impute

April 19, 2009

`impute.knn` *A function to impute missing expression data*

Description

A function to impute missing expression data, using nearest neighbor averaging.

Usage

```
impute.knn(data ,k = 10, rowmax = 0.5, colmax = 0.8, maxp = 1500, rng.seed=36243
```

Arguments

`data`	An expression matrix with genes in the rows, samples in the columns
`k`	Number of neighbors to be used in the imputation (default=10)
`rowmax`	The maximum percent missing data allowed in any row (default 50%). For any rows with more than `rowmax`% missing are imputed using the overall mean per sample.
`colmax`	The maximum percent missing data allowed in any column (default 80%). If any column has more than `colmax`% missing data, the program halts and reports an error.
`maxp`	The largest block of genes imputed using the knn algorithm inside `impute.knn` (default 1500); larger blocks are divided by two-means clustering (recursively) prior to imputation. If `maxp=p`, only knn imputation is done.
`rng.seed`	The seed used for the random number generator (default 362436069) for reproducibility.

Details

`impute.knn` uses k-nearest neighbors in the space of genes to impute missing expression values.

For each gene with missing values, we find the k nearest neighbors using a Euclidean metric, confined to the columns for which that gene is NOT missing. Each candidate neighbor might be missing some of the coordinates used to calculate the distance. In this case we average the distance from the non-missing coordinates. Having found the k nearest neighbors for a gene, we impute the missing elements by averaging those (non-missing) elements of its neighbors. This can fail if ALL the neighbors are missing in a particular element. In this case we use the overall column mean for that block of genes.

Since nearest neighbor imputation costs $O(p\log(p))$ operations per gene, where p is the number of rows, the computational time can be excessive for large p and a large number of missing rows. Our strategy is to break blocks with more than `maxp` genes into two smaller blocks using two-mean clustering. This is done recursively till all blocks have less than `maxp` genes. For each block, k-nearest neighbor imputation is done separately. We have set the default value of `maxp` to 1500. Depending on the speed of the machine, and number of samples, this number might be increased. Making it too small is counter-productive, because the number of two-mean clustering algorithms will increase.

For reproducibility, this function reseeds the random number generator using the seed provided or the default seed (362436069).

Value

`data`	the new imputed data matrix
`rng.seed`	the rng.seed that can be used to reproduce the imputation. This should be saved by any prudent user if different from the default.
`rng.state`	the state of the random number generator, if available, prior to the call to `set.seed`. Otherwise, it is `NULL`. If necessary, this can be used in the calling code to undo the side-effect of changing the random number generator sequence.

Author(s)

Trevor Hastie, Robert Tibshirani, Balasubramanian Narasimhan, and Gilbert Chu

References

Hastie, T., Tibshirani, R., Sherlock, G., Eisen, M., Brown, P. and Botstein, D., Imputing Missing Data for Gene Expression Arrays, Stanford University Statistics Department Technical report (1999), http://www-stat.stanford.edu/~hastie/Papers/missing.pdf

Olga Troyanskaya, Michael Cantor, Gavin Sherlock, Pat Brown, Trevor Hastie, Robert Tibshirani, David Botstein and Russ B. Altman, Missing value estimation methods for DNA microarrays BIOINFORMATICS Vol. 17 no. 6, 2001 Pages 520-525

See Also

set.seed, save

Examples

```
data(khanmiss)
khan.expr <- khanmiss[-1, -(1:2)]
##
## First example
##
if(exists(".Random.seed")) rm(.Random.seed)
khan.imputed <- impute.knn(as.matrix(khan.expr))
##
## khan.imputed$data should now contain the imputed data matrix
## khan.imputed$rng.seed should contain the random number seed used
## in imputation. In the above invocation, it is the default seed.
##
khan.imputed$rng.seed # should be 362436069
khan.imputed$rng.state # should be NULL
##
```

```
## Second example
##
set.seed(12345)
saved.state <- .Random.seed
khan.imputed <- impute.knn(as.matrix(khan.expr))
# Assuming all goes well with no guarantees in case of error...
.Random.seed <- khan.imputed$rng.state
sum(saved.state - khan.imputed$rng.state) # should be zero!
save(khan.imputed, file="khanimputation.Rda")
```

`khanmiss` *Khan microarray data with random missing values*

Description

A text file containing the Khan micorarray data with random missing values introduced for illustrative purposes

Usage

```
data(khanmiss)
```

Format

The data set `khanmiss` consists of 2310 rows and 65 columns. Row 1 has the sample labels, Row 2 has the class labels. The remaining rows are gene expression. Column 1 is a dummy gene number. Column 2 is the gene name. Remaining columns are gene expression.

Please note that this dataset was derived from the original by introducing some random missing values purely for the purpose of illustration.

Source

Khan, J. and Wei, J.S. and Ringner, M. and Saal, L. and Ladanyi, M. and Westermann, F. and Berthold, F. and Schwab, M. and Antonescu, C. and Peterson, C. and and Meltzer, P. (2001) Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural network. Nature Medicine 7, 673-679.

References

Robert Tibshirani, Trevor Hastie, Balasubramanian Narasimhan, and Gilbert Chu (2002). Diagnosis of multiple cancer types by shrunken centroids of gene expression PNAS 99: 6567-6572. Available at www.pnas.org

Examples

```
data(khanmiss)
```

Index