EPX: R package for the ensemble of subsets of variables for highly unbalanced binary classification

Grace G. Hsu[a], Jabed H. Tomal[b,*], William J. Welch[a]

[a] *Department of Statistics, University of British Columbia, 3182 Earth Sciences Building, 2207 Main Mall, Vancouver, BC, Canada V6T 1Z4*
[b] *Department of Mathematics and Statistics, Thompson Rivers University, 805 TRU Way, Kamloops, BC, Canada V2C 0C8*

Abstract

Background and Objective: In binary classification problems with a rare class of interest, there is relatively little information available for the rare class to build a model. On the other hand, the number of useful variables to develop a model for classification can be high-dimensional. For example, in drug discovery, there are usually a very few bioactive compounds in a large chemical library, whereas thousands of potentially useful explanatory variables characterize a compound's chemical structure. The sparsity of information for the rare class of interest makes it difficult for the standard classification models to exploit the richness of the useful explanatory variables. Thus, the objective of this paper is to develop an R package which clusters the variables into diverse subsets to be aggregated into a powerful ensemble for the detection of a rare class object.

Methods: The ensemble of phalanxes (EPX) builds a classifier by exploiting the richness of feature variables using several diverse subsets of variables, called *phalanxes*, to detect the rare class and outperforms the predictive ranking performance of many competitive state-of-the-art classification methods.

Results: We present an R package **EPX** which implements the algorithm to form the ensemble of phalanxes as well as its associated functions. We further show how the ensemble of phalanxes can be constructed using parallel computing to

*Corresponding author
Email address: jtomal@tru.ca (Jabed H. Tomal)

lower the computational burden given high-dimensional data.

Conclusions: The R package **EPX** shows a flexible way of clustering feature variable space into smaller and diverse subsets of variable to develop an ensemble of phalanxes which better predicts a rare class object in a highly unbalanced two class classification problem. The ensemble EPX will be useful to detect the rare drug-like active biomolecules for development in drug discovery and homologous proteins using similarity scores of amino acid sequences in protein homology. The package **EPX** is freely available to download from CRAN (`https://CRAN.R-project.org/package=EPX`).

Keywords: R package, EPX, Ensemble learning, Machine learning, Drug discovery, Protein homology

---

1. Introduction

The ensemble of phalanxes (EPX) classifier is the result of the algorithm proposed by Tomal et al. (2015, 2016, 2019) and motivated by the applications in drug discovery and protein homology where the goal is statistical detection of a rare class of objects (e.g., drug-like active biomolecules or homologous proteins) in a two-class classification problem. The algorithm clusters the explanatory variables (e.g., variables representing three dimensional structure of a chemical compound in drug discovery or similarity scores of aligned amino acid sequences in protein homology) into disjoint subsets called *phalanxes* such that the phalanxes work well in the constituent models when aggregated together. By "working well," we mean by some measure of how highly the observations belonging to the desired rare class are ranked via their estimated probabilities. Variables in different phalanxes and hence models can contribute to the overall ensemble without working against each other by way of deselection. Altogether with the relatively low-dimensionality of each phalanx, it follows that datasets with high-dimension can be exploited in the face of limited response information. For more details of the methods and applications in drug discovery and protein homology problems, please see Tomal et al. (2015, 2016, 2019).

The EPX classifier's better performance against the already competitive random forest (RF; Breiman, 2001) in the areas of QSAR studies (Tomal et al., 2015, 2016), protein homology (Tomal et al., 2019), and potential for application in other problems with little information in the response variable relative to the number of explanatory variables thus motivated the implementation of the phalanx-formation algorithm in R (R Core Team, 2020) package **EPX**. The **EPX** package is available from the Comprehensive R Archive Network (CRAN) at: `https://CRAN.R-project.org/package=EPX`.

The rest of article is organised as follows. Section 2 presents the methods by briefly introducing the performance assessment metrics and the phalanx formation algorithm. Section 3 describes the results from the **EPX** package starting with the `epx` function, which trains an EPX classifier. The remainder of the section describes an example from drug discovery and the main aspects of the other functions in the package. Throughout, we follow this drug discovery example and show how the package can be used for via summary, prediction, cross-validation and parallel computing. Section 4 discusses on further capabilities of the package in terms of ways to customise aspects of the algorithm via additional base classifiers and performance measures. Finally, section 5 concludes the article.

2. Methods

2.1. Performance measures

Consider a binary classification problem where the relevant class is extremely rare and $\hat{\pi}(\mathbf{x})$ estimates the probability of relevance $\pi(\mathbf{x})$ given the vector of features $\mathbf{x}$. In this highly unbalanced classification problem, building $\hat{\pi}$ to minimise the misclassification rate is not useful because this may often results in a trivial classifier that classifies all objects as irrelevant. Instead, the *average hit rate* (AHR) is considered as a better criterion for evaluating the performance of classifiers for unbalanced classification problems (Chapter 3; Wang, 2005).

The average hit rate summarises a classifier's hit curve, which plots the number of relevant objects against the number of total objects selected, where object selection is based on their ranking according to $\hat{\pi}$. Given n objects ranked in descending order according to their estimated probabilities of relevance $\hat{\pi}$, let this ordered list of objects have the true response values $y_{(1)}, y_{(2)}, ..., y_{(n)}$, where

$$y_{(i)} = \begin{cases} 1, \text{ if the } i\text{th object is relevant} \\ 0, \text{ if the } i\text{th object is irrelevant,} \end{cases}$$

and $M = \sum_{i=1}^{n} y_{(i)}$ is the total number of relevant objects. The AHR is hence defined as

$$\text{AHR} = \frac{1}{M} \sum_{i=1}^{n} \left[y_{(i)} \sum_{j=1}^{i} \frac{y_{(j)}}{i} \right]. \tag{1}$$

Note that AHR has a definition equivalent to that of *average precision* (APR) and a classifier is judged as performing well when it ranks (i.e., detects) the truly relevant objects more highly. Furthermore, when the $\hat{\pi}$ values for all n objects are not unique, we have objects with the same ranking. In such cases, we assume that the objects with tied ranks are in random order, thus allowing the calculation of the expected AHR in closed form (Chapter 3; Wang, 2005). The AHR is bounded between 0 and 1 where larger values indicate better predictive ranking.

While there are other performance measures to summarize a hit curve (further details in section 4), there are several advantageous properties of AHR detailed in (Chapter 3.4; Wang, 2005). Thus, AHR is the default performance measure in the **EPX** package.

2.2. Phalanx-formation algorithm

The main idea of the phlanx formation algorithm (Tomal et al., 2015, 2016) is that it groups the explanatory variables into disjoint subsets called *phalanxes* such that the variables in each phalanx work better in terms of some perfor-

mance measure together in a single model than in separate models, while also accounting for the performance of the ensemble of the models. The steps of phalanx-formation algorithm and subsequent construction of the ensemble of phalanxes is summarised in Figure 1.

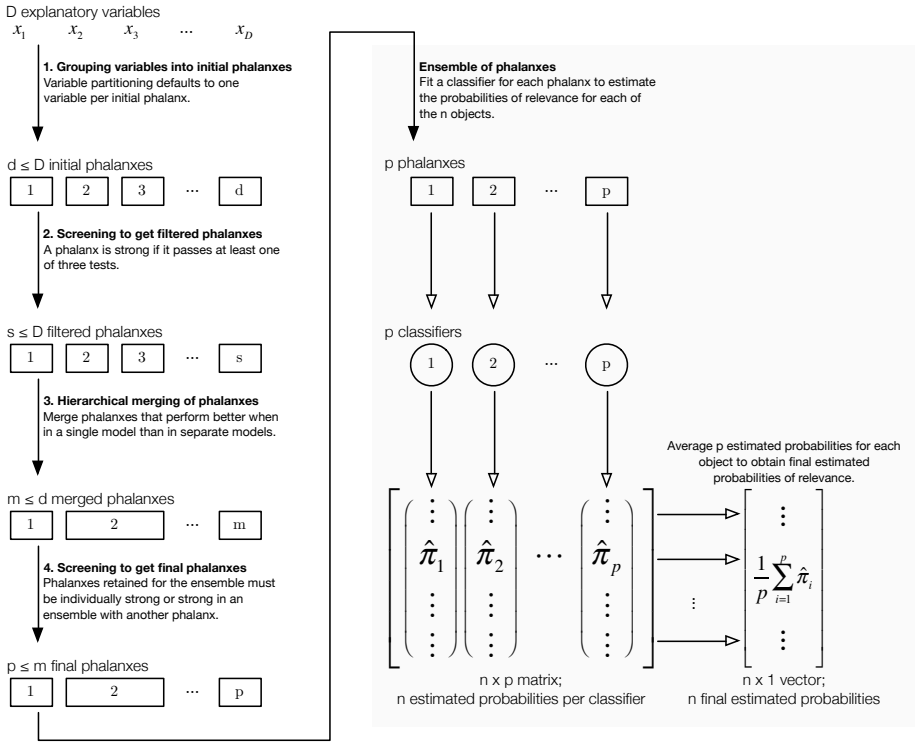

Figure 1: Algorithm for phalanx-formation and subsequent construction of the ensemble of the phalanxes. D variables are partitioned into d initial phalanxes, filtered to s phalanxes, merged into m phalanxes, and then filtered again to p final phalanxes used for the final ensemble ($D \geq d \geq s \geq m \geq p$).

Suppose we have a training data set with n observations (e.g., compounds in a chemical library) and D explanatory variables (e.g., variables representing three-dimensional structures of chemical compounds). The four steps of the phalanx-formation algorithm are as follows:

1. **Initial phalanxes**. The D explanatory variables are partitioned into d initial phalanxes where $1 < d \leq D$.

2. **Filtered phalanxes**. The d initial phalanxes are filtered to $s \leq d$ filtered phalanxes. Various comparisons are done between each phalanx and the reference distribution of some assessment criterion a, such as AHR, under random ranking. For the computation of the reference distribution see Section 4 of Tomal et al. (2015). In summary,

   - a_α: The α quantile of the reference distribution of the assessment criterion a. Default is $\alpha = 0.95$.
   - $\hat{\pi}_i$: The estimated probabilities of relevance from the classifier built only with the variables in phalanx i.
   - $a_i = a(\hat{\pi}_i)$: The performance measure of phalanx i found by passing $\hat{\pi}_i$ through the assessment criterion a.
   - $\hat{\pi}_{ij}$: The estimated probabilities of relevance from the classifier built with all the variables in phalanx i as well as those in phalanx j ($i \neq j$).
   - $a_{ij} = a(\hat{\pi}_{ij})$: The performance measure of a phalanx that includes all the variables in phalanxes i and j ($i \neq j$).
   - $a_{\overline{ij}} = a\left(\frac{\hat{\pi}_i + \hat{\pi}_j}{2}\right)$: The performance measure of an ensemble of two models built with phalanxes i and j, respectively.

   That is, phalanx i survives if it passes at least one of the following three tests.

   - Phalanx i performs well alone:

   $$a_i \geq a_\alpha. \tag{2}$$

   - Phalanx i improves the performance of another phalanx j when the two are used to build a single model:

   $$a_{0.5} + (a_{ij} - a_j) \geq a_\alpha. \tag{3}$$

   - Phalanx i improves the performance of another phalanx j when the

two are in an ensemble of two models:

$$a_{0.5} + \left(a_{\overline{ij}} - a_j\right) \geq a_\alpha. \tag{4}$$

The surviving s phalanxes are renamed accordingly from 1 to s.

3. **Merged phalanxes.** The s filtered phalanxes are merged hierarchically as follows: each iteration merges the pair of phalanxes i and j that minimises $m_{ij} = a_{\overline{ij}}/a_{ij}$. When $m_{ij} < 1$, this means that phalanxes i and j perform better in a single model than as an ensemble. Thus, each merge reduces the number of phalanxes by one until $m_{ij} \geq 1$ for all phalanxes i, j. We are left with $m \leq s$ phalanxes.
4. **Final phalanxes.** The m phalanxes are filtered in a way that a phalanx survives if it performs well alone as in condition (2), or if it performs well in an ensemble with at least one other phalanx as in condition (4). Note that condition (3) is automatically satisfied due to the merging done in the previous step.

After obtaining the final p phalanxes, we construct the ensemble of phalanxes by building a classifier for each of the phalanxes. Each classifier then produces an n-length vector of estimated probabilities of relevance, resulting in p such vectors: $\hat{\pi}_1, \hat{\pi}_2, ..., \hat{\pi}_p$. The final vector of probabilities for the n objects are the result of averaging the p vectors of probabilities.

3. Results

3.1. The EPX package

Training an EPX model is done by the `epx` function, where `x` and `y` are the explanatory variables contained in a data frame and the binary response variable vector (1 is the rare or relevant class) respectively. The main parameters of the `epx` are as follows:

- `phalanxes.initial`: designates the phalanx membership of each explanatory variable as a numeric vector. Defaults to one variable per phalanx.

- `classifier`: indicates what models we fit for each phalanx for the model as a string. Defaults to *random forest*.
- `classifier.args`: allows for some modification of the arguments for the choice of classifier, such as the number of trees allowed per random forest. Leaving an empty list means that the classifier will use its default settings.
- `performance`: indicates the choice of performance measure. Defaults to AHR.
- `computing`: indicates whether to compute in *parallel* and requires the user to register a parallel backend. Defaults to sequential computing, which registers a sequential backend.

Other parameters in `epx` are detailed in the documentation and allow for the tuning of the performance measure if appropriate, as well as some options regarding the filtering steps. These additional parameters all have defaults so we focus on the main arguments for the full example in this section.

3.2. An example from drug discovery

We demonstrate the **EPX** package via an example of QSAR modelling from drug discovery, where the activity of a chemical compound is related to the compound's molecular structure. In this case an active compound is the desired relevant, the rare class, and the compound's molecular structure are characterised by various descriptors. One such descriptor set of explanatory variables are Burden numbers (BN) (Burden, 1989). The BN descriptor set is one of multiple descriptor sets used in Tomal et al. (2015, 2016) and all its explanatory variables are continuous. Typically the data used to train an EPX model are of a size where parallel computing should be done to complete the phalanx-formation algorithm in a timely manner. This is because the computational complexity of phalanx formation is $O(d^2)$, where d is the number of initial phalanxes. For demonstration, we work with a training dataset of approximately 20% compounds randomly selected from the BN sample, which has 4946 compounds. In our training sample (`BNsample`) of size 1000, only 10 are active

compounds ($y = 1$) and the rest 990 are inactive compounds ($y = 0$). In our test sample (`BNhold`) of size 3946, only 38 are active compounds and the rest 3908 are inactive compounds. The R codes in lines $1 - 3$ below show how to build an ensemble of phalanxes using the `epx` function.

```
1 set.seed(761)
2 model <- epx(x = BNsample[,-25], y = BNsample[,25], classifier.args =
3              list(ntree = 150))
```

The developed ensemble of phalanxes is saved in `model` object. Note that since we use the default random forest as our base classifier, we set a seed for reproducibility. To reduce the computational burden on model development, we lowered the number of trees from the default 500 to 150. While the algorithm is running we get the following indication of progress in the console:

```
4 Performance measure: AHR
5 Performance measure additional arguments: none
6 Base classifier: random forest
7 Base classifier arguments specified in phalanx-formation:
8 $ntree
9 [1] 150
10 Phalanx formation is in progress, please wait
11 Reference distribution quantiles:
12 a0.95 = 0.0322050977060676
13 a0.50 = 0.0125942193317316
14 Number of deleted phalanxes after first filtering: 9 out of 24
15 Number of deleted phalanxes after final filtering: 0 out of 4
```

Lines $4 - 9$ summarise the performance metric and classifier settings used in the algorithm. Line 10 tells the user that the phalanx formation is in progress. Lines $11 - 13$ display the values of a_α ($\alpha = 0.95$ by default) and $a_{0.5}$ respectively as described in section 2.2. Finally, lines $14 - 15$ display the results of the two filtering steps as each stage is completed (the second and fourth step in Figure 1).

After about a minute, `epx` outputs an `S3` object of class "`epx`". We display the structure of the fitted model using `str(model)` as following.

```
16 > str(model)
17 List of 8
18  $ PHALANXES                  :List of 4
19   ..$ phalanxes.initial : num [1:24] 1 2 3 4 5 6 7 8 9 10 ...
20   ..$ phalanxes.filtered: num [1:24] 1 2 3 0 4 5 6 7 8 0 ...
21   ..$ phalanxes.merged  : num [1:24] 1 1 1 0 1 1 2 1 1 0 ...
22   ..$ phalanxes.final   : num [1:24] 1 1 1 0 1 1 2 1 1 0 ...
23  $ PHALANXES.FINAL.PERFORMANCE: num [1:4] 0.0845 0.1022 0.0646 0.101
24  $ PHALANXES.FINAL.FITS       : num [1:1000, 1:4] 0 0 0 0 0 0 0 0 0 0 ...
25   ..- attr(*, "dimnames")=List of 2
26   .. ..$ : NULL
27   .. ..$ : NULL
28  $ ENSEMBLED.FITS             : num [1:1000] 0 0 0 0 0 ...
29 > # 4 more items omitted
```

An "`epx`" object is a list starting with information regarding the phalanx membership of all the explanatory variables in all four steps of the phalanx-formation algorithm (lines 19 − 22). Every distinct positive integer is the "name" of a phalanx and so being assigned to 0 indicates that the variable does not belong to any phalanx at all. To see the variables belonging to each of the four steps of the phalanx-formation algorithm, use the following statement.

```
30 > model$PHALANXES
31 $phalanxes.initial
32  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
33 $phalanxes.filtered
34  [1]  1  2  3  0  4  5  6  7  8  0  9  0  10  0  11  0 12  0 13  0 14 0 15 0
35 $phalanxes.merged
36  [1] 1 1 1 0 1 1 2 1 1 0 1 0 1 0 3 0 2 0 1 0 2 0 4 0
37 $phalanxes.final
38  [1] 1 1 1 0 1 1 2 1 1 0 1 0 1 0 3 0 2 0 1 0 2 0 4 0
```

In our example, we begin with the default *initial phalanxes* where each vari-

able is in its own phalanx (line 32). After the filtering, explanatory variables 4, 10, 12, 14, 16, 18, 20, 22, 24 did not survive and are hence assigned 0 values for the *filtered phalanxes* (line 34). Note that those variables that did survive filtering are renumbered such that the maximum value in each vector is equal to the total number of phalanxes. After merging (line 36), the 15 phalanxes have been reduced to four *merged phalanxes*, and finally in the last step of the algorithm (line 38), phalanxes $1, 2, 3, 4$ survive final filtering, leaving a four *final phalanxes*. Other items in an "`epx`" object include

- `PHALANXES.FINAL.PERFORMANCE` (line 23): the performance of each of the final phalanxes according to the chosen performance measure.
- `PHALANXES.FINAL.FITS` (lines $24-27$): the $n \times p$ (p is the number of final phalanxes) matrix of estimated probabilities of relevance where the ith column is the estimated probabilities from phalanx i, which matches the matrix shown in Figure 1. Here, we have $n = 1000$ and $p = 4$.
- `ENSEMBLED.FITS` (line 28): an n-length vector of final estimated probabilities, obtained by averaging across the columns of `PHALANXES.FINAL.FITS`. This result is also shown in Figure 1 as rightmost column vector of the average of $\hat{\pi}_i$'s.

The four omitted items in an "`epx`" object consist of the data used to train the model, as well as information regarding base classifier and performance measure choices, which are required for other functions in the package described in the following sections.

3.3. Summary of results and plotting

Having fitted an EPX model, passing the resulting "`epx`" class object to the `summary` function provides an easily legible summary of the results of phalanx-formation along with the performance of each final phalanx (lines $39-59$). The first section (lines $40-42$) shows the number of variables used in the data, variables survived, and phalanxes produced. The second section (lines $44-59$)

shows each of the survived phalanxes, the constituent variables within each phalanx, and individual performances of the phalanxes.

```
> summary(model)
Phalanx-formation algorithm starts with 24 variable(s)
and ends with 15  variable(s) grouped into 4 phalanxes:
1 2 3 4
------
Phalanx 1 contains 10 variable(s):
WBN_GC_L_0.25, WBN_GC_H_0.25, WBN_GC_L_0.50, WBN_GC_L_0.75, WBN_GC_H_0.75,
WBN_GC_H_1.00, WBN_EN_L_0.25, WBN_EN_L_0.50, WBN_EN_L_0.75, WBN_LP_L_0.50
=> AHR is 0.08453896
------
Phalanx 2 contains 3 variable(s):
WBN_GC_L_1.00, WBN_LP_L_0.25, WBN_LP_L_0.75
=> AHR is 0.103273
------
Phalanx 3 contains 1 variable(s):
WBN_EN_L_1.00
=> AHR is 0.06456355
------
Phalanx 4 contains 1 variable(s):
WBN_LP_L_1.00
=> AHR is 0.1009757
```

Passing the "`epx`" object to the `plot` function generates a plot of the corresponding hit curve as shown in Figure 2. The hit curve is a step-function which increases by one step when one hit is found (plotted on the vertical axis). On the other hand, the numbers of hits found may remain fixed for a while as the number of shortlisted compounds grows (plotted on the horizontal axis). The hit curve shows the ranking performance of the model depending on when we cut-off the ranked list starting from the highest rank. A hit curve is superior to another hit curve if all its points lie above those of the other. Note that AHR is a numerical criterion (the larger the AHR, the better the ranking) that summarises a hit curve and in Figure 2, we see that the hit curve reflects the

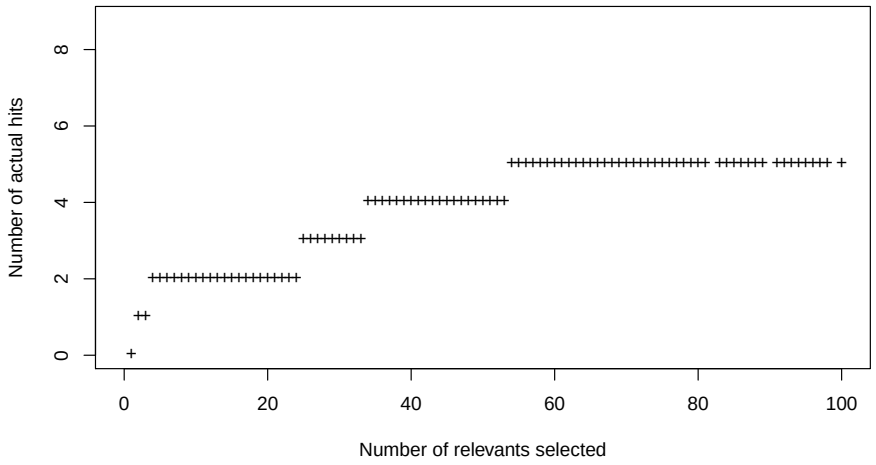

Figure 2: The hit curve is generated by calling `plot(model)`.

performance of the model fit by `epx`. We also provide a `hit.curve` function that generates a hit curve in a given vector of predicted probabilities of relevance and its corresponding true response vector.

3.4. Prediction

Passing an "`epx`" object to the `predict` function without providing data for the `newdata` argument simply returns the `ENSEMBLED.FITS` element of the object as demonstrated in lines $60 - 67$.

```
60 > preds0 <- predict(model)
61 Base classifier: random forest
62 Base classifier arguments specified in phalanx-formation:
63 $ntree
64 [1] 150
65 Base classifier arguments specified in prediction: none
66 > all.equal(preds0, model$ENSEMBLED.FITS)
67 [1] TRUE
```

When we provide some additional arguments for the classifier, `predict` will refit the classifiers for each phalanx as desired using the training data, thus resulting in different predictions than the original "`epx`" object's `ENSEMBLED.FITS` as shown in lines $68 - 78$.

```
68 > set.seed(761)
69 > preds <- predict(model, classifier.args = list(ntree = 500))
70 Base classifier: random forest
71 Base classifier arguments specified in phalanx-formation:
72 $ntree
73 [1] 150
74 Base classifier arguments specified in prediction:
75 $ntree
76 [1] 500
77 > all.equal(preds, model$ENSEMBLED.FITS)
78 [1] "Mean relative difference: 1.121363"
```

We can also provide new data using the usual `predict` function framework and `predict` will output a vector of estimated probabilities for the new data using the trained EPX model. Here we use the BN test data `BNhold` that were not included in `BNsample` training data. Note that as usual we can use the same base classifier arguments as those used in phalanx formation (lines $79 - 80$) or we may wish to make some changes as shown in lines $82 - 84$.

```
79 set.seed(761)
80 predshold0 <- predict(model, newdata = BNhold[,-25])
81
82 set.seed(761)
83 predshold <- predict(model, newdata = BNhold[,-25],
84                       classifier.args = list(ntree = 500))
```

3.5. Cross-validation

The `cv.epx` function performs a balanced $k-$fold cross-validation given an "`epx`" class object. Cross-validation is a way of assessing the performance of

the ensemble of phalanxes, and we use balanced cross-validation because we expect the data to have a rare class in each fold. The setup is as follows: the data with n observations is randomly divided into k approximately equal-sized groups (folds), where each of the k folds will have approximately $1/k$ of the relevant observations. We train an EPX model based on the given "`epx`" class object using $k-1$ of the folds, leaving one fold out to serve as the test set. This is repeated until all folds have served as a test set, at which point we will have final estimated probabilities of relevance for all n observations.

The `cv.epx` function by default performs a $10-$fold balanced cross-validation ($k = 10$) with observations randomly divided into folds. As with the `predict` function, `cv.epx` must take an "`epx`" object. Following is an example.

```
> set.seed(761)
> cv150 <- cv.epx(model)
Base classifier: random forest
Base classifier arguments specified in phalanx-formation:
$ntree
[1] 150
Base classifier arguments specified in balanced 10-fold cross-validation: none
```

The text in the console will indicate both the number of folds and any arguments made regarding the base classifier for cross-validation (line 89). This is similar to the behaviour of `predict` in that we still fit the same base classifier as that used during phalanx-formation, but we may change its arguments via the `classifier.args` argument as following.

```
> set.seed(761)
> cv500 <- cv.epx(model, classifier.args = list(ntree = 500))
Base classifier: random forest
Base classifier arguments specified in phalanx-formation:
$ntree
[1] 150
Base classifier arguments specified in balanced 10-fold cross-validation:
```

```
$ntree
[1] 500
```

The output of `cv.epx` by default is an $(n+1)\times(p+1)$ matrix, where n is the number of observations and p is the number of phalanxes. The ith column of the matrix is the predicted probabilities of relevance from the ith phalanx except for the last $(p+1)$th column, which has the predicted probabilities of relevance from the ensemble of phalanxes. The $(n+1)$th row of the matrix is the overall performance of the corresponding column, determined by the performance measure specified by the "`epx`" object (line 108). The following statement will show the *tail* probabilities and the overall performance measures for the phalanxes as well as for the ensemble.

```
> tail(cv500)
                     1         2          3          4  ensemble
            0.45000000 0.0760000 0.66800000 0.00400000 0.2995000
            0.00000000 0.0000000 0.00000000 0.00000000 0.0000000
            0.00000000 0.0000000 0.00000000 0.00000000 0.0000000
            0.07200000 0.0000000 0.00000000 0.00000000 0.0180000
            0.00000000 0.0000000 0.00000000 0.00000000 0.0000000
performance 0.07805713 0.1039471 0.05997207 0.09088224 0.1457195
```

Some additional arguments of `cv.epx` are briefly summarised below:

- `folds`: allows the use of custom folds by taking an n-length vector of positive integer values from 1 to k, such that the ith value in the vector indicates to which fold the ith observation should be assigned.
- `folds.out`: provides an n-length vector recording the fold membership for each observation; default is `FALSE`. Setting as `TRUE` reports the output of `cv.epx` into a list of two elements. The first being the $(n+1)\times(p+1)$ matrix attained from `cv.epx` by default, and the second is the $n-$length vector with integer values from 1 to k indicating to which fold each observation was shuffled for cross-validation.

3.6. Example with parallel computing

As the computational complexity of phalanx formation is high (for example, $O(d^2)$, where d is the number of initial phalanxes), we may need to run the `epx` function in parallel. The **EPX** package uses the **foreach** package (Revolution Analytics and Weston, 2020) to parallelise the loops where a_i, a_{ij}, and $a_{\overline{ij}}$ are calculated (i, j are phalanxes) using different nodes. In our package **EPX**, we use **doParallel** (Wallig et al., 2020) to register a parallel backend for the `%dopar%` function from **foreach**. The the general setup for parallel computing is as follows with uses mentioned in the comments.

```
clusters <- parallel::detectCores()      # detect the cores available
cl <- parallel::makeCluster(clusters)    # make clusters
doParallel::registerDoParallel(cl)       # register for parallel computing

########################################################################
############ Functions that use parallel computing run here ############
########################################################################

parallel::stopCluster(cl)                # close clusters
```

In the following example, we use the full BN dataset (4946 observations, 24 variables) and register 8 clusters to train the EPX model:

```
cl <- parallel::makeCluster(8)
doParallel::registerDoParallel(cl)

set.seed(761)
BN <- rbind(BNsample, BNhold)
model <- epx(x = BN[,-25], y = BN[,25], classifier.args = list(ntree = 150),
                    computing = "parallel")

parallel::stopCluster(cl)
```

We indicate to `epx` that we wish to compute in parallel via the `computing`

argument in line 122 – 124. Without specifying this argument, `epx` will override the clusters registered and compute sequentially.

4. Discussion of further capabilities

4.1. Additional base classifiers

Besides the default classifier random forest fitted via **randomForest** (Liaw and Wiener, 2002), the other base classifiers allowed in the `classifier` argument include logistic regression and neural networks. Logistic regression is fitted using the `glm` function from the **stats** package (R Core Team, 2020) (line 127). No additional arguments for *logistic regression* are needed. A single-hidden-layer *neural network* can also be specified as the base classifier and is fitted using the `nnet` function from the **nnet** package (Venables and Ripley, 2002). For neural network, we allow the number of units in the hidden layer to be set via `classifier.args` as demonstrated in lines 129 – 130. Other useful classifiers along with their arguments will be implemented in **EPX** in future versions.

```
127 model.log <- epx(x = BNsample[,-25], y = BNsample[,25], classifier = "logistic")
128
129 model.nn <- epx(x = BNsample[,-25], y = BNsample[,25], classifier = "neural",
130                 classifier.args = list(size = 2))
```

4.2. Additional performance measures

Apart from the default performance measure AHR, the three other measures that can be used are: initial enhancement (IE), TOP1, and rank last (RKL). All four metrics are available in **EPX** as separate functions named `AHR`, `IE`, `TOP1`, and `RKL`, respectively. Each function requires a vector of predicted probabilities of relevance for the argument `phat`, and a binary vector indicating the true response for all observations for the argument `y`. These metrics can be used to build the ensemble of phalanxes by providing argument `performance = "IE"` (for example) to `epx` function as defined in lines 2 – 3.

The metric Initial Enhancement (IE) is defined in Tomal et al. (2015, 2016), and since IE is a rescaling of the precision given a cutoff, it leads to similar conclusion (the larger the IE, the better the predictive ranking performance) as AHR. Though IE is often reported in QSAR studies, unlike AHR which is bounded by 1, IE is unbounded from above. Note that IE is the only performance metric in **EPX** where the user may specify an additional argument via `performance.args` in the functions `epx`, `predict`, and `cv.epx`. Specifically, the user may set their desired shortlist cutoff for IE.

TOP1 is a performance measure that produces a value of either 1 or 0. After sorting the observations by their predicted probabilities of relevance in decreasing order so the first ranked observation has the highest probability of relevance, if the first ranked observation is truly relevant, TOP1 has a value of 1. Otherwise TOP1 is 0. If there are ties for the first rank, all the corresponding observations must be relevant for TOP1 to score 1.

After ranking the observations as done for TOP1, RKL in contrast is the rank of the last relevant observation. Hence, RKL can take on integer values from 1 to n, where n is the total number of observations. If there are ties, the last object in the tied group determines RKL. That is, if all n objects are tied at the first rank but only one object is truly relevant, RKL will have a value of n. The smaller values of RKL indicates better predictive ranking.

Using the predictions attained in the Prediction section on the samples of the BN descriptor set withheld from `BNsample` (used to train the model), we calculate the four performance measures as follows:

```
> AHR(y = BNhold$y, phat = predshold)
[1] 0.2347868
> IE(y = BNhold$y, phat = predshold, cutoff = 50)
[1] 16.61474
> TOP1(y = BNhold$y, phat = predshold)
[1] 1
> RKL(y = BNhold$y, phat = predshold)
[1] 3946
```

The calculated IE of 16.62 (lines $133-134$) tells us that the model is performing 16.62 times better than a random ranking among the top ranked 50 compounds. The TOP1 of 1 (lines $135-136$) tells that the first compound ranked by the model is an active compound. The RKL of 3946 (lines $137-138$) tells us that the last active compound is ranked at position 3946.

5. Conclusion

Tomal et al. (2015, 2016) demonstrated how using an ensemble of phalanxes results in better predictive ranking of drug-like active biomolecules for development in drug discovery. Tomal et al. (2019) also demonstrated how the algorithm of phalanx formation can be used to split the feature variables of similarity scores of amino acid sequences to predict homologous protein in protein homology. That is, the ensemble of phalanxes attains better ranking using distinct statistical models trained on disjoint subsets of variables than the popular random forest, regularized random forest, balanced random forest and logistic regression model.

The **EPX** package demonstrates an implementation of the phalanx-formation algorithm and its related functions. Currently, **EPX** is prepared for similar use-cases as those shown in Tomal et al. (2015, 2016, 2019) when the aim is binary classification of a rare class of objects and performance is judged via ranking. This demonstration of **EPX** package will be useful for research in biomedicine. Future work will include the addition of other base classifiers (whenever found useful) and more notably, an adaptation for analogous regression and balanced classification problems. The package **EPX** will be updated on a continuous basis as more features are made available via research findings and publications.

Acknowledgments

The **EPX** package is funded by the Natural Sciences and Engineering Research Council of Canada and the Department of Statistics of the University of British Columbia.

Conflicts of interest statement

The authors declare no conflicts of interest.

References

Breiman, L., 2001. Random forests. Machine Learning 45, 5–32. doi:10.1023/A:1010933404324.

Burden, F.R., 1989. Molecular identification number for substructure searches. Journal of Chemical Information and Computer Sciences 29, 225–227. doi:10.1021/ci00063a011.

Liaw, A., Wiener, M., 2002. Classification and regression by randomforest. R News 2, 18–22. URL: http://CRAN.R-project.org/doc/Rnews/.

R Core Team, 2020. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing. Vienna, Austria. URL: https://www.R-project.org/.

Revolution Analytics, Weston, S., 2020. foreach: Provides Foreach Looping Construct for R. URL: https://CRAN.R-project.org/package=foreach. R package version 1.5.0.

Tomal, J.H., Welch, W.J., Zamar, R.H., 2015. Ensembling Classification Models Based on Phalanxes of Variables with Applications in Drug Discovery. The Annals of Applied Statistics 9, 69–93. doi:10.1214/14-AOAS778.

Tomal, J.H., Welch, W.J., Zamar, R.H., 2016. Exploiting multiple descriptor sets in qsar studies. Journal of Chemical Information and Modeling 56, 501–509. doi:10.1021/acs.jcim.5b00663.

Tomal, J.H., Welch, W.J., Zamar, R.H., 2019. Ensembles of phalanxes across assessment metrics for robust ranking of homologous proteins. arXiv:1706.06971.

Venables, W.N., Ripley, B.D., 2002. Modern Applied Statistics with S. Fourth ed., Springer, New York. URL: http://www.stats.ox.ac.uk/pub/MASS4. iSBN 0-387-95457-0.

Wallig, M., Microsoft Corporation, Weston, S., Tenenbaum, D., 2020. doParallel: Foreach Parallel Adaptor for the 'parallel' Package. URL: https://CRAN.R-project.org/package=doParallel. R package version 1.0.16.

Wang, M., 2005. Statistical Methods for High Throughput Screening Drug Discovery Data. Ph.D. thesis. University of Waterloo. URL: http://etd.uwaterloo.ca/etd/y32wang2005.pdf.