Пара слов о MLR

И. Куралёнок

Яндекс

СПб, 2014

1. Задача MLR
2. Метрики
   - Кренфилдский подход
   - Попарный подход
   - Настройка по пользовательским данным
3. Обучение
   - Факторы ранжирования
   - Особенности оптимизации
4. Эксплуатация
5. MLR в поиске

План

1. Задача MLR
2. Метрики
   - Кренфилдский подход
   - Попарный подход
   - Настройка по пользовательским данным
3. Обучение
   - Факторы ранжирования
   - Особенности оптимизации
4. Эксплуатация
5. MLR в поиске

Формальная постановка

По большому счету хотим такого:
Нужно найти сколько-то лучших элементов из большого множества.

$$r : \Gamma \to \mathbb{R}^n$$

$$\pi_r(s) = \{x_i \in s\}_1^{|s|} : r(x_i) \leq r(x_j) \Leftrightarrow i < j$$

$$\max_r \mu_{s \in 2^\Gamma}(\mathcal{M}(\pi_r(s))$$

А как можно еще?

Задачи где применяется MLR

- Поиск
- Рекомендательные системы
- Реклама
- etc.

План

1. Задача MLR
2. Метрики
   - Кренфилдский подход
   - Попарный подход
   - Настройка по пользовательским данным
3. Обучение
   - Факторы ранжирования
   - Особенности оптимизации
4. Эксплуатация
5. MLR в поиске

Эксперты и ранжирование

Как понять какое ранжирование верно?

Кренфилдский подход предлагает разбить задачу на несколько частей:

1. Определить классы эквивалентности J в Γ
2. Составить выборку s_j
3. Всем значимым элементам s_j назначить классы
4. Агрегировать оценки J в общую метрику $\mathcal{M}$

Методы агрегации экспертных оценок

По большому счету все они похожи на такое:

$$\mathcal{M}(s) = \sum_i d(i) g(J(x_i))$$

Воспрос только в том, как устроены дискаунт d и вес g.

Методы агрегации экспертных оценок

По большому счету все они похожи на такое:

$$\mathcal{M}(s) = \sum_i d(i) g(J(x_i))$$

Воспрос только в том, как устроены дискаунт d и вес g.
Попробуйте сформулировать p10

Методы агрегации экспертных оценок

По большому счету все они похожи на такое:

$$\mathcal{M}(s) = \sum_i d(i) g(J(x_i))$$

Воспрос только в том, как устроены дискаунт d и вес g.
Попробуйте сформулировать ndcg

О чем надо не забыть

- Сбор оценок — очень непростая и дорогая деятельность
- Оценки шумят, для того, чтобы уменьшить уровень шума можно задать правила оценки
- Выборка $\{s_j\}$ отражает распределение над 2^Γ, которое часто зависит от времени
- Оценки могут быть неполными и с этим надо что-то делать
- При подобном подходе у эксперта нужно сложить понимание о контексте

Парные оценки

Кренфилдский подход хороший, но “правила игры составлять тяжело”, вводить в контекст не хочется.
Уменьшить вес инструкции можно парными оценками:

1. Составить выборку s_j
2. Из s_j надергаем пар $(u, v)_t$ по закону $\mathcal{P}$
3. Оценим пары по шкале
4. Посчитаем

$$LL(r, s) = \sum_t \begin{aligned} & I\{x_u > x_v\} \log p(x_i > x_j | r) \\ & + I\{x_u = x_v\} \log p(x_i = x_j | r) \\ & + I\{x_u < x_v\} \log p(x_i < x_j | r) \end{aligned}$$

Свойства парных оценок

- Играем не на метрику $\mathcal{M}$
- Инструкция короче $\Rightarrow$ меньше менять, легко писать с 0
- Оценки примерно такие же по времени
- Оценок надо сильно больше
- Непонятно как делать $\mathcal{P}$

Когда применять парные оценки

- Когда хотите, чтобы эксперты больше работали мозгом, а не руками
- Есть время
- Все “низко висящие фрукты” уже собраны

Списочные оценки

Можно рассматривать ситуацию, когда пользователь создает идеальный список. Но эта задача сводится к поточечной, где в качестве точки выступает весь список.

Предпосылки

+ Этих данных очень много
+ Данные равномерны по популяции
– Холодный старт
– Смещены в аттрактивность
– Можно поразному интерпретировать одно и тоже поведение пользователя
– Поведение неполно
– Положительная обратная связь по факторам от поведения
– Аутентификация пользователя
– Нет универсального подхода к построению $\mathcal{M}$

Что нужно, чтобы использовать пользовательские оценки

В отличии от предыдущих подходов к оценке, где нужна была только система сбора экспертных оценок, сложная в основном с точки зрения моделирования, для использования пользовательского поведения нужно много техники:

- Пользователи :)
- Система сбора пользовательского поведения (пользовательские логи)
- Развитая система логгирования поведения системы
- Механизмы объединения и верификации полученных логов

Какие оценки лучше?

Все подходы к оценке обладают своими преимуществами и недостатками, поэтому:

Классно делать систему, где используются все подходы к оценке $\Rightarrow$ получаем более точное понимание истины.

План

1. Задача MLR
2. Метрики
   - Кренфилдский подход
   - Попарный подход
   - Настройка по пользовательским данным
3. Обучение
   - Факторы ранжирования
   - Особенности оптимизации
4. Эксплуатация
5. MLR в поиске

Виды

По зависимостям:

- Одиночные $f(x)$
- Выборочные $f(s)$
- Контекстные $f(x, s)$
- Зависимые от времени $f(x, s, t)$

Технически:

- Бинарные
- Небинарные
- Категориальные

Первичная обработка

Нужно привести все факторы в форму, удобную для оптимизации. Этот процесс зависит от метода оптимизации и решающей функции. Для линейной _можно_ так:

1. Берем бинарные, и категориальные, по ним бьем обучающее множество до разумных пределов (или через регуряризацию)
2. Небинарные whiten'им в $\bar{f}(x, s)$
3. Строим решающую функцию бложно типа $\sum_k b_k(x, s)\beta_k^T \bar{f}(x, s)$

Метафакторы

Часто встречаются факторы, которые работают только не некоторых точках. В этом случае можно расширить выборку, обучить метамодели на этой выборке и использовать результаты обучения в качестве факторов финального.

Сбор значений факторов

При наличии зависящих от времени факторов — это большая проблема.

Разработка новых факторов

Есть набор факторов $f_{i_1}^{n}$ хотим придумать фактор f_{n+1}, который будет полезен. Есть одна проблема, нам надо на самом деле такое:

$$f_{n+1}^{*} : \mu_{\Gamma}\left(\mathcal{M}(\arg\max_{r \in F(\{f_i\}_{\mathbf{1}}^{n+\mathbf{1}})} T(r, X))\right) > \mu_{\Gamma}\left(\mathcal{M}(\arg\max_{r \in F(\{f_i\}_{\mathbf{1}}^{n})} T(r, X))\right)$$

Обеспечить такого мы не можем, поэтому будем искать такие факторы, которые являются f^* с вероятностью не меньше $1 - \alpha$

Метрика не всегда целевая функция

Если мы понимаем чего хотим: $\mathcal{M}(F)(X)$ (линейка позволяющая измерить конкретное решение), то задачу оптимизации можно переписать так:

$$\max_{\mathcal{T}} \mathcal{M}\left(\arg\max_{F} \mathcal{T}(F, L)\right)(T)$$

Если выборка не смещена по параметрам оптимизации, то K.O. говорит нам:

$$\mathcal{M} \equiv \arg\max_{\mathcal{T}} \mathcal{M}\left(\arg\max_{F} \mathcal{T}(F, L)\right)(T)$$

Однако, все не так просто.

Используемые классы оптимизаций

Точечные: имеем оценку в точке, хотим ее приблизить напрямую, с помощью r, надеемся на независимость от s

Парные: каким-то образом получаем парные оценки, далее приближаем LL правильного порядка каждой пары

Списочные: имеем идеальный список, вводим функцию расстояния от одного π до другого. Минимизируем расстояния до идеальных.

Сведение списочных к попарным

Списочные не работают, так как очень мало точек (списков). Поэтому обычно их разбирают на пары.

$$\arg\max_r \sum_i \sum_j w_{ij} \log \left(\frac{1}{1 + e^{-1^{I\{x_i > x_j\}}(r(x_i) - r(x_j))}} \right)$$

Проблема современности — найти правильные w_{ij}

План

1. Задача MLR
2. Метрики
   - Кренфилдский подход
   - Попарный подход
   - Настройка по пользовательским данным
3. Обучение
   - Факторы ранжирования
   - Особенности оптимизации
4. Эксплуатация
5. MLR в поиске

Фильтрация

Нам никогда не нужен полный π. Нам, достаточно частичного (в смысле только топ q результатов). Поэтому мы хотим разбить задачу на 2:

- Найти первых $p >> q$ результатов:

$$s^p : y \in \Gamma \setminus s^p, p\left(\left|\{x \in s^p : r(x) > r(y)\}\right| > q\right) > 1 - \alpha$$

- Провести точное ранжирование в рамках s^p

Первая фаза называется фильтрацией. Фильтраций может быть несколько.

Обновление формул

Обновлять формулы прям надо. Отдельная проблема ранжирования — обновление формул. Для этого нужно обновить:

- все мета-факторы;
- фильтрацию;
- формулу.

Или пойти другим путем, и ввести контракты на значения мета-факторов и универсальную фильтрацию.

План

1. Задача MLR
2. Метрики
   - Кренфилдский подход
   - Попарный подход
   - Настройка по пользовательским данным
3. Обучение
   - Факторы ранжирования
   - Особенности оптимизации
4. Эксплуатация
5. MLR в поиске

Релевантность

Релевантность — соответствие документа запросу. Есть много видов (MDRF):

- Пертинентность
- Экспертная релевантность
- Техническая релевантность
- etc.

Разделение аттрактивности и релевантности

На поведение пользователя влияет не только релевантность, но и аттрактивность результата. Можно попробовать разделить эти 2 понятия:

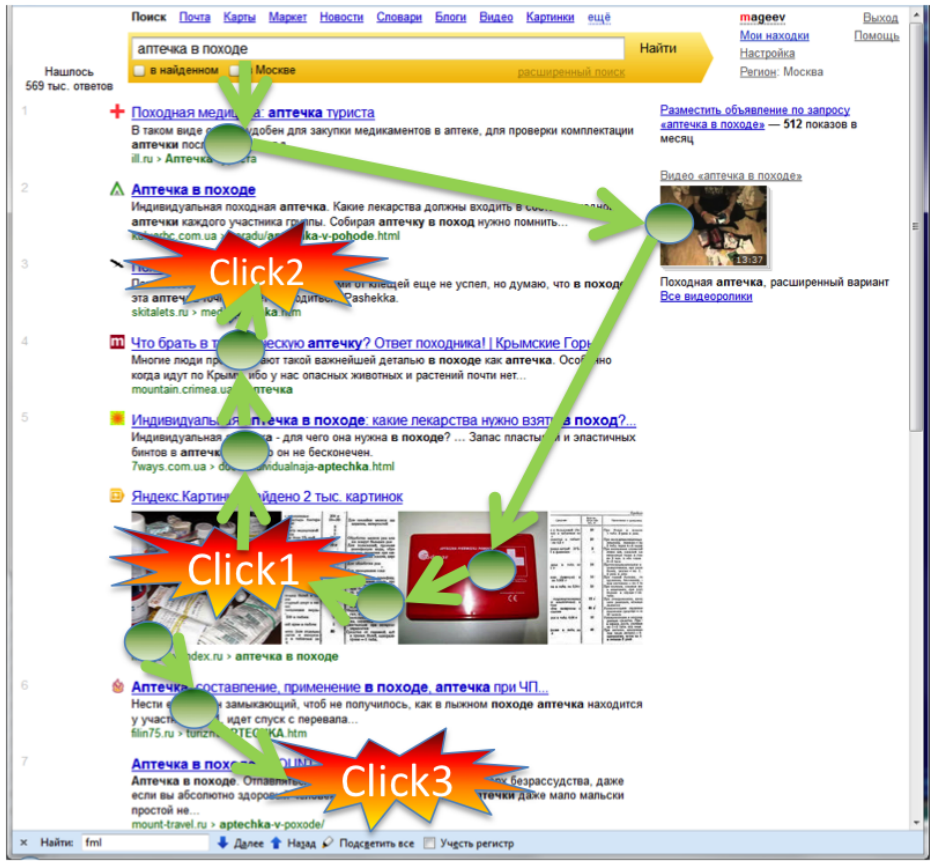
$$p(I, q, u, d) = p(I|d)p(d|q, u)$$

Первая часть зависит от свойств информации в документе, вторая от контекста в котором он представлен.

Разделение аттрактивности и релевантности

Свойства

- Произвольный порядок просмотра (уход от каскадной модели)
- Вероятность перехода зависит от аттрактивности и взаиморасположения блоков



Сколько задач ранжирования в поиске?

Сколько задач ранжирования в поиске?

Если не считать фильтраций, то:

1. Ранжирование (по разным роботам)
2. Смешивание результатов вертикалей
3. Персонализация ранжирования
4. Выбор оптимального представления выдачи

А нужно ли это улучшать?

Качество ранжирования определяет долю на рынке. Но борьба идет за 3-й 4-й знак, который пользователям не виден. Однако:

- Если модель не обновлять, то она протухнет:
  - Интернет меняется
  - Пользователи меняются еще быстрее
  - Сезонность
  - etc.
- Сейчас идет борьба за UI, который можно тоже ранжировать.

Что сейчас актуально? (IMHO)

- Категориальные факторы в ранжировании
- Учет аттрактивности
- Смешивание разных оценок для получения как оптимизации, так и более эффективных метрик
- Online метрики, способные принимать UI
- Уход от обратной связи и “отсосов”
- Метафакторы
- etc.