Paper SAS388-2017

Factorization Machines: A New Tool for Sparse Data

Jorge Silva and Raymond E. Wright, SAS Institute Inc.

ABSTRACT

Factorization machines are a new type of model that is well suited to very high-cardinality, sparsely observed transactional data. This paper presents the new FACTMAC procedure, which implements factorization machines in SAS® Visual Data Mining and Machine Learning. This powerful and flexible model can be thought of as a low-rank approximation of a matrix or a tensor, and it can be efficiently estimated when most of the elements of that matrix or tensor are unknown. Thanks to a highly parallel stochastic gradient descent optimization solver, PROC FACTMAC can quickly handle data sets that contain tens of millions of rows. The paper includes examples that show you how to use PROC FACTMAC to recommend movies to users based on tens of millions of past ratings, predict whether fine food will be highly rated by connoisseurs, restore heavily damaged high-resolution images, and discover shot styles that best fit individual basketball players.

INTRODUCTION

Factorization models, which include factorization machines as a special case, are a broad class of models popular in statistics and machine learning. For example, principal component analysis is a well-known type of factorization model that has long been a staple of dimensionality reduction. For another example, matrix factorization has been widely used in text analysis and recommender systems. More recently, Rendle (2010, 2012) has proposed factorization machines for recommender systems and click-through rate prediction. Factorization machines are a powerful model that significantly extends matrix factorization.

Factorization machines are included in SAS Visual Data Mining and Machine Learning. The initial release supported matrix factorization with biases, and the latest implementation supports pairwise-interaction tensor factorization and nonnegative factorization. A macro is provided in the Appendix so that you can still perform pairwise-interaction tensor factorization even if you have PROC FACTMAC from the first release of SAS Visual Data Mining and Machine Learning.

This paper begins by briefly explaining the most relevant technical details of factorization machines for data scientists. Then it focuses on applications of factorization machines to solve real-world business problems. The application sections, which can be read by non-experts, include usage tips as well as code. The following application examples are presented:

- recommending movies to users based on tens of millions of past ratings
- predicting whether a fine food item will be highly rated by connoisseurs
- restoring heavily damaged high-resolution images
- discovering shot styles that best fit individual basketball players

From these examples, you will learn how to spot which types of problems are good candidates for factorization machines, how to prepare data for PROC FACTMAC, how to score new data by using score code or PROC ASTORE, and what strategies to use for choosing parameters and training the best factorization machine models.

THE FACTORIZATION MACHINE MODEL

This section begins with a brief mathematical description of factorization machines. Assuming a training set $D = \{(\boldsymbol{x}_i, y_i)\}$, with $i = 1, \ldots, n$, where $\boldsymbol{x}_i$ refers to the *i*th observation and y_i refers to the *i*th target value, the factorization machine model of order 2 is written as

$$\hat{y}(x) = w_0 + \sum_{j=1}^{p} w_j x_j + \sum_{j=1}^{p} \sum_{j'=1}^{p} x_j x_{j'} \sum_{f=1}^{k} v_{jf} v_{j'f}$$

where $\boldsymbol{x} = (x_1, \ldots, x_p)$ is an observed *p*-dimensional input feature vector, $\hat{y}$ is the predicted target,

w_0 is a global bias, w_j are per-feature biases, and v_{jf} denotes coordinate *f* of the *k*-dimensional factor vector $\boldsymbol{v}_j$.

The overall factor matrix $\boldsymbol{V}$ of size $p \times k$ is the concatenation of the row vectors $\boldsymbol{v}_j$ for $j = 1, \ldots, p$.

The number of factors is *k*. The model parameters to be estimated are $w_0, w_1, \ldots, w_p$ and $\boldsymbol{V}$.

The estimation is done by minimizing the root mean square error (RMSE), which is defined as

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (\hat{y}(x_i) - y_i)^2}$$

over the training set D.

Interestingly, it is known that factorization machines approximate polynomial-kernel support vector machines and are more resistant to overfitting when the design matrix is sparse (Rendle 2010).

FACTORIZATION MACHINES FOR RECOMMENDATIONS

Recommender systems are a diverse class of algorithms that aim to learn user preferences in order to recommend items such as movies, books, or songs. The purpose is to predict which ratings a user would hypothetically give to a set of items and then to recommend items that the user is likely to prefer the most. As illustrated in Figure 1, users and items form a matrix. This matrix is potentially very large, because there can be millions of users and items. Moreover, it is very sparsely observed, because usually only a very small fraction of historical ratings are available.

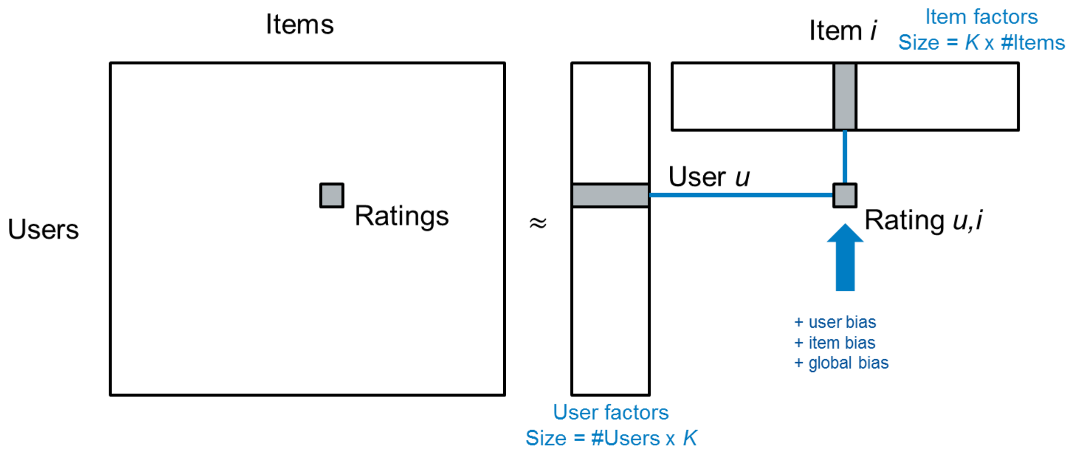


Figure 1. Matrix factorization for a recommender system. Users and items are characterized by their respective *k*-dimensional factor vectors.

You can overcome these challenges by factorizing the matrix into lower-dimensional user and item factors, which can be used to predict new ratings. For recommender systems, the input vector is typically constructed using binary indicator variables for user *u* and item *i*, as illustrated in Figure 2.

$$\boldsymbol{x} = (\underbrace{0,\ldots,0,1,0,\ldots,0}_{|\boldsymbol{U}|},\underbrace{0,\ldots,0,1,0,\ldots,0}_{|\boldsymbol{I}|})$$

Figure 2. Input vector for recommender systems.

The factorization machine model is then equivalent to the following equation for predicting new ratings:

$$\hat{y}(\boldsymbol{x}) \ = \hat{y}(u,i) \ = w_0 + w_u + w_i + \sum_{f=1}^{k} v_{uf} v_{if}$$

EXAMPLE: RECOMMENDING MOVIES

This example draws on data that are derived from companies that provide movies for online viewing. A company wants to offer its customers recommendations of movies that they might like. These recommendations are based on ratings that users provide.The `MovieLens` data set, which contains movie ratings, was developed by the GroupLens project at the University of Minnesota and is available at http://grouplens.org/datasets/movielens (Harper and Konstan 2015). This example uses the `MovieLens` 100K version.

The `MovieLens` 100K data set has four columns: user ID, item ID (each item is a movie), timestamp, and rating. This example predicts the rating for a specified user ID and an item ID. The data set is very sparse because most combinations of users and movies are not rated.

You can download the compressed archive file from the URL http://files.grouplens.org/datasets/movielens/ml-100k.zip and use any third-party unzip tool to extract all the files from the archive and store them in the destination directory of your choice. The file that contains the ratings is named `u.data`. Assuming that your destination directory is `~/data`, the following DATA step loads the data table from the directory into your CAS session:

```
proc casutil;
   load file="~/data/u.data"    /*or other user-defined location*/
   casout="movlens"
   importoptions=(filetype="CSV" delimiter="TAB" getnames="FALSE"
                vars=("userid" "itemid" "rating" "timestamp"));
run;
```

The following statements show how to use PROC FACTMAC to predict movie ratings:

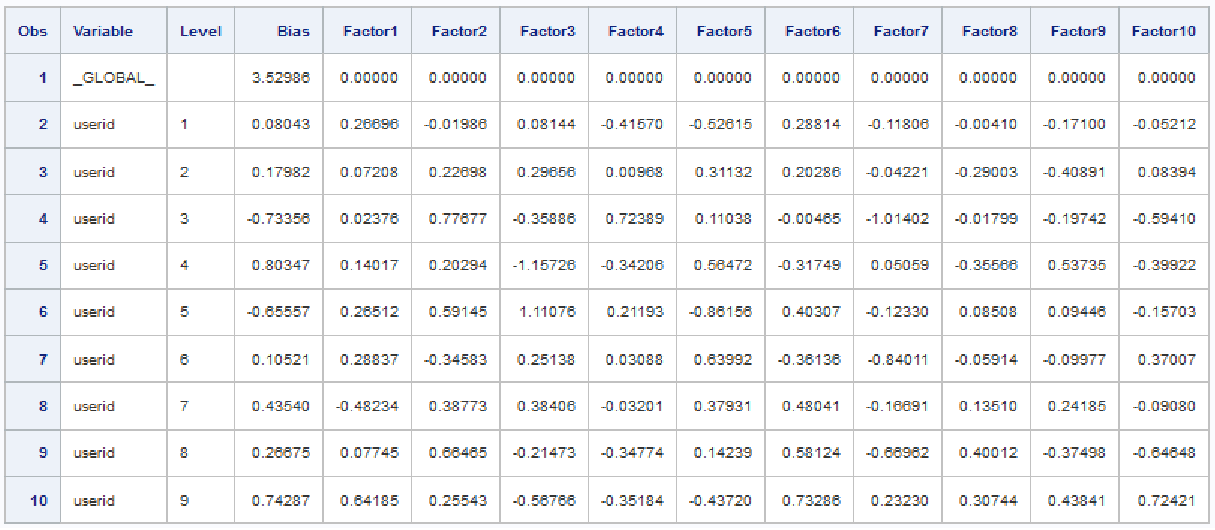
```
proc factmac data=mycas.movlens nfactors=10 learnstep=0.15
                                 maxiter=20  outmodel=factors;

   input userid itemid /level=nominal;
   target rating /level=interval;
   output out=mycas.out1 copyvars=(userid itemid rating);
run;
```

The NFACTORS parameter corresponds to *k* in the model equations. The LEARNSTEP parameter is an optimization parameter that controls how fast the stochastic gradient descent solver learns. Smaller values increase accuracy but might require a larger number of iterations to reach a good solution.

The following statements print the first 10 observations in the `Factors` data table, which is specified in the OUTMODEL= option in the PROC FACTMAC statement. The output is shown in Figure 3.

```
proc print data=factors(obs=10);
run;
```



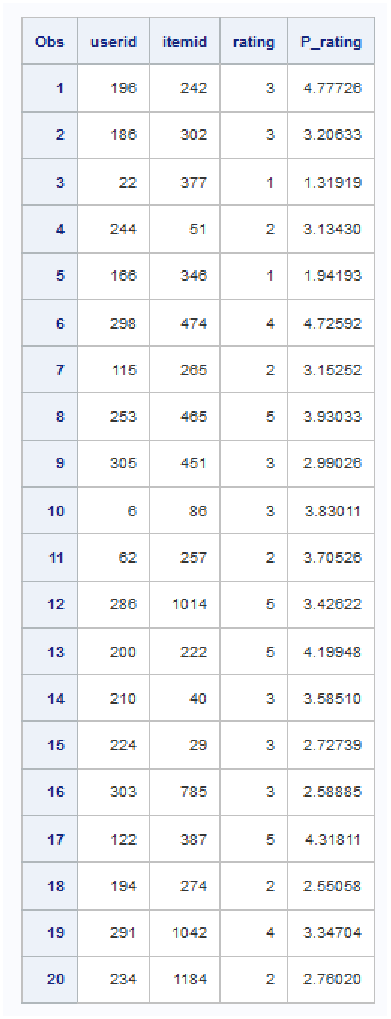
Obs	Variable	Level	Bias	Factor1	Factor2	Factor3	Factor4	Factor5	Factor6	Factor7	Factor8	Factor9	Factor10
1	_GLOBAL_		3.52986	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
2	userid	1	0.08043	0.26696	-0.01986	0.08144	-0.41570	-0.52615	0.28814	-0.11806	-0.00410	-0.17100	-0.05212
3	userid	2	0.17982	0.07208	0.22698	0.29656	0.00968	0.31132	0.20286	-0.04221	-0.29003	-0.40891	0.08394
4	userid	3	-0.73356	0.02376	0.77677	-0.35886	0.72389	0.11038	-0.00465	-1.01402	-0.01799	-0.19742	-0.59410
5	userid	4	0.80347	0.14017	0.20294	-1.15726	-0.34206	0.56472	-0.31749	0.05059	-0.35566	0.53735	-0.39922
6	userid	5	-0.65557	0.26512	0.59145	1.11076	0.21193	-0.86156	0.40307	-0.12330	0.08508	0.09446	-0.15703
7	userid	6	0.10521	0.28837	-0.34583	0.25138	0.03088	0.63992	-0.36136	-0.84011	-0.05914	-0.09977	0.37007
8	userid	7	0.43540	-0.48234	0.38773	0.38406	-0.03201	0.37931	0.48041	-0.16691	0.13510	0.24185	-0.09080
9	userid	8	0.26675	0.07745	0.66465	-0.21473	-0.34774	0.14239	0.58124	-0.66962	0.40012	-0.37498	-0.64648
10	userid	9	0.74287	0.64185	0.25543	-0.56766	-0.35184	-0.43720	0.73286	0.23230	0.30744	0.43841	0.72421

Figure 3. Factors data table for the MovieLens data set.

When the model is saved in the `mycaslib.astore` data table, you can predict new ratings by using the ASTORE procedure, as in the following statements:

```
proc astore;
   score data = mycaslib.valid
   out=mycaslib.ScoreValid copyvar = rating
   rstore = mycaslib.astore;
run;
```

The first 20 predicted ratings are shown in Figure 4.



Obs	userid	itemid	rating	P_rating
1	196	242	3	4.77726
2	186	302	3	3.20633
3	22	377	1	1.31919
4	244	51	2	3.13430
5	166	346	1	1.94193
6	298	474	4	4.72592
7	115	265	2	3.15252
8	253	465	5	3.93033
9	305	451	3	2.99026
10	6	86	3	3.83011
11	62	257	2	3.70526
12	286	1014	5	3.42622
13	200	222	5	4.19948
14	210	40	3	3.58510
15	224	29	3	2.72739
16	303	785	3	2.58885
17	122	387	5	4.31811
18	194	274	2	2.55058
19	291	1042	4	3.34704
20	234	1184	2	2.76020

Figure 4. Predicted movie ratings. The predictions are in the P_rating column.

EXAMPLE: RECOMMENDING FINE FOODS

In this example, you can use PROC FACTMAC to analyze fine food reviews. The Amazon Fine Foods data set, available at https://snap.stanford.edu/data/web-FineFoods.html, consists of ratings and text reviews of gourmet foods sold by Amazon (Leskovec and Krevl 2014). The data span a period of more than 10 years, including all ~500,000 reviews up to October 2012. The data include product and user information, ratings, and a plaintext review.

The input variables are product/productId, review/userId, review/profileName, review/helpfulness, review/score, review/time, review/summary and review/text. Here is an example of content in the review/text field:

I have bought several of the Vitality canned dog food products and have found them all to be of good quality. The product looks more like a stew than a processed meat and it smells better. My Labrador is finicky and she appreciates this product better than most.

Unlike the data set in the movie recommendation example, this data set include more than two nominal input variables. In this situation, the factorization machine model is equivalent to the following pairwise-interaction tensor factorization equation:

$$\begin{aligned}\hat{y}(\boldsymbol{x}) &= \hat{y}(user, product, time, helpfulness\,) \\ &= w_0 + w_{user} + w_{product} + w_{time} + w_{helpfulness} \\ &\quad +\langle \boldsymbol{v}_{user}, \boldsymbol{v}_{product}\rangle + \langle \boldsymbol{v}_{user}, \boldsymbol{v}_{time}\rangle + \cdots + \langle \boldsymbol{v}_{user}, \boldsymbol{v}_{helpfulness}\rangle\end{aligned}$$

This model considers every interaction between pairs of input variables. Although you might find this equation cumbersome, the corresponding PROC FACTMAC syntax is actually quite simple to specify. After importing the data into SAS, you can train the model by using the following code:

```
proc factmac data=mycas.amazon_foods
   nFactors=20 learnStep=0.01 maxIter=50 outModel=mycas.factors;
   input userId productId time helpfulness /level=nominal;
   target reviewScore /level=interval;
   output out=mycas.out1 copyvars=(userId productId time helpfulness);
run;
```

If you have PROC FACTMAC from the initial release of SAS Visual Data Mining and Machine Learning, you can alternatively use the following code:

```
%pairwiseFactMac(inputVarList=userId productId time helpfulness,

                 target=reviewScore,

                 dataset=amazon_foods,

                 maxIter=50,

                 nFactors=20,

                 learnStep=0.01,

                 configFile=);
```

The pairwiseFactMac macro is provided in the Appendix at the end of this paper.

The model achieves an RMSE of 0.91, which is competitive with other methods. You can visit https://github.com/sassoftware/enlighten-apply for additional code snippets and tips for incorporating the text of the reviews into the analysis. Interestingly, the Amazon Fine Food reviews are overwhelmingly positive, and coffee is by far the most popular product, well ahead of chocolate.

FACTORIZATION MACHINES FOR IMAGE RECONSTRUCTION

In image processing, it is sometimes necessary to perform reconstruction based on damaged copies of an image. You can use PROC FACTMAC for this purpose, by using the following code example:

```
proc factmac data=mycaslib.sparsePixels

   outmodel=factors

   maxiter=500

   nfactors=100

   learnstep=0.01

   seed=12345;

   input x y  /level=nominal;

   target pixelValue /level=interval;

   output out=mycaslib.FactMacScore copyvar = (x y pixelValue);

run;
```

The `sparsePixels` data table consists of three columns: x, y, and pixelValue. Each row corresponds to a nonmissing pixel. The results are shown in Figure 5. In this example, the corrupted image has 50% missing pixels.

Note that factorization machines are suitable for imputing many other types of data besides images.

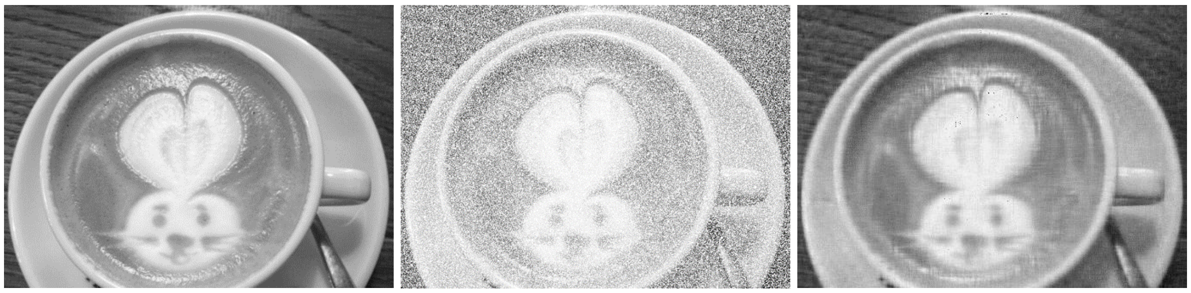


Figure 5. Image reconstruction. Left: Original image. Center: 50% missing pixels. Right: Image reconstructed using PROC FACTMAC.

FACTORIZATION MACHINES FOR PREDICTIVE MODELING IN BASKETBALL

The data for this example consist of basketball shots recorded during the 2015–2016 NBA season, from October 2015 through March 2016. The data set was downloaded using the API available from Sportradar.com. Every shot taken by every player is recorded, excluding free throws. Figure 6 shows how shot success varies by where on the court the shot was taken and whether the player is a center, forward, or guard.

The input variables that are used for the analysis are player_name, action_type, shot_zone_area, and shot_zone_range. The target variable is constructed by computing the log-odds of shot success per player.

In this example, as in the food reviews example, there are more than two nominal variables. Hence, you can perform pairwise-interaction tensor factorization.

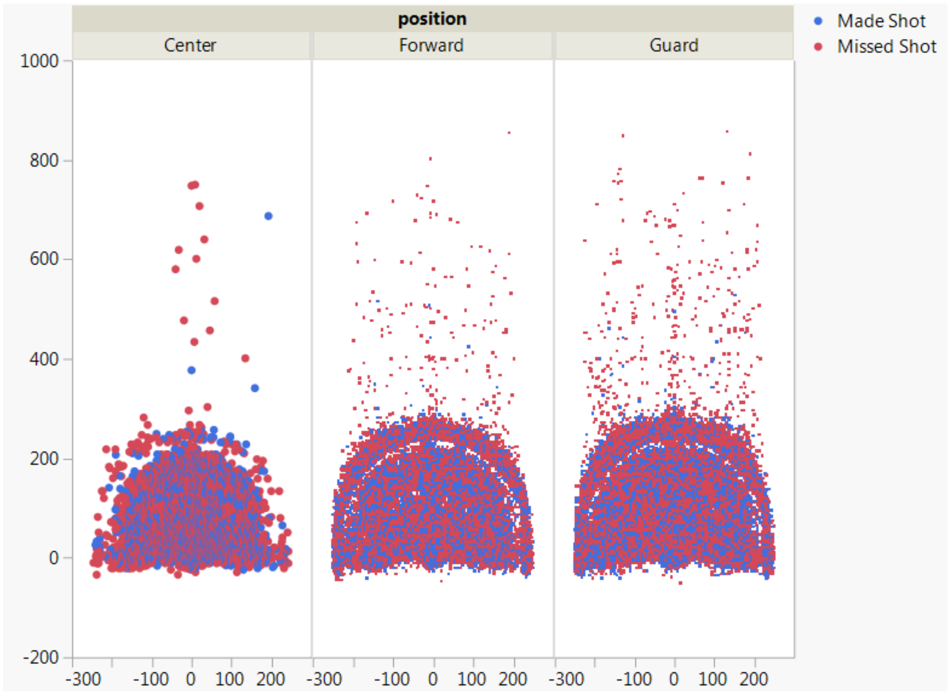


Figure 6. Shot success by court location and player position. Data from Sportradar.com.

The following code performs pairwise-interaction tensor factorization for this data set:

```
proc factmac data=mycaslib.nbaShooting_summarized
   outmodel=factors
   maxiter=50
   nfactors=10
   learnstep=0.03
   input player_name action_type shot_zone_area shot_zone_range
                /level=nominal;
   target logit /level=interval;
   output out=mycaslib.ScoreTrain copyvar=logit;
   savestate rstore=mycaslib.astore;
run;
```

You can score a held-out data set by using the following statements:

```
proc astore;
   score data = mycaslib.valid
   out=mycaslib.ScoreValid&i copyvar = (&target.)
   rstore = mycaslib.astore&i.;
run;
```

Alternatively, if PROC FACTMAC in your release of SAS Visual Data Mining and Machine Learning does not support tensor factorization, you can use the following code to perform pairwise-interaction tensor factorization and score a held-out validation data set:

```
   %pairwiseFactMac(inputVarList=player_name action_type shot_zone_area
                          shot_zone_range,
                    target=logit,
                    dataset=ray.nbaShooting_Summarized,
                    maxIter=50,
                    nFactors=10,
                    learnStep=0.03,
                    configFile=);
```

The pairwiseFactMac macro is provided in the Appendix at the end of this paper.

In addition to achieving an RMSE value of 0.93, which favorably compares to 1.39 for a support vector machine used with the same data, the factorization machine analysis reveals multiple insights. The following action types are most associated with shot success (they have the highest estimated bias values):

- running dunk shot
- running layup
- driving layup
- alley-oop dunk shot
- dunk shot
- cutting dunk shot
- putback layup
- driving dunk shot
- tip dunk shot
- driving dunk shot

As you can see, a large proportion of these action types are dunk shots. In contrast, the following action types are most associated with shot failure:

- turnaround hook shot
- turnaround jump shot
- fadeaway jump shot
- driving floating layup
- turnaround fadeaway shot
- running jump shot
- hook shot

- step-back jump shot
- pull-up jump shot
- jump shot

These action types are known to represent difficult, acrobatic shots. In addition, Figure 7 shows a visualization of the player and action factor vectors on the same plot. Because these are high-dimensional vectors, a 2D visualization is created using the t-distributed stochastic neighbor embedding (t-SNE) method (Van der Maaten and Hinton 2008). Blue points are actions, and red points are players. Similar shots appear close together, as do players who have similar shot profiles (such as Kobe Bryant and Russell Westbrook). Also, it appears from the figure that Manu Ginobili is proficient at driving floating layups, because his latent factor vector is embedded very near that of the corresponding action.

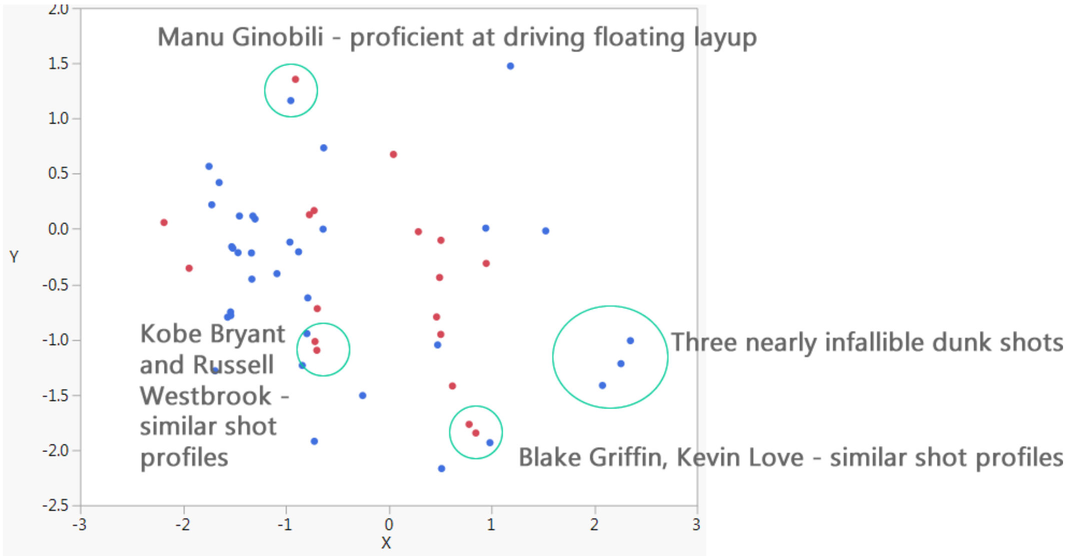


Figure 7. Visualization of player (red) and action (blue) factors. Data from Sportradar.com.

CONCLUSION

The FACTMAC procedure implements factorization machines in SAS Visual Data Mining and Machine Learning. This new model enables you to solve a variety of tasks, from recommendation to predictive modeling and image processing, all of which involve sparse data. Thanks to a highly parallel optimization solver, PROC FACTMAC can handle very large data sets. This powerful and flexible method provides not only predictions but also meaningful factor representations that can give you insights into many types of business problems.

REFERENCES

Harper, F. M., and Konstan, J. A. (2015). “The MovieLens Datasets: History and Context.” *ACM Transactions on Interactive Intelligent Systems (TiiS)* 5:19 pages. `http://dx.doi.org/10.1145/2827872.`

Leskovec, J., and Krevl, A. (2014). SNAP Datasets: Stanford Large Network Dataset Collection. `https://snap.stanford.edu/data.`

Rendle, S. (2010). “Factorization Machines.” *In Proceedings of the 10th IEEE International Conference on Data Mining (ICDM).* Piscataway, NJ: Institute of Electrical And Electronics Engineers.

Rendle, S. (2012). “Factorization Machines with libFM.” *ACM Transactions on Intelligent Systems and Technology* 3:1–22.

Sportradar AG. (2017). Sportradar.com. St. Gallen, Switzerland.

Van der Maaten, L. J. P., and Hinton, G. E. (2008). “Visualizing Data Using t-SNE.” *Journal of Machine Learning Research* 9:2579–2605.

APPENDIX

The following macro implements pairwise-interaction tensor factorization by combining multiple pairwise factorization machine models.

```
      %macro pairwisefactMac(
         inputVarList=,
         target=,
         dataset=,
         partitionFraction=.7,
         maxIter=100,
         nFactors=25,
         learnStep=0.10,
         configFile=
      );

      %let nInputs = %sysfunc(countw(&inputVarList.));
      %put nInputs = &nInputs.;
      %let k = 2; /*k=2 requests pairs*/

      %let nCombo = %sysfunc(comb(&nInputs.,&k.));
      %put nCombo = &nCombo.;

      %let listQuoted = ;

    *identify each pair inputs;
      data pairs (keep=pairs);
            length pairs $65.;
            array V{&nInputs.} $32 (

                /*quote each input*/
                        %do i = 1 %to &nInputs.;
                              %let currentVar = %scan(&inputVarList.,&i.);
                        "&currentVar."
                    %end;
            );

                do j=1 to &nCombo.;
                    call allcomb(j,&k.,of V[*]);
                    do i = 1 to &k.;
                        if i=1 then
                        do;
                                    pairs="";
                            counter=0;
                        end;
                        counter=counter+1;
                        pairs=cat(compress(pairs),' ',compress(V[i]));
                        if counter=&k. then output;
                    end;
                end;
run;

*save pairs as macro variables;
data _null_;
      set pairs end = eof;
    call symput ('pair'||strip(_n_),pairs);
run;

*call proc factmac, looping over the pairs;
libname mycaslib sasioca ;

data mycaslib.train mycaslib.valid;
      set &dataset.;
      if ranuni(0) le &partitionFraction. then output mycaslib.train;
            else output mycaslib.valid;
run;

%do i = 1 %to &nCombo.;
      proc factmac data=mycaslib.train
             maxiter=&maxIter.
            nfactors=&nFactors.
            learnstep=&learnStep.;
            input &&pair&i. /level=nominal;
            target &target. /level=interval;
            output out=mycaslib.ScoreTrain&i. copyvar = (&target.);
            savestate rstore=mycaslib.astore&i.;
      run;

      proc astore;
      score data = mycaslib.valid
      out=mycaslib.ScoreValid&i copyvar = (&target.)
            rstore = mycaslib.astore&i.;
      run;
   %end;

   data mycaslib.ScoreTrain;
      merge
      %do i = 1 %to &nCombo.;
            mycaslib.ScoreTrain&i. (rename=(p_&target. =
p_&target._&i.))
      %end;
     ;
      _partInd_ = 1;
   run;

   data mycaslib.ScoreValid;
      merge
      %do i = 1 %to &nCombo.;
            mycaslib.ScoreValid&i. (rename=(p_&target. =
p_&target._&i.))
      %end;
     ;

      _partInd_ = 0;
   run;

   data mycaslib.ScoreCombined;
      set mycaslib.ScoreTrain
```

```
            mycaslib.ScoreValid
        ;
      run;

   *build a regression model to predict target using predicted values for all
pairs;
   proc regselect  data = mycaslib.scoreCombined;
      model &target.=
       %do i=1 %to &nCombo.;
                  p_&target._&i.
         %end;
      ;
      partition rolevar=_partInd_ (TRAIN="1" VALIDATE="0");
   run;
   quit;
%mend pairwiseFactMac;
```

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Jorge Silva
jorge.silva@sas.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.