A demonstration of the RaSEn package

Ye Tian and Yang Feng

2022-02-18

We provide a detailed demo of the usage for the `RaSEn` package. This package implements the random subspace ensemble classification (RaSE) method (Tian and Feng (2021b)), the variable screening approach via RaSE (Tian and Feng (2021a)) and the super RaSE method (Zhu and Feng (2021)).

Random Subspace Ensemble Classification

Introduction

Suppose we have training data $\{\boldsymbol{x}_i, y_i\}_{i=1}^n \in \{\mathbb{R}^p, \{0,1\}\}$, where each $\boldsymbol{x}_i$ is a $1 \times p$ vector.

Based on training data, RaSE algorithm aims to generate B_1 weak learners $\{C_n^{S_j}\}_{j=1}^{B_1}$, each of which is constructed in a feature subspace $S_j \subseteq \{1, ..., p\}$ instead using all p features. To obtain each weak learner, B_2 candidates $\{C_n^{S_{jk}}\}_{k=1}^{B_2}$ are trained based in subspaces $\{S_{jk}\}_{k=1}^{B_2}$, respectively. To choose the optimal one among these B_2 candidates, some criteria need to be applied, including minimizing ratio information criterion (RIC, Tian and Feng (2021b)), minimizing extended Bayes information criterion (eBIC, Chen and Chen (2008), Chen and Chen (2012)), minimizing the training error, minimizing the validation error (if validation data is available), minimizing the cross-validation error, minimizing leave-one-out error etc. And the type of weak learner can be quite flexible.

To better adapt RaSE into the sparse setting, we can update the distribution of random feature subspaces according to the selected percentage of features in B_1 subspaces in each round. This can be seen as an adaptive strategy to increase the possibility to cover the signals that contribute to our model, which can improve the performance of RaSE classifiers in sparse settings.

The selected percentage of each of p features in B_1 subspaces can be used for feature ranking as well. And we could plot the selected percentage to intuitively rank the importance of each feature in a RaSE model.

Installation

`RaSEn` can be installed from CRAN.

```
install.packages("RaSEn", repos = "http://cran.us.r-project.org")
```

Then we can load the package:

```
library(RaSEn)
```

How to Fit a RaSE Classifier for Prediction

We will show in this section how to fit RaSE classifiers based on different types of base classifiers. First we generate the data from a binary guanssian mixture model (model 1 in Tian and Feng (2021b))

$$\boldsymbol{x} \sim (1-y)N(\boldsymbol{\mu}^{(0)}, \Sigma) + yN(\boldsymbol{\mu}^{(1)}, \Sigma),$$

where $\boldsymbol{\mu}^{(0)}, \boldsymbol{\mu}^{(1)}$ are both $1 \times p$ vectors, Σ is a $p \times p$ symmetric positive definite matrix. Here y follows a bernoulli distribution:

$$y \sim Bernoulli(\pi_1),$$

where $\pi_1 \in (0,1)$ and we denote $\pi_0 = 1 - \pi_1$.

Here we follow from the setting of Mai, Zou, and Yuan (2012), letting $\Sigma = (0.5^{|i-j|})_{p\times p}, \boldsymbol{\mu}^{(0)} = \mathbf{0}_{p\times 1}, \boldsymbol{\mu}^{(1)} = \Sigma^{-1} \times 0.556(3, 1.5, 0, 0, 2, \mathbf{0}_{1\times(p-5)})^T$. Let $n = 100, p = 50$. According to the definition of minimal discriminative set in Tian and Feng (2021b), here the minimal discriminative set $S^* = \{1, 2, 5\}$, which contribute to the classification.

Apply function `RaModel` to generate training data and test data of size 100 with dimension 50.

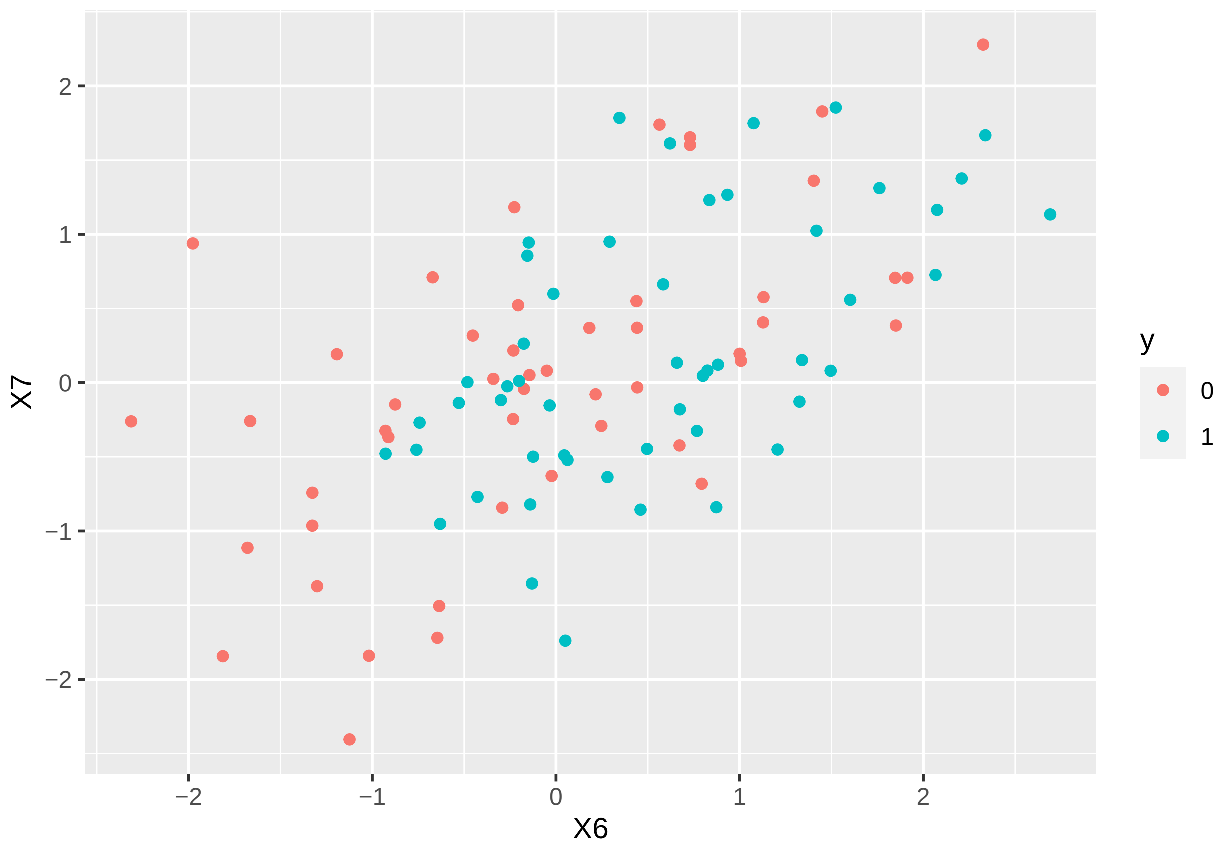
```
set.seed(0, kind = "L'Ecuyer-CMRG")
train.data <- RaModel("classification", 1, n = 100, p = 50)
test.data <- RaModel("classification", 1, n = 100, p = 50)
xtrain <- train.data$x
ytrain <- train.data$y
xtest <- test.data$x
ytest <- test.data$y
```

We can visualize the first two dimensions or feature 1 and 2 as belows:

```
library(ggplot2)
ggplot(data = data.frame(xtrain, y = factor(ytrain)), mapping = aes(x = X1,
    y = X2, color = y)) + geom_point()
```

Similarly, we can also visualize the feature 6 and 7:

```
ggplot(data = data.frame(xtrain, y = factor(ytrain)), mapping = aes(x = X6, y = X7,
    color = y)) + geom_point()
```



It's obvious to see that in dimension 1 and 2 the data from two classes are more linearly seperate than in dimension 6 and 7. Then we call `Rase` function to fit the RaSE classifier with LDA, QDA and logistic regression base classifiers with criterion of minimizing RIC and RaSE classifier with knn base classifier with criterion of minimizing leave-one-out error. To use different types of base classifier, we set `base` as "lda," "qda," "knn" and "logistic," repectively. `B1` is set to be 100 to generate 100 weak learners and `B2` is set to be 100 as well to generate 100 subspace candidates for each weak learner. Without using iterations, we set `iteration` as 0. `criterion` is set to be "ric" for RaSE classifier with LDA, QDA and logistic regression while it is "loo" for RaSE classifier with knn base classifier. To speed up the computation, we apply parallel computing with 2 cores by setting `cores = 2`.

```
fit.lda <- RaSE(xtrain, ytrain, B1 = 100, B2 = 50, iteration = 0, base = "lda",
    cores = 2, criterion = "ric")
fit.qda <- RaSE(xtrain, ytrain, B1 = 100, B2 = 50, iteration = 0, base = "qda",
    cores = 2, criterion = "ric")
fit.knn <- RaSE(xtrain, ytrain, B1 = 100, B2 = 50, iteration = 0, base = "knn",
    cores = 2, criterion = "loo")
fit.logistic <- RaSE(xtrain, ytrain, B1 = 100, B2 = 50, iteration = 0,
    base = "logistic", cores = 2, criterion = "ric")
```

We can print the summarized results of RaSE model by calling `print` function. For instance, we print the RaSE model with LDA base classifier:

```
print(fit.lda)
```

```
## Marginal probabilities:
## class 0 class 1
```

```
##    0.49    0.51
## Type of base classifiers: lda
## Criterion: ric
## B1: 100
## B2: 50
## D: 10
## Cutoff: 0.5589305
## Selected percentage of each feature appearing in B1 subspaces:
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
## 100  25  12  19  49  22  16   8  11  15  16  13  16  15   7  10  10  15  15  18
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
##  18  15  19  13  11   9  14  11   8   8  16  12  19  18   7   4  14  16  19   9
##  41  42  43  44  45  46  47  48  49  50
##  14  16  14  13  11   7  12  12  13  14
```

To evaluate the performance of four different models, we calculate the test error on test data:

```
er.lda <- mean(predict(fit.lda, xtest) != ytest)
er.qda <- mean(predict(fit.qda, xtest) != ytest)
er.knn <- mean(predict(fit.knn, xtest) != ytest)
er.logistic <- mean(predict(fit.logistic, xtest) != ytest)
cat("LDA:", er.lda, "QDA:", er.qda, "knn:", er.knn, "logistic:", er.logistic)
```

```
## LDA: 0.11 QDA: 0.13 knn: 0.12 logistic: 0.09
```

And the output of `Rase` function is an object belonging to S3 class "RaSE." It contains:

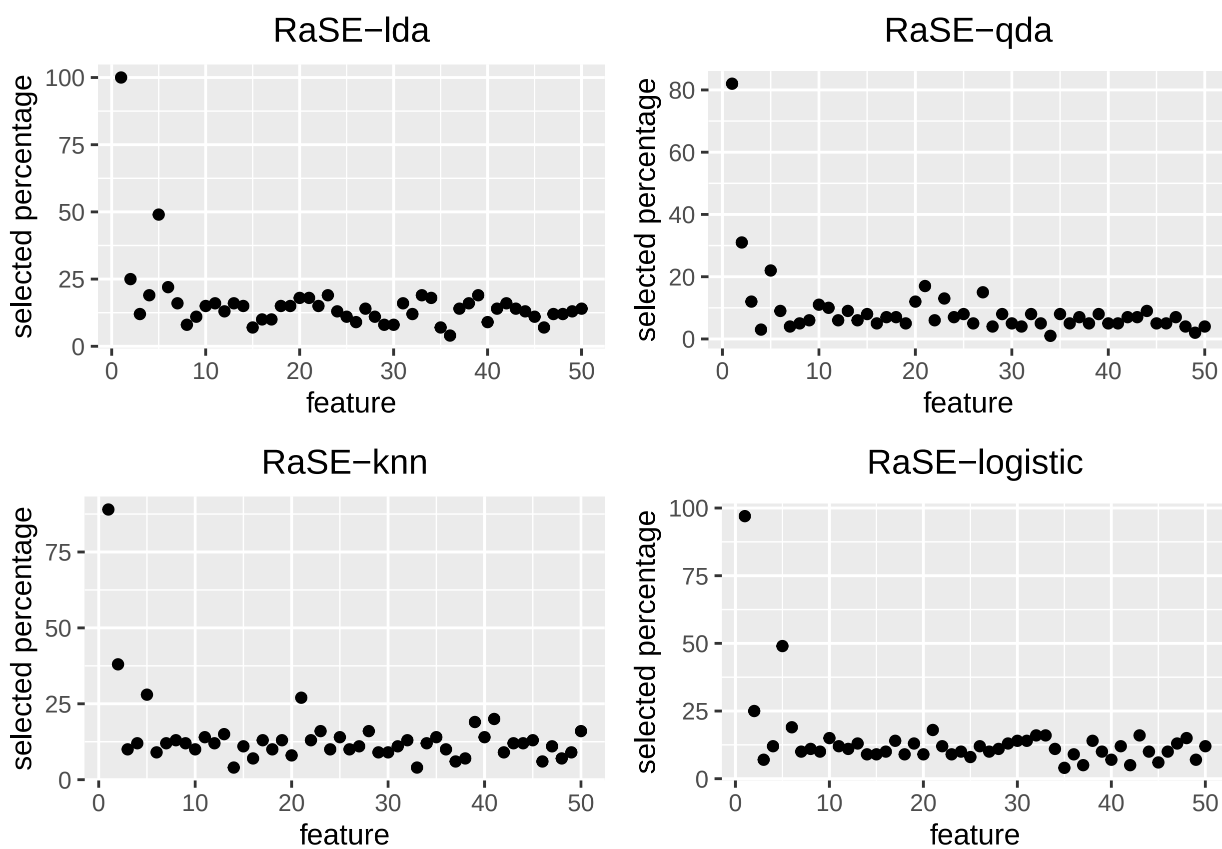
- marginal: the marginal probability for each class.
- base: the type of base classifier.
- criterion: the criterion to choose the best subspace for each weak learner.
- B1: the number of weak learners.
- B2: the number of subspace candidates generated for each weak learner.
- D: the maximal subspace size when generating random subspaces.
- iteration: the number of iterations.
- fit.list: a list of B1 fitted base classifiers.
- cutoff: the empirically optimal threshold.
- subspace: a list of subspaces correponding to B1 weak learners.
- ranking: the selected percentage of each feature in B1 subspaces.
- scale: a list of scaling parameters, including the scaling center and the scale parameter for each feature.

How to Use RaSE for Feature Ranking

The selected percentage of features in B_1 subspaces for four RaSE classifiers are contained in the output, which can be used for feature ranking. We can plot them by using `RaPlot` function:

```r
library(gridExtra)
plot_lda <- RaPlot(fit.lda)
plot_qda <- RaPlot(fit.qda)
plot_knn <- RaPlot(fit.knn)
plot_logistic <- RaPlot(fit.logistic)

grid.arrange(plot_lda, plot_qda, plot_knn, plot_logistic, ncol = 2)
```



From four figures, it can be noticed that feature 1, 2 and 5 obtain high selected percentage among all $p = 50$ features under LDA, QDA and kNN models, implying their importance in classification model. We can set a positive iteration number to increase the selected percentage of three signals among B_1 subspaces, which may improve the performance.

Super RaSE

In RaSE, we consider only a single type of base classifiers (e.g. LDA, QDA, kNN, etc.). Zhu and Feng (2021) extends the idea of RaSE by combining classifiers of different types. For each of the B_1B_2 weak learners, the base classifier type is sampled randomly from some given types with corresponding probabilities. For iterative super RaSE, not only the feature sampling probability will be updated based on the feature selected frequencies, but the classifier type sampling probability will be updated according to the type selected frequencies as well. Note that here the feature sampling probability is updated on the basis of feature frequencies in the last iteration for corresponding base classifier type. The user can also fix the classifier type sampling probability. It can be controled by component `base.update` in parameter `super`. The super RaSE will be fitted when the parameter `base` is a string vector of base classifier types or a numeric probability vector with classifier type names. In the first case, the base classifier type will be sampled uniformly, while in the second case, it will be sampled according to the provided probability.

The following example shows how to fit a super RaSE classifier which mixes kNN, LDA and logistic regression.

```
fit.super <- RaSE(xtrain = xtrain, ytrain = ytrain, B1 = 100, B2 = 50,
    base = c("knn", "lda", "logistic"), super = list(type = "separate",
        base.update = T), criterion = "cv", cv = 5, iteration = 0, cores = 2)
ypred <- predict(fit.super, xtest)
mean(ypred != ytest)
```

```
## [1] 0.1
```

We can look at the sampling percentage of each feature from each classifier type and the base classifier type selected percentages.

```
fit.super$ranking.feature
```

```
##                 1        2        3         4        5        6         7
## knn      75.60976 56.09756 17.07317 17.073171 26.82927 12.19512  2.439024
## lda      87.50000 34.37500 15.62500  9.375000 34.37500 25.00000 18.750000
## logistic 85.18519 37.03704 18.51852  7.407407 29.62963  0.00000  7.407407
##                 8         9       10       11        12       13        14
## knn      12.19512 14.634146 12.19512 21.95122  7.317073 14.63415  9.756098
## lda       9.37500  9.375000  6.25000 15.62500 12.500000 21.87500 12.500000
## logistic 18.51852  7.407407 11.11111 14.81481  7.407407 14.81481 14.814815
##                15       16        17       18        19        20       21
## knn      17.07317 12.19512 24.390244 17.07317  9.756098 14.634146 34.14634
## lda       0.00000 15.62500  9.375000  6.25000  9.375000  9.375000 12.50000
## logistic 11.11111 14.81481  7.407407  0.00000 18.518519  7.407407 25.92593
##                22       23        24       25       26       27       28
## knn      14.63415 26.82927  7.317073 14.63415 9.756098 7.317073 4.878049
## lda      18.75000 12.50000 12.500000  6.25000 6.250000 9.375000 3.125000
## logistic 11.11111 18.51852  7.407407 14.81481 3.703704 3.703704 7.407407
##                29       30        31       32        33       34        35
## knn      12.19512 2.439024 14.634146 9.756098  9.756098 7.317073  4.878049
## lda      21.87500 0.000000  3.125000 6.250000 12.500000 9.375000  6.250000
## logistic 11.11111 7.407407  3.703704 7.407407  3.703704 0.000000 11.111111
##                 36       37        38        39        40       41        42
## knn      17.073171 12.19512 17.073171 17.073171 17.073171 14.63415  2.439024
## lda      21.875000 12.50000 18.750000 15.625000  9.375000  6.25000 12.500000
## logistic  3.703704 11.11111  3.703704  3.703704  3.703704 14.81481 14.814815
##                43        44        45       46       47       48       49
## knn      7.317073  9.756098  7.317073 12.19512 12.19512 17.07317 14.63415
## lda      9.375000 18.750000 15.625000  9.37500  6.25000 18.75000 12.50000
## logistic 7.407407 22.222222 18.518519  0.00000 14.81481 18.51852 29.62963
##                50
## knn      14.63415
## lda      12.50000
## logistic 14.81481
```

```
fit.super$ranking.base
```

```
##      knn      lda logistic
##       41       32       27
```

Variable Screening via Random Subspace Ensemble

In this section, we describe how to apply RaSE for variable screening.

Introduction

We follow the aforementioned notations. Although Tian and Feng (2021b) only discusses the classification problem, RaSE framework can be imediately applied for continuous response with no extra effort. As before, we would like to select B_1 subspaces $\{S_{j*}\}_{j=1}^{B_1}$, for each of which B_2 candidate subspaces $\{S_{jk}\}_{k=1}^{B_2}$ are generated. Some specific criterion is required for subspace selection. The selected percentage of features in $\{S_{j*}\}_{j=1}^{B_1}$ can be used for variable screening (Tian and Feng (2021a)).

Variable screening via RaSE

We will present how to do variable screening through RaSE framework in this section. First we generate the data from the following model (model 1 in Tian and Feng (2021b), model II in Fan and Lv (2008)).

$$y = 5x_1 + 5x_5 + 5x_3 - \frac{15}{\sqrt{2}}x_4 + \epsilon,$$

where $\boldsymbol{x} = (x_1, \ldots, x_p)^T \sim N(\boldsymbol{0}, \Sigma)$, $\Sigma = (\sigma_{ij})_{p\times p}$, $\sigma_{ij} = 0.5^{\mathbb{1}(i \neq j)}$, $\epsilon \sim N(0, 1)$, and $\epsilon \perp\!\!\!\perp \boldsymbol{x}$. The signal set $S^* = \{1, 2, 3, 4\}$.

Let $n = 100$ and $p = 100$. Call function `RaModel` to generate the data.

```
train.data <- RaModel("screening", 1, n = 100, p = 100)
xtrain <- train.data$x
ytrain <- train.data$y
```

As Tian and Feng (2021b) describes, here x_4 is marginally independent of y. We first apply RaSE equipped with linear regression model and BIC by calling function `RaScreen`. Set `B1` = `B2` = 100, `model` = `lm` and `criterion` = `bic`. To demonstrate the power of iterations, we set `iteration` = 1.

```
fit.bic <- RaScreen(xtrain, ytrain, B1 = 100, B2 = 100, model = "lm", criterion = "bic",
    cores = 2, iteration = 1)
```

The output of `RaScreen` is a list including the selected percentage of variables, the model we use and other information. Note that the selected percentage of variables are stored in a list (when `iteration` = 0, it is a vector) called "selected.perc." All results from different iteration rounds are available. Function `RaRank` provides a convenient and automatic way to select variables from the output of `RaScreen`. We set `selected.num` = `n/logn` to select $[n/\log n] = 21$ variables. Let's compare the results from vanilla RaSE (no iteration) and RaSE with 1 interation round as follows.

```
RaRank(fit.bic, selected.num = "n/logn", iteration = 0)
```

```
##  [1]   1   2   3   4  11  22  44  49 100  27  80  10  15  18  52  59  93  95  17
## [20]  36  38
```

```
RaRank(fit.bic, selected.num = "n/logn", iteration = 1)
```

```
##  [1]  1  2  4  3 11 44 49 71 22 38 52 59 66 18 58 83 27 28 68 15 64
```

Observe that RaSE with linear regression model and BIC captures the signals very well. Interation indeed improves vanilla RaSE by ranking four signals on the top.

Note that there could be some variables with zero selected percentage, especially in iterative RaSE. In this case, such variables are indistinguishable. When the user requests more variables beyond the number of variables with positive selected percentages, `RaRank` function will randomly sample from variables with zero selected percentage and pop up a warning message as below.

```
RaRank(fit.bic, selected.num = "n-1", iteration = 1)
```

```
## Warning in RaRank(fit.bic, selected.num = "n-1", iteration = 1): Only 94
## variables have positive selected percentage but request 99 ones. The last 5
## variables are randomly sampled!
```

```
##  [1]   1   2   4   3  11  44  49  71  22  38  52  59  66  18  58  83  27  28  68
## [20]  15  64  73  93  95  10  16  17  21  97   8   9  19  24  29  40  63  80  94
## [39] 100   5  12  34  36  37  42  54  61  78  81  84  90  96   6  25  31  39  41
## [58]  50  51  53  56  57  60  65  74  76  79  92   7  14  26  33  47  48  69  72
## [77]  77  85  86  87  13  30  46  70  75  82  98  23  32  43  45  55  89  91  99
## [96]  35  67  20  62
```

Reference

Chen, Jiahua, and Zehua Chen. 2008. "Extended Bayesian Information Criteria for Model Selection with Large Model Spaces." *Biometrika* 95 (3): 759–71.

———. 2012. "Extended BIC for Small-n-Large-p Sparse GLM." *Statistica Sinica*, 555–74.

Fan, Jianqing, and Jinchi Lv. 2008. "Sure Independence Screening for Ultrahigh Dimensional Feature Space." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 70 (5): 849–911.

Mai, Qing, Hui Zou, and Ming Yuan. 2012. "A Direct Approach to Sparse Discriminant Analysis in Ultra-High Dimensions." *Biometrika* 99 (1): 29–42.

Tian, Ye, and Yang Feng. 2021a. "RaSE: A Variable Screening Framework via Random Subspace Ensembles." *Journal of the American Statistical Association*, no. just-accepted: 1–30.

———. 2021b. "RaSE: Random Subspace Ensemble Classification." *Journal of Machine Learning Research* 22 (45): 1–93.

Zhu, Jianan, and Yang Feng. 2021. "Super RaSE: Super Random Subspace Ensemble Classification." https://www.preprints.org/manuscript/202110.0042.