* Vapnik等人提出

Matlab函数：

fitcsvm

predict

fitcsvm Fit a classification Support Vector Machine (SVM)

MODEL=fitcsvm(TBL,Y) returns an SVM model MODEL for data in the table

TBL and response Y. TBL contains the predictor variables. Y can be any

of the following:

1. An array of class labels. Y can be a categorical array, logical

vector, numeric vector, or cell array of strings.

2. The name of a variable in TBL. This variable is used as the

response Y, and the remaining variables in TBL are used as

predictors.

3. A formula string such as 'y ~ x1 + x2 + x3' specifying that the

variable y is to be used as the response, and the other variables

in the formula are predictors. Any table variables not listed in

the formula are not used.

MODEL=fitcsvm(X,Y) is an alternative syntax that accepts X as an

N-by-P matrix of predictors with one row per observation and one column

per predictor. Y is the response and is an array of N class labels.

MODEL is a classification SVM model. If you use one of the following

five options and do not pass OptimizeHyperparameters, MODEL is of class

ClassificationPartitionedModel: 'CrossVal', 'KFold', 'Holdout',

'Leaveout' or 'CVPartition'. Otherwise, MODEL is of class

ClassificationSVM.

Use of a matrix X rather than a table TBL saves both memory and

execution time.

MODEL=fitcsvm(X,Y,'PARAM1',val1,'PARAM2',val2,...) specifies optional

parameter name/value pairs:

'Alpha' - Initial estimates of alpha coefficients, a vector

of non-negative elements, one per each row of X.

You cannot use this parameter for cross-

validation. Default: zeros(size(X,1),1) for

two-class learning and 0.5\*ones(size(X,1),1) for

one-class learning.

NOTE: The default setting can lead to long

training times for one-class learning.

To speed up training, set a large fraction of

the alpha coefficients to zero.

'BoxConstraint' - Positive scalar specifying the box constraint.

For one-class learning the box constraint is

always set to 1. Default: 1

'CacheSize' - Either positive scalar or string 'maximal'. If

numeric, this parameter specifies the cache size

in megabytes (MB). If set to 'maximal', fitcsvm

makes the cache large enough to hold the entire

Gram matrix of size N-by-N for N rows in X.

Optimizing the cache size can have a significant

impact on the training speed for data with many

observations. Default: 1000

'CategoricalPredictors' - List of categorical predictors. Pass

'CategoricalPredictors' as one of:

\* A numeric vector with indices between 1 and P,

where P is the number of columns of X or

variables in TBL.

\* A logical vector of length P, where a true

entry means that the corresponding column of X

or T is a categorical variable.

\* 'all', meaning all predictors are categorical.

\* A cell array of strings, where each element

in the array is the name of a predictor

variable. The names must match entries in

'PredictorNames' values.

Default: for a matrix input X, no categorical

predictors; for a table TBL, predictors are

treated as categorical if they are cell arrays of

strings, logical, or categorical.

'ClassNames' - Array of class names. Use the data type that

exists in Y. You can use this argument to order

the classes or select a subset of classes for

training. Default: All class names in Y.

'ClipAlphas' - Logical scalar. If true, fitcsvm sets the alpha

coefficients near zero to zero and sets the alpha

coefficients near the box constraint to the box

constraint at each iteration. This parameter can

affect SMO and ISDA convergence. Default: true

'Cost' - Square matrix, where COST(I,J) is the

cost of classifying a point into class J if its

true class is I. Alternatively, COST can be a

structure S with two fields: S.ClassificationCosts

containing the cost matrix C, and S.ClassNames

containing the class names and defining the

ordering of classes used for the rows and columns

of the cost matrix. For S.ClassNames use the data

type that exists in Y. Default: COST(I,J)=1 if

I~=J, and COST(I,J)=0 if I=J. fitcsvm uses the

input cost matrix to adjust the prior class

probabilities. fitcsvm then uses the adjusted

prior probabilities and the default cost matrix to

find the decision boundary.

'CrossVal' - If 'on', performs 10-fold cross-validation. You

use 'KFold', 'Holdout', 'Leaveout' and

'CVPartition' parameters to override this

cross-validation setting. You can only use one of

these four options ('KFold', 'Holdout', 'Leaveout'

and 'CVPartition') at a time. As an alternative,

you can cross-validate later using CROSSVAL

method. Default: 'off'

'CVPartition' - A partition created with CVPARTITION to use in

the cross-validated tree.

'Holdout' - Holdout validation uses the specified fraction

of the data for test, and uses the rest of the

data for training. Specify a numeric scalar

between 0 and 1.

'KFold' - Number of folds to use in cross-validated tree,

a positive integer. Default: 10

'Leaveout' - Use leave-one-out cross-validation by setting to

'on'.

'GapTolerance' - Non-negative scalar specifying tolerance for

feasibility gap obtained by SMO or ISDA. If zero,

fitcsvm does not use this parameter to check

convergence. Default: 0

'DeltaGradientTolerance'- Non-negative scalar specifying tolerance

for gradient difference between upper and lower

violators obtained by SMO or ISDA. If zero,

fitcsvm does not use this parameter to check

convergence. Default: 1e-3 if you set 'Solver' to

'SMO' and 0 if you set 'Solver' to 'ISDA'

'KKTTolerance' - Non-negative scalar specifying tolerance for

Karush-Kuhn-Tucker (KKT) violation obtained by SMO

or ISDA. If zero, fitcsvm does not use this

parameter to check convergence. Default: 0 if you

set 'Solver' to 'SMO' and 1e-3 if you set 'Solver'

to 'ISDA'

'IterationLimit' - Positive integer specifying the maximal number

of iterations for SMO and ISDA. fitcsvm returns

when this limit is reached, even if optimization

did not converge. Default: 1e6

'KernelFunction' - String specifying function for computing

elements of the Gram matrix. Pass as one of:

'linear', 'gaussian' (or 'rbf'), 'polynomial' or

name of a function on the MATLAB path. A kernel

function must be of the form

function G = KFUN(U, V)

The returned value, G, is a matrix of size M-by-N,

where M and N are the number of rows in U and V,

respectively. Default: 'linear' for two-class

learning and 'gaussian' (or 'rbf') for one-class

learning

'KernelScale' - Either string 'auto' or positive scalar specifying

the scale factor. If you pass 'auto', fitcsvm

selects an appropriate scale factor using a

heuristic procedure. To compute the Gram matrix,

fitcsvm divides elements in predictor matrix X by

this factor if the 'KernelFunction' value is one

of: 'linear', 'gaussian' (or 'rbf'), or

'polynomial'. If you pass your own kernel

function, you must apply scaling in that function.

Default: 1

NOTE: The heuristic procedure for estimation of

the scale factor uses subsampling. Estimates

obtained by this procedure can vary from one

application of fitcsvm to another. Set the

random number generator seed prior to calling

fitcsvm for reproducibility.

'KernelOffset' - Non-negative scalar. After fitcsvm computes an

element of the Gram matrix, fitcsvm adds this

value to the computed element. Default: 0 if you

set 'Solver' to 'SMO' and 0.1 if you set 'Solver'

to 'ISDA'

'OptimizeHyperparameters'

- Hyperparameters to optimize. Either 'none',

'auto', 'all', a cell array of eligible

hyperparameter names, or a vector of

optimizableVariable objects, such as that returned

by the 'hyperparameters' function. To control

other aspects of the optimization, use the

HyperparameterOptimizationOptions name-value pair.

'auto' is equivalent to {'BoxConstraint',

'KernelScale'}. 'all' is equivalent to

{'BoxConstraint', 'KernelFunction', 'KernelScale',

'PolynomialOrder', 'Standardize'}. Default:

'none'.

'HyperparameterOptimizationOptions'

- A struct specifying additional hyperparameter

optimization options. To enable hyperparameter

optimization, use the OptimizeHyperparameters

name-value pair. Recognized fields (all optional)

are:

Optimizer - One of {'bayesopt', 'gridsearch',

or 'randomsearch'} specifying the

optimization algorithm.

Default: 'bayesopt'

MaxObjectiveEvaluations

- Specifies the maximum number of

function evaluations to perform.

Default: 30 for Optimizer equal to

'bayesopt' or 'randomsearch'; The

full grid size for 'gridsearch'.

AcquisitionFunctionName

- When Optimizer is 'bayesopt',

specifies the Acquisition Function

to use in choosing the next point

to evaluate. See BAYESOPT for

accepted values. Default:

'expected-improvement-per-second-plus'

NumGridDivisions

- When Optimizer is 'gridsearch',

specifies the number of grid

divisions per dimension. Can be a

vector with the number of divisions

for each parameter, or a scalar

applied to all parameters. For

categorical parameters, the passed

value is ignored and all categories

are used.

Default: 10

ShowPlots - A logical scalar. If true,

a plot is displayed of the best

function value found as a function

of the number of function

evaluations. If there are 1 or 2

parameters to optimize, it will

also display a plot of a model of

the the Objective Function vs. the

parameters being optimized.

Default: true

SaveIntermediateResults

- A logical scalar. If true and

Optimizer is bayesopt, a variable

'BayesoptResults' will be

overwritten in the workspace after

each iteration.

Default: false

Verbose - 0, 1 or 2. Controls the level of

detail of command line display.

Default: 1

Repartition - A logical scalar. If true, a

new data partitioning is created

for each function evaluation. If

false, a single partitioning is

used for all evaluations.

Default: false

Use no more than one of the following three field

names, to define the objective function to be

optimized:

CVPartition - A cvpartition object.

Holdout - A scalar in the range (0,1).

KFold - An integer greater than 1.

Default: {KFold,5}

'PolynomialOrder' - Positive integer specifying the degree of

polynomial to be used for polynomial kernel.

fitcsvm uses this parameter only if you set

'KernelFunction' to 'polynomial'. Default: 3

'Nu' - Positive scalar specifying the Nu parameter for

one-class learning. fitcsvm fits coefficients

ALPHA such that sum(ALPHA)=Nu\*size(X,1).

Default: 0.5

'NumPrint' - Non-negative scalar. Diagnostic messages are

displayed during optimization by SMO or ISDA every

'NumPrint' iterations. fitcsvm uses this parameter

only if you set 'Verbose' to 1. Default: 1000

'OutlierFraction' - Scalar between 0 (inclusive) and 1 specifying

expected fraction of outlier observations in the

training set. For two-class learning, fitcsvm

removes observations with large gradients ensuring

that the specified fraction of observations will

be removed by the time convergence is reached. For

one-class learning, fitcsvm finds the bias term

such that the specified fraction of observations

in the training set has negative scores.

Default: 0

'PredictorNames' - A cell array of names for the predictor

variables, in the order in which they appear in X.

Default: {'x1','x2',...}. For a table TBL, these

names must be a subset of the variable names in

TBL, and only the selected variables are used. Not

allowed when Y is a formula. Default: all

variables other than Y.

'Prior' - Prior probabilities for each class. Specify as one

of:

\* A string:

- 'empirical' determines class probabilities

from class frequencies in Y

- 'uniform' sets all class probabilities equal

\* A vector (one scalar value for each class)

\* A structure S with two fields: S.ClassProbs

containing a vector of class probabilities, and

S.ClassNames containing the class names and

defining the ordering of classes used for the

elements of this vector.

If you pass numeric values, FITCTREE normalizes

them to add up to one. Default: 'empirical'

'RemoveDuplicates' - Logical scalar. If true, fitcsvm replaces

duplicate observations in the training data with a

single observation with weight equal to the

cumulative weight of these duplicates. Setting

this parameter to true can speed up training

considerably for data with many duplicate

observations. Default: false

'ResponseName' - Name of the response variable Y, a string. Not

allowed when Y is a name or formula. Default: 'Y'

'ScoreTransform' - Function handle for transforming scores, or

string representing a built-in transformation

function. Available functions: 'symmetric',

'invlogit', 'ismax', 'symmetricismax', 'none',

'logit', 'doublelogit', 'symmetriclogit', and

'sign'. Default: 'none'

'Solver' - String specifying the solver name. Specify as one

of:

'SMO' Sequential Minimal Optimization

'ISDA' Iterative Single Data Algorithm

'L1QP' L1 soft-margin minimization by quadratic

programming (requires an Optimization

Toolbox license)

All solvers implement L1 soft-margin minimization.

Default: 'ISDA' if you set 'OutlierFraction' to a

positive value for two-class learning and 'SMO'

otherwise

'ShrinkagePeriod' - Non-negative integer. fitcsvm moves

observations from active set to inactive set every

'ShrinkagePeriod' iterations. If you pass zero,

fitcsvm does not shrink the active set. Shrinking

can speed up convergence significantly when the

support vector set is much smaller than the

training data. If you want to apply shrinkage, set

'ShrinkagePeriod' to 1000 as a rule of thumb.

Default: 0

'Standardize' - Logical scalar. If true, standardize X by centering

and dividing columns by their standard deviations.

Default: false

'Verbose' - Verbosity level, one of:

0 (Default) fitcsvm does not display any

diagnostic messages and does not save values

of convergence criteria.

1 fitcsvm displays diagnostic messages and saves

values of convergence criteria every

'NumPrint' iterations.

2 fitcsvm displays a lot of diagnostic messages

and saves values of convergence criteria at

every iteration.

'Weights' - Vector of observation weights, one weight per

observation. fitcsvm normalizes the weights to

add up to the value of the prior probability in

the respective class. Default: ones(size(X,1),1).

For an input table TBL, the 'Weights' value can be

the name of a variable in TBL.

Example 1: Train an SVM model on data with two classes. Estimate its

error by cross-validation.

load ionosphere;

svm = fitcsvm(X,Y,'Standardize',true,'KernelFunction','rbf',...

'KernelScale','auto');

cv = crossval(svm);

kfoldLoss(cv)

Example 2: Train an SVM model by one-class learning. Assume that 5% of

observations are outliers. Verify that the fraction of

observations with negative scores in the cross-validated

data is close to 5%.

load fisheriris;

svm = fitcsvm(meas,ones(size(meas,1),1),'Standardize',true,...

'KernelScale','auto','OutlierFraction',0.05);

cv = crossval(svm);

[~,score] = kfoldPredict(cv);

mean(score<0)

See also ClassificationSVM,

classreg.learning.partition.ClassificationPartitionedModel.

fitcsvm 的参考页