ssc: An R Package for Semi-Supervised Classification

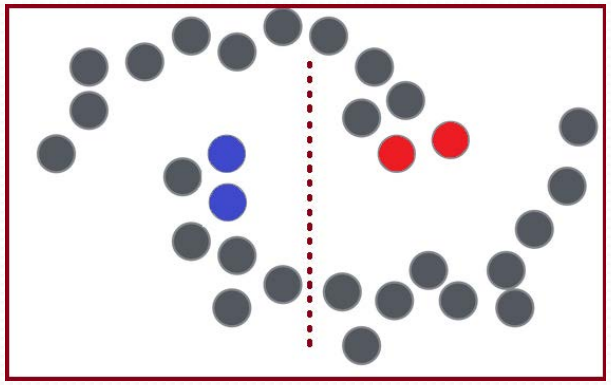
Authors:

Mabel González
Osmani Rosado
José D. Rodríguez
Christoph Bergmeir
Isaac Triguero
José M. Benítez

Semi-Supervised Classification

Training a classifier using:

- Labeled examples
- Unlabeled examples



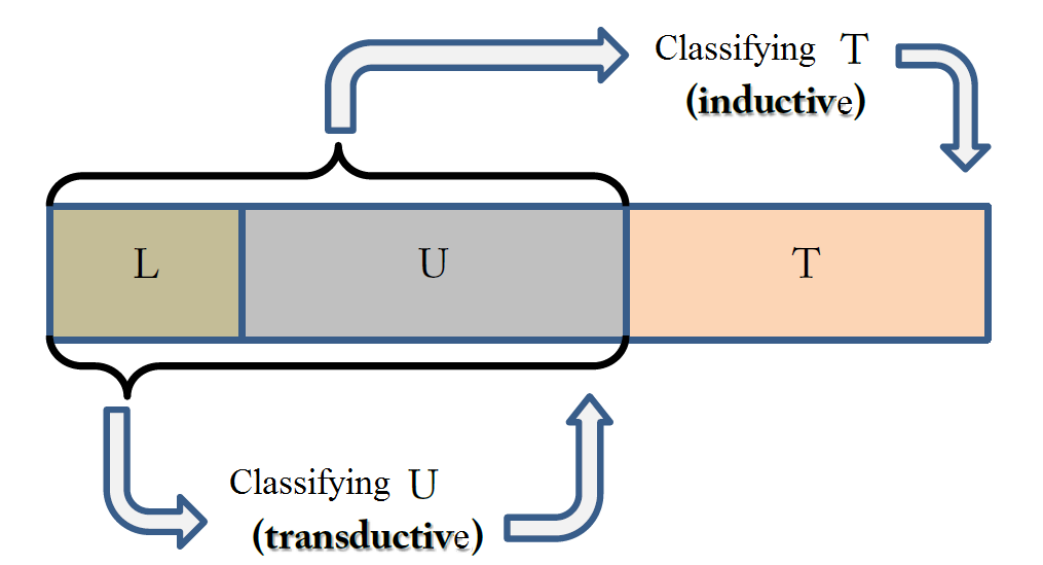
Application domains:

- Web page classification
- Speech recognition
- Bioinformatics
- …

Semi-Supervised Classification

Two main settings for classification:

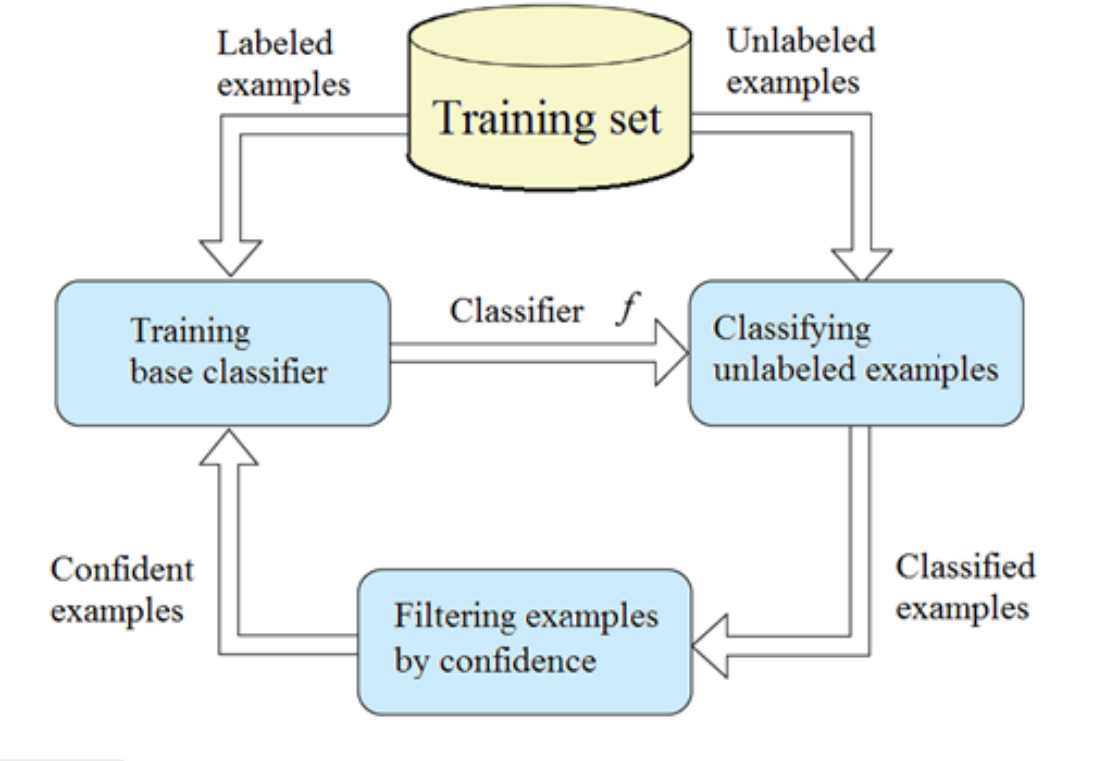
- **Transductive**: Predict classes for unlabeled data
- **Inductive**: Learn a classification model



Self-labeled methods

Self-labeled methods obtain an enlarged labeled set by the iterative classification of unlabeled examples.

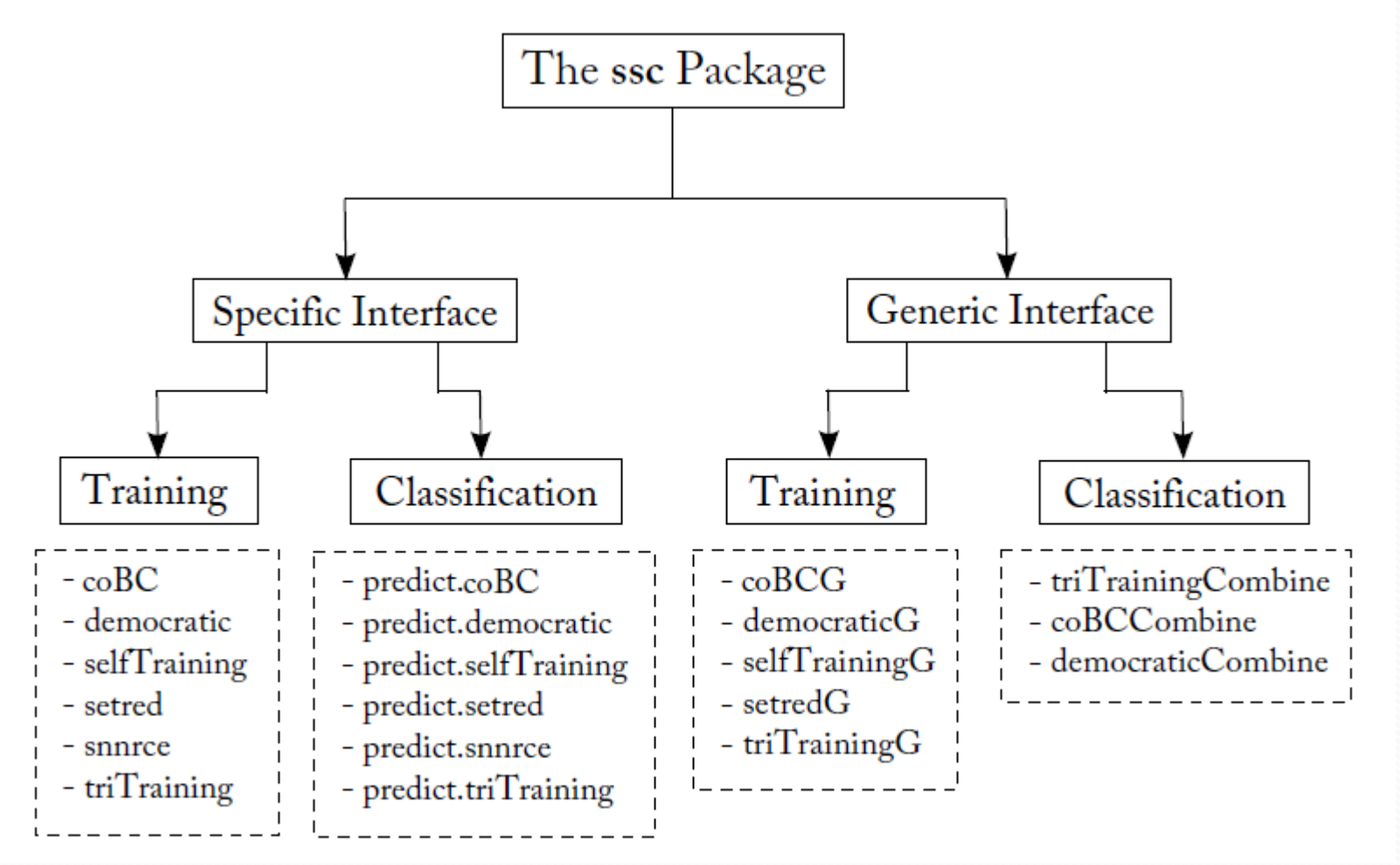
Self-training Process



Methods implemented in ssc

Method	Addition mechanism	Classifiers	Learning paradigm	Teaching
Self-training	incremental	single	single	self
SETRED	amending	single	single	self
SNNRCE	amending	single	single	self
Tri-training	incremental	multi	single	mutual
Co-Bagging	incremental	multi	single	mutual
Democratic-Co	incremental	multi	multi	mutual

Main functionalities in ssc



Setting up the data

```
library(ssc)
data(wine) # load the Wine dataset

cls <- which(colnames(wine) == "Wine")
x <- wine[, -cls] # instances without classes
y <- wine[, cls] # the classes
x <- scale(x) # scale the attributes for distance calculations
set.seed(3)

# Use 50% of instances for training
tra.idx <- sample(x = length(y), size = ceiling(length(y) * 0.5))
xtrain <- x[tra.idx,] # training instances
ytrain <- y[tra.idx] # classes of training instances

# Use 70% of train instances as unlabeled set
tra.na.idx <- sample(x = length(tra.idx),
                     size = ceiling(length(tra.idx) * 0.7))
ytrain[tra.na.idx] <- NA # remove class of unlabeled instances
```

Training the model

- From the raw instances

```
library(caret)
m.selft1 <- selfTraining(x = xtrain, y = ytrain, learner = knn3,
                         learner.pars = list(k = 1), pred = "predict")
```

- From a distance matrix

```
dtrain <- as.matrix(proxy::dist(x = xtrain, method = "euclidean", by_rows = TRUE))
m.selft2 <- selfTraining(x = dtrain, y = ytrain, x.inst = FALSE, learner = oneNN,
                         pred = "predict", pred.pars = list(type = "prob"))
```

- From a kernel matrix

```
library(kernlab)
ktrain <- as.matrix(exp(- 0.048 * dtrain^2))
m.selft3 <- selfTraining(x = ktrain, y = ytrain, x.inst = FALSE, learner = ksvm,
                         learner.pars = list(kernel = "matrix", prob.model = TRUE),
                         pred = function(m, k)
                           predict(m, as.kernelMatrix(k[, SVindex(m)]),
                                   type = "probabilities"))
```

Training with Democratic-Co

- Using three different learning schemes

```
library(caret)
library(kernlab)
library(C50)
m.demo <- democratic(x = xtrain, y = ytrain, learners = list(knn3, ksvm, C5.0),
                     learners.pars = list(list(k=1), list(prob.model = TRUE), NULL),
                     preds = list(predict, predict, predict), preds.pars =
                       list(NULL, list(type = "probabilities"), list(type = "prob"))
)
```

Transductive classification

- Setting up the testing set

```
# Use the unlabeled examples for transductive test
xttest <- x[tra.idx[tra.na.idx],] # transductive test instances
yttest <- y[tra.idx[tra.na.idx]] # classes of instances in xttest
```

- Performing transductive classification

```
p.selft3transd <- predict(m.selft3, as.kernelMatrix(ktrain[tra.na.idx,
                                                    m.selft3$instances.index]))
```

Inductive classification

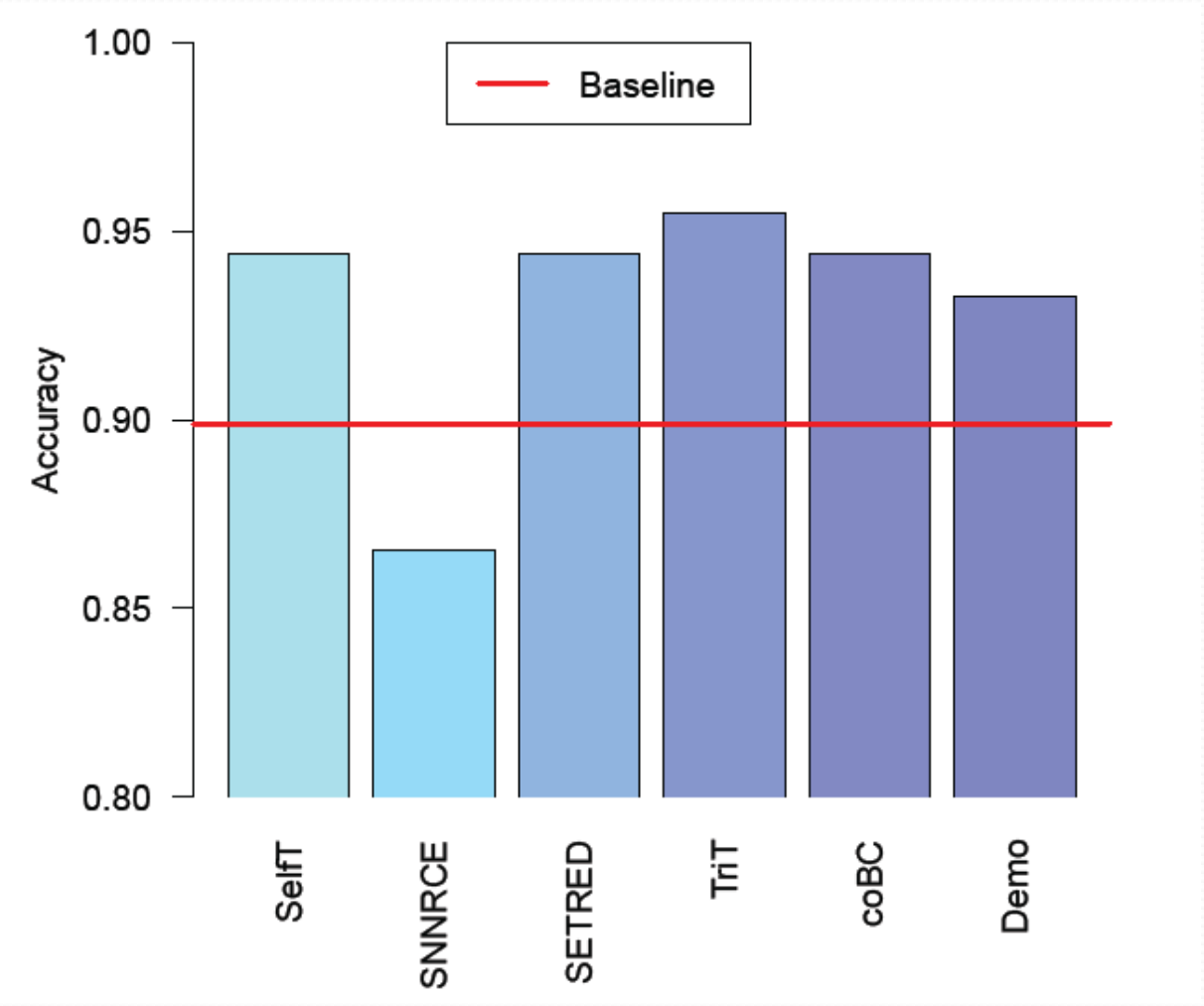
- Setting up the testing set

```
# Use the other 50% of instances for inductive test
tst.idx <- setdiff(1:length(y), tra.idx)
xitest <- x[tst.idx,] # test instances
yitest <- y[tst.idx] # classes of instances in xitest
```

- Performing inductive classification

```
p.selft1 <- predict(m.selft1, xitest)
p.selft2 <- predict(m.selft2, ditest[, m.selft2$instances.index])
p.selft3 <- predict(m.selft3, as.kernelMatrix(kitest[, m.selft3$instances.index]))
p.demo <- predict(m.demo, xitest)
```

Inductive classification of the wine problem



Comparison with the supervised paradigm

- Setting up the supervised training set

```
labeled.idx <- which(!is.na(ytrain))# indices of the initially labeled instances
xilabeled <- xtrain[labeled.idx,] # labeled instances
yilabeled <- ytrain[labeled.idx] # related classes
```

- Training a supervised classifier SVM (baseline)

```
svmBL <- ksvm(x = xilabeled, y = yilabeled, prob.model = TRUE) # build SVM
p.svmBL <- predict(object = svmBL, newdata = xitest) # classify with SVM
```

Empirical evaluation

Datasets	SVM	selfTraining	setred	coBC	triTraining
Iris	0,68	0,88	0,88	0,88	0,90
Parkinsons	0,86	0,86	0,87	0,87	0,86
Wine	0,95	0,97	0,97	0,95	0,97
Vertebral column	0,77	0,75	0,77	0,78	0,78
Fertility	0,90	0,90	0,90	0,72	0,90

SSC > supervised baseline 11

SSC = supervised baseline 7

SSC < supervised baseline 2

Conclusions (1/2)

- The implemented techniques in the R package ssc can take advantage of partially labeled datasets to create a classifier.
- The classifiers obtained can be used to perform either transductive or inductive classification.
- The ssc package offers a wrapper framework to train models from instances or directly from a precomputed distance or kernel matrix.

Conclusions (2/2)

- The ssc package supports a generic interface for base classifiers with other specifications, increasing the flexibility of this approach.

- The experimental results have shown that these techniques can provide better results than supervised classification at low ratios of labeled data.