Search Algorithms for Rule Learners

Finding a short and accurate decision rule in disjunctive normal form by exhaustive search

TECHNISCHE UNIVERSITÄT DARMSTADT

Gliederung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Gliederung
- Ansatz und Ziel
- Definitionen
- Induktionsprozess
- EXPLORE Algorithmus
  - Initialisierung
  - Regelerstellung
- Fazit
  - Evaluation
  - Ausblick

Ansatz und Ziel

TECHNISCHE UNIVERSITÄT DARMSTADT

- Probleme bisheriger Ansätze:
  - Abgedeckte Beispiele werden aus Trainingsdaten entfernt

  ⇒ Mit steigendem Fortschritt weniger Trainingsdaten verfügbar
  - Einsatz von Heuristiken um Suchraum einzuschränken

  ⇒ Beste Regel wird eventuell nicht gefunden
  - Generierte Regeln sind für Menschen zu komplex

  ⇒ Nachvollziehbarkeit ist aber je nach Domäne sehr wichtig
  - Optimierung nur nach bestimmtem Maß möglich

  ⇒ Zu optimierende Maße sind domänenspezifisch
- EXPLORE (Exhaustive Procedure for LOgic-Rule Extraction):
  - Finden der besten Regel, bezogen auf ein gewähltes Maß…
  - in DNF (Disjunktiver Normalform) mit geringer Komplexität …
  - welche die Mindestanforderungen anderer Maße erfüllt

Ansatz und Ziel

TECHNISCHE UNIVERSITÄT DARMSTADT

- Probleme bisheriger Ansätze:
  - Abgedeckte Beispiele werden aus Trainingsdaten entfernt

  ⇒ Mit steigendem Fortschritt weniger Trainingsdaten verfügbar
  - Einsatz von Heuristiken um Suchraum einzuschränken

  ⇒ Beste Regel wird eventuell nicht gefunden
  - Generierte Regeln sind für Menschen zu komplex

  ⇒ Nachvollziehbarkeit ist aber je nach Domäne sehr wichtig
  - Optimierung nur nach bestimmtem Maß möglich

  ⇒ Zu optimierende Maße sind domänenspezifisch
- EXPLORE (Exhaustive Procedure for LOgic-Rule Extraction):
  - Finden der besten Regel, bezogen auf ein gewähltes Maß…
  - in DNF (Disjunktiver Normalform) mit geringer Komplexität …
  - welche die Mindestanforderungen anderer Maße erfüllt

Ansatz und Ziel

TECHNISCHE UNIVERSITÄT DARMSTADT

- Probleme bisheriger Ansätze:
  - Abgedeckte Beispiele werden aus Trainingsdaten entfernt

  ⇒ Mit steigendem Fortschritt weniger Trainingsdaten verfügbar
  - Einsatz von Heuristiken um Suchraum einzuschränken

  ⇒ Beste Regel wird eventuell nicht gefunden
  - Generierte Regeln sind für Menschen zu komplex

  ⇒ Nachvollziehbarkeit ist aber je nach Domäne sehr wichtig
  - Optimierung nur nach bestimmtem Maß möglich

  ⇒ Zu optimierende Maße sind domänenspezifisch
- EXPLORE (Exhaustive Procedure for LOgic-Rule Extraction):
  - Finden der besten Regel, bezogen auf ein gewähltes Maß…
  - in DNF (Disjunktiver Normalform) mit geringer Komplexität …
  - welche die Mindestanforderungen anderer Maße erfüllt

Ansatz und Ziel

TECHNISCHE UNIVERSITÄT DARMSTADT

- Probleme bisheriger Ansätze:
  - Abgedeckte Beispiele werden aus Trainingsdaten entfernt

  ⇒ Mit steigendem Fortschritt weniger Trainingsdaten verfügbar
  - Einsatz von Heuristiken um Suchraum einzuschränken

  ⇒ Beste Regel wird eventuell nicht gefunden
  - Generierte Regeln sind für Menschen zu komplex

  ⇒ Nachvollziehbarkeit ist aber je nach Domäne sehr wichtig
  - Optimierung nur nach bestimmtem Maß möglich

  ⇒ Zu optimierende Maße sind domänenspezifisch
- EXPLORE (Exhaustive Procedure for LOgic-Rule Extraction):
  - Finden der besten Regel, bezogen auf ein gewähltes Maß…
  - in DNF (Disjunktiver Normalform) mit geringer Komplexität …
  - welche die Mindestanforderungen anderer Maße erfüllt

Ansatz und Ziel

TECHNISCHE UNIVERSITÄT DARMSTADT

- Probleme bisheriger Ansätze:
  - Abgedeckte Beispiele werden aus Trainingsdaten entfernt

  ⇒ Mit steigendem Fortschritt weniger Trainingsdaten verfügbar
  - Einsatz von Heuristiken um Suchraum einzuschränken

  ⇒ Beste Regel wird eventuell nicht gefunden
  - Generierte Regeln sind für Menschen zu komplex

  ⇒ Nachvollziehbarkeit ist aber je nach Domäne sehr wichtig
  - Optimierung nur nach bestimmtem Maß möglich

  ⇒ Zu optimierende Maße sind domänenspezifisch
- EXPLORE (Exhaustive Procedure for LOgic-Rule Extraction):
  - Finden der besten Regel, bezogen auf ein gewähltes Maß…
  - in DNF (Disjunktiver Normalform) mit geringer Komplexität …
  - welche die Mindestanforderungen anderer Maße erfüllt

Ansatz und Ziel

TECHNISCHE UNIVERSITÄT DARMSTADT

- Probleme bisheriger Ansätze:
  - Abgedeckte Beispiele werden aus Trainingsdaten entfernt

  ⇒ Mit steigendem Fortschritt weniger Trainingsdaten verfügbar
  - Einsatz von Heuristiken um Suchraum einzuschränken

  ⇒ Beste Regel wird eventuell nicht gefunden
  - Generierte Regeln sind für Menschen zu komplex

  ⇒ Nachvollziehbarkeit ist aber je nach Domäne sehr wichtig
  - Optimierung nur nach bestimmtem Maß möglich

  ⇒ Zu optimierende Maße sind domänenspezifisch
- EXPLORE (Exhaustive Procedure for LOgic-Rule Extraction):
  - Finden der besten Regel, bezogen auf ein gewähltes Maß…
  - in DNF (Disjunktiver Normalform) mit geringer Komplexität …
  - welche die Mindestanforderungen anderer Maße erfüllt

Ansatz und Ziel

TECHNISCHE UNIVERSITÄT DARMSTADT

- Probleme bisheriger Ansätze:
  - Abgedeckte Beispiele werden aus Trainingsdaten entfernt

  ⇒ Mit steigendem Fortschritt weniger Trainingsdaten verfügbar
  - Einsatz von Heuristiken um Suchraum einzuschränken

  ⇒ Beste Regel wird eventuell nicht gefunden
  - Generierte Regeln sind für Menschen zu komplex

  ⇒ Nachvollziehbarkeit ist aber je nach Domäne sehr wichtig
  - Optimierung nur nach bestimmtem Maß möglich

  ⇒ Zu optimierende Maße sind domänenspezifisch
- EXPLORE (Exhaustive Procedure for LOgic-Rule Extraction):
  - Finden der besten Regel, bezogen auf ein gewähltes Maß…
  - in DNF (Disjunktiver Normalform) mit geringer Komplexität …
  - welche die Mindestanforderungen anderer Maße erfüllt

Definitionen

TECHNISCHE UNIVERSITÄT DARMSTADT

Literal l:	Tripel bestehend aus einem Merkmal, Operator und Wert
Beispiel:	(A, =, True)
Regel r in DNF:	Besteht aus i Termen mit jeweils t_i Literalen
Form:	$\underbrace{(f_{1,1}, o_{1,1}, v_{1,1}) \wedge \dots \wedge (f_{1,t1}, o_{1,t1}, v_{1,t1})}_{\text{1. Term}} \vee \dots \vee (f_{i,1}, o_{i,1}, v_{i,1}) \wedge \dots \wedge (f_{i,ti}, o_{i,ti}, v_{i,ti})$
Beispiel:	(A, =, True) ^ (B, >, 8) V (A, =, False)
Beachten:	Bindungsstärke des ^-Operators ist stärker als die des V-Operators!
Regellänge n:	Anzahl Literale in der Regel
Term Tupel T:	Geordnete Liste, die die Termlängen einer Regel enthält.
Beispiel:	(2,1)
Literal Tupel L:	Geordnete Liste, die die Literale einer Regel enthält.
Beispiel:	((A, =, True), (B, >, 8), (A, =, False))
Beachten:	Term Tupel und Literal Tupel zusammen definieren eine Regel!

Definitionen

TECHNISCHE UNIVERSITÄT DARMSTADT

Literal l:	Tripel bestehend aus einem Merkmal, Operator und Wert
Beispiel:	(A, =, True)
Regel r in DNF:	Besteht aus i Termen mit jeweils t_i Literalen
Form:	$\underbrace{(f_{1,1}, o_{1,1}, v_{1,1}) \wedge \ldots \wedge (f_{1,t1}, o_{1,t1}, v_{1,t1})}_{\text{1. Term}} \vee \ldots \vee (f_{i,1}, o_{i,1}, v_{i,1}) \wedge \ldots \wedge (f_{i,ti}, o_{i,ti}, v_{i,ti})$
Beispiel:	(A, =, True) ^ (B, >, 8) V (A, =, False)
Beachten:	Bindungsstärke des ^-Operators ist stärker als die des V-Operators!
Regellänge n:	Anzahl Literale in der Regel
Term Tupel T:	Geordnete Liste, die die Termlängen einer Regel enthält.
Beispiel:	(2,1)
Literal Tupel L:	Geordnete Liste, die die Literale einer Regel enthält.
Beispiel:	((A, =, True), (B, >, 8), (A, =, False))
Beachten:	Term Tupel und Literal Tupel zusammen definieren eine Regel!

Definitionen

TECHNISCHE UNIVERSITÄT DARMSTADT

Literal l:	Tripel bestehend aus einem Merkmal, Operator und Wert
Beispiel:	(A, =, True)
Regel r in DNF:	Besteht aus i Termen mit jeweils t_i Literalen
Form:	$\underbrace{(f_{1,1}, o_{1,1}, v_{1,1}) \wedge \ldots \wedge (f_{1,t1}, o_{1,t1}, v_{1,t1})}_{\text{1. Term}} \vee \ldots \vee (f_{i,1}, o_{i,1}, v_{i,1}) \wedge \ldots \wedge (f_{i,ti}, o_{i,ti}, v_{i,ti})$
Beispiel:	(A, =, True) ^ (B, >, 8) V (A, =, False)
Beachten:	Bindungsstärke des ^-Operators ist stärker als die des V-Operators!
Regellänge n:	Anzahl Literale in der Regel
Term Tupel T:	Geordnete Liste, die die Termlängen einer Regel enthält.
Beispiel:	(2,1)
Literal Tupel L:	Geordnete Liste, die die Literale einer Regel enthält.
Beispiel:	((A, =, True), (B, >, 8), (A, =, False))
Beachten:	Term Tupel und Literal Tupel zusammen definieren eine Regel!

Definitionen

TECHNISCHE UNIVERSITÄT DARMSTADT

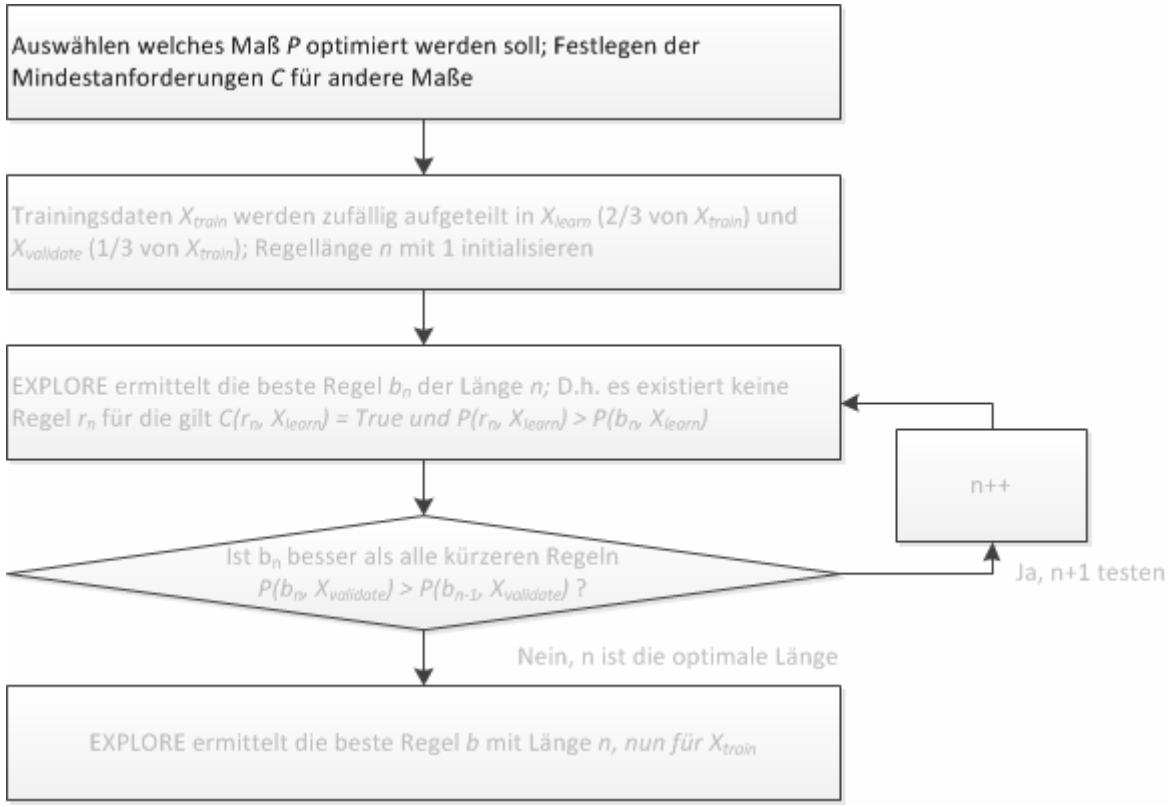
Literal l:	Tripel bestehend aus einem Merkmal, Operator und Wert
Beispiel:	(A, =, True)
Regel r in DNF:	Besteht aus i Termen mit jeweils t_i Literalen
Form:	$\underbrace{(f_{1,1}, o_{1,1}, v_{1,1}) \wedge \ldots \wedge (f_{1,t1}, o_{1,t1}, v_{1,t1})}_{\text{1. Term}} \vee \ldots \vee (f_{i,1}, o_{i,1}, v_{i,1}) \wedge \ldots \wedge (f_{i,ti}, o_{i,ti}, v_{i,ti})$
Beispiel:	(A, =, True) ^ (B, >, 8) V (A, =, False)
Beachten:	Bindungsstärke des ^-Operators ist stärker als die des V-Operators!
Regellänge n:	Anzahl Literale in der Regel
Term Tupel T:	Geordnete Liste, die die Termlängen einer Regel enthält.
Beispiel:	(2,1)
Literal Tupel L:	Geordnete Liste, die die Literale einer Regel enthält.
Beispiel:	((A, =, True), (B, >, 8), (A, =, False))
Beachten:	Term Tupel und Literal Tupel zusammen definieren eine Regel!

Definitionen

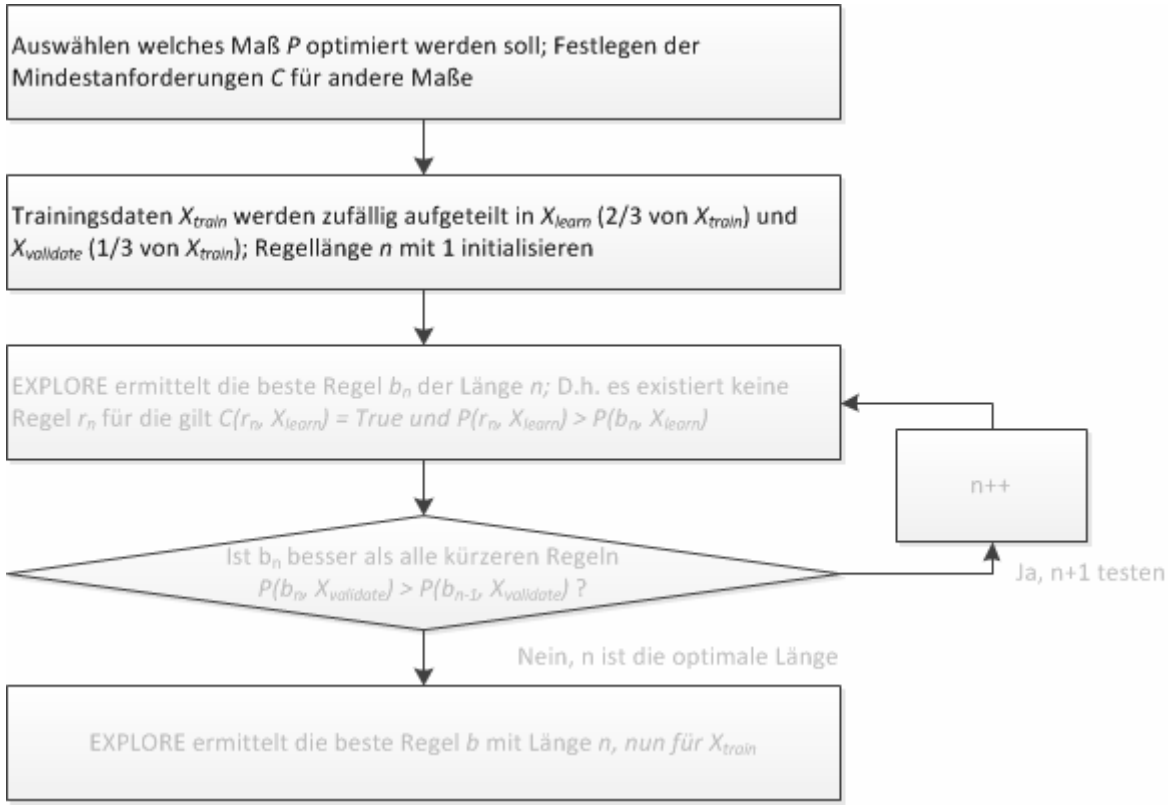
TECHNISCHE UNIVERSITÄT DARMSTADT

Literal l:	Tripel bestehend aus einem Merkmal, Operator und Wert
Beispiel:	(A, =, True)
Regel r in DNF:	Besteht aus i Termen mit jeweils t_i Literalen
Form:	$\underbrace{(f_{1,1}, o_{1,1}, v_{1,1}) \wedge \ldots \wedge (f_{1,t1}, o_{1,t1}, v_{1,t1})}_{\text{1. Term}} \vee \ldots \vee (f_{i,1}, o_{i,1}, v_{i,1}) \wedge \ldots \wedge (f_{i,ti}, o_{i,ti}, v_{i,ti})$
Beispiel:	(A, =, True) ^ (B, >, 8) V (A, =, False)
Beachten:	Bindungsstärke des ^-Operators ist stärker als die des V-Operators!
Regellänge n:	Anzahl Literale in der Regel
Term Tupel T:	Geordnete Liste, die die Termlängen einer Regel enthält.
Beispiel:	(2,1)
Literal Tupel L:	Geordnete Liste, die die Literale einer Regel enthält.
Beispiel:	((A, =, True), (B, >, 8), (A, =, False))
Beachten:	Term Tupel und Literal Tupel zusammen definieren eine Regel!

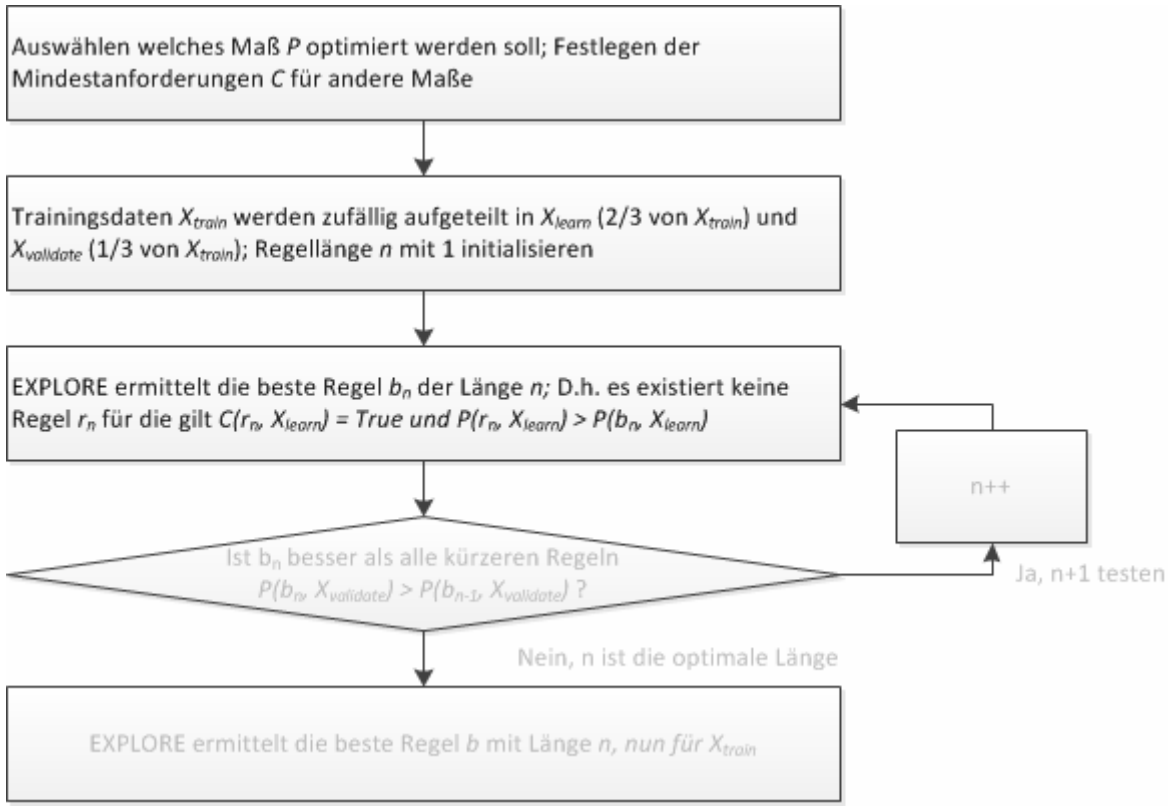
Induktionsprozess



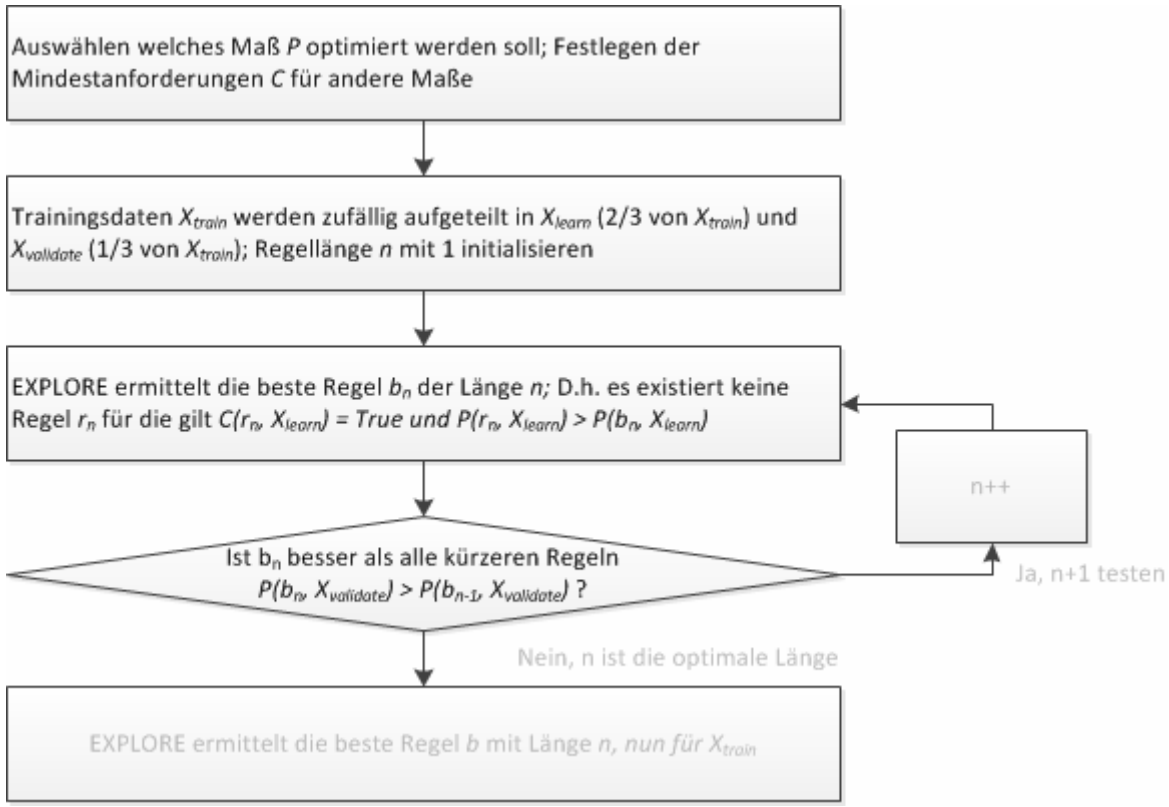
Induktionsprozess



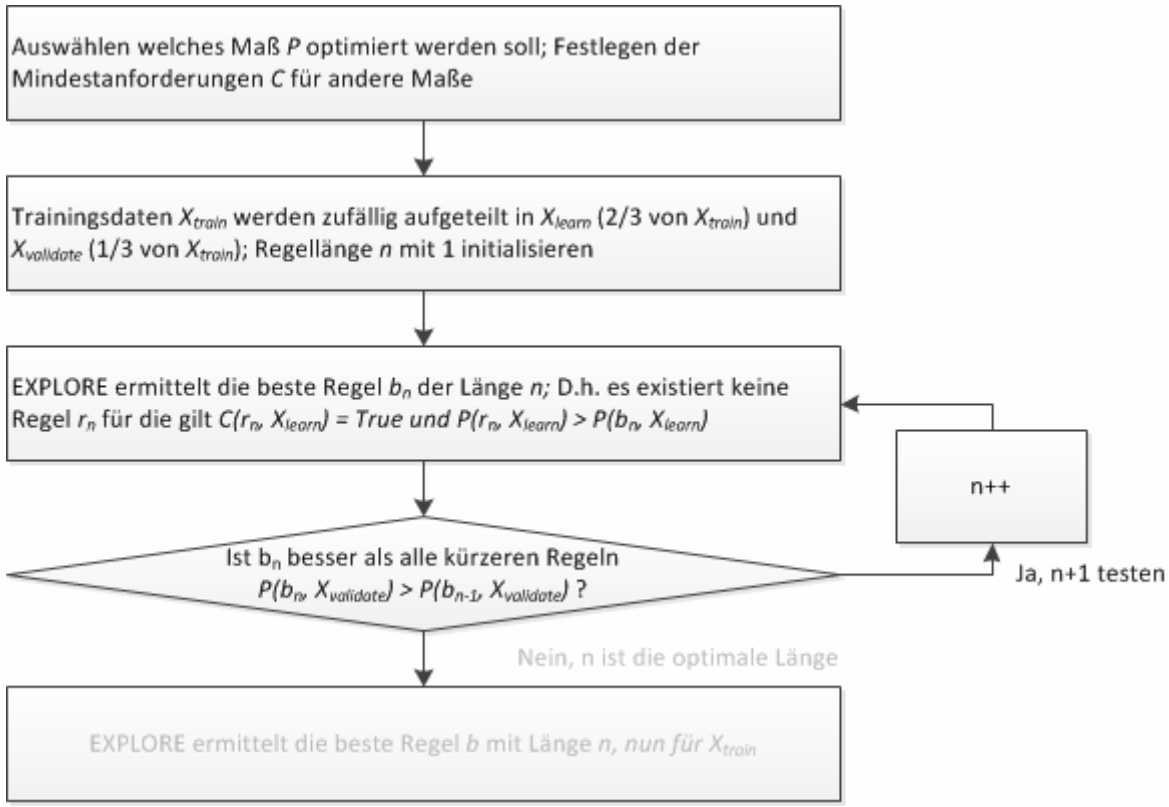
Induktionsprozess



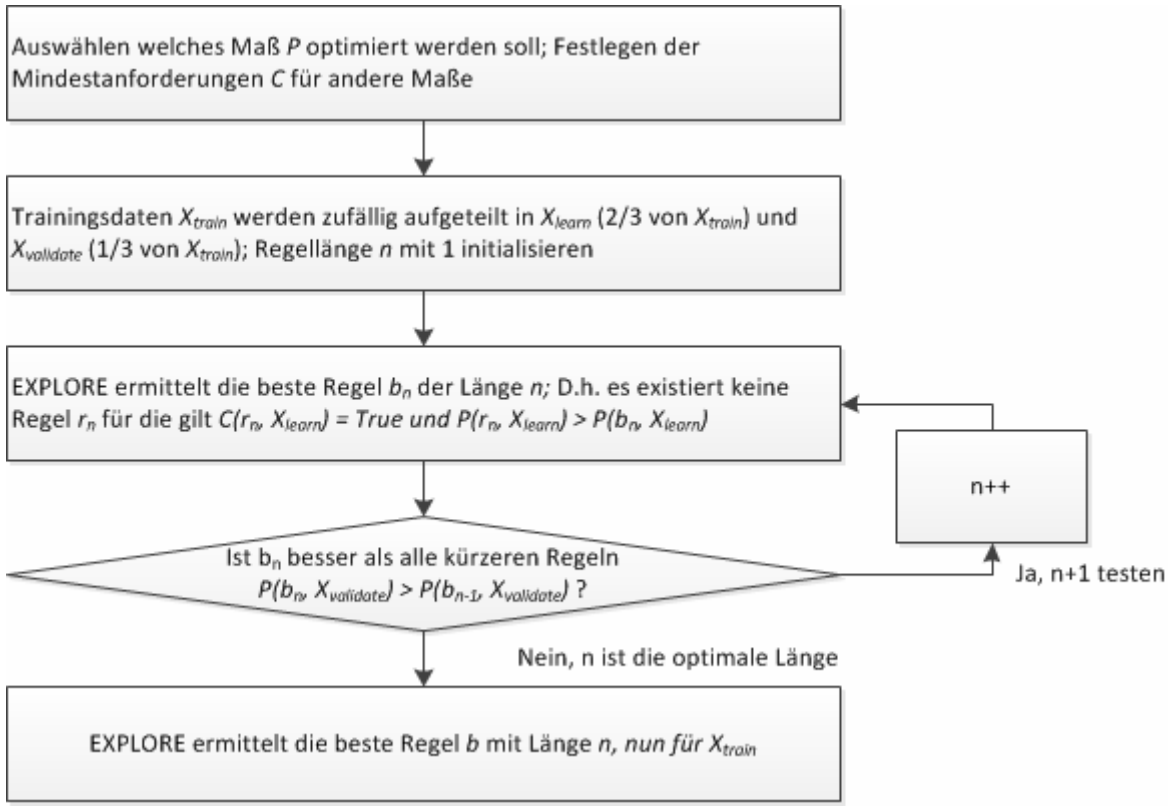
Induktionsprozess



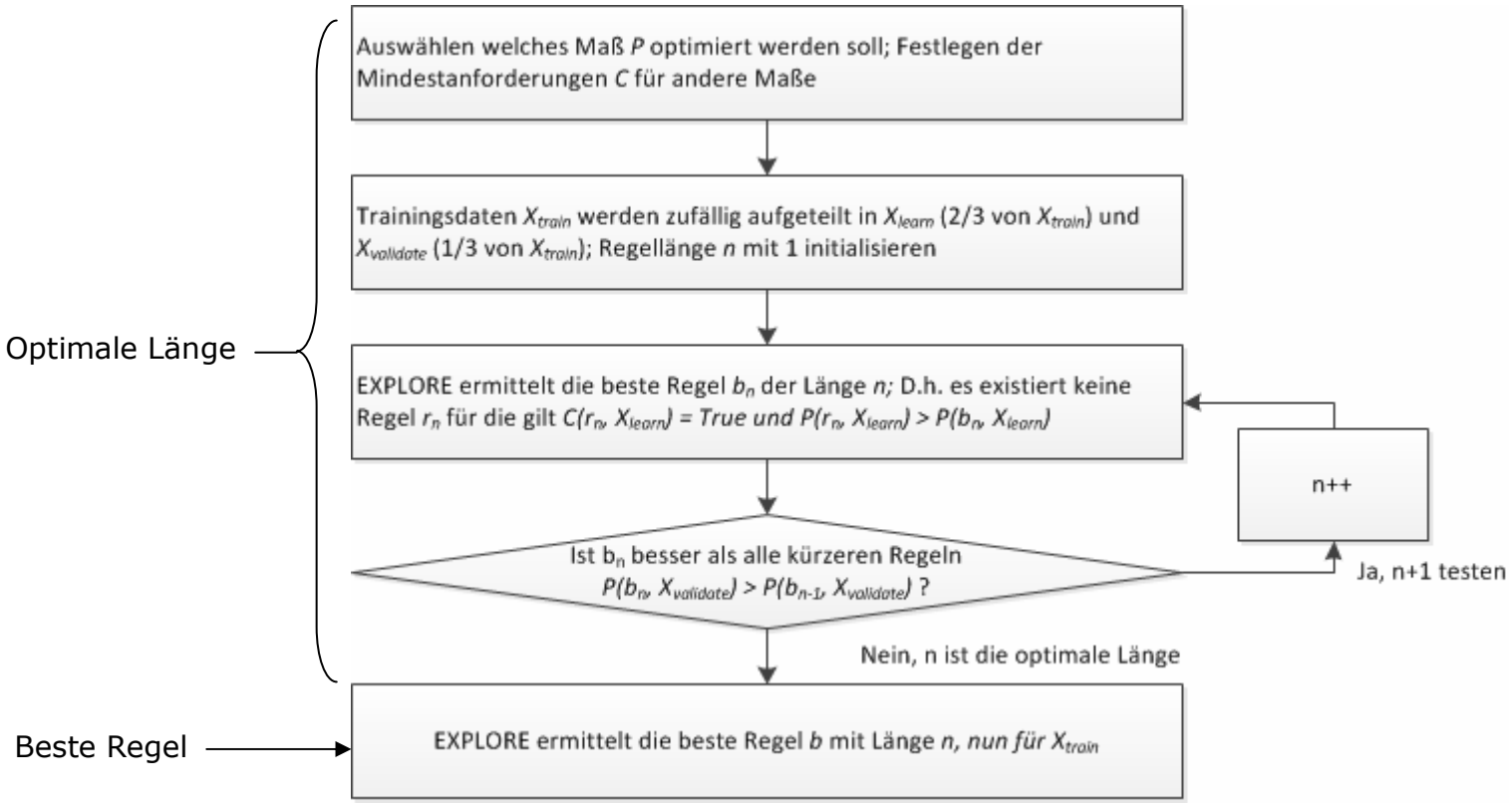
Induktionsprozess



Induktionsprozess



Induktionsprozess



EXPLORE Algorithmus

Initialisierung • Regelerstellung

TECHNISCHE UNIVERSITÄT DARMSTADT

```
    Input : X, examples; F, features; n, rule length; \
    P, performance measure; C, constraints.
    Output: b, best rule of length n.
01  FO = GenerateFeatureOperatorList(F);
02  V = GenerateValueLists(X, FO);
03  T = (n); // start with one term of size n
04  L = ∅;
05  b = ∅;
06  do
07      InitFeatureOperators(FO, L, T);
08      do
09          if InitValues(L, T ,V) then
10              do
11                  if (P(r, X) > P(b, X)) ^ \
                    C(r, X) then
12                      b = r;
13                  end
14              while NextValues(L, T ,V)
15          end
16      while NextFeatureOperators(FO, L, T)
17  while NextTermTuple(T)
18  return b;
```

EXPLORE Algorithmus

Initialisierung • Regelerstellung

TECHNISCHE UNIVERSITÄT DARMSTADT

```
    Input : X, examples; F, features; n, rule length; \
    P, performance measure; C, constraints.
    Output: b, best rule of length n.
01  FO = GenerateFeatureOperatorList(F);
02  V = GenerateValueLists(X, FO);
03  T = (n); // start with one term of size n
04  L = ∅;
05  b = ∅;
06  do
07      InitFeatureOperators(FO, L, T);
08      do
09          if InitValues(L, T ,V) then
10              do
11                  if (P(r, X) > P(b, X)) ^ \
                    C(r, X) then
12                      b = r;
13                  end
14              while NextValues(L, T ,V)
15          end
16      while NextFeatureOperators(FO, L, T)
17  while NextTermTuple(T)
18  return b;
```

Input : *F*, features.
Output: *FO*, lexicographically ordered \
list of feature-operator pairs.
…

Beispiel:
Input: *F* = {A, B}
Output: *FO* = {(A, ≤), (A, >), (B, =)}

EXPLORE Algorithmus

Initialisierung • Regelerstellung

TECHNISCHE UNIVERSITÄT DARMSTADT

```
    Input : X, examples; F, features; n, rule length; \
    P, performance measure; C, constraints.
    Output: b, best rule of length n.
01  FO = GenerateFeatureOperatorList(F);
02  V = GenerateValueLists(X, FO);
03  T = (n); // start with one term of size n
04  L = ∅;
05  b = ∅;
06  do
07      InitFeatureOperators(FO, L, T);
08      do
09          if InitValues(L, T ,V) then
10              do
11                  if (P(r, X) > P(b, X)) ^ \
                    C(r, X) then
12                      b = r;
13                  end
14              while NextValues(L, T ,V)
15          end
16      while NextFeatureOperators(FO, L, T)
17  while NextTermTuple(T)
18  return b;
```

```
Input : X, examples; \
FO, feature-operator list.
Output: V , value lists for all \
feature-operator pairs.
…

Beispiel: Folgende Folie!
```

EXPLORE Algorithmus

Initialisierung • Regelerstellung

TECHNISCHE UNIVERSITÄT DARMSTADT

Subsumtion

Input : X = {(A = 4.5, B = Medium), …, (A = 6.00, B = Low)}

Beispiel	A	Klasse
1	4.50	0
3	5.10	1
5	5.40	0

1. Durchschnitt von je zwei Werten bilden
2. TP und FP ermitteln

Literal	TP	FP
A > 4.75	3	2
A > 5.15	2	1
A > 5.70	1	0

3. Nur Literale behalten, die eine Obermenge TP und eine Teilmenge FP abdecken

Output: V (A, >) = {5.05, 5.70}
V (A, ≤) = {…}
V (B, =) = {Low, Medium, High}

EXPLORE Algorithmus

Initialisierung • Regelerstellung

TECHNISCHE UNIVERSITÄT DARMSTADT

Subsumtion

Input : X = {(A = 4.5, B = Medium), …, (A = 6.00, B = Low)}

Beispiel	A	Klasse
1	4.50	0
2	5.00	0
3	5.10	1
4	5.20	1
5	5.40	0
6	6.00	1

1. Durchschnitt von je zwei Werten bilden
2. TP und FP ermitteln

Literal	TP	FP
A > 4.75	3	2
A > 5.15	2	1
A > 5.70	1	0

3. Nur Literale behalten, die eine Obermenge TP und eine Teilmenge FP abdecken

Output: V (A, >) = {5.05, 5.70}
V (A, ≤) = {…}
V (B, =) = {Low, Medium, High}

EXPLORE Algorithmus

Initialisierung • Regelerstellung

TECHNISCHE UNIVERSITÄT DARMSTADT

Subsumtion

Input : X = {(A = 4.5, B = Medium), …, (A = 6.00, B = Low)}

Beispiel	A	Klasse
1	4.50	0
2	5.00	0
3	5.10	1
4	5.20	1
5	5.40	0
6	6.00	1

1. Durchschnitt von je zwei Werten bilden
2. TP und FP ermitteln

Literal	TP	FP
A > 4.75	3	2
A > 5.05	3	1
A > 5.15	2	1
A > 5.20	1	1
A > 5.70	1	0

3. Nur Literale behalten, die eine Obermenge TP und eine Teilmenge FP abdecken

Output: V (A, >) = {5.05, 5.70}
V (A, ≤) = {…}
V (B, =) = {Low, Medium, High}

EXPLORE Algorithmus

Initialisierung • Regelerstellung

TECHNISCHE UNIVERSITÄT DARMSTADT

Subsumtion

Input : X = {(A = 4.5, B = Medium), …, (A = 6.00, B = Low)}

Beispiel	A	Klasse
1	4.50	0
2	5.00	0
3	5.10	1
4	5.20	1
5	5.40	0
6	6.00	1

1. Durchschnitt von je zwei Werten bilden
2. TP und FP ermitteln

Literal	TP	FP
~~A > 4.75~~	~~3~~	~~2~~
A > 5.05	3	1
~~A > 5.15~~	~~2~~	~~1~~
~~A > 5.20~~	~~1~~	~~1~~
A > 5.70	1	0

3. Nur Literale behalten, die eine Obermenge TP und eine Teilmenge FP abdecken

Output: V (A, >) = {5.05, 5.70}
V (A, ≤) = {…}
V (B, =) = {Low, Medium, High}

EXPLORE Algorithmus

Initialisierung • Regelerstellung

TECHNISCHE UNIVERSITÄT DARMSTADT

Subsumtion

Input : X = {(A = 4.5, B = Medium), …, (A = 6.00, B = Low)}

Beispiel	A	Klasse
1	4.50	0
2	5.00	0
3	5.10	1
4	5.20	1
5	5.40	0
6	6.00	1

1. Durchschnitt von je zwei Werten bilden
2. TP und FP ermitteln

Literal	TP	FP
~~A > 4.75~~	~~3~~	~~2~~
A > 5.05	3	1
~~A > 5.15~~	~~2~~	~~1~~
~~A > 5.20~~	~~1~~	~~1~~
A > 5.70	1	0

3. Nur Literale behalten, die eine Obermenge TP und eine Teilmenge FP abdecken

Output: V (A, >) = {5.05, 5.70}
V (A, ≤) = {…}
V (B, =) = {Low, Medium, High}

EXPLORE Algorithmus

Initialisierung • Regelerstellung

TECHNISCHE UNIVERSITÄT DARMSTADT

```
    Input : X, examples; F, features; n, rule length; \
    P, performance measure; C, constraints.
    Output: b, best rule of length n.
01  FO = GenerateFeatureOperatorList(F);
02  V = GenerateValueLists(X, FO);
03  T = (n); // start with one term of size n
04  L = ∅;
05  b = ∅;
06  do
07      InitFeatureOperators(FO, L, T);
08      do
09          if InitValues(L, T ,V) then
10              do
11                  if (P(r, X) > P(b, X)) ^ \
                    C(r, X) then
12                      b = r;
13                  end
14              while NextValues(L, T ,V)
15          end
16      while NextFeatureOperators(FO, L, T)
17  while NextTermTuple(T)
18  return b;
```

EXPLORE Algorithmus

Initialisierung • Regelerstellung

TECHNISCHE UNIVERSITÄT DARMSTADT

```
    Input : X, examples; F, features; n, rule length; \
    P, performance measure; C, constraints.
    Output: b, best rule of length n.
01  FO = GenerateFeatureOperatorList(F);
02  V = GenerateValueLists(X, FO);
03  T = (n); // start with one term of size n
04  L = ∅;
05  b = ∅;
06  do
07      InitFeatureOperators(FO, L, T);
08      do
09          if InitValues(L, T ,V) then
10              do
11                  if (P(r, X) > P(b, X)) ^ \
                    C(r, X) then
12                      b = r;
13                  end
14              while NextValues(L, T ,V)
15          end
16      while NextFeatureOperators(FO, L, T)
17  while NextTermTuple(T)
18  return b;
```

```
Input : T, term tuple.
Output: True if the next term tuple T \
could be generated, False otherwise.
Result: T equals the next term tuple.
…

Beispiel 1:
Input : T = (5)
Output: True
Result: T = (4, 1)

Beispiel 1:
Input : T = (3, 1, 1)
Output: True
Result: T = (2, 2, 1)
```

EXPLORE Algorithmus

Initialisierung • Regelerstellung

TECHNISCHE UNIVERSITÄT DARMSTADT

```
    Input : X, examples; F, features; n, rule length; \
    P, performance measure; C, constraints.
    Output: b, best rule of length n.
01  FO = GenerateFeatureOperatorList(F);
02  V = GenerateValueLists(X, FO);
03  T = (n); // start with one term of size n
04  L = ∅;
05  b = ∅;
06  do
07      InitFeatureOperators(FO, L, T);
08      do
09          if InitValues(L, T ,V) then
10              do
11                  if (P(r, X) > P(b, X)) ^ \
                    C(r, X) then
12                      b = r;
13                  end
14              while NextValues(L, T ,V)
15          end
16      while NextFeatureOperators(FO, L, T)
17  while NextTermTuple(T)
18  return b;
```

```
Input : FO, feature-operator list; \
L, literal tuple; T, term tuple.
Result: L is instantiated with \
feature-operators.
…

Beispiel:
Input : FO = {(A, ≤), … (E, >)}; \
L = ∅; T = (5)
Result: L = ((A, ≤, ?), …, (E, ≤, ?))
```

```
Input : FO, feature-operator list; \
L, literal tuple; T, term tuple.
Output: True if L could be instantiated
    with the next set of \
    feature-operators, False otherwise.
Result : L is instantiated with the next \
set of feature-operators.
…

Beispiel:
Input : FO = {(A, ≤), …, (E, >)}; \
L = ((A, ≤, ?), …, (E, ≤, ?)); T = (5)
Output: True
Result: L = ((A, ≤, ?), …, (E, >, ?))
```

EXPLORE Algorithmus

Initialisierung • Regelerstellung

TECHNISCHE UNIVERSITÄT DARMSTADT

```
    Input : X, examples; F, features; n, rule length; \
    P, performance measure; C, constraints.
    Output: b, best rule of length n.
01  FO = GenerateFeatureOperatorList(F);
02  V = GenerateValueLists(X, FO);
03  T = (n); // start with one term of size n
04  L = ∅;
05  b = ∅;
06  do
07      InitFeatureOperators(FO, L, T);
08      do
09          if InitValues(L, T ,V) then
10              do
11                  if (P(r, X) > P(b, X)) ^ \
                    C(r, X) then
12                      b = r;
13                  end
14              while NextValues(L, T ,V)
15          end
16      while NextFeatureOperators(FO, L, T)
17  while NextTermTuple(T)
18  return b;
```

```
Input : FO, feature-operator list; \
L, literal tuple; T, term tuple.
Result: L is instantiated with \
feature-operators.
…

Beispiel:
Input : FO = {(A, ≤), … (E, >)}; \
L = ∅; T = (5)
Result: L = ((A, ≤, ?), …, (E, ≤, ?))
```

```
Input : FO, feature-operator list; \
L, literal tuple; T, term tuple.
Output: True if L could be instantiated
    with the next set of \
    feature-operators, False otherwise.
Result : L is instantiated with the next \
set of feature-operators.
…

Beispiel:
Input : FO = {(A, ≤), …, (E, >)}; \
L = ((A, ≤, ?), …, (E, ≤, ?)); T = (5)
Output: True
Result: L = ((A, ≤, ?), …, (E, >, ?))
```

EXPLORE Algorithmus

Initialisierung • Regelerstellung

TECHNISCHE UNIVERSITÄT DARMSTADT

```
    Input : X, examples; F, features; n, rule length; \
    P, performance measure; C, constraints.
    Output: b, best rule of length n.
01  FO = GenerateFeatureOperatorList(F);
02  V = GenerateValueLists(X, FO);
03  T = (n); // start with one term of size n
04  L = ∅;
05  b = ∅;
06  do
07      InitFeatureOperators(FO, L, T);
08      do
09          if InitValues(L, T ,V) then
10              do
11                  if (P(r, X) > P(b, X)) ^ \
                    C(r, X) then
12                      b = r;
13                  end
14              while NextValues(L, T ,V)
15          end
16      while NextFeatureOperators(FO, L, T)
17  while NextTermTuple(T)
18  return b;
```

```
Input : L, literal tuple; T, term tuple; \
V, set of values lists.
Output: True if L could be initialized \
with values, False otherwise.
Result : L is initialized with values.
…

Prinzip: Folgende Folie!
```

EXPLORE Algorithmus

Initialisierung • Regelerstellung

TECHNISCHE UNIVERSITÄT DARMSTADT

Branch and bound

- Zu Optimierendes Maß: Sensitivität (TP/(TP+FN))

⇒ Anzahl TP muss maximiert werden

Maximale Anzahl TP des Literals j im Term i: $TP_{i,j}$

Maximale Anzahl TP des Terms i: $TP_{max\ i} = \min(TP_{1,j}, \ldots, TP_{1,ti})$

Maximale Anzahl TP der Terme 1 bis i: $TP_{cum\ i}$

⇒ Mindestanzahl TP, die ein Term liefern muss, der mit neuen Werten initialisiert wird:

$$TP_{bound\ i} = TP_{best} - TP_{cum\ i-1} - \sum_{k=i+1}^{k=m} TP_{max\ k}$$

- Mit Mindestanforderung an: Spezifität (TN/(FP+TN))

⇒ FP_{bound} Konstanter Wert, ansonsten gleiches Prinzip

EXPLORE Algorithmus

Initialisierung • Regelerstellung

TECHNISCHE UNIVERSITÄT DARMSTADT

Branch and bound

- Zu Optimierendes Maß: Sensitivität (TP/(TP+FN))

⇒ Anzahl TP muss maximiert werden

Maximale Anzahl TP des Literals j im Term i: $TP_{i,j}$

Maximale Anzahl TP des Terms i: $TP_{max\ i} = \min(TP_{1,j}, \ldots, TP_{1,ti})$

Maximale Anzahl TP der Terme 1 bis i: $TP_{cum\ i}$

⇒ Mindestanzahl TP, die ein Term liefern muss, der mit neuen Werten initialisiert wird:

$$TP_{bound\ i} = TP_{best} - TP_{cum\ i-1} - \sum_{k=i+1}^{k=m} TP_{max\ k}$$

- Mit Mindestanforderung an: Spezifität (TN/(FP+TN))

⇒ FP_{bound} Konstanter Wert, ansonsten gleiches Prinzip

EXPLORE Algorithmus

Initialisierung • Regelerstellung

TECHNISCHE UNIVERSITÄT DARMSTADT

Branch and bound

- Zu Optimierendes Maß: Sensitivität (TP/(TP+FN))

⇒ Anzahl TP muss maximiert werden

Maximale Anzahl TP des Literals j im Term i: $TP_{i,j}$

Maximale Anzahl TP des Terms i: $TP_{max\ i} = \min(TP_{1,j}, \ldots, TP_{1,ti})$

Maximale Anzahl TP der Terme 1 bis i: $TP_{cum\ i}$

⇒ Mindestanzahl TP, die ein Term liefern muss, der mit neuen Werten initialisiert wird:

$$TP_{bound\ i} = TP_{best} - TP_{cum\ i-1} - \sum_{k=i+1}^{k=m} TP_{max\ k}$$

- Mit Mindestanforderung an: Spezifität (TN/(FP+TN))

⇒ FP_{bound} Konstanter Wert, ansonsten gleiches Prinzip

Branch and bound

- Zu Optimierendes Maß: Sensitivität (TP/(TP+FN))

⇒ Anzahl TP muss maximiert werden

Maximale Anzahl TP des Literals j im Term i: $TP_{i,j}$

Maximale Anzahl TP des Terms i: $TP_{max\ i} = \min(TP_{1,j}, \ldots, TP_{1,ti})$

Maximale Anzahl TP der Terme 1 bis i: $TP_{cum\ i}$

⇒ Mindestanzahl TP, die ein Term liefern muss, der mit neuen Werten initialisiert wird:

$$TP_{bound\ i} = TP_{best} - TP_{cum\ i-1} - \sum_{k=i+1}^{k=m} TP_{max\ k}$$

- Mit Mindestanforderung an: Spezifität (TN/(FP+TN))

⇒ FP_{bound} Konstanter Wert, ansonsten gleiches Prinzip

EXPLORE Algorithmus

Initialisierung • Regelerstellung

TECHNISCHE UNIVERSITÄT DARMSTADT

```
    Input : X, examples; F, features; n, rule length; \
    P, performance measure; C, constraints.
    Output: b, best rule of length n.
01  FO = GenerateFeatureOperatorList(F);
02  V = GenerateValueLists(X, FO);
03  T = (n); // start with one term of size n
04  L = ∅;
05  b = ∅;
06  do
07      InitFeatureOperators(FO, L, T);
08      do
09          if InitValues(L, T ,V) then
10              do
11                  if (P(r, X) > P(b, X)) ^ \
                    C(r, X) then
12                      b = r;
13                  end
14              while NextValues(L, T ,V)
15          end
16      while NextFeatureOperators(FO, L, T)
17  while NextTermTuple(T)
18  return b;
```

```
Input : L, literal tuple; T, term tuple; \
V, set of value lists.
Output: True if L could be instantiated \
with a next set of values, False otherwise.
Result : L is instantiated with the next \
set of values.
…

Prinzip: Wie bei NextFeatureOperators, nur
    mit Werten statt FeatureOperators.
Wieder Branch and bound verwendet.
```

EXPLORE Algorithmus

Initialisierung • Regelerstellung

TECHNISCHE UNIVERSITÄT DARMSTADT

```
    Input : X, examples; F, features; n, rule length; \
    P, performance measure; C, constraints.
    Output: b, best rule of length n.
01  FO = GenerateFeatureOperatorList(F);
02  V = GenerateValueLists(X, FO);
03  T = (n); // start with one term of size n
04  L = ∅;
05  b = ∅;
06  do
07      InitFeatureOperators(FO, L, T);
08      do
09          if InitValues(L, T ,V) then
10              do
11                  if (P(r, X) > P(b, X)) ^ \
                    C(r, X) then
12                      b = r;
13                  end
14              while NextValues(L, T ,V)
15          end
16      while NextFeatureOperators(FO, L, T)
17  while NextTermTuple(T)
18  return b;
```

Prüfen, ob aktuell betrachtete Regel *r* besser ist, als bisher beste Regel *b*, und ggf. als neue beste Regel *b* übernehmen

Beste Regel *b* zurückgeben

Fazit

Evaluation • Ausblick

TECHNISCHE UNIVERSITÄT DARMSTADT

- Laufzeiten für steigende Regellängen
  - Zu optimierendes Maß: Genauigkeit = (TP+TN)/(TP+FN+FP+TN)

Dataset	Examples	Features		Values	Rule length						
		Continuous	Nominal		1	2	3	4	5	6	7
australian	690	6	8	1128	00:00:00	00:00:01	00:01:23	10:32:25			
breast-cancer	277	0	10	41	00:00:00	00:00:00	00:00:01	00:00:05	00:01:21	00:24:16	06:33:36
breast-w	683	9	0	152	00:00:00	00:00:00	00:00:05	00:02:09	01:35:36		
diabetes	768	8	0	1220	00:00:00	00:00:03	00:30:23				
glass(G2)	214	9	0	380	00:00:00	00:00:00	00:00:04	00:03:04	02:42:16		
heart-statlog	270	7	6	426	00:00:00	00:00:00	00:00:31	01:20:05			
liver	345	7	0	448	00:00:00	00:00:01	00:01:57	08:59:39			
mammographic	830	3	2	127	00:00:00	00:00:00	00:00:01	00:00:22	00:17:21	05:02:41	
mushroom	5644	0	22	98	00:00:00	00:00:00	00:00:06	00:01:29	00:22:16	05:57:51	
sick	2643	6	21	954	00:00:00	00:00:02	00:01:40	01:41:03			
vote	232	0	16	32	00:00:00	00:00:00	00:00:05	00:00:33	00:05:18	00:50:47	04:14:38
wine	178	13	0	732	00:00:00	00:00:01	00:00:19	00:16:58	14:36:11		

Fazit

Evaluation • Ausblick

TECHNISCHE UNIVERSITÄT DARMSTADT

- Vergleich der Genauigkeit (SD) mit anderen Regellernern
  - EXPLORE ist signifikant • besser / ◦ schlechter

Dataset	C4.5R	CBA	CN2	1R	PART	RIPPER	RISE	SL2	EXPLORE
australian	85.1 (0.5)	85.3 (0.6)	80.8 (0.8)•	85.5 (0.0)	84.4 (0.8)•	85.4 (0.8)	83.0 (0.5)•	85.2 (0.4)	85.2 (0.4)
breast-cancer	70.8 (1.8)	73.2 (3.0)	67.3 (1.7)•	68.4 (1.3)•	71.6 (1.9)	71.4 (1.3)	73.4 (1.0)◦	69.1 (1.4)•	72.0 (0.9)
breast-w	95.4 (0.4)	94.8 (0.3)•	95.4 (0.6)•	91.7 (0.4)•	94.2 (0.4)•	94.2 (0.8)•	96.9 (0.3)◦	94.3 (0.2)•	95.9 (0.4)
diabetes	72.9 (0.9)•	60.7 (1.9)•	71.3 (0.9)•	72.4 (0.8)•	73.7 (1.1)•	74.3 (0.6)•	71.8 (0.8)•	72.1 (0.7)•	75.8 (0.3)
glass(G2)	92.5 (0.8)•	85.4 (4.7)•	93.7 (0.8)	90.1 (0.3)•	92.5 (1.1)•	91.2 (1.0)•	94.5 (0.9)	87.5 (2.1)•	93.8 (0.7)
heart-statlog	79.2 (1.4)	81.7 (1.1)◦	75.6 (1.7)•	71.3 (1.5)•	77.3 (1.3)•	79.7 (2.6)	79.2 (1.0)	70.8 (1.9)•	79.9 (1.4)
liver	63.9 (1.5)•	57.4 (1.5)•	64.0 (2.7)•	55.1 (1.7)•	64.4 (2.1)•	65.7 (1.2)•	63.9 (2.3)•	61.5 (2.9)•	68.1 (0.9)
mammographic	83.2 (0.4)	85.3 (0.4)◦	77.2 (0.7)•	82.8 (0.0)	81.7 (0.7)•	83.6 (0.5)	78.7 (0.5)•	82.3 (1.0)	83.5 (0.7)
mushroom	100.0 (0.0)	99.9 (0.0)•	100.0 (0.0)	98.4 (0.0)	100.0 (0.0)	100.0 (0.0	100.0 (0.0)	100.0 (0.0)	100.0 (0.0)
sick	97.8 (0.1)	80.3 (10.0)•	97.9 (0.1)	95.3 (0.1)•	98.2 (0.2)◦	98.0 (0.2)	97.6 (0.1)•	96.9 (0.2)•	97.8 (0.0)
vote	95.5 (0.4)•	90.6 (3.9)•	95.0 (0.5)•	97.0 (0.0)◦	95.6 (0.4)•	96.7 (0.3)◦	95.9 (0.4)•	96.8 (0.4)◦	96.3 (0.1)
wine	93.7 (0.7)	86.7 (2.7)•	93.3 (1.3)	86.4 (1.3)•	94.0 (1.1)	92.0 (1.0)•	96.8 (0.9)◦	92.6 (0.6)•	94.1 (0.5)
average	85.8 (0.8)	81.8 (2.5)	84.3 (1.0)	82.9 (0.6)	85.6 (0.9)	86.0 (0.9)	86.0 (0.7)	84.1 (1.0)	86.9 (0.5)

Fazit

Evaluation • Ausblick

TECHNISCHE UNIVERSITÄT DARMSTADT

- Vergleich der Termanzahl (SD) mit anderen Regellernern
  - EXPLORE ist signifikant • besser / ◦ schlechter

Dataset	C4.5R	CBA	CN2	1R	PART	RIPPER	RISE	SL2	EXPLORE
australian	15.3 (0.4)•	27.3 (0.4)•	31.8 (0.7)•	2.0 (0.0)•	30.3 (1.7)•	3.2 (0.6)•	256.7 (3.5)•	1.0 (0.0)◦	1.5 (0.2)
breast-cancer	8.8 (0.8)•	31.0 (1.0)•	54.0 (1.0)•	11.7 (0.7)•	16.8 (1.2)•	2.2 (0.3)◦	115.0 (2.5)•	1.4 (0.2)◦	2.5 (0.2)
breast-w	16.1 (0.3)•	21.0 (1.0)•	11.4 (0.2)•	10.0 (0.0)•	9.4 (0.8)•	11.7 (0.3)•	126.4 (4.1)•	1.9 (0.3)◦	3.6 (0.2)
diabetes	18.5 (0.6)•	2.3 (0.5)•	49.5 (0.7)•	7.6 (0.4)•	6.7 (0.4)•	2.8 (0.3)•	384.0 (2.0)•	1.2 (0.1)◦	1.6 (0.0)
glass(G2)	9.8 (0.2)•	11.3 (0.5)•	4.6 (0.2)•	2.0 (0.0)	3.9 (0.2)•	2.3 (0.4)	43.4 (2.1)•	1.9 (0.2)	2.1 (0.1)
heart-statlog	12.2 (0.2)•	37.3 (0.5)•	15.7 (0.3)•	2.0 (0.0)	16.4 (0.7)•	3.2 (0.4)•	90.1 (1.2)•	1.2 (0.1)◦	2.0 (0.0)
liver	14.7 (0.5)•	1.0 (0.1)◦	29.0 (0.6)•	10.9 (0.6)•	7.0 (0.6)•	3.4 (0.3)•	183.2 (1.4)•	2.6 (0.4)•	2.0 (0.0)
mammographic	8.7 (0.2)•	11.6 (0.4)•	96.2 (2.0)•	2.0 (0.0)◦	12.1 (1.3)•	2.3 (0.1)	228.0 (5.2)•	1.9 (0.2)◦	2.3 (0.1)
mushroom	11.6 (0.2)•	15.0 (0.0)	9.0 (0.0)	9.0 (0.0)	10.2 (0.3)•	5.0 (0.0)	13.8 (0.5)•	2.0 (0.1)◦	5.0 (0.0)
sick	17.0 (0.3)•	35.2 (5.2)•	17.1 (0.6)•	4.0 (0.3)•	14.5 (0.8)•	5.9 (0.5)•	422.3 (11.5)•	2.1 (0.2)•	1.0 (0.0)
vote	4.2 (0.3)•	11.2 (0.8)•	8.5 (0.2)•	2.0 (0.0)◦	4.2 (0.3)•	1.1 (0.1)◦	25.0 (0.6)•	1.3 (0.1)◦	2.2 (0.1)
wine	10.6 (0.2)•	14.1 (0.6)•	3.8 (0.2)•	2.6 (0.1)•	2.3 (0.1)	3.1 (0.1)•	19.7 (1.4)•	1.1 (0.1)◦	2.3 (0.1)
average	12.3 (0.3)	18.2 (0.9)	27.6 (0.6)	5.5 (0.2)	11.1 (0.7)	3.8 (0.3)	159.0 (3.0)	1.6 (0.2)	2.3 (0.1)

Fazit

Evaluation • Ausblick

TECHNISCHE UNIVERSITÄT DARMSTADT

- Vergleich der Literalanzahl (SD) mit anderen Regellernern
  - EXPLORE ist signifikant • besser / ◦ schlechter

Dataset	C4.5R	CBA	CN2	1R	PART	RIPPER	RISE	SL2	EXPLORE
australian	40.2 (1.4)•	84.4 (1.4)•	99.2 (1.1)•	2.0 (0.0)◦	75.1 (5.8)•	7.2 (1.8)•	2881.0 (44.1)•	2.0 (0.2)◦	2.3 (0.3)
breast-cancer	19.4 (2.4)•	92.1 (3.2)•	124.8 (2.3)•	11.7 (0.7)•	35.1 (2.5)•	4.3 (0.6)◦	797.5 (16.6)•	5.0 (1.6)	4.9 (0.5)
breast-w	19.8 (0.5)•	51.0 (2.0)•	29.2 (0.6)•	10.0 (0.0)•	10.9 (0.8)•	16.0 (1.2)•	1137.5 (37.1)•	15.5 (1.8)•	4.4 (0.3)
diabetes	36.8 (1.9)•	2.3 (0.5)	152.9 (1.1)•	7.6 (0.4)•	14.9 (1.4)•	7.8 (1.1)•	3071.8 (16.3)•	6.4 (1.3)•	2.4 (0.1)
glass(G2)	15.0 (0.5)•	11.7 (0.5)•	9.7 (0.5)•	2.0 (0.0)◦	6.9 (0.3)•	4.4 (1.0)•	390.2 (19.0)•	16.7 (2.1)•	2.3 (0.2)
heart-statlog	29.2 (0.6)•	112.9 (1.6)•	46.5 (0.9)•	2.0 (0.0)◦	51.1 (2.0)•	7.0 (1.0)•	1027.1 (15.1)•	4.2 (2.2)	3.7 (0.1)
liver	30.6 (1.2)•	1.0 (0.1)◦	88.6 (1.4)•	10.9 (0.6)•	19.7 (1.6)•	8.9 (1.1)•	1099.2 (8.6)•	37.1 (7.1)•	2.9 (0.1)
mammographic	16.1 (0.4)•	26.7 (1.1)•	283.9 (5.3)•	2.0 (0.0)◦	25.2 (3.7)•	3.9 (0.2)•	1084.4 (23.1)•	5.2 (0.9)•	3.0 (0.2)
mushroom	18.0 (0.4)•	32.0 (0.1)•	15.1 (0.1)	9.0 (0.0)	14.0 (0.2)•	6.0 (0.0)	172.9 (7.2)•	9.1 (0.3)•	6.0 (0.0)
sick	36.0 (0.9)•	80.9 (12.3)•	46.3 (1.6)•	4.0 (0.3)	50.6 (3.8)•	22.1 (2.1)•	10864.2 (306.7)•	18.8 (4.1)•	3.8 (0.0)
vote	8.1 (0.9)•	30.9 (2.4)•	15.8 (0.2)•	2.0 (0.0)◦	5.7 (0.5)•	1.4 (0.3)◦	260.9 (9.1)•	2.0 (0.3)◦	4.2 (0.1)
wine	14.3 (0.3)•	14.1 (0.6)•	7.7 (0.4)•	2.6 (0.1)◦	5.2 (0.2)•	4.9 (0.2)•	256.2 (18.7)•	8.3 (0.9)•	3.3 (0.1)
average	23.6 (0.9)	45.0 (2.1)	76.6 (1.3)	5.5 (0.2)	26.2 (1.9)	7.8 (0.9)	1920.2 (43.5)	10.9 (1.9)	3.6 (0.2)

Fazit

Evaluation • Ausblick

TECHNISCHE UNIVERSITÄT DARMSTADT

- In ausgewählten Fällen anwendbar.
  - Nur wenn das Problem klein genug ist,
  - oder wenn der Suchraum (Durch „Subsumtion“ und „Branch and bound“) gut einzuschränken ist

  ⇒ Ziel: Anwendbarkeit erhöhen
- Verkürzung der Laufzeit:
  - EXPLORE ist gut geeignet um parallelisiert zu werden
  - Ein Experte gibt eine gute „beste Regel“ bei der Initialisierung vor

    ⇒ Höherer „bound“ beim Start
- Einsatz von Heuristiken denkbar

  ⇒ Damit wird jedoch nicht unbedingt die „beste Regel“ gefunden!

Fazit

Evaluation • Ausblick

TECHNISCHE UNIVERSITÄT DARMSTADT

- In ausgewählten Fällen anwendbar.
  - Nur wenn das Problem klein genug ist,
  - oder wenn der Suchraum (Durch „Subsumtion“ und „Branch and bound“) gut einzuschränken ist

  ⇒ Ziel: Anwendbarkeit erhöhen
- Verkürzung der Laufzeit:
  - EXPLORE ist gut geeignet um parallelisiert zu werden
  - Ein Experte gibt eine gute „beste Regel“ bei der Initialisierung vor

    ⇒ Höherer „bound“ beim Start
- Einsatz von Heuristiken denkbar

  ⇒ Damit wird jedoch nicht unbedingt die „beste Regel“ gefunden!

Fazit

Evaluation • Ausblick

TECHNISCHE UNIVERSITÄT DARMSTADT

- In ausgewählten Fällen anwendbar.
  - Nur wenn das Problem klein genug ist,
  - oder wenn der Suchraum (Durch „Subsumtion“ und „Branch and bound“) gut einzuschränken ist

  ⇒ Ziel: Anwendbarkeit erhöhen
- Verkürzung der Laufzeit:
  - EXPLORE ist gut geeignet um parallelisiert zu werden
  - Ein Experte gibt eine gute „beste Regel“ bei der Initialisierung vor

    ⇒ Höherer „bound“ beim Start
- Einsatz von Heuristiken denkbar

  ⇒ Damit wird jedoch nicht unbedingt die „beste Regel“ gefunden!

Fazit

Evaluation • Ausblick

TECHNISCHE UNIVERSITÄT DARMSTADT

- In ausgewählten Fällen anwendbar.
  - Nur wenn das Problem klein genug ist,
  - oder wenn der Suchraum (Durch „Subsumtion“ und „Branch and bound“) gut einzuschränken ist

  ⇒ Ziel: Anwendbarkeit erhöhen
- Verkürzung der Laufzeit:
  - EXPLORE ist gut geeignet um parallelisiert zu werden
  - Ein Experte gibt eine gute „beste Regel“ bei der Initialisierung vor

    ⇒ Höherer „bound“ beim Start
- Einsatz von Heuristiken denkbar

  ⇒ Damit wird jedoch nicht unbedingt die „beste Regel“ gefunden!

Fazit

Evaluation • Ausblick

TECHNISCHE UNIVERSITÄT DARMSTADT

- In ausgewählten Fällen anwendbar.
  - Nur wenn das Problem klein genug ist,
  - oder wenn der Suchraum (Durch „Subsumtion“ und „Branch and bound“) gut einzuschränken ist

  $\Rightarrow$ Ziel: Anwendbarkeit erhöhen
- Verkürzung der Laufzeit:
  - EXPLORE ist gut geeignet um parallelisiert zu werden
  - Ein Experte gibt eine gute „beste Regel“ bei der Initialisierung vor

    $\Rightarrow$ Höherer „bound“ beim Start
- Einsatz von Heuristiken denkbar

  $\Rightarrow$ Damit wird jedoch nicht unbedingt die „beste Regel“ gefunden!

Fragen?