Московский Государственный Университет имени М.В. Ломоносова
Факультет Вычислительной Математики и Кибернетики
Кафедра Математических Методов Прогнозирования

Краткий обзор ключевых возможностей пакета “gbm” системы R

Фонарев Александр
317 группа

Апрель 2012

Общее описание пакета “gbm” и проделанной работы

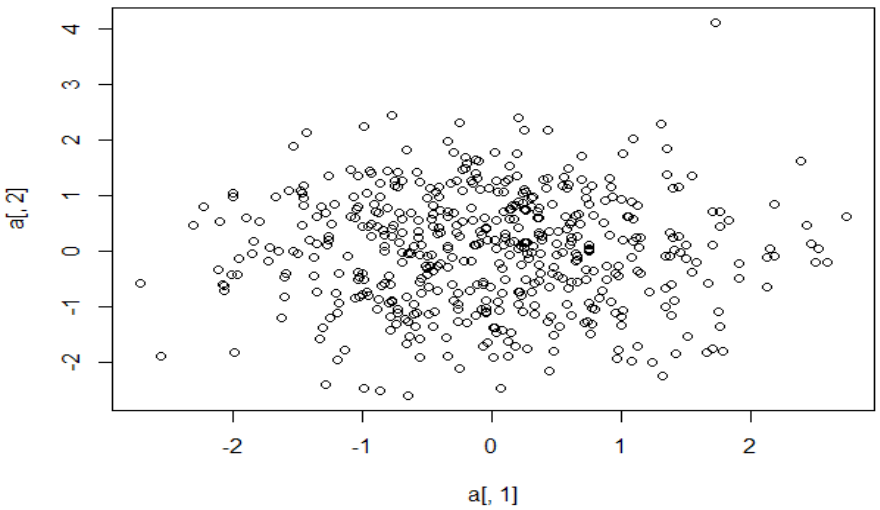
Полное название данного пакета – «Generalized Boosted Regression Models». Он включает в себя реализацию различных алгоритмов бустинга для решения задачи восстановления регрессии. Возможно использование различных функций потерь и других параметров.

Мною будут рассмотрены лишь несколько примеров, демострирующие основные возможности данного пакета. Далее будет рассмотрен только один вариант с построением композиции решающих деревьев. Для более детального изучения рекомендуется использовать официальное руководство к пакету (см. раздел «Использованные источники»).

Генерация данных

Для начала сгенерируем модельные данные, на которых мы будем изучать пакет. Для этого создадим 500 точек из нормального распределения со стандартными параметрами и сохраним их координаты в переменную a. Далее для каждой точки посчитаем длину ее радиус-вектора и сохраним это значение в переменную b. Будем пробовать восстановить зависимость длины радиус-вектора от координат точки.

Код	Результат

`a = matrix(rnorm(1000), 500, 2)`
`plot(a[,1],a[,2])`
`b = sqrt(a[,1]^2 + a[,2]^2)`
	

Создание модели

Пакет gbm предполагает обучение модели и сохранение ее в некоторый объект. Создадим объект-модель с названием ourmodel:

Код

```
ourmodel <- gbm(

# формула, показывающая, что ищется зависимость поля Y от полей X1 и X2 в соответствующем фрейме

Y ~ X1 + X2,

# создание фрейма из матрицы с данными

data = data.frame(X1 = a[,1], X2 = a[,2], Y = b),

# выбор функции потерь

distribution = "gaussian",

# масмимальное количество используемых решающих деревьев

n.trees = 2000,

# другие параметры алгоритма

shrinkage = 0.005,
interaction.depth=3,

# параметры валидации

bag.fraction = 0.5,
train.fraction = 0.5,
n.minobsinnode = 10,
cv.folds = 5,

# параметр, показывающий, надо ли сохранять копию данных

keep.data = TRUE,

# параметр, показывающий, нужна ли печать в консоль результатов процесса настройки модели

verbose=TRUE

)
```

Результат

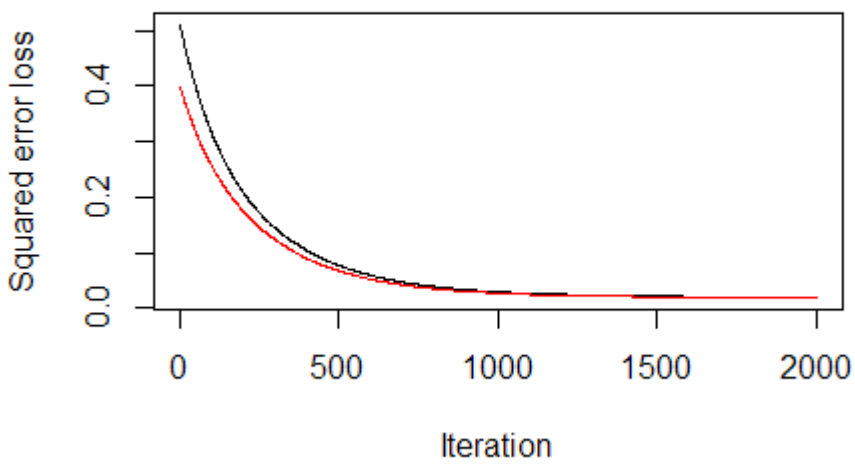
```
Сначала идет вывод результатов для каждого из 5 этапов кросс-валидации отдельно. А затем общие результаты:

Iter   TrainDeviance   ValidDeviance   StepSize   Improve
     1        0.5102          0.3978     0.0050    0.0021
     2        0.5074          0.3960     0.0050    0.0025
     3        0.5048          0.3942     0.0050    0.0022
     4        0.5022          0.3925     0.0050    0.0022
     5        0.4996          0.3906     0.0050    0.0026
     6        0.4971          0.3889     0.0050    0.0024
     7        0.4941          0.3868     0.0050    0.0025
     8        0.4913          0.3848     0.0050    0.0023
     9        0.4888          0.3829     0.0050    0.0025
    10        0.4864          0.3814     0.0050    0.0022
   100        0.3158          0.2569     0.0050    0.0013
   200        0.2078          0.1742     0.0050    0.0007
   300        0.1441          0.1233     0.0050    0.0005
   400        0.1039          0.0893     0.0050    0.0002
   500        0.0770          0.0672     0.0050    0.0002
   600        0.0593          0.0522     0.0050    0.0001
   700        0.0472          0.0422     0.0050    0.0001
   800        0.0390          0.0351     0.0050    0.0000
   900        0.0334          0.0304     0.0050    0.0000
  1000        0.0296          0.0273     0.0050    0.0000
  1100        0.0269          0.0248     0.0050    0.0000
  1200        0.0251          0.0232     0.0050   -0.0000
  1300        0.0236          0.0221     0.0050   -0.0000
  1400        0.0226          0.0214     0.0050    0.0000
  1500        0.0218          0.0207     0.0050   -0.0000
  1600        0.0210          0.0203     0.0050   -0.0000
  1700        0.0204          0.0199     0.0050   -0.0000
  1800        0.0199          0.0197     0.0050   -0.0000
  1900        0.0194          0.0194     0.0050   -0.0000
  2000        0.0190          0.0191     0.0050   -0.0000
```

Это происходит с помощью вызова функции gbm(). Она содержит множество параметров, некоторые из которых даже опущены в этом примере. В качестве вывода можно видеть результат на каждой из 5 итераций кроссвалидации и его зависимость от количества деревьев, а также общий результат по обучению модели. Видно, что при увеличении числа деревьев начиная примерно с 1200 результат перестает улучшаться.

Оптимальное количество алгоритмов

Посмотрим по-внимательнее на оптимальное количество деревьев после настройки модели:

Код	Результат

`# оптимальное число деревьев при оценке «out-of-bag»`
`# также возможны варианты «test» и «cv»`
 	
`best.iter <- gbm.perf(ourmodel,method="OOB")`
`print(best.iter)`
	
	1263

Красным показана ошибка на контроле в зависимости от количества деревьев, черным показана ошибка на обучении. Далее выводится оптимальное число деревьев в соответствии с данным методом проверки обучающей способности: 1263.

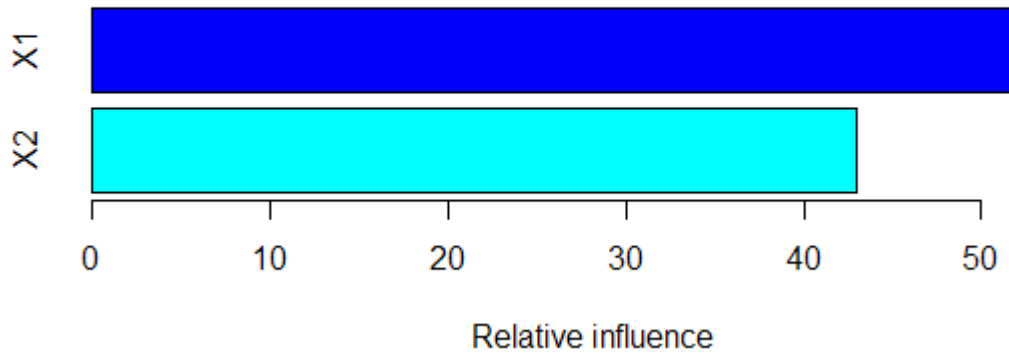
Информация о каждом дереве в отдельности

Также в пакете присутствуют функции для просмотра информации и анализа деревьев и наборов деревьев по-отдельности:

Код

```
# также можно посмотреть на «влияние» каждого из двух признаков на предсказанный ответ при фиксированном числе деревьев

summary(ourmodel, n.trees = 1543)


# можно вывести фрейм с информацией о внутреннем строении дерева

print(pretty.gbm.tree(ourmodel, 1543))
```

Результат

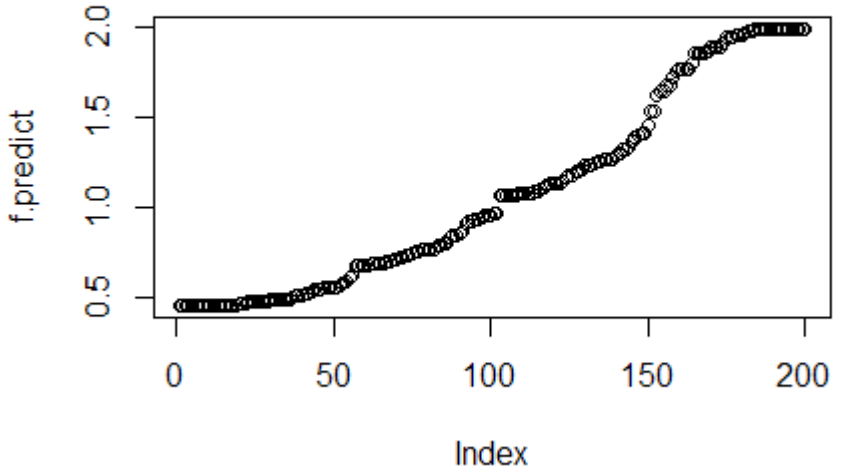



```
   var  rel.inf
1  X1 57.06188
2  X2 42.93812

SplitVar SplitCodePred LeftNode RightNode MissingNode ErrorReduction Weight    Prediction
0        1 -1.741156e+00        1         2           9     0.04995455    125  3.364079e-05
1       -1  3.726031e-04       -1        -1          -1     0.00000000     10  3.726031e-04
2        1  5.936047e-01        3         7           8     0.06429663    115  4.165803e-06
3        0 -2.090986e-01        4         5           6     0.06494628     82 -7.083488e-05
4       -1  1.096515e-04       -1        -1          -1     0.00000000     31  1.096515e-04
5       -1 -1.805423e-04       -1        -1          -1     0.00000000     51 -1.805423e-04
6       -1 -7.083488e-05       -1        -1          -1     0.00000000     82 -7.083488e-05
7       -1  1.905311e-04       -1        -1          -1     0.00000000     33  1.905311e-04
8       -1  4.165803e-06       -1        -1          -1     0.00000000    115  4.165803e-06
9       -1  3.364079e-05       -1        -1          -1     0.00000000    125  3.364079e-05
```

Использование настроенной модели для предсказания

Сгенерируем набор точек на оси абсцисс с шагом 0.01 (их радиус-вектор просто равен абсциссе) и попробуем предсказать радиус-вектора для точек этого набора:

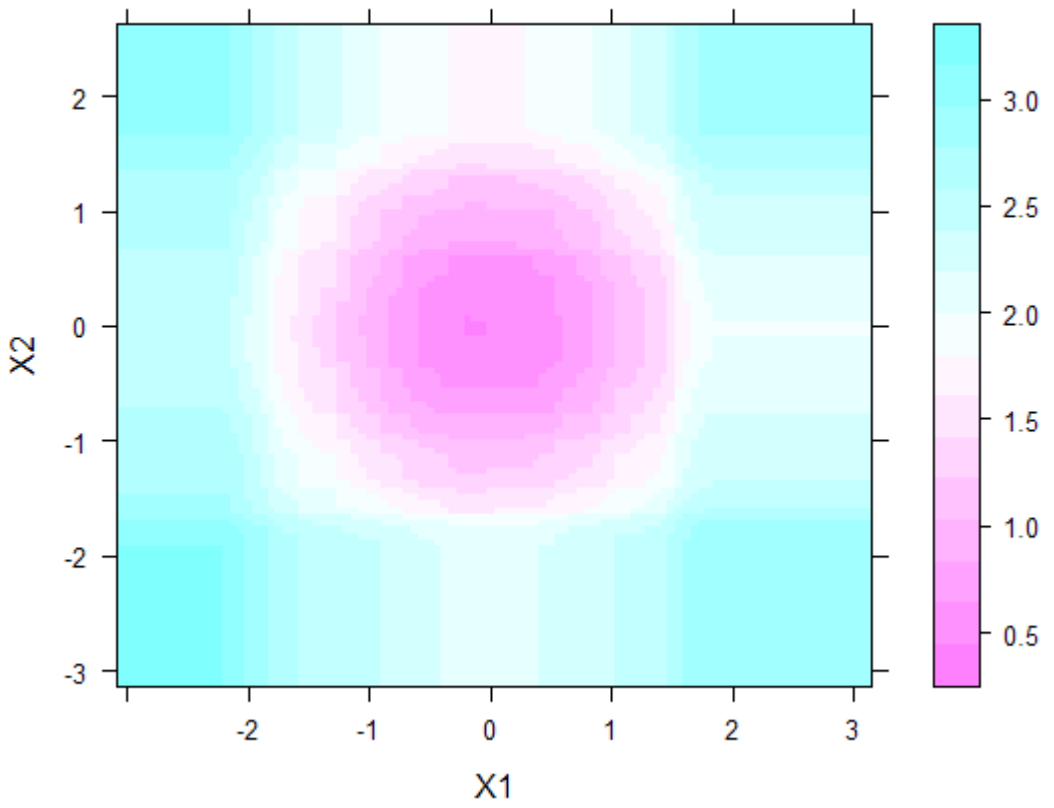
Код	Результат

`# генерируем данные`
 	
`a2 = cbind(1:200 / 100, 0)`
 	
`# предсказываем, используя модель и оптимальное количество алгоритмов`
 	
`f.predict <- predict.gbm(ourmodel, data.frame(X1 = a2[,1], X2 = a2[,2], Y = a2[,1]), best.iter)`
 	
`# изобразим наш предсказанный результат`
`plot(f.predict)`
	

Идеальный результат – линейная зависимость на графике. Видно, что нашему алгоритму ее найти не удалось. Однако все же он уже нашел достаточно неплохую закономерность в данных.

Визуализация карты восстановленной регрессии

Код	Результат

`# визуализируем то, как восстановилась регрессия`
`# в качестве координат используем признаки с первого по второй`
 	
`plot.gbm(ourmodel, 1:2, best.iter)`
	

Использованные источники

- Официальный сайт пакета – http://cran.gis-lab.info/web/packages/gbm/index.html
- Руководство к пакету “gbm” – http://cran.gis-lab.info/web/packages/gbm/gbm.pdf
- Методическое пособие А.Г. Дьяконова по системе R – http://alexanderdyakonov.narod.ru/upR.pdf
- Курс лекций К.В. Воронцова – http://www.machinelearning.ru/wiki/index.php?title=Машинное_обучение_%28курс_лекций%2C_К.В.Воронцов%29