Vergleich zwischen geordneten und ungeordneten Regelmengen

Bachelor-Thesis von Amir Naseri
September 2011

Fachbereich Informatik
Fachgebiet Knowledge Engineering
Prof. Dr. Johannes Fürnkranz
Betreuer: Dipl.-Inform. Frederik Janssen

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Vergleich zwischen geordneten und ungeordneten Regelmengen
Comparison between ordered and unordered Rulesets

Vorgelegte Bachelor-Thesis von Amir Naseri

1. Gutachten: Johannes Fürnkranz
2. Gutachten:

Tag der Einreichung:

Erklärung zur Bachelor-Thesis

Hiermit versichere ich, die vorliegende Bachelor-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 19. September 2011

(Amir Naseri)

Inhaltsverzeichnis

1 Einleitung

Das Gebiet des maschinellen Lernens beschäftigt sich damit, wie man Computerprogramme konstruieren kann, die automatisch von der Erfahrung lernen und ihr Verhalten verbessern können [14]. Ein mögliches Anwendungsgebiet des maschinellen Lernens ist die Texterkennung. Bei der Texterkennung wird der Text in einer Bilddatei automatisch erkannt, sodass man später die Textinformationen in einer Bilddatei benutzen und bearbeiten kann, um beispielsweise eine Textsuche zu ermöglichen.

Im Gebiet des maschinellen Lernens haben wir, wie in vielen anderen Gebieten der Informatik, in letzter Zeit enorme Änderungen bei den verwendeten Algorithmen erlebt, die die Grundlage für die Theorie und Praxis bilden. Eine Klasse von Basisalgorithmen des maschinellen Lernens sind die Regellern-Algorithmen. Mit den Regellern-Algorithmen lernt man eine Regelmenge, die das durch Lernverfahren erworbene Wissen beinhaltet und repräsentiert. Das erworbene Wissen bzw. die gelernte Regelmenge kann später verwendet werden, um neue Instanzen zu klassifizieren.

Separate-and-Conquer Regellern-Algorithmen sind eine Variante der Regellern-Algorithmen. In den Separate-and-Conquer Regellern-Algorithmen wird zuerst das Lernproblem in kleinere Teilprobleme aufgeteilt, indem wir zuerst eine Regel lernen, die einen Teil von den Instanzen in den Trainingsdaten abdeckt. Anschließend werden die abgedeckten Instanzen aus den Trainingsdaten entfernt. Danach wird das Verfahren solange fortgesetzt, d. h., wir lernen Regeln für die verbleibenden Instanzen, bis alle Instanzen von gelernten Regeln abgedeckt werden [7]. In der Zwischenzeit gibt es viele Separate-and-Conquer Regellern-Algorithmen mit unterschiedlichen Eigenschaften. Ein Beispiel für Algorithmen dieser Klassen ist CN2.

Eine Implementierung einiger dieser Algorithmen befindet sich im Framework SECO (Separate-and-Conquer). Zurzeit wird im SECO-Framework eine Entscheidungsliste verwendet um neue Beispiele zu klassifizieren. Eine Entscheidungsliste ist eine geordnete Liste von Regeln, bei der für die Klassifikation genau eine Regel verwendet wird. Das Lernen von Regeln und die Klassifikation eines neuen Beispiels werden im Folgenden kurz beschrieben. Die Trainingsdaten werden zuerst durchgegangen und dann wird unter Verwendung der Separate-and-Conquer Algorithmen, eine Regelmenge gelernt. Wenn man später ein neues Beispiel im SECO-Framework klassifizieren möchte, dann werden die Regeln in der Regelmenge nacheinander durchgegangen. Sobald eine Regel zutrifft, d. h., eine Regel das Beispiel abdeckt, wird die Klasse des neuen Beispiels festgelegt, und die Klassifikation kommt zu Ende.

In dieser Arbeit wird das SECO-Framework um neue Lern- und Klassifikationsverfahren erweitert. Nach der Implementierung stehen drei allgemeine Lernverfahren und drei Klassifikationsverfahren zur Verfügung. Diese neuen Lern- und Klassifikationsverfahren sind durch eine Konfigurationsdatei einstellbar. Nun besteht auch die Möglichkeit, die Klassifikation für die ungeordneten Regelmengen durchzuführen, wobei die Reihenfolge der Regeln irrelevant ist.

Nach der Erweiterung des SECO-Frameworks werden die folgenden vier Kombinationen von Lern- und Klassifikationsverfahren miteinander verglichen.

1. Lernen durch Defaultlernverfahren und Klassifizieren durch Defaultklassifizierungsverfahren
2. Lernen durch Multi-Class-Covering-Verfahren und Klassifizieren durch Defaultklassifizierungsverfahren
3. Lernen durch Majority-Voting-Verfahren und Klassifizieren durch Majority-Voting-Verfahren
4. Lernen durch Majority-Voting-Verfahren und Klassifizieren durch Gewichtetes-Majority-Voting-Verfahren

Die vier verwendeten Vergleichskriterien sind: durchschnittliche Genauigkeit (Macro Average), durchschnittlicher Rank (Average Rank), durchschnittliche Anzahl der Regeln (Average Number of Rules), und durchschnittliche Anzahl der Bedingungen (Average Number of Conditions).

Dieser Vergleich wird unter verschiedenen Aspekten, also für Datensätze mit binärwertigen Klassen vs. Datensätze mit mehrwertigen Klassen, und für fünf verschiedene Heuristiken, nämlich Correlation, Laplace, m-Estimate, Relative-Linear-Cost und Weighted-Relative-Accuracy, durchgeführt. Am Ende werden die Ergebnisse der Vergleiche zusammengefasst, so dass man im Allgemeinen die vier Kombinationen von Lern- und Klassifikationsverfahren vergleichen kann.

2 Separate-and-conquer Regellerner

2.1 Das Lernproblem und die Grundbegriffe

Gesetzt, es sei eine Menge von Beispielen (auch Instanzen genannt), die sogenannte Trainingsmenge (auch Trainingsdaten genannt), gegeben, besteht das Ziel eines Regellern-Algorithmus nun darin, eine Regelmenge zu finden, die die gesuchte Zieltheorie ist, oder dieser sehr nahe kommt. Wobei die Zieltheorie ein Konzept ist, das verwendet wurde oder verwendet werden könnte, um die Trainingsdaten zu generieren.

Die Tabelle 2.1 illustriert eine einfache Trainingsmenge, bestehend aus sechs Beispielen. Jedes Beispiel hat vier Attribute, nämlich Aussicht, windig, Temperatur und Fahrradfahren. Jedes Attribut kann Werte aus einer Menge, also aus der Domäne, annehmen. Beispielhaft kann das Attribut „Aussicht“ die Werte sonnig, bewölkt oder Regen annehmen. Das letzte Attribut in diesem Beispiel, also „Fahrradfahren“, wird als Klassenattribut bezeichnet.

Tabelle 2.1: Beispiel für Trainingsmenge

#	Aussicht	windig	Temperatur	Fahrradfahren
1	sonnig	nein	19	+
2	Regen	ja	15	-
3	sonnig	nein	17	+
4	bewölkt	ja	16	+
5	sonnig	ja	12	-
6	Regen	nein	14	-

Ein Regellern-Algorithmus versucht eine Regelmenge (auch Theorie genannt) zu erzeugen, die den Wert des Klassenattributs (auch Klasse genannt) einer neuen Instanz bestimmen kann. Ein Beispiel für eine Regel ist:

$$Aussicht = sonnig \wedge windig = nein \rightarrow Fahrradfahren = + \tag{2.1}$$

Eine Regel besteht aus einem Regelkoft und einem Regelkörper. Der Regelkörper enthält eine, mehrere oder gar keine Bedingungen. Ein Beispiel für eine Bedingung ist hierbei Aussicht = sonnig. Eine Bedingung kann wahr oder falsch sein. Der Regelkopf enthält die Klassenzuweisung, z. B. Fahrradfahren = + . Wenn wir sagen, eine Regel deckt ein Beispiel ab, bedeutet das, dass alle Bedingungen der Regel wahr sind, wenn das Beispiel auf die Regel angewendet wird. Die formale Definition des Attributs, der Regel und der Bedingung ist [16] zu entnehmen. Beim Lernen versuchen wir eine Regelmenge zu erzeugen, die alle positiven Beispiele und kein negatives Beispiel abdeckt. Die positiven Beispiele entsprechen dem zu lernenden Konzept.

Im Prinzip haben wir es mit Instanzen mit unbekannten Klassen zu tun und wir versuchen, die neuen Instanzen zu klassifizieren, indem wir unser Wissen, das aus Trainingsdaten kommt und durch eine Regelmenge dargestellt wird, verwenden. Eine Regel kann als eine Relation zwischen Attributen mit einer Vorhersage für die Klasse gefasst werden.

Ein Separate-and-Conquer Regellern-Algorithmus erzeugt zuerst eine oder mehrere Regeln für einen Teil der Instanzen in der Trainingsmenge, für die Instanzen mit einer bestimmten Klasse. Beispielsweise kann ein Regellern-Algorithmus zuerst Regeln für die Instanzen erzeugen, deren Klasse am wenigsten in der Trainingsmenge vorkommt. Danach werden diese Instanzen aus der Trainingsmenge entfernt und das Verfahren wird fortgesetzt, indem wir für die restlichen Instanzen rekursiv Regeln lernen.

Ein Separate-and-Conquer Regellern-Algorithmus sucht nach einem Konzept, das gelernt werden soll. Dieses Konzept muss so repräsentiert werden, dass der Algorithmus mit ihm arbeiten kann. Die Repräsentation des Konzepts wird Hypothesensprache genannt. Die Hypothesensprache ist eine Beschreibungssprache für die zu erlernenden Regeln. Verschiedene Separate-and-Conquer Regellerner verwenden verschiedene Suchalgorithmen und Suchstrategien um neue Regeln zu erzeugen. Der Suchalgorithmus findet Regeln, die als mögliche Regeln zur Beschreibung des Konzepts in Frage kommen. Diese Regeln werden auch Kandidatenregeln genannt. Da man meistens nicht alle Kandidatenregeln berechnen kann, werden sie oft mit einer Heuristik bewertet. Und die besten Kandidaten werden im Prinzip ausgewählt und verfeinert. Die Suchstrategie gibt die Richtung an, in die die Kandidatenregeln verfeinert werden. Zwei mögliche Suchstrategien sind Top-Down-Suchstrategie und Bottom-Up-Suchstrategie. Bei der Bewertung einer Regel wird oft ein Wert der Regel zugeordnet, der besagt, wie gut die Qualität der Regel ist. Die Qualität einer Regel ist abhängig davon, wie viele

Instanzen die Regel korrekt klassifiziert und wie viele Instanzen sie inkorrekt klassifiziert. Die Qualität einer Regel kann durch Verfeinerung der Regel verbessert werden. Eine Regel kann verfeinert werden, indem sie spezialisiert wird, d. h. eine Bedingung zur Regel hinzugefügt wird. Oder die Regel wird durch die Generalisierung verfeinert, d. h. eine Bedingung wird von der Regel entfernt. Einige Algorithmen nehmen auch eine Heuristik in Anspruch, um Überanpassung (Overfitting) zu vermeiden. Es wird auch häufig Pruning-Mechanismen und Post-Processing-Strategien eingesetzt, um die Überanpassung zu vermeiden. Mit der Überanpassung ist gemeint, dass die Regelmenge sich zu sehr an einzelne oder einige Instanzen der Trainingsdaten anpasst und sich von der Zieltheorie distanziert. Hierbei werden oft sehr viele Bedingungen zu einer Regel hinzugefügt und daher sollten Mechanismen verwendet werden, damit die gelernte Regel vereinfacht wird. Diese Mechanismen werden Pruning-Mechanismen genannt. Dabei kann man zwei Gruppen von Instanzen unterscheiden. Die erste Gruppe wird beim Regellernen verwendet und die zweite Gruppe besteht aus Instanzen, die unterschiedlich und unabhängig von Instanzen der ersten Gruppe sind. Die erste Gruppe wird auch als Trainingsmenge und die zweite Gruppe als Pruningsmenge bezeichnet. Die Pruningsmenge wird verwendet, um die Qualität der Regeln zu prüfen, die wir mittels der Trainingsmenge gelernt haben. Daher muss der Wert des Klassenattributs der Instanzen in der Pruningsmenge bekannt sein. Falls die gelernte Regel die Instanzen in der Pruningsmenge ausreichend korrekt klassifizieren kann, bleibt sie unverändert. Andernfalls wird die Regel vereinfacht, indem die Bedingungen der Regel abgeschnitten werden, damit die Regel allgemeiner wird und hoffentlich der Zieltheorie näher kommt und nicht zu spezifisch und fokussiert auf einzelne Instanzen in der Trainingsmenge wird.

2.2 Verwandte Arbeiten

2.2.1 CN2 Algorithmus

Der CN2 Algorithmus [4] ist ein Beispiel für Algorithmen der Separate-and-Conquer Klasse. Er hat Stärken hinsichtlich des Umgangs mit Rauschdaten und verwendet gleichzeitig auch eine flexible Suchstrategie.

Im Folgenden wird der CN2 bezüglich dreier Aspekte, nämlich der Hypothesensprache, des Lernverfahrens und der Heuristiken, vorgestellt.

Hypothesensprache

Die ursprüngliche Version des CN2 Algorithmus erzeugt eine Entscheidungsliste, wobei er auch eine ungeordnete Regelmenge erzeugen kann, wenn die Evaluationsfunktion angepasst wird. Die gelernte Regel kann auch als eine if-then-Regel formuliert werden, wobei zwischen dem if- und dem then-Ausdruck eine Konjunktion der Bedingungen bzw. der Attributtests steht.

$$if \ < Aussicht = sonnig \wedge windig = nein > \ then \ Fahrrad fahren = + \tag{2.2}$$

Die letzte Regel der Regelmenge ist die Defaultregel. Die Defaultregel ist eine bedingungslose Regel, die jedem Beispiel die am häufigsten vorkommende Klasse zuweist.

Zur Klassifikation eines neuen Beispiels wird nun jede Regel der gelernten Regelmenge der Reihe nach untersucht und geprüft, ob eine Regel das Beispiel abdeckt oder nicht. Die erste Regel, die das neue Beispiel abdeckt, bildet die Vorhersage der Klasse des neuen Beispiels. Damit kommt das Verfahren anschließend zu Ende. Deckt keine Regel das neue Beispiel ab, greift die Defaultregel und bildet die Vorhersage.

Lernverfahren

Die Suchstrategie des CN2 ist eine Top-Down-Suchstrategie. Sie beginnt mit der allgemeinsten Regel und diese Regel wird verfeinert, indem eine Bedingung in jedem Schritt gelernt und zur Regel hinzugefügt wird.

In jeder Iteration des Lernalgorithmus wird eine Konjunktion der Attributtests gefunden, die durch die Evaluationsfunktion evaluiert wird und als die bestmögliche Konjunktion der Attributtests erkannt wird. Seien nun $\acute{E}$ die Beispiele, die durch die gefundene Konjunktion der Attributtests abgedeckt werden. Diese Beispiele werden im nächsten Schritt aus der Trainingsmenge entfernt.

Sei nun C die am häufigsten vorkommende Klasse der Beispiele in $\acute{E}$. Im letzten Schritt wird eine Regel folgender Form erzeugt und als die letzte Regel zur Regelmenge hinzugefügt.

$$if \ < Die \text{ bestmögliche } Konjunktion \ der \ Attributtests > \ then \ Klasse = C \tag{2.3}$$

Das Verfahren wird solange fortgesetzt, bis keine Konjunktion der Attributtests gefunden wird, die vorher festgelegten Eigenschaften aufweisen kann, oder alle Instanzen aus der Trainingsmenge entfernt worden sind.

Heuristiken

In der ursprünglichen und verbesserten Variante des CN2 werden zwei Evaluationsfunktionen bzw. Heuristiken verwendet. Im Folgenden werden die verwendeten Heuristiken in der verbesserten Variante [2] kurz vorgestellt.

Die erste verwendete Heuristik bewertet die Qualität jeder Regel, indem sie der Regel einen Wert zuordnet. Dieser Wert wird folgendermaßen berechnet.

- LaplaceAccuracy

$$h_{LPA} = \frac{p+1}{p+n+k} \tag{2.4}$$

  mit:

  - p und n sind jeweils die Anzahl der positiven und negativen Beispiele, die durch die Regel abgedeckt werden.
  - k ist die Anzahl der Klassen.

Die zweite verwendete Heuristik verhindert die Überanpassung an die Instanzen in den Trainingsdaten.

Es wird hierbei ein Signifikanztest durchgeführt, der vermeiden soll, dass noch mehr Bedingungen zu einer Regel hinzugefügt werden, die zu wenig positive, aber zu viele negative Beispiele abdeckt. Der Signifikanztest des CN2 besteht aus Loglikelihood-Ratio-Statistic [16].

- Loglikelihood-Ratio-Statistic

$$h_{LRS} = 2 * (p * log(\frac{p}{e_p}) + n * log(\frac{n}{e_n})) \tag{2.5}$$

  mit:

  - $$e_p = (p+n) * \frac{P}{P+N} \tag{2.6}$$
  - $$e_n = (p+n) * \frac{N}{P+N} \tag{2.7}$$
  - P und N sind jeweils die Gesamtzahl der positiven und negativen Beispiele in Trainingsdaten.

2.2.2 Die Kombination der Entscheidungen von mehrfachen Regeln

Die Grundidee ist, dass mehrere Theorien anstatt einer einzigen Theorie gebildet werden, und bei der Klassifikation werden die Entscheidungen der gelernten Theorien kombiniert.

Hierbei wird ein Beispiel durch mehrere Regeln klassifiziert und die Ergebnisse der Klassifikation werden durch eine Kombinationsregel kombiniert, damit die endgültige Entscheidung getroffen werden kann [12].

Dieses Verfahren ist prinzipiell aufwändiger, denn man lernt durch dieses Verfahren mehrere Regelmengen anstatt einer einzigen. Es ist auch zu beachten, dass die Zentralfrage dabei die Definition der Kombinationsregel ist. Dabei müssen die erworbenen Wissen aller Theorien so miteinander kombiniert werden, dass wir uns der Zieltheorie nähern können.

Folgende Kombinierungsmethoden wurden in [12] erwähnt und beschrieben.

- Die beste Regel: Zwischen allen Regeln, die eine neue Instanz abdecken, wird nur die beste Regel zur Klassifikation verwendet [1].
- Die Aufsummierung der Verteilungen: Hierbei werden zuerst alle Regeln, die bei der Klassifikation einer neuen Instanz zutreffen, betrachtet. Die Anzahl der abgedeckten Beispiele mit verschiedenen Klassen für jede solcher Regeln wird aufsummiert. Die Instanz bekommt dann die Mehrheits-Klasse der Verteilung zugewiesen [3].
- Die Abstimmung (Voting): Jede Regel hat eine Stimme und wählt eine Klasse aus. Die neue Instanz bekommt die Klasse mit der größten Anzahl der Stimmen zugewiesen [11].

- Naive Bayessche Kombination: Hierbei wird für jede Klasse durch die naive Bayessche Formel die Wahrscheinlichkeit berechnet [10], wobei die Bedingungen A_i von k Regeln, die das neue Beispiel abdecken, verwendet werden [15].

$$P(C|A_1,\ldots,A_k) = P(C)\prod_{i=1}^{k}\frac{P(C|A_i)}{P(C)} \tag{2.8}$$

2.3 Separate-and-Conquer Regellerner im SeCo-Framework

Die Vorgehensweise beim Separate-and-Conquer Regellern-Algorithmus wurde oben beschrieben. Das Ergebnis des Separate-and-Conquer Regellern-Algorithmus ist eine Liste von Regeln, die nacheinander gelernt worden sind.

Die folgenden beiden Algorithmen aus [9] realisieren das Lernen in SeCo.

Algorithmus 1 AbstractSeCo (Examples)

```
Theory ← ∅
Examples ← SortExamplesByClassValue (Examples)
Growing = SplitData (Examples, Splitsize)
Pruning = SplitData (Examples, 1-Splitsize)
while Positive (Growing) ≠ ∅ do
  Rule = FindBestRule (Growing, Pruning)
  Covered = Cover (Rule, Growing)
  if RuleStoppingCriterion (Theory, Rule, Growing) then
    exit while
  end if
  Growing = WeightInstances (Covered)
  Theory = Theory ∪ Rule
end while
Theory = PostProcess (Theory, Growing, Pruning)
return Theory
```

Algorithmus 2 FindBestRule (Growing, Pruning)

```
InitRule = InitializeRule (Growing)
InitVal = EvaluateRule (InitRule)
BestRule = < InitVal, InitRule >
Rules = {BestRule}
while Rules ≠ ∅ do
  Candidates = SelectCandidates (Rules, Growing)
  Rules = Rules\Candidates
  for Candidate ϵ Candidates do
    Refinements = RefineRule (Candidate, Growing)
    for Refinement ϵ Refinements do
      Evaluation = EvaluateRule (Refinement, Growing)
      unless StoppingCriterion (Refinement, Evaluation, Pruning)
        NewRule = < Evaluation, Refinement >
        Rules = InseretSort (NewRule, Rules)
        if NewRule > BestRule then
          BestRule = NewRule
        end if
    end for
  end for
  Rules = FilterRules (Rules, Growing)
end while
return BestRule
```

Algorithmus 1 liefert eine Übersicht über die äußere Schleife des Lernverfahrens in SECO. Die Theorie „Theory“ wird zuerst mit der leeren Menge initialisiert und die Instanzen in der Trainingsmenge werden sortiert, wobei die Anzahl der Instanzen in jeder Klasse eine zentrale Rolle spielt. Daraufhin werden die Instanzen in zwei Teilmengen, nämlich „growing set“ und „pruning set“, aufgeteilt, um weitere Funktionalitäten, z. B. eine zusätzliche Pruning-Phase zu implementieren.

Falls die Trainingsmenge noch positive Beispiele enthält, wird die FINDBESTRULE Methode aufgerufen, die die bestmögliche Regel zurückliefert. Dann wird in RULESTOPPINGCRITERION überprüft, ob die gelernte Regel bestimmte Qualitätskriterien erfüllt. Falls die bestimmten Qualitätskriterien erfüllt sind, d. h., falls wir die gewünschte Regelmenge bzw. Theorie gelernt haben, kommen die iterativen Schritte des Regellern-Algorithmus zu Ende. Andernfalls werden die abgedeckten Beispiele aus den Trainingsdaten entfernt und die gelernte Regel zur Theorie hinzugefügt. Diese Schritte, also das Lernen einer Regel und die Entfernung der durch die Regel abgedeckten Beispiele aus der Trainingsmenge und das Hinzufügen der Regel zur Theorie, werden im Prinzip so lange fortgesetzt, bis kein positives Beispiel in den Trainingsdaten enthalten ist. Es besteht aber im Framework auch die Möglichkeit, ein anderes Abbruchkriterium auszuwählen. Im letzten Schritt und nachdem die Theorie vollständig gebildet wurde, wird eine Nachbearbeitungsphase(POSTPROCESS) zur Optimierung durchgeführt.

Algorithmus 2 bildet die innere Schleife des Lernverfahrens, wobei auf der Suche nach der bestmöglichen Regel verschiedene Methoden benutzt werden. Der FINDBESTRULE Algorithmus, erklärt in [16], beginnt mit einer initialen Regel, die am Anfang als die beste Regel bezeichnet wird, und in die Regelmenge „Rules“ aufgenommen wird. Solange die Regelmenge „Rules“ noch Regeln hat, werden aus dieser Menge Kandidatenregeln ausgewählt und der Kandidatenmenge „Candidates“ zugewiesen. In der äußeren for-Schleife wird für jede der Kandidatenregeln eine Verfeinerung (REFINERULE) durchgeführt. Angesichts der verwendeten Suchstrategie kann man verschiedene Varianten von Verfeinerungsprozessen, wie Top-Down-Strategie und Bottom-Up-Strategie, definieren und implementieren.

In der inneren for-Schleife wird später die verfeinerte Regel mittels der Evaluierungsfunktion bewertet und daraufhin mit STOPPINGCRITERION überprüft, um herauszufinden, ob eine weitere Verfeinerung dieser Regel sinnvoll ist. Ist eine weitere Verfeinerung der Regel nicht sinnvoll, so wird die Regel in die Regelmenge „Rules“ aufgenommen und wenn sie besser als die bisher beste gelernte Regel ist, wird sie als die beste Regel „BestRule“ gelten. Am Ende wird die beste Regel zurückgeliefert.

In dieser Arbeit und besonders bei der Erweiterung des Frameworks um das Majority-Voting-Lernverfahren werden auch diese zwei Hauptprozeduren verwendet und angepasst.

3 Erweiterung des SECO-Frameworks

Das SECO-Framework ist eine Implementierung von Lernalgorithmen der Klasse Separate-and-Conquer. In SECO sind verschiedene Eigenschaften von Lernalgorithmen durch eine XML-Konfigurationsdatei einstellbar. Eine Beschreibung der Elemente dieser Konfigurationsdatei befindet sich in [16]. Nach der Erweiterung besteht die Möglichkeit, das Lernen von Regeln und die Klassifikation sowohl für die ungeordneten Regelmengen als auch für die Entscheidungslisten durch die XML-Konfigurationsdatei einzustellen. Der folgende Ausschnitt der XML-Konfigurationsdatei illustriert, wie man in SECO das Majority-Voting-Verfahren zum Lernen und zur Klassifikation auswählen kann.

```
<secomp interface="rulelearning" classname="LearningByMajorityVoting"
package="de.tu_darmstadt.ke.seco.learning.method">
</secomp>

<secomp interface="instanceclassification" classname="ClassificationByMajorityVoting"
package="de.tu_darmstadt.ke.seco.classification.method">
</secomp>
```

Das oben dargestellte XML-Schema kann für verschiedene Lern- und Klassifikationsverfahren verwendet werden. Beim Lernen muss dabei in der folgenden Zeile „VerfahrenX“ durch das gewünschte Verfahren ersetzt werden.

```
<secomp interface="rulelearning" classname="VerfahrenX" ...
```

Analog dazu muss auch beim Klassifizieren folgende Zeile angepasst werden.

```
<secomp interface="instanceclassification" classname="VerfahrenX" ...
```

Im Folgenden werden verschiedene Lern- und Klassifikationsverfahren vorgestellt.

3.1 Umsetzung des Lernens von Regeln

3.1.1 Defaultlernverfahren

Vor der Erweiterung von SECO existierte ein Lernverfahren, das die in 2.3 dargestellten Algorithmen 1 und 2 realisiert hatte. Es ist auch zu beachten, dass die dargestellten Algorithmen generisch sind, und man eine Vielzahl unterschiedlicher Lernverfahren mit ihnen implementieren kann.

Bei diesem Lernverfahren, das hier Defaultlernverfahren genannt wird, wurden zuerst alle Werte des Klassenattributs nach Anzahl der Instanzen mit diesem Klassenattribut sortiert und dann für jeden möglichen Wert des Klassenattributs mit Ausnahme des letzten Wertes Regeln gelernt. Der letzte Wert des Klassenattributs, also der Wert mit der höchsten Anzahl an Beispielen in den Trainingsdaten, bildet die Defaultregel.

Folgendes Beispiel erläutert die Vorgehensweise bei diesem Verfahren. Es seien c1, c2 und c3 drei Werte des Klassenattributs in den Trainingsdaten. Des weiteren sei c3 der Wert des Klassenattributs mit der höchsten Anzahl an Beispielen in den Trainingsdaten, und c2 käme auch häufiger als c1 in den Trainingsdaten vor. Es wird zuerst eine Regelmenge für Beispiele mit dem Klassenattributswert c1 gelernt, dabei sind alle Beispiele in den Trainingsdaten mit der Klasse c1 die positiven Beispiele und die mit c2 und c3 die negativen. Nach diesem Schritt werden alle Beispiele mit dem Klassenattributswert c1 entfernt. Dann wird eine Regelmenge für die Beispiele mit der Klasse c2 gelernt. Hierbei sind alle Beispiele mit dem Klassenattributswert c2 positiv und die mit c3 negativ. Die Beispiele mit der Klasse c1 wurden vorher aus der Menge der zu betrachtende Beispiele entfernt. Für c3 wird keine Regelmenge gelernt, da c3 die Defaultregel bildet. Die unten stehende Tabelle fasst die Erklärung zusammen.

Tabelle 3.1: Defaultlernverfahren

Klassenattributswert	positiv	negativ
c1	c1	c2, c3
c2	c2	c3

3.1.2 Majority-Voting-Lernverfahren

Um ein neues Beispiel mittels des Majority-Voting-Verfahrens zu klassifizieren, mussten wir vorher den Lernvorgang anpassen. Diese Anpassung lässt sich am besten durch folgendes Beispiel erklären.

Es seien c1, c2 und c3 wie oben in 3.1.1 beschrieben. Nun fangen wir wieder mit c1 an, dann sind wieder die Beispiele mit der Klasse c1 die positiven Beispiele und die mit c2 und c3 die negativen. Setzen wir das Verfahren mit c2 fort, nun sind die Beispiele mit dem Klassenattributswert c2 die positiven Beispiele und die Beispiele mit c1 und c3 die negativen. An dieser Stelle kommt der Unterschied zum Defaultlernverfahren bei der Entfernung der Beispiele zum Vorschein, weil die Beispiele mit der Klasse c1 nicht aus der Menge der zu betrachtenden Beispiele entfernt wurden. Als letztes wird der Lernvorgang im Gegenteil zum Lernvorgang beim Defaultlernverfahren mit c3 fortgesetzt. Hier lernen wir wieder Regeln, die c3 vorhersagen, gleichzeitig bildet c3 unsere Defaultregel. Unten findet sich wieder eine zusammenfassende Tabelle.

Tabelle 3.2: Majority-Voting-Lernverfahren

Klassenattributswert	positiv	negativ
c1	c1	c2, c3
c2	c2	c1, c3
c3	c3	c1, c2

3.1.3 Multi-Class-Covering-Lernverfahren

Bei diesem Lernverfahren wurde folgender Algorithmus implementiert.

Algorithmus 3 Multi-Class-Covering-Lernverfahren

Input: ϵ *set of training examples*
Output: R *the learned rule set*
  $R \leftarrow \emptyset$
  repeat
    $r \leftarrow \emptyset$
    for each class c_i, i = 1 to C **do**
      $P_i \leftarrow \{subset\ of\ examples\ in\ \epsilon\ with\ class\ label\ c_i\}$
      $N_i \leftarrow \{subset\ of\ examples\ in\ \epsilon\ with\ other\ class\ labels\}$
      $\dot{r} =$ LERNONERULE (c_i, P_i, N_i)
      if $\dot{r}$ better than r according to quality criterion **then**
        $r = \dot{r}$
      end if
    end for
    $R = R \cup r$
    $\epsilon = \epsilon \backslash Covered(r, \epsilon)$
  until R satisfies a quality threshold or ϵ is empty
  $R = R \cup default\ rule$

Gegeben sei die Tainingsmenge ϵ, dann wird durch das Multi-Class-Covering-Lernverfahren eine Regelmenge R gelernt und zurückgeliefert. Die Regelmenge wird mit der leeren Menge initialisiert.

Die äußere Schleife des Multi-Class-Covering-Verfahrens beginnt mit der Initialisierung einer Regel r und wird solange fortgesetzt, bis eine Qualitätsschwelle erfüllt ist, oder die Trainingsmenge ϵ leer ist. Die Qualitätsschwelle kann folgendermaßen festgelegt werden: Eine Prozentzahl, z. B. 90% der Beispiele von Trainingsdaten ϵ sind abgedeckt worden.

In der inneren for-Schleife wird für jede der möglichen Klassen der Beispiele in den Trainingsdaten eine

Regel $\dot{r}$ gelernt. Wobei die Klasse des Beispiels als positive Klasse und alle anderen möglichen Klassen der Beispiele in den Trainingsdaten als negative Klasse betrachtet werden. Dabei ist zu beachten, dass die Beispiele in den Trainingsdaten in jedem Durchlauf der äußeren Schleife in der folgenden Zeile aktualisiert werden.

$$\epsilon = \epsilon \backslash Covered(r, \epsilon) \tag{3.1}$$

Die gelernte Regel $\dot{r}$ in der implementierten Version wird durch den Aufruf der Methode FINDBESTRULE erzeugt. Diese Methode wurde im Algorithmus 2 erklärt. Wir lernen demnach für jede Klasse der Instanzen die bestmögliche Regel. Nachdem wir die Regel $\dot{r}$ gelernt haben, wird $\dot{r}$ mit r durch ein Qualitätskriterium verglichen, und r die bessere Regel zugewiesen. Dabei ist bemerkenswert, dass am Anfang des ersten Durchlaufs der inneren for-Schleife die Regel r leer ist, und sie im ersten Durchlauf die gefundene Regel $\dot{r}$ mit c1 als die Klasse zugewiesen bekommt. Im zeiten Durchlauf der inneren for-Schleife kann r entweder die Klasse c1 oder die Klasse c2 haben. Man kann es fortsetzen und sagen, dass die Regel r im C-ten Durchlauf der inneren for-Schleife jede der möglichen Klassen der Instanzen haben kann. Hierbei sollte wieder berücksichtigt werden, dass einige Beispiele im ersten Durchlauf der äußeren Schleife in der Zeile 3.1 entfernt werden, und im zweiten Durchlauf der äußeren Schleife können eventuell nicht alle Klassen für r vorhanden sein, die im ersten Durchlauf der äußeren Schleife existierten.

Als Qualitätskriterium für den Vergleich zwischen Regeln kann man die gegebene Heuristik oder die Anzahl der abgedeckten Beispiele oder die Regellänge benutzen. Bei der Anzahl der abgedeckten Beispiele als einem Kriterium kann man beispielsweise die Regeln, die mehr Beispiele abdecken, den anderen vorziehen, d. h. die allgemeineren Regeln werden bevorzugt. Wobei dieses Kriterium, je nach der verwendeten Strategie, unterschiedlich aussehen kann. Es kann auch eine Kombination von verschiedenen Kriterien verwendet werden.

Um das Multi-Class-Covering-Verfahren den zwei anderen Lernverfahren gegenüberstellen zu können, nehmen wir wieder an, dass c1, c2 und c3 wie bei den zuvor beschriebenen Lernverfahren drei Werte des Klassenattributs sind. Ein wichtiger Unterschied zwischen dem Multi-Class-Covering-Verfahren und den zwei anderen Lernverfahren ist, dass es beim Multi-Class-Covering-Verfahren nicht strikt festgesetzt wird, dass man zuerst für c1, dann für c2 und danach eventuell für c3 Regeln lernen muss, sondern die gelernte Regel „r“ kann nach der inneren for-Schleife jeden der möglichen Werte c1, c2 oder c3 als Vorhersage haben, wobei die zwei anderen Lernverfahren zuerst nur Regeln für die Klasse c1 lernen.

3.2 Umsetzung der Klassifikation

3.2.1 Defaultklassifizierungsverfahren

Vor der Erweiterung von SECO war dieses Klassifikationsverfahren das einzig zur Verfügung stehende Klassifikationsverfahren. Dieses Verfahren wird eingesetzt, wenn wir vorher beim Lernen eine Entscheidungsliste gelernt haben, d. h. eine Regelmenge, wobei die Reihenfolge der gelernten Regeln in dieser Menge entscheidend ist. Zur Klassifikation eines neuen Beispiels gehen wir nun diese Regelmenge durch und untersuchen, ob eine Regel das Beispiel abdeckt oder nicht. Die erste Regel, die das neue Beispiel abdeckt, bildet die Vorhersage der Klasse des neuen Beispiels. Und das Verfahren kommt danach zu Ende, d. h. die restlichen Regeln der Regelmenge werden nicht untersucht. Das folgende Beispiel dient zur Erläuterung des DefaultKlassifizierungsverfahrens.

Es seien R1, R2 und die Defaultregel die Regeln der gefundenen Regelmenge, und c1, c2 und c3 die Klassenattributswerte. B1, B2 und B3 sind hierbei die Bedingungen, die wahr oder falsch sind.

$$R1:\ B1 \wedge B2 \rightarrow Klasse = c1 \tag{3.2}$$

$$R2:\ B2 \wedge B3 \rightarrow Klasse = c2 \tag{3.3}$$

$$Defaultregel:\ True \rightarrow Klasse = c3 \tag{3.4}$$

Unter Verwendung des Defaultklassifizierungsverfahrens wird folgendermaßen vorgegangen. Kommt ein neues Beispiel zur Klassifikation ins Framework, wird geprüft, ob die erste Regel R1 das Beispiel abdeckt oder nicht. Nehmen wir an, dass das Beispiel von der Regel R1 abgedeckt wird, d. h. B1 und B2 beide wahr sind. Folglich wird für das Beispiel die Klasse c1 vorhergesagt, und das Verfahren kommt zu Ende und R2 und die Defaultregel bleiben unberücksichtigt.

Die Entscheidungslisten haben die positive Eigenschaft, dass die Kollisionen bei der Klassifikation nicht auftreten. Eine Kollision wäre im vorgeführten Beispiel vorgekommen, falls wir R2 untersucht hätten, und R2 auch das neue Beispiel abgedeckt hätte. Die negative Eigenschaft der Entscheidungslisten taucht aber auf,

wenn man die Bedeutung von den Regeln verstehen möchte. Denn die Bedeutung von einer Regel ist abhängig von allen Regeln, die vor der Regel in der Entscheidungsliste stehen. Eine Entscheidungsliste kann hierbei als eine Liste von if-then-else-Ausdrücken betrachtet werden. Die negative Eigenschaft der Entscheidungslisten ist folgenschwer, denn eine gelernte Regel muss im Prinzip von Experten geprüft werden, bevor sie angewendet wird [2].

3.2.2 Majority-Voting-Klassifizierungsverfahren

Als das erste Verfahren zur Klassifikation bei ungeordneten Regelmengen wird das Majority-Voting-Verfahren umgesetzt. Es sei folgende Regelmenge gegeben:

$$R1: B1 \wedge B2 \rightarrow Klasse = c1 \tag{3.5}$$

$$R2: B2 \wedge B3 \rightarrow Klasse = c2 \tag{3.6}$$

$$R3: B4 \wedge B5 \rightarrow Klasse = c3 \tag{3.7}$$

$$R4: B6 \rightarrow Klasse = c3 \tag{3.8}$$

$$Defaultregel: True \rightarrow Klasse = c3 \tag{3.9}$$

An dieser Stelle wird eine kurze Beschreibung des implementierten Verfahrens gegeben. Wenn ein neues Beispiel zur Klassifikation ins SECO-Framework kommt, werden zuerst alle Klassenattributswerte, die in Regelköpfen vorkommen, initialisiert, indem sie mit einer Null als Frequenz abgespeichert werden. Im obigen Beispiel werden wir folgende Liste haben:

$$< Klasse; Frequenz > \ Liste: < c1; 0 >, < c2; 0 >, < c3; 0 > \tag{3.10}$$

Dabei ist zu beachten, dass Klassenattributswerte eindeutig sind. Dann wird jede Regel in der Regelmenge durchgegangen und dabei überprüft, ob die Regel das Beispiel abdeckt, oder nicht. Deckt die Regel das Beispiel ab, so wird die Frequenz des Klassenattributswertes um 1 erhöht. Nehmen wir an, dass das Beispiel von R2, R3 und R4 abgedeckt wird. Nun werden die Frequenzen folgendermaßen aktualisiert:

$$< Klasse; Frequenz > \ Liste: < c1; 0 >, < c2; 1 >, < c3; 2 > \tag{3.11}$$

Nachdem alle Regeln durchgegangen worden sind, wird der Klassenattributswert mit der höchsten Frequenz als der Klassenattributswert des neuen Beispiels genommen. In diesem Fall wird für das Beispiel der Klassenattributswert c3 vorhergesagt.

Im vorgeführten Beispiel wurde angenommen, dass c3 der Wert des Klassenattributs mit der höchsten Anzahl an Instanzen in Trainingsdaten ist, daher blidet es auch die Vorhersage der Defaultregel. Es ist zu beachten, dass die Defaultregel beim Hochzählen der Frequenzen keine Probleme verursacht, denn die Defaultregel gehört nicht zur Regelmenge, für die wir die Frequenzen berechnen. Beispielhaft werden hier die Regeln R1, R2, R3 und R4 zur Berechnung der Frequenzen betrachtet und nicht die Defaultregel. Deswegen wird die Frequenz für den Klassenattributswert c3 auf 2 und nicht auf 3 gesetzt. Die Defaultregel greift im Prinzip nur dann, wenn es keine Regel gibt, die das zu klassifizierende Beispiel abdeckt.

3.2.3 Gewichtetes-Majority-Voting-Klassifizierungsverfahren

Beim gewichteten-Majority-Voting-Verfahren wird zuerst genauso wie beim Majority-Voting-Verfahren vorgegangen, d. h. zuerst werden alle Klassenattributswerte, die in Regelköpfen vorkommen, initialisiert, indem sie mit einer Null als Frequenz abgespeichert werden. Dann wird jede Regel in der Regelmenge durchgegangen und dabei überprüft, ob die Regel das Beispiel abdeckt, oder nicht. Der Unterschied ist bei der Erhöhung der Frequenzen zu sehen. Deckt die Regel das Beispiel ab, so wird die Frequenz des Klassenattributswertes um den Regelwert erhöht. Wir benutzen wieder das vorher eingeführte Beispiel mit der zusätzlichen Annahme, dass R2, R3 und R4 jeweils 0,4, 0,1 und 0,2 als Regelwert haben. Nun werden die Frequenzen folgendermaßen aktualisiert:

$$< Klasse; Frequenz > \ Liste: < c1; 0,0 >, < c2; 0,4 >, < c3; 0,3 > \tag{3.12}$$

Wenn wir alle Regeln durchgegangen sind, bildet der Klassenattributswert mit der höchsten Frequenz den Klassenattributswert des neuen Beispiels, genauso wie beim Majority-Voting-Verfahren. In diesem Fall wird für das Beispiel der Klassenattributswert c2 vorhergesagt.

4 Evaluation der verwendeten Lern- und Klassifikationsverfahren

4.1 Einleitung

Nach der Erweiterung des SECO-Frameworks stehen vier verschiedene Kombinationen von Lern- und Klassifikationsverfahren zur verfügung. Die vier verschiedenen Kombinationen sind:

1. Lernen durch Defaultlernverfahren und Klassifizieren durch Defaultklassifizierungsverfahren
2. Lernen durch Multi-Class-Covering-Verfahren und Klassifizieren durch Defaultklassifizierungsverfahren
3. Lernen durch Majority-Voting-Verfahren und Klassifizieren durch Majority-Voting-Verfahren
4. Lernen durch Majority-Voting-Verfahren und Klassifizieren durch gewichtetes-Majority-Voting-Verfahren

Als nächstes werden diese vier verschiedenen Kombinationen anhand einer Datenmenge, bestehend aus 30 Datensätzen, verglichen. Dabei wird die Kreuz-Validierung mit 10 Folds verwendet. Bei einer Kreuz-Validierung mit 10 Folds wird ein Datensatz in zehn möglichst gleich große Teilmengen $T_1, \ldots, T_{10}$ aufgeteilt. Dann werden zehn Durchläufe gestartet, bei denen die jeweils i-te Teilmenge T_i für die Klassifikation und die verbleibenden neun Teilmengen für das Lernen verwendet werden. In den kommenden Abschnitten findet sich eine genauere Beschreibung der eingesetzten Heuristiken, Datensätze und Konfigurationen.

4.2 Ausgewählte Heuristiken

Die ausgewählten Heuristiken zur Evaluation sind: Correlation, Laplace, m-Estimate, Relative-Linear-Cost (RLC) und Weighted-Relative-Accuracy (WRAcc). Diese Heuristiken werden verwendet, um die Qualität der gefundenen Regeln zu bewerten. Die Qualität einer Regel wird dadurch bewertet, dass ein Regelwert der Regel zugeordnet wird. Die Auswahl verschiedener Heuristiken ermöglicht es, dass man das Verhalten von Lern- und Klassifikationsverfahren abhängig von Heuristiken untersuchen kann. Es kann untersucht werden, ob ein Verfahren beispielsweise unter Verwendung einer bestimmten Heuristik besser als die anderen ist oder ein Verfahren unabhängig von der verwendeten Heuristik das beste Verfahren ist.

Diese fünf Heuristiken haben auch verschiedene Eigenschaften, was den Vorteil hat, dass wir die Lern- und Klassifikationsverfahren auf breiterer Ebene und aus verschiedenen Perspektiven miteinander vergleichen können. Beispielhaft findet man unter Verwendung der Weighted-Relative-Accuracy Heuristik im Prinzip allgemeinere Regelmengen, während die Laplace Heuristik hingegen zur Überanpassung tendiert, wenn keine Pruning-Mechanismen eingesetzt werden. Im nächsten Schritt folgen die Definitionen von drei intuitiv verständlichen Heuristiken, dann erfolgt die Beschreibung der Eigenschaften der verwendeten Heuristiken. Eine ausführliche Beschreibung der vorgestellten Heuristiken und eine Analyse der vordefinierten Werte für parametrisierte Heuristiken sind in [8] verfügbar.

- True-Positive-Rate

$$h_{tpr} = \frac{p}{P} \tag{4.1}$$

- False-Positive-Rate

$$h_{fpr} = \frac{n}{N} \tag{4.2}$$

- Accuracy

$$h_{acc} = p - n \tag{4.3}$$

Die einfache Formel dieser Heuristik kann auch gleich zu $(p + (N - n))/(P + N)$ interpretiert werden, denn P und N sind im Prinzip Konstanten. Durch die zweite Formel wird die Prozentzahl der korrekt klassifizierten Instanzen geteilt durch die Gesamtzahl der Instanzen in den Trainingsdaten berechnet.

- Correlation

$$h_{corr} = \frac{pN - Pn}{\sqrt{PN(p+n)(P-p+N-n)}} \tag{4.4}$$

Diese Heuristik berechnet den Korrelationskoeffizienten zwischen den Vorhersagen, die von der Regel erzeugt werden, und den Ziellabels. Sie wird in Fossil [6], einem Separate-and-Conquer Regellernsystem, verwendet.

- Laplace

$$h_{Lap} = \frac{p+1}{p+n+2} \tag{4.5}$$

Diese Heuristik tendiert zur Überanpassung, wenn keine Pruning-Mechanismen eingesetzt werden. Sie wird im CN2-Algorithmus [3] verwendet.

- m-Estimate

$$h_m = \frac{p + m\frac{P}{P+N}}{p+n+m} \tag{4.6}$$

Die Grundidee dieser Heuristik [5] ist, dass die gefundene Regel apriorisch m Beispiele abdeckt, wobei die Verteilung von Beispielen in den Trainingsdaten auch in Betracht gezogen wird. Diese Heuristik hat einen Parameter m, und dieser Parameter hat im SECO-Framework defaultmäßig den Wert 22,466.

- Relative-Linear-Cost

$$h_{RLC} = c\frac{p}{P} - (1-c)\frac{n}{N} \tag{4.7}$$

Diese Heuristik sucht hinsichtlich gegebener Kosten nach bestmöglichen Punkten. Es ist auch zu beachten, dass der Parameter c in der Relative-Linear-Cost Heuristik im Allgemeinen einen Wert zwischen 0 und 1 und im SECO-Framework defaultmäßig den Wert 0,342 hat.

- Weighted-Relative-Accuracy

$$h_{WRA} = \frac{p}{P} - \frac{n}{N} \tag{4.8}$$

Diese Heuristik berechnet die Differenz zwischen der True-Positive-Rate und der False-Positive-Rate. Die Grundidee dieser Heuristik [13] ist die Berechnung von Accuracy auf einer normalisierten Verteilung von positiven und negativen Beispielen.

4.3 Verwendete Trainingsdaten

Die verwendete Datenmenge besteht aus 30 Datensätzen und sie kann in zwei Gruppen aufgeteilt werden. Die erste Gruppe besteht aus Datensätzen mit binären Klassenattributswerten, d. h. das Klassenattribut hat nur zwei mögliche Werte, z. B. wahr oder falsch. Bei der zweiten Gruppe hat das Klassenattribut mehr als zwei Werte, z. B. hat das Klassenattribut vom Datensatz „Zoo“ folgende Werte: mammal, bird, reptile, fish, amphibian, insect, invertebrate. Die Verwendung dieser Datensätze hat den Vorteil, dass man die vier Verfahren für Datensätze mit binärwertigen und mehrwertigen Klassen getrennt untersuchen kann. Die verwendeten Datensätze haben sowohl nominale als auch numerische Attribute, was zu einem allgemeineren Vergleich führt. Dazu kommt, dass die Anzahl der Instanzen in verschiedenen Datensätzen ausreichend unterschiedlich ist. Die Tabelle 4.1 illustriert die 15 verwendeten Datensätze mit mehr als zwei Klassenattributswerten, wobei die beiden letzten Spalten die Anzahl der nominalen und numerischen Attribute zeigen.

Die Tabelle 4.2 illustriert die 15 verwendeten Datensätze mit zwei Klassenattributswerten.

Tabelle 4.1: Die 15 verwendeten Datensätze mit mehrwertigen Klassen

Datensatz	Dateiname	# Instanzen	# Klassen	# nominale Att	# numerische Att
auto-mpg	auto-mpg.arff	398	4	3	5
autos	autos.arff	205	7	11	15
bridges2	bridges2.arff	108	6	11	1
balance-scale	balance-scale.arff	625	3	1	4
flag	flag.arff	194	6	18	10
hayes-roth	hayes-roth.arff	132	3	5	0
cleveland-14-heart-disease	heart-c.arff	303	5	8	6
hungarian-14-heart-disease	heart-h.arff	294	5	8	6
lymphography	lymph.arff	148	4	16	3
cpu	machine.arff	209	8	1	7
primary-tumor	primary-tumor.arff	339	22	18	0
segment	segment.arff	2310	7	1	19
flare	solar-flare.arff	333	8	11	0
vehicle	vehicle.arff	846	4	1	18
zoo	zoo.arff	101	7	17	1

Tabelle 4.2: Die 15 verwendeten Datensätze mit binärwertigen Klassen

Datensatz	Dateiname	# Instanzen	# nominale Att	# numerische Att
ballons	balloons.arff	76	5	0
wisconsin-breast-cancer-weka.filters.supervised.attribute.Discretize-R1-9	breast-w-d.arff	699	10	0
wisconsin-breast-cancer	breast-w.arff	699	1	9
horse-colic	colic.arff	368	16	7
horse-colic.ORIG	colic.ORIG.arff	368	22	7
credit-rating	credit-a.arff	690	10	6
german_credit	credit-g.arff	1000	14	7
pima_diabetes	diabetes.arff	768	1	8
echo	echocardiogram.arff	132	2	7
heart-statlog	heart-statlog.arff	270	1	13
house-votes	house-votes-84.arff	435	16	0
ionosphere	ionosphere.arff	351	1	33
labor-weka.filters.supervised.attribute.Discretize-R1-16	labor-d.arff	57	17	0
promoters	promoters.arff	106	58	0
sonar	sonar.arff	208	1	60

4.4 Ausgewählte Konfigurationen im SECO-Framework

Die fünf genannten Heuristiken und die vier Kombinationen von Lern- und Klassifikationsverfahren ergeben insgesamt 20 Konfigurationen. Die ersten vier Konfigurationen sehen folgendermaßen aus:

- Correlation Heuristik
  1. Lernen durch Defaultlernverfahren und Klassifizieren durch Defaultklassifizierungsverfahren
  2. Lernen durch Multi-Class-Covering-Verfahren und Klassifizieren durch Defaultklassifizierungsverfahren

3. Lernen durch Majority-Voting-Verfahren und Klassifizieren durch Majority-Voting-Verfahren
4. Lernen durch Majority-Voting-Verfahren und Klassifizieren durch gewichtetes-Majority-Voting-Verfahren

Die restlichen sechzehn Konfigurationen bilden sich dadurch, dass man die Correlation Heuristik durch die anderen vier Heuristiken ersetzt.

Die im SECO-Framework vordefinierten Komponenten ergänzen die ausgewählten Konfigurationen bei der Evaluation und werden bei allen 20 dargestellten Konfigurationen eingesetzt.

Tabelle 4.3: Die Dafaultkonfigurationen

Name	Verwendete Komponente
Rule Evaluator	RuleEvaluator
Rule Refiner	TopDownRefiner
Rule Filter	BeamWidthFilter with a beam size of 1
Stopping Criterion	NoNegativesCoveredStop
Rule Stopping Criterion	CoverageRuleStop
Post Processor	NoOpPostProcessor
Reduced Error Pruning	not used
Minimum Number of examples a rule has to cover	1
Weighted Covering	not used

Im Folgenden befindet sich eine Überblick gebende Erklärung der Defaultkomponenten aus [9].

- Rule Evaluator: Die Qualität einer Regel wird in der Komponente RuleEvaluator bewertet. Die Suchheuristik muss zuvor gesetzt sein und sie ist essenziell bei der Evaluierung der Regel. In dieser Arbeit werden fünf Heuristiken verwendet, die im vorigen Abschnitt vorgestellt wurden. Je nachdem, welche Heuristik eingesetzt wird, bekommt die Regel einen unterschiedlichen Regelwert zugewiesen.

- Rule Refiner: Eine Regel wird durch einen Verfeinerungsoperator verfeinert. Je nachdem, welche Suchstrategie verwendet wird, können die Verfeinerungsoperatoren unterschiedlich sein. Für die Top-Down-Strategie wird eine Spezialisierung einer Regel durch Hinzufügen einer Bedingung zur Regel implementiert.

- Rule Filter: Der Regel-Filter entscheidet, wie viele Regeln in jedem Zyklus verfeinert werden. Wenn eine Beam-Suche verwendet wird, dann liefert er die b besten Regeln, wobei b die Größe des Beams ist.

- Stopping Criterion: In der Komponente NoNegativesCoveredStop wird das Hinzufügen der Bedingungen zu einer Regel abgebrochen, wenn kein negatives Beispiel von der Regel abgedeckt wird. Mit anderen Worten, es werden bei einer Top-Down-Suchstrategie solange Bedingungen zu einer Regel hinzugefügt, bis die Regel kein negatives Beispiel abdeckt.

- Rule Stopping Criterion: Die Komponente CoverageRuleStop prüft, ob eine gegebene Regel mehr negative Beispiele als positive Beispiele abdeckt. Deckt eine Regel mehr negative Beispiele ab, wird die Regel verworfen und zur Regelmenge nicht hinzugefügt.

- Post Processor: Der Post-Processor setzt alle zusätzlichen Schritte um, die nach dem Lernen einer Theorie zur Optimierung ausgeführt werden. Bei der Komponente NoOpPostProcessor findet keine Post-Prozessphase statt.

- Reduced Error Pruning: Reduced-Error-Pruning ist ein Pruning-Mechanismus zur Vereinfachung der gelernten Regelmenge. Es wird aber hier nicht verwendet.

- Weighted Covering: Wird Weighted-Covering verwendet, werden die Gewichte der Beispiele bei der Aktualisierung der Gewichte begrenzt verringert. Andernfalls werden die Gewichte der Beispiele bei der Aktualisierung auf Null gesetzt. Auch Weighted-Covering wird hier nicht verwendet.

4.5 Bestimmung der prozentualen Anzahl der nicht abgedeckten Instanzen für das Multi-Class-Covering-Verfahren

Wie in 3.1.3 beschrieben, hat das Multi-Class-Covering-Verfahren einen Parameter, der angibt, wie lange die äußere Schleife des Verfahrens fortgesetzt wird. Um einen zweckmäßigen Wert für diesen Parameter bestimmen zu können, wurden fünf Werte für diesen Parameter ausgewählt, und die Ergebnisse in folgenden Tabellen illustriert.

Die fünf ausgewählten Werte sind: 80%, 85%, 90%, 95%, 100%. Beispielsweise bedeutet ein Parameterwert vom 80%, dass die äußere Schleife des Verfahrens solange fortgesetzt wird, bis 80% der Beispiele von Trainingsdaten abgedeckt werden.

Die Tabelle 4.4 illustriert die durchschnittlichen Genauigkeiten (Macro AVG) des Multi-Class-Covering-Verfahrens unter Verwendung verschiedener Parameterwerte und Heuristiken. Hierbei wurden alle 30 Datensätze verwendet. Es ist unabhängig von der verwendeten Heuristik Folgendes erkennbar:

Je größer der Parameter ist, desto besser ist die durchschnittliche Genauigkeit des Multi-Class-Covering-Verfahrens.

Tabelle 4.4: Das Multi-Class-Covering-Verfahren und seine durchschnittliche Genauigkeit

Heuristik	Macro AVG				
	80%	85%	90%	95%	100%
Correlation	77,818	78,260	78,469	78,515	78,519
Laplace	76,310	76,413	77,282	77,348	77,616
m-Estimate	76,186	77,089	77,429	78,160	78,347
RLC	76,485	76,717	76,922	77,025	77,186
WRAcc	76,358	76,669	76,771	77,079	77,197

Die Tabelle 4.5 illustriert die durchschnittlichen Anzahl der gelernten Regeln (AVG # of Rules) des Multi-Class-Covering-Verfahrens unter Verwendung verschiedener Parameterwerte und Heuristiken. Hierbei wurden alle 30 Datensätze verwendet. Es ist unabhängig von der verwendeten Heuristik Folgendes erkennbar:

Je kleiner der Parameter ist, desto geringer ist die durchschnittliche Anzahl der gelernten Regeln des Multi-Class-Covering-Verfahrens.

Tabelle 4.5: Das Multi-Class-Covering-Verfahren und die durchschnittliche Anzahl der Regeln

Heuristik	AVG # of Rules				
	80%	85%	90%	95%	100%
Correlation	7,166	7,766	8,166	8,753	9,633
Laplace	30,066	34,000	38,000	42,333	47,733
m-Estimate	9,933	11,466	13,233	15,400	19,166
RLC	8,766	10,166	12,133	13,733	45,100
WRAcc	3,566	4,033	4,333	5,033	6,400

Die Tabelle 4.6 illustriert die durchschnittliche Anzahl der gelernten Bedingungen (AVG # of Conditions) des Multi-Class-Covering-Verfahrens unter Verwendung verschiedener Parameterwerte und Heuristiken. Hierbei wurde alle 30 Datensätze verwendet. Es ist unabhängig von der verwendeten Heuristik Folgendes erkennbar:

Je kleiner der Parameter ist, desto geringer ist die durchschnittliche Anzahl der gelernten Bedingungen des Multi-Class-Covering-Verfahrens.

Tabelle 4.6: Das Multi-Class-Covering-Verfahren und die durchschnittliche Anzahl der Bedingungen

Heuristik	AVG # of Conditions				
	80%	85%	90%	95%	100%
Correlation	22,800	24,466	26,133	27,466	29,000
Laplace	59,933	67,033	74,066	81,400	88,666
m-Estimate	30,166	33,966	38,600	43,166	48,700
RLC	23,200	26,000	29,700	32,633	117,300
WRAcc	12,733	14,100	15,000	16,700	19,033

Bevor wir mit dem Vergleich zwischen verschiedenen Lern- und Klassifikationsverfahren anfangen, müssen wir uns für einen Parameter für das Multi-Class-Covering-Verfahren entscheiden. Hierbei wurden drei Entschei-

dungskriterien, nämlich durchschnittliche Genauigkeit, durchschnittliche Anzahl der Regeln und durchschnittliche Anzahl der Bedingungen, vorgeführt. Wir haben die durchschnittliche Genauigkeit als das wichtigste Kriterium festgelegt, weil wir in erster Linie überprüfen wollten, wie präzise die gefundenen Regelmengen des Multi-Class-Covering-Verfahrens im Vergleich zu anderen Lern- und Klassifikationsverfahren sind. Infolgedessen wurde der Parameter des Multi-Class-Covering-Verfahrens auf 100% gesetzt.

4.6 Ergebnisse der Evaluation

Im Folgenden werden zuerst die Evaluationsergebnisse bei allen verwendeten Heuristiken für die Datensätze mit binärwertigen Klassen untersucht, danach kommen die Ergebnisse der Datensätze mit mehrwertigen Klassen hinzu.

Bei jeder Heuristik werden zuerst die bei der Kreuz-Validierung korrekt klassifizierten Instanzen der vier in 4.1 aufgeführten Lern- und Klassifikationsverfahren durch eine Win-Loss-Tie-Analyse tabellarisch dargestellt. Hierbei und im Vergleich zwischen dem Defaultverfahren und dem Majority-Voting-Verfahren werden die erste und dritte aufgeführte Kombination in 4.1 verglichen. Dabei ist zu beachten, dass das Defaultlernverfahren hinsichtlich der Anzahl der gelernten Regeln und Bedingungen fast in allen Fällen weniger Regeln und Bedingungen lernt. Der Grund ist, dass man beim Majority-Voting-Lernverfahren sehr ähnlich zum Defaultlernverfahren vorgeht und für verschiedene Klassenattributswerte der Reihe nach, Regeln lernt. Der Unterschied ist, dass man beim Majority-Voting-Lernverfahren auch für den Klassenattributswert mit der höchsten Anzahl an Instanzen in den Trainingsdaten Regeln lernt.

Beim Vergleich zwischen dem Majority-Voting-Verfahren und dem gewichteten-Majority-Voting-Verfahren werden die zweite und dritte aufgeführte Kombination in 4.1 verglichen. Dabei ist zu beachten, dass beide Kombinationen das Majority-Voting-Verfahren als Lernverfahren verwenden, was zur Folge hat, dass beide Verfahren die gleiche Theorie lernen.

Nach der Win-Loss-Tie-Analyse werden die vier Lern- und Klassifikationsverfahren hinsichtlich der 4 Kriterien, nämlich durchschnittliche Genauigkeit (Macro Average), durchschnittlicher Rank (Average Rank), durchschnittliche Anzahl der Regeln (Average Number of Rules), und durchschnittliche Anzahl der Bedingungen (Average Number of Conditions), miteinander verglichen.

Die durchschnittliche Genauigkeit bezeichnet die korrekt klassifizierten Instanzen durch die gesamten klassifizierten Instanzen. Die durchschnittliche Genauigkeit wird in Prozent angegeben. Jede Konfiguration im SECO-Framework bekommt ein Rank zugewiesen. Hierbei besagt der kleinere Rank einer Konfiguration, dass die Konfiguration den anderen Konfigurationen hinsichtlich des Rankes vorgezogen wird. Bei der durchschnittlichen Anzahl der gelernten Regeln und Bedingungen wird eine geringere Anzahl an Regeln und Bedingungen erwünscht und vorgezogen.

4.6.1 Datensätze mit binärwertigen Klassen und Correlation Heuristik

Im Folgenden werden die Vergleichsergebnisse der 15 untersuchten binärwertigen Datensätze erläutert. Die Tabelle 4.7 illustriert die Win-Loss-Tie-Analyse. Dabei sind die eingetragenen Zahlen folgendermaßen zu interpretieren. Das Defaultverfahren findet im Vergleich zum Multi-Class-Covering-Verfahren bei der Kreuz-Validierung von sieben Datensätzen Regelmengen mit einer höheren Prozentzahl von korrekt klassifizierten Instanzen. Und das Defaultverfahren verliert gegen das Multi-Class-Covering-Verfahren bei fünf Datensätzen. Bei den verbleibenden drei Datensätzen liefern beide Verfahren ein gleiches Ergebnis. Daher wurde in die Zelle liegend auf der zweiten Zeile und dritten Spalte 7-5-3 eingetragen. Analog dazu sind alle anderen Zellen der folgenden Tabelle zu interpretieren.

Gemäß der Win-Loss-Tie-Tabelle gewinnt das gewichtete-Majority-Voting-Verfahren insgesamt alle drei Vergleiche gegen die anderen Verfahren, und ist somit das stärkste Verfahren. Das Majority-Voting-Verfahren ist hierbei das zweitstärkste Verfahren, wobei das Majority-Voting-Verfahren gegen das Defaultverfahren und das Multi-Class-Covering-Verfahren gewinnt und gegen das gewichtete-Majority-Voting-Verfahren knapp verliert. Hierbei verliert das Multi-Class-Covering-Verfahren gegen die drei anderen Verfahren und ist somit das schwächste Verfahren.

Die Tabelle 4.8 beinhaltet die vier Vergleichskriterien.

Hinsichtlich der durchschnittlichen Genauigkeit liefert das gewichtete-Majority-Voting-Verfahren das beste Ergebnis im Vergleich zu den anderen drei Verfahren. Das schlechteste Ergebnis weist hierbei das Multi-Class-Covering-Verfahren auf. Allerdings sind die vier verschiedenen Verfahren hinsichtlich der durchschnittlichen Genauigkeit nicht sehr unterschiedlich.

Bei der Untersuchung des durchschnittlichen Rankes zeigt das gewichtete-Majority-Voting-Verfahren wieder seine Stärke, während das Multi-Class-Covering-Verfahren als das schlechteste Verfahren eingestuft wird.

Tabelle 4.7: Win-Loss-Tie-Analyse für Datensätze mit binärwertigen Klassen und der Correlation Heuristik

Correlation	Default	Multi-Class-Covering	Majority-Voting	gewichtetes-Majority-Voting
Default	x	8-5-2	4-9-2	5-9-1
Multi-Class-Covering	5-8-2	x	5-9-1	3-12-0
Majority-Voting	9-4-2	9-5-1	x	6-7-2
gewichtetes-Majority-Voting	9-5-1	12-3-0	7-6-2	x

Tabelle 4.8: Ergebnisse der Evaluation für Datensätze mit binärwertigen Klassen und der Correlation Heuristik

Kriterium	Default	Multi-Class-Covering	Majority-Voting	gewichtetes-Majority-Voting
Macro AVG	82,216	81,669	82,604	83,172
AVG Rank	2,700	3,033	2,233	2,033
AVG # of rules	12,066	8,600	23,733	23,733
AVG # of conditions	38,400	29,466	73,400	73,400

Hinsichtlich der durchschnittlichen Anzahl der gelernten Regeln und Bedingungen ist das Multi-Class-Covering-Verfahren das deutlich bessere Verfahren. Das gewichtete-Majority-Voting-Verfahren und das Majority-Voting-Verfahren lernen hingegen Regelmengen mit der größten Anzahl der Regeln und Bedingungen. Wie auch vorher beschrieben, verwenden das gewichtete-Majority-Voting-Verfahren und das Majority-Voting-Verfahren dasselbe Lernprinzip, daher weisen diese zwei Verfahren hinsichtlich der durchschnittlichen Anzahl der gelernten Regeln und Bedingungen gleiche Ergebnisse auf.

Fazit ist, wenn man nach Verfahren mit der höchsten durchschnittlichen Genauigkeit und einem besseren durchschnittlichen Rank sucht, dann ist das gewichtete-Majority-Voting-Verfahren empfehlenswert. Wenn die durchschnittliche Anzahl der gelernten Regeln und Bedingungen von großer Bedeutung ist, dann sollte das Multi-Class-Covering-Verfahren verwendet werden, wobei man auch das Defaultverfahren als den zweiten Kandidaten in Betracht ziehen kann.

4.6.2 Datensätze mit binärwertigen Klassen und Laplace Heuristik

Im Folgenden werden die Vergleichsergebnisse der 15 untersuchten binärwertigen Datensätze erläutert.

Gemäß der Win-Loss-Tie-Tabelle 4.9 gewinnt das gewichtete-Majority-Voting-Verfahren insgesamt alle drei Vergleiche gegen die anderen Verfahren, und ist somit das stärkste Verfahren. Das Multi-Class-Covering-Verfahren verliert gegen die anderen drei Verfahren und ist das schwächste Verfahren. Es ist auch zu beachten, dass das Defaultverfahren dem Multi-Class-Covering-Verfahren sehr nahe liegt und es nicht stark spürbar gegen das Multi-Class-Covering-Verfahren gewinnen kann.

Tabelle 4.9: Win-Loss-Tie-Analyse für Datensätze mit binärwertigen Klassen und der Laplalce Heuristik

Laplace	Default	Multi-Class-Covering	Majority-Voting	gewichtetes-Majority-Voting
Default	x	8-7-0	2-9-4	1-11-3
Multi-Class-Covering	7-8-0	x	4-11-0	4-11-0
Majority-Voting	9-2-4	11-4-0	x	1-11-3
gewichtetes-Majority-Voting	11-1-3	11-4-0	11-1-3	x

Tabelle 4.10: Ergebnisse der Evaluation für Datensätze mit binärwertigen Klassen und der Laplace Heuristik

Kriterium	Default	Multi-Class-Covering	Majority-Voting	gewichtetes-Majority-Voting
Macro AVG	79,642	80,324	81,246	82,137
AVG Rank	3,033	3,000	2,367	1,600
AVG # of rules	40,266	39,800	84,800	84,800
AVG # of conditions	97,133	69,733	200,400	200,400

Hinsichtlich der durchschnittlichen Genauigkeit liefert das gewichtete-Majority-Voting-Verfahren das beste Ergebnis im Vergleich zu den anderen drei Verfahren. Das schlechteste Ergebnis hierbei zeigt das Defaultverfahren auf. Allerdings sind die vier verschiedenen Verfahren hinsichtlich der durchschnittlichen Genauigkeit nicht sehr unterschiedlich.

Dem durchschnittlichen Rank zufolge wird das gewichtete-Majority-Voting-Verfahren als das beste Verfahren eingestuft. Und das Defaultverfahren liefert hierbei das schlechteste Ergebnis.

Hinsichtlich der durchschnittlichen Anzahl der gelernten Regeln und Bedingungen ist das Multi-Class-Covering-Verfahren das beste Verfahren, wobei das Defaultverfahren auch Regelmengen mit der fast gleichen Anzahl an Regeln findet. Das gewichtete-Majority-Voting-Verfahren und das Majority-Voting-Verfahren lernen hingegen Regelmengen mit der größten Anzahl der Regeln und Bedingungen.

Fazit ist, wenn man nach Verfahren mit der höchsten durchschnittlichen Genauigkeit und einem besseren durchschnittlichen Rank sucht, dann ist das gewichtete-Majority-Voting-Verfahren empfehlenswert. Wenn die durchschnittliche Anzahl der gelernten Regeln und Bedingungen von großer Bedeutung ist, dann sollte das Multi-Class-Covering-Verfahren verwendet werden.

4.6.3 Datensätze mit binärwertigen Klassen und m-Estimate Heuristik

Im Folgenden werden die Vergleichsergebnisse der 15 untersuchten binärwertigen Datensätze erläutert.

Gemäß der Win-Loss-Tie-Tabelle 4.11 gewinnt das gewichtete-Majority-Voting-Verfahren insgesamt alle drei Vergleiche gegen die anderen Verfahren, und ist somit das stärkste Verfahren. Hierbei verliert das Multi-Class-Covering-Verfahren gegen die anderen drei Verfahren und ist das schwächste Verfahren. Es ist auch zu beachten, dass das Defaultverfahren dem Multi-Class-Covering-Verfahren sehr nahe liegt und es nicht stark spürbar gegen das Multi-Class-Covering-Verfahren gewinnen kann.

Tabelle 4.11: Win-Loss-Tie-Analyse für Datensätze mit binärwertigen Klassen und der m-Estimate Heuristik

m-Estimate	Default	Multi-Class-Covering	Majority-Voting	gewichtetes-Majority-Voting
Default	x	8-7-0	2-12-1	1-14-0
Multi-Class-Covering	7-8-0	x	3-10-2	3-11-1
Majority-Voting	12-2-1	10-3-2	x	5-7-3
gewichtetes-Majority-Voting	14-1-0	11-3-1	7-5-3	x

Hinsichtlich der durchschnittlichen Genauigkeit liefert das gewichtete-Majority-Voting-Verfahren das beste Ergebnis im Vergleich zu den anderen drei Verfahren. Das schlechteste Ergebnis zeigt hierbei das Multi-Class-Covering-Verfahren auf. Allerdings sind die vier verschiedenen Verfahren hinsichtlich der durchschnittlichen Genauigkeit nicht sehr unterschiedlich.

Dem durchschnittlichen Rank zufolge wird das gewichtete-Majority-Voting-Verfahren als das beste Verfahren eingestuft. Und das Defaultverfahren liefert hierbei das schlechteste Ergebnis.

Hinsichtlich der durchschnittlichen Anzahl der gelernten Regeln und Bedingungen ist das Defaultverfahren das deutlich bessere Verfahren. Das gewichtete-Majority-Voting-Verfahren und das Majority-Voting-Verfahren lernen hingegen Regelmengen mit der größten Anzahl der Regeln und Bedingungen.

Fazit ist, wenn man nach Verfahren mit der höchsten durchschnittlichen Genauigkeit und einem besseren durchschnittlichen Rank sucht, dann ist das gewichtete-Majority-Voting-Verfahren empfehlenswert. Wenn die

Tabelle 4.12: Ergebnisse der Evaluation für Datensätze mit binärwertigen Klassen und der m-Estimate Heuristik

Kriterium	Default	Multi-Class-Covering	Majority-Voting	gewichtetes-Majority-Voting
Macro AVG	81,529	81,519	82,654	83,001
AVG Rank	3,233	3,033	2,000	1,733
AVG # of rules	10,133	19,066	23,600	23,600
AVG # of conditions	37,466	48,266	85,666	85,666

durchschnittliche Anzahl der gelernten Regeln und Bedingungen von großer Bedeutung ist, dann sollte das Defaultverfahren verwendet werden.

4.6.4 Datensätze mit binärwertigen Klassen und Relative-Linear-Cost Heuristik

Im Folgenden werden die Vergleichsergebnisse der 15 untersuchten binärwertigen Datensätze erläutert.

Gemäß der Win-Loss-Tie-Tabelle 4.13 gewinnt das gewichtete-Majority-Voting-Verfahren insgesamt alle drei Vergleiche gegen die anderen Verfahren, und ist das somit stärkste Verfahren. Und mit drei Verlusten stellt sich das Defaultverfahren als das schwächste Verfahren dar.

Tabelle 4.13: Win-Loss-Tie-Analyse für Datensätze mit binärwertigen Klassen und der Relative-Linear-Cost Heuristik

RLC	Default	Multi-Class-Covering	Majority-Voting	gewichtetes-Majority-Voting
Default	x	5-9-1	1-12-2	3-10-2
Multi-Class-Covering	9-5-1	x	7-7-1	6-9-0
Majority-Voting	12-1-2	7-7-1	x	5-8-2
gewichtetes-Majority-Voting	10-3-2	9-6-0	8-5-2	x

Tabelle 4.14: Ergebnisse der Evaluation für Datensätze mit binärwertigen Klassen und der Relative-Linear-Cost Heuristik

Kriterium	Default	Multi-Class-Covering	Majority-Voting	gewichtetes-Majority-Voting
Macro AVG	80,868	81,705	82,320	82,525
AVG Rank	3,233	2,467	2,233	2,067
AVG # of rules	6,000	17,933	20,866	20,866
AVG # of conditions	19,466	38,400	59,400	59,400

Hinsichtlich der durchschnittlichen Genauigkeit liefert das gewichtete-Majority-Voting-Verfahren das beste Ergebnis im Vergleich zu den anderen drei Verfahren. Das schlechteste Ergebnis zeigt hierbei das Defaultverfahren auf. Allerdings sind die vier verschiedenen Verfahren hinsichtlich der durchschnittlichen Genauigkeit nicht sehr unterschiedlich.

Dem durchschnittlichen Rank zufolge wird das gewichtete-Majority-Voting-Verfahren als das beste Verfahren eingestuft. Und das Defaultverfahren liefert hierbei wieder das schlechteste Ergebnis.

Hinsichtlich der durchschnittlichen Anzahl der gelernten Regeln und Bedingungen ist das Defaultverfahren das deutlich bessere Verfahren. Das gewichtete-Majority-Voting-Verfahren und das Majority-Voting-Verfahren lernen hingegen Regelmengen mit der größten Anzahl der Regeln und Bedingungen.

Fazit ist, wenn man nach Verfahren mit der höchsten durchschnittlichen Genauigkeit und einem besseren

durchschnittlichen Rank sucht, dann ist das gewichtete-Majority-Voting-Verfahren empfehlenswert. Wenn die durchschnittliche Anzahl der gelernten Regeln und Bedingungen von großer Bedeutung ist, dann sollte das Defaultverfahren verwendet werden.

4.6.5 Datensätze mit binärwertigen Klassen und Weighted-Relative-Accuracy Heuristik

Im Folgenden werden die Vergleichsergebnisse der 15 untersuchten binärwertigen Datensätze erläutert.

Gemäß der Win-Loss-Tie-Tabelle 4.15 gewinnt das gewichtete-Majority-Voting-Verfahren insgesamt alle drei Vergleiche gegen die anderen Verfahren, und ist somit das stärkste Verfahren. Und mit drei Verlusten stellt sich das Defaultverfahren als das schwächste Verfahren dar.

Tabelle 4.15: Win-Loss-Tie-Analyse für Datensätze mit binärwertigen Klassen und der Weighted-Relative-Accuracy Heuristik

WRAcc	Default	Multi-Class-Covering	Majority-Voting	gewichtetes-Majority-Voting
Default	x	5-7-3	2-9-4	5-8-2
Multi-Class-Covering	7-5-3	x	5-6-4	6-8-1
Majority-Voting	9-2-4	6-5-4	x	5-8-2
gewichtetes-Majority-Voting	8-5-2	8-6-1	8-5-2	x

Tabelle 4.16: Ergebnisse der Evaluation für Datensätze mit binärwertigen Klassen und der Weighted-Relative-Accuracy Heuristik

Kriterium	Default	Multi-Class-Covering	Majority-Voting	gewichtetes-Majority-Voting
Macro AVG	81,608	82,090	81,980	82,007
AVG Rank	2,900	2,533	2,333	2,233
AVG # of rules	1,933	4,266	5,866	5,866
AVG# of conditions	6,333	14,466	19,600	19,600

Hinsichtlich der durchschnittlichen Genauigkeit liefert das Multi-Class-Covering-Verfahren das beste Ergebnis im Vergleich zu den anderen drei Verfahren. Das schlechteste Ergebnis zeigt hierbei das Defaultverfahren auf. Es ist auch zu beachten, dass der Unterschied zwischen dem besten und dem schlechtesten Verfahren hinsichtlich der durchschnittlichen Genauigkeit sehr klein ist und nur bei 0,482 Prozent liegt.

Dem durchschnittlichen Rank zufolge wird das gewichtete-Majority-Voting-Verfahren als das beste Verfahren eingestuft. Und das Defaultverfahren liefert hierbei das schlechteste Ergebnis.

Hinsichtlich der durchschnittlichen Anzahl der gelernten Regeln und Bedingungen ist das Defaultverfahren das deutlich bessere Verfahren. Das gewichtete-Majority-Voting-Verfahren und das Majority-Voting-Verfahren lernen hingegen Regelmengen mit der größten Anzahl der Regeln und Bedingungen. Es ist auch bemerkenswert, dass alle vier Verfahren unter Verwendung der Weighted-Relative-Accuracy Heuristik Regelmengen mit geringer Anzahl an Regeln und Bedingungen zurückliefern.

Fazit ist, wenn man nach Verfahren mit der höchsten durchschnittlichen Genauigkeit sucht, dann ist das Multi-Class-Covering-Verfahren empfehlenswert. Ist der durchschnittliche Rank das wichtigste Entscheidungskriterium, dann ist das gewichtete-Majority-Voting-Verfahren zu empfehlen. Wenn die durchschnittliche Anzahl der gelernten Regeln und Bedingungen von großer Bedeutung ist, dann sollte das Defaultverfahren verwendet werden.

4.6.6 Datensätze mit mehrwertigen Klassen und Correlation Heuristik

Ein interessantes Beispiel beim Vergleich zwischen den verschiedenen Lern- und Klassifikationsverfahren liefert uns die Datei primary-tumor.arff mit 22 Werten für das Klassenattribut. Bei der Kreuz-Validierung dieses Datensatzes sinkt die Prozentzahl der korrekt klassifizierten Instanzen durch das Defaultverfahren drastisch. Hierbei lernt das Defaultlernverfahren 64 Regeln mit insgesamt 317 Bedingungen, während das Multi-Class-Covering-Lernverfahren nur 6 Regeln mit insgesamt 21 Bedingungen lernt. Das Majority-Voting-Lernverfahren und das gewichtete-Majority-Voting-Lernverfahren lernen beide 60 Regeln mit 311 Bedingungen.

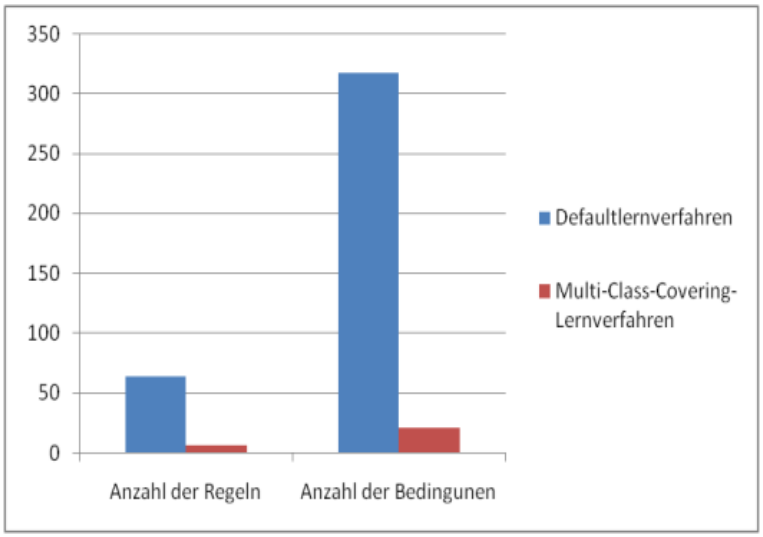

Abbildung 4.1: Anzahl der Regeln und Bedingungen beim primary-tumor Datensatz

Obwohl die korrekt klassifizierten Instanzen beim Defaultverfahren viel besser als beim Multi-Class-Covering-Verfahren sind, weisen beide Verfahren bei der Kreuz-Validierung einen viel kleineren Unterschied von 2,36 Prozent auf. Die korrekt klassifizierten Instanzen bei der Kreuz-Validierung liegen für das Defaultverfahren und das Multi-Class-Covering-Verfahren jeweils bei 36,28 und 33,92 Prozent. Hierbei ist zu beachten, dass beide Verfahren deutlich bessere Ergebnisse als ein randomisiertes Verfahren liefern. Das Majority-Voting-Verfahren und das gewichtete-Majority-Voting-Verfahren zeigen hier beide 60,47 und 39,82 Prozent jeweils für die korrekt klassifizierten Instanzen und für die korrekt klassifizierten Instanzen bei der Kreuz-Validierung auf.

Folgende Abbildung dient auch zur Veranschaulichung des Unterschiedes zwischen dem Defaultverfahren und dem Multi-Class-Covering-Verfahren. Hierbei werden durchschnittliche Werte für die korrekt klassifizierten Instanzen bei der Kreuz-Validierung angegeben.

Im Folgenden werden die Vergleichsergebnisse der 15 untersuchten mehrwertigen Datensätze erläutert.

Gemäß der Win-Loss-Tie-Tabelle 4.17 gewinnt das gewichtete-Majority-Voting-Verfahren insgesamt alle drei Wettkämpfe gegen die anderen Verfahren, und ist somit das stärkste Verfahren. Im Gegensatz dazu stellt sich das Majority-Voting-Verfahren mit drei Verlusten als das schwächste Verfahren dar.

Tabelle 4.17: Win-Loss-Tie-Analyse für Datensätze mit mehrwertigen Klassen und der Correlation Heuristik

Correlation	Default	Multi-Class-Covering	Majority-Voting	gewichtetes-Majority-Voting
Default	x	4-11-0	9-5-1	6-8-1
Multi-Class-Covering	11-4-0	x	10-5-0	5-8-2
Majority-Voting	5-9-1	5-10-0	x	4-9-2
gewichtetes-Majority-Voting	8-6-1	8-5-2	9-4-2	x

Die Tabelle 4.18 beinhaltet die vier Vergleichskriterien.

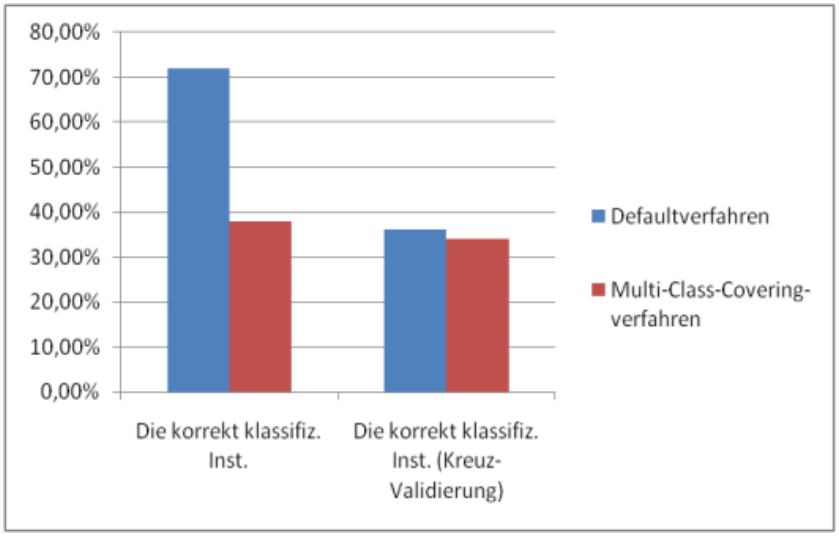

Abbildung 4.2: Die korrekt klassifizierten Instanzen beim primary-tumor Datensatz

Tabelle 4.18: Ergebnisse der Evaluation für Datensätze mit mehrwertigen Klassen und der Correlation Heuristik

Kriterium	Default	Multi-Class-Covering	Majority-Voting	gewichtetes-Majority-Voting
Macro AVG	75,058	75,370	74,854	75,978
AVG Rank	2,800	2,200	2,900	2,100
AVG # of rules	19,266	10,666	28,200	28,200
AVG # of conditions	60,333	28,533	93,933	93,933

Hinsichtlich der durchschnittlichen Genauigkeit liefert das gewichtete-Majority-Voting-Verfahren das beste Ergebnis im Vergleich zu den anderen drei Verfahren. Das Majority-Voting-Verfahren weist hingegen das schlechteste Ergebnis auf. Allerdings sind die vier verschiedenen Verfahren hinsichtlich der durchschnittlichen Genauigkeit nicht sehr unterschiedlich.

Bei der Untersuchung des durchschnittlichen Rankes zeigt das gewichtete-Majority-Voting-Verfahren wieder seine Stärke und wird als das beste Verfahren eingestuft. Das Majority-Voting-Verfahren liefert hierbei wieder das schlechteste Ergebnis.

Hinsichtlich der durchschnittlichen Anzahl der gelernten Regeln und Bedingungen ist das Multi-Class-Covering-Verfahren mit Abstand das deutlich bessere Verfahren. Das gewichtete-Majority-Voting-Verfahren und das Majority-Voting-Verfahren lernen hingegen Regelmengen mit der größten Anzahl der Regeln und Bedingungen.

Fazit ist, wenn man nach Verfahren mit der höchsten durchschnittlichen Genauigkeit und einem besseren durchschnittlichen Rank sucht, dann ist das gewichtete-Majority-Voting-Verfahren empfehlenswert. Wenn die durchschnittliche Anzahl der gelernten Regeln und Bedingungen von großer Bedeutung ist, dann ist die Verwendung des Multi-Class-Covering-Verfahrens zu empfehlen.

4.6.7 Datensätze mit mehrwertigen Klassen und Laplace Heuristik

Im Folgenden werden die Vergleichsergebnisse der 15 untersuchten mehrwertigen Datensätze erläutert.

Gemäß der Win-Loss-Tie-Tabelle 4.19 sind das gewichtete-Majority-Voting-Verfahren und das Defaultverfahren jeweils mit zwei Gewinnen die stärksten Verfahren. Hierbei stellt sich das Majority-Voting-Verfahren mit zwei Verlusten und einem Gleichstand gegen das Multi-Class-Covering-Verfahren als das schwächste Verfahren dar.

Hinsichtlich der durchschnittlichen Genauigkeit liefert das Multi-Class-Covering-Verfahren das beste Ergebnis im Vergleich zu allen anderen drei Verfahren. Es ist auch zu beachten, dass der Unterschied zwischen dem gewichteten-Majority-Voting-Verfahren und dem Defaultverfahren sehr klein ist. Hierbei weist das Majority-

Tabelle 4.19: Win-Loss-Tie-Analyse für Datensätze mit mehrwertigen Klassen und der Laplace Heuristik

Laplace	Default	Multi-Class-Covering	Majority-Voting	gewichtetes-Majority-Voting
Default	x	7-8-0	8-6-1	8-7-0
Multi-Class-Covering	8-7-0	x	7-7-1	6-9-0
Majority-Voting	6-8-1	7-7-1	x	3-10-2
gewichtetes-Majority-Voting	7-8-0	9-6-0	10-3-2	x

Tabelle 4.20: Ergebnisse der Evaluation für Datensätze mit mehrwertigen Klassen und der Laplace Heuristik

Kriterium	Default	Multi-Class-Covering	Majority-Voting	gewichtetes-Majority-Voting
Macro AVG	74,326	74,908	73,171	74,383
AVG Rank	2,433	2,567	2,800	2,200
AVG # of rules	49,933	55,666	96,066	96,066
AVG # of conditions	137,466	107,600	267,0	267,0

Voting-Verfahren das schlechteste Ergebnis auf. Allerdings sind die vier verschiedenen Verfahren hinsichtlich der durchschnittlichen Genauigkeit nicht sehr unterschiedlich.

Bei der Untersuchung des durchschnittlichen Rankes zeigt das gewichtete-Majority-Voting-Verfahren seine Stärke. Und das Majority-Voting-Verfahren liefert wieder das schlechteste Ergebnis.

Hinsichtlich der durchschnittlichen Anzahl der gelernten Regeln liefert das Defaultverfahren das beste Ergebnis. Es verliert aber hinsichtlich der durchschnittlichen Anzahl der gelernten Bedingungen gegen das Multi-Class-Covering-Verfahren und das Multi-Class-Covering-Verfahren ist hierbei als das beste Verfahren zu sehen. Das gewichtete-Majority-Voting-Verfahren und das Majority-Voting-Verfahren lernen hingegen Regelmengen mit der größten Anzahl der Regeln und Bedingungen.

Fazit ist, wenn man nach Verfahren mit der höchsten durchschnittlichen Genauigkeit und der minimalen Anzahl der Bedingungen sucht dann ist das Multi-Class-Covering-Verfahren empfehlenswert. Ist aber der durchschnittliche Rank für die Entscheidung essenziell, dann sollte das gewichtete-Majority-Voting-Verfahren benutzt werden. Wenn die durchschnittliche Anzahl der gelernten Regeln von großer Bedeutung ist, dann sollte das Defaultverfahren verwendet werden.

4.6.8 Datensätze mit mehrwertigen Klassen und m-Estimate Heuristik

Im Folgenden werden die Vergleichsergebnisse der 15 untersuchten mehrwertigen Datensätze erläutert.

Gemäß der Win-Loss-Tie-Tabelle 4.21 gewinnt das gewichtete-Majority-Voting-Verfahren insgesamt alle drei Wettkämpfe gegen die anderen Verfahren, und ist somit das stärkste Verfahren. Im Gegensatz dazu stellt sich das Defaultverfahren mit zwei Verlusten und einem Gleichstand als das schwächste Verfahren dar. Es ist auch zu beachten, dass das Majority-Voting-Verfahren dem Defaultverfahren sehr nahe liegt.

Hinsichtlich der durchschnittlichen Genauigkeit liefert das gewichtete-Majority-Voting-Verfahren das beste Ergebnis im Vergleich zu allen anderen drei Verfahren. Das Multi-Class-Covering-Verfahren weist hierbei das schlechteste Ergebnis auf. Allerdings sind die vier verschiedenen Verfahren hinsichtlich der durchschnittlichen Genauigkeit nicht sehr unterschiedlich.

Bei der Untersuchung des durchschnittlichen Rankes zeigt das gewichtete-Majority-Voting-Verfahren wieder seine Stärke. Und das Defaultverfahren liefert das schlechteste Ergebnis.

Hinsichtlich der durchschnittlichen Anzahl der gelernten Regeln liefert das Defaultverfahren das beste Ergebnis. Es verliert aber hinsichtlich der durchschnittlichen Anzahl der gelernten Bedingungen gegen das Multi-Class-Covering-Verfahren und das Multi-Class-Covering-Verfahren ist hierbei als das beste Verfahren zu sehen. Das gewichtete-Majority-Voting-Verfahren und das Majority-Voting-Verfahren lernen hingegen Regelmengen mit der größten Anzahl der Regeln und Bedingungen.

Fazit ist, wenn man nach Verfahren mit der höchsten durchschnittlichen Genauigkeit und einem besseren

Tabelle 4.21: Win-Loss-Tie-Analyse für Datensätze mit mehrwertigen Klassen und der m-Estimate Heuristik

m-Estimate	Default	Multi-Class-Covering	Majority-Voting	gewichtetes-Majority-Voting
Default	x	7-8-0	7-7-1	4-10-1
Multi-Class-Covering	8-7-1	x	7-7-1	7-8-0
Majority-Voting	7-7-1	7-7-1	x	5-9-1
gewichtetes-Majority-Voting	10-4-1	8-7-0	9-5-1	x

Tabelle 4.22: Ergebnisse der Evaluation für Datensätze mit mehrwertigen Klassen und der m-Estimate Heuristik

Kriterium	Default	Multi-Class-Covering	Majority-Voting	gewichtetes-Majority-Voting
Macro AVG	75,986	75,174	75,396	76,121
AVG Rank	2,667	2,500	2,633	2,200
AVG # of rules	16,733	19,266	29,733	29,733
AVG # of conditions	51,333	49,133	98,0	98,0

durchschnittlichen Rank sucht, dann ist das gewichtete-Majority-Voting-Verfahren empfehlenswert. Wenn die durchschnittliche Anzahl der gelernten Regeln von großer Bedeutung ist, dann sollte das Defaultverfahren verwendet werden. Ist aber die durchschnittliche Anzahl der gelernten Bedingungen richtungweisend, sollte das Multi-Class-Covering-Verfahren ausgewählt werden.

4.6.9 Datensätze mit mehrwertigen Klassen und Relative-Linear-Cost Heuristik

Im Folgenden werden die Vergleichsergebnisse der 15 untersuchten mehrwertigen Datensätze erläutert.

Gemäß der Win-Loss-Tie-Tabelle 4.23 gewinnt das Multi-Class-Covering-Verfahren insgesamt alle drei Wettkämpfe gegen die anderen Verfahren, und ist somit das stärkste Verfahren. Im Gegensatz dazu stellt sich das Majority-Voting-Verfahren mit drei Verlusten als das schwächste Verfahren dar.

Tabelle 4.23: Win-Loss-Tie-Analyse für Datensätze mit mehrwertigen Klassen und der Relative-Linear-Cost Heuristik

RLC	Default	Multi-Class-Covering	Majority-Voting	gewichtetes-Majority-Voting
Default	x	5-9-1	11-3-1	7-5-3
Multi-Class-Covering	9-5-1	x	8-6-1	8-6-1
Majority-Voting	3-11-1	6-8-1	x	2-12-1
gewichtetes-Majority-Voting	5-7-3	6-8-1	12-2-1	x

Hinsichtlich der durchschnittlichen Genauigkeit liefert das gewichtete-Majority-Voting-Verfahren das beste Ergebnis im Vergleich zu allen anderen drei Verfahren. Das Multi-Class-Covering-Verfahren weist hingegen das schlechteste Ergebnis auf. Allerdings sind die vier verschiedenen Verfahren hinsichtlich der durchschnittlichen Genauigkeit nicht sehr unterschiedlich.

Dem durchschnittlichen Rank zufolge wird das Multi-Class-Covering-Verfahren als das beste Verfahren eingestuft. Das gewichtete-Majority-Voting-Verfahren und das Defaultverfahren teilen sich hierbei gleichzeitig den zweiten Platz, während das Majority-Voting-Verfahren das schlechteste Ergebnis liefert.

Tabelle 4.24: Ergebnisse der Evaluation für Datensätze mit mehrwertigen Klassen und der Relative-Linear-Cost Heuristik

Kriterium	Default	Multi-Class-Covering	Majority-Voting	gewichtetes-Majority-Voting
Macro AVG	75,410	72,667	74,477	75,528
AVG Rank	2,300	2,233	3,167	2,300
AVG # of rules	10,866	13,133	18,6	18,6
AVG # of conditions	32,533	32,466	65,733	65,733

Hinsichtlich der durchschnittlichen Anzahl der gelernten Regeln liefert das Defaultverfahren das beste Ergebnis. Es verliert aber hinsichtlich der durchschnittlichen Anzahl der gelernten Bedingungen knapp gegen das Multi-Class-Covering-Verfahren und das Multi-Class-Covering-Verfahren ist hierbei als das beste Verfahren zu sehen. Das gewichtete-Majority-Voting-Verfahren und das Majority-Voting-Verfahren lernen hingegen Regelmengen mit der größten Anzahl der Regeln und Bedingungen.

Fazit ist, wenn man nach Verfahren mit der höchsten durchschnittlichen Genauigkeit sucht, dann ist das gewichtete-Majority-Voting-Verfahren empfehlenswert. Sind aber der durchschnittliche Rank und die durchschnittliche Anzahl der Bedingungen die wichtigsten Entscheidungskriterien, ist die Verwendung des Multi-Class-Covering-Verfahrens vorteilhaft. Wenn die durchschnittliche Anzahl der gelernten Regeln von großer Bedeutung ist, dann sollte das Defaultverfahren verwendet werden.

4.6.10 Datensätze mit mehrwertigen Klassen und Weighted-Relative-Accuracy Heuristik

Im Folgenden werden die Vergleichsergebnisse der 15 untersuchten mehrwertigen Datensätze erläutert.

Gemäß der Win-Loss-Tie-Tabelle 4.25 gewinnt das Multi-Class-Covering-Verfahren gegen die anderen drei Verfahren und ist somit das stärkste Verfahren. Im Gegensatz dazu stellen sich das Majority-Voting-Verfahren und das Defaultverfahren mit zwei Verlusten und einem Gleichstand als die schwächsten Verfahren dar.

Tabelle 4.25: Win-Loss-Tie-Analyse für Datensätze mit mehrwertigen Klassen und der Weighted-Relative-Accuracy Heuristik

WRAcc	Default	Multi-Class-Covering	Majority-Voting	gewichtetes-Majority-Voting
Default	x	4-10-1	7-7-1	4-9-2
Multi-Class-Covering	10-4-1	x	9-3-3	7-6-2
Majority-Voting	7-7-1	3-9-3	x	2-10-3
gewichtetes-Majority-Voting	9-4-2	6-7-2	10-2-3	x

Tabelle 4.26: Ergebnisse der Evaluation für Datensätze mit mehrwertigen Klassen und der Weighted-Relative-Accuracy Heuristik

Kriterium	Default	Multi-Class-Covering	Majority-Voting	gewichtetes-Majority-Voting
Macro AVG	70,599	72,304	69,062	72,025
AVG Rank	2,867	2,067	2,967	2,100
AVG # of rules	5,4	8,533	7,00	7,00
AVG # of conditions	19,466	23,600	26,00	26,00

Hinsichtlich der durchschnittlichen Genauigkeit liefert das Multi-Class-Covering-Verfahren das beste Ergebnis im Vergleich zu den anderen drei Verfahren. Das schlechteste Ergebnis zeigt hingegen das Majority-Voting-Verfahren. Allerdings sind die vier verschiedenen Verfahren hinsichtlich der durchschnittlichen Genauigkeit nicht sehr unterschiedlich.

Bei der Untersuchung des durchschnittlichen Rankes gewinnt das Multi-Class-Covering-Verfahren gegen die anderen drei Verfahren und liefert das beste Ergebnis. Hierbei liefert das Majority-Voting-Verfahren wieder das schlechteste Ergebnis.

Hinsichtlich der durchschnittlichen Anzahl der gelernten Regeln und Bedingungen liefert das Defaultverfahren bessere Ergebnisse. Es ist auch bemerkenswert, dass alle vier Verfahren unter Verwendung der Weighted-Relative-Accuracy Heuristik Regelmengen mit geringer Anzahl an Regeln und Bedingungen zurückliefern.

Fazit ist, wenn man nach Verfahren mit der höchsten durchschnittlichen Genauigkeit und einem besseren durchschnittlichen Rank sucht, dann ist das Multi-Class-Covering-Verfahren empfehlenswert. Wenn die durchschnittliche Anzahl der gelernten Regeln und Bedingungen von großer Bedeutung ist, dann sollte das Defaultverfahren verwendet werden.

4.7 Allumfassender Vergleich hinsichtlich der durchschnittlichen Genauigkeit, der durchschnittlichen Anzahl an gelernten Regeln und der durchschnittlichen Anzahl an gelernten Bedingungen

In diesem Abschnitt werden die vier verwendeten Lern- und Klassifikationsverfahren hinsichtlich der durchschnittlichen Genauigkeit, der durchschnittlichen Anzahl an gelernten Regeln und der durchschnittlichen Anzahl an gelernten Bedingungen unter Verwendung aller ausgewählten Heuristiken miteinander verglichen. Hierbei werden alle 30 Datensätze verwendet und die Datensätze mit binärwertigen und mehrwertigen Klassen werden voneinander nicht getrennt. Die Tabelle 4.27 beinhaltet die durchschnittlichen Genauigkeiten der vier Lern- und Klassifikationsverfahren unter Verwendung aller ausgewählten Heuristiken.

Tabelle 4.27: Die durchschnittlichen Genauigkeiten der vier Lern- und Klassifikationsverfahren

Heuristik	Default	Multi-Class-Covering	Majority-Voting	gewichtetes-Majority-Voting
Correlation	78,637	75,519	78,729	799,575
Laplace	76,984	77,616	77,208	78,260
m-Estimate	78,758	78,347	79,025	79,561
RLC	78,139	77,186	78,398	79,026
WRAcc	76,103	77,197	75,521	77,016
Mittelwert	77,724	77,173	77,776	78,687

Bei diesem Vergleich sind wir davon ausgegangen, dass die Heuristiken gleich wichtig sind, daher werden die durchschnittlichen Genauigkeiten über fünf Heuristiken ohne Gewichtung gemittelt. Wird aber eine Heuristik gegenüber den anderen Heuristiken vorgezogen, kann man einfach dieser Heuristik höhere Gewichte zuweisen und dann den Mittelwert gewichtet berechnen.

Dem Mittelwert der durchschnittlichen Genauigkeiten zufolge stellen wir fest, dass das gewichtete-Majority-Voting-Verfahren im Allgemeinen und ohne Unterscheidung zwischen verschiedenen Heuristiken und zwischen Datensätzen mit binärwertigen und mehrwertigen Klassen, das beste Verfahren hinsichtlich der durchschnittlichen Genauigkeit ist. Das Multi-Class-Covering-Verfahren ist im Durchschnitt mit einer Genauigkeit von 77,173 Prozent als das schlechteste Verfahren hinsichtlich der durchschnittlichen Genauigkeit zu bezeichnen. Es ist aber auch festzustellen, dass der Unterschied zwischen dem gewichteten-Majority-Voting-Verfahren und den anderen Verfahren ziemlich gering ist. Die restlichen drei Verfahren unterscheiden sich dabei noch geringer.

Tabelle 4.28: Die durchschnittliche Anzahl der gelernten Regeln

Heuristik	Default	Multi-Class-Covering	Majority-Voting	gewichtetes-Majority-Voting
Correlation	15,666	9,633	25,966	25,966
Laplace	45,100	47,733	90,433	90,433
m-Estimate	13,433	19,166	26,666	26,666
RLC	8,433	15,533	19,733	19,733
WRAcc	3,666	6,400	6,433	6,433
Mittelwert	17,259	19,693	33,846	33,846

Tabelle 4.29: Die durchschnittliche Anzahl der gelernten Bedingungen

Heuristik	Default	Multi-Class-Covering	Majority-Voting	gewichtetes-Majority-Voting
Correlation	49,366	29,000	83,666	83,666
Laplace	117,300	88,666	233,700	233,700
m-Estimate	44,400	48,700	91,833	91,833
RLC	26,000	35,433	62,566	62,566
WRAcc	12,900	19,033	22,8	22,8
Mittelwert	49,993	44,166	98,913	98,913

Gemäß der Tabellen 4.28 und 4.29 erzeugt das Defaultverfahren im Allgemeinen und ohne Unterscheidung zwischen verschiedenen Heuristiken und zwischen Datensätzen Regelmengen mit der geringsten Anzahl der Regeln, während das Multi-Class-Covering-Verfahren Regelmengen mit der geringsten Anzahl der Bedingungen generiert. Hierbei wird auch wie beim Vergleich hinsichtlich der durchschnittlichen Genauigkeit der ungewichtete Mittelwert berechnet. Es ist auch zu beachten, dass das Defaultverfahren und das Multi-Class-Covering-Verfahren hinsichtlich der durchschnittlichen Anzahl der gelernten Regeln und Bedingungen deutlich bessere Ergebnisse als das Majority-Voting-Verfahren und das gewichtete-Majority-Voting-Verfahren aufweisen, wobei das Majority-Voting-Verfahren und das gewichtete-Majority-Voting-Verfahren gleiche Ergebnisse zurückliefern. Es wird an dieser Stelle wieder bestätigt, dass die vier Lernverfahren unter Verwendung der Weighted-Relative-Accuracy Heuristik ihre allgemeinsten Regelmengen erzeugen, während die vier Lernverfahren unter Verwendung der Laplace Heuristik im Vergleich zu anderen Heuristiken Regelmengen mit der größten Anzahl an Regeln und Bedingungen generieren.

5 Zusammenfassung

Bei der Evaluation wurden die vier verschiedenen Kombinationen in 4.1 anhand einer Datenmenge, bestehend aus 30 Datensätzen, verglichen. Dabei wurde vier Kriterien als Vergleichskriterien festgelegt: Anzahl der gelernten Regeln, Anzahl der gelernten Bedingungen, die korrekt klassifizierten Instanzen und die korrekt klassifizierten Instanzen bei der Kreuz-Validierung. Die ausgewählten Heuristiken zur Evaluation waren: Correlation, Laplace, m-Estimate, Relative-Linear-Cost und Weighted-Relative-Accuracy.

Die vier Kombinationen unter Verwendung aller fünf Heuristiken liefern bei allen 30 Datensätzen für die korrekt klassifizierten Instanzen und die korrekt klassifizierten Instanzen bei der Kreuz-Validierung Ergebnisse, die deutlich besser als die Ergebnisse eines randomisierten Verfahrens sind.

Hinsichtlich der durchschnittlichen Genauigkeit zeigt das gewichtete-Majority-Voting-Verfahren unter Verwendung der Correlation, m-Estimate und Relative-Linear-Cost sowohl für Datensätze mit binärwertigen Klassen als auch für die mit mehrwertigen Klassen, das beste Ergebnis auf. Dafür hat aber das gewichtete-Majority-Voting-Verfahren fast bei allen Heuristiken und für die Datensätze mit binärwertigen und mehrwertigen Klassen Regelmengen mit der größten Anzahl an gelernten Regeln und Bedingungen.

Während das gewichtete-Majority-Voting-Verfahren unter Verwendung der Laplace Heuristik für die Datensätze mit binärwertigen Klassen wieder die beste durchschnittliche Genauigkeit hat, verliert es gegen das Multi-Class-Covering-Verfahren für die Datensätze mit mehrwertigen Klassen und steht auf dem zweiten Platz.

Für den Erfolg des gewichteten-Majority-Voting-Verfahrens ist die Rolle des Klassifizierungsverfahrens essenziell. Denn das Majority-Voting-Verfahren verwendet zwar dasselbe Lernverfahren, es kann aber nicht wie das gewichtete-Majority-Voting-Verfahren hohe Prozentzahlen für die durchschnittlichen Genauigkeiten aufweisen.

Unter Verwendung der Weighted-Relative-Accuracy Heuristik ist das Multi-Class-Covering-Verfahren hinsichtlich der durchschnittlichen Genauigkeit für die Datensätze mit binärwertigen und mehrwertigen Klassen als das beste Verfahren zu bezeichnen.

Die Untersuchung der niedrigsten Werte der durchschnittlichen Genauigkeiten sind für die Datensätze mit binärwertigen und mehrwertigen Klassen sehr interessant. Während das Defaultverfahren bei der Laplace, Relative-Linear-Cost und der Weighted-Relative-Accuracy Heuristik und das Multi-Class-Covering-Verfahren bei den restlichen zwei Heuristiken für die Datensätze mit binärwertigen Klassen hinsichtlich der durchschnittlichen Genauigkeit die schlechtesten Verfahren sind, liefern das Majority-Voting-Verfahren bei der Correlation, Laplace und der Weighted-Relative-Accuracy Heuristik und das Multi-Class-Covering-Verfahren bei den restlichen zwei Heuristiken für die Datensätze mit mehrwertigen Klassen die schlechtesten Ergebnisse. Es ist zu beachten, dass das Majority-Voting-Verfahren hinsichtlich der durchschnittlichen Genauigkeit für die Datensätze mit binärwertigen Klassen nach dem gewichteten-Majority-Voting-Verfahren insgesamt das zweitstärkste Verfahren ist.

Es sollte auch nicht vergessen werden, dass die Unterschiede, die die vier Verfahren hinsichtlich der durchschnittlichen Genauigkeit und unter Verwendung verschiedener Heuristiken aufweisen, nicht sehr stark sind. Im vorigen Abschnitt wurde auch ein allgemeiner Vergleich durchgeführt, indem der Mittelwert der durchschnittlichen Genauigkeiten über fünf Heuristiken berechnet wurde. Das gewichtete-Majority-Voting-Verfahren gewann dabei den Wettkampf gegen die anderen Verfahren, wobei der Unterschied zwischen den vier Verfahren wieder ziemlich gering war.

Nun schauen wir auf die besten Verfahren hinsichtlich der durchschnittlichen Anzahl der Regeln und Bedingungen. Für die Datensätze mit binärwertigen Klassen generiert das Defaultverfahren bei der m-Estimate, Relative-Linear-Cost und der Weighted-Relative-Accuracy Heuristik Regelmengen mit der geringsten Anzahl der Regeln und Bedingungen. Das Multi-Class-Covering-Verfahren weist bei den restlichen zwei Heuristiken die besten Ergebnisse auf.

Für die Datensätze mit mehrwertigen Klassen sollte zuerst die Anzahl der Regeln und Bedingungen getrennt betrachtet werden. Denn während das Defaultverfahren unter Verwendung der Laplace, m-Estimate und der Relative-Linear-Cost Heuristik die besten Ergebnisse für die durchschnittliche Anzahl der Regeln liefert, erzeugt das Multi-Class-Covering-Verfahren bei denselben Heuristiken Regelmengen mit der geringsten Anzahl der Bedingungen.

Unter Verwendung der Correlation und der Weighted-Relative-Accuracy Heuristik sind jeweils das Multi-Class-Covering-Verfahren und das Defaultverfahren die besten Verfahren hinsichtlich der durchschnittlichen Anzahl der Regeln und Bedingungen.

Abschließend stellen wir fest, dass das Defaultverfahren und das Multi-Class-Covering-Verfahren insge-

samt Regelmengen mit der geringsten Anzahl der Regeln und Bedingungen erzeugen. Wobei das Multi-Class-Covering-Verfahren gegenüber dem Defaultverfahren den Vorteil hat, dass man gemäß der Tabellen 4.5 und 4.6 noch kürzere Regelmengen generieren kann, indem man den Verfahrensparameter auf einen niedrigeren Wert setzt.

6 Anhang

6.1 Dokumentation der Implementierung

Um SECO-Framework um die Funktionalitäten, die in Abschnitt 3 beschrieben wurden, zu erweitern, sind folgende Komponenten in SECO implementiert worden.

Tabelle 6.1: Übersicht der zu SECO hinzugefügten Komponenten

Einbauort	Beschreibung
de.tu_darmstadt.ke.seco.learning.method.DefaultLearningMethod	Implementierung des Defaultlernverfahrens
de.tu_darmstadt.ke.seco.learning.method.LearningByMajorityVoting	Implementierung des Majority-Voting-Verfahrens
de.tu_darmstadt.ke.seco.learning.method.LearningByMultiClassCovering	Implementierung des Multi-Class-Covering-Verfahrens
de.tu_darmstadt.ke.seco.classification.method.DefaultClassificationMethod	Implementierung des Defaultklassifizierungsverfahens
de.tu_darmstadt.ke.seco.classification.method.ClassificationByMajorityVoting	Implementierung des Majority-Voting-Verfahrens
de.tu_darmstadt.ke.seco.classification.method.ClassificationByWeightedMajorityVoting	Implementierung des gewichteten-Majority-Voting-Verfahrens
de.tu_darmstadt.ke.seco.learners.core.IRuleLearning	Die Oberklasse für alle implementierten Lernverfahren. Das IRuleLearning Interface hat createClassifier und separateAndConquer Methoden
de.tu_darmstadt.ke.seco.learners.core.IInstanceClassification	Die Oberklasse für alle implementierten Klassifizierungsverfahren. Das IInstanceClassification Interface hat classifyInstance Methode

Neben der Implementierung neuer Klassen wurden auch SECO-Klassen geändert. Die folgende Tablle fasst die wichtigsten Änderungen zusammen.

Tabelle 6.2: Übersicht der in SECO durchgeführten Änderungen

Einbauort	Beschreibung
de.tu_darmstadt.ke.seco.learners.core.AbstractSeco	m_ruleLearning und m_instanceClassification wurden hinzugefügt. Zur Ausgabe wurde showShortConfig Methode angepasst. Zwei neue Methoden createClassifier und separateAndConquer jeweils mit 2 Argumenten wurden hinzugefügt.
de.tu_darmstadt.ke.seco.models.RuleSet	Die Methode classifyInstance mit 2 Argumenten, nämlich Instance und IInstanceClassification, wurde hinzugefügt.
de.tu_darmstadt.ke.seco.learners.factoryConfiguarableSeco	INSTANCECLASSIFICATION und RULELEARNING String-Objekte wurden hinzugefügt. Die Methode setSecoComponent wurde angepasst.

Bibliographie

[1] Bojan Cestnik and Ivan Bratko. Learning redundant rules in noisy domains. In *ECAI'88*, pages 348–350, 1988.

[2] Peter Clark and Robin Boswell. Rule induction with cn2: Some recent improvements. In *Proceedings of the Fifth European Working Session on Machine Learning*, pages 151–163, Berlin, 1991. Springer.

[3] Peter Clark and Robin Boswell. Rule induction with cn2: Some recent improvements. In *EWSL'91*, pages 151–163, 1991.

[4] Peter Clark and Tim Niblett. The cn2 induction algorithm. *Machine Learning*, 3:261–283, 1989.

[5] Sašo Džeroski, Bojan Cestnik, and Igor Petrovski. Using the m-estimate in rule induction. *J. Comput. Inf. Technol.*, 1:37–46, March 1993.

[6] Johannes Fürnkranz. FOSSIL: A robust relational learner. In F. Bergadano and Luc De Raedt, editors, *Proceedings of the 7th European Conference on Machine Learning (ECML-94)*, volume 784 of *Lecture Notes in Artificial Intelligence*, pages 122–137, Catania, 1994. Springer-Verlag.

[7] Johannes Fürnkranz. Separate-and-conquer rule learning. *Artif. Intell. Rev.*, 13:3–54, February 1999.

[8] Frederik Janssen and Johannes Fürnkranz. On the quest for optimal rule learning heuristics. *Machine Learning*, 78(3):343–379, 2010.

[9] Frederik Janssen and Johannes Fürnkranz. The seco-framework for rule learning. Technical Report TUD-KE-2010-02, TU Darmstadt, Knowledge Engineering Group, 2010.

[10] Igor Kononenko. Id3, sequential bayes, naive bayes and bayesien neural networks. In *Proc. of the fourth European Working Session on Learning*, pages 91–98, Montpellier, 1989.

[11] Igor Kononenko. An experiment in machine learning of redundant knowledge. In *Proc. Intern. Conf. MELECON*, pages 1146–1149, Ljubljana, 1991.

[12] Igor Kononenko. Combining decisions of multiple rules. In *AIMSA'92*, pages 87–96, 1992.

[13] Nada Lavrac, Peter A. Flach, and Blaz Zupan. Rule evaluation measures: A unifying view. In *Proceedings of the 9th International Workshop on Inductive Logic Programming*, ILP '99, pages 174–185, London, 1999. Springer-Verlag.

[14] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[15] Padhraic Smyth, Rodney M. Goodman, and Charles M. Higgins. A hybrid rule-based/bayesian classifier. In *ECAI'90*, pages 610–615, 1990.

[16] Matthias Thiel. Separate and conquer framework und disjunktive regeln. Master's thesis, TU Darmstadt, May 2005.

Abbildungsverzeichnis

Tabellenverzeichnis

Algorithmenverzeichnis