YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ÇOK FONKSİYONLU PROTEZLER İÇİN YAPAY SİNİR AĞLARI KULLANILARAK MİYOELEKTRİK KONTROL

Elek. Yük. Müh. Bekir KARLIK

F.B.E. Elektrik Mühendisliği Anabilim Dalında hazırlanan

DOKTORA TEZİ

Tez Savunma Tarihi	:	1 Mart 1994
Tez Danışmanı	:	Prof. Dr. Halit PASTACI (Y.T.Ü.)
Jüri Üyeleri	:	Prof. Dr. Ertuğrul YAZGAN (İ.T.Ü.)
		Doç. Dr. Mehmet KORÜREK (İ.T.Ü.)

İSTANBUL, MART 1994

TEŞEKKÜR

Bu tezin hazırlanmasında ve çalışmalarım sırasında yönlendirme, teşvik ve her türlü desteği ile bana yardımcı olan Sayın Hocam Prof. Dr. Halit PASTACI' ya teşekkürlerimi sunarım.

Biyolojik işaret işleme konularında yardımlarını esirgemeyen İTÜ Elektronik ve Haberleşme Mühendisliği Bölümü Tıp Elektroniği Anabilim Dalı öğretim üyelerinden Doç. Dr. Mehmet KORÜREK'e, yapay sinir ağları ile ilgili hususlarda bana yol gösteren Boğaziçi Üniversitesi Bilgisayar Mühendisliği Bölümü öğretim üyesi Sayın Doç Dr. Fikret GÜRGEN'e ve arkadaşlarıma yardımlarından dolayı teşekkürü bir borç bilirim.

Ayrıca her an ilgi ve desteğini üzerimden eksik etmeyen sevgili eşime en içten teşekkürlerimi sunarım.

Aralık 1993 Bekir Karlık

İÇİNDEKİLER :

ÖZET

Özellikle dirsek üstünden kolunu kaybedenler için kullanılan çok fonksiyonlu protezlerin kontrolu, bir çift yüzey elektrodundan alınan miyoelektrik işaretler kullanılarak yapılabilir. Kontrol stratejisi, herbiri için aynı olan ve tekrarlanabilen kas kasılma karakterleri setinin meydana getirdiği miyoelektrik işaretlerden kolaylıkla çıkarılan karakteristik parametrelerle olur. Bu parametreler kullanılarak farklı kas kasılma karaktarlerini sınıflara ayırmak mümkündür. Kas kasılması sınıflarının herbiri protez cihazının özel fonksiyonunu tetiklemek için kullanılır.

Yapay sinir ağı uygulaması miyoelektrik işaret analizlerini yerine getirmek için uygulanır. Bu araştırmanın amacı, çok açılı serbestiyete sahip kolun kontrolunu sağlayan daha çok güvenilir metodları incelemektir. Miyoelektrik işaretleri sınıflandırmak için üç katman içeren bir çok katmanlı perceptron özbağlanımlı (AR: Auto Regressive) model parametreleri ve işaret gücü özellikler olarak kullanılır. Bu özellikler kullanılarak, perceptron altı ayrı kol fonksiyonları arasındaki farkı ayırdetmek için eğitilir. Perceptron sınıflandırıcısı tarafından kullanılan iki-boyutlu karar sınırları tesbit edilir. Bu adaptif nitelik, perceptronların gelecekteki miyoelektrik işaret analizleri için faydalı bir araç sağlayabilir.

Bu çalışmada, farklı hareketler (dirsek açma, dirsek kapama, bilek bükme, bilek döndürme, kavrama ve dinlenme) gözönüne alınmıştır ve bu fonksiyonlar 3000 iterasyon için % 96.1'lik başarı oranında ayırdedilmiştir.

SUMMARY

The Control of multi functions prostheses can be accomplished using a myoelectric signal taken from a single pair of surface elektrodes. This has been demonstrated specifically in the case of prostheses for use by above elbow amputees. The control strategy in such situations is to have the user /subject generate a set of repeatable muscle contraction patterns, each having similar characteristic parameters that can be easily extracted from the myoelectric signal. By using these parameters, it is possible to segregate different muscle contraction patterns into classes. Each class of muscle contraction is used to trigger a particular function in the prosthetic device.

A Neural Network implementation is applied to myoelectric signal analysis tasks. The motivation behind this research is to explore more reliable methods of deriving control for multidegree of freedom arm prostheses. A Multi Layer Perceptrons (MLP) implementation involves using a three-layer perceptron for classifying myoelectric signals. Auto regressive (AR) model parameters and the signal power are used as features. Using these features, the perceptron is trained to distinguish between six separate arm functions. The two-dimensional decision boundaries used by the perceptron classifier are deliated. This adaptive quality suggests that perceptrons may provide a useful tool for future MES analysis.

In this study, different movements (elbow extension, elbow flexion, wrist supination, wrist pronation, grasp and relextion) are considered and these functions are discriminated with total success rate of 96.1% for 3000 iterations.

1.GİRİŞ

1.1. Konunun Tanıtılması

Çok fonksiyonlu protezlerin kontrolu bir çift yüzey elektrodundan sağlanan miyoelektrik işaretler kullanarak yapılabilir. Bu durum özellikle dirsek üstünden kolunu kaybetmiş vakalarda gösterilebilir. Benzer biçimde kontrol stratejisi, miyoelektrik işaretler kolayca elde edilebilen karakteristik parametreleri herbiri için aynı olan ve tekrarlanabilen kas kasılma karakterlerinin setini üretmektir. Bu parametreler kullanılarak farklı kas kasılması karakterlerini sınıflara ayırmak mümkündür. Kas kasılması sınıflarının her biri protez alette parça fonksiyonunu tetiklemek için kullanılır. Sınıflar arasında iyi bir ayrıma sahip olmak için, protez alet yerine takıldığındaki normal kol fonksiyonu ile hiçbir benzerlik göstermeyebilen olağan dışı kas kasılmalarının bir setini üretmek, denek için sık sık gerekebilir. Kullanıcının bu yeni kas kasılmalarını tekrar tekrar yapabilmeyi öğrendiği proses tipik olarak yavaştır ve sıklıkla güvenilir bir kontrol üretemez. Kullanıcı öğrenmesinin miktarını azaltmak için, protez kol kontrol fonksiyonunu harekete geçirmede kullanılan kasılmaların protez tarafından yapılmaya başlanan doğal fonksiyonuna benzemesi istenir. Pekçok geleneksel karakter tanıma kontrol sistemlerindeki bir başka kısıtlama da hesaplama ihtiyaçları büyük boyutta olma eğilimindedir. Bu pratik uygulamaları imkansızlaştıran kontrol mekanizmasında önemli gecikmelere yol açar.

Değişik kas kasılma karakterlerini ayırtedebilme kabiliyeti, miyoelektrik işarette tutulan bilgiye bağlıdır. Örnek olarak üç kasın yakınına yerleştirilmiş bir çift deri yüzey elektrodlarından elde edilen işaret, üç kasın herbirinin aktivitelerine ait bazı bilgiler içerecektir. Bu bilginin tanımlama problemi daha karmaşık olacaktır. Bu da miyoelektrik işaretin doğasından kaynaklanmaktadır. Yüzey elektrodu tarafından belirlenen miyoelektrik işaret, elektrod yakınında bulunan bütün motor birim işaretlerinin kabaca bir toplamıdır. Yüzey elektrodun tek bir kasın bütününden alındığı işaret tabiatta tamamen rastlantısal yapıdadır ve asenkron olarak tetiklenmiş motor birim darbeleri setinin toplamından oluşur. Tek kastan alınan ve diferansiyel olarak büyütülen miyoelektrik işaret sıfır ortalama değeri ve kas kasılma seviyesiyle orantılı değişiklik ile karakterize edilir.

Deri altındaki kas aktivitesi hakkında daha ileri bilgiler sağlayabilen bu işaretten başka özellikler çıkarmak mümkündür. Bu teknik bütün miyoelektrik işarete katkıda bulunan bir kastan daha fazlasının olduğu durumlara da uygulanabilir. Yüzey elektrodunun yakınlarında birkaç kas varsa; sonuç miyoelektrik işaret her kastaki aktiviteye ait bilgiler içerecektir. Mademki elektrot ile kas arasındaki doku yüzünden bir filtre etkisi vardır; bu bilgi kasların komşuluğuyla alakalı olarak yerleştirilen elektroda büyük ölçüde bağlı olan işarette nasıl gösterilecektir? Kas setinin içindeki kasılmalardan oluşan karakter, alınan işaretten çıkarılan özelliklere dayalı olarak tanımlanabilir.

Graupe tarafından gerçekleştirilen deneylere dayalı olarak (Graupe, 1975), bu tip kas hareketlerinin, miyoelektrik işaret özelliklerinin oluştuğu setlerden tanımlanabileceği söylenebilir. Normal kol fonksiyonlarının seti ile bileşik kas kasılma karakterleri arasında önemli bir fark varsa, normal kol fonksiyonlarının direkt tanımlanmasına bağlantılı olan miyoelektrik işaretin sınıflandırılması yapılabilir. Protez kontrol uygulamasında, farklı miyoelektrik işaretlerinin tanımlanması protez cihazlarının hareket kontrolünde kullanılabilir. Konvansiyonel kontrol metodlarına karşın, protez aleti tarafından gerçekleştirilen fonksiyonlar, miyoelektrik işaretleriyle tanınan doğal olanlara tamamen benzeyecektir. Bundan dolayı miyoelektrik işaretlerin sınıflaması protez cihazlarının kontrolünde çok kullanışlı bir metod olacaktır.

Yapay Sinir Ağları (YSA) sınıflayıcılarından olan çok katmanlı perceptron (idrak); miyoelektrik işaretlerinin iyi ayrımını yapabilmek için protez kullanıcısının ileri derecede öğrenmesini gerekli kılmaktadır. Yapısından dolayı YSA sınıflayıcıları uygulamada normal olarak gereken kullanıcı öğremesinin miktarını azaltmak için kullanılır. Bu metodla tek kullanıcı tarafından üretilen işaretlerin özel çeşitlerine, ağ kendini adapte edecektir. Uygulamada bütün amaç öğrenme işinin çoğunu kullanıcıdan makinaya devretmektir. YSA'lı sınıflayıcının diğer önemli özelliği; bilgi olasılık yayılmalarının esasını belirten hiçbir varsayım yapmayı gerektirmemesidir. Böyle varsayımlar miyoelektrik sınıflama uygulamasında sınırlanabilir. Bu ek hesaplama gücü, protez kontrol uygulamasında kabul edilebilir tanıma oranları gösterdiği için tercih edilir. Onun için bu çalışmada çok katmanlı perceptron, miyoelektrik işaret sınıflayıcısı olarak kullanılmıştır.

1.2. Miyoelektrik Kontrollü Protezlerin Gelişimi

İkinci dünya savaşından beri vücudun dış güçleri olan uzuvların (kol ve bacak gibi) hasara uğradığında tekrar kullanmaya yönelik yapay protez ve ortez kolları imal etmek için birçok çalışmalar yapıldı (Alderson, 1969). Bu çalışmalarda, vücudun zedelenmiş kaslarından gelen elektromiyografik (EMG) işaretlerinden yararlanıldı. Bunu yapanlar yapay uzuvları kontrol etmek için gerekli elektriksel kumandaları inceleyen araştırmacılardır (Graupe et al, 1978; Lyman et al, 1974; Saridis et al, 1977). Yapılan araştırmaların çoğu ana parçanın üzerinde özel çalışma veya efor gereken ya hep ya hiç biçimi tek hareket kontrolleri meydana getirdi. İlk miyoelektrik kontrollu protezler; kavrama/bırakma gibi tek bir fonksiyonu gerçekleştirmeyi amaçlamıştır. Bilgisayar bilgisi canlı protezlerinin daha detaylı işaret işleme ve hareket kontrolleriyle çalışmayı mümkün hale getirdi.

1940'larda miyoelektrik kontrollu bir yapay el gerçekleştiren Reiter, parmak sıkıcı (flexor) ve açıcı (extensor) kas gruplarından elde edilen EMG işaretlerini kullanmıştır (Scott et al, 1977). Bu işaretler doğrultulduktan sonra genlikleri karşılaştırılmış, hangisi daha büyükse onun temsil ettiği hareketin istendiğine karar verilmiştir. Buna göre parmak sıkıcı veya açıcı röleler çalıştırılarak elin kavrama yada

bırakma hareketini yapması sağlanmıştır. Aynı çalışma ilkesi günümüzdede tek fonksiyonlu yapay ellerde kullanılmaktadır. Ancak daha gelişmiş işaret işleme yöntemleriyle kavrama hız ve gücünü kapsayan oransal kontrol mümkün olmaktadır.

Dorcas ve Scott, birden fazla serbestlik derecesini kontrol edebilmek için bir noktadan ölçülen EMG işaretinin etkin değerini ayrık bölgelere bölme yöntemini geliştirmiştir (Jacobsen et al, 1982). Aynı kasın değişik düzeylerdeki kasılmaları farklı hareketlere karşı düşürülmüş EMG işareti hangi aralığa giriyorsa o aralığın belirttiği hareket gerçekleştirilmiştir.

Wirta ve arkadaşları, birçok noktadan ölçülen EMG işaretlerinden faydalanmıştır (Wirta et al, 1978). 10 tane elektrod, belli hareketlerin gerçekleştirilmesinde etkin olan kasların üzerine yerleştirilmiştir. Elektrodlarla edinilen EMG ölçümleri matris halinde dizilmiş ve bilgisayarda incelenmiştir. İnceleme sonucunda elde edilen sınıflama işlemleri daha sonra EMG ölçümlerinin sınanarak değişik hareket sınıflarına ayrılmasında kullanılmıştır. Çok elektrodlu miyoelektrik kontrolun farklı bireylere uyarlanması üzerinde çalışan Lyman, "öğrenme" ve "çalışma" olarak iki ayrı aşama tanımlamıştır (Lyman and Freedy, 1976). Öğrenme sırasında özürlü, sağlıklı koluyla hayali kolunu parelel olarak hareket ettirirken bilgi-işlemci EMG örüntüleri ile kol hareketleri arasındaki bağıntıyı saptar. Normal çalışma sırasında, özürlünün ürettiği EMG, eğitim süresinde elde edilenlerle kıyaslanıp en fazla benzeyen örüntü saptanır. Sonra o örüntüye ait hareketi amaçladığına karar verilir. (en yakın komşu algoritması) Yöntemin humerusu (pazı kemiği) döndürme, dirseği açma/kapama ve önkolu döndürme fonksiyonlarının kontrolunda başarılı olduğu bildirilmiştir. Aynı tekniği uyarlayan İsveçli araştırmacılar, "İsveç Eli" olarak bilinen çalışmalarında, kavrama/bırakma, önkolu döndürme ve bileği aşağı/yukarı bükme hareketlerini denetlemişlerdir (Almstrom et al, 1981). Ayrıca dizgede mikroişlemci kullanılarak protezin oransal kontrolu sağlanmıştır.

Graupe ise bir nesnenin (mesela kol) beraber çalışan EMG işaretlerinden gelen kontrol bilgisini tekrar elde etmek için bir zaman serisi tanımlama usulü teklif etti ve başarılı bir şekilde yerine getirdi (Graupe et al, 1982 and 1985). Bahsedilen EMG işaretleri omuz kaslarından elde edildi. Başka bir çalışmada ise dirsekten omuza kadar olan kemiği (pazı kemiği) kesilen veya felç olan bir şahsın biceps (kolun üst kısmındaki kaslar-iki başlı kas) ve triceps (üç başlı kas)'lerinden meydana gelen EMG işaretlerinin istatistiksel analizini yaptı (Saridis and Gotee, 1982). İstatistiksel işaret tanıma algoritmaları, kolun birleşik ilk hareketleri ve muhtemel işaretlerinin herbirine cevap veren işaretleri sınıflandırmak için kullanıldı. Bu çalışmada; kol kemiğine veya omuza ait iç/dış dönme, dirsek açma/kapama ve bileği aşağı/yukarı bükme hareketleri denetlenmiştir. El kavraması hareketi sınıflandırmaya katılmamıştır.

Özellikle Graupe'nin çalışmasından faydalanarak, Kelly, Parker ve Scott ilk olarak YSA (Hopfield algoritması) kullanarak miyoelektrik işareti sınıflamaya dayalı çok fonksiyonlu kontrol tasarımını gerçekleştirdiler (Kelly et al, 1990). Daha sonra bu ilk çalışmadan yararlanarak Hudgins,

Parker ve Scott tarafından yine YSA kullanılarak çok fonksiyonlu bir miyoelektrik kontrol çalışması geliştirildi (Hudgins et al, 1993). Son iki çalışma tek kanallı sistem olup, deney safhası aşamasındadır. Söz konusu çalışmalar; biceps ve triceps elektrodlardan alınan işaretlerin zaman serisi parametreleri çıkartılarak dört hareketin sınıflandırılması gerçeklenmiştir.

Yurdumuzda ise, 1989'da Ö. Kuyucu tarafından master tezi olarak bir çalışma yapılmıştır (Kuyucu, 1989). Burada, biceps ve triceps kaslarından yüzey elektrodları vasıtasıyle alınan EMG işaretlerinin zaman serisi parametreleri çıkartılarak, "Parelel Filtreleme" yöntemi ile sınıflandırılma yapılmıştır. Bu yöntemle dört hareket yaklaşık % 70 başarı ile tanımlanabilmektedir.

1.3. Mevcut Çalışmaların Eksiklikleri

Miyoelektrik sistemler, dirsekten yukarı kesik organlar veya benzeri protezlerde yaygın olarak kullanılmaktadır. Mevcut ticari amaçlı bütün sistemler, tek devre kontrolu için geçerlidir (Parker and Scott, 1986 and 1988). Bu sistemler; el , dirsek, parmak hareketlerinin EMG işaretlerinin genliklerine vaya değişik oranlarına dayalı işaret kontrolu üzerinedir (Dorcas, 1966; Childress, 1969). Kontrol işareti tek bir miyoelektrik kanalından çıkarılmamaktadır. İşaret genliğinin üç devre durumu vardır. Bunlar; genlik, aktivite ve hız şeklindedir. Bütün bu durumların başarısı bir tek kontrol devresi ile yapılmalıdır. Tek fonksiyonlu (Graupe ve diğerlerinin yaptıkları ilk çalışmalar) hareketler için başarılı sonuçlar alınmıştır. Fakat çok fonksiyonlu hareketlerin kontrolu için daha fazla bilgi gerektirdiğinden yetersiz kalınmaktadır. Herbir miyoelektrik işaretinden dolayı veya tek fonksiyonlu kontrolu, özel çok kanallı sistemden ayırmak için kontrol çıkış sayıları (yada fonksiyonları), kontrol girişleri veya kanallarından daha büyük olabilir. Beher kanalının fonksiyon sayısı bir seviyeye kadar veya sistem kod oranı ikiden fazlası için sınırlandırılır (Williams, 1990). Geri besleme durumu, kullanılan herbir kanalın durumlarının sayısını azaltmaya çalışacağından başarılı olmayabilir (Vodovnik et al, 1967). Daha fazla elektrod ihtiyacıyla genlik kodunun birkaç kanalı kullanılarak, çok fonksiyonlu protezler geliştirilebilir (Richard et al, 1983). Fakat yüksek seviyede kesiklerin kontrolu için gerçekleştirilmesi zordur. Eğer yüksek seviyede kesikler mevcut değilse, "Boston Eli" (Schmeild, 1973 and 1977) ve "Utah Kolu" (Jacobsen et al, 1982) gibi birkaç başarılı çalışmalar elde etmek mümkündür.

Dorcas ve Scott 'un yaptıkları ilk çalışmada kullandıkları yöntemin sakıncası; aşırı zihinsel çaba gerektirmesi ve denetlenebilecek hareket sayısı ile ulaşılacak başarının özürlünün kasını farklı ayırtedebilir ve yinelenebilir düzeylerde kasabilme yeteneğiyle sınırlıdır. Bu sakıncalar eğitimle kısmen ortadan kaldırılabilir. Bahsedilen bu sorunları çözmek üzere kimi araştırmacılar hastaların ameliyat sonrasında yaşadıkları "Hayalet kol" duygusundan yararlanmışlardır. Çoğu özürlü yitirdiği kolunu oynattığı zannıyla kalan kaslarında kolu yerindeyken oluşan kasılmaları meydana getirebilmektedir. Bu kasılmaların EMG işaretlerinde değişik hareketler için ayırtedilebilir farklı örüntüler oluşturmaktadır. Yapay kol kontrolunda bilgi kaynağı olarak bu işaretlerin kullanılmasıyla kullanıcı yönünden en az zihinsel çabayı gerektiren dizgeler geliştirilebilmektedir.

Bu noktaya kadar anlatılan yaklaşımlar, gerçekleştirilecek harekete karar verilmesinde EMG işaretlerinin güçlerinden yola çıkarak çeşitli kaslarda üretilmek istenen kuvveti kestirmektedirler. Sağlıklı bir insanın kol kasları ile protez motorları arasında kurulan ilişki içinde bu kestirimler belli motorların çalıştırılmasına yol açmaktadır. Sözgelimi dirseği içe bükücü kaslar kuvvetle kasılıyorsa protezin dirsek eklemindeki motor çalıştırılmaktadır. Basit yöntemler, ölçülen işaretlerin güçlerini kıyaslamakta, hangi kastan gelen işaret baskınsa o kasın kasılmasına yol açacağı hareketin istendiği sonucuna varmaktadır. Böylece herbir harekete bir kas karşılık gelmektedir. Diğer kaslardan gelen işaretlere "gürültü" gözüyle bakılmakta ve bunlar süzülmeye çalışılmaktadır. Wirta 'nın ki gibi daha ileri yöntemler ise, bir hareketin gerçekleştirilmesinde ana kasın yanında başka kaslarında kasıldığını kabul etmektedir. En güçlü işaretin seçilmesi yerine tüm elektrodlardaki işaretler gözönüne alınmakta, işaret gücünün uzamsal dağılımı incelenmektedir. Ancak yine kullanılan ayırıcı paremetre asıl işaret gücüdür ve tek paremetreyle en fazla üç hareket ayırt edilebilmektedir. Ayrıca, özellikle çok-fonksiyonlu hareket söz konusu olduğunda, çok sayıda elektrod bölgesi gerekmektedir. Bu ise, kolunun sadece küçük bir kısmı kalmış yada kas sinirleri büyük ölçüde zedelenmiş hastalarda zorluklara hatta imkansızlıklara yol açmaktadır. Yalnızca işaret gücünü kullanmanın işaretin tüm bilgi içeriğinden yararlanmamak olduğunu savunan kimi araştırmacılar, ileri işaret işleme algoritmalarının kullanılmasıyla tek bir noktadan (kanaldan) ölçülen EMG ile birden fazla hareketin tanınabileceğini ileri sürmüşlerdir.

Yaklaşımın öncülüğünü yapan Graupe'un yöntemi, değişik kas grupları kasılmalarının farklı dalga biçimleri üretmesine dayanmaktadır. Bir hareket gerçekleştirilirken belli kas grupları kasılacak ve dolayısıyla bunların karakteristik dalga biçimleri üretilecektir. Kas ve deri dokularının yerel tümleme etkisiylede yüzey elektrodları ile bir noktadan ölçülen işaret, yapılan harekete bağlı olarak bu dalga biçimlerinin katılaşımlarından oluşan belli örüntüler içerecektir. Kasların elektriksel etkinliğini özbağlanımlı bir süreç (autoregressive-AR) olarak modelleyen Graupe'nin geliştirdiği dizgede örüntü tanıma AR katsayıları uzayında gerçekleştirilmektedir. "Kalibrasyon" aşamasında özürlü, protezin gerçekleştireceği her hareketi hayelet koluyla yaparken ürettiği EMG işaretinin AR katsayıları hesaplanmakta ve her harekete ilişkin katsayı takımları bellekte saklanmaktadır. Çalışma sırasında ölçülen EMG nin katsayıları hesaplanarak bunların bellekteki katsayı takımlarından hangisine daha yakın olduğu bulunmakta ve bu takımın karşı düştüğü hareket gerçekleştirilmektedir. Bu yöntemle dirseği açma/kapama, önkolu döndürme ve kavrama/bırakma fonksiyonlarının özel eğitim görmemiş bir özürlüde 0.2 saniyede % 85 başarı ile ayırtedilebildiği açıklanmıştır. Graupe bir yerine iki EMG kanalının kullanılmasıyla başarı oranının arttırılabileceğini öne sürmektedir. Bununla birlikte bu yöntem protez kolu idare etmek için gerekli kontrol işaretlerini tekrar üretmede yeteri kadar hızlı değildir. Doerschuk ise yöntemi dört kanal içerecek biçimde geliştirmiştir. AR katsayılarının yanısıra, elektrodlar arasındaki bağlantılardan da yararlanılan bu çalışmada hareketlerin doğru belirlenme oranının arttırıldığı bildirilmektedir.

Kelly ve arkadaşlarının ilk defa YSA ile yapmış oldukları çalışmada; tek EMG kanalı kullanılmıştır. Test edilen dört kol fonksiyonu yaklaşık 2800 iterasyonla ayırt edilebilmektedir. Bu çalışmada program dili olarak Fortran, bilgisayar olarak da bir PC 80386 kullanılmıştır. En iyi tanıma ortalama % 90 olarak bulunmuştur. Daha sonra Hudgins, Parker ve Scott 1993 başlarında bir önceki çalışmayı biraz daha geliştirmişler ve en düşük % 70, en yüksek % 98 ve ortalama % 91.2'lik bir tanıma ile yine 4 hareket ayıredilmiştir. Bu çalışmada bir 8 MHz Intel 80286 PC kullanılmış olup , program dili Turbo C 'dir. Son iki çalışmadaki tek kanallı sistemler nispeten geniş hesaplama zorluğu gerektirmekte olup, henüz laboratuar çalışması aşamasındadır.

2. EMG'NİN OLUŞUMU VE ÖLÇÜLMESİ

EMG, kasın kasılması sonucu ortaya çıkan biyopotansiyel işaretlerdir. Bunların kaynağı vücutta meydana gelen çeşitli elektrokimyasal olaylardır. İstemli kas hareketleri beyinde sinirler yoluyla elektriksel uyarıların kasa iletilmesi sonucu ortaya çıkar. Kas liflerinin kasılmaları sinirlerce iletilen elektriksel uyarılar yoluyla gerçekleştiği gibi kasılmalarıda elektriksel bir işaret doğurur. Bu işaret iğne veya yüzey elektrodlarıyla ölçülür.

2.1 Uyarılabilen Hücrelerin Elektriksel Aktivitesi

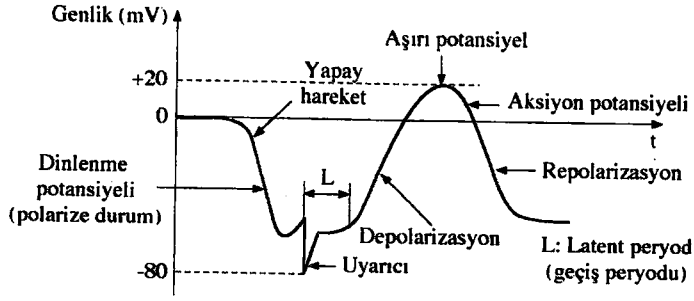
Uyarılabilen hücreler, sinir ve kas hücreleri olup, bunlardaki elektrokimyasal olaylar sonucunda bioelektrik potansiyeller meydana gelir. Bu hücreler de devamlı olarak bir "dinlenme" potansiyeli vardır. Uyarılınca "aksiyon" potansiyeli meydana gelir.

2.1.1 Dinlenme Durumu

Sukunette uyarılabilen hücrelerin içi ile dış ortam arasında bir elektrik potansiyel farkı vardır. Bu dinlenme gerilimi -50 ile -100 mV civarında bir genliğe sahiptir. Hücre zarı çok ince (7-15nm) ve vücut sıvısındaki iyonları geçirebilen bir yapıya sahiptir. Vücut sıvısı içinde sodyum (Na^+) potasyum (Cl^-) ve klor (Cl^-) iyonları mevcuttur. Hücre zarının potasyum iyonlarını geçirme kabiliyeti, sodyum iyonlarınınkinden 50-100 defa fazladır. Bu farkın nedeni henüz bilinmemekle beraber membrandaki (hücre zarı) gözeneklere bağlı olduğu düşünülmektedir. Potasyum iyonları gibi klor iyonlarıda hücre zarından daha kolay geçerler. Dinlenme durumunda vücut sıvısındaki K^+ ve Cl^- iyonlarının çoğu hücre içine girerek hücre içi negatif dışı pozitif olur. Bu duruma "polarize durum" denir.

2.1.2 Aksiyon Durumu

Uyarılabilen hücrelerin özelliklerinden biride, uyarıldığı zaman bir aksiyon potansiyeli üretmesidir. Hücre zarının eşik potansiyelinden daha düşük bir gerilim uygulandığı zaman şekil 2.1 de gösterildiği gibi hücre depolarize olarak belirli genlikle ve belirli süreli bir darbe oluşur. Sinir hücreleri için bu salınım 120 mV ve süresi 1 ms dir. Kas hücrelerinde aksiyon potansiyeli süresi 2-4 ms ve aksonlardaki yayılma hızıda 5 metre/s kadardır. Hücre uyarılınca, hücre zarının özelliği değişerek Na iyonları hücre içine girer ve az bir miktar K iyonu dışarıya çıkar. Böylece hücrenin içi pozitif, dışı negatif olur. Bu olaya "depolerizasyon" denir.



Şekil 2.1 Aksiyon potansiyelinin zamanla değişimi

Belirli bir tepe değere ulaştıktan sonra, yukarıdaki olayların tersi oluşarak tekrar dinlenme potansiyeli seviyesine düşülür. Bu olaya 'repolizasyon' denir. Aksiyon potansiyelinin depolizasyon ve repolizasyon süresi boyunca hücre tekrar uyarılamaz. Hücrenin tekrar uyarılamadığı bu süre 1-3 ms kadardır (Pastacı, 1989).

2.2 Elektromiyogram (EMG)

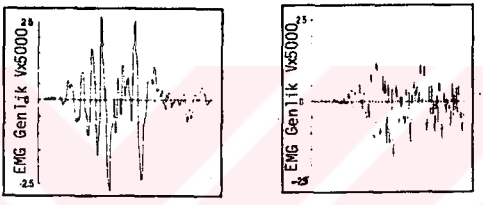
Yukarıda uyarılabilen tek bir hücrenin elektriksel davranışı anlatıldı. Oysa, kas liflerinin uyarılması tek tek değildir. Değişen sayıdaki kas lifi grupları bir motor siniri tarafından uyarılır. Bir motor siniri tarafından uyarılan kas lifi sayısı birden fazla olduğundan uyarılabilecek en küçük kas lifi gurubuda bir motor sinire bağlı olanlardır. Bu nedenle her zaman ancak bir grup kas lifinin uyarılması ve bu gruba bağlı liflerin elektriksel davranışlarının toplamının incelenmesi gerekmektedir. Uyarıya katılan motor birim sayısı ise kasdan istenen güce bağlıdır. Dolayısıyla kasın yapması gereken iş oldukça çok olmaktadır. Normal şartlarda kaslara verilen uyarıların kaynağı merkezi sinir sistemidir. Sağlıklı iskelet kasları motor sinirlerden bir emir gelmediği sürece kendiliğinden kasılmaz (Karaağaç, 1989). Motor hücreleri sayesinde bütün kas lifleri senkron bir şekilde çalışır. Motor hücreleri ile uyarılan kas liflerinin oluşturduğu alan potansiyeli süresi 3-15 ms ve genliği 20-2000 mV değerinde darbeler şeklindedir. Frekansı ise 6-30 Hz civarındadır.

EMG işaretleri iğne veya yüzey elektrodlarıyla vücuttan alınabilir. Kas faaliyetlerinin elektriksel davranışını incelemek üzere mikro elektrodlarla hücreye batırılarak elde edilen aksiyon potansiyelleri genellikle derinlerdeki kasların veya tek bir motor birimin incelenmesinde kullanılır. İnsan dokusunda aksiyon potansiyelinin genliği kas lifinin çapına kayıt yapılan bölgedeki aktif kas liflerine, elektrodların yerleştirilmesine ve elektrodların filtreleme özelliklerine bağlıdır (Korürek, 1991).

Yüzey elektrodlarıyla ölçüm yapılması halinde elde edilen işaret, o bölgedeki aktif liflerin ürettiği aksiyon potansiyellerinin toplamının oluşturduğu işarettir. Bu durumda toplama, bir çok lif, daha doğrusu bir çok motor birime ait aksiyon potansiyelleri katılacaktır. Çok miktarda motor birim

aksiyon potansiyellerinin karışmasından dolayı bu olay "girişim olayı" (interference pettern) olarak bilinir. Kasılma şiddeti ile aktif motor birim sayısı artar. Aynı anda birçok kas lifinin etkinleşmesiyle, her birinin ürettiği işaretler birbirini yok edebilir, veya toplanabilir. Sonuç olarak EMG'nin zaman içerisindeki görüntüsü rastgele değişen bir gürültüye benzerdir. Şekil 2.2 de, "dirsek kapama" ve "dirsek açma" hareketleri yapılırken elde edilen EMG dalga şekilleri gösterilmiştir.

EMG sıfır ortalamalı Guassıan bir süreç olarak incelenebilir. Frekans bandı 20-1000 Hz. arasındadır. Genliği iğne elektrodlar kullanıldığı zaman 10mV, yüzey elektrodları kullanıldığında 1mV civarındadır. Kas yorulmasıyla birlikte EMG'nin güç spektrumunda alçak frekanslara doğru kayma genliğinde ise artış olmaktadır.

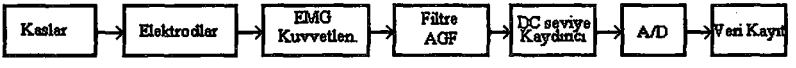


(a) (b)

Şekil 2.2 EMG dalga şekilleri (a) Dirsek kapama (b) Dirsek açma

2.3 Miyoelektrik İşaret Bilgisinin Ölçülerek Elde Edilmesi

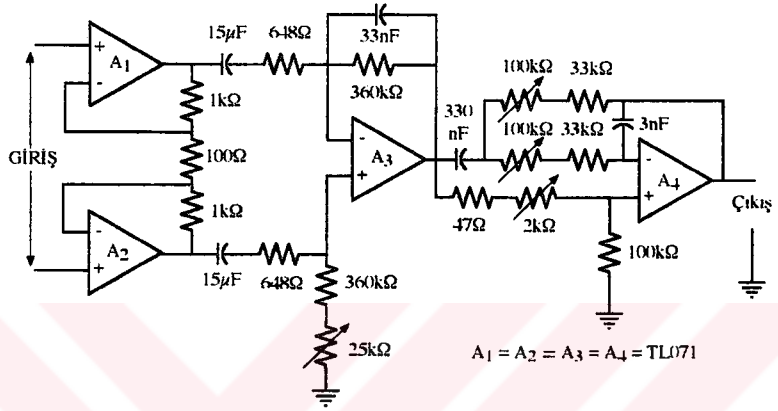
EMG işaretinin (incelenmesi için) ölçülerek elde edilmesinin blok diyagramı Şekil 2.3 de verilmiştir.



Şekil 2.3 EMG işaret bilgisinin elde edilmesi

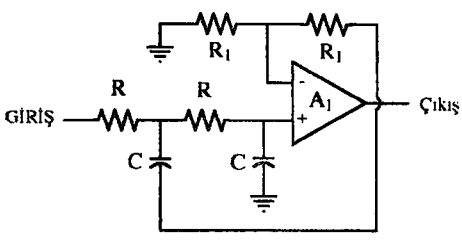
EMG işaretlerinin kaslardan alınmasında elektrodlar kullanılır. Bu çalışmada gümüş-gümüşklorür elektrod çifti kullanılır. Elektrod bağlanacak yüzey asetonlu pamukla silinerek deri üzerindeki iletkenli önleyecek vb. maddeler uzaklaştırılır. Yüzey temizlendikten sonra elektrod çifti iletken pasta sürülerek, ikisi arasında 3 cm aralık kalacak şekilde bicepslerin en çok karın yaptığı bölgeye yerleştirilir. Referans elektrod en çok hareketsiz duran diğer bileğe bağlanır. EMG kuvvetlendiricisi olarak standart ölçü amplisi kullanılır. Kuvvetlendiricinin devresi şekil 2-4 de verilmiştir. Bu devrede A1, A2 ölçü amplisi

olarak. A3 fark kuvvetlendiricisi olarak ve A4 dc 50 Hz şebeke frekansını bastırmak için kullanılan bir çentik olarak çalışmaktadır (Williams, 1990).

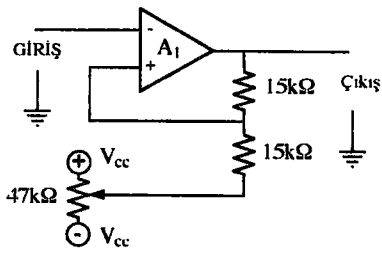


Şekil 2.4 EMG Kuvvetlendiricisi

Kuvvetlendiricinin çıkışında 2. dereceden ve üst kesim frekansı 14 Hz olan bir alçak geçiren Butterworth bir filtre kullanılır. Devre şeması şekil 2.5 'da gösterildiği gibidir. Kullanılan analog-dijital çevirici. girişine uygulanan 0 ile 5 volt değerleri arasındaki işaretleri çevirmektedir. EMG değerleri sıfır ortalama değerli olduğundan sözü edilen gerilim değerleri arasına ötelendikten sonra ADC girişine uygulanır. Bu amaçla kullanılan "dc seviye kaydırıcısının" devre şeması Şekil 2.6'de verilmiştir.



Şekil 2.5 2. Dereceden Butterworth AGF



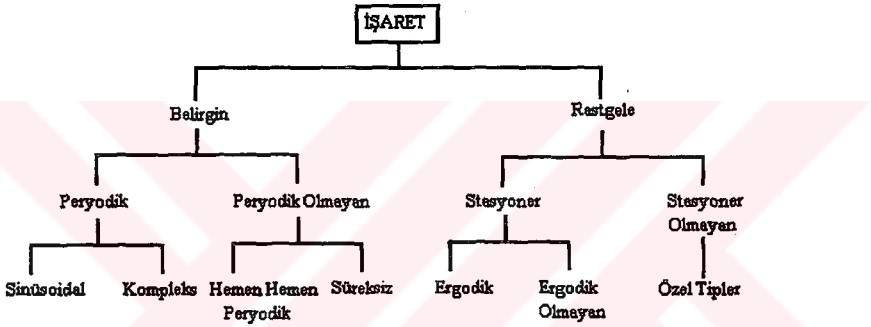
Şekil 2.6 DC seviye kaydırıcı

ADC devresi bir modül halinde DC içine yerleştirilmiş olup 12 bitliktir. ADC çıkışından elde edilen sayısal veriler kayıt edilerek işlenir.

3. BİYOLOJİK İŞARET İŞLEME

3.1 Temel Kavramlar ve İşaretlerin Sınıflandırılması

Biyolojik ve fiziksel sistemlerden elde edilen işaretlerin çeşitli özellik ve karakteristikleri vardır. Uygun işaret işleme yöntemlerinin uygulanabilmesi için işaretin genel karakteristiğinin bilinmesi önem kazanır. Şekil 3.1'de sınıflandırılan işaretler belirgin (deterministlik) ve rastgele(random) işaretler olmak üzere iki ana grup altında toplanır (Kohen, 1986).



Şekil 3.1 İşaretlerin Sınıflandırılması

Belirgin işaretler, açık ve bilinen matematiksel bağıntılarla ifade edilen işaretlerdir. Bu eşitlikler zaman yada frekansın birer fonksiyonu şeklindedir ve sonlu sayıda terim bulundurur. Rastgele işaretler ise tam olarak matematiksel bağıntılarla ifade edilen bir işaret bulmak zordur. Öte yandan rastgele işaretde bulmak mümkün değildir. EKG işaretine bakıldığında ve bunun QRS kompleksi ile ilgilenildiği işaretin bu QRS dilimi belirgin olacak ve fakat RR aralığı değişimi ile ilgilenildiğinde ise bu değişim rastgele olacaktır.

Belirgin işaretler iki alt gruba ayrılır; peryodik ve peryodik olmayan işaretler. Peryodik işaretler her t için $X(t)=X(t+T)$ bağıntısını sağlayan işaretlerdir. Burada T peryod adını alır. Peryodik işaretler, bir peryodluk kısımlarının tüm işareti belirlemesi nedeniyle işaret işleme ve işareti tanımlama açılarından kolaylık sağlar. Frekans domeninde ise, peryodik işaretler Fourier serileri ile temsil edilirler. Bu domende temel frekans ve harmonik bileşenler sözkonusu olur. Peryodik olmayan işaretlerde; hemen hemen peryodik(almost) süreksiz(transient) işaretler olmak üzere iki kısıma ayrılır. Hemen hemen peryodik işaretler birbirinden bağımsız. ve peryotları farklı (temel frekansları farklı) işaretlerin

kombinasyonu sonucu meydana gelirler. Süreksiz işaretler ise zaman içinde kısa bir süre ortaya çıkan ve peryodik olmayan işaretlerdir.

Rastgele işaretler işlenmesi daha zor olan işaretlerdir. Bir rastgele işaret bir rastgele işlemin, bir örnek fonksiyonudur. Bu örnek fonksiyonların oluşturduğu işaret "topluluk" (ensamble) adını alır. Rastgele işaret bileşik olasılık yoğunluk fonksiyonu ile belirlenir. Rastgele işaretler; "durağan" (stasyoner) ve "durağan olmayan" işaretler olmak üzere iki alt gruba ayrılır. Durağan işlemin, istatistiksel özellikleri zamanla değişmez. Bu gruba giren önemli bir rastgele işaret, "ergodik" işaret adını alır ve bu işaret için herhangi bir t anında topluluk (ensamble) elemanları üzerinde alınan topluluk ortalaması, elemanlardan birinin (örnek fonksiyonlardan birinin) üzerindeki zaman ortalamasına eşittir. Durağan olmayan işlem en zor işlenen işlemdir. Bu yüzden hatalı olduğunu bile bile bu işlem, "ergodik işlem" varsayılır. Mesela EEG işaretini işlerken elimizde tüm topluluk olmayıp sadece bir örnek fonksiyon bulunabilir. Bundan dolayı ergodiklik varsayım yapılarak gerekli istatistiksel özellikler, bu örnek fonksiyonun zaman içindeki değişiminden elde edilme yoluna gidilir. Durağan olmayan ışaretler için işlem yöntemleri pek etkili olmadığından durağan olmayan (non stasyoner) işaret, her biri durağan sayılan dilimlere (segmentlere) ayrılır. Dilimlerin uzunluğu durağan olmamanın özelliklerine bağlıdır. Örneğin ses işaretleri için dilim uzunluğu 10 ms iken EEG işaretleri için birkaç saniyedir.

İşaret işleme açısından işaretlerin diğer bir sınıflamasıda vardırki bu sınıflamada işaretler; "sürekli" ve "ayrık" işaretler olarak iki gruba ayrılır. Genel olarak sürekli işaretler zaman içinde herhangi bir anda tanımlanabilirler. Bu işaretlere uygulanan işaret işleme yöntemleri; Fourier ve Laplace dönüşümleri ve diğer "analog" yöntemlerdir. Donanım açısından bu işaretler; analog sistemlere (filtreler, kuvvetlendiriciler, analog bilgisayarlar) uygulanılırlar veya bu sistemlerde işlenirler. Ayrık işaretler ise ancak belli zaman noktalarında (belli anlarda) tanımlıdır. Genellikle bu işaretler genlik olarakda örneklenirler. Bu yüzden ayrık işaretler, sürekli işaretlerin zaman içinde "örneklenmiş" ve genlik olarak "kuantalanmış" şekilleridir. Bununla beraber tabiatı itibarıyla ayrık olan işaretlerde mevcuttur. Bu işaretlere, z-dönüşümü ve ayrık fourier dönüşümü (DFT) gibi ayrık işaret işleme yöntemleri uygulanır. Donanım açısın ise ayrık işaretler sayısal bilgileri ihtiva eden sayısal sistemlerde işlenirler. Sayısal işaret işleme teknikleri günümüzde gelişme göstermektedir. Bu nedenle sürekli (analog) işaretler, analog-dijital çeviriciler (ADC) yardımıyla, sayısal (dijital) işaret işleme teknikleriyle uygulanabileceği şekle, yani sayısal şekle getirilirler. Bu şekilde sürekli işaretlerle ayrık işaret durumuna getirilmiş olur.

3.2 İşaret İşlemenin Temelleri

İşaret işlemenin en temel öğesi filtreleme işlemidir. Filtreleme zaman domeninden çok frekans domeninde uygulanır. Fourier dönüşümü, zaman domenindeki bir işareti, frekans domenine dönüştüren

bir işlemdir. Frekans domeninde işaret, genliği ve fazı frekansına göre değişen olarak tanımlanır. Laplace dönüşümünde ise karmaşık s-düzlemi kullanılır. İşaretin istenen frekans bölgelerindeki kısımları tamamen atılmak veya zayıflatılmak üzere dağılım (spectral) özellikleri filtreler yardımıyla biçimlendirilir. Benzer işlemler karmaşık z-domeninde tanımlanan ayrık işaretlere de uygulanır.

Frekans filtreleme işlemi, belirgin işaretler için olduğu kadar, rastgele işaretler içinde etkilidir. Rastgele işaretleri işlerken Fourier dönüşümü, işaretin örnek fonksiyonu yerine otokorelasyon fonksiyonuna uygulanır. Bu durumda işaretin güç dağılımı yoğunluk fonksiyonu ile ilgilenir.

3.3 Rastgele İşaret Modelleri

Burada "modeller" terimi, ilgilenilen rastgele sürece ilgili verinin üretilmesinde etkili olan kuralları açıklayan veya tanımlayan hipotezler için kullanılmıştır. Bu nedenle modeller rastgele işaretin özelliklerini ve tabiatını karakterize etmektedir. Sözkonusu modellerin biyolojik işaret işlemede önemli bir rolü vardır. Çünkü işareti sentez etmek için modele ihtiyaç vardır. Ayrıca işarete ait model parametreleri kullanılarak yeniden elde etme imkanı sağlar (Orfanidis, 1988).

3.3.1 Wold Teoremi

1938 yılında Wold, herhangi bir rastgele işaretin "genel bir lineer işaret" ile "belirgin işarete" ayrıştırılabileceğini göstermiştir. Daha açık bir şekilde Wold teoremi, matematiksel olarak aşağıdaki gibi verilir.

Herhangi bir durağan rastgele işaret $\{y_n\}$ şu şekilde ifade edilebilir.

$$y_n = u_n + s_n \tag{3.1}$$

burada {U(n)} ve {S(n)} korelasyonsuz işaretlerdir. {U(n)} genel bir lineer olup;

$$u(n) = \sum_{k=0}^{\infty} b_{n-k} e_k \tag{3.2}$$

biçiminde ifade edilir

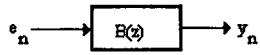
$b_0=1$ ve $\{ e_n \}$ işareti, $\{s_n\}$ ile korelasyonsuz bir beyaz gürültü işaretidir.

$$E[e_n s_k] = 0 \quad \text{..... tüm } n,k \text{ için} \tag{3.3}$$

{ Sn } belirgin işarettir ve sıfır hata varyansı ile geçmişinden tahmin edilebilir. Bu teoremden hareketle bir rastgele işaret "nedensel" ve "kararlı" olan B(z) lineer filtresinin bir durağan beyaz gürültü işaretinin $\{e_0,e_1,.....,e_n,.....\}$ çıkışı olarak ele alınabilir. Şekil 3.2'deki filtrenin transfer fonksiyonu;

$$B(z) = \sum_{n=0}^{\infty} b_n z^{-1} \tag{3.4}$$

olarak verilir.



Şekil 3.2 Basit Bir Filtre Modeli

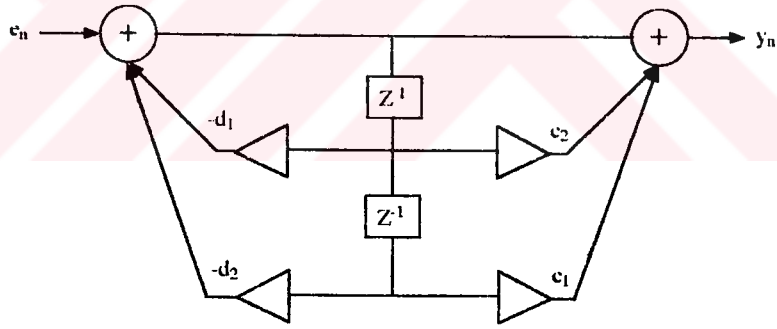
Çıkışta y_n rastgele işaretini, filtrenin impuls cevabı olan bn katsayılarıyla giriş dizisi e_n'in konvolüsyonu olarak elde edebiliriz.

$$y_n = \sum_{i=0}^{n} b_{n-i} e_i \ , \qquad n = 0,1,2,\ldots, \tag{3.5}$$

Yukarıda açıklanan Wold teoremine dayanan (3.4)'deki modelin pratikte kullanılması, sonsuz sayıda parametre gerektirmesi nedeniyle mümkün değildir. B(z) filtrenin sonsuz terimli bir filtre yerine Z değişkeninin iki sonlu polinom oranı biçiminde olması istenir. Böylece (3.5)'de verilen filtrenin giriş-çıkış ilişkisi olan konvolisyon toplamını bir fark denklemi ile ifade edebiliriz. Mesela, B(z) filtresinin transfer fonksiyonu;

$B(z) = \dfrac{1 + c_1 z^{-1} + c_2 z^{-2}}{1 + d_1 z^{-1} + d_2 z^{-2}}$ biçiminde verilsin, (5)'deki konvolisyon toplamı;

$y_n = -d_1 y_{n-1} - d_2 y_{n-2} + e_n + c_1 e_{n-1} + c_2 e_{n-2}$ fark denklemine eşittir. Şekil 3.3 de bu fark denkleminin gerçekleştirilmesi görülmektedir.



Şekil 3.3 B(z) Filtresinin Gerçekleştirilmesi

3.3.2 Doğrusal (lineer) Zaman Serileri Modellemesi

Çoğu sistem geçmişteki çıkışlarının, o andaki ve geçmişdeki girişlerinin (rekürsif) lineer konbinasyonları ile modellenebilmektedir.

$$S_k = -\sum_{i=1}^{p} a_i s_{k-i} + \sum_{j=0}^{q} b_j u_{k-j} \qquad (3.6)$$

Burada u_k ve s_k sistemin ayrık zamandaki giriş ve çıkış serileridir. Girişe ait herhangi bir bilgiye erişemiyorsak, EMG işaretlerinde olduğu gibi, giriş beyaz gürültü olarak alınabilmektedir. Yukarıdaki eşitlikten anlaşılacağı gibi model parametreleri olan a_{ij}; i=1,2,...p ve b_{ij} j=1,2,....,q bilindiği takdirde, geçmişteki giriş ve çıkış değerlerinden k anındaki s_k değeri hesaplanabilir. Bu nedenle modelleme "lineer tahmin" (linear prediction) olarak da adlandırılır. (3.6) nolu eşitliğin Z-dönüşümü alınarak sistemin transfer fonksiyonu:

$$H(z) = \frac{S(z)}{U(z)} = \frac{\sum_{j=0}^{q} b_j z^j}{1 + \sum_{i=0}^{p} a_i z^i} \qquad (3.7)$$

olarak bulunur.

Burada S(z) ve U(z) sırasıyla s_k ve u_k nın Z dönüşümleridir.

$$\sum_{i=0}^{p} a_i z^{-1} = A(z^{-1}) \qquad (3.8)$$

$$\sum_{j=0}^{q} b_j z^{-1} = B(z^{-1}) \qquad (3.9)$$

bu durumda; $A(z^{-1})S(z) = B(z^{-1})U(z)$ olup düzenlersek;

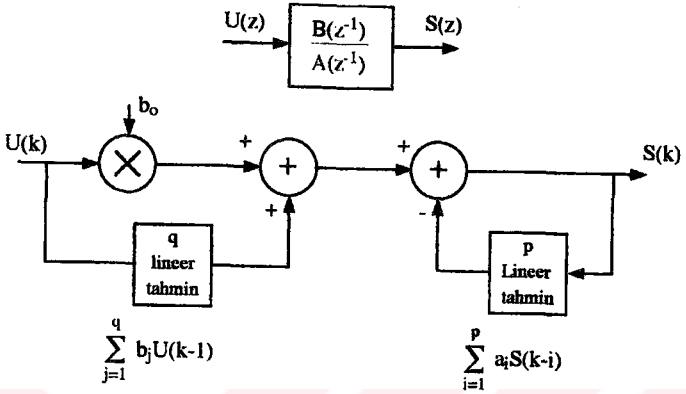
$$S(z) = \frac{B(z^{-1})}{A(z^{-1})} U(z) \qquad (3.10)$$

elde edilir. (Harici gürültü yok) Sıfırları ve kutupları ihtiva eden bu genel model ARMA (autoregressive moving average) model olarak isimlendirilir. Şekil 3.4'de ARMA modelinin Z ve zaman domenleri tasvir edilmiştir. Sistem sadece giriş işaretleri kullanılarak da modellenebilir. (a_i=0, i=1,2.....p) Bu durumda;

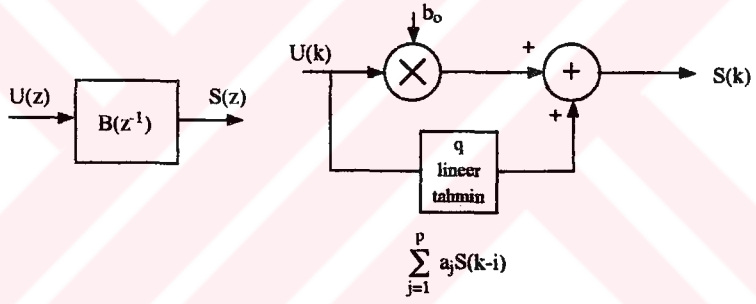
$$S_k = \sum_{j=0}^{q} b_j u_{k-j} \qquad (3.11)$$

$A(z^{-1}) = 1$ (3.12) olup, $S(z) = B(z^{-1}).U(z)$ (3.13)

şeklindedir. Bu model tüm sıfır veya MA (moving average) model olarak bilinir.



Şekil 3.4 ARMA modeli



Şekil 3.5 MA modeli

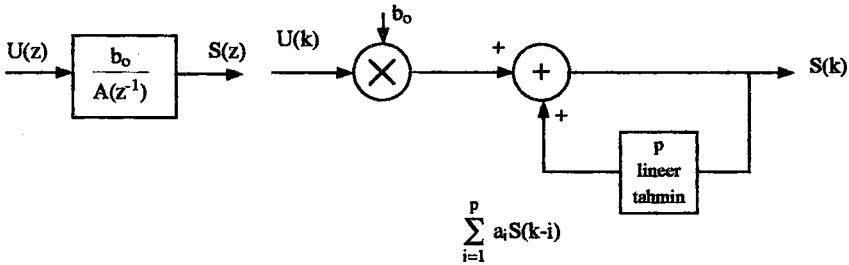
b_j katsayılarının sıfır olduğu durumda sistem, yalnızca çıkışları kullanılarak modellenir.

$B(z^{-1}) = b_0 = \text{sbt}$ (3.14) olup, $S(z) = \frac{b_0}{A(z^{-1})} U(z)$ (3.15)

bulunur. Tüm kutup veya AR (autoregressive) model olarak adlandırılan bu modelin fark denklemleriyle gösterimi;

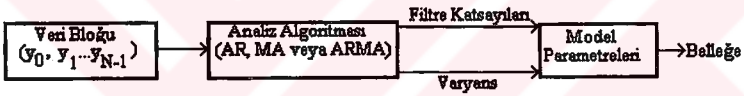
$$S_k = -\sum_{i=1}^{p} a_i s_{k-i} + b_0 u_k \quad (3.16)$$

şeklindedir. Şekil 3.6 da AR modeli tasvir edilmiştir (Kohen, 1986).



Şekil 3.6 AR modeli

Bu modeller özellikle, konuşma ve görüntü işlemede, jeofiziksel işaretleri değerlendirmede, EMG ve EEG işaretlerini işlemede, spektrum kestiriminde, veri (data) sıkıştırmada yaygın olarak kullanılmaktadır. Parametreleri bulmaya yarayan analiz sistemi şekil 3.7'de gösterilmektedir.



Şekil 3.7 İşaret analizi

En çok kullanılan model AR modelidir. AR parametre kestirimi için etkin algoritmalar geliştirilmiştir. Durağan (stasyoner) zaman serileri AR modeliyle modellenebilir. Tezde kullanılan EMG işareti her ne kadar durağan değilsede; yeterince küçük zaman aralıklarında durağan sayılabilmektedir (3). ARMA modeli ile durağan zaman serilerinin lineer bir modeli daha az sayıda parametre ile elde edilebilmektedir. Ancak ARMA parametrelerini elde etmek, AR modele göre daha karmaşıktır. MA modeli, durağan olmayan sistemlerin modellenmesinde daha uygundur. Bu çalışmada EMG işaretinin modellenmesinde AR modeli kullanılmıştır.

3.4 AR Katsayılarının Kestirimi

Bir sistem veya işaret modelinin parametrelerini elde etmek, diğer bir değişle parametre kestirimi için değişik yöntemler vardır. Bu yöntemlerden çoğu "en küçük kareler" (Least Squares, LS) yaklaşımından yola çıkan probleme değişik yollardan çözüm getiren algoritmalardır. Dört temel kestirim kriteri vardır. Bunlar:

1-Maksimum sonsal olasılık (Maximum a posteriori-MAP) kestirim kriteri

2-Karesel ortalama (Mean squared-MS) kestirim kriteri

3-Lineer karesel ortalama (Linear mean squared-LMS) kriteri

(veya en küçük kareler (Least squares, LS))

4-Maksimum olabilirlik (Maximum likelihood-ML) kriteri

En küçük kareler tekniğinde, kestirim hatasının karesinin, beklenen değeri en küçük yapacak şekilde sistem parametreleri belirlenir. LMS (LS) işlemesinin en önemli avantajı, sadece ikinci derece istatistiksel bilgiyi gerektirmesidir. Oysa diğer lineer olmayan kestirimciler daha detaylı olasılık yoğunluk bilgisi istemektedir.

3.4.1 Özilişki Korelasyon Yöntemi

Bir sistemin AR modeli (3.16) nolu formülden;

$$S_k = -\sum_{i=1}^{p} a_i s_{k-i} + e_k \qquad k=1,2,3,\ldots,N \tag{3.17}$$

olarak ifade edilir. Burada S_k işaretin k anındaki değeri p; modelin derecesi a_i; AR katsayıları N; örnek sayısı ve e_k ise beyaz gürültüdür. Geçmişteki çıkış değerlerini kullanarak k anındaki örneği;

$$\hat{s}_k = -\sum_{i=1}^{p} \hat{a}_i \, s_{k-i} \tag{3.18}$$

olarak kestirebiliriz. Bu kestirimden doğan hata;

$$e_k = S_k - \hat{s}_k = S_k + \sum_{i=1}^{p} \hat{a}_i \, s_{k-i} \tag{3.19}$$

olacaktır. Aynı zamanda ek erişilmeyen girişin kestirimidir. Hatanın en küçük olabilmesi için,

$$\frac{\partial \, E\left(e_k^2\right)}{\partial \, \hat{a}_i} = 0 \qquad i=1,2,\ldots,p \tag{3.20}$$

olmalıdır.

$$E\left(e_k^2\right) = E\left\{\left(s_k + \sum_{i=1}^{p} \hat{a}_i \, s_{k-i}\right)^2\right\} \tag{3.21}$$

(3.21)'i (3.20)'de yerine koyarsak;

$$\frac{\partial \, E\left(e_k^2\right)}{\partial \, \hat{a}_i} = 2E\left\{\left(s_k + \sum_{j=1}^{p} \hat{a}_j \, s_{k-j}\right) s_{k-i}\right\} = 0 \tag{3.22}$$

$$r_{ij} = \cong E\left(s_{k-j} \cdot s_{k-i}\right) = r_{j-i} \tag{3.23}$$

(3.22) ve (3.23) nolu eşitsizlikleri kullanarak i= 1,2.....p için p- bilinmeyenli p tane lineer denklem takımı elde edilir.

$$\sum_{j=1}^{p} \hat{a}_j \, r_{i-j} = -r_i \qquad i=1,2,....\hat{p} \tag{3.24}$$

Burada $r_0, r_1, ...r_p$ gibi (p+1) adet özilişki bilindiği taktirde p tane en iyi parametre (a_j, j=1,2.....p) bulunabilir. Bu eşitlik "normal" veya "Yule-Walker" eşitlikleri olarak adlandırılır. Bu eşitlikleri matris formunda göstermek mümkündür.

$$\begin{bmatrix} r_0 & r_1 & r_2 & \cdot & r_{p-1} \\ r_1 & r_0 & r_1 & \cdot & r_{p-2} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ r_{p-2} & \cdot & \cdot & r_0 & r_1 \\ r_{p-1} & r_{p-2} & \cdot & r_1 & r_0 \end{bmatrix} \begin{bmatrix} \hat{a}_1 \\ \hat{a}_2 \\ \cdot \\ \cdot \\ \hat{a}_p \end{bmatrix} = -\begin{bmatrix} r_1 \\ r_2 \\ \cdot \\ \cdot \\ r_p \end{bmatrix} \quad \text{veya} \quad R.\underline{\hat{a}} = -\underline{r} \tag{3.25}$$

Burada $\underline{\hat{a}}$; AR katsayısı vektörü, r özilişki vektörü, R simetrik ve köşegenler üzerindeki elemanları aynı olan
(Toeplitz) yapıdaki özilişki matrisidir. R matrisinin tersi alınarak;

$$\underline{\hat{a}} = -R^{-1}.\underline{r} \tag{3.26}$$

AR katsayısı vektörü bulunabilir.

3.4.2 PARCOR Yöntemi

Yukarıdaki yöntemde AR katsayılarını belirlemek için özilişki matrisinin tersini almak gerekmektedir. Bu işlemden kurtulmak için Levinson ve Durbin tarafından bir yöntem geliştirilmiş kısmi korelasyon yada PARCOR adı verilen bu yöntem, p bilinmeyenli problemi p tane bir bilinmeyenli eşitliğe indirger. (22) nolu denklemi açarsak;

$$E\{e_k^2\} = E\left\{\left(s_k + \sum_{i=1}^{\hat{p}} \hat{a}_i \, s_{k-i}\right) s_k\right\} + E\left\{\left(s_k + \sum_{i=1}^{\hat{p}} \hat{a}_i \, s_{k-i}\right) \sum_{i=1}^{p} a_i s_{k-i}\right\} = \min$$

$(r_0, r_1, ...r_p)$ gbi p+1 adet özilişki (korelasyon) (r_j; j = 0,1,2..p) bilindiği takdirde p adet optimal parametre

$(\hat{a}_j; j=1,2,\ldots,p)$ bulunabilir. O zaman en küçük ortalama hata;

$$E(e_k^2) = E_p = r_0 + \sum_{j=1}^{p} a_j\, r_j \tag{3.27}$$

şeklindedir. Normal olarak özilişkiler verilmez ve bunların verilen { s_k } k = 0,1,2,....,(N-1) dizisinden kestirilmesi gerekir. Bu durumda özilişkiler;

$$E\{s_k . s_{k+l}\} = \hat{r}_l = \frac{1}{N} \sum_{k=0}^{N-1-l} s_k . s_{k+l} \tag{3.28}$$

olarak hesaplanır. (3.27) nolu denklemden faydalanılarak;

$$p = 0 \quad \text{için} \qquad E_0 = r_0 \tag{3.29}$$

$$k_i = -\frac{\left(r_i + \sum_{j=1}^{i-1} a_j^{(i-1)} . r_{i-j} \right)}{E_{i-1}} \tag{3.30}$$

$$a_i^i = k_i\,, \qquad i = 1,2,\ldots,p \tag{3.31}$$

$$a_j^i = a_j^{(i-1)} + k_i . a_{i-j}^{(i-1)}\,, \quad j = 1,2,\ldots,(i-1) \tag{3.32}$$

$$E_i = (1-k_i^2) . E_{i-1} \tag{3.33}$$

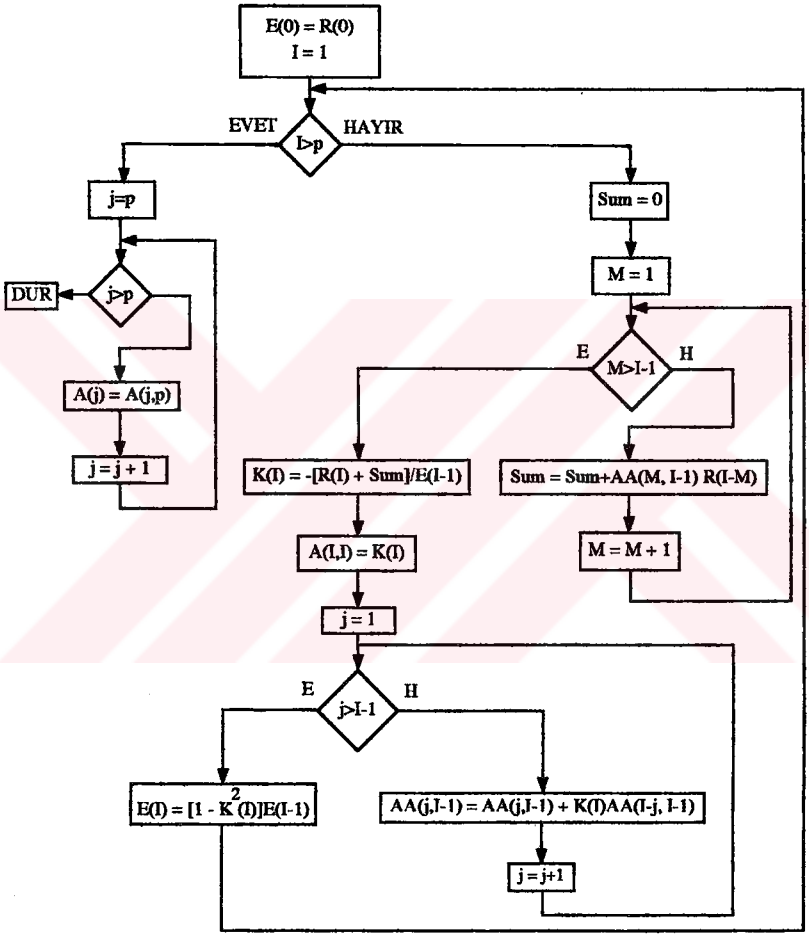
elde edilir. Burada a_j^i birinci dereceden modelin j'inci AR katsayısıdır. (3.30) ve (3.33) eşitlikleri i=1,2.....p için ardışık olarak çözülür ve sonuçta p'inci dereceden modelin AR katsayıları;

$$a_i = a_i^p \qquad i = 1,2,\ldots,p \tag{3.34}$$

olarak bulunur. (3.30) denkleminden hesaplanan k_i katsayıları "yansıma" veya "kısmi özilişki" katsayıları adını alır. Durbins algoritmasının j'inci dereceden en küçük ortalama hatası E_j;

$$0 \le E_j \le E_{j-1} \tag{3.35}$$

şeklindedir. (3.33) nolu eşitliğin hesabını gösteren akış diyağramı aşağıda Şekil 3.8'de verilmiştir. (Tezde kullanılan AR kestiri programı buna göre olmuştur)



Şekil 3.8 Durbins algoritmasının akış diyagramı (PARCOR için)

4. YAPAY SİNİR AĞLARI

4.1 Giriş

Yapay sinir ağları ya da kısaca YSA; insan beyninin çalışma sisteminin yapay olarak benzetimi çabalarının bir sonucu olarak ortaya çıkmıştır. En genel anlamda bir YSA insan beynindeki birçok nöronun, ya da yapay olarak basit işlemcilerin birbirlerine değişik etki seviyeleri ile bağlanması ile oluşan karmaşık bir sistem olarak düşünülebirir. Önceleri temel tıp birimlerinde insan beynindeki nöronların matematiksel madelleme çabaları ile başlayan çalışmalar, geçtiğimiz on sene içerisinde, disipline bir şekil almıştır. YSA bugün fizik, matematik, elektrik ve bilgisayar mühendisliği gibi çok farklı bilim dallarında araştırma konusu haline gelmiştir. YSA'nın pratik kullanımı genelde, çok farklı yapıda ve formlarda bulunabilen informasyon verilerini hızlı bir şekilde tanımlama ve algılama üzerinedir. Aslında mühendislik uygulamalarında YSA'nın geniş çaplı kullanımının en önemli nedeni, klasik tekniklerle çözümü zor problemler için etkin bir alternatif oluşturmasıdır.

4.2 Tarihsel Gelişimi

1940'larda Mc. Culloch ve Pitts nöronun lojik fonksiyonlarını sağlayan basit bir eşik cihazı olarak modellenebileceğini gösterdi (Culloch et al, 1943). Aynı zaman aralığında mühendislik temelleri geri beslenme ve beyin fonksiyonlarından faydalanan Wiener sibernetikin temelini atıyordu. 1949'da Donald Hebb "The organization behavior" adlı kitabında hücresel seviyede beyinin öğrenme mekanizmasında bahsetmiştir (Hebb,1949). Hebb'ın biyolojik öğrenme kuralı, bir nörondan dentrit yoluyla gelen bir aksonal giriş onun bir darbe üretmesine sebep olur. Sonraki aksonal girişlerin darbe üretmesi olasılığı artar. Böylelikle yapılan davranışın mükafatı ortaya çıkar. Hızlı hesaplamaya yönelik ilk YSA çalışmaları 1950'li yıllarda başlamış ve basit nöron modellerine dayalı bir hesaplama modeli 1950'lerde Rosenblatt tarafından önerilmiştir (Rosenblatt, 1958). 1960'lı yıllarda Widrow ve Hoff, bu basit nöron madellerini kullanarak ilk öğrenebilen adaptif sistemler üzerinde çalışmıştır (Widrow et al, 1960). Ancak 1969'da Minsky ve Papert yayınladıkları Perceptron adlı bir kitapta YSA yardımı ile öğrenmede ve hesaplamada aşılması zor engeller olduğunu iddia etmişler ve bu iddia YSA konusundaki çalışmaları büyük ölçüde yavaşlatmıştır (Minsky and Papert, 1969).

1969-1982 yılları arasındaki çalışmalarda teori artık oturuyor ve 1982'de J.J. Hopfield tarafından yayınlanan "Neural Networks and Physical systems" adlı çalışması ile çağdaş YSA devri başlar (Hopfield, 1982). Bu çalışmada Hopfield nöronların karşılıklı etkileşimlerine dayanan bir nöral hesaplama modeli önermiştir. Model bir enerji fonksiyonunu alabileceği en az değerine indiren 1. mertebe lineer olmayan diferansiyel denklemlerden oluşmuştur. Hopfield ağ seviyesinde, tek tek nöron seviyesinde var olmayan hesaplama kapasitesinin bulunduğunu öne sürdü. Bu tür YSA'na Hopfield ağı

denmektedir. Hopfield'in geri beslemeli YSA modelini ortaya atması ve bunun pratik optimizasyon problemlerinde kullanabilirliğini göstermesi YSA konusundaki çalışmaları yeniden hızlandırmıştır.

1976'da Grossberg ART(Adaptive Resonance Theory) adında bir YSA yapısı geliştirdi (Grossberg, 1976). ART çok gelişmiş bir YSA'dır ve henüz çok fazla probleme uygulanmamıştır. O sıralarda Kohonende (Self-Organizing Maps) "kendini düzenleyen nitelik haritasını" geliştiriyordu (Kohonen, 1982). Bu YSA nümerik airodinamik akış hesaplamaları için çoğu algoritmik yöntemden daha etkili olmuştur.

1986'da Rumelhart ve arkadaşlarının "Parellel Distributed Processing" grubu ileri beslemeli modellerde yeni öğrenme modeli olan hatanın geriye yayılması algoritmasını (back propagation algorithm) geliştirecek, bu konudaki daha önce iddia edilen aksaklıkların aşılabileceğini göstermişlerdir (Rumelhart et al, 1986). Bugün endüstride birçok YSA uygulamasında bu öğrenme yöntemi ile bunun değişik varyasyonları kullanılmaktadır. (Bu tez çalışmasındada kullanılan yöntemdir). Back-propagation algoritması, çok kullanılan ve öğrenilmesi kolay bir ağdır.

Widrow ve öğrencileri ise ADALINE'den (ADAptive LINear Element) sonra MADALINE'da geliştirip bu YSA'ya uygulama alanları buldular. MADALINE I'i 1987'de MADALINE II, 1988'de (David Andes'in keşfettiği) MADALINE III takip etti (Widrow and Lehr, 1990).

Günümüzde YSA'nın teorik çalışmaları büyük ölçüde tamamlanmış olup, 1986'dan bu yana uygulamaya yönelik çalışmalar son derece yoğun bir şekilde devam etmektedir.

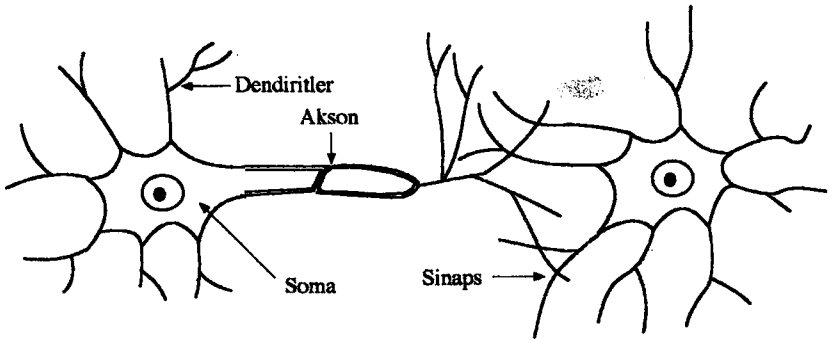
4.3 YSA'nın Tanımı ve Modeli

4.3.1 YSA'nın Tanımı

YSA paralel dağılmış bir bilgi işleme sistemidir. Bu sistem tek yönlü işaret kanalları (bağlantılar) ile birbirine bağlanan işlem elemanlarından oluşur. Çıkış işareti bir tane olup isteğe göre çoğaltılabilir. YSA yaklaşımının temel düşüncesiyle, insan beyninin fonksiyonları arasında benzerlik vardır. Bu yüzden YSA sistemine insan beyninin modeli denilebilir. YSA çevre şartlarına göre davranışlarını şekilleyebilir. Girişler ve istenen çıkışların sisteme verilmesi ile kendisini farklı cevaplar verebilecek şekilde ayarlayabilir. Ancak son derece karmaşık bir içyapısı vardır. Onun için bugüne kadar gerçekleştirilen YSA; biyolojik fonksiyonların temel nöronlarını örnek alarak yerine getiren kompoze elemanlardır.

4.3.2 Nöronun Biyolojik Yapısı ve Nöron Modeli

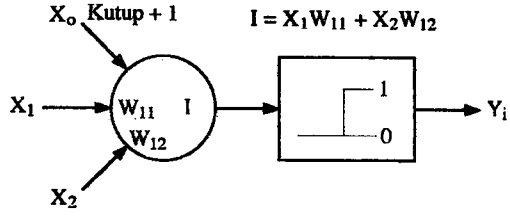
İnsanın bilgi işleme olayı beyninde gerçekleşir. Sinir sisteminin en basit yapısı nöronlardır. Vücudun değişik yerleri ile bilgi alışverişi yapan nöron hücresidir. Şekil 4.1 de basit bir nöron hücresi görülmektedir.



Şekil 4.1 Basit bir nöron yapısı

Nöron, soma adı verilen hücre gövdesi dentrit denilen kıvrımlı uzantılar ve somanın dalları sayesinde nöronu dallarına bağlayan tek sinir fiberli aksondan oluşur. Dendrit'ler hücreye gelen girişleri toplarlar. Dendrit tarafından alınan işaretler hücrede birleştirilerek bir çıkış darbesi üretilip üretilemeyeceğine karar verilir. Eğer bir iş yapılacaksa üretilen çıkış darbesi aksonlar tarafından taşınarak diğer nöronlarla olan bağlantılara veya terminal organlara iletilir. Beyindeki korteksde her nöronun bir karşılığı vardır. Bir nöronun çıkışı ona bağlı olan bütün nöronlara iletilir. Fakat korteks, işin yapılabilmesi için hangi nöron harekete geçirilecekse, sadece ona komut gönderir.

Somanın içinde ve çevresinde sodyum, kalsiyum, potasyum ve klor iyonları vardır. Potasyum yoğunluğu nöronun içinde, sodyum yoğunluğu dışındadır. Somanın zarı elektriksel olarak uyarılınca (söz konusu uyarı genellikle bir gerilim düşmesidir) zar, Na ve Ca gibi diğer iyonların içeri geçmesine izin verir ve somanın iç durumunu değiştirir nöronlar arasındaki bağlantılar hücre gövdesinde veya "sinaps" adı verilen dendritlerdeki geçişlerde olur. Yardımcı bir benzetme aksonlarla, dendritleri elektrik sinyallerini nörona ileten değişik empedansdaki yalıtılmış iletken olmasıdır. Sinir sistemi milyarlarca nöron ile tek bir nörondan çıkan aksonun 10000 kadar diğer nöronu bağlayan bir ağdır. Sinapslarla düzeltilen işaretleri taşıyan aksonlar ve dendritlerle içiçe geçmiş önöronlar bir sinir ağı oluştururlar. Şekil 4.2'de en basit formda gösterilen nöron modeli, bir eşik birimi olarak algılanabilir.



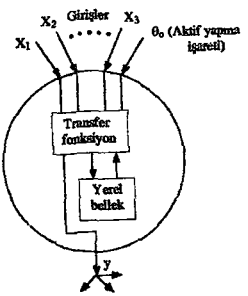
Şekil 4.2 Nöron modeli

Eşik birimi, çıkışları toplayan ve sadece girişin toplamı iç eşik değerini aştığında bir çıkış üreten işlem elemanıdır. Bir eşik birimi olarak nöron sinapslarındaki işaretleri alır ve hepsini toplar. Eğer toplanan işaret gücü eşiği geçecek kadar güçlü ise diğer nöronları ve dendritleri uyaran akson boyunca bir işaret gönderilir. Kesişen dendritlerden gelen sinapslarla kapılanan bütün işaretleri soma toplar. Toplam işaret daha sonra nöronun iç eşik değeri ile karşılaştırılır ve eşik değeri aşmışsa aksona bir işaret yayar. YSA, bu basit nöronların (düğümlerin yada ünitelerin) bağlanarak bir ağ'a dönüştürülmesiyle meydana getirilir.

4.4 YSA'nın Yapısı ve İşlem Elemanı

YSA temel olarak, basit yapıda ve yönlü bir graf biçimindedir. Her bir düğüm hücre denilen n. dereceden lineer olmayan bir devredir. Düğümler işlem elemanı olarak tanımlanır. Düğümler arasında bağlantılar vardır. Her bağlantı tek yönlü işaret iletim yolu (gecikmesiz) olarak görev yapar. Her işlem elemanı istenildiği sayıda giriş bağlantısı ve tek bir çıkış bağlantısı alabilir. Fakat bu bağlantı kopya edilebilir. Yani bu tek çıkış birçok hücreyi besleyebilir. Ağ'daki tek gecikme çıkışları ileten bağlantı yollarındaki iletim gecikmeleridir. İşlem elemanının çıkışı istenilen matematiksel tipte olabilir. Kısmen sürekli çalışma konumunda "aktif" halde eleman bir çıkış işareti üretir. Giriş işaretleri YSA'na bilgi taşır. Sonuç ise çıkış işaretlerinden alınabilir. Şekil 4.3 'de genel bir işlem elemanı (nöron, düğüm) gösterilmiştir.

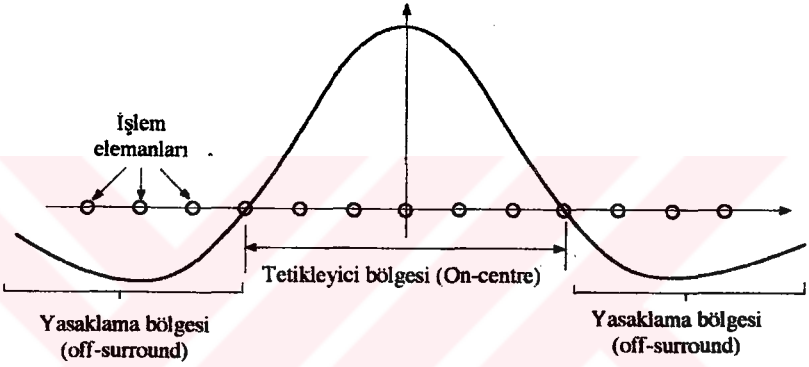
YSA birtakım alt kümelere ayrılabilir. Bu alt kümelerdeki elemanların transfer fonksiyonları aynıdır. Bu küçük gruplara "katman" layer adı verilir. (örn:çok katmanlı perceptron MLP) Ağ katmanların birbirlerine hiyerarşik bir şekilde bağlanmasından oluşmuştur. Dış dünyadan alınan bilgi giriş katmanı ile taşınır. Bir transfer fonksiyonları yoktur. YSA transfer fonksiyonu ve yerel bellek elemanı bir öğrenme kuralı ile giriş çıkış işareti arasındaki bağıntıya göre ayarlanır. Aktif yapma girişi için bir zamanlama fonksiyonu tanımlaması gerekebilir.



Şekil 4.3 Genel işlem elemanı yapısı

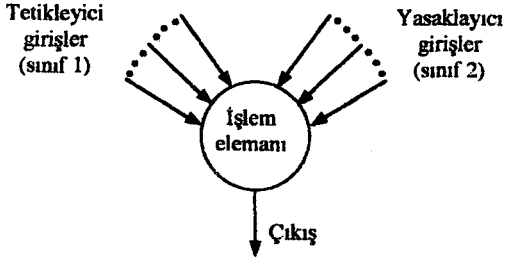
4.4.1 Giriş İşareti Sınıfları

İşlem elemanının transfer fonksiyonu gelen bütün giriş işaretleri için tanımlanır. Bazen değişik katman davranışlarının farklı olması tabiidir. İşaretlerin hangi bölgelerden geldiğinin bilinmesi gerekir. Değişik bölgelere göre işaretlerin sınıfları tamamlanabilir. Sıkça izlenen bir yapı ise merkezde evet/çevrede hayır (on centre/off surround) yapısıdır. Şekil 4.4'de bu yapı gösterilmektedir. Meksika şapkasına benzer bağlantı tipindedir.



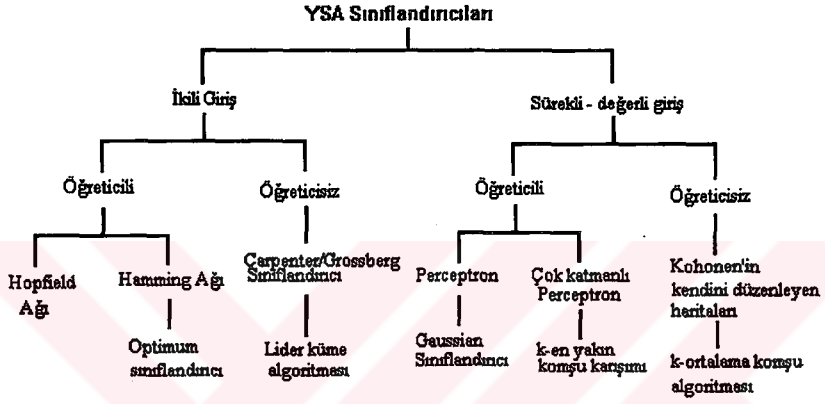
Şekil 4.4 Komşu hücrelerin merkez hücreye etkisi

İşlem elemanı tetikleyici girişlerin kendine yakın komşu girişlerden yasaklanan girişlerini daha uzaktan alır. Böylece işlem elemanına gelen girişler sınıflarına göre değerlendirilmiş olur. Tetikleyici bölgeden gelen girişler yasaklanan sınıfı oluşturur. Şekil 4.5 böyle bir işlem elemanını gösterir.



Şekil 4.5 Tetikleyici ve yasaklanan girişlere sahip bir işlem elemanı

Bir işlem elemanına gelen girişler matematiksel tiplerine göre etiketlendirilerek sınıflandırılır. YSA, giriş veri tiplertine göre ikili giriş (0,1) ve sürekli değerli giriş olmak üzere aşağıdaki gibi sınıflandırılır



Bu tezde giriş işareti olarak seçilen EMG işareti sürekli-değerli (reel sayı) olduğundan dolayı, sınıflandırıcı olarak öğreticili öğrenmeye sahip olan çok katmanlı perseptron kullanılmıştır.

4.4.2 Bağlantı Geometrileri

Bağlantılarda taşınan işaret verisinin cinsi tanımlanmalıdır. Bağlantı geometrisi YSA için çok önemlidir,bağlantı işareti her cinsten olabilir. Bağlantının nerede başlayıp nerede bittiğini bilmesi gerekir. 1'den N'e kadar olan bir işlem elemanı kümesinin bağlantıları aşağıda tanımlandığı gibi NxN boyutlu matris biçiminde gösterilebilir.

$$\left[w_{ij}\right] = \begin{matrix} w_{11} & w_{12} & . & . & w_{1n} \\ w_{21} & w_{22} & . & . & w_{2n} \\ . & . & . & . & . \\ . & . & . & . & . \\ w_{n1} & w_{n2} & . & . & w_{nn} \end{matrix}$$

$w_{ij} = w_{ji} = 1 \Leftarrow i.$ işlem elemanı j işlem elemanına bağlı

$w_{ij} = w_{ji} = 0 \Leftarrow$ bağlı değil

En fazla N^2 bağlantı olur. Bağlantılar çeşitli geometrik bölgeler arasında demetler halinde düşünülebilir. Bu bağlantı demetlerinin uyması gereken kurallar şunlardır.

1- Bağlantı demetini oluşturan işlem elemanları aynı bölgeden çıkmalıdır.

2- Bağlantı demetinin işaretleri aynı matematiksel tipten olmalıdır.

3- Bağlantı demetinin işaretleri aynı sınıftan olmalıdır.

4- Bağlantı demetinin bir seçim fonksiyonu (σ) olmalıdır.

$$\sigma : T \rightarrow 2^S$$ T: Hedef bölgesi S: kaynak bölgesi

Hedef bölgesindeki her işlem elemanı kaynak bölgesindeki her elemana giderse "tam" full bağlıdır. (örn:çok katmanlı perceptron). Eğer her hedef bölgesi elemanı N kaynak bölgesi elemanına bağlı ise " düzgün dağılmış" (uniform) olasıldır. Ayrıca her bir elemana, yine bir kaynak elemanı bağlı ise buna "bire-bir" bağlı denir.

4.4.3 Ağ Tipleri

Üç çeşit ağ tipi vardır

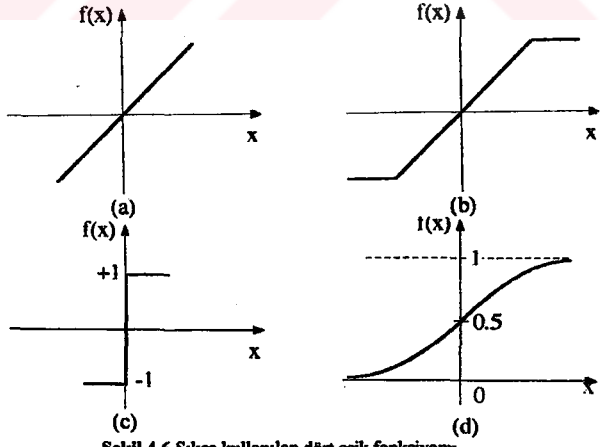
1- İleri beslemeli ağ: Her bir katmandaki hücreler sadece bir önceki katmanın hücrelerince beslenir.

2- Kaskat bağlantılı ağ: Hücreler sadece önceki katmanlardaki hücrelerce beslenir.

3- Geri beslemeli ağ: En az bir hücre sonraki katmanlardaki hücrelercede beslenir. (Bu çalışmada hem ileri, hemde geri beslemeli ağ tipi birlikte uygulanmalıdır).

4.4.4 Eşik Fonksiyonları

Transfer veya işaret fonksiyonları olarak da adlandırılan eşik fonksiyonları, muhtemel sonsuz domen girişli işlem elemanlarını önceden belirlenmiş sınırda çıkış olarak düzenler. Dört tane yaygın eşik fonksiyonu vardır. Bunlar, rampa, basamak ve sigmoid fonksiyonudur. Şekil 4.6'da bu fonksiyonlar gösterilmiştir.



Şekil 4.6 Sıkça kullanılan dört eşik fonksiyonu

Şekil 4.6 (a)'da gösterilen lineer fonksiyonun denklemi aşağıdaki gibidir.

$$f(x) = \alpha x$$

α: işlem elemanının x aktivitesini ayarlayan reel değerli bir sabittir. Lineer fonksiyon [-τ,+τ] sınırları arasında kısıtlandığında (b)'deki rampa eşik fonksiyonu olur ve denklemi;

$$f(x) = \begin{cases} +\tau : \text{eğer } x \geq \tau \text{ ise} \\ x : \text{eğer } |x| < \tau \text{ ise} \\ -\tau : \text{eğer } x \leq -\tau \text{ ise} \end{cases}$$ şeklini alır.

+τ (-τ) işlem elemanının maksimumu (minimumu) çoğu zaman doyma seviyesi olarak adlandırılan çıkış değeridir. Eğer eşik fonksiyonu bir giriş işaretine bağlı ise yaydığı +τ giriş toplamı pozitif, bağlı değilse eşik basamak fonksiyonu [-δ] olarak adlandırılır. Şekil 3.6 (c), basamak eşik fonksiyonunu gösterir ve denklemi;

$$f(x) = \begin{cases} +\tau : \text{eğer } x > 0 \text{ ise} \\ -\delta : \text{diğer durumlar} \end{cases}$$ şeklindedir.

Son ve en önemli eşik fonksiyonu (bu çalışmada kullanılan) sigmoid fonksiyonudur. Şekil 4.6 (d) de gösterilen S biçimindeki sigmoid fonksiyonu; seviyeli, lineer olmayan çıkış veren, sınırlı, monoton artan fonksiyondur. Denklemi;

$$f(x) = \frac{1}{1+e^{-x}}$$ biçimindedir

Her işlem elemanı kendisine verilen yerel veriye göre, kendisini ayarlayacak bütün YSA'nın enformasyon bölgesinin öğrenmesini sağlar. (Enformasyon bölgesi olasılık-yoğunluk fonksiyonu ilde tanımlanabilir). Enformasyon bölgesi birçok uygulamada, gerçek değerin "0" ile "1" arasında normalize edilmesi gerekir. (Normalize etmek:gerçek değeri 85 olan bir girişi 0.85 şeklinde ağa uygulamaktır.) Normalizasyon aynı anda bütün girişlere uygulanabilir.

4.4.5 Ağırlık Uzayı

Bir çok YSA öğrenme işlemi, işlemelemanlarının ağırlığı değiştirilerek sağlanır. Böylece tanımlanan ağırlık değiştirilerek öğrenmede iyi bir model kullanıp, ağırlıkların bu modele göre değiştirilmesi esastır. Basit bir matematiksel model olarak herbir işlem elemanının "n" adet gerçek ağırlığı olduğu düşünülerek ve N adet işlem elemanı gözönüne alınırsa;

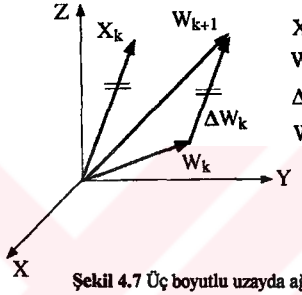
$$w = (w_{11}, w_{12}, \ldots, w_{1n}, w_{21}, w_{22}, \ldots, w_{2n}, \ldots, w_{N1}, w_{N2}, \ldots, w_{Nn})^T$$

$$w = (w_1^T, w_2^T, w_3^T, \ldots, w_N^T)$$

$w_1, w_2, \ldots, w_N$: işlem elemanlarının ağırlık vektörleridir.

$$w_1 = \begin{matrix} w_{11} \\ w_{12} \\ . \\ . \\ w_{1n} \end{matrix} \quad \ldots\ldots\ldots w_N = \begin{matrix} w_{N1} \\ w_{N2} \\ . \\ . \\ w_{Nn} \end{matrix}$$

YSA ağırlık vektörü N, n boyutlu orkid uzayında yayılır. YSA'nın enformasyon işleme performansı, ağın ağırlık vektörünün belirli bir değeri ile bulunacaktır. Aşağıda şekil 4.7 de ağırlıkların düzeltiminin vektörel çizimi verilmiştir.



X_k : Giriş işareti vektörü
W_{k+1} : Gelecekteki ağırlık vektörü
ΔW_k : Ağırlık vektörünün değişimi
W_k : Şimdiki ağırlık vektörü

Şekil 4.7 Üç boyutlu uzayda ağırlık vektörünün değişimi

Şekilde görüldüğü gibi $\Delta\vec{W}_k, \vec{X}_k$ ile aynı doğrultuda olduğunda istenen hata düzeltimini en küçük ağırlık değişimi ile elde etmek mümkündür. Böylece yeni bir giriş örüntüsü uygulandığında önceki eğitim örüntülerinin cevabı en az bozulmuş olur. Hata değişimini inceleyen iki çeşit kural vardır.

1- Hata düzeltme kuralları , 2- Gradyen kuralları

Hata düzeltme kuralları; Her bir giriş örüntüsünde ağırlıkları yeniden ağırlayarak çıktı hatasını en aza indirmeye çalışırlar. Gradyen kurallarında ise, ağırlıklar yeniden ayarlanarak ortalama karesel hatayı (MSE) en aza indirilmeye çalışılır.

Ağırlık vektörü ile çalışan YSA'da önemli noktalardan birisi, bir öğrenme kuralı geliştirip, enformasyon bölgesi kullanarak (eşik fonksiyonu ile) ağırlık vektörü "w" yı istenilen YSA performansı verecek noktaya yöneltmektir. Genellikle öğrenme kuralı için bir performans ya da maliyet fonksiyonu tanımlanır. Minimizasyon veya maksimizasyon ile "w" vektörü bulunur. Bir performasyon çeşidi olarak bilinen, MSE (karesel ortalama hata) şu şekilde tanımlanır.

$$F(w) = \oint_A |f(x) - G(x,w)|^2 \rho(x) dv(x)$$

Amaç F'i küçültmeye çalışmaktır.

y = G(w,x):sistemin giriş çıkış fonksiyonu.

y: çıkış işareti vektörü

x: giriş işareti vektörü

w: ağırlık vektörü
$\rho(x)$: olasılık yoğunluk fonksiyonu

4.5 YSA'da Eğitme (Training)

4.5.1 Eğitme Algoritmaları

Eğitme algoritmaları YSA'nın ayrılmaz bir parçasıdır. Eğitme algoritması eldeki problemin özelliğine göre öğrenme kuralını YSA'na nasıl adapte edeceğimizi belirtir. Üç çeşit eğitme algoritması yaygın olarak kullanılmaktadır.

1- Öğreticili eğitme (supervised traning).

2- Skor ile eğitme (graded training).

3- Kendini düzenleme ile eğitme (self-organization training)

Öğreticili eğitmede, elimizde doğru örnekler vardır. Yani $(x_1,x_2,....,x_n)$ şeklindeki giriş vektörünün, $(y_1,y_2,...,y_n)$ şeklindeki çıkış vektörü, tam ve doğru olarak bilinmektedir. Herbir (x_1,y_1), $(x_2,y_2),...,(x_N,y_N)$ çifti için ağ doğru sonuçları verecek şekilde seçilen bir öğrenme kuralıyla beraber eğitilir.

Skor ile eğitmede giriş işaretlerine karşılık gelen çıkış işaretleri tam olarak bilinmemektedir. Çıkış işareti yerine skor verilir ve ağın değerlendirilmesi yapılır. Özellikle kontrol uygulamaları için idealdir. Çeşitli maliyet (cost) fonksiyonları kullanılır.

Kendini düzenleyen ağ, giriş işaretine göre kendini düzenleyerek organize eder. Olasılık yoğunluk fonksiyonlarına, sınıflandırma ve şekil tanıma problemlerine uygulanabilir.

Ne tür eğitme yöntemi kullanılırsa kullanılsın, herhangi bir ağ için gerekli karekteristik özellik, ağırlıkların verilen eğitme örneğine nasıl ayarlanacağının belirtilerek öğrenme kuralının oluşturulmasıdır. Öğrenme kuralının oluşturulması için bir örneğin ağa defalarca tanıtılması gerekebilir. Öğrenme kuralı ile ilişkili parametreler ağın zaman içinde gelişme kaydetmesiyle değişebilir. Hangi YSA algoritmasında ne tür bir eğitme kullanıldığı bu bölümün giriş işaretlerinin sınıflandırılması kısmında gösterilmiştir.

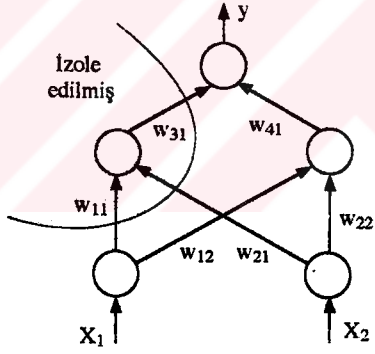
4.5.2 Bellek

YSA'nın önemli bir özelliği bilgiyi saklama şeklidir. YSA'da bellek, birçok yerel bellekler oluşturularak dağıtılır. Bağlantı ağırlıkları YSA bellek biçimleridir. Ağırlıkların değerleri ağın o anki bilgi durumunu temsil eder. Mesela; bir giriş/istenen çıkış çiftinin belirtilen bilgi parçası ağın içinde birçok bellek biçimine dağıtılmıştır. Bellek üniteleri ile diğer saklı bilgiler, bu bilgiyi paylaşırlar. Bazı

ilişkilidir. Öyleki eğitilen ağa birkısmı uygulanırsa, ağ bu girişe belleğindeki en yakın çıkışı bu giriş için seçer ve tam girişe bağlı çıkış ortaya çıkar. Eğer YSA oto-ilişkili ise, kısmi giriş vektörlerinin ağa verilmesi bu girişlerin tamamlanması ile sonuçlanır. YSA belleğinin yapısı; eksik, gürültülü ve tam seçilemeyen bir giriş uygulandığı zaman bile mantıklı çıkış üretmeye uygundur. Bu kurala "genelleme " adı verilir. Bir genellemenin kalitesi ve anlamı, uygulama çeşidine, ağın tipine ve karmaşıklığına dayanır. Lineer olmayan çok katmanlı ağlar (özellikle geriye yayınım ağları) gizli katmandaki özelliklerden öğrenirler ve bunları çıkışlar üretmek için birleştirirler. Gizli katmandaki bilgi, yeni giriş örüntülerine akılcı çözümler oluşturmak için kullanılabilir.

4.5.3 Hata Toleransı

Klasik hesaplama sistemleri çok az bir zarardan bile etkilenir. YSA için durum farklıdır. Bu farklılık YSA'nın hata toleranslı olmasıdır. İşlem elemanlarının az da olsa zarar görmesi sistemin bütününü etkiler. YSA paralel dağılmış parametreli bir sistem olduğundan her bir işlem elemanı izole edilmiş bir ada olarak düşünülebilir. Şekil 4.8 'de çok katmanlı perseptron (MLP) için bu durum gösterilmiştir.



Şekil 4.8 MLP'nin izole edilmiş hali

Daha çok işlem elemanın zarar görmesi ile sistemin davranışı biraz daha değişir. Performans düşer ama sistem hiç bir zaman durma noktasına gelmez. YSA sistemlerinin hata toleranslı olmasının nedeni bilginin tek bir yerde saklanmayıp, sisteme dağıtılmasıdır. Bu özellik sistemin durmasının önemli bir zarara neden olacağı uygulamalarda önem kazanır.

4.6 Öğrenme Kuralları

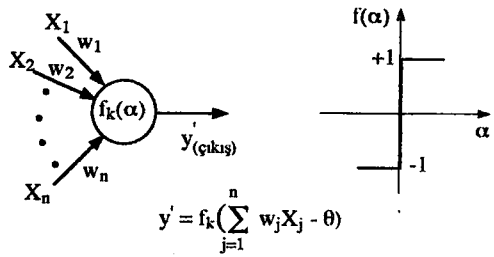
Bilginin kurallar şeklinde açıklandığı klasik uzman sistemlerin tersine, YSA gösterilen örnekten öğrenerek kendi kurallarını oluşturur. Öğrenme; giriş örneklerine veya (tercihen) bu girişlerin çıkışlarına bağlı olarak ağın bağlantı ağırlıklarını değiştiren veya ayarlayan öğrenme kuralı ile gerçekleştirilir. Öğreticisiz öğrenmede her giriş işareti için istenen çıkış sisteme tanıtılır ve YSA giriş/çıkış ilişkisini gerçekleştirene kadar kademe kademe kendini ayarlar. Günümüzde kullanılan birçok öçğrenme kuralı vardır. Bilinen en çok kullanılan öğrenme kuralları şunlardır.

- Raslantısal (Hebb) öğrenme kuralı
- Performans (Widrow ve ADALİNE) öğrenme kuralı
- Kompetetif (Kohonen) öğrenme
- Filtreleme (Grossberg)
- Spotitemporal öğrenme
- Genelleştirilmiş Delta Kuralı Öğrenme

Burada bütün öğrenme kuralları incelenmeyecektir. Sadece tezde kullanılan "Genelleştirilmiş Delta Kuralı" öğrenmesi ilk oluşumundan yani perceptron halinden başlayıp, tüm gelişimiyle son durumu anlatılacaktır.

4.6.1 Perceptron (idrak,almaç)

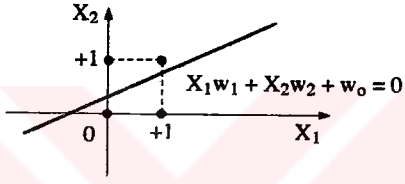
Perceptron ağı, ilk 1943 yılında Mc Culloch ve Pitts tarafından saptandı. Onların bahsettikleri YSA tipi aşağıda şekil 4.9'da gösterildiği gibidir (Culloch and Pitts, 1943).



Şekil 4.9 Tek katmanlı perceptron yapısı

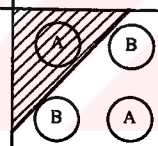
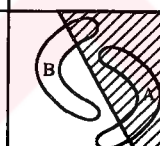
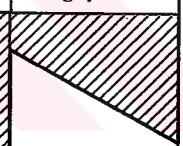
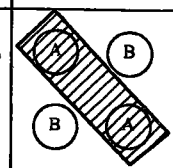
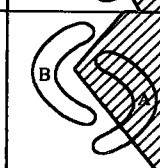
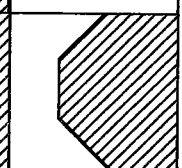
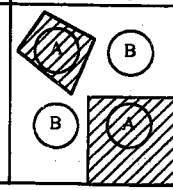
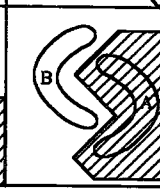
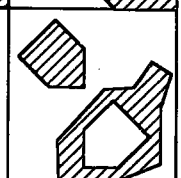
$\sum_{j=1}^{n} w_j x_j - \theta = 0$ n-boyutlu uzayda n-1 boyutlu bir düzlem belirler.

Yani bu ilk perkeptron modeline göre, giriş bilgisinin mevcut iki sınıftan hangisine eşit olabileceğini bulacak şekilde eğitilen basit bir ağdır. Daha sonra 1960 yıllarında F.Rosenblatt yukarıdaki ağ tipini biraz daha geliştirdi (Hebb,1949).Ama Minsky ve Papert bu tek katmanlı perceptronun XOR (ayrıcalıklı veya) işlemini gerçekleştiremediğini ispatladılar. Şekil 4.10 ve şekil 4.11'den anlaşılacağı gibi 0'ların bir tarafta 1'lerin bir tarafta ayıracak şekilde bir bölge oluşamıyor. XOR gibi 3 veya daha fazla sınıfa ihtiyaç duyulan problemleri çözmek için yapılması gereken işlem; YSA yeni katmanlar eklemektir. Eşik bağlarıyla oluşturulan karar bölgesi şeklinin karmaşıklığı sadece eklenmiş olan katmanların sayısıyla sınırlıdır. Şekil 4.11'de gösterildiği gibi içbükey ayrılabilir fonksiyonlar gerçeklenebilir (Pao, 1989).



X_1	X_2	f(x)
0	0	0
1	0	1
0	1	1
1	1	0

Şekil 4.10 Lineer yayılabilirliğin gösterimi

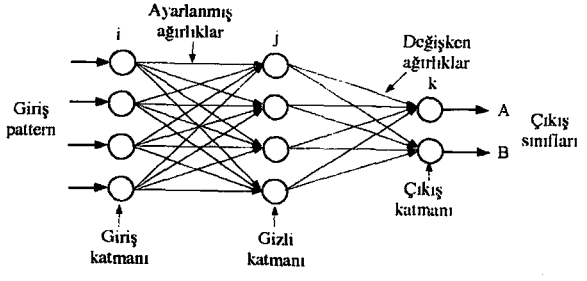
YAPI	Karar Bölgeleri tipi	XOR Problemi	Bölgelere Dayalı sınıflar	En iyi ayırdığı bölge şekilleri
Tek katman (a)	Doğrusal ayrıştırılabilir. Ayrıcaklı veya işlevini gerçekleyemez.			
İki katman (b)	Konveks (iç bükey) açık veya kapalı bölgeler.			
Üç katman (c)	İç bükey olmayan hatta bağlantılı olmayan bölgeler			

Şekil 4.11 Çok katmanlı perceptronda gizli katmanın rölü

Bilgi lineer yayılamıyorsa 2. katmanın çıkışı konveks bölgededir ve bunun neticesi olarak 3. katmandan gelecek olan çıkış bilgisinin şekli, herhangi bir bölgenin şeklinde olabilir. Bu sebeple ihtiyaç duyulan katman sayısı üç olmaktadır. Şekil 4.11'den anlaşılacağı gibi iç-bükey olmayan hatta basit bağlantılı olmayan bölgeleri kümelemek ancak 3-katmanlı ağ ile mümkündür. 3-katmanlı perceptronun 2. katmanında ihtiyaç olan düğümlerin sayısı, bir karar bölgesinin birleştirilmemiş hali veya bir ağ gözünün bir dış-bükey alandan meydana gelemediğinin birinden büyük olması lazımdır. 2. katmandaki düğüm sayısı en kötü durumda giriş bilgilerinin dağılımını yapan bölgenin bağlanmamış sayısına eşit olması gerekir. Birinci katmandakilerin sayısı her iki katmandaki değişim ile 3 ya da 4 köşeli dışbükey bir alan oluşturmaya yeterli seviyede olmalıdır. Bunun tipik bir sonucu olarakda en az 1. katmandakinden üç kat fazla miktarda olması gerekmektedir. Bununla beraber Gutierrez ve arkadaşları değişik perceptron ağlarının ihtiyacı olan düğüm sayıları hakkında çalışma yaptılar ve çok fazla düğümünde, çok az sayıda olduğu gibi zararlı etkisi olduğunu buldular (Gutierrez et al, 1989). Tek katmanlı perceptron uygulanan her eğitimin seti modelinin en önemli özelliği, lineer biçimde dağılmak zorunda olmasıdır. Şayet bu doğruysa, Rosenblatt perceptron ağının kararlı olacağını gösterdiler. (Mesela ağırlıklar iterasyonla yakınsar).

4.6.2 Çok Katmanlı Perceptron (Multi-Layer Perceptron)

Çok katmanlı perceptron giriş ve çıkış katmanları arasında birden fazla katmanın kullanıldığı YSA sistemleridir. Gizli katman (hidden layer) olarak isimlendirilen bu katmanlarda , düğümleri aracısız giriş olmayan ve aracısız çıkış veremeyen üniteler vardır. Şekil 4.12 'de çok katmanlı perceptronun genel yapısı verilmiştir.



Şekil 4.12 Çok katmanlı perceptron yapısı

İki katmanlı ağlarda veriler giriş katmanı tarafından kabul edilirler. Ağ içinde yapılan işlemler sonucunda çıkış katmanında oluşan sonuç değer işlenen cevap ile karşılaştırılır. Bulunan cevap ile istenen cevap arasındaki herhangi bir ayrılık varsa ağırlıklar bu farkı azaltarak şekilde yeniden düzenlenir. Girişteki değer , ağırlıklar uygun noktaya ulaşana kadar değişmez. Hesaplanan çıkışlar istenilen cevaplarla karşılaştırılarak sonuçta gerekirse hata belirtilir. Hata işareti gizli birimlerden çıkış birimine olan ağırlıkları değiştirmekte kullanılır. Ama bunu yaparken giriş katmanından gizli katmana gelenin değiştirilip değiştirelemediğini düşünmek gerekir. Gizli birimlerden ne tür bir çıkış istendiği bilinemeyeceği için gizli birimlerin çıkışında hata işareti verilmesi kolay bir şey değildir. Bunun yerine her bir birimin çıkış biriminin hatalarına olan etkisi bilinmelidir. Bu hatalı birim için gizli birime bağlı olan çıkış birimlerinin hata işaretlerinin ağırlıkları toplamı alınarak yapılır. Çok gizli katmana sahip sistemlerde her sistemin hata işaretleri, bir önceki katmanın düzeltilmiş işaretlerinden çıkartılarak işlem tekrarlanır. Sonuç olarak ağırlık düzeltme işlemi çıkış seviyesine bağlı ağırlıklardan başlar ve işlem ters yönde, giriş seviyesine varana kadar devam eder. Sonuçta sistem hatalar yapar, ama bu hatalardan birşeyler öğrenip isteneni bulana kadar işleme devam eder. Bu yönteme "hatanın geriye yayılması algoritması" (Back-propagation algorithms) denir.

4.6.3 Hatanın Geriye Yayılması Algoritması ve Genelleştirilmiş Delta Kuralı

Hatanın geriye yayılması algoritması, karesi alınmış hata fonksiyonunu minimize eden kodlu bir algoritma olup ve genelleştirilmiş delta kuralını eğitme için kullanılır. Şekil 4.13'de mimarisi gösterilen algoritma, ana hatlarıyla şöyledir:

m-boyutlu giriş örüntüleri set edildiğinde $X_i = [X_1, X_2 .... X_m]^t$ 'dir. Benzer şekilde istenilen n-boyutlu çıkış örüntüleri $d_k = [d_1, d_2 ..... d_n]^t$ belirtir. j. katmana giren ağ;

$$net_j = \sum_{i=1}^{m} w_{ji} . x_i \qquad (4.1)$$

j. düğüm çıkışı (transfer fonksiyonu çıkışı);

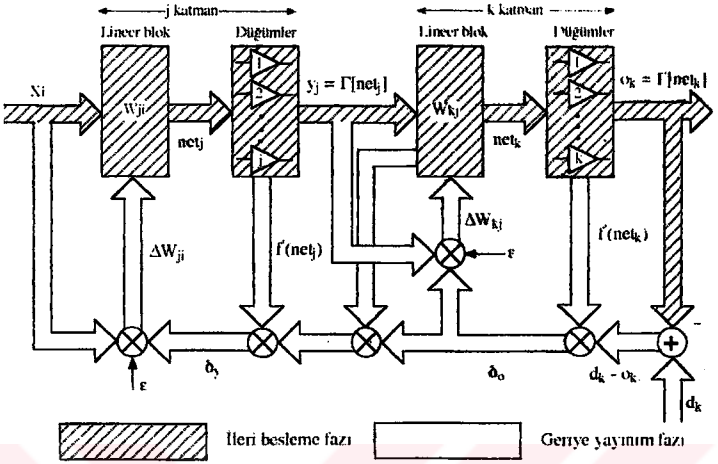
$$y_j = f(net_j); \ j = 1,2,....J \qquad (4.2)$$

k katmanına ait nörona giren ağ;

$$net_k = \sum_{j=1}^{J} w_{kj} . y_j \qquad (4.3)$$

k. düğümünün lineer olmayan çıkışı;

$$o_k = f(net_k) \ ; \ k = 1,2,....,K \qquad (4.4)$$

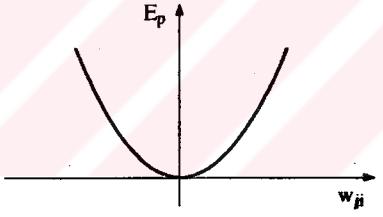


Şekil 4.13 Hatanın geriye yayılması algoritmasının blok diyagramı

Herbir pattern (örüntü) için karesel hata;

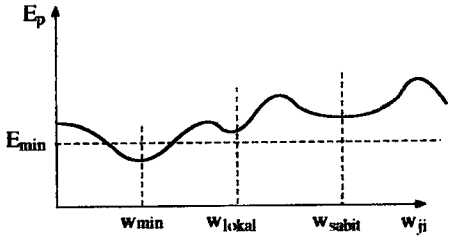
$$E = \frac{1}{2}\sum_k (d_k - o_k)^2 \qquad (4.5)$$

olur. Hata fonksiyonu Şekil 4.14 de gösterildiği gibidir.



Şekil 4.14 Gizli katmanı olmayan ağın hata fonksiyonu

Gizli katman olduğu zaman; Hata düzeyi şekil 4.14'de olduğu gibi sadece bir minimumdan oluşmuyor. Şekil 4.15'deki gibi çeşitli minimumlar oluşur. Öğrenmede en küçük minimuma ulaşılmak istenir.



Şekil 4.15 Gizli katmana ait ağın hata fonksiyonu

Ağırlıkların değişimi;

$$\Delta W_{kj} = -\varepsilon \frac{\partial E}{\partial W_{kj}} \tag{4.6}$$

Burada, ε; öğrenme oranı adı verilen küçük değerde bir pozitif sayıdır. (4.6)'de eşitliğin sağ tarafını açarsak,

$$\frac{\partial E}{\partial W_{kj}} = \frac{\partial E}{\partial net_k}$$

$$\frac{\partial E}{\partial W_{kj}} = \frac{\partial E}{\partial net_k} \frac{\partial net_k}{\partial W_{kj}} \tag{4.7}$$

$$\frac{\partial net_k}{\partial w_{kj}} = \frac{\partial}{\partial w_{kj}} \sum_j w_{kj} \cdot y_j = y_j \tag{4.8}$$

Amaç uygun w seçimiyle, $E = \sum_p E_p$ toplam hatayı yeterince küçük yapmaktır. Bu amacı gerçekleştirmek için, bir p ∈ P örüntüsü ard arda ve rasgele biçimde seçilir. k. nöronda oluşan ve 'delta' adı verilen hata işareti,

$$\delta_o = -\frac{\partial E}{\partial net_k} \tag{4.9}$$

(4.8) ve (4.9)'u (4.7) de ve onuda (4.6) da yerine koyarsak,

$$\Delta w_{kj} = \varepsilon \delta_o y_j \tag{4.10}$$

elde edilir. Bu terime 'Delta Kuralı' denir. (4.9) kısmi türevlerine ayrılırsa,

$$\delta_o = -\frac{\partial E}{\partial net_k} = -\frac{\partial E}{\partial o_k} \frac{\partial o_k}{\partial net_k} \tag{4.11}$$

$$\frac{\partial E}{\partial o_k} = -(d_k - o_k) \tag{4.12}$$

olup k. nöron çıkışının lokal hatasını verir.

$$\frac{\partial o_k}{\partial net_k} = f_k'(net_k) \tag{4.13}$$

Son iki formülü (4.11)'de yerine koyalım;

$$\delta_o = (d_k - o_k) f_k'(net_k) \tag{4.14}$$

bulunur. Bu son terim (4.10) da yerine konulduğunda k nöronu için;

$$\Delta w_{kj} = \varepsilon (d_k - o_k) f_k'(net_k) y_j \tag{4.15}$$

Eğer ağırlıklar çıkış nöronlarını direkt olarak etkilemiyorsa (yani arada gizli katman var ise) (4.10)'a benzer biçimde delta kuralını uygulayalım.

$$\Delta w_{ji} = -\varepsilon \frac{\partial E}{\partial w_{ji}} \tag{4.16}$$

$$= -\varepsilon \frac{\partial E}{\partial net_j} \frac{\partial net_j}{\partial w_{ji}} = -\varepsilon \frac{\partial E}{\partial net_j} x_i$$

$$= -\varepsilon \left(-\frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial net_j} \right) x_i = \varepsilon \left(\frac{\partial E}{\partial y_j} \right) f_j'(net_j) x_i$$

$$\Delta w_{ji} = \varepsilon \delta_y x_i \tag{4.17}$$

Bununla birlikte $\partial E/\partial y_j$ faktörü direkt olarak geliştirilemez. Özel olarak çıkış katmanına etkisi;

$$-\frac{\partial E}{\partial y_j} = -\sum_k \frac{\partial E}{\partial net_k} \frac{\partial net_k}{\partial y_j} = \sum_k \left(-\frac{\partial E}{\partial net_k} \right) \frac{\partial}{\partial y_j} \sum_m w_{km} y_j = \sum_k \left(-\frac{\partial E}{\partial net_k} \right) w_{kj} = \sum_k \delta_o w_{kj}$$

bulunur. Bu durumda;

$$\delta_y = f_j'(net_j) \sum_k \delta_o w_{kj} \tag{4.19}$$

Kısaca özetleyecek olursak; (4.17) formulü şayet j. nöron, çıkış katmanı nöronuysa (4.14)' ebenzer biçimde,

$$\delta_y = (d_y - o_y) f_j'(net_j) \tag{4.20}$$

olur. Eğer j nöronları gizli katmana ait nöronlar ise ozaman (4.19) nolu formül kullanılır. Transfer (eşik) fonksiyonu olarak sigmoid fonksiyonu kullanıldığında;

$$f(net_j) = y_j = \frac{1}{1+e^{-net_j}} \tag{4.21}$$

olur. Türevi alınırsa,

$$f'(net_j) = \frac{1}{1+e^{-net_j}} \frac{1+e^{-net_j}-1}{1+e^{-net_j}} \qquad \text{gerekli sadeleştirme yapıldığında} \quad ;$$

$$\frac{\partial y_j}{\partial net_j} = y_j(1-y_j) \tag{4.22}$$

Benzer biçimde k katmanı içinde yapılırsa;

$$\frac{\partial o_k}{\partial net_k} = f_k'(net_k) = o_k(1-o_k) \tag{4.23}$$

elde edilir. Bu durumda (4.14) ve (4.19) daki deltalar aşağıdaki gibi olur. (δ_o :çıkış katmanı, δ_y :gizli katman elemanları içindir);

$$\delta_o = (d_k - o_k) o_k (1-o_k) \tag{4.24}$$

$$\delta_y = y_j(1-y_j)\sum_k \delta_o w_{kj} \tag{4.25}$$

ω_{ji} 'yi öğrenme durumunda eğitme örüntülerinin seti, her örüntü seti için Δw_{ji} yi hesaplamak gerekir. Öğrenme oranı (ε) hızlı öğrenmeyi sağlar fakat osilasyona sebep olabilir. Rumelhart ve arkadaşları (4.15) ve (4.17) ifadelerinin bir tür momentum terimi içermek üzere düzenlenebileceğini ileri sürmüşlerdir. Bu durumda;

$$\Delta W_{ji}(n+1) = \varepsilon\delta_y x_i + \alpha\Delta W_{ji}(n) \tag{4.26}$$

$$\Delta W_{kj}(n+1) = \varepsilon\delta_o y_j + \alpha\Delta W_{kj}(n) \tag{4.27}$$

olarak yazabiliriz. Burada n:öğrenme saykıllarının (iterasyon) sayısını gösterir. Momentum terimi olan (α) küçük değerde pozitif bir sayıdır.

4.6.4 Öğrenme ve Momentum Katsayıları

YSA ile ilgili bir başka sorunda, düzgün bir öğrenme katsayısının (ε) ayarlanmasıdır. Ağırlıkları çok yüksek tutmak davranışın bozulmasına neden olabilir. O nedenle öğrenme katsayısını böyle bir davranışı önlemek için küçük tutmak gereklidir. Öğrenme katsayısı, $0.01 < \varepsilon < 10$ aralığında seçilen sabit bir sayıdır. Öte yandan çok küçük bir öğrenme oranıda, öğrenme işleminin yavaşlamasına yol açar.

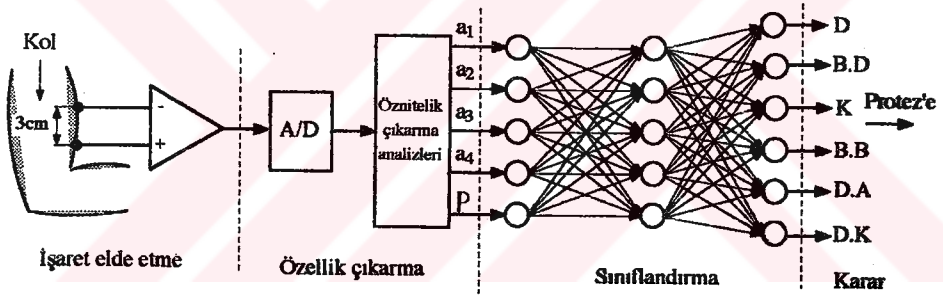
Momentum (α) fikri bu noktadan hareketle ortaya atılmıştır. Momentum mevcut delta ağırlığı üzerinden önceki delta ağırlığının belli bir kısmını besler. Böylece daha düşük öğrenme katsayısı ile daha hızlı öğrenme elde edilir. Momentum katsayısı genellikle $0 < \alpha < 1$ aralığında değişen sabit bir sayıdır.

5. ÇOK FONKSİYONLU PROTEZLER İÇİN YSA KULLANARAK MİYOELEKTRİK KONTROL

Bir miyoelektrik kontrollu protez temelde, işaret işleme birimi ile kontrol biriminden meydana gelmektedir. İşaret işleme birimi; EMG işaretini ölçer, kuvvetlendirme ve filtreleme gibi ön işlemlerden sonra bilgi taşıyan paremetreleri bulur. Kontrol biriminde ise; bulunan veriler kullanılarak çok fonksiyonlu hareket sınıflandırılması yapılır. Böylece özürlünün amaçladığı harekete karar verilir.

5.1 Sistem Kontrol Dizaynı

Bu çalışmada gerçekleştirilmesi istenen (amaç); ön kolun biceps ve triceps kaslarından alınan EMG işaretleriyle, protez motorlarını kumanda edecek lojik bilginin elde edilmesidir. İşte bu bilginin elde edilmesini sağlayan sistemin kontrol dizaynı Şekil 5.1'de gösterildiği gibidir.



Şekil 5.1 Sistem kontrol dizaynı

Şekilden de anlaşılacağı gibi, sistem kontrol dizaynı; işaret elde etme, özellik çıkarma, sınıflandırma ve karar aşamalarından oluşmaktadır. İşaret elde etme kısmında, gümüş-gümüş klorür elektrodlardan alınan EMG işareti önce kuvvetlendirilip, daha sonra 1kHz alçak geçiren filtre ile süzülerek dc seviye kaydırıcı devresine uygulanır. (Detaylı bilgi için bkz. Bölüm 2) Özellik çıkarma aşamasında; A/D çevirici (5kHz), veri kayıt ve AR kestirimi (Parcor yöntemi ile) yapılmaktadır. Enson aşamada ise; Yapay Sinir Ağları (YSA) ile verilen AR paremetrelerinin sınıflandırılması yapılarak karar verilir.

EMG işaretleri, 26 yaşında sağlıklı bir erkek denekten çeşitli kol hareketlerini sınayarak alınmıştır. Yapılan hareketler: dirsek kapama (elbow flexion), dirsek açma (elbow extension), bilek döndürme (wrist supination), bilek bükme (wrist flexion), kavrama (grasp) hareketleri ve dinlenme (resting)'dir. Her hareket 6 kez tekrarlanıp, her denemede yaklaşık bir saniyede 4800 örnek alınarak

saklanmıştır. Deneyler yapılırken hareketin başlangıç ve bitiş noktaları atılarak, hareketin lineer olduğu bölge alındı ve her denemede kaslara aynı kuvvet uygulanmaya gayret edildi.

Elde edilen verilerin işlenmesi aşamasında, EMG işaretleri normalize edildikten sonra dc seviyeleri bulundu ve herbir örnekten dc seviyeleri çıkartılarak işaret sıfır ortalamalı hale getirildi. Her deney için kaydedilen 4800 örneklik veri, herbiri 80 ms'lik 400 örneğe sahip olan 12 segmente ayrılarak incelendi. Her segment, "Blackman" tipi pencere fonksiyonu ile çarpılarak pencereleme işlemi yapıldı. Blackman tipi pencere fonksiyonunun ayrık zamandaki ifadesi aşağıdaki gibidir.

$$w(n) = 0.42 - 0.5Cos(\frac{2\pi n}{N-1}) + 0.08Cos(\frac{4\pi n}{N-1}),\ n = 0,1,2,....,(N-1)$$

Her hareket için elde edilen AR katsayıları (a_1, a_2, a_3, a_4) vektörlerinin oluşturduğu uzayda tek bir değerde olmayıp; istatistiksel, biyolojik vb. sapmalardan ötürü bir değerler kümesi şeklinde dağılım göstermektedir. Farklı hareketler için elde edilen bu değerler kümesi, belirtilen uzayda birbirinden yeterince uzaksa bu hareketler kolaylıkla ayırtedilebilmektedir. Yaptığımız deneylerde işlenecek her segmentin bir pencere fonksiyonu ile çarpılmasıyla elde edilen AR katsayıları, pencereleme yapılmadan elde edilenlere göre daha iyi gruplaşma sağlamıştır. Bu nedenle pencereleme işlemi yapılmıştır. Çeşitli hareketler için elde edilen AR katsayıları Tablo 5.1-5.6'da verilmiştir. Tabloda, sırasıyla a_1, a_2, a_3 ve a_4 katsayılarına ait değerler görülmektedir.

Tablo 5.1. Dinlenme AR Katsayıları

a_1	a_2	a_3	a_4
-9.6845563318E-01	-1.5400527307E-01	1.3427406528E-01	7.4366799392E-02
-1.0497303938E+00	2.4184004970E-03	4.0869415318E-02	1.2779659727E-01
-9.2388897922E-01	-1.4672540367E-01	2.6943985669E-01	-7.7113845847E-02
-9.4570361030E-01	-1.3319789043E-01	2.3991312436E-02	1.3348951770E-01
-8.6972619636E-01	-1.6419381431E-01	5.312518849E-02	8.6418984900E-02
-9.9760288423E-01	-2.1539712883E-01	3.4189008671E-01	-6.8069358428E-03
-.6308925263E-01	-2.1380003899E-01	1.2691625064E-01	8.8365856683E-02
-7.5374334159E-01	-2.7940743845E-01	1.4858593883E-01	1.1780134702E-01
-9.3169638653E-01	-2.2416827596E-01	3.0335611006E-01	-7.8712744674E-02
-1.0011006377E+00	-9.4817987105E-02	7.3991220989E-02	1.4613025027E-01
-8.8217377929E-01	-2.5375166024E-03	-1.3407080454E-02	1.4986096858E-01
-9.0702000103E-01	-1.8376654425E-01	1.0685136953E-01	7.0758531863E-02

Tablo 5.2. Bilek Döndürme AR Katsayıları

a_1	a_2	a_3	a_4
-1.6782944886E+00	2.6520971170E-01	6.1145966890E-01	-1.9385527057E-01
-1.6668844824E+00	2.9128419000E-01	5.7313242335E-01	-1.8845705196E-01
-1.7461513174E+00	3.6548327314E-01	6.5294178040E-01	-2.6558300438E-01
-1.3025860960E+00	-1.1446867276E-01	3.2965108775E-01	1.0345283121E-01
-1.6766072277E+00	3.8765627083E-01	3.9051372035E-01	-9.2938484190E-02
-1.6251298068E+00	2.3540441738E-01	6.1427725759E-01	-2.1594742888E-01
-1.5167089140E+00	1.8555941225E-01	3.1414453378E-01	2.4623479891E-02
-1.7621534596E+00	3.4742859380E-01	6.9550974971E-01	-2.7184063596E-01
-1.8006326425E+00	4.9785971212E-01	4.9213001705E-01	-1.862514748E-01
-1.6311482325E+00	2.3110173842E-01	5.4244144339E-01	-1.3354386907E-01
-1.6576312368E+00	2.5007500734E-01	6.2423480450E-01	-2.0160357415E-01
-1.7242866413E+00	3.3098455016E-01	6.2193987179E-01	-2.2082391743E-01

Tablo 5.3. Kavrama AR Katsayıları

a_1	a_2	a_3	a_4
-1.7628620203E+00	3.3509014048E-01	7.7903685144E-01	-3.3784912048E-01
-1.7147937194E+00	3.8440939067E-01	5.3351277970E-01	-1.9138849529E-01
-1.7847137462E+00	4.5364101095E-01	5.9422588725E-01	-2.5489654237E-01
-1.6969885901E+00	4.2314270627E-01	3.5911528476E-01	-7.5380301477E-02
-1.7920895718E+00	4.5149447486E-01	6.3829194507E-01	-2.8863118420E-01
-1.8890499350E+00	5.9260677875E-01	5.9514623768E-01	-2.9097552091E-01
-1.9145207019E+00	6.3425417746E-01	5.8029881792E-01	-2.9383511415E-01
-1.6212407393E+00	2.9392652232E-1	4.3690521059E-01	-1.0108151881E-01
-1.7480720586E+00	4.7166348138E-01	4.8403544742E-01	-1.9144358599E-01
-1.7411912333E+00	3.7700625993E-01	5.8395543349E-01	-2.0921690180E-01
-1.5918863811E+00	1.3577361182E-01	6.4716547454E-01	-1.8351180178E-01
-1.6465979555E+00	2.8133125987E-01	5.0332032768E-01	-1.2983359683E-01

Tablo 5.4. Bilek Bükme AR Katsayıları

a_1	a_2	a_3	a_4
-1.7177802376E+00	1.7923506031E-01	8.6416402389E-01	-3.2248749080E-01
-2.1556195534E+00	1.0727194727E+00	4.1624856169E-01	-3.2769068977E-01
-2.1094966700E+00	9.4973859050E-01	5.2513090298E-01	-3.6229070932E-01
-1.9881745683E+00	7.1473677244E-01	6.3532171690E-01	-3.5710443516E-01
-2.0302231521E+00	8.7372247668E-01	3.9424159375E-01	-2.3394918091E-01
-2.0361573993E+00	7.2691468604E-01	7.5100323360E-01	-4.3777890206E-01
-2.1158109897E+00	9.9905398463E-01	4.4321123616E-01	-3.2188584227E-01
-2.0156995333E+00	8.2924206776E-01	5.1270155057E-01	-3.1983913478E-01
-1.9381246455E+00	6.7198915940E-01	6.0503315396E-01	-3.2973071443E-01
-2.1954326032E+00	1.1464640424E+00	3.7764439479E-01	-3.2738111062E-01
-2.013877545E+00	7.1353238125E-01	7.0211842794E-01	-3.9699016671E-01
-1.9626837954E+00	6.7939541286E-01	6.1023028541E-01	-3.2306485047E-01

Tablo 5.5. Dirsek Açma AR Katsayıları

a_1	a_2	a_3	a_4
-2.2914128854E+00	1.4157880401E+00	1.5643946565E-01	-2.7576257484E-01
-2.2236665563E+00	1.1925184658E+00	3.7804077194E-01	-3.4387288667E-01
-2.5605990742E+00	2.1284516167E+00	-4.7985836909E-01	-8.5217132090E-02
-2.1855094142E+00	1.1668151587E+00	3.5497629656E-01	-3.0421272700E-01
-2.1335049591E+00	1.0162656173E+00	4.8782226231E-01	-3.6814453276E-01
-2.3205685494E+00	1.3983241604E+00	2.4926231425E-01	-3.2406063327E-01
-2.2736460701E+00	1.3090069187E+00	3.0759887510E-01	-3.4078144846E-01
-2.1544811453E+00	1.0935288607E+00	3.8948655289E-01	-3.2407480865E-01
-2.2177809049E+00	1.1889682963E+00	4.0943813260E-01	-3.7655125553E-01
-2.3595552024E+00	1.5141832312E+00	1.3451033751E-01	-2.8448465373E-01
-2.3105310227E+00	1.4117335132E+00	2.0540127137E-01	-3.0442982559E-01
-2.0866797895E+00	9.1354732122E-01	5.1311636760E-01	-3.3697516577E-01

Tablo 5.6. Dirsek Kapama AR Katsayıları

a_1	a_2	a_3	a_4
-2.3542062185E+00	1.4851184597E+00	1.4528583345E-01	-2.7379972680E-01
-2.4726544348E+00	1.8385442000E+00	-2.0527237386E-01	-1.5744447000E-01
-2.4573193843E+00	1.7527869504E+00	-8.6343603770E-02	-2.0686584583E-01
-2.3926002892E+00	1.6307035562E+00	9.9958495213E-03	-2.4512770954E-01
-2.3730717435E+00	1.5655434941E+00	3.6312152113E-02	-2.2689045467E-01
-2.2351576581E+00	1.2796459202E+00	2.7017863040E-01	-3.1305935123E-01
-2.4544280426E+00	1.7943195725E+00	-1.2409796096E-01	-2.1258192259E-01
-2.6320357985E+00	2.2931298851E+00	-6.1287175187E-01	-4.5692505305E-02
-2.5094879915E+00	1.9042301538E+00	-2.0069102896E-01	-1.9150689238E-01
-2.3592941518E+00	1.5443889449E+00	7.4672604464E-02	-2.5650593923E-01
-2.5892689199E+00	2.1397847674E+00	-4.3889054107E-01	-1.1007795451E-01
-2.4751362659E+00	1.8528491083E+00	-2.1320267626E-01	-1.8249904129E-01

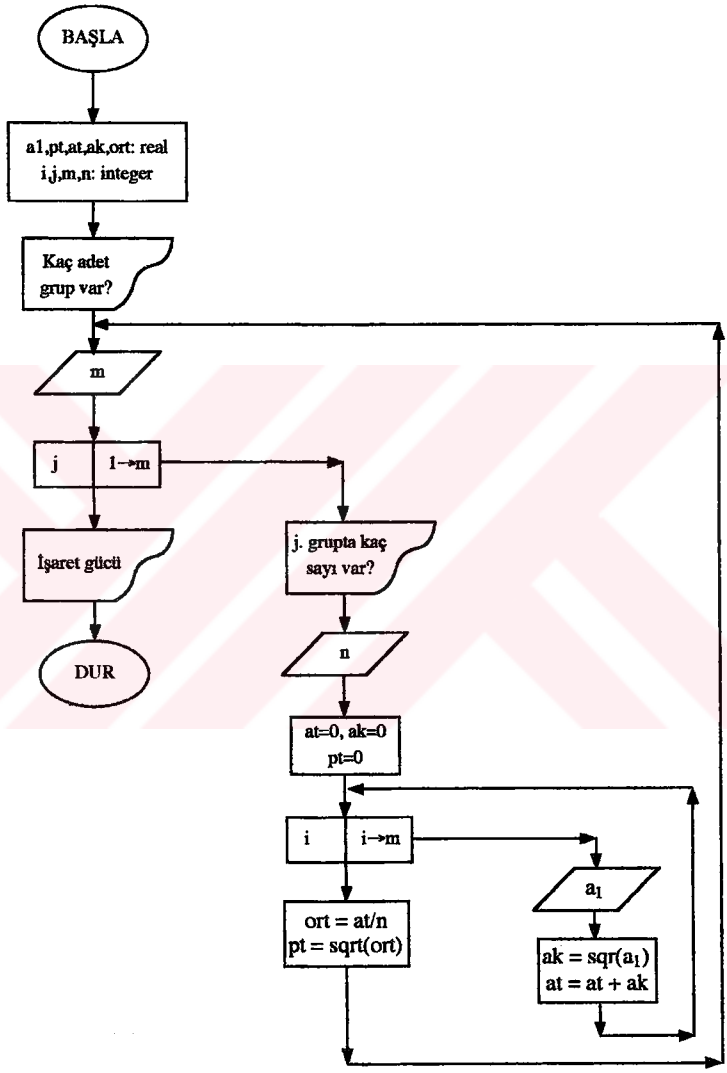
İşaret gücü (signal power) olarak adlandırılan ve (P) ile gösterilen ifade; bilinen güç kavramlarından farklı olup, AR parametrelerinin geometriksel ortalamasını (yada standart sapmasını) belirtir. Bu tanıma göre işaret gücü örneğin a_1 parametresine göre şu şekilde formülüze edilir.

$$P = \sqrt{\frac{a_{11}^2 + a_{12}^2 + a_{13}^2 + \ldots + a_{1n}^2}{n}}$$

Burada (n), denek sayısı olup bu çalışmada her hareket için 12 adet örnek alındığından n=12 olur. Altı farklı hareket için işaret güçleri ayrı ayrı hesaplanır. Şekil 5.2'deki akış diyağramında, işaret gücünün hesaplanması gösterilmektedir. İşaret gücü, yapay sinir ağı girişinde bir nevi kutup (bias) gibi davrandığı için etkilidir. İlk AR parametresi olan a_1 dışında sırayla a_2, a_3 ve a_4 lerin dahil olduğu durumlarda yukarda verilen eşitliğe benzer biçimde ilave edilerek hesaplanır. Her hareket için, parametrelerin tek tek ilave edilmesinde hesaplanan işaret güçleri Tablo 5.7'de verilmiştir.

Tablo 5.7

Parametre	P (D)	P (BD)	P (K)	P (BB)	P (DA)	P (DK)
a_1	0.927	1.655	1.744	2.028	2.237	2.443
a_1 , a_2	0.667	1.190	1.269	1.550	1.846	2.136
a_1 , a_2, a_3	0.518	0.918	1.013	1.234	1.517	1.738
a_1 , a_2, a_3 a_4	0.4821	0.8901	0.9488	1.1469	1.3278	1.5206

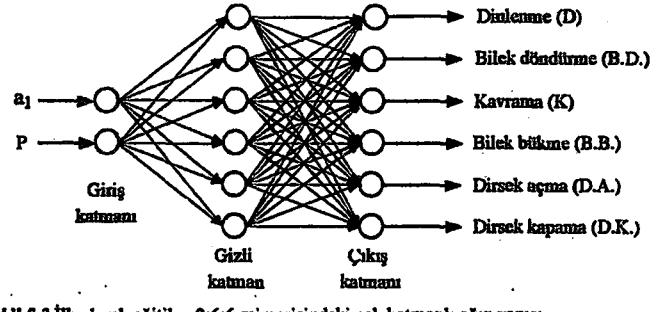


Şekil 5.2 İşaret gücü hesabı akış diyagramı

5.2 EMG İşaretlerini Sınıflamak İçin Kullanılan Çok Katmanlı Perceptron (idrak) Ağı

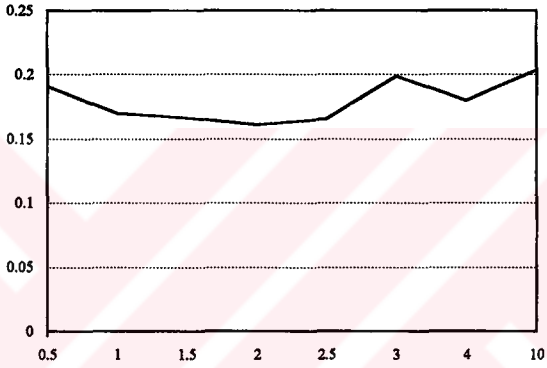
Örüntülerin tanınması ve miyoelektrik işaretin kontrol olarak sınıflandırılması konvansiyonel sınıflama teknikleri kullanılarak gerçekleştirilebilir. Bu çalışmada, biceps ve triceps kaslardan (tek kanallı) alınan miyoelektrik işaretlerden elde edilmiş AR parametreleri ile altı kol fonksiyonu arasında sınıflama yapabilmek için çok katmanlı perceptron (idrak) ağ yapısı kullanıldı. Sınıflayıcı olarak kullanılan YSA uygulaması, kullanıcı (protez takılan) öğrenmesinin miktarını azaltma gereği öncelikle motive edilir. Basit ileri-besleme (feed-forward) ağ iyi bir donanım uygulaması ile verilen yüksek tanıma oranları için gerekli potansiyele sahiptir. Bu çalışmadaki ilk denemelerde birinci zaman serileri parametresi -a_1- ve işaret güç seviyesi - p - özellik segmenti olarak kullanıldı.

Şekil 5.3'de gösterildiği gibi a_1 ve p özellikleri çok katmanlı perceptron için giriş verileridir. Bu sınıflayıcı ağ yapısı, analog girişler içeren problemler için idealdir ve bugüne kadar bir çok benzer uygulamalarda kullanılmıştır. Deneylerle tek katmanlı ağ yapısı kullanıldığında sınıflamanın başarısız olduğu gözlendiğinden dolayı; miyoelektrik işaret özelliğine en uygun sınıflayıcı ağ yapısı olarak çok katmanlı ağ yapısı seçilmiştir. Ağ, öğrenme işleyişi boyuca girişler ve arzu edilen (hedef) çıkışlar bilinecek şekilde "öğreticili" eğitme ile eğitilir. Öğrenme bilgisi, bütün giriş bilgileri arzu edilen çıkışları üretene kadar ağda var olur. Çok katmanlı perceptron'ların öğrenmesi, bağlantı ağırlıklarını ve işlem elemanı eşik değerlerini değiştirmek süretiyle ile olur. Bunun için "Hatanın geriye yayılması" (Back-propagation) algoritması kullanıldı. Algoritma adından da anlaşılacağı gibi, teknik bağlantı ağırlıklarını ayarlamak için çıkış hatasını ağın gerisine doğru yaymayı hedef alır. Burada çıkış hatası terimi, eğitme anında ağın o anki çıkış değeri ile istenen çıkış değeri arasındaki fark olarak bilinir. Çıkış düğümleri (üniteleri)'nin sigmoid transfer (eşik) fonksiyonları yüzünden, ağ çıkışları ikili (binary) formda olmaya eğilimlidir. Bundan dolayı sınıflama, herbir çıkış bilgi setininin belli bir parçasını temsil edecek şekilde oluşturulur. Mesela, eğer bilek döndürme hareketi (1) konumunda ise diğer hareketler (0) konumunda olacaktır. Sınıflamanın anlamsız olmaması için her giriş hareketi için, bir ağ çıkışı aktif olmalıdır.



Şekil 5.3 İlk olarak eğitilen 2:6:6 mimarisindeki çok katmanlı ağın yapısı

Kullanılan ağ mimarisi 2:6:6 düzeninde olup, iki giriş ünitesine, altı gizli ünite ve altı çıkış ünitesine sahiptir. Giriş ünitesine her hareketin 12 segmenti ve onlara ait işaret gücü 2x(6x12) şeklinde uygulanmıştır. Eğitme çalışmasında, öğrenme oranının (kazanç) farklı değerlerini denedikten sonra en iyi yakınsamanın ε =2 olduğu bulundu. Ayrıca gizli katmandaki ünite sayısının farklı değerleri için iterasyonlar yapılarak optimum sayı 6 olarak bulundu. Bulunan sonuç Gutierrez ve arkadaşlarının ortaya attıkları teoriye uygundur. Şekil 5.4'de ise öğrenme oranı ε'nun toplam hataya etkisi görülmektedir. Grafiğe dikkat edilirse en iyi sonuç; kazancın ε = 2 olduğu durumda gerçeklenmektedir.



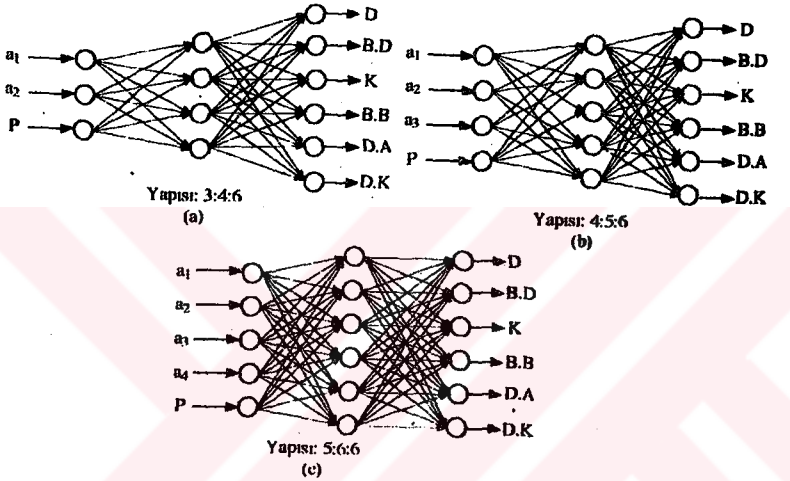
Şekil 5.4 3000 iterasyon için kazancın değişimi

Ağın çıkışı, sınıflandırılacak altı hareketi tanımlamaktadır. Çıkışda hedef (target) değerlerinin seti (12x6)x6 şeklindedir. YSA yapısından dolayı 0.1 ve 0.9 düzeyleri 0 ve 1 binary düzeylerine karşılık gelmektedir. Yapılan iterasyonlar sonucu ilk tanınan hareketler "dinlenme" ve "dirsek kapama" hareketleri olmuştur. İterasyonlar arttıkça tabiatıyla öğrenme (sınıflama) daha iyi oldu. YSA'nın sınıflama sonucu çıkışlarının binary düzende olması, kontrol aşamasında ilgili hareketleri sağlayacak servomotorları kolaylıkla kontrol etmeyi sağlar. Çok katmanlı perceptronda "Genelleştirilmiş Delta Kuralı" öğrenme metodu kullanıldı. Yazılım tez yazarı tarafından gerçekleştirilmiş olup, yazı dili Turbo PASCAL'dır. Programın tamamı başka araştırıcıların istifadesine sunmak amacıyla ekte verilmiştir. YSA programı genel amaçlı olup EKG aritmi teşhisi gibi çeşitli uygulamalarda da kullanılabilir.

5.3 Simülasyon Sonuçları

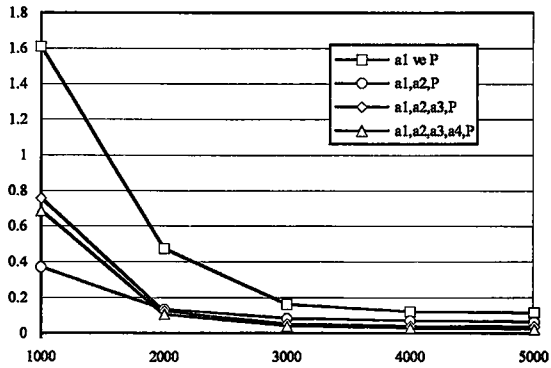
Şekil 5.3'deki girişte işaret gücü dışında sadece a_1 paremetresi uygulanan ağ mimarisine göre, yapılan 3000 iterasyonluk eğitme sonucu % 84'lük bir tanıma sağlanmıştır. Bu durumda, öğrenme oranı ε=2.0 ve momentum katsayısı α =0.01'dir. a_1 dışında diğer AR katsayı paremetreleri a_2, a_3, a_4 sırayla YSA girişine uygulandığında elde edilen çok katmanlı ağ mimarileri

Şekil 5.5'de gösterildiği gibidir. Şekle dikkat edilecek olunursa optimum sonuç veren "gizli katman" ünite sayılarının her AR paremetresini ilave ettikçe birer arttığı gözlenir. Şekil 5.5'deki üç ayrı ağ mimariside ekte verilen proğramda simule edilmiştir. Üç ağ mimarisi içinde momentum katsayısı (α = 0.1) ve öğrenme oranı (ε = 4) aynıdır. Hepside ekte verilen proğrama göre simüle edilmiştir.



Şekil 5.5 Diğer AR parametreleri (a_2, a_3, ve a_4) eklenince oluşan YSA mimarileri

Şekil 5.5 (a) daki yapıya göre giriş katmanına uygulanan a_1, a_2 ve P verilerinin matrisel yapısı (6x12)x3 biçimindedir. (b)'de a_1, a_2, a_3 AR katsayıları ve bunlara ait işaret gücü (P) den oluşan giriş verileri (6x12)x4 kadar, (c)'de ise a_1, a_2, a_3, $a_4$ve bunlara ait toplam işaret gücü P için (6x12)x5 kadardır. Çıkış değerleri her üçü içinde aynı sayıda olup, (6x12)x6 şeklindedir. (Altı farklı hareket sınıfı için) Bütün ağ mimarileri için değişik iterasyonlara göre elde edilen sınıflandırma hataları Şekil 5.6 deki grafikte karşılaştırılmıştır.



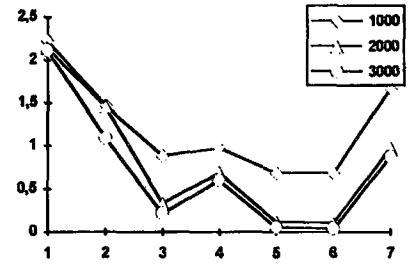
Şekil 5.6 Toplam hataların iterasyona göre değişimi

Yukarıdaki şekildende anlaşılacağı gibi AR parametreleri dahil edildikce sınıflandırma hatasının azaldığı, başka bir deyişle hareketleri ayırmaya yönelik öğrenme doğruluğunun arttığı bulundu. Oysa Kelly ve arkadaşları yaptıkları çalışmada, AR parametrelerinden sadece a_1'in yeterli kalacağını diğerlerinin ise öğrenmede fazla etkisinin olmayacağını ileri sürmüşlerdir. Ancak aşağıda Tablo 5.8 deki sonuçlardanda anlaşılacağı gibi a_2, a_3, a_4 ün öğrenmeye etkisi bariz bir şekilde belli olmaktadır.

Tablo 5.8

İterasyon Sayısı	a_1,p	a_1,a_2,p	a_1,a_2,a_3,p	a_1,a_2,a_3,a_4,p
1000	---	% 63.2	% 25.4	% 41.6
2000	% 52.8	% 86.5	% 87.6	% 89.4
3000	% 84	% 91.6	% 95	% 96.1
4000	% 87.8	% 93.1	% 93.8	% 97.2
5000	% 88.6	% 93.8	% 96.5	% 97.6

Tablo 5.8 de 1000, 2000, 3000, 4000 ve 5000 itersyon için sınıflanmadaki başarı yüzdeleri verilmiştir. Dördüncü paremetreden sonra tanıma yüzdesi fazla artmayacaktır. Zira yukardaki tabloya bakıldığında 3. paremetre ile 4. paremetrenin arasında tanıma yüzdesi farkı fazla değildir. (% 1'lik tanıma farkı olmaktadır.) Bu yüzden 4.parametreden sonraki parametreleri (a_5, a_6, ...) almaya gerek yoktur. Sınıflandırılan 6 hareket için tanıma yüzdesi en yüksek olan Şekil 5.4 (d)'deki YSA mimarisine göre (5:6:6 düzeninde, ε =4 ve α =0.1); değişik iterasyonlarda gizli katmandaki ünite sayısının sınıflandırma hatasına etkisi Şekil 5.6'da gösterilmiştir.



Şekil 5.6 Gizli katmandaki ünite sayısının etkisi

Grafiğe göre en küçük hata, gizli ünite sayısının 6 adet olduğu durumdur. Gizli ünite sayısını, giriş veya çıkış ünitesinin maksimum sayısından daha fazla olduğu anda (yani 6'dan büyük 7,8,..) hata büyümektedir. Şunu da unutmamak gerekir ki; gizli ünite sayısı arttıkça eğitme süreside azda olsa artmaktadır. İterasyonlar bir PC 486 SX'de yapılmıştır. 5000 iterasyon yaklaşık beş saat sürmüştür. Aynı iterasyon bir PC 486 DX (içinde matematik işlemci var) bilgisayarında sadece 11 dk. sürmüştür. İterasyon sayısı arttıkça öğrenme yüzdesi artmakta, diğer bir deyişle sınıflandırma hatası azalmaktadır. Aşağıda Tablo 5.8'de ise, tüm hareketlerin verilen 12 denek örüntüsünden ne kadarının tanındığı gösterilmektedir. 100 iterasyon için; Dinlenme (D) ve Dirsek Kapama (DK) örüntülerinin hepsi tanınmakta, Bilek Döndürme (BD) ile Kavrama (K) hareketleri tanınmamaktadır. Bilek Bükme (B) nin 4 tanesi, Dirsek Açmanın (DA) 5 tanesi ayırtedilebilmektedir. Buna karşın 2BB, 2DA ve 7DK örüntüleri yanlış tanınmaktadır. Dirsek açma hareketinin tamamı 500 iterasyonda, BB ve K hareketlerinin tamamı da 600 iterasyonda ayırtedilmektedir. Enson olarak Bilek döndürme hareketinin örüntüleri tanınmakta olup, ancak 2000 iterasyonda gerçeklenmektedir Bunun nedeni, BD ile K örüntü değerlerinin birbirine çok yakın olmasından kaynaklanmaktadır.

Şekil 5.2 de verilen 2:6:6 YSA mimarisi programında (ε =2, α =0.01), kayan noktalı aritmetiğinde bir taşma olmuştur. (Diğerlerinde olmamıştır) Bu hata, momentum katsayısının çok küçük seçilmesinden dolayı eşik fonksiyonu olarak kullanılan sigmoid fonksiyonunda; y(x)=1/1+exp(-x) x'in değeri çok çok büyük sayı olmaktadır. Bu problemi gidermek için eşik fonksiyonuna sınırlama getirildi. Bu durumda ekte verilen YSA programında sigmoid fonksiyonunun tanımlandığı kısım şu şekilde olacaktır;

```
if x>= 1000000000
        then y:= 1
        else if x<= -1000000000
                then y:= 0
                else y:= 1/ 1+exp(-x)
```

Tablo 5.8

İterasyon	D	BD	K	BB	DA	DK	Yanlış Tanıma
100	12	0	0	4	5	12	2BB,3DA,7DK
200	12	0	0	6	8	12	8BB,4DA,4DK
300	12	1	1	8	9	12	5K,2BD,4DA,3DK
400	12	1	6	9	11	12	10K,2BD,3DA,1DK
500	12	1	6	11	12	12	11K,1BB,1DA
600	12	1	11	12	12	12	11K
800	12	1	12	12	12	12	11K
1000	12	2	12	12	12	12	10K
1200	12	3	12	12	12	12	9K
1500	12	8	12	12	12	12	4K
1800	12	11	12	12	12	12	1K
2000	12	12	12	12	12	12	0

Yapılan bu ekle sadece problem giderilmemiş olup aynı zamanda N.B. Karayiannis ve arkadaşlarının söz ettikleri "hızlı öğrenme" (fast learning) avantajınada sahip olunmaktadır (Karayiannis et al, 1992). Yalnız bu durumda çok azda olsa öğrenme hatası artabilir.

Bu çalışmada YSA giriş değerleri olan veriler, denekten alınan verilerin kendisi olup herhangi bir eksiltme olmamıştır. Sadece virgülden sonra üç hane alınacak şekilde yuvarlatma yapılmıştır. Altı farklı hareketin AR parametrelerinden örtüşen değerler atılmamıştır. Şayet örtüşen değerler çıkarılırsa öğrenme yüzdesinin dahada artacağı aşikardır.

Ayrıca yapılan çalışmada ilk defa 6 hareket % 96.1 (3000 iterasyon için) gibi yüksek doğrulukta ayırt edilmiştir. 1. Bölümde de bahsedildiği gibi bundan önce en yüksek tanıma oranı yine yaklaşık 3000 iterasyonda ama 4 hareket için tanıma oranı % 92 olmuştur. Şimdilik simülasyon olarak gerçekleştirilen bu çalışma, tıp doktoru ve makina mühendisleriyle ortaklaşa pratik hale getirilebilir. Bilgisayarlarda ve diğer teknolojik gelişmelerle 6 temel hareket daha spesifik olarak incelenebilinir. Yani sadece dirsek kapama yerine, yavaş kapama, orta ve hızlı kapama gibi hareketlerde öğretilebilir. Son yıllarda popüler olan biorobotlar içinde kullanılabilir. Yani insan beyninden verilen komutla bir robotun kol hareketleri kontrol edilebilir.

KAYNAKLAR

1- Alderson, S., "The electric arm." in P.E. Klopsteg and P.D Vilson, Human Limbs and their Substitutes New York: Mc Graw-Hill, 1954, ch.13; reprinted by Hafner, 1969

2- Almstrom, Herberts, P. and Korner, L., "Experiences with swedish multifunction prosthetic hands controlled by pattern recognition of multiple myoelectric signals," Int. Orthopased vol. 5, pp.15-21 1981

3- Childress, "A myoelectric three state controller using rate sensivity," in Proc. 8th. ICMBE, Chicago, IL, pp.4-5, 1969

4- Culloch, S. M. and Pitts, W., "A logical calculus of the ideas immanent in nervous activity," Bulletin of Mathematical Biophysics, vol.5, pp. 115-133, 1943

5- Dorcas and Scott, R.N, "A three state myoelectric control," Med. Biol. Eng., vol. 4, pp. 367-372, 1966

6- Graupe, "Functional separation of EMG signals via ARMA identification methods for prosthesis control purposes," IEEE Trans. Syst. Man Cybern., vol. SMC-5, pp.252-259 Mar. 1975

7- Graupe, D., Salahi, J. and Kohn, K.H., "Multifonction prosthesis and orthosis control via microcomputer identification of temporal pattern differences in single-site myoelectric signals," J. Biomed. Eng., vol. 4, pp.17-22, 1982

8- Graupe, D., Salahi, J. and Zhang, D. "Stochastic analysis of myoelectric temporal signatures for multifonctional single-site activation of prostheses and orthoses," J. Biomed. Eng., vol. 7, no. 1, pp.18-29, 1985

9- Graupe, Magnuses, J. and Beex, A., "Microprocessor system for multifonctional control of upperlimbs prostheses," IEEE Trans. Automat. Control., vol. AC-23, pp. 538-544, Aug. 1978

10- Grossberg, "Adaptive pattern classification and üniversal recoding:I. Parallel development and coding of neural feature dedectors," Biological Cybernnetics vol. 23, pp. 121-134, 1976

11- Gutierrez, Wang, J. and Grandin, R.D., "Estimating hidden units for two layer perceptrons," Proceeding of the 1st. International Conference on Artificial Neural Networks," London, U.K, pp:120-124, October, 1989

12- Hebb, O., "The Organization of Behavior," Introduction and Chapter 4, pp. XI-XIX, and, 60-78, Wiley, New York, 1949

13- Hopfield, J., "Neural networks and physical systems with emergent collective computational abilities," Proceedings of the National Academy of Sciences, vol. 79, pp. 2554-2558, 1982

14- Hudgins, Parker, P., Scott, R.N., "A New Strategy for Multifonction Myoelectric Control," IEEE Trans. Biomed. Eng. vol. 40, no:1, January 1993

15- Jacobsen, Knutti, D.F., Johnson, R.T., Sears, H.H., "Development of the Utah Artificial Arm", IEEE Trans. Biomed. Eng. vol. BME-29, pp. 249, April 1982

16- Karaağaç, "EMG işaretlerine ait bazı parametrelerin PC yardımıyla bulunması," Yüksek Lisans Tezi İ.T.Ü sayfa:12-16 Şubat 1989

17- Karayiannis, B. ve Venetsanapoulos, A. N., "Fast Learning Algorithms for Neural Networks," IEEE Trans. an Circuits and Systems-II: Analog and Digital Signal Processing, vol. 39, no:7, Jully, 1992

18- Kelly, Parker, P. and Scott, R.N., "The application of neural networks to myoelectric, signal analysis: A preliminary study," IEEE Trans. Biomed. Eng., vol. BME-37, pp. 221-227, Mar.1990

19- Kohen, "Biomedical Signal Processing," CRC press vol:1, pp. 3-5, and 81-89, 1986

20- Kohonen, "Self-Organized formation of topologically correct feature maps," Biological Cybernetics, vol. 43, pp. 59-69, 1982

21- Korürek. "Tıp Elektroniğinde Kullanılan Kuvvetlendiriciler ve Dönüştürücüler," Ders Notları, İ.T.Ü 1991

22- Kuyucu "Elektromiyografik İşaretlerin Değerlendirilmesi," Yüksek Lisans tezi İ.T.Ü Fen Bil. Enst. Haziran 1989

23- Lippmann, "An Introduction to computing with Neural Nets," IEEE ASSP Magazine, pp. 4-22, April, 1987

24- Lyman and Freedy, A., "A pattern analysis and sequential decision methods for sensory feed back and control of upper limb prostheses," Res. Prop. to the Nat. Sci. Found., 1976

25- Lyman, A. Freedy, and H. Zadaca, "Studies and development of heuristic end-point control for artificial upper limbs," UCLA Biotechnol. Lab., Tech. Rep. 54, Oct 1974

26- Minsky and S. Papert, "Perceptrons," Cambridge, MA: MIT Press, Introduction, pp. 1-20 and 73, 1969

27- Naylor, C. J., "Artificial Neural Networks Review," University of Nottingham version 1.1 (U.K) 12 December 1990

28- Nielsen, R. H., "Neurocomputing" Addison Wesley, pp. 28-30, 1989

29- Orfanidis, J., "Optimum Signal Processing," second edition, Mc Millan Pub. pp. 58-66, 1988

30- Pao, Y. H., "Adaptive Pattern Recognition and Neural Networks," Addison-Wesley pp:130, 1989

31- Parker and Scott R.N., "Myoelectric control of prosthesis," CRC Crit. Rev. Biomed. Eng., vol. 13, issue 4 pp. 283-310, 1986

32- Pastacı "Mühendisliğin Tıptaki Uygulamaları," Ders Notları, Y.T.Ü 1989

33- Richard, P., Gander, R., Parker, P. and Scott, R.N. "Multistate myoelectric control: The feasibility of 5-state control," J. Rehab. R&D, vol. 20, BPR 10-38, pp. 84-86, 1983

34- Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain," Psychological Review vol.65, pp. 386-408, 1958

35- Rumelhart, E., Hinton, G.E. and R.J. Williams, "Learning representations by back-propagating errors," Nature 323: 533-536, 1986

36- Rumelhart, E., Hinton, G.E. and Williams, R.J., "Learning internal representations by error propagation," Parellel Distributed Processing Explorations in the Microstructures of Cognition, vol. 1, D.E. Rumelhart and J.L. Mc Clalland (Eds.) Cambridge, MA: MIT Press, pp:318-362, 1986

37- Saridis and Stephanou, H.E., "A hierarchical approach to the control of a prosthetic arm," IEEE Trans. Syst. Man Cybern., vol. SMC-7, pp. 407-420, June 1977

38- Saridis, G. N. and Gotee, T., "EMG Pattern Analysis and Classification for a Prosthetic Arm," IEEE Trans. Biomed. Eng. vol. BME-29 no:6 pp.403-412, June 1982

39- Schmeild, "The I.N.A.I.L. experience fitting upper-limb dysmelia patients with myoelectric control," Bull. Prosth. Res., vol. BPR 10-27, pp. 17-42, Spring 1977

40- Schmeild, "The INAIL-CECA Prostheses, "Orthotics Prosth., vol. 27, no. 1 pp. 6-12, 1973

41- Scott and Parker, P.A., "Myoelectric prosthesis: State of the art," J. Med. Eng. Technol., vol. 12, pp. 143-151, 1988

42- Scott, Parker, P.A., Dunfield, V.A., "Myoelectric Control," IEE Medical Electronics Monographs. Ed. D. W. Hill, B.W. Watson, Peter Peregrinus Ltd., Exeter 1977

43- Vodovnik, J., Kreifeldt, R., Caldwell, L. Green, E. Silgalis and P.Crarg, "Some topics an myoelectric control of orthotic/prosthetic systems," Rep. EDC 4-67-17, Case Western Reserve University, Cleveland, OH, 1967

44- Widrow and Lehr, M. A., "Perceptron, Madaline, and Back propagation," Proceedings of the IEEE vol.78, no:9, pp. 1417-1439, September, 1990

45- Widrow, B. and Hoff, M. E., "Adaptive switching circuits," IRE WESCON Convention Record, New York: IRE pp. 96-104, 1960

46- Williams, T.W., "Practical methods for controlling powered upper-extremity prostheses," Assitive Technol., vol. 2, no.1, pp. 3-18, 1990

47- Wirta, Taylor, D.R., Finley, F.R., "Pattern-recognition arm prosthesis: a historical perspective- a final report," Bulletin of Prosthesis Research, Fall 1978

```
PROGRAM EMG_Net;
{$N+,E+}
CONST
Layers_Max   = 3;        { Maximum katman sayısı }
Nodes_Max    = 6;        { Herbir katmandaki maximum düğüm sayısı }
Outputs_Max  = 72;       { Giriş bileşenlerinin sayısı }
Input_Nodes  = 5;        { Giriş katmanı düğümleri sayısı }
Output_Nodes = 6;        { Çıkış katmanı düğümleri sayısı }
Hidden_Nodes = 6;        { Gizli katman düğümleri sayısı }
Epsilon      = 4;
Alpha        = 0.1;
Iterative_Max  = 3000;   { Maximum iterasyon sayısı }
Every        = 1000;     { Her iterasyondan sonraki çıkış sonuçları }

LABEL 10,20,30;

VAR
i, ii, j, count,dcount,tt, isylla, num : integer;
eps, alp, err, error, terr, terror, total_rate: double;
Layers, Inputs, Outputs : integer;
Node         : array [0..Layers_Max-1] of integer;
matrix    : array [0..Outputs_Max-1, 0..(Input_Nodes+Output_Nodes-1)] of double; matrix_tmp: array
[0..trunc(Iterative_Max/Every), 0..Outputs_Max-1,
                                    0..(Input_Nodes+Output_Nodes-1)] of double;


des          : double;
rate         : array [0..Outputs_Max-1] of double;
out          : array [0..Layers_Max-1, 0..Nodes_Max-1] of double;
der          : array [0..Layers_Max-1, 0..Nodes_Max-1] of double;
theta   : array [0..Layers_Max-1, 0..Nodes_Max-1] of double; dtheta  : array
[0..Layers_Max-1, 0..Nodes_Max-1] of double;
weight  : array [0..Layers_Max-1, 0..Nodes_Max-1, 0..Nodes_Max-1] of double; dweight :
array[0..Layers_Max-1, 0..Nodes_Max-1, 0..Nodes_Max-1] of double; dess: array [0..Outputs_Max-
1,0..Output_Nodes-1] of double;

st1: string[6];
st2, st3, st4: string[9];
outfile : text;
weightfile : file of double;

procedure Wrand;
{ Düğüm ofsetlerini başlatır & Programın başlangıcındaki ağırlıklar }
var
li, ni, ni_end, no, no_end : integer;
drand48, power: double;
begin
    power:=1.0;
    for li:=1 to 31 do
            power:=power*2;
    power:=power-1;
    { Rastgele offsetler }
    for li:=1 to Layers_Max-1 do
    begin
        ni_end:=node[li];
```

```
        for ni:=0 to ni_end-1 do
        begin
            drand48:=Random(30)/100;
            theta[li,ni]:= drand48;
            dtheta[li,ni]:=0;
        end;
    end;

    { Rastgele ağırlıklar }
    for li:=1 to Layers_Max-1 do
    begin
        ni_end:=node[li];
        no_end:=node[li-1];
        for ni:=0 to ni_end-1 do
                for no:=0 to no_end-1 do
                begin
                    drand48:=Random(100)/100;
                    weight[li,ni,no]:=drand48;
                    dweight[li,ni,no]:=0;
                end;
    end;
end;  { procedure WRAND }


procedure Initial;
{   Her iterasyondan önce herbir çıkış düğümünü sıfırla  }
VAR
l, n, n_end: integer;
begin
    for l:= 0 to Layers-1 do
    begin
        n_end:= node[l];
        for n:= 0 to n_end-1 do
                out[l,n]:= 0;
    end;
end;            { procedure INITIAL }


procedure yread ( isylla: integer );
{   Giriş örneklerini oku ( 72 farklı giriş bileşeni için ) }
begin
    case isylla of
            0: begin out[0,0]:=-0.968; out[0,1]:=-0.154; out[0,2]:=0.134; out[0,3]:=0.074; out[0,4]:=0.4821 end;
            1: begin out[0,0]:=-1.049; out[0,1]:=0.0024; out[0,2]:=0.041; out[0,3]:=0.128; out[0,4]:=0.4821 end;
            2: begin out[0,0]:=-0.924; out[0,1]:=-0.147; out[0,2]:=0.269; out[0,3]:=-0.077; out[0,4]:=0.4821 end;
            3: begin out[0,0]:=-0.945; out[0,1]:=-0.133; out[0,2]:=0.024; out[0,3]:=0.133; out[0,4]:=0.4821 end;
            4: begin out[0,0]:=-0.869; out[0,1]:=-0.164; out[0,2]:=0.053; out[0,3]:=0.086; out[0,4]:=0.4821 end;
            5: begin out[0,0]:=-0.997; out[0,1]:=-0.215; out[0,2]:=0.342; out[0,3]:=-0.007; out[0,4]:=0.4821 end;
            6: begin out[0,0]:=-0.863; out[0,1]:=-0.213; out[0,2]:=0.127; out[0,3]:=0.088; out[0,4]:=0.4821 end;
            7: begin out[0,0]:=-0.754; out[0,1]:=-0.279; out[0,2]:=0.148; out[0,3]:=0.118; out[0,4]:=0.4821 end;
            8: begin out[0,0]:=-0.931; out[0,1]:=-0.224; out[0,2]:=0.303; out[0,3]:=-0.078; out[0,4]:=0.4821 end;
            9: begin out[0,0]:=-1.001; out[0,1]:=-0.095; out[0,2]:=0.074; out[0,3]:=0.146; out[0,4]:=0.4821 end;
            10: begin out[0,0]:=-0.882; out[0,1]:=-0.002; out[0,2]:=-0.013; out[0,3]:=0.150; out[0,4]:=0.4821 end;
            11: begin out[0,0]:=-0.907; out[0,1]:=-0.184; out[0,2]:=0.106; out[0,3]:=0.071; out[0,4]:=0.4821 end;
            12: begin out[0,0]:=-1.678; out[0,1]:=0.265; out[0,2]:=0.611; out[0,3]:=-0.194; out[0,4]:=0.8901 end;
```

```
13: begin out[0,0]:=-1.666; out[0,1]:=0.291; out[0,2]:=0.573; out[0,3]:=-0.188; out[0,4]:=0.8901 end;
14: begin out[0,0]:=-1.746; out[0,1]:=0.365; out[0,2]:=0.653; out[0,3]:=-0.265; out[0,4]:=0.8901 end;
15: begin out[0,0]:=-1.303; out[0,1]:=-0.114; out[0,2]:=0.329; out[0,3]:=0.103; out[0,4]:=0.8901 end;
16: begin out[0,0]:=-1.676; out[0,1]:=0.387; out[0,2]:=0.390; out[0,3]:=-0.093; out[0,4]:=0.8901 end;
17: begin out[0,0]:=-1.625; out[0,1]:=0.235; out[0,2]:=0.614; out[0,3]:=-0.216; out[0,4]:=0.8901 end;
18: begin out[0,0]:=-1.516; out[0,1]:=0.185; out[0,2]:=0.314; out[0,3]:=0.0246; out[0,4]:=0.8901 end;
19: begin out[0,0]:=-1.762; out[0,1]:=0.347; out[0,2]:=0.695; out[0,3]:=-0.271; out[0,4]:=0.8901 end;
20: begin out[0,0]:=-1.800; out[0,1]:=0.498; out[0,2]:=0.492; out[0,3]:=-0.186; out[0,4]:=0.8901 end;
21: begin out[0,0]:=-1.631; out[0,1]:=0.231; out[0,2]:=0.542; out[0,3]:=-0.133; out[0,4]:=0.8901 end;
22: begin out[0,0]:=-1.657; out[0,1]:=0.250; out[0,2]:=0.624; out[0,3]:=-0.201; out[0,4]:=0.8901 end;
23: begin out[0,0]:=-1.724; out[0,1]:=0.331; out[0,2]:=0.622; out[0,3]:=-0.221; out[0,4]:=0.8901 end;
24: begin out[0,0]:=-1.763; out[0,1]:=0.335; out[0,2]:=0.779; out[0,3]:=-0.338; out[0,4]:=0.9488 end;
25: begin out[0,0]:=-1.714; out[0,1]:=0.384; out[0,2]:=0.534; out[0,3]:=-0.191; out[0,4]:=0.9488 end;
26: begin out[0,0]:=-1.784; out[0,1]:=0.453; out[0,2]:=0.594; out[0,3]:=-0.255; out[0,4]:=0.9488 end;
27: begin out[0,0]:=-1.697; out[0,1]:=0.423; out[0,2]:=0.359; out[0,3]:=-0.075; out[0,4]:=0.9488 end;
28: begin out[0,0]:=-1.792; out[0,1]:=0.451; out[0,2]:=0.638; out[0,3]:=-0.288; out[0,4]:=0.9488 end;
29: begin out[0,0]:=-1.889; out[0,1]:=0.592; out[0,2]:=0.595; out[0,3]:=-0.291; out[0,4]:=0.9488 end;
30: begin out[0,0]:=-1.914; out[0,1]:=0.634; out[0,2]:=0.580; out[0,3]:=-0.294; out[0,4]:=0.9488 end;
31: begin out[0,0]:=-1.621; out[0,1]:=0.294; out[0,2]:=0.437; out[0,3]:=-0.101; out[0,4]:=0.9488 end;
32: begin out[0,0]:=-1.748; out[0,1]:=0.472; out[0,2]:=0.484; out[0,3]:=-0.191; out[0,4]:=0.9488 end;
33: begin out[0,0]:=-1.741; out[0,1]:=0.377; out[0,2]:=0.584; out[0,3]:=-0.209; out[0,4]:=0.9488 end;
34: begin out[0,0]:=-1.600; out[0,1]:=0.138; out[0,2]:=0.647; out[0,3]:=-0.183; out[0,4]:=0.9488 end;
35: begin out[0,0]:=-1.646; out[0,1]:=0.281; out[0,2]:=0.503; out[0,3]:=-0.130; out[0,4]:=0.9488 end;
36: begin out[0,0]:=-1.718; out[0,1]:=0.179; out[0,2]:=0.864; out[0,3]:=-0.322; out[0,4]:=1.1469 end;
37: begin out[0,0]:=-2.155; out[0,1]:=1.072; out[0,2]:=0.416; out[0,3]:=-0.328; out[0,4]:=1.1469 end;
38: begin out[0,0]:=-2.109; out[0,2]:=0.950; out[0,2]:=0.525; out[0,3]:=-0.362; out[0,4]:=1.1469 end;
39: begin out[0,0]:=-1.988; out[0,1]:=0.715; out[0,2]:=0.635; out[0,3]:=-0.357; out[0,4]:=1.1469 end;
40: begin out[0,0]:=-2.030; out[0,1]:=0.874; out[0,2]:=0.394; out[0,3]:=-0.234; out[0,4]:=1.1469 end;
41: begin out[0,0]:=-2.036; out[0,1]:=0.727; out[0,2]:=0.751; out[0,3]:=-0.438; out[0,4]:=1.1469 end;
42: begin out[0,0]:=-2.116; out[0,1]:=0.999; out[0,2]:=0.443; out[0,3]:=-0.322; out[0,4]:=1.1469 end;
43: begin out[0,0]:=-2.015; out[0,1]:=0.829; out[0,2]:=0.513; out[0,3]:=-0.320; out[0,4]:=1.1469 end;
44: begin out[0,0]:=-1.938; out[0,1]:=0.672; out[0,2]:=0.605; out[0,3]:=-0.330; out[0,4]:=1.1469 end;
45: begin out[0,0]:=-2.195; out[0,1]:=1.146; out[0,2]:=0.378; out[0,3]:=-0.327; out[0,4]:=1.1469 end;
46: begin out[0,0]:=-2.014; out[0,1]:=0.713; out[0,2]:=0.702; out[0,3]:=-0.397; out[0,4]:=1.1469 end;
47: begin out[0,0]:=-1.962; out[0,1]:=0.679; out[0,2]:=0.610; out[0,3]:=-0.323; out[0,4]:=1.1469 end;
48: begin out[0,0]:=-2.291; out[0,1]:=1.416; out[0,2]:=0.156; out[0,3]:=-0.276; out[0,4]:=1.3278 end;
49: begin out[0,0]:=-2.223; out[0,1]:=1.192; out[0,2]:=0.378; out[0,3]:=-0.344; out[0,4]:=1.3278 end;
50: begin out[0,0]:=-2.280; out[0,1]:=2.128; out[0,2]:=0.480; out[0,3]:=-0.085; out[0,4]:=1.3278 end;
51: begin out[0,0]:=-2.185; out[0,1]:=1.186; out[0,2]:=0.355; out[0,3]:=-0.304; out[0,4]:=1.3278 end;
52: begin out[0,0]:=-2.133; out[0,1]:=1.016; out[0,2]:=0.488; out[0,3]:=-0.368; out[0,4]:=1.3278 end;
53: begin out[0,0]:=-2.320; out[0,1]:=1.398; out[0,2]:=0.249; out[0,3]:=-0.324; out[0,4]:=1.3278 end;
54: begin out[0,0]:=-2.273; out[0,1]:=1.309; out[0,2]:=0.308; out[0,3]:=-0.341; out[0,4]:=1.3278 end;
55: begin out[0,0]:=-2.154; out[0,1]:=1.094; out[0,2]:=0.389; out[0,3]:=-0.324; out[0,4]:=1.3278 end;
56: begin out[0,0]:=-2.218; out[0,1]:=1.189; out[0,2]:=0.409; out[0,3]:=-0.376; out[0,4]:=1.3278 end;
57: begin out[0,0]:=-2.350; out[0,1]:=1.514; out[0,2]:=0.134; out[0,3]:=-0.284; out[0,4]:=1.3278 end;
58: begin out[0,0]:=-2.310; out[0,1]:=1.412; out[0,2]:=0.205; out[0,3]:=-0.304; out[0,4]:=1.3278 end;
59: begin out[0,0]:=-2.100; out[0,1]:=0.914; out[0,2]:=0.513; out[0,3]:=-0.337; out[0,4]:=1.3278 end;
60: begin out[0,0]:=-2.354; out[0,1]:=1.485; out[0,2]:=0.145; out[0,3]:=-0.274; out[0,4]:=1.5206 end;
61: begin out[0,0]:=-2.472; out[0,1]:=1.838; out[0,2]:=-0.205; out[0,3]:=-0.157; out[0,4]:=1.5206 end;
62: begin out[0,0]:=-2.457; out[0,1]:=1.753; out[0,2]:=-0.086; out[0,3]:=-0.207; out[0,4]:=1.5206 end;
63: begin out[0,0]:=-2.392; out[0,1]:=1.640; out[0,2]:=0.010; out[0,3]:=-0.245; out[0,4]:=1.5206 end;
64: begin out[0,0]:=-2.373; out[0,1]:=1.566; out[0,2]:=0.036; out[0,3]:=-0.227; out[0,4]:=1.5206 end;
65: begin out[0,0]:=-2.237; out[0,1]:=1.280; out[0,2]:=0.270; out[0,3]:=-0.313; out[0,4]:=1.5206 end;
66: begin out[0,0]:=-2.454; out[0,1]:=1.794; out[0,2]:=-0.124; out[0,3]:=-0.213; out[0,4]:=1.5206 end;
67: begin out[0,0]:=-2.632; out[0,1]:=2.290; out[0,2]:=-0.612; out[0,3]:=-0.046; out[0,4]:=1.5206 end;
```

```
            68: begin out[0,0]:=-2.510; out[0,1]:=1.904; out[0,2]:=-0.200; out[0,3]:=-0.191; out[0,4]:=1.5206 end;
            69: begin out[0,0]:=-2.360; out[0,1]:=1.545; out[0,2]:=0.075; out[0,3]:=-0.256; out[0,4]:=1.5206 end;
            70: begin out[0,0]:=-2.590; out[0,1]:=2.140; out[0,2]:=-0.439; out[0,3]:=-0.110; out[0,4]:=1.5206 end;
            71: begin out[0,0]:=-2.475; out[0,1]:=1.853; out[0,2]:=-0.213; out[0,3]:=-0.162; out[0,4]:=1.5206 end;
    end;
end;  { procedure YREAD }

function sigmoid (x:double):double;
        {sigmoid fonksiyonu}
var
y:double;
begin
    y:=1/(1+exp(-x));
    sigmoid:=y;
end;  { function SIGMOID }


procedure wforward;
{   wforward propagation & ileri yayılım }
var
li, ni, ni_end, no, no_end: integer;
x, th: double;
begin
    for li:=1 to Layers-1 do
    begin
        ni_end:= node[li];
        no_end:= node[li-1];
        for ni:= 0 to ni_end-1 do
        begin
            x:= 0;
            for no:= 0 to no_end-1 do
                    x:= x + weight[li,ni,no] * out[li-1,no];
            out[li,ni]:= sigmoid( x-theta[li,ni] );
        end;
    end;
end;   { procedure FORWARD }


function back: double;
{   back propagation & geriye yayılım }
var
li, ni, ni_end, no, no_end: integer; e, err, x,
d, w, y: double;
begin
    err:=0;
    for ni:=0 to outputs-1 do
    begin
        y:= out[2,ni];
        des:= dess[isylla,ni];
        e:= des - y;
        der[2,ni]:= e*y*(1-y);
        err:=err+ e*e;
    end;

    for li:=2 downto 1 do
```

```
    begin
        no_end:= node[li-1];
        ni_end:= node[li];
        for no:= 0 to no_end-1 do
        begin
            x:= 0;
            y:= out[li-1,no];
            for ni:= 0 to ni_end-1 do
            begin
                d:= der[li,ni];
                w:= weight[li,ni,no];
                x:= x + d*w;
            end;
            der[li-1,no]:= y*(1-y)*x;
        end;
    end;
    back:= 0.5*err;
end; { function BACK }


procedure learning (alp, eps: double );
{ update offsets & weights }
var
li, lo, ni, ni_end, no, no_end: integer;
di, yo, ew, et: double;
begin
    for li:= 1 to layers-1 do
    begin
        ni_end:= node[li];
        for ni:= 0 to ni_end-1 do
        begin
            et:= der[li,ni];
            dtheta[li,ni]:= -eps*et + alp*dtheta[li,ni];
            theta[li,ni]:= theta[li,ni] + dtheta[li,ni];
        end;
            end;
    for li:=1 to layers-1 do
    begin
        lo:= li-1;
        ni_end:= node[li];
        no_end:= node[lo];
        for ni:= 0 to ni_end-1 do
        begin
            di:= der[li,ni];
            for no:= 0 to no_end-1 do
            begin
                yo:= out[lo,no];
                ew:= di*yo;
                dweight[li,ni,no]:= eps*ew + alp*dweight[li,ni,no]; weight[li,ni,no]:= eight[li,ni,no] +
                dweight[li,ni,no];
            end;
        end;
    end;
end;  { procedure LEARNING }
```

```
procedure make_matrix( it, isylla: integer );
var
i,j: integer;
begin
    for i:= 0 to inputs-1 do
            matrix_tmp[it,isylla,i]:= out[0,i];
    for i:= 0 to outputs-1 do matrix_tmp[it,isylla,inputs+i]:= out[Layers_Max-
            ,i];
end;  { procedure MAKE_MATRIX }

procedure write_weights;
var
lc,nc,nc_end,ic,ic_end: integer;
begin
    rewrite(weightfile);
    for lc:=1 to layers-1 do
    begin
            nc_end:=node[lc];
            ic_end:=node[lc-1];
            for nc:=0 to nc_end-1 do
            begin
                    write(weightfile,weight[lc,nc,ic]);
                write(weightfile,theta[lc,nc]);
            end;
    end;
end;

procedure read_weights;
var
lc,nc,nc_end,ic,ic_end: integer;
begin
    reset(weightfile);
    for lc:=1 to layers-1 do
    begin
            nc_end:=node[lc];
            ic_end:=node[lc-1];
            for nc:=0 to nc_end-1 do
            begin
                    read(weightfile,weight[lc,nc,ic]);
                read(weightfile,theta[lc,nc]);
            end;
    end;
end;

begin { main }
{ ilk veri grubunu yaz }

    for i:= 0 to 0 do writeln ('BEKİR KARLIK');
    node[2]:= Output_Nodes;
    Outputs:= Output_Nodes;
    node[1]:= Hidden_Nodes;
    node[0]:= Input_Nodes;
    Inputs:= Input_Nodes;
    Layers:= Layers_Max;
    eps:= Epsilon;
```

```
    alp:= Alpha;
    num:= Outputs_Max;  { burada 72 }

{   ikinci veri grubunu yaz }


{   ****************************
    HEDEF DEĞER VERİLERİNİ OKU
    **************************** }
dess[0,0]:=0.9;dess[0,1]:=0.1;dess[0,2]:=0.1;dess[0,3]:=0.1;dess[0,4]:=0.1;dess[0,5]:=0.1;
dess[1,0]:=0.9;dess[1,1]:=0.1;dess[1,2]:=0.1;dess[1,3]:=0.1;dess[1,4]:=0.1;dess[1,5]:=0.1;
dess[2,0]:=0.9;dess[2,1]:=0.1;dess[2,2]:=0.1;dess[2,3]:=0.1;dess[2,4]:=0.1;dess[2,5]:=0.1;
dess[3,0]:=0.9;dess[3,1]:=0.1;dess[3,2]:=0.1;dess[3,3]:=0.1;dess[3,4]:=0.1;dess[3,5]:=0.1;
dess[4,0]:=0.9;dess[4,1]:=0.1;dess[4,2]:=0.1;dess[4,3]:=0.1;dess[4,4]:=0.1;dess[4,5]:=0.1;
dess[5,0]:=0.9;dess[5,1]:=0.1;dess[5,2]:=0.1;dess[5,3]:=0.1;dess[5,4]:=0.1;dess[5,5]:=0.1;
dess[6,0]:=0.9;dess[6,1]:=0.1;dess[6,2]:=0.1;dess[6,3]:=0.1;dess[6,4]:=0.1;dess[6,5]:=0.1;
dess[7,0]:=0.9;dess[7,1]:=0.1;dess[7,2]:=0.1;dess[7,3]:=0.1;dess[7,4]:=0.1;dess[7,5]:=0.1;
dess[8,0]:=0.9;dess[8,1]:=0.1;dess[8,2]:=0.1;dess[8,3]:=0.1;dess[8,4]:=0.1;dess[8,5]:=0.1;
dess[9,0]:=0.9;dess[9,1]:=0.1;dess[9,2]:=0.1;dess[9,3]:=0.1;dess[9,4]:=0.1;dess[9,5]:=0.1;
dess[10,0]:=0.9;dess[10,1]:=0.1;dess[10,2]:=0.1;dess[10,3]:=0.1;dess[10,4]:=0.1;dess[10,5]:=0.1;
dess[11,0]:=0.9;dess[11,1]:=0.1;dess[11,2]:=0.1;dess[11,3]:=0.1;dess[11,4]:=0.1;dess[11,5]:=0.1;
dess[12,0]:=0.1;dess[12,1]:=0.9;dess[12,2]:=0.1;dess[12,3]:=0.1;dess[12,4]:=0.1;dess[12,5]:=0.1;
dess[13,0]:=0.1;dess[13,1]:=0.9;dess[13,2]:=0.1;dess[13,3]:=0.1;dess[13,4]:=0.1;dess[13,5]:=0.1;
dess[14,0]:=0.1;dess[14,1]:=0.9;dess[14,2]:=0.1;dess[14,3]:=0.1;dess[14,4]:=0.1;dess[14,5]:=0.1;
dess[15,0]:=0.1;dess[15,1]:=0.9;dess[15,2]:=0.1;dess[15,3]:=0.1;dess[15,4]:=0.1;dess[15,5]:=0.1;
dess[16,0]:=0.1;dess[16,1]:=0.9;dess[16,2]:=0.1;dess[16,3]:=0.1;dess[16,4]:=0.1;dess[16,5]:=0.1;
dess[17,0]:=0.1;dess[17,1]:=0.9;dess[17,2]:=0.1;dess[17,3]:=0.1;dess[17,4]:=0.1;dess[17,5]:=0.1;
dess[18,0]:=0.1;dess[18,1]:=0.9;dess[18,2]:=0.1;dess[18,3]:=0.1;dess[18,4]:=0.1;dess[18,5]:=0.1;
dess[19,0]:=0.1;dess[19,1]:=0.9;dess[19,2]:=0.1;dess[19,3]:=0.1;dess[19,4]:=0.1;dess[19,5]:=0.1;
dess[20,0]:=0.1;dess[20,1]:=0.9;dess[20,2]:=0.1;dess[20,3]:=0.1;dess[20,4]:=0.1;dess[20,5]:=0.1;
dess[21,0]:=0.1;dess[21,1]:=0.9;dess[21,2]:=0.1;dess[21,3]:=0.1;dess[21,4]:=0.1;dess[21,5]:=0.1;
dess[22,0]:=0.1;dess[22,1]:=0.9;dess[22,2]:=0.1;dess[22,3]:=0.1;dess[22,4]:=0.1;dess[22,5]:=0.1;
dess[23,0]:=0.1;dess[23,1]:=0.9;dess[23,2]:=0.1;dess[23,3]:=0.1;dess[23,4]:=0.1;dess[23,5]:=0.1;
dess[24,0]:=0.1;dess[24,1]:=0.1;dess[24,2]:=0.9;dess[24,3]:=0.1;dess[24,4]:=0.1;dess[24,5]:=0.1;
dess[25,0]:=0.1;dess[25,1]:=0.1;dess[25,2]:=0.9;dess[25,3]:=0.1;dess[25,4]:=0.1;dess[25,5]:=0.1;
dess[26,0]:=0.1;dess[26,1]:=0.1;dess[26,2]:=0.9;dess[26,3]:=0.1;dess[26,4]:=0.1;dess[26,5]:=0.1;
dess[27,0]:=0.1;dess[27,1]:=0.1;dess[27,2]:=0.9;dess[27,3]:=0.1;dess[27,4]:=0.1;dess[27,5]:=0.1;
dess[28,0]:=0.1;dess[28,1]:=0.1;dess[28,2]:=0.9;dess[28,3]:=0.1;dess[28,4]:=0.1;dess[28,5]:=0.1;
dess[29,0]:=0.1;dess[29,1]:=0.1;dess[29,2]:=0.9;dess[29,3]:=0.1;dess[29,4]:=0.1;dess[29,5]:=0.1;
dess[30,0]:=0.1;dess[30,1]:=0.1;dess[30,2]:=0.9;dess[30,3]:=0.1;dess[30,4]:=0.1;dess[30,5]:=0.1;
dess[31,0]:=0.1;dess[31,1]:=0.1;dess[31,2]:=0.9;dess[31,3]:=0.1;dess[31,4]:=0.1;dess[31,5]:=0.1;
dess[32,0]:=0.1;dess[32,1]:=0.1;dess[32,2]:=0.9;dess[32,3]:=0.1;dess[32,4]:=0.1;dess[32,5]:=0.1;
dess[33,0]:=0.1;dess[33,1]:=0.1;dess[33,2]:=0.9;dess[33,3]:=0.1;dess[33,4]:=0.1;dess[33,5]:=0.1;
dess[34,0]:=0.1;dess[34,1]:=0.1;dess[34,2]:=0.9;dess[34,3]:=0.1;dess[34,4]:=0.1;dess[34,5]:=0.1;
dess[35,0]:=0.1;dess[35,1]:=0.1;dess[35,2]:=0.9;dess[35,3]:=0.1;dess[35,4]:=0.1;dess[35,5]:=0.1;
dess[36,0]:=0.1;dess[36,1]:=0.1;dess[36,2]:=0.1;dess[36,3]:=0.9;dess[36,4]:=0.1;dess[36,5]:=0.1;
dess[37,0]:=0.1;dess[37,1]:=0.1;dess[37,2]:=0.1;dess[37,3]:=0.9;dess[37,4]:=0.1;dess[37,5]:=0.1;
dess[38,0]:=0.1;dess[38,1]:=0.1;dess[38,2]:=0.1;dess[38,3]:=0.9;dess[38,4]:=0.1;dess[38,5]:=0.1;
dess[39,0]:=0.1;dess[39,1]:=0.1;dess[39,2]:=0.1;dess[39,3]:=0.9;dess[39,4]:=0.1;dess[39,5]:=0.1;
dess[40,0]:=0.1;dess[40,1]:=0.1;dess[40,2]:=0.1;dess[40,3]:=0.9;dess[40,4]:=0.1;dess[40,5]:=0.1;
dess[41,0]:=0.1;dess[41,1]:=0.1;dess[41,1]:=0.1;dess[41,3]:=0.9;dess[41,4]:=0.1;dess[41,5]:=0.1;
dess[42,0]:=0.1;dess[42,1]:=0.1;dess[42,2]:=0.1;dess[42,3]:=0.9;dess[42,4]:=0.1;dess[42,5]:=0.1;
dess[43,0]:=0.1;dess[43,1]:=0.1;dess[43,2]:=0.1;dess[43,3]:=0.9;dess[43,4]:=0.1;dess[43,5]:=0.1;
dess[44,0]:=0.1;dess[44,1]:=0.1;dess[44,2]:=0.1;dess[44,3]:=0.9;dess[44,4]:=0.1;dess[44,5]:=0.1;
dess[45,0]:=0.1;dess[45,1]:=0.1;dess[45,2]:=0.1;dess[45,3]:=0.9;dess[45,4]:=0.1;dess[45,5]:=0.1;
```

```
dess[46,0]:=0.1;dess[46,1]:=0.1;dess[46,2]:=0.1;dess[46,3]:=0.9;dess[46,4]:=0.1;dess[46,5]:=0.1;
dess[47,0]:=0.1;dess[47,1]:=0.1;dess[47,2]:=0.1;dess[47,3]:=0.9;dess[47,4]:=0.1;dess[47,5]:=0.1;
dess[48,0]:=0.1;dess[48,1]:=0.1;dess[48,2]:=0.1;dess[48,3]:=0.1;dess[48,4]:=0.9;dess[48,5]:=0.1;
dess[49,0]:=0.1;dess[49,1]:=0.1;dess[49,2]:=0.1;dess[49,3]:=0.1;dess[49,4]:=0.9;dess[49,5]:=0.1;
dess[50,0]:=0.1;dess[50,1]:=0.1;dess[50,2]:=0.1;dess[50,3]:=0.1;dess[50,4]:=0.9;dess[50,5]:=0.1;
dess[51,0]:=0.1;dess[51,1]:=0.1;dess[51,2]:=0.1;dess[51,3]:=0.1;dess[51,4]:=0.9;dess[51,5]:=0.1;
dess[52,0]:=0.1;dess[52,1]:=0.1;dess[52,2]:=0.1;dess[52,3]:=0.1;dess[52,4]:=0.9;dess[52,5]:=0.1;
dess[53,0]:=0.1;dess[53,1]:=0.1;dess[53,2]:=0.1;dess[53,3]:=0.1;dess[53,4]:=0.9;dess[53,5]:=0.1;
dess[54,0]:=0.1;dess[54,1]:=0.1;dess[54,2]:=0.1;dess[54,3]:=0.1;dess[54,4]:=0.9;dess[54,5]:=0.1;
dess[55,0]:=0.1;dess[55,1]:=0.1;dess[55,2]:=0.1;dess[55,3]:=0.1;dess[55,4]:=0.9;dess[55,5]:=0.1;
dess[56,0]:=0.1;dess[56,1]:=0.1;dess[56,2]:=0.1;dess[56,3]:=0.1;dess[56,4]:=0.9;dess[56,5]:=0.1;
dess[57,0]:=0.1;dess[57,1]:=0.1;dess[57,2]:=0.1;dess[57,3]:=0.1;dess[57,4]:=0.9;dess[57,5]:=0.1;
dess[58,0]:=0.1;dess[58,1]:=0.1;dess[58,2]:=0.1;dess[58,3]:=0.1;dess[58,4]:=0.9;dess[58,5]:=0.1;
dess[59,0]:=0.1;dess[59,1]:=0.1;dess[59,2]:=0.1;dess[59,3]:=0.1;dess[59,4]:=0.9;dess[59,5]:=0.1;
dess[60,0]:=0.1;dess[60,1]:=0.1;dess[60,2]:=0.1;dess[60,3]:=0.1;dess[60,4]:=0.1;dess[60,5]:=0.9;
dess[61,0]:=0.1;dess[61,1]:=0.1;dess[61,2]:=0.1;dess[61,3]:=0.1;dess[61,4]:=0.1;dess[61,5]:=0.9;
dess[62,0]:=0.1;dess[62,1]:=0.1;dess[62,2]:=0.1;dess[62,3]:=0.1;dess[62,4]:=0.1;dess[62,5]:=0.9;
dess[63,0]:=0.1;dess[63,1]:=0.1;dess[63,2]:=0.1;dess[63,3]:=0.1;dess[63,4]:=0.1;dess[63,5]:=0.9;
dess[64,0]:=0.1;dess[64,1]:=0.1;dess[64,2]:=0.1;dess[64,3]:=0.1;dess[64,4]:=0.1;dess[64,5]:=0.9;
dess[65,0]:=0.1;dess[65,1]:=0.1;dess[65,2]:=0.1;dess[65,3]:=0.1;dess[65,4]:=0.1;dess[65,5]:=0.9;
dess[66,0]:=0.1;dess[66,1]:=0.1;dess[66,2]:=0.1;dess[66,3]:=0.1;dess[66,4]:=0.1;dess[66,5]:=0.9;
dess[67,0]:=0.1;dess[67,1]:=0.1;dess[67,2]:=0.1;dess[67,3]:=0.1;dess[67,4]:=0.1;dess[67,5]:=0.9;
dess[68,0]:=0.1;dess[68,1]:=0.1;dess[68,2]:=0.1;dess[68,3]:=0.1;dess[68,4]:=0.1;dess[68,5]:=0.9;
dess[69,0]:=0.1;dess[69,1]:=0.1;dess[69,2]:=0.1;dess[69,3]:=0.1;dess[69,4]:=0.1;dess[69,5]:=0.9;
dess[70,0]:=0.1;dess[70,1]:=0.1;dess[70,2]:=0.1;dess[70,3]:=0.1;dess[70,4]:=0.1;dess[70,5]:=0.9;
dess[71,0]:=0.1;dess[71,1]:=0.1;dess[71,2]:=0.1;dess[71,3]:=0.1;dess[71,4]:=0.1;dess[71,5]:=0.9;


{ ********************
   BAŞLANGIÇ MATRİSİ
********************** }
for i:= 0 to num-1 do
      for j:= 0 to ( inputs+Outputs-2 ) do
            matrix[i,j]:= 0;

for ii:= 0 to ( Trunc(Iterative_Max/Every)-1 ) do
      for i:= 0 to num-1 do
            for j:= 0 to ( inputs+Outputs-2) do
                  matrix_tmp[ii,i,j]:=0;
assign(outfile,'outfile.txt');
assign(weightfile,'wof.num');
for i:=1 to 24 do
      writeln;
write('E°itme (0) veya Test (1) ? ');
readln(tt);
for i:=1 to 24 do writeln;
if tt=1 then goto 10;
writeln('E°itme ...');writeln; rewrite(outfile);
wrand;
j:=0;dcount:=trunc(iterative_max/every)+1;

{ ************************************
                ÖĞRENME FAZI
**************************************
} while ( j < Iterative_Max ) do
```

```
begin
dcount:=dcount-1;
    for count:= 0 to Every-1 do
    begin
        error:= 0;
        for isylla:=0 to num-1 do
        begin
            initial;
            yread( isylla );
            wforward;
            err:= back;
            learning( alp, eps );
            error:= error+err;
        end;
        error:= error / outputs;
        eps:= 0.5 * sqrt( error );
        Inc( j );
    end;


{   ************************************************
                        TEST ETME FAZI
    ************************************************ }
        terror:= 0;
        for isylla:= 0 to num-1 do
        begin
            yread( isylla );
            wforward;
            terr:= back;
            terror:= terror + terr;
            make_matrix( trunc( j/Every ) - 1, isylla ); end;
        writeln (outfile, 'NO:  A1   A2   A3  A4    P     D BD  K BB Da Dk     desired value');
        writeln (outfile, '----  ----  ----  ----  ----  ----  ---  ---  ---  ---  ---  ---  ----------');
        for i:= 0 to num-1 do
        begin
            str( i:2, st1 );
            write(outfile,st1+'');
            for ii:= 0 to inputs-1 do
            begin
                    str( matrix_tmp[trunc(j/Every)-1,i,ii]:6:3, st1 );
                write(outfile,st1+'');
            end;
            write(outfile,' ');
            for ii:= 0 to outputs-1 do
            begin
                str( matrix_tmp[trunc(j/Every)-1,i,inputs+ii]:3:1, st1); write( outfile,
                st1+' ');
            end;
            for ii:= 0 to outputs-1 do
            begin
                str(dess[i,ii]:3:1,st1);
                write( outfile,st1+' ');
            end;
        writeln(outfile,'');
        end;
```

```
        terror:= terror/outputs;
        str ( j:4, st1 ); str ( eps:9:6, st2 );
        str ( error:9:6, st3 ); str ( terror:9:6, st4 );
        writeln( outfile, 'iteration=',st1,'   eps=',st2,
                                    '   error_1=', st3, '   error_t=', st4 );
        writeln( outfile );
{   }
        total_rate:=0;
        for isylla:= 0 to outputs-1 do
        begin
            rate[isylla]:= trunc(matrix_tmp[trunc(j/Every)-1,isylla,isylla]); total_rate:= total_rate +
            rate[isylla];
        end;
        total_rate:= total_rate/outputs;
{   }
        if ( j = Iterative_Max ) then
        begin
            writeln( outfile, 'learn & test finished at time' );
        end;
    write_weights;
end;
    close( outfile );
    goto 30;
    10:writeln('Test ...');writeln;
    read_weights;
    20:write('A1 = ? ');readln(out[0,0]);
        write('A2 = ? ');readln(out[0,1]);
        write('A3 = ? ');readln(out[0,2]);
        write('A4 = ? ');readln(out[0,3]);
        write('P = ? ');readln(out[0,4]);
        wforward;
        writeln('  A1    A2   A3    A4    P       D  BD K BB DADK ');
        writeln('------  ----- ----- ----- -----    -- -- - -- -- -- ');
    for ii:=0 to inputs-1 do
    begin
            str(out[0,ii]:4:3,st1);
            write(st1+'  ');
    end;
    write('  ');
    for ii:=0 to outputs-1 do
    begin
            str(out[layers-1,ii]:3:1,st1);
            write(st1+'  ');
    end;
    writeln;writeln;
    goto 20;
30:end.
```

ÖZGEÇMİŞ

Bekir KARLIK	Elektrik Yüksek Mühendisi (1991 Yıldız Teknik Üniversitesi)
Doğum Tarihi ve Yeri	08.01.1964, Karaman
1984-1988	Yıldız Teknik Üniversitesi Elektrik-Elektronik Fakültesi Elektrik Mühendisliği Bölümü
1978-1981	İzmir Mithat Paşa Endüstri Meslek Lisesi Elektrik Bölümü
1975-1978	İzmir 9 Eylül Orta Okulu
1970-1975	Karaman İstiklal İlkokulu
GÖREVİ	
1989-1993	Araştırma Görevlisi Yıldız Teknik Üniversitesi Elektrik Elektronik Fakültesi Elektrik Mühendisliği Bölümü Kontrol ve Kumanda Sistemleri Anabilim Dalı
Yabancı Dil	İngilizce, Fransızca