Reinforcement learning for intelligent environments: A Tutorial

Zoltan Nagy, June Y. Park and Jose R. Vazquez-Canteli

Intelligent Environments Laboratory, Department of Civil, Architectural and Environmental Engineering, The University of Texas at Austin, Austin, TX, USA

ARTICLE HISTORY
Compiled January 9, 2020

ABSTRACT
There is no clear definition of what a smart building constitutes, i.e., what makes a building smart. In fact, since its inception in the early 1980, the term has had many different notions and meanings, largely depending on the area of study. In this chapter, we employ the approach that a smart or intelligent building, or, more generally, environment, is one that allows it to interact with its occupants. Based on this assumption we put forward that idea that one particular type of machine learning algorithms, namely reinforcement learning is particularly suitable for implementation and application in the built environment for two reasons: 1) it is a model-free approach requiring no prior knowledge of the system dynamics, and 2) it is inherently based on the notion of interaction, which happens to occur frequently, especially between occupants and building systems. Therefore, in this chapter, we provide an introduction to reinforcement learning in a tutorial form. We discuss the most important variations, the implications of particular parameter choices, and we provide typical algorithms. Finally, we discuss some challenges and opportunities of applying reinforcement learning in the built environment.

1. Introduction

The built environment accounts for 30% of global final energy consumption and greenhouse gas emissions. At the same time, buildings have a 50-90% emission reduction potential using existing technologies and their widespread implementation ([1]). Some of these technologies, such as cost-effective monitoring and data acquisition equipment, were made possible by recent developments in information and communication technologies (ICT). ([2])

The promise is that monitoring and analysis of building data with high spatiotemporal resolution allows building owners and building managers to a) understand their consumption patterns, and as a result b) make informed decisions about reducing the energy consumption of their building. Whether this is through active measures, i.e., updating the operation of their building systems (heating, ventilation, air conditioning, and lighting), or passive measures, i.e., upgrading the building envelope or fenestration, analyzing the monitored data is time-consuming, and typically requires specialized knowledge. Therefore, machine learning (ML) has been proposed to automate and inform the decision making process. ([3, 4, 5, 6, 7]), culminating in the idea of smart buildings and cities. Machine learning can be considered as the

> field of study that gives computers the ability to learn without being explicitly programmed.
> —(attributed to Arthur Samuel, 1959)

Or, more formally

> A computer program is said to learn from experience E with respect to some class of tasks and performance measure P if its performance at tasks in T, as measured by P, improves with experience E.— ([8])

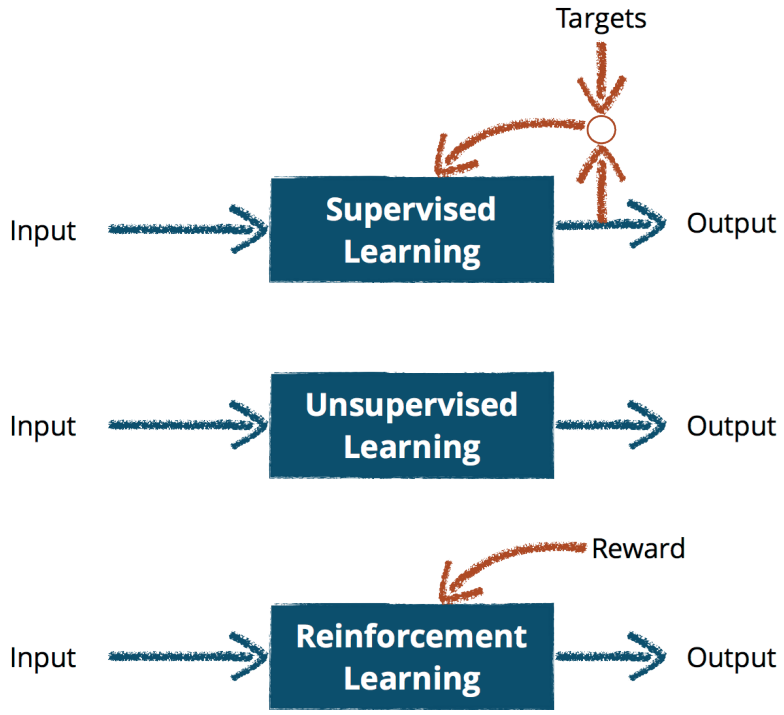

Figure 1. Overview of the three machine learning techniques (modified from [10])

As illustrated in Figure 1, ML techniques can be generally classified into three domains ([9]). In supervised learning, the learning agent receives a labelled dataset $\{x_1, x_2, \ldots, x_n, y_1, \ldots, y_m\}$, where x_i are called features, predictors, covariates, inputs or independent variables, and the y_j are called targets, outputs, or dependent variables. Typically, the goal of the agent is to find a suitable mapping that will predict the output for an unknown feature vector. When the targets are numeric, the learning task is regression, while for categorical targets the agent performs classification. Examples include linear or logistic regression, support vector machines, or nearest neighbour classifiers.

In unsupervised learning, the agent receives an unlabelled dataset $\{x_1, x_2, \ldots, x_n\}$, which only contains features x_i. Its goal is then to discover *interesting patterns* in the data. This is typically achieved through grouping of the feature vectors into similar clusters. Examples include k-means clustering and self-organizing maps.

Reinforcement learning (RL) is the third, machine learning technique in which the learning agent learns the optimal set of actions through interaction with its environment. In contrast to supervised learning, the agent does not receive large amounts of labelled data to learn from. In contrast to unsupervised learning, the agent receives a delayed feedback from the environment. In brief, for a given input, the agent chooses to perform a certain action. It then observes an immediate or delayed reward signal from the environment, and uses it to modify its knowledge on which action is best to choose under given circumstances. Because the agent acts in and learns from its environment, RL can be considered at the intersection between machine learning, and control theory ([11]), and has elements of game theory.

This interactive aspect is a key feature for the development of intelligent environments for two reasons. For one, it allows the agent to operate, and make decisions without the need for a predefined model. This is particularly interesting for large-scale, complex systems, such as buildings, or groups of buildings, where it is not cost-effective to develop such a model ([12]). Second, the learning agent interacts with its environment (and receives feedback from it), and part of this environment can be a person. Thus, RL provides a natural interface for occupants and facility managers to interact with the learning agent. Conversely, the learning agent can learn directly from the feedback provided by people.

The objective of this chapter is to review and discuss reinforcement learning in the context of

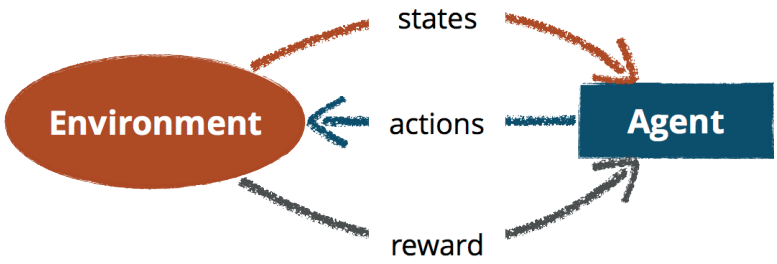

Figure 2. Basic structure of Reinforcement Learning ([13])

the built environment to motivate further research in this direction. In Section 2, we introduce reinforcement learning, and highlight the most typical variants of the algorithm. In Section 3, we discuss the opportunities for RL and show application examples from building energy and environments, while Section 4 discusses the challenges of implementation and algorithm development. Finally, Section 5 concludes the chapter.

2. Reinforcement Learning

In this Section, we review the basics of reinforcement learning (RL). For a complete introduction, we refer the reader to standard textbooks ([13]). Figure 2 illustrates the basic principle of RL. It can be formalized using a Markov Decision Process (MDP). An MDP is a tuple (S, A, P, R), where S is a set of states, A is a set of actions, P are the transition probabilities for taking an action $a \in A$ in a state $s \in S$, and $R : S \times A \mapsto \mathbb{R}$ is a reward function. The Markovian property refers to the fact that the probability of receiving a particular reward or transitioning from one state to another does not depend on the previous states or actions of the agent, but only on the current ones. The policy $\pi : S \mapsto A$ of the agent maps states to actions, and the value $V^\pi(s)$ of a state is the expected return for the agent when starting in that state and following the policy:

$$\begin{aligned} V^\pi(s) &= E_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r(s_k, \pi(s_k)) \;\middle|\; s_0 = s \right\} \\ &= r(s, \pi(s)) + \gamma \sum_{s' \in S} P(s, \pi(s), s') \cdot V^\pi(s') \end{aligned} \tag{1}$$

where $r \in R$ is the reward received for taking the action, and $\gamma \in [0, 1]$ is a discount factor. As seen in (1), γ allows it to balance between an agent that considers only immediate rewards ($\gamma = 0$) and one that strives towards long term rewards ($\gamma \to 1$).

The goal of the learning agent is to determine the optimal policy, π^*, i.e., the policy that leads to the highest expected return. The approach for this, i.e., for solving the MDP, depends on whether or not the probability transitions P and the reward function R, i.e., the dynamics of the system or model are known. This is shown in Fig. 3 and discussed in the next two Subsections.

2.1. *Model-based Learning*

If P and R are known, an optimal, or close to optimal, solution can be found through iterative methods, e.g., *value* or *policy iteration* as follows. First, for a given policy π, define the optimal value V^* of a state as ([14])

$$V^*(s) = \max_\pi E \left\{ \sum_{k=0}^{\infty} \gamma^k r(s_k, \pi(s_k)) \right\}. \tag{2}$$

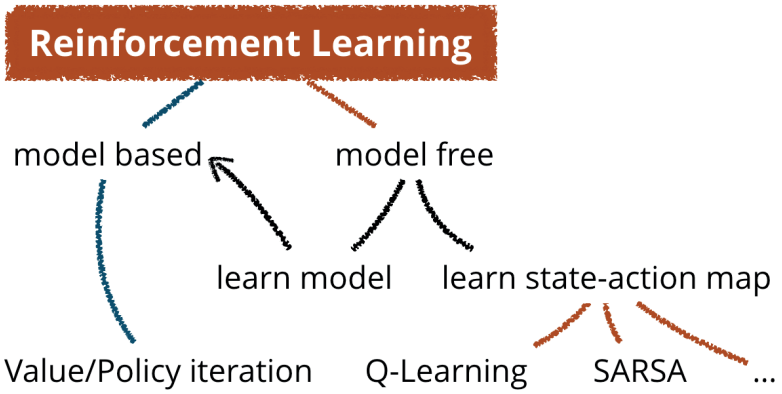

Figure 3. Types of Reinforcement Learning

Algorithm 1: Value Iteration

Initialize $V(s)$ arbitrarily
repeat
  foreach $s \in S$ **do**
    foreach $a \in A$ **do**
      $Q(s,a) \leftarrow r(s,a) + \gamma \sum_{s' \in S} P(s,a,s')V(s')$
    end
    $V(s) \leftarrow \max_a Q(s,a)$
  end
until *policy good enough*

It can be shown that $V^*(s)$ is unique and the solution of the following iterative equation

$$V^*(s) = \max_a \left\{ r(s,a) + \gamma \sum_{s' \in S} P(s,a,s')V^*(s') \right\}, \tag{3}$$

and once the optimal value function, V^*, is found, the optimal policy can be determined as

$$\pi^*(s) = \arg\max_a \left\{ r(s,a) + \gamma \sum_{s' \in S} P(s,a,s')V^*(s') \right\}. \tag{4}$$

This is called *value* iteration, and shown in Algorithm 1. An alternative, called *policy* iteration, is shown in Algorithm 2, where instead of iterating over the value function, the policy is improved iteratively until convergence, and the resulting optimal value function is determined.

Algorithm 2: Policy Iteration

Initialize to random policy π'
repeat
  $\pi \leftarrow \pi'$
  compute the value function of policy π
    $V^\pi(s) = r(s,\pi(s)) + \gamma \sum_{s' \in S} P(s,\pi(s),s') \cdot V^\pi(s')$
  improve policy:
    $\pi'(s) \leftarrow \arg\max_a \{ r(s,a) + \gamma \sum_{s' \in S} P(s,a,s')V^\pi(s') \}$
until $\pi = \pi'$

Algorithm 3: Tabular Q-Learning

Initialize $Q(s, a)$ arbitrarily
Initialize s
Define α, γ
while *true* **do**
  Choose a from s using policy (e.g. ϵ-greedy, softmax)
  Take action a, observe r, and s'
  $Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma \max_{a' \in A(s')} Q(s', a') - Q(s, a) \right]$
  $s \leftarrow s'$
end

2.2. *Model-free Learning*

If the model dynamics are not known, one can distinguish between two approaches (see Fig. 3). First, in the model-based approach, the probability transitions are first learned by observing the system, and then used in a planning procedure as mentioned above. Second, in the model-free approach, the agent learns to associate the optimal action for each state without explicitly determining transition probabilities between the states or the value of the states.

Q-Learning is the most widely used model-free reinforcement learning technique due to its simplicity ([15]). In simple tasks with small finite state sets, and discrete actions, all transitions can be represented using a table, hence the name *Tabular Q-Learning*, which stores the state-action values, i.e., Q-values (See Algorithm 3). Then, after taking an action a, given a state s, and observing the reward r, learning is achieved through updating $Q(s, a)$ as

$$Q(s, a) \leftarrow Q(s, a) + \alpha \big[r + \gamma \max_{a' \in A(s')} Q(s', a') - Q(s, a) \big] \tag{5}$$

where $\alpha \in [0, 1]$ is the learning rate, which explicitly defines to what degree new knowledge overrides old knowledge: for $\alpha = 0$, no learning happens, while for $\alpha = 1$, all prior knowledge is lost. Notice that in contrast to the model-based learning, where the value of a state was determined, here, the $Q(s, a)$-value represents the value of taking a specific action a when being in a certain state s. The relationship between model based value iteration, and model free Q-learning is that

$$V^*(s) = \max_{a' \in A} Q^*(s, a), \tag{6}$$

and as a consequence

$$\pi^*(s) = \arg\max_{a' \in A} Q^*(s, a). \tag{7}$$

In other words, the optimal policy, π^*, results from taking those actions a that maximize the respective Q-values in each state. In order for the algorithm (e.g., algorithm 3) to converge to the optimal policy, the requirement is that each state-action pair (s, a) be visited infinitely many times, such that the Q-values have converged. In the following, we discuss important features and extensions of Q-learning.

Action Selection In Q-learning, the process of accumulating knowledge happens through the trade-off between exploiting known, high-reward, actions, and exploring other, unknown, actions that have not been executed yet under that state. Two approaches are commonly used for this action selection. The first, called ϵ-greedy, selects a random action with probability ϵ (exploration), and the action with the highest expected return with probability $1 - \epsilon$ (exploitation). This balancing allows the agent to avoid local minima (exploration), while striving towards convergence (exploitation). In practice, ϵ is set relatively large in the beginning of the learning process, and then reduced progressively. The choice of the

initial value and the reduction strategy is domain specific and task of the designer.

Another method for action selection is Boltzmann exploration, which uses the softmax function to transform the Q-values into probabilities as

$$p(a) = \frac{\exp\{Q(a)/\tau\}}{\sum_{a' \in A} \exp\{Q(a')/\tau\}} \tag{8}$$

where the parameter τ can be interpreted as an artificial temperature that controls exploration and exploitation. In the beginning, τ is initialized to a large value, such that all actions are equally probable ($p(a) = 1/|A|$), and efficient exploration can happen. Over time, τ is reduced (or annealed), and for $\tau \to 0$, the action with the highest value will have probability 1 (exploitation). In other words, for $\tau \to 0$, Boltzmann exploration is reduced to ϵ-greedy. The advantage over ϵ-greedy is that instead of taking the current best (greedy) or a completely random action (as in ϵ-greedy), the softmax approach selects the action probabilistically, such that the probability of taking actions with larger Q-values is higher.

Curse of Dimensionality Tabular Q-Learning is affected by the curse of dimensionality: as the size of the state space increases due to, e.g., continuous sensor inputs, the size of the Q-table has to necessarily increase is well. In particular for building control, the curse of dimensionality is significant, considering the potentially large number of sensors measuring various quantities (temperature, humidity, energy consumption, illumination, etc.) continuously. This means that the agent has an exponentially increasing number of state-action pairs to explore before it can converge to an optimal solution. Function approximators, e.g., linear regression or artificial neural networks ([16]), have been proposed as solutions that allow generalization by directly mapping the state-action pairs, (s, a), to their respective Q-value, $Q(s, a)$.

Two neural network architectures are possible, both assuming a discrete action space (see Fig 4). In the first, Fig. 4(a), the inputs to the network are continuous sensory data (x_i) and one action, and its output is the expected Q-value of the action. In the second version, Fig. 4(b), only the states serve as inputs, and the output of the network are all the Q-values of all the possible actions. For large action spaces, the first version is more favorable as it allows to compute the Q-values of any possible continuous actions and states. On the other hand, the second version allows to compute the Q-values of any continuous states, but only of as many discrete actions as number of neurons in the output layer.

At the action-selection stage, it is necessary to compute the Q-values of many possible actions under a given state. This can be done by testing different possible combinations of actions in the neural network represented in Fig. 4(a). For example, if a greedy action-selection is to be followed, an optimization algorithm can be implemented to find the action that, when applied as an input into neural network, maximizes the function of Q-values. This allows to find a continuous optimal action.

Batch Reinforcement Learning Q-learning using a neural network is performed through updating the weights in the network, typically through the back-propagation algorithm ([16]). To avoid oscillation of the weights, and achieve good learning behavior of the network, many, dissimilar examples have to be presented to the network in one setting, rather than individual samples. This means, that weight updates of the network should not be performed after each state transition, but rather after many, e.g., m transitions. From another perspective, notice how in Algorithm 3, only the Q-values of the previous time-step are used to update the policy, i.e, previous state transitions or experiences of the agent are not retained. However, faster learning could be achieved by retaining not only the last experience, but many experiences, and use those as well in the policy update process. The set of these experiences is called a *batch*, and, thus, this approach is known as *Batch Q-Learning* ([17]), and shown in Algorithm 4.

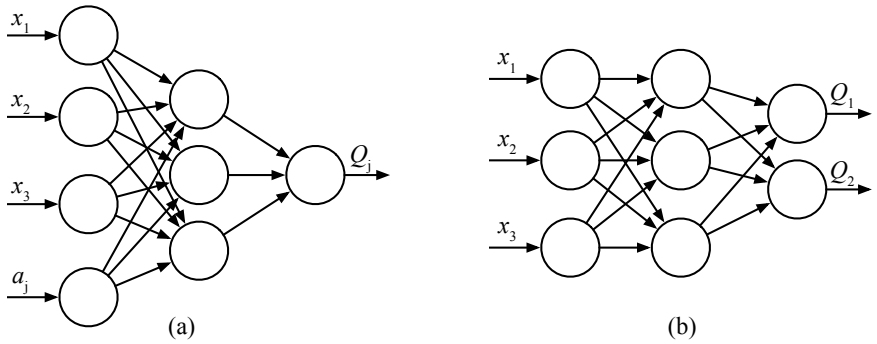

Figure 4. Two types of artificial neural networks can be used to map state-action pairs to Q values. Example for three states and two actions: (a) Input: 3 measurements (x_1, x_2, x_3) and 1 action (a_j). Output: $Q_j = Q(x_1, x_2, x_3, a_j)$. (b) Input: 3 measurements. Output $Q_1 = Q(x_1, x_2, x_3, a_1)$ and $Q_2 = Q(x_1, x_2, x_3, a_2)$.

Algorithm 4: Batch Q-Learning

```
while true do
    Initialize experience set D
    episodes ← 0
    repeat
        if new episode then
            episodes ← episodes + 1
        end
        Choose a from s using policy (e.g. ε-greedy)
        Take action a, observe r, and s'
        Add transition (s, a, r, s') in D
    until episodes=m
    Q ← UpdatePolicy(D)
end
```

The policy update function $UpdatePolicy(\mathcal{D})$ used in Algorithm 4, can be implemented in different ways. Two of the more prominent approaches are *Experience/Memory Replay* ([18]) and *Fitted Q-Iteration* ([19]). Experience replay is shown in Algorithm 5 ([17]), and essentially consists of applying the Q-value update of (5) repeatedly, k times, for all the experiences saved in $\mathcal{D}$. In contrast, Fitted Q Iteration approximates the Q-values successively in a supervised learning approach as shown in Algorithm 6.

Memory Replay Finally, since in many real-life applications the data contain transitions that occur much more frequently than others, e.g., the diurnal variations of building energy demand, the network should be only presented with the most representative samples in order to avoid biasing the learning process towards the most frequent patterns. This is achieved by extending batch Q-learning with previous experience to a technique called *Batch Q-Learning with Memory Replay* (see Algorithm 7 with $\gamma = 0$). In brief, all transitions (s, a, r) experienced by the agent are stored in a data-set $\mathcal{D}$. Then, during learning,

Algorithm 5: $UpdatePolicy(\mathcal{D})$: Experience/Memory Replay

```
Initialize Q-values
for iteration=1 to k do
    foreach (s, a, r, s') ∈ D do
        Q(s, a) ← Q(s, a) + α [r + γ max_{a'∈A(s')} Q(s', a') − Q(s, a)]
    end
end
```

Algorithm 6: $UpdatePolicy(\mathcal{D})$: Fitted Q-Iteration

for *epoch*=1 to E **do**
  foreach $(s, a, r, s') \in \mathcal{D}$ **do**
    $T_i \leftarrow r_i + \gamma_a \max_a Q_{epoch-1}(s_{i+1}, a)$
  end
  $Q_{epoch} \leftarrow Q_0$
  for *iteration*=1 to k **do**
    foreach $(s, a, r, s') \in \mathcal{D}$ **do**
      $Q_{epoch}(s_i, a_i) \leftarrow Q_{epoch}(s_i, a_i) + \alpha_{supervised}(T_i - T_{epoch}(s_i, a_i))$
    end
  end
end

Algorithm 7: Batch Q-Learning with Memory Replay

Initialize experience set $\mathcal{D}$
Initialize neural network (Q-values) with random weights
Initialize s
Define γ
while *true* **do**
  Choose a from s using policy (e.g. ϵ-greedy)
  Take action a, observe r, and s'
  Add transition (s, a, r, s') in $\mathcal{D}$
  Sample random batch of size N from $\mathcal{D}$
  Set neural network targets to
    $y_i = r_i$
end

a random batch is drawn from $\mathcal{D}$ and used to train the neural network. This algorithm has been successfully applied for building systems ([20]), and is the basis of the recent Deep Learning breakthroughs in Artificial Intelligence ([21, 22, 23]).

3. Opportunities for Smart Buildings and Cities

Consider the following two examples that are particularly interesting to transfer to the built environment. In the first example, Ng et al., over a series of papers have taught a helicopter to fly maneuvers autonomously using model-based RL (Coates et al. 2008; Ng et al. 2004). What is interesting is that no dynamic model has been used to design an optimal controller. Rather, the helicopter was flown manually by a pilot, and the recorded data (pilot commands, and resulting movement of the actuators) have been used to build a probabilistic model of the state transitions, which then have been used to solve the MDP.

In the second example, Knox and Stone present the TAMER (Training an Agent Manually via Evaluative Reinforcement) framework, in which the reward signal is explicitly provided by a human ([24, 25]). They demonstrate its effectiveness by teaching an agent to learn the computer game Tetris solely by having a human provide it with feedback on individual moves (*good move*, *bad move*, *indifferent*) rather than preprogramming that the objective of Tetris is to complete lines. Thus, the learning agent does not need to know the objective explicitly, it is able to learn it from its interaction with the human.

These two examples illustrate that is possible to a) learn complex dynamics and use them to derive optimal controllers, and b) learn directly from simple human feedback. These two characteristics are interesting because they occur in building control on a daily basis. Buildings, especially commercial buildings, are inherently complex systems with an increasing number of

sensors and actuators, resulting in a large number of states. Accurately modeling the energetic behavior and designing control systems for it is inherently complex and requires significant engineering efforts, which increases the costs.

Similarly, human-building interaction occurs on a daily basis, by way of how occupants use the building (occupancy, lights, blinds, windows, thermostats, etc), and have a large influence on the energy consumption of the building ([26]). In both cases, reinforcement learning is inherently suitable to offer an automatic, adaptive approach to increase energy efficiency and occupant comfort.

As an example, consider thermal comfort. Although models exists to understand the environmental conditions under which occupants will generally feel comfortable ([27, 28]), to this date it very challenging to predict the acceptable/comfortable environment for one particular person. Yet, the person typically knows whether or not he/she feels comfortable, and could give simple feedback ("I {am/am not} comfortable now"). In the formalized interactive approach of RL, this feedback becomes a reinforcement signal, which, together with environmental measurements, can be used by the agent to modify its knowledge of the environment, with the goal of potentially avoiding these situations in the future. Recently, Cheng et al. developed a satisfaction Q-learning algorithm for a lighting and blind control system. Using a graphical user interface to gather human comfort, their laboratory experiment demonstrated that Q-learning provided a more acceptable visual environment and energy saving control strategies. ([29])

For pure energy optimization, De Somer et al. used reinforcement learning to increase the self-consumption of the PV production in six residential buildings by storing energy in domestic hot water (DHW) buffers. They almost tripled the solar energy captured by the domestic hot water buffers compared to the default thermostat controller, and increased the total self-consumption of the PV production by 25%. ([30]) Similarly, Kazmi and D'Oca used reinforcement learning to improve the energy efficiency of an air-source heat pump and a DHW storage vessel. They carried out simulations using data from 40 houses, and tested their controller in an actual house, achieving 27% energy savings. ([31])

At the urban scale, it is important not only to balance the energy efficiency and comfort in individual buildings, but also to coordinate buildings amongst each other. Occupants tend to follow similar energy consumption patterns (e.g. people usually arrive home at similar times and create a peak in the electrical consumption), and buildings also tend to be more efficient at certain hours (e.g. when the indoor temperature is closest to the outdoor temperature). A lack of coordination can cause many buildings to consume electricity at the same hours, which would result in an inefficient management of the overall energy resources. Reinforcement learning would not just allow to learn from occupants and building energy systems, but it would also allow buildings to learn from each other in a multi-agent approach. ([32])

Finally, notice that reinforcement learning is essentially the formalization of a trial-and-error approach, which is very human and can be expressed poetically as

> Ever tried. Ever failed. No matter. Try again. Fail again. Fail better.
> —(Samuel Becket in Westward Ho, 1983)

And this human touch may actually increase its acceptance with the occupants, because the *smart* building or thermostat is not some obscure black-box system working in the background, but rather it works on the very familiar principles of trial-and-error, very much how humans train some animals.

4. Challenges

Despite the many advantages of RL for application in the built environment, many challenges remain. As RL is a relatively new and emerging field, there are no off-the shelf implementations available that researchers could experiment with, in contrast to supervised or unsupervised

methods. The reason for this is also that the application of RL depends heavily on the design of the state-action spaces and the reward signal, which are domain and application specific.

Further, as mentioned above, building systems and human-building interaction are highly complex systems. This results in many potential learning agents, which naturally constitute a multi agent system (MAS). Multi-agent RL has been recognized as the most suitable approach to tackle large scale complex real-world problems. However, the theoretical field is still in its infancy, and most available results are on stability and learning convergence for two agents. One of the main challenges here is the fact that the presence of multiple agents violates the stationary environment underlying most single agents learning approaches ([33, 34]). Here, research from the built environment can define clear goals and metrics by which the learning agents are to be evaluated to guide the theoretical developments.

One major challenge for reinforcement learning approaches in general is the relatively long learning time compared to model based approaches. This is typically the case for systems starting with zero prior knowledge. In buildings, however, a substantial prior and expert knowledge exists that can be leveraged to initialize the controllers and to guide them efficiently. In addition, it has been shown that it is possible to expect positive learning results already after one year of learning, which is relatively short when compared to the lifespan of a building ([20])

Finally, the interaction between individual systems and between occupants and systems are the key to achieve true balance between occupant comfort/satisfaction and energy efficient operation ([35, 36]. Since human inputs are required for the algorithms to be successful, the user interface between the occupants and the building systems is key. This could be done either by using the existing infrastructure, in which the occupants keep using the systems as provided, but are limited to the installed hardware and data logging options in the building management system. The other option is to install and upgrade new hardware (voice recognition, motion control, smartphone apps). However, the installation of such novel approaches needs to keep in mind that occupants tend to use easy-to-access and familiar interfaces.([37, 38]). In other words, in addition to the technical development of the control system, the occupant's perception of it is also important.

5. Conclusion

In this chapter, we have introduced reinforcement learning (RL) algorithms. Starting with basic Q-learning, we showed how mapping techniques, such as neural networks can deal with the curse of dimensionality, and how batch reinforcement learning and memory replay can improve the learning process. We have also provided typical algorithms in pseudo-code. We showed examples for opportunities for RL implementations in the built environment, from human-building interaction, to building energy optimization, to urban scale energy use coordination.

We discussed the major challenges of applying RL, and showed that the generally slow learning times might pose a barrier for success. Furthermore, the interesting and important application of multiple agents interacting with and learning from each other, will requires significant progress in the theory of multi-agent systems.

The human, i.e., trial-and-error, nature of RL is interesting because the agent can learn from direct feedback or observation of a human. In addition, it becomes easier to communicate to the occupants of how the learning procedure in their smart building may happen. Such improved communication will lead to improved acceptance, and as a consequence to improved satisfaction with, and success of, the learning system. As reinforcement learning inherently supports interaction, it is vital that researchers of the built environment embrace these opportunities and include learning approaches in their work, modify them and give practical guidance. This article hopes to provide a humble contribution in bridging these disciplines.

References

[1] O. Lucon and D. Ürge-Vorsatz. Fifth Assessment Report, Mitigation of Climate Change. *Intergovernmental Panel on Climate Change*, pages 674–738, 2014.

[2] June Young Park and Zoltan Nagy. Comprehensive analysis of the relationship between thermal comfort and building control research - A data-driven literature review. *Renewable and Sustainable Energy Reviews*, 82P3:2664–2679, 2018.

[3] A. I. Dounis and C. Caraiscos. Advanced control systems engineering for energy and comfort management in a building environment-A review. *Renewable and Sustainable Energy Reviews*, 13(6-7):1246–1261, 2009.

[4] Moncef Krarti. An Overview of Artificial Intelligence-Based Methods for Building Energy Systems. *Journal of Solar Energy Engineering*, 125(3):331, 2003.

[5] T. Agami Reddy. *Applied Data Analysis and Modeling for Energy Engineers and Scientists*. Springer US, Boston, MA, 2011.

[6] Clayton Miller, Zoltán Nagy, and Arno Schlueter. Automated Daily Pattern Filtering of Measured Building Performance Data. *Automation in Construction*, 49(A):1–17, jan 2014.

[7] Clayton Miller, Zoltán Nagy, and Arno Schlueter. A review of unsupervised statistical learning and visual analytics techniques applied to performance analysis of non-residential buildings. *Renewable and Sustainable Energy Reviews*, 81:1365–1377, 2018.

[8] Tom M Mitchell. *Machine Learning*. McGraw-Hill Book Company, 1997.

[9] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach, 3rd edition*. Pearson, 2010.

[10] Shouyi Wang. Machine Learning Algorithms in Bipedal Robot Control. *Systems, Man, and ...*, 42(5):728–743, 2012.

[11] M I Jordan and T M Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–60, 2015.

[12] Samuel Prívara, Jiří Cigler, Zdeněk Váňa, Frauke Oldewurtel, Carina Sagerschnig, and Eva Žáčeková. Building modeling as a crucial part for building predictive control. *Energy and Buildings*, 56:8–22, 2013.

[13] R Sutton and A Barto. Reinforcement Learning: An Introduction, 1998.

[14] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[15] Christopher J C H Watkins. Q-Learning. *Machine Learning*, 8:279–292, 1992.

[16] Simon Haykin. *Neural Networks and Learning Machines*. Pearson Education, Upper Saddle River, New Jersey 07458, 3ed edition, 2009.

[17] Shivaram Kalyanakrishnan and Peter Stone. Batch reinforcement learning in a complex domain. In *6th Int'l Conf on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 650–657, Honolulu, Hawai, 2007.

[18] Long Ji Lin. Self-Improving Reactive Agents Based on Reinforcement Learning, Planning and Teaching. *Machine Learning*, 8(3):293–321, 1992.

[19] Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-Based Batch Mode Reinforcement Learning. *Journal of Machine Learning Research*, 6(1):503–556, 2005.

[20] Lei Yang, Zoltan Nagy, Philippe Goffin, and Arno Schlueter. Reinforcement learning for optimal control of low exergy buildings. *Applied Energy*, 156:577–586, 2015.

[21] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning. *arXiv preprint:1312.5602*, dec 2013.

[22] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[23] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, and Laurent Sifre. Article Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[24] W Bradley Knox and Peter Stone. Training a Tetris Agent via Interactive Shaping: A Demonstration of the TAMER Framework. In *9th Int'l Conf. on Autonomous Agents and Multiagent Systems*

(AAMAS 2010), pages 1767–1768, Toronto, 2010.

[25] W. Bradley Knox and Peter Stone. Framing reinforcement learning from human reward: Reward positivity, temporal discounting, episodicity, and performance. *Artificial Intelligence*, 225:24–50, 2015.

[26] H. Burak Gunay, William O'Brien, and Ian Beausoleil-Morrison. A critical review of observation studies, modeling, and simulation of adaptive occupant behaviors in offices. *Building and Environment*, 70(0):31–47, 2013.

[27] ASHRAE. ANSI/ASHRAE 55:2010 Thermal Environmental Conditions for Human Occupancy, 2012.

[28] P O Fanger. Thermal comfort - Analysis and applications in environmental engineering. *McGraw-Hill Book Company*, 1970.

[29] Zhijin Cheng, Qianchuan Zhao, Fulin Wang, Yi Jiang, Li Xia, and Jinlei Ding. Satisfaction based Q-learning for integrated lighting and blind control. *Energy and Buildings*, 127:43–55, sep 2016.

[30] Oscar De Somer, Ana Soares, Tristan Kuijpers, Koen Vossen, Koen Vanthournout, and Fred Spiessens. Using Reinforcement Learning for Demand Response of Domestic Hot Water Buffers: a Real-Life Demonstration. *CoRR*, pages 1–6, 2017.

[31] Hussain Kazmi and Simona D'Oca. Demonstrating model-based reinforcement learning for energy efficiency and demand response using hot water vessels in net-zero energy buildings. *IEEE PES Innovative Smart Grid Technologies Conference Europe*, 2017.

[32] José Vázquez-canteli, Jérôme Kämpf, and Zoltán Nagy. Balancing comfort and energy consumption of a heat pump using batch reinforcement learning with fitted Q-iteration. *Energy Procedia*, 122:415–420, 2017.

[33] Karl Tuyls and Gerhard Weiss. Multiagent Learning : and Prospects. *AI Magazine*, Fall 2012:41–52, 2012.

[34] Robert Babuska, Bart De Schutter, and Lucian Buşoniu. A Comprehensive Survey of Multiagent. *Ieee Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews*, 38(2):156–172, 2008.

[35] Michael C Mozer. The neural network house: An environment that adapts to its inhabitants. *American Association for Artificial Intelligence Spring Symposium on Intelligent Environments*, pages 110–114, 1998.

[36] Diane J. Cook. How Smart Is Your Home? *Science*, 335(6076):1579–1581, 2012.

[37] Seyed Amir Sadeghi, Panagiota Karava, Iason Konstantzos, and Athanasios Tzempelikos. Occupant interactions with shading and lighting systems using different control interfaces: A pilot field study. *Building and Environment*, 97:177–195, feb 2016.

[38] Michal Luria, Guy Hoffman, and Oren Zuckerman. Comparing Social Robot, Screen and Voice Interfaces for Smart-Home Control. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems - CHI '17*, pages 580–628, New York, New York, USA, 2017. ACM Press.