ERASMUS UNIVERSITY ROTTERDAM

Erasmus School of Economics

Master Thesis Data Science & Marketing Analytics

Learning product associations by using a Generative Adversarial Network

Supervisor: F.J.L. van Maasakkers

Second Assessor: dr. J.E.M. van Nierop

Name student: Joeri Murk

Student ID number: 456131

Date final version: 24-7-2020

Abstract

A scalable model is proposed to capture complex product associations for grocery-like (online) shops. By using a Generative Adversarial Network with a generator that creates baskets where the products occur independently, a discriminator needs to learn product associations to distinguish this data from real data. Different evaluation methods are designed to test the model. We can conclude from the results that the model performs well. This emphasizes the potential of this model in the field of market basket analysis and recommendation systems. Further research could be done in optimizing this model, especially on unbalanced data sets.

Contents

1 Introduction **3**

2 Literature **4**

2.1 Recommendation Systems & Market Basket Analysis 4

2.2 Generative Adverserial Networks 9

3 Methodology **10**

3.1 Problem description 10

3.2 Idea behind the model 10

3.3 Generating fake data 11

3.4 Discriminator 12

3.5 Optimization 14

3.6 Validation model 16

3.7 Accuracy measures 17

3.8 Benchmarks 18

3.9 Product clusters 18

4 Data **20**

5 Simulation **22**

5.1 Data 22

5.2 Results 24

6 Results **26**

6.1 Results both models 26

6.2 Aisle mappings 28

6.3 Product mappings 30

7 Conclusion **31**

A Appendix **36**

A.1 Optimization models 36

1 Introduction

The search for relationships between products is a meaningful challenge for companies in the retail industry. Knowing that certain products are often bought together, could help to improve promotions, the lay-out of products in an aisle and the lay-out of aisles in a store. These relationships between products can be defined as product associations. The product association $\{p_a, p_b\} \rightarrow \{p_c\}$, means that if a customer buys product p_a and product p_b, this customer is likely to also buy product p_c. With this information, the supermarket could for example adjust their store lay-out by placing these products close to each other, to make the shopping experience better for their customers.

Product associations are not only useful to offline stores, but also to online stores. Creating product categories or improving promotions are equally important to online stores as they are to offline stores. Additionally, online stores could use recommendation systems to make a customer journey easier. These systems are suggesting products that could be interesting for customers. Many models have been tested for recommendation tasks (Koren et al., 2009; Rendle et al., 2010; Wang et al., 2015; Yu et al., 2016). A short review of these models can be found in the Literature section. Most of these models are focused on the personal preferences of customers. If someone wants to buy something at Amazon for example, Amazon may know from browsing behavior or previous purchases what this customer likes. With this info, Amazon can provide the customer a personalized homepage with products that might be interesting for him. Also, if a search results of a customer shows many products, the products can be ranked based on what this customer probably will like most. These recommendation systems could have much added value because people already get to see what they probably like and they will not be confronted with many uninteresting products.

In grocery shopping, recommendation systems could also have some added value. However, there is a big difference between grocery shopping and shopping at a non-grocery store. Customers generally purchase more products in a visit to a grocery store, than at a non-grocery store. Therefore, a recommendation system that makes it easier to find all products, would have much added value in an online grocery store. A way to do this, is by recommending items that are associated with products that a customer already has in its basket. This will help the customer navigate easily through the website without having to look up every single product. Also, by recommending products that are associated with products already in the baskets, customers will less likely forget items that are complementary. Furthermore, the recommendations will be more diverse than recommending based on personal preferences, since the recommendations based on associations can change when adding items to a basket.

The methods used for finding association rules are mostly data mining techniques. This means that the algorithm will count the occurrences of certain product combinations and decide based on that. However, product associations can become complex, especially when the data set is large. By counting the occurrences of certain product combinations, these complex relationships will not be uncovered. Therefore, we research the use of a scalable model that learns these complex relationships.

The model that is used in this paper, is the Generative Adversarial Network (GAN) (Goodfellow et al., 2014). This rather new network consists of two sub-networks, one sub-network that generates data and another that tries to distinguish the generated data from the real data. The idea of generating fake data and training a neural network in trying to distinguish this fake data from real data is very promising in finding associations. The generator can be constructed so it is untrainable and generates a data set where the products are independent. By letting the discriminator distinguish this generated data from the real data, the discriminator is forced to learn relationships between products. The use of this model to learn complex associations is tested in the current paper. Additionally, the scalability of the model is studied by including a large data set.

Firstly in the Literature Section, an overview is provided of the most important work done related to recommendation systems, finding associations and the GAN. Secondly, the model and evaluation methods will be explained in the Methodology section. Next, the data set used will be presented along with some descriptive statistics in the Data Section. To test some features of the model by using a data set with a know structure, a Simulation section is added. After the Simulation section, the results of the model on the data sets indicated in the Data section will be given in the Results section. Finally, the research will be summarized and a conclusion will be drawn together with a discussion in the Conclusion Section.

2 Literature

The literature review is divided into two parts. The first part covers the research already done related to recommendation systems, whereas the second part focuses on the applications of the GAN.

2.1 Recommendation Systems & Market Basket Analysis

Online retail is growing, retailers shift from selling their products in offline stores to selling their products in a web shop. This shift means that recommendation systems are becoming more and more important. Recommendation systems are suggesting a set of products that could be

interesting for a customer. They are designed so customers will not be overwhelmed with many products, but get a personalized product set presented (Chen, 2011).

Most recommendation systems rely on content-based filtering and/or collaborative filtering (Madadipouya and Chelliah, 2017). Content-based filtering is focused on recommending products similar to previously purchased products. Based on what customers have already bought, similar items can be found based on e.g. the description of products. These similar items can then be recommended to the customers. Collaborative filtering looks at similarities in the taste of customers (Balabanović and Shoham, 1997). Based on what customers have already bought, preferences are determined and with these preferences, similarities in customers can be found. In the end, products that a customer has not bought yet but similar customers have, are recommended.

Recommendation techniques can be used in trying to predict the next item(s) a customer will buy. These predictions can be a next item, mostly called next item recommendation, or a full basket, mostly called next basket recommendation. Two common methods for next item recommendation are matrix factorization and Markov chains. The model that won the Netflix Prize was based on matrix factorization (Koren, 2009). This competition was about predicting ratings a viewer would give on movies based on ratings he already gave to other movies. Since matrix factorization was best in doing this, it became an increasingly important method in the field of recommendation systems.

Matrix factorization combines user and item latent factors to build a recommendation that is based on similar preferences of customers (Koren et al., 2009). This is done by decomposing a user-item matrix that contains ratings or item purchases. The decomposition results in two matrices, one containing item embeddings and the other containing user embeddings. Both these matrices can then be used to find similar customers and recommend based on these similarities. Markov chains are focused on sequential effects. This method estimates a transition matrix with probabilities of buying each product conditioned on the products bought in previous orders (Kemeny and Snell, 1976).

The following example illustrates both methods. Suppose there are two customers that are similar in their purchases. Customer 1 bought products p_a, p_b and p_c and customer 2 bought products p_b, p_c and p_d. Since these two customers both bought products p_b and p_c, they could be considered as having similar preferences. Therefore, matrix factorization could recommend item p_d (p_a) to customer 1 (2). Since a customer with similar preferences bought this product, customer 1 (2) will probably prefer this product too. Markov chains work differently. Suppose 10 customers bought products p_a, p_b and p_c and 5 customers bought products p_a, p_b and p_d, both

in alphabetical order. Markov chains will recommend product p_c to a customers that already bought products p_a and p_b since this sequence occurs more. One can conclude that matrix factorization does not look at sequences and Markov chains do not look at preferences.

By combining the preferences of matrix factorization with the sequential effects of Markov chains, new models are created that are used in next basket recommendation. These recommendations focus on predicting preferences for the next order of a customer based on the previous one(s). Rendle et al. (2010) propose a Factorized Personalized Markov Chain (FPMC) where the general interest together with the previous basket creates the recommendation. This combined model outperforms the two single models. FPMC uses linear effects of different factors but does not include interactions between these factors. Wang et al. (2015) state that this is not realistic and therefore came up with the Hierarchical Representation Model (HRM). They tackle this problem by letting the factors interact, resulting in a hybrid representation. Both the FPMC and HRM model only take local sequential behavior of a customer into account. Yu et al. (2016) state that using the global sequential behavior will improve recommendations compared to only the previous order. They suggest a Dynamic REcurrent bAsket Model (DREAM) which takes the full (global) sequence of baskets of a certain user into account. In turn, this model outperforms the FPMC and HRM model.

In the field of next item recommendation, using association rules or sequential pattern mining are intuitively the most appropriate methods. Since computers were able to store basket data, researchers started to analyse market baskets (Agrawal et al., 1993). Market basket analysis is the study of finding meaningful associations between products being bought. The tool that is mostly used for this task is called association rule learning. Let $I = \{I_1, \ldots, I_n\}$ be the set of available items and $T = \{T_1, \ldots T_m\}$ be the set of transactions. Every transaction is a binary vector with the k'th element being 1 if item I_k is in the transaction and 0 otherwise. An association rule is defined as $X \rightarrow I_j$. Here, X is a set of items from I and I_j an item from I that is not in X. Since it needs to be a meaningful association, X needs to be in at least a certain percentage of the dataset (support threshold). The association rule is satisfied with confidence factor $0 \leq c \leq 1$ if at least $c\%$ of the transactions that contain the items in X, also contain the item I_j. Agrawal et al. (1993) were the first to introduce this method and mention that it is impossible to calculate all the possible associations for a dataset with many items. Therefore, smart algorithms were designed to exclude meaningless product sets, where the most popular are the AIS (Agrawal, Imielinski and Swami), SETM (Set-Oriented Mining) and Apriori algorithms. The AIS algorithm excludes item sets in a simple and intuitive way. If an item set is below the support threshold, adding more items to this set will only decrease the support and thus will

never be meaningful. Therefore, a step-wise algorithm will stop adding items to an item set if this is already below the support threshold (Agrawal et al., 1993). The SETM algorithm works likewise but aggregates in a different way by using SQL queries (Houtsma and Swami, 1995). Finally, the Apriori algorithm uses the fact that if the support of an item set is higher than the support threshold, all subsets of the item set are also above this threshold (Agrawal et al., 1994). By using this fact, the Apriori algorithm executes faster than the previous two algorithms.

Association rules can also be used for recommendation systems (Abel et al., 2008). Lin et al. (2002) state that association rules are an aggregation of customer behaviours and therefore can be perfectly used for market basket analysis. In the field of recommendation, they state that this method should be personalized and therefore they designed an algorithm that mines association rules for each user specifically. Najafabadi et al. (2017) also use individual association rules for recommendation. Where almost all collaborative filtering methods try to find similarities in customers based on the products purchased, they compare the individual association rules.

Sequential pattern mining is similar to association rule mining, the only difference is that it incorporates the sequence of products being bought. Algorithms for sequential pattern mining are based on the algorithms of association rule mining (Mooney and Roddick, 2013). Yap et al. (2012) state that most sequential pattern models are not personalized so they build a personalized sequential pattern mining-based recommendation framework. Huang and Huang (2009) propose to incorporate time in the sequential models. Baskets that are closer to each other in a certain time frame, are considered to be stronger related than baskets that have a lot of time in between them. With the rising success of neural networks, Hidasi et al. (2015) propose using recurrent neural networks for predicting the next item. Models like matrix factorization, Markov chains and recurrent neural networks mostly assume a rigid order in products, Wang et al. (2018) state that this is not always the case and all products should get different weights based on their importance in predicting the next item.

In this paper we focus on recommending items based on relationships with products already in a basket. The element of personal preferences that matrix factorization has, could then be interesting but should not be the main focus of the model. Matrix factorization performs well when building a recommendation system on products that a customer prefers or could prefer but has not bought previously. On the home page of a web shop one could already get to see these preferred products, but this is not the purpose of this paper. Recommending a certain type of deodorant because this is the preference of a customer for example, has nothing to do with the pasta that was just added to the basket. It makes more sense, to recommend products that are often bought together with products already in the basket. Markov chains look more promising

in this context but this method also has a downside. Products that get a high probability of being the next added product to a basket, do not necessarily have to be products that are associated with the products already in the basket. If for example fruit is bought in 90% of the baskets, this product will likely get a high probability of being the next product added to a basket, even after adding a product that has no relationship with fruit. This will again give recommendations that could have nothing to do with the products already in the basket.

Association rule learning has the same problem as Markov chains have, but this method can introduce a new variable called the lift. This variable looks at how the probability of a set of products in the data set changes, if they are compared to the probability of them occurring together at random. This would mean that if fruit is in 90% of the baskets and fruit is also in 90% of the baskets containing pasta, the lift variable would be one and you can conclude that these are independent. For pasta sauce, the probability of it occurring in general is probably lower than the probability of it occurring in a basket with pasta. This would mean that pasta sauce will get a positive lift and thus could get recommended. Focusing on the lift variable would thus create recommendations that are related to the products already in the basket.

However, association rule learning still needs some defined rules to make a recommendation. If one has a basket with products p_a, p_b and p_c, one can look at the lift variable of the combination of products p_a, p_b and p_c with all the other products. The product that is most positively associated with this combination of products could then be recommended. One can also look at the lift variable of only product p_a and all other products and do the same for products p_b and p_c. Then one can recommend the product that is most positive related to either product p_a, p_b or p_c. This is different from looking at the combination of products in a basket and only considering this combination in finding associations. A recommendation can be made based on the lift variable of all products in the basket with the remaining products, but also based on subsets of the products in the basket. Either way, rules need to be defined that make the recommendation. Will the recommendation be based on the combination of all products or also on subsets, and if subsets are also taken into account in the recommendation, do they get a different weight than the combination of all products? Also, if we define a rule that will only look at the lift variable of all products in the basket with all available products, there can be no recommendation made for baskets that are not included in the data yet. There is no product associated with this combination of products because it does not occur in the data.

To the best of our knowledge, a machine learning model that learns complex associations between products has not been researched yet. The advantage of using a model for this is that there is no need to define rules. A machine learning model will learn how to aggregate all the

possible associations present in a basket to a final prediction. This final prediction will be the extent to which every product is associated with the products in the basket. Also, a model can find hidden patterns in the data that data mining techniques cannot easily discover. Modelling these associations could therefore be a better idea. A model that can be designed in such a way that it could capture associations, is the Generative Adversarial Network. In the next subsection, the research done regarding this model will be summarized.

2.2 Generative Adverserial Networks

A model that has not been used much in market basket analysis or recommendation systems, is the Generative Adversarial Network. This new neural network class consists of two parts, the generator and the discriminator. The generator takes samples from a normal/uniform distribution and tries to generate real looking data, whereas the discriminator tries to separate this fake data from the real data. By optimizing both parts, the discriminator improves the generator and the generator improves the discriminator. After optimizing the model, the generator is that good in generating real looking data that the discriminator cannot distinguish it anymore from the true data.

The introduction of the Generative Adversarial Network (Goodfellow et al., 2014) has led to many applications. Most of these applications are focused on generating real looking images by adding convolutional layers to the generator (Radford et al., 2015). Also, high resolution images can be created out of low resolution images using GANs (Ledig et al., 2017). In image-to-image translation, which is the study of taking an image and changing it into the style of another image or changing characteristics, GANs are also found to be successful (Zhu et al., 2017; Isola et al., 2017). Finally, GANs can be used in de-raining images (Zhang et al., 2019a). This is the transformation of images to better visual quality by removing bad weather conditions like rain. Similarly, the model can be used with the same purpose in speech enhancement (Pascual et al., 2017).

Applications in the e-commerce or market basket analysis are still very minimal. Zhang et al. (2019b) use the GAN to generate short product titles based on product images and product descriptions. Also, the GAN is used in creating tilled clothing images out of images of people wearing these clothes (Zhang et al., 2018). The closest applications to the problem addressed in the current paper is researched by Kumar et al. (2018). They train a GAN on orders, existing of product embeddings, customer embeddings, price and seasonality. After optimizing, the model generates fake orders that could be real. They recommend based on these fake orders because they know which customers could buy a set of products for certain prices in a certain time of

the year. Also with the introduction of new products, the generator can be conditioned on this new products and generate which customers for which price and time of the year could buy this product.

The GAN can be constructed in a way that it will learn associations between products. If the generator generates baskets where each product is independently added to the basket, and the discriminator tries to distinguish this data from the real data, the discriminator is forced to find associations between products. This way, a model is designed where complex relationships between products can be captured. In the next section, more technical details are given on how we use the GAN.

3 Methodology

The Methodology Section starts with an introduction to the problem that is being researched. Then it continues with a more detailed description of the idea behind the model. The three subsections that follow after that, contain details concerning the model and its optimization. Lastly, the evaluation of the model is discussed in the final three subsections.

3.1 Problem description

The following subsection introduces notation and shortly describes the problem that is researched in this paper. Let $P = \{p_1, ..., p_{|P|}\}$ be all the available products and $B = \{b_1, ..., b_{|B|}\}$ all the baskets containing products. A basket b_i is a binary vector of size $|P|$ where the k'th element is 1 if product p_k is in basket b_i and 0 otherwise. The objective of this paper is to design a model that learns the product associations that are present in the baskets from B.

3.2 Idea behind the model

The idea behind the model that is researched in this paper, is explained in the following subsection. Two products are independent if the probability of them occurring together in a basket, is the multiplication of the individual probabilities of the products occurring in a basket. This means that if there are two independent products, where one occurs in 50% of the baskets and the other in 30% of the baskets, they will randomly occur together in around 15% of the baskets. However, if these two products are likely to be bought together, the probability of them occurring together in a basket will be higher than the multiplication of the individual probabilities. They do not occur together randomly anymore, which causes this percentage to be higher. The other way around, if these two products are unlikely to be bought together, this percentage will be lower than the multiplication of the individual probabilities. They occur even less than they

would randomly occur together, which implies a negative relationship. The model we create, is based on these statistics.

The model that is used learns to distinguish real from fake data. Both data sets contain baskets with products since we are researching product relationships. The real data contains baskets from a real data set, whereas the fake data is created by randomly filling empty baskets with products. This means that in the fake data all products occur independently. In the example given in the first paragraph of this subsection, the two products will thus occur in around 15% of the fake baskets. In the real data, this percentage could be higher if these two products are positively associated and lower if they are negatively associated. Therefore, by letting a model learn to differentiate the two data sets, the model will learn to find product associations. In the next subsection, we explain in detail how the fake data is being generated in such a way that the products are independent.

3.3 Generating fake data

As already explained, the model that is used for learning product associations, is the GAN. This class of neural networks consists of two sub neural networks, one that generates data and one that discriminates this data from the real data. In our application, we are not interested in generating fake data. However, we use this fake data to train the discriminator. Therefore, the fake data is created without a model in such a way that the discriminator learns to separate the real data from the fake data by finding product relationships.

The first step in generating the fake data is determining the number of products in each fake basket. The distribution of the number of products per basket in the generated data, needs to be the same as the distribution of the number of products per basket in the real data. This is necessary because of the following reason. Suppose that the number of products in the fake baskets are randomly chosen. Then it could be the case that in the real data, baskets mostly consist out of two or three products, while in the generated data, baskets can contain one up to ten products. The discriminator could then determine that all baskets that contain more than 3 products, are likely to be fake baskets. This could result in a good performance for your discriminator. However, the model uses other information than product associations in differentiating the two data sets, which is not desirable. In our case, we want the discriminator to be trained in finding product associations. Therefore, the first step is sampling the number of products in the fake baskets from the distribution of the number of products in the real baskets.

The next step is filling the baskets with products. For the same reason as described in the previous paragraph, the products are selected using the individual product probabilities. The

neural network could separate the real from the fake data based on product frequencies if a product occurs more in one of the two data sets. This is again not desirable and therefore, the number of times a product occurs in the real data needs to be approximately the same in the fake data.

Suppose that we have a real data set that consists of 12 baskets where the products p_a, p_b and p_c occur in 10, 3 and 7 baskets respectively. Also, suppose that in the first step we determined that one of the fake baskets needs to contain two product. This would mean that for the first product that gets added to the fake basket, the probability of adding product p_a would be 50%, adding p_b 35% and adding p_c 15%. This is determined by dividing the frequency of a product by the sum of the frequencies of all products. By using a weighted random selection, the first product can be selected. Suppose that the first selected product is product p_a. Then the product probabilities are adjusted and thus the probability of adding product p_b next is now 70% and adding product p_c next 30%. Again, a weighted random selection determines the next added product which finishes the fake basket. This example illustrates the procedure of selecting the products for a fake basket after selecting the number of products in this basket.

The two steps above will make sure that we get a fake data set where the products occur almost equally as in the real data. The only difference between the data sets is that the fake data is drawn from the independent product distributions, whereas in the real data the products are not independent.

3.4 Discriminator

In contrast to the fake data generating process, we do use a neural network for the discriminator part of the GAN. In this subsection, the details are explained of the feed forward neural network that is used as discriminator.

In Figure 1, we can see the structure of a feed forward neural network. Here, we can see that the model consists of layers that contain different amount of neurons. The first layer is called the input layer. In this layer, the input parameters for the model are passed. In the current application, the input vectors are the baskets from the real and fake data together. Since the size of the vectors that represent the baskets are equal to the number of distinct products in the data set, this is also the size of the input layer. A basket is represented by a binary vector and therefore, each neuron in the input layer gets a value 0 or 1 based on whether the product that belongs to that neuron, is present in that basket. The input values will be passed through the network which finally results in predicted probabilities of those baskets being real baskets.

Every neuron in the input layer is connected with every neuron of the next layer. Through

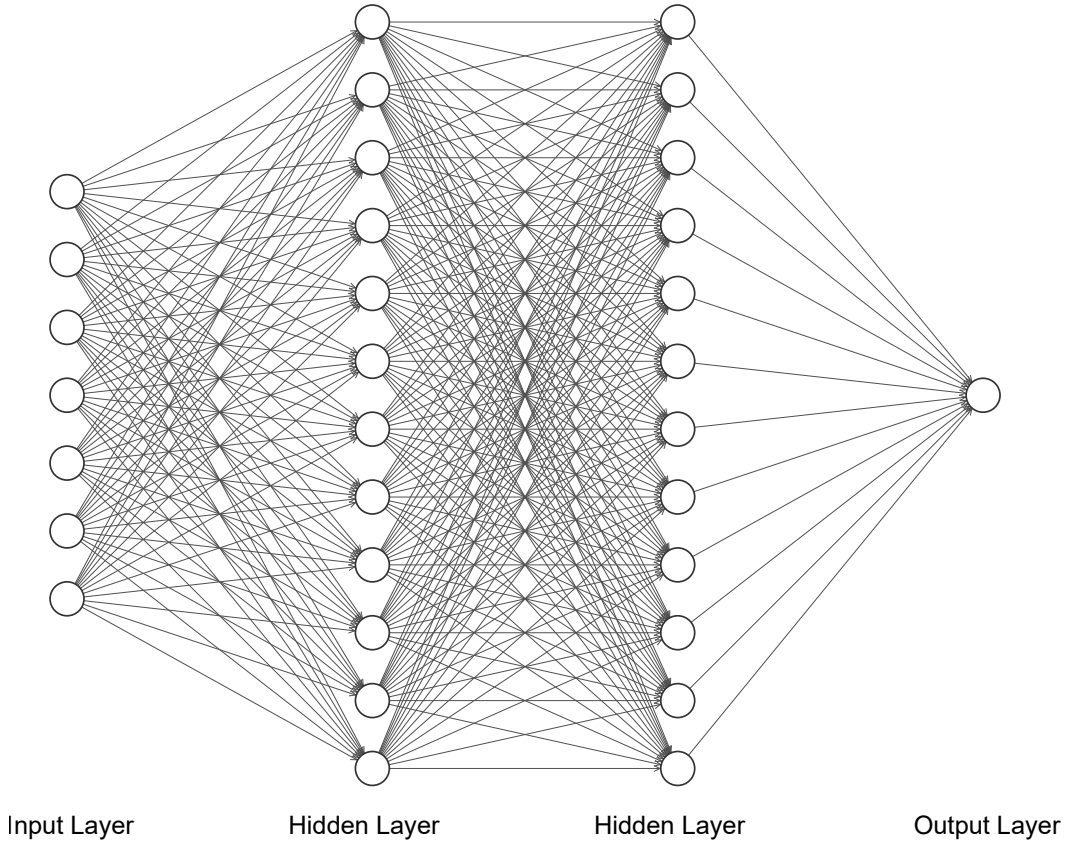


Figure 1: Structure feed forward neural network

these connections, the values from the input layer are passed onto the next layer. The process of passing the values from one layer to the subsequent layer, is the same for all layers. This can be described by

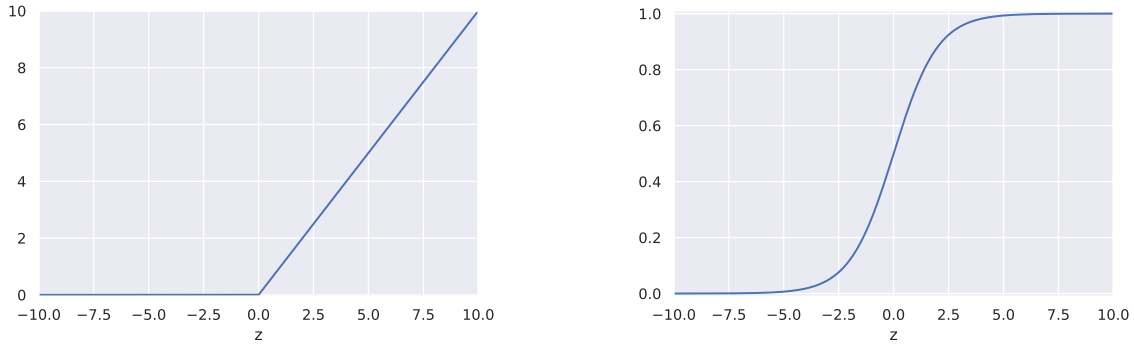
$$x_i = f(W_{(i-1,i)}x_{i-1} + bias_i), \tag{1}$$

where $f()$ is an activation function, x_i a vector containing the values of the neurons in layer i, $W_{(i-1,i)}$ the weight matrix belonging to the pass from layer $i-1$ to layer i, and $bias_i$ the bias vector belonging to layer i. First, all values from the incoming connections of layer i are multiplied by a weight. Then, the updated incoming values for each neuron in layer i, are summed and a bias is added. The final step of each pass is that the values of each neuron in layer i go through an activation function. The activation function is designed so the model can also capture non-linear relationships. This function can be one of many functions and maps the values of the neurons in layer i, to their final values. If layer i is not the final layer, these final values will be passed through the connections of layer i to layer $i+1$, where the procedure starts

all over again.

In Figure 1, we can see that the layers that follow after the input layer, are called the hidden layers. This part of the neural network can contain any number of layers just as every layer can contain any number of neurons. After passing the input values through all the hidden layers using the transformations explained in the previous paragraph, we end up at the output layer. The process of passing the values from the final hidden layer on to the output layer is the same as the rest. Each combination of a neuron in the final hidden layer with a neuron in the output layer gets multiplied by a weight, then the values are summed for each output neuron, a bias gets added and finally, the value are transformed by an activation function. The resulting values are the final output of the model. In our application, the neural network outputs one value, which is the probability of the input basket being a real basket.

The activation function that is used in the hidden layer(s), is the ReLU activation function. This function can be seen in the left graph of Figure 2.



(a) ReLU: $R(z) = max(0, z)$

(b) Sigmoid: $\sigma(z) = \frac{1}{1+e^{-z}}$

Figure 2: Activation functions

The ReLU function is simple and computationally fast, because the gradient is always either 0 or 1. Also, the ReLU function can deactivate neurons if there value is below zero. This property of deactivating neurons so the input for the next layer becomes more sparse, could be the reason for the success of this function (Glorot et al., 2011). The activation function that is used for the final output layer, is the Sigmoid function. This function maps an input value between 0 and 1 which makes it suitable for predicting probabilities.

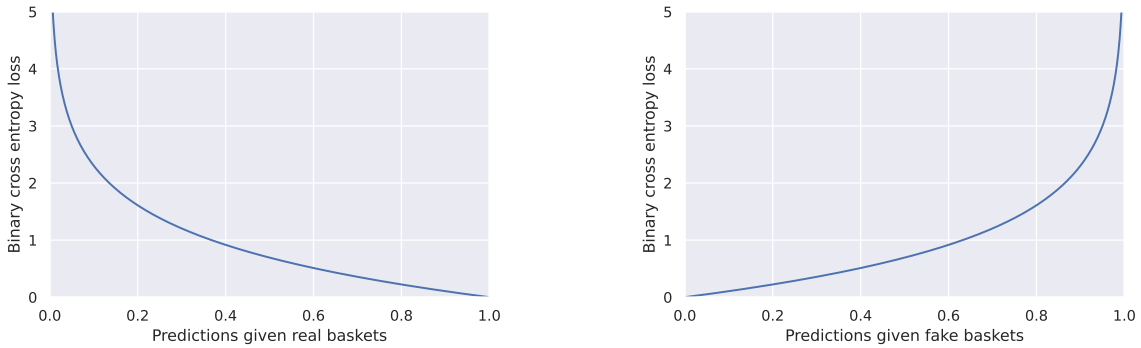
3.5 Optimization

The neural network needs a loss function based on a dependent variable for it to be trainable. The dependent variable we use is a binary vector where element i is 1 if basket i is a real basket and 0 otherwise. Since this paper deals with a probability as output on a two class problem,

the binary cross-entropy loss function is used. This loss function is defined as

$$L(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}(y_i * log(\hat{y}_i) + (1 - y_i) * log(1 - \hat{y}_i)), \tag{2}$$

where $\hat{y}$ contains the predicted probabilities on the input baskets, whereas y contains the true values of the dependent variable for the input baskets. For an observation is from the real data ($y_i = 1$), only the $log(\hat{y}_i)$ part gets added to the loss, whereas for a generated observation ($y_i = 0$), only the $log(1-\hat{y}_i)$ part gets added. For both, you want the values within the logarithm to be as close as possible to 1, because this would mean that the predicted probability on the true value is high. The logarithm is an increasing function which means that a minus needs to be added in the beginning of the formula so the loss function will decreases when the predictions become better. To better understand this loss function, Figure 3 is added.



(a) Binary cross-entropy loss given real basket

(b) Binary cross-entropy loss given fake basket

Figure 3: Binary cross-entropy loss

Here we can clearly see that the higher the predicted probability on the true outcome becomes, the lower the loss function becomes. The other way around, one can see that if the predicted probability on the true outcome decreases, the loss function increases more than linearly.

Now that a loss function is defined, every parameter has got a gradient. The gradients tell us what will happen with the loss function when we change a certain weight or bias. By adding a learning rate times this gradient to the parameter value, we hope to come closer to the minimum value of the loss function. In this paper we use the Adam optimizer (Kingma and Ba, 2014) to minimize the loss function. This optimizer is more complicated than just changing a parameter value by the learning rate times the gradient (gradient descent method). By adjusting the learning rate for each parameter specifically, Adam works more efficiently than the gradient descent method. For more technical details, we refer to the Kingma and Ba (2014) paper.

The data sets that are used to train the model are each split into a train set (70%), a

validation set (10%) and a test set (20%). The models are trained on the train sets using different amounts of hidden layers and nodes per layer. Also, three random starts are used to avoid ending up in a local minimum. The models are trained during 500 epochs using 10,000 training samples that are randomly selected. In each epoch, the model is updated in batches of 50 observations using a learning rate of 0.001. After each epoch, the loss on the validation data set is calculated which will determine the optimal model. The model with the lowest validation loss will be the final model used for that data set. Finally, the test set is used for the evaluation of the model.

3.6 Validation model

There is no easy way to test the model on learning the product associations. Therefore, the following subsection handles a method to indirectly test the model.

Most of the time in data analysis, there is a dependent variable which we try to explain based on independent variables. Testing the model can then be done by trying to predict this dependent variable with the independent variables. In our application, we can do something similar. We can randomly remove a product from a basket and try to predict this left out product based on associations with the products that are still in the basket. Even though the model is not trained on predicting a missing item from a basket, we expect that this prediction will still be substantially better than for example a random prediction. Therefore, this subsections continues with a method to predict a missing item from a basket.

The first step is deleting a product from a basket. This is simply done by taking all the baskets and randomly removing one item from each basket. Suppose that we take a basket $b_j \in B$ that contains the products $\{p_k, p_q, p_l\} \subset P$. Then we could randomly remove a product, suppose p_q, from this basket. The next step, is making a prediction for the removed products from the baskets. This is done by adding all remaining products one by one to this basket, and letting the model predict the probabilities of these basket being real basket. Thus, we would get $|P| - 2$ baskets that contain the products p_k, p_l and one of the remaining products in $P \setminus \{p_k, p_l\}$. One of these baskets contains the actual missing product p_q. The model could then predict the probabilities of these possible baskets being a real basket. Finally, these baskets can be ranked based on their predicted probabilities. The product that leads to the highest predicted probability of the basket being a real basket, is considered to be the most positive associated with the other products in the basket. Since in this paper we focus on finding product associations, the product that leads to the highest predicted probability, will be the prediction for the missing product.

3.7 Accuracy measures

In the following subsection, there are some accuracy measures defined that belong to the prediction explained in the previous subsection. The first accuracy measure that is used, is the top k accuracy. Since the predictions are probabilities, we can look at the top k probabilities and see if the correct one is in there. In our case, this would mean the k products that lead to the top k baskets that give the highest probability of being real. The number of predictions where the correct product is in the top k products, divided by the total amount of predictions, is defined as the top k accuracy.

Next to the top k accuracy, we can also look at the average rank of the actual missing product in the list of predictions. All the products that are not present in a basket are added and get a probability of being a real basket. These probabilities can be ranked where the actual missing product is somewhere in this ranking. The lower this rank is, the better the prediction is. Therefore, the average rank of the actual missing products in the predictions, is the second accuracy measure.

The two defined accuracy measures, are tested on two data sets. In the first data set, a product is removed from a basket at random. In the second data set however, a product is removed from a basket based on weights that are inversely proportional to the frequency in which the products occur in the data set. This is done because predicting the missing item in a basket from the first data set, can depend heavily on products that occur many times in the data set. By randomly removing a product, the products that occurs much more in the data set than other products, are likely to also get removed many times more from a basket than the other products. Therefore, a more balanced test set is created by removing a product based on weights that are inversely proportional to the frequency in which the products occur in the data set. An additional evaluation can be done by comparing the accuracy measures of the model on both data sets. Since associations are independent of the product frequencies, we expect that the way an item gets removed from the baskets, should not result in a difference in the accuracy measures.

The weight for product i to be removed from a basket can be calculated by

$$w_i = c/\#p_i, \tag{3}$$

where c is a constant and $\#p_i$ the frequency of product p_i in the data. By dividing a constant by the frequency of a product, we create weights that are inversely proportional to the frequency of a product. This indicates that products that occur many times in the data set, receive lower weights than products that occur less in the data. The constant can be any positive value, since

it can only change the magnitude of the weights and not the relative difference between the weights. Thus, the product that is being removed from a basket in the second data set, comes from a weighted random selection of one product from this basket.

3.8 Benchmarks

In this subsection, two benchmarks are created which can be compared to the results of the proposed model on the accuracy measures defined in the previous subsection. The first benchmark is a random prediction. More specifically, the products that are not present in a basket (where one product is already left out), are randomly ordered. The order of products that follow, is considered to be the rank of products from a random prediction. From these ranks, we can get the top k accuracy and the average rank of the true missing product.

The second benchmark is a prediction of the most bought products. This method also starts by creating a list of products but this time not in a random order. The products in the list, which are again all the products not present in the basket, are ordered by their frequency in the data set. This means products being bought a lot in general, are high in this list and products occurring not that often are low in this list. Again, from this list we can predict the top k products and we can calculate the average rank of the actual missing value.

Both benchmarks are calculated for the test data set where a product is randomly removed and for the test set where a product is removed inversely proportional to their frequency.

3.9 Product clusters

As already explained in the Introduction Section, product associations can be useful to retailers. They could for example promote items or adjust the lay out of products in a (online) store based on these associations. By trying to extract product associations from the model, we can also validate whether our model gives reasonable output. Therefore, this subsection handles a method to visualize the product associations in a low dimensional space.

The first step in visualizing the product associations, is defining an association measure which can be extracted from the model. A product association can be seen as an effect one product has on another product. The association $p_a \rightarrow p_b$ is saying that the presence of product p_a in a basket, gives a high likelihood of the basket also containing product p_b. We could look at the predicted probability of all baskets that include the products p_a and p_b. However, these baskets contain more products and thus also more associations which we are not interested in. Therefore, the cleanest way to extract the association of p_a to p_b, is by looking at the predicted probability the model gives on the basket only containing products p_a and p_b. The higher this

probability becomes, the more these two products are present in the real data compared to the fake data. In turn, this indicates a more positive association.

We can compare this predicted probability with all the other predicted probabilities of baskets containing product B and a product different from A. If the probability of the basket only containing products A and B is higher than the average of the others containing product B, this would imply that the presence of product A has a more positive effect on the basket also containing product B than the average effect of the other products. This difference can be defined as an association measure. In mathematical terms this results in

$$\widetilde{Association}(p_a \to p_b) = \tilde{y}_{\{p_a,p_b\}} - \frac{1}{|P|-2} \sum_{k \in P \setminus \{p_a,p_b\}} \tilde{y}_{\{k,p_b\}}, \tag{4}$$

where $\tilde{y}_{\{p_a,p_b\}}$ is the predicted probability of the basket only containing products p_a and p_b being a real basket. A positive number implies that product p_a results in a higher predicted probability in a basket with product p_b than the average of all other products. This would imply a positive association while a negative number implies a negative association. The above definition is suitable for illustration purposes. However, more research can be done in what the best way is to extract the associations from the model. Also, research can be done in extracting product association that contain more than 2 products from the model.

After defining an association measure, the associations need to be transformed to a distance measure for the final visualization. Firstly, the average is taken between the association of product a to b and the association of product b to a. The association $p_a \to p_b$ can be different from the association $p_b \to p_a$. The average is taken for all combinations of two products, to create a distance matrix which is symmetrical. Secondly, all numbers are extracted from the maximum number in the symmetrical associating matrix. This is done because positive associated products get a high predicted probability which we want to transform to a small distance. The distance matrix that follows, can be used in the final step to visualize the associations.

The visualization technique used is the multidimensional scaling algorithm, called 'Scaling by MAjorizing a COmplicated Function' (SMACOF) (Kruskal, 1964a,b). This method tries to create a two dimensional plot based on a given distance matrix. A loss function is used that compares the Euclidean distance in the plot with the distance matrix. By minimizing this loss function, the distances in the two dimensional plot are forced to be as close as possible to the values in the distance matrix. This way, a two dimensional plot is created which we can further analyse in the results.

4 Data

The data used in this paper to perform the analysis is the Instacart Online Grocery Shopping Dataset[1] to perform the analysis. At Instacart you can order groceries that a personal shopper will buy for you at a local retailer and deliver to you. The open source data set made available by them consists of more than 3 million orders on the Instacart website. These orders are from 200,000 users on about 50,000 products.

The data set contains 3,346,083 orders with more information than just the products the orders contain. Each order is linked to a customer which has a unique user ID. For each customer, the data set has an additional variable that indicates the time order of the orders from this customer. Additionally, the time in between two consecutive orders of a customer is also added to the data. Next to this, we know the day of the week and hour of the day of each order. Then, we can look at the products in each order. For each order, we know the time order in which the products were added to the basket. Also, we know whether this customer has already bought this product before or not. The products are indicated with an ID, but an additional data set is given with the names of the products linked to the product IDs. Each of the 49,685 products bought in the Instacart data set, can be linked to one of 134 aisles. Examples of aisles are 'prepared soups salads', 'specialty cheeses' and 'energy granola bars'. All aisles can again be linked to 21 departments like 'bakery', 'alcohol' and 'beverages'.

The current analysis does not look at a time effect or individual specific information and therefore will only look at what products were bought in each order. Due to computational restrictions, it is not possible to perform the analysis including all the products. Therefore, the first data set that is used only contains the aisles of the products that are bought in each order. We will thus have baskets containing aisles instead of products. This data set needs to be transformed to dummy variables that indicate for each aisle, whether a product from that aisle was bought in a specific order. The final aisle data set contains a dummy vectors for each order. By only considering the aisles, we reduce the size of the data set from more than 3 million orders with dummy variables on almost 50,000 products, to dummy variables on only 132 aisles.

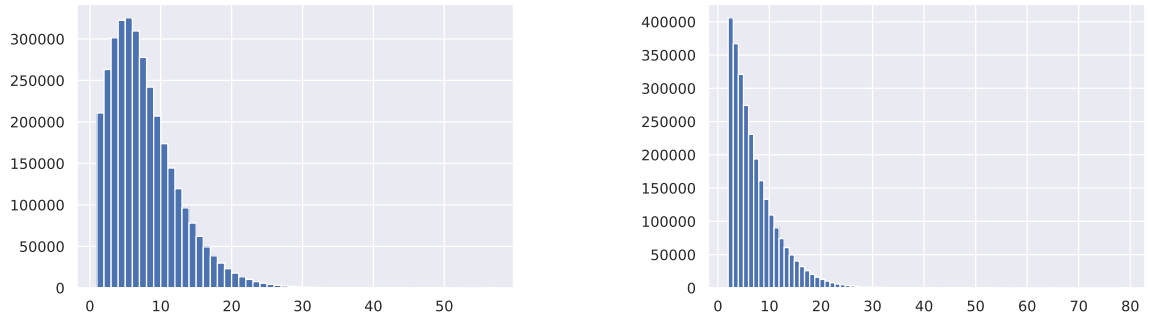
The Instacart data set names two aisles 'missing' and 'other'. Some products from these aisles are 'Organic Vanilla Soy Milk', 'Onion Crispbread' and 'Raw Pistachios', 'Unscented Foot Cream' respectively. Since these products are very specific they cannot be categorized and therefore, using these two aisles will only add noise to the model. For this reason, the 'missing' and 'other' aisle are both deleted from the aisle data set. Also, all orders that contain products

[1]The Instacart Online Grocery Shopping Dataset 2017, Accessed from https://www.instacart.com/datasets/grocery-shopping-2017 on 15 Mar. 2020.

from only one aisle, are deleted. This research is about relationships which means that at least two aisles are needed in a basket. The resulting data set contains 3,134,622 orders on 132 aisles, where each order is represented as a binary vector. A value 0 means that there were no products bought from that aisle in a specific order. Whereas, a value 1 means that there were products bought from that aisle in specific order.

The second data set that is used, only considers a subset of the 1000 most frequent products. With this data set, we can test the scalability of the model. Due to computational restrictions, a subset of 1000 products is selected instead of all products. The most frequent products in the data set are chosen, so the data set will still contain many observations. The data set is created the same way as the aisle data set. Only now, all baskets that contain less than 2 products from the 1000 most frequent products are deleted. The final product data set contains 2,663,388 orders that are represented by binary vectors that contain 1000 values.

The distribution of the amount of products/aisles in the baskets in both data sets is shown in Figure 4. The left histogram shows the distribution of the amount of aisles in the baskets of the aisle data set. In this histogram, we can see that frequency of baskets in the aisle data set rises from containing two aisles to containing five aisles in a basket. From there on the frequency decreases until at 30 aisles there are almost no baskets anymore. The right histogram, which shows the distribution of the amount of products in the baskets of the product data set, looks different than the left histogram. The top of this graph is already at two products and decreases from here on. This can be explained by the subset of data that has been chosen. Since there are almost 50,000 products in this data set, finding baskets that contain more than two products out of the 1,000 products in one basket, becomes rare. This difference in the distribution does not influence the associations that are in the data and therefore should not become a problem for the model.



(a) Distribution amount of aisles in baskets (aisle data set)

(b) Distribution amount of products in baskets (product data set)

Figure 4: Distribution amount of aisles/products in baskets

5 Simulation

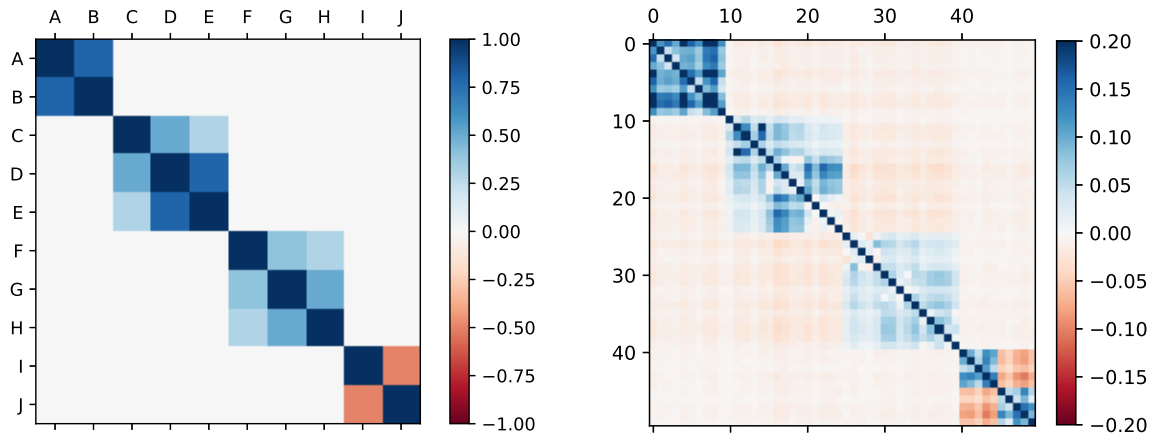
Before training the model on the Instacart data sets, the model is trained on a simulated data set. This is done so we can validate the model by training it on a data set with a known structure.

5.1 Data

In simulating the data, the simulation steps Gabel et al. (2019) take are followed. In their paper, they design a method to explore retail data by creating a two dimensional plot (P2V-MAP) where clusters of products could be found. We are not particularly interested in their method to create the plot. However, they create a simulated data set based on grocery data to validate their method. This suits the data and purpose of the simulation that we would like to use.

Gabel et al. (2019) use a sequential data simulation process which is very common in simulating grocery data. The first step is choosing the product categories in each basket prior to selecting the products. The categories that are chosen result from a multivariate probit model. The latent variable z_{ict} depends on a base utility (β_c) and an error term (ϵ_{ict}). The base utility is set to -0.4 for all categories to get baskets with a realistic amount of products. The error term is drawn from a multivariate normal distribution with a mean that is drawn from a uniform distribution between -0.5 and 0.5, so not every category is bought the same amount of time which is also not the case in real life. The correlation matrix Ω that is used in the multivariate Normal distribution, is shown in Figure 5a. From this figure we can see that we created three groups of categories that are positively correlated and two categories that are negatively correlated. This will allow us to validate the effect of both positive and negative correlations. If the latent variable, calculated by $z_{ict} = \beta_c + \epsilon_{ict}$, is higher than 0, customer i buys one or more products from category i in week t. In mathematical terms this means that $y_{ict} = 1$ if $z_{ict} > 0$ and $y_{ict} = 0$ otherwise.

After selecting the categories for each customer in the different weeks, the products can be chosen. This step follows a multinomial probit model where the utility of product j for customer i in week t is given by $u_{ijt}^{(c)} = \alpha_{ij}^{(c)} - \gamma^{(c)} p_j^{(c)} + \epsilon_{ijt}^{(c)}$. The first parameter $\alpha_{ij}^{(c)}$, is the base utility of the products in c, which is drawn from a multivariate Normal distribution. The mean of this distribution is set to 0 and the correlation matrix Σ is drawn from a Beta distribution. The parameters for the Beta distribution are $\alpha = 0.2$ and $\beta = 1$. These are chosen so most products within a category will be close to a correlation of zero, but there is also a small probability on high correlation values. The next two parameters $\gamma^{(c)}$ and $p_j^{(c)}$, determine the disutility of



(a) Correlations MVN distribution error term

(b) Correlations products in simulated data

Figure 5: Correlations

paying the price of a product. Here, $\gamma^{(c)}$ is set to 2 following Gabel et al. (2019). The prices are determined by first drawing a regular price $p^{(c)}$ for the products in a category from a Log-normal distribution with mean 0.5 and standard deviation 0.3. Then, the individual prices for the products $p^{(j)}$, are drawn from a Uniform distribution between $p^{(c)}/2$ and $p^{(c)} * 2$. Finally, an error ϵ, is added from the normal distribution with mean 0 and standard deviation 1. For each category that customer i buys products from in week t, a number is drawn from a uniform distribution between 0 and 1. If this value is lower than 0.5, only the product with the highest utility is bought from that category. If the value is higher than 0.5, the two products with the highest utilities are bought. This will results in a data set where in half of time, two products are bought from a category and in the other half only one product is bought from a category.

The simulated data set is created by using the above approach for 10.000 individuals in 100 weeks. The customers can buy from ten categories which we name category A until category J that all contain five products. In Figure 5b, we present the correlations between combinations of the products of the final simulated data set. Here we can see similar blocks as in Table 5a. Within the categories, the products are mostly positively correlated. However, there can also be some negative correlations which can be seen in category F which contains product 25 up to product 29. The correlations across products from two categories that are positively correlated, are all positive. This corresponds to the category correlations we created to simulate the data. Products across the final two categories are all negatively correlated, which again reflects the way we simulated the data by using the category correlations shown in Table 5a.

5.2 Results

The model is optimized as explained in Section 3.5. The amount of layers used could be 1 or 2 and the amount of neurons per layer is differentiated between 25, 50 and 75. The model with the lowest validation loss turned out to be the model containing two layers with 50 neurons.

The results from the optimal model together with the two benchmarks (Section 3.8) are stated in Table 1. The results that are shown are tested on the test set where one way of removing a product was with weights and the other way was randomly removing a product without weights. From the left part of the table, we can see that the model outperforms the random and most bought benchmark on the test set without weights for all accuracy measures. The model performs about four times better than a random prediction when we look at the top 3 and top 5 accuracy, while the most bought benchmark accuracy is much closer to the models' accuracy. When we compare the measures on the test set without weights with the measures on the test set with weights, a big drop can be seen for the most bought benchmark. By introducing the weights, the big dependency of the most bought products on the accuracy measures is thus reduced. For the prediction of the model and the random prediction, the accuracy measures stay almost the same. We could thus conclude that both predictions are independent of the product frequencies. For the model, this means that the learned associations are independent of the product frequencies, as we expected.

	Without weights			With weights		
	Av. rank	Acc. (top 3)	Acc. (top 5)	Av. rank	Acc. (top 3)	Acc. (top 5)
Model	8.3	25.9	40.5	8.3	26.5	40.9
Random	22.3	6.7	11.2	22.2	6.8	11.2
Most bought	10.1	22.9	33.9	15.3	12.8	20.2

Table 1: Accuracy measures simulated data

The next part of the analysis contains the MDS plot explained in Section 3.9. The result of this plot is shown in Figure 6. The products are colored after the category they are from. Starting with the first two categories A and B, we can see that both products from within the categories as between the two categories are all close to each other. The same can be seen for categories D and E, where the products from category C are a bit more distant. The distances between the products from the first five categories, confirm our expectations from the way the data is created. The products from category A and B are strongly positive related, and the products from category D and E as well. Category C is less positively correlated with category D and E than category D and E are correlated with each other. Therefore, the products from

category C are a bit more distant from category D and E. For the next three categories F, G and H, we can see that these are also positioned close to each other as expected. However, the products from category F are more distant from each other compared to products within other categories. From Figure 5b, we can see that some products within category F are not correlated, or even negatively correlated. These correlations motivate the distance between these products in the MDS plot. For the last two categories I and J, we expect by the way the data is simulated that the products between these two categories are very distant. In Figure 3.9, we can see that the products within the two categories are close to each other and the products between the two categories are distant. Since the products between the two categories are simulated with negative correlations, we could expect that the distance between these two categories would be more extreme.

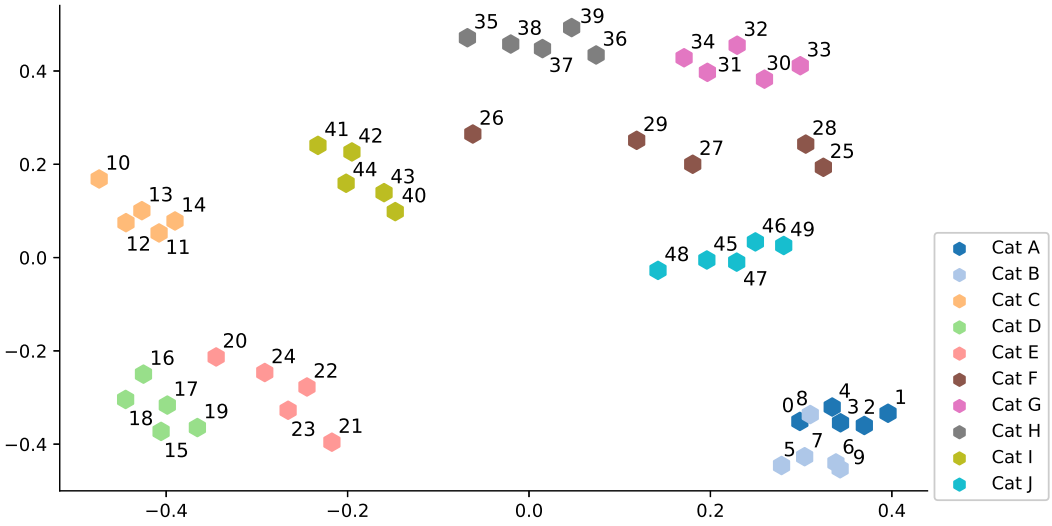


Figure 6: Structure neural network

To take a closer look at the symmetrical association matrix defined in section 3.9, Figure 7 is created. Here, we can see that even though the distance between the categories I and J is not very extreme in the MDS plot, the associations of the products between the two categories are very negative. The not so extreme distance in the plot can be explained by looking at the distances between products from category I and J with the remaining products. These distances are very similar which explains that the categories are not extremely distant. Furthermore, we can see that these distances are mostly positive while the correlations between these categories used for simulating the data, are all 0. This can be explained by the definition of the associations. The associations of a product are subtracted from a mean, implying that if there are negative

associations, there must also be positive associations. The positive associations between products from category I and J with product from the other categories weaken the positive associations of products between category C and D and product between category C and E. Another reason for the associations of the products between category C and D and products between category C and E to be close to 0, can be because of the low correlation. It could be the case that the model has a hard time distinguishing the two data sets if the difference is too small.

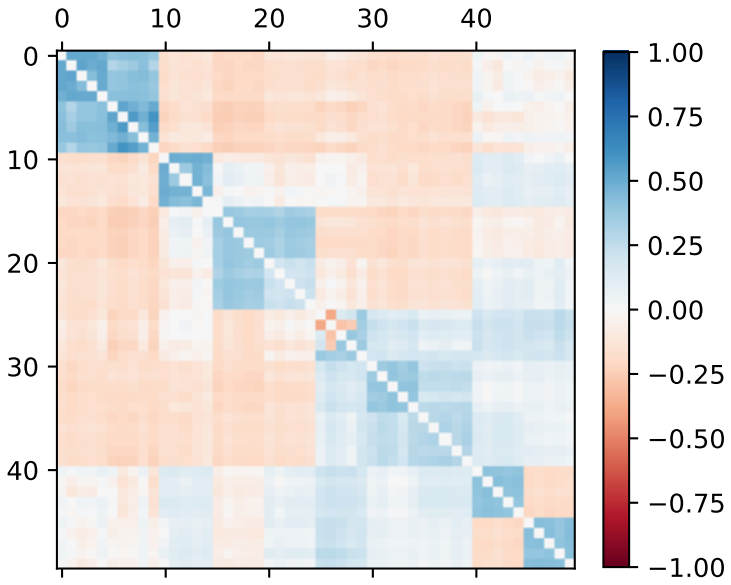


Figure 7: Structure neural network

6 Results

6.1 Results both models

The best performing model on the aisle data set is the neural network with one hidden layer and 256 nodes per layer. For the product data set, the model with two hidden layers and 256 nodes performs best on the validation data set (more details in Appendix A.1). The accuracy measures of both models with the defined benchmarks are stated in Tables 2 and 3. Beginning with the aisle data set without weights, we can see that the model performs about five times better than the random prediction. However, when we compare the model with the most bought benchmark, the model performs worse. This is different from the simulation where the model already outperformed the most bought benchmark on the data set without weights. This suggests that the associations in the real data are more complex than the simulation where we created a data set with very clear associations. When the weights are introduced to reduce the dependency of the most frequent products, we can see that the model still performs better than a random

	Without weights			With weights		
	Av. rank	Acc. (top 3)	Acc. (top 5)	Av. rank	Acc. (top 3)	Acc. (top 5)
Model	34.37	15.13%	20.14%	36.90	10.67%	15.40%
Random	62.16	2.37%	3.97%	62.15	2.44%	4.04%
Most bought	21.55	27.30%	34.29%	40.53	8.71%	12.32%

Table 2: Results model & benchmarks on aisle data set

	Without weights			With weights		
	Av. rank	Acc. (top 5)	Acc. (top 10)	Av. rank	Acc. (top 5)	Acc. (top 10)
Model	252.16	5.29%	8.32%	247.88	5.14%	8.25%
Random	496.20	0.49%	0.99%	496.97	0.50%	0.98%
Most bought	261.34	9.93%	14.32%	441.27	1.00%	1.81%

Table 3: Results model & benchmarks on product data set

prediction. However now, the model also performs better than the most bought benchmark. Another thing that can be noticed is the difference in the performance of the model on both data sets. The average rank slightly increases when introducing the weights, while the other accuracy measures both decrease with around 5%. Introducing the weights should not worsen the accuracy measures, unless many associations depend on the most bought products. This could be an explanation for the drop in the accuracy measures. However, when we compare the differences between the accuracy measures on both data sets, the accuracy measures of the model change relatively less compared to the most bought benchmark. The model thus depends on the most bought products, but in a much smaller way than the most bought benchmark depends on these.

Next, the product data set is analysed. Here we can notice that the model performs more than ten times better than the random prediction on the defined top 5 accuracy for both data sets. Also, the defined accuracy top 10 and the average rank are much better for the model than for the random prediction. The most bought benchmark performs better than the model on the data set without weights. After introducing the weights we can see that the model performs more than five times better than the most bought benchmark. Also, the model accuracy measures do not change much when comparing the performance on the two data sets. This means that the performance of the model does not depend on the most bought products, as expected.

Comparing the results of the model on the aisle data set with the model on the product data set, the following things can be seen. In both cases, the model outperforms a random prediction,

where this difference is bigger for the model on the product data set. Also, in both cases the most bought benchmark performs better than the model on the data sets without weights, suggesting that the predictions depend heavily on the most bought products. Comparing the most bought benchmark with the model on the data set with weights, we can see that the model on the aisles and the model on the products are both performing better. The difference that can be seen from Tables 2 and 3, is that the accuracy measures for the model on the products stay rather the same on both data sets, while with the model on the aisles, the accuracy measures get worse when reducing the dependency on the most bought products. This could indicate that model does not capture associations between products that occur infrequently.

6.2 Aisle mappings

Figure 8 shows the MDS plot explained in section 3.9. Every point in the plot illustrates an aisle that is colored after its department in the Instacart data set. We expect that the aisles from the same department are grouped together for most aisles. We think this because products from the same department are likely to be substitutes or complementary. If they are substitutes, they are strongly associated with each other which will group them together in the MDS plot. If they are complementary, they will be associated with the same products which will also group them together in the MDS plot.

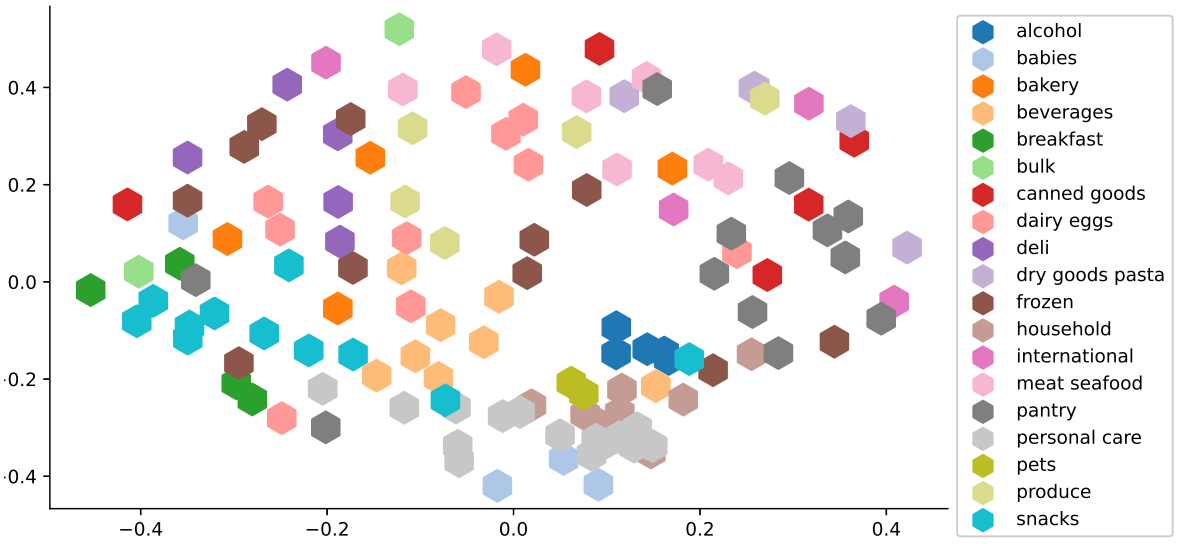


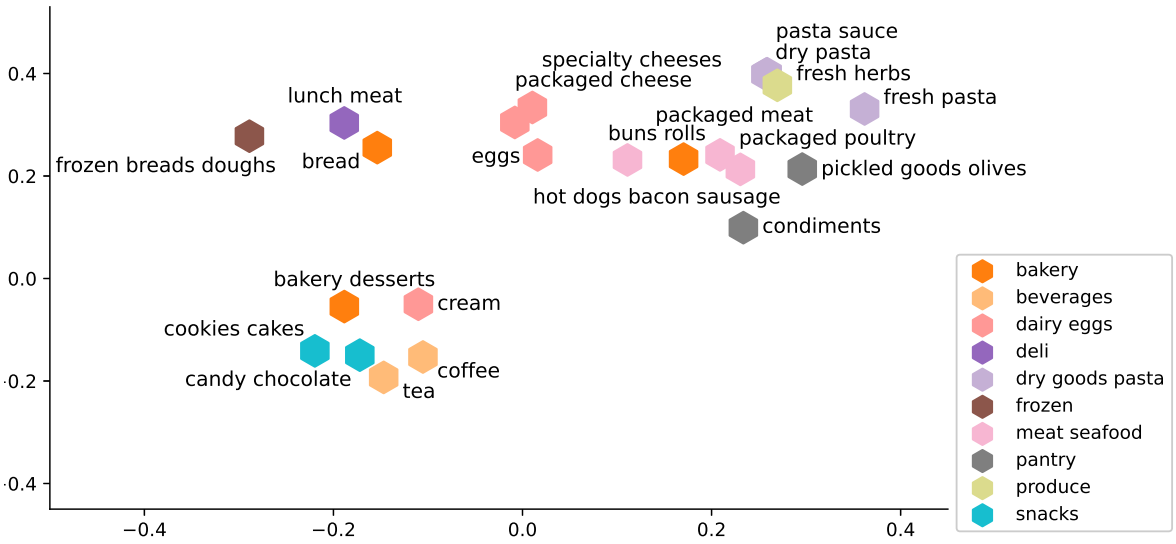
Figure 8: MDS plot on associations between aisles

In Figure 8, we can notice that there are a couple of evident clusters. The department 'alcohol' is one of these clusters. The four aisles from this department are all close to each

other in the plot, meaning that they are positive associated with each other. Similar to the 'alcohol' aisle, the 'deli', 'household', 'meat seafood', 'personal care' and 'pets' departments are also showing an evident cluster. Next, there are some departments where the aisles are divided in some smaller clusters or a big cluster with one or two outliers. The 'pantry' department for example, shows one evident cluster and two aisles that are distant from this cluster. A department which shows some smaller clusters is the 'frozen' department. Other departments that show the same pattern are the 'babies', 'beverages', 'breakfast', 'canned goods', 'dairy eggs', 'produce', 'snacks' and 'dry goods pasta' departments. These are all departments where the aisles do not show one complete cluster, but smaller clusters or an incomplete cluster. Finally, there are some departments that do not show any type of cluster. The 'bakery' department is an example of this. All aisles of this department are distant from each other and show no sign of a cluster. The 'bulk' and 'international' departments are the final two departments that also do not show a cluster. These departments are occur infrequently in the data set and could thus be badly captured by the model.

Figure 8 already confirms that most aisles within a department cluster together. However, it is also possible to find associations between aisles of different departments. To illustrate some clusters of aisles that are explainable even though they are from different departments, Figure 9 is designed. This Figure is the same as Figure 8 but only includes a couple of aisle clusters that we want to discuss. The cluster of aisles in the bottom left corner consists of aisles from the 'snacks', 'beverages', 'dairy eggs' and 'bakery' departments. Even though this cluster contains four different departments, the cluster is explainable. Coffee and tea is often combined with some kind of savoury snack, which could be from the 'cookies cakes', 'candy chocolate' and 'bakery desserts' aisles. The next cluster that is discussed contains the aisles 'buns rolls', 'packaged meat', 'packaged poultry', 'pickled goods olives', 'hot dogs bacon sausage' and 'condiments'. This cluster of aisles contains all products where customers could make hamburgers or hot dogs with. Next, there is a small cluster of three different aisles, 'frozen breads doughs', 'lunch meat' and 'bread' which can be considered as a lunch cluster. Finally, products from the 'dry goods pasta' department are clustered together with the 'fresh herbs' aisle, which is used a lot in preparing pasta.

The final argument that the model gives reasonable output, can be given by looking at the placement of the aisles from the 'dairy eggs' department. This cluster of aisles is placed in between the just defined pasta, lunch and hot dogs/hamburgers clusters. This is explainable since eggs and different kinds of cheese, can be eaten together with lunch products, pasta and hamburgers. Therefore, the placement of this cluster is plausible.



The markers of 'pasta sauce' and 'dry pasta' are overlapping and therefore look like one marker.

Figure 9: MDS plot on associations between a subset of departments

6.3 Product mappings

Figure 10 is similar to the MDS plots in the previous subsection. In this case however, we have associations between 1,000 products which is too much to create a readable plot. Therefore, the associations between the products are transformed to average associations between aisles. This is done by considering all pairs of aisles, and taking the average of the associations of all combination of two products between the two aisles of that pair. With the associations between the aisles, the MDS plot is created excluding the aisles that occur less than two times in this data set.

From the MDS plot, we can again see some clear clusters. This time the departments 'alcohol', 'babies' and 'beverages' show an evident cluster. These evident clusters are different from the aisle MDS plot since for the product data set, a subset of products was selected. Furthermore, we can see departments where there are smaller clusters, incomplete clusters or cluster with a bit more distance between the aisles. For example, the departments 'canned goods', 'dairy eggs' and 'deli' belong to these clusters. Finally, the departments 'bakery', 'breakfast', 'bulk', 'dry goods pasta' and 'international' show little sign of clusters. Overall, the MDS plot shows many clusters of aisles from the departments, indicating that the model captures the associations well.

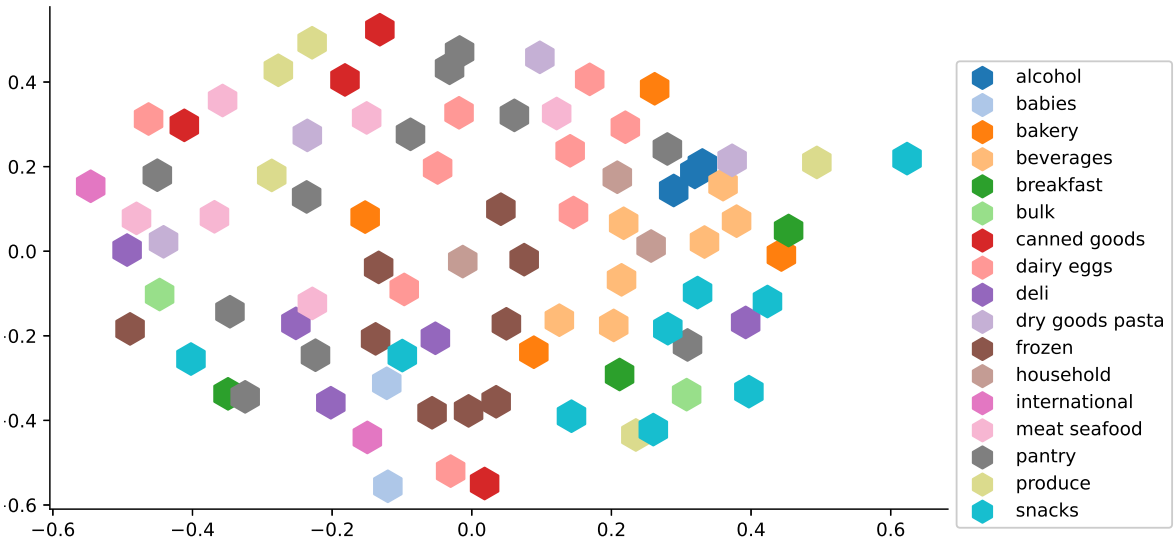


Figure 10: MDS plot on average associations of products between aisles

7 Conclusion

In this paper, a scalable model is proposed to capture complex product associations for grocery-like (online) shops. The model that is being researched, is a Generative Adversarial Network with an untrainable generator. This model learns to distinguish real from fake data, where we generate the fake ourselves. Fake baskets containing groceries are generated by step-wise randomly filling empty baskets with products. This way a data set is created where the products in the baskets occur independently. By using a feed forward neural network as discriminator to distinguish the real from the fake data, the network learns product associations. It is important that the distribution of the amount of products in the baskets and the frequencies of the products are similar in both data sets. This way, the model is forced to distinguish the two data sets by learning product associations.

To validate the model, the analysis starts with a simulation. Afterwards, two data sets are created from the Instacart data set. The 'aisle' data set consists of baskets that only contain the aisles from which the products are bought. To test the scalability of the model, a 'product' data set is created that consist of baskets that contain products from the 1000 most frequent product. The model is optimized separately for the three data sets.

Since there is no easy way to test a model that learns product associations, we propose methods to validate the model. The first method is trying to predict a missing product from a basket using product associations. Even though the model is not trained on predicting a missing

product from a basket, association are expected to still be useful in this process. The prediction of the model is compared to a random prediction and a prediction based on the most frequent products. The model outperforms a random prediction for both the simulated data sets as the two Instacart data sets. The models on the Instacart data sets do not outperform the prediction of the most frequent products when we randomly remove one product from a basket. However, when we create a way to make the prediction less dependent on the most frequent products, the model does outperform the most frequent benchmark. The accuracy measures for the 'product' data set stay similar when the method of removing a product from a basket is changed. This confirms that product associations are independent of the product frequencies. However, when reducing the dependency of the most bought aisles in the 'aisle' data set, the model performs slightly worse.

Another method used for validating the model is by visualizing the association using Muliti-Dimensional Scaling (MDS). First, an association measure between two products is created by only looking at baskets containing a pair of products. Based on these product associations, a distance measure is created to visualize the associations in a MDS plot. The MDS on the simulated data confirms the way the data was created, meaning that the model captures the associations well. Also, for the Instacart data sets most of the products are clustered based on the departments they belong to as we would expect.

Based on the validation of the model we can confirm that the model does well in capturing the associations. Also, when increasing the items in the data set by comparing the results of the 'aisle' data set with the 'product' data set, the results are similar confirming the models' scalability. The model on the 'aisle' data set, slightly dependents on the most frequent aisles. Therefore, further research needs to be done on how the model could deal with an unbalanced data set. Also, in our analysis the associations are extracted from the model in a simple and intuitive way. However, the defined associations can only contain two product while associations could contain more products. More research can therefore be done in extracting the associations from the model.

Even though the model created in this paper could be further analyzed, this paper confirms the potential of the GAN in learning product associations. By extracting the associations from the model, discoveries can be done in the field of market basket analysis. Our proposed model is a start for creating a new type of recommendation systems based on product associations.

References

Abel, F., Bittencourt, I. I., Henze, N., Krause, D., and Vassileva, J. (2008). A rule-based recommender system for online discussion forums. In *International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 12–21. Springer.

Agrawal, R., Imieliński, T., and Swami, A. (1993). Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, pages 207–216.

Agrawal, R., Srikant, R., et al. (1994). Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499.

Balabanović, M. and Shoham, Y. (1997). Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72.

Chen, Y. (2011). Interface and interaction design for group and social recommender systems. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 363–366.

Gabel, S., Guhl, D., and Klapper, D. (2019). P2v-map: mapping market structures for large retail assortments. *Journal of Marketing Research*, 56(4):557–580.

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*, pages 315–323.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Hidasi, B., Karatzoglou, A., Baltrunas, L., and Tikk, D. (2015). Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*.

Houtsma, M. and Swami, A. (1995). Set-oriented mining for association rules in relational databases. In *Proceedings of the eleventh international conference on data engineering*, pages 25–33. IEEE.

Huang, C.-L. and Huang, W.-L. (2009). Handling sequential pattern decay: Developing a two-stage collaborative recommender system. *Electronic Commerce Research and Applications*, 8(3):117–129.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134.

Kemeny, J. G. and Snell, J. L. (1976). *Markov chains*. Springer-Verlag, New York.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Koren, Y. (2009). The bellkor solution to the netflix grand prize. *Netflix prize documentation*, 81(2009):1–10.

Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37.

Kruskal, J. B. (1964a). Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27.

Kruskal, J. B. (1964b). Nonmetric multidimensional scaling: a numerical method. *Psychometrika*, 29(2):115–129.

Kumar, A., Biswas, A., and Sanyal, S. (2018). ecommercegan: A generative adversarial network for e-commerce. *arXiv preprint arXiv:1801.03244*.

Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., et al. (2017). Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690.

Lin, W., Alvarez, S. A., and Ruiz, C. (2002). Efficient adaptive-support association rule mining for recommender systems. *Data mining and knowledge discovery*, 6(1):83–105.

Madadipouya, K. and Chelliah, S. (2017). A literature review on recommender systems algorithms, techniques and evaluations. *BRAIN. Broad Research in Artificial Intelligence and Neuroscience*, 8(2):109–124.

Mooney, C. H. and Roddick, J. F. (2013). Sequential pattern mining–approaches and algorithms. *ACM Computing Surveys (CSUR)*, 45(2):1–39.

Najafabadi, M. K., Mahrin, M. N., Chuprat, S., and Sarkan, H. M. (2017). Improving the accuracy of collaborative filtering recommendations using clustering and association rules mining on implicit data. *Computers in Human Behavior*, 67:113–128.

Pascual, S., Bonafonte, A., and Serra, J. (2017). Segan: Speech enhancement generative adversarial network. *arXiv preprint arXiv:1703.09452*.

Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

Rendle, S., Freudenthaler, C., and Schmidt-Thieme, L. (2010). Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 811–820.

Wang, P., Guo, J., Lan, Y., Xu, J., Wan, S., and Cheng, X. (2015). Learning hierarchical representation model for nextbasket recommendation. In *Proceedings of the 38th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 403–412.

Wang, S., Hu, L., Cao, L., Huang, X., Lian, D., and Liu, W. (2018). Attention-based transactional context embedding for next-item recommendation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Yap, G.-E., Li, X.-L., and Philip, S. Y. (2012). Effective next-items recommendation via personalized sequential pattern mining. In *International conference on database systems for advanced applications*, pages 48–64. Springer.

Yu, F., Liu, Q., Wu, S., Wang, L., and Tan, T. (2016). A dynamic recurrent model for next basket recommendation. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 729–732.

Zhang, H., Sindagi, V., and Patel, V. M. (2019a). Image de-raining using a conditional generative adversarial network. *IEEE transactions on circuits and systems for video technology*.

Zhang, H., Sun, Y., Liu, L., Wang, X., Li, L., and Liu, W. (2018). Clothingout: a category-supervised gan model for clothing segmentation and retrieval. *Neural computing and applications*, pages 1–12.

Zhang, J.-G., Zou, P., Li, Z., Wan, Y., Pan, X., Gong, Y., and Yu, P. S. (2019b). Multi-modal generative adversarial network for short product title generation in mobile e-commerce. *arXiv preprint arXiv:1904.01735*.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232.

A Appendix

A.1 Optimization models

Table 4 shows the structures used in optimizing the model on the aisle data set. The lowest validation losses of the random starts are presented. Table 5 shows the same for the product data set.

Table 4: Lowest validation loss aisle data set

Structure		Validation loss
Layer(s)	Nodes	Aisles
1	32	0.547
2	32	0.543
3	32	0.551
1	64	0.542
2	64	0.544
3	64	0.546
1	128	0.540
2	128	0.539
3	128	0.549
1	256	0.538
2	256	0.542
3	256	0.543

Table 5: Lowest validation loss product data set

Structure		Validation loss
Layer(s)	Nodes	Products
1	256	0.399
2	256	0.390
1	512	0.397
2	512	0.398
1	1024	0.397
2	1024	0.399
1	2048	0.399
2	2048	0.405