ОБУЧЕНИЕ РАНЖИРОВАНИЮ II

Сергей Николенко

НИУ ВШЭ — Санкт-Петербург
12 мая 2017 г.

Random facts:

- 12 мая 2004 г. в англиканской церкви введена должность веб-священника; веб-священники опекают приходы, существующие исключительно онлайн
- а 12 мая 2010 г. запущен национальный домен верхнего уровня .рф
- 12 мая 1968 г. родился знаменитый скейтбордист Тони Хоук

MART

ДЕРЕВЬЯ РЕГРЕССИИ

- Теперь давайте градиентный бустинг применять к задаче ранжирования более напрямую.
- Начнём с чуть изменённых обозначений для деревьев принятия решений в случае регрессии.
- Рассмотрим датасет $\mathbf{X} = \{\mathbf{x}_i, y_i\}$.
- Можно определить *regression stump* (регрессионный пень) так: выбираем одну координату (атрибут) j (т.е. берём x_{ij}) и ищем там оптимальное разбиение – такое значение порога t, что

$$S_j = \sum_{i \in \text{Left}} (y_i - \mu_{\text{Left}})^2 + \sum_{i \in \text{Right}} (y_i - \mu_{\text{Right}})^2$$

минимизируется, где Left и Right – множества точек слева и справа от t по j-й координате.

Деревья регрессии

- Если повторить эту процедуру L раз (как-то выбирая для расщепления листья – например, по максимальной дисперсии), получится регрессионное дерево с L листьями.
- В каждом листе определим γ_l – среднее по y_i из этого листа.
- Тогда, чтобы применить регрессионное дерево, надо по нему спуститься до листа и взять γ_l из этого листа.

MART

- MART – это бустинг, сделанный на регрессионных деревьях.
- Иначе говоря, окончательная модель будет, опять же, по $\mathbf{x} \in \mathbb{R}^d$ искать $y \in \mathbb{R}$, и искать в виде

$$F_M(\mathbf{x}) = \sum_{m=1}^{M} \alpha_m f_m(\mathbf{x}),$$

где $f_m(\mathbf{x})$ задаётся регрессионным деревом, а $\alpha_m \in \mathbb{R}$ – веса бустинга, и в процессе обучения обучаются одновременно f_m и α_m.

MART

- Нам нужно понять, как обучать новое дерево F_{m+1}, если мы уже обучили m деревьев.
- Зафиксируем функцию ошибки C (она дана свыше).
- Идея: следующее дерево моделирует производные ошибки по текущей модели в точках из датасета:

$$\Delta C \approx \frac{\partial C(F_m)}{\partial F_m} \Delta F,$$

и ΔC будет отрицательным, если взять для $\eta > 0$

$$\Delta F = -\eta \frac{\partial C(F_m)}{\partial F_m}.$$

- Непараметрический метод: мы рассматриваем F_m *в каждой точке из датасета* как «параметр» и по ним оптимизируем.

MART

- Пример: бинарная классификация, $y_i \in \{\pm 1\}$, $\mathbf{x} \in \mathbb{R}^n$, $F(\mathbf{x})$.
- Обозначим $p_+ = p(y = 1 \mid \mathbf{x})$, $p_- = p(y = -1 \mid \mathbf{x})$, $I_+(\mathbf{x}_i) = [y_i = 1]$, $I_-(\mathbf{x}_i) = [y_i = -1]$.
- Будем использовать (как и в RankNet, кстати) перекрёстную энтропию

$$L(y, F) = -I_+ \log p_+ - I_i \log p_i.$$

- Логистическая регрессия моделирует log odds:

$$F_N(\mathbf{x}) = \frac{1}{2}\log\left(\frac{p_+}{p_-}\right),$$

$$p_+ = \frac{1}{1+e^{-2\alpha F_m(\mathbf{x})}}, \quad p_- = \frac{1}{1+e^{2\alpha F_m(\mathbf{x})}},$$

и мы получаем

$$L(y,F) = \log\left(1+e^{-2y\alpha F_m(\mathbf{x})}\right).$$

MART

- $L(y, F) = \log\left(1 + e^{-2y\alpha F_m(\mathbf{x})}\right).$
- От этой функции легко взять производную по $F(\mathbf{x})$:

$$\bar{y}_i = \left[\frac{\partial L(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}\right]_{F(\mathbf{x})=F_m(\mathbf{x})} = \frac{2y_i\alpha}{1 + e^{2y_i\alpha F_m(\mathbf{x})}}.$$

- И мы строим новое регрессионное дерево, которое пытается смоделировать $\bar{y}_i$.

MART

- Осталось только выбрать вес, с которым это новое дерево войдёт в сумму.
- Мы хотим выбрать (примерно) оптимальный шаг для каждого листа, т.е. минимизировать потери:

$$\gamma_{lm} = \arg\min_{\gamma} \log\left(1 + e^{2y_i \alpha(F_m(\mathbf{x})+\gamma)}\right) = \arg\min_{\gamma} g(\gamma).$$

- Минимизировать можно итеративно по методу Ньютона-Рапсона: $\gamma_{n+1} = \gamma_n - \frac{g'(\gamma_n)}{g''(\gamma_n)}$.

MART

- И мы аппроксимируем одним шагом этого метода, начиная с нуля:

$$\gamma_{lm} = -\frac{g'}{g''} = \frac{\sum_{x_i \in R_{lm}} \bar{y}_i}{\sum_{x_i \in R_{lm}} |\bar{y}_i| \left(2\alpha - |\bar{y}_i|\right)}.$$

Упражнение. Проверьте эту формулу.

LAMBDAMART

- Теперь уже понятно, как из MART сделать LambdaMART.
- Мы просто добавим в градиенты целевую метрику, например

$$\lambda_{ij} = S_{ij} \left| \Delta \mathrm{NDCG} \frac{\partial C_{ij}}{\partial o_{ij}} \right|, \quad o_{ij} = F(x_i) - F(x_j).$$

- Функция ошибки нам тоже уже известна:

$$\begin{aligned} C_{ij} =& C(o_{ij}) = s_j - s_i + \log\left(1 + e^{s_i - s_j}\right), \\ \frac{\partial C_{ij}}{\partial o_{ij}} =& \frac{\partial C_{ij}}{\partial s_i} = -\frac{1}{1 + e^{o_{ij}}}. \end{aligned}$$

LAMBDAMART

- Получается, что знак λ_{ij} зависит только от меток i и j, и в каждой точке мы можем собрать все «действующие силы» как

$$\lambda_i = \sum_j \lambda_{ij}.$$

- Это и называется LambdaMART.
- Вариант: LambdaSMART (submodel): мы инициализируем первое дерево какой-нибудь обученной хорошей базовой моделью, а всё дальнейшее – это её уточнение.

LAMBDAMART

- Остался только один вопрос: откуда взять веса α_i (при $f_i(\mathbf{x})$)?
- Базовый LambdaMART выбирает эти веса индивидуально для каждого листа дерева.
- Мы хотим сдвинуться в сторону минимума ошибки; значит, надо идти в сторону нуля производной. Один шаг метода Ньютона-Рапсона:

$$F_m(x_i) = F_{m-1}(x_i) + v \sum_l \gamma_{lm} \left[x_i \in R_{lm}\right],$$

где $\gamma_{lm} = \dfrac{F'_m(x_i)}{F''_m(x_i)}$, а v – регуляризатор.

- Первая производная – это просто λ_i; вторая производная – λ_i':

$$\gamma_{lm} = \frac{\sum_{x_i \in R_{lm}} \lambda_i}{\sum_{x_i \in R_{lm}} \frac{\partial \lambda_i}{\partial F_m(x_i)}}$$

- И теперь можно собрать весь алгоритм целиком.

LAMBDASMART: АЛГОРИТМ

1. $F_0(x_i) = \mathrm{BaseModel}(x_i)$, $i = 0..N$.
2. Для m от 1 до M (числа деревьев в сумме):
   - 2.1 $y_i = \lambda_i = \sum_j \lambda_{ij}$, $i = 0..N$ (считаем градиенты);
   - 2.2 $w_i = \frac{\partial y_i}{\partial F(x_i)}$, $i = 0..N$ (вторые производные);
   - 2.3 строим новое дерево $\{R_{lm}\}_{l=1}^{L}$ на вершинах $\{y_i, x_i\}_{n=1}^{N}$;
   - 2.4 $\gamma_{lm} = \frac{\sum_{x_i \in R_{lm}} y_i}{\sum_{x_i \in R_{lm}} w_i}$, $l = 1..L$ (веса узлов);
   - 2.5 $F_m(x_i) = F_{m-1}(x_i) + v \sum_l \gamma_{lm} \left[x_i \in R_{lm}\right]$, $i = 0..N$.

КАК ОПТИМИЗИРОВАТЬ α_m

- В Lambda[S]MART веса бустинга подбираются как шаг метода Ньютона для каждого листа дерева.
- Но благодаря тому, что наши IR-метрики дискретные, можно просто подобрать оптимальный α_m.
- Давайте рассмотрим общую задачу: предположим, что у нас есть две ранжирующих функции, R и R', и мы хотим их оптимально линейно скомбинировать, т.е. подобрать оптимальный коэффициент α в

$$s_i = (1 - \alpha)s_i^R + \alpha s_i^{R'}.$$

КАК ОПТИМИЗИРОВАТЬ α_m

- Идея простая: представьте себе, как α меняется от 0 (в чистом виде R) до 1 (в чистом виде R').
- Тогда метрики вроде NDCG реально меняются только в точках пересечения (да и то не всегда, а только если пересеклись документы с разными метками).
- Ну вот и давайте просто переберём все такие пары, найдём их точки пересечения и составим список интересных значений α.
- А затем отсортируем список, подсчитаем метрику в каждом интервале и найдём оптимальный α; для ситуации бустинга просто надо это делать на каждом шаге.
- Кажется, что это очень тупо, но на самом деле это квадратичный алгоритм, что не так уж страшно.

КАК ОПТИМИЗИРОВАТЬ α_m

- LambdaMART – победитель в Yahoo! Learning to Rank Challenge (2011).
- Что было нового с тех пор?
- Plackett-Luce model выражает ранжирование в виде вероятностей:
  - выбираем первый документ с вероятностью, пропорциональной релевантности;
  - выбираем второй документ из остальных тоже пропорционально релевантности...
- Это даёт непрерывную listwise-ошибку, которую можно оптимизировать.
- BayesRank оптимизирует её, MatrixNet, видимо, тоже.

СПАСИБО!

Спасибо за внимание!