FA01V01 ALGORİTMASI: YENİ BİR DİNAMİK YAPAY SİNİR AĞI ALGORİTMASI

Erkam GÜREŞEN[1]
Gülgün KAYAKUTLU[2]

ÖZ

Yapay sinir ağları yapıları oluşturulurken, kara kutu yaklaşımının benimsenmesinin etkisiyle, her zaman çıktı sinir hücresi sayısı sabitlenmektedir. Mevcut bir yapay sinir ağına hiçbir zaman yeni bir çıktı sinir hücresi eklenmemektedir. Bu durum yapay sinir ağlarının hiçbir zaman yeni bir yetenek veya beceri kazanamamasına sebep olmaktadır.Bu çalışmanın amacı mevcut bir yapay sinir ağına yeni beceri ve yetenek kazandırmak amacıyla yeni çıktı sinir hücreleri ekleyebilen bir algoritma geliştirmektir.

Böyle bir algoritmanın geliştirilebilmesi için öncelikle mevcut bir beceriyi arttırma ve düzeltme ile yeni bir yetenek kazandırma kavramlarının birbirinden ayrıştırılması gerekmektedir. Daha sonra ise yeni çıktı sinir hücresi oluşturma ve en uygun olanı tespit etme yöntemlerinin geliştirilmesi gerekmektedir. Bu çalışmada yeni eklenecek çıktı sinir hücresi adaylarını tespit etmek için çeşitli yöntemlerle bir aday havuzu oluşturuldu. Oluşan bu aday havuzundan K'nın L'li çapraz geçerliliğinde en uygun adaylar tespit edildi. Geliştirilen algoritma ve yöntemleri uygulamak için JAVA programlama dili kullanılarak NeuroBee adlı bir yazılım geliştirildi ve tüm uygulamalar bu yazılım kullanılarak yapıldı.

Uygulama için ise ve, veya ve özel veya mantık problemlerinin birleşiminden oluşan bir problem oluşturuldu. Bu problemde farklı sayıda gizli sinir hücreleri için sıfır çıktı sinir hücresine sahip çok katmanlı algılayıcılar ile çözüme başlandı. Daha sonra üç çıktı sinir hücreli yapı oluşuncaya kadar FA01V01 algoritması çalıştırıldı. Elde edilen yapay sinir ağları geri yayılım (backpropagation) algoritması ile eğitildi. Aday havuzundan yeni eklenecek çıktı sinir hücrelerinin seçimi için K'nın L'li çapraz geçerliliği (L of Kcross-validation) yöntemi gelişitirildi. Önerilen algoritmanın performansını karşılaştırmak için ise NeuroSolutions programının sunduğu çok katmanlı algılayıcılar (ÇKA) kullanılarak, farklı sayıda gizli sinir hücreleri için üç çıktı sinir hücreli olarak eğitildi.

Yapılan istatistiksel testler sonucunda ortalama hata karesi açısından FA01V01 algortiması ile oluşturulan ÇKA'lar ile sabit yapılı ÇKA'lar arasında fark olmadığı, ancak FA01V01 ile oluşturulan ÇKA'lar için geri yayılım algoritmasının daha az sayıda iterasyona ihtiyaç duyduğu tespit edilmiştir.

Anahtar kelimeler: *FA01V01 algoritması, yapay sinir ağı, mantık problemleri*

FA01V01 ALGORITHM: A NEW DYNAMIC NEURAL NETWORK ALGORITHM

ABSTRACT

[1] Y.Müh. Kara Harp Okulu Dekanlığı_End.ve Sist.Müh.Böl._Ankara_eguresen@kho.edu.tr
[2] Doç.Dr. İTÜ_End.ve Sist.Müh.Böl._İstanbul_gkayakutlu@itu.edu.tr

Due to black-box structure, number of outputs isalways fixed while constructing artificial neural networks (ANNs). Thusnone of the existing algorithms adds new outputs to an existing ANN. This lack of ability disables ANNs to learn new abilities or gain new properties.

Purpose of this study is to represent a new ANN constructing algorithm forenabling ANNs to learn new abilities or gain new properties.To represent such an algorithm, definitions of increasing or correcting an existing ability, and gaining a new ability concepts must be differentiated. Then new method for creating new output neurons and selecting best fitted one, must be developed. This study proposes different methods to create a candidate pool for selecting best suitable one. A new concept of L of K cross-validation is used to select best suitable output neuron from this candidate pool. For all these concepts described above, a new JAVA based software is developed and named NeuroBee. All applications of these concepts are done by using this software.

For comparisons a joint logic problem from well-known logic problems; AND, OR and XOR.In this problem construction ANNs are started with zero output neurons and then using FA01V01 constructed up to three output neurons. After adding each new output neuron back-propagation algorithm is used for training ANNs. Constructed ANNs are compared to ANNs produced from NeuroSolutions software with fixed number of output neurons during training.

Statistical tests showed that ANNs produced from FA01V01 or NeuroSolutions has no difference in terms of mean square error (MSE). But FA01V01 has a clear advantage over NeuroSolutions with respect to number of backpropagation iterations.

Keywords: *FA01V01 algorithm, artificial neural network, logic problems*

1. GİRİŞ

Yapay sinir ağları (YSA) yapıları oluşturulurken, kara kutu yaklaşımı benimsenir. Bu nedenle araştırmacılar ya sadece girdi sinir hücrelerinin sayısının ne olması gerektiğiyle ilgilenir ya da girdi ve çıktı sinir hücreleri sabitleyip, gizli katmandaki gizli sinir hücrelerinin sayısını tespit etmeye çalışırlar.

Mevcut yazın incelendiğinde yapay sinir ağlarına yeni yetenekler kazandıracak şekilde yeni bir çıktı sinir hücresi ekleyen bir algoritmanın olmadığı görülmektedir. Bu sebeble yapay sinir ağları, insanoğlunun belki de yapay zekâdan en başından beri beklediği, kendi kendine yeni sonuçlar üretebilme kabiliyetinden çok uzak bir konumdadır. Bu durumu daha iyi ifade edebilmek için bazı kavramların birbirinden net bir şekilde ayrılmasını sağlayacak tanımlar oluşturmak gerekmektedir.

Sadece yapısı oluşturupdaha önce hiç bir eğitim işlemi yapılmamış bir yapay sinir ağına istenilen çıktıları verdirmek amacıyla yapılan işlem *belletmek* ya da *kavratmak* (*knowing*) olarak adlandırılabilir. Değişen şartlar nedeniyle oluşan yeni gözlemleri kullanarak yapay sinir ağının mevcut bir çıktı sinir hücresinin ürettiği sonuçlardaki hatayı düzeltmek için yapılan yeniden eğitim ise *düzeltme* (*adjusting*) olarak tanımlanabilir. Bir yapay sinir ağının mevcut yeteneklerine ek olarak yeni bir yetenek kazandıracak şekilde yeni bir çıktı sinir hücresi eklenerek eğitimi ise *öğrenme* (*learning*) olarak adlandırılabilir.

a. Motivasyon (Motivation)

Mevcut yazın incelendiğinde kavrama, düzeltme ve öğrenme için tek bir kavram; "öğrenme" kullanıldığı görülmektedir. Bu da mevcut durumdaki eksikliğin tanımlanmasını zorlaştırmaktadır. Yazında bulunan algoritmalardan yapay sinir ağlarının ilk defa eğitenler kavramaya karşılık gelmektedirler. Değişen çevre şartlarına göre bir yapay sinir ağının tekrar kavratma yapılarak hata seviyesinin azaltılması ise düzeltmeye karşılık gelmektedir. Bütün bunlara karşılık bir yapay sinir ağına yeni bir yetenek kazandıracak şekilde yeni bir çıktı sinir hücresi ekleyen, kısaca öğrenme yapan algoritmalar ise bulunmamaktadır.

Yazındaki bütün YSA inşa algoritmaları YSA'nın çıktı katmanını ve burada bulanan çıktı sinir hücre sayısını sabit tutup, kavrama işlemini uygulamaktadırlar. Mevcut algoritmalardan bazıları çıktı sinir hücrelerini yapı olarak da sabit tutmaktadır. Bazıalgoritmalar ise çıktı sinir hücresini gizli sinir hücresine çevirerek yeni bir çıktı sinir hücresi eklemektedirler. İşlevi eski çıktı sinir hücresinin hata miktarını düzeltmek olan bu yaklaşım da YSA yapısına yeni bir işlev katmadığından öğrenme olarak adlandırılamaz.

b. Amaç (The Goal)

Bu çalışma yapay sinir ağlarına yeni işlev kazandırmak için yeni sinir hücresi eklemek üzere yoğunlaşmıştır. Daha önce çalışılmamış olan çıktı

hücrelerini ekleyen bir algoritma geliştirilmiştir. Bu yeni algoritmanın performansını sabit sayılı çıktı sinir hücresine sahip yapay sinir ağları ile karşılaştırarak performansının ölçülmesi amaçlanmıştır.

c. Katkı (Contribution to Literature)

Bu çalışmada boş bir çıktı katmanına sahip bir YSA ile başlayan ve istenilen miktarda yeni çıktı sinir hücresi ekleyerek YSA inşa eden yeni bir algoritma geliştirilmiştir. Bu algoritma bir yapay sinir ağının çıktı katmanını sıfırdan oluşturabileceği gibi daha önce eğitilmiş bir yapay sinir ağına yeni işlevler eklemek için çıktı sinir hücresi ekleyebilmektedir.

Her yeni eklenecek çıktı sinir hücresi için bir aday havuzu oluşturulmaktadır. Oluşturulan bu havuzda yakın işleve sahip hücrelerin bilgi birikimini kullanabilmek için rassal ağırlıklı adayların yanında mevcut hücrelerinin kopyalanması, toplanması, bileşimi ve bölünmesi ile oluşturulmuş çıktı sinir hücreleri de kullanılmaktadır.

Aday havuzundaki çıktı sinir hücreleri arasından en uygun adayı tespit etmek ve öğrenmeye olan katkısını geçerlemek için yeni bir çapraz geçerlilik yöntemi geliştirilmiştir. Bu yönteme *K*'nın*L*'li çapraz geçerliliği (*L* of *K* cross validation) adı verilmiştir.

ç.Çalışmanın organizasyonu (Organization of Study)

Amacı ve mevcut yazın taraması verilen çalışmanın geri kalan kısmı şu şekildedir: ikinci bölümde FA01V01 algoritması açıklanmış ve algoritmanın sözde programı verilmiştir. Aynı bölümde aday havuzu oluşturmada kullanılan yöntemler açıklanmış ve modellerin değerlendirme şekillerine değinilmiştir. İkinci Bölümün son kısmında ise bu çalışmada kullanılan "ve", "veya" ve "özel veya" bütünleşik mantık problemi açıklanmıştır.

Çalışmanın üçüncü bölümünde ise önerilen algoritma bahsedilen bütünleşik mantık problemine uygulanmıştır. Algoritmanın uygulanma platformu ve geliştirilen NeuroBee yazılımı açıklanmıştır. Ayrıca algoritmanın kıyaslamasını yapmak üzere aynı problem NeuroSolutions v.5.05 yazılımının çok katmanlı algılayıcı modellerine uygulanmıştır. Üçüncü bölümün sonunda elde edilen sonuçlar paylaşılmıştır. Son bölümde ise elde edilen bulgular değerlendirilmiş ve ileride yapılacak çalışmalar hakkında öneriler sunulmuştur.

2. LİTERATÜR TARAMASI (LITERATURE REVIEW)

Yapay sinir ağının en uygun yapısını bulmak için dört temel yaklaşım vardır. Bu yaklaşımlar yapıcı (constructive) algoritmalar, budama (pruning) algoritmaları, düzenleme (regularization) algoritmaları ve evrimsel (evolutionary) algoritmalardır (Kwok ve Yeung, 1997a).

Yapıcı algoritmalar göreli olarak daha küçük bir yapay sinir ağı ile başlarlar. Daha sonra gizli sinir hücreleri ve gizli katmanlar eklemek suretiyle yapay sinir ağının yapısını büyütürler. Bu algoritmalarda temel amaç mümkün olduğunca küçük bir ağ yapısı ile istenilen seviyede hata oranını yakalamaya çalışmaktır.

Budama algoritmaları ise göreli büyük bir yapay sinir ağı yapısı ile başlarlar ve istenilen hata seviyesini sağlayacak şekilde ağ yapısından bazı gizli sinir hücrelerini çıkartırlar. Mozer ve Smolensky (1989) bunu yapmak için daha az önemli gizli sinir hücrelerini (GSH'lerini) tespit edip bunları yok ederek yapay sinir ağı yapısını uygun boyuta getirmektedir. Karnin (1990) ise hata fonksiyonun mevcut GSH'lerin silinmesine duyarlılığını hesaplayarak budama işlemini gerçekleştirmektedir. Castellano ve diğ. (1993) ise 3 farklı budama algoritmasını; Sietsma ve Dow (1988), Burkitt (1991) ve Pelillo ve Fanelli (1993)'nin algoritmalarını karşılaştırmışlardır. Castellano ve diğ. (1997) ise problemden bağımsız tekrarlayan bir budama algoritması geliştirmişlerdir.

Düzenleme algoritmaları (Regularization) göreli büyük bir ağ yapısı ile başlar ve bu yapıdaki sonuca etkisi az olan ama hala hatanın düşürülmesine yardımcı olan bağlantıları düzenlemeye çalışırlar (Ma ve Khorasani, 2003). Bu bağlantılar eğitim verisinde hatayı düşürse de yeni bir veri ile karşılaşıldığında genelleme yeteneklerini kaybettiklerinden hatanın artmasına sebep olurlar (Ma ve Khorasani, 2003). Bu sorunu aşmak için düzenleme algoritmaları düzenleme kuralları oluşturarak bu kurallara göre daha az önemli olan ağırlıkların sıfıra çekilmesini yani silinmesini sağlarlar. Düzenleme kuralları Thodberg (1996) ile Buntine ve Weigend (1991)'de olduğu gibi Bayes kuramına dayalı yöntemlerle oluşturulabilir. Ancak Ma ve Khorasani (2003) bu kurallarda Bayesyen kanıtların olmadığını ve bazen yeterince iyi sonuçlar vermediğini belirlemişlerdir.

Evrimsel algoritmalar literatürde yapay sinir ağlarının parametre tespitinde kullanılmaktadır Çınar (2007). Kullanılan (veya kullanılması umulan) bu evrimsel algoritmaların çeşitleri Kiranyaz ve diğ. (2009)'da şöyle sıralamıştır; genetik algoritmalar (Goldberg, 1989), genetik programlama (Koza, 1992), evrimsel stratejiler (Back ve Kursawe, 1995), evrimsel programlama (Fayyad ve diğ., 1996), parcacıkk sürü algoritmaları (partical swarm optimization) (Kennedy ve Eberhart, 1995) olarak söylenebilir. Evrimsel algoritmalar popülasyon tabanlı stokastik süreçlerdir ve yerel optimumlara takılı kalmayı önlemek amacıyla kullanılmaktadır (Kiranyaz ve diğ., 2009).

Yapay sinir ağı parametrelerinden öğrenme oranını evrimsel algoritma kullanarak belirleyenlere Kim ve Bae (2005); Wang ve Huang (2007); Yi-Hui (2007)'nun çalışmaları örnek olarak verilebilir. Evrimsel algoritmalar ile başlangıç ağırlıklarını tespit edenlere ise Castillo ve diğ. (2000), Kim ve Bae (2005), Wang ve Huang (2007), Kuo (2001)'nın çalışmaları, gizli katman sayısını ve bu katmanlardaki işlem elemanı sayısını tespit edenlere ise Castillo ve diğ. (2000), Niska ve diğ. (2004), Ferentinos (2005), Kim ve Bae (2005), Wang ve Huang (2007), Yi-Hui (2007)'nın çalışmaları, işlem elemanlarının transfer fonksiyonunun türünü tespit edenlere ise Ferentinos (2005)'un

çalışması örnek gösterilebilir. Evrimsel algoritmalarla ayrıca hangi girdilerin kullanılacağı araştırılmış yani başlangıç elemanları tespit edilmiştir. Bu tip çalışmalara ise Niska ve diğ. (2004), Sexton ve diğ. (2004), Doganis ve diğ. (2006)'nin çalışmaları örnek verilebilir.

Dört temel yaklaşım incelendiğinde, yapıcı algoritmaların göreli küçük ağlarla başlaması, hesaplama süresi açısından yapıcı algoritmalara diğer yaklaşımlardan daha hızlı çalışma fırsatı sunmaktadır. Ayrıca budama ve düzenleme algoritmaları mevcut yapay sinir ağı yapısına herhangi bir ekleme yapmadığından öğrenme değil sadece kavrama yapaktadırlar. Benzer şekilde evrimsel algoritmalar da yazında sadece yapay sinir ağlarının parametre optimizasyonunda kullanıldığından kavrama yapmakta ancak öğrenme yapmamaktadırlar. Kavramanın yanında öğrenme yapabilecek algoritmalar olarak yapıcı algoritmalar işleyiş olarak ayrıntılı bir incelemeyi gerektirecek potansiyele sahiptirler.

a. Yapıcı Yapay Sinir Ağı Algoritmaları

Dinamik Düğüm Yaratma (Dynamic Node Creation) algoritmaları ilk olarak Ash (1989) tarafından ortaya atılmıştır. Temelde bu algoritmalar gizli katmana her seferinde yeni bir gizli sinir hücresi (GSH) eklerler ve tüm YSA tekrar eğitilir (Kwok ve Yeung, 1997a). Azimi-Sadjadi ve diğ. (1993) hem ağırlıkları güncelleyen hem de gizli katmandaki GSH sayısını tek tek arttıran bir algoritma oluşturmuştur. Zhang (1994) ise dinamik düğüm yaratma benzeri SELF adındaki algoritmayı geliştirmiştir ve bu algoritma ile gizli katmandaki en iyi GSH sayısını bulmaya çalışmaktadır.

İzdüşüm Takip Regresyonu (Projection Pursuit regression) algoritmaları temel olarak Friedman ve Stuetzle (1981)'nin bu konudaki çalışmalarını ele alırlar. Temel olarak bu algoritmalarda çok değişkenli veriyi iki katmanlı bir ağ yardımıyla çok boyutlu bir uzaydan izdüşümlerle daha az boyutlu bir uzaya taşırlar (Hwang ve diğ., 1994). Saha ve diğ. (1993) ise radyal temelli

fonksiyonlar kullanılan problemlerin gözlem/boyut oranını düşürmek için çoklu lineer izdüşüm kullanmışlardır. Shin ve Ghosh (1995) Ridge Polinom Ağlarını, izdüşüm takip regresyonunun özel bir durumu; her zaman aynı fonksiyonu kullanan versiyonu olarak özetlemiştir. Verkooijen ve Daniels (1994) ise izdüşüm takip regresyonunda kullanılacak fonksiyonları ayrı ayrı yapay sinir ağı olarak ele almış ve bu ağları eğiterek uygun fonksiyonu bulmuştur.

Basamak-Korelasyon algoritması (Cascade-Corelation Algorithm) ilk defa Fahlman ve Lebiere (1990) tarafından ilk olarak geliştirilmiştir. Bu algoritmada her seferinde yeni bir GSH eklenmektedir. Bu yeni eklenen GSH diğer tüm gizli katmandaki GSH'lere bağlanmaktadır. Böylece aslında her eklenen GSH, bir GSH'den oluşan bir gizli katman oluşturmaktadır (Kwok ve Yeung, 1997a). Phatak ve Koren (1994), basamak-korelasyon algoritmasını her bir gizli katman birden fazla GSH alabilecek şekilde değiştirmişler böylece tüm girdi sinir hücrelerinin GSH'lere bağlantısı olmamasını sağlamışlar. Sjogaard (1992) ikinci dereceden basamak-korelasyon algoritması kullanarak eklenen tüm GSH'lerin aynı gizli katmanda ve birbirlerinden etkilenmeyecek (birinin çıktısı diğerinin girdisi olmayacak) şekilde toplamıştır. Kwok ve Yeung (1997b) farklı korelasyon tabanlı farklı fonksiyonları kullanarak farklı zaman ve uzay ihtiyaçlarını karşılaştırarak performans analizleri yapmışlardır.

Kaynak dağıtım ağları (Resource Allocation Networks; RAN) her seferinde bir adet GSH'yi gizli katmana eklemektedirler ve diğer yapıcı algoritmalardan temel farkı öğrenme izlerini hatırlayarak bunları kullanmalarıdır (Kwok ve Yeung 1997a). Sadece gizli katmandaki GSH'leri artırarak ya da azaltarak adaptasyon yeteneğini artırırlar.

İlk defa Platt (1991) tarafından ortaya atılan kaynak dağıtım ağı iki katmanlı bir radyal tabanlı fonksiyon (RTF) ağıdır (Li ve diğ., 2010). Fritzke (1993, 1994a, 1994b) yaptığı çalışmalarda hem gözetimli hem de gözetimsiz eğitim için büyüyen hücre yapıları (growing cell structures) adlı algoritmayı geliştirmiştir. Daha sonra Fritzke (1995a, 1995b) bu algoritmayı büyüyen sinir

gaz ağları (growing neural gas network) ve büyüyen ızgara (growing grid) adı altında geliştirmiştir.

Grup metoduyla veri işleme yöntemleri (Group Method of Data Handling) genellikle Ivakhnenko (1984)'den esinlenilerek geliştirilmiş metotlardır. Diğer algoritmalardan farkı her katmanda sabit miktarda bağlantı alması ama bu bağlantıların girdi sinir hücreleri ve mevcut GSH'lerin çıktılarının herhangi bir kombinasyonundan oluşmasıdır (Kwok ve Yeung, 1997a). Dolayısıyla her bir GSH ekleme işlemi birçok ağ alternatifini gündeme getirir. Bu alternatif ağ seçimi sırasında çok farklı yöntemler kullanılabilir (Tenorio ve Lee, 1990; Parker ve Tummala, 1992).

Çıktı sinir hücresi değiştiren algoritmalarda ise mevcut çıktı sinir hücrelerini (ÇSH) GSH'ye dönüştürülerek yeni bir ÇSH eklenir ve bu yeni ÇSH'nin daha az hata vermesi istenir. Bu kategoriye Ma ve Khorasani (2003)'nin yapıcı stratejilerini, Ghiassi ve Saidane (2005)'nin DAN2'sini, Mezard ve Nadal (1989)'nin kiremit algoritmasını, Parekh ve diğ. (2000)'nin MTiling-Real adı altında reel değerli, çok çıktılı bir kiremit algoritmasını, Gallant (1990)'ın kule algoritmasını ve Parekh ve diğ. (2000)'nin MPyramid-Real adlı algoritmalarını koyabiliriz.

Upstart algoritması ilk olarak Frean (1990) tarafından oluşturulmuştur. Algoritmanın lineer olmayan verilerde de sağlıklı işleyip ağdaki ağırlıkları sürekli hatayı azaltacak şekilde güncelleyebilmek için Gallant (1986) tarafından önerilen cep algoritmasını (pocket algorithm) kullanmıştır. Upstart algoritması da sadece bir adet ÇSH kullanmaktadır. ÇSH değiştiren algoritmalardan temel farkı hatayı düzeltmek için ÇSH'yi iki yeni GSH ekleyerek düzeltmeye çalışmasıdır.

CARVE algoritması ilk defa Young ve Downs (1998) tarafından oluşturulmuştur. Bu algoritma ilk önce sadece başlangıç sinir hücreleri ve

eğitim setindeki her bir sınıf için bir ÇSH olacak şekilde bir YSA ile başlar. Bu durumda gizli katmanlarda hiç GSH yoktur.

Frattale-Mascioli ve Martinelli (1995) tarafından ortaya atılan petrol bulma algoritması (oil-spot algorithm) N-boyutlu uzayı 3 boyutlu birim küpe eşlemeye dayanır. Bu algoritma CARVE algoritmasının çıkış noktası niteliğindedir.

STRIP tabanlı yapay sinir ağı algoritması Ngom ve diğ. (2001) tarafından ortaya atılmıştır. Bu algoritma *strip* adı verilen iki hiper düzlem arasında kalan aynı sınıfa ait nokta kümelerinden en büyüğünü bulmaya dayanır.

Setiono (2001)'nın önerdiği N2C2S (Neural Network Construction with Cross-Validation Samples) algoritmasında 6 adım vardır.

Sınırlandırılmış Coulomb enerji ağı Reilly ve diğ. (1982) tarafından ortaya atılmıştır. Bu metot aslında bugün radyal tabanlı fonksiyon ağları dediğimiz yapıdır (Li ve diğ., 2010). *Radyal Tabanlı Fonksiyon Ağları* (*RTF*) ise gizli katmanında *h* adet taban fonksiyonu bulunduran çok katmanlı algılayıcılardır (Broomhead ve Lowe, 1988; Moody ve Darken, 1989).

Puma-Villanueva ve diğ. (2012)'nin geliştirdiği rassal bağlantılandırılmış ileri beslemeli sinir ağlarını birleştirme algoritması sadece girdi ve çıktı katmanlarından oluşan ve gizli katmanı boş bir ağ yapısıyla başlar. Bu ağda girdi ve çıktı katmanları arasında bağlantı oluşturulmaz. Sonra ilk olarak ortak bilgi (mutual information) kullanılarak her başlangıç elemanı (dolayısıyla her bir özellik) için bir önem hesaplanır ve buna bağlı olarak bu başlangıç elemanları çıktı katmanındaki bitiş elemanlarına bağlanmaya başlarlar. Daha sonra tanımlanan kurallar çerçevesinde yeni bağlantılar ve yeni gizli sinir hücreleri (GSH'ler), eklenmeye veya tanımlanan şartlar sağlandıkça bu bağlantılar veya GSH'ler silinmeye devam edilir. Bu işlemler algoritmanın durdurma kriterlerine ulaşılıncaya kadar devam eder.

Yang ve Chen (2012)'in geliştirdiği Evrimsel yapıcı ve budayıcı algoritma (ECPA) girdi katmanı, çıktı katmanı ve içerisinde bir adet gizli sinir hücresinin bulunduğu gizli katmana sahip ağlardan oluşan bir popülasyon ile başlamaktadır. Geri yayılım (back propagation) algoritması ile eğitilen bu ağlar gizli katmana hem ekleme yapmakta (yapıcı) hemde gereksiz görülen gizli sinir hücrelerini çıkararak budama yapmaktadır.

Han ve Qiao (2013) tarafından geliştirilen ileri beslemeli sinir ağlarında bir yapı optimizasyon algoritması (A structure optimization algorithm for feed forward neural network construction) hem yapıcı hem budayıcı özelliği olan bir algoritmadır. Bu algoritma sadece bir gizli katmanı olan ileri beslemeli yapay sinir ağlarında çalışmaktadır.

Wang (2008) tarafından geliştirilen Hızlı Yapıcı-Kaplayıcı Algoritma (Fast Constructive-Covering Algorithm for neural networks) girdi ve çıktı katmanlarını sabitleyip araya boş bir gizli katman koyarak başlamaktadır. Daha sonra gerektikçe ara katmana işlem elemanı koyarak yapay sinir ağını oluşturmaya çalışmaktadır.

Hata düzeltme ile rekabetçi çoğunluk ağı eğitim algoritması (C-mantec) Subirats ve diğ. (2012) tarafından geliştirilmiştir. Çoğunluk (majority) fonksiyonlarını kullanan bu algoritma tek gizli katmanlı ve tek çıktılı bir yapay sinir ağı oluşturur. Bu algoritmada kullanılan çoğunluk fonksiyonu N adet girdiyi tek bir bit'e dönüştüren mantıksal bir fonksiyondur.

3. FA01V01 ALGORİTMASI (FA01V01 ALGORITHM)

İnsanoğlunun bir gün kendi yarattığı zeki makineler tarafından esir edilip edilmeyeceği akılları kurcalayan, belki de yapay zeka kavramı hakkında ortaya atılan en önemli iddiadır. Ancak makine öğrenmesi metotlarından yapay sinir ağlarında, metotlar başlangıçta oluşturulan modeli daha zeki yapmaya, yani yeni işlev ve özellikler ekleyerek öğrenmesini sağlamaya çalışmamaktadır. Yapay sinir ağları açısından bakıldığında önceki bölümde tanımlandığı gibi

öğrenmenin olabilmesi için yapay sinir ağlarının yeni çıktı sinir hücreleri ekleyebilmeleri gerekmektedir.

Bu çalışmada yapay sinir ağlarına sonradan çıktı sinir hücreleri ekleyebilecek, veya yapay sinir ağı yapısı oluşturulurken çıktı sinir hücresi sayısını en baştan sabit tutmak yerine, eğitim aşamaları sırasında yavaş yavaş ekleyen bir algoritma sunulmuştur. Bu algoritma FA01V01 olarak adlandırılmıştır.

FA01V01 algoritması mevcut eğitilmiş ve çalışır haldeki bir yapay sinir ağına yeni çıktı sinir hücresi ekleyebilecek özelliktedir. Bu çalışmada çok katmanlı algılayıcılar (Multi-Layer Perceptron) için geri yayılım algoritması (backpropagation) kullanılarak gözetimli eğitim için FA01V01 algoritması önerilmiştir.

a. Sözde Program (Pseudo Code) FA01V01 algoritmasının sözde programı aşağıda görüldüğü gibidir.

Adımlar	**İşlem** *(Tablo FA01V01 Algoritması Sözde Programı)*
Adım 1	Öğrenmenin gerçekleşeceği bir yapay sinir ağı al/oluştur
Adım 2	Girdilerden *K*'nın *L*'li çapraz geçerlilik veri kümelerini oluştur
Adım 3	Algoritmanın durma koşullarını tanımla
Adım 4	Eğitim ve çapraz geçerlilik veri kümelerini hazırla
Adım 5	Eklenecek çıktı sinir hücreleri için aday havuzu hazırla
Adım 6	Eğitim adaylarının uygunluğunu *K*'nın *L*'li çapraz geçerliği için hesapla
Adım 7	En uygun adayı aday havuzu içinden seç ve mevcut yapay sinir ağı yapısına ekle
Adım 8	Yeni oluşan yapay sinir ağını eğit
Adım 9	Durma koşullarının sağlanıp sağlanmadığını kontrol et
Adım 10	Durma koşulları sağlanmıyorsa Adım 4'e git
Adım 11	Oluşan yapay sinir ağını sonuç olarak ver

b. Girdilerden veri kümelerinin oluşturulması

K'nın *L*'li çapraz geçerliliğinde, ayrılan *L* adet kümenin her birinin tüm veri kümesini tam olarak temsil edebilecek yeterlilikte bir örneklem olduğu, dolayısıyla her *L* kümesinde ayrı ayrı iyileştirme sağlayan işlemin yapay sinir ağının genelleştirme özelliğini destekleyen bir ilerleme olacağı varsayımına dayanır.

K'nın *L*'li çapraz geçerliliğinin geliştirilmesindeki temel amaç aday eleman havuzundan hangi elemanın seçilip yapay sinir ağının yapısına katılacağına karar vermektir. Öncelikle eklendiği taktirde en fazla sayıda çapraz geçerlilik kümesinde V_i hata azalması sağlayan aday yapay sinir ağının genelleme özelliğini arttırıyordur. En fazla sayıda çapraz geçerlilik kümesinde hata azalmasını sağlayan birden fazla aday varsa ortalama hata karelerini (OHK) en fazla azaltan aday seçilir.

c. Aday Havuzu Oluşturma Yöntemleri (Candidate Pool)

Yeni eklenecek çıktı sinir hücresinin tespiti için adayların oluşturulması gerekmektedir. Bu adayların önceki edinilmiş bilgilerden de faydalanabileceği şekilde adaylar oluşturulmalıdır. Bu amaçla bu çalışmada aşağıda belirtilen yöntemler kullanılarak yeni çıktı sinir hücresi adayları oluşturulmuştur. Oluşturulan yeni çıktı sinir hücresinin transfer fonksiyonu çıktı katmanındaki diğer çıktı sinir hücreleri ile aynıdır.

(1) Rassal Ağırlıklı Yeni Çıktı Sinir Hücresi

Çıktı katmanına bağlantısı olan i'inci gizli sinir hücresini ile yeni oluşturulan ÇSH arasındaki bağlantının ağırlığı v_i şöyle tanımlanabilir:

$$f(v_i)=\begin{cases}\frac{1}{2} & -1<v_i<1\\ 0 & \text{diğer}\end{cases} \quad (1)$$

$$P\{-1<v_i<1\}=\int_{-1}^{1} f(v_i)\,dv_i \quad (2)$$

(2) Ters Ağırlıklı Yeni Çıktı Sinir Hücresi

Çıktı katmanına bağlantısı olan i'inci gizli sinir hücresini ile seçilen çıktı sinir hücresi arasındaki bağlantının ağırlığı v_i^o, yeni oluşturulan hücre ile bağlantısının ağırlığını v_i^n göstermek üzere yeni çıktı sinir hücresinin ağırlığı şöyle ifade edilerbilir:

$$v_i^n = -v_i^o \quad (3)$$

(3) Mevcut Bir Çıktı Sinir Hücresinin, Birleşimleri Kendisini Verecek Şekilde İki Çıktı Sinir Hücresine Bölünmesi

Mevcut durumda çıktı katmanına bağlantısı olan i'inci gizli sinir hücresini ile seçilen çıktı sinir hücresi arasındaki bağlantının ağırlığı v_i^o, yeni eklenen j'inci çıktı sinir hücresi arasındaki bağlantının ağırlığı ise v_{ij}^n olsun.Ağırlık ikililerinin olasılık kütle fonksiyonu g, $p\in(0,1)$ için şöyledir:

$$g(v_{i1}^n, v_{i2}^n)=\begin{cases}p & v_{i1}^n = v_i^o \wedge v_{i2}^n = c\\ 1-p & v_{i1}^n = c \wedge v_{i2}^n = v_i^o\end{cases} \quad (4)$$

$$c = 0+\varepsilon \quad (5)$$

burada c sabit bir sayı, ε ise yeterince küçük bir sayıdır.

(4) Mevcut Bir Çıktı Sinir Hücresinin, Toplamları Kendisini Verecek Şekilde İki Çıktı Sinir Hücresine Bölünmesi

Mevcut durumda çıktı katmanına bağlantısı olan i'inci gizli sinir hücresini ile seçilen çıktı sinir hücresi arasındaki bağlantının ağırlığı v_i^o, yeni eklenen j'inci çıktı sinir hücresi arasındaki bağlantının ağırlığı ise v_{ij}^n olsun. Ağırlık ikililerinin olasılık kütle fonksiyonu g, $p,k \in (0,1)$ için şöyledir:

$$g\left(v_{i1}^n, v_{i2}^n\right) = \begin{cases} p & v_{i1}^n = v_i^o \times k \wedge v_{i2}^n = v_i^o \times (1-k) \\ 1-p & v_{i1}^n = v_i^o \times (1-k) \wedge v_{i2}^n = v_i^o \times k \end{cases} \quad (6)$$

(5) İki Çıktı Sinir Hücresinin Kısmi Birleştirilmesi ile Yeni Bir Çıktı Sinir Hücresi Oluşturulması

Mevcut durumda çıktı katmanına bağlantısı olan i'inci gizli sinir hücresini ile seçilen j'inci çıktı sinir hücresi arasındaki bağlantının ağırlığı v_{ij}^o, yeni eklenen çıktı sinir hücresi arasındaki bağlantının ağırlığı ise v_i^n olsun. Ağırlık ikililerinin olasılık kütle fonksiyonu g, $p \in (0,1)$ için şöyledir:

$$g\left(v_i^n\right) = \begin{cases} p & v_i^n = v_{i1}^o \\ 1-p & v_i^n = v_{i2}^o \end{cases} \quad (7)$$

(6) İki Çıktı Sinir Hücresinin Toplamına Eşit Yeni Bir Çıktı Sinir Hücresi

Mevcut durumda çıktı katmanına bağlantısı olan i'inci gizli sinir hücresini ile seçilen j'inci çıktı sinir hücresi arasındaki bağlantının ağırlığı v_{ij}^o, yeni eklenen çıktı sinir hücresi arasındaki bağlantının ağırlığı ise v_i^n olsun. Yeni sinir hücresinin ağırlığını şöyle tanımlayabiliriz:

$$v_i^n = v_{i1}^o + v_{i2}^o \tag{8}$$

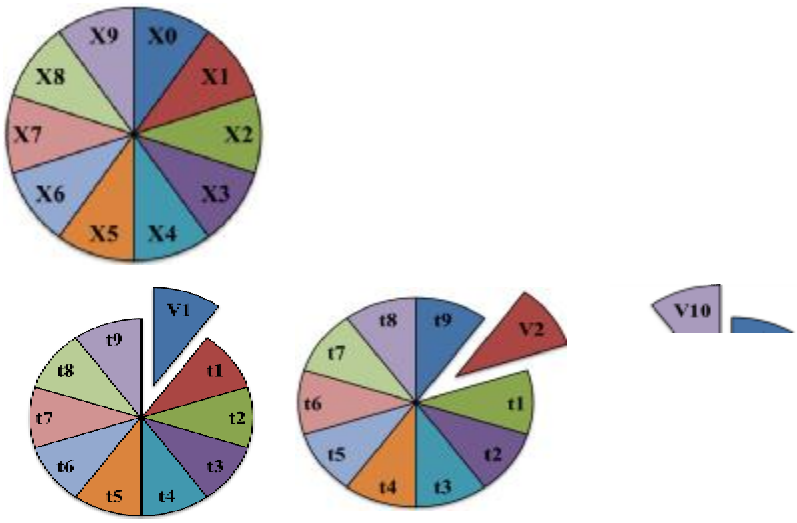
ç. K'nın L'li çapraz geçerliği (*L* of *K*cross-validation)

K-kat çapraz geçerlilikte *X* veri kümesinin rassal olarak *K* ayrık ve eşit parçaya bölünmesiyle $X_j, j = 0,\ldots,K-1$ çapraz geçerlilik alt kümesi elde edilir. Eğitim kümelerini oluşturmak için ise *K* adet kümeden biri geçerleme için ayırılır. Kalan *K*-1 adet küme birleştirilerek eğitim kümesi oluşturulur. Bu durumda oluşan *K* adet eğitim kümesinden V geçerleme kümesini, T eğitim kümesini ve A_i i'inci $(i = 1,\ldots,K)$ çapraz geçerlilik eğitim alternatifini göstermek üzere aşağıdaki gibi ifade edebiliriz:

$$A_i = \left\{ V_i = X_{i-1}, T_i = \bigcup_j \left(X_j \mid j \neq i-1 \right) \right\} \tag{9}$$

Yukarıdaki ifadeyi açık olarak yazarsak aşağıdaki ifadeyi elde ederiz Alpaydın (2010):

$$\begin{aligned}
A_1 &= \{V_1 = X_0, T_1 = X_1 \cup X_2 \cup X_3 \cup \ldots \cup X_{K-2} \cup X_{K-1}\} \\
A_2 &= \{V_2 = X_1, T_2 = X_2 \cup X_3 \cup \ldots \cup X_{K-2} \cup X_{K-1} \cup X_0\} \\
A_3 &= \{V_3 = X_2, T_3 = X_3 \cup \ldots \cup X_{K-2} \cup X_{K-1} \cup X_0 \cup X_1\} \\
&\vdots \\
A_K &= \{V_K = X_{K-1}, T_K = X_1 \cup X_2 \cup X_3 \cup \ldots \cup X_{K-3} \cup X_{K-2}\}
\end{aligned} \tag{10}$$



Şekil 1 K=10 için geçerlilik kümeleri

K-kat çapraz geçerlilik için $X_j, j = 0,...,9$ ayrık alt kümeler oluşturulur. BuX_j alt kümelerinden bir tanesi V_i olarak geçerleme için kalan 9 tanesi $\left(t_j \mid j \neq i\right)$ birleştirilerek eğitimde kullanılır. Eğitim kümelerinin bileşimi bir defa eğitimde kullanıldıktan sonra geçerleme için kullanılan alt küme sıradaki alt küme olarak belirlenir. Kalan 9 küme tekrar birleştirilerek yeni eğitim kümesi elde edilir. Her seferinde geçerlemede kullanılacak küme bir sonraki olacak şekilde kullanılır ve her alt küme bir defa geçerlemede kullanılmak üzere toplam 10 adet eğitim veri kümesi alternatifi oluşturulur. Her adımda farklı bir küme kullanılarak yapay sinir ağlarının ezberleme yapmasını engellenmeye çalışılır.

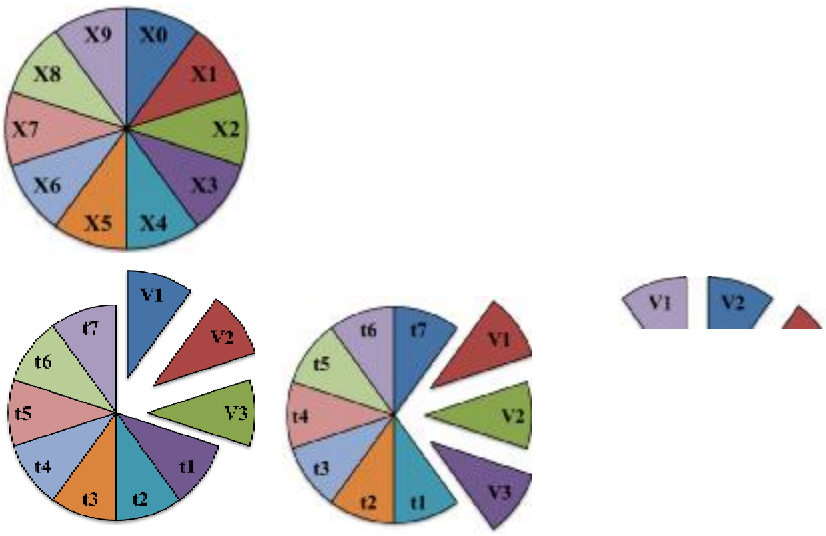
K'nın *L*'li çapraz geçerliliğinde ise X veri kümesinin rassal olarak K ayrık ve eşit parçaya bölünerek $X_j, j = 0,\ldots,K-1$ çapraz geçerlilik alt veri kümeleri

elde edilir. *K*'nın *L*'li çapraz geçerliliğinde ardışık L adet alt veri kümesi ayrı ayrı geçerleme için kullanılırken kalan $K-L$ adet küme birleştirilerek eğitim kümesi oluşturulur. Bu durumda oluşan *K*adet eğitim kümesini V geçerleme kümesini, T eğitim kümesini, A eğitim alternatifleri kümesini ve A_i i'inci $(i=1,\ldots,K)$ çapraz geçerlilik eğitim alternatifini göstermek üzere eğitim alternatif veri kümelerini aşağıdaki gibi ifade edebiliriz:

$$A=\bigcup_i A_i \tag{11}$$

$$A_i=\left\{T_i=\bigcup_j\left(x_j \mid x_j\notin\bigcup_{m=1}^{L}V_m\right),V_1=X_{(i-1)\bmod K},V_2=X_{(i)\bmod K},\ldots,V_L=X_{(i+L-2)\bmod K}\right\} \tag{12}$$

$$\begin{aligned}
A_1&=\left\{T_1=\bigcup_j\left(x_j \mid x_j\notin\bigcup_{m=1}^{L}V_m\right),V_1=X_{(0)\bmod K=0},V_2=X_{(1)\bmod K},\ldots,V_L=X_{(L-1)\bmod K}\right\}\\
A_2&=\left\{T_2=\bigcup_j\left(x_j \mid x_j\notin\bigcup_{m=1}^{L}V_m\right),V_1=X_{(1)\bmod K},V_2=X_{(2)\bmod K},\ldots,V_L=X_{(L)\bmod K}\right\}\\
&\vdots\\
A_K&=\left\{T_K=\bigcup_j\left(x_j \mid x_j\notin\bigcup_{m=1}^{L}V_m\right),V_1=X_{(K-1)\bmod K},V_2=X_{(K)\bmod K=0},\ldots,V_L=X_{(K+L-2)\bmod K}\right.
\end{aligned} \tag{13}$$



Şekil 2*K*=10, *L*=3 için geçerleme kümeleri

K-kat çapraz geçerlilik *K*'nın *L*'li çapraz geçerliliğinde $L=1$ için özel durumudur. Yukarıdaki şekilde $K=10$ ve $L=3$ için eğitim veri kümeleri alternatifleri görülmektedir. K-kat çapraz geçerlilik için $X_j, j=0,...,K-1$ ayrık alt kümeler oluşturulur. Bu X_j alt kümelerinden ardışık üç tanesi sırasıyla V_1, V_2 ve V_3 olarak ayrı ayrı geçerleme için, kalan 7 tanesi $\left(t_i \mid i=1,...,7\right)$ ise birleştirilerek eğitimde kullanılır. Eğitim kümelerinin bileşimi bir defa eğitimde kullanıldıktan sonra geçerleme için kullanılan ilk alt küme bir sonraki alt küme olarak belirlenir. Ardından gelen iki alt veri kümesi de ayrı ayrı geçerleme kümesi olmak üzere toplam üç geçerleme kümesi belirlenir. Kalan 7 alt küme tekrar birleştirilerek yeni eğitim kümesi elde edilir. Her seferinde geçerlemede kullanılacak ilk küme bir sonraki olacak şekilde değiştirilir ve her alt küme bir defa ilk geçerleme kümesi olmak üzere toplam 10 adet eğitim veri kümesi alternatifi oluşturulur. Her adımda farklı bir küme kullanılarak yapay sinir ağlarının ezberleme yapması engellenmeye çalışılır.

4. UYGULAMA ÖRNEĞİ VE SONUÇLAR (APPLICATION EXAMPLE AND RESULTS)

a. Ve, Veya ve Özel Veya Mantık Birleşik Problemi

Bu çalışmada "ve" mantık operatörünün "veya" mantık operatörünün ve "özel veya" mantık operatörlerinin birleşiminden oluşturulmuş yapay bir problem kullanılmıştır.

Ve Mantık Problemi

Doğruluk tablosu aşağıda verilen "ve" mantık operatörü her iki koşulun da sağlandığı durumları ifade etmede kullanılır. Örneğin iki kümenin keşişim kümesinde yer alan elemanları her iki kümeye birden ait olması gerekir, sadece birine üye olması veya ikisine birden üye olmaması durumunda kesişim kümesinde yer almaz.

Tablo 1 Ve mantık problemi

p	q	$p \wedge q$
Yanlış	Yanlış	Yanlış
Yanlış	Doğru	Yanlış
Doğru	Yanlış	Yanlış
Doğru	Doğru	Doğru

Veya Mantık Problemi

Doğruluk tablosu aşağıda verilen "veya" mantık operatörü ise herhangi bir koşulun sağlandığı durumları ifade etmede kullanılır. Örnek olarak iki kümenin birleşimi verilebilir. Bir kümeye ait olan veya her iki kümeye birden ait olan elemanlar bileşke kümeye de aittir. Ancak hiçbir kümeye ait olmayan nesneler ise bileşke kümeye de ait değillerdir

Tablo 2 Veya mantık Problemi

p	q	$p \vee q$
Yanlış	Yanlış	Yanlış
Yanlış	Doğru	Doğru
Doğru	Yanlış	Doğru
Doğru	Doğru	Doğru

Özel Veya Mantık Problemi

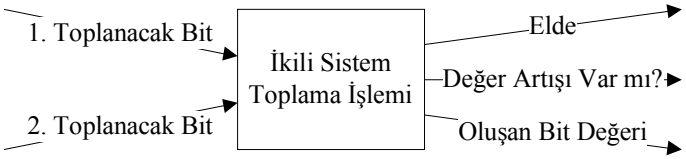
Doğruluk tablosu aşağıda verilen "özel veya" mantık operatörü "$\oplus$" sembolü ile gösterilir ve sadece bir koşulun sağlandığı durumları ifade etmede kullanılır. Örnek olarak iki kümeden sadece birine ait olan nesneleri ifade etmede kullanılabilir.

Tablo 3 Özel Veya mantık problemi

p	q	$p \oplus q$
Yanlış	Yanlış	Yanlış
Yanlış	Doğru	Doğru
Doğru	Yanlış	Doğru
Doğru	Doğru	Yanlış

Bütünleşik Mantık Problemi

Bu çalışmada yukarıda açıklaması verilen mantık problemlerinin birleşimiyle yeni bir problem oluşturulmuştur. Örnek olarak ikili sistemde toplama işlemi ele alınmıştır. İkili sistemde sayıların tüm basamakları ya "0" ya da "1" değerini almaktadır. Bu durumda ikili sistemde, iki sayı toplanırken herhangi bir basamak için toplama işlemi sonucunda artık değer olup olmadığı "ve" operatörü ile, basamakta değer artışı olup olmadığı "veya" operatörü ile sonuç bit değeri, "özel veya" operatörü ile gösterir. Oluşturulan sistemin karakutu gösterimi aşağıda verilmiştir.



Şekil 3Bit toplama işlemi karakutu gösterimi

Bütün sayısal işlemlerin temeli olduğu için bu toplama işlemi ve elektronik uygulaması çok önemlidir.

Tablo 4Bit toplama işlemi

G1: 1. Toplananın İlgili Basamak Bit Değeri	G2: 2. Toplananın İlgili Basamak Bit Değeri	Ç1: Toplama işlemi sırasında elde var mı? (Ve)	Ç2: Değer artışı var mı? (Veya)	Ç3: Sonuç Bit Değeri (Özel Veya)
1	1	1	1	0
1	0	0	1	1
0	1	0	1	1
0	0	0	0	0

b. Modellerin Değerlendirilmesi (Karşılaştırılması) (Evaluation of Models)

(1) Ortalama Hata Karesi

Ortalama Hata Karesi, N adet model çıktısının hatasının hesaplanmasında aşağıdaki gibi hesaplanır:

$$\text{OHK} = \frac{1}{N}\sum_{i=1}^{N}\left(\hat{y}_i - y_i\right)^2 \qquad (14)$$

burada $\hat{y}_i$ modelin i'inci sonucu, y_i ise i'inci sonucun olması gereken değerdir.

(2)İterasyon Sayısı

Algoritmaların iterasyon sayısı çalışma süresini belirlemektedir. Bu çalışmada FA01V01 algoritması bir adet çok katmanlı algılayıcı üretmektedir. Oluşan çok katmanlı algılayıcı geri yayılım algoritması ile eğitilmektedir. Bu çalışma kapsamında oluşan çok katmanlı algılayıcıların aynı şartlar altında geri yayılım algoritması ile eğitiminde geçen iterasyon sayıları karşılaştırılmıştır. Burada amaç FA01V01 algoritması ile yapısı oluşturulan ÇKA'nın geri yayılım algoritması ile ara ara eğitilmesinin, ÇKA'nın yapısının sabit tutularak geri yayılım algoritması ile eğitilmesi arasındaki farkı tespit etmektir.

c. FA01V01 Algoritmasının Uygulanması

Yeni geliştirilen algoritmanın uygulaması için Java programlama dili kullanılarak yeni bir yazılım geliştirilmiştir. Geliştirilen yazılım (NeuroBee) Ve, Veya ve Özel Veya problemine uygulanmıştır. Normal sabit yapıda oluşturulan çok katmanlı algılayıcılar ile karşılaştırma yapmak için NeuroSolutions adlı program kullanılmıştır.

Uygulamada üç katmanlı (tek gizli katmana sahip) bir çok katmanlı algılayıcı (ÇKA) yapısı benimsenmiştir. NeuroSolutions yazılımı ve NeuroBee yazılımı için farklı gizli sinir hücreleri için çalışma tekrarlanmıştır. Tüm bu tekrarlarda eğitim için geri yayılım algortiması (backpropagation) kullanılmış, sinaptik ağırlıklar bu algoritma ile güncellenmiştir. Geri yayılım algoritmasının durma koşulları olarak;100 iterasyon boyunca 0,0001'den daha fazla iyileşme sağlanamaması veya 10 000 iterasyon'a ulaşılması belirlenmiştir. Her iki yazılımda da aşağıda formülü verilen adaptif öğrenme oranı kullanılmıştır (Alpaydın, 2010):

$$\Delta\eta = \begin{cases} +a \text{ if } E^{t+T} < E^{T} \\ -b\eta \text{ diğer} \end{cases} \qquad (15)$$

burada $\Delta\eta$ öğrenme oranınındaki değişimi, E^{T} hataları, a ve b ise sabit değerlerdir. Böylece eğitim kümesinin hatası azaldıkça öğrenme oranı sabit miktarlarda arttırılır, eğitim kümesinin hata oranı arttıkça ise geometrik olarak öğrenme oranı azaltılır (Alpaydın, 2010).

Her iki yazılım da adaptif öğrenme oranı kullansa da NeuroSolutions yazılımı her katman için ayrı bir öğrenme oranı kullanırken NeuroBee tüm dinamik ypay sinir ağı için tek öğrenme oranı kullanmaktadır. Ayrıca her iki yazılımda da geri yayılım algoritmasında momentum kullanılmış ve bu değer NeuroSolution yazılımının otomatik ayarı olan 0,7 olarak kullanılmıştır.

Algoritmaların performanslarını kıyaslamadan önce algoritmaların sonuçlarının normal dağılıma uygunluğunun test edilmesi gerekmektedir. SPSS yazılımında uygulanan normallik testi sonucunda hata karalerinin normal dağılıma uymadıkları gözlenmiş ve bu yüzden normal dağılım kabulu olmayan Mann-Whitney-U testi SPSS yazılımıyla uygulanmıştır. Bu amaçla "H0: İki yöntemin sonuçlarının ortalama OHK'leri eşittir." Hipotezine karşılık aşağıdaki hipotezler ile ortalamaların eşitliği test edilmiştir

: Tablo 5Elde edilen ortalama hata kareleri

S.N	Hipotez No	Hipotez	Mann-Whitney-U	Anlamlılık	Hipotez Sonucu
1	**H1**	2 GSH için FA01V01 ile sabit yapılı ÇKA'nın ortalama OHK'leri eşit değildir.	120	0,030	H0 RED
2	**H2**	3 GSH için FA01V01 ile sabit yapılı ÇKA'nın ortalama OHK'leri eşit değildir.	80	0,001	H0 RED
3	**H3**	4 GSH için FA01V01 ile sabit yapılı ÇKA'nın ortalama OHK'leri eşit değildir.	80	0,001	H0 RED
4	**H4**	5 GSH için FA01V01 ile sabit yapılı ÇKA'nın ortalama OHK'leri eşit değildir.	20	0,000	H0 RED

Yapılan istatistiksel tesler sonucunda H0 hipotezleri reddedilmiştir. Bu durumda hiçbir GSH sayısı için hata kareleri eşit çıkmamıştır. 2 GSH için sabit yapılı ÇKA daha iyi sonuç verirken, 3,4, ve 5 GSH'li ÇKA için FA01V01 algoritması sabit yapıya göre daha düşük ortalama hata kareleri vermiştir.

Algoritmaların Ortalama Hata Karesi (OHK) açısından 20'şer tekrar sonucunda elde edilen genel özellikleri aşağıdaki tabloda verilmiştir:

	2 GSH için		3 GSH için		4 GSH için		5 GSH için	
	FA01V01 ile ÇKA	Sabit Yapılı ÇKA	FA01V01 ile ÇKA	Sabit Yapılı ÇKA	FA01V01 ile ÇKA	Sabit Yapılı ÇKA	FA01V01 ile ÇKA	Sabit Yapılı ÇKA
% 100 Doğru Bulma Oranı	0,90	1,00	0,95	1,00	0,90	1,00	0,95	1,00
Maks OHK	0,102	0,025	0,095	0,025	0,096	0,025	0,092	0,024
Min OHK	0,021	0,024	0,018	0,024	0,014	0,024	0,013	0,024
Ortalama OHK (μ)	0,035	0,025	0,026	0,024	0,029	0,024	0,019	0,024
σ	0,023	0,000	0,017	0,000	0,028	0,000	0,017	0,000
Ortalama Sıra	24,50	16,50	14,50	26,50	14,50	26,50	11,50	29,50

Tablo 6 FA01V01 ve sabit yapılı ÇKA için elde edilen OHK istatiskleri

Algoritmaların geri yayılım iterasyon sayıları bakımından 20 tekrar sonucunda elde edilen genel özellikleri aşağıdaki tabloda verilmiştir.

		2 GSH için		3 GSH için		4 GSH için		5 GSH için	
		FA01V01 ile ÇKA	Sabit Yapılı ÇKA	FA01V01 ile ÇKA	Sabit Yapılı ÇKA	FA01V01 ile ÇKA	Sabit Yapılı ÇKA	FA01V01 ile ÇKA	Sabit Yapılı ÇKA
Maks İterasyon Sayısı		1495	2137	1328	1559	1052	1362	938	1330
Min İterasyon Sayısı		822	1190	686	998	535	904	622	776
Ortalama İterasyon Sayısı (μ)		1147,25	1539,95	1013,6	1239,85	841,7	1092,7	786,2	993,55
σ		180,15	257,38	135,49	186,99	138,91	116,20	68,01	129,30
Ortalama		12,25	28,75	13,50	27,50	11,45	29,55	11,45	29,55

Tablo 7 FA01V01 ve sabit yapılı ÇKA için elde edilen geri yayılım iterasyon sayısı istatiskleri

Algoritmaların performanslarını kıyaslamadan önce algoritmaların sonuçlarının normal dağılıma uygunluğunun test edilmesi gerekmektedir. SPSS yazılımında uygulanan normallik testi sonucunda hata karalerinin sadece 2 ve 3 GSH'li ÇKA için normal dağılıma uydukları gözlenmiştir. Normal dağılıma uymayan durumlar için yüzden normal dağılım kabulu bulunmayan Mann-Whitney-U testi SPSS yazılımıyla uygulanmıştır. 4 ve 5 GSH'li ÇKA'lar için öncelikle varyans testi yapılması gerekmektedir. Bu amaçla "H0: İki yöntemin iterasyon sayılarının varyansları eşittir." Hipotezine karşılık aşağıdaki hipotezler ile ortalamaların eşitliği test edilmiştir:

Tablo 6 İterasyon sayıları için eşit varyans testi

Sıra No	Hipotez No	Hipotez	F	Anlamlılık	Hipotez Sonucu
1	**H1**	4 GSH için FA01V01 ile sabit yapılı ÇKA'nın iterasyon sayılarının varyansları eşit değildir.	0,0116	0,915	H0 KABUL
2	**H2**	5 GSH için FA01V01 ile sabit yapılı ÇKA'nın iterasyon sayılarının varyansları eşit değildir.	4,366	0,043	H0 RED

Yukarıdaki tabloda verilen sonuçlar doğrultusunda H2hipotezi kabul edilmekte ve H1 hipotezi ise reddedilmektedir. Bu durumda her iki yöntem için beş gizli sinir hücreli (GSH) durumlarda iterasyon sayılarının varyanslarının eşit olmadığını, dört gizli sinir hücreli yapılar için ise iterasyon sayılarının varyanslarının eşit olduğunu görülmektedir. Ortalamaların karşılaştırması yapılırken (t-testi) eşit varyans durumları dikkate alınarak t-testi icra edilecektir. Bu durumda "H0: İki yöntemin sonuçlarının iterasyon sayıları eşittir." Hipotezine karşılık aşağıdaki hipotezler ile ortalamaların eşitliği SPSS yazılımıyla test edilmiştir:

Tablo 7 İterasyon sayılar için Mann-Whitney-U testi sonuçları

Sıra No	Hipotez No	Hipotez	Mann-Whitney-U / t	Anlamlılık	Hipotez Sonucu
1	**H1**	2 GSH için FA01V01 ile sabit yapılı ÇKA'nın iterasyon sayıları eşit değildir.	35	0,000	H0 RED
2	**H2**	3 GSH için FA01V01 ile sabit yapılı ÇKA'nın iterasyon sayıları eşit değildir.	60	0,000	H0 RED
3	**H3**	4 GSH için FA01V01 ile sabit yapılı ÇKA'nın iterasyon sayıları eşit değildir.	-6,198	0,000	H0 RED
4	**H4**	5 GSH için FA01V01 ile sabit yapılı ÇKA'nın iterasyon sayıları eşit değildir.	-6,347	0,000	H0 RED

Yukarıdaki tablo incelendiğinde dört durumda da H0 hipotezinin reddedilerek alternatif hipotezin kabul edildiği görülmektedir. Bu durumda iki yöntem arasında geri yayılım algoritmasının eşit sayıda iterasyonla sonuca ulaşmadığını söyleyebiliriz. Ortalamalara baktığımızda FA01V01 algoritmasının ortalama olarak sürekli daha az iterasyonda sonuca ulaştığını görmekteyiz. Bu durumda FA01V01 algortimasının geri yayılım ile öğrenme algoritmasının toplam iterasyon sayısını azalttığını söyleyebiliriz.
Dikkat çeken diğer bir nokta ise gizli katmandaki sinir hücresi sayısı arttıkça özellikle FA01V01 algoritmasında ihtiyaç duyulan geri yayılım algoritması iterasyon sayısının azalmasıdır. Ayrıca 20'şer denemede bati yapılı ÇKA'nın 2, 3, 4 ve 5 GSH'li yapıda 0,024 OHK değerinin altına inemediği FA01V01 algoritması ile oluşturulan ÇKA'nın ise 2 GSH için 0,021, 3 GSH için 0,018; 4 GSH için 0,014 ve 5 GSH için 0,013 OHK değerlerine kadar inebildikleri tespit edilmiştir.

5. SONUÇ VE ÖNERİLER (DİSCUSSİON AND SUGGESTİONS)

Bu çalışmada geri yayılım algoritması ile eğitilen tek gizli katmanlı ÇKA modeli ile K-küme çapraz geçerlilik ile oluşturulan tek katmanlı ÇKA modeli karşılaştırılmıştır. Gizli katmandaki gizli sinir hücreleri sabitlenerek farklı sayıda gizli sinir hücresini içeren ÇKA için çalışma tekrarlanmıştır. Elde edilen sonuçlar ortalama hata karesi (OHK) ve geri yayılım algoritmasında geçen toplam iterasyon sayısı dikkate alınarak incelenmiştir. Bu incelemede bir iterasyon olarak, geri yayılım algoritmasının tüm eğitim veir kümesini bir tam defa kullanması (epoch) kabul edilmiştir.

Sabit yapılı ÇKA için NeuroSolutions yazılımı, FA01V01 ile oluşturulan ÇKA için ise JAVA programlama dili kullanılarak geliştirilen NeuroBee yazılımı kullanılmıştır. Karşılaştırma yapılabilmesi için her iki yazılımda geri yayılım algoritmasının ve momentum yönteminin parametreleri aynı alınmıştır.

Yapılan istatitiksel incelemeler sonucunda 2, 3, 4 ve 5 gizli sinir hücreli ÇKA için ve, veya, özel veya bütünleşik probleminde ortalama hata kareleri açısından her iki yöntem arasında istatistiksel olarak anlamlı bir fark oluştuğu gözlenmiştir. Bu fark 2 GSH için sabit yapılı ÇKA'nın daha küçük OHK ürettiği, 3,4 ve 5 GSH'li ÇKA'lar için ise FA01V01'in daha küçük OHK'si ürettiğini göstermektedir. Ayrıca FA01V01 algoritması ile oluşturulan ÇKA'da gözlenen en küçük OHK sabit yapılı ÇKA'da gözlenen en küçük OHK'nden de daha küçük olmuştur. Geri yayılım algoritması için geçen toplam iterasyon sayılarının ise FA01V01 algoritmasında çok daha az iterasyon sonrasında sinaptik ağırlıkların yakınsadığı ve anlamlı iyileşme kaydedilemediği gözlenmiştir.

Geri yayılım algoritmasında FA01V01 algortimasının daha az iterasyon (ya da epoch) gerektirmesinin sebebi olarak iki temel farklılık öne çıkmaktadır. Bunlardan ilki girdi ve çıktıyı sabitleyerek eğitim yapmak yerine, çıktı hücrelerini sırasıyla eğiterek eğitimi basit kademelere bölmesi ve her seferinde bir adet çıktı sinir hücresi ekleyerek devam etmesidir.

İkinci önemli farklılık ise *K*'nın *L*'li çapraz geçerliliğinde zaten genelleme yeteneğini arttırması muhtemel çıktı sinir hücrelerinin tespit edilerek mimari yapıya eklenmesidir. Eklenen her yeni çıktı sinir hücresinin L adet çapraz geçerlilik kümesinde ayrı ayrı iyileşme sağlamasının geri yayılım algoritmasının iterasyon sayısının düşmesine neden olduğu değerlendirilmektedir.

Önerilen FA01V01 algoritmasının sağladığı performans artışı, özellikle yapay sinir ağlarının oluşturulması ve eğitilmesine farklı bir bakış açısı getirmesine dayanmaktadır. Bu farklı bakış açısı aslında yapay zeka yöntemlerinin düşünebilen bir yapıya ulaşması için aşılması gereken üç temel problemden birine çözüm önerisidir.

Yapay zeka çalışmalarının insan gibi düşünebilen yapıya yaklaşabilmeleri için aşmaları gereken ilk problem, yapay zeka modellerinin hangi yeni işlevleri veya özellikleri kazanması gerektiğidir. İkinci temel problem, bu yeni işlev ve özellik kazanma işlemlerinin yapılması için gerekli koşulların tespitidir. Üçüncüsü ve sonuncusu ise yeni işlev ve özellik kazanma işlemlerinin ne şekilde yapılacağıdır. Yapay zeka önünde duran bu üç temel engeli "neyi", "ne zaman" ve "nasıl" olarak kısaca özetleyebiliriz. Bu çalışma "nasıl" sorusunu sorarak, üçüncü temel engelin aşılması için sade bir çözüm getirmiştir. Bu noktadan sonra yapılabilecek yapay zeka çalışmaları "nasıl" sorusuna cevap arayarak öğrenme kavramını geliştirecek yeni yöntemler geliştirebilirler veya bir sonraki aşamaya geçerek "ne zaman" veya "neyi" sorusuna cevap arayabilirler.

Kaynakça (References)

Alpaydın, E., 2010. **Introduction to Machine Learning**. The MIT Press, London, England.

Ash, T.**,** 1989. "Dynamic node creation in backpropagation networks". **Connection Science**, 1(4): 365-375.

Azimi-Sadjadi, M. R., Sheedvash, S. ve Trujillo, F. O.**,** 1993. "Recursive Dynamic Node Creation". **IEEE TRANSACTIONS ON NEURAL NETWORKS**, 4(2): 242-256.

Back, T. ve Kursawe, F.,1995. "Evolutionary algorithms for fuzzy logic: A brief overview". **Fuzzy logic and soft computing** (Ed.). World Scientific, Singapore, (Sf.3-10).

Broomhead, D. S. ve Lowe, D.**,** 1988. "Multivariable Functional Interpolation and Adaptive Networks". **Complex Systems**, 2: 321-355.

Buntine, W. L. ve Weigend, A. S.**,** 1991. "Bayesian backpropagation". **Complex Systems**, 5: 603–643.

Burkitt, A. N.**,** 1991. "Optimization of the architecture of feed-forward neural networks with hidden layers by unit elimination". **Complex Systems**, 5: 371-380.

Castellano, G., Fanelli, A. M. ve Pelillo, M., 1993. "An Empirical Comparison of Node Pruning Methods for Layered Feed-forward Neural Networks". **International Joint Conference on Neural Networks**. Nagoya, Japan. 1**:** 321-326.

Castellano, G., Fanelli, A. M. ve Pelillo, M.**,** 1997. "An Iterative Pruning Algorithm for Feedforward Neural Networks". **IEEE TRANSACTIONS ON NEURAL NETWORKS,** 8(3): 519-531.

Castillo, P. A., Merelo, J. J., Prieto, A., Rivas, V. ve Romero, G.**,** 2000. "G-Prop: Global optimization of multilayer perceptrons using GAs". **Neurocomputing**, 35, 149-163.

Çınar, D., 2007. **Hidroelektrik Enerji Üretiminin Hibrid Bir Model ile Tahmini**. **Industrial Engineering**. Istanbul, Istanbul Teknik Üniversitesi. M.S.**:** 108.

Doganis, P., Alexandridis, A., Patrinos, P. ve Sarimveis, H.**,** 2006. "Time series sales forecasting for short shelf-life food products based on artificial neural networks and evolutionary computing". **Journal of Food Engineering**, 75: 196-204.

Fahlman, S. E. ve Lebiere, C.,1990. **The cascade-correlation learning architecture**. **Advances in Neural Information Processing Systems**. Touretzky, D. S. (Ed.). Morgan Kaufmann, San Mateo, CA, (Sf.524–532).

Fayyad, U. M., Shapire, G. P., Smyth, P. ve Uthurusamy, R., 1996. **Advances in knowledge discovery and data mining**. MIT Press, Cambridge, MA.

Ferentinos, K. P.**,** 2005. "Biological engineering applications of feedforward neural networks designed and parameterized by genetic algorithms". **Neural Networks**, 18: 934-950.

Frattale-Mascioli, F. M. ve Martinelli, G.**,** 1995. "A constructive algorithm for binary neural networks: the oil-spot algorithm". **IEEE Transactions on Neural Networks**, 6(3): 794 - 797

Frean, M.**,** 1990. "The Upstart Algorithm a method for constructing and training feedforward neural networks". **Neural Computation,**2: 198-209.

Friedman, J. H. ve Stuetzle, W.**,** 1981. "Projection Pursuit Regression". **Journal of the American Statistical Association**, 73(376): 817-823.

Fritzke, B., 1993. **Growing cell structures--A self-organizing network for unsupervised and supervised learning**, International Computer Science Institution.

Fritzke, B.**,** 1994a. "Growing cell structures--A self-organizing network for unsupervised and supervised learning". **Neural Networks**: 1441-1460.

Fritzke, B.,1994b. **Supervised Learning with Growing Cell Structures**. **Advances in Neural Information Processing Systems** Cowan, J. D., Tesauro, G. ve Alspector, J. (Ed.). Morgan Kaufmann, San Mateo, CA, USA, (Sf.255-262).

Fritzke, B.**,** 1995a. "Growing Grid - a self - organizing network with constantneighborhood range and adaptation strength". **Neural Processing Letters**, 2(5): 9-13.

Fritzke, B.,1995b. **A Growing Neural Gas Network Learns Topologies**. **Advances in Neural Information Processing Systems**. Tesauro, G., Touretzky, D. S. ve Leen, T. K. (Ed.). MIT Press, Cambridge, MA).

Gallant, S. I., 1986. **Optimal Linear Discriminants**. **IEEE 8th Conferance on Pattern Recognition**. Paris, France.

Gallant, S. I.**,** 1990. "Perceptron-based learning algorithms". **IEEE Transactions on Neural Networks**, 1(2): 179-191.

Ghiassi, M. ve Saidane, H.**,** 2005. "A dynamic architecture for artificial neural networks". **Neurocomputing**, 63: 397-413.

Goldberg, D., 1989. **Genetic algorithms in search, optimization and machine learning**. Addison-Wesley, Reading, MA.

Han, H.-G. ve Qiao, J.-F.**,** 2013. "A structure optimisation algorithm for feedforward neural network construction". **Neurocomputing**, 99: 347-357.

Hwang, J.-N., Lay, S.-R., Maechler, M., Martin, R. D. ve Schimert, J.**,** 1994. "Regression Modeling in Back-Propagation and Proiection Pursuit

Learning ". **IEEE TRANSACTIONS ON NEURAL NETWORKS**, 5(3): 342-353.

Ivakhnenko, A. G.,1984. **Self-Organizing Methods in Modeling: GMDH Type Algorithms**. **Statistics: Textbooks and Monographs**. Farlow, S. J. (Ed.). Marcel Dekker Inc., New York:).

Karnin, E. D.**,** 1990. "A simple procedure for pruning back-propagation trained neural networks". **IEEE Transactions on Neural Networks**, 1(2): 239–242.

Kennedy, J. ve Eberhart, R., 1995. "Particle swarm optimization." *IEEE International Conferance on Neural Networks*, Perth, Australia.

Kim, B. ve Bae, J.**,** 2005. "Prediction of plasma processes using neural network and genetic algorithms". **Solid-State Electronics**, 49: 1576-1580.

Kiranyaz, S., Ince, T., Yildirim, A. ve Gabbouj, M.**,** 2009. "Evolutionary artificial neural networks by multi-dimensional particle swarm optimization". **Neural Networks**, 22(10): 1448-1462.

Koza, J., 1992. **Genetic programming: On the programming of computers by means of natural selection**. MIT Press, Cambridge, MA.

Kuo, R. J.**,** 2001. "A sales forecasting system based on fuzzy neural network with initial weights generated by genetic algorithm". **European Journal of Operational Research**, 129: 496-517.

Kwok, T.-Y. ve Yeung, D.-Y.**,** 1997a. "Constructive Algorithms for Structure Learning in Feedforward Neural Networks for Regression Problems". **IEEE TRANSACTIONS ON NEURAL NETWORKS**, 8(3): 630-645.

Kwok, T.-Y. ve Yeung, D.-Y.**,** 1997b. "Objective Functions for Training New Hidden Units in Constructive Neural Networks". **IEEE TRANSACTIONS ON NEURAL NETWORKS**, 8(5): 1131-1148

Li, Z., Cheng, G. ve Qiang, X., 2010. **Some Classical Constructive Neural Networks and their New Developments**. **2010 International Conference on Educational and Network Technology (ICENT 2010)**. Qinhuangdao, China 174-178.

Ma, L. ve Khorasani, K.**,** 2003. "A new strategy for adaptively constructing multilayer feedforward neural networks". **Neurocomputing**, 51: 361-385.

Mezard, M. ve Nadal, J.-p.**,** 1989. "Learning in feedforward layered networks: The tiling algorithm". **Journal of Physics A: Math. Gen**, 22(12): 2191-2203.

Moody, J. ve Darken, C.**,** 1989. "Fast Learning in Networks of Locally-Tuned Processing Units". **Neural Computation**, 1: 281-294.

Mozer, M. C. ve Smolensky, P.,1989. **Skeletonization: a technique for trimming the fat from a network via relevance assessment**. **Advances in Neural Information Processing** Touretzky, D. S. (Ed.). Morgan Kaufman, San Mateo, CA, (Sf.107–115).

Ngom, A., Stojmenovic, I. ve Milutinovic, V.**,** 2001. "STRIP—A Strip-Based Neural-Network Growth Algorithm for Learning Multiple-Valued Functions". **IEEE TRANSACTIONS ON NEURAL NETWORKS**, 12(2): 212-227.

Niska, H., Hiltunen, T., Karppinen, A., Ruuskanen, J. ve Kolehmainen, M.**,** 2004. "Evolving the neural network model for forecasting air pollution time series". **Engineering Applications of Artificial Intelligence**, 17: 159-167.

Parekh, R., Yang, J. ve Honavar, V.**,** 2000. "Constructive Neural-Network Learning Algorithms for Pattern Classification". **IEEE TRANSACTIONS ON NEURAL NETWORKS**, 11(2): 436-451.

Parker, R. E. ve Tummala, M., 1992. “Identification of Volterra systems with a polynomial neural network”.**IEEE International Conference on**

Acoustics, Speech, and Signal Processing. San Francisco, CA, USA 561–564.

Pelillo, M. ve Fanelli, A. M.,1993. **A method of pruning layered feed-forward neural networks**. **New Trends in Neural Computation (Lecture Notes in Computer Science)**. Mira, J., Cabestany, J. ve Prieto, A. (Ed.). Springer-Verlag, Berlin, (Sf.278-283).

Phatak, D. S. ve Koren, I.**,** 1994 "Connectivity and performance tradeoffs in the cascade correlation learning architecture". **IEEE Transactions on Neural Networks**, 5(6): 1045-9227

Platt, J.**,** 1991. "A Resource-Allocating Network for Function Interpolation". **Neural Computation**, 3(2): 213-225.

Puma-Villanueva, W. J., dos Santos, E. P. ve Von Zuben, F. J.**,** 2012. "A constructive algorithm to synthesize arbitrarily connected feedforward neural networks". **Neurocomputing**, 75(1): 14-32.

Reilly, D. L., Cooper, L. N. ve Elbaum, C.**,** 1982. " A neural model for category learning". **Biological Cybernetics**, 45: 35-41.

Saha, A., Wu, C.-L. ve Tang, D.-S.**,** 1993. "Approximation, Dimension Reduction, and Nonconvex ODtimization Using Linear Superpositions of Gaussians". **IEEE TRANSACTIONS ON COMPUTERS**, 42(10): 1222-1233.

Setiono, R.**,** 2001. "Feedforward Neural Network Construction Using Cross Validation". **Neural Computation**, 13: 2865-2877.

Sexton, R. S., Dorsey, R. E. ve Sikander, N. A.**,** 2004. "Simultaneous optimization of neural network function and architecture algorithm". **Decision Support Systems**, 36: 283-296.

Shin, Y. ve Ghosh, J.**,** 1995. "Ridge Polynomial Networks". **IEEE TRANSACTIONS ON NEURAL NETWORKS**, 6(3): 610-622.

Sietsma, J. ve Dow, R. J. F., 1988. **Neural net pruning - Why and how**. **International Conferance on Neural Networks**. San Diego, CA, USA**:** 325-333.

Sjogaard, S., 1992. **Generalization in cascade-correlation networks**. **Neural Networks for Signal Processing**. Helsingoer, Denmark 59 - 68.

Subirats, J. L., Franco, L. ve Jerez, J. M.**,** 2012. "C-Mantec: A novel constructive neural network algorithm incorporating competition between neurons". **Neural Networks**, 26: 130–140.

Tenorio, M. F. ve Lee, W. T.**,** 1990. "Self-organizing network for optimum supervised learning". **IEEE Transactions on Neural Networks**, 1(1): 100-110.

Thodberg, H. H.**,** 1996. "A review of Bayesian neural networks with an application to near infrared spectroscopy". **IEEE Transactions on Neural Networks**, 7: 56–72.

Verkooijen, W. ve Daniels, H.**,** 1994. "Connectionist Projection Pursuit Regression". **Computational Economics**, 7: 155-161.

Wang, D.**,** 2008. "Fast Constructive-Covering Algorithm for neural networks and its implement in classification". **Applied Soft Computing**, 8(1): 166-173.

Wang, T. ve Huang, C.**,** 2007. "Applying optimized BPN to a chaotic time series problem". **Expert Systems with Applications**, 32: 193-200.

Yang, S.-H. ve Chen, Y.-P.**,** 2012. "An evolutionary constructive and pruning algorithm for artificial neural networks and its prediction applications". **Neurocomputing**, 86: 140-149.

Yi-Hui, L.**,** 2007. "Evolutionary neural network modeling for forecasting the field failure data of repairable systems". **Expert Systems with Applications**, 33: 1090-1096.

Young, S. ve Downs, T., 1998. "CARVE - A Constructive Algorithm for Real Valued Examples". **IEEE Transactions on Neural Networks**, 9(6): 1180-1190.

Zhang, B.-T., 1994. **An incremental learning algorithm that optimizes network size and sample size in one trial**. **IEEE World Congress on Neural Networks**. Orlando, Florida. 1**:** 215-220.