Package 'SSL'

May 14, 2016

Type Package

Title Semi-Supervised Learning

Version 0.1

Date 2016-05-01

Author Junxiang Wang

Maintainer Junxiang Wang <xianggebenben@163.com>

Description Semi-supervised learning has attracted the attention of machine learning community because of its high accuracy with less annotating effort compared with supervised learning.The question that semi-supervised learning wants to address is: given a relatively small labeled dataset and a large unlabeled dataset, how to design classification algorithms learning from both ? This package is a collection of some classical semi-supervised learning algorithms in the last few decades.

License GPL (>= 3)

LazyData TRUE

RoxygenNote 5.0.1

Depends R (>= 3.2)

Imports NetPreProc (>= 1.1), Rcpp (>= 0.12.2), caret (>= 6.0-52), proxy (>= 0.4-15), xgboost (>= 0.4), klaR (>= 0.6-12), e1071 (>= 1.6-7), stats (>= 3.2)

LinkingTo Rcpp

NeedsCompilation yes

Repository CRAN

Date/Publication 2016-05-14 23:12:09

R topics documented:

`sslCoTrain` *Co-Training*

Description

Co-Training

Usage

```
sslCoTrain(xl, yl, xu, method1 = "nb", method2 = "nb", nrounds1, nrounds2,
  portion = 0.5, n = 10, seed = 0, ...)
```

Arguments

`xl`	a n * p matrix or data.frame of labeled data
`yl`	a n * 1 integer vector of labels.
`xu`	a m * p matrix or data.frame of unlabeled data
`method1, method2`	a string which specifies the first and second classification model to use.xgb means extreme gradient boosting,please refer to `xgb.train`.For other options,see more in `train`.
`nrounds1, nrounds2`	parameter needed when `method1` or `method2` =xgb. See more in `xgb.train`
`portion`	the percentage of data to split into two parts.
`n`	the number of unlabeled examples to add into label data in each iteration.
`seed`	an integer specifying random number generation state for data split
`...`	other parameters

Details

`sslCoTrain` divides labeled data into two parts ,each part is trained with a classifier, then it chooses some unlabeled examples for prediction and adds them into labeled data. These new labeled data help the other classifer improve performance.

Value

a m * 1 integer vector representing the predictions of unlabeled data.

Author(s)

Junxiang Wang

References

Blum, A., & Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. COLT: Proceedings of the Workshop on Computational Learning Theory.

See Also

`train` `xgb.train`

Examples

```
data(iris)
xl<-iris[,1:4]
#Suppose we know the first twenty observations of each class
#and we want to predict the remaining with co-training
# 1 setosa, 2 versicolor, 3 virginica
yl<-rep(1:3,each=20)
known.label <-c(1:20,51:70,101:120)
xu<-xl[-known.label,]
xl<-xl[known.label,]
yu<-sslCoTrain(xl,yl,xu,method1="xgb",nrounds1 = 100,method2="xgb",nrounds2 = 100,n=60)
```

---

sslGmmEM *Gaussian Mixture Model with an EM Algorithm*

Description

`sslGmmEM` implements Gaussian Mixture Model with an EM algorithm, and weights the unlabeled data by introducing lambda-EM technique.

Usage

```
sslGmmEM(xl, yl, xu, seed = 0, improvement = 1e-04, p = 0.3)
```

Arguments

`xl`	a n * p matrix or data.frame of labeled data
`yl`	a n * 1 integer vector of labels.
`xu`	a m * p matrix or data.frame of unlabeled data
`seed`	an integer specifying random number generation state for spliting labeled data into training set and cross-validation set.
`improvement`	numeric. Minimal allowed improvement of parameters.
`p`	percentage of labeled data are splitted into cross-validation set.

Details

sslGmmEM introduces unlabeled data into parameter estimation process. The weight lambda is chosen by cross-validation. The Gaussian Mixture Model is estimated based on maximum log likelihood function with an EM algorithm. The E-step computes the probabilities of each class for every observation. The M-step computes parameters based on probabilities obtained in the E-step.

Value

a list of values is returned:

Fields

para a numeric estimated parameter matrix in which the column represents variables and the row represents estimated means and standard deviation of each class. for example, the first and second row represents the mean and standard deviation of the first class, the third and fourth row represents the mean and standard deviation of the second class,etc.

classProb the estimated class probabilities

yu the predicted label of unlabeled data

optLambda the optimal lambda chosen by cross-validation

Author(s)

Junxiang Wang

References

Kamal Nigam, Andrew Mccallum, Sebastian Thrun, Tom Mitchell(1999) Text Classification from Labeled and Unlabeled Documents using EM

Examples

```
data(iris)
xl<-iris[,-5]
#Suppose we know the first twenty observations of each class
#and we want to predict the remaining with Gaussian Mixture Model
#1 setosa, 2 versicolor, 3 virginica
yl<-rep(1:3,each=20)
known.label <-c(1:20,51:70,101:120)
xu<-xl[-known.label,]
xl<-xl[known.label,]
l<-sslGmmEM(xl,yl,xu)
```

sslLabelProp *Label Propagation*

Description

sslLabelProp propagates a few known labels to a large number of unknown labels according to their proximities to neighboring nodes. It supports many kinds of distance measurements and graph representations.

Usage

```
sslLabelProp(x, y, known.label, graph.type = "exp", dist.type = "Euclidean",
  alpha, alpha1, alpha2, k, epsilon, iter = 1000)
```

Arguments

x	a n * p matrix or data.frame of n observations and p predictors
y	a vector of k known labels. The rows of y must be the same as the length of known.label.
known.label	a vector indicating the row index of known labels in matrix x.
graph.type	character string; which type of graph should be created? Options include knn,enn,tanh and exp. • knn :kNN graphs.Nodes i, j are connected by an edge if i is in j 's k-nearest-neighborhood. k is a hyperparameter that controls the density of the graph. • enn :epsilon-NN graphs. Nodes i, j are connected by an edge, if the distance d(i, j) < epsilon. The hyperparameter epsilon controls neighborhood radius. • tanh:tanh-weighted graphs. w(i,j) = (tanh(alpha1(d(i,j) - alpha2)) + 1)/2. where d(i,j) denotes the distance between point i and j. Hyperparameters alpha1 and alpha2 control the slope and cutoff value respectively. • exp :exp-weighted graphs.w(i,j) = exp(-d(i,j)^2/alpha^2),where d(i,j) denotes the distance between point i and j. Hyperparameter alpha controls the decay rate.
dist.type	character string; this parameter controls the type of distance measurement.(see dist or pr_DB).
alpha	numeric parameter needed when graph.type = exp
alpha1	numeric parameter needed when graph.type = tanh
alpha2	numeric parameter needed when graph.type = tanh
k	integer parameter needed when graph.type = knn
epsilon	numeric parameter needed when graph.type = enn
iter	iteration

Details

sslLabelProp implements label propagation algorithm in iter iterations.It supports many kinds of distance measurements and four types of graph creations.

Value

a n * 1 vector indicating the predictions of n observations in C class

Author(s)

Junxiang Wang

References

Xiaojin Zhu(2005),Semi-Supervised Learning with Graphs

See Also

dist, pr_DB

Examples

```
data(iris)
x<-iris[,1:4]
#Suppose we know the first twenty observations of each class and we want to propagate
#these labels to unlabeled data.
# 1 setosa, 2 versicolor, 3 virginica
y<-rep(1:3,each =20)
known.label <-c(1:20,51:70,101:120)
f1<-sslLabelProp(x,y,known.label,graph.type="enn",epsilon = 0.5)
f2<-sslLabelProp(x,y,known.label,graph.type="knn",k =10)
f3<-sslLabelProp(x,y,known.label,graph.type="tanh",alpha1=-2,alpha2=1)
f4<-sslLabelProp(x,y,known.label,graph.type="exp",alpha = 1)
```

---

sslLapRLS *Laplacian Regularized Least Squares*

Description

Laplacian Regularized Least Squares

Usage

```
sslLapRLS(xl, yl, xu, graph.type = "exp", dist.type = "Euclidean", alpha,
  alpha1, alpha2, k, epsilon, kernel = "gaussian", c1, c2, c3, deg, gamma,
  alpha3, alpha4, gammaA = 1, gammaI = 1)
```

Arguments

xl	a n * p matrix or data.frame of labeled data.
yl	a n * 1 binary labels(1 or -1).
xu	a m * p matrix or data.frame of unlabeled data.
graph.type	character string; which type of graph should be created? Options include knn,enn,tanh and exp. • knn :kNN graphs.Nodes i, j are connected by an edge if i is in j 's k-nearest-neighborhood. k is a hyperparameter that controls the density of the graph. • enn :epsilon-NN graphs. Nodes i, j are connected by an edge, if the distance d(i, j) < epsilon. The hyperparameter epsilon controls neighborhood radius. • tanh:tanh-weighted graphs. w(i,j) = (tanh(alpha1(d(i,j) - alpha2)) + 1)/2.where d(i,j) denotes the distance between point i and j. Hyperparameters alpha1 and alpha2 control the slope and cutoff value respectively. • exp :exp-weighted graphs.w(i,j) = exp(-d(i,j)^2/alpha^2),where d(i,j) denotes the distance between point i and j. Hyperparameter alpha controls the decay rate.
dist.type	character string; this parameter controls the type of distance measurement.(see dist or pr_DB).
alpha	numeric parameter needed when graph.type = exp
alpha1	numeric parameter needed when graph.type = tanh
alpha2	numeric parameter needed when graph.type = tanh
k	integer parameter needed when graph.type = knn
epsilon	numeric parameter needed when graph.type = enn
kernel	character string; it controls four types of common kernel functions:linear,polynomial,gaussian and sigmoid. • linear:Linear kernel;k(x,y)=dot(x,y)+c1,where dot(x,y) is the dot product of vector x and y,c1 is a constant term. • polynomial:Polynomial kernel;k(x,y)=(alpha3 *dot(x,y)+c2)^deg,where dot(x,y) is the dot product of vector x and y.Adjustable parameters are the slope alpha3, the constant term c2 and the polynomial degree deg. • gaussian:Gaussian kernel;k(x,y)=exp(-gamma*d(x,y)^2),where d(x,y) is Euclidean distace between vector x and y,gamma is a slope parameter. • sigmoid:Hyperbolic Tangent (Sigmoid) Kernel;k(x,y)=tanh(alpha4*dot(x,y)+c3),where d(x,y) is dot product of vector x and y.There are two adjustable parameters in the sigmoid kernel, the slope alpha4 and the intercept constant c3.
c1	numeric parameter needed when kernel = linear
c2	numeric parameter needed when kernel = polynomial
c3	numeric parameter needed when kernel = sigmoid
deg	integer parameter needed when kernel = polynomial
gamma	numeric parameter needed when kernel = gaussian
alpha3	numeric parameter needed when kernel = polynomial

alpha4	numeric parameter needed when kernel = sigmoid
gammaA	numeric; model parameter.
gammaI	numeric; model parameter.

Value

a m * 1 integer vector representing the predicted labels of unlabeled data(1 or -1).

Author(s)

Junxiang Wang

References

Olivier Chapelle, Bernhard Scholkopf and Alexander Zien (2006). Semi-Supervised Learning.The MIT Press.

See Also

pr_DB dist

Examples

```
data(iris)
xl<-iris[c(1:20,51:70),-5]
xu<-iris[c(21:50,71:100),-5]
yl<-rep(c(1,-1),each=20)
# combinations of different graph types and kernel types
# graph.type =knn, kernel =linear
yu1<-sslLapRLS(xl,yl,xu,graph.type="knn",k=10,kernel="linear",c1=1)
# graph.type =knn, kernel =polynomial
yu2<-sslLapRLS(xl,yl,xu,graph.type="knn",k=10,kernel="polynomial",c2=1,deg=2,alpha3=1)
# graph.type =knn, kernel =gaussian
yu3<-sslLapRLS(xl,yl,xu,graph.type="knn",k=10,kernel="gaussian",gamma=1)
# graph.type =knn, kernel =sigmoid
yu4<-sslLapRLS(xl,yl,xu,graph.type="knn",k=10,kernel="sigmoid",c3=-10,
alpha4=0.001,gammaI  = 0.05,gammaA = 0.05)
# graph.type =enn, kernel =linear
yu5<-sslLapRLS(xl,yl,xu,graph.type="enn",epsilon=1,kernel="linear",c1=1)
# graph.type =enn, kernel =polynomial
yu6<-sslLapRLS(xl,yl,xu,graph.type="enn",epsilon=1,kernel="polynomial",c2=1,deg=2,alpha3=1)
# graph.type =enn, kernel =gaussian
yu7<-sslLapRLS(xl,yl,xu,graph.type="enn",epsilon=1,kernel="gaussian",gamma=1)
# graph.type =enn, kernel =sigmoid
yu8<-sslLapRLS(xl,yl,xu,graph.type="enn",epsilon=1,kernel="sigmoid",c3=-10,
alpha4=0.001,gammaI  = 0.05,gammaA = 0.05)
# graph.type =tanh, kernel =linear
yu9<-sslLapRLS(xl,yl,xu,graph.type="tanh",alpha1=-2,alpha2=1,kernel="linear",c1=1)
# graph.type =tanh, kernel =polynomial
yu10<-sslLapRLS(xl,yl,xu,graph.type="tanh",alpha1=-2,alpha2=1,
kernel="polynomial",c2=1,deg=2,alpha3=1)
```

```
# graph.type =tanh, kernel =gaussian
yu11<-sslLapRLS(xl,yl,xu,graph.type="tanh",alpha1=-2,alpha2=1,kernel="gaussian",gamma=1)
# graph.type =tanh, kernel =sigmoid
yu12<-sslLapRLS(xl,yl,xu,graph.type="tanh",alpha1=-2,alpha2=1,
kernel="sigmoid",c3=-10,alpha4=0.001,gammaI  = 0.05,gammaA = 0.05)
# graph.type =exp, kernel =linear
yu13<-sslLapRLS(xl,yl,xu,graph.type="exp",alpha=1,kernel="linear",c1=1)
# graph.type =exp, kernel =polynomial
yu14<-sslLapRLS(xl,yl,xu,graph.type="exp",alpha=1,kernel="polynomial",c2=1,deg=2,alpha3=1)
# graph.type =exp, kernel =gaussian
yu15<-sslLapRLS(xl,yl,xu,graph.type="exp",alpha=1,kernel="gaussian",gamma=1)
# graph.type =exp, kernel =sigmoid
yu16<-sslLapRLS(xl,yl,xu,graph.type="exp",alpha=1,kernel="sigmoid",
c3=-10,alpha4=0.001,gammaI  = 0.05,gammaA = 0.05)
```

`sslLDS` *Low Density Separation*

Description

`sslLDS` implements low density separation with Transductive Support Vector Machines(TSVM) for semi-supervised binary classification

Usage

```
sslLDS(xl, yl, xu, rho = 1, C = 1, dist.type = "Euclidean", p = 0.3,
  improvement = 1e-04, seed = 0, delta = 0.01, alpha = 0.01)
```

Arguments

`xl`	a n * p matrix or data.frame of labeled data
`yl`	a n * 1 binary labels(1 or -1).
`xu`	a m * p matrix or data.frame of unlabeled data.
`rho`	numeric;a parameter for connectivity kernel.It defines minimal rho-path distances.
`C`	numeric; a parameter in the TSVM training model.
`dist.type`	character string; this parameter controls the type of distance measurement.(see `dist` or `pr_DB`).
`p`	the percentage of data used for cross-validation set.
`improvement`	numeric; minimal allowed improvement of parameters.
`seed`	an integer specifying random number generation state for spliting labeled data into training set and cross-validation set.
`delta`	numeric; a allowed cutoff for the cumulative percent of variance to lose by multidimensional scaling.
`alpha`	numeric; a learning rate in the gradient descent algorithm.

Details

sslLDS constructs a low density graph with connectivity kernel.It implements multidemensional scaling for demensionality reduction and chooses optimal C.star by cross-validation. Finally, it trains the TSVM model with gradient descent algorithm.

Value

a list of values is returned:

Fields

yu the predicted label of unlabeled data

optC.star the optimal C.star chosen by cross-validation. C.star weights the unlabeled data in the TSVM model.

para estimated parameters of TSVM, including w and b

Author(s)

Junxiang Wang

References

Chapelle, O., & Zien, A. (2005) Semi-supervised classification by low density separation.In Proceedings of the tenth international workshop on artificial intelligence and statistics.(pp. 57-64). Barbados.

Examples

```
data(iris)
xl<-iris[c(1:20,51:70),-5]
xu<-iris[c(21:50,71:100),-5]
yl<-rep(c(1,-1),each=20)
l<-sslLDS(xl,yl,xu,alpha=0.1)
```

---

sslLLGC *Local and Global Consistency*

Description

Local and Global Consistency

Usage

```
sslLLGC(xl, yl, xu, dist.type = "Euclidean", alpha = 0.01, gamma = 1,
  iter = 1000)
```

Arguments

xl	a n * p matrix or data.frame of labeled data
yl	a n X C matrix representing labels of n observations in C classes.If observation i belongs to class j, then yl(i,j)=1, and other elements in the same row equal 0.
xu	a m * p matrix or data.frame of unlabeled data.
dist.type	character string; this parameter controls the type of distance measurement.(see dist or pr_DB).
alpha	a numeric parameter controls convergence rate.
gamma	a numeric parameter in the affinity matrix
iter	the number of iteration.

Value

a m * 1 integer vector representing the predicted labels of unlabeled data.

Author(s)

Junxiang Wang

References

Zhou, D., Bousquet, O., Lal, T., Weston, J. and Scholkopf, B. (2004). Learning with local and global consistency.

See Also

pr_DB dist

Examples

```
data(iris)
xl<-iris[c(1:20,51:70,101:120),-5]
yl<-matrix(0,ncol=3,nrow=60)
yl[1:20,1]<-1
yl[21:40,2]<-1
yl[41:60,3]<-1
xu<-iris[-c(1:20,51:70,101:120),-5]
yu<-sslLLGC(xl,yl,xu)
```

`sslMarkovRandomWalks` *t-step Markov Random Walks*

Description

t-step Markov Random Walks

Usage

```
sslMarkovRandomWalks(xl, yl, xu, t = 10, dist.type = "Euclidean", k = 10,
  gamma = 1, improvement = 1e-04)
```

Arguments

`xl`	a n * p matrix or data.frame of labeled data.
`yl`	a n * 1 binary labels(1 or -1).
`xu`	a m * p matrix or data.frame of unlabeled data.
`t`	step size.
`dist.type`	character string; this parameter controls the type of distance measurement.(see `dist` or `pr_DB`).
`k`	an integer parameter controls a k-nearest neighbor graph.
`gamma`	a numeric parameter in the affinity matrix.
`improvement`	numeric. Maximum allowed distance between computed parameters in two successive iterations at the steady state.

Details

`sslMarkovRandomWalks` transmits known labels to unlabeled data by t-step Markov random walks.Parameters are estimated by an EM algorithm.

Value

a m * 1 integer vector representing the predicted labels of unlabeled data.

Author(s)

Junxiang Wang

References

Szummer, M., & Jaakkola, T. (2001). Partially labeled classification with M random walks. Advances in Neural Information Processing Systems, 14.

See Also

`pr_DB` `dist`

Examples

```
data(iris)
xl<-iris[c(1:20,51:70),-5]
xu<-iris[c(21:50,71:100),-5]
yl<-rep(c(1,-1),each=20)
yu<-sslMarkovRandomWalks(xl,yl,xu)
```

`sslMincut` *Mincut*

Description

`sslMincut` implements the Mincut algorithm for maxflow graph partition in the k-nearest neighbor graph.

Usage

```
sslMincut(xl, yl, xu, simil.type = "correlation", k = 10)
```

Arguments

`xl`	a n * p matrix or data.frame of labeled data
`yl`	a n * 1 binary labels(1 or -1).
`xu`	a m * p matrix or data.frame of unlabeled data.
`simil.type`	character string; this parameter controls the type of similarity measurement.(see `simil` or `pr_DB`).
`k`	an integer parameter controls a k-nearest neighbor graph.

Details

`sslMincut` creates a k-nearest neighbor graph and finds a maxflow from the first postive observation to the first negative one based on MPLA algorithm. This maxflow partitions the graph into postive labels and negative ones.

Value

a m * 1 integer vector representing the predicted labels of unlabeled data.

Author(s)

Junxiang Wang

References

Blum, A., & Chawla, S. (2001). Learning from labeled and unlabeled data using graph mincuts. Proc. 18th International Conf. on Machine Learning.

See Also

pr_DB simil

Examples

```
data(iris)
xl<-iris[c(1:20,51:70),-5]
xu<-iris[c(21:50,71:100),-5]
yl<-rep(c(1,-1),each=20)
yu<-sslMincut(xl,yl,xu)
```

---

sslRegress *Regression on graphs*

Description

sslRegress develops a regularization framework on graphs.It supports many kinds of distance measurements and graph representations. However, it only supports binary classifications.

Usage

```
sslRegress(xl, yl, xu, graph.type = "exp", dist.type = "Euclidean", alpha,
  alpha1, alpha2, p = 2, method = "Tikhonov", gamma = 1)
```

Arguments

xl	a n * p matrix or data.frame of labeled data.
yl	a n * 1 binary labels(1 or -1).
xu	a m * p matrix or data.frame of unlabeled data.
graph.type	character string; which type of graph should be created? Options includetanh and exp. • tanh:tanh-weighted graphs. w(i,j) = (tanh(alpha1(d(i,j) - alpha2)) + 1)/2.where d(i,j) denotes the distance between point i and j. Hyperparameters alpha1 and alpha2 control the slope and cutoff value respectively. • exp :exp-weighted graphs.w(i,j) = exp(-d(i,j)^2/alpha^2),where d(i,j) denotes the distance between point i and j. Hyperparameter alpha controls the decay rate.
dist.type	character string; this parameter controls the type of distance measurement.(see dist or pr_DB).
alpha	numeric parameter needed when graph.type = exp
alpha1	numeric parameter needed when graph.type = tanh
alpha2	numeric parameter needed when graph.type = tanh
p	an ineger parameter controls the power of Laplacian for regularization.
method	character string; this parameter choose two possible algorithms:"Tikhonov" means Tikhonov regularization;"Interpolated" means Interpolated regularization.
gamma	a parameter of Tikhonov regularization.

Value

a m * 1 integer vector representing the predicted labels of unlabeled data(1 or -1).

Author(s)

Junxiang Wang

References

Belkin, M., Matveeva, I., & Niyogi, P. (2004a). Regularization and semisupervised learning on large graphs. COLT

See Also

pr_DB dist

Examples

```
data(iris)
xl<-iris[c(1:20,51:70),-5]
xu<-iris[c(21:50,71:100),-5]
yl<-rep(c(1,-1),each=20)
# Tikhonov regularization
yu1<-sslRegress(xl,yl,xu,graph.type="tanh",alpha1=-2,alpha2=1)
yu2<-sslRegress(xl,yl,xu,graph.type="exp",alpha = 1)
# Interpolated regularization
yu3<-sslRegress(xl,yl,xu,graph.type="tanh",alpha1=-2,alpha2=1,method="Interpolated")
yu4<-sslRegress(xl,yl,xu,graph.type="exp",alpha = 1,method="Interpolated")
```

---

sslSelfTrain *Self-Training*

Description

Self-Training

Usage

```
sslSelfTrain(xl, yl, xu, n = 10, nrounds, ...)
```

Arguments

xl	a n * p matrix or data.frame of labeled data
yl	a n * 1 integer vector of labels(begin from 1).
xu	a m * p matrix or data.frame of unlabeled data
n	number of unlabeled examples to add into labeled data in each iteration
nrounds	the maximal number of iterations, see more in xgb.train
...	other parameters

Details

In self-training a classifier is first trained with the small amount of labeled data using extreme gradient boosting. The classifier is then used to classify the unlabeled data. The most confident unlabeled points, together with their predicted labels, are added to the training set. The classifier is re-trained and the procedure repeats.

Value

a m * 1 integer vector representing the predictions of unlabeled data.

Author(s)

Junxiang Wang

References

Rosenberg, C., Hebert, M., & Schneiderman, H. (2005). Semi-supervised selftraining of object detection models. Seventh IEEE Workshop on Applications of Computer Vision.

See Also

xgb.train

Examples

```
data(iris)
xl<-iris[,1:4]
#Suppose we know the first twenty observations of each class
#and we want to predict the remaining with self-training
# 1 setosa, 2 versicolor, 3 virginica
yl<-rep(1:3,each = 20)
known.label <-c(1:20,51:70,101:120)
xu<-xl[-known.label,]
xl<-xl[known.label,]
yu<-sslSelfTrain(xl,yl,xu,nrounds = 100,n=30)
```

Index