T.C.
ULUDAĞ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ARAMA MOTORLARI KULLANICI OTURUMLARINDAKİ KONU DEĞİŞİKLİKLERİNİN TESPİT VE TAHMİN YÖNTEMLERİ

Fatih ÇAVDUR

YÜKSEK LİSANS TEZİ
ENDÜSTRİ MÜHENDİSLİĞİ ANABİLİM DALI

BURSA 2005

T.C.
ULUDAĞ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ARAMA MOTORLARI KULLANICI OTURUMLARINDAKİ KONU DEĞİŞİKLİKLERİNİN TESPİT VE TAHMİN YÖNTEMLERİ

Fatih ÇAVDUR

YÜKSEK LİSANS TEZİ
ENDÜSTRİ MÜHENDİSLİĞİ ANABİLİM DALI

BURSA 2005

T.C.
ULUDAĞ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ARAMA MOTORLARI KULLANICI OTURUMLARINDAKİ KONU DEĞİŞİKLİKLERİNİN TESPİT VE TAHMİN YÖNTEMLERİ

Fatih ÇAVDUR

YÜKSEK LİSANS TEZİ
ENDÜSTRİ MÜHENDİSLİĞİ ANABİLİM DALI

Bu tez …….. tarihinde aşağıdaki jüri tarafından oybirliği/oy çokluğu ile kabul edilmiştir.

Yrd. Doç. Dr. H. Cenk Özmutlu …….. ……..
(Danışman)

ÖZET

İnternet'in çok hızlı büyümesiyle birlikte, bu geniş veritabanında, İnternet kullanıcılarının aradıkları bilgilere ulaşım sorunu ortaya çıkmış ve bu sorunu ortadan kaldırmak amacıyla, arama motoru denilen sistemler geliştirilmiştir. İnternet kullanıcıları, bir oturum boyunca, arama motorlarını kullanarak, birden çok sayıda konuyla ilgili aramalar yapabilmektedirler. Bu durumda, bir oturum boyunca yapılan aramalar arasında, arama yapılan konuda bir veya daha fazla sayıda değişiklikler olmaktadır. İnsan yardımı olmaksızın, bu değişikliklerin tespit edilebilmesini sağlayan sistemlerin geliştirilmesi, kullanıcıların davranışlarını analiz etmek açısından önemli bir potansiyele sahiptir. Bu nedenle, bir oturumdaki konu değişikliklerinin belirlenmesi önemlidir. Bu çalışma, arama motorları kullanıcı oturumlarındaki konu değişikliklerinin tespit ve tahminiyle ilgilidir.

Bu çalışmada, arama motorları kullanıcı oturumlarındaki konu değişikliklerinin belirlenmesi için temel olarak iki farklı yaklaşım uygulanmıştır: (i) Dempster-Shafer teorisi ve genetik algoritmalar yaklaşımı ve (ii) yapay sinir ağları yaklaşımı.

Yaklaşımlar Excite ve Fast arama motorlarının veri kümelerinden alınan örnekler üzerinde uygulanmıştır. Uygulanan yaklaşımlar, örneklerdeki her kayıt için "konu değişikliği yok" ve "konu değişikliği var" şeklinde atamalar yapmaktadırlar. Bu atamaları yapmak için iki farklı veri kullanılmaktadır. Bunlar, ardışık iki arama arasındaki süreyi gösteren zaman aralığı (time interval – ti) ve ardışık iki aramada girilmiş olan sorgular arasındaki yapısal ilişkileri gösteren arama yapısı (search pattern – sp) verileridir. Yaklaşımlar tarafından yapılan atamalar, insanlar tarafından yapılan atamalarla karşılaştırılmış ve yaklaşımların performans ölçüleri hesaplanmış ve incelenmiştir.

Anahtar Kelimeler

Dempster-Shafer Teorisi, Genetik Algoritmalar, Yapay Sinir Ağları, Arama Motorları, Kullanıcı Davranışlarının İncelenmesi, Konu Değişikliklerinin Belirlenmesi

ABSTRACT

With the rapid expansion of the Internet, for Internet users the problem of getting the necessary information in this large database, occurred and to solve this problem, systems called search engines were developed. Internet users can do searches about more than one topic by using search engines in a session. So, one or more topic changes can be occurred among the searches in the session. Developing systems which make possible to identify these changes without human help has an important potential for analyzing users' behaviors. So, it is important to identify the topic changes in a session. This study is about identifying and predicting topic changes in search engines user sessions.

In this study, basically, two different approaches are applied for identifying topic changes in search engines user sessions: (i) Dempster-Shafer theory and genetic algorithms approach and (ii) neural networks approach.

The approaches are applied on the samples selected from Excite and Fast search engines' datasets. Using these approaches "topic continuation" or "topic shift" assignments are made for each record in the sample. Two different parameters are used for making these assignments. These are time interval (ti) parameter which indicates the time between two consecutive searches and search pattern (sp) parameter which indicates structural relations between two consecutive searches. The assignments which are made by proposed approaches are compared with the assignments which are made by humans, and performance measures are calculated and analyzed.

Keywords

Dempster-Shafer Theory, Genetic Algorithms, Neural Networks, Search Engines, User Behavior Analysis, Identifying Topic Changes

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

1 – GİRİŞ

İnternet'in çok hızlı büyümesiyle birlikte, bu çok büyük bilgi kaynağı içinde aranılan bilgilerin bulunması amacıyla arama motorlarının kullanımı giderek artmaktadır. Bu artış, giderek daha iyi çalışan arama motorlarının geliştirilmesini zorunlu hale getirmektedir. Bu gelişim sürecinin bir aşaması da bir oturumdaki ardışık arama işlemlerinde gerçekleşen konu değişikliklerinin tespit ve tahmin edilmesidir. Bu çalışma, arama motorları kullanıcı oturumlarındaki konu değişikliklerinin tespit ve tahminiyle ilgilidir.

İnternet kullanıcıları arama motorlarını kullanarak aynı konuda veya farklı konularda çok sayıda ardışık arama işlemleri yapabilmektedirler. Bu ardışık arama işlemlerinde konu değişikliklerinin olup olmadığının belirlenmesi, arama motorlarının performanslarının arttırılması amacıyla kullanılabileceğinden, insan yardımı olmaksızın bunları tespit edebilen sistemlerin geliştirilmesi önem kazanmaktadır. Bu çalışmada, arama motorları kullanıcı oturumlarındaki konu değişikliklerini tespit etmek amacıyla temel olarak iki farklı yaklaşım uygulanmıştır. Uygulanan bu yaklaşımlar, Excite (http://www.excite.com) ve Fast (http://www.fast.com) arama motorlarının veri kümelerinden alınan örnekler üzerinde denenmiştir. Bu örneklerde kullanılan veriler, İnternet Protokolü (Internet Protocol – IP) adresi, arama zamanı ve sorgudan oluşmaktadır. Veriler üzerinde bazı işlemler uygulanarak, bu çalışmada kullanılabilecek şekle getirilmiştir. Bu amaçla, aynı IP adresine ait arama zamanları arasındaki farklar alınarak zaman aralığı (time interval – ti) elde edilmiştir. Benzer şekilde, aynı IP adresine ait ardışık aramalarda girilmiş olan sorgular incelenerek bunlar arasındaki yapısal ilişkileri gösteren arama yapısı (search pattern – sp) bulunmuştur. Her iki arama motorunun veri kümelerinden alınan örnekler de yaklaşık 10.000 kayıttan oluşmaktadır. Bu örnekler, yaklaşık 5.000 kayıt içerecek şekilde iki kısma ayrılmış ve her iki örneğin ilk kısmı eğitim (veya yöntemlerin uygulanabilir hale gelmeleri için bazı değerlerin belirlenmesi) ve ikinci kısmı da test amacıyla kullanılmıştır.

Yukarıda da belirtildiği gibi, bu çalışmada, zaman aralığı ve arama yapısı verilerini kullanarak, arama motorları kullanıcı oturumlarındaki konu değişikliklerini tespit etmek amacıyla temel olarak, iki farklı yaklaşım uygulanmıştır: (i) Dempster-Shafer teorisi ve genetik algoritmalar yaklaşımı ve (ii) yapay sinir ağları yaklaşımı. Bu

yaklaşımların başarılarını, uygulanan yöntemlerin, “konu değişikliği yok” ve “konu değişikliği var” olan durumları en yüksek doğrulukla belirleyebilmeleri oluşturmaktadır. Diğer bir deyişle, bir yaklaşımın başarısının diğerinden daha yüksek olabilmesi için bu yaklaşımın “konu değişikliği yok” ve “konu değişikliği var” olan durumları doğru olarak tespit etme sayısının diğerinden daha fazla olması gerekmektedir.

İlk yaklaşımda, Excite eğitim örneğindeki her kayıt için “konu değişikliği var” olasılıkları hesaplanmıştır. Bu amaçla, öncelikle, her kayıt için belirli bir zaman aralığı ve belirli bir arama yapısı değerine bağlı olarak iki farklı “konu değişikliği var” olasılığı hesaplanmıştır. Bu durumda, iki farklı kaynaktan (zaman aralığı ve arama yapısı) elde edilen olasılıklar (bilgiler veya deliller) bulunmakta ve bunların birleştirilmesi gerekmektedir ve bunun için Dempster-Shafer teorisi kullanılmıştır. Dempster-Shafer teorisi kullanılarak bu iki olasılık birleştirilmiş, her kayıt için bu iki farklı kaynağı de dikkate alan bir “konu değişikliği var” olasılığı hesaplanmıştır. Daha sonra, elde edilen bu olasılığa dayanarak “konu değişikliği yok” veya “konu değişikliği var” şeklinde karar verilmektedir. Yaklaşımda kullanılan performans ölçülerinin değerleri, yaklaşımın başarısına, yani doğru olarak belirlenen “konu değişikliği yok” ve “konu değişikliği var” sayılarına; bu da yaklaşımda kullanılan parametrelere bağlıdır. Bu parametrelerin aldığı değerler, yaklaşımın başarısında belirleyici olan unsurlardan biridir. Bu yüzden, parametreler, yaklaşımın en iyi performansı göstermesini sağlayacak şekilde belirlenmelidir. Öte yandan, bu işlemin analitik yöntemlerle yapılması mümkün olmadığından, en iyi parametre değerlerinin belirlenebilmesi için bir genetik algoritma kullanılmıştır.

Genetik algoritma kullanılarak en iyi parametre değerleri belirlendikten sonra, bu parametreler kullanılarak, yaklaşım Excite test örneğine uygulanmış ve performans ölçüleri hesaplanmıştır. Bununla birlikte, yaklaşımda sadece Excite eğitim örneğinden hesaplanan parametre değerleri kullanılmamış, aynı zamanda, benzer bir çalışma yapmış olan He ve arkadaşlarının (2002) çalışmasında verilmiş olan ve rassal olarak belirlenmiş olan parametre değerleri de Excite test örneğine uygulanmıştır. Ayrıca, bu değerlerle ilgili çeşitli istatistiksel analizler de yapılmıştır.

İkinci yaklaşımda ise konu değişikliklerini belirlemek için bir yapay sinir ağı kullanılmıştır. Kullanılan yapay sinir ağının iki girişi, “zaman aralığı” ve arama yapısı

olarak alınmıştır. Yapay sinir ağının tek çıkışı ise “konu değişikliği yok” veya “konu değişikliği var” durumlarını belirlemek amacıyla kullanılan bir sayıdır. Bu sayı, ilk yaklaşımda olduğu gibi, bir eşik değeriyle karşılaştırılarak “konu değişikliği yok” veya “konu değişikliği var” şeklinde sonuç veren bir ikili değişkene dönüştürülmektedir. Öncelikle, yapay sinir ağı, Excite eğitim örneği kullanılarak eğitilmiştir. Bu amaçla, yapay sinir ağına, zaman aralığı ve arama yapısı girişleri ve bunlara karşılık gelen ve insanlar tarafından belirlenmiş olan “konu değişikliği yok” veya “konu değişikliği var” şeklindeki çıkışlar verilmiştir. Yapay sinir ağının eğitimi tamamlandıktan sonra, Excite test örneği üzerinde çalıştırılmıştır. Bu şekilde, çok sayıda eğitim ve test işlemleri yapılarak, elde edilen performans ölçülerinin değerleri incelenmiştir. Dikkate alınan performans ölçüleri önceki yaklaşımdakilerle aynıdır.

Kullanılan yapay sinir ağının göstermiş olduğu performansın, eğitim için kullanılan örneğe bağlı olup olmadığının belirlenebilmesi amacıyla, yapay sinir ağının eğitim ve testi için farklı veri kümelerine ait örnekler seçilerek, yukarıda anlatılan işlemler tekrarlanmıştır. Bu amaçla, Excite örneğine ek olarak Fast örneği kullanılmıştır. Yapay sinir ağı, Excite eğitim örneği ile eğitilip, sadece Excite test örneği ile test edilmek yerine, Fast test örneği ile de test edilmiştir. Bunun tersi de yapılarak, Fast eğitim örneği ile eğitilen yapay sinir ağının testi için de hem Fast, hem de Excite test örnekleri kullanılmıştır. Eğitim ve test için farklı veri kümelerine ait örnekler kullanıldığında elde edilen sonuçlar, eğitim ve test örneklerinin aynı veri kümesinden alındığı durumda elde edilen sonuçlarla karşılaştırılmıştır.

2 – KONU İLE İLGİLİ ÇALIŞMALAR

2.1 – Kuramsal Bilgiler

İnternet, bir bilgisayarın, birbiriyle bağlantılı milyonlarca bilgisayardan herhangi birisiyle veri alışverişi yapabilmesine olanak sağlayan bilgisayarlar sistemidir. Son on yıl içinde kullanım alanı çok genişlemiş olmakla birlikte, farklı şekillerde de olsa, 1960'lardan beri mevcuttur.

İnternet'in temeli sayılabilecek olan ARPANET, ABD Savunma Bakanlığı İleri Araştırma Projeleri Ajansı (Defense Advanced Research Projects Agency – DARPA) tarafından, üniversiteler ve araştırma kuruluşları işbirliğiyle tasarlanmıştır. Başlangıçta ARPANET, iletişim teknolojilerini araştırma ve geliştirme amacıyla, dünyanın çeşitli yerlerindeki bilim adamlarının birbirleriyle bağlantı kurarak, bilgilerini paylaşmaları için kullanılmaktaydı.

1970 ve 1980'ler boyunca ARPANET, çoğu askeri amaçlara hizmet eden bir dizi başka ağları ortaya çıkaracak şekilde gelişmiştir. Bu evrim sürecinin en önemli aşaması 1989 yılında gerçekleşmiştir. Askeri amaçlı bu ağlar, yerlerini Ulusal Bilim Vakfı'nın (National Science Foundation – NSF) NSFNET adlı ağına bırakmıştır. Bu olay, İnternet'in artık sadece askeri amaçlara hizmet etmekten çıkarak, diğer insanlar tarafından da kullanılabilir hale gelmiş olması dolayısıyla, İnternet tarihinde çok önemli bir değişimin göstergesidir. İnternet, 1990 yılından itibaren, önemli oranda büyümüştür.

İnternet'in yapısını açıklayabilmek için öncelikle bilgisayar ağlarından söz etmek gerekmektedir. Bilgisayar ağları çeşitli biçimlerde olabilmelerine karşın, temel olarak iki tip ağ vardır: (i) yerel alan ağı (Local Area Network – LAN) ve (ii) geniş alan ağı (Wide Area Network – WAN). Birinci tip yerel alan ağları, genellikle birbirlerine yakın mesafedeki bilgisayarlardan oluşur. İkinci tip ağ olan geniş alan ağları ise genellikle birbirlerine uzak mesafedeki bilgisayarlardan oluşur. Aslında bir ağın yerel alan ağı mı, yoksa geniş alan ağı mı olduğunu aralarındaki uzaklık değil, bağlantı için kullanılan yöntem belirlemektedir.

İnternet, bir bilgisayar ağı, ağlardan oluşan bir ağdır. Bir dizi ağ, bir araya gelerek daha geniş bir ağ oluşturmaktadırlar. Bir bilgisayarı İnternet'e bağlamak demek, aslında bir ağ omurgası (network backbone) üzerinden başka ağlara bağlı olan bir ağa bağlanmak demektir. Ağ omurgası, başka ağlarla bağlantı halinde olan geniş bir ağdır.

Ağ omurgaları arasında, iki ayrı ağ üzerinde bulunan, iki ayrı bilgisayarın birbirleriyle mesaj ve veri alış verişi yapabilmesini sağlayan ve ağ geçidi (gateway) adında ara bağlantılar vardır. ABD ve Kanada'daki çoğu ağ omurgası, AT&T veya MCI gibi özel telekomünikasyon şirketleri ve Netcom veya Uunet gibi ağ servis şirketleri tarafından işletilmektedir. Diğer ülkelerdeki ağ omurgaları, telekomünikasyon kurumları tarafından yönetilmektedir.

Günümüzde, İnternet denildiğinde, genelde, İnternet'in sadece belirli bir parçası olan World Wide Web – WWW veya kısaca Web anlaşılmaktadır. Ancak en hızlı gelişen ve en çok kullanılanı olsa da İnternet'in tek bileşeni Web değildir. Bu bileşenler kısaca şu şekildedir:

- Web, İnternet kaynaklarını, İnternet üzerindeki bilgilere kolayca erişilebilecek şekilde düzenlemektedir.
- Elektronik posta (e-mail), İnternet kullanıcıları arasında mesaj ve dosya alışverişine olanak sağlamaktadır.
- Telnet, bir bilgisayarın başka bir bilgisayara bağlanmasına ve o bilgisayardaki bilgileri kullanmasına olanak sağlayan bir yazılımdır.
- Dosya Transfer Protokolü (File Transfer Protocol – FTP), bir bilgisayardan diğerine dosya aktarımı için kullanılan İnternet yazılımıdır. Ağ üzerindeki bir bilgisayardan, kullanılan bilgisayara dosya aktarma işlemine indirme (download), kullanılan bilgisayardan, ağ üzerindeki bir bilgisayara dosya aktarma işlemine de yükleme (upload) denir.
- Gopher, geniş bir bilgi alanını kapsayan, erişim ve elde etme sistemidir.
- Sohbet grupları (chat groups), ortak ilgi alanları olan kullanıcıların eşzamanlı olarak karşılıklı sohbet ettikleri forumlardır.
- Haber grupları (news groups), ortak ilgi alanları olan üyelere yönelik tartışma ve yazışma ortamlarıdır.

Yukarıda da belirtildiği gibi, İnternet'in en çok ilgi gören yönü Web'dir. Web'de uğrayabileceğiniz alanlara Web siteleri denir. Örneğin, "http://www.uludag.edu.tr"

Uludağ Üniversitesi'nin Web sitesinin adresidir. Web sitelerini görmek için kullanılan programlara da Web tarayıcı (Web browser) adı verilmektedir.

Web, bilgiye erişim sağlayan kaynakları düzenler. Bunu yaparken de Web sayfaları (Web pages) veya Web belgeleri (Web documents) denilen dosyaları kullanır. Bir Web sitesi ise içinde bir veya daha fazla sayıda Web sayfası bulundurabilen bir Web alanıdır. Her Web sayfasının kendine has bir evrensel kaynak konumlandırıcısı (Universal Resource Locator – URL) vardır. Örneğin, "http://www.uludag.edu.tr" Uludağ Üniversitesi Web sitesinin URL'sidir. Bir URL'nin üç bileşeni vardır. İlk bileşen, "//" ifadesinin solundaki kısımdır ve HTTP, Gopher, Telnet veya FTP gibi erişim için kullanılan İnternet aracı veya yöntemini göstermektedir. HTTP (HyperText Transfer Protocol) en çok kullanılan İnternet erişim yöntemi olup, İnternet'in Web'de kullandığı dildir. HTTP kullanan iki tür bilgisayar vardır: (i) Web istemcileri (Web clients) ve (ii) Web sunucuları (Web servers). Birbirleriyle iletişime geçecek olan tüm Web istemcileri ve Web sunucuları HTTP kullanabilmelidirler. URL'lerdeki "//" ifadesi, erişim için kullanılan araç veya yöntemin adını, URL'nin geri kalan kısmından ayırmak için kullanılmaktadır.

URL'nin ikinci kısmı, ana bilgisayar (host computer) ve alan (domain) tanımlarından oluşmaktadır. Ana bilgisayar, Web sitesinin yerleştirildiği bilgisayardır. Alan, Web sitesinin coğrafi konumunu veya sponsor kuruluşu ifade eder. Alan adı, ana bilgisayara erişim hiyerarşisini yansıtır. Alan adı içinde soldan sağa doğru ilerledikçe, bu hiyerarşinin üst düzeylerine doğru ilerlenmiş olur. Yukarıdaki örnekte, "uludag" bir eğitim kurumudur (Uludağ Üniversitesi) ve "edu" da İnternet'teki tüm eğitim kurumlarının içinde olduğu gruptur. Alan adının "edu" kısmı, aynı zamanda üst düzey alan olarak da adlandırılır. Çizelge 2.1'de çeşitli standart üst düzey alanlar görülmektedir.

Öte yandan, ".tr" gibi uzantılar da ülkeleri temsil etmektedir (".tr" – Türkiye gibi).

URL'nin her zaman kullanılmayan üçüncü kısmı da bulunmaktadır ve bu kısım Web sunucusundaki veya uzak bilgisayardaki dizinleri ve belirli dosyaları göstermektedir. Örneğin, "http://www.uludag.edu.tr/english/index.html" URL'sinde "english" bir bilgisayar dizini ve "index.html" ise bir dosyadır.

Çizelge 2.1: Standart Üst Düzey Alanlar

Uzantı	Standart Üst Düzey Alan Adı
.com	Ticaret (Commercial)
.edu	Eğitim (Educational)
.gov	Kamu (Governmental)
.mil	Askeri (Military)
.net	Ağ (Network)
.org	Kurum (Organization)

URL, bir Web dosyasının bulunduğu yeri tam olarak ifade eden Web adresidir. İnternet'te iki farklı türde adres kullanılmaktadır: (i) bilgisayar adları ve (ii) bilgisayar numaraları. Bilgisayar adları, insanların URL'yi tanıyabilmeleri için, bilgisayar numaraları ise bilgisayarların URL'yi tanıyabilmeleri için kullanılmaktadır. Bilgisayarlar aslında, İnternet Protokolü (Internet Protocol – IP) adreslerini kullanmaktadır. IP adresleri, noktalarla birbirinden ayrılan dört grup sayıdan oluşmaktadır. İnternet Servis Sağlayıcı'ları (Internet Service Provider – ISP) ve Web siteleri, İnternet kullanıcıları tarafından girilen bilgisayar adlarını, bilgisayar numaralarına dönüştüren Alan Ad Sistemi'ni (Domain Name System – DNS) kullanırlar. Yani insanlar, "http://www.uludag.edu.tr" URL'sini girdiklerinde, DNS bilgisayarı o adı yorumlar ve ona karşılık gelen bilgisayar numarasını araştırır.

İnternet kullanıcılarının, İnternet üzerinde aradıkları bilgilere ulaşabilmeleri için konulara göre düzenlenmiş başlıkların yer aldığı İnternet siteleri bulunmaktadır. İnternet kullanıcıları bu başlıkları tarayarak aradıkları bilgilere ulaşabilirler. Başlıkları tarama sırasında genel kapsamlı kategorilerden (eğlence, coğrafya, tarih) daha dar kapsamlı kategorilere (sinema, şehirler, savaşlar) doğru ilerlenir ve sonunda belirli bir konuya (Türk Sineması, Bursa, Birinci Dünya Savaşı) ulaşılır. İnternet üzerinde bu şekilde düzenlenmiş olan sitelere İnternet indeksleri (index) denir.

İndeks kullanarak arama yapılırken, doğrudan indeks Web sayfasında bulunan seçenekleri kullanma olanağı bulunmaktadır ve dolayısıyla aranan konuya ulaşmak için çeşitli anahtar sözcükler kullanmaya gerek yoktur.

Verimli bir arama yapmak için öncelikle bazı temel özelliklerin öğrenilmesi gerekmektedir. Web indeksleri, başka sayfalara bağlanmayı sağlayan, çeşitli sözcüklerden oluşan listelerdir. İndekste ana konulardan başlayarak ve giderek alt düzeylere inerek, aranılan konuya doğru ilerlenebilir.

Öte yandan, indeks kullanan kullanıcılar, kendileri için en uygun seçeneği indeks üzerinde kendileri bulmak ve ayrıca, karşılarına çıkacak olan ve çok sayıda sayfadan oluşan sayfa listesinde hangi sayfaların işlerine yarayacağını kendileri belirlemek zorundadırlar. Çoğu kullanıcının böyle bir işlemi tam olarak yerine getirecek kadar zamanı olmamaktadır. Geçmiş yıllarda, İnternet kullanıcılarının aradıkları bilgilere ulaşmak için indeksleri kullanmaları mantıklı olmakla birlikte, günümüzde İnternet'in çok büyük bir bilgi kaynağı haline gelmesiyle, indeksleri kullanarak arama yapmak giderek daha da zorlaşmıştır. Bu yüzden, günümüzde İnternet kullanıcılarının büyük bir çoğunluğu, İnternet'te aradıkları bilgilere ulaşabilmek amacıyla arama motorlarından (search engine) yararlanmaktadırlar.

Web üzerinde bulunan bazı Web siteleri, İnternet kullanıcılarının aradıkları bilgileri bulmalarına yardım etmek amacıyla tasarlanmışlardır. Bu tür siteler, arama motoru (search engine) adı verilen bir yazılım içerirler. Arama motorları, aranılan bilgiye ulaşılmasına yardımcı olan araçlardır. İnternet kullanıcıları, ihtiyaçları olan bilgiyi aratmak için anahtar sözcükler yazarak arama işlemi yaptırabilirler. Örneğin, Uludağ Üniversitesi hakkında bilgi almak için "Uludağ Üniversitesi", "Uludağ University" gibi anahtar sözcükler kullanılabilir. Arama işlemi sona erdiğinde, girilen anahtar sözcüklerin bulunduğu Web siteleri listelenmektedir.

İndeks kullanarak arama yapılırken, indeks sayfaları üzerindeki seçenekleri inceleyecek ve buna bağlı olarak uygun seçimi yapacak olan kullanıcıdan başkası değildir. Arama motorları kullanıldığında, bu işlemi bir bilgisayar kullanıcının yerine ve çok daha hızlı olarak yapmaktadır. İndeksler gibi arama motorları da Web üzerindedir.

Arama motorları, kayıtlı tüm Web sayfalarının sözcüklerinden oluşturulmuş büyük veritabanlarına sahiptirler. Kullanıcı bir arama yaptığında, arama motoru kullanıcının belirttiği anahtar sözcüklerin kendi veritabanında bulunup bulunmadığını kontrol etmekte ve eğer bu sözcükleri kendi veritabanında bulursa, bu sözcükleri içeren

Web sitelerinin bir listesini kullanıcıya sunmaktadırlar. Kullanıcı, karşısına çıkan listedeki Web sayfalarından herhangi birisinin bağlantısına tıklayarak o siteye erişebilir.

Belirtilen anahtar sözcüklerin, arama motorunun veritabanındaki milyonlarca sözcük ile karşılaştırılması olanağının bulunması avantaj sağlamakla birlikte, belirli bir konu ile ilgili doğru anahtar sözcüklerin bulunması önemlidir. Başarılı bir arama yapabilmek için aşağıdaki yöntemler uygulanabilir:

- Anahtar sözcükler doğru belirlenmelidir.
- Arama sonucunda çok az sayıda belge listeleniyorsa, veya hiç listelenmiyorsa arama kapsamı genişletilmeli; tersi durumda ise arama kapsamı daraltılmalıdır. Arama kapsamını genişletmek için anahtar sözcük sayısı azaltılabilir, daraltmak için de arttırılabilir.
- Anahtar sözcükler girilirken, arama yapılan dilin yazım kuralları ve kısaltmalar dikkate alınmalıdır.

Günümüzde, gelişmiş arama motorlarında aşağıdakilere benzer arama seçenekleri de bulunmaktadır:

- Belirli terimleri içeren / içermeyen arama yapma.
- Mantıksal operatörleri (AND, OR, NOT gibi) kullanarak arama yapma.
- Belirli bir ülke veya dilde arama yapma.
- Güncelleme zamanına göre arama yapma.
- Çeşitli dosya türlerine göre arama yapma.
- Belirli bir site veya alan üzerinde arama yapma, veya belirli bir site veya alan üzerinde olmayan sitelerde veya alanlarda arama yapma.
- Aynı siteden olan Web sayfalarını sınırlandırma seçeneğiyle arama yapma.
- Sayfanın herhangi bir yerinde, başlığında, içeriğinde, adreslerinde veya bağlantılarında arama yapma.
- Bir sayfaya benzer sayfaları arama seçeneği ile arama yapma.

- Bir sayfaya bağlantısı olan sayfaları arama seçeneği ile arama yapma.
- Yetişkinlere yönelik içeriğin filtrelenebilmesi seçeneği ile arama yapma.

İzleyen kısımda, Web üzerindeki en çok kullanılan arama motorlarından olan AltaVista, Google, MSN ve Yahoo! arama motorları hakkında kısa bilgiler verilmiştir.

Altavista Arama Motoru (http://www.altavista.com)

Digital Equipment Corporation, AltaVista arama motorunu Aralık 1995'te hizmete sokmuştur ve Web sayfalarını bulmak için kullandığı veritabanının büyüklüğü nedeniyle kısa sürede popüler olmuştur.

DEC'in Scooter diye adlandırdığı bir bilgisayar programı hergün milyonlarca Web sayfasını tarayıp, AltaVista bilgisayarına getirmektedir. Bu Web sayfaları, AltaVista'nın indeks yazılımı tarafından indekslenmekte ve AltaVista'nın arama motoruna anahtar sözcükler girildiğinde bu indeksle karşılaştırılmakta ve eşleştirilen Web sitelerinin bağlantı listesi kullanıcıya sunulmaktadır.

Şekil 2.1'de AltaVista arama motoru görülmektedir.

Google Arama Motoru (http://www.google.com)

Stanford'da doktora yapan iki öğrenci, Larry Page ve Sergey Brin, Google arama motorunu 1998'de kurmuştur. Şirket kısa sürede büyüyerek, Haziran 1999'da, 25 milyon dolar yasal sermayeye sahip olduğunu duyurmuştur. Google arama motoru, servislerini kendi sitesinden vermektedir. Şirket ayrıca, içerik sağlayıcı firmalara özel web arama çözümleri de sunmaktadır.

Milyarlarca web sayfasına ulaşarak, yarım saniyeden daha kısa bir sürede, arama terimleriyle ilgili sonuçlar getiren Google arama motoru, günde 100 milyondan fazla kullanıcının sorgularına cevap vermektedir.

Şekil 2.1: AltaVista Arama Motoru

Google arama motoru PageRank olarak adlandırılan teknolojiyi kullanmaktadır. Bu teknoloji, ağ sayfalarının önemini nesnel bir ölçeğe uyarlamaktadır; bu, 500 milyon değişken ve 2 milyar terimden oluşan bir denklemin çözülmesiyle hesaplanmakta ve Web'in çok sayıda bağlantılı yapısını, düzenleyici bir araç olarak kullanmaktadır. Bir Web sayfasından diğer bir Web sayfasına olan her bağlantıyı, bağlantının olduğu sayfadan, bağlantı olan sayfaya bir "oy" olarak yorumlayan Google arama motoru, bir sayfanın önemini aldığı oylarla belirlemekte, ayrıca oyu veren sayfayı da incelemektedir.

Şekil 2.2'de Google arama motoru görülmektedir.

MSN Arama Motoru (http://search.msn.com)

Yazılım şirketi Microsoft tarafından desteklenen MSN arama motoruna, MSN portalı üzerinden de ulaşılabilmektedir. MSN arama motorunun arama servisleri çok sayıda seçenek içermektedir.

MSN arama motoru üç bileşenden oluşmaktadır; (i) MSNBot (Web crawler), (ii) indeks oluşturucusu (bilgi deposu) ve (iii) sorgu sunucusu (arama terimleri-site indeksi eşleştirici).

Şekil 2.2: Google Arama Motoru

MSNBot, aralarındaki bağlantılardan yararlanarak, Web sayfalarını bulmakta ve indekslemektedir. Daha önce bulunmuş olan Web sayfaları da incelenmektedir. Daha sonra, bu milyarlarca Web sayfası, indeks oluşturucusu tarafından, çeşitli kriterlere göre sınıflandırıldıktan sonra, bu sınıf içinde sıralanmaktadırlar. Sıralama işleminde, sayfanın dili, sayfaya olan linkler ve sayfa içeriğinin kalite ve kapsamı önemli ölçütler olarak kullanılmaktadır. Son bileşen sorgu sunucusu da kullanıcının girdiği terimlerle, indeksteki sıralanmış Web sayfalarını eşleştirmektedir. Bu işlemin yapılması için kullanıcının girdiği terimler incelenmekte ve sıralanmış Web sayfalarıyla karşılaştırılarak, bu sayfaların kullanıcının girdiği terimlerle ne kadar ilgili olduğu belirlenmektedir.

Şekil 2.3'te MSN arama motoru görülmektedir.

Şekil 2.3: MSN Arama Motoru

Yahoo! Arama Motoru (http://search.yahoo.com)

Yahoo! arama motoruna, Yahoo! portalı üzerinden de ulaşılabilmektedir.

Yahoo! arama motoru, Yahoo! Arama Teknolojisi'ni (Yahoo! Search Technology – YST) kullanmaktadır. YST, kullanıcılarına kapsamlı, güncel ve ilgili sonuçlar sunmak amacıyla, İnternet üzerindeki milyarlarca belgeyi sürekli olarak değerlendirmektedir. Kullanıcılar tarafından arama yapıldığında, YST kendi Web sayfaları veritabanını tarayarak, bu sayfaların girilen arama terimleriyle ilgililiğini belirleyip, sonuçları sıralanmış olarak listelemektedir. YST Web sayfalarını belirli arama terimlerine göre sıralarken, metin, başlık ve tanım doğruluğu, kaynak, ilgili bağlantılar ve diğer benzersiz belge karakteristiklerini de içeren unsurları incelemektedir. Yahoo! arama motoru veritabanı her gün güncellenmektedir.

Şekil 2.4'te Yahoo arama motoru görülmektedir.

Şekil 2.4: Yahoo! Arama Motoru

2.2 – Kaynak Araştırması

Son yıllarda, arama motorları kullanıcı davranışlarını inceleyen çeşitli çalışmalar yapılmıştır.

Bilal ve Kirby (2002), çalışmalarında çocuklar ile yetişkinlerin Web arama motorları kullanımlarını karşılaştırmışlardır. Yedinci-sınıf düzeyindeki çocuklar ve doktora öğrencilerinden oluşan iki grubun Yahooligans! Web arama motorunu / dizinini kullanma özellikleri incelenmiştir. Çalışmada, çocuk ve yetişkinlerin, arama hareketleri (searching moves), tarama hareketleri (browsing moves), geriye dönüş hareketleri (backtracking moves), döngü hareketleri (looping moves), ekran kaydırma (screen scrolling), hedef yerleştirme ve sapma hareketleri (target locating and deviation moves) ve de yapmaları gereken işi tamamlamaları için geçen süreyi kapsayan Web işlemlerinin genel yapısı incelenmiş ve kıyaslanmıştır. 9 çocuk ve 14 yetişkinin katıldığı çalışma, Web tarayıcılarındaki çevirim-içi hareketleri kaydeden ve tekrar gösteren bir yazılım olan Lotus ScreenCam yazılımı kullanılarak çevrim-içi olarak tutulmuştur. Grupların Yahooligans!'ı kullanma performanslarının ölçümünde Bilal'in Web Aykırılık Ölçüsü

(Bilal's Web Traversal Measure) kullanılmıştır. Çalışma sonucunda yetişkinlerin %89'u doğru sonuca ulaşırken, çocuklarda bu oran %50 olmuştur. Yetişkinler aranan konunun bulunmasında daha başarılı performans göstermelerine rağmen, genel bilgi-arama davranışlarını çocuklarla paylaşmışlardır. İki grup arasındaki temel farklılıklar, geriye dönüşlerden kurtulma yeteneği, gezinti stili ve göreve odaklanma konularında ortaya çıkmıştır. Ayrıca, yetişkinlerin basit kullanıcı arabirimine sahip arama motorunu kullanmış olmaları performanslarını arttırmıştır. Öte yandan, her iki grubun da arama motorunu kullanırken zorlandıkları, gruplardaki kişilerin arama motorunun kullanımı konusunda yeterli bilgiye sahip olmadıkları belirtilmiştir. Bundan başka yazarlar, eğitim sonucunda bu konuda gelişme sağlanabileceğini ve çocukların ayrıca bilgiyi tarama, inceleme, değerlendirme, çıkarma ve sentezlemeyi öğrenmeleri gerektiğini belirtmişlerdir.

Can ve arkadaşları (2003), Web arama motorlarının otomatik olarak değerlendirilmesini sağlayacak bir yaklaşım geliştirmişlerdir. Çalışmada, Internet kullanıcılarının, Web'de aradıklarını bulabilmek amacıyla yoğun bir şekilde arama motorlarından yararlandıkları belirtilmiş ve bu yüzden en iyi çalışan arama motorlarının belirlenmesinin öneminden bahsedilmiştir. Öte yandan, arama motorlarının insanlar tarafından değerlendirilmesinin maliyetli olabileceği belirtilmektedir. Bu yüzden, bir otomatik Web arama motoru değerlendirme yöntemi (automatic Web search engine evaluation method) AWSEEM geliştirilmiştir. Bu yöntemin, insan-bazlı değerlendirme yönteminin yerine kullanılabileceği görülmüştür. Yapılan değerlendirme çalışması için 8 arama motoru ve 25 sorgu kullanılmıştır. Çalışmada değerlendirmeye alınan 8 arama motoru (AlltheWeb, Alta Vista, HotBot, InfoSeek, Lycos, MSN, Netscape ve Yahoo!) arasından, dikkate alınan değerlendirme ölçütlerine göre, performans açısından öne çıkanlar Alta Vista ve Yahoo! olarak bulunmuştur.

Caramia ve arkadaşları (2003), tematik bir arama motorunda, veri madenciliği ile arama sonuçlarını geliştirmek için bir çalışma yapmışlardır. Burada uygulanan yöntemde, başlangıçta ele alınan bir Web sayfaları kümesi bulunmaktadır. Bu küme bir sorgu sonucu elde edilmiş olan Web sayfaları kümesi olabilir. Yönteme göre bu başlangıç kümesinin, bilgiyi daha iyi gösteren bir alt kümesi belirlenmelidir. Bu işlem üç aşamada gerçekleştirilmektedir. İlk aşamada, arama içeriği ve kullanıcı profili; bir karakteristik vektörü oluşturmak için kullanılan, sonlu bir anlamlı kelime veya sayfa

karakteristikleri kümesi oluşturmakta kullanılır. İkinci aşamada bu sayfalar, bir kümeleme algoritmasıyla, konularına göre benzer alt kümelere bölünmektedirler. Bu aşamada, sayfalar, her sayfanın kendi kümesi içinde de bir sırası olduğundan iki-boyutlu bir sıralamayla bulunmaktadırlar. Burada amaç, kullanıcıya kümeler tarafından belirlenen yapıyı ve orijinal fonksiyonu dikkate alan bir liste sunmaktır. Bu işlem de üçüncü aşamada gerçekleştirilmektedir. Bu aşamada, her bir kümedeki daha yüksek puana sahip olan sayfalar alınarak daha sonra bir genetik algoritma ile analiz edilen bir başlangıç nüfusu oluşturulmaktadır. Yapılan çalışma sonucunda yazarlar, kalitenin basitçe sayfa kalitelerinin toplamı yerine, altkümenin genel karakteristikleri olarak alındığı durumda; yöntemin iyi kalitede küçük sayfa altkümeleri seçiminde etkili olabileceğini belirtmişlerdir. Aynı zamanda, genetik algoritma ve kümeleme algoritması gösteriminin de kişisel bir bilgisayarda birkaç saniye gibi kısa bir sürede sonuç elde edilmesine olanak sağladığını da açıklanmışlardır.

Chuang ve Chien (2002), bir sorgu-sınıflandırma yaklaşımı geliştirmişlerdir. Bunun için üç arama motorunun (Dreamer, GAIS ve Openfind) kayıtları toplanmış ve bu kayıtlar üzerinde, popüler Web arama konularını gösterecek şekilde, 14 ana ve 100 alt kategoriden oluşan iki aşamalı bir konu sınıflandırması yapılmıştır. Yazarların çalışmalarında geliştirdikleri yaklaşım üç aşamadan oluşmaktadır; (i) sorgu terimi kayıt analizi, (ii) ilgili belge geri alma ve (iii) konu sınıflandırma. Sorgu terimi kayıt analizi, arama motoru kayıtlarının incelenmesiyle, konu sınıflandırma sistemi ve sınıflandırılmış terim kümesinin elde edilmesidir. İlgili belge geri alma, gerçek arama motorlarının birlikte çalışarak getirdikleri en çok ilgili belgelerin getirilmesi ve konu sınıflandırma da her bir konu terimi için uygun konu kategorileri tanımlamak içindir. Yazarlar, deneysel sonuçların, kullanıcılar tarafından verilen çeşitli yeni terimlerin otomatik olarak sınıflandırılabildiğini gösterdiğini ve bu yüzden, çalışmalarının, sorgu terimleri kullanarak Web sınıflandırmaları yapmakta veya var olanları geliştirmekte iyi bir başlangıç olabileceğini belirtmişlerdir.

Liaw ve Huang (2003) çalışmalarında, Internet'in çok hızlı büyümesiyle Web'in çok fazla bilgi içerir duruma geldiğini ve artık arama motorlarını kullanarak Web araması yapmanın, Internet'in temel kullanım şekillerinden biri haline geldiğini belirtmişlerdir. Yazarlar, kişilerin, arama motorlarının bu şekilde bir bilgi elde etme aracı olarak kullanılmasına karşı yaklaşımlarını incelemişlerdir. Kişilerin

yaklaşımlarının belirlenmesinde çeşitli faktörler dikkate alınmıştır (kelime-işlem yazılımları deneyimi, işletim sistemleri deneyimi, arama motorlarının kalitesi, İnternet cevap süresi vb.). Yazarlar, bilgisayar deneyiminin (özellikle işletim sistemi ve kelime-işlem yazılımları deneyiminin), kişilerin arama motorlarını kullanmasındaki en önemli faktörlerden biri olduğunu belirtmişlerdir. Bundan başka, sistem kalitesi, kişilerin sistemden hoşlanmaları veya sistemi kullanımı kolay bulmalarında etkili olmaktadır. Yazarlar çalışmalarından elde ettikleri sonuçların, Teknoloji Kabul Modeli (Technology Acceptance Model – TAM) ve motivasyon teorilerini destekler nitelikte olduğunu belirtmişlerdir.

Jansen ve Spink (2003) çalışmalarında, arama motoru kullanıcılarının davranışlarının genel olarak aynı olup olmadığını incelemişlerdir. Yazarlar, kullanıcı davranışlarının incelenmesi amacıyla yapılan çalışmaların çoğunun Amerika odaklı olduğunu ve dünyanın diğer yerlerindeki kullanıcılar için bu türdeki çalışmaların çok az olduğunu belirtmişlerdir. Çalışmada, Avrupa'da Web araması yapan kullanıcıların davranışlarının, Amerika'dakilere göre farklı özellikler gösterebildiği belirtilmiştir. Bu amaçla yazarlar, Avrupa merkezli ve çoğunlukla Avrupalıların kullanmakta olduğu bir arama motoru olan AlltheWeb.com arama motorunun kullanıcıları üzerinde böyle bir çalışma yapmışlar ve bu arama motorunun kayıtlarını incelemişlerdir. Yazarlar çalışmalarında; Avrupa'daki kullanıcıların Web araması karakteristiklerindeki eğilimleri, bu kullanıcıların kaç tane sayfa görüp, bunlar üzerinde ne kadar süre harcadıklarını ve inceledikleri belgelerin konu bakımından ne kadar ilişkili olduklarını belirlemeye çalışmışlardır. Buna göre, Avrupa'da Web aramasının belirli yönlerde geliştiği sonucunu bulmuşlardır. Sorgu uzunluklarında azalmalar bulunmaktadır. Bazı konularla ilgili olarak yapılan arama sayılarında azalma olduğu görülmüştür. Kullanıcıların çoğunluğunun incelediği Web belgesi sayısı beşten daha az olup, bir Web belgesi üzerinde geçirdikleri süre de saniyelerle ölçülmektedir. Bu kullanıcılar tarafından görülen belgelerin yaklaşık olarak %50'si konu bakımından ilişkilidir.

Özmutlu ve arkadaşları (2003) Excite ve FAST arama motorları verilerini incelemişler ve bu arama motorları kullanıcılarının aramalarında gün içinde değişiklikler olduğunu belirtmişlerdir. Yazarlar yaptıkları çalışmada, oturum ve sorguların gelişleri ve süreleri gibi bazı karakteristiklerin, sabah saatlerinde en yüksek düzeylerinde olup, ilerleyen zamanlarda azalmakta olduğunu belirtmişlerdir. Sorgu

başına terim sayısı cinsinden sorgu kalitesi ve sorguların yeniden düzenlenmesi gibi diğer karakteristikler gün boyunca aynı kalmaktadır. Yazarlar, bu analizden elde edilen sonuçların ve daha başka veri kümelerinin de bu şekilde incelenmesinin, arama motorlarının arama yapılarını değiştirmelerinde ve farklı zamanlara göre kaynak atamaları yapmalarında faydalı olabileceğini belirtimişlerdir.

Özmutlu ve arkadaşları (2002), yaptıkları çalışmada, büyük Web veri kümelerinin analizi için etkili bir örnekleme stratejisi geliştirmek için Poisson ve sistematik örnekleme tekniklerini karşılaştırmışlardır. Yazarlar çalışmalarında Excite arama motoru verilerini kullanmışlardır. Web arama motorları kayıtlarının çok büyük boyutlara ulaştığını belirten yazarlar, bu boyutlardaki veriler üzerinde, istatistiksel yöntemleri uygulamanın zorluğuna ve yazılım paketlerinin sıralama gibi basit işlemleri bile yapmakta zorlandığına dikkat çekmişlerdir. Bu amaçla, istatistiksel olarak kitleyi doğru bir şekilde yansıtacak ve veri miktarını analiz edilmeye uygun boyutlara getirecek etkili bir örnekleme tekniği geliştirmek gerektiğini belirten yazarlar, çalışmalarında Poisson ve sistematik örnekleme tekniklerini Excite arama motoru verilerine uygulamışlardır. Çalışmaları sonucunda, Poisson örnekleme tekniği kullanılarak kitlenin istatistiksel özelliklerini kaybetmeden örnekleme yapılabileceğini ve bu tekniğin, en yaygın kullanılan örnekleme tekniği olan sistematik örneklemeden daha iyi sonuç verdiğini bulmuşlardır. Yazarlar, yaptıkları çalışmaların geliştirilerek, Web kullanıcılarının arama davranışlarının daha iyi anlaşılabileceğini ve daha etkili arama motorları tasarlanmasına katkıda bulunabileceğini belirtmişlerdir.

Spink (2002), yaptığı çalışmada, NEC araştırma enstitüsünde geliştirilmiş olan, Inquirus adındaki Web arama aracını değerlendirmiştir. Bu çalışmadaki yaklaşımda, Inquirus etkinlik (effectiveness) ve kullanılabilirlik (usability) olmak üzere iki açıdan değerlendirilmiştir. 22 kişi bu değerlendirme çalışmasına katılarak, kendilerine ait, kişisel arama konularında Inquirus'u kullanarak arama yapmışlardır. Kullanıcı anketleri ve Inquirus kayıtları üzerinde veri analizi gerçekleştirilmiştir. Yapılan çalışma sonucunda; kullanıcıların Inquirus'u kullanılabilir bir araç olarak değerlendirdikleri, kullanıcıların arama konularında ve süreçlerinde değişme olduğu, farklı kullanıcılar için bu değişimlerin de farklı olduğu belirtilmiştir.

Spink ve Ozmutlu (2002), soru biçimli Web sorgularının karakteristiklerini incelemişlerdir. Çalışmada, Ask Jeeves gibi bazı arama motorlarının, kullanıcıların soru biçimli sorgular girmelerini gerektirdiği belirtilmiştir. Yazarlar çalışmalarında iki arama motorunu incelemişlerdir. Bunlar, sorguların soru biçiminde girilmesine yönlendiren Ask Jeeves ve bu şekilde bir yönlendirme yapmayan Excite arama motorlarıdır. Yazarlar çalışmalarında bu arama motorlarındaki sorguların soru biçiminde olmasının yaygınlığının, her soru biçimli sorgu için ortalama terim sayısının, soru biçimli sorgular için genel başlangıç terimlerinin, soru biçimli sorgularda Boolean operatörleri ve soru işareti kullanılıp kullanılmadığının ve soru biçimli sorgular için genel konuların ne olduğunun belirlenmesi amacıyla incelemeler yapmışlardır. Bu amaçla, bu arama motorlarından alınan çok sayıda kayıt incelenmiş ve Ask Jeeves sorgularının %50'sinin, Excite sorgularının ise %1'inden daha azının soru biçiminde olduğu; kullanıcıların çoğunun, sorguda küçük düzenlemeler yaparak soru biçiminde sadece bir sorgu girdikleri; soru yapılarının benzer olduğu; en yaygın soru biçiminin "… nereden bulabilirim?" ("Where can I find …") şeklinde olduğu ve soru-olmayan sorguların istek biçiminde olduğu görülmüştür. Çalışmada, genel olarak, anahtar sözcük, Boolean, soru ve istek olmak üzere, dört tip kullanıcı sorgusu belirlenmiştir. Sonuç olarak, yazarlar, soru biçimli Web sorgularının yapısı için genel kalıplar bulunduğunu ve bu kalıpların daha başka veriler için de test edilmesi gerektiğini belirtmişlerdir. İleriki araştırmalarda, soru biçimli ve soru biçimli olmayan sorgulardan hangisinin daha fazla sayıda anlamlı terim içerdiğinin belirlenmesi, soru ve istek biçimli sorgu yapılarının nasıl geliştirilebileceği gibi konular hakkında çalışmalar yapılabileceği belirtilmiştir.

Vaughan (2003), çalışmasında, arama motorlarının değerlendirilmesi için yeni ölçüler önermiş ve test etmiştir. Yazar, arama motorları için kullanılan değerlendirme ölçülerinin duyarlık (precision) ve anma (recall) olduğunu belirtmiştir. Bunların geleneksel değerlendirme ölçüleri olduğu ve bunlara ek olarak bazı yeni ölçülerin de geliştirildiği belirtilmiştir. Çalışmada, bu yeni ölçülerin test edilmesi amacıyla üç arama motorunun kayıtları kullanılmıştır (Alta Vista, Google ve Teoma). Geleneksel ölçülerin tamamlayıcısı olarak önerilen bu yeni ölçüler, sonuç sıralama kalitesi (the quality of result ranking) ve en üstte sıralanan sayfaları getirme yeteneği (the ability to retrieve top ranked pages) şeklindedir. Test edilen bu yeni ölçüler ile geleneksel ölçüler arasındaki fark; yeni ölçülerde test belgelerinin sürekli sıralaması (en fazla ilgili olandan en az

ilgili olana doğru sıralama) baz alınırken, diğerlerinde kesikli ilgi yargısı (ilgili, kısmen ilgili, ilgisiz gibi) bulunmasıdır. Dört sorguya karşılık gelen dört Web sayfası kümesi, arama motorlarınca getirilmiş ve bunların her bir arama motorundaki sıralamaları da kaydedilmiştir. Yapılan çalışma sonuçları, yeni ölçülerin, incelenen üç arama motorunun performansını ayırt edebildiğini göstermiş ve bu ölçülere göre en iyi performansı gösteren arama motoru Google olmuştur. Farklı ölçülerin kullanıldığı başka bir çalışmada Google'ın yine daha iyi bir performans gösterdiği ve aynı zamanda, Google'ın göreceli üstünlüğünün, bu arama motorunun Web üzerindeki en popüler arama motoru olmasından da anlaşılmakta olduğu belirtilmiştir. Öte yandan, yazar, bu çalışmanın, değerlendirmeye alınan arama motorlarının kalitesini kesin olarak ortaya koymadığını, bunun için daha fazla verinin incelenmesi gerektiğini belirtmiştir.

Spink ve arkadaşları (2002), yaptıkları çalışmada, cinsellikle ilgili bilgi için Web araması konusunu incelemişlerdir. Yazarlar çalışmalarında, cinsellikle ilgili sorguların oranını belirlemeyi ve cinsellikle ilgili ve cinsellikle ilgili olmayan sorguların karakteristiklerini incelemeyi amaçlamışlardır. Bunun için Excite arama motorunun verilerinden alınan bir örnek üzerinde inceleme yapılmıştır. Yazarlar, diğer aramalarla kıyaslandığında, cinsellikle ilgili aramalardaki sınırlı sözcük sayısının, bu aramanın ilginç bir özelliği olduğunu belirtmektedirler. Bu sorgularda daha az benzersiz terim bulunmakta olup, cinsellikle ilgili terimler sorgularda sıkça tekrarlanmaktadır. Ayrıca, cinsellikle ilgili olan oturumlar, cinsellikle ilgili olmayanlara göre daha uzun olup, daha fazla sorgu içermektedirler. Genel olarak, cinsellikle ilgili olan sorgular, diğerlerine göre daha uzun olup, cinsellikle ilgili bir oturum büyük bir olasılıkla 20 sorgudan daha uzun olmaktadır. Bundan başka, cinsellikle ilgili arama yapanlar, diğerlerine göre daha fazla sayıda sayfa görmektedirler. Cinsellikle ilgili oturumlar, görüntü indirmenin zaman alması dolayısıyla da daha uzun sürebilmektedir. Yazarlar, genel olarak cinsellikle ilgili arama yapanların, diğerlerine göre daha fazla zaman ve çaba harcamaya istekli olduklarını belirtmişlerdir. Yazarlar ayrıca, yaptıkları çalışmanın tek bir arama motoru verilerine dayalı olmasından dolayı sınırlı olduğunu ve biri Avrupa diğeri de Amerika bazlı olmak üzere iki ayrı arama motoru kayıtlarını, bu şekilde incelemekte olduklarını ve aynı zamanda tıp ve sağlıkla ilgili Web araması konusunda benzer bir çalışma sürdürdüklerini belirtmişlerdir.

He ve arkadaşları (2002), arama motorları kullanıcı oturumlarındaki konu değişikliklerini tespit etmek için olasılık (delil) birleştirme yaklaşımı uygulamışlardır. Çalışmada, arama motoru kullanıcılarının, arama konularını değiştirmeleri halinde, bunu otomatik olarak tespit edecek bir yaklaşım önerilmektedir. Bu yaklaşım, arama motoru kayıtlarından hesaplanan olasılıkları (delilleri), Dempster-Shafer teorisini kullanarak birleştirmektedir. Yaklaşım, Reuters arama motorunun kayıtları üzerinde test edilmiştir. Çalışmada kullanılan veriler, kullanıcıların IP adresleri, arama süreleri ve sorgularından oluşmaktadır. Çalışmada, öncelikle, belirli bir kayda ait "ti" ve "sp" değerleri incelenerek "konu değişikliği yok" ve "konu değişikliği var" olasılıkları hesaplanmaktadır. Daha sonra, bu olasılıklar, Dempster-Shafer teorisi kullanılarak birleştirilmekte ve her kayıt için tek bir olasılık elde edilmektedir. Elde edilen olasılıkları ikili değişkenlere ("konu değişikliği yok" veya "konu değişikliği var" şeklinde) dönüştürmek amacıyla bir eşik değeri kullanılmaktadır. Sonuçta, örnekteki her kayıt için "konu değişikliği yok" veya "konu değişikliği var" şeklinde atamalar yapılmaktadır. Yapılan bu atamalar, insanlar tarafından yapılan atamalarla karşılaştırılmıştır. Yaklaşımın performansını değerlendirmek için "duyarlık" ve "anma" performans ölçülerini birlikte dikkate alan bir performans ölçüsü kullanılmıştır. Bu performans ölçüsünün en büyük değerini alabilmesinde, eşik değeri ve ağırlıklardan oluşan parametreler belirleyici olmaktadır ve en iyi sonucu elde edebilmek için bu parametrelerin en uygun değerlerini alması gerekmektedir. Bu yüzden, yazarlar, incelenen veri örneğindeki kayıtların yaklaşık yarısını, bu parametrelerin en uygun değerlerinin belirlenmesi için kullanmışlardır. Bu amaçla, bir genetik algoritma kullanılarak, en uygun parametre değerleri belirlenmiş ve veri örneğinin ikinci kısmında bu parametre değerleri kullanılarak yaklaşım test edilmiştir. Yazarlar yaklaşımlarının arama motorları kullanıcı oturumlarındaki konu değişikliklerini belirlemede başarılı olduğunu belirtmişlerdir.

3 – MATERYAL VE YÖNTEM

3.1 – Materyal

3.1.1 – Delil Birleştirme Yöntemleri ve Dempster-Shafer Teorisi

Yapay zeka literatüründe delil birleştirmek amacıyla kullanılan başlıca üç yöntem bulunmaktadır: (i) Bayes teorisi, (ii) Dempster-Shafer teorisi ve (iii) Kabul (Endorsement) teorisi. Bunlardan Bayes ve Dempster-Shafer teorileri en çok bilinenlerdir. Bayes ve Dempster-Shafer teorilerindeki argümanlarının tanımı Shafer ve Pearl'de (1990) verilmiştir. Kabul teorisi, delil birleştirmek amacıyla kullanılabilecek diğer bir yöntemdir. Bu yöntemin, sayısal-olmayan yaklaşımı, Cohen (1985) tarafından geliştirilmiş ve farklı tipte delillerin birleştirildiği bazı otomatik haritalama uygulamalarında kullanılmıştır.

Kabul teorisi, kesin olmayan delilleri birleştirmek için sayısal-olmayan bir yaklaşımdır. Delil, bir hipotezin doğru olmasına katkıda bulunan inancın tanımlanması olarak verilir. Farklı inanç düzeyleri tanımlanır ve farklı inanç ve inançsızlık durumlarının tümüne bağlı olan, delilden gelen bir kabul verilir. Kabul ve inanç adları semboliktir. Bunların anlamları, uygulanabildikleri durumdan, nasıl birleştiklerinden ve nasıl sıralandıklarından elde edilir. Bu, problemin dört yönden tanımlanmasını gerektirir (Comber, Law ve Lishman, 2004):

- İnançlar belirlenmeli ve adlandırılmalıdırlar
- İnançların birleştiklerindeki kesişimleri, genel kabuller elde etmek için belirlenmelidir.
- Kabulleri sıralamak için bir sistem belirlenmelidir. Bu örnekteki kabul gücü, "kesin", "güvenilir", "beklenen", "belirtilmiş" ve "yanlış" şeklinde ilerlemektedir.
- Delil "yeterli derecede inanılır" olduğunda, karar vermek için kurallar tanımlanmalıdır. Bu karar, delilin kabulüne ve bu delilin kullanım amacına bağlıdır, ve bir hipotezde kesin inanç sağlayan delil parçasının süreci durdurmak ve sonuçlar çıkarmak için yeterli olması olabilir.

Kabul modeli, belirsizlik kaynaklarını açık hale getirmekte ve belirsizliği anlama konusunda, Bayes ve Dempster-Shafer yaklaşımlarından çok daha sezgisel bir yaklaşım ortaya koymaktadır. Birincisi, genel bilgiyi (uzman haritalama kuralları gibi), bir uzmanın kendi düşüncesini tam olarak yansıtmayabilen sayısal değerlere dönüştürmesine gerek kalmadan, doğal bir formda sunabilmektedir. İkincisi, sembolik yaklaşımı, gerçek-dünya problemlerinin gösterimi ve anlaşılması için kullanılabilirdir. Üçüncüsü, bu şekilde sonuç çıkarma, kısmi bilgiden sağlanabilmektedir. Kabul bazlı yaklaşımın cevap bulmaya çalıştığı soru, "Süreçteki belirsizlik kaynakları nelerdir ve nerede ortaya çıkmışlardır?" sorusudur. Bu cevabın anlamı, hangi kabullerin birleştiği ve nasıl sıralandıklarıyla yorumlanmaktadır.

Matematiksel olasılık teorisinin temeli 18. yy'a, Bayes'in 1763 yılındaki çalışmasına dayanmaktadır. Bu çalışma, istatistiksel sonuç çıkarma ve belirsizlik altında karar vermenin temellerini içermektedir. Karar analizinin temelleri, 1930 ve 1940'larda oluşturulmuştur. Wald, "bütün sınıf teoremi" (complete class theorem) adlı teoremi geliştirmiştir. Burada, istatistiksel karar problemindeki herhangi bir prosedürün yok edilebileceği veya en azından Bayes prosedürleri performansıyla eşleşebileceği belirtilmiştir.

Bazı yazarlar, zayıf bilgi kaynakları söz konusu olduğunda, geleneksel Bayes analizinin zorluğundan bahsetmişlerdir. Böyle durumlarda, "duyarlığın Bayes dogması" (Bayesian dogma of precision) kullanılabilir. Burada, belirsiz olan istatistiksel parametreler, geleneksel, belirli, istatistiksel dağılımlarla gösterilmelidirler.

Güçlüklerin bazıları, Wilson'un "Yetersiz Neden Prensibi" (Principle of Insufficient Reason) çalışmasından incelenebilir. 1 ile 6 arasında, bilinemeyen şansla, rasgele tamsayılar üreten bir birim olsun. Burada sonraki sayının 1 olacağı hakkındaki inanç nedir? Bayes yaklaşımına göre, bu 1/6 olacaktır. Genelde, "ayırt etme yapısı" (frame of discernmet) öznel olasılıklar içerse bile, bilgisizlik (ignorance) durumunda Bayes'e göre bu prensip kullanılmak zorundadır.

Bayes yaklaşımını, özellikle bilgisizliğin gösterimi açısından daha iyi anlayabilmek için konu bir örnekle ele alınmıştır (Beynon, Morgan, 2000). Aşağıdaki a önermesi söz konusu olsun:

"Ben King Roads, Cardiff'te yaşıyorum".

Bu durumda a'da bir Bayes inancı $P(a)$ nasıl oluşturulabilir? Öncelikle Θ ile gösterilen bir "ayırt etme yapısı" ve Θ'nın a'yı gösteren bir alt kümesi A tanımlanmalıdır. Daha sonra bir Bayes inancına ulaşabilmek için yetersiz neden prensibi uygulanmalıdır. Burada problem, kaç tane Cardiff yolunun bulunduğuna bağlı olarak, seçilebilecek çok sayıda Θ olmasıdır. Eğer sadece iki cadde belirlenebiliyorsa, bu durumda, $\Theta = \{x_1, x_2\}$ ve $A = \{x_1\}$ olur. Bu durumda, yetersiz neden prensibi, Θ'daki atanmış öznel olasılıklara rağmen $P(A)$'nın 0,5 olacağını söyler. Eğer Cardiff'te yaklaşık olarak 1000 yolun olduğu belirlendiyse, bu kez diğer x_i'ler diğer yolları göstermek üzere $\Theta = \{x_1, x_2, \ldots, x_{1000}\}$ ve yine $A = \{x_1\}$ olur. Bu durumda, yetersiz neden prensibi, $P(A) = 0{,}001$ sonucunu verir.

Bu yapıların ikisi de mantıklı olabilir, ancak A'ya atanan olasılık seçilen yapıya bağlı olmaktadır. Bu yüzden, birisinin Bayes inancı, sadece verilen bilgi ve geçmiş bilgisine bağlı olmayıp, aynı zamanda, bazen keyfi seçilen Θ'ya bağlı olmaktadır.

Dempster-Shafer teorisinin temeli A. P. Dempster'in, alt ve üst olasılık sınırları sistemi çalışmasına dayanmaktadır. Bunun ardından, öğrencisi G. Shafer 1976 yılında, "Delilin Matematiksel Teorisi" (A Mathematical Theory of Evidence) adlı kitabında, Dempster'in çalışmasına inanç fonksiyonlarıyla ilgili daha fazla açıklama eklemiştir. Özel olarak yapay zeka ile ilgili şekilde ortaya çıkmamakla birlikte, Dempster-Shafer teorisi, J. A. Barnett tarafından, inanç fonksiyonlarının yapay zeka literatürüne girişini gösteren bir makalede bulunmuştur. Çok sayıda yazar tarafından incelenmiş ve popülerliği artmışsa da daha çok yapay zeka ve uzman sistemler literatüründe, belirsizlik altında modelleme tekniği olarak kullanılmaktadır. Bu anlamda, istatistikteki daha geleneksel yöntemler ve Bayes karar teorisine göre bazı avantajlara sahip olduğu görülebilir. Hajck, Dempster-Shafer teorisi yöntemlerinin gerçek, pratik uygulamalarının az olduğunu, ancak teoriyi kullanan uygulamalarda bir artış olduğunu belirtmiştir. Dempster-Shafer teorisinin, geniş bir kullanım alanı olmasa da yüz tanıma, istatistiksel sınıflandırma ve hedef belirleme gibi konularda başarılı bir şekilde uygulanmaktadır. Diğer uygulamalar çok-kaynaklı bilgi etrafında yoğunlaşmıştır.

Dempster-Shafer modelinin temel özelliklerinden birisi, olasılık veya inanç ölçülerini birleştirmek için toplama zorunluluğunun olmamasıdır. Verilen bir önermeye

bağlı olmayan bir inancın, bunun tersine bağlı olması şart değildir. Aşağıda görüleceği gibi bu, "ayırt etme yapısı" kavramının daha esnek bir biçimde oluşturulmasına ve analiz edilmesine olanak sağlamaktadır. İnancın toplam ataması, sahip olunan bilgiye uyacak şekilde değişebilmektedir.

Dempster-Shafer teorisinin ikinci temel fikri, belirsizliğin sayısal ölçüsünün, örtüşen hipotez kümeleri ve alt kümelerine, ayrı hipotezler gibi atanabilmesidir. Bunun gösterimi için Comber ve arkadaşlarının (2004) çalışmasında verilmiş olan aşağıdaki suçlu belirleme ile ilgili bilgi ifadesi göz önüne alınsın (Beynon, Morgan, 2000).

Bay Jones öldürülmüştür ve suçlunun Peter, Paul ve Mary adındaki üç şüpheliden biri olduğu bilinmektedir. Bu durumda bir hipotez bulunmaktadır, ayırt etme yapısı, $\Theta = \{\text{Peter}, \text{Paul}, \text{Mary}\}$ şeklindedir. Sahip olunan tek delil, suçluyu gören kişinin, suçlunun %80 oranında erkek olduğundan emin olmasıdır, $P(erkek) = 0{,}8$. Bu belirsizlik ölçüsü, Dempster-Shafer terminolojisinde "temel olasılık ataması" (basic probablity assignment-*bpa*) olarak bilinir. Bu durumda, $\{\text{Peter}, \text{Paul}\}$ odak elemanına verilen değeri 0,8 olan bir m_1 *bpa*'sı bulunmaktadır, $m_1(\{\text{Peter}, \text{Paul}\}) = 0{,}8$. Geri kalan olasılıkla ilgili bir şey bilinmediğinden dolayı, bu, tüm ayırt etme yapısına atanır, $m_1(\{\text{Peter}, \text{Paul}, \text{Mary}\}) = 0{,}2$.

Buradaki önemli nokta, tekli kümelere yapılan atamaların, çok sayıda önerme içeren kümelere yapılan atamalarla aynı anda çalışabildiğinin fark edilmesidir. Önerme grupları için önceki olasılıkların Bayes ataması yapılabilmesine rağmen, böyle bir duruma, geleneksel Bayes yapısında izin verilmez. Schubert tarafından gösterildiği gibi bu anlamda, Dempster-Shafer teorisi Bayes teoreminin bir genelleştirilmesidir. Bu teori, uygun olmayan önceki olasılıkların atanması probleminden kaçınır ve uygun olmayan olasılıklar hakkında varsayımda bulunmaz.

Ele alınacak sonraki soru, *bpa*'larla ne yapılabileceğidir. İlk bakışta iki cevabın olduğu görülmektedir. Öncelikle, genel inancın gösterimi için *bpa*'lar toplanabilir. Bu aşağıda incelenmiştir. Daha önemlisi, farklı kaynaklardan gelen *bpa*'lar birleştirilebilir. Örneğe devam edilirse, eğer Peter'in %60 güvenilirlikle, cinayet anında bir uçakla ayrılmakta olduğu bilgisi elde edilebilirse, bu durumda $m_2(\{\text{Paul}, \text{Mary}\}) = 0{,}6$ *bpa*'sı

elde edilir. Geri kalan olasılıkla ilgili bir şey bilinmediğinden dolayı bu tüm ayırt etme yapısına atanır, $m_2(\{\text{Peter, Paul, Mary}\}) = 0{,}4$.

Geleneksel olasılıkların birleştirilmesi çarpma işlemiyle yapılır. Daha genelleştirilmiş Dempster-Shafer teorisi yaklaşımında iki delili birleştiren daha karmaşık bir çarpma kuralı gereklidir. Bu örnek için nasıl sonuç verdiği Çizelge 3.1'de görülmektedir.

A'nın m_1 atamasından M_1 kütlesine sahip olduğu ($m_1(\{A\}) = M_1$) ve B'nin m_2 atamasından M_2 kütlesine sahip olduğu ($m_2(\{B\}) = M_2$) kesişen herhangi iki küme A ve B için bunların kesişimine ait inanç M_1 ve M_2'nin çarpımıdır. Örneğin,

$$m_3(\{\text{Paul, Mary}\}) = m_1(\{\text{Peter, Paul, Mary}\}) \times m_2(\{\text{Paul, Mary}\})$$

$$m_3(\{\text{Paul, Mary}\}) = 0{,}2 \times 0{,}6$$

$$m_3(\{\text{Paul, Mary}\}) = 0{,}12$$

olmaktadır.

Çizelge 3.1: Geleneksel Olasılıkların Birleştirilmesi

	$m_1(\{\text{Peter, Paul}\}) = 0{,}8$	$m_1(\{\text{Peter, Paul, Mary}\}) = 0{,}2$
$m_2(\{\text{Paul, Mary}\}) = 0{,}6$	$m_3(\{\text{Paul}\}) = 0{,}48$	$m_3(\{\text{Paul, Mary}\}) = 0{,}12$
$m_2(\{\text{Peter, Paul, Mary}\}) = 0{,}4$	$m_3(\{\text{Peter, Paul}\}) = 0{,}32$	$m_3(\{\text{Peter, Paul, Mary}\}) = 0{,}08$

Yeni delil, olasılıkların, ayırt etme yapısının çeşitli altkümelerine daha fazla yayılmasını içermektedir. Bu delil, inanç düzeyi bulmak için kullanılabilir: herhangi bir kümedeki inanç, bu kümenin tüm alt kümelerinin olasılıklarının toplamına eşittir. Örneğin,

$$Bel(\{\text{Peter, Paul}\}) = m_3(\{\text{Peter}\}) + m_3(\{\text{Paul}\}) + m_3(\{\text{Peter, Paul}\})$$

$$Bel(\{\text{Peter, Paul}\}) = 0 + 0{,}48 + 0{,}32$$

$$Bel(\{\text{Peter, Paul}\}) = 0{,}8$$

olur. İzleyen kısımda daha biçimsel bir inceleme verilmiştir.

Yukarıdaki örnek temel kavramları tanıtmaktadır. Dempster-Shafer teorisinin terminolojisi, olasılık teorisine göre biraz farklıdır. $\Theta = \{h_1, h_2, \ldots, h_n\}$ sonlu bir hipotez kümesi (ayırt etme yapısı) olsun. $m : 2^{\Theta} \to [0,1]$ *bpa* fonksiyonu

$$m(\phi) = 0 \tag{3.1}$$

ve

$$\sum_{x \in 2^{\Theta}} m(x) = 1 \tag{3.2}$$

şeklinde tanımlanmaktadır. Burada, 2^{Θ} gösteriminin kullanılmasının sebebi, Θ'nın tüm alt kümelerinindeki, tüm elemanların dikkate alınması gerektiğindendir. Tüm atanmış olasılıklar birleşme için toplanır ve boş kümede inanç olmaz (yanlış olduğu bilinen bir hipotez olarak düşünülebilir). $m(x)$ değerinin sıfır olmadığı, Θ'nın herhangi bir alt kümesi x, bir odaksal elemandır ve x tarafından tanımlanan önermenin kesin inancını gösterir. Göz önüne alınabilecek olası önermeler, "y'nin Y'deki doğru değerleri" olmaktadır ve burada $Y \subseteq \Theta$ ve y, bazı olası değerleri Θ ayırt etme yapısını içeren bir niceliktir. Böyle önermeler alt kümelerdir. $m(Y)$, "y'nin Y'de olan, ancak Y'nin herhangi bir özalt kümesinde olmayan doğru değerlerinin" güvenilirliğini göstermektedir (eğer $Z \subset Y$ ancak $Z \neq Y$ ise Z, Y'nin bir özalt kümesidir).

bpa temel alınarak, diğer güvenilirlik ölçüleri tanımlanmıştır. Bir inanç (belief) ölçüsü, $Bel : 2^{\Theta} \to [0,1]$ bir fonksiyondur ve sorudaki olasılıkların alt kümeleri olan olasılıkların toplamından bulunur ve aşağıdaki şekilde tanımlanır:

$$Bel(A) = \sum_{B \subseteq A} m(B), \quad \text{tüm } A \subseteq \Theta \text{ için} \tag{3.3}$$

Bu, y değerlerinin A veya A'nın herhangi bir alt kümesinde olmasının inancı gösterir. Bir olabilirlik (plausibility) ölçüsü, $Pls : 2^{\Theta} \to [0,1]$ bir fonksiyondur ve aşağıdaki şekilde tanımlanır:

$$Pls(A) = \sum_{B \cap A \neq \varnothing} m(B), \quad \text{tüm } A \subseteq \Theta \text{ için} \tag{3.4}$$

Açık bir şekilde, $Pls(A)$, A'nın olabilirlik derecesini gösterir. Bu ölçüler, açık bir şekilde birbirleriyle bağlantılıdırlar. Örneğin,

$$Bel(A) = 1 - Pls(\neg A) \tag{3.5}$$

ve

$$Pls(A) = 1 - Bel(\neg A) \tag{3.6}$$

şeklindedir. Burada, $\neg A$ "A değil" (not A) anlamına gelmektedir ve $Bel(\neg A)$ da çoğu kez A'da şüphe (doubt in A) olarak adlandırılır. Diğer bazı ilişkiler,

$$Bel(A) + Bel(\neg A) \leq 1 \tag{3.7}$$

ve

$$Pls(A) + Pls(\neg A) \geq 1 \tag{3.8}$$

şeklindedir. Yukarıdaki iki eşitsizlik, Bayes yaklaşımında kullanılan daha geleneksel basit olasılık fonksiyonundan önemli bir farklılığı göstermektedir. Aynı zamanda, odaksal elemanların tek oldukları durumda, $Bel(A) = Pls(A)$ olduğundan, normal olasılık teorisini içine alan Bayes analizine geri dönülmektedir.

Diğer bir nicelik, genellik (commonality) olarak bilinir ve $Q(A)$ ile gösterilmiştir ve

$$Q(A) = \sum_{A \subseteq B} m(B) \tag{3.9}$$

şeklinde tanımlanır. A'nın genelliği, potansiyel olarak A'yı içeren süper kümelerden A'ya bağlı olan bpa'daki tüm olasılıkları toplar. Yukarıdaki üç ölçü ve bpa'nın herbiri, kendisi dışındaki ölçülerden birinin bir fonksiyonu olarak gösterilebilir.

Yukarıdaki ölçüler ortak olarak bir olay ve onun değili hakkında, belirtilmiş bir bilgisizlik ölçüsü (measure of ignorance) ile Dempster-Shafer teorisini sunarlar. Ölçü, $[Bel(A), Pls(A)]$ aralığının uzunluğu olarak verilir ($Bel(A) \leq Pls(A)$ olduğu Yager tarafından gösterilmiştir). Bu aynı zamanda A'nın "doğru olasılığı" olarak da yorumlanabilir. Θ'ya atanan kütle, delilin ağırlığı hipotezler arasında ayırt edilmediği için genel bilgisizlik olarak da yorumlanabilir. Belirsizlik ölçülerinin hesaplanması hakkında daha fazla teorik çalışma çok sayıda yazar tarafından yapılmıştır.

Daha önce de belirtildiği gibi Dempster-Shafer teorisi, Dempster'in birleştirme kuralını kullanarak farklı kaynaklardan gelen delilleri birleştirmek için bir yöntem sunar. Bu kural, kaynakların bağımsız olduklarını varsaymaktadır. Eğer m_1 ve m_2 sırasıyla Bel_1 ve Bel_2'ye ait *bpa*'lar ve Bel_1 ve Bel_2 bağımsız ise bu durumda, aşağıdaki şekilde tanımlanan $m_1 \oplus m_2 : 2^{\Theta} \rightarrow [0,1]$ fonksiyonu

$$[m_1 \oplus m_2](y) = \begin{cases} 0 & y = \varnothing \\ \dfrac{\sum_{A \cap B = y} m_1(A) m_2(B)}{1 - \sum_{A \cap B = \varnothing} m_1(A) m_2(B)} & y \neq \varnothing \end{cases} \tag{3.10}$$

bir *bpa*'dır. Dempster kuralı, temel aksiyomlardan geleneksel olasılık teoremlerinin türetilmesine benzer şekilde, tam olarak biçimsel bir gerekçe (justification) içermemektedir. Bununla birlikte gerekli değişme ve birleşme özelliklerine sahiptir. Bayes kuralı, bu kuralın özel bir durumudur. Dempster kuralının gerektirdiği bağımsız olma, olasılıklı bağımsızlıktır ve sadece olasılıkları bilinen hipotez veya küme elemanlarına uygulanır.

Yukarıdaki formülde bulunan önemli bir özellik, paydadaki $1 - \sum_{A \cap B = \varnothing} m_1(A) m_2(B)$ ifadesi k ile gösterilmek üzere, bunun, kaynaklar arasındaki çelişkinin bir ölçüsü olması ve normalizasyon faktörünün kombinasyonunda doğrudan hesaba katılmasıdır. Ölçü, eğer kütleler normalize edilmediyse, boş kümeye atanacak olan kütleyi göstermektedir. Kombinasyonun kalitesini değerlendirmek için bu çok önemlidir: bu değer yüksek olduğunda ($k \approx 1$, güçlü çelişki durumu), kombinasyon yorumlanamayabilir ve soru işaretleri kalabilir. Bu özelliğin bulunması,

kombinasyondan sonra, ancak normalizasyondan önce, kütlenin boş kümeye atanması sezgisel sonuçlara zıt olabileceğinden, bazı yazarları normalizasyon kullanımını incelemeye yöneltmiştir.

Birkaç inanç fonksiyonu birleştirilmek istendiğinde, bu basitçe ilk ikisinin birleştirilmesi, daha sonra sonucun üçüncüyle birleştirilmesi ve bu şekilde devam edilmesiyle yapılabilir. Dempster Kombinasyon kuralının daha iyi anlaşılmasına yardımcı olması amacıyla Beynon ve Morgan'ın (2000) çalışmasında verilmiş olan aşağıdaki örnek incelenebilir. Yarın öğlen Cardiff'teki hava durumunun, bugünkü hava durumundan tahmin edildiği varsayılsın. Burada sadece şu durumların söz konusu olduğu varsayılmaktadır: karlı (snowing – S), yağmurlu (raining – R) ve yağışsız (dry – D). Bu durumda ayırt etme yapısı, $\Theta = \{S, R, D\}$ şeklinde olacaktır. Aşağıdaki iki delilin elde edildiği varsayılsın:

- Bugünkü sıcaklık sıfırın altındadır (below freezing).
- Barometre basıncı düşmektedir; bir fırtına (storm) olasıdır.

Bu iki delil, Çizelge 3.2'de iki *bpa*, m_{freeze} ve m_{storm} ile gösterilmiştir.

Çizelge 3.2: Bağımsız Delillerin Temel Olasılık Atamaları

	$\varnothing$	$\{S\}$	$\{R\}$	$\{D\}$	$\{S, R\}$	$\{S, D\}$	$\{R, D\}$	$\{S, R, D\}$
m_{freeze}	0,0	0,2	0,1	0,1	0,2	0,1	0,1	0,2
m_{storm}	0,0	0,1	0,2	0,1	0,3	0,1	0,1	0,1

Tanımdan, ne m_{freeze}, ne de m_{storm}, $\varnothing$ önermesinde herhangi bir olasılık kütlesi alır. m_{freeze} fonksiyonu, kütlesini $\{S\}$, $\{S, R\}$ ve $\{S, R, D\}$ üzerinde ekstra ağırlık olmak üzere, Θ kümesinin kalan elemanlarına dağıtır. m_{storm} fonksiyonu da kütlesini Θ kümesinin boş-olmayan altkümelerine dağıtır, ancak en fazla ağırlığı $\{S, R\}$'ye verir ve $\{R\}$'ye biraz vurgu yapar. m_{both}, m_{freeze} ve m_{storm}'un birleştirilmiş delilini göstersin. Eğer m_{freeze} ve m_{storm} birbirinden bağımsız delilleri gösteriyorsa, m_{both} Dempster

kombinasyon kuralıyla verilir; $m_{\text{both}} = m_{\text{freeze}} \oplus m_{\text{storm}}$. Bunlar Çizelge 3.3'te görülmektedir.

Hesaplamalar sonucunda elde edilen değerler $\{\text{S}\}$, $\{\text{R}\}$ ve $\{\text{D}\}$ elemanlarının her birine, başlangıçtaki temel olasılık atamasının atadığından daha fazla ağırlık atamaktadır. Bununla birlikte, $\{\text{S}\}$ ve $\{\text{R}\}$'nin ikisi de $\{\text{D}\}$'nin sahip olduğundan daha fazla ağırlığa sahiptir. Θ kümesinin diğer elemanlarının ağırlığında azalma olmuştur.

$\{\text{S, R}\}$ önermesindeki birleştirilmiş delilden alınmış olan inanç derecesi (0,282 + 0,282 + 0,180 = 0,744), m_{freeze}'den alınmış olan 0,5 ve m_{storm}'dan alınmış olan 0,6'dan daha yüksektir. Doğal olarak, $\{\text{S, R, D}\}$'deki inanç derecesi değişmemiş olup, 1'e eşittir.

Çizelge 3.3: Bağımsız Delillerin Birleştirilmesi

	$\varnothing$	$\{\text{S}\}$	$\{\text{R}\}$	$\{\text{D}\}$	$\{\text{S, R}\}$	$\{\text{S, D}\}$	$\{\text{R, D}\}$	$\{\text{S, R, D}\}$
m_{both}	0,0	0,282	0,282	0,128	0,180	0,051	0,051	0,026

Bel ve *Pls* fonksiyonları eşit olduğunda, bu "Bayes" özel durumuna işaret etmektedir. Bunların aralarındaki fark bazen belirsizlik ölçüsü olarak verilir. Bununla birlikte, Bayes durumunda, 1 olasılığına sahip bir önerme dışındaki bütün durumlarda hala belirsizlik bulunduğundan, bu yeterince uygun değildir. Belki de örneğin, Bayes özel durumu yaklaşımı kapsamında, *Bel* ve *Pls* arasındaki farklılığı bir "tekillik" (singularity) ölçüsü olarak almak, bunun daha iyi anlaşılmasını sağlayabilir.

3.1.2 – Genetik Algoritmalar

3.1.2.1 – Genel Bilgiler

1960'tan beri zor eniyileme problemleri için güçlü algoritmalar geliştirmeye olan ilgi artmıştır. Bu tür yöntemler için evrimsel hesaplama terimi kullanılmaktadır. Bu sınıftaki en iyi bilinen algoritmalar, Holland tarafından geliştirilen genetik algoritmalar (genetic algorithms), Rechenberg ve Schwefel tarafından geliştirilen gelişim stratejileri (evolution strategies), Fogel ve arkadaşları tarafından geliştirilen evrimsel programlama

(evolutionary programming) ve Koza tarafından geliştirilen genetik programlama (genetic programming) algoritmalarıdır.

Güçlü ve geniş uygulama alanına sahip stokastik arama ve eniyileme yöntemi olarak genetik algoritmalar, günümüzde belki de evrimsel hesaplama yöntemlerinin en çok bilinen tipidir.

Genel olarak, genetik algoritmalar aşağıdaki beş temel bileşene sahiptirler:

- Problem çözümlerinin genetik gösterimi.
- Çözümlerin bir başlangıç popülasyonunun oluşturulması.
- Çözümleri uygunlukları bakımından değerlendiren bir değerlendirme fonksiyonu.
- Yeniden üretim boyunca, yeni nesillerin genetik niteliğini değiştiren genetik operatörler.
- Genetik algoritma parametreleri için değerler.

Genel amaçlı bir arama yöntemi olan genetik algoritmalar, olası çözümlerin kodlandığı dizilerin (birey) bir kümesi (popülasyon) ile biyolojik özelliği taklit eden operatörlerin bir kümesinden oluşur. Herhangi bir problemin çözümünde kullanılan basit bir genetik algoritma Şekil 3.1'deki algoritma ile açıklanabilir.

Basit bir genetik algoritmanın ilk aşamasında, tüm olası çözümlerin bir alt kümesi olan bir başlangıç popülasyonu ($P(0)$) elde edilir. Popülasyonun her bireyi, bir dizi olarak kodlanır. Her dizi biyolojik olarak bir kromozoma eşdeğerdir. Genetik algoritmanın herhangi bir adımındaki popülasyonu ($P(t)$), nesil (generation) olarak adlandırılır. Popülasyondaki her dizi bir uygunluk değerine (fitness value) sahiptir.

Genetik algorimanın her çevriminde; (i) seçim, (ii) genetik operatörlerin uygulanması ve (iii) uygunluk değerinin hesaplanması olmak üzere üç ana adım vardır.

Birinci adım, mevcut popülasyondan gelecek popülasyona taşınacak olan dizilerin seçilmesi işlemidir. Seçim işlemi dizilerin uygunluk değerine göre gerçekleştiğinden,

uygunluk değerleri yüksek olan dizilerin bir sonraki popülasyonda bulunma olasılıkları yüksektir.

İkinci adım, genetik operatörlerin uygulanmasıdır. Seçim işlemi ile oluşturulan yeni popülasyondaki dizilerin bir kısmına uygulanan genetik operatörler, çaprazlama ve mutasyon operatörleridir. Bu operatörler yeni popülasyon içinde yeni dizileri (olası yeni çözümleri) elde ederler. Çaprazlama operatörü farklı diziler arasında bilgi değişimini sağlayarak yeni çözümleri elde ederken, mutasyon operatörü mevcut dizilerin bir kısmında rassal değişimle çözüm uzayında yeni noktaların elde edilmesini sağlamaktadır.

Üçüncü adım da uygunluk değerinin hesaplanmasıdır. Uygunluk değeri yeni popülasyona taşınacak dizilerin belirlenmesinde kullanılan bir araçtır. Bu nedenle, algoritmanın her çevriminde popülasyondaki dizilerin uygunluk değeri hesaplanır.

```
Procedure: Genetik Algoritmalar
begin
        t ← 0 ;
        P(t) başlangıç popülasyonunu oluştur;
        P(t) 'yi değerlendir;
        while not (sonlandırma koşulu) do
                begin
                        C(t) 'yi elde etmek için P(t) 'yi yeniden oluştur;
                        C(t) 'yi değerlendir;
                        P(t) ve C(t) 'den P(t+1) 'i seç;
                        t ← t+1 ;
                end
end
```

Şekil 3.1: Genetik Algoritmaların Çalışması

Kodlama

Genetik algoritmaları diğer arama yöntemlerinden ayıran en önemli özellik, parametrelerin kendisi yerine, parametreleri temsil eden dizilerin kullanılmasıdır. Bu nedenle ilk adım, problem için arama uzayını en iyi temsil eden uygun bir kodlama yapısının seçimidir. Genetik algoritmaları kullanırken, problem çözümünün bir kromozoma nasıl kodlanacağı önemli bir konudur.

Genetik algoritmalarda kullanılan kodlama yöntemleri aşağıdaki gibi sınıflandırılabilir:

- İkili Kodlama
- Reel-Sayı Kodlama
- Tamsayı veya Literal Permütasyon Kodlama
- Genel Veri Yapısı Kodlama

Literatürde en yaygın kullanılan kodlama düzenlerinden birisi, ikili kodlamadır. Bu kodlamada her dizi 0 ve 1 değerlerinden oluşmaktadır. Dizinin uzunluğu parametrelerin alt ve üst sınırları arasındaki tüm noktaları temsil edecek şekilde belirlenmektedir. Alt ve üst sınırı sırasıyla $U_{\min}$ ve $U_{\max}$ olarak verilen bir parametre için dizi uzunluğu l, aşağıdaki eşitlikten hesaplanmaktadır:

$$\pi = \frac{U_{\max} - U_{\min}}{2^l - 1} \tag{3.11}$$

İlgilenilen parametre reel sayı ise π, ondalık kesirden sonra istenilen hassaslık değerini gösterir ve dizinin uzunluğu π değerine bağlıdır. Birden fazla parametreye sahip bir fonksiyon için dizi uzunluğu her parametre için belirlenen dizi uzunluklarının toplamına eşittir. İkili düzede bir dizi olarak kodlanan bir parametrenin gerçek değerinin hesaplanması için aşağıdaki eşitlikten yararlanılır:

$$x = U_{\min} + \left(\sum_{i=0}^{l-1} 2^i S_i \right) \frac{U_{\max} - U_{\min}}{2^l - 1} \tag{3.12}$$

Burada, S_i dizideki i. değeri göstermektedir (Goldberg 1989).

İkili düzende kodlama, çok kullanılan bir kodlama tipi olmasına rağmen, bazı sakıncaları vardır. Örneğin çok değişkenli bir fonksiyonun eniyilenmesi için değişkenlerin alt ve üst sınırlarına bağlı olarak elde edilen dizi çok uzun olabilir. Aynı zamanda gezgin satıcı, çizelgeleme, karesel atama gibi kombinatoryal eniyileme problemlerinde ikili düzende kodlama arama uzayını tam olarak temsil edememektedir.

Reel-sayı kodlama, fonksiyon eniyileme problemleri için en iyi kodlama şeklidir. Fonksiyon eniyileme veya kısıtlı eniyileme için reel-sayı kodlamanın, ikili kodlamadan daha iyi sonuç verdiği bilinmektedir. Reel-sayı kodlama için genotip uzayının topolojik yapısı fenotip uzayınınkiyle aynı olduğundan, geleneksel yöntemlerden yararlı tekniklerin alınarak etkili genetik operatörlerin kullanılması kolaydır.

Tamsayı veya literal permütasyon kodlama, kombinatoryal eniyileme problemleri için en iyi kodlama şeklidir. Kombinatoryal eniyileme problemlerinde, en iyi permütasyon veya kombinasyonun aranması söz konusu olduğundan, literal permütasyon kodlama bu tür problemler için en iyi yoldur.

Daha karmaşık problemler için genel veri yapısı kodlama kullanılabilir. Burada, problemin içeriğini yakalayabilmek amacıyla, bir genin aleli olarak uygun bir veri yapısı önerilir. Böyle durumlarda, gen karmaşık bir veri yapısı olabilir.

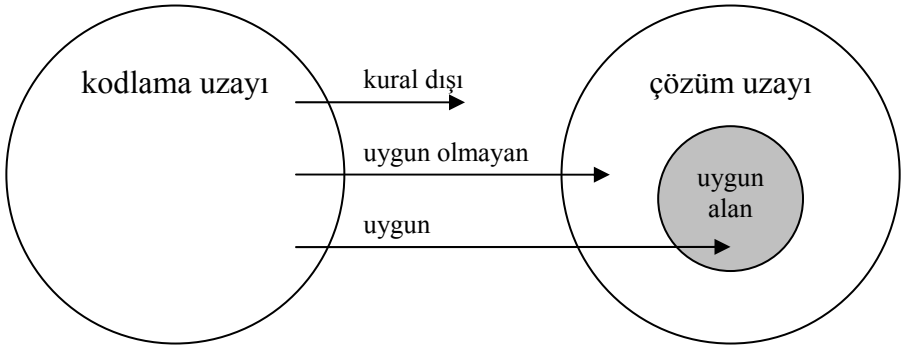
Kodlamaların yapısına göre, kodlama yöntemleri de ikiye ayrılabilir: (i) bir-boyutlu kodlama ve (ii) çok-boyutlu kodlama. Uygulamada çoğu kez bir boyutlu kodlama kullanılmaktadır. Bununla birlikte, çoğu gerçek-dünya problemi, çok boyutlu yapıları için çözümler gerektirirler. Bu çözümlerin gösterimi için çok boyutlu kodlama kullanmak doğaldır.

Kodlamanın içeriğine göre, aynı zamanda, şu kodlama yöntemleri kullanılabilir: (i) sadece çözüm ve (ii) çözüm + parametreler. Uygulamada, verilen bir problem için uygun kodlama geliştirmede, ilk yöntem daha çok kullanılmaktadır. İkinci yöntem, Rechenberg ve Schwefel stratejilerini değerlendirmek amacıyla kullanılmaktadır. Bir birey iki kısımdan oluşmaktadır: birinci kısım verilen problemin çözümü ve ikinci kısım, mutasyon için normal dağılımın varyans ve kovaryansını da kapsayan strateji parametreleri. Bu parametrelerin, bireylerin gösterimi içine dahil edilmesindeki amaç, değerlendirme operatörleri kullanarak bunların uyarlanmasını sağlamaktır. Bu durumda, artık arama, çözüm ve strateji parametreleri uzayında birlikte gerçekleştirilir. Bu şekilde, mutasyon parametrelerinin keyfi durumlar altında ayarlaması ve değişimi sağlanmaktadır.

Genetik algoritmalar iki uzayda çalışmaktadırlar: kodlama uzayı ve çözüm uzayı veya diğer bir deyişle genotip uzayı ve fenotip uzayı. Genetik operatörler, genotip

uzayında ve fenotip uzayının değerlendirilmesi ve seçiminde çalışırlar. Doğal seçim, kromozomlar ile geri kodlanmış çözümlerin performansı arasındaki bağlantıdır. Genotip uzayından fenotip uzayına haritalama, genetik algoritmanın performansında önemli bir etkiye sahiptir. Haritalama ile ilgili bir sorun, bazı bireylerin verilen problemde uygun olmayan çözümlere karşılık gelmesidir. Bu, kısıtlı ve kombinatoryal eniyileme problemleri için sorun oluşturabilir.

Uygun olmama (infeasibility) ve kural dışı olma (illegality) kavramları ayırt edilmelidir (Şekil 3.2). Bunlar literatürde çoğu kez yanlış kullanılmaktadır. Uygun olmama, bir kromozomdan geri kodlanan bir çözümün verilen bir problemin uygun çözüm alanının dışında kalmasıdır. Kural dışı olma ise bir kromozomun verilen bir probleme bir çözüm sunmamasıdır.



Şekil 3.2: Uygun Olmama ve Kural Dışı Olma

Kromozomların uygun olmaması, kısıtlı eniyileme probleminin doğasından gelmektedir. Hangi teknik kullanılırsa kullanılsın, geleneksel yöntemler veya genetik algoritmalar, kısıtları sağlamalıdır. Çoğu eniyileme problemi için uygun alan eşitlikler veya eşitsizlikler sistemi olarak gösterilebilir. Böyle durumlar için ceza yöntemleri kullanılabilir. Kısıtlanmış eniyileme problemlerinde, en iyi çözüm, doğal olarak, uygun ve uygun olmayan alanlar arasındaki sınırda oluşur. Ceza yaklaşımı, genetik aramayı, uygun ve uygun olmayan alanlar tarafından en iyiye yaklaşmaya zorlayacaktır.

Kromozomların kural dışı olması, kodlama tekniklerinin doğasından gelmektedir. Çoğu kombinatoryal eniyileme problemi için, probleme has kodlamalar kullanılmaktadır ve bu kodlamalar, basit bir-noktalı çaprazlama sonucunda çoğu kez

kural dışı bir yeni nesil oluşmasına neden olur. Kural dışı bir kromozom, bir çözüme geri kodlanamayacağından, bu durumda ceza teknikleri kullanılamamaktadır. Tamir teknikleri kural dışı bir kromozomu kurala uygun şekle dönüştürmek için kullanılmaktadır. Örneğin, ünlü PMX operatörü permütasyon gösterimi için iki-noktalı çaprazlamadır. Orvosh ve Davis, çoğu kombinatoryal eniyileme problemi için uygun olmayan veya kural dışı kromozomun tamirinin, diğer stratejilere göre nispeten daha kolay olduğunu göstermişlerdir.

Yukarıda açıklanan, uygun olmayan veya kural dışı olan çözümlerin oluşması durumunda, kullanılabilecek çeşitli yöntemler bulunmaktadır. Bunlar ilerleyen kısımlarda açıklanmıştır.

Genetik Operatörler

Genetik algoritmalarda yeniden üretim mekanizması ile arama, yüksek uygunluğa sahip bölgelere doğru yönlendirilir. Bu bölgelerdeki yeni çözüm noktalarına ulaşmada ise genetik operatörler kullanılır. Genetik operatörler, popülasyonun genetik bilgilerini kullanarak yeni çözümler elde ederler. Genetik algoritmalarda kullanılan en genel iki genetik operatör; çaprazlama ve mutasyon operatörleridir.

Çaprazlama, farklı çözümler arasında bilgi değişimini sağlayarak, arama uzayının benzer, ancak araştırılmamış bölgelerine ulaşmayı sağlayan bir arama operatörüdür. Bir popülasyona çaprazlama operatörü, p_c olasılığı ile uygulanır. İkili düzen kodlama için literatürde bilinen ve çok sık kullanılan üç çaprazlama operatörü vardır. Bunlar; bir noktalı, iki noktalı ve uniform çaprazlama operatörleridir. Bir noktalı çaprazlama operatöründe, çaprazlama noktası 1 ile $l-1$ (dizi uzunluğunun bir eksiği) arasından rassal olarak seçilir. Eşleşen iki dizide, bu çaprazlama noktasından sonraki bölümler yer değiştirerek yeni iki dizi elde edilir. Örneğin, bir popülasyondan A1 ve A2 gibi iki dizi seçilsin ve dizi uzunlukları 12 ($l = 12$) olsun. 1 ile 11 arasında rassal olarak seçilen çaprazlama noktasının değeri 7 ise bu çaprazlama noktasına göre A1' ve A2' gibi iki yeni dizi Şekil 3.3'teki gibi elde edilir. Burada | işareti çaprazlama noktasını göstermektedir.

Literatürde bu üç yöntem ve çok noktalı (çaprazlama noktası ikiden büyük olan) çaprazlama operatörünün etkinliği Syswerda (1989), Spears ve DeJong (1991) ile DeJong ve Spears (1992) tarafından araştırılmıştır. Sysweda (1989), yaptığı çalışmada uniform çaprazlamanın diğer iki çaprazlama operatörüne göre daha etkin olduğunu göstermiştir. Spears ve DeJong (1991) ile Dejong ve Spears (1992) çalışmalarında, 2 noktalı çaprazlama operatörünün, her zaman, bir noktalı çaprazlama operatöründen daha etkin olduğu sonucuna varmışlardır (Dengiz, Altıparmak, 1998).

	1	2	3	4	5	6	7		8	9	10	11	12
A1:	0	1	0	0	1	1	1	|	0	1	0	1	0
A2:	1	0	0	1	1	1	0	|	0	0	1	0	0
A1':	0	1	0	0	1	1	1		0	0	1	0	0
A2':	1	0	0	1	1	1	0		0	1	0	1	0

Şekil 3.3: Çaprazlama

Mutasyon, genetik algoritmanın çalışmasında ikinci derecede önemli rol oynar. Genetik algoritmada mutasyon operatörü, küçük bir olasılıkla bir dizi içindeki bir veya birkaç değeri rassal olarak değiştirerek, popülasyonda yeni dizilerin (yani, arama uzayında yeni çözüm noktalarının) elde edilmesini sağlar (Goldberg, 1989). Bir popülasyona mutasyon operatörü, p_m olasılığıyla uygulanır. İkili düzende kodlamanın kullanıldığı bir dizide, mutasyon operatörü ile rassal olarak seçilen elemanın değeri 1 ise 0, 0 ise 1 olarak değiştirilerek yeni bir dizi elde edilir. Şekil 3.4'te verilen örnek, mutasyon operatörünün bir diziye uygulanışını göstermektedir. Görüldüğü gibi A1 dizisinin 2 ve 10. elemanları mutasyona uğratılarak A1' dizisi elde edilmiştir.

	1	2	3	4	5	6	7	8	9	10	11	12
A1:	0	1	0	0	1	1	1	0	1	0	1	0
		↓								↓		
A1':	0	0	0	0	1	1	1	0	0	1	0	0

Şekil 3.4: Mutasyon

Özellikle genetik algoritmanın son çevrimlerinde mutasyonun etkinliği artmaktadır. Çünkü son çevrimlerde popülasyon iyi çözümlere yakınsadığından diziler birbirlerine çok benzemektedir. Bu durum, çaprazlama operatörünün farklı yeni diziler oluşturmasını engelleyerek aramayı kısıtlar. Bu aşamada mutasyon, popülasyondaki değişkenliği gerçekleştirerek arama uzayında yeni çözüm noktalarının elde edilmesini sağlamaktadır.

Seçim

Başlangıç popülasyonu oluşturulduktan sonra, algoritmanın her çevriminde, yeni popülasyonun dizileri, bir olasılıklı seçim süreci ile mevcut popülasyonun dizileri arasından seçilir. Yüksek uygunluk değerine sahip olan diziler, yeni dizilerin elde edilmesinde yüksek olasılığa sahiptirler. Seçim yöntemi, doğal seçimi, yapay olarak gerçekleştirmektedir. Doğal popülasyonların uygunluğu, bireyin büyümesi ve çoğalmasında engellere karşı koyma yeteneği ile belirlenir. Amaç fonksiyonun, bir dizinin yaşamasında veya elenmesinde son karar verici olarak kullanımı ile "doğal seçim" yapay olarak gerçekleştirilir. Goldberg ve Deb (1991), literatürde mevcut ve çok sık kullanılan seçim yöntemlerini; (i) orantılı yeniden üretim mekanizması (proportionate reproduction), (ii) sıralı yeniden üretim mekanizması (ranking reproduction), (iii) turnuva yeniden üretim mekanizması (tournament reproduction) ve (iv) denge durumu yeniden üretim mekanizması (steady state reproduction veya Genitor) olmak üzere dört ana sınıfta toplamışlardır. Literatürde, orantılı, sıralı ve turnuva yeniden üretim mekanizmaları ile birlikte elitist stratejisinin de çok sık kullanıldığı görülmektedir. Elitist stratejisi ile mevcut popülasyondaki en iyi bir veya birkaç dizi bir sonraki popülasyona taşınır. Amaç, elde edilen en iyi uygunluk değerine sahip dizinin veya dizilerin, örnekleme hatası veya genetik operatörlerin kullanımı sonucunda kaybolmasını önlemektir (Dengiz, Altıparmak, 1998).

Orantılı yeniden üretim mekanizmasında, popülasyondaki her bireyin seçilme olasılıkları belirlenir ve bu olasılıklar kullanılarak bir sonraki popülasyon oluşturulur. Bu grupta bulunan yeniden üretim yöntemleri; rulet tekerleği, stokastik artan ve stokastik evrensel yöntemidir.

Sıralı yeniden üretim mekanizmasında, popülasyondaki bireyler uygunluk değerlerine göre küçükten büyüğe doğru sıralanır. En iyiden başlayarak bir azalan fonksiyon yardımı ile dizilere kopya sayısı atanır ve orantılı yeniden üretim mekanizmalarından birisi kullanılarak yeni popülasyon elde edilir.

Turnuva yeniden üretim mekanizmasında, popülasyondan rassal olarak bir grup dizi (yerine koyarak / yerine koymadan) seçilir ve grup içindeki eniyi uygunluk

değerine sahip olan dizi yeni popülasyona kopyalanır. Bu işlem, popülasyon büyüklüğüne ulaşılıncaya kadar devam eder. Grup, en az iki diziden oluşur.

Denge durumu yeniden üretim mekanizmasında, sıralı yeniden üretim mekanizması kullanılarak seçilen bir veya iki bireye genetik operatörler uygulanır. Elde edilen yeni diziler mevcut popülasyondaki uygunluk değeri en küçük diziler ile yer değiştirerek yeni popülasyon oluşturulur.

Karma Genetik Algoritmalar

Genetik algoritmalar olası çözümlerin elde edilmesinde sadece kodlamayı ve amaç fonksiyonu değerini kullanmaktadırlar. Probleme özgü bilgiyi kullanmaması, genetik algoritmaların uygulama alanının geniş olmasını sağlamaktadır. Ancak, bir probleme ait mevcut bilginin kullanılmaması, genetik algoritmaların bu problem için geliştirilmiş sezgiseller kadar iyi olmasını engellemektedir. Bu nedenle, genetik algoritmalar ile probleme özgü bilginin birleştirilmesi için literatürde çeşitli yöntemler önerilmiş ve elde edilen algoritma karma (hybrid) genetik algoritma olarak adlandırılmıştır. Karma genetik algoritmaların çalışması şekli Şekil 3.5'te özetlenmiştir. Genetik algoritmaların genel yapısı içinde probleme özgü bilginin kullanılabileceği çeşitli aşamalar vardır. Bu aşamalar kısaca aşağıda açıklanmaktadır.

```
Procedure: Karma Genetik Algoritmalar
begin
        t ← 0;
        P(t) başlangıç popülasyonunu oluştur;
        P(t) 'yi değerlendir;
        while not (bitiş koşulu) do
                begin
                        C(t) 'yi elde etmek için P(t) 'yi yeniden oluştur;
                        C(t) 'yi yerel olarak eniyile;
                        C(t) 'yi değerlendir;
                        P(t) ve C(t) 'den P(t+1) 'i seç;
                        t ← t+1;
                end
end
```

Şekil 3.5: Karma Genetik Algoritmaların Çalışması

Çözümü aranan problem için geliştirilmiş sezgisel algoritmalar varsa, bu algoritmaların genetik algoritmalar ile birlikte kullanılması mümkündür. Bu tür algoritmalar bir genetik algoritma içinde üç farklı şekilde kullanılabilirler. Birincisi, uygun başlangıç popülasyonun oluşturulmasında sezgisel algoritmanın ürettiği çözüm veya çözümlerin kullanılmasıdır. Bu yöntem ile bir karma genetik algoritmanın sezgisel ile elde edilen çözümden daha kötü çözüm bulması engellenir. İkincisi, karma genetik algoritmanın operatör kümesinde sezgisel algoritma kullanılmasıdır. Sezgisel algoritma, kodlamada ardışık dönüşümleri gerçekleştirebiliyorsa, genetik operatörlerden sonra popülasyondaki her diziye uygulanabilir. Bu işlem ile aramanın daha umut verici bölgelere yönlendirilmesi sağlanır (Davis 1991). Üçüncüsü ise, yerel arama yönteminin genetik algoritmadan sonra kullanılmasıdır. Bu yöntemde, aramanın yüksek uygunluğa sahip bölgeleri bir genetik algoritma ile belirlenir. Son çözümde elde edilen yüksek uygunluğa sahip çözümlerin %5 veya %10'unda yerel arama yöntemi ile aramaya devam edilerek daha iyi çözümlere ulaşılır (Dengiz, Altıparmak, 1998).

Probleme özgü bilginin genetik operatörlerde de kullanılması mümkündür. Bu durumda birçok problem için yeni genetik operatör tanımlanabilir. Bu operatörler mevcut dizlerden yeni dizilerin oluşturulmasında da probleme özgü bilgiyi dikkate alırlar.

Literatürde karma genetik algoritmaların kullanıldığı çeşitli çalışmalar vardır. Bu çalışmaların bir kısmında mevcut sezgisellerden yararlanılırken, diğer bir kısmında özel çaprazlama operatörleri geliştirilmiş ve mutasyon operatörü olarak yerel arama algoritmaları kullanılmıştır. Grefenstette (1987), probleme özgü bilginin kullanıldığı genetik algoritmanın performansını gezgin satıcı problemi üzerinde incelemiştir. Jog ve arkadaşları (1989) da gezgin satıcı probleminde mutasyon operatörü olarak literatürde mevcut 2-opt ve or-opt sezgisellerinden yararlanmış ve bir sezgisel çaprazlama operatörü geliştirmişlerdir. Homaifar ve arkadaşları (1993) ise gezgin satıcı problemi için ikili düzende matris gösterimini kullanmışlar ve matris çaprazlama operatörünü geliştirmişlerdir. Çalışmalarında her çevrimdeki popülasyonun dizilerine 2-opt sezgiselini uygulamışlardır. Thangiah ve arkadaşları (1993), genetik algoritmaları araç yönlendirme probleminde kullanmışlardır. Çalışmalarında, bir yerel arama yöntemini, genetik algoritmanın son çevriminde elde edilen çözümlere uygulamışlardır. Kapsalis ve arkadaşları (1993), steiner ağaç probleminde, başlangıç popülasyonunun oluşturulması

için minimum ağaç yaklaşımını kullanmışlar ve bu yaklaşımın algoritmanın yakınsamasındaki etkilerini incelemişlerdir. Denzig ve arkadaşları(1997b), haberleşme şebekelerinin tasarımı için geliştirdikleri genetik algoritmada, mutasyon operatörü olarak ekleme-çıkarma sezgiselini kullanmışlardır (Dengiz, Altıparmak, 1998).

3.1.2.2 – Genetik Algoritmaların Uyarlanması

Genetik algoritmalarda evrim fikrinden esinlenildiği için uyarlamanın sadece verilen bir problemin çözümlerini bulmada değil, aynı zamanda genetik algoritmaların belirli bir probleme uyarlanmasında da beklenmesi doğaldır. Geçmiş yıllarda, genetik algoritmaların gerçek dünya problemlerine etkili olarak uygulanmasını sağlamak amacıyla çok sayıda uyarlama tekniği önerilmiş ve geliştirilmiştir. Genel olarak, (i) problemlere uyarlama ve (ii) evrimsel sürece uyarlama olmak üzere iki tip uyarlama vardır.

Bunların arasındaki fark, ilkinin, verilen problem için algoritmanın uygun şeklini seçebilmek amacıyla gösterim, çaprazlama, mutasyon ve seçim gibi bazı genetik algoritma bileşenlerinin değiştirilmesini ifade etmesidir. İkincisi, problemi çözerken genetik algoritmaların konfigürasyonunu değiştiren parametrelerin ayarlanması için bir yol önermektedir. Herrera ve Lozano'ya göre yeni şekliyle uyarlama aşağıdaki sınıflara ayrılabilir (Gen, Cheng, 1999):

- Uyarlanır Parametre Ayarlaması
- Uyarlanır Genetik Operatörler
- Uyarlanır Seçim
- Uyarlanır Gösterim
- Uyarlanır Uygunluk Fonksiyonu

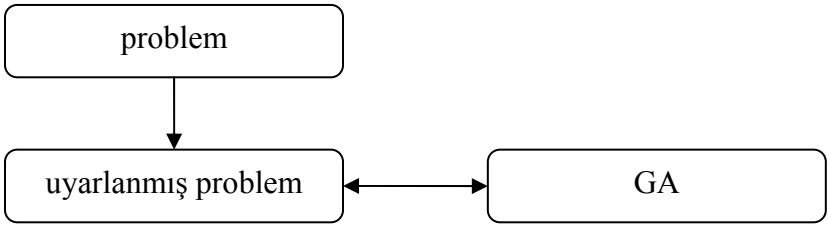
Bu sınıflar arasında, geçmiş yıllarda parametre uyarlaması hakkında kapsamlı çalışmalar yapılmıştır, çünkü mutasyon oranı, çaprazlama oranı ve popülasyon büyüklüğü gibi strateji parametreleri anahtar parametrelerdir. Bu strateji

parametrelerinin genetik algoritmalar üzerinde önemli bir etkiye sahip olduğu uzun süreden beri bilinmektedir.

Yapı Uyarlaması

Genetik algoritmalar, ikili kodlama ve ikili genetik operatörler içeren bir yöntem olarak oluşturulmuştur. Bu yaklaşım, orijinal problemin genetik algoritmalar için uygun bir forma dönüştürülmesini gerektirmektedir (Şekil 3.6). Yaklaşım, potansiyel çözümler ve ikili gösterim arasında haritalama, tamir prosedürleri vs. içermektedir. Karmaşık problemler için böyle bir yaklaşım genellikle başarılı uygulamalar sağlayamaz.

Bu yüzden, belirli problemler için genetik algoritmaların standart olmayan çeşitli gösterimleri oluşturulmuştur. Şekil 3.7'de görüldüğü gibi bu yaklaşım genetik problemde bir değişiklik yapmaz ve potansiyel çözümde bir kromozomun gösteriminin değiştirilmesiyle genetik algoritmalar üzerinde uyarlama yapar.

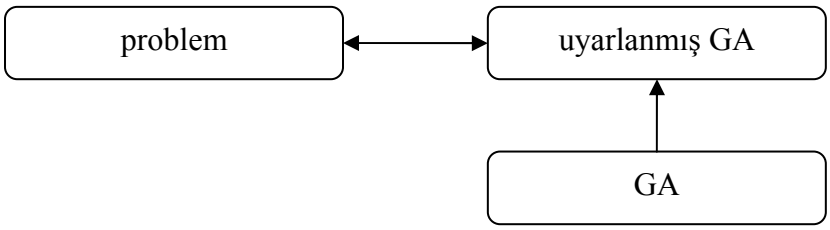


Şekil 3.6: Problemin Genetik Algoritmalara Uyarlanması

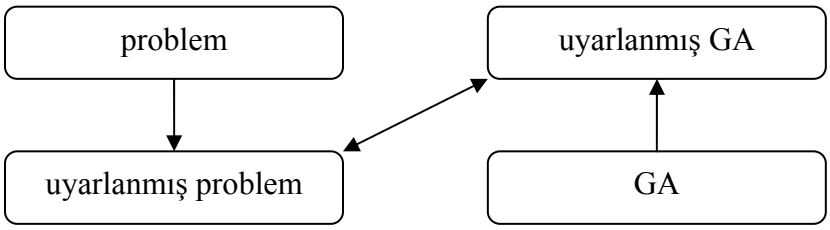
Diğer bir yaklaşım da Şekil 3.8'de görüldüğü gibi hem genetik algoritmaların hem de problemin uyarlanmasıdır. Kombinatoryal eniyileme problemlerinin genel özelliği kısıtlarla ilgili olarak birimlerin bir permütasyon ve / veya kombinasyonunun bulunmasıdır. Eğer permütasyon ve /veya kombinasyon belirlenebilirse, probleme-özgü bir yöntemle çözüm elde edilebilir. Bu yaklaşımda, genetik algoritmalar uygun bir permütasyon ve / veya kombinasyon elde etmek için kullanılır ve permütasyon ve kombinasyona göre çözüm bulmak için de bir sezgisel yöntem kullanılır. Bu yaklaşım endüstri mühendisliği alanında başarıyla uygulanmıştır ve genetik algoritmaların uygulamada kullanımı için ana yaklaşım olmuştur.

Parametre Uyarlaması

Genetik algoritmaların davranışı, arama uzayının tamamen kapsanması ve taranması arasındaki dengeyle tanımlanabilir. Bu denge, popülasyon büyüklüğü, en büyük nesil sayısı, çaprazlama oranı, ve mutasyon oranı gibi strateji parametreleri ile sağlanmaktadır. Her parametre için bir değerin nasıl seçileceği ve etkili değerlerin nasıl bulunacağı, genetik algoritmalarla ilgili çalışmalarda çok önemli ve ümit verici alanlardır. Herrera ve Lozano ve Hinterding, Michalewicz ve Eiben uyarlama konusunda güncel çalışmalar yapmışlardır.



Şekil 3.7: Genetik Algoritmaların Probleme Uyarlanması



Şekil 3.8: Genetik Algoritmaların ve Problemin Uyarlanması

Genetik algoritma uygulamalarının çoğunda, sabit parametreler kullanılmaktadır. Parametre değerleri ayarla-ve-test et yaklaşımıyla belirlenmektedir. Genetik algoritma dinamik ve uyarlanır bir süreç olduğundan, sabit parametrelerin kullanımı bir zıtlık oluşturmaktadır. Bu yüzden, algoritmanın çalışması sırasında strateji parametrelerinin değiştirilmeye çalışılması doğaldır. Bunu çeşitli şekillerde yapmak mümkündür: (i) bir kural kullanarak, (ii) aramanın mevcut durumundan geri besleme bilgisi çıkararak veya (iii) kendinden-uyarlanır bir mekanizma kullanarak. Hinterding, Michalewicz ve Eiben'in uyarlamayı sınıflandırmasına göre üç kategori bulunmaktadır: (i) deterministik, (ii) uyarlanır ve (iii) kendinden-uyarlanır.

Deterministik uyarlamada, bir strateji parametresinin değeri deterministik bir kuralla değiştirilir. Genel olarak, nesil sayısıyla ölçülen, zamanla değişen bir yaklaşım kullanılır. Uyarlanır uyarlama, strateji parametresindeki değişimin yönünü belirlemek için kullanılan, evrim sürecinden elde edilen bir geri beslemenin olması durumunda kullanılır. Kendinden-uyarlanır uyarlama, strateji parametrelerinin evrim süreci içinde değişmesine olanak sağlar. Parametreler, bireylerin kromozomlarına kodlanır ve genetik operasyonlara uğrarlar. Kodlanan parametreler, bireylerin uygunluklarına doğrudan etki etmezler, ancak daha iyi değerler, yaşamaya ve yeni nesil üretmeye daha yakın olan, böylece daha iyi parametre değerlerini yeni nesle geçiren, daha iyi bireylere götürecektir.

Bulanık Mantık Denetleyicisi

1980'lerde, bulanık küme teorisinin uygulanması araştırmalarında en çok kullanılan ve verimli alanlardan birisi bulanık kontrol olmuştur. Bu araştırma hakkında kapsamlı bir inceleme Lee tarafından yapılmıştır. Bulanık mantık, insan düşünmesine ve doğal diline geleneksel mantık sistemlerinden daha yakındır. Temel olarak, gerçek dünyanın yaklaşık, kesin olmayan doğasını anlamada etkili olmaktadır. Bu açıdan bakıldığında, bulanık mantık denetleyicisinin gerekli kısmı, bulanık akıl yürütme ve sonuç çıkarma kuralının derlenmesi gibi iki kavramın bir kontrol kümesidir. Gerçekte, bulanık mantık denetleyicisi, uzman bilgisine dayanan kontrol stratejisini, otomatik kontrol stratejisine dönüştüren bir algoritma sağlamaktadır. Bu yöntem, süreçler geleneksel yöntemlerle incelenmek için çok karmaşık olduğunda veya yeterli bilgi kaynağı bulunmadığında çok yararlı olmaktadır. Bu yüzden, bulanık mantık denetimi matematiksel kontrol ve insan gibi karar verme arasında bir adım olarak görülebilir.

Bulanık mantık tekniğinin, genetik algoritma strateji parametrelerinin dinamik olarak ayarlanması amacıyla kullanılmasına öncülük eden çalışmalar Lee ve Takagi ve Xu ve Vukovich'in çalışmalarıdır. Buradaki temel fikir, genetik algoritmalar tarafından kullanılacak olan strateji parametresinin hesaplanmasında bulanık mantık denetleyicisinin kullanılmasıdır. Denetleyicinin girişleri, performans ölçülerinin herhangi bir kombinasyonu ve genetik algoritmaların mevcut parametreleridir. Çıktılar parametreler olmaktadır.

3.1.2.3 – Genetik Eniyilemeler

Eniyileme birkaç değişkene sahip bir fonksiyonu, eşitlik ve / veya eşitsizlik kısıtları altında enbüyükleme veya enküçükleme problemleriyle ilgilenir. Eniyileme, yöneylem araştırması, yönetim bilimi ve mühendislik tasarımında önemli bir rol oynar. Çoğu endüstri mühendisliği tasarım problemi çok karmaşık olup, geleneksel eniyileme yöntemleriyle çözülmesi zordur. Son yıllarda, yeni bir eniyileme tekniği olarak genetik algoritmalar önem kazanmıştır. Basitlik, operasyon kolaylığı, düşük gereksinimler ve paralel ve genel perspektif gibi nedenlerden dolayı çok geniş bir kullanım alanı bulmuştur. Bu kısımda, genetik eniyileme tekniklerine kısa bir giriş yapılmıştır Gen, Cheng, 1999).

Genel Eniyileme

Genel eniyileme, ilgilenilen bölgedeki genel en iyi ve çok sayıdaki yerel en iyi arasındaki ayrımı yapabilen teknikleri kullanır. Genel eniyileme problemleri, genellikle kısıtlı olmayan eniyileme şeklindedirler. Bu, bir fonksiyonu, sınırlama olmaksızın enbüyükleme veya enküçükleme problemidir. Genel olarak kısıtlanmamış bir eniyileme problemi matematiksel olarak şu şekilde ifade edilebilir:

$$\min f(x) \tag{3.13}$$

$$x \in \Omega \tag{3.14}$$

Burada, f reel-değerli bir fonksiyon ve uygun küme Ω, E^n kümesinin bir alt kümesidir. $\Omega = E^n$ olduğu durumda, bu tamamen kısıtlı olmayan duruma karşılık gelmektedir. Birçok uygulamada, Ω kümesinin, E^n kümesinin belirli bir alt kümesi olduğu durumun incelenmesine ihtiyaç olmaktadır. Eğer x^*'ın ε komşuluğundaki bütün $x \in \Omega$ için $f(x) \geq f(x^*)$ olacak şekilde bir $\varepsilon > 0$ varsa, $x^* \in \Omega$ gibi bir noktanın, Ω üzerinde bir yerel en küçük olduğu söylenir. Eğer bütün $x \in \Omega$ için $f(x) \geq f(x^*)$ olacak şekilde bir $\varepsilon > 0$ varsa, $x^* \in \Omega$ gibi bir noktanın, Ω üzerinde bir genel en küçük olduğu söylenir. Uygulamadaki çoğu eniyileme problemi, sağlanması

gereken kısıtlar içerse de kısıtlı olmayan eniyileme ileriki çalışmalar için bir temel oluşturmaktadır.

Geleneksel genel eniyileme yöntemleri iki sınıfa ayrılabilir: (i) deterministik yöntemler ve (ii) stokastik yöntemler. Genetik algoritmalar, geleneksel yöntemlerle çözülmesi çok zor olan problemleri çözmede oldukça başarılıdır. Bu tür problemlere örnek olarak, çoklu-model (multimodal), türevlenemeyen ve kesikli problemler verilebilir. Genetik algoritmaların 1970'lerde ortaya çıkmasından beri, genel eniyileme ana hedef olmuştur ve genel eniyileme problemleri için güçlü algoritmalar geliştirmek amacıyla çok sayıda çalışma yapılmıştır.

Genetik algoritmaları, genel eniyileme problemlerini çözmek için kullanmanın normal yolu, her bir karar değişkenini standart ikili kodlama veya Gray kodlama kullanarak kodlamaktır. n değişkene sahip bir problem için $x = (x_1, x_2, \ldots, x_n)$, bir kromozom aşağıdaki şekilde, n parçalı bir bit dizisi içerir:

$$\underbrace{1001\ldots01}_{x_1} \quad \underbrace{1001\ldots01}_{x_2} \quad \cdots \quad \underbrace{1001\ldots01}_{x_n}$$

Böylece, x_i değişkeni, l_i bitle kodlanırsa, bütün bir kromozom $\sum_{i=1}^{n} l_i$ uzunluğunda olacaktır. Genetik algoritmalardaki ikili gösterim tercihi, genetik algoritmaları, beklenen şema örnek davranışları cinsinden incelemeye çalışan, şema teorisinden gelmektedir. Bununla birlikte, ikili kodlamanın önemli dezavantajları olduğu bilinmektedir. Bu kodlama, amaç fonksiyonuna çok-şekillilik eklemekte ve böylece orijinal halinden daha karmaşık hale getirebilmektedir. Bu yüzden, fonksiyon eniyileme problemlerini çözmede çok etkili olan, reel-kodlu genetik algoritma diye de adlandırılan, reel-sayı gösterimi benimsenmektedir.

Reel-sayı gösteriminde, her bir kromozom bir reel sayı vektörü olarak kodlanır. n değişkenli bir problem için reel-sayı vektörü $x = (x_1, x_2, \ldots, x_n)$ şeklindedir. Geçmiş yıllarda, reel-sayı kodlama için dört sınıfa ayrılabilen çeşitli genetik operatörler önerilmiştir:

- Geleneksel Operatörler
- Aritmetik Operatörler

- Yön-Bazlı Operatörler
- Stokastik Operatörler

Kısıtlı Eniyilemeler

Kısıtlı eniyileme, eşitlik ve / veya eşitsizlik kısıtları altında, bir amaç fonksiyonunu eniyileme problemiyle ilgilidir ve mühendislik, yöneylem araştırması ve matematiğin her alanında son derece önemli bir araçtır. Genel kısıtlı eniyileme problemi aşağıdaki şekilde yazılabilir:

$$\max f(x) \tag{3.15}$$

$$g_i(x) \le 0\,,\ i = 1,2,\ldots,m_1 \tag{3.16}$$

$$h_i(x) = 0\,,\ i = m_1 + 1,\ldots,m\ (= m_1 + m_2) \tag{3.17}$$

$$x \in X \tag{3.18}$$

Burada, f, $g_1, g_2, \ldots, g_{m_1}$, $h_{m_1}, h_{m_2}, \ldots, h_m$, E_n üzerinde tanımlı reel-değerli fonksiyonlar, X, E_n'in bir alt kümesi ve x, elemanları $x_1, x_2, \ldots, x_n$ olan n-boyutlu bir reel vektördür. Yukarıdaki problem, kısıtları sağlayan ve f fonksiyonunu enküçükleyen $x_1, x_2, \ldots, x_n$ değişkenlerinin değerleri için çözülmelidir. f fonksiyonu genellikle amaç fonksiyonu veya ölçüt fonksiyonu olarak adlandırılır. $g_i(x) \le 0$ kısıtlarının her biri eşitsizlik kısıtı ve $h_i(x)$ fonksiyonlarının her biri de eşitlik kısıtı olarak adlandırılır. Değişkenler üzerindeki alt ve üst sınırları içerebilecek olan X kümesi genellikle ana kısıt (domain constraint) olarak adlandırılır. Tüm kısıtları sağlayan bir $x \in X$ vektörü probleme bir uygun çözüm olarak adlandırılır. Bu şekildeki çözümlerin birleşimi uygun alanı oluşturur. Kısıtlı eniyileme problemi, her bir uygun nokta x için $f(x) \le f(\bar{x})$ olacak şekilde bir $\bar{x}$ uygun noktasının bulunmasıdır. Bu şekildeki bir nokta eniyi çözüm olarak adlandırılır. Genel eniyilemenin tersine, burada, geleneksel çözüm yöntemleri çok karmaşık, ama çok etkin değildir. Son yıllarda,

genetik algoritmaları kısıtlı eniyileme problemlerine uygulamak için çalışmalar artarak sürmektedir.

Kombinatoryal Eniyileme

Kombinatoryal eniyileme problemleri, sınırlı sayıdaki uygun çözüm ile tanımlanır. Prensip olarak, böyle sınırlı bir probleme çözüm bulmak kolay gibi görünse de uygulamada problemlerin aşırı derecede büyük boyutlu olmasından dolayı, bu imkansız olabilmektedir. Bu tür problemleri çözmek için genellikle sezgisel arama yapmak düşünülmektedir. Geçmiş yıllarda, genetik algoritmalar ile bu tür problemleri çözme eğiliminde artış olmuş ve genetik algoritmaların kullanılmasından ümit verici sonuçlar alınmıştır.

Kombinatoryal eniyileme farklı özelliklerde problemler içermektedir. Bu problemler, birbirlerinden farklı olsalar da problem gereksinimi aşağıdaki tiplerden biri ile tanımlanabilir:

- Bazı birimlerin permütasyonunun belirlenmesi.
- Bazı birimlerin kombinasyonunun belirlenmesi.
- Bazı birimlerin hem permütasyon hem de kombinasyonunun belirlenmesi.
- Bazı kısıtlar için yukarıdakilerden herhangi biri.

Örneğin, kaynak-kısıtlı proje çizelgeleme problemleri ve araç rotalama ve çizelgeleme problemleri için bazı kısıtlar altında, birimlerin permütasyonu, küme-kapsama problemleri ve gruplama problemleri için birimlerin kombinasyonu ve paralel makine çizelgeleme problemi için belirli kısıtlar altında, birimlerin hem permütasyonu hem de kombinasyonu gerekmektedir.

Problemlerin genel özelliği, permütasyon ve / veya kombinasyon belirlenebilirse, çözümün, probleme-özgü (problem-spesific) bir yöntemle kolayca elde edilebilmesidir. Bu yüzden, genetik algoritmaların, bu problemlere uygulanmasındaki yaklaşım aşağıdaki şekildedir:

- Varsayımlar altında uygun bir permütasyon ve / veya kombinasyon bulmak için genetik algoritmaları kullan.
- Daha sonra permütasyon ve kombinasyona göre bir çözüm bulmak için sezgisel bir yöntem kullan.

Genetik algoritmaların kombinatoryal eniyileme problemlerine uygulanmasında anahtar adım, problemin çözümünün kromozoma nasıl kodlanacağıdır. Kombinatoryal eniyileme problemlerinde, karmaşık kısıtların bulunmasından dolayı, uygun olmayan hatta kural dışı çözümler vereceğinden basit bir ikili dizi burada uygun değildir. Bunun için sipariş-bazlı genetik algoritmalar adındaki, yeni ve etkin kodlama yöntemi geliştirilmiştir.

Çok-Amaçlı Eniyileme

1960'lardan beri çok-amaçlı eniyileme problemleri üzerinde çalışmalar yapılmaktadır. Bir çok-amaçlı eniyileme probleminde, birden fazla sayıda amaç fonksiyonunun eşzamanlı olarak eniyilenmesi gerekmektedir. Birden fazla amaç durumunda, amaçlar arasındaki karşılaştırılamazlık ve çatışmadan dolayı tüm amaçlar için en iyi çözümün bulunmasına gerek yoktur. Bir amaç için en iyi olan çözüm diğeri için en kötü olabilir. Bu yüzden, çok-amaçlı durum için genellikle basit olarak birbirleriyle karşılaştırılamayan bir çözüm kümesi bulunmaktadır. Böyle çözümler için, herhangi bir amaç fonksiyonunda iyileşme sağlamak, en az bir diğer amaç fonksiyonunda kötüleşme olmaksızın mümkün değildir.

Son yıllarda, evrimsel çok-amaçlı eniyileme veya genetik çok-amaçlı eniyileme olarak bilinen, çok-amaçlı eniyileme problemlerinin çözümü için genetik algoritmaların uygulanmasında büyük bir artış olmuştur. Burada genetik algoritmaların özelliği, nesilden nesile sürdürülen potansiyel çözümlerin bir popülasyonunda çok yönde ve genel aramadır. Popülasyondan-popülasyona yaklaşımı Pareto çözümlerinin aranmasında yararlıdır.

Çok-amaçlı eniyileme probleminin genetik algoritmalar kullanılarak çözümünde ortaya çıkan özel sorunlardan biri, birden çok amaca göre uygunluk değerinin nasıl

belirleneceğidir. Geçmiş yıllarda uygunluk atama mekanizmaları konusunda kapsamlı çalışmalar yapılmış ve bazı yöntemler önerilmiş ve test edilmiştir.

Uygun Olmayan Çözümlerin Uygun Hale Getirilmesi

Genetik algoritmaların, kısıtlı eniyileme problemlerine uygulanmasındaki en önemli zorluk; klasik genetik operatörler ile uygun olmayan (kısıtları sağlamayan) çözümlerine elde edilmesidir. Bu tür problemlerde uygun çözümlerin bulunduğu başlangıç popülasyonu elde edilse bile, çaprazlama ve mutasyon operatörleri sonucu uygun yeni çözümlerin elde edilmesi çok kolay değildir.

Gezgin satıcı, şebeke topolojisi tasarımı, çizelgeleme, montaj hattı dengeleme, sırt-çantası, tesis yerleşimi gibi çeşitli kombinatoryal eniyileme problemlerinde bu tür bir sorunla karşılaşılmaktadır. Literatürde bu sorunun ortadan kaldırılması için kullanılan çeşitli yaklaşımlar vardır. Bu yaklaşımlar, (i) atma yöntemi, (ii) ceza fonksiyonu yöntemi, (iii) düzenleyici algoritma veya tamir yöntemi ve (iv) yeni genetik operatörler yöntemi olmak üzere dört grupta incelenmektedir.

Atma yöntemi, değerlendirme sürecinde tüm uygun olmayan kromozomları atmaktadır. Bu yöntem, en kolay ama en az etkin yöntemdir.

Ceza fonksiyonu yönteminde, kısıtlar dikkate alınmaksızın olası çözümler elde edilir. Kısıtları bozan diziler uygunluk değerleri düşürülerek cezalandırılır. Diğer bir deyişle, kısıtlı eniyileme problemi, kısıt bozulmalarının cezalandırıldığı kısıtlı olmayan eniyileme problemine dönüştürülür. Cezalar değerlendirme fonksiyonu içinde yer alır.

Ceza fonksiyonu yaklaşımının kısıt sayısı az olan problemler için uygun olduğunu savunan Michalewicz ve Janikow (1991), kısıt sayısı çok olan problemler için yeni genetik operatörlerin geliştirilmesinin genetik algoritmanın etkinliğini artıracağını vurgulamaktadırlar (Dengiz, Altıparmak, 1998).

Dengiz ve Altıparmak (1998), tesis yerleşimi problemi için Hossage ve Goodchild (1986), Tate ve Smith (1995a,b), Tam (1992); montaj hattı problemi için Anderson ve Ferris (1994); kafes sistemi tasarımı için Goldberg (1987); haberleşme şebeklerinin tasarımı probleminde Dengiz ve arkadaşları (1997a,b) geliştirdikleri genetik algoritmalarda, ceza fonksiyonu yaklaşımını kullandıklarını belirtmişlerdir.

Düzenleyici algoritma veya tamir yöntemi, genetik operatörler sonucu elde edilen uygun olmayan dizileri, uygun duruma getirebilmek için özel bir algoritmanın geliştirilmesine dayalıdır. Bu algoritmaların en büyük dezavantajı probleme özgü olmasıdır. Örneğin, Michalewicz (1992) ulaştırma problemleri için bir düzenleyici algoritma geliştirmiştir. Diğer bir düzenleyici algoritma ise bilgisayar şebekelerinin tasarımı problemi için Denzig ve arkadaşları (1997b) tarafından geliştirilmiştir (Dengiz, Altıparmak, 1998).

Yeni genetik operatörler yönteminde, probleme özgü genetik operatörler geliştirilir. Amaç, genetik operatörlerle elde edilen yeni dizilerin uygun birer çözüm olmasını sağlamaktır. Sayısal kodlamanın kullanıldığı gezgin satıcı problemi için Goldberg ve Lingle (1985) PMX (partially crossover), Davis (1985) OX (order based crossover), Oliver ve arkadaşları (1987) CX (cycle crossover) adını verdikleri çaprazlama operatörlerini geliştirmişlerdir. Bu operatörler, diğer problemler için de yaygın olarak kullanılmaktadır (Dengiz, Altıparmak, 1998).

3.1.3 – Yapay Sinir Ağları

3.1.3.1 – Genel Bilgiler

Yapay sinir ağları, beynin çalışma ilkelerinin sayısal bilgisayarlar üzerinde taklit edilmesi fikri ile ortaya çıkmış ve ilk çalışmalar beyni oluşturan biyolojik hücrelerin veya literatürdeki ismiyle nöronların matematiksel olarak modellenmesi üzerinde yoğunlaşmıştır. Bu çalışmaların ortaya çıkardığı bulgular, her bir nöronun komşu nöronlardan bazı bilgiler aldığı ve bu bilgilerin biyolojik nöron dinamiğinin öngördüğü biçimde bir çıktıya dönüştürüldüğü şeklindeydi. Günümüzde, yapay sinir ağları olarak isimlendirilen alan, birçok nöronun belirli şekillerde bir araya getirilip, bir işlevin gerçeklenmesi üzerindeki yapısal olduğu kadar matematiksel ve felsefi sorunlara yanıt arayan bir bilim dalı olmuştur. İnsan beyninin çalışma mekanizmasını taklit etmeye çalışan bu sistemler, her ne kadar günümüz teknolojisinin ürettiği, birim işlem zamanı nanosaniyeler mertebesinde olan silikon lojik kapılar ile gerçeklenebilseler de insan beyninin birim işlem zamanı milisaniyeler mertebesindeki nöronlarının toplu biçimde ele alındıklarındaki işlevselliklerinden çok uzakta kalmaktadırlar. Yapay sinir ağları, karar hızı açısından insan beyni ile yarışabilecek aşamaya henüz gelmemiş olmalarına

rağmen, karmaşık eşleştirmelerin hassas bir biçimde gerçeklenebilmesi ve yapısal gelişmişliğe sahip olmaları nedeniyle, her geçen gün uygulama alanları genişlemektedir.

Bu anlamda, yapay sinir ağları konusu üzerinde çalışırken, bir ağ yapısının çözebileceği problem uzayının, insan beyninin çözebildiği problem uzayının oldukça kısıtlanmış bir alt kümesi olacağı gözden kaçırılmamalıdır. Diğer bir deyişle, insan beyninin bilgiyi işlemesindeki, kavramların ilişkilendirmesindeki ve çıkarım mekanizmalarındaki üstünlüğü açıktır. Yapay sinir ağları kavramını çekici kılan aşağıda sıralanmış temel özelliklerin algılanışında, bu noktanın gözden kaçırılmamasında yarar vardır (Efe, Kaynak, 2000).

Birinci özellik, sistemin paralelliği ve toplamsal işlevin yapısal olarak dağılmışlığıdır. Diğer bir deyişle, birçok nöron eşzamanlı olarak çalışır ve karmaşık bir işlev çok sayıda küçük nöron aktivitesinin bir araya gelmesinden oluşur. Bu da zaman içerisinde herhangi bir nöronun işlev dışı kalması durumunda ağ başarımını dikkate değer bir ölçüde etkilemeyeceği anlamına gelmektedir.

İkinci bir özellik ise genelleme yeteneğidir. Diğer bir deyişle, ağ yapısının eğitim sırasında kullanılan nümerik bilgilerden eşleştirmeyi betimleyen kaba özellikleri çıkarsaması ve böylelikle eğitim sırasında kullanılmayan girdiler için de anlamlı yanıtlar üretebilmesidir.

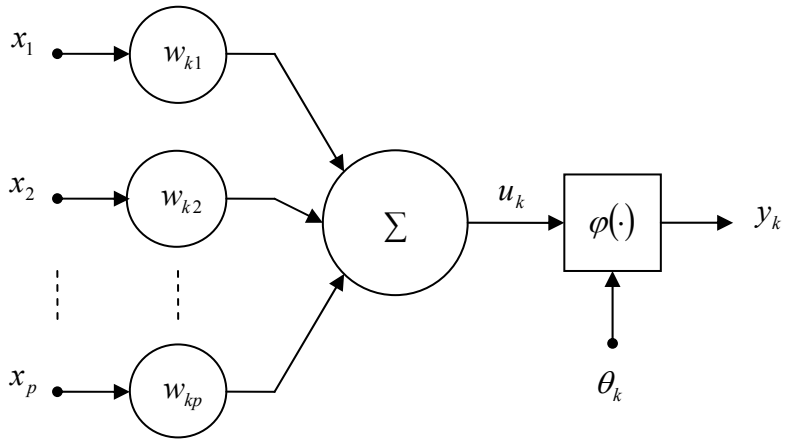
Bir başka özellik ise ağ fonksiyonunun doğrusal olmamasıdır. Yapı üzerinde dağılmış belli tipteki doğrusal olmayan alt birimler özellikle, istenen eşleştirmenin doğrusal olmadığı durumlarda, işlevin doğru olarak gerçeklenebilmesini matematiksel olarak olanaklı kılarlar. Burada, işlevin doğru biçimde gerçeklenebilmesi için yapısal bir esneklik gerekliliği vurgulanmalıdır. Yani ağ parametreleri, başarımı arttıracak veya maliyeti azaltacak şekilde değiştirilebilmelidir. Bu konuyla ilgili olarak, ilerleyen kısımlarda çeşitli aktivasyon fonksiyonları ele alınmıştır.

Matematiksel Nöron Modeli

Nöron, yapay sinir ağının çalışmasında temel eleman olan bir bilgi-işlem birimidir. Şekil 3.9 matematiksel nöron modelini göstermektedir. Nöron modelinin üç temel elemanından söz edilebilir:

- Her biri kendine ait bir ağırlıkla (weight veya strength) karakterize edilen, sinapsisler (synapses veya connecting links) kümesi. Özel olarak, j sinapsisindeki k nöronuna bağlı olan x_j sinyali, w_{kj} ağırlığıyla çarpılır. w_{kj} ağırlığında indislerinin yazım şekli önemlidir. İlk indis incelenen nörona, ikinci indis de sinapsisin giriş tarafındaki nörona karşılık gelmektedir. Literatürde bu gösterimin tersi de kullanılmaktadır. İlgili sinapsis yükseltici (excitatory) ise w_{kj} ağırlığı pozitif, indirgeyici (inhibitory) ise negatiftir.
- Sırasıyla nöronun sinapsisleri tarafından ağırlıklandırılmış giriş sinyallerini toplamak için bir toplayıcı (adder). Burada açıklanan işlemler bir doğrusal birleştirici (linear combiner) oluşturmaktadır.
- Nöron çıkışının büyümesini sınırlandırmak için bir aktivasyon fonksiyonu (activation veya squashing function). Tipik olarak, normalize edilmiş bir nöron çıkışı $[0,1]$ veya $[-1,+1]$ aralıklarındadır.

Şekil 3.9'da görülen nöron modeli, ayrıca, aktivasyon fonksiyonunun net girişini azaltıcı etki yapan, dışsal olarak uygulanan ve θ_k ile gösterilen bir eşik içermektedir. Öte yandan, eşik yerine bir bias kullanılarak, aktivasyon fonksiyonun net girişi arttırılabilir. Bias eşiğin negatifidir.



Şekil 3.9: Matematiksel Nöron Modeli

k nöronu,

$$u_k = \sum_{j=1}^{p} w_{kj} x_j \tag{3.22}$$

ve

$$y_k = \varphi(u_k - \theta_k) \tag{3.23}$$

denklemleriyle tanımalanabilir. Burada, $x_1, x_2, \ldots, x_p$ giriş sinyalleri, $w_{k1}, w_{k2}, \ldots, w_{kp}$ sinaptik ağırlıklar, u_k doğrusal birleştirici çıktısı, θ_k eşik, $\varphi(\cdot)$ aktivasyon fonksiyonu, ve y_k çıkış sinyalidir.

θ_k eşiğinin uygulanmasıyla, doğrusal birleştiricinin u_k çıkışında,

$$v_k = u_k - \theta_k \tag{3.24}$$

şeklinde bir dönüşüm olmaktadır.

θ_k eşik değerinin pozitif veya negatif olmasına bağlı olarak, k nöronun içsel aktivite düzeyi veya aktivasyon potansiyeli v_k ve doğrusal birleştirici çıkışı u_k arasındaki ilişki Şekil 3.10'da görüldüğü gibi değişmektedir. Bu dönüşümün sonucu olarak, v_k'nın u_k'ya göre grafiğinin orijinden geçmediği görülmektedir.

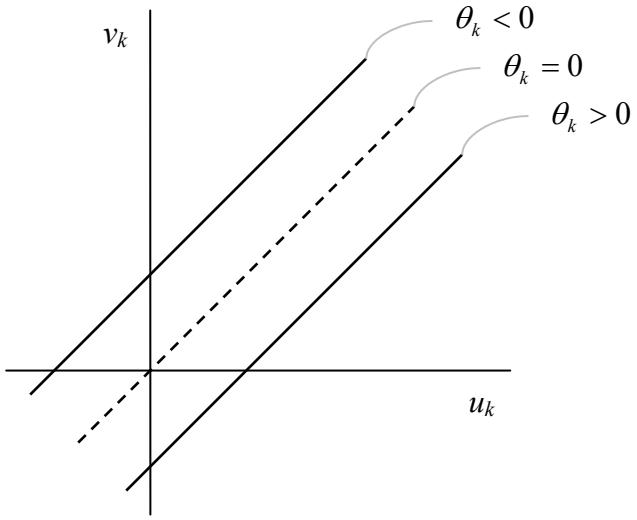
θ_k eşiği, yapay nöron k'nın dışsal bir parametresidir. Bu parametrenin dikkate alınmasıyla, yukarıdaki denklemler,

$$v_k = \sum_{j=0}^{p} w_{kj} x_j \tag{3.25}$$

ve

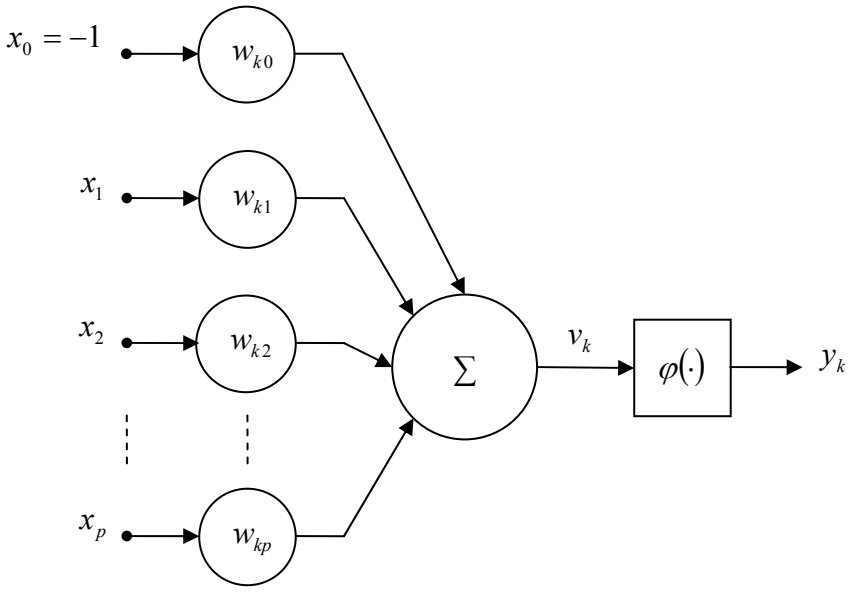
$$y_k = \varphi(v_k) \tag{3.26}$$

şeklinde yazılabilirler. Burada, girişi $x_0 = -1$ ve ağırlığı $w_{k0} = \theta_k$ olan yeni bir sinapsis eklenmiştir.



Şekil 3.10: u_k ve v_k Arasındaki İlişki

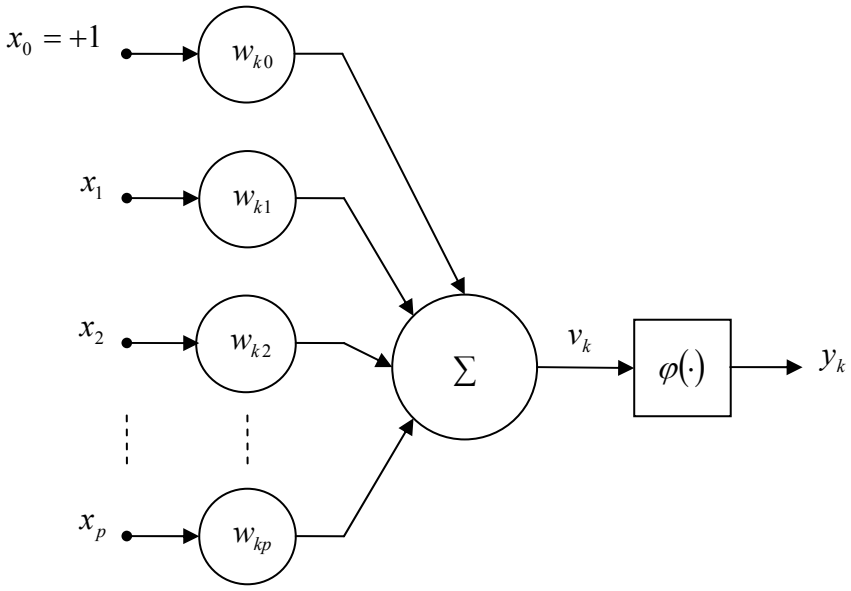
Bu durumda, nöron modeli Şekil 3.11'de görüldüğü gibi yeniden düzenlenebilir.



Şekil 3.11: Eşik Değerinin Giriş Olarak Alınması

Şekilde, eşiğin etkisi iki şey yapılarak gösterilmiştir: (i) -1 sabit değerli yeni bir sinyalin eklenmesi, ve (ii) θ_k eşik değerine eşit yeni bir sinapsisin eklenmesi.

Buna alternatif olarak, sabit girişin $x_0 = +1$ ve ağırlığın $w_{k0} = b_k$ olarak alınmasıyla, Şekil 3.12'deki nöron modeli elde edilir. Burada, b_k bias değerine karşılık gelmektedir. Şekil 3.10, 3.11 ve 3.12'deki modeller farklı görünümde olsalar da matematiksel olarak farklılıkları bulunmamaktadır.



Şekil 3.12: Bias Değerinin Giriş Olarak Alınması

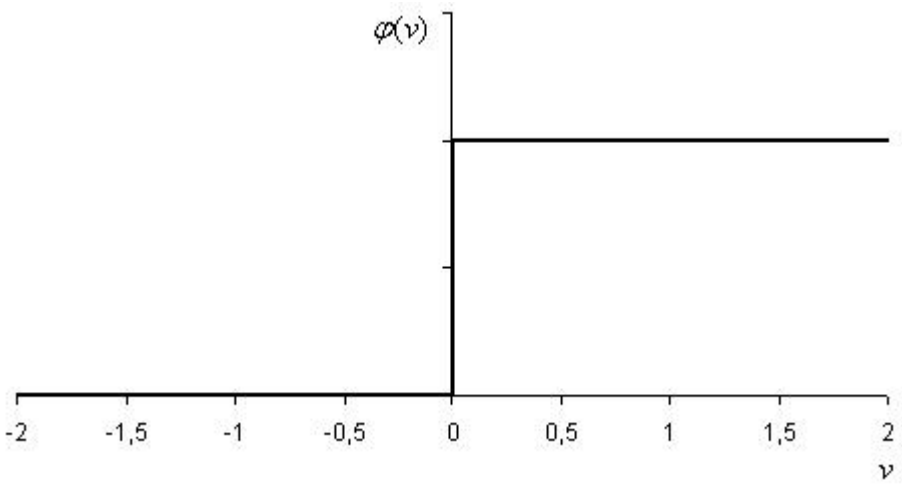
Aktivasyon Fonksiyonları

$\varphi(\cdot)$ ile gösterilen aktivasyon fonksiyonu, nöron çıkışını, girişindeki net aktivite düzeyi cinsinden tanımlamaktadır. Temel olarak; (i) sert geçişli, (ii) parçalı ve (iii) sigmoid olmak üzere üç farklı tipte aktivasyon fonksiyonundan söz edilebilir.

Şekil 3.13'te görülen sert geçişli aktivasyon fonksiyonu örneği,

$$\varphi(v) = \begin{cases} 1 & v \geq 0 \\ 0 & v < 0 \end{cases} \tag{3.27}$$

şeklinde tanımlanmaktadır.



Şekil 3.13: Sert Geçişli Fonksiyon

Böyle bir aktivasyon fonksiyonu içeren bir k nöronunun çıkışı,

$$y_k = \begin{cases} 1 & v_k \geq 0 \\ 0 & v_k < 0 \end{cases} \tag{3.28}$$

şeklinde tanımlanmaktadır. Burada, v_k nöronun içsel aktivite düzeyidir ve

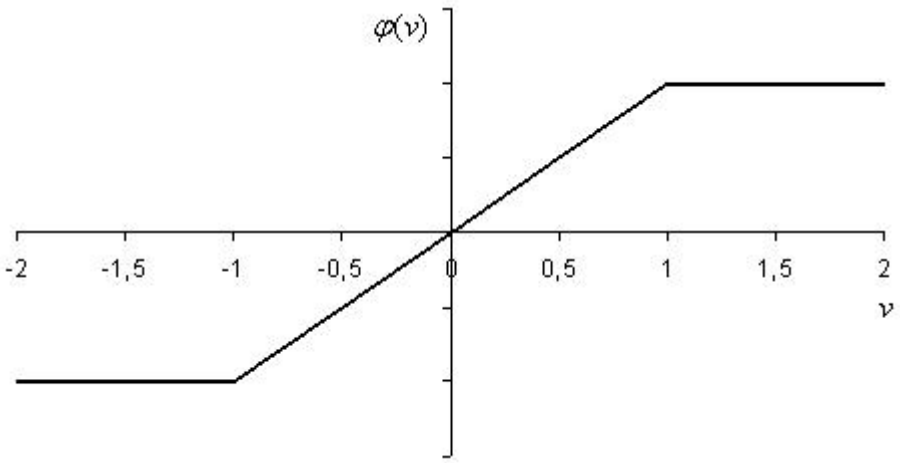
$$v_k = \sum_{j=1}^{p} w_{kj} x_j - \theta_k \tag{3.29}$$

şeklinde tanımlanmaktadır. Böyle bir nöron, literatürde McCulloch-Pitts modeli olarak bilinmektedir. Bu modelde, nöronun toplam içsel aktivite düzeyi negatif değilse, nöron çıkışı 1 değerini, diğer durumlarda 0 değerini almaktadır. Bu ifade, McCulloch-Pitts modelinin hepsi-veya-hiçbiri (all-or-none) özelliğini tanımlamaktadır.

Şekil 3.14'te görülen parçalı fonksiyon örneği,

$$\varphi(v) = \begin{cases} +1 & v \geq +1 \\ v & +1 > v > -1 \\ -1 & v \leq -1 \end{cases} \tag{3.30}$$

şeklinde tanımlanmaktadır. Aktivasyon fonksiyonun bu şekli, doğrusal olmayan yapıya benzerlik göstermektedir.



Şekil 3.14: Parçalı Fonksiyon

Yapay sinir ağlarında en çok kullanılan aktivasyon fonksiyonu şekli sigmoid fonksiyonudur. Örnek bir sigmoid fonksiyonu,

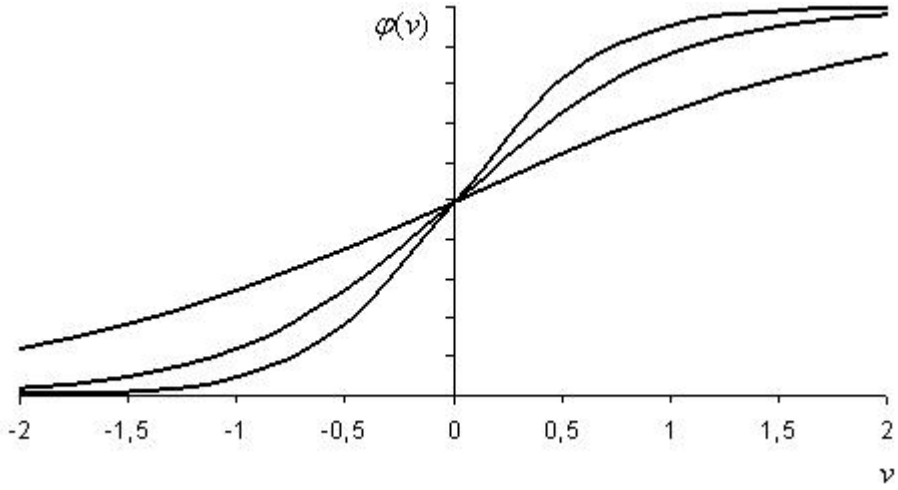
$$\varphi(v) = \frac{1}{1+\exp(-av)} \tag{3.31}$$

şeklinde tanımlanabilir. Burada, a sigmoid fonksiyonun eğim parametresidir. Bu parametreye farklı değerler verilerek, Şekil 3.15'de görüldüğü gibi farklı özellikler gösteren sigmoid fonksiyonları elde edilebilmektedir. Eğim parametresinin sonsuz olması durumunda, sigmoid fonksiyonu, eşik fonksiyonu şeklini almaktadır. Eşik fonksiyonu 0 veya 1 değerlerini alırken, sigmoid fonksiyonu 0 ile 1 arasında sürekli değerler almaktadır. Ayrıca, eşik fonksiyonu türevlenebilir değilken, sigmoid fonksiyonu türevlenebilirdir.

Yukarıda verilen sigmoid fonksiyonu 0 ile 1 arasında değerler almaktadır. -1 ile $+1$ arasında değerler alan bir sigmoid fonksiyonu gerektiğinde, Şekil 3.16'da görülen ve

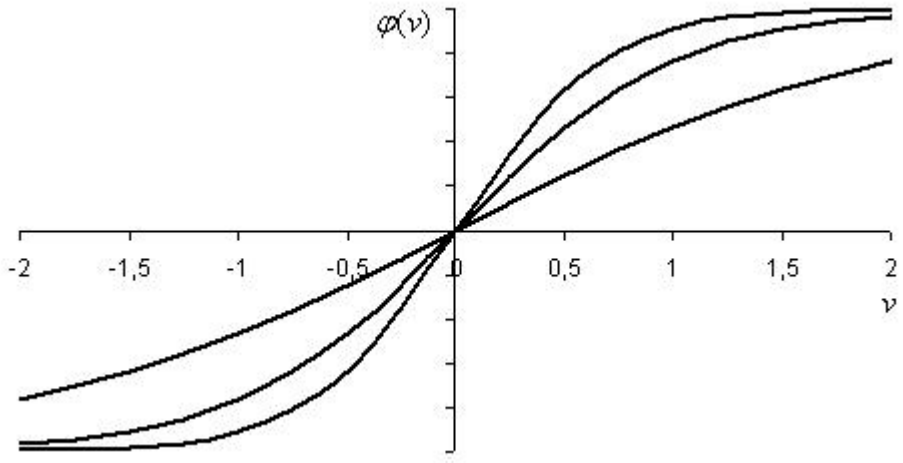
$$\varphi(v) = \tanh\left(\frac{v}{2}\right) = \frac{1-\exp(-av)}{1+\exp(-av)} \tag{3.32}$$

şeklinde tanımlanan, hiperbolik tanjant fonksiyonu kullanılabilir.



Şekil 3.15: Sigmoid Fonksiyonu

Burada, a sigmoid fonksiyonun eğim parametresidir. Bu parametreye farklı değerler verilerek, Şekil 3.16'da görüldüğü gibi farklı özellikler gösteren sigmoid fonksiyonları elde edilebilmektedir.



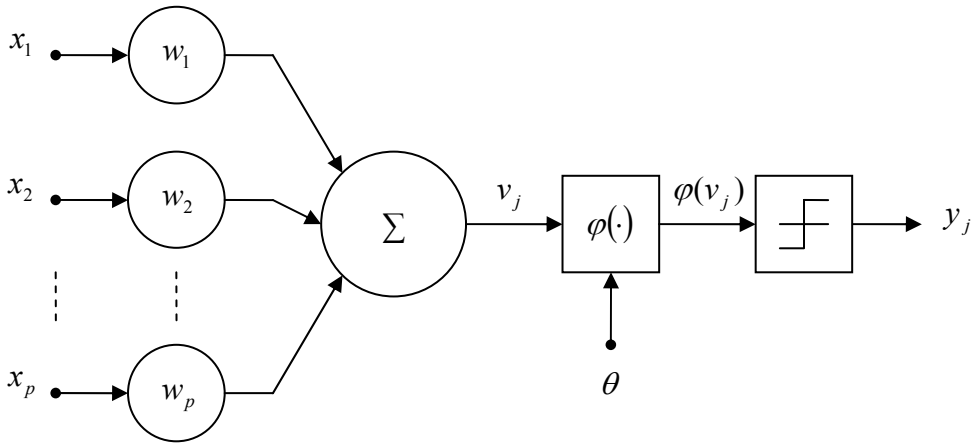
Şekil 3.16: Hiperbolik Tanjant Tipi Sigmoid Fonksiyonu

Perceptron

Perceptron, doğrusal olarak ayrılabilir örüntü tiplerini sınıflandırmak için kullanılan en basit yapay sinir ağı şeklidir. Temel olarak, Şekil 3.18'de görüldüğü gibi bir nöron, ayarlanabilir sinaptik ağırlıklar ve eşikten oluşmaktadır. Bu yapay sinir ağının

bağımsız parametrelerini ayarlamak için kullanılan algoritma, ilk olarak Rosenblatt tarafından geliştirilen bir öğrenme yöntemi olan perceptron beyin modelinde kullanılmıştır. Rosenblatt, perceptronu eğitmek için kullanılan vektörler, doğrusal olarak ayrılabilir iki ayrı sınıftan geliyorlarsa, perceptron algoritmasının, yakınsayarak karar yüzeyini bu iki sınıf arasında bir hiper-düzlem şeklinde konumlandıracağını ispatlamıştır. Yakınsama algoritmasının ispatı, perceptron yakınsama teoremi olarak bilinmektedir.

Şekil 3.17'de görülen bir-katmanlı perceptron bir nörondan oluşmaktadır. Böyle bir perceptron, örüntü sınıflandırma işlemini sadece iki sınıf için yapabilmektedir. Perceptronun çıkış katmanını bir nörondan daha fazla sayıda nöron içerecek şekilde genişleterek, sınıflandırma işlemi ikiden fazla sayıda sınıf için gerçekleştirilebilir. Bununla birlikte, perceptronun doğru olarak çalışabilmesi için sınıfların doğrusal olarak ayrılabilir olması gerekmektedir.



Şekil 3.17: Perceptron

Perceptron'un çalışmasına temel teşkil eden McCulloch-Pitts nöron modelidir. Şekil 3.17'de görülen, doğrusal birleştirici içeren böyle bir nöron modeli, sert geçişli bir sınırlandırıcı ile devam etmektedir. Nöron modeli, sinapsislerine uygulanan girişlerin doğrusal birleşimini hesaplamakta ve aynı zamanda, dışsal olarak uygulanan eşik değerini de dikkate almaktadır. Elde edilen sonuç sert geçişli sınırlandırıcıya verilmektedir. Sınırlandırıcı girişi pozitifse, nöron tarafından $+1$ çıkışı; negatifse -1 çıkışı üretilir.

Şekil 3.17'de, bir-katmanlı perceptronun sinaptik ağırlıkları $w_1, w_2, \ldots, w_p$ ile perceptrona uygulanan girişler de $x_1, x_2, \ldots, x_p$ ile gösterilmektedir. Perceptronda sadece bir nöron söz konusu olduğundan, ağırlıkların gösteriminde nörona karşılık gelen ikinci indis kullanılmamıştır. Dışsal olarak uygulanan eşik θ ile gösterilmektedir. Böylece, doğrusal birleştirici çıkışı,

$$v = \sum_{i=1}^{p} w_i x_i - \theta \tag{3.33}$$

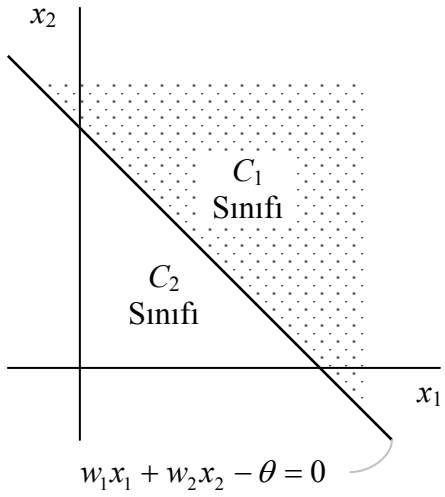
olarak elde edilmektedir. Perceptronun amacı dışsal olarak uygulanan $x_1, x_2, \ldots, x_p$ kümesinin elemanlarını C_1 ve C_2 sınıflarından birisine atamaktır. Sınıflandırma için karar verme kuralı, $x_1, x_2, \ldots, x_p$ girişleriyle gösterilen noktanın, perceptron çıkışı y, $+1$ ise C_1 ve -1 ise C_2 sınıfına atanması şeklindedir.

Örüntü sınıflandırıcısının davranışını göstermek amacıyla, giriş değişkenleri $x_1, x_2, \ldots, x_p$ tarafından bölünen p-boyutlu sinyal uzayında karar bölgeleri çizilebilir. Temel bir perceptron için

$$\sum_{i=1}^{p} w_i x_i - \theta = 0 \tag{3.34}$$

ifadesi ile tanımlanan iki karar bölgesi bulunmaktadır. Bu durum, Şekil 3.18'de, x_1 ve x_2 gibi iki giriş değişkeninin olduğu durum için gösterilmiştir. Burada, karar sınırı doğru şeklindedir. Sınırı oluşturan doğrunun üstünde kalan bir (x_1, x_2) noktası C_1 sınıfına, altında kalan bir (x_1, x_2) noktası da C_2 sınıfına atanmaktadır. θ eşik değerinin, sadece karar sınırının orijine olan uzaklığını değiştirdiği görülmektedir.

$w_1, w_2, \ldots, w_p$ sinaptik ağırlıkları iteratif olarak ayarlanabilirler. Bu işlem için bir hata-düzeltme kuralı kullanılabilir.



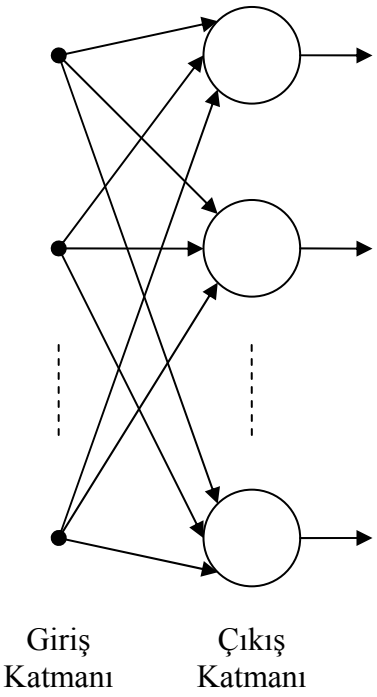
Şekil 3.18: Perceptron ile İki Boyutlu Doğrusal Sınıflandırma

3.1.3.2 – Ağ Yapıları

Ağdaki nöronların yapısı, eğitim için kullanılan öğrenme algoritmasıyla yakından ilişkilidir. Öğrenme algoritmalarının sınıflandırılması izleyen kısımlarda verilmiştir. Haykin'e (1994) göre, genel olarak, dört farklı ağ yapısından söz edilebilir.

Bir-Katmanlı İleri-Beslemeli Ağlar (Single-Layer Feedforward Networks)

Bir-katmanlı yapay sinir ağı, nöronların katman şeklinde organize olduğu bir ağdır. Bir katmanlı ağın en basit şeklinde, sadece kaynak düğümlerinden oluşan bir giriş katmanı, çıkış katmanındaki nöronlara bağlıdır. Bu ağ ileri-beslemeli olmak zorundadır. Şekil 3.19'da böyle bir ağ örneği görülmektedir. Böyle bir ağ, bir-katmanlı ağ olarak adlandırılır. Buradaki katman çıkış katmanını ifade etmektedir. Diğer bir deyişle, kaynak düğümlerinden oluşan giriş katmanı, burada herhangi bir işlem yapılmadığı için dikkate alınmamaktadır.



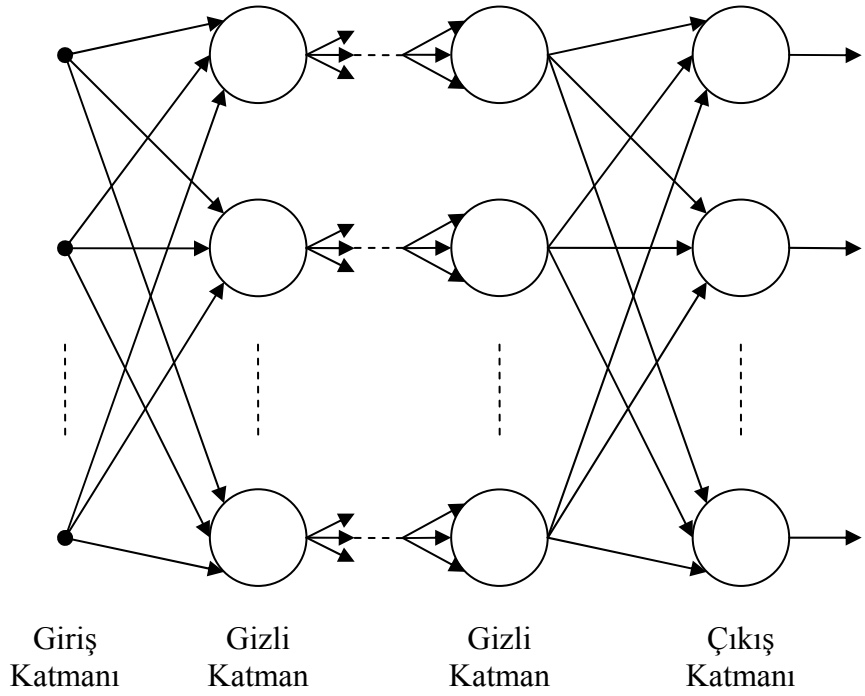
Şekil 3.19: Bir-Katmanlı İleri-Beslemeli Yapay Sinir Ağı

Çok-Katmanlı İleri-Beslemeli Ağlar (Multilayer Feedforward Networks)

Bu ileri-beslemeli yapay sinir ağlarında, gizli nöronlar (hidden neurons) veya gizli birimler (hidden units) olarak adlandırılan hesaplama birimleri olan, bir veya daha fazla sayıda gizli katman bulunmaktadır. Gizli nöronların fonksiyonu, ağın giriş ve çıkışı arasındaki bağlantıyı sağlamaktır. Bir veya daha fazla sayıda gizli katman eklenmesiyle, yapay sinir ağı daha karmaşık ilişkileri modelleyebilecek hale gelmektedir. Bu durum, giriş katmanının boyutu büyüdükçe daha da önem kazanmaktadır.

Yapay sinir ağının giriş katmanındaki kaynak düğümleri, giriş vektörü elemanlarını ikinci katmandaki veya ilk gizli katmandaki nöronlara giriş sinyalleri olarak verirler. İkinci katmanın çıkış sinyalleri, üçüncü katmanın girişi olarak kullanılır ve ağın geri kalanında bu şekilde devam eder. Tipik olarak, herhangi bir katmandaki nöronlar, girişlerini, sadece bir önceki katmandaki nöronların çıkışlarından alırlar. Çıkış

katmanındaki nöronların çıkış sinyalleri kümesi, yapay sinir ağının giriş vektörüne verdiği cevabı oluşturur. Şekil 3.20'de böyle bir ağ görülmektedir.



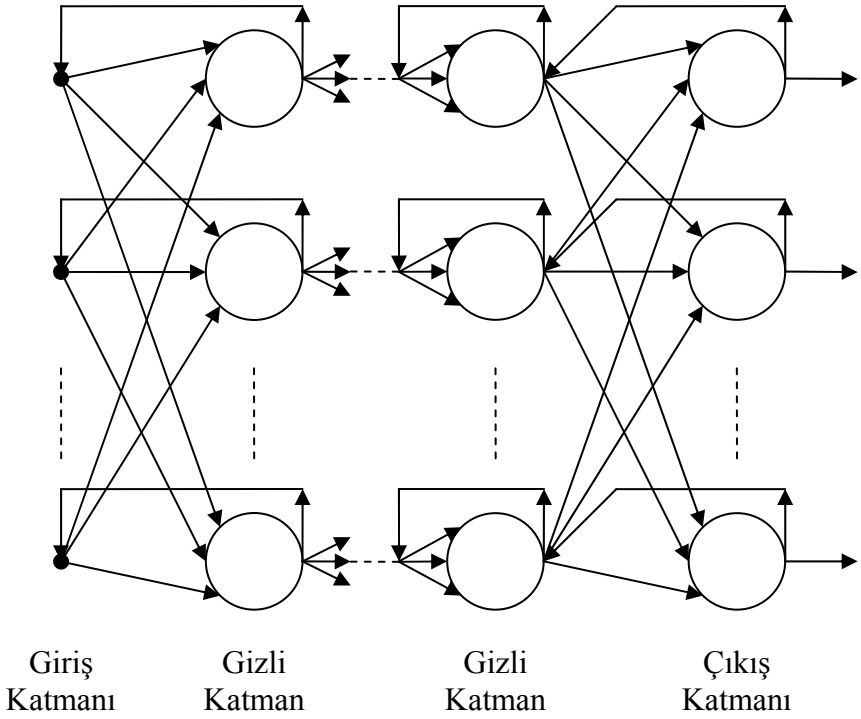
Şekil 3.20: Çok-Katmanlı İleri-Beslemeli Yapay Sinir Ağı

Kaynak düğüm sayısı p, birinci gizli katmandaki nöron sayısı h_1, ikinci gizli katmandaki nöron sayısı h_2, $i.$ katmandaki nöron sayısı h_i ve çıkış katmanındaki nöron sayısı q olan bir yapay sinir ağı p-h_1-h_2-…-h_i-…-q ağı olarak adlandırılır.

Bu yapay sinir ağları genellikle, çok-katmanlı perceptronlar (multilayer perceptrons – MLP) olarak bilinmektedir.

Yinelemeli Ağlar (Recurrent Networks)

Yinelemeli ağlar, en az bir geri besleme döngülerinin olması nedeniyle ileri-beslemeli ağlardan ayrılırlar. Örneğin, bir katmandan oluşan bir yinelemeli ağda, bu katmandaki her nöronun çıkışı, diğer nöronlara giriş olarak verilebilir. Şekil 3.21'de böyle bir ağ görülmektedir.



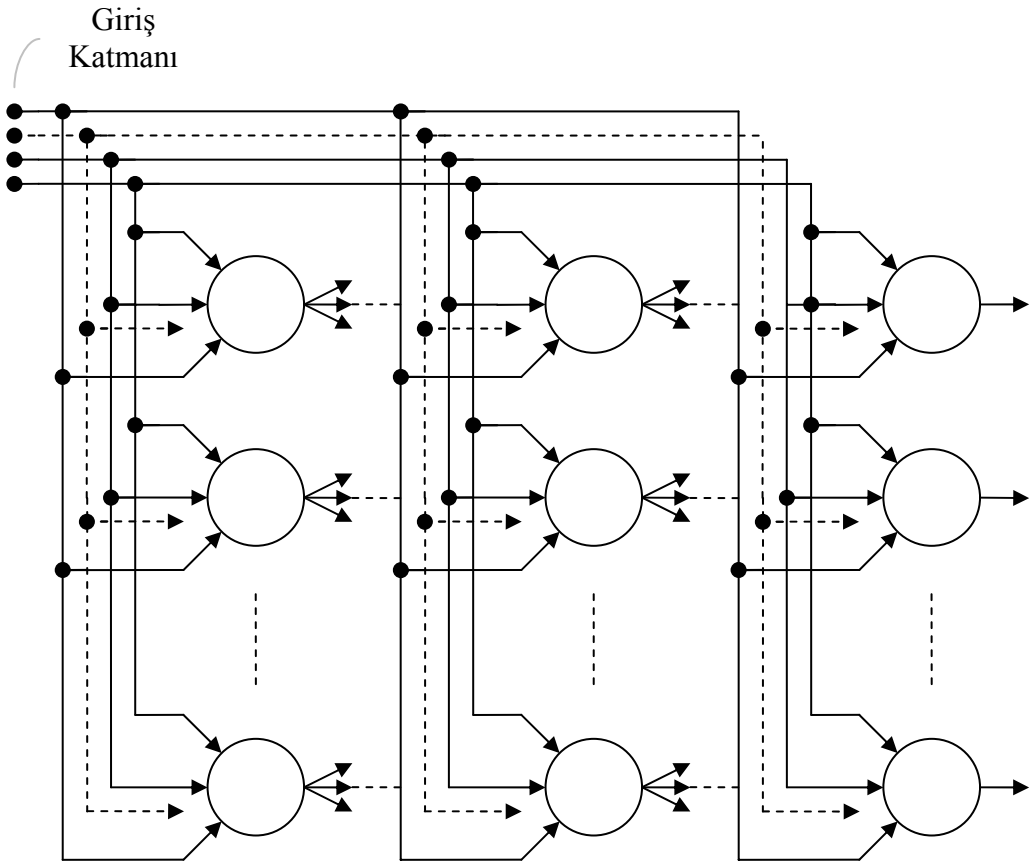
Şekil 3.21: Yinelemeli Yapay Sinir Ağı

Yinelemeli ağlar, kendine-geribesleme (self-feedback) döngüsü olanlar ve olmayanlar olmak üzere ikiye ayrılabilir. Kendine-geribesleme, bir nöron çıkışının, aynı nörona giriş olarak verilmesini ifade etmektedir. Yinelemeli ağlar, gizli nöronlar içerebilirler. Yinelemeli ağlarda bulunan geribesleme döngüleri, ağın öğrenme yeteneği ve performansı üzerinde önemli etki yapmaktadırlar. Bunun dışında, geribesleme döngüleri birim-gecikme elemanları (unit-delay elements) içermektedirler. Bu elemanlar, nöronların doğrusal olmayan yapısına dayanarak, doğrusal olmayan dinamik bir davranış sağlarlar. Doğrusal olmayan dinamik davranış şekli, yinelemeli ağların önemli bir özelliğidir.

Kafes Yapıları (Lattice Structures)

Kafes, bir-boyutlu, iki-boyutlu veya daha fazla sayıda boyutlu nöron dizileri ve bunlara karşılık gelen ve giriş sinyallerini diziye veren kaynak düğümleri kümesinden oluşmaktadır. Kafes boyutu, grafiğin bulunduğu uzayın boyut sayısını göstermektedir.

Şekil 3.22'de böyle bir yapay sinir ağı görülmektedir. Giriş katmanındaki her kaynak düğümünün, kafesteki her nöronla bağlantılı olduğu görülmektedir. Kafes ağı, çıkış nöronlarının satır ve sütunlar şeklinde düzenlendiği ileri-beslemeli bir ağdır.



Şekil 3.22: Kafes Yapılı Yapay Sinir Ağı

3.1.3.3 – Öğrenme

Yapay sinir ağlarının çok sayıda ilginç özelliklerinden birisi de ağın öğrenme ve performansını geliştirme yeteneğinin olmasıdır. Yapay sinir ağlarının öğrenme işlemi, sinaptik ağırlıklarına ve eşiklerine uygulanan iteratif bir ayarlama süreciyle gerçekleşmektedir. Öğreneme konusunda literatürde çeşitli tanımlamalar bulunmaktadır. Haykin'e (1994) göre öğrenme aşağıdaki şekilde tanımlanmıştır:

Öğrenme, bir yapay sinir ağının bağımsız parametrelerinin, ağın bulunduğu çevre tarafından oluşturulan etkilerle ayarlandığı, sürekli bir süreçtir. Öğrenme tipi, parametre değişimlerinin nasıl gerçekleştiği ile ilgilidir.

Bu tanımdan, öğrenme sürecinde aşağıdaki olayların varlığından söz edilebilir:

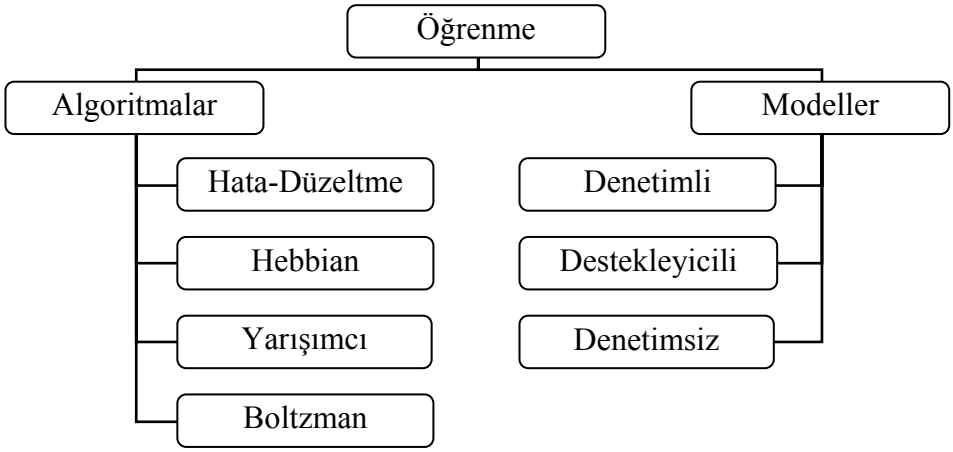
- Yapay sinir ağı bir çevre tarafından etkilenmektedir.
- Bu etkilenmenin bir sonucu olarak yapay sinir ağında değişiklikler olmaktadır.
- İç yapısında meydana gelen değişiklikler dolayısıyla, yapay sinir ağı, çevreye farklı bir şekilde cevap vermektedir.

Öğrenme süreci, öğrenme algoritmaları (learning algorithms) ve öğrenme modelleri (learning paradigms) olmak üzere iki kısımda incelenmiştir (Haykin, 1994). Öğrenme algoritmaları kısmında, dört temel öğrenme algoritması ele alınmıştır: (i) hata-düzeltme (error-correction) öğrenme algoritması, (ii) Hebbian öğrenme algoritması, (iii) yarışımcı (competitive) öğrenme algoritması ve (iv) Boltzman öğrenme algoritmsı. Öğrenme modelleri kısmında da üç temel model ele alınmıştır: (i) denetimli (supervised) öğrenme, (ii) destekleyicili (reinforcement) öğrenme ve (iii) denetimsiz (unsupervised) öğrenme. Bu sınıflandırma Şekil 3.23'te görülmektedir.

Öğrenme Algoritmaları

Hata-Düzeltme Öğrenme Algoritması (Error-Correction Learning Algorithm)

$d_k(n)$, n zamanında, k nöronundaki istenen cevabı göstersin. Bu nöronun karşılık gelen gerçek cevabı da $y_k(n)$ olsun. $y_k(n)$ cevabı, k nöronunu içeren ağa $x(n)$ vektörünün uygulanmasıyla üretilmiştir. Giriş vektörü $x(n)$ ve k nöronu için istenen çıkış $d_k(n)$, n zamanında ağa uygulanan belirli bir örneği oluşturmaktadırlar. Ağa uygulanan bu ve diğer bütün örneklerin olasılıklı bir çevre tarafından oluşturulduğu varsayılmakla birlikte, olasılık dağılımı bilinmemektedir.



Şekil 3.23: Öğrenme Algoritmaları ve Öğrenme Modelleri

Tipik olarak, k nöronunun $y_k(n)$ gerçek cevabı, $d_k(n)$ istenen cevabından farklıdır. Bu yüzden, $d_k(n)$ istenen cevabı ile $y_k(n)$ gerçek cevabı arasındaki fark, bir hata sinyali olarak,

$$e_k(n) = d_k(n) - y_k(n) \tag{3.35}$$

şeklinde tanımlanabilir. Hata-düzeltme öğrenme algoritmasının amacı, $e_k(n)$ hata sinyaline bağlı bir maliyet fonksiyonunu, ağdaki her çıkış nöronunun gerçek cevabının, bu nöronun istenen cevabına yaklaşmasını sağlayacak şekilde, enküçüklemektir. Bir maliyet fonksiyonu seçildikten sonra, hata-düzeltme öğrenmesi bir eniyileme problemi haline gelmektedir. Maliyet fonksiyonu için genellikle kullanılan kriter olan ortalama-karesel-hata,

$$J = E\left[\frac{1}{2}\sum_k e_k^2(n)\right] \tag{3.36}$$

şeklinde tanımlanmaktadır. Burada, E beklenen değer operatörüdür ve toplam, ağın çıkış katmanındaki bütün nöronları içermektedir. $1/2$ çarpanı, J maliyet fonksiyonunun ağın bağımsız parametrelerine göre enküçüklenmesi için türevi alındığında, sadeleşme olması için kullanılmaktadır. Bu denklem, sürecin geniş-anlamda durağan (wide-sense stationary) olduğunu varsaymaktadır. J maliyet

fonksiyonunun ağ parametrelerine göre enküçüklenmesi, eğim düşümü yöntemi (method of gradient descent) olarak adlandırılmaktadır. Bununla birlikte, bu eniyileme sürecinin zorluğu, sürecin istatistiksel karakteristiklerinin bilinmesini gerektirmesidir. Bu zorluğu aşmak için eniyileme problemine yaklaşık bir çözüm bulunur. Spesifik olarak, hata kareleri toplamının anlık değeri dikkate alınır:

$$\varepsilon(n) = \frac{1}{2}\sum_{k} e_k^2(n) \tag{3.37}$$

Bundan sonra, ağ $\varepsilon(n)$ hata kareleri toplamı anlık değerinin, ağın sinaptik ağırlıklarına göre enküçüklenmesiyle eniyilenir. Böylece, hata-düzeltme öğrenme algoritmasına göre, n zamanında w_{kj} sinaptik ağırlığında yapılan ayarlama $\Delta w_{kj}(n)$,

$$\Delta w_{kj}(n) = \eta e_k(n) x_j(n) \tag{3.38}$$

şeklinde tanımlanmaktadır. Burada, η öğrenme oranını (rate of learning) belirleyen pozitif bir katsayıdır. Diğer bir deyişle, sinaptik ağırlığa yapılan ayarlama, hata sinyali ile ele alınan sinapsis girişinin çarpımına eşittir. Bu girişin, incelenen nörona çıkış veren nöronun çıkışıyla aynı olduğu görülmektedir.

Hata-düzeltme öğrenme algoritması, $w_{kj}(n)$ sinaptik ağırlılığına uygulanan, $\Delta w_{kj}(n)$ düzeltmesini hesaplamak için $e_k(n)$ hatasını dikkate almaktadır. $e_k(n)$ hata sinyali Denklem 3.35'ten hesaplanmaktadır. Güncellenmiş (yeni) ağırlıkları hesaplamak için Denklem 3.38 kullanılmaktadır. Hata-düzeltme öğrenme algoritması, literatürde delta kuralı olarak da adlandırılmaktadır.

Hebbian Öğrenme Algoritması (Hebbian Learning Algorithm)

Hebb'in öğrenme varsayımı, bütün öğrenme algoritmaları içinde en eski ve en çok bilinenidir. Aşağıdaki tanım Hebb'in Davranış Organizasyonu (The Organization of Behavior) kitabından alınmıştır:

A hücresinin bir aksonu bir B hücresini harekete geçirecek kadar yakın olduğunda ve onu tetiklemek için tekrarlı ve sürekli olarak hareket ettiğinde, bu

hücrelerden birinde veya her ikisinde; B'yi tetikleyen bir hücre olarak, A hücresinin etkinliği arttırılacak şekilde; bir miktar gelişme süreci veya matabolik değişimler oluşur.

Hebb bu değişimi (hücresel düzeyde) çağrışımlı öğrenmenin temeli (a basis of associative learning) olarak adlandırmıştır.

Yukarıdaki ifade kullanılarak, aşağıdaki iki kural tanımlanabilir:

- Bir sinapsisin her iki yanındaki iki nöron eşzamanlı olacak şekilde aktive edilmişlerse, bu sinapsisin ağırlığı artmaktadır.
- Bir sinapsisin her iki yanındaki iki nöron eşzamanlı olmayacak şekilde aktive edilmişlerse, bu sinapsisin ağırlığı azalmakta veya yok olmaktadır.

Böyle bir sinapsis Hebbian sinapsisi olarak adlandırılır. Orijinal Hebb kuralı, yukarıdaki kurallardan ikincisini içermemektedir. Hebbian sinapsisi, sinaptik etkinliği arttırmak için önceki-sinaptik (presynaptic) ve sonraki-sinaptik (postsynaptic) arasındaki ilişkinin bir fonksiyonu olarak; zamana bağlı, yüksek derecede yerel ve güçlü interaktif bir mekanizma kullanmaktadır. Bu tanımdan, bir Hebbian sinapsisini karakterize eden dört anahtar mekanizma tanımlanabilir:

- Zamana bağlı mekanizma. Bu mekanizma, bir Hebbian sinapsisindeki değişikliklerin, önceki-sinaptik ve sonraki-sinaptik aktivitelerin oluş zamanına bağlı olduğunu ifade etmektedir.
- Yerel mekanizma. Doğası gereği, bir sinapsis, bilgi-taşıma sinyallerinin geçici olarak aynı yerde olduğu bir aktarım yeridir. Bu yerel bilgi, Hebbian sinapsisi tarafından, girişe-özel, yerel bir sinaptik değişiklik için kullanılır. Hebbian sinapsislerinden oluşan bir yapay sinir ağına denetimsiz öğrenme olanağı sağlayan bu yerel mekanizmadır.
- İnteraktif mekanizma. Bir Hebbian sinapsisindeki değişim olma sayısının, sinapsisin her iki trafındaki aktivite düzeylerine bağlı olduğu görülmektedir. Bu, iki aktivitenin herhangi birisinden bir tahmin yapılamaması anlamında, önceki-sinaptik ve sonraki-sinaptik aktiviteler arasındaki "doğru etkileşime" bağlı bir Hebbian öğrenme şeklidir. Bu

bağımlılık veya etkileşimin deterministik veya istatistiksel olabileceği görülmektedir.

- Birleşimsel veya ilişkisel mekanizma. Hebb'in öğrenme varsayımının bir yorumu, sinaptik etkinlikteki bir değişim için şartın, önceki-sinaptik ve sonraki-sinaptik aktivitelerin birleşimi olması şeklindedir. Bu yoruma göre, önceki-sinaptik ve sonraki-sinaptik aktivitelerin kısa bir zaman aralığındaki gerçekleşme sayıları sinaptik değişiklikleri hesaplamak için yeterlidir. Bu yüzden, Hebbian sinapsisi, birleşimsel sinapsis (conjunctional synapse) olarak de adlandırılmaktadır. Hebb'in öğrenme varsayımının bir diğer yorumuna göre, Hebbian sinapsisini istatistiksel terimlerle karakterize eden interaktif bir mekanizmadan söz edilebilir. Özel olarak, önceki-sinaptik ve sonraki-sinaptik aktiviteler arasındaki ilişki sinaptik bir değişim nedeni olarak görülmektedir. Hebbian sinapsisi, aynı zamanda ilişkisel bir sinapsise karşılık gelmektedir.

Yarışımcı Öğrenme

Yarışımcı öğrenmede, adından da anlaşıldığı gibi, yapay sinir ağının çıkış nöronları aktif olan (tetiklenen) nöron olmak için birbirleriyle yarışmaktadır. Bu yüzden, Hebbian öğrenme algoritmasına göre aynı anda birden fazla nöron aktif olabilirken, yarışımcı öğrenme algoritmasında aynı anda sadece bir nöron aktif olmaktadır. Yarışımcı öğrenmenin, farklı giriş örüntülerini sınıflandırmada uygun olmasının nedeni budur.

Yarışımcı öğrenme algoritmasının üç temel elemanı bulunmaktadır:

- Bazı rassal dağılmış sinaptik ağırlıklar ve bu yüzden, verilen bir giriş örüntü kümesine farklı şekilde cevap veren sinaptik ağırlıklar dışında hepsi aynı olan nöronlar kümesi.
- Her nöronun "etkisini" sınırlandıran bir limit.
- Verilen bir giriş altkümesine cevap vermek için nöronlara, sadece bir çıkış nöronu veya her grup için sadece bir nöron aktif olacak şekilde, yarışma

olanağı veren bir mekanizma. Kazanan nöron, "kazanan-hepsini-alır nöronu" (winner-tekes-all nuron) olarak adlandırılmaktadır.

Ağın her nöronu, benzer örüntü kümeleri üzerinde uzmanlaşmak için öğrenmektedir ve bu yüzden, bu nöronlar, "özellik bulucular" (feature detectors) olarak adlandırılmaktadırlar.

Yarışımcı öğrenmenin en basit şeklinde, yapay sinir ağı, giriş düğümlerinin tamamına bağlı olan, çıkış nöronları katmanından oluşmaktadır. Ağ, aynı katmandaki nöronlar arasında "yan bağlantılar" (lateral connections) da içerebilmektedir. Burada tanımlanan ağ yapısında, yan bağlantıları olan nöronlar birbirlerini etkilemektedirler.

Kazanan j nöronunun, belirli bir giriş örüntüsü x için içsel aktivite düzeyi v_j, ağdaki bütün nöronlardan daha büyük olmalıdır. Kazanan j nöronunun çıkış sinyali y_j bir, kaybeden diğer nöronların çıkış sinyalleri de sıfır olarak alınır.

w_{ji}, i giriş düğümünden j çıkış nöronuna olan sinaptik ağırlığı göstersin. Her nörona, giriş düğümleri arasında dağıtılan sinaptik ağırlıktan (bütün sinaptik ağırlıklar pozitiftir) sabit bir miktar,

$$\sum_i w_{ji} = 1, \text{ tüm } j \text{'ler için} \tag{3.39}$$

şeklinde atanır. Bir nöron, sinaptik ağırlıklarını, aktif olmayan giriş düğümlerinden aktif olanlara değiştirerek öğrenmektedir ve belirli bir giriş örüntüsüne cevap vermezse, bu nöronda öğrenme olmaz. Belirli bir nöron kazanırsa, bu nöronun her giriş düğümü, sinaptik ağırlığının bir kısmını alır ve daha sonra, bu ağırlık, aktif giriş düğümleri arasında eşit olarak dağıtılır. Standart yarışımcı öğrenme algoritmasına göre, w_{ji} sinaptik ağırlığına uygulanan Δw_{ji} değişim miktarı aşağıdaki şekilde tanımlanmaktadır:

$$\Delta w_{ji} = \begin{cases} \eta(x_i - w_{ji}) & \text{nöron } j \text{ kazanırsa} \\ 0 & \text{nöron } j \text{ kaybederse} \end{cases} \tag{3.40}$$

Burada, η öğrenme-oranı parametresidir. Bu kural, kazanan j nöronunun w_j sinaptik ağırlık vektörünü, x giriş örüntüsüne doğru hareket ettiren genel bir etkiye sahiptir.

Boltzman Öğrenme Algoritması (Boltzman Learning Algorithm)

Boltzman öğrenme algoritması, termodinamik varsayımlarından türetilmiştir. Bir Boltzman makinesinde, nöronlar tekrarlayan bir yapı oluştururlar, ve $+1$ ile gösterilen bir "açık" (on) veya -1 ile gösterilen bir "kapalı" durumunda olacak şekilde ikili bir mantıkla çalışırlar. Makine, değeri belirli durumlar tarafından belirlenen bir E enerji fonksiyonuyla karakterize edilir:

$$E = -\frac{1}{2}\sum_i \sum_j w_{ji} s_j s_i \ , \ i \neq j \tag{3.41}$$

Burada, s_i, i nöronunun durumu ve w_{ji}, i nöronunu, j nöronuna bağlayan sinaptik ağırlıktır. $i \neq j$ ifadesi, makinedeki nöronların hiçbirisinin kendisine geri beslemesi olmadığı anlamına gelmektedir. Makine rassal olarak bir j nöronu seçip, bu nöronu, bir T sıcaklığında, s_j durumundan $-s_j$ durumuna,

$$W(s_j \rightarrow -s_j) = \frac{1}{1+\exp(-\Delta E_j / T)} \tag{3.42}$$

olasılığına göre çevirmektedir. Burada, ΔE_j çevirme sonucunda oluşan enerji değişimi olup, T fiziksel bir sıcaklığı ifade etmemektedir. Bu kural tekrarlı olarak uygulanırsa, makine termal dengeye ulaşacaktır.

Boltzman makinesindeki nöronlar görünür ve gizli olmak üzere iki fonksiyonel gruba ayrılmaktadır. Görünür nöronlar, ağ ve ağın çalıştığı çevre arasında bir etkileşim sağlarlarken, gizli nöronlar bağımsız olarak çalışırlar.

Öğrenme Modelleri

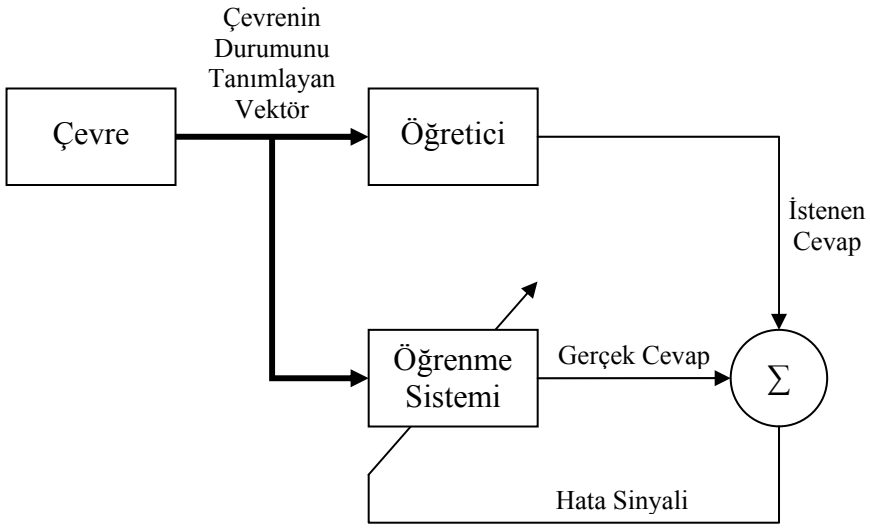
Denetimli Öğrenme Modeli (Supervised Learning Paradigm)

Denetimli öğrenmenin gerekli bir bileşeni, Şekil 3.24'te de görüldüğü gibi dışsal bir öğreticinin bulunmasıdır. Öğretici, çevre hakkında, giriş-çıkış örnekleri ile gösterilen bilgiye sahiptir. Bununla birlikte, çevre yapay sinir ağı tarafından bilinmemektedir. Öğretici ve yapay sinir ağına, bir eğitim vektörü verildiği varsayılsın. Öğreticinin bu eğitim vektörüne karşılık gelen istenen cevapları yapay sinir ağına vermesi gerekmektedir. İstenen cevap, yapay sinir ağı tarafından yapılacak eniyi hareketi göstermektedir. Ağ parametreleri, eğitim vektörü ve hata sinyalinin birleştirilmiş etkisiyle ayarlanmaktadır. Bu ayarlama, yapay sinir ağının öğreticiye benzer hale gelmesini sağlayacak şekilde iteratif olarak devam etmektedir. Diğer bir deyişle, öğreticinin çevre hakkındaki bilgisi, mümkün olduğu ölçüde yapay sinir ağına aktarılmaktadır. Bu şart sağlandıktan sonra, yapay sinir ağı öğreticisiz olarak çevreyle etkileşime geçmeye hazır hale gelmektedir.

Burada tanımlanan öğreticili öğrenme modeli, daha önce tanımlanmış olan hata-düzeltme öğrenme algoritması şeklindedir. Bu, bir kapalı-döngü geribesleme sistemi olmakla birlikte, bilinmeyen çevre döngü içinde değildir. Sistemin performans ölçüsü, sistemin bağımsız parametrelerinin bir fonksiyonu olarak tanımlanan ortalama-karesel hata alınabilir. Bu fonksiyon, bağımsız parametrelerin koordinatları oluşturduğu, çok-boyutlu bir hata yüzeyi veya kısaca hata yüzeyi olarak gösterilebilir.

Doğru hata yüzeyi, olası bütün giriş-çıkış örneklerinin ortalamasıdır. Sistemin öğreticinin denetimi altındaki herhangi bir işlemi, hata yüzeyi üzerinde bir nokta olarak gösterilmektedir. Sistemin zaman içinde performansının artması ve öğreticiden öğrenebilmesi için işlem noktasının hata yüzeyi üzerinde bir enküçük noktaya doğru hareket etmesi gerekmektedir. Bu enküçük nokta, yerel enküçük veya genel enküçük olabilir. Denetimli öğrenme sistemi bunu, sistemin o anki davranışına karşılık gelen hata yüzeyinin eğimi hakkında sahip olduğu bilgiye dayanarak gerçekleştirebilmektedir. Hata yüzeyinin herhangi bir noktadaki eğimi, enbüyük düşmenin olduğu noktayı işaret eden bir vektördür. Örneklerden denetimli öğrenmede, sistem, indisi zamana karşılık gelen eğim vektörünün anlık tahminini kullanmaktadır. Böyle bir tahminin kullanımı, işlem noktasının hata yüzeyinde bir "rassal-ilerleme" şeklinde sonuçlanmaktadır.

Bununla birlikte, söz konusu maliyet fonksiyonunu enküçüklemek için tasarlanmış bir algoritma, yeterli bir giriş-çıkış örnekleri kümesi ve eğitim için yeterli süre verildiğinde, denetimli öğrenme sistemi örüntü sınıflandırma ve fonksiyon uydurma gibi problemlerde yeterli düzeyde başarılı olmaktadır.



Şekil 3.24: Denetimli Öğrenme Modeli

Denetimli öğrenme algoritmalarına örnek olarak, enküçük-ortalama-kare (least-mean-square – LMS) algoritması ve onun genelleştirilmiş şekli olan, geriye yayılma (backpropagation – BP) algoritmaları verilebilir. LMS algoritması bir nöron için geçerliyken, BP algoritması çok katmanlı ağları kapsamaktadır. BP algoritması adını, ağdaki hata terimlerinin ağ boyunca geriye doğru yayılmasından almaktadır. Doğal olarak, BP algoritması, uygulamada LMS algoritmasından daha güçlüdür. BP algoritması, LMS algoritmasını özel bir durum olarak incelemektedir.

Denetimli öğrenme, "bağlantısız" (off-line) veya "bağlantılı" (on-line) olarak gerçekleştirilebilir. Bağlantısız durumda, denetimli öğrenme sistemini tasarlamak için ayrı bir hesaplama birimi kullanılmaktadır. İstenen performans bir kez sağlandığında, tasarım dondurulmakta ve bu da yapay sinir ağının statik olması anlamına gelmektedir. Diğer taraftan, bağlantılı öğrenmede, öğrenme yöntemi, ayrı bir hesaplama birimine gerek olmaksızın, sadece sistemin içinde uygulanmaktadır. Diğer bir deyişle, yapay sinir ağının dinamik olmasına neden olan gerçek zamanlı bir öğrenme

gerçekleştirilmektedir. Doğal olarak, bağlantılı öğrenmenin, diğerine göre daha çok sayıda şartı (requirement) bulunmaktadır.

Bağlantılı veya bağlantısız olmasından bağımsız olarak, denetimli öğrenmenin dezavantajı, bir öğretici olmadığı durumda, ağı eğitmek için kullanılan örnek kümesi tarafından içerilmeyen belirli durumları için ağın yeni stratejiler öğrenememesidir. Bu kısıtlama, destekleyicili öğrenmenin kullanılmasıyla aşılabilmektedir.

Destekleyicili Öğrenme Modeli (Reinforcement Learning Pardigm)

Destekleyicili öğrenme, destek sinyali (reinforcement signal) olarak adlandırılan bir performans ölçüsünü enbüyükleyecek şekilde tasarlanmış bir deneme ve hata sürecinde, giriş-çıkış haritalama şeklindeki bir bağlantılı öğrenmedir. Destekleyicili öğrenme kavramı, Minsky'nin yapay zekayla ilgili ilk çalışmalarında ve bundan bağımsız olarak, Waltz ve Fu tarafından kontrol teorisinde kullanılmıştır. Bununla birlikte, "destekleyicili" kavramının temel fikri, psikolojideki bazı deneysel çalışmalarda ortaya çıkmıştır. Thonrndike tarafından ortaya atılan klasik etki kanunun düzenlenmesiyle destekleyicili öğrenme aşağıdaki şekilde tanımlanmaktadır:

Eğer öğrenen bir sistem tarafından alınan bir hareketi, yeterli bir durum izliyorsa, sistemin bu belirli hareketi üretmesi eğilimi güçlenmekte veya desteklenmektedir. Diğer durumda, sistemin bu hareketi üretmesi eğilimi zayıflamaktadır.

Destekleyicili öğrenme modeli aşağıda tanımlandığı gibi çağrışımsız veya çağrışımlı tipte olabilir.

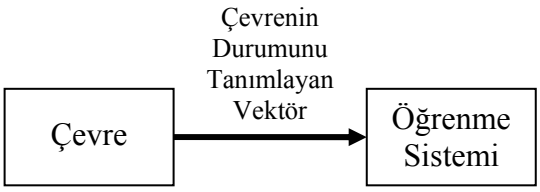
- Çağrışımsız destekleyicili öğrenmede, öğrenen sistem bir eniyi hareket seçmektedir. Böyle bir öğrenme probleminde, destek öğrenen sistemin dışarıdan aldığı tek giriştir. Çağrışımsız öğrenme, genetik algoritmalar altında fonksiyon eniyileme ve stokastik öğrenme özdevinim teorisi (stochastic learning automata theory) konularında ele alınmıştır.
- Çağrışımlı destekleyici öğrenmede, çevre, destek dışında ek bilgi şekilleri sunar ve uyarı-hareketi çağrışımı şeklindeki bir haritalama öğrenilmek zorundadır. Thonrndike tarafından ortaya atılan klasik etki kanununa daha

yakın olan çağrışımlı destekleyicili öğrenme, yapay sinir ağlarında çoğunlukla araştırma olarak incelenmektedir.

Denetimsiz Öğrenme Modeli (Unsupervised Learning Paradigm)

Şekil 3.25'te görüldüğü gibi denetimsiz (self-organized) öğrenmede, öğrenme sürecini gözleyecek dışsal bir öğretici veya kritik, diğer bir deyişle, yapay sinir ağının öğrenmesi gereken fonksiyona ait spesifik örnekler bulunmamaktadır. Bunun yerine genellikle, yapay sinir ağının öğrenmesi gereken ve bağımsız ağ parametrelerinin bu ölçüye göre eniyilendiği, görevden-bağımsız bir sunum kalitesi ölçüsü için sağlama yapılmaktadır. Yapay sinir ağı bir kez giriş verisinin istatistiksel düzenine ayarlandığında, girişin kodlama özelliklerinin içsel gösterimleri şekillendirme özelliğini kazanmakta ve böylece, yeni sınıfları otomatik olarak oluşturmaktadır.

Denetimsiz öğrenmeyi gerçekleştirmek için bir yarışımcı öğrenme kuralı kullanılabilir. Örneğin, biri giriş ve biri de yarışımcı olmak üzere iki katmanlı bir yapay sinir ağı kullanılabilir. Giriş katmanı, uygun veriyi alır. Yarışımcı katman, giriş verisinin içerdiği özelliklere cevap vermek için birbirleriyle yarışan nöronlardan oluşmaktadır. En basit şekliyle, yapay sinit ağı, "kazanan-hepsini-alır" (winner-takes-all) stratejisine uyacak şekilde çalışmaktadır. Böyle bir stratejide, en büyük toplam girişe sahip olan nöron yarışmayı kazanıp açılmakta, diğer bütün nöronlar kapanmaktadırlar.



Şekil 3.25: Denetimsiz Öğrenme Modeli

3.1.3.4 – Bir Nöronlu Doğrusal Ağlar ve LMS Algoritması

Bu kısımda, bir nörondan oluşan ve doğrusallık varsayımıyla çalışan yapay sinir ağları incelenmiştir (Haykin, 1994). Bu sınıftaki yapay sinir ağları üç nedenden dolayı önemlidir. Birincisi, bir nörondan oluşan doğrusal uyarlanır filtre teorisinin, haberleşme, radar, sonar ve biyomedikal mühendislik alanlarında başarıyla kullanılıyor olmasıdır. İkincisi, 1960'lı yıllarda yapay sinir ağları konusundaki çalışmalar sırasında bulunmuş olmasıdır. Sonuncusu da doğrusal uyarlanır filtrelerin, doğrusal olmayan elemanlar içeren çok katmanlı ağların teorik gelişiminde yardımcı olmalarıdır.

Bu kısımda öncelikle, doğrusal eniyi filtreleme problemi özetlenmiştir. Daha sonra LMS algoritması verilmiştir. LMS algoritması aynı zamanda, delta kuralı veya Widrow-Hoff kuralı olarak da bilinmektedir. Bu algoritma doğrusal bir nöron modeli ile çalışmaktadır. LMS algoritmasını çalıştırmak için kullanılan makineye Adaline (Adaptive Linear Element) adı verilmektedir.

Wiener-Hopf Denklemleri

Şekil 3.26'da görüldüğü gibi uzayda farklı noktalarda bulunan, p sayıda sensörden oluşan bir küme olduğu varsayılsın. $x_1, x_2, \ldots, x_p$, bu sensörler tarafından üretilen sinyaller olsun. Bu sinyaller, $w_1, w_2, \ldots, w_p$ ağırlıklarına uygulansın. Bu ağırlıklı sinyaller, y çıkış sinyalini üretmek için toplanmaktadırlar. Burada yapılmak istenen, y sistem çıkışı ile d istenen çıkışı arasındaki farkı enküçükleyecek şekilde eniyi $w_1, w_2, \ldots, w_p$ ağırlıklarının belirlenmesidir. Bu problemin çözümü için Wiener-Hopf denklemleri kullanılmaktadır.

Şekil 3.26'da tanımlanan sistem bir uzaysal filtre (spatial filter) olarak görülebilir. Filtrenin giriş-çıkış ilişkisi,

$$y = \sum_{k=1}^{p} w_k x_k \tag{3.43}$$

toplamıyla tanımlanmaktadır. d istenen çıkışı göstermek üzere, hata sinyali,

$$e = d - y \tag{3.44}$$

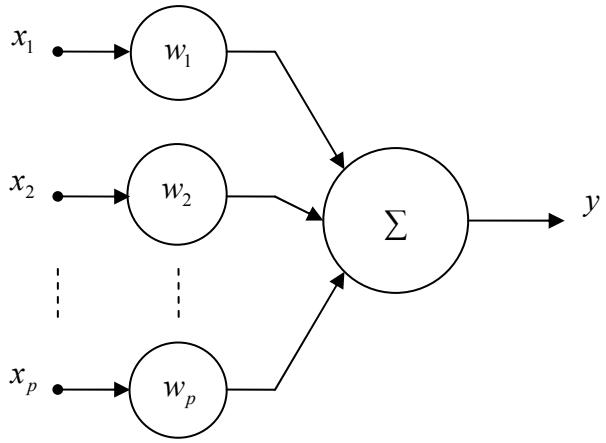
şeklinde tanımlanabilir.

Performans ölçüsü veya maliyet fonksiyonu olarak, ortalama-karesel-hata,

$$J = \frac{1}{2} E\left[e^2\right] \tag{3.45}$$

şeklinde tanımlanmaktadır. Burada, E beklenen değeri göstermektedir. $1/2$ ifadesi, daha sonra görüleceği gibi gösterim kolaylığı sağlamak amacıyla kullanılmıştır. Bu durumda, doğrusal filtreleme problemi aşağıdaki şekilde tanımlanabilir:

J ortalama-karesel hatayı enküçükleyecek, eniyi $w_1, w_2, \ldots, w_p$ ağırlıklarının belirlenmesi.



Şekil 3.26: Uzaysal Filtre

Denklem 3.43 ve 3.44'ü, 3.45'te yerine yazarak,

$$J = \frac{1}{2} E\left[d^2\right] - E\left[\sum_{k=1}^{p} w_k x_k d\right] + \frac{1}{2} E\left[\sum_{j=1}^{p}\sum_{k=1}^{p} w_j w_k x_j x_k\right] \tag{3.46}$$

elde edilmektedir. Burada, toplamın karesini göstermek için çift toplam kullanılmaktadır. Beklenen değer doğrusal bir işlem olduğundan, bununla toplamın yerleri değiştirilebilir. Bu durumda, denklem,

$$J = \frac{1}{2}E\left[d^2\right] - \sum w_k E\left[x_k d\right] + \frac{1}{2}\sum_{j=1}^{p}\sum_{k=1}^{p} w_j w_k E\left[x_j x_k\right] \tag{3.47}$$

şeklinde yeniden yazılabilir.

Bu aşamada, denklemdeki üç farklı beklenen değerden söz edilebilir:

- $E\left[d^2\right]$ Beklenen Değeri

d istenen cevabının ortalama-kare değeri aşağıdaki şekilde tanımlansın:

$$r_d = E\left[d^2\right] \tag{3.48}$$

- $E\left[dx_k\right]$ Beklenen Değeri

d istenen cevabı ile x_k sensör sinyali arasındaki çapraz-korelasyon fonksiyonu aşağıdaki şekilde tanımlansın:

$$r_{dx}(k) = E\left[dx_k\right],\ k = 1,2,\ldots,p \tag{3.49}$$

- $E\left[x_j x_k\right]$ Beklenen Değeri

Sensör sinyalleri kümesindeki oto-korelasyon fonksiyonu aşağıdaki gibi tanımlansın:

$$r_x(j,k) = E\left[x_j x_k\right],\ j,k = 1,2,\ldots,p \tag{3.50}$$

Bu tanımlar sonucunda, Denklem 3.47,

$$J = \frac{1}{2}r_d - \sum_{k=1}^{p} w_k r_{dx}(k) + \frac{1}{2}\sum_{j=1}^{p}\sum_{k=1}^{p} w_j w_k r_x(j,k) \tag{3.51}$$

şeklinde yazılabilir. J maliyet fonksiyonunun, $w_1, w_2, \ldots, w_p$ ağrılıklarına (bağımsız parametreler) göre grafiği filtrenin hata-performans yüzeyi veya kısaca hata yüzeyi olarak adlandırılır. Hata yüzeyinin bir genel enküçük noktası bulunmaktadır. Uzaysal filtrenin ortalama-karesel hatasının enküçük $J_{\min}$ değerini aldığı nokta bu noktadır.

Bu eniyi koşulu belirlemek için J maliyet fonksiyonunun w_k ağırlığına göre türevi alınır ve sonuç tüm k değerleri için sıfıra eşitlenir. J'nin w_k'ya göre türevi hata yüzeyinin, söz konusu ağırlığa göre eğimi olarak adlandırılır. Bu eğim $\nabla_{w_k} J$ ile gösterilirse,

$$\nabla_{w_k} J = \frac{\partial J}{\partial w_k},\ k = 1,2,\ldots,p \tag{3.52}$$

şeklinde tanımlanabilir. Denklem 3.51'in w_k'ya göre türevi alınırsa,

$$\nabla_{w_k} J = -r_{dx}(k) + \sum_{j=1}^{p} w_j r_x(j,k) \tag{3.53}$$

elde edilir. Böylece, filtre için eniyi şart,

$$\nabla_{w_k} = 0,\ k = 1,2,\ldots,p \tag{3.54}$$

şeklinde tanımlanmaktadır. w_{ok}, eniyi w_k değerini göstersin. Bu durumda, Denklem 3.53'ten uzaysal filtrenin eniyi ağırlık değerleri,

$$\sum_{j=1}^{p} w_{oj} r_x(j,k) = r_{dx}(k),\ k = 1,2,\ldots,p \tag{3.55}$$

denklem kümesiyle belirlenmektedir. Bu denklem sistemi Wiener-Hopf denklemleri olarak bilinmekte ve ağırlıkları Wiener-Hopf denklemlerini sağlayan filtre de Wiener filtresi olarak adlandırılmaktadır.

Enbüyük Eğim Yöntemi (Method of Steepest Descent)

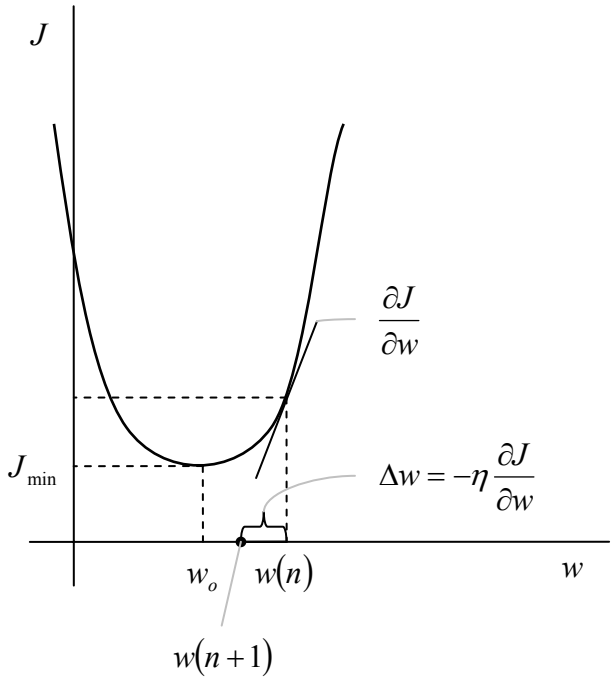
Eniyi ağırlık değerlerini bulmak için, Wiener-Hopf denklemlerini çözmek amacıyla $k = 1,2,\ldots,p$ için $r_x(j,k)$ otokorelasyon fonksiyonunun farklı değerlerinden oluşan bir $p \times p$ matrisinin tersinin bulunması gerekmektedir. Enbüyük eğim yöntemi kullanılarak bu matrisin tersinin alınmasına gerek kalmayabilir. Bu yönteme göre, filtrenin ağırlıkları zamanla-değişen bir şekildedir ve değerleri, hata yüzeyi boyunca

eniyi çözüme doğru hareket edilen iteratif bir yönteme göre ayarlanmaktadır. Yöntem, sürekli olarak hata yüzeyinin en alttaki noktasını aramaktadır. Burada, ardışık ayarlamaları enbüyük eğim yönünde yapmak mantıklıdır. Bu, elemanları, $k = 1,2,\ldots,p$ için $\nabla_{w_k} J$ ile tanımlanan eğim vektörünün zıt yönüdür. Böyle bir ayarlama Şekil 3.27'de görülmektedir.

$w_k(n)$, $n.$ yinelemede enbüyük eğim yöntemiyle hesaplanmış olan w_k ağırlığını göstersin.

$$\nabla_{w_k} J(n) = -r_{dx}(k) + \sum_{j=1}^{p} w_j(n) r_x(j,k) \tag{3.56}$$

Denklem 3.56'dan görüldüğü gibi bu ağırlığa göre hata yüzeyinin eğimi, zamanla-değişen şekildedir.



Şekil 3.27: Enbüyük Eğim Yöntemi

Burada, n indeksinin iterasyon numarasına, j ve k indislerinin uzaydaki farklı sensörlerin konumlarına karşılık geldiği görülmektedir. En büyük eğim yöntemine göre, $w_k(n)$ ağırlığına uygulanan ayarlama,

$$\Delta w_k(n) = -\eta \nabla_{w_k} J(n),\ k = 1,2,\ldots,p \tag{3.57}$$

denklemiyle verilmektedir. Burada, η öğrenme-oranı parametresi adı verilen pozitif bir sabittir.

n. iterasyondaki $w_k(n)$ k. ağırlığının $n+1$. iterasyondaki güncelleştirilmiş değeri,

$$w_k(n+1) = w_k(n) + \Delta w_k(n) = w_k(n) - \eta \nabla_{w_k} J(n),\ k = 1,2,\ldots,p \tag{3.58}$$

şeklinde verilmektedir. Böylece, enbüyük eğim yöntemi aşağıdaki şekilde tanımlanabilir:

Bir Wiener filtresindeki k. ağırlığın güncelleştirilmiş değeri, ağırlığın eski değeri ile hata yüzeyinin söz konusu ağırlığa göre eğiminin negatif değerli bir oranının toplanmasıyla bulunmaktadır.

Denklem 3.56'yı Denklem 3.58'de yerine yazarak enbüyük eğim yöntemi korelasyon fonksiyonları $r_x(j,k)$ ve $r_{dx}(k)$ cinsinden aşağıdaki şekilde yazılabilir:

$$w_k(n+1) = w_k(n) + \eta \left[r_{dx}(k) - \sum_{j=1}^{M} w_j(n) r_x(j,k) \right],\ k = 1,2,\ldots,p \tag{3.59}$$

Enbüyük eğim yönteminin türetilmesinde hiçbir varsayımda bulunulmamıştır. Bu anlamda, kesin bir yöntemdir. Burada geliştirilen yöntemde,

$$J(n) = \frac{1}{2} E\left[e^2(n)\right] \tag{3.60}$$

denklemi ile verilen ortalama-karesel-hata kullanılmıştır. Bu maliyet fonksiyonu, belirli bir n zamanında, aynı şekilde tasarlanmış, ancak aynı popülasyondan gelen farklı

girişler içeren uzaysal filtreler topluluğu üzerinde, genel bir ortalamadır. Enbüyük eğim yöntemi hata kareleri toplamı enküçüklenerek de geliştirilebilir:

$$\varepsilon_t(n) = \sum_{i=1}^{n} \varepsilon(i) = \frac{1}{2}\sum_{i=1}^{n} e^2(i) \tag{3.61}$$

Burada, toplam algoritmanın tüm iterasyonları üzerinde, ancak uzaysal filtrenin belirli bir şekli için alınmaktadır. Bu ikinci yaklaşım, Denklem 3.59'daki ile aynı sonucu vermekte, ancak korelasyon fonksiyonlarını farklı şekilde yorumlamaktadır. Burada, otokorelasyon fonksiyonu r_x ve çapraz-korelasyon fonksiyonu r_{dx} topluluk ortalamaları değil, zaman ortalamaları olmaktadırlar.

Enbüyük eğim yönteminin çalışmasında, η öğrenme-oranı parametresinin seçimi önemlidir. Yöntemin bir diğer kısıtı da $r_{dx}(k)$ ve $r_x(j,k)$ korelasyon fonksiyonlarının bilinmesini gerektirmesidir. Filtrenin bilinmeyen bir ortamda çalışması durumunda bu fonksiyonları kullanmak mümkün olmamaktadır. Bu durumda, bunların yerine tahminleri kullanılmak durumundadır. LMS algoritması bu tahminleri sağlamada etkili bir yöntemdir.

LMS algoritması, $r_{dx}(k)$ ve $r_x(j,k)$ korelasyon fonksiyonlarının anlık tahminlerinin kullanılmasına dayanmaktadır. Bu tahminler doğrudan,

$$\hat{r}_x(j,k;n) = x_j(n)x_k(n) \tag{3.62}$$

ve

$$\hat{r}_{dx}(k;n) = x_k(n)d(n) \tag{3.63}$$

denklemlerinden elde edilmektedir. r_x ve r_{dx} yerine $\hat{r}_x$ ve $\hat{r}_{dx}$ kullanılmasının nedeni, buradaki ifadelerin tahmin olduğunun gösterilmesi içindir. Denklem 3.62 ve 3.63'teki tanımlar, tüm giriş sinyalleri ve istenen çıkışların zamanla değişen şekilde, durağan olmayan bir ortamı kapsaması için genelleştirilmişlerdir. Böylece, Denklem 3.59'da, $r_x(j,k)$ ve $r_{dx}(k)$ yerine, $\hat{r}_x(j,k;n)$ ve $\hat{r}_{dx}(k;n)$ yazılarak

$$\hat{w}_k(n+1) = \hat{w}_k(n) + \eta\left[x_k(n)d(n) - \sum_{j=1}^{p} \hat{w}_j(n)x_j(n)x_k(n) \right]$$

$$\hat{w}_k(n+1) = \hat{w}_k(n) + \eta\left[d(n) - \sum_{j=1}^{p} \hat{w}_j(n)x_j(n) \right]x_k(n) \tag{3.64}$$

$$\hat{w}_k(n+1) = \hat{w}_k(n) + \eta\left[d(n) - y(n)\right]x_k(n),\ k = 1,2,\ldots,p$$

elde edilir. Burada, $y(n)$,

$$y(n) = \sum_{j=1}^{p} \hat{w}_j(n)x_j(n) \tag{3.65}$$

şeklinde ifade edilen, uzaysal filtrenin *n.* iterasyondaki çıkışıdır. Denklem 3.64'te, $w_k(n)$ yerine $\hat{w}_k(n)$ yazılmasıyla uzaysal filtrenin tahmini ağırlıklarının kullanıldığı vurgulanmaktadır.

Şekil 3.28, LMS algoritmasının çalışmasını özetlemektedir. Şekilde, algoritmanın başlangıcı için bütün ağırlıkların sıfıra eşitlendiği görülmektedir.

"Bilinen" bir ortama uygulanan enbüyük eğim yöntemi, $w_1(n), w_2(n), \ldots, w_p(n)$ ağırlıklarından oluşan $\mathbf{w}(n)$ vektörü, bir $\mathbf{w}(0)$ başlangıç değerini almakta ve daha sonra, hata yüzeyinde $\mathbf{w}_o$ eniyi çözümüne doğru ilerlemektedir. "Bilinmeyen" bir ortama uygulanan LMS algoritmasında ise $\mathbf{w}(n)$ ağırlık vektörünün bir tahmini olan $\hat{\mathbf{w}}(n)$ ağırlık vektörü, rassal bir şekilde ilerlemektedir. Bu yüzden, LMS algoritması "stokastik eğim algoritması" olarak da bilinmektedir.

Adım 1: Başlangıç. Aşağıdaki atamayı yap.

$\hat{w}_k(1) = 0$, $k = 1,2,\ldots,p$ için

Adım 2: Filtreleme. $n = 1,2,\ldots$ için aşağıdaki hesaplamaları yap.

$$y(n) = \sum_{j=1}^{p} \hat{w}_j(n)x_j(n)$$

$$e(n) = d(n) - y(n)$$

$\hat{w}_k(n+1) = w_k(n) + \eta e(n)x_k(n)$, $k = 1,2,\ldots,p$ için

Şekil 3.28: LMS Algoritmasının Çalışması

Enbüyük eğim yöntemi ve LMS algoritması arasındaki diğer bir fark da hata hesaplamalarındadır. Enbüyük eğim yöntemi, n. iterasyonda, $J(n)$ ortalama-karesel hatasını enküçüklemektedir. Bu maliyet fonksiyonu bir topluluk ortalaması vermektedir ve bunun etkisi, n değerinin artmasıyla daha doğru sonuçlar veren "kesin" bir eğim vektörünün verilmesi şeklindedir. Öte yandan, LMS algoritması $J(n)$ maliyet fonksiyonunun anlık bir tahminini enküçüklemektedir. Sonuç olarak, LMS algoritması "rassal" olup, doğruluğu, n değerinin artmasıyla "ortalama" olarak artmaktadır.

Enbüyük eğim yöntemi ve LMS algoritması arasındaki temel farklılıklardan birisi de zaman açısından ele alınabilir. Enbüyük eğim yöntemi, başlangıçtan n. iterasyona kadar olan, $\varepsilon_t(n)$ hata kareleri toplamını enküçüklemektedir. Sonuç olarak, r_x ve r_{dx} korelasyon fonksiyonlarının tahminleri için bilgi depolamayı gerektirmektedir. LMS algoritması ise $\varepsilon(n)$ anlık hata karesini enküçüklemektedir. Filtrenin o andaki ağırlıkları dışında herhangi bir bilgi depolamaya gerek olmamaktadır.

LMS algoritmasının durağan veya durağan olmayan ortamlarda çalışabilmesi de önemlidir. Durağan olmayan bir ortam, istatistiklerin zamanla değiştiği bir ortamı anlatılmaktadır. Bu yüzden, LMS algoritması sadece hata yüzeyinin enküçük noktasını aramayıp, aynı zamanda onu izlemektedir. Bu anlamda, η öğrenme-oranının daha küçük değerleri algoritmanın daha iyi bir izleme davranışı göstermesini sağlamakla birikte, uyarlama hızının daha yavaş olmasına neden olmaktadır.

3.1.3.5 – MLP'ler ve BP Algoritması

Bu kısımda, MLP'ler ve bunların eğitiminde çok sık kullanılan BP algoritması açıklanmmıştır (Haykin, 1994).

MLP'ler, kaynak düğümlerinden oluşan bir giriş katmanı, hesaplama düğümlerinden oluşan bir veya daha fazla sayıda gizli katman ve hesaplama düğümlerinden oluşan bir çıkış katmanı içermektedirler. Giriş sinyali, ağda, ileri yönde, katmanlar boyunca, yayılmaktadır.

MLP'ler, BP algoritması kullanılarak (denetimli öğrenme) eğitilip, çeşitli zor problemlerin çözümünde başarıyla kullanılmaktadırlar. Bu algoritma, hata-düzeltme

öğrenme algoritmasına göre çalışmaktadır ve önceki kısımda açıklanan LMS algoritmasının genel halidir.

Temel olarak, BP algoritması sürecinde, ağ katmanları boyunca iki geçişten söz edilebilir: (i) ileri yönde geçiş (forward pass) ve (ii) geri yönde geçiş (backward pass). İleri yönde geçişte, ağın giriş düğümlerine bir giriş vektörü verilir ve bunun etkisi ağda katmanlar boyunca yayılır. Sonuç olarak, ağın gerçek çıkışı olan bir çıkışlar kümesi üretilir. İleri yönde geçiş sırasında, ağın sinaptik ağırlıkları sabittirler. Öte yandan, geri yönde geçiş sırasında, sinaptik ağırlıklar hata-düzeltme kuralına göre ayarlanırlar. Ağın gerçek cevabı istenen cevaptan çıkarılarak, bir hata sinyali üretilir. Daha sonra bu hata sinyali, sinaptik bağlantılara karşı yönde, ağ boyunca geriye doğru yayılır. Sinaptik ağırlıklar, ağın gerçek cevabını, istenen cevaba yaklaştırmak için ayarlanırlar.

Bir MLP aşağıdaki ayırt edici karakteristiklere sahiptir:

- Ağdaki her nöron modeli çıkışta doğrusal olmayan bir yapı içerir. Burada vurgulanması gereken nokta, doğrusal olmayan yapının sürekli olmasıdır. Bu gereksinimi karşılayan çok kullanılan bir doğrusal olmayan yapı şekli,

- $y_j = \dfrac{1}{1+\exp(-v_j)}$ sigmoid fonksiyonuyla tanımlanmaktadır. Burada, v_j, j nöronunun net içsel aktivite düzeyi ve y_j de nöron çıkışıdır. Doğrusal olmayan yapıların bulunması önemlidir, çünkü diğer durumda, ağın giriş-çıkış ilişkisi bir-katmanlı bir perceptron şekline dönüşebilmektedir. Bu fonksiyonun kullanımı, gerçek nöronlara benzerlik gösterdiği için biyolojik açıdan da önemlidir.

- Ağ, giriş ve çıkışın parçası olmayan ve gizli nöronlardan oluşan, bir veya daha fazla sayıda katman içermektedir. Bu gizli nöronlar, giriş vektöründen daha anlamlı özellikleri çıkararak, daha karmaşık görevlerin öğrenilmelerini sağlamaktadırlar.

- Ağ, sinapsisleri tarafından belirlenen yüksek bir bağlantı yapısı göstermektedir. Ağın bağlantı yapısında meydana gelen bir değişiklik, sinaptik bağlantılar veya ağırlıklarının değiştirilmesini gerektirmektedir.

MLP'lerin hesaplama gücünü sağlayan, eğitim ile öğrenme yeteneği, yukarıda belirtilen karakteristiklerin birleşmesinden oluşmaktadır. Bununla birlikte, aynı karakteristikler, ağın davranışı hakkındaki bilgi eksikliğinden de sorumludurlar. Bunun nedenelerinden birincisi, doğrusal olmamanın dağıtılmış yapısı ve ağın yüksek bağlantı yapısının, MLP'lerin teorik olarak incelenmesini zorlaştırmasıdır. İkincisi, gizli nöronların kullanılmasının, öğrenme sürecini görmeyi zorlaştırmasıdır. Öğrenme sürecinin, giriş örüntüsündeki hangi özelliklerinin gizli nöronlar tarafından gösterileceğine karar vereceği bir yapı daha belirleyici olabilir. Bu durumda, öğrenme süreci daha zor olmaktadır, çünkü arama, çok daha büyük bir olası fonksiyonlar uzayında yapılacak ve giriş örüntüsünün alternatif gösterimleri arasında seçim yapmak gerekecektir.

MLP'lerin eğitiminde, hesaplama açısından etkin bir yöntem olması nedeniyle, BP algoritmasının geliştirilmesi, yapay sinir ağlarında önemli bir adımdır. Çözülmesi mümkün olan tüm problemler için çözüm sunduğu söylenemese bile, çok-katmanlı makinelerin öğrenmesi hakkında olumlu düşüncelerin gelişmesini sağladığı görülmektedir.

Bir MLP'de iki tür sinyal bulunmaktadır:

- Fonksiyon Sinyalleri. Fonksiyon sinyali ağın girişine gelen ve nöronlar boyunca ileri yönde yayılan bir giriş sinyalidir. Böyle bir sinyale iki nedenden dolayı "fonksiyon sinyali" denilmektedir. Birincisi, bu sinyalin ağ çıkışında yararlı bir fonksiyonu gerçekleştirdiği düşünülmektedir. İkincisi, yapay sinir ağında bir fonksiyon sinyalinin geçtiği her nöronda, sinyal, bu nörona uygulanan girişlerin ve ilgili ağırlıkların bir fonksiyonu olarak hesaplanmaktadır.
- Hata Sinyalleri. Hata sinyali ağın bir çıkış nöronundan başlayarak, katmanlar boyunca geri yönde yayılmaktadır. Bu sinyale "hata sinyali" denilmesinin nedeni, ağdaki her nöron için hataya-bağlı bir fonksiyon olarak hesaplanmasıdır.

Çıkış nöronları (hesaplama düğümleri), ağın çıkış katmanını oluşturmaktadırlar. Diğer nöronlar, (hesaplama düğümleri) de ağın gizli katmanlarını oluşturmaktadırlar. Bu gizli birimler, ağ giriş veya çıkışının parçası değildirler. Birinci gizli katman girişlerini, kaynak düğümlerinden oluşan giriş katmanından almaktadır. Birinci gizli katmanın çıkışları da sonraki gizli katmana uygulanmakta ve bu işlem ağ boyunca benzer şekilde devam etmektedir.

Bir MLP'deki her gizli nöron veya çıkış nöronu iki hesaplama yapmak için tasarlanmıştır:

- Giriş sinyallerinin ve ilgili ağırlıkların doğrusal olmayan sürekli bir fonksiyonu olarak ifade edilen, nöron çıkışındaki fonksiyon sinyalinin hesaplanması.
- Geri yönde geçiş için gerekli olan eğim vektörünün anlık bir tahmininin hesaplanması.

BP algoritmasının matematiksel gösteriminde kullanılan notasyonlar aşağıdaki şekildedir:

- i, j ve k, ağ boyunca soldan sağa doğru yayılan farklı nöronları göstermektedir. j nöronu i nöronunun sağındaki katmanda ve k nöronu j nöronunun sağındaki katmanda bulunmaktadır.
- n iterasyonu, ağa verilen $n.$ eğitim örneğini göstermektedir.
- $\varepsilon(n)$, $n.$ iterasyondaki hata kareleri toplamını göstermektedir. Tüm n değerleri için $\varepsilon(n)$ ortalaması, ε_{av} ortalama karesel hatasını göstermektedir.
- $e_j(n)$, $n.$ iterasyonda, j nöronu çıkışındaki hata sinyalini göstermektedir.
- $d_j(n)$, $n.$ iterasyonda, j nöronu çıkışındaki istenen cevabı göstermektedir.
- $y_j(n)$, $n.$ iterasyonda, j nöronu çıkışındaki fonksiyon sinyalini göstermektedir.

- $w_{ji}(n)$, n. iterasyonda, i nöronu çıkışını, j nöronu girişine bağlayan sinaptik ağırlığı göstermektedir. Bu ağırlığa n. iterasyonda uygulanan düzeltme $\Delta w_{ji}(n)$ ile gösterilmektedir.
- $v_j(n)$, n. iterasyonda, j nöronunun net içsel aktivite düzeyini göstermektedir. Bu değer, j nöronundaki doğrusal olmayan yapıya uygulanan sinyali oluşturmaktadır.
- $\varphi_j(\cdot)$, j nöronundaki giriş-çıkış arasındaki doğrusal olmayan yapının fonksiyonel ilişkisini tanımlayan aktivasyon fonksiyonunu göstermektedir.
- θ_j, j nöronuna uygulanan eşik değerini göstermektedir. Eşik değerinin etkisi, -1 değerine eşit olan bir $w_{j0} = \theta_j$ ağırlık sinapsi ile gösterilmektedir.
- $x_i(n)$, giriş vektörünün, n. iterasyondaki, i. elemanını göstermektedir.
- $o_k(n)$, çıkış vektörünün, n. iterasyondaki, k. elemanını göstermektedir.
- η, öğrenme-oranını göstermektedir.

n. iterasyonda, j nöronu çıkışındaki hata sinyali,

$$e_j(n) = d_j(n) - y_j(n),\ j \text{ nöronu bir çıkış düğümü} \tag{3.66}$$

şeklinde tanımlanmaktadır. j nöronu için karesel hatanın anlık değeri $\frac{1}{2}e_j^2(n)$ olarak tanımlanmaktadır. Benzer şekilde, $\varepsilon(n)$ karesel hataların toplamının anlık değeri, çıkış katmanındaki tüm nöronlar için $\frac{1}{2}e_j^2(n)$ değerlerinin toplanmasıyla elde edilmektedir. Bu nöronlar, kendileri için hata sinyalleri hesaplanabilen "görünür" nöronlardır. Böylece, ağın anlık hata kareleri toplamı,

$$\varepsilon(n) = \frac{1}{2}\sum_{j \in C} e_j^2(n) \tag{3.67}$$

olarak yazılabilir. Burada, C ağın çıkış katmanındaki tüm nöronları içermektedir. N, eğitim kümesindeki toplam örnek sayısını göstersin. Bu durumda, ortalama karesel hata, $\varepsilon(n)$ değerinin tüm n değerleri için toplanması ve N değeri ile normalize edilmesiyle elde edilir:

$$\varepsilon_{av} = \frac{1}{N}\sum_{n=1}^{N} \varepsilon(n) \tag{3.68}$$

Anlık hata kareleri toplamı $\varepsilon(n)$ ve buna bağlı olarak, ortalama karesel hata ε_{av}, ağın bağımsız parametrelerinin (sinaptik ağırlıklar ve eşikler gibi) bir fonksiyonudur. Verilen bir eğitim kümesi için ε_{av}, eğitim kümesi öğrenme performansının ölçüsü olarak maliyet fonksiyonunu göstermektedir. Öğrenme sürecinin amacı, ε_{av} değerini enküçükleyecek şekilde ağ parametrelerinin ayarlanmasıdır. Bu enküçüklemenin yapılması için LMS algoritmasındakine benzer bir yaklaşım uygulanmaktadır. Burada, ağırlıkların örnekler boyunca güncellendiği basit bir eğitim yöntemi göz önüne alınmmıştır. Ağırlık ayarlamaları, ağa verilen her örnek için hesaplanan ağırlıklara göre yapılmaktadır. Bu yüzden, bu anlık ağırlık değişimlerinin aritmetik ortalaması gerçek değişimlerin bir tahmini olmaktadır.

Solundaki nöron katmanı tarafından üretilen fonksiyon sinyalleri kümesini giriş olarak alan bir j nöronu ele alınsın. j nöronunun net içsel aktivite düzeyi,

$$v_j(n) = \sum_{i=0}^{p} w_{ji}(n) y_i(n) \tag{3.69}$$

olarak tanımlanmaktadır. Burada, p, j nöronuna uygulanan eşik değeri dışındaki toplam giriş sayısıdır. $y_0(n) = -1$ sabit ağırlığına karşılık gelen $w_{j0}(n)$ sinaptik ağırlığı, j nöronuna uygulanan $\theta_j(n)$ eşik değerine eşittir. Böylece, *n.* iterasyonda, j nöronuna uygulanan fonksiyon sinyali,

$$y_j(n) = \varphi_j(v_j(n)) \tag{3.70}$$

şeklindedir. LMS algoritmasına benzer şekilde, BP algoritması da $w_{ji}(n)$ sinaptik ağırlığına, $\partial\varepsilon(n)/\partial w_{ji}(n)$ anlık eğimiyle orantılı $\Delta w_{ji}(n)$ düzeltme değerini uygulamaktadır. Zincir kuralına göre, bu eğim,

$$\frac{\partial\varepsilon(n)}{\partial w_{ji}(n)} = \frac{\partial\varepsilon(n)}{\partial e_j(n)}\frac{\partial e_j(n)}{\partial y_j(n)}\frac{\partial y_j(n)}{\partial v_j(n)}\frac{\partial v_j(n)}{\partial w_{ji}(n)} \tag{3.71}$$

şeklinde yazılabilir. $\partial\varepsilon(n)/\partial w_{ji}(n)$ eğimi, ağırlık uzayında w_{ji} ağırlığı için aramanın yönünü belirleyen bir duyarlılık faktörüdür.

Denklem 3.67'nin $e_j(n)$ 'ye göre türevi alınırsa,

$$\frac{\partial\varepsilon(n)}{\partial e_j(n)} = e_j(n) \tag{3.72}$$

elde edilir. Denklem 3.66'nın $y_j(n)$ 'ye göre türevi alınırsa,

$$\frac{\partial e_j(n)}{\partial y_j(n)} = -1 \tag{3.73}$$

elde edilir. Denklem 3.70'in $v_j(n)$ 'ye göre türevi alınırsa,

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \varphi_j'(v_j(n)) \tag{3.74}$$

elde edilir. Son olarak, Denklem 3.69'un $w_{ji}(n)$ 'ye göre türevi alınırsa,

$$\frac{\partial v_j(n)}{\partial w_{ji}(n)} = y_i(n) \tag{3.75}$$

elde edilir. Bu sonuçların Denklem 3.71'de yerine konulmasıyla,

$$\frac{\partial \varepsilon(n)}{\partial w_{ji}(n)} = -e_j(n)\varphi'_j(v_j(n))y_i(n) \tag{3.76}$$

elde edilir. $w_{ji}(n)$ sinaptik ağırlığına uygulanan $\Delta w_{ji}(n)$ düzeltme değeri,

$$\Delta w_{ji}(n) = -\eta \frac{\partial \varepsilon(n)}{\partial w_{ji}(n)} \tag{3.77}$$

şeklindeki delta kuralıyla tanımlanmaktadır. Burada, η öğrenme oranını belirleyen bir sabittir ve BP algoritmasının öğrenme oranı olarak adlandırılır. Denklem 3.77'de kullanılan eksi işareti, ağırlık uzayındaki eğim düşmesini göstermektedir. Denklem 3.76'nın 3.77'de yerine yazılmasıyla,

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n) \tag{3.78}$$

elde edilir. Burada, yerel eğim,

$$\delta_j(n) = -\frac{\partial \varepsilon(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} = e_j(n)\varphi'_j(v_j(n)) \tag{3.79}$$

şeklinde tanımlanmaktadır. Denkleme göre, j nöronu için yerel eğim, $e_j(n)$ hata sinyali ile $\varphi'_j(v_j(n))$ aktivasyon fonksiyonunun türevinin çarpımıdır.

Denklem 3.78 ve 3.79'da $\Delta w_{ji}(n)$ ağırlık ayarlamasının hesaplanmasında, j nöronu çıkışındaki $e_j(n)$ hata sinyalinin anahtar faktör olduğu görülmektedir. Bu anlamda, j nöronunun konumuna bağlı olarak iki farklı durum belirlenebilir. Birincisi, j nöronunun bir çıkış düğümü olmasıdır. Bu durumun incelenmesi kolaydır, çünkü ağdaki her çıkış düğümüne ait istenen bir cevap vardır ve ilgili hata sinyali doğrudan hesaplanabilir. İkincisi, j nöronunun gizli düğüm olmasıdır. Gizli nöronlar doğrudan erişilebilir olmamakla birlikte, ağ çıkışında oluşan herhangi bir hatanın sorumluluğunu paylaşmaktadırlar. Buradaki problem, gizli nöronların bu hata sorumluluğu paylaşımlarının nasıl olacağının belirlenmesidir ve hata sinyallerinin ağ boyunca geri yönde yayılmasıyla çözülmektedir.

- Durum I: j Nöronunun Çıkış Düğümü Olması

j nöronu ağın çıkış katmanında yer aldığında, kendisine ait bir istenen cevabı bulunmaktadır. Böylece, $e_j(n)$ hata sinyalini hesaplamak için Denklem 3.66 kullanılabilir. $e_j(n)$ belirlendiğinde, Denklem 3.79 kullanılarak $\delta_j(n)$ yerel eğimi kolaylıkla hesaplanabilir.

- Durum II: j Nöronunun Gizli Düğüm Olması

j nöronu gizli bir katmandaysa, bu nöron için belirli bir istenen cevap bulunmamaktadır. Dolayısıyla, gizli bir nöron için hata sinyali, bu nöronun doğrudan bağlı olduğu nöronların hata sinyalleri cinsinden belirlenmek durumundadır. Bu noktada, BP algoritması karmaşık hale gelmektedir. j nöronunun gizli nöron olduğu durum göz önüne alınsın. Denklem 3.79'a göre, j gizli nöronunun $\delta_j(n)$ yerel eğimi,

$$\delta_j(n) = -\frac{\partial \varepsilon(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} = -\frac{\partial \varepsilon(n)}{\partial y_j(n)} \varphi_j'(v_j(n)),\ j \text{ nöronu gizli düğüm} \tag{3.80}$$

şeklinde yeniden tanımlanabilir. Burada, eşitliğin ikinci kısmında Denklem 3.74 kullanılmıştır. $\partial \varepsilon(n) / \partial y_j(n)$ kısmi türevi,

$$\varepsilon(n) = \frac{1}{2} \sum_{k \in C} e_k^2(n),\ k \text{ nöronu çıkış düğümü} \tag{3.81}$$

şeklinde hesaplanabilir. Bu denklem Denklem 3.67'de j indisi yerine k indisi yazılarak elde edilmiştir. Bu gösterim, Durum II'de gizli nörona karşılık gelen j indisinin karışıklığa neden olmasını önlemek için kullanılmıştır. Denklem 3.81'in $y_j(n)$ fonksiyon sinyaline göre türevi alınırsa,

$$\frac{\partial \varepsilon(n)}{\partial y_j(n)} = \sum_k e_k \frac{\partial e_k(n)}{\partial y_j(n)} \tag{3.82}$$

elde edilir. Daha sonra, $\partial e_k(n)/\partial y_j(n)$ kısmi türevi için zincir kuralı kullanılarak, Denklem 3.82 aşağıdaki şekilde yazılabilir:

$$\frac{\partial \varepsilon(n)}{\partial y_j(n)} = \sum e_k(n) \frac{\partial e_k(n)}{\partial v_k(n)} \frac{\partial v_k(n)}{\partial y_j(n)} \tag{3.83}$$

Bununla birlikte,

$$e_k(n) = d_k(n) - y_k(n) = d_k(n) - \varphi_k(v_k(n)),\ k \text{ nöronu çıkış düğümü} \tag{3.84}$$

olduğu görülmektedir. Böylece,

$$\frac{\partial e_k(n)}{\partial v_k(n)} = -\varphi_k'(v_k(n)) \tag{3.85}$$

olmaktadır. Ayrıca, k nöronu için net içsel aktivite düzeyinin,

$$v_k(n) = \sum_{j=0}^{q} w_{kj}(n) y_j(n) \tag{3.86}$$

olduğu görülmektedir. Burada, q, k nöronuna uygulanan eşik değeri dışındaki toplam giriş sayısıdır. $y_0 = -1$ sabit ağırlığına karşılık gelen $w_{k0}(n)$ sinaptik ağırlığı, k nöronuna uygulanan $\theta_k(n)$ eşik değerine eşittir. Denklem 3.86'nın $y_j(n)$'ye göre türevi alınırsa,

$$\frac{\partial v_k(n)}{\partial y_j(n)} = w_{kj}(n) \tag{3.87}$$

elde edilir. Böylece, Denklem 3.85 ve 3.87, 3.83'te yerine yazılarak, istenen kısmi türev aşağıdaki şekilde elde edilmektedir:

$$\frac{\partial \varepsilon(n)}{\partial y_j(n)} = -\sum_k e_k(n) \varphi_k'(v_k(n)) w_{kj}(n) = -\sum_k \delta_k(n) w_{kj}(n) \tag{3.88}$$

Burada, eşitliğin ikinci kısmında, Denklem 3.79'da verilmiş olan, j yerine k indisinin bulunduğu, $\delta_k(n)$ yerel eğimi kullanılmaktadır.

Sonuç olarak, Denklem 3.88'in 3.80'de yerine yazılmasıyla, j gizli nöronu için $\delta_j(n)$ yerel eğimi,

$$\delta_j(n) = \varphi_j'(v_j(n))\sum_k \delta_k(n) w_{kj}(n),\ j \text{ nöronu gizli düğüm} \tag{3.89}$$

şeklinde elde edilir. Denklem 3.89'da, $\delta_j(n)$ yerel eğiminin hesaplanmasında kullanılan $\varphi_j'(v_j(n))$ faktörü, sadece j gizli nöronunun aktivasyon fonksiyonuna bağlıdır. Bu hesaplamadaki diğer faktör k'ya göre alınan toplam, iki terim kümesine bağlıdır. Birincisi, j gizli nöronunun bir sağındaki katmanda yer alan ve bu nörona doğrudan bağlı olan tüm nöronlar için $e_k(n)$ hata sinyalleri. İkincisi, bu bağlantılara ait $w_{kj}(n)$ sinaptik ağırlıkları.

Böylece, BP algoritması için geliştirilen ilişkiler özetlenebilir.

Öncelikle, i nöronunu j nöronuna bağlayan sinaptik ağırlığa uygulanan $\Delta w_{ji}(n)$ düzeltme değeri delta kuralıyla

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n) \tag{3.90}$$

şeklinde verilmektedir. Daha sonra, nöronun çıkış düğümü veya gizli düğüm olmasına bağlı olarak $\delta_j(n)$ yerel eğimi aşağıdaki şekilde hesaplanmaktadır:

- j nöronu çıkış düğümüyse, $\delta_j(n)$, j nöronuna ait $\varphi_j'(v_j(n))$ türevi ile $e_j(n)$ hata sinyalinin çarpımına eşittir.
- j nöronu gizli düğümse, $\delta_j(n)$, j nöronuna ait $\varphi_j'(v_j(n))$ türevi ile, j nöronuna doğrudan bağlı olan, sonraki gizli katman veya çıkış katmanındaki nöronlara ait δ değerlerinin ağırlıklı toplamının, çarpımına eşittir.

3.2 – Yöntem

3.2.1 – Genel Bilgiler

Burada, He ve arkadaşları (2002) tarafından önerilen otomatik konu değişikliği belirleme (automatic topic change identification) yaklaşımı Excite (http://www.excite.com) arama motoru veri kümesinden alınan bir örnek üzerinde uygulanmış ve bu amaçla kullanılabilecek yeni bir yaklaşım önerilerek, Excite ile birlikte Fast (http://www.fast.com) arama motoru veri kümelerinden alınan örnekler üzerinde uygulanmıştır.

Günümüzde İnternet, çok sayıda konu hakkında, farklı düzeylerde bilgiler içeren büyük bir bilgi kaynağı haline gelmiştir. Bu büyük bilgi kaynağı üzerinde, aradıkları bilgileri bulabilmek amacıyla, İnternet kullanıcıları, çoğunlukla arama motorlarını kullanmaktadırlar. Literatürde arama motorları kullanıcı davranışlarının incelenmesiyle ilgili çeşitli çalışmalar bulunmaktadır. Bu çalışma, arama motorları kullanıcı oturumlarındaki konu değişikliklerinin tespitiyle ilgili olup, benzer bir çalışma He ve arkadaşları (2002) tarafından yapılmıştır. Yazarlar, Reuters arama motoru veri kümesinden alınan bir örneği inceleyerek, arama motorları kullanıcı oturumlarındaki konu değişikliklerini belirlemek için bir yaklaşım önermişlerdir. Bu çalışmada, arama motorları kullanıcı oturumlarındaki konu değişikliklerinin belirlenmesi için (i) Dempster –Shafer teorisi ve genetik algoritmalar yaklaşımı ve (ii) yapay sinir ağları yaklaşımı olmak üzere iki farklı yaklaşım uygulanmıştır.

İlk yaklaşımda, konu değişikliklerini belirlemek için Excite arama motoru veri kümesinden alınan örnekten elde edilen istatistiksel veriler, Dempster-Shafer teorisi (Shafer, 1976) ve genetik algoritmalardan yararlanılmaktadır. Dempster-Shafer teorisi, çok sayıdaki bağımsız olayın olasılıklarını (delilleri) kullanarak, bu olasılıkları birleştirip, bu olaylara bağlı toplam olasılığın bulunmasına olanak sağlamaktadır. Bu teori kullanılarak bir "konu değişikliği var" olasılığı hesaplanmakta ve bu olasılık, bir eşik değeri ile karşılaştırılarak, "konu değişikliği yok" veya "konu değişikliği var" şeklinde bir ikili-değişkene dönüştürülmektedir. Dempster-Shafer teorisi kullanılarak birleştirilen olasılıklar, arama motoru veri kümesinden alınan örnekten hesaplanmıştır. Yaklaşımın parametrelerinin belirlenmesi için bir genetik algoritma (Goldberg, 1989) kullanılmıştır.

Yaklaşımın uygulanmasında kullanılan veri örnekleri, yaklaşık olarak iki eşit kısma ayrılmış; ilk yarısı eğitim veya yaklaşımda kullanılan bazı değerlerin belirlenmesi ve ikinci kısmı da yaklaşımın testi için kullanılmıştır.

Yaklaşımın başarısı, duyarlık (precision) ve anma (recall) performans ölçülerinin birleşmesinden oluşan bir performans ölçüsüyle değerlendirilmektedir. He ve arkadaşları (2002) çalışmalarında, bu performans ölçüsü açısından, önerdikleri yaklaşımın başarısını 0,8183 olarak belirtmişlerdir. Bu çalışmada, söz konusu yaklaşımın, farklı olasılıklara sahip olan farklı örnekler üzerinde uygulanması sonucunda elde edilen sonuçlar incelenmiş ve aşağıdaki soruların cevaplandırılmasına çalışılmıştır:

- Yaklaşımın performansı incelenen örneğe bağlı mıdır? Kullanıcı profilleri bir arama motorundan diğerine değişebilir. Yaklaşımın performansını, başka bir arama motorunun veri kümesinden alınan bir örnek üzerinde incelemek önemlidir. Bu amaçla, bu çalışmada Excite arama motorunun veri kümesinden alınan bir örnek kullanılmıştır.
- Yaklaşımın parametreleri, performans üzerinde ne kadar önemlidir? Yaklaşımın performansı, parametrelerdeki değişime ne kadar duyarlıdır? Bu sorular, aynı zamanda, He ve arkadaşlarının (2002) önerdikleri yaklaşım için evrensel bir parametre kümesi olup olmadığının belirlenmesi açısından da önem taşımaktadır. Böyle bir durumda, yeni veri kümeleri için parametrelerin belirlenmesi amacıyla genetik algoritma veya diğer yöntemlerin uygulanmasına gerek olmayacaktır.
- Örnekten elde edilen olasılıklar, performans üzerinde ne kadar önemlidir? Yaklaşımın performansı olasılıklardaki değişime ne kadar duyarlıdır? Bu sorular, aynı zamanda, He ve arkadaşlarının (2002) önerdikleri yaklaşım için evrensel bir olasılık kümesi olup olmadığının belirlenmesi açısından da önem taşımaktadır. Böyle bir durumda, yeni veri kümeleri için olasılıkların belirlenmesine gerek olmayacaktır.

İkinci yaklaşımda, konu değişikliklerini belirlemek için bir yapay sinir ağı kullanılmıştır. Yaklaşımın konu değişikliklerini belirlemek için kullanıdığı veriler, ilk yaklaşımda kullanılanlarla aynıdır. Bu veriler, yapay sinir ağına giriş olarak verilmiştir. Yapay sinir ağının çıkışı "konu değişikliği yok" ve "konu değişikliği var" durumlarını belirlemek için kullanılan bir sayıdır. Bu sayı, bir eşik değeri ile karşılaştırılarak, "konu değişikliği yok" veya "konu değişikliği var" şeklinde bir ikili-değişkene dönüştürülmektedir.

Yaklaşımın uygulanmasında kullanılan veri örnekleri, yaklaşık olarak iki eşit kısma ayrılmış; ilk yarısı yapay sinir ağının eğitimi ve ikinci yarısı da yapay sinir ağının testi için kullanılmıştır. Eğitim, istenen hata düzeyine veya belirli bir tekrar sayısına ulaşılıncaya kadar sürmektedir. Kullanılan yapay sinir ağı bir MLP olup, BP algoritmasına göre eğitilmektedir. Ağ yapısı (katman ve nöron sayıları, öğrenme oranı vs.) için çeşitli kombinasyonlar denenerek en uygun yapının bulunmasına çalışılmıştır. Eğitim tamamlandıktan sonra, yapay sinir ağı, örneğin ikinci yarısı üzerinde test edilmiştir. Bu yaklaşımda kullanılan performans ölçüleri de ilk yaklaşımda kullanılanlarla aynıdır. Yaklaşımın uygulanması sonucunda elde edilen sonuçlar incelenmiş ve aşağıdaki soruların cevaplandırılmasına çalışılmıştır:

- Yaklaşımın performansı incelenen verilere bağlı mıdır? Kullanıcı profilleri bir arama motorundan diğerine değişebilir. Yaklaşımın performansını, başka bir arama motorunun verileri üzerinde incelemek önemlidir. Bu amaçla, bu çalışmada Excite ile birlikte Fast arama motoru veri kümesinden alınan bir örnek de kullanılmıştır.
- Bir arama motoru veri örneğinin ilk yarısı kullanılarak eğitilen ve ikinci yarısı kullanılarak test edilen yapay sinir ağının göstermiş olduğu başarı sadece bu arama motorunun verileri için mi geçerlidir? Yapay sinir ağı benzer başarıyı başka bir arama motorundan alınan örnekler üzerinde de gösterebilir mi? Örneğin bir arama motoru veri örneğinin ilk yarısı (eğitim verisi) ile eğitilen bir yapay sinir ağı, başka bir arama motoru veri örneğinin ikinci yarısı (test verisi) üzerinde ne kadar başarılı olabilecektir? Bu sorular, yapay sinir ağı başarısının, veri kümesine bağlı olup olmadığının belirlenmesi açısından önemlidir.

- Yaklaşımın performansı, ilk yaklaşımınkiyle karşılaştırıldığına genel olarak nasıldır?

3.2.2 – Kullanılan Terminoloji

Bu çalışmada, Excite ve Fast arama motorları veri kümelerinden alınan örnekler kullanılmışıtır. Burada yeni oturumların (new sessions), bir kullanıcı kimliği (user ID) ile belirlenmesi mümkün olup, her sorgu üç alandan meydana gelmektedir: (i) Kimlik: arama motoru tarafından kullanıcıya atanan anonim bir kod, (ii) Günün zamanı (Time of day): saat, dakika ve saniye, (iii) Sorgu (Query): kullanıcı tarafından girilen terimler.

He ve arkadaşlarının (2002) çalışmasındakine yakın bir boyutta olması açısından, Excite ve Fast arama motorları veri kümelerinden yaklaşık 10.000 sorgudan oluşan örnekler seçilmiştir.

He ve arkadaşlarının (2002) çalışmalarında kullandıkları terminoloji ile Ozmutlu ve arkadaşlarının (2003a) ve Spink ve arkadaşlarının (2002a, 2002b) çalışmalarında, kullanılan terminoloji arasında bazı farklılıklar bulunduğundan, öncelikle bunlar açıklanacaktır. Çizelge 3.4'te kullanılan terminolojiler arasındaki farklılıklar görülmektedir.

IP değişikliği sorgu dizisinin sona erdiğini göstermeyebileceğinden, tek bir kullanıcı tarafından gönderilen sorgular bu çalışmada oturumu göstermek için kullanılmıştır. Bu durum, özellikle kullanıcının İnternet bağlantısı çevirmeli ağ bağlantısı ise gerçekleşebilir. Kullanıcı İnternet'e her bağlandığında arama motoru farklı bir IP adresi belirlemekle birlikte, bir çerez (cookie) yardımıyla kullanıcıyı tanıyabilir. Böyle bir durumda, arama motoru oturum devamını belirleyecek ve aynı anda farklı IP adresini kaydedecektir. Bu yüzden, oturum tek bir kullanıcı tarafından gönderilen sorgu grubunu göstermektedir.

He ve arkadaşlarının (2002) çalışmalarında kullandıkları terminoloji ile Spink ve arkadaşlarının (2000) çalışmalarında kullanılan terminoloji arasındaki diğer bir farklılık da arama yapısı sınıfları terimlerinde bulunmaktadır. Her iki çalışmada da arama yapılarının sınıflandırılması tek bir oturumdaki ardışık sorgulara dayanmakla birlikte, sınıf tiplerinin sayısı bazı farklılıklar göstermektedir. Spink ve arkadaşları (2000) dört

farklı sınıf (benzersiz (unique), değiştirilmiş (modified), sonraki sayfa (next page) ve ilgili geri-besleme (relevant feedback)) tanımlarken, He ve arkadaşları (2002) sekiz farklı sınıf (tarama (browsing), genelleştirme (generalization), özelleştirme (specialization), düzenleme (reformulation), tekrarlama (repetition), yeni (new), ilgili geri-besleme (relevance feedback) ve diğer (other)) tanımlamışlardır. Sorgu sınıflarının karşılaştırılması Çizelge 3.5'te görülmektedir.

Çizelge 3.4: Terminoloji Farkı

	Bu Çalışma	He ve arkadaşlarının (2002) Çalışması
A	Sorgu	Sorgu veya aktivite
B	Konu	Oturum
C	Oturum	IP değişikliği

A: Kayıt
B: Tek bir konuda tek bir kullanıcı tarafından gönderilen sorgu grubu
C: Tek bir kullanıcı tarafından gönderilen sorgu grubu

Spink ve arkadaşlarının (2000) kullandığı terminolojideki "değiştirilmiş" arama yapısının, He ve arkadaşlarının (2002) kullandığı terminolojideki "genelleştirme", "özelleştirme" ve "düzenleme" arama yapıları olarak ayrılmasının daha yararlı olabileceği düşünülmüştür. Bu yüzden "değiştirilmiş" arama yapısı sınıfının yerine "genelleştirme", "özelleştirme" ve "düzenleme" arama yapısı sınıfları kullanılmıştır.

Çizelge 3.5: Arama Yapısı Sınıflarının Karşılaştırılması

Spink ve arkadaşları	He ve arkadaşları
Benzersiz	Yeni
Değiştirilmiş	Genelleştirme
	Özelleştirme
	Düzenleme
Sonraki Sayfa	Tarama
İlgili Geri-Besleme	İlgili Geri-Besleme
Diğer	Diğer
	Tekrarlama

Geri kalan arama yapısı sınıfları için “tekrarlama” ve “diğer” arama yapısı sınıfları dışında, bire-bir eşleşme bulunmaktadır. He ve arkadaşlarının (2002) yaptıkları çalışmada, “tekrarlama” arama yapısı sınıfı için “ikinci sorgu birincisiyle aynı, ancak yapı tarama değil” şeklinde bir tanım yapmışlardır. İki aynı ardışık sorgunun “tekrarlama” (“sonraki sayfa” değil) olarak belirlenebilmesi için sorgu kaydında, Excite veri kümesinde olmayan özel bir göstergenin bulunması gerekmektedir. Bu yüzden, “tekrarlama” arama yapısı sınıfı bu çalışmada uygulanabilir değildir.

Ek olarak, He ve arkadaşları (2002) diğer bir sorgu tarafından izlenen bir “ilgili geri besleme” arama yapısı sınıfını olduğunda, “diğer” arama yapısı sınıfını kullanmışlardır. Öte yandan, “ilgili geri besleme” arama yapısı sınıfı boş bir sorgu kaydettiğinden, iki ardışık sorgu arasındaki ilişkinin tanımlanması mümkün değildir. Böyle bir durumda, “ilgili geri besleme” sorgusunu izleyen sorgu ile olan ilişkinin belirlenmesi için “ilgili geri besleme” sorgusundan önce gelen sorguyu kullanmak mantıklıdır. Bu çalışmada, He ve arkadaşlarının (2002) çalışmasına bu özellik eklenmiştir. Oturum ilk sorgusunun “ilgili geri besleme” olması durumunda, sonraki sorguyla ilişkisinin belirlenmesi mümkün olmadığından, sorgu “diğer” olarak işaretlenmiştir.

He ve arkadaşlarının (2002) çalışmasındakine göre yapılan diğer bir değişiklik de aynı zamanda Web’i tarama anlamına da gelmesinden dolayı, “tarama” teriminin “sonraki sayfa” ile değiştirilmesidir.

Arama yapısı sınıflarının detaylı tanımı aşağıdaki şekildedir:

- Yeni (New): İkinci sorgu birinci sorguyla ortak terim içermemektedir.
- Sonraki Sayfa (Next Page): İkinci sorgu birinci sorguyla aynıdır. Yani, ikinci sorgu, birinci sorguyla ilgili diğer bir sonuç kümesini istemektedir.
- Genelleştirme (Generalization): İkinci sorgu birinci sorgudan daha az terim içermektedir ve ikinci sorgunun bütün terimleri birinci sorguda yer almaktadır.
- Özelleştirme (Specialization): İkinci sorgu birinci sorgudan daha fazla terim içermektedir ve birinci sorgunun bütün terimleri ikinci sorguda yer almaktadır.

- Düzenleme (Reformulation): İkinci sorgu terimlerinin bazıları (tamamı değil) birinci sorguda yer almaktadır, ancak birinci sorgu ikinci sorguda yer almayan bazı terimler de içermektedir. Bu durum, kullanıcının birinci sorgudan bazı terimleri çıkardığını ve yine birinci sorguya yeni terimler eklediğini göstermektedir. Aynı zamanda, kullanıcı ilk sorgudaki terimleri, ikinci sorguda farklı bir sırada yazarsa, bu da "düzenleme" olarak düşünülmüştür. Bu durum sorguların aynı olmasını gerektiren "sonraki sayfa" olarak düşünülemez.
- İlgili Geri-Besleme (Relevance Feedback): İkinci sorgu hiç terim içermemektedir (boştur) ve kullanıcı ilgili sayfalar seçeneğini seçtiğinde sistem tarafından oluşturulmaktadır.
- Diğer (Other): İkinci sorgu yukarıdaki kategorilerin hiçbirine uymuyorsa, "diğer" olarak işaretlenir.

Çalışmada kullanılan diğer gösterimler aşağıdaki şekildedir:

- N_{shift} : algoritma tarafından "konu değişikliği var" olarak işaretlenen sorgu sayısı.
- N_{contin} : algoritma tarafından "konu değişikliği yok" olarak işaretlenen sorgu sayısı.
- $N_{true\ shift}$: insanlar tarafından "konu değişikliği var" olarak işaretlenen sorgu sayısı.
- $N_{true\ contin}$: insanlar tarafından "konu değişikliği yok" olarak işaretlenen sorgu sayısı.
- $N_{shift\ \&\ correct}$: algoritma ve insanlar tarafından "konu değişikliği var" olarak işaretlenen sorgu sayısı.
- $N_{contin\ \&\ correct}$: algoritma ve insanlar tarafından "konu değişikliği yok" olarak işaretlenen sorgu sayısı.

- w_{ti} : sorgunun zaman aralığına dayalı, "konu değişikliği var" ve "konu değişikliği yok" olasılıkları güven ağırlığı. Sadece birinci yaklaşımda kullanılmaktadır.
- w_{sp} : sorgunun arama yapısına dayalı, "konu değişikliği var" ve "konu değişikliği yok" olasılıkları güven ağırlığı. Sadece birinci yaklaşımda kullanılmaktadır.
- t_{shift} : sorguda konu değişikliği olup olmadığını belirlenmek için Dempster-Shafer teorisinden elde edilen olasılıkla karşılaştırılan eşik değeri. Sadece birinci yaklaşımda kullanılmaktadır.
- A Tipi Hata : aynı konuyla ilgili sorguların farklı konu olarak alınması durumunda oluşan hata.
- B Tipi Hata : farklı konuyla ilgili sorguların aynı konu olarak alınması durumunda oluşan hata.

Arama yapısı sınıfları Pascal'da yazılan bir bilgisayar programı ile belirlenmiştir. Üzerinde küçük değişiklikler yapılarak bu çalışmada kullanılan, He ve arkadaşları (2002) tarafından önerilmiş olan arama yapısı sınıfı belirleme algoritması Şekil 3.29'da görülmektedir.

Bu çalışmada uygulanan yaklaşımların performanslarını belirlemek amacıyla temel olarak iki performans ölçüsü kullanılmıştır. Bunlar, duyarlık (precision) P ve anma (recall) R performans ölçüleridir. Asıl performans ölçüsü olarak da bu iki performans ölçüsünü de dikkate alan (bunların belirli oranlarda birleşmesinden oluşan) F_β performans ölçüsü kullanılmıştır. İlk performans ölçüsü P aşağıdaki şekilde formüle edilmektedir:

$$P = \frac{N_{shift\&correct}}{N_{shift}} \tag{3.91}$$

İkinci performans ölçüsü R,

$$R = \frac{N_{shift\&correct}}{N_{true\ shift}} \tag{3.92}$$

şeklinde tanımlanmaktadır.

Input: Q_{i-1}, Q_i, Q_{i+1} sorguları
Local: Q_c : birinci sorgu
Q_n : ikinci sorgu
$B = \{t \mid t \in Q_c \wedge t \in Q_n\}$ // her iki sorguda da ortak olan terimler
$C = \{t \mid t \in Q_c \wedge t \notin Q_n\}$ // sadece birinci sorguda olan terimler
$D = \{t \mid t \notin Q_c \wedge t \in Q_n\}$ // sadece ikinci sorguda olan terimler
Output: sp // arama yapısı (search pattern)
Begin
If ($Q_i == \varnothing$) Then
If ($i == 1$) Then $sp =$ Diğer
Else $Q_c = Q_{i-1}$
End If
Else
$Q_c = Q_i$
$Q_n = Q_{i+1}$
End If
$sp =$ Diğer
If ($Q_n == \varnothing$) Then $sp =$ İlgili Geri Besleme End If
If ($B \neq \varnothing \wedge C \neq \varnothing \wedge D == \varnothing$) Then $sp =$ Genelleştirme End If
If ($B \neq \varnothing \wedge C == \varnothing \wedge D \neq \varnothing$) Then $sp =$ Özelleştirme End If
If ($B \neq \varnothing \wedge C \neq \varnothing \wedge D \neq \varnothing$) Then $sp =$ Düzenleme End If
If ($Q_n \neq Q_c \wedge B \neq \varnothing \wedge C == \varnothing \wedge D == \varnothing$) Then $sp =$ Düzenleme End If
If ($Q_c \neq \varnothing \wedge B == \varnothing$) Then $sp =$ Yeni End If
End

Şekil 3.29: Arama Yapısı Sınıfı Belirleme Algoritması

Bu çalışmada, bu iki performans ölçüsünün birleşmesinden meydana gelen F_β performans ölçüsü kullanılmıştır:

$$F_\beta = \frac{(1+\beta^2)PR}{\beta^2 P + R} \tag{3.93}$$

Görüldüğü gibi bu performans ölçüsü diğer iki performans ölçüsünü de içermektedir. Bu yüzden, performans ölçüsü olarak F_β kullanılmış ve R'ye daha fazla ağırlık verilmesi amacıyla, β parametresi 1,5 olarak alınmıştır.

3.2.3 – Dempster-Shafer Teorisi ve Genetik Algoritmalar Yaklaşımı

Zaman aralıkları ve arama yapıları belirlendikten sonra bunlara dayalı olarak konu değişikliği olma olasılıkları hesaplanmış ve daha sonra da hesaplanan bu olasılıklar (deliller), Dempster-Shafer teorisiyle birleştirilerek, bu iki olasılık tek bir olasılık haline getirilmiştir.

Dempster-Shafer teorisi farklı kaynaklardan elde edilen olasılıkları (delilleri) birleştirmek amacıyla kulanılabilmektedir.

Eğer y hipotezi için olasılıklar birleştirilecekse ve olasılıkların elde edildiği m_1 ve m_2 gibi iki kaynak varsa, teoriye göre bunlar aşağıdaki şekilde birleştirilebilirler:

$$[m_1 \oplus m_2](y) = \begin{cases} 0 & y = \varnothing \\ \dfrac{\sum_{A \cap B = y} m_1(A) m_2(B)}{1 - \sum_{A \cap B = \varnothing} m_1(A) m_2(B)} & y \neq \varnothing \end{cases} \tag{3.10}$$

Buradaki durumda, ayırt etme yapısı aşağıdaki şekildedir:

$$\Theta = \{P_s, P_c\} \tag{3.95}$$

Burada, P_c kaydın konu değişikliği içermemesi (konu değişikliği yok) ve P_s de kaydın konu değişikliği içermesi (konu değişikliği var) olarak belirlenmiştir. Bu olasılıkları, temel olasılık atamalarına dönüştürmek için bir w_i ağırlığı kullanılmıştır. Bu ağırlık, konu değişikliği belirlemek için bu olasılığın kullanılmasına olan güveni göstermektedir. Böylece, temel olasılık ataması, şartlı olasılık ve ilgili ağırlık w_i'nin çarpılmasıyla bulunmaktadır:

$$m_i(P_s) = P(shift \mid i) w_i \tag{3.96}$$

ve

$$m_i(P_c) = P(contin \mid i) w_i \tag{3.97}$$

burada, i, ti (zaman aralığı) veya sp (arama yapısı) olmaktadır.

Bu ağırlıklar, temel olasılık atamasındaki olasılıkların bir kısmının Θ ile ilişkilendirilmesine ve diğer olasılıkla ilişkilendirilmesine olanak sağlamaktadır. $m_i(\Theta)$ 'ların hesaplanması aşağıdaki şekilde yapılır:

$$m_{ti}(\Theta) = 1 - m_{ti}(P_s) - m_{ti}(P_c) \tag{3.98}$$

ve

$$m_{sp}(\Theta) = 1 - m_{sp}(P_s) - m_{sp}(P_c) \tag{3.99}$$

Θ içinde sadece iki hipotez olmasından dolayı, buradaki durum için Dempster kuralı aşağıdaki şekilde yazılabilir:

$$\left[m_{ti} \oplus m_{sp}\right](P_s) = \frac{m_{ti}(P_s)m_{sp}(P_s) + m_{ti}(P_s)m_{sp}(\Theta) + m_{ti}(\Theta)m_{sp}(P_s)}{1 - (m_{ti}(P_s)m_{sp}(P_c) + m_{ti}(P_c)m_{sp}(P_s))} \tag{3.100}$$

Birleştirilmiş değerleri ikili kararlara dönüştürmek için bir t_{shift} eşiği kullanılmıştır. Buna göre, bir kayıttan hesaplanmış olan birleştirilmiş değer bu eşiği aşarsa, bu kaydın konu değişikliği içerdiği düşünülmektedir. Bu işlemlerin gerçekleştirilmesinden sonra, konu değişikliklerini belirlemek amacıyla kullanılan algoritma Şekil 3.30'da görülmektedir.

Araştırma sorularının cevaplandırılabilmesi için Excite arama motoruna ait yaklaşık 10.000 sorgudan oluşan bir veri örneği incelenmiştir. He ve arkadaşları (2002) çalışmalarında iki tip bilgi kullanmışlardır. Birincisi, veri örneğinden elde edilen olasılıklar (belirli bir zaman aralığı veya arama yapısı için "konu değişikliği yok" ve "konu değişikliği var" olasılıkları). İkincisi de uygunluk fonksiyonu F_β'yı enbüyüklemek için genetik algoritma ile belirlenmiş olan w_{ti}, w_{sp} ve t_{shift} parametreleri. Bu çalışmada, He ve arkadaşlarının (2002) çalışmasındaki algoritmanın

performansını doğrulamak amacıyla söz konusu iki tip bilgi için farklı değerler kullanılmıştır. Çizelge 3.6, kullanılan bu farklı değerleri göstermektedir.

```
Begin
    For (her kayıt) Do
        Begin
            If (IP adresi değişir) Then IP değişikliği olarak işaretle;
            ElseIf ( m_ti&sp(P_s) > t_shift ) Then "konu değişikliği var" olarak işaretle;
            Else "konu değişikliği yok" olarak işaretle;
        End
End
```

Şekil 3.30: Konu Değişikliği Belirleme Algoritması

Çizelgeden de görüldüğü gibi Excite ve Reuters arama motorlarının veri örnekleri için hesaplanmış olan "konu değişikliği yok" ve "konu değişikliği var" olasılıkları kullanılmıştır. Aynı zamanda, genetik algoritma ile belirlenmiş olan w_{ti}, w_{sp} ve t_{shift} parametre değerleri ve He ve arkadaşlarının (2002) çalışmalarında verilmiş olan w_{ti}, w_{sp} ve t_{shift} parametre değerleri ve ayrıca, rasgele üretilmiş w_{ti}, w_{sp} ve t_{shift} parametre değerleri kullanılarak, algoritmanın performansı test edilmiştir. Aşağıda bu altı test durumunun nasıl kullanıldığı açıklanmaktadır.

Çizelge 3.6: Farklı Olasılık ve Parametre Kümeleri için Durumlar

	Excite Olasılıkları	Reuters Olasılıkları
PK I	Durum 1	Durum 2
PK II	Durum 3	Durum 4
PK III	Durum 5	Durum 6

PK I: Genetik algoritma ile belirlenmiş olan parametre kümesi.
PK II: He ve arkadaşlarının (2002) çalışmasındaki parametre kümesi.
PK III: Rasgele üretilmiş parametre kümesi.

Durum 1: Excite veri örneğinden hesaplanan olasılıklar ve genetik algoritma ile belirlenmiş olan w_{ti}, w_{sp} ve t_{shift} parametreleri kullanılarak He ve arkadaşlarının (2002) çalışmasındaki yaklaşımın tekrarlanması.

Excite veri örneğinden hesaplanan olasılık değerleri genetik algoritmada ağırlık değerlerini ve eşik değerini belirlemek için kullanılmıştır. Genetik algoritmanın

girişleri, “konu değişikliği yok” ve “konu değişikliği var” olasılıklarıdır. w_{ti}, w_{sp} ağırlıkları ve t_{shift} eşik değeri, F_β uygunluk fonksiyonunu enbüyükleyecek şekilde genetik algoritma tarafından belirlenmiştir. He ve arkadaşlarının (2002) uyguladıkları genetik algoritma aynı şekilde bu çalışmada da uygulanmıştır. w_{ti}, w_{sp} ağırlıkları ve t_{shift} eşik değeri bir kromozomun üç geni olarak alınmıştır. Popülasyon büyüklüğü 50 ve nesil sayısı 500 olarak belirlenmiştir. Uygunluk fonksiyonu F_β performans ölçüsünün değeridir. Excite veri örneğindeki verilerin yaklaşık olarak yarısı (5014 sorgu) üzerinde genetik algoritma çalıştırılarak parametre değerleri belirlenmiştir. Genetik algoritmanın çalışması sonucunda w_{ti}, w_{sp} ve t_{shift} için sırasıyla 0,5684, 0,7236 ve 0,1162 değerleri bulunmuştur. Burada elde edilen P, R ve F_β değerleri de sırasıyla 0,4672, 0,7138 ve 0,6140 şeklindedir.

He ve arkadaşlarının (2002) çalışmasındakine benzer şekilde, algoritma, belirlenmiş olan w_{ti}, w_{sp} ve t_{shift} değerleri ve Excite veri örneğinin “konu değişikliği yok” ve “konu değişikliği var” olasılıklarıyla, Excite veri örneğinin ikinci yarısı üzerinde uygulanmıştır. Performans ölçüleri P, R ve F_β değerleri, He ve arkadaşlarının (2002) çalışmasındaki değerlerle karşılaştırılmıştır.

Durum 2: Reuters veri örneğinden hesaplanan olasılıklar ve genetik algoritma ile belirlenmiş olan w_{ti}, w_{sp} ve t_{shift} parametreleri kullanılarak He ve arkadaşlarının (2002) çalışmasındaki yaklaşımın tekrarlanması.

F_β performans ölçüsünü enbüyüklemek amacıyla w_{ti}, w_{sp} ve t_{shift} parametreleri genetik algoritma ile belirlenmiştir. Bununla birlikte, Durum 1’deki Excite veri örneğinden elde edilen olasılıkların kullanılması yerine, Reuters veri örneğinden (He ve arkadaşları, 2002) elde edilen olasılıklar kullanılmıştır. Genetik algoritmanın çalışması sonucunda w_{ti}, w_{sp} ve t_{shift} için sırasıyla 0,1104, 0,9424 ve 0,6016 değerleri bulunmuştur. Burada elde edilen P, R ve F_β değerleri de sırasıyla 0,4672, 0,7138 ve 0,6140 şeklindedir.

Konu değişikliği belirleme algoritması belirlenmiş olan w_{ti}, w_{sp} ve t_{shift} değerleri ve Reuters veri örneğinden (He ve arkadaşları, 2002) elde edilen olasılıklarla Excite veri örneğinin ikinci yarısı üzerinde uygulanmıştır. Durum 1 ve Durum 2, sonuçların genetik algoritmada kullanılan olasılıklara duyarlılığını test etmek için kullanılabilir.

Durum 3: Excite veri örneğinden hesaplanan olasılıklar ve He ve arkadaşlarının (2002) çalışmasında verilmiş olan w_{ti}, w_{sp} ve t_{shift} parametreleri kullanılarak He ve arkadaşlarının (2002) çalışmasındaki yaklaşımın tekrarlanması.

Genetik algoritma kullanmak yerine, parametre değerleri olarak He ve arkadaşlarının (2002) çalışmasında verilmiş olan değerler alınmıştır. "konu değişikliği yok" ve "konu değişikliği var" olasılıkları Excite veri örneğinden hesaplanmıştır. Konu değişikliği belirleme algoritması belirlenmiş olan w_{ti}, w_{sp} ve t_{shift} parametreleri ve hesaplanan olasılıklarla Excite veri örneğinin ikinci yarısı üzerinde uygulanmış, P, R ve F_{β} değerleri hesaplanmıştır. Bu durumda, konu değişikliği belirleme algoritmasının He ve arkadaşlarının (2002) çalışmasında verilen parametreler ile Excite veri örneği üzerinde başarılı olup olmadığı belirlenebilir. Diğer bir deyişle, He ve arkadaşlarının çalışmasında (2002) verilen parametreler Excite veri örneği üzerinde çalışıyorsa, bu değerlerin, Web kullanıcı oturumları için evrensel olarak kabul edilebilir değerler olduğu şeklinde yorum yapılabilir.

Durum 4: Reuters veri örneğinden hesaplanan olasılıklar ve He ve arkadaşlarının (2002) çalışmasında verilmiş olan w_{ti}, w_{sp} ve t_{shift} parametreleri kullanılarak He ve arkadaşlarının (2002) çalışmasındaki yaklaşımın tekrarlanması.

Genetik algoritma kullanmak yerine, Durum 3'e benzer şekilde, parametre değerleri olarak He ve arkadaşlarının (2002) çalışmasında verilmiş olan değerler alınmıştır. Durum 3'ten farklı olarak "konu değişikliği yok" ve "konu değişikliği var" olasılıkları olarak, Reuters veri örneğinden hesaplanmış olan değerler kullanılmıştır. Konu değişikliği belirleme algoritması, belirlenmiş olan w_{ti}, w_{sp} ve t_{shift} parametreleri

ve hesaplanan olasılıklarla Excite veri örneğinin ikinci yarısı üzerinde uygulanmış, P, R ve F_β değerleri hesaplanmıştır. Bu duruma kadar olan sonuçlar (Durum 1'den Durum 4'e), olasılık değerleri ve w_{ti}, w_{sp} ve t_{shift} parametrelerinin kaynağının önemini belirlemede yardımcı olabilir.

Durum 5: Excite veri örneğinden hesaplanan olasılıklar ve rasgele üretilmiş olan w_{ti}, w_{sp} ve t_{shift} parametreleri kullanılarak He ve arkadaşlarının (2002) çalışmasındaki yaklaşımın tekrarlanması.

Genetik algoritma kullanmak yerine, rasgele üretilmiş olan w_{ti}, w_{sp} ve t_{shift} parametreleri kullanılmıştır. (0, 1) aralığında düzgün dağılama uyacak şekilde 10 rasgele parametre değeri üretilmiştir. "konu değişikliği yok" ve "konu değişikliği var" olasılıkları Excite veri örneğinden hesaplanmıştır. Konu değişikliği belirleme algoritması, belirlenmiş olan w_{ti}, w_{sp} ve t_{shift} parametreleri ve hesaplanan olasılıklarla Excite veri örneğinin ikinci yarısı üzerinde uygulanmış, P, R ve F_β değerleri hesaplanmıştır.

Durum 6: Reuters veri örneğinden hesaplanan olasılıklar ve rasgele üretilmiş olan w_{ti}, w_{sp} ve t_{shift} parametreleri kullanılarak He ve arkadaşlarının (2002) çalışmasındaki yaklaşımın tekrarlanması.

Genetik algoritma kullanmak yerine, rasgele üretilmiş w_{ti}, w_{sp} ve t_{shift} parametreleri kullanılmıştır. (0, 1) aralığında düzgün dağılama uyacak şekilde 10 rasgele parametre değeri üretilmiştir. "konu değişikliği yok" ve "konu değişikliği var" olasılıkları Reuters veri örneğinden hesaplanmıştır. Konu değişikliği belirleme algoritması, belirlenmiş olan w_{ti}, w_{sp} ve t_{shift} parametreleri ve hesaplanan olasılıklarla Excite veri örneğinin ikinci yarısı üzerinde uygulanmış, P, R ve F_β değerleri hesaplanmıştır.

3.2.3.1 – Örnek Olasılıkları ile F_β Arasındaki İlişkinin İstatistiksel Testi

"konu değişikliği yok" ve "konu değişikliği var" olasılıklarının F_β üzerindeki etkilerini incelemek için (Durum 1-6 sonuçlarına dayanarak) çift yönlü t-testi uygulanabilir. Çift yönlü t-testi ile Excite ve Reuters veri örneklerinden hesaplanan olasılıkların kullanıldığı durumlar kıyaslanabilir (Durum 1, 3, ve 5 ile Durum 2, 4 ve 6). Çift yönlü t-testi uygulandığında, karşılaştırılan örneklerin F_β dışındaki tüm parametreleri aynı olmalıdır. Bu durumda Durum 1 ve Durum 2 çift yönlü t-testinden çıkarılmalıdır. Genetik algoritma ile belirlendiğinden dolayı, bu durumlarda w_{ti}, w_{sp} ve t_{shift} parametrelerinin değerleri aynı değildir. Bu yüzden sırasıyla Durum 3 ve 5 ile Durum 4 ve 6 kıyaslanmıştır. Çift yönlü t-testi olasılık kaynaklarının F_β üzerinde herhangi bir etkisi olup olmadığını gösterebilir. Eğer olasılıklar konu değişikliği belirleme algoritmasında kritik rol oynuyorlarsa, sırasıyla Durum 3 ve 5'teki F_β değerinin, Durum 4 ve 6'dakine göre daha yüksek olması beklenmektedir.

3.2.3.2 – Parametreler ile Performans Ölçüleri Arasındaki İlişkinin Belirlenmesi

Önceki adımlardaki sonuçlara dayanarak, daha detaylı bir istatistiksel analiz gerekebilir. İstatistiksel analiz Ozmutlu ve Harmonosky'den (2003, baskıda-a, baskıda-b) uyarlanmıştır. Çoklu doğrusal regresyon denklemleri konu değişikliği belirleme algoritmasının parametreleri (w_{ti}, w_{sp} ve t_{shift}) ile performans ölçüleri (P, R ve F_β) arasındaki ilişkinin anlaşılması için kullanılabilir. Aynı zamanda regresyon ile eşzamanlı olarak varyans analizi da uygulanabilir. Bu analizleri yapmak için aşağıdaki adımlar uygulanmıştır:

- Burada, üç çoklu regresyon denklemi ve varyans analizi gereklidir. F_β için üçüncü çoklu doğrusal regresyon denklemi, F_β doğrudan w_{ti}, w_{sp} ve t_{shift} parametrelerine bağlı olmayıp, P ve R değerlerinden hesaplandığı için oluşturulmamıştır. P ve R'yi belirlemek için bağımsız ve bağımlı değişkenler arasındaki ilişkiyi ortaya çıkaran, ayrı bir çoklu doğrusal regresyon modeli düşünülmüştür. Düşünülen modelin yapısı şu şekildedir:

$Y = X\beta + \varepsilon$, $\varepsilon \sim$ iid $N(0, \sigma^2)$. Burada, Y, bağımlı değişkenin n sayıdaki bir kümesi, X bağımsız değişkenlerin değerlerinin matris gösterimi, β regresyon denklemi için katsayılar vektörü ve ε da n bağımsız aynı dağılımlı bileşen için normal hata vektörüdür. Varyans analizi regresyon denkleminin toplam etkisini ve her bir değişkenin bağımlı değişkene etkisini gösterecektir.

- Doğrusal regresyon denkleminin bağımsız değişkenleri belirlenmiştir. Performans ölçüleri P ve R 'nin konu değişikliği belirleme algoritmasının parametreleri ile açıklanması istendiğinden, çoklu doğrusal regresyon denklemleri ve varyans analizinin bağımsız değişkenleri w_{ti}, w_{sp} ve t_{shift} parametreleridir.

- Çoklu doğrusal regresyon denkleminin ve varyans analizi için kareler toplamının katsayılarının belirlenmesi için bir deneysel tasarım hazırlanmıştır. Deneysel tasarımın faktörleri çoklu doğrusal regresyon denklemi ve varyans analizinin bağımsız değişkenleridir. Deneysel tasarım aşağıdaki şekilde hazırlanmıştır:

- Bağımsız değişkenlerin düzeyleri belirlenmiştir. Deneysel tasarımın tüm faktörleri (w_{ti}, w_{sp} ve t_{shift}) 0 ile 1 arasında değişmektedir. Deneysel tasarımın faktörleri mümkün olduğunca eşit-aralıklı olmalıdır. Buradaki durum için bağımsız faktörlere, 0, 0,2, 0,4, 0,6, 0,8 ve 1,0 şeklinde altı eşit-aralıklı düzey vermek uygundur. Bunların tümü anlamlı olmamakla birlikte bütünlüğün sağlanması bakımından dikkate alınmıştır.

- Modele eklenen çoklu doğrusal regresyon denkleminin terimleri her bir faktörün düzey sayısına göre belirlenmiştir. Regresyon denkleminin her bir etkisi, aynı zamanda varyans analizinde de bir etki olarak gösterilmiştir. Bu çalışmadaki faktörlerin her biri altı düzey ile gösterildiğinden, ilgili çoklu doğrusal regresyon denklemi ve varyans analizi beşinci dereceye kadar terimleri içerebilir. Bununla birlikte hemen her zaman, ikinci dereceye kadar olan terimleri ve iki düzeyli faktör etkileşimlerini incelemek yeterli olmaktadır. Bu yüzden, her bir bağımlı faktör için

regresyon denklemi ve varyans analizi doğrusal ana etkileri, kuadratik ana etkileri ve iki faktör etkileşimlerini içerebilir (doğrusal-doğrusal, doğrusal-kuadratik, kuadratik-doğrusal ve kuadratik-kuadratik). Böylece, düşünülen doğrusal regresyon denklemi ve varyans analizine eklenen etkiler aşağıdaki şekildedir:

Bağımlı değişken (P veya R),

$$\begin{aligned} P \vee R = {} & \beta_0 + \beta_1 w_{ti} + \beta_2 w_{sp} + \beta_3 t_{shift} + \\ & \beta_4 w_{ti} w_{sp} + \beta_5 w_{ti} t_{shift} + \beta_6 w_{sp} t_{shift} + \\ & \beta_7 w_{ti}^2 + \beta_8 w_{sp}^2 + \beta_9 t_{shift}^2 + \\ & \beta_{10} w_{ti} w_{sp}^2 + \beta_{11} w_{ti} t_{shift}^2 + \beta_{12} w_{sp} t_{shift}^2 + \\ & \beta_{13} w_{ti}^2 w_{sp} + \beta_{14} w_{ti}^2 t_{shift} + \beta_{15} w_{sp}^2 t_{shift} + \\ & \beta_{16} w_{ti}^2 w_{sp}^2 + \beta_{17} w_{ti}^2 t_{shift}^2 + \beta_{18} w_{sp}^2 t_{shift}^2 + \varepsilon_i \end{aligned} \tag{3.101}$$

Burada, $\varepsilon_i \sim$ iid $N\left(0, \sigma^2\right)$ şeklindedir.

- Deneysel tasarımın çalışma noktaları belirlenmiştir. Deneyin çalışma süresi açısından uygun olmasından dolayı, tam deneysel tasarım uygulanabilir. Faktörlerin altı seviyeli tüm farklı kombinasyonlarıyla, tam deneysel tasarım $6^3 = 216$ deney olarak uygulanır. Bu deneyler yoluyla, önerilen doğrusal regresyon denklemindeki tüm katsayıların ve varyans analizindeki kareler toplamı değerlerinin belirlenmesi mümkündür.
- Regresyon denklemi veya varyans analizi için istatistiksel analiz gerçekleştirilmiştir. Verilen doğrusal regresyon göz önüne alındığında hipotezler aşağıdaki gibidir:

$$H_0 = \beta_i = 0 \tag{3.102}$$

ve

$$H_1 = \beta_i \neq 0,\ i = 1,\ldots,18 \tag{3.103}$$

Etkiler %95 anlamlılık düzeyinde test edilmiştir. %95 anlamlılık düzeyi için kritik F -değeri $F_{0,05,18,197} \sim 3,84$'tür. Sonuçlar üzerindeki diğer istatistiksel analizler, regresyon modelinin genel anlamlılığını test etmektedir. Regresyon modelinin anlamlılığını test eden ilgili hipotez aşağıdaki şekildedir:

$$H_0 = \beta_0 = \beta_1 = \ldots = \beta_{18} = 0 \tag{3.104}$$

ve

$$H_1 = \beta_i \neq 0 \text{, herhangi } i = 1,\ldots,18 \tag{3.105}$$

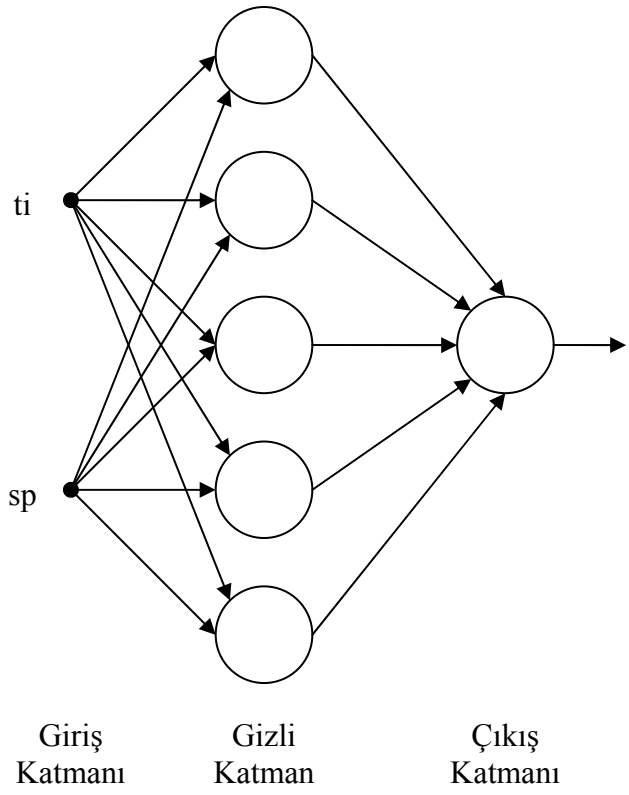
Etkiler %95 anlamlılık düzeyinde test edilmiştir. %95 anlamlılık düzeyi için kritik F -değeri $F_{0,05,18,197} \sim 1,61$'dir.

3.2.4 – Yapay Sinir Ağları Yaklaşımı

Bu çalışmada kullanılan ikinci yaklaşım, konu değişikliklerini belirlemek için bir yapay sinir ağının kullanılmasıdır. Kullanılan yapay sinir ağının iki girişi olup, bunlar zaman aralığı ve arama yapısı şeklindedir. Yapay sinir ağının tek çıkışı ise "konu değişikliği yok" veya "konu değişikliği var" durumlarını belirlemek amacıyla kullanılan bir sayıdır. Kullanılan ağın bir gizli katmanı olup, bu katmanda beş nöron bulunmaktadır. Sonuç olarak, yapılan deneyler sonucunda en uygun yapay sinir ağı olarak bulunmuş olan bu ağın yapısı 2-5-1 şeklindedir. Şekil 3.31'de bu yapay sinir ağı görülmektedir.

Yapay sinir ağı Excite ve Fast arama motorları veri kümelerinden alınan örnekler üzerinde denenmiştir. Örnekler yaklaşık olarak iki kısma ayrılmış olup, ilk kısım yapay sinir ağının eğitilmesi, ikinci kısım da yapay sinir ağının test edilmesi için kullanılmıştır. Kullanılan örneklerin büyüklüğü önceki yaklaşımda kullanılanlarla aynı olup, yaklaşık 10.000 veriden oluşmaktadır. Yapay sinir ağı MATLAB programı kullanılarak tasarlanmıştır.

Kullanılan ağ bir MLP olup, ağın eğitiminde BP algoritması kullanılmıştır. Yapay sinir ağında, momentumlu toplu eğim düşümü (batch gradient descent with momentum) eğitim yöntemi kullanılmıştır. Gizli katmada tanjant tipi sigmoid ve çıkış katmanında da doğrusal aktivasyon fonksiyonları bulunmaktadır.



Şekil 3.31: Kullanılan Yapay Sinir Ağı

Yapay sinir ağının performansı üzerinde önemli bir etkisi olan öğrenme oranının değeri de 0,1 olarak alınmıştır. Ayrıca yapay sinir ağının daha iyi bir performans gösterebilmesi için momentum da kullanılmış ve değeri 0,8 olarak alınmıştır. Yapay sinir ağı eğitimi, iterasyon sayısı 1.000 değerine ulaşıncaya veya hata düzeyi %4,5'in altına düşünceye kadar sürdürülmüştür. Buradaki, hata düzeyi, ortalama karesel hata (mean squared-error – mse) performans ölçüsüdür.

Yapay sinir ağına arama motoru kayıtlarından elde edilen zaman aralığı ve arama yapısı değerleri verilmiştir. İstenen çıkışlar ise yapay sinir ağına giriş olarak

verilmiş olan zaman aralığı ve arama yapısı değerlerine karşılık, insanlar tarafından belirlenmiş olan "konu değişikliği yok" ve "konu değişikliği var" (1 ve 2) durumlarıdır.

Doğal olarak, veri örneğindeki kayıtlar insanlar tarafından konu değişikliği olup olmaması açısından incelenirken, "konu değişikliği yok" (1) ve "konu değişikliği var" (2) durumlarına ek olarak, "IP değişikliği" (3) şeklinde üçüncü bir durum daha söz konusudur. Ancak, burada amaç yapay sinir ağının konu değişikliklerini tespit etmesi olduğundan, bu üçüncü durum yapay sinir ağının eğitiminde kullanılacak olan veriler belirlenirken dikkate alınmamış olup, sadece "konu değişikliği yok" (1) ve "konu değişikliği var" (2) durumları istenen çıkışlar olarak alınmıştır.

Yapay sinir ağının eğitimi sona erdiğinde (istenen hata düzeyine veya iterasyon sayısı olarak 10.000 değerine ulaşıldığında), ağ örneğin geriye kalan ikinci kısmı kullanılarak test edilmiştir. Test aşamasında, yapay sinir ağına ilgili girişler (zaman aralığı ve arama yapısı) verilerek, ağın bu girişlere cevap vermesi beklenmiştir. Burada, önerilen yaklaşımın performansının incelenebilmesi için yapay sinir ağının vermiş olduğu bu çıkışlar, yine insanlar tarafından belirlenmiş olan çıkışlarla karşılaştırılarak, duyarlık (precision), anma (recall) ve bunların ikisinin birleşiminden oluşan bir performans ölçüsü hesaplanmaktadır (P, R ve F_β). Test aşamasında, yapay sinir ağı kendisine verilen girişlere bağlı olarak bir çıkış vermektedir. Ancak yapay sinir ağının verdiği çıkış doğrudan 1 veya 2 şeklinde olmadığından, çıkışın bu değerlerden birisine dönüştürülmesi gerekmektedir. Bunun için bir eşik değeri kullanılmıştır. Yapay sinir ağının verdiği çıkış, eşik değerinden küçük olduğunda, çıkış değeri 1 olarak alınmakta (konu değişikliği yok), diğer durumlarda ise 2 olarak alınmaktadır (konu değişikliği var). f eşik fonksiyonu, y yapay sinir ağının çıkışı ve θ da eşik değeri olmak üzere

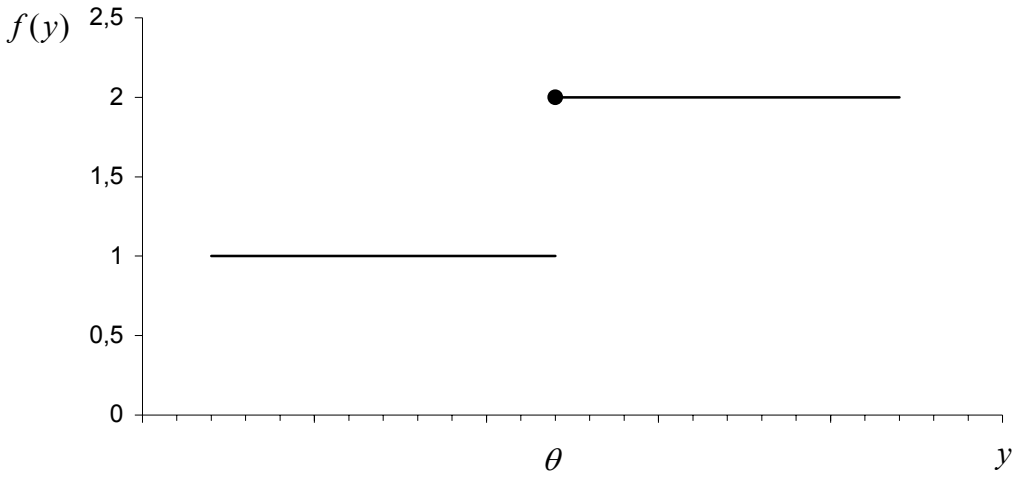
$$f(y) = \begin{cases} 1 & y < \theta \\ 2 & y \geq \theta \end{cases} \tag{3.106}$$

olmaktadır. Bu durum, Şekil 3.32'de görülmektedir.

Eşik değeri olarak 1,1'den 1,5'e kadar olan değerler alınmış ve her eşik değeri için ağ, 300 kez eğitilip test edilmiştir. Eşik değeri olarak, 1,5'ten daha küçük değerlerin alınma nedeni ise kullanılan performans ölçüsü F_β'nın hesaplanmasında R'nin daha

çok önem taşıması ve bu yüzden, yapay sinir ağı çıkışının 2 (konu değişikliği var) olması yönünde bir eğilimin, daha iyi sonuç verecek olmasıdır.

Yapay sinir ağının test aşamasında, ağın vermiş olduğu değerler 1 veya 2'ye yuvarlandıktan sonra, gerçek çıkışlarla karşılaştırılmaktadır. Yapay sinir ağının verdiği doğru ve yanlış cevapların sayısı belirlenerek, P, R ve F_β performans ölçüsü değerleri hesaplanmaktadır.



Şekil 3.32: Yapay Sinir Ağı Çıkışının Yuvarlanması

3.2.4.1 – Eğitim ve Test Verilerinin Farklı Veri Kümelerinden Seçilmesi

Arama motorları kullanıcı oturumlarındaki konu değişikliklerinin belirlenmesi için kullanılan yapay sinir ağının eğitim ve testi için kullanılan örnekler, farklı veri kümelerinden seçilerek, ağın performansı iki farklı açıdan incelenebilir: (i) farklı eğitim örnekleri kullanılarak eğitilen yapay sinir ağının, belirli bir test örneği üzerindeki performansı ve (ii) belirli bir eğitim örneği kullanılarak eğitilen yapay sinir ağının, farklı test örnekleri üzerindeki performansı. Bu amaçla, Excite ve Fast arama motorları veri kümelerinden alınan örneklerden yararlanılmıştır. Yapılan deneyler sırasında, yapay sinir ağının eğitilmesi aşamasında herhangi bir değişiklik bulunmamaktadır. Bununla birlikte, herhangi bir veri örneğinin ilk yarısı kullanılarak eğitilen yapay sinir ağı, sadece o veri örneğinin değil, diğer veri örneğinin de ikinci yarısı üzerinde test edilmiştir. Örneğin, Excite veri örneğinin ilk yarısı kullanılarak eğitilen yapay sinir ağı,

sadece Excite veri örneğinin ikinci yarısı üzerinde test edilmeyip, aynı zamanda, Fast veri örneğinin ikinci yarısı üzerinde de test edilmiştir. Benzer şekilde, Fast veri örneğinin ilk yarısı üzerinde eğitilen yapay sinir ağı da sadece Fast veri örneğinin ikinci yarısı üzerinde test edilmeyip, aynı zamanda, Excite veri örneğinin ikinci yarısı üzerinde de test edilmiştir. Çizelge 3.7, bu durumları özetlemektedir.

İlk olarak, yapay sinir ağı, 50 kez Excite eğitim örneğiyle eğitilmiş ve her eğitim sonunda, Excite ve Fast test örnekleriyle test edilmiştir. Daha sonra, yapay sinir ağı, 50 kez Fast eğitim örneğiyle eğitilmiş ve her eğitim sonunda, Excite ve Fast test örnekleriyle test edilmiştir. Testler sonucunda, P, R ve F_β performans ölçülerinin değerleri hesaplanmıştır.

Burada yapılan işlemler; farklı eğitim örnekleri kullanılarak eğitilen yapay sinir ağının, belirli bir test örneği üzerindeki performansının ve belirli bir eğitim örneği kullanılarak eğitilen yapay sinir ağının, farklı test örnekleri üzerindeki performansının incelenmesi açısından önem taşımaktadır.

Çizelge 3.7'de özetlenen durumlar aşağıda açıklanmıştır. Durum 1 ve 2 ile Durum 3 ve 4 için (eğitim örneği aynı, test örnekleri farklı) yapay sinir ağının bu durumlardaki davranışları karşılaştırılmıştır. Durum 1 ve 3 ile Durum 2 ve 4 için (eğitim örnekleri farklı , test örneği aynı) yapay sinir ağının bu durumlardaki davranışları ve elde edilen performans ölçülerinin değerleri karşılaştırılmıştır.

Çizelge 3.7: Farklı Eğitim ve Test Örnekleri için Durumlar

		Test	
		Excite	Fast
Eğitim	Excite	Durum 1	Durum 2
	Fast	Durum 3	Durum 4

Durum 1: Eğitim: Excite – Test: Excite

Yapay sinir ağı, Excite veri kümesinden alınan örneğin ilk yarısı kullanılarak eğitilmiş, Excite veri kümesinden alınan örneğin ikinci yarısı kullanılarak test

edilmiştir. Bu durumda, yapay sinir ağının eğitim ve test verileri aynı veri kümesinden alınmaktadır. Eğitilmiş olan yapay sinir ağı, Excite veri kümesinden alınan örneğin ikinci yarısında test edildikten sonra, bu şekliyle, Durum 2'de, Fast veri kümesinden alınan örneğin ikinci yarısında da test edilmiştir. Yapay sinir ağı, 50 kez eğitilmiş ve bu eğitimler için test edilmiştir. Her test sonucunda, performans ölçüleri, P, R ve F_β değerleri hesaplanmıştır.

Durum 2: Eğitim: Excite – Test: Fast

Yapay sinir ağı, Excite veri kümesinden alınan örneğin ilk yarısı kullanılarak eğitilmiş, Fast veri kümesinden alınan örneğin ikinci yarısı kullanılarak test edilmiştir. Bu durumda, yapay sinir ağının eğitim ve test verileri farklı veri kümelerinden alınmaktadır. Eğitilmiş olan yapay sinir ağı, Durum 1'de, Excite veri kümesinden alınan örneğin ikinci yarısında da test edilmiş olduğu için, Fast veri kümesinden alınan örneğin ikinci yarısında test edildikten sonra, sonuçlar karşılaştırılabilir. Yapay sinir ağı, 50 kez eğitilmiş ve bu eğitimler için test edilmiştir. Her test sonucunda, performans ölçüleri, P, R ve F_β değerleri hesaplanmıştır.

Durum 3: Eğitim: Fast – Test: Excite

Yapay sinir ağı, Fast veri kümesinden alınan örneğin ilk yarısı kullanılarak eğitilmiş, Excite veri kümesinden alınan örneğin ikinci yarısı kullanılarak test edilmiştir. Bu durumda, yapay sinir ağının eğitim ve test verileri farklı veri kümelerinden alınmaktadır. Eğitilmiş olan yapay sinir ağı, Excite veri kümesinden alınan örneğin ikinci yarısında test edildikten sonra, bu şekliyle, Durum 4'te, Fast veri kümesinden alınan örneğin ikinci yarısında da test edilmiştir. Yapay sinir ağı, 50 kez eğitilmiş ve bu eğitimler için test edilmiştir. Her test sonucunda, performans ölçüleri, P, R ve F_β değerleri hesaplanmıştır.

Durum 4: Eğitim: Fast – Test: Fast

Yapay sinir ağı, Fast veri kümesinden alınan örneğin ilk yarısı kullanılarak eğitilmiş, Fast veri kümesinden alınan örneğin ikinci yarısı kullanılarak test edilmiştir. Bu durumda, yapay sinir ağının eğitim ve test verileri aynı veri kümesinden alınmaktadır. Eğitilmiş olan yapay sinir ağı, Durum 3'te, Excite veri kümesinden alınan örneğin ikinci yarısında da test edilmiş olduğu için, Fast veri kümesinden alınan örneğin ikinci yarısında test edildikten sonra, sonuçlar karşılaştırılabilir. Yapay sinir ağı, 50 kez eğitilmiş ve bu eğitimler için test edilmiştir. Her test sonucunda, performans ölçüleri, P, R ve F_β değerleri hesaplanmıştır.

3.2.5 – Bir Düzeltme

He ve arkadaşlarının (2002) çalışmasında, performans ölçülerinden duyarlık,

$$P = \frac{N_{shift\&correct}}{N_{shift} + N_{contin}} \tag{3.107}$$

şeklinde belirtilmiştir. Algoritma tarafından doğru olarak belirlenen konu değişikliği sayısının ($N_{shift\&correct}$) alabileceği en büyük değer, insanlar tarafından belirlenen konu değişikliği sayısı ($N_{true\ shift}$) kadar olabileceğinden (tüm konu değişiklikleri doğru olarak belirlendiğinde), bu formülde bir hata olduğu görülmektedir.

Burada, $N_{shift} + N_{contin} = N_{true\ shift} + N_{true\ contin}$ eşitliği ve $N_{shift\&correct} \le N_{true\ shift}$ eşitsizliği her zaman geçerli olduğundan P için üst sınır,

$$P \le \frac{N_{true\ shift}}{N_{true\ shift} + N_{true\ contin}} \tag{3.108}$$

formülüyle belirlenebilir.

He ve arkadaşları (2002) çalışmalarında, "konu değişikliği var" sayısı 625 ve "konu değişikliği yok" sayısı 3574'tür. Bu yüzden, P için üst sınır 0,149 (625 / (625+3574)) olmaktadır. Öte yandan, genetik algoritmanın çalıştırılması sonucunda P değerinin 0,6543 olarak bulunduğu belirtilmiştir. Bu değer, P için üst sınırı aştığından, duyarlık formülünde bir hata olduğu görülmektedir.

Doğru formülasyonu belirlemek için deneyler yapıldığında, He ve arkadaşlarının (2002) çalışmalarında, F_β en büyük değerini, $N_{shift\&correct}$ değerinin bir önemi olmaksızın, $N_{shift\&correct}$'in $N_{true\ shift}$'e eşit olduğunda almaktadır. Diğer bir deyişle, tüm sorgular "konu değişikliği var" olarak işaretlenirse (bütün doğru konu değişikliklerini doğru olarak tahmin etmeyi garanti etmek için), bu durumda F_β en yüksek değerini alacaktır. Bu matematiksel olarak aşağıda gösterilmiştir.

$N_{shift\&correct} = N_{true\ shift}$ olduğunda duyarlık ve anma değerleri,

$$P = \frac{N_{shift\&correct}}{N_{shift} + N_{contin}} = \frac{N_{true\ shift}}{N_{shift} + N_{contin}} \tag{3.109}$$

ve

$$R = \frac{N_{shift\&correct}}{N_{true\ shift}} = \frac{N_{true\ shift}}{N_{true\ shift}} = 1 \tag{3.110}$$

olacaktır. Burada, P en büyük değerini ve R de yine alabileceği en büyük değer olan 1 değerini almıştır. Buna göre F_β fonksiyonu güncellenirse,

$$F_\beta = \frac{\left(1+\beta^2\right)PR}{\beta^2 P + R} = \frac{\left(1+\beta^2\right)P}{\beta^2 P + 1} = \frac{1+\beta^2}{\beta^2 + 1/P} \tag{3.111}$$

olur. P en büyük değerini aldığından, $1/P$ en küçük değerini almıştır. Sonuç olarak, verilen bir β değeri için $1/P$ en küçük değerini aldığında F_β fonksiyonu en büyük değerini almaktadır.

He ve arkadaşlarının (2002) çalışmasında verilen formülasyonlara göre en yüksek amaç fonksiyonu değeri, karmaşık hesaplamalar yapmadan, basitçe tüm sorgular "konu değişikliği var" şeklinde işaretlenerek elde edilebilir. Yazarların kullandıkları P formülünün, verilen formülden farklı olduğu açıkça görülmektedir.

Yazarların kullandıkları P formülü bilinmemekle birlikte, bu formül içinde N_{contin} bulunmaması gerektiği düşünülmektedir. N_{contin} olmadığı durumda P formülü

için üst sınır 1 olmaktadır. Bu yüzden, bu çalışmada, P için aşağıdaki formül kullanılmıştır:

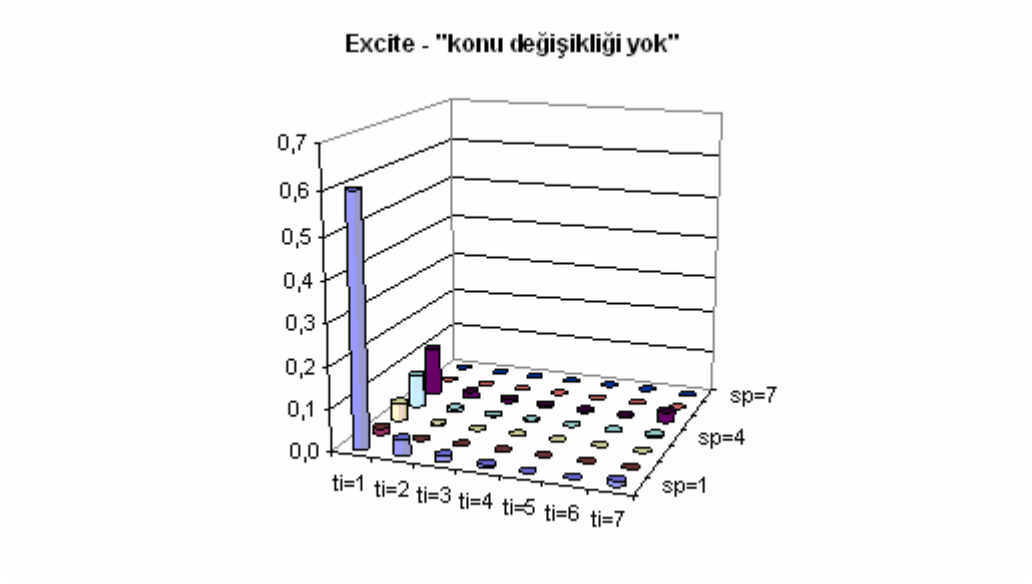
$$P = \frac{N_{shift\&correct}}{N_{shift}} \tag{3.112}$$

4 – ARAŞTIRMA SONUÇLARI

4.1 – Zaman Aralığı ve Arama Yapısı Verilerinin Etkileri

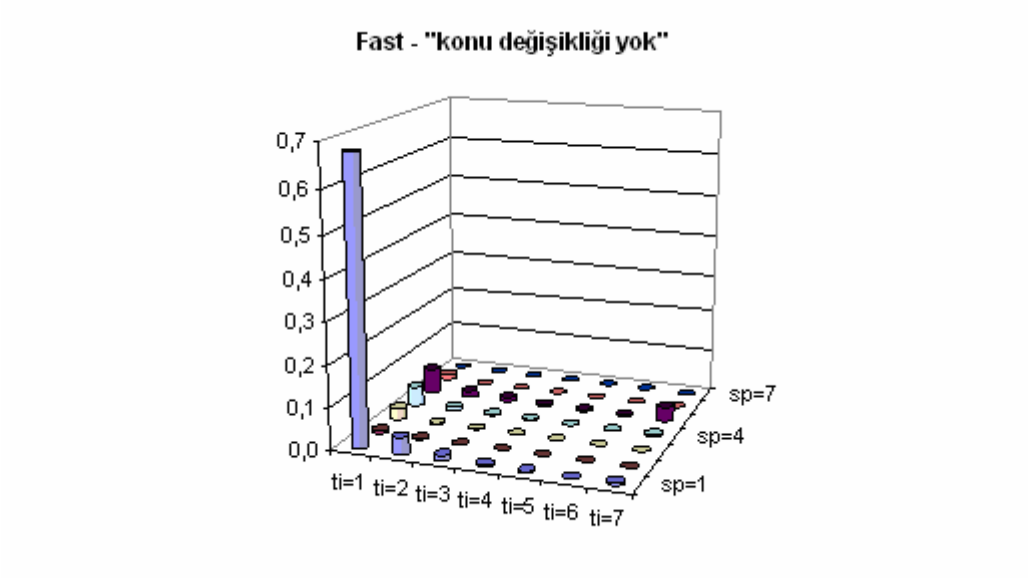
He ve arkadaşları (2002) yılında yapmış oldukları çalışmada, zaman aralığı (ti) ve arama yapısı (sp) verilerinin konu değişikliklerini tespit etmek amacıyla kullanılabileceğini belirtmişlerdir. Öte yandan, bu çalışmada kullanılan verilerin de benzer özellikler taşıdığını göstermek amacıyla, uygulanan yaklaşımlarının sonuçlarını incelemeden önce, bu yaklaşımlarda kullanılan veriler olan zaman aralığı ve arama yapısı verilerinin farklı değerlerinin "konu değişikliği yok" ve "konu değişikliği var" durumlarıyla olan ilişkisi incelenmiştir. Bu amaçla, her zaman aralığı (ti) ve arama yapısı (sp) sınıfına karşılık gelen "konu değişikliği yok" ve "konu değişikliği var" durumların sayısı, toplam "konu değişikliği yok" ve "konu değişikliği var" durumlarının sayısına bölünerek, ilgili zaman aralığı-arama yapısı (ti-sp) ikilisine karşılık gelen "konu değişikliği yok" ve "konu değişikliği var" oranları elde edilmiştir. Sonuçlar Excite ve Fast arama motorlarından alınan örnekler için verilmiştir.

Excite eğitim örneği kullanılarak elde edilen "konu değişikliği yok" oranları Şekil 4.1'de görülmektedir.



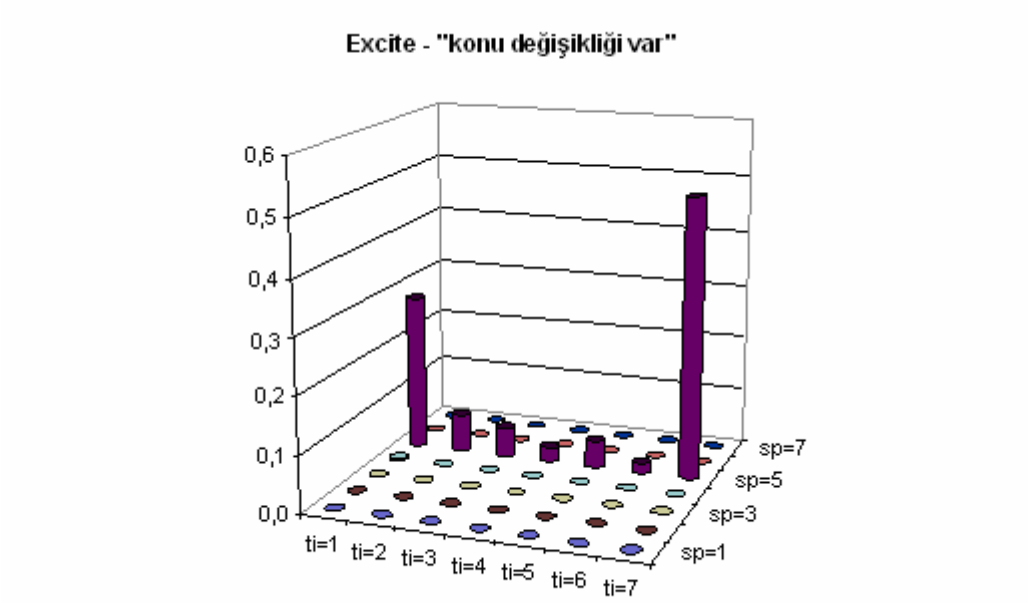
Şekil 4.1: Excite - "konu değişikliği yok" Oranları

Benzer şekilde, Fast eğitim örneği kullanılarak elde edilen "konu değişikliği yok" oranları Şekil 4.2'de görülmektedir.



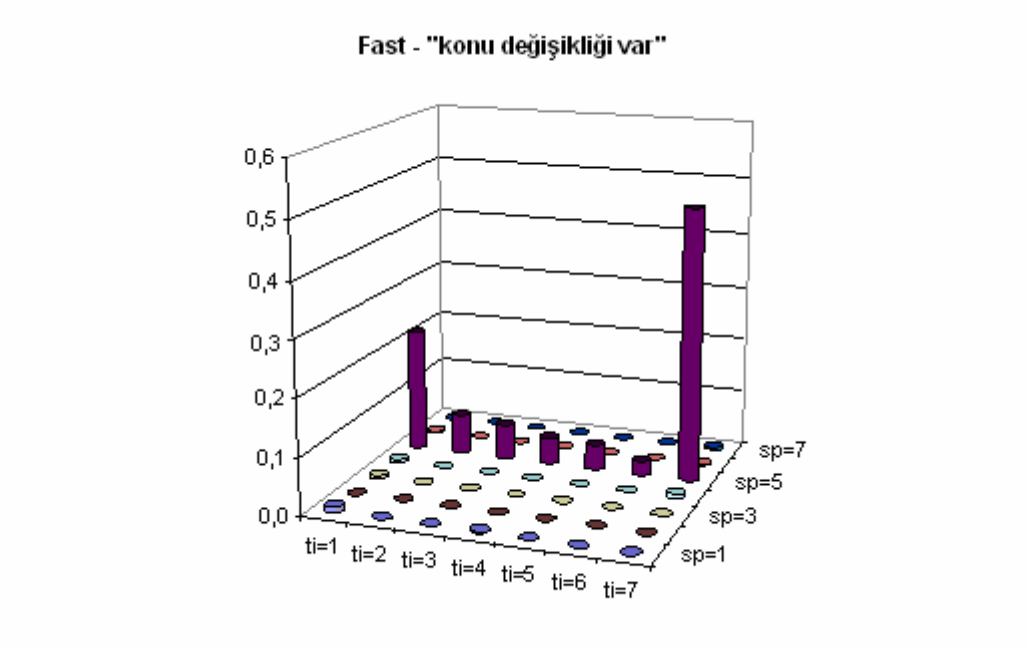
Şekil 4.2: Fast - "konu değişikliği yok" Oranları

Excite eğitim örneği kullanılarak elde edilen "konu değişikliği var" oranları Şekil 4.3'te görülmektedir.



Şekil 4.3: Excite - "konu değişikliği var" Oranları

Benzer şekilde Fast eğitim örneği kullanılarak elde edilen "konu değişikliği var" oranları de Şekil 4.4'te görülmektedir.



Şekil 4.4: Fast - "konu değişikliği var" Oranları

"konu değişikliği yok" oranları açısından Excite ve Fast verileri arasında benzer yapıların olduğu görülmektedir. Şekil 4.1 ve 4.2'de sırasıyla Excite ve Fast için "konu değişikliği yok" oranları görülmektedir. Bunlar incelendiğinde, "konu değişikliği yok" olan durumların büyük bir çoğunluğunun ti=1 (0-5 dakika) ve sp=1 (sonraki sayfa – next page) sınıflarında gerçekleştiği görülmektedir. sp=3 (özelleştirme – specialization), sp=4 (düzenleme – reformulation) ve sp=5 (yeni – new) sınıflarında da belirli zaman aralığı sınıfları için "konu değişikliği yok" durumları bulunmaktadır.

"konu değişikliği var" oranları açısından da Excite ve Fast verileri arasında benzer yapıların olduğu görülmektedir. Şekil 4.3 ve 4.4'te sırasıyla Excite ve Fast için "konu değişikliği var" oranları görülmektedir. Bunlar incelendiğinde, "konu değişikliği var" olan durumların büyük bir çoğunluğunun sp=5 (yeni – new) sınıfında gerçekleştiği görülmektedir. Bu sınıfa karşılık gelen zaman aralığı sınıfları incelendiğinde ise en yoğun "konu değişikliği var" durumu ti=7 (>30 dakika) sınıfında görülmektedir. Daha küçük zaman aralığı değerleri için bu yoğunluk azalmakta, ti=1 (0-5 dakika) sınıfında ise tekrar artış olmaktadır.

Sonuçlardan, He ve arkadaşlarının (2002) çalışmalarında belirttikleri gibi zaman aralığı ve arama yapısı sınıflarının "konu değişikliği yok" ve "konu değişikliği var"

durumlarıyla ilişkili oldukları görülmektedir. Öte yandan, sadece zaman aralığı veya sadece arama yapısı değerini dikkate almak yerine, bunların ikisinin birlikte incelenmesi daha iyi sonuçlar elde edilmesini sağlayacaktır.

4.2 – Dempster-Shafer Teorisi ve Genetik Algoritmalar Yaklaşımı

Öncelikle, zaman aralığı ve arama yapısına göre "konu değişikliği yok" (contin) ve "konu değişikliği var" (shift) olasılıkları hesaplanmıştır. Olasılıklar, Çizelge 4.1 ve 4.2'de verilmiştir.

Çizelge 4.1: Zaman Aralığına Göre Olasılıklar (Excite)

ti (dakika)	Konu-İçi	Konular-Arası	P(ti)	P(shift | ti)	P(contin | ti)
0-5	3001	77	0,8072	0,0250	0,9750
5-10	218	18	0,0619	0,0763	0,9237
10-15	85	14	0,0260	0,1414	0,8586
15-20	47	7	0,0142	0,1296	0,8704
20-25	22	13	0,0092	0,3714	0,6286
25-30	20	5	0,0066	0,2000	0,8000
30+	151	135	0,0750	0,4270	0,5280
Toplam	3544	269			

Çizelge 4.2: Arama Yapısına Göre Olasılıklar (Excite)

sp Sınıfı	Konu-İçi	Konular-Arası	P(sp)	P(shift | sp)	P(contin | sp)
Sonraki Sayfa	2371	0	0,6218	0,0000	1,0000
Genelleştirme	58	0	0,0152	0,0000	1,0000
Özelleştirme	166	0	0,0435	0,0000	1,0000
Düzenleme	327	1	0,0660	0,0030	0,9970
Yeni	622	268	0,2334	0,3011	0,6989
İlişkili G.B.	0	0	0,0000	-	-
Diğer	0	0	0,0000	-	-
Toplam	3544	269			

Benzer bir çalışma yapmış olan He ve arkadaşları (2002) çalışmalarında Reuters arama motoru veri kümesinden alınan bir örneği kullanmışlardır. Yazarların çalışmalarında vermiş oldukları Reuters arama motoru örneğinden elde edilmiş olan bu

olasılıklar, bu çalışmada da karşılaştırma yapmak amacıyla kullanılmıştır. He ve arkadaşlarının (2002) çalışmasında verilen zaman aralığına göre olasılıklar ve arama yapısına göre olasılıklar Çizelge 4.3 ve 4.4'te görülmektedir.

Çizelge 4.3: Zaman Aralığına Göre Olasılıklar (Reuters) *

ti (dakika)	Konu-İçi	Konular-Arası	P(ti)	P(shift | ti)	P(contin | ti)
0-5	3264	173	0,8185	0,0503	0,9497
5-10	108	31	0,0332	0,2230	0,7770
10-15	36	25	0,0145	0,4098	0,5902
15-20	30	19	0,0117	0,3878	0,6122
20-25	15	15	0,0071	0,5000	0,5000
25-30	13	11	0,0057	0,4583	0,5417
30+	108	351	0,1093	0,7647	0,2353
Toplam	3574	625			
* He ve arkadaşlarının (2002) çalışmasındaki verilmiş olan olasılıklardır.					

Çizelge 4.4: Arama Yapısına Göre Olasılıklar (Reuters) *

sp Sınıfı	Konu-İçi	Konular-Arası	P(sp)	P(shift | sp)	P(contin | sp)
Sonraki Sayfa	2024	0	0,4820	0,0000	1,0000
Genelleştirme	121	0	0,0288	0,0000	1,0000
Özelleştirme	286	0	0,0681	0,0000	1,0000
Düzenleme	280	15	0,0703	0,0508	0,9492
Tekrarlama	404	0	0,0962	0,0000	1,0000
Yeni	312	605	0,2184	0,6587	0,3413
İlişkili G.B.	9	0	0,0021	0,0000	1,0000
Diğer	138	5	0,0341	0,0350	0,9650
Toplam	3574	625			
* He ve arkadaşlarının (2002) çalışmasındaki verilmiş olan olasılıklardır.					

Genetik algoritma, Excite veri örneğinin ilk 5014 sorgusu üzerinde çalıştırılmıştır. Bu sayı, herhangi bir oturum bölünmeksizin Excite veri örneğinin yaklaşık yarısını oluşturmaktadır. Her bir kullanıcı oturumu sonundaki sorgu, zaman aralığı veya arama yapısının belirlenmesinde kullanılamayacağından, bu 5014 sorgunun tamamının kullanılmadığına dikkat edilmelidir. Kullanıcı oturumundaki son sorgudan sonra gelen ardışık bir sorgu daha olmadığından (aynı oturuma ait), bu sorgu için zaman aralığı veya arama yapısı belirlenemez. Veri örneğinin ilk kısmında 1201 kullanıcı

oturumu bulunmaktadır. Veri örneğinin ikinci kısmındaki sorgu sayısı ise (test kısmı) 5014'ten 3813'e inmiştir. İnsanlar tarafından konu değişiklikleri belirlendikten sonra, 3812 sorgu içerisinde 2544 "konu değişikliği yok" ve 269 "konu değişikliği var" durumu olduğu belirlenmiştir.

Durum 1, 3 ve 5 için Excite, Durum 2, 4 ve 6 için Reuters olasılıkları kullanılmıştır. Konu değişikliği belirleme algoritması, ilgili w_{ti}, w_{sp} ve t_{shift} değerleri için çalıştırılmış ve P, R ve F_β değerleriyle, A ve B tipi hata sayıları hesaplanmıştır. Durum 1-6 sonuçları, Çizelge 4.5 ve 4.6'da görülmektedir.

Çizelge 4.5: Excite Olasılıklarından Elde Edilen Sonuçlar

	w_{ti}	w_{sp}	t_{shift}	P	R	A Tipi Hata	B Tipi Hata	F_β
D1	0.5684	0.7236	0,1162	0,29	0,763	284	36	0,508
D3	0,8640	0,9360	0,3450	-*	0,000	0	152	0,000
D5-R1	0,6500	0,0310	0,0390	0,145	0,770	691	35	0,331
D5-R2	0,5160	0,7210	0,1220	0,161	0,993	789	1	0,383
D5-R3	0,8500	0,7700	0,9050	-*	0,000	0	152	0,000
D5-R4	0,7170	0,2000	0,3620	-*	0,000	0	152	0,000
D5-R5	0,7970	0,0360	0,5630	-*	0,000	0	152	0,000
D5-R6	0,5350	0,8680	0,0920	0,161	0,993	789	1	0,383
D5-R7	0,8500	0,0870	0,2250	0,241	0,546	261	69	0,393
D5-R8	0,7140	0,9690	0,2830	0,354	0,526	146	72	0,458
D5-R9	0,1460	0,9890	0,5140	-*	0,000	0	152	0,000
D5-R10	0,8930	0,0710	0,4740	-*	0,000	0	152	0,000

* Sıfıra bölmeden dolayı duyarlık değeri hesaplanamaz.

Durum 1: Excite veri örneğinden hesaplanan olasılıklar ve genetik algoritma ile belirlenmiş olan w_{ti}, w_{sp} ve t_{shift} parametreleri kullanılarak He ve arkadaşlarının (2002) çalışmasındaki yaklaşımın tekrarlanması.

Excite veri örneğinin ikinci yarısı üzerinde, w_{ti}, w_{sp} ve t_{shift} parametreleri kullanılarak Durum 1 için konu değişikliği belirleme algoritması çalıştırıldığında, P, R ve F_β değerleri, Çizelge 4.5'te görüldüğü gibi sırasıyla 0,290, 0,7632 ve 0,5081 olarak bulunmuştur. A tipi ve B tipi hatalar da sırasıyla 284 ve 36 olarak bulunmuştur. Excite olasılıkları ve parametreleri ve He ve arkadaşlarının (2002) çalışmalarındaki konu

değişikliği belirleme algoritması kullanılarak bu yazarların çalışmasında belirtilen 0,8183'lük F_β başarı düzeyine ulaşılamamıştır. Bu sonuçtan, He ve arkadaşlarının yaklaşımının veriden etkilendiği sonucu çıkarılabilir. Öte yandan, Reuters ile kıyaslandığında Excite veri örneğindeki konu değişikliği sayısının oldukça az olduğu görülmektedir (Reuters – 325 veya %14,9, Excite – 269 veya %7). Bu Excite veri örneğinde algoritma performansının düşmesine yol açmış olabilir.

Çizelge 4.6: Reuters Olasılıklarından Elde Edilen Sonuçlar

	w_{ti}	w_{sp}	t_{shift}	P	R	A Tipi Hata	B Tipi Hata	F_β
D2	0,110	0,942	0,602	0,290	0,763	284	36	0,508
D4	0,864	0,936	0,345	0,290	0,763	284	36	0,508
D6-R1	0,650	0,031	0,039	0,113	1,000	1196	0	0,292
D6-R2	0,516	0,721	0,122	0,149	1,000	870	0	0,362
D6-R3	0,850	0,770	0,905	-*	0,000	0	152	0,000
D6-R4	0,717	0,200	0,362	0,255	0,566	251	66	0,412
D6-R5	0,797	0,036	0,563	0,263	0,533	227	71	0,405
D6-R6	0,535	0,868	0,092	0,158	1,000	813	0	0,378
D6-R7	0,850	0,087	0,225	0,190	0,678	439	49	0,379
D6-R8	0,714	0,969	0,283	0,161	0,993	789	1	0,383
D6-R9	0,146	0,989	0,514	0,161	0,993	789	1	0,383
D6-R10	0,893	0,071	0,474	0,263	0,533	227	71	0,405

* Sıfıra bölmeden dolayı duyarlık değeri hesaplanamaz.

Durum 2: Reuters veri örneğinden hesaplanan olasılıklar ve genetik algoritma ile belirlenmiş olan w_{ti}, w_{sp} ve t_{shift} parametreleri kullanılarak He ve arkadaşlarının (2002) çalışmasındaki yaklaşımın tekrarlanması.

Excite veri örneğinin ikinci yarısı üzerinde, w_{ti}, w_{sp} ve t_{shift} parametreleri kullanılarak Durum 2 için konu değişikliği belirleme algoritması çalıştırıldığında, P, R ve F_β değerleri, Çizelge 4.6'da görüldüğü gibi sırasıyla 0,290, 0,7632 ve 0,5081 olarak bulunmuştur. A tipi ve B tipi hatalar da sırasıyla 284 ve 36 olarak bulunmuştur. Excite veri örneği yerine Reuters veri örneği olasılıklarının kullanılması genetik algoritmanın farklı parametre değerleri seçmesine neden olmuştur. Bu durumda elde edilen sonuç ilginçtir. Excite yerine Reuters veri örneğinden elde edilmiş olan olasılıklar

kullanılmasına rağmen, konu değişikliği belirleme algoritmasının performansında herhangi bir değişiklik olmamıştır. P, R ve F_β performans ölçülerinin, farklı veri örneğinden elde edilmiş olan olasılıklardan ve w_{ti}, w_{sp} ve t_{shift} parametrelerinden etkilenmediği görülmektedir. Bu durumdan elde edilen sonuçlara göre, w_{ti}, w_{sp} ve t_{shift} parametrelerinin değerlerinin yakın olduğu durumda, veri örneği olasılıklarının konu değişikliği belirlenme algoritmasının performansı üzerinde önemli bir faktör olmadığı söylenebilir. Bununla birlikte, olasılıkların algoritma performansı üzerindeki etkileri sonraki kısımlarda istatistiksel olarak incelenmiştir.

Ek olarak, şu sorular sorulabilir: P, R ve F_β, "konu değişikliği yok" ve "konu değişikliği var" olasılıklarına bağlı mıdır? P, R ve F_β, performans ölçüleri w_{ti}, w_{sp} ve t_{shift} parametrelerine bağlı mıdır? Excite veri örneğinin kendi olasılıkları yerine, Reuters olasılıklarını kullanarak daha yüksek başarı elde edilebilir mi? Bir veri örneğinin (Excite) kendi olasılıklarının kullanmanın, başka bir veri örneğinin olasılıklarının kullanmaktan daha iyi sonuç vermesi gerekmez mi?

Durum 3: Excite veri örneğinden hesaplanan olasılıklar ve He ve arkadaşlarının (2002) çalışmasında verilmiş olan w_{ti}, w_{sp} ve t_{shift} parametreleri kullanılarak He ve arkadaşlarının (2002) çalışmasındaki yaklaşımın tekrarlanması.

Excite veri örneğinin ikinci yarısı üzerinde, w_{ti}, w_{sp} ve t_{shift} parametreleri kullanılarak Durum 3 için konu değişikliği belirleme algoritması çalıştırıldığında, P, R ve F_β değerleri, Çizelge 4.5'te görüldüğü gibi sırasıyla "-", 0 ve 0 olarak bulunmuştur. A tipi ve B tipi hatalar da sırasıyla 0 ve 152 olarak bulunmuştur. Bu senaryo, tüm sorguları "konu değişikliği yok" olarak işaretlediği için konu değişikliklerini belirlemede çok başarısızdır. Bu durumun sonuçlarına göre He ve arkadaşlarının (2002) çalışmasında verilmiş olan w_{ti}, w_{sp} ve t_{shift} parametre değerlerinin evrensel olarak kabul edilebilecek değerler olmadığı söylenebilir.

Durum 4: Reuters veri örneğinden hesaplanan olasılıklar ve He ve arkadaşlarının (2002) çalışmasında verilmiş olan w_{ti}, w_{sp} ve t_{shift} parametreleri kullanılarak He ve arkadaşlarının (2002) çalışmasındaki yaklaşımın tekrarlanması.

Excite veri örneğinin ikinci yarısı üzerinde, w_{ti}, w_{sp} ve t_{shift} parametreleri kullanılarak Durum 4 için konu değişikliği belirleme algoritması çalıştırıldığında, P, R ve F_β değerleri, Çizelge 4.6'da görüldüğü gibi sırasıyla 0,290, 0,7632 ve 0,5080 olarak bulunmuştur. A tipi ve B tipi hatalar da sırasıyla 284 ve 36 olarak bulunmuştur.

Durum 1, 2 ve 4 için w_{ti}, w_{sp} ve t_{shift} parametre değerleri oldukça farklı olmasına rağmen, bu durumlara ilişkin P, R ve F_β değerleri aynıdır. Burada, performans ölçüleri parametrelerden etkilenmiyor gibi görünse bile, Durum 3 bunun tersini göstermektedir. Bu sorunun cevabı ilerleyen kısımlarda, deneysel tasarım ve çoklu regresyon modeli kullanılarak belirlenmiştir.

Durum 5: Excite veri örneğinden hesaplanan olasılıklar ve rasgele üretilmiş olan w_{ti}, w_{sp} ve t_{shift} parametreleri kullanılarak He ve arkadaşlarının (2002) çalışmasındaki yaklaşımın tekrarlanması.

Bu durumda, rasgele üretilmiş w_{ti}, w_{sp} ve t_{shift} parametreleri kullanılarak algoritma çalıştırılmıştır. Excite veri örneğinin ikinci yarısı üzerinde, her seferinde rasgele üretilmiş w_{ti}, w_{sp} ve t_{shift} parametreleri kullanılarak Durum 5 için konu değişikliği belirleme algoritması 10 kez çalıştırılmıştır. Bu 10 simülasyonun 5 tanesinde, konu değişikliği belirleme algoritması çok başarısız olmuştur (F_β = 0). Geri kalan durumlarda algoritma, 0,331 ile 0,458 arasında değişen F_β değerleri vermiştir. Buradan, algoritmanın rasgele parametre değerleri için bile belirli bir düzeyde başarılı olabildiği görülmektedir (Ortalama F_β değeri 0,198). w_{ti}, w_{sp} ve t_{shift} parametreleri ve amaç fonksiyonu arasındaki ilişkinin incelenmesi gerekmektedir.

Durum 6: Reuters veri örneğinden hesaplanan olasılıklar ve rasgele üretilmiş olan w_{ti}, w_{sp} ve t_{shift} parametreleri kullanılarak He ve arkadaşlarının (2002) çalışmasındaki yaklaşımın tekrarlanması.

Durum 5'tekine benzer şekilde, konu değişikliği belirleme algoritması Excite veri örneğinin ikinci yarısı üzerinde, rasgele üretilmiş w_{ti}, w_{sp} ve t_{shift} parametreleri kullanılarak çalıştırılmıştır. Durum 5'ten farklı olarak, Reuters veri örneğinin olasılıkları kullanılmıştır. Sonuçlar Çizelge 4.6'da görülmektedir.

Tek bir durum dışında, konu değişikliği belirleme algoritmasının belirli bir düzeyde başarılı olduğu görülmüştür (F_β değeri 0,292 ile 0,405 arasında değişmektedir). Buradan, algoritmanın rasgele parametre değerleri için bile belirli bir düzeyde başarılı olabildiği görülmektedir (Ortalama F_β değeri 0,34). Burada da w_{ti}, w_{sp} ve t_{shift} parametreleri ve amaç fonksiyonu arasındaki ilişkinin incelenmesi gerekmektedir.

Durum 5 ve Durum 6 kıyaslandığında, Excite yerine Reuters olasılıkları kullanıldığında F_β değerinde ortalama olarak yaklaşık 0,14'lük bir artış olduğu görülmektedir. Sonraki kısımda, olasılıkların F_β üzerindeki etkisinin istatistiksel olarak anlamlılığı incelenmiştir.

4.2.1 – Örnek Olasılıkları ile F_β Arasındaki İlişkinin İstatistiksel Testi

Durum 3 ve 5 için ortalama F_β değeri 0,18 iken Durum 4 ve 6 için aynı değer 0,36 olmaktadır. Reuters verisinin uygulandığı Durum 4 ve Durum 6 için ortalama F_β değeri Durum 3 ve 5 için elde edilen aynı değerden daha yüksektir. Ortalama F_β değerleri arasındaki farkın test edilmesi için çift yönlü t-testi uygulandıktan sonra –2,49 gibi bir t istatistiği elde edilmiştir. Bu test için kritik t istatistiği –2,23'tür. Ortalama F_β değerleri arasındaki fark %95 anlamlılık düzeyinde istatistiksel olarak anlamlıdır. Reuters olasılıklarının istatistiksel olarak Excite olasılıklarından daha iyi sonuç verdiği açıktır. Bu ilginç bir sonuçtur, çünkü doğal olarak, konu değişikliği belirleme

algoritmasını herhangi bir veri örneğine, bu veri örneğinin kendi olasılıklarıyla uygulamanın daha iyi sonuç vermesi beklenmektedir. Bu ilginç durumu açıklamanın iki yolu vardır: (1) amaç fonksiyonu değerini enbüyüklemek için evrensel bir olasılıklar kümesi vardır ve Reuters olasılıkları bu olasılıklara Excite olasılıklarına göre daha yakındır veya (2) Durum 6'da olduğu gibi rasgele parametreler oldukça başarılıdır ve konu belirleme için olasılık-temelli Dempster-Shafer teorisinin kullanımının geçerliliğiyle ilgili soru işaretleri bulunmaktadır.

4.2.2 – Parametreler ile Performans Ölçüleri Arasındaki İlişkinin Belirlenmesi

Burada, konu değişikliği belirleme algoritmasının parametreleri (w_{ti}, w_{sp} ağırlıkları ve t_{shift} eşik değeri) ile performans ölçüleri (P ve R) arasındaki ilişkinin test edilmesi için daha detaylı bir analiz yapılmıştır. F_β, P ve R kullanılarak hesaplandığı ve doğrudan w_{ti}, w_{sp} ve t_{shift} parametrelerine bağlı olmadığı için çoklu doğrusal regresyon denkleminin üçüncü bağımlı değişkeni olarak alınmamıştır. Daha önce belirtilen çoklu regresyon denklemi ve deneysel tasarım kullanılmıştır. Ayrıca bağımlı değişkenler için varyans analizi de yapılmıştır.

Regresyon denklemi için belirleme katsayısı %27,9'dur. Bu düşük bir belirleme katsayısı değeridir ve P'yi bağımsız değişkenler ve bunların kesişim etkileriyle açıklamada yetersiz kalmaktadır. Regresyon modeli için F değeri, ilgili $F_{0,05,18,197} \sim 1,61$ değerini aşan 4,23'tür ve bu yüzden regresyon istatistiksel olarak anlamlıdır. Bu sonuç, P ile parametreler ve bunların kesişimleri arasındaki etkinin anlamlı olduğunu göstermektedir. Varyans analizi aynı zamanda her bir etkinin P üzerindeki anlamlılığını test etmektedir. Her bir etkinin anlamlılığını test eden kritik F değeri, $F_{0,05,1,197} \sim 3,84$'tür. Etkilerin hiçbiri kritik F değerini aşmamaktadır. Etkiler birlikte P üzerinde zayıf bir etkiye sahiptirler, ancak etkilerin hiçbiri tek başına P üzerinde anlamlı bir etkiye sahip değildir.

R için benzer analizlerin sonuçlarına bakıldığında, regresyon denkleminin R üzerindeki değişikliğin %88,6'sını açıkladığı görülmektedir. Regresyon modeli için F değeri 84,99'dur ve bu regresyon denkleminin istatistiksel olarak anlamlı olduğunu

göstermektedir. Her bir etki için varyans analizine bakıldığında, t_{shift} ana etkisinin ve $w_{ti} * t_{shit}^2$ kesişiminin kritik F değeri 3,84'ü aştığı görülmektedir. t_{shift}'in R formülünün yapısı gereği R üzerinde etkili olması beklenmektedir. t_{shift}, R'nin payı olan $N_{shift\&correct}$'i doğrudan etkilemektedir. Herhangi bir veri örneği için $N_{true\ shift}$ sabit olduğundan, R'nin paydası sabittir. Böylece, t_{shift} 0'a yaklaştığında, R de 1'e yaklaşmaktadır.

P ve R performans ölçülerinin bağımlı değişkenler olduğu, çoklu doğrusal regresyon denklemi ve varyans analizi sonuçları Çizelge 4.7 ve 4.8'de görülmektedir.

Çizelge 4.7: Regresyon ve Varyans Analizi (Bağımlı Değişken P)

Değişim Kaynağı	Katsayı	F_β	Değişim Kaynağı	Katsayı	F_β
Sabit	0,0137	-	$w_{ti}^2 \times w_{sp}$	-0,6371	0,0031
w_{ti}	0,1662	0,0845	$w_{ti}^2 \times t_{shift}$	0,7657	0,0054
w_{sp}	0,0534	0,0054	$w_{sp}^2 \times t_{shift}$	-0,0278	0,0005
t_{shift}	-0,2575	0,3998	$w_{ti}^2 \times w_{sp}^2$	0,6482	0,0101
$w_{ti} \times w_{sp}$	0,7066	0,0093	$w_{ti}^2 \times t_{shift}^2$	-0,6138	0,00562
$w_{ti} \times t_{shift}$	-0,5122	0,0416	$w_{sp}^2 \times t_{shift}^2$	0,0777	0,0002
$w_{sp} \times t_{shift}$	-0,1125	0,0087	w_{ti}^2	0,0743	0,00002
$w_{ti} \times w_{sp}^2$	-0,6955	0,0019	w_{sp}^2	-0,0022	0,0003
$w_{ti} \times t_{shit}^2$	0,2054	0,0199	t_{shift}^2	0,1727	0,0006
$w_{sp} \times t_{shift}^2$	-0,0283	0,0046			

	DF	SS	F
Kaynak			
Regresyon	18	0,6016	4,23
Geri Kalan Hata	197	1,5574	
Toplam	215	2,1591	

Deneysel tasarım ve varyans analizi sonuçları w_{ti}, w_{sp} ve t_{shift} parametrelerinin P üzerinde zayıf, R üzerinde ise daha güçlü bir etkiye sahip olduklarını

göstermektedir. Bu sonuç şu şekilde yorumlanabilir: Durum 5 ve 6'da, parametreler rasgele seçilmiş olmasına rağmen belirli bir başarı düzeyi elde edilmişti. Bununla birlikte, Durum 1, 2 ve 4'te F_β değeri 0,508 olarak bulunmuştu. w_{ti}, w_{sp} ve t_{shift} parametreleri için rasgele değerler yerine belirli değerler kullanılarak, F_β'da gelişme sağlanmıştır. Bununla birlikte, w_{ti}, w_{sp} ve t_{shift} parametreleri için rasgele değerlerle bile 0,458 gibi bir F_β değeri elde edildiğinden, bu gelişme zayıftır. Bu zayıf etkinin sebebi; F_β'nın sadece P ve R'ye bağlı olarak değişmesi ve sadece R'nin w_{ti}, w_{sp} ve t_{shift} parametrelerine güçlü olarak bağlı olması ve bu yüzden de F_β'nın da bu parametrelere zayıf olarak bağlı olmasındandır.

Çizelge 4.8: Regresyon ve Varyans Analizi (Bağımlı Değişken R)

Değişim Kaynağı	Katsayı	F_β	Değişim Kaynağı	Katsayı	F_β
Sabit	0,82153		$w_{ti}^2 \times w_{sp}$	-0,7960	0,0000
w_{ti}	0,1559	0,1119	$w_{ti}^2 \times t_{shift}$	0,9660	0,0077
w_{sp}	-0,0465	0,1086	$w_{sp}^2 \times t_{shift}$	0,0110	0,0012
t_{shift}	-2,8033	21,7200	$w_{ti}^2 \times w_{sp}^2$	0,6484	0,0009
$w_{ti} \times w_{sp}$	0,7330	0,3074	$w_{ti}^2 \times t_{shift}^2$	-0,8890	0,5624
$w_{ti} \times t_{shift}$	-0,6510	0,0404	$w_{sp}^2 \times t_{shift}^2$	-0,1914	0,3803
$w_{sp} \times t_{shift}$	-0,1890	0,1193	w_{ti}^2	0,3518	0,0000
$w_{ti} \times w_{sp}^2$	-0,9090	0,0002	w_{sp}^2	0,0388	0,0344
$w_{ti} \times t_{shit}^2$	0,4570	4,9415	t_{shift}^2	2,0204	0,9911
$w_{sp} \times t_{shift}^2$	0,1680	1,2655			

	DF	SS	F
Kaynak			
Regresyon	18	30,5930	84,99
Geri Kalan Hata	197	1,6996	
Toplam	215	34,5325	

4.3 – Yapay Sinir Ağları Yaklaşımı

Yapay sinir ağı, eğitim için Excite veri örneğinin ilk 5014 sorgusu üzerinde çalıştırılmıştır. Bu sayı, herhangi bir oturum bölünmeksizin Excite veri örneğinden alınan örneğin yaklaşık yarısını oluşturmaktadır. Her bir kullanıcı oturumu sonundaki sorgu, zaman aralığı veya arama yapısının belirlenmesinde kullanılamayacağından, bu 5014 sorgunun tamamının kullanılmadığına dikkat edilmelidir. Kullanıcı oturumundaki son sorgudan sonra gelen ardışık bir sorgu daha olmadığından (aynı oturuma ait), bu sorgu için zaman aralığı veya arama yapısı belirlenemez. Veri örneğinin ilk kısmında 1201 kullanıcı oturumu bulunmaktadır. Veri örneğinin ikinci kısmındaki sorgu sayısı ise (test kısmı) 5014'ten 3813'e inmiştir. İnsanlar tarafından konu değişiklikleri belirlendikten sonra, 3813 sorgu içerisinde 2544 "konu değişikliği yok" ve 269 "konu değişikliği var" durumu olduğu belirlenmiştir. Yapay sinir ağının eğitimi %5 hata (ortalama karesel hata) düzeyine veya belirli bir iterasyon sayısına (10.000 iterasyon) ulaşılıncaya kadar sürdürülmüştür. Bununla birlikte, çoğu kez yapay sinir ağının hata düzeyi %5'in altına indiğinden, iterasyon sayısı üst sınıra ulaşmamıştır. Veri örneğinin ilk kısmı üzerinde eğitilen yapay sinir ağı, ikinci kısım üzerinde test edilmiştir. Uygulanan yaklaşımın testi için kullanılan performans ölçüleri, P, R ve F_β olarak belirlenmiştir.

Veri örneğinin test kısmında, konu değişikliklerini belirlemesi için yapay sinir ağı çalıştırılmış ve P, R ve F_β değerleriyle, A tipi hata ve B tipi hata sayısı hesaplanmıştır.

Yapay sinir ağının verdiği sonuçların ikili değişkenlere (1 = konu değişikliği yok veya 2 = konu değişikliği var) dönüştürülmesi için bir eşik kullanılmıştır. Normal şartlar altında, bu eşik değerinin 1,5 olması gerektiği düşünülebilir. Öte yandan, He ve arkadaşlarının (2002) çalışmasında B tipi hataların A tipi hatalara göre daha önemli olduğu belirtilmektedir. Bu amaçla, kullanılan performans ölçüsünün hesaplanmasında B tipi hataların az olması, A tipi hataların az olmasına göre daha önceliklidir. Dolayısıyla, A tipi hataların bir miktar artmasına neden olsa da B tipi hataların azaltılması daha önemli olabilmektedir. Bu nedenle, kullanılan eşik değerinin 1,5'ten daha az olmasının daha iyi bir sonuç verme olasılığı olduğu düşünülerek, farklı eşik değerleri denenerek simülasyonlar yapılmış ve en iyi sonucu hangi eşik değerinin

verdiği belirlenmeye çalışılmıştır. Eşik değerleri 1,1, 1,2, 1,3, 1,4 ve 1,5 olarak belirlenmiş ve yapay sinir ağı, her eşik değeri için 300 kez eğitilmiş ve test edilmiştir. Testler sonucunda her eşik değeri için elde edilen en iyi sonuçlar belirlenmiştir. Bu sonuçlar Çizelge 4.9'da görülmektedir.

Sonuçlardan görüldüğü gibi en iyi amaç fonksiyonu (F_β) değeri olan 0,5088, eşik değerinin 1,2 ve 1,3 olduğu durumlarda elde edilmiştir. İkinci en iyi değer olan 0,5008, eşik değerinin 1,4 ve 1,5 olduğu durumlarda elde edilmiş olup, 1,1 eşik değerinde en kötü değer olan 0,4741 elde edilmiştir.

Çizelge 4.9: Farklı Eşik Değerleri için En İyi Sonuçlar

Eşik	A Tipi Hata	B Tipi Hata	P	R	F_β
1,1	343	35	0,2543	0,7697	0,4741
1,2	283	36	0,2907	0,7632	0,5088
1,3	283	36	0,2907	0,7632	0,5088
1,4	218	50	0,3187	0,6711	0,5008
1,5	218	50	0,3187	0,6711	0,5008

Önceki yaklaşımla elde edilen sonuçlarla karşılaştırıldığında, amaç fonksiyonu değerinde çok küçük bir iyileşmenin olduğu görülmektedir (0,5088 – 0,5080 = 0,0008). Bu küçük fark, dikkate alınmazsa, uygulanan yaklaşımların konu değişikliklerini belirlemede gösterdikleri başarıların eşit olduğu söylenebilir.

4.3.1 – Eğitim ve Test Verilerinin Farklı Veri Kümelerinden Seçilmesi

Önceki bölümde uygulanan yaklaşımda, tasarlanan yapay sinir ağı belirli bir veri örneğinin (Excite veri örneği) yaklaşık olarak yarısı kullanılarak eğitildikten sonra, yine aynı veri örneğinin ikinci yarısı üzerinde test edilerek performansı ölçülmüştür. Burada, şöyle bir soru sorulabilir: Bu şekilde eğitilmiş olan yapay sinir ağı, farklı bir veri örneğinden seçilmiş olan veriler üzerinde de aynı başarıyı gösterebilecek midir? Yoksa böyle bir yaklaşımın uygulanabilmesi için yapay sinir ağının, her seferinde test edileceği verilerin bulunduğu bir veri kümesinden seçilen bir örnek ile eğitilmesi mi gerekmektedir?

Bu sorulara cevap bulabilmek amacıyla, tasarlanan yapay sinir ağı için eğitim ve test verileri farklı veri kümelerinden seçilmiştir. Bunun için Excite ve Fast arama motorlarının veri kümelerinden alınan örnekler kullanılmıştır. Yapay sinir ağı önce Excite arama motoru veri örneğinin ilk yarısı kullanılarak eğitilmiş, ikinci yarısı kullanılarak test edilmiş (bundan önce uygulanan yaklaşımdaki ile aynı) ve ayrıca, Fast arama motoru veri örneğinin ikinci yarısı üzerinde de test edilmiştir. Daha sonra bunun tersi yapılarak, yapay sinir ağı önce Fast arama motoru veri örneğinin ilk yarısı kullanılarak eğitilmiş, ikinci yarısı kullanılarak test edilmiş ve ayrıca, Excite arama motoru veri örneğinin ikinci yarısı üzerinde de test edilmiştir. Örneklerin test kısmında, konu değişikliklerini belirlemesi için yapay sinir ağı çalıştırılmış ve P, R ve F_β değerleriyle, A tipi hata ve B tipi hata sayısı hesaplanmıştır.

Durum 1: Eğitim: Excite – Test: Excite

Yapay sinir ağı, Excite veri kümesinden alınan örneğin ilk yarısı kullanılarak eğitilmiş, Excite veri kümesinden alınan örneğin ikinci yarısı kullanılarak test edilmiştir. Bu durumda, yapay sinir ağının eğitim ve test verileri aynı veri kümesinden alınmaktadır. Yapay sinir ağı, 50 kez eğitilmiş ve bu eğitimler için test edilmiştir. Her test sonucunda, performans ölçüleri, P, R ve F_β değerleri hesaplanmıştır. Bu testler sonucunda, P, 0,2427 ile 0,3182; R, 0,5526 ile 0,7697 ve F_β, 0,4093 ile 0,4850 arasında değişen değerler almışlardır.

Durum 2: Eğitim: Excite – Test: Fast

Yapay sinir ağı, Excite veri kümesinden alınan örneğin ilk yarısı kullanılarak eğitilmiş, Fast veri kümesinden alınan örneğin ikinci yarısı kullanılarak test edilmiştir. Bu durumda, yapay sinir ağının eğitim ve test verileri farklı veri kümelerinden alınmaktadır. Yapay sinir ağı, 50 kez eğitilmiş ve bu eğitimler için test edilmiştir. Her test sonucunda, performans ölçüleri, P, R ve F_β değerleri hesaplanmıştır. Bu testler sonucunda, P, 0,3892 ile 0,4790; R, 0,5161 ile 0,7452 ve F_β, 0,4799 ile 0,5930 arasında değişen değerler almışlardır.

Durum 3: Eğitim: Fast – Test: Excite

Yapay sinir ağı, Fast veri kümesinden alınan örneğin ilk yarısı kullanılarak eğitilmiş, Excite veri kümesinden alınan örneğin ikinci yarısı kullanılarak test edilmiştir. Bu durumda, yapay sinir ağının eğitim ve test verileri farklı veri kümelerinden alınmaktadır. Yapay sinir ağı, 50 kez eğitilmiş ve bu eğitimler için test edilmiştir. Her test sonucunda, performans ölçüleri, P, R ve F_β değerleri hesaplanmıştır. Durum 1 ve 3 için performans ölçülerinin değerleri karşılaştırılmıştır. Bu testler sonucunda, P, 0,2389 ile 0,3187; R, 0,5526 ile 0,7632 ve F_β, 0,3981 ile 0,5088 arasında değişen değerler almışlardır.

Durum 4: Eğitim: Fast – Test: Fast

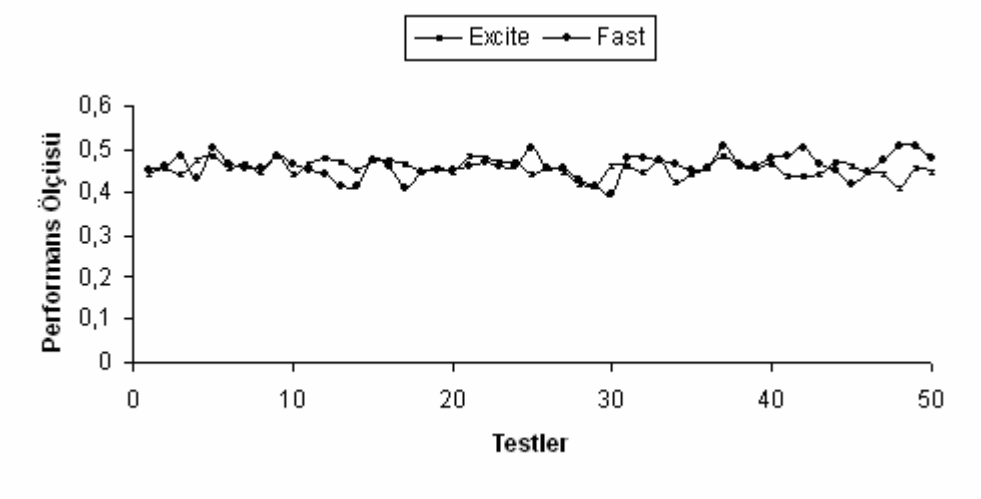
Yapay sinir ağı, Fast veri kümesinden alınan örneğin ilk yarısı kullanılarak eğitilmiş, Fast veri kümesinden alınan örneğin ikinci yarısı kullanılarak test edilmiştir. Bu durumda, yapay sinir ağının eğitim ve test verileri aynı veri kümesinden alınmaktadır. Yapay sinir ağı, 50 kez eğitilmiş ve bu eğitimler için test edilmiştir. Her test sonucunda, performans ölçüleri, P, R ve F_β değerleri hesaplanmıştır. Durum 2 ve 4 için performans ölçülerinin değerleri karşılaştırılmıştır. Bu testler sonucunda, P, 0,3902 ile 0,4737; R, 0,5161 ile 0,7290 ve F_β, 0,4955 ile 0,5974 arasında değişen değerler almışlardır.

Sonuçlar iki farklı açıdan yorumlanabilir: (i) farklı eğitim örnekleri kullanılarak eğitilen yapay sinir ağının, belirli bir test örneği üzerindeki performansı ve (ii) belirli bir eğitim örneği kullanılarak eğitilen yapay sinir ağının, farklı test örnekleri üzerindeki performansı.

Sonuçlar; farklı eğitim örnekleri kullanılarak eğitilen yapay sinir ağının, belirli bir test örneği üzerindeki performansının incelenmesi açısından, aşağıdaki şekilde yorumlanabilir:

- Excite ve Fast eğitim örnekleriyle eğitilen yapay sinir ağının, Excite test örneğiyle test edilmesiyle elde edilen F_β değerlerinin grafiği Şekil 4.5'te

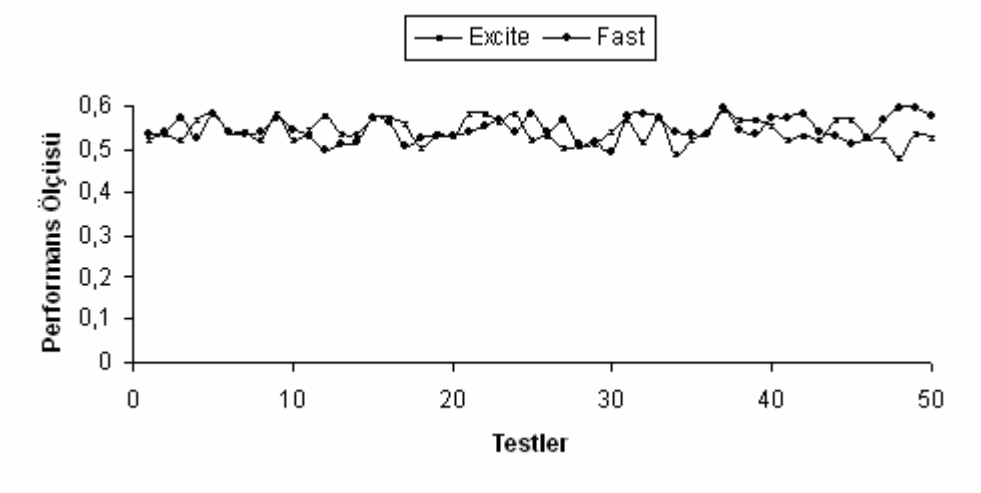
görülmektedir. Şekilden görüldüğü gibi yapay sinir ağının, Excite eğitim örneğiyle eğitilerek, Excite test örneği ile test edilmesi sonucunda elde edilen F_β değerleri ile Fast eğitim örneğiyle eğitilerek, Excite test örneği ile test edilmesi sonucunda elde edilen F_β değerleri arasında bazı küçük sapmalar olmakla birlikte, bu değerlerin genel olarak birbirlerine yakın oldukları görülmektedir. Yapay sinir ağının, Excite eğitim örneğiyle eğitilerek, Excite test örneği ile test edilmesi sonucunda elde edilen F_β değerlerinin aritmetik ortalaması 0,456458; Fast eğitim örneğiyle eğitilerek, Excite test örneği ile test edilmesi sonucunda elde edilen F_β değerlerinin aritmetik ortalaması 0,461660 olmaktadır. Öte yandan, ağın, Fast eğitim örneğiyle eğitimi sonucunda, performansının küçük bir farkla da olsa, daha iyi olması dikkati çekmektedir.



Şekil 4.5: Farklı Eğitim ve Test Örnekleri (Test: Excite - Performans Ölçüsü: F_β)

- Excite ve Fast eğitim örnekleriyle eğitilen yapay sinir ağının, Fast test örneğiyle test edilmesiyle elde edilen F_β değerlerinin grafiği Şekil 4.6'da görülmektedir. Şekilden görüldüğü gibi yapay sinir ağının, Excite eğitim örneğiyle eğitilerek, Fast test örneği ile test edilmesi sonucunda elde edilen F_β değerleri ile Fast eğitim örneğiyle eğitilerek, Fast test örneği ile test edilmesi sonucunda elde edilen F_β değerleri arasında bazı küçük sapmalar

olmakla birlikte, bu değerlerin genel olarak birbirlerine yakın oldukları görülmektedir. Yapay sinir ağının, Excite eğitim örneğiyle eğitilerek, Fast test örneği ile test edilmesi sonucunda elde edilen F_β değerlerinin aritmetik ortalaması 0,542186; Fast eğitim örneğiyle eğitilerek, Fast test örneği ile test edilmesi sonucunda elde edilen F_β değerlerinin aritmetik ortalaması 0,548050 olmaktadır. Ağın, Fast eğitim örneğiyle eğitimi sonucunda, performansının küçük bir farkla da olsa, daha iyi olduğu görülmektedir.

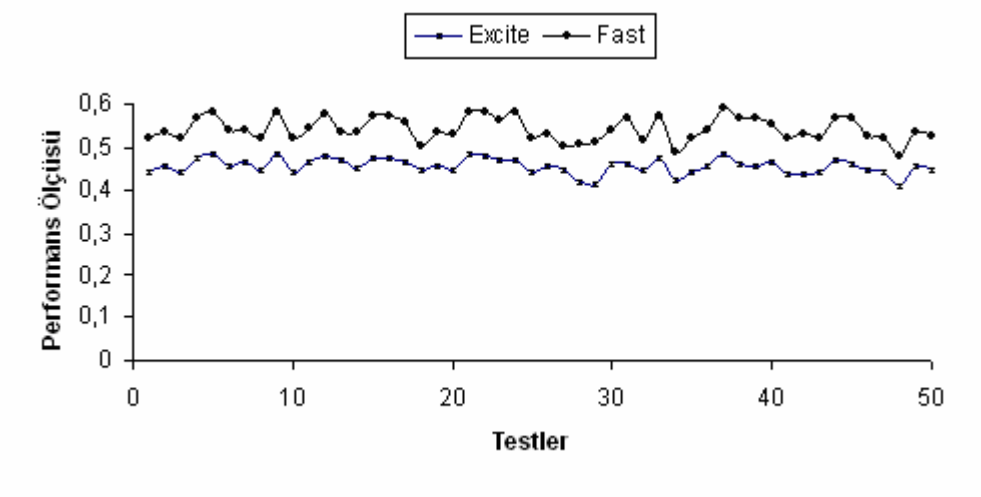


Şekil 4.6: Farklı Eğitim ve Test Örnekleri (Test: Fast - Performans Ölçüsü: F_β)

Sonuçlar; belirli bir eğitim örneği kullanılarak eğitilen yapay sinir ağının, farklı test örnekleri üzerindeki performansının incelenmesi açısından aşağıdaki şekilde yorumlanabilir:

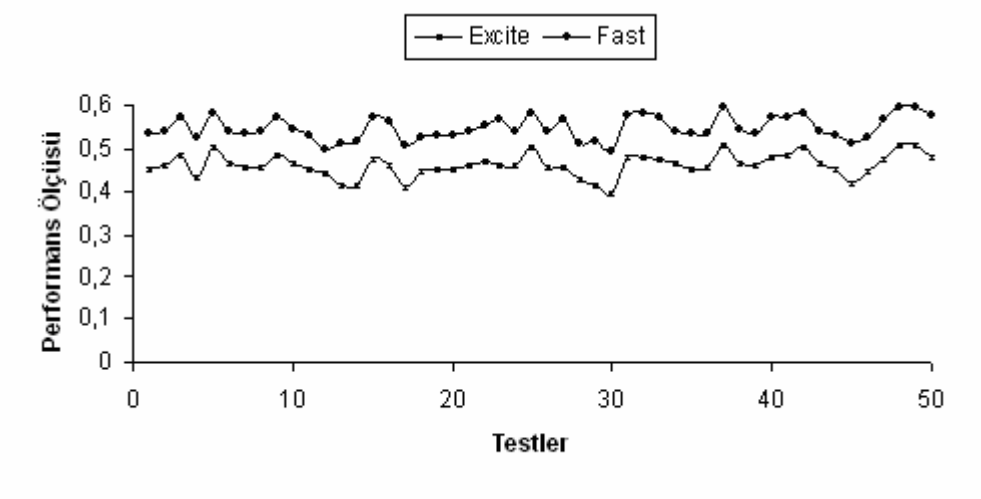
- Excite eğitim örneğiyle eğitilen yapay sinir ağının, Excite ve Fast test örnekleriyle test edilmesiyle elde edilen F_β değerlerinin grafiği Şekil 4.7'de görülmektedir. Şekilden görüldüğü gibi yapay sinir ağının, Excite eğitim örneğiyle eğitilerek, Excite test örneği ile test edilmesi sonucunda elde edilen F_β değerleri; Fast test örneği ile test edilmesi sonucunda elde edilen F_β değerlerinden daha düşüktür, ancak normal şartlar altında da Excite test örneğinden elde edilen değerlerin daha düşük olduğu

bilinmektedir. Öte yandan, iki örneğe ait değerlerin, genel olarak, benzer bir eğilim gösteriyor olması önemlidir.



Şekil 4.7: Farklı Eğitim ve Test Örnekleri (Eğitim: Excite - Performans Ölçüsü: F_β)

- Fast eğitim örneğiyle eğitilen yapay sinir ağının, Excite ve Fast test örnekleriyle test edilmesiyle elde edilen F_β değerlerinin grafiği Şekil 4.8'de görülmektedir. Şekilden görüldüğü gibi yapay sinir ağının, Fast eğitim örneğiyle eğitilerek, Excite test örneği ile test edilmesi sonucunda elde edilen F_β değerleri; Fast test örneği ile test edilmesi sonucunda elde edilen F_β değerlerinden daha düşüktür, ancak normal şartlar altında da Excite test örneğinden elde edilen değerlerin daha düşük olduğu bilinmektedir. Öte yandan, iki örneğe ait değerlerin, genel olarak, benzer bir eğilim gösteriyor olması önemlidir.



Şekil 4.8: Farklı Eğitim ve Test Örnekleri (Eğitim: Fast - Performans Ölçüsü: F_β)

5 – TARTIŞMA

Bu tez çalışmasında, İnternet arama motorları kullanıcı oturumlarındaki konu değişikliklerini otomatik olarak tespit etmek için iki farklı yaklaşım uygulanmıştır: (i) genetik algoritmalar ve Dempster-Shafer teorisi yaklaşımı ve (ii) yapay sinir ağları yaklaşımı.

Yaklaşımların test edilebilmesi için Excite ve Fast arama motorlarının veri kümelerinden örnekler alınmıştır. Alınan örneklerin yaklaşık yarısı eğitim veya diğer bir deyişle, yaklaşımında kullanılan bazı değerlerin (olasılık ve parametre değerleri gibi) belirlenmesi için kullanılırken, örneğin geri kalan kısmı da test için kullanılmıştır.

Birinci yaklaşımda, He ve arkadaşlarının (2002) çalışmasına benzer bir çalışma yürütülmüştür. Yazarların Reuters arama motorun veri kümesinin örneğini kullandıkları çalışmalarındaki yaklaşım, Excite arama motoru verilerine uygulanmıştır. Buna göre, arama motoru kayıtlarında yer alan zaman aralığı ve arama yapısı verileri kullanılarak her kayıt için bu verilere bağlı olarak iki farklı "konu değişikliği var" olasılığı hesaplanmıştır. Bu iki veriden elde edilen olasılıkların birleştirilmesi için Dempster-Shafer teorisi kullanılarak, bu iki olasılık birleştirilmiş ve tek bir olasılık elde edilmiştir. Elde edilen bu olasılık bir eşik değeriyle kıyaslanarak "konu değişikliği yok" ve "konu değişikliği var" şeklinde ikili değişkenlere dönüştürülmüştür. Eğitim aşamasında, yaklaşımın başarısını ölçmek için kullanılan performans ölçülerinin en iyi değerlerini verecek olan parametre değerlerinin belirlenmesi gerekmektedir. Performans ölçülerinin hesaplanma şeklinden dolayı bu işlem analitik yöntemlerle yapılamadığından, bir genetik algoritma kullanılarak, en iyi parametre değerleri belirlenmiştir. Bu parametre değerleri belirlendikten sonra, yaklaşım örneğin test kısmında çalıştırılmıştır. Bu yaklaşımda kullanılan bazı parametre ve olasılık değerlerinin etkilerini incelemek için çeşitli istatistiksel analizler de uygulanmıştır.

Yaklaşım oldukça başarılı bulunmuştur. Öte yandan, genetik algoritma kullanılarak belirlenen bazı parametre değerleri için rasgele değerler kullanılarak da belirli bir düzeyde başarı sağlanmış olması ilginçtir. Böyle bir durumda, bu parametre değerlerinin belirlenmesi için kullanılan matematiksel yöntemlerin çok fazla önemi olmadığı düşünülebilir. Diğer bir ilginç sonuç da bu çalışmada Excite verileri

kullanılmış olmasına rağmen, yaklaşımda He ve arkadaşlarının (2002) çalışmasında verilen Reuters olasılıkları kullanıldığında da başarılı olmasıdır.

Uygulanan ikinci yaklaşım olan yapay sinir ağları yaklaşımında ise konu değişikliklerinin belirlenmesi için bir yapay sinir ağı kullanılmıştır. İlk olarak kullanılan yapay sinir ağı iki girişli olup, bu girişler zaman aralığı ve arama yapısı şeklindedir. Bu girişleri alan yapay sinir ağına, bu girişlere karşılık gelen "konu değişikliği yok" ve "konu değişikliği var" çıkışları verilerek ağ eğitilmiştir. Eğitim için örneğin ilk yarısı kullanılmıştır. Yapay sinir ağı eğitildikten sonra örneğin ikinci kısmı kullanılarak test edilmiş ve ilk yaklaşımda kullanılan performans ölçülerine göre yaklaşımın başarısı incelenmiştir. Burada ayrıca, yapay sinir ağının eğitimi ve testi için farklı örnekler kullanılmış ve yapay sinir ağı bu şekilde çalıştırılmıştır. Yani Excite örneği ile eğitilen yapay sinir ağı sadece Excite örneğiyle test edilmeyerek, aynı zamanda, Fast örneği ile de test edilmiştir. Tersi de yapılarak, Fast örneği ile eğitilen yapay sinir ağının testi hem Fast hem de Excite örneği ile gerçekleştirilmiştir. Yapay sinir ağının, eğitim ve test verilerinin aynı örnekten olduğu durumda göstermiş olduğu başarıyı, eğitim ve test verilerinin farklı örnekten olduğu durumda da gösterip gösteremediği incelenmiştir.

Bu yaklaşım da oldukça başarılı olmuştur. Bu yaklaşımla, elde edilen sonuçlar, ilk yaklaşımla elde edilen sonuçlara yakın değerler olarak bulunmuşlardır. Öte yandan, herhangi bir veri örneğinin verileriyle eğitilen yapay sinir ağı, başka bir veri örneğiyle test edildiğinde de başarılı olmaktadır. Bu durum, yapay sinir ağının genel bir kullanıma sahip olabileceğini göstermektedir. Buradaki çalışma için Excite örneğiyle eğitilen yapay sinir ağı, Fast örneğiyle test edildiğinde, eğitimin Fast örneğiyle gerçekleştirildiği duruma yakın bir başarı göstermiştir. Aynı zamanda, Fast örneğiyle eğitilen yapay sinir ağı, Excite örneğiyle test edildiğinde de eğitimin Excite örneğiyle gerçekleştirildiği duruma yakın bir başarı elde edilmiştir.

Sonuç olarak, burada uygulanan yaklaşımların, arama motorları kullanıcı oturumlarındaki konu değişikliklerini belirlemede başarılı oldukları söylenebilir. Yaklaşımların başarıları yaklaşık olarak eşit olmakla birlikte, ilk yaklaşımda elde edilen sonuçların, daha kolay bir şekilde elde edilebiliyor olmaları ilginçtir. Bu durumda, yaklaşım için gösterilen çabanın gereksiz olabileceği şeklinde bir düşünce oluşabilir. İkinci yaklaşım, bu açıdan daha tutarlı görünmektedir. Eğitim ve testi için farklı veriler

kullanılması durumunda da başarılı olduğu için genel bir kullanıma sahip olabileceği söylenebilir. Kullanılan yöntemlerde yapılacak bazı geliştirmelerin, yaklaşımların performansları üzerinde olumlu etkileri olabilir. Ayrıca, bu yaklaşımların daha büyük ve farklı örnekler üzerinde uygulanması yeni ve ilginç sonuçlar elde edilmesini sağlayabilir.

KAYNAKLAR

Allen, D. W., Johnson, S. 1998. İnternet Öğrenim Kılavuzu. Alfa Çeviri Grubu, İstanbul. 365 s.

Bauer, M. 1997. Approximation algorithms and decision making in the Dempster-Shafer theory of evidence- An emprical study. International Journal of Approximate Reasoning, 17 (1997), 217-237.

Beynon, M., Curry, B., Morgan, P. 2000. The Dempster-Shafer theory of evidence: an alternative approach to multicriteria decision modelling. The International Journal of Management Science, 28 (2000), 37-50.

Bilal, D., Kirby, J. 2002. Differences and similarities in information seeking: children and adults as Web users. Information Processing and Management, 38 (2002), 649-670.

Can, F., Nuray, R., Sevdik, A. B. 2003. Automatic performance analysis of Web search engines. Information Processing and Management, 40 (2004), 495-514.

Caramia, M., Felici, G., Pezzoli, A. 2003. Improving search results with data mining in a thematic search engine. Computers & Opearations Research, (baskıda).

Chuang, S.-L., Chien, L.-F. 2002. Enriching Web taxonomies through subject categorization of query terms from search engine logs. Decision Support Systems, 35 (2003), 113-127.

Comber A. J., Law A.N.R., Lishman J.R. 2004. A comparison of Bayes', Dempster-Shafer and Endorsement theories for managing knowledge uncertainty in the context of land cover monitoring. Computers, Environment and Urban Systems, 28 (2004), 311-327.

Dengiz, B., Altıparmak, F. 1998. Genetik Algoritmalara Genel Bir Giriş. Endüstri Mühendisliği Dergisi, 9 (3), 3-14.

Durkin, J., 1994. Expert Systems: Design and Development, Prentice Hall, New Jersey. 800 p.

Efe, Ö., Kaynak, O. 2000. Yapay Sinir Ağları ve Uygulamaları. Boğaziçi Üniversitesi Yayını, İstanbul. 141 s.

Gen, M., Cheng R., 1999. Genetic Algorithms and Engineering Optimization, John Wiley & Sons, USA. 495 p.

Goldberg, D., E., 1989. Genetic Algoritms in Search, Optimization, and Machine Learning, Addision-Wesley, USA. 412 p.

Haykin, S. 1994. Neural Networks, Macmillan College Publishing Company, USA. 696 p.

He, D., Göker, A., Harper, D. J. 2002. Combining evidence for automatic Web session identification. Information Processing and Management, 38 (2002), 727-742.

Jansen, J. B., Spink, A. 2003. An analysis of Web searching by European AlltheWeb.com users. Information Processing and Management, (baskıda).

Liaw, S. - S., Huang, H. - M. 2003. An investigation of user attitudes towards search engines as an information retrieval tool. Computers in Human Behavior, 19 (2003), 751-765.

Ma, L., Khorasani, K. 2003. A new strategy for adaptively constructing multilayer feedforward neural networks. Neurocomputing, 51 (2003), 361-385.

Montgomery, D. C., 1991. Design and Analysis of Experiments, John Wiley & Sons, USA. 649 p.

Nilsson, N. J. 1998. Artificial Intelligence: A New Synthesis, Morgan Kaufmann Publishers, Inc. San Francisco, California. 513 p.

Oliver, I. 1993. Programming Classics: Implementing the World's Best Algorithms. New Jersey. 386 p.

Ozmutlu, H. C., Cavdur, F. 2005. Application of automatic topic identification on Excite Web search engine data logs. Information Processing and Management, 41 (2005), 1243-1262.

Ozmutlu, H. C., Spink, A., Ozmutlu, S. 2002. Analysis of large data logs: an application of Poisson sampling on excite web queries. Information Processing and Management, 38 (2002), 473-490.

Ozmutlu, S., Cavdur, F. 2005. Neural network applications for automatic new topic identification. Online Information Review, 29 (2005), 34-53.

Ozmutlu, S., Harmonosky, C. M. (2005). A real time methodology for minimizing mean flowtime in FMSs with routing flexibility: threshold based Alternate Routing. European Journal of Operational Research, 166 (2005), 369-384.

Ozmutlu, S., Spink, A., Ozmutlu, H. C. 2003. A day in the life of Web searching: an exploratory study. Information Processing and Management, 40 (2004), 319-345.

Spink, A. 2002. A user-centered approach to evaluating human interaction with Web search engines: an exploratory study. Information Processing and Management, 38 (2002), 401-426.

Spink, A., Ozmutlu, H. C., Lorence, P. D. 2002. Web searching for sexual information: an exploratory study. Information Processing and Management, 40 (2004), 113-123.

Spink, A., Ozmutlu, H. C. 2002. Characteristics of question format Web queries: an exploratory study. Information Processing and Management, 38 (2002), 453-471.

Vaughan, L. 2003. New measuraments for search engine evaluation proposed and tested. Information Processing and Management, 40 (2004), 677-691.

http://www.google.com.tr

http://search.msn.com

http://search.yahoo.com

TEŞEKKÜR

Bu çalışma ve yüksek lisans eğitimim süresince bana yardımcı olan bölüm başkanım sayın Prof. Dr. Erdal Emel'e, yüksek lisans tezini birlikte yaptığım danışmanım sayın Yrd. Doç. Dr. H. Cenk Özmutlu'ya ve eşi sayın Yrd. Doç. Dr. Seda Özmutlu'ya, sayın Öğr. Gör. Dr. A. Yurdun Orbak'a teşekkürü bir borç bilirim. Ayrıca eğitimim boyunca bana her zaman destek olan aileme teşekkür ederim.

ÖZGEÇMİŞ

Fatih Çavdur 04.03.1978 tarihinde Bursa Orhaneli'de doğmuştur. İlk ve orta öğrenimini Harmancık İlköğretim Okulu'nda, lise öğrenimini Ali Osman Sönmez Anadolu Teknik Lisesi'nde tamamlamıştır. 2002 yılında Uludağ Üniversitesi Endüstri Mühendisliği Bölümün'den mezun olan Fatih Çavdur, aynı yıl Uludağ Üniversitesi Fen Bilimleri Enstitüsü Endüstri Mühendisliği Anabilim Dalı'nda yüksek lisans eğitimine başlamıştır. Halen Uludağ Üniversitesi, Mühendislik-Mimarlık Fakültesi, Endüstri Mühendisliği Bölümü'nde araştırma görevlisi olarak çalışmaktadır.

EKLER

Ek 1: Arama Yapısı Belirleme Algoritması Pascal Kodu

```
program search_pattern;

uses crt;

const
     browsing_code              =1;
     generalisation_code        =2;
     specialisation_code        =3;
     reformulation_code         =4;
     new_code                   =5;
     relevance_feedback_code    =6;
     others_code                =7;
     IP_change_code             =8;

var
   code:integer;
   key:char;
   i,j,k:integer;
   counter,control:byte;

   filevar_query:text;
   filevar_scv:text;
   filevar_sp:text;

   query1,query2:string[50];
   scv1,scv2:byte;

   character1,character2:array[1..50] of string[1];
   word1,word2:array[1..20] of string[20];
   now1,now2,now1_2,now1_new,now2_new:byte;
   rf_control:byte;

   intra_browsing,intra_generalisation,intra_specialisation,intra_reformulation,
   intra_repetition,intra_new,intra_relevance_feedback,intra_others:integer;

   inter_browsing,inter_generalisation,inter_specialisation,inter_reformulation,
   inter_repetition,inter_new,inter_relevance_feedback,inter_others:integer;

   intra_total,inter_total:integer;

   p_sp_browsing,p_contin_sp_browsing,p_shift_sp_browsing:real;
   p_sp_generalisation,p_contin_sp_generalisation,p_shift_sp_generalisation:real;
   p_sp_specialisation,p_contin_sp_specialisation,p_shift_sp_specialisation:real;
   p_sp_reformulation,p_contin_sp_reformulation,p_shift_sp_reformulation:real;
   p_sp_new,p_contin_sp_new,p_shift_sp_new:real;
   p_sp_relevance_feedback,p_contin_sp_relevance_feedback,p_shift_sp_relevance_feedback:real;
   p_sp_others,p_contin_sp_others,p_shift_sp_others:real;

   B,C,D,B_new,C_new,D_new:array[1..20] of string[20];

begin
     clrscr;
     assign(filevar_query,'program\veriler\fast\q_tum.txt');
     {$i-} reset(filevar_query); code:=ioresult; {$i+}
     if code<>0 then
        begin
             writeln('File not found!');
             key:=readkey;
             halt;
        end;
```

```
assign(filevar_scv,'program\veriler\fast\c_tum.txt');
{$i-} reset(filevar_scv); code:=ioresult; {$i+}
if code<>0 then
   begin
        writeln('File not found!');
        key:=readkey;
        halt;
   end;
assign(filevar_sp,'program\veriler\fast\s_tum.txt');
rewrite(filevar_sp);
readln(filevar_query,query1);
readln(filevar_scv,scv1);
while not eof(filevar_query) do
      begin
           rf_control:=0;
           readln(filevar_query,query2);
           readln(filevar_scv,scv2);
           if scv1=1 then
              begin
                   if query2=' ' then
                      begin
                           inc(intra_relevance_feedback);
                           writeln(filevar_sp,relevance_feedback_code);
                           now1:=0;
                           now2:=0;
                           rf_control:=1;
                      end;
              end
           else if scv1=2 then
                begin
                     if query2=' ' then
                        begin
                             inc(inter_relevance_feedback);
                             writeln(filevar_sp,relevance_feedback_code);
                             now1:=0;
                             now2:=0;
                             rf_control:=1;
                        end;
                end
           else writeln(filevar_sp,IP_change_code);
           if rf_control=0 then
              begin
                   now1:=1;
                   for i:=1 to length(query1) do
                       begin
                            character1[i]:=copy(query1,i,1);
                            if character1[i]=' ' then inc(now1)
                            else word1[now1]:=concat(word1[now1],character1[i]);
                       end;
                   for i:=1 to now1 do
                       begin
                            C[i]:=word1[i];
                       end;
                   for i:=1 to now1 do delete(word1[i],1,length(word1[i]));
                   now2:=1;
                   for i:=1 to length(query2) do
                       begin
                            character2[i]:=copy(query2,i,1);
                            if character2[i]=' ' then inc(now2)
                            else word2[now2]:=concat(word2[now2],character2[i]);
                       end;
                   for i:=1 to now2 do
                       begin
                            D[i]:=word2[i];
                       end;
                   for i:=1 to now2 do delete(word2[i],1,length(word2[i]));
                   now1_2:=0;
                   for i:=1 to now1 do
```

```
                    begin
                         for j:=1 to now2 do
                             begin
                                  if C[i]=D[j] then
                                     begin
                                          inc(now1_2);
                                          B[now1_2]:=C[i];
                                          delete(C[i],1,length(C[i]));
                                          delete(D[j],1,length(D[j]));
                                     end;
                             end;
                    end;

               counter:=0;
               control:=0;
               j:=0;
               for i:=1 to now1 do
                    begin
                         if length(C[i])>0 then
                            begin
                                 inc(j);
                                 C_new[j]:=C[i];
                            end;
                         now1_new:=j;
                    end;

               counter:=0;
               control:=0;
               j:=0;
               for i:=1 to now2 do
                    begin
                         if length(D[i])>0 then
                            begin
                                 inc(j);
                                 D_new[j]:=D[i];
                            end;
                         now2_new:=j;
                    end;
               if scv1=1 then
                  begin
                       if (now1_2>0) and (now1_new=0) and (now2_new=0) then
                          begin
                               inc(intra_browsing);
                               writeln(filevar_sp,browsing_code);
                          end
                       else if (now1_2>0) and (now1_new>0) and (now2_new=0) then
                            begin
                                 inc(intra_generalisation);
                                 writeln(filevar_sp,generalisation_code);
                            end
                       else if (now1_2>0) and (now1_new=0) and (now2_new>0) then
                            begin
                                 inc(intra_specialisation);
                                 writeln(filevar_sp,specialisation_code);
                            end
                       else if (now1_2>0) and (now1_new>0) and (now2_new>0) then
                            begin
                                 inc(intra_reformulation);
                                 writeln(filevar_sp,reformulation_code);
                            end
                       else if now1_2=0 then
                            begin
                                 inc(intra_new);
                                 writeln(filevar_sp,new_code);
                            end
                       else
                           begin
                                inc(intra_others);
                                writeln(filevar_sp,others_code);
                           end
                  end
```

```
                    else if scv1=2 then
                         begin
                              if (now1_2>0) and (now1_new=0) and (now2_new=0) then
                                 begin
                                      inc(inter_browsing);
                                      writeln(filevar_sp,browsing_code);
                                 end
                              else if (now1_2>0) and (now1_new>0) and (now2_new=0) then
                                   begin
                                        inc(inter_generalisation);
                                        writeln(filevar_sp,generalisation_code);
                                   end
                              else if (now1_2>0) and (now1_new=0) and (now2_new>0) then
                                   begin
                                        inc(inter_specialisation);
                                        writeln(filevar_sp,specialisation_code);
                                   end
                              else if (now1_2>0) and (now1_new>0) and (now2_new>0) then
                                   begin
                                        inc(inter_reformulation);
                                        writeln(filevar_sp,reformulation_code);
                                   end
                              else if now1_2=0 then
                                   begin
                                        inc(inter_new);
                                        writeln(filevar_sp,new_code);
                                   end
                              else
                                  begin
                                       inc(inter_others);
                                       writeln(filevar_sp,others_code);
                                  end;
                         end;
               end;
            query1:=query2;
            scv1:=scv2;
      end;
close(filevar_query);
close(filevar_scv);
close(filevar_sp);
intra_total:=intra_browsing+intra_generalisation+intra_specialisation+intra_reformulation+
             intra_repetition+intra_new+intra_relevance_feedback+intra_others;
inter_total:=inter_browsing+inter_generalisation+inter_specialisation+inter_reformulation+
             inter_repetition+inter_new+inter_relevance_feedback+inter_others;
p_sp_browsing:=(intra_browsing+inter_browsing)/(intra_total+inter_total);
if(intra_browsing+inter_browsing>0) then
   begin
        p_contin_sp_browsing:=intra_browsing/(intra_browsing+inter_browsing);
        p_shift_sp_browsing:=1-p_contin_sp_browsing;
   end
else
    begin
         p_contin_sp_browsing:=9.9999;
         p_shift_sp_browsing:=9.9999;
    end;

p_sp_generalisation:=(intra_generalisation+inter_generalisation)/(intra_total+inter_total);
if(intra_generalisation+inter_generalisation>0) then
   begin
        p_contin_sp_generalisation:=intra_generalisation/(intra_generalisation+inter_generalisation);
        p_shift_sp_generalisation:=1-p_contin_sp_generalisation;
   end
else
    begin
         p_contin_sp_generalisation:=9.9999;
         p_shift_sp_generalisation:=9.9999;
    end;

p_sp_specialisation:=(intra_specialisation+inter_specialisation)/(intra_total+inter_total);
if(intra_specialisation+inter_specialisation>0) then
   begin
        p_contin_sp_specialisation:=intra_specialisation/(intra_specialisation+inter_specialisation);
        p_shift_sp_specialisation:=1-p_contin_sp_specialisation;
   end
```

```
     else
         begin
              p_contin_sp_specialisation:=9.9999;
              p_shift_sp_specialisation:=9.9999;
         end;

     p_sp_reformulation:=(intra_reformulation+inter_reformulation)/(intra_total+inter_total);
     if(intra_reformulation+inter_reformulation>0) then
        begin
             p_contin_sp_reformulation:=intra_reformulation/(intra_reformulation+inter_reformulation);
             p_shift_sp_reformulation:=1-p_contin_sp_reformulation;
        end
     else
         begin
              p_contin_sp_reformulation:=9.9999;
              p_shift_sp_reformulation:=9.9999;
         end;

     p_sp_new:=(intra_new+inter_new)/(intra_total+inter_total);
     if(intra_new+inter_new>0) then
        begin
             p_contin_sp_new:=intra_new/(intra_new+inter_new);
             p_shift_sp_new:=1-p_contin_sp_new;
        end
     else
         begin
              p_contin_sp_new:=9.9999;
              p_shift_sp_new:=9.9999;
         end;

     p_sp_relevance_feedback:=(intra_relevance_feedback+inter_relevance_feedback)/(intra_total+inter_total);
     if(intra_relevance_feedback+inter_relevance_feedback>0) then
        begin

p_contin_sp_relevance_feedback:=intra_relevance_feedback/(intra_relevance_feedback+inter_relevance_feedback);
             p_shift_sp_relevance_feedback:=1-p_contin_sp_relevance_feedback;
        end
     else
         begin
              p_contin_sp_relevance_feedback:=9.9999;
              p_shift_sp_relevance_feedback:=9.9999;
         end;

     p_sp_others:=(intra_others+inter_others)/(intra_total+inter_total);
     if(intra_others+inter_others>0) then
        begin
             p_contin_sp_others:=intra_others/(intra_others+inter_others);
             p_shift_sp_others:=1-p_contin_sp_others;
        end
     else
         begin
              p_contin_sp_others:=9.9999;
              p_shift_sp_others:=9.9999;
         end;
     clrscr;

     writeln('Search Pattern Identification Algorithm Report');
     writeln('-------------------------------------------------------------------------------');
     writeln('SP classes         Intra-s.  Inter-s.     P(SP)   P(shift|SP)    P(contin|sp)');
     writeln('-------------------------------------------------------------------------------');
     writeln('Browsing');
     writeln('Generalisation');
     writeln('Specialisation');
     writeln('Reformulation');
     writeln('New');
     writeln('Relevance feedback');
     writeln('Others');
     writeln('-------------------------------------------------------------------------------');
     writeln('Total');
```

```
      gotoxy(22,5);writeln(intra_browsing:5);
      gotoxy(32,5);writeln(inter_browsing:5);
      gotoxy(42,5);writeln(p_sp_browsing:5:4);
      gotoxy(56,5);writeln(p_shift_sp_browsing:5:4);
      gotoxy(72,5);writeln(p_contin_sp_browsing:5:4);
      gotoxy(22,6);writeln(intra_generalisation:5);
      gotoxy(32,6);writeln(inter_generalisation:5);
      gotoxy(42,6);writeln(p_sp_generalisation:5:4);
      gotoxy(56,6);writeln(p_shift_sp_generalisation:5:4);
      gotoxy(72,6);writeln(p_contin_sp_generalisation:5:4);
      gotoxy(22,7);writeln(intra_specialisation:5);
      gotoxy(32,7);writeln(inter_specialisation:5);
      gotoxy(42,7);writeln(p_sp_specialisation:5:4);
      gotoxy(56,7);writeln(p_shift_sp_specialisation:5:4);
      gotoxy(72,7);writeln(p_contin_sp_specialisation:5:4);
      gotoxy(22,8);writeln(intra_reformulation:5);
      gotoxy(32,8);writeln(inter_reformulation:5);
      gotoxy(42,8);writeln(p_sp_reformulation:5:4);
      gotoxy(56,8);writeln(p_shift_sp_reformulation:5:4);
      gotoxy(72,8);writeln(p_contin_sp_reformulation:5:4);
      gotoxy(22,9);writeln(intra_new:5);
      gotoxy(32,9);writeln(inter_new:5);
      gotoxy(42,9);writeln(p_sp_new:5:4);
      gotoxy(56,9);writeln(p_shift_sp_new:5:4);
      gotoxy(72,9);writeln(p_contin_sp_new:5:4);
      gotoxy(22,10);writeln(intra_relevance_feedback:5);
      gotoxy(32,10);writeln(inter_relevance_feedback:5);
      gotoxy(42,10);writeln(p_sp_relevance_feedback:5:4);
      gotoxy(56,10);writeln(p_shift_sp_relevance_feedback:5:4);
      gotoxy(72,10);writeln(p_contin_sp_relevance_feedback:5:4);
      gotoxy(22,11);writeln(intra_others:5);
      gotoxy(32,11);writeln(inter_others:5);
      gotoxy(42,11);writeln(p_sp_others:5:4);
      gotoxy(56,11);writeln(p_shift_sp_others:5:4);
      gotoxy(72,11);writeln(p_contin_sp_others:5:4);

      gotoxy(22,13);writeln(intra_total:5);
      gotoxy(32,13);writeln(inter_total:5);
      writeln;
      write('Press any key to exit...');
      key:=readkey;
end.
```

Ek 2: GA ve Konu Değişikliği Belirleme Algoritması Pascal Kodu

```
program sga;
uses crt;

const maxpop    = 100;
      maxstring = 50;

type allele     = boolean;
     chromosome = array [1..maxstring] of allele;
     individual = record
                    chrom: chromosome;
                    Tshift,Wti,Wsp: real;
                    fitness: real;
                    parent1,
                    parent2,
                    xsite: integer;
                  end;
     population = array [1..maxpop] of individual;

var
  oldpop,newpop: population;
  popsize, lchrom, gen, maxgen: integer;
  pcross, pmutation, sumfitness: real;
  nmutation, ncross: integer;
  avg, max, min: real;
  lchrom1, lchrom2, lchrom3: integer;
  noip:integer;
  Mti_sp_ps,Mti_ps,Msp_ps,Mti_pc,Msp_pc,Mti_teta,Msp_teta:real;
  Pshift_ti,Pshift_sp,Pcontin_ti,Pcontin_sp:real;
  IPswitch,Nshift,Ncontin:integer;
  Nshift_and_correct,Ncontin_and_correct,Ntrue_shift:integer;
  beta:real;
  P,R,Fbeta:real;
  Type_A_Error,Type_B_Error:integer;
  Pbest,Rbest,Fbeta_best:real;
  Wti_best,Wsp_best,Tshift_best:real;
  filevar_ip,filevar_sp,filevar_ti,filevar_scv:text;
  ip1,ip2:string[20];
  ti1,ti2:real;
  sp1,sp2:byte;
  scv1,scv2:byte;

function power(x,y:real):real;
begin
  power:= exp(y*ln(x));
end;

function rand: real;
begin
  rand:= random(23767)/23767.0;
end;

function flip(probability: real): boolean;
begin
  if probability = 1.0 then
    flip:= true
  else
    flip:= (rand <= probability);
end;

function rnd(low,high: integer): integer;
var
  i: integer;
begin
  rnd:= random(high-low) + low;
end;
```

```
procedure pause;
var
  ch: char;
begin
  writeln;
  write('Press any key to continue...');
  ch:= '*';
  repeat
    ch:= ReadKey;
  until (KeyPressed) or (ch<>'*');
end;

function objfunc(Tshift,Wti,Wsp: real): real;
var
   j,j1:integer;
   code:integer;
   valcode:integer;
   key:char;
   i:integer;

   Nshiftdummy:integer;

procedure calcprob;
begin
     assign(filevar_ip,'c:\tp\program\veriler\excite\ilk_5000\ip_f.txt');
     {$i-} reset(filevar_ip); code:=ioresult; {$i+}
     if code<>0 then
        begin
             clrscr;
             writeln('File not found!');
             key:=readkey;
             halt;
        end;

     assign(filevar_sp,'c:\tp\program\veriler\excite\ilk_5000\sp_f.txt');
     {$i-} reset(filevar_sp); code:=ioresult; {$i+}
     if code<>0 then
        begin
             clrscr;
             writeln('File not found!');
             key:=readkey;
             halt;
        end;

     assign(filevar_ti,'c:\tp\program\veriler\excite\ilk_5000\ti_f.txt');
     {$i-} reset(filevar_ti); code:=ioresult; {$i+}
     if code<>0 then
        begin
             clrscr;
             writeln('File not found!');
             key:=readkey;
             halt;
        end;

     assign(filevar_scv,'c:\tp\program\veriler\excite\ilk_5000\scv_f.txt');
     {$i-} reset(filevar_scv); code:=ioresult; {$i+}
     if code<>0 then
        begin
             clrscr;
             writeln('File not found!');
             key:=readkey;
             halt;
        end;

     j1:=0;
     IPswitch:=0;
     Nshift:=0;
     Ncontin:=0;
     Nshift_and_correct:=0;
     Ncontin_and_correct:=0;
     readln(filevar_ip,ip1);
     readln(filevar_ti,ti1);
     readln(filevar_sp,sp1);
     readln(filevar_scv,scv1);
```

```
     while not eof(filevar_ip) do
           begin
                readln(filevar_ip,ip2);
                readln(filevar_ti,ti2);
                readln(filevar_sp,sp2);
                readln(filevar_scv,scv2);
                if ip1<>ip2 then
                   begin
                        ip1:=ip2;
                        inc(IPswitch);
                   end
                else
                    begin

                         (*Change these values for the problem!*)
                         if (ti1>=0)  and (ti1<5)  then Pshift_ti:=0.0250;
                         if (ti1>=5)  and (ti1<10) then Pshift_ti:=0.0763;
                         if (ti1>=10) and (ti1<15) then Pshift_ti:=0.1414;
                         if (ti1>=15) and (ti1<20) then Pshift_ti:=0.1296;
                         if (ti1>=20) and (ti1<25) then Pshift_ti:=0.3714;
                         if (ti1>=25) and (ti1<30) then Pshift_ti:=0.2000;
                         if (ti1>=30)              then Pshift_ti:=0.4720;
                         (************************************)

                         Pcontin_ti:=1-Pshift_ti;

                         (*Change these values for the problem!*)
                         if (sp1=1) then Pshift_sp:=0.0000;
                         if (sp1=2) then Pshift_sp:=0.0000;
                         if (sp1=3) then Pshift_sp:=0.0000;
                         if (sp1=4) then Pshift_sp:=0.0030;
                         if (sp1=5) then Pshift_sp:=0.3011;
                         if (sp1=6) then Pshift_sp:=0.0000;
                         if (sp1=7) then Pshift_sp:=0.0000;
                         (************************************)

                         Pcontin_sp:=1-Pshift_sp;

                         Mti_ps:=Pshift_ti*Wti;
                         Msp_ps:=Pshift_sp*Wsp;
                         Mti_pc:=Pcontin_ti*Wti;
                         Msp_pc:=Pcontin_sp*Wsp;
                         Mti_teta:=1-Mti_ps-Mti_pc;
                         Msp_teta:=1-Msp_ps-Msp_pc;

                         Mti_sp_ps:=((Mti_ps*Msp_ps)+(Mti_ps*Msp_teta)+
                                     (Mti_teta*Msp_ps))/(1-(Mti_ps*Msp_pc)-(Mti_pc*Msp_ps));

                         if Mti_sp_ps>Tshift then
                            begin
                                 inc(Nshift);
                                 if (scv1=1) then inc(Type_A_Error);
                                 if (scv1=2) then inc(Nshift_and_correct);
                            end
                         else
                             begin
                                  inc(Ncontin);
                                  if (scv1=2) then inc(Type_B_Error);
                                  if (scv1=1) then inc(Ncontin_and_correct);
                             end;
                    end;
           ip1:=ip2;
           ti1:=ti2;
           sp1:=sp2;
           scv1:=scv2;
           end;
close(filevar_ip);
close(filevar_ti);
close(filevar_sp);
close(filevar_scv);
end;
```

```
begin
         calcprob;
         (*****************************************)
         (*For Excite dataset use "Ntrue_shift:=269;".*)
         (*For Fast dataset use "Ntrue_shift:=386;".***)
         (*****************************************)
         Ntrue_shift:=269;
         beta:=1.5;
         if(Nshift=0)
            then P:=0
            else P:=(Nshift_and_correct)/(Nshift);
         R:=Nshift_and_correct/Ntrue_shift;
         if (P=0) and (R=0)
            then Fbeta:=0
            else Fbeta:=((1+power(beta,2))*P*R)/(power(beta,2)*P+R);
         if Fbeta>Fbeta_best then
            begin
                 Wti_best:=Wti;
                 Wsp_best:=Wsp;
                 Tshift_best:=Tshift;
                 Pbest:=P;
                 Rbest:=R;
                 Fbeta_best:=Fbeta;
            end;
         objfunc:=Fbeta;
end;

function decode(chrom: chromosome; lbits, vi: integer): real;
var
  j,sp,ep: integer;
  accum, powerof2: real;
begin
  accum:= 0.0;
  powerof2:= 1;
  if vi=1 then
     begin
          sp:=1;
          ep:=10;
     end;
  if vi=2 then
     begin
          sp:=11;
          ep:=20;
     end;
  if vi=3 then
     begin
          sp:=21;
          ep:=30;
     end;
  for j:= sp to ep do begin
    if chrom[j] then accum:= accum + powerof2;
    powerof2:= powerof2 * 2;
  end;
  decode:= accum/1024;
end;

procedure statistics(pop: population);
var
  j: integer;
begin
  sumfitness:= pop[1].fitness;
  min       := pop[1].fitness;
  max       := pop[1].fitness;
  for j:= 2 to popsize do with pop[j] do begin
    sumfitness:= sumfitness + fitness;
    if fitness>max then max:= fitness;
    if fitness<min then min:= fitness;
  end;
  avg:= sumfitness/popsize;
end;
```

```
procedure initdata;
var
  j: integer;
begin
  clrscr;
  writeln('------------------------------------------------------------------------------');
  writeln('          Genetic Algorithm Program - Automatic Topic Change Identification    ');
  writeln('------------------------------------------------------------------------------');
  writeln;
  writeln('Entering Initial Genaration Datas');
  writeln('---------------------------------');
  writeln('Population Size      : ');
  writeln('Max Generations      : ');
  writeln('Crossover Probability: ');
  writeln('Mutation Probability : ');
  writeln('---------------------------------');
  gotoxy(24,7); readln(popsize);
  gotoxy(24,8); readln(maxgen);
  gotoxy(24,9); readln(pcross);
  gotoxy(24,10); readln(pmutation);
  writeln;

  (*Change these values for the problem.*)
  lchrom:=30;
  lchrom1:=10;
  lchrom2:=10;
  lchrom3:=10;
  (************************************)

  randomize;
  nmutation:= 0;
  ncross:= 0;
  Pbest:=0;
  Rbest:=0;
  Fbeta_best:=0;
  pause;
  writeln;
  write('Working...!');
end;

procedure initreport;
begin
  writeln;
  writeln;
  writeln('Initial Generation Statistics    ');
  writeln('---------------------------------');
  writeln('Maximum Fitness:', max);
  writeln('Average Fitness:', avg);
  writeln('Minimum Fitness:', min);
  writeln('Sum of Fitness :',sumfitness);
  writeln('---------------------------------');
  pause;
end;

procedure initpop;
var
  j, j1: integer;
begin
  for j:= 1 to popsize do with oldpop[j] do begin
    for j1:= 1 to 10 do chrom[j1]:= flip(0.5);
    Tshift:= decode(chrom,lchrom1,1);
    for j1:= 11 to 20 do chrom[j1]:= flip(0.5);
    Wti:= decode(chrom,lchrom2,2);
    for j1:= 21 to 30 do chrom[j1]:= flip(0.5);
    Wsp:= decode(chrom,lchrom3,3);
    fitness:= objfunc(Tshift,Wti,Wsp);
    parent1:= 0; parent2:= 0; xsite:= 0;
  end;
end;
```

```
procedure initialize;
begin
  initdata;
  initpop;
  statistics(oldpop);
  initreport;
end;

procedure writechrom(chrom: chromosome);
var
  j: integer;
begin
  for j:= lchrom downto 1 do
    if chrom[j] then
      write('1')
    else
      write('0')
end;

procedure report;
var
  j: integer;
begin
  clrscr;
  writeln('Population Report');
  writeln('-------------------------------------------------------------------------------');
  writeln('Generation: ',gen);
  writeln('-------------------------------------------------------------------------------');
  writeln('            String                   Wsp        Wti      Tshift      Fbeta ');
  writeln('------------------------------    ---------  ---------  ---------    ------');
  for j:= 1 to popsize do begin
    with newpop[j] do begin
      writechrom(chrom);
      writeln(' ->',Wsp:10,' ',Wti:10,' ',Tshift:10,' -> ',fitness:6:4);
    end;
  end;
  writeln('-------------------------------------------------------------------------------');
  writeln('maximum: ',max:6:4,'     minimum: ',min:6:4,'     avarage: ',avg:6:4);
  writeln('-------------------------------------------------------------------------------');
  writeln('sumfitness: ',sumfitness:6:4,'     nmutation: ',nmutation,'     ncross: ',ncross);
  writeln('-------------------------------------------------------------------------------');
  writeln('Wti: ',Wti_best:6:4,'  Wsp: ',Wsp_best:6:4,'  Tshift: ',Tshift_best:6:4,
          '  P: ',Pbest:6:4,  '  R: ',Rbest:6:4,  '  Fbt: ',Fbeta_best:6:4);
  pause;
end;

function select: integer;
var
  rand, partsum: real;
  j: integer;
begin
  partsum:= 0.0; j:= 0;
  rand:= random * sumfitness;
  repeat
    j:= j + 1;
    partsum:= partsum + oldpop[j].fitness;
  until (partsum >= rand) or (j = popsize);
  select:= j;
end;

function mutation(alleleval: allele): allele;
var
  mutate: boolean;
begin
  mutate:= flip(pmutation);
  if mutate then begin
    nmutation:= nmutation + 1;
    mutation:= not alleleval;
  end else
    mutation:= alleleval;
end;
```

```
procedure crossover(parent1, parent2: chromosome;
                    var child1, child2: chromosome;
                    var jcross: integer);
var
  j: integer;
begin
  if flip(pcross) then begin
    jcross:= rnd(1,lchrom-1);
    ncross:= ncross + 1;
  end else
    jcross:= lchrom;

  for j:= 1 to jcross do begin
    child1[j]:= mutation(parent1[j]);
    child2[j]:= mutation(parent2[j]);
  end;
  if jcross<>lchrom then begin
    for j:= jcross+1 to lchrom do begin
      child1[j]:= mutation(parent2[j]);
      child2[j]:= mutation(parent1[j]);
    end;
  end;
end;

procedure generation;
var
  j, mate1, mate2, jcross: integer;
begin
  j:= 1;
  repeat
    mate1:= select;
    mate2:= select;
    crossover(oldpop[mate1].chrom, oldpop[mate2].chrom, newpop[j].chrom,
              newpop[j+1].chrom,jcross);
    with newpop[j] do begin
      Tshift:= decode(chrom,lchrom1,1);
      Wti:= decode(chrom,lchrom2,2);
      Wsp:= decode(chrom,lchrom3,3);

      fitness:= objfunc(Tshift,Wti,Wsp);
      parent1:= mate1;
      parent2:= mate2;
      xsite:= jcross;
    end;
    with newpop[j+1] do begin
      Tshift:= decode(chrom,lchrom1,1);
      Wti:= decode(chrom,lchrom2,2);
      Wsp:= decode(chrom,lchrom3,3);
      fitness:= objfunc(Tshift,Wti,Wsp);
      parent1:= mate1;
      parent2:= mate2;
      xsite:= jcross;
    end;
    j:= j + 2;
  until j>=popsize;
end;

begin
  gen:= 0;
  initialize;
  repeat
    gen:= gen + 1;
    generation;
    statistics(newpop);
    if((gen)mod(maxgen)=0) then report;
    oldpop:= newpop;
  until (gen>= maxgen);
end.
```

Ek 3: YSA ve Konu Değişikliği Belirleme Algoritması MATLAB Kodu

```
function ysa_fast(esik_degeri);

baslangic_zamani = cputime;

dosya_csi_egitim = fopen('program\tez\veriler\excite\csi_egitim_5014.txt');
dosya_ti_egitim = fopen('program\tez\veriler\excite\ti_egitim_5014.txt');
dosya_sp_egitim = fopen('program\tez\veriler\excite\sp_egitim_5014.txt');
dosya_csi_test = fopen('program\tez\veriler\excite\csi_test.txt');
dosya_ti_test = fopen('program\tez\veriler\excite\ti_test.txt');
dosya_sp_test = fopen('program\tez\veriler\excite\sp_test.txt');

csi_egitim = fscanf(dosya_csi_egitim,'%i',[1 inf]); % contin=1, shift=2, ip change=3
ti_egitim = fscanf(dosya_ti_egitim,'%g',[1 inf]);
sp_egitim = fscanf(dosya_sp_egitim,'%i',[1 inf]);
csi_test = fscanf(dosya_csi_test,'%i',[1 inf]); % contin=1, shift=2, ip change=3
ti_test = fscanf(dosya_ti_test,'%g',[1 inf]);
sp_test = fscanf(dosya_sp_test,'%i',[1 inf]);

j = 1;

for i = 1:length(csi_egitim)
    if csi_egitim(1,i) == 1 || csi_egitim(1,i) == 2
        csi_egitim_ysa(j) = csi_egitim(1,i);
        ti_egitim_ysa(j) = ti_egitim(1,i);
        sp_egitim_ysa(j) = sp_egitim(1,i);
        j = j+1;
    end
end

evs = j-1; %evs=Eğitim Verilerinin Sayısı

j = 1;

for i = 1:length(csi_test)
    if csi_test(1,i) == 1 || csi_test(1,i) == 2
        csi_test_ysa(j) = csi_test(1,i);
        ti_test_ysa(j) = ti_test(1,i);
        sp_test_ysa(j) = sp_test(1,i);
        j = j+1;
    end
end

tvs = j-1; %tvs=Test Verilerinin Sayısı

for i = 1:evs
    if ti_egitim_ysa(i) < 5
        ti_egitim_ysa_grup(i) = 1;
    elseif ti_egitim_ysa(i) < 10
        ti_egitim_ysa_grup(i) = 2;
    elseif ti_egitim_ysa(i) < 15
        ti_egitim_ysa_grup(i) = 3;
    elseif ti_egitim_ysa(i) < 20
        ti_egitim_ysa_grup(i) = 4;
    elseif ti_egitim_ysa(i) < 25
        ti_egitim_ysa_grup(i) = 5;
    elseif ti_egitim_ysa(i) < 30
        ti_egitim_ysa_grup(i) = 6;
    else
        ti_egitim_ysa_grup(i) = 7;
    end
end
```

```
for i = 1:tvs
    if ti_test_ysa(i) <= 5
        ti_test_ysa_grup(i) = 1;
    elseif ti_test_ysa(i) <= 10
        ti_test_ysa_grup(i) = 2;
    elseif ti_test_ysa(i) <= 15
        ti_test_ysa_grup(i) = 3;
    elseif ti_test_ysa(i) <= 20
        ti_test_ysa_grup(i) = 4;
    elseif ti_test_ysa(i) <= 25
        ti_test_ysa_grup(i) = 5;
    elseif ti_test_ysa(i) <= 30
        ti_test_ysa_grup(i) = 6;
    else
        ti_test_ysa_grup(i) = 7;
    end
end

fclose(dosya_csi_egitim);
fclose(dosya_ti_egitim);
fclose(dosya_sp_egitim);
fclose(dosya_csi_test);
fclose(dosya_ti_test);
fclose(dosya_sp_test);

ti_egitim_ysa_min = min(ti_egitim_ysa); %Minimum ti değeri.
ti_egitim_ysa_max = max(ti_egitim_ysa); %Maksimum ti değeri.
ti_egitim_ysa_grup_min = min(ti_egitim_ysa_grup); %Minimum ti grubu değeri.
ti_egitim_ysa_grup_max = max(ti_egitim_ysa_grup); %Maksimum ti grubu değeri.
sp_egitim_ysa_min = min(sp_egitim_ysa); %Minimum sp değeri.
sp_egitim_ysa_max = max(sp_egitim_ysa); %Maksimum sp değeri.

ti_test_ysa_min = min(ti_test_ysa); %Minimum ti değeri.
ti_test_ysa_max = max(ti_test_ysa); %Maksimum ti değeri.
ti_test_ysa_grup_min = min(ti_test_ysa_grup); %Minimum ti grubu değeri.
ti_test_ysa_grup_max = max(ti_test_ysa_grup); %Maksimum ti grubu değeri.
sp_test_ysa_min = min(sp_test_ysa); %Minimum sp değeri.
sp_test_ysa_max = max(sp_test_ysa); %Maksimum sp değeri.

ysa_girisleri=[ti_egitim_ysa_grup; sp_egitim_ysa];
ysa_cikislari=csi_egitim_ysa;

ysa=newff([minmax(ti_egitim_ysa_grup); minmax(sp_egitim_ysa_)],[5 1],{'tansig','purelin'},'traingdm');
ysa=init(ysa);

ysa.trainparam.show=50;
ysa.trainparam.lr=0.1;
ysa.trainparam.mc=0.8;
ysa.trainparam.epochs=1000;
ysa.trainparam.goal=0.05;
[ysa,egitim]=train(ysa, ysa_girisleri, ysa_cikislari);

close all;

dcs=0; %dcs=Doğru contin Sayısı
dss=0; %dss=Doğru shift Sayısı

cois=0; %cois=contin Olarak İşaretlenenlerin Sayısı
sois=0; %sois=shift Olarak İşaretlenenlerin Sayısı

gss=0; %gss=Gerçek shift Sayısı

a_tipi_hata=0;
b_tipi_hata=0;

esik_degeri
```

```
for i=1:tvs
    cevap=sim(ysa,[ti_test_ysa_grup(i); sp_test_ysa(i)]);
    if cevap<esik_degeri
        cevap=1;
        cevap_vektor(i)=1;
    else cevap=2;
        cevap_vektor(i)=2;
    end
    if csi_test_ysa(i)==2
        gss=gss+1;
    end
    if cevap==1
        cois=cois+1;
    elseif cevap==2
        sois=sois+1;
    end
    if cevap==csi_test_ysa(i)
        if cevap==1
            dcs=dcs+1;
        end
        if cevap==2
            dss=dss+1;
        end
    elseif cevap==2 && csi_test_ysa(i)==1
        a_tipi_hata=a_tipi_hata+1;
    elseif cevap==1 && csi_test_ysa(i)==2
        b_tipi_hata=b_tipi_hata+1;
    end
end

toplam_hata = a_tipi_hata + b_tipi_hata

if sois==0
    p=0
else
    p=dss/sois
end

r=dss/gss

beta=1.5;

if p==0 && r==0
    f_beta=0
else
    f_beta=((1+beta^2)*p*r)/((beta^2)*p+r)
end

global toplam_tekrar_sayisi;
wsy='program\tez\veriler\sonuclar\excite\workspaces\'; %wsy: Workspace Yolu
if toplam_tekrar_sayisi<10
    wsa1='ws00';
elseif toplam_tekrar_sayisi<100
    wsa1='ws0';
else
    wsa1='ws';
end
wsa2=int2str(toplam_tekrar_sayisi);
wsid=strcat(wsa1,wsa2); %wsid: Workspace ID
wsa3='.mat';
wsa=strcat(wsa1,wsa2,wsa3); %wsa: Workspace Adı
wsd=strcat(wsy,wsa); %wsd: Workspace Dosyası
save(wsd);

calisma_suresi = cputime - baslangic_zamani

dosya_rapor=fopen('program\tez\veriler\sonuclar\excite\ysa_excite.txt','a');
fprintf(dosya_rapor,'%6s %2.1f %5d %5d %5d %5d %5d %5.4f %5.4f %5.4f %8.4f\n', ...
        wsid,esik_degeri,dcs,dss,a_tipi_hata,b_tipi_hata,toplam_hata,p,r,f_beta,calisma_suresi);
fclose(dosya_rapor);
```