ДЕРЕВЬЯ ПРИНЯТИЯ РЕШЕНИЙ И ГРАДИЕНТНЫЙ БУСТИНГ

Сергей Николенко

НИУ ВШЭ — Санкт-Петербург
28 апреля 2017 г.

Random facts:

- 28 апреля 1503 г. французы потерпели тяжёлое поражение при Чериньоле, близ Бари; это было одно из первых сражений, выигранных благодаря аркебузирам, и оно считается началом «эры пороха»
- 28 апреля -- Всемирный день охраны труда

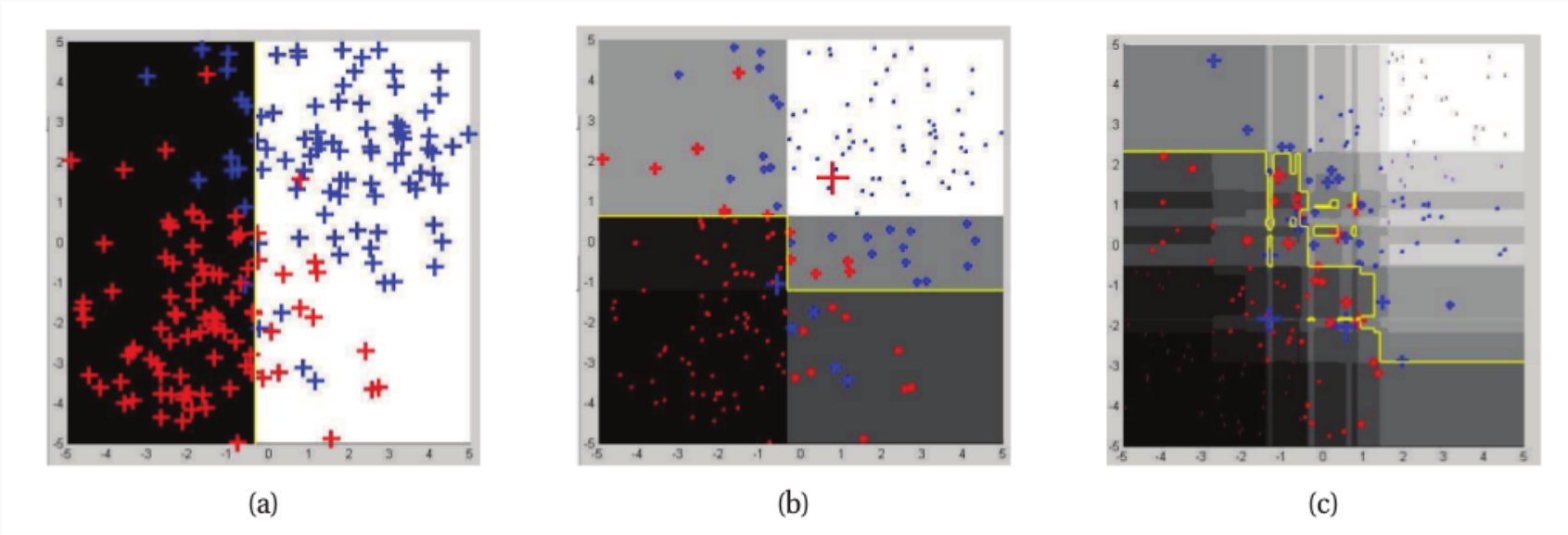
ДЕРЕВЬЯ ПРИНЯТИЯ РЕШЕНИЙ

WEAK LEARNERS: DECISION TREES

- 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.

WEAK LEARNERS: DECISION TREES

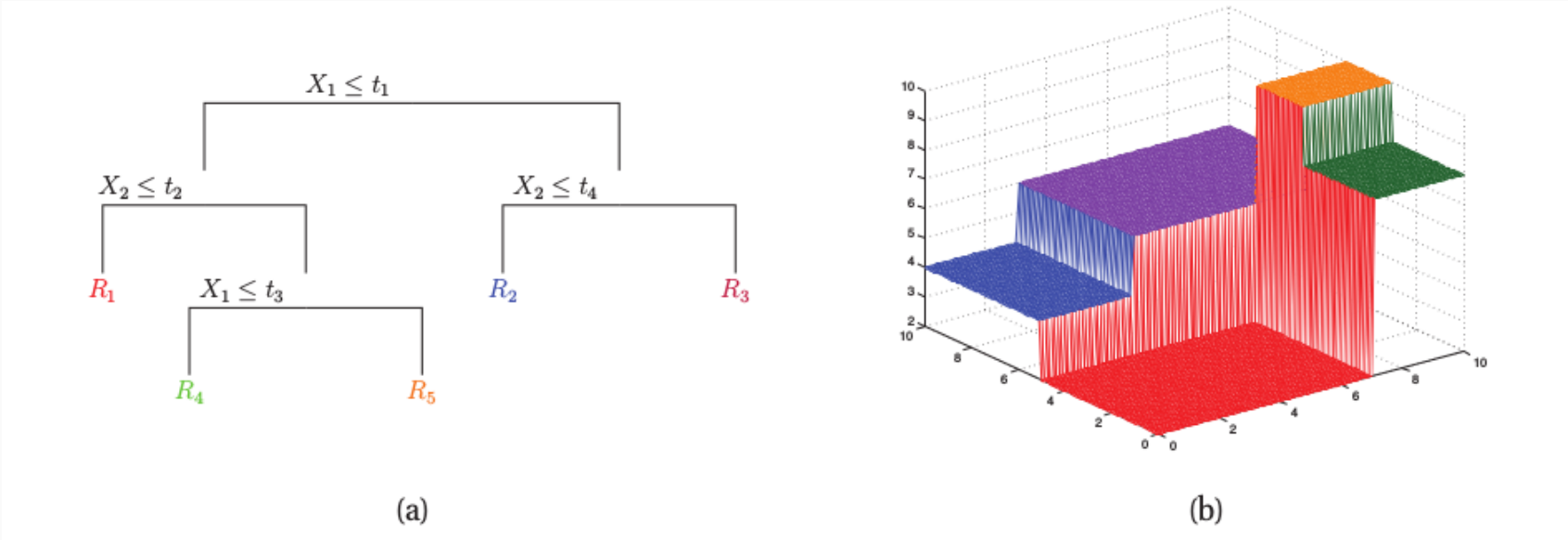
- Пример бустинга на пнях:



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

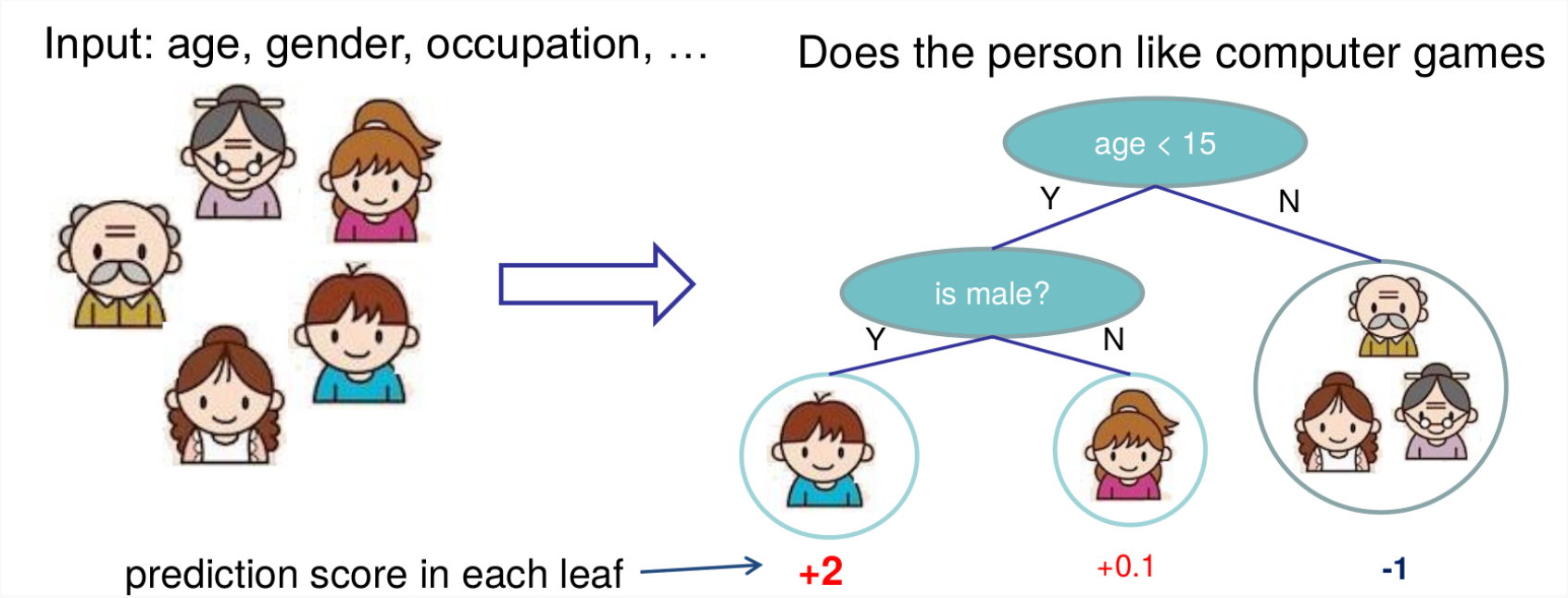
WEAK LEARNERS: DECISION TREES

- Дерево принятия решений — это дерево. 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.



WEAK LEARNERS: DECISION TREES

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



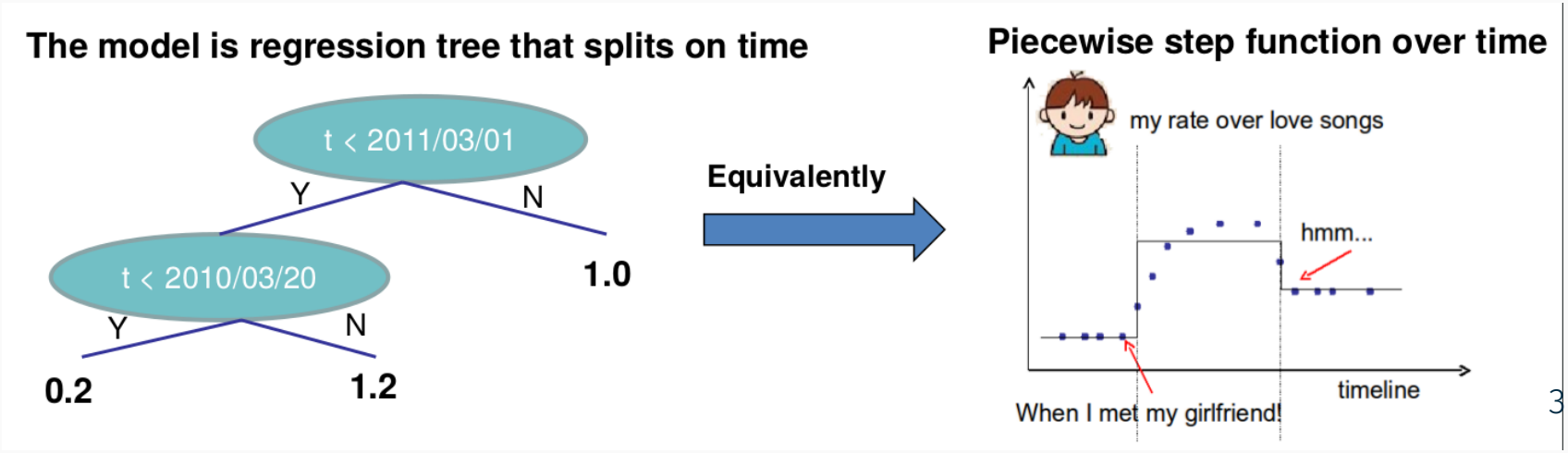
WEAK LEARNERS: DECISION TREES

- 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?

WEAK LEARNERS: DECISION TREES

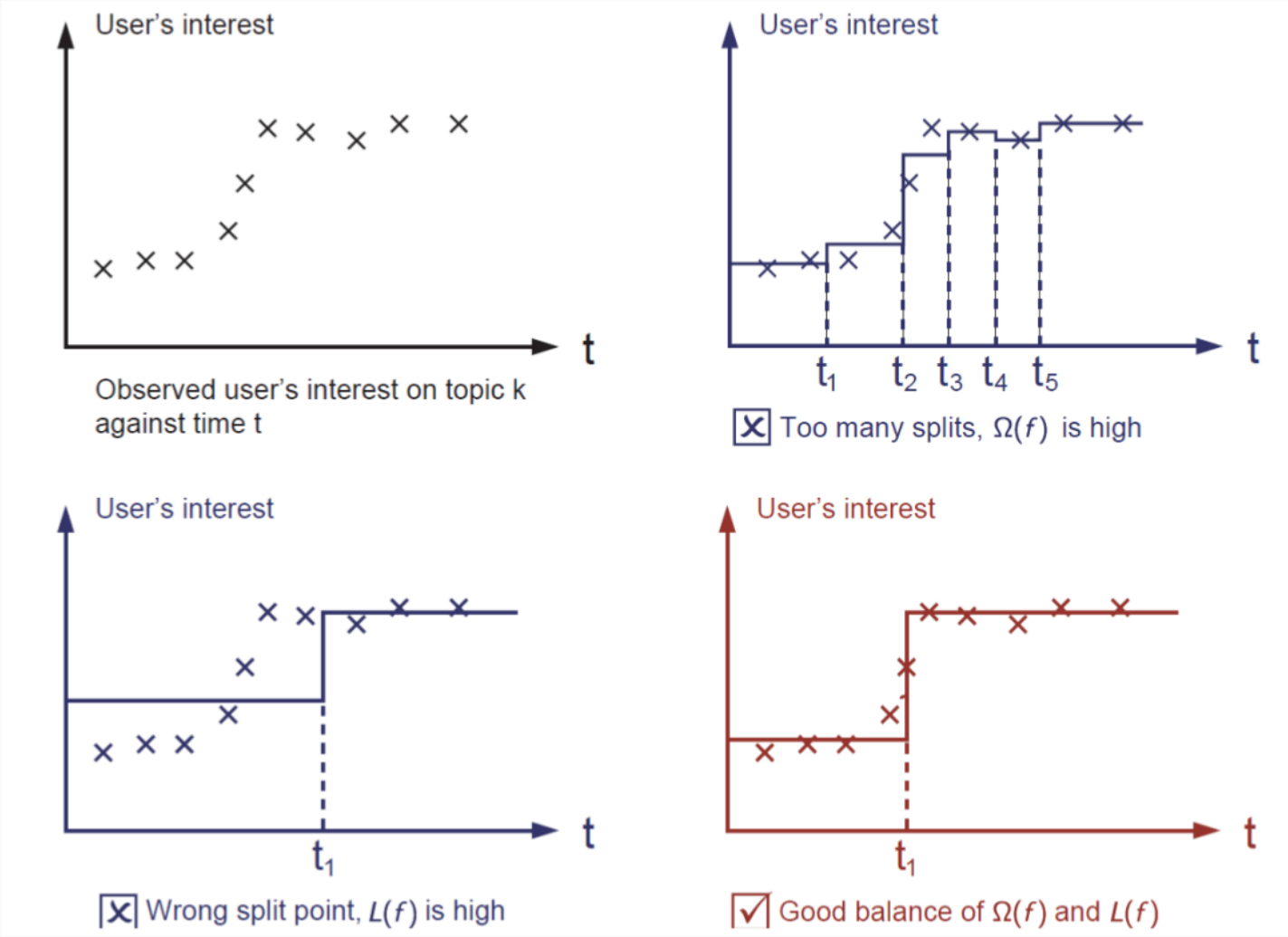
- Если атрибут бинарный или дискретный с небольшим числом значений, то просто по значениям.
- Если непрерывный – можно брать среднее арифметическое (тем самым минимизируя сумму квадратов).
- Выбирают атрибут, оптимизируя целевую функцию. Для задачи регрессии просто минимизируем среднеквадратическую ошибку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

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



WEAK LEARNERS: DECISION TREES

- Как обучить дерево:



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

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

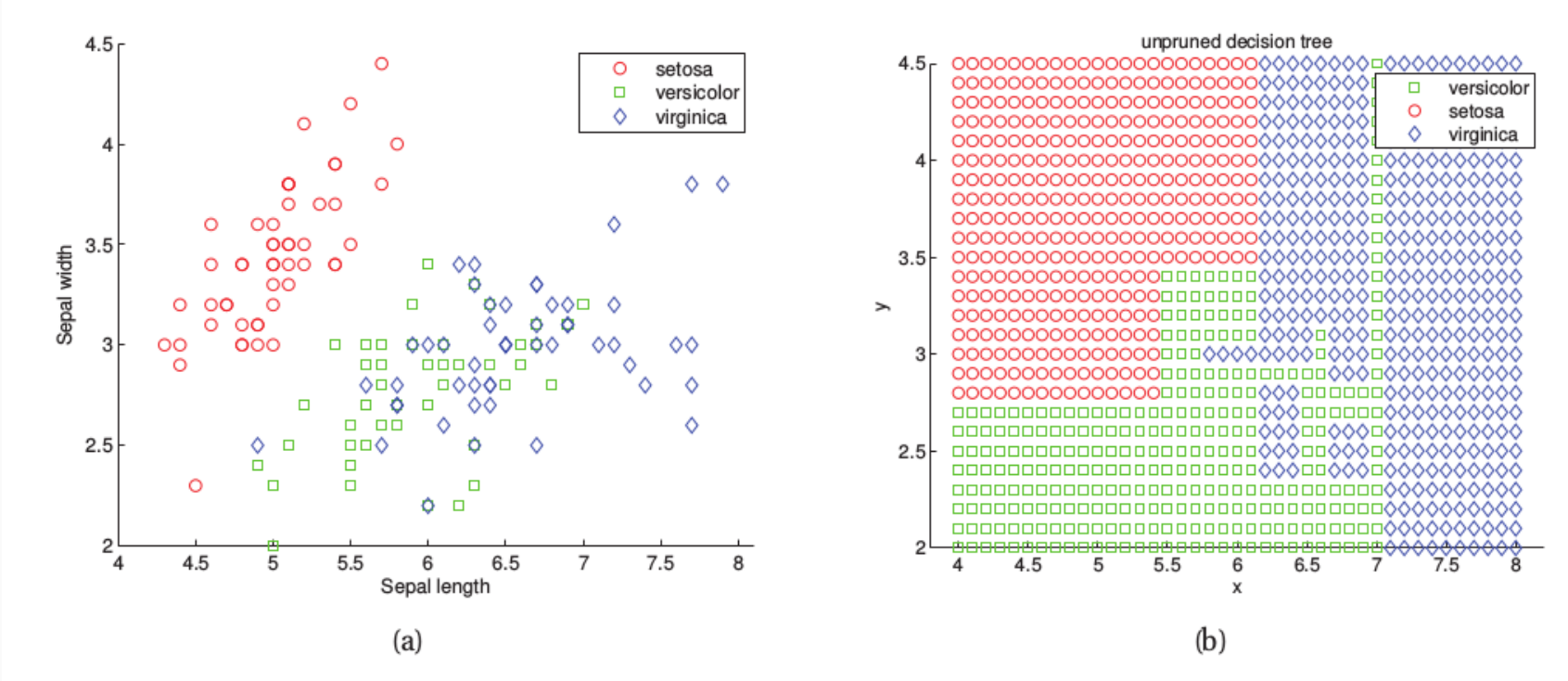
- Иногда ещё используют *индекс Джини* (Gini index): $G(\mathbf{X}_\tau) = \sum_{k=1}^{K} p_{\tau,k}(1 - p_{\tau,k})$.

WEAK LEARNERS: DECISION TREES

- 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): поддерево 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).

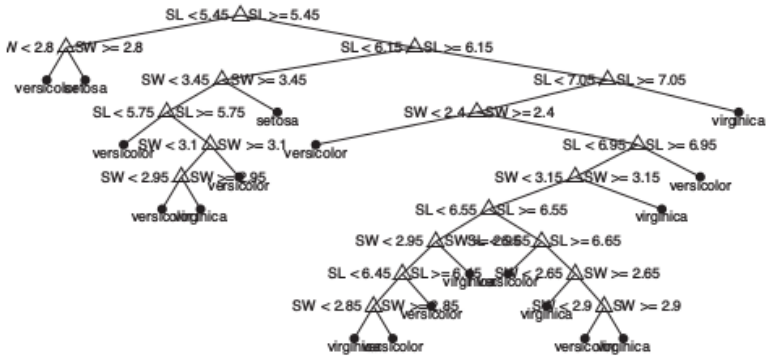
ПРИМЕР

- Пример на датасете iris.
- Вход:

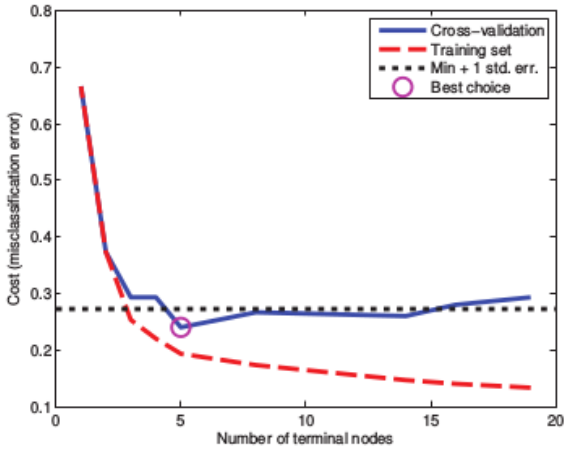


ПРИМЕР

- Слишком глубокое дерево и его ошибки:



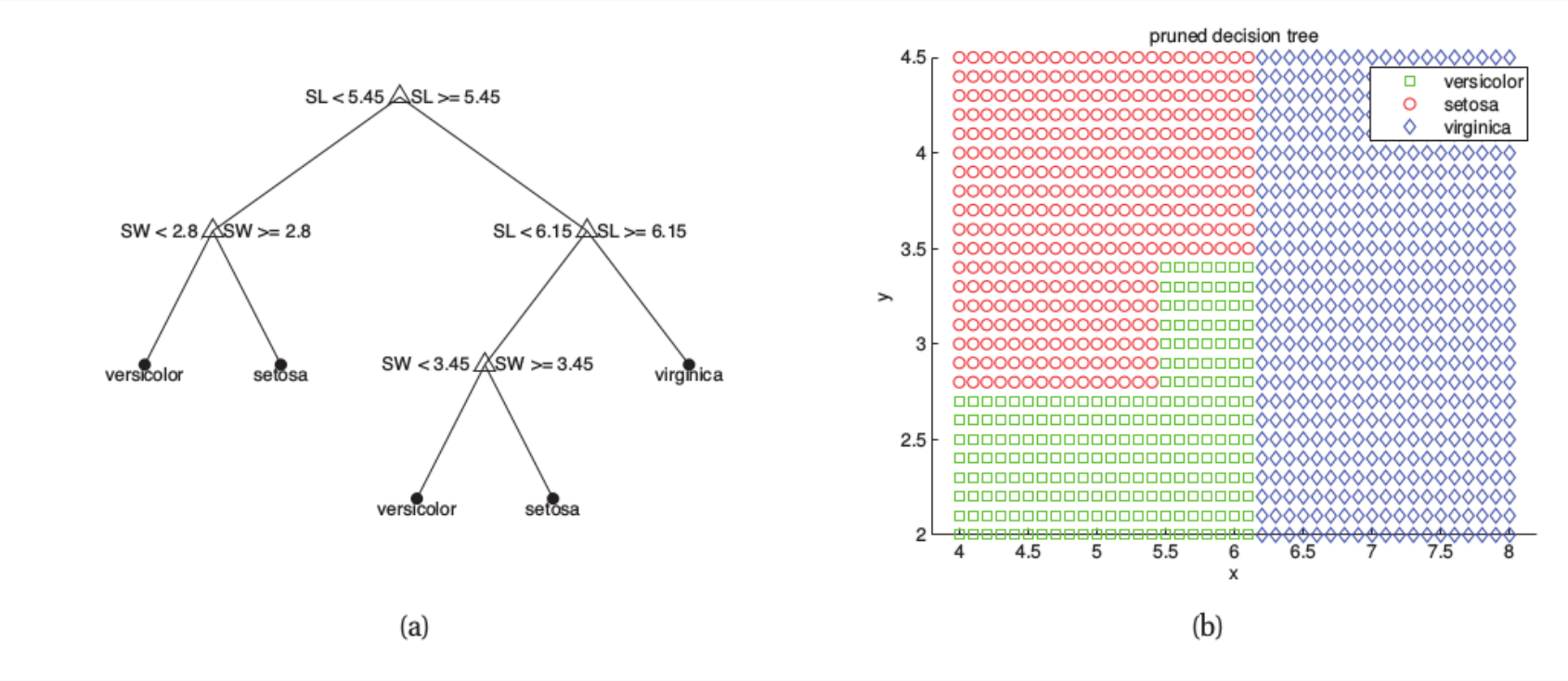
(a)



(b)

ПРИМЕР

- Обрезанное дерево:



ГРАДИЕНТНЫЙ БУСТИНГ

ГРАДИЕНТНЫЙ БУСТИНГ

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

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

- 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

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

ГРАДИЕНТНЫЙ БУСТИНГ

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

$$\begin{aligned}
\widehat{y}_i^{(0)} &= 0, \\
\widehat{y}_i^{(1)} &= f_1(x_i) = \widehat{y}_i^{(0)} + f_1(x_i), \\
\widehat{y}_i^{(2)} &= f_1(x_i) + f_2(x_i) = \widehat{y}_i^{(1)} + f_1(x_2), \\
&\dots,
\end{aligned}$$

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

ГРАДИЕНТНЫЙ БУСТИНГ

- Чтобы добавить следующее дерево, нужно оптимизировать

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

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

- Например, для квадратов отклонений

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

ГРАДИЕНТНЫЙ БУСТИНГ

- 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

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

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

ГРАДИЕНТНЫЙ БУСТИНГ

- 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:

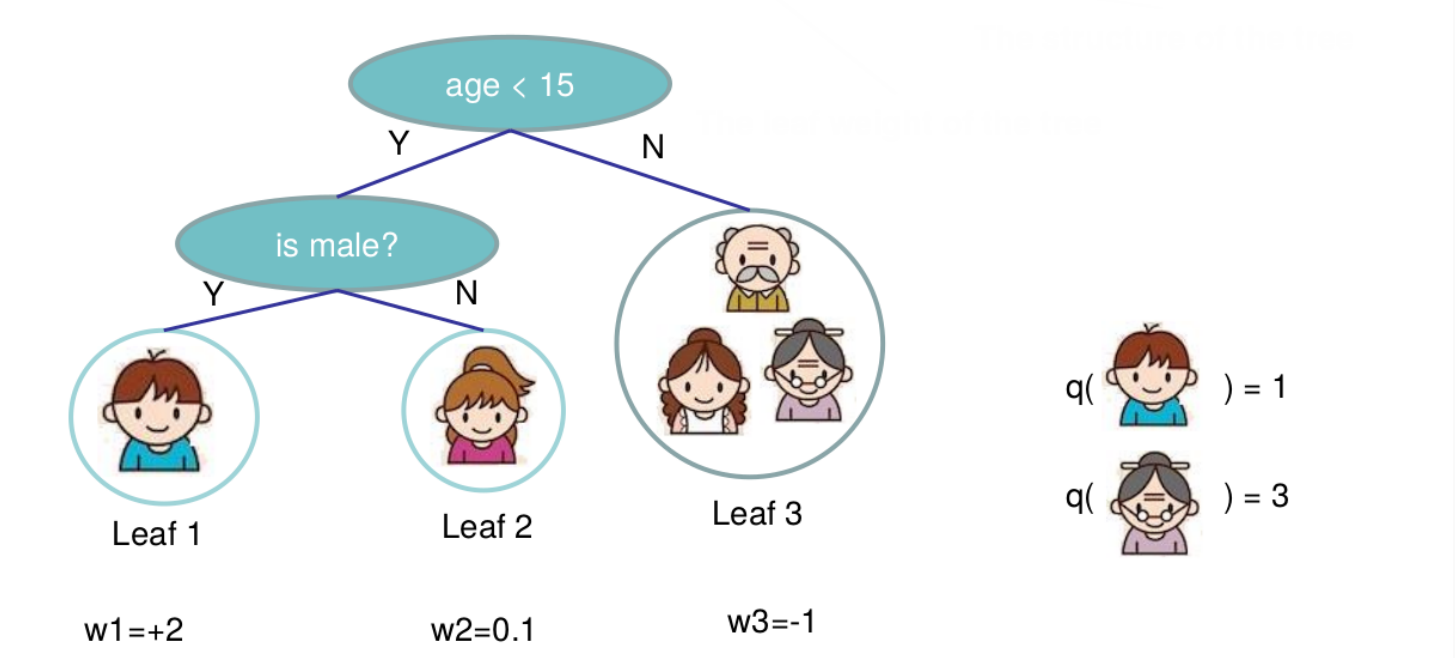
$$\text{Obj}^{(t)} \approx \sum_{i=1}^{N} \left(g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right) + \Omega(f_t).$$

- 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:

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



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

ГРАДИЕНТНЫЙ БУСТИНГ

- Теперь перегруппируем слагаемые относительно листьев:

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

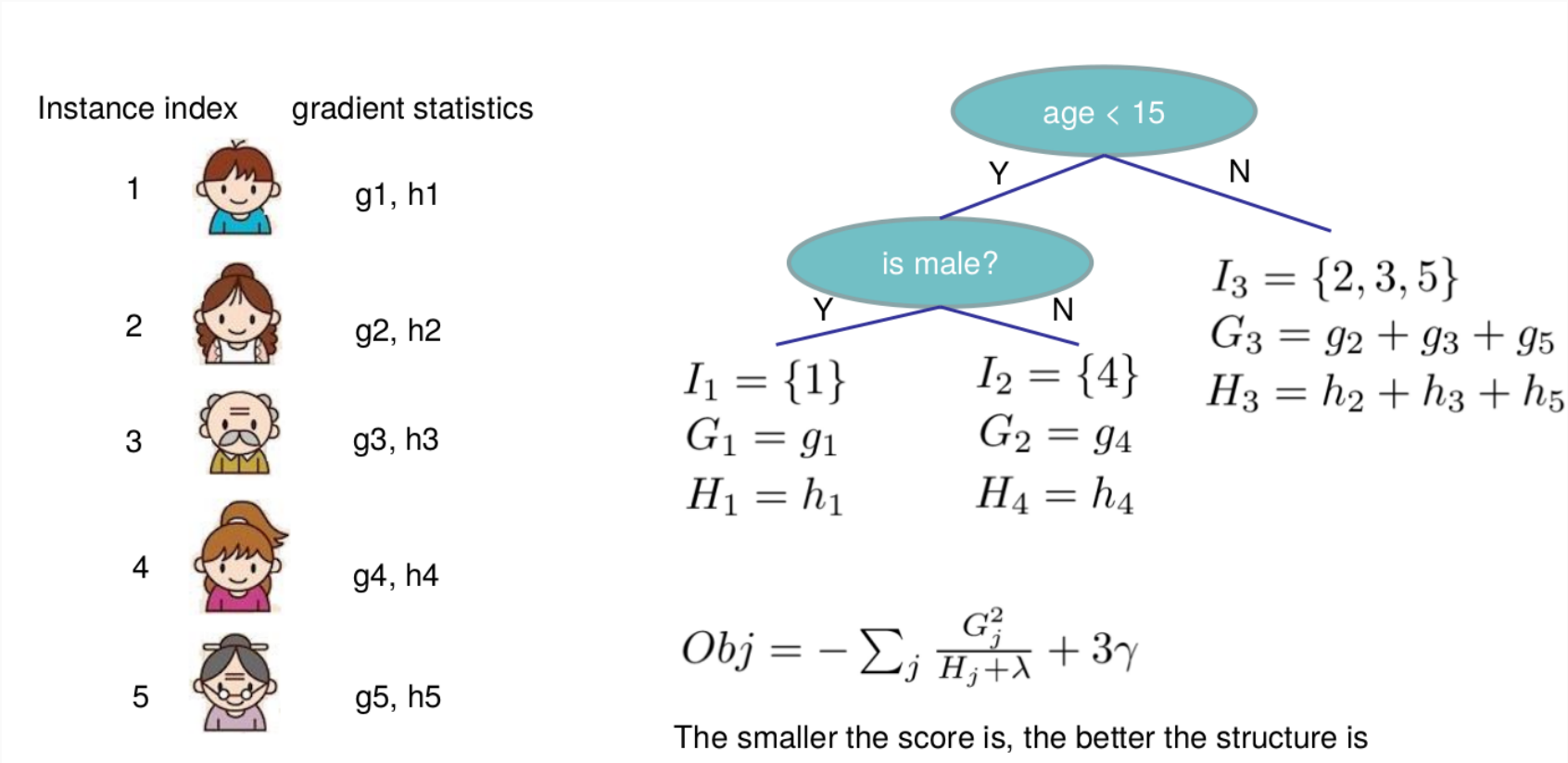
где $I_j = \{i \mid q(x_j) = i\}$, $G_i = \sum_{i \in I_j} g_i$, $H_i = \sum_{i \in I_j} h_i$.

- Это сумма T независимых квадратичных функций, так что

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

ГРАДИЕНТНЫЙ БУСТИНГ

- Пример из (Chen, Guestrin):



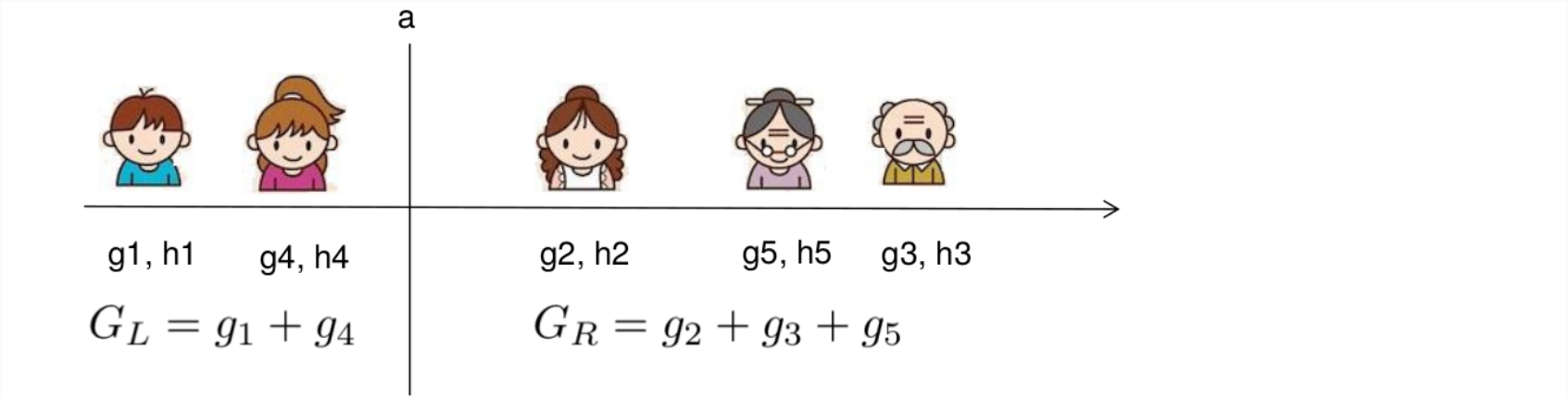
ГРАДИЕНТНЫЙ БУСТИНГ

- 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

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

ГРАДИЕНТНЫЙ БУСТИНГ

- Самое лучшее разбиение – то, которое максимизирует gain:



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

MATRIXNET

- 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.

СПАСИБО!

Спасибо за внимание!