

# netReg

Simon Dirmeyer

2017-05-05

## Introduction

Modelling biological associations or dependencies using linear regression models is often complicated when the analysed data-sets are high-dimensional and less observations than variables are available ( $n \ll p$ ). For these scenarios methods utilizing a priori knowledge, e.g. in the form of biological networks, have been proposed, arguing that this information might provide better estimates for regression coefficients. Recently several network-based regularization techniques have been proposed (C. Li and Li 2008, Kim, Pan, and Shen (2013), Cheng et al. (2014)).

**netReg** provides a highly-efficient implementation of these graph-penalized regression model. The models introduce a priori generated biological graph information into generalized linear models yielding sparse or smooth solutions for regression coefficients.

**netReg** computes coefficients using *cyclic coordinate descent* as previously introduced (Fu 1998, Friedman et al. (2007)), (Friedman, Hastie, and Tibshirani 2010). The package is an R-wrapper to an external C++ library that uses **RcppArmadillo** (Eddelbuettel and Sanderson 2014) for fast matrix calculations and **dlib** (King 2009) for gradient-free convex optimization for model selection.

## Edgenet tutorial

This section explains how to fit a linear model and do parameter estimation using **edgenet**-regularization. The model is a truncated version from (Cheng et al. 2014) that is able to introduce prior graphs for the design and response matrices for penalization.

At first we generate some toy data randomly:

```
set.seed(23)
X <- matrix(rnorm(1000*5), 1000)
Y <- matrix(rnorm(1000*5), 1000)
```

Then we load the **netReg** library:

```
library(netReg)
```

For **edgenet** we need to create an affinity matrix for the co-variables first. We also can create a graph for the responses, but this is not necessary to demonstrate the method. We could create a random graph like this:

```
aff.mat <- matrix(rbeta(25, 1, 5), 5)
aff.mat <- (t(aff.mat) + aff.mat) / 2
diag(aff.mat) <- 0
```

We created the affinity matrix absolutely random, although in practice a *real* (biological) observed affinity matrix should be used, because in the end the affinity matrix decides the shrinkage of the coefficients.

## Model fitting

Fitting a model using edge-based regularization with **netReg** is easy:

```
fit <- edgnet(X=X, Y=Y, G.X=aff.mat, lambda=1, psigx=1, family="gaussian")
print(fit)
```

```
##
## Call: edgnet.default(X = X, Y = Y, G.X = aff.mat, lambda = 1, psigx = 1,
##      family = "gaussian")
##
## Coefficients:
##           [,1]           [,2]           [,3]           [,4]           [,5]
## [1,]  0.01497323  0.015657011  0.0006453391  0.04699207 -0.009302215
## [2,] -0.01076841  0.069246936 -0.0876302568 -0.00405287 -0.007441449
## [3,] -0.02396619 -0.006507272  0.0244506149  0.02685324 -0.014316626
## [4,]  0.00000000 -0.039063322 -0.0301731092 -0.02050031  0.001355670
## [5,]  0.04213578  0.021257125  0.0242694757 -0.02222326 -0.008547463
## Intercept:
##           [,1]
## [1,]  0.019672657
## [2,] -0.047832210
## [3,] -0.004606925
## [4,] -0.025844237
## [5,]  0.032155794
## Parameters:
## lambda psi_gx psi_gy
##      1      1      0
## Family:
## [1] "gaussian"
```

In this case we used a single affinity matrix  $G.X$  which represents the relationship of the covariables  $X$ . If the design matrix has  $p$  variables,  $G.X$  has to be an  $(p \times p)$ -dimensional symmetric matrix. We can also include a matrix for the response matrix  $Y$  with  $q$  dependent variables. In that case the affinity matrix  $G.Y$  has to be  $(q \times q)$ -dimensional (and also symmetric).

The `fit` object contains information about coefficients, intercepts, residuals, etc. Having the coefficients estimated we are able to predict novel data-sets:

```
X.new <- matrix(rnorm(10*5),10)
pred <- predict(fit, X.new)
```

The `pred` objects contains the predicted values for the responses.

## Model selection

In most cases we do not have the optimal shrinkage parameters  $\lambda$ ,  $\psi_{gx}$  and  $\psi_{gy}$ . For these settings you can use `netReg`'s included model selection. We use Powell's BOBYQA algorithm ((Powell 2009)) for gradient-free optimization that is included in the C++ library `Dlib`. Doing the model selection only requires calling `cv.edgnet`:

```
cv <- cv.edgnet(X=X, Y=Y, G.X=aff.mat, family="gaussian", maxit=1000)
print(cv)
```

```
##
## Call: cv.edgnet.default(X = X, Y = Y, G.X = aff.mat, maxit = 1000,
##      family = "gaussian")
##
## Parameters:
```

```
## lambda  psigx  psigy
##      0      0      0
## Family:
## [1] "gaussian"
```

You can use the fitted parameters for the normal `edgenet` function. In this scenario  $\lambda$ ,  $\psi_{gx}$  and  $\psi_{gy}$  should be roughly 0 for three reasons:

- we had enough data and a small number of covariables ( $n > p$ ), so we can find the *BLUE* estimator,
- we created  $X$  and  $Y$  independent of each other,
- our prior graph `aff.mat` had little weight.

Let's do a scenario where we need to shrink some coefficients, i.e.  $n \ll p$ . We choose a small  $p$ , such that the computation does not take too long.

```
p <- 25
X <- matrix(rnorm(10*p), 10)
Y <- matrix(rnorm(10*p), 10)
aff.mat <- matrix(rgamma(p * p, 5, 1), p)
aff.mat <- (t(aff.mat) + aff.mat)
diag(aff.mat) <- 0
cv <- cv.edgenet(X=X, Y=Y, G.X=aff.mat, family="gaussian", maxit=1000)
print(cv)
```

```
##
## Call: cv.edgenet.default(X = X, Y = Y, G.X = aff.mat, maxit = 1000,
##      family = "gaussian")
##
## Parameters:
##   lambda    psigx    psigy
## 6.882653 0.000000 0.000000
## Family:
## [1] "gaussian"
```

In the above example  $\lambda$  should have changed quite a bit, while  $\psi_{gy}$  should still be 0. Since we generated `aff.mat` randomly  $\psi_{gx}$  should be roughly (or exact) zero as well. This makes sense intuitively since we did not put any biological relationships into the affinity matrices.

## References

- Cheng, Wei, Xiang Zhang, Zhishan Guo, Yu Shi, and Wei Wang. 2014. "Graph-Regularized Dual Lasso for Robust EQTL Mapping." *Bioinformatics* 30 (12). Oxford Univ Press: i139–i148.
- Eddelbuettel, Dirk, and Conrad Sanderson. 2014. "RcppArmadillo: Accelerating R with High-Performance C++ Linear Algebra." *Computational Statistics & Data Analysis* 71. Elsevier: 1054–63.
- Friedman, Jerome, Trevor Hastie, and Rob Tibshirani. 2010. "Regularization Paths for Generalized Linear Models via Coordinate Descent." *Journal of Statistical Software* 33 (1). NIH Public Access: 1.
- Friedman, Jerome, Trevor Hastie, Holger Höfling, Robert Tibshirani, and others. 2007. "Pathwise Coordinate Optimization." *The Annals of Applied Statistics* 1 (2). Institute of Mathematical Statistics: 302–32.
- Fu, Wenjiang J. 1998. "Penalized Regressions: The Bridge Versus the Lasso." *Journal of Computational and Graphical Statistics* 7 (3). Taylor & Francis: 397–416.
- Kim, Sunkyung, Wei Pan, and Xiaotong Shen. 2013. "Network-Based Penalized Regression with Application to Genomic Data." *Biometrics* 69 (3). Wiley Online Library: 582–93.
- King, Davis E. 2009. "Dlib-Ml: A Machine Learning Toolkit." *Journal of Machine Learning Research* 10

(Jul): 1755–8.

Li, Caiyan, and Hongzhe Li. 2008. “Network-Constrained Regularization and Variable Selection for Analysis of Genomic Data.” *Bioinformatics* 24 (9). Oxford Univ Press: 1175–82.

Powell, Michael JD. 2009. “The Bobyqa Algorithm for Bound Constrained Optimization Without Derivatives.” *Cambridge NA Report NA2009/06*, University of Cambridge, Cambridge. Citeseer.