Package 'svmplus'

April 25, 2018

Type Package

Title Implementation of Support Vector Machines Plus (SVM+)

Date 2018-04-25

Version 1.0.1

Author Niharika Gauraha and Ola Spjuth

Maintainer Niharika Gauraha <niharika.gauraha@farmbio.uu.se>

Description Implementation of Support Vector Machines Plus (SVM+) for classification problems. See (Vladimir et. al, 2009, <doi:10.1016/j.neunet.2009.06.042>) for theoretical details and see (Li et. al, 2016, <https://github.com/okbalefthanded/svmplus_matlab>) for implementation details in 'MATLAB'.

Depends R (>= 2.15.0), quadprog, methods, Matrix, MASS

License GPL-3

Encoding UTF-8

NeedsCompilation no

LazyData true

RoxygenNote 6.0.1

Repository CRAN

Date/Publication 2018-04-25 14:21:48 UTC

R topics documented:

SVMP *Creates and returns an instance of the class specified in the svm_type.*

Description

Creates and returns an instance of the class specified in the svm_type. In future, the current solver used for quadratic programming (quadprog) will be replaced by the equivaent quadprog solver defined in CVXR package. Also, LIBSVM and LIBLINEAR based faster implementaions are planned to be supported.

Usage

```
SVMP(cost = 1, gamma = 1, kernel_x = "rbf", degree_x = 3,
gamma_x = 0.001, kernel_xstar = "rbf", degree_xstar = 3,
gamma_xstar = 0.001, tol = 1e-05, svm_type = "QP")
```

Arguments

`cost`	cost of constraints violation
`gamma`	parameter needed for priviledged information
`kernel_x`	the kernel used for standard training data
`degree_x`	parameter needed for polynomial kernel for training data
`gamma_x`	parameter needed for rbf kernel for training data
`kernel_xstar`	the kernel used for priviledged information (PI)
`degree_xstar`	parameter needed for polynomial kernel for PI
`gamma_xstar`	parameter needed for rbf kernel for PI
`tol`	tolerance of dual variables
`svm_type`	optimization techiniques used: QP, LibSVM, LibLinear etc. Currently it supports only QP.

Value

an instance of the class specified in the svm_type. Currently it suports only "QP", hence returns instance of the class QPSvmPlus. The return instance can be used to call fit, project and predict methods of the QPSvmPlus.

Author(s)

Niharika Gauraha and Ola Spjuth

Examples

```
# This example is similar to the example given in the section 3.3 of the article:
# https://doi.org/10.1007/s10472-017-9541-2

#Generate train data
  mean1 = rep(0, 2)
  mean2 = rep(1, 2)
  cov2  = cov1 = .5 * diag(2)
  n = 20
  X1 = mvrnorm(n, mean1, Sigma = cov1)
  X2 = mvrnorm(n, mean2, Sigma = cov2)
  X_train = rbind(X1, X2)
  y_train = matrix(c(rep(1, n), rep(-1, n)), 2*n, 1)
# geberate privileged information data
  X1Star = matrix(0, n, 2)
  X2Star = matrix(0, n, 2)
  for(i in 1:n)
  {
    X1Star[i, 1] = norm(X1[i,] - mean1, type = ”2”)
    X1Star[i, 2] = norm(X2[i,] - mean2, type = ”2”)
  }
  for(i in 1:n)
  {
    X2Star[i, 1] = norm(X1[i, ] - mean2, type = ”2”)
    X2Star[i, 2] = norm(X2[i, ] - mean1, type = ”2”)
  }
  XStar = rbind(X1Star, X2Star)
# generate test data
  n_test = 10
  X1 = mvrnorm(n_test, mean1, Sigma = cov1)
  X2 = mvrnorm(n_test, mean2, Sigma = cov2)
  X_test = rbind(X1, X2)
  y_test = matrix(c(rep(1, n_test), rep(-1, n_test)), 2*n_test, 1)
# create instance of the class type QP, using RBF kernel
  qp = SVMP(cost = 100, gamma = .01,
            kernel_x = ”rbf”, gamma_x = .001,
            kernel_xstar = ”rbf”, gamma_xstar = .001,
            tol = .00001, svm_type = ”QP”)
# call the fit function
  qp$fit(X_train, XStar, y_train)
# call the predict function
  y_predict = qp$predict(X_test)
  print(length(y_predict[y_predict == y_test]))
  print(”correct classification out of 20”)



  #using polynomial kernel
  qp = SVMP(cost = 100, gamma = .01,
            kernel_x = ”poly”, degree_x = 3,
            kernel_xstar = ”poly”, gamma_xstar = 3,
            tol = .00001)
```

```
qp$fit(X_train, XStar, y_train)

y_predict = qp$predict(X_test)

print(length(y_predict[y_predict == y_test]))
print("correct classification out of 20")

#using linear kernel
qp = SVMP(cost = 10, gamma = .1,
          kernel_x = "linear",
          kernel_xstar = "linear",
          tol = .00001)

qp$fit(X_train, XStar, y_train)

y_predict = qp$predict(X_test)

print(length(y_predict[y_predict == y_test]))
print("correct classification out of 20")
```

svmplus *Implementation of SVM Plus*

Description

Implementation of SVM plus for classification problems.

Details

The classical machine learning paradigm assumes, training examples in the form of iid pair:

$$(x_1, y_1), ..., (x_l, y_l), \quad x_i \in X, \quad y_i \in \{-1, +1\}.$$

Training examples are represented as features x_i and the same feature space is required for predicting future observations. However, this approach does not make use of other useful data that is only available at training time; such data is referred to as Privileged Information (PI).

Learning Under Privileged Information (LUPI) is a novel machine learning paradigm. It offers faster convergence of the learning process by exploiting the privileged information. In other words, “fewer training examples are needed to achieve similar predictive performance" or “the same number of examples can provide a better predictive performance". In LUPI paradigm, training examples come in the form of iid triplets

$$(x_1, x_1^*, y_1), ..., (x_l, x_l^*, y_l), \quad x_i \in X, \quad x_i^* \in X^*, \quad y_i \in \{-1, +1\}$$

where x^* denotes PI. SVM+ is one realization of LUPI paradigm. In SVM+, privileged information is used to estimate a linear model of the slack variables, namely

$$\xi_i = (w^*)^T z_i^* + b^*,$$

where $z_i = \phi(x_i)$ represents the kernel mapping.

The SVM+ objective function is defined as:

$$\min_{w,b} \left\{ \frac{1}{2} w^T w + \frac{\gamma}{2} (w^*)^T (w^*) + C \sum_{i=1}^{l} [(w^*)^T z_i^* + b^*] \right\}$$

$$s.t. \quad y_i(w^T z_i + b) \geq 1 - [(w^*)^T z_i^* + b^*],$$

$$(w^*)^T z_i^* + b^* \geq 0, \forall i$$

The dual SVM+ problem is defined as follow.

$$\max_{w,b} \left\{ \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j K(x_i, x_j) - \frac{1}{2\gamma} \sum_{i,j=1}^{l} (\alpha_i + \beta_i - C)(\alpha_j + \beta_j - C) K^*(x_i^*, x_j^*) \right\}$$

$$s.t. \quad \sum_{i=1}^{l} \alpha_i y_i = 0, \quad \sum_{i=1}^{l} (\alpha_i + \beta_i - C) = 0,$$

$$\alpha_i \geq 0, \quad \beta_i \geq 0$$

This package offeres a Quadratic Programming (QP) based convex optimization solution for the dual SVM+ problem. In future, LIBSVM and LibLinear based faster implementaions are planned to be supported. We refer to [1] for theoretical details of LUPI and SVM+, and we refer to [2] for implementation details of SVM+ in MATLAB.

References

[1] Vladimir et. al, Neural Networks, 2009, 22, pp 544–557. https://doi.org/10.1016/j.neunet.2009.06.042

[2] Li et. al, 2016. https://github.com/okbalefthanded/svmplus_matlab

[3] Bendtsen, C., et al., Ann Math Artif Intell, 2017, 81, pp 155–166. https://doi.org/10.1007/s10472-017-9541-2

Index