Package

com.psychofree.mining

com.psychofree.mining
Class Apriori

```
java.lang.Object
   |
   +-com.psychofree.mining.Apriori
```

public class **Apriori**
extends java.lang.Object

This class implements the Apriori algorithm for finding large itemsets. (see "Fast Algorithms for Mining Association Rules" by Rakesh Agrawal and Ramakrishnan Srikant from IBM Almaden Research Center 1994) This file is a part of the ARMiner project. (P)1999-2000 by ARMiner Server Team: Dana Cristofor Laurentiu Cristofor

Constructor Summary

public	Apriori()

Method Summary

java.util.Vector	findLargeItemsets(java.sql.ResultSet resultset, float minSupport) Find the frequent itemsets in a database
int	getPass_num()
void	setPass_num(int pass_num)

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

Apriori

```
public Apriori()
```

Methods

findLargeItemsets

```
public java.util.Vector findLargeItemsets(java.sql.ResultSet resultset,
        float minSupport)
```

Find the frequent itemsets in a database

Parameters:

`resultset` - The object used to read from the database
`minSupport` - The minimum support

(continued on next page)

(continued from last page)

Returns:
The large itemsets found

getPass_num

```
public int getPass_num()
```

setPass_num

```
public void setPass_num(int pass_num)
```

com.psychofree.mining
Class AprioriRules

```
java.lang.Object
   |-com.psychofree.mining.MiningFunction
      |-com.psychofree.mining.AprioriRules
```

public class **AprioriRules**
extends MiningFunction

This class runs the Apriori algorithm for Association Rules using the interface supplyed by com.psychofree.mining.Apriori class

Constructor Summary

public	AprioriRules()

Method Summary

java.util.Vector	findAssociations(java.util.Vector generatedItemsets, float minSupport, float minConfidence) Find association rules in a database, given the set of frequent itemsets.
static void	main(java.lang.String[] args) Used for testing an instance of the class outside the PBMS Engine
java.lang.String	minePatterns(java.lang.String[] args, java.sql.Connection conn) Start the process of rule extraction and store the result in the Pattern Base

Methods inherited from class com.psychofree.mining.MiningFunction

createOutputTable, minePatterns

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

AprioriRules

public **AprioriRules**()

Methods

(continued from last page)

minePatterns

```
public java.lang.String minePatterns(java.lang.String[] args,
         java.sql.Connection conn)
  throws java.sql.SQLException
```

Start the process of rule extraction and store the result in the Pattern Base

Parameters:

`args` - The array containing the parameters for the mining function, which are: mininum value for support and confidence, name of the data source and the pattern type name
`conn` - The existing database connection to be used

Returns:

The temporary table name storing the results

---

findAssociations

```
public java.util.Vector findAssociations(java.util.Vector generatedItemsets,
         float minSupport,
         float minConfidence)
```

Find association rules in a database, given the set of frequent itemsets.

Parameters:

`minSupport` - the minimum support
`minConfidence` - the minimum confidence

Returns:

a Vector containing all association rules found

---

main

```
public static void main(java.lang.String[] args)
```

Used for testing an instance of the class outside the PBMS Engine

com.psychofree.mining
Class AprioriRulesWeka

```
java.lang.Object
   |-com.psychofree.mining.MiningFunction
      |-com.psychofree.mining.AprioriRulesWeka
```

public class **AprioriRulesWeka**
extends MiningFunction

This class runs the Apriori algorithm for Association Rules supplyed by WEKA (weka.associations.Apriori)

Constructor Summary

public	AprioriRulesWeka()

Method Summary

static weka.core.Instances	initMarketBasketDataSet(weka.core.Instances instances, java.lang.String strFalse) Initialize a dataset for market basket analysis, all 'false' values have to be replaced with '?'
static java.lang.String	itemsetToString(weka.associations.ItemSet set, weka.core.Instances instances) Returns a String representation of a given weka.associations.ItemSet
static void	main(java.lang.String[] args) Used for testing an instance of the class outside the PBMS Engine
java.lang.String	minePatterns(java.lang.String[] args, java.sql.Connection conn) Start the process of rule extraction and store the result in the Pattern Base

Methods inherited from class com.psychofree.mining.MiningFunction

createOutputTable, minePatterns

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

AprioriRulesWeka

public **AprioriRulesWeka**()

Methods

(continued from last page)

minePatterns

```
public java.lang.String minePatterns(java.lang.String[] args,
         java.sql.Connection conn)
  throws java.sql.SQLException,
         java.lang.Exception
```

Start the process of rule extraction and store the result in the Pattern Base

Parameters:

`args` - The String array containing the required mining function parameters, which are: mininum value for support and confidence, name of the data source, the number of rule to extract, a flag indicating if we are performing market basket analysis, the data source name and the pattern type name
`conn` - The existing connection to be used

Returns:

The temporary table name storing the results

---

initMarketBasketDataSet

```
public static weka.core.Instances initMarketBasketDataSet(weka.core.Instances
instances,
         java.lang.String strFalse)
```

Initialize a dataset for market basket analysis, all 'false' values have to be replaced with '?' as they are missing values in this case, in order to get only associations between true items

Parameters:

`instances` - the instances to be initialized
`strFalse` - the string (nominal value) that represents the false value

Returns:

The update instances of the dataset

---

itemsetToString

```
public static java.lang.String itemsetToString(weka.associations.ItemSet set,
          weka.core.Instances instances)
```

Returns a String representation of a given weka.associations.ItemSet

Parameters:

`set` - The weka.associations.ItemSet
`instances` - The dataset of the itemset

---

main

```
public static void main(java.lang.String[] args)
```

Used for testing an instance of the class outside the PBMS Engine

com.psychofree.mining
Class AssociationRule

```
java.lang.Object
   |
   +-com.psychofree.mining.AssociationRule
```

All Implemented Interfaces:
java.io.Serializable

---

public class **AssociationRule**
extends java.lang.Object
implements java.io.Serializable

AssociationRule.java An association rule has two parts: the antecedent of the rule and the consequent of the rule, both of which are sets of items. Associated with these are a support and a confidence. The support tells how many rows of a database support this rule, the confidence tells what percentage of the rows that contain the antecedent also contain the consequent.

Field Summary

public static final	ANTECEDENT_SIZE Value: **1**
public static final	CONFIDENCE Value: **4**
public static final	CONSEQUENT_SIZE Value: **2**
public static final	SUPPORT Value: **3**

Constructor Summary

public	AssociationRule(Itemset antecedent, Itemset consequent, float support, float confidence) Creates a new association rule.

Method Summary

int	antecedentSize() Return size of antecedent.
int	compareTo(AssociationRule ar, int criteria) Compare two AssociationRule objects on one of several criteria.
int	consequentSize() Return size of consequent.
boolean	equals(java.lang.Object obj) Compare two AssociationRule objects on one of several criteria.

int	getAntecedentItem(int i) Return i-th item in antecedent.
float	getConfidence() Return confidence of association rule.
int	getConsequentItem(int i) Return i-th item in consequent.
float	getSupport() Return support of association rule.
java.lang.String	toString() Return a java.lang.String representation of the AssociationRule.

Methods inherited from class `java.lang.Object`

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Fields

ANTECEDENT_SIZE

`public static final int` **`ANTECEDENT_SIZE`**

Constant value: **1**

CONSEQUENT_SIZE

`public static final int` **`CONSEQUENT_SIZE`**

Constant value: **2**

SUPPORT

`public static final int` **`SUPPORT`**

Constant value: **3**

CONFIDENCE

`public static final int` **`CONFIDENCE`**

Constant value: **4**

Constructors

(continued from last page)

AssociationRule

```
public AssociationRule(Itemset antecedent,
                       Itemset consequent,
                       float support,
                       float confidence)
```

Creates a new association rule.

Parameters:

`antecedent` - the antecedent of the association rule
`consequent` - the consequent of the association rule
`support` - the support of the association rule
`confidence` - the confidence of the association rule

Throws:

`IllegalArgumentException` - `antecedent` or `consequent` are null or `support` or `confidence` are not between 0 and 1

Methods

antecedentSize

```
public int antecedentSize()
```

Return size of antecedent.

Returns:

size of antecedent

---

consequentSize

```
public int consequentSize()
```

Return size of consequent.

Returns:

size of consequent

---

getSupport

```
public float getSupport()
```

Return support of association rule.

---

getConfidence

```
public float getConfidence()
```

Return confidence of association rule.

---

getAntecedentItem

```
public int getAntecedentItem(int i)
```

Return i-th item in antecedent.

Parameters:

(continued from last page)

`i` - the index of the item to get

Returns:
the `i`-th item in antecedent

Throws:
`IndexOutOfBoundsException` - `i` is an invalid index

---

getConsequentItem

```
public int getConsequentItem(int i)
```

Return i-th item in consequent.

Parameters:
`i` - the index of the item to get

Returns:
the `i`-th item in consequent

Throws:
`IndexOutOfBoundsException` - `i` is an invalid index

---

compareTo

```
public int compareTo(AssociationRule ar,
          int criteria)
```

Compare two AssociationRule objects on one of several criteria.

Parameters:
`ar` - the AssociationRule object with which we want to compare this object
`criteria` - the criteria on which we want to compare, can be one of ANTECEDENT_SIZE, CONSEQUENT_SIZE, SUPPORT or CONFIDENCE.

Returns:
a negative value if this object is smaller than `ar`, 0 if they are equal, and a positive value if this object is greater.

Throws:
`IllegalArgumentException` - `ar` is null or criteria is invalid

---

equals

```
public boolean equals(java.lang.Object obj)
```

Compare two AssociationRule objects on one of several criteria.

Returns:
true if the objects are equal in terms of antecedent and consequent items; false otherwise.

---

toString

```
public java.lang.String toString()
```

Return a java.lang.String representation of the AssociationRule.

Returns:
java.lang.String representation of AssociationRule

com.psychofree.mining
Class AssociationRuleWeka

```
java.lang.Object
   |
   +-com.psychofree.mining.AssociationRuleWeka
```

All Implemented Interfaces:
java.io.Serializable

---

public class **AssociationRuleWeka**
extends java.lang.Object
implements java.io.Serializable

Weka version of association rule

Field Summary

public static final	ANTECEDENT_SIZE Value: **1**
public static final	CONFIDENCE Value: **4**
public static final	CONSEQUENT_SIZE Value: **2**
public static final	SUPPORT Value: **3**

Constructor Summary

public	AssociationRuleWeka(weka.associations.ItemSet antecedent, weka.associations.ItemSet consequent, double support, double confidence) Creates a new association rule.

Method Summary

int	antecedentSize() Returns size of antecedent.
java.lang.String	antecedentToString(weka.core.Instances instances) Return a java.lang.String representation of the AssociationRule
int	compareTo(AssociationRuleWeka ar, int criteria) Compare two AssociationRule objects on one of several criteria.
int	consequentSize() Returns size of consequent.
java.lang.String	consequentToString(weka.core.Instances instances)

boolean	equals(java.lang.Object obj)
weka.associations.ItemSet	getAntecedent() Return the antecedent.
double	getConfidence() Return confidence of association rule.
weka.associations.ItemSet	getConsequent() Return the consequent.
double	getSupport() Return support of association rule.

Methods inherited from class `java.lang.Object`

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Fields

ANTECEDENT_SIZE

```
public static final int ANTECEDENT_SIZE
```

Constant value: **1**

CONSEQUENT_SIZE

```
public static final int CONSEQUENT_SIZE
```

Constant value: **2**

SUPPORT

```
public static final int SUPPORT
```

Constant value: **3**

CONFIDENCE

```
public static final int CONFIDENCE
```

Constant value: **4**

Constructors

AssociationRuleWeka

```
public AssociationRuleWeka(weka.associations.ItemSet antecedent,
                           weka.associations.ItemSet consequent,
                           double support,
                           double confidence)
```

(continued from last page)

Creates a new association rule.

Parameters:

`antecedent` - the antecedent of the association rule
`consequent` - the consequent of the association rule
`confidence` - the confidence of the association rule

Methods

antecedentSize

```
public int antecedentSize()
```

Returns size of antecedent.

Returns:

size of antecedent

---

consequentSize

```
public int consequentSize()
```

Returns size of consequent.

Returns:

size of consequent

---

getSupport

```
public double getSupport()
```

Return support of association rule.

---

getConfidence

```
public double getConfidence()
```

Return confidence of association rule.

---

getAntecedent

```
public weka.associations.ItemSet getAntecedent()
```

Return the antecedent.

---

getConsequent

```
public weka.associations.ItemSet getConsequent()
```

Return the consequent.

---

compareTo

```
public int compareTo(AssociationRuleWeka ar,
                     int criteria)
  throws java.lang.IllegalArgumentException
```

Compare two AssociationRule objects on one of several criteria.

(continued from last page)

Parameters:

`ar` - the AssociationRule object with which we want to compare this object

`criteria` - the criteria on which we want to compare, can be one of ANTECEDENT_SIZE, CONSEQUENT_SIZE, SUPPORT or CONFIDENCE.

Returns:

a negative value if this object is smaller than `ar`, 0 if they are equal, and a positive value if this object is greater.

---

equals

```
public boolean equals(java.lang.Object obj)
```

Returns:

true if the objects are equal in terms of antecedent and consequent items, false otherwise.

---

antecedentToString

```
public java.lang.String antecedentToString(weka.core.Instances instances)
```

Return a java.lang.String representation of the AssociationRule

Parameters:

`instances` - dataset containing header informations for the antecedent

Returns:

java.lang.String representation of AssociationRule

---

consequentToString

```
public java.lang.String consequentToString(weka.core.Instances instances)
```

com.psychofree.mining
Class BasicKMeans

```
java.lang.Object
   |
   +-com.psychofree.mining.BasicKMeans
```

All Implemented Interfaces:
KMeans

---

public class **BasicKMeans**
extends java.lang.Object
implements KMeans

Basic implementation of K-means clustering. Since it's a Runnable, it's designed to be executed by a dedicated thread, but that thread does not create any other threads to divide up the work.

Constructor Summary

public	`BasicKMeans(double[][] coordinates, int k, int maxIterations, long randomSeed)` Constructor

Method Summary

void	`addKMeansListener(KMeansListener l)` Adds a KMeansListener to be notified of significant happenings.
static double[][]	`generateCoordinates(int coordCount, int dimensions, int clusterCount, long randomSeed)` Generates sample coordinates datasets.
Cluster[]	`getClusters()` Get the clusters computed by the algorithm.
double[][]	`getDistances()`
void	`removeKMeansListener(KMeansListener l)` Removes a KMeansListener from the listener list.
void	`run()` Run the clustering algorithm.

Methods inherited from class `java.lang.Object`

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Methods inherited from interface `com.psychofree.mining.KMeans`

`addKMeansListener, getClusters, removeKMeansListener`

Methods inherited from interface `java.lang.Runnable`

`run`

Constructors

BasicKMeans

```
public BasicKMeans(double[][] coordinates,
                   int k,
                   int maxIterations,
                   long randomSeed)
```

Constructor

Parameters:

`coordinates` - two-dimensional array containing the coordinates to be clustered.
`k` - the number of desired clusters.
`maxIterations` - the maximum number of clustering iterations.
`randomSeed` - seed used with the random number generator.

Methods

addKMeansListener

```
public void addKMeansListener(KMeansListener l)
```

Adds a KMeansListener to be notified of significant happenings.

Parameters:

`l` - the listener to be added.

removeKMeansListener

```
public void removeKMeansListener(KMeansListener l)
```

Removes a KMeansListener from the listener list.

Parameters:

`l` - the listener to be removed.

getClusters

```
public Cluster[] getClusters()
```

Get the clusters computed by the algorithm. This method should not be called until clustering has completed successfully.

Returns:

an array of Cluster objects.

getDistances

```
public double[][] getDistances()
```

(continued from last page)

generateCoordinates

```
public static double[][] generateCoordinates(int coordCount,
         int dimensions,
         int clusterCount,
         long randomSeed)
  throws InsufficientMemoryException
```

Generates sample coordinates datasets.

Parameters:

`coordCount` - the number of coordinates.
`dimensions` - the length of the coordinates.
`clusterCount` - the number of clusters in the distribution.
`randomSeed` - the seed used by the random number generator.

Returns:

sample coordinates generated

---

run

```
public void run()
```

Run the clustering algorithm.

com.psychofree.mining
Class Cluster

```
java.lang.Object
   |
   +-com.psychofree.mining.Cluster
```

public class **Cluster**
extends java.lang.Object

Class to represent a cluster of coordinates.

Constructor Summary

public	`Cluster(int[] memberIndexes, double[] center)` Constructor.

Method Summary

`double[]`	`getCenter()` Get the cluster center.
`int[]`	`getMemberIndexes()` Get the member indices.
`java.lang.String`	`toString()` Returns a string representation of the cluster cointaining the centroid and the cluster dimension (the number of instance data contained)

Methods inherited from class `java.lang.Object`

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

Cluster

```
public Cluster(int[] memberIndexes,
               double[] center)
```

Constructor.

Parameters:

`memberIndexes` - indices of the member coordinates.
`center` - the cluster center.

Methods

getMemberIndexes

```
public int[] getMemberIndexes()
```

Get the member indices.

(continued from last page)

Returns:
an array containing the indices of the member coordinates.

getCenter

`public double[] getCenter()`

Get the cluster center.

Returns:
a reference to the cluster center array.

toString

`public java.lang.String toString()`

Returns a string representation of the cluster cointaining the centroid and the cluster dimension (the number of instance data contained)

Returns:
a String representing the cluster.

com.psychofree.mining
Class ExampleSimpleKMeans

```
java.lang.Object
   |-com.psychofree.mining.MiningFunction
      |-com.psychofree.mining.ExampleSimpleKMeans
```

public class **ExampleSimpleKMeans**
extends MiningFunction

This class runs the K-means algorithm for Clustering supplyed by WEKA (weka.clusterers.SimpleKMeans) using a dataset containing numeric and nominal attributes

Constructor Summary

public	ExampleSimpleKMeans()

Method Summary

static void	main(java.lang.String[] args) Used for testing an instance of the class outside the PBMS Engine
java.lang.String	minePatterns(java.lang.String[] args, java.sql.Connection conn) Start the process of rule extraction and store the result in the Pattern Base

Methods inherited from class com.psychofree.mining.MiningFunction

createOutputTable, minePatterns

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

ExampleSimpleKMeans

```
public ExampleSimpleKMeans()
```

Methods

minePatterns

```
public java.lang.String minePatterns(java.lang.String[] args,
        java.sql.Connection conn)
  throws java.sql.SQLException,
        java.lang.Exception
```

Start the process of rule extraction and store the result in the Pattern Base

(continued from last page)

Parameters:

`args` - The String array containing the mining function parameters
`conn` - The connection

main

```
public static void main(java.lang.String[] args)
```

Used for testing an instance of the class outside the PBMS Engine

com.psychofree.mining
Class HashTree

```
java.lang.Object
   |
   +-com.psychofree.mining.HashTree
```

public class **HashTree**
extends java.lang.Object

A HashTree is a special data structure that is used to index a Vector of Itemset objects for more efficient processing.

Constructor Summary

public	`HashTree(int listSize, int hashSize, java.util.Vector itemsets)` Create a new HashTree.
public	`HashTree(java.util.Vector itemsets)` Create a new HashTree.

Method Summary

void	`add(int index)` This method indexes in the HashTree the Itemset at index `index` from Vector `itemsets` which was passed to the constructor of this HashTree.
void	`checkLargeness(Itemset itemset)` Verifies if any of the indexed Itemsets is not large by checking whether they're included in the frequent itemset `itemset`.
long	`countFrequentSubsets(Itemset itemset, long minWeight)` Count how many frequent Itemsets (frequent = having weight greater than a specified minimum weight) are included in `itemset`
long	`countSubsets(Itemset itemset)` Count how many Itemsets are included in `itemset`
void	`prepareForDescent()` This method should be called before calling update() to gather all leaves of the HashTree for more efficient processing.
void	`update(Itemset row)` Update the weights of all indexed Itemsets that are included in `row`
void	`update(Itemset row, long[][] counts)` Update the weights of all indexed Itemsets that are included in `row` and also update the matrix `counts`

Methods inherited from class `java.lang.Object`

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

(continued from last page)

HashTree

```
public HashTree(int listSize,
                int hashSize,
                java.util.Vector itemsets)
```

Create a new HashTree. The `listSize` parameter determines after how many inserts in a ListNode we have to change it to a HashNode (i.e. perform a split). The `hashSize` parameter can be specified to improve the efficiency of the structure.

Parameters:

`listSize` - the size of the internal lists in the list nodes
`hashSize` - the size of the internal hashtables in the hash nodes
`itemsets` - the Vector of Itemsets that we should index

Throws:

`IllegalArgumentException` - `itemsets` is null or `listSize <= 0` or `hashSize <= 0`

HashTree

```
public HashTree(java.util.Vector itemsets)
```

Create a new HashTree. This initializes the HashTree with default parameters.

Parameters:

`itemsets` - the Vector of Itemsets that we should index

Throws:

`IllegalArgumentException` - `itemsets` is null

Methods

prepareForDescent

```
public void prepareForDescent()
```

This method should be called before calling update() to gather all leaves of the HashTree for more efficient processing.

---

add

```
public void add(int index)
```

This method indexes in the HashTree the Itemset at index `index` from Vector `itemsets` which was passed to the constructor of this HashTree.

Parameters:

`index` - the index of the Itemset that we need to index in this HashTree.

---

update

```
public void update(Itemset row)
```

Update the weights of all indexed Itemsets that are included in `row`

Parameters:

`row` - the Itemset (normally a database row) against which we test for inclusion

---

(continued from last page)

update

```
public void update(Itemset row,
         long[][] counts)
```

Update the weights of all indexed Itemsets that are included in `row` and also update the matrix `counts`

Parameters:

`row` - the Itemset (normally a database row) against which we test for inclusion

`counts` - a matrix used by some algorithms to speed up computations; its rows correspond to Itemsets and its columns correspond to items; each value in the matrix tells for how many times had an item appeared together with an itemset in the rows of the database.

---

countFrequentSubsets

```
public long countFrequentSubsets(Itemset itemset,
         long minWeight)
```

Count how many frequent Itemsets (frequent = having weight greater than a specified minimum weight) are included in `itemset`

Parameters:

`itemset` - the Itemset for which we count the subsets

`minWeight` - the minimum weight

---

countSubsets

```
public long countSubsets(Itemset itemset)
```

Count how many Itemsets are included in `itemset`

Parameters:

`itemset` - the Itemset for which we count the subsets

---

checkLargeness

```
public void checkLargeness(Itemset itemset)
```

Verifies if any of the indexed Itemsets is not large by checking whether they're included in the frequent itemset `itemset`. If an Itemset is not large then it will be marked.

Parameters:

`itemset` - the Itemset we check

com.psychofree.mining
Class InsufficientMemoryException

```
java.lang.Object
   |-java.lang.Throwable
      |-java.lang.Exception
         |-com.psychofree.mining.InsufficientMemoryException
```

All Implemented Interfaces:
java.io.Serializable

---

public class **InsufficientMemoryException**
extends java.lang.Exception

Exception thrown when insufficient memory is available to perform an operation. Designed to be throw before doing something that would cause a `java.lang.OutOfMemoryError`.

Constructor Summary

public	`InsufficientMemoryException(java.lang.String message)` Constructor.
public	`InsufficientMemoryException()` Default constructor.

Methods inherited from class `java.lang.Throwable`

`fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString`

Methods inherited from class `java.lang.Object`

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

InsufficientMemoryException

`public InsufficientMemoryException(java.lang.String message)`

Constructor.

Parameters:
`message` - an explanatory message.

InsufficientMemoryException

`public InsufficientMemoryException()`

Default constructor.

com.psychofree.mining
Class Itemset

```
java.lang.Object
   |
   +-com.psychofree.mining.Itemset
```

All Implemented Interfaces:
java.io.Serializable

---

public class **Itemset**
extends java.lang.Object
implements java.io.Serializable

An itemset is an ordered list of integers that identify items coupled with a float value representing the support of the itemset as a percentage.

Constructor Summary

public	`Itemset()` Creates a new empty itemset.
public	`Itemset(int c)` Create a new empty itemset of specified capacity.
public	`Itemset(Itemset itemset)` Create a new itemset by copying a given one.

Method Summary

Itemset	`add(Itemset itemset)` Return a new Itemset that contains all those items that appear in this Itemset and in `itemset`.
boolean	`addItem(int item)` Add a new item to the itemset.
boolean	`canCombineWith(Itemset itemset)` Check whether two itemsets can be combined.
Itemset	`combineWith(Itemset itemset)` Combine two itemsets into a new one that will contain all the items in the first itemset plus the last item in the second itemset.
boolean	`doesIntersect(Itemset itemset)` Return true if this itemset has items in common with `itemset`.
int	`getFirstItem()` Return first item in set.
int	`getItem(int i)` Return i-th item in set.
int	`getNextItem()` Return next item in set.

float	getSupport() Return support of itemset.
long	getWeight() Return weight of itemset.
boolean	hasMoreItems() Return true if there are more items in the itemset.
void	incrementWeight() Increment the weight of the itemset.
boolean	isEqualTo(Itemset itemset) Checks equality with a given itemset.
boolean	isIncludedIn(Itemset itemset) Checks inclusion in a given itemset.
boolean	isMarked() Return itemset mark.
boolean	mark() Mark the itemset.
static void	pruneDuplicates(java.util.Vector v) Remove all duplicate itemsets from the vector v
static void	pruneNonMaximal(java.util.Vector v) Remove all non-maximal itemsets from the vector v
boolean	removeItem(int item) Removes a given item from the itemset.
boolean	removeLastItem() Removes last item (which has the greatest value) from the itemset.
void	setSupport(float newSupport) Set the support of the itemset.
void	setWeight(long newWeight) Set the weight of the itemset.
int	size() Return size of itemset.
Itemset	subtract(Itemset itemset) Return a new Itemset that contains only those items that do not appear in itemset.
java.lang.String	toString() Return a String representation of the Itemset.
boolean	unmark() Unmark the itemset.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

(continued from last page)

Constructors

Itemset

`public Itemset()`

Creates a new empty itemset.

Itemset

`public Itemset(int c)`

Create a new empty itemset of specified capacity.

Parameters:
`c` - the capacity of the itemset

Throws:
`IllegalArgumentException` - `c` is negative or zero

Itemset

`public Itemset(Itemset itemset)`

Create a new itemset by copying a given one.

Parameters:
`itemset` - the itemset to be copied

Throws:
`IllegalArgumentException` - `itemset` is null

Methods

getSupport

`public float getSupport()`

Return support of itemset.

---

getWeight

`public long getWeight()`

Return weight of itemset.

---

getItem

`public int getItem(int i)`

Return i-th item in set.

Parameters:
`i` - the index of the item to get

Returns:
the `i`-th item

(continued from last page)

Throws:
`IndexOutOfBoundsException` - `i` is an invalid index

getFirstItem

`public int getFirstItem()`

Return first item in set.

Returns:
first item

Throws:
`IndexOutOfBoundsException` - there is no first item

getNextItem

`public int getNextItem()`

Return next item in set.

Returns:
next item

Throws:
`IndexOutOfBoundsException` - there is no next item

hasMoreItems

`public boolean hasMoreItems()`

Return true if there are more items in the itemset. You can call this method to find out whether you can call getNext without raising an exception.

Returns:
true if there are more items, false if not

size

`public int size()`

Return size of itemset.

Returns:
size of itemset

doesIntersect

`public boolean doesIntersect(Itemset itemset)`

Return true if this itemset has items in common with `itemset`.

Parameters:
`itemset` - the itemset with which we compare

Returns:
true if `itemset` contains items of this itemset, false otherwise.

Throws:

(continued from last page)

`IllegalArgumentException` - `itemset` is null

---

subtract

`public Itemset subtract(Itemset itemset)`

Return a new Itemset that contains only those items that do not appear in `itemset`.

Parameters:
`itemset` - the itemset whose items we want to subtract

Returns:
an Itemset containing only those items of this Itemset that do not appear in `itemset`.

Throws:
`IllegalArgumentException` - `itemset` is null

---

add

`public Itemset add(Itemset itemset)`

Return a new Itemset that contains all those items that appear in this Itemset and in `itemset`.

Parameters:
`itemset` - the itemset whose items we want to add

Returns:
an Itemset containing all those items that appear in this Itemset and in `itemset`.

Throws:
`IllegalArgumentException` - `itemset` is null

---

addItem

`public boolean addItem(int item)`

Add a new item to the itemset.

Parameters:
`item` - the item to be added

Returns:
true if item was added, false if it wasn't added (was already there!)

Throws:
`IllegalArgumentException` - `item` is <= 0

---

removeItem

`public boolean removeItem(int item)`

Removes a given item from the itemset.

Parameters:
`item` - the item to remove

Returns:
true if item was removed, false if it wasn't removed (was not found in itemset!)

(continued from last page)

Throws:
`IllegalArgumentException` - `item` is <= 0

---

removeLastItem

`public boolean removeLastItem()`

Removes last item (which has the greatest value) from the itemset.

Returns:
true if item was removed, false if it wasn't removed (the itemset was empty)

---

setSupport

`public void setSupport(float newSupport)`

Set the support of the itemset.

Parameters:
`newSupport` - the support of the itemset

Throws:
`IllegalArgumentException` - `newSupport` is < 0 or > 100

---

setWeight

`public void setWeight(long newWeight)`

Set the weight of the itemset.

Parameters:
`newWeight` - the weight of the itemset

Throws:
`IllegalArgumentException` - `newWeight` is < 0

---

incrementWeight

`public void incrementWeight()`

Increment the weight of the itemset.

---

isEqualTo

`public boolean isEqualTo(Itemset itemset)`

Checks equality with a given itemset.

Parameters:
`itemset` - the itemset against which we test for equality

Throws:
`IllegalArgumentException` - `itemset` is null

---

isIncludedIn

`public boolean isIncludedIn(Itemset itemset)`

(continued from last page)

Checks inclusion in a given itemset.

Parameters:
`itemset` - the itemset against which we test for inclusion

Throws:
`IllegalArgumentException` - `itemset` is null

mark

`public boolean mark()`

Mark the itemset.

Returns:
true if itemset was already marked, false otherwise

unmark

`public boolean unmark()`

Unmark the itemset.

Returns:
true if itemset was marked, false otherwise

isMarked

`public boolean isMarked()`

Return itemset mark.

Returns:
true if itemset is marked, false otherwise

toString

`public java.lang.String toString()`

Return a String representation of the Itemset.

Returns:
String representation of Itemset

canCombineWith

`public boolean canCombineWith(Itemset itemset)`

Check whether two itemsets can be combined. Two itemsets can be combined if they differ only in the last item.

Parameters:
`itemset` - itemset with which to combine

Returns:
true if the itemsets can be combined, false otherwise

Throws:
`IllegalArgumentException` - `itemset` is null

combineWith

`public Itemset combineWith(Itemset itemset)`

Combine two itemsets into a new one that will contain all the items in the first itemset plus the last item in the second itemset.

Parameters:
`itemset` - itemset with which to combine

Returns:
an itemset that combines the two itemsets as described above

Throws:
`IllegalArgumentException` - `itemset` is null

pruneNonMaximal

`public static void pruneNonMaximal(java.util.Vector v)`

Remove all non-maximal itemsets from the vector v

Parameters:
`v` - the collection of itemsets

pruneDuplicates

`public static void pruneDuplicates(java.util.Vector v)`

Remove all duplicate itemsets from the vector v

Parameters:
`v` - the collection of itemsets

com.psychofree.mining
Interface KMeans

All Known Implementing Classes:
BasicKMeans

public interface **KMeans**
extends java.lang.Runnable

Simple K-Means clustering interface.

Method Summary

void	addKMeansListener(KMeansListener l) Adds a KMeansListener to be notified of significant happenings.
Cluster[]	getClusters() Get the clusters computed by the algorithm.
void	removeKMeansListener(KMeansListener l) Removes a KMeansListener from the listener list.

Methods inherited from interface `java.lang.Runnable`

`run`

Methods

addKMeansListener

`public void` **`addKMeansListener`**`(KMeansListener l)`

Adds a KMeansListener to be notified of significant happenings.

Parameters:
`l` - the listener to be added.

removeKMeansListener

`public void` **`removeKMeansListener`**`(KMeansListener l)`

Removes a KMeansListener from the listener list.

Parameters:
`l` - the listener to be removed.

getClusters

`public Cluster[]` **`getClusters`**`()`

Get the clusters computed by the algorithm. This method should not be called until clustering has completed successfully.

(continued from last page)

Returns:

an array of Cluster objects.

com.psychofree.mining
Interface KMeansListener

public interface **KMeansListener**
extends

Defines object which register with implementation of `KMeans` to be notified of significant events during clustering.

Method Summary

void	kmeansComplete(Cluster[] clusters, long executionTime) KMeans is complete.
void	kmeansError(java.lang.Throwable t) An error occurred during KMeans clustering.
void	kmeansMessage(java.lang.String message) A message has been received.

Methods

kmeansMessage

```
public void kmeansMessage(java.lang.String message)
```

A message has been received.

Parameters:
`message`

kmeansComplete

```
public void kmeansComplete(Cluster[] clusters,
          long executionTime)
```

KMeans is complete.

Parameters:
`clusters` - the output of clustering.
`executionTime` - the time in milliseconds taken to cluster.

kmeansError

```
public void kmeansError(java.lang.Throwable t)
```

An error occurred during KMeans clustering.

Parameters:
`t`

com.psychofree.mining
Class KMeansPoints

```
java.lang.Object
   |-com.psychofree.mining.MiningFunction
      |-com.psychofree.mining.KMeansPoints
```

public class **KMeansPoints**
extends MiningFunction

This class runs the K-means algorithm for Clustering provided by WEKA (weka.clusterers.SimpleKMeans) using a dataset containing only numeric attributes

Constructor Summary

public	KMeansPoints()

Method Summary

static void	main(java.lang.String[] args) Used for testing an instance of the class outside the PBMS Engine
java.lang.String	minePatterns(java.lang.String[] args, java.sql.Connection conn) Start the process of rule extraction and store the result in the Pattern Base.

Methods inherited from class com.psychofree.mining.MiningFunction

createOutputTable, minePatterns

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

KMeansPoints

```
public KMeansPoints()
```

Methods

minePatterns

```
public java.lang.String minePatterns(java.lang.String[] args,
        java.sql.Connection conn)
  throws java.sql.SQLException,
        java.lang.Exception
```

Start the process of rule extraction and store the result in the Pattern Base.

(continued from last page)

Parameters:

`args` - The String array containing the mining function parameters
`conn` - The database connection

main

```
public static void main(java.lang.String[] args)
```

Used for testing an instance of the class outside the PBMS Engine

com.psychofree.mining
Class MiningFunction

```
java.lang.Object
   |
   +-com.psychofree.mining.MiningFunction
```

Direct Known Subclasses:
AprioriRulesWeka, AprioriRules, ExampleSimpleKMeans, KMeansPoints, UsoBasicKMeans

---

public abstract class **MiningFunction**
extends java.lang.Object

This is the superclass of all mining functions. User who wsh to create a new java mining function should extend this class. As all PSYCHOfree MUST return a table name containing mined patterns with (at least) an initialized structure method createOutputTable is supplied; therefore, user only need to create a Pattern array containg all mined patterns and everything else is left to this method

---

Constructor Summary

public	MiningFunction()

Method Summary

static java.lang.String	createOutputTable(java.sql.Connection conn, Pattern[] patterns) Creates the temporary output table which stores the extracted patterns
abstract java.lang.String	minePatterns(java.lang.String[] args, java.sql.Connection conn) Method wich performs the patterns extraction

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

---

Constructors

MiningFunction

```
public MiningFunction()
```

Methods

minePatterns

```
public abstract java.lang.String minePatterns(java.lang.String[] args,
                                 java.sql.Connection conn)
  throws java.sql.SQLException,
         java.lang.Exception
```

Method wich performs the patterns extraction

createOutputTable

```
public static java.lang.String createOutputTable(java.sql.Connection conn,
         Pattern[] patterns)
  throws java.sql.SQLException
```

Creates the temporary output table which stores the extracted patterns

Parameters:

`conn` - The existing database connection to be used
`patterns` - The array containing the extracted patterns

com.psychofree.mining
Class SET

```
java.lang.Object
   |
   +-com.psychofree.mining.SET
```

public class **SET**
extends java.lang.Object

Implements a Set Enumeration Tree, which is a prefix tree used for storing and retrieving itemset information.

Constructor Summary

public	SET() Create a new empty SET.

Method Summary

java.util.Vector	getItemsets() Return the itemsets of the SET.
java.util.Vector	getLargeItemsets() Return the maximal itemsets of the SET.
float	getSupport(Itemset itemset) Return the support for a given itemset.
void	insert(Itemset itemset) Insert a new itemset in the SET.

Methods inherited from class `java.lang.Object`

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

SET

`public SET()`

Create a new empty SET.

Methods

insert

`public void insert(Itemset itemset)`

Insert a new itemset in the SET.

Parameters:

(continued from last page)

`itemset` - the itemset to be inserted

Throws:

`IllegalArgumentException` - `itemset` is null or is empty

---

getSupport

```
public float getSupport(Itemset itemset)
  throws java.util.NoSuchElementException
```

Return the support for a given itemset.

Parameters:

`itemset` - the itemset for which we want to obtain the support

Returns:

support

Throws:

`IllegalArgumentException` - `itemset` is null or is empty
`NoSuchElementException` - `itemset` not found in SET

---

getLargeItemsets

```
public java.util.Vector getLargeItemsets()
```

Return the maximal itemsets of the SET.

Returns:

a vector containing the maximal itemsets from the SET

---

getItemsets

```
public java.util.Vector getItemsets()
```

Return the itemsets of the SET.

Returns:

a vector containing the itemsets from the SET

com.psychofree.mining
Class UsoBasicKMeans

```
java.lang.Object
   |-com.psychofree.mining.MiningFunction
      |-com.psychofree.mining.UsoBasicKMeans
```

public class **UsoBasicKMeans**
extends MiningFunction

This class runs the K-means algorithm for Clustering provided by BasiKMeans class

Constructor Summary

public	UsoBasicKMeans()

Method Summary

static void	main(java.lang.String[] args) Used for testing the mining function outside the PMBS Engine
java.lang.String	minePatterns(java.lang.String[] args, java.sql.Connection conn) Start the process of rule extraction and store the result in the Pattern Base.

Methods inherited from class com.psychofree.mining.MiningFunction

createOutputTable, minePatterns

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

UsoBasicKMeans

```
public UsoBasicKMeans()
```

Methods

minePatterns

```
public java.lang.String minePatterns(java.lang.String[] args,
        java.sql.Connection conn)
  throws java.sql.SQLException,
        java.lang.Exception
```

Start the process of rule extraction and store the result in the Pattern Base.

(continued from last page)

Parameters:

`args` - The String array containing the mining function parameters
`conn` - The database connection

main

```
public static void main(java.lang.String[] args)
```

Used for testing the mining function outside the PMBS Engine