Deep Q-learning for the selection of optimal isocratic scouting runs in liquid chromatography

Kensert, Alexander; Collaerts, Gilles; Efthymiadis, Kyriakos; Desmet, Gert; Cabooter, Deirdre

Published in:
Journal of Chromatography A

DOI:
10.1016/j.chroma.2021.461900

Publication date:
2021

Deep Q-learning for the selection of optimal isocratic scouting runs in liquid chromatography

Alexander Kensert[1], Gilles Collaerts[1], Kyriakos Efthymiadis[2], Gert Desmet[3] and Deirdre Cabooter[1,*]

[(1)]University of Leuven (KU Leuven), Department for Pharmaceutical and Pharmacological Sciences, Pharmaceutical Analysis, Herestraat 49, 3000 Leuven, Belgium

[(2)]Vrije Universiteit Brussel, Department of Computer Science, Artificial Intelligence Lab, Pleinlaan 9, 1050 Brussel, Belgium

[(3)]Vrije Universiteit Brussel, Department of Chemical Engineering, Pleinlaan 2, 1050 Brussel, Belgium

(*) corresponding author:

e-mail: deirdre.cabooter@kuleuven.be

tel.: (+) 32 (0)16.32.34.42

ABSTRACT

An important challenge in chromatography is the development of adequate separation methods. Accurate retention models can significantly simplify and expedite the development of adequate separation methods for complex mixtures. The purpose of this study was to introduce reinforcement learning to chromatographic method development, by training a double deep *Q*-learning algorithm to select optimal isocratic scouting runs to generate accurate retention models. These scouting runs were fit to the Neue-Kuss retention model, which was then used to predict retention factors both under isocratic and gradient conditions. The quality of these predictions was compared to experimental data points, by computing a mean relative percentage error (MRPE) between the predicted and actual retention factors. By providing the reinforcement learning algorithm with a reward whenever the scouting runs led to accurate retention models and a penalty when the analysis time of a selected scouting run was too high (> 1h); it was hypothesized that the reinforcement learning algorithm should by time learn to select good scouting runs for compounds displaying a variety of characteristics. The reinforcement learning algorithm developed in this work was first trained on simulated data, and then evaluated on experimental data for 57 small molecules – each run at 10 different fractions of organic modifier (0.05 to 0.90) and four different linear gradients. The results showed that the MRPE of these retention models (3.77% for isocratic runs and 1.93% for gradient runs), mostly obtained via 3 isocratic scouting runs for each compound, were comparable in performance to retention models obtained by fitting the Neue-Kuss model to all (10) available isocratic datapoints (3.26% for isocratic runs and 4.97% for gradient runs) and retention models obtained via a "chromatographer's selection" of three scouting runs (3.86% for isocratic runs and 6.66% for gradient runs). It was therefore concluded that the reinforcement learning algorithm learned to select optimal scouting runs for retention modeling, by selecting 3 (out of 10) isocratic scouting runs per compound, that were informative enough to successfully capture the retention behavior of each compound.

Keywords: method development; retention models; machine learning; reinforcement learning; deep q-learning;

1 INTRODUCTION

One of the main challenges in (liquid) chromatography is to obtain adequate methods to separate complex samples. Method development is still often done based on a trial-and-error approach, leading to lengthy optimization procedures. Decision-algorithms can be used to help find adequate separations for complex samples (1–8). These algorithms need to be able to predict future runs and select the correct runs to produce the most desirable outcome, which means accurate retention models need to be incorporated into the algorithms. Retention models can generally be divided into two types: quantitative structure retention relationship (QSRR) models, that model retention times of compounds for specific separation conditions based on their molecular descriptors (9–16) and empirical models, that model retention times of compounds based on experimental instrument or system variables, e.g. the fraction of strong eluent in the mobile phase (17–24). The two types of retention models are complementary and usually answer different questions. For instance, QSRR models are commonly developed to improve the identification of known analytes, while empirical retention models are rather developed to improve the separation of complex mixtures of known or unknown compounds. As the present study focuses on improving separation methods for compounds for which molecular structures are not always available, empirical retention models will be used.

An often employed empirical retention model is the Neue-Kuss model (21,26–28):

$$k(\varphi) = k_w(1 + S_2\varphi)^2 \exp\left(-\frac{S_1\varphi}{1+S_2\varphi}\right) \tag{1}$$

Wherein φ is the fraction of strong eluent in the mobile phase, k(φ) the retention factor for a specific φ and k_w, S_1 and S_2 are retention parameters that need to be fit to experimental retention factors to yield a retention model for a given compound. More specifically, k_w refers to the retention factor of a compound for φ=0 (in reversed-phase liquid chromatography (RPLC) this corresponds to a purely aqueous mobile phase), S_1 refers to the slope and S_2 to the curvature of the correlation between k and φ. Since the Neue-Kuss model requires three parameters for the prediction of the retention factor (k_w, S_1 *and* S_2), at least three datapoints (*i.e.*, experimental retention factors obtained at three different mobile phase compositions) are needed as input. In RPLC, the polarity of a compound is one of the main factors influencing its retention behaviour. Since compounds can have large differences in polarity, the selection of the scouting runs, *i.e.*, the selection of the mobile phase compositions that need to be selected to obtain adequate retention models, is an important problem as the most informative conditions can strongly differ for each compound. This importance is emphasized making the economic consideration that the number of scouting runs should preferably be kept as low as possible. Thus, there is a clear need to develop algorithms that can select optimal scouting runs leading to accurate retention models.

Decision algorithms for chromatographic applications, such as method development, have been studied since the 1980's. These algorithms, also sometimes referred to as *expert systems,* focused on different aspects of the chromatographic workflow: method selection (28,29), mobile phase selection (30,31), mobile phase/selectivity optimization (32,33), system optimization (34,35) and retention optimization (36–39). All these algorithms aimed at improving the decision-making process for method development and reducing the extent of trial-and-error that is commonly practiced. However, in the late 90's these expert systems lost their popularity due to the fact that the high expectations for these systems were not met, and the computational power available at that time was inadequate.

Traditional approaches, like those used in the 80's and 90's, usually take the form of predefined rules and if-else statements, which are directly programmed/handcrafted by the software engineer and experts in the field. The inherent problem of this approach is its great difficulty to generalize to new data or scale to large, diverse datasets. Algorithms like machine learning (40) can, to some extent, solve the problem of generalization by letting the machine program itself (learn its own rules) via highly parameterized and regularized function approximators that are trained on a so-called training dataset. Supervised machine learning algorithms have been around for many decades (41–44) but came to proliferate in the late 2000's when the availability of data in both private and public repositories grew considerably and sufficiently powerful computer hardware became readily available. This family of algorithms only needs to be supplied with input and output pairs, and everything in-between (mapping from input to output) is learned by the algorithm itself. Given an input, *supervised learning* algorithms learn how to minimize the error between their prediction and the target (the associated output for that input). A typical application of supervised learning is classification, in which, given an input, the algorithm outputs the correct class. However, the limitation of supervised learning is the need to label every input, which is often either very time-consuming or impossible to actualize. *Unsupervised learning* deals with situations where no labels are required, with a typical application being clustering. Instead of directly getting a prediction from some input, it clusters the input data and finds patterns in it. Either of these two paradigms is important, but neither fulfills the criteria for intelligent decision making as required in method development. In method development, especially when dealing with novel or unknown compounds, a prediction of the next step to perform is needed, but the effects of the steps are unknown and impossible to plan.

Reinforcement learning is a machine learning paradigm that deals with sequential decision making (45,46). In contrast to supervised learning, where each prediction is *independent* of the others, decisions are sequential and *dependent* on each other. This aligns well with method development, which is clearly a sequential procedure (cfr. the trial-and-error approach when method development is carried out in its most simple form). Furthermore, in reinforcement learning, there is no need for input-output pairs as in supervised learning, because no label is needed to teach the algorithm to perform well on a given task (which is of great importance as labels are commonly unavailable in method development). Instead of a label, a scalar reward signal reinforces a desirable behavior; *i.e.* the reinforcement learning algorithm learns a mapping from some input to some desirable output by receiving feedback based on the output. The goal of the reinforcement learning algorithm is not to optimize itself towards accurate label predictions, but instead maximize the cumulative future reward (*i.e.* learn to optimally perform a specific task). Due to its generality, reinforcement learning algorithms have been widely utilized and studied in many applications and areas of research and industry, including robotics (44–49). Besides its many applications, reinforcement learning is possibly most known for its achievements in attaining superhuman level performance in board games, such as Chess and Go (53–55), and in video and computer games such as Atari (56), Starcraft II (57,58) and Dota 2 (59).

In the present study, the possibilities of reinforcement learning are investigated for a basic chromatographic problem: selecting optimal isocratic scouting runs in RPLC to fit retention models. Specifically, the aim is to train a reinforcement learning algorithm to select a minimum number of scouting runs that allow to obtain accurate retention models for a diverse set of compounds. The present study is considered a simple first step in introducing reinforcement learning to the area of chromatography.

2 THEORY

Reinforcement Learning is a goal directed paradigm in which a computer program, called the *agent,* is continuously interacting with the *environment*. The environment is the world in which the agent resides and encapsulates the task the agent tries to solve. By performing actions, the agent receives a numerical feedback from the environment based on the quality of the chosen action. Over time, the agent learns to optimize the actions it takes by trying to maximize the future rewards. Typically, reinforcement learning uses a Markov Decision Process (MDP) as its mathematical model. An MDP is a 4-tuple <S, A, T, R>, where S is the state space, A the action space, $T(s, a, s') = \Pr(s' \mid s, a)$ the probability that action a in state s will lead to state s', and $R(s, a, s')$ the immediate reward (r) received when action (a) taken in state (s) results in a transition to state (s'). The problem of solving an MDP is to find a policy (π) (i.e., mapping from states to actions) that maximizes the accumulated reward (i.e., the sum of all scalar reward signals). A typical interaction goes as follows: given a state s, the agent takes an action a, which causes the environment to change its state to a new state s'. In addition to the new state s', the environment provides a specific reward signal, indicating how good it is in the new state. Moving across states by performing an action is known as a *transition*, and a sequence of transitions is called a *trajectory*. Specifically, in this study, a trajectory (also referred to as an episode) is defined as a sequence of transitions and ends with a *termination* signal indicating that no more scouting runs are to be run for a given compound.

The state space and action space of the environment define the space in which the reinforcement learning algorithm (or the agent) can perform intelligently. The state of the environment can be represented in many ways, everything from video/image input (a continuous state space), to a vector containing the positions of marks on a tic-tac-toe board (a discrete state space). Similarly, the action space may also be continuous and/or discrete. The environment also sets restrictions for what actions are legal – for instance, is the agent allowed to adjust the percentage of modifier to any percentage? Or is it only allowed to change it between certain discrete, predefined, ranges?

To formalize the learning procedure, the agent tries to learn a policy – what action to take given a certain state. The ultimate goal of the agent is to learn a policy π that maximizes the expected cumulative future reward (known as the expected return) over a trajectory τ. This optimal policy denoted as π^* is defined as:

$$\pi^* = \arg\max_{\pi} E_{\tau\sim\pi}[R(\tau)]\,, \quad (2)$$

$$\text{where } R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t \quad (3)$$

$E_{\tau\sim\pi}[R(\tau)]$ denotes the expected return of trajectory τ following policy π and $\arg\max_{\pi}$ specifies the policy maximizing the expected return. $\gamma \in (0,1)$ is known as the *discount factor* that discounts the reward r depending on how far off in the future the reward was obtained. If the discount factor is smaller than one, it is more valuable to get rewards early on rather than later.

For the agent to judge the quality of a given state in an environment, it learns its value through a *value function*. The value function denotes the desirability of being in a state at a given point in time. In the present study, the *optimal action-value function* Q^* will be used – which expresses the expected return if the agent starts in a state s, takes an action a, and then acts according to the optimal policy forever after:

$$Q^*(s,a) = \max_{\pi} E_{\tau\sim\pi}[R(\tau)|s,a] \tag{4}$$

$\max_\pi$ indicates the policy yielding the maximum expected return and $\tau\sim\pi$ indicates that the trajectory is sampled according to the policy. Importantly, Q^* can be rewritten as a recursive function, called the Bellman equation for Q^* (60):

$$Q^*(s,a) = E_{s'\sim P}\left[r(s,a,s') + \gamma \max_{a'} Q^*(s',a')\right] \tag{5}$$

$s'\sim P$ indicates that the next state is sampled according to the environment's transition rules, $r(s,a,s')$ is the reward received in the next state s' by taking action a in state s, and $\gamma \max_{a'} Q^*(s',a')$ is the discounted maximum expected return in the next state s'. The expected return Q^*, also referred to as the *Q*-value, of each state-action pair can be computed – allowing for the optimal action a^* in a given state to be obtained via:

$$a^*(s) = \arg\max_a Q^*(s,a) \tag{6}$$

$\arg\max_a$ specifies the action which maximizes $Q^*(s,a)$ and yields the highest *Q*-value.

The optimal action-value function (Q^*) is not known at the start and needs to be solved. A common approach to solve Q^* is via *Q-learning* (61). The *Q*-learning algorithm incorporates the Bellman equation for Q^* to iteratively update Q:

$$Q_{i+1}(s,a) = E_{s'}\left[r + \gamma \max_{a'} Q_i(s',a')\,|s,a\right] \tag{7}$$
where $Q_i \to Q^*$ when $i \to \infty$

Q is known as the *Q*-function, which approaches or represents Q^* at any given time. i indicates iteration i, or transition i. By experiencing states and actions, Q can iteratively be updated until it reaches Q^* or is close enough to it. The agent will select actions based on an action selection mechanism. A typical method for action selection is ϵ-*greedy* and is defined as:

$$a \leftarrow \begin{cases} \arg\max_a Q(s,a) & \text{with probability } 1-\epsilon \\ \text{random action} & \text{with probability } \epsilon \end{cases}$$

In essence, the agent will choose to perform a random action with probability ϵ, and will select the optimal action (based on what the agent knows about the environment so far) with a probability 1-ϵ. Through this continual interaction, the agent will eventually learn which actions to perform at each state.

However, a problem with this tabular approach is that it is impractical, because the state space and action space in most interesting problems are often too large. This approach may not converge to Q^* and lacks generalization capabilities (only able to yield accurate expected returns given actions and states that it has experienced previously). In reinforcement learning, generalization is typically solved by some form of function approximation by representing the value function as a linear function or as a neural network. A major breakthrough in reinforcement learning was the use of the representational power of deep neural networks instead of tabular value functions which led to the creation of Deep Q Networks (DQN) (56) and spurred an entire research field called Deep Reinforcement Learning.

Specifically, a function approximator, in this case a deep neural network, approximates Q^*: $Q_\theta \approx Q^*$. The parameterized action-value function Q_θ is thus a neural network, referred to as the *Q*-network, which receives as input a state *s* and outputs the expected return for each possible action in that state. The parameters θ of the *Q*-network are estimated by *minimizing* an *objective function* [with respect to θ], which aims to yield an accurate approximation of *Q**:

$$\min_{\theta_i}\left[\left(r(s_i, a_i, s_i') + \gamma \max_{a_i'} Q(s_i', a_i', \theta_i^-)\right) - Q(s_i, a_i, \theta_i)\right]^2 \tag{8}$$

$\min_{\theta_t}$ indicates that θ is being adjusted at each iteration (*i*) to minimize the objective function. The objective function is the mean squared error between the target (left term between brackets) and the prediction (right term between brackets). For stability purposes, the *target* network has its parameters θ^- held fixed when minimizing the objective function to not have a continually moving target. The method is referred to as *DQN*, and the minimization procedure referred to as *training* – which is an iterative procedure that iteratively tries to minimize the objective function over the course of thousands of episodes.

To better understand the concept of reinforcement learning from a higher level, and to tie it to the process of method development, let us consider the agent-environment interaction. This can be thought of as an analyst performing liquid chromatography (LC). If the environment is the LC with its internal physics and chemistry (for example the interactions of analytes with the stationary phase inside the column), then the agent would be the analyst, interacting with the LC. The analyst merely sees a representation of the interaction between a mixture of compounds and the mobile/stationary phase. This representation will take the form of a chromatogram (or retention times), which is the output of the LC. Based on this output, the analyst decides the next action, and as a consequence of this new action, a new representation of a state (new output) is presented to the analyst, and so on. These abstractions are important to understand the roles of the agent and the environment and how they interact, as well as the important differences between them.

3 MATERIALS AND METHODS

3.1 Chemicals

The compounds used in this study, together with their structure, Log P and pKa values, stock solvent and supplier, are mentioned in Table S-1 in the Supporting Information. Structures were drawn in Marvin Sketch (v20.9.0, 2020, ChemAxon (www.chemaxon.com)) and values were gathered from PubChem (62). The solvents used to prepare stock solutions and mobile phases were: acetonitrile (ACN, HPLC grade) purchased from Fisher Chemical (Loughborough, UK); formic acid (99%) purchased from Acros Organics (Geel, Belgium); and ammonium formate (≥ 99.995%) purchased from Sigma-Aldrich (Diegem, Belgium). Ultra-pure water (H_2O, conductivity = 0.1 µS/cm, pH 6.00) was produced in the laboratory using a Milli-Q gradient purification system from Millipore (Bedford, MA, USA).

3.2 Apparatus

All measurements were done on an Agilent Infinity 1290 system from Agilent Technologies (Waldbronn, Germany) that consisted of the following modules: a quaternary pump (G4204A), an autosampler (G4226A), a thermostatted column compartment (G1316C) and a diode array detector with a 1.0 µL flow cell (G4212A). OpenLab software (Agilent Technologies) was used to operate the system and acquire and analyse the data. Further data treatment was done in Microsoft Excel. The maximum pressure of the system was 1200 bar. The injection volume was kept constant at 0.50 µL and the flow rate was 0.500 mL/min. The detector was set at an acquisition rate of 40 Hz to measure the retention times of the compounds and at 80 Hz to measure the column void and extra-column times. The considered wavelengths were 240 nm, 254 nm and 275 nm. The column was placed in an oven with a constant temperature of 30.0 °C. Analyses were performed on an Xbridge C_8 column (2.1 x 50 mm; 2.5 µm) from Waters (Wexford, Ireland). Connections between the autosampler, column and detector were made with nanoViper tubing (inner diameter: 75 µm, total length: 1000 mm) from Thermo Scientific (Germering, Germany).

3.3 Stock and working solutions

Stock solutions of all compounds were prepared in pure organic (ACN) or aqueous solvent, depending on their solubility, as mentioned for each compound in Table S-1, in a concentration up to 10,000 µg/mL. Working solutions were made by diluting the stock solutions to 20 µg/mL in a final solvent mixture of 1.000 mL 5/95 ACN/H_2O.

3.4 Dataset

Retention factors were collected for 82 representative small molecules (see Table S-1), covering a wide range of physiochemical properties, at ten different isocratic strengths wherein the fraction of ACN (φ_{ACN}) was varied between 0.05 and 0.90 (in 0.10 intervals between 0.10 and 0.90). The aqueous component of the mobile phase was an ammonium formate buffer (brought to pH 3.00 with formic acid). Each compound was injected in triplicate and obtained retention times were averaged. The obtained retention times were converted into retention factors using the following equation:

$$k = \frac{t_r - t_0}{t_0 - t_e} \tag{9}$$

Wherein t_r is the retention time of a particular compound, t_0 is the elution time of an unretained marker (thiourea in this study) and t_e is the extra-column dead time, obtained by replacing the column in the system by a zero dead-volume union and injecting thiourea under the same experimental conditions as for t_r and t_0.

Some compounds in the dataset were so strongly retained at low φ_{ACN} (e.g., 0.05 and 0.10) that it became impractical to record their retention factors in a reasonable amount of time. Therefore, the retention factors at low φ_{ACN} for these compounds were estimated by fitting the Neue-Kuss model (Eq. 1) to the available retention factors, and then using the fitted Neue-Kuss model to determine the missing retention factors. Some other compounds were highly polar and did not retain adequately (defined as having a retention factor below 2.5 at 0.05 φ_{ACN} or having less than five retention factor values above 0.001) resulting in unreliable data. These compounds were therefore discarded in this study. The final remaining number of compounds was 57. Furthermore, all datapoints were clipped to have a lowest k-value of 0.001, which was necessary because inherent measurement errors caused low retention factor values to be unreliable, with fluctuations between small positive (<0.001) and small negative (> -0.001) values. All experimentally obtained retention factors are shown in Table S-2 in the supporting information. In addition to isocratic retention data, experimental gradient retention data were additionally collected for the 57 compounds. For this purpose, four different gradient profiles were experimentally run (see Table 1).

3.5 Compound simulator

To train the Q-network, a compound simulator was created to generate sufficient training data for the Q-learning algorithm. By fitting the Neue-Kuss model to all the experimentally obtained retention data for the 57 compounds described above, ranges of parameter values (S_1, S_2 and k_w) were deduced, and these ranges and their mutual relations laid the basis to generate retention parameters for simulated compounds (see Figure S-1 and S-2 in the Supporting Information). The simulator generated a simulated compound in five steps:

1. S_1 was randomly sampled between $10^{0.9}$ and $10^{1.8}$;
2. S_2 was sampled from its relationship with S_1: $S_2 = 2.5010 \cdot \log S_1 - 2.0822 + r_1$, where r_1 is a random number sampled from a uniform distribution $U \sim (-0.35, 0.35)$;
3. k_w was sampled from its relationship with S_1: $k_w = 10^{0.0839 \cdot S_1 + 0.5054 + r_2}$, where r_2 is a random number sampled from a uniform distribution $U \sim (-1.2, 1.2)$;
4. the resulting parameter values were input to the Neue-Kuss model to output ten datapoints (retention factors) between φ_{ACN}= 0.05 and 0.90;
5. random noise sampled from a normal distribution $N \sim (1.0, 0.1)$ was added to each datapoint via multiplication.

The last step was included to mimic the noisy nature of the experimental data. A total of 10,000 compounds were simulated according to the five-step procedure above; each of which would occupy a single episode in the complete training of the deep Q-learning algorithm. Although not directly evaluated, 10,000 simulated compounds were considered sufficient to capture the compound space for the specific HPLC setup.

3.6 Double deep *Q*-learning algorithm

3.6.1 Environment

The dynamics of the reinforcement learning environment aimed to mimic that of a real chromatographic workflow: sequentially selecting [isocratic] scouting runs for *one* compound at a time (see Figure 1).

For consistency, and to allow for evaluation on experimental data later on, the state and action space of the environment was restricted to the fractions of organic modifier (φ_{ACN}) that had been experimentally run (namely 0.05 and 0.10 to 0.90 in steps of 0.10; Table S-2). Specifically, a state was defined as an array of 10 elements, each of which was a placeholder for a retention factor value at a certain φ_{ACN} (e.g. the first element was a placeholder for the retention factor value at 0.05 φ_{ACN}, the second element was a placeholder for the retention factor value at 0.10 φ_{ACN}, and so on). The default value of the elements in the array when a given placeholder was not holding any retention factor value was set to -1, as is the case when no scouting runs had been run (see Fig. 1). The value was set to -1 and not 0 in an attempt to make the *Q*-network more easily distinguish between a non-retention factor value and a low retention factor value. Similar to the state space, the action space was defined to select from the different φ_{ACN} that had been run experimentally, with an addition of a *STOP* action, allowing the agent to stop performing scouting runs (see Fig. 1). At *each* step, the agent was given the choice to select *one* specific scouting run, or to stop.

To teach the agent to select desirable scouting runs that can be used to accurately model the retention behavior of a given compound, a reward function was defined. The reward function was defined in such a way as to teach, or *reinforce*, the agent to select optimal scouting runs. After the agent had finished performing scouting runs (and at least three different scouting runs had been selected and run), the resulting retention factors (from those three scouting runs) were used to fit the Neue-Kuss model (see appendix A for more information). Subsequently, the reward function computed a reward based on the inverse of the mean relative error (MRE^{-1}) between the target retention factors (y) and the retention factors predicted by the Neue-Kuss model ($\hat{y}$):

$$\mathrm{reward} = \mathrm{MRE}^{-1}, \tag{10}$$

$$\text{where MRE } = \frac{1}{10}\sum_{i=1}^{10}\frac{|y_i-\hat{y}_i|}{|y_i|} \tag{11}$$

The targets y are assumed to be the *true* retention factors which we want the predictions $\hat{y}$ to be as close as possible to. In the best-case scenario, true retention factors would be obtained from real experiments. However, as mentioned before, because *Q*-networks need a lot of training data, simulated compounds were used instead; and the relatively few (57) experimental compounds were held-out as a *test set*, and later used to evaluate the agent after it had been trained. Although the training data were randomly sampled using the simulator, which used ranges of Neue-Kuss parameter values (S_1, S_2 and k_w) that were based on those of the compounds in the held-out test set, these parameters were merely roughly deduced from the overall trends of the parameter values of the test compounds and should therefore not undermine the purpose of the held-out test set, which is to evaluate the agent on real 'unseen' data. Importantly, the MRE was always calculated based on all (10) retention factors (0.05 $\rightarrow$ 0.90 φ_{ACN}), as the purpose was to obtain models that predicted well for all (10) φ_{ACN}.

Furthermore, if the agent stopped before three scouting runs had been selected, the MRE evaluation (Eq. 10) was not carried out and the agent was rewarded zero (0). Finally, the agent was penalized (negatively rewarded) in two ways:

1. by selecting scouting runs that had already been selected before, penalized -5;
2. by selecting scouting runs that resulted in long analysis times (> 1 hour), penalized based on a sigmoidal (s-shaped) function f_s:

$$f_s(k) = \frac{1}{1-e^{-(k\cdot 0.00425-4.0)}} \times 20 \qquad (12)$$

where k is the retention factor. The function f_s is adapted in such a way as to have a maximum penalty close to the maximum reward that can be obtained (Eq. 10), and a minimum penalty at zero (0). These penalties were implemented to enable the agent to quickly learn not to select the same scouting run multiple times, and to avoid selecting scouting runs that would result in high analysis times, respectively. The analysis time penalty (Eq. 12) was important because the Q-network had to learn to tailor the choice of scouting runs depending on the retention behavior of the compound. For example, if the agent had to select scouting runs for a highly retained compound, it was expected to avoid running scouting runs at low φ_{ACN}.

3.6.2 Agent

In this study, as an extension to the deep Q-learning algorithm introduced in section 2, a *double deep Q-learning algorithm* (63) with *experience replay* (64,65) was implemented. Additionally, an *epsilon-greedy policy* (60) was incorporated for selecting actions (see appendix A, B and C for details on these implementations respectively).

Instead of having a single Q-network for both the target and the prediction together, the double deep Q-learning algorithm utilizes two separate neural networks with two separate Q-functions to approximate Q^*: a *target* network and a *prediction* network, for the target and the prediction respectively. The minimization procedure is based on the procedure of a single Q-network as described in (Eq 8). The target network has fixed parameters θ^- which are updated every N episodes by having the prediction network copying over its parameters θ, while the prediction network is updated every episode via the minimization procedure (Eq. 8).

It is of great importance to have a separate network to produce the target, because it counters the issue of overestimating and biasing the target. The overestimation (and bias) of the target naturally occurs for standard Q-learning and DQN due to the max operator (cfr. $\max_a$ in eq. 5 and 6) both selecting and evaluating the actions. Hasselt et al. have shown how Double deep Q-learning produces more accurate estimations of the expected return as well as better policies (63,66).

A crucial quality of the minimization procedure (Eq. 8) is that it can be formulated to accept diverse batches of non-correlated examples. Specifically, the minimization step can be done batch-wise, where each batch contains transitions from different trajectories. In other words, each batch may contain tuples (s,a,s') from entirely different scouting runs. To be able to sample such batches, a memory unit called *replay memory* was implemented to store transitions that were experienced by the agent while it was performing scouting runs. These stored transitions were scheduled to be sampled (in batches) after each episode, and to be directly used to train the agent (minimize the objective function). Although we did not test any algorithm without experience replay (replay memory), it is believed that this technique greatly stabilizes and smoothens the training process (64,65). The discount factor γ was set to 0.95 as an attempt to give slightly higher weight to early rewards.

3.7 Evaluation

After the double deep *Q*-learning agent was trained, it was evaluated on real, experimental data. Similar to how the agent selected scouting runs for simulated data during training, the agent selected scouting runs for isocratic experimental compounds. The resulting Neue-Kuss model (fitted on the retention factors resulting from the scouting runs selected by the agent) were then used to predict the retention factors for *all* 10 isocratic scouting runs. Additionally, the same Neue-Kuss parameters were also used in the Neue-Kuss equation for gradient elution, which was used to predict retention data under four different gradient conditions (Table 1). The model for gradient elution is defined as follows:

$$t_{r,gradient} = \frac{1}{S\beta}\frac{(1+S_2\varphi_0)^2 \ln(1+S_1\beta k_w \exp\left(\frac{-S_1\varphi_0}{1+S_2\varphi_0}\right)\left(t_0-\frac{t_D}{k_0}\right))}{1-S_2\left(\frac{1+S_2\varphi_0}{S_1}\right)\ln(1+S_1\beta k_w \exp\left(\frac{-S_1\varphi_0}{1+S_2\varphi_0}\right)\left(t_0-\frac{t_D}{k_0}\right))} + t_0 + t_D \qquad (13)$$

where S_1, S_2 and k_w are the same parameters as for the isocratic model (cfr Eq. 1), $t_{r,gradient}$ is the gradient retention time, φ_0 the fraction of strong eluting solvent in the mobile phase at the start of the gradient, β the slope of the gradient (= $\Delta\varphi/t_G$), k_0 the retention factor at φ_0, t_0 the elution time of an unretained compound and t_D the dwell time of the system (the time the mobile phase needs to flow from the pump to the head of the column). The predicted retention factors (for both isocratic and gradient runs) were then compared to *all* experimental data available for that compound by calculating the mean relative percentage error (MRPE) with modification, as follows:

$$\text{MRPE} = \frac{1}{10}\sum_{i=1}^{10}\frac{|y_i-\hat{y}_i|}{|1+y_i|} \times 100 \qquad (14)$$

The (modified) MRPE was used as it was less sensitive to small k-values (<0.1), creating a more stable and robust evaluation metric. To further illustrate how well the resulting model (based on the selected scouting runs) performed, a comparison was made to a model fit on all available isocratic datapoints (0.05 $\rightarrow$ 0.90 φ_{ACN}), a model fit on a random selection of three isocratic scouting runs, as well as a model fit on a chromatographer's selection of three scouting runs, namely φ_{ACN}= 0.1, 0.5 and 0.8.

4 RESULTS AND DISCUSSION

4.1 Learning curves: training progression

To illustrate how the agent learned to perform optimal scouting run selection, the *rewards* and *number of selected scouting runs* were collected during the training procedure. Figure 2 illustrates both how the reward and the number of selected actions changed over the first 3500 episodes. In addition, for comparison, the theoretical optimum (highest possible reward) was also calculated and plotted. Notice that the theoretical optimum does not reflect the best possible performance by an agent, but rather illustrates what in theory could be obtained. Specifically, the theoretical optimum was obtained by selecting, for each and every compound separately, the combination of (three) scouting runs (out of the 120 combinations that exist for 3 actions and 10 possible states) that resulted in the highest reward. Due to random action selection, leading to poor retention models and high penalties, the reward given to the agent started off at high negative values (around -5 to -15). Specifically, the low reward as well as the high number of selected scouting runs in the first 100 episodes confirmed that the selected actions were highly random (due to the high ϵ for the epsilon-greedy policy). Between episode 100 and 200 however, the agent started to select scouting runs non-randomly, which resulted in the agent selecting much fewer scouting runs (around 2 on average). Notice that, although not yielding a retention model due to too few scouting runs (< 3), the agent had at this point learned to select better actions than before, because no or fewer penalties were given. After 200 episodes, the agent went from an insufficient number of scouting runs (which resulted in a reward around 0) to selecting enough scouting runs for a retention model, resulting in a positive reward outweighing potential penalties. In other words, after 200 episodes, the agent started to learn how to optimize the selection of scouting runs for retention modelling – selecting around four scouting runs on average, which is on average one scouting run more than the minimum required to fit a three-parameter model. This number was overestimated due to the epsilon-greedy policy ($\epsilon > 0$), which is likely to cause the agent to select scouting runs sub-optimally (i.e. not selecting actions with the greatest *Q*-value). Notice, although not visualized, if only the greedy policy was followed (epsilon = 0), the agent would select closer to three scouting runs on average.

4.2 Scouting run selection

After the agent, or the *Q*-network, was optimized or trained to select scouting runs for simulated data, it was evaluated on experimental data. Figure 3 illustrates how the agent selected scouting runs for four representative compounds it had never seen before (see Table S-3 for complete results). For the four compounds considered in Figure 3, φ_{ACN} of 0.20 and 0.90 were selected by the trained agent; and for all four compounds, three scouting runs were run in total (which is the minimum requirement). What differed between compounds of different retention behavior, specifically between less retained compounds (e.g. nitrobenzene and o-methylacetophenone) and more retained compounds (e.g. iodobenzene and anthracene), was the selection of the third scouting run (φ_{ACN} of 0.05 for the less retained compounds and 0.50 for the more retained compounds), which indicates that the selection of scouting runs was tailored based on the retention behavior of the compounds. Because nitrobenzene and o-methylacetophenone are less retained compounds, they were both selected to be run at as low as φ_{ACN}=0.05. Intuitively, a scouting run at φ_{ACN}=0.05 holds important information for a retention model but requires significantly longer analysis time for highly apolar compounds, like iodobenzene and anthracene. For these highly retained compounds, φ_{ACN}=0.05was avoided and instead $\varphi_{ACN=}$0.50 was selected by the agent. Furthermore, in all cases, the scouting run selection spanned a wide range of selectable φ_{ACN}, in a relatively evenly spaced-out manner, which intuitively should result in more accurate retention models. Finally, these results illustrate that the penalty given to the agent for selecting scouting runs resulting in high (>> 1 hour) analysis times (i.e. high retention factors), made the agent avoid the lower percentages for compounds like iodobenzene and anthracene, but not for compounds like nitrobenzene and o-methylacetophenone, which in regards to this study was highly desirable.

4.3 Q-values: taking action

While Figure 3 illustrates which actions the agent took depending on the retention behavior of the compounds, Figure 4 illustrates how the agent, or the *Q*-network and its *Q*-values, decided which actions to take, given a certain state. It also illustrates how the *Q*-values varied between steps and between compounds with a different retention behavior (specifically, in Figure 4, acetophenone and biphenyl). Interestingly, the penalty given for high analysis times, forced the *Q*-values for $\varphi_{ACN}= 0.05$ and $\varphi_{ACN}= 0.10$ to differ significantly between the two compounds. Specifically, biphenyl, which is more retained than acetophenone, was predicted by the *Q*-network to result in a high penalty if run at $\varphi_{ACN}= 0.05$ or $\varphi_{ACN}= 0.10$ (the analysis time would be too long), and therefore predicted higher *Q*-values at higher φ_{ACN}. Furthermore, the penalty given when the same φ_{ACN} was selected more than once, resulted in a significant lowering of the *Q*-value in the succeeding steps for that φ_{ACN}.

Acetophenone had higher *Q*-values on average, suggesting the reward/penalty on average was greater for less retained compounds (like acetophenone) than higher retained compounds (like biphenyl). This suggests that (1) $\varphi_{ACN}= 0.05$ was an important datapoint to accurately model retention behaviors (resulting in a low MRPE) – a datapoint which did not get selected for highly retained compounds due to high penalties; (2) the continual higher penalties for highly retained compounds forced down the *Q*-values.

Although not directly presented in this study, the *number* of selected scouting runs, as well as the specific selection of scouting runs may differ between agents trained on slightly different data. Specifically, Figure 4 illustrates how several actions have similar *Q*-values given a certain state, which could easily nudge the agent in a different direction. This does not necessarily mean that the agent (or the *double deep Q-learning algorithm*) is not robust, but rather that there are several solutions to the problem – *i.e.* several directions (combinations of scouting runs) that result in good retention models.

4.4 Retention model performance: isocratic and gradient prediction errors

To get a better understanding of how well the retention models resulting from the scouting runs selected by the agent perform, the prediction accuracy of these retention models was compared to the prediction accuracy of retention models obtained by fitting to (1) all 10 available isocratic data points (φ_{ACN} of 0.05 to 0.90); (2) a random selection of three isocratic scouting runs and (3) a chromatographer's selection of three isocratic scouting runs, namely $\varphi_{ACN}= 0.1$, 0.5 and 0.8. These four differently obtained retention models were also used to predict gradient retention factors for four different gradients (Table 2).

Table 2 illustrates the MRPE between the experimentally obtained data and the predictions, for both isocratic and gradient data (see Table S-4 for complete results). The retention model obtained by the agent (based on (mostly) 3 scouting runs), performed comparably and better to the retention models based on all experimental datapoints and the retention models based on the chromatographer's selection of three scouting runs, for the isocratic and gradient data respectively; and performed significantly better than the retention models based on the random scouting run selection. Importantly, the models resulting from the agent's selection of scouting runs were, compared to retention models based on all datapoints, obtained via seven fewer datapoints; *i.e.* seven fewer scouting runs, saving significant time and costs. These results indicate that the agent successfully learned to optimize the scouting run selection.

The reason why the retention model obtained by the agent had a significantly lower gradient prediction error compared to the retention model fitted on all datapoints and the chromatographer's selection, lies in the high prediction error for the lowly retained compounds. It is speculated that (1) having many datapoints at medium-high φ_{ACN} for the lowly retained compounds, where most φ_{ACN} have retention factors close to 0, is forcing the estimated parameters (k_w, S_1 and S_2) to model the behavior of the given lowly retained compound favorably in the regions of low φ_{ACN}, but highly unfavorably in the regions of high φ_{ACN}, which is more relevant for gradient retention modelling; and (2) not having φ_{ACN} of 0.05 (in the case of the chromatographer's selection) misses out on valuable information for gradient predictions.

5 Conclusions

In this study, a reinforcement learning algorithm, specifically the double deep *Q*-learning algorithm, was shown to be able to learn to optimally select informative scouting runs in a fully self-taught way. Although only isocratic scouting runs were considered for a specific RPLC setup, these experiments illustrate how the agent learned to tailor the selection of scouting runs for different compounds depending on the retention behavior (mainly defined by its polarity, or k_w). The experiments also illustrate how the agent limited the number of selected scouting runs yet still yielding a retention model with low prediction error (MRPE of 3.77% and 1.93% for isocratic and gradient data, respectively). The strategy of the agent was to select relatively evenly spaced-out scouting runs (in terms of φ_{ACN}), including at least one scouting run at low φ_{ACN} (as long as the analysis time was not too long), and at least one scouting run at high φ_{ACN}. Intuitively, selecting scouting runs as such will cover a greater space, better capturing the complete behavior of the compound. The retention models based on the agent's selection of scouting runs (MRPE of 3.77% for the isocratic data and 1.93% for the gradient data) compared well to both the retention models fitted on all datapoints (MRPE of 3.26% for isocratic data and 4.97% for gradient data) and the chromatographer's selection (3.86% for isocratic data and 6.66% for gradient data); and performed significantly better than retention models based on the random selection of three scouting runs (MRPE of 46.30% for isocratic data 7.60% for gradient data).

Although the double deep *Q*-learning algorithm presented in this study shows potential, it has only been tested on its ability to learn from isocratic data for a specific RPLC setup. This double deep *Q*-learning algorithm is also limited to a discrete, and preferably small, action space, which could potentially be a problem for tasks like selecting scouting runs for gradient elution – where the action space is either larger or continuous. The prospect is to develop a reinforcement learning algorithm that can deal with a more complex/larger state and action space (e.g. Branching Dueling Q-learning (67)) and/or continuous action space (e.g. Twin-Delayed Deep Deterministic Policy Gradient (TD3) or Soft Actor-Critic (SAC) algorithm (68,69)), to be able to perform scouting runs for a mixture of compounds (*i.e.* selecting scouting runs for multiple compounds at the same time) in a more complex setting such as gradient conditions. If such an algorithm can be developed successfully, it would be worthwhile integrating it into normal practice.

Availability

All code used in this study (except the plots), including the complete implementation of the agent and the environment, can be found at https://github.com/akensert/ddqn-isocratic-scouting-runs. Due to stochasticity in training an artificial neural network, results may differ somewhat from run to run. InChI for each experimental compound evaluated in this study can be found in supplementary table 5 (Table S-5).

Conflicts of interest

There are no conflicts of interest.

Acknowledgements

Alexander Kensert and Gilles Collaerts are funded by a joint-initiative of the Research Foundation Flanders (FWO) and the Walloon Fund for Scientific Research (FNRS) (EOS – research project "Chimic" (EOS ID: 30897864)). Kyriakos Efthymiadis is funded by the VLAIO O&O project "Amedes" (AIO/HBC.2017.0996/AMEDES). Janssen Pharmaceutica is thanked for financially supporting this work, and especially Peter Van Broeck is thanked for helpful discussions.

Credit Author Statement

Alexander Kensert: Conceptualization; Data curation; Formal analysis; Investigation; Methodology; Validation; Visualization; Writing - original draft.
Gilles Collaerts: Data curation; Formal analysis; Investigation; Validation; Writing – review & editing.
Kyriakos Efthymiadis: Formal analysis; Investigation; Methodology; Writing – review & editing.
Gert Desmet: Conceptualization; Funding acquisition; Investigation; Methodology; Supervision; Writing - review & editing.
Deirdre Cabooter: Conceptualization; Funding acquisition; Formal analysis; Methodology; Investigation; Supervision; Writing - review & editing.

References

1. Xu H, Yang L, Freitas MA. A robust linear regression based algorithm for automated evaluation of peptide identifications from shotgun proteomics by use of reversed-phase liquid chromatography retention time. BMC Bioinformatics. 2008 Aug 19;9:347.

2. Fong SS, Rearden P, Kanchagar C, Sassetti C, Trevejo J, Brereton RG. Automated peak detection and matching algorithm for gas chromatography-differential mobility spectrometry. Anal Chem. 2011 Mar 1;83(5):1537–46.

3. Samanipour S, Reid MJ, Bæk K, Thomas KV. Combining a Deconvolution and a Universal Library Search Algorithm for the Nontarget Analysis of Data-Independent Acquisition Mode Liquid Chromatography-High-Resolution Mass Spectrometry Results. Environ Sci Technol. 2018 17;52(8):4694–701.

4. Peters S, Vivó-Truyols G, Marriott PJ, Schoenmakers PJ. Development of an algorithm for peak detection in comprehensive two-dimensional chromatography. J Chromatogr A. 2007 Jul 13;1156(1–2):14–24.

5. Cramer JA, Hammond MH, Loegel TN, Morris RE. Evolving window factor analysis-multivariate curve resolution with automated library matching for enhanced peak deconvolution in gas chromatography-mass spectrometry fuel data. J Chromatogr A. 2018 Dec 21;1581–1582:125–34.

6. López-Ureña S, Torres-Lapasió JR, Donat R, García-Alvarez-Coque MC. Gradient design for liquid chromatography using multi-scale optimization. J Chromatogr A. 2018 Jan 26;1534:32–42.

7. Freier L, von Lieres E. Multi-objective global optimization (MOGO): Algorithm and case study in gradient elution chromatography. Biotechnol J. 2017 Jul;12(7).

8. Woldegebriel M, Gonsalves J, van Asten A, Vivó-Truyols G. Robust Bayesian Algorithm for Targeted Compound Screening in Forensic Toxicology. Anal Chem. 2016 Feb 16;88(4):2421–30.

9. Zapadka M, Kaczmarek M, Kupcewicz B, Dekowski P, Walkowiak A, Kokotkiewicz A, et al. An application of QSRR approach and multiple linear regression method for lipophilicity assessment of flavonoids. J Pharm Biomed Anal. 2019 Feb 5;164:681–9.

10. Taraji M, Haddad PR, Amos RIJ, Talebi M, Szucs R, Dolan JW, et al. Chemometric-assisted method development in hydrophilic interaction liquid chromatography: A review. Anal Chim Acta. 2018 Feb 13;1000:20–40.

11. Bouwmeester R, Martens L, Degroeve S. Comprehensive and Empirical Evaluation of Machine Learning Algorithms for Small Molecule LC Retention Time Prediction. Anal Chem. 2019 05;91(5):3694–703.

12. Maljurić N, Golubović J, Otašević B, Zečević M, Protić A. Quantitative structure - retention relationship modeling of selected antipsychotics and their impurities in green liquid chromatography using cyclodextrin mobile phases. Anal Bioanal Chem. 2018 Apr;410(10):2533–50.

13. Ramezani AM, Yousefinejad S, Shahsavar A, Mohajeri A, Absalan G. Quantitative structure-retention relationship for chromatographic behaviour of anthraquinone derivatives through considering organic modifier features in micellar liquid chromatography. J Chromatogr A. 2019 Aug 16;1599:46–54.

14. Wen Y, Amos RIJ, Talebi M, Szucs R, Dolan JW, Pohl CA, et al. Retention Index Prediction Using Quantitative Structure-Retention Relationships for Improving Structure Identification in Nontargeted Metabolomics. Anal Chem. 2018 07;90(15):9434–40.

15. Wen Y, Talebi M, Amos RIJ, Szucs R, Dolan JW, Pohl CA, et al. Retention prediction in reversed phase high performance liquid chromatography using quantitative structure-retention relationships applied to the Hydrophobic Subtraction Model. J Chromatogr A. 2018 Mar 16;1541:1–11.

16. Zarei K, Atabati M, Ahmadi M. Shuffling cross-validation-bee algorithm as a new descriptor selection method for retention studies of pesticides in biopartitioning micellar chromatography. J Environ Sci Health B. 2017 May 4;52(5):346–52.

17. Cabooter D, Clicq D, De Boever F, Lestremau F, Szucs R, Desmet G. A variable column length strategy to expedite method development. Anal Chem. 2011 Feb 1;83(3):966–75.

18. Tyteca E, De Vos J, Vankova N, Cesla P, Desmet G, Eeltink S. Applicability of linear and nonlinear retention-time models for reversed-phase liquid chromatography separations of small molecules, peptides, and intact proteins. J Sep Sci. 2016 Apr;39(7):1249–57.

19. Snyder LR, Dolan JW, Gant JR. Gradient elution in high-performance liquid chromatography: I. Theoretical basis for reversed-phase systems. Journal of Chromatography A. 1979 Mar 21;165(1):3–30.

20. Dolan JW, Gant JR, Snyder LR. Gradient elution in high-performance liquid chromatography: II. Practical application to reversed-phase systems. Journal of Chromatography A. 1979 Mar 21;165(1):31–58.

21. Neue UD. Nonlinear Retention Relationships in Reversed-Phase Chromatography. Chroma. 2006 Jun 1;63(13):S45–53.

22. Tyteca E, Desmet G. On the inherent data fitting problems encountered in modeling retention behavior of analytes with dual retention mechanism. J Chromatogr A. 2015 Jul 17;1403:81–95.

23. Tyteca E, Périat A, Rudaz S, Desmet G, Guillarme D. Retention modeling and method development in hydrophilic interaction chromatography. J Chromatogr A. 2014 Apr 11;1337:116–27.

24. Tyteca E, Guillarme D, Desmet G. Use of individual retention modeling for gradient optimization in hydrophilic interaction chromatography: separation of nucleobases and nucleosides. J Chromatogr A. 2014 Nov 14;1368:125–31.

25. Česla P, Vaňková N, Křenková J, Fischer J. Comparison of isocratic retention models for hydrophilic interaction liquid chromatographic separation of native and fluorescently labeled oligosaccharides. J Chromatogr A. 2016 Mar 18;1438:179–88.

26. Tyteca E, De Vos J, Vankova N, Cesla P, Desmet G, Eeltink S. Applicability of linear and nonlinear retention-time models for reversed-phase liquid chromatography separations of small molecules, peptides, and intact proteins. J Sep Sci. 2016 Apr;39(7):1249–57.

27. Roca LS, Schoemaker SE, Pirok BWJ, Gargano AFG, Schoenmakers PJ. Accurate modelling of the retention behaviour of peptides in gradient-elution hydrophilic interaction liquid chromatography. J Chromatogr A. 2020 Mar 15;1614:460650.

28. Hindriks R, Maris F, Vink J, Peeters A, De Smet M, Massart DL, et al. Expert system for the selection of initial high-performance liquid chromatographic conditions for the analysis of pharmaceuticals. Journal of Chromatography A. 1989 Dec 27;485:255–65.

29. Maris F, Hindriks R, Vink J, Peeters A, Vanden Driessche N, Massart L. Validation of an expert system for the selection of initial high-performance liquid chromatographic conditions for the analysis of basic drugs. Journal of Chromatography A. 1990 May 11;506:211–21.

30. De Smet M, Peeters A, Buydens L, Massart DL. Expert system for the selection of high-performance liquid chromatographic methods in pharmaceutical analysis: Validation of the rules for the selection of the mobile phase. Journal of Chromatography A. 1988 Jan 1;457:25–42.

31. Szepesi G, Valkó K. Prediction of initial high-performance liquid chromatographic conditions for selectivity optimization in pharmaceutical analysis by an expert system approach. Journal of Chromatography A. 1991 Jan 1;550:87–100.

32. Gros N, Gorenc B. Expert system for the ion chromatographic determination of alkali and alkaline earth metals in mineral waters. Journal of Chromatography A. 1995 Apr 21;697(1):31–43.

33. Fell AF, Bridge TP, Williams MH. Design and application of an expert system for mobile phase optimisation in reversed-phase liquid chromatography. Journal of Pharmaceutical and Biomedical Analysis. 1988 Jan 1;6(6):555–64.

34. Schoenmakers PJ, Peeters A, Lynch RJ. Optimization of chromatographic methods by a combination of optimization software and expert systems. Journal of Chromatography A. 1990 May 11;506:169–84.

35. Schoenmakers PJ, Dunand N. Explanations and advice provided by an expert system for system optimization in high-performance liquid chromatography. Journal of Chromatography A. 1989 Dec 27;485:219–36.

36. Smith RM, Burr CM. Retention prediction of analytes in reversed-phase high-performance liquid chromatography based on molecular structure: I. Monosubstituted aromatic compounds. Journal of Chromatography A. 1989 Jan 1;475(2):57–74.

37. Valkó K, Szabó G, Röhricht J, Jemnitz K, Darvas F. Prediction of retention of metabolites in high-performance liquid chromatography by an expert system approach. Journal of Chromatography A. 1989 Dec 27;485:349–63.

38. Hamoir T, Bourguignon B, Massart DL, Hindriks H. Model building for the prediction of initial chromatographic conditions in pharmaceutical analysis using reversed-phase liquid chromatography. Journal of Chromatography A. 1993 Feb 24;633(1):43–56.

39. Fekete J, Morovján G, Csizmadia F, Darvas F. Method development by an expert system advantages and limitations. Journal of Chromatography A. 1994 Feb 4;660(1):33–46.

40. Domingos P. A Few Useful Things to Know About Machine Learning. Commun ACM. 2012 Oct 1;55:78–87.

41. Cortes C, Vapnik V. Support-vector networks. Mach Learn. 1995 Sep 1;20(3):273–97.

42. Tin Kam Ho. Random decision forests. In: Proceedings of 3rd International Conference on Document Analysis and Recognition. 1995. p. 278–82 vol.1.

43. Rumelhart DE, Hinton GE, Williams RJ. Learning representations by back-propagating errors. Nature. 1986 Oct;323(6088):533–6.

44. Hopfield JJ. Neural networks and physical systems with emergent collective computational abilities. PNAS. 1982 Apr 1;79(8):2554–8.

45. Kaelbling LP, Littman ML, Moore AW. An Introduction to Reinforcement Learning. In: Steels L, editor. The Biology and Technology of Intelligent Autonomous Agents. Berlin, Heidelberg: Springer; 1995. p. 90–127. (NATO ASI Series).

46. Arulkumaran K, Deisenroth MP, Brundage M, Bharath AA. Deep Reinforcement Learning: A Brief Survey. IEEE Signal Processing Magazine. 2017 Nov;34(6):26–38.

47. Ertefaie A, Shortreed S, Chakraborty B. Q-learning residual analysis: application to the effectiveness of sequences of antipsychotic medications for patients with schizophrenia. Stat Med. 2016 15;35(13):2221–34.

48. Shi W, Song S, Wu C, Chen CLP. Multi Pseudo Q-Learning-Based Deterministic Policy Gradient for Tracking Control of Autonomous Underwater Vehicles. IEEE Trans Neural Netw Learn Syst. 2019 Dec;30(12):3534–46.

49. Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, et al. Playing Atari with Deep Reinforcement Learning. arXiv:13125602 [cs] [Internet]. 2013 Dec 19 [cited 2020 Jul 29]; Available from: http://arxiv.org/abs/1312.5602

50. Hwangbo J, Lee J, Dosovitskiy A, Bellicoso D, Tsounis V, Koltun V, et al. Learning agile and dynamic motor skills for legged robots. Science Robotics [Internet]. 2019 Jan 16 [cited 2020 Jul 29];4(26). Available from: https://robotics.sciencemag.org/content/4/26/eaau5872

51. Liu Z, Abbaszadeh S. Double Q-Learning for Radiation Source Detection. Sensors (Basel). 2019 Feb 24;19(4).

52. Komorowski M, Celi LA, Badawi O, Gordon AC, Faisal AA. The Artificial Intelligence Clinician learns optimal treatment strategies for sepsis in intensive care. Nature Medicine. 2018 Nov;24(11):1716–20.

53. Silver D, Schrittwieser J, Simonyan K, Antonoglou I, Huang A, Guez A, et al. Mastering the game of Go without human knowledge. Nature. 2017 Oct;550(7676):354–9.

54. Silver D, Hubert T, Schrittwieser J, Antonoglou I, Lai M, Guez A, et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. Science. 2018 Dec 7;362(6419):1140–4.

55. Silver D, Huang A, Maddison CJ, Guez A, Sifre L, van den Driessche G, et al. Mastering the game of Go with deep neural networks and tree search. Nature. 2016 Jan;529(7587):484–9.

56. Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, et al. Human-level control through deep reinforcement learning. Nature. 2015 Feb;518(7540):529–33.

57. Vinyals O, Ewalds T, Bartunov S, Georgiev P, Vezhnevets AS, Yeo M, et al. StarCraft II: A New Challenge for Reinforcement Learning. arXiv:170804782 [cs] [Internet]. 2017 Aug 16 [cited 2020 Jul 29]; Available from: http://arxiv.org/abs/1708.04782

58. Pang Z-J, Liu R-Z, Meng Z-Y, Zhang Y, Yu Y, Lu T. On Reinforcement Learning for Full-length Game of StarCraft. arXiv:180909095 [cs, stat] [Internet]. 2019 Feb 3 [cited 2020 Jul 29]; Available from: http://arxiv.org/abs/1809.09095

59. OpenAI, Berner C, Brockman G, Chan B, Cheung V, Dębiak P, et al. Dota 2 with Large Scale Deep Reinforcement Learning. arXiv:191206680 [cs, stat] [Internet]. 2019 Dec 13 [cited 2020 Jul 29]; Available from: http://arxiv.org/abs/1912.06680

60. Press TM. Reinforcement Learning, Second Edition | The MIT Press [Internet]. The MIT Press; [cited 2020 Aug 17]. Available from: https://mitpress.mit.edu/books/reinforcement-learning-second-edition

61. Watkins CJCH, Dayan P. Q-learning. Mach Learn. 1992 May 1;8(3):279–92.

62. Kim S, Chen J, Cheng T, Gindulyte A, He J, He S, et al. PubChem 2019 update: improved access to chemical data. Nucleic Acids Res. 2019 Jan 8;47(D1):D1102–9.

63. Hasselt H van, Guez A, Silver D. Deep reinforcement learning with double Q-Learning. In: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence. Phoenix, Arizona: AAAI Press; 2016. p. 2094–2100. (AAAI'16).

64. Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, et al. Playing Atari with Deep Reinforcement Learning. arXiv:13125602 [cs] [Internet]. 2013 Dec 19 [cited 2020 Aug 14]; Available from: http://arxiv.org/abs/1312.5602

65. Zhang S, Sutton RS. A Deeper Look at Experience Replay. arXiv:171201275 [cs] [Internet]. 2018 Apr 30 [cited 2020 Aug 14]; Available from: http://arxiv.org/abs/1712.01275

66. Hasselt H van. Double Q-learning. In: Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 2. Red Hook, NY, USA: Curran Associates Inc.; 2010. p. 2613–2621. (NIPS'10).

67. Arash Tavakoli, Fabio Pardo, Petar Kormushev. Action Branching Architectures for Deep Reinforcement Learning. In: 31st Conference on Neural Information Processing Systems (NIPS 2017). California, USA; 2017.

68. Fujimoto S, Hoof H, Meger D. Addressing Function Approximation Error in Actor-Critic Methods. In: International Conference on Machine Learning [Internet]. PMLR; 2018 [cited 2020 Oct 27]. p. 1587–96. Available from: http://proceedings.mlr.press/v80/fujimoto18a.html

69. Haarnoja T, Zhou A, Abbeel P, Levine S. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In: International Conference on Machine Learning [Internet]. PMLR; 2018 [cited 2020 Oct 27]. p. 1861–70. Available from: http://proceedings.mlr.press/v80/haarnoja18b.html

Figure Captions

Figure 1: A simple schematic illustration of the interaction between agent and environment. A) the environment supplies the agent with a state *s,* which is an array of retention factors (k) where each element corresponds to a φ_{ACN}; based on this state *s* the agent produces an array of *Q*-values for each action. B) the agent takes the greedy action, namely the action with the highest *Q*-value, resulting in a new state *s'*. For readability, the figure excludes the reward and only covers one single transition (s,a,s').

Figure 2: Illustration of how the reward (top; blue) and the number of scouting runs selected (bottom; orange) by the agent increased over the number of episodes. To reduce the noise of these learning curves, a moving average filter was applied to the data (window size of 35). The lighter shaded areas represent a 95% confidence interval of the average lines. The black line in the top plot illustrates the theoretical optimum (the highest possible reward) that can be obtained by selecting the best combination of three scouting runs out of 120 different combinations.

Figure 3: Illustration of how the scouting runs were selected sequentially for four representative compounds with a significantly different retention behavior. The estimated Neue-Kuss parameter k_w can be used to assess the retention behavior of the compounds, with a high k_w indicating strong retention, and a low k_w indicating weaker retention. The blue dots indicate the experimental data, the orange triangles pointing to the x-axis indicate the scouting runs selected (sequentially, $1 \rightarrow 3$), and the orange lines indicate the predicted retention model resulting from the scouting runs. MRPE stands for mean relative percentage error. The y-axis has been set to log scale to better visualize the prediction of lower φ_{ACN} values.

Figure 4: Illustration of how the scouting runs were sequentially (from top to bottom) selected based on the highest Q-value at each step, for two representative compounds. Similar to Figure 3, the estimated Neue-Kuss parameter k_w indicates the retainability of the compounds, with a high k_w indicating strong retention, and a low k_w indicating weaker retention. Orange bars indicate the selected actions.

Appendix:

Deep Q-learning for the selection of optimal isocratic scouting runs in liquid chromatography

A: Retention Model

The retention model used in this study was the Neue-Kuss model (Eq. 1). The model is obtained by *fitting* the model's parameters (S_1, S_2 and k_w) to the available datapoints for a given compound. The fitting is done via Nelder-Mead optimization (doi: https://doi.org/10.1093/comjnl/7.4.308), which is a type of simplex method.

B: Double Deep Q-network

In this study, the two *Q*-networks had a 10-unit input layer (corresponding to the state of the environment), two 1024-unit hidden layers with rectified linear unit activation and a dropout rate of 0.2, and a final 11-unit linear output layer (corresponding to the actions of the agent). While, the prediction network was updated each episode via the minimization procedure, the target network was updated (by copying over the prediction network's parameters) every 64 steps. The batch-size, which is the number of transitions used per minimization step (Eq. 8), was set to 128. To minimize the objective function (Eq. 8), a stochastic gradient descent optimizer was used, with a momentum and an initial learning rate of 0.9 and 0.001 respectively. The learning rate was scheduled to decay for 4096 iterations (episodes) until it reached a minimum learning rate of 0.0001.

Before selecting the hyperparameter values mentioned above, some preliminary testing was done (i.e. the *Q*-network was tested with different hyperparameter values). The number of hidden units was varied between 512 and 2048 (for which 1024 was equal or better to the other values and was therefore selected), the number of layers were varied between 1 and 3 (for which 2 was selected), and the dropout rate was varied between 0.1 and 0.5 (for which 0.2 was selected). Furthermore, the initial learning rate was varied between 0.01 and 0.001 (for which 0.001 was selected), and both rectified linear unit (ReLU) and sigmoid activation was tested (for which ReLU was selected). Finally, discount factor γ was varied between 0.0 and 1.0 (for which 0.95 was selected).

C: Replay Memory/Experience Replay

Because the batch-size was set to 128, the number of samples sampled from the replay memory unit was also 128. The capacity (maximum number of transitions stored) for the replay memory unit was restricted to 2048. This restriction was added to avoid having the *Q*-network learn from old experiences (transitions). The sampling of transitions from the replay memory unit did not start until it had at least 512 transitions stored; because the *Q*-network (or specifically the prediction network) was updated every episode, having at least 4 times the batch-size of transitions helped to avoid having the *Q*-network train on similar batches multiple times in a row; and although no evidence is presented here, it was thought that this could potentially bias the *Q*-network at the beginning of training and hence slow down learning. As for the hyperparameters of the *Q*-network the batch size was varied between 32 and 512 (for which 128 was selected).

D: Epsilon-greedy policy

The epsilon-greedy policy is important for balancing the exploration-exploitation tradeoff. The epsilon-greedy policy in this study was divided into $\varepsilon_{initial}$, ε_{decay}, and $\varepsilon_{minimum}$, which were set to 1.0, 0.99, and 0.1 respectively. This made the agent take mostly random actions (exploring) in the beginning, and increasingly by time started to take actions based on the output of the Q-function (the Q-values), namely $\arg\max_{\mathrm{a}} Q(s, a)$ (exploiting), The minimum epsilon was kept at 0.1 to force the agent to explore throughout the entire run. As for the hyperparameters of the Q-network, the initial epsilon, the decay rate, as well as the minimum decay was varied between 0.1-1.0, 0.99-0.999 and 0.0-1.0 respectively. For which 1.0, 0.99 and 0.1 was selected respectively.