36-350: Data Mining

Lab 2
Date: September 6, 2002 **Due: end of lab**

---

1 Introduction

This lab teaches you the steps involved in classifying images based on similarity. You will compare the two methods taught in class: nearest-neighbor vs. nearest-prototype.
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.) Open a `Notepad` file in which to save your work (`Notepad` is better than `Word` for computer output). Don't forget to check that the response makes sense!

2 Starting R

Start R as in lab 1. (You may find it useful to keep lab 1 handy for doing this lab.) On the class web page, go to "computer labs" and download the files for lab 2 into your work folder. Read the special functions into your running R application via the commands

```
source("lab1.r")
source("lab2.r")
```

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

3 The data

The images to classify are described by the data file `lab2.rda`. There are 20 images of "Action Sailing" and 20 images of "Auto Racing". They have already been converted into color counts over 64 prototypical colors in HSV space. Read the data in with

```
load("lab2.rda")
```

This will define a matrix of color counts named `imgs` and a vector of labels named `img.labels`. You can examine them with

```
dim(imgs)
str(imgs)
```

The function `str` is like `dim`, but gives a bit more information. You can get help on most functions by preceding them with a question mark, as in

```
?str
```

You can also search via the Help menu.

> **Question 1:** How many pixels in `sailing5` have color "6" (or a color most similar to color "6")?

4 Distances

The first thing you want to do is measure the distances between all images. Recall these functions from lab 1. Remove infrequent colors:

```
imgs <- remove.singletons(imgs)
```

Weight the colors by inverse picture frequency:

```
imgs <- idf.weight(imgs)
```

Divide each image's count vector by its Euclidean length:

```
x <- div.by.euc.length(imgs)
```

Compute a distance matrix:

```
d <- distances(x)
```

5 Nearest-neighbor classification

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 remaining images are "training" images, which define what we know about the two classes so far. In this section, you will use nearest-neighbor to automatically classify the test images, based on the training images.

> **Question 2:** When you index by a vector, you extract multiple values. For example, `imgs[c(1:3),]` and `imgs[c("sailing1","sailing2","sailing3"),]` extract the first three rows of `imgs`. Define two index vectors, `i.test` and `i.train`, so that `imgs[i.test,]` are the test images and `imgs[i.train,]` are the training images. (There is more than one way to do this.)

There is a function called `closest`, which will find the minimum value in each column of a (partial) distance matrix. This only makes sense when the rows and columns are different, e.g.

```
closest(d[1:3,4:6])
```

will tell you which of images 1, 2, and 3 is closest to each of the images 4, 5, and 6. In this case, you're extracting a 3 by 3 sub-matrix of `d`. Take a look at `d[1:3,4:6]` to make sure you understand how `closest` works.

> **Question 3:** (a) Using the full distance matrix computed earlier, the index vectors, and the function `closest`, determine the training image which is the nearest-neighbor of each test image, and the resulting classifications of the test images. Out of the 20 test images, how many are misclassified, and which are they?
>
> (b) Repeat part (a) *without* using IDF weights on the colors.

6 Nearest-prototype classification

In this method, you compute prototypes to describe the training data, and then measure distance to them. Using the index vectors in the previous section, and the labels in `img.labels`, the following code will compute prototypes:

```
xp <- as.matrix(prototypes(x[i.train,],img.labels[i.train],sum))
x2 <- rbind(xp, x[i.test,])
```

The matrix `x2` now contains prototypes for `sailing` and `racing`, as well as the test images. Note that IDF weights need to be assigned *before* taking prototypes, and division by Euclidean length needs to be done *after* taking prototypes.

> **Question 4:** (a) Compute a distance matrix between the prototypes and test images, and use `closest` to determine which prototype is closest to each test image. Out of the 20 test images, how many are misclassified, and which are they?
>
> (b) Repeat part (a) *without* using IDF weights on the colors.