Borgelt, Christian, and Rudolf Kruse. 2006. Section 3.4 Artificial Intelligence Methodologies, pp. 153-168 of Chapter 3 Methods, Algorithms, and Software, in CIGR Handbook of Agricultural Engineering Volume VI Information Technology. Edited by CIGR-The International Commission of Agricultural Engineering; Volume Editor, Axel Munack. St. Joseph, Michigan, USA: ASABE. *Copyright American Society of Agricultural Engineers.*

Çevirmen: Yusuf DİLAY
Çeviri Editörleri: Sefa TARHAN ve Mehmet Metin ÖZGÜVEN

3.4 Yapay Zeka Yöntemleri

Yazarlar: C. Borgelt ve R. Kruse

Çevirmen: Yusuf DİLAY

Özet: *Yapay Zeka bilgisayar ve robot gibi insan yapımı araçlar kullanarak insanlar ve hayvanlar gibi doğal sistemleri taklit etmekle alakalıdır. Bu yöntem, bilginin - özellikle de kesin olmayan belirsiz bilginin- bilgisayar hafızasında depolanabilmesi ve bu bilgiden otomatik olarak çıkarımlar yapılabilmesi amacıyla bilginin nasıl temsil edilebileceğini anlamayı içermektedir. Ayrıca depolanan bilgiyi esas alarak kararların nasıl yapılabileceği ve eylem planlarının nasıl oluşturulabileceği ve örnek veriden öğrenerek veya insan uzmanları sorgulayarak bilgisayarda işlenebilir bilginin nasıl edinebileceğini anlamayı da içermektedir.*
Açıkçası bu bölüm sadece bazı temel yöntemleri vurgular ve birkaç temel yaklaşıma değinir. Bu konuyla ilgilenen bir okuyucu için daha detaylı açıklamalar 1 2 de bulunabilir. Yapay zekanın belli alanlarına (formal mantık, bulanık mantık, yapay sinir ağları vb.) yönelik kitaplara ilgili bölümlerde değinilmiştir.

Anahtar Kelimeler: *Yapay zeka, Bilgi sunumu, Mantık yürütme, Planlama, Öğrenme*

3.4.1 Bilgi Sunumu ve Mantık Yürütme

Zekayı tanımlamak imkansız değilse de zor olmasına rağmen, genelde neden gösterme ve sonuç çıkarma yeteneklerinin zekanın kilit özelliklerinden ikisi olduğu kabul edilmektedir. Bu nedenle, eğer robotların ve bilgisayarların zekice hareketler sergilemelerini istiyorsak, onlara mantık yürütmek için gerekli olan araçları sağlamalıyız. Yani onları bilgiden sonuçlar çıkarmalarına yönelik uygun bir şekilde kodlanmış bilgi ve prosedürlerle donatmalıyız. Bunu yaparken, sahip olduğumuz bilgilerin çoğu kesin ve tam olmadığı için belirsizlik ile muğlaklığı ve bu olguları ele almaya yönelik yöntemler üzerinde durmalıyız.

Formal Mantık

Bir başkasına bilgi aktarmak istiyorsak ya da vermek zorunda olduğumuz bir kararla ilgili lehte ve karşı olan argümanları analiz etmek istiyorsak, bilgiyi ya da argümanları bir dilde ifade etmeliyiz. Bu amaçla, kullanılan dilin belli bir yapı sergilemesi gerekir. Tabi ki İngilizce, Fransızca, Almanca gibi bütün doğal diller

gerekli yapıya sahiptir. Fakat insanlar arası iletişimde bazı kelimelerin anlamlarını tam belirlemek için bazı ifadelerin yapıldığı bağlam ve aynı zamanda ortak bilgi dolaylı olarak kullanıldığı için, doğal diller genellikle bilgisayar ifadeleri için yeterince net değildir. Bunun yanısıra, doğal dil çok esnektir ve aynı ifadeyi çok farklı yollarla söylemeye imkan sağlar. Her ne kadar bu insanlar arası iletişimde kesinlikle bir avantaj olsa da, her hangi iki ifadenin aynı şeyi söyleyip söylemediğini otomatik olarak kontrol etmek çok zor olduğundan, bu durum bilgisayarda bilginin temsilinde bir engel olarak karşımıza çıkar.

Bu nedenle, yapay zekada her bir terimin anlamının net bir şekilde belirlendiği ve mantık yürütmek için gerekli çekirdek yapıyı veren formülleştirilmiş diller kullanılır. Bu özel diller (formal) mantık alanında çalışılmaktadır: *Mantık, bir kişinin üzerinde tartışabileceği nedensel dillerin çekirdek yapısını oluşturmaktadır.* Formal mantık doğal dilin karmaşıklığını ve çift anlamlılığını, bilgisayarda bilgiyi yönetmeyi mümkün kılacak seviyeye indirir.

Karmaşıklık indirgemenin ne kadar ileri gittiğine bağlı olarak farklı mantıksal hesaplamalar vardır. En temel ve basit olanı *önermesel mantıktır.* Kombine ifadelerin gerçeklik değerinin, kendisini oluşturan temel ifadelerin doğruluk değeriyle ilişkisini tanımlar. Dikkate alınan temel ifadeler "Hanibal bir köpektir" gibi basit önermeler içermektedir. Bu tür önermeler bir yapıya sahip olmasına rağmen, bu yapı göz ardı edilmektedir. Bu ifadeler basit gerçeklik değişkenleriyle ifade edilir ve *doğru* ya da *yanlış* değerlerinden birini alır. Gerçeklik değişkenleri, *ve, veya, değil* ve *eğer* gibi *mantıksal bağlantılarla* bağlanabilirler. Önermesel mantık, örneğin sadece ve sadece A ve B her ikisi de yanlış ise "A ve B" önermesi yanlış olduğunu ifade etmektedir. Bu *gerçeklik işlevselliği* basit çıkarımlar yapmamızı sağlar. Eğer "A ve B" bileşik ifadesinin doğru olduğunu biliyorsak ve daha sonra *A*'nın yanlış olduğunu öğrenirsek, *B*'nin doğru olması gerektiği çıkarımını yapabiliriz.

Yine de, önermesel mantık basit önermelerin yapısını göz ardı ettiği için çoğu uygulama için yeterince güçlü değildir. Bu nedenle önermesel mantığın bir uzantısı, tam adıyla (birinci dereceden) yüklemler (teyit) mantığı sık sık kullanılır. Önermesel mantık gibi yüklem mantığı da mantıksal bağlayıcıların gerçek işlevselliğini modeller. Buna ek olarak, *sabitler, değişkenler, işlevler ve teyitler* kullanarak basit önermelerin yapısını da (kısmen) yakalar. Sabitler belli nesneleri (doğal dildeki isimler gibi) belirtir. Örneğin, "John Steinback" sabiti belli bir Amerikan yazarını gösterir. Değişkenler, "Hepsi için.." ve " .....vardır" gibi sözde *miktar belirteçleri* ile ilişkili olarak, evrensel ya da var olan ifadeleri kurmak için belirgin olmayan bir yolla nesneleri referans eder. Bir nesneye başka nesneler yoluya değinmek gibi fonksiyonlar dolaylı karakterizasyonları modellemektedir. Örneğin, John Steinback'den aynı zamanda "Gazap Üzümleri'nin yazarı" olarak bahsedebiliriz. Burada "yazar", "Gazap Üzümleri" sabitine uygulanan bir fonksiyondur. Son olarak, teyitler nesnelere özellik atfetmek veya nesneler arasındaki ilişkileri tanımlamak için kullanılırlar. Örneğin, John Steinbeck'in "Gazap Üzümleri"nin 1939'da yazdığı

gerçeği "yazdı ('John Steinbeck', 'gazap üzümleri',1939)" olarak ifade edilebilir. Burada "yazdı" bir yüklemdir ve onun argümanları sabitlerdir. Teyit ifadelerinin doğru ya da yanlış olabileceğini ve bu nedenle önermesel mantığın gerçeklik değişkenlerine karşılık geldiklerine de dikkat edilmelidir.

Yüklem analizi ile daha genel çıkarımlar mümkün olabilir. Örneğin, "bütün x ler için: eğer x bir insan ise, x ölümlü olmalıdır"şeklinde ifade edilen tüm insanlar ölümlüdür gerçeğini kullanarak, "Sokrates bir insandır" gerçeğinden Sokrates ölümlü olmalıdır çıkarımında bulunabiliriz. Yani yüklem analizi ile nesnelerin özellikleri ve nesneler arasındaki ilişkiler hakkında uygun bir şekilde mantık yürütebiliriz. Yapay zekada, mantıksal formüllerle tanımlanan durumları sorgulayan ve bu durumlar hakkındaki soruları cevaplayabilen *otomatik teorem kanıtlayıcılarında* yüklem mantığı kullanılmaktadır. Özel yüklem mantığı alt dizinlerine dayalı olan ve yapay zeka uygulamaları için çok güçlü araçlar olan PROLOG gibi *özel programlama dilleri* vardır.

Mantığın gelişmiş uzantıları doğal dilin özelliklerini daha fazla yakalamaktadır: örneğin, *temporal (zamansal) mantık* zaman hakkında mantık yürütmeyi mümkün kılar ve zaman içerisinde doğruluk değerlerinde değişiklikleri ele alır ("güneş parlıyor" bugün doğru ama yarın için yalnış olabilir). *Modal mantık* ifadeleri *olası veya gerekli* olarak işaretlememizi sağlar ve böylece belirsizliğin (sınırlıda olsa) ortadan kaldırılmasına yönelik imkanlar sağlar. Fakat standart (birinci dereceden) yüklem analizinin bu uzantılarının tartışılması bu bölümün kapsamının dışındadır. Farklı mantıksal analizler ve onların uygulamalarına yönelik daha fazla detay [3-5] de bulunabilir.

Kural Tabanlı Sistemler

Kurallar, yani eğer o zaman ifadeleri (if-then ifadeleri), genel de kolay anlaşılabilir olarak düşünüldüklerinden bilginin birçok formunun uygun sunum şekilleridir. Kurallar yapılabilecek çıkarımları belirlemek veya eylemleri altında uygulanmak zorunda oldukları koşullar ile ilişkilendirmek için kullanılabilir. Bu nedenle, kurallar uzman veya karar destek sistemlerinin ve aynı zamanda bilgi tabanlı kontrollerin en popüler yapı taşlarıdır. İnsan düşüncesiyle bilgisayarda ifade için gerekli olan soyutlama ve formal netliği uzlaştırmanın en iyi yollarından birini sağlar görünmektedirler.

Kural tabanlı bir sistem genellikle bir *bilgi tabanı*, bir *çıkarım motoru* ve *uygulama arayüzü'*den oluşmaktadır. Bilgi tabanı uygulama alanı hakkında bilgileri temsil eden kuralları içermektedir. Çıkarım motoru, hangi kuralların uygulanabileceğini kontrol eder ve çıkarımlarda bulunur veya eylemleri tetikler. Uygulama arayüzü kullanıcının sistemle iletişim kurabilmesini sağlar veya hangi sensör bilgisinin kontrolöre aktarılacağına aracılık eder. Karar destek sistemi sistem tarafından yapılan önerilerilere yönelik gerekçeler üreten bir açıklama öğesi de içerebilir. Sorumluluk, bir kararı bilinçli şekilde yapabilmek için gerekli sebepleri

bilmek isteyen karar verici insanın elinde olduğu için yukarıda açıklanan durum gereklidir. Bilgi tabanlı kontrolörde ek bir öğe olarak durum değişkenleri de olabilir. Bu öğe kontrolörün davranışını önceki kontrol durumuna bağlı hale getirmek, yani zaman gecikmelerini halledebilecek kontrol stratejilerini programlamak için kullanılır.

Belirsiz (Uncertain) Bilgi

Uzman insanların bilgisi nadiren kesin olmaktadır. Çoğu kurallar sadece genellikle doğrudur ve istisnalara sahiptir. En iyi bilinen örnek tabiki "bütün kuşlar uçabilir" ifadesidir. Bir çok kuş uçabilse de penguen ve devekuşu gibi kuşlar uçamaz öyleyse "Tweety bir kuştur"ifadesinden "Tweety uçabilir"çıkarımında bulunursak yalnış olabilir. Fakat çoğu bilgi kesin değildir demek bu bilgiler faydasızdır demek değildir. Örneğin tıpta belli belirtiler genellikle (ama her zaman değil) belli hastalıklara işaret edebilir. Bu nedenle doktor genelde hastalığı kesin olarak bilemeyeceği bir durumdadır. Fakat teşhisleri genellikle doğrudur. Ne yazık ki sadece mantık tabanlı yaklaşımlarla, genellikle doğru fakat istisnaları olan kurallar kolaylıkla uygulanamaz. Klasik mantıkta bir ifadenin ya doğru ya da yanlış olması problemin kendisidir. "Genellikle doğru" bir ifade kesin anlamda ne tamamen doğru nede yanlış olduğundan dolayı tutarlılık problemlerini tanıtmadan kullanılamaz. Aksine tam bir bilgi olmadığı sürece yukarıda bahsettiğimiz modal mantıkla "genel doğru" kurallarını kullanılır olarak işaretlenebilir. Fakat, sadece "genelde doğru" bilgiden çıkarılan sonuçlar arasında çıkarım yapmak ve tercihleri ifade etmek için yeterince güçlü araçlardan yoksundur.

Kesin olmayan bilgiyi ele almaya yönelik en doğal yaklaşım, doğru ile yanlış arasında ek doğru değerleri tanıtarak doğru kavramını yumuşatmaktır. Örneğin, birisi "belki" seviyelerini tanıtabilir veya 0 yanlışla 1 ise doğruyla eşleştirerek, 0 ile 1 arasında gerçek sayılar kullanabilir. Kural tabanlı sistemlerin, kuralların güvenirliliğini belirten kesinlik faktörleriyle ve bu faktörler için kombinasyon kuralları ile kuvvetlendirilmesi bakteriojenik enfeksiyonların teşhisi gibi bazı özel uygulamalarda başarılı olmuştur. Fakat, onların genel durumda tutarsız sonuçlar verdiği görülmüştür. Bu yaklaşımdaki ana problemler, birçok uygulamada tamamen gerekçelendirilemeyen ve bu yaklaşımın içinde gizli olan örtülü bağımsızlık kabulleridir.

Belirsizliği ele almak için, son yıllarda populerlik kazanmış ve çok daha fazla gelecek vaad eden bir yaklaşımı ise Bayesian ağlarıdır. Bayesian ağlarının altında yatan temel fikir, bağımlılık ve bağımsızlık ilişkilerini (ilgi alanı tanımlamak için kullanılan özellikler arasındaki ilişkiler) açık hale getirmektir ve böylece örtülü bağımsızlık varsayımlarından kaynaklanan sorunlardan kaçınılabilir. Bağımlılık ve bağımsızlık ilişkileri, her bir düğümün bir özelliği temsil ettiği ve özellikler arasında doğrudan bağımlı sınırların olduğu bir grafik yardımıyla kodlanır. Bu grafikten basit grafik teorisi kriterleriyle geçerli bağımsızlıklar elde edilebilir. Grafik (tahminsel)

aynı zamanda alan konusundaki (olasılık) bilgisinin nasıl parçalanabileceğini belirler. Sonuç olarak, olasılık çıkarım yollarını tayin eder ve böylece matematiksel olarak sağlam ve etkin olan kanıt çoğaltma yöntemleri çıkarılabilir. Fakat, bu dekompozisyonun ve çıkarım sürecinin matematiği burada tartışılmayacak kadar karmaşıktır. İlgilenen okuyucu 6 ve 7. bölümlerde Bayesian ağları ve yaklaşımlarla ilgili detaylı bilgi bulabilir.

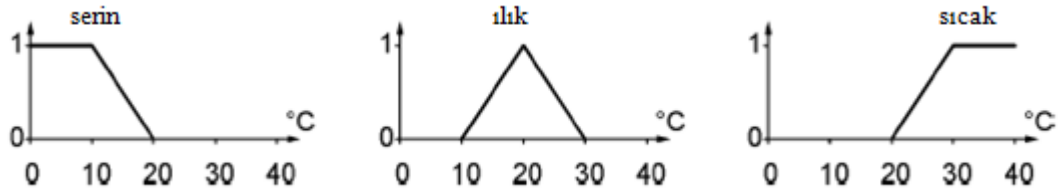
Muğlak (Vague) Bilgi

Uzman insanların bilgisi sadece *belirsiz* değil aynı zamanda sıklıkla *muğlak*tır. Belirsizlik *(*uncertainty), genellikle bilgi eksikliğimizden dolayı elde edilen durumu, alternatif ifadelerin oluşturduğu kümelerden hangisinin doğru bir şekilde tanımlayabileceğine karar veremediğimiz anlamına gelmektedir. Muğlaklık (vagueness) ise doğal dilin kelimelerinin tam olarak sınırlandırılmamış uygulama alanına sahip olmadığı gerçeğinden kaynaklanmaktadır. Bu tür kavramların kesinlikle uygulanabilir olduğu ve kesinlikle uygulanamaz olduğu durumlar var olmasına rağmen onların uygulamasının açık olmadığı ve bu iki net durum arasında bir *yar*ı *gölgeli alan (penumbra)* vardır. Örneğin 35 derece kesinlikle sıcak havadır ve 10 derece kesinlikle sıcak değildir. Fakat, peki ya 25 derece nedir? Açıkçası, "sıcak" teriminin uygulanabilirliği konusunda kesin bir ayrım çizgisi yoktur. Bu nedenle verilen karar biraz rastgele nitelik taşır.

Bulanık set teorisi ve bulanık mantık altında yatan fikir "sıcak" gibi bir dilsel terimin uygulanabilirlik alanını kesin bir çizgi ile değil bulanık yumuşak sınırları olan bulanık bir set ile tanımlamaktır. Bu yumuşak sınır bir sete 0 ile 1 arasında aşamalı üyeliğe izin verilerek oluşturulur. Sıfır bir elemanın o sete dahil olmadığı anlamına gelir ve 1 ise hiç bir kısıtlama olmadan dahil olduğu anlamına gelir. Böyle bir aşamalı üyelik ile, serin, ılık veya sıcak gibi dilsel terimler Şekil 1'de görüldüğü gibi yorumlanabilir. Fakat, 0 ile 10 arasındaki dereceler kesinlikle serin ve kesinlikle ne ılık ne de sıcaktır. Ancak 10 ile 20 arasında sıcaklıklar ılık ya da serin olarak nitelendirilebilir ama iki sete üyeliklerinin dereceleri farklıdır. Sıcaklık ne kadar düşük olursa, serin setine üyelik derecesi o kadar yüksek olacak ve sıcaklık ne kadar yüksek ise ılığa üyelik derecesi o kadar yüksek olacaktır. Tabiki, üyelik işlevlerinin kesin şekli uygulamaya bağlıdır.

Bulanık mantığın ve bulanık set teorisinin temel avantajı bilgimizi, üyelik işlevleriyle çok sezisel olarak yorumlanan dilsel ifadeleri kullanan kurallarla ifade etmemize müsade etmesidir. Aynı zamanda, eğer birden fazla kural uygulanabilir ise bu tür kurallar arasında ara değerlendirme yapmamız için çok uygun imkanlar sağlar. İşte bu nedenle bulanık mantık bilgi tabanlı kontrolde çok popülerdir. Bu mantık, dil kuralları yoluyla uzman insan kilit girdi/çıktı ilişkilerini belirleyerek bir kontrolör oluşturulmasına izin verir. Çıkarım motoru bu dil kuralları arasında ara değerlendirme yaparak kontrol işlevini tamamlar. Bulanık mantık kontrolörlerinin büyük ticari başarısı olmuştur ve bugün birçok ev eşyasında bulunabilir. Bulanık

mantığının ve uygulamalarının daha kapsamlı bir şekilde anlatımı 8-10. bölümlerde bulunabilir.



Şekil 1. Bulanık kümede dilsel terim olarak serin, ılık ve sıcak.

Bilgi Edinimi

Ne yazik ki yapay zeka sistemleri için ihtiyaç duyulan çoğu bilgi insanların zihninde "gömülü"dür ve bu bilgiyi bilgisayara verecek şekilde ortaya çıkartmanın oldukça zor olduğu görülmüştür. Örneğin, tarım uzmanı tarlaya baktıktan, toprağı inceledikten ve bölgedeki iklimi öğrendikten sonra bu alanın hangi ürün için uygun olduğunu söyleyebilir. Fakat, net ve basit kurallar olarak ifade etmek bir yana tam olarak mantığını açıklayamayabilir. Bir başka deyişle, bir şeyi bilmek ile öğretmek aynı şey değildir ve eğer bilgisayarların bize yardımcı olmalarını istiyorsak, onlara belirli şeyleri nasıl yapacaklarını öğretmek zorundayız.

Sonuç olarak, bilgi edinimi yapay zekanın önemli bir alanıdır. Temel olarak, süreçleri otomatikleştirmek ve bilgisayar tabanlı karar desteği sağlamak için gerekli bilginin alanında uzman olan kişiden nasıl alınacağı sorunuyla ilgilenir. Bu nedenle, bilgisayar biliminden çok bazen psikolojinin bir parçasıdır. Çünkü uzman insanlarla görüşmeler, anketlerin oluşturulmasını, alana özgü terimlerin haritalandırılmasını ve bilgisayarda işlenebilir kavramları ve yüklemleri içerir. Bilgi edinimi çok yorucu ve zaman alıcı süreç olabileceği için, hesaplama gücünde, depolama ortamı ve sensör teknolojisinde önemli gelişmelerle desteklenen son araştırmalar, veriden veya uzman insanın yaptığını gözlemleyerek (söyleyebileceğimiz bir şekilde) öğrenmeye ve onu taklit etmek için gerekli kuralları otomatik olarak bulmaya odaklanmıştır.

Vaka Tabanlı Mantık Yürütme

Bilgi edinim dar boğazını geçmenin popüler bir yolu da *vaka tabanlı mantık yürütme*dir. Bu yaklaşım sıklıkla insanoğlunun, bir durumu geçmişte yaşadığı durumlarla karşılaştırarak değerlendirdikleri ve karşılaştıkları veya duydukları çoğu benzer durumda, davrandıklarına benzer bir yolla davrandıkları fikrini esas almaktadır. Bunun sonucunda vaka tabanlı bir mantık yürütme sisteminde, bilgi kurallar halinde kodlanmak yerine geçmiş olaylar kütüphanesinde bulundurulur. Kütüphanedeki her bir başlık sonuçları ve çözümleriyle birlikte bir durum ya da problemi tanımlar. Bu nedenle, sonuç çıkarma ve çözüm bulmadaki mantık ya da bilgi açıkça verilmez, fakat tanımlar halinde gizli bir şekilde bırakılır. Bu nedenle

buna, sadece örnekler vakalar halinde kütüphanede tutulduğu için *tembel öğrenme* de denir.

Vaka tabanlı sistemlerde mantık yürütme sadece mevcut duruma en uygun olan ilgili vakaları kütüphaneden almaktan ibarettir. Daha sonra bu durumların sonuçları veya bunların çözümleri yeni duruma uygulanabileceği varsayılır. Daha sonra çözümlerin deneme uygulamalarıyla, mevcut duruma uymaları için gözden geçirilip uyarlanmaları gerekip gerekmediğine karar verilir. Tabiî ki, mevcut durumun sonucu ya da sorunun çözümü bulunduktan sonra kütüphaneye yeni bir durum olarak eklenir. Vaka tabanlı mantık yürütmenin özel ve basit bir şekli de en yakın k-komşu sınıflandırmasıdır. Vakaların benzerliklerine işaret eden mesafe ölçümüne bağlı olarak, kütüphanedeki en yakın k örnekleri alınır ve yeni vakanın sınıfı bu vakalardan alınan oy çokluğuna göre tahmin edilir.

Vaka tabanlı mantık yürütmenin avantajı insanın mantık yürütmesiyle benzer bir şekilde sezgisel olarak çalışmasıdır. Bu nedenle bu sistemin yaptığı öneriler karar verici insanlar tarafından genellikle hemen kabul edilir. Buna ek olarak bilgi edinimi oldukça basitleşir. Bu sistemin dezavantajları da örnek vakalar kütüphanesinin çok geniş olmasıdır ki bu da etkin arama metotlarını geliştirmeyi ve benzer vakaların bulunmasını zorlaştırır. Dahası bu tür metotların başarısı seçilen benzerlik ölçümünün kalitesine bağlıdır. Vaka tabanlı mantık yürütme ve onun uygulamaları hakkında daha fazla bilgi 11 ve 12. bölümlerde bulunabilir.

3.4.2 Planlama

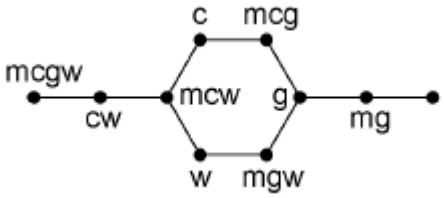
Düşünmenin en basit seviyedeki uygun tanımı, hayali bir ortamda deneme uygulaması olmasıdır. Bu tür deneme uygulaması, eylemlerin olası sonuçlarını keşfederek gerçek dünya da bir plan ortaya çıkartma amacına hizmet eder. Bunun avantajı başarısız olan eylemlerin zararlı sonuçlarının olmamasıdır. Ya da bir başka değişle Konrad Lorenz'in düşünmenin devrimsel avantajı hakkında belirttiği şekilde insan yerine hipotez ölür.

Durum Grafikleri

Bir planlama problemi bir durum grafiği olarak en uygun gösterilir. Bu durum grafiğinde her bir düğüm ilgili alanın durumunu temsil eder ve her biri belli bir eylem sonucu durumlar arasında olası direk geçişi belirtir. Bazı durumlar, alanın başlangıçta olduğu başlangıç durumu ve erişilmesi gereken hedef durumları gibi özel bir rol oynar. Burada amaç, *plan* olarak adlandırılan bir eylem dizimi bulmak ve böylece bu eylem diziminin yol açtığı durum değişiklikleriyle ilk durumdan hedef durumlardan birine gitmektir. Durum grafiğinde bu plan yola karşılık gelir, yani ilk durumu temsil eden düğümden hedef durumu temsil eden düğüme kenarların bir (edge) dizimidir.

Bu konuyla ilgili, bir lahana bir keçi ve bir kurt taşıyan adamın bilindik hikayesini örnek olarak verebiliriz. Nehrin karşısına geçmek zorundadır fakat ne

yazık ki botu sadece kendisini ve üçünden birini taşıyabilmektedir. Ne lahana ile keçiyi ne de keçi ile kurdu bırakamaz. Çünkü her iki durumda da ikincisi birincisini yiyecektir. Amaç onu ve yanındaki nesneleri nehrin karşı tarafına güvenle geçirecek bir plan bulmaktır. Bu problem için durum grafiği Şekil 2 de gösterilmiştir. Sol tarafta başlangıç durumu, sağ tarafta ise hedef durumu bulunmaktadır. Durumların yanında yazılı olan harfler hangi öğelerin nehrin bu tarafında olduğunu gösterir (M = Adam, C= Lahana, G=Keçi, W=Kurt). Açıkçası bu sunum ile başlangıç durumdan elit duruma bir yol arayarak çözüm kolayca bulunabilir.



Şekil 2. Kabak, keçi kurt problemi durumu. Başlangıç durumu solda amaç durumu ise sağdadır.

Elbette bu örnek bir çocuk oyuncağıdır. Gerçek dünya problemlerine yönelik hazırlanan durum grafikleri yüzlerce hatta binlerce durum içerirler ve çok daha büyük olurlar. Sonuç olarak bu örnekteki gibi bütün durum grafiğini oluşturmak bazen imkânsız olabilir. Böyle bir durumda durum grafiği örtülü temsiller kullanılarak gösterilir. Bu grafik başlangıç durumu belli bir seviyeden doğrudan ulaşılabilecek durumları oluşturmaya yönelik işlemler ve hedef durumları belirleyecek bir fonksiyondan oluşur.

Durum Grafiklerinde Arama

Durum grafikleri sadece planlama problemlerinin temsilleridir. Derhal çözüm sağlamazlar. Çünkü bizim hala başlangıç durumundan hedef duruma bir yol aramamız gerekmektedir. Bu arayış esnasında her bir eylemin maliyeti olacağı ve bizimde en düşük maliyetli çözümü istiyor olabileceğimiz gibi ek sınırlılıkları hesaba katmak zorunda olabiliriz. Patika arama problemlerine doğrudan çözümler *genişlik öncelikli arama* veya daha genel versiyonuyla *tek düze maliyet arama* ve *derinlik öncelikli arama*lardır. Genişlik öncelikli ve tek düze maliyetli aramalarda, artan mesafe (kenar sayısına göre ölçülür) veya artan maliyete göre başlangıç durumundan başlayarak durumlar ziyaret edilir. Derinlik öncelikli aramalarda ise her zaman en son ziyaret edilmiş durumdan başlanılır ve kenar sayısıyla belirlenen bir derinlik limitine ulaşıncaya kadar devam edilir. Bu limitte veya hiçbir yeni duruma ulaşılamaz ise arama yeni durumlara ulaşılabilecek şekilde daha önceden ziyaret edilmiş duruma doğru geriye gider. Genişlik öncelikli (tek düze maliyetli) aramanın avantajı her zaman için en az sayıda eylem gerektiren çözümü bulmasıdır (bu da en az maliyet demektir). Fakat bu aramanın dezavantajı ise birçok durumun paralel olarak işlenmesini gerektirmesidir ki bu şekilde işlem yapan bir program çok fazla

bilgisayar hafızası kullanır. Derinlik öncelikli araştırmanın avantajı hafıza açısından daha ekonomik olmasıdır. Fakat hedef duruma ulaşılır ulaşılmaz arama durdurulur ise optimum sonucu bulamayabilir. Buna ek olarak, eğer bütün hedef durumlar derinlik limiti ile belirlenen "arama ufku" 'nun ötesinde ise çözüm bulamayabilir.

Bu yaklaşımların hiçbiri aramaya rehberlik etmek için uygulama alanı hakkında özel bilgi kullanmadığından, bütün bu yaklaşımlar tek düze olmayan araştırma olarak adlandırılır. Fakat uygulamada aramayı yönlendirmek için alana özel bilgi kullanmak genellikle kaçınılmazdır. Çünkü aksi takdirde arama yapılamayacak kadar uzun sürer. Örneğin yol ağında her bir kasabayı veya hatta her bir kavşağı bir durum olarak ele alıp derinlik öncelikli arama ile A şehrinden B şehrine en kısa rotayı arıyor olsaydık A ve B şehirleri birbirine çok yakın değilseler arama başarısız olurdu. Bunun yerine aramayı sadece A şehrinden B şehri üzerine giden yolları veya en yakın anayola çıkan yolları takip ederek başlatmak için faydalı kestirme yolları kullanırız. Yani aramayı en muhtemel alternatiflerle sınırlandırır ve diğer alternatifleri sadece başarısızlık olursa düşünürüz.

Bu tür kestirme yolları kullanan genel bir metot ise A* algoritmasıdır. Bu algoritma durumları değerlendiren *g* ve *h* olmak üzere iki fonksiyona dayalıdır. *g* fonksiyonu başlangıç durumundan bir duruma varmanın maliyetini hesaplar. Bu maliyetler genellikle o ana kadar bulunmuş olan en iyi yol üzerinden hesaplanır. *h* fonksiyonu ise bir durumdan hedef duruma ulaşmak için gerekli maliyetleri hesaplar ve probleme has sezgisel işlev ile hesaplanır. Yukarıda ele aldığımıza benzer bir ulaşım probleminde, biz sadece B şehri ile bir kasaba veya kavşak arasındaki düz mesafeyi kullanabiliriz. A* algoritması aramaya iki işlevin toplam hesaplamalarının en düşük olduğu durumdan devam etmeyi önerir. Bunun avantajı, sezgisel hesaplama uygun olduğu sürece hedef duruma giden yol nispeten daha az adımda bulunabilmesidir. A* algoritmasının güzel bir özelliği de *h* fonksiyonu maliyeti fazladan hesaplamadıkça optimum (yani minumum maliyetli) çözümün bulunması garantidir.

Genel olarak aramayla ve özellikle de sezgisel arama teknikleriyle ilgili daha fazla bilgi 13 ve 14. bölümlerde bulunabilir.

Durumlar ve Eylemlerle İlgili Mantık Yürütme

Planlamaya yönelik daha karmaşık yaklaşımlar durumların mantık tabanlı sunumlarını ve eylemlerin durumları ile onların özelliklerini nasıl değiştirdiğine dair bilgiyi kullanır. Bu durumda, bir eylemin sonuçları daha genel olarak tanımlanabilir ve bir eylemin belli bir durumdaki sonuçlarını belirlemek planlama sisteminin altında yatan mantıksal çıkarım motoruna bırakılır. Tabi ki bu yaklaşımın çok yüksek esneklik gibi bir avantajı vardır. Fakat bu aynı zamanda yüksek hesaplama maliyetlerine yol açar ki bu maliyetler ise bu yaklaşımı bazı durumlar için uygulanamaz hale getirir.

3.4.3 Öğrenme

Bütün yüksek seviyedeki doğal organizmaların en çarpıcı özelliklerinden biri *öğrenmeleri* yani geçmişteki tecrübelerine göre davranışlarını değiştirmeleridir. Öğrenme bir (doğal veya yapay) ajanın aynı görevleri daha etkin bir şekilde yapmasını ve daha önce beceremediği görevleri yapabilir hale gelmesini sağlar. Bu nedenle öğrenme becerisi genellikle zeki davranışın köşetaşı olarak görülür. Bu bölümde otomatik veya makine öğrenmesine yönelik birkaç yaklaşımı ele alacağız. Daha detaylı incelemeler 15 ve 16. bölümde bulunabilir.

Endüktif Mantık Programlama

Endüktif mantık programlama veriye uygulanabilen bütün mantıksal formüller alanını sistematik olarak araştırarak belli bir veriyi uygun bir şekilde tanımlayan mantıksal formül bulmaktan oluşur. Genellikle arama kurallarla sınırlıdır, (örneğin eğer o zaman ifadeleri "if-then statements") çünkü bu tür kurallar çoğu uygulamada en uygun kurallardır.

Mantıksal formül alanında arama genellikle genelden özele doğru giderek yapılır (fakat bunun tersini takip eden birkaç yaklaşımda vardır). Yani araştırma genellikle bütün örnek durumlara uygulanabilen bir kuralla, çok geniş bir formül ile başlatılır. Daha sonra bu kuralın destek sayısını, yani kuralın uygulanabildiği örnek durum sayısını aşırı derecede azaltmadan daha özel hale getirir. Daha sonra her bir kuralın kalitesi bulunan kuralların sunumuna dair bir sıralama veren fakat aynı zamanda aramaya rehberlik içinde kullanılabilen sezgisel ölçümlerle değerlendirilir. Eğer en yüksek skoru alan kurallar ilk olarak genişletilir ise arama sürecinin başında en iyi ve en faydalı kuralları bulma şansı daha yüksektir. Bunun avantajı bütün arama alanını gezmeden ve genel sonuçtan çok fazla kayıp vermeden arama erkenden durdurulabilir.

Endüktif mantık programlaması sınıflandırma ve kavram tanımlama işlemi için çok uygundur. Bu durumlarda öncesi boş olan ve sonrasında ise sınıf veya kavram olan her bir sınıf veya kavram için bir kuralla başlar. Önceki kuralı daha özel yapmak için şartlar sadece bu kurala eklenmektedir. Aramada bulunan en iyi kurallar henüz sınıfı bilinmeyen durumları sınıflandırmak veya kavramın özelliklerini anlamak için kullanılabilir. Örnek uygulamalar bir bankanın kredi vermeye değer müşterilerinin karakteristik özelliklerini değerlendirmek (bu müşterileri muhtemelen krediyi geri ödemeyecek olanlardan ayırmak için) veya bir araba üreticisinin hataya yatkın araçlarını belirlemek için (nedenlerini bulmak ve ürün kalitesini arttırmak için) değerlendirmelerde bulunması gibi görevlerdir.

Endüktif mantık programlamasının avantajı eğitim verisinin tek bir tabloda sunulmasıyla sınırlı olmaması fakat referanslar veya anahtarlar yoluyla birbirine bağlanan çoklu ilişkileri de ele alabiliyor olmasıdır. Bu, verinin genellikle ilişkisel veritabanı sistemlerinde saklandığı ve birbirine atıfta bulunan birkaç ilişkiye bölündüğü durumlarda çok önemli olabilir. Böyle bir durumda veriyi tablonun her

bir satırının bir örnek vakayı tanımladığı tabular şekline dönüştürmek çok zor hatta imkansız olabilir. Aşağıda tanımlanan karar ağaçları ve yapay sinir ağları bu tür tekli tablo sunumları gerektirir.

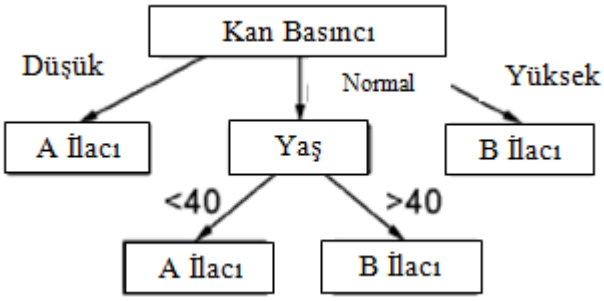
Endüktif mantık programlamasının temel dezavantajı, uygulamalarının genellikle yavaş olmasıdır. Bu kısmen, bu programlamaların hızlı uygulamaya uygun olmayan bir dil olan PROLOG ile programlanmış olmalarından kaynaklanır. Fakat problemin özü daha çok endüktif mantık programlaması yaklaşımlarında tarama alanı genellikle çok geniş olması ve bu nedenle çok uzun işlem uygulama süreleri ortaya çıkmasından kaynaklanır. Sonuç olarak, işlemin iptal edilmesine neden olacak kadar uzun işlem süreleri ortaya çıkabilir. Bu yüzden, arama alanını kısıtlamak ve formal gramerler veya kural şablonlarıyla aranacak kuralların türlerini belirleyen bir *bildirimsel önyargı* ve her bir adımda, bir sonraki adımda bulunan en iyi k kurallar üzerinde odaklanan ışın araştırması gibi sezgisel araştırmalar uygulayarak araştırmaya rehberlik sağlamak çok önemlidir.

Endüktif mantık programlama yaklaşımları ile ilgili daha fazla bilgi 17 ve 18. bölümlerde bulunabilir.

Karar Ağaçları

Karar ağacı en popüler makine öğrenme yöntemidir. Çünkü çok basit ve hızlıdır ve kolay anlaşılabilir sonuçlar verebilir. Karar ağacı bir sınıflandırıcıdır. Yani söz konusu bir vaka ya da nesneyi önceden tanımlanmış bir dizi arasından bir sınıfa koyma yöntemidir. Bu yöntem bir dizi tanımlayıcı özelliklerin değerlerine dayalıdır.

Adından da anlaşılabileceği gibi karar ağacı ağaç şeklindedir. Her bir iç düğüm (yani kendisini takip eden bir düğümü olan) tanımlayıcı bir özelliğin testini belirler, her bir yaprak (yani kendinden sonra gelen olmayan düğüm) bir sınıfa atama yapar. Tahmin edilen sınıfın okunabildiği yaprağa ulaşıncaya kadar, karar ağacı ile bir vaka ya da nesne kökten başlayıp dallar boyunca aşağı inerek sınıflandırılır. Bu dallar iç düğüm tarafından belirlenen testlerin sonuçlarına karşılık gelir. Buna bir örnek olarak, Şekil 3'de bir ilaç konusunda verilecek karar ile ilgili tıbbi bir iş için hazırlanmış çok basit bir karar ağacı verilmiştir. Ağaç öncelikle hastanın kan basıncını ölçmeyi önerir. Eğer kan basıncı düşük ise, A ilacı kullanılmalı, eğer yüksekse B ilacı kullanılmalıdır. Eğer kan basıncı normal ise, ikinci bir test gerekir. Hastanın yaşının kırkın üzerinde olup olmamasına bağlı olarak A ilacını veya B ilacını kullanmayı önerir.



Şekil 3. Hastanın kan basıncı ve yaşına bağlı olarak yöneticiye sahada ilaç öneren basit bir karar ağacı

Karar ağaçları bir dizi önceden sınıflanmış örnek vakalardan "böl ve fethet" yaklaşımı ile birlikte test özelliklerinin "aç gözlü" seçimine dayalı olan bir rekürsif (yinelemeli) prosedürle otomatik olarak oluşturulabilir. Bu yaklaşımla tepeden aşağıya bir karar ağacı oluşturulur (Şekil 3'te temsili görülmektedir). İşte bu yüzden bu yaklaşım *karar ağaçlarının tepeden aşağı endüksiyonu* olarak adlandırılır.

Şu şekilde çalışır: Sınıf ile ilgili mevcut tanımlayıcı özelliklerden her birinin sağladığı bilgi miktarı bir değerlendirme ölçümü ile hesaplanır. En yüksek notu olan özellik (ya da daha doğrusu bu özelliğin spesifik testi) seçilir ve ağacın kökünü oluşturur. Daha sonra eğitim durumları seçilen testin sonucuna göre alt dizinlere ayrılır. Ağacın geri kalanı aynı prosedürün her bir alt dizine uygulanarak rekürsif olarak oluşturulur. Eğer bir alt dizindeki bütün durumlar aynı sınıfa atanırsa, bir başka ayrım yapmak için çok az bulunması durumunda veya yapılacak başka bir test kalmamış ise yenileme durur.

Bir karar ağacının bir dizi kuralın öz bir sunumu olarak da görülebileceği de bilinmelidir. Kuralın sonucunu veren yapraktaki sınıf atamasını ve öncekini veren yol boyuncaki test sonuçlarının birleştirilmesi neticesinde kökten yaprak düğümüne giden her bir yol, bir kuralı tanımlamaktadır. Bu yüzden karar ağaçlarının endüksiyonu kurallar bulmak için de kullanılabilir. Fakat oluşturulma şekillerinden dolayı sonucun bir alan için en açıklayıcı kuralları vermeyebileceği de unutulmamalıdır. Amaç daha ziyade hep birlikte bütün eğitim örneklerinde iyi bir sınıflandırma sağlayan bir dizi (ilgili) kurallar bulmaktır.

Karar ağaçlarının avantajı öğrenmenin genellikle çok hızlı olması ve sonucun uzman insanlar tarafından kolayca yorumlanabilir olması ve böylece makullük açısından kontrol edilebilir olmasıdır. Karar ağaçlarının dezavantajları endüksiyon sürecinin mevcut özellikler arasından sadece çok azını seçmesidir. Bu nedenle çok fazla sayıda özelliğe dağılmış olan bilginin (her bir özellik sınıf hakkında sadece sınırlı bir bilgi taşır) etkin bir şekilde ele alınamamasıdır ki bu da optimum tahmin doğruluğunun altına düşülmesine yol açar. Bu tür durumlarda Bayes sınıflandırıcılar ve yapay sinir ağları genellikle daha üstündür.

Karar ağaçlarının yapısının altında yatan algoritmalar ve matematikle ilgili detaylı anlatım 19 ve 20. kaynaklarda bulunabilir. Yirminci kaynak aynı zamanda sınıfların yerine sayısal değerleri tahmin edebilen regresyon ağaçlarını da ele almaktadır.

Yapay Sinir Ağları

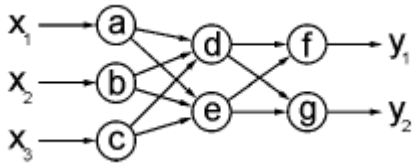
Yapay sinir ağları insanların ve hayvanların sinir sistemine "özellikle de beyine" göre yapıları ve fonksiyonellikleri oluşturulmuş bilgi işleme sistemleridir. Bu ağların altında yatan temel fikir, zeki davranış ve özellikle de öğrenme yeteneğinin, doğal zeki sistemlerin "biyolojik donanımının" belli özelliklerini taklit ederek gerçekleştirilebileceğinden kaynaklanmaktadır. Yapay sinir ağları paralel olarak çalışan ve nöron denilen (genellikle) çok sayıda basit işlemci birimlerden oluşur. Bu nöronlar bilgiyi, birbirlerine doğrudan bağlantılar yoluyla etkinleştirme sinyalleri göndererek işlerler.

Çeşitli işler için kullanılabilecek çok değişik türde yapay sinir ağları vardır. Bu makalede kendimizi, sınıflandırma ve tahmin amaçları için kullanılabilen ve en yaygın tip olan *çok katmanlı perseptron (algılayıcı)la* sınırlandırıyoruz. Çok katmanlı perseptronun nöronları temel olarak eşik birimlerdir: Eğer girdi toplamanın ağırlığı bir nörona has eşiği geçerse bir nöron aktif hale gelir ve bağlı olduğu nöronlara sinyal gönderir. Aksi halde pasif kalır. Fakat eğitim amaçları için eşik davranışı sabit değildir, bir sigmoit (S şeklinde) fonksiyon ile tanımlanır. Bu fonksiyon eşikte 0,5'tir ve eşiğin altındaki değerlerde sıfıra ve eşikten büyük değerler için bire yaklaşır. Böyle bir "yumuşatılmış" eşik fonksiyonunun avantajı onun değiştirilebilir olmasıdır ki bu da eğitim için önemlidir.

İsminden de anlaşıldığı gibi çok katmanlı perseptronun nöronları katmanlar halinde organize edilmiştir. Bağlantılar sadece birbirini takip eden katmanlar arasındadır ve aktarılan aktivasyon sinyalleri ile çarpılan ağrılıkları taşırlar. Dahası çok katmanlı perseptron ileri beslemeli bir ağdır. Yani bağlantılar sadece bir yöne doğru gider. Bir başka deyişle geriye doğru bağlantı yoktur. En yaygın şekliyle üç katman kullanılır: girdi katmanı, bir gizli katman ve bir çıktı katmanı (Şekil 4'e bakılabilir). Girdi katmanı girdi değerlerini alır ve onları genellikle değiştirmeden gizli katmandaki nöronlara dağıtır. Gizli katmandaki nöronlar (bunlara gizli denmesinin sebebi çevre ile etkileşim halinde olmamaları yani doğrudan girdi almamaları veya çıktı üretmemelerinden dolayıdır) yukarıda tarif edildiği gibi eşik şeklinde girdi nöronlarından gelen sinyalleri işlerler ve sonuçları çıktı katmanının nöronlarına gönderirler. Daha sonra çıktı katmanının nöronları gizli katmandan gelen sinyalleri yine yukarıda tarif edilen eşik şeklinde işlerler ve böylece ağın çıktısını üretirler.

Her ne kadar nöronların tek başlarına yaptıkları hesaplamalar çok basit olsa da düzgün bir şekilde kurulmuş olan çok katmanlı perseptronlar çok karmaşık fonksiyonları hesaplayabilir. Aslında herhangi bir Riemann integrallenebilir

fonksiyonuna rastgele doğrulukla yaklaşmak (her ne kadar bu durum çok fazla sayıda nöron gerektirse de) için en fazla iki gizli katman gerektiğini kanıtlamak oldukça basittir.



Şekil 4. Üç girişli üç çıkışlı yedi nöronlu basit bir üç katmanlı perseptron

Çok katmanlı perseptronlar *hata geri yayılımı* denilen bir metot ile, sadece girdi çıktı şeklinde verilen bir fonksiyonu tahmin edecek şekilde eğitilebilir. Bu metodun altında yatan fikir geniş anlamda şöyledir: Ağın parametreleri (bağlantı ağırlıkları ve eşik değerleri) rastgele değerler olarak başlatılır. Daha sonra ağ ile girdi yapıları işlenir ve ortaya çıkan sonuç istenilen ile karşılaştırılır. Genellikle istenilen ile ortaya çıkan sonuçlar arasındaki farkların karesi alınmış toplamından oluşan hata, ağ boyunca geriye doğru ilerletilir ve hata daha küçük olacak şekilde parametreler ayarlanır. Bu işlem hata yeterince küçük oluncaya veya artık değişmez oluncaya kadar tekrarlanır. Matematiksel olarak geri yayılım işlemi hata fonksiyonu üzerinde bir *eğim düşümü (gradient descent)*'dür. Yani hatanın küçüldüğü yönde ufak adımlar tekrarlı olarak atılır bu yön hata fonksiyonunu değiştirerek belirlenir.

Yapay sinir ağlarının avantajı onların sıklıkla diğer metotlarla karşılaştırıldığında genellikle w.r.t doğruluğunda en iyi sonuçları üretmeleridir. Bu ağların dezavantajı ise eğitilen ağ bir "kara kutu" dur. Ağın "bilgisi" bağlantı ağırlıklarında ve eşik değerlerinde depolandığı için yaptığı hesaplamalar neredeyse yorumlanamazdır.

Yapay sinir ağlarının metotları ve özellikle de diğer ağ tipleri ve uygun oldukları işlerle ilgili daha detaylı açıklamalar [21] ve [22]. kaynaklarda bulunabilir. Bu ağların ilginç bir türü ise yapay sinir ağlarının öğrenme becerisi ile bulanık sistemlerin kapsamlılığını birleştiren nörobulanık sistemlerdir [23].

Genetik Algoritmalar

Genetik algoritmalarla birisi biyolojik evrimin optimizasyon işlemini taklit etmeye çalışır. Kaliteye bağlı üremenin yanında rastgele mutasyonlar ve aday çözümlerin tekrarlı kombinasyonları (birleştirmeyle ilgili) optimizasyon problemlerine optimal veya optimuma yakın çözümler bulmayı amaçlar. Çoğu öğrenme görevleri (birleştirmeyle ilgili) optimizasyon problemleri olarak yeniden formülize edilebildikleri için genetik algoritmalar neredeyse evrensel olarak uygulanabilirler.

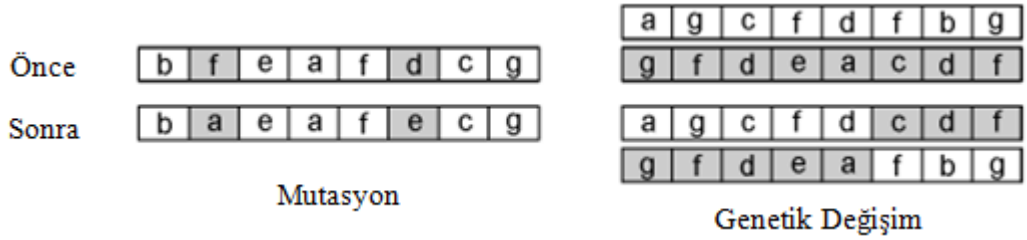
Genelde bir genetik algoritma yaklaşımı aşağıdaki adımlardan oluşur:

1. Olası çözümleri kodlamak için uygun bir yol bulmak. Olası bir çözümün kodlanmasına *kromozom* denir;
2. Olası bir çözümün kalitesini değerlendirmek için bir *uygunluk fonksiyonu* tanımlamak;
3. Rastgele olası çözümlerin ilk popülasyonunu (kodlamaları) oluşturmak;
4. Popülasyonun olası çözümlerini değerlendirmek;
5. Kalitelerine göre olası çözümlerin gelecek neslini seçmek. Olası bir çözümün uygunluğu ne kadar yüksekse çocuk (bir sonraki nesilde kendisinin bir kopyası) sahibi olma olasılığı da o kadar yüksektir. Aynı aday çözüm birçok kez seçilebilir;
6. *Mutasyon* (bir kromozomun küçük bir kısmını rastgele değiştirmek) ve *çaprazlama* (2 anne kromozomu parçalarını rastgele iyileştirmek) gibi *genetik operasyon*lar uygulayarak seçilen olası çözümü temsil eden kromozomları değiştirmek;
7. En iyi bireyin minumum kalite vermesi, belli bir sayıda adım atılması sonucu hiçbir ilerleme olmaması veya önceden belirlenmiş sayıda nesillerin oluşturulması gibi bir sonlandırma kriterine ulaşıncaya kadar 4 ile 6 arasındaki adımları tekrarlayın;
8. Son neslin en iyi olası çözümü, bulunan çözümdür (veya eğer kayıt edilirse tüm süreç boyunca karşılaşılan en iyi çözüm olarak değerlendirilebilir).

Olası çözümlerin hangi kodlamasını en uygun olacağı probleme bağlıdır. Genellikle biyolojik kromozomlardaki bilginin düğümsel organizasyonuna benzer şekilde karakter dizileri kullanılmaktadır. Dizim içerisinde her bir pozisyon bir gene karşılık gelir ve belli bir genin konumunda bulunabilecek her bir karakter ise genin *eş gen (alel)'*ine karşılık gelir. Öyle bir kodlamanın pratikteki avantajı dizim sunumları için birçok standart genetik operatörün kullanılabilir olmasıdır. Örneğin mutasyon, dizinde rastgele birkaç pozisyonu seçmek ve bu pozisyonlardaki özellikleri değiştirmekten oluşabilir (Şekil 5'in sol tarafına bakılabilir). Çaprazlama, rastgele kesim noktası seçmek ve kesim noktasının bir ucunda dizinlerin parçalarında yer değişikliği yapmak olarak tanımlanabilir (buna bir nokta çaprazlama denir, Şekil 5'in sağ tarafına bakılabilir). Uygunluğa bağlı seçim yapmanın basit bir yöntemi ise *çarkıfelek seçimi*dir. Çarkıfelek her bir dilimi mevcut popülasyon içinde olası bir çözümle ilişkilendirilmiş olan bir çarktır. Dilimin büyüklüğü, olası çözümün uygunluğunu yansıtır. Böylece çarkıfeleğin her bir dönüşü bir sonraki nesil için olası bir çözümü seçer. Daha iyi olası çözümler, daha büyük dilimlerle ilişkilendirildikleri için seçilme şansları daha yüksektir.

Daha karmaşık genetik algoritmalar seçkincilik içeren (bu algoritmada her zaman en iyi olası çözüm bir sonraki nesile değiştirilmeden aktarılır) ve kalabalıktan kaçınma teknikleri (bir popülasyondan daha az düşük çeşitlilikte olası çözümlerin dahil edilmesi) vardır. Bu durum sınırlı sayıda genetik malzeme kaldığı için iyileştirmeleri engelleyebilir.

Genetik algoritmaların en büyük avantajları onların modelden bağımsız (yani ele alınan alanla ilgili spesifik bir model varsayımında bulunmazlar) olmaları ve uygulanabilirliği konusunda neredeyse hiç sınır olmamasıdır. Dezavantajları ise genellikle çözüme ulaşmanın uzun sürmesi ve başarılarının büyük oranda seçilen problem kodlamasına bağlı olmasıdır. Eğer kodlama uygun değilse genetik algoritma tamamen başarısız bile olabilir. Bu durum onların modelden bağımsız olma avantajını azaltır. Çünkü iyi bir kodlama bulmak çok çaba gerektirir ve alana özel model oluşturmak kadar maliyetli olabilir. Benzetilmiş tavlama, genetik programlama ve evrim stratejileri gibi genetik algoritmaların ve onların türlerinin geniş kapsamlı muameleleri [24] ve [26]. kaynaklarda ele alınmıştır.



Şekil 5. Genetik işletmeciler mutasyon ve genetik değişim

Kaynaklar

1. Russell, S., and P. Norvig. 2003. Artificial Intelligence: A Modern Approach, 2nd ed. Upper Saddle River, NJ: Prentice Hall.
2. Nilsson, N. J. 1998. Artificial Intelligence: A New Synthesis. San Francisco, CA: Morgan Kaufmann.
3. Genesereth, M. R., and N. J. Nilsson. 1987. Logical Foundations of Artificial Intelligence. San Mateo, CA: Morgan Kaufmann.
4. Levesque, H. J., and G. Lakemeyer. 2001. The Logic of Knowledge Bases. Cambridge, MA: MIT Press.
5. Reiter, R. 2001. Knowledge in Action. Cambridge, MA: MIT Press.
6. Jensen, F. V. 2001. Bayesian Networks and Decision Graphs. New York, NY: Springer-Verlag.
7. Borgelt, C., and R. Kruse. 2002. Graphical Models: Methods for Data Analysis and Mining. Chichester, UK: J. Wiley and Sons.
8. Kruse, R., J. Gebhardt, and F. Klawonn. 1994. Foundations of Fuzzy Systems. Chichester, UK: J. Wiley and Sons.
9. Klir, G. J., and B. Yoan. 1995. Fuzzy Sets and Fuzzy Logic: Theory and Applications. Upper Saddle River, NJ: Prentice-Hall.
10. Zimmermann, H.-J. 1996. Fuzzy Set Theory and Its Applications. Dordrecht, Netherlands: Kluwer.
11. Kolodner, J. 1993. Case-Based Reasoning. San Mateo, CA: Morgan Kaufmann.
12. Watson, I. 1997. Applying Case-Based Reasoning: Techniques for Enterprise Systems. San Francisco, CA: Morgan Kaufmann.
13. Pearl, J. 1984. Heuristics: Intelligent Search Strategies for Computer Problem Solving. Reading, MA: Addison-Wesley.
14. Rayward-Smith, V. J., I. H. Osman, and C. R. Reeves, eds. 1996. Modern Heuristic Search Methods. New York, NY: J. Wiley and Sons.
15. Langley, P. 1995. Elements of Machine Learning. San Mateo, CA: Morgan Kaufmann.

16. Mitchell, T. 1997. Machine Learning. New York, NY: McGraw-Hill.
17. Lavrac, N., and S. Dzeroski. 1994. Inductive Logic Programming: Techniques and Applications. Upper Saddle River, NJ: Prentice Hall.
18. Bergadano, F., and D. Gunetti. 1995. Inductive Logic Programming. Cambridge, MA: MIT Press.
19. Quinlan, J. R. 1993. C4.5: Programs for Machine Learning. San Mateo, CA: Morgan Kaufmann.
20. Breiman, L., J. H. Friedman, R. A. Olshen, and C. J. Stone. 1984. Classification and Regression Trees. Belmont, CA: Wadsworth.
21. Haykin, S. 1994. Neural Networks: A Comprehensive Foundation. Upper Saddle River, NJ: Prentice-Hall.
22. Anderson, J. A. 1995. An Introduction to Neural Networks. Cambridge, MA: MIT Press.
23. Nauck, D., F. Klawonn, and R. Kruse. 1997. Foundations of Neuro-Fuzzy Systems. Chichester, UK: J. Wiley and Sons.
24. Goldberg, D. E. 1989. Genetic Algorithms in Search, Optimization and Machine Learning. Reading, MA: Addison-Wesley.
25. Michalewicz, Z. 1996. Genetic Algorithms + Data Structures = Evolution Programs. Berlin, Germany: Springer-Verlag.
26. Mitchell, M. 1996. An Introduction to Genetic Algorithms. Cambridge, MA: MIT Press.