

МАШИННОЕ ОБУЧЕНИЕ  
ЛАБОРАТОРНЫЙ ПРАКТИКУМ  
(предварительный вариант)

Н.Ю. Золотых, А.Н. Половинкин

25 ноября 2007 г.

# 1. Практическое машинное обучение

## 1.1. Бустинг

*Описание:*

```
adaboost.M1(formula, data, boos = TRUE, mfinal = 100, coeflearn = 'Breiman',  
minsplit = 5, cp = 0.01, maxdepth = nlevels(vardep))
```

*Аргументы:*

`formula` — символьное описание модели;  
`data` — набор данных, в котором интерпретируются переменные из символьного описания модели;  
`boos` — если TRUE (по умолчанию), то из тренировочного множества генерируется bootstrap набор данных, используя веса прецедентов на данной итерации.  
`mfinal` — число итераций в алгоритме бустинга (число деревьев решений, которое необходимо построить). По умолчанию число итераций `mfinal=100`;  
`coeflearn` — если 'Breiman' (по умолчанию), то используется  $\alpha=1/2\ln((1-\text{err})/\text{err})$ . Если Freund, то используется  $\alpha=\ln((1-\text{err})/\text{err})$ , где  $\alpha$  — коэффициент обновления весов прецедентов.  
`minsplit` — минимальное число прецедентов, которое должно быть в узле, чтобы к нему можно было применить разбиение (сплит);  
`cp` — параметр сложности. Разбиение, которое не уменьшает the overall lack of fit на величину `cp`, не применяется;  
`maxdepth` — максимальная глубина любого узла в последнем дереве (глубина корневого узла считается равной 0). По умолчанию равно числу классов.

*Детали:* `Adaboost.M1` представляет собой простое обобщение `Adaboost` на случай более чем 2 классов.

*Возвращаемое значение:* Объект класса `adaboost.M1`, который является списком, состоящим из следующих компонент:

`formula` — использованное символьное описание модели;  
`trees` — набор деревьев, используемых при построении модели;  
`weights` — вектор весов деревьев на всех итерациях;  
`votes` — матрица, описывающая для каждого прецедента число деревьев (вес каждого дерева учитывается с учетом коэффициента `alpha`), «проголосовавших» за принадлежность прецедента тому или иному классу;  
`class` — класс, предсказанный для каждого прецедента ансамблем классификаторов;  
`importance` — относительная «важность» каждой переменной в задаче классификации (сколько раз каждая переменная выбиралась для построения сплита).

**Описание:**

```
bagging(formula, data, mfinal = 100, minsplit = 5, cp = 0.01,  
        maxdepth = nlevels(vardep))
```

**Аргументы:**

`formula` — символьное описание модели;  
`data` — набор данных, в котором интерпретируются переменные из символьного описания модели;  
`mfinal` — число итераций в алгоритме бэггинга (число деревьев решений, которое необходимо построить). По умолчанию число итераций `mfinal=100`;  
`minsplit` — минимальное число прецедентов, которое должно быть в узле, чтобы к нему можно было применить разбиение (сплит);  
`cp` — параметр сложности. Разбиение, которое не уменьшает the overall lack of fit на величину `cp`, не применяется;  
`maxdepth` — максимальная глубина любого узла в последнем дереве (глубина корневого узла считается равной 0). По умолчанию равно числу классов.

**Возвращаемое значение:** Объект класса `bagging`, который является списком, состоящим из следующих компонент:

`formula` — использованное символьное описание модели;  
`trees` — набор деревьев, используемых при построении модели;  
`votes` — матрица, описывающая для каждого прецедента число деревьев, «проголосовавших» за принадлежность прецедента тому или иному классу;  
`class` — класс, предсказанный для каждого прецедента ансамблем классификаторов;  
`samples` — бутстрап наборы данных (для всех итераций).  
`importance` — относительная «важность» каждой переменной в задаче классификации (сколько раз каждая переменная выбиралась для построения сплита).

**Пример:**

**Пример:** Загрузим библиотеку для работы с деревьями решений (`rpart`) и библиотеку, содержащую наборы данных (`mlbench`):

```
> library(rpart)  
> library(mlbench)  
Загрузим набор данных Vehicle:  
> data(Vehicle)
```

Сгенерируем вектор номеров прецедентов, которые войдут в обучающую выборку (их число равно  $\frac{2}{3}l$ , где  $l$  — число прецедентов в наборе данных `Vehicle`)

```
> l <- length(Vehicle[,1])  
> sub <- sample(1:l, 2*l/3)
```

Зададим максимальное число итераций в алгоритме `adaboost.M1`, равным 25:

```
> mfinal <- 25
```

Зададим максимальную глубину каждого дерева решений в алгоритме `adaboost.M1`, равным 5:

```

> maxdepth <- 5

```

Построим одиночное дерево решений на подмножестве прецедентов набора данных `Vehicle` с номерами из `sub`:

```

> Vehicle.rpart <- rpart(Class~.,data=Vehicle[sub,],maxdepth=maxdepth)

```

Используя построенную модель, предскажем ответы на прецедентах из набора данных `Vehicle` с номерами, не вошедшими в `sub`:

```

> Vehicle.rpart.pred <- predict(Vehicle.rpart,newdata=Vehicle[-sub, ],type="class")

```

Вычислим ошибку предсказания:

```

> tb <- table(Vehicle.rpart.pred,Vehicle$Class[-sub])
> error.rpart <- 1-(sum(diag(tb))/sum(tb))

```

Построим ансамбль деревьев решений на подмножестве прецедентов набора данных `Vehicle` с номерами из `sub`, используя `adaboost.M1`:

```

> Vehicle.adaboost <-adaboost.M1(Class ~.,data=Vehicle[sub,],
  mfinal=mfinal, maxdepth=maxdepth)

```

Используя построенную модель, предскажем ответы на прецедентах из набора данных `Vehicle` с номерами, не вошедшими в `sub`:

```

> Vehicle.adaboost.pred <- predict.boosting(Vehicle.adaboost,
  newdata=Vehicle[-sub, ])

```

Построим ансамбль деревьев решений на подмножестве прецедентов набора данных `Vehicle` с номерами из `sub`, используя `bagging`:

```

> Vehicle.bagging <- bagging(Class ~.,data=Vehicle[sub,],
  mfinal=50, maxdepth=5)

```

Используя построенную модель, предскажем ответы на прецедентах из набора данных `Vehicle` с номерами, не вошедшими в `sub`:

```

> Vehicle.bagging.pred <- predict.bagging(Vehicle.bagging,
  newdata=Vehicle[-sub, ]) Vehicle.bagging.pred[-1]

```

Выведем тестовые ошибки, полученные при использовании одиночного дерева и ансамбля деревьев решений, построенных `adaboost.M1` и `bagging`:

```

> error.rpart
> Vehicle.adaboost.pred$error
> Vehicle.bagging.pred$error

```

### 1.1.1. Задания для лабораторной работы

- 1) Исследуйте зависимость тестовой ошибки от количества деревьев в ансамбле для алгоритма `adaboost.M1` на наборе данных `Vehicle` из пакета `mlbench` (обучающая выборка должна состоять из  $\frac{7}{10}$  всех прецедентов, содержащихся в данном наборе данных). Постройте график зависимости тестовой ошибки при числе деревьев, равном 1, 11, 21, ..., 301, объясните полученные результаты.
- 2) Исследуйте зависимость тестовой ошибки от количества деревьев в ансамбле для алгоритма `bagging` на наборе данных `Glass` из пакета `mlbench` (обучающая выборка должна состоять из  $\frac{7}{10}$  всех прецедентов, содержащихся в данном наборе данных). Постройте график зависимости тестовой ошибки при числе деревьев, равном 1, 11, 21, ..., 201, объясните полученные результаты.

- 3) Реализуйте бустинг алгоритм с 3 классификаторами (ССЫЛКА). Сравните тестовую ошибку, полученную с использованием данного классификатора на наборах данных `Vehicle` и `Glass`, с тестовой ошибкой, полученной с использованием единичного дерева классификации.