4.3 Karar Ağaçları (ağaç algoritmaları)

Ağaç tabanlı öğrenme algoritmaları, en çok kullanılan ve denetimli öğrenme yöntemlerinden biri olarak düşünülmektedir. Ağaç tabanlı yöntemler, yüksek doğruluk, kararlılık ve yorumlanma kolaylığına sahiptir. Doğrusal modellerin aksine doğrusal olmayan ilişkileri de oldukça iyi eşleyebilirler. Sınıflandırma veya regresyon, elde edilen her türlü sorunun çözümünde uyarlanabilirler. Karar ağaçları, rastgele orman, gradyan güçlendirme gibi yöntemler, her türlü veri bilimi probleminde yaygın şekilde kullanılmaktadır

Karar ağacı öğrenmesi, endüktif(inductive) çıkarım için en yaygın kullanılan pratik yöntemlerden birisidir. Karar ağacı öğrenmesi, öğrenilen fonksiyonun bir karar ağacı tarafından temsil edildiği, kesikli değerli hedef fonksiyonlarını yaklaştırmak için kullanılan bir yöntemdir. Karar ağacı, sınıflandırma problemlerinde çoğunlukla kullanılan bir denetimli öğrenme algoritmasıdır (önceden tanımlanmış bir hedef değişkene sahiptir). Hem kategorik hem de sürekli giriş ve çıkış değişkenleri için çalışır. Öğrenilen ağaçlar insan okunabilirliğini artırmak için if-then kural setleri olarak temsil edilebilirler. Karar ağacı, bir ağaç yapısı biçiminde sınıflandırma veya regresyon modelleri oluşturur. Bir veri kümesini daha küçük ve daha küçük alt kümelere bölerken, aynı zamanda ilişkili bir karar ağacı aşamalı olarak geliştirilir. Nihai sonucu, karar düğümleri ve yaprak düğümleri olan bir ağaçtır. Bir karar düğümü, iki veya daha fazla dallara sahiptir. Yaprak düğüm bir sınıflandırma veya kararı temsil eder. Bir ağaçtaki en üstteki karar düğümü, kök düğüm olarak adlandırılan en iyi belirleyiciye karşılık gelir. Karar ağaçları hem kategorik hem de sayısal verileri işleyebilir. Karar ağaçlarını şöyle ikiye ayırabiliriz:

- **Kategorik Değişken Karar Ağacı:** Kategorik hedef değişkeni olan Karar Ağacı, kategorik değişken karar ağacı olarak adlandırılır. Sınıflandırma Karar Ağaçları da denilebilir.
- **Sürekli Değişken Karar Ağacı:** Karar Ağacı sürekli hedef değişkenine sahipse, Sürekli Değişken Karar Ağacı olarak adlandırılır. Regresyon Karar Ağaçları da denilebilir.

Karar ağaçları da diğer ağaç veri yapıları gibidir ve aynı terminolojiyi kullanmaktadır. Kısaca karar ağaç terminolojine bakacak olursak:

- **Kök Düğüm:** Tüm örneği temsil eder ve bu düğüm daha sonra iki veya daha fazla kümeye ayrılır.
- **Parçalama**: Bir düğümün iki veya daha fazla alt düğümlere bölünmesi işlemidir.
- **Karar Düğümü:** Bir alt düğüm başka alt düğümlere bölünürse, karar düğümü olarak adlandırılır.
- **Yaprak Düğümü:** Bölünmeyen düğümlere Yaprak veya Terminal düğümü denir.
- **Budama**: Karar düğümünün alt düğümlerini kaldırdığımızda, bu işleme budama denir. Yani parçalama işleminin tersi diyebiliriz.
- **Alt Ağaç:** Tüm ağacın bir alt kısmı şube veya alt-ağaç olarak adlandırılır.
- **Ana ve Çocuk Düğümü:** Alt düğümlere ayrılmış olan bir düğüme, alt

düğümlerin ana düğümü ve alt düğümlerine de çocuk düğümü adı verilir.

Karar Ağaç algoritmalarından bazılarını şöyle sıralayabiliriz, ID3, C4.5, C5.0 ve CART.

Tek karar ağacından daha iyi tahmin edici performans elde etmek için çeşitli karar ağaçlarını birleştiren topluluk yöntemleri vardır. Bunlara ağaç topluluk algoritmaları denir. Ağaç topluluk algoritmalarının ana amacı bir grup zayıf öğrenicinin bir araya gelerek güçlü bir öğrenici topluluk oluşturmaktır.

Topluluk karar ağaçlarını gerçekleştirmek için birkaç teknik vardır, bunlar:

1. Torba (bagging)

2. Arttırma (boosting)

Bagging (Bootstrap Aggregation), bir karar ağacının varyansını azaltmak istediğimiz zaman kullanılır. Buna örnek olarak Rastgele Orman algoritması verilebilir.

Boosting, bir öngörü koleksiyonu oluşturmak için kullanılan diğer bir topluluk tekniğidir. Bu teknikte öğreniciler, erken öğrenenlerin basit modellerini verilerle uyuşturduktan sonra hatalar için verileri analiz ederek sırayla öğrenmeye dayanır. Bu tekniğe örnek olarak Gradient Boosting örnek verilebilir.

4.3.1 Gini Dizini

Gini indeksi(dizini) veya Gini katsayısı, İtalyan istatistikçi Corrado Gini tarafından 1912'de geliştirilen istatistiksel bir ölçüdür. Katsayı, 0 (%0) ile 1 (%100) aralığındadır; 0, mükemmel eşitliği temsil eder ve 1, mükemmel eşitsizliği temsil eder. Gini indeksi, rastgele seçilen bir öğenin ne sıklıkta yanlış tespit edildiğini ölçmek için kullanılan bir metriktir. Düşük gini indeksi olan bir özellik tercih edilmelidir. Gini indeksi kategorik hedef değişkeni için başarılı veya başarısız olarak çalışır. Gini indeks, yalnızca ikili bölmeleri (binary: 1 veya 0) gerçekleştirir ve yüksek gini indeksi homojenliği artırır. CART (Sınıflama ve Regresyon Ağacı) ikili bölmeler oluşturmak için gini yöntemini kullanır.

$$\text{Gini Index} = 1 - \sum_{j} P_j^2$$

4.3.2 Entropi

Entropi, rasgele bir değişkenin belirsizliğinin ölçüsüdür, örneklerin keyfi bir koleksiyonunun saf olmayanlığını karakterize eder. Entropi ne kadar yüksek olursa elde edilen bilgi de o kadar fazla olur. Sezgisel olarak, belirli bir olayın öngörülebilirliğinden bahseder.

if a random variable x can take N different value ,the i^{th} value x_i with probability $p(x_i)$,we can assocoate the folloeing entropy with x:

$$H(x) = -\sum_{i=1}^{N} p(x_i) \log_2 p(x_i)$$

4.3.3 Bilgi Kazanımı

Entropi, tipik olarak, eğitim örneklerini daha küçük alt gruplara bölmek için bir karar ağacında bir düğümü kullandığımızda değişir. Bilgi kazancı, entropideki bu değişimin bir ölçüsüdür.

Definition: Suppose S is a set of instances, A is an attribute, S_v is the subset of s with A = v and Values(A) is the set of all possible of A,then

$$Gain(S,A) = Entropy(S) - \sum_{v \in values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

$|S|$ denotes the size of set S

4.3.4 Ağaç Algoritmaları

4.3.4.1 CART

CART Algoritması, Breiman ve arkadaşları (1984) tarafından bulunmuş, sınıflandırma ve regresyon ağaçlarına dayanmaktadır. Bir CART ağacı, tüm öğrenme örneğini içeren kök düğümünden başlayarak, bir düğümün iki çocuk düğümüne tekrar tekrar bölünmesiyle oluşturulan bir ikili karar ağacıdır. CART, her düğümde en büyük bilgi kazanımını sağlayan özellik ve eşiği kullanarak ikili ağaçlar oluşturur.

Ağacı geliştirmenin temel fikri, her düğümde olası tüm bölünmeler arasında bir bölünme seçmektir, böylece ortaya çıkan çocuk düğümler “en saf” olacaktır. Bu algoritmada, sadece tek değişkenli bölünmeler dikkate alınır. Yani, her bölünme sadece bir tahmin değişkeninin değerine bağlıdır. Tüm olası bölünmeler, her bir tahmin edicinin muhtemel bölünmesinden oluşur.

CART'ın ana unsurları şu şekildedir:

1. Bir değişkenin değerine bağlı olarak bir düğümde veri bölme kuralları;
2. Bir düğüm terminal olduğunda karar vermek için durdurma kuralları ve artık bölünemez; ve
3. Son olarak, her bir terminal düğümündeki hedef değişken için bir tahmin.

CART'ın bazı yararlı özellikleri ve avantajları:

- CART parametrik olmayan bir parametredir ve bu nedenle belirli bir dağıtım türüne ait verilere güvenmez.
- CART, giriş değişkenlerinde aykırı değerlerden önemli ölçüde etkilenmez.
- Karar ağaçlarını "aşırı büyümeye" zorlamak için kuralları durdurabilir ve ardından ağacı en uygun boyuta getirebilirsiniz. Bu yaklaşım, veri kümesindeki önemli yapının çok yakında durdurarak gözden kaçırılma olasılığını en aza indirir.
- CART, her iki testin de bir test verisi seti ve testin doğruluğunu daha doğru bir şekilde değerlendirmek için çapraz doğrulama ile birleştirir.
- CART aynı değişkenleri ağacın farklı bölümlerinde birden fazla kez kullanabilir. Bu yetenek, değişken kümeleri arasındaki karmaşık bağımlılıkları ortaya çıkarabilir.
- CART, değişkenlerin giriş setini seçmek için diğer tahmin yöntemleriyle birlikte kullanılabilir.

CART Algoritması, birkaç basit adımı içerir ve bunlar şunlardır:

1. Etiketli Girdi verilerini - Bir Hedef Değişkeni ve Bağımsız Değişkenler listesiyle birlikte alın
2. Best Split: Bağımsız değişkenlerin her biri için En İyi Split'i bulun
3. En İyi Değişken: Bölme için En İyi Değişkeni Seçin
4. Giriş verilerini Sol ve Sağ Düğümlere ayırın
5. Durma kriterlerini karşılayana kadar düğümlerin her birinde adım 2-4'e devam edin.
6. Karar Ağacı Budama: Karar Ağacı inşa etmek için adımlar

4.3.4.2 ID3

Çok basit bir şekilde, ID3 sabit bir örnek kümesinden bir karar ağacı oluşturur. Ortaya çıkan ağaç, gelecekteki örnekleri sınıflandırmak için kullanılır. Örnekte birkaç özellik vardır ve bir sınıfa aittir (evet veya hayır gibi). Karar ağacının yaprak düğümleri sınıf adını içerirken, yapraksız düğüm bir karar düğümüdür. Karar düğümü, her dalın (başka bir karar ağacına) özniteliğin olası bir değeri olan bir özellik testidir. ID3, hangi özelliğin bir karar düğümüne gittiğine karar vermesine yardımcı olmak için bilgi kazancı kullanır. Bir karar ağacını öğrenmenin avantajı, bir bilgi mühendisinden ziyade bir programın bir uzmandan bilgi almasıdır.

ID3, sınıflarını sabit bir eğitim örneklerinden türetmesi anlamına gelen, bir artışlı

olmayan algoritmadır. Artışlı bir algoritma, gerekirse mevcut kavram tanımını yeni bir örnekle gözden geçirir. ID3 tarafından yaratılan sınıflar endüktif, yani, küçük bir dizi eğitim örneği verildiğinde, ID3 tarafından oluşturulan belirli sınıfların tüm gelecek örnekler için çalışması beklenir. Bilinmeyenlerin dağılımı, test vakaları ile aynı olmalıdır. Endüksiyon sınıflarının her durumda, sonsuz sayıda örneği sınıflandırabildikleri için çalıştıkları kanıtlanamaz. ID3'ün (veya herhangi bir endüktif algoritmanın) verileri yanlış sınıflandırması olabileceğini unutmayın.

ID3 tarafından kullanılan örnek verilerin belirli gereksinimleri vardır; bunlar:

- Öznitelik değeri açıklaması - aynı öznitelikler her bir örneği tanımlamalı ve sabit bir sayı değerine sahip olmalıdır.
- Önceden tanımlanmış sınıflar - bir özniteliğin öznitelikleri zaten tanımlanmış olmalıdır, yani ID3 tarafından öğrenilmemelidir.
- Ayrık sınıflar - sınıflar keskin bir şekilde belirtilmelidir. "Sert, oldukça sert, esnek, yumuşak, oldukça yumuşak" bir metal gibi belirsiz kategorilere ayrılan sürekli sınıflar şüphelidir.
- Yeterli örnekler - endüktif genelleme kullanıldığı için (yani, kanıtlanamaz), geçerli modelleri rastlantısal olaylardan ayırmak için yeterli test vakası olmalıdır.

ID3 hangi niteliğin en iyi olduğuna nasıl karar verirken, bilgi kazancı olarak adlandırılan bir istatistiksel özellik kullanılır. Kazanç, belirli bir özniteliğin eğitim örneklerini hedeflenen sınıflara ne kadar iyi ayırdığını ölçer. En yüksek bilgiye sahip olan (sınıflandırma için en kullanışlı bilgi) seçilir. Kazanımı tanımlamak için önce entropi denilen bilgi teorisinden bir fikir borçluyuz. Entropi, bir öznitelikteki bilgi miktarını ölçer.

4.3.4.3 C4.5

C4.5, sınıflandırma problemleri için kullanılabilecek Karar Ağaçları (DT) üreten Ross Quinlan tarafından geliştirilen bir algoritmadır. C4.5, birtakım örneklerden gelen karar ağaçları formunda sınıflandırma kurallarını uyaracak bir bilgisayar programıdır. C4.5, verileri sınıflandırmak için entropi gibi bilgi teorik kavramlarını kullanır.

ID3 algoritmasında bazı eksiklikler ve sorunlar vardı ve bu sorunlar Quinlan'ın geliştirdiği bu C4.5 algoritmasıyla giderilmiştir. C4.5 Algoritması ID3 algoritmasının bütün özelliklerine sahip olmakla birlikte, ID3 için bahsedilen özelliklere yenilerin eklenmesi ile oluşmuştur. Bu yeni eklenen özellikler şunlardır:

- Bölünme-Dağılma Bilgisi: Bir kategorik özelliğin olası değer çeşitliliği ne kadar yüksek olursa o özelliğin bilgi kazancı da bir o kadar yüksek çıkar ve bu durum ağacın doğruluğunu kötü bir şekilde etkiler. Bu tip özellikler işe yaramadıkları gibi bilgi kazancı yüksek özelliklerin de önüne geçip veride gizlenmiş kuralların bulunmasına engel olurlar.
- Sayısal özellik değerlerinin hesaba katılması: ID3 algoritması daha önce sadece nominal değerlere sahip veri tipleri ile işlemler yapabiliyorken, C4.5 algoritması ise sayısal tipteki veriler için de bir yöntem geliştirmiştir.
- Özelliklerin kayıp değerleriyle baş edilmesi

4.3.4.4 C5.0

C5.0, ID3 karar ağacı algoritmasının ileri versiyonlarından biridir. C5.0 algoritması ise C4.5'in geliştirilmiş hali olup, özellikle büyük veri setleri için kullanılmaktadır. C5.0 algoritması, doğruluğu arttırmak için boosting algoritmasını kullandığından, boosting ağaçları olarak da bilinir. C5.0 algoritması C4.5'e göre çok daha hızlı olup, hafızayı daha verimli kullanmaktadır. Her iki algoritmanın sonuçları aynı olsa da C5.0 biçim olarak daha düzgün karar ağaçları elde etmemizi sağlamaktadır.

4.3.5 Ensemble Yöntemler

Ensemble öğrenme, birkaç modeli birleştirerek makine öğrenimi sonuçlarını geliştirmeye yardımcı olur. Bu yaklaşım, tek bir modele kıyasla daha iyi bir tahmin performansının üretilmesine izin verir.

Ensemble yöntemleri, çeşitli makine öğrenme tekniklerini, varyansı, önyargıyı azaltmak veya tahminleri arttırmak için tek bir tahmin modelinde birleştiren meta algoritmalardır. Topluluk yöntemleri iki gruba ayrılabilir:

- Temel öğrenicilerin sıralı olarak oluşturulduğu sıralı topluluk yöntemleri (örn. AdaBoost). Sıralı yöntemlerin temel motivasyonu, temel öğrenenler arasındaki bağımlılığı kullanmaktır. Genel performans, daha önce yanlış yazılmış olan daha yüksek ağırlıktaki örneklerin tartılmasıyla artırılabilir.
- Temel öğrenicilerin paralel olarak oluşturulduğu paralel topluluk yöntemleri (ör. Rastgele Orman). Paralel metotların temel motivasyonu, temel öğrenenler arasındaki bağımsızlığı istismar etmektir çünkü hata, ortalama olarak önemli ölçüde azaltılabilir.

Çoğu ensemble metodu homojen taban öğrenicilerini, yani aynı tipteki öğrenicileri homojen topluluklara götüren tek bir temel öğrenme algoritması kullanır. Ayrıca heterojen öğrenenleri kullanan farklı yöntemler de vardır, yani farklı türden öğrenenler, heterojen topluluklara yol açarlar. Ensemle yöntemlerinin bireysel üyelerinden daha doğru olması için, temel öğrenenler olabildiğince doğru ve mümkün olduğunca farklı olmak zorundadır.

4.3.5.1 Bagging

Torbalama (Bagging), bootstrap agregasyonu anlamına gelir. Bir tahminin varyansını azaltmanın bir yolu, çoklu tahminlerin birlikte ortalamasıdır. Örneğin, M farklı ağaçları veriyi farklı alt kümeler üzerinde (değişim ile rastgele seçilen) eğitebilir ve topluluğu hesaplayabiliriz.

Torbalama, temel öğrenicileri eğitmek için veri alt kümelerini elde etmek için önyükleme örneklemesi kullanır. Temel öğrenenlerin çıktılarını toplamak için, torbalama, sınıflandırma için oylama ve regresyon için ortalama kullanır.

4.3.5.2 Boosting

Yükseltme (Boosting), zayıf öğrenicileri güçlü öğrenicilere dönüştürebilen bir algoritma ailesini ifade eder. Yükseltmenin temel ilkesi, küçük karar ağaçları gibi verilerin basitleştirilmiş versiyonları gibi rasgele tahminlerden sadece biraz daha iyi olan zayıf öğrenenlerin bir dizisine uymaktır.

Daha önceki hallerde yanlış sınıflandırılmış örneklere daha fazla ağırlık verilir. Tahminler daha sonra nihai tahmin üretmek için ağırlıklı bir çoğunluk oyu (sınıflandırma) veya ağırlıklı bir toplam (regresyon) ile birleştirilir. Boosting ve comitee yöntemleri arasında, bagging gibi temel fark, temel öğrenenlerin, verilerin ağırlıklı bir versiyonu üzerinde sırayla eğitilmesidir.