ДЕРЕВЬЯ И БУСТИНГ

Сергей Николенко

СПбГУ — Санкт-Петербург

11 февраля 2025 г.

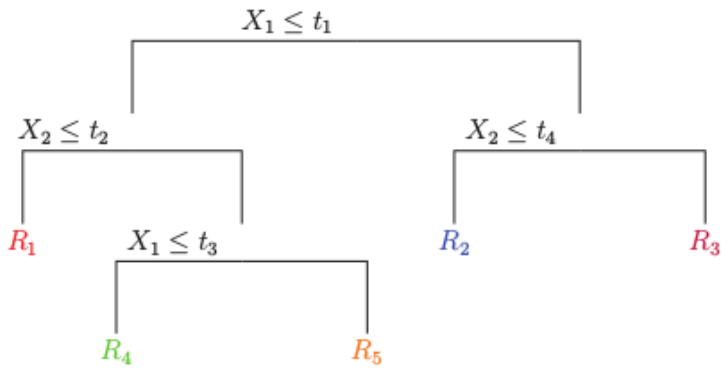
Random facts:

- 11 февраля — Международный день женщин и девочек в науке, учреждённый Генеральной ассамблеей ООН по инициативе ЮНЕСКО в 2015 году
- 11 февраля 660 года до н.э. взошёл на престол император Дзимму; ведомый вороном ятагарасу, Дзимму совершил поход из Химука на острове Кюсю, куда его предки спустились с Небес, в провинцию Ямато на Хонсю, где он основал японское государство
- 11 февраля 2016 г. объявлено об экспериментальном открытии гравитационных волн коллаборациями LIGO и VIRGO; гравитационные волны предсказывались общей теорией относительности, но проверить не удавалось из-за очень малой величины
- 11 февраля 1978 г. в Китае был отменён запрет на чтение книг Аристотеля, Шекспира и Диккенса; вскоре «культурная революция» стала переосмысляться как национальная трагедия (а ответственность за неё начала возлагаться на Мао Цзэдуна)
- 11 февраля 1858 г. 14-летняя Бернадетта Субиру заметила озарённый светом грот неподалёку от Лурда; внутри Бернадетта увидела (и позже видела ещё 17 раз) «что-то белое, похожее на барышню»; место явления Богородицы св. Бернадетте стало центром католического паломничества; в Лурд приезжает до 5 млн паломников в год

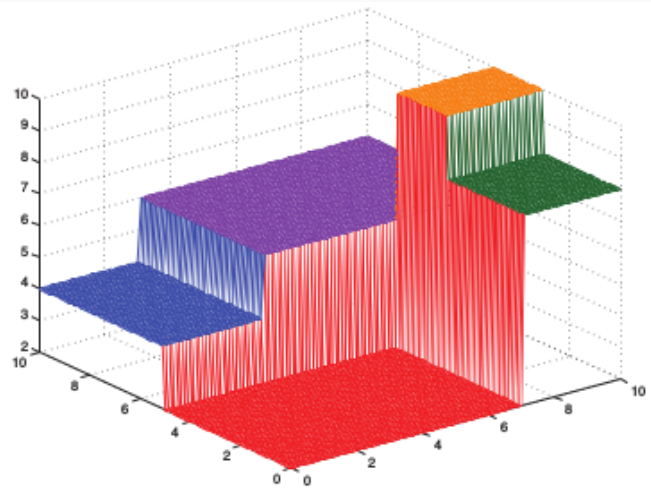
Деревья принятия решений

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.



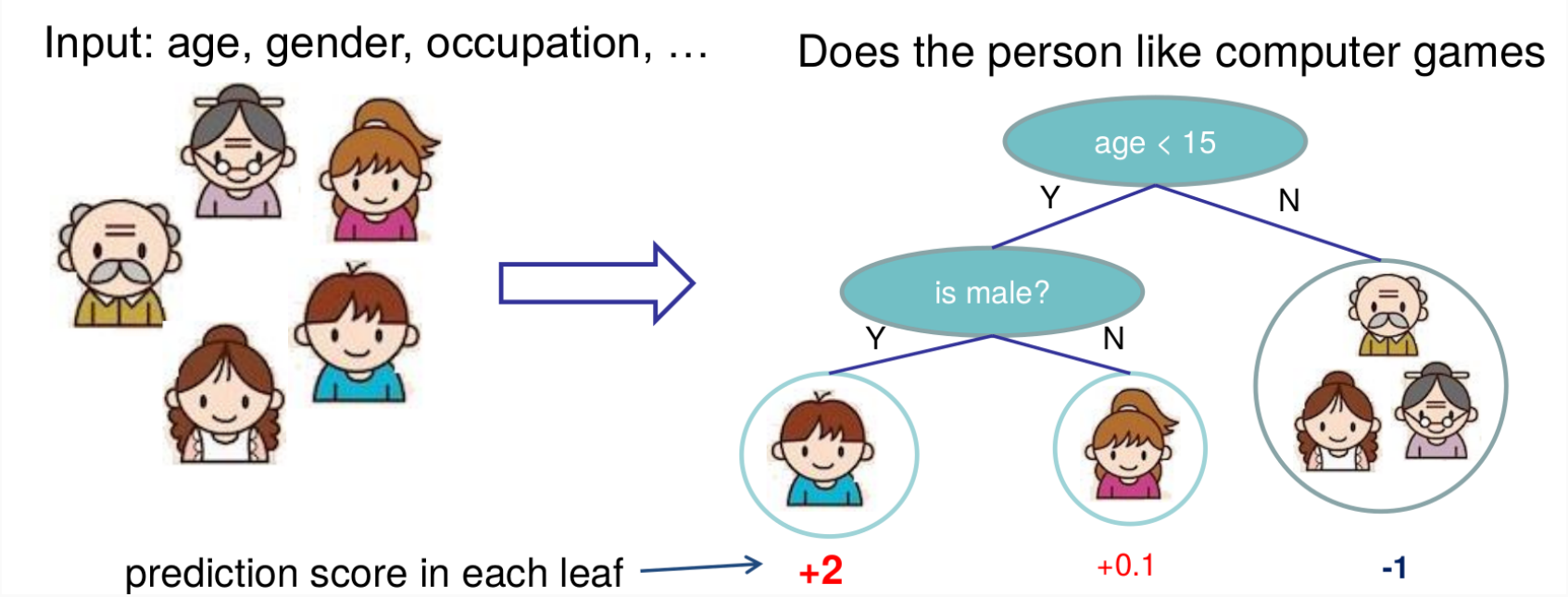
(a)



(b)

Деревья принятия решений

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



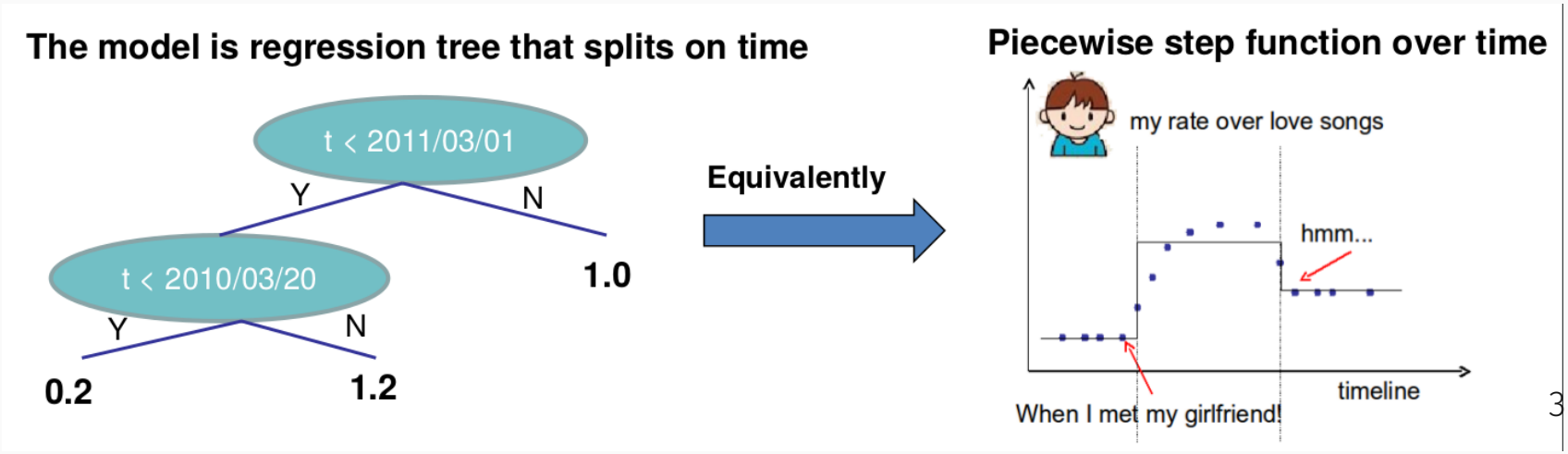
Деревья принятия решений

- 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

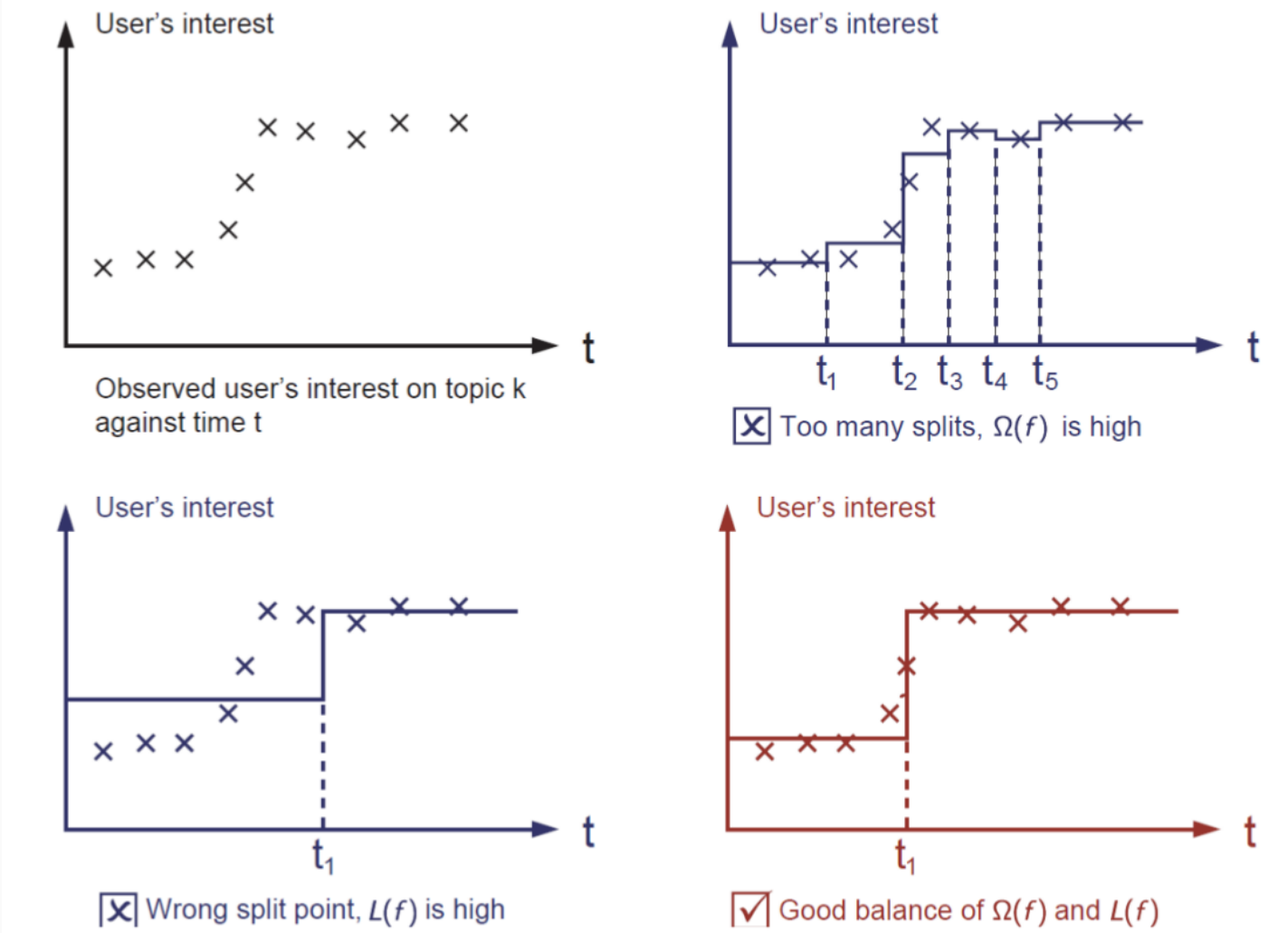
- Если атрибут бинарный или дискретный с небольшим числом значений, то просто по значениям.
- Если непрерывный – можно брать среднее арифметическое (тем самым минимизируя сумму квадратов).
- Выбирают атрибут, оптимизируя целевую функцию. Для задачи регрессии просто минимизируем среднеквадратическую ошибку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

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



Деревья принятия решений

- Как обучить дерево:



Деревья принятия решений

- 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*:

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

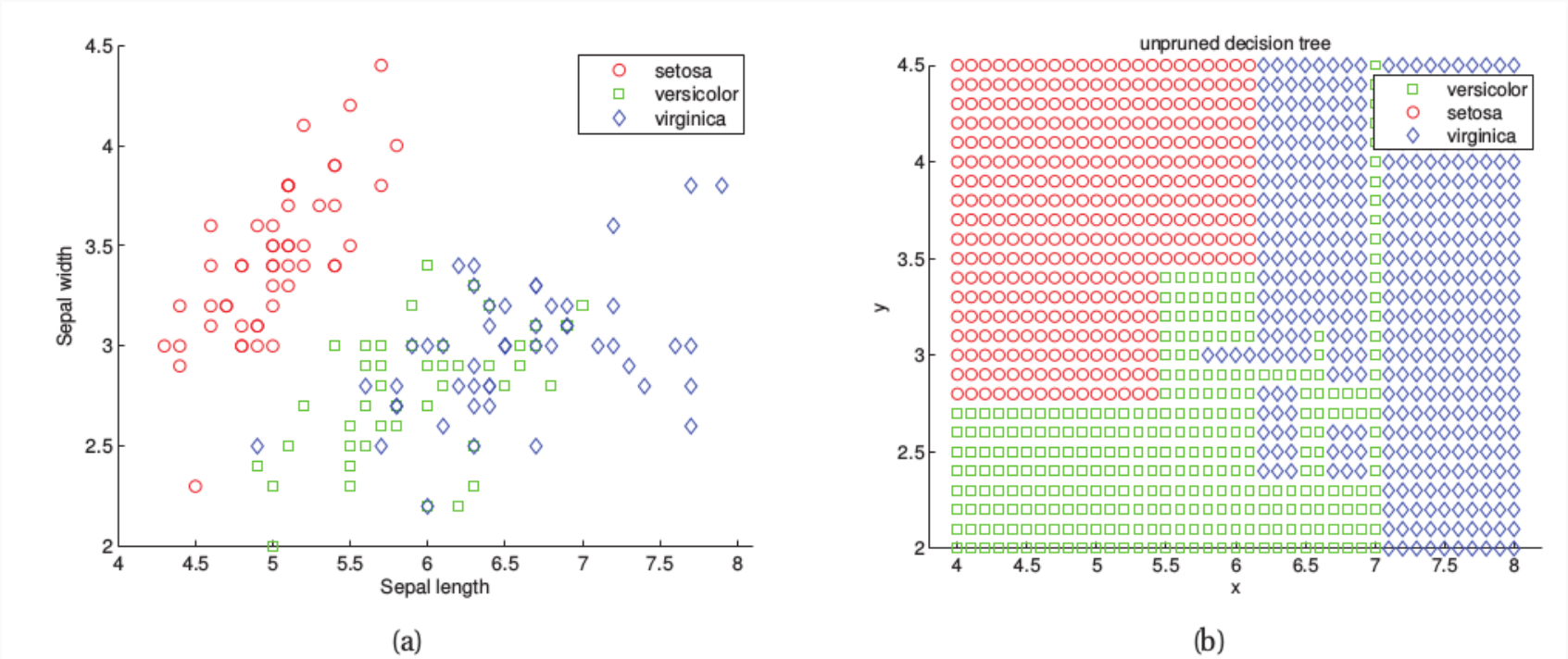
- Иногда ещё используют *индекс Джини* (Gini index): $G(\mathbf{X}_\tau) = \sum_{k=1}^{K} p_{\tau,k}(1 - p_{\tau,k})$.

Деревья принятия решений

- 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).

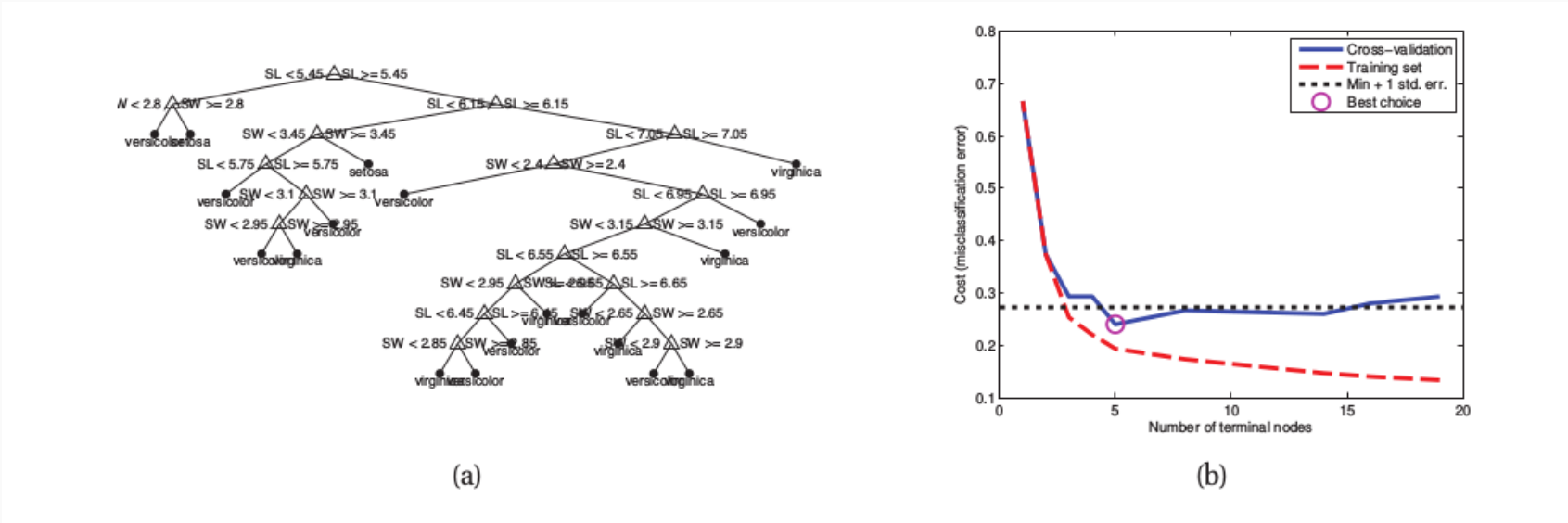
Пример

- Пример на датасете iris.
- Вход:



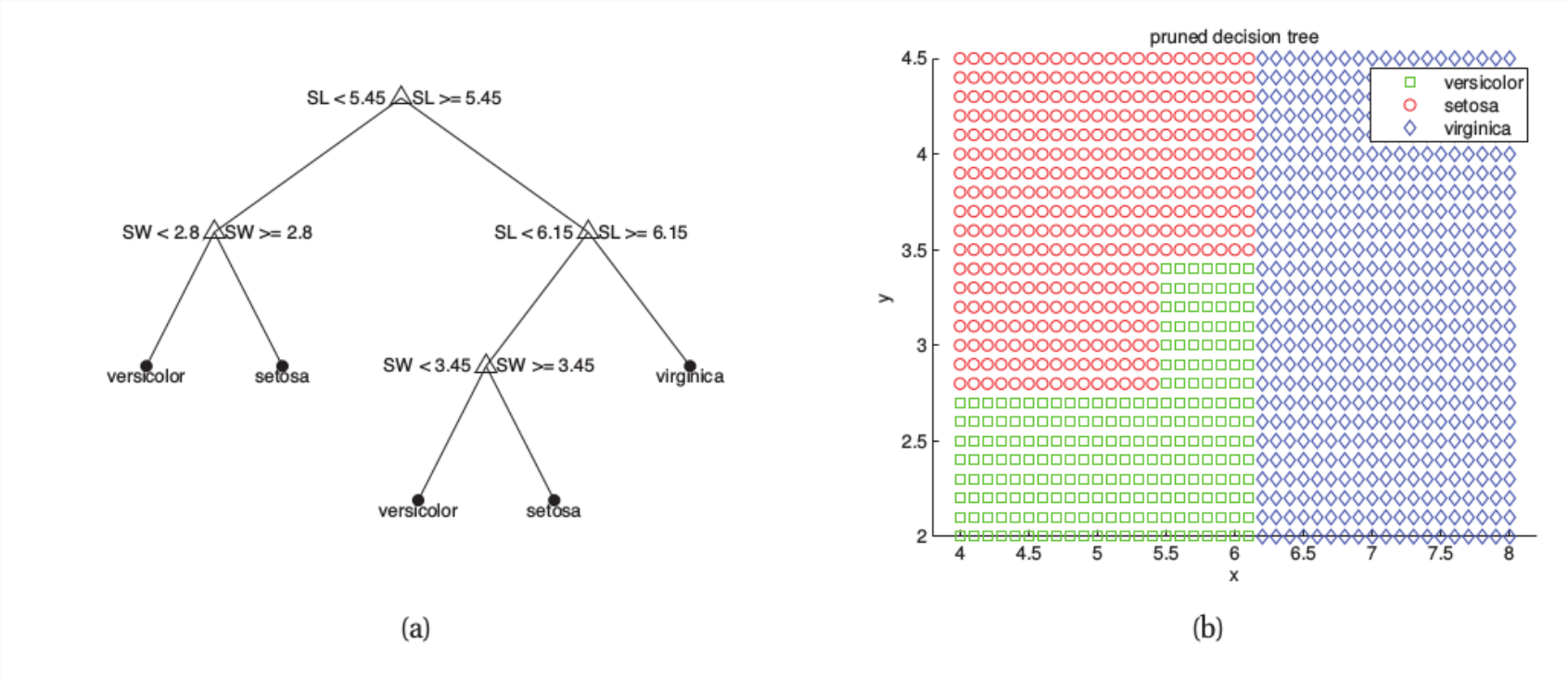
Пример

- Слишком глубокое дерево и его ошибки:



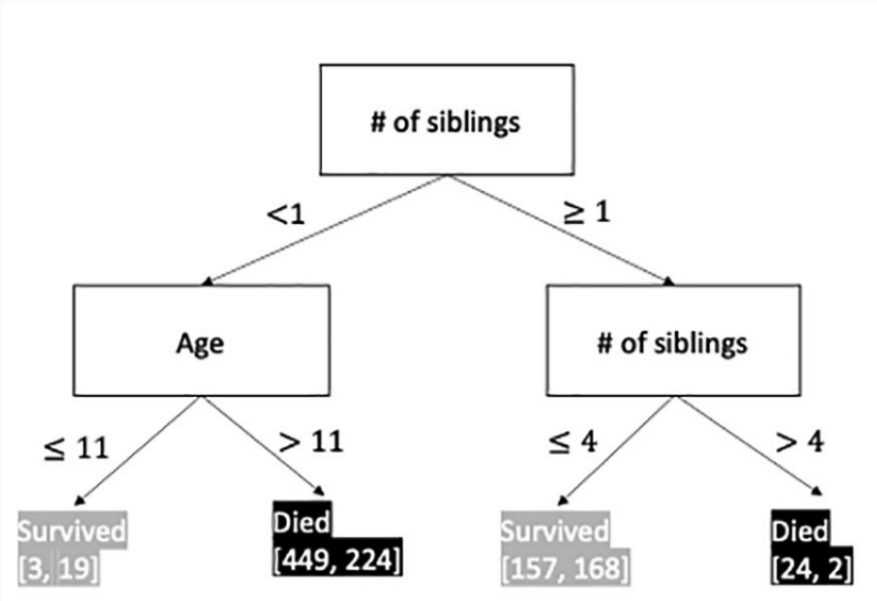
Пример

- Обрезанное дерево:



Титаник

- Пример на датасете о пассажирах «Титаника»:

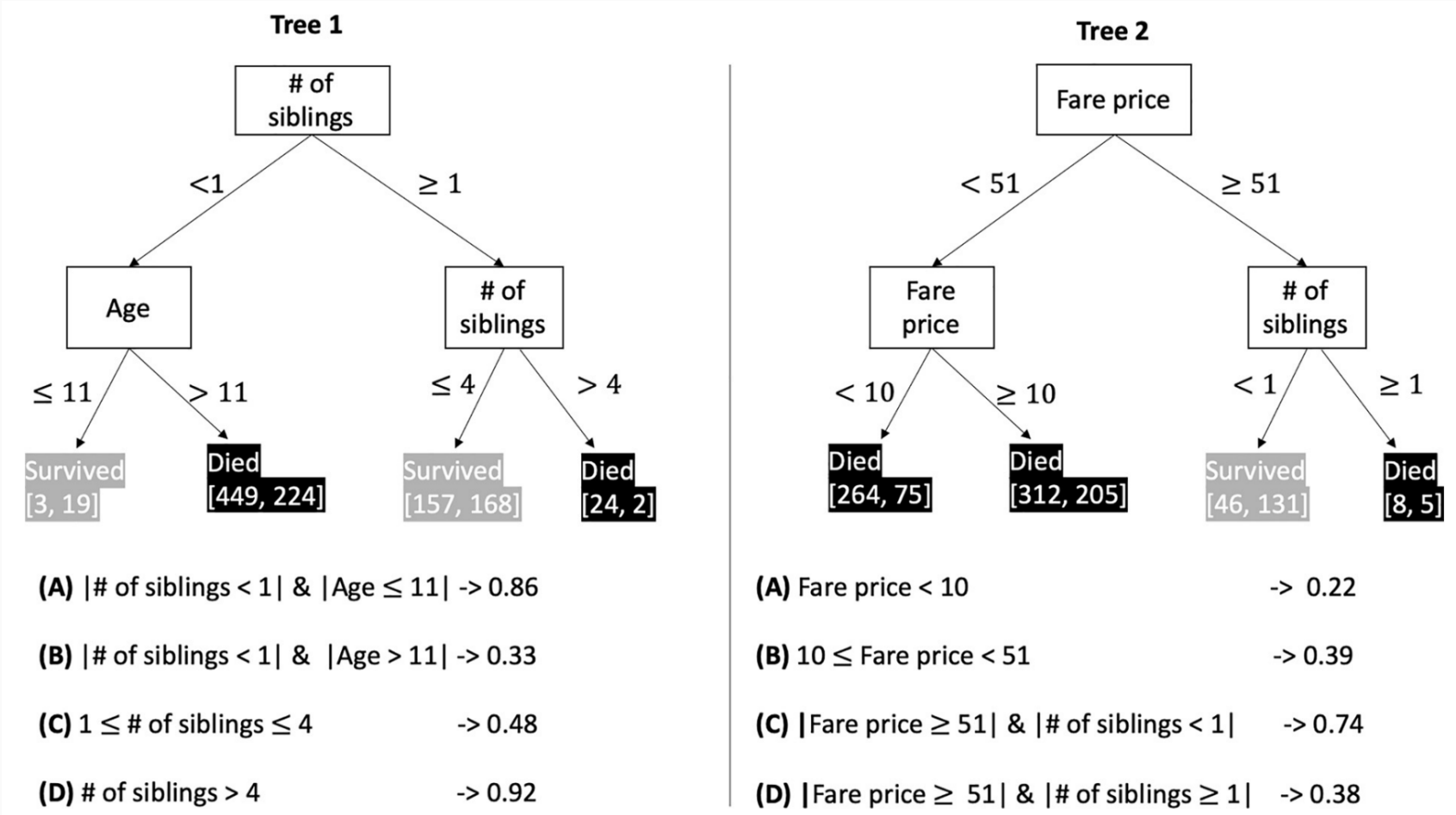


Rules conjunction	Survival probability
(1) |# of siblings < 1| & |Age ≤ 11|	0.86
(2) |# of siblings < 1| & |Age > 11|	0.33
(3) 1 < # of siblings ≤ 4	0.48
(4) # of siblings > 4	0.92

- Обратите внимание, насколько хорошо интерпретируется дерево.

Титаник

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



- К этой мысли мы ещё вернёмся...

Объединение моделей

Как объединять модели

- 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?

Основные подходы

- 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).

Комбинация моделей и байесовское усреднение

- 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.

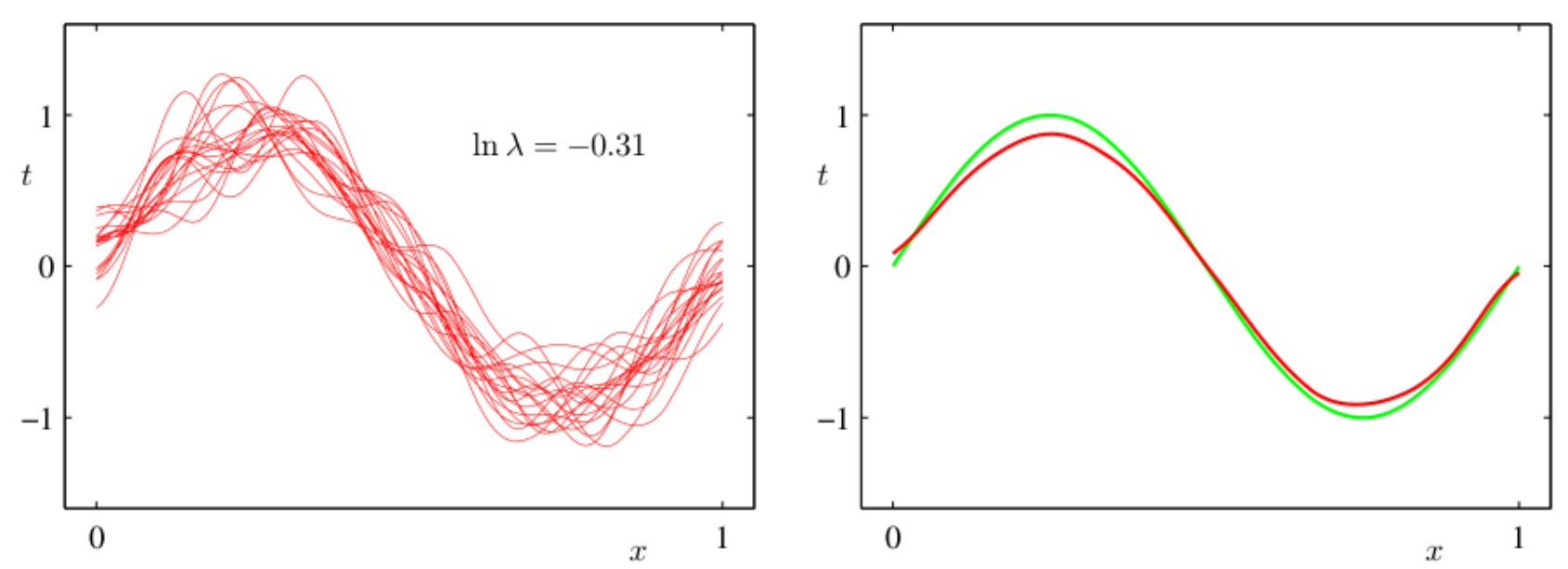
Комбинация моделей и байесовское усреднение

- А байесовское усреднение будет выглядеть как

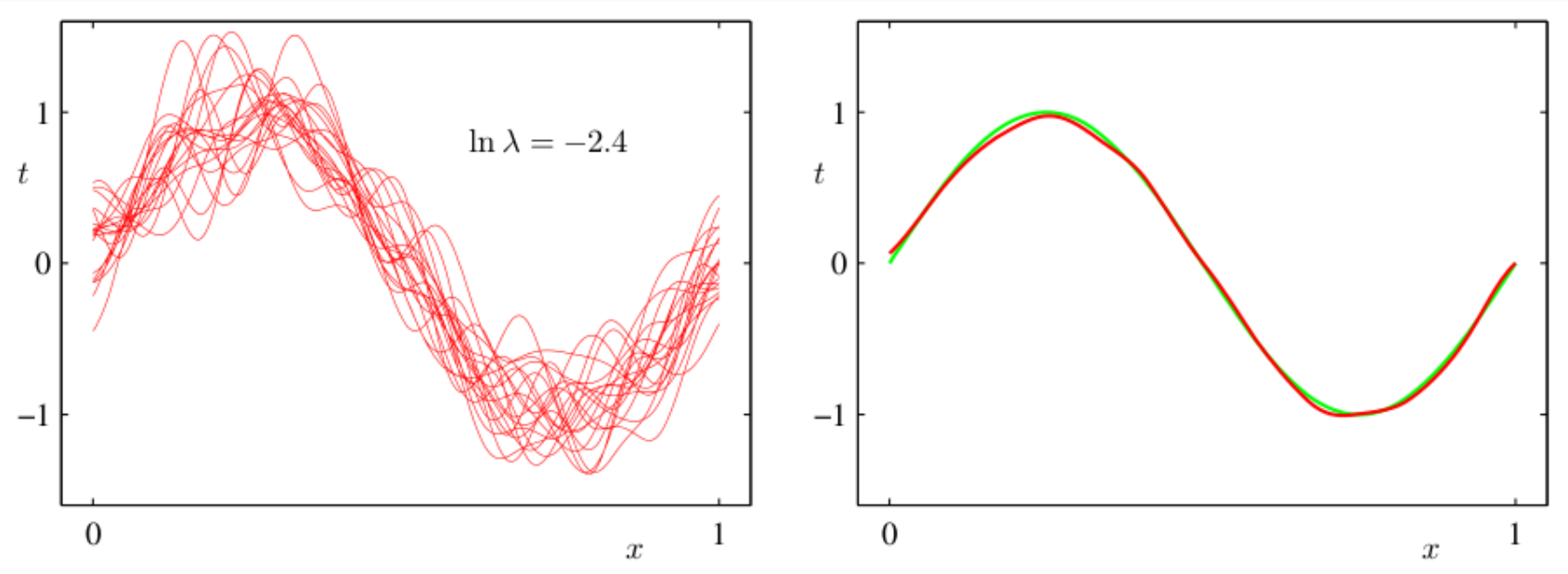
$$p(\mathbf{X}) = \sum_{h=1}^{H} p(\mathbf{X} \mid h) p(h).$$

- Смысл теперь в том, что генерирует $\mathbf{X}$ только одна модель, но мы просто не знаем какая именно; когда $\mathbf{X}$, апостериорные распределения $p(h \mid \mathbf{X})$ 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



Bagging

- 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?

Bagging

- 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*.

Bagging

- 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...

Bagging

- 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

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

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

$$E_{\mathrm{avg}} = \frac{1}{M}\sum_{m=1}^{M} \mathrm{E}_{\mathbf{x}}\left[\epsilon_m(\mathbf{x})^2\right].$$

Bagging

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

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

- А ошибка комитета – это

$$E_{\text{com}} = \mathrm{E}_{\mathbf{x}} \left[\left(\frac{1}{M} \sum_{m=1}^{M} (y_m(\mathbf{x}) - h(\mathbf{x})) \right)^2 \right] =$$

$$= \mathrm{E}_{\mathbf{x}} \left[\left(\frac{1}{M} \sum_{m=1}^{M} \epsilon_m(\mathbf{x}) \right)^2 \right].$$

Bagging

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

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

Bagging

- $E_{\text{com}} = \frac{1}{M} E_{\text{avg}}$!
- 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: AdaBoost

AdaBoost

- 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

- 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).

AdaBoost

- 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

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

  2. вычисляем

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

  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 $Y_M(\mathbf{x}) = \operatorname{sign} \left(\sum_{m=1}^{M} \alpha_m y_m(\mathbf{x}) \right)$.

AdaBoost

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

Теоретические свойства AdaBoost

- 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

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

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

Теоретические свойства AdaBoost

- 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 $y_l(\mathbf{x})$.

Теоретические свойства AdaBoost

- 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.

Теоретические свойства AdaBoost

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

$$E = e^{-\frac{\alpha_m}{2}} \sum_{\text{correct}} w_n^{(m)} + e^{\frac{\alpha_m}{2}} \sum_{\text{wrong}} w_n^{(m)} =$$

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

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

Теоретические свойства AdaBoost

- 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.

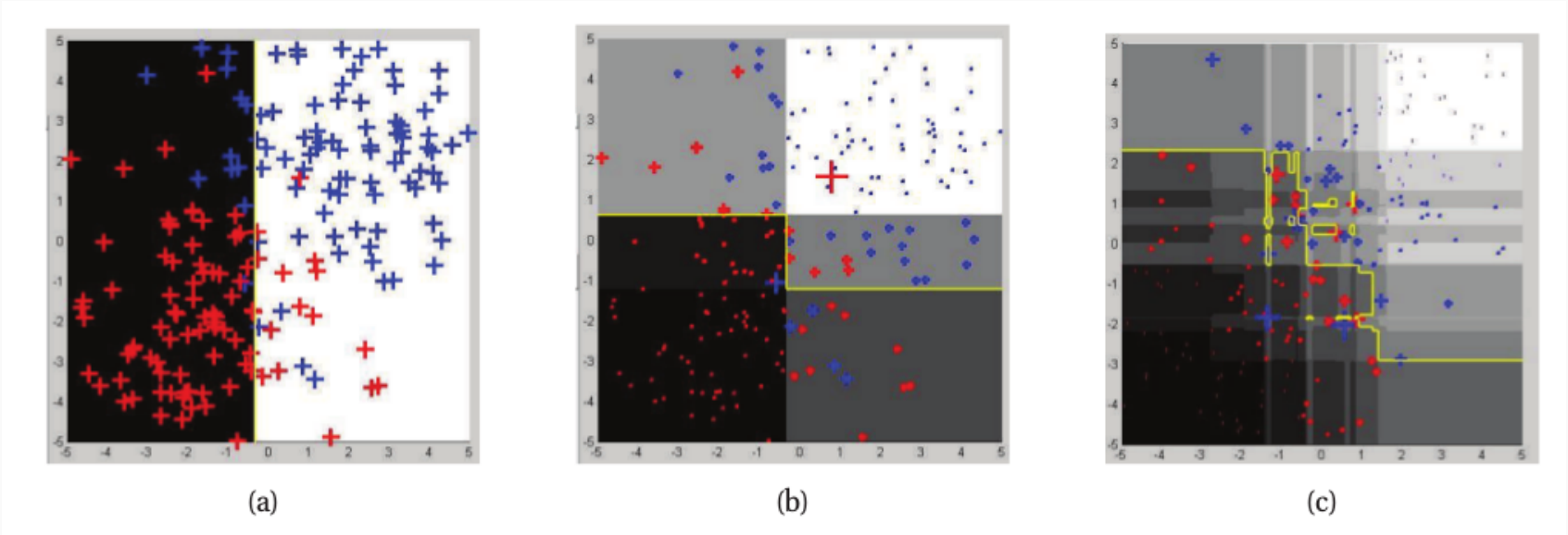
- Таким образом, бустинг можно рассматривать как оптимизацию экспоненциальной ошибки.

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).

Градиентный бустинг

Градиентный бустинг

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

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

- 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:

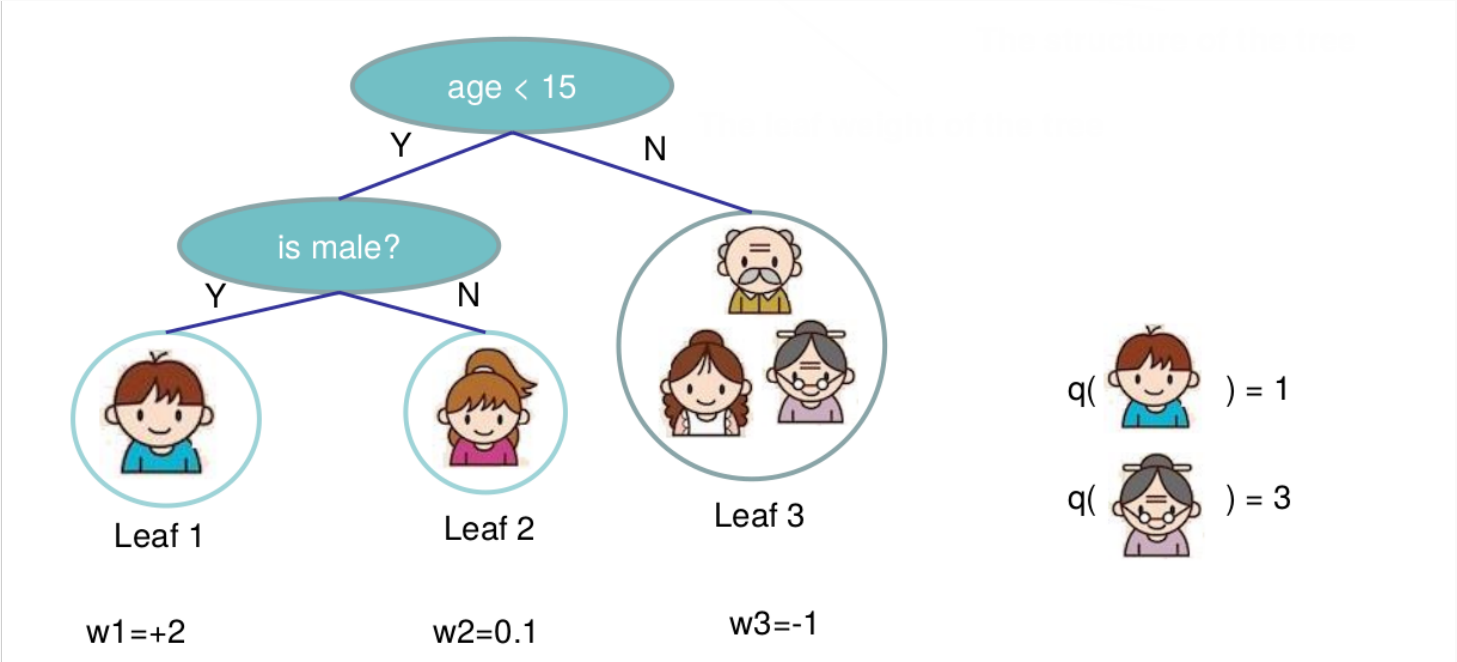
$$\text{Obj}^{(t)} \approx \sum_{i=1}^{N} \left(g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right) + \Omega(f_t).$$

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

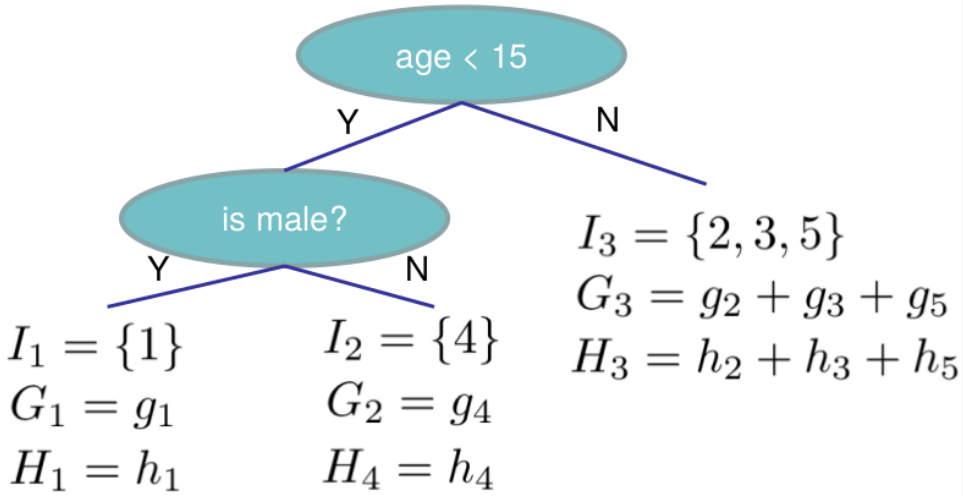
- Это сумма T независимых квадратичных функций, так что

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

Градиентный бустинг

- Пример из (Chen, Guestrin):

Instance index		gradient statistics
1		g1, h1
2		g2, h2
3		g3, h3
4		g4, h4
5		g5, h5



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

The smaller the score is, the better the structure is

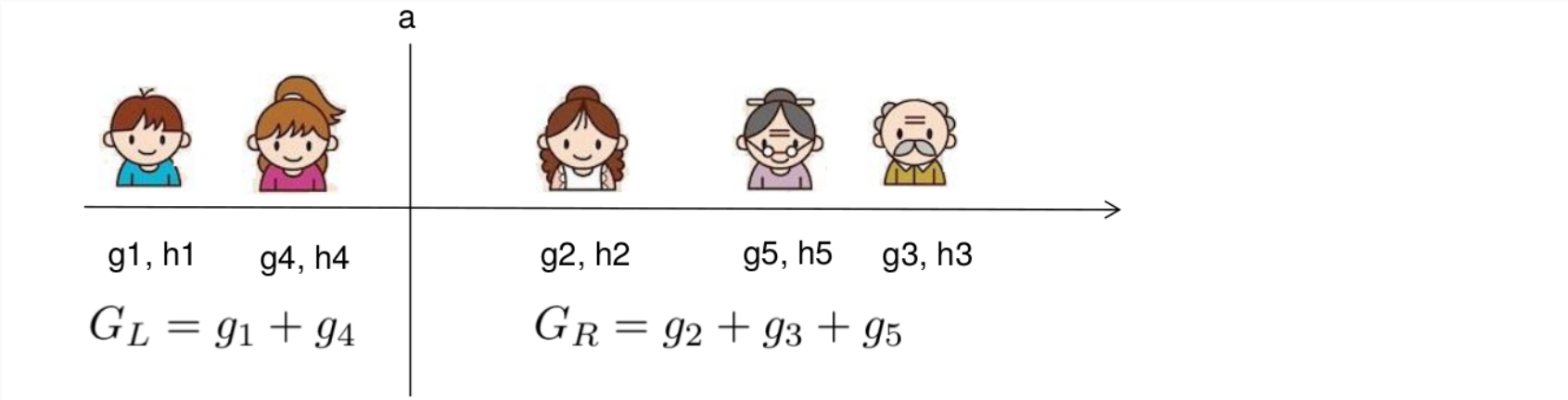
Градиентный бустинг

- 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.

CatBoost

- Разработанный в «Яндекс» вариант градиентного бустинга для ранжирования – *MatrixNet*, позже *CatBoost* (Prokhorenkova et al., 2018):
  - *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).
- ”Cat” означает ”category”, так что тут ещё есть интересный разговор про категориальные признаки...

CatBoost

- Что делать с категориальными признаками? One-hot encoding не работает, когда значений много
- Target statistics: давайте заменим каждое значение на ожидаемое среднее

$$\hat{x}_k^i = \frac{\sum_{j=1}^n \left[x_j^i = x_k^i\right] y_j + ap}{\sum_{j=1}^n \left[x_j^i = x_k^i\right] + a},$$

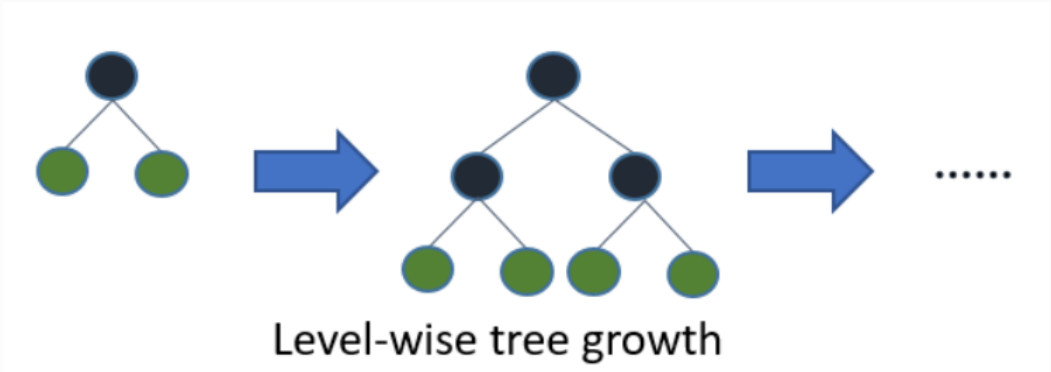
  то есть мы оцениваем
- Но это ведёт к утечкам. Крайний пример: если для бинарной целевой переменной $y \in \{0, 1\}$ все значения признака x^i разные, и он вообще не важен, $p\left(y = 1 \middle| x^i = v\right) = \frac{1}{2}$, достаточно будет сделать сплит с порогом $t = \frac{0.5+ap}{1+a}$, и обучающая выборка разделится идеально; а для любого тестового примера будет $\hat{x}^i = p$
- Хотелось бы, чтобы $\mathbb{E}\left[\hat{x}^i \mid y = v\right] = \mathbb{E}\left[\hat{x}_k^i \mid y_k = v\right]$
- Что делать?

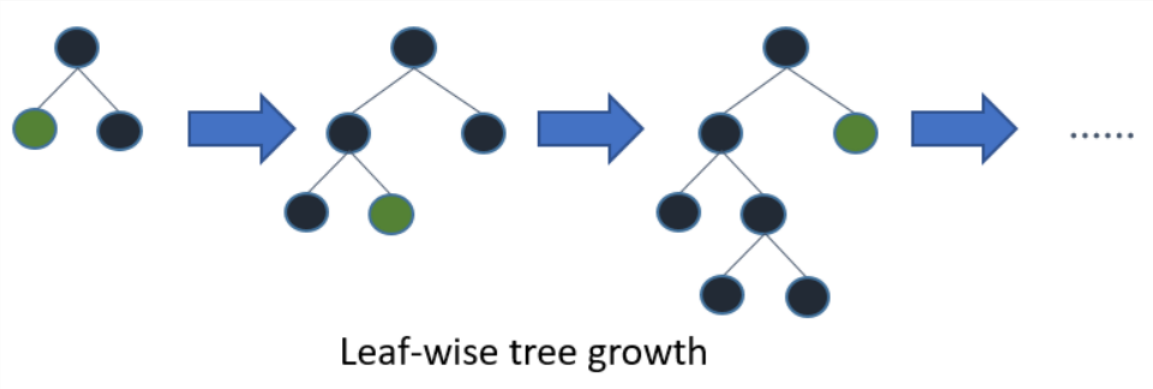
CatBoost

- Можно, конечно, просто вычислять статистику на другом подмножестве, без утечки – holdout TS; это решит проблему, но сильно сократит датасет, что нежелательно
- Leave-one-out TS: давайте для примера $\mathbf{x}_k$ считать статистику на $D_k = D \setminus \{\mathbf{x}_k\}$; но это тоже ведёт к утечкам – рассмотрим константный признак $\mathbf{x}^i = a$, например
- CatBoost предлагает ordered TS: упорядочим примеры и будем считать статистику на $D_k = \{\mathbf{x}_j | j < k\}$; перестановку можно менять между шагами градиентного бустинга

LightGBM

- LightGBM — до сих пор стандартная библиотека; градиентный бустинг как в XGBoost с небольшими изменениями
- Выращивают деревья жадно по листьям, а не уровень за уровнем





LightGBM

- Exclusive Feature Bundling (EFB): признаки почти всегда разреженные, и если признаки не принимают ненулевые значения одновременно, их можно объединить в один

Algorithm 3: Greedy Bundling

Input: F: features, K: max conflict count
Construct graph G
searchOrder $\leftarrow G$.sortByDegree()
bundles $\leftarrow \{\}$, bundlesConflict $\leftarrow \{\}$
for i ***in*** *searchOrder* **do**
  needNew $\leftarrow$ True
  for $j = 1$ ***to*** *len(bundles)* **do**
    cnt $\leftarrow$ ConflictCnt(bundles[j],F[i])
    if *cnt + bundlesConflict[i]* $\leq K$ **then**
      bundles[j].add(F[i]), needNew $\leftarrow$ False
      break
  if *needNew* **then**
    Add $F[i]$ as a new bundle to *bundles*
Output: *bundles*

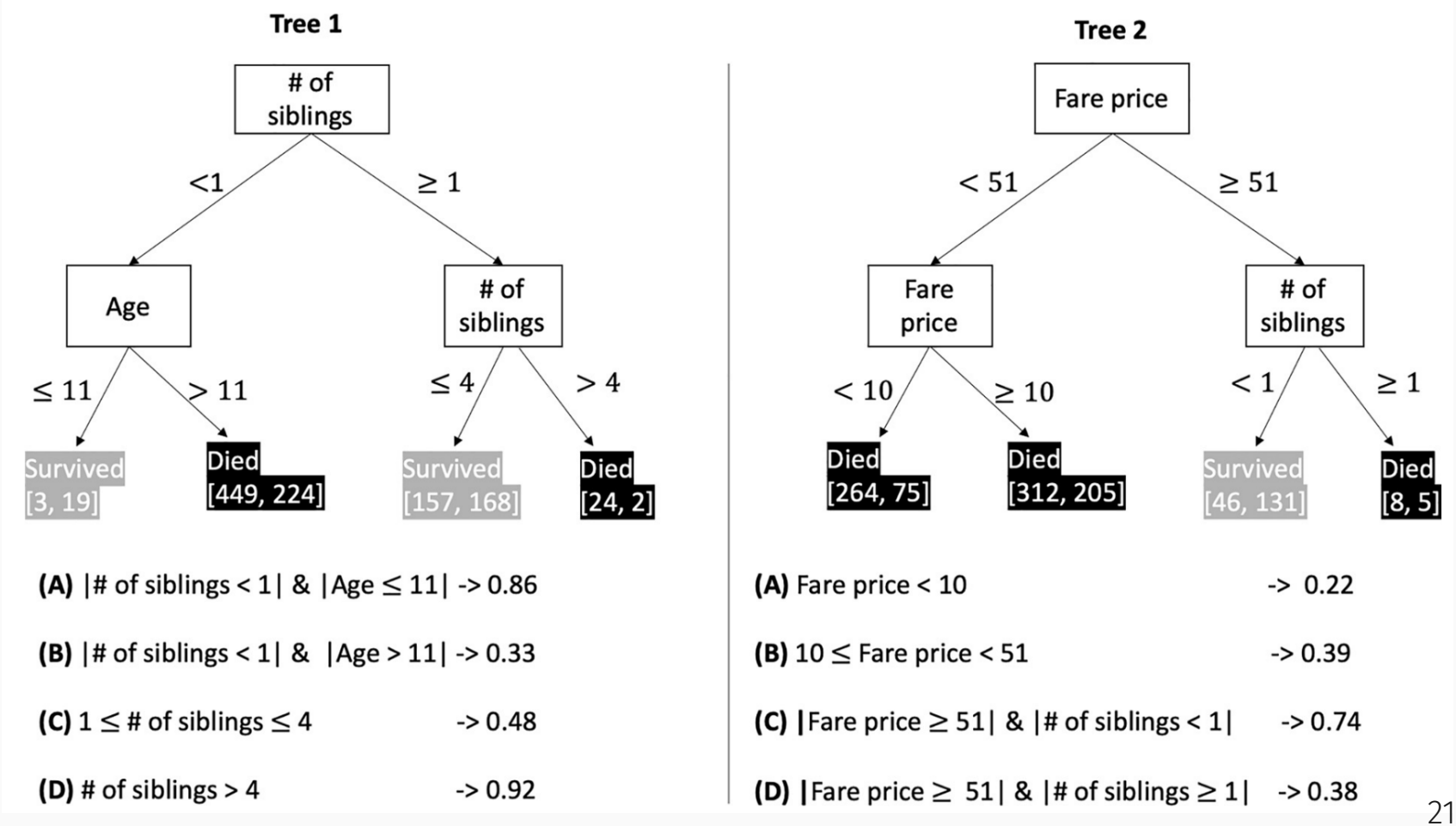
Algorithm 4: Merge Exclusive Features

Input: $numData$: number of data
Input: F: One bundle of exclusive features
binRanges $\leftarrow \{0\}$, totalBin $\leftarrow 0$
for f ***in*** F **do**
  totalBin += f.numBin
  binRanges.append(totalBin)
newBin $\leftarrow$ new Bin(numData)
for $i = 1$ ***to*** $numData$ **do**
  newBin[i] $\leftarrow 0$
  for $j = 1$ ***to*** *len*(F) **do**
    if $F[j].bin[i] \neq 0$ **then**
      newBin[i] $\leftarrow$ F[j].bin[i] + binRanges[j]
Output: *newBin*, *binRanges*

- И ещё алгоритмы для вычисления gain на основе гистограмм, но в них не будем углубляться

Обратно к деревьям

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



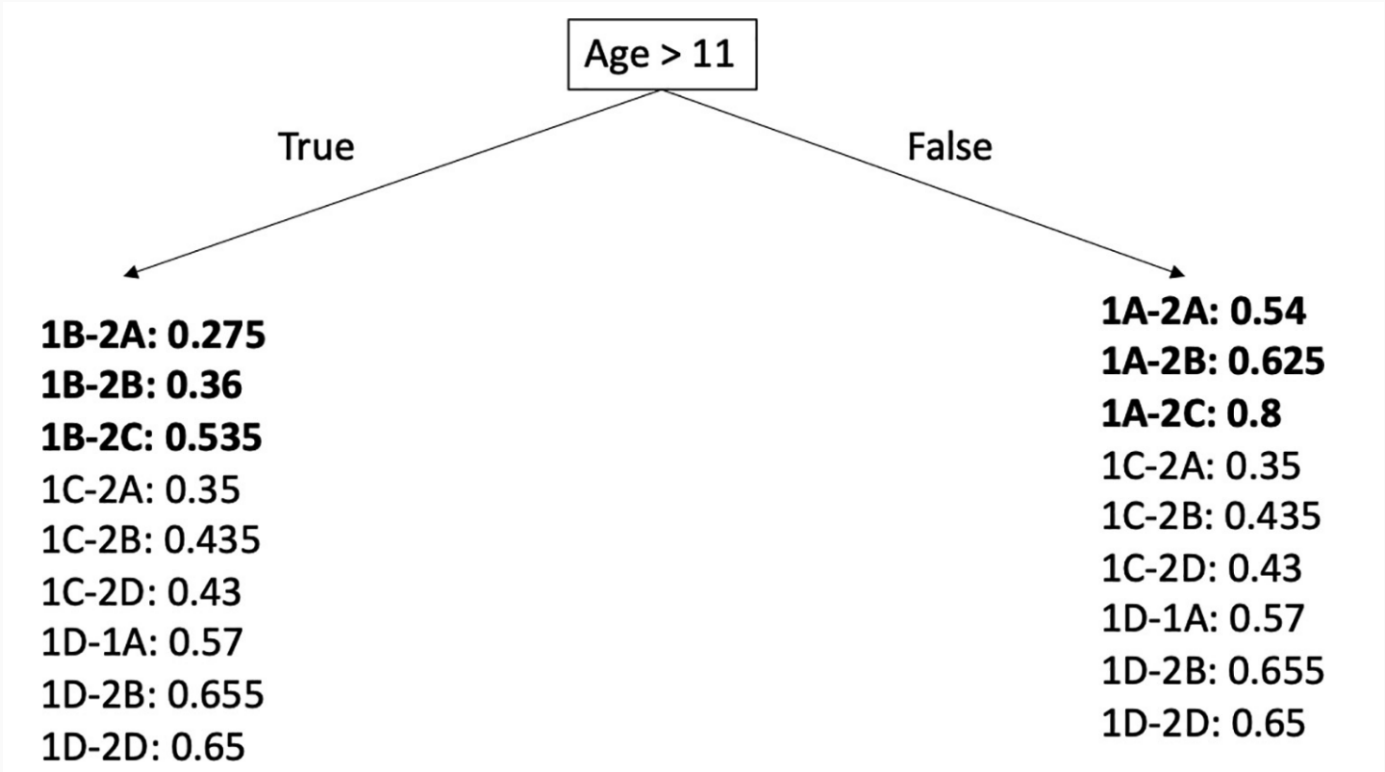
Обратно к деревьям

- 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

Обратно к деревьям

- 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 за внимание!