Лекция 2
Алгоритмические композиции
Кульминация

Владимир Гулин
https://goo.gl/Yuvr8Z

13 февраля 2016 г.

План лекции

Задача обучения с учителем

Постановка задачи

Пусть дан набор объектов $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}, \ \mathbf{x}_i \in \mathcal{X}, \ y_i \in \mathcal{Y}, \ i \in 1, \dots, N$, полученный из неизвестной закономерности $y = f(\mathbf{x})$. Необходимо построить такую $h(\mathbf{x})$, которая наиболее точно аппроксимирует $f(\mathbf{x})$.

Будем искать неизвестную

$$h(\mathbf{x}) = C(a_1(\mathbf{x}), \dots, a_T(\mathbf{x}))$$

$a_i(\mathbf{x}) : \mathcal{X} \to \mathcal{R}, \ \forall i \in \{1, \dots, T\}$ - базовые модели
$C : \mathcal{R} \to \mathcal{Y}$ - решающее правило

Алгоритмические композиции

Simple Voting

$$h(\mathbf{x}) = \frac{1}{T}\sum_{i=1}^{T} a_i(\mathbf{x})$$

Weighted Voting

$$h(\mathbf{x}) = \frac{1}{T}\sum_{i=1}^{T} b_i a_i(\mathbf{x}),\ \ b_i \in \mathcal{R}$$

Mixture of Experts

$$h(\mathbf{x}) = \frac{1}{T}\sum_{i=1}^{T} b_i(\mathbf{x}) a_i(\mathbf{x}),\ \ b_i(\mathbf{x}) : \mathcal{X} \to \mathcal{R}$$

Идея бустинга

Снова выбираем ноутбук

Вместо того, чтобы спрашивать у всех экспертов, какой ноутбук выбрать, будем после каждого очередного мнения менять наш вопрос.

- Более реалистичная модель
- Уточняем наш вопрос в зависимости от ответов предыдущих экспертов

Boosting vs Bagging

Стрельба в тире

Игра в гольф

Бустинг для задачи бинарной классификации

Пусть

$$Y = -1, +1, \quad a_i : X \to \{-1, 0, +1\},\ i = 1 \dots T$$

Будем строить модель взвешенного голосования:

$$h(\mathbf{x}) = sign\left(\sum_{i=1}^{T} b_i a_i(\mathbf{x})\right),\ b_i \in \mathcal{R}$$

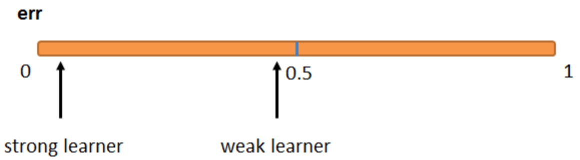
Функция потерь

$$err(h) = \frac{1}{N}\sum_{j=1}^{N} I(y_j \neq h(\mathbf{x}_j))$$

Вопрос

- Возможно ли построение “сильного” алгоритма из набора слабых?

Слабые модели



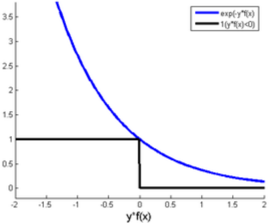
Ключевые идеи бустинга

- Используем “слабые” модели a_i
- Последовательно применяем слабые модели к “слегка” изменяемым версиям оригинальных данных
- При добавлении очередной модели a_i, предыдущие $i - 1$ моделей не меняются
- Аппроксимируем функцию потерь гладкой функцией

Аппроксимация пороговой функции экспонентой

Очевидно, что верно соотношение

$$I(y \neq a(\mathbf{x})) = I(y \cdot a(\mathbf{x}) \leq 0) \leq e^{-y \cdot a(\mathbf{x})}$$



Аппроксимация пороговой функции экспонентой

Очевидно, что верно соотношение

$$I(y \neq a(\mathbf{x})) = I(y \cdot a(\mathbf{x}) \leq 0) \leq e^{-y \cdot a(\mathbf{x})}$$

Тогда ошибку композиции $h(\mathbf{x})$ можно оценить сверху

$$0 \leq err(h) = \frac{1}{N} \sum_{j=1}^{N} I(y_j \neq h(\mathbf{x}_j)) \leq$$

$$\leq \frac{1}{N} \sum_{j=1}^{N} exp\left(-y_j \sum_{i=1}^{T-1} b_i a_i(\mathbf{x}_j)\right) \cdot exp(-y_j b_T a_T(\mathbf{x}_j)) = \hat{err}(h) = \hat{err}(h, T) \to 0$$

Обозначим

$$w_j = exp\left(-y_j \sum_{i=1}^{T-1} b_i a_i(\mathbf{x}_j)\right), \quad \widehat{w}_j = \frac{w_j}{\sum_{i=1}^{N} w_i} \quad j = 1, \ldots, N$$

Аппроксимация пороговой функции экспонентой

Введем обозначения

$$N(a, \widehat{\mathbf{w}}) = \sum_{j=1}^{N} \widehat{w}_j I(a(\mathbf{x}_j) \neq y_j)$$

$$P(a, \widehat{\mathbf{w}}) = \sum_{j=1}^{N} \widehat{w}_j I(a(\mathbf{x}_j) = y_j)$$

Тогда справедливо утверждение:

Утверждение:

Пусть для нормированного вектора весов $\widehat{\mathbf{w}}$ существует алгоритм a, способный классифицировать таким образом, что $P(a, \widehat{\mathbf{w}}) > N(a, \widehat{\mathbf{w}})$. Тогда минимум функционала $\hat{err}(h)$ достигает при

$$a_T = arg \max_a \sqrt{P(a, \widehat{\mathbf{w}})} - \sqrt{N(a, \widehat{\mathbf{w}})}$$

$$b_T = \frac{1}{2} ln \frac{P(a, \widehat{\mathbf{w}})}{N(a, \widehat{\mathbf{w}})}$$

Аппроксимация пороговой функции экспонентой

Доказательство:

Воспользуемся тождеством
$e^{-ba} = e^{-b}I(a=1) + e^{b}I(a=-1) + I(a=0), \quad b \in \mathcal{R},\ a \in -1, 0, +1.$
Тогда

$$\hat{err}(T) = \sum_{j=1}^{N} w_j \{ e^{-b} \sum_{j=1}^{N} \widehat{w}_j I(a(\mathbf{x}_j) = y_j) + e^{b} \sum_{j=1}^{N} \widehat{w}_j I(a(\mathbf{x}_j) \neq y_j) +$$

$$+ \sum_{j=1}^{N} \widehat{w}_j I(a(\mathbf{x}_j) = 0) \} = \hat{err}(T-1)(e^{-b}P + e^{b}N + (1 - P - N))$$

Дифференцируем $\hat{err}(T)$ по параметру b и приравниваем к нулю производную, получим:

$$b_T = \frac{1}{2} ln \frac{P}{N}$$

Аппроксимация пороговой функции экспонентой

$$\hat{err}(T) = \hat{err}(T-1)(e^{-b}P + e^{b}N + (1 - P - N))$$

Дифференцируем $\hat{err}(T)$ по параметру b и приравниваем к нулю производную, получим:

$$b_T = \frac{1}{2} ln \frac{P}{N}$$

Подставляем это значение $\hat{err}(T)$:

$$\hat{err}(T) = \hat{err}(T-1)(1 - (\sqrt{P} - \sqrt{N})^2)$$

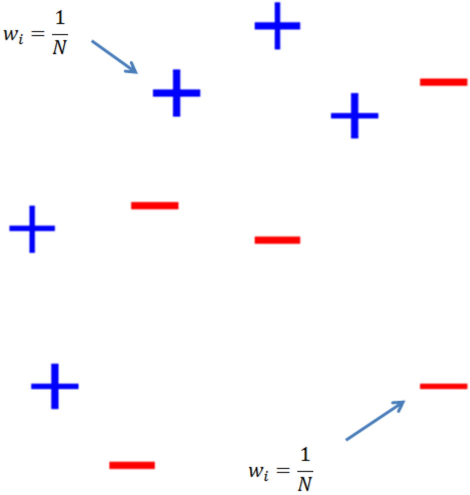
Таким образом

$$a_T = arg \max_a \sqrt{P} - \sqrt{N}$$

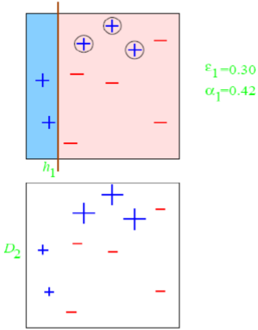
AdaBoost(Freund & Shapire 1995)

1. Инициализировать веса объектов $w_j = 1/N$, $j = 1, 2, \ldots, N$.
2. Для всех i от 1 до T:
   (a) Построить классификатор $a_i(\mathbf{x})$, используя веса w_j
   (b) Вычислить
   $$err_i = \frac{\sum_{j=1}^{N} w_j I(y_j \neq a_i(\mathbf{x}_j))}{\sum_{j=1}^{N} w_j}$$
   (c) Вычислить
   $$b_i = \frac{1}{2} \log \frac{1 - err_i}{err_i}$$
   (d) Присвоить $w_j \to w_j \cdot exp[b_i \cdot I(y_j \neq a_i(\mathbf{x}_j))]$, $j = 1, \ldots, N$.
   (e) Нормируем веса объектов
   $$w_j \to \frac{w_j}{\sum\limits_{k=1}^{N} w_k}, \ j = 1, \ldots, N.$$
3. $h(\mathbf{x}) = sign\left[\sum_{i=1}^{T} b_i a_i(\mathbf{x})\right]$

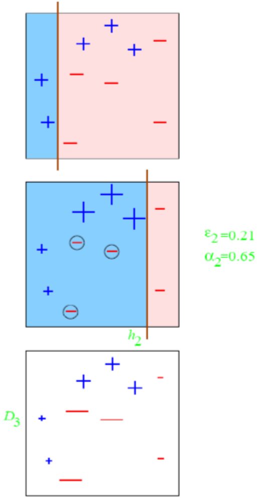
AdaBoost. Пример



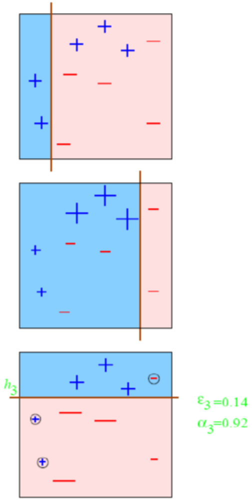
AdaBoost. Пример



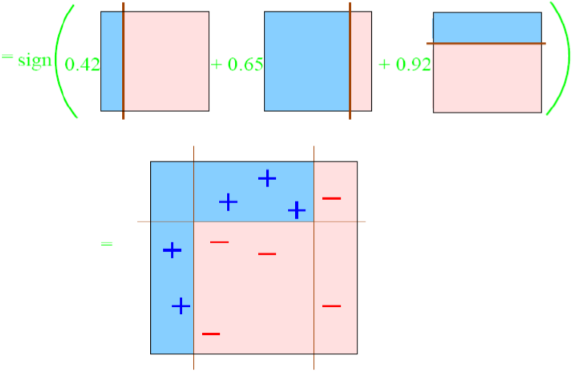
AdaBoost. Пример



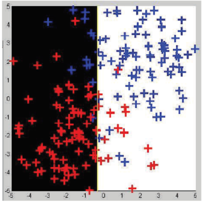
AdaBoost. Пример



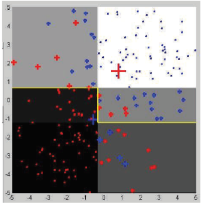
AdaBoost. Пример



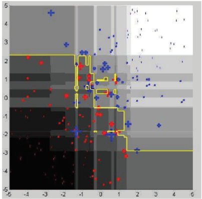
AdaBoost. Еще пример



(a)

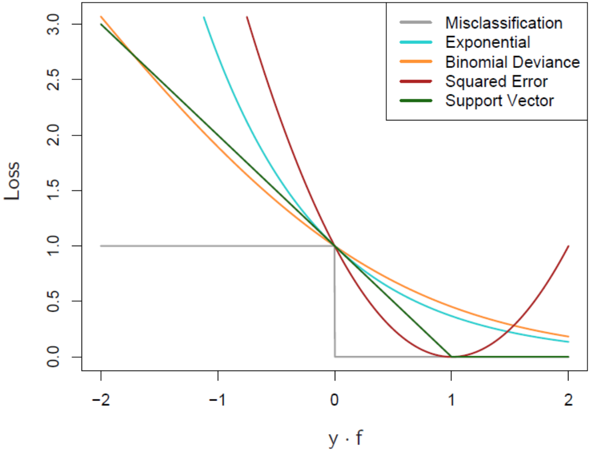


(b)



(c)

Почему именно аппроксимация экспонентой?



Проблемы с экспоненциальной функцией потерь

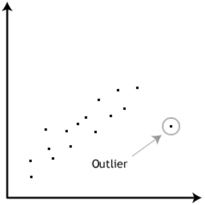
Вопрос:

- Какие проблемы с использованием экспоненциальной функцией потерь вы видите?

Проблемы с экспоненциальной функцией потерь

Вопрос:

- Какие проблемы с использованием экспоненциальной функцией потерь вы видите?



Какие базовые модели выбрать?

Теорема:

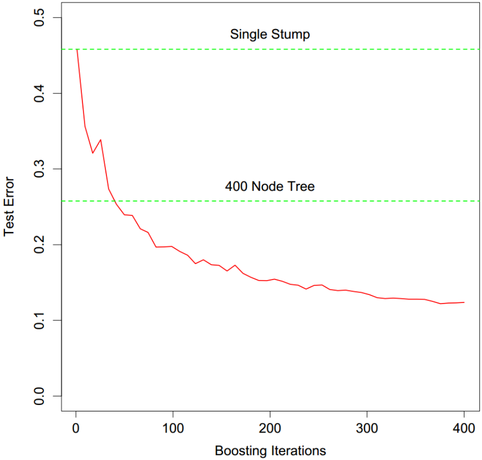
Если на каждой итерации метод обучения позволяет построить базовую модель, такую что число верных классификаций больше, чем число неверных. Тогда метод обучения сходится за конечное число шагов.

- Базовые модели должны обладать достаточной сложностью, чтобы обеспечить это требование.

Вопрос:

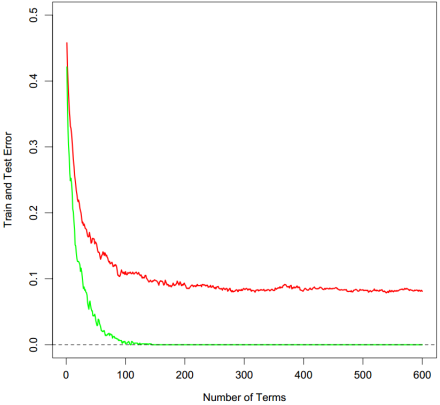
- Можно ли использовать в качестве базовых моделей линейную регрессию для алгоритма AdaBoost?

Boosting decision stumps



$$Y = 1, \quad if \sum_{j=1}^{10} X_j^2 > 9.34, \; otherwise \quad 0$$

Удивительные факты о бустинге



- не переобучается с увеличением числа итераций

Bias & variance

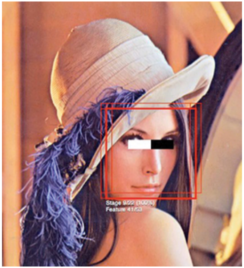
Оценка ожидания ошибки на некоторой точки $\mathbf{x}_0$

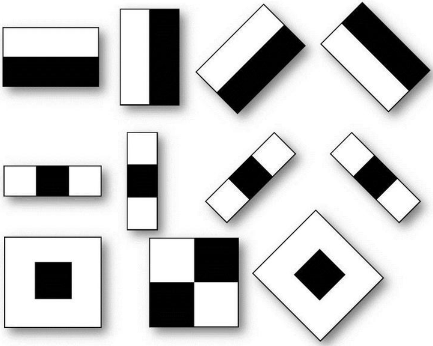
$$Err(\mathbf{x}_0) = Bias^2 + Variance + Noise(Irreducible\ Error)$$

- Бэггинг уменьшает *Variance*
- Бустинг уменьшает и *Variance* и *Bias*. Причем начальные модели отвечают за уменьшение именно *Bias*, а остальные за уменьшение *Variance*.

Применение AdaBoost

Viola–Jones object detection framework





$$a_i(\mathbf{x}_j) = \left[\begin{array}{ll} \alpha_i & \text{если } f_i(\mathbf{x}_j) > \theta_i \\ \beta_i & \text{иначе} \end{array} \right.$$

AdaBoost. Итоги

- ✔ Алгоритм прост
- ✔ Накладные расходы бустинга минимальны. Время построения определяется временем построения базовых моделей
- ✔ Показывает хорошую обобщающую способность
- ✔ Имеет возможность идентификации шумовых объектов
- ✖ Жадное добавление алгоритмов приводит к неоптимальности композиции
- ✖ Склонен к переобучению при наличии шума в данных (опять же из-за экспоненциальной функции потерь)
- ✖ Переобучается при “малом” количестве данных

Градиентный бустинг

Модель взвешенного голосования

$$h(\mathbf{x}) = \sum_{i=1}^{T} b_i a_i(\mathbf{x}), \quad \mathbf{x} \in X, b_i \in R$$

Ошибка композиции на обучающей выборке

$$err(h) = \sum_{j=1}^{N} L(y_j, \sum_{i=1}^{T-1} b_i a_i(\mathbf{x}_j) + b \cdot a(\mathbf{x}_j)) \to \min_{b,a}$$

- Применяем жадную стратегию добавления моделей
- Как и раньше оставляем построенные алгоритмы с весами неизменными

Градиентный бустинг

Применим метод минимизации (метод наискорейшего спуска по параметрам $[h(x_1), \dots, h(x_N)]$) $err(h)$:

Тогда

$$h_{i,j} = h_{i-1,j} - \eta \cdot err'(h_{i-1,j}), \quad j = 1, \dots, N$$

η - шаг спуска

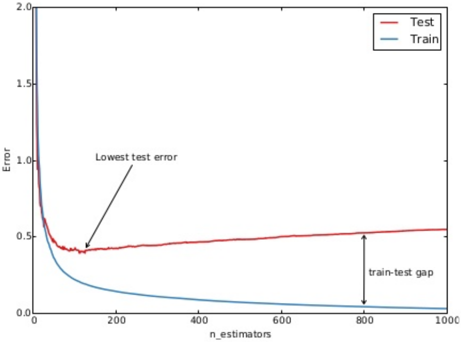
Основная идея:

На каждом шаге алгоритма будем искать модель a_i, которая аппроксимировала бы вектор антиградиента.

Gradient boosting algorithm

1. Инициализировать $h_0(\mathbf{x}) = argmin_\gamma \sum_{j=1}^{N} L(y_j, \gamma)$
2. Для всех i от 1 до T:
   (a) Для всех $j = 1, 2, \ldots, N$ вычислить
   $$g_{i,j} = -\left[\frac{\partial L(y_j, h(\mathbf{x}_j))}{\partial h(\mathbf{x}_j)}\right]_{h=h_{i-1}}$$
   (b) Построить базовую модель a_i на ответах $g_{i,j}$
   $$a_i = \arg\min_a \sum_{j=1}^{N} (g_{i,j} - a(\mathbf{x}_j))^2$$
   (c) Определить вес b_i
   $$b_i = \arg\min_b \sum_{j=1}^{N} L(y_j, h_{i-1}(\mathbf{x}) + b \cdot a_i(\mathbf{x}_j))$$
   (d) Присвоить $h_i(\mathbf{x}) = h_{i-1}(\mathbf{x}) + b_i \cdot a_i(\mathbf{x})$
3. Вернуть $h(\mathbf{x}) = h_T(\mathbf{x})$

Gradient boosting & overfitting

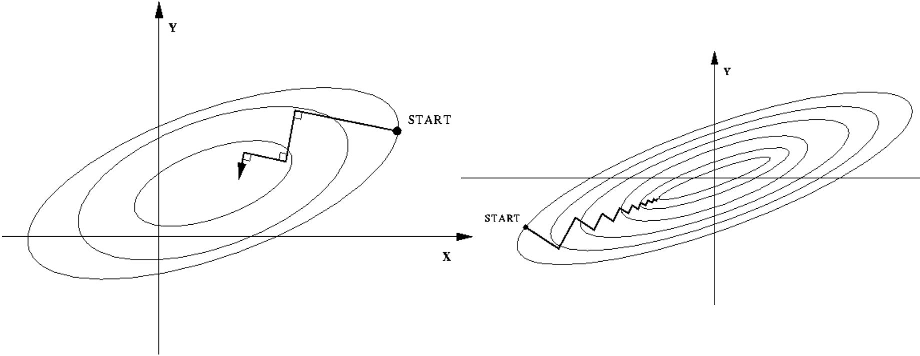


- Необходимо подбирать число деревьев на валидационной выборке

Shrinkage

Идея:

- Будем делать шаг каждым алгоритмом с некоторым дисконтом



$$h_i(\mathbf{x}) = h_{i-1}(\mathbf{x}) + \mu \cdot b_i \cdot a_i(\mathbf{x})$$

- Дадим гольфисту тяжелую клюшку, чтоб он не мог ударить сильно

Stohastic gradient boosting

Stohastic gradient boosting = Gradient Boosting + Bagging

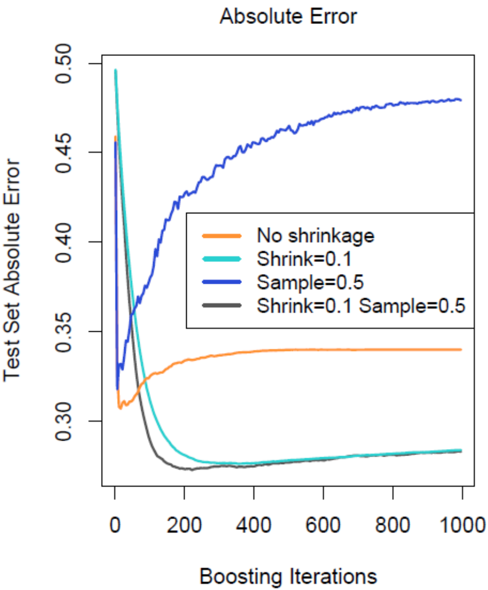
Идея:

- Для построения очередного алгоритма используются случайная подвыборка, полученная алгоритмом выбора без возвращения
- Выдадим гольфисту шары разной формы

Преимущества

- Уменьшается время обучения
- Лучше сходится (эквивалентно методу стохастического градиентного спуска)
- Имеет более высокую обобщающую способность

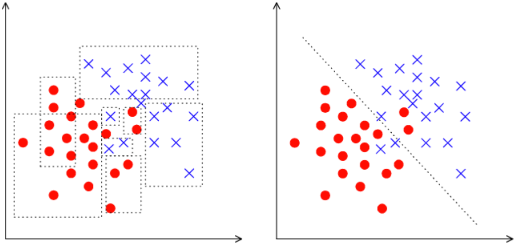
Shrinkage & Subsampling



Regularization in leafs

Идея:

- Не все листы имеют одинаковый вклад
- “Важность” листа зависит от числа объектов в нем
- Дадим гольфисту палку вместо клюшки



Варианты решения:

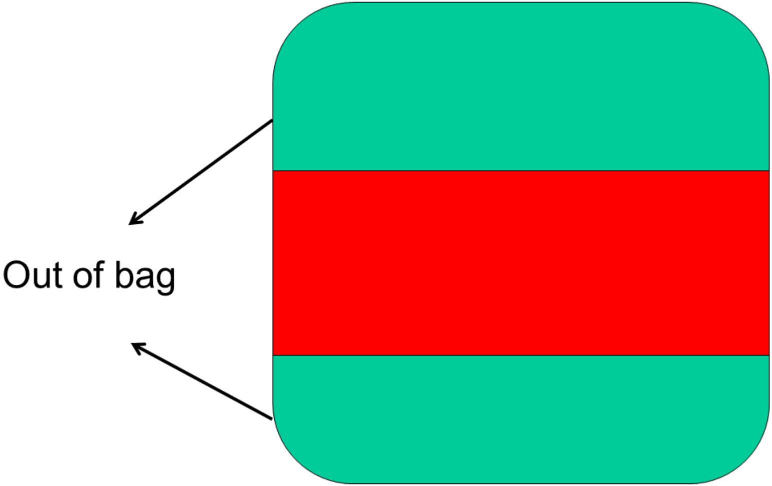
- Ограничить минимальное число примеров в листьях
- Домножить значения в листьях на некоторую фукцию, зависящую от количества примеров

Height of trees

Какая высота деревьев правильная?

- Берем всегда маленькие деревья
- Берем всегда большие деревья
- Подбираем высоту по критерию качества
- Ваше мнение...

OOB



- Оцените алгоритмическую сложность алгоритма градиентного бустинга на деревьях решений

Gradient boosting algorithm complexity

- Вычисление градиента $O(g(N))$
- Построение модели (дерева) $O(ND)$
- Вычисление предсказания $O(N)$

Построение дерева может быть эффективно распараллелено по фичам. Поэтому в целом быстро.

Gradient boosting. Итоги

- Gradient boosting - общий алгоритм бустинга. Позволяет работать с произвольными функциями потерь и пространствами ответов.
- Чаще всего применяется с деревьями решений.
- Показывает наилучшее качество для задач классификации, регресии и ранжирования.
- Применять надо с регуляризацией, иначе результаты могут получиться удручающими.

Вопрос:

- А что делать если функция не дифференцируема?

Стохастические алгоритмы и бустинг. Итоги

- ✔ Бустинг лучше работает для больших обучающих выборок в ситуациях когда в данных имеются сложные зависимости.
- ✔ Стохастические методы лучше работают для коротких обучающих выборок
- ✔ Стохастические алгоритмы можно эффективно распараллелить. Бустинг предполагает последовательное построение композиции.
- ✔ RSM наиболее эффективен в пространствах большой размерности
- ✔ Для бустинга лучше строить длинные композиции из слабых моделей, чем короткие из сильных

Вопрос:

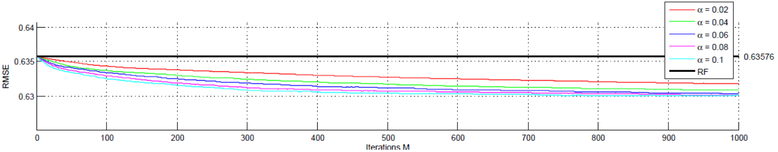
- А всегда ли стоит использовать деревянные модели bagging & boosting?

iGBRT

Initialized Gradient Boosted Regression Trees

Идея:

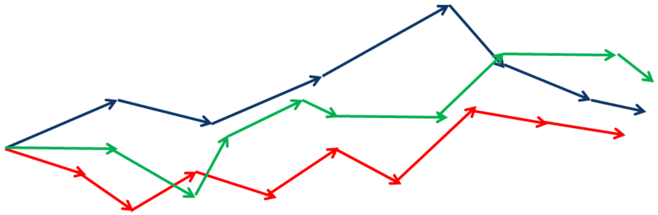
Проинициализируем градиентный бустинг Random Forest-ом



BagBoo & BooBag

Идея:

- Объединим стратегии бустинга и бэггинга



- Если объединять бустинг бэггингом то BagBoo
- Если объединять бэггинг бустингом то BooBag

Итоги

- Бустингом бэггинг не испортишь! (и наоборот)
- При таком подходе реальное качество почти неограничено... все упирается в число деревьев
- К сожалению, не получается применять в больших высоконагруженных системах
- Если есть много машин, то можно оценить верхнюю границу качества системы машинного обучения

Задача

Дано: Имеется набор данных из системы поискового антиспама.
Требуется: Требуется сравнить классификаторы, основанные на алгоритмических композициях, с методом опорных векторов.

Пошаговая инструкция

1. Скачать данные и запустить шаблон кода на python
   `https://goo.gl/Yuvr8Z`

```
$ python compos.py -h
$ python compos.py -tr spam.train.txt -te spam.test.txt
```

2. Подобрать параметры 3х алгоритмических композиций, чтобы они превосходили по качеству SVM.
3. Построить графики качества классификации, в зависимости от числа базовых моделей.

Дз по алгоритмическим композициям:

Задание:

Реализовать один из алгоритмов машинного обучения, являющегося композицией алгоритмов. (Максимум 20 баллов)

Имеется 27 вариантов задания:
Для того, чтобы узнать свой вариант необходимо выполнить функцию:

```
def ComputeMyTaskNumber(your_name):
    return 1 + hash(your_name) % 27
```

где your_name - это ваши фамилия и имя латиницей (например 'Pupkin Vasiliy')

Варианты:

1. Реализация модельного дерева решений с линейной регрессией в листьях (задача регрессии)
2. Реализация модельного дерева решений с логистической регрессией в листьях (задача классификации)
3. Реализация модельного дерева решений с методом ближайших соседей в листьях (задача классификации)
4. Реализация модельного дерева решений с полиномиальной регрессией в листьях (задача регрессии)
5. Реализация алгоритма Random Forest (базовый алгоритм CART) с добавлением линейных признаков (задача регресии)
6. Реализация алгоритма Random Forest (базовый алгоритм CART) с добавлением линейных признаков (задача классификации)
7. Реализация алгоритма Random Forest (базовый алгоритм CART) + Реализация алгоритма градиентного бустинга с квадратичной функцией потерь. В качестве базового алгоритма использовать алгоритм CART.
8. Реализация алгоритма Random Forest (базовый алгоритм CART) + Реализация алгоритма градиентного бустинга с логистической функцией потерь. В качестве базового алгоритма использовать алгоритм CART.

Варианты:

9. Реализация алгоритма AdaBoost (базовый алгоритм CART)
10. Реализация алгоритма градиентного бустинга с абсолютной функцией потерь (базовый алгоритм CART)
11. Реализация алгоритма градиентного бустинга с квадратичной функцией потерь. В качестве базового алгоритма использовать алгоритм CART.
12. Реализация алгоритма градиентного бустинга с логистической футкцией потерь. В качестве базового алгоритма использовать алгоритм CART.
13. Реализация алгоритма градиентного бустинга с квадратичной функцией потерь. В качестве базового алгоритма использовать алгоритм CART с RSM.
14. Реализация алгоритма градиентного бустинга с логистической футкцией потерь. В качестве базового алгоритма использовать алгоритм CART с RSM.
15. Реализация алгоритма стохастического градиентного бустинга с квадратичной функцией потерь. В качестве базового алгоритма использовать алгоритм CART.

Варианты:

16. Реализация алгоритма стохастического градиентного бустинга с логистической футкцией потерь. В качестве базового алгоритма использовать алгоритм CART.
17. Реализация алгоритма стохастического градиентного бустинга с квадратичной функцией потерь. В качестве базового алгоритма использовать алгоритм CART с RSM.
18. Реализация алгоритма стохастического градиентного бустинга с логистической футкцией потерь. В качестве базового алгоритма использовать алгоритм CART с RSM.
19. Реализация алгоритма Extremely randomized trees с линейными признаками для задачи регресии
20. Реализация алгоритма Extremely randomized trees с линейными признаками для задачи классификации

Варианты:

21. Реализация алгоритма BagBoo. В качестве базового алгоритма использовать алгоритм градиентного бустинга с функцией потерь (регрессия).
22. Реализация алгоритма BooBag. В качестве базового алгоритма использовать алгоритм градиентного бустинга с функцией потерь (регрессия).
23. Реализация алгоритма BagBoo. В качестве базового алгоритма использовать алгоритм градиентного бустинга с логистической функцией потерь.
24. Реализация алгоритма BooBag. В качестве базового алгоритма использовать алгоритм градиентного бустинга с логистической функцией потерь.

Варианты:

25. Взвешенное голосование методов градиентного бустинга с логистической функцией потерь (базовый алгоритм CART), обученных на разных подмножествах фичей
26. Взвешенное голосование методов градиентного бустинга с логистической функцией потерь (базовый алгоритм CART) и методов Random Forest, обученных на разных подмножествах фичей
27. Взвешенное голосование методов ближайшего соседа, градиентного бустинга с логистической функцией потерь (базовый алгоритм CART), и логистической регрессии

Данные UCI:

Для задач регрессии следует использовать тестовые датасеты:

- https://archive.ics.uci.edu/ml/datasets/Housing
- https://archive.ics.uci.edu/ml/datasets/Auto+MPG
- https:
  //archive.ics.uci.edu/ml/datasets/Computer+Hardware

Для задач классификации следует использовать тестовые датасеты:

- https://archive.ics.uci.edu/ml/datasets/Wine
- https://archive.ics.uci.edu/ml/datasets/Iris
- https:
  //archive.ics.uci.edu/ml/datasets/Liver+Disorders

Приложение

О правилах сдачи данного ДЗ:

Ваше решение должно быть не более чем на 3% хуже, чем решение из scikitlearn-а в относительных числах при одинаковых параметрах запуска для спамовского датасета. В момент проверки дз первое, что вы показываете это графики ошибки на обучении и на тесте для вашего алгоритма и библиотечного. Также на графике должна быть обозначена относительная 3-х процентная граница. Пока данный результат не достигнут баллы за данное дз получить невозможно.

Приложение

Квадратичная функция потерь

$$L(h) = \sum_{i=1}^{N} (y_i - h(\mathbf{x}_i))^2$$

Абсолютная функция потерь

$$L(h) = \sum_{i=1}^{N} |y_i - h(\mathbf{x}_i)|$$

Логистическая функция потерь

$$L(h) = \sum_{i=1}^{N} (-y_i \log(f(\mathbf{x}_i)) - (1 - y_i) \log(1 - f(\mathbf{x}_i)))$$

$$f(\mathbf{x}_i) = \sigma(h(\mathbf{x}_i)) = \frac{1}{1 + e^{-h(\mathbf{x}_i)}}$$

Вопросы