Non-Paramatric Statistics Exercise 8

Osman Ceylan, Jiahui Wang, Zhuoyao Zeng

15. Januar 2021

Exercise 3.2

Write a program that downloads the data set bank-marketing from the UCI data repository and then suitably converts this data set into a file `bank-marketing.csv` of the "usual" format with labels $y_i \in \{-1, 1\}$ and input vectors$x_{i,1}, ..., x_{i,d} \in \mathbb{R}^d$. In addition, justify each transformation of the original data you made. Finally, split the data set into two approximately equal sized data sets and save them to the files `bank-marketing.train.csv` and `bank-marketing.test.csv`.

Our Considerations for the transformations:

Generally speaking, we try to assign each variable to a linear order and normalise the respective entries in $[-1, 1]$.

Binary or trinary options, such as *yes/unknown/no* or *telephone/unknown/cellular*, are converted to $-1 / 0 / 1$ respectively.

Numeric entries, such as balance or duration are all divided by its maximum value. Notice that months are considered to be numerical values, and the "maximum"for age is assumed to be 100.

For the remaining variables, which are all categorial, the transformation is as follows:

Concerning job:

We consider the average annual salary of the jobs as the linear order. Especially, *Unknown* is assigned to the average salary of all jobs, *unemployed* is assigned to 5000 EUR ($\approx 450 * 12$), and *student* is assigned with 4000 EUR. The resulting linear oder is:

$student(4.000) < unemployed(5.000) < housemaid(33.000) < blue{-}collar(35.000) < retired(40.500)$
$< self-employed(42.000) < admin.(43.000) < technician(45.000) < services(49.300)$
$< unknown(48.000) < management(74.000) < entrepreneur(75.000)$.

Then all entries are normalised with 75.000.

Concerning marital:

The linear order taken here is: $married(-1) < divorced(0) < single(1)$.

Concerning education:

The linear order taken here is: $unknown(0) < primary(1/3) < secondary(2/3) < tertiary(1)$.

Exercise 3.3

Given a cubic partition $\mathcal{A} = (A_j)_{j\in\mathbb{N}}$ of $\mathbb{R}^d$ of width $s > 0$ and a data set $D = (\,(y_1, x_1), ..., (y_n, x_n)\,)$, and the histogram least squares rule is defined as

$$f_{D,s}(x) := \frac{\sum_{i=1}^{n} y_i \mathbf{1}_{A(x)}(x_i)}{\sum_{i=1}^{n} \mathbf{1}_{A(x)}(x_i)} \tag{1}$$

with the convention $\frac{0}{0} := 0$. Write a program which firstly reads a train data set D as well as a test data set D' from the current folder and then computes $f_{D,s}, \mathcal{R}_{L,D}(f_{D,s}), \mathcal{R}_{L,D'}(f_{D,s})$. The program should also display the times for theses calculations.

Our Solution:

At first, we implemented $f_{D,s}$ with an intuitive idea, namely: It checks in which cell $A(x)$ the point x is located, then it finds all points from D in $A(x)$ and sums up the number of points as well as their labels and returns the desired fraction.

However, this idea is inefficient and so we made following considerations:

Whenever s is choosen, the cubic partition $\mathcal{A}$ will be fixed. When the data set D is also given, we can then actually already sum up the number of points from D landing in each cell of $\mathcal{A}$ as well as their labels. Therefore, we calculate these information and store them in a dictionary. For every new input x, we just need to check in which cell x is located and then call the respective value from the dictionary.

The new implementation with the above described idea is indeed much more efficient and here we present some of the calculation results for different s (T denotes the time for the calculation):

s	$\mathcal{R}_{L,D}(f_{D,s})$	$\mathcal{R}_{L,D'}(f_{D,s})$	$T(f_{D,s})$	$T(\mathcal{R}_{L,D}(f_{D,s}))$	$T(\mathcal{R}_{L,D'}(f_{D,s}))$
1	0.10735	0.11561	0.15521	0.1571	0.149
0.8	0.09852	0.11688	0.17526	0.17146	0.18016
0.6	0.09251	0.11857	0.1782	0.17871	0.1747
0.4	0.06231	0.15956	0.17946	0.18126	0.17618
0.2	0.02457	0.29759	0.18263	0.17917	0.17013
0.1	0.00505	0.57652	0.17885	0.1797	0.1659
0.08	0.00263	0.66317	0.17959	0.18074	0.16375
0.06	0.0011	0.78449	0.21694	0.19649	0.16602
0.04	0.00044	0.92395	0.1978	0.17951	0.15108
0.02	0.00012	0.98909	0.2119	0.18245	0.15296
0.01	0.0	0.99774	0.17593	0.18312	0.14882
0.008	0.0	0.9988	0.21738	0.34856	0.17705
0.006	0.0	0.99929	0.17891	0.19099	0.29538
0.004	0.0	0.99951	0.27577	0.18162	0.14741
0.002	0.0	0.99982	0.24959	0.19278	0.14798
0.001	0.0	1.0	0.18281	0.18001	0.15209