Homework 8: Optimizing On Ice

36-350

31 October 2014

Agenda: Optimizing functional alignments.

In the last lab, you practiced "registration" of a one-dimensional signal to make sure that it aligned with what was expected. In this homework, we'll expand this to a two-dimensional setting.

In this assignment, you will practice how to register a signal by matching two point clusters as best you can in two dimensions. The first set, `true.signal`, is what we expect to appear in the data; the second, `distorted.signal` is what we observe in the real world. Your mission (and you must accept it) is to transform the signal so that it best matches the original.

Part 1: Getting the data ready

1. Load the file `hw-08.RData` from the course website, which contains the two variables. Note that these are both matrices with two columns. How many rows are in each variable? Display the summary statistics for both point clusters. What are the ranges of the data in x and y?
2. First, explore the two data sets with a simple scatterplot. Use

```
plot(your.first.matrix)
points(your.second.matrix, col="color.of.your.choice")
```

to place them on the same plot window. Make sure that your `xlim` and `ylim` ensure that all the data is visible with a comfortable margin.

3. Install the package `KernSmooth` from CRAN. (Comment this out in your R Markdown file.) This will let us make a two-dimensional kernel density estimate of each of the point clusters. Once this is done, choose one of your two objects, use the function

```
my.2d.kde <- bkde2D(your.matrix.of.choice)
```

and investigate the object returned. Try using these options to begin with: `bandwidth = 0.1`, `range.x = list(c(-20,20),c(-20,20))`, `gridsize=c(201,201)`. Plot this using the command

```
image (my.2d.kde$x1, my.2d.kde$x2, my.2d.kde$fhat)
```

Try changing the initial options for the KDE and seeing how they affect the plot.

4. Once you have decided on the parameters to use for your kernel density estimates, produce one for each of `true.signal` and `distorted.signal`. Check that the x and y coordinates (`x1` and `x2`) are the same for both estimates, and that the sums of the `fhat` vectors are identical (or close enough). Once this is done, plot the difference of the two kernel density estimates

```
image (my.2d.kde$x1, my.2d.kde$x2, my.2d.kde$fhat - my.second.2d.kde$fhat)
```

We want a white value to represent a difference of zero. Repeat the previous plot, but before doing so, calculate the maximum difference between the two densities

```
max.z.diff <- max(abs(my.2d.kde$fhat - my.second.2d.kde$fhat))
```

and add these options to the "image" command

```
zlim=max.z.diff*c(-1,1)
col=colorRampPalette(c("red","white","blue"))(21)
```

Compare the two plots. Do the regions in white correspond to differences of zero?

Finally, what is the sum of the difference in the densities?

Part 2: Building loss functions

5. Defining the loss function. Take the sum of squared differences between the densities as the loss to minimize. Calculate this for the data as is; repeat this calculation by adding 1 to every value of the `x` coordinate of the distorted set before calculating the density; repeat again by subtracting 1 from the `x` values. (Consider this a "shift" of 1 or -1 to the data.)

6. Write a function that takes as its input a single value, the shift to apply to `x`, and calculates the difference of the density plots of the shifted distorted set and the `true.signal` for the sum of squared differences. Test 5 values of your choice for the shift on this function. Which gives the best value?

7. Write a function that takes as its input a two-length vector, the shifts to apply to `x` and `y`, and calculates the difference of the density plots of the shifted distorted set and the `true.signal` for the sum of squared differences. Test 5 values of your choice for the shift on this function. Which gives the best value?

8. Now, use `nlm` or `optim` on your two-dimensional shift to find the ideal shift value. How much does your loss function change? Verify this by plotting the difference in the transformed `distorted.signal` data against the `true.signal`, using the same `zlim` as before. How much does the total difference appear to change?

Part 3: Advanced transforms

The shift of the data is the first part of what we call an affine transformation:

$$X_{new} = \vec{s} + X_{old} * A$$

A is a 2-by-2 transformation matrix of rotations and stretches; $\vec{s}$ is the shift you've found part of in the previous question. X_{old} is the n-by-2 data matrix.

9. Write a function that takes as its input a six-length vector. The first two are the shifts to apply to each coordinate `x` and `y`; the third through sixth are the elements in the A matrix. This function should output the new data matrix X_{new}. Verify that this function works correctly by taking the `distorted.signal` matrix, applying these transformations and plotting them:

a) `c(-10, -10, 1, 0, 0, 1)` – a simple translation of 10 units in each direction.

b) `c(0,0, -1, 0, 0, -1)` – a rotation of 180 degrees about the origin.

c) `c(0,0, 0, -1, 1, 0)` – a rotation of 90 degrees. (In which direction?)

10. Use either `nlm` or `optim` on this function to find the ideal value for the shift according to these functions. Use the default vector `c(0,0,1,0,0,1)` as your starting point. Confirm this by plotting the density of the transformed values minus the density of the target as in Question 8.

11. Repeat the previous step four more times. Choose a number of different starting vectors for your optimization routine, given your intuitions about the routine in the previous 2 questions. Find which one of these gives you the smallest `minimum` and plot the resulting difference as compared to the original. How much reduction are you able to achieve?