The Heterogeneous Agent Model
A Calibration Based on a Simulation

Barcelona GSE

Skander Garchi Casal

June 30, 2017

Acknowledgements

I would like to thank my advisor, Michael Creel, for guiding me over the past several months and expressing as much genuine interest in my project as I did myself.

Abstract

This paper aims to replicate Creels work in Neural Nets for Indirect Inference [3] by calibrating a heterogeneous agent model. He proposes to use neural networks to approximate the limited posterior mean, $E(\theta|Z_G)$, where θ is the parameter vector of the simulatable model and Z_G is a vector of statistics. Simulating and calibrating macroeconomic models can be a particularly difficult task. The goal of this thesis is to provide the tools necessary for an easy and fast calibration of a heterogeneous agent model. The use of neural nets will be essential to identify the mapping from the statistics space to the parameters space.

Introduction

Neural nets is a machine learning tool that has been used by data scientists and econometricians for many years. Nonetheless, even though neural nets have proven their utility and are extensively used in various areas, there are few authors who have attempted to use neural nets to address the kind of problems I face in this thesis. A few reasons come to mind and we can obviously talk about the relatively recent popularity of neural nets compared to other econometric tools.

My study begins with a few assumptions. Let θ be a draw of a parameter vector, from a prior distribution. We can simulate the model using the draw and generate a random sample of n observations Y_n, from which we can compute a vector of statistics $Z_G(Y_n)$. The purpose of this exercise is to reiterate the simulation as many times as it is possible. The following relationship can be established and estimated $\theta = E(\theta|Z_G)+\epsilon$, where $E(\epsilon) = 0$. The neural net is trained using the simulated data to learn about the limited information posterior mean, $E(\theta|Z_G)$. $\hat{E}(\theta|Z_G)$ will be the result of the identification of the mapping $\hat{E} : \Theta \mapsto \zeta$, where $\theta \in \Theta$ and $Z_G \in \zeta$.

Targeting the estimation of the limited information posterior mean $E(\theta|Z_G)$ does not appear to be a conventional approach to this problem; usually, people will tend to use all the information available by targeting the estimation of the full information posterior mean, $E(\theta|Y_n)$, where Y_n is the sample data. However, using the whole sample data instead of statistics, in this case, can be costly in terms of memory storage and computational time. Moreover, we observe that when the statistics are selected carefully they explain most of the variation of θ_s, where $s = 1, ..., S$ and S is the number of iterations in the simulation.

This exercise will provide the calibration of a macroeconomic model. However, the emphasis will not be on the economic interpretation of the results obtained but rather on the convenience and performances of this method as an econometric tool. The list of models that could have been explored is extensive; for instance, in Creel (2017) [3] the estimation of the limited posterior mean is targeted to solve a DSGE and a Moving Average Model. I will be working on the calibration of a heterogeneous agent model. The simulation will be performed on Matlab and the estimation by neural nets will be performed on Julia.

Literature Review

The current section provides a brief overview of the relevant literature in the field of neural networks for indirect inference.

Creel (2017) [3] is the main source that I have referenced. Instead of using the full sample as an input to the neural network, Creel uses a vector of statistics. This allows reducing the size of the input layer. Also, the convenience of this method is revealed when the sample changes. For instance, if the sample grows the number of statistics is fixed and the inputs do not have to be changed, however, the same feature does not hold when we use the full sample as input. Finally, the accuracy of the method was shown estimating a DSGE and a moving average model.

A similar proposal was made by Jiang et al. (2015) [4], in which neural nets were used to capture the full posterior mean. Contrary to the limited posterior mean, the full posterior mean uses the full sample rather than a set of statistics. Targeting the full posterior mean clearly provides more accurate results, however, a trade-off between computational time and accuracy has to be made. In my case, following Jiang et al. (2015) paper was not a possibility, for reasons that I will expose in a later section.

LeCun and al. Efficient BackProp [7] and Chapter 4 of Simon Haykin - Neural Networks" [6]. A Comprehensive Foundation contributes to the literature by improving the performance of the back-propagation algorithm, which is explained briefly in one of the following chapters. Designing a neural net using the back-propagation algorithm requires one to make many arbitrary choices that will often depend on the problem being faced. LeCun and al.

highlight some aspects of the theory that can be used as a guide to making better choices in the design of the back-propagation.

Michael Reiters course on Computational Methods contributed largely to the method by which I performed my simulation. The model that I calibrated uses as a base some of the functions that Reiter created [9].The estimation by neural net uses Creels code that is available at [2]. It uses the package Mocha.jl for the Julia language.

Multilayer Perceptron

An artificial neural network is a system that is based on the biological neural network, such as the brain. The multilayer perceptrons are an important class of neural networks. It is composed of an input layer, one or more hidden layers of neurons and one output layer of neurons. The multilayer perceptron is commonly known for a forward propagation of the input signal layer by layer.

For the following explanation, I describe a neural net with two hidden layers, $H1$ and $H2$, as I will be using the same layer configuration later. In this section, I will introduce some notation. Also, to be consistent with the introduction, let Z_G be the input layer as a vector of dimension $(G, 1)$, θ be the desired output as a vector of dimension $(K, 1)$, and O be the output layer that is of the same size as the desired output.For simplicity, I will let G = 3 and K = 2, 3 statistics and 2 parameters. Each hidden layer has a certain number of neurons that have to be determined by the user, $H_j = (H_{1,j}, H_{2,j}, ..., H_{i,j}, ..., H_{n_j,j})$, where n_j is the number of neurons in the hidden layer j. Also, I inroduce three vectors of weights that will be used in the explanation. $W_{j-1}(k)$ is the vector of weights assigned to the input layer or the hidden layer $j-1$ to send a signal to neuron k of the layer j, $W_{j-1}(k) = (w_{1,j-1}(k), w_{2,j-1}(k), ..., w_{n_{j-1},j-1}(k))$, $j = 1, 2, 3$, and k is the neuron from the layer j that is connected to layer $j-1$ and $n_0 = G = 3$. This notation will help to describe the process by which the forward pass is performed. Finally, the activation function will be defined as f(.) and it is a non-linear function. This property makes it more suitable to estimate non-linear mappings. The choice of the activation function is discussed in LeCun et al (2012) and I will be using $tanh(.)$.

The first step for the forward pass is to initialize the weights by randomizing between 0 and 1 the vectors of weights. The neurons of the hidden layers are computed as follow:

$$H_{1,1} = f(\begin{pmatrix} 1 & Z'_G \end{pmatrix} \begin{pmatrix} \beta_0(1) \\ W_0(1) \end{pmatrix}) = f(\beta_0(1) + \sum_{i=1}^{G} w_{i,0}(1)),$$

where $\beta_0(1)$ is called the bias. The neural net can have another input known as bias; the bias would be the equivalent of an intercept in a linear regression. The same principal is applied for $H_{k,1}$ $\forall k = 1, ..., n_1$. This way we can get the vector H_1 and use it as input for the second hidden layer.

$$H_{k,2} = f(\begin{pmatrix} 1 & H'_1 \end{pmatrix} \begin{pmatrix} \beta_1(k) \\ W_1(k) \end{pmatrix}) \ \forall k = 1, ..., n_2$$

we obtain H_2 the same way as we did for H_1. Finally the elements of the output layer will be such that:

$$O_k = f(\begin{pmatrix} 1 & H'_2 \end{pmatrix} \begin{pmatrix} \beta_2(k) \\ W_2(k) \end{pmatrix}) \ \forall k = 1, ..., K.$$

The first pass ends comparing θ to O. Given that the weights we choose were completely random, we need to be able to change them optimally given a defined criterion, the error $\epsilon = \theta - O$. One of the most well-known algorithms in machine learning that contribute to optimally change the weights is the back-propagation algorithm.

Back Propagation and Batches

This section will introduce the concepts of Back Propagation and Batch Learning, as they are important to the understanding of neural nets and explain why the parameters estimation performs so well. LeCun and al. present the concepts in detail and provide tricks that are rarely exposed in serious technical publications. The back-propagation algorithm consists in propagating backward the error that is computed in the forward pass, ϵ. Training a network using Backprop requires making many arbitrary choices such as the number of layers or the number of neurons by layer. The BackProp is a gradient- based learning method, which signifies that changes in the weights of the net that we described before will evolve according to the changes in the error ϵ. As for the Ordinary Least Squared the function that is minimized here is the mean squared error.

$E_t = \frac{1}{K} \sum\limits_{k=1}^{K} \epsilon_k^2.$

From this criterion, a law of motion for the weights can be determined. The most simple learning procedure in such a setting is the gradient descent algorithm where W, vector of weights, is adjusted as follow:

$(*)\ W(t) = W(t-1) - \eta \frac{\partial E_t}{\partial W},$

where t determines the pass position and η can be a scalar or a diagonal matrix that calibrates the speed of convergence of $(*)$. The purpose of the BackProp is to find $W(t)$ such that $d(W(t), W(t-1)) < e$; where $e > 0$ and d is a distance measure. We see here the importance of the choice in parameter η that will determine the convergence or the divergence of the BackProp algorithm.

The data that I simulate will be divided into a training set, in which weights will be chosen optimally and a test set that is used to evaluate the performances of the estimation. The training set is divided into sub-samples called batches. This practice has two main advantages. It is important to perform the stochastic gradient descent on batches and not on the data as a whole. If the data is truly redundant, it is more than likely to result in similar gradients for all the batches. Therefore, it is a wasted effort to compute the gradient on the whole data set and it would be best to compute the gradient on a subset of the data and update the weights, then use these new weights as input for the next batch. Finding the weights for the next batches will be almost instantaneous, as I began with an excellent first guess, which is the weight for the first batch. This practice also allows us to avoid converging to local extremums by jumping from a region of the parameter space to another.

Data

The Survey of Consumer Finance published by the Federal Reserve, is a cross- sectional survey of U.S. families. Participation in the study is voluntary; 6,500 families were interviewed for the last survey. For this reason, it is important to be able to address selection biases. The Federal Reserve addresses this problem by selecting randomly families among the respondents. For the calibration of the model only one cross-section needed. The 2013 survey is used for this exercise; from the sample, a few variables were gathered that helped constructing important data by relying on equilibrium conditions or laws of motion. The following variables were used from the survey:
weights, total income, wage income, saving, asset, equity and return on equity. From these variables, I computed the statistics I needed for the calibration.

Methodology

The model is inspired by Aiyagari (1994) [1]. It solves for the invariant distribution of capital in an economy. Agents are ex ante identical but they face different idiosyncratic shocks that will affect their labor income. Markets are incomplete and the heterogeneity can not be insured; therefore, agents are ex post different. The only way to insure against the future is by using precautionary saving by holding capital. There is a borrowing constraint such that, $k_{i,t+1} > 0$. There are competitive firms producing a consumption good. The available technology uses capital and labor and uses constant returns to scale. The firm's optimal decision is such that:
$r_t = \alpha K_t^{\alpha-1} L_t^{1-\alpha}$
$w_t = (1-\alpha) K_t^{\alpha} L_t^{-\alpha}$
Here the consumer can choose among a grid of capital for his or her optimal saving choice and choose his or her consumption in the same way; $gridk = k_1, k_2, ..., k_{100}$. The consumer's wage evolves according to a Markovian process $w(z) = w(t)z$, where z follows a finite-state Markov process of persistence λ and standard deviation σ. I assigned to z a grid of seven possible states, $gridz = z_1, z_2, ..., z_7$, and get the transition probability matrix Π, with $\Pi_{i,j}$ being the probability of switching from state z_i to z_j. Therefore, the consumer solves the following bellman equation:
$(**)\ v(k, z_i) = \max_{k'} u(k, k', z_i) + \beta \sum_{l=1}^{7} \Pi_{i,l} v(k', z_l)$
where $u(c_t) = \frac{c_t^{1-\gamma}}{(1-\gamma)}$

My work was divided into two steps. First, I simulated on Matlab the statistics given a draw for θ, and I then finished using the package MXNet on Julia in order to estimate the $E(\theta|Z_G)$ with functions that Creel provides at [2]. The main function is called *sendstat*. It takes as input the number of endogenous states, the number of exogenous states, and the number of iterations. The output is the simulated data. In *sendstat.m* I initialize my parameters and for each iteration, I solve for the invariant distribution of capital given our draw in the parameter space. To be precise, the following is included in θ.

$$\theta = \begin{pmatrix} \alpha \\ \lambda \\ \sigma \\ \gamma \end{pmatrix},$$

where $\alpha \sim U(0.01, 0.99)$ is the share of capital in the production of the good, $\lambda \sim U(0.7, 1)$ is the persistence of the Markovian process, $\sigma \sim U(0.5, 2)$ is the standard deviation of the Markovian process, and $\gamma \sim U(0.01, 0.99)$ is the coefficient of relative risk aversion. *sendstat.m* uses the equilibrium condition and search for the value function using the function *consaccel.m*. *consaccel.m* provides as output a structure containing among other things the value function and the optimal choice of savings given the state variables.

It is important to know which procedure I used to compute the invariant distribution in this economy, which is exactly what we are looking for. For this we need to combine endogenous choices with exogenous states and get the transition matrix, to do so let me introduce some notation.

Let $\Phi(j)$ be the transition matrix for the endogenous choice given state j, $\Phi(j)_{i,k} = 1$ when k_k is the optimal choice given that we previously saved k_i in state j, $\Phi(j)_{i,k} = 0$ otherwise.

From the function *consaccel.m* we get the optimal choice of the agent in each state, therefore, we can easily build $\Phi(j)$. Lets construct Tendog and Texog such that:

$$Tendog = \begin{pmatrix} \Phi(1)' & 0 & \cdots & 0 \\ 0 & \Phi(2)' & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \Phi(7)' \end{pmatrix}, \text{ with } dim(Tendog) = (700, 700)$$

$$Texog = \begin{pmatrix} \Pi_{1,1} I_{100} & \Pi_{2,1} I_{100} & \cdots & \Pi_{7,1} I_{100} \\ \Pi_{1,2} I_{100} & \Pi_{2,2} I_{100} & \cdots & \Pi_{7,2} I_{100} \\ \vdots & \vdots & \vdots & \vdots \\ \Pi_{1,7} I_{100} & \Pi_{2,7} I_{100} & \cdots & \Pi_{7,7} I_{100} \end{pmatrix} = Kron(\Pi', I_{100})$$

The final transition matrix is $T = Texog.Texog$

$$T = \begin{pmatrix} \Pi_{1,1}\Phi(1)' & \Pi_{2,1}\Phi(1)' & \cdots & \Pi_{7,1}\Phi(1)' \\ \Pi_{1,2}\Phi(2)' & \Pi_{2,2}\Phi(2)' & \cdots & \Pi_{7,2}\Phi(2)' \\ \vdots & \vdots & \vdots & \vdots \\ \Pi_{1,7}\Phi(7)' & \Pi_{2,7}\Phi(7)' & \cdots & \Pi_{7,7}\Phi(7)' \end{pmatrix}$$

Finally we obtain T of dimension $(700, 700)$ that will be computed by the routine *transmat.m*. It is important when we have the transition matrix to compute the invariant distribution, there is different methods available, I used the following one in my code *invdistr.m*. I solve for the system $P_{t+1} = T.P_t$ at the steady state. Therefore, I solved for P^* such that $P^* = T.P^*$. This is equivalent to look for the eigen vector of matrix T associated to the eigen value one. $dim(P^*) = (700, 1)$ we reshape it as a matrix $(100, 7)$ and get the invariant distribution. From that point a set of statistics is computed using the invariant distribution and a few other equilibrium conditions. I simulated the model 101000 times with different values for the parameters and started the estimation by neural nets.

The statistics I use for the training and the calibration are the following ones. The 9 first deciles of capital level $[Kd1, ..., Kd9]$, the mass of capital located in the last grid level $Kmass$, aggregate consumption AgC, agregate investment AgI, aggregate output AgY, consumption's standard deviation

C_s, investment's standard deviation I_s, mean consumption over output $\frac{\bar{C}}{Y}$, mean capital over output $\frac{\bar{K}}{Y}$, standard deviation of consumption over output $\frac{C}{Y}(s)$, standard deviation of capital over output $\frac{K}{Y}(s)$, correlation between consumption and capital $corr(C, K)$, correlation between capital and wage $corr(K, w)$, correlation between consumption and wage $corr(C, w)$, log of capital over labor $log(\frac{K}{L})$.

Empirical Results

This section will be dedicated to explaining the process of choosing the statistics, evaluating the performances of different nets and presenting the results of the calibration.

The set of statistics that I presented in the previous section were selected following rigorous and specific criteria. First, every statistic that is chosen for the simulated data has to be chosen for the real world data in order to be able to perform the calibration. Therefore, some statistics can be eliminated from the sample if there is no way to construct them in the real world data. It is also important to make the difference between statistics that carry very little information and informative statistics; to do so I proceed to small experiments. Simulating 101,000 observations take 2 days, however, simulating 1000 observations takes, in general, a little less than one hour. In order to save some time, I performed many small experiments like this one in order to identify a combination of statistics that seems to carry a large part of the information available and that can be constructed with the data available in the Survey of Consumer Finance (2013).

Figure 2 shows the weights that a 2-layer neural net with 100 and 50 neurons respectively attributes to each statistic. The net identifies the statistics that carry less information by attributing to them lower weights. We observe that statistics 1 to 17 have relatively small weights compared to statistics 18 to 23 with weights that are two to three times bigger. Then, $\frac{\bar{C}}{Y}(s)$, $\frac{\bar{K}}{Y}(s)$, $corr(C,K)$, $corr(K,w)$, $corr(C,w)$, and $log(\frac{K}{L})$, are the most relevant statistics that I found to explain the parameters α , λ , σ , and γ

.λ and σ are parameters of the Markovian process, I believe that they are well explained by the standard deviation of consumption and the standard deviation of capital. Indeed, the more uncertain the economy, the higher the inequality and therefore the dispersion in capital and consumption can be expected to be higher as well. γ is a parameter related to the preferences of the consumer, therefore, the two previous statistics can explain it, but I also suspect $corr(C, K)$, $corr(K, w)$, $corr(C, w)$, to be relevant statistics in the estimation of this parameter. α , share of capital in the production function, is mostly explained by $log(\frac{K}{L})$, as we can observe the link between them looking at the following equilibrium condition.
$\frac{K}{L} = (\frac{r}{\alpha})^{\frac{1}{1-\alpha}} \leftrightarrow log(\frac{K}{L}) = \frac{1}{1-\alpha} log(\frac{r}{\alpha})$.
The performances of the estimation are outstanding; the R^2 are higher than 80% for all of the parameters and fluctuations of the parameter α are explained at 99.7%. However, even if these results appear encouraging, they are nevertheless conditional on the model being true.

The last step is to check for the robustness of our net. I came to understand that a 2-layer net with 100 and 50 neurons is one of the best configurations available. To my knowledge, there is no test available to check for the robustness of the net specification. However, I studied the results under different specification of the net and based my reasoning on the mean squared error, the R^2 and the computational time to determine the best specification. Among all the candidates considered, the most relevant ones are:
$[100, 10]$, $[100, 50]$, $[500, 50]$and $[30, 5]$, $[100, 100]$and $[500, 250]$
After studying the results from Table 1 and Table 2, we can identify two outstanding neural net specifications $[100, 50]$ and $[500, 50]$. The choice between the two will depend on the preferences of the user. If estimating σ is a priority, then choosing $[500, 50]$ will yield more accurate results. However, if estimating γ with accuracy is the goal of the study, then using the specification $[100, 50]$ may be more useful. In this case, I decided to work with $[100, 50]$, because of the computational time that is divided by two compared to specification $[500, 50]$. Discernibly, computational time is not a significant

problem for this project; the training only takes a few minutes. However, if I complicate the model by adding more parameters or more layers, the computational time becomes a relevant criterion. Finally, $[100, 50]$ specification ranks statistic 24 as having one of the highest weights. As aforementioned, statistic 24 comes from an equilibrium condition and is one of the few statistics that shows a proportional relationship with one of the parameters. Therefore, it is important to keep the weight of that statistic high.

After constructing the statistics for the Survey of Consumer Finance (2013), I decided to select 98% of the population. The 2% are excluded from the sample for the following reason: when I selected the whole sample from the S.C.F., my calibration gave me results that were in favor of an economy with very low uncertainty and low persistence of individual shocks. I realized after manipulating the data that the wealthiest 2% were radically changing the invariant distribution of capital. With the whole sample, we conclude that individual shocks have no significant impact on the agents behavior. They do not use precautionary saving as much as we would expect. The reason for that behavior is that 2% of the agents in this economy hold much more wealth than the rest of the individuals. Therefore, when we compare their saving to the rest of the population it seems that the 98%have no desire to hold capital, which is not the case. One solution to this problem might be to consider a model that takes into account heritage, for instance, an Overlapping Generation model. The results of the calibration are available in Table 3.

Figures 7,8, and 9 gather invariant distributions and value functions. Again, these figures only account for 98% of the population. Figure 7 represents the Unconditional Invariant Distribution of capital under the calibrated parameters. The results appear intuitive: a large part of the population is located in the first capital levels and even though there is one spike in level 90 and a mass of agents in the last level I can explain it with choices I made during the calibration. The high levels of capital (level 85 to level 100) concern approximately 3% of the population. The other element that gives me confidence regarding the accuracy of the calibration is that the neural net identified the mass of consumers that were located in the last level of capital. When I gathered 98% of the population for the calibration, I also added a mass of 2% in the last level of capital, representing the 2% richest that I excluded from my sample. Figures 8 and 9 represent the conditional value functions and invariant distributions. The results are relatively standard. Under a high state, we value wealth less than when we are in a low state. Finally, one of the last results of the calibration is the distribution of the wage income in the economy. As I explained previously, heterogeneity in this model is carried by wages. Table 4 shows how different are supposed to be wages in the US. We see that between the lowest state and the highest state, wages are multiplied 1438.3.

The results I obtained show how well the neural net performed. Nonetheless, there is an issue to be noted: some of the parameter (λ,γ) are out of their parameter space. The training of the net was performed with statistics generated by parameters in the parameter space. There are a few possible explanations. The first explanation is that I was maybe too restrictive on the boundaries of the prior distribution and if that is the reason it would be appropriate to reiterate the simulation with different prior distributions. However, I believe that the prior distributions I set were not particularly restrictive, therefore, I think that the model as I presented it here does not capture all the features of the economy. It might be interesting to calibrate an Aiyagri model [1] with Overlapping Generations, for example. Finally, the most convincing explanation should be that I didnt relax the borrowing constraints in my model and had to proceed with some modifications while constructing the statistics in the real world data that might have led to a bias in the estimation of the parameters.

Conclusion

This master thesis aims to use neural nets to approximate the limited posterior mean $E(\theta|Z_G)$ as proposed in Creel (2017) [3]. As observed, targeting the limited posterior mean instead of the full information posterior mean $E(\theta|Y_n)$ltively uncommon approach. However, the convenience of this method was also discussed. Estimating the limited posterior mean is feasible at low cost whereas targeting the full information posterior mean is extremely costly in terms of computational time and computers memory. Also, it has been established from previous papers that the targeting of $E(\theta|Z_G)$ using the right set of statistics usually performs as well as targeting $E(\theta|Y_n)$.

The use of the neural nets was a essential to identifying the non-linear mapping from the statistics space to the parameters space. As I explained in the section Multilayer Perceptron the value of a neuron is determined by the summation of neurons in the previous layer that is transformed by an activation function. The non-linearity of that activation function helps to learn more easily about that mapping.

The calibration of the model began with a simulation. Given a prior belief that I hold for the distribution of the parameters that we aim to calibrate, I simulate the model, identify paths of consumption and capital, and compute the statistics. The next step is to learn about the mapping from the statistics space to the parameters space and apply this to the real world data to find the results of the calibration.

The methods results are satisfying, but it is important to perform a few changes to the model and see how the results of the calibration change. As

I mentioned in the previous section, some of the parameters are out of the parameter space. It would be interesting to see which of the aforementioned reasons isbefore is the cause of this occurrence..

Figures and Tables

	Net 100_10	Net 100_50	Net 500_50	Net 30_5	Net 100_100	Net 500_250
α	0.996	0.997	0.999	0.990	0.997	0.997
λ	0.837	0.842	0.827	0.806	0.838	0.844
σ	0.814	0.833	0.851	0.779	0.772	0.836
γ	0.886	0.905	0.892	0.850	0.881	0.903

Table 1: Net Performences R^2

	Net 100_10	Net 100_50	Net 500_50	Net 30_5	Net 100_100	Net 500_250
α	0.000	0.000	0.000	0.001	0.000	0.000
λ	0.001	0.001	0.001	0.001	0.001	0.001
σ	0.035	0.031	0.028	0.042	0.043	0.031
γ	0.009	0.007	0.008	0.012	0.009	0.007

Table 2: Net Performences MSE

	α	λ	σ	γ
Calibration	0.233241	0.618915	1.42781	1.29711

Table 3: Calibration

	$S(\omega_1)$	$S(\omega_2)$	$S(\omega_3)$	$S(\omega_4)$	$S(\omega_5)$	$S(\omega_6)$	$S(\omega_7)$
$WageIncome$	1	3.4	11.3	37.9	127.4	428.1	1438.3

Table 4: Wage income relative to the smallest wage

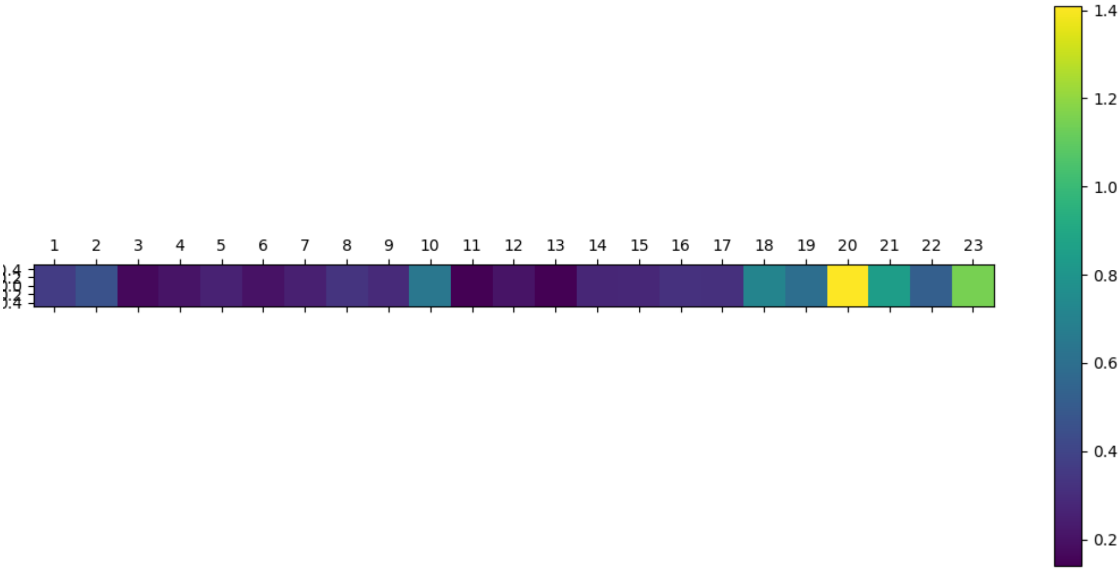

Figure 1: Weights of the Net 100_10

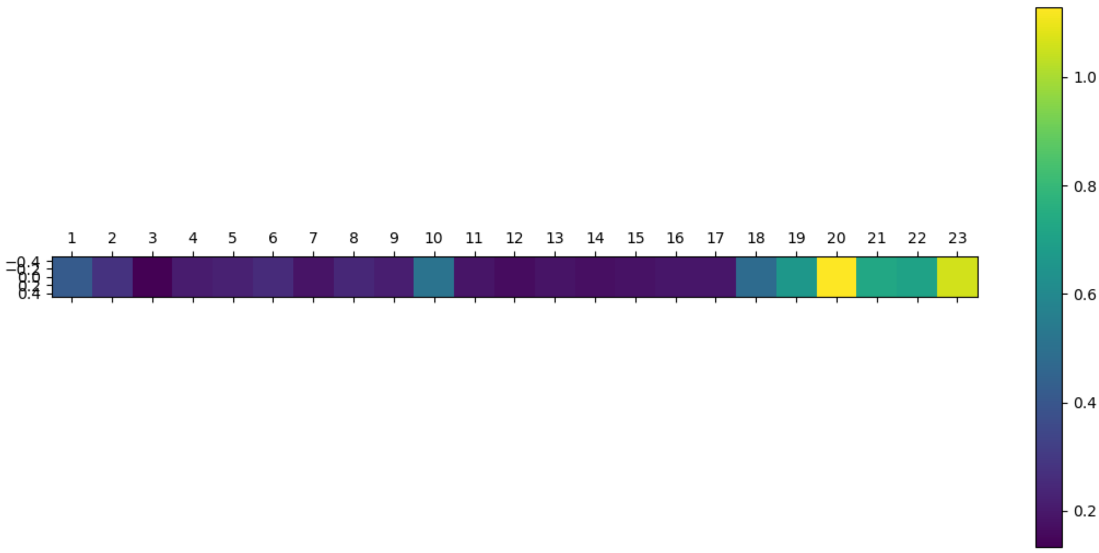

Figure 2: Weights of the Net 100_50

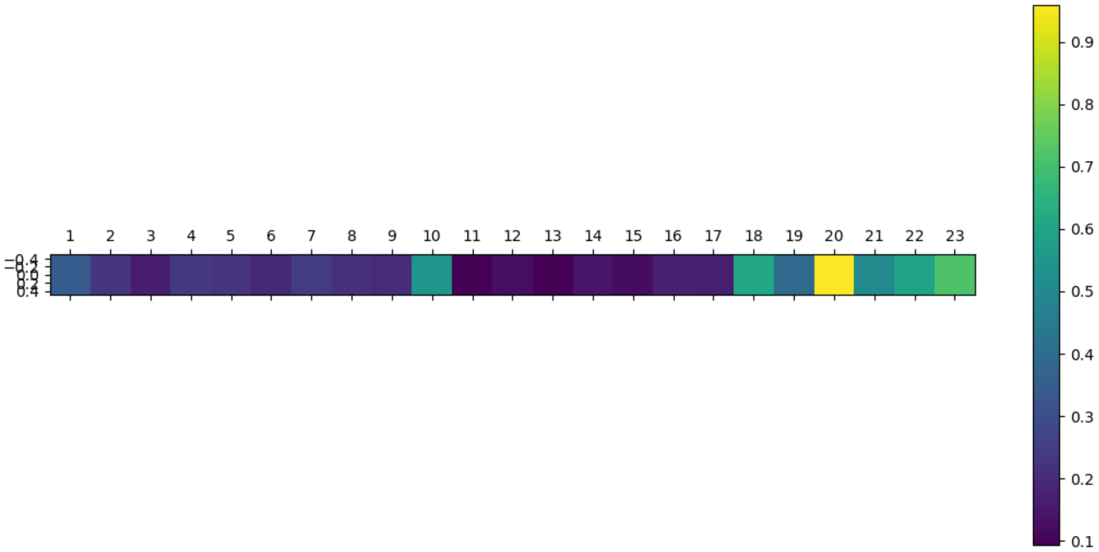

Figure 3: Weights of the Net 500_50

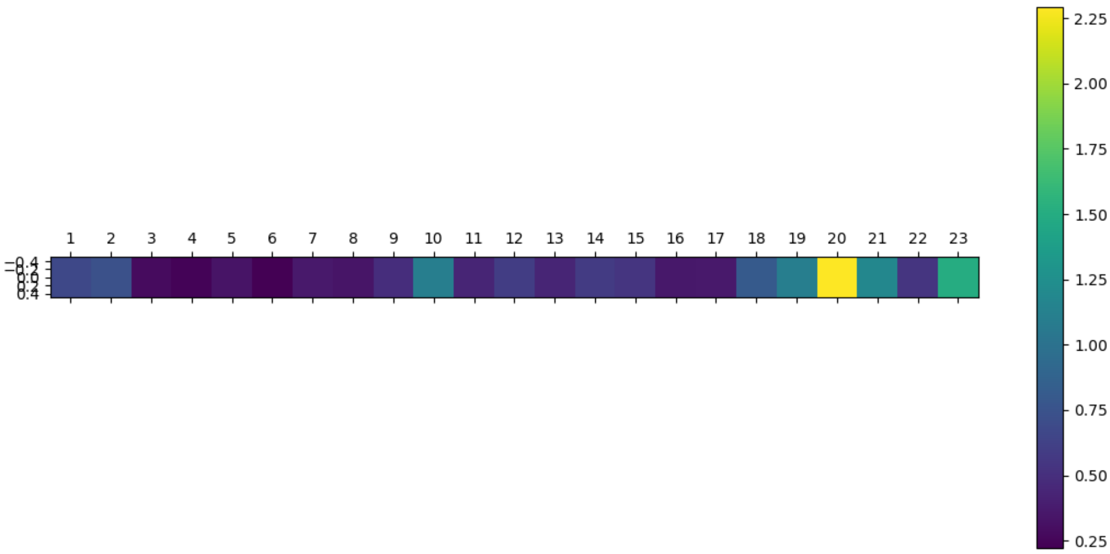

Figure 4: Weights of the Net 30_5

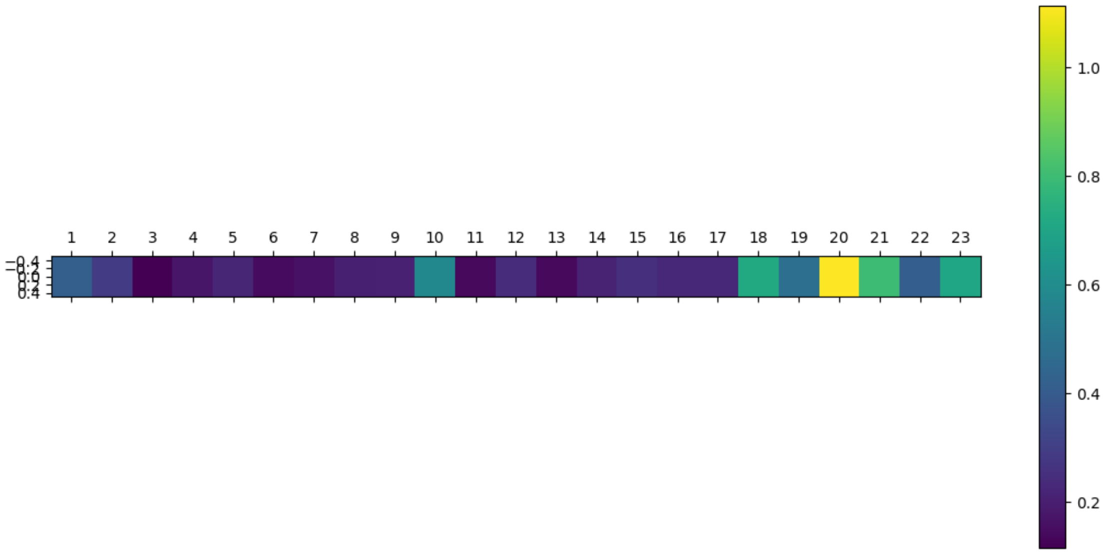

Figure 5: Weights of the Net 100_100

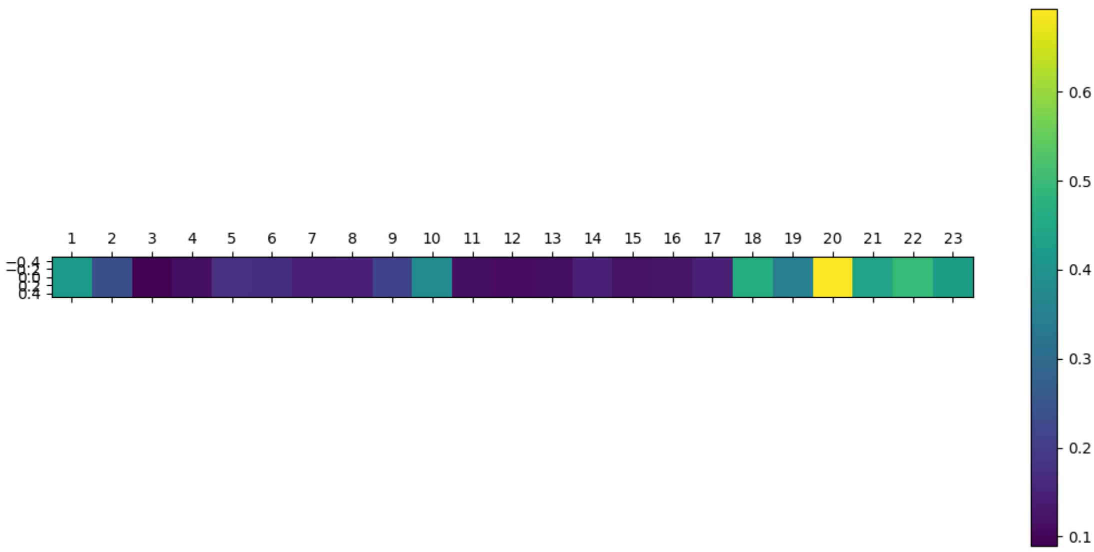

Figure 6: Weights of the Net 500_250

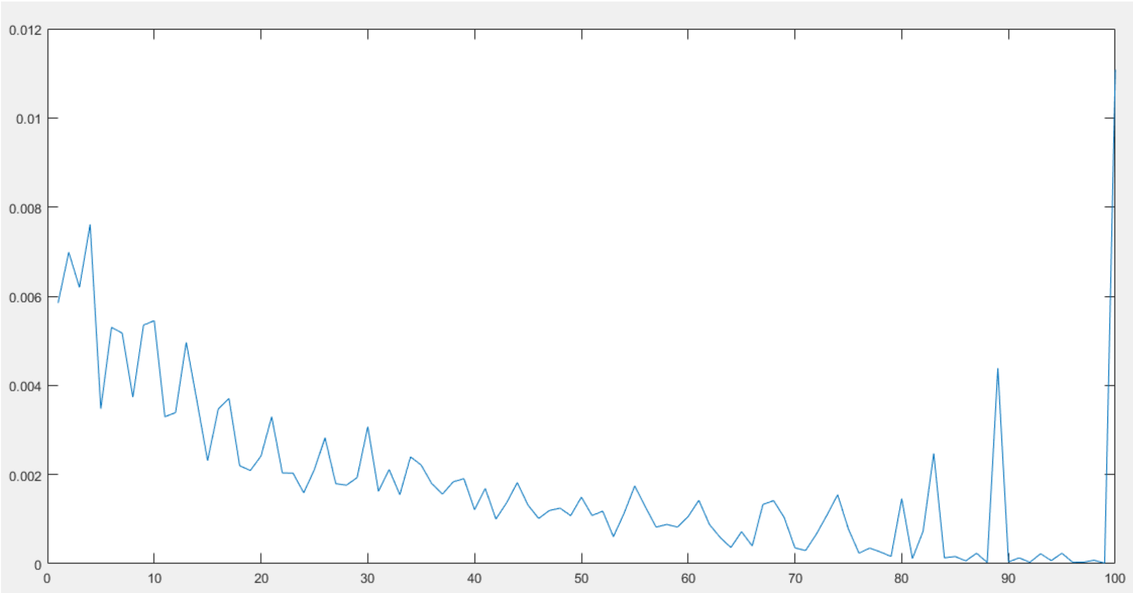

Figure 7: Unconditional Invariant Distribution

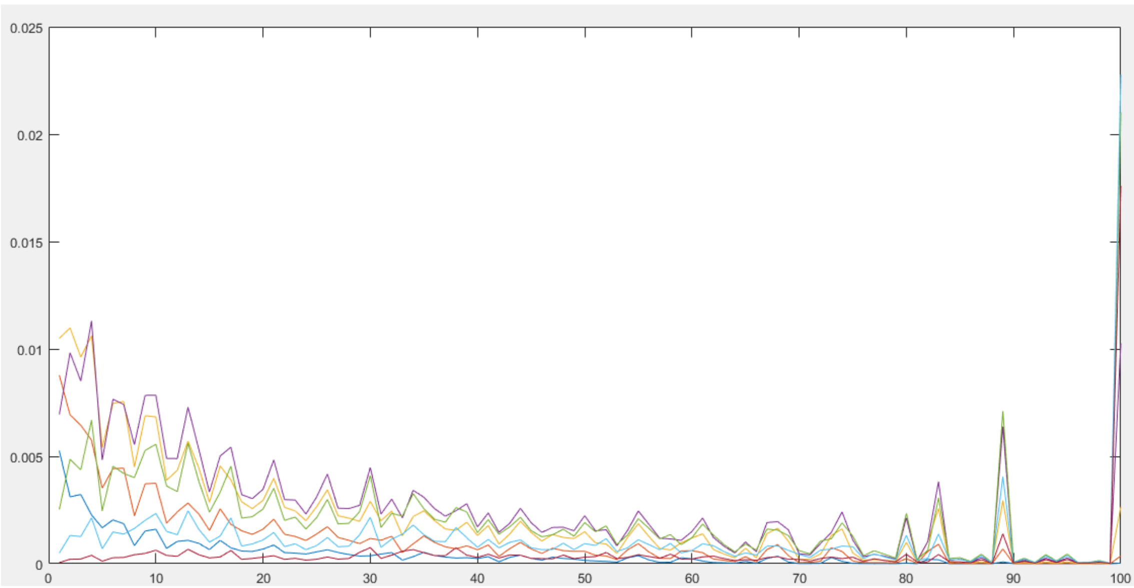

Figure 8: Conditional Invariant Distribution

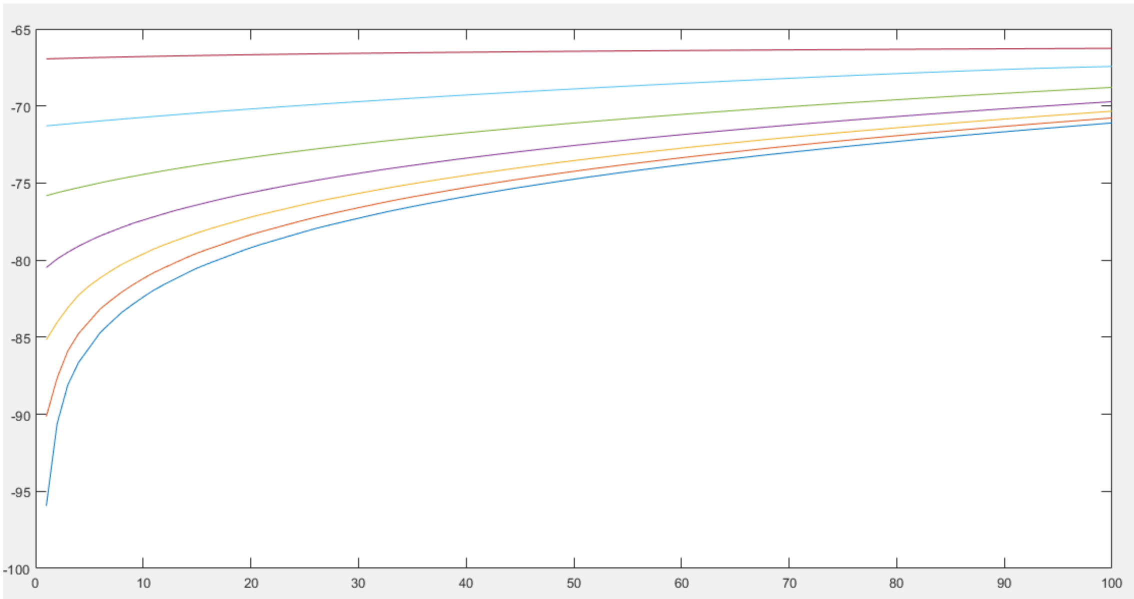

Figure 9: Conditional Value Function

Bibliography

[1] S Rao Aiyagari. Uninsured idiosyncratic risk and aggregate saving. *The Quarterly Journal of Economics*, 109(3):659–684, 1994.

[2] Michael Creel. Neural Nets For Indirect Inference. `https://github.com/mcreel/NeuralNetsForIndirectInference.jl`.

[3] Michael Creel. Neural nets for indirect inference. *Econometrics and Statistics*, 2:36–49, 2017.

[4] Bai Jiang, Tung-yu Wu, Charles Zheng, and Wing H Wong. Learning summary statistic for approximate bayesian computation via deep neural network. *arXiv preprint arXiv:1510.02175*, 2015.

[5] Chung-Ming Kuan and Halbert White. Artificial neural networks: an econometric perspective. *Econometric reviews*, 13(1):1–91, 1994.

[6] Miroslav Kubat. Neural networks: a comprehensive foundation by simon haykin, macmillan, 1994, isbn 0-02-352781-7. *The Knowledge Engineering Review*, 13(4):409–412, 1999.

[7] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

[8] Survey of Consumer Finance. Survey of Consumer Finance. `https://www.federalreserve.gov/econres/scfindex.htm`, 2013.

[9] Michael Reiter. Computational Methods. https://elaine.ihs.ac.at/~mreiter/cmbgse.html.