dmt:
probabilistic dependency modeling toolkit

Leo Lahti[1,2]*and Olli-Pekka Huovilainen[1]
(1) Dpt. Information and Computer Science, Aalto University, Finland
(2) Dpt. Veterinary Bioscience, University of Helsinki, Finland

August 11, 2012

1 Introduction

This package provides general tools for the discovery and analysis of statistical dependencies between co-occurring measurement data [5], including well-established models such as probabilistic canonical correlation analysis and its regularized variants [1, 2, 4, 9]. Probabilistic framework deals rigorously with the uncertainties associated with small sample sizes, and allows incorporation of prior information in the analysis through Bayesian priors [6]. The applicability of the models has been demonstrated in case studies [4, 9].

Dependency models help to discover regularities and interactions that are not seen in individual data sets. Multiple, complementary views of the same objects are available in many fields including computational biology, economics, linguistics, neuroinformatics, open data initiatives, social sciences, and other domains. Demand for methods to analyze such data is increasing with the availability of co-occurring observations. Open access implementations of the algorithmic solutions help to realize the full potential of these information sources. Your feedback and contributions are welcome.[1]

1.1 Installation

Install *dmt* from within R with 'install.packages("dmt")'. This will fetch and install the package automatically. Source code of the latest stable release is available at CRAN[2]. Source code of the development version is available at R-forge[3].

*leo.lahti@iki.fi
[1]See the project page at R-Forge: http://dmt.r-forge.r-project.org/
[2]http://cran.r-project.org/web/packages/dmt/index.html
[3]http://dmt.r-forge.r-project.org/

2 Probabilistic dependency modeling framework

The *fit.dependency.model* function implements the probabilistic dependency modeling framework presented in [2] and subsequent extensions [1, 3, 4]. The latent variable model assumes that the two data sets, X and Y, can be decomposed in *shared* and *data set-specific* components (Figure 1). We provide tools to discover these components.

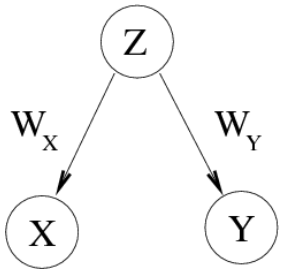

Figure 1: Graphical description of the shared latent variable model with observations X, Y and a shared latent variable Z.

The shared signal between two (multivariate) observations $\mathbf{x}, \mathbf{y}$ is modeled with a shared latent variable $\mathbf{z}$. The shared component can have different manifestation in each data set, described by the linear transformations W_x and W_y. A standard Gaussian model for the shared latent variable $\mathbf{z} \sim N(0, I)$ and data set-specific effects gives the following model:

$$\begin{aligned} X &\sim W_x \mathbf{z} + \varepsilon_x \\ Y &\sim W_y \mathbf{z} + \varepsilon_y \end{aligned} \tag{1}$$

The data set-specific effects are modelled by multivariate Gaussians $\varepsilon_\cdot \sim \mathcal{N}(0, \Psi_\cdot)$ with covariances Ψ_x, Ψ_y, respectively.

Particular constraints on the model structure can be set through prior parameters in the fit.dependency.model function. For instance, it is possible to tune the dimensionality of the latent variable, require non-negativity for W, or to set particular structure on the marginal covariances. An overview is provided below.

2.1 Dependency detection algorithm

The modeling framework described in Section˜2 is implemented in the fit.dependency.model function. To use the default method to detect dependencies between two data sets, X and Y, try:

```
> library(dmt)
> data(modelData) # load example data X, Y
> model <- fit.dependency.model(X, Y)
```

2.2 Regularized dependency detection

Various options are available to tune the model structure and to guide dependency modeling through Bayesian priors implemented in the 'priors' option in the fit.dependency.model function [4]. For

instance, the following will fit a model with $W_x = W_y$, with non-negative (but otherwise unconstrained) W_x, W_y and with full marginal covariances (Ψ_x, Ψ_y) for the dataset-specific effects.

```
> model <- fit.dependency.model(X, Y,
+                                       priors = list(Nm.wx.wy.sigma = 0, Nm.wx.wy.mean = 1, W = 1
+                                  marginalCovariances = "full")
```

2.2.1 Available regularization options

Here is a brief summary of the available options. See help(fit.dependency.model) for further options and examples.

zDimension Dimensionality of the latent variable, which is used to characterize the latent effects. By default, full dimensionality is used, but in many applications the relevant dependencies can be described with lower-dimensional representation of the shared effects.

W By default, no constraints are applied on W_x and W_y. However, non-negative solutions can be obtained by setting an exponential prior with rate parameter W: $W_x \sim exp(-WW_x(i))$ for each element i of the matrix W_x, and respectively for W_y. Small values of the rate parameter enforce non-negativity but are otherwise non-informative.

marginalCovariances Dataset-specific effects can come from Gaussian distribution with either full, diagonal, isotropic, or identical isotropic covariance structure. The last option refers to a model with $\Psi_x = \Psi_y$.

matched If matched = TRUE, it is possible to tune the relationship between W_x and W_y. See Nm.wx.wy.sigma.

Nm.wx.wy.sigma Assuming that $W_y = TW_x$, it is possible to tune the relationship between W_x and W_y through a prior on T. This can be useful for guiding the modeling to focus on certain types of dependencies, and to avoid overfitting. Here, a matrix normal distribution is applied: $T \sim N_m(H, \sigma I, \sigma I)$, with mean and covariance H = Nm.wx.wy.mean, $\sigma I = Nm.wx.wy.sigmaI$, respectively. By default, Nm.wx.wy.mean = I. The prior can be tuned through σ. When $\sigma = 0$, $W_x = W_y$; when $\sigma \to \infty$, the relationship between W_x and W_y is not constrained.

2.3 Special cases

Special cases of the model include probabilistic versions of canonical correlation analysis, factor analysis, and principal component analysis, and their regularized variants.

Probabilistic CCA (pCCA) assumes full covariance matrices Ψ_x, Ψ_y, and arbitrary linear transformations W_x, W_y that maximize the likelihood of the model in Eq.˜1. Full marginal covariances provide the most detailed model for the data set specific effects. The connection of this latent variable model and the traditional canonical correlation analysis has been established in [2].

Probabilistic SimCCA (pSimCCA) is equivalent to pCCA, but has the additional constraint $W_x = W_y$. This less flexible model is more robust against overfitting when the data is scarce [4].

Probabilistic factor analysis (pFA) is equivalent to pCCA, except diagonal covariances Ψ_x, Ψ_y. The simplified structure regularizes the solution and can potentially reduce overfitting in applications. The model is identical to concatenating X, Y, and fitting ordinary probabilistic factor analysis on the concatenated data set. In addition, a special case is implemented where each

covariance matrix $\Psi_\cdot$ is isotropic (but not necessarily identical). The standard probabilistic factor analysis model for a single data set is also available [7].

Probabilistic PCA (pPCA) is obtained with identical isotropic covariances $\Psi_x = \Psi_y = \sigma I$. This model is identical to concatenating X, Y, and fitting ordinary probabilistic PCA on the concatenated data. The standard probabilistic PCA model for a single data set is also available [8].

2.4 Interpreting the results

The *fit.dependency.model* function provides an object of DependencyModel class. To retrieve options and model parameters corresponding to the model in Equation 1, investigate the output as follows:

```
> W <- getW(model)      # model parameters Wx, Wy
> psi <- getPhi(model)  # model parameters Psix, Psiy
> Z <- getZ(model)      # ML-estimate of the shared latent variable
```

For a full list of model output, including the original inputs for function call, check:

```
> slotNames(model)

[1] "W"      "phi"    "score"  "method" "params" "data"   "z"
```

For detailed explanation of the output, see help(fit.dependency.model).

A comprehensive set of examples with toy data is available online.[4]

2.5 Parameter estimation

Model parameters are estimated with EM algorithm. Conventional optimization methods are used when no analytical solution exists for particular variables.

3 Dependency-based dimensionality reduction

The drCCA algorithm [9] provides tools for dimensionality reduction and data fusion that retains the variation shared between the original data sources, while reducing data set-specific effects based on a linear projections. The algorithms combine data sets with co-occurring samples into a common representation of low dimensionality based on linear transformations through generalized CCA. This helps to discover dependencies between multiple data sets (two or more) simultaneously. Regularization options and automated tools are available to select the final dimensionality of the combined data set.

The following example shows how to perform dependency-based dimension reduction on two data sets (samples x features matrices).

```
> data(expdata1)
> data(expdata2)
> drcca <- drCCAcombine(list(expdata1, expdata2)) # data fusion
```

[4] http://dmt.r-forge.r-project.org/tests/complete/)

```
[1] "Number of input matrices"
[1] 2
[1] "Number of test and training matrices created"
[1] 3
[1] "Normal gCCA"
[1] 2000   12
```

```
> r <- regCCA(list(expdata1,expdata2))            # regularized CCA
```

```
[1] "Normal gCCA"
```

```
> shared <- sharedVar(list(expdata1,expdata2),r,4)          # shared effects
> #specific <- specificVar(list(expdata1,expdata2),r,4)        # data set-specific effects
> #tmp <- plotVar(list(expdata1,expdata2),r,c(1:2),4)     # visualization
```

Linear projections are identified for each individual data set, and a combined representation of the dependent components is constructed on a lower-dimensional space. See the original publication for further details [9]:

Acknowledgements

Particular thanks go to contributors Arto Klami and Abhishek Tripathi regarding the drCCA functionality.

Details

- *Licensing terms:* the package is licensed under FreeBSD open software license.
- *Citing DMT:* When using the package, please cite [5, 6]; for particular algorithms (drCCA, fit.dependency.model, etc.), see also additional citation information in the documentation.

This document was written using:

```
> sessionInfo()
```

```
R version 2.15.1 Patched (2012-08-09 r60217)
Platform: x86_64-unknown-linux-gnu (64-bit)

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=C
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=C                 LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
```

```
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] dmt_0.8.11       MASS_7.3-20      Matrix_1.0-7     lattice_0.20-6
[5] mvtnorm_0.9-9992

loaded via a namespace (and not attached):
[1] grid_2.15.1  tools_2.15.1
```

References

[1] C. Archambeau, N, Delannay, and M. Verleysen. Robust probabilistic projections. In W.W. Cohen and A.˜Moore, editors, *Proceedings of the 23rd International conference on machine learning*, pages 33–40. ACM, 2006.

[2] F.R. Bach and M.I. Jordan. A probabilistic interpretation of canonical correlation analysis. Technical Report 688, Department of Statistics, University of California, Berkeley, 2005.

[3] A. Klami and S. Kaski. Probabilistic approach to detecting dependencies between data sets. *Neurocomputing*, 72(1-3):39–46, 2008.

[4] L. Lahti, S. Myllykangas, S. Knuutila, and S. Kaski. Dependency detection with similarity constraints. In *Proc. MLSP'09 IEEE International Workshop on Machine Learning for Signal Processing*, IEEE, Piscataway, NJ, 2009.

[5] L. Lahti *et al.* Dependency modeling toolkit. International Conference on Machine Learning (ICML-2010). Workshop on Machine Learning Open Source Software. Haifa, Israel, 2010. Project url: http://dmt.r-forge.r-project.org

[6] L. Lahti. Probabilistic analysis of the human transcriptome with side information. PhD thesis. Aalto University School of Science and Technology, Department of information and Computer Science, Espoo, Finland, 2010. http://lib.tkk.fi/Diss/2010/isbn9789526033686/

[7] D.B. Rubin and D.T. Thayer. EM algorithms for ML factor analysis. Psychometrika 47(1):69–76, 1982. url: http://www.springerlink.com/content/f7x37l8656877311/

[8] M,E. Tipping and C.M. Bishop. Probabilistic principal component analysis. Journal of Royal Statistical Society B 61(3):611–622, 1999.

[9] A. Klami, A. Tripathi and S. Kaski. Simple integrative preprocessing preserves what is shared in data sources. *BMC Bioinformatics*, 9(111), 2008.