Pushkar Mandot Mengikuti
18 Agustus 2017 · 8 menit membaca · · Mendengarkan

Apa itu LightGBM, Bagaimana cara menerapkannya? Bagaimana cara menyempurnakan parameter?

Halo,

Machine Learning adalah bidang yang paling cepat berkembang di dunia. Setiap hari akan ada peluncuran banyak algoritma baru, beberapa di antaranya gagal dan beberapa mencapai puncak kesuksesan. Hari ini, saya menyentuh salah satu algoritma pembelajaran mesin paling sukses, Light GBM.

> *Apa yang memotivasi saya untuk menulis blog di LightGBM?*

Saat mengerjakan kompetisi ilmu data kaggle, saya menemukan beberapa algoritme yang kuat. LightGBM adalah salah satunya. LightGBM adalah algoritma yang relatif baru dan tidak memiliki banyak sumber bacaan di internet kecuali dokumentasinya. Sulit bagi pemula untuk memilih parameter dari daftar panjang yang diberikan dalam dokumentasi. Hanya untuk membantu geek baru, saya membuat blog yang indah ini.

Saya akan berusaha sebaik mungkin untuk menjaga blog ini tetap kecil dan sederhana karena menambahkan ratusan halaman informasi yang tidak relevan akan membingungkan Anda.

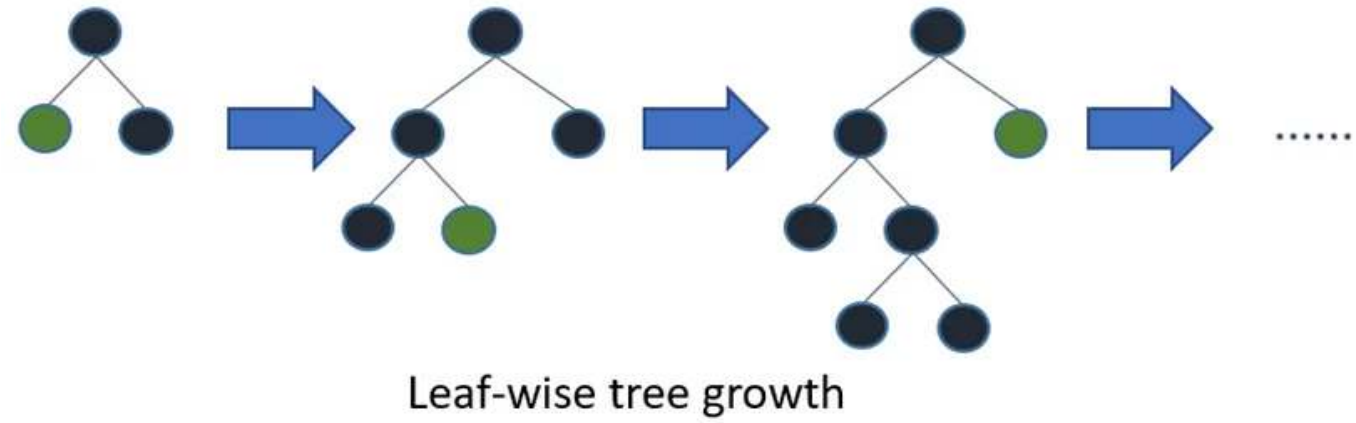
Apa itu GBM Ringan?

Light GBM adalah kerangka peningkatan gradien yang menggunakan algoritma pembelajaran berbasis pohon.

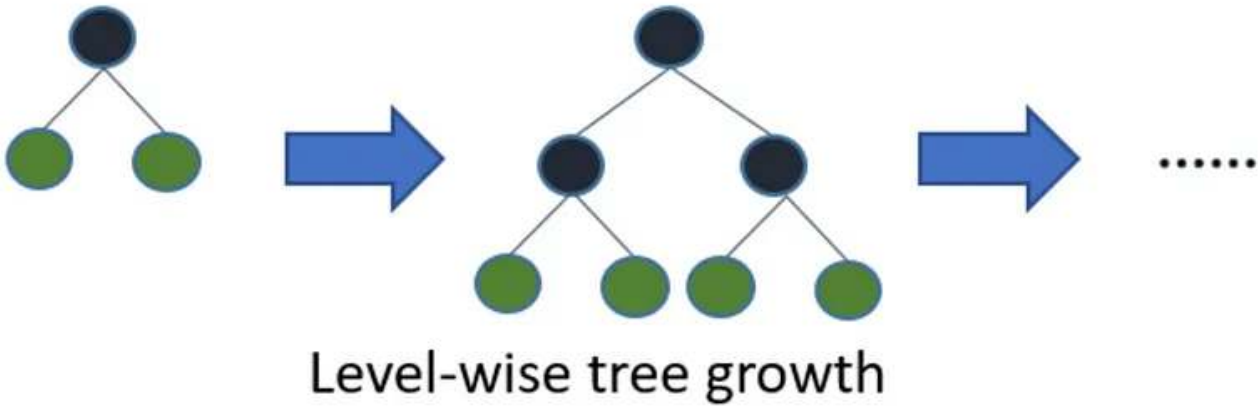
Apa bedanya dengan algoritma berbasis pohon lainnya?

Light GBM menumbuhkan pohon secara vertikal sementara algoritma lain menumbuhkan pohon secara horizontal yang berarti bahwa Light GBM menumbuhkan pohon **berdasarkan daun** sementara algoritma lain menumbuhkan secara level. Ini akan memilih daun dengan kehilangan delta maksimal untuk tumbuh. Saat menumbuhkan daun yang sama, algoritma Leaf-wise dapat mengurangi lebih banyak kerugian daripada algoritma level-wise.

Diagram di bawah ini menjelaskan penerapan LightGBM dan algoritme penguat lainnya.



Menjelaskan cara kerja LightGBM



Cara kerja algoritme penguat lainnya

> *Mengapa Light GBM semakin populer?*

Ukuran data meningkat dari hari ke hari dan menjadi sulit bagi algoritme ilmu data tradisional untuk memberikan hasil yang lebih cepat. Light GBM diawali dengan 'Ringan' karena kecepatannya **yang tinggi.** Light GBM dapat **menangani ukuran** data yang besar dan **membutuhkan memori yang lebih rendah untuk dijalankan** . Alasan lain mengapa Light GBM populer adalah karena **berfokus pada keakuratan hasil** . LGBM juga **mendukung pembelajaran GPU** dan dengan demikian ilmuwan data banyak menggunakan LGBM untuk pengembangan aplikasi sains data.

> *Bisakah kita menggunakan Light GBM di mana saja?*

Tidak, tidak dianjurkan untuk menggunakan LGBM pada kumpulan data kecil. Light GBM **peka terhadap overfitting** dan dapat dengan mudah menyesuaikan data kecil. Tidak ada ambang batas pada jumlah baris tetapi pengalaman saya menyarankan saya untuk menggunakannya hanya untuk data dengan 10.000+ baris.

> *Tadi sempat kita bahas konsep Light GBM, nah bagaimana implementasinya?*

Implementasi Light GBM itu mudah, yang rumit hanyalah penyetelan parameter. Light GBM mencakup lebih dari 100 parameter tetapi jangan khawatir, Anda tidak perlu mempelajari semuanya.

Sangat penting bagi pelaksana untuk mengetahui setidaknya beberapa parameter dasar GBM Ringan. Jika Anda dengan hati-hati mengikuti parameter LGBM berikut, saya yakin Anda akan menganggap algoritme yang kuat ini sangat mudah.

Mari kita mulai membahas parameter.

Parameter

> *Parameter Kontrol*

max_depth: Ini menjelaskan kedalaman maksimum pohon. Parameter ini digunakan untuk menangani model overfitting. Setiap kali Anda merasa bahwa model Anda overfitted, saran pertama saya adalah menurunkan max_depth.

min_data_in_leaf: Ini adalah jumlah minimum record yang dimiliki daun. Nilai default adalah 20, nilai optimal. Ini juga digunakan untuk mengatasi pemasangan

feature_fraction: Digunakan saat boosting Anda (dibahas nanti) adalah hutan acak. Fraksi fitur 0,8 berarti LightGBM akan memilih 80% parameter secara acak di setiap iterasi untuk membangun pohon.

bagging_fraction: menentukan fraksi data yang akan digunakan untuk setiap iterasi dan biasanya digunakan untuk mempercepat pelatihan dan menghindari overfitting.

early_stopping_round: Parameter ini dapat membantu Anda mempercepat analisis. Model akan berhenti berlatih jika satu metrik dari satu data validasi tidak membaik di babak early_stopping_round terakhir. Ini akan mengurangi iterasi yang berlebihan.

lambda: lambda menentukan regularisasi. Nilai tipikal berkisar dari 0 hingga 1.

min_gain_to_split: Parameter ini akan menjelaskan perolehan minimum untuk melakukan pemisahan. Ini dapat digunakan untuk mengontrol jumlah pemisahan yang berguna di pohon.

max_cat_group: Ketika jumlah kategorinya besar, menemukan titik pisah di atasnya dengan mudah terlalu pas. Jadi LightGBM menggabungkannya ke dalam grup 'max_cat_group', dan menemukan titik perpecahan pada batas grup, default:64

Parameter Inti

Tugas: Ini menentukan tugas yang ingin Anda lakukan pada data. Ini mungkin melatih atau memprediksi.

aplikasi: Ini adalah parameter yang paling penting dan menentukan penerapan model Anda, apakah itu masalah regresi atau masalah klasifikasi. LightGBM secara default akan menganggap model sebagai model regresi.

- regresi: untuk regresi
- biner: untuk klasifikasi biner
- multiclass: untuk masalah klasifikasi multiclass

boosting: menentukan jenis algoritme yang ingin Anda jalankan, default=gdbt

- gbdt: Pohon Keputusan Peningkatan Gradien tradisional
- rf: hutan acak
- dart: Putus sekolah bertemu dengan Pohon Regresi Aditif Berganda
- goss: Sampling Satu Sisi Berbasis Gradien

num_boost_round: Jumlah iterasi peningkatan, biasanya 100+

learning_rate: Ini menentukan dampak setiap pohon pada hasil akhir. GBM bekerja dengan memulai perkiraan awal yang diperbarui menggunakan keluaran dari setiap pohon. Parameter pembelajaran mengontrol besarnya perubahan estimasi ini. Nilai tipikal: 0,1, 0,001, 0,003…

num_leaves: jumlah daun dalam pohon penuh, default: 31

perangkat: default: cpu, juga bisa lewat gpu

Parameter metrik

metrik: sekali lagi salah satu parameter penting karena menentukan kerugian untuk pembuatan model. Di bawah ini adalah beberapa kerugian umum untuk regresi dan klasifikasi.

- mae: berarti kesalahan mutlak
- mse: kesalahan kuadrat rata-rata
- binary_logloss: kerugian untuk klasifikasi biner
- multi_logloss: kerugian untuk multi klasifikasi

> *parameter IO*

max_bin: itu menunjukkanjumlah maksimum bin yang memiliki nilai fiturakankeranjangdi dalam.

categorical_feature: Ini menunjukkan indeks fitur kategorikal. Jika fitur_kategori = 0,1,2 maka kolom 0, kolom 1 dan kolom 2 adalah variabel kategori.

abaikan_kolom: sama seperti categorical_features alih-alih menganggap kolom tertentu sebagai kategori, itu akan sepenuhnya mengabaikannya.

save_binary: Jika Anda benar-benar berurusan dengan ukuran memori file data Anda, tentukan parameter ini sebagai 'True'. Menentukan parameter benar akan menyimpan dataset ke file biner, file biner ini akan mempercepat waktu membaca data Anda untuk waktu berikutnya.

Mengetahui dan menggunakan parameter di atas pasti akan membantu Anda mengimplementasikan model. Ingat saya mengatakan bahwa penerapan LightGBM itu mudah tetapi penyetelan parameter itu sulit. Jadi mari kita mulai dengan implementasi dan kemudian saya akan memberikan ide tentang penyetelan parameter.

Penerapan

Memasang LGBM:

Menginstal LightGBM adalah tugas penting. Saya menemukan **ini** sebagai sumber terbaik yang akan memandu Anda dalam instalasi LightGBM.

Saya menggunakan Anaconda dan menginstal LightGBM di anaconda sangat mudah. Jalankan saja perintah berikut pada prompt perintah Anaconda Anda dan whoosh, LightGBM ada di PC Anda.

```
conda install -c conda-forge lightgbm
```

Himpunan data:

Data ini sangat kecil hanya 400 baris dan 5 kolom (khusus digunakan untuk tujuan pembelajaran). Ini adalah masalah klasifikasi dimana kita harus memprediksi apakah pelanggan akan membeli produk dari iklan yang diberikan di website. Saya tidak menjelaskan dataset karena dataset sudah cukup jelas. Anda dapat mengunduh dataset dari drive saya .

Catatan: Dataset bersih dan tidak ada nilai yang hilang. Tujuan utama di balik memilih data yang jauh lebih kecil ini adalah untuk menjaga agar hal-hal tersebut lebih sederhana dan mudah dipahami.

Saya berasumsi bahwa Anda semua tahu dasar-dasar python. Ikuti langkah-langkah preprocessing data, mereka cukup mudah tetapi jika Anda ragu, tanyakan kepada saya di komentar, saya akan menghubungi Anda secepatnya.

Pemrosesan awal data:

```
impor numpy sebagai np
impor matplotlib.pyplot sebagai plt
impor panda sebagai pd

# Mengimpor dataset
dataset = pd.read_csv('...input\\Social_Network_Ads.csv')
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values

# Memisahkan dataset menjadi set Pelatihan dan set Tes
dari sklearn.cross_validation import train_test_split
```

```
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size =
0.25, random_state = 0)

# Penskalaan Fitur
dari sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

Pembuatan model dan pelatihan:

Kami perlu mengonversi data pelatihan kami ke dalam format kumpulan data LightGBM (ini wajib untuk pelatihan LightGBM).

Setelah membuat dataset konversi, saya membuat kamus python dengan parameter dan nilainya. Keakuratan model Anda sepenuhnya bergantung pada nilai yang Anda berikan ke parameter.

Di blok kode terakhir, saya hanya melatih model dengan 100 iterasi.

```
impor lightgbm sebagai lgb

d_train = lgb.Dataset(x_train, label=y_train)

params = {}
params['learning_rate'] = 0,003
params['boosting_type'] = 'gbdt'
params['objective'] = 'binary'
params['metric'] = 'binary_logloss'
params['sub_feature'] = 0,5
params['num_leaves'] = 10
params['min_data'] = 50
params['max_depth'] = 10

clf = lgb.train(params, d_train, 100)
```

Beberapa hal yang perlu diperhatikan dalam parameter:

- Menggunakan 'biner' sebagai tujuan (ingat ini adalah masalah klasifikasi)
- Menggunakan 'binary_logloss' sebagai metrik (alasan yang sama, masalah klasifikasi biner)

- 'num_leaves'=10 (karena datanya kecil)
- 'tipe penguat' adalah gbdt, kami menerapkan peningkatan gradien (Anda dapat mencoba hutan acak)

Prediksi model:

kita hanya perlu menulis baris untuk prediksi.

Output akan menjadi daftar probabilitas. Saya mengonversi probabilitas menjadi prediksi biner yang menjaga ambang batas = 0,5

```
#Prediksi
y_pred=clf.predict(x_test)

#convert ke nilai biner

for i in range(0,99):
    if y_pred[i]>=.5: # menetapkan ambang batas ke .5
       y_pred[i]=1
    else:
       y_pred[i]=0
```

Hasil:

Kami dapat memeriksa hasil baik menggunakan matriks kebingungan atau langsung menghitung akurasi

Kode:

```
#Matriks kebingungan

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)

#Accuracy

from sklearn.metrics import accuracy_score
accuracy=accuracy_score(y_pred,y_test)
```

Screenshots of result:

```
In [15]: # Confusion Matrix of light GBM
         cm

Out[15]: array([[65,  3],
                [ 5, 2711)
```



```
In [18]: #Accuracy of Light GBM model
         accuracy

Out[18]: 0.92000000000000004
```

Accuracy Score

Many of you must be thinking that I used smaller dataset and still my model has 92% accuracy. Why there is no overfitting? The simple reason is I fine tuned model parameters.

So now let's jump into parameter fine tuning.

Parameter Tuning:

Data scientists always struggle in deciding when to use which parameter? and what should be the ideal value of that parameter?

Following set of practices **can be used to improve your model efficiency.**

1. **num_leaves:** This is the main parameter to control the complexity of the tree model. Ideally, the value of num_leaves should be less than or equal to 2^(max_depth). Value more than this will result in overfitting.
2. **min_data_in_leaf:** Setting it to a large value can avoid growing too deep a tree, but may cause under-fitting. In practice, setting it to hundreds or thousands is enough for a large dataset.
3. **max_depth:** You also can use max_depth to limit the tree depth explicitly.

For Faster Speed:

- Use bagging by setting `bagging_fraction` and `bagging_freq`
- Use feature sub-sampling by setting `feature_fraction`
- Use small `max_bin`

- Use `save_binary` to speed up data loading in future learning
- Use parallel learning, refer to parallel learning guide.

For better accuracy:

- Use large `max_bin` (may be slower)
- Use small `learning_rate` with large `num_iterations`
- Use large `num_leaves` (may cause over-fitting)
- Use bigger training data
- Try `dart`
- Try to use categorical feature directly

To deal with over-fitting:

- Use small `max_bin`
- Use small `num_leaves`
- Use `min_data_in_leaf` and `min_sum_hessian_in_leaf`
- Use bagging by set `bagging_fraction` and `bagging_freq`
- Use feature sub-sampling by set `feature_fraction`
- Use bigger training data
- Try `lambda_l1`, `lambda_l2` and `min_gain_to_split` to regularization
- Try `max_depth` to avoid growing deep tree

Conclusion:

I implemented LightGBM on multiple datasets and found that its accuracy challenged other boosting algorithms. From my experience, I will always recommend you to try this algorithm at least once.

I hope you guys enjoyed this blog and it was useful to all. I would request you you to give suggestion which will help to improve this blog.

Sumber: Dokumentasi Microsoft LightGBM

Terima kasih,

Pushkar Mandot

Pembelajaran mesin Penyesuaian Parameter Xgboost Lightgbm Parameter