Applied Statistics Project - On the predictability of life course

VEILLON, Juliette
first1.last1@xxxxx.com

ANDRU, Kilian
first2.last2@xxxxx.com

LACOUR, Xavier
first2.last2@xxxxx.com

MASSIN, Keryann
first2.last2@xxxxx.com

April 24, 2023

Abstract

Based on socioeconomic and genetic data, is it possible to predict one's health across time? The question has been discussed in many previous research without satisfying answer. This paper explains how we approached the problem. The data used for our purposes are the ones from the Health and Retirement Study led by the RAND Corporation. We first explain how me chose the global health index, using Stochastic Neighbors Embedding paradigm, before applying a specific kind of Lasso Regression and further machine learning algorithms.

1 Introduction

The work presented in this document has been realized in the framework of the **Applied Statistics Project** (projet Statapp) by the joint participation of Juliette Veillon, Kilian Andru, Xavier Lacour and Keryann Massin. It is dedicated to the question of the **predictability of life course**, a sociological question approached from a data science perspective.

At the era of overly present information, machine learning reveals links and hidden relationships between variables that used to be hard to tackle. Data Science techniques spread among experts of all fields, and social sciences see major breakthroughs thanks to artificial intelligence. The question of the predictability of life course comes in this context as a challenging one: there is an obvious relationship between age, body-fat ratio and health for instance, but how could one exploit these links and other available information for prediction purpose? For instance, is it possible to forecast one's overall health for the upcoming year based on the socioeconomic and genetic data at our disposal? That is the problem we wish to address with this STATAPP project.

Thanks to our supervisor M. Tropf, we have had access to the Heath and Retirement Study (HRS) led by the RAND Corporation, that consists in interviews of seniors in the United States followed over several years (28 years at most). **REMPLIR**

2 Presentation and cleaning of the database

2.1 Presentation

REPRENDRE LE TEXTE DE LA NOTE DE MI-PARCOURS

2.2 Initial merges

REPRENDRE LE TEXTE DE LA NOTE DE MI-PARCOURS

2.3 Removal of unusable variables

At this point, our database gathers $n =$ **REMPLIR** individuals and $p =$ **REMPLIR** variables. With a size of about **REMPLIR** gigabytes and quadruple in RAM, our laptops could not handle the whole dataset and perform appropriate analysis. Plus, considering that the dataset presents a high missing values rate and that $p \approx n$, machine learning algorithms would not be able to converge in the current state of our data. We need to clean it.

2.3.1 Removal of columns with a missing rate too high

The quickest way to efficiently reduce the size of our database is to directly drop useless columns. Our data mainly corresponding to answers to interviews, some columns present a missing rate really high. We calculated for each wave the number of columns for a given missing rate, and below is the graph we got.
REMPLIR
The genetic data are available for around **REMPLIR**% of the respondents, so we decided to drop all variables that presented a missing rate higher than **REMPLIR**%. This single operation drastically reduced the size of the database from **REMPLIR** to **REMPLIR**.

2.3.2 Removal of Spouses' related variables

The subject of the variables always fall into one of the three categories 'Respondent, Spouse, Household'. Because Spouses' related variables amounted to around **REMPLIR**, we decided to drop them all, considering the fact that **REMPLIR**% of spouses were also interviewed as respondents. At the end of this operation, our database is of size **REMPLIR**.

2.3.3 Removal of health variables

The whole point of our project is to predict a global health index. As explained further on, we used the health variables to construct our index. To remove endogeneity issues, we decided to drop all health variables (even though we used only **REMPLIR** of the **REMPLIR** available to create our index) in the main database, and save them in a separate database. At the end of this operation, our database is of size **REMPLIR**. No obvious columns removal could be made for now.

3 Definition of the objective

As explained in the introduction, we want to see whether it is possible to forecast someone's health with the socioeconomic and genetic data we have about him at our disposal. We formally need a *Global Health Index*, that could summarize the whole of health-related information into one real number. Our goal will then be to attempt to predict it. This section is dedicated to the creation of this index.

We carefully selected a few quantitative health-variables, from his age and body mass index to the number of overnight stays at hospital or in nursing home he went through over a year. We also took into account qualitative variables indicating whether the respondent had had hypertension, cancer, diabete, heart disease, etc. The problem was to summarize 27 health-variables into one Global Health Index.

The scientific literature points out the benefits of Principal Component Analysis (PCA) in the search of such an index. Therefore our first attempts were to use PCA to summarize the 27 variables into one. However, the index created this way did not present the expected consistency: one's index looked random across time, and the index was awkwardly positively correlated with each health-variable, independently of the actual effect of the variable on one's health.

Hence, we looked for another dimensional reduction algorithm. The t-distributed Stochastic Neighbor Embedding algorithm has caught our attention, and we decided to apply it on our health data.

3.1 t-distributed Stochastic Neighbor Embedding

3.1.1 Presentation

The PCA is designed to reduce a dataset of high dimension d into a smaller space of dimension $d' << d$ by finding the *principal components*, that is the d' maximum variance directions, and project the original data in the smaller space. Globally, this is an linear algebraic oriented approach.
On the other hand, the t-distributed Stochastic Neighbor Embedding (t-SNE) tries to find similarities in the data using a non-linear probabilistic approach. The overall idea consists in assigning to the original data a probability measure P and another one Q to the reduced data, then trying to alter the reduced data such that the Kullbach-Leibler divergence of the distribution P and Q is minimal.

3.1.2 Formal details

Formally, considering $(x_1, x_2, \dots x_n) \in \mathbb{R}^{n\times d}$ the original high-dimensional dataset and $(y_1, y_2, \dots, y_n) \in \mathbb{R}^{n\times d'}$ the reduced dataset, we compute the similarity of x_j to x_i by the conditional probability $p_{j|i}$ that x_i would pick x_j as its neighbor under a Gaussian distribution centered at x_i, where:

$$p_{i|i} = 0 \quad \text{and} \quad p_{j|i} = \frac{\exp(-\|x_i - x_j\|^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-\|x_i - x_k\|^2/2\sigma_k^2)}, \quad \forall i \neq j \tag{1}$$

with σ_i the bandwidth of the Gaussian kernels.
Once $p_{ij} := (p_{i|j} + p_{j|i})/(2n)$ defined, we also define a probability measure Q on the reduced dataset following a Student t-distribution:

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_k \sum_{\ell\neq k}(1 + \|y_k - y_\ell\|^2)^{-1}} \tag{2}$$

The locations of points y_i are determined by minimizing the Kullbach-Leibler divergence of P from Q, i.e minimizing

$$KL(P\|Q) = \sum_{i\neq j} p_{ij} \ln\left(\frac{p_{ij}}{q_{ij}}\right) \tag{3}$$

with respect to $y_1, y_2, \ldots, y_n$ by a gradient descent.

3.1.3 Performances

The main objective of this algorithm is to reflect the similarities in the original data. We directly applied the `sklearn.manifold.TSNE` implementation on our health data and checked its consistency. Below are the results.
REMPLIR

4 First machine learning procedures to format the database

At this point, the dataset we are working on still presents a few thousands variables. Considering that it approaches the number of individuals (that is, $p \approx n$), machine learning algorithms will have struggle converging, or would converge to unsatisfying solutions. Hence, we shall reduce the number of such variables by other means than deletion (every possible removals having already been previously realized).
According to the literature, among the many possible techniques to reduce dimensionality, the Lasso Regression is the suitable one. The idea is to train a Lasso Regression on (X, y), where X are the available data and y is the global health index, to come up with an estimator. The Lasso estimator has the property to shutdown to zero its coordinates when the associated variable does not present a prediction power strong enough. This technique has the advantage of being quick to train and to apply. *A priori*, the estimator we would get from this approach will not be really accurate at predicting, but it will not matter as we are only interested in its coefficients' nullity.

The library **scikit-learn** in Python would allow us to reach our goal through the following code:

```
from sklearn.linear_model import Lasso
lasso = Lasso()
lasso.fit(X, y)
estimator = [lasso.intercept_]+list(lasso.coef_)
```

However this code is bound to fail in our specific case, because X presents many missing values, as discussed in sections before. Hence, we have to adapt the method. For the record, we will present the three method we thought of, the two first ones having failed.

4.1 Imputation of missing data

To still apply the lasso as discussed before, we could try imputing the missing data. Some imputers are already implemented in **scikit-learn**, such as the `SimpleImputer(strategy="mean")` that naturally imputes missing values in one column with the empiric mean of this column. The problem with

this approach is that it does not take into account the inter-variable relationships in imputation at all. To overcome this issue, we would rather use the `IterativeImputer()`, which relies on the correlations and relationships between columns to impute missing values. Its inner algorithm calculates at each iteration a new imputed database $\tilde{X}_t$ according to specific constraints, and tries to minimize the difference $\Delta_t = \tilde{X}_t - \tilde{X}_{t-1}$. When this difference is small enough, the algorithm stops and the solution $\tilde{X}_t$ is returned. This resulting database would present no missing values, and imputed data would be a lot more likely than what the `SimpleImputer()` would obtain. But despite all our attempts, the algorithm did not converge in any way. This is due to the fact that X is too big and the computer had trouble processing. This idea failed.

4.2 Convex conditioned Lasso

Instead of looking for an application in two steps of the Lasso (first imputing the data, then fitting the Lasso), another idea is to directly modify the Lasso optimization program so that it handles missing values by itself, without too heavy imputations. The state-of-art algorithm for that problem is the *CoCoLasso* (which stands for *Convex Conditioned Lasso*). This algorithm badly performs when missing rates are too high, which is our case. We then cannot use it in its first form. However, further researches helped us discover a very similar estimator, that we shall now present.

4.3 Lasso with High Missing Rate

The idea of directly applying the Lasso still being interesting, our final approach is also based on it. According to the paper REF, it is possible to edit the CoCoLasso to make it more appropriate for dataset with high missing rate. Let us dive into the mathematical details.

4.3.1 Why is CoCoLasso not fitted for high missing rate

Consider a linear regression model $y = X\beta + \varepsilon$, where $X \in \mathbb{R}^{n\times p}$, $y \in \mathbb{R}^n$ are the data, $\varepsilon \in \mathbb{R}^n$ a noise and $\beta \in \mathbb{R}^p$ the estimator we wish to determine. We assume that y and each column of X are centered.
If X does not contain missing values, Lasso is applicable, and its optimization program is:

$$\hat{\beta} = \arg\min_{\beta} \frac{1}{2n}\|y - X\beta\|_2^2 + \mu\|\beta\|_1 \tag{4}$$

where $\mu > 0$ is a regularization parameter, $\|\cdot\|_1$ is the ℓ_1 norm and $\|\cdot\|_2$ is the ℓ_2 norm. By the way, the presence of ℓ_1 norm is responsible for the solution $\hat{\beta}$ to have many null coordinates (the property we are interested in).
The equation (4) can be equivalently rewritten as following:

$$\hat{\beta} = \arg\min_{\beta} \frac{1}{2}\beta^T S\beta - \rho^T\beta + \mu\|\beta\|_1 \tag{5}$$

where $S = \frac{1}{n} X^T X$ is the sample covariance matrix of X and $\rho = \frac{1}{n} X^T y$ is the sample covariance vector of X and y.
In presence of missing values, This means that we can just estimate S and ρ without having to impute the whole database, and this is in fact a much easier task to achieve. Indeed, we can construct unbiased estimators of S and ρ using the pairwise covariance, that is, $S^{\text{pair}} = (S^{\text{pair}}_{jk})$ and $\rho^{\text{pair}} = (\rho^{\text{pair}}_{j})$ where:

$$S^{\text{pair}}_{jk} = \frac{1}{n_{jk}} \sum_{i \in I_{jk}} X_{ij} X_{jk} \quad \text{and} \quad \rho^{\text{pair}}_{j} = \frac{1}{n_{jj}} \sum_{i \in I_{jj}} X_{ij} y_i \tag{6}$$

for $I_{jk} = \{i \mid X_{ij} \text{ and } X_{jk} \text{ are observed}\}$ and $n_{jk} = \operatorname{card} I_{jk}$. Thus, we can relace S and ρ by S^{pair} and ρ^{pair} in (5).
The major problem here is that S^{pair} may *not* be positive semidefinite (PSD). This would lead (5) to have no real solution and is therefore a fatal issue. The easiest way to resolve this problem is to replace S^{pair} by $\tilde{\Sigma}$, defined as:

$$\tilde{\Sigma} = \underset{\Sigma \succeq 0}{\arg\min} \|\Sigma - S^{\text{pair}}\|_{\max} \tag{7}$$

where $\|A\|_{\max} := \max_{i,j} |A_{ij}|$. Denoting $\Sigma \succeq 0$ a PSD matrix Σ, the optimization problem becomes:

$$\begin{cases} \tilde{\Sigma} & = \underset{\Sigma \succeq 0}{\arg\min} \|\Sigma - S^{\text{pair}}\|_{\max} \\ \hat{\beta} & = \underset{\beta}{\arg\min} \frac{1}{2} \beta^T \tilde{\Sigma} \beta - {\rho^{\text{pair}}}^T \beta + \mu \|\beta\|_1 \end{cases} \tag{8}$$

The system (8) is actually the optimization program solved by the CoCoLasso discussed in the previous section.

As stated before, a high missing rate can deteriorate estimations of the covariance matrix in CoCoLasso: if some pairwise observation numbers n_{jk} are very small, then the corresponding pairwise covariances S^{pair}_{jk} are quite statistically unreliable. As a result, other estimator elements can highly deviate from the corresponding elements in S^{pair}, even if their variables have few missing values. The core issue is that CoCoLasso does not account for the differences in reliability of the pairwise covariance. The next section describes how to overcome this problem.

4.3.2 Advantages of HMLasso over CoCoLasso

To fully understand the advantages oh HMLasso (High Missing rate Lasso) over CoCoLasso, let us first describe the program it solves.
Let Z be the mean imputed data of X. That is:

$$Z_{jk} = \begin{cases} X_{jk} & \text{if } X_{jk} \text{ is observed} \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

We then compute the mean imputed covariance matrix, where each coefficient is weighted

$$S_{jk}^{\text{imp}} = \frac{n_{jk}}{n} \cdot S_{jk}^{\text{pair}} \tag{10}$$

Thus, $S^{\text{imp}} = R \odot S^{\text{pair}}$ where $\odot$ denotes the element-wise product and $R = (n_{jk}/n)$ is the matrix of weights. This allows us to take into account the difference of reliability of each coefficients of S^{pair}
We can then state the HMLasso optimization program:

$$\begin{cases} \tilde{\Sigma} & = \underset{\Sigma \succeq 0}{\arg\min} \, \|W \odot (\Sigma - S^{\text{pair}})\|_2^2 \\ \hat{\beta} & = \underset{\beta}{\arg\min} \, \frac{1}{2} \beta^T \tilde{\Sigma} \beta - {\rho^{\text{pair}}}^T \beta + \mu \|\beta\|_1 \end{cases} \tag{11}$$

with W the weight matrix of coefficients $W_{jk} = R_{jk}^{\alpha}$ for a fixed constant $\alpha \geq 0$.

This program had the $\|\cdot\|_{\max}$ norm replaced by the ℓ_2 norm and a weight matrix (W) added in the first optimization problem to account for the reliability of each S_{jk}^{pair}. As a result, this implementation is apt to efficiently deal with high missing rates.

4.3.3 Implementation in Python

A research in the literature did not provide us with any already implemented instance of the HMLasso in Python, so we decided to create our own.
The `cvxpy` library provided the solver, and we were able to create a class `HMLasso(mu = 1, alpha = 1)` that fits (X, y) and returns an estimator `beta_opt` useful to drop variables whose coefficient in `beta_opt` are almost 0. Below is an example of what the algorithm does and a sample of its source code:

```
from HMLasso import HMLasso
lasso = HMLasso(mu = 10, alpha = 1) # set values of mu and alpha
lasso.fit(X, y)
estimator = lasso.beta_opt
y_pred = lasso.predict(X_test) # predict X_test using lasso
```

And the source code:

```
class HMLasso():
  def __init__(self, mu=1, alpha=1):
  ...

  def fit(self, X, y):
    self.__verify_centering__(X, y)
    S_pair, rho_pair, R = self.__impute_params__(X, y)
    self.Sigma_opt = self.__solve_first_problem__()
    self.beta_opt = self.__solve_second_problem__()

  def predict(self, X):
    return np.dot(X, self.beta_opt)

  ...
```

The implementation presented a lot of difficulties, one of them regarding the fact that the underlying solver was not able to assert the positive semidefinitiveness of `Sigma_opt` due to its large size, even though the matrix was indeed PSD. As this assertion was an integral part of the resolution process, getting around this check presented many challenges, not to mention that each correction attempt required 30 minutes of simulation. In the final version of the `HMLasso()`, the user is invited to handle such errors when they happen by manually setting `ERRORS_HANDLING = "ignore"`, meaning *'ignore PSD check'*.

References

[1] Donald E. Knuth (1986) *The TEX Book*, Addison-Wesley Professional.

[2] Leslie Lamport (1994) *LATEX: a document preparation system*, Addison Wesley, Massachusetts, 2nd ed.