: **döngü** durma koşulu sağlanmadı **ise**

4: Gradyan hesaplanır: $g \leftarrow \frac{1}{m} \nabla_\theta \sum_i L(f(x^{(i)};\theta), y^{(i)})$

4: Hessian hesaplanır: $H \leftarrow \frac{1}{m} \nabla_\theta^2 \sum_i L(f(x^{(i)};\theta), y^{(i)})$

6: Hessian'ın tersi hesaplanır: H^{-1}

7: Güncelleme hesaplanır: $\Delta\theta = -H^{-1}g$

8: Parametre güncellenir: $\theta = \theta + \Delta\theta$

9: **döngü bitti**

2.6.2 Eşlenik Gradyanlar

Eşlenik gradyanlar (conjugate gradients), iteratif şekilde eşlenik doğrultularda düşüş göstererek Hessian'ın tersini hesaplamaktan kaçınmayı sağlayan bir yöntemdir. Eşlenik gradyanlar yönteminde önceki arama yönünün eşleniği olan bir arama yönü bulunmaya çalışılır [1]. Bu yöntem Algoritma 2.8'de verilmiştir.

Algoritma 2.8 Eşlenik Gradyanlar

1: Başlangıç parametreleri θ

2: m örnekli eğitim kümesi alınır.

3: Başlangıç değeri $\rho_0 = 0$

4: Başlangıç değeri $g_0 = 0$

5: Başlangıç değeri $t = 1$

6: **döngü** durma koşulu sağlanmadı **ise**

7: Gradyanın başlangıç değeri $g_t = 0$

8: Gradyanı hesapla: $g_t \leftarrow \frac{1}{m} \nabla_\theta \sum_i L(f(x^{(i)}; \theta), y^{(i)})$

9: Polak-Ribiere hesaplanır: $\beta_t = \frac{(g_t - g_{t-1})^T g_t}{g_{t-1}^T g_{t-1}}$

(Lineer olmayan eşlenik gradyan: örneğin t, k=5 gibi bir sabitin katları ise isteğe bağlı olarak β_t sıfırlanır.)

10: Arama yönünü hesapla: $\rho_t = -g_t + \beta_t \rho_{t-1}$

11: Doğrusal arama: $\epsilon^* = argmin_\epsilon \frac{1}{m} \nabla_\theta^2 \sum_{i=1}^m L(f(x^{(i)}; \theta_t + \epsilon \rho_t), y^{(i)})$

(Tam olarak kuadratik bir maliyet fonksiyonunda, ϵ^* aramak yerine analitik çözümle bulunur.)

12: Parametre güncellenir: $\theta_{t+1} = \theta_t + \epsilon^* \rho_t$

13: $t \leftarrow t + 1$

14: **döngü bitti**

2.6.3 BFGS

Broyden-Fletcher-Goldfarb-Shanno (BFGS) algoritması Newton yönteminin bazı avantajlarını hesaplama maliyetine gerek olmadan elde etmeyi sağlar. Bu bağlamda, BFGS eşlenik gradyanlar yöntemine benzer. Ancak Newton'un güncelleştirme yöntemine daha doğrudan bir yaklaşım getirmektedir [1].

Newton'un güncellemesinin uygulanmasında hesaplamada ilk zorluk Hessian'ın tersinin bulunmasıdır. Quasi-Newton yöntemlerinin benimsendiği yaklaşımlar (en bilineni BFGS algoritması) Hessian'ın tersini bir matris ile tahmin eder. Daha iyi tahminler elde edebilmek için bu matris için düşük dereceli güncellemelerle iteratif olarak işlenir [1].

Eşlenik gradyanlar yöntemindeki gibi BFGS algoritmasında da ikinci derecede türevlerdeki bilgileri (yön bilgisi) almak için bir dizi arama gerçekleştirilir. Eşlenik gradyanların aksine bu yaklaşımın başarısı aramada gerçek minimuma en yakın noktayı bulmuş olmaya büyük oranda bağlı değildir. Bu sayede BFGS arama sırasında daha az zaman harcama avantajına sahiptir [1].

2.7 Optimizasyon Stratejileri

Birçok optimizasyon tekniği tam bir algoritma olmamakla birlikte algoritmaları daha verimli hale getirmek için birçok algoritmaya uygulanabilen stratejiler içerir. Yeniden parametrelendirme (reparametrization), katmanlar arasındaki güncellemeleri koordine etme sorununu önemli ölçüde azaltır. Toplu normalleştirme (batch normalization) [9], Yapay Sinir Ağlarını yeniden parametrelendirmek için bir yol önermektedir.

Bazı durumlarda, optimizasyon problemini ayrı parçalar halinde ele alarak hızlı bir şekilde çözmek mümkün olabilir. f(x) 'i önce tek bir x_i değişkenine göre, daha sonra başka bir x_j değişkenine göre minimize edip bu tüm değişkenler boyunca art arda tekrarlı şekilde devam edilirse bir yerel minimuma ulaşılacağı düşünülür. Bu uygulama koordinat düşümü (coordinate descent) olarak bilinir, çünkü her seferinde bir koordinat optimize edilmektedir [1].

Polyak ortalama (Polyak averaging) [10] bir optimizasyon algoritması tarafından parametre uzayı boyunca ziyaret edilen yörüngedeki birçok noktanın ortalamasını

almaya dayanan bir stratejidir. Bazı stratejiler ise istenen modeli istenilen görevi yerine getirmek üzere eğitmeden önce basit görevler üzerinde basit modeller yetiştirmeyi içermektedir. Bu stratejiler topluca ön eğitim (pretraining) adıyla bilinir.

Sürdürme yöntemleri (continuation methods), yerel optimizasyonun zamanının çoğunu parametre uzayının daha iyi bölgelerinde harcamasını sağlamak için başlangıç noktalarını seçerek optimizasyonu kolaylaştırabilen bir strateji ailesidir [1]. Planlı Öğrenme (Curriculum Learning) [11] de bir çeşit sürdürme yöntemi olarak sunulmuş bir optimizasyon stratejisidir. Planlı Öğrenme önce basit kavramları öğrenerek eğitime başlamayı ve basit kavramlardan sonra karmaşık kavramların öğrenilmesini sağlayacak şekilde öğrenme sürecini planlama fikrine dayanmaktadır.

BÖLÜM 3

PLANLI ÖĞRENME

2009 yılında Bengio ve ark. tarafından ortaya koyulan Planlı Öğrenme stratejisinde [11] eğitim örneklerinin öğreniciye belli bir sıralama ile verilmesinin optimizasyon sırasında daha iyi bir yerel minimum bulmayı sağladığı öne sürülmektedir.

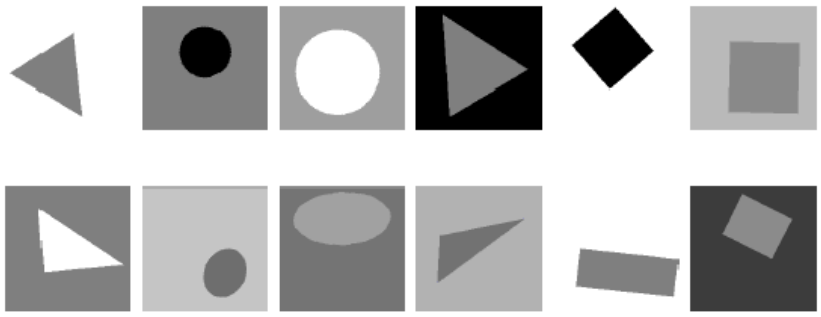
3.1 Planlı Öğrenme Nedir?

Planlı Öğrenme (Curriculum Learning) giriş örneklerinin zorluk seviyesine göre sırayla öğrenicilere verildiği bir optimizasyon yaklaşımıdır. Temeli insanların gerçek hayattaki öğrenme sürecinde bilgilerin karşılarına basitten karmaşığa doğru sıralı şekilde çıkmasına dayanmaktadır. Gerçek hayatta yaşam boyunca lazım olacak temel bilgiler hayatın ilk yıllarında alınır ve bu bilgiler kullanılarak üzerine yeni bilgiler eklenir. Aşamalı olarak gittikçe daha karmaşık hale gelen bilgileri anlayabilmek öğrenme sürecinin başlangıcında alınan temel bilgileri iyi bilmeye bağlıdır. Planlı Öğrenme insanların öğrenme sürecindeki bu doğal akışın makine öğrenmesi metotlarında da yarar sağlayacağı düşünülerek ortaya çıkmış bir fikirdir.

Elman 1993'te yapmış olduğu çalışmasında [12] konunun daha basit taraflarını önceden öğrenip sonradan zorluk derecesini kademeli olarak artırma fikrini önermiş ve dil bilgisi kurallarını özyinelemeli bir ağa öğrettiği denemesinin sonucunda bu kuralları başarılı bir şekilde öğrenmenin başlangıçta karmaşıklığı oldukça kısıtlayan ancak kaynakları yavaş yavaş öğrendikçe genişleyen bir mimari ile gerçekleştiğini göstermiştir. Krueger ve Dayan

da çalışmalarında [13] insanların öğrenme sürecindeki doğal kümülatif akışa benzer şekilde bütün bir karmaşık görevin alt bileşenlere ayrılarak öğrenicilere sunulması ile ciddi ölçüde zamandan kazanç sağlandığını göstermişlerdir. Robotik alanında yapılan bir çalışmada [14] da benzer fikirler ortaya çıkmıştır. Bengio ve ark. bu gelişmeler doğrultusunda Planlı Öğrenme (Curriculum Learning) kavramını makine öğrenmesi literatürüne eklemişlerdir [11].

Bengio ve ark. [11] zorluk derecesini dikkate alarak eğitim örneklerini kolaydan zora doğru sıralamışlar ve bu sıralama ile öğreniciye verildiğinde daha iyi bir yerel minimum noktasının bulunması neticesinde daha iyi sonuçlar elde edilebildiğini göstermişlerdir. Çalışmalarının uygulama kısmının ilk bölümünde önerdikleri yaklaşımla çok katmanlı bir yapay sinir ağını şekil tanıma için yapay bir veri kümesi ile eğitmişlerdir. Kullanılan veri kümesi geometrik şekillerden oluşmaktadır ve basit şekiller/tüm geometrik şekiller olarak ikiye ayrılmıştır. Şekil 3.1'de bu şekillere örnekler verilmektedir.



Şekil 3.1 Basit şekiller(üstte) ve tüm geometrik şekiller(altta) [11]

Eğitimin ilk aşamasında öğreniciye verilen basit şekiller kare, daire, eşkenar üçgen gibi aynı sınıfta bulundukları diğer geometrik şekillere nazaran farklılığı daha az olabilecek örnekler olarak belirlenmiştir. Örneğin basit üçgenlerin hepsi eşkenar üçgendir, tüm eşkenar üçgenlerin açıları aynı olmakla birlikte sadece kenar uzunlukları ve ön-arka plan rengi bakımından farklılık görülebilir. Basit şekilleri öğrenen sisteme daha sonra tüm geometrik şekiller verilerek eğitim tamamlanır. Bu planlama basit şekillerin basit olduğunun daha önceden bilinip ön bilgi olarak öğreniciye verilmesi ile mümkündür. Eğitim örneklerinin bu şekilde kolaydan zora sıralanması ile elde edilen sonuçların basit

ve tüm geometrik şekillerin birlikte tek bir aşamada verildiği klasik yöntemden daha iyi olduğu görülmüştür.

Bengio ve arkadaşlarının çalışmalarının devamında bir cümleyi takip eden en iyi kelimeyi bulmak için bir Özyinelemeli Sinir Ağı (Recurrent Neural Network, RNN) Planlı Öğrenme yöntemi ile eğitilmiştir. Bu uygulamada 20.000 eğitim örneği 5000'er örnekli gruplara ayrılarak 4 aşamada eğitim gerçekleştirilmiştir. Veri kümesi olarak İngilizce Wikipedia sözlüğü kullanılmış, en sık tekrarlanan 5000 kelime kolay olarak belirlenmiş ve ilk aşamada verilecek şekilde ayarlanmıştır. Sonraki aşamalarda da kullanım sıklığına göre yeni 5000'er kelime eklenmiştir. Eğitimin son aşamasında tüm örnekler verildiğinde Planlı Öğrenmenin klasik yöntemi geçtiği gözlemlenmiştir.

Planlı Öğrenme stratejisinin tanıtılmasından sonra birçok araştırmacı bu yöntemi teorik ve deneysel açıdan ele almıştır. 2009'dan sonra konu ile ilgili yayınlanmış çalışmalardan bazıları bir sonraki bölümde incelenecektir.

3.2 Önceki Çalışmalar

Örneklerin anlamlı bir sıralama ile öğreniciye verilmesinin bir optimizasyon sağladığının Bengio ve ark. [11] tarafından gösterilmesinden sonra bu stratejinin uygulanabilirliğini artırmak için çeşitli yöntemler önerilmiştir. Bunun yanı sıra literatürde Planlı Öğrenmenin nasıl bir optimizasyon sağladığı ile ilgili teorik açıklamalar arayan çalışmalar da mevcuttur.

Kumar ve ark. [15] 2010 yılında örneklerin kolaylık seviyesinin her aşamada öğrenici tarafından belirlendiği bir yöntem önermişlerdir. Kendini Planlayan Öğrenme (Self-paced Learning, SPL) adını verdikleri yöntemde eğitim kümesinden belirlenen sayıda rastgele örnek alınır ve eğitime başlanır. Bu örneklerle eğitilmiş sisteme tüm eğitim örnekleri verilerek öğrenicinin tahmin ettiği değerin örneğin gerçek sınıfına yakınlığı hesap edilerek tüm örneklerin zorluk derecesinin etiketlenmesi sağlanır. Bir sonraki aşamada belirlenen zorluk seviyesinin altında kalan örneklerle sistem tekrar eğitilir.

Kumar ve ark. çalışmalarının uygulama bölümünde öğrenici olarak gizli yapılı SVM (latent structural SVM, SSVM) kullanmışlar ve 4 farklı makine öğrenmesi uygulamasında bu öğrenicinin klasik eğitim metodu olan içbükey-dışbükey prosedür (Concave-convex procedure, CCCP) yöntemine karşı başarılı sonuçlar elde etmişlerdir. Uygulama yapılan alanlardan birisi verilen bir dokümanda geçen isimlerden bahsedilen konunun çıkarılmasıdır. Bu denemede önerdikleri yöntemin 40 denemeden 11'inde sonuçları iyileştirdiğini görmüşlerdir. Bir diğer uygulama alanı bir proteini ayırt edebilmek için DNA dizilimindeki motifin bulunması işlemidir. Her bir protein türüne ait 40.000'er tane pozitif ve negatif DNA örneği içeren veri kümesi kullanılmış ve test edilen 5 protein türünden 4 tanesinde CCCP yönteminden daha iyi sonuçlar elde edilmiştir. Uygulamalardan bir diğeri el yazısı rakam tanımadır. 0-9 arasındaki rakamlardan birbirine en çok benzeyen 1-7, 2-7, 3-8, 8-9 rakamlarının birbirinden ayırt edilmesi için yine bir SSVM eğitilmiş ve 1 ile 7'yi ayırt etmede Kendini Planlayan Öğrenme yönteminin daha iyi sonuçlar verdiği görülmüştür. Son uygulamalarında ise verilen bir görüntüdeki ana nesnenin yerini bulma işlemini gerçekleştirmişlerdir. Yerinin tespit edilmesi zor olan örneklerde kolay ve zor tüm örneklerin birlikte verildiği CCCP yönteminin başarılı olamadığı görülürken, önerdikleri yöntemin zor örneklerin verildiği iterasyondan sonra zor örneklerin de yerini tespit edebildiğini göstermişlerdir. Kumar ve ark. yaptıkları 4 uygulamanın sonucunda SPL yönteminin eğitim süresini artırmasına karşın daha iyi bir yerel minimuma eriştiğini ortaya koymuşlardır.

Jiang ve ark. eğitim sırasında örneklerin sıralamasını kolaylık seviyesinin yanı sıra çeşitliliğini (diversity) de hesaba katarak bulmayı öneren bir yöntem (Self-paced Learning with Diversity, SPLD) [16] sunmuşlardır. Bu yöntemde bir sonraki aşamada alınacak örnekler sadece tahmin edildiği doğruluk derecesine göre değil önceden verilmiş diğer örneklerden daha farklı bir örnek olmasına da bakılarak seçilir. Bu yöntemde çeşitlilik ve doğruluk oranını belirten iki parametre kullanılmaktadır. Çalışmada video tanıma üzerine 2 farklı uygulama yapılmıştır. İlk uygulama verilen bir videodaki olayın belirlenmesidir. Özellik çıkarımı işlemini Derin Evrişimsel Sinir Ağı (Deep Convolutional Neural Network, CNN) ile gerçekleştirdikten sonra elde edilen veri kümesi üzerinde SVM eğitilmiştir. İkinci uygulamada ise biri filmlerdeki sahnelerde geçen eylemlerin

tanınmasını diğeri ise videolarda hangi sporun yapıldığının belirlenmesi ile ilgili 2 veri kümesi üzerinde denemeler yapılmıştır. Yapılan karşılaştırmalar sonucunda SPLD yönteminin SPL'den daha başarılı olduğu gösterilmiştir. Bu sonuçlar geleneksel Planlı Öğrenmenin önerdiği kolaydan zora eğitimde birtakım modifikasyonlar yapılarak daha iyi bir başarı elde edilebildiğini ve örnek sıralamasında iyi bir planlamanın sadece kolaylık seviyesi ile ilgili olmadığını ortaya koymaktadır.

Bu gelişmelerden sonra birçok araştırmacı Planlı Öğrenme yaklaşımıyla en iyi verimi elde etmek için en etkili sıralamayı bulmaya çalışmıştır. Zaremba ve Sutskever'in çalışmasında [17] geleneksel Planlı Öğrenme iyi sonuç vermemiş ve yeni bir versiyon geliştirilmiştir. Bu çalışmada RNN'lerin bir türü olan Uzun Kısa-Süreli Bellek (Long Short-Term Memory, LSTM) ağlarına toplama ve ezberleme işlemi yaptırılmıştır. Toplama işleminde birincisi toplanacak sayıların kaç haneli olduğunu ifade eden uzunluk parametresi, ikincisi ise kaç tane işlemi birleştirmeye izin verileceğini belirleyen bir parametredir. Araştırmacılar 3 farklı Planlı Öğrenme stratejisini klasik plansız yöntemle karşılaştırmışlardır. Karşılaştırılan stratejilerden birincisi her aşamada uzunluk parametresinin artırıldığı sınırına ulaştıktan sonra işlem sayısı parametresinin artırıldığı geleneksel Planlı Öğrenmedir. İkinci strateji uzunluk ve işlem parametresinin herhangi bir değerini rastgele seçip bir aşamada sadece bu parametrelerdeki örneklerle eğitildiği karışık Planlı Öğrenmedir. Üçüncüsü ise geleneksel ve karışık stratejilerin birinden birinin her aşamada uygulandığı birleşik Planlı Öğrenme stratejisidir. Yapılan denemeler sonucunda birleşik Planlı Öğrenme stratejisinin geleneksel Planlı Öğrenmeyi geçtiği ve tüm toplama işlemlerinde %99 oranında başarılı olduğu gözlemlenmiştir.

Shi ve ark. [18] da Planlı Öğrenme yaklaşımının dil modelleri için RNN'lere uygulanmasında üç farklı planlama stratejisi önermişlerdir. Birincisi sözlükten başlayan planlama olup bir alt görev için eğitilecek ağın sözlüğünün tüm eğitim kümesi ile oluşturulduğu ancak eğitimin sadece o görevin örnekleri ile gerçekleştirildiği durumdur. İkincisi veri sıralama olup her alt görev için eğitilen ağın eğitimi için tüm eğitim kümesinin o alt görevin örnekleri ile bitecek şekilde sıralanıp kullanıldığı ve validasyonunun da bu örneklerle gerçekleştirildiği durumdur. Üçüncüsü ise tümden özele sıralama yöntemidir,

bu yöntemde eğitim periyotları belli bir devir ile sınırlanır ve her eğitim periyodunda ağlar öncelikle tüm örneklerle eğitim sonra ilgili alt görevin örnekleri ile eğitim gerçekleştirilir. Tüm örneklerin periyodunda validasyon için tüm örnekler kullanılırken, alt görevlerin periyodunda ilgili alt görevin örnekleri kullanılır. Araştırmacılar önerdikleri stratejileri alt görevin eğitim ve test sırasında bilindiği, eğitim sırasında bilinip test sırasında bilinmediği ve ne eğitim ne de test sırasında bilinmediği 3 durumda denemişlerdir. Denemeler sonucunda veri sıralama ve tümden özele sıralama yöntemlerinin klasik temel eğitim yönteminde daha başarılı sonuçlar elde ettiğini göstermişlerdir.

Bilgisayarla görme alanında Pentina ve ark. [19] çok görevli öğrenmede (multitask learning) öğrenilecek görevlerin en iyi sıralamasını aramışlardır. Görevlerin planlı öğrenilmesi için geliştirdikleri yaklaşımda tüm görevler için ortalama sınıflandırma performansını optimize edecek görev sıralaması seçilir. Her aşamada mevcut hedef fonksiyonu üzerinde hatası düşük olan ve bir önceki göreve benzeyen görev öğrenilmek üzere seçilir. Adaptif SVM'in öğrenici olarak kullanıldığı denemelerde birbiriyle ilişkili çok görevin sıralı şekilde öğretilmesinin birleşik halde öğretmekten daha verimli olduğu görülmüştür.

Literatürde denenmiş yöntemler genel olarak tüm örneklerin tek bir aşamada verildiği klasik yöntemden daha iyi performans göstermesine rağmen önerilen planlama yöntemlerinin en iyi planlama olup olmadığı bilinmemektedir. Planlama stratejisinin otomatik şekilde belirlenmesi ile uygulama kapsamı genişlemekte ve konuya eğilim daha da artmaktadır. Planlı Öğrenme son yıllarda oldukça popüler olmuş bir yaklaşımdır ve sadece 2017 yılında konu ile ilgili yapılan araştırma sayısı ile kendini ikiye katlamıştır.

3.3 YSA için Ölçeklenebilir Bir Planlı Öğrenme Yöntemi ve Uygulaması

Bu bölümde Planlı Öğrenme ve Ters Planlı Öğrenme yaklaşımlarının birçok alanda uygulanabilmesi için tez kapsamında geliştirilen ölçeklenebilir bir sıralama belirleme yaklaşımı ve bu yaklaşımla gerçekleştirilen uygulama açıklanacaktır.

Planlı Öğrenme yaklaşımı örneklerin zorluk seviyesine göre sıralanması temeline dayanır. Bu çalışmada da Bengio ve arkadaşlarının çalışmasında [11] açıklanan yaklaşımla eğitim örnekleri zorluk derecesi dikkate alınarak kolaydan zora (Planlı) ve zordan kolaya (Ters Planlı) sıralı şekilde modele verildiğinde elde edilen sonuçlar ile klasik yöntemde karışık (plansız) şekilde verildiğinde elde edilen sonuçlar farklı veri kümelerinde T-test yapılarak karşılaştırılmıştır. Burada ilk aşama örneklerin zorluk seviyesinin belirlenmesidir. Bu çalışmada birçok farklı veri kümesinde çalışılacağından örneklerin zorluk derecesinin önceden bilinmesi mümkün değildir. Bu nedenle zorluk seviyesine karar verebilmek için bir makine öğrenmesi metodu kullanılmıştır. Örnekler zorluk seviyelerine ayrıldıktan sonra eğitilen modele kolaydan zora verilerek Planlı Öğrenme, zordan kolaya verilerek Ters Planlı Öğrenme gerçekleştirilmiştir.

3.3.1 Zorluk Seviyesinin Belirlenmesi

Bir örneğin kolay veya zor olup olmadığına karar vermek için birçok yöntem kullanılabilir. Bu çalışmada iki farklı makine öğrenmesi yöntemi denenmiş ve karşılaştırılmıştır. Birincisi K-en yakın komşu (K-nearest neighbour, K-NN) ikincisi ise bir topluluk (ensemble) algoritmasıdır. Zorluk düzeylerini belirlemek için her iki yöntemin çıkışında da entropi kullanılmıştır.

K-En Yakın Komşu: Bu yöntemde bir örneğin en yakın k tane komşusunun sınıflarına bakılır ve örnek komşuları arasında en çok yaygın olan sınıfın etiketi ile etiketlenir [20]. Bu çalışmada ise zorluk derecesinin belirlenmesi için en yakın 7 komşunun sınıf dağılımları incelenmiştir. Seçilen komşu sayısı küçük bir değer olursa örneklerin zorluk derecesi açısından farklılığı düşük olacaktır. Yüksek bir değer olursa da her örnek için hesaplama maliyeti oldukça yüksek olacaktır. Bu nedenle birkaç deneme neticesinde en yakın 7 komşuya bakılması uygun olarak görülmüştür. Bu yaklaşıma göre, eğer örneğin sınıfı çevresindeki örneklerle aynı ise, bu kolay bir örnektir ancak çevresindeki örneklerden farklı bir sınıfa aitse zor bir örnek olarak kabul edilir. Her bir x_p örneği için ϑ_i etiketine sahip bir sınıfın olasılığı P_i, Denklem 3.1 kullanılarak hesaplanmaktadır.

$$P_i \leftarrow \frac{\sum_{j=1}^{k} \delta(\vartheta_i, f(x_j))}{k} \quad (3.1)$$

Burada x_j, x_p örneğinin j. en yakın komşusunu temsil etmekteyken $f(x_j)$, x_j örneğinin sınıfı ve $\delta(.)$, $\mathrm{f}(\mathrm{x_j}) = \vartheta_\mathrm{i}$ olduğunda 1, diğer durumlarda ise 0 çıktısını üreten eşitlik fonksiyonudur.

Kolektif Öğrenme: Makine öğrenmesi yöntemlerinde tek bir sınıflandırıcı yerine çoklu sınıflandırıcıların kararlarını birleştirilmesi daha yüksek doğruluk oranlarının elde edilmesini sağlar [21]. Bu çalışmada Breiman'ın önerdiği yöntemle [22] oluşturulan 10 ağaçlı bir sınıflandırıcı topluluğunun tahminleri örneklerin zorluk seviyesini belirlemek için kullanılmaktadır. Torbalama (Bagging) adı verilen bu yöntemde tekil sınıflandırıcıların her biri tüm özellikleri içeren farklı örnek alt kümeleri ile eğitilir. Her tekil sınıflandırıcı için veri kümesinden rastgele seçilen örneklerin sayısı veri kümesinin yaklaşık %63'ü kadardır. Sonuçta farklı örnek alt kümeleri ile eğitilen sınıflandırıcıların tekil başarıları yüksek olurken tüm sınıflandırıcılar özelliklerin tamamı ile eğitildiği için elde edilen tahminlerin farklılığı düşük olmaktadır. Tekil sınıflandırıcıların tahminleri demokrasi usulü (majority voting) ile birleştirilir. Bir örnek hakkında topluluğun ortak kararı tekil sınıflandırıcılar tarafından en çok tahmin edilen sınıf olarak belirlenir. Bu yaklaşımla bütün sınıflandırıcıların aynı sonucu ürettiği bir örnek kolay olarak belirlenirken sınıflandırıcıların farklı tahminlerde bulunduğu bir örnek zor olarak belirlenmektedir. Her bir ϑ_i etiketi için x_p örneğinin o sınıftan olma olasılığı P_i, Denklem 3.2 ile hesaplanır.

$$P_i \leftarrow \frac{\sum_{j=1}^{m} \delta(\vartheta_i, \varphi_j(x_p, L_j))}{m} \quad (3.2)$$

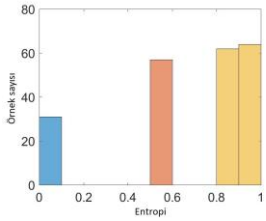
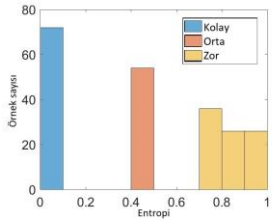
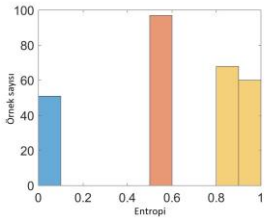
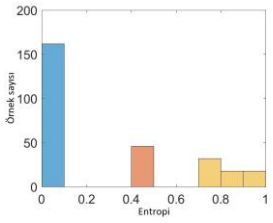
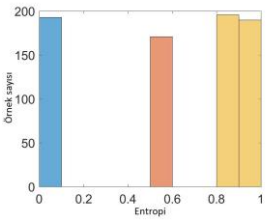
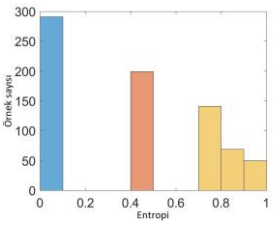
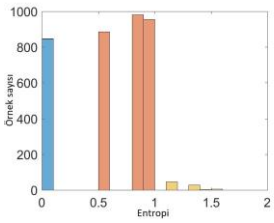
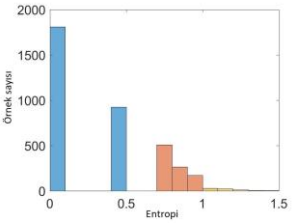
Burada m topluluktaki sınıflandırıcı sayısı, $\varphi_j(x_p, L_j)$, L_j eğitim kümesi ile eğitilen j. sınıflandırıcının tahmini ve $\delta(.)$, $\varphi_j(x_p, L_j) = \vartheta_i$ olduğunda 1, diğer durumlarda ise 0 çıktısını üreten eşitlik fonksiyonudur.

Entropi: Bir çeşit homojenlik ölçümüdür [20]. Bu çalışmada, her iki yöntemin çıktısında entropi kullanılmıştır. K-en yakın komşu metodunda, komşuların sınıflarının homojenliğini ölçmek için entropi kullanılmıştır. Burada düşük entropi, örneğin komşuları ile aynı sınıfta olduğu anlamına gelir. Topluluk yönteminde ise sınıflandırıcı tahminlerinin homojenliğini ölçmek için entropi kullanılmıştır. Burada ise düşük entropi, sınıflandırıcıların örneğin sınıfı konusunda hemfikir olduğu anlamına gelir. Bu nedenle her iki yöntem için bir örneğin düşük entropi değeri (yüksek homojenlik) varsa bu kolay bir örnektir veya yüksek entropi değeri (düşük homojenlik) varsa zor bir örnektir. x_p örneğinin entropisi Denklem 3.3 ile hesaplanır. Burada c veri kümesinin sınıf sayısını göstermektedir.

$$H(x_p) = -\sum_{i=1}^{c} P_i log_2 P_i \tag{3.3}$$

Eğitim kümesindeki her entropi değeri için örnek sayıları histogramla ifade edilmiştir. Histogram 3 eşit aralığa bölünerek düşük entropiye sahip olan örnekler kolay, yüksek entropisi olan örnekler zor, arada kalan örnekler ise orta seviye olarak belirlenmektedir. Şekil 3.2'de, her iki yöntem için bazı veri kümelerinin histogramları verilmiştir ve aralıklar farklı renklerle gösterilmiştir.

Zorluk belirleme yöntemlerinin çıktısını direkt örneklerin gerçek sınıfıyla karşılaştırmak yerine entropi kullanılarak çok sınıflı veri kümelerinde daha iyi bir seviyelendirme elde edilmiştir. Veri kümeleri kendilerine özel eşik değerlerine göre zorluk seviyelerine ayrıldığından örneklerin zorluk seviyesi birbirlerine göre belirlenmiştir.

	K-en yakın komşu tabanlı zorluk seviyesi belirleme	Kolektif öğrenme tabanlı zorluk seviyesi belirleme
(a)		
(b)		
(c)		
(d)		

Şekil 3.2 (a) Göğüs kanseri (Breast-cancer), (b) Kolik (Colic), (c) Kredi-g notu (Credit-g), (d) Dalga formu (Waveform) veri kümelerinin histogramları

3.3.2 Eğitim Kümelerinin Oluşturulması ve Modelin Eğitilmesi

Planlı ve Ters Planlı Öğrenme yaklaşımları Yapay Sinir Ağı üzerinde denenmiştir. Kullanılan Yapay Sinir Ağı modeli ileri beslemeli çok katmanlı perceptrondur. 3 gizli katmanı bulunan ağın gizli katmanındaki nöronların sayıları sırasıyla 10, 5 ve veri kümesinin sınıf sayısı olarak belirlenmiştir. Ağ her örnekte ağırlıkların güncellendiği artımsal (incremental) metot kullanılarak momentumlu gradyan düşümü kuralıyla

eğitilmiştir. Tüm örneklerin üzerinden 1 defa geçilmesi ile 1 devir (epoch) tamamlanmaktadır.

Planlı Öğrenme için eğitim süreci ardışık üç aşama olarak tasarlanmıştır. Bölüm 3.1.1'de açıklanan şekilde her örnek için zorluk seviyelerinin kolay, orta ve zor olarak belirlenmesinden sonra, örnekler sırayla ağa sunulmaktadır. Çizelge 3.1'de, her bir aşamada gösterilen örnek sayıları verilmektedir.

Çizelge 3.1 Her aşamada gösterilen örnek sayıları

Aşama no.	**Kullanılan zorluk seviyesi ve devir sayısı**
1	kolay (n/3)
2	orta (2*n/3), kolay (n/3)
3	zor (n), orta (n/3), kolay (n/3)

Çizelge 3.1'de, ‘n’ klasik (rastgele sıralı artımsal) yöntemin ortalama devir sayısıdır. Tüm aşamalar tamamlandığında, tüm örnekler klasik yöntemde olduğu gibi ağa n kez sunulmuş olacaktır. Yeni örnekler eklenirken eski örneklerin ihmal edilmesinden dolayı ağın başarısının olumsuz bir şekilde etkilenmemesi için önceki aşamadaki örnekler her aşamada tekrar sunulmaktadır. Ters Planlı Öğrenme yönteminde ise Planlı Öğrenme yaklaşımı ters sırayla gerçekleştirilir. Bu yöntemde ağ en zor örneklerden başlayarak her aşamada örneklerin zorluk seviyeleri azalan örneklerle eğitilir.

3.3.3 Deney Tasarımı ve Test Sonuçları

Planlı ve Ters Planlı yaklaşımların etkisini farklı uygulama alanlarında görebilmek için çeşitli veri kümeleri üzerinde denemeler yapılarak klasik yöntemle karşılaştırılmıştır.

Veri kümeleri: Deneylerde kullanılan veri kümeleri [23] yakınsadıkları ortalama devir sayıları birlikte Çizelge 3.2'de verilmiştir. Yakınsama koşulu, validasyon kümesinin hata oranının üst üste 6 devir boyunca yükselmesidir. Denemeler her veri kümesi için 20 farklı başlangıç değeri ile tekrarlanmış ve elde edilen sonuçların ortalaması hesaplanmıştır. Her yöntem aynı 20 değerden başlatılarak adil bir karşılaştırma olması sağlanmıştır.

Çizelge 3.2 Veri kümeleri

No.	Veri kümesi	Örnek sayısı	Özellik sayısı	Sınıf sayısı	Ortalama yakınsama devri
1	labor	57	26	2	21
2	zoo	84	16	4	33
3	lymph	142	37	2	21
4	iris	150	4	3	30
5	hepatitis	155	19	2	21
6	audiology	169	69	5	23
7	autos	202	71	5	17
8	glass	205	9	5	15
9	sonar	208	61	2	21
10	heart-statlog	270	13	2	20
11	breast-cancer	286	38	2	12
12	primary-tumor	302	23	11	12
13	ionosphere	351	33	2	21
14	colic	368	60	2	15
15	vote	435	16	2	24
16	balance-scale	625	4	3	24
17	soybean	675	83	18	14
18	credit-a	690	42	2	12
19	breast-w	699	9	2	15
20	diabetes	768	8	2	21
21	vehicle	846	18	4	29
22	anneal	890	62	4	32
23	vowel	990	11	11	24
24	credit-g	1000	59	2	13
25	col10	2019	7	10	20
26	segment	2310	18	7	24
27	splice	3190	287	3	15
28	kr-vs-kp	3196	39	2	27
29	hypothyroid	3770	31	3	26
30	sick	3772	31	2	33
31	abalone	4153	10	19	14
32	waveform	5000	40	3	13
33	d159	7182	32	2	15
34	ringnorm	7400	20	2	23
35	mushroom	8124	112	2	15
36	letter	20000	16	26	12

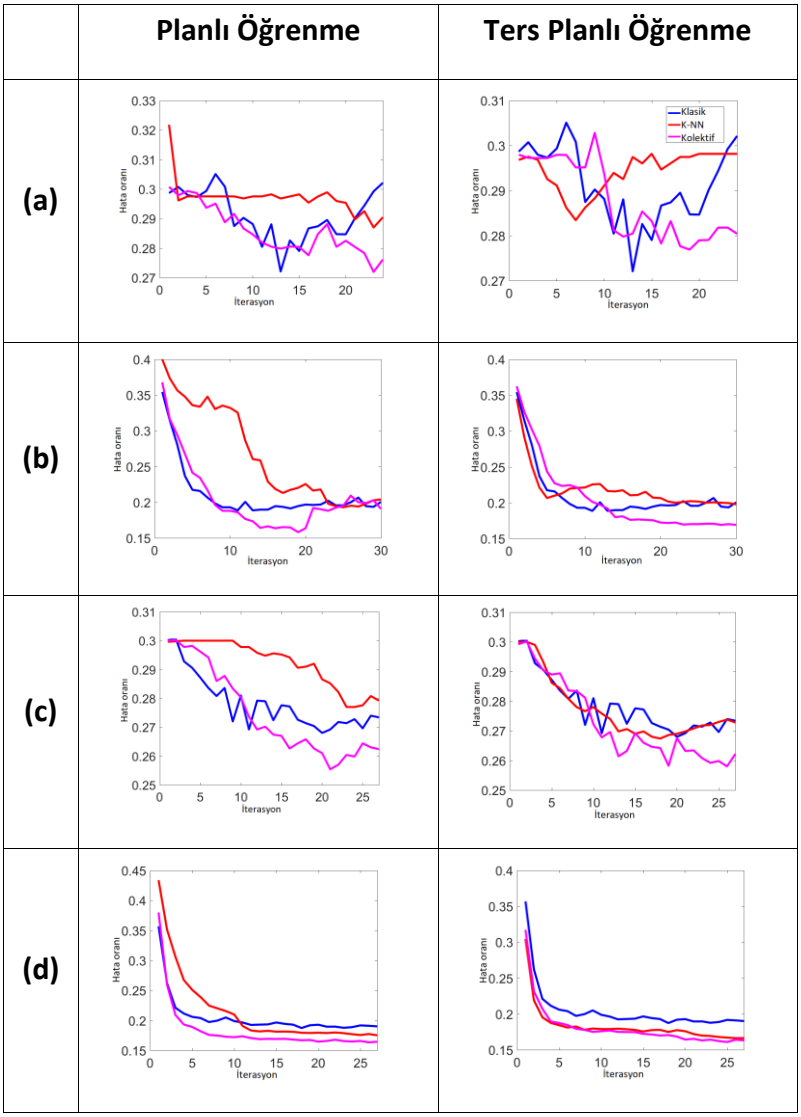
a. Zorluk Belirleme Yöntemlerine Genel Bir Bakış:

Planlı ve Ters Planlı Öğrenme için hazırlanan eğitim kümeleri kullanılarak bu yöntemler klasik yöntemle karşılaştırılmıştır. Yöntemlerde eğitim örneklerinin toplam gösterilme sayısının eşit olmasını sağlamak için eğitim belirli bir iterasyon sayısı ile sınırlandırılmıştır. Bu sayı validasyon kümesindeki hatanın 6 devir üst üste yükselmesi koşuluyla eğitimin durdurulmasıyla elde edilir. Her veri kümesi için 20 deneme alınarak ortalama yakınsama devri bulunmuştur.

Şekil 3.3'de bazı veri kümeleri için ortalama devir sayısının iki katıyla sınırlı olan yöntemlerin hata oranları gösterilmektedir. Burada hem K-en yakın komşu hem de Kolektif öğrenme tabanlı Planlı ve Ters Planlı Öğrenme yöntemleri için elde edilen hata oranları verilmiştir. Grafiklerdeki değerler elde edilen 20 hatanın ortalamasıdır. Planlı ve Ters Planlı Öğrenme yöntemlerinin ortalama hata oranlarının genellikle klasik yöntemden daha düşük olduğu görülmektedir. Ayrıca, K-en yakın komşu tabanlı sıralama ile hata oranlarının eğitim sırasında yavaş yavaş azaldığı söylenebilir. K-en yakın komşu tabanlı Planlı ve Ters Planlı Öğrenme yaklaşımlarının hata oranlarının (c) gibi bazı durumlarda klasik yöntemden daha yüksek olduğu görülmüştür.

b. Eğitim ve Test Kümesi Hatalarının İncelenmesi:

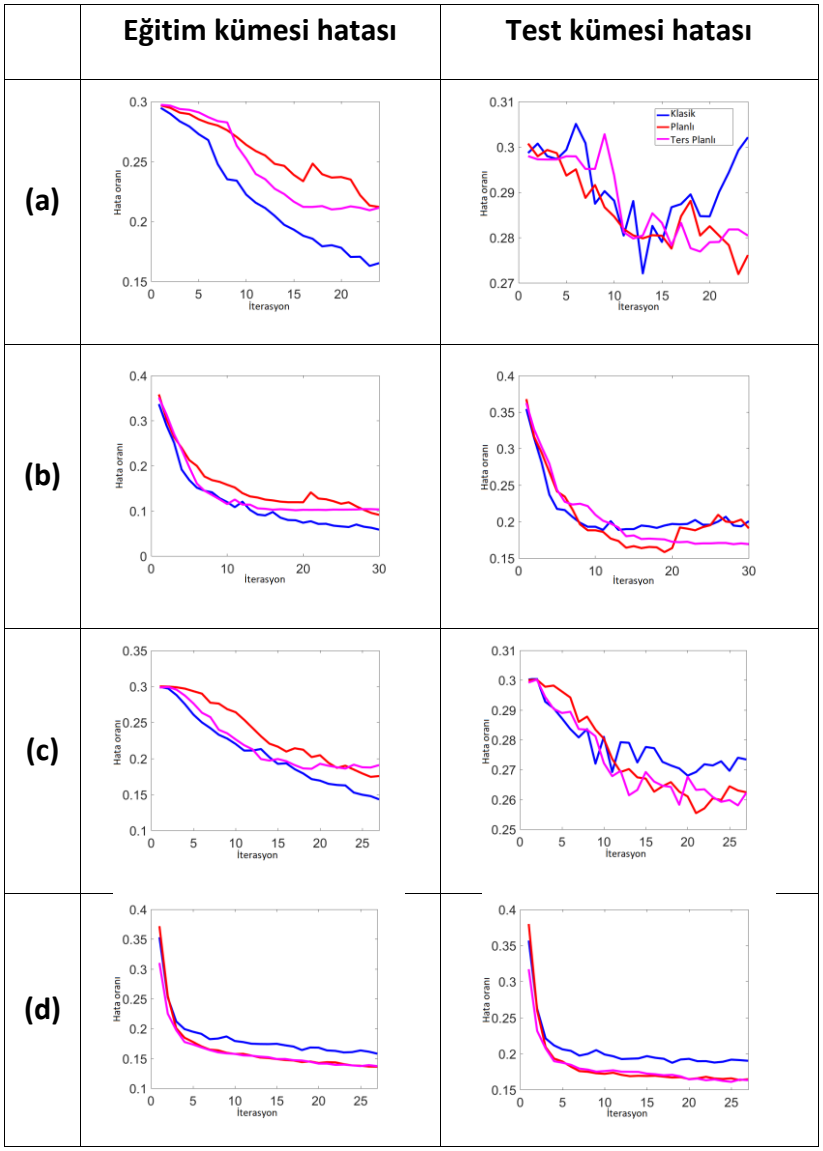
Eğitim kümesi ve test kümesindeki hata oranları arasındaki ilişkiyi açıklamak üzere Şekil 3.4'te bazı veri kümelerinin grafikleri verilmiştir. Bu grafikler Kolektif öğrenme tabanlı zorluk derecesi belirleme yöntemi kullanılarak eğitim süreci boyunca hata oranlarının değişimini göstermektedir. Klasik yöntemde eğitim boyunca eğitim kümesi hatası azalmaya devam ederken, (a)'daki gibi bazı durumlarda test kümesi hatası artmaktadır. Bu, ağın devir sayısı sınırına ulaşana kadar aşırı eğitilmiş olduğu anlamına gelmektedir. Planlı ve Ters Planlı Öğrenme yaklaşımlarının eğitim kümesi hatalarının klasik yöntemin elde ettiği eğitim kümesi hatalarından daha yüksek olduğu görülmektedir. Bu yaklaşımların eğitim kümesi hataları bir değerin yakınlarında sabitlenmekte ve belirlenen devir sayısı sınırında bu değerden daha fazla düşüş göstermemektedir. Bu yaklaşımlarda



Şekil 3.3 (a) Göğüs kanseri (Breast-cancer), (b) Kolik (Colic), (c) Kredi-g notu (Credit-g), (d) Dalga formu (Waveform) veri kümeleri üzerinde önerilen yöntemlerin her iterasyondaki ortalama test hataları

her aşamada farklı örnekler sunmak ağın aynı örnekleri ezberlememesini sağlar. Dolayısıyla Planlı ve Ters Planlı Öğrenme yöntemlerinin test kümesi hataları klasik yöntemden daha düşük çıkmaktadır.

3 yöntemin karşılaştırılmasında Planlı ve Ters Planlı Öğrenme yaklaşımlarının birçok veri kümesi üzerinde eğitimin sonunda daha iyi hata oranları elde edebildiği görülmektedir. Ters Planlı Öğrenme, daha fazla veri kümesinde iyi sonuçlara sahiptir ve bu yaklaşımdaki hatalar hem eğitim hem de test kümelerinde daha pürüzsüz şekilde düşmektedir.



Şekil 3.4 (a) Göğüs kanseri (Breast-cancer), (b) Kolik (Colic), (c) Kredi-g notu (Credit-g), (d) Dalga formu (Waveform) veri kümeleri üzerinde kolektif öğrenme tabanlı zorluk seviyesi belirleme ile eğitim boyunca ortalama eğitim ve test kümesi hataları

c. Karşılaştırma Testleri ve Sonuçları:

Her deneme için validasyon kümesi hatası 6 kez üst üste arttığında durdurulan 20 denemenin ortalama devir sayısı ‘n’ olmak üzere, 36 UCI veri kümesinde [23] T-test ile 6 yöntem karşılaştırılmıştır. Klasik yöntemin yanı sıra, hesaplanan devir sınırı (n) ile sınırlandırılan Planlı ve Ters Planlı Öğrenme yöntemleri ve bu yöntemlerin 2n devir sayısı ile sınırlı versiyonları karşılaştırılmıştır.

Aşağıda listelenen 6 algoritma karşılaştırılmıştır:

- Klasik yöntem, 6 kez üst üste validasyon hatası yükseldiğinde durduruldu.
- Klasik yöntem, ortalama devir sayısının iki katıyla sınırlı (Klasik (2n)).
- Planlı Öğrenme, ortalama devir sayısı ile sınırlı (Planlı (n))
- Planlı Öğrenme, ortalama devir sayısının iki katı ile sınırlı (Planlı (2n))
- Ters Planlı Öğrenme, ortalama devir sayısı ile sınırlı (Ters Planlı (n))
- Ters Planlı Öğrenme, ortalama devir sayısının iki katı ile sınırlı (Ters Planlı (2n))

Tüm yöntemler aynı başlangıç koşullarıyla başlatılmıştır.

Denemeler sırasında öncelikle, 5x4 katlı çapraz doğrulama (Cross Validation, CV) ile elde edilen 20 hata oranı kullanılarak iki zorluk belirleme yöntemi karşılaştırılmıştır. Çizelge 3.3'te, K-yakın komşu ve Kolektif öğrenme tabanlı yöntemler için klasik yöntem ile 2n sınırlı olan Planlı ve Ters Planlı Öğrenme yaklaşımlarının karşılaştırma sonuçları sunulmuştur. Çizelgedeki kazanma ve kayıp sayıları, T-testi ile istatistiksel olarak anlamlı sonuç elde edilen veri kümesi sayısına karşılık gelir. Burada Kolektif öğrenme tabanlı tekniğin daha başarılı olduğu görülmektedir.

Çizelge 3.3 Zorluk belirleme yöntemlerinin karşılaştırması

Zorluk belirleme yöntemleri		**Kazanma**	**Beraberlik**	**Kaybetme**
K-en yakın komşu	**Planlı**	7	26	3
	Ters planlı	11	21	4
Kolektif öğrenme	**Planlı**	11	25	0
	Ters planlı	**14**	22	0

Kolektif öğrenme tabanlı Ters Planlı Öğrenme, veri kümelerinin önemli bir bölümünde daha iyi sonuçlar elde etmiştir. K-en yakın komşu tabanlı zorluk tespiti yönteminin hem Planlı hem de Ters Planlı yaklaşımlarda bazı veri kümelerinde kaybederken, Kolektif öğrenme tabanlı yöntem klasik yönteme karşı kazanmakta veya berabere kalmaktadır.

K-en yakın komşu yönteminde, örneklerin yerleri dikkate alınarak zorluğu belirlenir. Bir örneğin diğer sınıflardan komşusu varsa, komşuları kendi sınıfından olan bir örneğe göre otomatik olarak daha zor olmaktadır. Kolektif öğrenme yöntemi ise sınıflandırıcıların tahminleri ile zorluğu belirler. Bu yöntemde, en az bir sınıflandırıcı bir örneği yanlış sınıflandırırsa, bu tüm sınıflandırıcıların doğru şekilde sınıflandırdığı bir örneğe göre zor bir örnek olduğu anlaşılır. Bu nedenle Kolektif öğrenme yönteminin daha güvenilir zorluk seviyeleri ürettiği söylenebilir.

Denemelerin ikinci aşamasında elde edilen, önceki denemelerde daha başarılı olan Kolektif öğrenme tabanlı zorluk belirleme yöntemi için bahsedilen 6 algoritmanın 36 veri kümesindeki sonuçları Çizelge 3.4'te verilmektedir. Hücrelerdeki kazanma ve kaybetme sayıları, T-testi ile istatistiksel olarak önemli hata oranlarını elde edilen veri kümesi sayılarını ifade etmektedir. T-testine 5x4 katlı çapraz doğrulama ile elde edilen toplam 20 denemenin hata oranları verilmiştir. Ortalama devir sayısının iki katı ile sınırlandırılan yöntemlerde, her aşamadaki örnekler yeterince öğrenilmiş olduktan sonra bir sonraki aşamaya geçilmesi sağlanır. Bu sayede Planlı ve Ters Planlı Öğrenme yöntemleri daha fazla veri kümesi üzerinde klasik yönteme karşı istatistiksel olarak düşük hata oranları elde etmiştir. Planlı Öğrenme yöntemleri arasındaki karşılaştırmayı incelediğimizde, 2n sınırlı olan yöntem, daha fazla veri kümesinde başarılı olmuştur. Klasik yöntem için, örnekleri daha fazla sunmanın genel bir yararı olmadığı da görülmektedir.

Birkaç veri kümesinde, Klasik (2n) tarafından elde edilen hatalar, validasyon kümesi hatasının artmasıyla durdurulan yöntemden daha yüksektir. Klasik yöntemde, 2n devir sınırı, bu veri kümelerinde aşırı eğitime neden olmuştur. Aynı devir sayısı ile sınırlı Planlı ve Ters Planlı Öğrenme yöntemleri için ise aşırı eğitim meselesi ortaya çıkmamaktadır.

Burada Planlı ve Ters Planlı Öğrenmenin başarısının nedeni, klasik yöntemin aşırı eğitilmesi olarak düşünülmemelidir. Çünkü bu yaklaşımlar normal klasik yönteme karşı da önemli bir başarı sağlamaktadır. Çizelge 3.4'te görülebileceği gibi, Kolektif öğrenme

Çizelge 3.4 Algoritmaların T-test sonuçları

T-test Sonuçları[a]		(X)					
		Klasik	**Klasik (2n)**	**Planlı (n)**	**Planlı (2n)**	**Ters Planlı (n)**	**Ters Planlı (2n)**
(Y)	**Klasik**	-	1/33/2	5/30/1	**10/26/0**	4/32/0	**14/22/0**
	Klasik (2n)	2/33/1	-	6/29/1	**11/25/0**	6/30/0	**14/22/0**
	Planlı (n)	1/30/5	1/29/6	-	8/28/0	4/31/1	11/25/0
	Planlı (2n)	0/26/10	0/25/11	0/28/8	-	0/32/4	2/33/1
	Ters Planlı (n)	0/32/4	0/30/6	1/31/4	4/32/0	-	4/32/0
	Ters Planlı (2n)	0/22/14	0/22/14	0/25/11	1/33/2	0/32/4	-

[a.] X(kazanır)/berabere/Y(kazanır)

tabanlı ters planlı öğrenme (2n), hem Klasik hem de Klasik (2n) yöntemine karşı 14 veri kümesinde istatistiksel olarak daha başarılı sonuçlar elde etmiştir. Kolektif öğrenme tabanlı planlı öğrenme (2n) de birçok veri kümesinde istatistiksel olarak anlamlı performansa sahiptir.

3.4 Planlı Öğrenmeden Çıkarılan Sonuçların Değerlendirilmesi

Bu bölümde, Planlı Öğrenmenin uygulama alanını sınırlandıran zorluk derecesi belirleme probleminin üstesinden gelmek için veri kümelerinden örneklerin zorluk seviyelerini otomatik olarak üreten iki farklı yöntem önerilmiştir. Zorluk derecesi belirlemek için Kolektif öğrenme kullanılarak bir sınıflandırıcı topluluğunun tahminlerini kullanmanın, en yakın komşuların sınıflarını kullanmaktan daha iyi olduğunu görülmüştür.

Zorluk seviyesi belirleme sorununu çözdükten sonra Planlı ve Ters Planlı yaklaşımlar birçok farklı alanda uygulanabilir olmuştur. Böylece çeşitli uygulama alanlarında deneyler yapılmıştır. Planlı ve Ters Planlı Öğrenme yaklaşımları klasik yöntemle karşılaştırılmıştır. Karşılaştırmaların sonuçları, girdi örneklerini rastgele sırayla değil, zorluk seviyelerine ilişkin bir sırayla sunmanın birçok uygulama alanında etkili bir prosedür olduğunu göstermiştir. Planlı Öğrenmeyle 11 veri kümesinde klasik yöntemden daha iyi sonuçlar elde edilirken Ters Planlı Öğrenmeyle 14 veri kümesinde daha iyi sonuçlar elde edilmiştir.

Planlı Öğrenme yaklaşımının eğitim sonunda daha iyi bir yerel minimum elde ettiği düşünülmektedir. Bu yaklaşımın nasıl daha iyi bir performans gösterdiği ve arka planında neler olabileceği hakkında bazı çalışmalar [24,25] vardır. Öte yandan Planlı Öğrenmede olduğu gibi öğreniciye gittikçe zorlaşan örnekler verildiğinde daha iyi doğruluk oranı elde etmek akla uygun ve mantıklıdır ancak bu çalışmanın sonuçlarına göre Ters Planlı Öğrenme yaklaşımı da klasik yönteme göre iyi ve hatta Planlı Öğrenmeden daha başarılıdır. Ters Planlı Öğrenmenin neden daha iyi sonuçlar elde ettiği araştırılması gereken bir konudur.

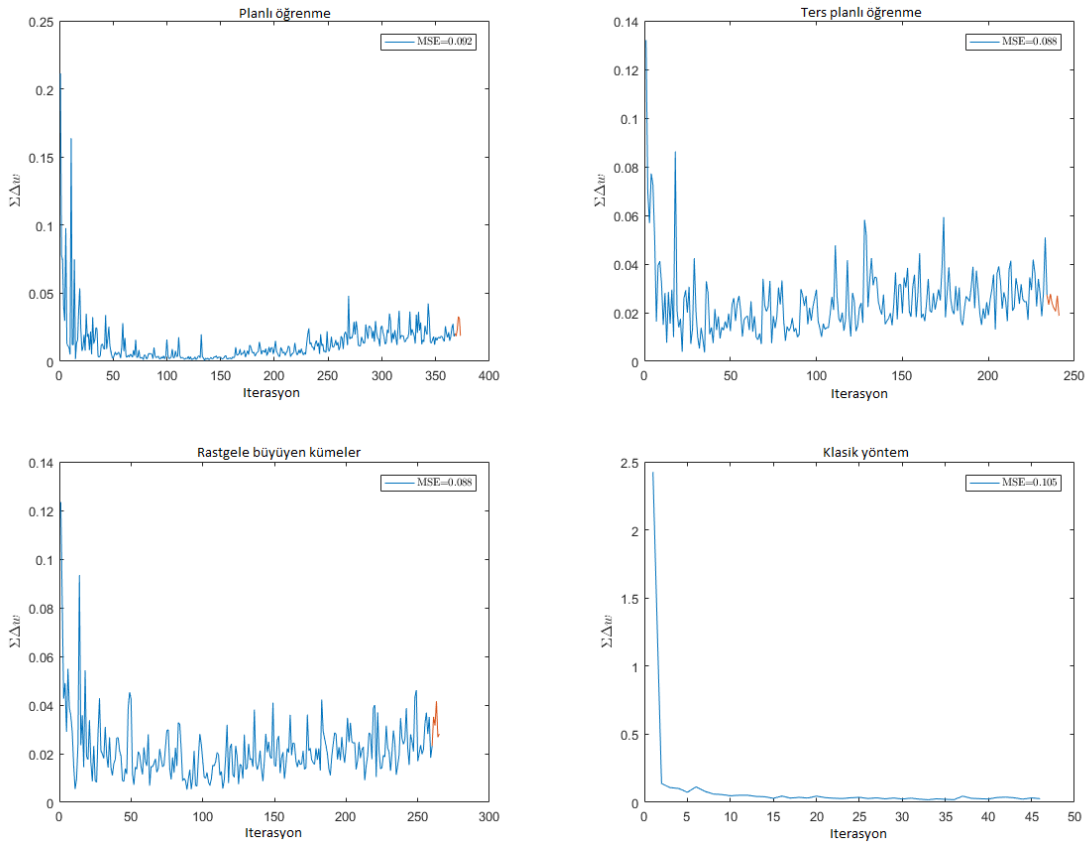
3.5 Planlı Öğrenme Neden Optimizasyon Sağlar?

Literatürdeki çalışmalarda hem Planlı Öğrenme ve Kendini Planlayan Öğrenme hem de bunların ters versiyonlarının başarılı olduğu görülmüştür. Yöntemlerin düz ve ters versiyonlarının ortak noktalarını bulmak için eğitim sırasında her iterasyonda ağırlıkların toplamının değişimi incelenmiştir. Gizli katmanında 10 nöron bulunan 3 katmanlı bir Yapay Sinir Ağında, araç (vehicle) veri kümesi için denenen tüm yöntemlerin grafikleri Şekil 3.5'te verilmiştir. Bu grafiklerde toplam ağırlık değişiminin her aşamanın başında birden artmakta ve aşama süresince azalmakta olduğu görülmektedir. Aşamaların başında optimizasyon yüzeyinde büyük bir adım atılması takılmış olunması muhtemel bir yerel minimumdan kaçınmayı sağlayan bir özellik olabilir. Her aşamada yeni bir grup eklenmesinin önceki aşamadaki minimumdan büyük bir adımla kaçmayı sağladığı düşünülebilir.

Uygulanan yöntemlerin örneklerin sıralanmasıyla ilgili bir ortak özelliği yoktur. Sadece her aşamada eğitim kümesinin yeni bir grup eklenerek büyümesi yerel minimumdan kaçınılarak optimizasyona devam etmeyi sağlamaktadır. Bu durumda sıralama yapmadan sadece büyüyen kümelerle eğitim de daha iyi bir minimum bulmayı sağlıyor olabilir.

Şekil 3.5'te tüm eğitim kümesinin Yapay Sinir Ağına verildiği son aşamadaki iterasyonlar turuncu ile işaretlenmiştir. Aşamalı eğitim yapılan tüm yöntemlerde bütün eğitim kümesinin verildiği aralıkta klasik yöntemden daha kısa sürede minimum bulunmaktadır.

Bu durum büyüyen eğitim kümeleri ile optimizasyonun bu aşamaya kadar daha iyi bir minimuma getirildiğine işaret etmektedir. Büyüyen kümeler yöntemi tüm eğitim kümesinin verildiği aşamada rastgele ağırlıklarla başlatmaktan daha iyi bir başlangıç koşulu sağlamaktadır.



Şekil 3.5 Tüm yöntemler için her iterasyonda toplam ağırlık değişimi

BÖLÜM 4

BÜYÜYEN KÜMELERLE EĞİTİM

Bu bölümde Planlı ve Ters Planlı Öğrenme yöntemlerinin ortak özelliği olarak eğitimin büyüyen kümelerle gerçekleştirilmesinin Yapay Sinir Ağlarının dışbükey olmayan fonksiyonlarının optimizasyonunda etkili olduğu önerilmiştir. Büyüyen kümeler yönteminin teorik açıklamasına yer verilmiş ve uygulama kısmında önceki çalışmalarla karşılaştırılarak sonuçları incelenmiştir.

4.1 Büyüyen Kümelerle Eğitim Nedir?

Bengio ve arkadaşlarının [11], eğitim sırasında zorlukla ilgili bir sıralama izlemenin dışbükey olmayan hedef fonksiyonlar için bir optimizasyon sağladığını Planlı Öğrenme adıyla sunmasından sonra birçok araştırmacı bu yaklaşımla en iyi verimi elde etmek için en etkili planlamayı bulmaya çalışmıştır.

Bazı durumlarda kolaydan zora sıralamaya gürültüler ekleyerek daha yüksek performans elde edilebilmektedir. Jiang ve ark. [16] hem kolay hem de farklı örnekleri öncelik vererek Kumar ve arkadaşlarının algoritmasına [15] göre daha iyi performans elde etmişlerdir. Chang ve ark. [26] belirsiz, emin olunmayan örneklere ağırlık vermeyi önermiş ve bunun daha doğru ve sağlam bir SGD optimizasyonu sağladığını iddia etmişlerdir. Avramova yüksek lisans tezinde [27] SPL ve SPLD' nin ters versiyonlarını incelemiş ve bu yöntemlerin düzlerinden daha iyi bir performans gerçekleştirdiğini göstermiştir. Bu araştırmalar zihinlere bir soru getirmektedir: Örnekleri kolaydan zora

sıralayarak daha iyi sonuçlar elde etmek mantıklı ve doğal olmasına rağmen aynı zamanda zordan kolaya sıralamanın da iyi olmasının nedeni nedir?

Bu bölümde, küçük bir eğitim kümesi ile başlayıp her aşamada yeni örnekler eklemenin Planlı ve Ters Planlı Öğrenme yaklaşımlarının ortak özelliği olmasından yola çıkılarak bu özelliğin bir optimizasyon sağladığını gösteren bir çalışmaya yer verilecektir. Dolayısıyla, anlamlı bir sıralama olmadan sadece aşamalı olarak yeni örnekler eklemenin daha iyi sonuçlar elde etmeyi mümkün kıldığı düşünülmektedir. Önerilen yöntem, her aşamada eğitim kümesine yeni bir grup eklenerek gerçekleştirilmiş ve önceki iki stratejiyle karşılaştırılmıştır. Birinci strateji, zorluk seviyelerinin ön bilgi olarak verildiği Planlı Öğrenme, ikincisi ise örneklerin zorluk seviyelerini her aşamada ağın kendisinin belirlediği Kendini Planlayan Öğrenmedir. Her iki strateji ile kolaydan zora ve zordan kolaya sıralanan örneklerle gerçekleştirilen eğitim yöntemlerinin yanı sıra normal tek aşamalı artımsal eğitim (incremental training) ile eşleştirilmiş T-testi (paired T-test) kullanılarak karşılaştırılacak ve sonuçlar incelenecektir.

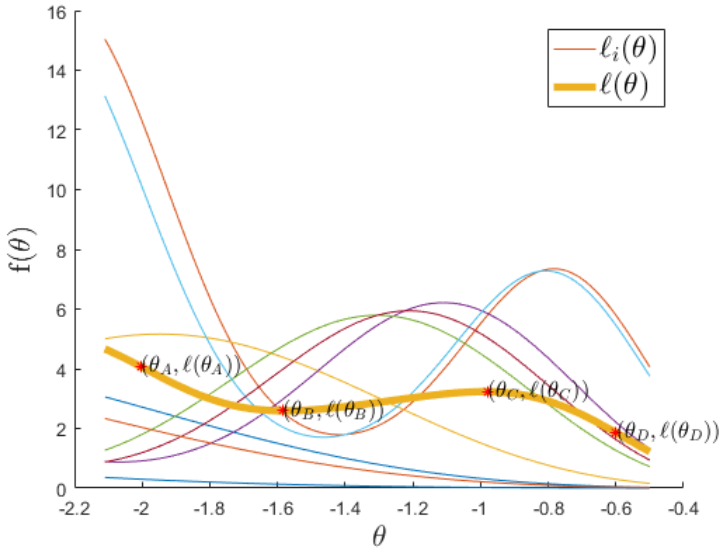
4.2 Teorik Bir Açıklama ile Büyüyen Kümeler Yöntemi

Tanım 4.1 N toplam eğitim örneği sayısı olmak üzere tek bir θ ϵ R parametresine sahip ℓ(θ) kayıp fonksiyonu Denklem 4.1'de verilmiştir. Her bir örnek için kayıp fonksiyonu $\ell_i(\theta)$ ise Denklem 4.2'de verilmiştir.

$$\ell(\theta) = \frac{1}{N}\sum_{i=1}^{N}(f(x_i,\theta) - y_i)^2 \tag{4.1}$$

$$\ell_i(\theta) = (f(x_i,\theta) - y_i)^2 \tag{4.2}$$

Tanım 4.2 θ_B noktası ℓ(θ) fonksiyonunun kesin bir yerel minimumu olarak bilinmektedir. Şekil 4.1'de kesin bir yerel minimum noktasının kayıp fonksiyonları ile geometrik gösterimi verilmiştir. 10 farklı örneğin her biri için kayıp fonksiyonları ince çizgilerle ve ortalamaları kalın çizgi ile gösterilmiştir. ℓ(θ)'nın beklenen değeri E[ℓ(θ)] olarak gösterilirse, verilen noktalarda kayıp fonksiyonun beklenen değerleri Denklem 4.3'teki gibi sıralanır.



Şekil 4.1 Her bir örneğin (ince çizgiler) ve ortalamalarının (kalın çizgi) kayıp fonksiyonları

$$\mathrm{E}[\ell(\theta_A)] > \mathrm{E}[\ell(\theta_C)] > \mathrm{E}[\ell(\theta_B)] > \mathrm{E}[\ell(\theta_D)] \tag{4.3}$$

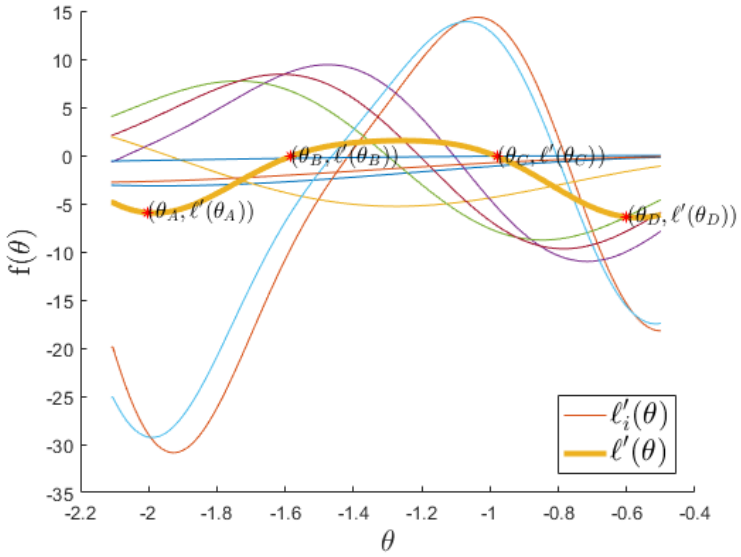
Kayıp fonksiyonlarının türevlerinin verilen noktalardaki beklenen değerleri aşağıdaki denklemlerde verilmiş ve türevlerin grafikleri Şekil 4.2'de gösterilmiştir.

$$\ell'(\theta_A) = E\left[\frac{1}{N}\sum_{i=1}^{N}\ell_i{}'(\theta_A)\right] < 0 \tag{4.4}$$

$$\ell'(\theta_B) = E\left[\frac{1}{N}\sum_{i=1}^{N}\ell_i{}'(\theta_B)\right] = 0 \tag{4.5}$$

$$\ell'(\theta_C) = E\left[\frac{1}{N}\sum_{i=1}^{N}\ell_i{}'(\theta_C)\right] = 0 \tag{4.6}$$

$$\ell'(\theta_D) = E\left[\frac{1}{N}\sum_{i=1}^{N}\ell_i{}'(\theta_D)\right] < 0 \tag{4.7}$$



Şekil 4.2 Her bir örneğin (ince çizgiler) ve ortalamalarının (kalın çizgi) kayıp fonksiyonlarının birinci türevleri

Tanım 4.3 Kayıp fonksiyonunun karmaşıklığı fonksiyondaki kritik noktaların sayısı olarak tanımlanıp K ile ifade edilmek suretiyle ℓ(θ)'nın karmaşıklığı K[ℓ(θ)] olarak gösterilir.

Varsayım 4.1 Her bir bireysel örneğin karmaşıklığı en fazla tüm örneklerin ortalama fonksiyonun karmaşıklığı kadar olabilir.

$$\mathrm{K}[\ell_i(\theta)] <= \mathrm{K}[\ell(\theta)] \qquad \forall\, \mathrm{i} \in \{1,2,3,\dots \mathrm{N}\} \qquad (4.8)$$

Varsayım 4.2 Örneklerin yarısından fazla bir sayı olan m tane örneğin bireysel kayıp fonksiyonu ortalama kayıp fonksiyonundan daha düşük karmaşıklığa sahiptir.

$$\mathrm{K}[\ell_i(\theta)] < \mathrm{K}[\ell(\theta)] \qquad (m > \frac{N}{2}) \qquad (4.9)$$

Lemma 4.1 θ_B noktasındaki türevlerin Olasılık Yoğunluk Fonksiyonu (Probability Density Function, PDF) sıfır ortalama ve çarpık (skewed) dağılıma sahiptir.

İspat: Şekil 4.2'deki türevler bireysel örneklerin yarısından fazlası için θ_B noktasının altında kalmaktadır. Ancak θ_B noktasının yukarısında kalan örneklerin $\ell_i'(\theta_B)$ değerleri

yüksektir. Türevlerin θ_B noktasındaki olasılık yoğunluk fonksiyonu $\mathrm{PDF}(\ell_i'(\theta_B))$ ile gösterilirse,

- Tanım 4.1.'e göre θ_B nin bir yerel minimum olması dolayısıyla $\mathrm{PDF}(\ell_i'(\theta_B))$ sıfır ortalamaya sahiptir.
- Şekil 4.2'de olduğu gibi K[ℓ(θ)]=2 ise Varsayım 4.1.'den K[ℓ$_i$(θ)] <= 2 olur ve Varsayım 4.2'den örneklerin yarısından fazlası için K[ℓ$_i$(θ)] < 2 'dir. K[ℓ$_i$(θ)] < 2 olan örnekler için $\ell_i'(\theta_B) < 0$ 'dır. Örneklerin yarısından fazlası için $\ell_i'(\theta_B) < 0$ geçerli olması durumunda $\mathrm{PDF}(\ell_i'(\theta_B))$ 'nin çarpık dağılıma sahip olduğu söylenir.

Corollary 4.1 Eğitim kümesinden n sayıda elemanı olan bir alt küme alınırsa bu alt kümenin kayıp fonksiyonu aşağıdaki gibi yazılabilir.

$$\ell_s(\theta_B) = \frac{1}{n}\sum_{i=1}^{n} \ell_i(\theta_B) \tag{4.10}$$

Kayıp fonksiyonunun türevinin θ_B noktasında sıfırdan küçük olduğu bir alt küme seçme olasılığı sıfırdan büyük olduğu bir alt küme seçme olasılığından büyüktür. Olasılık Pr ile ifade edilirse Denklem 4.11 bu durumu göstermektedir.

$$\mathrm{Pr}(\ell_s'(\theta_B) < 0) \; > \; \mathrm{Pr}(\ell_s'(\theta_B) > 0) \tag{4.11}$$

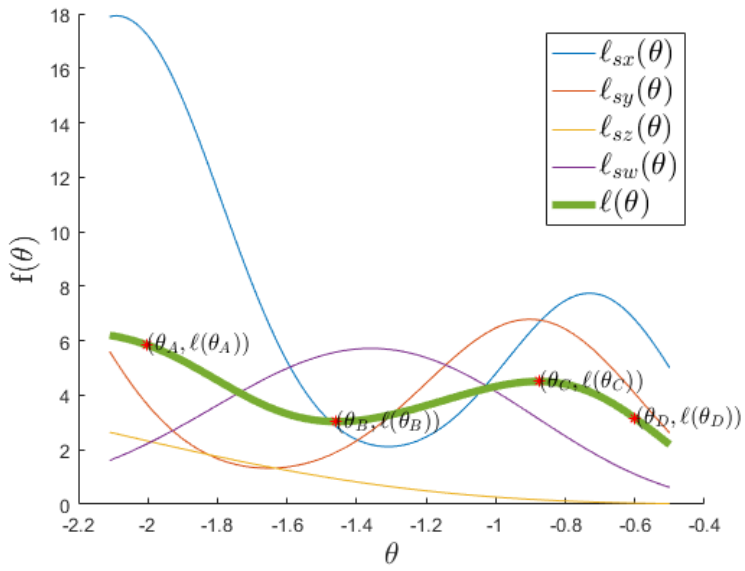
İspat: Varsayım 4.2'ye göre kayıp fonksiyonunun türevi θ_B noktasının altında olan örneklerin sayısı fazla olduğundan seçilen bir örneğin bu gruptan gelme olasılığı daha yüksektir. θ_B bir yerel minimum olduğundan $\mathrm{E}[\ell_s'(\theta_B)]$=0 'dır. Yani çok sayıda düşük değerli örneğin kayıp fonksiyonunun türevi θ_B noktasında negatifken, az sayıda yüksek değerli örneğin kayıp fonksiyonunun türevi θ_B noktasında pozitiftir.

Teorem 4.1 n örnekli bir alt küme alınıp toplu metotla gradyan düşümü kuralıyla (batch gradient descent) eğitilirse bu alt küme için tüm eğitim örnekleri ile eğitildiğinde bulunduğundan daha iyi bir yerel minimum bulunur.

İspat: $\ell(\theta)$ yüzeyinde θ_A noktasından başlayan bir optimizasyon büyük ihtimalle θ_B noktasında yakınsayacaktır. Ancak optimizasyon örneklerin alt kümesi için yapılıp $\ell_s(\theta)$ yüzeyinde gerçekleşirse Çıkarım 4.1'in getirisi olarak mevcut alt küme için daha iyi bir minimum bulmak üzere devam edecektir.

Teorem 4.2 Tüm örneklerle optimizasyona devam etmek genel olarak bütün örnekler için alt kümenin minimumundan daha iyi bir minimum bulmayı sağlar.

İspat: Tüm örnekler için hata yüzeyi $\ell(\theta)$ ve bazı örnek alt kümeleri için hata yüzeyleri $\ell_{sx}(\theta), \ell_{sy}(\theta), \ell_{sz}(\theta), \ell_{sw}(\theta)$ Şekil 4.3'te verilmiştir. Bu örnekler $\theta_A, \theta_B, \theta_C$ noktaları için optimizasyonun yakınsaması ile ilgili tüm olası durumları göstermektedir.



Şekil 4.3 Örnek alt kümeleri ve tüm örneklerin kayıp fonksiyonları

Optimizasyona θ_A noktasından başlandığı düşünülürse,

1. Eğer tüm örneklerin fonksiyonu $\ell(\theta)$ kullanılırsa θ_B noktasında yakınsar.
2. Eğer S_x alt kümesinin fonksiyonu $\ell_{sx}(\theta)$ kullanılırsa $\theta_B < \theta_{sx} < \theta_C$ olmak üzere bir θ_{sx} noktasında yakınsar. Eğer optimizasyon burada durdurulursa tüm örneklerin yüzeyindeki hata daha yüksek olacaktır ($E[\ell(\theta_{sx})] > E[\ell(\theta_B)]$). Bu durumda θ_{sx} noktasından başlayarak $\ell(\theta)$ yüzeyinde optimizasyona devam ederek daha kötü bir noktada kalmaktan korunulur.

3. Eğer S_y alt kümesinin fonksiyonu $\ell_{sy}(\theta)$ kullanılırsa $\theta_A < \theta_{sy} < \theta_B$ olmak üzere bir θ_{sy} noktasında yakınsar. Burada $E[\ell(\theta_{sy})] > E[\ell(\theta_B)]$ olduğundan tüm örnekler için tekrar bir optimizasyon gereklidir.
4. Eğer S_z alt kümesinin fonksiyonu $\ell_{sz}(\theta)$ kullanılırsa $\theta_C < \theta_{sz}$ olmak üzere bir θ_{sz} noktasında yakınsar. Bu durumda θ_{sz} noktasından başlayarak $\ell(\theta)$ yüzeyinde optimizasyona devam edilirse tüm örnekler için daha iyi bir yerel minimum noktasında yakınsama olasılığı vardır.
5. Eğer S_w alt kümesinin fonksiyonu $\ell_{sw}(\theta)$ kullanılırsa $\theta_A > \theta_{sw}$ olmak üzere bir θ_{sw} noktasında yakınsar. Bu durumda θ_{sw} noktası tüm örnekler için iyi bir minimum olmayabilir ve $\ell(\theta)$ yüzeyinde optimizasyon yapmak bu kötü minimumdan kaçınmayı sağlayabilir.

Teorem 4.3 Büyüyen kümeler ile eğitim gerçekleştirilirse daha iyi bir yerel minimum elde edilebilir.

İspat: Teorem 4.2'de verildiği üzere alt kümeden sonra daha büyük eğitim kümesi ile eğitime devam etmek daha iyi bir yerel minimum bulmayı sağlar. Benzer şekilde alt kümenin alt kümesinin yüzeyi de alt küme için daha iyi bir minimum bulmayı sağlayabilir. Tüm eğitim kümesinin hata yüzeyini optimize etmek için öncelikle aşağıdaki yüzeyler sırasıyla optimize edilmelidir.

$$\ell_{S_1}(\theta), \ell_{S_2}(\theta), \ell_{S_3}(\theta), \dots, \ell(\theta) \qquad s(S_1) < s(S_2) < s(S_3) < \cdots < N \qquad (4.12)$$

4.3 Karşılaştırılan Yöntemlerin Algoritmaları

Bu bölümde 2 farklı strateji ile belirlenen 2 sıralama türünün ve sıralamasız olarak büyüyen kümeler yönteminin nasıl uygulandığı açıklanacaktır. 2 farklı strateji örneklerin zorluk seviyesinin önceden belirlenip öğreniciye ön bilgi olarak verildiği Planlı Öğrenme ve örneklerin zorluk seviyesinin her aşamada öğrenici tarafından belirlendiği Kendini Planlayan Öğrenme stratejileridir. Her iki strateji ile hem kolaydan zora sıralama hem de zordan kolaya sıralama türleri incelenmiştir.

4.3.1 Planlı Öğrenme

Planlı öğrenmede öğrenici eğitim sırasında önceden belirlenmiş örnek sıralamasını izler. Bu nedenle örneklerin zorluk seviyesinin bilinmesi gereklidir. Bu bilgi, yapılacak işe özel sıralamayı tanımlayarak veya başlangıçta elle/otomatik olarak örnekleri etiketleyerek elde edilebilir. Örnekleri zorluk seviyeleri ile etiketleme işi yapay veri kümeleri için kolay fakat ve gerçek dünyadan alınmış veri kümeleri için masraflı ve zordur. Ayrıca insanlara göre kolay olan bir örnek makinelere göre kolay olmayabilir. Bu çalışmada örneklerin zorluğunu otomatik olarak belirlemek için bir Kolektif öğrenme (Ensemble learning) yöntemi kullanılmıştır. Bölüm 3.3.1'dekine benzer şekilde Breiman'ın önerdiği [22] yöntem ile bir sınıflandırıcı topluluğu oluşturulmuş ve her eğitim örneğinin zorluk seviyeleri buna göre belirlenmiştir. Ardından kolaydan zora sıralanan eğitim kümesi Planlı Öğrenme (Curriculum Learning, CL) için ve zordan kolaya sıralanan eğitim kümesi Ters Planlı Öğrenme (Anti Curriculum Learning, ACL) için gruplar halinde ele alınmıştır.

CL için örnekler kolaydan zora sıralanır ve aynı sayıda örnekten oluşan gruplara ayrılır. Her gruptaki örnek sayısı toplam eğitim örneği sayısının aşama sayısına bölünmesi ile elde edilir. Kalan örnekler ilk aşamaya eklenir. Böylece her aşamada eklenecek yeni örnek sayısı eşittir. Eğitime en kolay gruptan başlanır ve gruplar kolaydan zora tek tek eklenerek devam edilir. Son aşamada en zor örneklerden oluşan grup ile birlikte tüm eğitim kümesi verilir ve eğitim belirlenen sayıda aşamada tamamlanır.

ACL için eğitim örnekleri zorluk seviyesi azalacak şekilde sıralanır ve belirlenen sayıda gruba ayrılır. Eğitim en zor olan grupla başlar ve son aşamada eklenen grup en kolay örnekleri içerir. Çizelge 4.1'de eğitim kümesi sıralanmış olarak düşünüldüğünde n aşamalı CL ve ACL için her aşamada verilen gruplar işaretlenmiştir.

Çizelge 4.1 CL ve ACL'nin her aşamasında eğitim kümeleri

	Eğitim kümesi				
	En kolay	**…**	**Orta**	**…**	**En zor**
Aşama 1	Grup 1	Grup 2	…	…	Grup n
Aşama 2	Grup 1	Grup 2	…	…	Grup n
…	Grup 1	Grup 2	…	…	Grup n
…	Grup 1	Grup 2	…	…	Grup n
Aşama n	Grup 1	Grup 2	…	…	Grup n

Her iki yöntem için her adımda önceki örneklerle birlikte yeni bir örnek grubu verilir. İki yöntemin de son aşamasında klasik eğitim gibi tüm eğitim kümesi verilir. CL için izlenen Algoritma 4.1'de verilmiştir. Aynı algoritma zordan kolaya sıralı eğitim kümesi kullanılarak ACL için de uygulanmıştır.

Planlı Öğrenmenin ilk aşamasında eğitim rastgele atanan ağırlıklar ile başlar. Her bir sonraki aşamanın başlangıç ağırlıkları bir önceki aşamada bulunan optimum ağırlıklar olarak atanır. Başka bir deyişle bir sonraki aşamadaki optimizasyon, önceki aşamada bulunan minimumdan başlar.

Algoritma 4.1 Planlı Öğrenme

1: D ← kolaydan zora sıralanmış eğitim kümesi

2: n ← aşama sayısı

3: s ← her aşamada eklenecek örnek sayısı

4: f ← yapay sinir ağının θ parametresine sahip dışbükey olmayan hedef fonksiyonu

5: **prosedür** CL (D, n, s, f)

6: Başlangıç ağırlıkları θ_0 'ı rastgele ayarla

7: D_0 = D'nin ilk s tane örneği

8: $\theta_1 = argmin_\theta f(D_0, \theta_0)$

9: **döngü** t=1; t<=n-1; t++ **ise**

10: D_t = D'nin ilk (t+1) * s tane örneği

11: $\theta_{t+1} = argmin_\theta f(D_t, \theta_t)$

12: **döngü bitti**

13: θ_n 'i **döndür**

14: **prosedür bitti**

4.3.2 Kendini Planlayan Öğrenme

Kendi Planlayan Öğrenme (Self paced Learning, SPL), Planlı Öğrenme için zorluk seviyelerini bulmak amacıyla geliştirilen bir çözümdür. Bu yöntemde öğrenici, öğreneceği örnekleri hedef fonksiyonunun mevcut durumuna göre belirler. Eğitim,

rastgele seçilen bir grup örnek ile başlar ve bu örneklerle eğitilmiş mevcut model uzayına en uygun örnekler kolay olarak etiketlenir. Ağırlıklar bir sonraki aşamada kolay olarak etiketlenmiş örneklerle güncellenir. Ağ, her bir sonraki adımda daha fazla örnekle eğitilir ve en son adımda tüm eğitim kümesi verilir. Bu, her aşamada kendini planlama parametresinin (self pace parameter) ayarlanmasıyla gerçekleştirilir. Eğitimin her aşamasını homojen ilerleyecek şekilde ayarlamak için her aşamada aynı sayıda örnek eğitim kümesine eklenebilir.

Bu çalışmada SPL için ayarlanan tek parametre aşama sayısıdır. Her aşamada eklenecek örnek sayısı CL 'deki gibi eşittir. Eğitim rastgele seçilen örneklerle başlar. Her aşamanın sonunda örnekler ortalama karesel hata (mean squared error, MSE) ile hesaplanan kayıplara göre azdan çoğa doğru sıralanır. Bir sonraki aşamada verilecek örnekler bu sıralamadan seçilir. Model uzayı ile en tutarlı örnekler bir sonraki aşamada kullanılmak üzere seçilir.

Kendini Tersten Planlayan Öğrenme (Self paced learning Inversed, SPLI) için, en zor örneklerin önceliği vardır. SPL ile benzer şekilde eğitim rastgele seçilen örneklerle başlar. Bir sonraki aşamadaki eğitim örnekleri model uzayı ile uyuşmazlık dikkate alınarak seçilir. Her aşamanın sonunda tüm eğitim kümesi kayıplara göre çoktan aza doğru sıralanır. Tahmin hatası yüksek olan örnekler bir sonraki aşamada kullanılır. SPL'in uygulanmasında izlenen yol Algoritma 4.2'de verilmiştir. Aynı algoritma eğitim kümesi her aşamada tersten sıralanarak SPLI için de uygulanmıştır.

SPL'de bir sonraki aşamadaki eğitim, CL'de de olduğu gibi önceki aşamadaki optimum ağırlıklar kullanılarak başlatılmıştır. SPL önceki aşamadaki tüm örneklerin bir sonraki aşamada tekrar alınacağını garanti etmez. Tüm eğitim örnekleri arasından hedef fonksiyona en iyi uyan örnekler alınır ve önceki aşamada kullanılan örneklerden bazıları bu örneklerin içinde olmayabilir. Örneğin, eğer örneklerin rastgele seçildiği ilk aşamada gürültülü örnekler varsa bu örnekler ikinci aşamadaki eğitim kümesinde muhtemelen olmayacaktır.

Algoritma 4.2 Kendini Planlayan Öğrenme

1: T ← sıralanmamış eğitim kümesi

2: n ← aşama sayısı

3: s ← her aşamada eklenecek örnek sayısı

4: f ← yapay sinir ağının θ parametresine sahip dışbükey olmayan hedef fonksiyonu

5: **prosedür** SPL (T, n, s, f)

6: Başlangıç ağırlıkları θ_0 'ı rastgele ayarla

7: T_0 = T'den rastgele seçilmiş s tane örnek

8: $\theta_1 = argmin_\theta f(T_0, \theta_0)$

9: **döngü** t=1; t<=n-1; t++ **ise**

10: $D_t = f(T, \theta_t)$ ile kayıplarına göre azdan çoğa sıralanmış eğitim kümesi

11: T_t = D_t'nin ilk (t+1) * s tane örneği

12: $\theta_{t+1} = argmin_\theta f(T_t, \theta_t)$

13: **döngü bitti**

14: θ_n 'i **döndür**

15: **prosedür bitti**

4.3.3 Rastgele Düzenli Büyüyen Kümeler

Bu tez kapsamında geliştirilen Büyüyen Kümelerle Eğitim yönteminin çıkış noktası CL, SPL ve ters versiyonlarının sahip olduğu ortak noktanın keşfedilmesi olmuştur. Bu ortak nokta her bir sonraki adımda yeni bir grup ilavesi ile eğitim kümesinin büyüyor olmasıdır. Her iki versiyonun (kolaydan zora ve zordan kolaya) klasik yöntemden daha iyi bir performans sergilediği düşünülerek, bu ortak özelliğin bir optimizasyon sağladığı tahmin edilmiştir. Planlı Öğrenmeden yola başlayarak Büyüyen Kümeler fikrinin ortaya çıkması Bölüm 3.5'te açıklanmıştır. Bu durumda örneklere anlamlı bir düzen vererek eğitmek yerine sadece birbiri üzerine birikerek büyüyen gruplarla eğitmek önemli olmaktadır. Bu nedenle eğitim kümesini sırasız gruplar halinde ele almak ve her aşamada yeni bir grup eklemek de öğrenicinin performansını artırabilir.

Rastgele Düzenli Büyüyen Kümeler (Random Ordered Growing Sets, ROGS) yönteminde, sıralanmamış eğitim kümesi eşit sayıda örnek içeren gruplara ayrılmıştır. Eğitim ilk aşaması SPL gibi rastgele seçilmiş bir grup ile başlar ve daha sonra her aşamada anlamlı bir düzen olmadan yeni bir rastgele grup eklenir. Algoritma 4.3, Rastgele Düzenli Büyüyen Kümeler yönteminde izlenen yolu göstermektedir. ROGS yönteminde diğer yöntemlerle aynı olarak sonraki aşamadaki başlangıç ağırlıkları önceki aşamada bulunan optimum ağırlıklar olarak ayarlanmıştır.

Algoritma 4.3 Rastgele Düzenli Büyüyen Kümeler

1: T ← sıralanmamış eğitim kümesi
2: n ← aşama sayısı
3: s ← her aşamada eklenecek örnek sayısı
4: f ← yapay sinir ağının θ parametresine sahip dışbükey olmayan hedef fonksiyonu
5: **prosedür** ROGS (T, n, s, f)
6: Başlangıç ağırlıkları θ_0 'ı rastgele ayarla
7: T_0 = T'den rastgele seçilmiş s tane örnek
8: $\theta_1 = argmin_\theta f(T_0, \theta_0)$
9: **döngü** t=1; t<=n-1; t++ **ise**
10: T_t = T'den rastgele (t+1) * s tane örnek seç
11: $\theta_{t+1} = argmin_\theta f(T_t, \theta_t)$
12: **döngü bitti**
13: θ_n 'i **döndür**
14: **prosedür bitti**

Bu bölümde incelenen 5 yöntem aşağıda listelenmiştir.

- Planlı Öğrenme (Curriculum Learning, CL)
- Ters Planlı Öğrenme (Anti Curriculum Learning, ACL)
- Kendini Planlayan Öğrenme (Self Paced Learning, SPL)
- Kendini Tersten Planlayan Öğrenme (Self Paced Learning Inversed, SPLI)

- Rastgele Düzenli Büyüyen Kümeler (Random Ordered Growing Sets, ROGS)

Tüm bu yöntemlerde optimum ağırlıklar Denklem 4.13'te verilen aynı hedef fonksiyonu ile bulunmaktadır.

$$\theta_{t+1} = argmin_{\theta \epsilon R^d} \sum_{i=1}^{(t+1)*s} (f(x_i, \theta_t) - y_i)^2 \quad (4.13)$$

Burada s her aşamada eklenecek örnek sayısını, tüm yöntemlerde ağırlıklar her aşamada aynı sayıda örnek tarafından belirlenir. CL için, x_i ve y_i önceden kolaydan zora sıralanmış eğitim kümesinden alınır. ACL için önceden zordan kolaya sıralanmış eğitim kümesinden alınır. SPL için her aşamada öğrenicinin kendisi eğitim kümesini kolaydan zora sıralayıp örnek seçer. SPLI için ise her aşamada öğreniciye göre zordan kolaya sıralanan eğitim kümesinden sırayla örnek alınır. ROGS yönteminde örnekler sıralama yapılmamış eğitim kümesinden rastgele seçilir.

4.4 Yöntemlerin Uygulanması ve Karşılaştırma Sonuçları

Gizli katmanında 10 nöron bulunan üç katmanlı bir Yapay Sinir Ağı, tüm yöntemler için SGDM algoritması ile eğitilmiştir. Ağın parametreleri öğrenme katsayısı 0,01 ve momentum katsayısı 0,9 olarak ayarlanmıştır. Aktivasyon fonksiyonları olarak girdi katmanından ara katmana geçişte tanjant-sigmoid, ara katmandan çıktı katmanına geçişte lineer transfer fonksiyonu uygulanmıştır. Durma koşulu 6 iterasyon boyunca validasyon kümesinde hesaplanan hata oranının yükselmesidir. Klasik yöntem ve tüm aşamalı eğitim yöntemlerinde her örnekte ağırlıkların güncellendiği artımsal eğitim uygulanmıştır. Tüm aşamalı yöntemler için aşama sayısı 25 olarak ayarlanmıştır. Ağ her bir aşamadan bir sonraki aşamaya geçmek için durma koşulunu sağlamalıdır. Aşama sayısı belirlenirken aşamalı eğitimin karakteristiğinin daha belirgin olmasını sağlamak için yapılan denemeler neticesinde bu değerin uygun olduğu tespit edilmiştir. Önceki aşamada optimize edilmiş ağırlıklar bir sonraki aşamada başlangıç ağırlıkları olarak verilir.

Önerilen yöntemler, UCI veritabanından alınan 36 veri kümesinde [23] denenmiştir. Veri kümelerinin örnek sayıları 57 ile 20.000 arasında değişmektedir. Her denemede veri kümesi 5 parçaya ayrılmış, 3'ü eğitim, 1'i validasyon ve 1'i test için ayrılmıştır. Her bir veri kümesi için, 4x5 kat çapraz doğrulama ile elde edilen 20 hata oranı (MSE), 0.95 anlamlılık düzeyli eşleştirilmiş T-testi ile karşılaştırılmıştır. Tüm veri kümeleri için Klasik yöntem ile yapılan karşılaştırmaların sonuçları Çizelge 4.2'de örnek sayısı, özellik sayısı, sınıf sayısı ve Klasik yöntemin ortalama hata oranı ile birlikte verilmiştir. İlgili metot Klasik yönteme karşı kazanmış ise (+), kaybetmiş ise (-) ile işaretlenmiştir.

Çizelge 4.2 Yöntemlerin Klasik metotla karşılaştırılması

Veri kümesi	Örnek sayısı	Özellik sayısı	Sınıf sayısı	MSE	CL	ACL	SPL	SPLI	ROGS
labor	57	26	2	0.15	+		+	+	+
zoo	84	16	4	0.02				+	
lymph	142	37	2	0.14					
iris	150	4	3	0.03		+		+	+
hepatitis	155	19	2	0.15					
audiology	169	69	5	0.08		+		+	+
autos	202	71	5	0.13	+	+		+	+
glass	205	9	5	0.12					
sonar	208	61	2	0.17				+	
heart-statlog	270	13	2	0.14		-			
breast-cancer	286	38	2	0.21		-		-	-
primary-tumor	302	23	11	0.08	+		+	+	+
ionosphere	351	33	2	0.10					
colic	368	60	2	0.15		-		-	-
vote	435	16	2	0.04	+				
balance-scale	625	4	3	0.06				+	
soybean	675	83	18	0.03	+		+	+	+
credit-a	690	42	2	0.12					
breast-w	699	9	2	0.03					
diabetes	768	8	2	0.16					
vehicle	846	18	4	0.09	+	+	+	+	+
anneal	890	62	4	0.01		+		+	+
vowel	990	11	11	0.06	+	+	+	+	+
credit-g	1000	59	2	0.20		-		-	-
col10	2019	7	10	0.05	+			+	+
segment	2310	18	7	0.02	+	+		+	+
splice	3190	287	3	0.03		-			
kr-vs-kp	3196	39	2	0.02		+		+	+
hypothyroid	3770	31	3	0.04		+			+
sick	3772	31	2	0.04		+		+	+
abalone	4153	10	19	0.04	+	+	+	+	+
waveform	5000	40	3	0.07		-			
d159	7182	32	2	3e-5	+	+	+	+	+
ringnorm	7400	20	2	0.10					
mushroom	8124	112	2	3e-5		+		+	
letter	20000	16	26	0.03	+			+	+

Tüm yöntemlerin birbirleri ile karşılaştırmalarının sonuçları Çizelge 4.3'te verilmiştir. (Kısaltmalar şu şekildedir: Planlı Öğrenme = CL, Ters Planlı Öğrenme = ACL, Kendini Planlayan Öğrenme = SPL, Kendini Tersten Planlayan Öğrenme = SPLI, Rastgele Düzenli Büyüyen Kümeler = ROGS). Her hücre ilgili satırda bulunan yöntemin ilgili sütunda bulunan yönteme karşı kazanma/berabere kalma/kaybetme bilgilerini içermektedir. Örneğin; Planlı Öğrenme, Klasik yönteme karşı 12 veri kümesinde kazanmakta, 24 veri kümesinde berabere kalmakta, hiçbir veri kümesinde kaybetmemektedir.

ROGS yönteminin 17 veri kümesinde Klasik yöntemden daha iyi olduğu görülmektedir. Klasik yöntemle karşılaştırmalar düşünüldüğünde ROGS yöntemi CL, ACL ve SPL'ye göre daha fazla veri kümesinde kazanmaktadır. Bu denemelerde anlamlı bir sıralama olmadan yalnızca eğitim kümesini büyüyen alt kümeler halinde ele almanın da daha iyi sonuçlar elde etmeyi sağladığı görülmüştür. Ayrıca, SPLI yöntemi Klasik yönteme karşı veri kümelerinin yarısından fazlasında kazanmaktadır. Bu yöntem toplam kazanma sayısı açısından en iyi yöntem olarak görünmektedir.

CL ve SPL yöntemlerinin hiçbir bir veri kümesinde Klasik yönteme karşı kaybetmemesi çarpıcıdır. Bu durum gürültülü veri kümelerinde bu yöntemlerin dayanıklı olduğunu göstermektedir. Zordan kolaya sıralı ve Rastgele Düzenli Büyüyen Kümeler yöntemlerinde gürültülere öncelik vermek mümkündür ve bu durum yüksek hata oranına sahip veri kümelerinde eğitimi yanlış yönlendirebilir. Kolay örneklere öncelik veren yöntemler ise bu tür veri kümelerinde daha güvenli bir eğitim sağlamaktadır.

Çizelge 4.3 T-test sonuçları

	Klasik	**CL**	**ACL**	**SPL**	**SPLI**	**ROGS**
Klasik	-	0/24/12	6/17/13	0/29/7	3/13/20	3/16/17
CL	12/24/0	-	11/20/5	4/32/0	4/20/12	7/24/5
ACL	13/17/6	5/20/11	-	7/18/11	0/24/12	1/30/5
SPL	7/29/0	0/32/4	11/18/7	-	3/18/15	5/24/7
SPLI	20/13/3	12/20/4	12/24/0	15/18/3	-	7/29/0
ROGS	17/16/3	5/24/7	5/30/1	7/24/5	0/29/7	-

4.5 Teorik ve Deneysel Bilgilerin Değerlendirilmesi

Kolaydan zora ve zordan kolaya sıralanan yöntemlerin her ikisinde de klasik yöntemden daha iyi performans görüldüğünden yola çıkılarak bu yöntemlerin ortak özellikleri araştırılmıştır. Her iki versiyonun ortak noktalarının 'eğitim kümesini kümülatif büyüyen kümeler halinde ele almak' olduğu düşünülmüştür. Bu nedenle örnekleri zorluk seviyesine göre sıralama aşamasını atlayıp yalnızca her aşamada bir grup rastgele örnek eklenerek eğitim yapılması önerilmiştir. Denemelerde zorluk seviyelerinin kullanıldığı Planlı, Ters Planlı, Kendini Planlayan ve Kendini Tersten Planlayan Öğrenme yöntemleri ile önerilen yöntem karşılaştırılmıştır. Elde edilen sonuçlara göre Planlı Öğrenme ve Kendini Planlayan Öğrenme yaklaşımlarının başarısının anlamlı bir sıralama izlediklerinden değil de büyüyen eğitim kümeleri ile eğitildiklerinden kaynaklandığı öne sürülebilir.

Şekil 4.1'de bazı bireysel örnekler ve ortalamalarının kayıp fonksiyonları verilmiştir. Optimizasyona θ_A' dan başlanmış ve (θ_A, $\ell(\theta_A)$) noktasının altında kalan örnekler kolay, yukarısındaki örnekler zor olarak düşünülmüştür. Bu noktada alınan kolay bir örneğin optimizasyonu daha iyiye veya daha kötüye doğru yönlendirmesi mümkündür ancak en kötü ihtimalle θ_B yerel minimum noktasında durulacaktır. Benzer şekilde zor bir örneğin alınması ile daha iyi bir sonuç elde edebilmek de edememek de mümkündür. Uygulama sonuçları da hem kolaydan zora hem de zordan kolaya sıralama yapılan yöntemlerin başarılı olabildiğini göstermiştir. Dolayısıyla örneklerin zorluk seviyesinin optimizasyona rehberlik etmede çok önemli olmadığı söylenebilir.

Şekil 4.2'de θ_B ve θ_C arasındaki mesafenin kısaldığı düşünülürse, yerel minimum noktasının kolayca atlatılabilmesi mümkündür. Bir eyer noktası için bu noktalar aynı olduğunda büyüyen kümelerle eğitim muhtemelen bu noktayı aşacak ve daha iyi bir minimum bulacaktır. Dauphin ve arkadaşlarının çalışmasında [28] belirtildiği gibi eyer noktalarının yüksek boyutlu fonksiyonlarda yerel minimumlardan çok daha fazla olduğu düşünülürse bu tip görevlerde büyüyen kümeler yönteminin iyi bir optimizasyon sağlayabileceği daha olası gözükmektedir.

Farklı dağılımlara sahip birçok veri kümesinde Planlı Öğrenmede örneklerin zorluk seviyelerini otomatik olarak belirlemek için Kolektif öğrenme yöntemi kullanılmıştır.

Zorluk seviyesi belirlemenin ön işlem gerektiriyor olmasının eğitimi yavaşlattığı düşünülebilir ancak Yapay Sinir Ağının eğitimi sırasında SPL'den daha hızlı bir performans görülmüştür. SPL yönteminde eğitimin her aşamasında bir sıralama yapıldığından eğitim süreci yaklaşık 1,5 kat daha uzun sürmektedir. Ayrıca Kolektif öğrenme yönteminin, Klasik yönteme karşı kazanma sayısı düşünülürse SPL' den daha iyi bir planlama elde edildiği söylenebilir.

CL ve SPL yöntemlerinin ters versiyonları olan ACL ve SPLI, yüksek hata oranına sahip bazı veri kümelerinde zayıf bir performans sergilemiştir. Eğitim süresince örneklerin farklı noktalarda verilmesinin etkisi Erhan ve ark. [29] tarafından incelenmiştir. Bu yöntemlerde gürültülü örneklerin başta verilmesinden dolayı sonucu daha fazla etkileyebileceği düşünülmektedir. Yine de yaklaşımların ters versiyonları standartlarından daha iyi bir performans göstermiştir. Ancak CL ve SPL yöntemleri herhangi bir veri kümesinde kaybetmemektedir ve bu durum sağlam bir özellik taşıdıklarını gösterir. Bu yöntemlerin gürültüye karşı dirençli olması hakkında teorik açıklamalar aranmıştır. Gong ve ark. [24] bu yöntemlerin neden özellikle büyük ve gürültülü veri kümelerinde etkili olduğunu araştırmışlardır. Klasik yönteme karşı en çok kazanan yöntem SPLI olmuştur. Bu yöntemde her aşamada öğrenilecek örnekleri seçme stratejisi, Lewis ve Gale [30] tarafından önerilen havuz tabanlı aktif öğrenmeyi (pool-based active learning) hatırlatmaktadır. Havuz tabanlı aktif öğrenmede de öğrenici etiketsiz veri havuzunun emin olunmayan örneklerini öğrenmek ister. Ayrıca CL ve SPL'nin kaybetmemesi ve ACL ve SPLI'nin daha çok kazanması, kolaylık tabanlı bir sıralama yerine değerli örnek tabanlı bir sıralama belirlemenin gerekliliğine işaret etmektedir.

Bu çalışma neticesindeki katkılar aşağıda listelenmiştir:

- CL ve SPL ile ilgili önceki çalışmalarda sınırlı alanlarda denemeler yapmıştır. Bu çalışmada ise bu yöntemlerin çeşitli ve çok sayıda veri kümesi üzerindeki performansı gösterilmiştir.
- CL ve SPL yöntemlerinin ters versiyonları olan ACL ve SPLI yöntemleri ile de geniş bir uygulama alanı üzerinde denemeler yapılmıştır.

- CL ve SPL'de olduğu gibi zorluk düzeylerinin belirlenmesine gerek olmayan dolayısıyla daha hızlı yeni bir yöntem önerilmiş ve önerilen yöntemin bu yöntemler kadar başarılı olduğu gösterilmiştir.
- Büyüyen kümeler ile eğitimin teorik perspektifi açıklanmıştır.
- Gelecekteki CL ve SPL çalışmalarında bir kıyaslama olarak ROGS yönteminin kullanılması önerilmektedir. Büyüyen eğitim kümelerinin üzerinde bir başarı varsa, ancak o zaman bunun doğru sıralamanın etkisi ile gerçekleştiği söylenebilir.

BÖLÜM 5

SONUÇ VE ÖNERİLER

Yapay Sinir Ağlarının öğrenme fonksiyonlarının optimize edilmesi için öne sürülen stratejilerden biri olan Planlı Öğrenme yaklaşımı ile insanların eğitim sürecine benzer şekilde eğitim örneklerinin kolaydan zora sıralanmasının makinelerin de öğrenme verimliliğini artırdığı iddia edilmektedir. Bu minval üzere eğitimi zorluk derecesine göre sıralanmış örneklerle gerçekleştirmek için örneklerin zorluk seviyesinin Kolektif öğrenme ile otomatik olarak belirlendiği bir yöntem önerilmiştir. Bu yöntem sayesinde birçok veri kümesi için uygulanabilir olan Planlı Öğrenme yaklaşımı ve ters versiyonunun öğrenme performansının arttırdığı görülmüştür.

Örneklerin kolaydan zora sıralanmasının yanı sıra zordan kolaya sıralanması ile de klasik yöntemden daha iyi sonuçlar elde edilmesi bu yöntemlerin ortak bir özelliğinin optimizasyon sağladığını akla getirmektedir. Bölüm 3.5'te eğitim sırasında her iterasyonda yöntemlerin parametrelerinin değişimi incelenmiş ve her aşama başında yeni örneklerin eklenmesi ile model uzayında mevcut kümenin yerel minimumundan kurtaracak bir adım atılabildiği görülmüştür. Bu adımı sağlayan etmenin örneklerin zorluk seviyesi ile ilgisiz olarak sadece yeni örnekler eklenmesi ile gerçekleştiği düşünülerek Bölüm 4'te rastgele gruplar halinde büyüyen kümelerle eğitim yaklaşımı önerilmiştir.

Büyüyen kümelerle eğitim yaklaşımının teorik açıklaması yapılmış ve çoğu durumda yerel minimumlardan kaçınmayı sağlayabildiği gösterilmiştir. Uygulama kısmında ise

Rastgele Düzenli Büyüyen Kümeler yöntemi, Klasik yöntemin yanı sıra eğitim örneklerinin önceden belirlenmiş zorluk derecelerine göre sıralandığı Planlı Öğrenme ve her aşamada öğrenicinin kendisinin belirlediği Kendini Planlayan Öğrenme ve bu yaklaşımların ters versiyonları ile karşılaştırılmıştır. Karşılaştırmalar sonucunda önerilen yöntemin Klasik yöntemden daha başarılı olduğu ve diğer yöntemlere karşı da çoğu açıdan üstünlük sağladığı görülmüştür.

Büyüyen kümelerle eğitim yönteminin teorik açıklamasında özellikle ilk aşamalarda seçilecek örneklerin optimizasyonu yönlendirdiği görülmüştür. Bu durum her aşamada alınan örnek sayısının az olmasının başta seçilecek örneklerin gürültülerden gelme ihtimali nedeniyle optimizasyonu yanlış yönlendirmesinin mümkün olduğunu ortaya çıkarmaktadır. Ancak gürültülü örneklerden gelmediği sürece optimizasyonun iyiye gitme ihtimali daha yüksektir. Aşama sayısının düşük olması her aşamada alınacak örnek sayısının fazla olmasını ve bu örnek gruplarının dağılımlarının genel dağılıma daha yakın olmasını gerektirir. Bu nedenle az aşamalı eğitimler klasik yönteme daha çok benzeyeceğinden optimizasyonun kötüye gitme ihtimali daha düşüktür. Bölüm 3.3'te büyüyen kümelerle 3 aşamalı Planlı ve Ters Planlı Öğrenme yöntemleri hiçbir veri kümesinde kaybetmiyorken Bölüm 4.4'te bu yöntemlerden Ters Planlı öğrenmenin 25 aşamalı versiyonunun 6 veri kümesinde kaybettiği görülmüştür. Elde edilen deneysel sonuçlar da teorik bilgileri destekler niteliktedir.

Elde edilen sonuçlara göre Planlı Öğrenme ve varyasyonlarının başarısının örneklerin zorluk seviyesine göre anlamlı bir sıralamada verilmesinden bağımsız olarak her aşamada eğitim kümesine yeni örnekler eklenerek büyütülmesi neticesinde sağlandığı görülmüştür. Rastgele Düzenli Büyüyen Kümeler yönteminin bundan sonraki Planlı Öğrenme çalışmalarında kıyaslama yöntemi olarak kullanılması önerilmektedir ve eğer büyüyen kümelerden kaynaklanan iyileştirmenin üzerinde bir iyileştirme görülürse bunun anlamlı ve doğru bir sıralamadan kaynaklandığı söylenebilir.

KAYNAKLAR

[1] Goodfellow, I., Bengio Y., ve Courville A., (2016). Deep Learning, MIT Press, http://www.deeplearningbook.org, 9 Haziran 2017.

[2] Güncel Türkçe Sözlük, Türk Dil Kurumu, http://www.tdk.gov.tr/, 14 Ocak 2018.

[3] Cauchy, A., (1847). "Méthode générale pour la résolution de systèmes d'équations simul-tanées", In Compte rendu des séances de l'académie des sciences, 536–538.

[4] Polyak, B. T., (1964). "Some methods of speeding up the convergence of iteration methods", USSR Computational Mathematics and Mathematical Physics, 4(5):1–17.

[5] Sutskever, I., Martens, J., Dahl, G., ve Hinton, G., (2013). "On the importance ofinitialization and momentum in deep learning", International Conference on Machine Learning, 1139-1147.

[6] Duchi, J., Hazan, E., ve Singer, Y., (2011). "Adaptive subgradient methods for online learning and stochastic optimization", Journal of Machine Learning Research, *12*(Jul):2121-2159.

[7] Hinton, G., (2012). "Neural networks for machine learning", Coursera video dersleri.

[8] Kingma, D. ve Ba, J., (2015). "Adam: A method for stochastic optimization", International Conference on Learning Representations.

[9] Ioffe, S. ve Szegedy, C., (2015). "Batch normalization: Accelerating deep network training by reducing internal covariate shift", International Conference on Machine Learning, 448-456.

[10] Polyak, B. ve Juditsky, A., (1992). "Acceleration of stochastic approximation by averaging", SIAM Journal on Control and Optimization, 30(4):838–855.

[11] Bengio, Y., Louradour, J., Collobert, R. ve Weston, J., (2009). "Curriculum Learning", International Conference on Machine Learning, 41-48.

[12] Elman, J. L., (1993). "Learning and development in neural networks: The importance of starting small", Cognition, 48:781-799.

[13] Krueger, K. A. ve Dayan, P., (2009). “Flexible shaping: How learning in small steps helps”, Cognition, 110:380-394.

[14] Sanger, T. D., (1994). “Neural network learning control of robot manipulators using gradually increasing task difficulty”, IEEE Transactions on Robotics and Automation, 323-333.

[15] Kumar, M. P., Packer, B. ve Koller, D., (2010). “Self-paced learning for latent variable models”, In Advances in Neural Information Processing Systems, 1189-1197.

[16] Jiang L., Meng D., Yu S., Lan Z., Shan S., ve Hauptmann A., (2014). “Self-paced learning with diversity”, In Advances in Neural Information Processing Systems, 2078–2086

[17] Zaremba, W., ve Sutskever, I., (2014). “Learning to execute”, arXiv ön baskısı arXiv:1410.4615.

[18] Shi, Y., Larson, M., ve Jonker, C. M., (2015). “Recurrent neural network language model adaptation with curriculum learning”, Computer Speech & Language, 33(1):136-154.

[19] Pentina, A., Sharmanska, V., ve Lampert, C. H., (2015). “Curriculum learning of multiple tasks”, In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 5492-5500.

[20] Mitchell, T., (1997). Machine Learning, McGraw-Hill Science.

[21] Opitz, D. ve Maclin, R., (1999). “Popular ensemble methods: An empirical study”, Journal of Artificial Intelligence Research, 11:169–198.

[22] Breiman, L., (1996). “Bagging predictors”, Machine Learning, 24(2):123-140.

[23] UCI Makine Öğrenmesi Veritabanı, http://archive.ics.uci.edu/ml/datasets 29 Eylül 2017.

[24] Gong, T., Zhao, Q., Meng, D., ve Xu, Z., (2016). “Why Curriculum Learning & Self-Paced Learning Work In Big/Noisy Data: A Theoretical Perspective”, American Institute of Mathematical Sciences Big Data and Information Analytics, 1(1):111-127.

[25] Meng D., Zhao Q. ve Jiang L., (2015). “What objective does self-paced learning indeed optimize?”, arXiv ön baskısı arXiv:1511.06049.

[26] Chang, H. S., Learned-Miller, E., ve McCallum, A., (2017). “Active Bias: Training a More Accurate Neural Network by Emphasizing High Variance Samples”, arXiv ön baskısı arXiv:1704.07433.

[27] Avramova, V., (2015). “Curriculum Learning with Deep Convolutional Neural Networks”, Yüksek Lisans Tezi, KTH School of Computer Science and Communication.

[28] Dauphin, Y. N., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., ve Bengio, Y., (2014). “Identifying and attacking the saddle point problem in high-dimensional

non-convex optimization", In Advances in Neural Information Processing Systems, 2933-2941.

[29] Erhan, D., Bengio, Y., Courville, A., Manzagol, P. A., Vincent, P., ve Bengio, S., (2010). "Why does unsupervised pre-training help deep learning?", Journal of Machine Learning Research, 11(Feb):625-660.

[30] Lewis, D. D., ve Gale, W. A., (1994). "A sequential algorithm for training text classifiers", International ACM SIGIR Conference on Research and Development in Information Retrieval, 3-12.

EK-A

TERİMLER LİSTESİ

Kullanılan terim	Literatürdeki karşılığı
Hedef fonksiyonu	Objective function
Hata fonksiyonu	Error function
Kayıp fonksiyonu	Loss function
Maliyet fonksiyonu	Cost function
Dışbükey optimizasyon	Convex optimization
Dışbükey olmayan optimizasyon	Non-convex optimization
Yerel minimum	Local minimum
Yerel maksimum	Local maximum
Eyer noktası	Saddle point
Planlı öğrenme	Curriculum learning
Ters planlı öğrenme	Anti curriculum learning
Kendini planlayan öğrenme	Self paced learning
Kendini tersten planlayan öğrenme	Self paced learning inversed
Destek vektör makineleri	Support vector machines
Gradyan düşümü	Gradient descent
Etiketli öğrenme	Supervised learning
Eşlenik gradyanlar	Conjugate gradients
Toplu normalleştirme	Batch normalization
Yeniden parametrelendirme	Reparametrization
Sürdürme yöntemleri	Continuation methods
Özyinelemeli sinir ağı	Recurrent neural network
Uzun Kısa-Süreli Bellek	Long Short-Term Memory
Gizli yapılı SVM	Latent structural SVM
Derin evrişimsel sinir ağı	Deep convolutional neural network
Çok görevli öğrenme	Multitask learning
Kolektif öğrenme	Ensemble learning
Topluluk	Ensemble
Torbalama	Bagging
Artımsal eğitim	Incremental training

Toplu eğitim	Batch training
Devir	Epoch
Çarpık	Skewed
Eşleştirilmiş T-testi	Paired T-test
Çapraz doğrulama	Cross validation

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı : Melike Nur MERMER

Doğum Tarihi ve Yeri : 26.07.1993 / Fatih

Yabancı Dili : İngilizce

E-posta : melikenurmermer@gmail.com

ÖĞRENİM DURUMU

Derece	Alan	Okul/Üniversite	Mezuniyet Yılı
Lisans	Bilgisayar Mühendisliği	Sakarya Üniversitesi	2016
Lisans	Mekatronik Mühendisliği	Sakarya Üniversitesi	2015
Lise	Sayısal	Ergün Öner – Mehmet Öner Anadolu Lisesi	2011

İŞ TECRÜBESİ

Yıl	Firma/Kurum	Görevi
2016-devam	İstanbul Sabahattin Zaim Üniversitesi	Araştırma Görevlisi

YAYINLARI

Makale

1. Mermer M. N. ve Amasyali M. F., (2017). Scalable Curriculum Learning for Artificial Neural Networks, IPSI BgD Transactions on Internet Research, 13(2):57-62.