Available online at www.sciencedirect.com

ScienceDirect

Procedia Manufacturing 39 (2019) 1793–1800

Procedia MANUFACTURING

www.elsevier.com/locate/procedia

25th International Conference on Production Research Manufacturing Innovation: Cyber Physical Manufacturing
August 9-14, 2019 | Chicago, Illinois (USA)

An Entity Embeddings Deep Learning Approach for Demand Forecast of Highly Differentiated Products

Davide Mezzogori*, Francesco Zammori

University of Parma, Parco Area delle Scienze 181/A, Parma 43100, Italy

Abstract

The paper deals with Deep Learning architectures applied to demand forecasting in a complex environment. The focus is on a famous Italian Fashion Company, which periodically performs a sales campaign, to presents its new products' line and to collect customers' orders. Although production follows an MTO strategy, fabrics must be purchased in advance and a forecasting system is required to predict the total quantity sold for each product, at the early stages of the campaign. Due to high product variability, the forecasting system must consider products' similarities and the evolution of customers taste. Additionally, customer and product data are mostly described by categorical variables (hard to reconcile with a predictive task) and, unfortunately, time-series techniques cannot be used because of a sparse dataset. Given these criticalities, we propose an end-to-end approach based on Deep Neural Networks and on Entity Embeddings. A first neural network is trained to predict the total quantity of a given product ordered by a specific customer. Different Embeddings are learned for each customer and product categorical attribute. This gives the network the ability to effectively learn the complex and evolving relationships between products characteristics and customers taste. Next, freezing the learned product's embeddings, a second Recurrent Neural Network is trained to predict the total amount ordered for a given product, incorporating real-time data of customers' orders of the ongoing sales campaign. Ten years of sales have been analyzed and the approach, tested on unseen sales campaigns, has outperformed the forecasting algorithm currently adopted by the fashion firm.

Keywords: Machine Learning; Deep Learning; Forecasting; Embeddings

* Corresponding author. Tel.: +39-0521-905887. *E-mail address:* davide.mezzogori@unipr.it

1. Introduction

Recent initiatives, namely Industry 4.0, have raised awareness within industry practitioners of data analytics and of intelligent algorithms, as an essential tool to support decision-making processes, and to gain competitive advantages over competitors. As said by Kusiak [1], as smart manufacturing can make industry more efficient, profitable and sustainable, the design of forecasting models, leveraging historical data, can help companies to predict future sales and to better manage the production and procurement process.

Moreover, recent developments in the field of Machine Learning and, above all, of Deep Learning, enable researchers to study the application of more advanced computational techniques to deal with problems that are generally tackled with traditional statistical techniques. Specifically, having the ability of learning representations autonomously, Deep Learning can extract knowledge directly from raw data [2]. One of the main application fields is demand forecast, especially in case of fashion retail, where a high demand uncertainty creates many hurdles, for an efficient logistics management [3]. In literature there are many works that faced the problem of forecasting demand in the fashion sector where, to cope with fashion trend and market response, demand is highly variable, erratic and subjected to unpredictable peaks [4]. However, as reported by Gutierrez [5], traditional statistical demand forecasting systems do not always capture non-linear patterns present in historical data of fashion products. Conversely, recent literature on nonlinear models has shown that neural networks can be used as promising and versatile tools for forecasting. Interesting results of neural networks applied to demand forecasting can be found in [6, 7] and, relatively to the fashion market, in [8, 9]. However, these works, while not old, do not make use of the last innovations in the field of Deep Learning: for example, the focus is on the use of evolutionary research techniques (to get the best architecture), or on feature engineering tasks. Our believe is that recent developments, in the field of deep learning, could substantially improve forecasting results obtained so far.

This work belongs to this stream of research and focuses on the implementation of a demand forecast system, for a famous Italian fashion company. The company operates in an MTO way, but raw materials must be purchased in advance. Anticipating as much as possible raw materials' supply, to streamline the production planning and scheduling, is a critical success factor. Hence, an accurate forecasting system is strongly required.

It must be noted that, in contrast to the preceding mentioned works, which focused on the problem of forecasting demand at the retail level, that is the demand of end consumers, the aim is to predict the final quantity sold to business customers, at the end of a sales campaign. The forecasting system is based on neural networks and presents several peculiarities, as the forecasting problem poses two main challenges, intrinsic to the reference market. The first one, exogenous to the company, is linked to the continuous changes in tastes of customers, dictated by fashion trends. The second one, a direct consequence of the first, is the need to respond to these never-ending changes, with collections of products that are always new and different from the preceding ones. Consequently, products can be either very similar or very different, with respect to products proposed in the same year or in past years and it is thus very difficult to tackle this problem as in a classical time-series framework, given that, for current products, a time series of sales is not available.

To faces these challenges, the proposed approach exploits two different neural networks: the first aims to reconstruct possible similarities between current products and those commercialized in the past, so that time-series for the current products can be "reconstructed" using the quantities sold in the past of the most similar products. The second neural network aims to predict, on the basis of these reconstructed time-series, the final quantity sold at the end of a sales campaign.

The rest of the paper is structured as follows. Section 2 contextualizes the problem faced in this paper. Section 3 describes the proposed forecasting system, broken down in its three main parts. Lastly. Section 4 reports experimental results and Section 5 draws conclusions and outlines possible issues for future researches.

2. Contextualization

The goal of the forecasting system is to predict the total quantity that will be ordered, for each commercialized fashion garment or accessory, at the end of a sales campaign. Each campaign follows a design phase and it is used to allow customers to review the new collection and, next, to collect customers' orders. Although the company operates according to a Make to Order (MTO) logic, for operational purposes, it uses the orders collected in the very first phase of the sales campaign, to forecast the total quantity that will be sold for each new product. In this way, the company can anticipate the supply of raw materials, required to manufacture the ordered clothes. This strategy makes it possible to distribute production on a longer time period, thus avoiding capacity issues and reducing probability of late deliveries or, even worse, out-of-stock at the retailers.

At present, the company makes use of a heuristic algorithm based, mainly, on the expected total sales evaluated by human experts. The forecasting task is highly challenged by the specific characteristics of the market, which requires that new products conform to the ever-changing market trends. Consequently, the company creates different products every year, either modifying old ones, introducing completely new garments, or removing old offerings. This implies that, new collections are always made of totally new products, for which a time-series of sales does not exist. This issue poses a huge problem because, in absence of time series, neither statistical nor Machine Learning techniques can be used to generate accurate forecasts. Nonetheless, human experts are able to trace, based on the physical appearance, a chain of similarities linking products that are currently on sale, with the ones offered during the past seasons. The main hypothesis is that physically similar products may have similar sales patterns, and that these similarities could be exploited to reconstruct a *pseudo* time-series (for each current product) based on the past sales of the most similar products.

Another challenge is due to the way in which products are described (i.e., codified) in the company's Data Base. Unfortunately, products are descripted, mostly, by categorical attributes, a condition that makes it difficult to use any kind of metric to trace similarities among current and past products, based on their physical attributes. The final hindrance is that, as the company operates in the global marketplace, the customer base varies considerably, about 15-20%, from year to year. Moreover, as well as evolving their tastes over time, customers can have very different preferences, for example due to cultural reasons.

3. Proposed Approach

To address the criticalities highlighted in the previous section, we propose a demand forecast model based on two sequential steps. In the first one, similarities among new and old products are traced back, and they are leveraged to construct, for each new product, a *pseudo* time-series. i.e., given a certain product P_0 offered at present time t_0, its time series is created using as quantity q_{-i} the orders of the most similar products in previous years $t_{-1}, t_{-2}, \ldots, t_{-n}$. In the second step, time-series are used, together with the orders collected in the very first phases of the sales campaign, to predict the required quantity of the new products. These steps are preceded by an analysis of the customer base, used to identify, by geographical area and/or by type of product, a subset of reference customers, i.e., the ones most aligned with the overall purchasing behavior. Each of these steps are described in the following sub-sections.

3.1. Analysis of customer base

The objective is to identify a subset of relevant customers, (to focus on in the next phases), excluding those that are relatively unstable and unnecessary for further analysis. Mainly, with "lack of stability" we refer to customers which are relatively new (i.e., customers who have only been active in recent years), or to customers whose tastes have changed a lot both in terms of purchased quantities and/or in terms of style. Once unstable customers have been eliminated, the residual subset of the customer base will be used to reconstruct the time series of the products included in the current collection. Stable customers play another key role, as only orders placed by them in the early phases of the sales campaign will be used in the demand forecasting system.

To identify such subset, the following steps were used: *i)* customers are divided into three geographical areas, according to the country in which they sell to end consumers, namely Europe, Asia, and North America; *ii)* for each geographical area, a Pareto analysis is carried out and the subset of customers corresponding to the 80% of the total quantities sold is found; *iii)* the last ten years are considered and for each geographical area A the total quantities $Q_{A,t}$ and the total quantities $Q_{A,C,t}$ for each product class C sold in year $t \in [1, 10]$ are computed. Next, dividing $Q_{A,C,t}$ by $Q_{A,t}$, a reference vector $\vec{\boldsymbol{r}}_{A,C}$ is defined for each geographical area A and for each product class C:

$$\vec{\boldsymbol{r}}_{A,C} = \langle r_t^{A,C} \rangle, \quad t \in [1, 10] \tag{1}$$

with $r_t^{A,C}$ defined as follows:

$$r_t^{A,C} = \frac{\sum_{\{i \in A, j \in C\}} q_{i,j,t}}{\sum_{\{i \in A\}} q_{i,t}} = \frac{Q_{A,C,t}}{Q_{A,t}} \tag{2}$$

As it can be seen, $\vec{\mathbf{r}}_{A,C}$ represents the percentage incidence of each class *C* with respect to the total sales in a given area *A*.
Furthermore, *iv)* for each stable customer a similar analysis is conducted, to obtain the following reference vector $\vec{\mathbf{r}}_{i,C}$:

$$\vec{\boldsymbol{r}}_{i,C} = \langle r_t^{i,C} \rangle, \quad t \in [1, 10] \tag{3}$$

where:

$$r_t^{i,C} = \frac{\sum_{\{j \in C\}} q_{i,j,t}}{q_{i,t}} \tag{4}$$

This vector describes, for each year *t*, the amount of goods belonging to class *C*, purchased by customer *i*, relatively to the total amount purchased by the same customer in the same year.
Finally, the Euclidean distance between each reference vector $\vec{\mathbf{r}}_{i,C}$ and $\vec{\mathbf{r}}_{A,C}$ is computed as in Eq. (5) to obtain a score for each customer *i* in each class *C*.

$$score_{i,C} = \left\| \vec{\boldsymbol{r}}_{A_i,C} - \vec{\boldsymbol{r}}_{i,C} \right\|_2 \tag{5}$$

The lower is the score (distance) the more customer *i* is aligned with the purchasing behavior of the overall customer base. Thus, for each area and product class, customers are sorted by score in ascending order, and only the first 50 percent is held and aggregated among geographical area. By operating in this way, it is possible to identify, for each product class, a set of reference customers, that will be used in the following phases.

3.2. Entity Embeddings

As mentioned before, the prediction task is hindered by the lack of time-series and by difficulties in tracing back similarities among new and past products. In particular, the need to trace similarities, with the aim of building *pseudo* time-series to use in prediction, is hampered by the fact that most of the attributes available to characterize these products are categorical, therefore difficult to reconcile with similarity metrics. Indeed, in the present case, products are described by 7 categorical variables, 7 binary variables, and 2 continuous variables. Some of the categorical variables describe, for example, the type of product, its color, its brand and other descriptive characteristics.

A first approach to trace similarities between products could be that to partition the products' set based on the values of categorical and binary variables. Within each partition, the search for similar products could be carried out measuring similarities (distances) between products relatively to their continuous variables' values. This approach, however, could reveal to be too restrictive, as it could be argued that similarities should also be traced between goods with different categorical attributes, such as different colored clothing, as similarities should measure purchasing behavior and not be bound to certain sub-sets identified by the combination of physical characteristics.

It would be ideal to automatically learn the best representation of each product, present and past, that correctly encodes the information of its attributes (categorical and not) according to purchase history, and, based on such representation, implement a nearest neighborhood search of the most similar products in the past given a current one.

To successfully tackle this issue, a neural network is implemented with the aim to learn an efficient and effective representation of each categorical variable, and thus of each past and future product. The neural network is trained in a supervised fashion with respect to the order history, meaning the network is trained to predict, given a tuple (product, customer), the quantity sold of the given product to the given customer.

To efficiently exploit the information encoded in the categorical variables (both product attributes and customer attributes), we leveraged the concept of Entity Embeddings, described by Guo and Berkhahn [10], in which categorical variables are mapped into Euclidean spaces. A traditional alternative to Embedding is One-Hot-Encoding, in which each categorical variable with cardinality C is encoded in a vector of C binary entries. However, such representations suffer from many limitations, in particular, they are hardcoded, sparse (mostly made of zeros), and high-dimensional representations (same dimensionality as the cardinality of the encoded categorical variable).

Embeddings, on the other hand, are more compact, lower-dimensional and dense representations, learned directly from data. Entity embeddings works by mapping each value of a categorical variable to a d-dimensional continuous vector. Operatively, they are equivalent to a hidden layer of d-linear neurons following the input layer, where the input is one-hot encoded. Essentially, they are implemented as a lookup table of a matrix $\boldsymbol{E}$ of dimensions $C\times d$, which corresponds to the weight matrix of the linear d-neurons hidden layer. The number of rows of this matrix is equal to the cardinality of the encoded categorical variable, and the number of columns d ($<< C$) is a hyper-parameter which the user can set to get a more compact data representation, while preserving all relevant information. Operatively, given a sample (product, customer) *i)* for each categorical variable describing either the product or the customer a corresponding one-hot encoding is built; *ii)* the corresponding embedding matrix $\boldsymbol{E}$ is pre-multiplied by the one-hot encoding row vector, essentially performing a lookup in the rows of the embedding matrix $\boldsymbol{E}$; *iii)* this row-vector is then concatenated with the rest of the embedded vectors and other continuous inputs; *iv)* this concatenated representation of the tuple (product, customer) is then fed to the rest of the network, and tweaked, as other parameters, during the back-propagation of errors. Each embedding row-vectors, randomly initialized, is thus continuously modified and learned based on the task at hand.

As a clarifying example, let's assume products can be produced in *10* different colors. The cardinality of the categorical variable *color* is thus $C = 10$. Suppose the user chose to embed the variable color in a d-dimensional space with $d = 2$. The entity embedding matrix $\boldsymbol{E_{color}}$ is thus of dimension *10* x *2*, where to each color correspond a *2*-dimensional real-valued row-vector. Suppose at a given time a product red colored is given as a sample. The one hot encoding version $\vec{\boldsymbol{x}}_{\boldsymbol{one-hot}}$ is given by the sparse vector [1, …, 0]. The embedded representation of the color $\vec{\boldsymbol{x}}_{\boldsymbol{color}}$, is obtained pre-multiplying the entity embedding matrix with the one-hot encoded vector as follows:

$$\vec{\boldsymbol{x}}_{\boldsymbol{color}} = \vec{\boldsymbol{x}}_{\boldsymbol{one-hot}} \cdot \boldsymbol{E}_{\boldsymbol{color}} = [1, \dots, 0] \cdot \begin{bmatrix} 0.42, & 1.41 \\ \vdots & \vdots \\ 2.71, & 1.61 \end{bmatrix} = [0.42, \ 1.41] \tag{6}$$

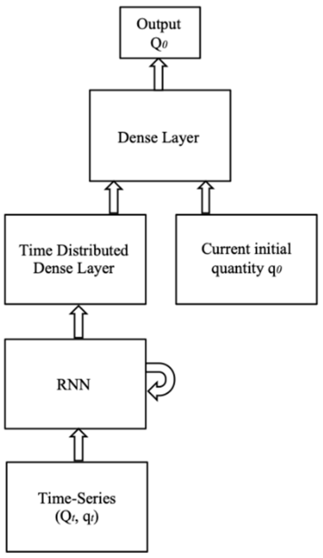
The given representation, extracted as a lookup in the entity embedding matrix, is learnable, as it is the whole matrix $\boldsymbol{E}$. This representation is then concatenated with all the other embedded version for each categorical variable and with the continuous and binary ones.

The final network architecture is described in the experiment section.

3.3. Demand Forecasting

The demand forecast procedure leverages both on the learned similarities (between products) and on the analysis of the customer base. First, for all past products i produced at time t, the embedded representation $\hat{P}_{i,(-t)}$, is obtained using the entity embedding network. Then, given a product P_0 belonging to the current collection and for which no past sales data is available, its embedded representation $\hat{P}_0$ is obtained, too. Based on the embedded representations of the current product $\hat{P}_0$ and of old products $\hat{P}_{i,(-t)}$, a similarity search is carried out. This is done computing the Euclidean distance between $\hat{P}_0$ and each embedded representation $\hat{P}_{i,(-t)}$ of past products. For each year, the most similar product i^* is found and two quantities are calculated: the total quantity sold $Q_{i^*,-t}$ and the initial quantity $q_{i^*,-t}$ ordered by reference customers (of the product class of the current product P_0), previously determined. In this way, a 2-dimensional time-series is generated made of the total final quantity $Q_{i^*,-t}$ and the initial quantity $q_{i^*,-t}$ (for the most similar product) for each preceding year. Additionally, to the generated pseudo-time series, the initial quantity q_0 sold in the first phases of the sales campaign, of the current product P_0, is used as a secondary forecasting input.

These inputs are processed by a neural network with multiple inputs and multiple modules. The first module consists of a Recurrent Neural Network (RNN) module that processes the time series. The second module, which fed on the output of the first and on the secondary input, is a feed-forward fully-connected module in which the output of the RNN network is concatenated with the secondary input to produce the final prediction. It must be noted that the RNN



module includes a recurrent cell, which outputs a sequence of hidden states, one for each timesteps, followed by a dense layer applied to every hidden state. The final architecture is shown in figure 1.

Fig. 1. Demand forecasting neural network architecture.

4. Experimental results

Experiments have been conducted using ten years of order history. Each year consist of two sales campaigns. The neural networks were implemented using Python© 3.6 and the Keras© module, based on Tensorflow©.

The Entity Embeddings neural network was trained over the whole order history, except for the last sales campaign of the last year, which was kept as test set. Thus, Entity Embeddings are learned over the past sales campaigns, and used to project both past and current (unseen) products of the last sales campaign.

The Entity Embeddings neural network architecture consists of 15 inputs for categorical variables, each followed by its corresponding Embedding matrix, 9 inputs for Boolean variables and 2 inputs for continuous variables.
Table 1 shows, for each categorical variable, the corresponding cardinality and embedding dimensionality. Attributes name are anonymized due to privacy issues.

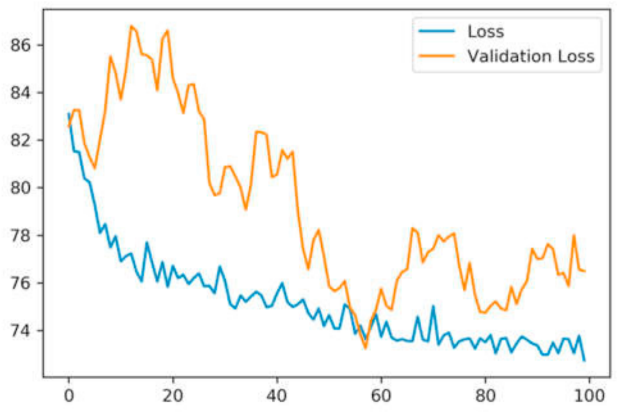
After the inputs' concatenation, a set of 3 dense hidden layers follows, with 128, 64 and 32 neurons each, and based on the *relu* activation function. Finally, an output linear layer with one neuron is used to generate the final forecast for each (customer-product) tuple. We remember that the forecast made by the Entity Embedding Neural Network is not the one that will be used by the company. It is, in fact, just a "feedback signal" needed to train the network to recognize similarities (both physical and commercial) among the products. The Entity Embedding network was trained with the *Adam* optimizer, with a batch size of 2048, for a total of 10 epochs; the mean squared error was used as loss function.

Concerning the RNN network used to process pseudo-time series and to generate the real forecast, preliminary tests were made focusing on five specific product class (trousers, skirts, jackets, sweaters, shirts) to control variability otherwise introduced by considering different kinds of products altogether. This setting was used both to reconstruct the *pseudo* time-series and thus to train the RNN-based neural networks. As a result, a total of around 1020 products per class and corresponding time-series have been considered. Of these, 80% were used for training, the remaining 20% for test. The RNN cell contains 16 neurons, and it is followed by a time-distributed dense layer of 8 neurons, with a *relu* activation function. After the concatenation of this dense layer and the secondary input, two dense layer follows, with 8 neurons each and with *relu* activation function. A final output linear layer closes the topology. The training is done over 100 epochs, with early stopping with a patience of 50. The optimizer implemented is *Adam* with learning rate equal to 0.001 and with mean absolute error loss function. The training loss and validation loss are shown in figure 2, relatively to the trousers class.

Table 1. Entity Embeddings cardinality and dimensionality

Attribute	**Cardinality**	**Dimensionality**
X1	3	2
X2	7	2
X3	49	8
X4	6	2
X5	20	4
X6	14	6
X7	14	6
X8	55	8
X9	5	2
X10	5	2
X11	5	2
X12	5	2
X13	112	8
X14	112	8
X15	12	4

Fig. 2. Loss and validation loss trend of demand forecasting network.



The MAPE errors over the training sets are reported in table 2, showing very promising values, considering that, at present, the forecasting model of the fashion firm has a MAPE ranging from 10 to 15%, on average.

Table 2. MAPE results for each product class

Class	MAPE
Trousers	3.5%
Skirts	3.9%
Jacket	4.1%
Sweaters	3.4%
Shirts	5.3%

5. Conclusions

This article presented a demand forecast model for a famous Italian fashion company. The need to have a system for forecasting demand stems from the dynamics of the market in which the company operates, purely MTO, but where it is strategic to anticipate as much as possible the production of clothes, in order to avoid problems of production capacity and delays to the retailers. The problem was firstly addressed through a neural network based on Entity Embeddings, to reconstruct chains of similarities between current and past products, thereby reconstructing a historical series against which a second recurrent neural network can be trained, also leveraging the information of partial purchases during the first phases of the sales campaign (during which orders are collected). Moreover, these partial purchases are filtered based on a preceding customer analysis. The proposed system has been tested in a preliminary way on a small group of products. The results proved to be satisfying and prompt the extension to other product groups for further field testing. Future developments could concern the study of alternative architectures of the recurrent network, such as Time Delayed Neural Networks, or architectures based on attention mechanisms. In addition, the implementation of a specialized forecasting model by product type could be studied, too.

References

[1] A. Kusiak, Smart manufacturing must embrace big data, Nature News, 544.7648 (2017), 23-25.

[2] Y. LeCun, Y. Bengio, G. Hinton, Deep learning. Nature, 521.7553 (2015), 436-444.

[3] J. H. Hammond, Quick Response in the Apparel Industries. Harvard Business School, Cambridge, MA (1990).

[4] M.E. Nenni, L. Giustiniano, L. Pirolo, Demand Forecasting in the Fashion Industry: A Review. International Journal of Engineering Business Management 5 (2013), 37.

[5] R.S. Gutierrez, A. Solis, S. Mukhopadhyay, Lumpy Demand Forecasting Using Neural Networks. International Journal of Production Economics, 111 (2008), 409-420.

[6] P.C. Chang, Y.W. Wang, C.H. Liu, The development of a weighted evolving fuzzy neural network for PCB sales forecasting. Expert Systems with Applications, 32(1) (2007)., 86-96.

[7] S.H. Ling, Genetic Algorithm and Variable Neural Networks: Theory and Application, Lambert Academic Publishing, (2010) German.

[8] Y. Yu, T. Choi, C. Hui, An intelligent fast sales forecasting model for fashion products. Expert Systems with Applications, 38(6) (2011), 7373–7379.

[9] K.F. Au, T.M. Choi, Y. Yu, Fashion retail forecasting by evolutionary neural networks. International Journal of Production Economics, 114(2) (2008), 615–630.

[10] G. Cheng, F. Berkhahn, Entity embeddings of categorical variables. arXiv preprint arXiv:1604.06737 (2016).