БУСТИНГ

Сергей Николенко

СПбГУ — Санкт-Петербург

02 ноября 2021 г.

Random facts:

- 2 ноября 1721 г. Россия стала империей: Пётр I принял титул Отца Отечества, Императора Всероссийского и Петра Великого
- 2 ноября 1948 г. Маргарет Чейз-Смит стала первой женщиной, избранной в Сенат США, а 2 ноября 1993 г. палата лордов одобрила решение о посвящении женщин в духовный сан англиканской церкви
- 2 ноября 1959 г. Жак Плант из «Montreal Canadiens» в очередной раз получил шайбой в лицо; когда ему наложили семь швов, Плант вернулся на площадку в маске, которую сам изготовил из стеклопластика и резины
- 2 ноября 1960 г. суд присяжных постановил, что роман Дэвида Герберта Лоуренса «Lady Chatterley's Lover» не является непристойным
- 2 ноября 1988 г. вирус, запущенный студентом Робертом Моррисом, вывел из строя часть сети ARPANET и был обнаружен в MIT; студент попал под суд, был оштрафован и 26 июля 1989 условно осуждён

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.

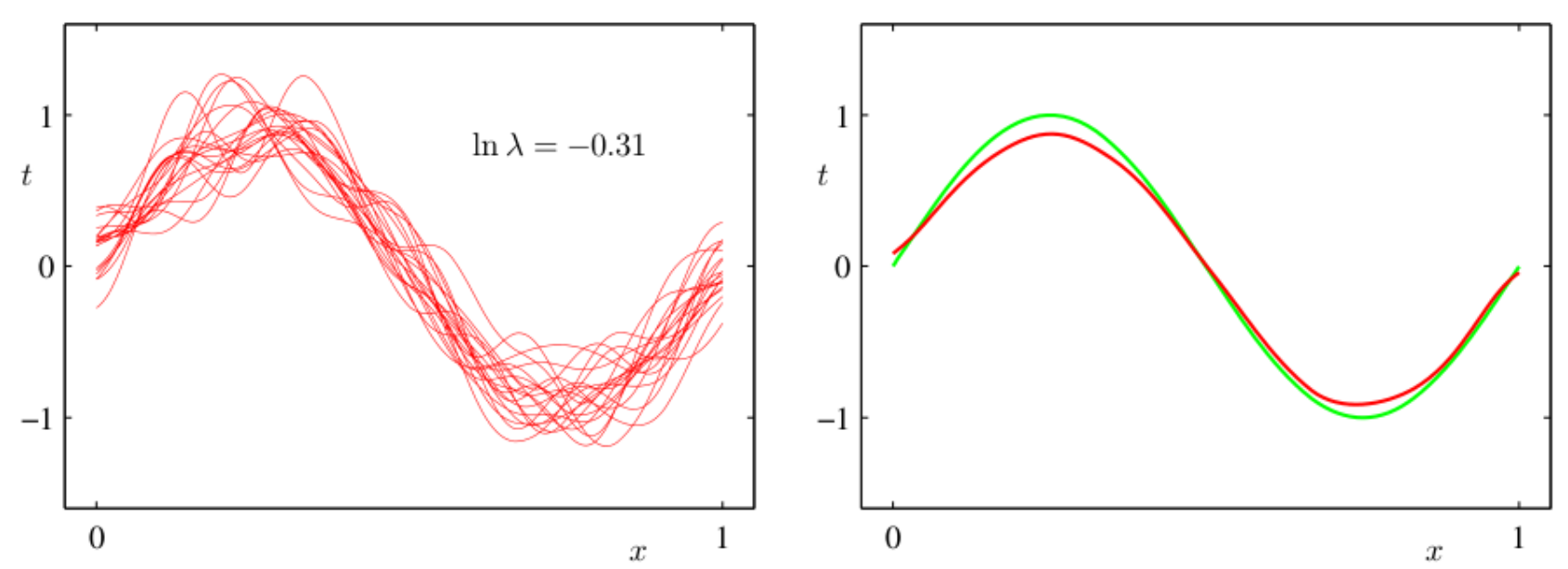
Комбинация моделей и байесовское усреднение

- А байесовское усреднение будет выглядеть как

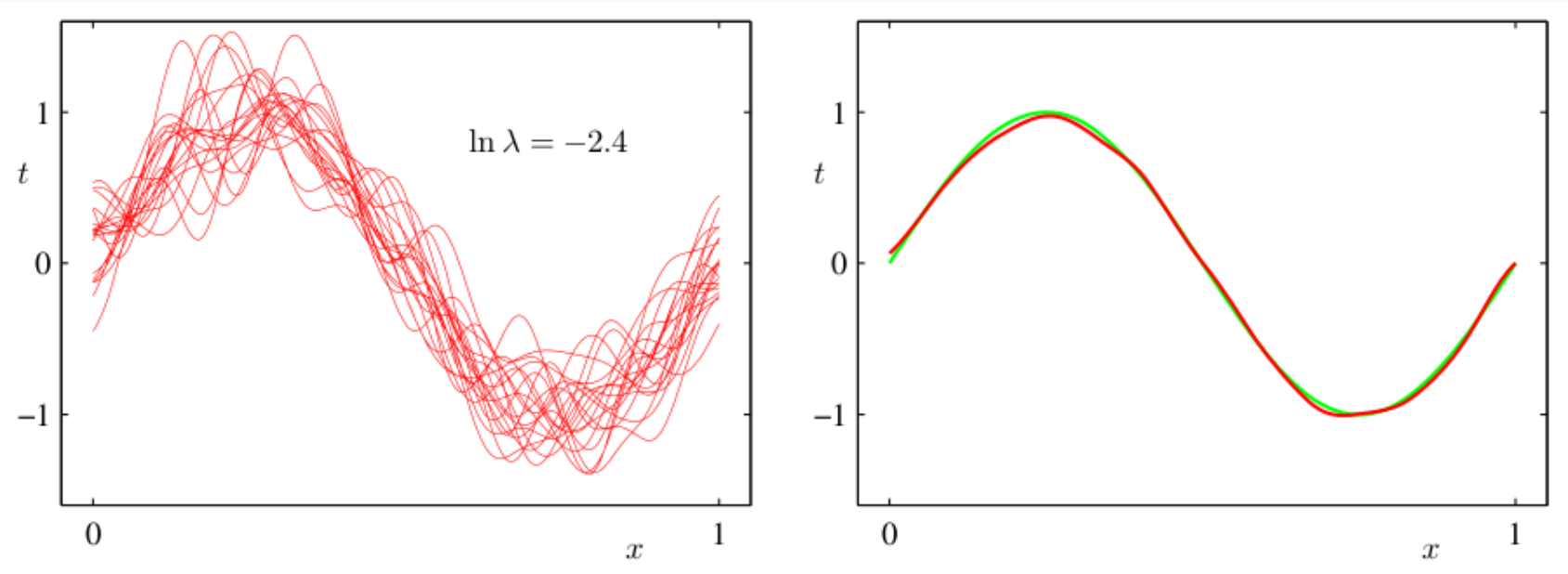
$$p(\mathbf{X}) = \sum_{h=1}^{H} p(\mathbf{X} \mid h) p(h).$$

- 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.

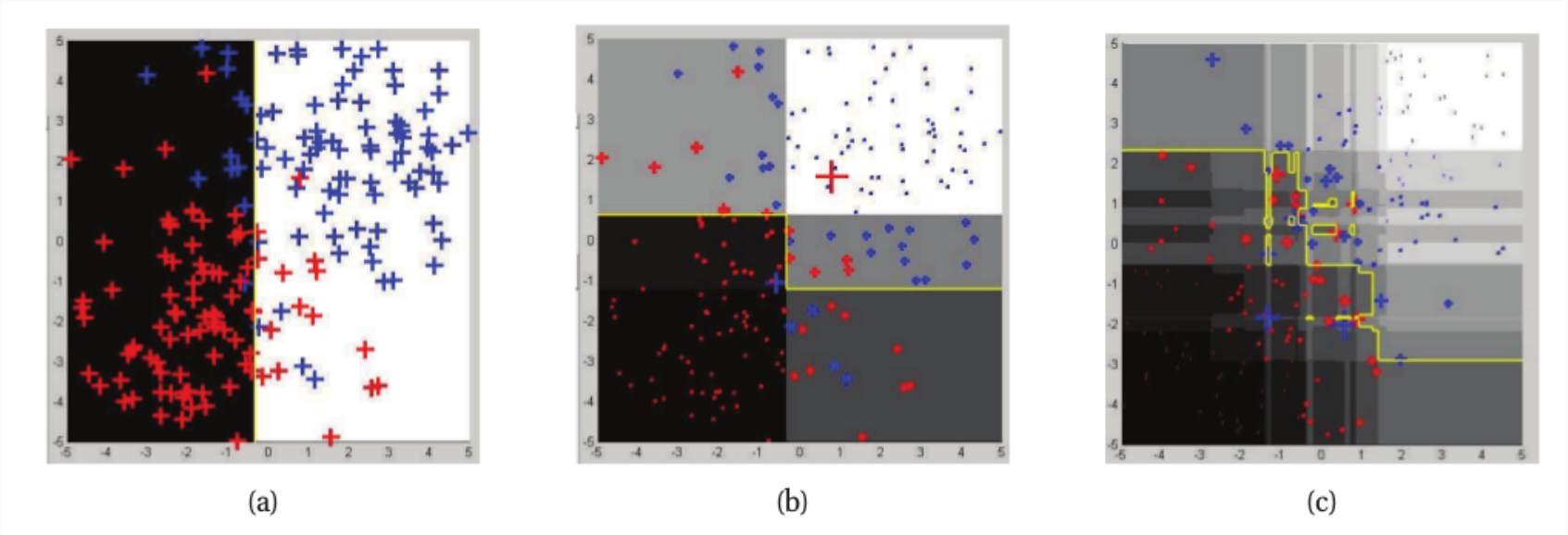
- Таким образом, бустинг можно рассматривать как оптимизацию экспоненциальной ошибки.

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:

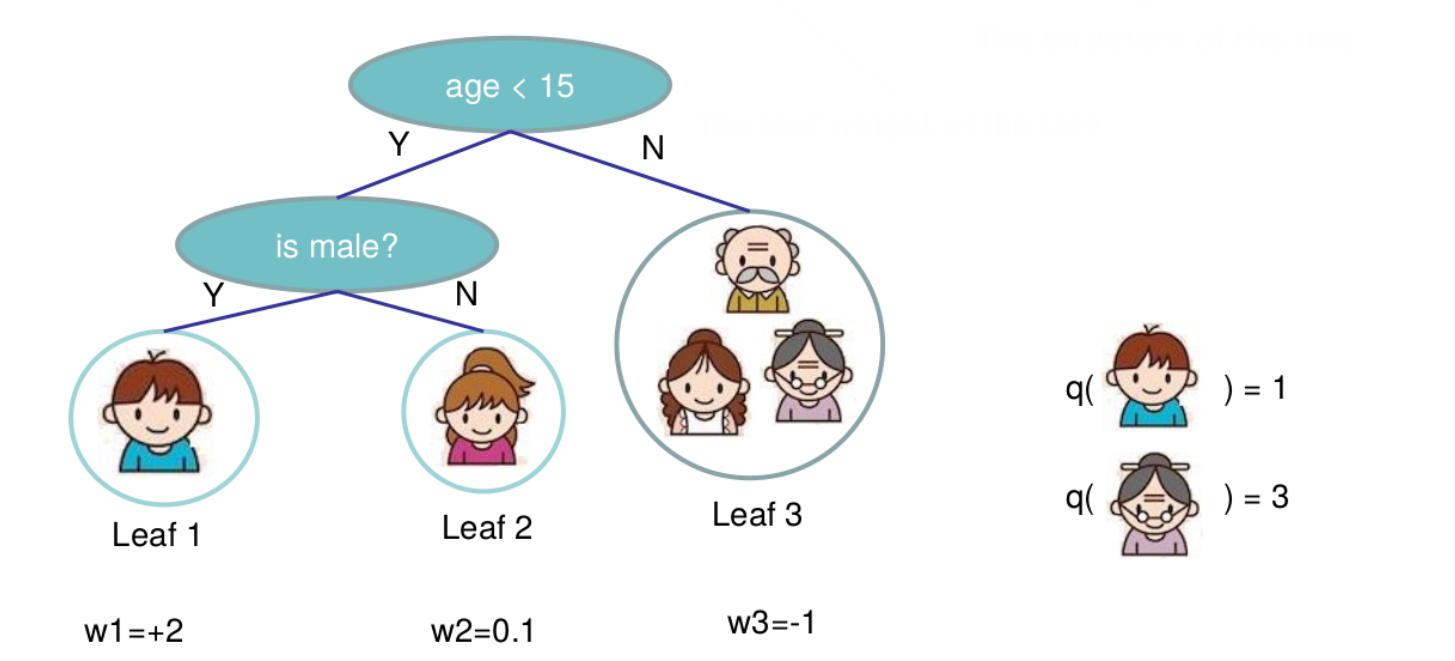
$$\text{Obj}^{(t)} \approx \sum_{i=1}^{N} \left(g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right) + \Omega(f_t).$$

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

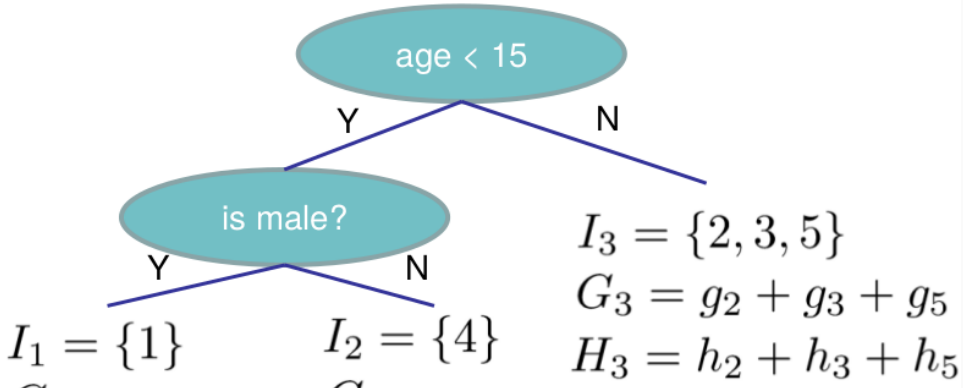
- Это сумма T независимых квадратичных функций, так что

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

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

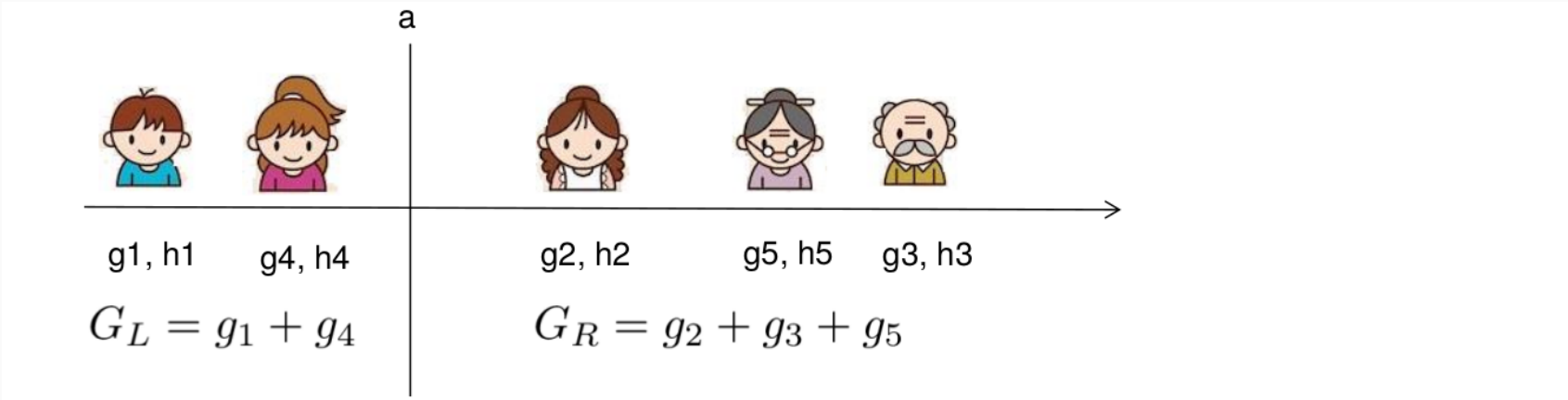
Градиентный бустинг

- 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.

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!