Stat4DS / Homework 02

(All-in-One)

Pierpaolo Brutti

Due week: from Monday, Jan 11 to Sunday, Jan 17 (on Moodle)

General Instructions

I expect you to upload your solutions on Moodle as a **single running** `R Markdown` file (`.rmd`) + its `html` output, named with your surnames.

You will give the commands to answer each question in its own code block, which will also produce plots that will be automatically embedded in the output file. Your responses must be supported by both textual explanations and the code you generate to produce your results. *Just examining your various objects in the "Environment" section of RStudio is insufficient – you must use scripted commands and functions.*

R Markdown Test

To be sure that everything is working fine, start `RStudio` and create an empty project called `HW1`. Now open a new `R Markdown` file (`File > New File > R Markdown...`); set the output to `HTML mode`, press `OK` and then click on `Knit HTML`. This should produce a web page with the knitting procedure executing the default code blocks. You can now start editing this file to produce your homework submission.

Please Notice

- For more info on `R Markdown`, check the support webpage that explains the main steps and ingredients: R Markdown from RStudio.
- For more info on how to write math formulas in LaTex: Wikibooks.
- Remember our **policy on collaboration**: *Collaboration on homework assignments with fellow students is **encouraged**. However, such collaboration should be clearly acknowledged, by listing the names of the students with whom you have had discussions concerning your solution. You may **not**, however, share written work or code after discussing a problem with others. The solutions should be written by **you**.*

Exercise: Whack a MoG

1. Background

Do you remember **The Bart Simpson density**? It is formally defined as the following very peculiar mixture of 6 Gaussians

$$p(x) = \frac{1}{2}\phi(x \,|\, 0, 1) + \frac{1}{10}\sum_{j=0}^{4} \phi\big(x \,|\, \tfrac{j}{2} - 1, \tfrac{1}{10}\big),$$

where $\phi(x \,|\, \mu, \sigma)$ denotes a Normal density with mean μ and standard deviation σ. Marron and Wand call this density "the claw" and over the years it has been extensively used as a benchmark model to test the performance of density estimators.

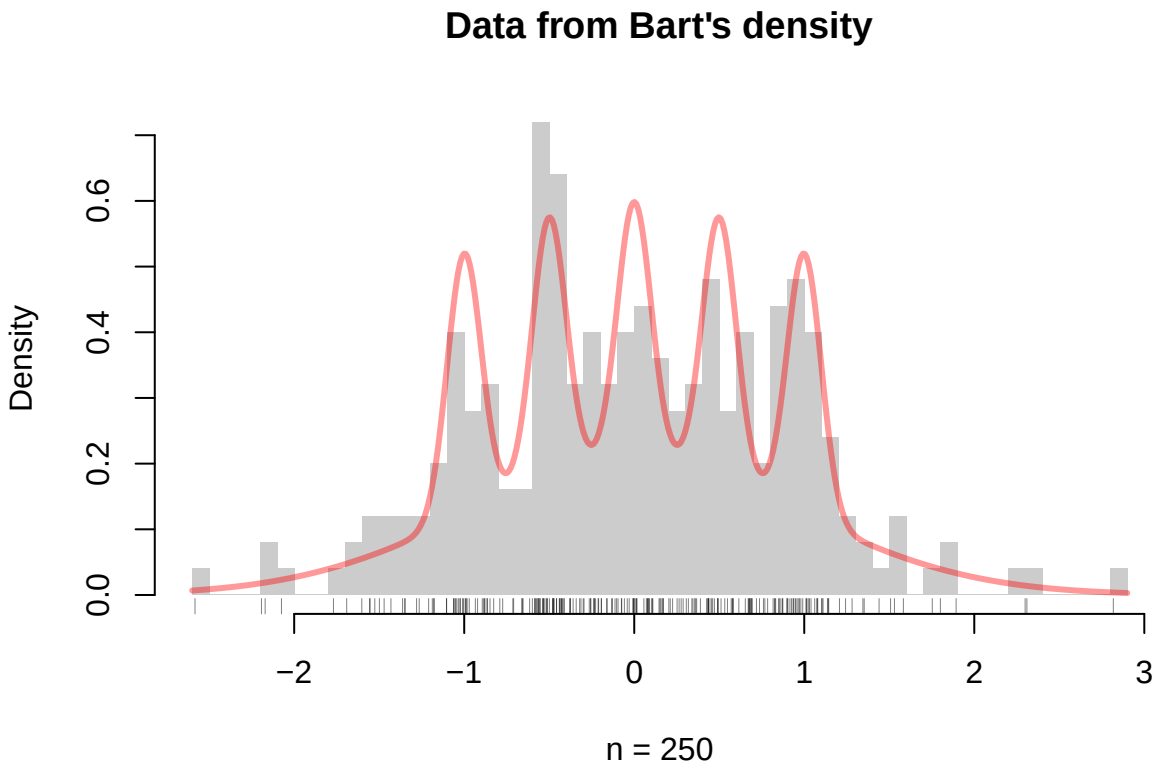
We personally met this beauty in a previous script. Let's recycle the code there to simulate from Bart

```
suppressMessages(require(mixtools, quietly = T)) # Package and function
?rnormmix

n  <- 250 # Sample size
XX <- rnormmix(n,
           lambda = c(0.5, rep(0.1,5)),
           mu     = c(0, ((0:4)/2)-1),
           sigma  = c(1, rep(0.1,5)) )
```

```r
# Make an histogram of the data
hist(XX, prob = T, col = gray(.8), border = NA,
     main = paste("Data from Bart's density",sep=""),
     xlab = paste("n = ", n, sep = ""),
     breaks = 50)
# Show the data points
rug(XX, col = rgb(0,0,0,.5))

# Plot the true density
true.den = function(x) 0.5*dnorm(x, 0, 1) +
                                0.1*dnorm(x,-1.0, 0.1) + 0.1*dnorm(x, -0.5, 0.1) +
                                0.1*dnorm(x, 0.0, 0.1) + 0.1*dnorm(x,  0.5, 0.1) +
                                0.1*dnorm(x, 1.0, 0.1)
curve(true.den, col = rgb(1,0,0,0.4), lwd = 3, n = 500, add = TRUE)
```



This is all nice and known, hence, to kick things up a notch, let's introduce a brand new loss to be placed side by side with our beloved classical choice, the Kullback-Liebler divergence: the **Wasserstein distance**.

Broadly speaking, if $\boldsymbol{X}$ is a random vector in $\mathbb{R}^m$ with distribution $\mathcal{P}_X$ and $\boldsymbol{Y}$ is another random vector in $\mathbb{R}^m$ with distribution $\mathcal{P}_Y$ then, for $p \geq 1$, the p–Wasserstein distance based on the ground distance d is defined as

$$W_{d,p}(\mathcal{P}_X, \mathcal{P}_Y) \equiv W_{d,p}(\boldsymbol{X}, \boldsymbol{Y}) = \left(\inf_{\mathcal{J}} \int_{\mathbb{R}^m \times \mathbb{R}^m} d(\mathbf{x}, \mathbf{y})^p \, d\mathcal{J}(\mathbf{x}, \mathbf{y}) \right)^{1/p},$$

where the infimum is over all joint distributions $\mathcal{J}$ having $\mathcal{P}_X$ and $\mathcal{P}_Y$ as given marginals... sounds complicated, does it? Anyway, this optimal transport techniques are extremely useful and widespread nowdays with tons of "*... truly marvelous Machine Learning applications, which this homework is too narrow to contain...*".

Nevertheless, when $m = 1$, that is for univariate distributions like *The Bart*, things simplify quite a bit and it can be shown that, picking the $\mathtt{L}_1$ metric (i.e. the *absolute distance*) as ground distance, we obtain

$$\mathrm{W}_{\mathtt{L}_1,p}(\mathcal{P}_X, \mathcal{P}_Y) = \left(\int_0^1 \left| \mathrm{F}_X^{-1}(z) - \mathrm{F}_Y^{-1}(z) \right|^p dz \right)^{1/p},$$

where $\mathrm{F}_X(\cdot)$ and $\mathrm{F}_Y(\cdot)$ are the cumulative distribution functions of $\mathcal{P}_X$ and $\mathcal{P}_Y$, respectively and, consequently, $\mathrm{F}_X^{-1}(\cdot)$ and $\mathrm{F}_Y^{-1}(\cdot)$ are essentially their associated **quantile functions**.

⇝ Your job ⇜

Starting with the idea of learning *The Bart* from data using a `MoG`, your grand job is to design a simulation study in order to compare the performance of different model selection techniques (more info in our notes).

More specifically:

1. Extend my handmade code to get the EM-fit of a generic mixture of $k \geqslant 1$ (instead of just 2) Gaussians.
2. Pick two different sample sizes, say $n_1 < n_2$ to probe a *clearly* non-asymptotic and a *reasonably* asymptotic regime.
3. For each of these sample sizes, simulate M times from *The Bart*, fit a $k \in \{1, 2, \ldots, k_{\texttt{max}}\}$ `MoG` with your handmade code and select the number of components k using the following 8 methods:
   (a) AIC;
   (b) BIC;
   (c) Sample Splitting with 50% in train and 50% in test;
   (d) Sample Splitting with 70% in train and 30% in test;
   (e) Sample Splitting with 30% in train and 70% in test;
   (f) 5-fold Cross-Validation;
   (g) 10-fold Cross-Validation;
   (h) As a last method, we consider a *very (VERY!) raw* Wasserstein based score inspired by $\mathrm{W}_{\mathrm{L}_1,p}$ with $p = 1$, that is

$$\mathrm{W}_k = \int_0^1 \left| \mathrm{F}_k^{-1}(z \,|\, \boldsymbol{\theta}) - \mathrm{F}_X^{-1}(z) \right| dz,$$

   where $\mathrm{F}_X^{-1}(\cdot)$ denotes the quantile function of the true population model, and $\mathrm{F}_k^{-1}(z \,|\, \boldsymbol{\theta})$ is the quantile function of a k components `MoG`.

   Since $\boldsymbol{\theta}$ has to be estimated and $\mathrm{F}_X^{-1}(\cdot)$ is unknown to us, to get a computable score $\widehat{W}_k$ for each $k \in \{1, 2, \ldots, k_{\texttt{max}}\}$, we adopt the canonical Sample Splitting scheme: **(1)** divide the sample in half; **(2)** on the $1^{\texttt{st}}$ half (the *training set*) calculate the MLE $\widehat{\boldsymbol{\theta}}_{\texttt{Tr}}$; **(3)** on the $2^{\texttt{nd}}$ half (the *test set*) calculate the ECDF $\widehat{F}_{\texttt{Te}}(\cdot)$ and its associated quantile function $\widehat{F}_{\texttt{Te}}^{-1}(\cdot)$; **(4)** finally get the following `Wass-score` by numerical integration

$$\widehat{\mathrm{W}}_k = \int_0^1 \left| \mathrm{F}_k^{-1}(z \,|\, \widehat{\boldsymbol{\theta}}_{\texttt{Tr}}) - \widehat{\mathrm{F}}_{\texttt{Te}}^{-1}(z) \right| dz,$$

4. Bearing in mind that, by picking $k_{\texttt{max}}$ larger that $k^\star = 6$ (the true number of components), we are actually working in a **well-specified** setup, summarize the performance of each method (under each sample size) with suitable numerical and graphical summaries (e.g. how far the selected values $\widehat{k}$ are from their target $k_{\texttt{max}}$), and **extensively** comment.

HINT A: help yourself and choose n_1, n_2, M and $k_{\texttt{max}}$ also taking into account your computational resources.

HINT B: you have to implement sample splitting and cross-validation from scratch, but, to create the folds, you may use some basic functionality from additional packages like `caret`. Take a look in particular at the createDataPartition() function and its relatives like `createResample()` and `createFolds()`.
