36-350: Data Mining

Lab 10

Date: November 1, 2002 **Due: end of lab**

---

1 Introduction

This lab teaches you how to mine data for interactions between predictors of a response.

There are 4 questions. For each one, submit your commands and a response from R demonstrating that they work. (Only hand in commands relevant to the question.) To submit a plot, click on the plot window and select

```
File -> Save as -> Postscript...
```

This saves the plot to a file which can be printed, incorporated into a Word document, or mailed to us as an attachment.

2 Starting R

Start R as in lab 1. On the class web page, go to "computer labs" and download the files for lab 10 into your work folder. Read the special functions into your running R application via the commands

```
source("lab10.r")
```

If this fails, check that the files were downloaded correctly.

3 The data

The dataset used in this lab is 196 weeks of grocery sales, similar to that used in class, but for a different store. The variables are:

Price.1	DOLE PINEAPPLE ORANG 64 OZ
Price.2	FIVE ALIVE CTRUS BEV 64 OZ
Price.3	HH FRUIT PUNCH 64 OZ
Price.4	HH ORANGE JUICE 64 OZ
Price.5	MIN MAID O J CALCIUM 64 OZ
Price.6	MIN MAID O J PLASTIC 96 OZ
Price.7	MM PULP FREE OJ 64 OZ
Price.8	SUNNY DELIGHT FLA CI 64 OZ
Price.9	TREE FRESH O J REG 64 OZ
Price.10	TROP PURE PRM HOMEST 64 OZ
Price.11	TROP SB HOMESTYLE OJ 64 OZ
Sold.4	Number of units sold for HH ORANGE JUICE 64 OZ

Your job is to determine how the price variables interact in predicting `Sold.4`.
Load this data via

```
load("lab9.rda")
```

This defines a matrix called `x`.

4 Standardizing

The response variable `Sold.4` should be transformed with a logarithm. Do this now.
Also, as you may have noticed from lab 9, `Price.5` and `Price.7` are highly correlated:

```
plot(x[,"Price.5"],x[,"Price.7"])
```

Since these products always have essentially the same price, let's combine them into one average predictor. With this change, the prices are relatively uncorrelated, which helps in searching for interactions and interpreting the regression coefficients.

> **Question 1:** Submit code to construct a new predictor `Price.5.7` which is the average of `Price.5` and `Price.7`. The code should look like
>
> ```
> x[,"Price.5.7"] = ???
> ```
>
> where you have to fill in `???`.

Use the following code to remove the old predictors:

```
x <- x[,setdiff(colnames(x),c("Price.5","Price.7"))]
```

Standardize the variables to have zero mean and unit variance. `sx` should end up with the standardized data.

5 Linear regression to search for interactions

In lab 9, you constructed a linear model by adding predictors one by one. When searching for interactions, it is a better idea to remove predictors (or predictor combinations) one by one using the `step` function. First you construct a model with all predictors included:

```
fit <- lm(Sold.4 ~ .,sx)
```

The formula `Sold.4 . ` is shorthand for `Sold.4 Price.1 + Price.2 + ....` Next you want to try expanding the model by including bilinear cross terms such as `Price.1:Price.2`, which is simply the product of `Price.1` and `Price.2`. These are also called 'interaction terms'. The function `expand.cross` helps you do this:

```
expand.cross(fit)
```

it returns a formula containing all possible cross terms. Give this formula to `step`, and it will automatically add and remove predictors, returning a new model:

```
fit <- step(fit,expand.cross(fit))
summary(fit)
```

This model should have five interaction terms, which is a large reduction over the number of possible interaction terms. The interactions which have been selected by `step` are good candidates for further investigation.

The interaction terms can simply be viewed as additional predictors, and plotted using `predict.plot`. A partial residual plot will show the importance of each predictor in the model, including the interaction terms:

```
predict.plot(fit,partial=T)
```

> **Question 2:** (a) Which interaction term has the greatest weight in the model? Which has the least weight? Does the plot agree? (b) One of the interaction terms has greater importance than either of the individual predictors involved. Which is it? (You can use the plot or the weights in the model to decide this.)

6 Interaction plots

The command `interact.plot` works similarly to `predict.plot`, except it plots the predictors in pairs instead of one by one. For each pair, it shows a contour plot against the response, the residuals of a model, or the partial residuals of a model. For example, this shows partial residuals:

```
interact.plot(fit,partial=T,lwd=2,nlev=6)
```

The partial residuals for each square are computed using a smaller model which uses all predictors except those two. Thus each square tells you what those two predictors contribute to predicting the response, with the effect of the other predictors already taken into account (as if they were held fixed). You should be able to see, for example, that `Price.4` dominates `Price.2` in its effect on `Sold.4`, when other prices are held fixed. The interactions which are currently in the model are outlined in red.

> **Question 3:** (a) Turn in the interaction plot described above and keep a copy for the homework. (b) Find the 'strongest' and 'weakest' interactions that you identified in question 2. Does the `interact.plot` agree with that ranking? (Remember that a 'weak' interaction has contours nearly straight, and a 'strong' interaction has contours very curved.)

7 Slice plots

Now that you've located the interactions, you'll want to explain them. Some of the interactions can be understood directly from the contour plot. Others are easier to understand from looking at slice plots. Let's focus on one, the interaction between `Price.4` and `Price.5.7`. The first step is to get the data that was plotted in that square:

```
r <- partial.residual.frame(fit,"Price.4*Price.5.7")
```

This sets `r` to a matrix with three columns: `Price.4`, `Price.5.7`, and `Sold.4` (`Sold.4` is the partial residuals for that square). Notice that you put `*` between the variable names, not `:`.

This repeats the contour plot for that square:

```
color.plot(loess(formula(r),r),nlevels=8,lwd=2)
```

To make a slice plot, use `predict.plot` as follows:

```
predict.plot(Sold.4 ~ Price.5.7 | Price.4, r, nlevels=2)
```

This slices the data based on `Price.5.7`, and plots `Sold.4` versus `Price.4` in each slice. The `nlevels` number is the number of slices you want (default is 2). Generally 2 is sufficient but for very strong interactions you may want to try nlevels=3.

In this plot, you should find that `Price.4` controls the importance of `Price.5.7`. `Price.5.7` is only important when `Price.4` is low. This suggests that a sale on products 5 and 7 will compete with product 4 if product 4 is also on sale, but will have no effect on product 4 if product 4 is at its regular price (which is 0 in standardized units).

Usually it is best to slice on the stronger predictor. You can determine which predictor is stronger by looking at their individual weights in the linear model. Here `Price.4` was stronger. This shows what happens if you slice on the wrong one:

```
predict.plot(Sold.4 ~ Price.4 | Price.5.7, r, nlevels=2)
```

The slope of `Sold.4` versus `Price.4` is the same in both slices, so it looks as though there is no interaction between `Price.5.7` and `Price.4`, which we know is false. This asymmetry in slice plots makes them somewhat tricky to use, compared to contour plots which display both predictors symmetrically.

> **Question 4:** The linear model should have 4 interaction terms besides the `Price.4:Price.5.7` term analyzed above. For each one, make a slice plot in which you slice on the stronger predictor. Turn in the plots and keep copies for the homework. (You do not actually need these plots to complete the homework, but you will probably find them helpful. If you do use them, be sure to include them with your homework solutions. You may also want to make additional slice plots, beyond what are required here, to aid you in doing the homework.)