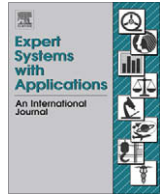




Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Assigning discounts in a marketing campaign by using reinforcement learning and neural networks

Gabriel Gómez-Pérez^a, José D. Martín-Guerrero^{a,*}, Emilio Soria-Olivas^a, Emili Balaguer-Ballester^b, Alberto Palomares^b, Nicolás Casariego^c

^a Digital Signal Processing Group, Department of Electronic Engineering, University of Valencia, CL. Dr. Moliner, 50. 46100 Burjassot, Valencia, Spain

^b Tissat S.A., iSUM Department, Avenue Leonardo Da Vinci, 5. 46980 Paterna, Valencia, Spain

^c Espirius Europa Ltd., Spain

ARTICLE INFO

Keywords:

Reinforcement learning
Function approximation
State aggregation
Neural networks
Marketing

ABSTRACT

In this work, RL is used to find an optimal policy for a marketing campaign. Data show a complex characterization of state and action spaces. Two approaches are proposed to circumvent this problem. The first approach is based on the self-organizing map (SOM), which is used to aggregate states. The second approach uses a multilayer perceptron (MLP) to carry out a regression of the action-value function. The results indicate that both approaches can improve a targeted marketing campaign. Moreover, the SOM approach allows an intuitive interpretation of the results, and the MLP approach yields robust results with generalization capabilities.

© 2008 Elsevier Ltd. All rights reserved.

1. Introduction

The latest marketing trends are more concerned about maintaining current customers and optimizing their behaviour than getting new ones. For this reason, relational marketing focuses on what a company must do to achieve this objective (Reichheld, 2001). The relationships between a company and its costumers follow a sequence of action-response cycles, where the customers can modify their behaviour in accordance with the marketing actions developed by the company.

One way to increase the loyalty of customers is by offering them the opportunity to obtain some gifts as the result of their purchases from the company. The company can give *virtual credits* to anyone who buys certain articles, typically those that the company is interested in promoting. After a certain number of purchases, the customers can exchange their virtual credits for the gifts offered by the company.

The problem is to establish the appropriate number of virtual credits for each promoted item. In accordance with the company policy, it is expected that the higher the credit assignment, the higher the amount of purchases. However, the company's profits are lower since the marketing campaign adds an extra cost to the company. The goal is to achieve an optimal trade-off by establishing a policy. The development of such a policy is not easy because there are many variables to take into account.

Applications of this kind can be viewed as a Markov decision problem, in which a company decides what action to take once the customer properties in the current state (time t), are known. We propose the use of reinforcement learning (RL) to solve this task since previous applications have demonstrated its suitability in this area. In Sun (2003), RL has been applied to analyses of mailing by studying how an action in time t influences actions in following times. In Abe et al. (2002) and Pednault et al. (2002), several RL algorithms have been benchmarked in mailing problems. In Abe, Verma, Schroko, and Apte (2004), RL has been used to optimize cross channel marketing.

The main difference between the mailing problem and the credit assignment problem is that the action space becomes multi-valued instead of binary. In the mailing problem only two actions can be considered: to send a catalogue or not to send a catalogue. In a credit assignment application, the optimal policy should recommend how many credits should be assigned to each transaction. This is the main novelty of this work, since there has been no other attempt, to authors' knowledge, to tackle a problem of this kind using RL. However, it seems to be a suitable application of RL, as this is a problem with clearly identifiable states and actions, and there is also an obvious reward function to be maximized.

Marketing problems tend to have a very complex characterization of the transactions that are involved. This high-dimensionality requires the implementation of RL algorithms by means of state aggregators or function regressors, which under certain conditions may lead to convergence problems (Baird & Moore, 1999; Sutton & Barto, 1998; Sutton, McAllester, Singh, & Mansour, 2000). In this work, we propose two different approaches:

* Corresponding author. Tel.: +34 963160198; fax: +34 963160466.
E-mail address: jose.d.martin@uv.es (J.D. Martín-Guerrero).

- (1) The state space is clustered using a vectorial quantization carried out by algorithms based on the self-organizing map (SOM); this approach enables us to work with RL tabular methods (Smith, 2002).
- (2) A Multilayer perceptron (MLP) is used to predict the response of customers when different actions are carried out by the company. This prediction is then used to obtain an optimal policy.

The remainder of the paper is outlined as follows. RL, SOM and the MLP are described in Sections 2–4, respectively. Problem modelling is presented in Section 5. Section 6 shows the results achieved. Section 7 discusses the relevance, implications, and limitations of this work, ending up the paper with some conclusions and proposals for further work in Section 8.

2. Reinforcement learning

RL algorithms are based on the interaction between an agent and its environment as shown in Fig. 1. The agent is the learner which interacts with the environment, making decisions according to observations made from it. The environment is every external condition that cannot be modified by the agent (Sutton & Barto, 1998).

The task of the learning agent is to optimize a certain objective function; this optimization is carried out using only information from the state of the environment, i.e. without any external advisor. Specifically, the task of the learning agent can be accomplished by modelling the system as a *Markov decision process* (MDP). Due to their wide use throughout the paper and in order to facilitate comprehension, the following terms are defined as follows:

- **State of the environment** (s_t): Available information to define the environment at time t .
- **Action** (a_t): Action taken by the agent at time t .
- **Policy** ($\pi(s, a)$): Probability distribution over the actions in the state s .
- **Immediate reward** (r_{t+1}): Value returned by the environment to the agent depending on the action at time $t + 1$.
- **Long-term reward** (R_t): Sum of all the immediate rewards throughout a complete decision process. It is the objective function that the agent is interested in maximizing by taking the right actions (Sutton & Barto, 1998). Its mathematical expression is:

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \quad (1)$$

where γ is called the *discount-rate* parameter, which ranges between 0 and 1. This factor determines the present value of future rewards: a reward received k time steps in the future is worth only γ^{k-1} times what it would be worth if it were received immediately.

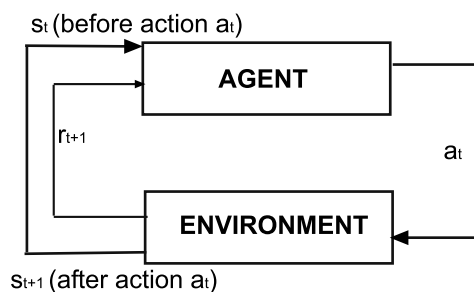


Fig. 1. Characterization of the reinforcement learning model. a_t is the action taken by the agent at time t , s_t is the state of the environment at time t , and r_{t+1} denotes the reward at time $t + 1$.

Small values of γ indicate that only next rewards are taken into account, i.e., the agent maximizes r_{t+1} (*myopic agent*). However, as γ approaches 1, further future rewards become more and more relevant, thus making the agent be more farsighted.

The goal is to maximize R_t by means of an optimal policy, which tells the agent the best action to take for an optimal performance. Therefore, an estimation of the expected R_t as the result of an action a_t from a state s_t is required. This estimation is usually called *action-value function*:

$$Q^\pi(s, a) = E_\pi[R_t | s_t = s, a_t = a], \quad (2)$$

where $Q^\pi(s, a)$ is the expected R_t starting at state s and taking action a , following policy $\pi(s, a)$. Therefore, the optimal policy is given by the following expression (Sutton & Barto, 1998):

$$\pi^*(s, a) = \arg \max_{a \in A} Q^\pi(s, a), \quad (3)$$

where A stands for the set of possible actions. Optimal policies thus share the same optimal action-value function denoted $Q^*(s, a)$:

$$Q^*(s, a) = \max_{\pi} Q^\pi(s, a). \quad (4)$$

Eq. (4) shows a deterministic policy; that is, given a fixed state only one action can be taken. Another approach is to have stochastic policies where actions are taken according to their probabilities, which are given by $\pi(s, a)$ (Sutton & Barto, 1998). In any case, the optimal policy is always deterministic in single-agent MDPs, and in particular, it is also stationary in infinite horizon problems. Therefore, this work is focused on deterministic policies.

RL algorithms are devoted to the computation of the action-value function $Q^\pi(s, a)$ for a given arbitrary policy in order to obtain the optimal policy using (4). Numerous methods for carrying out this task are proposed in the bibliography, but they can be grouped into three main methodologies (Sutton & Barto, 1998):

- **Dynamic programming (DP)**: When a complete model of the environment is available, it is possible to obtain the value functions by means of the Bellman equation (see (Bertsekas, 2001) for a detailed description).
- **MonteCarlo (MC) methods**: These methods do not require any kind of environment modelling since they only require experience (sample sequences of states, actions and rewards from on-line or simulated action with an environment). When MC methods are used, R_t is computed when an episode finishes (off-line algorithms); then, $Q(s, a)$ is updated. For every episode:

$$Q(s, a) \leftarrow \text{Average}(R_t(s, a)). \quad (5)$$
- **Temporal difference methods (TD)**: Techniques of this kind do not require a model of the environment like DP because the values of $Q_{t+1}(s, a)$ are updated using information from the environment (r_{t+1} and s_{t+1}) as well as estimations of $Q_t(s, a)$.

Sarsa and *Q-learning* are the best known TD methods. *Sarsa* is an on-line algorithm that modifies the starting policy towards the optimal one. *Q-learning* computes the optimal policy while the agent is interacting with the environment by means of another arbitrary policy as shown in (6). Since *Q-learning* is easy to implement and enables early convergence, this algorithm is used in this work:

$$Q_{t+1}(s_t, a_t) \leftarrow Q_t(s_t, a_t) + \alpha [r_{t+1} + \gamma \max_{a \in A} (Q_t(s_{t+1}, a)) - Q_t(s_t, a_t)], \quad (6)$$

where α is the step-size of the update. Q_t stands for the action-value function for a particular state before being visited at time t . Q_{t+1} is the updated value of that state once it has been visited.

Methods that explicitly save the values obtained from the interaction are called tabular methods. In this case, $Q(s, a)$ becomes a table with the state and the action as entries. This table is filled in with the observations from the real interaction between the agent and the environment. As the number of states and/or actions increases, it becomes more and more difficult to accurately fill in the Q -table due to the *curse of dimensionality* (Sutton & Barto, 1998). In addition, generalization capabilities are required to handle new data. This problem is usually solved by using function approximation techniques to predict $Q(s, a)$ (Bertsekas & Tsitsiklis, 1996; Bertsekas, Borkar, & Nedi, 2003). In particular, artificial neural networks (ANNs) are the methods that are most commonly used to solve RL problems that involve large state spaces. The problem arises from the lack of data from those states that have never been experienced before, since the optimal policy requires data corresponding to different actions for every state. The desired signal for training, R_t , can be computed by any of the three methods mentioned above. However, the use of MC methods is recommended for convergence reasons. TD methods not for replacing eligibility traces provide a biased estimation because the targets are computed based on a previous estimation, whereas Monte Carlo methods are unbiased since they work with the real value of R_t . The use of tabular methods is also possible by mapping many similar states into only a few. This task can be carried out by algorithms based on the SOM, as shown in Section 3.

3. Self-organizing map

Neural models based on the SOM map an n -dimensional input space into a lower-dimensional space, which is usually one-dimensional or two-dimensional (Kohonen, 2001). Each neuron has a vector of weights $\mathbf{w} = [w_1, w_2, \dots, w_n]$, where w_i represents the i th variable in the neuron; each input pattern is related to some neuron in the output map. Thus, similar input patterns (in the n -dimensional space) are mapped into the same neuron in the lower-dimensionality space. The main advantage of these models stems from the fact that this dimensionality reduction preserves the topological relationships among data. The operation of SOM can be summarized in two steps:

- (1) For a given input vector $\mathbf{x} = [x_1, x_2, \dots, x_n]$, the distances between this vector and all neurons are computed. The closest neuron to the input pattern is the so-called *best matching unit*, (BMU).
- (2) The weight vector values w_i are then updated so that the match of the new set of weights in the active area or *neighbourhood* around BMU and the input pattern \mathbf{x} is improved:

$$\mathbf{w}_i = \mathbf{w}_i + \alpha N(i, \text{BMU})(\mathbf{x} - \mathbf{w}_i), \quad (7)$$

where α is the learning rate and $N(i, \text{BMU})$ represents the active area inside which weight vectors are updated. Several neighbourhood functions are proposed (Kohonen, 2001). We use the Gaussian neighbourhood function:

$$N(i, \text{BMU}) = N_0 \exp \frac{-\|r_i - r_{\text{BMU}}\|^2}{2\sigma^2}, \quad (8)$$

where r_{BMU} represents the centre of the active area; $\|r_i - r_{\text{BMU}}\|$ is the distance between the i th input pattern and its BMU, σ is the standard deviation of the neighbourhood region, and N_0 may be regarded as the height of the neighbourhood kernel. Note that not only are the weights of the BMU updated, but also the weights of the units in the neighbourhood.

Thus, the weights of the network form a new data space where close neurons have similar properties according to the original dis-

tribution of data. This can be viewed as a vector quantization, where the number of features required to identify an input pattern is reduced by choosing some representative models which are tuned by the SOM. For this reason, SOM has been used to quantize input spaces in RL frameworks (Smith, 2002). In this work, the transaction data set is used as the input patterns to train a SOM network. This way, the dimensionality of the state space is reduced, and similar customer behaviours are located in areas of the map that are close to each other.

4. Multilayer perceptron

An MLP is built by combining neurons that are arranged in different layers: an input layer, an output layer, and one or more hidden layers. The input layer receives the samples of the data set, whereas the output layer computes the output of the network. Hidden layers allow the network to carry out non-linear mappings (Haykin, 1999).

An artificial neuron implements a non-linear transformation of its inputs:

$$y = \varphi \left(\sum_{i=1}^m w_i x_i + b \right), \quad (9)$$

where m is the number of inputs to the neuron, w_i is the i th synaptic weight, b is the so-called bias, and φ is a non-linear function, the so-called activation function. This activation function is usually the sigmoid function. Practical implementation usually considers bias b as a synaptic weight that is connected to an additional input whose value is one. The structure is completely defined by taking x_i to be the external inputs, and y to be the output of the neuron.

As we are dealing with a modelling problem whose output shows a continuous range, non-linearity in the output neurons is not necessary. Therefore, the function φ is removed in the output neurons, and only the linear combination of the inputs defines the processing of these neurons.

Among the different algorithms that can be used to train the network, to find optimal values for the synaptic weights, we used the widely used Levenberg–Marquardt algorithm (Bishop, 1995) in this work.

5. Problem modelling

5.1. Data collection and setup

The data were collected from a company¹ that was interested in designing a campaign to encourage their clients to buy more of their products. Table 1 shows the details of the data used in the study. This marketing campaign was based on assigning virtual credits to customers. The information used for this study corresponds to the first five months of the campaign.

Although a confidentiality agreement prevents a number of details of the campaign from being released, the main characteristics of the campaign can be made public:

- (1) The company assigned virtual credits to customers according to the items they bought. When customers had enough credits, they could exchange their credits for gifts.
- (2) Customers could obtain virtual credits by buying specific items which were indicated as “encouraged”. The company selected these promoted items monthly according to internal criteria.

¹ The name of the company cannot be made public due to a confidentiality agreement.

Table 1
Marketing campaign: main characteristics of the data used in the study.

Number of transactions	1,264,862
Number of different articles involved in the transactions	1004
Number of customers	3573
Number of (monthly) episodes	5

- (3) Since the assignment of these virtual credits involved a cost to the company, immediate profits decreased as a direct consequence of the campaign.

The credit assignment took place at the end of every month and was computed by taking into account how many “encouraged” articles were bought by customers during that month. The so-called life-time value (LTV) at time t (reward) for a certain customer was obtained as follows:

$$LTV(t) = \sum_i P_i(t) \cdot A_i(t) - K_C V_C(t), \quad (10)$$

where A_i is the amount of type i articles purchased by the customer, P_i is the price of type i articles, V_C is the number of virtual credits assigned to the customer, and K_C is a coefficient that reports the costs incurred by the company for the credits. The aim of this work is to increase LTV for every customer by using RL as the strategy to achieve an optimal policy.

Since it was not possible to carry out the improvement of the policy on-line, a batch method was used. Episodes were repeatedly shown to the RL algorithm until the policy convergence.

5.2. Action and state spaces

The first task to tackle in an RL algorithm is the design of state and action spaces. This requires an exhaustive analysis of both the clients and their actions. Since an MDP approach was used, the state and action vectors had to contain all the information required to model the past evolution of the system.

The vast amount of information stored by the company showed that there were many features that defined the customer behaviour. Specifically, the following features were included in the study:

- (1) The identification of the shop that sold the products.
- (2) The geographical area where the client made the purchase.
- (3) The date of the purchase.
- (4) The number of items purchased by the client.
- (5) Family².
- (6) The item identification number.
- (7) Whether the items were regular items or “encouraged” items.
- (8) The price of the item purchased.

An initial classical data mining study was carried out to analyse the data. The following tasks were carried out: frequency analysis, distribution of values for the different variables, correlation analysis, cross-tables to analyse possible relationships among variables, periodical analysis, discretization of continuous variables based on percentile analysis, and clustering analysis (using K-means and the adaptive resonance theory). These analyses were used for data filtering, thus removing outliers and irrelevant data. In particular, it should be pointed out that this study showed that neither geographical nor temporal information were relevant; therefore, this information was removed from further analyses.

² Articles were grouped into families. A family was a label which gathered similar products.

Moreover, marketing studies consider that an optimal set of features to profile the future behaviour of a customer is given by the so-called RFM variables (Pfeifer & Carraway, 2000):

- **Recency (R)**: The most recent date that the customer made a transaction (usually a purchase, but it can also be a refund request, for instance).
- **Frequency (F)**: The number of times the customer made a purchase.
- **Monetary (M)**: The monetary amount of products purchased by the customer.

It has been confirmed experimentally that the most valuable customers have high frequency and monetary values, and low recency values (Pfeifer & Carraway, 2000). Although this set of features is the most suitable one to define the state space, in the case of RL algorithms, it is not advisable to use the same information in the definition of the state space and in the computation of the long-term reward (Sutton & Barto, 1998). Therefore, the monetary feature was split into two different variables:

- **Amount (A)**: The number of items purchased in an episode by each customer.
- **Average price (A-P)**: The average price of all the purchases in an episode.

These two features provide the same customer information as the monetary feature, but it is avoided the use of the same data in the state vector and the reward computation. In addition, another feature was considered in the state space. Since customers obtained virtual credits by purchasing some items marked as “encouraged”, it made sense to add a feature that showed how many “encouraged” items were bought by the customers (variable **Encouraged (E)**).

The action to be taken was the assignment of a certain number of virtual credits to customers depending on their purchases. Since there was a wide range of credits, their quantization was required to have a manageable number of possible actions. This quantization procedure was carried out taking into account the different gift values for a certain number of credits. The company established 10 categories of gifts according to their price, so the action space was also divided into 10 categories.

Based on the data set and the action and state spaces described above, the use of two algorithms is proposed. First, a SOM was used to carry out a state aggregation in order to avoid the problem of high dimensionality. Second, an MLP was used to predict the values of $Q^\pi(s, a)$ that were not present in the data, thus allowing us to compute a good policy.

5.3. The SOM approach

In this work, a SOM was used to model the state space. This way, all the features present in the input data could be taken into account without having to deal with a high-dimensional problem. A SOM was trained with all the input patterns from the data collection, and then, the BMU for a particular input vector was computed. This BMU represented the state of the customer. This approach located similar customers in the same state; i.e., an aggregation of states was obtained as shown in Fig. 2 (left-side). Given an input vector $s_i = [R, F, A, A - P, E]$, its BMU was computed. This value was used as an entry for the Q-table. Therefore, the input patterns actually contained information about the marketing-oriented features.

An additional advantage of this proposal is that the interpretation of results was quite straightforward due to the intuitive maps provided by the SOM. Moreover, there was no need to use function

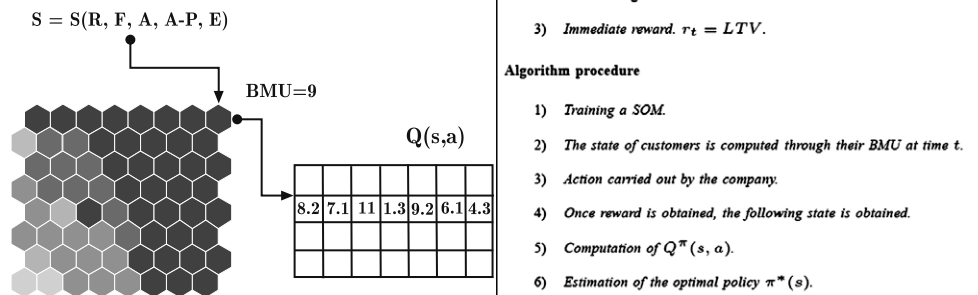


Fig. 2. Left: procedure of state aggregation using the self-organizing map (SOM). The Q-table is filled in with the values of the best matching units (BMUs). Right: estimation of the optimal policy by using a SOM approach for state aggregation, followed by reinforcement learning (RL) tabular methods to obtain the optimal policy.

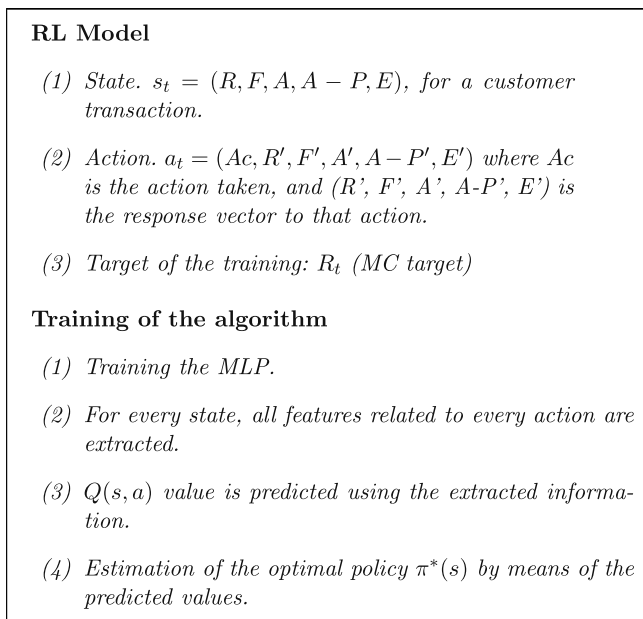


Fig. 3. Procedure of estimation of the optimal policy by using an MLP approach for $Q^{\pi}(s, a)$ prediction, followed by an RL estimation of the optimal policy using the predicted values.

Table 2
Characteristics of the best MLP model obtained in the regression of $Q(s, a)$.

Input space	11 features
Training algorithm	Levenberg–Marquardt
Hidden layers	2
Neurons in each hidden layer	8
Activation function	Hyperbolic tangent (only in hidden layers)

regressors, thereby avoiding the danger of non-convergent policies (Baird & Moore, 1999). The entire process is summarized in Fig. 2 (right-side).

5.4. The MLP approach

ANNs were used in this study to predict the values of $Q(s, a)$. Specifically, the MLP was the neural network used. The input space was made up of the set of transaction features, and the output space consisted of the long-term rewards. Moreover, in the predic-

tion problem, it was necessary to include not only the state features, but also the features of the actions taken by the company. It was then feasible to predict the Q -values when taking different actions in the same state. Therefore, new features were included in the MLP model: the action, i.e., the number of credits given if certain products were purchased, and the response of customers to actions.

The MLP was trained using the data provided by the company. The targets of the training were computed using MC methods because they ensured convergence of the regression algorithm. Fig. 3 shows the procedure carried out. The main characteristics of the best MLP regressor are shown in Table 2.

6. Results

Since the algorithms used in this work were off-policy, the data had to be arranged in episodes. Each episode was made up of the transactions of each customer in each one of the temporal steps that appeared in the data set (five episodes).

6.1. SOM results

Fig. 4 shows the maps yielded by SOM in terms of the characteristics being considered. These maps led to the following conclusions:

- There was a high correlation among the variables amount, encouraged, frequency, and action. All these maps were very similar; i.e., both high and low values appeared in the same states for each variable. Therefore, customers who had high values for the amount and frequency features were those who received more virtual credits in accordance with their purchases. In addition, these customers were interested in the “encouraged” items.
- Customers who had not recently purchased any product (i.e. those with high values of recency) were not profitable for the company because they bought the cheapest articles and in small quantities.
- Products with high prices showed low values of frequency and amount. This might be due to the fact that these articles were long-duration products with a very specific use; hence, customers needed to buy only one product over a long time period.

The policy followed by the company and the comparison between the company policy and that proposed by the SOM–RL approach are shown in Fig. 5, in which a SOM visualization is used

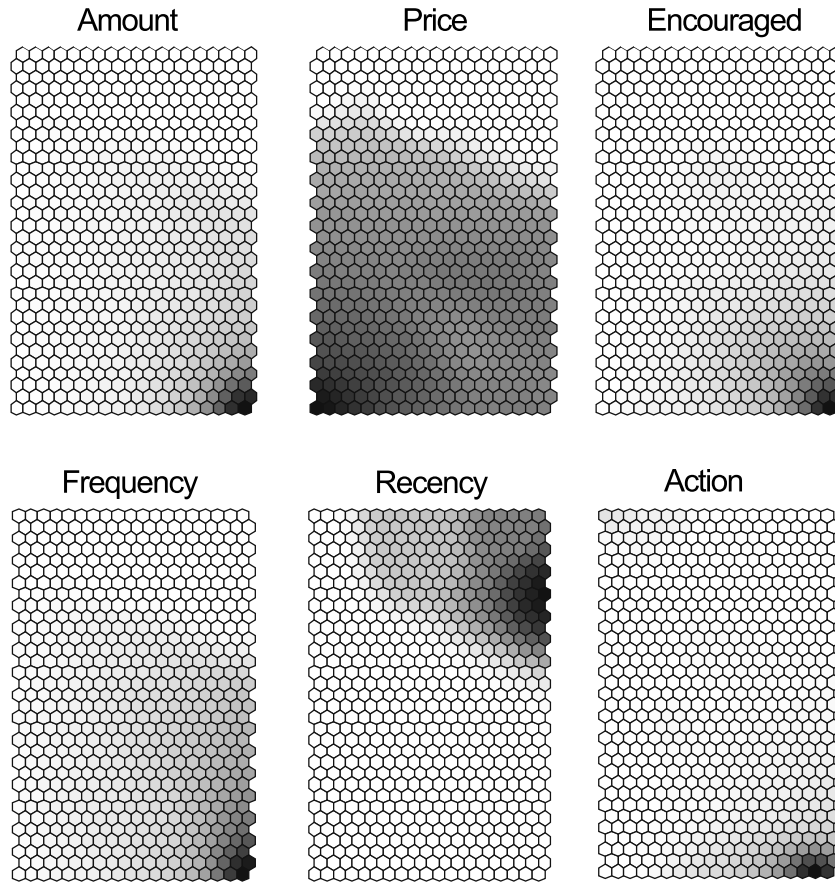


Fig. 4. SOMs achieved with the data set provided by the company. The six maps stand for the six features used in the modelling. The darker the colour, the higher the value of the feature.

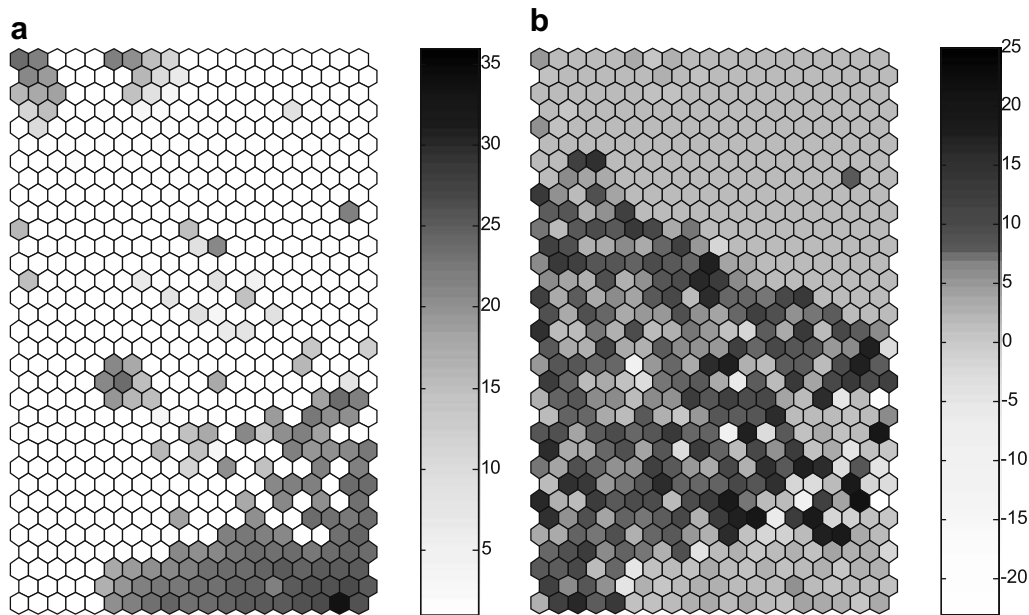


Fig. 5. SOM visualizations to show the policy followed by the company (a) and improvements suggested by the SOM-RL algorithm (b). In the company policy map, (a), the gray-shaded bar shows the number of credits given to customers; therefore, the darkest area depicts a high number of credits given to customers; the medium gray area shows intermediate numbers, and the white area indicates that no credits were given to customers. In the comparison map, (b), the gray-shaded bar shows the difference between the number of credits given by the SOM-RL policy and that given by the company policy; therefore, the darkest areas mean that an increase in the credit assignment is recommended by the optimal policy; medium gray areas mean no modification in the company policy is suggested and the light gray areas (very small) indicate that a decrease in the credit assignment should be made.

for the sake of clarity. In particular, Fig. 5a shows that the policy followed by the company was very inflexible. Three zones are well-defined in this map. The darkest area depicts high number of credits given to customers; the gray area depicts intermediate amounts, and the white area indicates that no credits were given to customers. This guideline of the company policy is a disadvantage for the RL algorithms since they need all actions to be taken in all states. This is known as the *lack of exploration* (Kakade, 2003).

In spite of this lack of exploration, the RL algorithm was able to find a good enough solution to this problem. In fact, when computing the $Q(s, a)$ function by means of the SOM approach, several modifications of the actual policy were suggested as shown in Fig. 5b. There are two main areas in this comparison map: the upper-right and the lower-right areas where no modifications were suggested, and the central area where the optimal policy suggested an increase in the number of credits given to customers. The former represents well-defined customers (“loyal” customers in the case of the lower-right area and “lost” customers for the company in the case of the upper-right area). The central area represents “average” customers. There are two reasons why the optimal policy did not recommend any changes for the well-defined customers:

- (1) The company policy was absolutely strict for these customers, and therefore, there was no way to improve the policy by means of RL algorithms.
- (2) Since the behaviour of these customers was well known by the company, the company policy was probably optimal.

The area where several improvements were suggested was related to the average values of all features. Customers in these states were the most likely to strengthen their relationship with the company according to the recommendations of the optimal policy. The company should focus its attention on customers of this kind.

6.2. MLP results

The data was split into three data sets: a training data set made up of 9460 patterns to train the network; a validation data set that consisted of 2360 patterns to carry out a cross-validation; and finally, a test data set that consisted of another 2360 patterns which had not yet been seen by the network. Unbiased models with good generalization capabilities were obtained in this way.

Very similar values of the mean-square error (MSE) between the targets and the network outputs were obtained in these three data sets. In particular, errors showed low values, which corresponded to an accurate regression, as shown in Fig. 6. However, the use of MLPs to solve RL problems does not require a perfect regression of the values of $Q(s, a)$. Since the goal is to achieve an optimal policy, the regressor must learn the difference between several states and actions, but an accurate estimation of the $Q(s, a)$ function is not required. Fig. 6 shows how the network did correctly model the tendency of the desired signal.

Once the MLP was trained, it was used to estimate the optimal policy using the scheme shown in Fig. 3. The results achieved with this algorithm were difficult to visualize because the input space had 11 dimensions. Fig. 7 shows several plots comparing the MLP-RL policy and the company policy with the features that characterize states. Note that in Fig. 7c (which shows both policies vs. the characteristics “frequency” and “encouraged”), the company policy assigned virtual credits to those customers who were prone to buying “encouraged” articles. The MLP-RL policy suggested that credits should be assigned to customers who purchased products more frequently even when these products were not “encouraged”.

The overall behaviour of the MLP-RL policy was more “aggressive” than the company policy. For instance, in the company policy, actions were spread over the entire action space, whereas the two main groups in the MLP-RL policy were located at the top and at the bottom of the action space. Similar states seemed to lead to

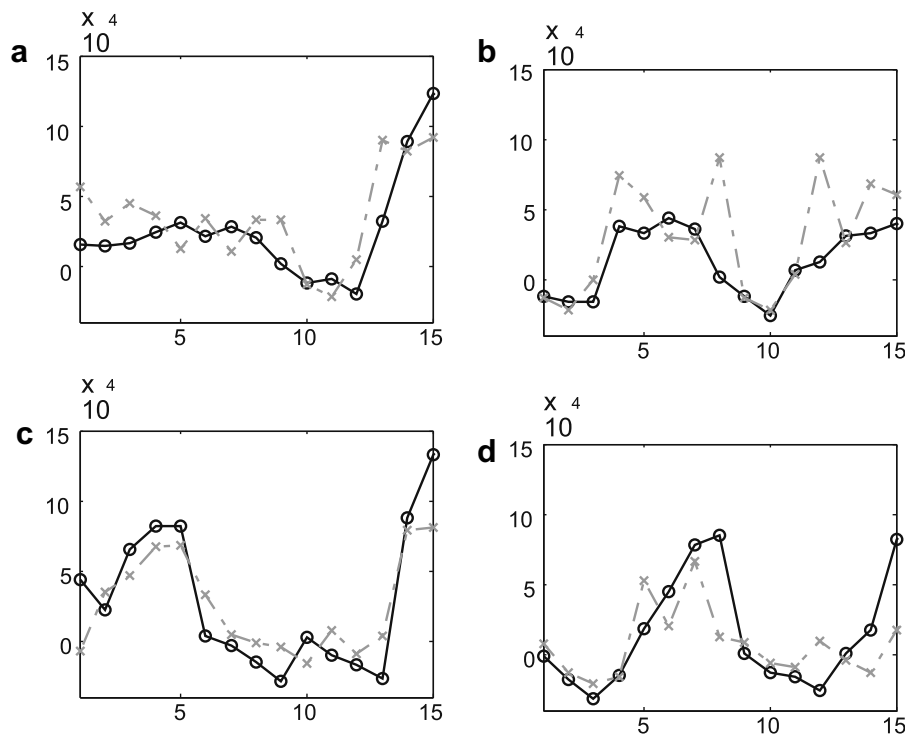


Fig. 6. Regression results in the test (a and b) and validation data sets (c and d). Each plot includes three customers (15 episodes). The solid lines show the desired signal and the dashed lines show the output of the network. This figure shows extreme cases: accurate regressions are shown in (a) and (c), whereas the worst regressions obtained by the model are shown in (b) and (d).

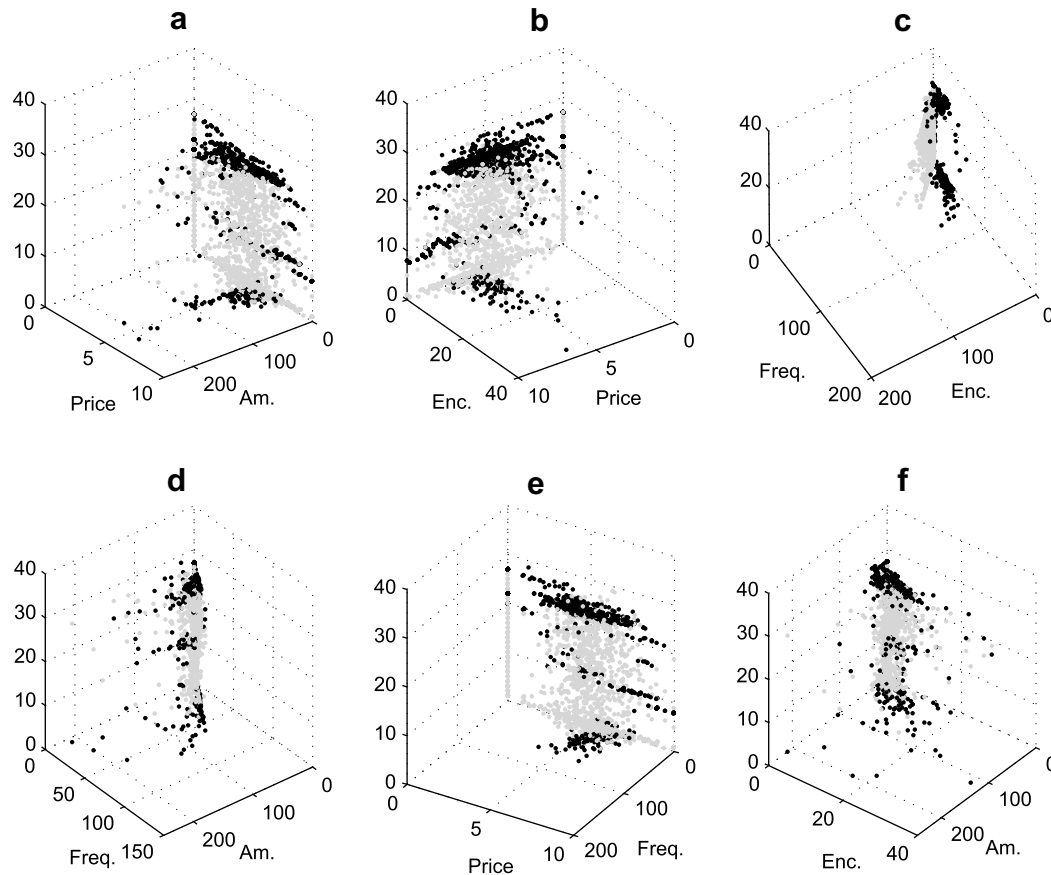


Fig. 7. Several plots showing the comparison of the MLP-RL policy and the company policy with the features of the state space. The gray dots represent the company policy, while the black dots represent the MLP-RL policy. The vertical axis represents the number of credits assigned to customers vs. a horizontal plane defined by two characteristics of the state space in each plot, namely, “price” and “amount of items purchased” in (a); “encouraged” and “price” in (b); “frequency” and “encouraged” in (c); “frequency” and “amount” in (d); “price” and “frequency” in (e); “encouraged” and “amount” in (f).

opposite actions. The explanation of this behaviour might be related to the features used to define the state space. The fact that opposing actions were recommended in similar states suggests that these features might not be a complete state representation.

Once an optimal policy was developed, its performance had to be tested. However, in this application, this proved to be quite difficult, as will be explained in the next section.

7. Discussion

As stated in Section 2, $Q^\pi(s, a)$ is the value of the long-term reward starting in state s , and taking action a , following a policy π . The off-line algorithms used in this work yielded an optimal policy which led us to the $Q^\pi(s, a)$ functions shown in Figs. 8 and 9.

Fig. 8 shows how the SOM-RL policy helps the company to increase its profits considerably. In those areas in which improvements were suggested, the Q -values for the SOM-RL policy were four or even five times larger than the Q -values of the company policy.

The results yielded by the MLP approach are shown in Fig. 9. The black dots are more uniformly distributed than the gray dots, which might be due to the fact that the MLP-RL policy tried to make customers loyal to the company by ensuring a certain minimum number of sales. The behaviour of the company policy was either excellent or very poor. The Q -values obtained by the MLP-RL policy were not as high as some of the results obtained by the company policy in some cases. Nonetheless, the MLP-RL policy was less likely to obtain poor results than the company policy. Moreover, the fact that the MLP-RL policy could not achieve very

high Q -values might be due to the use of MLPs, since the modelling of extreme patterns is rather difficult when using these models.

It is important to point out that the comparison of the Q -functions presented in this work was not completely fair. This is because the reward given to a customer in a certain state might be very different from that given to another customer in the same state taking the same action. In order to circumvent this problem, it might be possible to develop an off-line validation by analysing the actions suggested by the RL policies in the data set. However, in our study, it was not possible to find an episode in the data set provided by the company which fully agreed with the RL policies. Therefore, a fair validation of the optimal policy would require its use by the company for several months. By doing this, the data obtained during that period would be an unbiased estimation of the actual performance of the proposed approaches.

8. Conclusions and further work

In this work, two different approaches to RL applications in targeted marketing have been developed and benchmarked. The application of both algorithms has proven to be appropriate for problems of this kind because promising results have been achieved.

The SOM approach is more straightforward to understand than the MLP approach due to the properties of the SOM algorithm, which allows an intuitive representation of the results. The results suggest that the use of SOM could be recommended to improve a marketing campaign.

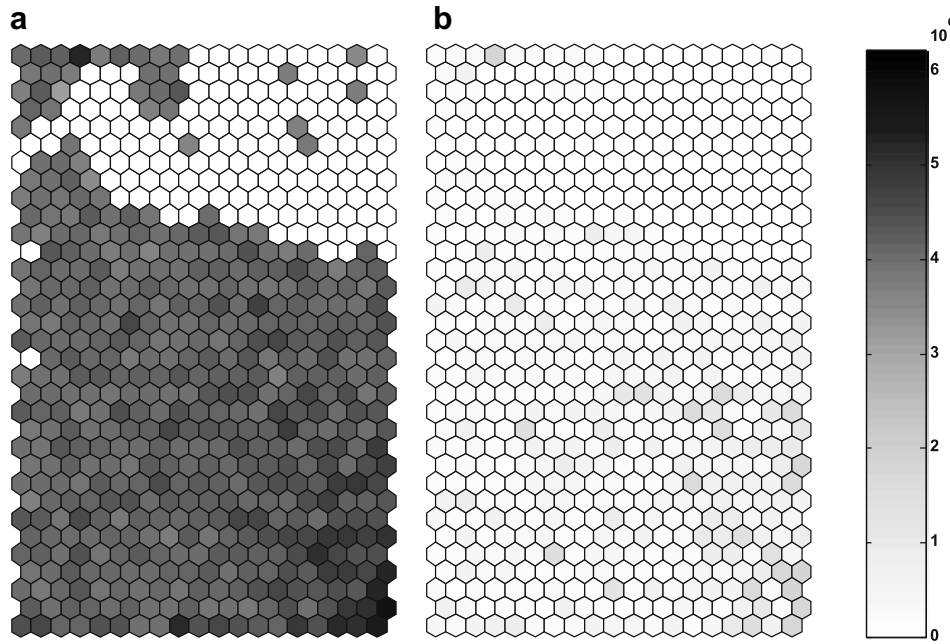


Fig. 8. Q-function in a SOM topology for the SOM-RL policy (a) and for the company policy (b). The darker the colour, the higher the value of the action-value function $Q^\pi(s, a)$. The SOM-RL policy provides much larger profits than the company policy.

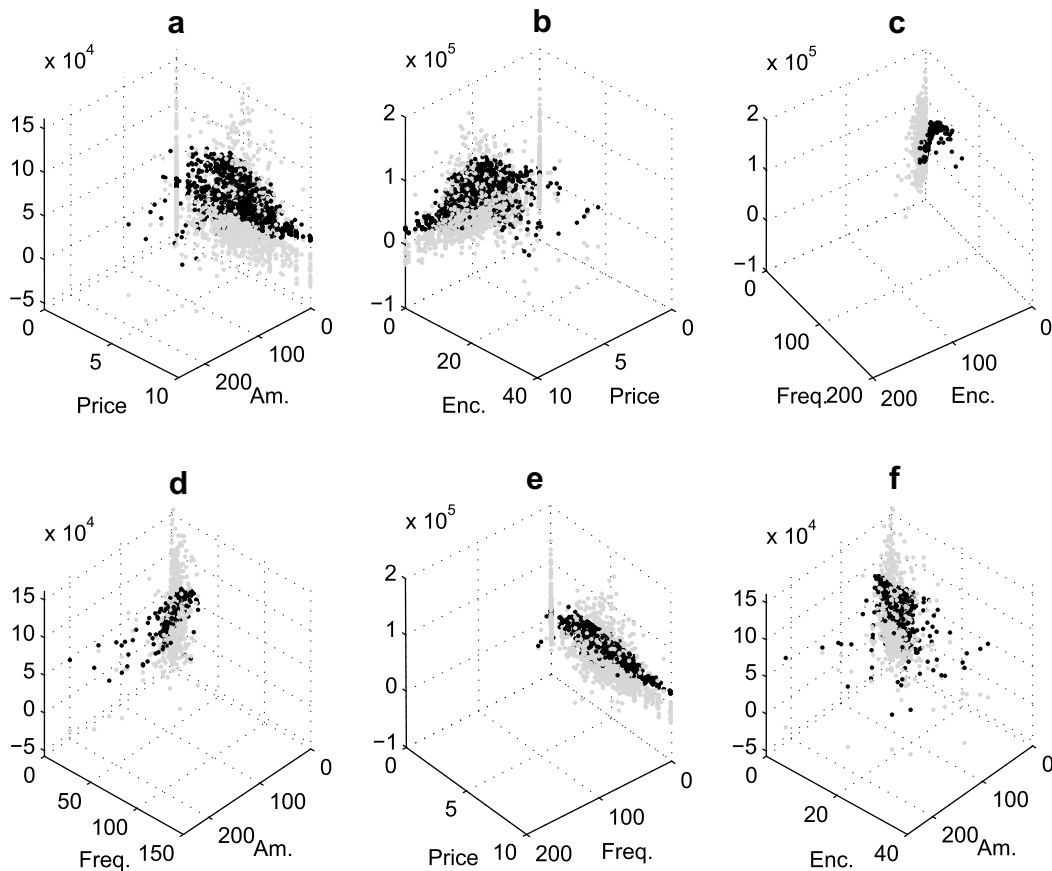


Fig. 9. Q-function for the MLP approach. Gray dots stand for the profits expected if the company policy is followed, whereas black dots correspond to the MLP-RL policy values. The vertical axis represents the value of the Q-function vs. a horizontal plane defined by two characteristics of the state space in each plot, namely, “price” and “amount of items purchased” in (a); “encouraged” and “price” in (b); “frequency” and “encouraged” in (c); “frequency” and “amount” in (d); “price” and “frequency” in (e); “encouraged” and “amount” in (f).

The MLP regressor is more powerful because the generalization process is more robust than in the SOM algorithm. This could help to obtain a more accurate policy in states which are not very frequent but are especially relevant to the company. In our study, since the input space had many dimensions, the presentation of the results was quite a bit more difficult than the SOM approach.

Future work will be devoted to the improvement of the state space characterization and to the use of other regression algorithms. The first would be supported by marketing studies to analyse how to better understand the customer behaviour; the second could involve the use of other artificial intelligence methods for function approximation, such as support vector regressors or fuzzy systems.

Acknowledgements

This work has been partially supported by the Spanish Ministry of Science and Education under Projects TIN2007-61006 and CSD2007-00018, and by the Valencian Regional Government (*Generalitat Valenciana*) under Projects GVA05/009 and ARVIV/2007/094. The authors would like to express their thanks to Dr. Faustino Gomez (Istituto Dalle Molle di Studi sull'Intelligenza Artificiale, Manno, Switzerland) for reviewing this paper, and improving it with his comments.

References

- Abe, N., Pednault, E., Wang, H., Zadrozny, B., Wei, F., Apte, C. (2002). Empirical comparison of various reinforcement learning strategies for sequential targeted marketing. In *Proceedings of the ICDM* (pp. 3–10).
- Abe, N., Verma, N., Schroko, R., Apte, C. (2004). Cross channel optimized marketing by reinforcement learning. In *Proceedings of the KDD* (pp. 767–772).
- Baird, L., Moore, A. (1999). Gradient descent for general reinforcement learning. In *Proceedings of the 1998 conference on advances in neural information processing systems II* (pp. 968–974). Cambridge, MA, USA: MIT Press.
- Bertsekas, D. (2001). *Dynamic programming and optimal control* (Vol. 1). Belmont, MA, USA: Athena Scientific.
- Bertsekas, D. P., Borkar, V. S., Nedi, A. (2003). *Improved temporal difference methods with linear function approximation*. Tech. Rep. LIDS-P-2573. MIT.
- Bertsekas, D., & Tsitsiklis, J. (1996). *Neuro-dynamic programming*. Belmont, MA, USA: Athena Scientific.
- Bishop, C. (1995). *Neural networks for pattern recognition*. Oxford, United Kingdom: Oxford University Press.
- Haykin, S. (1999). *Neural networks – a comprehensive foundation* (2nd ed.). Upper Saddle River, NJ, USA: Prentice Hall.
- Kakade, S. M. (2003). *On the sample complexity of reinforcement learning*. Ph.D. thesis. Gatsby computational neuroscience unit. University College London.
- Kohonen, T. (2001). *Self-organizing maps* (3rd ed.). Berlin, Germany: Springer.
- Pednault, E., Abe, N., Zadrozny, B. (2002). Sequential cost-sensitive decision making with reinforcement learning. In *Proceedings of the eighth ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 259–268). ACM.
- Pfeifer, P. E., & Carraway, R. L. (2000). Modelling customer relationships as markov chains. *Journal of Interactive Marketing*, 14(2), 43–55.
- Reichheld, F. F. (2001). *The loyalty effect: The hidden force behind growth, profits, and lasting value*. Boston, MA, USA: Harvard Business School Press.
- Smith, A. J. (2002). Applications of the self-organising map to reinforcement learning. *Neural networks*, 15(8–9), 1107–1124.
- Sun, P. (2003). *Constructing learning models from data: The dynamic catalog mailing problem*. Ph.D. thesis. Tsinghua University.
- Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. Cambridge, MA, USA: MIT Press.
- Sutton, R., McAllester, D., Singh, S., Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of advances in neural information processing systems 12* (pp. 1057–1063). The MIT Press.