Deliverable D4.1

Report on Algorithm Evaluation and Selection

Grant Agreement	Health-F5-2008-200787
Acronym	OpenTox
Name	An Open Source Predictive Toxicology Framework
Coordinator	Douglas Connect

Contract No.	Health-F5-2008-200787	
Document Type:	Deliverable Report	
WP/Task:	WP4/4.1	
Name	Report on Algorithm Evaluation and Selection	
Document ID:	OpenTox Deliverable Report WP4	
Date:	Feb 28, 2009	
Status:	Final Version	
Organisation:	Technische Universität München (TUM)	
Contributors	Stefan Kramer	TUM
	Tobias Girschick	TUM
	Fabian Buchwald	TUM
	Haralambos Sarimveis	NTUA
	Andreas Maunz	IST
	Andreas Karwath	ALU-FR
	Martin Gütlein	ALU-FR
	Alexey Zakharov	IBMC
	Dmitry Filimonv	IBMC
	Vladimir Poroikov	IBMC
	Om Prakash	SIT-JNU
	Gaurav Singhai	SIT-JNU
	Indira Ghosh	SIT-JNU
	Sunil Chawla	SL
	Steve Bowlus	SL
	Nina Jeliazkova	IDEA
	Barry Hardy	DC
	Roman Affentranger	DC

Distribution:	Public

Purpose of Document:	To document results for this deliverable

Document History:	1 – Template filled with content from existing internal draft report version 2.2 on Feb 5, 2009
	2 – Addition of Summary, conclusion and section 4 by Fabian Buchwald and Tobias Girschick
	3 – Final Review by Stefan Kramer after February virtual meeting on Feb 12, 2009
	4 – Final layout refinement by Tobias Girschick on Feb 13, 2009
	5 – Final Version prepared by DC Feb 28, 2009
	6 – Revised Final Version prepared by DC Feb 24, 2010

Table of Contents

Summary

This report on algorithm evaluation and selection gives an overview of the progress that has been made in OpenTox Work Package 4. As discussed at the kick-off meeting in Basel in September 2008, the first tasks were to document, evaluate and discuss available and possibly interesting or useful algorithms. To make this selection more objective, one had to agree on a set of selection criteria for the OpenTox framework prototype. The report gives a detailed description of the results and a comprehensive documentation of the algorithms and implementations relevant for OpenTox. Obviously there is some focus on algorithms provided by the project participants, as those algorithms serve as a starting point in the development of the framework, according to the project proposal. Starting with a short introduction, the text gives an overview of the algorithm selection criteria that we have chosen, followed by detailed description of the algorithms in a uniform tabular manner. This material is then evaluated, and a selection for the prototype is made.

1. Introduction

Ongoing scientific efforts in various complementary fields have led to a significant number of algorithms that are available and potentially useful for (Q)SAR and related tasks. To meet the project specific user requirements and long term goals of OpenTox, it is crucial to establish and subsequently maintain a set of algorithm selection criteria. The initial criteria were proposed by TUM and discussed by the project partners on the OpenTox online forum and at the December 2008 and February 2009 virtual meetings.

To make a reasonable comparison of the available algorithms possible, they were grouped into three categories*: descriptor calculation algorithms, classification and regression algorithms and feature selection algorithms*. For each algorithm a short text description and a uniform (for each of the three categories) table was generated to facilitate a comparison with respect to the selection criteria. The text description of the algorithm gives a brief overview of the algorithm's background, its capabilities, dependencies and technical features. The uniform tables have three logical parts. The first one enables a black-box point of view of the algorithm and has the same fields for every algorithm category. It contains a field for the name, the input and output (semantically), the input and output format, user-specific parameters and reporting information. The second logical part is variable for the three algorithm categories and describes some intrinsic properties of the algorithms. It comprises fields for the algorithm's background and its performance. The descriptor calculation algorithms have a special field for the type of descriptor that is generated. The classification and regression algorithms have special fields for the applicability domain and the confidence in the prediction, the bias, the type of learning (lazy or eager learning) and the interpretability of the generated model. The feature selection algorithms have special fields for type of feature selection (class-blind or class-sensitive), for the distinction of optimal, greedy or randomized methods and for the distinction of filter and wrapper approaches. The third part of the description table is again identical for the different algorithm categories. It gives information about the algorithm's availability within the OpenTox consortium, the license and dependencies, the convenience of integration, the priority of integration, the author of the algorithm and the author of the description. Additionally there are fields for a contact address (email) and for comments.

In section 2 of this document, the fields of the description tables are explained briefly. In section 3 all considered algorithms are listed with their descriptions in the respective category. In section 4 the algorithms are evaluated and the ones that will be used in the initial OpenTox prototypes are selected.

2. Algorithm selection criteria

In the following sections, the fields of the description table for the algorithms are explained briefly.

2.1 Input, Output, Input format and Output format

Those four fields are used to describe the semantic input and output of the algorithm as well as the file formats for input and output that can be used with the suggested or described implementation of the algorithm.

2.2 User-specified parameters and Reporting information

The user-specified parameters are the parameters that have to be or can be adjusted to configure the algorithm. Standard parameters like input or output file name should not be stated here. The reporting information is the algorithm (implementation) output including available statistics and reports.

2.3 Background

Here the publication date, the popularity in the (Q)SAR and toxicology community, the level of familiarity of (Q)SAR users with the algorithm, the rationale of the approach and further comments on the background of the method/algorithm can be noted.

2.4 Type of descriptor

This field is exclusive for descriptor calculation algorithms. It should be filled with a description or explanation of the type of descriptor(s) that are calculated, e.g. physico-chemical or substructural descriptors. Furthermore, comments on the expressiveness and the suitability for similarity and/or distance calculations can be made.

2.5 Applicability domain/confidence in prediction

This field is exclusive for the classification and regression algorithms. The OECD *guidance document on the validation of (quantitative) structure-activity relationships [(Q)SAR] Models* [OEC07] states in paragraph 93 of chapter 3 ("Guidance on principle of a defined domain of applicability") that a (Q)SAR should be associated with a defined domain of applicability. As the grasp of the concept of applicability domain (AD) is not completely formally defined, we will briefly introduce how AD is used throughout this document. Informally, AD is restricted to what is seen on the input and output side during training. A further definition of AD which is also used by the OECD is the following [NET05]:

> *"The applicability domain of a (Q)SAR model is the response and chemical structure space in which the model makes predictions with a given reliability."*

Furthermore, the OECD advises that the AD principle should be applied in a model-specific manner. Thus, every model should be associated with its own AD derived not only on the chemicals in the training set but also on the descriptors and (statistical) approach used to develop the model. Ideally, the AD should be defined and documented by the model developer. Consequently it only makes sense to apply the concept of AD to our

second domain of algorithms, namely the classification and regression algorithms, which will be used in OpenTox to derive the (Q)SAR models. Apart from the composition of the training set and the initially calculated descriptors, the methods' inherent bias and methodology has an influence on the AD of the resulting model, as they have an effect on the model's response space.

Related to the concept of an applicability domain is the concept of a confidence in predictions inherent in most machine learning algorithm. Clearly, most modern machine learning algorithms do not only provide a categorical class label, but also a probability with which the class is predicted. The confidence in predictions comes in many flavors (e.g., margins, ...), but in most cases it can be transformed back into probability estimates (in the case of margins by methods like Platt scaling). Most considerations concerning *abstaining* from prediction in the machine learning literature are centered on the confidence in predictions. The main difference is that the confidence is only known when the model is already applied, that is, in hindsight, whereas the applicability domain seems to be defined for the input space directly. As both concepts are obviously related, statements about the applicability as well as about the confidence in predictions can be entered in this field of the template.

2.6 Bias, lazy/eager learning and interpretability of models

These three fields are exclusive for the classification and regression algorithms. They contain information if the algorithm has an intrinsic bias, e.g. feature-selection bias or instance-selection bias. Furthermore it is stated if the method is an eager or a lazy learning method. The third field contains information of how easy it is to interpret the model or if the algorithm learns or involves complete black box models.

2.7 Class-blind/class-sensitive feature selection

This field is exclusive for feature selection algorithms. It contains information if the algorithm selects the features class-blind or class-sensitive.

2.8 Type of feature selection and of approach

These are two fields exclusive for feature selection algorithms. The type of feature selection algorithm is either an optimal, a greedy or a randomized algorithm. The type of the approach is either a filter, a wrapper or a hybrid approach.

2.9 Performance

This field gives information on the algorithms performance regarding time and space usage. Exemplary running times and memory consumption can be stated as well as theoretical considerations.

2.10 OpenTox availability, License/Dependencies

On those two fields the availability of the algorithm/implementation to the OpenTox project is to be explained. For example a project partner can be stated here. In the license and dependencies field information about the license the implementation is published under and about other software packages the implementation is dependent on are gathered.

2.11 Convenience of Integration and Priority

The convenience of integration field gives information about how easy it will be to integrate the software into the OpenTox prototype and/or overall framework. Relevant are for example, if the implementation is dependent on a specific operating system or not, or if parts of it have to be adjusted before integration or the

like. The priority (divided into three categories *A*, *B* and *C*) is not to be understood as prescriptive but just as a guidance for the prototype development.

2.12 Author of method, author of description, contacts and comments

The last fields are used to facilitate the communication regarding the algorithms. The first field shall be filled with the name(s) of the author(s) of the algorithm/implementation and the contact email, if available. The second field states the author who filled the description table and the contact within OpenTox gives a contact email address within the OpenTox consortium. The remaining comments field can be used for any further comment on the method including reviews.

3. Algorithm documentation

3.1 Descriptor calculation algorithms

3.1.1 FreeTreeMiner (TUM)

The FreeTreeMiner (FTM) software [RUE04] computes all acyclic substructures (in mathematical terms: *free* or *unrooted trees*) occurring at a given minimum frequency in a set of molecules. The substructures are computed by a depth-first search. Additionally to the minimum frequency support, a maximum frequency constraint can be set. This constraint can either refer to the same database/set or to a second one, meaning that all substructures frequent in the first and infrequent in the second are returned by FTM. The frequent substructures are returned as SMARTS strings together with their occurrences in the given set of structures.

The software is implemented in the programming language C++ and was developed for the Linux and Mac OS X operating systems. The FTM software is dependent on the open source chemistry toolbox OpenBabel (www.openbabel.org). FTM itself provides no graphical user interface (GUI) and is executed via the command line. The input format accepted by FTM is the widely used MDL Molfile (sometimes called SD file or SDF; specification URL: www.symyx.com/downloads/public/ctfile/ctfile.jsp). FTM's output formats are program specific plain text files and/or Weka's [WIT99] ARFF format. For further information, we refer to the original publication [RUE04] and the website wwwkramer.in.tum.de/research/data_mining/pattern_mining/graph_mining/

FTM		
Input		
2D chemical structure information		
Output		
Frequent substructures		
Input format		
SD file (MDL Mol)		

Output format Program specific text files and/or Weka's ARFF format
User-specified parameters Minimum support
Reporting information Frequent free trees (SMARTs) with occurrence maps, border elements
Background (publication date, popularity/level of familiarity, rationale of approach, further comments) Published in 2004. A further development of the MolFea approach for acyclic substructures. Acyclic substructures were chosen, as they still allow advanced computations like the calculation of borders. On typical structure databases, the number of frequent acyclic substructures is not much less than the number of frequent unconstrained (i.e., also including cyclic) substructures.
Type of descriptor (substructural/physico-chemical, expressiveness: paths, trees, subgraphs, wildcards?, suitability for similarity/distance calculations?, ...) Substructural descriptors, acyclic substructures, currently no wildcards used or other more advanced features of the SMARTS language, results can be used in all fingerprint-based similarity and distance measures.
Performance (time/space complexity, running times, memory consumption, ...) Dependent on number and size of instances, minimum support and structural diversity of the data set. The output grows exponentially when decreasing the minimum support threshold. The higher the structural diversity, the smaller the output of the algorithm.
OT availability TUM
Licence /Dependencies OpenBabel (open source)
Convenience of integration C++ => OS dependent compilation (Win vs. Linux); command line tool; compiled for Win and Linux
Priority (A, B, C) A
Author of method / Contact Ulrich Rückert (rueckert@icsi.berkeley.edu), Stefan Kramer (kramer@in.tum.de)
Author of description

	Tobias Girschick
Contact within OT	
	kramer@in.tum.de
Comments (including reviews)	

3.1.2 FMiner (IST)

Fminer is a novel method for efficiently mining relevant tree-shaped subgraph descriptors with minimum frequency and correlation constraints, each representing a set of fragments sharing a common core structure (backbone), thereby reducing feature set size and runtime. The approach is able to optimize structural inter-feature entropy as opposed to occurrences, which is characteristic for open or closed fragment mining. In the experiments, the proposed method reduces feature set sizes by >90% and >30% compared to complete tree mining and open tree mining, respectively. Evaluation using cross validation runs shows that their classification accuracy is similar to the complete set of trees but significantly better than that of open trees. Compared to open or closed fragment mining, a large part of the search space can be pruned due to an improved statistical constraint (dynamic upper bound adjustment), which is also confirmed in the experiments in lower runtimes compared to ordinary (static) upper bound pruning. Further analysis using large-scale datasets yields insight into important properties of the proposed descriptors, such as dataset coverage and class size represented by each descriptor. A final cross validation run confirms that the novel descriptors render large training sets feasible which previously might have been intractable for computational models.

FMiner was developed in C++ for the Linux platform and depends on the OpenBabel (openbabel.org) chemistry toolbox and GNU Scientific Library (GSL). It should however be portable to other platforms. It is a library with a thin command line frontend, requiring SMILES or gSpan-Format based input as well as target class activity input in the same format as Lazar. As output format it supports plain text and YAML (SMARTS patterns) as well as gSpan format.

FMiner	
Input	
Output	
Input format	Plain text in gSpan or SMILES format and custom activity format (tab-separated)
Output format	Plain text in YAML or Lazar compatible format (substructures in SMARTS format), or gSpan format
User-specified parameters	Minimum frequency, minimum correlation

Reporting information
Background (publication date, popularity/level of familiarity, rationale of approach, further comments) Not yet published. Built on top of the feature miner Gaston. It is an extension of Gaston in that it supports class-correlated pattern mining and a novel feature set compression technique, improving expressiveness as well as runtime performance. Acyclic substructures were chosen with the same rationale as FTM.
Type of descriptor (substructural/physico-chemical, expressiveness: paths, trees, subgraphs, wildcards?, suitability for similarity/distance calculations?, ...) Substructural descriptors, acyclic substructures, currently no wildcards used or other more advanced features of the SMARTS language, results can be used in all fingerprint-based similarity and distance measures.
Performance (time/space complexity, running times, memory consumption, ...) Gaston complexity results apply, i.e. linear complexity in the refinement of paths and trees. It uses embedding lists which increases memory consumption but decreases runtime. The approach works fastest for equally distributed target classes, also feature set size can be minimized under this conditions. Aromatic perception is used by default.
OT availability IST
Licence /Dependencies OpenBabel (open source), GSL
Convenience of integration C++ => OS dependent compilation (Win vs. Linux); command line tool
Priority (A, B, C) B
Author of method / Contact Andreas Maunz
Author of description Andreas Maunz
Contact within OT maunza@fdm.uni-freiburg.de
Comments (including reviews)

3.1.3 gSpan' (TUM)

The gSpan' algorithm [JK05] implements two optimizations of the widely known gSpan algorithm [HAN02] for mining molecular databases. Both optimizations apply to the enumeration of subgraph occurrences in a graph database, which is, also according to our profiling, the most expensive operation of gSpan. The first optimization reduces the number of subgraph isomorphisms that need to be accessed for proper support computation in considering the symmetries inherent in many chemical molecules, and the second speeds up subgraph isomorphism tests by making use of the non-uniform frequency distribution of atom and bond types.

The software is implemented in the programming language C and was developed for the Linux operating system. The gSpan' implementation has no dependencies on other software packages. The gSpan algorithm has a specific input format, but we already have conversion scripts for the widely used MDL Molfiles (sometimes called SD file or SDF; specification URL: www.symyx.com/downloads/public/ctfile/ctfile.jsp) available at TUM. There exists no graphical user interface (GUI) and the program is executed via the command line. gSpan' s output consists of program specific plain text files.

For further information, we refer to the original publication [JK05] and the website: wwwkramer.in.tum.de/research/data_mining/pattern_mining/graph_mining

gSpan'
Input 2D chemical structure information
Output Frequent substructures
Input format gSpan' specific; [SDF with existing converters]
Output format DFScode file, relabel.txt file (plain text files)
User-specified parameters - Restriction choices for fragments - Minimum support
Reporting information DFS codes for each frequent linear/acyclic fragment, (number of) instances that posses the fragment
Background (publication date, popularity/level of familiarity, rationale of approach, further comments)

Published in 2005. An optimization of the gSpan algorithm for molecular graphs.
Type of descriptor (substructural/physico-chemical, expressiveness: paths, trees, subgraphs, wildcards?, suitability for similarity/distance calculations?, ...) Substructural descriptors, currently no wildcards used or other more advanced features of the SMARTS language, results can be used in all fingerprint-based similarity and distance measures. The user can restrict the search to acyclic and/or linear fragments and/or fragments with a maximum number of edges (bonds).
Performance (time/space complexity, running times, memory consumption, ...) Dependent on number and size of instances, minimum support and structural diversity of the data set. The output grows exponentially when decreasing the minimum support threshold. The higher the structural diversity, the smaller the output of the algorithm.
OT availability TUM
Licence /Dependencies GPL
Convenience of integration C => OS dependent compilation (Win vs. Linux); command line tool
Priority (A, B, C) B
Author of method / Contact Katharina Jahn, Stefan Kramer (kramer@in.tum.de)
Author of description Tobias Girschick
Contact within OT kramer@in.tum.de
Comments (including reviews)

3.1.4 MakeMNA (IBMC)

MakeMNA is a software product for generating MNA descriptors.

These descriptors are based on the molecular structure representation, which includes the hydrogens according to the valences and partial charges of other atoms and does not specify the types of bonds. MNA descriptors are generated as recursively defined sequence:

- zero-level MNA descriptor for each atom is the mark A of the atom itself;
- any next-level MNA descriptor for the atom is the sub-structure notation A(D1D2..Di…), where Di is the previous-level MNA descriptor for i-th immediate neighbor's of the atom A.

The mark of atom may include not only the atomic type but also any additional information about the atom. In particular, if the atom is not included into the ring, it is marked by "-". The neighbor descriptors D1D2...Di… are arranged in unique manner, e.g., in lexicographic order. Iterative process of MNA descriptors generation can be continued covering first, second, etc. neighborhoods of each atom.

MakeMNA
Input 2D, 3D chemical structure information
Output Fragments of structures
Input format SDfile ISIS V2000 file format
Output format SDfile ISIS V2000 file format
User-specified parameters None
Reporting information Log file
Background (publication date, popularity/level of familiarity, rationale of approach, further comments) [FIL99]
Type of descriptor (substructural/physico-chemical, expressiveness: paths, trees, subgraphs, wildcards?, suitability for similarity/distance calculations?, ...) Substructural
Performance (time/space complexity, running times, memory consumption, ...) Approximately 1000 chemical compounds at 2.5 seconds.
OT availability IBMC

Licence /Dependencies GPL
Convenience of integration Delphi => OS dependent compilation (Windows); command line tool
Priority (A, B, C) A
Author of method / Contact Filimonov Dmitry
Author of description Filimonov Dmitry
Contact within OT dmitry.filimonov@ibmc.msk.ru
Comments (including reviews)

3.1.5 MakeQNA (IBMC)

MakeQNA is a software product for generating QNA descriptors.

Quantitative Neighborhoods of Atoms (QNA) descriptors are based on quantities of ionization potential (IP) and electron affinity (EA) of each atom of the molecule. They are calculated as follows:

- Pi = Bi-½Σk(exp(-½C))ikBk-½,
- Qi = Bi-½Σk(exp(-½C))ikBk-½Ak,
- Ai = ½(IPi + EAi), Bi = IPi - EAi,

Where IPi is the ionization potential (the energy required to remove the outermost electron from a neutral gaseous atom), and EAi is the electron affinity (the energy released when an electron is added to a neutral gaseous atom of that element) of atom i.

MakeQNA
Input 2D, 3D chemical structure information
Output Real values of QNA descriptors
Input format SDfile ISIS V2000 file format
Output format SDfile ISIS V2000 file format
User-specified parameters None
Reporting information
Background (publication date, popularity/level of familiarity, rationale of approach, further comments) [FIL05], [LAG07]
Type of descriptor (substructural/physico-chemical, expressiveness: paths, trees, subgraphs, wildcards?, suitability for similarity/distance calculations?, ...) Numerical reflecting the interatomic interaction for each atom in a molecule.
Performance (time/space complexity, running times, memory consumption, ...) Approximately 1000 chemical compounds at 3.5 seconds.
OT availability

IBMC
Licence /Dependencies
GPL
Convenience of integration
Delphi => OS dependent compilation (Windows); command line tool
Priority (A, B, C)
B
Author of method / Contact
Filimonov Dmitry
Author of description
Filimonov Dmitry
Contact within OT
dmitry.filimonov@ibmc.msk.ru
Comments (including reviews)

3.1.6 JOELIB2

JOELIB2 is a platform independent open source computational chemistry package written in Java. JOELIB2 consists of an algorithm library that was designed for prototyping, data mining and graph mining of chemical compounds. JOELib2 is the Java successor of the OELib library from OpenEye.

The software was developed for the Linux and Windows operating system. The JOELIB2 implementation has no dependencies on other software packages. There exists no graphical user interface (GUI) and the program is executed via the command line or via Java code integration.

For further information, we refer to the JOELIB tutorial [JOETUT] and the website www.ra.cs.uni-tuebingen.de/software/joelib/index.html

JOELIB2
Input
2D, 3D chemical structure information
Output
Real valued physicochemical descriptors, binary fingerprints
Input format
SMILEs, MDL Molfile/SD format, GAUSSIAN, CML, MOPAC
Output format
Plain text files

User-specified parameters Descriptors to calculate
Reporting information Numeric or binary (fingerprints) values
Background (publication date, popularity/level of familiarity, rationale of approach, further comments) Latest release in March 2007.
Type of descriptor (substructural/physico-chemical, expressiveness: paths, trees, subgraphs, wildcards?, suitability for similarity/distance calculations?, ...) Physicochemical, geometrical descriptors, functional groups, atom properties, fingerprints, transformations (see Tutorial pages 24-35 www.ra.cs.uni-tuebingen.de/software/joelib/tutorial/JOELibTutorial.pdf)
Performance (time/space complexity, running times, memory consumption, ...) Dependent on number and size of instances and the number and type of selected descriptors to calculate. Simple atom counts are simpler/faster to calculate than more elaborate topological descriptors.
OT availability TUM
Licence /Dependencies GPL
Convenience of integration Versions available for Windows and Linux
Priority (A, B, C) B
Author of method / Contact J.K.Wegner (me@joergkurtwegner.de)
Author of description Fabian Buchwald, Tobias Girschick
Contact within OT kramer@in.tum.de
Comments (including reviews)

3.1.7 OpenBabel

Open Babel is a chemical toolbox designed to speak the many languages of chemical data. It's an open, collaborative project allowing anyone to search, convert, analyze, or store data from molecular modeling, chemistry, solid-state materials, biochemistry, or related areas.

OpenBabel is an open source computational chemistry package written in C++.

The software is available for the Linux, Windows and MAC operating system. The OpenBabel implementation has no dependencies on other software packages.

For further information, we refer to the OpenBabel website openbabel.org

OpenBabel
Input 2D, 3D chemical structure information
Output Real valued physicochemical descriptors, binary fingerprints
Input format Can read, write and convert over 90 chemical file formats
Output format Can read, write and convert over 90 chemical file formats
User-specified parameters Descriptors to calculate
Reporting information Numeric or binary (fingerprints) values
Background (publication date, popularity/level of familiarity, rationale of approach, further comments) Current release of OpenBabel is 2.2.0. Further functionalities are under development.
Type of descriptor (substructural/physico-chemical, expressiveness: paths, trees, subgraphs, wildcards?, suitability for similarity/distance calculations?, ...)
Performance (time/space complexity, running times, memory consumption, ...) Dependent on number and size of instances and the number and type of selected descriptors to calculate. Simple atom counts are simpler/faster to calculate than more elaborate topological descriptors.
OT availability

	TUM
Licence /Dependencies	
	GPL Depends on several C/C++ libraries
Convenience of integration	
	Versions available for Windows and Linux
Priority (A, B, C)	
	A
Author of method / Contact	
	The original Babel (origin of OpenBabel) was written by Pat Walters and Matt Stahl
Author of description	
	Fabian Buchwald, Tobias Girschick
Contact within OT	
	kramer@in.tum.de
Comments (including reviews)	

3.1.8 MOPAC

MOPAC	
Input	
Output	
Input format	
	Mopac DAT files
Output format	
	Mopac OUT files
User-specified parameters	
	MOPAC options, as per specification
Reporting information	
Background (publication date, popularity/level of familiarity, rationale of approach, further	

comments) MOPAC (Molecular Orbital PACkage) was started in 1981, and has been under continuous development since then. MOPAC 7.1 is a FORTRAN 90 version of MOPAC 7. It supports the methods: MNDO, AM1, and PM3, as well as Sparkle/AM1 for the lanthanides. All published NDDO parameter sets are supported.
Type of descriptor (substructural/physico-chemical, expressiveness: paths, trees, subgraphs, wildcards?, suitability for similarity/distance calculations?, ...) Semiempirical quantum chemistry descriptors based on Dewar and Thiel's NDDO approximation.
Performance (time/space complexity, running times, memory consumption, ...)
OT availability openmopac.net source available at openmopac.net/Downloads/Mopac_7.1source.zip Integrated within AMBIT, Toxtree, Toxmatch (IDEA)
Licence /Dependencies Public domain
Convenience of integration MOPAC 7.1 is a FORTRAN 90 version of MOPAC 7. Both Windows and Linux versions are supported. Can be integrated as an external executable. AMBIT and Toxtree provide Java classes for such integration.
Priority (A, B, C) C
Author of method / Contact James Stewart, 15210 Paddington Circle, Colorado Springs, CO 80921 E-mail : MrMOPAC@OpenMOPAC.net SKYPE: Jimmy.Stewart2 (between 1500 and 2200 GMT)
Author of description Nina Jeliazkova
Contact within OT nina@acad.bg, David Gallagher
Comments (including reviews) Newer versions with extended functionality are available under dual academic/ commercial licenses.

3.1.9 The Chemistry Development Kit

The Chemistry Development Kit (CDK) is a Java library for structural chemo- and bioinformatics. It is now developed by more than 50 developers all over the world and used in more than 10 different academic as well as industrial projects world wide. A number of descriptor implementations are available.

Various descriptors implemented by The Chemistry Development Kit (CDK) library
Input
Output
Input format A Java class, representing chemical structure in CDK library
Output format A Java class, representing descriptor value in CDK library
User-specified parameters Depends on descriptor
Reporting information
Background (publication date, popularity/level of familiarity, rationale of approach, further comments) Started in 2000, large code base, references : [CDK], [STE03], [STE06]
Type of descriptor (substructural/physico-chemical, expressiveness: paths, trees, subgraphs, wildcards?, suitability for similarity/distance calculations?, ...) Substructural, physicochemical, topological, etc: cdk.qsar.BCUTDescriptor cdk.qsar.CPSADescriptor cdk.qsar.WHIMDescriptor cdk.qsar.APolDescriptor cdk.qsar.AromaticAtomsCountDescriptor cdk.qsar.AromaticBondsCountDescriptor cdk.qsar.AtomCountDescriptor cdk.qsar.AtomDegreeDescriptor cdk.qsar.AtomHybridizationDescriptor

cdk.qsar.AtomHybridizationVSEPRDescriptor

cdk.qsar.AtomValenceDescriptor

cdk.qsar.InductiveAtomicHardnessDescriptor

cdk.qsar.InductiveAtomicSoftnessDescriptor

cdk.qsar.BondCountDescriptor

cdk.qsar.BondsToAtomDescriptor

cdk.qsar.BPolDescriptor

cdk.qsar.ConnectivityOrderZeroDescriptor

cdk.qsar.CarbonConnectivityOrderZeroDescriptor

cdk.qsar.ValenceConnectivityOrderZeroDescriptor

cdk.qsar.ValenceCarbonConnectivityOrderZeroDescriptor

cdk.qsar.ConnectivityOrderOneDescriptor

cdk.qsar.CarbonConnectivityOrderOneDescriptor

cdk.qsar.ValenceConnectivityOrderOneDescriptor

cdk.qsar.ValenceCarbonConnectivityOrderOneDescriptor

cdk.qsar.DistanceToAtomDescriptor

cdk.qsar.EccentricConnectivityIndexDescriptor

cdk.qsar.EffectivePolarizabilityDescriptor

cdk.qsar.GravitationalIndexDescriptor

cdk.qsar.HBondDonorCountDescriptor

cdk.qsar.HBondAcceptorCountDescriptor

cdk.qsar.IsProtonInAromaticSystemDescriptor

cdk.qsar.IsProtonInConjugatedPiSystemDescriptor

cdk.qsar.KappaShapeIndicesDescriptor

cdk.qsar.RuleOfFiveDescriptor

Performance (time/space complexity, running times, memory consumption, ...)

OT availability

[CDK]

Licence /Dependencies

LGPL

Convenience of integration Implemented in Java, easy to integrate
Priority (A, B, C) B
Author of method / Contact multiple
Author of description Nina Jeliazkova
Contact within OT nina@acad.bg
Comments (including reviews) A dictionary of the descriptors with references is available at : qsar.sourceforge.net/dicts/qsar-descriptors/index.xhtml

3.1.10 AMBIT

AMBIT is a software package for chemoinformatic data management, implemented by IDEA. The descriptor calculation relies on CDK library, but also implements several descriptors, listed below, which are not available from the library. The descriptor calculation is a separate module and packaged in ambit2-descriptors.jar, which depends only on CDK library, core ambit module (ambit2-core.jar) and ambit SMARTS (ambit2-smarts.jar) implementation.

Several descriptor implemented by ambit package
Input
Output
Input format A Java class, representing chemical structure in CDK library
Output format A Java class, representing descriptor value in CDK library
User-specified parameters Depend on descriptor
Reporting information

Background (publication date, popularity/level of familiarity, rationale of approach, further comments)

Various publications

Type of descriptor (substructural/physico-chemical, expressiveness: paths, trees, subgraphs, wildcards?, suitability for similarity/distance calculations?, ...)

Various

ambit2.descriptors.PKASmartsDescriptor Acid dissociation constant, [LEE08] ambit2.descriptors.SpherosityDescriptor Spherosity descriptor [TOD00] 7.1ambit2.descriptors.CrossSectionalDiameterDescriptor Crossectional diameter of a molecule . Requires 3D coordinates

ambit2.mopac.DescriptorMopacShell A shell to calculate quantum chemical descriptors by MOPAC ambit2.descriptors.FunctionalGroupDescriptor The presence of arbitrary functional groups, defined as SMARTS pattern. Full support for SMARTS language, including recursive SMARTS.

toxtree.descriptors. SubstituentsDescriptor Partial molar refractivity and Sterimol descriptors of substituents, as found in [HAN95]

Similarity/distance calculations:

ambit2.similarity module encapsulates similarity calculations – all distance classes implement the same interface. Supports pairwise similarity/distance, similarity/distance to a set of points and similarity/distance based on nearest neighbors

Tanimoto Distance, Atom Environments Distance, BinaryKernelDistance, Hamming Distance, Levenstein Distance, MCSSDistance, Hellinger distance, Kullback – Leibler distance between probability distributions

Performance (time/space complexity, running times, memory consumption, ...)

OT availability

IDEA, ambit.sourceforge.net

Licence /Dependencies

LGPL

Dependencies : CDK, Jama

MOPAC 7.1 for the quantum chemical descriptors only

Convenience of integration

Implemented in Java, easy to integrate

Priority (A, B, C) C
Author of method / Contact Various authors, implementation by IDEA
Author of description Nina Jeliazkova
Contact within OT nina@acad.bg
Comments (including reviews)

3.2 Classification and regression algorithms

3.2.1 Gaussian Processes for Regression

GPR (Gaussian Processes for Regression) is a way of supervised learning. A Gaussian process is a generalization of the Gaussian probability distribution. Whereas a probability distribution describes random variables which are scalars or vectors (for multivariate distributions), a stochastic process governs the properties of functions [RAS05]. Just as a Gaussian distribution is fully specified by its mean and covariance matrix, a Gaussian process is specified by a mean and a covariance function. Here, the mean is a function of x (which we will often take to be the zero function), and the covariance is a function C(x, x') that expresses the expected covariance between the values of the function y at the points x and x'. The function y(x) in any one data modeling problem is assumed to be a single sample from this Gaussian distribution.

Gaussian processes are already well established models for various spatial and temporal problems – for example, Brownian motion, Langevin processes and Wiener processes are all examples of Gaussian processes. Gaussian processes are implementations are available via various software packages and in most programming languages, e.g. Weka (Java), R, Matlab, python, C, C++.

Gaussian Processes
Input Instances, feature vectors, real-numbered target values
Output Regression model
Input format Dependent on implementation, e.g., Weka's ARFF format
Output format

Dependent on implementation, e.g., Weka's ARFF format
User-specified parameters Kernel Covariance function, e.g. radial basis function ("squared exponential")
Applicability domain
Reporting information Performance measures (Correlation coefficient, mean absolute error, root mean squared error, relative absolute error, root relative squared error)
Background (publication date, popularity/level of familiarity, rationale of approach, further comments)
Bias (instance-selection bias, feature-selection bias, combined instance-selection/feature-selection bias, independence assumptions?, ...) The chosen covariance function, which encodes the assumption about the function we want to learn, is a bias.
Lazy learning/eager learning Eager learning
Interpretability of models (black box model?, ...) Depends on the covariance function (kernel).
Performance (time/space complexity, running times, memory consumption, ...) Gaussian processes typically scale O(n^3); large problems (n >10.000) can be problematic (time and space)
OT availability Available in statistical packages, e.g., in the Weka open source data mining workbench
Licence /Dependencies GPL
Convenience of integration Webservices: very easy / implemented in Java
Priority (A, B, C) C
Author of method / Contact Matheron, G., "Principles of geostatistics", Economic Geology, 58, pp 1246--1266,

1963
Author of description
Tobias Girschick
Contact within OT
kramer@in.tum.de
Comments (including reviews)

3.2.2 MLR

MLR (Multiple Linear Regression) is a simple and popular statistical technique that uses several explanatory (independent) variables to predict the outcome of a response (dependent) variable. The model creates a relationship in the form of a straight line (linear) that best approximates all the individual data points.

MLR
Input
Instances, feature vectors, real-numbered target values
Output
Regression model
Input format
Dependent on implementation, e.g., Weka's ARFF format
Output format
Dependent on implementation, e.g., Weka: plain text; binary models
User-specified parameters
None
Applicability domain
The leverage of a chemical provides a measure of the distance of the chemical from the centroid of its training set. Chemicals in the training set have leverage values between 0 and 1. A warning leverage is generally fixed at 3p/n, where n is the number of training chemicals, and p the number of descriptors plus one. A leverage value greater than the warning leverage is considered large. Prediction bounds on a predicted response can be computed by adding or subtracting the quantity $t_{a/2} S \sqrt{1+\mathbf{x}'(\mathbf{X}'\mathbf{X})^{-1}\mathbf{x}}$, where $t_{a/2}$ is the appropriate point based on the T_{n-p+1} distribution, S is an estimate of the variance corresponding to the dependent variable, $\mathbf{X}$ is the model specification matrix and $\mathbf{x}$ is a vector containing the values of the independent variables for the specific response.

Reporting information Apart from the model coefficients, several other statistical results are reported by the MLR method concerning the training data: coefficient of determination, adjusted coefficient of determination, F-statistic, t-statistic for each individual independent variable, confidence intervals, residuals and variance inflation factor.
Background (publication date, popularity/level of familiarity, rationale of approach, further comments) Multiple linear regression (MLR) is the most widely used mathematical technique in QSAR analysis.
Bias (instance-selection bias, feature-selection bias, combined instance-selection/feature-selection bias, independence assumptions?, ...) Feature-selection bias The error is assumed to be a random variable with a mean of zero conditional on the explanatory variables. The independent variables are error-free. The predictors must be linearly independent, i.e. it must not be possible to express any predictor as a linear combination of the others. The errors are uncorrelated, that is, the variance-covariance matrix of the errors is diagonal and each non-zero element is the variance of the error.
Lazy learning/eager learning Eager learning
Interpretability of models (black box model?, ...) Good (linear model, i.e., produces a simple linear weighting of given features), If the variables are standardized to have mean of zero and standard deviation of one, then the regression coefficients (beta coefficients). Allow the comparison of the relative contribution of each independent variable in the prediction of the dependent variable.
Performance (time/space complexity, running times, memory consumption, ...) Regression coefficients in MLR model can be estimated using the least squares procedure, which minimizes the sum of the squared residuals. The aim of this procedure is to give the smallest possible sum of squared differences between the true dependent variable values and the values calculated by the regression model. The least-squares problem can be formulated as an unconstrained quadratic optimization problem and is of low computational complexity, i.e. the method is suitable for large databases. Orthogonal decomposition methods for solving the problem are slower, but more numerically stable. The suitability of the method for deriving QSARs has been

illustrated in numerous applications.
OT availability Available in any statistical package, e.g., in the Weka open source data mining workbench
Licence /Dependencies None
Convenience of integration Webservices: very easy / implemented in Java
Priority (A, B, C) A
Author of method / Contact -
Author of description Haralambos Sarimveis
Contact within OT hsarimv@central.ntua.gr
Comments (including reviews)

3.2.3 SVM

Support vector machines (SVM) are a set of supervised learning methods used for classification and regression. In the most widely used two-class SVM classification method, input data are viewed as two sets of vectors in the multi-dimensional input space. The SVM classifier constructs a separating hyperplane in that space, one which maximizes the margin between the two data sets. The method is extended to multi-class and nonlinear classification problems by using nonlinear kernel function. To obtain an optimum classifier for nonseparable data, a penalty is introduced for misclassified data. This penalty is zero for patterns classified correctly, and has a positive value that increases with the distance from the corresponding hyperplane for patterns that are not situated on the correct side of the classifier. Similar concepts are used in the SVM regression problem, where the objective is to identify a function that for all training patterns has a maximum deviation ϵ from the target (experimental) values.

The LIBSVM library is a popular open-source software tool that has implemented both classification and regression SVM methods. The software has no dependencies, receives input data in plain text format and its output is also plain test.

For further information we refer to the Website: www.csie.ntu.edu.tw/~cjlin/libsvm/

SVM
Input

Instances, feature vectors, real-numbered target values / class values
Output Regression / classification model
Input format Dependent on implementation, e.g., Weka's ARFF format
Output format Dependent on implementation, e.g., Weka: plain text; binary models
User-specified parameters The user needs to select the kernel function. The LIBSVM library gives four options: linear, polynomial, radial basis function and sigmoid function. Each kernel function except from the linear kernel is associated with a number of tuning parameters. If the user select the polynomial function, he needs to define three tuning parameters, the radial basis function includes one tuning parameters and two tuning parameters need to be adjusted for the sigmoid function. For classification problems, the user also needs to adjust the parameter C, which controls the penalty for classification errors. For regression problems, the user needs to adjust the parameter ϵ, which determines the limits of the approximations tube and the parameter C, which controls the penalty associated with deviations larger than ϵ.
Applicability domain The applicability domain can calculated from the distribution of similarities between each compound and its k nearest neighbors in the training set (similarities are computed as Euclidean distances between compounds represented by their multiple chemical descriptors). The standard cutoff value to define the applicability domain for a QSAR model places its boundary at one-half of the standard deviation calculated for the distribution of distances between each compound in the training set and its k nearest neighbors in the same set (assuming a Boltzmann-like distribution of these distances). Thus, if the distance of the test compound from any of its k nearest neighbors in the training set exceeds the threshold, the prediction is considered unreliable. The method is described in [TRO03] Probability information can be computed using the methods described in Wu et al. (2004) for classification and in Lin and Weng (2004) for regression
Reporting information The following information is reported: model parameters, predictions on the training set, (residuals, sum of squared errors, root mean squared error, F- statistic, coefficient of determination in regression problems), (overall %accuracy, %accuracy for each individual class, probability estimates in classification problems)
Background (publication date, popularity/level of familiarity, rationale of approach, further comments)

Support vector machines represent an extension to nonlinear models of the generalized portrait algorithm developed by Vapnik and Lerner. The SVM algorithm is based on the statistical learning theory and the Vapnik Chervonenkis (VC) dimension. In the current formulation, the SVM algorithm was developed at AT&T Bell Laboratories by Vapnik et al. [COR95]. The algorithm was extended to tackle regression problems [VAP98]. SVM methods have been applied with success for developing QSAR, where in addition to standard kernel function, molecular similarity kernel, such as the Tanimoto similarity kernel, have been utilized.
Bias (instance-selection bias, feature-selection bias, combined instance-selection/feature-selection bias, independence assumptions?, ...)
Instance-selection bias
Lazy learning/eager learning
Eager learning
Interpretability of models (black box model?, ...)
It depends on the kernel function. A linear kernel function produces a linear model, i.e. a simple linear weighting of given features with good interpretability. A nonlinear kernel function generates a nonlinear model, which can be considered as a black box model.
Performance (time/space complexity, running times, memory consumption, ...)
An SVM classification or regression problem is formulated as a constrained quadratic programming optimization problem. Typically, the dual optimization problem is solved, which allows to easily incorporate nonlinear kernel functions. Solution of the SVM optimization problem is more computationally intensive compared to the unconstrained MLR optimization problem, but it is still suitable for large databases.
OT availability
Available in many statistical or machine learning packages, e.g., for the LIBSVM library there exist interfaces for Python, R, Splus, Perl, Ruby and Weka languages
Licence /Dependencies
Use of LIBSVM sources, with or without modification, are permitted provided that redistributions of source code retain the copyright notice, the list of conditions and a disclaimer.
Convenience of integration
Web services: very easy / implemented in Java and C++
Priority (A, B, C)
B
Author of method / Contact

V. Vapnik
Author of description Haralambos Sarimveis
Contact within OT hsarimv@central.ntua.gr
Comments (including reviews)

3.2.4 RUMBLE (TUM)

RUMBLE (RUle and Margin Based LEarner) is a statistically motivated rule learning system based on the Margin Minus Variance (MMV) optimization criterion [RUE08]. It can be adapted flexibly to a given dataset: First, different types of data (structures, physico-chemical properties, logical background knowledge, ...) can be handled by different plug-ins of the system (e.g., FTM plugin, Prolog plugin, Meta plugin, ...). Second, the learning algorithm can be adapted to the noise level in the data by two regularization parameters. The main algorithm performs a forward selection of variables as for linear or logistic regression models. The models learned by RUMBLE are linear classifiers, i.e., they provide a linear weighting of the input features.

The software is implemented in the C++ programming language and was developed for the Linux and Mac OS X operating systems. The RUMBLE software is dependent on the OpenBabel (www.openbabel.org) chemistry toolbox. In case the Prolog plugin is used, there is also a dependency on the specific Prolog system used. RUMBLE provides no graphical user interface (GUI) and is executed via the command line. The input format accepted at the moment is Weka's [WIT99] ARFF format. XML input is under development. RUMBLE's output is plain text.

For further information, we refer to the original publication [RUE08] and the website wwwkramer.in.tum.de/research/machine_learning/margin_based

RUMBLE
Input Instances, feature vectors, class values
Output Classification model
Input format Weka's ARFF format plus text; Soon XML
Output format Plain text
User-specified parameters Norm used for learning

Bound constant
Applicability domain
Reporting information Performance measures (sensitivity, specificity, AUC, prediction accuracy)
Background (publication date, popularity/level of familiarity, rationale of approach, further comments) Published 2006-2008, best theory paper award at ILP 2006. Adopts the concept of a margin from the Support Vector Machine (SVM), but focuses on the selection of features instead of the selection of instances. Does not use kernels. Useful tool with regularization parameter for noise handling and plug-ins for various data types (e.g., chemical structures and quantitative descriptors)
Bias (instance-selection bias, feature-selection bias, combined instance-selection/feature-selection bias, independence assumptions?, ...) Feature-selection bias
Lazy learning/eager learning Eager learning
Interpretability of models (black box model?, ...) Good (linear classifier, i.e., produces a simple linear weighting of given features)
Performance (time/space complexity, running times, memory consumption, ...) Optimization is linear in the number of instances -- thus, theoretically suitable for large datasets -- and cubic in the number of features. Practically reasonable running times on standard (Q)SAR data. Excellent predictive performance in practice (see [RUE08]).
OT availability TUM
Licence /Dependencies OpenBabel (open source), [Prolog (open source)]
Convenience of integration C++ => OS dependent compilation (Win vs. Linux); command line tool
Priority (A, B, C) C
Author of method / Contact

	Ulrich Rückert (rueckert@icsi.berkeley.edu), Stefan Kramer (kramer@in.tum.de)
Author of description	
	Tobias Girschick
Contact within OT	
	kramer@in.tum.de
Comments (including reviews)	

3.2.5 KNN

The k-nearest neighbors algorithm (kNN) is a method for classifying objects based on closest training examples in the feature space. It is a type of instance-based learning, or lazy learning where the function is only approximated locally and all computation is delayed until classification. A majority vote of an object's neighbors is used for classification, with the object being assigned to the class most common amongst its k (positive integer, typically small) nearest neighbors. If k is set to 1, then the object is simply assigned to the class of its nearest neighbor. The kNN algorithm can also be applied for regression in the same way by simply assigning the property value for the object to be the average of the values of its k nearest neighbors. It can be useful to weight the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. No explicit training step is required since training consists of just storing training instance feature vectors and corresponding class labels. In order to identify neighbors, the objects are represented by position vectors in a multidimensional feature space. It is usual to use the Euclidean distance, but also further distance measures, such as the Manhattan distance could be used instead. In the classification/testing phase, the test sample is represented as a vector in the feature space. Distances from this vector to all stored vectors are computed and the k closest samples are selected to determine the class/real-value of the test instance.

The k-nearest neighbor algorithm is sensitive to the local structure of the data. The best choice of k depends upon the data; generally, larger values of k reduce the effect of noise on the classification, but make boundaries between classes less distinct. A good k can be selected by various heuristic techniques like cross-validation. The accuracy of the kNN algorithm can be severely degraded by the presence of noisy or irrelevant features, or if the feature scales are not consistent with their importance.

For further information we refer the reader to the literature [AHA91][MIT97].

KNN	
Input	
	Instances, feature vectors, class values
Output	
	Classification model (actually training instances are stored; lazy learning method)
Input format	
	Weka's ARFF format

Output format Plain text, model binary
User-specified parameters • k (the number of neighbors to use) • whether hold-one-out cross-validation will be used to select the best k value • whether to use distance weighting • whether the mean squared error is used rather than mean absolute error when doing cross-validation for regression problems • distance function.
Applicability domain
Reporting information Performance measures (Confusion matrix, precision, recall, AUC, F-measure, true (false) positive rate, prediction accuracy)
Background (publication date, popularity/level of familiarity, rationale of approach, further comments) Very popular method in the machine learning community. Simple approach that often yields high predictive power. Can be used for classification and regression.
Bias (instance-selection bias, feature-selection bias, combined instance-selection/feature-selection bias, independence assumptions?, ...) Instance-selection bias
Lazy learning/eager learning Lazy learning
Interpretability of models (black box model?, ...) Good
Performance (time/space complexity, running times, memory consumption, ...) Linear in the number of instances and features -- thus, theoretically suitable for large datasets.
OT availability TUM
Licence /Dependencies WEKA (open source)
Convenience of integration

Webservices: very easy / implemented in Java
Priority (A, B, C) A
Author of method / Contact Fix E., Hodges J.L., 1951
Author of description Fabian Buchwald
Contact within OT kramer@in.tum.de
Comments (including reviews)

3.2.6 Lazar (IST)

Lazar is a k-nearest-neighbor approach to predict chemical endpoints from a training set based on structural fragments. It uses a SMILES file and precomputed fragments with occurrences as well as target class information for each compound as training input. It also features regression, in which case the target activities consist of continuous values. Lazar uses activity-specific similarity (i.e. each fragment contributes with its significance for the target activity) that is the basis for predictions and confidence index for every single prediction.

For classification, a weighted nearest neighbor voting is the standard prediction, whereas for regression a kernel model based on activity-specific similarity is used by default. A kernel model is also available for classification, as well as a multilinear model for regression.

The software is implemented in the C++ programming language and was developed for Linux. Lazar is dependent on the OpenBabel (openbabel.org) chemistry toolbox, GNU Scientific Library, as well as on R and the R package kernlab. Lazar is a plugin for Ruby on rails to exhibit its functionality as webservice, in which case it also provides a graphical user interface (GUI), however it can still be executed from the command line. The input format accepted at the moment is flat files, each line a SMILES string / a YAML formatted fragment with occurrence numbers / an id followed by target activity name and value, respectively. Lazar's output is YAML, yielding reach information about query compound, predicted and database activity, neighbors and significant fragments. For further information we refer the reader to the according literature [MAU08,HEL06].

Lazar
Input
Output
Input format

Plain text in custom tab separated format
Output format Plain text in YAML format
User-specified parameters None
Applicability domain
Reporting information Neighbors and significant features for each prediction
Background (publication date, popularity/level of familiarity, rationale of approach, further comments) Published 2006 (classification) and 2008 (regression), presently shipped with a lot of classification and regression endpoint datasets. A web-based prototype is available from lazar.in-silico.de. Provides self-contained, information rich predictions, suitable for one-click interfaces. Usable without expert knowledge, provides automatic applicability domain estimation.
Bias (instance-selection bias, feature-selection bias, combined instance-selection/feature-selection bias, independence assumptions?, ...) Feature-selection bias
Lazy learning/eager learning Lazy learning
Interpretability of models (black box model?, ...) Intuitive (neighbors, significant fragments, visual depiction).
Performance (time/space complexity, running times, memory consumption, ...) Linear in the number of neighbors for standard classification, polynomial for regression. Memory consumption has recently been improved
OT availability IST
Licence /Dependencies GPL
Convenience of integration C++ => Linux dependent compilation for command line tool, RoR-Webservice platform independent.

Priority (A, B, C) B
Author of method / Contact Christoph Helma (Classification), Andreas Maunz (Regression)
Author of description Andreas Maunz
Contact within OT helma@in-silico.de, maunza@fdm.uni-freiburg.de
Comments (including reviews)

3.2.7 iSAR(TUM)

iSAR (instance-based structure-activity relationships) is an implementation of a lazy SAR algorithm. In lazy SARs, classifications are particularly tailored for each test compound. Therefore, it is possible to make the most of the structure of a test compound. iSAR uses subgraphs and paths that are generated by e.g., gSpan' [JK05] or unrooted trees that are generated by e.g., Free Tree Miner [RUE04] as features for the classification task. These substructures are derived from a test compound to determine similar structures. In order to obtain a well-balanced and representative set of structural descriptors, this set can be enriched by strongly activating or deactivating fragments from the training set and subsequently redundant fragments (use only closed features) can be removed. Finally, a k-Nearest Neighbor classification with one k or for several values of k is performed and a vote among the resulting predictions is taken. The validation is performed via leave-one-out cross validation (LOOCV).

iSAR is implemented in the Perl programming language. The iSAR software is dependent on a substructural feature generator, e.g., gSpan' or Free Tree Miner (FTM), JOELIB [JOELIB] and Weka [WIT99]. iSAR provides no graphical user interface and is executed via the command line. The input format accepted is an internal iSAR format. Perl scripts that convert the output of FTM or gSpan' to this format are provided. iSARs output is program specific plain text.

For further information, we refer to the original publication [SOM07] and the website wwwkramer.in.tum.de/research/pubs/articlereference.2008-03-17.2708343675

iSAR
Input Instances, chemical substructure feature vectors, class values
Output Classification model (actually training instances are stored; lazy learning method)
Input format lazySAR internal format (.ibf, .fbi, .count)

Output format Program specific plain text (.result, .info)
User-specified parameters The user can choose to use all features, an upper limit for the number of features or to use only closed features. Further he can choose the number of non-occurring substructures to add as features to test instance feature set. Only the most significant (in relation to the class) non-occurring features are added. The method to determine significance can be chosen amongst Chi Square, Cole, G- index and Information Gain.
Applicability domain
Reporting information (Combined) prediction for each instance, overall statistics (confusion matrix)
Background (publication date, popularity/level of familiarity, rationale of approach, further comments) Published 2007. Follows the concept of lazy instance-based learning. Similar to lazar. Extends simple instance-based learners by the three techniques: enrichment (use of strongly activating or deactivating fragments from the training set), removing redundancy (use only closed features), and voting (building several KNN-Classifier und vote amongst their predictions). Useful tool for SAR datasets with congeneric and non-congeneric compounds.
Bias (instance-selection bias, feature-selection bias, combined instance-selection/feature-selection bias, independence assumptions?, ...) Instance-selection bias
Lazy learning/eager learning Lazy learning
Interpretability of models (black box model?, ...) Good (kNN classifier)
Performance (time/space complexity, running times, memory consumption, ...) Linear in the number of instances and features -- thus, theoretically suitable for large datasets. Good predictive performance in relation to more complex models (see [SOM07]).
OT availability TUM
Licence /Dependencies

FTM, gSpan'(open source), JOELIB(open source), WEKA (open source)
Convenience of integration
FTM, gSpan': C++ => OS dependent compilation (Win vs. Linux) JOELIB, WEKA: Java command line tool
Priority (A, B, C)
C
Author of method / Contact
Selina Sommer, Stefan Kramer (kramer@in.tum.de)
Author of description
Fabian Buchwald
Contact within OT
kramer@in.tum.de
Comments (including reviews)

3.2.8 SMIREP/SMIPPER (ALU-FR)

SMIREP/SMIPPER [KAR06] is based on combining feature generation and rule learning into one integrated package. It constructs features, or sub graphs, by defragmenting the SMILES representations of the training data, and refining these on the fly during the learning process. The underlying learning algorithm is similar to that of the IREP rule learner employing a reduced error pruning approach. SMIREP is able to incorporate external, predefined SMART patterns – like functional groups – as well as able to incorporate physico-chemical properties during rule construction. The resulting models learned by SMIREP are sets of rules. SMIPPER employs essentially a similar approach, by refining the found rule set repeatedly. The system can be run in three modes: train/test, k-fold cross validation, or leave-one-out cross validation. Optionally, for each test set or fold receivers operating characteristic curves are constructed for visualization purposes.

The software is implemented in the Python programming language and was developed for the Linux operating system. The SMIREP software is dependent on the OpenBabel (www.openbabel.org) chemistry toolbox. SMIREP is executed via a command line interface. The input format accepted are plain SMILES file or Weka's [WIT99] ARFF format – containing the attribute SMILES and the pre-computed physico-chemical properties. The additional SMARTS file for functional groups is a plain ASCII text file, containing the SMARTS pattern as well as a group identifier.

For further information, we refer to the original publication [KAR06] and the website www.karwath.org/systems/smirep.html.

SMIREP/SMIPPER
Input

Output
Input format SMILES files or Weka's ARFF format
Output format Plain text
User-specified parameters Evaluation heuristic: compute_v or wracc (weighted relative accuracy) Minimum number of instances covered Minimum number of seeds Stopping error rate (default – apriori distribution)
Applicability domain None
Reporting information
Background (publication date, popularity/level of familiarity, rationale of approach, further comments) Published 2006. Employs heuristic way of determining activity by defragmenting SMILES strings of instances and refines the resulting fragments during rule construction. Does not require pre-constructed fragments or features.
Bias (instance-selection bias, feature-selection bias, combined instance-selection/feature-selection bias, independence assumptions?, ...) Feature-selection bias
Lazy learning/eager learning Eager learning
Interpretability of models (black box model?, ...) Very good (sets of rules of SMILES string (or constraints based on physico-chemical properties and/or predefined SMARTS pattern))
Performance (time/space complexity, running times, memory consumption, ...) Due to a heuristic selection of possible refinements good running times on standard (Q)SAR data. Comparable predictive performance (see [KAR06]), aimed at a first investigation tool.

OT availability ALU-FR
Licence /Dependencies GPL / OpenBabel (open source)
Convenience of integration Python => OS dependent compilation (Win vs. Linux); command line tool
Priority (A, B, C) B
Author of method / Contact Andreas Karwath, Luc De Raedt
Author of description Andreas Karwath
Contact within OT karwath@informatik.uni-freiburg.de
Comments (including reviews)

3.2.9 J48

J48 [QUI93] implements Quinlan's C4.5 algorithm [QUI92] for generating a pruned or unpruned C4.5 decision tree. C4.5 is an extension of Quinlan's earlier ID3 algorithm. The decision trees generated by J48 can be used for classification. J48 builds decision trees from a set of labeled training data using the concept of information entropy. It uses the fact that each attribute of the data can be used to make a decision by splitting the data into smaller subsets. J48 examines the normalized information gain (difference in entropy) that results from choosing an attribute for splitting the data. To make the decision, the attribute with the highest normalized information gain is used. Then the algorithm recurs on the smaller subsets. The splitting procedure stops if all instances in a subset belong to the same class. Then a leaf node is created in the decision tree telling to choose that class. But it can also happen that none of the features give any information gain. In this case J48 creates a decision node higher up in the tree using the expected value of the class.
J48 can handle both continuous and discrete attributes, training data with missing attribute values and attributes with differing costs. Further it provides an option for pruning trees after creation.

For further information, we refer to the original publications [QUI93].

J48
Input Instances, feature vectors, class values
Output

Classification model (decision tree)
Input format Weka's ARFF format
Output format Plain text, model binary
User-specified parameters The user can choose whether to use binary splits on nominal attributes when building the trees, the minimum number of instances per leaf, whether counts at leaves are smoothed based on Laplace, whether pruning is performed, whether to consider the subtree raising operation when pruning , the confidence factor used for pruning (smaller values incur more pruning), whether reduced-error pruning is used instead of C.4.5 pruning (amount of data used for reduced-error pruning (one fold is used for pruning, the rest for growing the tree), seed used for randomizing the data when reduced-error pruning is used).
Applicability domain
Reporting information Performance measures (Confusion matrix, precision, recall, AUC, F-measure, true (false) positive rate, prediction accuracy)
Background (publication date, popularity/level of familiarity, rationale of approach, further comments) Published in 1993. Implementation of the well-known C4.5 decision tree learner. Extends C4.5 by providing besides C4.5pruning reduced error pruning.
Bias (instance-selection bias, feature-selection bias, combined instance-selection/feature-selection bias, independence assumptions?, ...) Feature-selection bias
Lazy learning/eager learning Eager learning
Interpretability of models (black box model?, ...) Good (produced is a decision tree)
Performance (time/space complexity, running times, memory consumption, ...) Fast, applicable to large datasets
OT availability WEKA

Licence /Dependencies WEKA (open source)
Convenience of integration Webservices: very easy / implemented in Java
Priority (A, B, C) A
Author of method / Contact Ross Quinlan [QUI93]
Author of description Fabian Buchwald
Contact within OT kramer@in.tum.de
Comments (including reviews)

3.2.10 M5P

M5P [WAN97] is a reconstruction of Quinlan's M5 algorithm [QUI92] for inducing trees of regression models. M5P combines a conventional decision tree with the possibility of linear regression functions at the nodes.

First, a decision-tree induction algorithm is used to build a tree, but instead of maximizing the information gain at each inner node, a splitting criterion is used that minimizes the intra-subset variation in the class values down each branch. The splitting procedure in M5P stops if the class values of all instances that reach a node vary very slightly, or only a few instances remain.

Second, the tree is pruned back from each leaf. When pruning an inner node is turned into a leaf with a regression plane.

Third, to avoid sharp discontinuities between the subtrees a smoothing procedure is applied that combines the leaf model prediction with each node along the path back to the root, smoothing it at each of these nodes by combining it with the value predicted by the linear model for that node.

Techniques devised by Breiman et al. [BRE84] for their CART system are adapted in order to deal with enumerated attributes and missing values. All enumerated attributes are turned into binary variables so that all splits in M5P are binary. As to missing values, M5P uses a technique called "surrogate splitting" that finds another attribute to split on in place of the original one and uses it instead. During training, M5P uses as surrogate attribute the class value in the belief that this is the attribute most likely to be correlated with the one used for splitting. When the splitting procedure ends all missing values are replaced by the average values of the corresponding attributes of the training examples reaching the leaves. During testing an unknown attribute value is replaced by the average value of that attribute for all training instances that reach the node, with the effect of choosing always the most populous subnode.
M5P generates models that are compact and relatively comprehensible.

For further information, we refer to the original publications [WAN97], [QUI92], [BRE84].

M5P
Input Instances, feature vectors, real-numbered target values
Output Tree of regression models
Input format Weka's ARFF format
Output format Plain text, model binary
User-specified parameters The user can choose whether instead of a model tree a regression tree is built, the minimum number of instances to allow at a leaf node, whether the tree should be pruned and whether to use unsmoothed predictions.
Applicability domain
Reporting information Performance measures (Correlation coefficient, mean absolute error, root mean squared error, relative absolute error, root relative squared error)
Background (publication date, popularity/level of familiarity, rationale of approach, further comments) Published in 2007. Uses features from the well-known CART system and reimplements Quinlan's well-known M5 algorithm with modifications and seems to outperform it. M5P can deal effectively with enumerated attributes and missing values. Smoothing substantially increases prediction accuracy.
Bias (instance-selection bias, feature-selection bias, combined instance-selection/feature-selection bias, independence assumptions?, ...) Feature-selection bias
Lazy learning/eager learning Eager learning
Interpretability of models (black box model?, ...) Good (produced is a model tree)

Performance (time/space complexity, running times, memory consumption, ...) Fast, applicable to large datasets
OT availability WEKA
Licence /Dependencies WEKA (open source)
Convenience of integration Webservices: very easy / implemented in Java
Priority (A, B, C) C
Author of method / Contact Y. Wang, I. H. Witten (ihw@cs.waikato.ac.nz)
Author of description Fabian Buchwald
Contact within OT kramer@in.tum.de
Comments (including reviews)

3.2.11 Fuzzy-means (NTUA)

Fuzzy-means is a training method for Radial Basis Function (RBF) neural networks and is based on the fuzzy partition of the input space, which is produced by defining a number of triangular fuzzy sets in the domain of each input variable. The centers of these fuzzy sets form a multidimensional grid on the input space. A rigorous selection algorithm chooses the most appropriate vertices on the grid, which are then used as the hidden node centers in the resulting RBF network model. The so called "fuzzy-means" training method does not need the number of centers to be fixed before the execution of the method. Due to the fact that it is a one-pass algorithm, it is extremely fast, even in the case of a large database of input-output training data. The method was originally developed for solving nonlinear regression problems. A variant of the method for solving classification problems has also been developed.

The algorithm has been implemented in the Matlab programming environment. Translation into C++ programming language is under development. The input formats accepted are Excel files and plain text. The output is plain text.

For further information, we refer to the original publications [SAR02], [SAR06].

Fuzzy-means
Input

Output
Input format Plain text or Excel file
Output format Plain text
User-specified parameters The user needs to define one tuning parameter, namely the number of fuzzy sets that are utilized to partition each input dimension.
Applicability domain The interpolation space of the model is defined by computing the smallest convex area that contains the descriptors of the training set. For the classification problem, one output node is used for each possible class. The confidence for a particular prediction is higher when the value of a single output node is closer to 1, while the values of all remaining output nodes are closer to 0.
Reporting information The following information is reported: number of hidden nodes, hidden node centers, widths of Gaussian function, output weights, predictions on the training set, (residuals, sum of squared errors, root mean squared error, F- statistic, coefficient of determination in regression problems), (overall %accuracy and %accuracy for each individual class in classification problems).
Background (publication date, popularity/level of familiarity, rationale of approach, further comments) Fuzzy means for regression, published 2002 [SAR02]. Fuzzy means for classification, published 2006 [SAR06]. The idea behind the selection algorithm is to place the centers in the multidimensional input space, so that the distance between any two center locations is guaranteed to be greater than a lower limit, which is defined by the length of the edges on the grid. At the same time, the algorithm assures that for any input example in the training set there is at least one selected hidden node that is close enough, according to an appropriately defined distance criterion.
Bias (instance-selection bias, feature-selection bias, combined instance-selection/feature-selection bias, independence assumptions?, ...) Feature-selection bias
Lazy learning/eager learning Eager learning

Interpretability of models (black box model?, ...) Black box model
Performance (time/space complexity, running times, memory consumption, ...) Implementation of the method requires $n^*I-(n^2+n)/2$ distance calculations (where n is the number of training chemicals and I is the number of hidden nodes) and the solution of a least-squares problem where the independent variables are equal to I. The method is orders of magnitude faster compared to the standard RBF training algorithms and is suitable for large databases. The method has been successfully tested in various regression and classification problems, including QSAR problems [MEL06].
OT availability NTUA
Licence /Dependencies Matlab
Convenience of integration Translation to C++ is under development
Priority (A, B, C) B
Author of method / Contact H. Sarimveis [SAR02]
Author of description Haralambos Sarimveis
Contact within OT hsarimv@central.ntua.gr
Comments (including reviews)

3.2.12 MakeSCR (IBMC)

Self-consistent regression (SCR)

Delphi implementation of a self-consistent regression algorithm. Using self-consistent regression one can obtain the best QSAR/QSPR model for the training set with a large number of descriptors. SCR is based on least-squares' regularized method. The main features of SCR are the following:

- variable selection
- model building
- parameters of model calculation (R2, Q2, SD, Fisher)
- validation by LOOCV
- y-scrambling

Self-consistent regression (SCR)
Input Feature vectors, real-numbered target values
Output Regression model
Input format Text file format
Output format Text file format
User-specified parameters None
Applicability domain The leverage of a chemical provides a measure of the distance of the chemical from the centroid of the training set. Chemicals in the training set have leverage values varied from 0 to 1.
Reporting information Performance measures (Correlation coefficient, Q^2 values, standard deviation, Fisher coefficient, number variables
Background (publication date, popularity/level of familiarity, rationale of approach, further comments) [LAG07]
Bias (instance-selection bias, feature-selection bias, combined instance-selection/feature-selection bias, independence assumptions?, ...)
Lazy learning/eager learning Eager learning
Interpretability of models (black box model?, ...) Good (linear model, i.e., produces a simple linear weighting of given features), If the variables are standardized to have mean of zero and standard deviation of one, then the regression coefficients (*beta* coefficients) allow the comparison of the relative contribution of each independent variable in the prediction of the dependent variable.
Performance (time/space complexity, running times, memory consumption, ...)

Matrix of 1000 x 500 dimensions is calculated in 5 minutes; while matrix of 12000 x 3000 dimensions is calculated in 4 hours (usual PC).
OT availability IBMC
Licence /Dependencies GPL
Convenience of integration Delphi => OS dependent compilation (Windows); Windows interface.
Priority (A, B, C) B
Author of method / Contact Filimonov Dmitry
Author of description Alexey Zakharov
Contact within OT alexey.zakharov@ibmc.msk.ru
Comments (including reviews) Good predictivity was demonstrated during the testing of the method in a dozen of case-studies covering different chemical series and diverse types of biological activity.

3.2.13 MaxTox (SIT-JNU)

The algorithm uses 2-D based QSAR to determine toxicity of molecules by comparing to a set of known toxic molecules. The QSAR in this case consists of finding descriptors from the database of toxic molecules using the maximum common substructure determination algorithm and then using these descriptors to develop a predictive model for toxicity. The test molecule is fed to this predictive model to get a score regarding its toxicity.

At every level (mentioned below), the algorithm consists of two parts – screening and rigorous graph matching. The main function of screening is to eliminate those molecules which are beyond some minimum similarity threshold (in terms of their graphs) so that the computationally complex graph matching is optimized. Broadly the algorithm consists of the following steps:

Toxicity Data will be acquired from the other members of the consortium.

Clustering of the molecules within this database based on Toxicity endpoints (EP). Minimum Common Substructure (MCS) scores are generated based on clique detection algorithm [BRO73] within each EP cluster.

Comparing the query molecule to each cluster (EP based) and finding an MCS score with respect to molecules of each cluster [JWR02].

Using MCS score(s) in a Machine Learning algorithm, to generate predictive models.

The software is primarily implemented in the JAVA and will be developed for a Linux based system. MaxTox software is dependent on the open source chemistry development kit (CDK) (sourceforge.net/projects/cdk) and OpenBabel (openbabel.org). MaxTox may provide a basic graphical user interface (GUI) in future. Currently it is executed via the command line.

The input format accepted by MaxTox is the widely used MDL file format (www.symyx.com/downloads/public/ctfile/ctfile.jsp). MaxTox output formats are program specific plain text files and MCS in format SDF format.

MaxTox
Input SDF files containing structure + activity (toxicity)
Output MCS Score
Input format SDF(MDL)
Output format Comma separated values and SDF
User-specified parameters - minimum number of matching atoms and bonds - minimum number of ring atoms, hetero-atoms
Applicability domain Can be applied on diverse(non-congeneric) chemical structures
Reporting information MCS , similarity matrices for building model
Background (publication date, popularity/level of familiarity, rationale of approach, further comments) Published in 2006, [PRA06] elaborates the scope of the hypothesis, that it may be possible to find a set of common scaffold(s) from diverse compound set which contribute significantly (positively/negatively) towards the biological activity. In the present algorithm, we propose to extend this hypothesis to derive a predictive toxicity score. This score will be based on MCS (Maximum Common Substructure) score with respect to clusters of compounds (based on toxicological endpoints).
Type of descriptor (substructural/physico-chemical, expressiveness: paths, trees, subgraphs,

wildcards?, suitability for similarity/distance calculations?, ...) MCS and similarities
Performance (time/space complexity, running times, memory consumption, ...) Dependent on number, size and structural complexity of molecules. Multi-threading algorithm may also be used to decrease the run times.
OT availability SIT-JNU: Being Developed for OpenTox specifically
Licence /Dependencies GPL, CDK (Chemistry Development Kit) (open source), OpenBabel (open source), R (statistical modeling tool) , python,C++
Convenience of integration JAVA and C++ => OS dependent compilation (Win vs. Linux); command line tool; Webservices: tentatively AJAX based implementation.
Priority (A, B, C) C
Author of method / Contact Indira Ghosh
Author of description Surajit Ray
Contact within OpenTox Indira Ghosh <indirag@mail.jnu.ac.in>, sunil@seascapelearning.com
Comments (including reviews)

3.2.14 ToxTree (IDEA)

Toxtree is a full-featured and flexible user-friendly open source application, which is able to estimate toxic hazard by applying a decision tree approach. Currently it includes the following modules:

1. Cramer rules [CRA78]
2. Verhaar scheme for predicting toxicity mode of actions [VER92]
3. A decision tree for estimating skin irritation and corrosion potential, based on rules published in [WAL05]
4. A decision tree for estimating eye irritation and corrosion potential, based on rules published in [GER05]
5. A decision tree for estimating carcinogenicity and mutagenicity [BEN07], [BEN08]

Toxtree could be applied to datasets from various compatible file types. User-defined molecular structures are also supported - they could be entered by SMILES, or by using the built-in 2D structure diagram editor.

The Toxtree has been designed with flexible capabilities for future extensions in mind (e.g. other classification schemes that could be developed at a future date). New decision trees with arbitrary rules can be built with the help of graphical user interface or by developing new plug-ins.

Structural alerts and property conditions arranged as a decision tree
Input
Output
Input format MOL, SDF, CSV, TXT, SMILES, CML file or Java class, representing the chemical structure in CDK library
Output format MOL, SDF, CSV, TXT, SMILES, CML file or Java class, representing the assigned categorical value
User-specified parameters None
Applicability domain Implicit applicability domain
Reporting information
Background (publication date, popularity/level of familiarity, rationale of approach, further comments) [PAT08] , [CRA78], [VER92], [WAL05], [GER05], [BEN07], [BEN08]
Bias (instance-selection bias, feature-selection bias, combined instance-selection/feature-selection bias, independence assumptions?, ...) Predefined rules, based on publications. No learning phase, no feature selection.
Lazy learning/eager learning Predefined rules ; does not involve a learning phase
Interpretability of models (black box model?, ...) Highly interpretable, structural alerts and properties

Performance (time/space complexity, running times, memory consumption, ...) Fast
OT availability toxtree.sourceforge.net
Licence /Dependencies GPL Dependencies: CDK MOPAC 7.1 for the Benigni/Bossa rules for predicting carcinogenicity and mutagenicity
Convenience of integration Implemented in Java , easy to integrate
Priority (A, B, C) B
Author of method / Contact Various authors or original decision trees, Implementation by IDEA [PAT08]
Author of description Nina Jeliazkova
Contact within OT nina@acad.bg
Comments (including reviews)

3.2.15 PLS

One way to understand Partial-least squares regression (PLS) is that it simultaneously projects the x and y variables onto the same subspace in such a way that there is a good relationship between the predictor and response data. Another way to see PLS is that it forms "new" x variables as linear combinations of the old ones, and subsequently uses these new linear combinations as predictors of y.

Hence, as opposed to MLR PLS can handle correlated variables, which are noisy and possibly also incomplete. An easy open source implementation of PLS is available in the latest WEKA release.

PLS

Input
Output
Input format Weka's ARFF format
Output format Weka's ARFF format
User-specified parameters -
Applicability domain
Reporting information Statistical measures of performance; number of components
Background (publication date, popularity/level of familiarity, rationale of approach, further comments) Standard statistical based method. Belongs to the family of NILES (Non-linear iterative least squares)
Bias (instance-selection bias, feature-selection bias, combined instance-selection/feature-selection bias, independence assumptions?, ...)
Lazy learning/eager learning Eager learning
Interpretability of models (black box model?, ...)
Performance (time/space complexity, running times, memory consumption, ...)
OT availability Open source implementation in Weka;
Licence /Dependencies GPL
Convenience of integration

Easy because of Java implementation (considering WebServices)
Priority (A, B, C) A
Author of method / Contact H. Wold (1966)
Author of description Tobias Girschick
Contact within OT kramer@in.tum.de
Comments (including reviews)

3.3 Feature selection algorithms

3.3.1 Information Gain Attribute Evaluation

InfoGainAttributeEval evaluates the worth of an attribute by measuring the information gain with respect to the class.

$$InfoGain(Class,Attribute) = H(Class) - H(Class \mid Attribute),$$

where H is the information entropy.

InfoGainAttrivuteEval		
Input Instances, feature vectors, class values		
Output Instances, feature vectors, class values		
Input format Weka's ARFF format		
Output format Weka's ARFF format		
User-specified parameters Number of features to select (non-mandatory) Information Gain Threshold (non-mandatory)		
Reporting information		

Attributes ranked by Information Gain
Background (publication date, popularity/level of familiarity, rationale of approach, further comments)
Widely used standard feature selection method, disadvantage: does not take into account feature interaction
Class-blind/class-sensitive feature selection
Class-sensitive feature selection
Type (optimal, greedy, randomized)
Optimal
Filter/wrapper/hybrid approach
Filter
Performance (time/space complexity, running times, memory consumption, ...)
Fast. Each feature is compared against the target variable. As it is a filter approach, the evaluation of feature sets is computationally cheap.
OT availability
Available, e.g., in the Weka open source data mining workbench. Java source code is provided free of charge
Licence /Dependencies
GPL (see OT availability)
Convenience of integration
Webservices: very easy / implemented in Java
Priority (A, B, C)
A
Author of method / Contact
-
Author of description
Tobias Girschick
Contact within OT
kramer@in.tum.de
Comments (including reviews)

3.3.2 FCBF

The FCBF (Fast Correlation-Based Filter) algorithm consists of two stages: the first one is a relevance analysis, aimed at ordering the input variables depending on a relevance score, which is computed as the symmetric uncertainty with respect to the target output. This stage is also used to discard irrelevant variables, which are those whose ranking score is below a predefined threshold. The second stage is a redundancy analysis, aimed at selecting predominant features from the relevant set obtained in the first stage. This selection is an iterative process that removes those variables which form an approximate Markov blanket. The method is described in details in [YUL04].

More information can be found in the following Web page: www.public.asu.edu/~huanliu/FCBF/FCBFsoftware.html

FCBF (Fast Correlation Based Filter)		
Input		
Output		
Input format Weka's ARFF format		
Output format Weka's ARFF format		
User-specified parameters A predefined threshold		
Reporting information The optimal subset of variables		
Background (publication date, popularity/level of familiarity, rationale of approach, further comments) Widely used standard feature selection method, disadvantage: the input variables should be discretized		
Class-blind/class-sensitive feature selection Class-sensitive feature selection		
Type (optimal, greedy, randomized) Optimal		

Filter/wrapper/hybrid approach Filter
Performance (time/space complexity, running times, memory consumption, ...) Fast. FCBF compares only individual features with each other
OT availability Available, e.g., in the Weka open source data mining workbench. Java source code is provided free of charge
Licence /Dependencies GPL (see OT availability)
Convenience of integration Webservices: very easy / implemented in Java
Priority (A, B, C) B
Author of method / Contact Yu and Liu
Author of description Haralambos Sarimveis
Contact within OT hsarimv@central.ntua.gr
Comments (including reviews)

3.3.3 PCA

The Principle Component Analysis (PCA) is mathematically defined as an orthogonal linear transformation that transforms the data to a new coordinate system such that the greatest variance by any projection of the data comes to lie on the first coordinate, the second greatest variance on the second coordinate and so forth. The coordinates are here called principal components.

PCA (Principal component analysis)		
Input		
Output		

Input format Weka's ARFF format
Output format Weka's ARFF format
User-specified parameters - Variance covered - Maximum number of attributes to include in transformation
Reporting information The optimal subset of variables
Background (publication date, popularity/level of familiarity, rationale of approach, further comments) PCA is closely related to factor analysis; synonyms: Karhunen–Loève transform (KLT), Hotelling transform or proper orthogonal decomposition (POD);
Class-blind/class-sensitive feature selection Class-blind
Type (optimal, greedy, randomized) Optimal (PCA is theoretically the optimum transform for a given data in least square terms)
Filter/wrapper/hybrid approach Filter
Performance (time/space complexity, running times, memory consumption, ...) Fast.
OT availability Available, e.g., in the Weka open source data mining workbench. Java source code is provided free of charge
Licence /Dependencies GPL (see OT availability)
Convenience of integration Webservices: very easy / implemented in Java
Priority (A, B, C) B

Author of method / Contact Pearson K. (1901) [PEA01]
Author of description Fabian Buchwald
Contact within OT Kramer@in.tum.de
Comments (including reviews)

3.3.4 Chi Square Feature Evaluation

Feature Selection via chi square (X^2) test is another, very commonly used method [LIU95]. The X^2 method evaluates features individually by measuring their chi-squared statistic with respect to the classes.

Chi Square Feature Evaluation		
Input		
Output		
Input format Weka's ARFF format		
Output format Weka's ARFF format		
User-specified parameters		
Reporting information		
Background (publication date, popularity/level of familiarity, rationale of approach, further comments) Widely used standard feature selection method, disadvantage: does not take into account feature interaction		
Class-blind/class-sensitive feature selection Class-sensitive feature selection		
Type (optimal, greedy, randomized)		

Filter/wrapper/hybrid approach Filter
Performance (time/space complexity, running times, memory consumption, ...) Fast. Evaluation of one feature is linear in number of instances.
OT availability Available, e.g., in the Weka open source data mining workbench
Licence /Dependencies GPL (see OT availability)
Convenience of integration Webservices: very easy / implemented in Java
Priority (A, B, C) B
Author of method / Contact
Author of description Martin Gütlein
Contact within OT guetlein@informatik.uni-freiburg.de
Comments (including reviews)

3.3.5 CFS Feature Set Evaluation

CFS is a correlation-based filter method CFS from [Hal98]. It gives high scores to subsets that include features that are highly correlated to the class attribute but have low correlation to each other Let S be an attribute subset that has k attributes, r_{cf} models the correlation of the attributes to the class attribute, r_{ff} the intercorrelation between attributes.

$$merit_S = k\, r_{cf} / sqrt(k + k(k-1)\, r_{ff})$$

CFS Feature Set Evaluation		
Input		
Output		

Input format Weka's ARFF format
Output format Weka's ARFF format
User-specified parameters
Reporting information
Background (publication date, popularity/level of familiarity, rationale of approach, further comments) Default Feature Set Evaluator in Weka. Advantage: fast filter method that can evaluate sets (instead of single features only)
Class-blind/class-sensitive feature selection Class-sensitive feature selection
Type (optimal, greedy, randomized)
Filter/wrapper/hybrid approach Filter
Performance (time/space complexity, running times, memory consumption, ...) Evaluation of a feature set is quadratic in number of attributes. Compared to a wrapper approach, the evaluation of feature sets is computationally cheap.
OT availability Available, e.g., in the Weka open source data mining workbench
Licence /Dependencies GPL (see OT availability)
Convenience of integration Webservices: very easy / implemented in Java
Priority (A, B, C) B
Author of method / Contact Mark Hall
Author of description

Martin Gütlein
Contact within OT guetlein@informatik.uni-freiburg.de
Comments (including reviews)

3.3.6 Wrapper Feature Set Evaluation

The wrapper approach depends on the classifier that should be used with the resulting attribute subset. Wrapper methods evaluate subsets by running the classifier on the training data, using only the attributes of the subset. The better the classifier performs, usually based on cross-validation, the better is the selected attribute set. One normally uses the classification-accuracy as the score for the subset. Though this technique has a long history in pattern recognition, [JOH94] introduced the term wrapper that is now commonly used.

Wrapper Feature Set Evaluation		
Input		
Output		
Input format Weka's ARFF format (see OT availability)		
Output format Weka's ARFF format (see OT availability)		
User-specified parameters Minimum support		
Reporting information Frequent free trees (SMARTs) with occurrence maps, border elements		
Background (publication date, popularity/level of familiarity, rationale of approach, further comments) Standard feature selection method. Leads to superior results compared to Filter methods. Slow. Resulting feature set is specific to the QSAR model that is used by the wrapper.		
Class-blind/class-sensitive feature selection Class-sensitive feature selection		
Type (optimal, greedy, randomized)		

Filter/wrapper/hybrid approach Wrapper
Performance (time/space complexity, running times, memory consumption, ...) Very slow, as the classifier (QSAR model) hast to be built and applied using only the features in the current set (using internal cross-validation). Performance depends on the particular QSAR model.
OT availability Available, e.g., in the WEKA open source data mining workbench. Has to be (re-) implemented in the OT Framework, if OT QSAR models should be used by the wrapper that are not in WEKA.
Licence /Dependencies GPL (see OT availability)
Convenience of integration Webservices: very easy / implemented in Java (see OT availability)
Priority (A, B, C) C
Author of method / Contact
Author of description Martin Gütlein
Contact within OT guetlein@informatik.uni-freiburg.de
Comments (including reviews)

3.4 Algorithms for the aggregation of results from multiple QSAR models

3.4.1 Consensus models

Consensus models are developed by averaging the predicted values for every compound using many QSAR models, with or without taking into account their respective applicability domains.

More information can be found in reference [ZHU08]

Development of consensus models
Input
Output
Input format Multiple QSAR models developed by different regression or classification algorithms and the associated domains of applicability constitute the input to the method
Output format Plain text
User-specified parameters The user needs to define if the consensus prediction for an external compound is constructed by averaging all the available predicted values from multiple QSAR models or only those that include the compound in their applicability domain. In the second case the user also needs to define the parameter b, which means that the compound should be in the applicability domain of at least b models, in order to consider the consensus prediction valid.
Reporting information A single prediction is provided for each individual molecule
Background (publication date, popularity/level of familiarity, rationale of approach, further comments) It has been found that consensus models afford higher prediction accuracy for the external validation data sets with the highest space coverage as compared to individual constituent models [ZHU08]. However, opposite results have also been reported [HEW07].
Performance (time/space complexity, running times, memory consumption, ...) Fast. The method uses already developed QSAR models.
OT availability Not currently available
Licence /Dependencies None
Convenience of integration Web services: easy since the method will be implemented in Java or C++
Priority (A, B, C)

C
Author of method / Contact A. Tropsha
Author of description Haralambos Sarimveis
Contact within OT hsarimv@central.ntua.gr
Comments (including reviews)

4. Algorithm evaluation and selection of algorithms for the prototype

After giving an overview of the relevant algorithm selection criteria and the algorithms under consideration, we will evaluate the algorithms according to those criteria and our needs for the prototype. The list of algorithms has to be considered an ongoing work, but as the most basic and prominent (Q)SAR algorithms are on the list, it is sufficient for the decisions that have to be made before developing the initial prototypes.

If you look up a definition of prototype in the dictionary you get something similar to "*An original type, form, or instance serving as a basis or standard for later stages.*" What we are looking for is a functional prototype that will evolve into the final OpenTox framework during the subsequent stages of the project. The prototype should have the basic OpenTox functionality including all three categories of algorithms.

During our February virtual conference we decided on some basic points for the restriction of the prototype. Those decisions included not using wrapper feature selection algorithms, and no genetic algorithms and consensus models (algorithms for the aggregation of results from multiple QSAR models) for reasons of convenience in this early project stage. It was decided to be not too restrictive in the selection of algorithms, but to introduce a prioritization (A, B, C) that roughly corresponds to the stage at which the algorithm and implementation will be integrated into the framework. We plan to include the whole range of algorithms, but algorithms provided by partners and free (open source) software that is operating system independent, is clearly preferred. We divide the rest of this section into three parts, one for each category of algorithms.

4.1 Descriptor calculation algorithms

Basically this section provides two different types of molecular descriptors, namely physico-chemical and (sub)structural descriptors. We do want to include at least one algorithm of each type in the initial prototype.

In the group of structural and sub-structural descriptors we decided to include only FTM and MakeMNA in the initial prototype and include FMiner and gSpan' subsequently. The reason for this choice is that first of all FMiner, gSpan' and FTM are similar approaches and FTM is the only algorithm of the named four that is compiled for Windows and Linux operating systems. The fact that, e.g. gSpan' is performing better than FTM is an issue that will be more important in later stages of the development.

In the group of physico-chemical descriptors we will include open babel with highest priority as some other proposed software packages have dependencies on it. We plan to integrate the Chemistry Development Toolkit (CDK), JOELib and MakeQNA in the second stage of the prototype and MOPAC 7.1 and AMBIT (which depends on MOPAC) in the second or third stage of the framework.

4.2 Classification and regression algorithms

As in the previous section we have to cover two "classes" of algorithms with the prototype selection: at least one classification and one regression algorithm. A further criterion is to include at least one eager and one lazy learning method. The more sophisticated methods will be integrated later than the basic and more prominent (Q)SAR algorithms.

For the first prototype we plan to integrate MLR as basic regression method, kNN as basic instance based (lazy learning) classification method and J48 decision trees as eager classification algorithm implementations. Those algorithms are available in platform independent Java implementations and therefore very easy to integrate. As PCA and PLS also are very prominent in the (Q)SAR community we will try to integrate them in the late first or early second stage of the prototype. Lazar and iSAR constitute two similar lazy learning approaches so we will

include lazar in the second and iSAR in the third phase to keep the algorithm selection homogeneous. But we have to keep in mind that both algorithms are up to now only available for Linux operating systems. Further popular methods are SVM and FuzzyMeans (neuronal net algorithm) which will be incorporated in stage two. Maybe FuzzyMeans will be shifted to stage three because of its dependency on Matlab, which is not open source. ToxTree will be added in the second stage as it is easy to integrate and operating system independent.

SMIREP/SMIPPER is dependent on the OpenBabel package which will be available after the first prototype stage and its compilation is operating system dependent (python) so we will include it in the second phase. Furthermore we will try to also include MakeSCR in the second prototype stage, keeping in mind that at this point only a Windows version of the implementation exists. As the MaxTox implementation is still under development we plan to add it to the third stage of the framework, just like the more sophisticated methods RUMBLE, Gaussian Processes for Regression and M5P. RUMBLE additionally is at the moment only compiled for Linux operating systems.

Regarding the bias of the algorithms we have algorithms with feature-selection (e.g., MLR, J48) and with instance-selection bias (e.g., kNN) integrated from stage one on.

4.3 Feature selection algorithms

Regarding the variable selection methods, we had to choose among three families of methods: filter methods, wrapper methods and embedded methods [SAE2007]. Filter techniques assess the relevance of features by looking only at the intrinsic properties of the data. In most cases a feature relevance score is calculated, and low-scoring features are removed. The remaining subset of features consists of the descriptors that are used as input to the regression or classification algorithm. The key advantages of filter techniques are that they easily scale to very high-dimensional datasets, they are computationally simple and fast, and they are independent of the classification algorithm. As a result, feature selection needs to be performed only once for a given set of data.

Whereas filter techniques treat the problem of finding a good feature subset independently of the model selection step, wrapper methods embed the model hypothesis search within the feature subset search. In wrapper methods, a search procedure in the space of possible feature subsets is defined, and various subsets of features are generated and evaluated. In the third class of feature selection techniques, termed embedded or hybrid techniques, the search for an optimal subset of features is built into the classifier construction, and can be seen as a search in the combined space of feature subsets and hypotheses. Just like wrapper approaches, embedded approaches are specific to a given learning algorithm. Embedded methods have the advantage that they include the interaction with the classification model, while at the same time being less computationally intensive than wrapper methods. However, compared to filter methods, they are still far more computationally intensive.

Our key selection criterion, besides other criteria such as state of-the-art algorithms, availability from OT members, licence and convenience of integration with the OT framework was the low computational complexity, taken into account that the methods will be used in huge databases as far as chemical compounds and available descriptors are concerned. We decided to select state-of-the-art *filter* methods, because they are of low computational complexity, but most importantly solve the feature selection problem just once for a given set of data. As mentioned above, variable selection algorithms that belong to the two other families depend on the regression and classifications algorithms and need to be executed for each individual regression or classification algorithm. Moreover, they create additional complexities in two popular model validation

methods, namely cross-validation and Y-randomization, because they need to be executed for each fold in the cross-validation method and each random scramble in the Y-randomization method.

As feature selection algorithms are of lower functional priority than the descriptor calculation or classification and regression algorithms we chose only one algorithm for the initial prototype, to ensure full testing possibilities and complete functionality of the prototype. Therefore we choose the Information Gain Attribute Evaluation algorithm arbitrarily for this first prototype stage. The other algorithms will be integrated in the second and third stage.

5. Conclusions

In this document we report on the algorithm selection criteria and algorithm evaluation that has been done in the first phase of the OpenTox project. We provide a list of algorithms with their characteristics regarding the chosen selection criteria. Furthermore we made a prioritization of the algorithms which indicates in which stage of the project we plan to integrate the algorithm into the project framework. A summary of our considerations and the resulting prioritization is given in table 1.

Algorithm category	Priority	Algorithm
Descriptor calculation	A	FTM (TUM) OpenBabel MakeMNA (IBMC)
	B	FMiner (IST) gSpan‘(TUM) MakeQNA (IBMC) JOELib CDK
	C	MOPAC AMBIT
Classification and regression	A	MLR kNN J48 PLS
	B	SVM Lazar (IST) SMIREP/SMIPPER (ALU-FR) ToxTree (IDEA)

		Fuzzy-means (NTUA) MakeSCR (IBMC)
	C	Gaussian Processes for Regression iSAR (TUM) RUMBLE (TUM) M5P MaxTox (SIT-JNU)
Feature selection	A	InfoGainAttributeEval
	B	FCBF PCA Chi Square Feature Evaluation CFS Feature Set Evaluation
	C	Wrapper Feature Set Evaluation
Algorithms for the aggregation of results from multiple QSAR models	C	Consensus models

Table 1: Prioritization summary

6. References

[AHA91] D. Aha, D. Kibler (1991). Instance-based learning algorithms. *Machine Learning*. 6:37-66.

[BEN07] R. Benigni, C. Bossa, T. Netzeva, A. Rodomonte, and I. Tsakovska (2007) Mechanistic QSAR of aromatic amines: new models for discriminating between mutagens and nonmutagens, and validation of models for carcinogens. Environ mol mutag 48:754-771

[BEN08] R. Benigni, C. Bossa, N. Jeliazkova, T. Netzeva, and A. Worth (2008) The Benigni / Bossa rulebase for mutagenicity and carcinogenicity - a module of Toxtree, *JRC Scientific and Technical Reports,* ecb.jrc.it/documents/QSAR/EUR_23241_EN.pdf

[BRE84] Breiman, L., Friedman, J.H., Olshen, R.A. and Stone, C.J.: *Classification and Regression Trees*. Wadsworth, Belmont CA. (1984)

[BRO73] Bron, C.; Kerbosch, Finding All Cliques of an Undirected Graph. J. Commun. ACM 1973, 16, 575-577.

[CDK] sourceforge.net/projects/cdk

[COR95] Cortes, C., Vapnik, V., (1995). Support-vector networks, *Machine Learning*, 20:273-297.

[CRA78] Cramer G. M., R. A. Ford, R. L. Hall (1978), Estimation of Toxic Hazard - Decision Tree Approach, *J. Cosmet. Toxicol.*, Vol.16, pp. 255 -276, Pergamon Press

[FIL99] Filimonov D., Poroikov V., Borodina Yu., Gloriozova T. Chemical Similarity Assessment through multilevel neighborhoods of atoms: definition and comparison with the other descriptors. *J. Chem. Inf. Comput. Sci.* (1999), Vol. 39, P. 666-670.

[FIL05] Filimonov, D.; Lagunin, A.; Poroikov. V. In *Proceedings of the 15th European Symposium on Structure-Activity Relationships (QSAR) and Molecular Modelling*; Aki, E.; Yalcin, I., Ed.; CADD&D SOCIETY IN TURKEY: Ankara, 2005; pp 98-99.

[GER05] Ingrid Gerner, Manfred Liebsch & Horst Spielmann (2005) Assessment of the eye irritating properties of chemicals by applying alternatives to the Draize rabbit eye test: the use of QSARs and in vitro tests for the classification of eye irritation, *Alternatives to Laboratory Animals*, 33, pp. 215-237

[HAN95] C. Hansch, A. Leo and D. Hoeckman, (1995) Exploring QSAR, hydrophobic, electronic and steric constants, *ACS,* Washington DC

[HAN02] X. Yan, J. Han: gSpan: Graph-based substructure pattern mining. *Proc. 2nd IEEE Int. Conf. on Data Mining (ICDM 2002, Maebashi, Japan)*, 721-724. IEEE Press, Piscataway, NJ, USA 2002.

[HEL06] C. Helma. Lazy structure-activity relationships (lazar) for the prediction of rodent carcinogenicity and Salmonella mutagenicity. *Molecular Diversity*, 10:147-158, 2006

[HEW07] Hewitt, M., Cronin, M.T.D, Madden, J. C., Rowe, P. H., Johnson, C., Obi, A., and Enoch, S. J. (2007). Consensus QSAR Models: Do the Benefits Outweigh the Complexity?, *Journal of Chemical Information and Modeling* 48(4):766-784.

[JOELIB] www.ra.cs.uni-tuebingen.de/software/joelib/index.html

[JOETUT] www.ra.cs.uni-tuebingen.de/software/joelib/tutorial/JOELibTutorial.pdf

[JOH94] George H. John, Ron Kohavi and Karl Pfleger, Irrelevant Features and the Subset Selection Problem. *ICML*, 1994

[JK05] Jahn, K. and Kramer, S. (2005). Optimizing gSpan for Molecular Datasets In: *Proceedings of the Third International Workshop on Mining Graphs, Trees and Sequences (MGTS-2005).*

[JWR02] John W Raymond ,Eleanor J. G. Ardiner, Peter Willet, RASCAL: Calculation of Graph Similarity using Maximum Common Edge Subgraphs, *The Computer Journal*, 45 (6). pp. 631-644. ISSN 1460-2067

[LAG07] Lagunin, A.; Zakharov, A.; Filimonov, D.; Poroikov, V. A new approach to QSAR modelling of acute toxicity. *SAR and QSAR in Environmental Research* 2007, 18, 285-298.

[LEE08] Adam C. Lee, Jing-yu Yu, Gordon M. Crippen, (2008) pKa Prediction of Monoprotic Small Molecules the SMARTS Way, *Journal of Chemical Information and Modeling*, 48 (10), 2042-2053

[LIU95] Liu, H. and Setiono, R., Chi2: Feature selection and discretization of numeric attributes, *Proc. IEEE 7th International Conference on Tools with Artificial Intelligence*, 338-391, 1995

[MAU08] A. Maunz, C. Helma. Prediction of chemical toxicity with local support vector regression and activity-specific kernels, *SAR and QSAR in Environmental Research*, Vol. 19, No. 5-6. (July 2008), pp. 413-431.

[MEL06] Melagraki, G. Afantitis A., Sarimveis, H., Iglessi-Markopoulou, O., Alexandridis, A (2006). A novel RBF neural network training methodology to predict toxicity to Vibrio fischeri, *Molecular Diversity*, 10(2): 213-221.

[MIT97] Tom Mitchell, *Machine Learning*, McGraw Hill, 1997.

[PAT 08] Patlewicz G, Jeliazkova N, Safford RJ, Worth AP, Aleksiev B. (2008) An evaluation of the implementation of the Cramer classification scheme in the Toxtree software. *SAR QSAR Environ Res.*19(5-6):495-524.

[PEA01] Pearson, K., On Lines and Planes of Closest Fit to Systems of Points in Space, *Philosophical Magazine*, 2 (6): 559-572.

[PRA06] Prakash, O.; Ghosh, I, Developing an Antituberculosis Compounds Database and Data Mining in the Search of a Motif Responsible for the Activity of a Diverse Class of Antituberculosis Agents, *J. Chem. Inf. Model.*, 2006, 46, 17-23

[QUI92] Ross J. Quinlan: Learning with Continuous Classes. In: *5th Australian Joint Conference on Artificial Intelligence,* Singapore, 343-348, 1992.

[QUI93] Ross Quinlan (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, San Mateo, CA.

[RAS05] Rasmussen, C. E.; Williams, C. K. I. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*; The MIT Press: 2005.

[RUE04] Rückert, U and Kramer, S., Frequent Free Tree Discovery in Graph Data, in: *SAC '04: Proceedings of the 2004 ACM Symposium on Applied Computing*, pp. 564-570 (New York, NY, USA: ACM Press, 2004).

[RUE08] Rückert, U and Kramer, S (2008). Margin-Based First-Order Rule Learning, *Machine Learning*, 70(2-3):189-206.

[SAR02] Sarimveis, H., Alexandridis, A., Tsekouras, G and Bafas, G, (2002). A fast and efficient algorithm for training radial basis function neural networks based on a fuzzy partition of the input space, *Industrial & Engineering Chemistry Research*, 41:751-759.

[SAR06] Sarimveis, H., Doganis, P., Alexandridis, A (2006). A classification technique based on radial basis function neural networks, *Advances in Engineering Software*, 37(4):218-221.

[SOM07] Sommer, S., Kramer, S. (2007). Three Data Mining Techniques To Improve Lazy Structure-Activity Relationships for Non-Congeneric Compounds, *Journal of Chemical Information and Modeling* 47(6):2035-2043.

[STE03] Steinbeck, C.; Han, Y. Q.; Kuhn, S.; Horlacher, O.; Luttmann, E.; Willighagen, E.L.(2003) The Chemistry Development Kit (CDK): An open-source Java library for chemo- and bioinformatics. *Journal of Chemical Information and Computer Sciences*, 43, 493-500. doi:10.1021/ci025584y

[STE06] Steinbeck, C.; Hoppe, C.; Kuhn, S.; Floris, M.; Guha, R.; Willighagen, E.L. (2006) Recent developments of the chemistry development kit (CDK) - an open-source java library for chemo- and bioinformatics. *Current pharmaceutical design*, 12, 2111-20. doi:10.2174/138161206777585274

[TOD00] Roberto Todeschini, Viviana Consonni, Raimund Mannhold, Hugo Kubinyi, Hendrik Timmerman, *Handbook of Molecular descriptors*, 2000

[TRO03] Tropsha, A., Gramatica, P. and Gombar, V. K. (2003). The Importance of Being Earnest: Validation is the Absolute Essential for Successful Application and Interpretation of QSPR Models. *Quant. Struct.-Act. Relat. Comb. Sci.*, *22*:69-77.

[VAP98] Vapnik, V., *Statistical learning theory* (Wiley-Interscience, 1998).

[VER92] Verhaar HJM, van Leeuwen CJ and Hermens JLM (1992) Classifying environmental pollutants. Structure-activity relationships for prediction of aquatic toxicity. *Chemosphere* 25, 471- 491

[WAL05] John D. Walker, Ingrid Gerner, Etje Hulzebos, Kerstin Schlegel, The Skin Irritation Corrosion Rules Estimation Tool (SICRET), *QSAR Comb. Sci. 2005*, 24, pp378-384

[WAN97] Wang ,Y., Witten, I. H.: Induction of model trees for predicting continuous classes. In: *Poster papers of the 9th European Conference on Machine Learning*, 1997.

[WIT99] Witten, I.H. Frank, E., *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations* (Morgan Kaufmann, 1999).

[YUL04] Yu, L., Liu, H. (2004). Efficient Feature Selection via Analysis of Relevance and Redundancy, *Journal of Chemical Machine Learning Research* 5:1205-1224.

[ZHU08] Zhu, H., Tropsha, A., Fourches, D., Varnek, A., Papa, E., Gramatical, P., Öberg, T., Dao, P., Cherkasov, A., Tetko, I.V. (2008). Combinatorial QSAR modeling of chemical toxicants tested against Tetrahymena pyriformis, Journal of Chemical Information and Modeling 48(4):766-784.