Geosciences

AIMS Geosciences, 8(3): 488–502.
DOI: 10.3934/geosci.2022027
Received: 06 May 2022
Revised: 18 July 2022
Accepted: 31 July 2022
Published: 08 August 2022

http://www.aimspress.com/journal/geosciences

Research article

Reinforcement learning in optimization problems. Applications to geophysical data inversion

Paolo Dell'Aversana*

Eni S.p.A., San Donato Milanese, Milan, Italy

* **Correspondence:** Email: dellavers@tiscali.it; Tel: 390252063217.

Abstract: In this paper, we introduce a novel inversion methodology that combines the benefits offered by Reinforcement-Learning techniques with the advantages of the Epsilon-Greedy method for an expanded exploration of the model space. Among the various Reinforcement Learning approaches, we applied the set of algorithms included in the category of the Q-Learning methods. We show that the Temporal Difference algorithm offers an effective iterative approach that allows finding an optimal solution in geophysical inverse problems. Furthermore, the Epsilon-Greedy method properly coupled with the Reinforcement Learning workflow, allows expanding the exploration of the model-space, minimizing the misfit between observed and predicted responses and limiting the problem of local minima of the cost function. In order to prove the feasibility of our methodology, we tested it using synthetic geo-electric data and a seismic refraction data set available in the public domain.

Keywords: reinforcement learning; geophysical inversion; optimization; refraction seismic; electric tomography; machine learning

1. Introduction

In mathematics, computer science and economics, as well as in other disciplines like geophysics, solving an optimization problem consists of finding the best of all possible solutions in a given model space [1]. This target can be realized by minimizing (or maximizing) some type of objective function that includes, in many practical cases, the difference between observed and predicted quantities. For instance, in geophysics, a typical optimization problem is finding an Earth-model consisting of seismic-velocity spatial distribution that minimizes the differences between observed and predicted seismic travel times [2].

Optimization techniques can be divided into approaches that allow exploring locally the model space, and approaches that allow a global or quasi-global search of the solution. In the first case, we generally incur in the problem of convergence towards local minima (or local maxima) of the cost function. In fact, the final solution will depend strongly on the initial model and on the exploration path in the parameters space. In general, when we apply local optimization techniques, we search for a solution in a limited portion of the model space, converging towards solutions that could not correspond with the best one for our specific problem. In order to face this problem, Global optimization techniques are addressed to find the global minimum (or the global maximum) of the objective function over the given set. Unfortunately, finding the global minimum (or maximum) of a function commonly represents a difficult task. Analytical methods are frequently not applicable and the use of numerical solution strategies often is not sufficient [3]. Typical techniques based on global or quasi-global search in the model space [4], include stochastic methods like Direct Monte-Carlo sampling approaches. Other methods are based on heuristic approaches to explore the model space in a more or less intelligent way. These include, for instance, Ant Colony optimization (ACO), Simulated annealing, Evolutionary algorithms (e.g., genetic algorithms and evolution strategies), and so forth. Despite the many advantages, these types of global optimization methods are generally difficult to put in practice in many situations, especially in three dimensions, due to the very expensive computational process when dealing with large parameter spaces.

In order to face the intrinsic problems of both local and global optimization methods, in this paper, we propose to reformulate the optimization problems in terms of Reinforcement Learning (RL). Our approach aims to teach an "artificial agent" to search for the global minimum of the cost function in the model space using the advantages offered by a large suite of Reinforcement Learning algorithms. These are aimed at mapping situations to actions through the maximization of a "numerical reward signal" [5–13]. In every particular state, an artificial agent learns progressively by continuous interaction with its environment. This can be a true physical environment, as it happens, for instance, in case we desire to teach an agent to move through a real physical space. More in general, the environment can consist of a virtual space with which one or more artificial agents interact. The effects of every agent's action will be returned by the modified environment in terms of a reward (or a punishment) and a new state. The reward depends on the "quality" of the agent's actions. High rewards correspond with positive impact of the actions on the agent's target, and vice versa. For instance, if the objective of the artificial agent is to find the exit from a maze in the shortest possible time (or through the shortest path), the agent will receive a positive reward every time it moves properly to reach the exit.

The final objective of such a learning strategy is to maximize the *total reward* accumulated during all iterations (cumulative reward), and not just the immediate reward. In the example of the maze, it means that the agent's objective is to find a global strategy to escape from the maze, rather than just selecting a single local step forward that could lead him into a dead end. This is a crucial point, because the goal of Reinforcement Learning methods is optimizing the agent's actions for a long-term horizon. Such an intrinsic forward-looking approach of RL algorithms can be used with profit to find global solution(s) in many optimization/inversion problems in geophysics (as well as in other fields). In fact, it is easy to grasp the analogies and possible points of connection between geophysical inversion problems and Reinforcement Learning. In the first case, the goal is to find an Earth model that corresponds to a minimum value of a certain cost function. In the second case, the goal is to find an optimal policy through which an agent can maximize its total reward. These are both examples of optimization problems.

In the next methodological section, we will see how the geophysical inverse problem can be reformulated in terms of Reinforcement Learning strategy. For that purpose, we will use a combination of Q-Learning, Temporal Difference and Epsilon-Greedy algorithms. We will see that these methods fit the purpose of optimizing the exploration of the parameter-space in inversion problems. Finally, we will test our approach using synthetic geo-electric data, plus a seismic data set available in the public domain.

2. Theoretical framework

Reinforcement Learning includes a suite of algorithms and techniques through which an "artificial agent" learns an optimal "behavior" by interacting with a dynamic "environment" and by maximizing a "reward metric" for the task, without being explicitly programmed for that task and without human intervention. The artificial agent selects those actions that allow increasing the cumulative reward, $r \in R$, achievable from a given state, $s \in S$ (Figure 1).

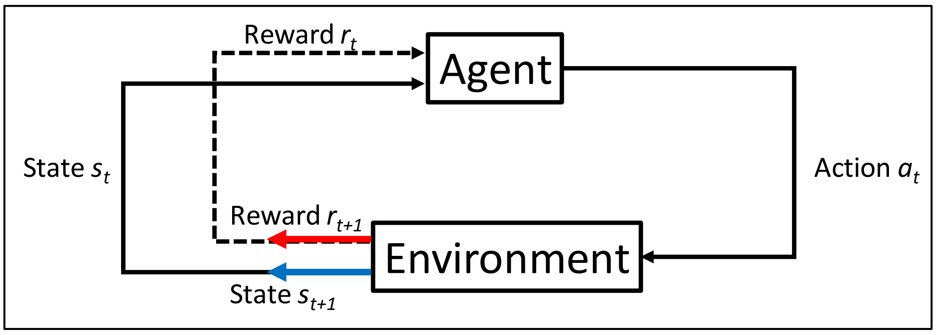


Figure 1. Conceptual scheme of Reinforcement Learning.

A "discount factor", γ, is applied to the long term rewards with the scope of giving progressively lower weights to rewards received far in the future. The agent's goal is to learn, by trials and errors, a "policy" for maximizing such cumulative long-term reward. The policy is often denoted by the symbol π. It consists of a function of the current environment state, s, belonging to the set S of all possible states, and returns an action, a, belonging to the set A of all possible actions.

$$\pi(s): S \rightarrow A. \tag{1}$$

There are many different Reinforcement Learning techniques. Among the various methods, the Q-Learning method [14] is a suitable approach for solving optimization/inverse problems. The name derives from the Q-function that provides a measure of the *Quality* (in terms of effectiveness for a given task) of an action that the agent takes starting from a certain state. It is defined as follows:

$$Q(s, a) = S \times A \rightarrow R. \tag{2}$$

The Bellman equation below provides an operative definition of the maximum cumulative reward. This is given by the reward r that the agent received for entering the current state s and action a, plus the maximum future reward for the next state s', taking all the possible actions a' from that state:

$$Q(s, a) = r + \gamma \, max_{a'} \, Q(s', a'). \tag{3}$$

In formula (3), the symbol γ indicates the "discount factor". It is introduced for balancing the contribution of future rewards with respect to the immediate reward. The value of $Q(s, a)$ can be found recursively: the algorithm starts by using random values (or any guess value) for the Q-function. Then, when the agent proceeds exploring its environment, the initial Q values progressively converge towards the optimal ones, based on the positive and/or negative feedback that the agent receives from its environment. The "Temporal Difference" (briefly TD) method (formula 4 below) provides a practical way for updating the Q values, as follows:

$$Q^{new}(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \cdot \left[r_t + \gamma \cdot \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right] \tag{4}$$

We can see that the new value of Q for state s_t and action a_t, is obtained by adding to the previous Q value a new term (in the square parenthesis) called *temporal difference*. This, in turn, is multiplied by a factor α that represents the *learning rate* and is commonly determined empirically by the user. The temporal difference consists of the immediate reward, r_t, plus the difference between the maximum Q value for all the actions that the agent can take from the state s_{t+1}, minus the old value of Q. The $\max_a Q(s_{t+1}, a)$ term is multiplied by the above mentioned discount factor, γ.

Now, we must explain how we define the Q values in the frame of our integrated Inversion-Reinforcement Learning (called, briefly, *RL-Inv*) approach. In other words, we must clarify how we assign a reward to the artificial agent (the optimization algorithm) while it explores the model space. In our method, we set the Q-function inversely proportional to the cost function (that, in turn, depends on the difference between observed and predicted responses) after a certain number N of iterations. The user determines such N value empirically. Indeed, we assume that a good convergence path towards a final low misfit represents a reasonable long-term reward for our Reinforcement Learning agent. In that case, low misfit (as well as low values of the cost function) correspond to high rewards and high Q values.

For instance, let us suppose that we apply a Least Square optimization algorithm to solve our inverse problem; that algorithm coincides with our agent. In that case, we can define the cost function $\Phi(\boldsymbol{m})$ as follows:

$$\Phi(\boldsymbol{m}) = (\boldsymbol{d}_{obs} - g(\boldsymbol{m}))^T \, \boldsymbol{W_d} \, (\boldsymbol{d}_{obs} - g(\boldsymbol{m})) + \eta \cdot \boldsymbol{m}^T \boldsymbol{Rm} \, . \tag{5}$$

In formula (5), $\boldsymbol{m}$ represents the vector of model parameters, or model vector; $\boldsymbol{d_{obs}}$ represents the data vector (observations); $g(\boldsymbol{m})$ is the forward operator by which we calculate the predicted response in the model vector $\boldsymbol{m}$; the symbol T indicates "transpose"; $\boldsymbol{W_d}$ is he data covariance matrix for taking data uncertainties into account; $\boldsymbol{R}$ is a smoothing operator applied to the model vector $\boldsymbol{m}$ as a regularization term; η is a factor regulating the weight of the smoothing term in the cost function.

In our procedure, we calculate $\Phi(\boldsymbol{m})$ at each iteration and store its value at every iteration. In such a way, we can calculate and store the correspondent Q value as follows:

$$Q(s_t, a_t) \approx {}^{1}/_{\Phi\,(\boldsymbol{m})}. \tag{6}$$

Next, let us clarify how the Q-Learning formulas contribute to the inversion. In the frame of the Q-Learning approach, we need to estimate a cumulative reward by taking into account both the immediate as well as the long-term reward. In our approach, the immediate reward is given by the inverse of the cost function after just one or two iterations, as in formula (6). Instead, the long-term reward is given by the inverse of the cost function estimated after a "significant number" of iterations (such number depends on the inverse problem and is decided by the user, case by case). In such a way, we intend to set a policy that minimizes the cost function through a balanced combination of both short-term and-long term views. This concept will be further expanded in the next two sections.

The Bellman equation (3) and the Temporal Difference iterative method (4) allow us estimating and progressively updating the values of the Q-function during the optimization (inversion) process. These values depend on the starting models and on the exploration paths in the model-space. The goal of our approach is to find an optimal policy for our optimization agent. Such a policy will coincide with the "optimal" exploration/exploitation path in the model space aimed at maximizing the Q-function. Hence, a crucial point is how the model space (that represents the environment of our Reinforcement Learning approach) is explored.

2.1. Q-Learning, model space exploration and inversion

In the frame of geophysical inversion (as well as in other optimization problems), the environment of the Reinforcement Learning problem is represented by the space of model parameters, or model space (Figure 2). As we said earlier, the agent corresponds with the optimization algorithm through which we try to minimize the cost function. At each iteration, the algorithm performs an action: it explores the environment in order to update the current geophysical model with the goal to reduce the misfit between observed and predicted responses. In our approach, we perform such an exploration using the Epsilon-Greedy algorithm. This provides an effective strategy for facing the well-known "Exploration vs. Exploitation" question. Let us explain the basics of this strategy and the reason why we included it in our approach.

Exploration allows an agent improving its current state at each action, leading to a *long-term benefit*. In the frame of geophysical inversion, this corresponds to retrieve a distribution of model parameters that allows lowering the cost function (or the misfit) and, consequently, improving the Earth model. On the other hand, *exploitation* means to choose the greedy action to get the most *short-term reward* by exploiting the agent's current action-value. For instance, in case of Gradient-based optimization methods, this action corresponds to taking repeated steps in the opposite direction of the gradient of the cost function. The crucial point is that by being greedy with respect to immediate action-reward estimates, may not actually lead towards the maximum long-term reward, causing a sub-optimal behaviour. In other words, trying to minimize the cost function at each step could not represent the optimal inversion policy.

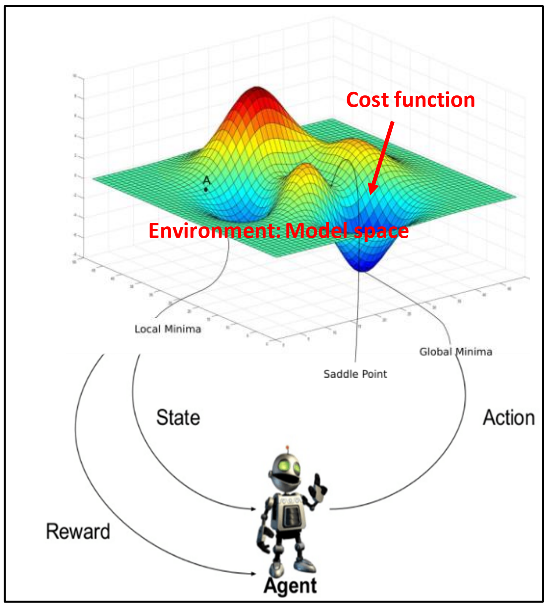
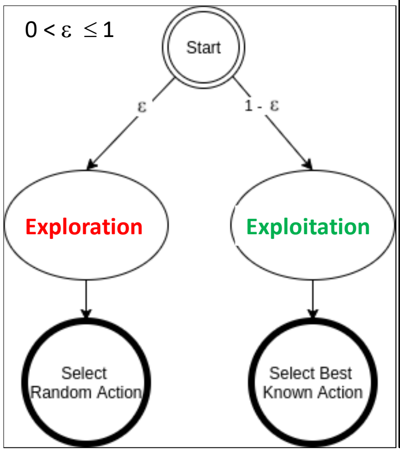


Figure 2. Conceptual link between the Reinforcement Learning approach and the exploration of the model space in optimization problems.

2.2. Epsilon-Greedy approach

Epsilon-Greedy is an effective approach aimed at balancing *exploration* and *exploitation* by choosing randomly between these two possibilities. The term "epsilon" refers to the probability of choosing to explore that is commonly lower than the probability to exploit. In other words, the optimization/inversion algorithm exploits most of the time with a small chance of exploring. It means that it updates the model parameters respecting the condition of reducing the cost function at each iteration (exploitation). However, it explores the model parameters (with lower probability: epsilon << 1) in different directions too, even if that choice could imply a temporary increase of the cost function. Figure 3 shows a scheme of such approach and its pseudo-code.



```
Choose epsilon; # exploration probability
Choose n; # number of iterations

for i = 1 to n do:
        p = pick a random number from 0 to 1
        if p < epsilon:
                current_model = pick model at random # explore
        else:
                current_model = pick best model so far # exploit
reward = current_model.Bellman_Q # use Bellman eq. to calculate Q
Store Q value and update Q-Table
```

Figure 3. Scheme of the Epsilon-Greedy approach (left) and its pseudo-code (right).

At the same time, by applying the Bellman equation and the Temporal Difference method, we aim to a long-term reward that is minimizing the cost function after a significant number *N* of iterations (and not just the cost function at each individual iteration). This strategy allows us sampling large portions of the model space that otherwise would be excluded by a traditional greedy optimization strategy. Finally, we will get the optimal inversion policy. This uses the best exploitation/exploration strategy, produces the lowest final value of the cost function and the best inverted model.

The block diagram of figure 4 summarizes the entire procedure, showing the sequence of steps through which we update the model parameters by maximizing the Q-function through a combination of Epsilon-Greedy exploration strategy and Bellman/Temporal Difference equations.

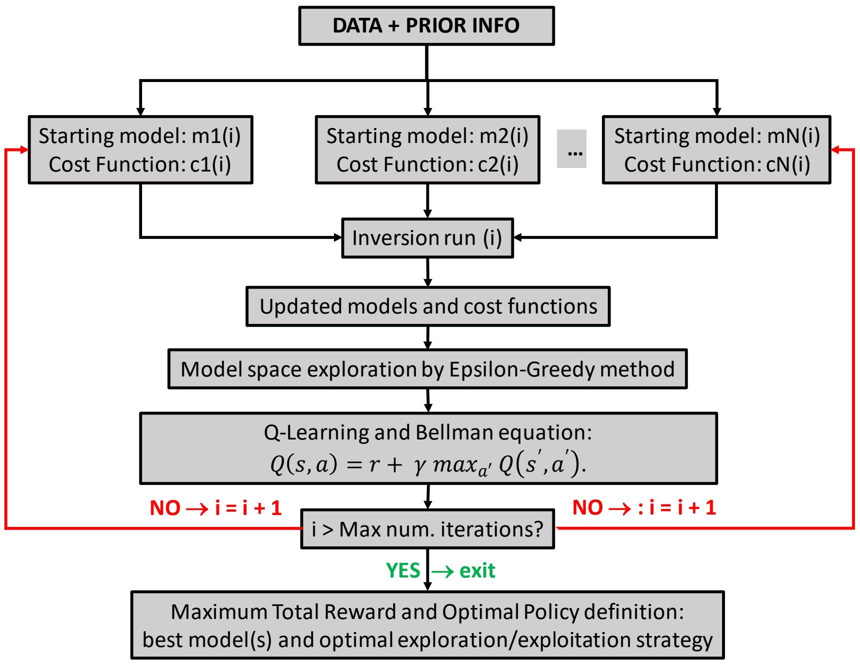

DATA + PRIOR INFO
Starting model: m1(i) Cost Function: c1(i)
Starting model: m2(i) Cost Function: c2(i)
...
Starting model: mN(i) Cost Function: cN(i)
Inversion run (i)
Updated models and cost functions
Model space exploration by Epsilon-Greedy method
Q-Learning and Bellman equation: $Q(s,a) = r + \gamma\, max_{a'}\, Q(s',a')$.
NO → i = i + 1
NO → : i = i + 1
i > Max num. iterations?
YES → exit
Maximum Total Reward and Optimal Policy definition: best model(s) and optimal exploration/exploitation strategy

Figure 4. Block diagram of the Reinforcement Learning-Inversion (*RL-Inv*) approach.

With reference to figure 4, in order to clarify better how and where the Q-Learning formulas contribute to the inversion process, we schematize the entire workflow through the following key steps: 1) Create *m* starting models (process initialization). 2) Choose *n* (number of iterations). 3) Run *n* iterations for each model. 4) Update each model after *n* iterations. 5) Calculate the inverse of cost function (eq.6) after *1* or 2 iterations (short-term reward for each model). 6) Calculate the inverse of cost function (eq.6) after *n* iterations (long-term reward for each model). 7) Calculate (or update) the cumulative reward (Q values) using the Bellman and TD formulas (eqq.3 and 4). 8) Store Q values and update the Q-Table. 9) Chose epsilon (for the epsilon-Greedy method), as shown in figure 3. 10) Select model with the highest total reward with probability = 1-epsilon (exploitation). 11) Alternatively, select random model with probability = epsilon (exploration). 12) Use the selected model, perturb it and create other *m* initial models. 13) Iterate from step 3. 14) Exit from the loop when the cost function and the cumulative reward Q is stationary. 15) Finally, select the model with the highest Q-value (lowest cost function).

3. Examples

In this section, we discuss two tests where we apply the RL-Inv method to two types of data set. In the first case, we use synthetic data obtained through a simulated resistivity survey. In the second case, we use refraction seismic data available in the public domain. For each test, we compare the final models obtained through a "standard" inversion/optimization approach and the RL-Inv methodology.

3.1. Synthetic test on geo-electric data

In this test, we simulated the acquisition of DC (Direct Current) geo-electric data along a line 550 m long, with electrodes deployed with a regular spacing of 10 m. The upper panel of figure 5 shows the "true" resistivity scenario in which we simulated the resistivity survey. The model consists of two stacked resistive layers embedded in a conductive uniform background. The lower panel of the same figure shows the data (apparent resistivity section) of the simulated DC response. After adding 5% of Gaussian noise to the simulated response, our goal was to invert the synthetic data in order to retrieve the correct resistivity model. We started from a half-space initial guess, assuming no a priori information.

Despite its apparent simplicity, the resistivity model shown in figure 5 is not easy to retrieve by data inversion without using any prior information. Many equivalent geophysical models can honour the data equally well if we do not use any constraint. The inversion algorithm that we used in this case is a "standard" Damped Least Square optimization algorithm that minimises iteratively the cost function, like the one expressed by eq. (5). The regularization operator consists of a smoothing functional that allows finding smoothed model solutions. The effect will be that the two resistive layers cannot be adequately distinguished and, after the inversion process, they appear "mixed" into a unique layer. This is clearly shown in figure 6.

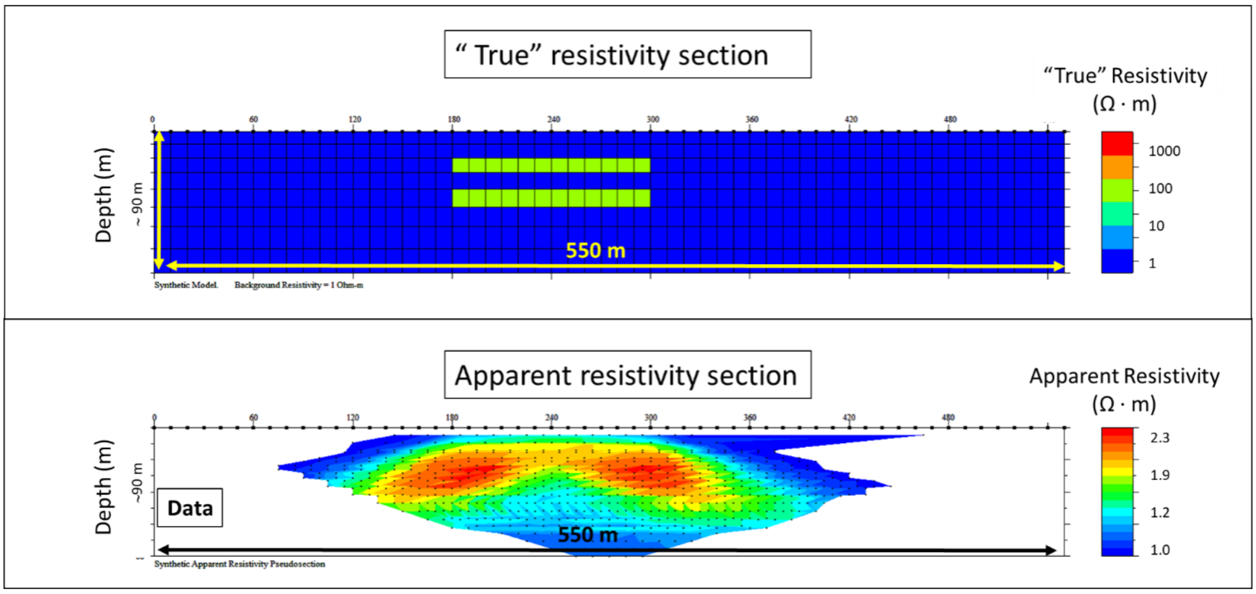


Figure 5. "True" (original) resistivity model (upper panel) and observed apparent resistivity (lower panel). Colour scale represents resistivity, in Ω·m.

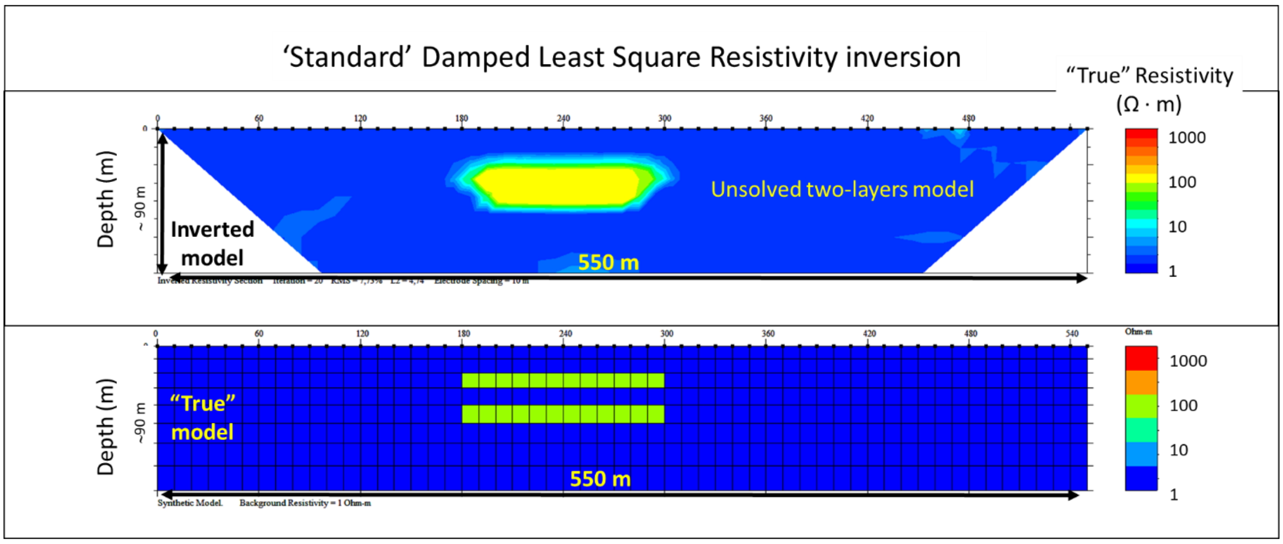


Figure 6. Inverted resistivity model (upper panel) using a Damped Least Square Optimization algorithm. The "true" model is shown again in the lower panel, for comparison.

Next, we performed again the inversion of the same synthetic data, but this time through our Reinforcement Learning approach (RL-Inv), in order to verify if it was possible to find an inverse solution more consistent with the original resistivity model. Figure 7 shows the inverted resistivity model (upper panel). In this case, the RL-Inv solution shows the two resistive layers properly separated. Furthermore, they were retrieved with almost correct resistivity values, although the resistivity of the upper layer is slightly overestimated.

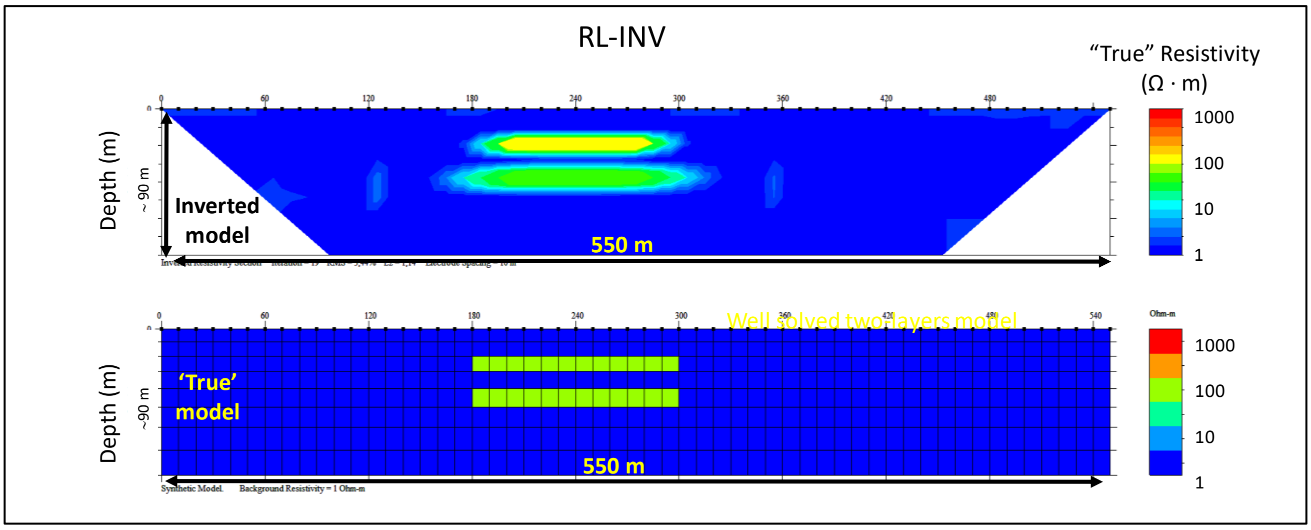


Figure 7. Inverted resistivity model (upper panel) using the RL-Inv approach. The "true" model is shown again in the lower panel, for comparison.

Figure 8 shows the cross plot of predicted vs. observed apparent resistivity for both inversion results. This type of graph is useful because it provides a synoptic view of the misfit between observed and predicted geo-electrical responses. In case of perfect fit, the points should be on a 45-degree tilted line (green line in the figure). The scattering of the points above the ideal best-fit line is a measure of the misfit and of the noise in the data. Both cross plots show some level of scattering and of resistivity overestimation; however, the misfit of the second inversion result (from RL-Inv) is less than the one obtained through the traditional Damped Least Square approach. Furthermore, the second scattering cross-plot shows two clusters of scattered points that are related with the two separate resistivity layers.

In summary, the RL-Inv approach produced results that are more consistent with the original resistivity scenario used for the simulation.

3.2. *Test with public seismic data*

In this second example, we applied the RL-Inv method to a classical refraction seismic data set with heterogeneous overburden and some high-velocity bedrock. This data set is included in the examples provided in the public-domain repository prepared for testing the open source "pyGIMLi software library" [15]. Figure 9 shows the data set in terms of travel times vs. offsets. The complex trends of the travel-time curves vs. offset suggest significant variability in the velocity field. We can observe frequent variations in the slope of the curves that indicate lateral as well as vertical velocity changes. Such complexity in the data space corresponds to a similar complexity in the model space. In scenarios like this, our RL-Inv approach can be useful to find a global solution for the refraction tomography problem, limiting the risk to fall in local minima of the cost function during the inversion process. We followed the scheme of Figure 3 by exploring the model space through the Epsilon-Greedy approach. First, we created an initial Q-Table based on the cost function values (here expressed in terms of *Chi*^{2} values) for a set of different starting models (Table 1). Next, the optimization agent started exploring the model-space (in this case, the unknown model parameter is P-Velocity, V_p) through the Epsilon-Greedy approach.

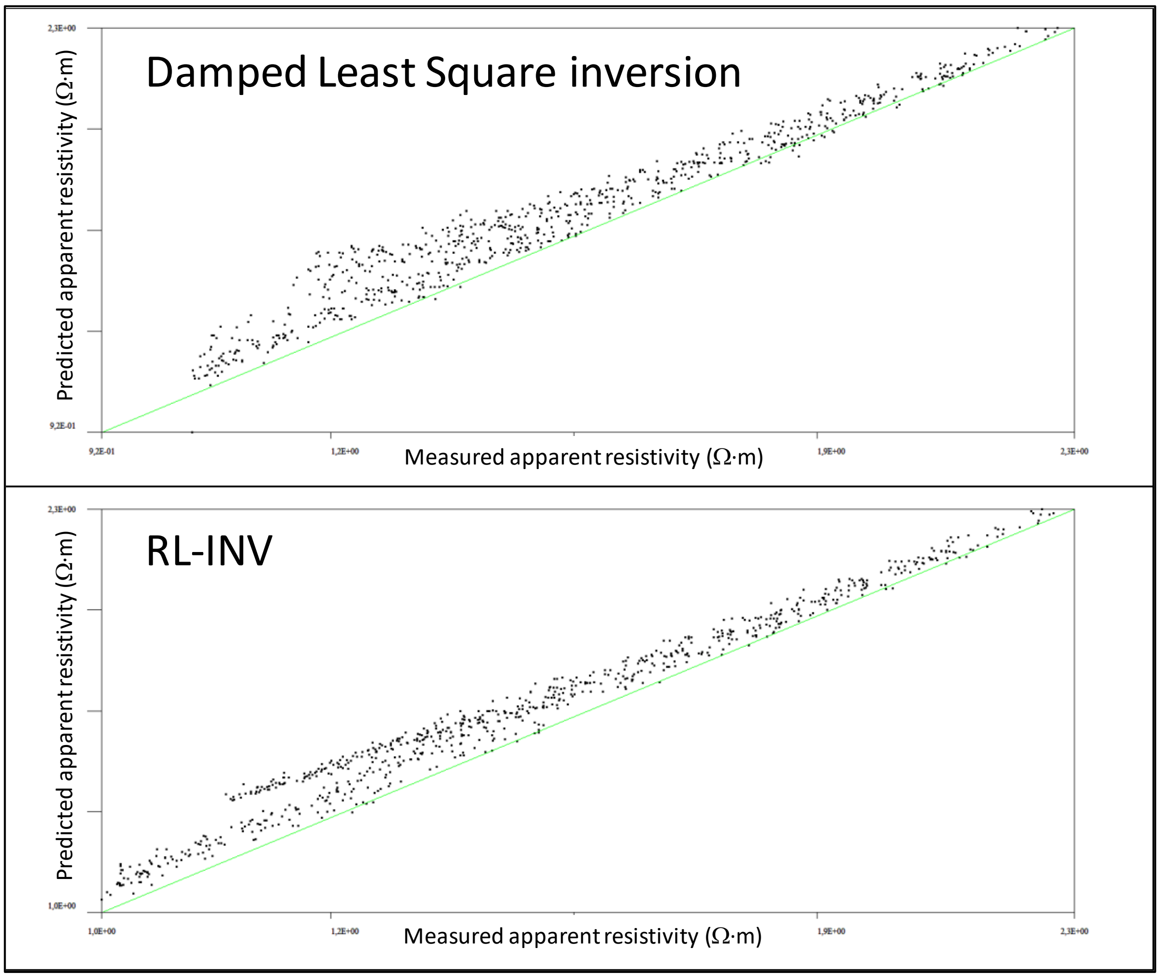


Figure 8. Cross plot of predicted vs. observed apparent resistivity for the Damped Least Square inversion result (upper panel), compared with the cross plot for RL-inv results (lower panel).

Figure 10 shows an example of "Model selection histogram" obtained through exploration of the model space with the Epsilon-Greedy method. The bars of each histogram are proportional to the probability to select one model among many possible starting models. In this example, we have considered just 20 possible candidate models, for illustrative purposes. For each model, we calculated the cumulative reward using the Bellman formulas, as explained earlier in the methodological section. We can see that for low values of the epsilon parameter, the method selects almost exclusively the model(s) with high cumulative reward (some examples are indicated by the arrows in the figure 10). This corresponds to adopt a greedy strategy, with prevalence of exploitation of the model(s) with high reward. On the other side, by choosing high values of epsilon, model selection tends to be random, allowing exploring the model space through directions that would otherwise have been ignored. In other words, an appropriate setting of the epsilon parameter allows a balanced policy between exploration and exploitation in the model space during the inversion process. In this specific test, we performed many tests by setting the epsilon parameter in the range between 0.0 and 1.0. There is not any absolute rule to find the optimal value of epsilon. However, a good strategy is to make epsilon variable: as trials increase, epsilon should decrease. Indeed, as trials increase, we have less need of exploration and more convenience of exploitation, in order to get the maximum benefit from our policy.

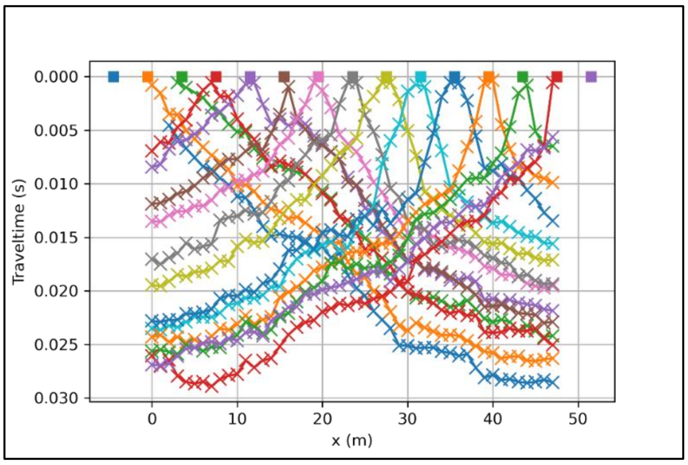


Figure 9. Data set: refraction travel-times (s) vs. offsets, x(m).

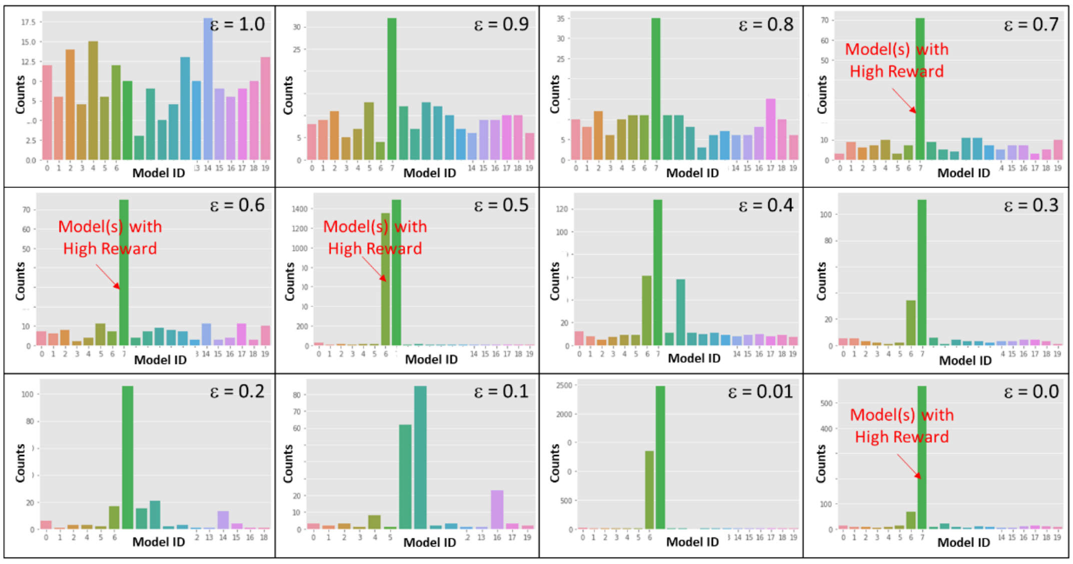
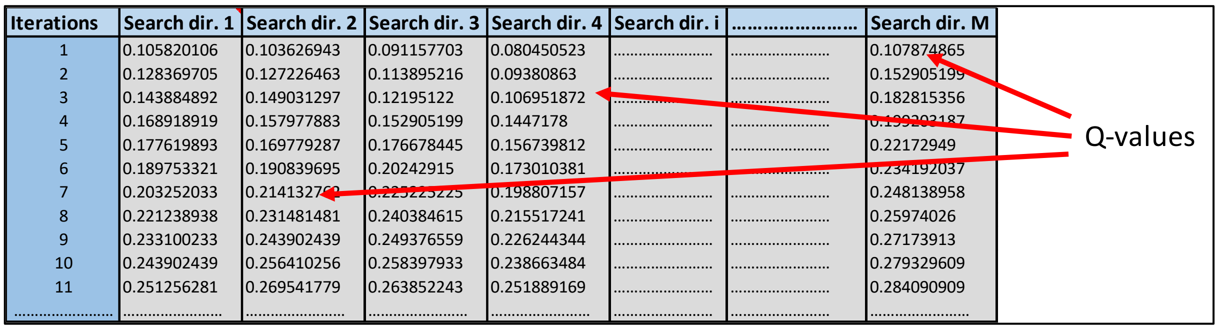


Figure 10. Example of "Model selection histograms" using the Epsilon-Greedy method, for variable values of epsilon. Test on 20 different models.

During the inversion process, the Q-Table was progressively updated. As explained earlier, the rule for updating the Q-Table is given by the Bellman equation and the iterative Temporal-difference method. In summary, the agent (the minimization algorithm) explores the model space and selects the optimal path that corresponds with the direction in the space of parameters with the highest cumulative reward. At the same time, it does not neglect to explore alternative directions in the model space, although with lower probability. After many iterations, the agent learns to move in the model space following the most convenient policy. This corresponds with the one that allows finding the global minimum of the cost function. Our inversion test seems to confirm the effectiveness of such strategy, as in the previous test. Figure 11 shows some examples of velocity models obtained by travel-time tomography, with the correspondent ray tracing. Each individual model corresponds to a certain point of the cost function in the model space. For each path explored in the model space, we have a correspondent suite of values of the cost function. Finally, the best model (left panel of Figure 12) is the one retrieved through the RL-Inv approach. It shows the V_p parameter distribution that corresponds to the highest cumulative reward. For comparison, the right panel of the same figure shows the V_p model obtained without the support of the RL approach, using a "standard" optimization approach. Compared with the RL-Inv solution, the "standard" solution tends to overestimate the bedrock velocity and is not able to highlight properly the heterogeneities in the overburden.



Iterations	Search dir. 1	Search dir. 2	Search dir. 3	Search dir. 4	Search dir. i	....	Search dir. M
1	0.105820106	0.103626943	0.091157703	0.080450523	....	....	0.107874865
2	0.128369705	0.127226463	0.113895216	0.09380863	....	....	0.152905199
3	0.143884892	0.149031297	0.12195122	0.106951872	....	....	0.182815356
4	0.168918919	0.157977883	0.152905199	0.1447178	....	....	0.199203187
5	0.177619893	0.169779287	0.176678445	0.156739812	....	....	0.22172949
6	0.189753321	0.190839695	0.20242915	0.173010381	....	....	0.234192037
7	0.203252033	0.214132762	0.225225225	0.198807157	....	....	0.248138958
8	0.221238938	0.231481481	0.240384615	0.215517241	....	....	0.25974026
9	0.233100233	0.243902439	0.249376559	0.226244344	....	....	0.27173913
10	0.243902439	0.256410256	0.258397933	0.238663484	....	....	0.279329609
11	0.251256281	0.269541779	0.263852243	0.251889169	....	....	0.284090909
....	....	....	....	....	....	....	....

Table 1. Q-Table filled with the inverse values of the cost function for each search direction.

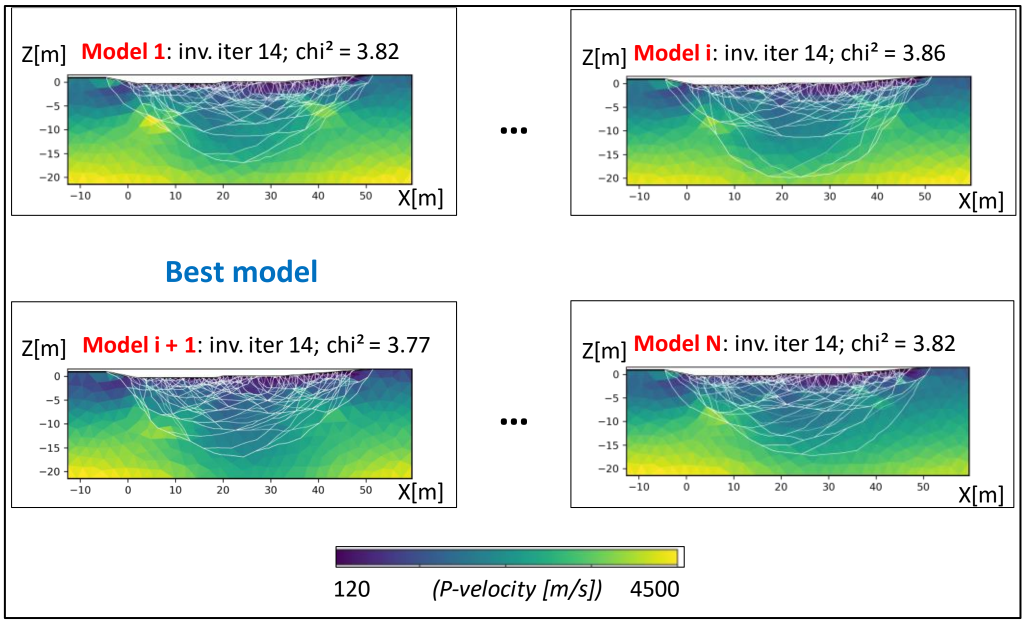


Figure 11. Examples of velocity models obtained by travel-time tomography, with the correspondent ray tracing.

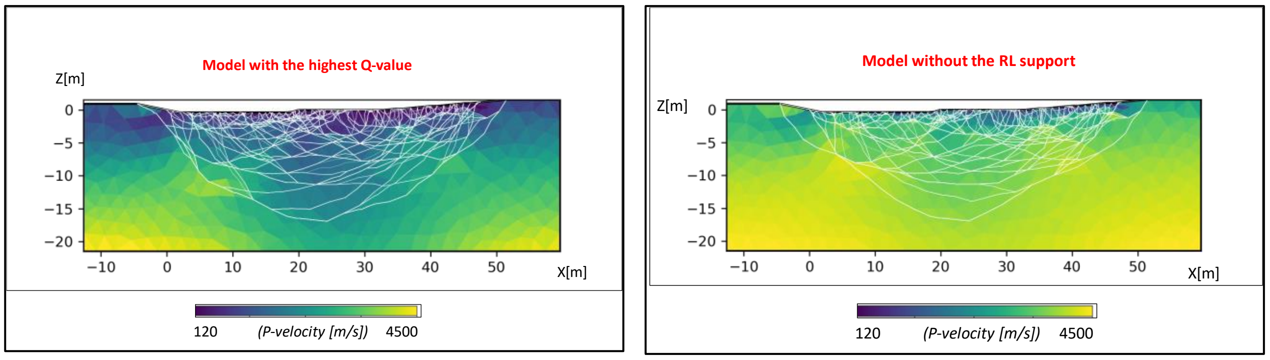


Figure 12. Comparison between the inverted Vp models obtained by RL-Inv (left) and by a "standard" seismic refraction tomography approach (based on generalized Gauss-Newton optimization method (right).

4. Conclusions

We introduced a new optimization/inversion approach fully integrated with Q-Learning, Temporal Difference and Epsilon-Greedy methods. These allow expanding the exploration of the model-space, minimizing the misfit and limiting the problem of falling in local inversion minima. The advantages of our approach are clearly highlighted through the comparative test results on multidisciplinary data (electrical and seismic). Finally, we remark that we expect the greatest benefits from our method in those applications where an extended exploration of the model-space is difficult or prohibitive, due to the size of the data-model space and the complexity of the inversion problem. For instance, interesting cases include full-wave seismic inversion and simultaneous joint inversion of multi-physics data.

Conflict of interest

The author declares no conflict of interest.

References

1. Boyd SP, Vandenberghe L (2004) *Convex Optimization*, Cambridge University Press, 129.
2. Tarantola A (2005) *Inverse Problem Theory and Methods for Model Parameter Estimation*, SIAM. https://doi.org/10.1137/1.9780898717921
3. Horst R, Tuy H (1996) *Global Optimization: Deterministic Approaches*, Springer.
4. Neumaier A (2004) Complete Search in Continuous Global Optimization and Constraint Satisfaction. *Acta Numerica* 13: 271–369. https://doi.org/10.1017/S0962492904000194
5. Raschka S, Mirjalili V (2017) *Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow*, PACKT Books.
6. Russell S, Norvig P (2016) *Artificial Intelligence: A Modern approach*, Pearson Education, Inc.
7. Ravichandiran S (2020) *Deep Reinforcement Learning with Python*, Packt Publishing.

8. Duan Y, Chen X, Houthooft R, et al. (2016) Benchmarking deep reinforcement learning for continuous control. *ICML* 48: 1329–1338. https://arxiv.org/abs/1604.06778
9. Ernst D, Geurts P, Wehenkel L (2005) Tree-based batch mode reinforcement learning. *J Mach Learn Res* 6: 503–556.
10. Geramifard A, Dann C, Klein RH, et al. (2015) RLPy: A Value-Function-Based Reinforcement Learning Framework for Education and Research. *J Mach Learn Res* 16: 1573–1578.
11. Lample G, Chaplot DS (2017) Playing FPS Games with Deep Reinforcement Learning. *AAAI* 2140–2146. https://doi.org/10.48550/arXiv.1609.05521
12. Littman ML (1994) Markov games as a framework for multi-agent reinforcement learning. *Proc Eleventh Int Conf Mach Learn*, 157–163. https://doi.org/10.1016/b978-1-55860-335-6.50027-1
13. Nagabandi A, Kahn G, Fearing RS, et al. (2018) Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. *ICRA*, 7559–7566. https://doi.org/10.1109/ICRA.2018.8463189
14. Ribeiro C, Szepesvári C (1996) Q-learning combined with spreading: Convergence and results. *Proc ISRF-IEE Int Conf Intell Cognit Syst*, 32–36.
15. Rücker C, Günther T, Wagner FM (2017) pyGIMLi: an open-source library for modelling and inversion in geophysics. *Comput Geosci* 109: 106–123. https://doi.org/10.1016/j.cageo.2017.07.011

Web links

pyGIMLi examples data repository: https://github.com/gimli-org/example-data/blob/master/traveltime/koenigsee.sgt.