UNIVERSITY OF COPENHAGEN
DEPARTMENT OF ECONOMICS

Master Thesis: Economics

by Jeppe Søndergaard Johansen *(pcv439)*

Deep Reinforcement Learning and Dynamic Models

A Case Study of the Relationship Between Female Labour Supply and Fertility

Supervisor Thomas Høgsholm Jørgensen
Submission: 31.05.2020
ECTS: 30.0
Strokes: app. 135.000

Deep Reinforcement Learning and Dynamic Models

A Case Study of the Relationship Between Female Labour Supply and Fertility

Jeppe Søndergaard Johansen (pcv439)

May 2020

Abstract

This paper investigates reinforcement learning as a solution method for dynamic models. A discrete time, finite horizon, discrete choice model of female labour supply and fertility is formulated, and the model is solved using value function iteration and two reinforcement learning methods, namely deep Q-learning and double deep Q-learning. After estimating the model using method of simulated moments, and finding the simple model inadequate to describe real data from Statistic Denmark, the model is extended. The extension consists of 14 + 1 states. The extenden model is solved only by double deep Q-learning and estimated using simulated method of moments. The results of the extended model convincingly matches data from Statistics Denmark and contemporary findings by Kleven, Landais, and Søgaard (2019). I conclude that the field of economics should further investigate reinforcement learning, as it allows solving dynamic models with high dimensional state space.

Contents

1 Introduction

The last 10 years have lead to numerous breakthroughs within the field of machine learning. The sub field reinforcement learning has been especially prolific. In 2013 the company DeepMind achieved super human performance in various Atari games (Mnih et al. 2013). In 2018 the same company beat professional human players in the board game Go, a feat which was not deemed possible in a foreseeable future (Silver et al. 2018), and as late as 2019 DeepMind showed that a reinforcement learning agent was able to play the game of Star Craft 2 on the same level as the best human players (Vinyals et al. 2019). All the games mentioned can be considered dynamic models. Dynamic models play a central role within the field of economics. The results imply a new way of solving dynamic economic models using reinforcement learning.

Dynamical models usually involve agents that take sequential actions, trying to maximize the cumulative utility throughout the lifecycle. This class of economic models conform to a set of properties economists like: They have a micro foundation; agents are utility maximizing. They model time, allowing for agents to foresee the future, and act in accordance to their expectations. Some dynamic models can be solved analytically. This is true for the canonical Ramsey model. However, when the scope of the model grows, different approaches are necessary. Dynamic programming is usually the tool utilized for solving such models. Even though dynamic programming is a flexible tool, it does have its limitations. Solving a model using dynamic programming, requires a limited sized state space, otherwise the computation involved becomes infeasible. In practice this leaves high dimensional dynamic models impossible to solve using contemporary techniques. Deep reinforcement learning allows for solving such models. Because these techniques are relatively novel, they have not yet been introduced into the field of economics. Deep reinforcement learning, does unfortunately not guarantee that the solution converges to the global maximum, but results have shown that learning is possible in hard, high dimensional environments.

This paper uses the aforementioned techniques to investigate the effect of children on female labour supply. Inspired by the model specification of Francesconi (2002) and Adda, Dustmann, and Stevens (2011) I formulate a discrete time, finite horizon model that models discreet female labour supply and its relationship to fertility. First a simple model is formulated, where women can choose the number of supplied hours, letting fertility be exogenous, with an income process following the Mincer equation of human capital. The husband of the household, is assumed to follow a deterministic path regarding the number of hours supplied to the labour force, and the wage rates they receive. Households are assumed to face a budget constraint, that neither allow for borrowing or saving. Utility is assumed to be a function of leisure and consumption, and children are assumed to reduce leisure by mirroring additional work for the woman, that is not financially compensated. Later an extension to the original model is presented with exogenous education combined with a transfer system for women in the education system. Additionally, the extension tracks children on an individual level.

Using three different solution methods (value function iteration, deep Q-learning and double deep Q-learning) I solve the simple model. I show that one can get comparable performance using deep reinforcement learning when compared to using value function iteration solution methods, yielding a new way to solve more complex dynamic models. The parameters of the Mincer equation are calibrated using data from Statistics Denmark. The model is estimated using method of simulated moments, where a simple grid search approach is applied due to fact the optimization problem being one dimensional. Only double deep Q-learning is used to solve the extended model. Again the model is estimated using method of simulated moments and grid search. The data used for the optimization is from Statistics Denmark.

My two main findings are: 1) Deep reinforcement learning can yield comparable performance to value function iteration solution methods. Considering this allows for solving dynamic models with high dimensional state space, I argue these methods should be explored further in the field of economics. 2) Simulating from the estimated model, I find the initial simple model is not able fit the data, whereas the extended model does fit the data surprisingly well. Both participation rates and average number of supplied hours to labour force, are surprisingly close to what the data from Statistics Denmark suggest. Comparing to Kleven, Landais, and Søgaard (2019), this paper finds results very similar regarding earnings, participation rate, supplied hours to the labour force and wage rates, when a woman gives birth to a child.

The paper follows the structure: A literature review is conducted highlighting the main findings and articles of endogenous female labour supply and the effect of children. A model is formulated based on key takeaways from the literature. The parameters of the income process is calibrated using data from Statistics Denmark. Next, I introduce the reader to both reinforcement learning and deep learning. Settling on three different solution methods I solve and estimate the model. I go on to extend the model, and solve it using double deep Q-Learning. I end by comparing the results to contemporary findings, and data from Statistics Denmark.

2 Review of Literature

A rich literature on both labour supply and fertility exists. As this paper tries to investigate the relationship between women's labour supply and fertility by formulating a dynamic structural model, the primary focus will be on literature that has the same scope. The main inspiration for this dissertation has been the paper by Francesconi (2002), where he proposes a joint dynamic model of fertility and labour supply of married women. Labour supply is split into three choices: *not work*, *work part-time* and *work full-time*. Furthermore, Francesconi proposes human capital accumulation to be a function supplied labour. Additionally, he assumes a budget constraint of all income in period t should be consumed in period t, and the husband's income and labour supply is modelled to follow an exogenous process. The main findings of

the paper is that a relationship exists between earnings ability and preference for work and women with the highest earnings profiles has the lowest marginal utility of children.

Work has also been done to extend the basic framework proposed by Francesconi (2002), that models the joint decision between fertility and labour supply. Adda, Dustmann, and Stevens (2011) extends the lifecycle model by allowing for savings, and more sophisticated skill atrophy processes, Gayle and Miller (2006) allows for the participation rate of women to be continuous and Keane and Wolpin (2010) focuses on the marriage market while still allowing for fertility and labour supply being part of the choice set. Adda, Dustmann, and Stevens (2011) suggests that fertility might be falling in developed countries due to significant costs to the careers and future earnings of women associated with child birth. They find that the cost of career interruptions as a consequence of children is non-linear over the career cycle, and it has the biggest impact around mid-career. They also find that children influence career planning even before the first child is born. Gayle and Miller (2006) employ a semi-parametric approach to a panel data setting. Inline with Francesconi (2002), Gayle and Miller (2006) finds that having children is less desirable for women on high income trajectories. Keane and Wolpin (2010) finds that differences in skill rental price between black and white women can explain the number of teenage pregnancies, again implying that a relationship between fertility and career trajectories is present.

The joint decision between fertility and labour supply has been investigated before Francesconi (2002) proposed his dynamic structural model. Both Moffitt (1984) and Hotz and Miller (1988) utilizes a closed form approach to the problem of fertility and labour supply to estimate an econometric model. Moffitt (1984) employs a cross-sectional approach. While controlling for the number of children he finds that more children in general imply women work less. Hotz and Miller (1988) makes the interesting addition to their model, that fertility is only somewhat a choice of the household, i.e., with some probability the fertility outcome will diverge from the household's expectation. They find that the maternal time required for a newborn child is 660 hours per year, which equates to 12.7 hours per week, decreasing geometrically as the child ages. They also find that children do not reduce women's labour supply 1-to-1, rather the time spent on children will be taken from other activities as well.

Not all research has lead to the same unambiguous conclusion that children lead to a reduction in the labour supply of women. Angrist and Evans (1996) investigates the causal link (using IV-estimation) between fertility and labour supply, and finds that fertility only have a small effect on the labour supply of women. Additionally, they find the effect seems to disappear as the child turns 13. Altuğ and Miller (1998) employs a semi-parametric approach to estimate the effect of experience on female labour supply while controlling for children. They find that the impact of children on women's participation rate is ambiguous, but for married women, additional children increases number of hours worked.

Much of the literature on female labour assumes an income constraint, such that households are not allowed

to consume what is not yet earned. Attanasio, Low, and Sánchez-Marcos (2008) proposes a structural model, that investigates the extensive margin of women that work, with the addition to their model of letting households borrow, separating them from Francesconi (2002) among other earlier studies. Human capital is accumulated when women enter the work force. They investigate 3 cohorts to see the what can account for the different participation rates observed. Their main finding is that some of the differences observed in the cohorts, can be explained by reduced child-care costs and a reduction in the wage gender gap. Attanasio has investigated female labour supply further. Attanasio, Levell, et al. (2018) formulates a model where they investigate the participation rate of women. In this formulation they control for family composition and importantly include "taste-shifters" in the utility function. These are latent variables that is used to explain how the participation rate can change under different circumstances. Their main finding is that heterogeneity of demographics, wealth distribution and the point of the business cycle can explain a lot of the aggregate responses observed in female labour supply.

The impact of institutions and benefits that support mothers is also investigated in the literature. Del Boca, Pasqua, and Pronzato (2009) uses cross-country European data to investigate the joint decision about fertility and labour market participation. They control for personal characteristics as well as country-specific childcare systems, parental leave systems, family allowances as well as part-time opportunities. Their main findings include that the labour market and social environment do not affect fertility significantly. They do however find that institutions that can support women's labour market participation does indeed have an impact on women's labour market participation. This effect is the most pronounced in less educated women. These results are somewhat in line with Haan and Wrohlich (2009) that investigates the impact of financial incentives on female labour supply by exploiting the variance stemming from the tax and transfer system. They also find that child care subsidies increase labour supply. In contrast with Del Boca, Pasqua, and Pronzato (2009) they do find child care subsidies to increase fertility as well.

This paper solely focuses on women in relationships. It is not unreasonable to think, that differences between single and married women exists regarding fertility and labour supply. Blundell et al. (2016) uses a quasi-experiment focusing on the UK tax and welfare reforms of the 1990s and 2000s to investigate the labour supply of women. They construct a dynamic model where women can save and accumulate human capital, along with education. They make the women choose education level and their participation rate in the labour market. In contrast with Francesconi (2002) they do not let fertility be part of the choice space, rather they model it as a random event. They control for demographics etc. They find that labour supply elasticities are on average high (but below 1), except for single mother that seem to have above 1. Another interesting result from the paper is that tax credits do seem to let low-education women into the workforce, however it does not seem to have long-term influence on employment or wages for this group. Eckstein and Lifshitz (2011) also investigates the discrepancy between lone mothers and married couples by constructing a dynamic lifecycle model, their motivation being that while the labour supply of women

the last 50 years have had a sharp rising trend, the same cannot be said for unmarried women[1]. The authors conclude that the rise in female employment in large part can be explained by the increase in years of schooling and the rise of female wages. They also find that changes in fertility do not have a big impact.

Briefly discussing results in a Danish context. Using Danish data Kleven, Landais, and Søgaard (2019) find that the long-term effects of the of a child reduces earnings by about 20 %, and the hours worked by 10% for women, while no effects are notable for men (10 year horizon). The same goes for participation rates which fall about 13 % and wage rates which falls about 9 % in the long-run for women (10 year horizon). Jørgensen (2017) investigates the relationship between consumption of non-durable goods and the average number of children, since these follows the same trajectory over the life cycle. He finds that income of the households fall, but in contrast with Kleven, Landais, and Søgaard (2019) the economic effect is negligible with a reduction around 1% in households with one child compared to households with no children.

Considering that children can have an effect on the labour force participation of women one might ask why? This can be framed in two different ways: women want to spend more time with their kids because it is a more joyful activity. The second option is that some of the time children take up can be considered work, implying less leisure time for women, which causes women to work less. Firestone and Shelton (1988) Investigates that latter hypothesis and find that women will in general lose about 3 hours of leisure per week for each child. This falls in line with Thrane (2000) that finds that 0.31 hours per day of leisure is lost which on a weekly basis would accumulate to 2.2 hours a week. Ekert-Jaffé and Grossbard (2015) finds that leisure time falls at about half an hour per day yielding 3.5 five hours of leisure foregone per week. However, for children below the age of three about 1.4 hours for women in foregone leisure time is observed, leading to about 10 hours of lost leisure time for women per week. These numbers corresponds to those found by Hotz and Miller (1988) which, as mentioned earlier, is about 12.7 hours per week, falling geometrically.

3 Model Specification

This paper presents a discrete time, finite horizon, discrete choice model of female labour supply. More specifically I model a household consisting of wife, husband and a zero or more children. The model attempts to address what the effect of children is on the labour supply of women. The model consists of 3 components: 1) An income and human capital component, 2) A fertility process, 3) A leisure and utility component. I model the households from age $Q_{min} = 18$ to the terminal age $Q_{max} = 60$. Here it should be noted, that I make the simplifying assumption that the husband and wife has the same age and that the

[1] Married women has a gone from 30% to 60% employment, where single and divorced women have been at around 70% employment throughout the sample.

couple is married from age $Q = 18$. The age evolves in a deterministic fashion, i.e. the household grows one year older for each step in the model:

$$Q_{t+1} = Q_t + 1 \tag{1}$$

For each time step in the model, the agent has to choose how many hours the woman of the household should supply on a weekly basis. The choice is discreet, and consists of four action values: $H_t \in \{0, 25, 37, 45\}$. The labour supply of the man in the households is not a choice variable and is therefore considered an exogenous variable.

The households has two income streams, the husband, which is perfectly deterministic, and the wife. The income process of the wife consists of an idiosyncratic component Z_t, which follows a random walk:

$$Z_{t+1} = Z_t + \epsilon_t, \qquad \epsilon_t \sim \mathcal{N}(0, \sigma_\epsilon) \tag{2}$$

This allows for agents to do display heterogeneity owing to the fact that some unobserved carrier choices will lead to higher wages, even though two otherwise identical agents have been part of the labour force equally long. Since these job characteristics is unobserved, it is assumed to be a random walk. The second component of the income process is a human capital component based on the Mincer equation, as described by Lemieux (2006). That is the the log-transformed wage rate/wage level can be described by the human capital accumulated:

$$\log \tilde{W}_t = \alpha + \eta_G G_t + \eta_{G^2} G_t^2 \tag{3}$$

A couple of things to note. In the original formulation by Lemieux (2006), the education level is also included. However, due to the lack of availability of such data I am not able to condition on this. It should also be noted, that the state G_t represents the human capital accumulated of the woman in the household. Finally this equation governs only the wage rage of the women (not of the husband), for whom the wage rate and the supplied number of hours is considered exogenous. The exogenous wage rate of the husband is found using the data set **LONS50** from Statistics Denmark. And the number of supplied hours for the husband is found using the data set **LIGEF15**, again supplied by Statistics Denmark. The wage rate of the women will be capped at the minimum wage if the sum of the two components $\tilde{W}_t + Z_t$ is not above the the minimum wage $W_{min} = 120$:

$$W_t = \max(W_{min}, \tilde{W} + Z_t) \tag{4}$$

The human capital accumulation process, follows a formulation allowing for depreciation (or skill atrophy):

$$G_{t+1} = G_t(1-\delta) + \frac{1}{37}H_t \tag{5}$$

Where 37 being the standard number of hours worked in Denmark. The total income Y_t of the household can now be formulated as:

$$Y_t = 46 \cdot W_t \cdot H_t + f^M(Q_t) \tag{6}$$

Where f^M represents the income from the husband as a function of age. And $46 \cdot W_t \cdot H_t$ is the number of supplied hours pr. week, H_t, times the number of weeks, 46, in a year for the average person on the labour market[2] times the wage rate, W_t. The income process is a function of the number of supplied working hours, H_t, and the states (Q_t, Z_t, G_t). The parameters $\delta, \sigma_\epsilon, \eta_G, \eta_{G^2}$ will be calibrated in section 4.

The second component of the model is the fertility process. The fertility is assumed to be exogenous depending on the age of the woman. This is summed up in the equation below:

$$K_{t+1} = K_t + \psi_t, \qquad \psi_t \mid Q_t \sim Bernoulli(p_\psi(Q_t)) \tag{7}$$

K_t is the number of children in the household. The household is assumed to start with $K_t = 0$ at age $Q = 18$. At each step with probability $p_\psi(Q_t)$ the wife gives birth to a child. Allowing for the accumulation of children. The number of children is capped at a maximum of 5 in the model. The probability $p_\psi(Q_t)$ is modelled using data from Statistics Denmark using the data set **FOD33**. The number of children K_t is part of the state space.

The third component of the model is the utility and leisure component. The agent is assumed to get utility from leisure, L_t, and consumption, Y_t. Following Francesconi (2002) the households are assumed to face a budget constraint such that all income of period t must be consumed period t.The utility U_t is given by:

$$U_t = \beta_L \ln(L_t + 1) + \beta_Y \ln(Y_t + 1) \tag{8}$$

Following the formulation of Adda, Dustmann, and Stevens (2011), dividing the utility into sub-utility functions, where each sub-utility function allows for curvature by specifying a constant relative risk-averse (CRRA) function for each sub-utility. Assuming the special case of $\ln(\cdot)$. The parameters β_L, β_Y is the individual weighing of the different sub-utilities. Note that for identification I will restrict $\beta_Y = 1$. The total number of hours leisure the agent receives in a year follows:

[2]Assuming 6 weeks of holiday.

$$L_t = 46\left((24 \cdot 7) - \omega \cdot K_t - H_t\right) \tag{9}$$

Following Firestone and Shelton (1988), Thrane (2000) and Ekert-Jaffé and Grossbard (2015) I assume that some of the time spent with children can be considered work. The number of hours spent on children each week is captured by $-\omega \cdot K_t$. I let $\omega = 3.5$ be the time spent of extra house work pr. child each week. This number is taken from Ekert-Jaffé and Grossbard (2015). The weekly number of hours supplied to the labour market H_t is also subtracted from the total amount of leisure. The number of hours is aggregated to annual level subtracting 6 weeks for holiday. To conclude β_L will be a parameter estimated to give the best fit of the model

Summarizing the model; the model contains 4 states: (G) human capital, (Z) the idiosyncratic wage path, (K) the number of children in the household and lastly (Q) age. The action taken in each period (H) represents the number of hours the woman supplies to the labour market on a weekly basis. Other important variables are: (W) the wage rate, $(\tilde{W})$ the human capital dependent wage rate, (U) the utility and (L) leisure. Formally this imply:

$$\textbf{State space: } \mathcal{S} = \mathbb{R}^2 \times \{0, 1, 2, 3, 4, 5\} \times \{18, 19, \cdots, 60\} \tag{10}$$

$$\textbf{Action space: } \mathcal{A} = \{0, 15, 25, 37, 45\} \tag{11}$$

$$\textbf{States: } \{G, Z, K, Q\}, \qquad \textbf{Actions: } \{H\} \tag{12}$$

The model furthermore contains the following parameters: $\alpha, \eta_G, \eta_{G^2}, \delta, \sigma_\epsilon, \beta_L, \beta_Y = 1, W_{min} = 120, \omega = 3.5$. Where the parameters governing the income process $(\alpha, \eta_G, \eta_{G^2}, \delta, \sigma_\epsilon)$ will be calibrated using a simple agent based model, and β_L will be estimated using the full model. The recursive formulation of the model is given below:

$$U_t(L_t, Y_t) = \beta_L \ln(L_t + 1) + \beta_Y \ln(Y_t + 1) \tag{13}$$

$$L_t(K_t, H_t) = 46 \cdot \left((24 \cdot 7) - \omega \cdot K_t - H_t\right) \tag{14}$$

$$\log \tilde{W}_t(G_t) = \alpha + \eta_G G_t + \eta_{G^2} G_t^2 \tag{15}$$

$$W_t(\tilde{W}_t, Z_t) = \max(W_{min}, \tilde{W}_t + Z_t) \tag{16}$$

$$Y_t(Q_t, H_t, W_t) = 46 \cdot H_t \cdot W_t + f^M(Q_t) \tag{17}$$

$$\tag{18}$$

Law of motion:

$$Q_{t+1}(Q_t) = Q_t \tag{19}$$

$$K_{t+1}(K_t, Q_t) = K_t + \psi_t, \qquad \psi_t \mid Q_t \sim Bernoulli(p_\psi(Q_t)) \tag{20}$$

$$Z_{t+1}(Z_t) = Z_t + \epsilon_t, \qquad \epsilon_t \sim \mathcal{N}(0, \sigma_\epsilon) \tag{21}$$

$$G_{t+1}(G_t) = G_t(1-\delta) + \frac{1}{37}H_t \tag{22}$$

$$\tag{23}$$

4 Calibrating the Parameters of the Mincer Equation

To solve the model some reasonable values for the parameters of the wage process is necessary. The wage process contains the following parameters: $(\eta_G, \eta_{G^2}, \delta, \alpha, \sigma_\epsilon)$. As mentioned in the model specification in section 3 the wage process follows the Mincer earnings equation, not accounting for education, and where the idiosyncratic wage path is added linearly to the wage. I assume that the parameters driving the wage process are the same for both men and women, and these can be by calibrated independent of the entire model. This is due to computational constraint. Friedman, Hastie, and Tibshirani (2001) argues that calibrating multiple parameters at the same time suffers from the *curse of dimensionality*. Another more important reason is that the wage of the husband in the model is assumed perfectly deterministic, which imply I am not able to calibrate the parameters driving the wage path under the assumption these are the same for men and women. As mentioned in the model specification, the idiosyncratic wage path is added linearly. This allows for a two-step calibration of the parameters. First calibrate $(\eta_G, \eta_{G^2}, \delta, \alpha)$ that drives the age and sex specific expected wage rate (referred to as *wage path*). Second the scale of the random walk can be calibrated by σ_ϵ (referred to as *wage variance*). In other words the parameter calibration of the income process is broken into two phases: First calibrating age and sex specific expected wage rate, second calibrating the variance of the wages. It is important to note that this is agent based modelling, where the agent is supplied with a predetermined course of action dependent on the state!

I calibrate the wage path by using a **LONS50**, a data set from Statistics Denmark (Danmarks Statistik Bank). This data set contains the wage trend for men and women at any given age. My objective is to minimize the squared distance between the empirical wage path and the simulated wage path. The simulated wage path is constructed by simulating from the partial model with given parameters and taking the average. Certain things should be noted about this partial model. The wage path is a function of human capital which again is a function of the choice variable H of the model specified in section 3. This obviously makes it problematic to calibrate the parameters. I work around the problem by using the data

set **LIGEF15** containing the number of worked hours for both men and women, which is supplied by Statistics Denmark. These numbers do not take into account people leaving the labour force temporarily, which women are known to do when giving birth. I make the assumption that women leave the work force for 1 year when giving birth to a child. I use the data set **FOD33** to get fertility rates of women. Again this data is supplied by Statistics Denmark. Formally this can summed up in the policy described below:

$$H_t = \begin{cases} H^{men}(Q) & \text{if sex=}\textit{male} \\ H^{women}(Q) & \text{if sex=}\textit{female} \text{ and birth=}\textit{false} \\ 0 & \text{if sex=}\textit{female} \text{ and birth=}\textit{true} \end{cases} \tag{24}$$

The rest of the wage process follows the model described in section 3. The minimization problem follows the following process:

$$\text{Cohort Average: } \tilde{\mu}_i(C_i) = \frac{1}{\mid C_i \mid} \sum_{\tilde{w}_{n,q} \in C_i} \tilde{w}_{n,q} \tag{25}$$

$$(\hat{\eta_G}, \hat{\eta_{G^2}}, \hat{\delta}, \hat{\alpha}) = \underset{\eta_G, \eta_{G^2}, \delta, \alpha}{\arg\min} \frac{1}{2} \frac{1}{\mid C \mid} \sum_{C_i \in C} \left((\mu_i^{men}(C_i) - \tilde{\mu}_i^{men}(C_i))^2 + (\mu_i^{women}(C_i) - \tilde{\mu}_i^{women}(C_i))^2 \right) \tag{26}$$

Where $\tilde{w}_{n,q}$ denotes a simulated wage rate at age q for the n'th simulated individual and w_q is the true value wage rate for a given age. $C_i \in C$ denotes a given age cohort, where the age cohorts are $C = \{(20, 24), (25, 29), \cdots, (55, 59)\}$. μ_i denotes the empirical average for a given cohort. I solve the minimization problem using Nelder-Mead (Simplex method). Essentially The simplex method allows for numerical optimization without the need to supply neither the gradient, Hessian or Jacobian matrix. I initialize the algorithm with the values: $(\alpha = 4, \eta_G = 0.5, \eta_{G^2} = 0.01, \delta = 0.5)$, and let the algorithm run for a maximum of 100 iterations.

Given the now calibrated parameters driving the wage path, only a single parameter σ_ϵ needs to be calibrated. This is again done by numerical optimization. By minimizing the mean squared error between the empirical quartiles (upper and lower) for men and women, to those found by simulating, the optimal scale for the random walk, σ_ϵ, is found:

$$\text{Cohort Quartile: } \tilde{\omega}_{i,j}(C_i) = \text{quartile}_j(C_i), \qquad j \in \text{upper}, \text{lower} \tag{27}$$

$$\hat{\sigma_\epsilon} = \underset{\sigma_\epsilon}{\arg\min} \frac{1}{2} \frac{1}{2} \frac{1}{\mid C \mid} \sum_j \sum_{sex} \sum_{C_i} \left(\omega_{i,j}^{sex}(C_i) - \tilde{\omega}_{i,j}^{sex}(C_i) \right)^2, \tag{28}$$

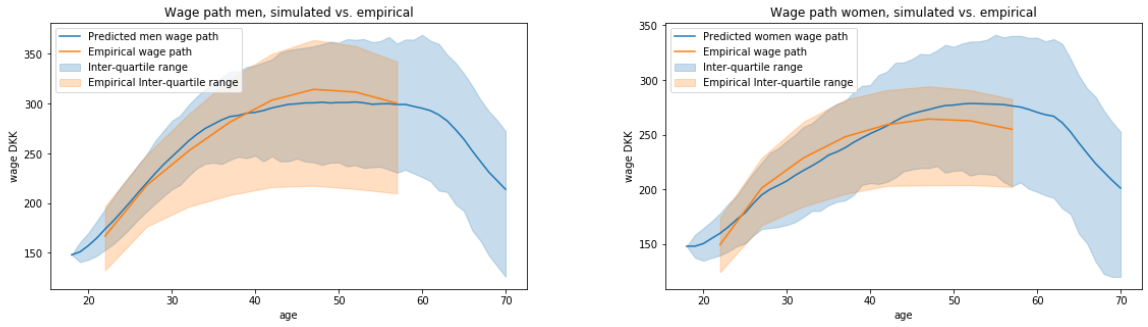
Where equation (28) assumes $C_i \in C$, $j \in \{\text{upper}, \text{lower}\}$, and $sex \in \{\text{men}, \text{women}\}$. Again I use Nelder-Mead for optimization, set a max number of iterations of 100, and set the starting value of $\sigma_\epsilon = 0.5$. The results of the optimized parameters are listed in table 1:

Table 1: Optimized parameters for wage process

	$\hat{\alpha}$	$\hat{\eta_G}$	$\hat{\eta_{G^2}}$	$\hat{\delta}$	$\hat{\sigma_\epsilon}$
Parameters	4.609	0.164	0.015	0.209	15.11

The parameters seem to have reasonable values comparing to other values found in the literature. First $\hat{\alpha}$ is the constant in the wage equation. The Mincer equation is log transformed implying an $\exp(\cdot)$ transformation is required for evaluating the value. Assuming no human capital, $G = 0$, yields $\exp(\hat{\alpha}) = \exp(4.609) \approx 100$. Or in other words, in this model, a totally inexperienced worker would receive an hourly wage of 100 DKK (not far from the actual minimum wage). Looking to δ the rate of human capital depreciation/skill atrophy, I find a value of approximately 20% per year. Compared to other studies this does seem a bit on the high side, however not unreasonable. Kunze (2002) finds that parental leave reduces wages for women with about 13 to 18% per year, while other work interruptions is about 2 to 5% a year. Light and Ureta (1995) finds values for human capital depreciation at about 13% a year. Considering the other parameters, they are harder to compare to other literature due to denomination in DKK.

Figure 1: Simulated wage process vs Empirical wage process

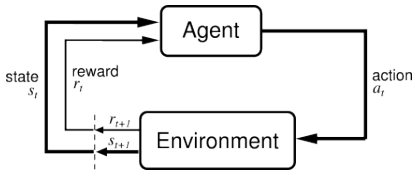


(a) Men

(b) Women

Figure 1 compares the empirical wage process with the simulated. First looking to the *wage path*, I conclude that the Mincer equation seems to give a relatively good fit to the data even without taking education into account. The mean absolute error of the women wage path of women is 10.61 DKK, and for men the number is 5.84 DKK, which considering the relatively simple parametric form of the process, is very good. Figure 1 also show the empirical wage process is skewed. In this formulation of the mincer equation this is not possible. The width of the inter-quartile range seem to fit the wage path of men reasonably, but

Figure 2: Agent/Environment Interface



for women it slightly overshoots. I conclude the parameter values of the Mincer equation do seem to yield reasonable results, and they will be used throughout the paper. A side note about the empirical wage path of men and women is that men do seem to experience a greater variance in their wage rates than women. Considering the work of both Francesconi (2002) and Gayle and Miller (2006) that suggests that women on high income trajectories finds it less desirable to work, this is a puzzling result. The literature suggests that children could be one of the main drivers of heterogeneity observed in the wage rates of women, so when men have even higher wage rate differences, and these are not driven by children, one must conclude that other, probably equally important factors drive wage rate heterogeneity.

5 Reinforcement Learning

5.1 The Environment/Agent Interface

In the real world learning happens by trial and error. Reinforcement learning is an attempt to formalize this study of "learning by interaction". The problem of learning by trial and error has a natural formulation by the environment/agent interface, which is graphically represented in figure 2. Note, this entire section draws heavily from the book Reinforcement Learning by Sutton and Barto (2018), regarding naming conventions, algorithms and notation.

The concept can be phrased the following way: An agent will over a series of discreet time steps ($t = 1, 2, \cdots, T$) take an action which will lead to some sort of reward (or in economic terms utility). For each time step the agent receives information about the state S_t of the environment $\mathcal{E}$. The agent then takes an action A_t, which prompts the environment $\mathcal{E}$ to return a reward R_{t+1}, and a new state S_{t+1}. This process continues until the game terminates. The agent's sole purpose is to maximize the cumulative rewards throughout the game. It should be noted here that $R_t \in \mathbb{R}$ such that the agent is optimizing over a sum of scalars. The game will lead to a trajectory of states, actions and rewards that look like:

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, \cdots, R_{T-1}, S_{T-1}, A_{T-1}, R_T \tag{29}$$

Certain assumptions is necessary to perform any sort of modelling. The most fundamental assumption

reinforcement learning relies on is that of the Markov decision process: A Markov decision process (MDP) abides to:

$$p(s_t, r_t \mid s_{t-1}, a_{t-1}) = p(s_t, r_t \mid s_{t-1}, \cdots, s_0, a_{t-1}, \cdots, a_0) = P(S_t = s_t, R_t = r_t \mid S_{t-1} = s_t, A_{t-1} = a_t) \tag{30}$$

Breaking equation (30) down reveals that the MDP follows a true probability distribution. That is, a joint probability distribution describes R_t and S_t. Another important feature is that the probability distribution of S_t and R_t only depends on the last state and action. This turns out to be an instrumental assumption for doing any sort modelling, the implication being that the state s_t contains all relevant information about the past, hence the name, *Markov* Decision Process. This condition/assumption yields certain important features. First, it implies that size of the probability distribution would not grow linearly as more and more states and actions was represented for the agent, yielding the computations more and more expensive. Second, it allows for backward induction and dynamic programming (a topic which will be discussed later). Lastly one must ensure that when a system is modelled, the state represents all relevant information about the past. Multiple statements can be derived from (30), but most importantly one can derive the expected reward:

$$\mathbb{E}[R_t \mid A_{t-1} = a_{t-1}, S_{t-1} = s_{t-1}] = \int_{r_t} r_t \int_{s_t} p(r_t, s_t \mid a_{t-1} s_{t-1}) ds_t dr_t \tag{31}$$

The agent's goal is to maximize the cumulative rewards. This can be formulated as:

$$G_t = R_{t+1}, R_{t+2}, R_{t+3}, \cdots R_T \tag{32}$$

The formulation in (32) can be problematic with continuing tasks. That is if $T \to \infty$. Therefore discounting of rewards is usually implemented, which have the nice economic implications, that agents in the real world tend to be impatient, and therefore more realistically model real human agents:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{T-t} \gamma^k R_{t+k+1} \tag{33}$$

With γ being the discount rate, yielding a geometric series, that is known to converge, if R_k is bounded.

5.2 Value Function, Q-Function, Policy Function and the Bellman Equation

The value function represents the expected discounted cumulative returns from following a policy, π. The value function can be said to approximate the value of a strategy:

$$v_t^\pi(s_t) = \mathbb{E}_t[G_t \mid S_t = s_t] = \mathbb{E}_t\left[\sum_{k=0}^{T-t} \gamma^k R_{t+k+1} \middle| S_t = s_t\right] \tag{34}$$

A couple of things to note about (34): Since the expectation is taken over a sum, one can instead take the sum over the expectations, yielding it possible to calculate the individual expected returns from following a policy, and using those to calculate the value function. Another concept that closely resembles the value-function is the Q-function:

$$q_t^\pi(s_t, a_t) = \mathbb{E}_t[G_t \mid S_t = s_t, A_t = a_t] = \mathbb{E}_t\left[\sum_{k=0}^{T-t} \gamma^k R_{t+k+1} \middle| S_t = s_t, A_t = a_t\right] \tag{35}$$

The Q-function only differs from the value function by also conditioning on the action, and not only the state. The value function and the Q-function shares the property that it maps the expected value of a state (or state action pair) to a scalar value, where this value represents the cumulative, discounted rewards of following a certain policy. This has the nice property of allowing the agent to choose an action which maps to the highest expected value, G_t.

The Bellman equation can be expressed, using the expression for the value function:

$$v_t^\pi(s_t) = \mathbb{E}_t[G_t \mid S_t] = \mathbb{E}_t\left[R_{t+1} + \gamma G_{t+1} \mid S_t\right] = \mathbb{E}_t\left[R_{t+1} + \gamma v_{t+1}^\pi(S_{t+1})\right] \tag{36}$$

One should consider that $\mathbb{E}[v_{t+1}^\pi(s_{t+1})]$ does not imply $v_{t+1}^\pi(\mathbb{E}[s_{t+1}])$, which has the consequence of considerable computational markup when solving a model using the Bellman Equation as an update rule.

Lastly some information about the policy function. A policy function defines how the agent chooses an action:

$$\pi_t : \mathcal{S} \mapsto \mathcal{A} \tag{37}$$

In general reinforcement learning algorithms falls in two distinct categories: Algorithms that directly estimate the policy function or algorithms that work by estimating the value function or Q-function, and using these to find the optimal policy. In this paper the primary focus will be on the latter category of algorithms.

5.3 Relationship to Dynamic Programming

Dynamic programming, invented by Richard Bellman, allows for a way to find the optimal policy, π^*, and the associated value function, v^{π^*}. Dynamic programming has a set of practical and formal requirements. First, the size of the state space should be limited. This is due to the fact, that number of computations will increase exponentially with the number of states, what Richard Bellman described as "The Curse of Dimensionality". Secondly it requires that the entire MDP is known. Here one should distinguish between an environment and a model of an environment. In this paper the model and the environment coincide, but this is not always the case. An example could be a self driving car. Because the self driving car does not have a perfect model of the environment, dynamic programming cannot be used as an algorithm for navigating the environment. In other words, unless the joint probability distribution of states and actions can be formulated explicitly, dynamic programming is not feasible. In general, it can be said that dynamic programming (and also reinforcement learning) is to use the value function to structure the search for good policies (Sutton and Barto 2018). The optimal value function implies that one knows the optimal policy:

$$v_t^*(s_t) = \max_a \mathbb{E}\left[R_{t+1} + \gamma v_{t+1}^*(S_{t+1}) \mid S_t = s_t, A_t = a\right] \tag{38}$$

Dynamic programming problems can be solved by two different approaches: *value function iteration* and *policy function iteration*.

Policy function iteration consists of two steps: 1) an evaluation step. Which calculates the value of a policy, and 2) a policy improvement step. The first step can be considered a prediction step. By following a given policy the associated value function can be calculated. This is done by sweeping through the state space calculating the expected value of the state following the policy. This process continues until the algorithm has converged. In this context convergence implies the difference between the previous estimation of the value function and the current estimation of the value function, only differs below some threshold. The second step, policy improvement, works by searching through the state space, seeing if diverging from the current policy yields higher expected cumulative returns. An alternative phrasing is: Assume that the policy used for the evaluation step is optimal. Then there should be no other strategy that would yield a higher value-function for all possible states. This can be expressed more formally as:

$$\mathbb{E}\left[R_{t+1} + \gamma v_{t+1}^*(S_{t+1}) \mid S_t = s_t, A_t = \pi_t^*(s_t)\right] \geq \mathbb{E}\left[R_{t+1} + \gamma \tilde{v}_{t+1}(S_{t+1}) \mid S_t = s_t, A_t = \tilde{\pi}_t(s_t)\right] \qquad \forall s_t \in \mathcal{S} \tag{39}$$

Where $\tilde{\pi}$ is any arbitrary policy. If at any point in the state space one can find a policy that yields a higher value function than the current, then one should switch policy, which yields an updated, superior policy! The process alternates between policy evaluation and policy improvement, until no better policy can be

found. A graphical representation of the process can be considered as shown in equation (40) where $\xrightarrow{\mathbf{E}}$ denotes a policy evaluation and $\xrightarrow{\mathbf{I}}$ denotes policy improvement:

$$\pi^0 \xrightarrow{\mathbf{E}} v^{\pi^0} \xrightarrow{\mathbf{I}} \pi^1 \xrightarrow{\mathbf{E}} v^{\pi^1} \xrightarrow{\mathbf{I}} \cdots \xrightarrow{\mathbf{I}} \pi^* \xrightarrow{\mathbf{E}} v^{\pi^*} \tag{40}$$

The second approach *value function iteration* computes a max over the value function implying only a single sweep through the state space in each iteration of the loop. The max operation yields value function iteration a considerably faster solution method. Value function iteration can be considered using the Bellman equation as an update rule (Sutton and Barto 2018). The algorithm for value function iteration is described below:

Algorithm 1: Value Function Iteration

Result: Yielding π^*, v^*

Algorithm parameter $\theta > 0$ determining accuracy of estimation;

Initialize $V(s) \quad \forall s \in \mathcal{S}$ except $V(terminal) = 0$;

while $\Delta > \theta$ **do**

  $\Delta \leftarrow 0$;

  foreach $s \in \mathcal{S}$ **do**

    $v \leftarrow V(s)$;

    $V(s) \leftarrow \max_a \mathbb{E}[R_{t+1} + \gamma V(S_{t+1}) \mid A_t = a, S_t = s]$;

    $\Delta \leftarrow \max(\Delta, \mid v - V(s) \mid)$

  end

end

So for each sweep through the state space a single sweep of policy evaluation and a single sweep of policy improvement is performed. Again this algorithm terminates when the difference between the value function of the last sweep and the current value function is below some threshold.

In economics a dynamic programming solution will usually have the addition of using backwards induction. This is due to the fact the age will evolve deterministically. The model assumes an agent acting over T time steps, terminating when the agent reaches a certain age. So in a sense the agent moves in a deterministic fashion towards the termination of the environment. This implies that one can solve such a model by only doing a single sweep through the state space! The agent will maximize its value function in the terminating period. Now using this value associated with the terminating period, the agent can consider his actions in $T-1$, remembering the Bellman equation can be used as an update rule:

$$V_{T-1}(s_{t-1}) = \max_a \mathbb{E}[R_T + \gamma V_T(S_T) \mid S_{T-1} = s_{T-1}, A_{T-1} = a] \tag{41}$$

In other words, one can model all possible states that the agent can encounter in each time step of the model, making it possible to find the optimal value function v^{π^*} and policy function π^* by a single sweep through the state space using a combination of dynamic programming and backwards induction. The implementation of value function iteration in this paper will be explored in section 7. In the reinforcement learning literature, this concept of updating estimates of values of states based on value estimates of other states, is called bootstrapping[3].

Before moving on to other reinforcement learning techniques, the concept of *Generalized Policy Iteration* (GPI) is introduced. GPI is the concept of letting policy iteration and policy improvement interact with the intention of having the policy converge to the optimal policy. In practice the policy evaluation will be done with respect to the current policy. Policy improvement will be greedy with respect to the current value function. As described by Sutton and Barto (2018): The value function stabilizes only when it is consistent with the current policy, and the policy stabilizes only when it is greedy with respect to the current value function. Thus, both processes stabilize only when a policy has been found that is greedy with respect to its own evaluation function. This implies that the Bellman optimality equation holds, and thus that the policy and the value function are optimal.

5.4 Overview of Reinforcement Learning Techniques

Two different reinforcement learning methods will be presented in section 7. Here I present some basic information that makes the reader able to digest the material presented. First it is important to address why not only use dynamic programming. Dynamic programming requires that a perfect model of the environment is accessible. This is due to the fact, that when calculating the expected value function for each action, the probability distribution of the reward and the next state, needs to formulated in an explicit form. The other techniques presented here does not have the same requirement. Secondly DP methods require that the size of the state space must be limited. The implementations presented later does not have the same requirements, allowing for approximating the value function and/or the policy function. Below I explain two different methods of learning *Monte Carlo Methods* and *Temporal Difference Learning* these being the two foundations of the reinforcement learning methods presented later. It is useful to distinguish between *control* and *prediction*. Prediction can be considered the step of estimating the value function, whereas control relates to how to approximate the optimal policies. Finally, one should make a distinction between *off-policy* and *on-policy* methods. Sutton and Barto (2018) describes the difference as: On-policy methods attempt to evaluate or improve the policy that is used to make decisions, whereas off-policy methods evaluate or improve a policy different from that used to generate the data.

[3]In economics bootstrapping will usually refer to a non-parametric, sample based approach for doing inference of a parameter estimate. These things are unrelated.

5.4.1 Monte Carlo Methods

Before delving into MC-methods it is appropriate to introduce some more terminology: When talking about an *episode* it should be understood as an agent moving through the environment from start to termination. When talking about a *step* it should be understood as going from one state to the next in the environment.

First consider the problem of Monte Carlo prediction. The best way to get a sense of prediction with Monte Carlo methods is to present an algorithm:

Algorithm 2: First-Visit MC prediction, for estimating v^π

Result: Yielding v^π

Input: policy π to be evaluated;

$Returns(s) \leftarrow$ an empty list $s \in \mathcal{S}$;

while *Forever* **do**

  Generate an episode: $S_0, A_0, R_1, S_1, A_1, \cdots S_{T-1}, A_{t-1}, R_T$;

  $G \leftarrow 0$;

  foreach *step in episode,* $t = \{T-1, T-2, \cdots, 0\}$ **do**

    $G \leftarrow \gamma G + R_{t+1}$;

    if $S_t \notin \{S_{t-1}, S_{t-2}, \cdots S_0\}$ **then**

      Append G to $Returns(S_t)$;

      $V(S_t) \leftarrow average(Returns(S_t))$;

    end

  end

end

Algorithm 2 shows how the general concept of estimating the value function for given policy. The agent follows the policy until the game terminates. Starting from the terminating state, a reversed experience replay is performed using the discounted rewards to approximate the value function for each state the agent visited. However, the algorithm above relies on a model of the environment. If not such a model is present, one will need to estimate the value of each action. Instead of considering the value function, the Q-function (state-action pair) is used for Monte Carlo estimation. It is assumed that the policy stays constant. That is, the policy does not update as more and more episodes are experienced, which defeats the purpose of learning how to interact with the environment. This can be addressed by updating the policy, and then discard old experience. Another possibility is to accept the non-stationarity of the data collected, and update the policy using the data from a previous policy, hoping that with time the algorithm will converge. This is in fact generalized policy iteration.

Consider now how to do Monte Carlo Control, i.e., approximating the optimal policy. Just as with policy

iteration the pattern followed is:

$$\pi^0 \xrightarrow{\mathbf{E}} q^{\pi^0} \xrightarrow{\mathbf{I}} \pi^1 \xrightarrow{\mathbf{E}} q^{\pi^1} \xrightarrow{\mathbf{I}} \cdots \xrightarrow{\mathbf{I}} \pi^* \xrightarrow{\mathbf{E}} q^{\pi^*} \tag{42}$$

Policy evaluation is done as described above. Policy improvement is done by making the policy greedy with respect to the current estimated Q-function (Sutton and Barto 2018). Since the Q-function instead of the value function is used, no model is needed to construct a greedy policy (Sutton and Barto 2018). The greedy policy is the one that for each $s \in \mathcal{S}$ deterministically chooses an action with maximal action-value:

$$\pi(S_t) = \arg\max_a Q(S_t, a) \tag{43}$$

Now this algorithm rests on the assumption of *exploring starts* and on the assumption of *infinite episodes*. The assumption of infinite episodes is to ensure convergence, and in practice the algorithm is usually run until the algorithm has converged by some high number of episodes. The more problematic assumption is that of exploring starts. This is due to the fact, that in reality one cannot assume that there is a non-zero probability of starting in all states $s \in \mathcal{S}$, and proceed the episode from that starting point. This is an essential assumption, because otherwise, one could not know the value function of unexplored states without visiting them.

The problem of not visiting every state is closely related to the exploration vs. exploitation trade-off. If an algorithm acts greedy (only exploits) new and better policies will never be discovered. The exploration of the state space can be done by either on-policy control or off-policy control. The exploration will in this paper be done by ϵ-greedy algorithms. ϵ-greedy algorithms chooses with probability $1 - \epsilon$ a (uniformly) random action each step, else it greedily chooses the action that corresponds to the highest Q-value. The next section will give an example of both on-policy and off-policy methods.

5.4.2 Temporal Difference Learning

Temporal Difference (TD) learning combines ideas taken from Dynamic Programming and Monte Carlo Methods. It takes from Monte Carlo methods, that you do not need a perfect model of the environment. It takes from Dynamic Programming the bootstrapping, i.e., it uses learned estimates to update, without needing the final outcome. Just as Monte Carlo methods, Temporal difference methods has a prediction and control element.

TD prediction can be summarized in the equation:

$$V(S_t) \leftarrow V(S_t) + \alpha \left[R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \right] \tag{44}$$

The equation states that $V(S_t)$ should be updated according to the return in a given period + the value function of the next period $V(S_{t+1})$. This method is called TD(0), since it only uses 1 step to update the value function. In essence TD methods learn a guess from a guess. This allows for not having a complete model of the environment (of rewards and next-state probability distributions). Compared to MC methods the TD methods can learn at each step; update its estimate at each point in time. It is also proven, that for any policy π kept fixed, then a TD(0) algorithm will converge to v^π (Sutton and Barto 2018). Even though, an open mathematical question, in practice it is found that TD methods converge faster than MC methods (Sutton and Barto 2018).

Control with temporal difference learning, can also be separated into on-policy control and off-policy control. The most used on-policy control for TD methods is called SARSA (state, action, reward, state, action). Just as with MC methods a need to trade off exploitation and exploration is present. The agent want to see new parts of the state space (exploration), but should also use (exploit) what is assumed to be the optimal choice given the value function associated with the current value function. Consider the policy generating the data being an ϵ-greedy algorithm. In this case, instead of using the value function, consider the case of using the state-action pair function $Q(S_t, A_t)$, yielding the update rule:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\right] \tag{45}$$

This update is made after each step in a non-terminal state of the environment. Now notice here that the update rule uses the realized state S_{t+1} and A_{t+1}. In other words the data that is used to update the estimate is the data generated from following the epsilon greedy policy!

Off-policy control can be done by using Q-learning. Q-learning has the slight twist on update the rule for SARSA that:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\right] \tag{46}$$

Actions is still chosen by an ϵ-greedy policy, the difference here being the updating rule presenting a different policy than the actual policy followed. This is due to the fact that the Q-function is updated under the assumption of greedily choosing an action in time $t+1$ conditional on the state. Q-learning and double Q-learning will be explored in depth later when describing the algorithms used in this paper.

5.5 Landmarks

Finally, a brief overview of environments/games where reinforcement learning have been instrumental to the current hype of reinforcement learning. The first big success of reinforcement learning was made by

Tesauro in 1992 creating an agent capable of learning to play backgammon trough self play (Sutton and Barto 2018). Using an artificial neural network to approximate the value function, and using a temporal difference algorithm, the algorithm was capable of playing expert level backgammon. In 2011 the IBM Watson algorithm won in jeopardy using the same methods as Tesauro did for his backgammon agent (Sutton and Barto 2018). In 2013 the company DeepMind (now acquired by google), showed that it was possible for a reinforcement learning algorithm to learn to play video games. Here an important feat was, that it was fed the raw image input and used an artificial neural network to transform this image into a representation of the state space allowing for navigating in the environment (Mnih et al. 2013). In 2016 DeepMind created AlphaGo and a year later AlphaGo Zero, which learned to master the game of Go. This was assumed in a long time to be a hard problem for learning algorithms due to its very large state and action space (Silver et al. 2018). The first iteration used expert players to learn the game, while the AlphaGo zero used only selfplay. These examples show that the capabilities of these algorithms to learn to navigate in complicated environments, might leave a way for high dimensional dynamic economic models to be solved using reinforcement learning methods.

6 Deep Learning

6.1 Why Machine Learning

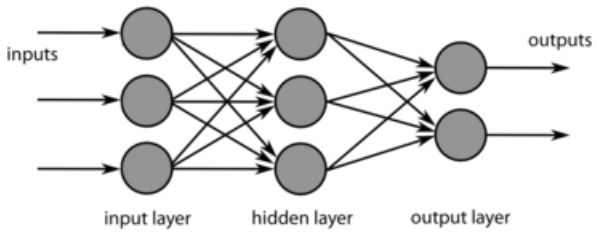
Machine learning methods try to model either the joint or the marginal distribution of some covariates and some target. In statistics a model is usually explicitly formulated. A typical example could be linear regression. The covariates is decided on, and transformed such that they fit the desired model. The linear regression will yield a parameter vector which can then be interpreted. This interpretation is usually the focus; parameter inference. Example: Does an increase in the minimum salary have a negative effect on BNP pr. capita. Machine Learning focuses on prediction. That is, the objective is to predict some target conditional on some covariates. The specific model is not necessarily important, instead a focus on Out-Of-Sample error is the focus. These predictive methods lend themselves well where causal inference is not needed. Example: What is the expected consumption on a monthly basis by person with a given set of characteristics. When formulating a traditional econometric method, f.x. OLS, there are standard ways to infer if the model is well specified. In general machine learning do not the same asymptotic results regarding regularization of a model. Usually sample splitting will be used instead. Take a data set, split the data set into two partitions a test set and a training set. First the hyper parameters of the machine learning algorithm is tuned, usually by finding which set of hyper parameters that yield the best performance on the training data. Cross validation is the go-to procedure to find optimize the hyper parameters. The final algorithm is only used on the test data set once yielding the out-of-sample performance. Machine learning

methods, as mentioned before usually has associated hyper parameters. These are parameters of the model, which is not fitted by training on the data, rather these are specifications of the algorithm before training the model. Much of machine learning is about finding the right hyper parameters and regularizing the model in an intelligent way (Friedman, Hastie, and Tibshirani 2001). Now why is this paper concerned with ML methods? This is due to the fact, that when estimating the value function, or the policy function, one cannot be sure that this follows a linear function. In fact value functions and policy functions might be highly non-linear, which is where machine learning methods shine. Considering that the value function is a conditional expectation: $\mathbb{E}[Y \mid X = x]$, where Y is the expected cumulative discounted rewards and X is the state, implies that machine learning methods is the most appropriate choice for value function estimation. In this case the causal interpretation of the influence of a specific state on the value function is not of interest, rather it's the accuracy of the expected value function[4]. Deep neural networks has been the standard way to implement reinforcement algorithms, however other machine learning methods can also be used. The reason for deep Learning methods being the standard implementation is the convenient property of online updating of the networks weights. In other words, as more data comes in, the neural network can be incrementally fitted to the new data.

6.2 Deep Neural Networks

Deep learning (feed forward networks) which is used in this paper is in fact just layered non-linear functions. Figure 3 illustrates the architecture of a deep neural network[5].

Figure 3: Illustration of feed forward neural network.



The network can be described as having an *input layer*, that takes the covariates, $\mathbf{x}$. The *hidden layer* makes a transformation of the previous layer. This also implies that a hidden layer, can be followed by an arbitrary number of other hidden layers. Lastly an *output layer* that maps the representation of the hidden layer into the desired output. For classification that could be an one-hot encoding of the classes, and for regression a single real valued scalar.

[4]Obviously, intelligent agents usually do causal inference on their actions. They might not do a certain action exactly because they have some causal notion of how the environment will evolve conditional on their action.

[5]Figure found at `https://upload.wikimedia.org/wikipedia/commons/thumb/c/c2/MultiLayerNeuralNetworkBigger_english.png/381px-MultiLayerNeuralNetworkBigger_english.png`

As illustrated in figure 3, each layer is broken down into smaller cells. The number of cells in each layer corresponds to the width of the layer. The wider the layer the more flexible representation the given layer is capable of doing. The hidden cells work as mentioned by creating a non-linear transformation of the input from last layer:

$$z_i^+ = g(\mathbf{z}; \theta) = g(\mathbf{w}^T \cdot \mathbf{z} + b) \tag{47}$$

The output of cell i is the real valued scalar z_i^+. g is the activation function that maps the input into the output. $\mathbf{w}$ is the weights of dot product, $\mathbf{z}$ is a vector of the outputs from the last layer, and b is bias or the constant in the activation. In that sense the activation looks like a linear regression squashed through an activation function g. Multiple different activation functions has been proposed. Originally the logistic function was preferred, but in later years the rectified linear unit activation function has been popular (Goodfellow, Bengio, and Courville 2016):

$$\textbf{Rectified Linear Unit: } \max\left\{0, \mathbf{w}^T \cdot \mathbf{z} + b\right\} \tag{48}$$

The neural network can in other words be considered a function f, that takes an input $\mathbf{x}$ and maps it to some output y, parameterized by θ, which is a collection of all the weights and biases associated with each individual cell.

6.3 Stochastic Gradient Descent and Optimization

The neural network is estimated (or trained) by using stochastic gradient descent. This is possible due to the fact, that the networks can be represented as set of nested functions, such that the chain rule can be applied. The loss function can in other words be differentiated with respect to the parameter vector θ as shown below:

$$\frac{\partial}{\partial \theta}\mathcal{L}(\mathbf{X}, \mathbf{Y}, \theta) = \frac{\partial}{\partial \theta}\left(\sum \ell_i\right) = \frac{\partial}{\partial \theta}\left(\sum \left(\hat{Y}_i - Y_i\right)^2\right) = \frac{\partial}{\partial \theta}\left(\sum \left(f^\theta(X_i) - Y_i\right)^2\right) \tag{49}$$

In equation (49) the loss function $\mathcal{L}$ is assumed be a mean squared error loss function, in other words a regression problem. The optimization works by minimizing the loss with respect to the parameters θ. In modern neural network architectures it is not unusual that such network has in the excess of a million parameters. The objective function of the optimization cannot be assumed to be convex due to the non-linearity of the activation functions. For this reason deep neural networks is not solved analytically. Instead gradient descent is used for estimating the parameters of the network. The update rule of the parameters can be described as (Goodfellow, Bengio, and Courville 2016):

$$\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\mathbf{X}, \mathbf{Y}, \theta) \tag{50}$$

So for each step in the algorithm the derivative of the loss function with respect to the parameters can be calculated, and the parameters can be updated by taking a small step in parameter space of size α in the direction that reduces the loss. Stochastic gradient is a response to the fact that it can be computationally expensive to calculate the gradient for the entire data set in each update step. This is important for deep neural networks, since the training period of a large network, even on optimized hardware, can take a very long time, so any speed up for the training is important. In practice this implies that the training data is split into mini batches usually of size 32 to 128. The optimization is then performed on each of the small batches, taking a small step of size α for each step.

7 Soution Methods

7.1 Value Function Iteration

Three different solution methods is presented in this section. I lead with modification of value function iteration, next deep Q-Learning is presented ending with double deep Q-learning iteration.

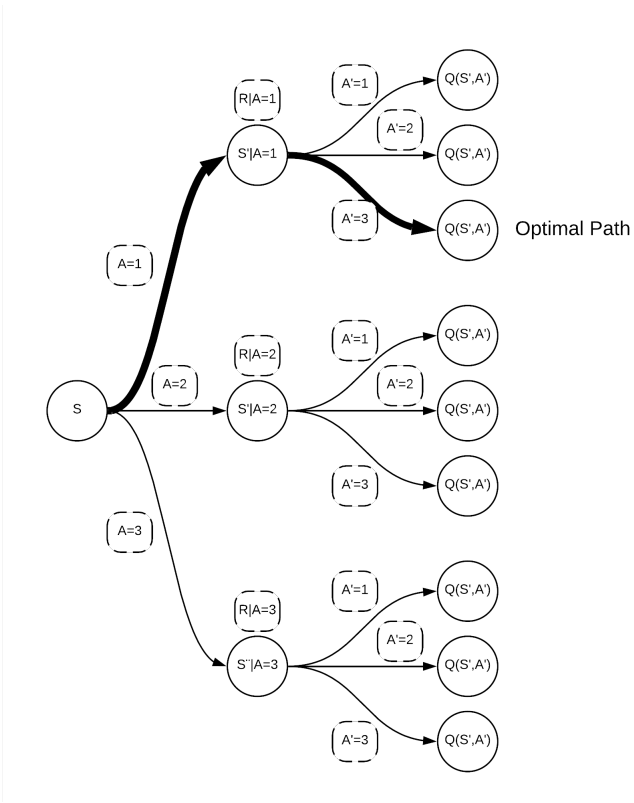
The value function iteration presented in this paper has certain modifications to the algorithm presented in 5.3. First and foremost, I consider the Q-function instead of the value function allowing to store state-action value pairs. Second it should be noted that in this formulation some states are continuous not allowing for tabular solutions. Thirdly, the dimension of the state-space + the size of the grid made it infeasible[6] to solve the model by classical ways of solving a model by discretizing the state space.

The model is solved using backwards induction. This is due to the fact the model terminates in a deterministic fashion when the agents reach a certain age. For each step a large random sample of states is drawn conditioning on a given age. For each of the states the agent takes each of the possible actions storing the results. This way a large sample of rewards and states can be generated. Furthermore for each action taken (if the state is not terminal) the Q-function can be evaluated in the new state, taken the max of each possible action, allowing for the estimation of the value function. A graphical representation of the concept is shown in figure 4.

The Q-function for each action in the action space is made by mapping each $a \in \mathcal{A}$:

[6] In my initial attempt to solve the model by value function iteration, I attempted to get the expectation of the value function using Gauss-Hermite integration, and discretizing the state space. The solution time was infeasible, which was why my approach changed.

Figure 4: Value function iteration using Q-function



$$Q(S_t, a) = R_{t+1} + \gamma \max_{a' \in \mathcal{A}} Q(S_{t+1}, a') \tag{51}$$

In that sense this can be considered akin to Monte Carlo integration, however instead of approximating the expectation in a single point, rather find the distribution of the rewards + discounted value function over the entire state space. The idea is to approximate the integral by using a statistical method, in this case deep learning even though another machine learning method would be equally good. Consider f to be a deep neural network, that has the property:

$$f : \mathcal{S} \mapsto \mathbb{R}^{|\mathcal{A}|} \tag{52}$$

For a given point in state space a prediction of the value function is computed for each possible action. This implies the method only is feasible for discrete state space. By trying to reduce the mean squared error between the true values of the Q-function, and the prediction, the $\mathbb{E}[Q(a, s)]$ can be found, which corresponds to integration as could be done using Gauss Hermite or Monte Carlo integration. The algorithm is presented in algorithm 3.

Algorithm 3: Value Function Iteration Solution Method

Result: Estimated Q function

Initialize $\tilde{Age} = Age_{max}$;

Initialize empty lists for storing results: X, Y;

Initialize memory counter $j = 1$;

while $\tilde{Age} > Age_{min}$ **do**

  Draw $\{s_i\}_{i=1}^{N}$, where $s_i \sim Uniform(\mathcal{S}) \mid Age = \tilde{Age}$;

  foreach s_i **do**

    Create empty array Z of length $\mid \mathcal{A} \mid$;

    if $\tilde{Age} = Age_{max}$ **then**

      foreach $a_k \in \mathcal{A}$ **do**

        $Z[k] \leftarrow R_{t+1}, \quad R_{t+1} \sim \mathcal{E} \mid A_t = a_k, S_t = s_i$;

      end

    else

      foreach $a_k \in \mathcal{A}$ **do**

        $Z[k] \leftarrow R_{t+1} + \gamma \max_{a \in \mathcal{A}} \hat{Q}(S_{t+1}, a), \quad R_{t+1}, S_{t+1} \sim \mathcal{E} \mid A_t = a_k, S_t = s_i;$

      end

    end

    $Y[j] \leftarrow Z, X[j] \leftarrow s_i;$

    $j = j + 1;$

  end

  Estimate $\hat{Q}$ by training a Deep NN using samples from X, Y;

  Decrease $\tilde{Age}$ be one;

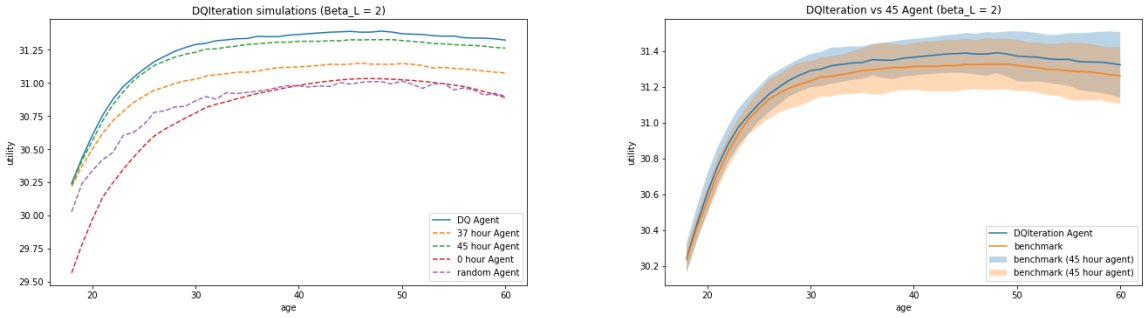
end

Since I use a deep neural network to approximate the Q-function, the architecture and hyper parameters of the network needs to be considered. The same is true for the sampling scheme.

For each age 20.000 random samples is drawn. This is because any smaller number of draw seemed to be detrimental to the performance. This is inline with standard deep learning practices, where neural networks are known to be very data hungry. A random sample of 100.000 observations is used when training the network. If I have not yet accumulated 100.000 observations the algorithm draws all observations. The architecture of network is fairly simple being a two-layer fully connected network. First layer being 16 nodes wide, second fully connected layer being 8 nodes wide. I found that mini batching, did not seem to work well on this particular task, and instead I train on all observations, using a validation split of 30 %,

training for a maximum of 150 epochs[7] and finally I allow for early stopping, that is, when the validation loss is not furthering decreasing I stop the training of the network. I do allow the algorithm a patience of 5. Implying that the algorithm will try to lower its validation loss for five additional epochs before terminating the training.

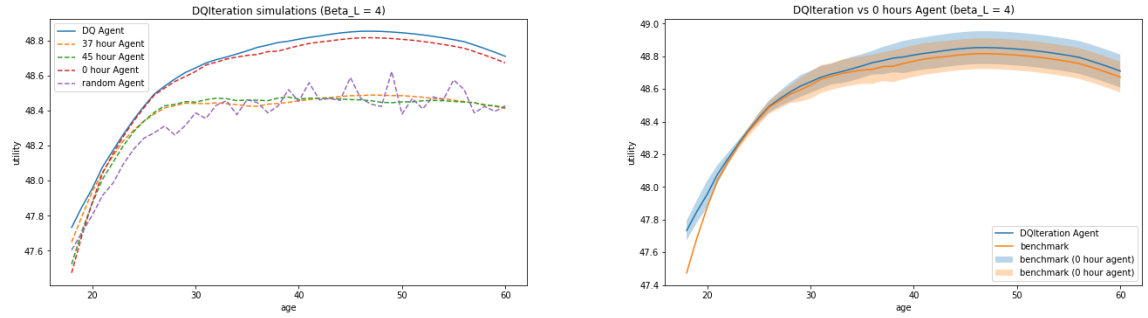
Figure 5: Value Function Iteration solution vs. benchmark ($\beta_L = 2$)



(a) Simulated Paths

(b) Variance of Paths

Figure 6: Value Function Iteration solution vs. benchmark ($\beta_L = 4$)



(a) Simulated Paths

(b) Variance of Paths

Figure 5 and 6 shows the results of the Value Function Iteration algorithm[8] compared to 4 benchmarks. Three deterministic agents working either 0, 37 or 45 hours pr. week and one agent taking random actions. Figure 5 shows the utility for each step over the life cycle when the preference for leisure is fixed at $\beta_L = 2$. As the figure shows the VFI agent learns to navigate the environment, just as well as the best deterministic agent. Looking to the RHS plot of 5 it is clear that there is a substantial overlap between the two agents. The variance of the path is represented as one standard deviation of the utility for a given age for all episodes of the given agent. Figure 6 compares the benchmark agents with the VFI solution when

[7] A epoch corresponds to a full sweep through the the data set.

[8] The plots use the name DQIteration (Deep Q-Function Iteration) instead of Value Function Iteration.

considering a preference for leisure $\beta_L = 4$. Again the VFI solution is as good as the best benchmark.

7.2 Deep Q-learning

In this paper I implement[9] the deep Q-learning algorithm used by Mnih et al. (2013) for beating Atari games as a way to solve model. A few modifications is made to the original implementation, due to the fact, the environment they were navigating only returned sensory data (an RGB representation of an image). A part of their achievement was to transform these images into features that the value function accurately could map into scores of the game. Another difference is that this paper implements a scaling module of the variables for better performance.

Just as described in section 5 the algorithm tries to maximize the Bellman equation. However now the value-function is estimated using Deep neural network as a function approximator. Mathematically this can be described as finding:

$$Q^*(S_t, A_t) = \mathbb{E}[R_{t+1} + \max_a Q^*(S_{t+1}, a) \mid S_t, A_t] \tag{53}$$

However here $Q^*(S, A)$ is approximated by a parametric function (in this case a Deep Neural Network) $Q(S, A; \theta)$. Following the terminology of Mnih et al. (2013), this function approximator is referred to as the Q-network. The Q-network can be trained using stochastic gradient descent as described in section 6:

$$\mathcal{L}_i(\theta_i) = \mathbb{E}\left[(Y_i - Q(S_t, A_t; \theta_i))^2\right] \tag{54}$$

where:

$$Y_i = \mathbb{E}[R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \theta_{i-1}) \mid S_t, A_t] \tag{55}$$

where i implies the iteration of the algorithm, such i increments by one for each update of the parameters θ. The equation used for updating the weight of the neural network is described below:

$$\nabla_{\theta_i} \mathcal{L}_i(\theta_i) \mathbb{E}\left[(Y_i - Q(S_t, A_t; \theta)) \nabla_{\theta_i} Q(S_t, A_t; \theta_i)\right] \tag{56}$$

Following the formulation of Mnih et al. (2013) such that $Q(S_{t+1}, A_{t+1}; \theta_{i-1})$ is held fixed, allowing for just writing Y_i, and not the parametric form of Y_i.

[9] Originally I used my own implementation, however slow performance, caused me to use and modify an existing implementation allowing for a speed up of a factor of 10. The same is true for the Double Deep Q-learning algorithm. The original implementation can be found at Tabor (2020).

A traditional choice would be to update the weight after each step in the algorithm only using the last sample. This would the correspond to the traditional Q-learning algorithm. This was the approach used by the TD Gammon agent created by Tesauro[10] as described by Sutton and Barto (2018). The approach of using Q-learning with a non-linear function approximator has been shown to diverge under certain circumstances, and did not extend itself well to learning any other game than backgammon (Tsitsiklis and Van Roy 1997). To accommodate this problem a replay buffer is implemented. At each step an entry is made to the replay memory containing (s_t, a_t, r_t, s_{t+1}). This data set $\mathcal{D}$ has a capacity of N entries. Using the replay memory a random mini batch is sampled used to update the weights of the Q-network. Note here that the capacity of the memory buffer should be greater, by a substantial margin than the number of samples drawn. The random sample has a couple of advantages. It decorrelates the observations used to update the weights, allowing for better training Mnih et al. (2013). Using a replay buffer requires off-policy learning which is the reason for using Q-learning. This is due to the fact, that current parameters θ is not the same as those generating the data. Note that experiences is drawn randomly, and no experiences (which could have important insights) is prioritized. The full algorithm is summarized in algorithm 4.

Algorithm 4: Deep Q-learning

Initialize replay memory $\mathcal{D}$ with capacity N;

Initialize action-value function Q with random weights;

Initialize memory counter $j = 1$;

foreach *episode* $\in \{1, 2, \cdots M\}$ **do**

  Initialize sequence with an initial state s_1. This is drawn randomly.;

  for $t \in \{1, 2, \cdots, T\}$ **do**

    With probability ϵ select a random action a_t;

    Otherwise select $A_t = \arg\max_a Q^*(S_t, a; \theta)$;

    Execute action A_t in environment and observe reward R_t and the new state S_{t+1};

    Store transition $(S_t, A_t, R_{t+1}, S_{t+1})$ in $\mathcal{D}$;

    Sample random mini-batch of transitions $(S_t, A_t, R_{t+1}, S_{t+1})$ from $\mathcal{D}$;

    Set $Y_j = \begin{cases} R_j & \text{if terminal states} \\ R_j + \gamma \max_a Q(S_{t+1}, a; \theta) & \text{if non-terminal states} \end{cases}$;

    Perform gradient descent step on $(Y_j - Q(S_t, A_t; \theta))^2$;

    Increment j;

  end

end

Following the implementation Mnih et al. (2013) a trick is applied to reduce the number of computations

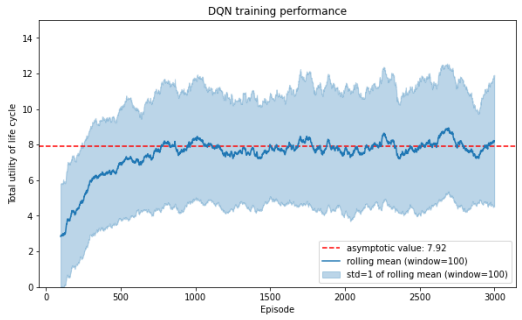
[10] I have not been able to get access to the original paper, so I have relied on the description made by Sutton and Barto.

needed when running the algorithm. Ordinarily the Q-function maps a state action pair to a scalar value estimate of the value function. The most obvious implementation would let the action be part of the input to the function, however, such implementation would require $\mid \mathcal{A} \mid$ number of look ups, at each step. This is due to the fact that each action in the action space must be used for evaluation. Instead the Q-function in this implementation maps the state space to a scalar value for each action $Q : \mathcal{S} \mapsto \mathbb{R}^{|\mathcal{A}|}$.

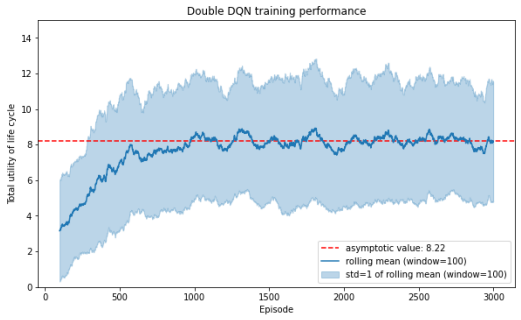
The solution used the following hyper parameters and architecture decisions: The Deep Neural network consists of an input layer of same size as the state space, followed by two fully connected layers of width 256 using rectified linear units as activation functions. Finally the network uses a linear output layer for its predictions. The output layer is of the same size as the action space. The learning rate, $\alpha = 0.0005$ and I let epsilon be decremented after each update by: $\epsilon_i = \max(\epsilon_{i-1} \cdot 0.9999, 0.01)$, where 0.01 is the minimum exploration that will be done. The replay buffer has a capacity of 1 million rows, and the mini batches used for updating the parameters is 64 rows. I scale the state space so that each variable approximately has mean zero and standard deviation of 1 when doing the batch training. Goodfellow, Bengio, and Courville (2016) argues that it increases performance, and in general accepted as being an important step for getting good performance out of neural networks. The impatience parameter is set to $\gamma = 0.99$, using a standard value. I make the parameter β_L a part of the state space, drawing a random value (uniformly) in the interval $[0.2, 6.0]$ at the beginning of each episode, letting the agent navigate through the environment with given preference for leisure. This is done, so only a single solution of the model is necessary for estimating the parameter later. I scale the rewards (so they are approximate zero mean and have a standard deviation of 1, conditional on the β_L value. Again this is done to improve the training performance and to do introspection; it becomes possible to see if there is any trend in the training performance. The algorithm is trained for 3000 episodes. Figure 7 (left plot) shows the training performance of the Deep Q-learning algorithm. The plot shows the agent's cumulative rewards over the life cycle of each episode. The performance begins at around 2.5 and ends at an average of 7.9 as the asymptotic value. The performance seems to reach this asymptotic level at around 1000 episodes. I use the Adam optimizer for the gradient descent step when updating the weights.

Figure 8 and 9 compares the performance of the algorithm to two different benchmarks. One benchmark is the environment with $\beta_L = 2$ another with $\beta_L = 4$. These differences in preferences will yield different optimal policies conditional on the β_L. I compare the policy chosen by the agent with different deterministic policies. Figure 8 (left plot) shows either using $0, 37, 45$ hours pr. week as comparisons and an agent picking randomly. All agents have been simulated for 300 episodes, yielding the plots being averages, and the standard deviations are calculated on a pr. age basis for each agent. It should also be noted that ϵ (the exploration ratio of the agent) is set to 0, such that the agent now only greedily navigates the environment. Comparing this to the VFI agent I find it reasonable to assume that the DQN agent has learned to navigate the environment. A small dip from around age 55 can be observed, but I believe this does not make the

Figure 7: Comparing training perfance of Deep Q-Learning and Double Deep Q-Learning



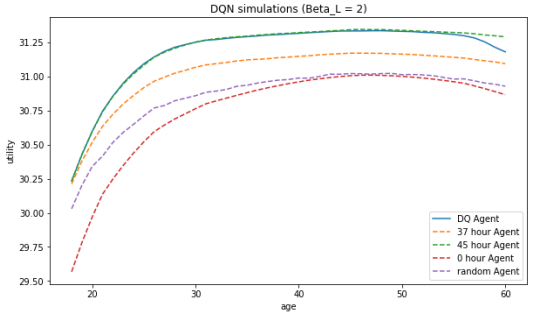
(a) DQN Training Performance



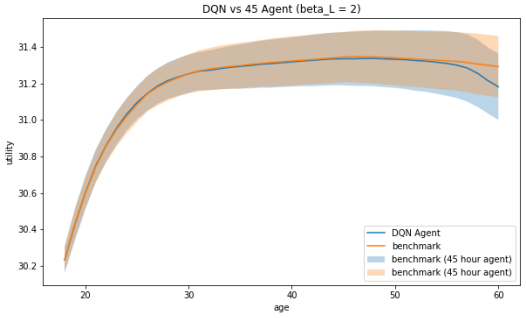
(b) Double DQN Training Performance

feat any less impressive, and furthermore I believe that the solution fairly accurately approximates the correct optimal policy. Again the best benchmark for $\beta_L = 2$ is an agent that works 45 hours a week. The deterministic and DQN agent seem to follow the same policy until the last periods, where they diverge a little. I still to conclude that the differences in performance is very small. Figure 9 shows the same plot just for $\beta_L = 4$. Again the DQN agent fairly accurately approximates the optimal policy, with even less variance than observed in with $\beta_L = 2$. One thing to note is, that a small hump can be observed for the DQN agent at around age 28, where it has found a way to beat the deterministic policy. Again this might just be variance of the simulations.

Figure 8: Deep Q-Learning solution vs. benchmark ($\beta_L = 2$)



(a) Simulated Paths

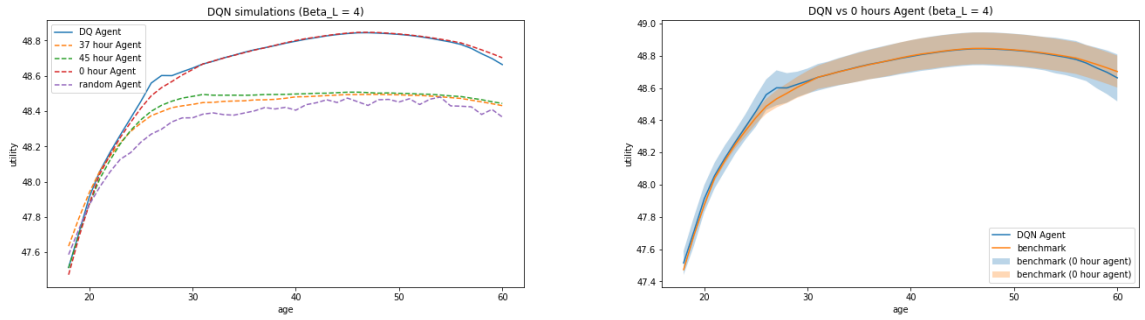


(b) Variance of Paths

7.3 Double Deep Q-learning

Deep Q-learning is good starting point for a learning algorithm using non-linear function approximation, however certain properties of the algorithm is problematic. Hasselt, Guez, and Silver (2015) argues this is

Figure 9: Deep Q-Learning solution vs. benchmark ($\beta_L = 4$)



(a) Simulated Paths

(b) Variance of Paths

mainly due to the fact that Deep Q-learning can tend to be overoptimistic in a problematic way. In general there exists two ways in which overoptimism can be good or at least not detrimental to performance. 1) If the algorithm uniformly overestimates the Q-function for all actions, then this is not associated with any problems, since taking the max of the Q-function w.r.t. to the action, will lead to the same result had the estimations been correct. In other words, performance would not change, however introspection of the algorithm might be harder. 2) It can be a good thing for an algorithm to be optimistic when faced with uncertainty. If an action would lead to observing an unexplored part of the state space, it might be a good thing to be optimistic in regard to exploration, allowing for possible new policies with higher yielding returns over the episode. Deep Q-learning do not conform to any of these properties, instead it usually overestimates the value of the action it has taken due to the max operator in the value estimation: $R_{t+1}+\gamma \arg\max_a Q(S_{t+1}, a)$. A simple extension presented by Hasselt, Guez, and Silver (2015) is to decouple the prediction with the evaluation. This can be done by having two different sets of weights, where one θ is used for the policy (policy weights) and one $\tilde{\theta}$ is used for the evaluation (target weights). This can be presented the following way. The Q-network target follows the equation below:

$$Y_t^Q = R_{t+1} + \gamma Q(S_{t+1}, \arg\max_a (S_{t+1}, a; \theta); \theta) \tag{57}$$

The Double DQN target is contrasts the DQN-target shown above, by using two sets of weights:

$$Y_t^{DoubleQ} = R_{t+1} + \gamma Q(S_{t+1}, \arg\max_a Q(S_{t+1}, a; \theta); \tilde{\theta}) \tag{58}$$

This alleviates the problem of overestimating the values returned by the algorithm for the following reason. Consider (57). When calculating the value of the state action pair, one uses the same function to choose the action, prompting a real risk of overestimation of the state action pair of the chosen action. The equation

below (58) instead estimates the value of the position by using a different set of parameters, $\tilde{\theta}$. Following the implementation of Hasselt, Guez, and Silver (2015) instead of training 2 separate networks the target weights are inherited from the evaluation weights. This is to reduce time of training the algorithm, even though the policy and target network can not be considered perfectly decoupled. The implication of this design choice is $\tilde{\theta} = \theta^{previous}$, such that, the target estimation is made by the old weights, while the value of greedy policy uses the new weights:

$$Y_t^{DoubleQ} = R_{t+1} + \gamma Q(S_{t+1}, \arg\max_a Q(S_{t+1}, a; \theta_t); \theta_t^{previous}) \tag{59}$$

The full algorithm, which to large extend mirrors the algorithm for Deep Q-learning is presented in algorithm 5:

Algorithm 5: Double Deep Q-learning

Initialize replay memory $\mathcal{D}$ with capacity N;

Initialize action-value function Q with random weights: $\theta, \tilde{\theta}$;

Initialize memory counter $j \leftarrow 1$;

Initialize k when to replace target weights $\tilde{\theta}$;

foreach *episode* $\in \{1, 2, \cdots M\}$ **do**

  Initialize sequence with an initial state s_1. This is drawn randomly.;

  for $t \in \{1, 2, \cdots, T\}$ **do**

    With probability ϵ select a random action a_t;

    Otherwise select $A_t = \arg\max_a Q^*(S_t, A_t; \theta)$;

    Execute action A_t in environment and observe reward R_t and the new state S_{t+1};

    Store transition $(S_t, A_t, R_{t+1}, S_{t+1})$ in $\mathcal{D}$;

    Sample random mini-batch of transitions $(S_t, A_t, A_{t+1}, S_{t+1})$ from $\mathcal{D}$;

    Set $Y_j = \begin{cases} R_{t+1} & \text{if terminal states} \\ R_{t+1} + \gamma Q(S_{t+1}, \arg\max_a Q(S_t, a; \theta_t); \tilde{\theta}_t) & \text{if non-terminal states} \end{cases}$;

    Perform gradient descent step on $(Y_j - Q(S_t, A_t; \theta))^2$ w.r.t θ ;

    If j is divisible by k replace target weights $\tilde{\theta} \leftarrow \theta$;

    Increment j ;

  end

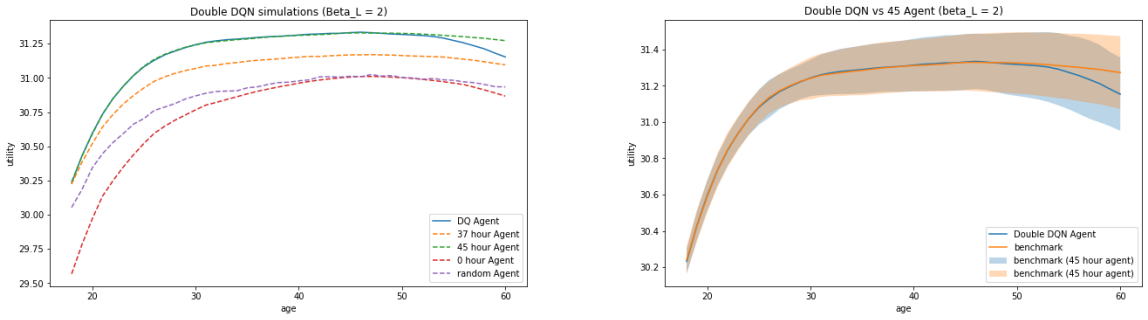
end

Just as with the Deep Q-learning implementation the Q-function map from: $\mathcal{S} \mapsto \mathbb{R}^{|\mathcal{A}|}$, to reduce the number of computations.

For the Double DQN agent I in general use the same hyper parameters used for the DQN agent: The network

architecture is again an input layer of the size of the state space, followed by two fully connected layers with Rectified Linear units as activation functions ending with an output layer with linear activation of same size as the action space. Again I let the learning rate be $\alpha = 0.0005$ and the $\epsilon_i = \max(\epsilon_{i-1} \cdot 0.9999, 0.01)$, such that exploration is performed through out the training period. Again the replay buffer has a capacity of a million rows and I use mini batches of size 64 to perform stochastic gradient descent on the weight of the neural network approximating the Q-function. I transform the states such that they are mean zero and a standard deviation of 1. The same is true for the rewards conditional on the β_L value. I uniformly draw β_L from the interval $[0.2, 6]$ and sets this to be part of the environment in the beginning of each episode. I use the Adam optimizer when updating the weights using gradient descent. I let the target network inherit the old weights every 100'th iteration of the algorithm. The algorithm is trained for 3000 episodes. Figure 7 (right plot) shows the performance. Very similar to the DQN I find that the performance stabilized at around 1000 episodes, and reaches an asymptotic performance of 8.22, which is slightly higher compared to the DQN agent which had an asymptotic value of 7.92.

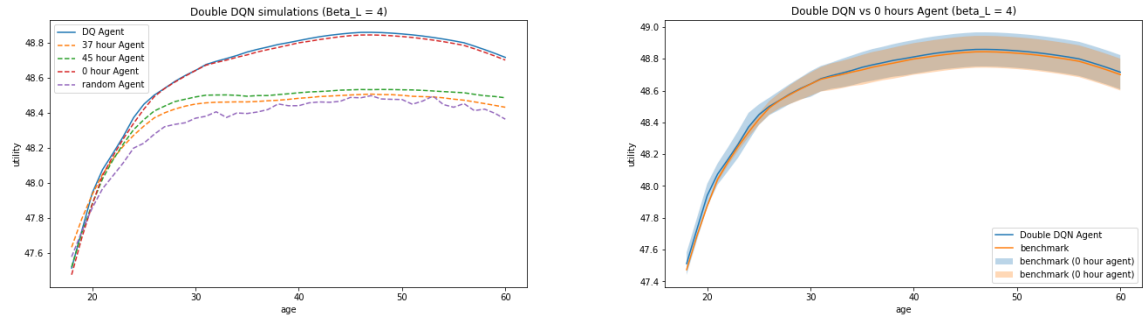
Figure 10: Double Deep Q-Learning solution vs. benchmark ($\beta_L = 2$)



(a) Simulated Paths

(b) Variance of Paths

Figure 11: Double Deep Q-Learning solution vs. benchmark ($\beta_L = 4$)



(a) Simulated Paths

(b) Variance of Paths

Figure 10 and 11 shows the performance of the Double DQN agent comparing it to two different preferences of leisure: $\beta_L = 2$ and $\beta_L = 4$. Four agents is used for comparison: An agent that chooses actions randomly, and three agents that in an deterministic fashion works $0, 37$ or 45 hours a week. The results of the Double DQN agent is comparable to those found within the other agents (VFI and DQN). Figure 10 shows the performance where the preference for leisure $\beta_L = 2$. Again the best deterministic policy is the 45 hour agent, which is very close to what the DQN agent chooses, except for the last 5 years of the agent's life cycle where the utility is slightly lower. Comparing the variance of the utility on RHS plot of figure 10, there is overlap of the simulations of the Double DQN agent and the 45-hours a week agent. Figure 11 compares the Double DQN agent to the agents when the preference for leisure $\beta_L = 4$. Here again the optimal policy pretty closely mirrors the best deterministic agent, the one which does not work, with a clear overlap presented in figure on RHS.

8 Estimation

Only a single parameter of the model needs to be estimated: β_L. For this estimation, method of simulated moments is employed. Eisenhauer, Heckman, and Mosso (2015) suggests a diagonal matrix as a standard choice of weight matrix when using methods of moments, allowing for writing the estimation as the sum of squared errors between the simulated and empirical moments. Phrased differently: the objective function minimizes the mean squared error (MSE) between the empirical moments and the simulated. Since I use aggregate data from Statistics Denmark, I have not access to the standard errors of the estimates of the empirical moments usually used for the weight of the individual moments. Instead I weigh each moment equally. Since this application relies on grid search over the single parameter β_L, I choose the β_L corresponding to the lowest loss of the objective function.

Usually when estimating a structural model using simulated methods of moments one would not supply the parameters as part of the state space. The reason for this is, that as the size of the state space expands, it gets exponentially more expensive to solve the model. The algorithm would go: For a given set of parameters needed to be estimated, supply a random initialization. Solve the model for given parameters. Use the solved model to calculate the moments. Use numerical optimization to minimize given optimization problem. However in this application I have chosen to make the parameter part of the state space, since this paper tries to establish that high dimensional dynamic models can be solved using Deep Reinforcement Learning, and therefore not being a limited by the size of the state space.

This paper has implemented the following strategy for estimating the model: I create a grid of 50 points with β_L values in the range $[0.2, 8]$ evenly spaced. I simulate $N = 300$ agents for a given value of β_L in the grid. Using the 300 agents the mean number of supplied hours pr. agent can be calculated. Here it is

important to note, that if an agent works 0 hours, the agent is assumed to be out of the workforce and not appear in the statistics, therefore the objective function is:

$$\text{Objective} = \sum_{q=Q_{\min}}^{Q_{\max}} \left[\mu_q - \frac{1}{\sum_{i=1}^{N} \mathbf{1}\{H_{i,q} > 0\}} \sum_{i=1}^{N} (H_{i,q}) \right]^2 \tag{60}$$

Where q denotes the age of an individual agent and i denotes an individual. μ_q is the empirical average of supplied ours of women of a age q. Again I use **LIGEF15** supplied by Statistics Denmark to get the average number of supplied of hours. I use the same seed for the simulation, every time the N agents is simulated with a new β_L value. The estimation is performed for each of the solution methods in this paper: Value function Iteration, Deep Q-Learning and Double Deep Q-Learning.

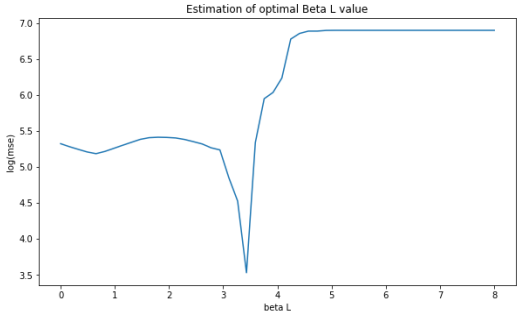
The estimation yields the following optimal values using VFI: $\beta_L = 3.43$, Optimal Values using DQN $\beta_L = 3.59$ and optimal value of Double DQN $\beta_L = 3.59$. 3.43 and 3.59 are neighbouring grid points. I believe it is safe to conclude that these methods yields if not identical, then almost identical results, and I am willing to conclude that Deep Q-Learning and Double Deep Q-Learning are excellent methods for solving dynamic models - at least in this setting!

Table 2: Estimation of β_L

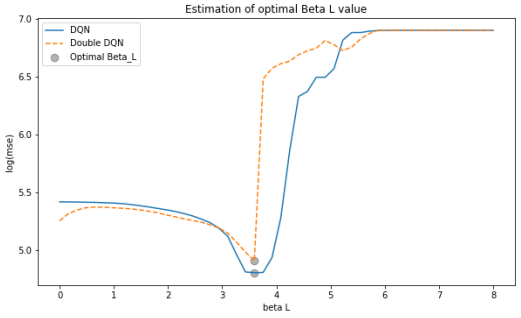
	VFI	DQN	Double DQN
$\hat{\beta_L}$	3.43	3.59	3.59

Looking to figure 12 it can be seen that the estimations follow the same trajectory. The plot shows the log(MSE) for β_L values in the range $[0.2, 8]$. Starting with low β_L having medium high MSE, falling slightly at around $\beta_L = 3$ reaching a minimum at about $\beta_L = 3.5$, and after that a sharp raise in MSE is observed.

Figure 12: Estimation of β_L

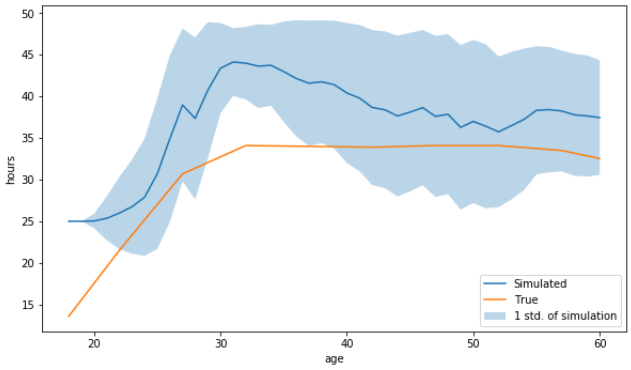


(a) Value Function Iteration Estimation of β_L



(b) DQN and Double DQN estimation of β_L

Figure 13: Average Number of Supplied Working Hours - Empirical Vs. Simulated

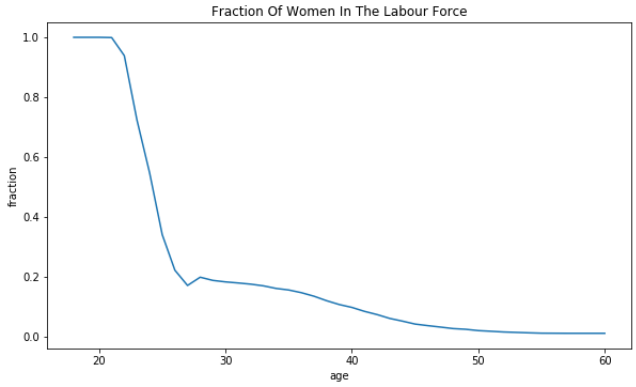


9 Results

I arbitrarily choose value function iteration when investigating the results of the model, since all the solution methods yielded identical results. 5000 agents is simulated using the optimal $\beta_L = 3.43$. Relevant summary statistics can be calculated over the 5000 simulations. The statistics can be used to infer whether or not the model fit the data and compare to what is expected from the real world. Looking to figure 13 the empirical number of hours supplied by women **LIGEF15** is compared to the simulated. Again only women actually in the labour force ($H > 0$) is considered. The fit of the data does seem reasonable. In general the simulated data set slightly overestimates the number of supplied hours. Especially around age 30 does the simulated number of supplied hours seem to be higher than the empirical. The simulated data show heterogeneity over the life cycle, where especially the later stages of the life cycle show higher variance. This could very well stem from the idiosyncratic wage path increasing over time. Considering the estimation method relied on the number of supplied hours, it is not surprising that this is where the model performs the best.

Figure 14 shows the participation rate of women in the workforce in the simulated data set. From here it is very clear, that the model underestimates the number of women working! From age 25, 80 % of the women is out of the workforce, followed by a slow decline in participation going forward to retirement. This does obviously not correspond to what can be observed in the real world. Even though it is not possible to get access to the exact number of women out of the labour force conditional on age, it can be approximated. Using **FOLK1B** supplied by Statistics Denmark the number of women in the age (15 to 60) can be found. Using that data the number of relevant women (considering the age) is around 1.664 million people. The number of women not working for relevant reasons can be summarized to: women not working due to: *working in the home*, *Studying*, *Other people outside the labour force*. These add up to: $15 + 161 + 66 = 242$ thousand people or 0.242 million people. The number of women in the relevant age

Figure 14: Fraction of Women in The Labour Force



group outside the labour force can be approximated to the number given in equation 61

$$\textbf{Women outside the labour force} = \frac{0.242}{1.664} = 0.145 = 14.5\% \approx 15\% \tag{61}$$

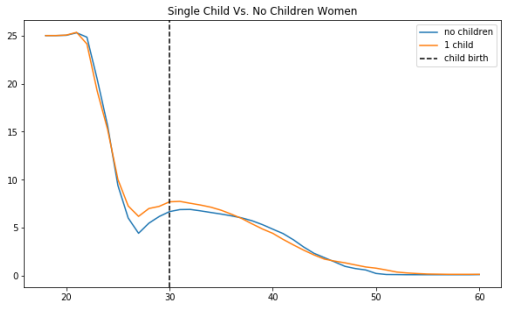
The estimation of β_L did not take this measure into account, which might be a way to improve the results.

Both figure 15b and figure 15a is inspired by the article of Kleven, Landais, and Søgaard (2019). In the article the authors show how participation of a woman is affected when she gives birth to a child. In the paper they find that women, when giving birth to a child on average use 20 % time less on work, compared to the husband, immediately after the birth of the child. The number of supplied hours slowly return to steady state with a long run penalty of around 6 %. This is not the case for the simulated household. Getting a child, do seem to decrease the number of hours supplied however, the difference is indistinguishable from the mothers not getting children. This is clear in figure 15a, where first time mothers of age 30 is compared with women that have no children. The paths are indistinguishable, where the paper by Kleven, Landais, and Søgaard (2019) would suggest a decrease of about 20 percent immediately after the birth (at age 31). These figures do not sort out women outside the labour force. I conclude the model is insufficient to capture the real world, and I move on to extend this original model.

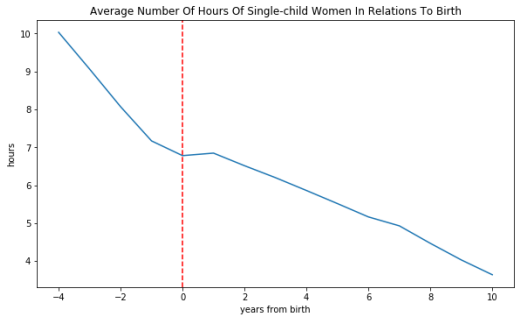
10 An Extended Model

After asserting that a deep reinforcement learning agent can navigate the environment, the model is extended to address the issues with the simple model. The first issue that should be addressed is that women should not permanently withdraw from the labour force. Second the log-function used for the sub-utilities caps the utility very hard, which implies that very small perturbations of β_L makes the agent

Figure 15: Impact of Children



(a) First time mothers vs. Women with no children



(b) hours compared to birth onset.

substitute into one of the extremes. The agent end up working either 45 hours exclusively or working 0 hours exclusively, de facto making leisure and consumption perfect substitutes. The agent should ideally be more balanced, implying the functional form of the sub-utilities should have a steeper curvature. This leads to a new expression of the utility function:

$$U_t = \beta_L L_t^{\zeta} + \beta_Y Y_t^{\zeta} \tag{62}$$

The fertility process should also be extended. I now assume that each household can have a maximum of four children. The environment tracks how many children the family has, and the associated age. In each period the number of children in the household can potentially increment by one.

$$K_{t+1} = K_t + \psi_t, \qquad \psi_t \mid Q_t \sim Bernoulli(p_\psi(Q_t)) \tag{63}$$

I let B be a vector of size 4 equal to the maximum number of children. Furthermore I let the vector C denote the individual age of the children also of size 4. This implies that for index i of vector B corresponds to the k'th child, index i of vector C implies the same child's corresponding age.

$$B_t[i] = \begin{cases} 1 & \text{if } i \leq K_t \\ 0 & \text{else} \end{cases} \tag{64}$$

The age vector is defined as:

$$C_{t+1} = C_t + B_{t+1} \tag{65}$$

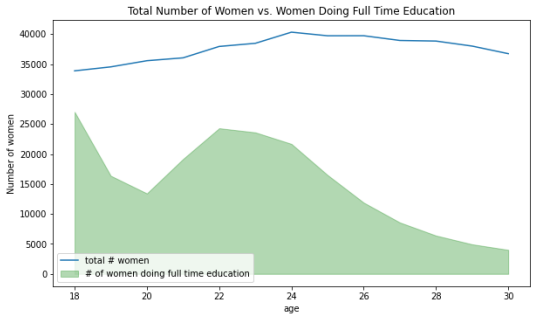
The formulation ensures the age of a child increments if and only if the child is born.

Finally the leisure component of the model needs to be adjusted to take into account the number of hours used pr. child. Let J denote the number of hours used on children. Children below age 3 takes 10 hours pr week, children in the age 3 up to age 15 takes 3.5 hours pr. week, and children above takes 0 hours of leisure pr. week for women, using the numbers found by Ekert-Jaffé and Grossbard (2015).

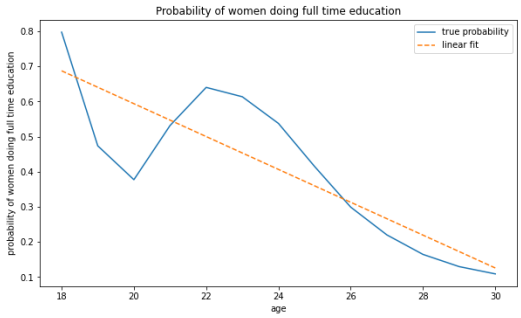
$$J_t = \sum_{i=1}^{4} \begin{cases} 0.0 & \text{if } B_t[i] = 0 \\ 10.0 & \text{if } B_t[i] = 1 \wedge C_t[i] < 3 \\ 3.5 & \text{if } B_t[i] = 1 \wedge 3 \leq C_t[i] < 16 \\ 0 & \text{if } B_t[i] = 1 \wedge 16 \leq C_t[i] \end{cases} \tag{66}$$

Another thing to address is, that empirically women tend to work less when they are young (in the age of 18 to 30), gradually increasing the number of working hours as shown in figure 13. Looking to figure 16a, it could appear to be a consequence of women doing full time education, not having the time to work (more than a few hours a week). To address this problem I add an education variable to the model. In this setup the education does not add to higher salary instead it only goes in the model as additional time spent. Education is assumed to be exogenous, and over the life cycle the women will with some probability be done with education, not to return ever again to the education system. Using **FOLK1A** from Statistics Denmark I find the total number of women in a given age group. The total number of women for a given age group doing full time education is found in the source **UDDAKT10** also from Statistics Denmark. I fit this with a linear regression and let the number of women doing full time education decrease linearly from age 18 to 30. This can be seen in Figure 16b. I model the education the following way:

Figure 16: Women and Full Time Education (FTE)



(a) FTE vs. Total Number of Women



(b) Fraction of Women in FTE vs. Linear Fit

$$E_{t+1} = E_t \cdot \iota_t, \qquad \iota_t \mid Q_t \sim Binomial(p_\iota(Q_t)) \tag{67}$$

So in each period with some probability $E_t^{prob} \equiv p_\iota(Q_T)$ the women will end full time education. Both E and E_t^{prob} is part of the state space. Since women do full time education, I assume time used is 37 hours pr. week.

It is assumed for simplicity that human capital accumulation does not depreciate or accumulate when the agent is under full time education:

$$G_{t+1} = \begin{cases} G_t(1-\delta) + \frac{H_t}{37}, & \text{if } E_t > 0 \\ G_t & \text{else} \end{cases} \tag{68}$$

Finally note that the utility is calculated pr. week basis instead of on an annual basis. This is again done, such that the sub utilities, do not squeeze the values too hard, as was the case in model 1, making estimating of the model very hard, due to small perturbations of β_L having too large effect. Compared to the original model $f^M(Q_t)$ now represents the number total income pr. week of the husband rather than the annual total income.

$$L_t = 24 \cdot 7 - H_t - J_t - E_t \cdot 37 \tag{69}$$

In Denmark education is coupled with a transfer from the state. This transfer is added as well, such that the household will a transfer of $tr = \frac{6000DKK \times 12}{52} \approx 1400DKK$ each week from the government, while the women is studying. This ratio is approximately doubled if the women gets a child during the education (*Satser for tillæg til forældre - su.dk* 2020). The total income of the household is equal to:

$$Y_t = H_t \cdot W_t + f^M(Q_t) + E_t(tr + \mathbf{1}\{K_t > 0\} \cdot tr) \tag{70}$$

Summarizing the model; using a lot of the framework from the simple model a couple of extensions is made. The state space is now of 14 dimensions containing (Q) age, (G) human capital, (Z) idiosyncratic wage path, (K) the number of children in at time t in the household, a vector of size four (B) corresponding to the number of children the household contain, a vector of size four (C) containing the age of the individual children, a dummy (E) that indicates if the women is under education, and lastly (E^{prob}) a number between 0 and 1 that indicates the probability of being enrolled in full time education next period. The action space is slightly tweaked to contain $\{0, 15, 25, 37\}$ representing the discreet choices of hours the woman can choose to work. This formulation is closer to that of Francesconi (2002), where he only considers

the choices {nonwork, part-time, full-time}. In contrast with Francesconi (2002) this model distinguishes between two types of part-time work; 15 and 25 hours a week. The model specification is:

$$\textbf{State space: } \mathcal{S} = \mathbb{R}^2 \times \{0,1,2,3,4\} \times \{18,19,\cdots,60\} \times \{0,1\}^4 \times \{0,1,\cdots,18\}^4 \times \{0,1\} \times [0,1] \quad (71)$$

$$\textbf{Action space: } \mathcal{A} = \{0,15,25,37\} \quad (72)$$

Additionally the model contains the following parameters: $\alpha = 4.609, \eta_G = 0.164, \eta_{G^2} = 0.015, \delta, \sigma_\epsilon = 15.11, \beta_L, \beta_Y = 1, W_{min} = 120, tr = 1600, \zeta = 0.5$, using the values of the Mincer equation from the parameter calibration performed earlier! A recursive formulation of the model is presented below:

$$U_t(L_t, Y_t) = \beta_L L_t^\zeta + \beta_Y Y_t^\zeta \quad (73)$$

$$L_t(K_t, H_t, J_t, E_t) = 24 \cdot 7 - H_t - J_t - E_t \cdot 37 \quad (74)$$

$$\log \tilde{W}_t(G_t) = \alpha + \eta_G G_t + \eta_{G^2} G_t^2 \quad (75)$$

$$W_t(\tilde{W}_t, Z_t) = \max(W_{min}, \tilde{W}_t + Z_t) \quad (76)$$

$$Y_t(Q_t, H_t, W_t, E_t, K_t) = 46 \cdot H_t \cdot W_t + f^M(Q_t) + E_t(tr + \mathbf{1}\{K_t > 0\} \cdot tr) \quad (77)$$

$$J_t(B_t, C_t) = \sum_{i=1}^{4} \begin{cases} 0.0 & \text{if } B_t[i] = 0 \\ 10.0 & \text{if } B_t[i] = 1 \wedge C_t[i] < 3 \\ 3.5 & \text{if } B_t[i] = 1 \wedge 3 \geq C_t[i] < 16 \\ 0.0 & \text{if } B_t[i] = 1 \wedge 16 \geq C_t[i] \end{cases} \quad (78)$$

$$B_t[i](K_t) = \begin{cases} 1 & \text{if } i \leq K_t \\ 0 & \text{else} \end{cases} \quad (79)$$

Law of motion:

$$Q_{t+1}(Q_t) = Q_t \tag{80}$$

$$K_{t+1}(K_t, Q_t) = K_t + \psi_t, \qquad \psi_t \mid Q_t \sim Bernoulli(p_\psi(Q_t)) \tag{81}$$

$$Z_{t+1}(Z_t) = Z_t + \epsilon_t, \qquad \epsilon_t \sim \mathcal{N}(0, \sigma_\epsilon) \tag{82}$$

$$G_{t+1}(G_t, H_t, E_t) = \begin{cases} G_t(1-\delta) + \frac{H_t}{37}, & \text{if } E_t > 0 \\ G_t & \text{else} \end{cases} \tag{83}$$

$$C_{t+1}(C_t, B_{t+1}) = C_t + B_{t+1} \tag{84}$$

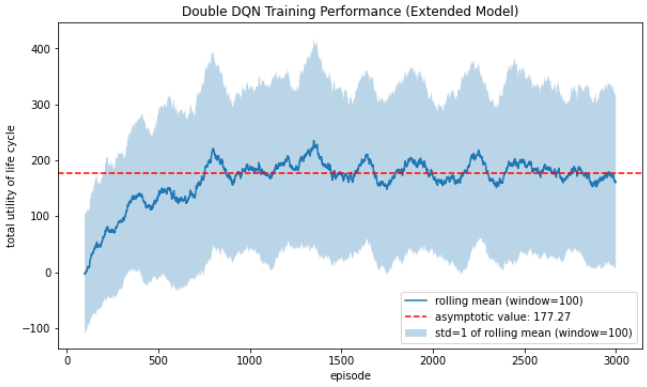
$$E_{t+1}(E_t, Q_t) = E_t \cdot \iota_t, \qquad \iota_t \mid Q_t \sim Binomial(p_\iota(Q_t)) \tag{85}$$

11 Solving and Estimating the Extended Model

11.1 Solution

For this extended model I only use the Double Deep Q-Network agent, since it was shown with the simpler environment, to have comparable performance to the VFI implementation and Deep Q-network. The model is solved by running 3000 episodes. As was true for the simple model, the deep learning algorithm has the best performance if states are scaled mean 0 and a standard deviation of 1. In practice this is done by implementing a transformer in my agent that scales the states. The transformer is also capable of doing an inverse transformation. The rewards are also scaled. This slightly improves performance, but the primary reason is, that it helps track the learning. I scale the rewards to have a variance of about 1, and let them have the same mean conditional of the β_L value! I train the algorithm for 3000 episodes, have the hyper parameters of the algorithm being: $\gamma = 0.99$ representing the impatience of the agent. The agent is initialized with $\epsilon = 1.0$ which decays by 0.9999 for each step the agent performs. The learning rate is $\alpha = 0.0005$. The size of the memory buffer is one million rows, and I train with a batch size=64. I let the minimum exploration rate in training be 0.01. The architecture of the network is identical to the one used previously: An Input layer of same size as the state space, 2 fully connected layers of size 256 with rectified linear units as activation functions, and a linear output layer of the same size as the action space. At the beginning of each episode a random $\beta_L \sim Uniform(0, 60)$ is drawn, allowing the agent to step through an episode until termination with given β_L. Finally the target network inherits the policy weights every 100'th step. A score for every 50'th episode is calculated, taking the average of the previous 50 episodes. If the current iteration of the model beats the high score, the weights of the neural network is saved. The estimation and simulation is performed with the weights that yield the highest score for 50 consecutive episodes.

Figure 17: Estimation of β_L for extended model



Looking to figure 17 the agent clearly learns to navigate in the environment. After about 1000 episodes the agent seems to have learned to navigate the environment! The maximum score appears at around episode 1400, which is the weights used for estimation and simulations. The asymptotic training performance is about 177 comparing to an initial score of approximately zero.

11.2 Estimation

As was the case with the simpler model, only a single parameter β_L needs to be estimated. Again, just as was the case before with the simpler model, I extend the state space to contain β_L. I use a grid search in the range $\beta_L \in [0, 60]$, simulating $N = 800$ observations, calculating the objective function. I use the same seed in each iteration of the optimization process. To address the short comings of the simple model I expand the objective function of the optimization problem. In the initial model too many women choose not to be part of the labour force yielding unrealistic results. In response to this problem the new objective function extends to two broad goals: Let the right number of women be out of the labour force (around 15 %) and fit the curve of number of working hours for women. The first objective, *objective 1*, is formulated as:

$$\text{objective 1} = \left[\frac{\sum_{i=1}^{N} \sum_{q=Q_{\min}}^{Q_{\max}} \mathbf{1}\{H_{i,q} = 0\}}{N \cdot (Q_{\max} - Q_{\min})} - 15\% \right]^2 \tag{86}$$

The second objective is formulated the same way as it was when first estimating the simple model, which correspond the conditional expectation of supplied number of working hours conditional on the age and of being in the labour force $\mathbb{E}[H \mid Q = q, H > 0]$. The desired outcome is for this to be true for all ages from 18 to 60, μ_q is the empirical moment found using the data **LIGEF15**:

$$\text{Objective 2} = \sum_{q=Q_{\min}}^{Q_{\min}} \left[\mu_q - \frac{1}{\sum_{i=1}^{N} \mathbf{1}\{H_{i,q} > 0\}} \sum_{i=1}^{N} (H_{i,q}) \right]^2 \tag{87}$$

The description of the objectives can be translated into a more formal formulation. The number of moments are # moments $= 1 + (Q_{\max} - Q_{\min}) = 1 + 60 - 18 = 43$. The weighing of these moments would usually be done by some weight matrix W in an equation of the form: $[\hat{m} - \tilde{m}]^\top W^{-1} [\hat{m} - \tilde{m}]$, where $\tilde{m}$ is a vector of empirical moments and $\hat{m}$ is a vector of simulated moments allowing for the methods of moments estimation. The choosing of the weight matrix is application specific. Eisenhauer, Heckman, and Mosso (2015) suggests the choice in that setting should be the inverse variance of the empirical moments $\tilde{m}_j$, on the diagonal of the weight matrix (zeros else), letting the j'th moment correspond to the j'th weight. This is however not possible in this application, due to the fact, that I only have access to aggregated data from statistics Denmark. Instead, I manually choose scales such that *objective 1* is equally weighted to *objective 2*. This first and foremost requires a scaling of the moments. And second of all this requires a re-weighing since there are 42 moments composing *objective 2* and only a single moment composing *objective 1*. Since the distance between the empirical and the simulated moment is squared, the scaling must be performed before the squaring. Finally, since the weight matrix is chosen as it is, a transformation can be performed such that instead of a matrix product it can be considered a sum: $\sum_{j=1}^{43} (\hat{m}_j - \tilde{m}_j)^2$, becoming a mean squared error optimization problem. It is implied that N agents is simulated conditional on a value of β_L. The objective function is:

$$\begin{aligned} \text{Objective}(\beta_L) = &\left[37 \cdot \left(\frac{\sum_{i=1}^{N} \sum_{q=Q_{\min}}^{Q_{\max}} \mathbf{1}\{H_{i,q} = 0\}}{N \cdot (Q_{\max} - Q_{\min})} - 15\% \right) \right]^2 \\ &+ \frac{1}{Q_{\max} - Q_{\min}} \sum_{q=Q_{\min}}^{Q_{\max}} \left[\mu_q - \frac{1}{\sum_{i=1}^{N} \mathbf{1}\{H_{i,q} > 0\}} \sum_{i=1}^{N} (H_{i,q}) \right]^2 \end{aligned} \tag{88}$$

The estimated value of $\beta_L = 24.49$. Figure 18 shows the grid search, and finds that optimization problem, contains a minimum at $\beta_L = 24.49$, suggesting that the range of the grid is adequate for the estimation problem. I take the $\log(\cdot)$ of the mean squared error to better represent it in figure 18.

12 Results of the Extended Model

I simulate 200.000 households using the optimal $\beta_L = 24.49$. Using the simulations a counter factual analysis of households that do get children, compared to households that do not can be made. Figure 19 show the average number of working hours of the simulations compared to the true number of hours worked by women. This extended model fits the data considerably better than the simple model, as shown in figure

Figure 18: Estimation of β_L for extended model

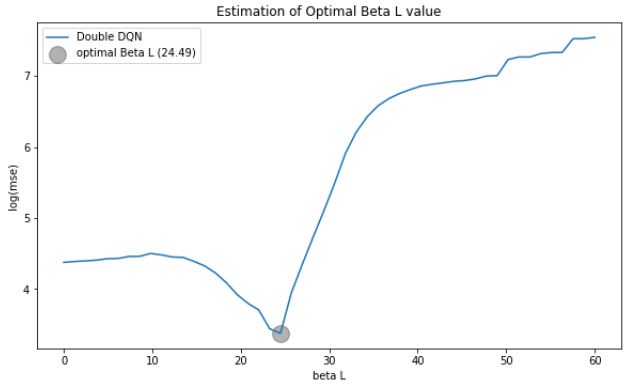
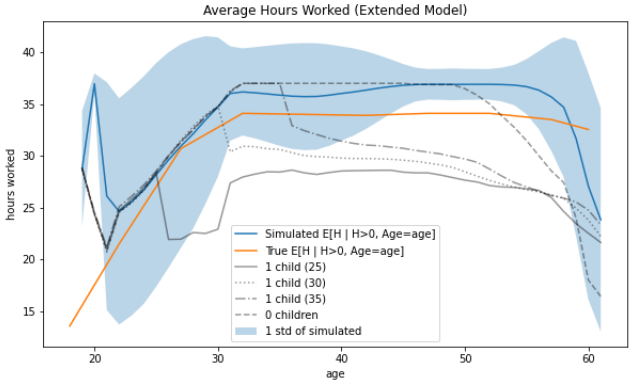


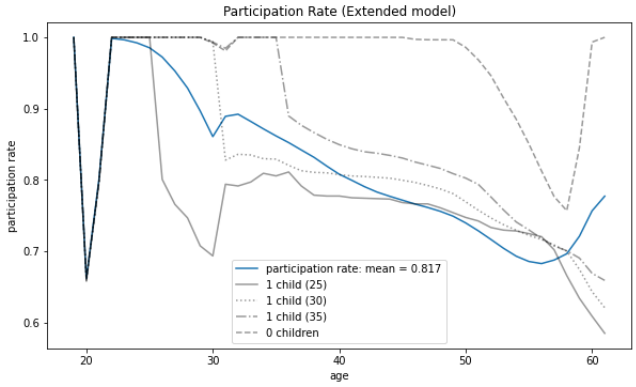
Figure 19: Extended Model Working Hours - Empirical Vs. Simulated



13. Again the models also differ in the action space, where this extended model do not allow for working more than 37 hours a week. The average number of hours worked is conditional on the supplied number of hours working must be above 0. The figure also shows the average number of hours supplied by women when they get children at their specific age. This is true for age 25, 30 and age 35. These households get one child and only one child, where I do not condition on $H > 0$. Looking to these households steep dips in the supplied number of hours are present, at about 5 hours for each of the three age groups. The cohorts of households that get a child at age 30 and 35 year dip to a permanently lower number of supplied hours, whereas the cohort that get a child at age 25, get an initial dip, but readjust after a certain period. For all three cohorts the initial dip the households experience has a magnitude of about 5 hours.

Figure 20 shows the participation rate over the life cycle. The average participation of all simulations is 81.7% letting the overall number of women out of the workforce be equal to 18.3. Comparing to the number used for the estimation of β_L, which was 15%, the results must be considered a good fit. I have not

Figure 20: Participation rates

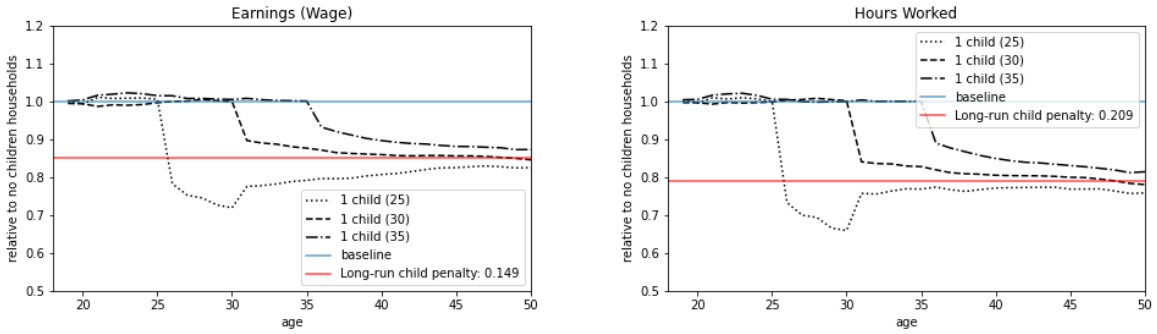


found good data from Statistics Denmark showing women's average participation rate over the life cycle, implying it is hard to make inference over the participation rate over the life cycle. However a couple of things can be noted. First and foremost the simulations show a downward sloping trend of participation rates over the life cycle. Grimm and Bonneuil (2001) have investigated female labour participation rate in France over the life cycle. The data they present suggests this picture is not entirely unrealistic. Certain things should be noted about the comparison. Their sample ends in 1998, and looking to that data, the downward sloping trend do seem to be less pronounced comparing to earlier generations. I have shown four different cohorts in the same figure. As before I have focused on families getting 0 children, a cohort of households getting a child at age 25, a cohort of households getting a child at age 30, and lastly a cohort of households getting their first and only child at age 35. The picture is clear, when the child is born, it has a very strong effect on participation rate. Households getting a child at age 25, do seem to have a penalty of 30 percent reduction in participation rate for the next five years. Families getting their child at age 30 experience a reduction in participation rate slightly below 20 percent, and at last the cohort of households getting a child at age 35 experience a reduction in participation rate at around 10 percent.

Figure 21 and 22 displays event graphs of households getting a (single) child at different ages. These are compared against the women of households that do not get children. These event graphs compares to the event graphs by Kleven, Landais, and Søgaard (2019), however there are a couple of differences. Kleven, Landais, and Søgaard (2019) considers the husband of the household as benchmark whereas I consider women of households with no children the benchmark. The husband of the household, in this model, is considered to follow a deterministic path, and household heterogeneity only comes from the different choices of the woman of the household, letting this be a more natural comparison for this model. Comparing the results, even though the benchmarks do diverge, does lead to surprisingly similar conclusions. First notice that the earnings do seem to have the largest reduction for households that get children at an early age. The same goes for hours worked, participation rates and wage rates. Secondly the long run penalties do

converge to approximately the same levels. Table 3 lists the different long run penalties compared to Kleven, Landais, and Søgaard (2019). Comparing the long run penalties, this analysis find 5 percentage points lower long run penalties for earnings, 10 percentage points higher penalty for hours works, 8 percentage points higher penalties for participation rates and 14 percentage points lower penalty for wage rates in this model than the results found by Kleven, Landais, and Søgaard (2019).

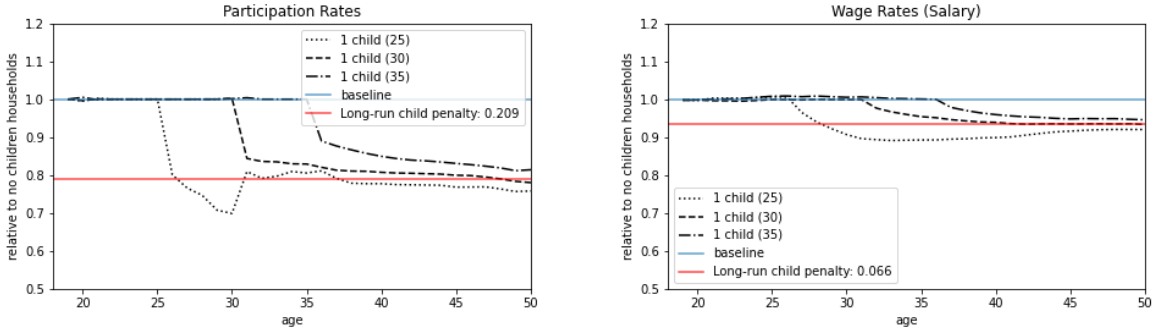
Figure 21: Impact of Children (Earnings and Worked Hours)



(a) Earnings

(b) Hours Worked

Figure 22: Impact of Children (Participation Rates and Wage Rates)

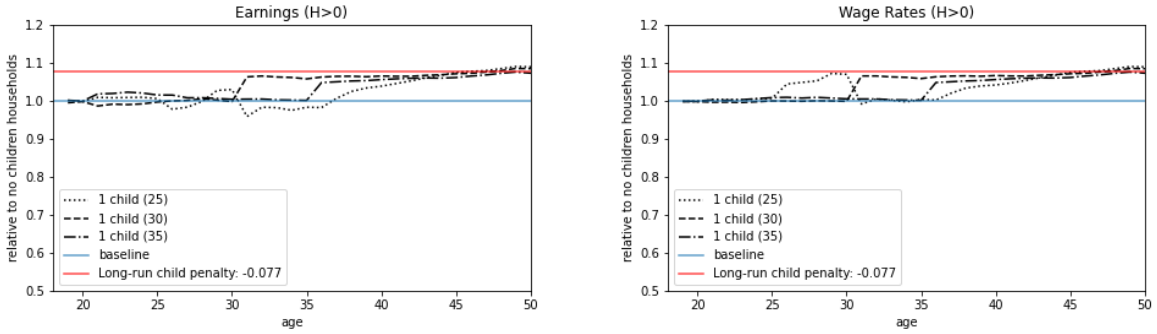


(a) Participation Rates

(b) Wage Rates

Kleven, Landais, and Søgaard (2019) writes that the event graphs they present of wage rates and hours worked is conditional on labour market participation. Figure 23 shows comparable event graphs of the simulations. Here I find the reverse picture, that on average the wage rates and hours worked increase when a child is born. This effect is a consequence of selection effects. In other words, this model primarily finds effects, by letting women select out of the labour market, if they are not on high income trajectories. This clashes with the results found by Kleven, Landais, and Søgaard (2019). The RHS column of table 3 shows the long run penalties, conditioning on labour market participation.

Figure 23: Impact of Children conditional on $H > 0$



(a) Earnings Conditional on $H > 0$

(b) Wage Rates Conditional on $H > 0$

Multiple issues can be attributed to cause these discrepancies between the model presented in this paper and the results found by Kleven, Landais, and Søgaard (2019). First, it cannot be rejected that in real danish registry data, labour market participation is defined in a less naive way, than in this paper. This paper simply assumes, that if a woman supplies zero hours of work $H = 0$, then said woman is unemployed. Danish registry data have a more intricate way to ascribe unemployment status to individuals, than just looking at the number of hours worked by an individual, implying that some individuals of the sample in Kleven, Landais, and Søgaard (2019) could be working 0 hours. Secondly this model naively assumes an idiosyncratic wage path following a random walk, this might also contribute to the issues. In general, I believe, that to more accurately find comparable results to Kleven, Landais, and Søgaard (2019) then the model, should contain more intricate family characteristics, endogenous male labour supply, a more elaborate labour market with multiple sectors and include education in the income process. This would compound to a very complex model, however as shown in this paper, using deep reinforcement learning for solving dynamic models, could make it feasible.

Table 3: Long run penalties comparison

	Kleven et al.	result	result ($H > 0$)
Earnings	0.194	0.149	-0.077
Hours worked	0.097	0.209	0.000
Participation rates	0.130	0.209	NaN
Wage rates	0.194	0.066	-0.077

13 Conclusion

This paper has tried to establish how reinforcement learning can be used to solve dynamic models. I show that using these solution methods allow for solving models otherwise computationally infeasible. I argue that when using such methods, there is no guarantee of finding the optimal policy, and there exists a real risk of ending up in local maxima. Furthermore have I presented an economic model of exogenous fertility and endogenous female labour supply, that can give a surprisingly good fit to data, and show comparable long term penalties for female participation rates, earnings, wage rates and hours worked to contemporary findings by Kleven, Landais, and Søgaard (2019), when a household gets a child.

The first part of the paper, I draw on the literature of female labour supply and fertility to formulate a simple dynamic model of female labour supply and fertility, tracking households over the life cycle, where women can supply a discreet number of hours to the labour force. I use aggregate data for Statistics Denmark to calibrate the parameters of the income process. For this task I assume that the households follows a deterministic strategy with regards to labour supply. I find, the model seem to approximate the income paths of both men and women very well. Next, I present the reader for reinforcement learning and deep learning allowing the reader to have an understanding of the algorithms used for solving the models presented in this paper. I go on to present three different solution methods: Value function Iteration using deep neural networks for value function approximation, Deep Q-learning and Double Deep Q-Learning. I find that these three solution methods have comparable performance, but the model does not seem to fit the data well! I formulate an extension to the inadequate model, and use Double Deep Q-learning as solution method. I estimate the extra parameters using method of simulated moments. Simulating from the model using the estimated parameters, I find that both the participation rate and the average number of supplied hours to the labour force is comparable to what is found in data. Finally I compare event graphs to those found by Kleven, Landais, and Søgaard (2019) and find striking similarities, finding that women do experience a real penalty when getting a child. I find a long-term penalty of 15 % on earnings for women with one child where Kleven, Landais, and Søgaard (2019) find a 19 % penalty and a long term reduction of 21 % in the number of hours worked compared to 10 % found by Kleven, Landais, and Søgaard (2019).

I hope this paper gives a convincing argument for the use of reinforcement learning in the field of economics. When economists formulates models the solution method often constraints what is explored. After writing this paper, I have come to believe, that reinforcement learning can be a way for exploring what previously has been assumed to be unsolvable using contemporary methods.

References

[1] Jerome Adda, Christian Dustmann, and Katrien Stevens. *The Career Costs of Children*. SSRN Scholarly Paper ID 1976532. Rochester, NY: Social Science Research Network, Dec. 24, 2011. URL: https://papers.ssrn.com/abstract=1976532 (visited on 12/10/2019).

[2] Sumru Altuğ and Robert A. Miller. "The Effect of Work Experience on Female Wages and Labour Supply". In: *The Review of Economic Studies* 65.1 (Jan. 1, 1998), pp. 45–85. ISSN: 0034-6527. DOI: 10.1111/1467-937X.00035. URL: https://academic.oup.com/restud/article/65/1/45/1589921 (visited on 12/10/2019).

[3] Joshua D Angrist and William N Evans. *Children and Their Parents' Labor Supply: Evidence from Exogenous Variation in Family Size*. Working Paper 5778. National Bureau of Economic Research, Sept. 1996. DOI: 10.3386/w5778. URL: http://www.nber.org/papers/w5778.

[4] Orazio Attanasio, Peter Levell, et al. "Aggregating Elasticities: Intensive and Extensive Margins of Women's Labor Supply". In: *Econometrica* 86.6 (2018), pp. 2049–2082. ISSN: 1468-0262. DOI: 10.3982/ECTA15067. URL: https://onlinelibrary-wiley-com.ep.fjernadgang.kb.dk/doi/abs/10.3982/ECTA15067.

[5] Orazio Attanasio, Hamish Low, and Virginia Sánchez-Marcos. "Explaining Changes in Female Labor Supply in a Life-Cycle Model". In: *American Economic Review* 98.4 (Sept. 2008), pp. 1517–1552. ISSN: 0002-8282. DOI: 10.1257/aer.98.4.1517. URL: https://www.aeaweb.org/articles?id=10.1257/aer.98.4.1517 (visited on 12/13/2019).

[6] Richard Blundell et al. "Female Labor Supply, Human Capital, and Welfare Reform". In: *Econometrica* 84.5 (2016), pp. 1705–1753. ISSN: 1468-0262. DOI: 10.3982/ECTA11576. URL: https://onlinelibrary.wiley.com/doi/abs/10.3982/ECTA11576.

[7] Daniela Del Boca, Silvia Pasqua, and Chiara Pronzato. "Motherhood and market work decisions in institutional context: a European perspective". In: *Oxford Economic Papers* 61 (suppl_1 Apr. 1, 2009), pp. i147–i171. ISSN: 0030-7653. DOI: 10.1093/oep/gpn046. URL: https://academic.oup.com/oep/article/61/suppl_1/i147/2568573 (visited on 12/13/2019).

[8] Zvi Eckstein and Osnat Lifshitz. "Dynamic Female Labor Supply". In: *Econometrica* 79.6 (2011), pp. 1675–1726. ISSN: 1468-0262. DOI: 10.3982/ECTA8803. URL: https://www.onlinelibrary.wiley.com/doi/abs/10.3982/ECTA8803.

[9] Philipp Eisenhauer, James J. Heckman, and Stefano Mosso. "ESTIMATION OF DYNAMIC DISCRETE CHOICE MODELS BY MAXIMUM LIKELIHOOD AND THE SIMULATED METHOD OF MOMENTS: ESTIMATION OF DYNAMIC MODELS". In: *International Economic Review* 56.2

(May 2015), pp. 331–357. ISSN: 00206598. DOI: 10.1111/iere.12107. URL: http://doi.wiley.com/10.1111/iere.12107 (visited on 05/15/2020).

[10] Olivia Ekert-Jaffé and Shoshana Grossbard. "Time Cost of Children as Parents' Foregone Leisure". In: *Mathematical Population Studies* 22.2 (Apr. 3, 2015), pp. 80–100. ISSN: 0889-8480. DOI: 10.1080/08898480.2013.836332. URL: https://doi.org/10.1080/08898480.2013.836332 (visited on 02/17/2020).

[11] Juanita Firestone and Beth Anne Shelton. "An Estimation of the Effects of Women's Work on Available Leisure Time". In: *Journal of Family Issues* 9.4 (Dec. 1, 1988), pp. 478–495. ISSN: 0192-513X. DOI: 10.1177/019251388009004004. URL: https://doi.org/10.1177/019251388009004004 (visited on 02/17/2020).

[12] Marco Francesconi. "A Joint Dynamic Model of Fertility and Work of Married Women". In: *Journal of Labor Economics* 20.2 (2002), pp. 336–380. ISSN: 0734-306X. DOI: 10.1086/338220. URL: www.jstor.org/stable/10.1086/338220.

[13] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Vol. 1. 10. Springer series in statistics New York, 2001.

[14] George-Levi Gayle and Robert A. Miller. *Life-Cyle Fertility Behavior and Human Capital Accumulation*. 784. Society for Economic Dynamics, Dec. 3, 2006. URL: https://ideas.repec.org/p/red/sed006/784.html (visited on 12/10/2019).

[15] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. Cambridge, Massachusetts: The MIT Press, Nov. 18, 2016. 800 pp. ISBN: 978-0-262-03561-3.

[16] Michael Grimm and Noël Bonneuil. "Labour Market Participation of French Women over the Life Cycle, 1935–1990". In: *European Journal of Population / Revue européenne de Démographie* 17.3 (Sept. 1, 2001), pp. 235–260. ISSN: 1572-9885. DOI: 10.1023/A:1011873830194. URL: https://doi.org/10.1023/A:1011873830194 (visited on 05/17/2020).

[17] Peter Haan and Katharina Wrohlich. "Can child care policy encourage employment and fertility? Evidence from a structural model". In: *IDEAS Working Paper Series from RePEc; St. Louis* (2009). URL: https://search-proquest-com.ep.fjernadgang.kb.dk/docview/1698255965?rfr_id=info%3Axri%2Fsid%3Aprimo.

[18] Hado van Hasselt, Arthur Guez, and David Silver. "Deep Reinforcement Learning with Double Q-learning". In: *arXiv:1509.06461 [cs]* (Dec. 8, 2015). arXiv: 1509.06461. URL: http://arxiv.org/abs/1509.06461 (visited on 04/07/2020).

[19] V. Joseph Hotz and Robert A. Miller. "An Empirical Analysis of Life Cycle Fertility and Female Labor Supply". In: *Econometrica* 56.1 (1988), pp. 91–118. ISSN: 0012-9682. DOI: 10.2307/1911843. URL: www.jstor.org/stable/1911843.

[20] Thomas H. Jørgensen. "Life-Cycle Consumption and Children: Evidence from a Structural Estimation". In: *Oxford Bulletin of Economics and Statistics* 79.5 (2017), pp. 717–746. ISSN: 1468-0084. DOI: 10.1111/obes.12170. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/obes.12170.

[21] Michael P. Keane and Kenneth I. Wolpin. "The Role of Labor and Marriage Markets, Preference Heterogeneity, and the Welfare System in the Life Cycle Decisions of Black, Hispanic, and White Women*". In: *International Economic Review* 51.3 (2010), pp. 851–892. ISSN: 1468-2354. DOI: 10.1111/j.1468-2354.2010.00604.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1468-2354.2010.00604.x.

[22] Henrik Kleven, Camille Landais, and Jakob Egholt Søgaard. "Children and Gender Inequality: Evidence from Denmark". In: *American Economic Journal: Applied Economics* 11.4 (Oct. 2019), pp. 181–209. ISSN: 1945-7782. DOI: 10.1257/app.20180010. URL: https://www.aeaweb.org/articles?id=10.1257/app.20180010 (visited on 12/13/2019).

[23] Astrid Kunze. *The Timing of Careers and Human Capital Depreciation*. SSRN Scholarly Paper ID 319961. Rochester, NY: Social Science Research Network, June 1, 2002. URL: https://papers.ssrn.com/abstract=319961 (visited on 05/01/2020).

[24] Thomas Lemieux. "The "Mincer Equation" Thirty Years After Schooling, Experience, and Earnings". In: *Jacob Mincer A Pioneer of Modern Labor Economics*. Ed. by Shoshana Grossbard. Boston, MA: Springer US, 2006, pp. 127–145. ISBN: 978-0-387-29175-8. DOI: 10.1007/0-387-29175-X_11. URL: https://doi.org/10.1007/0-387-29175-X_11 (visited on 12/21/2019).

[25] Audrey Light and Manuelita Ureta. "Early-Career Work Experience and Gender Wage Differentials". In: *Journal of Labor Economics* 13.1 (1995). Publisher: [University of Chicago Press, Society of Labor Economists, NORC at the University of Chicago], pp. 121–154. ISSN: 0734-306X. URL: https://www.jstor.org/stable/2535309 (visited on 05/01/2020).

[26] Volodymyr Mnih et al. "Playing Atari with Deep Reinforcement Learning". In: 2013 (2013), p. 9. URL: https://www.cs.toronto.edu/~vmnih/docs/dqn.pdf.

[27] Robert Moffitt. "The Estimation of a Joint Wage-Hours Labor Supply Model". In: *Journal of Labor Economics* 2.4 (Oct. 1, 1984), pp. 550–566. ISSN: 0734-306X. DOI: 10.1086/298047. URL: https://www.journals.uchicago.edu/doi/abs/10.1086/298047.

[28] *Satser for tillæg til forældre - su.dk*. URL: https://www.su.dk/su/saerlig-stoette-til-foraeldre-handicappede-mv/foraeldre/satser-for-tillaeg-til-foraeldre/ (visited on 05/14/2020).

[29] David Silver et al. "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play". In: *Science* 362.6419 (Dec. 7, 2018). Publisher: American Association for the Advancement of Science Section: Report, pp. 1140–1144. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/

science.aar6404. URL: https://science.sciencemag.org/content/362/6419/1140 (visited on 05/20/2020).

[30] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: an introduction*. Second edition. Adaptive computation and machine learning series. Cambridge, Massachusetts: The MIT Press, 2018. 526 pp. ISBN: 978-0-262-03924-6.

[31] Phil Tabor. *Deep Q-Learning Implementation*. original-date: 2018-08-09T00:21:29Z. May 27, 2020. URL: https://github.com/philtabor/Youtube-Code-Repository (visited on 05/27/2020).

[32] Christer Thrane. "Men, Women, and Leisure Time: Scandinavian Evidence of Gender Inequality". In: *Leisure Sciences* 22.2 (Apr. 1, 2000), pp. 109–122. ISSN: 0149-0400. DOI: 10.1080/014904000272885. URL: https://doi.org/10.1080/014904000272885 (visited on 02/17/2020).

[33] J.N. Tsitsiklis and B. Van Roy. "An analysis of temporal-difference learning with function approximation". In: *IEEE Transactions on Automatic Control* 42.5 (May 1997), pp. 674–690. ISSN: 00189286. DOI: 10.1109/9.580874. URL: http://ieeexplore.ieee.org/document/580874/ (visited on 04/07/2020).

[34] Oriol Vinyals et al. "Grandmaster level in StarCraft II using multi-agent reinforcement learning". In: *Nature* 575.7782 (Nov. 2019). Number: 7782 Publisher: Nature Publishing Group, pp. 350–354. ISSN: 1476-4687. DOI: 10.1038/s41586-019-1724-z. URL: https://www.nature.com/articles/s41586-019-1724-z (visited on 05/20/2020).

Appendix

List of Figures

List of Tables

Figure 24: Event Graphs of Kleven, Landais, and Søgaard (2019)

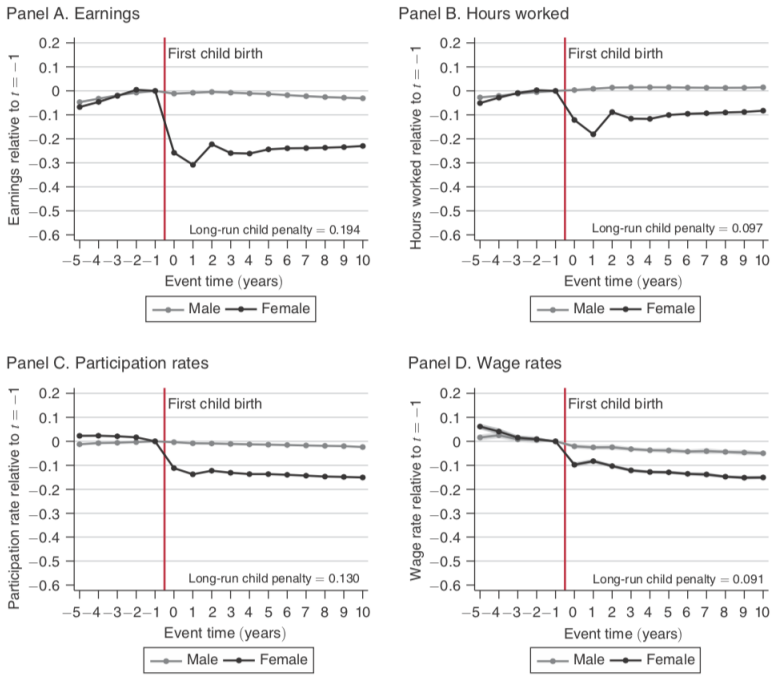


FIGURE 1. IMPACTS OF CHILDREN

Figure 25: Fertility of Women