## Modelling consumer choices under different assumptions

This part of the work aims at presenting the results of the estimation for our selection of the econometric and ML models. We should particularly underline the fact, that this section does not focus on the performances of the models as they will be discussed more in detail latter. There is still a double objective for this section, as before presentation of the obtained results, we should discuss the methods and techniques, which were implemented in order to estimate the models, presented earlier.

The estimation procedure and choice of the estimation algorithms as well as their numeric implementation in the statistical software are important in the context of model performance comparison. The different estimation procedures may lead to different results and different conclusions.

We consecutively estimate the chosen models over the two datasets: with and without the presence of heterogeneous preferences of the individuals for the environmental attributes. Then we compare the estimates with the target values we have used previously as inputs in defining the relative utility functions.

### Estimation procedures

In this section we will discuss the different techniques implemented in order to estimate the different models, which were described in the first theoretical part of this work. The different algorithms may result in discrepancy in seminally identical mathematical models. This particular difference will be demonstrated in comparison of the MNL results and the estimates obtained through estimation of a CNN model imitating MNL model. What is more, different models can provide different insights into the real world state. For example, MMNL model should account for heterogeneity in consumer preferences in the presence of random alternative specific effects.

The econometric models focused on inference and understanding of the underlying effects are usually estimated over the full dataset as there is no question about the precision of the obtained results, but rather the statistical power achieved in idenfication of the effects. We will follow the same approach in order not to face the different question related to external validity and verification of the estimated model, as well as the questions related to verification and testing of the models' performances over some external dataset.

In this part of the work we will firstly present the different estimation techniques, starting with *maximum likelihood* (**???**) estimator for the MNL, as well as it's algorithmic implementation within *R*, and the *Adam* algorithm (**???**) traditionally used to estimate the NN models. Afterwards, we will discuss the results of the estimations we obtain over the generated datasets, presented in the previous part.

### Maximum-Likelihood for MNL and MMNL

The MNL and MMNL models, both are estimated by the maximum likelihood method. In this technique the estimator is used to derive the parameters, which were the most likely to produce the observed results (observed dataset).

Assuming we face probabilities defined by some function $f(.)$ parametrized $\theta$, the joint probability density may be defined as:

$$\mathcal{L}(\theta) = \prod_{j}^{\Omega} P_i(j \mid \theta) \text{ with } \theta : max_\theta \mathcal{L}(\theta) \tag{1}$$

This function is also known as likelihood function. The log-likelihood is obtained through a *log* transformation of the likelihood function:

$$L(\theta) = \sum_{j}^{\Omega} log(P_i(j \mid \theta)) \text{ with } \theta : min_\theta L(\theta) \tag{2}$$

As we can see the obtained function is then minimised by adjusting $\theta$ in order to obtain the optimal parameters. The optimisation problem is non-linear and requires an implementation of some iterative technique to be solved. Under "general conditions" they are consistent, asymptotically efficient and asymptotically normally distributed (McFadden 2001).

Speaking about the algorithmic implementation within the statistical software, the optimization is performed by iteratively updating the vector of parameters by the amount given by *step* $\times$ *direction*. The *step* in this case is a positive scalar and the *direction* is given by:

$$D = H^{-1} \times g \tag{3}$$

Where $g$ represents the gradient, while $H^{-1}$ is an estimate of the inverse of the Hessian matrix.

In this procedure the main question is the choice and estimation procedure of $H^{-1}$, which has several possible definitions. For example, *Broyden–Fletcher–Goldfarb–Shanno (BFGS)* (**???**) algorithm may be implemented, which is an iterative method for solving unconstrained non-linear optimization problems. This algorithm updates $H^{-1}$ at each iteration using the variations of the vector of parameters and the gradient. The initial value of the matrix in this particular case is the inverse of the outer-product of the gradient. The initial step equals to 1 and, if the new value of the function is inferior to the previous value, it is divided by two, until a higher value is obtained. This iterative procedure stops when the gradient is sufficiently close to 0, which is achieved through comparison of the $g \times H^{-1} \times g$ product with the *tolerance* argument. An alternative stopping condition is achieved by introduction of the maximum number of iterations for the algorithm, which ensures the impossibility to fall into an eternal loop. We may summarise this algorithm as follows, as described in *mlogit* package documentation by (**???**):

1. The likelihood for the baseline model is calculated (assuming all the parameters are 0);
2. The function is then evaluated, assuming a step equals to one;
3. If the value of likelihood function is lower than the baseline value, the step is divided by two until the likelihood increases;
4. The gradient $g$ is then computed;

The authors of *mlogit* package insist that this method is more efficient than other functions available in *R* at this time. The codes used to estimate MNL model are available in Appendix (Appendix D.1).

For the MMNL model there exists an interesting modification for the algorithm, because we need to estimate the random effects variances and covariances in case of correlated random effects. The parameters are not directly introduced inside the likelihood function, but rather the elements of the Choleski decomposition of the covariance matrix are used. The Choleski decomposition matrix $L$ is defined in this case as follows:

$$L = \begin{bmatrix} chol_{11} & 0 & 0 & 0 \\ chol_{12} & chol_{22} & 0 & 0 \\ chol_{13} & chol_{23} & chol_{33} & 0 \\ chol_{14} & chol_{24} & chol_{34} & chol_{44} \end{bmatrix} \tag{4}$$

Where indices correspond to the random effects variables, for example, in our case study we have four parameters: *Buy* dummy variable, eco-*Label*, *Carbon* footprint and the $LC$, which stands for the *Label* and *Carbon* cross-product. Once the estimates of the matrix elements are obtained a variance-covariance matrix can be obtained:

$$LL^T = \begin{bmatrix} \sigma_1^2 & \sigma_{21} & \sigma_{31} & \sigma_{41} \\ \sigma_{12} & \sigma_2^2 & \sigma_{32} & \sigma_{42} \\ \sigma_{13} & \sigma_{23} & \sigma_3^2 & \sigma_{43} \\ \sigma_{14} & \sigma_{24} & \sigma_{34} & \sigma_4^2 \end{bmatrix} = \Sigma \tag{5}$$

Where $\sigma_i^2$ stands for the variance of effect $i$ and $\sigma_{ij}$ represents the covariance between two random parameters $i$ and $j$. The codes used to estimate MMNL model may be found in Appendix (Appendix D.2).

### Backpropagation algorithm for NN

For the estimation of the NN model we benefit from the flexibility offered by *Keras* (**???**), which is a high-level NN API developed with a focus on the speed of computation, offering at the same time an astonishing level of control over the models. The port of *Keras* inside *R* offered by (**???**) allows us to correctly specify our model, devised to imitate the structure of the traditional MNL. This particular ML library offers a choice of different model estimation algorithms, ranging from the *state-of-the-art* to the most recent and advanced techniques. In this particular application it was decided to implement the *Adam* algorithm (**???**), which can be considered as rather outdated estimation method by the standards of ML field, because it was introduced only in 2014.

*Adam* is an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments. This method was proved to be computationally efficient, as well as to have low memory requirements. It is invariant to diagonal rescaling of the gradients, and is well suited for problems that are large in terms of data or parameters. This algorithm is also considered appropriate for non-stationary objectives, as well as the problems with very sparse gradients. What is more, one of the particular advantage for us is that the hyper-parameters do not typically require advanced tuning.

Historically, the *Adam* algorithm is an extension to the *stochastic gradient descent* or *SGD* (**???**) method. The latter is an iterative method for optimizing a differentiable or sub-differentiable objective functions. It can be considered as a stochastic approximation of the *gradient descent* (GD) optimization, because it replaces the actual gradient, which is typically calculated from the entire data set, by an estimate, which is calculated from a randomly selected subset of the data. In high-dimensional optimization problems this technique reduces the computational complexity and hence the computation time, resulting in faster iterations. SGD has a single learning rate, denoted

$\alpha$ by convention, for all weight updates during training. The learning rate is considered to fixed through an entire estimation procedure as well. The two latter features, are sometimes regarded as disadvantage of the particular estimation technique and may not be suitable in all the contexts.

Once we have briefly presented its original predecessor, we may pass directly to *Adam* algorithm description as well as the procedures, which influenced its creation. The chosen method combines the advantages of two other extensions of SGD, which are:

- *Adaptive Gradient Algorithm* (AdaGrad), where the per-parameter learning rate is maintained fixed, which is suitable for sparse data learning problems (**???**);
- *Root Mean Square Propagation* (RMSProp), for which the per-parameter learning rates are adapted based on the average of recent values of the gradients for the weight. Tthis is an unpublished method supported by the community, more information may be found in (**???**).

This properties make the algorithm especially well performing on a non-stationary problems, including noisy data, as well as any other problem types. Instead of adapting the parameter learning rates based on the mean values (first moments) as in RMSProp, *Adam* uses of the average of the second moments of the gradients as well.

The *Adam* is configured using a following set of hyper-parameters (for more details on numerical implementation and working with *Keras* see Appendix D.3):

- *alpha*, which stands for the learning rate or step size, designing the proportion at which the weights are updated. Traditionally, as it was proposed by authors (**???**), the value of $\alpha$ equals to $1e-8$, but in our application we approach it to the values used in the DFGS algorithm, assuming that $\alpha = 1e-1 = 0.1$. Large values results in faster initial learning rate, before it is updated, while inferior values slow learning significantly and require more runs;
- $beta_1$, describes the exponential decay rate for the first moment estimates. We assume this value to be fixed to the defaults of *Keras*, which is $\beta_1 = 0.9$;
- $beta_2$ is the exponential decay rate for the second moment estimates, which is by default $\beta_2 = 0.999$;
- $\epsilon$ is the last hyper-parameter, which is a very small number to prevent any division by zero in the algorithm implementation. In *Keras* this value is $\epsilon = 1e-8$.

## Estimation results presentation

The comparison of the estimates obtained by the different models over different datasets can be done in two steps. First of all, we are interested in the observed mean effects over the datasets, because the possibility to correctly identify the means for the coefficients is of utmost importance for the analysis, regardless of the assumption on the heterogeneity of these effects. Then we are going to explore the additional dimension, provided by the MMNL estimates, which comprises the estimates for the variance-covariance matrix of the correlated random effects. The estimates obtained directly are the entries of the Choleski decomposition matrix and need to be transformed in order to observe the variances and covariances.

The results for the means estimates are regrouped in the table 1 on page 5. Now we can pass to the discussion of the obtained results and demonstrate the differences of the performances observed for different algorithms. We are going to start with the discussion of the estimates obtained with more traditional to econometric field MNL and MMNL models. Effectively, the MNL model allows us to obtain the exact estimates, due to the fast convergence rate and the relative simplicity of the problem. What is more, and what is of particular interest for us, it is how the MMNL model performs on the MNL specific dataset with fixed effects. The estimates obtained with the MMNL model for the fixed effects dataset demonstrate quasi-identical estimates as traditional MNL model, which nearly all the Choleski decomposition matrix element estimates statistically insignificant to zeros. Observing the estimates obtained from the two models we may rightfully conclude, that there is no evident danger in implementing a MMNL model in place of a MNL model on the fixed effects dataset, because the obtained estimates will point out the absence of the heterogeneous preferences in such case. The only disadvantage of the models misspecification in this case resides in the significantly increased estimation time, which requires significantly more iteration in order to estimate correctly the variance-covariance matrix elements and, consequently, the estimation complexity.

Table 1: Estimation results: mean effects

| | *Fixed effects* | | | *Random effects* | | | *Target* |
|---|---|---|---|---|---|---|---|
| | MNL | MMNL | CNN | MNL | MMNL | CNN | |
| **Characteristics** | | | | | | | |
| Sex | 1.401$^{***}$ | 1.400$^{***}$ | 1.369 | 0.712$^{***}$ | 1.297$^{***}$ | 0.719 | 1.420 |
| | (0.031) | (0.031) | | (0.016) | (0.024) | | |
| Age | 0.009$^{***}$ | 0.009$^{***}$ | 0.010 | 0.007$^{***}$ | 0.010$^{***}$ | 0.005 | 0.009 |
| | (0.001) | (0.001) | | (0.001) | (0.001) | | |
| Salary | 0.048$^{***}$ | 0.048$^{***}$ | 0.060 | 0.066$^{***}$ | 0.120$^{***}$ | 0.062 | 0.057 |
| | (0.010) | (0.010) | | (0.005) | (0.008) | | |
| Habit | 1.070$^{***}$ | 1.071$^{***}$ | 1.056 | 0.361$^{***}$ | 0.641$^{***}$ | 0.343 | 1.027 |
| | (0.030) | (0.030) | | (0.016) | (0.024) | | |
| **Attributes** | | | | | | | |
| Price | −1.626$^{***}$ | −1.628$^{***}$ | −1.618 | −0.886$^{***}$ | −1.586$^{***}$ | −0.886 | −1.631 |
| | (0.010) | (0.010) | | (0.006) | (0.010) | | |
| Buy | 2.311$^{***}$ | 2.313$^{***}$ | 2.228 | 0.662$^{***}$ | 2.180$^{***}$ | 0.665 | 2.285 |
| | (0.065) | (0.066) | | (0.036) | (0.054) | | |
| Label | 2.815$^{***}$ | 2.817$^{***}$ | 2.810 | 1.279$^{***}$ | 1.922$^{***}$ | 1.277 | 2.824 |
| | (0.022) | (0.022) | | (0.015) | (0.023) | | |
| Carbon | 6.654$^{***}$ | 6.662$^{***}$ | 6.634 | 3.259$^{***}$ | 5.430$^{***}$ | 3.250 | 6.665 |
| | (0.032) | (0.033) | | (0.016) | (0.030) | | |
| LC | −2.781$^{***}$ | −2.782$^{***}$ | −2.765 | −1.546$^{***}$ | −2.663$^{***}$ | −1.558 | −2.785 |
| | (0.028) | (0.028) | | (0.019) | (0.030) | | |

*Note:* $^{*}$p<0.1; $^{**}$p<0.05; $^{***}$p<0.01

On the contrary, in the case of presence of the correlated random effects in the preferences of the population the estimates are significantly biased for the MNL model. Moreover, the estimates obtained with the MMNL model are not identical to the input parameters, which were used during the simulation step. In this situation the MNL model tends to significantly underestimate the effects of all the characteristics and attributes for the choice situation. This can potentially lead to a notorious bias in case we were using incorrect model specification during a field experiment data exploration.

The results for the Choleski matrix entries estimates are regrouped into a single table 2 on page 6 Based on these estimates, we can comment as well the potential inefficiency of the implemented algorithm, even if it is one of the best available to us. Even though the estimates of the means obtained with MMNL in the presence of the random effects are close to the theoretical ones, the estimates of the variance-covariance matrix elements are rather close, but not perfectly calculated. Which is important, as we had a rather large dataset compared to the datasets typically collected during field studies: 1000 individuals with 10 replications of 16 choice sets situations for each totalling to 160000 choice situations. While in the original field study 102 individuals with only 2 replications of 6 choice sets were present, mounting to 1224 observations.

This situation demonstrates the existing trade-off between the need to correctly specify the model from the start and the potential computation inconveniences in the case of implementation of a more complex model in case of uncertainty. In other words, the scientists always face the choice either to simply use more complex model, which requires more data, calculation time and resources, or to perform an extensive theoretical study beforehand in order to correctly specify and delimit the model from the start.

Table 2: Estimation results: standard deviations and covariances

| | *Fixed effects* | *Random effects* | *Target* |
|---|---|---|---|
| | MMNL | MMNL | |
| **Standard deviations** | | | |
| Buy | 0.095 | 2.960$^{***}$ | 3.202 |
| | (0.061) | (0.028) | |
| Label | 0.031 | 2.687$^{***}$ | 2.654 |
| | (0.077) | (0.023) | |
| Carbon | 0.164$^{*}$ | 3.734$^{***}$ | 3.535 |
| | (0.076) | (0.026) | |
| LC | 0.145$^{*}$ | 2.851$^{***}$ | 2.711 |
| | (0.071) | (0.031) | |
| **Covariances** | | | |
| Buy:Label | −0.948 | −0.311$^{***}$ | −0.54 |
| | (5.116) | (0.026) | |
| Buy:Carbon | −0.886 | −0.565$^{***}$ | −4.39 |
| | (1.954) | (0.026) | |
| Label:Carbon | 0.891 | 0.959$^{***}$ | 8.77 |
| | (1.578) | (0.003) | |
| Buy:LC | 0.669 | 0.789$^{***}$ | 6.17 |
| | (0.501) | (0.005) | |
| Label:LC | −0.576 | −0.490$^{***}$ | −2.33 |
| | (4.423) | (0.032) | |
| Carbon:LC | −0.568 | −0.651$^{***}$ | −4.82 |
| | (1.604) | (0.030) | |

*Note:* $^{*}$p<0.1; $^{**}$p<0.05; $^{***}$p<0.01

Now we can switch to the discussion of the estimates obtained with *Adam* estimated CNN model, identical in structure to the MNL model. For reminder, we use convolution layers to calculate the relative deterministic utilities for the population $V_j$ for three alternatives, which are then converted to probabilities using a softmax dense layer with predefined unit weights for corresponding neurons.

As for the CNN estimates, the table 1 demonstrates, that the obtained estimates are technically identical to the means, we could see in the previous part for the MNL model estimates. These results demonstrate the flexibility of the NN models and the hypothetical possibility to implement them in place of traditional econometric models with only inconvenience being the relative complexity to obtain the variances for the weights estimates, as non known to us method allows this. This only inconvenience renders impossible to analyse the statistical significance for the obtained weight estimates, which can be seen only over the marginal effects graphs for particular variable on the probability, but this is other discussion's topic. For now, the most important part is that the CNN imitation of the MNL models, estimated with a high learning rate ($\alpha = 1e - 1$) *Adam* algorithm, allows to obtain correct estimates for the means of the theoretical utility function, assuming the variables were chosen correctly.

Because of the nature of the constructed CNN model latter performs similarly to the traditional MNL model. This situation implies that the proposed CNN algorithm is, identically to MNL model, unable to identify correct parameters and consequently derive the true means for the underlying coefficients of the relative utility function in the presence of heterogeneous preferences among individuals. Nevertheless, given the flexibility of the NN it is theoretically possible to device an algorithm imitating the MMNL model's behaviour or even propose some alternative modelling techniques which will be able to supply, not directly through estimated weights but rather after a supplementary study, the correct estimates for marginal effects of the attributes on the choice probabilities.

To summarise this section, we can underline the successful implementation of the chosen mathematical models over the artificially created datasets simulating different choice situations. The effects identified by all of the models are close to the target values, although there exists clear evidence that the MMNL models perform significantly better in mean effects identification in all the contexts. At the same time the MNL model and its synthetically recreated NN counterpart underestimate the coefficient of the given relative utility functions in presence of the correlated random parameters in the individual utilities.

McFadden, Daniel. 2001. "Economic Choices." *The American Economic Review* 91 (3). American Economic Association: 351–78. http://www.jstor.org/stable/2677869.