



# ТЕХНОСФЕРА

## Лекция 2 Алгоритмические композиции Бустинг

Владимир Гулин

9 сентября 2019 г.

# План лекции

Напоминание

Бустинг

Градиентный бустинг

Мета-алгоритмы

# 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

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

$a_i(\mathbf{x}) : \mathcal{X} \rightarrow \mathcal{R}$ ,  $\forall i \in \{1, \dots, T\}$  - базовые модели

$C : \mathcal{R} \rightarrow \mathcal{Y}$  - решающее правило

# Алгоритмические композиции

## Simple Voting

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

## Weighted Voting

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

## Mixture of Experts

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

# 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

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

Будем строить модель взвешенного голосования:

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

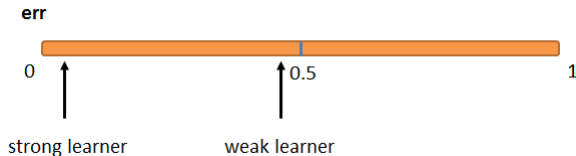
Функция потерь

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

# Вопрос

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

## Слабые модели





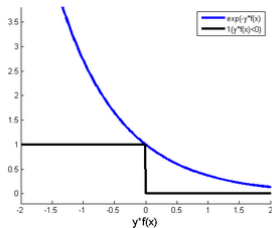
# Ключевые идеи бустинга

- ▶ 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

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

Обозначим

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

# Аппроксимация пороговой функции экспонентой

Введем обозначения

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

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

Тогда справедливо утверждение:

Утверждение:

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

$$a_T = \arg \max_a \sqrt{P(a, \hat{\mathbf{w}})} - \sqrt{N(a, \hat{\mathbf{w}})}, \quad b_T = \frac{1}{2} \ln \frac{P(a, \hat{\mathbf{w}})}{N(a, \hat{\mathbf{w}})}$$

# Аппроксимация пороговой функции экспонентой

## Доказательство:

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

$$\begin{aligned} \hat{err}(T) &= \sum_{j=1}^N w_j \left\{ e^{-b} \sum_{j=1}^N \hat{w}_j I(a(\mathbf{x}_j) = y_j) + e^b \sum_{j=1}^N \hat{w}_j I(a(\mathbf{x}_j) \neq y_j) + \right. \\ &\quad \left. + \sum_{j=1}^N \hat{w}_j I(a(\mathbf{x}_j) = 0) \right\} = \hat{err}(T - 1)(e^{-b}P + e^bN + (1 - P - N)) \end{aligned}$$

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

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

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  $w_j = 1/N, j = 1, 2, \dots, N$ .
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

$$b_i = \frac{1}{2} \log \frac{1 - err_i}{err_i}$$

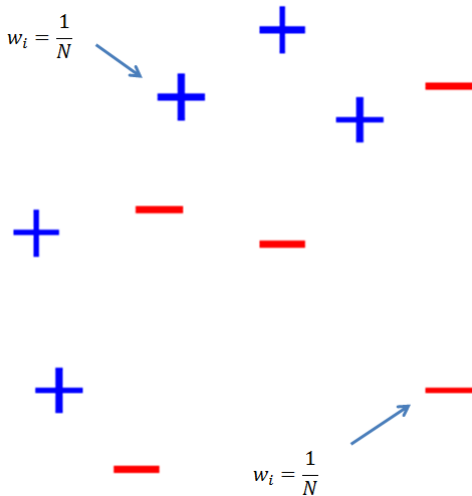
(d) Присвоить  $w_j \rightarrow w_j \cdot \exp[b_i \cdot I(y_j \neq a_i(\mathbf{x}_j))], j = 1, \dots, N$ .

(e) Нормируем веса объектов

$$w_j \rightarrow \frac{w_j}{\sum_{k=1}^N w_k}, j = 1, \dots, N.$$

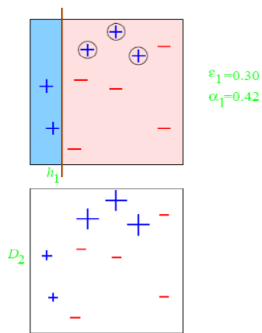
3.  $h(\mathbf{x}) = \text{sign} \left[ \sum_{i=1}^T b_i a_i(\mathbf{x}) \right]$

## AdaBoost. Пример

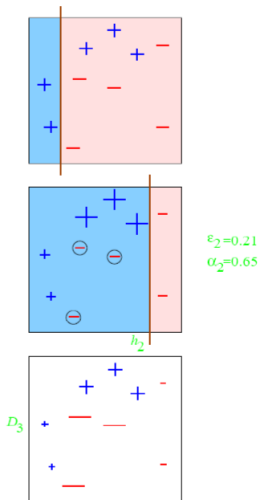




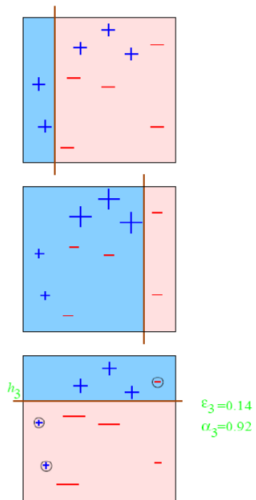
# AdaBoost. Пример



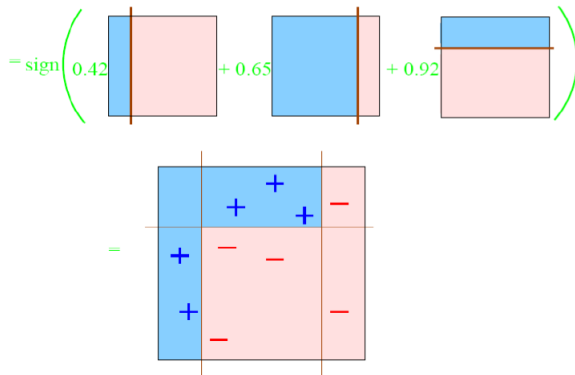
# AdaBoost. Пример



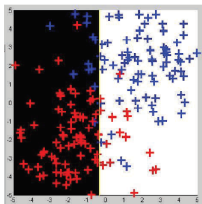
# AdaBoost. Пример



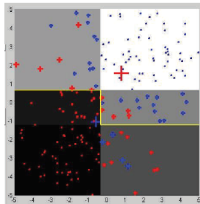
# AdaBoost. Пример



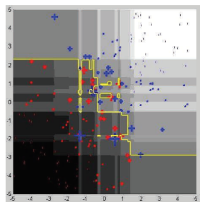
# AdaBoost. Еще пример



(a)

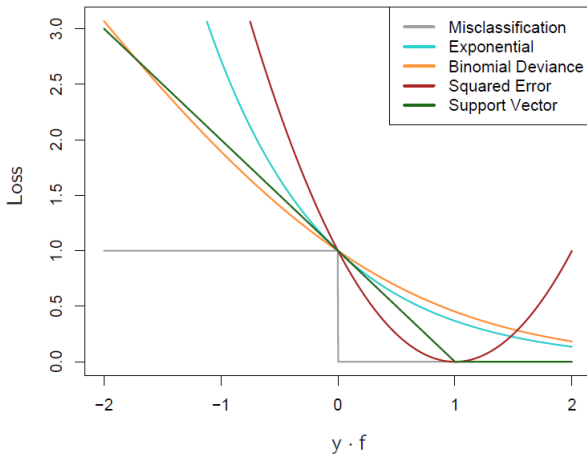


(b)



(c)

# Почему именно аппроксимация экспонентой?



# Проблемы с экспоненциальной функцией потерь

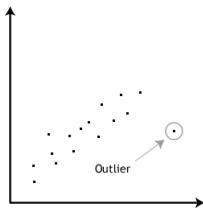
Вопрос:

- ▶ 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:

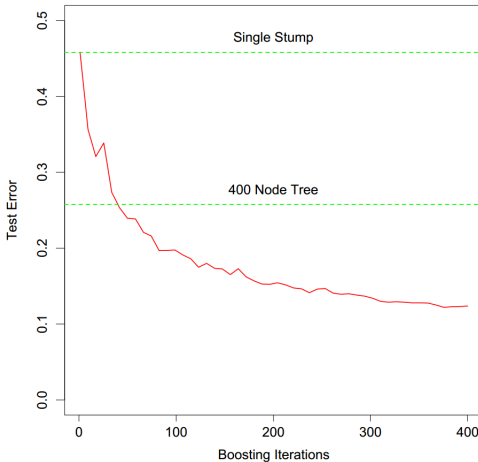
Если на каждой итерации метод обучения позволяет построить базовую модель, такую что число верных классификаций больше, чем число неверных. Тогда метод обучения сходится за конечное число шагов.

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

## Вопрос:

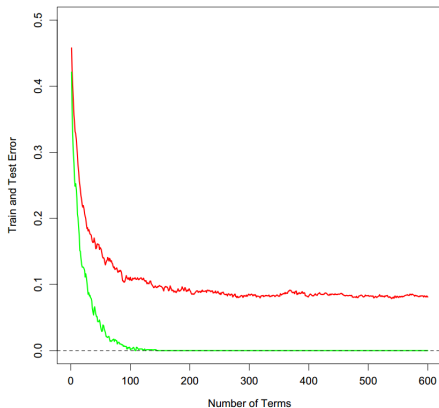
- ▶ Можно ли использовать в качестве базовых моделей линейную регрессию для алгоритма AdaBoost?

# Boosting decision stumps



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

# Удивительные факты о бустинге



- ▶ не переобучается с увеличением числа итераций
- ▶ ошибка на тесте продолжает уменьшаться даже после достижения безошибочной классификации на train выборке

# Bias & variance

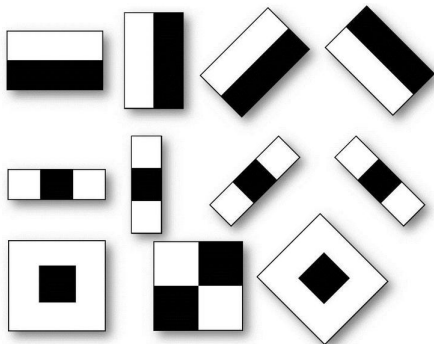
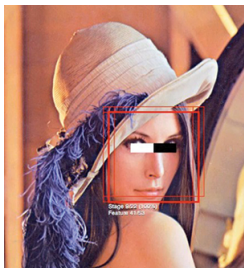
Оценка ожидания ошибки на некоторой точки  $\mathbf{x}_0$

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

- ▶ 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



$$a_i(\mathbf{x}_j) = \begin{cases} \alpha_i & \text{если } f_i(\mathbf{x}_j) > \theta_i \\ \beta_i & \text{иначе} \end{cases}$$

# 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

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

- ▶ 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

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

$\eta$  - шаг спуска

Основная идея:

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



# Gradient boosting algorithm

1. Инициализировать  $h_0(\mathbf{x}) = \operatorname{argmin}_{\gamma} \sum_{j=1}^N L(y_j, \gamma)$
2. Для всех  $i$  от 1 до  $T$ :
  - (a) Для всех  $j = 1, 2, \dots, N$  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  $g_{i,j}$

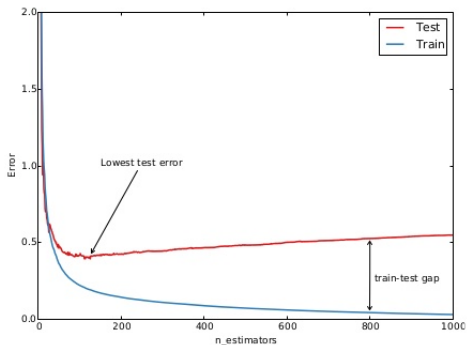
$$a_i = \operatorname{argmin}_a \sum_{j=1}^N (g_{i,j} - a(\mathbf{x}_j))^2$$

- (c) Определить вес  $b_i$

$$b_i = \operatorname{argmin}_b \sum_{j=1}^N L(y_j, h_{i-1}(\mathbf{x}) + b \cdot a_i(\mathbf{x}_j))$$

- (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

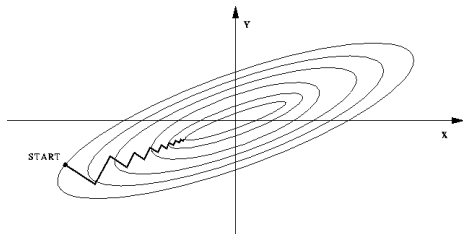
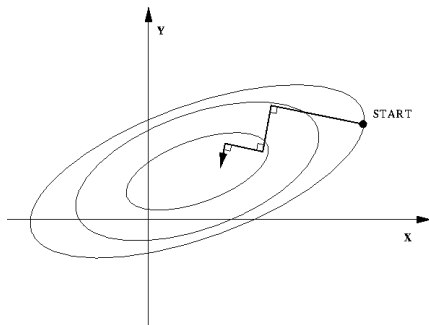


- Необходимо подбирать число деревьев на валидационной выборке

# 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

# Stochastic gradient boosting

Stochastic gradient boosting = Gradient Boosting + Bagging

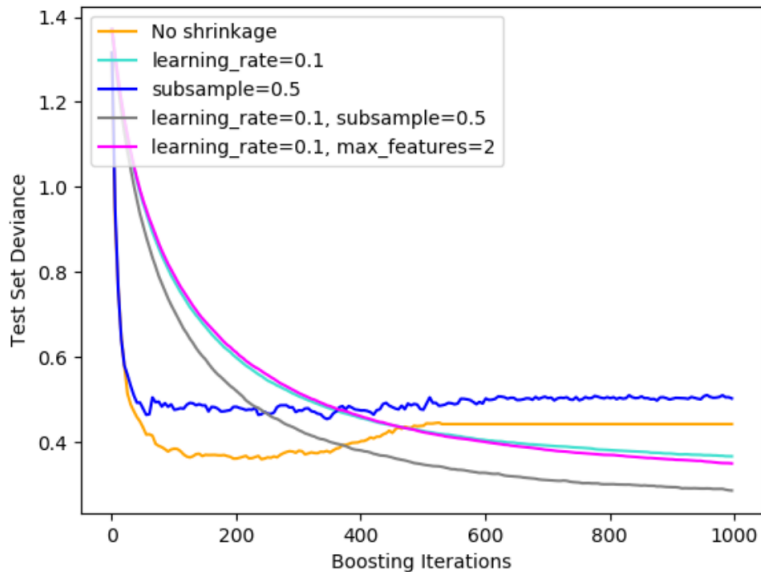
## Идея:

- ▶ 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

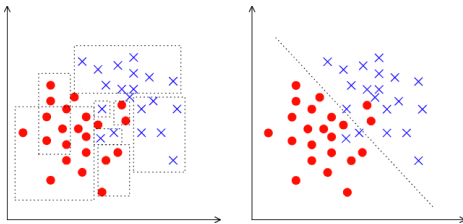
# Shrinkage & Subsampling



# 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 функцию, зависящую от количества примеров

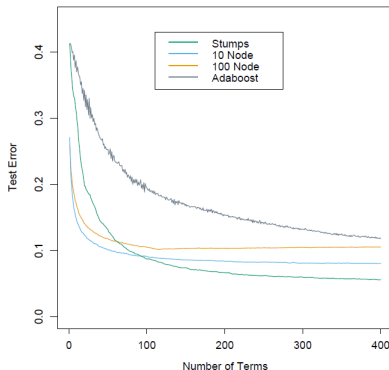
# Height of trees

Какая высота деревьев правильная?

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

# Height of trees

Высота деревьев определяет feature interaction модели

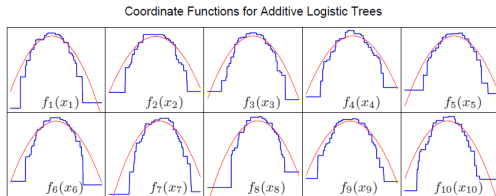


$$\eta(X) = \sum_j \eta_j(X_j) + \sum_{jk} \eta_{jk}(X_j, X_k) + \sum_{jkl} \eta_{jkl}(X_j, X_k, X_l) + \dots$$



# Height of trees

## Пример



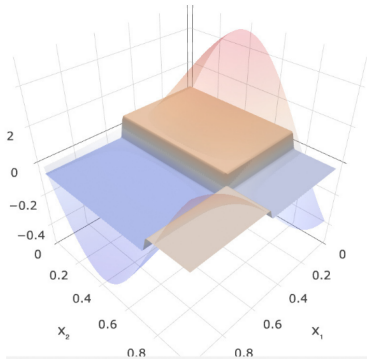
Так как истинная разделяющая поверхность между данными описывается сферой

$$f(X) = X_1^2 + X_2^2 + \dots + X_{10}^2 - c = 0$$

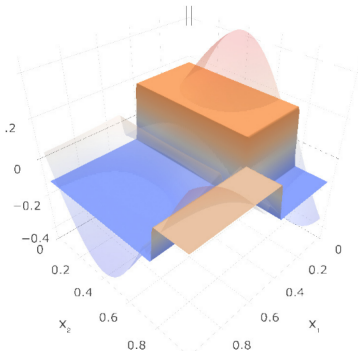
выигрывают градиентно забущенные штампы.

# Побаловаться

target function  $f(\mathbf{x})$  and prediction of previous trees  $D(\mathbf{x})$



residual  $R(\mathbf{x})$  and prediction of next tree  $d_n(\mathbf{x})$



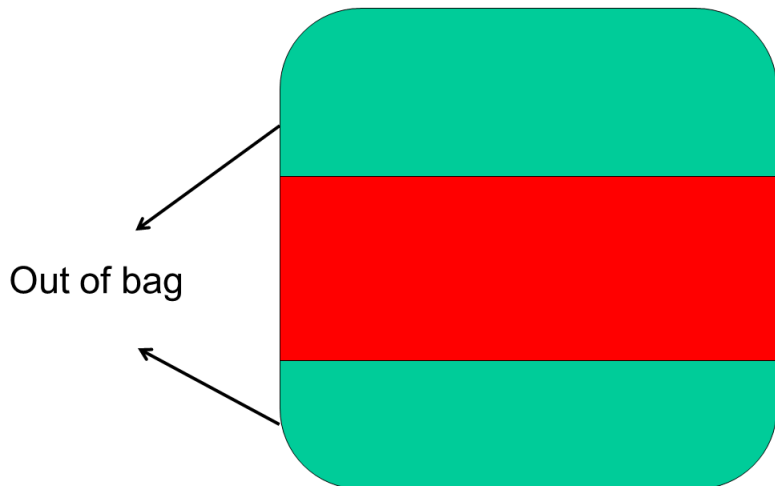
Tree depth: 2



Number of built trees: 1



OOB



- ▶ Оцените алгоритмическую сложность алгоритма градиентного бустинга на деревьях решений

# Gradient boosting algorithm complexity

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

где  $H$  - высота дерева.

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, регрессии 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

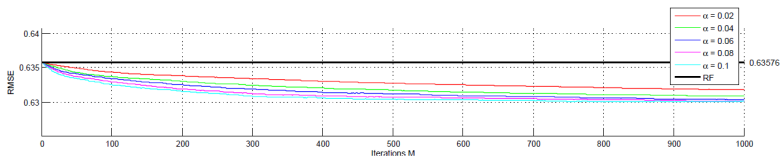
## Вопрос:

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

## Initialized Gradient Boosted Regression Trees

Идея:

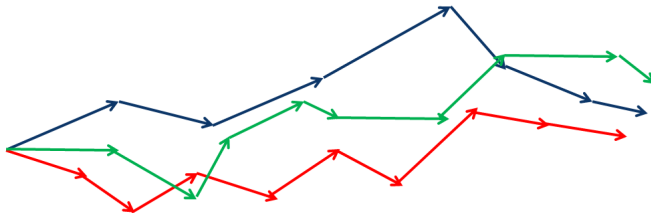
Проинициализируем градиентный бустинг Random Forest-ом



# 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 классическими алгоритмами классификации и нейросетями.

Пошаговая инструкция

1. Скачать данные и запустить шаблон кода на python  
`goo.gl/CCM2Yo`

```
$ 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, логистическую регрессию и двуслойный персептрон.
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 21 вариантов задания:

Для того, чтобы узнать свой вариант необходимо выполнить функцию:

```
def ComputeMyTaskNumber(your_name):  
    return 1 + hash(your_name) % 21
```

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

## Варианты:

1. Реализация модельного дерева решений с логистической регрессией в листьях (задача классификации)
2. Реализация алгоритма Random Forest (базовый алгоритм CART) с добавлением линейных признаков (задача классификации)
3. Реализация алгоритма градиентного бустинга с квадратичной функцией потерь. В качестве базового алгоритма использовать алгоритм CART (с линейными признаками) (задача регрессии)
4. Реализация алгоритма градиентного бустинга с логистической функцией потерь. В качестве базового алгоритма использовать алгоритм CART (задача классификации)
5. Реализация алгоритма стохастического градиентного бустинга с квадратичной функцией потерь. В качестве базового алгоритма использовать алгоритм CART с RSM (задача регрессии)

## Варианты:

6. Реализация алгоритма BagBoo. В качестве базового алгоритма использовать алгоритм градиентного бустинга с функцией потерь (регрессия).
7. Реализация алгоритма BooBag. В качестве базового алгоритма использовать алгоритм градиентного бустинга с логистической функцией потерь.
8. Реализация алгоритма Extremely randomized trees с линейными признаками для задачи регрессии
9. Взвешенное голосование методов ближайшего соседа, градиентного бустинга с логистической функцией потерь (базовый алгоритм CART), и логистической регрессии
10. Реализация алгоритма градиентного бустинга на модельных деревьях решений с линейной регрессией в листьях (задача регрессии).

## Варианты:

11. Реализация алгоритма градиентного бустинга с Bernoulli loss. В качестве базового алгоритма использовать алгоритм CART
12. Реализация алгоритма градиентного бустинга с Adaboost loss. В качестве базового алгоритма использовать алгоритм CART
13. Реализация алгоритма градиентного бустинга с Laplacian loss. В качестве базового алгоритма использовать алгоритм CART (задача регрессии)
14. Blending градиентного бустинга с Bernoulli loss (базовый алгоритм CART), логистической регрессии и двухслойного персептрона (logloss)
15. Логистическая регрессия над градиентным бустингом с Adaboost loss

## Варианты:

16. Реализация алгоритма iGBRT с квадратичной функцией потерь
17. Реализация алгоритма iGBRT с логистической функцией потерь
18. Stacking градиентного бустинга с Bernoulli loss, логистической регрессии, kNN и NaiveBayes
19. Реализация алгоритма градиентного бустинга с Laplacian loss. В качестве базового алгоритма использовать алгоритм CART с линейными признаками (задача регрессии)
20. Blending градиентного бустинга с квадратичной функцией потерь (базовый алгоритм CART), двухслойного персептрона и линейной регрессии (задача регрессии)

## Варианты:

21. Линейная регрессия над градиентным бустингом с квадратичной функцией потерь (базовый алгоритм CART) (задача 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.

## Квадратичная функция потерь

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

## Логистическая функция потерь

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

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

## Laplacian loss

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

## Bernoulli loss

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

## Adaboost loss

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

Вопросы

