Category

Random Forest HOG

Random Forest

Author: Michał Pomarański,

Po-Tsung Chiu,

Chun-Jen Peng

Version : 2.0
Publication : 22/01/2014
Copyright:

Modeliosoft

MINES Paristech

Table of Contents

1 Introduction

Les ***forêts décisionnelles aléatoires*** (de l'anglais « *Random decision forest* ») ont été formellement proposées en 2001 par Leo Breiman et Adèle Cutler. Elles font partie des techniques d'apprentissage automatique. Cet algorithme combine les concepts de sous-espaces aléatoires et de « *bagging* ». L'algorithme des forêts d'arbres décisionnels effectuer un apprentissage sur de multiples arbres de décision entraînés sur des sous-ensembles de données légèrement différents et des sous-ensembles totalement différents de variables (composantes des vecteurs de "features ").

Un ***histogramme de gradient orienté*** (HOG) est une caractéristique utilisée en vision par ordinateur pour la classification d'images et la détection visuelle d'objets d'une catégorie donnée. La technique calcule des histogrammes locaux de l'orientation du gradient sur une grille dense, c'est-à-dire sur des zones régulièrement réparties sur l'image. Elle possède des points communs avec les "Scale-invariant feature transform" (SIFT), les Shape contexts et les histogrammes d'orientation de contours, mais s'en diffère notamment par l'utilisation d'une grille dense. La méthode s'est montrée particulièrement efficace pour la détection de personnes.

Nous allons utiliser les forêts décisionnelles aléatoires pour classifier des images qui ont été transformées en histogramme de gradient orienté.

2 Dictionnaire

2.1 Dictionnaire de forêts decisionelles Aléatoires

Nom	Définition	Nom UML

arbre de décision	flow-chart like structure
	node => some attribute
	branch => outcome of test
	path from root to leaf => classification rule
		Class DecisionTree

forêt d'arbres décisionnels	a classifier algorithm which contains multiple decision trees and each tree vote for classification	Class RandomForest
L'apprentissage des arbres décisionnels	The training algorithm for random forests applies the general technique of bootstrap aggregating, or bagging, to tree learners. Given a training set X = x1, …, xn with responses Y = y1 through yn, bagging repeatedly selects a bootstrap sample of the training set and fits trees to these samples	Method bootStrapSample De la class DecisionTree
L'histogramme de gradient orienté (HOG)	a descriptor represented by histograms of the "Orientation" of gradients in the images	Histogram of Oriented Gradients
classification d'image	Classify an image into one of several categories	Method classify(Sample) in any subclass of Classifier
Classifier (6.4 p.19)	a class that implements a prototype of classifiers such as AdaBoost or MultilayerPerception	classifier.Classifier in jLevis
Learner (7.3 p.27)	an abstract class that reflects a learning system	learner.Learner in jLevis
Sample (9.9 p.50)	a class that reflects an image	sample.Sample in jLevis
jLevis	existing framework containing classification algorithms	jLevis
Vote majoritaire	la classification fournie par la forêt	Méthode voteMajor

		de la classe RandomForest
Vote d'un arbre	chaque arbre vote pour une des classes	Méthode evaluate de la classe DecisionTree
bagging (bootstrap aggregating)	also known as "bootstrap aggregating". A machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used instatistical classification and regression. It also reduces variance and helps to avoid overfitting	Probablement un attribut de la classe DecisionTree, qui sera fixé lors de création, et utilisé dans méthode learn(Database)
arbre décisionnel classique	Un arbre de décision est un outil d'aide à la décision qui représente la situation plus ou moins complexe que l'on représente sous la forme graphique d'un arbre de façon à faire apparaître à l'extrémité de chaque branche (ou feuille) les différents résultats possibles en fonction des décisions prises à chaque étape.	Class DecisionTree

2.2 Dictionnaire de L'histogramme de gradient orienté (HOG)

Histogram of oriented gradients	features of image pixels which are calculated with 1-D derivative masks	Attribut _histogram de classe HOG
block normalization	a method to normalize the intensity of cells within blocks (larger region) it results in better invariance to changes in illumination and shadowing	method blockNormalization of class HOG
1-D derivative mask	kernels filtering the color or intensity data of images to compute gradient value usually applied vertically and horizontally which have forms like: $[-1, 0, 1]$ and $[-1, 0, 1]^T$. $G_h(x,y) = f(x+1,y) - f(x-1,y) \quad \forall x,$ $G_v(x,y) = f(x,y+1) - f(x,y-1) \quad \forall x,$	X
Image Filter (9.3 p.45)	a class that provides DSP Image filters	X
Image Utils (9.4 p.45)	a class that provides utilities such as resize image, get image data, save to buffer...etc	X
Integral Image (9.5 p.46)	a class that helps us compute integral images	X

3 Use Cases

3.1 Actors

Actor	Description
Utilisateur final	the final user is a human operator who choose the images and give it to the system for classification
jLevis	existing framework containing classification algorithms

Table 1 Table of Actors

3.2 Use Cases

Use-Cases	Description
Choisir les images	choose the images manually
Classification des images	give the chosen images for classification
Learn an image classifier	classify a set of images into different categories by random forest
Creation d'une forêt décisionnelle aléatoire	create a random forest which contains multiple decision trees and each tree vote for classification
choisir profondeur maximale des arbres de la forêt	choose the maximum depth of trees in the random forest
choisir nombre d'arbres de décision contenus dans la forêt aléatoire	choose the number of trees in this random tree
Creation d'une arbre de décision	create a flow-chart like structure which classifies information through each node, from root to leaf
Choisir un sous-ensemble aléatoire de données d'apprentissage	select a bootstrap sample of the training set and fits trees to these samples in the "bagging" process
Apprentissage d'une forêt	The training algorithm for random forests applies the general technique of bootstrap aggregating (bagging) to tree learners.
Fournir des données d'apprentissage	provide the data of the HoG process
Fournir HOG des images d'apprentissage	The technique which counts occurrences of gradient orientation in blocks of an image.
Informations sur la forêt	attributes of the random forest
Le nombre d'arbres dans la forêt construite	number of decision trees in the random forest
Calculer l'erreur	calculate the testing error number of trees have not been fit
Calculer l'erreur d'apprentissage	calculate the learning error number of trees have not been fit
Choisir classifier	choose a classifier
Charger et pré-traiter images	load the features of image pixels calculated by HOG
Pré-calculer HOG	use "Histogram of oriented gradients" (HOG) to calculate features of image pixels
Charger classifieur (depuis fichier)	load classifier from file
Sauvegarder classifieur (sur fichier)	Save the data and information of classifier

Use-Cases	Description
Choix du pour cent des variables à utiliser par chaque arbre	calculate the percentage of the variables to be used by each tree
Choix du subset de variables à utiliser	Choosing the subset of variables to be used as samples in a decision tree

Table 2 Table of Use Cases

3.2.1 Use Case "Avant projet Use Case diagram"

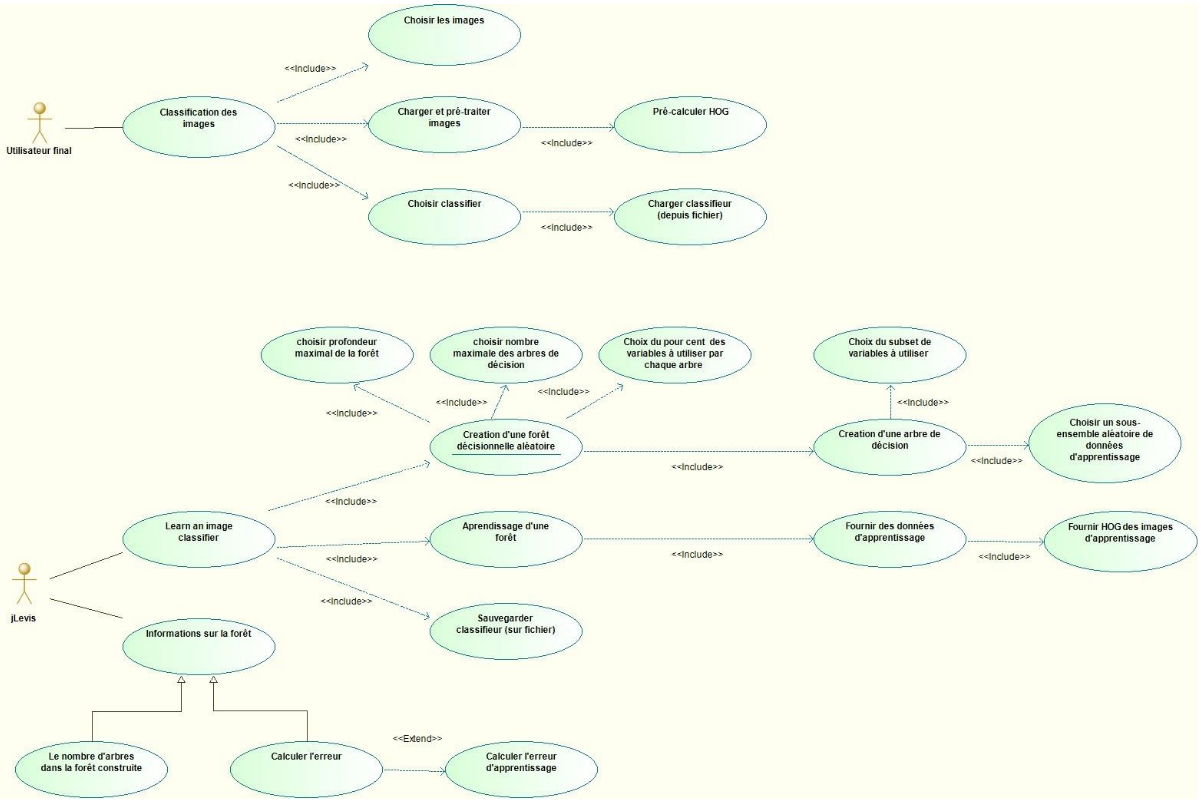

Choisir les images
<<Include>>
Utilisateur final
Classification des images
Charger et pré-traiter images
Pré-calculer HOG
<<Include>>
<<Include>>
<<Include>>
Choisir classifier
Charger classifieur (depuis fichier)
<<Include>>
choisir profondeur maximal de la forêt
choisir nombre maximale des arbres de décision
Choix du pour cent des variables à utiliser par chaque arbre
Choix du subset de variables à utiliser
<<Include>>
<<Include>>
<<Include>>
<<Include>>
Creation d'une forêt décisionnelle aléatoire
Creation d'une arbre de décision
Choisir un sous-ensemble aléatoire de données d'apprentissage
<<Include>>
<<Include>>
<<Include>>
Learn an image classifier
Aprendissage d'une forêt
Fournir des données d'apprentissage
Fournir HOG des images d'apprentissage
<<Include>>
<<Include>>
<<Include>>
<<Include>>
jLevis
Informations sur la forêt
Sauvegarder classifieur (sur fichier)
Le nombre d'arbres dans la forêt construite
Calculer l'erreur
<<Extend>>
Calculer l'erreur d'apprentissage

Figure 1 : Avant projet Use Case diagram

4 Package Index

RandomForestHOG	Main Package
HOG	Package responsible for creating HOG representations of images
DecisionTree	Package contains the classes used in decision tree
RandomForest	Package contains class RandomForest and class RandomForestLearner.
NotifyingThread	Implement listener for threading
Utils	Package contains utility classes

5 Package "RandomForestHOG"

Stereotypes: Java Package

Name	Summary
HOG	Package responsible for creating HOG representations of images
DecisionTree	Package contains the classes used in decision tree
RandomForest	Package contains class RandomForest and class RandomForestLearner.
NotifyingThread	Implement listener for threading

Table 3 Owned Packages of Package "RandomForestHOG"

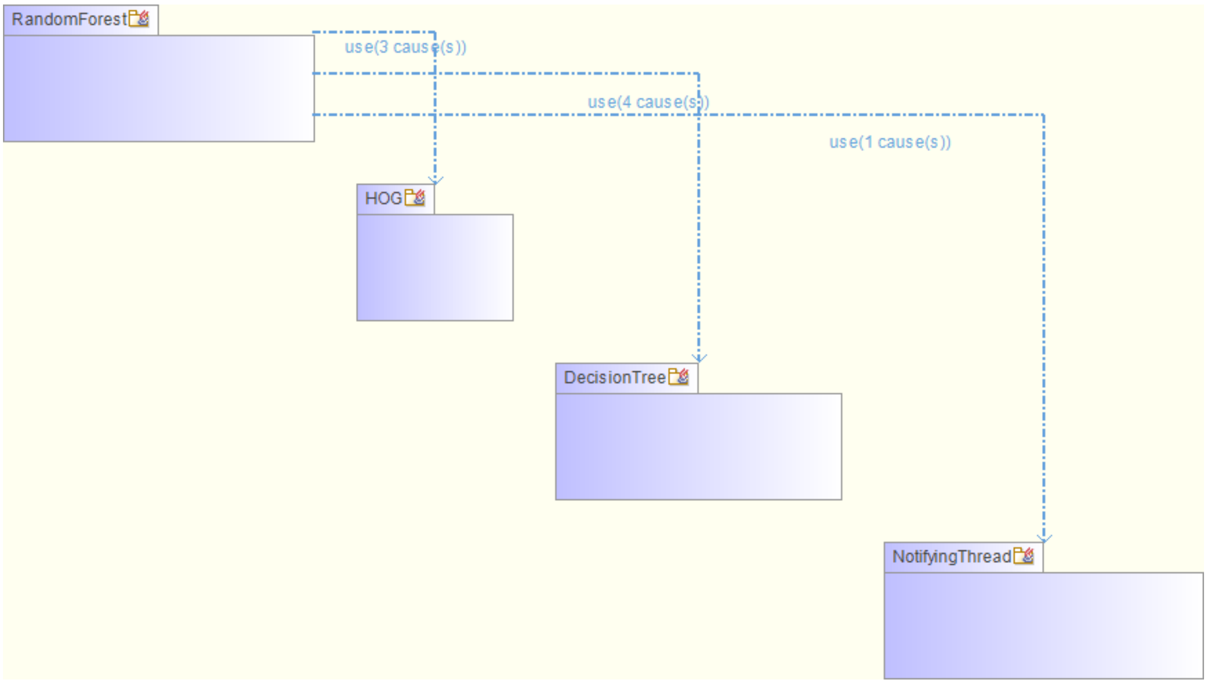


Figure 2 RandomForestHOG (subpackage_structure_autodiagram)

6 Package "HOG"

from Package RandomForestHOG

Stereotypes: Java Package

Name	Summary
HOGParam	Class contains a set of parameters for HOG
HOG	Class responsible for creating HOG representations of images
HOGAppli	convert an image into HOG data set

Table 4 Owned Classes of Package "HOG"

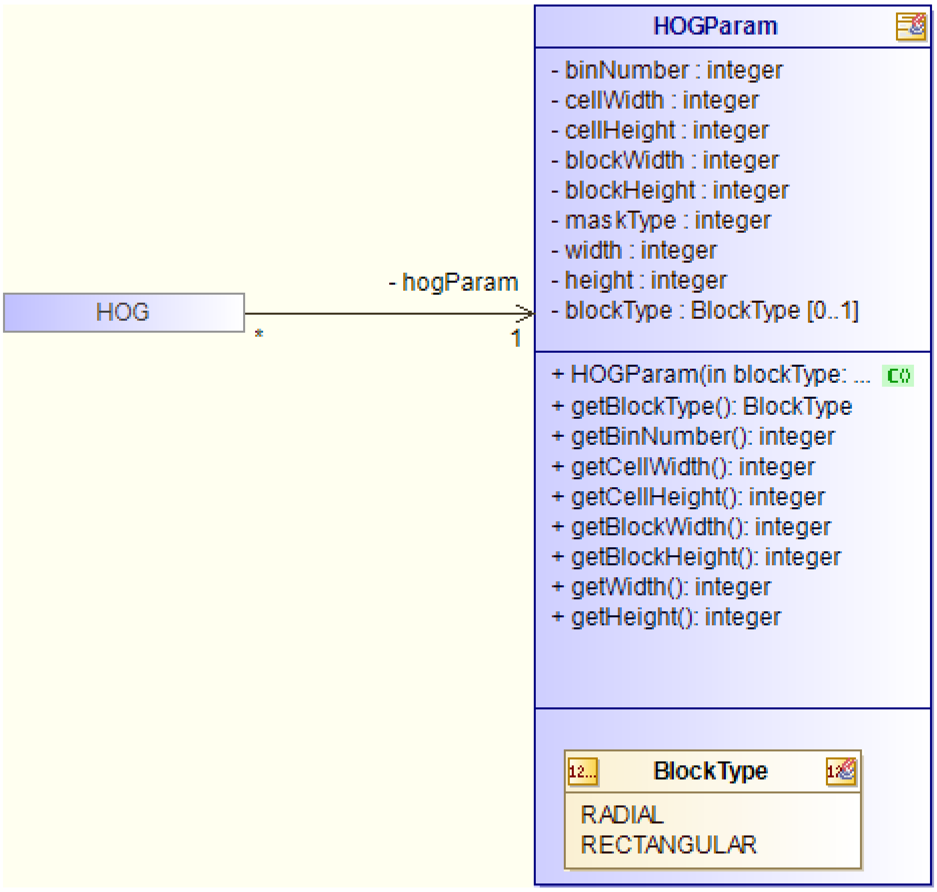
6.1 Class "HOGParam"

from Package .RandomForestHOG.HOG

Stereotypes: Java Class

This class bundles the essential parameters to perform HOG.

Figure 3 HOGParam



Name	Description
HOGParam (In blockType BlockType,In binNumber integer,In cellWidth integer,In cellHeight integer,In blockWidth integer,In blockHeight integer,In maskType integer,Inout width integer,Inout height integer)	initialize instance of HOGParam with manual setting
BlockType getBlockType ()	return blockType variable
integer getBinNumber ()	return binNumber variable
integer getCellWidth ()	return cellWidth variable
integer getCellHeight ()	return cellHeight variable
integer getBlockWidth ()	return blockWidth variable
integer getBlockHeight ()	return blockHeight variable
integer getWidth ()	return width variable
integer getHeight ()	return height variable

Table 5 Operations of Class "HOGParam"

Name	Description
nBin : [1..1] integer	number of bins for creating histogram
cellWidth : [1..1] integer	width of each cell
cellHeight : [1..1] integer	height of each cell
blockWidth : [1..1] integer	width of each block
blockHeight : [1..1] integer	height of each block
maskType : [1..1] integer	type of the mask to calculate gradients
width : [1..1] integer	number of pixels for the width of each image
height : [1..1] integer	number of pixels for the height of each image
blockType : [0..1] BlockType	shape of the block

Table 6 Attributes of Class "HOGParam"

Name	Values	Description
BlockType	RADIAL RECTANGULAR	BlockType contains two type, RADIAL and RECTANGULAR

Table 7 Owned Enumerations of Class "HOGParam"

6.2 Class "HOG"

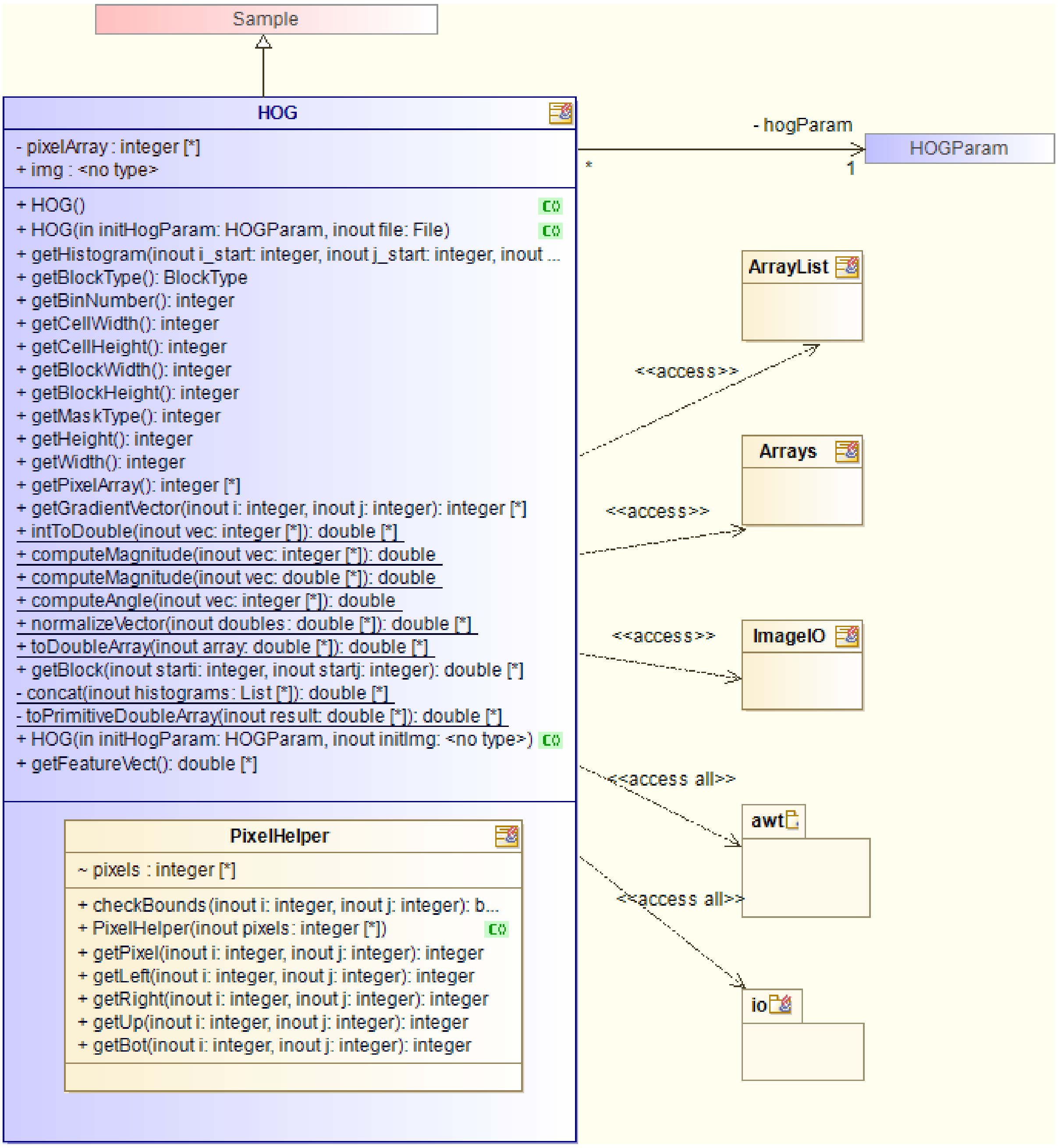
from Package .RandomForestHOG.HOG

Inherits from: Sample

Stereotypes: Java Class

This class implements the operations involved in HOG, and create arrays of pixel's data of each stage.

Figure 4 HOG


Sample
HOG
- pixelArray : integer [*]
+ img : <no type>
+ HOG()
+ HOG(in initHogParam: HOGParam, inout file: File)
+ getHistogram(inout i_start: integer, inout j_start: integer, inout ...
+ getBlockType(): BlockType
+ getBinNumber(): integer
+ getCellWidth(): integer
+ getCellHeight(): integer
+ getBlockWidth(): integer
+ getBlockHeight(): integer
+ getMaskType(): integer
+ getHeight(): integer
+ getWidth(): integer
+ getPixelArray(): integer [*]
+ getGradientVector(inout i: integer, inout j: integer): integer [*]
+ intToDouble(inout vec: integer [*]): double [*]
+ computeMagnitude(inout vec: integer [*]): double
+ computeMagnitude(inout vec: double [*]): double
+ computeAngle(inout vec: integer [*]): double
+ normalizeVector(inout doubles: double [*]): double [*]
+ toDoubleArray(inout array: double [*]): double [*]
+ getBlock(inout starti: integer, inout startj: integer): double [*]
- concat(inout histograms: List [*]): double [*]
- toPrimitiveDoubleArray(inout result: double [*]): double [*]
+ HOG(in initHogParam: HOGParam, inout initImg: <no type>)
+ getFeatureVect(): double [*]
PixelHelper
~ pixels : integer [*]
+ checkBounds(inout i: integer, inout j: integer): b...
+ PixelHelper(inout pixels: integer [*])
+ getPixel(inout i: integer, inout j: integer): integer
+ getLeft(inout i: integer, inout j: integer): integer
+ getRight(inout i: integer, inout j: integer): integer
+ getUp(inout i: integer, inout j: integer): integer
+ getBot(inout i: integer, inout j: integer): integer
- hogParam
*
1
HOGParam
ArrayList
<<access>>
Arrays
<<access>>
<<access>>
ImageIO
<<access all>>
awt
<<access all>>
io

Name	Description
HOG ()	initialize HOG instance with default parameter setting
HOG (In **initHogParam HOGParam,**Inout **file File)**	initialize HOG instance with certain parameter setting
HOG (In **initHogParam HOGParam,**Inout **initImg)**	
double getHistogram (Inout **i_start integer,**Inout **j_start integer,**Inout **end_i integer,**Inout **end_j integer)**	calculate the histogram of a block and return a double array
BlockType getBlockType ()	return blockType of HOGParam
integer getBinNumber ()	return binNumber of HOGParam
integer getCellWidth ()	return cellWidth of HOGParam
integer getCellHeight ()	return cellHeight of HOGParam
integer getBlockWidth ()	return blockWidth of HOGParam
integer getBlockHeight ()	return blockHeight of HOGParam
integer getMaskType ()	return maskType of HOGParam
integer getHeight ()	return height of HOGParam
integer getWidth ()	return width of HOGParam
integer getPixelArray ()	return the grayscale of the image
integer getGradientVector (Inout **i integer,**Inout **j integer)**	get the gradient vector of each pixel
double intToDouble (Inout **vec integer)**	convert an integer array to a double array
double computeMagnitude (Inout **vec integer)**	compute the magnitude of the gradient vector
double computeMagnitude (Inout **vec double)**	compute the magnitude of the gradient vector
double computeAngle (Inout **vec integer)**	compute the angle of the gradient vector
double normalizeVector (Inout **doubles double)**	normalize the gradient vector
double toDoubleArray (Inout **array double)**	convert primitive double array to Double array
double getBlock (Inout **starti integer,**Inout **startj integer)**	getting historams from each cell in a block
double concat (Inout **histograms List)**	concat all historam data in the same block into a double array
double toPrimitiveDoubleArray (Inout **result double)**	convert a Double array to a primitive double array
double getFeatureVect ()	return all the histogram data of an image

Table 8 Operations of Class "HOG"

Name	Description
_featureVector : [0..*] integer	array of feature vectors of each pixel after calculation
img : [1..1]	

Table 9 Attributes of Class "HOG"

Name	Description
->hogParam : [1..1] HOGParam	set of HOG parameters used in HOG computation

Table 10 Associations of Class "HOG"

Name	Summary
PixelHelper	

Table 11 Owned Classes of Class "HOG"

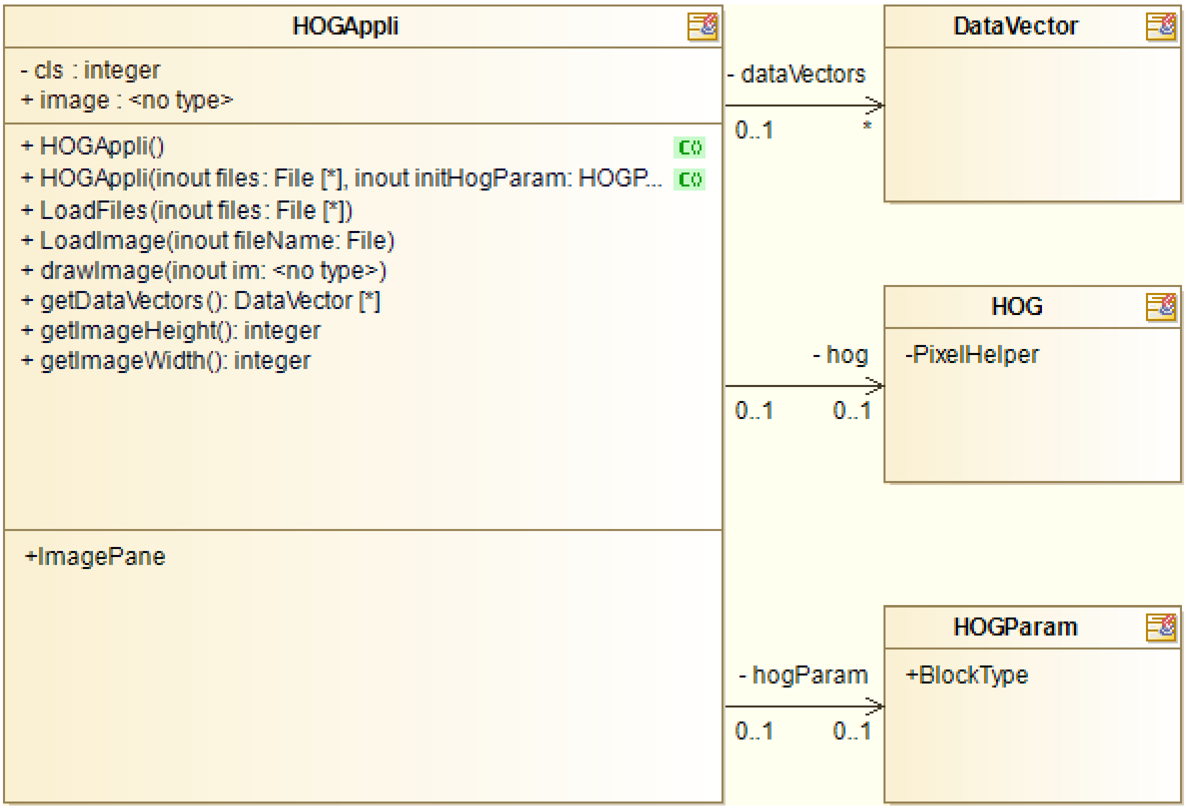
6.3 Class "HOGAppli"

from Package .RandomForestHOG.HOG

Stereotypes: Java Class

load files, caculate the HOGs and generate data sets

Figure 5 HOGAppli Class diagram


HOGAppli
- cls : integer
+ image : <no type>
+ HOGAppli()
+ HOGAppli(inout files : File [*], inout initHogParam: HOGP...
+ LoadFiles(inout files : File [*])
+ LoadImage(inout fileName: File)
+ drawImage(inout im: <no type>)
+ getDataVectors (): DataVector [*]
+ getImageHeight(): integer
+ getImageWidth(): integer
+ImagePane
- dataVectors
0..1
*
DataVector
- hog
0..1
0..1
HOG
-PixelHelper
- hogParam
0..1
0..1
HOGParam
+BlockType

Name	Description
HOGAppli ()	set up hogParam
HOGAppli (Inout **files File,**Inout **initHogParam HOGParam)**	input file and setup hogParam
LoadFiles (Inout **files File)**	recursively load all files in the input directory
LoadImage (Inout **fileName File)**	load and resize image and compute its HOG
drawImage (Inout **im)**	show image on the screen
DataVector getDataVectors ()	return dataVector variable
integer getImageHeight ()	return height of the image
integer getImageWidth ()	return width of the image

Table 12 Operations of Class "HOGAppli"

Name	Description
cls : [1..1] integer	the class of the input image
image : [1..1]	input image

Table 13 Attributes of Class "HOGAppli"

Name	Description
->dataVectors : [0..*] DataVector	list of data from HOG
->hog : [0..1] HOG	the HOG instance
->hogParam : [0..1] HOGParam	the hogParam instance

Table 14 Associations of Class "HOGAppli"

Name	Summary
ImagePane	ImagePane extends JPanel

Table 15 Owned Classes of Class "HOGAppli"

7 Package "DecisionTree"

from Package .RandomForestHOG

Stereotypes: Java Package

Create decision tree and classify testing data

Name	Summary
DecisionTree	Class contains interface for using Decision Trees.
TreeNode	Class defines the nodes of a tree data structure

Table 16 Owned Classes of Package "DecisionTree"

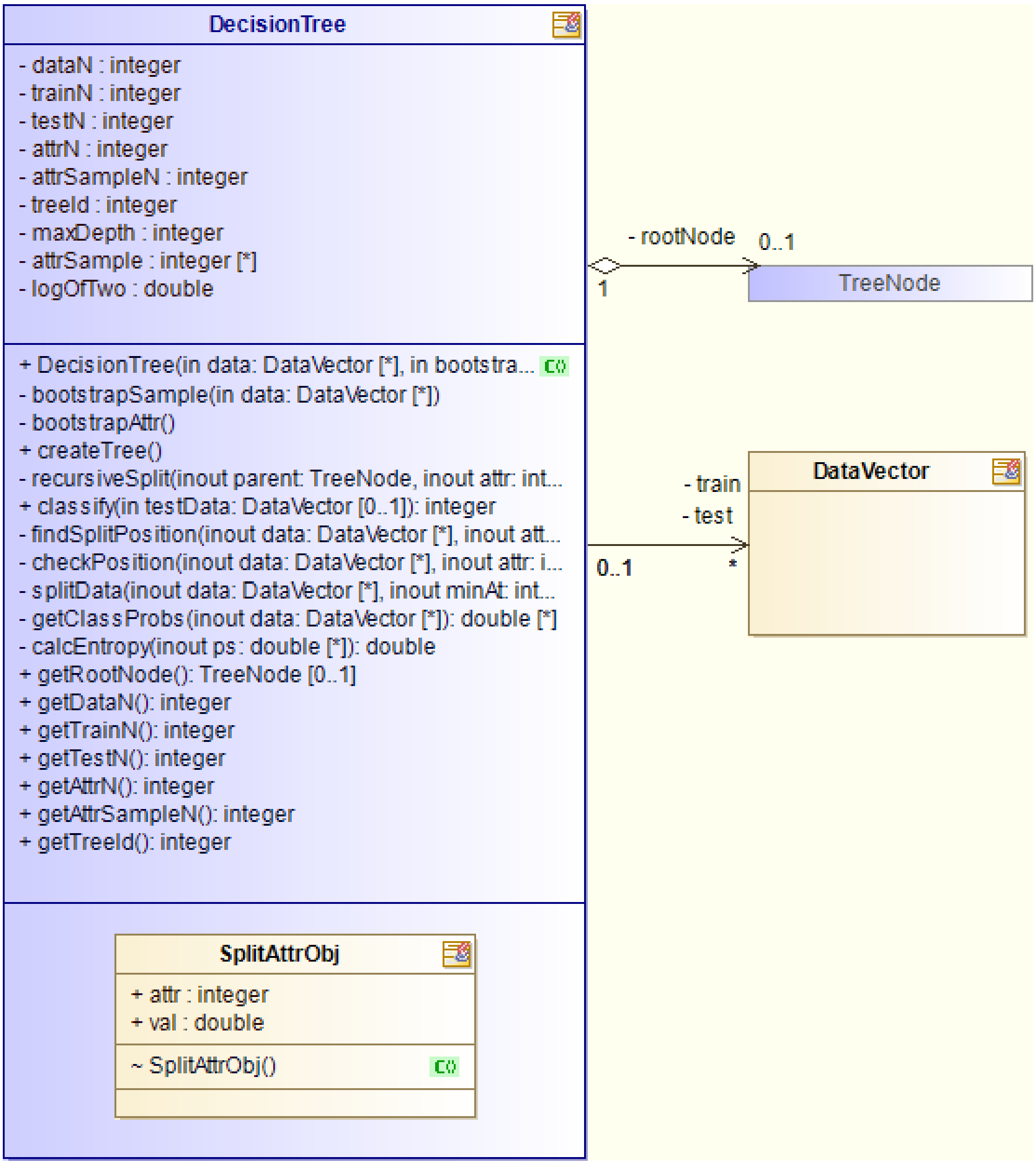
7.1 Class "DecisionTree"

from Package .RandomForestHOG.DecisionTree

Stereotypes: Java Class

This class implements operations about decision tree in random forest learning.

Figure 6 DecisionTree


DecisionTree
- dataN : integer
- trainN : integer
- testN : integer
- attrN : integer
- attrSampleN : integer
- treeId : integer
- maxDepth : integer
- attrSample : integer [*]
- logOfTwo : double
+ DecisionTree(in data: DataVector [*], in bootstra...
- bootstrapSample(in data: DataVector [*])
- bootstrapAttr()
+ createTree()
- recursiveSplit(inout parent: TreeNode, inout attr: int...
+ classify(in testData: DataVector [0..1]): integer
- findSplitPosition(inout data: DataVector [*], inout att...
- checkPosition(inout data: DataVector [*], inout attr: i...
- splitData(inout data: DataVector [*], inout minAt: int...
- getClassProbs(inout data: DataVector [*]): double [*]
- calcEntropy(inout ps: double [*]): double
+ getRootNode(): TreeNode [0..1]
+ getDataN(): integer
+ getTrainN(): integer
+ getTestN(): integer
+ getAttrN(): integer
+ getAttrSampleN(): integer
+ getTreeId(): integer
SplitAttrObj
+ attr : integer
+ val : double
~ SplitAttrObj()
- rootNode
0..1
1
TreeNode
- train
- test
0..1
*
DataVector

Name	Description
DecisionTree (In data DataVector,In bootstrapRate double,In attrSampleN integer,In maxDepth integer,In treeId integer)	constructs a decision tree from a data matrix.
bootStrapSample (In data DataVector)	create a boostrap sample of size trainN
bootStrapAttr ()	selects attributes of size attrN from sample
createTree ()	creates the decision tree according to the specifications of random forest trees
recursiveSplit (In parent TreeNode,Inout attr integer)	critical function to create the decision tree with the selected attributes of bootstrap sample
findSplitPosition (Inout data DataVector,Inout attr integer,Inout attrObj SplitAttrObj)	get attribute index and value to split data
double checkPosition (Inout data DataVector,Inout attr integer,Inout val double)	return the entropy of current split position
List splitData (Inout data DataVector,Inout minAt integer,Inout minAtVal double)	split data with given attribute index and value
double getClassProbs (Inout data DataVector)	get probabilities of classes
double calcEntropy (Inout ps double)	calculate the entropy given classes' probabilities
integer classify (In testData DataVector)	traverses the tree and returns the prediction of the given test data
TreeNode getRootNode ()	return rootNode
integer getDataN ()	return dataN
integer getTrainN ()	return trainN
integer getTestN ()	return testN
integer getAttrN ()	return attrN
integer getAttrSampleN ()	return attrSampleN
integer getTreeId ()	return treeId

Table 17 Operations of Class "DecisionTree"

Name	Description
dataN : [1..1] integer	total size of the data (training and testing)
trainN : [1..1] integer	size of the bootstrap samples to train (assigned by RFLearner)
attrN : [1..1] integer	size of the attributes to train (assigned by RFLearner)
attrN : [1..1] integer	size of all attributes
attrSampleN : [1..1] integer	size of bootstrapped attributes to train
treeId : [1..1] integer	the number of this tree
maxDepth : [1..1] integer	maximum level of decision tree
attrSample : [0..*] integer	an array of bootstrapped attributes' index
logOfTwo : [1..1] double	log 2

Table 18 Attributes of Class "DecisionTree"

Name	Description
->rootNode : [0..1] TreeNode	root node of this decision tree
->train : [0..*] DataVector	training dataset
->test : [0..*] DataVector	testing dataset

Table 19 Associations of Class "DecisionTree"

Name	Summary
SplitAttrObj	Attribute index and value

Table 20 Owned Classes of Class "DecisionTree"

7.2 Class "TreeNode"

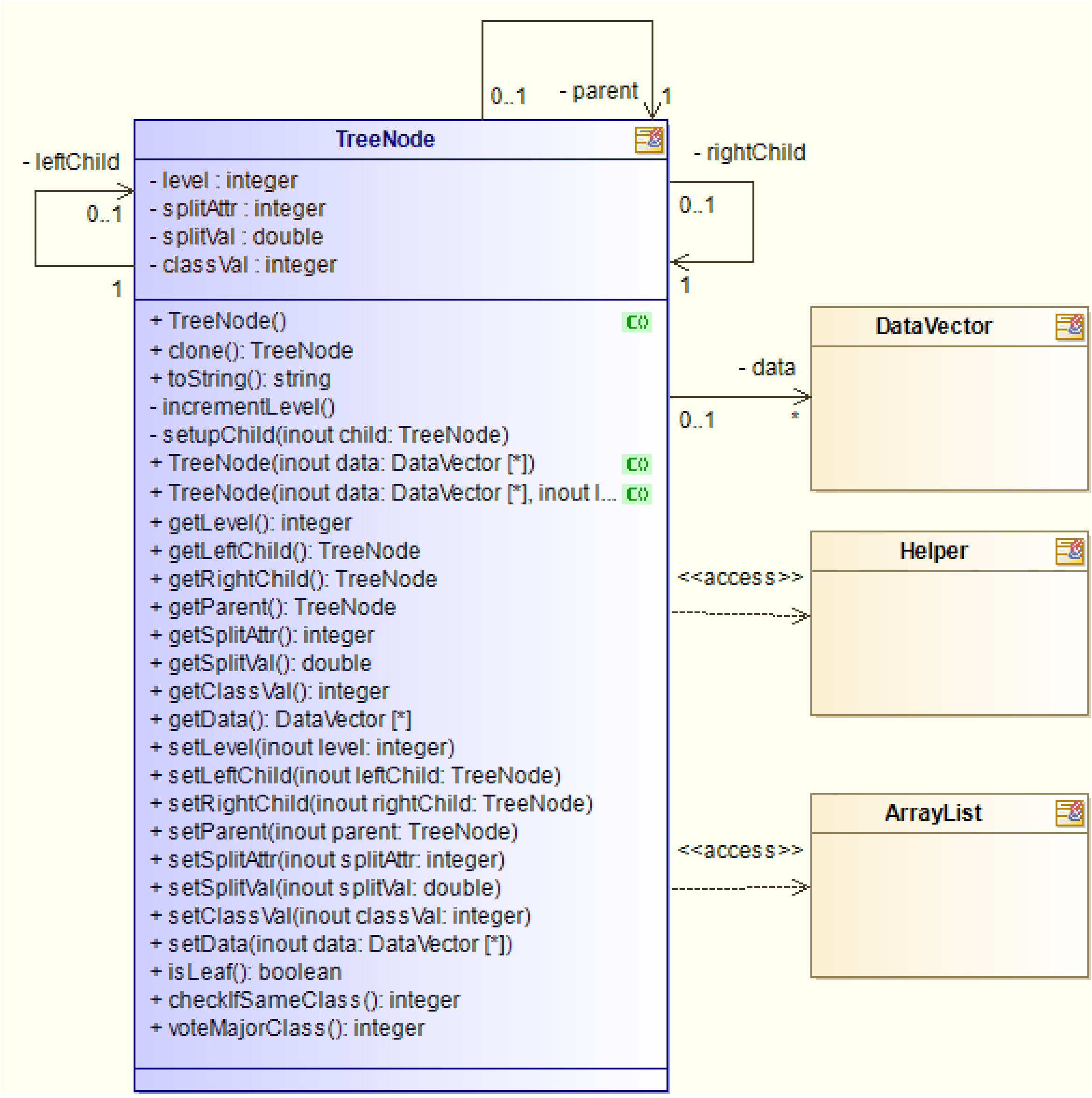
from Package .RandomForestHOG.DecisionTree

Implements: Cloneable

Stereotypes: Java Class

This class contains the elements of a node in tree data structure.

Figure 7 TreeNode


0..1
- parent
1
TreeNode
- leftChild
0..1
1
- rightChild
0..1
1
- level : integer
- splitAttr : integer
- splitVal : double
- classVal : integer
+ TreeNode()
+ clone(): TreeNode
+ toString(): string
- incrementLevel()
- setupChild(inout child: TreeNode)
+ TreeNode(inout data: DataVector [*])
+ TreeNode(inout data: DataVector [*], inout l...
+ getLevel(): integer
+ getLeftChild(): TreeNode
+ getRightChild(): TreeNode
+ getParent(): TreeNode
+ getSplitAttr(): integer
+ getSplitVal(): double
+ getClassVal(): integer
+ getData(): DataVector [*]
+ setLevel(inout level: integer)
+ setLeftChild(inout leftChild: TreeNode)
+ setRightChild(inout rightChild: TreeNode)
+ setParent(inout parent: TreeNode)
+ setSplitAttr(inout splitAttr: integer)
+ setSplitVal(inout splitVal: double)
+ setClassVal(inout classVal: integer)
+ setData(inout data: DataVector [*])
+ isLeaf(): boolean
+ checkIfSameClass(): integer
+ voteMajorClass(): integer
DataVector
- data
0..1
*
Helper
<<access>>
ArrayList
<<access>>

Name	Description
TreeNode ()	constructs the tree node
TreeNode (Inout **data DataVector)**	constructs the tree node
TreeNode (Inout **data DataVector,**Inout **level integer,**Inout **splitAttr integer,**Inout **splitVal double)**	constructs the tree node
TreeNode clone ()	copy TreeNode instance
string toString ()	display TreeNode as string
incrementLevel ()	increment level
setupChild (Inout **child TreeNode)**	set up child nodes
integer getLevel ()	return level
TreeNode getLeftChild ()	return leftChild
TreeNode getRightChild ()	return rightChild
TreeNode getParent ()	return parent
integer getSplitAttr ()	return splitAttr
double getSplitVal ()	return splitVal
integer getClassVal ()	return classVal
DataVector getData ()	return data
setLevel (Inout **level integer)**	set level
setLeftChild (Inout **leftChild TreeNode)**	set leftChild
setRightChild (Inout **rightChild TreeNode)**	set rightChild
setParent (Inout **parent TreeNode)**	set parent
setSplitAttr (Inout **splitAttr integer)**	set splitAttr
setSplitVal (Inout **splitVal double)**	set splitVal
setClassVal (Inout **classVal integer)**	set classVal
setData (Inout **data DataVector)**	set data
boolean isLeaf ()	check if current node is leaf
integer checkIfSameClass ()	check if data is the same class
integer voteMajorClass ()	get majority of the class of current data

Table 21 Operations of Class "TreeNode"

Name	Description
level : [1..1] integer	current level in the tree
splitAttr : [1..1] integer	attribute to split on
splitVal : [1..1] double	value to split splitAttr
classVal : [1..1] integer	

Table 22 Attributes of Class "TreeNode"

Name	Description
->leftChild : [0..1] TreeNode	left child of this node
->rightChild : [1..1] TreeNode	right child of this node
->parent : [1..1] TreeNode	parent of this node
->data : [0..*] DataVector	data at current level

Table 23 Associations of Class "TreeNode"

8 Package "RandomForest"

from Package *.RandomForestHOG*

Stereotypes: Java Package

Name	Summary
RandomForest	Class define the classifier generated by random forest.
RandomForestLearner	Class responsible for random forest learning
MainRun	Main class

Table 24 Owned Classes of Package "RandomForest"

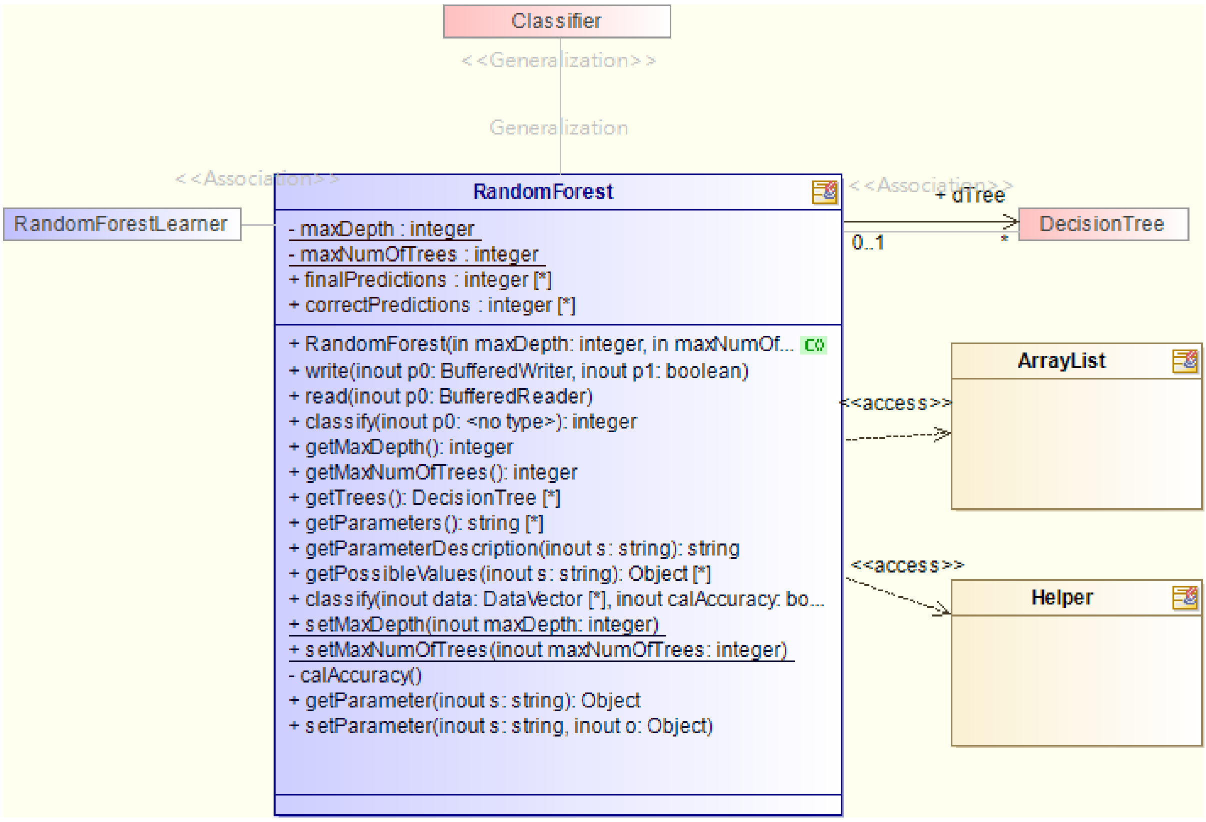
8.1 Class "RandomForest"

from Package .RandomForestHOG.RandomForest

Stereotypes: Java Class

This class contains information about the classifier trained with random forest and provides operation to classify images.

Figure 8 RandomForest


Classifier
<<Generalization>>
Generalization
<<Association>>
RandomForestLearner
RandomForest
- maxDepth : integer
- maxNumOfTrees : integer
+ finalPredictions : integer [*]
+ correctPredictions : integer [*]
+ RandomForest(in maxDepth: integer, in maxNumOf...
+ write(inout p0: BufferedWriter, inout p1: boolean)
+ read(inout p0: BufferedReader)
+ classify(inout p0: <no type>): integer
+ getMaxDepth(): integer
+ getMaxNumOfTrees(): integer
+ getTrees(): DecisionTree [*]
+ getParameters(): string [*]
+ getParameterDescription(inout s: string): string
+ getPossibleValues(inout s: string): Object [*]
+ classify(inout data: DataVector [*], inout calAccuracy: bo...
+ setMaxDepth(inout maxDepth: integer)
+ setMaxNumOfTrees(inout maxNumOfTrees: integer)
- calAccuracy()
+ getParameter(inout s: string): Object
+ setParameter(inout s: string, inout o: Object)
<<Association>>
+ dTree
0..1
*
DecisionTree
<<access>>
ArrayList
<<access>>
Helper

Name	Description
RandomForest (In maxDepth integer,In maxNumOfTrees integer)	the constructor of random forest
write (Inout p0 BufferedWriter,Inout p1 boolean)	implement write method of Classifier
read (Inout p0 BufferedReader)	implement read method of Classifier
integer classify (Inout p0)	classify dataset p0
integer classify (Inout data DataVector,Inout calAccuracy boolean)	classify dataset and calculate accuracy
integer getMaxDepth ()	return maxDepth
integer getMaxNumOfTrees ()	return maxNumOfTrees
DecisionTree getTrees ()	return dTree
string getParameters ()	return forest parameter
string getParameterDescription (Inout s string)	return forest parameter description
Object getPossibleValues (Inout s string)	return possible values
setMaxDepth (Inout maxDepth integer)	set maxDepth
setMaxNumOfTrees (Inout maxNumOfTrees integer)	set maxNumOfTrees
calAccuracy ()	calculate accuracy of classified predictions
Object getParameter (Inout s string)	return forest parameters
setParameter (Inout s string,Inout o Object)	set forest parameters

Table 25 Operations of Class "RandomForest"

Name	Description
maxDepth : [1..1] integer	maximum depth of tree, set to -1 if depth has no limit
maxNumOfTrees : [1..1] integer	maximum number of trees in random forest
finalPredictions : [0..*] integer	forest predictions of test data
correctPredictions : [0..*] integer	actual class of test data

Table 26 Attributes of Class "RandomForest"

Name	Description
->dTree : [0..*] DecisionTree	trees in random forest

Table 27 Associations of Class "RandomForest"

8.2 Class "RandomForestLearner"

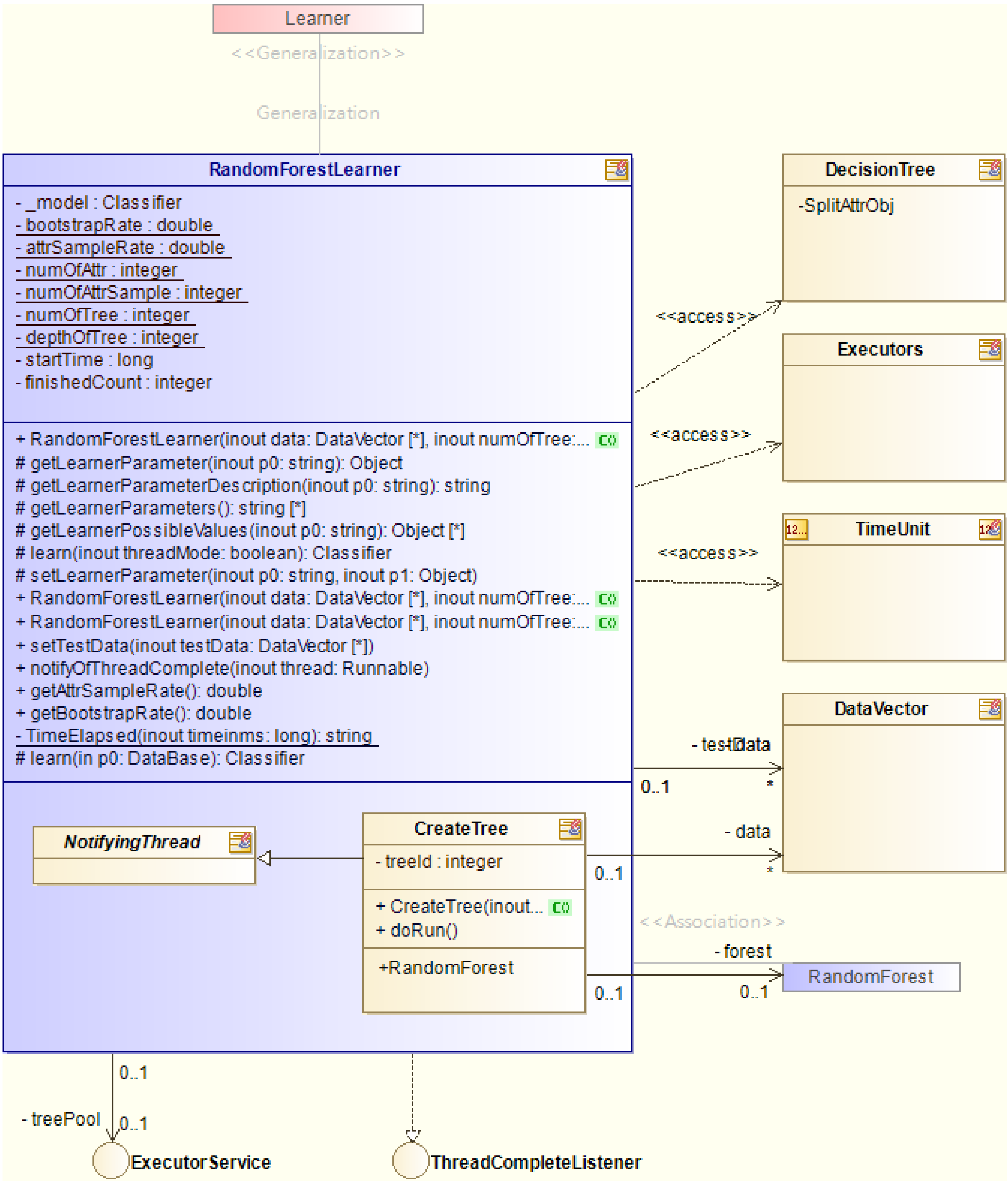
*from Package .RandomForestHOG.*RandomForest

Implements: ThreadCompleteListener

Stereotypes: Java Class

This class implements the learning algorithm of random forest inheriting from learner

Figure 9 RandomForestLearner


Learner
<<Generalization>>
Generalization
RandomForestLearner
- _model : Classifier
- bootstrapRate : double
- attrSampleRate : double
- numOfAttr : integer
- numOfAttrSample : integer
- numOfTree : integer
- depthOfTree : integer
- startTime : long
- finishedCount : integer
+ RandomForestLearner(inout data: DataVector [*], inout numOfTree:...
# getLearnerParameter(inout p0: string): Object
# getLearnerParameterDescription(inout p0: string): string
# getLearnerParameters(): string [*]
# getLearnerPossibleValues(inout p0: string): Object [*]
# learn(inout threadMode: boolean): Classifier
# setLearnerParameter(inout p0: string, inout p1: Object)
+ RandomForestLearner(inout data: DataVector [*], inout numOfTree:...
+ RandomForestLearner(inout data: DataVector [*], inout numOfTree:...
+ setTestData(inout testData: DataVector [*])
+ notifyOfThreadComplete(inout thread: Runnable)
+ getAttrSampleRate(): double
+ getBootstrapRate(): double
- TimeElapsed(inout timeinms: long): string
# learn(in p0: DataBase): Classifier
NotifyingThread
CreateTree
- treeId : integer
+ CreateTree(inout...
+ doRun()
+RandomForest
DecisionTree
-SplitAttrObj
<<access>>
Executors
<<access>>
TimeUnit
<<access>>
DataVector
- testData
0..1
*
- data
0..1
*
<<Association>>
- forest
0..1
0..1
RandomForest
0..1
- treePool
0..1
ExecutorService
ThreadCompleteListener

Name	Description
RandomForestLearner (Inout **data DataVector,**Inout **numOfTree integer,**Inout **depthOfTree integer)**	The constructor of RandomForestLearner
RandomForestLearner (Inout **data DataVector,**Inout **numOfTree integer,**Inout **depthOfTree integer,**Inout **attrSampleRate double)**	constructor of RandomForestLearner
RandomForestLearner (Inout **data DataVector,**Inout **numOfTree integer,**Inout **depthOfTree integer,**Inout **attrSampleRate double,**Inout **bootstrapRate double)**	constructor of RandomForestLearner
Object getLearnerParameter (Inout **p0 string)**	implements getLearnerParameter method of Learner
string getLearnerParameterDescription (Inout **p0 string)**	implements getLearnerParameterDescription method of Learner
string getLearnerParameters ()	implements getLearnerParameters method of Learner
Object getLearnerPossibleValues (Inout **p0 string)**	implements getLearnerPossibleValues method of Learner
Classifier learn (Inout **threadMode boolean)**	implements learn method of Learner and trains a random forest classifier
Classifier learn (In **p0 DataBase)**	train decision trees of random forest
setLearnerParameter (Inout **p0 string,**Inout **p1 Object)**	implements setLearnerParameter method of Learner
setTestData (Inout **testData DataVector)**	set testData
notifyOfThreadComplete (Inout **thread Runnable)**	callback method when thread finished
double getAttrSampleRate ()	return attrSampleRate
double getBootstrapRate ()	return bootstrapRate
string TimeElapsed (Inout **timeinms long)**	calculate time elapsed after timeinms

Table 28 Operations of Class "RandomForestLearner"

Name	Description
_model : [1..1] Classifier	random forest mode
bootstrapRate : [1..1] double	ratio to split training and testing data
attrSampleRate : [1..1] double	ratio of bootstrapped attribute sample
numOfAttr : [1..1] integer	number of all attributes
numOfAttrSample : [1..1] integer	number of attribute samples
numOfTree : [1..1] integer	number of trees in random forest
depthOfTree : [1..1] integer	maximum depth of decision tree
startTime : [1..1] long	starting time of learn()
finishedCount : [1..1] integer	count of finished thread

Table 29 Attributes of Class "RandomForestLearner"

Name	Description
->data : [0..*] DataVector	training dataset
->testData : [0..*] DataVector	testing dataset
->treePool : [0..1] ExecutorService	thread pool for learning

Table 30 Associations of Class "RandomForestLearner"

Name	Summary
CreateTree	Create decision tree
NotifyingThread	Implement listener for threading
DataVector	

Table 31 Owned Classes of Class "RandomForestLearner"

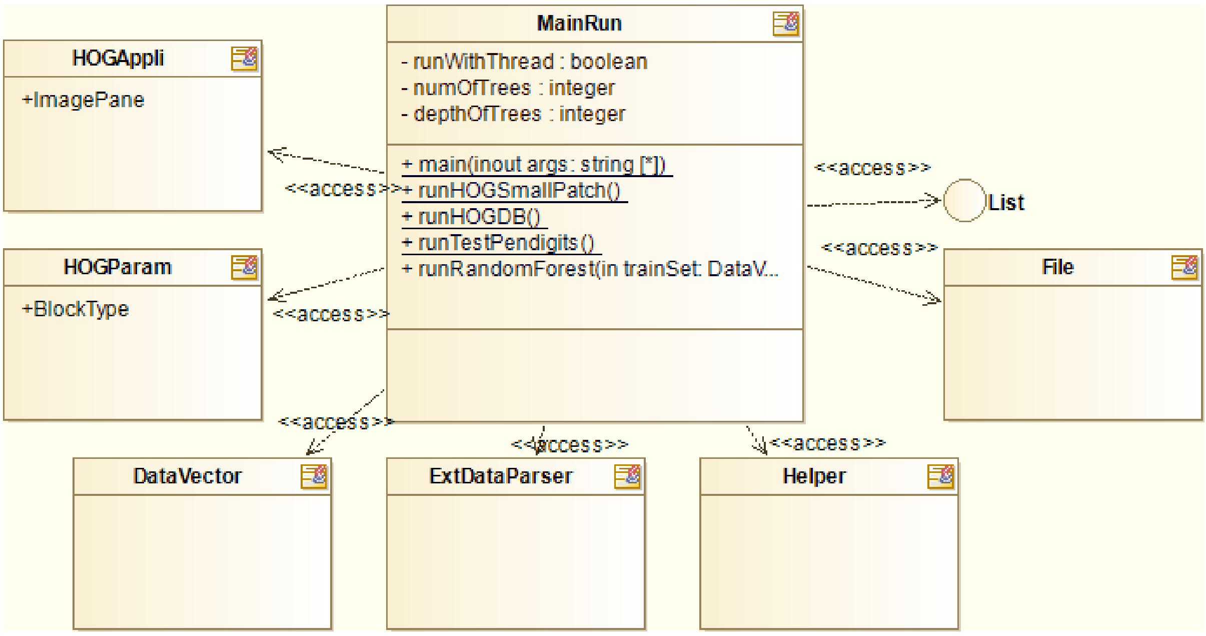
8.3 Class "MainRun"

from Package .RandomForestHOG.[*RandomForest*]

Stereotypes: Java Class

main running class with general testing

Figure 10 MainRun Class diagram



Name	Description
main (In**out args string)**	main method to start general test
runHOGSmallPatch ()	test small patch of images
runHOGDB ()	test images of DB
runTestPendigits ()	test handwritten digits
runRandomForest (In **trainSet DataVector,**In **testSet DataVector)**	start random forest learning

Table 32 Operations of Class "MainRun"

Name	Description
runWithThread : [1..1] boolean	set to train random forest with threads
numOfTrees : [1..1] integer	number of decision trees in random forest
depthOfTrees : [1..1] integer	maximum depth of decision tree

Table 33 Attributes of Class "MainRun"

9 Package "NotifyingThread"

from Package *.RandomForestHOG*

Stereotypes: Java Package

contain abstract class and interface for setting up listener of threads

Name	Summary
ThreadCompleteListener	Interface for thread listeners

Table 34 Owned Interfaces of Package "NotifyingThread"

Name	Summary
NotifyingThread	Implement listener for threading

Table 35 Owned Classes of Package "NotifyingThread"

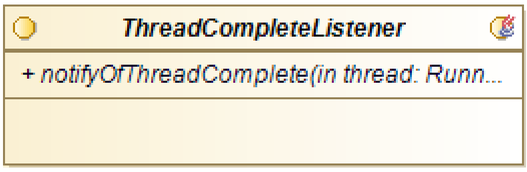
9.1 Interface "ThreadCompleteListener"

from Package .RandomForestHOG.[NotifyingThread]

Stereotypes: Java Interface

Notify listeners when thread finished

Figure 11 ThreadCompleteListener Class diagram



Name	Description
notifyOfThreadComplete (In thread Runnable)	notify listener when thread finished

Table 36 Operations of Interface "ThreadCompleteListener"

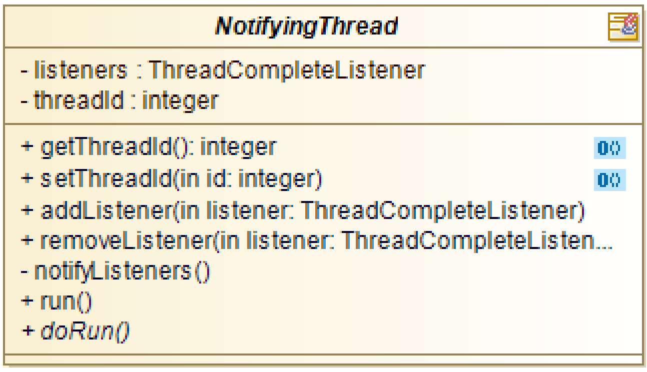
9.2 Class "NotifyingThread"

from Package .RandomForestHOG.NotifyingThread

Stereotypes: Java Class

Implement Runnable as abstract class to set up listener of threads

Figure 12 NotifyingThread Class diagram


NotifyingThread
- listeners : ThreadCompleteListener
- threadId : integer
+ getThreadId(): integer
+ setThreadId(in id: integer)
+ addListener(in listener: ThreadCompleteListener)
+ removeListener(in listener: ThreadCompleteListen...
- notifyListeners()
+ run()
+ doRun()

Name	Description
integer getThreadId ()	return threadId
setThreadId (In id integer)	set threadId
addListener (In listener ThreadCompleteListener)	add new listener when starting new thread
removeListener (In listener ThreadCompleteListener)	remove listener
notifyListeners ()	notify all listeners when current thread finished
run ()	implement run of Runnable
doRun ()	actual run method to implement

Table 37 Operations of Class "NotifyingThread"

Name	Description
listeners : [1..1] ThreadCompleteListener	listeners of finished thread
threadId : [1..1] integer	id of current thread

Table 38 Attributes of Class "NotifyingThread"

10Package "Utils"

Stereotypes: Java Package

Name	Summary
ExtDataParser	Parser for external testing data
Helper	General helper methods
DataVector	Customed format for data vector

Table 39 Owned Classes of Package "Utils"

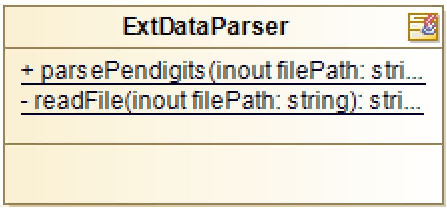
10.1 Class "ExtDataParser"

from Package .[Utils]

Stereotypes: Java Class

Various parser for testing data in assets/external folder

Figure 13 ExtDataParser Class diagram


ExtDataParser
+ parsePendigits(inout filePath: stri...
- readFile(inout filePath: string): stri...

Name	Description
DataVector parsePendigits (Inout **filePath string,**Inout **type integer)**	parse pendigits dataset
string readFile (Inout **filePath string)**	read dataset file given filePath

Table 40 Operations of Class "ExtDataParser"

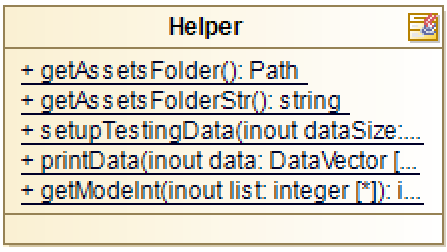
10.2 Class "Helper"

from Package .Utils

Stereotypes: Java Class

contain utility methods for all classes

Figure 14 Helper Class diagram


Helper
+ getAssetsFolder(): Path
+ getAssetsFolderStr(): string
+ setupTestingData(inout dataSize:...
+ printData(inout data: DataVector [...
+ getModeInt(inout list: integer [*]): i...

Name	Description
Path getAssetsFolder ()	return Path object of asset folder
string getAssetsFolderStr ()	return string of path of asset folder
DataVector setupTestingData (Inout **dataSize integer,**Inout **attrSize integer,**Inout **classSize integer)**	generate testing data with format compatible with DecisionTree
printData (Inout **data DataVector)**	print list of DataVector
integer getModeInt (Inout **list integer)**	return the mode of an integer list

Table 41 Operations of Class "Helper"

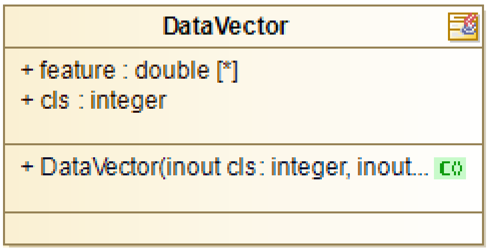
10.3 Class "DataVector"

from Package .Utils

Stereotypes: Java Class

contain class and features of a data vector

Figure 15 DataVector Class diagram



Name	Description
DataVector (Inout **cls integer,**Inout **feature double)**	constructor of DataVector

Table 42 Operations of Class "DataVector"

Name	Description
feature : [0..*] double	features (attributes) of data vector
cls : [1..1] integer	class (label) of data vector

Table 43 Attributes of Class "DataVector"