

# EPSRC Centre for Doctoral Training in Industrially Focused Mathematical Modelling



## Matrix completion generalization to tensor completion and multi-way tables

Matteo Croci



## Contents

1. Introduction .....	2
Background .....	2
2. The mathematical model .....	3
Linear regression and tensor completion	3
Solvability and algorithms .....	4
3. Numerical results .....	4
Computational aspects .....	4
Maximum achievable complexity estimation .....	4
Parameter dependency .....	5
Nielsen data results .....	6
4. Discussion, Conclusions and Recommendations .....	7
5. Potential Impact .....	7

# 1. Introduction

## Background

Nielsen's Retailer Measurement Services (RMS) department monitors shopper behaviour for more than 6.7 billion transactions in more than 100 countries so as to measure what customers buy and estimate key business trends by product, category or market. This is achieved through weekly measurements of product sales of tens of thousands of retailers. In order to discover these trends, it is important to identify what key factors influence product sales. This is the ultimate target of this project.

Nielsen categorises products into groups of different coarseness level. Starting from the finest level we have: product code (single product), module, **module group** (or **type of product**), super-category. A realistic example of this could be a specific cola brand (single product), colas (module), carbonated soft drinks (module group) and drinks (super-category). Shops are classified as well into shop ID (single shop), region, shop type (i.e. grocery store, supermarket, ...), retailer (i.e. a supermarket chain).

The data is contained in a database of product sales by different descriptors. We divide the descriptors into two groups: (1) **Categorical descriptors**: product code, module, category, super-category, shop ID, region, shop type and retailer. (2) **Continuous descriptors**: the total number of products of that module which are sold in that shop and the number of weeks for which products of that module have been actually sold in the shop. The different categorical descriptor types (the actual retailer or product names) are called **levels**.

Key question: can we estimate the expected sales when we have missing information?

In this project, we aim to determine the expected sales for a particular choice of descriptors. Our model should be able to deliver the expected sales for the observed shop, product, retailer, etc. combinations, but also for some of the missing combinations. For example, if we are provided with the amount of sales of pasta in a particular shop in a street in London, we want to be able to estimate what the pasta sales would be if we were to sell the same product in a shop in Oxford.

## Parameters and multi-way tables

We quantify the effect that each level has on sales by associating a **parameter** to each level, the value of which has to be determined as part of the problem. The magnitude of these parameters will give us information about how much the related level is affecting sales. For example, let us consider the sales of soft drinks in a shop in the city centre. If the parameter which describes the effect of the shop position is zero, this means that the shop position is not influencing sales at all, if it is large, then position has a big impact on sales.

Different levels can interact with each other and have a joint effect on sales. For example, the sales of alcohol in a student town are higher than usual: the joint effect of the type of item sold (alcoholic beverages) and where it is sold (the student town) makes the sales amount larger than what is usually measured for that item or for that type of location. We call these effects the **interactions** between different levels and we can define an unknown parameter for each of the possible interactions as well. Finding all these parameters enables us to estimate the sales behaviour for all the possible combinations of descriptors.

The prediction of missing entries within a dataset or a table is called **matrix completion**. If the table is multi-dimensional, we call it **tensor completion**.

These parameters can naturally be gathered in various tables, one for each unique combination of descriptors. There will be a table for the parameters describing the interaction between region and retailer, a table for the interaction between shop type and region and retailer, etc. If the interaction is between two descriptors, this table will be a matrix, if it is between more than two descriptors, it will be a multi-dimensional (or multi-way) table: a **tensor**.

It is possible to keep these tables as separate or to aggregate them together into a single tensor. In the first case we have a more complex problem (more tensors and matrices), but

we also have more flexibility. In the second case we just have a single tensor, which makes the problem simpler to handle, but less flexible.

## 2. The mathematical model

### Linear regression and tensor completion

If we find the values of the parameters, then we can predict the sales behaviour and the problem is solved. However, in order to find all these parameters, we need to deal with two questions: (1) How do we compute the parameters from the sales data? (2) If we do not have sales data for a given combination, then we do not have information about the related parameter at all. How can we recover these **missing parameters**?

An item which is not sold in a shop, a supermarket chain which does not operate in a neighborhood: these plausible scenarios all give rise to missing information.

The answer to the first question comes from statistics. We can relate each parameter to the sales observation influenced by it. It is possible to express this relation in a form which often arises in statistics: a linear regression model. The solution of this model allows the extraction of the parameters from a given sales dataset. However, using this formulation alone, we are not able to recover the parameters for which we have missing information. This is often the case for real datasets and the data Nielsen works on makes no exception. We therefore need to address the second question as well.

The answer to the second question comes from a field called **matrix completion**. The typical problem in matrix completion is to complete a whole matrix for which we only know a portion of its entries by recovering the missing values. The generalisation of matrix completion to multi-way tables (or tensors) is called tensor completion. Similarly as in tensor completion, in our problem we have information about a portion of the parameters and we need to recover the rest. We show an example of tensor completion in Figure 1.




Figure 1: Matrix and tensor completion techniques can be used to clear pictures from unwanted foreground objects, such as tourists, from pictures. We can consider the objects to be removed as 'missing data' and complete the rest of the picture. Picture taken from <http://www.di.ens.fr/willow/research/inpainting/>.

We impose inner simplicity in our parameters through a tensor low-rank constraint.

Something that is important to avoid in tensor completion is the overfitting of the parameters: if we allow the parameter **complexity** to grow too much, they will become too specifically tailored to the data we have and they will lose in generality. Overfitted parameters will give very good results if used to predict our data (which we already have), but they will give terrible results when applied to the prediction of out-of-sample sales behaviour. So as to avoid this, what is done (also in matrix completion) is to constrain the complexity of the parameters by imposing some inner simplicity of a certain degree. This inner simplicity is called a **low-rank** property and makes the problem easier to solve.

In tensor completion, the more observed entries we have, the easier it is to recover the full tensor. Similarly, the more parameter information we have the easier it will be to find them all. If the dataset does not give us enough information, then it will be harder to solve the problem and, in the worst case, it will not be possible at all. The same happens with the low-rank constraint: the higher the (low-)rank, the higher the complexity and the hardest it will be to solve the problem.





This is because matrix completion is like assembling a puzzle with missing tiles: we have to make the missing tiles ourselves, but we can only guess which pattern we should draw on them. We can deduce the pattern from the other tiles we have, the less tiles we have or the more complex the overall picture of the puzzle is, the harder it will be to draw the correct pattern on the missing tiles. The complexity of the puzzle picture is similar in this sense to the high rank property in matrix completion.

To formulate our model, we combine ideas from two fields: statistics and matrix/tensor completion. We relate the sales data to the parameters in a linear regression-like way and we impose a low-rank constraint on the parameters as it is done in tensor completion.

## Solvability and algorithms

Our problem can be easier or harder to solve according to the required complexity and to how much information we have, to the extent that in some cases it might just be impossible to find a solution at all. It is therefore essential to know *a priori* what level of complexity we can expect to achieve given the information present in our dataset.

The maximum theoretical recoverable complexity can be computed. If we attempt to find a solution of complexity higher than this optimal value, any algorithm will fail. We can study the efficacy of a numerical method by empirically estimating the maximum complexity it can recover given the amount of available information. The closer this threshold is to the optimal level, the higher the estimation accuracy of the parameter is and the higher the complexity will be which the algorithm can recover.

A convex formulation has always a unique solution. In the non-convex case, different solutions might be found, of which only one is truly optimal.

Nielsen cannot afford to have very different sub-optimal predictions for a single dataset, hence they prefer a convex approach.

We can divide the methods for the solution of our problem into two approaches: the **non-convex** and **convex** approaches. The non-convex approach uses low-rank constraints to impose inner simplicity in the parameters as previously described. It has maximum recoverable complexity close to the optimal value, but it has many possible solutions amongst which only one is the best, which might not be found. The convex approach simplifies the problem by replacing the low-rank constraint with an easier-to-handle approximation. The convex problem is easier to solve and always enables the best solution to be found. However, the maximum recoverable complexity is significantly worse than in the non-convex case (up to 5 times lower in the multi-way case).

In this project, we focus on the **convex approach**. The algorithm chosen is an extension and generalisation of the **Alternating Direction method of Multipliers (ADM)** for tensor completion. If enough information is given by the data and if the output complexity is low enough, it has been proven that ADM converges to a solution of the problem.

## 3. Numerical results

### Computational aspects

The ADM algorithm separates the main problem into two sub-routines, which are solved alternatively. The first subroutine is a linear regression and it requires the solution of a linear system of equations at every algorithm iteration. The second subroutine is a matrix completion algorithm-like step that requires computing many singular value decompositions, which is extremely computationally expensive. Both of these steps take most of the computational time of the algorithm and they are therefore the bottlenecks of the ADM method. This means that by reducing the computational time for these operations, we can significantly speed up the solver. This is something important for Nielsen as this could change the waiting time for results from hours to minutes.

### Maximum achievable complexity estimation

It is important to determine what the maximum achievable complexity (rank) is for a given amount of information provided. We can quantify the information carried by the dataset as the percentage of parameters for which we have information. Similarly, we can assign a percentage value to the complexity achievable, where 100% is the optimum value for any

algorithm. We randomly generate test datasets of various sizes with only three categorical descriptors and we keep increasing the rank of the output parameters up until the maximum is achieved and ADM stops working. We show this threshold in Figure 2.

We cannot go above the transition line or our method will fail to give a sensible solution.

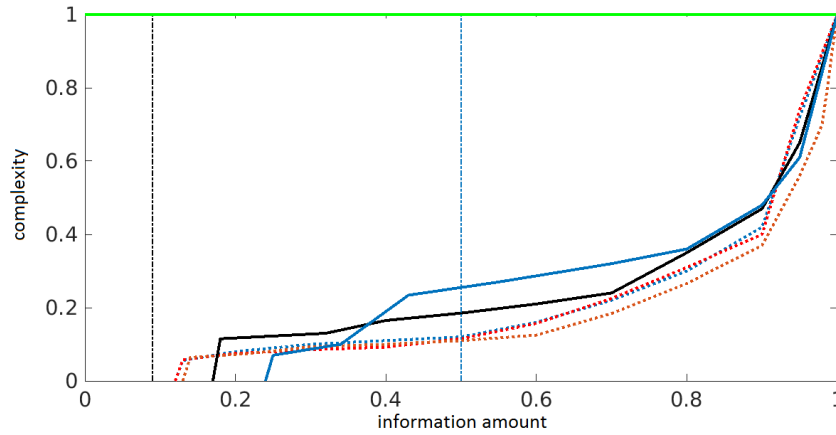


Figure 2: Graph showing the best maximum rank/complexity achievable by ADM for different problem sizes (non-vertical lines). In green, the optimal complexity achievable (100%). The vertical lines identify the amount of information given by Nielsen datasets.

For complexity and amount of information values below the coloured lines, ADM is able to recover the parameters, otherwise, it cannot. We note that the non-vertical lines, called **transition lines**, are generally around 25% of the optimum maximum achievable complexity. This is the toll we have to pay for using a convex formulation. Nevertheless, Nielsen now knows what can be achieved with the data they have available. In case a higher complexity was needed, we could then move to a non-convex formulation for which the transition line is generally around 90% of the optimum.

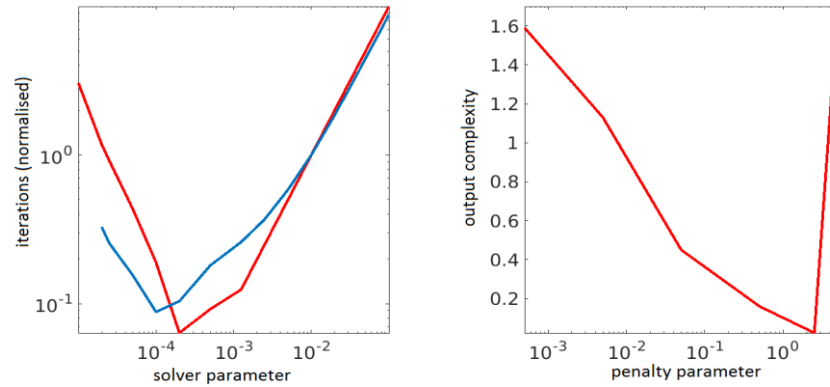


Figure 3: Graphs showing the variation of the number of iterations with solver parameters and output complexity with penalty parameter. The less iterations are needed for convergence, the faster the algorithm will be.

## Parameter dependency

The ADM solver depends on two parameters: a penalty parameter which directly affects the output complexity of the solution and a solver-specific parameter that affects the number of iterations needed for convergence. Studying how these parameters exactly affect the solution is important to make a good parameter choice and obtain better results in less time. We tested the solver dependency on parameters. Results are shown in Figure 3. By choosing the penalty parameter, we can select the desired output complexity.

## Nielsen data results

We use ADM to solve two real life problems given by Nielsen. One of these is well above the transition line, but we can reformulate the problem in a way that allows us to solve it, even though this is a convex formulation. The results we obtain fit the observations well and give good predicting power.

The dataset was divided into a 'train' and a 'test' part. The train part was fed to the ADM solver to obtain the results, the test part was used to validate the solution obtained.

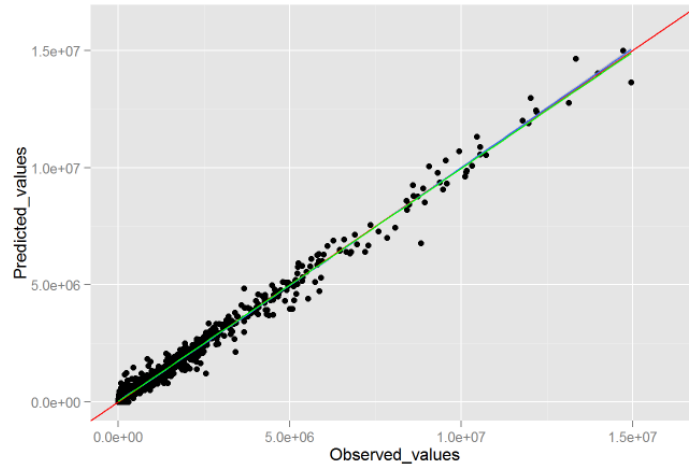


Figure 4: Graph showing predicted sales values against observed sales values. The more the fitting of the observations and the prediction are accurate, the more the dots and the green and red lines are aligned. We see how the result of our analysis gives almost perfect alignment.

It is also possible to use the output parameters to identify key business trends within the dataset. In particular, we can group the different retailers according to their business models. The main business model features are learned and extracted from the data by the ADM method. This allows us to determine which retailers have a similar business model and which act differently and what makes them different.

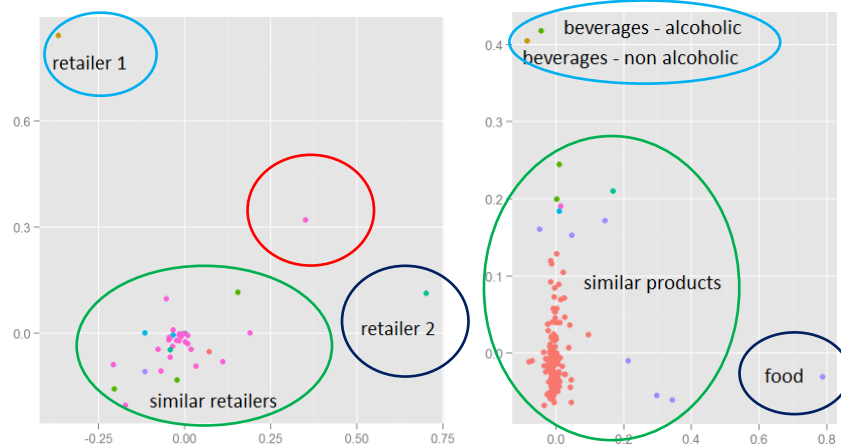



Figure 5: Scatter diagrams showing the analysis of the output parameters. Retailers (figure on the left) and products (figure on the right) are grouped according to common features extracted from the data. Each colour corresponds to a single retailer or to a single product type (module group).

For example, we see in the left plot in Figure 5 how a particular retailer, marked as retailer 1, is separated from the other retailers and placed in the top left corner of the figure. This matches, in the right-hand plot, with the separation of beverages from the other product types. This retailer actually has a particular business model selling cheap beverages in large amounts. This particular feature is picked up by our model and matches with reality. In this case we are aware of the peculiar business model of this retailer, but this shows how the



method can be used to detect unknown features as well. For instance, the fact that retailer 2 and the food product type are in the same region in the two plots means that there is a strong relation between them.

## 4. Discussion, Conclusions and Recommendations

We wrote down the problem Nielsen is interested in solving in mathematical terms. The link to matrix completion is extremely useful in determining the achievable complexity of the output solution *a priori*. The algorithm we created and used yields sensible results on Nielsen-provided datasets, which match with what is already known within Nielsen. Therefore, such an approach looks promising.

We recommend Nielsen use a non-convex formulation, which is more robust to scarcity of information and achieves better results.

The few Nielsen datasets we examined do not carry much information. This, combined with a convex formulation of the problem, might sensibly reduce the maximum achievable solution complexity, hence its quality. Although Nielsen prefers a convex formulation (since there is only one solution), we strongly recommend to complement the use of a convex ADM solver with a non-convex solver as well to deal with worst case scenarios. Non-convex formulations have the downside that they may not pick up the global optimal solution, but there are techniques that always find the best possible solution in practice. Furthermore, these formulations are extremely powerful, even when the data does not carry a large amount of information.

We also recommend that Nielsen always checks where the transition lines are for a given amount of information in the dataset. This can give extremely useful insights into what complexity can be achieved and into which formulation should be used

## 5. Potential Impact

Ludo Daemen, Vice President Data Science Research & Development at Nielsen, commented:

*“In market research one often deals with granular categorical variables (store types, store banners, geographical regions, product categories, etc.). Using such variables in predictive models is hard, especially when it comes to evaluate the impact of their interactions. This is due to the fact that there are only few (and sometimes none) observations on all specific combinations of these heavily fragmented predictors. Our thoughts went already for quite a while in the direction of leveraging the analytical frameworks developed for matrix completion to the area of regression models with categorical interaction effects.*

*Matteo’s work has been mostly on the theory and on the implementation. This work has confirmed the overall meaningfulness and feasibility of the approach. Where our own thoughts only considered interaction matrices, Matteo pushed further and he successfully generalised the approach so that it can handle as well interaction tensors (describing how three or more dimensions interact with one another). For us, these are all significant outcomes and give us a platform for further in depth tests and refinement.”*