EleWise проекты

Лекция 6

Градиентный бустинг

6.1. Градиентный бустинг

6.1. Градиентный бустинг

Задача восстановления зависимости *y: X→Y* по точкам обучающей выборки (x_i, y_i), *yi = y(xi), i = 1, …., ℓ*

Определение: *Линейной композицией базовых алгоритмов at(x) = C(bt(x)), t=1,...., T, называется суперпозиция функций*

$$a(x) = C(\sum_{t=1}^{T} \alpha_t b_t(x))$$

где C: R → Y - решающее правило $\alpha_t \geq 0$.

- Пример 1: классификация на 2 класса, Y={-1, +1};

  $C(b) = \text{sign}(b), \quad a(x) = \text{sign}(b(x)),$
  $b: X \to \mathbb{R}$ — дискриминантная функция.

- Пример 2: регрессия, Y=R,
  $C(b) = b, \quad a(x) = b(x)$, решающее правило не используется

6.1. Градиентный бустинг

Линейная композиция базовых алгоритмов:

$$b(x) = \sum_{t=1}^{T} \alpha_t b_t(x), \quad x \in X, \quad \alpha_t \in \mathbb{R}_+.$$

Функционал качества с произвольной ф-ией потерь $\mathscr{L}(b, y)$:

$$Q(\alpha, b) = \sum_{i=1}^{\ell} \mathscr{L}\Big(\underbrace{\underbrace{\sum_{t=1}^{T-1} \alpha_t b_t(x_i)}_{u_{T-1,i}} + \alpha b(x_i)}_{u_{T,i}}, y_i\Big) \to \min_{\alpha, b}.$$

Ищем вектор $u = (b(x_i))_{i=1}^{\ell}$ из R^{ℓ} минимизирующий $Q(\alpha, b)$

$u_{T-1} = (u_{T-1,i})_{i=1}^{\ell}$ - текущее приближение вектора u

$u_T = (u_{T,i})_{i=1}^{\ell}$ - следующее приближение вектора u

6.1. Градиентный бустинг

Градиентный метод минимизации $Q(u) \to \min, \quad u \in \mathbb{R}^\ell$:

$$u_0 := \text{начальное приближение};$$

$$u_{T,i} := u_{T-1,i} - \alpha g_i, \quad i = 1, \ldots, \ell;$$

$g_i = \mathscr{L}'(u_{T-1,i}, y_i)$ компоненты вектора градиента,

α - градиентный шаг

Добавление базового алгоритма b_T:

$$u_{T,i} := u_{T-1,i} + \alpha b_T(x_i), \quad i = 1, \ldots, \ell$$

Будем искать такой базовый алгоритм b_T, чтобы вектор $(b_T(x_i))_{i=1}^\ell$ приближал вектор антиградиента $(-g_i)_{i=1}^\ell$:

$$b_T := \arg\max_b \sum_{i=1}^{\ell} \big(b(x_i) + g_i\big)^2$$

6.1. Градиентный бустинг

Вход: обучающая выборка X^ℓ; параметр T

Выход: базовые алгоритмы и их веса $\alpha_t b_t$, $t = 1, \ldots, T$;

1: инициализация: $u_i := 0$, $i = 1, \ldots, \ell$;

2: **для всех** $t = 1, \ldots, T$

3: найти базовый алгоритм, приближающий градиент:

$$b_t := \arg\min_b \sum_{i=1}^{\ell} \big(b(x_i) + \mathscr{L}'(u_i, y_i)\big)^2;$$

4: решить задачу одномерной минимизации:

$$\alpha_t := \arg\min_{\alpha>0} \sum_{i=1}^{\ell} \mathscr{L}\big(u_i + \alpha b_t(x_i), y_i\big);$$

5: обновить значения композиции на объектах выборки:

$$u_i := u_i + \alpha_t b_t(x_i); \quad i = 1, \ldots, \ell;$$

6.1. Градиентный бустинг

Алгоритм:

- Построение начальной модели (дерево)
- Расчет остатков (градиентов функции потерь) модели
- Изменение модели, где **остатки** используются **в качестве целевой переменной**.

- Последовательность деревьев регрессии

6.1. Градиентный бустинг

- Набор данных, используемый для создания дерева в последовательности деревьев, является случайной выборкой из обучающего набора данных.

- На каждом шаге генерируется случайная выборка без возвращения.

- В сформированной выборке наблюдения взвешиваются в соответствии с ошибками модели, полученной на предыдущем шаге.

6.2. Градиентный бустинг. Модификации и эвристика

6.2. Градиентный бустинг. Модификации и эвристика

Известно, что рандомизации могут повышать качество за счет повышения различности базовых алгоритмов (на этом основаны bagging, RF, RSM)

Идея:

на шагах 3-5 использовать не всю выборку X^ℓ, в случайную подвыборку с повторениями, как в бэггинге.

Преимущества:

- улучшается качество
- улучшается сходимость
- уменьшается время обучения

6.2. Градиентный бустинг. Модификации и эвристика

Исторически первый вариант бустинга (1995)

Задача классификации на два класса $Y=\{-1, +1\}$,

$\mathscr{L}(b(x_i), y_i) = e^{-b(x_i)y_i}$ - экспоненциальная ф-ия потерь, убывающая ф-ия отступа $M=bi(x_i)y_i$

Преимущества:

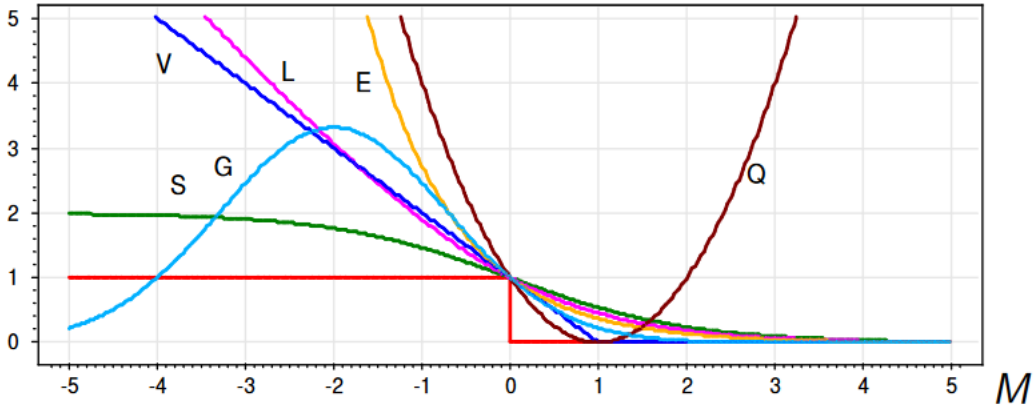
- для обучения b_t на каждом шаге t решается стандартная задача минимизации взвешенного эмпирического риска
- задача оптимизации α_t решается аналитически

Недостатки:

- AdaBoost слишком чувствителен к выбросам из-за экспоненциального роста потерь при $M_i < 0$

6.2. Градиентный бустинг. Модификации и эвристика

Функции потерь $\mathscr{L}(M)$ в задачах классификации на два класса



$E(M) = e^{-M}$ — экспоненциальная (AdaBoost);
$L(M) = \log_2(1 + e^{-M})$ — логарифмическая (LogitBoost);
$G(M) = \exp\big(-cM(M + s)\big)$ — гауссовская (BrownBoost);
$Q(M) = (1 - M)^2$ — квадратичная;
$S(M) = 2(1 + e^{M})^{-1}$ — сигмоидная;
$V(M) = (1 - M)_+$ — кусочно-линейная (SVM);

6.3. Градиентный бустинг. Настройка

6.3. Градиентный бустинг. Настройка

Параметры, определяющие тип бустинга и способы построения деревьев:

- booster – какой бустинг проводить: над решающими деревьями или линейный,
- grow_policy – порядок построения дерева: на следующем шаге расщеплять вершину, ближайшую к корню, или на которой ошибка максимальна,
- criterion – критерий выбора расщепления (при построении деревьев),
- init – какой алгоритм использовать в качестве первого базисного (именно его ответы будет улучшать бустинг).

6.3. Градиентный бустинг. Настройка

Основные параметры:

- eta / learning_rate – темп (скорость) обучения,
- num_iterations / n_estimators – число итераций бустинга,
- early_stopping_round – если на отложенном контроле заданная функция ошибки не уменьшается такое число итераций, обучение останавливается.

6.3. Градиентный бустинг. Настройка

Параметры, ограничивающие сложность дерева:

- max_depth – максимальная глубина,
- max_leaves / num_leaves / max_leaf_nodes – максимальное число вершин в дереве (иногда, если значение меньше нуля, то игнорируется ограничение по максимальной глубине),
- gamma / min_gain_to_split – порог на уменьшение функции ошибки при расщеплении в дереве,
- min_data_in_leaf / min_samples_leaf – минимальное число объектов в листе,
- min_sum_hessian_in_leaf – минимальная сумма весов объектов в листе,
- min_samples_split – минимальное число объектов, при котором делается расщепление,
- min_impurity_split – расщепление при построении дерева не будет проведено, если impurity изменяется при расщеплении на величину меньшую этого порога.

6.3. Градиентный бустинг. Настройка

Параметры формирования подвыборок:

- subsample / bagging_fraction – какую часть объектов обучения использовать для построения одного дерева,
- colsample_bytree / feature_fraction – какую часть признаков использовать для построения одного дерева,
- colsample_bylevel / max_features – какую часть признаков использовать для построения расщепления в дереве.

6.3. Градиентный бустинг. Настройка

Параметры регуляризации

- lambda / lambda_l2 (L2),
- alpha / lambda_l1 (L1).

6.3. Градиентный бустинг. Настройка

Что еще?

- число используемых потоков,
- на CPU или GPU,
- хранить модель в ОЗУ при обучении или нет,
- метод поиска расщепления.

6.3. Градиентный бустинг

- В отличие от случайных деревьев, в бустинге увеличение числа деревьев не всегда приводит к улучшению качества решения на тесте.
- Чем меньше темп, тем больше деревьев нужно (зависимость нелинейная)
- При оптимизации параметров обычно фиксируют число деревьев (оно должно быть не очень большим, чтобы алгоритм быстро обучался), подбирая под него темп и значения остальных параметров. При построении итогового алгоритма увеличивают число деревьев и находят соответствующее значение темпа (оставляя остальные параметры неизменными).

6.3. Градиентный бустинг

- Градиентный бустинг - наиболее общий из всех бустингов:
  - произвольная ф-ия потерь
  - произвольное пространство оценок R
  - подходит для регрессии, классификации, ранжирования
- Важное открытие середины 90х: обобщающая способность бустинга не ухудшается с ростом сложности *T*
- Стохастический вариант SGB - лучше и бустрее
- Градиентный бустинг над решающими деревьями часто работает лучше, чем RF
- Технология Yandex.Matrixnet - это градиентный бустинг над “небрежными” решающими деревьями ODT oblivious decision tree)