САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Математико-механический факультет

Кафедра Системного Программирования

Белоус Михаил Андреевич

Полиномиальная аппроксимация листов деревьев решения

Курсовая работа

Научный руководитель:
к. ф.-м. н. Куралёнок И. Е.

Санкт-Петербург
2013

Оглавление

Введение

Машинное обучение — это набор методов выявления закономерностей в эмпирических данных. "Обучение с учителем" — это класс методов, в котором имеется набор данных и для них есть ответ. И такой набор мы будем называть learn. На наборе learn программа должна научиться принимать решение. После того как программа прошла обучение, происходит проверка её работы на закрытом наборе данных, который ниже будем называть validate. Подобный метод оценки моделей применяется во многих системах, связанных с автоматической обработкой данных, где невозможно вывести явный критерий, или этот критерий слишком быстро меняется. Областью применения подобных алгоритмов на практике являются: поисковая выдача, распознавание спама, биоинформатика, распознавание образов.

Данные в машинном обучении с учителем обычно представляются в следующем виде: каждый пример — точка в n-мерном пространстве и для него есть экспертная оценка.

Переобучение — явление в машинном обучении, когда система улучшает свою точность на learn, но не улучшает свою точность на генеральной совокупности.

1. Обзор средств и подходов, используемых в работе

Оценки качества работы метода:

1. Невязка — сумма квадратов отклонений нашего решения от истинного значения

2. Работа в композиции с другими алгоритмами см. boosting.

3. Приемлемое использование времени и памяти.

4. Возможность использования на разных данных без изменения коэффициентов.

Регуляризация — это штраф за сложность модели, так как сложные модели больше описывают входные данные, чем генеральную совокупность. Данные, которые я использовал в работе, являются реальными web-данными. Learn состоит из 12465 образцов, validate состоит из 46596 образцов. Каждый образец состоит из 50 характеристик и экспертной оценки.

2. Деревья решений

Деревья решений — это структуры, у которых в каждой вершине есть условие, по которому будет выбираться следующий шаг. В листовых вершинах находится ответ. Поиск дерева, которое было бы оптимально, является NP полной задачей. Поэтому на практике применяют жадный метод постороения — на каждом шаге выбирают разбиение с минимальной суммарной дисперсией оценок[1].

Несмотря на свою простоту, деревья решений являются мощным методом машинного обучения. Проблема деревьев решений в том, что они кусочно-постоянные. Это уменьшает их точность, мешает применять их на реальных данных и в ансамбле с другими методами машинного обучения. В своей работе я хочу сделать деревья более непрерывными, но при этом сохранить, а возможно и улучшить их точность.

Идея моей работы заключается в том, чтобы взять характеристики, по которым мы строим деревья, и в листьях деревьев построить многочлен, тогда значение в точке мы будем вычислять по формуле

$$f(C, x) := \sum_{i,j} C_{i,j} \cdot x_i \cdot x_j \tag{1}$$

где C, это коэффициенты соответствующего листа дерева. Тогда невязка вычисляется как

$$miss(C) := \sum_{i \in Dataset} (f(C, x_i) - ans_i)^2 \tag{2}$$

Регуляризация для этой модели

$$reg(C) := \sum C_i^2 \tag{3}$$

Введем требование равенства многочленов соседних листов на границе областей, то есть строгое равенство многочленов в гиперплоскости. $mask$, $nmask$ - граничащие листья; B - условие, по которому мы разделяем области. 0 — номер мнимой характеристики , всюду равной 1, f — номер характеристики, по которому мы разделяем условием. Так как характеристика f разбивает пространство плоскостью с координатой f, равной B, то x_f в этой плоскости всегда равен B.

$$C_{mask,0,i} + B \cdot C_{mask,f,i} = C_{nmask,0,i} + B \cdot C_{nmask,f,i} \quad \forall\, i \notin \{f, 0\} \tag{4}$$

$$C_{mask,i,j} = C_{nmask,i,j} \quad \forall\, i, j \notin \{0, f\} \tag{5}$$

$$C_{mask,0,0} + B^2 \cdot C_{mask,f,f} = C_{nmask,0,0} + B^2 \cdot C_{mask,f,f} \tag{6}$$

(4) — условие равенства линейных членов в данной плоскости, (5) — условие квадратичных членов и (6) — условие на константу. Такие условия нужно ввести для всех характеристик и всех граничащих листов. В дальнейшем будем именовать это

множество условий $margins$.

3. Мягкие условия на границе

Рассмотрим следующий способ исполнения этих условий: введем штраф за несоблюдение условий в виде $e^{\lambda \cdot margin^2}$, где $margin$ – это отклонение от равенства. Рассмотрим задачу минимизации невязки и регуляризации с этим штрафом. Тогда выразим C как

$$\operatorname{argmin} miss(C) + reg(C) + \sum_{i \in margins} e^{\lambda \cdot margin_i(C)^2} \tag{7}$$

Поскольку экспонента и квадрат являются выпуклыми функциями, у целевой функции имеется единственная точка локального минимума, в связи с этим мы можем использовать градиентный спуск[3]. Однако для того чтобы градиентный спуск сходился, наша функция должна удовлетворять условию Липшица (расти не быстрее некоторой степени), но экспонента растет быстрее любой степени. Варианты решения этой проблемы:

1. Сделать маленький шаг для градиентного спуска. Этот метод работает, но достаточно медленно.

2. В предположении, что решение не имеет сильных отклонений от условий непрерывности, разложим экспоненту около нуля в ряд до второго члена, тогда градиентные условия станут линейными. Останется только решить систему линейных уравнений. Эту идею мы сразу отбросили, так как многочлен плохо заменяет экспоненту далеко от нуля.

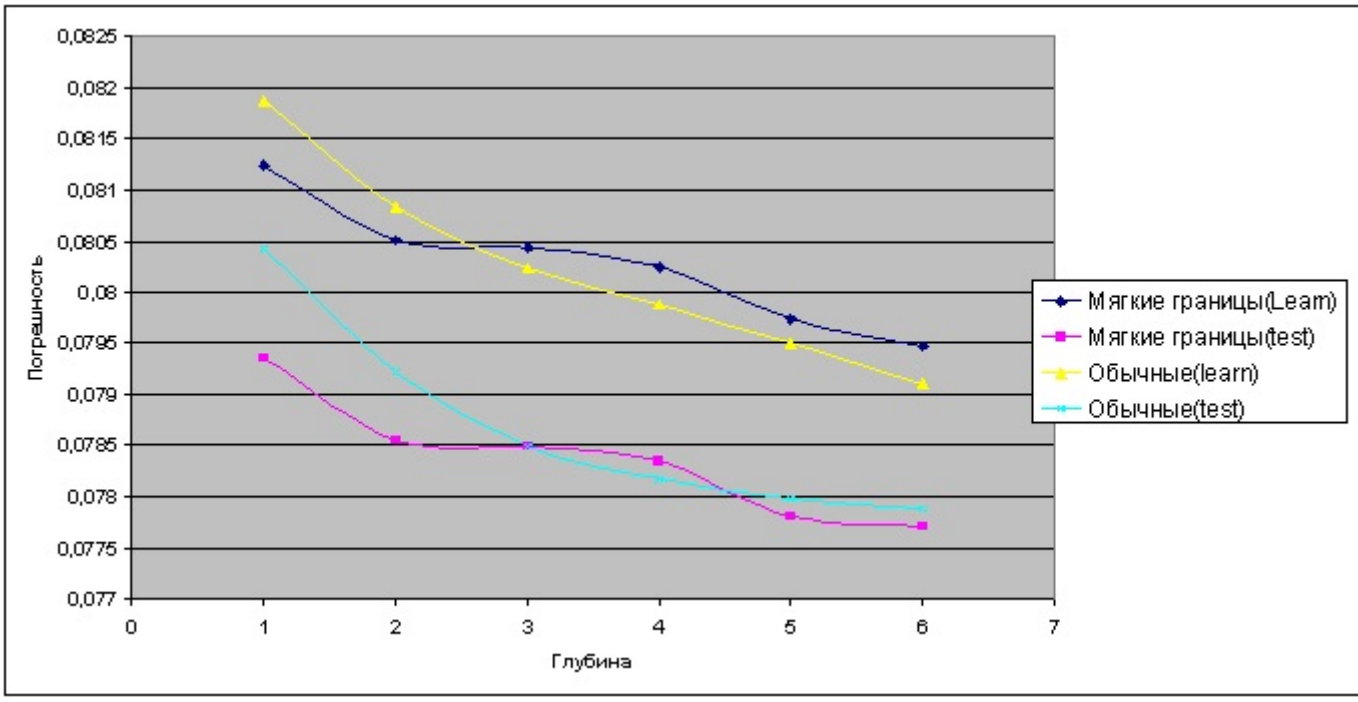

Рис. 1: Работа стандартных деревьев и деревьев с мягкими границами

4. Ослабленные условия на границе

Введем ослабленные условия на границы: $\forall i \quad -\epsilon < margin_i(C) < \epsilon$. Для решения задачи минимизации невязки и регуляризации при условии границ, используем метод логарифмических барьеров[3], тогда мы найдем C как точку минимума следующей функции

$$-\sum_{i \in margins} (\log(\epsilon + margin_i) + \log(\epsilon - margin_i)) + miss(C) + reg(C) \tag{8}$$

Теперь мы перешли к безусловной оптимизации и можем использовать стандартные методы поиска минимума. Так как логарифм не обладает свойством Липшица, мы будем использовать малый шаг для сохранения сходимости градиентного спуска.

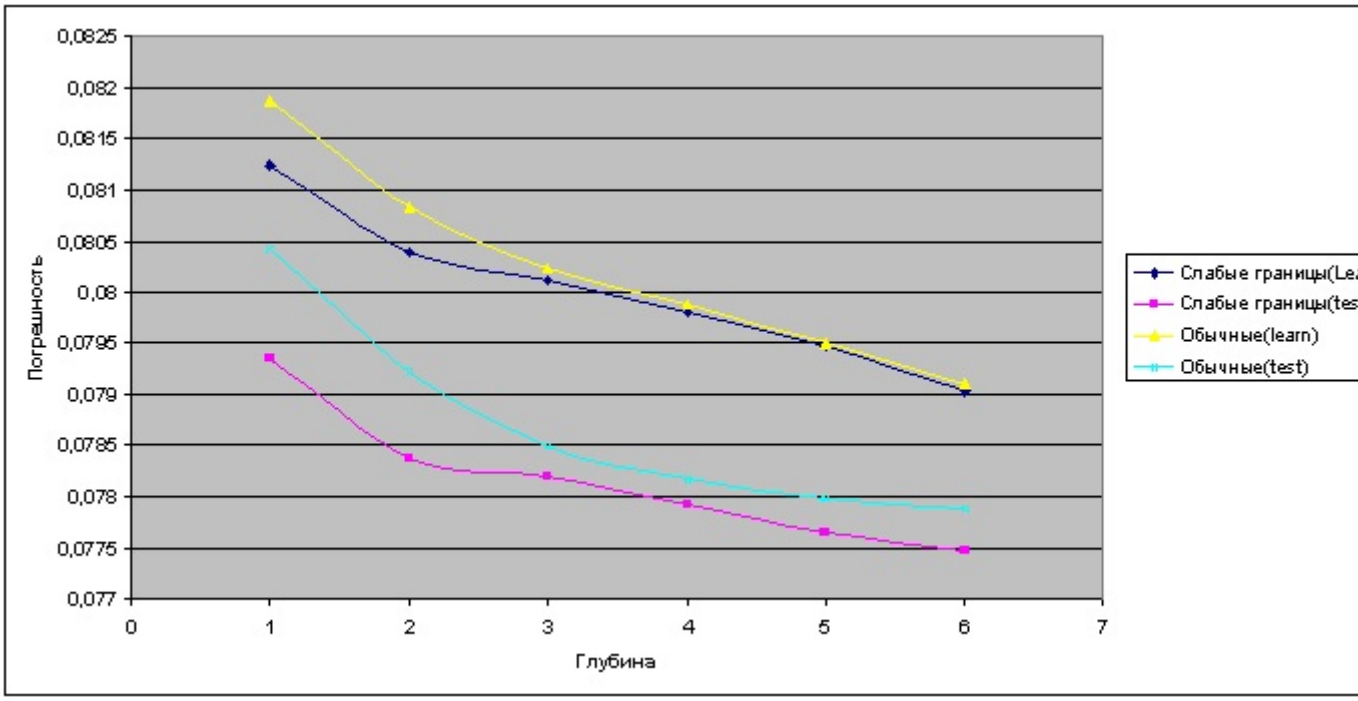

Рис. 2: Работа стандартных деревьев и деревьев со слабыми границами

5. Сравнение методов

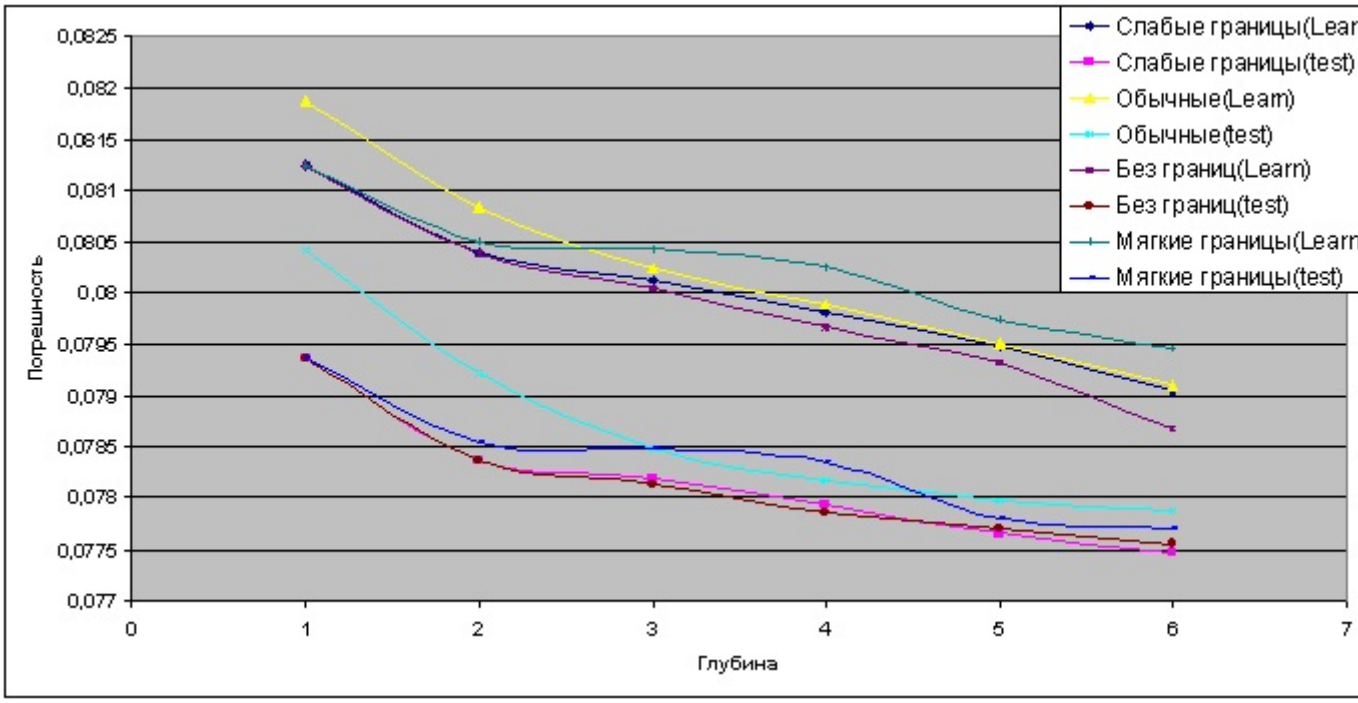

Рис. 3: Все деревья

По графику мы можем сделать вывод, что, несмотря на дополнительные ограничения, модели, учитывающие непрерывность целевой функции, лучше работают на тестовых данных, а, следовательно, меньше подвержены переобучению и лучше отражают генеральную совокупность. Из этого можно сделать вывод о том, что использование непрерывных на границах методов является перспективной областью для изучения.

6. Boosting

Бустинг (англ. boosting — улучшение) — это процедура последовательного построения композиции алгоритмов машинного обучения, когда каждый следующий алгоритм стремится компенсировать недостатки композиции всех предыдущих алгоритмов. Бустинг представляет собой жадный алгоритм построения композиции алгоритмов[2]. Так как tree boosting дает один из наиболее точных результатов среди известных методов регрессии, мне было необходимо настроить мой метод для того, чтобы мои деревья были применимы с boostingom. Проблемы, возникшие с алгоритмами мягких и ослабленных границ в том, что они работают медленно, а вычислять их надо очень много раз. Для того чтобы они работали приемлемое время, мне пришлось предподсчитать все коэффициенты, необходимые для вычисление градиента.

Минимальное значение погрешности на тесте бустингов разных деревьев: обычных деревьев: 0,074927464,деревьев со слабыми границами: 0,074833299, деревьев с мягких границами: 0,074878645, деревьев без границ: 0,074847414. Посмотрим на поведение ответа на тест в зависимости от шага бустинга.

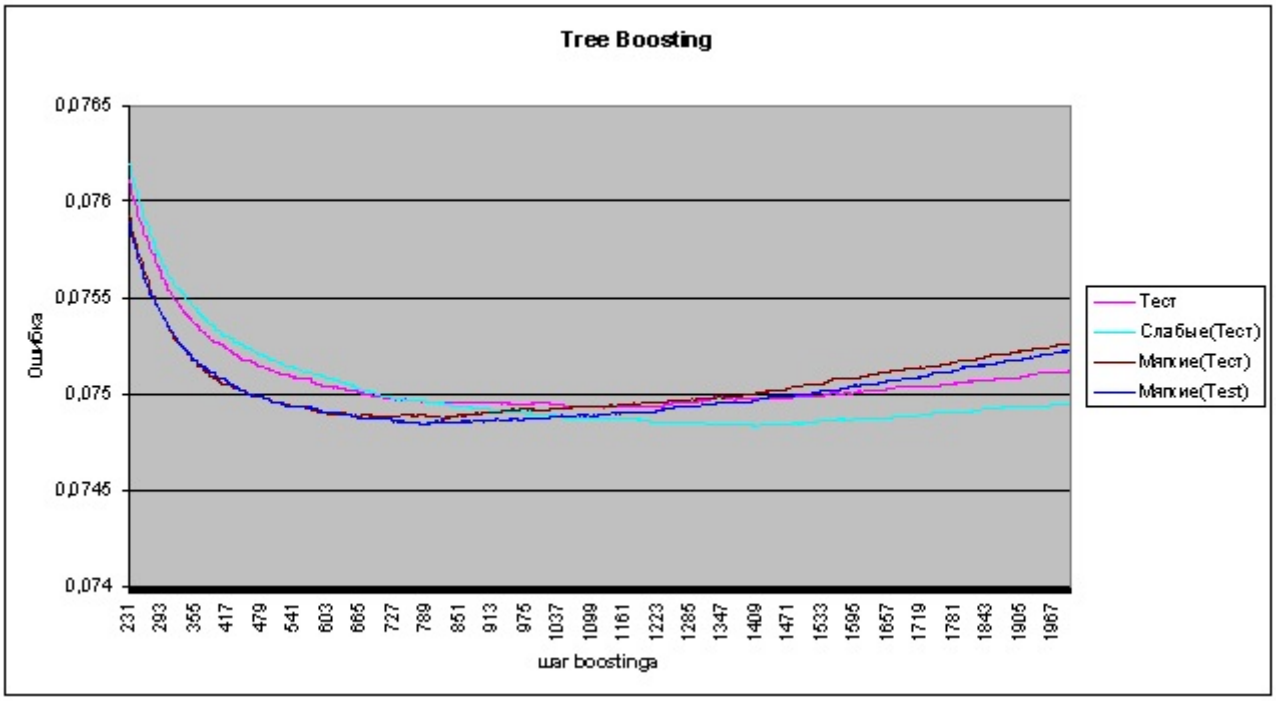

Рис. 4: Погрешность ансамбля

Как видно из графика, первым переобучается метод, не учитывающий границы, так как он минимизирует лишь штраф.

Заключение

В ходе работы был реализован новый метод машинного обучения, была достигнута превосходящая исходный алгоритм точность работы, высокая скорость работы. Реализация метода не имеет утечек памяти. Разработанные мною методы превосходят многие методы машинного обучения.

Так как мною было достигнуто достойное время работы алгоритмов, то я считаю, что имеет смысл продолжать данную работу. Я вижу интересным идею не использовать жадно построенное дерево, а попробовать изменить геометрию деревьев, а так же использовать деревья в ансамбле с другими методами.

Список литературы

[1] Shalizi Cosma. Classification and Regression Trees. — 2009. — http://www.stat.cmu.edu/ cshalizi/350/lectures/22/lecture-22.pdf.

[2] Константинов Дмитрий. Бустинг. — 2010. — http://goo.gl/23Ib6j.

[3] Нестеров Юрий Евгеньевич. Методы выпуклой оптимизации. — 2008.