South African Statistical Journal
Vol. 55, No. 2, pp. 125–144
https://doi.org/10.37920/sasj.2021.55.2.3

ADVANTAGES OF USING FACTORISATION MACHINES AS A STATISTICAL MODELLING TECHNIQUE

Erika Slabber[1]
Centre for BMI, North-West University, Potchefstroom, South Africa
e-mail: *erika.slabber@nwu.ac.za*

Tanja Verster
Centre for BMI, North-West University, Potchefstroom, South Africa
e-mail: *tanja.verster@nwu.ac.za*

Riaan de Jongh
Centre for BMI, North-West University, Potchefstroom, South Africa
e-mail: *riaan.dejongh@nwu.ac.za*

Factorisation machines originated from the field of machine learning literature and have gained popularity because of the high accuracy obtained in several prediction problems, in particular in the area of recommender systems. This article will provide a motivation for the use of factorisation machines, discuss the fundamentals of factorisation machines, and provide examples of some applications and the possible gains by using factorisation machines as part of the statistician's model-building toolkit. Data sets and existing software packages will be used to illustrate how factorisation machines may be fitted and in what context it is worth being used.

Keywords: Factorisation machines, Linear regression, Machine learning, Recommender systems.

1. Introduction

Factorisation machines (FMs) are extensions of linear models and can model all the higher-order interactions between predictor variables in a very efficient way. It can be applied to a variety of prediction and classification tasks and present an alternative method for well-known statistical modelling techniques such as regression, logistic regression and regularised versions thereof. FMs were introduced by Rendle (2010) and most of the subsequent research appeared in the machine learning literature. A review of the literature reveals some benefits and drawbacks of FMs. Benefits include simpler models (less parameters are used), fast computation (caused by approximating interaction terms in a linear fashion), automatic feature engineering (since all variables and interactions are considered) and the estimation of parameters in a very sparse and high-dimensional data setting. A drawback is that it may underfit dense data since the parameter space is constrained. The interested reader is referred to Rendle (2010) and McClory (2018) for more information on the benefits and drawbacks. In the machine learning literature, FMs have gained popularity because of spectacular

[1] Corresponding author.
MSC2020 subject classifications. 62H99.

successes obtained in the area of recommender systems. Recommender systems usually deal with the problem of predicting a categorical or interval response variable with categorical predictor variables. Often these variables are nominal, with many levels that lead to a high-dimensional sparse data setting, an area in which FMs thrive.

In this paper we will investigate the above claims and, compared to the above-mentioned statistical techniques, illustrate the possible gain of using FMs. We will do this by the use of examples in selected application areas. In order to fit FMs, optimisation algorithms and software implementations thereof are required. The practical implementation of these software routines is not straightforward and can be complex when categorical variables are present. In the various examples studied in this paper these difficulties will be highlighted and solutions provided. Lastly, since the statistical literature is relatively silent on FMs as a modelling technique, we hope that this paper will be able to convince practising statisticians to include this technique in their modelling toolkit.

The layout of this paper is as follows. In the next section, a short background covering the origination of FMs and the successes obtained in the area of recommender systems are provided by means of a literature review. In Section 3, FMs are defined and explained using familiar regression notation. Some guidelines for the practical implementation of these models are included in this section. Section 4 contains a discussion of the algorithms and software routines for fitting these models to data. In Section 5, the practical application of the software is illustrated empirically. A well-known recommender system data set, as well as simulated data sets from regression and logistic regression applications will be used to illustrate the advantage of using FMs in a large data set context. Since the application of these algorithms is not straightforward, software code will be provided, and the performance of the algorithms compared. Some concluding remarks and suggestions for future research are given in the last section.

2. Background on factorisation machines

When Rendle (2010) introduced the machine learning community to FMs, he motivated the method as one that combines the advantages of support-vector machines with those of factorisation models. Support-vector machines and factorisation models, such as matrix factorisation (Srebro and Jaakkola, 2003), tensor factorisation (Tucker decomposition; Tucker, 1966), parallel factor analysis (Harshman, 1970) and pairwise tensor factorisation (Rendle and Schmidt-Thieme, 2010) are frequently used for prediction tasks in machine learning and intelligent information systems. One well-known example of matrix factorisation is singular value decomposition (SVD) where SVD is interpreted as a method to compute the low-rank approximation of a matrix by minimising the squared error loss. As will become clear later, matrix factorisation is a key concept in FMs, which have now surpassed the successes of factorisation models as recommender systems. Similar to support-vector machines (Steinwart and Christmann, 2008), FMs are general predictors working with any real-valued feature vector, but in contrast to support-vector machines, they model all interactions between variables using factorised parameters. Furthermore, FMs are able to estimate interactions even in problems with huge sparsity (such as recommender systems) where support-vector machines fail, as stated by Rendle (2010). Therefore, Rendle (2010) claimed that FMs combine the power of factorisation models with that of feature engineering, which is desirable in a model-building context and especially in recommender systems where the data are typically sparse and the number of predictor variables

frequently more than the number of observations.

In a very general way, recommender systems are concerned with suggesting relevant products to users. For example, Netflix uses recommender systems to advise their clients on the next best movie to watch (based on their viewing preferences); Amazon uses these systems to recommend to clients which products to consider buying next (based on their buying behaviour); and other technology companies such as Google and YouTube use these systems to advise their clients in similar ways. Of course, the end goal for all companies is to become more profitable by selling more products and by growing their client base. Using recommender systems that can accurately advise clients and convince them to buy more products can therefore be a highly effective marketing tool for companies. Today, because of the wide availability of customer information, recommender systems have become one of the most rapidly growing branches of artificial intelligence. The growth of available customer information is attributable to the wide accessibility of recorded data on customer behaviour as well as an exponential growth in the number of characteristics or variables that can aid in increasing the prediction accuracy of recommender systems. Often the number of variables is so large that they overshadow the number of data points recorded. Building recommender systems for problems where the number of variables outnumbers the data points (also referred to as sparsity) gave rise to the introduction of FMs by Rendle (2010). Because factorisation machines can potentially model the high-dimensional interactions between the predictor variables and incorporate these into the prediction model, it is claimed that FMs cater for feature engineering as well as modelling.

FMs achieved numerous rewards in a number of data science challenges (see Parsons, 2017), for example, in the KDD Cup, an annual data mining and knowledge discovery competition organised by ACM Special Interest Group, FMs achieved first place in 2010 in the Grockit Challenge for predicting the next exam question, first place in 2011 for rating prediction on the MovieLens data set, and third place in 2012 for click prediction. FMs also achieved first place in the Kaggle 'Blue Book for Bulldozers' challenge for predicting the auction sale price. For more information on recommender systems, see Bhatnagar (2016).

3. Factorisation machines in a regression context

One of the earliest and best known model-building tools in statistics is regression. The technique has been used both in a descriptive and prediction context with a well-established statistical theory of inference. Interestingly, in some machine learning literature, regression is often considered as part of the 'machine learning' model-building toolkit, but mostly in the context of minimising the mean squared error (not maximum likelihood) (Rendle, 2010). Machine learning is all about the model and how to fit the model to data in a prediction context. So the loss function (usually mean squared error), and the optimisation method to minimise the loss function are of cardinal importance. In most machine learning models (FMs, neural networks, etc.) inference (e.g. the significance tests about model parameters) is largely absent. Although regression, specifically linear regression, has served the statistical community well, it has difficulty dealing with modelling all higher-order interactions between predictor variables. In fact, when the number of prediction variables is large, the current methods even struggle with modelling only the two-factor interactions, see Yurochkin et al. (2017), and are unable to fit a model where the number of predictor variables is more than the number of observations. Modelling higher-order interactions results in a combinatorial explosion that renders

the fitting of regression models inefficient, especially in a sparsity context. In the remainder of this section, it is shown how FMs become a very useful alternative.

As in the regression case, FMs are concerned with predicting a response Y from some input data X assuming a relationship of the form $Y = f(X)$ with a parametrically specified f for which the parameter values are to be estimated. The prediction is $\hat{Y} = \hat{f}(X)$. As stated previously, regression and support-vector machines encounter difficulties when the input data X are very sparse, a condition that is often encountered when the input consists mainly of large-sized categorical variables, as in recommender systems. Refer to Hastie et al. (2015) for more information. Rendle (2010) introduced FMs, which posit a very specific relationship between the explanatory data X and the response variable of interest, Y. The claim is that the FM generalises the usual models and can also handle very sparse data more successfully. Below a rather simplistic explanation of the rationale underlying the FM model is given. The linear regression prediction model is

$$Y = \beta_0 + \sum_{k=1}^{K} \beta_k X_k,$$

where Y is the response variable, β_0 the intercept, β_k the coefficient for the kth predictor variable, and X_k the kth predictor variable (where $k = 1, \ldots, K$). In total, $K+1$ parameters have to be estimated in the linear regression model. Typically, interactions between feature variables (X_k) have to be taken into account in order to get predictions of acceptable quality. If the linear model structure is to be preserved, the extension of the linear regression model is as follows:

$$Y = \beta_0 + \sum_{k=1}^{K} \beta_k X_k + \sum_{k=1}^{K-1} \sum_{j=k+1}^{K} \beta_{kj} X_k X_j, \tag{1}$$

which includes all possible two-way interactions. The model (1) will be referred to as the regression model with two-way interactions. Note that an additional $K(K-1)/2$ parameters β_{kj} have to be estimated. Thus the total number of parameters (P) is given by $P = K + 1 + K(K-1)/2$. Clearly this could lead to a "small N (with N the number of observations), large P" situation where ordinary least squares regression becomes unworkable when the X-matrix is singular and its inverse does not exist. This happens when the number of parameters to be estimated, P, is more than the number of observations. One approach to avoid this is to use a nonlinear model:

$$\begin{aligned} Y &= \beta_0 + \sum_{k=1}^{K} \beta_k X_k + \left(\sum_{k=1}^{K} \beta_k X_k \right)^2 \\ &= \beta_0 + \sum_{k=1}^{K} \beta_k X_k + \sum_{k=1}^{K} \beta_k{}^2 X_k{}^2 + 2 \sum_{k=1}^{K-1} \sum_{j=k+1}^{K} \beta_k \beta_j X_k X_j. \end{aligned}$$

By removing the "purely quadratic" terms and the factor 2, this leads to

$$Y = \beta_0 + \sum_{k=1}^{K} \beta_k X_k + \sum_{k=1}^{K-1} \sum_{j=k+1}^{K} \beta_k \beta_j X_k X_j.$$

This model contains no extra parameters — the β_{kj} have been factorised into simple products $\beta_k \beta_j$ of the original linear model parameters. This would generally be too restrictive a model. The

interaction β_{kj} can be better approximated by a sum of simple products consisting of G factors:

$$\beta_{kj} = \sum_{g=1}^{G} v_{kg} v_{jg}. \tag{2}$$

Accordingly, Rendle (2010) puts forward the following extension:

$$Y = \beta_0 + \sum_{k=1}^{K} \beta_k X_k + \sum_{k=1}^{K-1} \sum_{j=k+1}^{K} \sum_{g=1}^{G} v_{kg} v_{jg} X_k X_j, \tag{3}$$

with $(K \times G)$ "interaction parameter matrix" $\{v_{kg}\}$. Typically, G is chosen to be small, say 2 or 3, in which case the number of additional interaction parameters, $(K \times G)$, may be substantially less than $K(K-1)/2$, the number required in a regression model including all two-way interactions. As described by Freudenthaler et al. (2009), the formulation (3) replaces the two-way interaction effects by their factorised analogues as given by (2). Note that it is well known that for any $(K \times K)$ positive definite matrix $\beta = \{\beta_{kj}\}$ there exists a $(K \times G)$ matrix $V = \{v_{kg}\}$ such that $\beta = VV^T$, provided that G is sufficiently large. Refer to Srebro et al. (2005). This shows that an FM can express any interaction matrix $\{\beta_{kj}\}$ if G is large enough. In general, (2) implies that the data for one interaction assist in the estimation of parameters for related interactions, making the model-fitting process more time efficient. In high-dimensional sparse-data settings, like the classic movie recommender system example (see Section 4.1), it is challenging, if not impossible, to estimate interactions between variables independently. FMs break this independence by the factorisation of higher-order interactions which enable efficient coefficient estimation (Rendle, 2010). FMs can handle sparse data quite successfully due to the fact that interactions appear as products $v_{kg}v_{jg}$, hence allowing the estimation of each component to "borrow strength" from other components. Although we will not consider higher order interactions in this paper, it is sufficient to say that FMs can model d-way variable interactions, where d is the polynomial order. Note that according to Harris (2015), in all the current implementations the order has been fixed at two. This is still the case for the implementations used in this paper (LibFM and PROC FACTMAC - see Section 4). The formulation in (3) easily generalises to a model for d-way interactions (see Blondel et al. (2016) for more information).

Returning to the FM formulation in (3) it should be clear that the choice of the number of factors is of practical concern. Rendle (2010) recommended that a small G (2 or 3) should be used when working with sparse data since there are usually not enough data to estimate complex interactions. If the number of factors is restricted, it leads to better prediction and improved interaction matrices under sparsity. As mentioned before, when K variables are used it will result in $(K)(K-1)/2$ two-way interaction terms. Should a value of G be chosen that is too high, the FM model will not provide a gain in terms of modelling since we will end up with the same number of parameters as when a traditional linear regression model with interactions is fitted. G should thus be chosen such that $K(K-1)/2 > KG$. For example in (3), when K is set to 500, it will result in 124 750 interactions terms that have to be estimated, while a choice of $G = 10$ will result in estimating only 5 000 interaction parameters (instead of 124 750). At this stage, no mechanism has been developed to assist with the choice of G, so it remains rather arbitrary and practitioners will have to evaluate this based on their particular applications. However, Rendle (2010) and others recommend the use

Table 1. Number of interaction parameters to estimate at given choices of K and G.

		KG		
K	$K(K-1)/2$	$G=2$	$G=3$	$G=4$
3	3	6	9	12
4	6	8	12	16
5	10	10	15	20
6	15	12	18	24
7	21	14	21	28
8	28	16	24	32
9	36	18	27	36
10	45	20	30	40
11	55	22	33	44
12	66	24	36	48
13	78	26	39	52
14	91	28	42	56
15	105	30	45	60

of a smaller G since it leads to better generalisation, especially under sparsity conditions. Table 1 illustrates the choices available to the modeller when $K = 3, 4, \ldots, 15$ and $G = 2, 3, 4$. Clearly, when $K = 3, 4, 5$ it makes no sense to use FM (4) instead of the regression model (2) since there is no gain in factorising the interaction terms. However, when $K = 15$ and $G = 2$, the benefit of using the FM is clear since far fewer parameters need to be estimated (30 under FM (4) instead of 105 under regression model (2)). The shaded area in the table indicates where the FM comes into its own.

As stated before, high-dimensional problems and sparsity are the feeding areas for FMs that excel in this environment as they not only deal with a large number of variables but also with their higher-order interactions. Because of this, Rendle claimed that FMs have the ability to combine automatic feature engineering with the benefits of factorisation models. A key aspect of the modelling process is to do proper encoding of the categorical variables (e.g. binary, nominal and ordinal) and to formulate the prediction problem in such a way that it can logically be translated into a predictor variable matrix (referred to as the X-matrix in this paper) that is encoded in an appropriate way. Many papers about proper encoding exist in the machine learning literature, as do encoding techniques such as one-hot coding, ordinal coding, label coding, hashing, leave-one-out coding, sum coding, Helmert coding, polynomial coding, backward difference coding and binary coding (see, for example, Potdar et al., 2017; Hancock and Khoshgoftaar, 2020). Often the encoding method requires a proliferation in the number of predictor variables. For example, one nominal variable with 10 levels using dummy variables will result in an extra 8 predictor variables in the X-matrix.

4. Fitting factorisation machines

Assume we have a training data set $(Y_n, X_{n,1}, \ldots, X_{n,K})$, $n = 1, \ldots, N$. The parameters in model (3) are estimated by minimising some distance measure between Y_n and the predicted value $\hat{Y}_n$ generated

from model (3). Let $\ell\left(Y, \hat{Y}\right)$ denote a typical loss function or distance measure. We then have to minimise

$$\frac{1}{N}\sum_{n=1}^{N}\ell\left(Y_n, \hat{Y}_n\right).$$

For continuously distributed responses, squared error loss is typically used, where

$$\ell_s\left(Y, \hat{Y}\right) = \left(Y - \hat{Y}\right)^2.$$

Using this loss function in the above expression gives the well-known mean squared error (MSE) distance measure. Note that the root mean squared error (RMSE) is sometimes used instead of MSE, where $RMSE = \sqrt{MSE}$.

For binary responses $Y_n = +1$ or -1, we can either use hinge loss,

$$\ell_H\left(Y, \hat{Y}\right) = \max\{0, 1 - Y\hat{Y}\},$$

or logit loss,

$$\ell_L\left(Y, \hat{Y}\right) = \frac{\log\left(1 + \exp\left(-Y\hat{Y}\right)\right)}{\log 2}.$$

See Rosasco et al. (2004) for more detail.

Several FM software implementations currently exist, for example the software package LibFM developed by Rendle (2012), PROC FACTMAC in SAS® Visual Data Mining and Machine Learning (SAS Viya), libFMexe in R (a wrapper of the LibFM package that calls the LibFM executable) and fastFM in Python. In this paper, only LibFM and PROC FACTMAC will be used to demonstrate how FMs may be fitted to a data set. LibFM is the most flexible of the two algorithms since it can handle both regression and classification problems. More specifically, LibFM implements both MSE and logit loss and potentially caters for categorical and continuous predictor and response variables. On the other hand, PROC FACTMAC only implements RMSE and only caters for classification problems for an interval-scale response variable and nominal categorical predictor variables. This currently puts a restriction on the type of problems that PROC FACTMAC can handle. A limitation of LibFM is that the output provided is restricted to the predicted values of the response variable, while PROC FACTMAC provides the predicted response values as well as the estimated coefficients of the predicted variables and associated factors. In conclusion, while both algorithms can be used to fit FM models to typical recommender system type data sets, LibFM can also be used in a regression and logistic regression context.

As far as algorithms for the minimisation of the loss function are concerned, both LibFM and PROC FACTMAC implement stochastic gradient descent (SGD) optimisation, while only LibFM gives the user the option to use three further algorithms, namely adaptive stochastic gradient descent (SGDA), alternating least squares (ALS), and Markov Chain Monte Carlo (MCMC). The algorithms implemented in LibFM are described in detail in Rendle (2012) and the algorithm used in PROC FACTMAC in SAS Institute Inc. (2019b). Note that both LibFM and PROC FACTMAC have adapted the above-mentioned loss functions by including regularisation of the model parameters. The reason for this is that if the number of factors is large, many parameters have to be estimated, which makes FMs prone to overfitting. PROC FACTMAC implements a max-norm regularisation of the factor parameters, while LibFM implements an L2 regularisation of all model parameters. Furthermore,

LibFM provides the user with the option to group the model parameters and specify individual regularisation parameters for different parts of the model. All the above-mentioned algorithms require initial values for the so-called hyper parameters that are inputs to the optimisation algorithms. Although default values are supplied, the modeller has to be very careful of the algorithms getting stuck in local optima. This is why experimentation with these inputs is advised. PROC FACTMAC circumvents this problem by choosing the input values based on a grid search combined with a genetic algorithm.

Rendle (2012) claims that "even though factorisation models have a high prediction quality in many applications, it is non-trivial to work with them." For each problem that cannot be described with categorical variables (as in PROC FACTMAC), a new specialisation model has to be derived and a learning algorithm has to be developed and implemented. This is very time-consuming, error-prone and only applicable for experts in FM models. Although LibFM goes a long way to alleviate this problem, the previous remark by Rendle is certainly true and it took the authors of this paper considerable time to learn how to use the software packages when fitting FM models to various data sets. In his paper, Rendle recommends that the inexperienced user should rather choose the MCMC algorithm, since it determines the regularisation values automatically, while initial estimates for this have to be provided for the other LibFM algorithms. Rendle (2012) gives the following practical advice for applying LibFM to prediction problems:

- For inexperienced users, use MCMC since it is most simple to work with and only requires one input value, i.e. the standard deviation. The standard deviation refers to the standard deviation of the normal distribution that is used in the MCMC algorithm to generate the initial values of the factor parameter matrix V; see (3).
- When a predictive model for a new data set is being built, start with a low factorisation dimensionality and first determine the standard deviation for initialisation, because intelligently chosen values will speed up the MCMC sampling. Refer to Rendle (2012) for more information.
- Several values for the standard deviation should be tested (e.g. $0.1, 0.5, 0.9$). The success can be seen quickly on the first few iterations by monitoring the training error.
- After determining an appropriate initial value for the standard deviation, MCMC can be run with a large number of iterations and larger factorisation dimensionality. The output and convergence can be monitored by the output of LibFM.

Remark. It should be noted that SAS® is still in the process of developing all the different features of PROC FACTMAC. As stated above, one of the limitations of PROC FACTMAC is that only nominal predictor variables and interval response variables are allowed as input (SAS Institute Inc., 2017). A new update of PROC FACTMAC was recently released that can now handle interval predictor variables as well, but for some reason at least two nominal predictor variables have to be included (SAS Institute Inc., 2019a). So, at this stage the application domain of PROC FACTMAC is quite restricted when compared to LibFM. However, an advantage of PROC FACTMAC over LibFM is that it automatically encodes nominal variables.

5. Empirical examples

The objective of this section is to give an overview of how to use the LibFM and PROC FACTMAC packages to fit FMs to data sets in different settings including a recommender system, a regression setting and a logistic regression setting. In the first example, the well-known MovieLens data set (developed by the GroupLens project at the University of Minnesota and freely available on the internet — see Harper and Konstan, 2015) will be used to demonstrate how both packages may be used to fit an FM as a recommender system where predictor variables are typically nominal or categorical. The results from these two implementations are then compared to ordinary least squares (OLS) regression and the LASSO. Then in the next example, a simulated regression data set is used to demonstrate how LibFM may be used to fit an FM to a regression data set that typically consists of continuous predictor variables. Note that, at the moment, PROC FACTMAC cannot be applied on a data set containing continuous predictor variables. In the last example, logistic regression data was simulated and the performance of PROC LOGISTIC compared to that of LibFM. Some performance comparisons are made to compare the various algorithms.

5.1 Recommender system

The well-known and well-analysed MovieLens data set is used in this example to demonstrate a relatively simple application of a recommender system. In particular, we will show how a seemingly straightforward and simple data set can become a tricky prediction problem due to the nature of the data. The data set was collected from companies that provide movies for online viewing — see Silva and Wright (2017). It consists of ratings that users (viewers) give for items (movies) that they have viewed. Suppose there are three users and four items. The rating matrix could then look like the one in Table 2.

The objective is to build a recommender system that will be able to predict the unknown ratings (indicated by '?') so that companies would know which movie should be recommended to which user next. Although users ($U1, U2, U3$) and items ($I1, I2, I3, I4$) are indexed here, users will typically be uniquely identified by a user number or ID, and movies by a specific name. Clearly the users and items are nominal variables and the rating an interval-scale variable on a scale of 1 to 5, where 5 indicates the highest preference. Therefore the natural way to represent this data set for input into an FM is to follow so-called one-hot encoding (i.e. $0/1$ indicator variables) as indicated in Table 3.

Since the values of the predictor variables $U1$, $U2$, $U3$, $I1$, $I2$, $I3$ and $I4$ are indicator variables,

Table 2. Rating matrix.

	$I1$	$I2$	$I3$	$I4$
$U1$	2	3	1	?
$U2$	?	5	?	1
$U3$	?	?	4	1

Table 3. Transformed X-matrix using one-hot encoding.

Obs	$U1$	$U2$	$U3$	$I1$	$I2$	$I3$	$I4$	Y
1	1	0	0	1	0	0	0	2
2	1	0	0	0	1	0	0	3
3	1	0	0	0	0	1	0	1
4	0	1	0	0	1	0	0	5
5	0	1	0	0	0	0	1	1
6	0	0	1	0	0	1	0	4
7	0	0	1	0	0	0	1	1

the algorithms will estimate the model parameters for each level of the variables (see the discussion of the PROC FACTMAC output later in this section). The MovieLens data set is given in several sizes and for the purposes of this illustration we will use the 100K data set. For a more detailed discussion of this data set refer to Candillier et al. (2007) and Mondal (2018).

The MovieLens 100K data set comprise of 100 000 ratings completed by 943 users for 1 682 movies. The input X-matrix is clearly very sparse since it contains only 100 000 ratings out of a possible 1 586 126$(943 \times 1\,682)$ and, using one-hot encoding, a number of 2 625$(943 + 1\,682)$ 'new' predictors. The extent of the sparsity is due to the fact that not all users rate all the movies, and the considerable increase in predictors due to the coding of the nominal variables.

In the next subsections it is shown how the LibFM and PROC FACTMAC packages may be used to achieve this and in the last subsection a comparison is made of the performance of the algorithms.

LibFM

To illustrate the format of files required in LibFM, an illustrative example, as given in the LibFM user manual, is explained before moving to the MovieLens data. The LibFM implementation supports two file formats for input data: a text format and a binary format. Since the text format is easier and recommended for new LibFM users, it was the format used in this example. The first few observations of an example data set are given in Table 4.

When text format is used, the user needs to ensure that each row states first the value Y and then the nonzero values of X as given in Exhibit 1. The response of each of the three cases is thus 4 for the first, 2 for the second and -1 for the third. After the response, each line contains the nonzero elements of X, where an entry like 0:1.5 reads $X_0 = 1.5$ and thus $X_3 = -7.9$, etc. The left side of INDEX:VALUE states the index within X whereas the right side states the value of $X_{\text{INDEX}} = \text{VALUE}$. Once the data are in this format, an FM model can be developed in LibFM.

Additionally, in recommender systems a file format like *userid*, *itemid* and *rating* are typically used. When such models are developed in LibFM, a two-step process is followed. Start by using a Perl script to convert such data sets to the LibFM file format (this is included in the script's directory when LibFM is installed). It is important to note that Perl needs to be installed before the conversion script can be executed in LibFM. An extract for the sorted 100K MovieLens data used is given in Table 5.

The code as given in Exhibit 2 is used to convert a recommender system file to LibFM format. Note that this code specifies that the response values are found in column 2 (numbering starts at 0) and that the TimeStamp variable be dropped.

Exhibit 3 gives an extract of the resulting LibFM file ready to be used in model development. This shows that user 1 gave a rating of 5 for movie 1, 3 for movie 2 and 4 for movie 3. The last entry for

Table 4. Extract of sample data set.

Y	X_0	X_1	X_2	X_3	X_4	X_5	X_6
4	1.5	0	0	−7.9	0	0	0
2	0	10^{-5}	0	2	0	0	0
−1	0	0	0	0	0	0	1

Table 5. Extract of sorted 100K MovieLens data.

UserID	*ItemID*	*Rating*	*TimeStamp*
1	1	5	874965758
1	2	3	876893171
1	3	4	878542960

Exhibit 1: LibFM data format (text)

```
4 0:1.5 3:-7.9
2 1:1e-5 3:2
-1 6:1
...
```

Exhibit 2: LibFM conversion code

```
./triple_format_to_libfm.pl -in ratings.dat -target 2 -delete_column 3
-separator "\t"
```

Exhibit 3: LibFM converted recommender system file

```
5 0:1 1:1
3 0:1 2:1
4 0:1 3:1
. . .
3 0:1 272:1
4 273:1 1:1
2 273:1 10:1
. . .
```

Exhibit 4: Code used to create FM model in LibFM

```
SGD:
libFM -task r -train Ratings.dat -test Ratings.dat -dim '1,1,10'
-iter 1000 -method sgd -learn_rate 0.01 -init_stdev 0.5
-out Predicted.csv
ALS:
libFM -task r -train Ratings.dat -test Ratings.dat -dim '1,1,10'
-iter 1000 -method als -init_stdev 0.5 -out Predicted.csv
MCMC:
libFM -task r -train Ratings.dat -test Ratings.dat -dim '1,1,10'
-iter 1000 -method mcmc -init_stdev 0.5 -out Predicted.csv
SGDA:
libFM -task r -train Ratings.dat -test Ratings.dat -dim '1,1,10'
-iter 1000 -method sgda -learn_rate 0.01 -init_stdev 0.5
-validation Ratings.dat -out Predicted.csv
```

Exhibit 5: Code used to create FM model in PROC FACTMAC

```
cas mysession;
libname mycas cas;
libname MyData "/home/user3/sasuser.viya";

/* Import movielens into SAS */

proc factmac data= mycas.movielens nfactors=10 learnstep=0.01 maxiter=20
outmodel= mycas.factors;
input User Item /level=nominal;
target Rating /level=interval;
output out= mycas.out copyvars=(User Item Rating);
run;
```

user 1 is 3 0:1 272:1. This means that user 1 rated 272 movies, with a rating of 3 for movie 272. The first entry for user 2 is thus 4 273:1 1:1. Note that user 2 is now numbered as 273 (the next available number after the 272 movies rated by user 1). User 2 also watched movie 1 and gave a rating of 4. The second entry for user 2 is 2 273:1 10:1. This means that user 2 didn't rate movies $2 - 9$, but gave a rating of 2 to movie 10. Once the file is in the correct format, the LibFM algorithms can be performed by making use of the code given in Exhibit 4. Refer to the LibFM manual for more information on the selection of parameter values used in the code.

SAS PROC FACTMAC

PROC FACTMAC is one of the machine learning procedures that is available in SAS Visual Data Mining and Machine Learning. It is specifically developed to take advantage of the distributed environment that the SAS Viya platform provides. In order to develop an FM model on the MovieLens data in PROC FACTMAC, the code as given in Exhibit 5 can be used. Note that here we can specify whether the variables used are interval- or nominal-scaled variables and has the advantage that it automatically handles the dummy coding for nominal variables.

Performance comparison

This section compares the performance of the LibFM algorithms (MCMC, ALS and SGD) and SAS's PROC FACTMAC on the MovieLens data. Since SAS provides algorithms for fitting OLS and LASSO regression with all two-way interactions, we included these methods in our comparison. SAS PROC GLM was used to calculate the OLS fits. As stated previously, OLS regression becomes unworkable when the X matrix becomes singular. This happens when $P > N$ and/or when very high correlations between the predictor variables are present (the well-known problem of multi-collinearity). Naturally, there are also regularised regression methodologies that cater for the low-N high-P problem (like the Ridge and LASSO regression) so it was decided to include the LASSO for comparison purposes. Refer to Hastie et al. (2015) for more information on these models, and for statistical learning with sparsity in general. Note that the LASSO is given as an option in PROC

GLMSELECT together with various regularisation options. In our case, the Schwartz–Bayesian Criterion (SBC) was used throughout.

In this set, two categorical variables, UserID and ItemID, are considered with 943 and 1 682 levels respectively. PROC GLM was performed but an error message occurred stating that GLM cannot handle this number of dummy variables. When the LASSO was performed using PROC GLM SELECT, a solution could not be obtained due to "insufficient memory".

In Table 6 the MSE of the errors obtained for LibFM and SAS algorithms are given where the maximum number of iterations are set at 20, 1 000 and 10 000 respectively. The number of factors, G, is set to 10 for all implementations. Note that this choice is rather arbitrary, but the value seemed good when the number of new predictors created due to the nominal variables is taken into account, i.e. the selection of G is still small (when compared to the number of variables) as recommended by Rendle (2010). From Table 6, as far as improvement in MSE is concerned, we can conclude that 1 000 iterations is a reasonable setting since not much fitting accuracy is gained by increasing the maximum number of iterations beyond 1 000. The MSE values obtained by all these algorithms are good, with SGD and ALS the better algorithms in this case. However, to get a clear answer to the best algorithm to use, a comprehensive simulation study is required. The run time of the algorithms dramatically increased when the maximum iterations are set at 10 000. This was especially the case with the LibFM algorithms that took 13 minutes to complete, while PROC FACTMAC finished in less than 3 minutes. For the lower settings (20 and 1 000) the run time of all the algorithms were less than 2 minutes.

Box-and-whisker plots of the error distribution of the fits obtained for the algorithms (with the iterations set to 1 000) appear in Figure 1. These plots are graphical representations of some summary statistics. The length of the box represents the interquartile range (the distance between the 25th and 75th percentiles). The symbol in the box interior represents the group mean and the horizontal line represents the group median. The vertical lines issuing from the box extend to the group minimum and maximum values not deemed outlier values. The circles above or below these extended lines are outlier values. Again, the error distributions are similar, with MCMC and PROC FACTMAC slightly wider when comparing the box (observations between the 25th and 75th percentiles).

In conclusion, both the LibFM and FACTMAC packages fit the MovieLens data set well (where OLS and LASSO could not obtain a solution). This clearly indicates that FMs are very successful in the estimation of parameters in a very sparse or high-dimensional setting. Next, we consider a similar study on a regression data set.

Table 6. Comparison of the MSE achieved by the algorithms.

	LibFM			**SAS**		
Iterations	**MCMC**	**ALS**	**SGD**	**FACTMAC**	**OLS**	**LASSO**
20	0.580	0.463	0.610	0.700	No solution	Insufficient memory
1 000	0.519	0.428	0.425	0.472	No solution	Insufficient memory
10 000	0.520	0.427	0.421	0.467	No solution	Insufficient memory

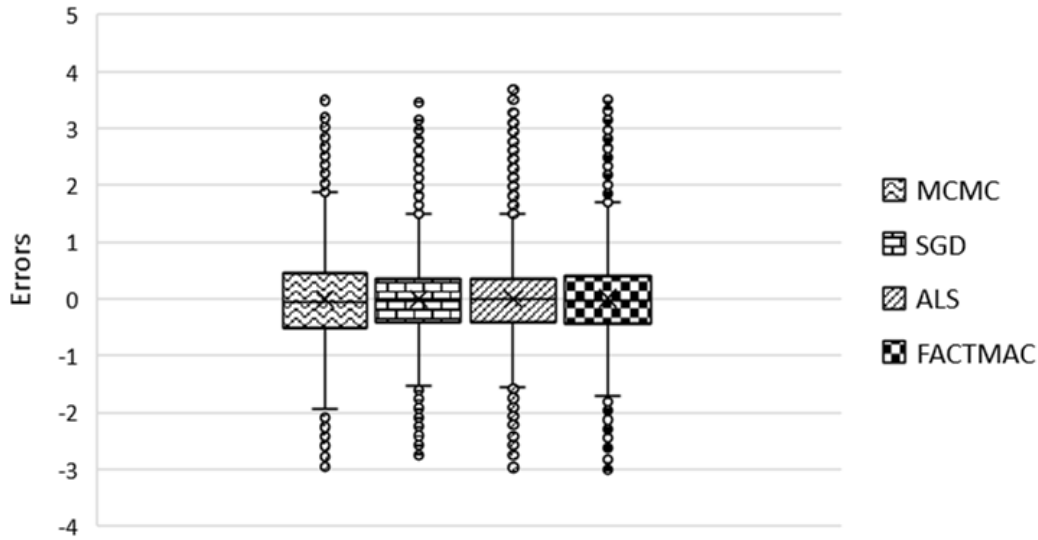

Figure 1. Box-and-whisker plots of the errors obtained by the algorithms.

5.2 Regression example

In order to study the performance of FMs in a regression context, the regression model with two-way interactions as given in (1) was used. Note that all analyses were performed on a laptop computer (i7 processor, 1.8 GHz CPU) unless otherwise specified. Different sized data sets were generated where the X-matrix consisted of predictors that were generated as $N(0, 1)$ variables. The response variable Y was then obtained by assuming that all regression parameters have a value of 1 and by adding an error term from the $N(0, 0.25)$ distribution. Data sets were generated for values of $K = 10, 20, 50, 100, 200$ and $N = 1\,000, 5\,000, 10\,000$ in order to compare the LibFM algorithms with the performance of ordinary least squares (OLS) regression with two-way interactions and LASSO regression. One would expect a good model to obtain (at best) an MSE around 0.25. A value lower than 0.25 indicates that the model tends to overfit and a value higher than 0.25 that the model tends to underfit. SAS PROC GLM was used to calculate the regression fits and PROC GLMSELECT for the LASSO fits. In order to illustrate the performance of the different methods, the results obtained for the $K = 50$ case are given in Table 7, with the run time in seconds given in brackets within the table. The number in brackets after the type of model used indicates the number of parameters.

When $K = 50$ a total of $1\,276$ $(1 + 50 + 50 \times 49/2)$ parameters have to be estimated by GLM and GLMSELECT for both the OLS and LASSO regressions, respectively. For FMs, G was taken as $G = 2, 5, 10$ and 20, resulting in 151 ($G = 2$), 301 ($G = 5$), 551 ($G = 10$) and $1\,051$ ($G = 20$) coefficients to be estimated by LibFM algorithms SGD, ALS and MCMC. From Table 7 the following conclusions can be drawn:

- Regression with all two-way interactions and the LASSO tend to overfit the data for all sample sizes studied. The extent of overfitting reduces with increasing sample size.
- The ALS and MCMC algorithms produce good fits (close to 0.25) for $G = 2$ at all the sample

Table 7. MSE results and run time of regression data set with 50 predictor variables.

		Number of observations		
Modelling technique used		**1 000**	**5 000**	**10 000**
Regression with 2-way interactions (1 276)		no answer (10.43)*	0.183 (6.62)	0.215 (23.16)
LASSO (1 276)		$2.97E^{-07}$ (18.17)**	0.185 (30.17)	0.216 (33.15)
LibFM	SGD – 2 factors (151)	0.356 (6.13)	0.408 (11.88)	0.386 (15.98)
	SGD – 5 factors (301)	0.307 (9.43)	0.400 (20.26)	0.379 (30.45)
	SGD – 10 factors (551)	0.108 (12.30)	0.387 (34.87)	0.356 (51.48)
	SGD – 20 factors (1 051)	0.045 (18.95)	0.357 (50.58)	0.332 (70.23)
	ALS – 2 factors (151)	0.246 (6.13)	0.247 (14.73)	0.252 (25.66)
	ALS – 5 factors (301)	0.180 (9.37)	0.232 (26.20)	0.244 (45.22)
	ALS – 10 factors (551)	0.104 (13.86)	0.219 (43.92)	0.235 (84.28)
	ALS – 20 factors (1 051)	0.026 (19.81)	0.204 (76.78)	0.225 (155.99)
	MCMC – 2 factors (151)	0.236 (6.47)	0.263 (9.5)	0.299 (26.75)
	MCMC – 5 factors (301)	0.181 (9.41)	0.241 (16.33)	0.249 (50.31)
	MCMC – 10 factors (551)	0.123 (12.60)	0.230 (29.18)	0.242 (88.20)
	MCMC – 20 factors (1 051)	0.052 (19.80)	0.210 (53.78)	0.231 (157.04)

* Note: The $X'X$ matrix was found to be singular and a generalised inverse was used to solve the normal equations.
** The adaptive weights for the LASSO method are not uniquely determined because the full least squares model is singular.

sizes. However, when the number of factors increases, both methods seem to overfit the data, especially at smaller sample sizes. This conclusion confirms Rendle's remark that a small number of factors is generally a better choice for model generalisation.

- With FM regression, as the number of factors increases, the MSEs improve but, as expected, the run time is longer. At small choices of G ($G \leq 5$) the run time of the FM algorithms are mostly better than the LASSO.
- Overall in this example we can conclude that the FM algorithms (ALS and MCMC) outperform the other modelling techniques when $G = 2$ since they produce good fits (no overfitting), the run times are similar or less and the number of parameters used in these models is greatly reduced.

In summary, when the number of observations and predictor variables are 'small' and $P < N$, OLS and LASSO regression with interaction overfit the data. In this example and for the cases considered here, overfitting was achieved for $K \leq 100$ and $N \leq 10\,000$. However, for the case $K = 200$ and $N = 100\,000$, the OLS regression did not obtain a solution after seven days while the MCMC algorithm with $G = 10$ obtained a solution in about three hours (performance results not shown). The OLS regression was then performed on a server (128GB DDR RAM @ 800MHz

and speed of hard drive is 10k RPM) and obtained a solution within 24 hours. However, the MSE obtained by MCMC was much higher than that obtained by OLS regression, although the latter tends to overfit. Although this cannot be regarded as a conclusive study, the results indicate that FMs do not have a substantial advantage over traditional statistical methods in a dense classical regression context (except with a reduction of the number of parameters required), however it might lead to better generalisation when a small number of factors is chosen. This will have to be determined by conducting a simulation study in a prediction setting. Harris (2015) stated that FMs perform extremely well on sparse data, including data of very high sparsity. But as a trade-off, they do not perform well on dense data so other algorithms are more suited to this class of data, as seen in our analyses with results given in Table 6 and Table 7.

To conclude, in all cases considered here, SAS PROC GLM and GLMSELECT with OLS and LASSO (with all two-way interactions) outperformed the LibFM in terms of accuracy. The LibFM algorithms for small choices of G sometimes have significant speed advantage (fewer parameters to estimate) and may lead to better generalisation. It should be noted that in a big data context the conclusions might be different, since the data sets generated here are very much 'small-scale data sets', so the potential benefit of FMs might arise when studying large-scale data sets. This is left for future research.

5.3 Logistic regression example

In finance, logistic regression is frequently used to build credit scorecards for assessing the credit-worthiness of clients. Typically large data sets containing millions of records are used, and the data are split into a training and test set in order to evaluate the predictive power of the scorecards. One issue is that these data sets are typically classified. In order to investigate the performance of FMs in this context, differently sized data sets with $N = 100, 1\,000, 10\,000$ and $100\,000$ were generated by assuming a logistic regression model and by using simulation.

The linear part of the model contained ten predictor variables and all their two-way interactions plus an error term. The predictor variables were independently generated from a standard normal distribution $N(0, 1)$ and the error term from an $N(0, 0.25)$ distribution. The coefficients of the predictor variables and their interactions were assumed to be 1. The binary target variable (Y) is then obtained by applying the logistic link function to the linear term and by comparing the result to a uniformly generated variable. The interested reader is referred to de Jongh et al. (2015) for more information.

The simulated training and test data sets were then augmented by including another five variables (again generated from $N(0, 1)$) with their two-way interactions and the FMs and logistic regression were fitted on these augmented data sets in order to determine how well the true model is estimated in the presence of noise.

The results, by using the Gini as performance measure, are given in Table 8. The Gini statistic measures the discriminatory power of a model with values varying between 0 and 1. Large values correspond to stronger associations between the predicted and observed values. Refer to Allison (2003) for more information.

Note that, because the data sets are generated using a logistic regression model, one would expect that the traditional model will perform better than FMs since the latter use much less model parameters. However, the performance of the FM-based logistic regression is quite surprising:

Table 8. Gini results of logistic regression data sets with ten predictor variables.

	Gini statistic							
	SAS: Proc Logistic		**LibFM: MCMC**		**LibFM: MCMC**		**LibFM: MCMC**	
Number of parameters	55		30		40		50	
Number of factors	N/A		2		3		4	
Data set	Train	Test	Train	Test	Train	Test	Train	Test
$N = 100$	1.0000*	0.1680	1.0000	0.8368	1.0000	0.8333	1.0000	0.7804
$N = 1\,000$	0.9690	0.8643	0.9431	0.9123	0.9468	0.9101	0.9502	0.9079
$N = 10\,000$	0.9405	0.9287	0.9390	0.9312	0.9394	0.9308	0.9396	0.9305
$N = 100\,000$	0.9357	0.9340	0.9355	0.9342	0.9355	0.9342	0.9355	0.9342

* Warning: There is complete separation of data points. The maximum-likelihood estimate does not exist. The LOGISTIC procedure continues in spite of the above warning. Results shown are based on the last maximum-likelihood iteration. Validity of the model fit is questionable.

- As far as the performance on the training sets is considered, traditional logistic regression does slightly better than FM logistic regression. Note that the extent of the outperformance is, as expected, less as the number of FM factors and sample size increase.
- When performance on the test data sets is studied, FM -based logistic regression does better than traditional logistic regression and the extent of the outperformance is more pronounced when the number of factors ($G = 2$) and sample sizes ($N <= 10\,000$) are smaller.

To conclude, from the three examples studied here, the potential gains of using FMs as modelling technique have been illustrated. These include:

- The efficient estimation of model parameters in a high-dimensional and sparse-data setting (MovieLens example).
- Comparable performance to traditional methods, but much faster computationally due to the fact that fewer parameters and thus simpler models are used.
- An indication that FM-based regression and logistic regression with a small number of factors seem to generalise better, i.e. their prediction performance is better than traditional methods especially at smaller sample sizes.

6. Conclusion and direction for further research

FMs have obtained huge successes in the area of recommender systems. Recommender system data sets are characterised by high dimensionality and sparsity. Thus, problems where the number of predictor variables typically outnumbers the number of observations and when all possible values in the predictor space are considered, many of the observations are 'missing' (not observed). As illustrated in this paper, the above comment is clear from our analysis of the MovieLens data (sparse or high-dimensional data). Our analysis shows that both the LibFM and PROC FACTMAC packages fit the MovieLens data set well (where OLS and LASSO could not obtain a solution).

The performance benefits of using FMs in a classical regression setting are less obvious. Our analysis showed that, in a classical regression context, OLS and LASSO outperformed FMs (LibFM) in terms of accuracy but they tend to overfit the data set studied here. The LibFM algorithms for small choices of G sometimes have significant speed advantage (fewer parameters to be estimated) and may lead to better generalisation. When the number of parameters is considered, we can argue the advantage of the use of FM models (LASSO used 1 276 parameters versus FMs from as little as 151 parameters).

When logistic regression models are considered, and the performance of logistic regression models is compared to FM models, we can conclude that FMs clearly outperformed logistic regression models when smaller samples are considered from our analysis (in terms of Gini values obtained on the test sets). As the sample size increases, FMs still slightly outperformed logistic regression models by using fewer parameters.

It should be clear from the list of references given in this paper that the overwhelming majority are in conference proceedings, and a few are in computer science and engineering journals. It is interesting to note that neural networks, a well-known machine learning technique that originated in the 1940s (see e.g. McCulloch and Pitts (1943) and Rosenblatt (1958)), also took some time before it was being included in practising statisticians' model-building toolkit. The first paper on neural networks appeared almost fifty years later in the local statistical journal (see de Jongh and de Wet (1993)). Nowadays, practising statisticians frequently make use of neural networks and we hope that this paper will contribute to the incorporation of FMs in the model-building toolkit of statisticians.

From the nonstandardised use of statistical terminology and the lack of rigorous statistical theory in the literature, it is evident that FM research is still at the beginning of the statistical learning curve. Therefore, future research on FMs presents many opportunities. Apart from the standard statistical questions that can be researched, many open problems exist. For instance, what is the optimum choice for the number of factors for a particular problem? According to Rendle (2010), in sparse settings, typically the number of factors used should be small because there is not enough data to estimate complex interactions. He claims that when the number of factors is restricted it leads to better generalisation and thus improved interaction matrices under sparsity. See Rosasco et al. (2004). These claims should be researched by, among others, comprehensive simulation studies. In the field of model development, computationally efficient, time efficient and robust learning methods on big data and complex interactions are necessary. More advanced modelling aspects such as the distribution of the errors and multi-collinearity issues currently play no role. So research into statistical inference is needed, such as model selection, over- and underfitting, and the effects of outliers.

In conclusion, the successes of FMs warrant the attention of statisticians both for practical application and as a fruitful research area. We hope that statisticians will take up this challenge.

Acknowledgements. The authors would like to thank Prof. Hennie Venter and Prof. Freek Lombard for valuable contributions to the paper and would also like to thank Prof Tertius de Wet for careful reading of the final manuscript. This work is based on research supported in part by the Department of Science and Innovation (DSI) of South Africa. The grant holder acknowledges that opinions, findings and conclusions or recommendations expressed in any publication generated by DSI-supported research are those of the authors and that the DSI accepts no liability whatsoever in this regard.

References

Allison, P. (2003). *Logistic Regression: Using the SAS System*. SAS Institute and Wiley & Sons.

Bhatnagar, V. (2016). *Collaborative Filtering Using Data Mining and Analysis*. IGI Global, Hershey, PA.

Blondel, M., Fujino, A., Ueda, N., and Ishihata, M. (2016). Higher-order factorization machines. *In Advances in Neural Information Processing Systems 29*. Barcelona, Spain.

Candillier, L., Meyer, F., and Boullé, M. (2007). Comparing state-of-the-art collaborative filtering systems. *In Machine Learning and Data Mining in Pattern Recognition. MLDM 2007. Lecture Notes in Computer Science*, volume 4571. Springer, Berlin, 548–562.

de Jongh, P., de Jongh, E., Pienaar, M., Gordon-Grant, H., Oberholzer, M., and Santana, L. (2015). The impact of pre-selected variance inflation factor thresholds on the stability and predictive power of logistic regression models in credit scoring. *ORiON*, **31**, 17–37.

de Jongh, P. and de Wet, T. (1993). An introduction to neural networks. *South African Statistical Journal*, **27**, 103–128.

Freudenthaler, C., Schmidt-thieme, L., and Rendle, S. (2009). Factorization machines: Factorized polynomial regression models.
URL: *http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.364.8661*

Hancock, J. T. and Khoshgoftaar, T. M. (2020). Survey on categorical data for neural networks. *Journal of Big Data*, **7**, 1–41.

Harper, F. M. and Konstan, J. A. (2015). The MovieLens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems*, **5**, 1–19.

Harris, B. (2015). Factorization machines: A new way of looking at machine learning.
URL: *https://securityintelligence.com/factorization-machines-a-new-way-of-looking-at-machine-learning/*

Harshman, R. A. (1970). Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multimodal factor analysis. *UCLA Working Papers in Phonetics*, **16**, 1–84.

Hastie, T., Tibshirani, R., and Wainwright, M. (2015). *Statistical Learning with Sparsity: The Lasso and Generalizations*. CRC Press, Boca Raton, FL.

McClory, P. (2018). Factorization machines for machine learning.
URL: *https://medium.com/@pmdev/factorization-machines-for-machine-learning-63b549c86111*

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, **5**, 115–133.

Mondal, A. (2018). Factorization machines for movie recommendations.
URL: *http://www.stokastik.in/factorization-machines-for-movie-recommendations/*

Parsons, N. (2017). Factorization machines for recommendation systems.
URL: *https://getstream.io/blog/factorization-recommendation-systems/*

Potdar, K., Pardawala, T. S., and Pai, C. D. (2017). A comparative study of categorical variable encoding techniques for neural network classifiers. *International Journal of Computer Applications*, **175**, 7–9.

Rendle, S. (2010). Factorization machines. *In 2010 IEEE International Conference on Data Mining*. Sydney, Australia, 995–1000.

Rendle, S. (2012). Factorization machines with LibFM. *ACM Transactions on Intelligent Systems and Technology*, **3**, 1–22.

Rendle, S. and Schmidt-Thieme, L. (2010). Pairwise interaction tensor factorization for personalized tag recommendation. *In Proceedings of the Third ACM International Conference on Web Search and Data Mining*. New York, USA, 81–90.

Rosasco, L., Vito, E. D., Caponnetto, A., Piana, M., and Verri, A. (2004). Are loss functions all the same? *Neural Computation*, **16**, 1063–1076.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, **65**, 386.

SAS Institute Inc. (2017). SAS® Visual Data Mining and Machine Learning 8.2 Procedures. URL: *https://documentation.sas.com/api/docsets/casml/8.2/content/casml.pdf?locale=en*

SAS Institute Inc. (2019a). An Introduction to SAS® Viya® 3.4 Programming. URL: *https://documentation.sas.com/api/docsets/pgmdiff/3.4/content/pgmdiff.pdf?locale=en*

SAS Institute Inc. (2019b). SAS® Visual Data Mining and Machine Learning 8.5: Procedures. URL: *https://documentation.sas.com/?docsetId=casml&docsetTarget=casml_factmac_syntax07.htm&docsetVersion=8.5&locale=ja*

Silva, J. and Wright, R. (2017). Factorization machines: A new tool for sparse data. *In SAS Global Forum*. Orlando, FL.

Srebro, N. and Jaakkola, T. (2003). Weighted low-rank approximations. *In Proceedings of the 20th International Conference on Machine Learning*. Washington DC, 720–727.

Srebro, N., Rennie, J. D. M., and Jaakola, T. S. (2005). Maximum-margin matrix factorization. *In Advances in Neural Information Processing Systems 17*. MIT Press, Cambridge, MA, 1329–1336.

Steinwart, I. and Christmann, A. (2008). *Support Vector Machines*. Springer, New York, NY.

Tucker, L. R. (1966). Some mathematical notes on three-mode factor analysis. *Psychometrika*, **31**, 279–311.

Yurochkin, M., Nguyen, X., and Vasiloglou, N. (2017). Multi-way interacting regression via factorization machines. *In Proceedings of the 31st Conference on Neural Information Processing Systems*. Long Beach, CA, 2595–2603.

Manuscript received 2020-07-03, revised 2021-02-04, accepted 2021-03-24.