Fortlaufendes inkrementelles Lernen und Vergessen

Dominikus Herz

Seminar KI: gestern, heute, morgen
Angewandte Informatik, Universität Bamberg

Zusammenfassung. Da ein lernendes System darauf angewiesen ist, aus Beispieldaten abstraktes Wissen abzuleiten, ist es von Bedeutung diesen Prozess zu optimieren. Um dies zu erreichen wird in dieser Seminararbeit ein inkrementelles fortlaufendes lernendes System vorgestellt, das durch den Mechanismus des Vergessens die eigene Lernfähigkeit steigert. Um festzustellen, welches Wissen auf lange Zeit gültig ist und welche Erkenntnisse wieder vergessen werden können, nutzt das System eine hierarchische Speicherstruktur. Diese zeigt Abhängigkeiten zwischen den gespeicherten Fakten an und erlaubt somit Redundanzen festzustellen und zu beseitigen.

Schlüsselwörter: Computing methodologies → Rule learning, minimum message length, learning with forgetting

1 Einleitung

Intelligente Systeme begegnen uns tagtäglich. Beispielsweise bei individualisierten Radiosendern auf unserer bevorzugten Musik-Streaming-Plattform. Zurecht wird von uns erwartet, dass ein solches System lernt unseren Musikgeschmack zu erkennen und diese Musikstücke in die Playlist des Radiosenders einbindet. Natürlich ist es sinnvoll, dass diese Systeme sich an Änderungen unseres Musikgeschmacks fortlaufend anpassen können. Wenn wir im Dezember bestimmte Weihnachtslieder bevorzugt hören, erwarten wir, dass im nächsten Jahr zur gleichen Zeit unsere favorisierten Weihnachtslieder vom System nicht vergessen wurden. Es ist also von großer Bedeutung, dass lernende Systeme neues Wissen erlangen können, ohne das zuvor gefestigte zu vergessen.

Ein System, das während seiner Laufzeit fortlaufend neues Wissen erlangen möchte, muss selektiv entscheiden können, welches neu gewonnenes Wissen gespeichert werden soll, da es zu sehr hohen Speicherplatzverbrauch führen würde alle Informationen ohne weitere Verarbeitung zu speichern. Neues Wissen muss laut Martínez-Plumed, Ferri, Hernández-Orallo und Ramírez-Quintana (2015) dahingehend überprüft werden, ob es Redundanzen bezüglich des schon gespeicherten Wissen aufweist und ob es mit diesem übereinstimmt. Carpenter und Grossberg (1988) beschreiben dieses Problem als das „stability-plasticity dilemma“, welches die Frage aufwirft, wie ein lernendes System desinged werden kann, sodass es veränderbar oder adaptiv gegenüber signifikanten Events reagieren

kann, jedoch das eigene gefestigte Wissen nicht durch neu gelernte irrelevante Fakten vergisst. Es muss daher sichergestellt werden, dass bereits Gelerntes erhalten bleibt, während das Lernen von Neuem kontinuierlich möglich ist.

Durch den in der folgenden Seminararbeit beschriebenen inkrementellen Ansatz des fortlaufenden Lernens werden kostenintensive Neuerstellungen des gesamten Systemwissens bei Hinzufügen neuer Erkenntnisse vermieden. Der Fokus des Ansatzes liegt darauf, relevantes Wissen zu erkennen und nur jenes in das konsolidierte Wissen des lernenden Systems mitaufzunehmen. Umgesetzt wird dies mit Hilfe von induktivem und deduktivem Reasoning, welches Abstraktion durch den Prozess der Generalisierung ermöglicht.

2 Hintergrund

Im Artikel „Is Learning The n-th Thing Any Easier Than Learning The First?“ kommt Thrun (1996) zum Schluss, dass ein fortlaufend lernendes System neue Lernaufgaben umso schneller meistern kann je mehr es schon gelernt hat. Er begründet dies damit, dass ein fortlaufend lernendes System Wissen von schon gelernten Aufgaben transferieren kann, um neue ähnliche Probleme zu lösen. Dies ist nur möglich, wenn auf dieses schon gelernte Wissen zugegriffen werden kann. Allerdings fehlt diesem Ansatz die Möglichkeit auf Concept Drifts zu reagieren und bereits gelerntes Wissen zu revidieren. Gama, Žliobaitė, Bifet, Pechenizkiy und Bouchachia (2014) erstellen in „A survey on concept drift adaptation“ die These, dass Concept Drifts nur umgangen werden können, wenn das System auch in der Lage ist altes Wissen zu vergessen. Des Weiteren schlagen sie vor ein zweistufiges Prinzip für den Lernprozess bereitzustellen.

1. Ein Kurzzeitgedächtnis, das Inputdaten repräsentiert
2. Ein Langzeitgedächtnis, das Generalisierungen der Daten speichert

Diese Seminararbeit beruht primär auf den Artikeln „Knowledge acquisition with forgetting: an incremental and developmental setting“ von Martínez-Plumed et al. (2015), „A knowledge growth and consolidation framework for Lifelong Machine Learning systems “ von Martinez Plumed, Ferri, Hernandez Orallo und Ramirez Quintana (2014) sowie der Dissertation „Incremental and developmental perspectives for general-purpose learning systems“ von Martínez-Plumed (2016), da diese Arbeiten die zuvor beschriebenen Vorteile von lernenden Systemen zusammenfassen. Sie bieten ein inkrementelles fortlaufendes Lernen, das sogar mehrere Stufen von Generalisierungen verarbeiten kann. Analog dazu bietet es einen Arbeitsbereich, der als Kurzzeitgedächtnis interpretiert werden kann und ein konsolidiertes Wissen, welches der Vorstellung eines Langzeitgedächtnisses nahe kommt. Durch die Möglichkeit des Vergessens, wird das Lernen effizienter gestaltet.

3 Inkrementelles Lernen und Vergessen

Martínez-Plumed et al. (2015) beschreiben die Idee hinter ihrem inkrementell lernendem System als ein fortlaufendes System, welches Wissen hierarchisch struk-

turieren, speichern und auch wieder vergessen kann. Es werden Regeln benutzt um Beispieldaten der Problemstellung, neue Hypothesen und Hintergrundwissen zu modellieren.

3.1 Architektur

Das lernende System besteht aus mehreren Komponenten, die mit dem Arbeitsbereich interagieren. Der Arbeitsbereich repräsentiert dabei den momentanen Status des gelernten Wissens. In Bezug auf Beispiele ist die Klasse einer Regel $c = class(\rho)$, wobei $c \in C$ die Ausprägung eines Klassensets ist, wie beispielsweise $C = \{true, false\}$. Wie in Abbildung 1 zu sehen ist, besteht der Arbeitsbe-

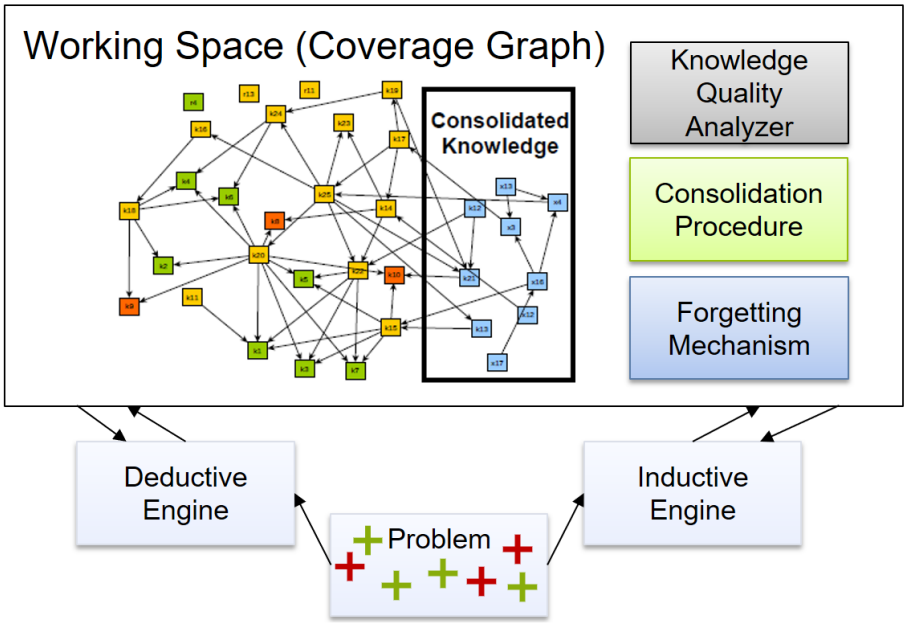

Abb. 1. Coverage Graph Architektur (Abbildung angelehnt an Martinez Plumed et al. (2014))

reich aus positiven $class(\rho) = true$ (grün) und negativen $class(\rho) = false$ (rot) Beispielen, die eine Problemstellung definieren, sowie aus den von der induktiven Engine generierten Hypothesen (gelb), welche entweder in das konsolidierte Wissen (blau) aufgenommen werden oder mit der Zeit vergessen werden sollen. Das konsolidierte Wissen beinhaltet Hintergrundwissen, das zuvor modelliert werden muss und jene Regeln, die vom System als wahr und relevant angesehen werden und somit in das konsolidierte Wissen mitaufgenommen werden. R

beschreibt das Set aller möglichen Regeln, während $W \subset R$ für den Arbeitsbereich steht. Das Hintergrundwissen, welches initial modelliert werden muss, wird durch $K \subset R$ beschrieben. Die deduktive Engine überprüft mit Hilfe des konsolidierten Wissens, welche anderen Regeln von jeder neu generierten Hypothese abgedeckt werden und ordnet sie in den Coverage Graph ein. Eine Regel ρ_a ist von einer anderen Regel ρ_b abgedeckt wenn $(K \cup \rho_b) \vDash \rho_a$ erfüllt ist. Dies bedeutet, dass durch das konsolidierte (Hintergrund-) Wissen K und der Regel ρ_b die Regel ρ_a bereits modelliert ist. Der „Knowledge Qulity Analyzer“ bewertet die neu erstellten Hypothesen anhand ihrer Komplexität, Redundanz in Bezug auf andere Hypothesen und Regeln sowie ihrer Validität in Bezug auf die verschiedenen Klassen der gegebenen Beispiele der Problemstellung. Von diesen Qualitätsmerkmalen abhängig können Hypothesen und Regeln in das konsolidierte Wissen des Systems mit Hilfe der „Consolidation Procedure“ mitaufgenommen, im Arbeitsbereich gehalten oder vergessen werden.

3.2 Coverage Graph

Um einen Coverage Graph zu visualisieren werden die Regeln durch gerichtete Kanten miteinander verbunden. Die gerichteten Kanten beschreiben, ob eine Regel durch eine andere abgedeckt wird. In Abbildung 2 wird die Regel ρ_2 durch die

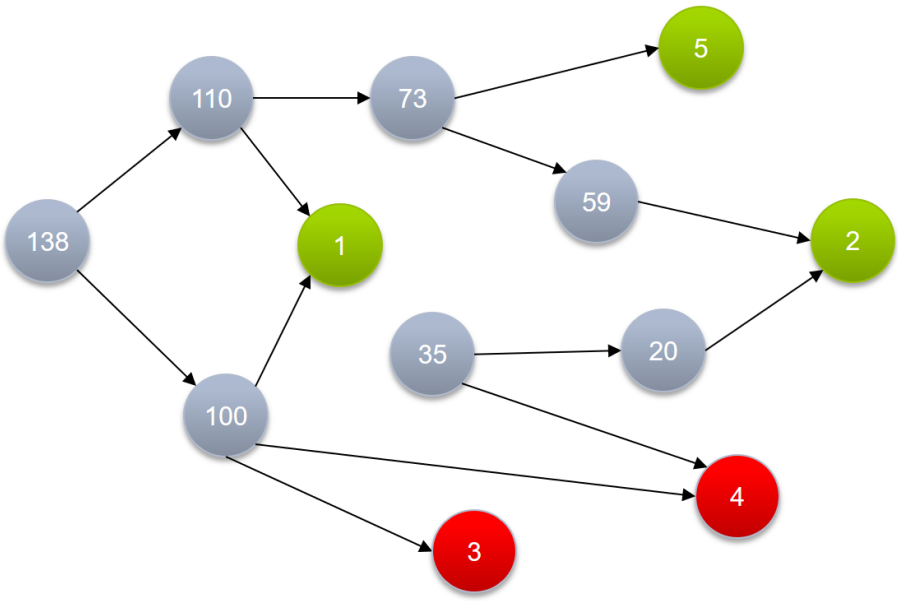

Abb. 2. Coverage Graph

Regel ρ_{59} abgedeckt. Das heißt es existiert ein Modell $(K \cup \rho_{59}) \vDash \rho_2$. Diese Ab-

deckung (Coverage) wird transitiv definiert (Martínez-Plumed, 2016). Das heißt, dass die Regel ρ_{73}, welche die Regel ρ_{59} modelliert $(K \cup \rho_{73}) \vDash \rho_{59}$, ebenfalls ein Modell der Regel ρ_2 darstellt $(K \cup \rho_{73}) \vDash \rho_2$. Daher müssen im Coverage Graphen diese zusätzlichen Kanten nicht eingezeichnet werden. $(K \cup \rho_{59}) \vDash \rho_2$ kann durch $\rho_{59} \rightarrow \rho_2$ kürzer ausgedrückt werden. Die Nachfahren einer Regel werden als $suc(\rho_a) = \{\rho_b | \rho_a \rightarrow \rho_b\}$ sowie die Vorfahren als $anc(\rho_a) = \{\rho_b | \rho_b \rightarrow \rho_a\}$ definiert. Je mehr Ebenen von Nachfahren eine Regel besitzt, um so stärker generalisiert ist das darin enthaltene Wissen. Daraus folgt, dass eine Regel ρ_a, welche ein Beispiel der Problemstellung widerspiegelt, nur als Knoten ohne Nachfahren $|suc(\rho_a)| = 0$ in einen Coverage Graphen eingefügt werden kann. Knoten ohne Nachfahren werden fortan als Blattknoten bezeichnet, während Knoten ohne Vorfahren als Wurzelknoten bezeichnet werden, siehe Gleichung 1.

$$\forall \rho \in W : |suc(\rho)| = 0 \implies \rho \in leaves \tag{1}$$

$$\forall \nu \in W : |anc(\nu)| = 0 \implies \nu \in roots \tag{2}$$

Abbildung 3 zeigt ein Beispiel des Inductive Logic Programming, welches sich an „Inductive logic programming: Theory and methods“ von Muggleton und de Raedt (1994) anlehnt. Das konsolidierte Wissen des lernendem Systems sind durch die Regeln k_1 - k_7 dargestellt. Nun soll anhand der klassifizierten Beispiele ρ_1 - ρ_5 eine Generalisierung der Tochterbeziehung $daughter(X, Y)$ zwischen zwei Menschen gelernt werden. Die weiteren in Abbildung 3 aufgelisteten Regeln werden von der induktiven Engine als Hypothesen der Generalisierung der Tochterbeziehung bereitgestellt. Der dazugehörige Coverage Graph in Abbildung 2 zeigt den Arbeitsbereich ohne dem dazugehörigen konsolidierten Wissen. Der Arbeitsbereich des Coverage Graphen zeigt drei positiv klassifizierte ρ_1, ρ_2 und ρ_5 sowie zwei negativ klassifizierte Beispiele ρ_3 und ρ_4.

3.3 Bewertungskriterien

Um zu erkennen, ob neu erstellte Hypothesen in das konsolidierte Wissen mitaufgenommen werden sollen, müssen diese geprüft werden können. Dabei spielen die Komplexität der Hypothese, die Redundanz in Bezug auf andere Regeln und Hypothesen sowie ihre Relevanz und Korrektheit eine große Rolle.

Minimal Message Length In der Annahme, dass der Nutzen einer Regel durch die eigene Komplexität in Bezug auf die Komplexität der abgedeckten Regel getroffen werden kann, schlägt Martínez-Plumed (2016) in „Incremental and developmental perspectives for general-purpose learning systems“ ein auf dem Ockhams Rasiermesser basierendes Bewertungsverfahren vor. Ockhams Rasiermesser besagt, dass von mehreren Erklärungen, die den selben Sachverhalt beschreiben, die Einfachere bevorzugt werden soll. Auf dieser Grundlage beschreiben Wallace und Boulton (1968) in „An information measure for classification“ das Prinzip der Minimal Message Length. Sie machen den Vorschlag, dass die beste Klassifikation jene ist, welche die kürzeste Beschreibung der Information liefert. Martínez-Plumed et al. (2015) greifen diese Idee auf und setzt

Background Knowledge		Rules	
ID		ID	
k1	parent(ann, mary).	1	daughter(mary,ann).
k2	parent(ann, tom).	2	daughter(eve,tom).
k3	parent(tom, eve).	3	daughter(tom,ann).
k4	parent(tom, ian).	4	daughter(eve,ann).
k5	female(ann).	5	daughter(cris,tom).
k6	female(mary).	20	daughter(eve,tom):- female(eve).
k7	female(eve).	35	daughter(eve,Y):- female(eve).
		59	daughter(eve,tom):- female(eve),parent(tom,eve).
		73	daughter(X,tom):- female(X),parent(tom,X).
		100	daughter(X,Y):- female(Y),parent(Y,mary).
		110	daughter(X,Y):- female(X),parent(Y,X).
		138	daughter(V,W):- female(X),parent(Y,Z).

Abb. 3. Regeln

sie für den Coverage Graphen um. Wir können die Wahrscheinlichkeit einer Hypothese H für eine Aussage (ein Beispiel) E als die bedingte Wahrscheinlichkeit $P(H|E)$ ausdrücken, welche sich mit Hilfe des Satz von Bayes (Mitchell, 2005) wie in Gleichung 3 darstellen lässt.

$$L(H|E) = \frac{P(H) \cdot P(E|H)}{P(E)} \tag{3}$$

Martínez-Plumed et al. (2015) führen des weiteren an, dass mit Hilfe der informationstheoretischen Interpretation der Minimal Message Length (MML) die Aussage E mit der Wahrscheinlichkeit $P(E)$ als Message der Länge $L(E) = -log_2 P(E)$ ausgedrückt werden kann. Den negativen Logarithmus auf Gleichung 3 anzuwenden, kann laut Martínez-Plumed (2016) unter Betracht der MML zu einer Interpretation führen, dass die Länge einer Hypothese H mit gegebener Aussage E als $L(H|E)$ beschrieben werden kann. Diese is definiert aus der Summe von drei Heuristiken:

- Ein Komplexitätsheuristik $H_{comp}(L(H))$, welche die Komplexität der Hypothese H beschreibt.
- Eine Coverageheuristik $H_{cov}(L(E|H))$, welche die Höhe der benötigten Zusatzinformationen beschreibt, die benötigt werden, um die Aussage (E) bei gegebener Hypothese (H) auszudrücken.
- Die Länge der Aussage $(L(E))$, die für konkurrierende Hypothesen gleich ist.

$$L(H|E) = L(H) + L(E|H) - L(E) \tag{4}$$

Minimiert man die sich daraus ergebende Gleichung 4, maximiert man die zuvor definierte bedingte Wahrscheinlichkeit. Es wird also versucht nach dem Modell mit der kürzesten Message zu suchen. Lässt man nun zu, dass die Aussage E nicht nur von Beispielen, sondern auch von anderen Regeln eingenommen werden kann, kann man das Prinzip der MML auf das hierarchische Modell des Coverage Graphen anwenden. Um dies zu erreichen, müssen wir die Länge einer Hypothese mit gegebener Aussage $L(H|E)$ dahingehend umgestalten, dass wir nun die Länge einer Regel ρ in unserem Arbeitsbereich W in Bezug zu den restlichen Regeln in W setzen, zu welchen Hypothesen und Beispiele zählen, da beide durch eine Regel ρ modelliert werden können. Die Länge einer Regel unter der Voraussetzung des Arbeitsbereichs $L(\rho|W)$ wird daher definiert als die Summe aus der Länge dieser Regel $L(\rho)$ und der Länge der Regeln im Arbeitsbereich, die nicht durch ρ modelliert sind $L(W|\rho)$ minus der Länge aller Regeln im Arbeitsbereich W.

$$L(\rho|W) = L(\rho) + L(W|\rho) - L(W) \tag{5}$$

Die Länge einer Regel $L(\rho)$ kann in Abhängigkeit der Repräsentationssprache auf vielerlei Arten definiert werden. Es ist sinnvoll den Anwendungsfall miteinzubeziehen um entscheiden zu können, welche Rechenoperationen mit welchem Ressourcenverbrauch in Zusammenhang steht. Als Anhaltspunkt können beispielsweise, wie auch Martínez-Plumed et al. (2015) vorschlägt die Anzahl der Konstanten und Variablen sowie die Anzahl von Funktionsparametern einer Regel in die Berechnung der Länge $L(\rho)$ miteinbezogen werden.

Für das Beispiel anhand dessen das lernende System die Tochterbeziehung zwischen zwei Menschen lernen soll nehmen wir für die weiteren Bewertungskriterien eine Regellänge von $L(\rho) = 2 \mid \forall \rho \in W$ an.

Support Der Support einer Regel $\rho \in W$ sagt aus wie hoch der Mehrwert dieser Regel in Bezug auf die Abdeckungskraft dieser Regel ist.

$$S(\rho, W) = L(W) - L(W|\rho) \tag{6}$$

Somit kann dieser wie in Gleichung 6 definiert werden durch die Länge aller Regeln im Arbeitsbereich $L(W)$ minus die Länge der Regeln, welche nicht durch ρ abgedeckt werden. Der Support einer Regel ρ sagt also aus wie hoch die Länge der Regeln ist, die durch diese Regel ρ modelliert werden.

$$S(\rho, W) = \sum_{\nu : \rho \models \nu} L(\nu) \tag{7}$$

Durch einsetzen der Gleichung 6 in die Gleichung 5 erhalten wir Gleichung 8.

$$L(\rho|W) = -S(\rho, W) + L(\rho) \tag{8}$$

Gleichung 8 verdeutlicht, dass die Länge einer Regel in Abhängigkeit zu ihrem Arbeitsbereich $L(\rho|W)$ minimiert werden kann, indem der Support dieser Regel

$S(\rho, W)$ maximiert und die eigene Länge $L(\rho)$ minimiert wird. Es wird also wie von Martínez-Plumed et al. (2015) zu Beginn angedacht, nach der Regel gesucht, die am kürzesten ist und möglichst viele andere Regeln modelliert.

Da die Coverage transitiv definiert ist (Kapitel 3.2), müssen wir für die Berechnung des Supports diesen nach oben weiter propagieren. Dabei muss beachtet werden, dass für jeden Knoten im Coverage Graphen der Support, der in ihn von unten hinein propagiert wird, gleich ist, wie der Support den er selbst auf seine Vorfahren weiter propagiert. In Abbildung 4 beispielsweise hat die Regel ρ_2 bzw. der Knoten mit dem Support $S(\rho, W) = 2$ mehr als einen Vorfahren $anc(\rho_2) = \{\rho_{59}, \rho_{20}\}$. Dieser Support wird durch die beiden grünen Quadrate in Abbildung 4 visualisiert. Dieser wird nun zu gleichen Teilen an seine Vorfahren

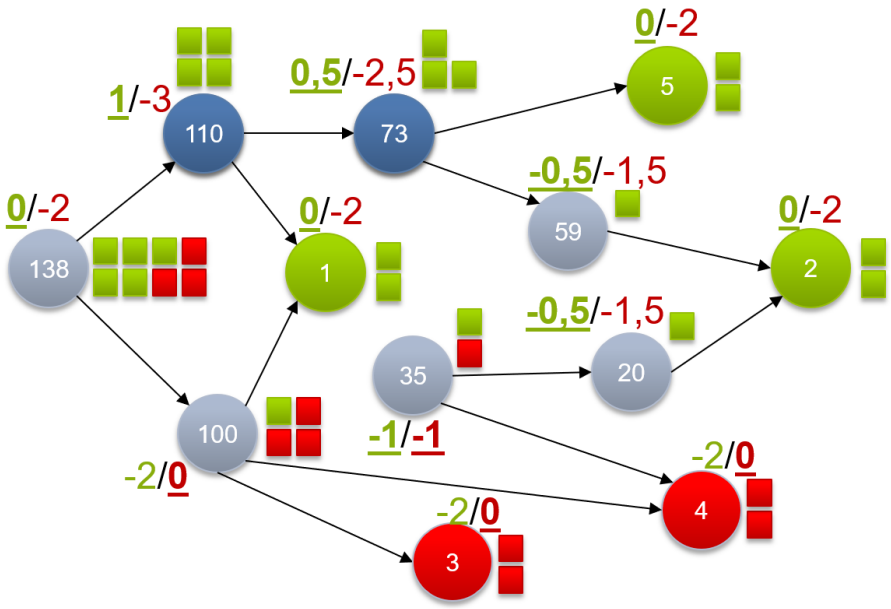

Abb. 4. Coverage Graph mit Support- und Optimality-Werten

ρ_{59} und ρ_{20} weiter propagiert. Dies führt uns zu Gleichung 9, welche besagt, dass die Summe des Supports aller Wurzelknoten gleich der Summe des Supports aller Blattknoten ist.

$$\sum_{\nu \in leaves} \dot{S}(\nu, W) = \sum_{\rho \in roots} \dot{S}(\rho, W) \tag{9}$$

In Abbildung 4 ist ebenfalls zu sehen, dass der Support für jede Klasse der Problemstellung definiert werden muss, um Beispiele den einzelnen Klassen zu-

zuordnen. In unserem Fall findet die Klassifizierung der Beispiele ρ_1 - ρ_5 aus Abbildung 3 in zwei Klassen $C = \{true, false\}$ statt.

Formalisieren lässt sich der Support einer Regel $\dot{S}_c(\rho, W)$ so, dass er für alle Beispiele der Länge des Beispiels entspricht $\dot{S}_c(\rho, W) = L(\rho)$. Für eine Regel ρ, die andere Beispiele oder Regeln abdeckt, gilt, dass ihr Support $\dot{S}_c(\rho, W)$ der Summe des Supports ihrer Nachfahren $\dot{S}_c(\nu, W) : \nu \in suc(\rho)$ entspricht. Wobei alle Nachfahren ihren eigenen Support auf ihre Vorfahren zu gleichen Teilen propagieren. Daraus ergibt sich die Gleichung 10 für den Support einer Regel ρ.

$$\dot{S}_c(\rho, W) = \begin{cases} L(\rho) & \text{wenn } \rho \in leaves \\ \sum_{\nu \in suc(\rho)} \frac{\dot{S}_c(\nu, W)}{|anc(\nu)|} & \text{ansonsten} \end{cases} \tag{10}$$

Unsere Gleichung 8 muss nun also auch die Klasse mitberücksichtigen und kann wie in Gleichung 11 dargestellt werden. Da wir immer noch dem Prinzip der Minimal Message Length folgen, wird durch das geänderte Vorzeichen deutlich, dass für $-L_c$ ein möglichst großer Wert zu einer MML führt.

$$-\dot{L}_c(\rho|W) = \dot{S}_c(\rho, W) - L(\rho) \tag{11}$$

Da ein Beispiel ρ eine Regel ohne Nachfolger $|suc(\rho_a)| = 0$ darstellt und der Support für leaves gleich der Länge ist $\forall \rho \in leaves | \dot{S}_c(\rho, W) = L(\rho)$, gilt für dieses Beispiel ρ: $-\dot{L}_c(\rho|W) = 0$.

Der Support beschreibt wie effizient eine Regel andere Regeln für eine bestimmte Klasse abdeckt. Existieren in der Problemstellung aber mehr als eine Klasse, müssen Regeln allerdings auch daraufhin geprüft und bewertet werden, wie genau sie die Klassifizierungen voneinander abgrenzen können. Mit einem Blick auf Abbildung 4 wird deutlich, dass Regel ρ_{138} den höchsten Support für die Klasse $c = true$ besitzt $\dot{S}_{true}(\rho_{138}, W) = 5$. Allerdings hat Regel ρ_{138} auch den Supportwert für die Klasse $c = false$ von $\dot{S}_{false}(\rho_{138}, W) = 3$. Die Regel ρ_{110} hingegen besitzt die Supportwerte $\dot{S}_{true}(\rho_{110}, W) = 4$ und $\dot{S}_{false}(\rho_{110}, W) = 0$. Sie scheint ein deutlich besser Kandidat zu sein, um in das konsolidierte Wissen übernommen zu werden. Dies bringt uns dazu die Reinheit, mit welcher Regeln klassifizieren, in unsere Bewertung für die Übernahme in das konsolidierte Wissen miteinzubeziehen.

Optimailty Um die Reinheit der Klassifizierung einer Regel miteinzubeziehen, definieren Martínez-Plumed et al. (2015) die Optimality. Sie definieren diese für eine Regel ρ der Klasse $c \in C$, als die Kosten eine Regel zu definieren aus Gleichung 11 minus der Kosten für Ausnahmen, welche diese Regel ebenfalls zulässt. Diese Ausnahmekosten werden festgelegt als die Summe des Supports für andere Klassen dieser Regel $c' \in C | c' \neq c$. Überdies wird ein Faktor β definiert, um die Gewichtung für die Reinheit der Regeln zu beeinflussen, welcher zur

Gleichung 12 führt.

$$opt_c(\rho, W) = -\beta \cdot \dot{L}_c(\rho|W) - (1-\beta) \cdot \sum_{\substack{c' \in C \\ c' \neq c}} \dot{S}_c(\rho, W) \tag{12}$$

Das Maximum der Optimality-Werte der Klassen einer Regel bestimmt die Optimality einer Regel.

$$opt(\rho, W) = \max_{c \in C}(opt_c(\rho, W)) \tag{13}$$

Berechnen wir nun die Optimality für die Klasse $c = true$ der Regel ρ_{100} aus unserem Beispiel. $-\beta \cdot \dot{L}_{true}(\rho_{100}|W)$ setzt sich laut Gleichung 11 zusammen aus dem Support $\dot{S}_{true}(\rho_{100}, W) = 1$ minus der Länge, die wir auf $L(\rho_{100}) = 2$ festgesetzt haben, mal dem Faktor $\beta = 0,5$. Die Summe des Supports aller anderen Klassen $\sum_{\substack{c' \in C \\ true' \neq c}} \dot{S}_c(\rho_{100}, W) = 3$ kann ebenfalls leicht aus Abbildung 4 abgelesen werden. Eingesetzt ergibt dies Gleichung 14

$$opt_{true}(\rho_{100}, W) = -0,5 \cdot -(1-2) - (1-0,5) \cdot 3 = -0,5 - 1,5 = -2 \tag{14}$$

Die Optimality-Werte der beiden Klassen $\{true, false\}$ sind für alle Regeln des Beispiels in Abbildung 4 farblich kodiert eingetragen. Die Werte der Klasse $c = true$ sind grün gekennzeichnet, während die Werte der Klasse $c = false$ rot dargestellt sind. Die sich aus dem Maximum ergebenden Optimality-Werte einer Regel sind zusätzlich unterstrichen. Anhand dieser Abbildung erkennt man schnell, dass die Regel ρ_{110} die höchste Optimility aufweist. Da sie alle Beispiele der Klasse $c = true$ modelliert und gleichzeitig keines der anderen Klassenbeispiele, nimmt es die höchste Optimality an. Beispiele nehmen für ihre Klasse immer einen Optimality-Wert von 0 an, also muss eine Regel, die in das konsolidierte Wissen mitaufgenommen werden soll, eine Optimality besitzen, welche größer ist als ein Beispiel um einen Mehrwert gegenüber einem Beispiel zu generieren. In unserem Fall sind die beiden blau markierten Regeln ρ_{110} und ρ_{73} potentielle Kandidaten, um ins konsolidierte Wissen mitaufgenommen zu werden. Martínez-Plumed (2016) schlägt einen Threshold vor, um neue Regeln aufzunehmen.

Permanence Die Berechnung der Optimality ist eine gute Methode, um herauszufinden, wie gut eine Regel andere Regeln modelliert und festzustellen, wie rein diese Regel ist, in Bezug auf ihre Klassifizierung. Es ist aber ebenfalls wichtig überflüssige Regeln zu erkennen. Regeln können für eine Klasse überflüssig werden, wenn sie von anderen Regeln abgedeckt werden, die eine höhere Optimality besitzen. Dies führt zu der einfachen Gleichung 15 für den Permanence-Wert einer Regel für eine Klasse.

$$perm_c(\rho, W) = opt_c(\rho, W) - max(0, \max_{\nu : \nu \vDash \rho} opt_c(\nu, W)) \tag{15}$$

Dies führt analog zur Optimality zu der generellen Permanance in Gleichung 16.

$$perm(\rho, W) = \max_{c \in C}(perm_c(\rho, W)) \tag{16}$$

Je niedriger dieser Permanence-Wert für eine Regel ausfällt, umso höher ist die Chance, dass sie vergessen werden kann. In unserem Beispiel besitzt die Regel ρ_{59} den niedrigsten Permanance-Wert.

$$perm_{true}(\rho_{59}, W) = opt_{true}(\rho_{59}, W) - opt_{true}(\rho_{110}, W)$$
$$perm_{true}(\rho_{59}, W) = -0,5 - 1 = -1,5$$

Wird ein solcher Schritt des Vergessens durchgeführt, verändert sich der Coverage Graph. Um Informationen über den Support nicht zu verlieren, muss beim Vergessen einer Regel dessen Support neu im Coverage Graph angeordnet werden. Dabei unterscheiden Martínez-Plumed et al. (2015) zwischen zwei Arten:

- Wird ein Knoten entfernt, der kein Blattknoten ist, beispielsweise ρ_{59} , wird der Support seiner Nachfahren ρ_2 an dessen Vorfahren ρ_{20} sowie die Vorfahren des zu entfernenden Knotens ρ_{73} verteilt.
- Beim Entfernen eines Blattknotens wird dessen Support zu gleichen Teilen an die Regeln verteilt, die den entfernten Knoten direkt abdecken. Jene erhalten diesen Teil als zusätzlichen eignen Support res_c.

Dadurch entsteht für den Support die neue Gleichung 17 mit dem initialen Wert $res_c = 0$.

$$\mathring{S}_c(\rho, W) = \begin{cases} L(\rho) & \text{wenn } \rho \in leaves \\ res_c + \sum_{\nu \in suc(\rho)} \frac{\mathring{S}_c(\nu, W)}{|anc(\nu)|} & \text{ansonsten} \end{cases} \tag{17}$$

3.4 Speichern und Vergessen von Wissen

Diese Bewertungskriterien können nun für das Übernehmen von Regeln in das konsolidierte Wissen verwendet werden, sowie für deren Vergessen.

Regeln mit guten Werten ihrer Bewertungskriterien haben gute Chancen als konsolidiertes Wissen gespeichert und ab diesem Zeitpunkt als wahr betrachtet zu werden. Da die deduktive Engine die Coverage anhand des konsolidierten Wissen berechnet, würde eine irrtümlich aufgenommene Regel zu weitreichenden Inkonsistenzen führen. Martínez-Plumed et al. (2015) schlagen einen Threshold vor, um mit Hilfe der Optimality eine Regel in das konsolidierte Wissen aufzunehmen. Nachdem eine Regel in das konsolidierte Wissen übergegangen ist, kann sie nicht mehr vergessen werden. Um irrtümlich aufgenommene Regeln doch wieder vom konsolidierten Wissen zu entfernen setzen Martínez-Plumed et al. (2015) gespiegelt zu ihrem Promotionssystem ein Demotionssystem ein, welches erlaubt, Regeln des konsolidierten Wissens wieder in den Arbeitsbereich

zu degradieren. Ist eine Regel wieder im Arbeitsbereich, kann sie von diesem aus vergessen werden. Somit kann verhindert werden, dass eine falsche Regel für immer im konsolidierten Wissen verankert bleibt. Nur das ursprüngliche konsolidierte Wissen zum Startzeitpunkts des Systems, ist von einer solchen Demotion ausgeschlossen.

4 Fazit

Das Konzept für ein inkrementell lernendes System von Martínez-Plumed et al. (2015) ist gut umgesetzt um das von Carpenter und Grossberg (1988) beschriebene Problem des „stability-plasticity dilemma“ anzugehen. Es ist sinnvoll für fortlaufend lernende Systeme Beispiele und Hypothesen in das vorhandene Wissen zu integrieren, um mit dem vorhandenen Wissen konsistente Hypothesen zu erkennen, ohne bei einer neuen Erkenntnis erst im Nachhinein die Konsistenz überprüfen zu können.

Der dabei wichtigste Punkt hierfür liegt meiner Meinung nach darin, dass Martínez-Plumed et al. (2015) versucht haben den Schwerpunkt darauf zu setzten, möglichst gute Generalisierungen zu finden. Eine Regel die viele Beispiele einer Klasse mit einer hohen Optimality modelliert, ist als gute Abstraktion dieser Beispiele anzusehen.

Durch die Möglichkeit des Vergessens, wird dem lernenden System die Option gegeben redundante und falsche Regeln und Hypothesen schnell wieder zu verwerfen. Die Flexibilität des Systems hebt sich auch durch den Demotionsprozess des gefestigten Wissens hervor, da sogar zu einem früheren Zeitpunkt als wahr angenommene Regeln wieder entfernt werden können, was dem System beispielsweise erlaubt besser mit Concept Drifts umzugehen.

Das Prinzip der Minimal Message Length sehe ich ebenfalls als gut gewählte Methodik, um Regeln zu bewerten, da kurze Regeln im allgemeinen weniger anfällig für Overfitting sind. Auch hier kann der Nutzer das System einfach für seine Anwendung anpassen. Wie genau die Länge einer Regel bestimmt wird und von welchen Parametern sie abhängt kann der Nutzer beispielsweise dem Rechenaufwand der Applikation, welche das Wissen verarbeitet, anpassen.

Die Flexibilität des Ansatzes zeichnet sich auch dadurch aus, dass die induktive sowie die deduktive Engine, sowie weitere Komponenten, wie der Threshhold für den Promotion-Prozess oder die bereits erwähnte Längenberechnung einer Regel austauschbar oder anpassbar sind. Beispielsweise kann man je nach Klasse unterscheiden wie wichtig die Reinheit der Abdeckung einer Regel für die Berechnung der Optimality ist. Der von Martínez-Plumed et al. (2015) verwendete Faktor β unterscheidet nur zwischen zur Klasse gehörenden und nicht dazugehörenden Support-Werten (vgl. Gleichung 12). Hier kann man klassenbezogen weitere Anpassungen vornehmen, da für einen Anwendungsfall möglicherweise eine Klasse von keiner nicht dazugehörigen Regel abgedeckt werden darf, da es sich um ein sehr sicherheitskritisches System handelt.

Nach meinem Befinden ist der größte Schwachpunkt dieses Ansatzes das Hintergrundwissen, welches zum Startzeitpunkt des lernenden Systems benötigt

wird. Andere Ansätze wie beispielsweise das Deep Learning benötigen zwar kein derartiges Hintergrundwissen, sind jedoch deutlich schwieriger für den Menschen nachzuvollziehen. Daher ist dieser Ansatz jedoch für alle die transparente Lösungen für fortlaufend lernende Systeme suchen, sehr empfehlenswert.

Literatur

Carpenter, G. A. & Grossberg, S. (1988). The art of adaptive pattern recognition by a self-organizing neural network. *Computer*, *21*, 77-88.

Gama, J. a., Žliobaitė, I., Bifet, A., Pechenizkiy, M. & Bouchachia, A. (2014, März). A survey on concept drift adaptation. *ACM Comput. Surv.*, *46* (4), 44:1–44:37. Zugriff auf `http://doi.acm.org/10.1145/2523813` doi: 10.1145/2523813

Martínez-Plumed, F. (2016). *Incremental and developmental perspectives for general-purpose learning systems* (Unveröffentlichte Dissertation). Technical University of Valencia, Spain.

Martinez Plumed, F., Ferri, C., Hernandez Orallo, J. & Ramirez Quintana, M. (2014, Dec). A knowledge growth and consolidation framework for lifelong machine learning systems. In *Machine learning and applications (icmla), 2014 13th international conference on* (S. 111-116).

Martínez-Plumed, F., Ferri, C., Hernández-Orallo, J. & Ramírez-Quintana, M. J. (2015). Knowledge acquisition with forgetting: an incremental and developmental setting. *Adaptive Behavior*, *23* (5), 283-299.

Mitchell, T. M. (2005). *Machine learning* (internat. ed., [Nachdr.] Aufl.). New York [u.a.]: McGraw-Hill. (XVII, 414 S. : graf. Darst.)

Muggleton, S. & de Raedt, L. (1994). Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, *19-20*, 629 - 679. Zugriff auf `http://www.sciencedirect.com/science/article/pii/0743106694900353` (Special Issue: Ten Years of Logic Programming) doi: https://doi.org/10.1016/0743-1066(94)90035-3

Thrun, S. (1996). Is learning the n-th thing any easier than learning the first? In D. Touretzky & M. Mozer (Hrsg.), *Advances in neural information processing systems (nips) 8* (S. 640–646). Cambridge, MA: MIT Press.

Wallace, C. S. & Boulton, D. M. (1968). An information measure for classification. *The Computer Journal*, *11* (2), 185-194. Zugriff auf `+http://dx.doi.org/10.1093/comjnl/11.2.185` doi: 10.1093/comjnl/11.2.185