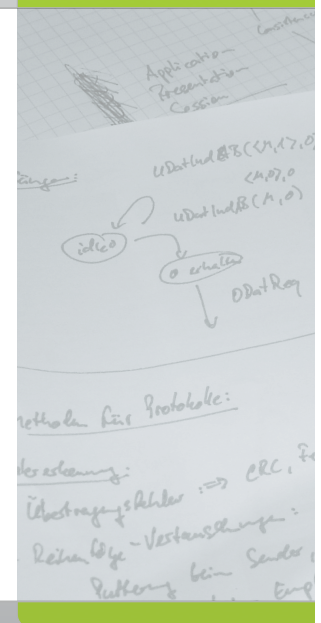


Maschinelles Lernen mit RapidMiner



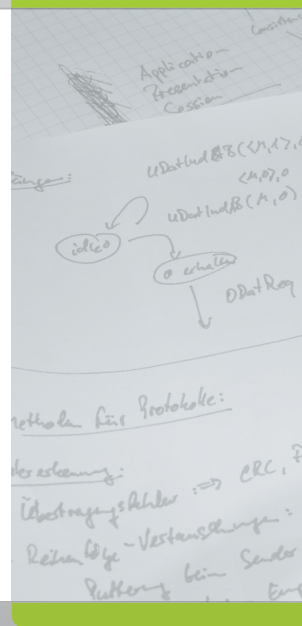
Inhalt

- Motivation / Ziel
- Konzept von RapidMiner
- Beispiele
- Details, Implementierung
 - ExampleSet, Attribute
 - Übungen



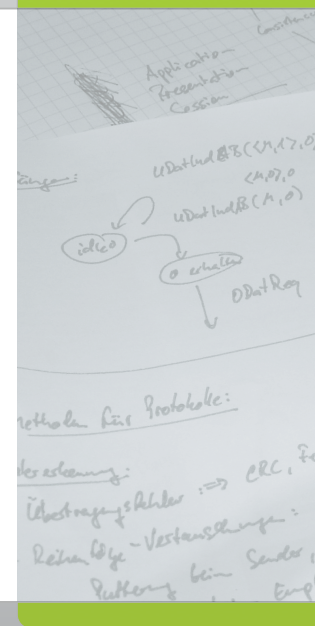
Motivation

- Abstrakte maschinelle Lernaufgaben aus der Vorlesung
 - Clustering
 - Klassifikation/Regression
 - Funktionslernen
- Darüber hinaus
 - Merkmalsextraktion, -generierung, -selektion
 - Zeitreihenanalyse, Text-Mining



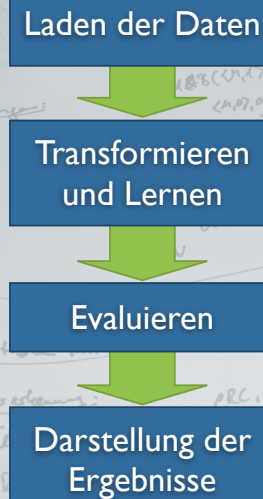
Motivation

- In der Praxis leider häufig:
 - Jeder implementiert „seinen“ Algorithmus für seine Anwendung neu
 - Jede Anwendung setzt auf unterschiedliche Datenformate
 - Modell-/Verfahrensselektion durch Ausprobieren auf den eigenen Daten



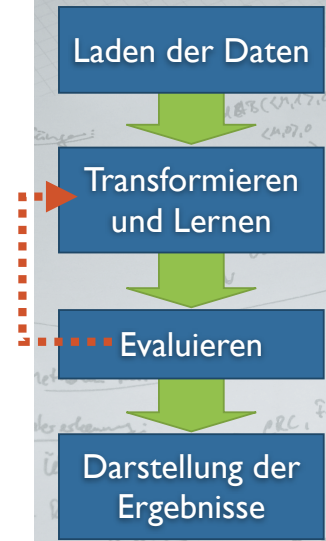
Motivation

- Dabei häufig viele Schritte gleich
- ML-Experiment besteht aus unterschiedlichen Teilaufgaben
- Datenvorverarbeitung spielt wichtige Rolle für Lernaufgaben/-algorithmen
- Evaluationen erfordern flexible Experimentierumgebung
- Ggf. periodische Wiederholungen von Analysen notwendig



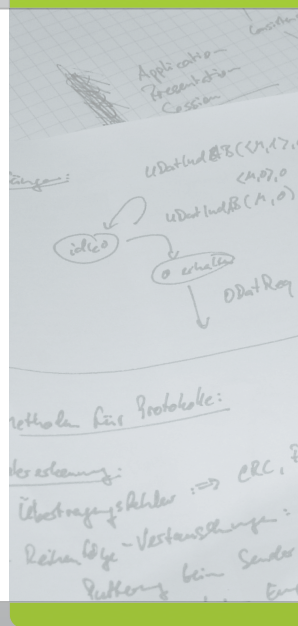
Motivation

- Dabei häufig viele Schritte gleich
- ML-Experiment besteht aus unterschiedlichen Teilaufgaben
- Datenvorverarbeitung spielt wichtige Rolle für Lernaufgaben/-algorithmen
- Evaluationen erfordern flexible Experimentierumgebung
- Ggf. periodische Wiederholungen von Analysen notwendig



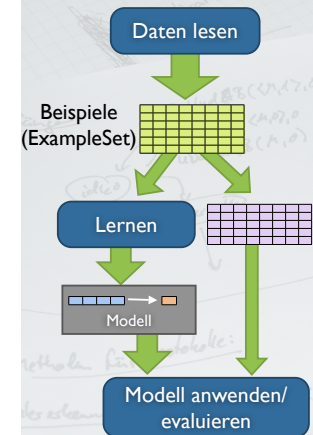
Ziele (RapidMiner)

- Abstraktion der ML-Experimente
- Einfache, wiederverwendbare Beschreibung von Experimenten
- Austauschbarkeit von Lern-Verfahren, insbesondere:
 - Durchführen von Vergleichen unterschiedlicher Verfahren
- Kombination/Verschachtelung von Verfahren



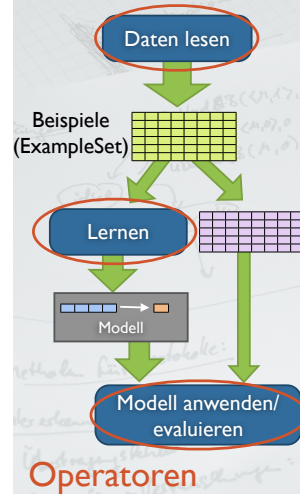
RapidMiner

- Modellierung von ML-Experimenten als Abfolge von Operatoren (Ketten)
- Verschachtelung von Operatoren
- Transparente/effiziente Datenhaltung
- Leichte Erweiterbarkeit durch Plugins
- GUI-Modus/Batch-Modus
- Einbindung externer Programme (z.B. Weka, SVM-Implementierungen)



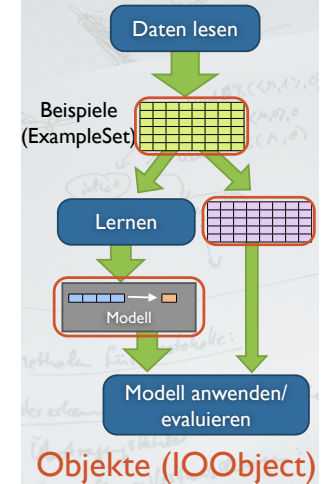
RapidMiner

- Modellierung von ML-Experimenten als Abfolge von Operatoren (Ketten)
- Verschachtelung von Operatoren
- Transparente/effiziente Datenhaltung
- Leichte Erweiterbarkeit durch Plugins
- GUI-Modus/Batch-Modus
- Einbindung externer Programme (z.B. Weka, SVM-Implementierungen)



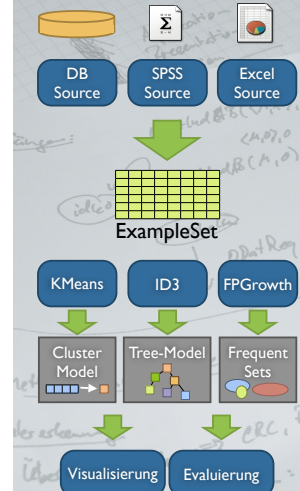
RapidMiner

- Modellierung von ML-Experimenten als Abfolge von Operatoren (Ketten)
- Verschachtelung von Operatoren
- Transparente/effiziente Datenhaltung
- Leichte Erweiterbarkeit durch Plugins
- GUI-Modus/Batch-Modus
- Einbindung externer Programme (z.B. Weka, SVM-Implementierungen)



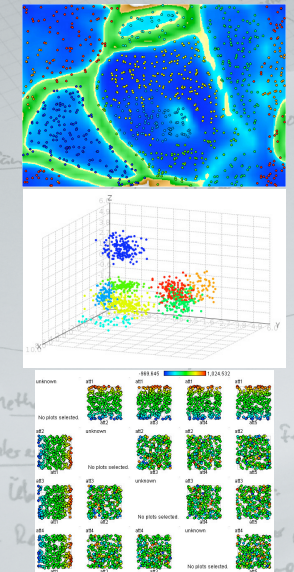
Integrierte Operatoren

- Operatoren zur Ein-/Ausgabe
- Datenvorverarbeitung
- Zahlreiche Lernverfahren (Weka-Lerner, Clustering, ...)
- Performanzbewertung von Lernverfahren
- Verwaltung/Ausgabe von Lernergebnissen



Information

- Open-Source (GPL-Lizenz)
- Erfolgreiche Anwendung auf unterschiedliche Lernaufgaben
- Weltweite Verbreitung (Anwender / Wissenschaftler in über 30 Ländern)
- Dokumentation/Download/uvvm. unter <http://rapid-i.com>



Aufbau/
Ablauf

Aufbau/
Ablauf

Parameter

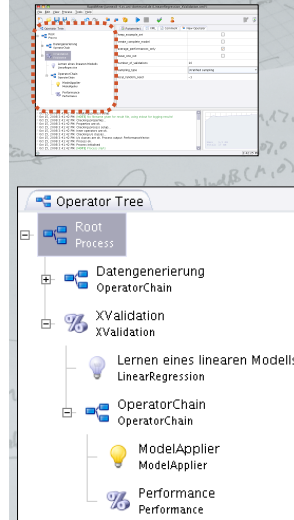
Aufbau/
Ablauf

Parameter

Logfenster

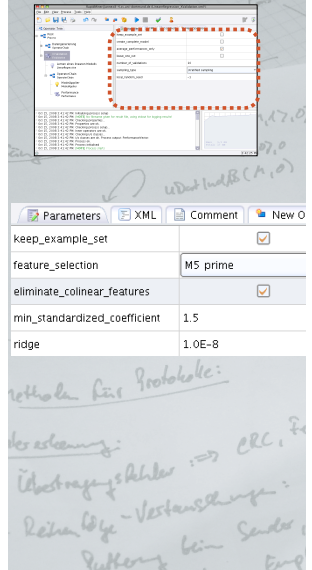
Operatorbaum

- Operatorbaum ist die zentrale Darstellung eines Experimentes
- Operatoren sind in einer Baumstruktur aufgebaut
- Einige Operatoren erlauben innere Operatoren
- Abarbeitung der Operatoren in DFS Reihenfolge



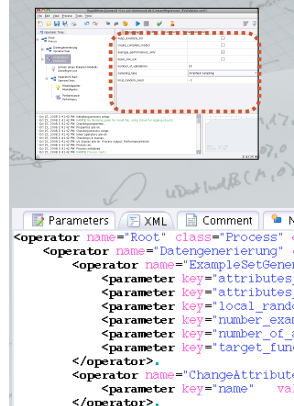
Parameter-Ansicht

- Viele Operatoren benötigen eine Reihe von zusätzlichen Parametern
- Die Parameter-Ansicht zeigt für den aktuell ausgewählten Operator die verfügbaren Parameter an
- Einige Parameter sind nur im „Experten-Modus“ sichtbar



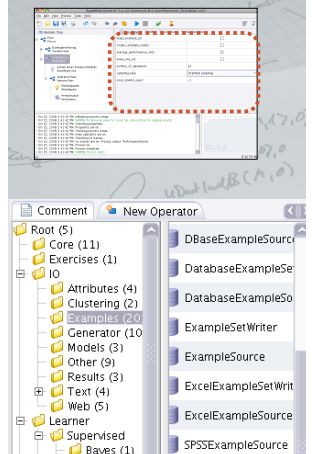
XML-Sicht

- Jedes Experiment ist in XML beschrieben
- Über den XML-Reiter kann das Experiment in XML angezeigt und verändert werden



Operator-Menü

- Eine der Stärken von RapidMiner ist die Vielzahl unterschiedlicher Operatoren
- Der Reiter „New Operator“ erlaubt das Hinzufügen neuer Operatoren zum Experiment per DnD
- **Hinweis:** Über das sehr praktische Suchfeld lassen sich Operatoren schneller finden!

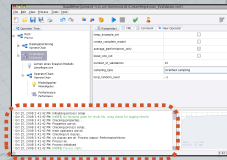


Log-Fenster

- Im Log-Fenster werden allerhand Informationen angezeigt:
 - Über den gerade laufenden Operator
 - Ausgaben durch den Operator

```

P Oct 15, 2008 11:04:24 PM: Checking properties...
P Oct 15, 2008 11:04:24 PM: Properties are ok.
P Oct 15, 2008 11:04:24 PM: Checking process setup...
P Oct 15, 2008 11:04:24 PM: Inner operators are ok.
P Oct 15, 2008 11:04:24 PM: Checking i/o classes...
P Oct 15, 2008 11:04:24 PM: i/o classes are ok. Process output: PerformanceVector.
P Oct 15, 2008 11:04:24 PM: Process ok.
P Oct 15, 2008 11:04:24 PM: Process initialised
P Oct 15, 2008 11:04:24 PM: [NOTE] Process starts
P Oct 15, 2008 11:04:24 PM: Process:
  Root[0] (Process)
  +- Datengenerierung[0] (OperatorChain)
  | +- ExampleSetGenerator[0] (ExampleSetGenerator)
  | +- ChangeAttributeRole[0] (ChangeAttributeRole)
  | +- MergeNominalValues[0] (MergeNominalValues)
    
```

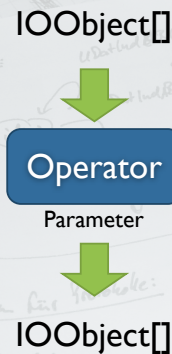


RapidMiner

- Kurze Demo

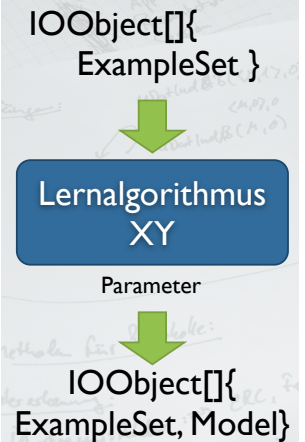
Details

- **Operatoren** erhalten `IObject`s und liefern ggf. mehrere `IObject`s zurück
- Operatoren erweitern die Klasse **Operator**
- `IObject` ist eine abstrakte Klasse, von der z.B. `ExampleSet`, `Model`, usw. erben



Details

- **Operatoren** erhalten `IObject`s und liefern ggf. mehrere `IObject`s zurück
- Operatoren erweitern die Klasse **Operator**
- `IObject` ist eine abstrakte Klasse, von der z.B. `ExampleSet`, `Model`, usw. erben
- Beispiel für einen Lernalgorithmus:
 - Operator, der ein `ExampleSet` Objekt erhält und ein `Model` Objekt erzeugt



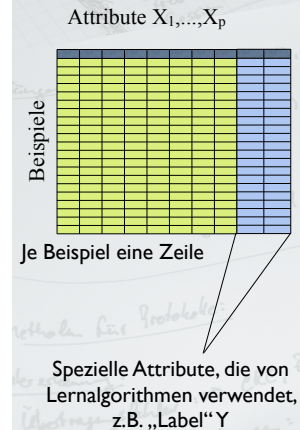
Details

- Objekte, die zwischen Operatoren ausgetauscht werden:
 - **ExampleSet** (eine Menge von Daten)
 - **Model** (gelerntes Model)
 - **PerformanceVector** (Menge von Leistungsmaßen)
 - **Merkmalsgewichte**
 - ...



Details

- Die wichtigsten Klassen sind
 - **Example, ExampleSet**
 - **Attribute, Attributes**
- Jedes **Attribut** referenziert eine „Spalte“ bzw. Eigenschaft der Daten
- Jedes **Example** entspricht einer Zeile, also einem Beispiel unserer Daten
- **ExampleSet** kann man sich vorstellen als (N x p)-Matrix



Attribute

- Attribute legen den Wertebereich einer „Spalte“ fest
- Neben einfachen double-Werten sind nominale, Integer oder auch Datums-Attribute möglich
- Die Attribute teilen sich in reguläre und spezielle Attribute auf
 - Regulär: Daten-Werte (in der Vorlesung mit X bezeichnet)
 - Speziell: z.B. das „Label“-Attribut

Temp	Outlook	Play (Label)
85	sunny	no
80	sunny	no
83	overcast	yes
70	rain	yes
68	rain	yes

Att1	Att2	Cluster-ID
0,35	4,87	1
0,29	9,27	2
0,47	2,31	0
0,98	7,53	2
0,56	6,82	1

ExampleSet

- ExampleSet stellt eine Menge von Beispielen über diesen Attributen dar
- Implementiert u.a. **Iterable<Example>** :

```

ExampleSet examples = ...

for( Example ex : examples ){
    ...
}
    
```

ExampleSet

RapidMiner bietet Darstellung von ExampleSets als Tabelle

Data Table | Meta Data View | **Data View** | Plot View

ExampleSet (2000 examples, 1 special attribute, 2

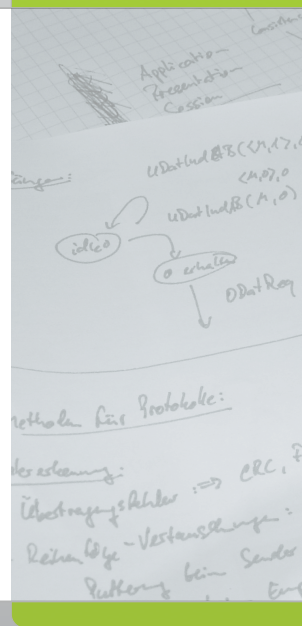
row no.	label	att1	att2
1	class0	0.185	-0.866
2	class1	-0.629	-0.640
3	class1	-0.605	-0.430
4	class0	-0.872	-0.598
5	class0	-0.376	-0.500
6	class0	-0.246	-0.804
7	class1	-1.451	0.064

Example

- Example entspricht einem Beispiel, d.h. einer Menge von Werten für Attribute
- Beispiele werden intern durch eine Menge von double Werten repräsentiert

Example ex = ...

```
for(Attribute a : ex.getAttributes()){
    double value = ex.getValue( a );
}
```



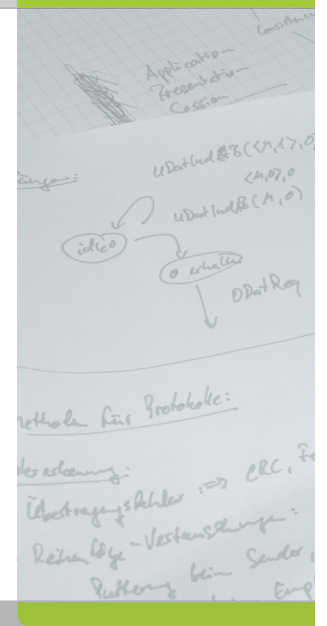
Example

- Example entspricht einem Beispiel, d.h. einer Menge von Werten für Attribute
- Beispiele werden intern durch eine Menge von double Werten repräsentiert

Example ex = ...

```
for(Attribute a : ex.getAttributes()){
    double value = ex.getValue( a );
}
```

Iteration über die **regulären Attribute!**

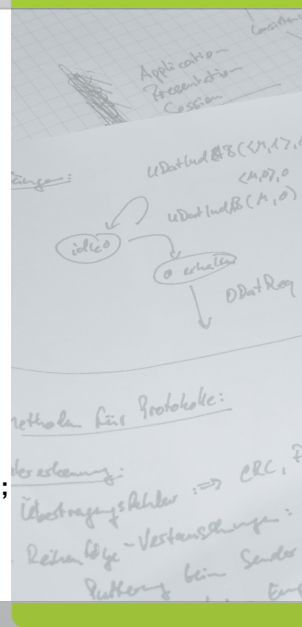


Attribute

- Nominale Attribute enthalten implizit ein Mapping
double -> nominaler Wert

Example ex = ...
Attribute a = ...

```
double value = ex.getValue( a );
if( a.isNominal() ){
    String s = ex.getNominalValue( value );
}
```



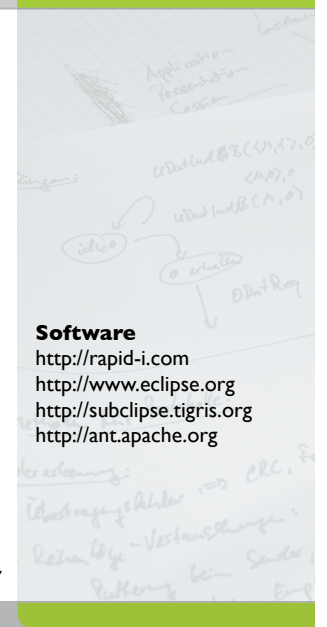
Übungen

- Über *subversion* ist der Lesezugriff auf ein Plugin-Projekt möglich
- **mlv-uebung** ist ein Java-Eclipse Projekt, welches ein eigenständiges RapidMiner-Plugin darstellt
- Das Plugin wird z.B. mit dem Ant-Tool erstellt:
ant plugin

Software

<http://rapid-i.com>
<http://www.eclipse.org>
<http://subclipse.tigris.org>
<http://ant.apache.org>

<https://kissen.cs.uni-dortmund.de/svn/mlv-uebung/>





Übungen

- Das Projekt `mlv-uebung` enthält einige Basis-Klassen (`com.rapidminer.plugin`)
- Die zu implementierenden Übungen sind im Paket `com.rapidminer.exercises`
- Die Lösungen können im Ordner „solutions“ innerhalb des Projektes erstellt werden
- Mit „ant plugin“ läßt sich das Plugin erstellen

<https://kissen.cs.uni-dortmund.de/svn/mlv-uebung/>

Software

<http://rapid-i.com>
<http://www.eclipse.org>
<http://subclipse.tigris.org>
<http://ant.apache.org>



Übungen

- Mit Hilfe von **Ant** läßt sich das Plugin erstellen
- Dazu im Verzeichnis `mlv-uebung` den Befehl „ant plugin“ aufrufen
- Das entstehende Jar-Archiv ist das Plugin mit Eurer Lösung
- Nach Kopie in `lib/plugins/` sind die Operatoren aus dem Plugin beim nächsten Start verfügbar

<https://kissen.cs.uni-dortmund.de/svn/mlv-uebung/>

Software

<http://rapid-i.com>
<http://www.eclipse.org>
<http://subclipse.tigris.org>
<http://ant.apache.org>



Übungen

- Ausgabe der Übungsblätter in der Vorlesung
- Abgabe per Mail bis Dienstag danach
- Implementationsaufgaben:
 - Nach Möglichkeit in **einer** Datei (PDF) + Quellen (.java-Dateien)
 - Ansonsten: RM-Experiment (XML) + Ausgabe
- Gruppenabgaben (bis 4 Studierende) **erwünscht**



Übungen

- Fragen zu den Übungen, Eclipse/ Subversion oder ähnliches?
- Sprechstunde:
Montags 14 Uhr in GB IV, Raum 119
- Per Telefon: 755-6487
- Per Mail:
christian.boeckermann@cs.uni-dortmund.de