Reinforcement learning for supply chain optimization

Lukas Kemmer
Karlsruhe Institute of Technology
1 PUBLICATION **1** CITATION
SEE PROFILE

Nikolaos Tziortziotis
Tradelab
31 PUBLICATIONS **104** CITATIONS
SEE PROFILE

Jesse Read
École Polytechnique
75 PUBLICATIONS **2,890** CITATIONS
SEE PROFILE

Reinforcement learning for supply chain optimization

Lukas Kemmer LUKAS.KEMMER@STUDENT.KIT.EDU
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany

Henrik von Kleist HENRIK.KLEIST@TUM.DE
Technical University of Munich (TUM)
Munich, Germany

Diego de Rocheboüët DDEGRIMAUDETDEROCHEB@ITBA.EDU.AR
Buenos Aires Institute of Technology (ITBA)
Buenos Aires, Argentina

Nikolaos Tziortziotis NTZIORZI@GMAIL.COM
DaSciM team, LIX, École Polytechnique
Palaiseau, France

Jesse Read JESSE.READ@POLYTECHNIQUE.EDU
DaSciM team, LIX, École Polytechnique
Palaiseau, France

Abstract

In this paper we investigate the performance of two reinforcement learning (RL) agents within a supply chain optimization environment. We model the environment as a Markov decision process (MDP) where during each step it needs to be decided how many products should be produced in a factory and how many products should be shipped to different warehouses. We then design three different agents based on a static (ς, Q)-policy, the approximate SARSA and the REINFORCE algorithm. Here we pay special attention to different feature mapping functions that are used to model the value of state and state-action pairs respectively. By testing the agents in different environment initializations, we find that both the approximate SARSA and the REINFORCE algorithms can outperform the static (ς, Q) agent in simple scenarios and that the REINFORCE agent performs best even in more complex settings.

Keywords: Reinforcement-Learning, Approximate SARSA, REINFORCE, Supply chain management

1. Introduction

Supply chain optimization is a problem faced by companies whose supply chain consists of a factory and multiple warehouses (so called hub-and-spoke networks (Arnold, 2009)). The main decision is how many products should be produced in the factory and how much stock should be built up in the warehouses. Seasonal demand can further complicate the decision problem since it might require the companies to start building up stock early to satisfy future demands (e.g., eggnog for Christmas, needing to be built up during November and December). While small companies can still maintain a supply chain management manually, automatization is necessary for big businesses. Standard policies such as the (ς, Q)-policy (Tempelmeier, 2011) are often too simple and cannot adapt to complex environments. Due to the multi-step decision characteristic of the problem, we propose an RL (Sutton and Barto, 1998) approach for the supply chain network. Previous research has already shown promising results for the application of RL to supply chain management (Pontrandolfo et al., 2002; Chaharsooghi et al., 2008; Giannoccaro and Pontrandolfo, 2002).

Even for small supply chain networks we find that due to the curses of dimensionality, (Powell, 2007), the state and action spaces of the respective decision problems become unfeasibly large. Therefore, we turn to function-approximation and policy-search methods of reinforcement learning that are less affected by these problems. In our case we chose approximate SARSA (Rummery and Niranjan, 1994; Sutton, 1996) and the REINFORCE (Williams, 1992) algorithm as a basis for the agents.

2. Problem setting

Within this paper we model a supply chain optimization problem that consists of 1 factory (i.e., a butter-factory) and multiple warehouses regarded over a fixed number of periods. In each period, the agent has to decide how much butter should be produced and stored at the factory and how much butter should be shipped to the individual warehouses. A representation of a network with 5 warehouses is depicted in Figure 1. For each of the

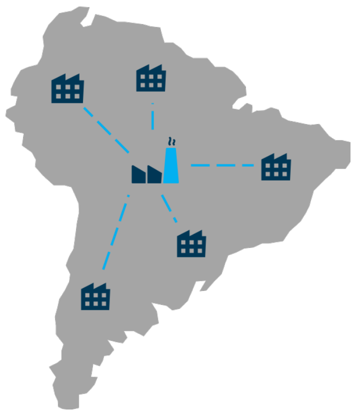


Figure 1: Example of a supply chain network with 5 warehouses and 1 factory.

warehouses, we model a stochastic, seasonal, demand for butter. If the demand can not be satisfied at a specific warehouse, it will lead to a penalty cost that will occur until the location is able to satisfy said demand. To make the problem more realistic we introduce limits to the production capacity and storage at the factory and warehouses as well as storage and shipment costs. Furthermore, we model the demand such that it can surpass the production capacity, thus requiring the agent to build up stock at the individual warehouses. This ultimately requires the agent to learn the seasonality of the demand curves and to build up stock accordingly but as efficiently as possible.

MDPs are multi-step stochastic decision problems that rely on the Markov property which implies that the transition probability $\mathbb{P}(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a})$ between two states $\mathbf{s}_{t+1}$ and $\mathbf{s}_t$ only relies on the current state $\mathbf{s}_t$ and the selected action $\mathbf{a}$. We describe the MDP similar to Moritz (2014) and Powell (2007) by defining a state-space, a random environment process for the demand, an action space, a set of feasible actions, a transition function , a one-step

Table 1: Components of the one-step reward function.

Component	Formular	Variables
Revenue from sold products	$p\sum_{j=1}^{K} d_j$	Price p, demand d_j
Production cost	$\kappa_{pr}a_0$	Unit cost κ_{pr}, production level a_0
Storage cost	$\sum_{j=0}^{K}\kappa_{st,j}\max\{s_j,0\}$	Storage cost $\kappa_{st,j}$, stock level s_j
Penalty cost	$\kappa_{pe}\sum_{j=1}^{k}\min\{s_j,0\}$	Penalty cost κ_{pe}, stock level s_j
Transportation cost	$\sum_{j=1}^{k}\kappa_{tr,j}\lceil a_j/\zeta_j\rceil$	Truck cost $\kappa_{tr,j}$,Truck capacity ζ_j Transportation volume a_j

reward function and a discount factor γ. Throughout the whole chapter we use $j = 0, \dots, K$ as an identifier for the factory ($j = 0$) and the K warehouses ($j = 1, \dots, K$) and $t = 1, \dots, N$ as an identifier for each period (where N is the terminal period).

The state space at period t is defined as $\mathbf{s}_t = [s_0, \dots, s_K, \mathbf{d}_{t-1}, \mathbf{d}_{t-2}]$ where each $s_j \in [0, c_j]$ represents the stock levels of the factory s_0 and warehouses $(s_1, \dots, s_K)$, up to some maximum capacity c_j. For the demand vector $\mathbf{d}_t = [d_{1,t}, \dots, d_{K,t}]$ we describe the individual demand at warehouse j and time t by $d_{j,t}$ which can be modeled as an arbitrary stochastic process. The reason we add the last demands $(\mathbf{d}_{t-1}, \mathbf{d}_{t-2})$ to the state space is to allow the agent to have limited knowledge of the demand history to be able to gather a basic understanding of its changes. We note that the actual stochastic demand in a period t will not be observed until the next period $t + 1$.

In each period the agent can now set the factory's production level for the next period $a_0 \in \{0, .., \rho_{max}\}$ (with a maximum production of $\rho_{max} \in \mathbb{N}$) as well as the number of products shipped to each location $a_j \in \mathbb{N}$ that is naturally limited by the current storage level in the factory $(\sum_{j=1}^{K} a_j \leq s_0)$. We can now define the action space as $\mathbf{a} = [a_0, \dots, a_K]$ and the set of feasible actions in a state $\mathbf{s}$ by $\mathcal{X}(\mathbf{s}_t) := \{a \in \mathbb{N}_0^{K+1} \mid 0 \leq a_0 \leq \rho_{max} \wedge \sum_{j=1}^{K} a_j \leq s_0\}$. Based on this information we can describe the state transitions by

$$T(\mathbf{s}_t, \mathbf{d}_t, \mathbf{a}) := (\min\{s_0 + a_0 - \sum_{j=1}^{K} a_j, c_0\}, \min\{s_1 + a_1 - d_1, c_1\}, \dots, \min\{s_K + a_K - d_K, c_K\}, \mathbf{d}_t, \mathbf{d}_{t-1}). \tag{1}$$

The one-step reward function models the profit that occurs in each period. It is defined based on the revenue and cost components presented in Table 1.

$$r(\mathbf{s}_t, \mathbf{d}, \mathbf{a}) := p\sum_{j=1}^{K} d_j - \kappa_{pr}a_0 - \sum_{j=0}^{K}\kappa_{st,j}\max\{s_j, 0\} + \kappa_{pe}\sum_{j=1}^{K}\min\{s_j, 0\} - \sum_{j=1}^{K}\kappa_{tr,j}\left\lceil\frac{a_j}{\zeta_j}\right\rceil \tag{2}$$

with $\lceil\cdot\rceil$ to be the ceiling function. The terminal reward after the last period is zero, meaning that we will not account for remaining positive or negative stock. Furthermore we chose a discounting factor $\gamma \in \mathbb{R}_+$ that can be interpreted, e.g., as a result of inflation.

3. Our approach

To solve the supply chain optimization problem, we compare the performance of the two reinforcement learning algorithms (approximate SARSA and REINFORCE) with an agent that acts according to a fixed heuristic based on the (ς, Q)-Policy[1] as described by Tempelmeier (2011). In order to test the agents we model the previously undefined elements of the demand vector $\mathbf{d}_t = [d_{1,t}, \ldots, d_{K,t}]$ as a sinusoidal function with stochastic shocks to simulate a simple seasonal demand behavior. This leads us to

$$d_{j,t} = \left\lfloor \frac{d_{max}}{2} \sin\left(\frac{2\pi(t+2j)}{12}\right) + \frac{d_{max}}{2} + \epsilon_{j,t} \right\rfloor \tag{3}$$

where $\lfloor\cdot\rfloor$ is the floor function and $\mathbb{P}(\epsilon_{j,t} = 0) = \mathbb{P}(\epsilon_{j,t} = 1) = 0.5$. In RL (Sutton and Barto, 1998) we design an agent that acts in an environment to independently learn a strategy based on the rewards he collects after each action. Often this is done by approximating a Q-function that maps state-action pairs to a value which is used by the agent to select the action with the highest associated value for each state.

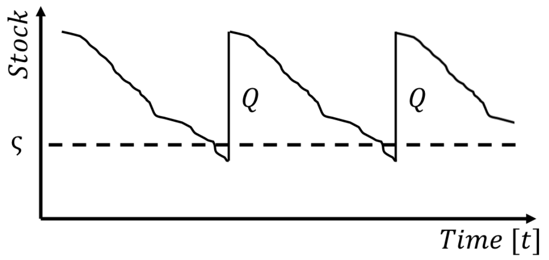


Figure 2: Schematic visualization of (ς, Q)-Policy for a single warehouse. A fixed amount Q is replenished when stock falls under threshold ς.

The (ς, Q)-Policy based agent is not smart in the way that it does not learn over time. Due to the popularity of the (ς, Q)-Policy in practice (Janssen et al., 1996), we use it as a baseline for performance evaluation of the other agents. In our heuristic we iterate over $s_1, ..., s_K$ and replenish the respective warehouses by some amount $a_j = Q_j$ if the current stock is below a level ς_j and there is still stock left in the factory s_0. At the end we set the production level for the next period to Q_0 if $s_0 - \sum_{j=1}^{K} a_j < \varsigma_0$ and zero otherwise. The thresholds ς and replenishment levels Q need to be set by the user when initializing the agent.

Approximate SARSA Rummery and Niranjan (1994) use a linear approximation $Q_{\mathbf{w}}(\mathbf{s}, \mathbf{a}) = \mathbf{w}^T \phi(\mathbf{s}, \mathbf{a})$ of the Q-function (which describes the value of being in a state and choosing a specific action) where $\mathbf{w}$ is a vector of parameters and $\phi(\mathbf{s}, \mathbf{a})$ is a function of features that we will call a feature-map. We chose this method as it solves the problem of exponentially growing state and action spaces and allows us to use our knowledge of the environment to design $\phi(\mathbf{s}, \mathbf{a})$ such that it preserves some of the MDPs structure. One of the crucial tasks when designing the approximate SARSA agent is the model for ϕ. In our case we use the states $\mathbf{s}$ and actions $\mathbf{a}$ to compute over 15 different features to find a good approximation of the Q-function. One of the main ideas is to explore our knowledge of the

1. In the literature this policy is usually called (s, Q)-Policy. We chose a slightly different name to avoid notation conflicts in our model description.

transition function to get a rough estimate of the next demand and state by

$$\begin{aligned} \hat{\mathbf{d}}(\mathbf{s}_t) &= \mathbf{d}_{t-1} + (\mathbf{d}_{t-1} - \mathbf{d}_{t-2}) = 2\mathbf{d}_{t-1} - \mathbf{d}_{t-2}, \\ \hat{\mathbf{s}}_{t+1}(\mathbf{s}, \mathbf{a}) &= T(\mathbf{s}_t, \hat{\mathbf{d}}(\mathbf{s}_t), \mathbf{a}). \end{aligned} \tag{4}$$

Note that this does not imply that the agent fully understands all transition dynamics since the actual realizations of the demand follow an unknown stochastic process. This way the agent can get a basic understanding of rewards and penalties associated with $\hat{\mathbf{s}}_{t+1}$. Among others we use the expected penalty costs, the expected reward and include the respective rewards and costs for two scenarios where $\hat{\mathbf{d}}^+ = \hat{\mathbf{d}}(\mathbf{s}_t) + \mathbb{1}$ and $\hat{\mathbf{d}}^+ = \hat{\mathbf{d}}(\mathbf{s}_t) - \mathbb{1}$. A list of features can be found in appendix A. When testing the approximate SARSA algorithm we found that for some environments the parameters $\mathbf{w}$ would increase until computations became numerically unstable. To avoid this issue, a simple solution is to restrict the temporal difference that is used to update $\mathbf{w}$ within the interval $[-10^{100}, +10^{100}]$ and to initialize the agent with a very small learning-rate (such as 10^{-10} vs. 0.02 which we normally use). We deliberately chose a big interval in order to only affect the results in cases where $\mathbf{w}$ tends to extreme values. In future work this basic approach could be improved e.g., by using the softmax of the weights.

REINFORCE (Williams, 1992) is based on a parametrized policy for which the expected reward has to be maximized. Due to the high dimensionality of the problem, we discretize the action space and only allow 3 different actions for each location (i.e., the factory and the warehouses). This can be for example to send 0,1 or 2 trucks to a warehouse or to produce 0, 3 or 9 units at the factory. The maximum number of possible actions combining all locations becomes $n_a = 3^{(K+1)}$ and we denote $\mathbf{a}^{(i)}$ as the i^{th} action. Note that the constraints from $\mathcal{X}(\mathbf{s}_t)$ still apply, i.e., that the number of allowed actions in a specific state can be smaller. We parametrize our policy as a softmax function for multiple actions:

$$p(\mathbf{a} = \mathbf{a}^{(i)}|\mathbf{s}) = \pi_\Theta(\mathbf{a} = \mathbf{a}^{(i)}|\mathbf{s}) = \frac{e^{\phi(\mathbf{s})^T w_{\mathbf{a}^{(i)}}} \cdot f(\mathbf{a}^{(i)}|\mathbf{s})}{\sum_{j=1}^{n_a} e^{\phi(\mathbf{s})^T w_{\mathbf{a}^{(j)}}} \cdot f(\mathbf{a}^{(j)}|\mathbf{s})} := \sigma_i(\mathbf{s}) \tag{5}$$

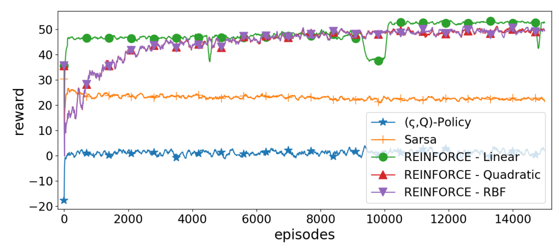
where

$$f(\mathbf{a}^{(j)}|\mathbf{s}) = \begin{cases} 1 & \text{if } \mathbf{a}^{(j)} \text{ is allowed in state } \mathbf{s}, \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

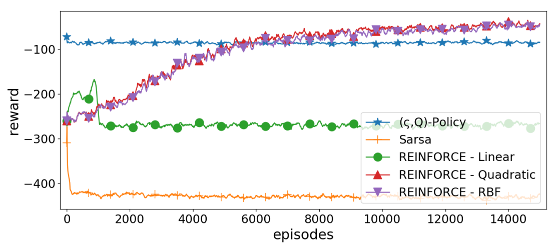
For REINFORCE, we used a simplified feature map $\phi(\mathbf{s})$ with only the state $\mathbf{s}$ as input and a basis consisting of a bias with linear terms. Furthermore, we added quadratic or Radial Basis Function (RBF) kernel terms. The RBF kernels were designed individually for each location with means chosen to be zero, half and full capacity and a small constant standard deviation. The parameters $w_i \in \mathbb{R}^{3K+1}$, assembled in the matrix $\Theta = \big(w_1|w_2|...|w_{n_a}\big) \in \mathbb{R}^{(3K+1)\times n_a}$, are initialized to zeros in order to start with equal probabilities for each action. The gradient of our objective function $J(\Theta) = \mathbb{E}_{\pi_\Theta}[\sum_{t=1}^{N} r_t]$ evaluates to:

$$\nabla_{w_j} J(\Theta) = \nabla_{w_j} \ln(\pi_\Theta(\mathbf{a}^{(i)}, \mathbf{s}))D_t = \begin{cases} (1 - \sigma_j(\mathbf{s}))\phi(\mathbf{s})D_t & \text{if } i = j \\ -\sigma_j(\mathbf{s})\phi(\mathbf{s})D_t & \text{if } i \neq j \end{cases}$$

where $D_t = \sum_{t'=t}^{N} r'_t$. The full derivation for this gradient is shown in Appendix B.

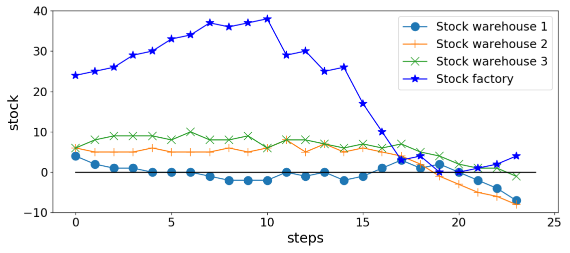


(a) Simple scenario (one warehouse).

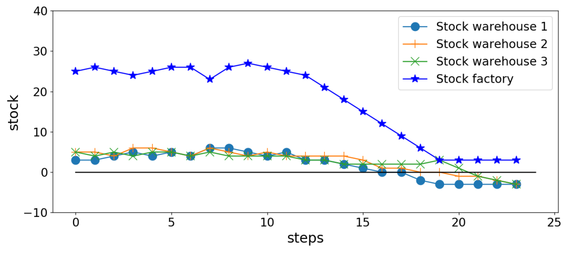


(b) Complex scenario (three warehouses).

Figure 3: Average reward on a sliding window of size 100.



(a) Stocks for the REINFORCE agent using a quadratic ϕ.



(b) Stocks for the (ς, Q)-Policy based agent.

Figure 4: Stocks in the second test.

4. Results and Discussion

We carry out two tests, each of 24 steps to simulate a full demand cycle, where demand increases along the length of the episode, thereby asking the question – can the agent learn to build up stock? During the tests we simulate 15000 episodes for the agents to learn and to track their performance which we will later display on a sliding window.

The first test consists of only the factory and one warehouse and includes cost for production, storage (except for the factory), transportation and penalty cost for unsatisfied demands. The agent should learn to invest in storage and transportation, despite short-term negative rewards. Figure 3a shows that both the approximate SARSA and REINFORCE agents (three versions thereof) are successful and outperform the baseline (ς, Q)-policy. The REINFORCE methods are clearly superior to approximate SARSA.

We then test a more complex environment that consists of three warehouses where the second and third warehouse have no storage costs and the third warehouse also has no transportation cost. The results are depicted in Figure 3b. In this case only the quadratic and RBF REINFORCE agents improve over the baseline (ς, Q)-Policy.

Figures 4a and 4b depict storage levels within the best episodes of both agents in the second test. In this analysis it is clear that it is the higher flexibility of the REINFORCE agent that builds up more stock than the (ς, Q)-Agent in the beginning and thus satisfies high demands at the end of the episode. It also highlights that the (ς, Q)-agent cannot adapt to changes in the environment.

Furthermore we see that both agents manage to keep positive stock levels most of the time but only REINFORCE is able to allocate stock more efficiently at warehouses two and three that do not have storage costs while keeping the stock of warehouse one close to zero.

Results indicate that the REINFORCE approach is certainly a viable option to tackle the supply chain optimization environment. Drawbacks come from the softmax parametrization which requires to fit parameters for each action. This makes it necessary to conduct a lot of training and can lead to difficulties for actions that are rarely feasible. Moreover, the actions to replenish 0, 1 or 2 trucks are treated as completely separate, dropping the possibility to make use of their natural order. Another parametrization that would make use of this order would be e.g., a Gaussian setting. A big advantage of the used feature map for the REINFORCE algorithm is that it does not exploit any knowledge about the environment and can thus also be used if e.g., the demand process is unknown.

A crucial aspect for both approaches is feature engineering. In fact a different set of features might improve the performance of the approximate SARSA agent. Often a linear dependency of the optimal action on the current state is reasonable enough (e.g., the best action for the replenishment of products for a warehouse might depend linearly on the current stock in that warehouse) but also more complex, non-linear dependencies might be present. Thus, a promising alternative might be to use a deep neural network in order for the agent to be able to adapt to any non-linear dependency.

5. Conclusion and Future Work

The supply chain environment poses a demanding task faced by many companies in real life contexts. We have shown a way to solve instances of this problem by policy gradient methods that yield encouraging results, indicating that we can design agents that are able to understand simple market trends, regulate production levels and allocate stock efficiently in a simple model scenario. In future work we will model the stochastic policy of REINFORCE by a deep neural network, and deploy it in more complex versions of the environment using different demand curves. This way we will explore potential improvements to the agents and evaluate how robust they react to different experiment designs and demand curves. Lastly, the REINFORCE algorithm will be tested with real world data in order to examine if the algorithm can improve supply chain networks in practice.

References

Dieter Arnold. Materialfluss in logistiksystemen, 2009. URL `http://swbplus.bsz-bw.de/bsz312838174cov.htmhttp://dx.doi.org/10.1007/978-3-642-01405-5`.

S Kamal Chaharsooghi, Jafar Heydari, and S Hessameddin Zegordi. A reinforcement learning model for supply chain ordering management: An application to the beer game. *Decision Support Systems*, 45(4):949–959, 2008.

Ilaria Giannoccaro and Pierpaolo Pontrandolfo. Inventory management in supply chains: a reinforcement learning approach. *International Journal of Production Economics*, 78(2): 153–161, 2002.

Fred Janssen, R Heuts, and Ton de Kok. The value of information in an (r,s,q) inventory model. 02 1996.

Lars Norman Moritz. *Target Value Criterion in Markov Decision Processes.* PhD thesis, 2014. URL `http://digbib.ubka.uni-karlsruhe.de/volltexte/1000047288`. Karlsruhe, KIT, Diss., 2014.

Pierpaolo Pontrandolfo, Abhijit Gosavi, O Geoffrey Okogbaa, and Tapas K Das. Global supply chain management: a reinforcement learning approach. *International Journal of Production Research*, 40(6):1299–1317, 2002.

Warren B. Powell. *Approximate dynamic programming : solving the curses of dimensionality.* Wiley series in probability and statistics. Wiley-Interscience, Hoboken, NJ, 2007. ISBN 978-0-470-17155-4. URL `http://swbplus.bsz-bw.de/bsz275179400cov.htm;http://www.gbv.de/dms/ilmenau/toc/527185191.PDF`. Includes bibliographical references.

Gavin A Rummery and Mahesan Niranjan. On-line q-learning using connectionist systems. Technical report, Cambridge University, 1994.

Richard S. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems 8*, pages 1038–1044, 1996.

Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning.* MIT Press, 1998.

Horst Tempelmeier. *Inventory management in supply networks : problems, models, solutions.* Books on Demand, Norderstedt, 2. ed. edition, 2011. ISBN 978-3-8423-4677-2. URL `http://deposit.d-nb.de/cgi-bin/dokserv?id=3676720&prov=M&dok_var=1&dok_ext=htm`.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, 1992.

Appendix A.

Table 2: Features of the Q-function approximation where $\hat{d}$ and $\hat{s}$ are simple estimates of the next demand and state.

Feature	**Description**	**Computation**
Bias	-	1
Sales reward	Expected reward from sales	$p\sum_{j=1}^{K}\hat{d}_j$
Production cost	Production cost in the factory	$\kappa_{pr}a_0$
Storage cost	Per location $j = 0, ..., K$	$-\kappa_{st,j}\max\{\hat{s}_j, 0\}$
Penalty cost	Per warehouse $j = 1, ..., K$	$\kappa_{pe}\min\{\hat{s}_j, 0\}$
Transportation cost	Per warehouse $j = 1, ..., K$	$\kappa_{tr,j}\left\lceil\frac{a_j}{\zeta_j}\right\rceil$
Factory stock	Sufficient factory stock to satisfy demand	$\hat{s} \geq \sum_{j=1}^{K}\hat{d}_j$
Positive stock	Per warehouse $j = 1, ..., K$	$\hat{s} \geq 0$
Estimated demand	Per warehouse $j = 1, ..., K$	$\hat{d}$
Sq. estimated demand	Per warehouse $j = 1, ..., K$	$\hat{d}^2$
Storage level deviation	Squared difference from different storage levels q_j	$(\hat{s}_j - q_j)^2$
Demand satisfaction	Is production able to satisfy expected demand	$a_0 \geq \sum_{j=1}^{K}\hat{d}_j$

Appendix B.

$$\nabla_{w_j}J(\Theta) = \nabla_{w_j}\ln(\pi_\Theta(\mathbf{a}(t) = \mathbf{a}^{(i)}, \mathbf{s}(t)))D_t$$

where for $i = j$

$$\begin{aligned}
&\nabla_{w_j}\ln(\pi_\Theta(\mathbf{a}(t) = \mathbf{a}^{(i)}, \mathbf{s}))\\
&\quad= \nabla_{w_j}(\phi(\mathbf{s})^T w_{\mathbf{a}^{(i)}}) + \nabla_{w_j}\ln(f(\mathbf{a}^{(i)}|\mathbf{s}))\\
&\quad- \nabla_{w_j}\ln(\sum_{l=1}^{n_a} e^{\phi(\mathbf{s})^T w_{\mathbf{a}^{(l)}}}\cdot f(\mathbf{a}^{(l)}|\mathbf{s}))\\
&\quad= \phi(\mathbf{s}) - \frac{e^{\phi(\mathbf{s})^T w_{\mathbf{a}^{(j)}}}\cdot f(\mathbf{a}^{(j)}|\mathbf{s})}{\sum_{l=1}^{n_a} e^{\phi(\mathbf{s})^T w_{\mathbf{a}^{(l)}}}\cdot f(\mathbf{a}^{(l)}|\mathbf{s})}\cdot\phi(\mathbf{s})\\
&\quad= (1 - \sigma_j(\mathbf{s}))\cdot\phi(\mathbf{s})
\end{aligned}$$

and for $i \neq j$:

$$\begin{aligned}
&\nabla_{w_j}\ln(\pi_\Theta(\mathbf{a}(t) = \mathbf{a}^{(i)}, \mathbf{s}))\\
&\quad= \nabla_{w_j}(\phi(\mathbf{s})^T w_{\mathbf{a}^{(i)}}) + \nabla_{w_j}\ln(f(\mathbf{a}^{(i)}|\mathbf{s}))\\
&\quad- \nabla_{w_j}\ln(\sum_{l=1}^{n_a} e^{\phi(\mathbf{s})^T w_{\mathbf{a}^{(l)}}}\cdot f(\mathbf{a}^{(l)}|\mathbf{s}))\\
&\quad= -\frac{e^{\phi(\mathbf{s})^T w_{\mathbf{a}^{(j)}}}\cdot f(\mathbf{a}^{(j)}|\mathbf{s})}{\sum_{l=1}^{n_a} e^{\phi(\mathbf{s})^T w_{\mathbf{a}^{(l)}}}\cdot f(\mathbf{a}^{(l)}|\mathbf{s})}\cdot\phi(\mathbf{s})\\
&\quad= -\sigma_j(\mathbf{s})\cdot\phi(\mathbf{s})
\end{aligned}$$