sciendo

Transport and Telecommunication, 2019, volume 20, no. 3, 251–259
Transport and Telecommunication Institute, Lomonosova 1, Riga, LV-1019, Latvia
DOI 10.2478/ttj-2019-0021

METAMODELLING OF INVENTORY-CONTROL SIMULATIONS BASED ON A MULTILAYER PERCEPTRON

Ilya Jackson [1], Jurijs Tolujevs [2], Sebastian Lang [3], Zhandos Kegenbekov [4]

[1,2] Transport and Telecommunication Institute (TTI)
Lomonosova iela 1, Riga, Latvia
[1] jackson.i@tsi.lv
[2] tolujevs.j@tsi.lv

[3] Fraunhofer Institute for Factory Operation and Automation (IFF)
Sandtorstraße 22, Magdeburg, Germany
sebastian.lang@iff.fraunhofer.de

[4] Kazakh-German University
Pushkin 111, Almaty, Kazakhstan
kegenbekov@dku.kz

Inventory control problems arise in various industries, and each single real-world inventory is replete with non-standard factors and subtleties. Practical stochastic inventory control problems are often analytically intractable, because of their complexity. In this regard, simulation-optimization is becoming more and more popular tool for solving complicated business-driven problems. Unfortunately, simulation, especially detailed, is both time and memory consuming. In the light of this fact, it may be more reasonable to use an alternative cheaper-to-compute metamodel, which is specifically designed in order to approximate an original simulation. In this research we discus metamodelling of stochastic multiproduct inventory control system with perishable products using a multilayer perceptron with a rectified linear unit as an activation function.

Keywords: metamodelling, simulation, inventory control, artificial neural network, industrial artificial intelligence

1. Introduction

Mankind deals with inventory management, since it started to mine and stockpile resources of the planet. For the last 80 years, the field of inventory management has attracted massive attention in both academic and business worlds (Jalali and Nieuwenhuyse, 2015). This is not surprising, since inventory-related issues constitute significant expenses for various businesses. According to IHL Group's report, a tremendous share of capital, namely $1.1 trillion in cash or equivalent to 7% of the U.S. GDP are tied up in inventory (IHL and Buzek, 2015). Moreover, companies are losing $634.1 Billion each year due to backorders, which makes up 4.1% of revenue for an average retailer (Dynamic Action and IHL, 2015). Traditionally inventory control problems are solved with dynamic programming methods (Domschke *et al*., 2015). The basic idea of dynamic programming is based on separation of a problem into various sub-problems. For each sub-problem objective function is recursively assigned in such a way that the combination of the partial problems' optima corresponds to the optimum of the main problem (Bellman, 1957). However, the computational effort for solving problems with more than one state variable and many periods to be considered can be still very huge. Furthermore, determining a global optimum with dynamic programming can be even practically impossible, if one deals with a stochastic inventory control problem, where periodical demands, replenishment lags or other parameters are random variables. Besides that, inventory control problems arise in various industries, and each single real-world inventory is replete with non-standard factors and subtleties. In this regard, it is extremely unlikely that the same set of assumptions and considerations will be equally applicable to all real systems.

Moreover, as it recognized by such researchers as Duan and Liao (2013) and Tsai and Zheng (2013), practical stochastic inventory problems are often analytically intractable, because of their complexity. Due to such restrictions of both analytical models and approaches based on dynamic programming, simulation-optimization is becoming more and more popular tool for solving complicated business-driven problems (Jalali and Nieuwenhuyse, 2015). Unfortunately, simulation, especially detailed, is both time and memory consuming. Besides that, simulation-optimization algorithms do not "learn", more specifically, whenever

the input parameters change slightly, a metaheuristic search (or another optimization method) must be executed once again. Taking into account a scale and dimensionality of real-world inventory control problems, such a never-ending search becomes an unacceptable luxury for business. In the light of these facts, it may be more reasonable to use an alternative cheaper-to-compute model, which is specifically designed in order to approximate an original simulation with a sufficient degree of accuracy (Merkuryeva, 2004). Such a "model of the model" is conventionally called a metamodel (Blanning, 1975). Pursuing this goal, we suggest taking advantage on the property of feed-forward neural networks to generalize. More specifically, the pivotal idea is based on the fact that during a metaheuristic search candidate solutions may be stored and used later to train an artificial neural network (ANN). Considering the fact that ANNs with more than one hidden layer are distinguished for such properties as the ability to learn nonlinear relations and robustness to noise, they become a promising metamodelling candidate.

2. Related Work

The simulation model under consideration is mainly based on the recent work (Jackson and Tolujevs, 2019) and (Jackson *et al*., 2018) adopting several new features from related papers. Namely, the model incorporates perishability mechanics quite similar to one used by Duan and Liao (2013). Besides that, the model operates with multiple products under the constrained total inventory in a similar way as the transhipment inventory simulation described by Hochmuth and Kochel (2012).

Metamodelling is a fairly old and well-known approach in simulation community (Law and Kelton, 2000). Besides, metamodelling did not bypass logistics and production (Tolujev *et al*., 1998). However, with recent revolution in deep-learning, metamodelling with ANN has once again sparked the surge of interest and become a "hot-topic" in simulation community (Lechevalier *et al*., 2015). Among applications related to inventory-control, it is worth to mention the research conducted by Prestwich *et al*. (2012), who managed to efficiently combine a single-layered ANN with an evolutionary algorithm to find an optimal policy for a simulation-based stochastic multi-echelon inventory-control system. Additionally, it is important to emphasize the study (Lin *et al*., 2009) that proposes an algorithm for defining a work-in-process inventory level for wafer fabrication processes. This study adopts a simulation model based on a real wafer fabrication factory to generate data and demonstrates an optimization algorithm combining a multilayer feedforward neural network and sequential quadratic programming. Furthermore, Farhat and Owayjan (2017) have recently presented the simulation of an enterprise resource planning system that utilizes 30-layered ANN as an inventory-policy controller.

3. Methodology

In order to formalize metamodelling with ANN, let's consider a nonlinear input-output mapping described by the functional relationship *d=f(x)*, where the vector *x* is the input and *y* is the output. The simulation *f(.)* is a "black-box". However, we are given the set of observations $\tau = \{(x_i, y_i)\}_{i=1}^{N}$. The requirement is to design an ANN that approximates the "black-box", such that the network's prediction *F(.)* is close enough to *f(.)* in Euclidean sense $\|F(x) - f(x)\| < \varepsilon$, where ε is a positive number, small enough in the context of the task (Haykin, 2009).

In this research we rely on the universal approximation theorem proved by Cybenko (1989) and later extended by Hornik (1991), which states that a feed-forward network with at least one hidden layer containing a finite number of neurons can approximate any simulation model distinguished for nonlinear relation between input and output variables. The developed metamodel may be classified as a multilayer perceptron (MLP) with a rectified linear unit (ReLU) as an activation function. Since ReLU includes only such operations as comparison, addition and multiplication, it becomes extremely efficient in terms of computation (Glorot *et al*., 2011), which is, along with accuracy, the most important requirement for a metamodel. The considered MLP uses mean squared error as a loss function and the extension to stochastic gradient descent "Adam" for weights recalculations (Kingma and Ba, 2014). In order to test the ability of the metamodel to predict simulation's output for inputs that were not used in training, this research takes advantage on k-fold cross-validation. With such an approach, common machine learning-related problems like overfitting will be spot (Cawley and Talbot, 2010).

4. Model Description

The model is designed for further algorithmic implementation in the form of a discrete-event simulation (DES), and described using Hurlimann's indexed notation (Hurlimann, 2007) as a base complemented with set theory notation (Winskel, 2010) and Iverson's brackets (Iverson, 1962).

4.1. Material flow

The inventory-control system (ICS) under consideration operates with a sequence of products $P = (p_1, p_2, \ldots, p_n)_{n\in\mathbb{N}^+}$ and has a limited total storage capacity I_{max}. The model considers only those moments of time, in which the system parameters change (discrete events of particular interest happen). Timings of such events is given as a sequence $T = (t_1, t_2, \ldots, t_n)_{n\in\mathbb{N}^+}$. In this regard, the value of $t_n - t_1$ may be considered as the planning horizon. Since the ICS deals with perishable products, the storage is represented as a sequence of lots $S_t^p = (s_1^{p,t}, s_2^{p,t}, \ldots, s_n^{p,t})_{n\in\mathbb{N}^+}$ replenished at different moments of time $t\in T$. Such that for each S_t^p there is a corresponding sequence of days to expiration (DTE) $E_t^p = (e_1^{p,t}, e_2^{p,t}, \ldots, e_n^{p,t})_{n\in\mathbb{N}^+}, |S_t^p| = |E_t^p| \; \forall p \in P, \forall t \in T$. Thus, for each single product at each moment of time, there is an inventory level $I_t^p = \sum_{i=1}^{n} S_t^p$. DTE decrease during the time and, in order to model that in iterative way, the following function is introduced:

$$E_{t+1}^p = \varepsilon(E_t^p, t_i, t_{i+1}) = (e_0^{p,t} - (t_{i+1} - t_i), e_1^{p,t} - (t_{i+1} - t_i), \ldots, e_n^{p,t} - (t_{i+1} - t_i)). \quad (1)$$

Such that in each discrete time interval $\Delta t = t_i - t_{i-1}$, all empty and expired lots are removed, $\forall e_i^{p,t} \leqslant 0,\ S_t^p \leftarrow S_t^p / s_i^{p,t}, E_t^p \leftarrow E_t^p / e_i^{p,t}$ and $\forall s_i^{p,t} = 0,\ S_t^p \leftarrow S_t^p / s_i^{p,t}, E_t^p \leftarrow E_t^p / e_i^{p,t}$ (Fig. 1). It is done for an obvious reason, namely, expired commodity is unfit for use, consumption, or sale. It is also crucially important to trace the number of expired products for later total expenses calculation (2).

$$Expired_t^p = \sum_{i=1}^{n} s_i^{p,t}[\Phi], \quad (2)$$

where: $[\Phi]$ is the Iverson bracket $[\Phi] = \begin{cases} 1 \; if \; e_i^{p,t} \leqslant 0 \\ else \; 0 \end{cases}$ (Iverson, 1962).

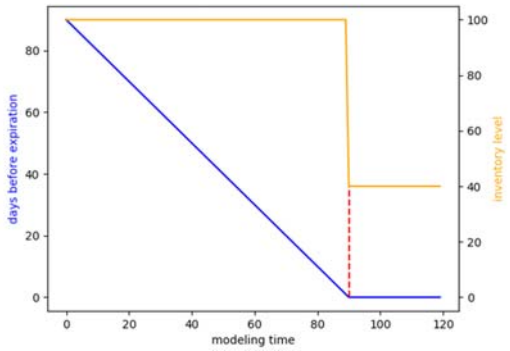

Figure 1. Perishability mechanics

Let's assume that $T_{demands}^p = (\hat{t}_1, \hat{t}_2, \ldots, \hat{t}_n)$ consists only of time instances, when the new demand d_t^p for a product p arises. Since the model under consideration is stochastic, demand size is a random variable D under a known distribution $F_d = P(d \leqslant D)$. In this regard, we may denote demand inter-arrival time as $a_i = \hat{t}_i - \hat{t}_{i-1}$, which is a value of a random variable A under a specified continuous distribution $F_a = P(a \leqslant A)$. We also define a recursive function $f^{i=1}(.)$ that fulfils arising demands depending on the available inventory capacity (3).

$$f^{i=1}(s_i^{p,t}, d_t^p) = \begin{cases} s_i^{p,t} \leftarrow s_i^{p,t} - d_t^p \; if \; s_i^{p,t} \geq d_t^p \\ else \; s_i^{p,t} \leftarrow 0, f^{i+1}(s_{i+1}^{p,t}, d_t^p - s_i^{p,t}) \end{cases}, \quad (3)$$

where: i stands for a lot's index to fulfil the demand from. We also keep track on fulfilled demand $Sales_t^p = \begin{cases} d_t^p \; if \; I_t^p \geq d_t^p \\ else \; I_t^p \end{cases}$ for later net profit calculation.

For each product $p\in P$ there is a pair of control parameters (Q^p, r^p) that determine an inventory control policy. It is important to note that such parameters are constant and do not vary over time. As soon as the current inventory level reaches the threshold r^p (reorder point) $I_t^p \leq r^p$, the ICS orders a new batch of size Q^p. Besides, a Boolean variable $status^p \in \{\text{True, False}\}$ is declared in order to know if the batch is already ordered and on the way (4).

$$o_t^p = \begin{cases} Q^p, \; status^p \leftarrow True \; if \; I_t^p \leq r^p \text{and} \; status^p is \; False \\ else \; 0 \end{cases}. \quad (4)$$

If an order is placed, namely $o_t^p > 0$, the inventory-level will not be replenished immediately. Instead, there will be a lag between the time, when the order has been placed and the time, when the order is delivered. We denote such a delivery lag (or replenishment lead time) as a random variable L under a known distribution $F_l = P(L \leqslant l)$. This literally means that the order o_{t-l}^p made at the moment of time $t_i \in T$ will be appended to the storage $S_t^p \cup \{o_{t-l}^p\}, E_t^p \cup \{e^p\}$ at the moment of time $t_j \in T$ such that $t_j - t_i = l$. For this reason, we introduce a supply function $g(.)$ (5).

$$(S_{t+1}^p, E_{t+1}^p) = g(S_t^p, E_t^p, Q^p) = \begin{cases} S_t^p,\ E_t^p\ if\ o_{t-l}^p = 0 \\ else\ S_t^p \cup \{o_{t-l}^p\}, E_t^p \cup \{e^p\},\ status^p \leftarrow False \end{cases}. \quad (5)$$

It is important to note that either a backorder-event $d_t^p > I_t^p$ or an overflow-event $\sum_{t=1}^n S_t^p > I_{\max}$ may take place. The model also keeps track on these events for later cost function calculation (6,7).

$$Backorders_t^p = \begin{cases} 0\ if d_t^p \leq I_t^p \\ else\ d_t^p - I_t^p \end{cases}, \quad (6)$$

$$Overflow_t^p = \begin{cases} 0\ if \sum_{t=1}^n S_t^p \leq I_{\max} \\ else \sum_{t=1}^n S_t^p - I_{\max} \end{cases}. \quad (7)$$

Obeying the following natural order of operations: 1) check expiration dates 2) replenish previously ordered goods 3) fulfil the demand, we may finally end up with the equation that will simulate the dynamics (8).

$$(S_{t+1}^p, E_{t+1}^p) = f(g(S_t^p, \varepsilon(E_t^p), Q^p)). \quad (8)$$

Considering that $I_t^p = \sum_{t=1}^n S_t^p$, The overall inventory dynamics may be expressed in the following way:

$$I_{t+1}^p = min(max(I_t^p + o_t^p - d_t^p - Expired_t^p, 0), I_{max}). \quad (9)$$

With this equation we also emphasize that inventory cannot be negative or bigger than I_{max}.

4.2. Monetary flow

At the monetary level the model makes a mild assumption. Namely, income is received immediately after commodity is sold, and transfer does not take time. The model takes into account 5 composite costs: ordering costs, inventory costs, backorder costs, overflow fee and recycle fee.

Ordering cost includes both purchase price and transportation cost. This model adopts the cut-off point quantity discount (Buffa and Taubert, 1972), which is given only to the extent that the order exceeds a cut-off point (10).

$$c^p(Q^p) = \begin{cases} c^p for\ 0 < Q^p \leq \beta_1 \\ k_2 c^p for\ \beta_1 < Q^p \leq \beta_2 \\ k_n c^p for\ \beta_{n-1} < Q^p \leq \beta_n \end{cases}, \quad (10)$$

where: ordering costs for a unit (e.g. pallet) $c^p(Q^p)$ is a function of an order quantity, such that $B^p = (\beta_1^p, \beta_2^p, \ldots, \beta_n^p)$ is a series of cut-off points and $K^p = (1, k_2^p, \ldots, k_n^p)$, $\forall k_i^p \in [0,1]$ is a series of corresponding discount factors.

From a business point of view, a quantity discount is a stimulus offered to a customer that results in a decreased cost per unit, when a commodity is purchased in a greater amount. That is a common practice to entice customers to purchase more. As the result, a seller can increase turnover, and a customer receives more favourable prices. In this regard, the cut-off point quantity discount introduces a quite realistic return-to-scale mechanics to the model. Depending on the real-world context, unit inventory cost h^p may consist of handling costs and opportunity loss. By opportunity loss, we mean the possibility of using the capital for other purposes (frozen capital). In this model inventory cost is constant and set for a unit of modelling time. Such that $Inventorycost^p = h^p \sum_{i=1}^n I_i^p \Delta t_i$. We also consider that every single backorder (out-of-stock) is associated with a loss of business reputation. When a product is backordered, a customer is stimulated to search for a substitute. This fact provides a possibility that once loyal customer may potentially switch, which leads to eventual loss of a market share. In this regard, for every product in the ICS backorder is associated with a constant fee b^p. Overflows are also penalized by a constant fee w^p. In real world, such expenses may be related to the sudden need for reverse logistics. Additionally, when a lot is perished, quite

similar penalty x^p related to reverse logistics of expired goods arises. Namely, such a batch must be sent for recycling. Based on the introduced variables total costs related to a product p may be calculated as follows (11).

$$TC^p = c^p \sum o_t^p + h^p \sum I_t^p \Delta t + b^p \sum Backorders_t^p + w^p \sum Overflow_t^p + x^p \sum Expired_t^p \,. \quad (11)$$

Taking into account that each unit of product p is sold by a constant $price^p$, the net profit associated with this product is $Profit^p = price^p \sum sales^p - TC^p$. Thus, total net profit of the ICS is $\sum_{p=1}^{n} Profit^p$, which is treated as the simulation output.

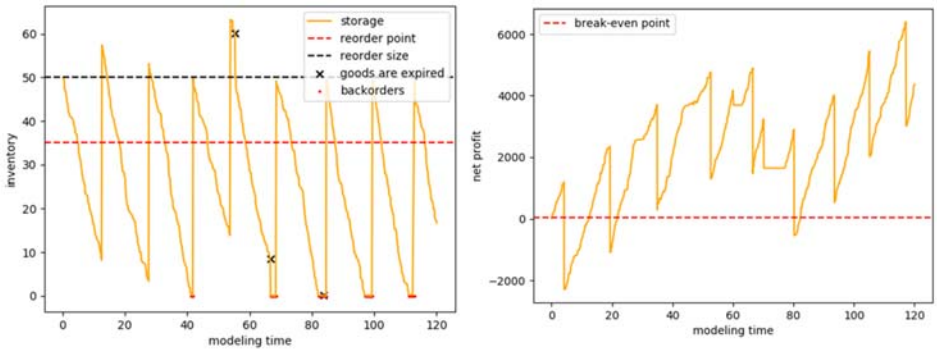

Figure 2. The dynamics of physical and monetary flows

4.3. Algorithmic implementation

Despite such a verbose formal description, the simulation model may be executed by a relatively simple iterative algorithm consisting of three functions (Table 1).

Table 1. Pivotal functions behind the simulation algorithm

Function 1: Check the expiry()	**Function 2: Replenishment()**	**Function 3: Fulfil demand()**

for *all* elements *in* E_t^p ***do***
  if $e_i^p \leq 0$ ***do***
  $E_t^p.pop(i)$
  end if
end for
if $r^p > I_t^p$ *and status* = False ***do***
  status ← True
  $o_t^p \leftarrow Q^p$
end if
	if *status* = True *and* $o_{t-1}^p \neq 0$ ***do***
	  $S_t^p.append(Q^p)$
	  $E_t^p.append(e^p)$
	  status ← False
	end if
		if $I_t^p \geq d_t^p$ ***do***
		  $i \leftarrow 1$
		  while $d_t^p > 0$ ***do***
		  $tmp \leftarrow s_i$
		  $s_i \leftarrow s_i - d_t^p$
		  $d_t^p \leftarrow d_t^p - tmp$
		  if $s_i < 0$ ***do***
		  $s_i \leftarrow 0$
		  end if
		  $i \leftarrow i+1$
		  end while
		else do
		  $Backorder_t^p = d_t^p - I_t^p$
		  $S_t^p \leftarrow ()$
		end if
		if $r^p > I_t^p$ *and status* = False ***do***
		  status ← True
		  $o_t^p \leftarrow Q^p$
		end if

Source-code implementation in Python 3.7 is available in the GitHub repository (Jackson, 2019).

5. Defining the Number of Replications

Due to the fact that some input variables of the model are random, namely in considered numerical example D~N(μ, σ²), L~N(μ, σ²) and A~Exp(λ), the output (net profit) is also a random variable under some distribution. Thus, a persistent question arises "how many runs of the model are adequate to produce a meaningful prediction?". We approach this problem using a method based on confidence intervals (12).

$$n = \left(\frac{z_{\alpha/2}}{w} CV\right)^2 , \quad (12)$$

where: n is a minimum number of model's replications to achieve desired confidence interval width w (expressed as a multiplier by the mean) for model with a coefficient of variation CV and (Byrne, 2013).

We generated a set of random inputs in feasible range. After that the simulation model was replicated 10.000 times. Such that each single replication had the same inputs and covered 120 modelling days. With such a procedure a sample was generated (Fig. 3), which was used to calculate a coefficient of variation and test normality (Table 2).

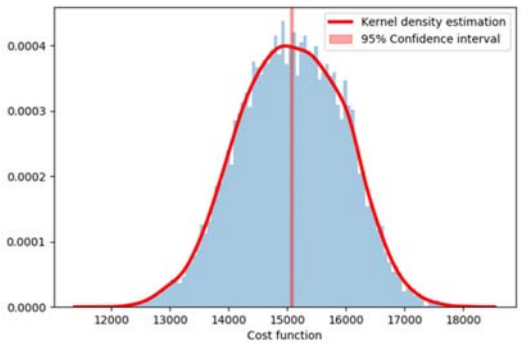

Figure 3. The empirical distribution of the output variable

In order to test normality, we have chosen the Anderson-Darling test, which is recognized as one of the most reliable, especially in cases of samples with more than 100 observations (Razali and Wah, 2011).

Table 2. The Anderson-Darling normality test

Statistic	Critical value	Significance level
7.16	0.787	0.5

We assume the confidence level of 95% and corresponding $z_{\alpha/2} = 1.96$. Calculating the coefficient of variation using sample mean and variance CV = 1512.8/9968.2 = 0.15, we have decided to work with a confidence interval of length 498.4 (5% of the mean) running each simulation 35 times to take the average output.

6. ANN-Based Metamodel

In the numerical experiment we consider the ICS that operates with 10 products and comprises 150 input parameters. We have begun with generating 5 datasets using Monte Carlo sampling. Firstly, we have generated random inputs in feasible range. Secondly, simulation model was run 35 consecutive times with these inputs. After that, the average output value is calculated. Each of 5 generated datasets contains exactly 1000 observations (Fig. 4).

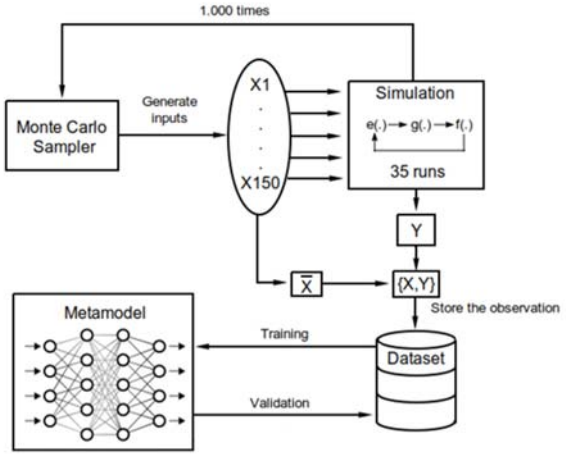

Figure 4. Outline of the experiment

The MLP-based metamodel comprises 3 hidden layers (30/30/10 neurons) and 5371 trainable parameters. It is worth to note that this research is not focused on neural architecture search. In this regard, the network architecture was derived using "trial and error" method and may be not optimal. The metamodel is trained in 200 epochs with the batch size of 20, and successfully validated for all 5 datasets with10-fold cross-validation (Fig. 5.).

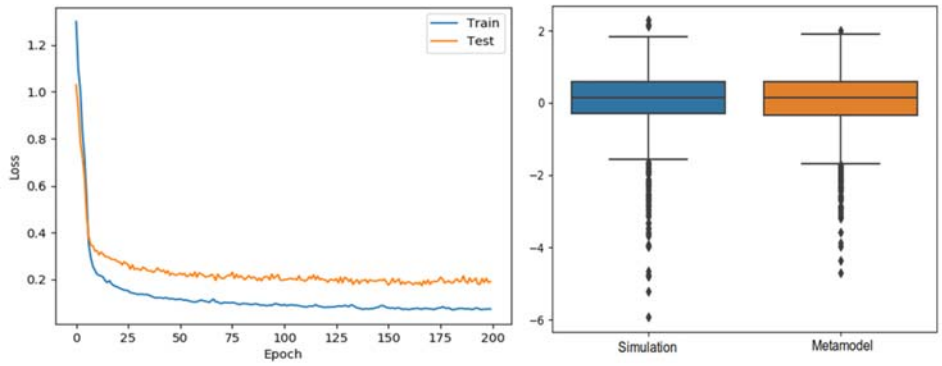

Figure 5. The learning procedure and predicted output (standardized)

As Table 3 demonstrates the MLP-based metamodel was capable to generalize from a given sample and learn all the nonlinearity within the original simulation model. However, there is an upper-bound for accuracy due to inevitable presence of stochastic noise in training sample. On the other hand, this noise may be reduced by increasing the number of simulation's runs, which puts forward a classical trade-of between performance and computational time. Additionally, such overfitting-preventing techniques as regularization and dropout look quite promising and will be reviewed in future research.

Table 3. Accuracy of the metamodel

	Training								Test	
Exp. №	**df1**	**df2**	**F-statistic**	**F-critical**	**p-level**	**R^2**	**adj. R^2**	**SEE**	**R^2**	**adj. R^2**
1	149	850	2.54	1.22	0.95	0.89	0.87	0.22	0.85	0.82
2	149	850	1.98	1.22	0.95	0.84	0.82	0.21	0.81	0.79
3	149	850	2.62	1.22	0.95	0.90	0.87	0.22	0.86	0.83
4	149	850	2.47	1.22	0.95	0.89	0.86	0.22	0.84	0.81
5	149	850	2.51	1.22	0.95	0.89	0.87	0.22	0.84	0.81

7. Conclusions

Summing up, MLP is capable to "learn" and generalize complex nonlinear relations between simulation variables and, thus, may be efficiently applied for metamodelling of real-world inventory-control systems. Despite the fact that ANN is generally robust to stochastic noise in a training sample, a MLP-based metamodel will anyway have some upper-bound for accuracy. However, the amount of noise may be controlled by increasing the number of simulation's runs within the restrictions of computational budget. Moreover, both numerical and categorical variables can be used as inputs and outputs. Besides that, MLP-based metamodel equipped with ReLU needs much less computational time and memory than the corresponding simulation model, which makes it extremely useful for simulation-based optimization.

In order to make the proposed metamodelling approach more tailored to the industrial needs, in future research it worth to pay attention on automatic neural architecture search and such overfitting-preventing techniques as regularization and dropout.

References

1. Bellman, R. (1957) *Dynamic Programming*. Princeton, Princeton University Press.

2. Blanning, R.W. (1975) The construction and implementation of metamodels, *Simulation*, 24, 177–184. DOI:10.1177/003754977502400606.
3. Buffa, E.S. and Taubert, W.H. (1972) *Production-inventory systems planning and control* (NTIS No. 658.4032 B8).
4. Byrne, M.D. (2013) How many times should a stochastic model be run? An approach based on confidence intervals. In: *Proceedings of the 12th International conference on cognitive modelling*, Ottawa, July 2013.
5. Cawley, G.C. and Talbot, N.L. (2010) On over-fitting in model selection and subsequent selection bias in performance evaluation. *Journal of Machine Learning Research*, 11(Jul), 2079–2107.
6. Cybenko, G. (1989) Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4), 303–314.
7. Domschke, W., Drexl, A., Klein, R. and Scholl, A. (2015) *Einführung in Operations Research.* 9th Ed., Berlin, Heidelberg: Springer Gabler.
8. Duan, Q. and Liao, T.W. (2013) Optimization of replenishment policies for decentralized and centralized capacitated supply chains under various demands. *International Journal of Production Economics*, 194–204.
9. DynamicAction and IHL-group, (2015) Research Study: Retailers and the Ghost Economy $1.75 Trillion Reasons to be Afraid.
10. Farhat, J. and Owayjan, M. (2017) ERP Neural Network Inventory Control. *Procedia computer science*, 114, 288-295.
11. Glorot, X., Bordes, A. and Bengio, Y. (2011) Deep sparse rectifier neural networks. In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, Lauderdale, FL, USA, June 2011, pp. 315-323.
12. Haykin, S.S. (2009) *Neural networks and learning machines*. Pearson (ISBN-10: 0131471392).
13. Hochmuth, C.A. and Kochel, P. (2012) How to order and transship in multi-location inventory systems: The simulation optimization approach. *International Journal of Production Economics*, 140, 646–654.
14. Hornik, K. (1991) Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2), 251–257.
15. Hurlimann, T. (2007) *Index notation in mathematics and modelling language LPL: theory and exercises*. Department of Informatics University of Fribourg.
16. IHL-group and Buzek G. (2015) *Research Study:* We Lost Australia! Retail's $1.1 Trillion Inventory Distortion Problem.
17. Iverson, K.E. (1962) A programming language. In: *Proceedings of the spring joint computer conference.* ACM, May 3, 1962, pp. 345-351.
18. Jackson, I. (2019) *GitHub repository "metainventory"* - https://github.com/Jackil1993/metainventory, last accessed 2019/04/05.
19. Jackson, I. and Tolujevs, J. (2019) The Discrete-Event Approach to Simulate Stochastic Multi-Product (Q, r) Inventory Control Systems. *Information Modelling and Knowledge Bases XXX*, 312, 32–39.
20. Jackson, I., Tolujevs, J. and Reggelin, T. (2018) The Combination of Discrete-Event Simulation and Genetic Algorithm for Solving the Stochastic Multi-Product Inventory Optimization Problem. *Transport and Telecommunication Journal*, 19(3), 233–243.
21. Jad, F. and Owayjan, M. (2017) ERP Neural Network Inventory Control. *Procedia Computer Science*, 114, 288–295.
22. Jalali, H. and Nieuwenhuyse, I.V. (2015) Simulation optimization in inventory replenishment: a classification. *IIE Transactions*, 47(11), 1217–1235.
23. Kingma, D.P. and Ba, J. (2014) Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980.*
24. Law, A.M. and Kelton, W.D. (2000) *Simulation modelling and analysis*. New York: McGraw-Hill.
25. Lechevalier, D., Hudak, S., Ak, R., Lee, Y.T., and Foufou, S. (2015) A neural network meta-model and its application for manufacturing. In: *Proceedings of the IEEE International Conference on Big Data*, Santa Clara, USA, 2015.
26. Lin, Y., Shie, J. and Tsai, C. (2009) Using an artificial neural network prediction model to optimize work-in-process inventory level for wafer fabrication. *Expert Systems with Applications* 36(2) 3421–3427.
27. Merkuryeva, G. (2004) Metamodelling for simulation applications in production and logistics. In: *Proceedings of the Sim-Serv Workshop: Roadmap of simulation in manufacturing and logistics,* pp. 1-6.

28. Prestwich, S.D., Tarim, S.A., Rossi, R. and Hnich, B. (2012) A neuroevolutionary approach to stochastic inventory control in multi-echelon systems. *International Journal of Production Research*, 50, 2150–2160.
29. Razali, N.M. and Wah, Y.B. (2011) Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests. *Journal of statistical modelling and analytics*, 2(1), 21–33.
30. Tolujev, J., Lorenz, P., Beier, D. and Schriber, T.J. (1998) Assessment of simulation models based on trace-file analysis: a metamodeling approach. In: *Proceedings of the Winter Simulation Conference.* IEEE. December 1998, pp. 443-450.
31. Tsai, S.C. and Zheng, Y.X. (2013) A simulation optimization approach for a two-echelon inventory system with service level constraints. *European Journal of Operational Research*, 229, 364–374.
32. Winskel, G. (2010) Set theory for computer science. *Unpublished lecture notes*.