Gradient Boosting Machine

Generated by Doxygen 1.8.9.1

Mon Jun 22 2015 09:31:22

Contents

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

Class Documentation

2.1 BinaryTree Class Reference

Public Member Functions

- BinaryTree ()
- void generateTree (unsigned int classLabel, DataSet ∗data)
- ∼BinaryTree ()
- float computeVariance (float value, float sumSQValue, float weight)
- int getNumberOfRegions ()
- float makeAPrediction (const float ∗example) const
- void saveToBinaryFile () const
- void readFromBinaryFile ()
- void readFromBinaryFile (std::ifstream &mFile)
- void saveToBinaryFile (std::ofstream &oFile) const
- void print (const DataSet ∗data)
- void printToFile (const DataSet ∗data)
- float getGamma (const float ∗example) const

2.1.1 Constructor & Destructor Documentation

2.1.1.1 BinaryTree::BinaryTree ( )

The object constructor.

2.1.1.2 BinaryTree::∼BinaryTree ( )

The object destructor.

2.1.2 Member Function Documentation

2.1.2.1 float BinaryTree::computeVariance (float *value,* float *sumSQValue,* float *weight*)

It computes the variance of a single instance.

Parameters

value	is the value for computing a sample mean.
sumSQValue	is the sum of squared values.
weight	is the normalization term.

Returns

the sample variance.

2.1.2.2 void BinaryTree::generateTree (unsigned int *classLabel,* DataSet * *data*)

It builds the tree.

Parameters

classLabel	is the class for the tree.
data	is the data to fit with a tree.

2.1.2.3 float BinaryTree::getGamma (const float * *example*) const

It gets the gamma value for the given example.

Parameters

example	is the given example.

2.1.2.4 int BinaryTree::getNumberOfRegions ( )

It returns the number of regions Ri in the tree.

Returns

the number of regions Ri of the tree.

2.1.2.5 float BinaryTree::makeAPrediction (const float * *example*) const

It makes a prediction of the expected value.

Parameters

example	is the given example to predict.

Returns

the expected value.

2.1.2.6 void BinaryTree::print (const DataSet * *data*)

It prints by command line the model.

Parameters

data	is the data set.

2.1.2.7 void BinaryTree::printToFile (const DataSet * *data*)

It copies to a string the model.

Parameters

data	is the data set.

2.1.2.8 void BinaryTree::readFromBinaryFile ( )

It reads the model to a file.

2.1.2.9 void BinaryTree::readFromBinaryFile (std::ifstream & *mFile*)

It reads the file.

Parameters

mFile	is the model file to read.

2.1.2.10 void BinaryTree::saveToBinaryFile ( ) const

It writes the model to a file.

2.1.2.11 void BinaryTree::saveToBinaryFile (std::ofstream & *oFile*) const

It writes the model to a file.

Parameters

oFile	is the output file.

The documentation for this class was generated from the following files:

- BinaryTree.h
- BinaryTree.cpp

2.2 Config Class Reference

Static Public Member Functions

- static void print ()

Static Public Attributes

- static const unsigned int MAXBUFFER = 3000
- static const int NOTVALID = std::numeric_limits<int>::min()
- static const float UNKNOWNVALUE = std::numeric_limits<float>::max()
- static float MINVARIANCE = 0.000001f

- static int MINIMNUMBEROFINSTANCESPERLEAF = 4
- static int MAXIMUMDEPTH = 1
- static int MAXIMUMNUMBEROFLEAVES = 0
- static std::string inputFile = "../GradientBoosting/DataSet/notas/notas-10-1tst.dat"
- static std::string modelFile = "model.bin"
- static float learningRate = 0.1f
- static int numberOfTrees = 10
- static bool verbose = false
- static bool multipleExperiments = false
- static float * classWeight = NULL
- static bool applyWeight = false

2.2.1 Member Function Documentation

2.2.1.1 void Config::print ( ) [static]

It prints the configuration.

2.2.2 Member Data Documentation

2.2.2.1 bool Config::applyWeight = false [static]

Apply weight.

2.2.2.2 float * Config::classWeight = NULL [static]

It weights.

2.2.2.3 std::string Config::inputFile = "../GradientBoosting/DataSet/notas/notas-10-1tst.dat" [static]

Data set file name.

2.2.2.4 float Config::learningRate = 0.1f [static]

The learning rate of the GBM.

2.2.2.5 const unsigned int Config::MAXBUFFER = 3000 [static]

The maximum amount of characters in a char* string.

2.2.2.6 int Config::MAXIMUMDEPTH = 1 [static]

It indicates the maximum depth of the tree.

2.2.2.7 int Config::MAXIMUMNUMBEROFLEAVES = 0 [static]

It indicates the maximum number of leaves in the tree.

2.2.2.8 int Config::MINIMNUMBEROFINSTANCESPERLEAF = 4 [static]

It indicates the minimum number of instances that a leaf node can have.

2.2.2.9 float Config::MINVARIANCE = 0.000001f [static]

It indicates the minimum variance required for split.

2.2.2.10 std::string Config::modelFile = "model.bin" [static]

The model file.

2.2.2.11 bool Config::multipleExperiments = false [static]

Prepares te algorithm to launch multiple experiments.

2.2.2.12 const int Config::NOTVALID = std::numeric_limits<int>::min() [static]

It indicates a non valid feature value.

2.2.2.13 int Config::numberOfTrees = 10 [static]

The number of trees (rounds).

2.2.2.14 const float Config::UNKNOWNVALUE = std::numeric_limits<float>::max() [static]

An unknown value.

2.2.2.15 bool Config::verbose = false [static]

It prints the tree.

The documentation for this class was generated from the following files:

- Config.h
- Config.cpp

2.3 DataSet Class Reference

Public Member Functions

- DataSet (const std::string &fileName)
- DataSet (const DataSet *dataset)
- DataSet (unsigned int feature, const std::string &symbol, float threshold, const DataSet *dataset)
- ~DataSet ()
- void sort (unsigned int feature)
- void print () const
- unsigned int **getNumberOfInstances** () const
- unsigned int **getNumberOfClasses** () const
- unsigned int **getNumberOfFeatures** () const
- unsigned int **getClassPosition** () const
- float **getValueAt** (unsigned int index, unsigned int feature) const
- float getExampleAt (unsigned int index, float *example) const
- float getWeight () const
- float getResidualAt (unsigned int index, unsigned int label) const

- void setResidualAt (unsigned int index, unsigned int label, float value)
- float getOutputVariableAt (unsigned int index) const
- int getNumberOfInstancesWithThresholdAt (unsigned int feature, const std::string &symbol, float threshold) const
- std::string **getFeatureName** (unsigned int index) const
- void computeResidual (unsigned int exampleIndex, unsigned int classIndex, float value)
- int **getNumberOfInstancesOfClass** (int classLabel)

2.3.1 Constructor & Destructor Documentation

2.3.1.1 DataSet::DataSet (const std::string & *fileName*)

The data set constructor.

Parameters

fileName	is the name of the file to open.

2.3.1.2 DataSet::DataSet (const DataSet * *dataset*)

Copy constructor.

Parameters

dataset	is the data set to clone.

2.3.1.3 DataSet::DataSet (unsigned int *feature*, const std::string & *symbol*, float *threshold*, const DataSet * *dataset*)

It copies the desired number of instances with threshold.

Parameters

feature	is the desired feature to copy data from.
symbol	is the desired symbol [<= | >].
threshold	is the threshold value.
dataset	is the data set to clone.

2.3.1.4 DataSet::~DataSet ( )

The data set destructor.

2.3.2 Member Function Documentation

2.3.2.1 void DataSet::computeResidual (unsigned int *exampleIndex*, unsigned int *classIndex*, float *value*)

It computes the residual for the given class.

Parameters

exampleIndex	is the index of the desired example.
classIndex	is the index of the desired class.

value	is the value to substract to the residual.

2.3.2.2 float DataSet::getExampleAt (unsigned int *index,* float * *example*) const

It gets the example at the given position and example.

2.3.2.3 int DataSet::getNumberOfInstancesWithThresholdAt (unsigned int *feature,* const std::string & *symbol,* float *threshold*) const

It counts the number of instances which follow a given threshold.

Parameters

feature	is the feature index.
symbol	is the symbol [$<=$ | $>$].
threshold	is the threshold value.

2.3.2.4 float DataSet::getOutputVariableAt (unsigned int *index*) const [inline]

It gets the output variable (class) at the given position.

Parameters

index	is the index of the data matrix.

Returns

the desired output variable.

It gets the output variable.

2.3.2.5 float DataSet::getResidualAt (unsigned int *index,* unsigned int *label*) const

It gets the residual at the given position and label.

Parameters

index	is the index of the data matrix.
label	is the index of the desired label.

Returns

the desired residual.

2.3.2.6 float DataSet::getWeight ( ) const [inline]

Weight per example.

It gets the weight of each example.

2.3.2.7 void DataSet::print ( ) const

It prints the data set.

2.3.2.8 void DataSet::setResidualAt (unsigned int *index,* unsigned int *label,* float *value*)

It sets the residual at the given position and label.

Parameters

index	is the index of the data matrix.
label	is the index of the desired label.
value	is the new value for the residual.

2.3.2.9 void DataSet::sort (unsigned int *feature*)

It sorts the data according to the given feature index by using iterative quicksort algorithm.

Parameters

feature	is the feature to sort.

The documentation for this class was generated from the following files:

- DataSet.h
- DataSet.cpp

2.4 GradientBoosting Class Reference

Public Member Functions

- GradientBoosting ()
- ~GradientBoosting ()
- void train ()
- void test ()

2.4.1 Constructor & Destructor Documentation

2.4.1.1 GradientBoosting::GradientBoosting ( )

It constructs the gradient boosting object.

2.4.1.2 GradientBoosting::~GradientBoosting ( )

It destructs the gradient boosting object.

2.4.2 Member Function Documentation

2.4.2.1 void GradientBoosting::test ( )

It predicts.

2.4.2.2 void GradientBoosting::train ( )

It trains.

The documentation for this class was generated from the following files:

- GradientBoosting.h
- GradientBoosting.cpp

2.5 NominalFeature Class Reference

Public Member Functions

- NominalFeature ()
- NominalFeature (const std::string &featureName)
- ~NominalFeature ()
- void setName (std::string name)
- std::string getName () const
- void add (const std::string &featureValue)
- int getIntValue (std::string value) const
- std::string getStringValue (int index) const
- unsigned int getNumberOfValues () const

2.5.1 Constructor & Destructor Documentation

2.5.1.1 NominalFeature::NominalFeature ( ) [inline]

It sets the object to an empty state.

2.5.1.2 NominalFeature::NominalFeature (const std::string & *featureName*) [inline]

It constructs the nominal feature object.

Parameters

name	is the name of the feature.

2.5.1.3 NominalFeature::~NominalFeature ( ) [inline]

The nominal feature destructor.

2.5.2 Member Function Documentation

2.5.2.1 void NominalFeature::add (const std::string & *featureValue*) [inline]

It adds a new value to the object.

Parameters

featureValue	is the string containing the feature value.

2.5.2.2 int NominalFeature::getIntValue (std::string *value*) const [inline]

It returns the index value of the feature.

Returns

the index value.

2.5.2.3 std::string NominalFeature::getName ( ) const [inline]

It gets the name.

2.5.2.4 unsigned int NominalFeature::getNumberOfValues ( ) const [inline]

It returns the number of values in the feature.

Returns

the number of values.

2.5.2.5 std::string NominalFeature::getStringValue (int *index*) const [inline]

It returns the string value of the feature.

Returns

the value.

2.5.2.6 void NominalFeature::setName (std::string *name*) [inline]

It sets the name.

The documentation for this class was generated from the following file:

- NominalFeature.h

2.6 SAUXMap Struct Reference

Public Attributes

- int **nivell**
- int **repetits**

The documentation for this struct was generated from the following file:

- BinaryTree.h

Index