Machine learning techniques available in *pRoloc*

Laurent Gatto*

June 25, 2016

Contents

*lg390@cam.ac.uk

NB This vignette provides more general background about machine learning (ML) and its application to the analysis of organelle proteomics data. This document is currently still under construction; please see the *pRoloc* tutorial for details about the package itself.

TODO Define nomenclature

1 Introduction

For a general practical introduction to *pRoloc*, readers are referred to the tutorial, available using `vignette("pRoloc-tutorial", package = "pRoloc")`. The following document provides a overview of the algorithms available in the package. The respective section describe unsupervised machine learning (USML), supervised machine learning (SML), semi-supervised machine learning (SSML) as implemented in the novelty detection algorithm and transfer learning.

2 Data sets

We provide 41 test data sets in the *pRolocdata* packagethat can be readily used with *pRoloc*. The data set can be listed with `pRolocdata` and loaded with the `data` function. Each data set, including its origin, is individually documented.

The data sets are distributed as *MSnSet* instances. Briefly, these are dedicated containers for quantitation data as well as feature and sample meta-data. More details about *MSnSet*s are available in the *pRoloc* tutorial and in the *MSnbase* package, that defined the class.

```
> library("pRolocdata")
> data(tan2009r1)
> tan2009r1

MSnSet (storageMode: lockedEnvironment)
assayData: 888 features, 4 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: X114 X115 X116 X117
  varLabels: Fractions
  varMetadata: labelDescription
featureData
  featureNames: P20353 P53501 ... P07909 (888
    total)
  fvarLabels: FBgn Protein.ID ... markers.tl (16
    total)
  fvarMetadata: labelDescription
```

```
experimentData: use 'experimentData(object)'
  pubMedIds: 19317464
Annotation:
- - - Processing information - - -
Added markers from  'mrk' marker vector. Thu Jul 16 22:53:44 2015
 MSnbase version: 1.17.12
```

Other omics data

While our primary biological domain is quantitative proteomics, with special emphasis on spatial proteomics, the underlying class infrastructure on which *pRoloc* and implemented in the Bioconductor *MSnbase* package enables the conversion from/to transcriptomics data, in particular microarray data available as *ExpressionSet* objects using the as coercion methods (see the *MSnSet* section in the `MSnbase-development` vignette). As a result, it is straightforward to apply the methods summarised here in detailed in the other *pRoloc* vignettes to these other data structures.

3 Unsupervised machine learning

Unsupervised machine learning refers to clustering, i.e. finding structure in a quantitative, generally multi-dimensional data set of unlabelled data.

Currently, unsupervised clustering facilities are available through the `plot2D` function and the *MLInterfaces* package [1]. The former takes an *MSnSet* instance and represents the data on a scatter plot along the first two principal components. Arbitrary feature meta-data can be represented using different colours and point characters. The reader is referred to the manual page available through `?plot2D` for more details and examples.

pRoloc also implements a `MLearn` method for *MSnSet* instances, allowing to use the relevant infrastructure with the organelle proteomics framework. Although provides a common interface to unsupervised and numerous supervised algorithms, we refer to the *pRoloc* tutorial for its usage to several clustering algorithms.

Note Current development efforts in terms of clustering are described on the *Clustering infrastructure* wiki page[1] and will be incorporated in future version of the package.

4 Supervised machine learning

Supervised machine learning refers to a broad family of classification algorithms. The algorithms learns from a modest set of labelled data points called the training data. Each training data example consists

[1] https://github.com/lgatto/pRoloc/wiki/Clustering-infrastructure

of a pair of inputs: the actual data, generally represented as a vector of numbers and a class label, representing the membership to exactly 1 of multiple possible classes. When there are only two possible classes, on refers to binary classification. The training data is used to construct a model that can be used to classifier new, unlabelled examples. The model takes the numeric vectors of the unlabelled data points and return, for each of these inputs, the corresponding mapped class.

4.1 Algorithms used

k-nearest neighbour (KNN) Function `knn` from package *class*. For each row of the test set, the k nearest (in Euclidean distance) training set vectors are found, and the classification is decided by majority vote over the k classes, with ties broken at random. This is a simple algorithm that is often used as baseline classifier.

Partial least square DA (PLS-DA) Function `plsda` from package *caret*. Partial least square discriminant analysis is used to fit a standard PLS model for classification.

Support vector machine (SVM) A support vector machine constructs a hyperplane (or set of hyperplanes for multiple-class problem), which are then used for classification. The best separation is defined as the hyperplane that has the largest distance (the margin) to the nearest data points in any class, which also reduces the classification generalisation error. To assure liner separation of the classes, the data is transformed using a *kernel function* into a high-dimensional space, permitting liner separation of the classes.

pRoloc makes use of the functions `svm` from package *e1071* and `ksvm` from *kernlab*.

Artificial neural network (ANN) Function `nnet` from package *nnet*. Fits a single-hidden-layer neural network, possibly with skip-layer connections.

Naive Bayes (NB) Function `naiveBayes` from package *e1071*. Naive Bayes classifier that computes the conditional a-posterior probabilities of a categorical class variable given independent predictor variables using the Bayes rule. Assumes independence of the predictor variables, and Gaussian distribution (given the target class) of metric predictors.

Random Forest (RF) Function `randomForest` from package *randomForest*.

Chi-square (χ^2) Assignment based on squared differences between a labelled marker and a new feature to be classified. Canonical protein correlation profile method (PCP) uses squared differences between a labelled marker and new features. In [2], χ^2 is defined as *the [summed] squared deviation of the normalised profile [from the marker] for all peptides divided by the number of data points*, i.e. $\chi^2 = \frac{\sum_{i=1}^{n}(x_i - m_i)^2}{n}$, whereas [3] divide by the value the squared value by the value of the reference

feature in each fraction, i.e. $\chi^2 = \sum_{i=1}^{n} \frac{(x_i - m_i)^2}{m_i}$, where x_i is normalised intensity of feature x in fraction i, m_i is the normalised intensity of marker m in fraction i and n is the number of fractions available. We will use the former definition.

PerTurbo From [4]: PerTurbo, an original, non-parametric and efficient classification method is presented here. In our framework, the manifold of each class is characterised by its Laplace-Beltrami operator, which is evaluated with classical methods involving the graph Laplacian. The classification criterion is established thanks to a measure of the magnitude of the spectrum perturbation of this operator. The first experiments show good performances against classical algorithms of the state-of-the-art. Moreover, from this measure is derived an efficient policy to design sampling queries in a context of active learning. Performances collected over toy examples and real world datasets assess the qualities of this strategy.

The PerTurbo implementation comes from the *pRoloc* packages.

4.2 Estimating algorithm parameters

It is essential when applying any of the above classification algorithms, to wisely set the algorithm parameters, as these can have an important effect on the classification. Such parameters are for example the width $sigma$ of the Radial Basis Function (Gaussian kernel) $exp(-\sigma\|x - x'\|^2)$ and the $cost$ (slack) parameter (controlling the tolerance to mis-classification) of our SVM classifier. The number of neighbours k of the KNN classifier is equally important as will be discussed in this sections.

Figure 1 illustrates the effect of different choices of k using organelle proteomics data from [5] (`dunkley2006` from *pRolocdata*). As highlighted in the squared region, we can see that using a low k ($k = 1$ on the left) will result in very specific classification boundaries that precisely follow the contour or our marker set as opposed to a higher number of neighbours ($k = 8$ on the right). While one could be tempted to believe that *optimised* classification boundaries are preferable, it is essential to remember that these boundaries are specific to the marker set used to construct them, while there is absolutely no reason to expect these regions to faithfully separate any new data points, in particular proteins that we wish to classify to an organelle. In other words, the highly specific $k = 1$ classification boundaries are *over-fitted* for the marker set or, in other words, lack generalisation to new instances. We will demonstrate this using simulated data taken from [6] and show what detrimental effect *over-fitting* has on new data.

Figure 2 uses 2×100 simulated data points belonging to either of the orange or blue classes. The genuine classes for all the points is known (solid lines) and the KNN algorithm has been applied using $k = 1$ (left) and $k = 100$ (right) respectively (purple dashed lines). As in our organelle proteomics examples, we observe that when k = 1, the decision boundaries are overly flexible and identify patterns in the data that do not reflect to correct boundaries (in such cases, the classifier is said to have low bias but very high variance). When a large k is used, the classifier becomes inflexible and produces approximate and nearly linear separation boundaries (such a classifier is said to have low variance but high bias). On this simulated data set, neither $k = 1$ nor $k = 100$ give good predictions and have test error rates (i.e. the proportion of wrongly classified points) of 0.1695 and 0.1925, respectively.

To quantify the effect of flexibility and lack thereof in defining the classification boundaries, [6] calculate

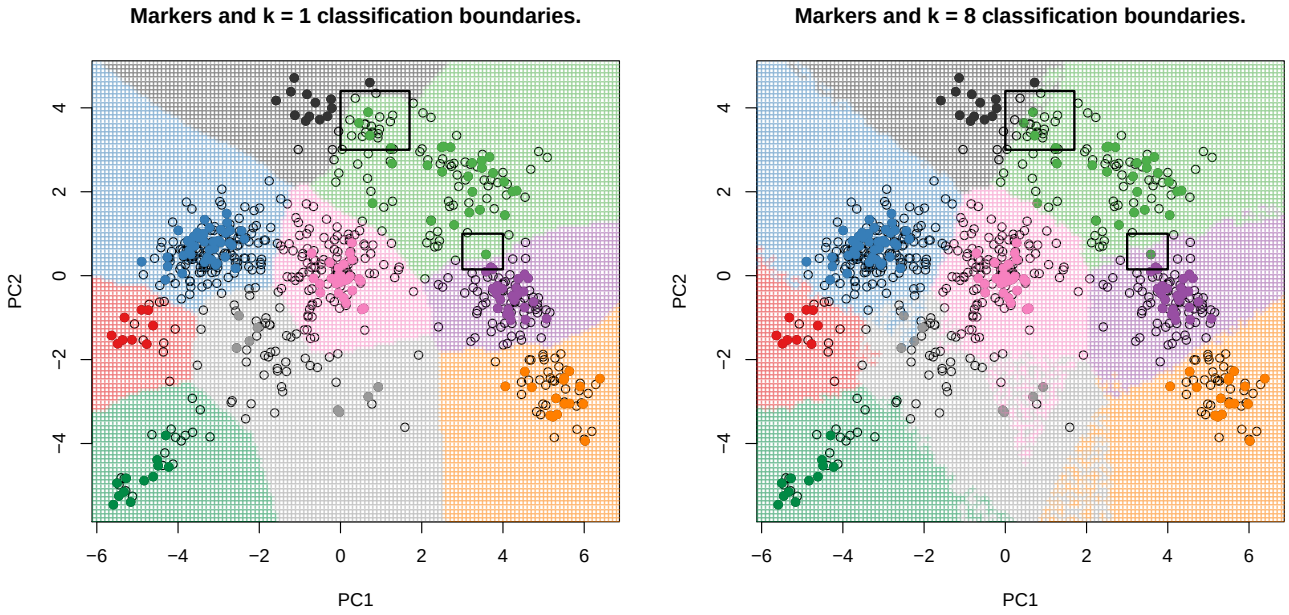

Figure 1: Classification boundaries using $k = 1$ or $k = 8$ on the `dunkley2006` data.

the classification error rates using training data (training error rate) and testing data (testing error rate). The latter is completely new data that was not used to assess the model error rate when defining algorithm parameters; one often says that the model used for classification has not *seen* this data. If the model performs well on new data, it is said to generalise well. This is a quality that is required in most cases, and in particular in our organelle proteomics experiments where the training data corresponds to our marker sets. Figure 3 plots the respective training and testing error rates as a function of $1/k$ which is a reflection of the flexibility/complexity of our model; when $1/k = 1$, i.e. $k = 1$ (far right), we have a very flexible model with the risk of over-fitting. Greater values of k (towards the left) characterise less flexible models. As can be seen, high values of k produce poor performance for both training and testing data. However, while the training error steadily decreases when the model complexity increases (smaller k), the testing error rate displays a typical U-shape: at a value around $k = 10$, the testing error rate reaches a minimum and then starts to increase due to over-fitting to the training data and lack of generalisation of the model.

These results show that adequate optimisation of the model parameters are essential to avoid either too flexible models (that do not generalise well to new data) or models that do not describe the decision boundaries adequately. Such parameter selection is achieved by cross validation, where the initial marker proteins are separated into training data used to build classification models and independent testing data used to assess the model on new data.

We recommend the book *An Introduction to Statistical Learning*[2] by [6] for a more detailed introduction of machine learning.

[2] http://www-bcf.usc.edu/~gareth/ISL/

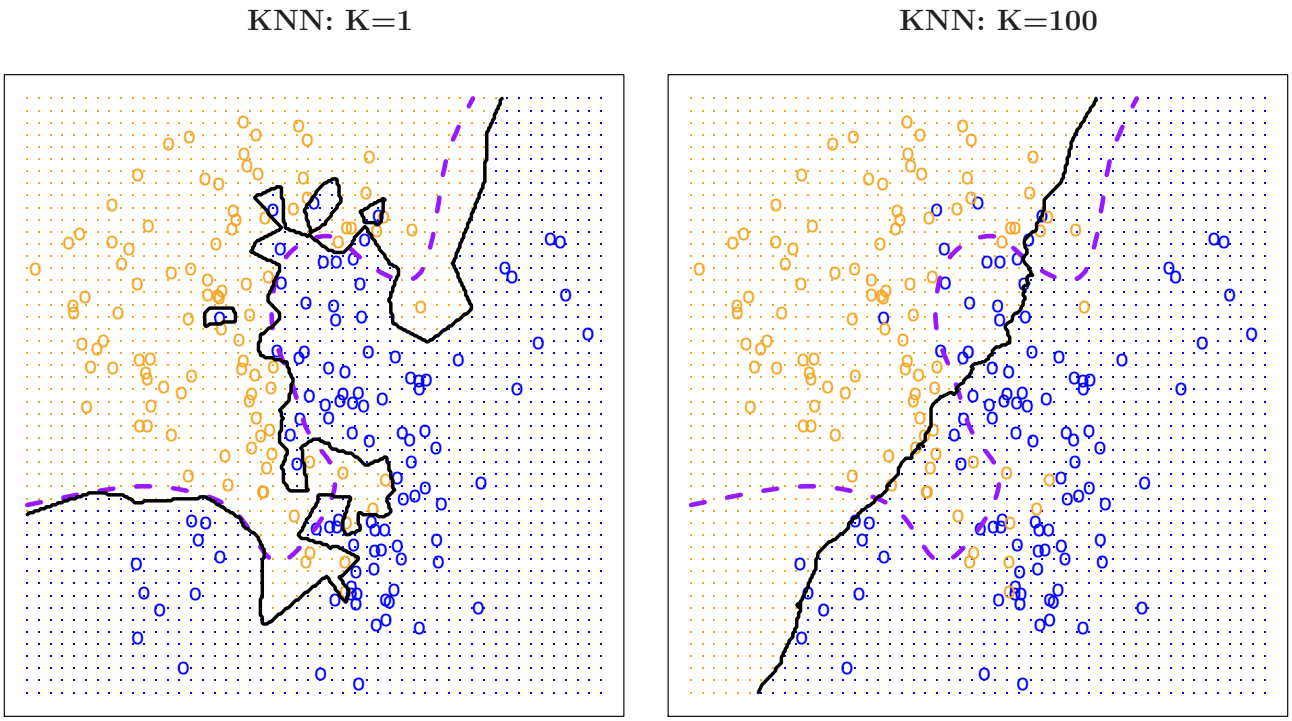

Figure 2: The KNN classifier using $k = 1$ (left, solid classification boundaries) and $k = 100$ (right, solid classification boundaries) compared the Bayes decision boundaries (see original material for details). Reproduced with permission from [6].

4.3 Default analysis scheme

Below, we present a typical classification analysis using *pRoloc*. The analysis typically consists of two steps. The first one is to optimise the classifier parameters to be used for training and testing (see above). A range of parameters are tested using the labelled data, for which the labels are known. For each set of parameters, we hide the labels of a subset of labelled data and use the other part to train a model and apply in on the testing data with hidden labels. The comparison of the estimated and expected labels enables to assess the validity of the model and hence the adequacy if the parameters. Once adequate parameters have been identified, they are used to infer a model on the complete organelle marker set and used to infer the sub-cellular location of the unlabelled examples.

Parameter optimisation

Algorithmic performance is estimated using a stratified 20/80 partitioning. The 80% partitions are subjected to 5-fold cross-validation in order to optimise free parameters via a grid search, and these parameters are then applied to the remaining 20%. The procedure is repeated $n = 100$ `times` to sample n accuracy metrics (see below) values using n, possibly different, optimised parameters for evaluation.

Models accuracy is evaluated using the F1 score, $F1 = 2\,\frac{precision \times recall}{precision + recall}$, calculated as the harmonic

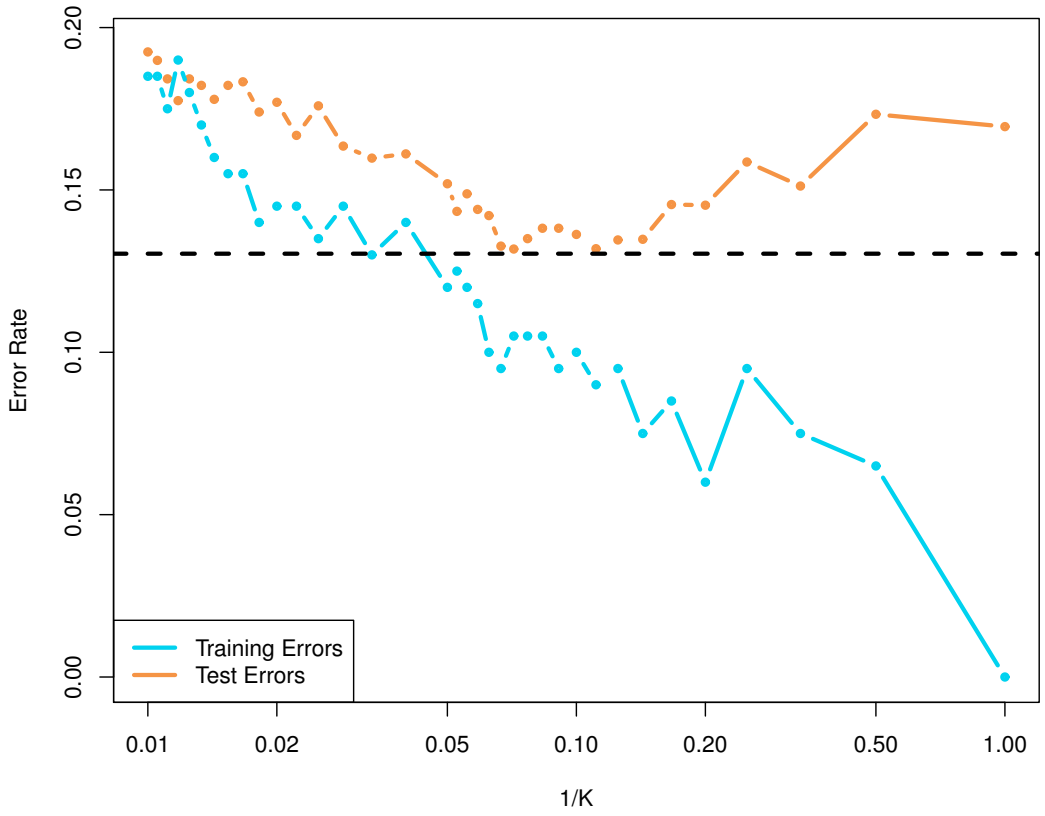

Figure 3: Effect of train and test error with respect to model complexity. The former decreases for lower values of k while the test error reaches a minimum around $k = 10$ before increasing again. Reproduced with permission from [6].

mean of the precision ($precision = \frac{tp}{tp+fp}$, a measure of *exactness* – returned output is a relevant result) and recall ($recall = \frac{tp}{tp+fn}$, a measure of *completeness* – indicating how much was missed from the output). What we are aiming for are high generalisation accuracy, i.e high $F1$, indicating that the marker proteins in the test data set are consistently correctly assigned by the algorithms.

The results of the optimisation procedure are stored in an *GenRegRes* object that can be inspected, plotted and best parameter pairs can be extracted.

For a given algorithm `alg`, the corresponding parameter optimisation function is names `algOptimisation` or, equivalently, `algOptimization`. See table 1 for details. A description of each of the respective model parameters is provided in the optimisation function manuals, available through `?algOptimisation`.

```
> params <- svmOptimisation(tan2009r1, times = 10, xval = 5, verbose = FALSE)
> params

Object of class "GenRegRes"
Algorithm: svm
Hyper-parameters:
 cost: 0.0625 0.125 0.25 0.5 1 2 4 8 16
 sigma: 0.001 0.01 0.1 1 10 100
```

```
Design:
 Replication: 10 x 5-fold X-validation
 Partitioning: 0.2/0.8 (test/train)
Results
 macro F1:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.7979  0.8403  0.8699  0.8739  0.8943  0.9748
 best sigma: 1 0.1
 best cost: 8 2 16 4
Use getWarnings() to see warnings.
```

Classification

```
> tan2009r1 <- svmClassification(tan2009r1, params)
> tan2009r1

MSnSet (storageMode: lockedEnvironment)
assayData: 888 features, 4 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: X114 X115 X116 X117
  varLabels: Fractions
  varMetadata: labelDescription
featureData
  featureNames: P20353 P53501 ... P07909 (888
    total)
  fvarLabels: FBgn Protein.ID ... svm.scores (18
    total)
  fvarMetadata: labelDescription
experimentData: use 'experimentData(object)'
  pubMedIds: 19317464
Annotation:
- - - Processing information - - -
Added markers from  'mrk' marker vector. Thu Jul 16 22:53:44 2015
Performed svm prediction (sigma=1 cost=8) Sat Jun 25 22:01:45 2016
 MSnbase version: 1.17.12
```

4.4 Customising model parameters

Below we illustrate how to weight different classes according to the number of labelled instances, where large sets are down weighted. This strategy can help with imbalanced designs.

```
> w <- table(fData(markerMSnSet(dunkley2006))$markers)
> wpar <- svmOptimisation(dunkley2006, class.weights = w)
> wres <- svmClassification(dunkley2006, wpar, class.weights = w)
```

parameter optimisation	classification	algorithm	package
knnOptimisation	knnClassification	nearest neighbour	class
knntlOptimisation	knntlClassification	nearest neighbour transfer learning	pRoloc
ksvmOptimisation	ksvmClassification	support vector machine	kernlab
nbOptimisation	nbClassification	naive bayes	e1071
nnetOptimisation	nnetClassification	neural networks	nnet
perTurboOptimisation	perTurboClassification	PerTurbo	pRoloc
plsdaOptimisation	plsdaClassification	partial least square	caret
rfOptimisation	rfClassification	random forest	randomForest
svmOptimisation	svmClassification	support vector machine	e1071

Table 1: Supervised ML algorithm available in *pRoloc*.

5 Comparison of different classifiers

Several supervised machine learning algorithms have already been applied to organelle proteomics data classification: partial least square discriminant analysis in [5, 7], support vector machines (SVMs) in [8], random forest in [9], neural networks in [10], naive Bayes [11]. In our HUPO 2011 poster (see `vignette("HUPO_2011_poster", package = "pRoloc")`), we show that different classification algorithms provide very similar performance. We have extended this comparison on various datasets distributed in the *pRolocdata* package. On figure 4, we illustrate how different algorithms reach very similar performances on most of our test datasets.

6 Semi-supervised machine learning and novelty detection

The *phenoDisco* algorithm is a semi-supervised novelty detection method by [12] (figure 5). It uses the labelled (i.e. markers, noted D_L) and unlabelled (i.e. proteins of unknown localisation, noted D_U) sets of the input data. The algorithm is repeated N times (the `code` argument in the `phenoDisco` function). At each iteration, each organelle class D_L^i and the unlabelled complement are clustered using Gaussian mixture modelling. While unlabelled members that systematically cluster with D_L^i and pass outlier detection are labelled as new putative members of class i, any example of D_U which are not merged with any any of the D_L^i and are consistently clustered together throughout the N iterations are considered members of a new phenotype.

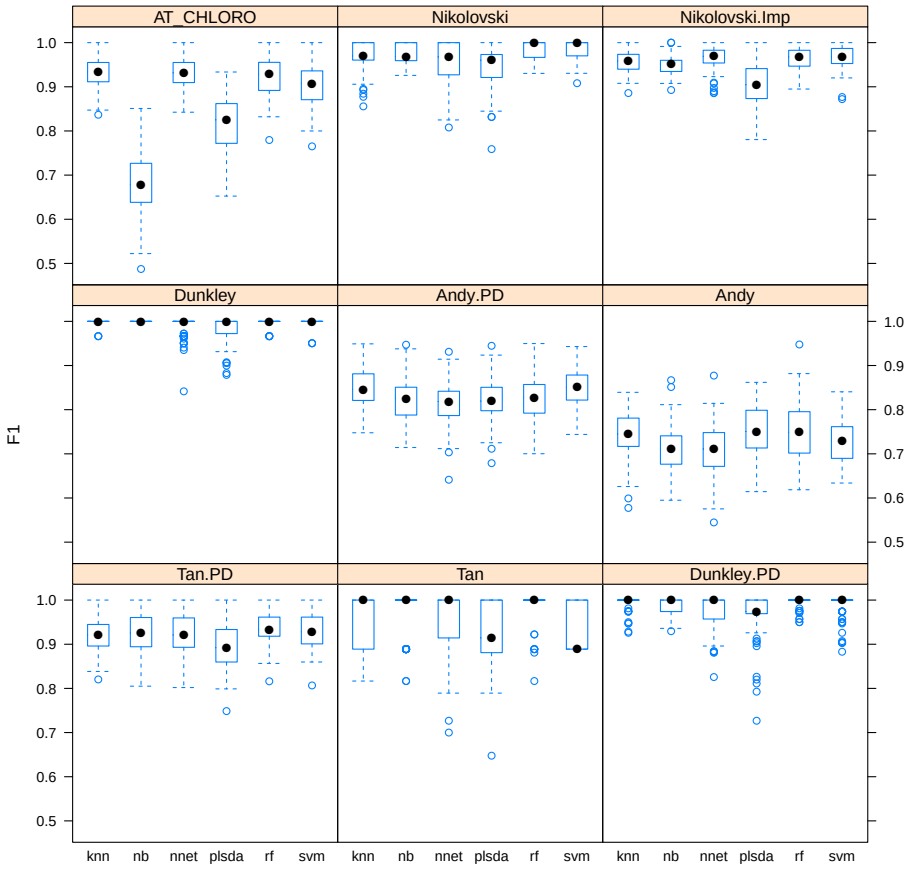

Figure 4: Comparison of classification performances of several contemporary classification algorithms on data from the *pRolocdata* package.

7 Transfer learning

When multiple sources of data are available, it is often beneficial to take all or several into account with the aim of increasing the information to tackle a problem of interest. While it is at times possible to combine these different sources of data, this can lead to substantially harm to performance of the analysis when the different data sources are of variable signal-to-noise ratio or the data are drawn from different domains and recorded by different encoding (quantitative and binary, for example). If we defined the following two data source

- *primary* data, of high signal-to-noise ratio, but general available in limited amounts;
- *auxiliary* data, of limited signal-to-noise, and available in large amounts;

then, a *transfer learning* algorithm will efficiently support/complement the primary target domain with auxiliary data features without compromising the integrity of our primary data.

We have developed a transfer learning framework and applied to the analysis of spatial proteomics data, as described in the `pRoloc-transfer-learning` vignette.

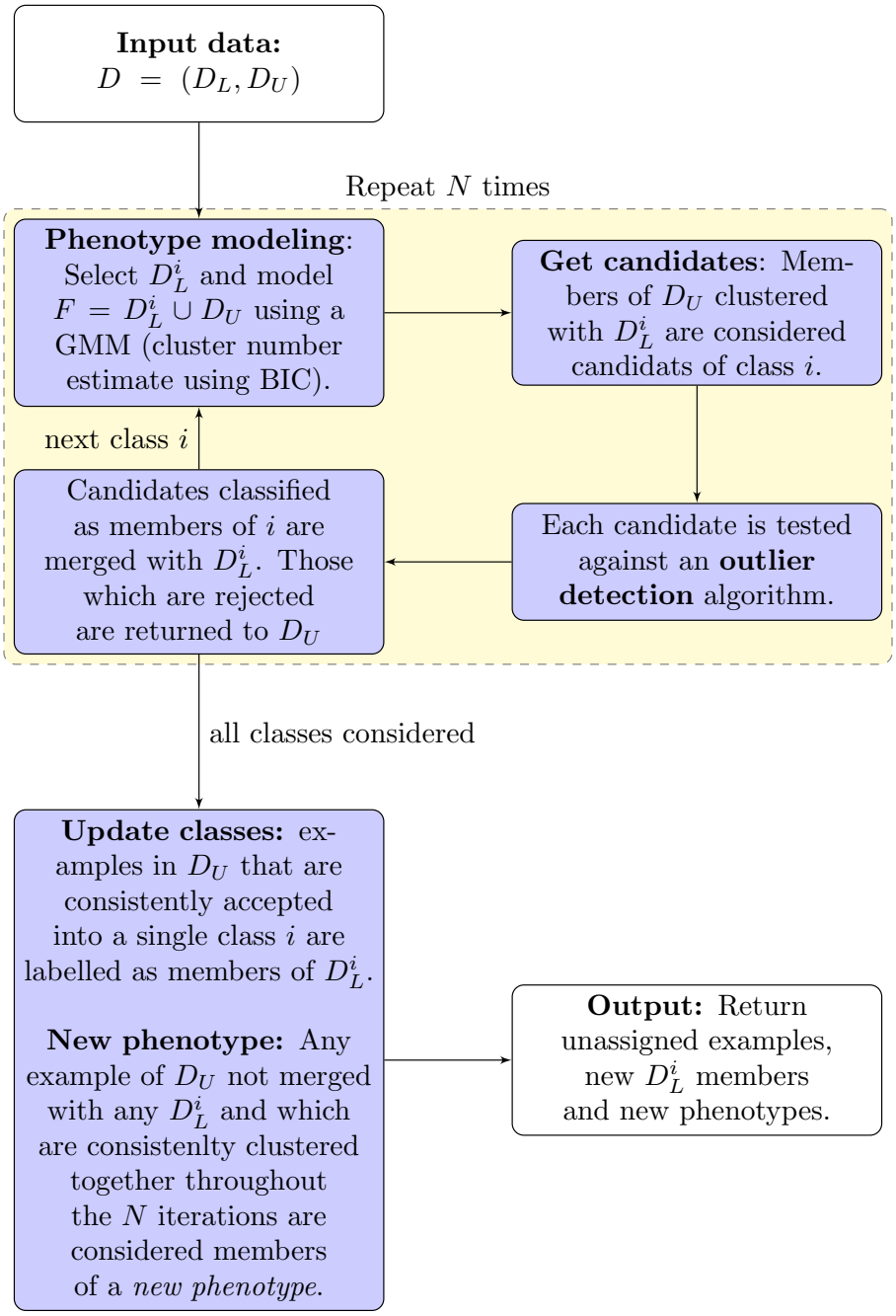

Figure 5: The PhenoDisco iterative algorithm.

Session information

All software and respective versions used to produce this document are listed below.

- R version 3.3.0 (2016-05-03), `x86_64-w64-mingw32`
- Locale: `LC_COLLATE=C`, `LC_CTYPE=English_United States.1252`, `LC_MONETARY=English_United States.1252`, `LC_NUMERIC=C`, `LC_TIME=English_United States.1252`
- Base packages: base, datasets, grDevices, graphics, methods, parallel, stats, stats4, utils
- Other packages: AnnotationDbi 1.34.3, Biobase 2.32.0, BiocGenerics 0.18.0, BiocParallel 1.6.2, BiocStyle 2.0.2, GO.db 3.3.0, IRanges 2.6.1, MLInterfaces 1.52.0, MSnbase 1.20.7, ProtGenerics 1.4.0, Rcpp 0.12.5, S4Vectors 0.10.1, XML 3.98-1.4, annotate 1.50.0, cluster 2.0.4, knitr 1.13, mzR 2.6.2, pRoloc 1.12.4, pRolocdata 1.10.0, xtable 1.8-2
- Loaded via a namespace (and not attached): BiocInstaller 1.22.3, DBI 0.4-1, DEoptimR 1.0-4, FNN 1.1, MALDIquant 1.14, MASS 7.3-45, Matrix 1.2-6, MatrixModels 0.4-1, R6 2.1.2, RColorBrewer 1.1-2, RCurl 1.95-4.8, RSQLite 1.0.0, SparseM 1.7, affy 1.50.0, affyio 1.42.0, assertthat 0.1, base64enc 0.1-3, biomaRt 2.28.0, bitops 1.0-6, car 2.1-2, caret 6.0-70, class 7.3-14, codetools 0.2-14, colorspace 1.2-6, digest 0.6.9, diptest 0.75-7, doParallel 1.0.10, dplyr 0.5.0, e1071 1.6-7, evaluate 0.9, flexmix 2.3-13, foreach 1.4.3, formatR 1.4, fpc 2.1-10, gbm 2.1.1, gdata 2.17.0, genefilter 1.54.2, ggplot2 2.1.0, ggvis 0.4.2, grid 3.3.0, gtable 0.2.0, gtools 3.5.0, highr 0.6, htmltools 0.3.5, htmlwidgets 0.6, httpuv 1.3.3, hwriter 1.3.2, impute 1.46.0, iterators 1.0.8, jsonlite 0.9.22, kernlab 0.9-24, lattice 0.20-33, limma 3.28.12, lme4 1.1-12, lpSolve 5.6.13, magrittr 1.5, mclust 5.2, mgcv 1.8-12, mime 0.4, minqa 1.2.4, mlbench 2.1-1, modeltools 0.2-21, munsell 0.4.3, mvtnorm 1.0-5, mzID 1.10.2, nlme 3.1-128, nloptr 1.0.4, nnet 7.3-12, pbkrtest 0.4-6, pcaMethods 1.64.0, pls 2.5-0, plyr 1.8.4, prabclus 2.2-6, preprocessCore 1.34.0, proxy 0.4-15, quantreg 5.26, randomForest 4.6-12, rda 1.0.2-2, reshape2 1.4.1, rgl 0.95.1441, rmarkdown 0.9.6, robustbase 0.92-6, rpart 4.1-10, sampling 2.7, scales 0.4.0, sfsmisc 1.1-0, shiny 0.13.2, splines 3.3.0, stringi 1.1.1, stringr 1.0.0, survival 2.39-4, threejs 0.2.2, tibble 1.0, tools 3.3.0, trimcluster 0.1-2, vsn 3.40.0, yaml 2.1.13, zlibbioc 1.18.0

References

[1] Vince Carey, Robert Gentleman, Jess Mar, , contributions from Jason Vertrees, and Laurent Gatto. *MLInterfaces: Uniform interfaces to R machine learning procedures for data in Bioconductor containers*. R package version 1.41.1.

[2] Jens S. Andersen, Christopher J. Wilkinson, Thibault Mayor, Peter Mortensen, Erich A. Nigg, and Matthias Mann. Proteomic characterization of the human centrosome by protein correlation profiling. *Nature*, 426(6966):570–574, Dec 2003. URL: http://dx.doi.org/10.1038/nature02166, doi:10.1038/nature02166.

[3] Sebastian Wiese, Thomas Gronemeyer, Rob Ofman, Markus Kunze, Cludia P. Grou, Jos A. Almeida, Martin Eisenacher, Christian Stephan, Heiko Hayen, Lukas Schollenberger, Thomas Korosec,

Hans R. Waterham, Wolfgang Schliebs, Ralf Erdmann, Johannes Berger, Helmut E. Meyer, Wilhelm Just, Jorge E. Azevedo, Ronald J. A. Wanders, and Bettina Warscheid. Proteomics characterization of mouse kidney peroxisomes by tandem mass spectrometry and protein correlation profiling. *Mol Cell Proteomics*, 6(12):2045–2057, Dec 2007. URL: http://dx.doi.org/10.1074/mcp.M700169-MCP200, doi:10.1074/mcp.M700169-MCP200.

[4] Nicolas Courty, Thomas Burger, and Johann Laurent. Perturbo: A new classification algorithm based on the spectrum perturbations of the laplace-beltrami operator. In Dimitrios Gunopulos, Thomas Hofmann, Donato Malerba, and Michalis Vazirgiannis, editors, *in the proceedings of ECML/PKDD (1)*, volume 6911 of *Lecture Notes in Computer Science*, pages 359–374. Springer, 2011.

[5] Tom P. J. Dunkley, Svenja Hester, Ian P. Shadforth, John Runions, Thilo Weimar, Sally L. Hanton, Julian L. Griffin, Conrad Bessant, Federica Brandizzi, Chris Hawes, Rod B. Watson, Paul Dupree, and Kathryn S. Lilley. Mapping the arabidopsis organelle proteome. *Proc Natl Acad Sci USA*, 103(17):6518–6523, Apr 2006. URL: http://dx.doi.org/10.1073/pnas.0506958103, doi:10.1073/pnas.0506958103.

[6] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning: with Applications in R*. Springer Texts in Statistics. Springer, 2013.

[7] Denise J Tan, Heidi Dvinge, Andy Christoforou, Paul Bertone, Alfonso A Martinez, and Kathryn S Lilley. Mapping organelle proteins and protein complexes in drosophila melanogaster. *J Proteome Res*, 8(6):2667–78, Jun 2009. doi:10.1021/pr800866n.

[8] Matthew W. B. Trotter, Pawel G. Sadowski, Tom P. J. Dunkley, Arnoud J. Groen, and Kathryn S. Lilley. Improved sub-cellular resolution via simultaneous analysis of organelle proteomics data across varied experimental conditions. *PROTEOMICS*, 10(23):4213–4219, 2010. URL: http://dx.doi.org/10.1002/pmic.201000359, doi:10.1002/pmic.201000359.

[9] S Ohta, J C Bukowski-Wills, L Sanchez-Pulido, Fde L Alves, L Wood, Z A Chen, M Platani, L Fischer, D F Hudson, C P Ponting, T Fukagawa, W C Earnshaw, and J Rappsilber. The protein composition of mitotic chromosomes determined using multiclassifier combinatorial proteomics. *Cell*, 142(5):810–21, Sep 2010. doi:10.1016/j.cell.2010.07.047.

[10] M Tardif, A Atteia, M Specht, G Cogne, N Rolland, S Brugire, M Hippler, M Ferro, C Bruley, G Peltier, O Vallon, and L Cournac. Predalgo: a new subcellular localization prediction tool dedicated to green algae. *Mol Biol Evol*, 29(12):3625–39, Dec 2012. doi:10.1093/molbev/mss178.

[11] N Nikolovski, D Rubtsov, M P Segura, G P Miles, T J Stevens, T P Dunkley, S Munro, K S Lilley, and P Dupree. Putative glycosyltransferases and other plant golgi apparatus proteins are revealed by LOPIT proteomics. *Plant Physiol*, 160(2):1037–51, Oct 2012. doi:10.1104/pp.112.204263.

[12] Lisa M Breckels, Laurent Gatto, Andy Christoforou, Arnoud J Groen, Kathryn S Lilley, and Matthew W B Trotter. The effect of organelle discovery upon sub-cellular protein localisation. *J Proteomics*, Mar 2013. doi:10.1016/j.jprot.2013.02.019.