Machine Learning (CS-433) – Project 2

Regularized maximum likelihood estimation

Supervisor: Ortelli Nicola, Michel Bierlaire **Laboratory:** TRANSP-OR

Thomas Benchetrit, Romain Palazzo, Eliott Zemour

1 Introduction

Discrete Choice Models (DCMs) aim to understand and model choices of individuals among a finite and discrete set of alternatives. Such models are mainly used by forecasters and policymakers to study consumer demand (e.g for transport mode choice). DCMs are based on Utility theory, which introduces the so-called utility functions designed to represent the preference structure of individuals among a given choice set.

Specifications in DCMs are often introduced according to intuition, judgment and prior behavioral assumptions of the modeller. Therefore, utility specification represents a difficult task that one could see as a variable selection problem. In this report, we investigate an automatized approach by introducing a regularization term in the likelihood function, aiming to push towards zero the parameters associated with the least relevant explanatory variables. Finally, the most penalized variables are discarded from the model with the expectation of not loosing too much information.

2 Theory

DCMs requires defining some important concepts and quantities. From now on, the subscript i (or j) will denote the alternative and n will specify the individual considered. First, the choice set $\mathcal{C}_n$ contains all the possible choice (available alternatives) of individual n. For each alternative i within $\mathcal{C}_n$, we refer to the utility function as U_{in} corresponding to the level of attractiveness perceived by person n. The *random utility* approach assumes that individuals are rational utility maximizers, that is they always select the alternative with the highest utility. However, analysts are assumed to be unable to understand and model all the relevant factors that affect human behavior. The utility functions are therefore treated by the analyst as random variables. Equation 1 shows that U_{in} can be separated into two additive parts with V_{in} an observable (deterministic) component and $\varepsilon_{in} \sim \text{EV}(0, \mu)$ a non observable one which follows an extreme value distribution with unknown parameter μ.

$$U_{in} = V_{in} + \varepsilon_{in} \tag{1}$$

Thus, the probability that individual n chooses alternatives i is $P(i|\mathcal{C}_n) = P(U_{in} \geq U_{jn}, \forall j \in \mathcal{C}_n)$. Moreover there is the condition $\sum_{i \in \mathcal{C}_n} P(i|\mathcal{C}_n) = 1$. The subtraction of two random variables following an extreme value distribution with mode 0 and same μ follows a Logistic distribution of mode zero and μ. Therefore, the probabilistic law of the model is given by Eq.2:

$$\begin{aligned} P(i|\mathcal{C}_n) = & \quad P(\varepsilon_{jn} - \varepsilon_{in} \leq V_{in} - V_{jn}, \forall j \in \mathcal{C}_n) \\ = & \quad \frac{e^{\mu V_{in}}}{\sum_{j \in \mathcal{C}_n} e^{\mu V_{jn}}} \end{aligned} \tag{2}$$

It remains to define the so-called deterministic part V_{in} of the utility in Eq.1. This term involves the explanatory variables of the model, which contains alternative attributes and socioeconomic characteristics (income, gender, age...). By denoting $\boldsymbol{x}_n$ the vector of explanatory variables of individual n, one can write $V_{in} = V_{in}(\boldsymbol{x}_n, \boldsymbol{\beta})$ with $\boldsymbol{\beta}$ being the vector a parameters of the model.

The choice of unknowns and the functional form of V_{in} belongs to the analyst. However, in most cases, these functions are linear in the unknowns parameters $\boldsymbol{\beta}$ and a feature expansion in $\boldsymbol{x_n}$ is potentially performed. Moreover, thanks to this form, μ does not need to be specified because one can write $\boldsymbol{\beta} = \mu\boldsymbol{\beta}'$ and just consider $\boldsymbol{\beta}$ as a variable. Therefore, V_{in} can be defined by Eq.3 with K variables:

$$\begin{aligned} V_{in}(\boldsymbol{x}_n, \boldsymbol{\beta}) = & \quad \boldsymbol{\beta}^T \boldsymbol{x}_n \\ = & \quad \beta_1 x_{in1} + \cdots + \beta_n x_{inK} \end{aligned} \tag{3}$$

The logistic model studied in the Machine Learning course is very similar to the one developed above. However, in such models, the model involves more than two alternatives (classifications) and we refer to it as a multinomial logit (MNL) model.

The parameters β_i's are usually estimated through Maximum Likelihood Estimation (MLE). In this report, the log-likelihood defined in Eq.4 shall be maximized (which is equivalent to minimize the negative), where y_{in} values 1 provided individual n has chosen the alternative i, 0 otherwise (y_{in} are known in the case of stated preference data). N refers to the number of sample.

$$\mathcal{L}(\beta) = \frac{1}{N} \sum_{n=1}^{N} \sum_{i \in \mathcal{C}_n} y_{in} \log \left[P(i|\boldsymbol{x}_n, \beta) \right] \tag{4}$$

2.1 Purpose of the project

This project focuses on the initial step of DCMs, that is the feature selection problem in the framework of utility specification. Indeed, by introducing a regularization term in the cost function (which is the negative of the log-likelihood), the model aims to effectively push the least relevant variables towards zero. In the absence of prior behavioral assumptions, this approach should be considered to perform an unbiased relevance check of features instead of performing the utility specification task.
In DCMs, there is a trade-off between the goodness of fit and the complexity of the model and the law of parcimony should always be kept in mind. For two models with similar likelihoods, the preferred one shall be the one with the lowest number of parameter.

2.2 Data preprocessing

To regularize the number of parameters, the first step is to generate the model which has the highest number of parameter possible. A good way to do this is to use **segmentation**. Indeed, in a dataset, there is features that can take

continuous values or discreet ones. According to the definition of V_{in}, they are defined the same way as the utilities. The segmentation corresponds to the division of a feature into several ones according to a rule. For instance, in the transport mode choice, if one has a feature 'season' which can take integer value between 1 and 4 and a continuous feature 'travel time', this last can be segmented into four features travel time : spring, summer, autumn and winter. The information about the season is therefore displayed into the new features which are continuous and the number of parameters has increased because one has passed from one β for travel time and one for season to four. Moreover, the regularization approach requires data normalization, so that no parameters are boosted because of the magnitude of the associated explanatory variable. The normalization chosen is the MinMax one, defined by Eq.5 for the feature n. The modified feature corresponds to $\hat{\boldsymbol{x}_n}$. This choice of normalisation is justified by the need of positive value in the BoxCox transformation developed in the next section.

$$\hat{\boldsymbol{x}}_n = \frac{\boldsymbol{x}_n - \min_{x_i \in \boldsymbol{x}_n}(x_i)}{\max_{x_i \in \boldsymbol{x}_n}(x_i) - \min_{x_i \in \boldsymbol{x}_n}(x_i)} \tag{5}$$

2.3 Box-Cox transformation

In machine learning, a feature is used at its best when it is taken to a specific power, that's why power expansion is realised. The Box Cox transformation allows one to estimate the power λ in which the features is the more relevant. This induces non linearity in the model . The Box-Cox transformation of the feature j is given in Eq.6 where $\boldsymbol{x}_j^{(\lambda)}$ is the transformed data. As the β's, the λ's are estimated from the data. To be more general, as there is one β by feature, there is also one λ by feature. Therefore this transformation multiplies the number of parameters by 2.

$$\boldsymbol{x}_j^{(\lambda)} = \begin{cases} \frac{\boldsymbol{x}_j^{\lambda}-1}{\lambda} & \text{if } \lambda > 0 \\ \log(\boldsymbol{x}_j) & \text{if } \lambda = 0 \end{cases} \tag{6}$$

2.4 Regularization

Finally, the LASSO (Least Absolute Shrinkage and Selection Operator) and Ridge regularization are added to the model in order to select the most relevant features. They are respectively defined by Eq.7 and Eq.8 with $\mathcal{L}_L$ for LASSO, $\mathcal{L}_R$ for Ridge, K the number of β parameter and λ_L and λ_R two hyper parameter to optimize in order to regularize properly. The important thing to notice is that the regularization is not done for the λ of Box Cox because they are only here to improve the dataset.

$$\mathcal{L}_L(\boldsymbol{\beta}, \lambda_L) = \lambda_L \frac{1}{N} \sum_{i=1}^{K} |\beta_i| \tag{7}$$

$$\mathcal{L}_R(\boldsymbol{\beta}, \lambda_R) = \lambda_R \frac{1}{N} \sum_{i=1}^{K} \beta_i^2 \tag{8}$$

2.5 The model

After these modifications, the observable part of the utility of a person n and alternative i becomes Eq.9 with $\hat{\boldsymbol{x}}_n$ the normalized row of the dataset, K the number of features and ASC the alternative specific constant.

$$V_{in}(\hat{\boldsymbol{x}}_n, \beta, \lambda) = \mathrm{ASC}_i + \beta_1 \frac{\hat{x}_{in1}^{\lambda_1} - 1}{\lambda_1} + \cdots + \beta_K \frac{\hat{x}_{inK}^{\lambda_K} - 1}{\lambda_K} \tag{9}$$

Then, from Eq.2,4,7,8,9, the 'cost function' of the problem can be defined. As there is the possibility to use just one of the 2 regulations or both, one have 3 possibles cost function defined respectively in Eq.10 :

$$\begin{aligned} \mathcal{L}_I &= -\mathcal{L}(\boldsymbol{\beta}, \boldsymbol{\lambda}) + \mathcal{L}_{\mathrm{L}}(\boldsymbol{\beta}, \lambda_L) + \mathcal{L}_{\mathrm{R}}(\boldsymbol{\beta}, \lambda_R) \\ \mathcal{L}_{II} &= -\mathcal{L}(\boldsymbol{\beta}, \boldsymbol{\lambda}) + \mathcal{L}_{\mathrm{L}}(\boldsymbol{\beta}, \lambda_L) \\ \mathcal{L}_{III} &= -\mathcal{L}(\boldsymbol{\beta}, \boldsymbol{\lambda}) + \mathcal{L}_{\mathrm{R}}(\boldsymbol{\beta}, \lambda_R) \end{aligned} \tag{10}$$

3 Implementation : PyLogit package

PyLogit ([2], [3]) is a python package which performs the maximum likelihood estimation (MLE) for various choice type model. It's the package used for this project because it computes this in an efficient way thanks to long format data (as in R). However some modifications of the package must be realised to include regularization and Box-Cox transformation. The MLE is realised with the following step :

Step 1 : Convert the data file into long format. Initially, the data contains one row per observation while the long format will contain one row per observation per alternative. Therefore the number of row increases. It's at this step that the **segmentation** is done. The initial data file is load thanks to Panda package. The dataframe obtained allows one to add and manipulate features easier. Then after modification of the dataframe, the pylogit function `create_long_format()`is called to generate the long format.

Step 2 : Specify the utilities and generate the model. The utilities and the model are specified thanks to Python dictionary which defines the variables and indicates their place in the utilities. Then the PyLogit function `create_choice_model()`generates a PyLogit instance of the class of the model chosen. In our case, it's the multinomial logit model because there is more than 2 alternatives.

Step 3 : Computes the MLE. The instance previously created contains a function `fit_mle()` which does the MLE according to some optimization parameters like the maximum number of iteration `maxiter` for instance. It's here that the regularization and the Box-Cox transformation are specified thanks to new input parameters : `lasso, ridge` and `boxcox`, which correspond to λ_L, λ_R and a boolean to precise if the Box-Cox transformation needs to be done. Then, `fit_mle()` will call the `minimize()` scipy function to perfom the minimization thanks to BFGS algorithm and the optimisation parameters as input. This last takes also as input a method which computes the function and the gradient of the function to minimize. In PyLogit these lasts are `calc_likelihood()` and `calc_gradient()` and must be modified to include Box-Cox and regularization.

3.1 Log-likelihood and grad-log likelihood computation

The log-likelihood is computed from Eq.4 by calling the method `calc_probabilities` which returns a vector of 'chosen probabilities' (the probability of the alternative chosen by the person) as they are the only non vanishing terms in Eq.4. These probabilities are defined following Eq.2. When Box-Cox is not realised, V_{in} corresponds to Eq.1 and thanks to long format shape, all the utilities are contained in a vector which can be obtained by a matrix product between a 'design' matrix and the parameter vector. The 'design' matrix is generated by the method `create_design_matrix()` in PyLogit and is constructed from the utility specification done in Step 2. Therefore, when there is Box-Cox, the transformation is realised over this matrix according to the vector of $\boldsymbol{\lambda}$ thanks to the `boxcox()` method of scipy before doing the product with $\boldsymbol{\beta}$. Then, the regularizations are computed by adding terms at the end of the two methods `calc_likelihood()` and `calc_gradient()`. These terms come from the methods `lasso()` and `ridge()` which can return the regularization or its gradient depending on an boolean input. To ensure that $\lambda_i \geq 0$ $i = 1, \ldots, K$ boundaries are defined with the input argument `bounds` of `minimize()`. At last, some instructions are written where Box-Cox and its gradient are defined to avoid overflow issues and to manage the case of '0' in the design matrix with λ=0.

4 Result

To estimate our model we used the London Parameter Model Choice (LPMC) dataset which countains 80 000 entries, gathered over 3 years. However as the dataset requires tremendous CPU power, the simulations were conducted with a sample of 26 320 entries, which corresponds to the data gathered over a single year.

4.1 Box-Cox transformation

Using the Box-Cox transformation seen in Eq.6, one can estimate the λ_i which maximizes the likelihood for each parameter x_i. Therefore one should obtain a better maximized likelihood using Box-Cox transformed variables

neg-LL without Box-Cox	neg-LL with Box-Cox
-0.889	-0.874

Table 1: Negative Log-likelihood of the model with and without Box-Cox transformation

The results of both the simulations are referenced in Tab. 1. As expected, the likelihood of the model with the Box-Cox transformation is higher than the one obtained without it. Therefore the Box-Cox transformation helps to give the model a better predictive power. By taking a look to the optimized λ_i, the modeler can deduce the best shape for each feature and therefore better leverage the dataset.

4.2 Regularization as a method to identify the most relevant variables

Having implemented the regularization methods mentioned above, a grid search have been performed to optimize the associated hyper-parameters λ_L and λ_R (Eqs.7,8). For each pair (λ_L, λ_R) considered in the grid search, the array of estimated and regularized parameters $\hat{\boldsymbol{\beta}}_{\text{reg}}$ is stored. These parameters are then sorted by absolute value, which is assumed to represent the relevance of the associated variable in the DCM. The main objective is to track the evolution of the fitted log-likelihood as a function of the number of included parameters in the model. Specifically, for each regularization pair, the absolute value sorting provides an order in which the parameters are added. Note that when a β-parameter is added, its associated Box-Cox λ-parameter is also included in the model.
The model is first evaluated with only three intercepts (and labelled as "0-added parameters"), therefore it is common to all regularization pair. Then, parameters are sequentially added to the model and the fitted log-likelihood is computed. The process ends when the full set of parameters is included in the model, and the resulting likelihood is again common to all regularization pair. Moreover, this absolute value sorting for adding parameters is compared with a random order, averaged over several seeds. In this case, the shape of the plot is expected to be linear, since each parameter added should contribute, on average, equally to the growth of the log-likelihood. Finally, when no regularization is performed, the parameters are still sorted by absolute value and these results are shown under the label "no regularization" on the plots.

4.2.1 Likelihood estimation

On Fig.1 is displayed the result of the log-likelihood depending on the number of added parameter for different regularization settings. The optimized Lasso, Ridge and (Lasso + Ridge) parameters were found by doing a grid search .

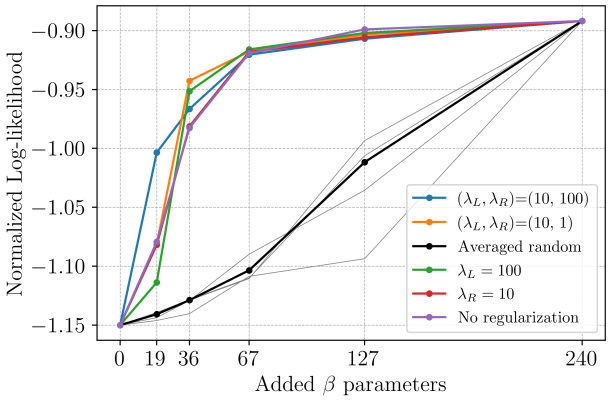

Figure 1: Log-likelihood vs. number of added parameters for different hyper-parameters (λ_L, λ_R). The dark random line represents the average over all random simulation performed and shown in grey.

4.2.2 Parameters estimation

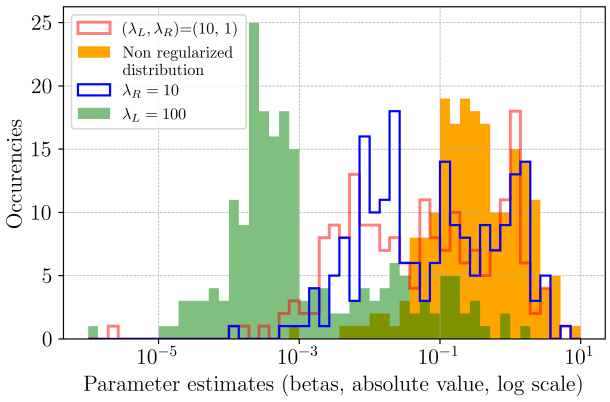

Figure 2: magnitude of the fitted parameters β

On Fig.2 is displayed the order of magnitude of the fitted parameters according to the method of regularization.

4.3 Out of sample validation

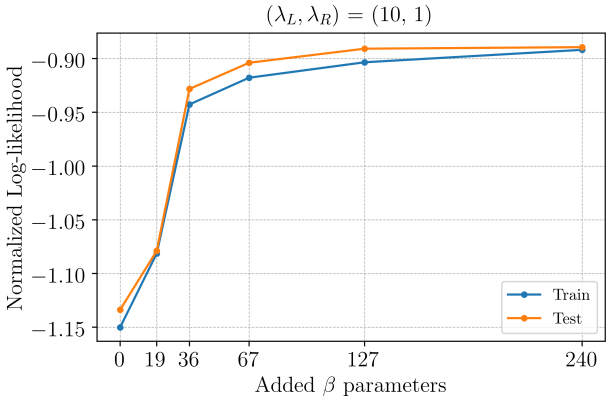

Figure 3: Out of sample validation for the model

An out of sample validation was done to estimate how the model performs on unseen data and the results are compiled in Fig.3. The test set was made using the data gathered on two years that are available in the LPMC dataset.

5 Discussion

5.1 Box-Cox transformation

As seen on Tab.1, the Box-Cox model gives a better log-likelihood. Therefore one can assume that the dependencies of the prediction on the parameters is non-linear. In DCM applied on transport choice, a marginal bump on transport cost is assumed to have a lower effect the bigger the cost is. Therefore we expect for the prediction to be of logarithmic dependency for the transport cost. However, thanks to the Box-Cox transformation, the modeler can choose the best shape of a feature with a non-arbitrary method.

5.2 Regularization

On Fig.1, even though the hyper-parameters were different, the simulations yield close results especially when a few number of parameters were already added. From this can be deduced that all the regularization methods permits to estimate accurately which parameters are the most relevant in the utility specification. Moreover the "no regularization" simulation also permits to estimate which parameter are the more relevant, but the parameters are not all well-chosen, as we can see that for example the (10,1) simulation has a better likelihood with the same number of parameters kept. Nevertheless, as displayed on Fig.2, the simulations look rather differently if one considers the distribution of the parameters. Indeed, when the model is not regularized, the parameters are not constrained whereas the regularization pushes the βs to lower values. When the lasso regularization is used, the parameters get "separated" into two groups: those under 10^{-3} and those above. The first group are the ones that the modeler may discard in a DCM if the number of parameters need to be reduced. However those above kept a consequent value even after a regularization, which means that they must be of some importance to determine the prediction. This distinction cannot be that easily done with the other methods of regularization, or even with the non-regularized simulation.

5.3 Out of sample validation

In Fig.3, the out-of-sample validation reveals that the test log-likelihood is higher than the one obtained using the train dataset. This result means that the model trained is under-fitting. During the simulations, the scipy `minimize()` function was given a `maxiter` parameter which determined the maximum running time the minimizer was allowed to run. As the dataset was too big to allow the computers to give exploitable results in a reasonable amount of time it has been chosen to reduce the `maxiter` to 50. Therefore the model may not have enough time to reach a local maximum, and therefore underfits. Another reason for the underfitting is that the main purpose of the project was to select relevant variables through regularization. Therefore, the regularization has been pushed to optimize the separation between usefull and useless parameters, not to find the best values for the parameters that fits the data.

6 Conclusion

In this report, we investigated regularization as a method to identify the most relevant variables for the utility specification in a DCM. Ridge and LASSO regularization were implemented in the Python pylogit package and performed on the LPMC dataset. As a result, through grid search, we found hyperparameters (λ_L, λ_R) for which this method seemed to be very effective and performed much better than a random consideration of parameters (Fig.1).
To go further, a more extensive segmentation in the parameters could have been pursued, as well as simulations over the full LPMC dataset. The number of iterations allowed for the minimization algorithm have also been reduced, and more accurate results could have been obtained by increasing this parameter. However, due to insufficient computational power, the results presented in this report are close to the best we could reach.

References

[1] Nicola Ortelli (2019). *Automatic Utility Specification in Discrete Choice Models* [Master project in Civil Engineering]. École Polytechnique Fédérale de Lausanne (EPFL).

[2] Brathwaite, Timothy, and Joan Walker (2016).
Asymmetric, Closed-Form, Finite-Parameter Models of Multinomial Choice. arXiv preprint arXiv:1606.05900. http://arxiv.org/abs/1606.05900.

[3] Brathwaite, Timothy, Pylogit, (2016). GitHub
repository, https://github.com/timothyb0912/pylogit