JavaScript is disabled on your browser.

* [Overview](http://docs.google.com/overview-summary.html)
* [Package](http://docs.google.com/package-summary.html)
* Class
* [Tree](http://docs.google.com/package-tree.html)
* [Index](http://docs.google.com/index-all.html)
* [Help](http://docs.google.com/help-doc.html)
* [Prev Class](http://docs.google.com/org/opencv/ml/StatModel.html)
* [Next Class](http://docs.google.com/org/opencv/ml/SVMSGD.html)
* [Frames](http://docs.google.com/index.html?org/opencv/ml/SVM.html)
* [No Frames](http://docs.google.com/SVM.html)
* [All Classes](http://docs.google.com/allclasses-noframe.html)
* Summary:
* Nested |
* [Field](#3znysh7) |
* Constr |
* [Method](#tyjcwt)
* Detail:
* [Field](#2s8eyo1) |
* Constr |
* [Method](#1pxezwc)

org.opencv.ml

## Class SVM

* java.lang.Object
  + [org.opencv.core.Algorithm](http://docs.google.com/org/opencv/core/Algorithm.html)
    - [org.opencv.ml.StatModel](http://docs.google.com/org/opencv/ml/StatModel.html)
      * org.opencv.ml.SVM
* public class SVM  
  extends [StatModel](http://docs.google.com/org/opencv/ml/StatModel.html)  
  Support Vector Machines. SEE: REF: ml\_intro\_svm

### Field SummaryFields

| Modifier and Type | Field and Description |
| --- | --- |
| static int | [**C**](http://docs.google.com/org/opencv/ml/SVM.html#C) |
| static int | [**C\_SVC**](http://docs.google.com/org/opencv/ml/SVM.html#C_SVC) |
| static int | [**CHI2**](http://docs.google.com/org/opencv/ml/SVM.html#CHI2) |
| static int | [**COEF**](http://docs.google.com/org/opencv/ml/SVM.html#COEF) |
| static int | [**CUSTOM**](http://docs.google.com/org/opencv/ml/SVM.html#CUSTOM) |
| static int | [**DEGREE**](http://docs.google.com/org/opencv/ml/SVM.html#DEGREE) |
| static int | [**EPS\_SVR**](http://docs.google.com/org/opencv/ml/SVM.html#EPS_SVR) |
| static int | [**GAMMA**](http://docs.google.com/org/opencv/ml/SVM.html#GAMMA) |
| static int | [**INTER**](http://docs.google.com/org/opencv/ml/SVM.html#INTER) |
| static int | [**LINEAR**](http://docs.google.com/org/opencv/ml/SVM.html#LINEAR) |
| static int | [**NU**](http://docs.google.com/org/opencv/ml/SVM.html#NU) |
| static int | [**NU\_SVC**](http://docs.google.com/org/opencv/ml/SVM.html#NU_SVC) |
| static int | [**NU\_SVR**](http://docs.google.com/org/opencv/ml/SVM.html#NU_SVR) |
| static int | [**ONE\_CLASS**](http://docs.google.com/org/opencv/ml/SVM.html#ONE_CLASS) |
| static int | [**P**](http://docs.google.com/org/opencv/ml/SVM.html#P) |
| static int | [**POLY**](http://docs.google.com/org/opencv/ml/SVM.html#POLY) |
| static int | [**RBF**](http://docs.google.com/org/opencv/ml/SVM.html#RBF) |
| static int | [**SIGMOID**](http://docs.google.com/org/opencv/ml/SVM.html#SIGMOID) |

### Fields inherited from class org.opencv.ml.[**StatModel**](http://docs.google.com/org/opencv/ml/StatModel.html)[COMPRESSED\_INPUT](http://docs.google.com/org/opencv/ml/StatModel.html#COMPRESSED_INPUT), [PREPROCESSED\_INPUT](http://docs.google.com/org/opencv/ml/StatModel.html#PREPROCESSED_INPUT), [RAW\_OUTPUT](http://docs.google.com/org/opencv/ml/StatModel.html#RAW_OUTPUT), [UPDATE\_MODEL](http://docs.google.com/org/opencv/ml/StatModel.html#UPDATE_MODEL)

### Method SummaryMethods

| Modifier and Type | Method and Description |
| --- | --- |
| static [SVM](http://docs.google.com/org/opencv/ml/SVM.html) | [**\_\_fromPtr\_\_**](http://docs.google.com/org/opencv/ml/SVM.html#__fromPtr__(long))(long addr) |
| static [SVM](http://docs.google.com/org/opencv/ml/SVM.html) | [**create**](http://docs.google.com/org/opencv/ml/SVM.html#create())() Creates empty model. |
| double | [**getC**](http://docs.google.com/org/opencv/ml/SVM.html#getC())() SEE: setC |
| [Mat](http://docs.google.com/org/opencv/core/Mat.html) | [**getClassWeights**](http://docs.google.com/org/opencv/ml/SVM.html#getClassWeights())() SEE: setClassWeights |
| double | [**getCoef0**](http://docs.google.com/org/opencv/ml/SVM.html#getCoef0())() SEE: setCoef0 |
| double | [**getDecisionFunction**](http://docs.google.com/org/opencv/ml/SVM.html#getDecisionFunction(int,%20org.opencv.core.Mat,%20org.opencv.core.Mat))(int i, [Mat](http://docs.google.com/org/opencv/core/Mat.html) alpha, [Mat](http://docs.google.com/org/opencv/core/Mat.html) svidx) Retrieves the decision function |
| static [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) | [**getDefaultGridPtr**](http://docs.google.com/org/opencv/ml/SVM.html#getDefaultGridPtr(int))(int param\_id) Generates a grid for %SVM parameters. |
| double | [**getDegree**](http://docs.google.com/org/opencv/ml/SVM.html#getDegree())() SEE: setDegree |
| double | [**getGamma**](http://docs.google.com/org/opencv/ml/SVM.html#getGamma())() SEE: setGamma |
| int | [**getKernelType**](http://docs.google.com/org/opencv/ml/SVM.html#getKernelType())() Type of a %SVM kernel. |
| double | [**getNu**](http://docs.google.com/org/opencv/ml/SVM.html#getNu())() SEE: setNu |
| double | [**getP**](http://docs.google.com/org/opencv/ml/SVM.html#getP())() SEE: setP |
| [Mat](http://docs.google.com/org/opencv/core/Mat.html) | [**getSupportVectors**](http://docs.google.com/org/opencv/ml/SVM.html#getSupportVectors())() Retrieves all the support vectors The method returns all the support vectors as a floating-point matrix, where support vectors are stored as matrix rows. |
| [TermCriteria](http://docs.google.com/org/opencv/core/TermCriteria.html) | [**getTermCriteria**](http://docs.google.com/org/opencv/ml/SVM.html#getTermCriteria())() SEE: setTermCriteria |
| int | [**getType**](http://docs.google.com/org/opencv/ml/SVM.html#getType())() SEE: setType |
| [Mat](http://docs.google.com/org/opencv/core/Mat.html) | [**getUncompressedSupportVectors**](http://docs.google.com/org/opencv/ml/SVM.html#getUncompressedSupportVectors())() Retrieves all the uncompressed support vectors of a linear %SVM The method returns all the uncompressed support vectors of a linear %SVM that the compressed support vector, used for prediction, was derived from. |
| static [SVM](http://docs.google.com/org/opencv/ml/SVM.html) | [**load**](http://docs.google.com/org/opencv/ml/SVM.html#load(java.lang.String))(java.lang.String filepath) Loads and creates a serialized svm from a file Use SVM::save to serialize and store an SVM to disk. |
| void | [**setC**](http://docs.google.com/org/opencv/ml/SVM.html#setC(double))(double val) getC SEE: getC |
| void | [**setClassWeights**](http://docs.google.com/org/opencv/ml/SVM.html#setClassWeights(org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) val) getClassWeights SEE: getClassWeights |
| void | [**setCoef0**](http://docs.google.com/org/opencv/ml/SVM.html#setCoef0(double))(double val) getCoef0 SEE: getCoef0 |
| void | [**setDegree**](http://docs.google.com/org/opencv/ml/SVM.html#setDegree(double))(double val) getDegree SEE: getDegree |
| void | [**setGamma**](http://docs.google.com/org/opencv/ml/SVM.html#setGamma(double))(double val) getGamma SEE: getGamma |
| void | [**setKernel**](http://docs.google.com/org/opencv/ml/SVM.html#setKernel(int))(int kernelType) Initialize with one of predefined kernels. |
| void | [**setNu**](http://docs.google.com/org/opencv/ml/SVM.html#setNu(double))(double val) getNu SEE: getNu |
| void | [**setP**](http://docs.google.com/org/opencv/ml/SVM.html#setP(double))(double val) getP SEE: getP |
| void | [**setTermCriteria**](http://docs.google.com/org/opencv/ml/SVM.html#setTermCriteria(org.opencv.core.TermCriteria))([TermCriteria](http://docs.google.com/org/opencv/core/TermCriteria.html) val) getTermCriteria SEE: getTermCriteria |
| void | [**setType**](http://docs.google.com/org/opencv/ml/SVM.html#setType(int))(int val) getType SEE: getType |
| boolean | [**trainAuto**](http://docs.google.com/org/opencv/ml/SVM.html#trainAuto(org.opencv.core.Mat,%20int,%20org.opencv.core.Mat))([Mat](http://docs.google.com/org/opencv/core/Mat.html) samples, int layout, [Mat](http://docs.google.com/org/opencv/core/Mat.html) responses) Trains an %SVM with optimal parameters |
| boolean | [**trainAuto**](http://docs.google.com/org/opencv/ml/SVM.html#trainAuto(org.opencv.core.Mat,%20int,%20org.opencv.core.Mat,%20int))([Mat](http://docs.google.com/org/opencv/core/Mat.html) samples, int layout, [Mat](http://docs.google.com/org/opencv/core/Mat.html) responses, int kFold) Trains an %SVM with optimal parameters |
| boolean | [**trainAuto**](http://docs.google.com/org/opencv/ml/SVM.html#trainAuto(org.opencv.core.Mat,%20int,%20org.opencv.core.Mat,%20int,%20org.opencv.ml.ParamGrid))([Mat](http://docs.google.com/org/opencv/core/Mat.html) samples, int layout, [Mat](http://docs.google.com/org/opencv/core/Mat.html) responses, int kFold, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) Cgrid) Trains an %SVM with optimal parameters |
| boolean | [**trainAuto**](http://docs.google.com/org/opencv/ml/SVM.html#trainAuto(org.opencv.core.Mat,%20int,%20org.opencv.core.Mat,%20int,%20org.opencv.ml.ParamGrid,%20org.opencv.ml.ParamGrid))([Mat](http://docs.google.com/org/opencv/core/Mat.html) samples, int layout, [Mat](http://docs.google.com/org/opencv/core/Mat.html) responses, int kFold, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) Cgrid, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) gammaGrid) Trains an %SVM with optimal parameters |
| boolean | [**trainAuto**](http://docs.google.com/org/opencv/ml/SVM.html#trainAuto(org.opencv.core.Mat,%20int,%20org.opencv.core.Mat,%20int,%20org.opencv.ml.ParamGrid,%20org.opencv.ml.ParamGrid,%20org.opencv.ml.ParamGrid))([Mat](http://docs.google.com/org/opencv/core/Mat.html) samples, int layout, [Mat](http://docs.google.com/org/opencv/core/Mat.html) responses, int kFold, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) Cgrid, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) gammaGrid, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) pGrid) Trains an %SVM with optimal parameters |
| boolean | [**trainAuto**](http://docs.google.com/org/opencv/ml/SVM.html#trainAuto(org.opencv.core.Mat,%20int,%20org.opencv.core.Mat,%20int,%20org.opencv.ml.ParamGrid,%20org.opencv.ml.ParamGrid,%20org.opencv.ml.ParamGrid,%20org.opencv.ml.ParamGrid))([Mat](http://docs.google.com/org/opencv/core/Mat.html) samples, int layout, [Mat](http://docs.google.com/org/opencv/core/Mat.html) responses, int kFold, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) Cgrid, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) gammaGrid, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) pGrid, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) nuGrid) Trains an %SVM with optimal parameters |
| boolean | [**trainAuto**](http://docs.google.com/org/opencv/ml/SVM.html#trainAuto(org.opencv.core.Mat,%20int,%20org.opencv.core.Mat,%20int,%20org.opencv.ml.ParamGrid,%20org.opencv.ml.ParamGrid,%20org.opencv.ml.ParamGrid,%20org.opencv.ml.ParamGrid,%20org.opencv.ml.ParamGrid))([Mat](http://docs.google.com/org/opencv/core/Mat.html) samples, int layout, [Mat](http://docs.google.com/org/opencv/core/Mat.html) responses, int kFold, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) Cgrid, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) gammaGrid, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) pGrid, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) nuGrid, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) coeffGrid) Trains an %SVM with optimal parameters |
| boolean | [**trainAuto**](http://docs.google.com/org/opencv/ml/SVM.html#trainAuto(org.opencv.core.Mat,%20int,%20org.opencv.core.Mat,%20int,%20org.opencv.ml.ParamGrid,%20org.opencv.ml.ParamGrid,%20org.opencv.ml.ParamGrid,%20org.opencv.ml.ParamGrid,%20org.opencv.ml.ParamGrid,%20org.opencv.ml.ParamGrid))([Mat](http://docs.google.com/org/opencv/core/Mat.html) samples, int layout, [Mat](http://docs.google.com/org/opencv/core/Mat.html) responses, int kFold, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) Cgrid, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) gammaGrid, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) pGrid, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) nuGrid, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) coeffGrid, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) degreeGrid) Trains an %SVM with optimal parameters |
| boolean | [**trainAuto**](http://docs.google.com/org/opencv/ml/SVM.html#trainAuto(org.opencv.core.Mat,%20int,%20org.opencv.core.Mat,%20int,%20org.opencv.ml.ParamGrid,%20org.opencv.ml.ParamGrid,%20org.opencv.ml.ParamGrid,%20org.opencv.ml.ParamGrid,%20org.opencv.ml.ParamGrid,%20org.opencv.ml.ParamGrid,%20boolean))([Mat](http://docs.google.com/org/opencv/core/Mat.html) samples, int layout, [Mat](http://docs.google.com/org/opencv/core/Mat.html) responses, int kFold, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) Cgrid, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) gammaGrid, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) pGrid, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) nuGrid, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) coeffGrid, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) degreeGrid, boolean balanced) Trains an %SVM with optimal parameters |

### Methods inherited from class org.opencv.ml.[**StatModel**](http://docs.google.com/org/opencv/ml/StatModel.html)[calcError](http://docs.google.com/org/opencv/ml/StatModel.html#calcError(org.opencv.ml.TrainData,%20boolean,%20org.opencv.core.Mat)), [empty](http://docs.google.com/org/opencv/ml/StatModel.html#empty()), [getVarCount](http://docs.google.com/org/opencv/ml/StatModel.html#getVarCount()), [isClassifier](http://docs.google.com/org/opencv/ml/StatModel.html#isClassifier()), [isTrained](http://docs.google.com/org/opencv/ml/StatModel.html#isTrained()), [predict](http://docs.google.com/org/opencv/ml/StatModel.html#predict(org.opencv.core.Mat)), [predict](http://docs.google.com/org/opencv/ml/StatModel.html#predict(org.opencv.core.Mat,%20org.opencv.core.Mat)), [predict](http://docs.google.com/org/opencv/ml/StatModel.html#predict(org.opencv.core.Mat,%20org.opencv.core.Mat,%20int)), [train](http://docs.google.com/org/opencv/ml/StatModel.html#train(org.opencv.core.Mat,%20int,%20org.opencv.core.Mat)), [train](http://docs.google.com/org/opencv/ml/StatModel.html#train(org.opencv.ml.TrainData)), [train](http://docs.google.com/org/opencv/ml/StatModel.html#train(org.opencv.ml.TrainData,%20int))

### Methods inherited from class org.opencv.core.[**Algorithm**](http://docs.google.com/org/opencv/core/Algorithm.html)[clear](http://docs.google.com/org/opencv/core/Algorithm.html#clear()), [getDefaultName](http://docs.google.com/org/opencv/core/Algorithm.html#getDefaultName()), [getNativeObjAddr](http://docs.google.com/org/opencv/core/Algorithm.html#getNativeObjAddr()), [save](http://docs.google.com/org/opencv/core/Algorithm.html#save(java.lang.String))

### Methods inherited from class java.lang.Objectequals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### Field Detail

#### C public static final int CSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.ml.SVM.C)

#### C\_SVC public static final int C\_SVCSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.ml.SVM.C_SVC)

#### CHI2 public static final int CHI2See Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.ml.SVM.CHI2)

#### COEF public static final int COEFSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.ml.SVM.COEF)

#### CUSTOM public static final int CUSTOMSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.ml.SVM.CUSTOM)

#### DEGREE public static final int DEGREESee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.ml.SVM.DEGREE)

#### EPS\_SVR public static final int EPS\_SVRSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.ml.SVM.EPS_SVR)

#### GAMMA public static final int GAMMASee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.ml.SVM.GAMMA)

#### INTER public static final int INTERSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.ml.SVM.INTER)

#### LINEAR public static final int LINEARSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.ml.SVM.LINEAR)

#### NU public static final int NUSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.ml.SVM.NU)

#### NU\_SVC public static final int NU\_SVCSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.ml.SVM.NU_SVC)

#### NU\_SVR public static final int NU\_SVRSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.ml.SVM.NU_SVR)

#### ONE\_CLASS public static final int ONE\_CLASSSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.ml.SVM.ONE_CLASS)

#### P public static final int PSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.ml.SVM.P)

#### POLY public static final int POLYSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.ml.SVM.POLY)

#### RBF public static final int RBFSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.ml.SVM.RBF)

#### SIGMOID public static final int SIGMOIDSee Also:[Constant Field Values](http://docs.google.com/constant-values.html#org.opencv.ml.SVM.SIGMOID)

### Method Detail

#### \_\_fromPtr\_\_ public static [SVM](http://docs.google.com/org/opencv/ml/SVM.html) \_\_fromPtr\_\_(long addr)

#### create public static [SVM](http://docs.google.com/org/opencv/ml/SVM.html) create() Creates empty model. Use StatModel::train to train the model. Since %SVM has several parameters, you may want to find the best parameters for your problem, it can be done with SVM::trainAuto.Returns:automatically generated

#### getC public double getC() SEE: setCReturns:automatically generated

#### getClassWeights public [Mat](http://docs.google.com/org/opencv/core/Mat.html) getClassWeights() SEE: setClassWeightsReturns:automatically generated

#### getCoef0 public double getCoef0() SEE: setCoef0Returns:automatically generated

#### getDecisionFunction public double getDecisionFunction(int i, [Mat](http://docs.google.com/org/opencv/core/Mat.html) alpha, [Mat](http://docs.google.com/org/opencv/core/Mat.html) svidx) Retrieves the decision functionParameters:i - the index of the decision function. If the problem solved is regression, 1-class or 2-class classification, then there will be just one decision function and the index should always be 0. Otherwise, in the case of N-class classification, there will be \(N(N-1)/2\) decision functions.alpha - the optional output vector for weights, corresponding to different support vectors. In the case of linear %SVM all the alpha's will be 1's.svidx - the optional output vector of indices of support vectors within the matrix of support vectors (which can be retrieved by SVM::getSupportVectors). In the case of linear %SVM each decision function consists of a single "compressed" support vector. The method returns rho parameter of the decision function, a scalar subtracted from the weighted sum of kernel responses. Returns:automatically generated

#### getDefaultGridPtr public static [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) getDefaultGridPtr(int param\_id) Generates a grid for %SVM parameters.Parameters:param\_id - %SVM parameters IDs that must be one of the SVM::ParamTypes. The grid is generated for the parameter with this ID. The function generates a grid pointer for the specified parameter of the %SVM algorithm. The grid may be passed to the function SVM::trainAuto. Returns:automatically generated

#### getDegree public double getDegree() SEE: setDegreeReturns:automatically generated

#### getGamma public double getGamma() SEE: setGammaReturns:automatically generated

#### getKernelType public int getKernelType() Type of a %SVM kernel. See SVM::KernelTypes. Default value is SVM::RBF.Returns:automatically generated

#### getNu public double getNu() SEE: setNuReturns:automatically generated

#### getP public double getP() SEE: setPReturns:automatically generated

#### getSupportVectors public [Mat](http://docs.google.com/org/opencv/core/Mat.html) getSupportVectors() Retrieves all the support vectors The method returns all the support vectors as a floating-point matrix, where support vectors are stored as matrix rows.Returns:automatically generated

#### getTermCriteria public [TermCriteria](http://docs.google.com/org/opencv/core/TermCriteria.html) getTermCriteria() SEE: setTermCriteriaReturns:automatically generated

#### getType public int getType() SEE: setTypeReturns:automatically generated

#### getUncompressedSupportVectors public [Mat](http://docs.google.com/org/opencv/core/Mat.html) getUncompressedSupportVectors() Retrieves all the uncompressed support vectors of a linear %SVM The method returns all the uncompressed support vectors of a linear %SVM that the compressed support vector, used for prediction, was derived from. They are returned in a floating-point matrix, where the support vectors are stored as matrix rows.Returns:automatically generated

#### load public static [SVM](http://docs.google.com/org/opencv/ml/SVM.html) load(java.lang.String filepath) Loads and creates a serialized svm from a file Use SVM::save to serialize and store an SVM to disk. Load the SVM from this file again, by calling this function with the path to the file.Parameters:filepath - path to serialized svm Returns:automatically generated

#### setC public void setC(double val) getC SEE: getCParameters:val - automatically generated

#### setClassWeights public void setClassWeights([Mat](http://docs.google.com/org/opencv/core/Mat.html) val) getClassWeights SEE: getClassWeightsParameters:val - automatically generated

#### setCoef0 public void setCoef0(double val) getCoef0 SEE: getCoef0Parameters:val - automatically generated

#### setDegree public void setDegree(double val) getDegree SEE: getDegreeParameters:val - automatically generated

#### setGamma public void setGamma(double val) getGamma SEE: getGammaParameters:val - automatically generated

#### setKernel public void setKernel(int kernelType) Initialize with one of predefined kernels. See SVM::KernelTypes.Parameters:kernelType - automatically generated

#### setNu public void setNu(double val) getNu SEE: getNuParameters:val - automatically generated

#### setP public void setP(double val) getP SEE: getPParameters:val - automatically generated

#### setTermCriteria public void setTermCriteria([TermCriteria](http://docs.google.com/org/opencv/core/TermCriteria.html) val) getTermCriteria SEE: getTermCriteriaParameters:val - automatically generated

#### setType public void setType(int val) getType SEE: getTypeParameters:val - automatically generated

#### trainAuto public boolean trainAuto([Mat](http://docs.google.com/org/opencv/core/Mat.html) samples, int layout, [Mat](http://docs.google.com/org/opencv/core/Mat.html) responses) Trains an %SVM with optimal parametersParameters:samples - training sampleslayout - See ml::SampleTypes.responses - vector of responses associated with the training samples. subset is used to test the model, the others form the train set. So, the %SVM algorithm is balanced cross-validation subsets that is proportions between classes in subsets are close to such proportion in the whole train dataset. The method trains the %SVM model automatically by choosing the optimal parameters C, gamma, p, nu, coef0, degree. Parameters are considered optimal when the cross-validation estimate of the test set error is minimal. This function only makes use of SVM::getDefaultGrid for parameter optimization and thus only offers rudimentary parameter options. This function works for the classification (SVM::C\_SVC or SVM::NU\_SVC) as well as for the regression (SVM::EPS\_SVR or SVM::NU\_SVR). If it is SVM::ONE\_CLASS, no optimization is made and the usual %SVM with parameters specified in params is executed. Returns:automatically generated

#### trainAuto public boolean trainAuto([Mat](http://docs.google.com/org/opencv/core/Mat.html) samples, int layout, [Mat](http://docs.google.com/org/opencv/core/Mat.html) responses, int kFold) Trains an %SVM with optimal parametersParameters:samples - training sampleslayout - See ml::SampleTypes.responses - vector of responses associated with the training samples.kFold - Cross-validation parameter. The training set is divided into kFold subsets. One subset is used to test the model, the others form the train set. So, the %SVM algorithm is balanced cross-validation subsets that is proportions between classes in subsets are close to such proportion in the whole train dataset. The method trains the %SVM model automatically by choosing the optimal parameters C, gamma, p, nu, coef0, degree. Parameters are considered optimal when the cross-validation estimate of the test set error is minimal. This function only makes use of SVM::getDefaultGrid for parameter optimization and thus only offers rudimentary parameter options. This function works for the classification (SVM::C\_SVC or SVM::NU\_SVC) as well as for the regression (SVM::EPS\_SVR or SVM::NU\_SVR). If it is SVM::ONE\_CLASS, no optimization is made and the usual %SVM with parameters specified in params is executed. Returns:automatically generated

#### trainAuto public boolean trainAuto([Mat](http://docs.google.com/org/opencv/core/Mat.html) samples, int layout, [Mat](http://docs.google.com/org/opencv/core/Mat.html) responses, int kFold, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) Cgrid) Trains an %SVM with optimal parametersParameters:samples - training sampleslayout - See ml::SampleTypes.responses - vector of responses associated with the training samples.kFold - Cross-validation parameter. The training set is divided into kFold subsets. One subset is used to test the model, the others form the train set. So, the %SVM algorithm isCgrid - grid for C balanced cross-validation subsets that is proportions between classes in subsets are close to such proportion in the whole train dataset. The method trains the %SVM model automatically by choosing the optimal parameters C, gamma, p, nu, coef0, degree. Parameters are considered optimal when the cross-validation estimate of the test set error is minimal. This function only makes use of SVM::getDefaultGrid for parameter optimization and thus only offers rudimentary parameter options. This function works for the classification (SVM::C\_SVC or SVM::NU\_SVC) as well as for the regression (SVM::EPS\_SVR or SVM::NU\_SVR). If it is SVM::ONE\_CLASS, no optimization is made and the usual %SVM with parameters specified in params is executed. Returns:automatically generated

#### trainAuto public boolean trainAuto([Mat](http://docs.google.com/org/opencv/core/Mat.html) samples, int layout, [Mat](http://docs.google.com/org/opencv/core/Mat.html) responses, int kFold, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) Cgrid, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) gammaGrid) Trains an %SVM with optimal parametersParameters:samples - training sampleslayout - See ml::SampleTypes.responses - vector of responses associated with the training samples.kFold - Cross-validation parameter. The training set is divided into kFold subsets. One subset is used to test the model, the others form the train set. So, the %SVM algorithm isCgrid - grid for CgammaGrid - grid for gamma balanced cross-validation subsets that is proportions between classes in subsets are close to such proportion in the whole train dataset. The method trains the %SVM model automatically by choosing the optimal parameters C, gamma, p, nu, coef0, degree. Parameters are considered optimal when the cross-validation estimate of the test set error is minimal. This function only makes use of SVM::getDefaultGrid for parameter optimization and thus only offers rudimentary parameter options. This function works for the classification (SVM::C\_SVC or SVM::NU\_SVC) as well as for the regression (SVM::EPS\_SVR or SVM::NU\_SVR). If it is SVM::ONE\_CLASS, no optimization is made and the usual %SVM with parameters specified in params is executed. Returns:automatically generated

#### trainAuto public boolean trainAuto([Mat](http://docs.google.com/org/opencv/core/Mat.html) samples, int layout, [Mat](http://docs.google.com/org/opencv/core/Mat.html) responses, int kFold, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) Cgrid, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) gammaGrid, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) pGrid) Trains an %SVM with optimal parametersParameters:samples - training sampleslayout - See ml::SampleTypes.responses - vector of responses associated with the training samples.kFold - Cross-validation parameter. The training set is divided into kFold subsets. One subset is used to test the model, the others form the train set. So, the %SVM algorithm isCgrid - grid for CgammaGrid - grid for gammapGrid - grid for p balanced cross-validation subsets that is proportions between classes in subsets are close to such proportion in the whole train dataset. The method trains the %SVM model automatically by choosing the optimal parameters C, gamma, p, nu, coef0, degree. Parameters are considered optimal when the cross-validation estimate of the test set error is minimal. This function only makes use of SVM::getDefaultGrid for parameter optimization and thus only offers rudimentary parameter options. This function works for the classification (SVM::C\_SVC or SVM::NU\_SVC) as well as for the regression (SVM::EPS\_SVR or SVM::NU\_SVR). If it is SVM::ONE\_CLASS, no optimization is made and the usual %SVM with parameters specified in params is executed. Returns:automatically generated

#### trainAuto public boolean trainAuto([Mat](http://docs.google.com/org/opencv/core/Mat.html) samples, int layout, [Mat](http://docs.google.com/org/opencv/core/Mat.html) responses, int kFold, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) Cgrid, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) gammaGrid, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) pGrid, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) nuGrid) Trains an %SVM with optimal parametersParameters:samples - training sampleslayout - See ml::SampleTypes.responses - vector of responses associated with the training samples.kFold - Cross-validation parameter. The training set is divided into kFold subsets. One subset is used to test the model, the others form the train set. So, the %SVM algorithm isCgrid - grid for CgammaGrid - grid for gammapGrid - grid for pnuGrid - grid for nu balanced cross-validation subsets that is proportions between classes in subsets are close to such proportion in the whole train dataset. The method trains the %SVM model automatically by choosing the optimal parameters C, gamma, p, nu, coef0, degree. Parameters are considered optimal when the cross-validation estimate of the test set error is minimal. This function only makes use of SVM::getDefaultGrid for parameter optimization and thus only offers rudimentary parameter options. This function works for the classification (SVM::C\_SVC or SVM::NU\_SVC) as well as for the regression (SVM::EPS\_SVR or SVM::NU\_SVR). If it is SVM::ONE\_CLASS, no optimization is made and the usual %SVM with parameters specified in params is executed. Returns:automatically generated

#### trainAuto public boolean trainAuto([Mat](http://docs.google.com/org/opencv/core/Mat.html) samples, int layout, [Mat](http://docs.google.com/org/opencv/core/Mat.html) responses, int kFold, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) Cgrid, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) gammaGrid, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) pGrid, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) nuGrid, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) coeffGrid) Trains an %SVM with optimal parametersParameters:samples - training sampleslayout - See ml::SampleTypes.responses - vector of responses associated with the training samples.kFold - Cross-validation parameter. The training set is divided into kFold subsets. One subset is used to test the model, the others form the train set. So, the %SVM algorithm isCgrid - grid for CgammaGrid - grid for gammapGrid - grid for pnuGrid - grid for nucoeffGrid - grid for coeff balanced cross-validation subsets that is proportions between classes in subsets are close to such proportion in the whole train dataset. The method trains the %SVM model automatically by choosing the optimal parameters C, gamma, p, nu, coef0, degree. Parameters are considered optimal when the cross-validation estimate of the test set error is minimal. This function only makes use of SVM::getDefaultGrid for parameter optimization and thus only offers rudimentary parameter options. This function works for the classification (SVM::C\_SVC or SVM::NU\_SVC) as well as for the regression (SVM::EPS\_SVR or SVM::NU\_SVR). If it is SVM::ONE\_CLASS, no optimization is made and the usual %SVM with parameters specified in params is executed. Returns:automatically generated

#### trainAuto public boolean trainAuto([Mat](http://docs.google.com/org/opencv/core/Mat.html) samples, int layout, [Mat](http://docs.google.com/org/opencv/core/Mat.html) responses, int kFold, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) Cgrid, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) gammaGrid, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) pGrid, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) nuGrid, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) coeffGrid, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) degreeGrid) Trains an %SVM with optimal parametersParameters:samples - training sampleslayout - See ml::SampleTypes.responses - vector of responses associated with the training samples.kFold - Cross-validation parameter. The training set is divided into kFold subsets. One subset is used to test the model, the others form the train set. So, the %SVM algorithm isCgrid - grid for CgammaGrid - grid for gammapGrid - grid for pnuGrid - grid for nucoeffGrid - grid for coeffdegreeGrid - grid for degree balanced cross-validation subsets that is proportions between classes in subsets are close to such proportion in the whole train dataset. The method trains the %SVM model automatically by choosing the optimal parameters C, gamma, p, nu, coef0, degree. Parameters are considered optimal when the cross-validation estimate of the test set error is minimal. This function only makes use of SVM::getDefaultGrid for parameter optimization and thus only offers rudimentary parameter options. This function works for the classification (SVM::C\_SVC or SVM::NU\_SVC) as well as for the regression (SVM::EPS\_SVR or SVM::NU\_SVR). If it is SVM::ONE\_CLASS, no optimization is made and the usual %SVM with parameters specified in params is executed. Returns:automatically generated

#### trainAuto public boolean trainAuto([Mat](http://docs.google.com/org/opencv/core/Mat.html) samples, int layout, [Mat](http://docs.google.com/org/opencv/core/Mat.html) responses, int kFold, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) Cgrid, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) gammaGrid, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) pGrid, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) nuGrid, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) coeffGrid, [ParamGrid](http://docs.google.com/org/opencv/ml/ParamGrid.html) degreeGrid, boolean balanced) Trains an %SVM with optimal parametersParameters:samples - training sampleslayout - See ml::SampleTypes.responses - vector of responses associated with the training samples.kFold - Cross-validation parameter. The training set is divided into kFold subsets. One subset is used to test the model, the others form the train set. So, the %SVM algorithm isCgrid - grid for CgammaGrid - grid for gammapGrid - grid for pnuGrid - grid for nucoeffGrid - grid for coeffdegreeGrid - grid for degreebalanced - If true and the problem is 2-class classification then the method creates more balanced cross-validation subsets that is proportions between classes in subsets are close to such proportion in the whole train dataset. The method trains the %SVM model automatically by choosing the optimal parameters C, gamma, p, nu, coef0, degree. Parameters are considered optimal when the cross-validation estimate of the test set error is minimal. This function only makes use of SVM::getDefaultGrid for parameter optimization and thus only offers rudimentary parameter options. This function works for the classification (SVM::C\_SVC or SVM::NU\_SVC) as well as for the regression (SVM::EPS\_SVR or SVM::NU\_SVR). If it is SVM::ONE\_CLASS, no optimization is made and the usual %SVM with parameters specified in params is executed. Returns:automatically generated

* [Overview](http://docs.google.com/overview-summary.html)
* [Package](http://docs.google.com/package-summary.html)
* Class
* [Tree](http://docs.google.com/package-tree.html)
* [Index](http://docs.google.com/index-all.html)
* [Help](http://docs.google.com/help-doc.html)
* [Prev Class](http://docs.google.com/org/opencv/ml/StatModel.html)
* [Next Class](http://docs.google.com/org/opencv/ml/SVMSGD.html)
* [Frames](http://docs.google.com/index.html?org/opencv/ml/SVM.html)
* [No Frames](http://docs.google.com/SVM.html)
* [All Classes](http://docs.google.com/allclasses-noframe.html)
* Summary:
* Nested |
* [Field](#3znysh7) |
* Constr |
* [Method](#tyjcwt)
* Detail:
* [Field](#2s8eyo1) |
* Constr |
* [Method](#1pxezwc)

Generated on 2021-04-02 03:15:03 / OpenCV 3.4.14