36-350: Data Mining

Lab 2
Date: September 5, 2003 **Due: end of lab**

Interspersed throughout this lab are questions that you will have to answer at check-off.

1. Download the files for this lab from the course web page to the desktop:

   ```
   http://www.stat.cmu.edu/~minka/courses/36-350/lab/
   ```

2. Open a Word or Notepad document to record your work.

Start R

3. `Start -> All Programs -> Class software -> R 1.7.0`

4. Load the special functions for this lab:

   ```
   File -> Source R code...
   ```

   Browse to the desktop and pick `lab2.r` (it may have been renamed to `lab2.r.txt` when you downloaded it). Another window will immediately pop up for you to pick the `mining.zip` file you downloaded.

The image dataset

5. The dataset is in a table called `imgs`, containing 20 images of "Action Sailing" and 20 images of "Auto Racing". They have already been converted into color counts over 64 prototypical colors in the HSV cube. A vector of labels named `img.labels` is also defined. *How many pixels in* `sailing5` *have color* `thistle`*?*

6. Using the commands from lab 1, compute a matrix of distances between the images. For example:

   ```
   imgs = remove.singletons(imgs)
   imgs = idf.weight(imgs)
   x = div.by.euc.length(imgs)
   d = distances(x)
   ```

Nearest-neighbor classification

7. For this lab, let's pretend that the images `sailing1`–`sailing10` and `racing11`–`racing20` just arrived, and we don't know what class any of them belongs to. These are the "test" images. The other images are "training" images. Using the commands described below, create a variable called `i.test` which holds the row numbers of the test images, and a variable called `i.train` which holds the row numbers of the training images. If you do this right, `imgs[i.test,]` should be the test images and `imgs[i.train,]` should be the training images.

8. Using the index vectors and the function `closest`, determine the training image which is the nearest neighbor of each test image. *Out of the 20 test images, which would be misclassified by the nearest-neighbor method?*

Nearest-prototype classification

9. Using the training images and the labels in `img.labels`, compute prototypes for sailing and racing:

   ```
   xp = prototypes(imgs[i.train,],img.labels[i.train],sum)
   ```

10. Concatenate the prototypes with the rows for test images, to get a new data matrix, and compute all distances. Note that IDF weights need to be assigned *before* taking prototypes, and division by Euclidean length needs to be done *after* taking prototypes. *Out of the 20 test images, how many are misclassified, and which are they? Which is the better classification method here?*

11. You can now get checked off.

Constructing vectors There are two basic ways to create a vector:

```
1:10
c(1,2,3,4,5,6,7,8,9,10)
```

The first way creates a vector of all integers from a starting value to an ending value. The second way uses the function `c` to concatenate a bunch of numbers together. You can also use it to concatenate vectors, such as

```
c(1:10,15:20)
```

The result can be assigned to a variable:

```
x = 5:10
x[3]    # answer is 7
```

Selecting a block from a matrix You know that you can use numbers and names to select from a matrix, as in `imgs["racing4",4]`. But you can also use vectors of numbers or names, to select an entire block of the matrix. For example, `imgs[,1:4]` is the first 4 columns of the matrix, `imgs[c(2,4),]` is rows 2 and 4 of the matrix, and `imgs[c(2,4),1:4]` is the first 4 columns of rows 2 and 4.

Computing nearest-neighbors If `d` is a distance matrix, then

```
closest(d[i.train,i.test])
```

will tell you which training image is closest to each of the test images. In general, `closest` takes a sub-block of a distance matrix, and returns the closest row for each column.

Concatenating matrices To concatenate the rows of matrix `x` with the rows of matrix `y`, type:

```
z = rbind(x,y)
```

This defines a new matrix `z` containing all rows.