YILDIZ TEKNİK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

YAPAY SİNİR AĞLARI KULLANILARAK TÜRKÇEDEKİ SESLİ HARFLERİN TANINMASI

Elek.-Elektronik Müh. Seydi Vakkas ÜSTÜN

F.B.E. Elektrik Mühendisliği Anabilim Dalında
hazırlanan

YÜKSEK LİSANS TEZİ

Prof. Dr. Halit PASTACI Prof. Dr. Asım KASAPOĞLU Doç. Dr. Abdullah YILDIZ

Tez Danışmanı : Prof. Dr. Halit PASTACI

İSTANBUL, 1997

TEŞEKKÜR

Tez çalışmam sırasında bana her türlü yardımda bulunan danışman hocam Sayın Prof. Dr. Halit PASTACI'ya ve Yrd. Doç. Dr. Bekir KARLIK, Araş. Gör. Kayhan GÜLEZ'e ve bana destek olan tüm arkadaşlarıma teşekkür ederim.

İstanbul, 1997 Seydi Vakkas ÜSTÜN

İÇİNDEKİLER

ÖZET	iv
SUMMARY	v
1. GİRİŞ	1
2. DOĞRUSAL ÖNGÖRÜ ANALİZİ	2
2.1. Konu ve Önemi	2
2.2. Bazı Temel Kavramlar	4
2.2.1. Beklenen Değer	4
2.2.2. Moment	4
2.2.3. Raslantı Değişkeni	5
2.2.4. Rasgele İşleme	5
2.2.5. Durağanlık	6
2.2.6. Ergodiklik	7
2.3. Doğrusal Öngörü Analizi	8
2.3.1. Doğrusal Öngörü Katsayılarını Saptanması	11
2.3.1.1. Deterministik İşaret Hali	13
2.3.1.1.1. Özilişki (Autocorrelation) Yöntemi	14
2.3.1.1.2. Kovaryans Yöntemi	15
2.3.1.2. Rastgele İşaret Hali	17
2.3.1.2.1. Durağan Olma Durumu	17
2.3.1.2.2. Durağan Olmama Durumu	18
2.3.2. Kazancın (G) Saptanması	18
2.4. Kaynak Modeli	22
2.5. Direkt Form Analizi	23
2.5.1. Veri Pencerelemesi	23
2.5.2. Artık Pencereleme	25
3. LPC DENKLEMLERİNİN ÇÖZÜMÜ	27
3.1. Yinelemeli (Recursive) Durbin Çözüm Yöntemi	27
3.2. Cholesky Ayrışım (Decomposition) Çözüm yöntemi	32
4. KONUŞMA İŞARETİNİN ÖZELLİKLKERİNİN SAPTANMASI	
4.1. Konuşma İşaretlerinin Özelliklerinin Saptanması	35
4.2. Kısa Zaman Süreli İşaret Enerjisinin Bulunması	35
4.3. Ortalama Büyüklük	38
4.4. Ortalama Sıfır Eksenini Kesme Düzeyi	39
4.5. Konuşma ve Sessizliğin Ayrılması	40
4.6. Tekrarlama (pitch) Periyodunun Saptanması	40
4.7. Fourier Dönüşüm Süzgeç Grup Yöntemi	42
5. YAPAY SİNİR AĞLARI (YSA)	
5.1. YSA'nın Tanımı	45
5.2. YSA'nın Yapısı ve İşlem Elemanı	45
5.3. Bağlantı Geometrileri	47
5.4. Eşik Fonksiyonları	48

5.5. Ağırlık Uzayı 48
5.6. YSA'da Eğitme 49
5.7. Bellek 50
5.8. Hata Toleransı 51
5.9. Öğrenme Kuralları 51
5.10. Perceptron (İdrak, Almaç) 51
5.11. Çok Katmanlı İdrak 52
5.12. Hatanın Geriye Yayılma Algoritması ve Genelleştirilmiş Delta Kuralı 53
5.13. Öğrenme ve Momentum Katsayıları 57
6. UYGULAMA 58
7. SONUÇLAR VE ÖNERİLER 65
KAYNAKLAR 68
EKLER 69
EK1 70
EK2 73
EK3 82
ÖZGEÇMİŞ

ÖZET

Bu tezin amacı konuşma işaretinin incelenmesi ve son günlerde en popüler tanıma yöntemi olan Yapay Sinir Ağlarını (YSA) kullanarak Türkçe sesli harflerin tanınmasına dayalı bir uygulama yapmaktır. Bu uygulamada kelime tanımanın temelini oluşturacak şekilde bir yapı kurulmaya çalışıldı. Bu yüzden kelime tanımanın yapı taşı olan sesli seslerin tanınması asıl amaç olarak ele alındı. Tanıma işlemi genellikle işaretin işlenmesi, belirgin özelliklerinin çıkarılması, örü karşılaştırılması safhalarından oluşmaktadır.

Sesin analizinde LPC (linear predictive coding) analizi kullanılarak ilgili sistem parametreleri bulunmuştur. Konuşma işaretinin üretilmesini bir süzgeç olarak düşünürsek, bu süzgecin bir transfer fonksiyonu vardır. Bu transfer fonksiyonunun rezonans frekanslarına konuşma literatüründe formant denilir. Genelde ses tanımada ilk üç formant kullanılmaktadır. Fakat bu formantlar kişiden kişiye değişir, yani sabit olmayıp belirli bir alan içinde haraket ederler. Bu tezde DFT (Discrete Fourier transform) yöntemiyle ses birimlerinin, formatlarının çıkarılması yerine, tam bir spektrumu alınmıştır. Alınan bu spektrum 16 banda ayrılıp her bandın ortalaması hesaplanarak 16 tane parametre elde edilmiştir. Ses tanıma işlemi bu iki tip parametre kullanılarak Yapay Sinir Ağları ile gerçekleştirilmiştir. YSA algoritması olarak çok katmanlı genelleştirimiş delta kurallı hatanın geriye yayılması algoritması ile çalışıldı.

Bu işlemleri yapabilmek amacıyla çeşitli bölümlerde ilgili algoritma ve yöntemler tartışılacak ve yorumlanacaktır. Altıncı bölümde ise tezin asıl gerçekleştirme aşamasında yapılanlar uygulama anlatılacak ve yazılan program tanıtılacaktır. Son bölümde ise sonuçlar ve öneriler yer almaktadır.

SUMMARY

The aim of the thesis is to research speech signals and to do an application based on the recognition of Turkish vowel letters by using artificial neural networks (ANN) which is a popular recognition method recently. In the thesis, it was worked to establish a structure based on the recognition of a word. Because of this, it was application by vowel letters what is the base unit of the word recognition as the main goal. The operation of the recognition has generally some parts such as signal processing, a founding out specific characters of signals, comparing with pattern.

The system parameters in speech analysis have founded by using LPC (linear predictive coding) analysis. If we thing as a filter to produce a speech signal, the filter will have transfer function are called formant in the speech literature. It is generally used first-3 formant for speech recognition. The formants change person to person, that is not fixed (definite) and they move in a determined area. In the study, it has been taken a complete spectrum instead of taking out the formants of the speech units by using DFT (discrete fourier transform) method. The spectrum has separated 16 bands and calculated the average of it. So, it was obtained 16 parameters. The speech recognition operation was done by using this two types parameters with ANN. Multi-layer generalized delta-rule error back-propagation algorithm is used as ANN algorithm..

The algorithm and methods which is explained above to reach to results will be discussed. In 6. chapter, will be explained the main application and software part will be introduced. Results and conclusions are also in the last section.

1. GİRİŞ

Otomatik ses tanıma teknolojisinin amacı insanlar birbiriyle zahmetsiz biçimde konuştukları gibi makinelerle de konuşabilmelerine bir araç sağlamaktır. Bu amaç şu anda oldukça uzak olduğu halde, savaş-sonrası dönemde ses tanımayla ilgili ilk deneyler gerçekleştirildiğinden beri önemli gelişmeler başarıldı.

YSA ve bilgisayarlı (hesaplamaya dayalı) ses işleme hala gelişme sürecinde olan iki teknolojidir. Ses işlemedeki araştırma basit olarak insan-makine ilişkisine odaklanır. YSA'nın amaçlarından biri biyolojik sinir sistemlerine benzer bir şekilde bilgiyi işlemektir.

Ses tanıma, anlama kabiliyeti ve anlama işlemleri 1950'den beri aktif araştırma alanları oldu. Ancak, 1960'ların ortasında genel amaçlı sayısal bilgisayarların ortaya çıkması şimdi otomatik ses tanıma (ASR-automatic speech recognition) dediğimiz konuyu ortaya çıkardı. Başlangıç olarak ses tanıma tek bir kullanıcıdan gelen birkaç kelimeyi veya konuşmacıya bağımlı tanımaya dayalıydı. Ancak, hesaplama gücü ve depolama kapasitesi arttığından ASR sistemlerinin amaçları büyük-sözlüklü ve konuşmacıdan bağımsız sistemlere genişledi.

Sinir ağı (neural net) modelinin kullanılma alanı, çok miktarda hesaplamaya gereksinim duyulan ses ve şekil tanıma gibi alanlarda büyük bir potansiyele sahiptir.

YSA'nın kullanım alanı genişledi ve çok büyük ölçekli entegre sinir ağı çipleri farklı firmalar tarafından üretildi. Bilimsel çalışmalar devam ediyor, fakat sinir (neural) hesaplama hâlâ gelişmenin ilk adımlarında ilerlemektedir.

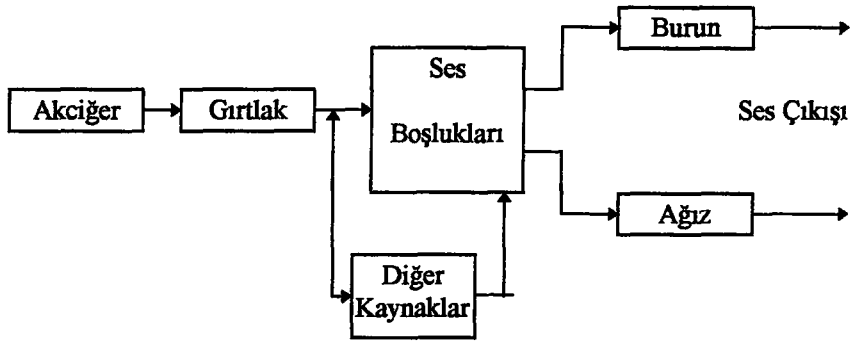
Bugünlerde YSA bilimindeki araştırmalar artarak devam etmektedir. Kitaplar, makaleler ve eğitime yönelik birçok yayın mevcuttur.

2. DOĞRUSAL ÖNGÖRÜ ANALİZİ

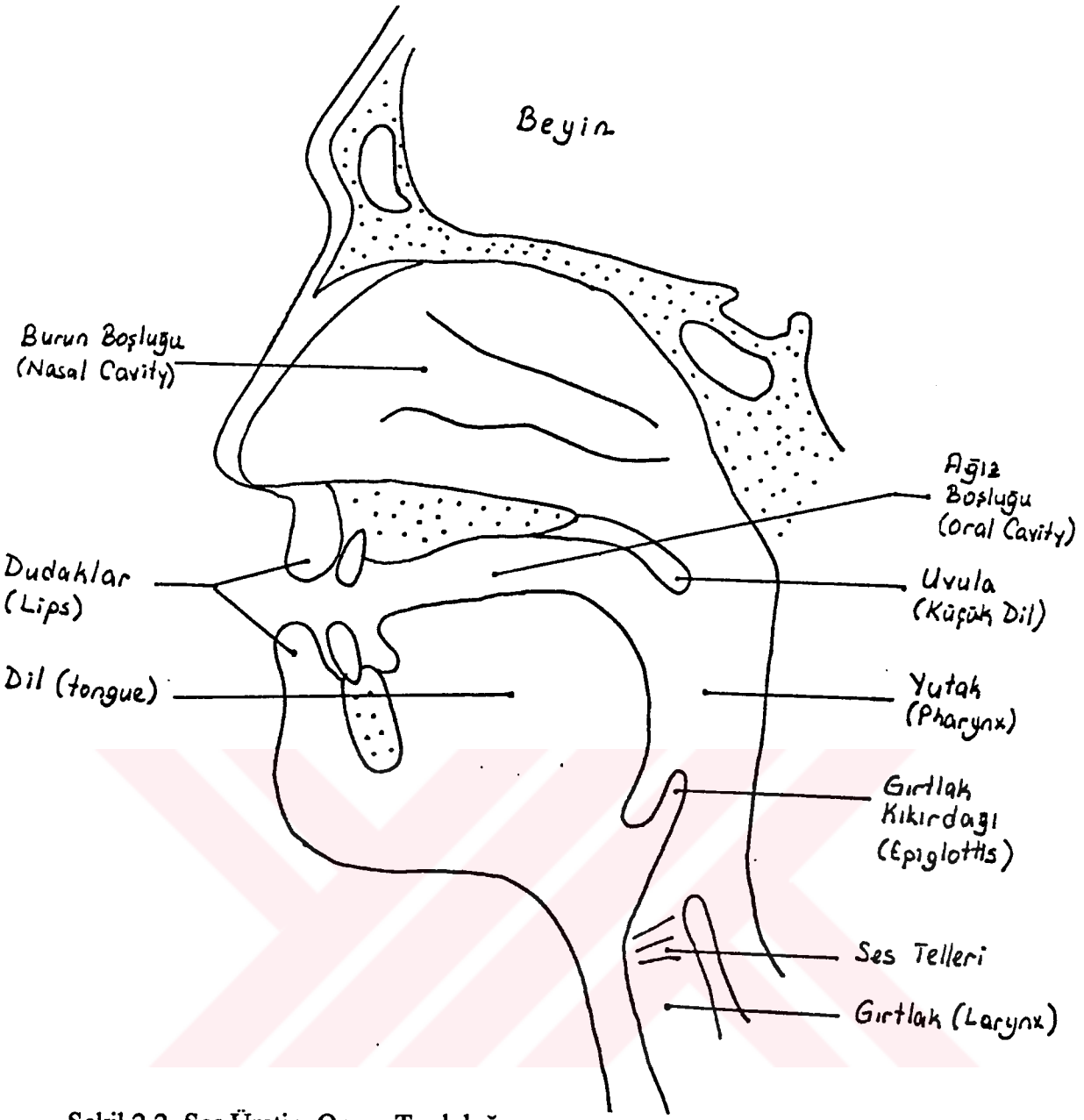
2.1. Konu ve Önemi

Doğrusal öngörü yönteminde ses işaretleri içindeki kısa süreli spektral özellikler yaklaşıklık yapılarak tüm kutup süzgeçler ile modellenir. Genel anlamda doğrusal öngörü, bir sisteminin çıkış işaretinin geçmiş çıkışlar, geçmiş ve şimdiki giriş işaretinin doğrusal fonksiyonu olarak hesaplama işlemine denir.

İnsan sesi, vücut içinden üflenerek ağza, dudaklara kadar gelip konuşma şekline dönüşünceye kadar uzun bir yol katetmekte, değişik boşluklardan geçmektedir.
Bu yol ve boşluklardaki pek çok etkenin etkisi ile farklı seslerin çıkması sağlanmakta ve konuşma gerçeklenmektedir. İnsandaki ses üretmeye yarayan organ topluluğu bir başka deyişle ses üretim şeması şekil 2.1., (Higgins, 1990) ve ses üretim sistemi şekil 2.2., (Flanagan et al, 1970) de verilmiştir.



Şekil 2.1 Ses Üretim Şeması



Şekil 2.2. Ses Üretim Organ Topluluğu

Buradan anlaşılacağı gibi insandaki ses üretim sisteminde uyarı işaretleri ses bölgesi (vocal tract) adını verebileceğimiz bir bölgeden geçtikten sonra çıkış işaretlerine, yani konuşma şekline dönüşmektedir.

Bu uyarı işaretleri ya peşi sıra gelen dürtülerin oluşturduğu sesli sesler (voiced-speech)(sesli harfler; a, o, e, i, ı, u, ü, ö, sesli-sessiz harfler: b, c, d, g, ğ, j, l, m, n, r, v, z, y) yada beyaz gürültü şeklindeki sessiz-seslerdir. (unvoiced-speech)(ç, f, h, k, p, s, ş, t), (Geçkinli et al,1985).

Ses üretim sistemindeki ses bölgesinin zaman bağımlı doğrusal bir sistem, yani bir LP (Linear Predictive - Doğrusal Öngörü) filtresi olduğu düşünülebilir. Esas amaç bu filtreyi tanımlamaktadır. Ancak bu tanımlama yapılırken sistem çıkışının tüm konuşma özelliklerini taşıması gerekir. Bu ise, doğrusal öngörü (Linear Prediction) analiz tekniği ile yapılabilir. Bu analiz tekniğine geçmeden önce bazı temel kavramlar açıklanacaktır.

2.2. Bazı Temel Kavramlar

2.2.1. Beklenen Değer

İstatiksel yöntemlerle ilgili en önemli ve temel kavramlardan biri rastlantı değişkenini yada rastlantı değişkenli fonksiyonların ortalama değerlerinin bulunmasıdır. Zaman fonksiyonunun tek bir zamana karşı gelen değeri ile tanımlanmış tek bir rastlantı değişkeni söz konusu olduğunda rastlantı değişkeninin kabul edilebilir olması değerleri sınır değerleri boyunca entegre edilmek suretiyle ortalama değerin bulunması gerekir. Bu tip bir işlem bizi beklenen değer tanımına ulaştırır.

E[x] sembolü genellikle 'x' in beklenen değeri ya da 'x' in matematiksel beklendiği olarak okunur. Beklenen değer işlemi lineerdir.

$$E[f(x)+g(x)] = E[f(x)] + E[g(x)]$$

$$E[c.f(x)] = c.E[f(x)]$$

2.2.2. Moment

$f(x)=x^n$ fonksiyonunu ele alalım, bu fonksiyonunun beklenen değeri ,

$$E[x^n] = \int_{-\infty}^{\infty} x^n . p(x) . dx \qquad (2.2.1)$$

n. moment olarak tanımlanır. 1. monent x' in ortalaması olarak adlandırılır.

$$\mu = E[x] = \int_{-\infty}^{\infty} x \cdot p(x) \cdot dx \qquad (2.2.2)$$

Benzer biçimde merkezi momentler de tanımlanabilir. Bunlar sadece rastlantı değişkenleri ile onların beklenen değerleri arasındaki farkların momentleridir. σ^2 varyans ise,

Var [x] = E $[x-\mu]^2$ ' den hesaplanır. (2.2.3)

$$= E[x^2] - 2\mu E[x] + \mu^2$$
$$= E[x^2] - E^2[x] = \sigma^2$$

görüldüğü gibi varyans karesel beklenen değer ile beklenen değerin karesi arasındaki farka eşittir. Varyansın karekökü σ "standart sapma" olarak bilinir.

2.2.3. Rastlantı Değişkeni

Birçok gözlemsel veri periyodik ve geçici verilere benzer biçimde bağlantılarla veya tekrarlanan deneylerle belirlenemez. Bu verilere rasgele değişken denir. Rastgele verilerin geçmişteki veya şimdiki değerlerinden gelecekteki değerleri kestirilemez. Gerçekte bir gözlemsel veriye önem kazandıran da bu önceden kestirilemeyiş özelliğidir. x rastgele değişkeni, olasılık yoğunluk fonksiyonu ile tamamen karakterize edilebilir.

2.2.4. Rastgele İşleme

Sonsuz sayıda rastgele veriler topluluğuna rastgele süreç adı verilir. Başka bir deyişle olası bütün X_j örnek fonksiyonlarına süreç denir. Süreç $\{x_j\}$ ile gösterilir. Rastgele sürecin temel ilkelerini Wiener ve Kolmogorof 1930'larda vermişlerdir. Rastgele verilerin analizlerinde ilişki (Correlation) fonksiyonları önemli yer tutar. Çünkü rastgele verilerin analitik bağıntılarla gösterme olanağı olmadığından ancak istatistik özellikleri ile tanımlanabilir. Rastgele veriler sonsuz zaman aralığında ne periyodik olaylar

gibi tanımlanmışlardır ne de geçici olaylar gibi sıfıra yaklaşırlar. Bu yüzden frekans ortamı analizlerinde fourier dizileri ve fourier integralinden doğrudan yararlanılamaz.

Bir rastgele süreci tamamen tanımlayabilmek için bir ya da birden çok fonksiyon yeterli olmayabilir. Rastlantı değişkenleri olasılık dağılım fonksiyonları ile tanımlanırlar. Bunlar bir n zamanında { x_j } nin alabileceği değerlerin olasılığını verir.

2.2.5. Durağanlık

Bir işlemenin bütün basit ve bileşik fonksiyonları, bir zaman başlangıcının seçimine bağlı değilse bu işlemenin durağan olduğu söylenir. Bu durumda daha önce incelenen bütün beklenen değerleri ve momentleri sabit olup mutlak değerlerinden bağımsızdır.

Eğer herhangi olasılık yoğunluk fonksiyonları zaman başlangıcının seçimine bağlı ise bu işlemeye durağan olmayan işleme denir. Bu durumda beklenen değerlerin ve momentlerin biri yada birkaçı zamana bağlı olacaktır.

Herhangi bir işlemenin, geçmişte sonlu herhangi bir zamanda başlaması ve gelecekte sonlu bir zamanda bitmesi kaçınılmaz olduğundan, gerçekte durağan bir işleme olmadığı söylenebilir. Ancak gözlem süresi boyunca farkedilir bir değişim göstermeyen işlemler pekçok durum için geçerlidir. Bu gibi durumlarda işleme durağan varsayılabilir.

$X(t_1)$ sürecinin t_1 zamanında bağımsız olması ve iki rastlantı değişkeni arasındaki ilişkinin; $E [x (t_1) . x (t_2)]$ yalnızca $t_2 - t_1$ zaman farkına bağlı olması durumunda " Geniş anlamda durağan" adını almaktadır.

2.2.6. Ergodiklik

Rastgele bir işlemenin zamanda veya uzamsal koordinatlardaki ortalaması "ensemble" (tüm örnek kümelerinin tümüne verilen addır.) ortalamasına eşitse bu işlemeye ergodik işlem denir. Başka bir ifade ile rastgele işlemenin bir örnek fonksiyonunun aritmetik ortalaması ve özilişki fonksiyonu, örneğin bir t_1 zamanındaki topluluğun aritmetik ortalaması ve özilişki fonksiyonuna eşitse bu işleme ergodik işleme denir. Ergodik rastgele işlemenin bütün fonksiyonları için bu özellik geçerlidir. Özet olarak ergodik işlemlerde yeterince uzun bir zaman aralığında örnek fonksiyondan hesaplanan istatistikler, topluluktan tek bir anda hesaplanan istatistiklerle aynıdır.

Sadece durağan rastgele işlemlerin ergodik olabileceğini belirtmek gerekir. Durağan olmayan rastgele işlemler ergodik olmayacağı gibi bazı durağan rastgele işlemlerde ergodik olmayabilir. Ergodik olan bir süreç peryotlu (bir süreç) olamaz. Bir sürecin (Homojen Markof süreci) geçiş elemanlarının negatif olmayan satır elemanları toplamının bir olması gerekir.

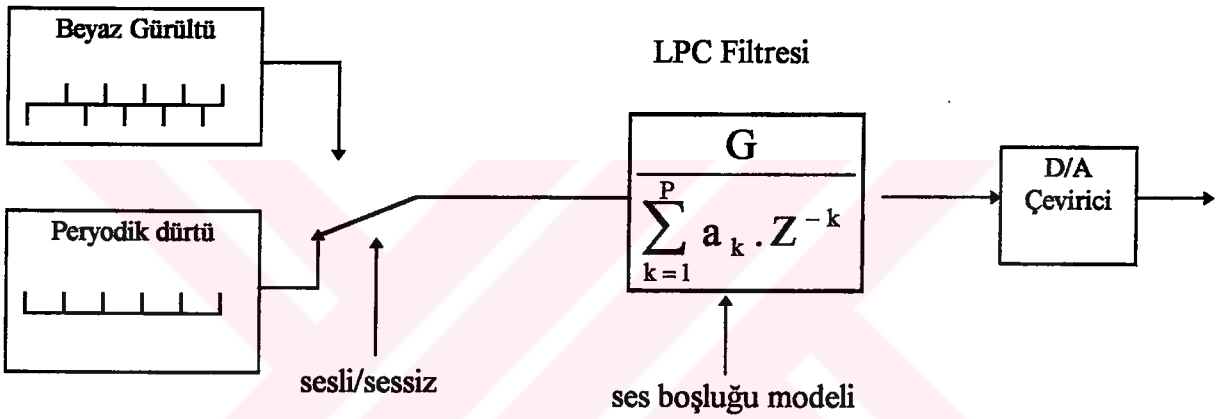
Ergodik varsayımı uygulamada önemlidir. Çünkü böylece birtek örnek fonksiyondan zaman ortalaması alınarak rastgele işlemenin istatistik özellikleri saptanabilir.

$$\mu_x = E[\, X_m \,] = \lim_{N\to\infty} \frac{1}{(2N+1)} \cdot \sum_{-N}^{N} X_m \qquad (2.2.4)$$

Eğer ergodik özelliği olmasaydı μ_x'in bulunması daha önce verilen (2.2.2) ifadesinden bulmak üzere olacaktı. Yani işlemdeki herhangi bir m için tüm mümkün olan çıkışların ortalamasını bulmak gerekecekti. Oysa ergodik durumda bir tek {x(m)}kümesinden ortalama bulunabilmektedir. Yani çeşitli m'ler için yapılan ortalamalar sıfır varyansta olmaktadır.

2.3. Doğrusal Öngörü Analizi

Konuşma analizi tekniklerinden en etkin olanı, Doğrusal öngörü analiz tekniğidir. Bu teknikteki temel ilke konuşma örneğinin (speech sample) geçmiş konuşma örneklerinden elde edilmesidir. Konuşma örneğinin, geçmiş konuşma örneklerinin doğrusal birleşimi şeklinde olduğu düşünülüp yaklaşıklık değeri bulunmakta ve esas örneklerle elde edilen yaklaşık örnekler arasındaki farklar en küçüğe indirilmek suretiyle doğrusal birleşimi oluşturan sistemin katsayıları bulunmaktadır.



Şekil 2.3.1 Yapay ses üretim blok diyagramı

Ses bölgesi (Vocal tract) bu şekilde zaman bağımlı sayısal bir filtre olarak düşünmek mümkündür. Bu durumda konuşma olayı basit bir diyagramla Şekil 2.3.1'de gösterildiği gibi olur.

Sürekli bir S(t) işaretinin t aralıklarıyla elde edilen sayısal değerlerinin $S(n.\Delta t)$ yada S_n olarak gösterelim n bir tam sayı Δt örnekleme aralığı olduğuna göre $1/\Delta t$ örnekleme frekansıdır.

Doğrusal öngörü analizinde S_n çıkış işaretinin geçmiş çıkış ile geçmiş giriş işaretinin U_n doğrusal bir birleşimden oluştuğu düşünüldüğünden,

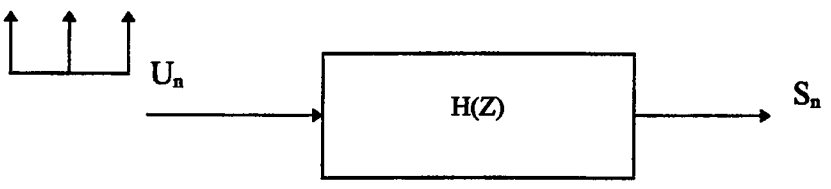
$$S_n = -\sum_{k=1}^{P} a_k . S_{n-k} + G . \sum_{l=0}^{q} b_l . U_{n-l} \qquad (2.3.1)$$

bağıntısı yazılabilir. Doğrusal öngörü ile şu andaki S_n çıkış işareti örneği, geçmiş çıkış ve geçmiş ve şimdiki giriş işaretlerinin örneklerinden tahmin edilmektedir.

Denklem (2.3.1) verilen a_k , $1 \leq k \leq p$ ve b_l $0 \leq l \leq q$ öngörü katsayıları, G sistemin kazanç orantısıdır. Bunlar tanımlanmak istenen doğrusal öngörü sisteminin parametreleridir. (2.3.1) denkleminin frekans ortamındaki ifadesi ise her iki tarafin Z dönüşümü alınarak bulunur.

$$H(z) = G . \frac{1 + \sum_{l=0}^{q} b_l . Z^{-l}}{1 + \sum_{k=1}^{p} a_k . Z^{-k}} \qquad (2.3.2)$$

Denklem 2.3.2 den $S(z) = -\Sigma S_n . Z^{-n}$ S_n'in, U(z) ise U(n) in Z dönüşümüdür.



Şekil 2.3.2. Ayrık Ses Üretim Modeli

Denklem (2.3.2) deki H(z) kutup sıfir modelidir. Pay ve paydadaki polinomların kökleri sırasıyla Z dönüşümündeki sıfir ile kutupları verecektir. Genel kutup-sıfir modelinin tüm-kutup ve tür-sıfir olarak iki özel durumu vardır.

a-) Tüm - sıfir modeli $a_k = 0$ (k=1,2,3.....p)

b-) Tüm - kutup modeli $b_l = 0$ (l=1,2,3.....q)

Tüm - sıfir modeli istatiksel literatürde moving - average (MA) ve tüm - kutup modeli ise Autoregressive (AR) model olarak tanımlanır. Kutup - Sıfir modeli ise Autoregressive - moving average (ARMA) olarak adlandırılır.

Bu çalışmada algoritmik çözümlerin daha kolay ve açık olması nedeniyle salt - kutuplu model kullanılmıştır. Bu durum için,

$$S_n = -\sum_{k=1}^{p} a_k . S_{n-k} + G.U_n \qquad (2.3.3)$$

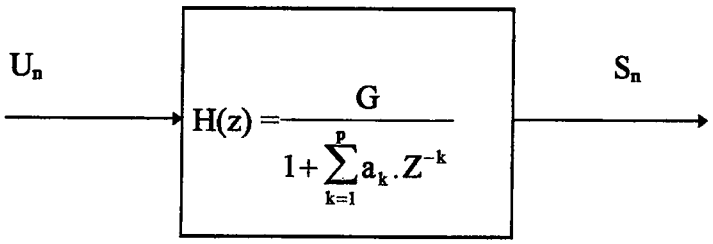
ve transfer fonksiyonu H (z) de,

$$H(z) = \frac{G}{1+\sum_{k=1}^{p} a_k . Z^{-k}} \qquad (2.3.4)$$

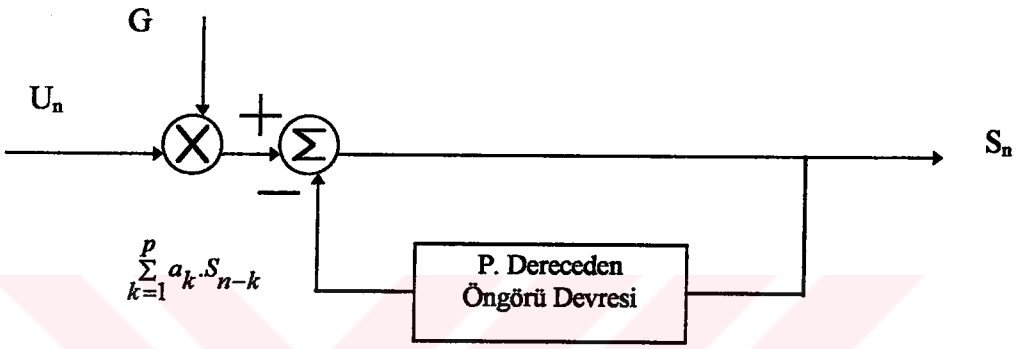
şeklinde yazılabilir.

Tüm kutup modelinde şu andaki çıkış işareti örneği S_n , şu andaki giriş işareti örneği U_n ve geçmiş çıkış işaretlerinin doğrusal kombinasyonu ile tahmin edilir. Belirli bir S_n işareti için a_k öngörü katsayıları ve G kazancını bulma doğrusal öngörü problemidir.

Bu modelin frekans ve zaman ortamındaki gösterimi şekil 2.3.3 deki gibidir.



(a)



(b)

Şekil 2..3.3. a) Ayrık frekans domeni tüm - kutup modeli
b) Ayrık zaman domeni tüm - kutup modeli

Transfer fonksiyonu H(z) olan bir filtre sesli - sesler (voiced speech) için bir dürtü katarı sessiz - sesler (unvoiced speech) için rastgele gürültü ile uyarılacaktır. Bu sistemin parametreleri; sistemin kazancı (G) ve sayısal filtrenin katsayılarından (a_k) ibarettir. Burada esas problem öngörü katsayılarının ve kazancın saptanmasıdır. Doğrusal öngörü analizi yöntemi ile parametreler kolaylıkla elde edilebilir.

2.3.1. Doğrusal öngörü katsayılarının saptanması

U_n girdi işareti model gereğince bilinmeyen varsayıldığı için, S_n'in yaklaşık kestiriminin sadece geçmiş çıktıların doğrusal birleşiminden elde edilebileceği düşünülürse, bu yaklaşık değer,

$$\overline{S}_n = -\sum_{k=1}^{p} a_k . S_{n-k} \qquad (2.3.5)$$

şeklinde yazılabilir. Geçmiş çıktıların $\overline{S}_n$ değerini ne ölçüde etkiledikleri doğal olarak a_k katsayılarının değerlerine bağlı olmakla birlikte, a_k nın toplam sayısı P' ye de bağlıdır. P'nın büyük olması halinde çok uzak geçmişteki çıkış değerleri de $\overline{S}_n$'i etkileyeceklerdir. Bu nedenle P önemli bir parametredir ve süzgeç boyu olarak adlandırılır. (2.3.5.) bağıntısı bir evrişim (convolution) işlemidir ve evrişim S_n'in geçmiş değerleri ile a_k arasında olmaktadır. Burada a_k bir süzgeç görevini görmekle ve verinin geçmiş değerleri süzerek n'deki değerini vermekle, yani verinin geçmişini kullanarak geleceğini, belirtmektedir. Buna dayanarak a_k bir önkestirme süzgeci (prediction filter) olarak adlandırılmaktadır. Bu konu, önkestirme taslaklanması olarak incelendiğinde, verinin geçmiş değerinin a_k ile evrişimi sonucunda bir değer oluşmakta, ancak bu değer bir yanılgı içermektedir. Yani gerçek çıktı S_n ile yaklaşık olarak kestirilen çıktı $\overline{S}_n$ arasında,

$$e_n = S_n - \overline{S}_n = S_n + \sum_{k=1}^{p} a_k . S_{n-k} \qquad (2.3.6)$$

gibi bir önkestirme yanılgısı (predicition error) mevcut olacaktır. a_k ların S_n lerden hareketle en iyi biçimde saptanması kestirme yanılgısı e_n'in en küçük hale getirilmesi ile mümkün olacaktır. Yani bu hataların karelerinin toplamını en küçük yapan a_k katsayıları bu sistemin katsayıları olacaktır. Bu da en küçük kareler yönteminin uygulanması ile mümkündür.

Toplam kare hatanın en küçük yapılması işlemi S_n işaretinin, deterministtik ya da rastgele işaret gibi iki hali için ayrı ayrı incelenebilir.

2.3.1.1. Deterministik İşaret Hali

Toplam kare hatayı,

$$E_n = \sum_n e_n^2 = \sum_n (S_n + \sum_{k=1}^{p} a_k . S_{n-k})^2 \qquad (2.3.7)$$

şeklinde yazalım, toplamlardaki sınırlar daha sonra belirtileceklerdir. Toplam kare hatayı en küçük yapan a_k katsayılarını bulmak için, toplam karesel hata E öngörü katsayılarına göre türevi alınıp sıfıra eşitleyerek en aza indirgenir.

$$\frac{\partial E_n}{\partial a_n} = 0 \qquad (i = 1, 2, 3.....,p) \qquad (2.3.8)$$

yapmak yeterli olacaktır. Bu işlem sonucunda ,

$$\sum_{k=1}^{p} a_k . \sum_n S_{n-k} . S_{n-i} = \sum_n S_n . S_{n-k} \qquad (i = 1, 2, 3.....,p) \qquad (2.3.9)$$

minimum toplam karesel hata E_p denklem (2.3.9) ile denklem (2.3.7)' den elde edilir.

$$E_p = \sum_n S_n^2 + \sum_{k=1}^{p} a_k \sum_n S_n . S_{n-k} , \qquad (2.3.10)$$

normal denklem takımı elde edilecektir. (p bilinmeyenli p denklem takımı) Burada bilinmeyenler a_k (k = 1, 2,.....,p) katsayılarıdır.

Doğrusal öngörü analizinde temel problemler konuşma işaretinin (speech signal) spektral özelliklerinin en iyi şekilde yansıtan a_k (k = 1, 2,.....,p) katsayılarının doğrudan doğruya konuşma işaretinden elde edilebilmesidir. Ancak konuşma işaretinin zamana bağımlı olması özelliği nedeniyle a_k öngörü katsayılarının konuşma işaretinin kısa süreli parçalarından elde etme zorunluluğu vardır. Temel yaklaşım, konuşma işareti üzerindeki kısa süreli bir parçaya ait öngörü hatalarının karelerinin toplamını en küçük yapan a_k katsayılarının bulunmasıdır. Konuşma işaretinin kısa süreli parçalarına ilişkin öngörü katsayıların, konuşmayı oluşturan modelde H(z) sistem fonksiyonunun parametreleri olacaktır.

Yukarıda bahsedildiği gibi kısa süreli işaret parçalarına analiz işlemi uygulanacağından (2.3.7) ve (2.3.10) denklemlerindeki $\sum_n$ ile gösterilen toplamların sınırlarının belirtilmesi gerekir. Bu konuda iki yaklaşım vardır (Makhoul, 1975).

2.3.1.1.1. Özilişki (Autocorrelation) Yöntemi

S_n işaretinin $0 \leq n \leq N-1$ aralığı dışında 0 olduğu düşünülür, yani işaret W_n penceresi ile pencerelenmiş olursa, bu yeni işaretin özilişki fonksiyonu,

$$R(i) = \sum_{n=0}^{N-1-i} S_n . S_{n+i} \qquad (2.3.11)$$

bağıntısı ile ifade edilebilir. Özilişki fonksiyonu çift fonksiyondur.

$$R(i) = R(-i)$$

o halde (2.3.9) denkleminden eşitliğin sağ tarafi R (i) olacaktır. Aynı düşünce ışığında (2.3.9) denklemi,

$$\sum_{k=1}^{p} a_k . R(|i-k|) = -R_n(i) \qquad (i = 1, 2, \ldots\ldots\ldots, p) \qquad (2.3.12)$$

olarak yazılabilir. Bu denklem takımı matris çarpımı şeklinde gösterilebilir.

$$\begin{bmatrix} R_n(0) & R_n(1) & R_n(2) & \ldots\ldots\ldots & R_n(P-1) \\ R_n(1) & R_n(0) & R_n(1) & \ldots\ldots\ldots & R_n(P-2) \\ R_n(2) & R_n(1) & R_n(0) & \ldots\ldots\ldots & R_n(P-3) \\ . & . & . & . & . \\ . & . & . & . & . \\ R_n(P-1) & R_n(P-2) & R_n(P-3) & \ldots\ldots\ldots & R_n(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ . \\ . \\ a_P \end{bmatrix} = - \begin{bmatrix} R_n(1) \\ R_n(2) \\ R_n(3) \\ . \\ . \\ R_n(P) \end{bmatrix} \qquad (2.3.13)$$

Özilişki değerlerinin oluşturduğu (pxp) boyutundaki matris bir toeplitz matrisidir. Yani hem bir simetrik matris hem de esas köşegene paralel köşegenler üzerindeki elemanlar birbirine eşittir. (2.3.12) yada (2.3.13) ile verilen denklemlerin çözümü ve a_k (k=1, 2, ...,p) katsayılarının elde edilmesinden daha sonraki bölümde bahsedilecektir.

2.3.1.1.2. Kovaryans Yöntemi

Bu yöntem konuşma işaretinin kısa süreli parçalarının tanımı ve (2.3.7) ile (2.3.10) denklemlerindeki $\sum$ toplamlarının sınırlarının saptanması için uygun bir yöntemdir. Toplam kare hata,

$$E_n = \sum_{n=0}^{N-1} e_n^2 \qquad (2.3.14)$$

şeklinde tanımlanır.

$$\phi_n(i,k) = \sum_{n=0}^{N-1} S_{n-i} = \sum_{n=-i}^{N-i-1} S_n . S_{n+i-k}$$

$$= \sum_{n=-k}^{N-k-1} S_n . S_{n+k-1} \qquad (i = 1,2,.....,p)$$

$$(k = 1,2,.....,p) \qquad (2.3.15)$$

bağıntılarını yazalım. Buna gör (2.3.9) denklemi ,

$$\sum_{k=1}^{p} a_k . \phi(i,k) = \phi(i,0) \qquad (i = 1,2,.....,p) \qquad (2.3.16)$$

şekline girecektir. Burada ϕ (i, k), S_n işaretinin verilen aralıktaki kovaryansıdır. (2.3.16) denkleminin matris çarpımı şeklindeki ifadesi de aşağıdaki gibidir.

$$\begin{bmatrix} \phi_n(1,1) & \phi_n(1,2) & \phi_n(1,3) & ..... & \phi_n(1,P) \\ \phi_n(2,1) & \phi_n(2,2) & \phi_n(2,3) & ..... & \phi_n(2,P) \\ \phi_n(3,1) & \phi_n(3,2) & \phi_n(3,3) & ..... & \phi_n(3,P) \\ . & . & . & . & . \\ . & . & . & . & . \\ \phi_n(P,1) & \phi_n(P,2) & \phi_n(P,3) & ..... & \phi_n(P,P) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ . \\ . \\ a_P \end{bmatrix} = \begin{bmatrix} \phi_n(1,0) \\ \phi_n(2,0) \\ \phi_n(3,0) \\ . \\ . \\ \phi_n(P,0) \end{bmatrix}$$

(2.3.17)

$\phi_n(i,k)=\phi(k,i)$ olduğundan yukarıdaki (pxp) boyundaki matris simetrik bir matristir. Fakat Toeplitz matrisi değildir.

2.3.1.2. Rastgele işaret hali

S_n rastgele bir işaretin bir örneği (sample) ise e_n hatası da rastgele bir işaretin bir örneği olacaktır. Bu durumda e_n hatalarının karelerinin beklenilen değerinin en küçük yapmak suretiyle a_k katsayıları elde edilebilecektir.

$$E_n = \xi(e_n^2) = \xi(S_n + \sum_{k=1}^{p} a_k . S_{n-k})^2 \qquad (2.3.18)$$

$$\frac{\partial E_n}{\partial a_i} = 0 \qquad (i = 1, 2,.....,p)$$

Şartından,

$$\sum_{k=1}^{p} a_k . \xi(S_{n-k} . S_{n-i}) = \xi(S_n . S_{n-i}) \qquad (i = 1, 2,.....,p) \qquad (2.3.19)$$

elde edilir.

(2.3.19) denklemindeki beklenti, S_n'nin durağan olup olmamasına bağlı olarak hesaplanır.

2.3.1.2.1. Durağan Olma Durumu

Bu durum için ,

$$\xi (S_{n-k} . S_{n-i}) = R (i-k) \qquad (2.3.20)$$

dır. Burada R(i) özilişkiyi göstermektedir. Böyle olunca (2.3.19) ve (2.3.12) denklemleri benzer olacaktır ve a_k katsayıları özilişki yönteminden yol izlenerek saptanabilmektedir, (Makhoul, 1975).

2.3.1.2.2. Durağan olmama durumu

Bu durum için,

$$\xi (S_{n-k} . S_{n-i}) = R (n-k, n-i) \qquad (2.3.21)$$

dir. Bu durumda (2.3.19) denklemi,

$$\sum_{k=i}^{p} a_k . R(-k,-i) = R(0,-i) \qquad (2.3.22)$$

olarak yazılabilecektir, (Makhoul, 1975).

Görüldüğü gibi özilişki yönteminde işaret N noktalı bir pencere ile pencerelenmekte ve kısa süreli özilişki fonksiyon değerleri hesaplanmaktadır. Bunların oluşturduğu özilişki matrisi bir Toeplitz matrisidir. Kovasyans yönteminde ise sadece $P \leq n \leq N-1$ aralığındaki işaret değerlerinin bilinmesi yeterlidir. Bu yöntemde elde edilen korelasyon matrisi simetrik bir matristir. Fakat Toeplitz matris değildir.

2.3.2. Kazancın (G) Saptanması

Daha önce yanılgı için bulunan (2.3.6) denklemini ele alalım.

$$e_n = S_n + \sum_{k=1}^{p} a_k . S_{n+k}$$

bu denklemi aşağıdaki gibi yazalım.

$$S_n = e_n - \sum_{k=1}^{p} a_k . S_{n-k} \qquad (2.3.23)$$

(2.3.3) ile (2.3.23) denklemi karşılaştırıldığında,

$e_n = G \, . \, x_n$

olduğu görülür. Bu ise, giriş işaretinin yanılgı işareti ile orantılı olduğu anlamına gelmekte, ayrıca $G.x_n$' in toplam enerjisinin yanılgı işaretinin toplam enerjisine eşit olduğu görülür.

$$E_n = \sum_{n=0}^{N-1} e_n^2 = G^2 . \sum_{n=0}^{N-1} x_n^2 \qquad (2.3.24)$$

uyarı işareti ya sesli - sesler yada sessiz - sesler olacağına ve sesli sesler de periyodik dürtü biçiminde olduklarına göre sesli sesler için uyarının yeri giriş işaretinin $U_n = \delta_n$, bir başka deyişle n = 0 için birim dürtü olduğu düşünülür. Sessiz - seslerde ise U_n girişinin ortalaması sıfir ve varyansı 1 olan stasyoner beyaz gürültü olduğu kabul edilir.

Bilindiği gibi bir salt - kutuplu modelde H(z) filtresinin girişi birim dürtü olursa [n=0 , $U_n=\delta_n$] filtrenin çıkışı da birim dürtü tepkisi h(n) olacaktır.

$$h(n) = \sum_{k=1}^{p} a_k . h_{n-k} + G.\delta_n$$

Bu denklemin her iki tarafını h_{n-i} çarparsak,

$$\overline{R}(i) = \sum_{n=0}^{\infty} h_n . h_{n-i}$$

olduğuna göre,

$$R(i) = \sum_{k=1}^{p} a_k . \overline{R}(i,k) \qquad (i = 1, 2, \ldots\ldots\ldots p) \qquad (2.3.25)$$

ve

$$\overline{R}(0) = \sum_{k=1}^{p} a_k . \overline{R}(k) + G^2$$

(2.3.25) ve (2.3.12) bağıntıları bezer olduğundan ve işaret ile dürtü tepkisinin enerjilerinin sırasıyla R (0) ve$\overline{R}(0)$ eşit olması gerektiğinde sonuçta,

$$G^2 = R_n(0) - \sum_{k=1}^{p} a_k . R_n(k) = E \qquad (2.3.26)$$

bağıntısı elde edilir.

Sessiz - Sesler için giriş işaretinin, ortalaması sıfir ve varyansı 1 olan beyaz gürültü şeklinde olduğu varsayılmaktadır.

$$\xi\ [\, U(i) . U(n-i)\,] = \delta(i)$$

Bu nedenle filtrenin çıkışı g_n de durağan rastgele bir işaret olacaktır.

$$g_n = \sum_{k=1}^{p} a_k . g_{n-k} + G.U_n \qquad (2.3.27)$$

Yukarıdaki bağıntının her iki tarafi g_{n-i} ile çarpılıp beklentisi alınacak olursa, ayrıca eğer$\overline{R}(i)$ g_n' nin özilişki fonksiyonu ise,

$$R(i) = \xi[g_n . g_{n-i}] \sum_{k=1}^{p} a_k . [g_{n-k} . g_{n-i}] + \xi[G.U_n . g_{n-i}]$$

$$= \sum_{k=1}^{p} a_k . \overline{R}(i-k) \qquad i \neq 0 \qquad (2.3.28)$$

U_n, U_i'den önceki hiçbir değerle ilişkili olmadığından $i > 0$ için $\xi [U_n.g_{n-i}] = 0$ olacak $i=0$ için ise,

$$\overline{R}(0) = \sum_{k=1}^{p} a_k . \overline{R}(k) + G.\xi[U_n . g_n] \qquad (2.3.29)$$

ve

$[U_n . g_n] = [U_n . (G . U_{n+n}$' den önceki terimler $)] = G$

olduğundan,

$$\overline{R}(0) = \sum_{k=1}^{p} a_k . \overline{R}(k) + G^2 \qquad (2.3.30)$$

olacaktır. Yine işaretin enerjisi ile $G.x_n$'e karşı tepkinin enerjisinin aynı olması gerektiğinden, yani,

$$\overline{R}(i) = R(i) \qquad 0 \le i \le p$$

G kazancına ilişkin,

$$G^2 = R(0) - \sum_{k=1}^{p} a_k . R(k) \qquad (2.3.31)$$

bağıntısı elde edilecektir.

2.4.Kaynak Modeli

LP analizinde A (z) aşağıdaki ifade ile verilmiştir.

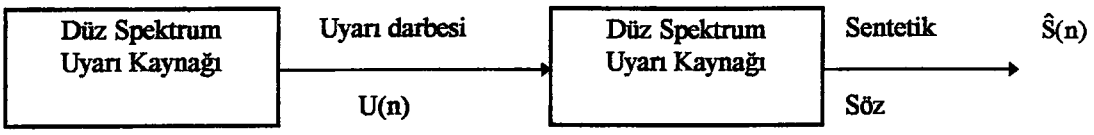
$$A(z) = 1 + \sum_{k=1}^{p} a(k).Z^{-k} \qquad 1 \le k \le p$$

Burada a_k öngörü katsayıları ve z^{-1} de birim gecikme operatörüdür. Artık e(n) işareti ise;

$$e(n) = s(n) + \sum_{k=1}^{p} a(k).s(n-k)$$

e (n) in kullanılmasında temel problem, depolanması için gerekli bit miktarının çok fazla olmasıdır. Örneğin 10 kHz lik bir örnekleme hızında, herbir örneğin sadece tek bir bitle kuantalanması durumunda bile 10.000 bits/s'lik bir belleğe gereksinim duyulur, bu durum pratik uygulamalarda sorun doğurur. Etkin bir çözüm uyarıyı iki kaynaktan sağlayan bir modelle sağlanabilir.

Şekil 2.4.1. LP sentezinde kullanılan temel modeli vermektedir. Model, 2 ana bileşene sahiptir. Bir düz spektrum uyarı kaynağı ve birde spektral şekilleme filtresi H(z). Uyarı kaynağı U(n) işaretinin düz bir spektral zarf ile birlikte üretir. Spektral zarf H(z) filtresinin $\overline{S}(n)$ sentetik söz işaretinin üretilmesini sağlar. H(z) filtresinin düz bir spektruma sahip olduğu için, çıkış işaretinin spektral zarfi H(z) filtresinin spektrumunun şeklini alır.



Şekil 2.4.1. Temel söz sentezleme modeli .

Bu modelde bir darbe kaynağı ve bir de gürültü kaynağı bulunur. Darbe kaynağı bir perde periyodu ile ayrılmış, ünlü seslerin üretilmesi için kullanılan, bir dizi darbe üretir. Gürültü kaynağı ise beyaz gürültü kaynağıdır. Rastgele sayı üretir ve düz bir spektral zarfa sahiptir. Bu kaynak sessizlerin üretilmesi içindir.

Birçok ses ya darbe kaynağı veya gürültü kaynağı tarafından üretilirken, bazı seslerde (z, v, gibi), bu iki kaynağın ortak kullanımdan elde edilir. Bu karma kaynak modeli sözün daha doğal olarak elde edilmesini sağlar.

Bu model pratik uygulamalarda büyük kolaylık sağlar.

2.5. Direkt Form Analiz

Direkt form analizinde öngörü katsayıları {a(k)}, e(n) artık işaretindeki enerjinin en aza indirgenmesini sağlayacak şekilde hesaplanır. Bu hesaplama söz dinamiğinin takip edilebilmesi için kısa süreli olarak yapılır. Aşağıda hesaplama için kullanılan iki yöntem tanıtılmaktadır.

2.5.1. Veri Pencerelenmesi

Veri Pencerelemesi yönteminde söz işareti s(n) bir veri penceresi w(n) ile çarpılır ve ters filtrelenir. En çok aşağıda işaret edildiği gibi sonlu veri pencereleri kullanılır.

$$x(n) = \begin{cases} W(n).s(n) & 0 \le n \le N-1 \\ 0 & \text{diğer durumlarda} \end{cases} \qquad (2.5.1)$$

Burada, $(0 \le n \le N\text{-}1)$ ile verilen aralığın dışında pencerenin sıfır değerini aldığı kabul edilebilir. Pencere genişliği N, 20 - 30 ms' lik kısa süreli bir analizi sağlayacak şekilde seçilir.

Enerji en aza indirgenmiş artık işaret, x(n) in A(z) filtresinden geçirilmesi ile elde edilir.

Artık enerji aşağıdaki gibidir.

$$E = \sum_{n=-\infty}^{\infty} e_n^2(n) = \sum_{n=-\infty}^{\infty} [x(n) + \sum_{k=1}^{p} a(k).x(n-k)]^2 \qquad (2.5.2)$$

burada $e_x(n)$ percelenmiş işaret x(n)'in artık işaretidir. E ise x(n)' in öngörülmesindeki toplam karesel hatadır. E'yi minimum yapan {a(k)} katsayıları, E' nin kısmi türevinin {a(k)} katsayılarının sıfira eşitlenmesi ile elde edilir. Sonuç aşağıda (2.5.3) eşitliği ile verilmiştir. (2.5.3) eşitliği x(n) pencerelenmiş sinyalinin özilişkisidir. (2.5.3) eşitlikleri p bilinmeyenle p adet lineer eşitliktir. Bu eşitlikler çözülerek istenen öngörü katsayıları elde edilir. Bu yöntem özilişki yöntemi olarak bilinir. Çünkü R(i-k) katsayıları işaretin özilişki katsayılarıdır.

$$\sum_{k=1}^{p} a(k).R(i-k) = -R(i) \qquad 1 \le i \le P \qquad (2.5.3)$$

$$R(i) = \sum_{n=-\infty}^{\infty} x(n).x(n-1) = \sum_{n=1}^{N-1} x(n).x(n-i) \qquad 0 \le i \le P \qquad (2.5.4)$$

Minimum artık enerji E_p (2.4.3)'ün (2.4.2)'de kullanılması ile elde edilir.

$$E_p = R(0) + \sum_{k=1}^{p} a(k).R(k)$$

E_p minimum toplam karesel hata olarak bilinir veya minimum öngörü hatasıdır.
Sentez için H (z) filtresinin kazancı,

$$G^2 = E_p \qquad (2.5.5)$$

şeklinde eşitlenir ve (2.4.4) eşitliği, $\overline{R}(0) = \sum_{k=1}^{p} a(k).\overline{R}(k) + G^2$

(2.4.3) eşitliği, $\overline{R}(i) = -\sum_{k=1}^{p} a(k).\overline{R}(i-k) \qquad 1 \le i \le \infty$

ile karşılaştırılırsa;

$$\overline{R}(i) = R(i) \qquad o \le i \le p \qquad (2.5.6)$$

olduğu, yani pencerelenmiş işaretin ilk (p+1) özilişki katsayısının eşit olduğu görülür. $R(0)=\overline{R}(0)$ olması sentetik ve pencerelenmiş işaretlerdeki enerjinin eşit olduğu sonucunu ortaya koyar.

2.5.2. Artık Pencereleme

Burada en aza indirgenecek enerji,

$$E = \sum_{n=-\infty}^{\infty} W_e(n).e^2(n) \qquad (2.5.7)$$

eşitliği ile verilir. Eşitlikte e(n) (artık sinyal) ve $W_e(n)$ penceredir. (2.5.7) nin minimum yapılması (2.5.8) eşitliklerini sağlar.

$$\sum_{k=1}^{p} a(k).R(i,k) = -R(0,i) \qquad 1 \le i \le p \qquad (2.5.8)$$

$$R(i,k) = \sum_{n=-\infty}^{\infty} W_e(n).S(n-i).S(n-k) \qquad (2.5.9)$$

En çok kullanılan dirdörtgen penceredir.

$$W_e(n) = \begin{cases} 1 & p < n < N-1 \\ 0 & \text{diğer durumlarda} \end{cases} \qquad (2.5.10)$$

bu durumda (2.5.9) (2.5.11)'e dönüşür.

$$R(i,k) = \sum_{n=p}^{N-1}.S(n-i).S(n-k) \qquad 0 \le i, k \le p \qquad (2.5.11)$$

R(i,k), S(n) işaretinin kovaryansı için bir tahmin sağlandığından, bu yöntem kovaryans yöntemi olarak bilinir.

(2,5,8) eşitlikleri p bilinmeyenli p adet lineer eşitlikten oluşur. Bu eşitliklerden öngörü katsayıları çözülebilir. (2,5,8)'i (2,5,7) de kullanarak minimum artık enerji veya minimum öngörü hatası;

$$E_p = R(0,0) + \sum_{k=1}^{p} a(k).R(0,k) \qquad (2.5.12)$$

olarak bulunur.

(2.5.11)'de S(n) $0 \le n \le N-1$ aralığında bilinmelidir. Aynı özilişki yönteminde olduğu gibidir, ancak eşitliklerin çözülmesi farklıdır.

3. LPC DENKLEMLERİNİN ÇÖZÜMÜ

Özilişki yönteminde bulunan (2.3.12) denklemi,

$$\sum a_k . R_n(i-k) = R_n(i) \qquad (i = 1, 2, \ldots\ldots, p)$$

kovaryans yönteminde bulunan (2.3.16) denklemi ise,

$$\sum_{k=1}^{p} a_k . \phi_n(i,k) = \phi_n(i,0) \qquad (i = 1, 2, \ldots\ldots, p)$$

idi . Bu ifadeler P sayıda bilinmeyen a_k için p sayıda denklemden oluşmaktadır. Amaç herhangi bir yöntemle bu denklem takımlarını çözüp a_k katsayılarını elde etmektir. Her iki yöntemdeki katsayılar matrisi [R_n] ve [ϕ_n]'nin değişik özellikleri olduğundan uygun çözüm yöntemleri uygulanarak sonuç elde edilmeye çalışılacaktır, (Ölçer, 1993).

3.1.Yinelemeli (recursive) Durbin Çözüm yöntemi

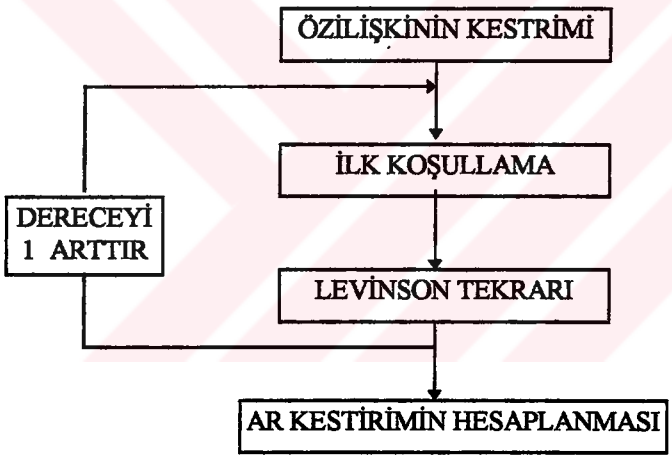
(2.3.13) denkleminde bilinmeyen süzgeç katsayıları a_1 , a_2 , ... a_m bilinen özilişki ifadelerinden (R (0) R (1),.....)'den hesaplanır. Matris simetrik bir matristir.

a_j katsayılarının hesabı için soldaki özilişki matrisinin tersini bulmak gerekir. Süzgeç boyu M büyük olduğunda alışılagelmiş yöntemlerle süzgeç katsayılarının hesabı için gerekli bilgisayar zamanı katsayı sayısının küpüyle orantılıdır.

Levinson'un önerdiği ve daha sonra geliştirilen tekrarlamalı yöntemle bu oran katsayı sayısının karesine kadar indirgenmiştir. Tekrarlamalı yöntem özilişki matrisinin bütün köşegenleri boyunca elemanların eşit olmasından yararlanarak uygulanır.

Aşağıda gösterildiği gibi m = 0 dan m = M'e kadar katsayılar hesaplandığında m+1 inci adımdaki katsayıların bunlardan nasıl hesaplanacağı gösterilecektir. m+1 katsayısının hesabı için m = 0 ilk adım m = M son adım olacaktır. Takılardan ilki adım sayısını gösterir. İkincisi ise o elemanın, gösterilen numaralı katsayısıdır. Örneğin m=M son adımda ise a(m,0) = a(M,0) aranan a(0) katsayısıdır. Her adımda süzgeç boyuna bağlı yanılgı hesaplanabildiğinden, istenildiğinde süzgeç boyunu sabit alma yerine yanılgı belirli bir düzeye ulaştığında tekrarlama işlemi durdurulabilir. Bunun için ya belirli sınır değerler alınır veya süzgeç boyu uzadıkça sağlanan yanılgı azalması önemli olmadığında işlemler durdurulur.

Algoritma şu şekilde özetlenebilir.



Şekil 3.1.1 Durbin - Levinson Algoritması

Özilişki matrisi bir Toeplitz matrisidir, yani simetrik ve esas köşegene paralel köşegenler üzerindeki elemanlar benzerdir. Bu matrisin çözümü için en iyi bilinen çözüm algoritması Levinson - Robinson algoritması ise de daha etkili bir yöntem, yinelemeli Durbin çözüm algoritmasıdır. Yukarıda blok diyagram olarak gösterilen bu algoritma aşağıdaki bağıntılarla tanımlanmıştır ve Levinson - Durbin adı ile bilinmektedir (Rabiner et al, 1978).

1) Özilişki katsayıları daha önce verilen formda hesaplanır.

$$R(i) = \frac{1}{N} \cdot \sum_{n=1}^{N} S_n . S_{n+i} \qquad i = 0, 1, \ldots\ldots\ldots\ldots, p \qquad (3.1.1)$$

p filtrenin derecesi

2) İlk koşullama için şu şekilde başlangıç değerleri belirlenir.

$$a(0, 0) = 1, \qquad E_0 = R(0) = E[S^2_n]$$

3) Bu verilerle, aşağıdaki algoritma kullanılarak, a(m,k) değerleri bulunur. Bunlar m. nci adımda bulunan filtrenin a_k değerleridir.

Algoritma

$$E_0 = R(0) \qquad (3.1.2a)$$

$$k_i = [R(i) - \sum_{j=1}^{i-1} a_j(i-1).R(i-j)] / E_{i-1} \qquad (3.1.2b)$$

$$a_i(i) = k_i \qquad (3.1.2c)$$

$$a_j(i) = a_j(i\text{-}1) + k_i \, . \, a_{i-j}(i\text{-}1) \;, \qquad 1 \le j \le i\text{-}1 \qquad (3.1.2d)$$

$$E_i = (1\text{-}k_i)^2 \, . \, E_{i\text{-}1} \qquad (3.1.2e)$$

3.1.2a-3.1.2e yinelemeli olarak çözüldükten sonra, son çözüm

$$a_j = a_j(p) \qquad 1 \le j \le p \qquad (3.1.2f)$$

bağıntısı ile elde edilecektir. Ayrıca gerektiğinde 3.1.2.h eşitliği de kullanılabilir.

$$a_{i,i} = k_i \qquad (3.1.2h)$$

Yukarıda eşitlikleri verilen (3.1.2) denklemlerinde;

k_i: Yansıma katsayıları (salt - kutup Lattice filtresinde kullanılan katsayıların ters işaretleri)

a_j: LPC filtresi için bir sonraki giriş işaretine ilişkin filtre katsayıları.

R (i): Giriş işaretinin özilişki fonksiyonu.

E_i: Gerçek giriş örneği ile elde edilen yaklaşık örnek arasındaki farkların karelerinin ortalamasıdır. Filtrenin kararlı (stable) olması için ,

$$| k_i | < 1$$

şartının sağlanması gerekir. Bu durumda a_j değerleri doğrusal öngörü süzgecinin katsayıları olacaktır.

Minimum hata E_m, öngörü derecesi arttıkça azalır. $E_m = (1-k^2_m) E_{m-1}$ idi ve $\{a_m (k), 1 \le k \le m \}$ m. dereceden öngörü katsayılarıdır. Bunlara göre:

$$E_m = R(0).\prod_{i=1}^{m}(1-k_i^2) \qquad (3.2.3)$$

yazılabilir.

Minimum hata E_m, öngörü derecesi arttıkça azalır (3.1.3)'den tüm kutuplu filtre için kararlılık şartı olarak;

$$| k_i | < 1 \qquad 1 \le m \le p \qquad (3.1.4)$$

koşulu elde edilir.

K_m katsayıları yansıma katsayıları olarak bilinir. (3.1.4) koşulu sağlandığı zaman tüm kutuplar birim dairenin içindedir. Filtre kararlılığı söz sentezinde çok önemlidir. Kararlı olmayan bir filtre söz çıkışında istenmeyen seslere neden olabilir.

$| k_i | = 1$ durumunda tüm kutuplar birim dairenin üzerine düşer. Bu da kararsız bir durumdur.

(3.1.3.) de E_p'yi R(0) enerjisine bölerek normalize minimum hata elde edilir;

$$V_p = \frac{E_p}{R(0)} = \prod_{m=1}^{p}(1-k_m^2) \qquad (3.1.5)$$

(3.1.4) ve (3.1.5)'den

$$0 < V_p \leq 1 \qquad (3.1.6)$$

koşulu elde edilir.

Eğer $| k_m | = 1$ ise, hatanın sıfır olduğunu görürüz, bu da işaretin mükemmel olarak öngörülebileceği anlamını taşır. Ancak pencerelenmiş bir işaret için bu şartı elde etmek imkansızdır.

Kovaryans yönteminde filtre kararlılığı garantili değildir. Belli bir öngörü sonucunun kararlı bir filtreye götürüp götüremeyeceğini test etmek için,

$$a_m(k) = a_{m-1}(k) - k_m \, . \, a_{m-1}(m-k) \qquad 1 \leq k \leq m-1 \qquad (3.1.7)$$

(3.1.7)'den k_m'i hesaplayan ters bir rekürsiyon türetilebilir:

$$a_p(k) = a(k) \qquad 1 \le k \le p$$

$$k_m = a_p(p) \qquad m = P, P-1, \ldots\ldots\ldots\ldots, 2 \text{ için}$$

$$a_{m-1} = \frac{a_m(i) - k_m . a_m(m-i)}{1 - k_m^2} \qquad 1 \le i \le m-1$$

$$k_{m-1} = a_{m-1}(m-1) \qquad (3.1.8)$$

Eğer tüm k_m ler (3.1.4)'e uyarsa H (z) kararlıdır; aksi halde değildir.

Kafes Süzgeçler, ileri yönde tahmin hata süreçleri TDL yapı (yani doğrusal tahmin ileri yönde kullanıldığında tahmin edilecek olan S_n değeri geçmişteki değerlerin doğrusal olarak birleştirilmesi ile elde edilir.) aynı anda gerçeklenmek istenirse iki ayrı süzgece ihtiyaç duyulacaktır. Üstelik süzgeç mertebesi değişirse, süzgeç katsayıları da tekrar hesaplanmak durumundadır. Oysa 1-B kafes süzgeç yapısı bu olumsuzlukları içermeksizin ileri ve geri yöndeki tahmin hata süzgeçlerini tek bir yapıda birleştirir.

Süzgeç yapısında kullanılan mertebe sayısı tahmini hata süzgecinin derecesine eşittir. Kafes süzgeç ileri ve geri yöndeki tahmini hata dizilerini aynı anda üretir.

3.2. Cholesyky Ayrışımı (Decomposition) Çözüm Yöntemi

Bu yöntem kovaryans yönteminden elde edilen denklem takımına uygulanacak bir yöntemdir. Daha önce bulunan (2.3.16),

$$\phi(i,0) = \sum_{k=1}^{p} a_k \phi(i,k) \qquad 1 \le i \le p$$

denkleminin matris şeklinde yazılımı aşağıdaki gibidir;

$$\phi \,.\, a = \Psi \qquad (3.2.1)$$

ϕ simetrik bir matristir. Bu matris,

$$\phi = V \,.\, D \,.\, V^T \qquad (3.2.2)$$

şeklinde gösterilebilir. V esas köşegen elemanları 1 olan bir alt üçgen matris, D bir köşegen matris V^T ise V matrisinin evriği (transpozu) dur. Buna göre (3.2.2) bağıntısının açık şekli aşağıdaki gibi olacaktır.

$$\phi_n(i,j) = \sum_{k=1}^{j} V_{i,k} \,.\, d_k \,.\, V_{j,k} \qquad j = 1, 2, \ldots\ldots\ldots\ldots i\text{-}1 \qquad (3.2.3)$$

$$V_{i,j} \,.\, d_j = \phi_n(i,j) - \sum_{k=1}^{j-1} V_{i,k} \,.\, d_k \,.\, V_{j,k} \qquad j = 1, 2, \ldots\ldots\ldots\ldots i\text{-}1 \qquad (3.2.4)$$

köşegen üzerindeki elemanlar ise,

$$\phi_n(i,j) = \sum_{k=1}^{j-1} V_{i,k} \,.\, d_k \,.\, V_{i,k} \qquad (3.2.5)$$

veya

$$d_1 = \phi_n\,(\,1,1\,) \qquad (3.2.6)$$

olması koşuluyla,

$$d_i = \phi_{i,i} - \sum_{k=1}^{i-1} V_{i,k}^2 \,.\, d_k \qquad (3.2.7)$$

dır. [V] ve [D] matrisleri saptandıktan sonra [a_k] sütun vektörünü elde etmek oldukça kolaydır. (3.2.1) ve (3.2.2) bağıntılarından

$$V.D.V^T.a = \Psi \qquad (3.2.8)$$

$$V.Y = \Psi \qquad (3.2.9)$$

$$D.V^T.a = Y \qquad (3.2.10)$$

$$V^T.a = D^{-1}.Y \qquad (3.2.11)$$

$$Y_i = \Psi_i - \sum_{j=1}^{i-1} V_{i,j}.Y_j \qquad 2 \le i \le P$$

buradan başlangıç koşulu $Y_1 = U_1$ dir.

$$a_i = \frac{Y_i}{d_i} - \sum_{j=i+1}^{p} V_{j,i}.a_j \qquad 1 \le i \le P\text{-}1 \qquad (3.2.12)$$

(3.2.12) denkleminden yararlanarak (3.2.11) denklemindeki a yinelemeli olarak çözülebilir. Burada başlangıç koşulu $a_p = V_p / d_p$ dir.

4. KONUŞMA İŞARETLERİNİN ÖZELLİKLERİNİN SAPTANMASI

4.1. Konuşma İşaretlerinin Özelliklerinin Saptanması

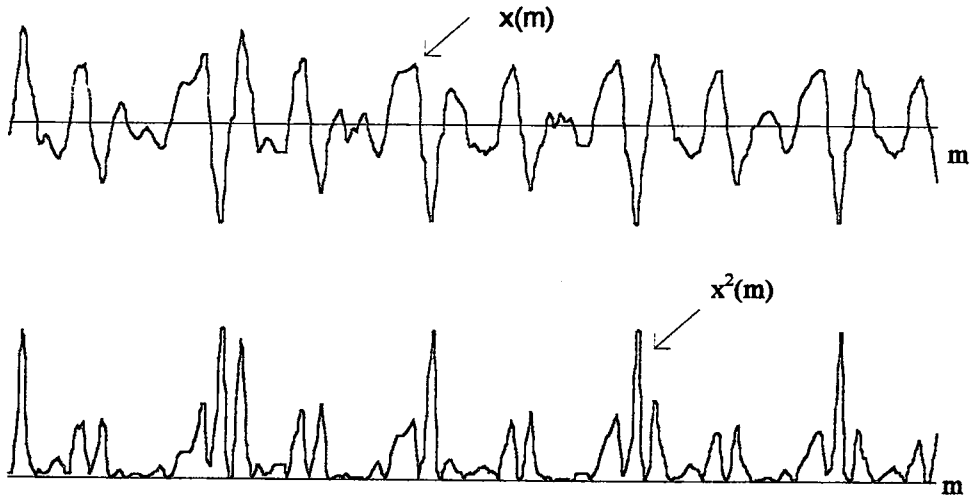
Doğrusal öngörü analizinde esas amaç, konuşma işaretlerinin özelliklerini en iyi şekilde yansıtmaktadır. Bu nedenle de bu işaretin tüm özelliklerinin bilinmesi, örneğin işaretin sesli-ses, sessiz-ses yada sessizlik olduğunu ortaya çıkarmak gerekir. Bunun için değişik yöntemler mevcuttur.

4.2. Kısa Zaman Süreli İşaret Enerjisinin Bulunması

Genelde konuşma işaretinin özelliklerinin zamana bağlı olduğu, sesli ve sessiz-seslerde uyarı şeklinin değişik olduğu bilinmektedir. Bu seslerin genlikleri ve temel frekansları arasında belirgin farklılıkları vardır. Zaman ortamında bu farklılıkları ortaya çıkarmak mümkündür. Yine de konuşma işaretindeki özelliklerin zaman içerisindeki değişimlerin nispeten yavaş olduğu varsayılmaktadır. Bu nedenle daha önceki bölümlerde de değinildiği gibi konuşma işareti incelenirken kısa zaman süreli (short time) analiz yöntemlerinden yararlanılarak, konuşma işaretinin kısa süreli parçaları incelenir. Böylece kısa zaman süreli bir işaret parçasının kısa süreli enerjisi,

$$E_n = \sum_{m=n-N+1}^{n} x^2(m) \qquad (4.1)$$

bağıntısı ile hesaplanır. Burada n. örnekteki enerji, n-M+1 ile n arasındaki N örneğin karelerinin toplamından ibarettir. Kısa zaman süreli bir işaret parçasının ele alınması esas işareti w(n) gibi bir pencere ile pencerelemek anlamına gelmektedir. Bu şekil 4.1'de gösterilmiştir.



Şekil 4.1. Kısa süreli enerjinin gösterimi

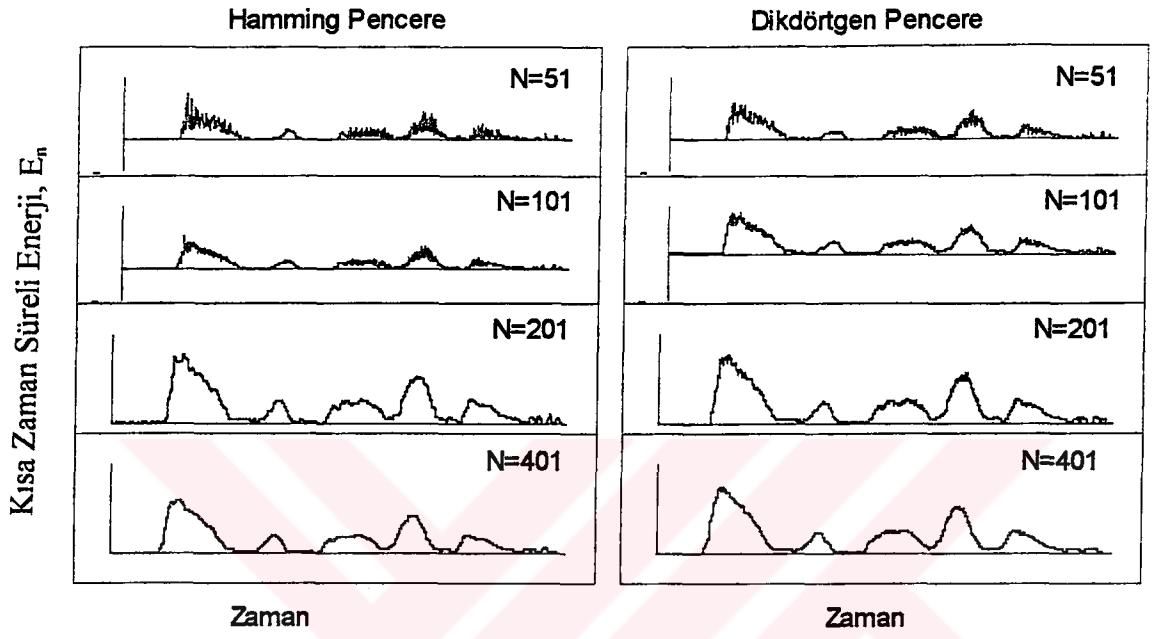
E_n, sesli-ses ile sessiz-ses parçalarının ayrımında çok önemli bir rol oynar. Konuşma işaretinde genelde sessiz-ses parçasının genliği, sesli-ses parçasının genliğinden çok küçüktür. Bu nedenle E_n, bu iki tür işareti birbirinden ayırt etmekte kullanılır, (Yıldırım, 1995).

Kısa süreli enerjiyi şu şekilde yazmak mümkündür.

$$E_n = \sum[x(m).w(n-m)]^2 = \sum[x^2(m).h(n-m)] \qquad (4.2)$$

burada $h(n)=w^2$ dir. Bu pencere dikdörtgen veya başka bir pencere olabilir. Ancak, kısa zaman süreli enerji hesabında, göz önünde bulundurulması gereken parametre kullanılan pencere uzunluğudur. Sesli-sesler periyodik dürtülerden oluştuğu için tekrarlanma periyodu (pitch) vardır. Bu periyot değerini yakalayabilmek için pencere boyu iyi seçilmelidir. Eğer pencere boyu N çok küçükse E_n, işaretin dalga şekline bağlı dalgalanmalar gösterir. Büyük ise E_n, çok yavaş değişeceğinden konuşma işaretinin

özelliklerini yansıtmayacaktır. Şekil 4.2'de değişik pencere uzunluklarına göre kısa zaman süreli enerji fonksiyonları çizilmiştir.



Şekil 4.2. "O Ne Söyledi" cümlesinin kısa zaman süreli enerji fonksiyonu (Hamming penceresi ve dikdörtgen pencere)

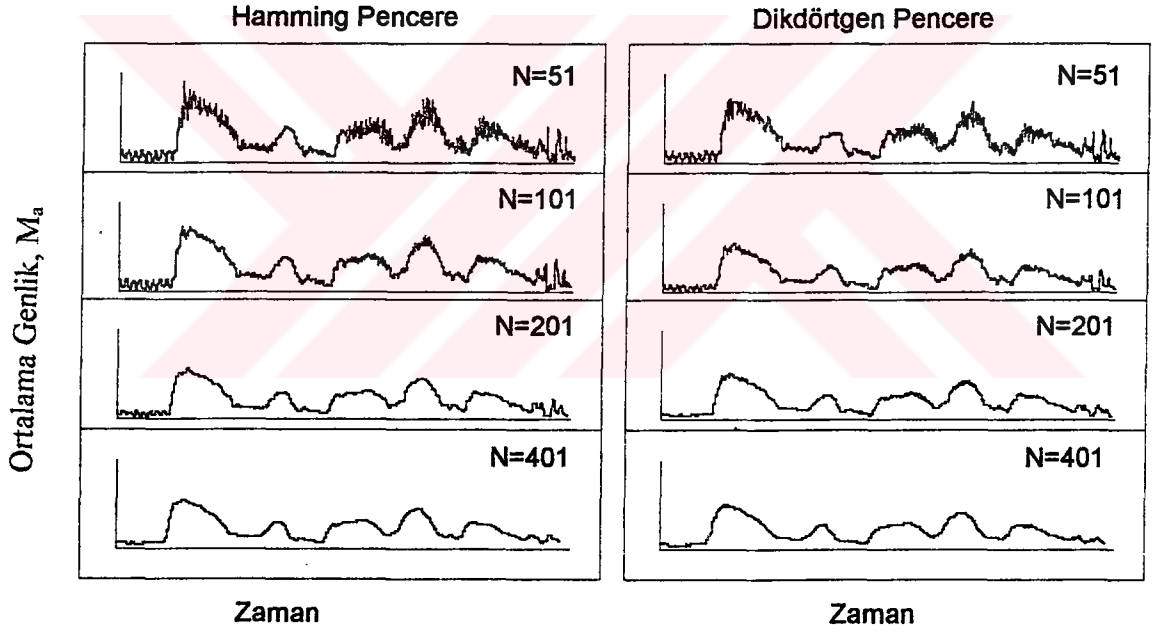
Burada pencere boyu N değeri artıkça enerji daha yumuşak kenarlı olmaktadır. N, 10 kHz'lik bir örnekleme aralığı için 20 örnekten 250 örneğe kadar değişmektedir. N, 10 kHz frekans örneklemesi için 100-200 örnek arasında seçilmelidir. Ayrıca enerjinin değeri büyük olduğu kısımlar sesli sesler, küçük olduğu kısımlar ise sessiz sesler olarak ayırt edilebilir. Bundan başka enerjinin sıfır olduğu değerlerde sessizlik olan kısımlardır.

4.3. Ortalama Büyüklük

Kısa zaman süreli enerji hesabı yerine ortalama büyüklük fonksiyonu kullanılabilir. Bu sayede kare alma işleminden kurtulmuş oluruz. Ortalama büyüklük ifadesi;

$$M_a = \sum |x(m).w(n-m)| \qquad (4.3)$$

dir. Görüldüğü gibi kare alma işlemi yapılmamaktadır. Sesli-ses ile sessiz-ses arasındaki fark kısa süreli enerji fonksiyonundaki kadar olmasa da istenilen sonucu sağlamaktadır. Şekil 4.3'de verilen aynı konuşma için değişik pencere uzunluklarına göre ortalama büyüklük eğrilerinin çizimleri yapılmıştır.



Şekil 4.3. "O Ne Söyledi" cümlesinin ortalama büyüklük fonksiyonları (Hamming penceresi ve dikdörtgen pencere)

Her iki yöntemde de farklı pencereler kullanılabilir. Ayrıca fonksiyon daha az örnek sayıda alınsa da gerekli sonucu sağlayabilmektedir. Mesela 20 K örnek/s için

pencere uzunluğu N=20 ms (200 örnek) seçilse, ortalama büyüklük ya da kısa süreli enerji, 100 örnek/s alınabilir.

4.4. Ortalama Sıfır Eksenini Kesme Düzeyi

İşaretin frekans özellikleri ile işaretin sıfır eksenini kesme sayısı arasında bir ilişki vardır, örneğin sinüs eğrisinde bir periyodu da sıfır ekseni iki kez kesildiği göz önüne alınırsa sıfır ekseni kesme sayısı,

$z=2.f_o/f_a$

olacaktır (Köseoğlu, 1988). Bir pencere için ifadeyi daha da genelleştirirsek;

$$z_n = \Sigma \mid sgn [\, x(m) \,] - sgn [x(m-1)] \mid w(n-m) \qquad (4.4)$$

Burada,

$$sgn [\, x(n) \,] = 1 \qquad x(n) \geq 0 \qquad (4.5)$$
$$sgn [\, x(n) \,] = -1 \qquad x(n) < 0$$

ve

$$w(n) = \frac{1}{2.N} \qquad 0 \leq n \leq N-1 \qquad (4.6)$$

dır. Buradan anlaşılacağı üzere yöntem ardışık örnekleri ikişer ikişer alıp sıfır kesme olayını araştırıp saptıyor, eğer sıfır ekseni kesiliyorsa kesme sayısı bir arttırılıyor. Bu işlem N ardışık örnek için tekrarlanıyor.

Yüksek frekanslarda sıfır kesme sayısı alçak frekanslara nazaran daha fazladır. Dolayısıyla enerji ile bu fonksiyonun bir bağlantısı vardır. O halde bu yöntemde sesli ve

sessiz-seslerin ayırımında kullanılabilir. Sıfır ekseni kesme sayısı büyük ise konuşma işareti sessiz-ses, aksi taktirde sesli-ses olduğuna kanaat getirilebilir. Ancak bu olay gürültü yüzünden bozulabileceğine dikkat edilmelidir.

4.5. Konuşma ile Sessizliğin Ayrılması

Konuşma başlamadan önce bir sessizlik bölgesi vardır. Aynı şey konuşma sonunda da gerçekleşir. Bu olay süreyi konuşma bölgesiymiş gibi analiz etmek gereksiz hesaplama yapmak demektir. Bu nedenle bu lüzumsuz işten kurtulmak için sessizlik bölgelerinin konuşmadan ayıklanması yerinde olur. Böylece konuşmanın sınırları da belirlenir. Bu işlev gerçeklemek için enerji fonksiyonunu kullanmak mümkündür. Enerji işaretinin sıfırdan ayrıldığı nokta ile sıfıra döndüğü noktaya kadar bir konuşma yapılmış olarak kabul edilebilir. Ancak bu durumu aşırı gürültü bozabilir. Ayrıca konuşmanın başı f, h, k, p, t ile başlarsa veya sonu m, n, v, z ile biterse hem enerjiyi hem de sıfır geçiş hızını hesaplayarak karşılaştırmak yerinde olur.

4.6. Tekrarlanma (Pitch) Periyodunun Saptanması

Sesli-seslerin periyodik dürtülerden oluştuğu sessiz seslerin de beyaz gürültüden oluştuğu söylenmişti. Sesli-seslerdeki bu dürtülerin tekrarlanma periyodunu (Pitch periyot) bulmak için yöntemler vardır. Tekrarlanma periyodunun bulunmasında kullanılan yöntemlerden biri Değiştirilmiş Özilişki Yöntemidir.

a_i katsayılarının özilişki fonksiyonu

$$R_a(j) = \sum_{i=1}^{p} a_j . a_{j+i} \qquad (4.7)$$

ve giriş işaretinin özilişki fonksiyonu

$$R_x(k) = \sum_{m=0}^{N-k-1} x(m).x(m+k) \tag{4.8}$$

ise Değiştirilmiş Özilişki Fonksiyonu

$$R_a(k) = \sum_{j=1}^{P} R_a(j).R_x(k-j) \tag{4.9}$$

şeklinde tanımlanmaktadır. Burada k=0'dan pencere uzunluğuna kadar alınacaktır.

Daha sonra $R_e(k)$ / $R_e(0)$ alınarak fonksiyon normalleştirilir. Sonra bunun en büyük değeri aranır. Ancak bu işlem bir pencerenin % 25 ile % 75'i arasında yapılır. Bu aralıkta en büyük genlik 0.25'den büyük ise pencere sesli-ses olarak alınır. Tekrarlanma periyodu olarak bu değerin indisi alınır. Genlik 0.25'den küçük ise pencere sessiz-ses olarak alınır. Tekrarlanma periyodu da 0 olarak alınır.

Sesli sesler için özilişki değerleri büyük ve periyodik olarak gözükürken, sessiz ses için aynı şey söylenemez.

Özilişki fonksiyonunun çift olma özelliğinden faydalanarak

$$R_n(k) = R_n(-k) = \sum_{m=-\infty}^{\infty} x(m).x(m-k).[w(n-m).w(n+k-m)] \tag{4.10}$$

yazılabilir.

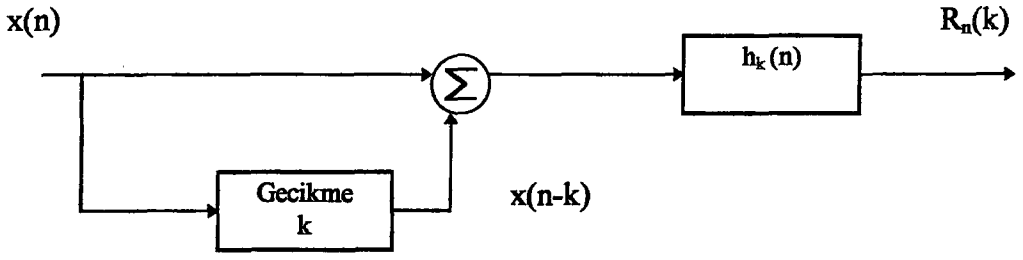
$$h_k(n) = w(n) \,.\, w(n+k) \tag{4.11}$$

şeklinde tanımlanırsa,

$$R_n(k) = \sum_{m=-\infty}^{\infty} x(m).x(m-k).h_k(n-m) \qquad (4.12)$$

elde edilir.

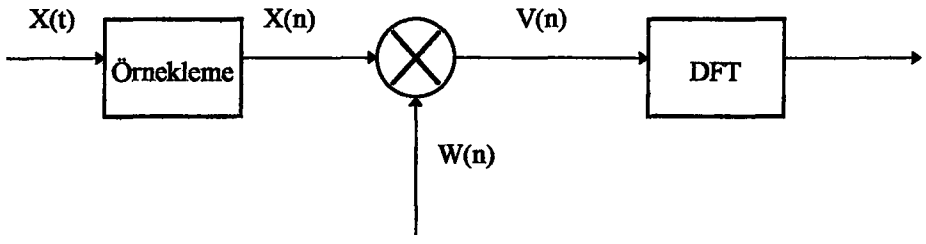
Bu da, n zamandaki k özilişki değerinin [x (n).x(n-k)]'yi dürtü tepkisi $h_k(n)$ olan bir filtre ile filtrelemek anlamına gelmektedir. Bu şekil 4.4 da gösterilmektedir.



Şekil 4.4. Kısa zaman süreli özilişkinin blok diyagramı

4.7. Fourier Dönüşüm Süzgeç Grup Yöntemi

Fourier Dönüşüm Süzgeç Grup yönteminde konuşma işaretinin ayrık Fourier dönüşümü (DFT) hesaplanır ve oluşan spektral kanallar istenen sayıda bileşene göre gruplandırılır. DFT sürekli-zaman işaretlerin analizinde önemli bir uygulamaya sahiptir.



DFT' nin hesaplanabilmesi için sonlu sayıda işaret dizisi gerekmektedir. Bunun için sayısal işaret olan x(n), sonlu uzunlukta w(n) penceresi ile çarpılır.

$$v(n) = x(n) .w(n)$$

Bu çarpma işleminin frekans domeninde etkisi periyodik bir konvolusyondur.

$$V(e^{jw}) = \frac{1}{2\pi}\int_{-\pi}^{\pi} X(e^{j\theta})W(e^{j(w-\theta)})d\theta$$

Burada $X(e^{jw})$ x (n) işaretinin, $W(e^{jw})$ w(n) penceresinin fourier dönüşümüdür. $W(e^{jw})$ ile $X(e^{jw})$in konvolusyonu $X(e^{jw})$ in keskin tepelerini ve süreksizliklerini yumuşatır. Pencerelenmiş bir v (n) dizisinin DFT si aşağıdaki formülle verilir.

$$V(k) = \sum_{n=0}^{N-1} v(n)e^{j(2\pi/N)kn} \qquad ; k = 0, 1,.....,N-1$$

Burada görüleceği üzere pencerelenmiş v(n) işaretinin DFT'si v(n)'in FT'sinin eşit aralıklarla örneklenmesinden elde edilir. Buna spektral örnekleme denilir ve aşağıdaki ifadeyle gösterilir.

$$V(k) = V(e^{jw})|_{w=2\pi k/N}$$

DFT'yi kullanarak işaretin analiz yapılmasında pencerelemenin ve spektral örneklemenin önemli bir etkisi vardır. İşaret işlemede en çok kullanılan pencereler Hamming, Hannig, Kaiser'dir. Öğretici amaçla en çok kullanılan ise dikdörtgen penceredir. İşaret analizinde pencere boyunun uygun seçilmesi gerekmektedir. Pencere boyu kısa seçilirse pencerenin frekans domenindeki bant genişliği azalacak yani çözünürlük artacak fakat zaman domeninde fazla sayıda çarpma işlemi olacağından belli bir zaman gecikmesi olacaktır. Pencere seçiminde dikkat edilmesi gereken nokta pencerenin frekans domenindeki ana bileşeninin dar ve yan bileşenlerinin genlik olarak düşük olmasıdır. Ana bileşen ile birinci yan bileşen arasındaki genlik farkı pencere

boyundan bağımsız olup pencere şekline bağlıdır. Daha önce bahsedildiği gibi DFT, N eşit aralıkla işaretin fourier dönüşümünün ayrık zaman frekanslarında örneklenmesidir. Bu olaya spektral örnekleme denir. Spektral örnekleme bizi bazen yanlış sonuçlara götürebilir. Mesela spektral örnekleme aralığı geniş seçilirse yani N değeri küçükse, işareti oluşturan frekansların tepe değerleri DFT'den elde edilen iki spektral arasında kalabilir. Bu durumda DFT değerlerindeki tepelerin yerleri ile Fourier dönüşümündeki aynı frekanslardaki tepeler uyuşmayacaktır. Keskin ve büyük genlikli frekans bileşenleri DFT de küçük genlikli gözükecektir ve bu durum bizi yanıltacaktır. Bunun için DFT yi hesaplamada N nokta sayısını arttırmak yani örnekler arasını azaltmak gerekecektir, (Gücümoğlu, 1995).

5. YAPAY SİNİR AĞLARI (YSA)

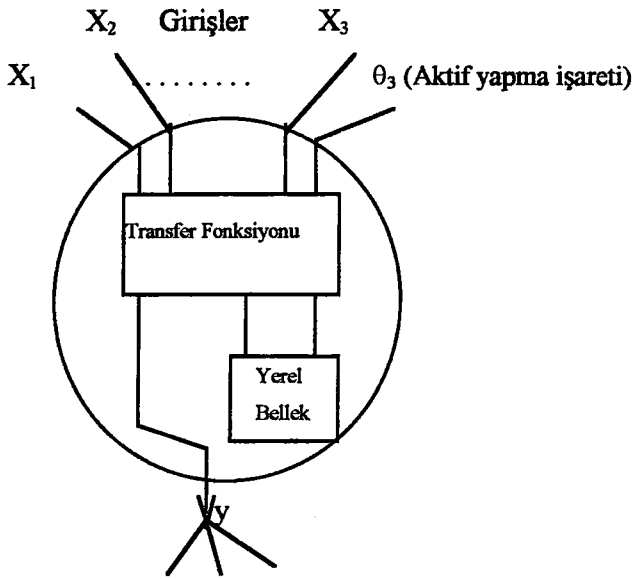
5.1. YSA'nın Tanımı

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, (Karlık, 1994).

5.2. YSA'nın Yapısı ve İşlem Elemanı

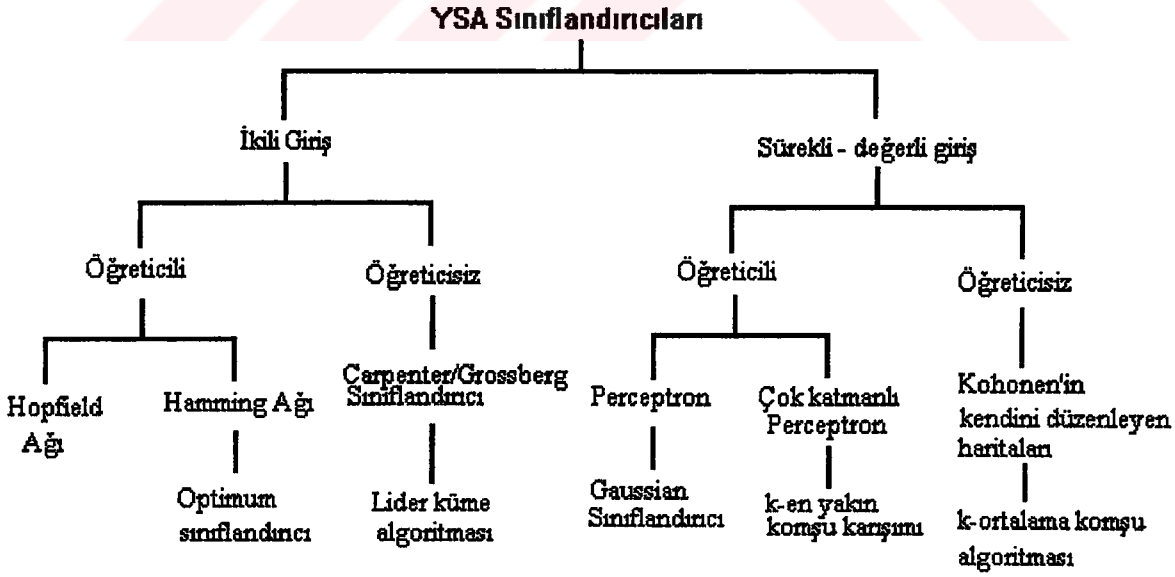
YSA temel olarak, basit yapıda ve yönlü bir graf biçimindedir. Her bir düğüm hücre denilen n. dereceden lineer olmayan bir devredir. Düğümler işlem elemanı olarak tanımlanır. Düğümler arasında bağlantılar vardır. Her bağlantı tek yönlü işaret iletim yolu (gecikmesiz) olarak görev yapar. Her işlem elemanı 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. Şekil5.1. 'de genel bir işlem elemanı (nöron, düğüm) gösterilmiştir.

Her işlem elemanı kendisine verilen yerel veriye göre, kendisini ayarlayacak bütün YSA'nın enformasyon bölgesinin öğrenmesini sağlar. (Enformasyon bölgesi olasılık-yoğunluk fonksiyonu ile de 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.



Şekil 5.1. Genel işlem elemanı yapısı

YSA birtakım alt kümelere ayrılabilir. Bu alt kümelerdeki elemanların transfer fonksiyonları aynıdır. Bu küçük gruplara katman ("layer") adı verilir. (örn: "multilayer perceptron, MLP") Ağ katmanların birbirlerine hiyerarşik bir şekilde bağlanmasından oluşmuştur. Dış dünyadan alınan bilgi, giriş katmanı ile taşınır. Giriş katmanlarının bir transfer fonksiyonları yoktur. YSA transfer fonksiyonu ve yerel bellek elemanı bir



Şekil 5.2. YSA' nın sınıflandırıcıları

öğrenme kuralı ile giriş çıkış işareti arasındaki bağıntıya göre ayarlanır. Aktif yapma girişi için bir zamanlama fonksiyonu tanımlaması gerekebilir.

Bir işlem elemanına gelen girişler matematiksel tiplerine göre etiketlendirilerek sınıflandırılır. YSA, giriş veri tiplerine göre ikili giriş (0,1) ve sürekli değerli giriş olmak üzere Şekil 5.2.'deki gibi sınıflandırılır.

Bu tezde giriş işareti, sürekli-değerli (reel sayı) olduğundan dolayı, öğreticili öğrenmeye sahip olan çok katmanlı idrak ("perceptron", almaç) kullanılmıştır.

5.3. Bağlantı Geometrileri

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

$w_{ij} = w_{ji} = 1 \Leftarrow$ 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 belgesi S:kaynak bölgesi

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

5.4. Eşik Fonksiyonları

Transfer veya işaret fonksiyonları olarak da adlandırılan eşik fonksiyonları, muhtemel sonsuz domen girişli işlem elemanlarını önceden belirlenmiş sınırda çıkış olarak düzenler. Üç tane yaygın eşik fonksiyonu vardır. Bunlar, rampa, basamak ve sigmoid fonksiyonudur. Bu çalışmada eşik fonksiyonu olarak kullanılan S biçimindeki sigmoid fonksiyonu; seviyeli, lineer olmayan çıkış veren, sınırlı, monoton artan fonksiyondur.

5.5. Ağırlık Uzayı

Bir çok YSA öğrenme işlemi, işlem elemanlarının 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;

$$w = \left(w_1, w_{12}, \ldots, w_{1n}, w_{21}, w_{22}, \ldots, w_{2n}, \ldots\ldots, w_{N1}, w_{N2}, \ldots, w_{Nn}\right)^T$$

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

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

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

YSA ağırlık vektörü N, n boyutlu oklid uzayında yayılır. YSA'nın enformasyon işleme performansı, ağın ağırlık vektörünün belirli bir değeri ile bulunacaktır.

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 performansını verecek noktaya yöneltmektir. Genellikle öğrenme kuralı için bir performans yada maliyet fonksiyonu tanımlanır. Minimizasyon veya maksimizasyon ile "w" vektörü bulunur. Bir performasyon çeşidi olarak bilinen, MSE (karesel ortalama hata) şu şekilde tanımlanır.

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

Burada amaç F'i küçültmeye çalışmaktır.
y=G(w,x):sistemin giriş çıkış fonksiyonu.
y:çıkış işareti vektörü
x:giriş işareti vektörü
w:ağırlık vektörü
ρ(x):olasılık yoğunluk fonksiyonu

5.6. YSA'da Eğitme ("Training")

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.

5.7. Bellek

YSA'nın önemli bir özelliği bilgiyi saklama şeklidir. YSA'da bellek 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 YSA bellekleri 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.

5.8. 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. 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.

5.9. 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. Günümüzde kullanılan birçok öğrenme kuralı vardır. Bilinen en çok kullanılan öğrenme kuralları şunlardır.

- Raslantısal (Hebb) öğrenme kuralı
- Performans (Widrow ve ADALİNE) öğrenme kuralı
- Kompetif (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.

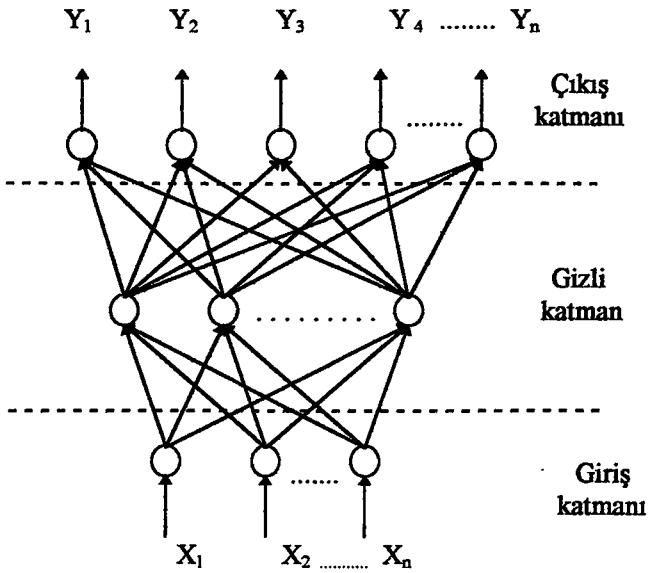
5.10. Perceptron (idrak,almaç)

Perceptron ağı, ilk 1943 yılında Mc Culloch ve Pitts tarafından saptandı. Bu ilk perceptron 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. Ama Minsky ve Papert bu tek katmanlı perceptronun XOR (ayrıcalıklı veya) işlemini gerçekleştiremediğini ispatladılar. 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. Bilgi lineer yayılamıyorsa 2.katmanın çıkışı konveks bölgededir bunun neticesi olarak 3.katmandan gelecek olan çıkış bilgisinin şekli, herhangi bir bölgenin şeklinde olabilir. Bu sebeple ihtiyaç duyulan katman sayısı üç olmaktadır. Üç-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 yada 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. Tek katmanlı perceptron uygulanan her eğitimin seti modelinin en önemli özelliği, lineer biçimde dağılmak zorunda olmasıdır.

5.11. Çok Katmanlı İdrak (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 5.3'de çok katmanlı perceptronun genel yapısı verilmiştir.



Şekil 5.3. 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 kartşılştırılarak sonuçta gerekirse hata işaret edilir. Hata işareti gizli birimlerden çıkış birimine olan ağırlıkları değiştirmekte kullanılır. Ama bunu yaparken giriş katmanından gizli katmana gelen değiştirilip değiştirelemediğini düşünmek gerekir. Gizli birimlerden ne tür bir çıkış istendiği bilinemeyeceği için gizli birimlerin çıkışında hata işareti verilmesi koly 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. Şimdi bu algoritmayı inceleyelim.

5.12. 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. Algoritmaya göre her bir j biriminin çıkışı o_j şu şekilde tanımlanır;

$$o_j = f(net_j) = f(x) \quad ise \quad net_j = \sum_i^j w_{ji} o_i + \theta_j \qquad (5.1)$$

Burada o_i; i. biriminin çıkışı w_{ji}; j biriminden i birimine bağlantının ağırlığı, θ_j; j biriminin kutbu (bias), Σ_i; çıkışı j birimine akan her i biriminin toplamıdır. f(x) bir monoton artan ve türevi alınabilen fonksiyondur. Pratikte bir lojistik aktivasyon fonksiyonu olarak $f(x)=1/\ 1 + e^{-x}$ (sigmoid) daha çok kullanılır.

m-boyutlu giriş örüntüleri set edildiğinde { $i_p = (i_{p1}, i_{p2} \ldots . i_{pn})$; $p \in P$}'dir. Benzer şekilde istenilen n- boyutlu çıkış örüntüleri { $t_p = (t_{p1}, t_{p2} \ldots \ldots t_{pn})$ $p \in P$} belirtir. Burada; P:YSA uygulanan işaret, şekil vb gibi örüntülerini verir.

Bir görüntü için karesel hata (MSE) fonksiyonu E_p şu şekilde tanımlanır;

$$E_p = \frac{1}{2} \sum_{j \in \text{çıkış katmanı}} (t_{pj} - o_{pj})^2 \qquad (5.2)$$

Amaç uygun w_{ji} ve θ_j seçimiyle toplam hatayı yeterince küçük yapmaktır. Bu amacı gerçekleştirmek için, giriş örüntüsü ard arda ve rasgele biçimde seçilir. Daha sonra w_{ji} ve θ_j şöyle değiştirilir;

i_{pi} : Giriş işaretinin i bileşeni;

t_{pj} : Çıkış vektörünün j bileşeni;

o_{pj} : YSA uygulanan P örüntü setinin ürettiği çıkış ise;

$$\delta_{pj} = (t_{pj} - o_{pj}) \qquad (5.3)$$

$$\Delta_p w_{ji} = -\varepsilon \left(\frac{\partial E_p}{\partial w_{ji}} \right) \qquad (5.4)$$

$$\Delta_p \theta_j = -\varepsilon \left(\frac{\partial E_p}{\partial \theta_j} \right) \qquad (5.5)$$

Burada ε : öğrenme oranı adı verilen küçük bir pozitif sabit sayılır. Şayet gizli katman yok ise; (5.4) ve (5.5)'in sağ tarafı hesaplanır, o zaman;

$$\frac{\partial E_p}{\partial w_{ji}} = \frac{\partial E_p}{\partial o_{pj}} \frac{\partial o_{pj}}{\partial w_{ji}} \qquad (5.6)$$

$$\frac{\partial E_p}{\partial o_{pj}} = -(t_{pj} - o_{pj}) = -\delta_{pj} \qquad (5.7)$$

$$o_p = \sum_i w_{ji} . i_{pi} \qquad \text{ise;} \tag{5.8}$$

$$\frac{\partial o_{pj}}{\partial w_{ji}} = i_{pi} \tag{5.9}$$

elde edilir. (5.7) ve (5.9) ifadelerini (5.6)'da yerine koyarsak;

$$-\frac{\partial E_p}{\partial w_{ji}} = \delta_{pj} i_{pi} \tag{5.10}$$

olur. Öğrenmede en küçük minimuma ulaşılmak istenir. Bu durumda j. düğümün lineer olmayan çıkışı;

$$o_{pj} = f_j(netp_j) \quad \Rightarrow \quad netp_j = \sum_i w_{ji} o_{pj} \tag{5.11}$$

şeklindedir. Bu durumda;

$$\frac{\partial E_p}{\partial w_{ji}} = \frac{\partial E_p}{\partial netp_j} \frac{\partial netp_j}{\partial w_{ji}} \quad \Leftrightarrow \quad \frac{\partial netp_j}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \sum_k w_{jk} . o_{pk} = o_{pi} \tag{5.12}$$

$$\delta_{pi} = -\frac{\partial E_p}{\partial netp_j} = -\frac{\partial E_p}{\partial o_{pj}} \frac{\partial o_{pj}}{\partial netp_j} = -\frac{\partial E_p}{\partial o_{pj}} f_j'(netp_j) \tag{5.13}$$

Burada iki durum vardır:

1- o_{pj} YSA'nın çıkışı ise; (5.7) ifadesini (5.14)'de yerine koyarsak,

$$\delta_{pj} = \left(t_{pj} - o_{pj}\right) f_j'(netp_j) \tag{5.14}$$

bulunur.

2- Eğer gizli katmanların çıkış işaretinden bahsediliyorsa;

$$\sum_k \frac{\partial E_p}{\partial netp_k} \frac{\partial netp_k}{\partial p_j} \tag{5.15}$$

şeklinde ise,

$$\sum_k \frac{\partial E_p}{\partial netp_k} \frac{\partial}{\partial o_{pj}} \sum_i w_{ki}\, o_{pj} = -\sum_k \delta_{pk} w_{kj} \tag{5.16}$$

olur. Bulduğumuz son işlemi (5.13)'de yerine koyarsak;

$$\delta_{pj} = f'(net_{pj}) \sum_k \delta_{pk} w_{kj} \tag{5.17}$$

elde edilir. (5.16) denklemindeki (-) işareti, ağırlıkların ters yönde değiştiğini belirtir. Bütün yaptığımız işlemleri kısaca özetleyecek olursak;

1. Genelleştirilmiş Δ (delta) kuralı:

$$\Delta_p w_{ji} = \varepsilon \delta_{pj} i_{pi}$$

2. Çıkış katmanı elemanları için;

$$\delta_{pj} = (t_{pj} - o_{pj}) f_j'(netp_j)$$

3. Gizli katman elemanları için;

$$\delta_{pj} = f_j'(netp_j) \sum_k \delta_{pk} w_{kj}$$

olur. İşlem elemanında, transfer (eşik) fonksiyonu olarak "sigmoid" fonksiyonu kullanılarak türevi alınır ve gerekli kısaltmalar yapılırsa;

$$\frac{\partial o_{pj}}{\partial netp_j} = o_{pj}(1 - o_{pj}) \tag{5.18}$$

bulunur. Bunu (5.14) de yerine koyarsak, çıkış elemanı için;

$$\delta_{pj} = (t_{pj} - o_{pj}) o_{pj} (1 - o_{pj}) \tag{5.19}$$

elde edilir. (5.18)'u (5.17) de yerine koyarsak, gizli katman elemanı için;

$$\delta_{pj} = o_{pj} (1 - o_{pj}) \sum_{k} \delta_{pk} w_{kj} \tag{5.20}$$

bulunur. Yukarıda toplam içerisinde gösterilen k'nın, j çıkış birimine akan her birim k olduğuna dikkat edilmelidir. Hesaplamayı hızlandırmak için momentum terimleri (α) eklenirse, en genel halde çıkış ve gizli katman ifadeleri şu şekilde olur:

$$\Delta_p w_{ji}(n+1) = \varepsilon \delta_{pj} o_{pi} + \alpha \Delta_p w_{ji}(n) \tag{5.21}$$

$$\Delta_p \theta_j(n+1) = \varepsilon \delta_{pj} + \alpha \Delta_p \theta_j(n) \tag{5.22}$$

Burada; n: öğrenme saykıllarının sayısını gösterir. (α) küçük pozitif bir sayıdır.

5.13. 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.

6. UYGULAMA

Bu bölümde, verilerin hazırlanması,eğitimin yapılması, test aşamasının gerçekleştirilmesi ve bu işlemleri yapan programla ve ilgili açıklamalar verilecektir. Bu uygulamada kelime tanımanın temelini oluşturulacak şekilde bir yapı kurulmaya çalışıldı. Bu yüzden kelime tanımanın yapı taşı olan sesli seslerin tanınması asıl amaç olarak ele alındı. YSA'nın Türkçe'deki sesli seslerin tanınmasında kullanılması işlemi başlıca 4 adımda incelenebilir. Aşağıda anlatılan çalışma 486 DX-4 75 Mhz işlemcili 8 RAM'li bilgisayarla yapılmıştır.

1. Adımda: Sesin dosyaya kayıt edilmesi.

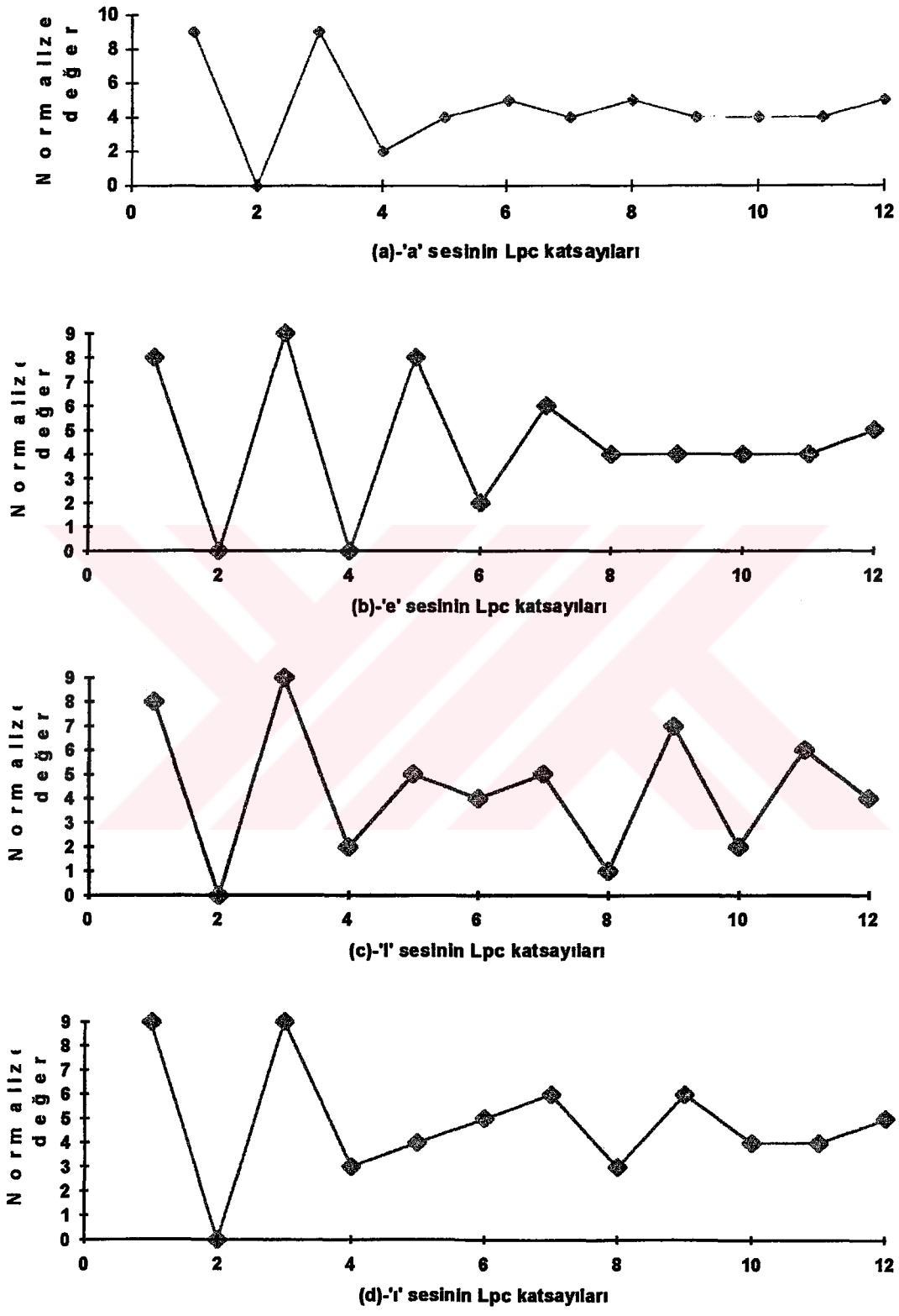
Bilgisayar aracılığıyla sesli sesler örneklenip dosyalandı. Bunun için Windows altında çalışan Soundrec programından yararlanıldı. 18-30 ms'lik harf örnekleri farklı isimlerde (örnek olarak den1_a,den1_e,.... gibi) depo edildi. Bu işlem 8 grup oluşturacak şekilde tekrarlandı. Bu örnekleme ve sesi dosyalama işlevini yürüten ses örnekleyici kart olarak 16 bitlik Sound Blaster Pro kartı kullanılmıştır. Bu işlem için kullanılan mikrofon orta kaliteli bir dinamik mikrofondur. Ortam gürültüsü işlem performansı ve sonuç başarısı üzerinde önemli etkiye sahiptir.

2. Adım: Sesin karakteristik değerlerinin bulunması.

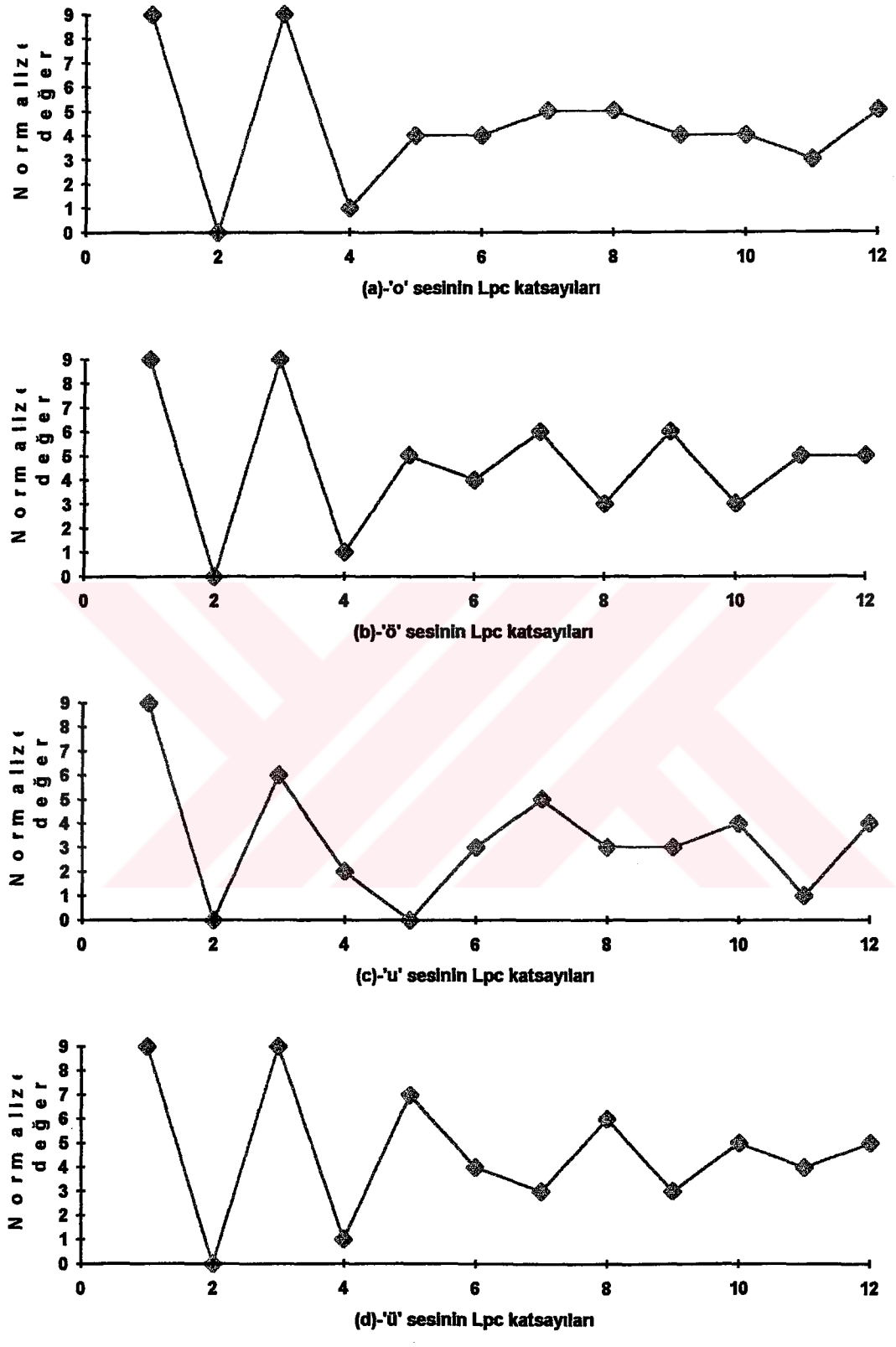
Dosyalanmış olan örneklenmiş ses bilgisinin YSA ile tanıma işlemine geçmek için bu sayısal ses bilgisinden hareketle sesin karakteristik değerlerinin bulmak için DFT ve LPC metotları ile çalışma yapıldı. Dosyalanmış ses verileri 128'şer çerçeveler halinde filtrelendi ve Hamming penceresinden geçirilerek ilgili pencerenin LPC katsayıları bulundu. Bu alınan çerçevelerden ardarda gelen iki çerçeve arasında %50'lik bir örtüşme sağlandı. LPC katsayılarının bulunmasında Levinson-Durbin algoritması kullanılmıştır. Her çerçeve için 12 LPC katsayısı elde edildi. Bu katsayılardan birer örnek Şekil 6.1 ve Şekil 6.2 'de gösterilmiştir. Her örnek sesin uzunluğuna bağlı olarak 30-40 kadar LPC katsayısı grubu elde edildi.

LPC katsayılarının hesaplandığı her çerçeve için ayrıca DFT değerleri hesaplandı. 128 veriden oluşan her çerçeve için DFT ile oluşturulan spektral kanallar 16 bileşene göre grublandı. Her grubun ortalaması alındı. Böylelikle LPC katsayılarına ilave olarak DFT'le hesaplanan 16 katsayıda tanıma işlemine katıldı. Bu katsayılardan birer örnek her harf için Şekil 6.3 ve Şekil 6.4 de gösterilmiştir.

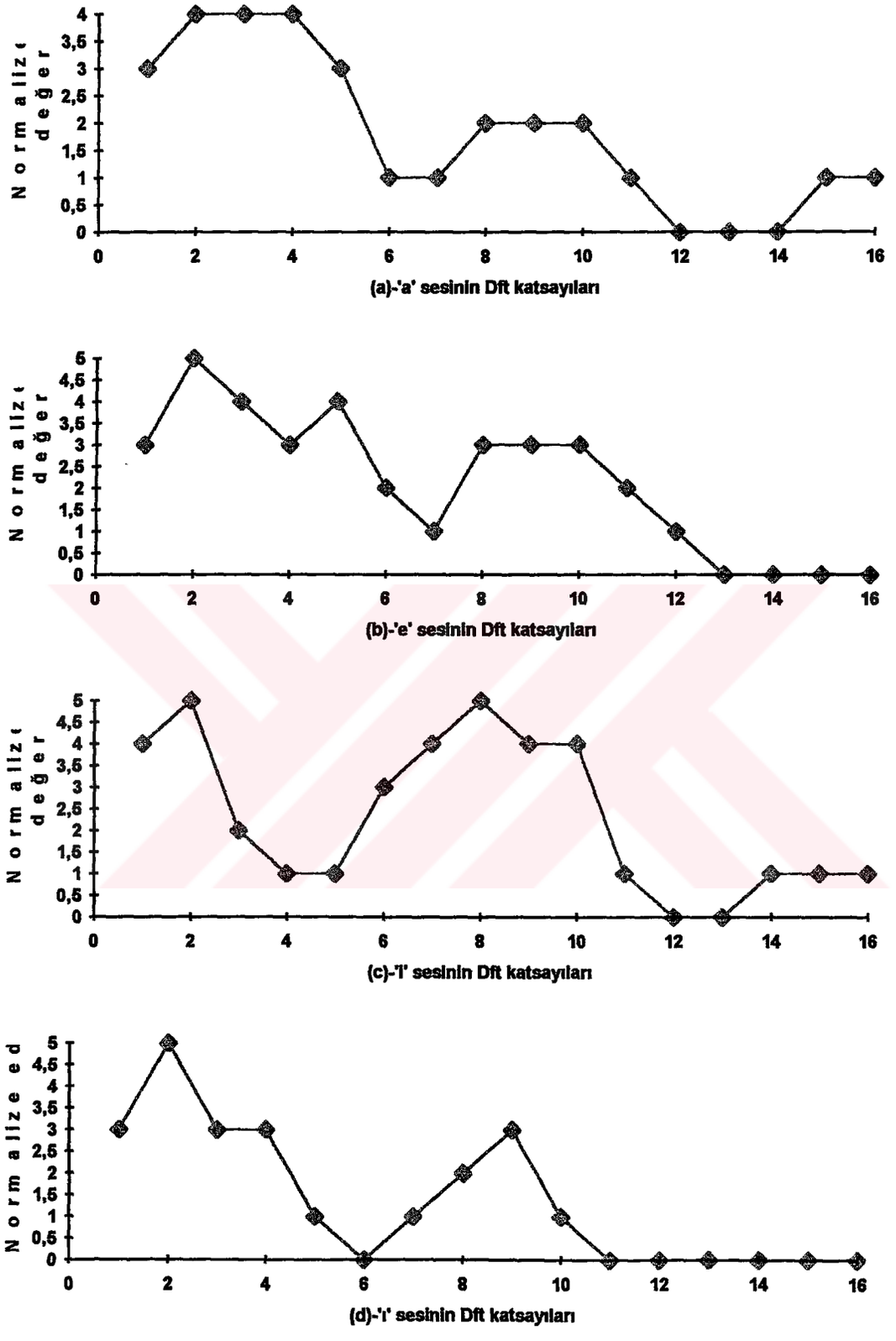
Her harf için 12 tane LPC ve 16 DFT olmak üzere 28 katsayıdan oluşan gruplar oluşturuldu. Elde edilen LPC ve DFT parametreleri YSA'nın eğitme işleminde kullanımına uygun olarak 0-1 arasında normalize edildi Bu verilerin hazırlanması çalışmasını yapan C programı Ek1'de verilmiştir. Bu program çalıştırıldığında ismi girilen ses dosyasından verileri okuyup yukarıda belirtilen katsayıları hesaplayıp ve aynı isimli farklı uzantıda bir dosyaya atmaktadır. Bu işlemde 8 harfli her grup için her harften 10 tane 28 parametreli bir bilgi alınıp depolandı.



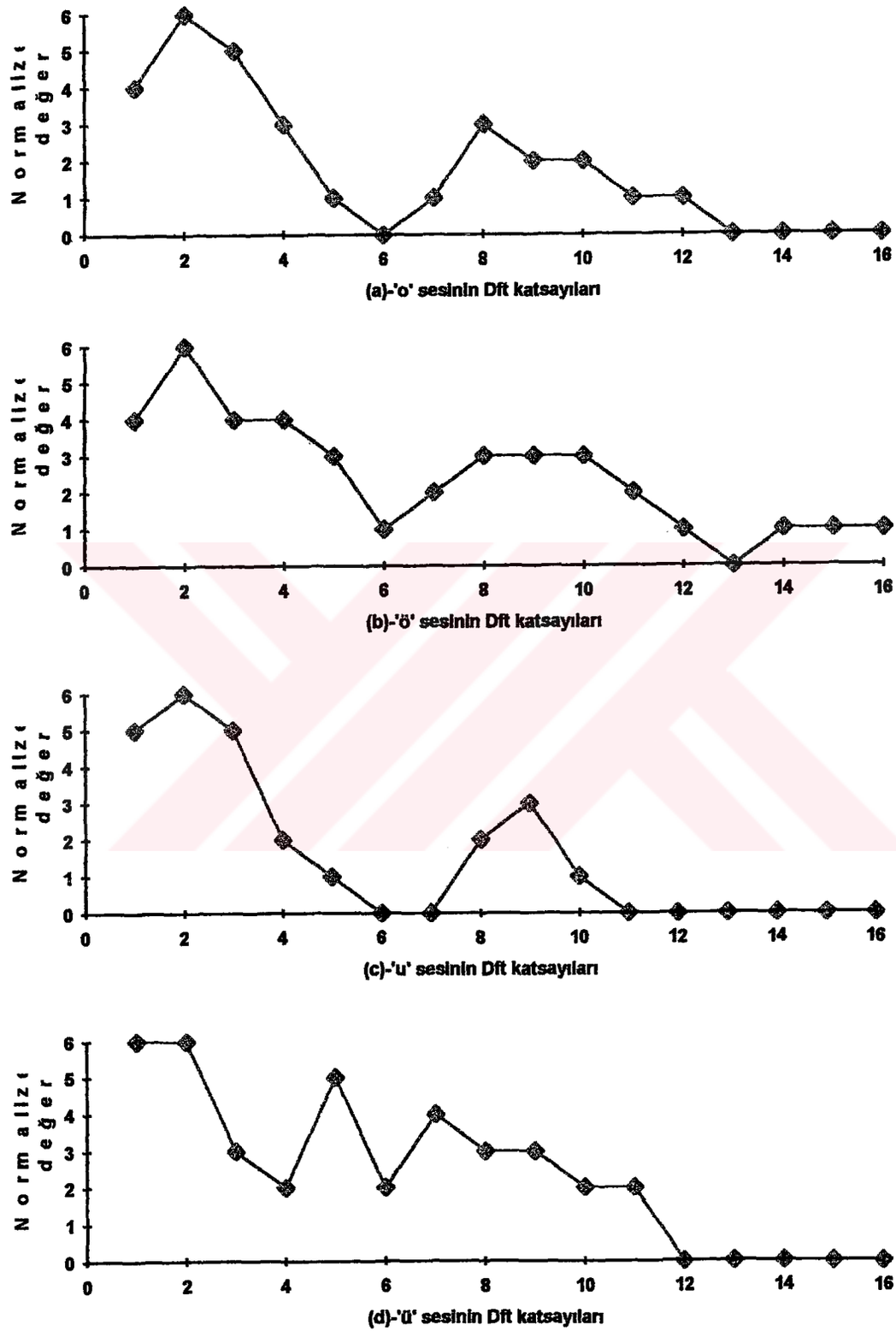
Şekil 5.1 Hamming pencereleme kullanılarak hesaplanmış LPC katsayıları



Şekil 5.2 Hamming pencereleme kullanılarak hesaplanmış LPC katsayıları



Şekil 5.3 Hamming pencereleme kullanılarak hesaplanan DFT'nin 16 kanallı süzgeç grubuna ayrılarak her kanalın ortalaması alınmak suretiyle hesaplanan katsayılar.



Şekil 5.4 Hamming pencereleme kullanılarak hesaplanan DFT'nin 16 kanallı süzgeç grubuna ayrılarak her kanalın ortalaması alınmak suretiyle hesaplanan katsayılar.

3. Adımda: Verilerin YSA'ya uygulanması ve Eğitilmesi.

Yukarıda anlatıldığı şekilde hazırlanan ses veri gruplarından bir kısmı eğitme amacıyla bir dosyada birleştirildi. Bu dosya herhangi bir anda 28 parametreden oluşan her ses verisi çerçevesi YSA'ya giriş olarak sunulmak üzere 480 tane giriş grubundan oluşmaktadır. Bu uygulamada her harfin eğitilmesi için ayrı bir YSA' mimarisi oluşturuldu. Giriş,gizli katman ve çıkıştan oluşan bu yapı 28-5-1 şeklinde belirlendi. Girişe uygulanan bu değerlere karşılık çıkış olarak ilgili harf için çıkış 1.0, diğer harfler için 0.0 olacak şekilde YSA'nın eğitim programına tanıtıldı. Yani örnek olarak 'a' harfi girişe uygulandığında çıkış 1.0 diğer harfler geldiğinde 0.0 olacak şekilde ağırlıkların ayarlanması programla sağlandı. Hata belirlenen toleransın altına düştüğünde her harf için eğitilmiş ağırlık değerleri farklı isimlerde (örnek olarak lf10a,lf10e,lf10i,....şeklinde) dosyaya kayıt edildi. Eğitme işleminin yapıldığı program Ek 2'de verilmiştir.

4. Adımda Test işlemi

Ek 3'de verilen program yardımıyla test işlemi için ayrılan ses bilgileri, eğitilmiş ağırlık değerleri kullanılarak tanınmaya çalışılmıştır. Burada her bir harf bilgisi 8 harfin eğitilmiş ağırlıkları ile belirlenen YSA'ya giriş olarak verilerek çıkışlara bakılmıştır. Hangi harfin eğitilmiş ağırlıklarında daha çok 1.0 civarında bir sonuç görülüyorsa test için kullanılan harf bu harf olarak tanınmıştır. Bu şekilde yapılan çalışmanın sonuncunda test için ayrılan sesli harflerin tanıma işleminin neticesi bir sonraki bölümde anlatılmıştır.

7. SONUÇLAR VE ÖNERİLER

Bu çalışmada LPC ve DFT analizi metodu ve YSA sınıflayıcısı ile sesli sesler tanınmaya çalışılmıştır. Konuşma seslerinin tanınması amacıyla program Borland C++ 3.0v ile yazılmıştır.

DFT analizi sonucu elde edilen spectral güçler 16 farklı kanalda gruplandı. Tüm bu kanalların ortalaması bulunup normalize edilmiş ve sesin temel özelliklerini karakterize eden vektör grupları bulunmuştur. Ayni şekilde her ses çerçevesi için 12 tane LPC katsayısı da bu vektör gruplarına eklenerek 28 tane katsayı içeren veri grupları oluşturuldu. Bu veri gruplarından bir kısmı YSA eğitimi için, bir kısmı ise ileride test amacıyla kullanılmak üzere ayrıldı. Sonuç olarak eğitim işlemi sonunda test işlemi yapılmıştır.

Tablo 7.1'de eğitim verilerini oluşturmak için sesin alındığı kişiden, farklı zamanlarda alınan beş tane ses grubunun tanıma sonucu gösterilmiştir.

Tablo 7.1

	A	E	I	İ	O	Ö	U	Ü	% Başarı
1	A	E	I	İ	O	Ö	U	Ü	100
2	A	E	I	İ	O	Ö	U	Ü	100
3	O	E	I	İ	O	Ö	U	Ü	88
4	A	E	I	İ	O	Ö	U	Ü	100
5	A	E	I	İ	O	Ö	U	Ü	100
% Başarı	80	100	100	100	100	100	100	100	

Tablo 7.2'de değişik kişilerden (eğitim verilerini oluşturmak için sesin alındığı kişinin haricindeki) alınan beş tane ses grubunun tanıma sonucu gösterilmiştir

Tablo 7.2

	A	E	I	İ	O	Ö	U	Ü	% Başarı
1	A	I	I	İ	O	Ü	Ü	Ü	88
2	A	I	I	İ	U	Ü	Ü	Ü	73
3	A	E	I	İ	O	Ö	U	Ü	100
4	A	Ü	Ö	İ	O	Ü	O	Ü	50
5	A	E	Ö	İ	O	E	Ü	Ü	73
% Başarı	100	40	60	100	80	20	20	100	

Yapılan çalışmanın doğruluk oranın, yukarıdaki tablolarda da görüldüğü gibi yapılan bir çok deneme sonucu yüksek olduğu görülmüştür. Konuşmacıya bağlı olarak çalışma yapılmış sonuç olarak tanımada % 100'e yakın bir başarı sağlanmıştır. Aynı sistemle farklı kişilerin sesleri tanınmaya çalışıldığında bu başarı düşmektedir. Bunun sebebi eğitme işleminin konuşmacıya bağlı olarak ses grupları alınarak yapılmış olmasıdır. Fakat bu çalışma, farklı kişilerden çok miktarda ses alınarak eğitme yapıldığında başarının çok yüksek olacağını göstermektedir. Buna ek olarak başarının arttırılmasında önemli bazı hususlar vardır. Bunlar maddeler halinde;

1. Ses verisinin örneklenmesi aşamasında:

Örneklemenin yapıldığı ortamın gürültüden mümkün oldukça yalıtılması tanımanın daha başarılı olmasında önemli rol oynadığı görülmüştür. Ayrıca mikrofonun kalitesi de başarıyı etkilemektedir.

2. Eğitim aşamasında:

Ses tanıma işlemi, sesi alınan kişiden farklı zamanlarda alınan seslerin sayısını artırarak büyük eğitim grupları oluşturulduğu durumda, daha başarılı olmaktadır. Doğal olarak büyük eğitim grubu, yapılan hesaplamaların sayısını arttırmakta ve eğitim işleminin uzamasına neden olmaktadır. Bu nedenle çalışılan bilgisayarın hızlı olması, daha büyük eğitim gruplarıyla çalışılmasına fırsat verdiğinden, çalışmaların başarılı olması için önemlidir.

Benzer şekilde konuşmacıdan bağımsız ses tanıma işlemi için de büyük eğitim grupları oluşturmak gerekmektedir.

Eğitim aşamasında belirlenen YSA mimarisinin uygun seçilmesi tanıma işleminin başarısını etkilemektedir. Uygun YSA mimarisinin seçimi, çalışılan sistemin karmaşıklığına göre gizli katman sayısını artırmak veya azaltmakla mümkün olmaktadır.

Tüm bu iyileştirmeler yapıldığı ve belirtilen hususlar dikkate alındığı takdirde ses tanıma için daha iyi bir yapı gerçekleştirilebileceği açıktır.

KAYNAKLAR

Higgins, R.J. Digital Signal Processing in VLSI, prentice Hall, 1990

Flanagan, J.L., C.H. Coker, L.R. Rabiner, R.W. Schafer, and N. Umeda, Synthetic Voices for Computers, IEEE Spectrum Vol.7, No.10, pp.22-45, october 1970

Geçkinli, N.C.,T. Gürgen and Etişkol, Speech Synthesis Using AM/FM Sinusoids and Band-Pass Noise, Signal processing 8, pp.339-361, Elsevier Science Publishers, 1985

Makhoul, J., Linear Prediction : A Tutorial Rewiew, IEEE, Vol.63 No.4 April 1975.

Ölçer, E. Türkçede Ayrık Konuşma Tanıma İTÜ Y. Lisans Tezi 4 Ekim 1993

Rabiner, L.R. and R.W. Schafer, Digital Processing of speech Signals, Prentice-Hall, 1978

Yıldırım, S. Ses Analizi ve Ses Sıkıştırma ve Kodlama Algoritmaları İTÜ Y. Lisans Tezi 6 Şubat 1995

Köseoğlu, M. Fatih Söz Sentezleme Teknikleri ve Söz Sentezleyici İçeren Devrelerin Tasarımı İTÜ Y. Lisans Tezi Haziran 1988.

Gücümoğlu, H. Vowel Recognition in Neural Networks İTÜ Y. Lisans Tezi 29 Haziran 1995

Karlık, B. Çok Fonksiyonlu Protezler İçin Yapay Sinir Ağları Kullanarak Miyoelektrik Kontrol Y.T.Ü. Doktora Tezi Mart 1994

EKLER

EK1

SESİN KRAKTERİSTİK KATSAYILARINI LPC VE DFT ANALİZİNİ KULLANARAK BULAN PROGRAM

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <conio.h>
#include <stdlib.h>
#include <process.h>
#include <alloc.h>
#include<ctype.h>
#ifndef VAX
#include<alloc.h>
#endif
#define ORNSAY   16200
#define PBOYU    128
#define CERSAY   300
#define ACERSAY  ORNSAY/PBOYU/(M*3)
/*********************************************************
main()
 {
 int ornek[ORNSAY],w[PBOYU],c,top;
 int long k;
 int Osayisi,i,h,j,m,n,cercevebas,Cersay;
 char testname[20],word1[20],word[20];
 float r[PBOYU],*ax,enbu,enku,f,y1[16];
 int p;
 FILE *fp1,*fp2;
 float sum,a[16][16],kx[16],e[16];
 p=12;
while(1){
 printf("Dosya ismi:");scanf("%s",testname);
 strcpy(word,testname);
 if((fp1=fopen(strcat(word,".lft"),"w"))==NULL)
 {printf("Acilamiyor");exit(1); }
 if((fp2=fopen(strcat(testname,".wav"),"rb"))==NULL)
 {printf("Acilamiyor");exit(1);
 }
 k=0;
 for(i=0;i<44;i++)fgetc(fp2);
 while (!feof(fp2))
```

```
  {
  ornek[k]=(fgetc(fp2)^0xff)-128;
  k++;
  }
  fclose(fp2);
  Osayisi=k;
  Cersay=(int)Osayisi/PBOYU*2;
  for(h=0;h<(Cersay);h++){
  cercevebas=PBOYU/2*h;
  w[0]=ornek[cercevebas];
//*********LPC katsayıları bulunuyor*****************************
 for(j=1;j<PBOYU;j++)w[j]=ornek[cercevebas+j]-.95*ornek[cercevebas+j-1];
   for(j=0;j<PBOYU;j++){w[j]=(float) ornek[cercevebas+j]*(.54-.46*
   cos((2*3.14*j)/(PBOYU-1)));}
   for(k=0;k<PBOYU;k++){
    r[k]=0;
    for(n=0;n<(PBOYU-k);n++)r[k]= r[k]+ w[n]*w[k+n];
   }
   for(k=1;k<PBOYU;k++)r[k]=r[k]/(r[0]+10e-5);
   r[0]=1;
   for(i=0;i<12;i++) for(j=0;j<12;j++)a[i][j]=0;
   e[0]=r[0];
   for(i=1;i<=p;i++){
    sum=0;
    for(j=1;j<i;j++)sum=sum+a[j][i-1]*r[i-j];
    if(e[0]==0)e[0]=.00001;
    kx[i]=(r[i]-sum)/e[i-1];
    a[i][i]=kx[i];
    for(j=1;j<i;j++)a[j][i]=a[j][i-1]-kx[i]*a[i-j][i-1];
    e[i]=(1-kx[i]*kx[i])*e[i-1];
   }
  enbu=0 ;
  enku=100;
  for(j=1;j<=p;j++)if(a[j][p]<enku)enku=a[j][p];
  for(j=1;j<=p;j++)if(a[j][p]>enbu)enbu=a[j][p];
  for(j=1;j<=p;j++)a[j][p]=(a[j][p]-enku)/
  (enbu+.000001+fabs(enku));
  for(j=1;j<=p;j++)fprintf(fp1,"%f ",a[j][p]);;
//**************DFT Analizi kullanılarak 16 katsayı bulunuyor ***********
  int j,k,n,y[PBOYU/2+4],f[PBOYU/2+4];
  float rsum,isum,ak,wr;
  int max1c,max2c,max3c;
  int max1,max2,max3,ort;
  wr=6.28/PBOYU;
  for(k=0;k<PBOYU/2+4;k++){
```

```
 rsum=isum=0;
 for(n=0;n<PBOYU;n++){
  ak=w[n];
  if(ak!=0)
  {
  rsum+=ak*cos(wr*n*k);
  isum+=ak*sin(wr*n*k);
  }
 }
 y[k]=(int) (20*log10(1+pow(rsum,2)+pow(isum,2)));
}
top=0;
for(j=0;j<14;j++){
 for(i=0;i<4;i++)top=top+y[i+j*4];
 y1[j]=top/4.;
 top=0;
}
top=0;
for(i=0;i<6;i++)top=top+y[i+j*4];
y1[14]=top/4.;
top=0;
for(i=0;i<6;i++)top=top+y[i+j*4];
y1[15]=top/4.;
enbu=0 ;
enku=100;
for(j=0;j<16;j++)if(y1[j]<enku)enku=y1[j];
for(j=0;j<16;j++)if(y1[j]>enbu)enbu=y1[j];
for(j=0;j<16;j++)y1[j]=(y1[j]-enku)/(enbu+.000001+fabs(enku));
for(j=0;j<16;j++){i=10*y1[j];fprintf(fp3,"%d ",i);}
for(j=0;j<16;j++) fprintf(fp1,"%f ",y1[j]);
fprintf(fp1,"\n");
 }
 fclose(fp1);
}
return(0);
}
```

EK2

EĞİTME ÇALIŞMALARININ YAPILDIĞI PROGRAM

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <conio.h>
#include <stdlib.h>
#include <process.h>
#include <alloc.h>
#include <ctype.h>
#ifndef VAX
#include <alloc.h>
#endif
#define NMXUNIT   10
#define NMXHLR   10
#define NMXOATTR  8
#define NMXINP  440
#define NMXIATTR 120
#define SEXIT     3
#define RESTRT   2
#define FEXIT   1
#define CONTNE  0
float  eta, alpha, err_curr, maxe, maxep;
float  huge *wtptr[NMXHLR+1];
float  huge *outptr[NMXHLR+2];
float  huge *errptr[NMXHLR+2];
float  huge *delw[NMXHLR+1];
float  huge target[NMXINP][NMXOATTR];
float  huge input[NMXINP][NMXIATTR],se5,se6;
float  huge ep[NMXINP];
float  huge outpt[NMXINP][NMXOATTR];
int nunit[NMXHLR+2],nhlayer,ninput,ninattr,noutattr,sd;
int result,i,m,errorcode;
long int cnt_num,cnt;
```

```
int nsnew,nsold;
char task_name[20],dr;
FILE *fp1,*fp2,*fp3,*fopen();
int fplot10;
char *err_file="veri3.dat";
long randseed=568731L;
char def[30],def1[20];
int mx,my;
void *p;
void git()
{ clrscr();printf("YAPILAN İTERASYON=%ld",cnt);
  printf("\n ŞU ANKİ HATA    =%f\n",err_curr);
  printf("ÇIKMAK İSTİYORMUSUNUZ ?'d(dur)-HERHANGİ BİR TUŞA BASINIZ");
  if(getch()=='d')cnt=cnt_num-1;
 }
//********Rastgele sayı üretilmesi**********************************
int random()
 {
 randseed=15625L*randseed+22221L;
 return((randseed>>16)&0x7FFF);
 }
//********Gizli katmanlar için dinamik hafizada  yer ayrılması***********
init()
 {
  int len1,len2,i,k;
  float *p1,*p2,*p3,*p4;
  len1=len2=0;
  nunit[nhlayer+2]=0;
  for(i=0;i<(nhlayer+2);i++){
  len1+=(nunit[i]+1)*nunit[i+1];
  len2+=nunit[i]+1;
  }
  p1=(float *) calloc(len1+1,sizeof(float));
  p2=(float *) calloc(len2+1,sizeof(float));
  p3=(float *) calloc(len2+1,sizeof(float));
  p4=(float *) calloc(len1+1,sizeof(float));
  wtptr[0]=p1;
  outptr[0]=p2;
  errptr[0]=p3;
  delw[0]=p4;
  for (i=1;i<(nhlayer+1);i++){
  wtptr[i]=wtptr[i-1]+nunit[i]*(nunit[i-1]+1);
  delw[i]=delw[i-1]+nunit[i]*(nunit[i-1]+1);
  }
  for (i=1;i<(nhlayer+2);i++){
```

```
 outptr[i]=outptr[i-1]+nunit[i-1]+1;
 errptr[i]=errptr[i-1]+nunit[i-1]+1;
 }
 for (i=0;i<nhlayer+1;i++){
 *(outptr[i]+nunit[i])=1.0;
 }
 return(0);
 }
//******* Ağırlık değerleri başlangıçta rasgele atanır.*****************
initwt()
 {
 int i,j ;
 for (j=0;j<nhlayer+1;j++)
  for (i=0;i<(nunit[j]+1)*nunit[j+1];i++){
    *(wtptr[j]+i)=random()/pow(2.0,15.0)-0.5;
    *(delw[j]+i)=0.0;
  }
  return(0);
 }
//********YSA mimarisinin belirlenmesi**********************
set_up()
 {
 int i;
 eta=0.9;
 // printf("\n Alfa momentum katsayısı (default=0.9)?: ");
 // scanf("%f",&eta);
 eta=0.9;
 alpha=0.7;
 // printf("\n Epsilon öğrenme oranı (default=0.7)?: ");
 //scanf("%f",&alpha);
 alpha=.7;
 maxe=0.000001;maxep=0.000001;
 printf("\nMaksimum toplam hata (default=0.00001)?: ");
 printf("\nMaksimum başlangıç hatası(default=0.00001)?: ");
 cnt_num=1000;
 printf("\nMaksimum iterasyon sayısı  (default=1000)?: ");
 //scanf("%ld",&cnt_num);
 cnt_num=100000;
 printf("\nGizli katman sayısı?: ");
 scanf("%d",&nhlayer);
 for(i=0;i<nhlayer;i++){
 printf("\n\t %d. gizli katmandaki düğüm sayısı?:",i+1);
 scanf("%d",&nunit[i+1]);
 }
 fplot10=1;
```

```
  clrscr();
  printf("\nBilgisayar hesap yapiyor.");
  nunit[nhlayer+1]=noutattr;
  nunit[0]=ninattr;
  return(0);
  }

//********Eğitim sonunda çıkışların dosyaya atanması*****************
dwrite(char *taskname)
    {
    int i,j,c;
    char var_file_name[20];
    strcpy(var_file_name,taskname);
    strcat(var_file_name,"_v.dat");
    if ((fp1=fopen(var_file_name,"w+"))==NULL)
    {
    perror("\n Veri dosyası açılamıyor");
    exit(0);
    }
    fprintf(fp1,"%u %u %u %f %f %u %lu\n",
    ninput,noutattr,ninattr,eta,alpha,nhlayer,cnt_num);
    for (i=0; i<nhlayer+2;i++){
      fprintf(fp1,"%d ",nunit[i]);
    }
    fprintf(fp1,"\n");
    for (i=0; i<ninput;i++){
         for (j=0; j<noutattr;j++)
           fprintf (fp1,"%f     ",outpt[i][j]);
           fprintf (fp1,"\n");
    }
    if ((c=fclose(fp2))!=0)
     printf("\nFile cannot be closed %d",c);
     return(0);
    }
//********Eğitilmiş ağırlıkların dosyaya yüklenmesi***************
wtwrite( char *taskname)
    {
    int i,j,c,k;
    char wt_file_name[20];
    strcpy(wt_file_name,taskname);
    strcat(wt_file_name,"_w.dat");
    if ((fp2=fopen(wt_file_name,"w+"))==NULL)
    {
    perror("\n Veri dosyası açılamıyor");
    exit(0);
```

```
    }
    k=0;
    for (i=0; i<nhlayer+1;i++)
        for (j=0; j<(nunit[i]+1)*nunit[i+1];j++){
          if(k==8){
            k=0;
            fprintf (fp2,"\n");
           }
           fprintf (fp2,"%f ",*(wtptr[i]+j));
           k++;
      }
    if ((c=fclose(fp2))!=0)
     printf("\n%d.Dosya kapatılamıyor ",c);
     return(0);
    }
//************Çıkışların Üretilmesi ***************************
void forward(i)
    {
    int m,n,p,offset;
    float net;
    for (m=0; m<ninattr;m++)
    *(outptr[0]+m)=input[i][m];
    for (m=1; m<nhlayer+2;m++){
        for (n=0; n<nunit[m];n++){
          net=0.0;
          for (p=0; p<nunit[m-1]+1;p++){
            offset=(nunit[m-1]+1)*n+p;
            net += *(wtptr[m-1]+offset)
            *(*(outptr[m-1]+p));
           }
           *(outptr[m]+n)= 1/(1+exp(-net));
         }
     }
    for(n=0; n<nunit[nhlayer+1];n++)
        outpt[i][n]=*(outptr[nhlayer+1]+n);
    }
//*********Hatanın işlemin sona erdirilmesi için kontrolu*************
int introspective (int nfrom,int nto)
    {
    int i,flag;
    if (cnt>=cnt_num) return(FEXIT);
    nsnew=0;
    flag=1;
    for (i=nfrom;(i<nto)&&(flag==1);i++){
    if (ep[i]<=maxep)nsnew++;
```

```
    else flag=0;
    }
    if (flag==1) return(SEXIT);
    if (err_curr<=maxe) return(SEXIT);
    return(CONTNE);
    }
//***********Ağırlıkların ayarlanması**********************
int rumelhart(int from_snum,int to_snum)
    {
    int i,j,k,m,n,p,offset,index;
    float out;
    char *err_file="veri3.dat";
    nsold=0;
    cnt=0;
    result=CONTNE;
    if ((fp3=fopen(err_file,"w"))==NULL)
     {
    perror("Veri dosyası açılamıyor.");
    exit(0);
    }
    do {
    err_curr=0.0;
    for (i=from_snum;i<to_snum;i++){
    forward(i);
    for(m=0;m<nunit[nhlayer+1];m++){
    out=*(outptr[nhlayer+1]+m);
    *(errptr[nhlayer+1]+m)=(target[i][m]-out)
                        *(1-out)*out;
    }
    for (m=nhlayer+1;m>=1;m--){
        for (n=0; n<nunit[m-1]+1;n++){
          *(errptr[m-1]+n)=0.0;
          for (p=0; p<nunit[m];p++){
           offset=(nunit[m-1]+1)*p+n;
           *(delw[m-1]+offset)=eta*(*(errptr[m]+p))
           *(*(outptr[m-1]+n))
           +alpha*(*(delw[m-1]+offset));
           *(errptr[m-1]+n)+=*(errptr[m]+p)
           *(*(wtptr[m-1]+offset));
          }
         *(errptr[m-1]+n)=*(errptr[m-1]+n)*
        (1-*(outptr[m-1]+n))*(*(outptr[m-1]+n));
    }
   }
   for (m=1;m<nhlayer+2;m++){
```

```
      for(n=0;n<nunit[m];n++){
          for(p=0;p<nunit[m-1]+1;p++){
            offset=(nunit[m-1]+1)*n+p;
            *(wtptr[m-1]+offset)+=*(delw[m-1]+offset);
          }
      }
    }
    ep[i]=0.0;
    for(m=0;m<nunit[nhlayer+1];m++){
     ep[i]+=fabs((target[i][m]-
     *(outptr[nhlayer+1]+m)));
    }
    err_curr+=ep[i]*ep[i];
    }
    err_curr=0.5*err_curr/ninput;
    if(kbhit())git();
    fprintf(fp3,"%ld%f\n",cnt,err_curr);
    cnt++;
    result=introspective(from_snum,to_snum);
    }
    while (result==CONTNE);
    for (i=from_snum; i<to_snum;i++) forward(i);
     for(i=0; i<nhlayer+1;i++){
         index=0;
         for (j=0;j<nunit[i+1];j++){
          printf("\n\n Weights between unit %d of layer %d",j,i+1);
          printf("and units of layer %d\n",i);
          for (k=0; k<nunit[i];k++)
          printf ("%f",*(wtptr[i]+index++));
          printf ("\n Threshold of unit %d of layer%dis%f",
          j,i+1,*(wtptr[i]+index++));
     }
    }
    for (i=0; i<ninput; i++)
     for (j=0;j<noutattr;j++)
     printf("\n\n sample %d output %d=%f target %d=%f",
     i,j,outpt[i][j],j,target[i][j]);getch();
     printf("\n\n Toplam iterasyon sayısı %ld",cnt);
     printf("\n Normalize edilmiş sistem hatası %f\n\n\n",err_curr);
     return(result);
    }
//*************Eğitim verilerinin YSA'ya tanıtılması*****************
user_session()
    {int i,j,showdata;
     char fnam[20],t1sk_name[20],dtype[20];
```

```
FILE *fp;
printf("\n Eğitme safhasının başı ");
printf("\n\t Veri dosyasının adını giriniz (taskname) : ");
//scanf("%s",task_name);
strcpy(task_name,"top4");
printf("\n Kaç tane giriş düğümü (veri sayısı)?:");
//scanf("%d",&ninattr);
ninattr=28*4;
printf("\nKaç tane çıkış düğümü?: ");
//scanf("%d",&noutattr);
noutattr=1;
printf("\n Toplam kaç tane grup?: ");
scanf("%d",&ninput);
strcpy(fnam,task_name);
strcat(fnam,".lft");
printf("\n Giriş dosyasının adı %s",fnam);
if ((fp=fopen(fnam,"r"))==NULL)
{
printf("\n  %s dosyası yok", fnam);
exit(0);
}
//for (m=0;m<12;m++) fscanf(fp," %f",&se5);
printf("\n Verilere bakmak istermisiniz.?");
printf("\n   Answer yes or no :");
scanf("%s",dtype);
showdata=((dtype[0]=='y')||(dtype[0]=='Y'));
for (i=0;i<ninput;i++){
    for (j=0;j<ninattr;j++){
    fscanf(fp," %f",&se5);
    input[i][j]=(se5)/1.1;
    if(showdata)printf("%f ",input[i][j]);
    }
}
fclose(fp);
printf("\n\t Çıkış dosyası adı giriniz : ");
//scanf("%s",task_name);
strcpy(task_name,"tops");;
strcpy(fnam,task_name);
strcat(fnam,".top");
if ((fp=fopen(fnam,"r"))==NULL)
{
printf("\n  %s dosyası yok", fnam);
exit(0);
}
for (i=0;i<ninput;i++){
```

```
            for (j=0;j<noutattr;j++){
               fscanf(fp," %f",&se6);
               target[i][j]=se6;
               if (showdata)printf(" %f",target[i][j]);
             }
           }
           if ((i=fclose(fp))!=0)
           {
            printf("\n  %d dosya kapatılamıyor.",i);
            exit(0);
           }
           return(0);
      }
//*****************************************************
 learning()
      {
      int result;
      user_session();
      set_up();
      init();
      do {
          initwt();
          result=rumelhart(0,ninput);
           } while (result==RESTRT);
      dwrite(task_name);
      wtwrite(task_name);
      return(0);
      }
//*****************************************************
main()
 {  learning();
    printf("\nEğitim işlemi bitti. ");
    return(0);
 }
```

EK3

TANIMA İŞLEMİNİ YAPAN PROGRAM

```
#include <conio.h>
#include <stdlib.h>
#include <process.h>
#include <alloc.h>
#include<ctype.h>
#ifndef VAX
#include<alloc.h>
#endif
#define NMXUNIT   10
#define NMXHLR   10
#define NMXOATTR  128
#define NMXINP  120
#define NMXIATTR  120
#define SEXIT     3
#define RESTRT   2
#define FEXIT   1
#define CONTNE  0
float  eta, alpha, err_curr, maxe, maxep,nw,g1[100];
float  far *wtptr[NMXHLR+1];
float far *outptr[NMXHLR+2];
float far *errptr[NMXHLR+2];
float far *delw[NMXHLR+1];
float far target[NMXINP][NMXOATTR];
float  far input[NMXINP][NMXIATTR];
float  far ep[NMXINP];
float  far outpt[NMXINP][NMXOATTR];
int nunit[NMXHLR+2],nhlayer,ninput,ninattr,noutattr;
int i,j,c,t=0,gi;
long int cnt_num;
float enbu;
char task_name[20],tase[20];
FILE *fp1,*fp2,*fp4;
char var_file_name[20];
int  bas=0 ;
//******İlgili  harf için YSA Yapısı hakkındaki verilerin dosyadan okunması*****
dread(char *taskname)
  {
   strcpy(var_file_name,taskname);
   strcat(var_file_name,"_v.dat");
```

```
    if (( fp1=fopen(var_file_name,"r"))==NULL)
    {
    perror("\n1 Veri dosyası açılamıyor ");
    exit(0);
    }
    fscanf(fp1,"%u%u%u%f%f%u%lu\n",&ninput,&noutattr,
    &ninattr,&eta,&alpha,&nhlayer,&cnt_num);
    for(i=0; i<nhlayer+2; i++)
     fscanf(fp1,"%d",&nunit[i]);
    if ((c=fclose(fp1))!=0)
    printf("\nFile cannot be closed %d ",c);
    return(0);
  }
//*******İlgili harf için eğitilmiş ağırlıkların dosyadan okunması**********
wtread(char *taskname)
   {
    int i,j,c ;
    char wt_file_name[20];
    strcpy(wt_file_name,taskname);
    strcat(wt_file_name,"_w.dat");
    if ((fp2=fopen(wt_file_name,"r"))==NULL)
    {
     perror("\n2 Veri dosyası açılamıyor");
     exit(0);
    }
    for (i=0; i<nhlayer+1;i++){
      for (j=0; j<(nunit[i]+1)*nunit[i+1];j++) {
            fscanf (fp2,"%f",(wtptr[i]+j));
      }
    }
    if ((c=fclose(fp2))!=0)
    printf("\n%d dosya kapatılamıyor",c);
    return(0);
  }
//********************Çıkışların üretilmesi**********************
void forward(i)
   {
     int m,n,p,offset;
     float net;
     for (m=0; m<ninattr;m++)
      *(outptr[0]+m)=input[i][m];
      for (m=1; m<nhlayer+2;m++){
           for (n=0; n<nunit[m];n++){
            net=0.0;
            for (p=0; p<nunit[m-1]+1;p++){
```

```
            offset=(nunit[m-1]+1)*n+p;
            net += *(wtptr[m-1]+offset)
            *(*(outptr[m-1]+p));
            }
            *(outptr[m]+n)= 1/(1+exp(-net));
           }
      }
      for(n=0; n<nunit[nhlayer+1];n++)
         outpt[i][n]=*(outptr[nhlayer+1]+n);
  }
//********İlgili harf için YSA çıkışlarının hesaplanması********
a(char *task_name,char *dfile)
  {
   int len1,len2,k,m;
   float *p1,*p2,*p3,*p4,o=0;
   strcpy(tase,task_name);
   strcat(tase,"_v.dat");
   if ((fp1=fopen(tase,"r"))==NULL)
   {
   perror("\n Veri dosyası açılamıyor");
   exit(0);
   }
   fscanf(fp1,"%u %u %u %f %f %u %lu\n",
   &ninput,&noutattr,&ninattr,&eta,&alpha,&nhlayer,&cnt_num);
   for (i=0; i<nhlayer+2;i++){
    fscanf(fp1,"%d ",&nunit[i]);
   }
   fclose(fp1);
   len1=len2=0;
   nunit[nhlayer+2]=0;
   for(i=0;i<(nhlayer+2);i++){
   len1+=(nunit[i]+1)*nunit[i+1];
   len2+=nunit[i]+1;
   }
   p1=(float *) calloc(len1+1,sizeof(float));
   p2=(float *) calloc(len2+1,sizeof(float));
   p3=(float *) calloc(len2+1,sizeof(float));
   p4=(float *) calloc(len1+1,sizeof(float));
   wtptr[0]=p1;
   outptr[0]=p2;
   errptr[0]=p3;
   delw[0]=p4;
   for (i=1;i<(nhlayer+1);i++){
    wtptr[i]=wtptr[i-1]+nunit[i]*(nunit[i-1]+1);
    delw[i]=delw[i-1]+nunit[i]*(nunit[i-1]+1);
```

```
    }
    for (i=1;i<(nhlayer+2);i++){
     outptr[i]=outptr[i-1]+nunit[i-1]+1;
     errptr[i]=errptr[i-1]+nunit[i-1]+1;
    }
    for (i=0;i<nhlayer+1;i++){
    *(outptr[i]+nunit[i])=1.0;
    }
    dread(task_name);
    wtread(task_name);
    if (( fp4=fopen(dfile,"r"))==NULL)
    {
    perror("\nt Veri dosyası açılamıyor ");
    exit(0);
    }
    for (gi=0; gi<bas*ninattr;gi++)fscanf(fp4,"%f",&nw);
     for (gi=0; gi<(16*28/ninattr);gi++){
      for (m=0; m<ninattr;m++){fscanf(fp4,"%f",&nw);
       input[i][m]=(nw)/1.1;}
       forward(i);
       for(m=0;m<noutattr;m++){
           o=o+*(outptr[nhlayer+1]+m);
       }
     }
     g1[t]=o*ninattr/28;
     t++;
     fclose(fp4);fclose(fp2);fclose(fp1);
     free(p1);
     free(p2);
     free(p3);
     free(p4);
     return(0);
  }
//*******************************************************
main()
  {
   while(1){
   char dfile[20];
   printf("Dosya ismi:");scanf("%s",dfile);
   strcat(dfile,".lft");
   t=0;
   strcpy(task_name,"lf10a");
   a(task_name,dfile);
   strcpy(task_name,"lf10e");
   a(task_name,dfile);
```

```
 strcpy(task_name,"lf10i");
 a(task_name,dfile);
 strcpy(task_name,"lf10y");
 a(task_name,dfile);
 strcpy(task_name,"lf10u");
 a(task_name,dfile);
 strcpy(task_name,"lf10uu");
 a(task_name,dfile);
 strcpy(task_name,"lf10o");
 a(task_name,dfile);
 strcpy(task_name,"lf10oo");
 a(task_name,dfile);
 enbu=g1[0];
 for(i=1;i<8;i++)if(g1[i]>enbu)enbu=g1[i];
 if(g1[0]==enbu)printf("BU HARF A\n ");
 if(g1[1]==enbu)printf("BU HARF E\n ");
 if(g1[2]==enbu)printf("BU HARF İ\n ");
 if(g1[3]==enbu)printf("BU HARF I\n");
 if(g1[4]==enbu)printf("BU HARF U\n ");
 if(g1[5]==enbu)printf("BU HARF Ü\n");
 if(g1[6]==enbu)printf("BU HARF O\n ");
 if(g1[7]==enbu)printf("BU HARF Ö\n");
 }
 return(0);
}
```

ÖZGEÇMİŞ

Seydi Vakkas ÜSTÜN	Elektrik-Elektronik Mühendisliği (1992 Anadolu Üniversitesi)
Doğum Tarihi ve Yeri	01.01.1970, Adıyaman/Besni
1988-1992	Anadolu Üniversitesi Müh.-Mimarlık Fak. Elektrik-Elektronik Müh.
1981-1987	Besni Lisesi
19876-1981	Besni Dumlupınar İlkokulu
GÖREVİ	
1995	Araştırma Görevlisi Celal Bayar Üniversitesi Mühendislik Fakültesi Elektrik-Elektronik Müh. Bölümü
Yabancı Dil	İngilizce