

# Онлайн оптимизация параметров рекомендательной системы

Александр Лактионов, МФТИ, ФИВТ, 497

Февраль, 2018

## 1 Задача

### 1.1 Постановка задачи

Рекомендательные системы могут быть основаны на алгоритмах машинного обучения, бизнес-правилах и их комбинации. В большинстве случаев процесс создания модели, которая предсказывает интересные пользователю объекты, состоит из трёх частей:

1. Подготовка, фильтрация и обработка данных
2. Обучение модели
3. Предсказание рекомендаций для пользователя

В модели может быть большое количество гиперпараметров – переменных, значения которых не оптимизируются в процессе обучения. Такие переменные делятся на два типа: те, которые подбираются до и после обучения модели.

В случае оффлайн-обучения раз в сутки, онлайн-оптимизация гиперпараметров для обучения не представляет большого интереса – новые точки в пространстве параметров будут проверяться очень долго. Параметры, которые можно изменить в процессе предсказания, напротив, могут быть оптимизированы в процессе работы алгоритма.

В данной работе рассмотрена оптимизация гиперпараметров, которые подбираются в процессе предсказания рекомендаций.

### 1.2 Значимость задачи






Рекомендательные системы используются в большом количестве сервисов и помогают пользователям находить ту информацию, которая им интересна. Рекомендации особенно полезны в сфере продаж в интернете, так как позволяют повысить конверсию в целевое действие.

В данной работе рассматривается рекомендательная система для популярного сайта объявлений об аренде и продаже недвижимости в России. Детали реализации данного подхода изложены в тезисах [1]

### 1.3 Применение в рекомендательной системе

Рассматривается блок рекомендаций «Похожие объекты», расположенный на странице объявления, содержащий 5 объявлений, интересных пользователю и связанных с рассматриваемым объектом на странице

Похожие предложения

				
3-комн.кв, 61 м² Астраханская область, Астрахань, улица Чело... 1 900 000 Р	3-комн.кв, 68 м² Астраханская область, Астрахань, улица Раск... 1 800 000 Р	3-комн.кв, 83/51/9 м², этаж 2/12 Астраханская область, Астрахань, Румынская ... 2 390 000 Р	2-комн.кв, 46/33/6 м², этаж 5/5 Астраханская область, Астрахань, улица Нико... 1 550 000 Р	2-комн.кв, 45 м², этаж 3/3 Астраханская область, Астрахань, улица Богд... 1 700 000 Р

#### 1.3.1 Бизнес-правила

В случае пользовательских сервисов нередко возникают продуктовые требования, выполнить которые невозможно, ограничившись только работой с моделью. Некоторые из подобных правил:

- Не рекомендовать объекты из других категорий недвижимости (например, продажу новостройки к аренде офиса)
- Не рекомендовать объекты из других регионов
- Не рекомендовать объекты без фотографий

Подобные правила параметризуются индикаторами их использования.

#### 1.3.2 Информация про контент

Нередко про объекты известна дополнительная информация, которую можно использовать для модификации функции скоринга перед определением 5 наиболее интересных объектов.

Эта информация можно обновляться очень часто и может быть не учтена во время обучения модели:

- Оценка объекта и аккаунта владельца модераторами сайта
- Популярность объекта

#### 1.3.3 Влияние контекста

При просмотре страницы объявления пользователь выражает свой интерес к данному объекту. Однако помимо этого есть много информации про самого пользователя, его поведение и просмотренные им объекты в прошлом.

- Вес истории пользователя
- Вес информации про просматриваемый объект

### 1.3.4 Технические параметры

В рассматриваемой рекомендательной системе используется поисковый движок Elasticsearch, в котором документы хранятся распределённо и по шардам.

При параллельном поиске похожих на историю пользователя или объект объявлений существует возможность досрочно прекратить запрос в рамках шарда, как только внутри него было найдено определенное количество документов. По умолчанию этот параметр равен бесконечности, однако если найти приемлимое значение, скорость работы всей системы можно увеличить в несколько раз.

- es.search: terminate\_after

Значение оптимизируемой метрики можно сделать зависимым от времени работы алгоритма. Также возможно указать определенный интервал для значений этого параметра: естественно, что если он влияет на целевую метрику (конверсию), то монотонно. Значит, если при допустимых значениях, например, в интервале  $[10^2, 10^6]$  он не будет стремиться принять наибольшее значение, можно зафиксировать его внутри этого интервала. Стоит также попробовать пошевелить границы допустимых значений.

## 2 Обзор существующих решений (TBD)

### 2.1 Многорукий бандит

#### 2.1.1 Семплирование Томпсона

#### 2.1.2 Плюсы подхода

#### 2.1.3 Минусы подхода

### 2.2 Оптимизация функций от многих переменных из матанализа

#### 2.2.1 Плюсы подхода

#### 2.2.2 Минусы подхода

### 2.3 Обучение с подкреплением

#### 2.3.1 Плюсы подхода

#### 2.3.2 Минусы подхода

## 3 Бейзлайн

В качестве бейзлайна рассматривается батчевый многорукий бандит. Максимизируемая метрика - доля пользователей, которые хотя бы раз кликнули на блок рекомендаций.

Каждый набор значений гиперпараметров представляет из себя модель, а перед предсказанием рекомендательная система выбирает, какая модель будет использована.

Он состоит из нескольких частей:

1. Регулярный пересчёт весов моделей
2. Алгоритм выбора модели при предсказании

Код бейзлайна выложен в репозитории [2]

### 3.1 Батчевый пересчёт весов моделей

#### 3.1.1 Метрика

В рамках батча (1 час) множество моделей конечно. Раз в батч запускается алгоритм, который анализирует целевую метрику по срезам моделей, работающих на предыдущем батче.

Трафик по моделям следующего батча распределяется пропорционально конверсиям соответствующих моделей в клик по рекомендательному блоку.

Пусть

$\{user_1, \dots, user_n\}$  - посетившие сайт пользователи,

$Models = \{model_1, \dots, model_m\}$  - рассматриваемые модели,

$UsersShown(model_j) = \{i \mid user_i \text{ просмотрел рекомендации от } model_j\}$ ,

$UsersClicked(model_j) = \{i \mid user_i \text{ кликнул на рекомендации от } model_j\}$ ,

Очевидно, что  $UsersClicked(model_j) \subset UsersShown(model_j)$ .

Тогда значение метрики для модели  $model_j$

$$sticked\_users(model_j) = \frac{\|UsersClicked(model_j)\|}{\|UsersShown(model_j)\|}$$

Эта метрика также известна как  $1 - bounce\_rate$  по кликам [6]. Она и будет оптимизироваться далее.

#### 3.1.2 Веса моделей

После пересчёта батча для модели  $model_j$  известно значение  $sticked\_users(model_j)$ .

Пусть  $metrics\_sum = \sum_{j=1}^m stucked\_users(model_j)$

Определим следующий вектор:

$$models\_weights = [\frac{sticked\_users(model_1)}{metrics\_sum}, \dots, \frac{sticked\_users(model_m)}{metrics\_sum}].$$

Из определения следует, что  $\sum_{k=1}^m models\_weights[k] = 1$ .

### 3.1.3 Сглаживание метрики

При недостатке данных для новых моделей значение  $\|UsersShown\|$  может быть очень мало. Если модель плохая, то такая статистика при определенных обстоятельствах может принять слишком большое значение и за следующий батч принесёт много потерь метрике рекомендательной системы.

Распространён подход, когда в вычисление значения метрики добавляется сглаживание, чтобы уменьшить её дисперсию при малом количестве наблюдений [7]

В бейзлайне используется простой способ сглаживания: если значение  $\|UsersShown\| \leq 10^4$ , то к знаменателю прибавляется сглаживание:

$$UsersShown := UsersShown \sqcup \{mock\_user_1, \dots, mock\_user_{10^3}\}$$

Таким образом, если при подсчёте метрики для  $model_j$  применилось сглаживание, то значение  $sticked\_users(model_j)$  уменьшается (увеличивается его знаменатель).

## 3.2 Выбор модели перед предсказанием рекомендаций

Перед тем, как рекомендательная система начнёт предсказывать, выбирается модель.

### 3.2.1 Хранение весов моделей

После пересчёта каждого батча в базе данных хранится распределение весов моделей -  $models\_weights$ .

Чтобы не делать запрос к базе данных при построении каждой рекомендации, микросервис рекомендаций делает запрос через некоторое время после пересчёта батча и обновляет вектор весов моделей.

### 3.2.2 Распределение

Можно рассмотреть вектор  $models\_weights$  как параметры дискретного распределения, тогда

$$P(model_j) = models\_weights[j]$$

Таким образом, трафик между моделями распределяется пропорционально значениям метрик, полученных ими на предыдущем батче.

## 3.3 Недостаток данных

Чтобы запустить новую модель, по которой еще нет статистики, нужно задать ей какой-то вес.

В бейзлайне используется небольшое количество моделей, поэтому приемлемое значение для минимального веса - 0.05. Тогда каждая модель будет получать как минимум 5% трафика.

В отличие от стандартных многоруких бандитов, данная реализация упрощена и использует только exploitation. Такое упрощение сделано в силу того, что технически сложно получить мгновенный негативный отклик на рекомендацию. Exploration здесь есть только для моделей, по которым собрано недостаточно данных.

В таком случае необходимо следить за весами моделей и удалять те, которые в течение продолжительного времени имеют вес, не сильно превышающий 0.05

### 3.4 Метрика для оптимизатора

Чтобы оценить, насколько хорош оптимизатор параметров, необходимо на что-то опираться. Разные оптимизаторы нельзя сравнивать друг с другом, если они запущены не параллельно. Однако если начать устраивать А/Б-тесты оптимизаторов, можно прийти к рекурсии и начать делать оптимизаторы оптимизаторов и так далее до бесконечности. К тому же, технически это достаточно сложно реализовать.

Поэтому сравним результат с обычным А/Б-тестом с распределением трафика 50%/50% по двум моделям. Посчитаем приблизительно, какая конверсия могла бы быть при таком тесте для двух моделей.

Пусть две модели отработали один батч и известны

$UsersShown(model_j), UsersClicked(model_j)$  для  $j \in [1, 2]$ , а итоговая конверсия по всей рекомендательной системе составила  $total\_sticked\_users$ .

Обозначим множество всех пользователей, которым были показаны рекомендации как

$$AllUsersShown = \cup_{j=1}^2 UsersShown(model_j),$$

Рассмотрим величину

$$AB\_sticked\_users = \frac{\|AllUsersShown\| \cdot stuck\_users(model_1) \cdot 0.5 + \|AllUsersShown\| \cdot stuck\_users(model_2) \cdot 0.5}{\|AllUsersShown\|}.$$

Значение  $AB\_sticked\_users$  - величина ожидаемой конверсии в целом по рекомендательной системе, если бы был проведён А/Б-тест двух моделей с распределением 50%/50%.

В качестве метрики оптимизатора будем использовать меру того, насколько он увеличивает целевую метрику в целом по сравнению с равномерным параллельным тестированием.

$$optimizer\_metric(optimizer) = \frac{AB\_sticked\_users}{total\_sticked\_users} \cdot 100\%$$

Эта метрика для оптимизаторов легко обобщается и для большего количества моделей.

### 3.5 Результаты

На графике (1) результатов работы бейзлайна на сайте - целевая метрика по моделям и в целом по всей рекомендательной системе.

В качестве  $model_2$  использовался алгоритм, применяющий к результатам рекомендаций бизнес-правила:

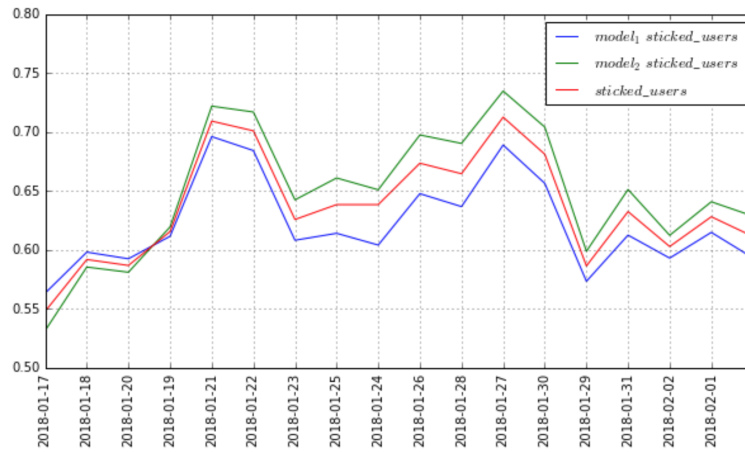
- Не рекомендовать объекты из других категорий недвижимости (например, продажу новостройки к аренде офиса)
- Не рекомендовать объекты из других регионов

$model_1$  - такой же алгоритм, но без этих правил.

Таким образом,  $model_1 = model(0, 0)$ ,  $model_2 = model(1, 1)$ ,

где аргументы модели - индикаторы использования соответствующих бизнес-правил.

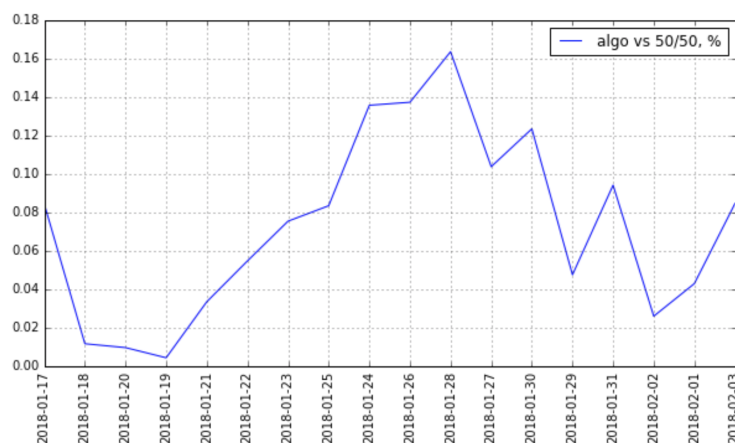
Срез данных за 18 дней:



Чем больше относительная разница между моделями - тем больше выгода использования оптимизатора из бейзлайна, что и ожидалось.

На графике (2) - метрика оптимизаторов  $optimizer\_metric(baseline)$

#### 4 ЭВРИСТИЧЕСКАЯ ОПТИМИЗАЦИЯ С ПОДБОРОМ НОВЫХ МОДЕЛЕЙ (TBD)8



Разница бейзлайна с простым А/Б-тестом совсем незначительная: порядка 0.1%. Это происходит по причине того, что у двух моделей очень похожие конверсии, хотя одна из них стабильно немного лучше другой. Из-за маленькой разницы распределение весов по моделям близко к равномерному.

#### 4 Эвристическая оптимизация с подбором новых моделей (TBD)

#### 5 Методы оптимизации функций многих переменных из матанализа (TBD)

#### 6 Оптимизация с помощью обучения с подкреплением (TBD)

#### 7 Результаты и выводы (TBD)

#### 8 Ссылки

- [1] [p.86, abitu.net/public/admin/mipt-conference/FPMI.pdf](http://p.86.abitu.net/public/admin/mipt-conference/FPMI.pdf)
- [2] [github.com/alexlokotochek/multiarmed-bandit](https://github.com/alexlokotochek/multiarmed-bandit)