Бустинг решающих деревьев и эволюционные алгоритмы

Борис Янгель
hr0nix@acm.org

Московский государственный университет им. М. В. Ломоносова

22 октября 2009 г.

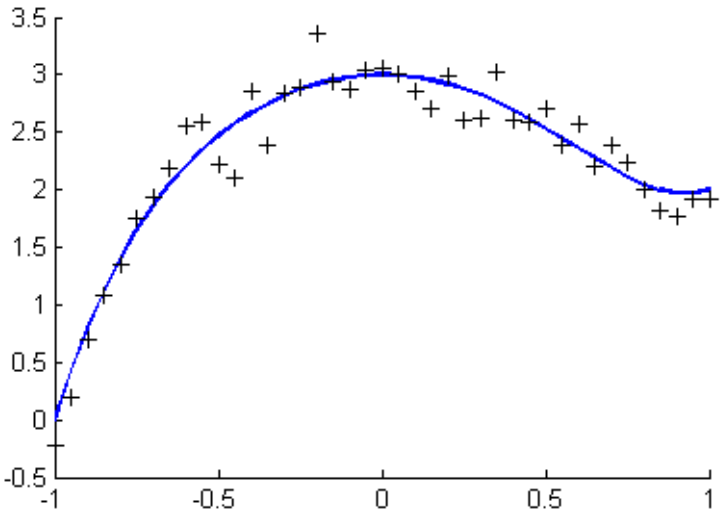
Обучение по прецедентам

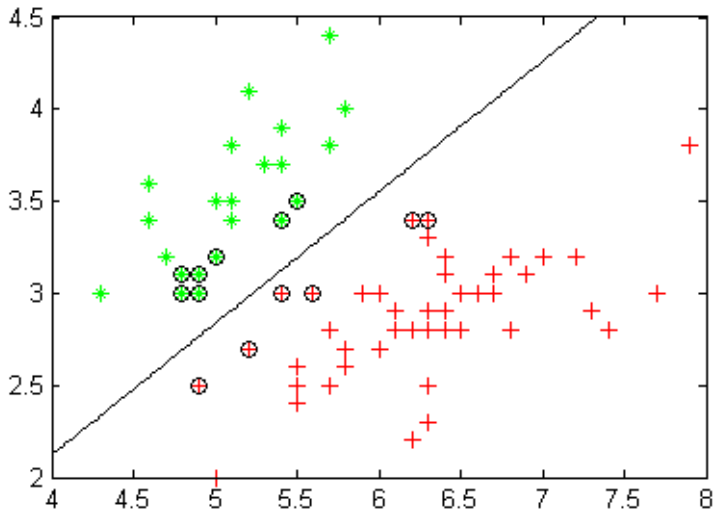
- Имеется неизвестная зависимость $\mathbf{y} \sim \mathbf{x}$.
- Имеются пары наблюдений $(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \ldots, (\mathbf{x}_n, \mathbf{y}_n)$ — *обучающая выборка*.
- По обучающей выборке необходимо некоторым образом восстановить неизвестную зависимость. Причем сделать это нужно так, чтобы восстановленные значения $\tilde{\mathbf{y}} = f^*(\mathbf{x})$ были близки к реальным $\mathbf{y}$ не только на обучающей выборке (отсутствие *переобучения*).
- Обычно f^* выбирается из множества некоторых потенциальных моделей F путем минимизации *функционала эмпирического риска* на обучающей выборке:

$$f^* = \arg\min_{f \in F} \sum_{i=1}^{n} L\left(\mathbf{y}_i, f(\mathbf{x}_i)\right).$$

Обучение по прецедентам

- Обычно $\mathbf{x}_i \in \mathbb{R}^m$ и называется вектором признаков.
- Стандартные задачи:
  - $\mathbf{y}_i \in \mathbb{R}$ — восстановление регрессии.
  - $\mathbf{y}_i \in \{1, 2, \ldots, K\}$ — классификация на K классов.
- Примеры:





Бустинг

- Будем восстанавливать зависимость как

$$f(\mathbf{x}) = \sum_{t=1}^{T} h_t(\mathbf{x});$$

- h_t — *слабые* предикторы из семейства H;
- Как выбрать h_t из H? Сложная оптимизационная задача!

Бустинг

- Будем восстанавливать зависимость как

$$f(\mathbf{x}) = \sum_{t=1}^{T} h_t(\mathbf{x});$$

- h_t — *слабые* предикторы из семейства H;
- Как выбрать h_t из H? Сложная оптимизационная задача!
- Которую можно решать жадно-итерационным методом:
  1. Положить $f_0(\mathbf{x}) = 0$;
  2. Для $t = 1, \ldots, T$:

$$h_t = \arg\min_{h \in H} \sum_{i=1}^{n} L\left(\mathbf{y}_i, f_{t-1}(\mathbf{x}_i) + h(\mathbf{x}_i)\right);$$

$$f_t(\mathbf{x}) = f_{t-1}(\mathbf{x}) + h_t(\mathbf{x});$$

  3. Вернуть $f(\mathbf{x}) = f_T(\mathbf{x})$.
- Сложная оптимизационная задача разбилась на T задач попроще.

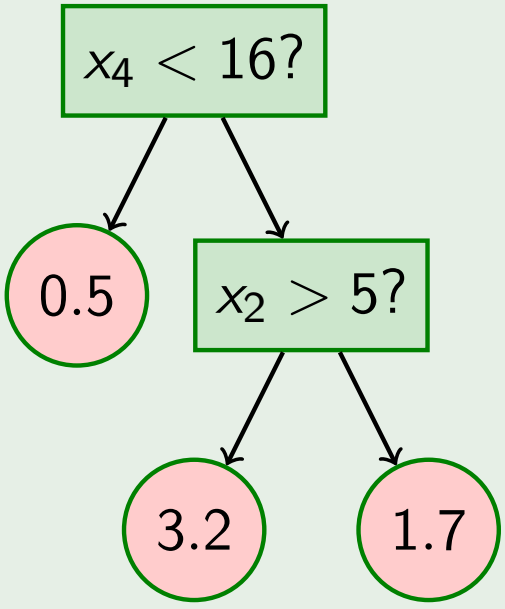
Почему бустинг?

- Как показывают теоретические и эмпирические результаты, для получения хорошей функции f достаточно иметь набор *весьма* слабых предикторов;
- Для ряда функционалов риска L алгоритму не свойственно переобучение;
- Существуют устойчивые к шумам в обучающей выборке разновидности бустинга;
- Много примеров успешного использования в прикладных задачах.

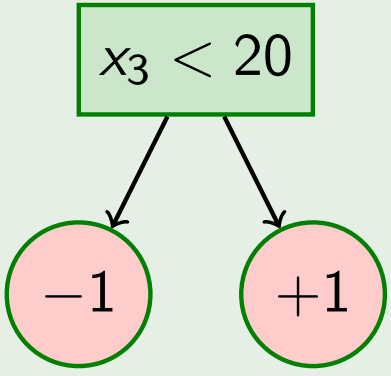
Слабые предикторы

В качестве слабых предикторов чаще всего используются решающие деревья небольшой высоты, иногда даже высоты 1 (пни):

Решающее дерево



Пень (*stump*)



Решающие деревья

Решающее дерево может быть представлено аналитически:

$$y(\mathbf{x}) = \sum_k \gamma_k I[\mathbf{x} \in R_k].$$

Тут R_k соответствуют листьям дерева, причем

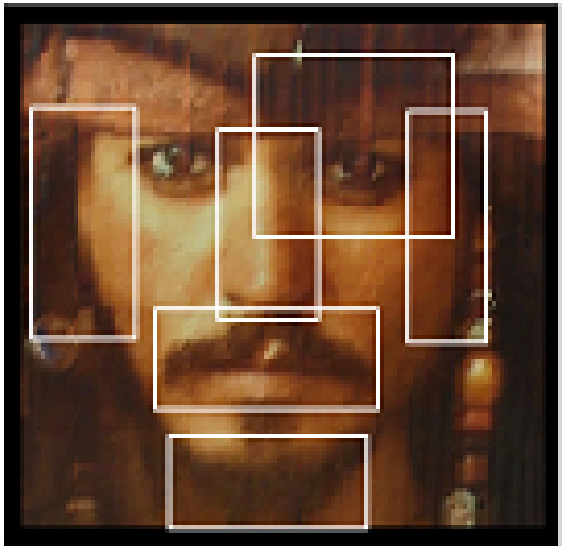
$$\bigcup_k R_k = \mathbb{R}^m,\ R_i \bigcap R_j = \emptyset \text{ при } i \neq j.$$

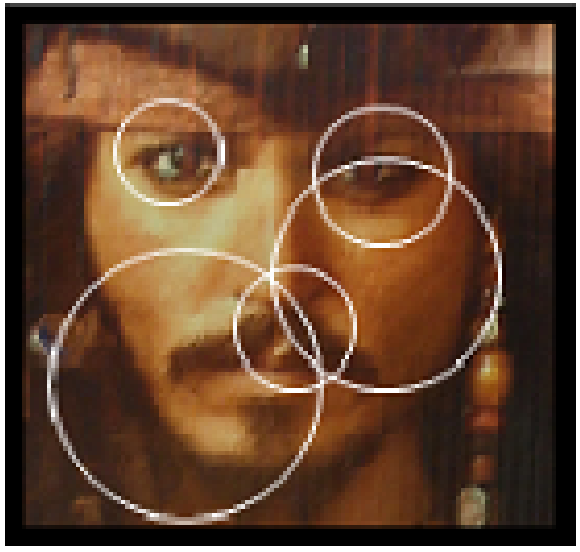
Для построения дерева по обучающей выборке нужно выбрать признак и порог для каждого внутреннего узла, а также выходные значения для всех терминальных. Обычно это делается рекурсивным жадным алгоритмом, спускающимся по дереву вниз от корня. При этом в каждом узле перебираются все признаки, и для каждого находится наилучшее значение знака и порога.

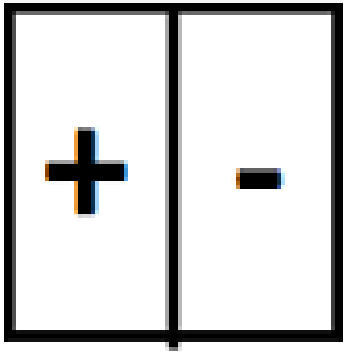
Бустинг и распознавание изображений 1

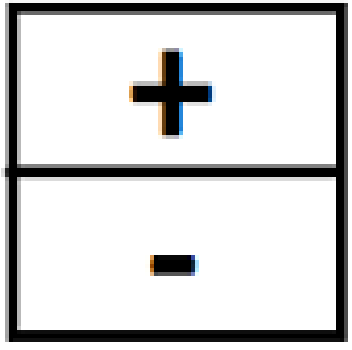
- Бустинг нередко используется в задачах распознавания изображений;
- Один из классических примеров использования — в алгоритме Джонса и Виолы для нахождения лиц на изображении;
- Признаки x_i, $i = 1 \dots m$ могут строиться как значения некоторой числовой характеристики на различных подобластях изображения;
- Число таких признаков может быть очень большим:
  - Число возможных прямоугольных субрегионов изображения размеров $N \times M$ составляет $\Theta(N^2 M^2)$, т.е. порядка 100000 признаков для изображений размерами 24×24 пиксела.
  - Число возможных окрестностей точек изображения с радиусом не более K — $\Theta(NMK)$.

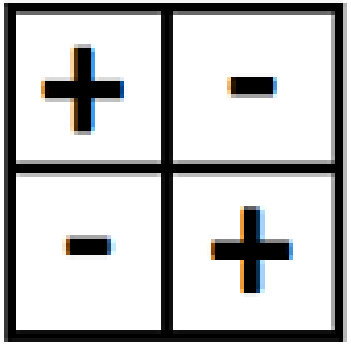
Примеры признаков

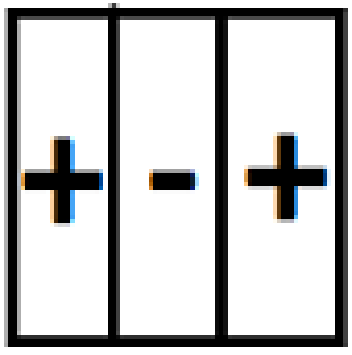












Бустинг и распознавание изображений 2

- При использовании большинства широко известных функционалов риска для выбора знака и порога существуют эффективные алгоритмы;
- Оптимизационная задача выбора разбивающего признака x_j — значительно сложнее:
  - Оптимизация дискретной мультимодальной функции;
  - Зачастую решается полным перебором по всему множеству признаков;
  - А процесс нужно повторять на каждой итерации бустинга для каждого внутреннего узла дерева;
  - Результат — очень большое время обучения классификатора (до нескольких недель в классических работах).
- В таких задачах хорошо себя зарекомендовали *генетические и эволюционные алгоритмы*!

Эволюционная оптимизация решающих деревьев 1

На каждой итерации бустинга можно сформулировать задачу эволюционной оптимизации:

- Решение (хромосома) — решающее дерево D;
- Функция приспособленности на итерации бустинга t есть

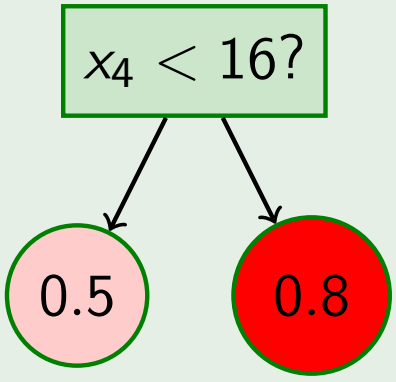
$$F_t(D) = -\sum_{i=1}^{n} L\left(y_i, f_{t-1}(\mathbf{x}_i) + D(\mathbf{x}_i)\right).$$

- Генетические операторы можно позаимствовать из генетического программирования:
  - Кроссовер меняет местами поддеревья двух деревьев;
  - Мутация изменяет структуру дерева или разбивающий признак в одном из узлов.
- Разбивающие пороги и значения, приписанные терминальным узлам, продолжим выбирать жадно, спускаясь вниз от корня.

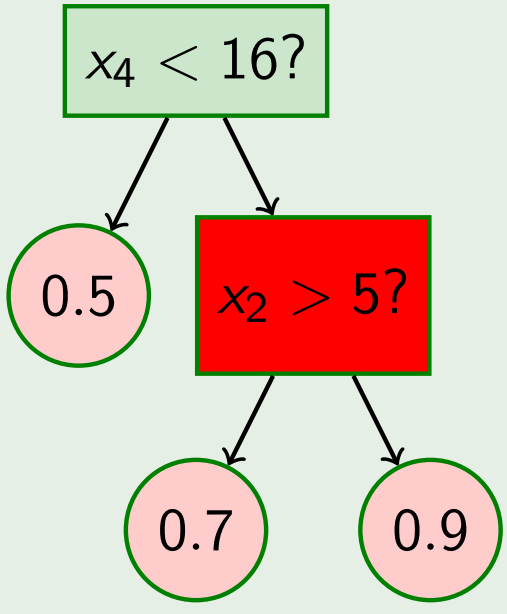
Мутация — разбиение узла

Превращаем случайный терминальный узел во внутренний, выбираем для разбиения произвольный признак.

Было



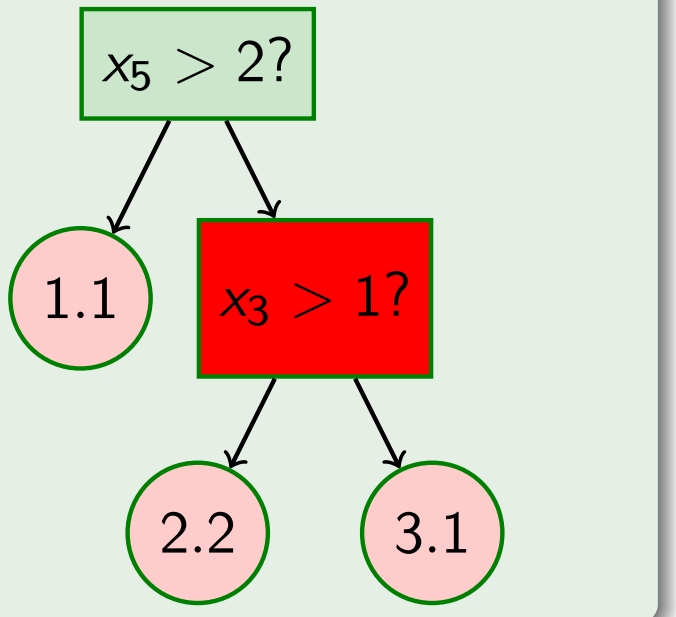
Стало



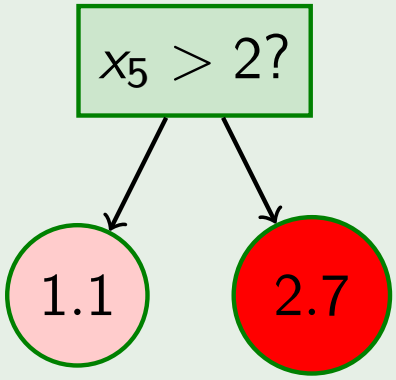
Мутация — склейка поддерева

Выбираем в дереве случайный внутренний узел и превращаем его в терминальный, сворачивая соответствующее ему поддерево.

Было



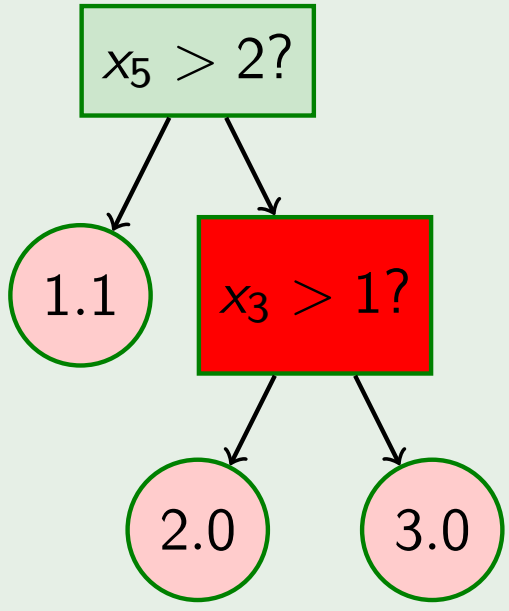
Стало



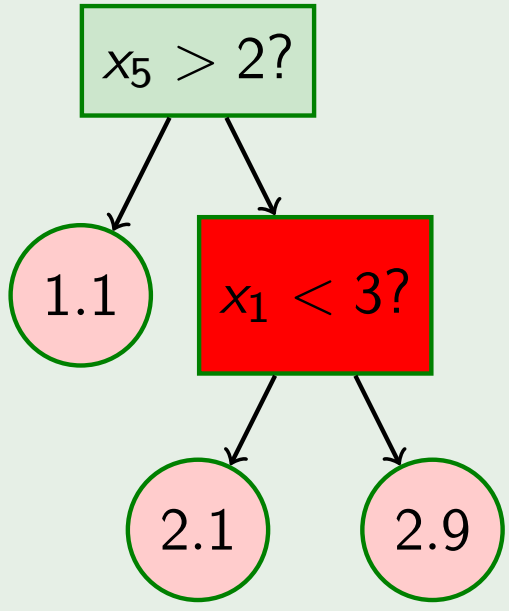
Мутация — изменение признака

Заменяем в случайном внутреннем узле признак на случайный (например, методом однобитовой мутации или прибавления гаусовского шума).

Было



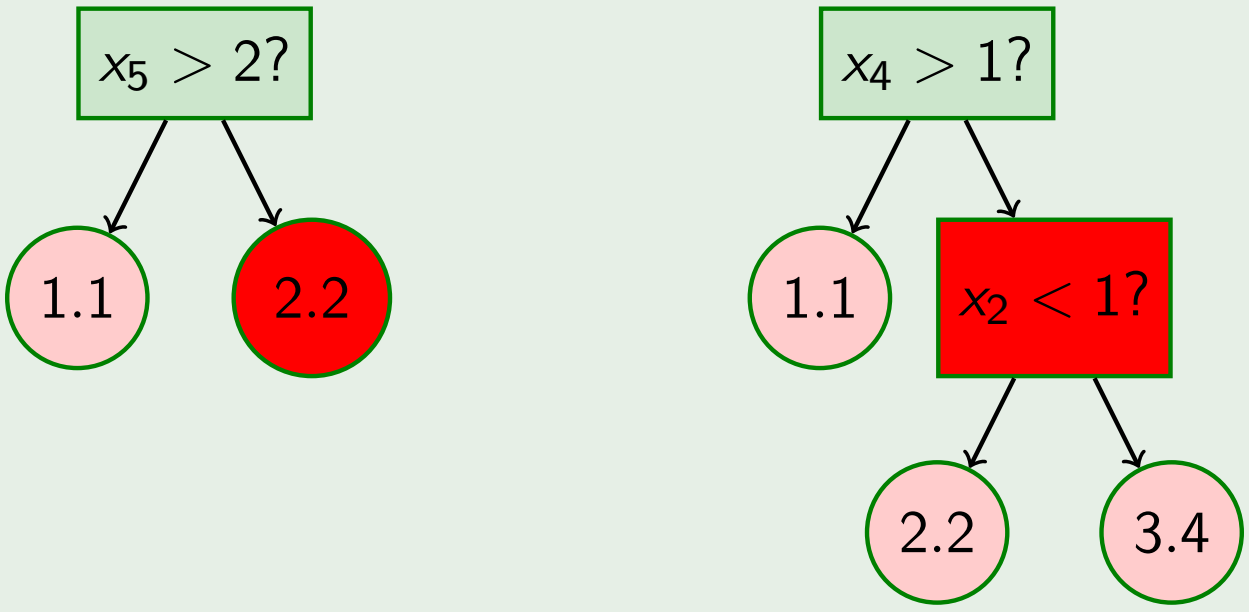
Стало



Реализация кроссовера

Выбираем по случайному, отличному от корня узлу в каждом из деревьев, затем меняем узлы местами вместе с поддеревьями.

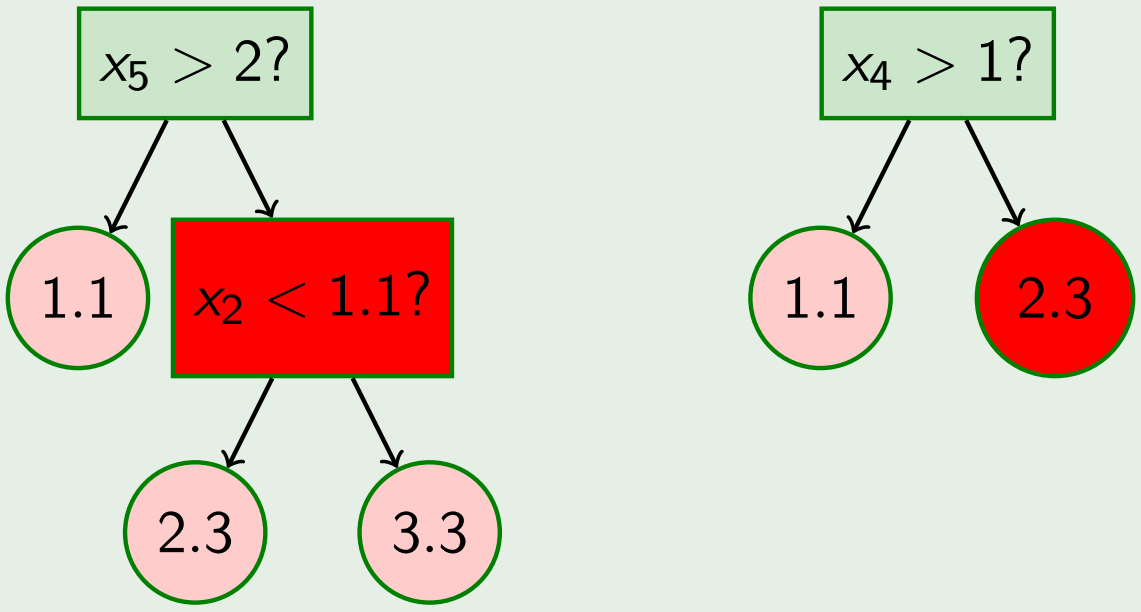
Было



Реализация кроссовера

Выбираем по случайному, отличному от корня узлу в каждом из деревьев, затем меняем узлы местами вместе с поддеревьями.

Стало



Почему это *хорошо*?

- Почему это работает быстрее?
  - Мы не перебираем все возможные признаки;
  - Можно ограничивать пересчет параметров узлов изменившимся поддеревом.

Почему это *хорошо*?

- Почему это работает быстрее?
  - Мы не перебираем все возможные признаки;
  - Можно ограничивать пересчет параметров узлов изменившимся поддеревом.
- Какие еще преимущества?
  - Мы рассматриваем несколько потенциальных структур дерева;
  - Даже если признаки не коррелируют, процесс не превратится в случайный поиск при использовании решающих деревьев с большим числом узлов.

Почему это *хорошо*?

- Почему это работает быстрее?
  - Мы не перебираем все возможные признаки;
  - Можно ограничивать пересчет параметров узлов изменившимся поддеревом.
- Какие еще преимущества?
  - Мы рассматриваем несколько потенциальных структур дерева;
  - Даже если признаки не коррелируют, процесс не превратится в случайный поиск при использовании решающих деревьев с большим числом узлов.
- Какие недостатки?
  - Работает медленнее (или хуже) полного перебора, когда число признаков невелико;
  - Не подходит для бустинга пней на некоррелирующих признаках.

Что уже сделано?

- Реализация бустинга пней эволюционными методами.
  - Сравнима по скорости и качеству с *state-of-the-art* методами тренировки классификаторов изображений методом бустинга пней.
  - Представлена на конференциях ММРО-14 и GraphiCon'2009.

Что уже сделано?

- Реализация бустинга пней эволюционными методами.
  - Сравнима по скорости и качеству с *state-of-the-art* методами тренировки классификаторов изображений методом бустинга пней.
  - Представлена на конференциях ММРО-14 и GraphiCon'2009.
- Библиотека для бустинга деревьев предложенным методом.
  - В данный момент идет интенсивная оптимизация кода.
  - Текущие эксперименты показывают преимущество метода в качестве классификации перед предыдущей работой без существенного падения скорости обучения.
  - Для запуска оптимизации нужно только выбрать функции мутации и вычисления признака, а также функционал эмпирического риска.

Еще вопросы?