Примеры работы с пакетом "LiblineaR"в системе R

Малышева Екатерина

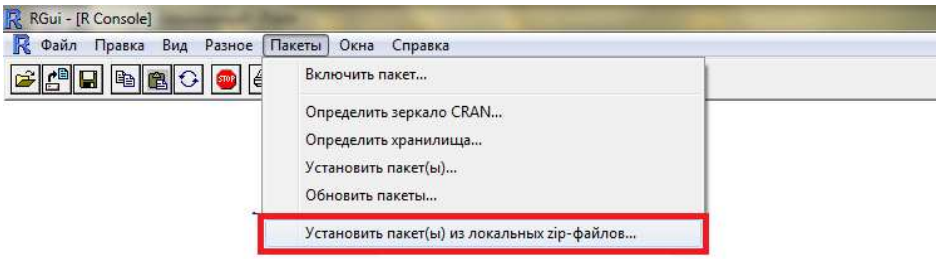
18 апреля 2012 г.

Краткое описание пакета "LiblineaR"

LiblineaR (интерфейс liblinear C/C++) – библиотека для машинного обучения линейного классификатора. Данный пакет поддерживает настройку линейного классификатора с помощью SVM или логистической регрессии. Для подбора оптимальных параметров в нем доступна кросс-валидация. Также возможен возврат вероятностных оценок для логистической регрессии и многое другое. Пакет достаточно популярен, его часто используют для решения реальных задач, он быстро работает и прост в использовании.

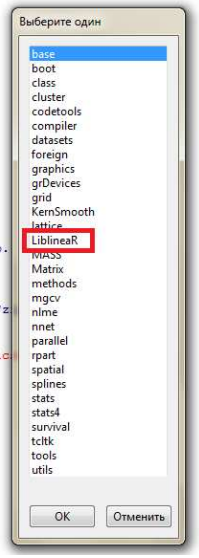
Работа в системе R. Установка пакета

1. Установите систему R – http://cran.gis-lab.info
2. Скачайте LiblineaR – http://cran.gis-lab.info/web/packages/LiblineaR/index.html
3. В системе R откройте вкладку пакеты



4. Выберите скаченный zip-файл, пакет будет установлен

5. Теперь необходимо подключить пакет к системе. Для этого во вкладке "Пакеты"выбираем пункт "Включить пакет"и в появившемся окне выбираем "LiblineaR"



6. Теперь система готова для работы с LiblineaR.

Краткая документация

1. heuristicC(data) - оценивает (эвристически) константу C для дальнейшего использования в алгоритме SVM (И только для него). Константа C - параметр настройки метода, который позволяет регулировать отношение между максимизацией ширины разделяющей полосы и минимизацией суммарной ошибки. Константа находится по формуле $c = \frac{1}{\frac{1}{n}\Sigma_{i=1}^{n}\sqrt{G[i,i]}}$, где $G = data * data'$

   Вход:

   data - матрица $N * P$, где N – количество объектов-точек, P – количество признаков-координат.

   Выход:

   Эвристическая оценка константы C

2. LiblineaR(data, labels, type, cost, epsilon, bias, wi, cross, verbose) - эта функция позволяет оценить предсказательную способность алгоритма линейной классификации. Также возвращает обученную модель.

   Вход:

   data - матрица $N * P$, где N – количество объектов-точек, P – количество признаков-координат.

   labels - вектор ответов длины N

   type - один из 8 вариантов:

   0 – L2-регуляризация для логистической регрессии

   1 – L2-регуляризация, L2 потери для SVM (dual)

   2 – L2-регуляризация, L2 потери для SVM (primal)

   3 – L2-регуляризация, L1 потери для SVM (dual)

   4 – Мульти-классовый SVM (Crammer and Singer)

   5 – L1-регуляризация L2 потери для SVM

   6 – L1-регуляризация логистическая регрессия

   7 – L2-регуляризация логистическая регрессия (dual)

   cost - константа C - параметр настройки метода, который позволяет регулировать отношение между максимизацией ширины разделяющей полосы и минимизацией суммарной ошибки при выборе SVM алгоритма. При выборе логистической регрессии константа C - штраф за нарушение при обучении.

   epsilon - "точность"настройки коэффициентов. Если type = 0, 2, 5, 6 то по умолчанию epsilon=0.01. Если type = 1, 3, 4, 7 то по умолчанию epsilon=0.1

   bias - если bias = TRUE, то data превращается в [data; 1]

   wi - вектор весов для различных классов, если количество объектов в классах разное. Не обязательно все веса должны быть указаны (по умолчанию вес 1). Все компоненты должны быть названы в соответствии с необходимой меткой класса.

   cross - если cross > 0, то проводится кросс-валидация для настройки параметров (с разбиением на cross блоков). Будьте бдительны! Если данные разных классов имеют различный объем (отличаются на порядок и более), то она бессмысленна. По умолчанию значение 0.

verbose - если значение TRUE, то информация о классификации будет выведена на экран.

Выход:

Обученная модель.

3. predict(object, newx, proba, decisionValues) - классифицирует по обученной модели объекты.

   Вход:

   object - обученная модель

   newx - классифицируемый объект

   proba - для логистической регрессии при установке этого параметра значения TRUE функция будет возвращать еще и вероятность принадлежности к классу.

   decisionValues - если значение TRUE, то будет вычислена и возвращена оценка принадлежности классу.

   Выход: Класс, к которому принадлежит объект, или вероятность принадлежности к классу.

Пример №1

Для начала разберем простейшую модельную задачу. Матрица признакового описания будет состоять из 1000 объектов и 7 признаков. Если сумма признаков умноженная на $(-1)^k$ (k – номер столбца признака в матрице)> 0, то относим к классу TRUE, иначе к классу FALSE. Произведем оценку константы C с помощью функции heuristicC и посмотрим на качество работы классификатора. Получили 99.2 процента.

```
> M = matrix(rnorm(7000), nrow = 1000, ncol = 7);
> Y = (M[,1]  - M[,2] + M[,3] - M[,4] + M[,5] - M[,6] + M[,7] > 0);
> costm = heuristicC(M);
> result = LiblineaR(M, Y, 1, costm, epsilon = 0.01, bias = TRUE, cross = 15, verbose = FALSE);
> result
[1] 0.992
>
```

Пример №2

Теперь рассмотрим реальную задачу. Данные были взяты из репозитория (http://archive.ics.uci.edu/ml/datasets/MAGIC+Gamma+Telescope). Это

данные по регистрации гамма частиц с высокой энергией (в атмосфере с телескопа). Размер данных – 19020 объектов на 10 признаков + класс. Всего 2 класса – 'g' - гамма-частица, 'h' – фон. Первоначальные данные - это изображения, снятые с телескопа. Изображение - это фон и некоторые пятна на нем. Нужно распознать, является ли пятно гамма излучением или это шум. Далее область с пятнами преобразуется в 10 признаков: площадь пятна, значение главной и мнимой оси у пятна, если рассматривать его как эллипс и другие геометрические характеристики.

1. Считываем данные, приводим их тип к необходимому виду

```
> a <- read.table("magic04.data", sep = "," ); #чтение данных из файла
> trainingData = a[, 1:10]; # выделение в отдельную матрицу признаков
> ansVec = a[,11]; # Вектор ответов
> trainingData = as.matrix(trainingData); # Установка типа данных матрица
> trainingData = as.numeric(trainingData); # Установка типа данных число, а не символ
> dim(trainingData)
NULL
> length(trainingData)
[1] 190200
> trainingData = matrix(trainingData, nrow = 19020, ncol = 10);
> #Теперь с точки зрения типов R у нас числовая матрица, с которой можно работать
>
```

2. Разбиваем на обучающую и контрольную выборки, подбираем константу C.

```
> A = rbind(M[1:2000,], M[17000:19000,]) # уменьшили объем данных
> Y = c(y[1:2000], y[17000:19000]) #уменьшили объем данных
> train = sample(1:4001, 3600) #создаем случайную выборку из объектов
> aTrain = A[train,]
> aTest = A[-train,]
> yTrain = Y[train]
> yTest = Y[-train]
> #разбили множество на обучающую выборку и контроль
> #нормируем данные
> s = scale(aTrain, center = TRUE, scale = TRUE)
```

3. Выбираем логистическую регрессию (t = 0) и смотрим на точность работы классификатора для различных C.

```
> t = 0
> try = c(1000, 100, 10, 1, 0.1, 0.01, 0.001);
> res = c();
> for (co in try){
+ acc = LiblineaR(data = s, labels = yTrain, type =t, cost = co, bias = 1, cross = 10, verbose = FALSE)
+ res = c(res,acc);
+ cat("Results for C =", co, ":", acc, "accuracy. \n", sep ="");
+ }
Results for C =1000:0.7752778accuracy.
Results for C =100:0.7755556accuracy.
Results for C =10:0.7758333accuracy.
Results for C =1:0.7747222accuracy.
Results for C =0.1:0.7744444accuracy.
Results for C =0.01:0.7725accuracy.
Results for C =0.001:0.7625accuracy.
> 
```

4. Выбрали наилучшую C. Получили модель классификации, нормировали и центрировали контрольные данные так же, как те, на которых производилось обучение. Вывели таблицу принадлежности классу, вывели точность работы классификатора.

```
> best = which.max(res)
> co = try[best]
> m = LiblineaR(data = s, labels = yTrain, type = t, cost = co, bias = 1, verbose = FALSE)
> s2 = scale(aTest, attr(s, "scaled:center"), attr(s, "scaled:scale"))
> p = predict(m, s2, proba = TRUE)
> res = table(p$predictions, yTest)
> print(res)
   yTest
      1   2
  1 173  55
  2  30 143
> BCR = mean(c(res[1,1]/sum(res[,1]),res[2,2]/sum(res[,2])))
> print(BCR)
[1] 0.7872195
> 
```