Matrix Factorization Techniques for Data Mining

Proposal for Google Summer of Code 2011
Orange – Data Mining Fruitful & Fun

April 7, 2011

Table 1: Personal Data. More information is available in CV (See section Biography and Experience).

Name and Surname	Marinka Zitnik
E-mail Address	marinka@zitnik.si
Blog	http://marinka.zitnik.si, http://helikoid.si
Skype	marinka.zitnik
Education	Faculty of Computer and Information Science, University of Ljubljan, Slovenia B.S., Interdisciplinary University Program Computer Science and Mathematics Attending third year of the four-year interdisciplinary university program Average grade: 10.0
General Interests	Networking, information technology, information systems, computational theory, approximation algorithms, optimization, numerical methods, linear algebra, calculus, data mining, machine learning, artificial intelligence, theory of fractals and chaos, advances in science and technology in general
Age	21 (born on 23^{rd} April 1989)
Languages	Actively speak and write Slovenian, English, German

1 Proposal

Title Matrix Factorization Techniques for Data Mining

Description Matrix factorization is a fundamental building block for many of current data mining approaches and factorization techniques are widely used in applications of data mining. Our objective is to provide the Orange community with a unified and efficient interface to matrix factorization algorithms and methods. For that purpose we will develop a scripting library which will include a number of published factorization algorithms and initialization methods and will facilitate the combination of these to produce new strategies. Extensive documentation with working examples that will demonstrate real applications, commonly used benchmark data and visualization methods will be

provided to help with the interpretation and comprehension of the results. Main factorization techniques and their variations planned to be included in the library are: Bayesian decomposition (BD) together with linearly constrained and variational BD using Gibbs sampling, probabilistic matrix factorization (PMF), Bayesian factor regression modeling (BFRM), family of nonnegative matrix factorizations (NMF) including sparse NMF, non-smooth NMF, local factorization with Fisher NMF, least-squares NMF. Different multiplicative and update algorithms for NMF will be analyzed which minimize LS error or generalized KL divergence. Further nonnegative matrix approximations (NNMA) with extensions will be implemented. For completeness algorithms such as NCA, ICA and PCA could be added to the library.

1.1 Proposal Timeline

Before April 20

- To familiarize myself completely with Orange architecture and its functionality.
- Study of the factorization techniques and their extensions.

April 20 – May 23 (Before the official coding time)

- To do some self coding to improve my further understanding of techniques.
- I will become absolutely clear about my future implementations, design and approaches I will follow.

May 23 – June 18 (Official coding period starts)

- Implementing family of NMF techniques; major techniques are: NMF, sNMF, nsNMF, lNMF, lsNMF, NNMA.

June 18 – July 5

- Extend NMF implementations with various extensions, additive and multiplicative update rules and initialization methods.
- Implement Bayesian methods. Bayesian decomposition using Gibbs sampler, MCMC techniques such Bayesian factor regression modeling.

July 5 – July 15

- Extend Bayesian methods (possibly variational BD, linearly constraned BD).
- Adapt PMF model to the interval-valued matrices and implement Interval-valued PMF (I-PMF) and Interval-valued NMF (I-NMF).
- Handling PMF on large, sparse and unbalanced datasets (algorithm for probabilistic sparse matrix factorization (PSMF)).

July 15th mid term evaluation

July 15 – July 25

- Be in constant touch with the Orange team to let them know about progress on the project.
- Improve efficieny of the code, bug removal, exception handling, testing.
- Devise working examples that demonstrate various types of applications.

July 24 – July 31

- For Documentation.

A buffer of two weeks has been kept for any unpredictable delay.

1.1.1 Availability

Until of mid June I will have some obligations concerning study at the Faculty of Computer and Information Science. Therefore I will spend approximately 8 to 10 hours working on the project per week. During the summer (from mid June until the program deadline at the end of August) I intend to be available each working week, 8 hours per day.

Flexibility is possible to assure the appropriate progress of the project and meeting up with the goals according to the timeline.

1.2 Proposal Objectives

1.2.1 Library Features

The `MF` library:

- 14 basic algorithms:
  - `l-BD, v-BD, BFRM, BD, PSMF, i-PMF, i-NMF, NNMA, PMF, ls-NMF, l-NMF, ns-NMF, s-NMF, NMF,`
- initialization methods,
- extensions of basic algorithms:
  - `Chib` method for estimating factorization rank for a given dataset in NMF when it is not evident from the nature of data,
  - Factorization quality measures:
    - measure based on the cophenetic correlation coefficient to assess the stability of clustering associated with a given factorization rank,
    - sparseness,
    - dispersion,
    - residuals,
  - `ICM` iterated conditional modes algorithm for NMF,
  - `vBD, lBD` variational and linearly constrained BD,
  - multiplicative and additive update rules for NMF,
  - `I-NMF, I-PMF,` adapt PMF and NMF models to interval-valued matrices,
- documentation,
- working examples that demonstrate various types of applications,
- support for plotting utility functions to interpret and visualize results,
- single interface to perform all algorithms and combine them with initialization methods and extensions.

1.2.2 Algorithms

```
# List factorizations following Bayesian approach
MF.bayes_factorization()
        "bd" "bfrm" "v-bd" "l-bd"
# List NMF factorizations
MF.nmf_factorization()
        "nmf" "s-nmf" "ns-nmf" "l-nmf" "ls-nmf" "pmf" "nnma"
        "i-nmf" "i-pmf" "psmf"
```

Table 2 gives short description of algorithms.

Table 2: Algorithms for matrix factorization with names, used to specify algorithm to execute.

`nmf`	Standard NMF based on Kullbach Leibler divergence, simple multiplicative updates, enhanced to avoid numerical overflow [2]. **Reference: [3]**
`s-nmf`	Sparse NMF based on alternating non-negativity constrained least squares, solved by a fast non-negativity constrained least squares. Sparseness imposed on the left, right factor. **Reference: [4]**
`ns-nmf`	Non-smooth NMF. Uses a modifed version of Lee and Seung's multiplicative updates for Kullbach-Leibler divergence. It is meant to give sparser results. **Reference: [5]**
`l-nmf`	Local Fisher NMF. Add the Fisher constraint to maximize the between-class scatter and minimize the within-class scatter. **Reference: [6]**
`ls-nmf`	Alternating Least Square NMF. It is meant to be very fast compared to other approaches. **Reference: [4]**
`pmf`	Probabilistic MF. Model which scales linearly with the number of observations and performs well on large, sparse, imbalanced datasets. **Reference: [7]**
`nnma`	Nonnegative matrix approximation. Method for dimensionality reduction with respect on the nonnegativity of input data. Multiplicative iterative scheme. **Reference: [8]**
`i-nmf`	Interval-valued NMF. **Reference: [9]**
`i-pmf`	Interval-valued PMF. **Reference: [9]**
`psmf`	Probabilistic Sparse MF. PSMF allows for varying levels of sensor noise, uncertainty in the hidden prototypes used to explain the data and uncertainty as to the prototypes selected to explain each data vector. **Reference: [10]**
`bd`	Bayesian decomposition. A Bayesian treatment of NMF, based on a normal likelihood and exponential priors, Gibbs sampler to approximate the posterior density. **Reference: [11]**
`bfrm`	Bayesian factor regression model. Markov chain Monte Carlo technique. **Reference: [12]**
`v-bd`	Variational Bayesian decomposition. **Reference: [13]**
`l-bd`	Linearly constrained Bayesian decomposition. **Reference: [13]**

1.2.3 Initialization

Success of the applications of some factorization techniques depends on the appropriate initialization of the basis and mixture matrices. `MF` library includes four basic initialization methods, where `none` means the user completely specifies the initial factorization by passing values for basis and mixture matrix. Custom user-defined initialization method can be specified.

```
# List initialization methods
MF.factorization_init()
        "none" "ica" "random" "nndsvd" "<custom␣init>"
```

If not specified in the call, it is assumed the used factorization model is the standard one, $X \approx WH$. However, some algorihtms have different underlaying model, such as introducing an extra matrix factor and changing the way the target matrix X is approximated. User can specify the factorization model to use in performing run through argument `model`.

```
# List available factorization models
MF.models()
        "std-model" "ns-model" "offset-model"
```

1.2.4 Performing run

Unified interface for factorization algorithms is provided.

```
# Performing run of the algorithm
res = MF.factorization(X, method, init, rank, model, nrun,
        callback, track, option, ...)
```

Argument names and explanation

`X` Target matrix to estimate. It is possibly `Orange.ExampleTable`.

`method` Name of the algorithm as specified in Table 2. `NMF` is default.

`init` Name of the initialization method or name of the custom initialization function. `random` is default. By specifying `init` to `none`, user can pass values for basis and mixture matrices through `model` and factorization is fixed.

`rank` Factorization rank.

`model` Factorization model to use. `st-model` is default.

```
#  Example of argument value when fixed factorization
model = Model.set_model(W = matrix(0.1, 40, 2),
        H = matrix(0.05, 40, 0.2))
```

`nrun` Specified to perform multiple runs of the algorithm. Specify the number of runs. Useful argument for stability with some initialization methods. When using multiple runs by default only best fitted model is returned in the `res` object. User can pass a callback function that is called after each run.

`callback` Callback function used only when `nrun > 1`. Saving summary measures or process the result of the current fitted model before it is discarded.

`track` When `true`, it enables error tracking. The returned object's `res` attribute `residuals` contains the trajectory of the objective functions/values.

`option` Options depending on the value of `method`. Some options are listed in Table 3.

Returned value
Result object `res` holds both details about run and factorization model. Details about run include specified algorithm name, initialization method name, number of iterations, distance metric, residuals, number of samples, number of features, rank. The result of a run is an object of class `Model`. This class handles in a generic way the results of the factorization algorithms.

Attributes (not comprehensive)

`residuals` Vector that contains the final residuals or the residuals track between the target matrix and its estimate(s) if tracking enabled.

`method` Contains the name of the algorithm used with factorization.

`init` Contains the name of the initialization method or user specified method.

`distance` Contains the distance function that measures the residual between the target matrix and its estimate (model).

`nrun` Stores the number of runs performed.

`parameters` Contains a dictionary of extra parameters specific to the algorithm used to estimate the target matrix.

Methods (not comprehensive)

`basis()` Extracts the matrix of basis vectors from the fitted model.

`mixture()` Extracts the matrix of mixture coefficents from the fitted model.

`loss()` Returns the value of the loss function given target matrix and fitted model.

`fitted()` Computes the estimated target matrix according to the fitted model.

`summary()` Returns the dictionary that contains the summary of the fitted model and computation time. Return value can be specific to the factorization algorithm.

`plot_residuals(filename)` Plots the residuals track of the run if tracking of residuals enabled.

Here are some utility functions that provide additional information on execution and model.

```
# Retrieve target matrix X
x = res.target()
# Retrieve basis matrix W
w = res.basis()
# Retrieve mixture matrix H
h = res.mixture()
# Retrieve quality measures about factorization
s = res.summary()
```

Table 3: Incomplete list of options.

`nrun`	Number of runs ot the algorithm.
`theta`	Smoothing parameter at `ns-nmf`, prior for `sigma` at `nnma`.
`alpha`	Prior for W ($X \approx WH$) at `nnma`.
`beta`	Prior for H ($X \approx WH$) at `nnma`.
`sigma`	Initial value for noise variance at `nnma`.
`distance`	Distance function.

1.3 Visualization

1.3.1 Error track

If error tracking is enabled on factorization computation, then the trajectory of the objective value can be plotted with the function `plot_residuals(filename)`. Example is shown on Figure 1.

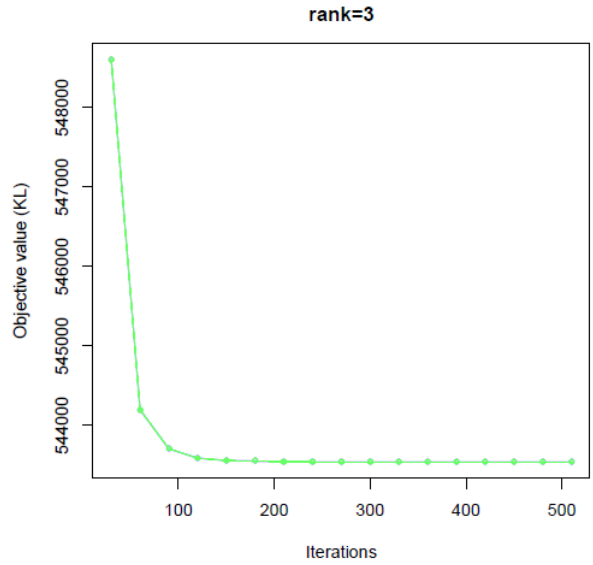

Figure 1: Example of the trajectory of the objective value (Kullbach Leibler divergence).

1.3.2 Heatmaps

Heatmaps would provide an easy way to visualize the resulting metaprofiles and metagenes. When plotting metagenes matrix, filter could be used to select only the genes that are specific to each metagene with the selection method as described in [4]. If multiple runs of the algorithm are performed, consensus matrix can be plotted - the average connectivity matrix across the runs (the number of runs could be selected by continuing until the connectivity matrix appears to stabilise).

Although visual inspection of the reordered connectivity matrix can provide substantial insight, it is important to have quantitative measure of stability for each value of factorization rank. Measure based on cophenetic correlation coefficient indicates the dispersion of the consensus matrix.

Clarification of names: Defining the factorization model as $X \approx WH$, different names are assigned to the columns of matrices W and H in the literature depending on the context of application.

columns of W *basis vector, metagenes, factors, source, image basis,*

columns of H *mixture coefficients, metagenes expression profiles, weights.*

1.4 Widget Sketch

Proposed widget sketches are on Figure 2 and Figure 3.

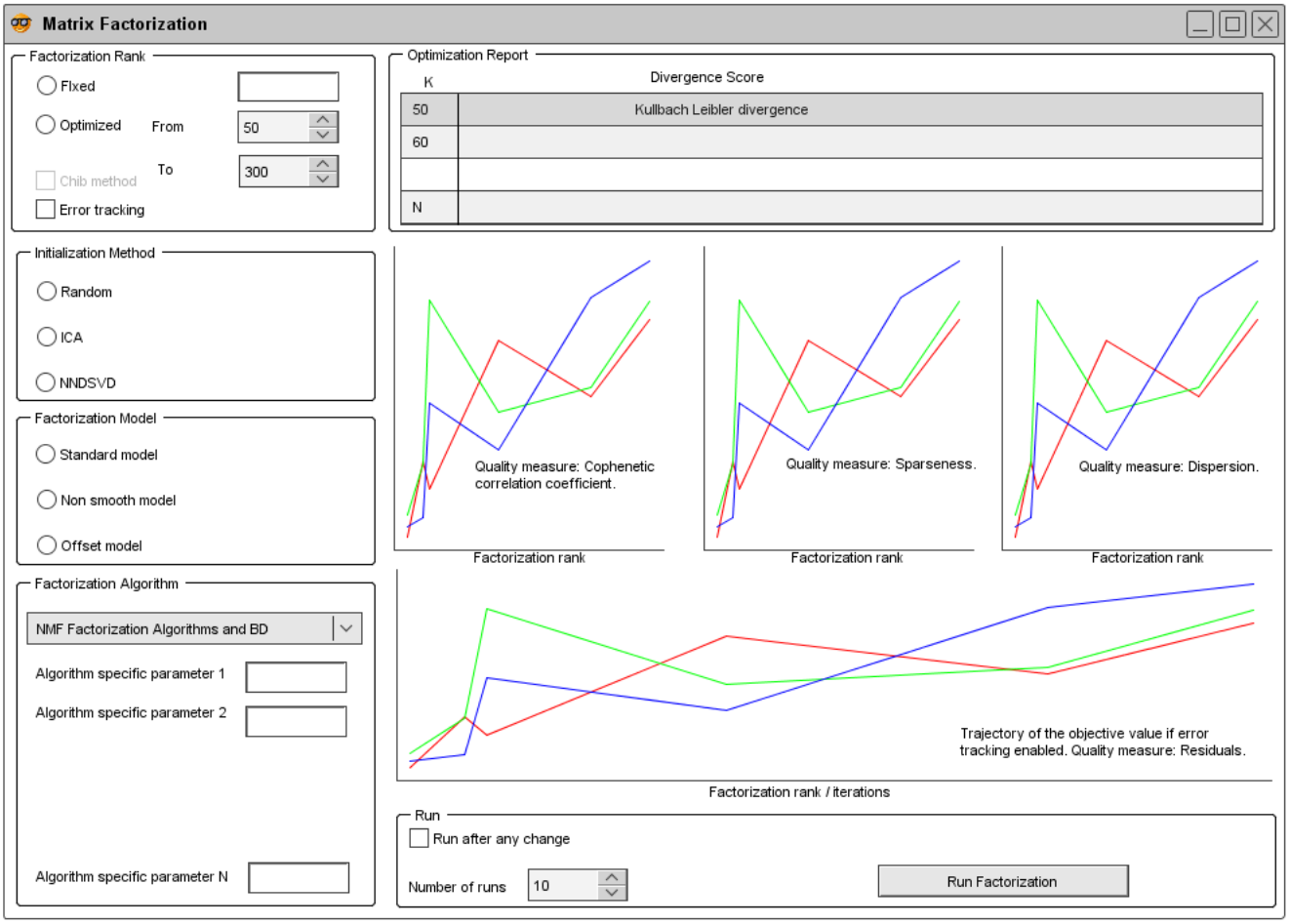

Figure 2: **Widget 1: Matrix Factorization** sketch.

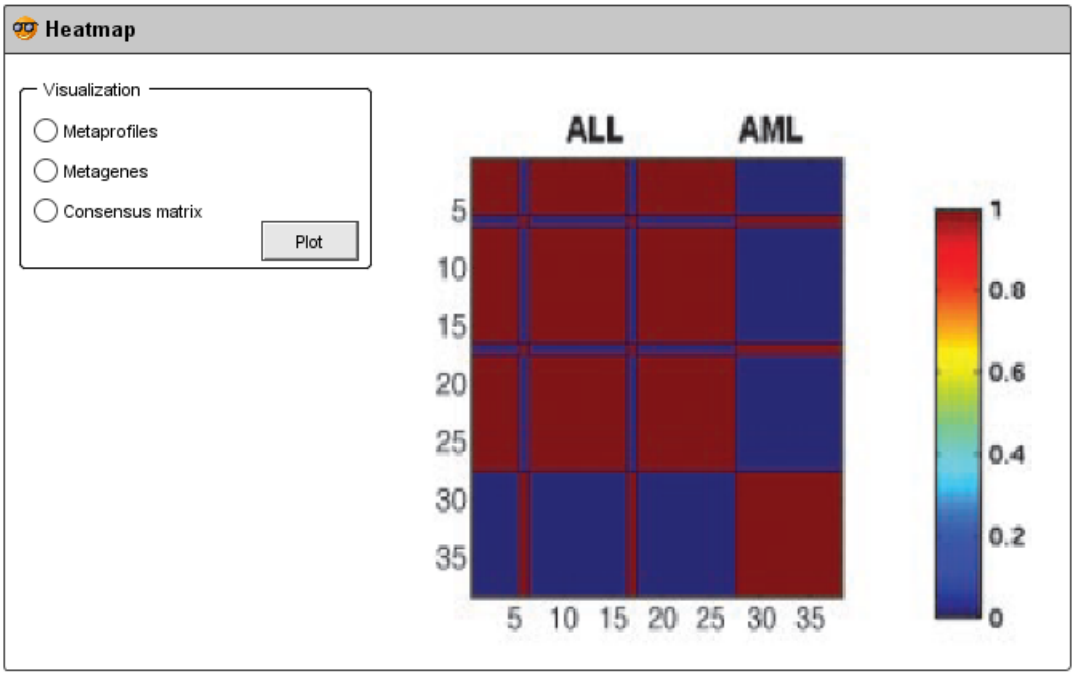

Figure 3: **Widget 2: Heatmap Visualization** sketch.

1.5 Research in MF field after GSoC

Upon need and request I would be glad to provide in the future additional algorithm implementations to the `MF` library. I am interested in doing some research in this field as well, as I find the content of the proposal of high applicative and theoretic value.

1.6 Broader Impact of the Proposed Work

Matrix factorization methods have been shown to be a useful decomposition for multivariate data as low dimensional data representations are cruical to numerous applications in statistics, signal processing and machine learning. With the library of factorization methods in Orange we will provide an easy-to-use interface to established algorithms and expand already vast usage of Orange methods in numerous applications without the need to use external programming packages for factorizations.

Incomplete list of some applications.

Environmetrics and chemometrics Ideas of PMF have been applied to environmental data to analyze atmospheric emission and pollution.

Image procesing and computer graphics NNMA algorithms have been used to obtain a parts based representation for image data.

Text analysis Clustering experiments with NNMA have suggested higher accuracy and ability to tackle semantic issues such as synonymy. Distinguish different meanings of words used in different contexts (polysemy).

Bioinformatics Improving molecular cancer class discovery through sparse NMF. Modeling cross-hybridization in microarray experiments. Estimating transcription factor activities in regulatory networks. Multi-way clustering of microarray data. Comparing NMF with HC, NMF can reveal hierarchical structure when it exists but does not force such structure on the data like HC does. In addition, agglomerative techniques sometimes struggle to properly merge clusters with many samples but NMF may have an advantage in exposing meaningful global hierarchy. Using NMF to reduce the dimensionality of expression data from thousands of genes to a handful of metagenes [3] and approximate the gene expression pattern of samples as positive linear combinations of these metagenes. Identification of distinct molecular patterns. Recognition of sequence patterns among related proteins.

Miscellaneous Usage of convolutional NNMA to extract speech features. Transcription of poypohnic music passages via NNMA. Other applications include: object characterization, spectral data analysis, learning sound dictionaries, mining ratio-rules, multiway clustering.

2 Biography and Experience

I have enclosed my CV with additional details about my experience, skills and contributions. It is available at the address *http://helikoid.si/GSoC/marinka_zitnik_CV.pdf*. This document is available at the address *http://helikoid.si/GSoC/orange_proposal.pdf*.

Backup address: *http://marinka.zitnik.si/GSoC/marinka_zitnik_CV.pdf*. This document is available at the address *http://marinka.zitnik.si/GSoC/orange_proposal.pdf*. Any additional information is available upon request.

References

[1] Kim H, Park H. Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis. Bioinformatics (Oxford, England) 2007, 23:1495–502.

[2] Lee, D. D. and Seung, H. S. Algorithms for non-negative matrix factorization. NIPS,2000, 556–562.

[3] Brunet, J. P., Tamayo, P., Golub, T. R., and Mesirov, J. P. Metagenes and molecular pattern discovery using matrix factorization. Proc Natl Acad Sci U S A, 2004, 101(12), 4164–4169.

[4] Kim, H., Park, H. Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis. Bioinformatics (Oxford, England), 2007, 23(12), 1495-502.

[5] Pascual-Montano, A., Carazo, J. M., Kochi, K., Lehmann, D., and Pascual-Marqui, R. D. Nonsmooth nonnegative matrix factorization (nsnmf). IEEE transactions on pattern analysis and machine intelligence, 2006 28(3), 403–415.

[6] Wang, Y., Turk, M. Fisher non-negative matrix factorization for learning local features.

[7] Salakhutdinov, R., Mnih, A. Probabilistic Matrix Factorization. Learning.

[8] Sra, S.,Dhillon, I. S. Nonnegative Matrix Approximation : Algorithms and Applications. Sciences-New York, 2006, 1–36.

[9] Zhiyong Shen, Liang Du, Xukun Shen, Yidong Shen, Interval-valued Matrix Factorization with Applications, Data Mining, IEEE International Conference on, pp. 1037–1042, 2010 IEEE International Conference on Data Mining, 2010.

[10] Dueck, D., Morris, Q. D., Frey, B. J. Multi-way clustering of microarray data using probabilistic sparse matrix factorization. Bioinformatics (Oxford, England), 21 Suppl 1, 2005, 144-51.

[11] Schmidt, M. N., Winther, O., Kai Hansen, L. Bayesian non-negative matrix factorization.

[12] Schmidt, M. N. Bayesian non-negative matrix factorization. Bayesian Statistics 7 (Oxford), 2003.

[13] Ochs, M. F., Kossenkov A. V. NIH Public Access. Methods, Methods Enzymol., 2009, 467: 59–77.