

ДЕРЕВЬЯ ПРИНЯТИЯ РЕШЕНИЙ И БУСТИНГ

Сергей Николенко

СПбГУ — Санкт-Петербург

15 февраля 2024 г.

Random facts:

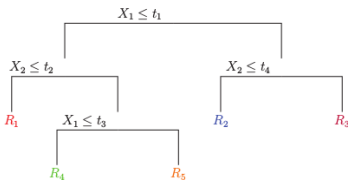


- 15 февраля на Вануату — День Джона Фрума, центрального персонажа в культе карго, изображаемого в виде американского военнослужащего Второй мировой войны; возможно, «Джон Фрум» — это искажение «John from (America)»; последователи Джона Фрума соорудили на острове Танна символическую взлётно-посадочную полосу с макетами самолётов и символическими вышками, на которых дежурили «диспетчеры»
- 15 февраля в США — День Сьюзен Б. Энтони (Susan B. Anthony), одной из главных суфражисток XIX века; в 1863 году они с Элизабет Стэнтон организовали Women's Loyal National League, которая собрала 400 тысяч подписей в поддержку отмены рабства, а затем лоббировала равноправие всех граждан США — и женщин, и афроамериканцев (но не вылоббировала, в 14-й и 15-й поправках про женщин ничего нет)
- 15 февраля 1717 г. в Петербурге было впервые напечатано «Юности честное зерцало»
- 15 февраля 2013 г. астероид диаметром ≈ 17 метров и массой ≈ 10000 тонн вошёл в атмосферу Земли на скорости ≈ 18 км/с и разрушился на высоте 15–25 км в окрестностях Челябинска; снятое на видеорегистратор падение было показано в начальных титрах фильмов World War Z и Edge of Tomorrow

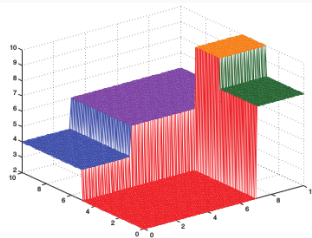
ДЕРЕВЬЯ ПРИНЯТИЯ РЕШЕНИЙ

ДЕРЕВЬЯ ПРИНЯТИЯ РЕШЕНИЙ

- *Дерево принятия решений* — это дерево (surprise!). На нём есть метки:
 - в узлах, не являющиеся листьями: атрибуты (фичи), по которым различаются случаи;
 - в листьях: значения целевой функции;
 - на рёбрах: значения атрибута, из которого исходит ребро.
- Чтобы классифицировать новый случай, нужно спуститься по дереву до листа и выдать соответствующее значение.



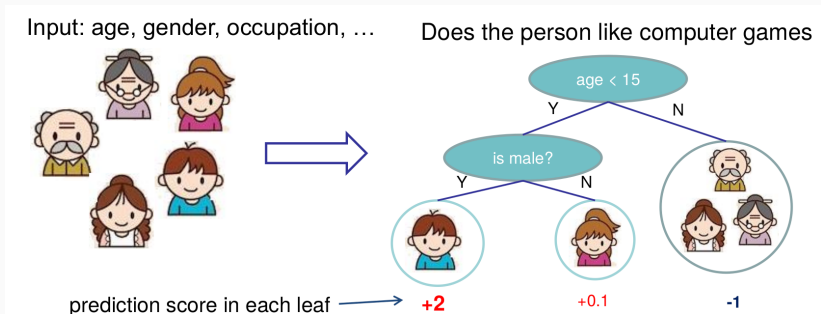
(a)



(b)

ДЕРЕВЬЯ ПРИНЯТИЯ РЕШЕНИЙ

- Картинки от Tianqi Chen и Carlos Guestrin, авторов XGBoost:



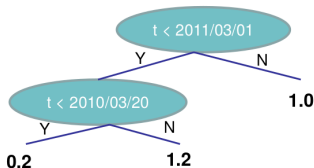
- Конечно, перебрать все деревья нельзя, их строят жадно.
 1. Выбираем очередной атрибут Q , помещаем его в корень.
 2. Выбираем оптимальное разбиение по атрибуту. Для всех интервалов разбиения:
 - оставляем из тестовых примеров только те, у которых значение атрибута Q попало в этот интервал;
 - рекурсивно строим дерево в этом потомке.
- Остались три вопроса:
 1. как проводить разбиение?
 2. как выбирать новый атрибут?
 3. когда останавливаться?

ДЕРЕВЬЯ ПРИНЯТИЯ РЕШЕНИЙ

- Если атрибут бинарный или дискретный с небольшим числом значений, то просто по значениям.
- Если непрерывный – можно брать среднее арифметическое (тем самым минимизируя сумму квадратов).
- Выбирают атрибут, оптимизируя целевую функцию. Для задачи регрессии просто минимизируем среднеквадратическую ошибку по отношению к текущему предсказателю

$$y_{\tau} = \frac{1}{|\mathbf{X}_{\tau}|} \sum_{\mathbf{x}_n \in \mathbf{X}_{\tau}} t_n.$$

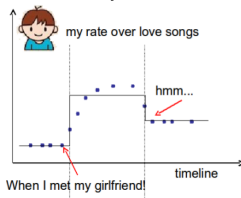
The model is regression tree that splits on time



Equivalently

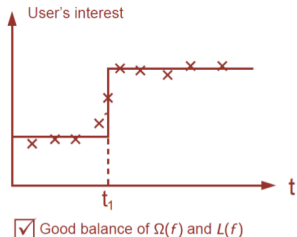
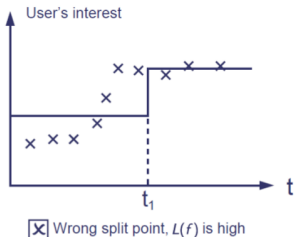
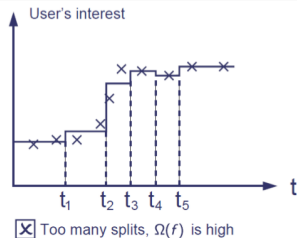
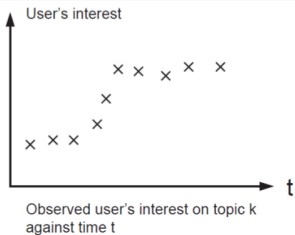


Piecewise step function over time



ДЕРЕВЬЯ ПРИНЯТИЯ РЕШЕНИЙ

- Как обучить дерево:



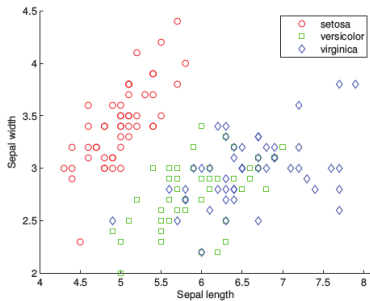
- Предположим, что мы решаем задачу классификации на K классов.
- Тогда «сложность» подмножества данных $\mathbf{X}_\tau$ относительно целевой функции $f(\mathbf{x}) : \mathbf{X} \rightarrow \{1, \dots, K\}$ характеризуется *перекрёстной энтропией*:

$$Q(\mathbf{X}_\tau) = \sum_{k=1}^K p_{\tau,k} \ln p_{\tau,k}.$$

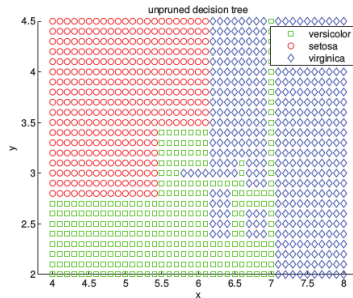
- Иногда ещё используют *индекс Джини* (Gini index):
$$G(\mathbf{X}_\tau) = \sum_{k=1}^K p_{\tau,k} (1 - p_{\tau,k}).$$

- Когда останавливаться? Недоучиться плохо и переучиться плохо.
- Останавливаться, когда ошибка перестанет меняться, тоже плохо (она может опять начать меняться ниже).
- Поэтому делают так: выращивают большое дерево, чтобы наверняка, а потом *обрезают* его (pruning): поддереву τ схлопывают в корень, а правило предсказания в корне считают как $y_\tau = \frac{1}{|\mathbf{X}_\tau|} \sum_{\mathbf{x}_n \in \mathbf{X}_\tau} t_n$.
- Обрезают, оптимизируя функцию ошибки с регуляризатором: $\sum_{\tau=1}^{|T|} Q(\mathbf{X}_\tau) + \lambda|T|$ (для классификаторов здесь можно использовать долю ошибок классификации).

- Пример на датасете iris.
- Вход:

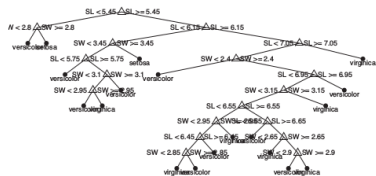


(a)

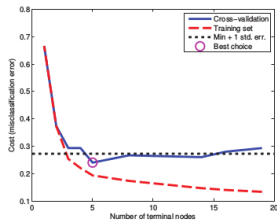


(b)

- Слишком глубокое дерево и его ошибки:

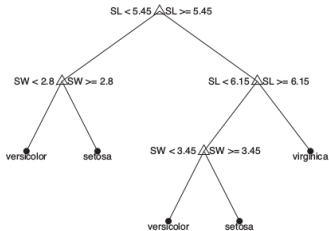


(a)

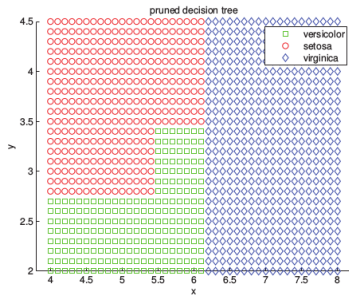


(b)

- Обрезанное дерево:

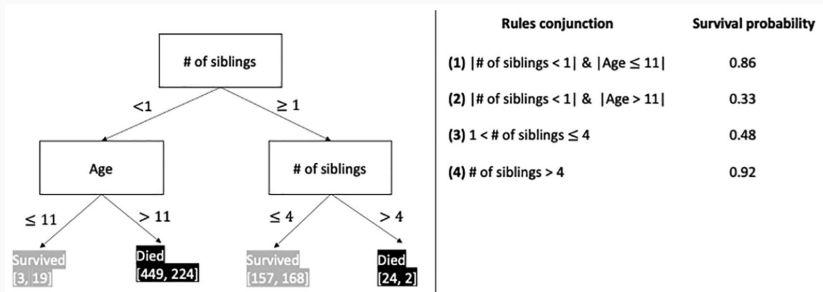


(a)



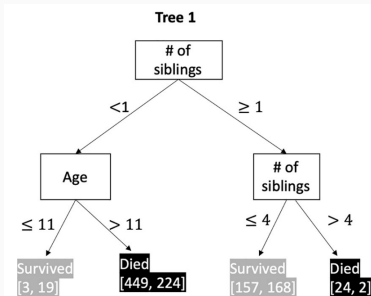
(b)

- Пример на датасете о пассажирах «Титаника»:



- Обратите внимание, насколько хорошо интерпретируется дерево.

- Но это дерево скорее одна эвристика, чем финальный ответ. Вот другая:

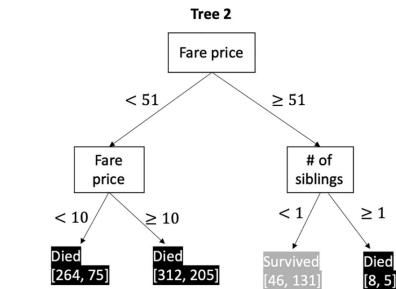


(A) |# of siblings < 1| & |Age ≤ 11| -> 0.86

(B) |# of siblings < 1| & |Age > 11| -> 0.33

(C) 1 ≤ # of siblings ≤ 4 -> 0.48

(D) # of siblings > 4 -> 0.92



(A) Fare price < 10 -> 0.22

(B) 10 ≤ Fare price < 51 -> 0.39

(C) |Fare price ≥ 51| & |# of siblings < 1| -> 0.74

(D) |Fare price ≥ 51| & |# of siblings ≥ 1| -> 0.38

- К этой мысли мы ещё вернёмся...

ОБЪЕДИНЕНИЕ МОДЕЛЕЙ

- До сих пор мы разрабатывали (и потом ещё будем разрабатывать) модели, которые делают предсказания (в основном для задач регрессии и классификации).
- Таким образом, мы можем попробовать обучить сразу много разных моделей!
- Model selection – это о том, как выбрать из них лучшую.
- Но, может быть, можно не выбирать, а использовать все сразу?

- Комитет: обучаем L разных моделей, а потом так или иначе усредняем-комбинируем их результаты.
- Альтернатива: обучаем L разных моделей, а потом обучаем отдельную модель о том, какую из них использовать для предсказания (например, дерево принятия решений).

- Начнём с самого простого – байесовского усреднения.
- Мы уже знаем, что такое комбинация моделей – например, линейная смесь гауссианов:

$$p(\mathbf{x}) = \sum_k \pi_k N(\mathbf{x} \mid \mu_k, \Sigma_k).$$

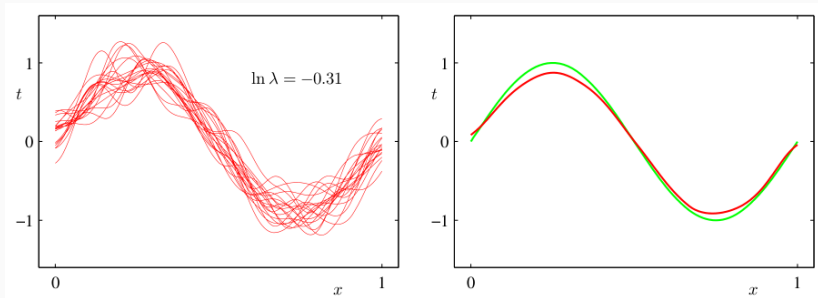
- Если её обучать, мы обучим коэффициенты смеси, и результат будет порождён как бы двухуровневым процессом.

- А байесовское усреднение будет выглядеть как

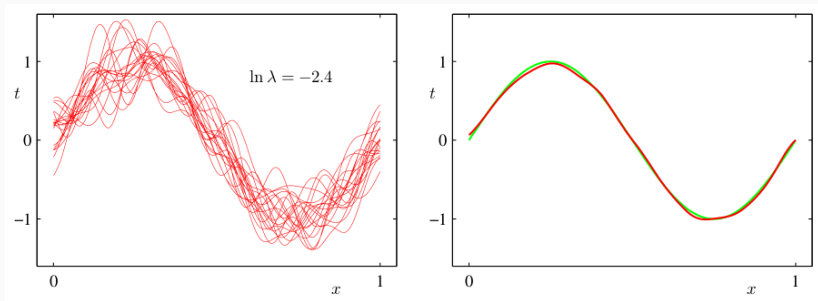
$$p(\mathbf{X}) = \sum_{h=1}^H p(\mathbf{X} | h)p(h).$$

- Смысл теперь в том, что генерирует $\mathbf{X}$ только одна модель, но мы просто не знаем какая именно; когда $\mathbf{X}$, апостериорные распределения $p(h | \mathbf{X})$ сужаются, и мы выбираем то, что надо.
- Но метод очень простой: взять много моделей и усреднить.
- Где-то мы это уже видели...

ГДЕ-ТО МЫ ЭТО УЖЕ ВИДЕЛИ



ГДЕ-ТО МЫ ЭТО УЖЕ ВИДЕЛИ



- На этих картинках – модели с высоким bias, которые обучены по разным датасетам, сгенерированным одним и тем же распределением.
- И если их усреднить, получится как раз то, что надо.
- Но в жизни у нас нет возможности генерировать много датасетов: сколько данных есть, столько есть.
- Просто разбивать датасет на части – не поможет. Что делать?

- Пусть у нас есть датасет $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$.
- Давайте сгенерируем много датасетов так: будем выбирать из $\mathbf{X}$ N точек с замещением, т.е. в новом датасете некоторые точки будут повторяться.
- Этот метод называется *bootstrapping*.

- Мы сделаем так M датасетов размера N (с повторяющимися точками), потом обучим M моделей, а потом образуем из них комитет и будем предсказывать как

$$y(\mathbf{x}) = \frac{1}{M} \sum_{m=1}^M y_m(\mathbf{x}).$$

- Это называется *bagging* (bootstrap aggregation).
- На первый взгляд кажется, что это какая-то ерунда: мы пытаемся получить что-то из ничего...

- Пусть настоящая функция, которую мы пытаемся предсказать – $h(\mathbf{x})$, т.е. модели наши выглядят как

$$y_m(\mathbf{x}) = h(\mathbf{x}) + \epsilon_m(\mathbf{x}).$$

- Тогда средняя ошибка модели – это

$$E_{\mathbf{x}} \left[(y_m(\mathbf{x}) - h(\mathbf{x}))^2 \right] = E_{\mathbf{x}} [\epsilon_m(\mathbf{x})^2].$$

- И средняя ошибка тех моделей, которые мы обучаем, получается

$$E_{\text{avg}} = \frac{1}{M} \sum_{m=1}^M E_{\mathbf{x}} [\epsilon_m(\mathbf{x})^2].$$

- И средняя ошибка тех моделей, которые мы обучаем, получается

$$E_{\text{avg}} = \frac{1}{M} \sum_{m=1}^M E_{\mathbf{x}} [\epsilon_m(\mathbf{x})^2] .$$

- А ошибка комитета – это

$$\begin{aligned} E_{\text{com}} &= E_{\mathbf{x}} \left[\left(\frac{1}{M} \sum_{m=1}^M (y_m(\mathbf{x}) - h(\mathbf{x})) \right)^2 \right] = \\ &= E_{\mathbf{x}} \left[\left(\frac{1}{M} \sum_{m=1}^M \epsilon_m(\mathbf{x}) \right)^2 \right] . \end{aligned}$$

- $E_{\text{avg}} = \frac{1}{M} \sum_{m=1}^M E_{\mathbf{x}} [\epsilon_m(\mathbf{x})^2], E_{\text{com}} = E_{\mathbf{x}} \left[\left(\frac{1}{M} \sum_{m=1}^M \epsilon_m(\mathbf{x}) \right)^2 \right].$
- Если предположить, что $E_{\mathbf{x}} [\epsilon_m(\mathbf{x})] = 0$, и ошибки некоррелированы: $E_{\mathbf{x}} [\epsilon_m(\mathbf{x})\epsilon_l(\mathbf{x})] = 0$, мы получим

$$E_{\text{com}} = \frac{1}{M} E_{\text{avg}}.$$

- $E_{\text{com}} = \frac{1}{M} E_{\text{avg}}!$
- Это кажется совершенно невероятным. На самом деле всё не так хорошо – конечно, ошибки на самом деле сильно коррелированы.
- И, конечно, на самом деле обычно уменьшение ошибки не такое большое.
- Но можно показать, что в любом случае $E_{\text{com}} \leq E_{\text{avg}}$, так что хуже от этого не будет, а лучше стать может.

БУСТИНГ: ADA BOOST

- Следующая идея объединения моделей: предположим, что у нас есть возможность обучать какую-нибудь простую модель (weak learner) на подмножестве данных.
- Тогда можно делать так: обучили модель, посмотрели, где она хорошо работает, обучили следующую модель на том подмножестве, где она работает плохо, повторили.
- Этот метод называется *бустинг* (boosting).

- AdaBoost: самый простой вариант. Рассмотрим задачу бинарной классификации; данные – это $\mathbf{x}_1, \dots, \mathbf{x}_N$ с ответами $t_1, \dots, t_N, t_i \in \{-1, 1\}$.
- Снабдим каждый тестовый пример весом w_i ; изначально положим $w_i = \frac{1}{N}$.
- Предположим, что у нас есть процедура, которая обучает некоторый классификатор, выдающий $y(\mathbf{x}) \in \{-1, 1\}$, на взвешенных данных (минимизируя взвешенную ошибку).

- Тогда в алгоритме AdaBoost мы инициализируем $w_n^{(1)} := 1/N$, а потом для $m = 1..M$:

1. обучаем классификатор $y_m(\mathbf{x})$, который минимизирует функцию ошибки

$$J_m = \sum_{n=1}^N w_n^{(m)} [y_m(\mathbf{x}_n) \neq t_n];$$

2. вычисляем

$$\epsilon_m = \frac{\sum_{n=1}^N w_n^{(m)} [y_m(\mathbf{x}_n) \neq t_n]}{\sum_{n=1}^N w_n^{(m)}}, \quad \alpha_m = \ln \left(\frac{1 - \epsilon_m}{\epsilon_m} \right);$$

3. пересчитываем новые веса

$$w_n^{(m+1)} = w_n^{(m)} e^{\alpha_m [y_m(\mathbf{x}_n) \neq t_n]}.$$

- После обучения предсказываем как $Y_M(\mathbf{x}) = \text{sign} \left(\sum_{m=1}^M \alpha_m y_m(\mathbf{x}) \right).$

- Смысл именно такой, как мы говорили: сначала тренируем абстрактно лучший классификатор. Потом увеличиваем веса неправильно классифицированным примерам, обучаем новый классификатор, и т.д.

- Изначально, когда AdaBoost придумали [Freund, Shapire, 1997], мотивация была такая: предположим, что ошибка каждого слабого классификатора h_t не превышает $\epsilon_t = \frac{1}{2} - \gamma_t$.
- Тогда можно показать, что окончательная ошибка не превосходит

$$\prod_t (2\sqrt{\epsilon_t(1 - \epsilon_t)}) = \prod_t \sqrt{1 - 4\gamma_t^2} \leq e^{-2\sum_t \gamma_t^2}.$$

- Однако на самом деле гарантий на γ_t обычно нету, и практические результаты AdaBoost лучше, чем можно было бы ожидать из этой оценки.

- Основная идея [Friedman et al., 2000]: давайте определим экспоненциальную ошибку

$$E = \sum_{n=1}^N e^{-t_n f_m(\mathbf{x}_n)},$$

где f_m – линейная комбинация базовых классификаторов:

$$f_m(\mathbf{x}) = \frac{1}{2} \sum_{l=1}^m \alpha_l y_l(\mathbf{x}).$$

- Мы хотим минимизировать E по α_l и параметрам $y_l(\mathbf{x})$.

- Минимизируем $E = \sum_{n=1}^N e^{-t_n f_m(\mathbf{x}_n)}$.
- Вместо глобальной оптимизации будем действовать жадно: пусть $y_1(\mathbf{x}), \dots, y_{m-1}(\mathbf{x})$ и $\alpha_1, \dots, \alpha_{m-1}$ уже зафиксированы. Тогда ошибка получается

$$E = \sum_{n=1}^N e^{-t_n f_{m-1}(\mathbf{x}_n) - \frac{1}{2} t_n \alpha_m y_m(\mathbf{x})} = \sum_{n=1}^N w_n^{(m)} e^{-\frac{1}{2} t_n \alpha_m y_m(\mathbf{x})},$$

где $w_N^{(m)} = e^{-t_n f_{m-1}(\mathbf{x}_n)}$ – это как раз и есть наши веса, и их теперь можно считать константами.

- На правильных классификациях произведение -1 , на неправильных $+1$:

$$\begin{aligned} E &= e^{-\frac{\alpha_m}{2}} \sum_{\text{correct}} w_n^{(m)} + e^{\frac{\alpha_m}{2}} \sum_{\text{wrong}} w_n^{(m)} = \\ &= \left(e^{\frac{\alpha_m}{2}} - e^{-\frac{\alpha_m}{2}} \right) \sum_{n=1}^N w_n^{(m)} [y_m(\mathbf{x}_n) \neq t_n] + e^{-\frac{\alpha_m}{2}} \sum_{n=1}^N w_n^{(m)}, \end{aligned}$$

и достаточно минимизировать $J_m = \sum_{n=1}^N w_n^{(m)} [y_m(\mathbf{x}_n) \neq t_n]$.

- Ну а когда мы обучим $y_m(\mathbf{x})$, из $E = \sum_{n=1}^N w_n^{(m)} e^{-\frac{1}{2} t_n \alpha_m y_m(\mathbf{x})}$ получится

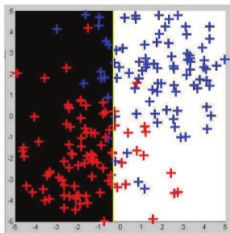
$$w_n^{(m+1)} = w_n^{(m)} e^{-\frac{1}{2} t_n \alpha_m y_m(\mathbf{x}_n)} = w_n^{(m)} e^{-\frac{1}{2} \alpha_m} e^{\alpha_m [y_m(\mathbf{x}_n) \neq t_n]},$$

и на $e^{-\frac{1}{2} \alpha_m}$ можно все веса сократить.

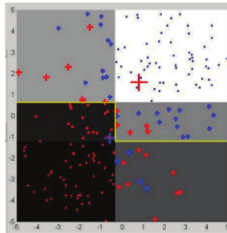
- Таким образом, бустинг можно рассматривать как оптимизацию экспоненциальной ошибки.

- А что за weak learners применяются в реальных приложениях?
- Обычно бустинг применяется, когда есть набор уже посчитанных фич (посчитанных из каких-то более сложных моделей), и нужно объединить их в единую модель.
- Часто слабые классификаторы очень, очень простые.
- *Пни принятия решений* (decision stumps): берём одну координату и ищем по ней оптимальное разбиение.

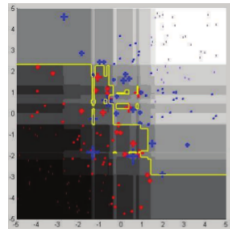
- Пример бустинга на пнях:



(a)



(b)



(c)

- Могут быть чуть посложнее: *деревья принятия решений* (decision trees).

ГРАДИЕНТНЫЙ БУСТИНГ

- Теперь – к градиентному бустингу (xgboost – это как раз градиентный бустинг).
- Предположим, что мы хотим обучить ансамбль из K деревьев:

$$\hat{y}_i = \sum_{k=1}^K f_k(x_i), \text{ где } f_k \in F.$$

- Целевая функция – это потери + регуляризаторы:

$$\text{Obj} = \sum_{i=1}^N l(y_i, \hat{y}_i) + \sum_{k=1}^K \Omega(f_k).$$

- Например, для регрессии $l(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2$.
- А для классификации в AdaBoost было

$$l(y_i, \hat{y}_i) = y_i \ln(1 + e^{-\hat{y}_i}) + (1 - y_i) \ln(1 + e^{\hat{y}_i}).$$

- Мы не можем просто взять и минимизировать общую ошибку – трудно минимизировать по всевозможным деревьям.
- Так что опять продолжаем жадным образом:

$$\begin{aligned}\hat{y}_i^{(0)} &= 0, \\ \hat{y}_i^{(1)} &= f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i), \\ \hat{y}_i^{(2)} &= f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i), \\ &\dots,\end{aligned}$$

а предыдущие деревья всегда остаются теми же самыми, они фиксированы.

- Чтобы добавить следующее дерево, нужно оптимизировать

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i), \text{ так что}$$

$$\text{Obj}^{(t)} = \sum_{i=1}^N l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) + \text{Const.}$$

- Например, для квадратов отклонений

$$\text{Obj}^{(t)} = \sum_{i=1}^N (2(\hat{y}_i^{(t-1)} - y_i)f_t(x_i) + f_t(x_i)^2) + \Omega(f_t) + \text{Const.}$$

- Чтобы оптимизировать

$$\text{Obj}^{(t)} = \sum_{i=1}^N l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) + \text{Const},$$

давайте заменим это на аппроксимацию второго порядка.

- Обозначим

$$g_i = \frac{\partial l(y_i, \hat{y}_i^{(t-1)})}{\partial \hat{y}^{(t-1)}}, \quad h_i = \frac{\partial^2 l(y_i, \hat{y}_i^{(t-1)})}{\partial (\hat{y}^{(t-1)})^2},$$

тогда

$$\text{Obj}^{(t)} \approx \sum_{i=1}^N \left(l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right) + \Omega(f_t) + \text{Const}.$$

- Например, для квадрата отклонения $g_i = 2(\hat{y}^{(t-1)} - y_i)$, $h_i = 2$.

- Итак,

$$\text{Obj}^{(t)} \approx \sum_{i=1}^N \left(l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right) + \Omega(f_t) + \text{Const.}$$

- Уберём константы:

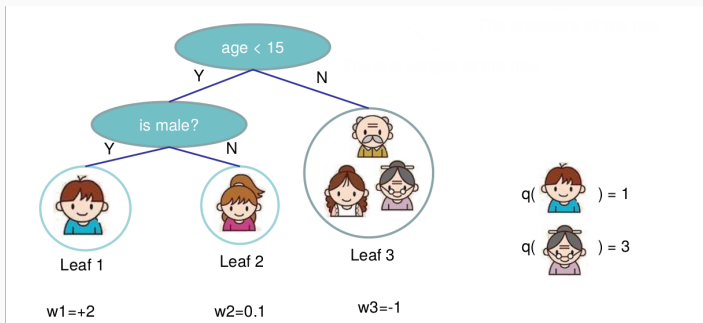
$$\text{Obj}^{(t)} \approx \sum_{i=1}^N \left(g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right) + \Omega(f_t).$$

- И это и есть основная идея *градиентного бустинга*.
- Теперь давайте вернёмся к обучению деревьев и сложим всё воедино.

ГРАДИЕНТНЫЙ БУСТИНГ

- Дерево – это вектор оценок в листьях и функция, которая вход отображает в лист:

$$f_t(x) = w_{q(x)}, \text{ где } w \in \mathbb{R}^T, q: \mathbb{R}^d \rightarrow \{1, \dots, T\}.$$



- Сложность дерева можно определить как $\Omega(f_t) = \gamma T + \frac{1}{2} \lambda \sum_{j=1}^T w_j^2$.

- Теперь перегруппируем слагаемые относительно листьев:

$$\begin{aligned}\text{Obj}^{(t)} &\approx \sum_{i=1}^N \left(g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right) + \Omega(f_t) \\&= \sum_{i=1}^N \left(g_i w_{q(x_i)} + \frac{1}{2} h_i w_{q(x_i)}^2 \right) + \Omega(f_t) \\&= \sum_{j=1}^T \left(w_j \sum_{i \in I_j} g_i + \frac{1}{2} w_j^2 \left(\sum_{i \in I_j} h_i + \lambda \right) \right) + \gamma T \\&= \sum_{j=1}^T \left(G_j w_j + \frac{1}{2} w_j^2 (H_j + \lambda) \right) + \gamma T,\end{aligned}$$

где $I_j = \{i \mid q(x_j) = i\}$, $G_j = \sum_{i \in I_j} g_i$, $H_j = \sum_{i \in I_j} h_i$.

- Это сумма T независимых квадратичных функций, так что

$$w_j^* = -\frac{G_j}{H_j + \lambda}, \quad \text{Obj}^{(t)} \approx -\frac{1}{2} \sum_{j=1}^T \frac{G_j^2}{H_j + \lambda} + \gamma T.$$

- Пример из (Chen, Guestrin):

Instance index gradient statistics

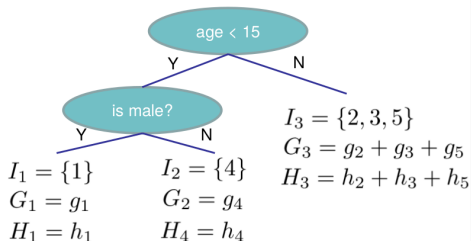
1  g_1, h_1

2  g_2, h_2

3  g_3, h_3

4  g_4, h_4

5  g_5, h_5



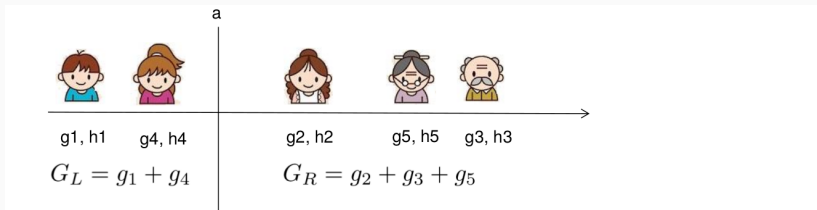
$$Obj = - \sum_j \frac{G_j^2}{H_j + \lambda} + 3\gamma$$

The smaller the score is, the better the structure is

- Так что мы находим наилучшую структуру дерева относительно $-\frac{1}{2} \sum_{j=1}^T \frac{G_j^2}{H_j + \lambda} + \gamma T$ и используем оптимальные веса листьев $w_j^* = -\frac{G_j}{H_j + \lambda}$.
- Как найти структуру? Жадно: для каждого листа попробуем добавить разбиение, и целевая функция меняется на

$$\text{Gain} = \frac{1}{2} \left(\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} \right) - \gamma.$$

- Самое лучшее разбиение – то, которое максимизирует gain:

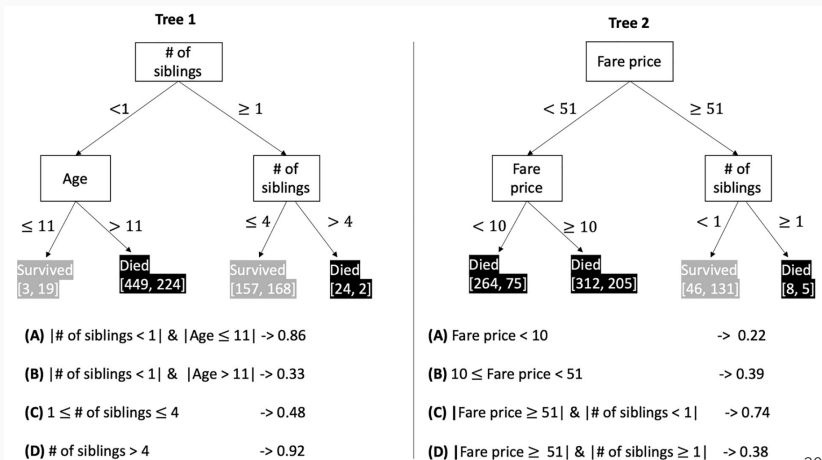


- Обрезание: сначала вырастим дерево до максимальной глубины, потом рекурсивно обрежем листья с отрицательным gain.

- Разработанный в «Яндекс» вариант градиентного бустинга для ранжирования – *MatrixNet*.
- Точные детали его не опубликованы, но основные особенности известны:
 - *oblivious decision trees* – все узлы одного уровня обязательно используют один и тот же атрибут; это дополнительная регуляризация, помогающая выделять меньше и более полезных признаков;
 - вместо ограничений на число сэмплов в листе – регуляризация самих значений в листьях;
 - сложность модели в бустинге зависит от итерации (сначала простые, потом более сложные).
- А основная идея та же самая.

ОБРАТНО К ДЕРЕВЬЯМ

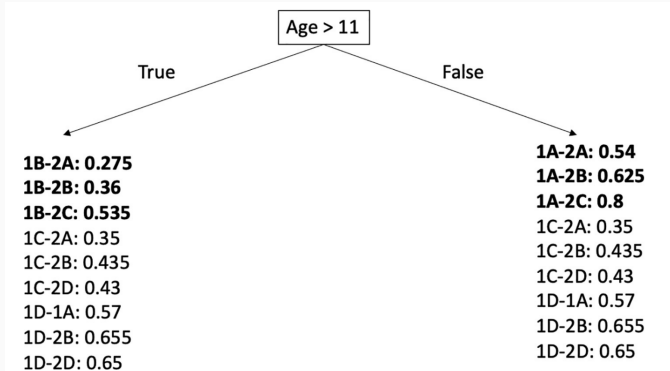
- Пример современной работы о бустинге (Sagi, Rokach, 2021)
- Бустинг улучшает результаты деревьев, но жертвует интерпретируемостью; сразу сотни примерно таких деревьев:



- То же самое происходит в случайных лесах.
- Давайте объединим всё то, что деревья нам говорят:

(1A-2A) # of siblings < 1 & Age ≤ 11 & Fare price < 10	-> 0.54
(1A-2B) # of siblings < 1 & Age ≤ 11 & 10 ≤ Fare price < 51	-> 0.625
(1A-2C) # of siblings < 1 & Age ≤ 11 & Fare price ≥ 51	-> 0.8
(1B-2A) # of siblings < 1 & Age > 11 & Fare price < 10	-> 0.275
(1B-2B) # of siblings < 1 & Age > 11 & 10 ≤ Fare price < 51	-> 0.36
(1B-2C) # of siblings < 1 & Age > 11 & Fare price ≥ 51	-> 0.535
(1C-2A) 1 ≤ # of siblings ≤ 4 & Fare price < 10	-> 0.35
(1C-2B) 1 ≤ # of siblings ≤ 4 & 10 ≤ Fare price < 51	-> 0.435
(1C-2D) 1 ≤ # of siblings ≤ 4 & Fare price ≥ 51	-> 0.43
(1D-2A) # of siblings > 4 & Fare price < 10	-> 0.57
(1D-2B) # of siblings > 4 & 10 ≤ Fare price < 51	-> 0.655
(1D-2D) # of siblings > 4 & Fare price ≥ 51	-> 0.65

- И будем строить новое дерево на основе этих конъюнкций, как обычно рекурсивно на основе энтропии потомков:



- Оказывается, что можно построить одно дерево (т.е. легко интерпретируемую модель) на основе сразу многих деревьев, не слишком жертвуя качеством.

СПАСИБО!

Спасибо за внимание!

