**BASAMAK KORELASYON SİNİR AĞI (CASCADE CORRELATION NEURAL NETWORK)**

Basamak Korelasyon ağı, yapay sinir ağları içerisinde yeni ve öğreticili/denetimli öğrenme algoritması olarak yer almaktadır. Bir ağın içindeki sabit topoloji de ağırlıkları ayarlamak yerine, Basamak Korelasyon ilk önce minimum ağ ile başlar, sonra otomatik olarak eğitir ve yeni gizli birimleri tek tek ekleyerek çok katmanlı bir yapı oluşturur. Eğer yeni gizli birim ağa eklenirse, giriş tarafındaki ağırlıklar dondurulur. Eklenmiş olan bu birim kalıcı bir hal alarak ağda özellik sezinleyici (feature-detector) olur ki bu durum daha karmaşık özellik sezinleyiciler için yeni çıkışlar üretme veya yeni yaratımlar için, olanak sağlar. Basamak Korelasyon yapısı var olan diğer algoritmalara nazaran bazı avantajlara sahiptir: çabuk öğrenir, ağ boyutuna ve topolojisine kendi karar verir, eğitim kümesi değişse dahi kurulmuş olan yapı kendini korur ve ağın bağlantıları aracılığıyla hata sinyallerinin hiçbiri geri yayılım gerektirmez.

**Geri Yayılımlı Öğrenme Neden Yavaş?**

Basamak Korelasyon öğrenme algoritması geri yayılım öğrenme algoritmasının getirdiği problemler ve kısıtlamalar yüzünden geliştirilmiştir (Rumelhart, 1986). En önemli kısıtlamalardan bir tanesi geri yayılımın örneklerden öğrenmesi sırasında yavaş adımla gerçekleşmesi olayıdır. Basit bir kıyaslama problemlerinde bile, bir geri yayılım algoritması örneklerden istenen davranışı binlerce kez epok öğrenmesi gerektirir Bir epok (epoch) eğitim örneklem kümesi üzerinde tek bir geçiş olarak tanımlanır. Eğer örneklem kümesi ne kadar büyükse geçiş o kadar zaman almaktadır). Fahlman ve Lebiere 1990 yılında geri yayılım algoritmasının neden yavaş olduğunu sorgulamışlar ve analizleri sonucu iki önemli problemi ortaya çıkarmışlardır. Adım-büyüklüğü (step-size) problemi ile hareketli hedef (moving target) problemidir. Fahlman ve Lebiere tarafından da kabul edilen bir gerçek şu ki; geri yayılım algoritmasının yavaş olma problemine etki eden başka faktörlerin de bulunduğu ama henüz tespit edilemediğidir (Fahlman ve Lebiere, 1990).

**Adım Büyüklüğü (Step-Size) Problemi**

Standart geri yayılım metodunda görülen adım büyüklüğü probleminin nedeni metodun sadece genel hata fonksiyonun kısmi türevini (E; genel hatayı temsil etmektedir) ağdaki her bir ağırlık için hesaplama yapıyor olmasıdır.

Verilen bu türevler ile her bir adımdaki hatayı azaltmak namına ağırlık uzayında gradyan iniş (gradient descent) gerçekleştirilebilir. Bu durum açıkça belirtir ki, hata fonksiyonunun lokal minimumuna ulaşılması için, sonsuz küçüklükte değer alınır, her bir adımda gradyanı tekrar hesaplamak için yeni bir eğitim epoku çalıştırılırsa, en sonunda hata fonksiyonunun lokal minimumuna ulaşılır. Deneyimler şunu gösterir ki, çoğu durumlarda lokal minimum ya global minimum veya en azından eldeki soruna ‘yeterince iyi’ olacaktır.

Pratik bir öğrenme sisteminde şu var ki, hızlı öğrenme için, sonsuz küçük adımlar alınmaz, alabildiğince büyük adımlar atılır. Maalesef, eğer adım büyüklüğü çok büyük seçilirse, ağ güvenilir ve iyi bir sonuca yakınsayamayacaktır. Mantıklı bir adım büyüklüğü seçmek için, sadece hata fonksiyonunun eğimi değil, ayrıca ağırlık uzayında geçerli noktanın çevresinde üst seviye türevleri ki bu eğrilik derecesidir, bilinmesi gerekmektedir.

Adım büyüklüğü problemi ile ilgili birçok tasarım ortaya atılmıştır. ‘Momentum’ un (Rumelhart, 1986) bazı formları hesaplamada önceki noktalarda hata yüzeyi eğimini özetleyen ham bir yol olarak çoğu kez kullanılır. Conjugate (eşlenik) gradyan metodları yapay sinir ağları kapsamında bazı araştırmacılar tarafından (Watrous, 1988, Lapedes, 1987, Kramer, 1989) incelenmiş ve iyi sonuçlar elde edilmiştir. Bazı tasarımlar, örneğin (Franzini, 1987), bir adımdan başka bir adıma gradyanın değişimi baz alınarak adım büyüklüğünün ayarlanması önerilmiştir. Becker ve LeCun (Becker, 1988) her bir adımda hata fonksiyonunun ikinci türevinin bir yaklaşığını hesaplamış ve bu bilgiyi inişin (descent) hızına rehberlik etmesi için kullanmıştır.

Fahlman’ın geliştirdiği Quickprop algoritması (Fahlman, 1988) geri yayılımlı sistemlerde adım büyüklüğü problemlerinin çözümlenmesinde en başarılı algoritmalardan biridir. Quickprop algoritması standart geri yayılımda olduğu gibi değerlerini hesaplar, fakat basit gradyan iniş metoduna göre, Quickprop ikinci derece (second-order) metodu- Newton’un metodu ile ilişkili- ağırlıkları güncellemek için kullanır.

Quickprop ağırlık güncelleme işlemi iki tahmine dayalıdır: ilki, bir ağırlıktaki küçük değişimler diğer ağırlıklarda gözlemlenen hata gradyanı üzerinde küçük değişimlere yol açar; ikincisi ise, her bir ağırlığın hata fonksiyonu lokal quadratiktir.

Her bir ağırlık için, Quickprop - önceki eğitim döngüsünde hesaplanan eğim- ve – güncel eğim- değerinin bir kopyasını saklı tutar. Ayrıca - son gerçekleşen döngüde ağırlıkta meydana gelen değişim- değerini de tutar. Her bir ağırlık, bağımsız olarak, eğrinin minimum noktasına atlar. Yukarıda tanımlanan tahminlere göre, bu nokta muhtemel olarak aranılan minimum noktası olmayacaktır. Tekrar eden bir süreçte tek bir adım, yine de, bu algoritmayı iyi çalıştırdığını gösterebilir.

**Hareketli Hedef (Moving Target) Problemi**

Geri yayılımlı öğrenmede ikincil etkisiz durum ise hareketli hedef problemidir. Kısaca belirtilirse, ağın içindeki her bir birim ağın tüm hesaplamaları üzerinde etkin bir rol oynamaya çalışırken, her biri birimin aynı anda değişime uğraması sebebiyle elde edilmiş iyi olanlarının da bu değişime girmesi ve çözümü zorlaştırmasıdır. Gizli birimler verilen bir net’in katmanı ile direkt olarak iletişim kuramaz; her biri birim sadece kendi girişini ve ağın çıkışındaki hata sinyal yayılımını görür. Hata sinyali problemi tanımlarken birim bunu çözmeye çalışır fakat bu problem sürekli değişir. Her biri birim hızlıca hareket etmesi ve kullanışlı rollere yönelmesi yerine, görünen şudur ki, bütün birimlerin durulmaları için uzun zaman gereken karmaşık bir dansa benzer hareketlerde bulunuyorlar.

Çoğu çalışmalar gösteriyor ki, ağdaki gizli katman sayısı ne kadar arttırılırsa geri yayılım öğrenmesi o kadar yavaşlıyor. Yavaşlamanın sebebi ise hata sinyalinin zayıflaması ve dilüsyonu neticesinde ağın katmanları boyunca geri yayılım gerçekleşmesidir. Ayrıca Fahlman’a göre bu yavaşlamanın diğer bir sebebi ise hareketli hedef durumudur. Net’in içindeki her bir katmandaki birimler sürekli yukarı ve aşağı değişimler görür, bu durum birimlerin kararlı bir halde iyi çözüme ulaşmasını imkânsızlaştırır.

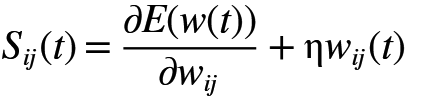
Hareketli hedef probleminin yaygın olarak kabul edilen diğer bir durumu ise ‘sürü’ etkisidir. Farz edilsin ki iki ayrı ikincil sayısallar mevcut, A ve B, bir ağda gizli birimler tarafından gerçekleştirilecekler. Ayrıca bu ikincil işler ile uğraşacak olan bazı gizli birimler olduğu farz edilsin. Birimler diğerleri ile iletişimi olmadığından, her birim bağımsız olarak bu iki problemi çözmeye çalışacaktır. Eğer A, B’den daha büyük veya daha uygun (cohorent) hata sinyali oluşturursa, tüm birimler A’ya odaklanma eğilimi gösterir ve B’yi göz ardı ederler. A çözüldüğü zaman, gereksizce, birimler B işini kalan hata olarak görür. Eğer tüm birimler B’ye aynı anda hareket ederlerse, A problemi gözden kaybolur. Çoğu durumlarda, ‘sürü’ şekilde hareket eden birimler sonunda ayrılacak ve aniden her iki işi halletmeye çalışacak, fakat bu durum oluşması çak fazla zaman gerektirecektir. Tüm birimlerin aynı davranmasını engellemek için bir geri yayılım ağındaki ağırlıklara tesadüfi olarak ilk değerler verilir, ama bu ilk değişkenlik ağ eğitildikçe yok olur.

Hareketli hedef problemi tek bir yolla yenilebilir, o da şudur ki; sadece birkaç tane ağırlığa veya birime izin verilir ağın değişimi için, diğerleri sabit tutulur. Basamak korelasyon algoritması bu tekniğin daha uç versiyonunu kullanır. Herhangi bir zamanda ki değişime sadece bir gizli birime izin verir. Ağın çoğunu sabit tutmak eğitimi yavaşlattığı gibi düşünülse de, test edilen bu strateji ağın daha hızlı öğrenmesine izin verir. Hareketli hedef problemi etkisi yok edildiği zaman, herhangi bir birim ki bu dondurulmamış olan, tüm sonuç üzerinde hemen faydalı bir rol seçer ve kararlı bir biçimde bu rolü doldurur (Fahlman ve Lebiere, 1990).

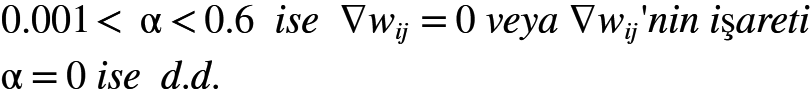
**QuickProp Algoritması Nedir?**

Yukarıda da bahsi geçen Quickprop algoritmasının aşamaları aşağıdaki gibidir;

 (1)

 (2)

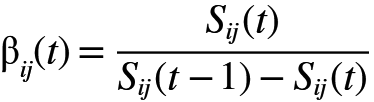
α öğrenme katsayısı, ϒ bellek sabiti (0.0001 aralığı) ağırlıkların küçülmesine ve çok fazla büyümesine limit koyar, ƞ her bir ağırlık için seçilen momentumu



 (3)

 (4)

Her bir ağırlık için seçilen momentumu değeri çok önemlidir bu aşamada. Quickprop algoritması bazen hesaplama zamanını yüzlerce kez kısaltır. Son olarak algoritmanın basit hali ise aşağıdaki gibidir (Wilamowski, 2003)

 (5)