LightGBM: высокоэффективный градиентный бустинг над деревьями решений

**Гаолин Ке, Ки Менг, Томас Финли, Тайфенг Вонг, Вэй Ченг, Вейдонг Ма, Кивей Йе, Тиян Лю**

Предисловие

Градиентный бустинг на деревьях решений – популярный алгоритм машинного обучения. Для него есть довольно много эффективных реализаций, наиболее успешными из которых являются XGBoost и pGBRT. Несмотря на то, что эти реализации содержат большое количество оптимизаций, их эффективность и масштабируемость все еще остаются неудовлетворительными для использования в больших пространствах признаков и для объемных данных. Главная причина заключается в том, что по каждому признаку необходимо исследовать все наблюдения для вычисления информационного выигрыша по всем возможным точкам расщепления, что является весьма вычислительно затратной процедурой. Для решения этой проблемы мы представим 2 новых техники: одностороннюю выборку на основе градиентов (*GOSS – Gradient-based One-Side Sampling*) и связывание взаимоисключающих признаков (*EFB – Exclusive Feature Bundling*). С помощью GOSS мы исключаем существенную часть наблюдений с малым градиентом, а оставшуюся часть используем для оценки информационного выигрыша. Мы докажем, что, поскольку наблюдения с большими градиентами играют более важную роль в вычислении информационного выигрыша, с помощью GOSS возможно получить довольно точную оценку информационного выигрыша, используя гораздо меньшее количество наблюдений. Применяя EFB, мы связываем взаимоисключающие предикторы (т.е. те, которые редко принимают ненулевые значения одновременно) для уменьшения количества используемых предикторов. Мы докажем, что поиск оптимального связывания взаимоисключающих признаков является NP-трудной задачей, однако жадный алгоритм позволяет получить довольно хороший коэффициент аппроксимации, тем самым эффективно сократив количество признаков без значительной потери точности при поиске точек расщепления. Мы назвали нашу новую реализацию градиентного бустинга над деревьями решений, использующую GOSS и EFB, LightGBM. Наши эксперименты на многочисленных общедоступных наборах данных показали, что предложенный алгоритм работает примерно в 20 раз быстрее обычных реализаций при почти одинаковой точности.

Введение

Градиентный бустинг на деревьях решений (GBDT) является широко используемым алгоритмом машинного обучения по причине его эффективности, точности и интерпретируемости. Градиентный бустинг позволяет добиться высочайших результатов в многочисленных задачах машинного обучения, таких, как многоклассовая классификация [2], прогнозирование кликов [3] и ранжирование [4]. В последние годы с появлением больших данных (как по количеству признаков, так и по количеству наблюдений) градиентный бустинг над деревьями решений столкнулся с новыми проблемами, особенно в части нахождения компромисса между точностью и эффективности. Чтобы вычислить информационный выигрыш по всем возможным точкам расщепления с помощью стандартных реализаций градиентного бустинга над деревьями решений, необходимо исследовать все наблюдения для каждого признака. По этой причине их вычислительная сложность будет пропорциональна количеству признаков и количеству наблюдений, что делает эти реализации весьма вычислительно затратными при работе с большими данными.

Простой способ решения этой проблемы заключается в том, чтобы попытаться каким-то образом уменьшить количество предикторов и наблюдений. Однако эта задача является весьма нетривиальной. Например, неясно, каким образом сформировать выборку наблюдений для градиентного бустинга над деревьями решений. Несмотря на то, что есть подходы, в которых предлагается случайный отбор наблюдений согласно их весам в целях ускорения процесса обучения бустинга [5, 6, 7], их нельзя напрямую применить к градиентному бустингу над деревьями решений, поскольку в градиентном бустинге над деревьями решений наблюдения вообще не имеют весов. В данной статье мы предложим две новые техники для достижения данной цели, описанные ниже.

**Односторонняя выборка на основе градиентов (GOSS).** Несмотря на то, что в градиентном бустинге наблюдения не имеют весов, можно заметить, что наблюдения с разными градиентами вносят различный вклад в процесс вычисления информационного выигрыша. В частности, согласно определению информационного выигрыша, наблюдения с большими градиентами (т.е. недообученные примеры) вносят больший вклад в информационный выигрыш. Поэтому, чтобы сохранить точность оценки информационного выигрыша, необходимо сохранять наблюдения с большими градиентами (т.е. брать наблюдения с градиентами выше заранее заданного порога или ориентироваться на верхние процентили) и случайным образом удалять наблюдения с маленькими градиентами. Будет доказано, что такой способ приводит к более точной оценке выигрыша по сравнению с равномерной случайной выборкой с той же частотой, особенно когда диапазон информационного выигрыша велик.

**Связывание взаимоисключающих признаков (EFB).** Обычно в реальных задачах, несмотря на присутствие большого количества признаков, признаковое пространство является довольно сильно разреженным, что дает нам возможность сократить количество признаков без больших потерь качества алгоритма. В частности, в разреженном пространстве признаков многие признаки (почти) уникальны, то есть редко принимают ненулевые значения одновременно. В качестве примера можно взять one-hot-кодирование признаков (например, представление слова для анализа текстов). Такие взаимоисключающие признаки можно связывать без потери качества. С этой целью нами был разработан эффективный алгоритм, в котором задача оптимального связывания сводится к задаче раскраски графа (назначением признаков в качестве вершин и добавлением ребер для каждых двух взаимоисключающих признаков) и решается жадным алгоритмом с постоянным коэффициентом аппроксимации.

Мы назвали новый алгоритм градиентного бустинга над деревьями решений, использующий GOSS и EFB, *LightGBM*. Наши эксперименты на многочисленных общедоступных наборах данных показали, что предложенный нами алгоритм работает примерно в 20 раз быстрее, при этом точность остается практически такой же.

Оставшаяся часть статьи организована следующим образом. Сначала в Разделе 2 будет приведен обзор алгоритмов градиентного бустинга над деревьями решений и связанных с ними работ. Затем, в Разделе 3 мы познакомимся с деталями метода GOSS, а в Разделе 4 – опишем метод EFB. Результаты наших экспериментов с LightGBM на общедоступных наборах данных приведены в Разделе 5. Наконец, Раздел 6 является заключительным.

2. Предварительное описание

2.1. Градиентный бустинг над деревьями решений и анализ его сложности

GBDT – это ансамбль деревьев решений, обученных последовательно. На каждой итерации GBDT обучает новое дерево по антиградиентам (также известным как остатки).

Основную часть вычислительных затрат градиентного бустинга над деревьями составляет обучение деревьев решений, а в обучении отдельного дерева самой вычислительно затратной частью является поиск наилучших точек расщепления в дереве. Один из самых популярных алгоритмов поиска точек расщепления – это алгоритм предварительной сортировки [8, 9], в рамках которого выполняется перебор всех возможных точек расщепления по отсортированным значениям каждого признака. Этот метод прост, способен найти наилучшие точки расщепления, но неэффективен как с точки зрения времени обучения, так и с точки зрения потребления памяти. Еще одним популярным алгоритмом является алгоритм на основе гистограммирования [10, 11, 12], показанный в Алгоритме 1. Вместо поиска точек расщепления по предварительно отсортированным значениям признаков алгоритм разбивает количественный признак на дискретные бины и использует эти бины для создания гистограмм признаков в ходе обучения. Поскольку алгоритм на основе гистограммирования более эффективен как с точки зрения потребления памяти, так и по скорости обучения, мы построим нашу работу на его основе.

Как показано в Алгоритме 1, алгоритм на основе гистограммирования находит наилучшие точки расщепления с помощью гистограмм признаков. Сложность такого алгоритма составляет при построении гистограмм и при поиске точек расщепления. Поскольку количество бинов обычно намного меньше количества наблюдений, гистограммирование более эффективно с точки зрения вычислительной сложности. Если мы сможем уменьшить количество наблюдений или количество признаков, то существенно ускорим обучение модели.

2.2. Связанные работы

В литературе описано довольно много реализаций градиентного бустинга над деревьями решений, например, XGBoost [13], pGBRT [14], реализация градиентного бустинга в библиотеке scikit-learn [15] и пакет gbm в R [16]. Реализация градиентного бустинга в scikit-learn и gbm в R основаны на алгоритме предварительной сортировки, а в pGBRT применяется гистограммирование. В XGBoost поддерживается как алгоритм предварительной сортировки, так и алгоритм на основе гистограммирования. Как показано в работе [13], XGBoost превосходит по скороcти остальные реализации, поэтому было решено использовать в наших экспериментах в качестве основы для сравнения именно его.

Обычным подходом при сокращении обучающей выборки является удаление некоторой части наблюдений. Например, в работе [5] наблюдения удалялись, если их веса были меньше заранее заданного порога. Стохастический градиентный бустинг, предложенный в работе [20], использует в каждой итерации случайную подвыборку для построения слабого классификатора. В работе [6] процент отбора наблюдений динамически меняется в зависимости от хода обучения. Однако, все эти работы, за исключением стохастического градиентного бустинга [20], основаны на алгоритме AdaBoost [21] и не могут быть непосредственно применены к градиентному бустингу над деревьями решений, поскольку в градиентном бустинге у наблюдений нет естественных весов. Хотя стохастический градиентный бустинг и можно применить в градиентном бустинге над деревьями решений, он обычно снижает точность и не является подходящий выбором.

По аналогии, для сокращения количества признаков общепринятой практикой является удаление слабых признаков [22, 23, 7, 24]. Обычно для этого используют анализ главных компонент или поиск наилучшей проекции. Однако все эти подходы в значительной мере основаны на предположении наличия у признаков значительной избыточности, что не всегда верно на практике (обычно признаки вносят свой уникальный вклад в прогноз, и удаление любого из них может, в определенной степени, повлиять на точность обучения).

Огромные массивы данных, использующиеся в реальных задачах, обычно довольно разрежены. Градиентный бустинг над деревьями решений с алгоритмом предварительной сортировки может уменьшить вычислительные затраты на обучение, игнорируя признаки с нулевыми значениями [13]. Напротив, у градиентного бустинга над деревьями решений, использующего гистограммирование, нет эффективных решений по оптимальной обработке разреженных данных. Причина заключается в том, что алгоритму на основе гистограммирования нужно извлекать значения бинов признака для каждого наблюдения, вне зависимости от того, является значение признака нулевым или нет (см. Алгоритм 1). Весьма желательно, чтобы градиентный бустинг над деревьями решений с гистограммированием был способен эффективно обрабатывать такие разреженные данные.

Для устранения ограничений предыдущих реализаций мы предложили две новые техники – односторонняя выборка, основанная на градиентах, и связывание взаимоисключающих предикторов. Более подробно о них будет рассказано в следующих разделах.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Алгоритм 1: Алгоритм на основе гистограммирования** | | | | |
| **Ввод:** *I*(обучающая выборка), *d*(максимальная глубина) | | | | |
| **Ввод:**  (пространство признаков) | | | | |
| *набор\_узлов* ← {0} ►узлы дерева на текущем уровне | | | | |
| *набор\_наблюдений* ← {{0, 1, 2, …}} ►индексы наблюдений в узлах дерева | | | | |
| **для** от 1 до **выполнять** | | | | |
|  | **для** *узла* в *наборе\_узлов* выполнять | | | |
|  | | *наблюдения, попавшие в узел* ← *набор\_наблюдений*[*узел*] | | |
|  | | **для** от 1 до **выполнять** | | |
|  | | | *H* ← новая гистограмма | |
|  | | | ► строить гистограмму | |
|  | | | для в *наблюдениях, попавших в узел* выполнять | |
|  | | | | бин ← *I*.f[k][j].бин |
|  | | | | *H*[бин].y ← *H*[бин].y + *I*.y[j] |
|  | | | | *H*[бин].n ← *H*[бин].n + 1 |
|  | | | Найти наилучшую точку расщепления по гистограмме *H* | |
|  | | | . . . | |
|  | | |  | |
|  | Обновить *набор\_наблюдений* и *набор\_узлов* в соответствии с наилучшими точками расщепления | | | |
|  | . . . | | | |

|  |  |
| --- | --- |
| **Алгоритм 2: Односторонняя выборка на основе градиентов** | |
| **Ввод:** *I*(обучающая выборка), *d*(количество итераций) | |
| **Ввод:** *a* (коэффициент отбора наблюдений с большими градиентами) | |
| **Ввод:** *b* (коэффициент отбора наблюдений с маленькими градиентами) | |
| **Ввод:** *loss* (функция потерь), *L* (базовый алгоритм) | |
| задаем список, в котором будем хранить модели, и константу *fact*, на которую будем умножать наблюдения с маленькими градиентами  models ← {}, fact ← | |
| получаем topN наблюдений с большими градиентами и randN наблюдений с маленькими градиентами:  topN ← *a* × len(*I*), randN ← *b* × len(*I*) | |
| **для** от 1 до **выполнять** | |
|  | вычислять прогнозы для всей обучающей выборки  preds ← models.predict(*I*) |
|  | вычислять градиенты, все веса равны 1  g ← *loss*(*I*, preds), w ← {1, 1, …} |
|  | сортировать наблюдения по абсолютным значениям градиентов  sorted ← GetSortedIndices(abs(g)) |
|  | в отсортированном наборе отбирать topN наблюдений с большими градиентами и формировать набор topSet  topSet ← sorted[1:topN] |
|  | из остальной части отсортированного набора наблюдений случайным образом отбирать randN наблюдений с маленькими градиентами и формировать набор randSet  randSet ← RandomPick(sorted[topN:len(*I*)], randN) |
|  | объединять наборы topSet и randSet в итоговый набор usedSet  usedSet ← topSet + randSet |
|  | умножать наблюдения с маленькими градиентами на вес *fact*  w[randSet] × = fact |
|  | строить новую модель  newModel ← L(*I*[usedSet], − g[usedSet], w[usedSet]) |
|  | добавлять модель в композицию  models.append(newModel) |

3. Односторонняя выборка на основе градиентов

В этом разделе будет предложен новый метод отбора наблюдений для градиентного бустинга над деревьями решений, позволяющий добиться хорошего баланса между уменьшением количества наблюдений и сохранением точности обученной модели.

3.1. Описание алгоритма

В AdaBoost вес наблюдения служит хорошим индикатором важности примера. Однако в градиентном бустинге над деревьями решений нет естественных весов наблюдений и, таким образом, методы отбора, предложенные для AdaBoost, не могут быть применены непосредственно. К счастью, в градиентном бустинге над деревьями решений градиент для каждого наблюдения дает нам полезную информацию, позволяющую осуществить отбор данных. Если наблюдение имеет маленький градиент, ошибка обучения для него невелика, и оно уже хорошо предсказывается моделью. Очевидная идея состоит в том, чтобы отбросить наблюдения с маленькими градиентами. Однако из-за этого изменится распределение данных, что может ухудшить точность обученной модели. Для устранения этой проблемы предлагается новый метод под названием односторонняя выборка на основе градиентов (GOSS – Gradient-based One-Side Sampling).

GOSS сохраняет все наблюдения с большими градиентами и выполняет случайный отбор наблюдений с малыми градиентами. Чтобы компенсировать влияние на распределение данных при вычислении информационного выигрыша, в рамках метода GOSS для наблюдений с маленькими градиентами вводится постоянный множитель (см. Алгоритм 2). В частности, в ходе процедуры GOSS сначала происходит сортировка наблюдения в соответствии с абсолютными значениями их градиентов и отбор первых наблюдений. Затем случайным образом отбираются наблюдений из оставшейся части данных. После этого в алгоритме GOSS при вычислении информационного выигрыша отобранные наблюдения с маленькими градиентами усиливаются умножением на константу . Поступая подобным образом, мы большее внимание уделяем плохо предсказываемым наблюдениям без значительного изменения исходного распределения данных.

3.2. Теоретический анализ

Градиентный бустинг использует деревья решений для поиска отображения пространства входов в пространство градиентов [1]. Предположим, имеется обучающий набор с независимыми одинаково распределенными наблюдениями , где – вектор размерности в пространстве . Антиградиенты функции потерь по прогнозам модели, получаемые на каждой итерации градиентного бустинга, обозначим как . В дереве решений производится разбиение по каждому узлу с помощью наиболее информативного признака (дающего наибольший информационный выигрыш). Для градиентного бустинга над деревьями решений информационный выигрыш обычно измеряется с точки зрения уменьшения дисперсии после разбиения, как показано ниже.

**Определение 1.** *Пусть будет обучающим набором для некоторого фиксированного узла дерева решений. Уменьшение дисперсии после разбиения признака в точке для этого узла определено как*

|  |  |
| --- | --- |
|  | , |

*где* и

.

Для признака алгоритм дерева решений выбирает и вычисляет наибольший выигрыш . Затем данные разбиваются по признаку в точке на левый и правый дочерние узлы.

В предложенном нами методе GOSS наблюдения обучающего набора сначала сортируются в соответствии с абсолютными значениями их градиентов, а затем первые наблюдений с большими градиентами сохраняются, что дает подмножество наблюдений . Затем из оставшейся части данных , состоящей из наблюдений с малыми градиентами, случайным образом формируется подмножество наблюдений размером . И, наконец, выполняется разбиение в соответствии с вычисленным уменьшением дисперсии для подмножества , т.е.

|  |  |
| --- | --- |
| , | (1) |

где , а коэффициент используется для нормирования суммы градиентов по обратно пропорционально размеру .

Таким образом, в рамках метода GOSS для определения точки расщепления вместо точной оценки , полученной по всем наблюдениям, используется оценка , вычисленная по подмножеству меньшего размера, что значительно снижает вычислительные затраты. Более того, теорема, приведенная ниже, доказывает, что GOSS не приводит к значительной потере точности и будет работать лучше, чем случайный отбор. В целях экономии места доказательство теоремы приводится в дополнительных материалах.

**Теорема 3.2** *Обозначим ошибку аппроксимации в методе GOSS как* и , . С *вероятностью, по меньшей мере, , получим*

|  |  |
| --- | --- |
| , | (2) |

*где*  и .

Согласно теореме, получаем следующие рассуждения. Во-первых, асимптотический коэффициент аппроксимации для GOSS равен . Если разбиение не слишком разбалансировано (т.е. и ), то в ошибку аппроксимации наибольший вклад будет вносить второй член Неравенства (2), который уменьшается до 0 в при . Это означает, что при большом количестве наблюдений аппроксимация будет довольно точной. Во-вторых, случайный отбор является частным случаем GOSS c . В большинстве случаев GOSS превосходит по точности случайный отбор при условии , что эквивалентно при .

Теперь проанализируем обобщающую способность метода GOSS. В алгоритме GOSS рассматривается ошибка обобщения , которая является разницей между выигрышем, вычисленным по отобранным наблюдениям обучающего набора в рамках метода GOSS, и фактическим выигрышем для соответствующего распределения. В результате получим, что . Таким образом, ошибка обобщения при использовании метода GOSS будет близка к ошибке обобщения вычисления по полному обучающему набору данных при условии достаточной точности коэффициента аппроксимации. С другой стороны, отбор будет увеличивать разнообразие базовых алгоритмов, которое, потенциально, способствует улучшению обобщающей способности [24].

4. Cвязывание взаимоисключающих признаков

В этом разделе предлагается новый метод эффективного уменьшения количества признаков.

|  |  |  |  |
| --- | --- | --- | --- |
| **Алгоритм 3: Жадное связывание** | | | |
| **Ввод:** *F*(признаки), *K*(порог, задающий максимально допустимую степень конфликтности) | | | |
| Создаем граф *G* | | | |
| сортируем признаки по степени и получаем упорядоченный список  searchOrder ← *G*.sortByDegree() | | | |
| задаем список, где будут храниться связки, и список  со значениями конфликтности в связках  bundles ← {}, bundlesConflict ← {} | | | |
| проходим по упорядоченному списку признаков и, если степень конфликтности меньше или равна порогу, то добавляем признак к существующей связке, а если степень конфликтности больше порога, то добавляем признак в качестве новой связки  **для** в **выполнять** | | | |
|  | needNew ← True  **для** от 1 до **выполнять** | | |
|  | | cnt ← ConflictCnt(bundles[j],*F*[i])  **если** + **то** | |
|  | | | bundles[j].add(*F*[i]), needNew ← False  break |
|  | **если** **то** | | |
|  | | Добавляем в качестве новой связки в список связок | |
|  | |  | |
| **Вывод:** | | список связок | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Алгоритм 4: Объединение взаимно исключающих признаков** | | | |
| **Ввод:** *numData*(количество наблюдений)  **Ввод:** *F*(одна связка взаимно исключающих признаков) | | | |
| инициализируем нулями binRanges и totalBin что это?  binRanges ← {0}, totalBin ← 0 | | | |
| что здесь делаем?  **для** в **выполнять** | | | |
|  | что здесь делаем?  totalBin += f.numBin  а здесь?  binRanges.append(totalBin) | | |
| что здесь делаем?  newBin ← new Bin(numData) | | | |
| проходим по всем наблюдениям  **для** от 1 до *numData* **выполнять** | | | |
|  | что здесь делаем?  newBin[i] ← 0  что здесь делаем?  **для** от 1 до  **выполнять** | | |
|  | | что здесь делаем?  **если** **то** | |
|  | | | newBin[i] ← *F*[j].bin[i] + binRanges[j] |
| **Вывод:** | |  | |

**Подсказка**



В подсказке вы можете увидеть, что признак1 и признак2 являются взаимно исключающими. Чтобы получить непересекающие бины, мы добавляем связку той же размерности, что и признак 1, в признак2. Это гарантирует нам, что наблюдения с ненулевыми значениями связанных признаков (признака 1 и признака 2) окажутся в разных бинах. Бины с 1 по 4 связки признаков содержат наблюдения с ненулевыми значениями признака 1, а бины 5, 6 содержат наблюдения с ненулевыми значениями признака 2.

Как будет выглядеть связка вот с такими признаками?

|  |  |  |  |
| --- | --- | --- | --- |
| **Признак 1** | **Признак 2** | **Признак 3** | **Связка** |
| 0 | 3 | 0 |  |
| 1 | 0 | 0 |  |
| 0 | 2 | 0 |  |
| 0 | 1 | 0 |  |
| 2 | 0 | 0 |  |
| 0 | 0 | 1 |  |

**А в этом случае?**

|  |  |  |  |
| --- | --- | --- | --- |
| **Признак 1** | **Признак 2** | **Признак 3** | **Связка** |
| 0 | 3 | 0 |  |
| 1 | 0 | 0 |  |
| 0 | 2 | 0 |  |
| 0 | 0 | 1 |  |
| 2 | 1 | 0 |  |
| 0 | 0 | 1 |  |

Данные с высокой размерностью обычно являются очень разреженными. Разреженность пространства признаков позволяет разработать подход к сокращению числа признакам, практически не дающий ухудшения качества модели. В частности, в разреженном пространстве признаков многие признаки являются взаимоисключающими, т. е. они никогда не принимают ненулевые значения одновременно. Мы можем смело связывать взаимоисключающие признаки в один (который мы называем *связкой взаимоисключающих признаков* – *exclusive feature bundle*). С помощью тщательно проработанного алгоритма исследования признаков возможно выполнить гистограммирование признаков на основе связок признаков, как это выполнялось для отдельных признаков. Таким образом, сложность построения гистограммы изменяется с до , где .Таким образом, можно значительно ускорить обучение градиентного бустинга над деревьями решений без ущерба для точности. Ниже будет детально показано, как именно это сделать.

Есть еще две задачи, которые необходимо решить. Первая задача заключается в определении того, какие признаки необходимо объединять. Вторая задача состоит в том, каким образом сконструировать связку.

**Теорема 4.1.** *Задача разбиения признаков на наименьшее количество связок признаков является NP-трудной.*

*Доказательство.* Сведем задачу раскраски графа [25] к нашей задаче сокращения количества признаков. Поскольку задача раскраски графа является NP-трудной, мы можем сделать вышеприведенный вывод.

Условимся, что дан некоторый образец задачи раскраски графа . Сформулируем наш вариант задачи следующим образом. Возьмем каждую строку матрицы инцидентности в качестве признака и получим наш вариант задачи с признаками. Легко увидеть, что в нашей задаче связка взаимоисключающих признаков соответствует набору вершин одного цвета и наоборот.

Если говорить о первой задаче, в Теореме 4.1 было доказано, что поиск стратегии оптимального связывания является NP-трудной задачей, что указывает на невозможность найти точное решение за полиномиальное время. Чтобы найти хороший алгоритм аппроксимации, сначала сведем задачу оптимального связывания к задаче раскраски графа, взяв признаки в качестве вершин и добавляя ребра для каждых двух признаков, если они не являются взаимоисключающими, а затем используем жадный алгоритм (с постоянным коэффициентом аппроксимации), который может дать достаточно хорошее решение задачи раскраски графа для получения связок признаков. Далее, обратим внимание на наличие довольно большого количества признаков, которые, хоть и не являются на 100% взаимоисключающими, но также редко принимают ненулевые значения одновременно. Если наш алгоритм допускает небольшую степень конфликтности (т.е. небольшое количество строк, где могут быть одновременно ненулевые значения), мы можем получить еще меньшее количество связок признаков и еще больше повысить вычислительную эффективность. Сделав простые расчеты, видим, что наличие случайных конфликтов для небольшой доли значений признаков будет влиять на точность обучения не более чем на (см. Приложение 2.1 в дополнительных материалах), где – это максимальная степень конфликтности в каждой связке. Таким образом, если выбрать относительно небольшое значение , мы сможем достичь хорошего баланса между точностью и вычислительной эффективностью.

Основываясь на вышеприведенных рассуждениях, мы предлагаем алгоритм для связывания взаимоисключающих признаков, структура которого показана в Алгоритме 3. Сначала построим граф с взвешенными ребрами с весами, соответствующими общей степени конфликтности между признаками. Затем, отсортируем признаки по их значению в порядке убывания. Наконец, исследуем каждый признак в упорядоченном списке и либо присоединим его к уже существующей связке с небольшой степенью конфликтности (регулируется значением ), либо создадим новую связку. Сложность Алгоритма 3 составляет *О(количество )* и выполняться он будет только один раз перед обучением. Эта сложность приемлема в ситуации, когда количество признаков не очень велико, но может быть по-прежнему слишком высока для работы с миллионами признаков. Для дальнейшего повышения скорости предлагается еще более эффективная стратегия упорядочивания без построения графа: упорядочивание по количеству ненулевых значений, что схоже с упорядочиванием по степеням, поскольку большее количество ненулевых значений обычно приводит к большей вероятности конфликтов. Поскольку мы лишь изменяем стратегии упорядочивания Алгоритма 3, детальное описание нового алгоритма опускается, чтобы избежать дублирования.

Для решения второй задачи необходим хороший способ объединения признаков в одной связке, чтобы уменьшить соответствующую сложность обучения. Ключевая мысль заключается в том, чтобы гарантировать возможность идентификации значений исходных признаков по значениям связок признаков. Поскольку алгоритм на основе гистограммирования вместо непрерывных значений признака хранит дискретные бины, возможно сконструировать связку признаков, позволив значениям взаимоисключающих признаков располагаться в разных бинах. Это можно сделать, добавив постоянное смещение к исходным значениям признаков. Предположим, имеются два признака, объединенные в связку. Первоначально признак A принимает значение [0, 10], а признак B принимает значение [0, 20). После добавления константы смещения 10 к значениям признака B обновленный признак принимает значения [10, 30]. После этого можно безопасно объединить признаки A и B и использовать связку признаков с диапазоном [0, 30] для замены оригинальных признаков A и B. Алгоритм подробно показан в Алгоритме 4.

Алгоритм EFB способен связывать большое количество взаимно исключающих признаков в гораздо более плотные признаки, что позволяет эффективно снизить количество ненужных вычислений для нулевых значений признаков. Фактически, возможно оптимизировать и базовый алгоритм гистограммирования, игнорируя нулевые значения признаков и записав данные с ненулевыми значениями по каждому признаку в таблицу. За счет поиска данных в этой таблице, вычислительная стоимость построения гистограммы для признака изменится с на . Однако такой метод потребует дополнительных затрат памяти и вычислительных мощностей для хранения таблиц по каждому признаку в ходе построения дерева. Мы реализовали эту оптимизацию в LightGBM в виде базовой функции. Стоит обратить внимание, что эта оптимизация не конфликтует с EFB, и мы по-прежнему можем использовать его для разреженных связок.