StatLearn / Homework 02

Pierpaolo Brutti

Due Sunday, June 17, 2018, 23:00 PM on Moodle

General Instructions

I expect you to upload your solutions on Moodle as a **single running** `R Markdown` file (`.rmd`), named with your surnames. In case you decide to work in Python or other programming language, you still have to provide a **running**, **sharable** notebook with detailed instructions on how to use it, **plus** a doc with the final result.

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**.*

Density Estimation Classification & Naïve Bayes

Introduction

The problem of predicting a **discrete** random variable Y from a random vector $\mathbf{X}$ is called **classification**, *supervised learning*, *discrimination*, or *pattern recognition*.

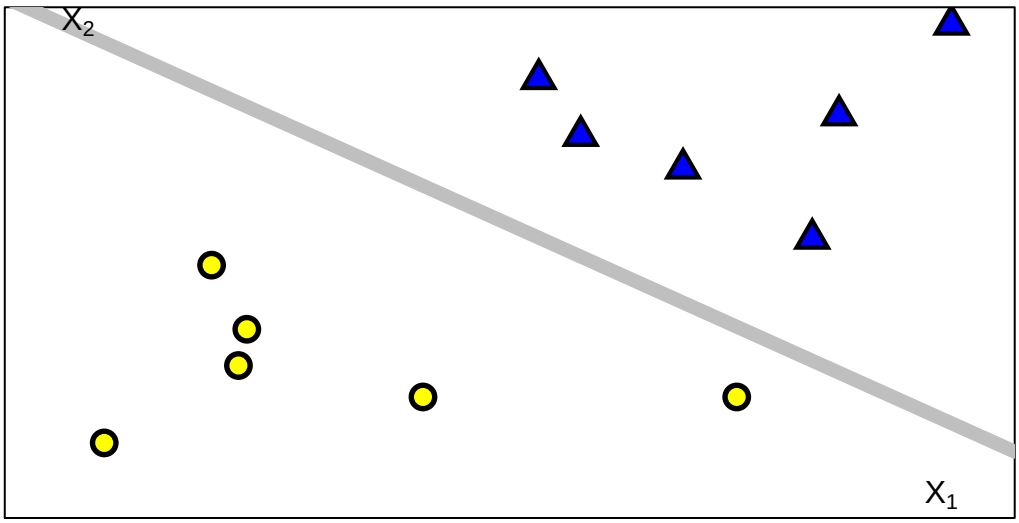
Let's consider the case of a *two–class* problem where we have IID data $\{(Y_1, \mathbf{X}_1), (Y_2, \mathbf{X}_2), \ldots, (Y_n, \mathbf{X}_n)\}$, with $Y_i \in \{0, 1\}$ the (binary) response variable and $\mathbf{X}_i \in \mathcal{X} \subset \mathbb{R}^d$ a vector of explanatory variables (or features, or covariates).

A **classification rule** or **classifier** is then any function $\eta : \mathcal{X} \mapsto \{0, 1\}$. When we observe a new covariate vector, $\mathbf{X}'$ say, we then predict the response Y' to be $\eta(\mathbf{X}')$.

Just as an example, consider the fake $n = 12$ data-points displayed in the figure below. The covariate $\mathbf{X} = (X_1, X_2)$ is 2–dimensional and the outcome $Y \in \{0, 1\} = \{\triangle, \bigcirc\}$. Also shown is a **linear** classification rule represented by the solid line. For suitable values of the parameters $(\alpha, \beta_1, \beta_2)$ to be estimated from the data, this rule is of the form

$$\eta(\mathbf{x}) = \begin{cases} 1 & \text{if} \quad \alpha + \beta_1 x_1 + \beta_2 x_2 > 0, \\ 0 & \text{otherwise.} \end{cases}$$

Everything above the line is classified as a 0 and everything below the line is classified as a 1. These two groups are perfectly separated by the linear decision boundary (see the definition below); you probably won't see such a simple pattern in real datasets.



Ok, now that we know what a classifier is, how do we evaluate its performance? Let's start introducing the following (quite natural) loss function and its *empirical* counterpart

- The **true error rate** of a classifier $\eta(\cdot)$ is: $L(\eta) = \mathbb{P}\big(\{\eta(\mathbf{X}) \neq Y\}\big)$.
- The **empirical error rate** or **training error rate** is: $\widehat{L}_n(\eta) = \frac{1}{n}\sum_{i=1}^n \mathbb{I}\big(\eta(\mathbf{X}_i) \neq Y_i\big)$.

To keep going, let's look a bit closer at the structure of the **regression function** (i.e. the conditional expected value of Y given $\mathbf{X}$) once specialized to this context

$$\begin{aligned} r(\mathbf{x}) &= \mathbb{E}(Y \,|\, \mathbf{X}) \underset{Y \text{ binary}}{=} \mathbb{P}(Y = 1 \,|\, \mathbf{X} = \mathbf{x}) \underset{\text{Bayes'TH}}{=} \frac{f(\mathbf{x} \,|\, Y = 1)\mathbb{P}(Y = 1)}{f(\mathbf{x} \,|\, Y = 1)\mathbb{P}(Y = 1) + f(\mathbf{x} \,|\, Y = 0)\mathbb{P}(Y = 0)} = \\ &= \frac{\pi_1 \cdot f_1(\mathbf{x})}{\pi_1 \cdot f_1(\mathbf{x}) + (1 - \pi_1) \cdot f_0(\mathbf{x})} \end{aligned}$$

where

$$f_0(\mathbf{x}) = f(\mathbf{x} \,|\, Y = 0), \quad f_1(\mathbf{x}) = f(\mathbf{x} \,|\, Y = 1), \quad \pi_1 = \mathbb{P}(Y = 1).$$

Nice. Last round of definitions and then *the* result

- The set $\mathcal{X}(\eta) = \{\mathbf{x} \in \mathcal{X} : \mathbb{P}(Y = 1 \,|\, \mathbf{X} = \mathbf{x}) = \mathbb{P}(Y = 0 \,|\, \mathbf{X} = \mathbf{x})\}$ is called the **decision boundary**.
- We define the **Bayes classification rule** $\eta^\star(\cdot)$ as

$$\eta^\star(\mathbf{x}) = \begin{cases} 1 & \text{if } r(\mathbf{x}) > \frac{1}{2} \\ 0 & \text{otherwise} \end{cases} = \begin{cases} 1 & \text{if } \mathbb{P}(Y = 1 \,|\, \mathbf{X} = \mathbf{x}) > \mathbb{P}(Y = 0 \,|\, \mathbf{X} = \mathbf{x}) \\ 0 & \text{otherwise} \end{cases} = \begin{cases} 1 & \text{if } \pi_1 f_1(\mathbf{x}) > (1 - \pi_1) f_0(\mathbf{x}) \\ 0 & \text{otherwise} \end{cases}$$

And here we go with the <u>main result</u> I mentioned before:

- The Bayes rule is **optimal**, that is, if $\eta(\cdot)$ is <u>any</u> other classifier, then $L(\eta^\star) \leq L(\eta)$.

The Bayes rule clearly depends on <u>unknown</u> quantities that we need to estimate from data, but **please notice**: the Bayes rule has **nothing** whatsoever to do with Bayesian inference. We could estimate the Bayes rule using either Bayesian or frequentist methods... and this is actually what you gonna do in this exercise!

So, once we have some data at hand, $\mathcal{D}_n = \{(Y_1, \mathbf{X}_1), (Y_2, \mathbf{X}_2), \ldots, (Y_n, \mathbf{X}_n)\}$ say, we may proceed to estimate:

- $f_1(\cdot)$ from the $\mathbf{X}_i$ for which $Y_i = 1$, obtaining $\widehat{f}_1(\cdot)$,
- $f_0(\cdot)$ from the $\mathbf{X}_i$ for which $Y_i = 0$, obtaining $\widehat{f}_0(\cdot)$,
- the subpopulation proportion π_1 with $\widehat{\pi}_1 = \frac{1}{n}\sum_{i=1}^n Y_i$.

and define

$$\widehat{r}(\mathbf{x}) = \frac{\widehat{\pi}_1 \cdot \widehat{f}_1(\mathbf{x})}{\widehat{\pi}_1 \cdot \widehat{f}_1(\mathbf{x}) + (1 - \widehat{\pi}_1) \cdot \widehat{f}_0(\mathbf{x})} \quad \rightsquigarrow \quad \widehat{\eta}(\mathbf{x}) = \begin{cases} 1 & \text{if } \widehat{r}(\mathbf{x}) > \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$

The simplest approach would consist in assuming a **parametric** model for the class–conditional densities $f_1(\cdot)$ and $f_0(\cdot)$. In case we pick a multivariate Gaussian for both, depending on the covariance structure we then get back well–known classifiers

known as **linear** and **quadratic discriminant analysis** models. But why would you be a dull parametric–guy? Let's go **nonparametric** and put a kernel density estimator at work here, as simple as that...

Nice and smooth, isn't it? Just a little caveat: in typical classification problems, the dimension d of the covariate vector $\mathbf{X}$ may be quite large and *any* standard density estimator (histograms, kernels, etc) will miserably fall under the blows of the curse of dimensionality. What now? Well, let's try to be *naïve* (...Bayes...) and assume that, conditionally on the class label, the feature vector $\mathbf{X}$ has independent components, that is

$$f_1(\mathbf{x}) = \prod_{j=1}^{d} f_{1,j}(x_j) \quad \text{and} \quad f_0(\mathbf{x}) = \prod_{j=1}^{d} f_{0,j}(x_j),$$

while this assumption is generally **not** true, it does simplify a lot the estimation process because:

- the individual class-conditional densities $\{f_{1,j}(\cdot)\}_j$ and $\{f_{0,j}(\cdot)\}_j$ can each be estimated separately using 1–dimensional kernel density estimates;
- if a component X_j of $\mathbf{X}$ is discrete, then an appropriate estimate can be used and seamlessly mixed with the others.

Notice that, in spite of the rather optimistic assumption, **naïve Bayes classifiers** often outperform far more sophisticated alternatives.

Finally, before we go to the actual exercise, just a few things on how to evaluate a classifier in practice. The keyword is always the same: **data splitting**. There are of course various schemes and options, but the basic one is quite simple and consists in *randomly* dividing the original dataset $\mathcal{D}_n$ in two subsets of size n_1 and n_2 with $n = n_1 + n_2$: a **training set** $\mathcal{D}_{n_1}^{\texttt{TR}} = \left\{(Y_i, \mathbf{X}_i)\right\}_{i=1}^{n_1}$ and a **test set** $\mathcal{D}_{n_2}^{\texttt{TE}} = \left\{(Y_i', \mathbf{X}_i')\right\}_{i=1}^{n_2}$ in typical proportions of 70% and 30% respectively[1].

- on the **training set** we estimate the classifier as usual. In our case, for example, this step simply consists in getting the optimal bandwidths by cross–validation to obtain $\widehat{\eta}(\cdot)$ as described above.
- on the **test set**, instead, we check the performance of $\widehat{\eta}(\cdot)$ *artificially* treating $\mathcal{D}_{n_2}^{\texttt{TE}}$ as if it was a *real* brand new dataset with no class–label inside. More specifically, we take each feature vector $\mathbf{X}_i'$ in $\mathcal{D}_{n_2}^{\texttt{TE}}$ and use the classifier $\widehat{\eta}(\cdot)$ to predict its associated outcome $\widehat{Y}_i' = \widehat{\eta}(\mathbf{X}_i')$ for $i \in \{1, \dots, n_2\}$.
  At this point, we take out of their grave the observed responses $\{Y_i'\}_{i=1}^{n_2}$ and compare them with the predictions $\{\widehat{Y}_i'\}_{i=1}^{n_2}$ in order to get the **test error rate**:

$$\widehat{L}_{n_2}^{\texttt{TE}} = \widehat{L}_{n_2}\big(\widehat{\eta}\big) = \frac{1}{n_2}\sum_{i=1}^{n_2} \mathbb{I}\big(\eta(\mathbf{X}_i') \neq Y_i'\big) = \frac{1}{n_2}\sum_{i=1}^{n_2} \mathbb{I}\left(\widehat{Y}_i' \neq Y_i'\right).$$

  Since the classifier $\widehat{\eta}(\cdot)$ and the *test error rate* $\widehat{L}_{n_2}^{\texttt{TE}}$ are based on independent samples, the latter surely is more realistic than the (rather optimistic) *training error rate*

$$\widehat{L}_{n_1}^{\texttt{TR}} = \widehat{L}_{n_1}\big(\widehat{\eta}\big) = \frac{1}{n_1}\sum_{i=1}^{n_1} \mathbb{I}\big(\eta(\mathbf{X}_i) \neq Y_i\big) = \frac{1}{n_1}\sum_{i=1}^{n_1} \mathbb{I}\left(\widehat{Y}_i \neq Y_i\right),$$

  as an estimate of the *true* missclassification error rate $L\big(\widehat{\eta}\big)$.

Remark: Beside basic `R`, for a unified approach to classification and prediction, I strongly suggest the `caret` package. There's a lot of documentation available online. You may start from the dedicated webpage and the package vignette, to continue reading a book written in `Markdown` (sic!) by the package developers, their blog and a not–so–recent jstatsoft paper. Some of the functions you might wanna look at are: `createDataPartition()`, `train()`, and `confusionMatrix()`; together with the long list of available models and a summary of the performance measures suitable for a classification problem – quite useful to interpret the output of `confusionMatrix()`.

Exercise

1. As I said above, in spite of the rather optimistic assumption, **naïve Bayes classifiers** often outperform far more sophisticated alternatives. Think a bit, look around if you wish – **citing all your sources, of course** – and write down why, in your opinion, this might actually be the case. In other words, give me your – personal but informed – point of view on the reason why, in a classification context, we don't really need all the gory details of the conditional marginals but only some particular feature that the simplifying hypothesis behind **naïve Bayes** can actually catch.

[1] Notice that in basic `R` you may simply use the function `sample()` and `setdiff()` to create them.

2. Consider the problem of distinguishing human activities performed while wearing inertial and magnetic sensor units. The dataset is publicly available at the UCI Machine Learning Repository and is described in Barshan and Yusek (2013), where it is used to classify 19 activities performed by 8 people that wear sensor units on the chest, arms and legs. For this exercise, I took the results on just 4 activities (*walking*, *stepper*, *cross trainer*, *jumping*) performed by a single person (*#1*) and as covariates the measurements taken by all the 9 sensors (x,y,z *accelerometers*, x,y,z *gyroscopes*, x,y,z *magnetometers*) on each of the 5 units on torso (**T**), right arm (**RA**), left arm (**LA**), right leg (**RL**), left leg (**LL**) for a total of 45 features.
   In this exercise you have to:

- Load the dataset `daily-sport.RData` into `R` and take a look with the usual functions (e.g. `str()`).
- Use the function `subset()` followed by `droplevels()` to reduce the dataset to two activities (e.g. *running* and *crosstr*) and only 3 sensors on one location (e.g. x,y,z *magnetometers* on the left leg). Call the resulting dataframe `ds.small`.
- Take a look at `ds.small` treating it as a 3D point cloud in the Euclidean space (e.g. plot 500 points at random using any function you like).
- Use the function `sample()` to split `ds.small` in `ds.train` and `ds.test` containing 70% and 30% of the original dataset respectively.
- Use the function `lda()` in the `MASS` package on `ds.train` to estimate a *linear discriminant analysis* model, call it `lda.out`.
- Use the function `predict()` on `lda.out` and `ds.train` to predict the class–labels on the training set, call this vector `pred.tr`. Use `pred.tr` to evalute the *training error rate* of the classifier.
- Use the function `predict()` on `lda.out` and `ds.test` to predict the class–labels on the test set, call this vector `pred.te`. Use `pred.te` to evalute the *test error rate* of the classifier.
- Compare test and training error rates and make some quick comment.
- Repeat everything using the function `naiveBayes()` in the package `e1071` and briefly comment on its performance.
- Estimate the 1-dimensional class–conditional densities of each of the three covariates using histograms *or* kernels *and* build 95% bootstrap confidence bands around them.
- Implement your own version of **naïve Bayes classifiers** and see how it compares with the `naiveBayes()` implementation you used in the previous point.

3. . . . and finally, do your best on this dataset! Crunch it as you wish to get the best classification accuracy!
