Reinforcement learning for analyzing climate policies

Laurens Samson (10448004) **Rik Helwegen** (10516034) **Petra Ormel** (10607005)[1]

Abstract

In this research, a new approach is proposed for finding optimal policies in economic climate models (DICE). Instead of the computationally heavy and restricted dynamic programming (DP) approach, deep deterministic policy gradient (DDPG) is introduced for optimizing the policy. We show that DDPG is able to provide policies which capture up to 99.5% of the maximal reward one could gain by adjusting policies in the DICE model.

1. Introduction

In recent years, various studies have revealed the consequences of climate change caused by human activity (Hodson, 2017). Global warming can cause several elements in the climate system to tip into a different state, with irreversible (economic) damages as a consequence (Cai et al., 2016). The Dynamic Integrated Climate Economy (DICE) framework models how the economy, climate development, and climate related policies interact. This allows for calculating a social cost of carbon (SCC). A term representing the current economic cost of carbon-dioxide emission per ton, quantifying the economic damage this emission will cause in the future. With the aid of this model, an optimal policy over time can be estimated on abatement, i.e., how much effort must be put into controlling carbon emissions, in order to maximize the expected value of global social welfare.
One recent study on this topic is conducted by Cai et al. (2016), making an attempt to find an optimal policy for controlling CO_2 emissions with the use of the Dynamic Stochastic Integration of Climate and Economy (DSICE) framework. Their work included 5 essential climate tipping points. Their baseline deterministic model was based on the 2013 version of DICE (Nordhaus, 2014). To address the multidimensional problem, they made use of Dynamic Programming. This approach has several limitations of which the most important one is the drastic increase of computation time in higher dimensions. Complexity of the model, and the difficulty of finding a policy, rises when including more dynamics to the system. It depends on various factors of the climate and economy system that highly interact with each other and of which some contain continuous action spaces. Moreover, incorporating climate tipping points, together with the uncertainties of when they will occur and the effect they will provoke, adds to the complexity of these systems.

Meanwhile, a rapid development has been going on in artificial intelligence due to enhancement of deep learning approaches in machine learning (Krizhevsky et al., 2012). The deep learning revolution started with image recognition for computer vision, subsequently it has spread out over different disciplines in artificial intelligence, such as reinforcement learning (Mnih et al., 2013). State-of-the-art results were accomplished utilizing the deep reinforcement learning approach, the learned policies were able to outperform expert human players in some of the Atari games. Originally, deep reinforcement learning was only capable dealing with discrete action spaces, such as going left, up right and down in Atari games. However, nowadays also techniques have been invented which are able to predict continuous action spaces, such as deep deterministic policy gradient (DDPG) methods (Lillicrap et al., 2015).

This research proposes a new way of solving optimal policies for climate economic models. In special, the use of the DDPG algorithm is investigated for optimizing abatement policies in a 2013 based DICE model. The performance is compared to a baseline policy which is known to be optimal. Therefore, the goal is to develop a fast, reliable and stable algorithm with performance acceptably close to the benchmark. In section 2, the background for both the DICE and the DDPG model is discussed. Section 3 elaborates on the experimental set-up and technical details. The results are presented and analyzed in section 4. After which section 5 concludes on the findings. Finally, section 6 provides discussion on the approach and results.

[1]Supervisor: Dr. H. van Hoof.
Laurens Samson <laurens.samson@student.uva.nl>
Rik Helwegen <rik.helwegen@student.uva.nl>
Petra Ormel <petra.ormel@student.uva.nl>.

2. Background

The background section provides explanation on the main theoretical components this work is based on.

2.1. DICE model

The DICE model is utilized to analyze future scenarios of interaction between climate and economy. Due to forecasting uncertainty and computational cost, the model is prone to restrictive constraints. One such aspect in the DICE model is expressing the global economy as a single simple system. The 2013 version of DICE comprises six state variables to estimate the SCC. One variable is capital stock (K), reflecting the state of the global economy. Three variables represent different reservoirs in the carbon cycle: the atmosphere (C_{AT}), the upper oceans and biosphere (C_{UP}), and the lower oceans (C_{LO}). Last, two variables model the global temperature on surface level (T_{AT}) and at sea level (T_{LO}). The state variables interact as a dynamic system over time. It is incorporated that slow transition times mean that damages tend to be discounted away. This implies that smaller damages which will be felt within a relatively short time period have a larger impact on the SCC today than bigger damages that will take centuries before being noticed. These variables and their relations can be used to calculate the optimal climate policy, which tells what the abatement rate should be given each state in order to achieve maximum global social welfare. More information about the DICE model can be found in the article of Nordhaus (2014). The implementation can be found on the github page: *https://github.com/petraormel/Reinforcement_Learning_for_Climate_Policies*

In order to test the results, a benchmark policy is used. This benchmark is a policy on the same DICE model, optimized with Sequential Least SQuares Programming. Note that this technique is possible because the DICE model is deterministic, a restriction which falls away when solving policies with reinforcement learning.

2.2. Algorithm

In this research, deep deterministic policy gradient (DDPG) is utilized to optimize the climate policy that we approach as a reinforcement learning problem. Policy gradient techniques are widely used in reinforcement learning problems with high dimensional observation spaces and continuous action spaces (Silver et al., 2014). Learning policies with continuous action spaces is impossible for algorithms such as the Deep Q Network (DQN), as they require an iterative optimization process at every step and therefore result in endless exploration (Lillicrap et al., 2015).

2.3. DDPG

A DDPG algorithm is an extended version of a policy gradient. The idea of a Policy Gradient is to represent the policy by a probability distribution that given a state, stochastically selects an action according to some parameter vector. Subsequently, samples are drawn from this stochastic policy and gradients are adjusted in the direction of the greatest cumulative reward. A deterministic policy gradient tries to invent an optimal deterministic policy, where a specific action is chosen from a state. In order to do so, the algorithm still needs to explore different actions that not immediately follow from the deterministic policy. Therefore, DDPG uses an off-policy learning algorithm, which means that an exploitative action is chosen with added noise. The idea is to choose actions according to the off-policy, but to learn about a deterministic target policy. This can be done with the use of an off-policy actor-critic algorithm that uses a differentiable function approximator to estimate the unknown action-value function. The action-value function describes the expected reward after taking an action in a state and thereafter following the policy. This action-value function is necessary such that the algorithm can update the policy parameter by following the approximate action-value gradient (Silver et al., 2014). A deep deterministic policy gradient estimates the action-value function with a neural function approximator (Lillicrap et al., 2015).

Generally, it was believed that learning value functions using neural function approximators was difficult and unstable (Lillicrap et al., 2015). In this work, the approaches by Lillicrap et al. (2015) are implemented in order to resolve this instability for a large part. A first addition is to sample the observations from a replay buffer, this overcomes the problem of highly correlated data. A second enhancement is to work with a source and target network. The target network is (slowly) updated in the direction of the source network and is used to predict the next Q-value. Due to this slower training target network, the training of the source network becomes more stable. (Lillicrap et al., 2015).

In order to make the learning more efficient, noise is generated by an Ornstein-Uhlenbeck process to obtain correlated exploration (Uhlenbeck & Ornstein, 1930). Moreover, *batch normalization* is applied. Different components in the state observation have different physical units and scales. This makes it difficult for the network to learn effectively and it is hard to find hyperparameters that generalize for different scales of state values. This problem is solved by batch normalization layers in the actor and critic networks (Ioffe & Szegedy, 2015).

2.4. Actor-Critic model

As the name suggests, the actor-critic model consists of two components. The actor is a neural network which determines the action from a given state and adjusts the parameters of the policy towards the maximum return. Since the unknown action-value function is necessary for optimizing the actor, an estimation of the action-value function is used instead, calculated by the critic. For DDPG this is also a neural network, using temporal difference learning as policy evaluation algorithm (Silver et al., 2014). More background information can be found in the papers of Silver et al. (2014) and Lillicrap et al. (2015).

2.5. Discounting factor

One important aspect of both the DICE and the DDPG model is discounting over time. In the DICE models, this factor is called the *rate of social time preference*. It can be interpreted as the factor by which the utility declines when the same reward is obtained one year later in time. Common practice in economic modelling is to put this factor equal to the rate of inflation. In the used DICE model this factor (δ) is set to 1.5%.

On the other hand, Reinforcement Learning algorithms include the parameter γ to express the discounting over time. In special, when looking at the Bellman equations which are used for training, the γ is incorporated as follows:

$$q^*(s,a) = r(s,a,s') + \gamma q(s',a') \tag{1}$$

Such that inherently when training, a discounting for time is already included. Therefore, in order to link the systems, the discounting needs to be taken out of the DICE model when training, and put back in when models are evaluated. Formally, the overall goal is maximizing: $\sum_{t=0}^{599}(1+\delta)^{-t}r_t$, such that this is what needs to be evaluated over. Using equation (1) we obtain:

$$\begin{aligned} q(s_0,a_0) &= r_0 + (1+\gamma)^{-1}q(s_1,a_1) \\ &= r_0 + (1+\gamma)^{-1}(r_1 + (1+\gamma)^{-1}q(s_2,a_2)) \\ &= (1+\gamma)^0 r_0 + (1+\gamma)^{-1}r_1 + (1+\gamma)^{-2}q(s_3,a_3) \\ &\dots \\ &= \sum_{t=0}^{599}(1+\delta)^{-t}r_t \end{aligned}$$

Which shows that the desired utility function is optimized when γ is set to δ and the non-discounted rewards are returned by the DICE model when training.

3. Experiments

3.1. Setup

In the model, states are represented as a vector of the seven variables described in section 2.1. The action space, representing the abatement rate, is a value between zero and one. The reward function is defined by the DICE model and shown in the following equation:

$$u(C_t,L_t) = \frac{\frac{C_t}{L_T}^{1-\frac{1}{\Psi}}}{1-\frac{1}{\Psi}}L_t \tag{2}$$

C_t represents the consumption, Ψ is the inter-temporal elasticity of substitution and L_t is the population size (Cai et al., 2015). More information about the reward function can be found in the supplementary information in the paper of Nordhaus, W. (2014). The rewards need normalization in order to prevent exploding gradients during training, therefore the mean and standard deviation were obtained for standard-score normalization using the Dynamic Programming policy.

A python adaptation of the DICE environment that is used is available here: https://sites.google.com/site/opensciencelетter/results/cjl—-one-year/gams-code

The DDPG implementation is downloaded from: https://github.com/vy007vikas/PyTorch-ActorCriticRL Note that the implementations contains several bugs, adjustments and contributions have been made in our implementation.

3.2. Network architecture

The actor network consists of three hidden ReLU layers of size 256, 128, 64, the input is the 7-dimensional state and the output is the action that lies between zero and one, which is enforced by a Sigmoid function. Prior to the input and all hidden layers, a batch normalization layer is added. The input for the critic network are the state and action, the output is an approximation of the Q-value. The state and action are concatenated in the third hidden layer, prior to the concatenation the state goes through two hidden ReLU layers, with batch normalization, of size 256 and 128. The action goes, before concatenation, through one hidden ReLU layer of size 128. Subsequently, the higher representation of the action and state go concatenated through another ReLU towards a representation of 128 dimensions. Finally, this goes through a linear layer to output the Q-value.

3.3. Hyperparameters

In Table 1, the hyperparameters of the optimized model are listed. Hyperparameters that may be not obvious are: The number of episodes, which represents after how many episodes the learning was interrupted; τ, that represents the rate in which the target network was updated.

Parameter	Value
Number of episodes	100
Batch size	128
Learning rate actor	0.0001
Learning rate critic	0.01
τ	0.001
Maximum buffer size	20,000
Exploration rate	0.2

Table 1. Fine-tuned hyperparameters

3.4. Reliability & Stability

In this work, reliability and stability are considered. Reliability in this context is defined as how reliable the end product is to find a satisfying policy. A very reliable algorithm would always find an optimal policy. When a RL algorithm is able to find a good policy, this is no guarantee it will produce the exact same output next time. Therefore the reliability is tested by running the same model multiple times and comparing results.
Stability here means how stable, or robust, the model is under small changes in either the environment or the hyperparameter setting. Lillicrap et al (2015) note that the target network greatly stabalizes the training. Therefore, a sensitivity analysis is conducted on τ.

4. Results and Analysis

In this section the results for the optimal abatement policy are presented, analyzed and interpreted.

4.1. Experimental results

First the quality of the DDPG is shown, after which it is put in perspective by considering reliability and stability.

4.1.1. DDPG performance

The trained DDPG policy is evaluated by comparing it to a random policy as baseline and the benchmark policy as aimed goal. The three policies are depicted in Figure 1.

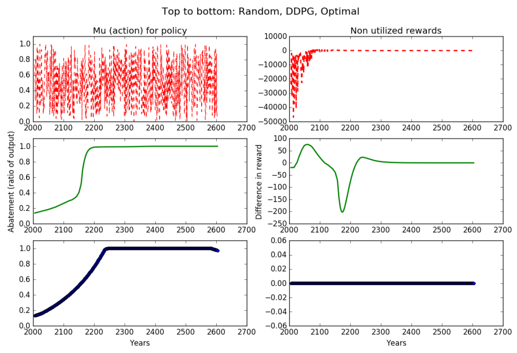

Figure 1. Random policy, DDPG policy and optimal policy with respective non-utilized reward with respect to DP rewards

On the left side, one can see the abatement rate of the different policies over the years. Top to down they are ordered; random, DDPG and benchmark. On the right hand side, the difference in rewards of all policies compared to the benchmark policy are displayed over the years. Furthermore, it can be observed that the random policy performs worse compared to the benchmark policy. Finally, the DDPG policy scores almost the same as the optimal policy over the entire period. However, at some time steps it gains reward and at some it loses reward compared to the benchmark solution.

In Figure 2, the three best policies obtained by training are shown in blue and as comparison the benchmark policy is depicted in red dashes. The policies are ranked best purely based on the highest sum of rewards. The two policies on the left look completely different compared to the benchmark, despite they gain a similar sum of rewards. The policy in the top right plot approximates the benchmark the best in terms of policy.

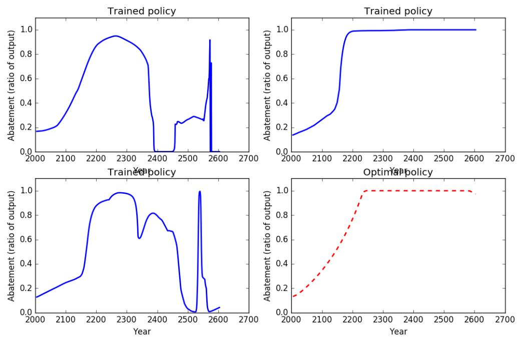

Figure 2. Top three policies of DDPG in blue and optimal policy in red.

In order to measure the performance of the DDPG policies, the difference in return between the trained policy and the random policy, and the difference between the return of the benchmark policy and the random policy is calculated. Subsequently, these values are divided in order to measure the performance of the trained policy on a scale between the performance of the random policy and the benchmark. From now on, this measurement will be appointed as *relative distance* (RD). Results are shown in Table 2.

	DDPG 1	DDPG 2	DDPG 3
Relative distance	99.68%	99.51%	99.47%

Table 2. Relative distance of different trained policies in Figure 2, in order; top-left, top-right and bottom-left.

4.1.2. Reliability

The results for reliability testing are presented in Figure 3.

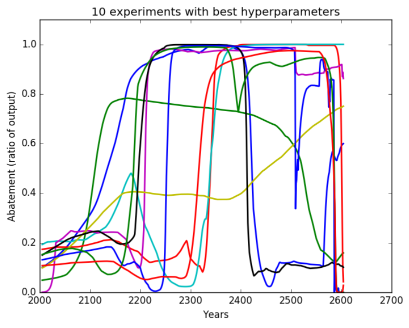

Figure 3. Policies for 10 experiments with tuned hyperparameters.

Mean RD	SD RD	Median RD
97.6%	1.1pp	97.5%

Table 3. Statistics on 10 experiment runs on tuned hyperparameters, ARD is the average relative distance. SD is the standard deviation in percent points.

It is striking that most of the policies behave differently, although there are some regular patterns present. In Table 3 the mean and variance of the relative distance are displayed, on average the policies score 97 percent on the relative distance measure with a standard deviation of 1.1 percent.

4.1.3. Stability

The results for testing the stability of the model are shown in Table 4. Experiments have been run with the hyperparameters of the best policy, only varying the value for the soft target update hyperparameter τ.

τ	0.0001	0.0005	0.001	0.005	0.01
Mean RD	98.5 %	98.6%	97.5%	95.6%	96.6%
SD RD	1.2pp	0.48pp	1.1pp	1.9pp	2.4pp

Table 4. Experiments for different values of τ with SD in percent points.

The results show that the high values for τ cause an increase in instability by showing an higher standard deviation in the relative distance measure. Further, the value 0.0005 appears to be optimal, since it performs the best on average and has the lowest standard deviation.

4.2. Result Analysis

Now that all results are displayed, the different parts can be interpreted. Building a foundation for the final conclusions.

4.2.1. DDPG performance

The DDPG optimization proves able to obtain policies for which the total reward is in a very close range to the benchmark reward. From the additional reward which there is to gain by using a smart policy, the best DDPG policies capture over 99.5 percent. These are highly promising results. Remarkable still remains the visual difference when looking at these obtained policies. Figure 2 shows that a policy is not merely able to obtain high reward when it looks closely like the benchmark policy. One important explanation for this is the discounting over time. The discounting is still an important topic of discussion in DICE modelling, partly because of its strong implications. We see for the top 3 models that all of them start with an almost identical policy for the first 200 years. Due to the time discounting, the obtained rewards later on in time are much less relevant for the total reward. This is also the explanation for the behaviour of the non utilized rewards in Figure 1. The non utilized rewards look like it is tending towards zero over time, but this is only the case because it is seen in one scale with other points in time.

4.2.2. Reliability & Stability

In order to see whether the above results are reliable, Figure 3 shows policies for 10 runs with the tuned DDPG model. There is a wide variance between the models, however some key features can be noticed. All policies seem to make a similar starting 200 years, this again is resulting from a time dependent importance of the policy due to the discounting. From a practical perspective, the optimal policy for the upcoming years is indeed by far the most important since that indicates what (political) actions need to be taken now. Further, it remains a drawback that RL returns different outcomes for different runs. This problem can be reduced by a more elaborated hyperparameter search.

To increase insight in the stability of the model, sensitivity analysis of the τ parameter is conducted. The results for this are shown in table 4. The τ parameter controls the speed by which the target network converges towards the (changing) source network. Since the source network is updated during this process by back propagation, the τ also controls how fast adjustments based on the data will be fully incorporated in the model. Therefore, we expect the tuning of the τ to be highly dependent on the learning rate settings. Given the used settings, the sensitivity of the model to τ does not seem to be out of proportion. For all values of τ the model still achieves high rewards with a low standard deviation. This is a positive finding. However, note that many other stability test would be relevant for this model.

5. Conclusion

To conclude, the deep deterministic policy gradient shows to be effective in optimizing abatement policies within DICE models, therefore significantly expanding the accessibility and flexibility of the set of DICE models. Although DDPG did not accomplish the exact same policy as the benchmark problem, it did invent a policy that achieves the same results up to 99 percent. In the near future, DDPG could contribute to the DICE model in several ways. The model's complexity could be increased drastically, as reinforcement learning is capable of dealing with high dimensional problems without ending in infinite exploration. Moreover, it can be used for a stochastic DICE model where uncertainties are incorporated, which is hard to handle in Dynamic Programming.

6. Discussion

Although the results of our model are promising, it remains unsatisfactory that the benchmark policy could not be matched exactly. There are several reasons that could explain the small gap. Many different hyperparameters are present in DDPG. After multiple experiments it was noticed that these parameters have a high influence on the performance. Unfortunately, due to a lack of time, not the entire hyperparameter space could be searched, although it might still improve the performance. Another possible reason could be an insufficient exploration of the state space, which would force the algorithm to find a proper solution for the terrain it is familiar with. Moreover, a lack of complexity of the actor and critic could cause the gap between the DDPG and the benchmark policy, which potentially can be solved with deeper or broader networks.

Furthermore, DDPG is well-known for its instability. Therefore, it is necessary to add target-networks, replay-memory and batch normalization layers. Still, instability was a major issue during training of the network, since there seems to be a very small hyperparameter space where the algorithms shows learning.

Moreover, running multiple experiments with exactly the same settings led to different policies. The algorithm often rolls into a local optimum, which means that the algorithm requires several runs. Nevertheless, the obtained optimal policies are reliable, they will behave identically in every run. However, the policies might not be reliable in the real world as the DICE model is a simplified version of the real worlds economy and climate.

Interestingly, different types of good policies were obtained. Some policies are behaving more consistently than others, which is not a problem in reinforcement learning perspectives. However, implementing such an inconsistent policy in society would likely arise questions to citizen. Therefore, consistent policies are likely to be more meaningful.

7. Future Work

As mentioned, the DICE model is heavily constrained in its form due to computational constraints. Now that Reinforcement learning shows to be capable of solving complex tasks, a wide range of opportunities arises. For example, stochasticity in the DICE model can be included much more elaborate, without having explosive increase in costs. This way, state transitions and tipping points can be modelled much more advanced. Besides, uncertainty can be measured more precisely, which is of major importance for these long-term forecasts.

References

Cai, Y., Judd, K. L., & Lontzek, T. S. (2015). The social cost of carbon with economic and climate risks. *arXiv preprint arXiv:1504.06909*.

Cai, Y., Lenton, T. M., & Lontzek, T. S. (2016). Risk of multiple interacting tipping points should encourage rapid co 2 emission reduction. *Nature Climate Change*, *6*(5), 520.

Hodson, R. (2017). Climate change. *Nature*, *550*(7675), S53–S53.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097–1105).

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., … Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Nordhaus, W. (2014). Estimates of the social cost of carbon: concepts and results from the dice-2013r model and alternative approaches. *Journal of the Association of Environmental and Resource Economists*, *1*(1/2), 273–312.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., & Riedmiller, M. (2014). Deterministic policy gradient algorithms. In *Icml*.

Uhlenbeck, G. E., & Ornstein, L. S. (1930). On the theory of the brownian motion. *Physical review*, *36*(5), 823.