1. Лабораторная работа «Градиентный бустинг деревьев решений»

1.1. Общие сведения об алгоритме градиентного бустинга.

Идея бустинг-подхода заключается в комбинации слабых (с невысокой обобщающей способностью) функций, которые строятся в ходе итеративного процесса, где на каждом шаге новая модель обучается с использованием данных об ошибках предыдущих. Результирующая функция представляет собой линейную комбинацию базовых, слабых моделей. Первым алгоритмом данного класса является AdaBoost, который осуществляет минимизацию экспоненциальной функции потерь на обучающей выборке. В общем же виде бустинг-алгоритм может быть представлен как жадный процесс минимизации *любой* штрафной функции. Однако проблема заключается в том, что для некоторой *базовой* модели может не существовать эффективного алгоритма обучения, работающего с учетом выбранной функции потерь. Данную проблему отчасти решает подход, носящий название *градиентного бустинга*, который позволяет без серьезных модификаций алгоритма обучения использовать любой дифференцируемый штраф *L(y, y')*.

Алгоритм 1. Градиентный бустинг.

1. Взять оптимальное константное решение за начальное приближение

$$h_0 = F_0(x) = \arg\min_{\rho} \sum_{i=1}^{N} L(y_i, \rho).$$

2. Для всех $m = 1, \dots, M$
   a. Вычислить компоненты вектора антиградиента

$$r_{i,m} = -\left[\frac{\partial L\big(y_i, F(x_i)\big)}{\partial F(x_i)}\right]_{F=h_{m-1}}, \qquad i = \overline{1, N}.$$

   b. Обучить базовую регрессионную модель на выборке $\{(x_i, r_{i,m}), i = \overline{1, N}\}$

$$\gamma_m = \arg\min_{\beta,\gamma} \sum_{i=1}^{N} \Psi(r_{i,m}, \beta b(x_i;\, \gamma)).$$

   c. Найти длину шага

$$\rho_m = \arg\min_{\rho} \sum_{i=1}^{N} L(y_i, h_{m-1}(x_i) + \rho b(x_i;\, \gamma_m)).$$

   d. Обновить модель

$$h_m(x) = h_{m-1}(x) + \nu \cdot \rho_m b(x; \gamma_m).$$

3. Конечная модель $\hat{f}(x) = h_M(x)$.

Таким образом, алгоритм градиентного бустинга выполянет M итераций, на каждой из которых происходит обучение базовой модели $b(x_i; \gamma)$, путем подбора оптимальных параметров γ. Следует отметить, что для настройки каждой базовой модели используется функция потерь $\Psi(y, y')$, вообще говоря, отличная от $L(y, y')$. Именно благодаря этому факту алгоритм является универсальным по отношению к применяемому штрафу.

В данной лабораторной работе рассматривается алгоритм градиентного бустинга, где в качестве базовых моделей используются деревья решений. Это обусловлено тем, что они позволяют производить обучение на исходных данных без их дополнительной предобработки, поддерживают наличие пропущенных значений, корректно обрабатывают переменные различных типов, и, что немаловажно, существуют эффективные алгоритмы их настройки (CART, C4.5). В рамках данной лабораторной работы для настройки одиночного дерева решений предполагается использование квадратичного штрафа в качестве функции $\Psi(y, y')$ и алгоритма CART.

1.2. Описание возможностей и интерфейса библиотеки GBM.

В данной лабораторной работе в качестве реализации алгоритма градиентного бустинга деревьев решений используется библиотека gbm[1], основная функциональность которой описывается в данном разделе.

1.2.1. Функции gbm и gbm.more

```
gbm( formula = formula(data), distribution = "bernoulli",
data = list(), weights, var.monotone = NULL, n.trees = 100,
interaction.depth = 1, n.minobsinnode = 10, shrinkage = 0.001,
bag.fraction = 0.5, train.fraction = 1.0, cv.folds = 0,
keep.data = TRUE, verbose = TRUE)
```

Функция `gbm` позволяет осуществить обучение модели градиентного бустинга деревьев решений.

[1] Данный пакет предоставляет возможность использования исключительно деревьев решений в качестве базовых моделей. Для применения других функций, могут быть использована библиотека mboost.

Входные параметры[2]*:*

`formula`	символическое описание модели
`distribution`	Строка, содержащая название штрафной функции. Возможны следующие варианты: • `"gaussian"` – квадратичный штраф, $L(y,F) = (y-F)^2$. Применяется для задач регрессии. • `"laplace"` – штраф по абсолютной величине отклонения, $L(y,F) = \lvert y-F \rvert$. Применяется для задач регрессии. • `"bernoulli"` – функция кросс-энтропии (*cross-entropy, negative log-likelihood, deviance*). Применяется для задач бинарной классификации. В случае $y \in \{-1,1\}$ записывается как $L(y,F) = \ln(1+\mathrm{e}^{-2yF})$, где $F(x) = \frac{1}{2}\log\left(\frac{\mathrm{P}(y=1\mid x)}{\mathrm{P}(y=-1\mid x)}\right)$, т.е. $F(x)$ представляет собой функцию, определяющую логарифм отношения вероятностей. Для случая $y' = \frac{y+1}{2} \in \{0,1\}$ функция представляется в виде $L(y',F) = -y'\ln\big(p(x)\big) - (1-y')\ln\big(1-p(x)\big) = -2y'F + \log(1+\exp(2\mathrm{F}))$, где $p(x) = \mathrm{P}(y=1\lvert x) = \frac{1}{1+\exp(-2F)}$. • `"adaboost"` – экспоненциальный штраф. Применяется для задач бинарной классификации. $L(y,F) = \exp(-y\,F)$, $y \in \{-1,1\}$. При указании данного значения параметра обучение производится *не* алгоритмом AdaBoost, а алгоритмом градиентного бустинга деревьев решений с использованием экспоненциального штрафа. • `"poisson"` – Пуассонов штраф. Для решения задач, где выход является натуральным числом. • `"quantile"`. Применяется для задач квантильной регрессии. При использовании данной функции потерь, в

[2] В данном пособии дается описание лишь основных параметров функций. В ходе выполнения лабортаторной работы предполагается использование неописанных параметров со значением по умолчанию. С полным описанием списка параметров функции можно ознакомиться в документации к библиотеке.

	качестве параметра `distribution` необходимо передать не только название функции, но и уровень квантиля, например `distribution = list(name="quantile",alpha=0.25)`. • `"coxph"` – пропорциональные риски Кокса.
`data`	Фрейм данных, содержащий значения переменных, используемых в модели.
`weights`	Вектор весов объектов обучающей выборки. Не обязан быть нормированным, но все компоненты должны быть неотрицательными. Параметр не является обязательным.
`n.trees`	Количество итераций алгоритма градиентного бустинга.
`interaction.depth`	Дерево решений является функцией тех переменных, которые участвуют в формировании разбиения в каждой не листовой вершине. Данный параметр определяет максимальное число переменных, от которых может зависеть каждое дерево решений (тем самым ограничивая его высоту) в модели градиентного бустинга.
`n.minobsinnode`	В каждый лист каждого обученного дерева должно попасть число объектов обучающей выборки не меньше, чем `n.minobsinnode`. Данный параметр ограничивает размер одиночного дерева решений.
`shrinkage`	Параметр регуляризации ν в Алгоритме 1.
`bag.fraction`	Вещественное число от 0 до 1, определяющее долю объектов обучающей выборки, используемых на каждой итерации алгоритма. Таким образом, если объем обучающей выборки равен N, то для обучения каждого одиночного дерева решений будет случайным образом выбрано [N*`bag.fraction`] объектов.
`keep.data`	Определяет, будет ли построенная модель хранить данные, использованные при обучении. Данный параметр является существенным при дальнейшем использовании функции `gbm.more`.
`verbose`	Булева переменная, отвечающая за вывод дополнительной

	информации о ходе обучения модели.

```
gbm.more(object, n.new.trees = 100, data = NULL, weights = NULL,
offset = NULL, verbose = NULL)
```

Позволяет сделать дополнительные итерации алгоритма градиентного бустинга, дополнив существующую модель.

Входные параметры:

`object`	Модель градиентного бустинга, полученная с помощью функции `gbm`.
`n.new.trees`	Количество деревьев, которые необходимо добавить в модель.
`data`	Обучающая выборка. Если модель `object` была натренированна функцией `gbm` с параметром `keep.data=TRUE`, то данный параметр может быть опущен.
`weights`	Вектор весов объектов обучающей выборки. Если модель `object` была натренированна функцией `gbm` с параметром `keep.data=TRUE`, то данный параметр может быть опущен.
`verbose`	Булева переменная, отвечающая за вывод дополнительной информации о ходе обучения модели. Если `verbose=NULL`, то принимается значение, использованное при обучении модели `object`.

1.2.2. Функция predict.gbm

```
predict(object, newdata, n.trees, type = "link", single.tree =
FALSE, ...)
```

Осуществляет предсказание на объектах тестовой выборки.

Входные параметры:

`object`	Модель градиентного бустинга, полученная с помощью функции `gbm`.
`newdata`	Тестовая выборка.
`n.trees`	Количество деревьев, начиная с первого, либо список деревьев (см. `single.tree`), которые будут использованы для предсказания.

type	Определяет формат, в котором будут выданы предсказания. Если значение type="link", то на выходе функции будет вектор из значений $\{h_M(x_i)\}_{i=1}^N$, в случае type="response" будет проведена предобработка результата. Данный параметр является существенным только функций потерь "bernoulli" и "poisson".
single.tree	Если single.tree=TRUE, то для осуществления предсказания будут использованы только деревья с номерами n.trees, указанными в виде списка. Если же single.tree=FALSE, то для предсказание будет осуществлено с помощью n.trees первых деревьев модели градиентного бустинга.

Для задач восстановления регрессии на выходе функции будет непосредственно вектор с предсказанными значениями, в то время как для задач классификации формат вывода зависит от функции потерь и значения параметра type. Для того, чтобы получить предсказанные метки классов необходимо для штрафа "adaboost" взять знак предсказанного значения (отрицателный выход соответствует классу 0, положительный – классу 1), для штрафа "bernoulli" – интерпретировать предсказания, как вероятность принадлежности классу 0.

1.2.3. Функция summary

```
summary(object, cBars = length(object$var.names), n.trees = object$n.trees, plotit = TRUE, order = TRUE, method = relative.influence, normalize = TRUE, ...)
```

Вычисляет относительную значимость переменных в модели градиентного бустинга.

Входные параметры:

object	Модель градиентного бустинга, полученная с помощью функции gbm.
cBars	Количество переменных значимость которых будет отображена. По умолчанию будет выведена информация обо всех переменных.
n.trees	Количество деревьев в модели object, начиная с первого, которые будут использованы.
plotit	Определяет, будет ли выведено графическое представление результата.

`order`	Если `order=FALSE`, то будет вычислена значимость первых (по порядку) `cBars` переменных, иначе будет выведена информация о самых значимых переменных.

1.3. Задания для лабораторной работы.

4. Напишите генератор выборки соответсвующей следующему закону:

$$f(x) = 10 \cdot \prod_{j=1}^{3} e^{-2x_j^2} + \sum_{j=4}^{10} x_j,$$

где $x \in U[0,1]^{50}$, т.е. точки равномерно распределенные в пятидесятимерном гиперкубе.

5. Пользуясь генератором из задания 0, создайте обучающую и тестовую выборки объемом 1000 и 500 объектов соответственно. Добавьте к значениям функции $f(x)$ из обучающей выборки случайный шум, удовлетворяющий *нормальному закону распределения* с нулевым мат. ожиданием и дисперсией в два раза меньшей, чем оценка дисперсии значений функции *f(x)* на обучающей выборке, т.е. $\frac{\mathrm{var}(\{f(x_i)\}_{i=1}^{N})}{2}$. Постройте гистограмму для значений *шума* с большим числом отрезков разбиения. Обучите модель градиентного бустинга деревьев решений с помощью реализации из библиотеки gbm с использованием следующих значений параметров: `n.trees = 1000`, `interaction.depth = 3`, `shrinkage = 0.25`, `bag.fraction = 1.0` и двух различных функций потерь: `"laplace"` и `"gaussian"`. Постройте график зависимости ошибки на тестовой выборке[3] от количества итераций алгоритма градиентного бустинга. Последовательно переберите следующие значения параметра `shrinkage`: 0.1, 0.05, 0.02, и посмотрите, как изменятся построенные графики. Примите параметр `bag.fraction` равным 0.65 и оцените изменения зависимости тестовой ошибки от числа деревьев в модели. Сделайте вывод о влиянии рассмотренных параметров на качество построенной модели. Какая функция потерь в данном случае позволила получить лучшую модель?

6. Пользуясь генератором из задания 0, создайте обучающую и тестовую выборки объемом 1000 и 500 объектов соответственно. Добавьте к значениям функции $f(x)$ из обучающей

[3] Под тестовой ошибкой понимается отношение средней абсолютной ошибки, полученной с помощью построенной модели, к аналогичному показателю оптимальной константной модели $err = \sum_{i=1}^{Ntest} \left| y_i^{test} - \hat{f}(x_i^{test}) \right| / \sum_{i=1}^{Ntest} |y_i^{test} - \overline{y^{train}}|$, где $\hat{f}$ – построенная модель, $\overline{y^{train}}$ – среднее арифметическое значений целевой переменной на обучающей выборке.

выборки шум $s \cdot n/u$, где $n \in N(0,1)$, $u \in U(0,1)$, $s = \sqrt{\frac{\mathrm{var}(\{f(x_i)\}_{i=1}^{N})}{3 \cdot \mathrm{var}(\{n/u\}_{i=1}^{N})}}$. Постройте гистограмму для значений *шума* с большим числом отрезков разбиения. В чем отличия данного шума от нормального? Постройте модель градиентного бустинга деревьев решений с помощью реализации из библиотеки gbm с использованием следующих значений параметров: `n.trees = 1000`, `interaction.depth = 3`, `shrinkage = 0.02`, `bag.fraction = 0.65` и двух различных функций потерь: `"laplace"` и `"gaussian"`. Постройте график зависимости ошибки на тестовой выборке от количества итераций алгоритма градиентного бустинга. Сравните результаты, полученные с помощью различных функций потерь. Оцените значимость переменных с помощью всей построенной модели.

7. Загрузите данные из файла «task3_train.data» и обучите на них модель градиентного бустинга деревьев решений с формулой `y~.` для решения задачи бинарной классификации. Используйте следующие параметры для обучения: `n.trees = 800`, `interaction.depth = 3`, `shrinkage = 1.0`, `bag.fraction = 1.0` и двух различных функций потерь: `"bernoulli"` и `"adaboost"`. Загрузите тестовую выборку из файла «task3_test.data». Отобразите графики зависимости тестовой ошибки (доли неправильно классифицированных объектов к объему тестовой выборки) от числа итераций алгоритма. Сравните полученные модели.