Effektives Widening mit Hashbasierter Partitionierung des Hypothesenraums

Alexander Fillbrunn[1]

Abstract: Vielen Algorithmen im Data Mining basieren auf gierigem Verhalten, um ein ausreichend gutes Modell schnell zu finden. Bei der Verwendung solcher Greedy-Algorithmen besteht allerdings die Gefahr, dass diese in lokalen Optima stecken bleiben. Mit Hilfe von Widening, einer Technik um den Hypothesenraum breiter zu durchsuchen, kann diese Gefahr verringert werden, indem parallel mehrere, möglichst unterschiedliche, Modelle erzeugt werden. Bisherige Verfahren im Widening leiden jedoch unter zweierlei Problemen: dem Overhead durch die Kommunikation zwischen den parallelen Recheneinheiten und der Notwendigkeit, Modelle bezüglich ihrer Ähnlichkeit miteinander vergleichen zu können. In der vorliegen Arbeit wird mit dem Bucket-Selektor ein randomisiertes und modellunabhängiges Widening-Verfahren vorgestellt, das schneller zu ähnlich guten Ergebnissen gelangt wie bisherige Verfahren.

1 Einleitung

Mit dem Aufkommen von Multiprozessorsystemen, Rechenclustern und verteilten Systemen wie Hadoop, Spark und Flink begann vor einigen Jahren die Arbeit daran, bisher sequentielle Algorithmen an diese neuen Gegebenheiten anzupassen und die vorhandene parallele Rechenkraft effizient auszunutzen. Das Hauptaugenmerk wurde dabei fast ausschließlich darauf gelegt, die Algorithmen zu beschleunigen und dabei die gleichen Ergebnisse wie bisher zu erzielen. Wenig Beachtung wurde dagegen dem Ansatz geschenkt, die zusätzlichen Recheneinheiten zu verwenden, um die Ergebnisse existierender Algorithmen zu verbessern und dabei die bisherige Geschwindigkeit beizubehalten. Auch im maschinellen Lernen fokussierte sich die Forschung lange Zeit auf die möglichen Geschwindigkeitszuwächse, obwohl auch die Genauigkeit der dort verwendeten Vorhersagealgorithmen von einer Parallelisierung profitieren kann.

Ein im maschinellen Lernen häufig anzutreffendes Problem ist die Suche nach einer den vorhandenen Daten zugrundeliegenden Struktur, beziehungsweise einem Modell, das diese Struktur beschreibt und das später für Vorhersagen oder für das bessere Verständnis der vorliegenden Daten herangezogen werden kann. Ein Algorithmus, der ein solches Modell automatisch lernt, legt sich meistens am Anfang des Lernprozesses auf einen Modelltyp fest, der durch Anpassung verschiedener Parameter so optimiert wird, dass die Struktur der Daten durch das Modell erklärt werden kann. Die optimalen Parameter für einen Modelltyp zu finden ist allerdings ein aufwändiger Prozess, der meist das Durchsuchen vieler verschiedener Wertekombinationen voraussetzt. Durch Ausprobieren kleiner Veränderungen der Parameter kann ein Algorithmus ein Modell Schritt für Schritt verbessern, nicht im-

[1] KNIME GmbH, alexander.fillbrunn@knime.com

mer führt dies aber zu einem optimalen Ergebnis. Stellt man sich den Parameterraum eines Modells als Landschaft vor, in der die Qualität des von den Parametern beschriebenen Modells die Höhe bestimmt, dann verhindern verschiedene topologische Strukturen die Konvergenz zum besten Modell, dem *globalen Optimum*. Das beste Beispiel für eine solche hinderliche Struktur ist ein *lokales Optimum*: ein Bereich im Parameterraum, der ein besseres Modell repräsentiert als alle Parameterkombinationen um ihn herum. Ein Algorithmus, der nur kleine Änderungen an den Parametern vornimmt und diese übernimmt, wenn sich das Modell dadurch verbessert, wird nie aus solch einer Region herauskommen.

Die Idee des *Widening* [Ak00] ist es, parallele Recheneinheiten gleichzeitig auf die Suche nach einer optimalen Lösung für ein Problem des maschinellen Lernens einzusetzen. Diese Lösungen sind im Allgemeinen Modelle, die im *Hypothesenraum* gesucht werden. Man möchte im Widening bessere Modelle finden, statt einen existierenden Algorithmus bei gleichem Ergebnis schneller zu machen („Better, not faster“ [AIB12]). Um dieses Ziel zu erreichen muss eine Balance zwischen der Exploration des Hypothesenraums und dem Fokus auf qualitativ hochwertige Modelle gefunden werden. Es handelt sich also um eine Optimierung mit mehreren Zielen, auch Pareto-Optimierung [Eh05] genannt.

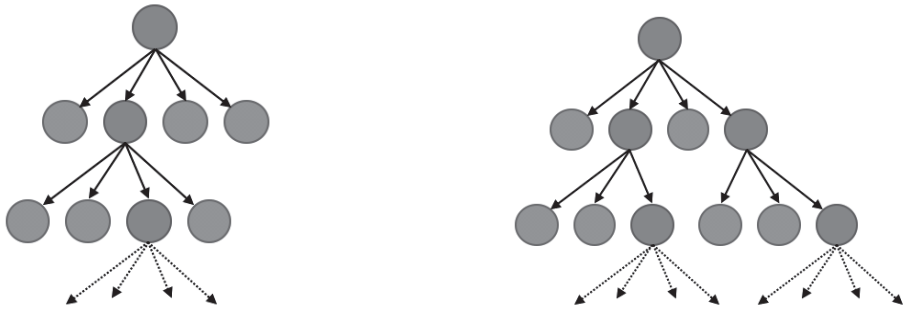
In dieser Zusammenfassung der Hauptarbeit [Fi19] soll zuerst das theoretische Konzept des *Widening* kurz erklärt und Probleme bisheriger Vorgehensweisen in diesem Bereich hervorgehoben werden. Anschließend wird der *Bucket-Selektor* als Hauptbeitrag der Arbeit vorgestellt. Es handelt sich um ein Widening-Verfahren, das schneller als bisherige Ansätze ist und auf zwei untersuchten Anwendungsfällen, dem hierarchischen agglomerativen Clustering und der Join Order Optimierung, gleich gute oder bessere Ergebnisse liefert. Von den beiden Anwendungsfällen wird hier aus Platzgründen nur das Clustering vorgestellt.

2 Widening

Viele Algorithmen im Data Mining und anderen Bereichen, die einen Hypothesenraum nach einem günstigen Modell durchsuchen, gehen dabei gierig, beziehungsweise greedy, vor und führen eine lokale Suche aus. Oft werden Greedy-Algorithmen verwendet, wenn eine vollständige Auflistung aller Modelle des Hypothesenraums zu zeitaufwändig ist. Greedy-Algorithmen sind meist schnell, führen aber nicht zwangsläufig zu einem optimalen Endergebnis. In vielen Fällen haben Greedy-Algorithmen das Problem, dass sie in einem Zwischenschritt eine zu diesem Zeitpunkt optimal erscheinende Entscheidung treffen, von welcher sich aber später herausstellt, dass eine andere Entscheidung längerfristig zu einem besseren Ergebnis geführt hätte. Der Algorithmus verfängt sich also in einem lokalen Optimum.

Im Widening [AIB12] sollen die Auswirkungen eines solchen Greedy-Verhaltens abgemildert werden, indem parallel mehrere Pfade durch den Hypothesenraum traversiert werden, wie es in Abbildung 1 zu sehen ist. Dabei ist es wichtig sicherzustellen, dass parallele Recheneinheiten nicht dieselben und idealerweise sogar möglichst diverse Lösungsansätze verfolgen, um die Gefahr, dass alle in das gleiche lokale Optimum fallen, zu verringern.

Es handelt sich gewissermaßen um einen Kompromiss zwischen der schnellen aber sehr begrenzten Greedy-Suche und der ausführlichen und sehr langsamen vollständigen Suche.



(a) Greedy (b) Widening

Abb. 1: Konzept des Widening, in welchem statt eines Pfades durch den Hypothesenraum parallel mehrere verfolgt werden.

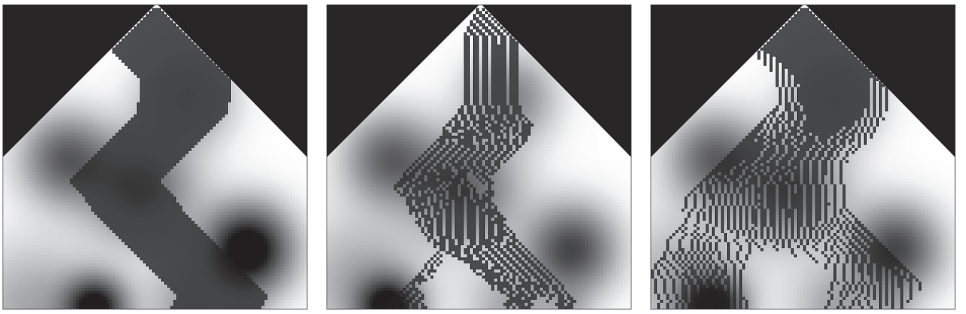
Die beiden Grundbausteine für die Formalisierung des Widening sind die beiden Operatoren *Verfeinerung* $r(\cdot)$ und *Selektion* $s(\cdot)$, deren Zweck in Abbildung 1 zu erkennen ist: Die Verfeinerung erzeugt aus einem Modell entsprechende Nachfolger (zum Beispiel das Hinzufügen eines neuen Asts in einem Entscheidungsbaum oder das Zusammenfügen zweier Cluster im hierarchischen Clustering). Bei der Selektion werden anschließend aus allen Verfeinerungen eine (Greedy) oder mehrere (Widening) ausgewählt. Einen Schritt in einem Greedy-Algorithmus kann mit der Formel $m' = s(r(m))$ beschrieben werden. Hierbei ist m ein Modell und m' dessen Nachfolger, der durch das Anwenden des Selektors auf die Verfeinerungen von m erzeugt wird. Der Greedy-Algorithmus setzt hier bei der Selektion auf eine heuristische Bewertungsfunktion ψ, die jedem Modell einen Qualitätswert zuweist. Das Modell mit der höchsten Bewertung wird in der nächsten Iteration weiterverfolgt. Im Widening dagegen werden mehrere Modelle verfolgt und aus der Greedy-Formel wird die Folgende:

$$\{m'_1, \ldots, m'_k\} = s_{global}(s_{lokal}(r(m_1)) \cup \ldots \cup s_{lokal}(r(m_k))) \; . \tag{1}$$

Hier wird auf Modellmengen gearbeitet und in jedem Schritt k Modelle verfeinert und ausgewählt. k sollte hierbei mindestens so groß wie die Anzahl der verfügbaren parallelen Recheneinheiten sein. Die Selektion wird dabei außerdem in lokale und globale Selektion aufgeteilt, da die Modelle auf die Recheneinheiten verteilt werden und diese bereits lokal vorselektieren können, bevor eventuell an zentraler Stelle die Modelle für die nächste Iteration ausgewählt werden. Benötigt wird die globale Selektion beispielsweise, um sicherzustellen, dass sich die Modelle der einzelnen Recheneinheiten nicht zu ähnlich werden.

Während der Verfeinerungsoperator oft durch den zugrundeliegenden Lernalgorithmus vorgegeben ist, ist die Wahl des Selektors entscheidend für das Erreichen guter Ergebnisse. Im Folgenden sollen einige Selektionsmethoden kurz vorgestellt werden. Zur Veranschaulichung dient hier ein einfaches Beispielproblem, bei dessen Zustandsgraphen es sich um einen endlichen, gerichteten und azyklischen Graphen handelt. Jeder Zustand wird durch

einen Pixel der Visualisierung repräsentiert und je dunkler der Pixel, desto höher die heuristische Bewertung. Jeder Zustand besitzt maximal 3 Nachfolger: die 3 Pixel unter ihm. Der Graph hat genau einen Startzustand und 199 Endzustände und alle Pfade zwischen dem Startzustand und allen Endzuständen haben die Länge 100. Die Algorithmen durchlaufen den durch das Raster repräsentierten Graphen von oben nach unten.



(a) Top-k (b) Diverse Top-k (c) Score erosion mit $\beta = 1$

Abb. 2: Explorationsverhalten verschiedener Selektoren. Dunkle Pixel symbolisieren bessere Zustände und die Algorithmen durchlaufen das Raster von oben nach unten.

Der Top-k-Algorithmus [IB13] ist die naive Erweiterung des einfachen Greedy-Algorithmus, denn statt eines besten Folgemodells werden hier die k besten ausgewählt. Da in vielen Problemen ähnliche Modelle auch ähnliche Bewertungen durch die Bewertungsfunktion erhalten, bedeutet das oft, dass der Top-k-Algorithmus sehr ähnliche Modelle wie der einfache Greedy-Algorithmus exploriert und wenig divers ist. Die Top-k-Selektion ist jedoch sehr schnell und kann in $\mathcal{O}(n \log k)$ mit Hilfe einer Min-Heap-Struktur durchgeführt werden. Wenn die Modelle von parallelen Recheneinheiten berechnet und gespeichert werden, dann kann zuerst jede Einheit lokal die besten k Modelle in ihrer Partition finden, bevor dann k^2 Modelle zentral gesammelt und davon wiederum k Modelle für die nächste Iteration ausgewählt werden können. Weil Modelle zentral gesammelt und wieder an die Recheneinheiten verteilt werden müssen, benötigt das Top-k-Verfahren Kommunikation mit der Komplexität $\mathcal{O}(k^2)$ zwischen den Einheiten.

Diverse-Top-k [IB13] ist eine Erweiterung des Top-k-Algorithmus. Das Verfahren benötigt eine Distanzfunktion δ und einen Schwellenwert θ und wählt die besten Modelle so aus, dass jedes ausgewählte Modell mindestens die Distanz θ von jedem anderen ausgewählten Modell hat. Da lokal und global unterschiedliche Schwellwerte verwendet werden können, sind hier s_{local} und s_{global} nicht zwingend gleich. Die Laufzeitkomplexität des Diverse-Top-k-Selektors entspricht $\mathcal{O}(n \log n + nk)$, da die Modelle sortiert werden und anschließend jedes Modell mit den bereits ausgewählten verglichen werden muss. Durch die Sortierung beträgt die Speicherkomplexität mindestens $\mathcal{O}(n)$ und wie bei Top-k werden in jeder Iteration $(k+1)(k-1) \in \mathcal{O}(k^2)$ Modelle ausgetauscht, weil die Modelle zentral gesammelt und neu verteilt werden müssen.

Der *Score-Erosion*-Algorithmus [Me10] wurde für die Auswahl aktiver und diverser Moleküle im Hochdurchsatz-Screening (*High-Throughput Screening*) entwickelt. Der dort durchsuchte Molekülraum enthält hauptsächlich inaktive Moleküle mit einigen sogenann-

ten *Aktivitätsinseln* (*activity islands*), also Nachbarschaften von Molekülen mit hoher Aktivität. Das Ziel des Score-Erosion-Algorithmus ist es, Moleküle mit hoher Aktivität zu finden und dabei zu vermeiden, mehrere Moleküle einer Aktivitätsinsel auszuwählen. Das Ergebnis ist eine Auswahl aktiver und diverser Moleküle. Um dieses Ziel zu erreichen, wählt der Algorithmus iterativ das Molekül mit der höchsten Aktivität und verringert (oder erodiert) anschließend die Aktivität der anderen Moleküle abhängig von deren Distanz zum gerade ausgewählten. Ein Parameter β kontrolliert die Stärke der Erosion. Der Score-Erosion-Selektor hat eine Laufzeitkomplexität von $\mathcal{O}(nk)$ und eine Speicherkomplexität von $\mathcal{O}(n)$. Parallelisieren lässt sich dieser Selektor nur schwierig, denn nach jeder Auswahl muss die Bewertung aller Modelle angepasst und daraufhin das global beste gefunden werden.

Die Selektoren, die in diesem Abschnitt vorgestellt wurden, haben Nachteile in Bezug auf die Laufzeit, Kommunikation, Parallelisierbarkeit und Speicherverbrauch. Besonders ungünstig sind Methoden, die die Verfeinerungen mehrfach durchlaufen müssen, oder die Zugriff auf alle Verfeinerungen auf einer zentralen Recheneinheit benötigen. Des Weiteren ist das Berechnen von Distanzen zwischen Modellen äußerst kostspielig. Der im nächsten Abschnitt vorgestellte Bucket-Selektor besitzt diese Nachteile nicht.

3 Der Bucket-Selektor

Der Bucket-Selektor unterteilt einerseits den Suchraum in zufällige Partitionen, die von unterschiedlichen Recheneinheiten durchsucht werden, vermeidet aber andererseits das Problem der Nicht-Erreichbarkeit von Modellen an Partitionsgrenzen, indem er einen Synchronisationsschritt zwischen den Einheiten durchführt, bei welchem Modelle ausgetauscht werden. Die Diversität der Auswahl wird bei dem Bucket-Selektor durch eine hashbasierte Partitionierungsfunktion erreicht. Der Selektor garantiert, dass das beste Modell immer und das *i*te Modell mit abnehmender Wahrscheinlichkeit ausgewählt wird. Somit haben auch Modelle, die nicht unter den besten k sind, eine Chance, in die nächste Iteration übernommen zu werden. Wie andere Widening-Selektoren auch geht der Bucket-Selektor level-synchron vor. Das heißt, dass die Iterationen der einzelnen Recheneinheiten durch einen Austauschschritt am Ende jeder Iteration synchronisiert werden. Dies entspricht auch einem möglichen Vorgehen bei der parallelen Breitensuche [BM11].

Der Bucket-Selektor partitioniert den Suchraum mit Hilfe einer Hashfunktion, die jedem Modell einen ganzzahligen Wert zwischen 1 und k zuweist, wobei idealerweise k parallele Recheneinheiten zur Verfügung stehen. Dann ist jede Recheneinheit ist für die Modelle genau einer Partition zuständig. In jeder Iteration verfeinert jede Recheneinheit zuerst ihr aktuelles Modell, dann werden die Verfeinerungen gehasht und für jeden Hashwert das beste Modell ermittelt. Anschließend werden diese besten Modelle an die jeweils verantwortliche Recheneinheit geschickt. Diese kann dann aus allen lokal besten Modellen das global beste auswählen und in der nächsten Iteration weiter verfeinern. Das Verfahren wird in Abbildung 3 dargestellt.

Dadurch, dass aus jeder Partition das beste Modell ausgewählt wird und die Partitionen zufällig mit Hilfe der Hashfunktion ermittelt werden, konkurriert bei N Modellen jedes

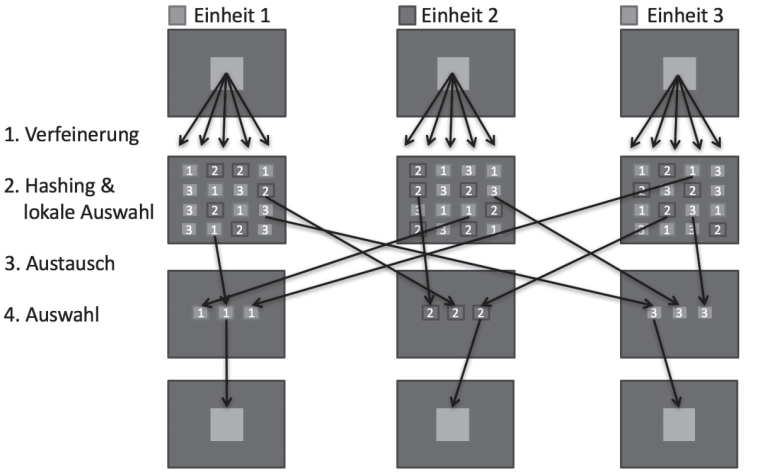


Abb. 3: Das Bucket-Selektionsverfahren. Jede Recheneinheit verfeinert ihr Modell, berechnet Hashwerte für die Verfeinerungen und wählt aus jeder Partition das beste Modell. Anschließend werden die Modelle so ausgetauscht, dass jede Einheit alle lokal besten Modelle ihrer Partition hält. Aus diesen wählt sie das insgesamt beste aus.

Modell mit $\frac{N}{k} - 1$ anderen Modellen um den ersten Platz in der jeweiligen Partition. Wichtig ist deshalb, dass die Werte der Hashfunktion möglichst gleichmäßig verteilt sind, also dass $P(h(m) = i) \approx \frac{1}{k}$, damit jedes Modell die gleiche Chance hat, das beste Modell seiner Partition zu sein. Eigenschaften kryptographischer Hashfunktionen, wie die Einweg-Eigenschaft und das Bestreben, möglichst wenige Kollisionen zuzulassen, spielen bei der Auswahl der Hashfunktion keine Rolle.

Bei N verschiedenen Modellen, aus denen k ausgewählt werden sollen, konkurriert jedes Modell im Mittel mit $\frac{N}{k}$ Modellen, solange die Hashfunktion gleichmäßig verteilte Werte erzeugt. Je besser also ein Modell verglichen mit allen anderen ist, desto wahrscheinlicher wird es ausgewählt. Da aber durch die zufällige Partitionierung auch ein schlechteres Modell das beste in seiner Partition sein kann, werden nicht nur die besten k Modelle gewählt. Für jedes Modell m gibt es genau $\mathrm{rg}_\psi(m) - 1$ Modelle, die eine bessere Bewertung $\psi(m)$ haben. Für jedes dieser Modelle beträgt die Wahrscheinlichkeit, in einer anderen Partition als m zu landen $(1 - \frac{1}{k})$. Folgende Formel kann somit verwendet werden, um die Auswahlwahrscheinlichkeit P_s für m zu berechnen:

$$P_s(m) = \left(1 - \frac{1}{k}\right)^{\left(\mathrm{rg}_\psi(m)-1\right)} . \tag{2}$$

Wie hier ersichtlich ist, beträgt die Auswahlwahrscheinlichkeit für das beste Modell mit $\mathrm{rg}_\psi = 1$ genau $\left(1 - \frac{1}{k}\right)^0 = 1$. Diese Garantie, dass das beste Modell immer ausgewählt wird, unterscheidet den Bucket-Selektor von anderen stochastischen Selektoren.

Im Gegensatz zu den meisten anderen in Abschnitt 2 vorgestellten Selektoren partitioniert der Bucket-Selektor durch seine Hashfunktion den Suchraum in k Partitionen, wobei jede Partition als individuellen Selektionsoperator einen Bucket-Selektor besitzt, der nur das beste Modell aus einem bestimmten Bucket behält und die anderen Modelle an die entsprechende Recheneinheit weiterleitet. Somit ist ein zentrales Sammeln der Modelle nicht nötig. Des Weiteren besitzt die lokale Selektion hier nur eine Laufzeitkomplexität von $\mathcal{O}(n)$, da für jedes Modell nur einmal die Hashfunktion evaluiert werden muss, um für jede Partition das beste Modell zu finden.

4 Hierarchisches Clustering mit dem Bucket-Selektor

Die hierarchische Clusteranalyse [RK90] ist ein Verfahren, in welchem eine Baumstruktur erzeugt wird, die die einzelnen Datenpunkte als Blattknoten und Cluster, also Gruppen von Datenpunkten, als innere Knoten besitzt. Die Evaluierung des Bucket-Selektors befasst sich im Folgenden mit dem agglomerativen hierarchischen Clustering. Hier ist zu Beginn jeder Datenpunkt ein Cluster und mit jeder Iteration des Algorithmus werden die zwei am nächsten beieinander liegenden Cluster zu einem zusammengefasst, bis am Ende ein einzelner Cluster mit allen Datenpunkten entsteht. Anschließend kann das entstandene Dendrogramm an beliebiger Stelle horizontal geteilt werden, um die gewünschte Anzahl an Clustern zu erhalten. Dabei handelt es sich um ein gieriges Vorgehen, da die Entscheidung darüber, welche Cluster miteinander verbunden werden, anhand lokaler Qualitätskriterien getroffen wird.

Die Qualität eines Clusterings ist meist sehr subjektiv und hängt von den gewünschten Eigenschaften der Gruppierung ab. Ist die gewünschte Anzahl von Clustern bekannt, kann man für die Evaluierung des Clusterings Maße wie den Silhouettenkoeffizient [Ro87] und den Davies-Bouldin Index [DB79] verwenden. Bei beiden handelt es sich um interne Qualitätsmaße, das heißt, es gibt keine *Ground Truth* und die Qualität wird allein über die Daten und die ihnen zugewiesenen Cluster bestimmt, ohne externe Informationen miteinzubeziehen.

Soll stattdessen das vom hierachischen agglomerativen Clustering erzeugte Dendrogramm bewertet werden, sind diese Maße nur bedingt geeignet. Stattdessen kann hier auf den *kophenetischen Korrelationskoeffizienten* [SR62] zurückgegriffen werden. Allgemein ist zu erwarten, dass im hierachischen Clustering ähnliche Datenpunkte früh im gleichen Cluster landen. Entsprechend kann man die Höhe des Knotens im Dendrogramm, in welchem zwei Datenpunkte zusammenkommen, als eine durch das Dendrogramm induzierte ultrametrische Distanz betrachten [Mi79]. Der kophenetischen Korrelationskoeffizient berechnet sich als die Korrelation dieser Distanzen mit den ursprünglichen, bekannten, Distanzen zwischen den Datenpunkten.

Bei der Evaluation des Bucket-Selektors wurde das Hauptaugenmerk auf den Vergleich bezüglich Qualität der erzeugten Modelle gelegt, weshalb in dieser Zusammenfassung nur darauf und nicht auf die Laufzeituntersuchungen eingegangen werden soll. Aus Platzgründen wird der Bucket-Selektor im Folgenden lediglich mit dem Greedy-Algorithmus

und mit Top-k verglichen. Für weitere Vergleiche mit explizit diversitätserhaltenden Selektoren wie Diverse Top-k und Score Erosion wird auf die Hauptarbeit [Fi19] verwiesen. Die Experimente wurden auf 7 verschiedenen öffentlich verfügbaren Datensätzen durchgeführt: Seeds, Ruspini, Iris, User Knowledge Modeling (U.K.M.), Breast Cancer (B.C.), SPECTF und Cement. Deren Quellen sind ebenfalls in der Hauptarbeit [Fi19] zu finden.

Alle hier vorgestellten auf hierarchisches Clustering basierenden Vergleiche wurden mit UPGMA-Linkage durchgeführt, da diese mit dem Greedy-Algorithmus von allen Linkage-Kriterien auf den getesteten Datensätzen den besten kophenetischen Korrelationskoeffizienten erzeugte. Als Distanzmetrik für das Clustering wurde die Euklidische Distanz verwendet. Tabelle 1 fasst die Ergebnisse des Vergleichs zusammen. Bis auf den SPECTF-Datensatz kann der Bucket-Selektor in allen Datensätzen bessere Ergebnisse erreichen als Greedy und Top-k. Der SPECTF-Datensatz ist insofern speziell, dass er 44 Dimensionen hat, während die anderen Datensätze maximal 10-dimensional sind. Bekanntermaßen ist die Euklidische Distanz für hochdimensionale Daten nicht gut geeignet, weshalb davon auszugehen ist, dass aus diesem Grund auch das parallele Lernen mehrerer Dendrogramme keine Verbesserung bringt.

k	Datens.	Bucket			Top-k			Greedy		
		μ	σ	Med.	μ	σ	Med.	μ	σ	Med.
100	Seeds	**0,783**	0,025	**0,784**	0,743	0,009	0,744	0,735	0,015	0,739
	Ruspini	**0,883**	0,014	**0,878**	0,876	0,004	0,876	0,875	0,006	0,875
	Iris	**0,855**	0,006	**0,855**	0,852	0,006	0,852	0,853	0,007	0,853
	U.K.M.	**0,583**	0,013	**0,584**	0,578	0,013	0,581	0,574	0,014	0,574
	B.C.	**0,863**	0,007	**0,864**	0,843	0,006	0,844	0,839	0,007	0,84
	SPECTF	0,94	0,004	0,94	0,94	0,004	0,94	**0,943**	0,006	**0,944**
	Cement	**0,706**	0,011	**0,706**	0,703	0,01	0,703	0,704	0,012	0,705
500	Seeds	**0,8**	0,017	**0,801**	0,744	0,008	0,744	0,736	0,012	0,738
	Ruspini	**0,9**	0,019	**0,896**	0,877	0,004	0,876	0,875	0,006	0,875
	Iris	**0,86**	0,006	**0,86**	0,851	0,005	0,852	0,852	0,006	0,852
	U.K.M.	**0,582**	0,01	**0,58**	0,579	0,013	0,579	0,577	0,012	0,576
	B.C.	**0,869**	0,005	**0,869**	0,844	0,006	0,845	0,839	0,008	0,84
	SPECTF	0,94	0,004	0,94	0,94	0,005	0,94	**0,942**	0,006	**0,943**
	Cement	**0,709**	0,011	**0,709**	0,703	0,012	0,702	0,703	0,012	0,702

Tab. 1: Versuchsergebnisse für den Top-k und den Hash-Bucket-Selektor mit $k = 100$ und $k = 500$. Durchschnitt (μ), Standardabweichung (σ) und Median ermittelt über 100 Durchläufe mit je zufälligen 90 % der Daten. Für den Greedy-Selektor gilt immer $k = 1$.

Abbildung 4 zeigt die Ergebnisse für den Breast-Cancer-Datensatz mit feineren Abstufungen für k. Zu sehen ist, dass die Top-k-Selektion zwar leicht besser ist als der Greedy-Algorithmus, von mehr gleichzeitig explorierten Modellen aber kaum profitiert. Der Bucket-Selektor dagegen erzielt mit höherem k auch bessere Ergebnisse.

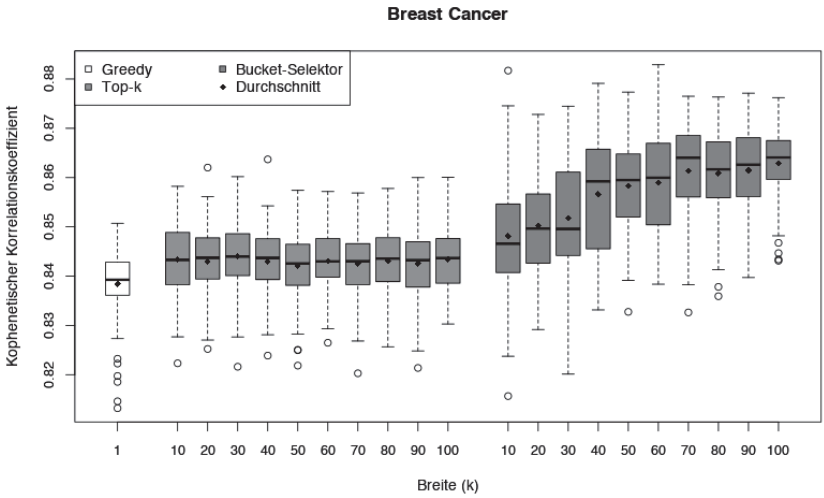


Abb. 4: Ergebnisse für das hierarchische Clustering des Breast-Cancer-Datensatzes und UPGMA-Linkage mit dem Greedy-, Top-*k*- und Bucket-Selektor.

5 Zusammenfassung

Widening betrachtet die lokale Suche als eine Abfolge von Verfeinerungs- und Selektionsschritten. In der Verfeinerung werden alle Nachbarn der aktuell betrachteten Modelle aufgelistet, in der Selektion werden einige dieser Nachbarn dann für die nächste Iteration ausgewählt. Während die Verfeinerungsoperation meist anwendungsabhängig ist und kein Optimierungspotential hat, spielt die Wahl eines passenden Selektors eine große Rolle für die Qualität der erzeugten Lösungen eines Algorithmus. Publizierte Selektoren sind beispielsweise Top-k, Diverse Top-k, K-Medoid und Score Erosion.

Der hier vorgestellte Hash-Bucket-Selektor ist ein Selektor für das Widening, der in jeder Iteration jede Verfeinerung der parallel verarbeiteten Modelle per Hashfunktion einer Partition zuweist und anschließend das beste Modell aus jeder Partition für die nächste Iteration auswählt. Dies garantiert einerseits, dass das global beste Modell immer ausgewählt wird, aber auch Modelle außerhalb der besten k eine Chance haben, weiterverfolgt zu werden. Gegenüber anderen Selektoren hat der Bucket-Selektor den Vorteil, dass Modelle nie zentral gesammelt werden müssen, sondern immer direkt zwischen den Recheneinheiten ausgetauscht werden können. Des Weiteren besitzt er eine geringere theoretische Laufzeitkomplexität als Top-k und benötigt ebensowenig Speicher, da die Verfeinerungen des Vorgängermodells nur einmal durchlaufen werden müssen. Im Gegensatz zu Selektoren wie K-Medoid oder Score Erosion benötigt der Bucket-Selektor keine Distanzmetrik für Modelle, da allein anhand der Hashfunktion bestimmt wird, welche Modelle anhand ihrer Qualität miteinander verglichen werden. Kann gar keine Hashfunktion für die Modelle eines Problems gefunden werden, so ist es auch möglich, die Hashfunktion durch eine Zufallsfunktion zu ersetzen und in vielen Fällen trotzdem gute Ergebnisse zu erzielen.

Literaturverzeichnis

[AIB12] Akbar, Zaenal; Ivanova, Violeta N.; Berthold, Michael R.: Parallel data mining revisited. Better, not faster. In: International Symposium on Intelligent Data Analysis. Springer, S. 23–34, 2012.

[Ak00] Akl, Selim G: Parallel real-time computation: Sometimes quantity means quality. In: Proceedings of the International Symposium on Parallel Architectures. IEEE, S. 2–11, 2000.

[BM11] Buluç, Aydin; Madduri, Kamesh: Parallel breadth-first search on distributed memory systems. In: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis. ACM, S. 65, 2011.

[DB79] Davies, David L; Bouldin, Donald W: A cluster separation measure. IEEE transactions on pattern analysis and machine intelligence, PAMI-1(2):224–227, 1979.

[Eh05] Ehrgott, Matthias: Multicriteria optimization, Jgg. 491. Springer Science & Business Media, 2005.

[Fi19] Fillbrunn, Alexander: Effektives Widening mit Hashbasierter Partitionierung des Hypothesenraums. in press, University of Konstanz, 2019.

[IB13] Ivanova, Violeta N.; Berthold, Michael R.: Diversity-driven widening. In: International Symposium on Intelligent Data Analysis. Springer, S. 223–236, 2013.

[Me10] Meinl, Thorsten: Maximum-score diversity selection. Dissertation, Universität Konstanz, 2010.

[Mi79] Milligan, Glenn W: Ultrametric hierarchical clustering algorithms. Psychometrika, 44(3):343–346, 1979.

[RK90] Rousseeuw, Peter J; Kaufman, L: Finding groups in data. Series in Probability&Mathematical Statistics, 34(1):111–112, 1990.

[Ro87] Rousseeuw, Peter J: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Journal of computational and applied mathematics, 20:53–65, 1987.

[SR62] Sokal, Robert R.; Rohlf, F. James: The Comparison of Dendrograms by Objective Methods. Taxon, 11(2):33–40, 1962.

Alexander Fillbrunn wurde am 28. Juni 1989 in Mannheim-Neckarau geboren. Nach seinem Abitur und einem einjährigen Freiwilligendienst in Malawi erhielt er seinen BSc in Angewandter Informatik an der Dualen Hochschule Baden-Württemberg in Mannheim. Während einiger Praktika, unter anderem am IBM Almaden Research Center in San Jose, USA und der IBM Forschungsabteilung in Böblingen entdeckte er seine Begeisterung für Data Mining und maschinelles Lernen. Seinen Masterabschluss machte Alexander Fillbrunn 2014 an der Universität Konstanz am Lehrstuhl für Bioinformatik und Information Mining, wo er anschließend für seine Doktorarbeit blieb. Heute arbeitet er als Solution Engineer bei der KNIME GmbH, einer Ausgründung der Universität Konstanz.