36-350: Data Mining

Lab 4
Date: September 18, 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 `lab4.r` (it may have been renamed to `lab4.r.txt` when you downloaded it). Another window will immediately pop up for you to pick the `mining.zip` file you downloaded.

The dataset

5. The dataset is the same as lab 2. There is a matrix called `imgs`, containing 20 images of "Action Sailing" and 20 images of "Auto Racing". There is also a vector of labels named `img.labels`. Using the commands from lab 2, prepare the images for Euclidean distance. For example:

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

Clustering

6. Partition the images into 2 groups using K-means. *Have any images been put into the wrong cluster? If so, which ones?*

7. Compute a hierarchy using Ward's method. Plot the merging costs and save it for the homework.

8. Cut the hierarchy to get 2 groups, just like K-means. *Have any images been put into the wrong cluster? If so, which ones? Does k-means or does Ward's method do better at clustering the images into sailing versus racing?*

9. *Which clustering is better according to the sum of squares value? How does it compare to the sum of squares of the correct labeling?*

Color image segmentation

10. `img` contains the “hand” image shown in class, which you can view as follows:

    ```
    plot.image(img)
    ```

    The matrix `img.rgb` has the pixels as rows and the three color channels as columns.

11. Use K-means to partition the image into 3 parts. The resulting cluster numbers can be viewed as an image, as follows:

    ```
    plot.image(array(f, dim(img)))
    ```

    This image should have three different colors, corresponding to the clusters. Pixels in the same cluster will have the same color in this plot.

12. You can now get checked off.

K-means clustering If `x` is a matrix and `k` is the desired number of clusters, then

```
f = kmeans(x,k)$cluster
```

makes `f` a vector of cluster numbers, one number for each row of `x`. Notice the result is not unique. `f` can be used just like `img.labels`. A quick way to see what images are in which cluster is:

```
split(f,rownames(x))
```

Ward’s method If `x` is a matrix, then

```
d = distances(x)
hc = hclust(as.dist(d^2),method="ward")
```

makes `hc` a hierarchy. Notice you don’t specify `k`, and the distances have to be squared on input. The hierarchy can be plotted with

```
plot(hc)
```

The merging costs can be plotted with

```
plot.hclust.trace(hc)
```

A partition of size `k` can be obtained by cutting the tree:

```
f = cutree(hc,k)
```

This makes `f` a vector of cluster numbers.

Sum of squares The sum of squares value for any partition `f` can be computed via

```
sum.of.squares(x,f)
```