A Q-Learning Approach to Derive Optimal Consumption and Investment Strategies

Alex Weissensteiner

***Abstract*—In this paper, we consider optimal consumption and strategic asset allocation decisions of an investor with a finite planning horizon. A Q-learning approach is used to maximize the expected utility of consumption. The first part of the paper presents conceptually the implementation of Q-learning in a discrete state-action space and illustrates the relation of the technique to the dynamic programming method for a simplified setting. In the second part of the paper, different generalization methods are explored and, compared to other implementations using neural networks, a combination with self-organizing maps (SOMs) is proposed. The resulting policy is compared to alternative strategies.**

***Index Terms*—Asset allocation, dynamic programming, finance, Q-learning, reinforcement learning, self-organizing maps (SOMs).**

I. Introduction

ONE of the classical problems in finance is to derive optimal consumption and asset allocation strategies for an investor with a finite planning horizon. Early work traces back to the pioneering papers of [33], [27], and [28] who focus on analytical tractability. Since then, a range of different methods has been proposed in the literature.

Using dynamic programming techniques, one group of papers extends the classical Merton framework along different lines, e.g., labor income or time-varying investment opportunities. Closed-form solutions can be derived only for special cases (see, e.g., [4], [23], and [40]). Another group proposes approaches which are not limited in this strict way to analytical tractability. Their results are exact for a broader set of problem settings and give approximations in their neighborhood (see, e.g., [8], [9], and [11]). For example, Campbell *et al.* [8] consider Epstein–Zin utility and an infinite planning horizon together with a first-order vector autoregression for the evolution of asset returns and state variables. Excluding short sale constraints on the asset allocation, they get an exact solution for the case of unit elasticity of intertemporal consumption.

To overcome these restrictions of analytical models, various numerical methods have been proposed. One approach works via a finite-difference approximation on a grid (see, e.g., [1], [7], [13], and [18]) to reduce the state–space dimension and solve the problem by backward induction. Others, such as Detemple *et al.* [15] and Brandt *et al.* [6] use simulation-based approaches. While Detemple *et al.* [15] propose a simulation-based method to approximate deviations from a closed-form solution, Brandt *et al.* [6] combine Monte Carlo simulation and regression techniques, inspired by Longstaff and Schwartz [25]. Finally, also stochastic linear programming (SLP), with many successful applications in asset-liability management problems (see, e.g., [14], [17], [19], and [43]), has been proposed to derive optimal consumption and investment decisions in the expected utility framework by [16]. The main advantage of this method is its ability to handle large scale problems with many assets and many constraints.

However, all these techniques require the specification of the dynamics describing the uncertainty of the future, which drives asset returns and state variables. For example, in the SLP approach, to maintain computational tractability, the multivariate distribution of the process is approximated with a few mass points (nodes) (see, e.g., [20], [21], and [31]). Given this scenario tree optimal decisions are calculated for each node. There are at least two conceptual problems when using this approach in practical applications (for a discussion, see, e.g., [14]). First, after one period, the realized returns and state variables are unlikely to coincide exactly with the values in scenario tree. Second, the stochastic properties of the process may change unexpectedly over time. To mitigate these problems, the SLP literature proposes a rolling-forward approach, where in every stage the new parameters of the stochastic process are estimated and a new scenario tree is generated in order to solve the optimization task again.[1]

In this paper, we explore the implementation of Q-learning [41], [42] to derive optimal consumption and asset allocation decisions in a finite-horizon model. Compared to the methods mentioned above, Q-learning can handle time- and path-dependent utility functions and does not require any assumption about the stochastic process. Instead of the rolling-forward approach used, e.g., in the SLP literature, this "online" learning technique allows for continuous adaptation to a changing environment. Like other numerical approaches, Q-learning also suffers from the curse of dimensionality. Therefore, for practical applications, it must be combined with generalization methods. In the second part of this paper, we explore function approximation methods (see, e.g., [10], [29], [30], and [39]) and present a combination with self-organizing maps (SOMs) [34], [36]. The finite-horizon structure of the problem poses some challenges, as for each decision stage a different map has to be created.

The paper is organized as follows. Section II presents the problem formulation and relates the optimization task to the well-known dynamic programming method. Section III studies

Manuscript received August 04, 2008; revised January 27, 2009; accepted April 05, 2009. First published June 02, 2009; current version published August 05, 2009.This work was supported by the Austrian Nationalbank under Jubiläumsfondsprojekt 13054.

The author is with the Department of Banking and Finance, University of Innsbruck, 6020 Innsbruck, Austria (e-mail: alex.weissensteiner@uibk.ac.at).

Digital Object Identifier 10.1109/TNN.2009.2020850

[1]A more comprehensive comparison with other numerical methods is very demanding and beyond the objective of this paper.

conceptually the application of the Q-learning algorithm for consumption and asset allocation decisions in a finite-horizon context. In Section IV, we give an illustrative example for a simplified task in a discrete state-action space. The general extension to function approximations is explored in Section V, while a combination of Q-learning with SOMs is proposed in Section VI. In Section VII, results are compared to a case where a closed-form solution is available. Section VIII concludes.

II. Problem Formulation

The objective is to maximize expected utility over a finite planning horizon T by taking into account optimal consumption and investment decisions at discrete points in time τ (called stages) from $\tau = t$ (now) to $\tau = T-1$. In $T-1$, the last consumption and investment decision is made by the individual as at the last stage T the individual consumes the remaining wealth completely. The optimization task can be formalized in terms of the value function

$$J_t^*(W_t) = \max_{\substack{\alpha_{c,\tau}\in\mathcal{A}_{c,\tau}\\ \alpha_{a,\tau}\in\mathcal{A}_{a,\tau}}} \mathrm{E}_{\varepsilon_\tau}\left\{\sum_{\tau=t}^{T} e^{-\delta(\tau-t)} U(W_\tau \alpha_{c,\tau}) \right\} \quad \tau\in[t,T-1] \tag{1}$$

where U is a time-additive, strictly concave von Neumann–Morgenstern utility function, δ is the time preference rate (for the sake of notation brevity set in the following equal to zero), and W_t is the total amount of wealth denominated in units of the consumption good, which serves as the numeraire. $\alpha_{c,\tau}$ and $\alpha_{a,\tau}$ refer to the two control variables available at the stages τ, at which the individual has to determine the consumption rate $\alpha_{c,\tau}$ and the asset allocation vector $\alpha_{a,\tau}$ for the remaining financial wealth. The admissible action sets are represented by $\mathcal{A}_{c,\tau}$ and $\mathcal{A}_{a,\tau}$. The uncertainty ε_τ in the model results from the possibility to invest wealth not only in a safe asset, but also in risky assets. This leads to an $\mathcal{F}_\tau$-adapted process of wealth W_τ. The expectation E in (1) is with respect to the multivariate distribution of the random variables ε_τ. The corresponding Bellman equation for this optimization task is given by (see, e.g., [27])

$$J_t^*(W_t) = \max_{\substack{\alpha_{c,t}\in\mathcal{A}_{c,t}\\ \alpha_{a,t}\in\mathcal{A}_{a,t}}} \mathrm{E}_{\varepsilon_t}\left\{U_t(W_t\alpha_{c,t}) + J_{t+1}^*\left(f_t(W_t,\alpha_{c,t},\alpha_{a,t},\varepsilon_t)\right)\right\} \tag{2}$$

where the migration function f_t describes the transition from the combination $(W_t,\alpha_{c,t},\alpha_{a,t},\varepsilon_t)$ at stage t to the wealth W_{t+1} at $t+1$, i.e., the migration from a given state and given actions to a new state at the following stage when the uncertainty ε_t is resolved.

III. Q-Learning

Q-learning is a reinforcement learning algorithm [41], [42], which calculates the optimal policy in a time-, state-, and action-discrete environment. Like dynamic programming, Q-learning is a *bootstrapping* method, as the update of estimates in τ is based on estimates in $\tau+1$. As a distinctive feature, however, reinforcement learning algorithms are simulation based, i.e., they use training information to evaluate the actions taken. Nevertheless, there is a strong interrelation between the Q-value of the optimal policy and the value function J_t^*.

Definition 1: The Q-value of the optimal policy $Q_\tau^*(W_\tau,\alpha_{c,\tau},\alpha_{a,\tau})$ is defined as the expected utility of choosing the (not necessary optimal) control variables $\alpha_{c,\tau}$ and $\alpha_{a,\tau}$ with given wealth W_τ at stage τ and following the optimal policy afterwards.

In this way, the Q-function maps state-action combinations to expected utility

$$Q_\tau^* : (\mathcal{A}_{c,\tau}\times\mathcal{A}_{a,\tau})\times\mathcal{W}_\tau \to \mathbb{R}. \tag{3}$$

To see the interrelation between the Q-function and the value function of the optimal policy, compare (4) and (5)

$$Q_\tau^*(W_\tau,\alpha_{c,\tau},\alpha_{a,\tau}) = \mathrm{E}_{\varepsilon_\tau}\left\{U_\tau(W_\tau\alpha_{c,\tau}) + J_{\tau+1}^*\left(f_\tau(W_\tau,\alpha_{c,\tau},\alpha_{a,\tau},\varepsilon_\tau)\right)\right\} \tag{4}$$

$$J_\tau^*(W_\tau) = \max_{\substack{\alpha_{c,\tau}\in\mathcal{A}_{c,\tau}\\ \alpha_{a,\tau}\in\mathcal{A}_{a,\tau}}} \mathrm{E}_{\varepsilon_\tau}\left\{U_\tau(W_\tau\alpha_{c,\tau}) + J_{\tau+1}^*(f_\tau(W_\tau,\alpha_{c,\tau},\alpha_{a,\tau},\varepsilon_\tau))\right\}. \tag{5}$$

From the above equations, one can verify that the Q-function of the optimal policy Q_τ^* corresponds to the value function of the optimal policy (see, e.g., [35, p. 76])

$$J_\tau^*(W_\tau) = \max_{\substack{\alpha_{c,\tau}\in\mathcal{A}_{c,\tau}\\ \alpha_{a,\tau}\in\mathcal{A}_{a,\tau}}} Q_\tau^*(W_\tau,\alpha_{c,\tau},\alpha_{a,\tau}) \tag{6}$$

and the optimal control variables at stage τ are given by

$$\left.\begin{matrix}\alpha_{c,\tau}^*\\ \alpha_{a,\tau}^*\end{matrix}\right\} = \arg\max_{\substack{\alpha_{c,\tau}\in\mathcal{A}_{c,\tau}\\ \alpha_{a,\tau}\in\mathcal{A}_{a,\tau}}} Q_\tau^*(W_\tau,\alpha_{c,\tau},\alpha_{a,\tau}). \tag{7}$$

Inserting the corresponding (6) for stage $\tau+1$ into (4) results in (8) shown at the bottom of the page.

Q-learning, as a simulation-based method, does not require all possible trajectories and the corresponding probabilities to solve the control task recursively, but estimates the Q-values $Q_\tau^*(W_\tau,\alpha_{c,\tau},\alpha_{a,\tau})$ by an incremental learning approach. This can be done either by simulating single trajectories from t to T or by using real market data (see, e.g., [10]). In the second case, there is no need to specify a stochastic process or to estimate its parameters (see, e.g., [22]). The estimated Q-values $\widehat{Q}_\tau^*(W_\tau,\alpha_{c,\tau},\alpha_{a,\tau})_{\overline{n}}$ for the state-action vector $(W_\tau,\alpha_{c,\tau},\alpha_{a,\tau})$ are maintained in a lookup table and updated

$$Q_\tau^*(W_\tau,\alpha_{c,\tau},\alpha_{a,\tau}) = \mathrm{E}_{\varepsilon_\tau}\left\{U_\tau(W_\tau\alpha_{c,\tau}) + \max_{\substack{\alpha_{c,\tau+1}\in\mathcal{A}_{c,\tau+1}\\ \alpha_{a,\tau+1}\in\mathcal{A}_{a,\tau+1}}} Q_{\tau+1}^*\left(f_\tau(W_\tau,\alpha_{c,\tau},\alpha_{a,\tau},\varepsilon_\tau),\alpha_{c,\tau+1},\alpha_{a,\tau+1}\right)\right\}. \tag{8}$$

on occurrence of new experiences i, $i = 1, \ldots, n$, on this specific combination (see, e.g., [2]) using the incremental updating formula

$$\widehat{Q}^*_\tau(\mathbf{x})_{\overline{n+1}} = \widehat{Q}^*_\tau(\mathbf{x})_{\overline{n}} + \lambda_n \left[\widehat{Q}^*_\tau(\mathbf{x})_{n+1} - \widehat{Q}^*_\tau(\mathbf{x})_{\overline{n}}\right] \quad (9)$$

where the vector $\mathbf{x}$ replaces $(W_\tau, \alpha_{c,\tau}, \alpha_{a,\tau})$ and λ_n indicates the learning step size. Convergence of the estimated Q-values $\widehat{Q}^*_\tau(W_\tau, \alpha_{c,\tau}, \alpha_{a,\tau})_{\overline{n}}$ to the optimal Q-values $Q^*_\tau(W_\tau, \alpha_{c,\tau}, \alpha_{a,\tau})$ in (8) would be guaranteed by the law of large numbers, visiting all state-action pairs infinitely often $(n \to \infty)$ [42] and by ensuring that λ_n satisfies (see [37])

$$\sum_{n=1}^{\infty} \lambda_n = \infty \quad \text{and} \quad \sum_{n=1}^{\infty} (\lambda_n)^2 < \infty. \quad (10)$$

In this way, also the actions for the learned policy will converge to their optimal values; see (7). In Section IV, we set $\lambda_n = 1/n$, satisfying the conditions (10).

Rearranging the ideas and formulas above, we construct the Q-learning algorithm for a consumption and asset allocation strategy (Algorithm 1).

Algorithm 1: Q-learning for a consumption and investment strategy

1: Initialize $\widehat{Q}^*_\tau(W_\tau, \alpha_{c,\tau}, \alpha_{a,\tau})$ arbitrarily (e.g., with 0) $\forall\, (W_\tau, \alpha_{c,\tau}, \alpha_{a,\tau})$

2: **for** trajectory $= 1$ to (number of trajectories) **do**

3: Initialize W_t with the initial wealth

4: **for** $\tau = t$ to $T - 1$ **do**

5: Choose a policy π to select $\alpha_{c,\tau}, \alpha_{a,\tau}$ from $\widehat{Q}^*_\tau(W_\tau, \alpha_{c,\tau}, \alpha_{a,\tau})$ (e.g., ϵ-greedy policy)

6: $W_{\tau+1} = f_\tau(W_\tau, \alpha_{c,\tau}, \alpha_{a,\tau}, \varepsilon_\tau)$

7: **if** $\tau < T - 1$ **then**

8:

$$\widehat{Q}^*_\tau(\mathbf{x}_\tau)_{\overline{n+1}} \leftarrow \widehat{Q}^*_\tau(\mathbf{x}_\tau)_{\overline{n}} + \lambda_{n_{(\mathbf{x}_\tau)}} \left[U_\tau(W_\tau \alpha_{c,\tau}) + \max_{\substack{\alpha_{c,\tau+1} \in \mathcal{A}_{c,\tau+1} \\ \alpha_{a,\tau+1} \in \mathcal{A}_{a,\tau+1}}} \widehat{Q}^*_{\tau+1}(\mathbf{x}_{\tau+1}) - \widehat{Q}^*_\tau(\mathbf{x}_\tau)_{\overline{n}} \right]$$

9: **else**

10:

$$\widehat{Q}^*_\tau(\mathbf{x}_\tau)_{\overline{n+1}} \leftarrow \widehat{Q}^*_\tau(\mathbf{x}_\tau)_{\overline{n}} + \lambda_{n_{(\mathbf{x}_\tau)}} \left[U_\tau(W_\tau \alpha_{c,\tau}) + U_T(W_T) - \widehat{Q}^*_\tau(\mathbf{x}_\tau)_{\overline{n}} \right]$$

11: **end if** {with $\mathbf{x}_\tau$ for $(W_\tau, a_{c,\tau}, a_{a,\tau})$}

12: **end for**

13: **end for**

Line 8 shows the update of $\widehat{Q}^*_\tau(\mathbf{x})_{\overline{n+1}}$, which is based on the utility of current consumption $U_\tau(W_\tau \alpha_{c,\tau})$ and on the estimated value of the optimal policy in state $W_{\tau+1}$ by

$$\max_{\substack{\alpha_{c,\tau+1} \in \mathcal{A}_{c,\tau+1} \\ \alpha_{a,\tau+1} \in \mathcal{A}_{a,\tau+1}}} \widehat{Q}^*_{\tau+1}(W_{\tau+1}, \alpha_{c,\tau+1}, \alpha_{a,\tau+1})$$

[see also (6)]. These two determine the new estimation (or target) $n+1$ for $\widehat{Q}^*_\tau(\mathbf{x})_{\overline{n+1}}$ in (9) and indicate the direction in which to move.

IV. Results in the Discrete State-Action Space

In the following illustrative example, we present a simplified task, where an individual with a time-additive, strictly concave von Neumann–Morgenstern utility function and initial wealth W_t wants to take optimal decisions from a set of discrete consumption and investment possibilities in order to maximize his expected utility over the planning horizon; see (1). We assume that the market offers two investment opportunities, a risk-free asset and a risky asset. To model the uncertainty, we further assume that at each stage τ two states of nature $\Omega = \{\omega_u, \omega_d\}$ can occur with probabilities $\mathcal{P} = \{p_u, p_d\}$, in which the risky asset will realize the returns r_u or r_d. In this way, the process of the risky asset corresponds to a binomial tree. To avoid dominance and the ensuing arbitrage opportunities, we set $r_d < r_f < r_u$, where r_f represents the return of the risk free asset. The total number of Q-values $\widehat{Q}^*_\tau(W_\tau, \alpha_{c,\tau}, \alpha_{a,\tau})$ to estimate and update can easily be calculated by

$$n_{\widehat{Q}} = \sum_{\tau=1}^{T-t} (n_{\alpha_c} \times n_{\alpha_a})^\tau \times 2^{(\tau-1)} \quad (11)$$

where n_{α_c} and n_{α_a} are the number of discrete consumption and investment possibilities at each stage τ.

Fig. 1 illustrates the Q-learning process at a specific stage τ, grouping the Cartesian product of consumption and investment possibilities, i.e., $(\mathcal{A}_{c,\tau} \times \mathcal{A}_{a,\tau})$, on the x-axis of the graphs. In the example used here, we allow for three consumption $(n_{\alpha_c} = 3)$ and three investment $(n_{\alpha_a} = 3)$ possibilities, resulting in a total of nine action combinations for each discrete level of wealth W_τ. On the y-axis, we plot the wealth W_τ, while the z-axis shows the corresponding Q-values. The estimated Q-value for the optimal policy $\widehat{Q}^*_\tau(W_\tau, \alpha_{c,\tau}, \alpha_{a,\tau})$ is indicated by a black bullet.

A total of 162 Q-values are tuned at this specific stage τ. During the learning process, each of them is updated. For a particular $(W_\tau, \alpha_{c,\tau}, \alpha_{a,\tau})$-combination, these updates are illustrated in Fig. 2. In this way, the rather crumpled Q-surface in the top graph of Fig. 1 becomes considerably smoother. Optimal consumption and investment decisions are taken according to (7). Given wealth W_τ, the individual will choose the $(\alpha_{c,\tau}, \alpha_{a,\tau})$-combination with the highest estimated Q-value $\widehat{Q}^*_\tau(W_\tau, \alpha_{c,\tau}, \alpha_{a,\tau})$, i.e., the combination with the highest expected utility analogous to the value function of the optimal policy; see (6). This corresponds to a theoretical cut of the surface in Fig. 1 at the given wealth W_τ and in the search for the maximum in the resulting graph. Performing this calculation for all discrete wealth levels and plotting the resulting

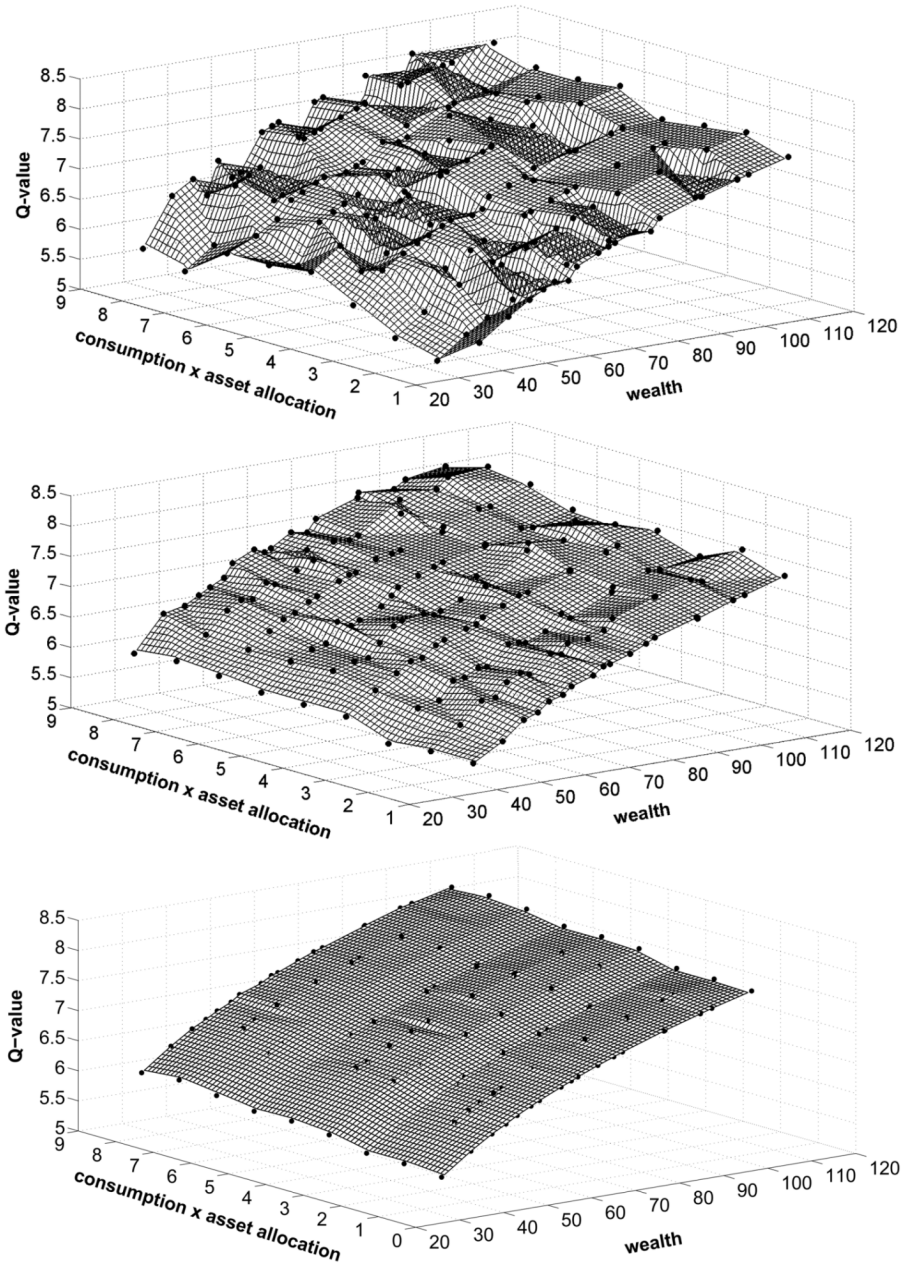

Fig. 1. Q-learning process at a specific stage τ. The graphic illustrates the Q-learning process, grouping the Cartesian product of consumption and investment possibilities on the x-axis, the wealth on the y-axis, and the Q-values on the z-axis. Each estimated Q-value for the optimal policy $\widehat{Q}^*_\tau(W_\tau, \alpha_{c,\tau}, \alpha_{a,\tau})$ is indicated by a black bullet. During the learning process, they are updated, smoothing the Q-surface in the top of the figure.

maximum Q-values in Fig. 3, we get a (discrete) approximation of the value function of the optimal policy at stage τ.

Next, we study the sensitivity of the Q-values $Q^*_\tau(W_\tau, \alpha_{c,\tau}, \alpha_{a,\tau})$ with respect to different consumption/investment combinations at stage τ. The results are plotted in Fig. 4. As can be seen from the scaling of the axes, for the chosen stage τ, the expected utility is much more sensitive to changes in consumption than to the investment decision (by a factor of ten).

V. Function Approximation

Section IV presented an example for a discrete state-action space. For the sake of computational tractability, we restricted the problem in two ways: the process of the risky asset was modeled by a binomial tree and the sets of consumption and investment possibilities were each limited to three elements, as the computational complexity increases exponentially; see (11). For high-dimensional and continuous state-action spaces, some form of generalization is required [5]. Generalization methods assume that "similar" wealth at a given stage τ will lead to "similar" expected utility. In this way, a compact storage of learned information and a transfer of knowledge between "similar" states and actions can be established [22]. A classical form of generalization is function approximation.

Instead of saving all possible combinations at stage τ in a lookup table, the Q-values are calculated by a function. For this

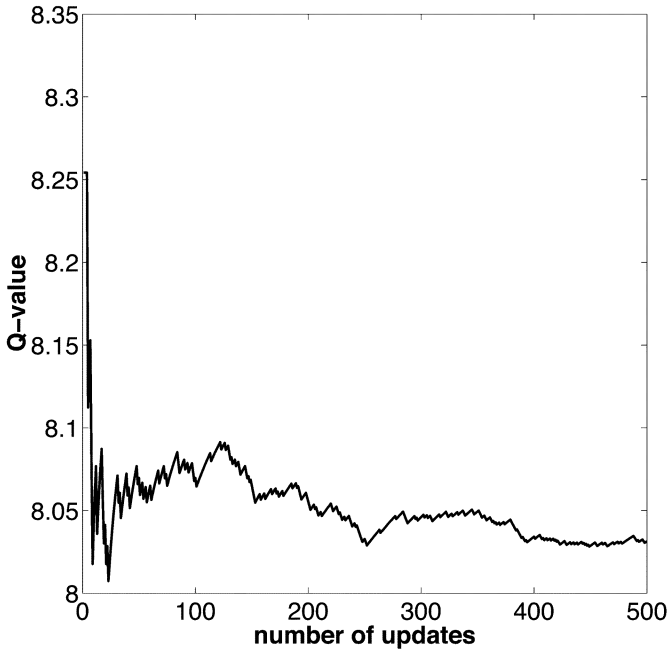

Fig. 2. Approximation of a particular Q-value $\widehat{Q}^*_\tau(W_\tau, \alpha_{c,\tau}, \alpha_{a,\tau})$. The x-axis indicates the number of updates, and the y-axis indicates the estimated Q-value for this state-action combination.

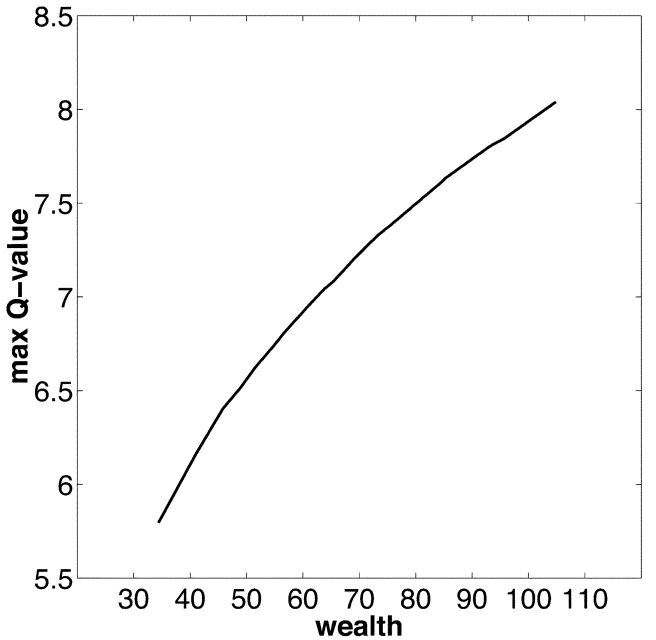

Fig. 3. Maximum of the optimal Q-values. The figure corresponds to a theoretical cut of the surface in Fig. 1 at a given wealth W_τ and in the search for the maximum in the resulting graph. It represents a discrete approximation of the value function of the optimal policy at stage τ.

approach, two steps are necessary. First, define a set of basis functions to calculate $\tilde{Q}_\tau(W_\tau, \alpha_{c,\tau}, \alpha_{a,\tau}, \boldsymbol{\theta}_\tau)$. The intuition or understanding of a user in providing this set of basis functions is the key to unlock the complexity of the underlying decision problem [32]. Second, after defining the functional form, estimate (or "learn") the parameter vector $\boldsymbol{\theta}_\tau = (\theta_{\tau,1}, \ldots, \theta_{\tau,m})^\top$.[2] As the number of parameters is normally much lower than the

[2]Note that for the finite horizon of the problem setting this vector is stage dependent.

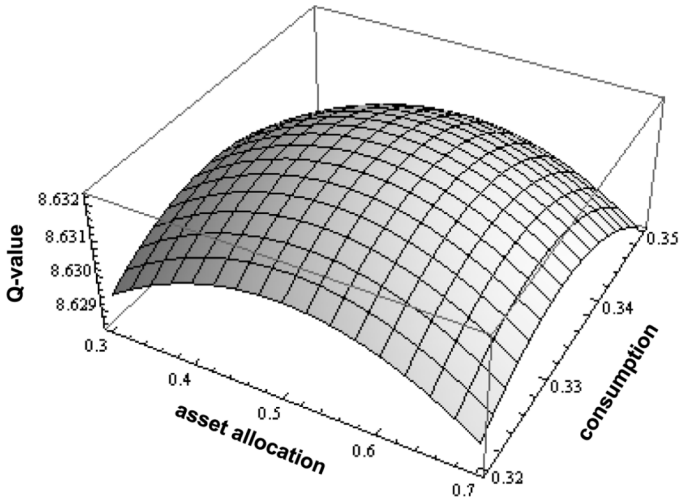

Fig. 4. Q-function sensitivity. As can be seen from the scaling of the axes, for the chosen stage τ, the expected utility is much more sensitive to changes in consumption than to the investment decision (by a factor of ten).

number of possible state-action combinations, changing one parameter will result in a modification of the expected utility of many (all) state-action combinations. This function is then used to derive optimal consumption and investment decisions, analogous to (7)

$$\left.\begin{matrix}\tilde{\alpha}^*_{c,\tau} \\ \tilde{\alpha}^*_{a,\tau}\end{matrix}\right\} = \underset{\substack{\alpha_{c,\tau}\in\mathcal{A}_{c,\tau} \\ \alpha_{a,\tau}\in\mathcal{A}_{a,\tau}}}{\arg\max}\ \tilde{Q}^*_\tau(W_\tau, \alpha_{c,\tau}, \alpha_{a,\tau}, \boldsymbol{\theta}_\tau). \tag{12}$$

The parameter vector $\boldsymbol{\theta}_\tau$ should be tuned in such a way that the estimation error between the approximated function $\tilde{Q}_\tau(W_\tau, \alpha_{c,\tau}, \alpha_{a,\tau}, \boldsymbol{\theta}_\tau)$ and their corresponding true value $Q^*_\tau(W_\tau, \alpha_{c,\tau}, \alpha_{a,\tau})$ is minimized. One measure to quantify this deviation is the mean squared error (MSE)

$$\mathrm{MSE}(\boldsymbol{\theta}_\tau) = \sum_{W_\tau\in\mathcal{W}_\tau} p(W_\tau)\Big[Q^*_\tau(W_\tau, \alpha_{c,\tau}, \alpha_{a,\tau}) - \tilde{Q}_\tau(W_\tau, \alpha_{c,\tau}, \alpha_{a,\tau}, \boldsymbol{\theta}_\tau)\Big]^2 \tag{13}$$

where $p(W_\tau)$ is the probability for wealth W_τ. Usually it is not possible to reduce the MSE for all state-action combinations considered, but instead some tradeoff is necessary. The probability weights reward small errors more in states with a higher probability of occurrence. However, finding a global minimum for the MSE (such that $\mathrm{MSE}\left(\boldsymbol{\theta}^*_\tau\right) < \mathrm{MSE}\ (\boldsymbol{\theta}_\tau)\ \forall\, \boldsymbol{\theta}_\tau$) is difficult. In most cases, a local optimum is the best we can get.

As the MSE given by (13) is a function dependent on the parameter vector $\boldsymbol{\theta}_\tau$, the gradient-descent method can be applied. Unfortunately, $Q^*_\tau(.)$ is not known ex-ante; this value is just being learned. Therefore, as an estimate for the true value $Q^*_\tau(.)$, the sum of $U_\tau(W_\tau\alpha_{c,\tau})$ and the maximum of $\tilde{Q}_{\tau+1}(\mathbf{x}_{\tau+1})$ is used

$$\boldsymbol{\theta}^{n+1}_\tau = \boldsymbol{\theta}^n_\tau + \lambda\left[U_\tau(W_\tau\alpha_{c,\tau}) + \max_{\substack{\alpha_{c,\tau+1} \\ \alpha_{a,\tau+1}}} \tilde{Q}_{\tau+1}(\mathbf{x}_{\tau+1}) - \tilde{Q}_\tau(\mathbf{x}_\tau)\right] \times \nabla_{\boldsymbol{\theta}_\tau}\tilde{Q}_\tau(\mathbf{x}_\tau) \tag{14}$$

where the vector $\mathbf{x}_\tau$ replaces $(W_\tau, \alpha_{c,\tau}, \alpha_{a,\tau}, \boldsymbol{\theta}_\tau)$. Calculating with an estimate instead of the true value, convergence is assured if and only if the estimator is unbiased [35, p. 198]. However, for bootstrapping methods such as Q-learning, this is not the case and convergence can no longer be guaranteed [38].

The method presented in this section reduces the complexity by the choice of a suitable parameterization of the function. To this end, some practical experience or theoretical analysis of the problem at hand is required [3]. However, while the concave shape of the value function is somewhat intuitive, guessing the functional form of the Q-values is not an easy task. Determining the functional interrelation between Q-value and combinations of wealth, consumption, and investment decisions requires a high degree of prior knowledge; the "correct" answer is known only after solving the problem. For this reason, we prefer to present a rather general approach in Section VI.

VI. Q-Learning With Self-Organizing Maps

In this section, we maintain the assumption that "similar" state-action combinations lead to "similar" results in terms of expected utility. Instead of saving and updating all discrete state-action combinations in a lookup table, "similar" combinations are represented by "hidden units" or neurons. In this case, as known from the literature, convergence can no longer be proved. Compared to an artificial neural network (ANN) with hidden layers (see, e.g., [10] and [39]), we follow the approach using SOMs (see, e.g., [26], [34], and [36]). Although ANNs have been successfully applied for different tasks, it is not unusual to criticize them as a nonlinear black-box model, which do not enhance the understanding of the problem at hand. SOMs are an effective tool for visualization and/or abstraction of high-dimensional data and have been very successfully applied in various fields [12]. A comparison of advantages and limitations of different neural network implementations when using Q-learning can be found in [36].[3]

The number, the positions, and the values of these neurons are not defined ex-ante, but created and updated during the learning process. This ensures an economical use of computer power: only the relevant neurons are updated and optimized with high frequency. A neuron $w(i_\tau)$ is representative for particular combinations $(W_\tau, \alpha_{c,\tau}, \alpha_{a,\tau})$ if the single elements are "near" the wealth and the actions of the neuron [24, p. 77]. To determine the distance, different measures are proposed in the literature, e.g., euclidean norm or radial basis functions (RBFs). In this paper, we focus on RBFs. Generally, given a neuron i with center c_i and dispersion r_i, the distance $\psi_s(i)$ to a state s is given by

$$\psi_s(i) = \exp\left(-\frac{|s - c_i|^2}{r_i^2}\right). \tag{15}$$

For our application, the distance of a particular state-action combination is calculated in the dimensions wealth W_τ, consumption $\alpha_{c,\tau}$, and investment $\alpha_{c,\tau}$ for all neurons. While the consumption rate and the investment decision (excluding short sales) take values between zero and one, neither the Q-values nor the wealth are restricted in this way. To compare and calculate the distance for each dimension, we normalize the last two values (from zero to one).[4] The following information vector is stored in a table for each neuron $w(i_\tau)$:

$$\left(w_W(i_\tau), w_{\alpha_c}(i_\tau), w_{\alpha_a}(i_\tau), w_Q(i_\tau), w_{W_n}(i_\tau), w_{Q_n}(i_\tau)\right) \tag{16}$$

where $w_W(i_\tau)$, $w_{\alpha_c}(i_\tau)$, and $w_{\alpha_a}(i_\tau)$ indicate the wealth, the consumption rate, and the investment decision of neuron $w(i_\tau)$, $w_Q(i_\tau)$ gives the corresponding, estimated Q-value, while the last two elements represent the normalized values.

The implementation of Q-learning using SOMs is illustrated in Algorithm 2 and the related Algorithms 3–6.

Algorithm 2: Q-learning using SOMs

1: Initialize: λ_Q, λ_α, λ_W, ϵ, r
2: **for** trajectory $= 1$ to (number of trajectories) **do**
3: Initialize W_t
4: **for** $\tau = t$ to $T - 1$ **do**
5: **if** trajectory $== 1$ **then**
6: **step 0**: create new neurons (Algorithm 3)
7: **else**
8: **step 1**: choose $\alpha_{c,\tau}$, $\alpha_{a,\tau}$ (Algorithm 4)
9: **step 2**: $u_\tau \leftarrow U(W_\tau \alpha_{c,\tau})$, $W_{\tau+1} \leftarrow f_\tau(W_\tau, \alpha_{c,\tau}, \alpha_{c,\tau}, \varepsilon_\tau)$
10: **step 3**: calculate Q-value (Algorithm 5)
11: **step 4**: update representative neuron (Algorithm 6)
12: **end if**
13: **end for**
14: **end for**

Algorithm 3: step 0—create new neurons

1: $\alpha_{c,\tau} \leftarrow$ RND, $\alpha_{c,\tau} \leftarrow$ RND
2: $W_{\tau+1} \leftarrow f_\tau(W_\tau, \alpha_{c,\tau}, \alpha_{c,\tau}, \varepsilon_\tau)$
3: **if** $\tau < T - 1$ **then**
4: $Q_\tau \leftarrow U(W\alpha_{c,\tau})$
5: **else**
6: $Q_\tau \leftarrow U(W\alpha_{c,\tau}) + U(W_T)$
7: **end if**
8: $w(n_\tau) \leftarrow [W_\tau, \alpha_{c,\tau}, \alpha_{c,\tau}, Q_\tau]$

[3]The author illustrates that Q-learning combined with SOMs improves in terms of memory requirements, learning time, and performance of the learned policy (compared to other approaches with fixed, *a priori* specified, structures of the neuronal networks).

[4]If short sales are not restricted, we also normalize the investment decisions.

Algorithm 4: step 1—choose $\alpha_{c,\tau}$, $\alpha_{a,\tau}$

1: $(\psi_{W,Q}(i_\tau^*), i_\tau^*) = \mathrm{SEEK}_{W,Q}(W_\tau)$

2: **if** $\psi_{W,Q}(i_\tau^*) < \epsilon$ **then**

3: $\alpha_{c,\tau} \leftarrow \mathrm{RND}$, $\alpha_{c,\tau} \leftarrow \mathrm{RND}$, $Q_\tau = 0$

4: $w(l_\tau) \leftarrow [W_\tau, \alpha_{c,\tau}, \alpha_{c,\tau}, Q_\tau]$

5: **else**

6: $\alpha_{c,\tau} \leftarrow w_{\alpha_c}(i_\tau^*) + \langle \mathrm{RND} \rangle$, $\alpha_{a,\tau} \leftarrow w_{\alpha_a}(i_\tau^*) + \langle \mathrm{RND} \rangle$

7: **end if**

Algorithm 5: step 3—calculate Q-value

1: **if** $\tau < T-1$ **then**

2: $\left(\psi_{W,Q}(j_{\tau+1}^*), j_{\tau+1}^*\right) = \mathrm{SEEK}_{W,Q}(W_{\tau+1})$

3: $q \leftarrow w_Q\left(j_{\tau+1}^*\right)$

4: **else**

5: $q \leftarrow U(W_T)$

6: **end if**

7: $Q_\tau = u_\tau + q$

Algorithm 6: step 4—update representative neuron

1: $(\psi_{W,\alpha}(k_\tau^*), k_\tau^*) = \mathrm{SEEK}_{W,\alpha}(\mathbf{x})$ with $\mathbf{x} = (W_\tau, \alpha_{c,\tau}, \alpha_{a,\tau})$

2: **if** $\psi_{W,\alpha} < \epsilon$ **then**

3: $w(m_\tau) \leftarrow [W_\tau, \alpha_{c,\tau}, \alpha_{c,\tau}, Q_\tau]$

4: **else**

5: $w_Q(k_\tau^*) \leftarrow w_Q(k_\tau^*) + \lambda_Q[Q_\tau - w_Q(k_\tau^*)]$

6: $w_W(k_\tau^*) \leftarrow w_W(k_\tau^*) + \lambda_W[W_\tau - w_W(k_\tau^*)]$

7: $w_{\alpha_c}(k_\tau^*) \leftarrow w_{\alpha_c}(k_\tau^*) + \lambda_\alpha[\alpha_{c,\tau} - w_{\alpha_c}(k_\tau^*)]$

8: $w_{\alpha_a}(k_\tau^*) \leftarrow w_{\alpha_a}(k_\tau^*) + \lambda_\alpha[\alpha_{a,\tau} - w_{\alpha_a}(k_\tau^*)]$

9: **end if**

After initializing the step size for learning λ,[5] the critical distance ϵ, and the dispersion r of the RBF, the algorithm starts the learning process by running different trajectories from t to $T-1$.

In the first simulation run, no neurons and no experience are available. Algorithm 3 shows that in this case consumption and investment decisions are chosen randomly, where "RND" represents a uniformly distributed variable with RND $\in [0,1]$. Given the decisions at stage τ and the general migration function, the wealth $W_{\tau+1}$ can be calculated. Due to the absence of knowledge, we assign to Q_τ the direct utility of consumption at state τ for $\tau < T-1$ and to Q_{T-1} the sum of utility of consumption at the stages $T-1$ and T. The vector $(W_\tau, \alpha_{c,\tau}, \alpha_{c,\tau}, Q_\tau)$ is then assigned to a new (the first) neuron $w(n_\tau)$ at stage τ.

After this initializing first simulation run the learning process starts. In Algorithm 4, we show how to choose consumption and investment decisions for a given wealth W_τ. The function $\mathrm{SEEK}_{W,Q}(W_\tau)$ combines normalized wealth and Q-values to search in the set of available neurons $\mathcal{I}_\tau$ the single neuron $w(i_\tau^*)$ that is close to the current wealth and shows a high Q-value, i.e., a high expected utility. The output arguments of this function are calculated according to

$$
\begin{aligned}
&i_\tau^* \\
&\quad = \arg\max_{i_\tau \in \mathcal{I}_\tau} \left(e^{-\left\{\left[(W_{n,\tau} - w_{W_n}(i_\tau))^2 + ((1 - w_{Q_n}(i_\tau))/2)^2\right]/r^2\right\}} \right) \\
&\psi_{W,Q}(i_\tau^*) \\
&\quad = \max_{i_\tau \in \mathcal{I}_\tau} \left(e^{-\left\{\left[(W_{n,\tau} - w_{W_n}(i_\tau))^2 + ((1 - w_{Q_n}(i_\tau))/2)^2\right]/r^2\right\}} \right).
\end{aligned}
\tag{17}
$$

If the distance measure $\psi_{W,Q}(i_\tau^*)$ is lower than the critical value, i.e., if no "near" neuron with a high Q-value is available, we act randomly, initialize Q_τ to zero, and assign these values to a new neuron $w(l_\tau)$. Otherwise, we use the actions imposed by neuron (i_τ^*). To ensure enough exploration, which is especially important at the beginning, we disturb these by some randomness $\langle \mathrm{RND} \rangle$ with mean zero and shrink its magnitude as the number of simulation runs increases.

Step 2 calculates the utility of consumption u_τ and the wealth for the next stage $W_{\tau+1}$. Step 3 determines the corresponding Q-value by using the function $\mathrm{SEEK}_{W,Q}(W_{\tau+1})$ for all $\tau < T-1$. If $\tau = T-1$, we assign the utility of consumption at the final stage T.

Finally, step 4 defines the update rule on how to generalize the experience of the particular state-action combination to a single neuron. For that purpose, the function $\mathrm{SEEK}_{W,\alpha}$ calculates the distance in the state-action space by the equations shown at the bottom of the page. The neuron k_τ^* is nearest to the particular combination $(W_\tau, \alpha_{c,\tau}, \alpha_{a,\tau})$, having a distance of $\psi_{W,\alpha}(k_\tau^*)$. If this distance is below the critical level ϵ, a new neuron m is created. Otherwise, the experience made is used to tune the neuron k_τ^*: the Q-value $w_Q(k_\tau^*)$, the wealth $w_W(k_\tau^*)$, and the

[5]We note that not only single, absolute values, but also their relation to each other is important for the learning success.

$$
\begin{aligned}
k_\tau^* &= \arg\max_{k_\tau \in \mathcal{K}_\tau} \left(e^{-\left\{\left[(W_{n,\tau} - w_{W_n}(k_\tau))^2 + (\alpha_{c,\tau} - w_{\alpha_c}(k_\tau))^2 + (\alpha_{a,\tau} - w_{\alpha_a}(k_\tau))^2\right]/r^2\right\}} \right) \\
\psi_{W,\alpha} &= \max_{k_\tau \in \mathcal{K}_\tau} \left(e^{-\left\{\left[(W_{n,\tau} - w_{W_n}(k_\tau))^2 + (\alpha_{c,\tau} - w_{\alpha_c}(k_\tau))^2 + (\alpha_{a,\tau} - w_{\alpha_a}(k_\tau))^2\right]/r^2\right\}} \right).
\end{aligned}
$$

decision variables $w_{\alpha_c}(k_\tau^*)$ and $w_{\alpha_a}(k_\tau^*)$ are updated toward $W_\tau, Q_\tau, \alpha_{c,\tau}$, and $\alpha_{a,\tau}$. After some tuning period, consumption and investment decisions can be selected according to neuron $w(i_\tau^*)$ in (17).

In order to speed up learning in Algorithm 2, we propose two further improvements. First, as the Q-value measures *expected* utility, we simulate more than one successor for the combination $(W_\tau, \alpha_{c,\tau}, \alpha_{a,\tau})$ and take the arithmetic mean of the resulting Q-values, instead of calculating step 2 and step 3 only for one $W_{\tau+1}$. This expected value is used in step 4 to update our knowledge. Second, as the Q-value and the wealth are interrelated (the higher the wealth, the higher the expected utility expressed in terms of the Q-value), we divide the set $\mathcal{J}_{\tau+1}$ in two subsets: $\mathcal{J}_{\tau+1}^d$ whose neurons have a wealth below $W_{\tau+1}$, and $\mathcal{J}_{\tau+1}^u$ with a higher wealth. In both sets, we use (17) to find a neuron nearby and interpolate linearly their Q-values to approximate the value of wealth $W_{\tau+1}$. Without such a correction, we would have a bias in the estimates: given two neurons equally distant from $W_{\tau+1}$ and both just tuned to the optimal policy, (17) will always choose the one with the higher wealth and the corresponding higher Q-value.

VII. Results Using Self-Organizing Maps

In the following, we study the application of the algorithm proposed in Section VI. In order to compare the performance of the learned policy with the *optimal* policy, we choose a simplified setting with four stages where a closed-form solution exists. Hence, we use log utility, a risk-free rate r_f set to 4%, and the risky asset following a geometric Brownian motion (GBM) with a drift (μ) of 8% and volatility (σ) of 25%. From the literature, we know that in this case the optimal consumption rate is given by the fraction $1/(T-\tau)$, i.e., 25% at the first stage, and the optimal asset allocation to the risky asset is myopic with $(\mu - r_f)/\sigma^2$, i.e., 64%.

For the learning period, we use 20 000 simulated trajectories and set the learning parameters by manual tuning to

$$\lambda_Q = 0.5 \quad \lambda_\alpha = 0.1 \quad \lambda_W = 0.04 \quad \epsilon = 0.15 \quad r = 0.1.$$

The resulting map for the first stage, starting with a financial wealth equal to 100, is given by 21 neurons. In Table I, we order these neurons i by their Q-value w_Q.[6] The last row indicates the neuron with the highest expected utility $w(i^*)$. As expected, the values of the control variables of this neuron $w_{\alpha_c}(i^*)$ and $w_{\alpha_a}(i^*)$ are near their analytical counterparts (25% and 64%). The last column gives the number of updates for each neuron i. The corresponding tuning process for $w(i^*)$ is illustrated in Fig. 5. In order to evaluate the performance of the learned strategy, we simulate a new set of 20 000 possible trajectories for the risky assets. We use this sample to compare the expected utility (J) given by the learned strategy to the expected utility of: 1) the analytical solution and 2) a naive strategy in which at all stages 50% of current wealth is consumed and 50% is invested in the risky asset. The results are indicated in Table II. As can be seen from the last column, the expected utility of our learned strategy is 0.69% below the optimal policy but outperforms the naive strategy by 4.84%. Given the flexibility of our approach, the results are considered as very promising. The performance can be improved by a longer learning period (more trajectories) and/or by tuning the indicated learning parameters above in the "right" direction. However, we defer a more thorough investigation of these issues to future work.

[6]At stage two and three, 84 and 163 neurons are created and tuned.

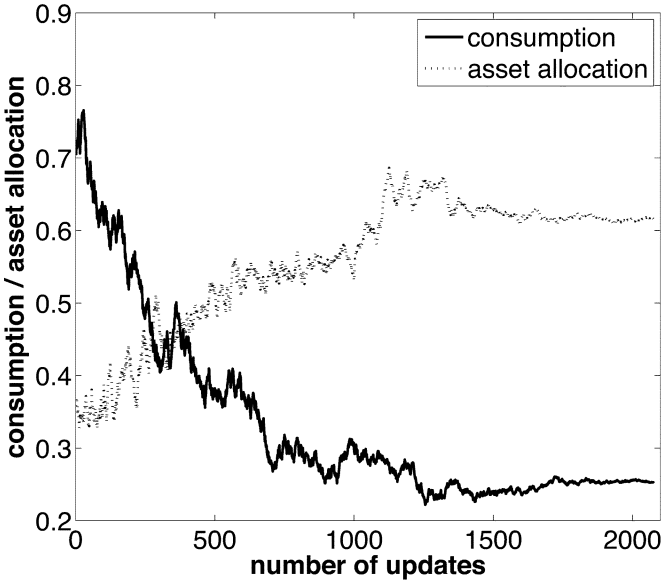

Fig. 5. Illustration of the tuning process for the neuron $w(i^*)$. The control variables for consumption $w_{\alpha_c}(i^*)$ and asset allocation $w_{\alpha_a}(i^*)$ can be compared to the known closed-form solution (consumption rate equal to 25% and asset allocation to the risky asset equal to 64%).

TABLE I
Self-Organized Map at Stage 1

i	w_W	w_{α_c}	w_{α_a}	w_Q	n
1	100	0.948	0.025	9.509	10
2	100	0.985	0.280	11.055	2
3	100	0.592	0.960	11.674	252
4	100	0.521	0.129	12.026	198
5	100	0.814	0.858	12.089	28
6	100	0.260	0.046	12.373	492
7	100	0.160	0.351	12.406	640
8	100	0.206	0.716	12.456	1024
9	100	0.234	0.566	12.466	1578
10	100	0.265	0.735	12.467	922
11	100	0.254	0.567	12.468	1694
12	100	0.251	0.427	12.469	626
13	100	0.244	0.673	12.469	1444
14	100	0.276	0.559	12.469	1027
15	100	0.248	0.541	12.470	1306
16	100	0.242	0.630	12.476	1383
17	100	0.269	0.596	12.478	946
18	100	0.278	0.641	12.483	1004
19	100	0.247	0.602	12.487	1831
20	100	0.264	0.621	12.487	1517
21	100	0.251	0.615	12.496	2077

TABLE II
Expected Utility Gain

strategy	J	deviation
self-organizing maps	13.099	–
analytical solution	13.190	-0.69%
naive strategy	12.494	4.84%

VIII. Conclusion and Outlook

The aim of this paper was to study conceptually the implementation of Q-learning for consumption and asset allocation

decisions. Many intuitive results were deduced from the figures in Section IV. The main advantages of Q-learning are as follows. First, time- and path-dependent utility functions, which normally are problematic for analytical tractability, can easily be incorporated in the model. Second, the possibility to use the algorithm without specifying the underlying stochastic process makes this technique a candidate for online learning with real market data, where little is known about the driving parameters and where a changing market environment is the rule rather than the exception.

In the illustrative example of Section IV, we have assumed that the estimates of the Q-values are represented by a table with one entry for each $(W_\tau, \alpha_{c,\tau}, \alpha_{a,\tau})$ combination. This is a clear and instructive case, but it is limited to a task with a small number of states and actions. The exponential growth of possible future wealth levels may lead to serious time and memory problems in the calculation, known as the *curse of dimensionality*. In this case, most Q-values will suffer from poor updates as they will be reached rarely or never during the learning process. However, also in this case, another argument can be reported in favor of Q-learning. The forward approach of Q-learning going from stage t to T ensures an "efficient use" of computer power: only the *practically* relevant combinations $(W_\tau, \alpha_{c,\tau}, \alpha_{a,\tau})$, which are encountered often in the state-action space, are estimated with high accuracy.

Overall, Q-learning is most attractive when one is faced with a large and complex system which requires approximation: the simulated trajectories are used to estimate a Q-function for continuous state-action spaces, rather than to update explicitly every state-action pair.

In the second part of the paper, we have illustrated the combination of Q-learning with SOMs. Compared to function approximation methods, where after choosing the functional form the parameter vector is tuned, SOMs try to set and calibrate a finite number of neurons to deduce an optimal strategy. The results seem promising for future work.

ACKNOWLEDGMENT

The author would like to thank A. Geyer, M. Hanke, A. Matt, and G. Regensburger for helpful comments.

REFERENCES

[1] N. C. Barberis, "Investing for the long run when returns are predictable," *J. Finance*, vol. 55, pp. 225–264, 2000.
[2] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Belmont, MA: Athena Scientific, 1995, vol. 2.
[3] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*. Belmont, MA: Athena Scientific, 1996.
[4] Z. Bodie, R. C. Merton, and W. F. Samuelson, "Labor supply flexibility and portfolio choice in a life cycle model," *J. Econom. Dyn. Control*, vol. 16, pp. 427–449, 1992.
[5] J. A. Boyan and A. W. Moore, "Generalization in reinforcement learning: Safely approximating the value function," in *Advances in Neural Information Processing Systems*, G. Tesauro, D. S. Touretzky, and T. K. Leen, Eds. Cambridge, MA: MIT Press, 1995.
[6] M. W. Brandt, A. Goyal, P. Santa-Clara, and J. R. Stroud, "A simulation approach to dynamic portfolio choice with an application to learning about return predictability," *Rev. Financial Studies*, vol. 18, no. 3, pp. 831–873, 2005.
[7] M. J. Brennan, E. S. Schwartz, and R. Lagnado, "Strategic asset allocation," *J. Econom. Dyn. Control*, vol. 21, pp. 1377–1403, 1997.
[8] J. Y. Campbell, Y. L. Chan, and L. M. Viceira, "A multivariate model of strategic asset allocation," *J. Financial Econom.*, vol. 67, pp. 41–80, 2003.
[9] J. Y. Campbell and L. M. Viceira, "Who should buy long-term bonds?," *Amer. Econom. Rev.*, vol. 91, no. 1, pp. 99–127, 2001.
[10] P. X. Casqueiro and A. J. Rodrigues, "Neuro-dynamic trading methods," *Eur. J. Operat. Res.*, vol. 175, pp. 1400–1412, 2006.
[11] G. Chacko and L. M. Viceira, "Dynamic consumption and portfolio choice with stochastic volatility in incomplete markets," *Rev. Financial Studies*, vol. 18, no. 2, pp. 1369–1402, 2005.
[12] F.-J. Chang, L.-C. Chang, and Y.-S. Wang, "Enforced self-organizing map neural networks for river flood forecasting," *Hydrological Processes*, vol. 21, pp. 741–749, 2007.
[13] J. F. Cocco, F. J. Gomes, and P. J. Maenhout, "Consumption and portfolio choice over the life cycle," *Rev. Financial Studies*, vol. 18, no. 2, pp. 491–533, 2005.
[14] M. A. H. Dempster, M. Germano, E. A. Medova, and M. Villaverde, "Global asset liability management," *British Actuarial J.*, vol. 9, pp. 137–216, 2003.
[15] J. B. Detemple, R. Garcia, and M. Rindisbacher, "A Monte Carlo method for optimal portfolios," *J. Finance*, vol. 58, pp. 401–446, 2003.
[16] A. Geyer, M. Hanke, and A. Weissensteiner, "Life-cycle asset allocation and optimal consumption using stochastic linear programming," *J. Comput. Finance*, vol. 12, no. 4, pp. 1–22, 2009.
[17] A. Geyer and W. T. Ziemba, "The Innovest Austrian pension fund financial planning model InnoALM," *Operat. Res.*, vol. 56, pp. 797–810, 2007.
[18] F. Gomes and A. Michaelides, "Optimal life-cycle asset allocation: Understanding the empirical evidence," *J. Finance*, vol. 60, pp. 869–904, 2005.
[19] J. Gondzio and R. Kouwenberg, "High performance computing for asset-liability management," *Operat. Res.*, vol. 49, pp. 879–891, 2001.
[20] H. Heitsch and W. Römisch, "Scenario reduction algorithms in stochastic programming," *Comput. Optim. Appl.*, vol. 24, pp. 187–206, 2003.
[21] H. K. øyland and S. W. Wallace, "Generating scenario trees for multistage decision problems," *Manage. Sci.*, vol. 47, no. 2, pp. 295–307, 2001.
[22] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *J. Artif. Intell. Res.*, vol. 4, pp. 237–285, 1996.
[23] T. S. Kim and E. Omberg, "Dynamic nonmyopic portfolio behavior," *Rev. Financial Studies*, vol. 9, pp. 141–161, 1996.
[24] T. Kohonen, *Self-Organizing Maps*. New York: Springer-Verlag, 1995.
[25] F. Longstaff and E. Schwartz, "Valuing American options by simulation: A simple least-square approach," *Rev. Financial Studies*, vol. 14, no. 1, pp. 113–147, 2001.
[26] A. Matt and G. Regensburger, "An adaptive clustering method for model-free reinforcement learning," in *Proc. IEEE 8th Int. Multitopic Conf.*, 2004, pp. 362–367.
[27] R. C. Merton, "Lifetime portfolio selection under uncertainty: The continuous-time case," *Rev. Economics and Statistics*, vol. 51, pp. 247–257, 1969.
[28] R. C. Merton, "Optimum consumption and portfolio rules in a continuous-time model," *J. Econom. Theory*, vol. 3, pp. 373–413, 1971.
[29] J. Moody and M. Saffell, "Learning to trade via direct reinforcement," *IEEE Trans. Neural Netw.*, vol. 12, no. 4, pp. 875–889, Jul. 2001.
[30] R. Neuneiner, "Optimal asset allocation using adaptive dynamic programming," in *Advances in Neural Information Processing Systems*, D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, Eds. Cambridge, MA: MIT Press, 1996, pp. 952–958.
[31] G. C. Pflug, "Optimal scenario tree generation for multiperiod financial planning," *Math. Programm.*, vol. 89, no. 2, pp. 25–271, 2001.
[32] S. Roy, "Theory of dynamic portfolio choice for survival under uncertainty," *Math. Social Sci.*, vol. 30, pp. 171–194, 1995.
[33] P. A. Samuelson, "Lifetime portfolio selection by dynamic stochastic programming," *Rev. Econom. Statist.*, vol. 51, pp. 239–246, 1969.
[34] J. M. Santos and C. Touzet, "Exploration tuned reinforcement function," *Neurocomputing*, vol. 28, pp. 93–105, 1999.
[35] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 2000.
[36] C. Touzet, "Neural reinforcement learning for behaviour synthesis," *Robot. Autonom. Syst.*, vol. 22, Special Issue on Learning Robot: The New Wave, no. 3, pp. 251–281, 1997.
[37] J. N. Tsitsiklis, "Asynchronous stochastic approximation and Q-learning," *Mach. Learn.*, vol. 16, pp. 185–202, 1994.

[38] J. N. Tsitsiklis and B. V. Roy, "An analysis of temporal-difference learning with function approximation," *IEEE Trans. Autom. Control*, vol. 42, no. 5, pp. 674–690, May 1997.
[39] N. J. van Eck and M. van Wezel, "Application of reinforcement learning to the game of Othello," *Comput. Operat. Res.*, vol. 35, pp. 1999–2017, 2008.
[40] J. Wachter, "Portfolio and consumption decisions under mean-reverting returns: An exact solution," *J. Financial Quantitative Anal.*, vol. 37, no. 1, pp. 63–91, 2002.
[41] C. J. Watkins, "Learning from delayed rewards," Ph.D. dissertation, Psychol. Dept., Cambridge Univ., Cambridge, U.K., 1989.
[42] C. J. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, pp. 279–292, 1992.
[43] W. T. Ziemba and J. Mulvey, *Worldwide Asset and Liability Modeling*. Cambridge, U.K.: Cambridge Univ. Press, 1998.

Alex Weissensteiner was born in Bolzano, Italy, on August 20, 1974. He received the M.Sc. and Ph.D. degrees in social and economic sciences from Leopold Franzens University, Innsbruck, Austria, in 1998 and 2003, respectively.

Since 2006 he has been a Research Assistant at the Department of Banking and Finance, Leopold Franzens University. His research interests include stochastic (linear) programming, control theory, neural networks, and financial risk management.