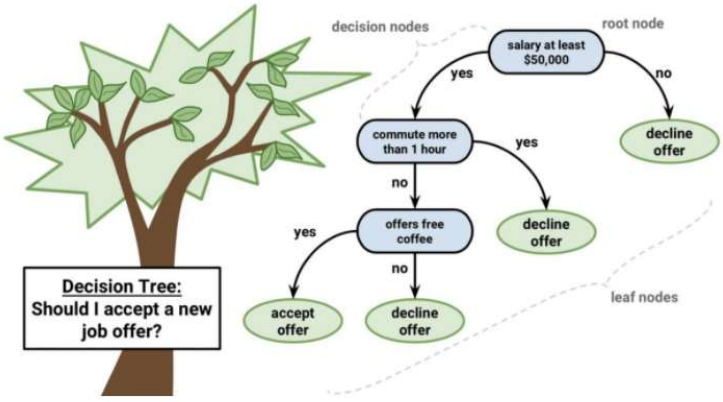
Not Defteri Makine Öğrenmesi

Herkes ona bağlanır ondan enerji alır: LightGBM

Microsoft tarafından geliştirilen, gradyan güçlendirmesi tabanlı karar ağacı algoritması olan LightGBM, tahmin modelleri için karar noktalarını optimum şekilde ayırarak ürettiği ağaçlar ile başarılı tahminler üretmemize yardımcı olur.

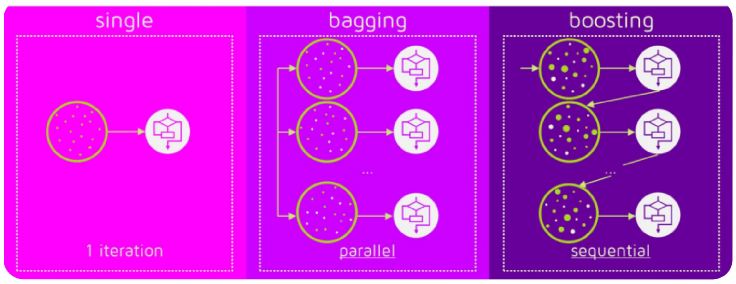
Cem İstanbullu • 07 EKİ 2021

Gradyan güçlendirmeli karar ağaçları günümüzde makine öğrenmesi alanında gerçekleştirilen projelerde en çok tercih edilen gözetimli öğrenme algoritmalarındandır. ***Boosting*** temelde birkaç zayıf öğreniciyi bir araya getirerek güçlü öğreniciler oluşturmayı hedefler.



Bu algortima kapsamında artıklar üzerine tek bir tahminsel model formunda olan modeller serisi kurulur. Önce bir model kurulur (F modeli) ve bu modelin hataları tespit edilir. Hatalar bağımlı değişken olur. Modellendikten sonra bir daha artık

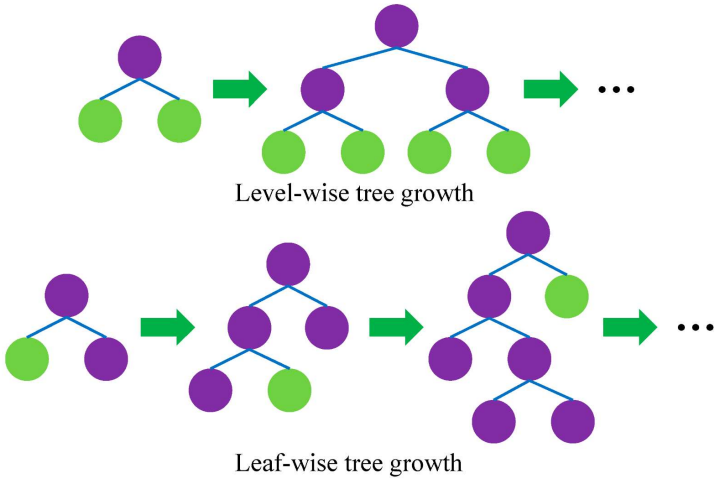
elde edilir ve artıklar model hazırlanır. Artıklar modellenerek zayıf sınıflandırıcılardan güçlü sınıflandırıcılar elde edilir. ***Boosting*** algoritmalarında amaç karar noktalarını ***gradient descent*** kullanarak hassaslaştırmaktır. Burada öngörücüler sırayla denenerek sonraki her model bir önceki modelin hatalarını düzelterek ilerler ve birbirine bağımlı ağaçlar her denemede optimize edilir.



Kaynak: Quantdare

LightGBM'in diğer boosting algoritmalarından farkı nedir?

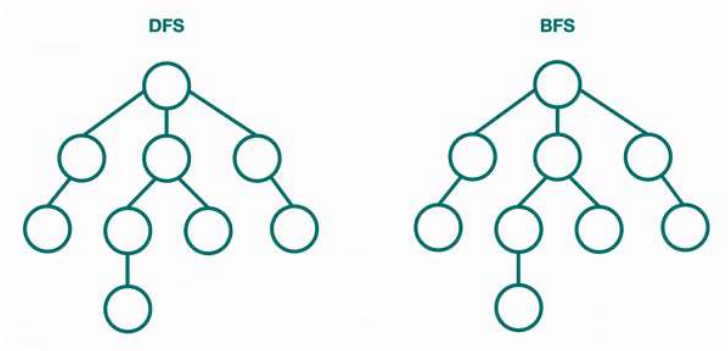
Öncelikle LightGBM'in *level-wise* büyüme stratejisi yerine *leaf-wise* büyüme stratejisi izlediğini ifade etmek gerek.



Level-wise büyüme ağaçları dengede tutarken, LightGBM *leaf-wise* stratejisini

kullanarak dengesiz bir ağacın büyümesine izin verir ve en fazla kaybı olan yaprağı bölmeyi hedefleyerek kaybı minimalize eder. *Level-wise* büyüyen bir ağacı *leaf-wise* yönteme çevirebilirken bunun tersi için geçerli değildir. *Leaf-wise* düzey olarak büyümediği, ancak yaprak açısından büyüdüğü için, veriler küçük olduğunda aşırı öğrenme (*overfit*) olabilir. Bu durumlarda ağaç derinliğini kontrol etmek önemlidir.

LightGBM'in bir diğer farkı ise *breadth-first* arama yerine *depth-first* aramayı kullanmasıdır.

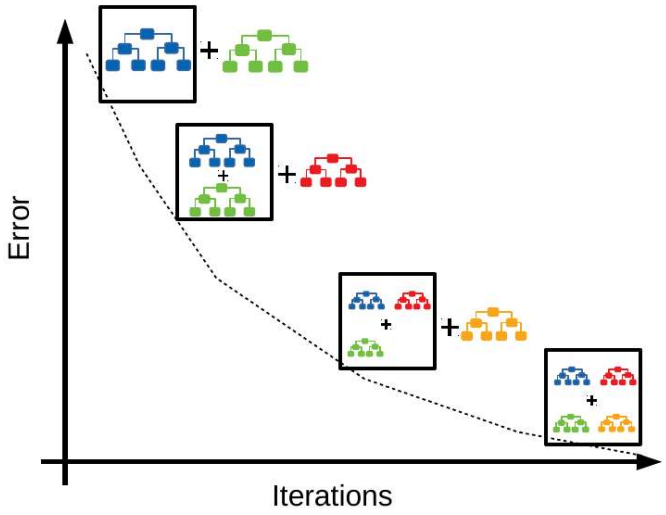


Kaynak

Breadth-first karar ağaçları, *depth-first* karar ağaçlarına benzer şekilde *brute-force* tarzında oluşturulur. Bir *depth-first* ağacında ilk olarak, kök düğüme yerleştirilecek bir öznitelik seçilir ve bu öznitelik için bazı kriterlere göre bazı dallar oluşturulur. Ardından, eğitim örnekleri kök düğümden uzanan her dal için bir tane olacak şekilde alt kümelere ayrılır. Seçilen bir dal için, yalnızca ona gerçekten ulaşan örnekler kullanılarak işlem tekrarlanır. Her adımda, genişletmeler için mevcut olan tüm alt kümeler arasından "en iyi" alt küme seçilir. Bu oluşturma işlemi, tüm düğümler

saf olana veya belirli bir genişleme sayısına ulaşılana kadar devam eder. *Breadth-first* ağaç yapısında yaprakların sırası her zaman aynıyken, *depth-first* ağacı için diğer sıralamaların duruma göre yeniden seçilebilecektir.

LightGBM'i "light" yapan özellik nedir?



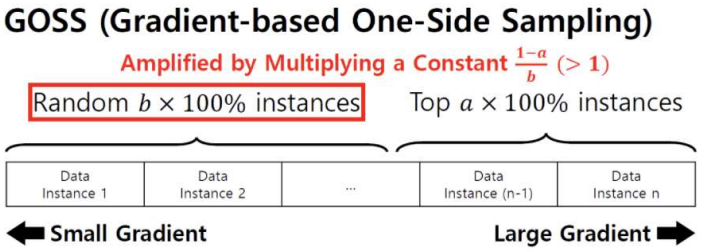
Karar ağaçlarında öğrenme sürecindeki eforun büyük çoğunluğu, ayrımın en iyi nereden yapılacağının bulunması kısmında harcanır. *Gradient boosting* algoritmasının önceki uygulamalarında, en iyi bölmeyi bulmak için *pre-sorted* (önceden sıralanmış) algoritma kullanılırken LightGBM'de *histogram-based* (histogram tabanlı) algoritmalar kullanılıyor.

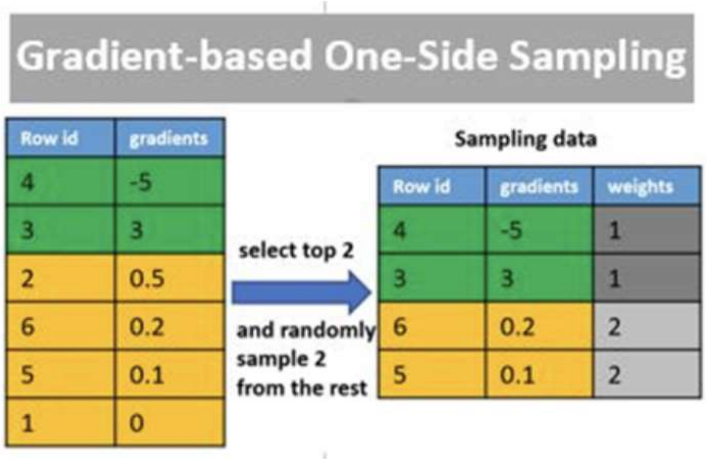
***Pre-sorted*:** Önceden sıralanmış değerlerin tüm olası bölünme noktaları numaralandırılarak değerlendirme yapılır.

***Histogram-based*:** Eğitim sırasında özellik histogramları oluşturmak için sürekli özellikleri ayrı kutulara yerleştirilir.

Histogram tabanlı algoritma, önceden sıralanmış algoritmadan daha verimli çalışır ancak hem gözlemler hem de özellikler açısından veri kümesinin boyutu arttıkça ikisi de yavaşlar.

LightGBM, kullanımı daha efektif olan ve bellek kullanımı azaltarak şartlar sağlandığında hızlı bir eğitim sunan histogram tabanlı algoritma ile başladı. Ancak burada karşılaşılan temel sorun, veri seti büyüdüğünde bilgi kazancı açısından en iyi bölünmeyi seçmek için bütün gözlemleri taramasıydı. Bu zaman kaybı olarak geri dönüyordu. Bu tarama her özellik için yapıldığından histogram tabanlı algoritmanın başarısı veri boyutu ve değişken sayısına bağlıydı. Buna çözüm olarak *gradient-based one-side sampling* (GOSS) ve *exclusive feature bundling* (EFB) uygulandı.





Eğitim sırasında modelin veri dağılımını etkileyip doğruluğu etkilememesi için ne sadece küçük eğimli veri örneklerine ne de sadece büyük gradyanlı olanlara odaklanmasını isteriz. GOSS, elimizdeki verinin dağılımını göz önünde bulundurarak bir ağaç çıkartır. En büyük gradyanlara sahip veri örnekleri seçilir. Daha küçükleri içerisinden rastgele olarak popülasyonun sadece belli bir kısmı alınır. Küçük gradyana sahip gözlemler belli bir ağırlıkla çarpılarak dağılım korunmaya çalışılır. GOSS, temel olarak verinin tümünde karar noktaları oluşturmak yerine bunları kümeleyerek temas edilecek noktaları azaltıp odağı yoğunlaştırmaktır.

EFB ile ise çok sütunlu bir veri setinde özellik bakımından kıt olanları (örneğin çok sayıda sıfır değer içeren özellikler) içeriklerini bozmayacak şekilde birleştirerek seyrek özelliklerin birbirini dışlamasının önüne geçilmiştir. Gereksiz boyutluluğu engelleyen bu yöntem karmaşıklığı azaltarak eğitime hız kazandırmıştır.

GBM
= Desicion Tree + Boosting + Gradient Descent
LightGBM = GBM + GOSS + EFB

Bu iki ana faktör LightGBM'i *light* yapan ana bileşenler olup kategorik olup kardinalitesi yüksek olan verilere karşı LightGBM'in neden dayanıklı olup diğer algoritmalara göre başarılı olduğunu açıklar niteliktedir.

Ne zaman LightGBM?

LightGBM, küçük boyutlu veri setleri için uygun bir algoritma değildir. Karar noktalarını hassaslaştırma özelliğinden dolayı aşırı öğrenme (*overfit*) durumu gözlemlenebilir. 10 bin üzeri satırlı veriler için uygun olduğu önerilse de bu konuda kesin bir yaklaşım da yoktur. Büyük boyutlu veri setlerinde yüksek doğruluk sağlayacaktır.

Parametre seçimi ve özellikleri

LightGBM modelini Scikit-learn API içerisinden değil de,

pip install lightgbm
import lightgbm as lgb

şeklinde kullanarak Training API içerisinden çağırdığımız vakit parametre seçenekleri kısmında birçok seçeneğimiz olacaktır. Bunları temel parametreler, kontrol parametreleri ve IO parametreleri olmak üzere üç gruba ayırabiliriz.

Temel Parametreler

task: Çalıştırılmak istenen görevi belirtir. Ya eldeki veriler ile *train* ederek model kurulur ya da *predict*

argümanı ile kurulan modelden tahminler üretilir. Varsayılan olarak *train* gelmektedir.

objective: Kurulmak istenen algoritmanın hangi probleme yönelik olduğunun belirtildiği parametredir. Varsayılan olarak *regression* gelmekle birlikte *binary* ve *multiclass* gibi problemler de verilebilmektedir.

boosting: Boosting etme türünü belirtir. *gbdt* (traditional gradient boosting decision tree), *rf* (random forest), *dart* (dropouts meet multiple additive regression tree) veya *goss* (gradient-based one-side sampling) seçilebilir. Varsayılan olarak *gbdt* gelmektedir.

- GBDT: Gradyan güçlendirmeli bir yöntemdir. Öğrenicileri makul bir hızda birleştirir.
- RF: *Random forest* temelli yöntemdir. Değer olarak *rf* ayarlandığı takdirde artık *bagging* metodu çalışacaktır.
- DART: GBDT'deki aşırı öğrenme sorununu çözmeye çalışır. Sinir ağları, desen düzenlemeyi iyileştirmek ve yakalaması güç noktaları yakalayabilmek için geliştirilmiştir. Daha iyi sonuç vermekle birlikte oldukça yavaştır ve kullanırken beraberinde gelen birçok parametreyi de ayarlamak gerekir. Bir DART güçlendirici kullanmayı tercih ederken, *drop-out* işlemini durdurmamız gerektiğinin farkında olmak önemlidir.
- GOSS: Örnekleri daha büyük gradyanlara ayırarak GBDT için yeni bir örnekleme yöntemi sağlar. Küçük veri setlerinde aşırı öğrenme sorunu vardır.

num_iterations: Gerçekleştirilecek güçlendirme turlarının sayısını, oluşturulacak ağaç sayısını kontrol eder. Varsayılan değer 100'dür. Birbiriyle zıt etkiye sahip olup bağlantılı olan *learning_rate* parametresini de beraber değiştirmek gerekir. Eğer *num_iterations* değerini arttırırsanız, *learning_rate* değerini azaltmalısınız. İterasyon sayısı arttıkça model kurma süresi de uzayacaktır. Modelin artık öğrenecek bir şeyi kalmadığında iterasyonu kesmek için *early_stop* parametresini de aktifleştirmek gerekir ki aşırı öğrenmeye gidilmeden eğitim kesilebilsin.

learning_rate: Model ağırlıkları her güncellendiğinde tahmini hataya yanıt olarak modelin ne kadar değiştirileceğini kontrol eden bir hiperparametredir. Varsayılan değer 0.1'dir. Çok küçük bir değerin seçilmesi zorlayıcıdır. Çok küçük bir değer, takılıp kalabilecek uzun bir eğitim süreci ile sonuçlanabilirken çok büyük bir değer optimal olmayan bir ağırlık setinin çok hızlı öğrenilmesine veya kararsız bir eğitim sürecine neden olabilir.

num_leaves: Ağaç yapısını ve karmaşıklığını kontrol etmek için en önemli parametredir. Tek bir ağaçtaki karar noktalarının sayısını belirler. 2^(*max_depth*)'ten daha küçük bir değere sahip olmalıdır. Varsayılan olarak 31 tanımlıdır.

device_type: LightGBM'i kendi kütüphanesinden çalıştırmanın en büyük avantajı GPU kullanılabiliyor olmasıdır. Varsayılan olarak CPU gelmekle beraber GPU ve CUDA da seçilebilir.

seed: Farklı kişiler tarafından gerçekleştirilen ya da defalarca tekrarlama gerektiren durumlarda her

seferinde aynı model kısmının oluşturularak karşılaştırmaların daha düzgün yapılmasına olanak tanır. Her defasında virgülden sonraki basamakların dahi aynı gelmesini sağlar. Varsayılan olarak *none* gelmektedir.

Kontrol Parametreleri

max_depth: Maksimum derinlik ne kadar yüksekse, ağaç o kadar fazla seviyeye sahiptir. Bu da onu daha karmaşık ve aşırı öğrenmeye iter hale getirir. Çok düşük değer verildiğinde de öğrenememe durumu ile karşılaşılacaktır. Model öğrenimi için önemli bir parametredir. Varsayılan değeri -1'dir.

min_data_in_leaf: Bir yapraktaki minimum örnekleri tanımlar. Optimal değeri eğitim örneklerinin sayısına ve *num_leaves* değerine bağlıdır. Modelinizin aşırı öğrenmeye gittiğini hissettiğinizde bu değeri artırabilirsiniz. Varsayılan değer 20'dir.

bagging_fraction: Satırlar için kullanılır ve yineleme yapmadan verilerin bir kısmını rastgele seçerek kullanılacak veri oranını belirtir. Eğitim süresini hızlandırıp aşırı öğrenmeyi azaltabilir. Varsayılan değeri 1'dir.

feature_fraction: Değeri 1.0'dan küçükse her yinelemede rastgele bir özellik alt kümesi seçilecektir. Örneğin 0.75 verilen değerde model ağaç oluşturmak için her yinelemede toplam özelliklerin %75'ini rastgele seçecektir. Eğitim süresini hızlandırıp aşırı öğrenmeyi azaltabilir. Varsayılan değeri 1'dir.

bagging_freq: Bagging sıklığını ifade eder. 0 demek *bagging* metodunu devre dışı bırakır. *bagging_fraction * N_train_examples* eğitim örneklerinin bir alt kümesini örnekler. k sırasındaki ağacın eğitimi bu altkümede gerçekleştirilir. Bu örnekleme her ağaç için (yani her yineleme) veya her *bagging_freq* ağacı eğitildikten sonra yapılabilir. *Bagging*'i etkinleştirmek için *bagging_fraction* değeri de 1'den küçük bir değere ayarlanmalıdır. Varsayılan değer 0'dır.

early_stopping: Doğrulama metriğince gelen değer önceki değerden daha iyi bir değere sahip olamayıp modelin gelişimi artık durduysa aşırı öğrenmeye kaçmamak için model eğitimi en optimum noktada kesilir. Varsayılan olarak *false* gelmektedir.

max_cat_threshold: Kardinalitesi yüksek olan kategorik değişkenleri *one-hot* kodlamak ağacı dengesiz bir yapıya sokma riski taşır ve iyi bir doğruluk elde etmek için çok derinlere büyümesi gerekir. Bu noktada kardinalitesi yüksek olan değişkenlere en uygun çözüm, kategorilerini iki alt kümeye ayırarak kategorik bir özelliği bölmektir. Bunu gerçekleştirirken de veriye uygun *max_cat_threshold* değeri belirlemeliyiz. 0'dan büyük olmak zorunda olup varsayılan değeri 32'dir.

IO Parametreleri

max_bin: Değişken başına gelecek maksimum benzersiz değer sayısıdır. Küçük

verilen *max_bin* değeri performansı arttırır. Varsayılan değer 255'tir.

use_missing: LightGBM yapısı gereği eksik veriler ile başa çıkabilir. Boş gelen değerler model oluşturulmasını engellemez. Varsayılan değer *true*'dur, ancak eksik veriyle başa çıkma işlemini devre dışı bırakmak için bu parametreyi kapatmak gerekir.

zero_as_missing: Varsayılan değeri *false* olarak gelmektedir. Veri setindeki boş değerleri 0 olarak atayıp boş değer doldurmayı LightGBM'e bırakmak istiyorsanız bunu *true* yapmanız gerekir.

Özetleyecek olursak LightGBM ile karar ağacı oluştururken dikkat edilmesi gereken nokta oluşturduğumuz ağacın *leaf-wise* bir büyüme stratejisi sergilediğidir. Bu bağlamda ağacımızın *num_leaves*, *max_depth* ve *min_data_in_leaf* parametrelerini kontrol altında tutmalıyız. Bunların oluşturulacak ağacın ana iskeletini, yaprakları ve derinliğini etkilediğini her zaman göz önünde bulundurmak gerekir. Parametre optimizasyonu yaparken eğer koşullarınız elverişli ise hızı arttırmak için GPU kullanımı ile ağacınızı oluşturabilirsiniz. Bu işlem hız kazandıracaktır. İterasyon sayısı ile öğrenme sıklığının ters ilişkiye sahip olduğunu ve birinin değiştirilmesi halinde ötekinin de revize edilmesi gerektiğini ise unutmamalı. Düşük öğrenme hızı ile ne kadar çok iterasyon yaparsanız o kadar ağaç oluşturulup eğitim süresiniz de o kadar artacaktır. Her iterasyon basamağında oluşturulan yeni ağaçların bir kazanım sağlayacağının garantisi

yoktur. Buna çözüm olarak iyileştirme sonlandığı vakit model geliştirmesini durduracak parametre olan *early_stop*'ı çalıştırmak gerekir. Böylece aşırı öğrenmeden bir adım daha uzaklaşmış oluruz.

LightGBM hakkında daha geniş kapsamlı bilgiye erişmek ve kariyerinizde Machine Learning bilginizle fark yaratmak isterseniz Miuul'un sunduğu Zaman Serileri eğitimine göz atabilirsiniz.

Kaynaklar

- Microsoft, LightGBM
- O'Reilly, LightGBM
- Neptune.AI, Understanding LightGBM Parameters (and How to Tune Them)
- Waikato Üniversitesi, Best-first Decision Tree Learning
- GitHub, LightGBM

Etiketler

LightGBM / makine öğrenmesi

Cem İstanbullu

İlginizi Çekebilir

Parametre optimizasyonuna pratik bir çözüm: Optuna

Cem İstanbullu

Python ile doğrusal regresyon uygulaması

Cem İstanbullu

Sınıflandırma problemleri için kullanılabilecek yöntemler

Cevahir Özgür

Miuul topluluğunun bir parçası ol!

E-posta adresiniz | Abone Ol

Abone ol butonuna tıklayarak Miuul'dan pazarlama ve haber içerikleri almayı onaylıyorum.

Skills of tomorrow

info@miuul.com