Implementation und Evaluation eines Regressionsregellerners

TECHNISCHE UNIVERSITÄT DARMSTADT

Diplomvortrag von Stefan Steger

02. November 2011

Betreuer: Prof. Johannes Fürnkranz
Frederik Janssen

Übersicht

TECHNISCHE UNIVERSITÄT DARMSTADT

- Regel-Lerner
- Separate-and-Conquer
- Regression
- Reduce-and-Conquer
- Evaluation

Regel-Lernen - Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

Wetter	Temperatur	Wind	Spiele Golf?
sonnig	-5	nein	nein
sonnig	25	ja	ja
regnerisch	23	ja	nein
sonnig	17	nein	ja

(Wetter = sonnig, Temperatur = 25, Wind = ja, SpieleGolf = ja)

Wetter	Temperatur	Wind	Spiele Golf?
sonnig	-5	nein	nein
sonnig	25	ja	ja
regnerisch	23	ja	nein
sonnig	17	nein	ja

(Wetter = sonnig ∧ Temperatur > -5) → SpieleGolf = ja

Separate-and-Conquer

TECHNISCHE UNIVERSITÄT DARMSTADT

Separate-and-Conquer Algorithmus:

```
procedure SeparateAndConquer(Examples)
Theory = Ø
while Positive(Examples) ≠ Ø do
   Rule = FindBestRule(Examples)
   Covered = Cover(Rule, Examples)
   if RuleStoppingCriterion(Theory, Rule, Examples) then
      exit while
   end if
   Examples = Examples \ Covered
   Theory = Theory ∪ Rule
end while
Theory = PostProcess(Theory)
return Theory

procedure FindBestRule(Examples)
InitRule = InitializeRule(Examples)
InitVal = EvaluateRule(InitRule)
BestRule = <InitVal, InitRule>
Rules = {BestRule}
while Rules ≠ Ø do
   Candidates = SelectCandidates(Rules, Examples)
   Rules = Rules \ Candidates
   for Candidate ∈ Candidates do
      Refinements = RefineRule(Candidate, Examples)
      for Refinement ∈ Refinements do
         Evaluation = EvaluateRule(Refinement, Examples)
         unless StoppingCriterion(Refinement, Evaluation, Examples)
         NewRule = <Evaluation, Refinement>
         Rules = InsertSort(NewRule, Rules)
         if NewRule > BestRule then
            BestRule = NewRule
         end if
      end for
   end for
   Rules = FilterRules(Rules, Examples)
end while
return BestRule
```

Separate-and-Conquer - Bias

TECHNISCHE UNIVERSITÄT DARMSTADT

Verschiedene Implementationen von Separate-and-Conquer

- AQ, CN2, RIPPER ...

Charakteristika von Regel-Lernern

- Repräsentationssprache
  statisch ↔ dynamisch

- Art der Suche
  Suchstrategie, Suchalgorithmus, Suchheuristik

- Strategie zur Vermeidung von Überbestimmtheit
  Pre-Pruning ↔ Post-Pruning

Regression

TECHNISCHE UNIVERSITÄT DARMSTADT

mean:

$$mean = \frac{1}{n}\sum_{i=1}^{n} y_i$$

median:

$$\text{median} = \begin{cases} y_{(\frac{n+1}{2})}, & \text{n ungerade,} \\ \frac{1}{2}\left(y_{(\frac{n}{2})} + y_{(\frac{n}{2}+1)}\right), & \text{n gerade.} \end{cases}$$

- n: Anzahl der (abgedeckten) Beispiele
- $i \in \{1, \ldots, n\}$: Index für das i-te Beispiel
- y_i: tatsächlicher Wert des i-ten Beispiels
- $\bar{y}_i$: Vorhersage für Beispiel i
- $\acute{y}$: Mittelwert von allen (abgedeckten) Beispielen

mean absolute error:

$$L_{MAE} = \frac{1}{n}\sum_{i=1}^{n} \left|y_i - \bar{y}_i\right|$$

mean of the absolute deviation:

$$L_{MD} = \frac{1}{n}\sum_{i=1}^{n} \left|y_i - median\right|$$

mean squared error:

$$L_{MSE} = \frac{1}{n}\sum_{i=1}^{n} \left(y_i - \bar{y}_i\right)^2$$

Regression

TECHNISCHE UNIVERSITÄT DARMSTADT

relative standard error:

$$L_{RSE} = \frac{\sum_{i=1}^{n} (y_i - \bar{y}_i)^2}{\sum_{i=1}^{n} (y_i - \acute{y})^2}$$

relative absolute error:

$$L_{RAE} = \frac{L_{MAE}}{\frac{1}{n}\sum_{i=1}^{n} |y_i - \acute{y}|}$$

root relative standard error:

$$L_{RRSE} = \sqrt{L_{RSE}}$$

Nicht regelbasierte Ansätze	↔	regelbasierte Ansätze
Lineare Regression		P-Class
Nearest Neighbour		M5Rules
Neuronale Netze		RuleFit
Support Vector Machine		RegENDER
…		SeCoReg
		…

Reduce-and-Conquer (ReCo)

TECHNISCHE UNIVERSITÄT DARMSTADT

- Adaption von SeCo für Regression
- Conquer-Schritt ähnlich wie in Separate-and-Conquer
- Aufteilen der Beispiele in eine Pruning- und Growingmenge
- Im Reduce-Schritt wird der Klassenwert der Beispiele verändert
- es wird eine unsortierte Regelliste verwendet
- zur Vorhersage werden alle das Beispiel abdeckenden Regeln verwendet
- die Vorhersage der Regeln wird addiert
- Regeln werden immer nur im Kontext aller Regeln betrachtet

Reduce-and-Conquer

TECHNISCHE UNIVERSITÄT DARMSTADT

procedure REDUCEANDCONQUER*(Examples)*
GrowingSet = SPLITEXAMPLES*(Examples)*
PruningSet = Examples \ GrowingSet
DefRule = INITIALIZERULE*(Examples)*
REDUCEEXAMPLEVALUES*(*COVER*(DefRule, GrowingSet, PruningSet))*
DefVal = EVALUATERULE*(DefRule, PruningSet)*
DefRule = <DefVal, DefRule>
Theory = {DefRule}
loop
    RefinementRules = FINDREFINEMENTRULES*(GrowingSet)*
    BestRule = FINDBESTREFINEMENTRULE*(RefinementRules, PruningSet)*
    if RULESTOPPINGCRITERION*(BestRule, PruningSet)* **then**
        exit loop
    end if
    Theory = Theory ∪ *BestRule*
    REDUCEEXAMPLEVALUES*(*COVER*(BestRule, GrowingSet, PruningSet))*
end loop
return *Theory*

Reduce-and-Conquer

TECHNISCHE UNIVERSITÄT DARMSTADT

```
procedure FindRefinementRules(GrowingSet)
DefVal = EvaluateRule(DefRule, GrowingSet)
DefRule = <DefVal, DefRule>
Rules = {DefRule}
Candidate = DefRule
BestRule = DefRule
loop
    Refinements = RefineRule(Candidate, GrowingSet)
    for Refinement ϵ Refinements do
        Evaluation = EvaluateRule(Refinement, GrowingSet)
        NewRule = <Evaluation, Refinement>
        if NewRule > BestRule then
            BestRule = NewRule
    end for
    if StoppingCriterion(BestRule, GrowingSet ) then
        exit loop
    Rules = Rules ∪ BestRule
    Candidate = BestRule
end loop
return Rules
```

Splitpoints

TECHNISCHE UNIVERSITÄT DARMSTADT

Beispiel Splitpoints:

Wert	1		2	2		4		5		7	7		8
Splitpoint		1,5			3		4,5		6			7,5	

Problem: Berechnung der Splitpoints sehr aufwendig

- für jedes numerische Attribut müssen alle Splitpoints berechnet werden
- im schlechtesten Fall werden *n-1* Splittpoints für *n* Attributwerte berechnet
- und das für jede mögliche Verfeinerung!

Lösung: Berechne nicht alle Splitpoints

- es werden äquidistante Stichproben genommen ($\sqrt{m}$, m = Anzahl Splitpoints)
- eine obere und untere Schranke werden in die Nähe der besten Stichprobe gesetzt
- der Abstand der Stichproben wird verringert, beste wird bestimmt...
- statt 1000 Splitpoints werden nur 49 Splitpoints evaluiert → ~ 20x schneller!

Testkonfiguration

TECHNISCHE UNIVERSITÄT DARMSTADT

- 21 möglichst unterschiedliche Datensätze
- RRSE wird für Evaluation verwendet (Unabhängigkeit von Datenset)
- 10-fach Cross-Validation
- erst alle Konfigurationen von ReCo miteinander vergleichen
- dann die beste Konfiguration mit anderen Regressionslernern vergleichen
- es wird Rankingverfahren mit anschließende Friedman/Nemenyi Test verwendet
  - Friedman Statistik beschreibt, ob es signifikante Unterschiede der Algorithmen gibt
  - Nemenyi Test stellt fest welche Algorithmen signifikant unterschiedlich sind

Konfigurationen von ReCo

TECHNISCHE UNIVERSITÄT DARMSTADT

Verschiedene Konfigurationen wurden getestet:

- Feste Regelanzahl
- Abbruchkriterium mit Vorschau
- Aufteilung der Growing- und Pruningmenge
- Mindestabdeckung
- Schätzer Median / MD

Feste Regelanzahl

TECHNISCHE UNIVERSITÄT DARMSTADT

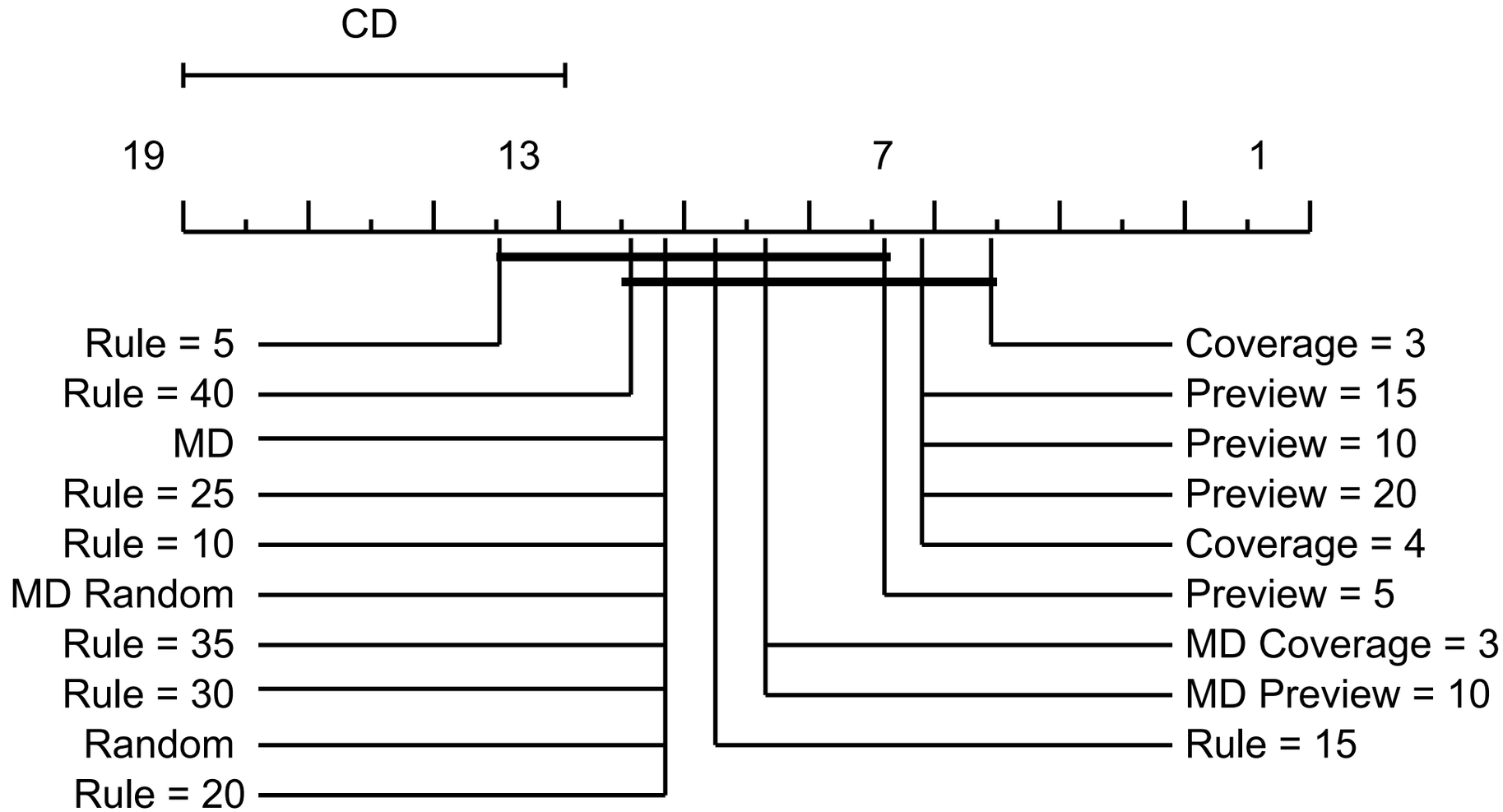
R_L =	5	10	15	20	25	30	35	40	beste K_F
auto93	99	100,3	99,5	99,8	99,8	99,8	99,9	100,4	5
auto-horse	60,3	58,1	56	55,4	55,2	54,7	54,5	54,1	40
auto-mpg	60,2	50,8	48,1	48	48	46,8	46,8	46,3	40
auto-price	51,9	47,4	45,6	47	46,6	46,6	46,7	47,1	15
cloud	71,3	77,9	77,4	77,7	77,5	76,9	76,4	76,3	5
compressive	66,3	54,9	52,4	49	47,3	46,2	45,2	44,3	40
concrete-slump	84,9	65,8	67,9	66,9	68,1	68,8	68,9	69,1	10
cpu	53	50,8	52	51,6	50,6	50,4	50,7	50,7	30
delta-elevators	70,2	66,2	64,6	63,8	63,5	63,3	63,2	63	40
diabetes	103,3	96	101	103,6	105,6	106,7	106,3	106,5	10
echo-month	86	98,5	105,6	110	107,9	108,6	108,9	110,1	5
housing	61,3	55,2	53,6	54,5	54,1	53,9	52,7	53,5	35
machine	42,4	38,5	39,8	41,6	41,1	40,4	41,4	42,1	10
meta	117,4	125,5	133,7	136,4	141,3	141,6	142,1	142,7	5
pyrim	91,9	81,9	81,8	80,8	84	82,3	82,3	82,1	20
r-wpbc	106,8	117,5	123,7	125,8	128,4	129,9	130,3	131,1	5
stock	37,7	29,8	27	26	25,9	25,9	25,5	25,1	40
strike	95,3	96,4	99,7	100,4	103,3	104,1	105,7	107	5
triazines	98,9	93,8	91,6	92,3	92,3	93,9	94,5	94	15
veteran	115,7	122,2	121,4	120,6	121,2	121	122,5	122,6	5
winequality-red	86,7	85,2	85,6	85,3	85,4	85,7	85,9	86,3	10
Durchschnitt	79,1	76,8	77,5	77,9	78,4	78,5	78,6	78,8	10

Ergebnisse ReCo

TECHNISCHE UNIVERSITÄT DARMSTADT

Datenmenge	RRSE	Rang
Cv = 3	72,81	6,26
Pv = 15	73,24	7,21
Pv = 10	73,46	7,26
Pv = 20	73,21	7,29
Cv = 4	73,20	7,31
Pv = 5	73,74	7,86
MD & Cv=3	72,97	9,76
MD & Pv=10	72,90	9,86
RL = 15	77,53	10,52
RD	77,88	11,24
RL = 20	77,91	11,24
RL = 30	78,46	11,29
MD & RD	75,67	11,38
RL = 35	78,59	11,38
MD	73,67	11,43
RL = 10	76,79	11,43
RL = 25	78,43	11,43
RL = 40	78,78	11,86
RL = 5	79,06	13,95

Nemenyi-Test mit den verschiedenen Konfigurationen von ReCo für p = 0,05

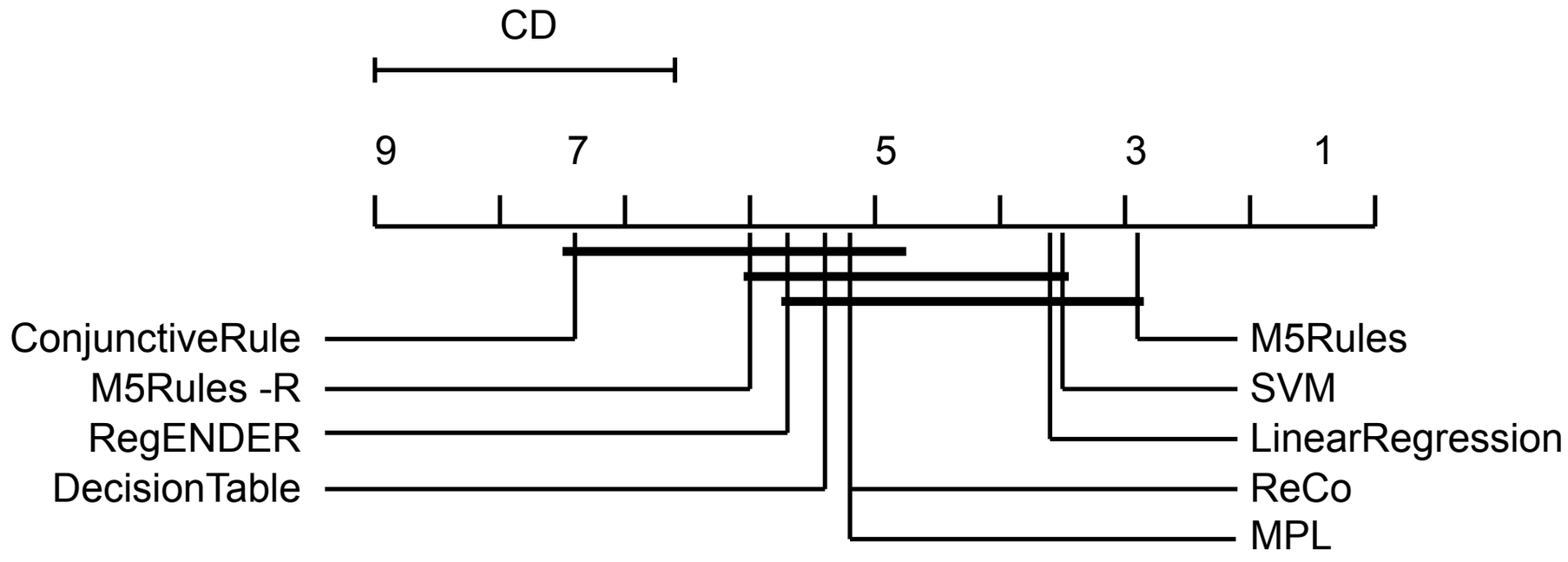


Vergleich mit anderen Lernern

TECHNISCHE UNIVERSITÄT DARMSTADT

M5R	66,66	2,90
SVM	63,35	3,48
LR	70,36	3,62
ReCo	72,81	5,24
MPL	78,96	5,24
DT	75,03	5,38
RegE	80,22	5,67
M5R-R	76,63	6,05
CR	85,50	7,43

Nemenyi-Test der verschiedenen Regressionslerner für p = 0,05



Runtime

TECHNISCHE UNIVERSITÄT DARMSTADT

Algorithmus	Zeit in min
CR	0:01
LR	0:02
RegE	0:20
M5R	0:49
DT	1:12
M5R-R	2:19
ReCo	3:37
MPL	6:24
SVM	10:04

Zusammenfassung / Ausblick

TECHNISCHE UNIVERSITÄT DARMSTADT

- Reduce-Strategie erstmals implementiert
- verschiedene Konfigurationen getestet
- Reduce-and-Conquer ähnlich gut wie andere Regressionsregellerner

Offene Punkte

- wie verhält sich ReCo mit stark verrauschten Daten?
- Reduce-and-Conquer auf Schnelligkeit optimieren
- zusätzliche Konfigurationen testen

  andere Repräsentationssprachen, Suchstrategien, Overfitting