Московский Государственный Университет имени М.В. Ломоносова

Факультет Вычислительной Математики и Кибернетики

Кафедра Математических Методов Прогнозирования

Задание по практикуму на ЭВМ:

«Работа с пакетом “nnet” в системе R»

Выполнила студентка 317 группы Любимцева Мария

2012 год

Начало работы

Пакет nnet позволяет настраивать и использовать нейронные сети с одним скрытым слоем и multinomial log-linear модели.

Перед началом работы необходимо включить пакет:
Пакеты -> Включить пакет… -> nnet

Для демонстрации взята задача классификации пациентов на предмет наличия у них диабета: 1 - нет диабета, 2 - предиабет, 3 - диабет.

```
> #загрузка данных

> M<-read.table("diabet_for_R.txt", header = TRUE, sep = ",")
```

Функции пакета

class.ind(factor_class)

Описание

На вход подается фактор, полученный из столбца, отвечающего за принадлежность объектов к классам.

На выходе получается матрица размера *m*k* (где *m* – количество объектов, *k* – количество классов). Ее *ij*-й элемент принимает значение 1, если *i*-ый объект принадлежит *j*-ому классу, и 0 в противном случае.

Если воспользоваться терминологией, введенной на АМА, то данная функция строит матрицу, состоящую из характеристических векторов классов эквивалентности.

Примеры использования

```
> M[c(1,90,120),] #возьмем объекты из разных классов
    rel_weight glufast glutest instest sspg clinical_group
1         0.81      80     356     124   55              1
90        1.18     108     486     297  220              2
120       0.92     303    1364      42  346              3

> class.ind(factor(M[c(1,90,120),'clinical_group']))
     Chem_Diabetic Normal Overt_Diabetic
[1,]             0      1              0
[2,]             1      0              0
[3,]             0      0              1
```

which.is.max(x)

Описание

Возвращает индекс элемента, случайно выбранного из максимальных элементов вектора *x*.

Примеры использования

```
> which.is.max(c(1, 0, 1, 1, 0, 1))

[1] 3
> which.is.max(c(1, 0, 1, 1, 0, 1))
[1] 6
> which.is.max(c(1, 0, 1, 1, 0, 1))
[1] 4
> which.is.max(c(1, 0, 1, 1, 0, 1))
[1] 1
```

multinom

Описание

Настройка multinomial log-linear модели, используя нейронную сеть.

Примеры использования

```
> multinom(formula = factor(M1[,ncol(M1)])~.,
data = M1[,-ncol(M1)], subset = ind_tr)
# weights:  21 (12 variable)
initial  value 119.748739
iter  10 value 30.620889
iter  20 value 10.496096
iter  30 value 4.719632
iter  40 value 2.812935
iter  50 value 2.405701
iter  60 value 2.335009
iter  70 value 2.283478
iter  80 value 2.162779
iter  90 value 2.109086
iter 100 value 2.050354
final  value 2.050354
stopped after 100 iterations
Call:
multinom(formula = factor(M1[, ncol(M1)]) ~ ., data = M1[, -
ncol(M1)], subset = ind_tr)

Coefficients:
  (Intercept) rel_weight   glufast  glutest   instest      sspg
2   -46.31949  -1.047263 -67.55795 237.2528  2.110525  3.241692
3  -121.99472 -28.191629 180.33115 259.5070 10.029133 13.002593

Residual Deviance: 4.100708
AIC: 28.10071
```

nnet

Описание

Настройка нейронной сети с одним скрытым слоем. Возможно два варианта использования.

Параметры:

x	Матрица обучающей выборки.
y	Матрица ответов для обучающей выборки (должна быть преобразована с использованием class.ind).
weights	Веса объектов, полагаются единицами в случае пропуска.
size	Число нейронов скрытого слоя.
formula	Формула вида `class ~ x1 + x2 + ...`
data	Data frame – выборка.
subset	Вектор индексов объектов, на которых следует провести обучение.
na.action	Параметр, определяющий, что делать с NA-значениями. По умолчанию – окончание процедуры.
Wts	Вектор начальных значений параметров. В случае пропуска задается случайно.
mask	Логический вектор, определяющий, какие из параметров необходимо оптимизировать. По умолчанию все.
linout	Переключение на linear output units. По умолчанию logistic output units.
entropy	Переключатель для энтропии.
softmax	Переключатель для softmax (log-linear model) и maximum conditional likelihood fitting. `linout`, `entropy`, `softmax` и `censored` являются взаимно исключающими.
censored	Вариант `softmax`, в котором ненулевые метки означают возможность принадлежности к данному классу. Таким образом для `softmax` вектор `(0, 1, 1)` означает, что объект

	принадлежит обоим классам 2 и 3, но для `censored` он значит, что объект принадлежит или классу 2, или классу 3.
skip	Переключение добавления связей между входом и выходом.
rang	Случайные начальные значения весов берутся из диапазона [-`rang`, `rang`].
decay	Параметр для weight decay (сокращение весов). По умолчанию 0.
maxit	Число максимального количества итераций. По умолчанию 100.
Hess	Если значение TRUE, то гессиан меры настройки на найденном лучшем наборе возвращается через компоненту `Hessian`.
trace	Включает tracing optimization. По умолчанию `TRUE`.
MaxNWts	Максимально допустимое количество весов.
abstol	Остановка, если значение критерия при настройке опускается ниже `abstol`.
reltol	Остановка, если оптимизатор не в состоянии уменьшить критерий настройки по крайней мере на `1 - reltol`.

predict

Описание

Классификация объекта с помощью настроенной сети

Примеры использования обеих функций.

```
> TestNet = function(M, num_u){
+
+ #выборка случайным образом делится в соотношении 3:1
+ ind = sample(1:nrow(M), nrow(M), replace = F);
+
+ train = M[ind[-(1:round(0.25*nrow(M)))], -ncol(M)];
+ y_tr = M[ind[-(1:round(0.25*nrow(M)))], ncol(M)];
+ y_tr_ind = class.ind(factor(y_tr));
+
+ test = M[ind[1:round(0.25*nrow(M))],-ncol(M)];
+ y_t = M[ind[1:round(0.25*nrow(M))], ncol(M)];
+ y_t_ind = class.ind(factor(y_t));
```

```
+
+ #обучение сети на большей части
+ My_net = nnet(train, y_tr_ind, size = num_u);
+ #тестирование на меньшей
+ ans = round(predict(My_net, test));
+
+ #возвращает количество ошибок на тестовой части, размер
+ которой в данном случае - 36 объектов
+ return (Num_Er(ans, y_t_ind))
+ }
```

Применение TestNet в таком виде может давать неплохой результат

```
> TestNet(M,50)

# weights:  453
initial  value 84.127948
iter  10 value 58.377427
iter  20 value 48.408303
iter  30 value 48.087930
iter  40 value 43.206240
iter  50 value 42.713290
iter  60 value 42.506362
iter  70 value 42.469849
iter  80 value 42.445111
final  value 42.445063
converged
[1] 6
```

Однако, может случиться следующая неприятность:

```
> TestNet(M,50)
# weights:  453
initial  value 190.010556
final  value 109.000000
converged
[1] 36 #т.е. алгоритм ошибся на всех объектах тестовой выборки
```

В TestNet изменим параметры настройки сети:

```
> My_net = nnet(train, y_tr_ind, size = num_u, maxit = 1000,
entropy = TRUE, decay = 5e-4);
```

(По результатам экспериментов наилучшего результата удалось добиться именно при entropy = TRUE).

Функция помощи в нахождении оптимального значения параметра size – числа нейронов в скрытом слое – при помощи метода скользящего контроля. Выдает матрицу размера *mnum_u*cv_it*, где *ij*-ому элементу соответствует количество ошибок на тестовой выборке при size = *i**10 на *j*-ой итерации :

```
> NetCV = function(M){
+
+ mnum_u = 10; #максимальное число нейронов/10
+ cv_it = 10;  #число итераций для каждого значения size
+ ans = matrix(numeric(mnum_u*cv_it), nrow=mnum_u, ncol=cv_it);
+
+ for (i in 1:mnum_u){
+ for (j in 1:cv_it){
+ ans[i,j] = TestNet(M,i*10);
+ }
+ }
+ return(ans);
+ }
```

Результат использования и наглядная демонстрация необходимости нормировки данных:

```
> NetCV(M) #без нормировки
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    5   36    8    2    0    7   17   21   17     3
 [2,]    5    1   23   18   11    7   15    5    5    17
 [3,]    7    8    5   10    6    4    5    8    9     2
 [4,]    7    7    8    8    9    4    6    6    6     8
 [5,]    6    3    7    4    8   10    8    5    7     7
 [6,]    8    6    7    3    5    7    7    6    6     6
 [7,]    6    7    6    7    7    6    6    6    8    11
 [8,]    7   10    8   10    3    5    7    4    9     8
 [9,]    2    4    8    4    5   10    7    7    7     4
[10,]    7    1    5   11    8    6    8    8   10     4
```

```
> #предварительно пронормируем матрицу по максимуму столбца
> max_M = numeric(6)
> max_M[6] = 1
> for (i in 1:5) max_M[i] = max(M[,i])
> h = rep(max_M,nrow(M))
> dim(h) = c(ncol(M),nrow(M))
> M1 = M / t(h)
>
> NetCV(M1)
```

```
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    1    1    5    5    4    3    1    3    6     2
 [2,]    3    2    3    3    1    0    3    3    3     4
 [3,]    2    3    3    3    1    2    3    4    4     0
 [4,]    1    3    2    2    4    3    2    4    2     4
 [5,]    1    1    2    3    4    2    4    3    3     1
 [6,]    2    3    5    4    3    2    3    4    3     4
 [7,]    1    3    4    3    2    2    1    3    1     4
 [8,]    2    2    2    1    2    4    4    2    4     1
 [9,]    2    1    2    3    4    3    3    3    1     3
[10,]    4    4    2    4    1    3    3    2    3     6
```

Остановимся на size = 80, так как при этом значении достигается наименьшее среднее количество ошибок по итерациям = 2.4 и наименьшая медиана = 2.0.

Продемонстрируем второй вариант использования функции nnet:

```
> TestNet2 = function(M){
+
+ #разделение выборки на обучение и контроль
+ ind = sample(1:nrow(M), nrow(M), replace = F);
+ ind_tr = ind[-(1:round(0.25*nrow(M)))];
+
+ #в отличие от первого варианта, на вход подается вся выборка и
+ параметром subset указывается, какая ее часть используется при
+ обучении
+ My_net = nnet(factor(M[,"clinical_group"])~., data = M[,-
ncol(M)], subset = ind_tr, size = 80, maxit = 1000, entropy =
TRUE, decay = 5e-4);
+
+ #тестирование на контроле
+ ans = predict(My_net, M[-ind_tr,-ncol(M)], type = "class");
+
+ #возвращает количество ошибок на контроле
+ return (sum(ans != M[-ind_tr, "clinical_group"]))
+ }

> NetCV(M1)
> #10 раз прогнали TestNet2
> #результат количество ошибок на каждой итерации
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    2    3    3    2    3    1    1    1    2     2
```

Список литературы:

1) http://alexanderdyakonov.narod.ru/upR.pdf

2) http://cran.gis-lab.info/web/packages/nnet/nnet.pdf

3) http://www.faqs.org/faqs/ai-faq/neural-nets/

4) http://r-analytics.blogspot.com/