Modelling Financial Crisis with Deep Reinforcement Learning

Georgi Demirev[a]

[a]Bulgarian National Bank Scholarship Recipient

This manuscript was compiled on August 19, 2018

This article presents a macroeconomic model of bank intermediation based on multi agent reinforcement learning. The model consists of heterogeneous households, banks, and firms, each acting to maximize individual utility. Agents acquire their policy autonomously via double Q learning. Simulating the model yields endogenous credit-driven business cycles. Two experiments are carried out in the simulated environment: the first one introduces deposit guarantees, which leads to lower interest rates on deposits; the second one varies the required reserve ratio for banks, and reveals that at low levels of required reserves household savings decrease dramatically. Several suggestions for future research on reinforcement-learning-based economic models are presented.

Agent Based Models | Reinforcement Learning | Financial crisis

Contemporaneous macroeconomic models tend to predominantly be dynamic stochastic general equilibrium models, following the pioneering work of Kydland and Prescott (1). Yet, despite their widespread academic success, DSGE models have recently come under heavy criticism in the face of the perceived failure to predict and explain the 2008 financial crisis (2).

One of the major points of criticism is the over-dependence of DSGE models on exogenous shocks - be it in productivity, preference, capital formation, etc (3). In this view, the reliance on external (unexplained) forces greatly diminishes the models' scientific usefulness, as the major driving forces are often attributed to mere chance.

A second line of criticism points to the critical role that equilibrium solution concepts play in such models. On one hand general equilibrium theory has been largely abandoned by microeconomists, who found it to be neither stable nor unique (4). On the other, as shown by Papadimitriou, finding a Brouwer fixed point (and hence a solution to GE exchange problem) is a likely non-polynomial time task (5). Coupled with the computational complexity of dynamic programming problems with complex multi-dimensional state dynamics, the standard DSGE model imposes rather unrealistic requirements on agents' cognitive prowess. This line of attack is often taken up by proponents of bounded rationality models.

Several alternatives to DSGEs have been put forth, one of the more promising of which being agent-based models. ABMs shun equilibrium concepts and instead try to simulate the individual behavior of many (possibly heterogenous) agents and their interactions (6). The modellers then study the resulting simulated economy to arrive at conclusions about the real world.

Despite their promising premise, ABMs too have come under fair amount of criticism. They often rely on ad-hoc rules-of-thumb to explain individual behavior (7). In contrast traditional macro models attempt to base themselves into the "micro foundations" of the individual agent's optimization. Furthermore, a typical agent-based simulation may have myriad of parameters, making it hard to analyze coherently (8). This arbitrary nature of ABMs is one of the main reasons for them to remain outside the academic mainstream.

In this paper I attempt to unify the agent-based approach of simulating all interactions between agents with the individual optimization assumption of DSGE models. A natural way to do this is by employing concepts and techniques from reinforcement learning - the sub-field of artificial intelligence that deals with adaptive decision making.

In order to do so I develop a model of financial intermediation with three types of agents - households, firms, and banks. All agents in the economy maximize individual utility as in a DSGE model. However instead of assuming that agents behave as per the solution of a dynamic programming problem, they are endowed with a learning mechanism that tries to approximate this optimal solution. This is done by a method known as double Q-learning with a deep neural network value function approximator.

This is not the first paper to employ reinforcement learning. Famously Roth and Erev (1995) use a simple reinforcement learning framework to model repeated games and Erev and Roth expand on this by comparing the implications of a reinforcement learning model to a number of experiments on games with mixed strategy Nash equilibria. More recently Fudenberg and Peysakhovich (2014) build a reinforcement learning model of a simple assymetric information game. There are also plenty of prior work on ABMs of the banking sector - (9), (10), (11), just to name a few.

However this paper is novel, as previous RL economic models tend to be much simpler (no function approximation) and previous bank ABMs tend to have no learning mechanism and rely on rules-of-thumb. Note a clear distinction from what is typically referred as "models of

This paper does not represent in any way the official views of the Bulgarian National Bank. All errors and omissions are to be attributed to the author.

[2]For correspondence: georgi.demirevbarcelonagse.eu

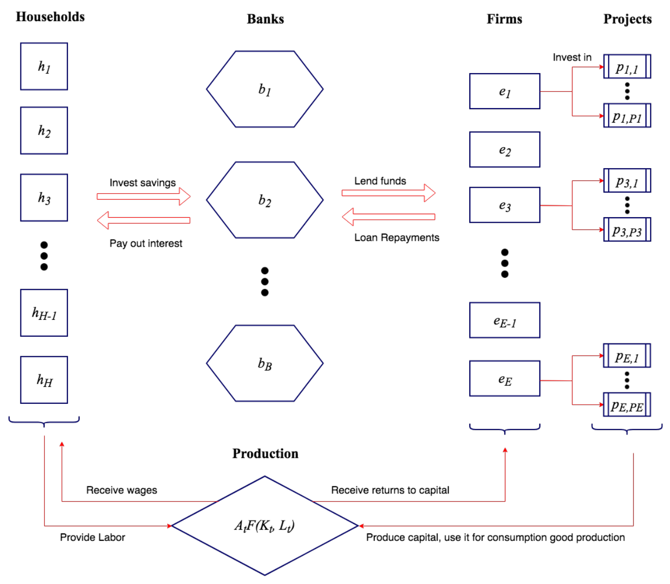

Fig. 1. Circular flow of the economy

learning" in economics: while the economic literature has most often focused on beliefs and predictions (12), reinforcement learning instead focuses on approximate optimization in the face of a high dimensional state space (or learning to optimize).

This paper borrows heavily concepts and algorithms from recent advances in computer science (13), (14). It aims to employ these novel methods in a self-contained agent-based macroeconomic model. The main objective of the paper is to serve as an early proof of concept of the usefulness of reinforcement learning in economics. As such the main focus is on the structure of the model itself - the results of the simulation are explored , but not in great detail.

Part 2 of the paper describes the agent based model and the interaction rules of the model economy. Part 3 gives more details about the reinforcement learning mechanism via which agents make decisions. Part 4 examines the emerging properties of the simulated economy. Part 5 details the result of two policy experiments in this economy. Part 6 presents some key issues that need to be solved in future research on reinforcement-learning-based economic models. Part 7 concludes.

Model Description

The model economy consists of three types of agents - banks, firms, and households. There are two goods - a consumption good and a capital good. Time passes in discrete periods.

Households provide labor for production and receive wages (in terms of the consumption good). They can either invest their wage or deposit it in a bank in return for interest (households posses no storage technology). Upon consumption households receive utility from a simple logarithmic utility function. Each household is restricted to hold deposits in only a single bank, but at each period households are free to withdraw their deposits and move them to another bank (or consume them).

Firms provide capital goods for production and receive a rental rate in terms of the consumption good. Capital is then decreased vi a depreciation rate, which for all results in the paper was set to 1. Firms have access to a production technology in the form of 'projects'. Each project costs a certain amount of the consumption good to set up and then brings in a random stream of the capital good for a fixed period of time. Firms have access to a new random project at each period. They can either discard the project, invest their own cash in it, or take out a loan to finance it. At each time point there is a probability that the project defaults and stops producing capital (any loans taken for the project however remain outstanding until repaid). Any leftover cash at the end of the period is consumed by the firm's management and utility is received. When firms cannot meet their loan repayment obligations they default - all their projects are terminated at a per-project termination income, and the proceedings go the the firm's creditors. The firm then begins next period on a clean slate.

Banks take in deposits and give out loans in exchange for interest. Each period banks set up the interest rates on loans and deposit. It is assumed that interest prices are rigid in the sense that a bank can only move the rate up or down by a fixed increment per period (the reasons for this are explained later on). Banks cannot refuse to take in a deposit if requested by a household, but can refuse to give out a loan. For each loan the expected sum of future capital inflows is calculated and then compared to the initial investment required. This 'capital per dollar' ratio is then compared to the bank's cut-off value and the loan is accepted only if it is above it. The cut-off is also controlled by the bank and can be moved in a similar manner as the interest rate. At the end of each period the bank receives utility equal to an exogenous percentage of the bank's capital (one can think of the manager receiving a bonus, or the shareholders receiving a dividend).

Each agent begins with an initial endowment of the consumption good (and a small endowment of the capital good for firms). In the case of banks the initial endowment represents their initial reserves. Each banks balance sheet consists of four items - cash reserves, outstanding loans, deposits, and capital. Capital is defined as the difference between assets (reserves and loans) and liabilities (deposits). Banks are subject to a reserve requirement - their reserves are not allowed to fall below a certain ratio of the deposits, and a capital requirement - their capital cannot fall below a certain ratio of the loans. If at any point a bank doesn't meet any of the two requirements it is not allowed to give out additional loans or receive utility from dividends. If at any point a bank's capital becomes negative it goes default - all depositors lose their savings, and all debtors don't have to pay back their loans. A defaulted bank begins next period with a reset balance (one can think of a government bail out), but is not allowed to give out dividends/bonuses for a fixed period of time (one can think of the bank being under supervision).

The economy's circular flow is illustrated in 1. A given period consists of the following steps:

- production of the consumption good takes place. Wages and rental rates are paid out
- potential bank defaults are resolved
- banks approve or reject loan applications. Approved projects are initiated
- banks adjust their interest rate and capital-per-dollar cut-off rate
- banks give out dividends to shareholders (receive utility)
- firms receive capital from outstanding projects and draw a new project opportunity
- firms make loan repayments
- firms decide whether to invest in the new project (and whether to apply for a loan)
- firms consume leftover cash
- households decide whether to keep their deposits and deposit their wage, switch banks, or withdraw and consume their savings.
- households receive interest on deposits

The production sector in this model is deliberately simplistic. Both factors are supplied inelastically, and are paid per their marginal costs. This is intentional, since the paper focuses on the financial sector.

For simplicity, each firm's projects are drawn from the same distribution over project characteristics. In that case there are no differences in the productive capacity of each firm. However at any point there will be difference in each firm's collection of active projects and hence the firm's risk profile.

In the current setup banks can only approve or reject loans based on the characteristics of the project. More specifically banks calculate the aforementioned "capital-to-dollar" ratio by first calculating the expected stream of the capital good:

$$\sum_{j=1}^{J} k_j(1-p)^j + K(1-p)^J \qquad [1]$$

and then dividing it by the initial investment amount. Here J is the total duration of the project, k is the mean per-period flow of capital, p is the per-period default probability, and K is the terminal income received in the last period. Note that in this approach the bank cannot screen riskier firms (it can only set the cut-off capital-to-dollar ratio). This can be improved upon in a future implementation.

Each project has different values for all variables in the capital-to-dollar formula, as well as different per-period standard deviations of capital income, and different liquidation values.

Notice also that both firms and households are somewhat constrained in their actions. Firms can choose to finance a project either only with cash, or only with loans. Households on the other hand must always have all their savings in a single bank, and can either deposits all their cash, or consume it (but not consume part and deposit the rest). These limitations are somewhat arbitrary and can also be relaxed in future work. Here however they are introduced mainly to ease implementation.

For simplicity it is assumed that all agents share the same (logarithmic) utility function, and have the same per-period discount factor (equal to 0.99).

Decision Making

Agents in this model make decisions following a reinforcement learning training strategy. The environment is designed in such a way, that whenever an agent has to make a decision, they must choose one of several discrete options:

Households must choose whether to keep their deposit and consume their cash at hand, whether to deposit their cash at the same bank, or whether to withdraw the existing deposit and put it along with the cash in one of the other (countably many) banks; Firms choose whether to forgo a project, whether to invest cash in it, or whether to apply for funding to one of the (countably many) banks. Banks decide whether to increase by a fixed increment, decrease by a fixed increment, or keep constant the interest rates on loans and deposits and the project profitability cut-off.

In the concrete implementation shown in the text there are ten banks, so this amounts to 13 possible mutually-exclusive decisions for each household per period, 12 for each firm, and 27 (3 variables to adjust with 3 degrees of freedom each) possible decisions for each bank.

Each period before making a decision, agents observe the economy's (and their own) state. The economy is fully described only by the state variables of all agents included in it (which in the models presented in the text consists of 500 households and 100 firms on top of the 10 banks), only a portion of the state is observable by each agent. This portion includes the balances of all banks (one can think of them as public companies), the individual agent's own balance (be it firm, household, or bank), and some macroeconomic variables (the level of output, the wage, the rate of return on capital). This 'observation

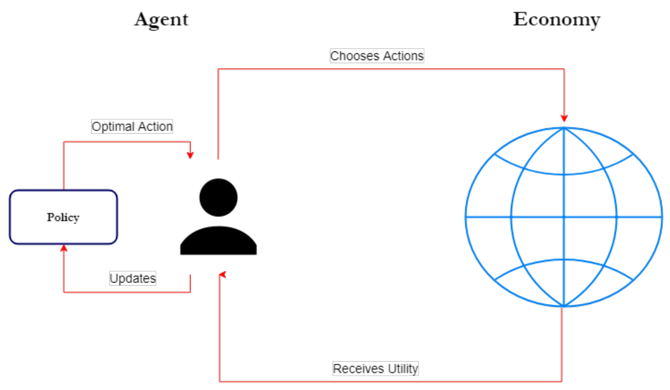

Fig. 2. Agent interacting with the economy

set' numbers around a 100 variables per agent. I will denote this observation vector (which represents the state relevant to the decision maker) by s.

The goal of each agent in the economy is to learn a mapping between states (observation sets) s and actions a (the action set is discrete), which maximizes their infinite sum of discounted future utilities. Call this mapping (policy) π^*.

To this end the agents in this types of model begin in a 'blank' state, where they don't know anything about optimal behavior. The economy than goes through several 'training' iterations which aim to let agents improve their policies as to resemble a good enough approximation of π^*

Q Learning. The fundamental learning algorithm that is to be employed by the agents in this model is known as Q-Learning. The Q in the name comes from the traditional notation for a action-value function in dynamic programming, where $Q^\pi(s,a)$ usually denotes the expected discounted stream of future rewards (i.e. utility), following state s in which action a is taken, and policy π is followed afterwards. Formally:

$$Q^\pi(s,a) = u_0(s,a) + \beta E_{s^t\sim\pi}\left[\sum_{t=1}^{\infty} u_t(s^t,\pi(s^t))\right] = u_0(s,a) + \beta E_{s^t\sim\pi}\left[Q^\pi(s',\pi(s'))\right] \quad [2]$$

Where $\pi(s)$ is the action prescribed by the policy π at state s, u_t is the utility at period t, β is the discount factor, $s^t \sim \pi$ denotes the sequence of future states under the policy, and s' denotes the very next state.

Note the similarity with the better known in Economics concept of a state value $V^\pi(s)$. In fact it holds that:

$$V^\pi(s) = Q^\pi(s,\pi(s)) \quad [3]$$

Q-learning relates to a general algorithm of finding an approximate solution to the dynamic programming program, given by the Bellman equation:

$$Q^*(s,a) = max_a u(s,a) + \beta E_{s,a} V^*(s') \quad [4]$$

Where

$$V^*(s') = max_{a'} Q(s',a') \quad [5]$$

With a' being any action available at s'.

Whereas in dynamic programming the problem can be solved leveraging the knowledge of system dynamics (e.g. by general policy-value iteration), this is unfeasible in our framework (as it is in most real-world applications), since the dynamics of the system are complex and generally unknown by the agent. Thus the agent has to rely on approximate solutions. A comprehensive reference is Sutton and Barto (2018).

Some of the practical aspects of the implementation are influenced by a seminal paper by Mihn, where deep Q networks (a main tool of this paper) were introduced. One of the few papers so far to apply deep Q networks to an economic problem is Leibo (2017) who uses them to model sequential social dilemmas.

Function Approximation. As already mentioned, the agents in the model are trying to approximate Q^*. To that end define the deferentialable function Q_ϕ (parametrized by ϕ) as the agent's current approximation to Q^*. Every time the agent is called to take an action in state s then, they choose $arg\max_a Q_\phi(s,a)$. As the agent interacts with the environment, they learn how to update the function Q_ϕ, so that it approximates Q^* better and better.

To describe the algorithm to do this, imagine that the agent interacts with the environment for N periods, following some policy. Let s_i, a_i for $i = 1,2,..N$ denote the states encountered, and the actions taken. Define the Bellman error (λ) at state s_i, after action a_i as:

$$\lambda_i = Q_\phi(s_i,a_i) - u(s_i,a_i) + \beta V_\phi(s_i',a_i') = Q_\phi(s_i,a_i) - u(s_i,a_i) - \beta \max_a Q_\phi(s_i',a_i') \quad [6]$$

Looking back at the definition of Q_ϕ it is evident, that the Bellman error measures the 'surprise' of the agent - the difference between the expected sum of feature utility before the action is taken, and the expected sum of future utility ex-post (when the immediate utility and the new state s_i' are observed). We want to update the parameters such as to minimize this 'surprise' (a.k.a to make the agent better at forecasting the consequences of each action). A simple algorithm to do this is:

1. Collect data $(s_i,a_i,s_i',u(s_i,a_i))$
2. Set $y_i = u(s_i,a_i) + \beta \max_{a'} Q_\phi(s_i',a_i')$
3. $\phi \leftarrow_\phi \sum_i ||Q_\phi(s_i,a_i) - y_i||^2$

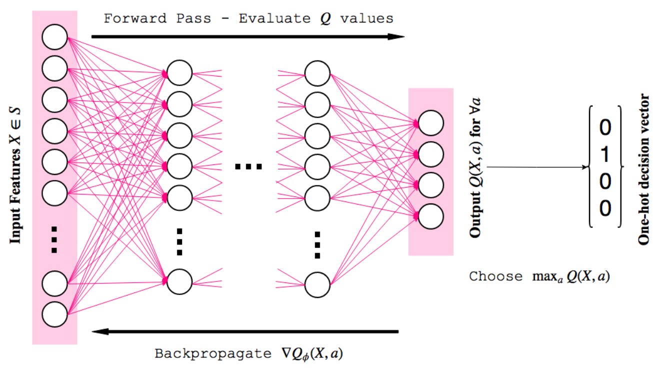
Forward Pass - Evaluate Q values
Input Features $X \in S$
Output $Q(X,a)$ for $\forall a$
0
1
0
0
One-hot decision vector
Choose $\max_a Q(X,a)$
Backpropagate $\nabla Q_\phi(X,a)$

Fig. 3. A representation of a Q-Network

Where step 2. and 3. are repeated K times (think of value function iteration with sampling). This is little more than fitting a regression to y_i as defined above.

The algorithm described above will achieve its goal, however we want to be able to make updates to the approximate Q function (and therefore to the policy function π) online. That is, after each interaction with the environment, so that our agents are adaptable to changes in the economy. We can do this by taking a gradient descent step for Q_ϕ after each period. Letting the learning rate of the gradient descent be denoted as γ the algorithm becomes:

1. Take an action a_i at state S_i and record $S_i, a_i, S_i', u(S_i, a_i)$
2. Calculate the Bellman error $\lambda_i = Q_\phi(S_i, a_i) - u(S_i, a_i) - \beta \max_{a'} Q_\phi(S_i', a_i')$
3. Set $\phi \leftarrow \phi - \gamma \nabla_\phi Q_\phi(s_i, a_i)\lambda_i$

Approximation with neural networks. So far we haven't mentioned anything about the functional form of Q_ϕ. The simple possible approximator is a linear one $Q(s,a) = \phi' s$. The linear approximator is simple and easy to compute. Furthermore we know comparatively more about its convergence properties (15).

However, the flexibility of a linear function is quite limited, and in many practical applications, a slightly more complex functional form is chosen despite less developed convergence theory (16). This paper does the same. I take Q_ϕ to be a simple feed forward neural network of the form:

$$Q(s_i, \mathbf{a_i}) = \sum_{l=1}^{L} \delta(\phi_l' s_l) \qquad [7]$$

Where L is the total number of layers, δ is a non-linear function (e.g. a sigmoid, or a rectified linear unit), ϕ_l are the parameters at layer l, and $s_1 = [1, S_i]^T$. This is illustrated in 3.

To update the weights of the network, we just use backpropagation, starting with $Q_\phi(s_i, a_i)\lambda_i$ at the outermost layer. As illustrated by the figure, we can convert the output of the network directly to actions as per the definitions of the previous section. This amounts to just setting the element in the action vector, corresponding to the highest Q value to 1, and the others to 0.

Double Q Learning. The above section describes the vanilla version of Deep Q-Learning. However, there are some known problems with this algorithm (17). Namely, the samples of state-utility sequences that the agent encounters while interacting with the environment are highly correlated (since they are taken from the almost the same policy). Furthermore, note that we calculate y_i in the algorithm above **before** taking the gradient of the Q network - in essence we are optimizing to match a moving target. Both these problems hinder the performance of deep Q-networks.

To fix this well known issues I introduce two ideas, first encountered in (17): the replay buffer, and the target network. The replay buffer is nothing more than a collection of past interactions (think of it as memory). When the network is updated we use not only the last action, but a random sample from the replay buffer - that way the Bellman errors will not be (as) correlated. We call the replay buffer $\mathcal{B}$

The target network fixes the second issue, by creating two networks - one for choosing actions (Q_ϕ), and another for evaluating them ($Q_{\phi'}$). This way when we update the parameters of the network that does the choosing (Q_ϕ) , the target is actually fixed (i.e. the network that evaluates them $Q_{\phi'}$). After a certain number of periods the current choosing network becomes the new target.

The algorithm described above is known as "Double Q Learning". For more details on it the reader is referred to (17). To recap the algorithm takes the form:

1. Take action a_i, record $(S_i, a_i, S_i', u(S_i, a_i))$. Add it to $\mathcal{B}$
2. Sample mini-batch $\{S_j, a_j, S_j', u(S_j, a_j)\}_{j=1}^N$ from $\mathcal{B}$
3. Compute the Bellman errors $\lambda_j = Q_\phi(S_j, a_j) - u(S_j, a_j) - \beta Q_{\phi'}(S_j', arg\max_{a_j'} Q_\phi(S_j', a_j'))$
4. Update $\phi \leftarrow \phi - \gamma \sum_j \nabla_\phi Q_\phi(S_j, a_j)\lambda_j$
5. Every T steps set $\phi' \leftarrow \phi$

This final learning mechanism is the one implemented for all agents in the model economy - households, firms, and banks.

In principle the 'purest' agent-based model will have one neural network per each simulated agent. In this model however there is one network per **agent type**. This means there is a single neural network that does the Q function approximation for every household (and another one for banks and another one for firms), but the inputs to the function are different, depending on the current state of the individual simulated agent.

The reason for this choice is mainly ease of computation [*]. However there is also a nice conceptual justification. Q learning does not distinguish between off-policy and on-policy training (14). This means that meaning that agents that employ this algorithm learn equally well from their own experience and from the experience of others. And since in this model we do not assume that there is private information, all agents of a single type can observe the actions and outcomes of all other agents of that type.

All this goes to say is that if I had implemented a version of the model where each single agent has their own independently trained Q-network, I would have to allow all agents of the same type access to the same replay buffer (since everyone also observe others' actions). If we have hundreds of neural networks training from the same data, then the only differences in weights will be due to mere chance in sampling the mini batches and should disappear asymptotically. Thus having a single network per agent type emulates the asymptotic behavior of having one network per individual agent [†].

Implementation. The model as described above was implemented from scratch, using both the R programming language and Python. The implementation uses an object-oriented paradigm, with the economy being the single most important class, encompassing the objects for the different agent types.

PyTorch was used for implementing the function approximators. Each neural net consists of an input layer, two hidden layers, one sigmoid activation layer (after the first hidden layer), one rectified linear unit layer (after the second hidden layer), and an output layer. The input layer and the two hidden layer's outputs were batch-normalized. Standard parameter values were used for the learning rate (0.001) and weight decay (0.2). A batch size of 512 was chosen and a replay buffer size of 2000. All these values as well as the network topology can in principle be subject to a more sophisticated parameter tuning process.

The (research quality) code is made available in the project's repository.

Emerging Dynamics

In order to implement the economic environment described above I created a simulated economy, consisting of 10 banks, 100 firms, and 500 households. The exact starting parameters chosen are available in the appendix. The agents in the economy were then left to interact with the environment (and each other) for around 20,000 periods. The initial policies of each agent type were random. However as the simulation proceeded they were progressively improved using the Q-Learning algorithm described above.

[*] the model was trained on a personal laptop with no GPU

[†] This is true for Q-learning but not in general.

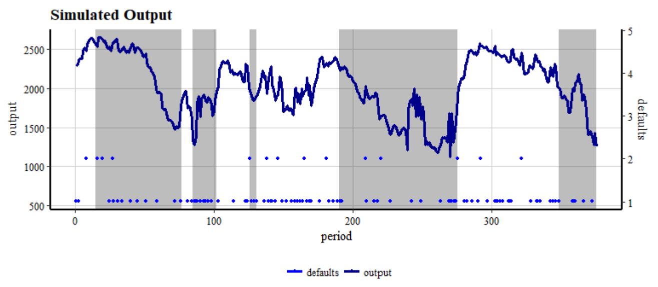

Fig. 4. Simulated Output

While in theory agents are optimizing an infinite sum of discounted returns, in practice the training was set up so, that in any given period there was a 1 : 1000 chance of the economy resetting to its initial state. In this section I detail the results of the simulated periods following the last such economy reset. This 'iteration' consisted of 570 periods, the first 100 of which were discarded to reduce the dependence on initial conditions.

Output. Figure 4 shows the resulting dynamics of total economic output. Few things are immediately noticeable. First, output seems to oscillate around a certain mean level around 2000 (2043 to be exact), sometimes going up or down, but never drifting away completely. This is desirable, since in the absence of technological change, we wouldn't expect to see any long-term trends in output. (a regression of output against time actually gives a significant negative estimate, but it's magnitude is less than 0.04 percent of the mean and becomes insignificant if we account for capital and bank variables).

The second noticeable feature is the graph's cyclicallity. Output is clearly non-stationary (confirmed with a Ljung-Box test), and has a high-degree of persistence (see the auto-correlation coefficient in 1).

More importantly, the economy exhibits expansionary and contractionary periods. On top of that, similar to the real world, cycles seem to be quite irregular. The shaded areas on the graph depict periods of recession of the simulated economy (recession starts are defined as 4 consecutive periods of output decline, and recession ends are defined as 4 consecutive periods of growth). The shortest recorded recession lasts only 5 periods, while the longest is more than 80. Compare this with NBER data on American recessions and we can see similar discrepancies in length. Quite unlike real economies however, the simulated environment seems to be characterized with very rapid expansions and gradual declines (the opposite is true in reality (18)).

Nevertheless, the emergence of endogenous business cycles in the model is an interesting fact by itself. It seems that the business cycle is driven mainly through credit availability (see 5).

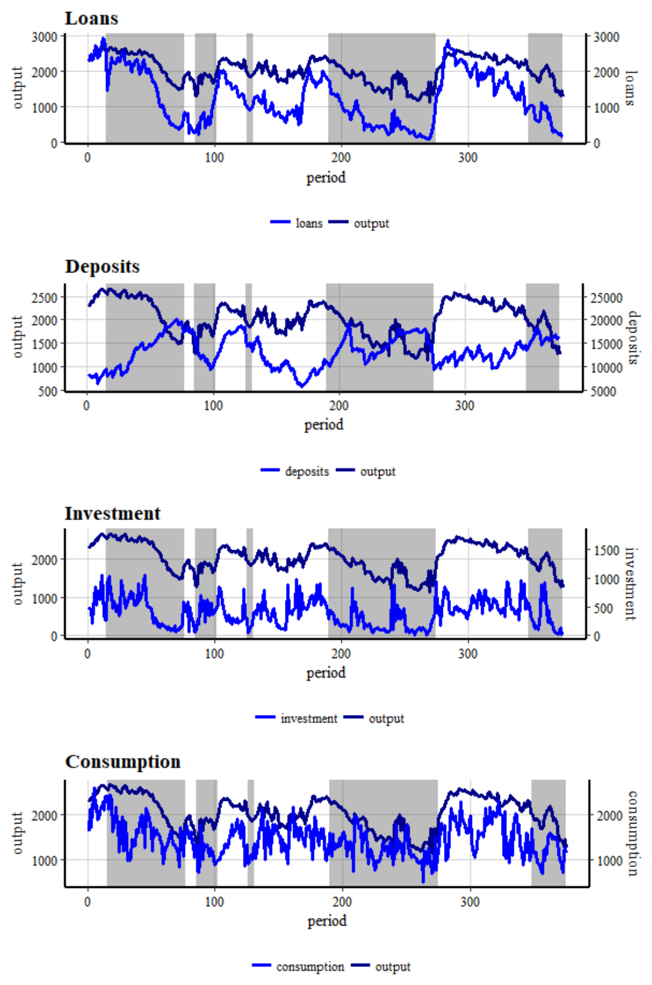

Fig. 5. Co-movements

Finally, 4 depicts the number of bank failures per period (the blue dots). We can see that in this economy bank failures are a common fact of life, occurring almost all of the time. In addition it does not seem that bank failures are more or less frequent during recessions than during expansions.

Co-movements. If we move to examine the other time series generated by the model, we would see another key feature of the simulated environment - strong co-movements of economic variables. This is exhibited in 5

We notice for example that loans tend to lead output - a jump in loans almost always succeeded by a jump in output. A similar feature is also exhibited by the plot of output against investment. Deposits, on the other hand, even though they exhibit a strong cyclicality of their own, tend to move in a pattern not obviously related to the business cycle.

A feature worth mentioning is the over-abundance of savings in the simulated economy. Deposits are almost an order of magnitude more than output, compared to .5 to .6 in real developed economies. This potentially points out the scarcity of good investment opportunities in the simulation.

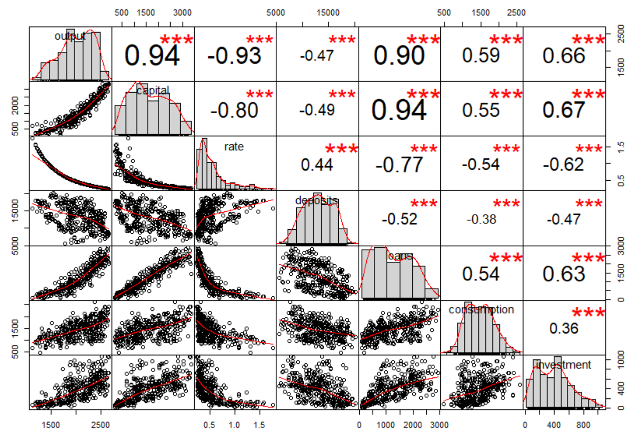

Fig. 6. Correlations

Table 1. Variation of simulated time series

Series	Coefficient of Variation	Auto-correlation (1 lag)	Contemporaneous correlation with output
output	0.186	0.948	1
capital	0.539	0.963	0.940
rate	0.559	0.880	-0.933
deposit interest	0.735	0.990	0.345
loan interest	0.377	0.990	-0.483
approval rate	0.018	0.993	0.317
deposits	0.255	0.966	-0.470
loans	0.589	0.962	0.903
consumption	0.269	0.658	0.589
investment	0.608	0.658	0.662

Note also that consumption, even though somewhat positively related to economic activity, does not follow output as closely as other variables. This is somewhat in agreement with real economic data, which shows weaker correlation between the two.

Capital also shows positive relationship to output and to loans. The rate of return on capital is negatively related to output (through the relationship between the rate of return and the amount of capital), while the wage follows output 1 : 1 (fixed inelastic labor supply and fixed labor share of income).

6 further spells out the relationships between different variables by giving correlation coefficients and scatter plots of various time series pairs.

Variance. We can also examine the variance of different time series and compare it to what we know about real world economic indicators. The first column of 1 shows the coefficient of variation (standard deviation over mean) of each of the variables derived from the simulation.

We see that investment is about 3 times as volatile as

Table 2. Correlations with lags and leads of output (y)

Series	y_{t-4}	y_{t-3}	y_{t-2}	y_{t-1}	y_t	y_{t+1}	y_{t+2}	y_{t+3}	y_{t+4}
output	0.845	0.874	0.919	0.948	1.	0.948	0.919	0.874	0.845
capital	0.806	0.844	0.872	0.909	0.940	0.972	0.933	0.907	0.871
rate	-0.770	-0.798	-0.846	-0.866	-0.933	-0.859	-0.832	-0.777	-0.745
deposit interst	0.221	0.251	0.282	0.312	0.345	0.368	0.392	0.414	0.434
loan interest	-0.472	-0.473	-0.476	-0.476	-0.483	-0.480	-0.481	-0.479	-0.474
approval rate	0.273	0.284	0.298	0.308	0.317	0.320	0.320	0.318	0.314
deposits	-0.346	-0.372	-0.406	-0.436	-0.470	-0.490	-0.506	-0.520	-0.534
loans	0.769	0.804	0.843	0.869	0.903	0.914	0.926	0.909	0.892
consumption	0.470	0.512	0.551	0.583	0.589	0.539	0.528	0.509	0.490
investment	0.435	0.461	0.529	0.570	0.662	0.679	0.680	0.632	0.619

consumption - a feature also exhibited by real economies (19). On the other hand however it also seems that consumption, albeit much less volatile than investment (which agrees with the data), has 1.5 times higher coefficient of variation than output (which doesn't).

Of the other variables, the wage and the rental rate have (expectedly) very similar volatilities. Loans tend to be much more volatile than deposits. The interest rate on deposits however tends to vary much more than the one on loans. Finally, the approval cut-off looks like the most stable variable of them all (it stays around its initial value of 1 throughout the simulation).

Serial Correlation. As mentioned above, output exhibits a strong serial correlation. In fact the same is true for all derived variables. In addition most of them are correlated with output at various lags and leads. These correlations are presented in 2.

Several observations can be made. First, investment tends to lead output (having higher correlation with future output than past output). The same goes for loans, deposits, capital, and the interest rate variables. The rate of return and consumption on the other hand seem to have higher correlations with past output than future output, however they mostly tend to go in sync with the cycle.

Running Economic Experiments

Probably the main advantage of an economic model based on reinforcement learning is its flexibility. The modeler needs to just tweak the parameters of interest a little bit and then let the agents learn for themselves the optimal policy in the new environment.

In order to demonstrate the usefulness of the model I conduct two "experiments" using the simulated economy. Both of them use as a baseline the economy described in the previous section, but change one of the rules of interaction, keeping everything else fixed. I then examine the differences in the resulting behavior.

Introducing deposit guarantees. In this experiment every aspect of the economy is identical to the benchmark, except now deposits of households are 'guaranteed'. This

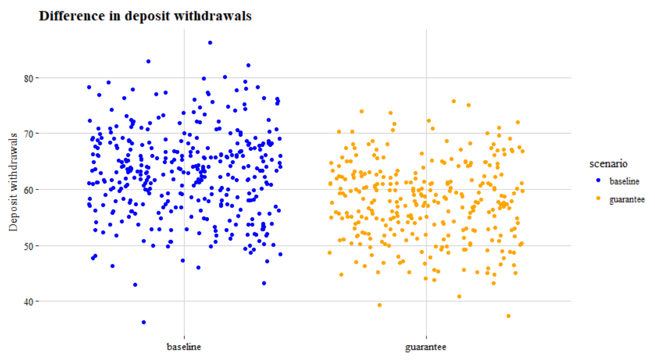

Fig. 7. Number of deposit withdrawals

means that whenever a bank defaults, instead of getting all their deposits wiped out, households receive them back as cash.

In order to make up for that, in the next turn the wage is decreased equally (for both those households who had deposits in the defaulted bank and those who did not) by an amount, such that the total amount equals the sum of paid back deposits (think of a government providing the guarantee, but then raising taxes to pay back for it).

Other than that, the banks can still default, losing their ability to pay back dividends for the next few periods and resetting their balances. Firms are also unaffected by the change, as a bank default still means that they don't have to repay any loans to that bank.

The simulation was ran for $20,000$ periods, for which agents learned their policies from scratch. All parameters were kept the same. The resulting simulated economy shares a lot of features with the baseline case - cyclical behavior, high autocorrelations, and the overall variance structure. However some key differences emerged.

7 shows one of those differences. Depicted are the number of withdrawals for each period of the final 300 periods or so (after restarting) of both simulations. It is evident that with the deposit guarantee, households reduce their propensity to withdraw their savings.

The significance of the difference between the two simulations was confirmed by a t test. A reasonable guess is that in the new learned policy the threat of default plays no role and hence the need to withdraw funds is

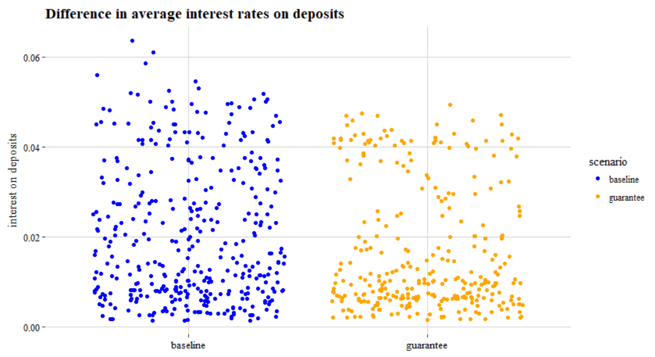

Fig. 8. Correlations

only driven by consumption needs.

As a result of the lower tendency to withdraw, banks can then afford to lower the interest on deposits (8), since they no longer need to pay a risk premium. This in hand means that they can also lower interest on loans further when competing in the loans market.

The end result is that in the economy with the deposit guarantee lending, and hence investment, increases. This in turn leads to higher stocks of capital, which boosts production and incomes. In the end the average consumption is also higher than in the benchmark (all differences in mean aggregates are significant at the 5% level.

Changing the required reserve ratio. The next experiment involves only changing the required reserves-to-deposits ratio of each bank. Five different economies were created, with reserve ratios of 0.04, 0.08, 0.12, 0.16 and 0.20 respectively.

The networks of the economy with a 4% reserve ratio were initialized by transferring the weights of the benchmark environment (where the required reserve ratio was 0.10). The agents were then trained for an additional 4098 periods. The weights of this model were transferred to the economy with an 8% reserve ratio, where training resumed for another 2048 periods (the agents should need less training to adjust to a smaller parameter difference). This training transfer was carried on until all five scenarios were covered.

Table 3 shows the resulting means and standard deviations from the five the last 300 or so periods of each simulation.

Several key differences are immediately obvious. First, it seems that average per-period output remains relatively stable around 2500 and if there is any effect of the required reserve ratio it is small. However the variance of output in the 4% required reserve scenario is much higher.

The average per-period sum of outstanding loans seem to be decreasing with the reserve ratio, while the average per-period investments seem to go up. Both of these effects however are small.

The most striking difference is the level of average per-period deposits in the banking system (figure 8). While there isn't much difference in the 8% to 12% range, the average deposits levels in the scenario with the lowest minimal reserve requirements are much lower.

Table 3. Means and standard deviations of various aggregates under different reserve requirements

Series	4%	8%	12%	16%	20%
output	2533	2532	2585	2546	2564
	(143)	*(89)*	*(60)*	*(76)*	*(66)*
deposits	3326	14291	13608	8098	13348
	(2457)	*(1723)*	*(1474)*	*(1939)*	*(1472)*
loans	2370	2022	2110	2161	2148
	(469)	*(285)*	*(228)*	*(266)*	*(239)*
consump-tion	2038	1552	1672	1907	1645
	(759)	*(287)*	*(361)*	*(810)*	*(318)*
investment	610	688	684	685	707
	(163)	*(116)*	*(99)*	*(123)*	*(116)*
deposit interest	0.037	0.028	0.031	0.029	0.029
	(0.010)	*(0.004)*	*(0.002)*	*(0.004)*	*(0.006)*
loan inter-est	0.031	0.041	0.040	0.040	0.041
	(0.013)	*(0.004)*	*(0.003)*	*(0.005)*	*(0.005)*

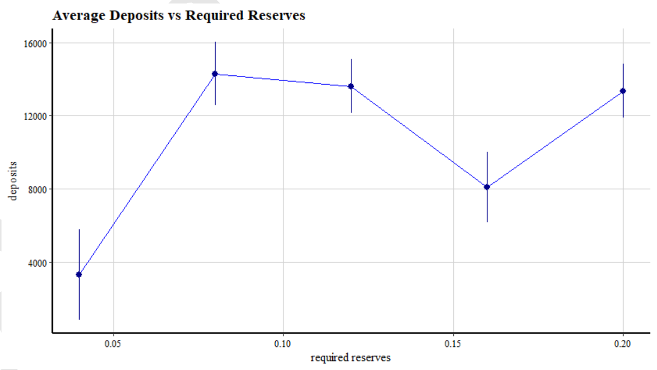

Fig. 9. Average deposits against required reserves

We can also notice that the interest rates on both deposits and loans are higher in the 4% scenario (and are almost identical for the scenarios with stricter reserve requirements). Paradoxically the average interest on loans is below the average interest on deposits. Finally, consumption in the 4% scenario is also higher on average, but also much more volatile. The difference with any other scenario however is not significant.

A possible explanation for these observations is that there is a cut-off reserve requirement ratio somewhere between 4 and 8 percent at which households decide at large scale that they'd rather consume then save in deposits. Even though in the final simulation periods, which are presented here we do not observe more bank defaults at the lower reserve level, it is possible that during some of the training iterations there was increased incidence of bank default, which made households learn a policy where they deposit less.

The decrease in households' willingness to deposit would also explain the higher variability in consumption - with less savings the ability to smooth out consumption is diminished. It may also explain the higher interest rate, as banks compete for scarcer savings.

Finally the relative stability of output across the five scenarios may be due to the fact that, as mentioned earlier, the simulated economy with this parameter settings is characterized with excessive deposit levels. Thus the bottleneck in production is the availability of projects in which the firms are willing to invest, and a reduction in deposits does not have adverse effects.

Future Research

While implementing this model I have encountered several minor and major difficulties, some of which have been hinted at above.

Policy Convergence. In this paper agents interact with the environment and with each other. This creates several problems. First, the environment becomes dynamic, as a the optimal policy of each agent changes with the current policies of all other agents. This means that convergences will be slower and likely non-monotone.

Second, since the three agent types have different population sizes, their replay buffers' size differ. Namely, because there are only 10 banks they accumulate experience to learn from much slower than the 500 households. The same holds for the training accuracy of the Q function approximator.

As a result it becomes hard to know how long to train the agents and to evaluate whether a certain network topology preforms better than another (20). In general the concept of 'good' performance of the reinforcement learning algorithm becomes blurry in a multi-agent setting.

In order for RL methods to be successfully implemented for economic modelling we need to develop clear objective measures of agent performance in any given environment.

Choosing hyper-parameters. The output of the model created in this paper is quite sensitive to the values of the chosen hyper parameters. Changing productivity for example may lead to much higher or much lower landing overall, and to much less stable output.

Figuring out how each parameter affects the conclusions is not a straight-forward task and careful techniques that balance plausibility with sensible results must be devised.

A further open question is how to take the outputs of models such as these to the data. As mentioned above some of the features of this model do not comfort with real world facts. Adjusting hyper parameters may change that. Criteria for deciding whether a simulated economy behaves like the data should be put forth, and methods to calibrate the model's parameters in order to meet these criteria while avoiding overfitting should be invented [‡]

Continuous actions spaces. The reader would have noticed that the action space of all agent types in the model is discrete. This was intentional, since the DQN algorithm works with discrete actions. DQN is relatively easy to implement, and yet has a solid track record in achieving good performance in complex tasks (13), therefore it was deemed most appropriate for the current paper.

However, many economic decisions are better suited to be modelled as continuous. For example, it is more natural for the banks to directly choose the interest rate, rather then only move it up or down by a fixed increment.

There are plenty of reinforcement learning algorithms suited for continuous actions spaces (see for example (21)). Their application to economic models can be explored in the future.

Concurrent simulation. Another defining feature of the model was that time was divided into discrete chunks called periods. Furthermore within each period actors acted in 'turns' - first the banks, then the firms, then the households.

This is unrealistic and in principle unneeded. The model can be implemented concurrently, with each agent acting and interacting with the environment 'when ready' and without waiting for the others. This of course requires considerable development effort.

Repeated simulation. A major weakness of the analysis of the simulated economy presented here, is that for all scenarios examined I have only used single simulations of around 300 periods. This does not guarantee that any of the results is not due to mere chance.

In reality a much better approach would be to produce numerous simulated histories (restarting the economy after each) and draw conclusions based on the aggregation of the simulations. This however was not done here due to the time requirements of optimizing the code and due to limitations on computing resources.

Policy interpretation. In this model agents learn a policy in a state space with around a hundred dimensions. The ability to deal with this high dimensionality is the main advantage of reinforcement learning, but it also presents the main challenge in drawing conclusions from the model.

It is sometimes hard to learn what the agents learned in a given scenario and how it differs from their policy in another one. Answering this questions is important for understanding the driving forces of the model.

Since the policy in this case amounts to choosing the highest predicted Q value, and since Q values are predicted using a deep neural network, the interpretation problem comes down to deciphering the inner workings of this neural network. This is not an easy task (22), but is an

[‡] DSGE-style calibration is one option to do that

active area of research and some promising methods based on local interpretability may be useful (23).

Causal Links. Finally, possibly the biggest difficulty with the model (which is shared with other ABMs) is discovering the cause and effect link between different components of the system.

While the simulations clearly show that the economy exhibits cyclical fluctuations, that deposit guarantees lead to lower interest rates, and that increasing required minimal reserves leads to ..., explaining *why* these phenomenons occur is not a straightforward task. I have tried to provide narrative explanations whenever possible, but scientific usefulness would require a much more rigorous approach.

Conclusion

This paper has presented a model of the financial sector of the economy in which heterogeneous agents learn how to maximize their utility in a bounded rational way using deep reinforcement learning. This approach has allowed for a high degree of flexibility on the side of the modeller in specifying the interactions and aggregate processes in the economy, with the effective state space dimensionality for each agent being in the hundreds [§].

The key feature of the resulting dynamics of macroeconomic aggregates is the presence of endogenous business cycles. The cycles are driven by expansions and contractions in the firm loan market. Each credit crunch leads to less investment from firms and less outputs. Subsequent expansions of credit on the other hand tend to lead expansions in production.

By having this model in our hands, we are free to run any experiment that we can think of. Agents will then use their learning capabilities to autonomously find an approximate optimal policy in the new environment. Two such experiments were shown in the main text, however the number of possible questions that can be answered with such a model is huge.

As far as the baseline model is considered an accurate enough representation of reality, the results of such experiments can then be used to inform monetary or fiscal policy by the government, or business decisions by the financial industry. While I make no claim that this particular model can serve this purpose, I strongly believe that it is possible to build a realistic enough model by employing the tools presented here.

In order to do so however several major challenges need to be solved. Some of these challenges lay in the domain of multi-agent reinforcement learning - e.g. how to deal with multiple agents optimizing at the same time, or how to interpret learned policies. Others lay in the field of agent-based economics - how to design rich simulation environments that also allow us to easily draw causal conclusions, or how to make the model interactions more like real world ones while balancing computational complexity.

Despite these challenges however, I hope that this paper will serve to show that agent based models based on reinforcement learning hold tremendous potential for Economics. With enough focus and dedication they may as well end up shaping the future of the field.

[§] there is no reason to exclude even higher dimensionalities

ACKNOWLEDGMENTS. I am grateful to the Bulgarian National Bank (and in particular the Student Scholarship Committee) for giving me the opportunity pursue the ideas presented in this paper. I am also thankful to Svilen Pachedzhiev for his guidance and advice.

1. Kydland F, Prescott E (1982) Time to build and aggregate fluctuations. *Econometrica* 50(6):1345–70.
2. Stiglitz JE (2017) Where modern macroeconomics went wrong, (National Bureau of Economic Research), Working Paper 23795.
3. Romer P (2016) The trouble with macroeconomics, (Stern School of Business New York University), Working paper.
4. Ackerman F (2002) Still dead after all these years: Interpreting the failure of general equilibrium theory. 9:119–139.
5. Daskalakis C, Goldberg PW, Papadimitriou CH (2009) On the complexity of computing nash equilibrium. *SIAM Journal of Computing* 39:195–259.
6. Epstein JM, Axtell R (1996) *Growing Artificial Societies: Social Science from the Bottom Up.* (The MIT Press) Vol. 1, 1 edition.
7. Richiardi M (2004) The promises and perils of agent-based computational economics, (University Library of Munich, Germany), Computational economics.
8. Buchanan MB (2009) Meltdown modelling. could agent-based computer models prevent another financial crisis? 460(7256):680–68.
9. Chan-Lau J (2017) Abba: An agent-based model of the banking system. 17:1.
10. Ismail OR (2012) An agentbased computational model for bank formation and interbank networks.
11. Grilli R, Tedeschi G, Gallegati M (2014) Bank interlinkages and macroeconomic stability. *International Review of Economics Finance* 34(C):72–88.
12. Fudenberg D, Levine DK (2016) Whither game theory? towards a theory of learning in games. *Journal of Economic Perspectives* 30(4):151–70.
13. Mnih V, et al. (2013) Playing atari with deep reinforcement learning.
14. Sutton RS, Barto AG (2018) *Reinforcement learning - an introduction, 2nd Edition.* (Final Draft).
15. Melo FS, Ribeiro MI (2007) Q-learning with linear function approximation in *Learning Theory*, eds. Bshouty NH, Gentile C. (Springer Berlin Heidelberg, Berlin, Heidelberg), pp. 308–322.
16. Thrun S, Schwartz A (1993) Issues in using function approximation for reinforcement learning in *Proceedings of the 1993 Connectionist Models Summer School*, eds. M. Mozer, P. Smolensky DTJE, Weigend A. (Erlbaum Associates).
17. van Hasselt H, Guez A, Silver D (2015) Deep reinforcement learning with double q-learning. *CoRR* abs/1509.06461.
18. Acemoglu D, Scott A (1997) Asymmetric business cycles: Theory and time-series evidence. *Journal of Monetary Economics* 40(3):501 – 533.
19. Stock JH, Watson MW (1998) Business cycle fluctuations in u.s. macroeconomic time series, (National Bureau of Economic Research), Working Paper 6528.
20. Zawadzki E, Lipson A, Leyton-Brown K (2014) Empirically evaluating multiagent learning algorithms. *CoRR* abs/1401.8074.
21. Sutton RS, McAllester D, Singh S, Mansour Y (1999) Policy gradient methods for reinforcement learning with function approximation in *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99. (MIT Press, Cambridge, MA, USA), pp. 1057–1063.
22. Olden JD, Jackson DA (2002) Illuminating the black box: a randomization approach for understanding variable contributions in artificial neural networks. *Ecological Modelling* 154(1):135 – 150.
23. Ribeiro MT, Singh S, Guestrin C (2016) "why should I trust you?": Explaining the predictions of any classifier in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016.* pp. 1135–1144.