technische universität
dortmund

Bachelorarbeit

Zyklische Concept Drifts

Tobias Rickhoff
August 2016

Gutachter:
Prof. Dr. Katharina Morik
Dipl. -Inf. Hendrik Blom

Technische Universität Dortmund
Fakultät für Informatik
Lehrstuhl für Künstliche Intelligenz (LS8)
http://www-ai.cs.uni-dortmund.de

Inhaltsverzeichnis

Kapitel 1

Einleitung

Moderne industrielle Produktionsprozesse sind sehr komplex und hochgradig spezialisiert. Durch viele verschiedene Ursachen ändern sich die Gegebenheiten in diesen Prozessen kontinuierlich: Verfahren werden optimiert, zeitliche Abläufe geändert und ganze Subprozesse umstrukturiert. Rahmenbedingungen verändern sich: Personal wechselt, Komponenten verschleißen und werden erneuert. Das alles kann zu unterschiedlich starken, abrupten oder schleichenden Veränderungen in der Prozessdynamik führen.
Auch dieser hohen Komplexität ist es geschuldet, dass während der Produktion mitunter einige Informationen nicht bekannt sind, die Aufschluss über prozessbezogene Aspekte geben. Das kann zum Beispiel die Qualität des Produktes oder die Effizienz des Prozesses selbst sein. Oft wird eine Art von Vorhersagemodell angewandt, um die Informationen vorauszuberechnen, Solche Modelle liefern unter anderem die Thermodynamik und Physik. Einen weiteren Ansatz bietet die Datenanalyse [17, 20]: Produktionsdaten werden erfasst und von Algorithmen aus dem Bereich Maschinelles Lernen verarbeitet.

1.1 Maschinelles Lernen für Vorhersagen zu Produktionsprozessen

Die Anforderungen an Analyseprogramme sind hoch; Wenn der weitere Prozessablauf von einer Vorhersage abhängt, muss in Echtzeit ein Ergebnis vorliegen. Es bleibt keine Zeit, erst auf Anforderung Daten zu sammeln und zu analysieren, sondern das Lernen findet *online* und während des Prozesses statt. Außerdem muss die Vorhersage eine verlässliche Größe sein, insbesondere wenn wichtige Kontrollflüsse davon abhängen.
Je besser Vorhersageverfahren auf den Produktionsdaten funktionieren, desto geringer sind Energieaufwand und Ausschuss; Lässt sich beispielsweise ein fehlerhaftes Bauteil vor einem endgültigen Arbeitsschritt identifizieren, kann frühzeitig abgebrochen oder das Bauteil u.U. sogar durch zusätzliche Arbeitsschritte auf Weiterverarbeitungsqualität gebracht werden [17]. Bessere Lernverfahren bedeuten auch weniger Zeitaufwand für gewünschte Produkt-

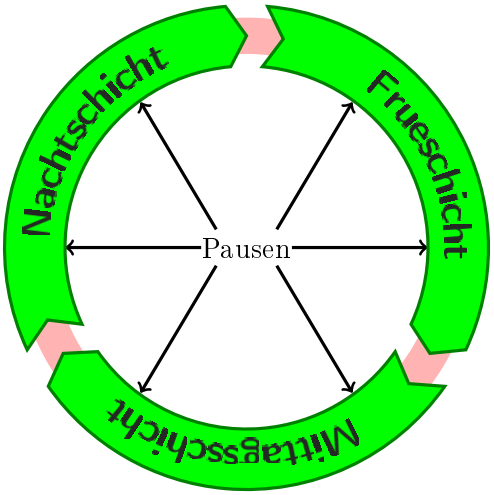

Abbildung 1.1: Beispiel eines Arbeitszyklus

qualität (durch weniger Nebenprodukte und Ausschuss) und, damit einhergehend, geringere Kosten.

Die Praxis zeigt, dass die Daten durch die o.g. Ursachen strukturellen Schwankungen unterliegen, sogenannten Concept Drifts [18, 21], und dass deshalb die Vorhersagequalität statischer, unveränderter Modelle abnimmt [15]. Dies gibt Ansporn, Online-Lernverfahren dahingehend zu verändern, dass die Modelle sich während der Anwendung den Daten anpassen. Einige Behandlungsansätze für Concept Drift sind bereits bekannt [18] und implementiert [15].

Durch immer wiederkehrende Abläufe (siehe Abb. 1.1) und starke Strukturierung von industriellen Produktionsprozessen[1] liegt die Vermutung nahe, dass die Änderung der Produktionsdaten einer Regelmäßigkeit folgt. Die Hypothese dieser Bachelorarbeit ist, dass sich ein Erkennungsmechanismus die wiederholende Eigenschaft der Daten zu Nutze machen und dadurch bessere Vorhersagemodelle oder Modellinstanzen einsetzen kann.
Dabei steht die Automatisierung im Vordergrund: Alle einsatz- und domänenspezifischen Einstellungen der Vorhersagesoftware werden *ohne* menschliches Eingreifen ermittelt und umgesetzt.

Am Beispiel einer modernen Werkzeugmaschine (Abb. 1.2) sollen einige der o.g. Punkte verdeutlicht werden. Eine solche Maschine wird zur Dreh- und/oder Fräsbearbeitung von komplexen Produkten verwendet. Massive Metallteile werden hier mit einer Genauigkeit von wenigen µm vollautomatisiert spangebend bearbeitet. Dafür stehen i.d.R. viele Werkzeuge zur Verfügung, die nacheinander oder sogar simultan Material abnehmen. Die

[1] Wartungs- und Zertifizierungspläne, festgelegte Einsatzdauer von Komponenten usw.

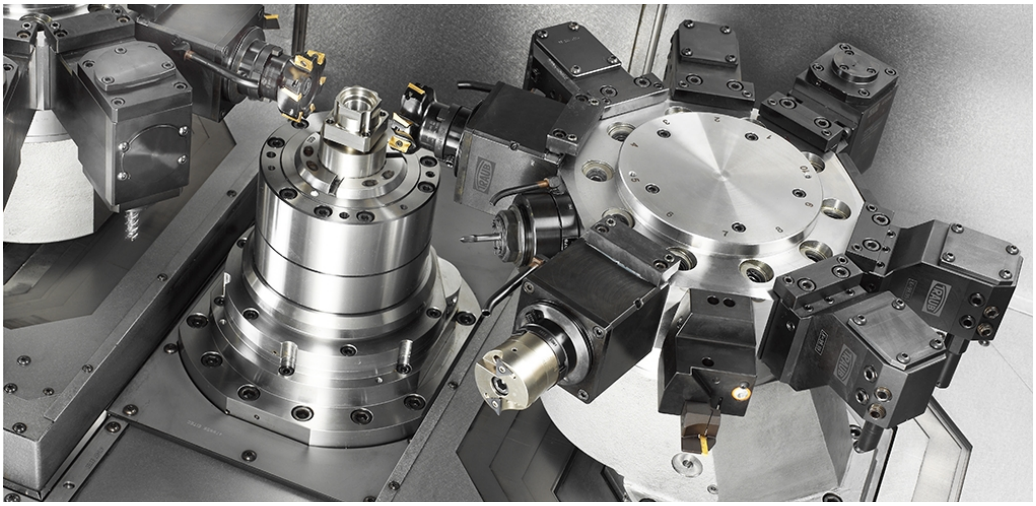

Abbildung 1.2: Mehrachsiges Bearbeitungszentrum mit einer Vielzahl von Werkzeugen[2].

Maschine in Abb. 1.2 verfügt über eine Hauptspindel, welche einerseits mit maximal 5000 min^{-1} Umdrehungen zum Drehen von Werkstücken benutzt wird, sich andererseits auf 0,001° genau positionieren lässt um Fräsarbeiten durchzuführen. Anforderungen in diesem Maßstab sind in der Industrie üblich. Für Fräs- und Dreharbeiten stehen bis zu vier Werkzeugrevolver[3] zur Verfügung, welche sich in drei Achsen unabhängig voneinander bewegen. Alle Komponenten der Maschine selbst (Motoren, Sensorik, Steuerung etc.), sowie jedes Werkzeug unterliegen individuellen Einflüssen durch Verschleiß, Temperatur und Art des Einsatzes. Die Verwendung und Wartung der Komponenten erfolgt prinzipbedingt periodisch: Für ein bestimmtes Produkt ist die Abfolge der Bearbeitungsschritte immer gleich.

Um den Produktionsprozess nach einer solchen Bearbeitung zu planen soll nun beispielhaft die Qualität einer, für die Funktion kritischen, Oberfläche des Werkstücks vorhergesagt werden. Ist diese Qualität nicht ausreichend, wird ein Zwischenschritt benötigt bevor das Produkt weiter verarbeitet werden kann. Bei ausreichender Qualität kann dieser Zwischenschritt übersprungen werden.
Ein *einzelnes* Modell, was diese Größe vorhersagt müsste durch die enorme Anzahl der genannten Einflüsse sehr viele Umstände berücksichtigen und flexibel für jedes Werkzeug anwendbar sein. Ein solches komplexes und flexibles Modell liefert im Einzelfall selten die Vorhersagegüte, die von einem auf diesen Fall spezialisierten Modell erreicht würde (Beispielsweise kann eine Modellinstanz auf ein bestimmtes Werkzeug „zugeschnitten“ sein.). Daher müssen mehrere Modelle oder Modellinstanzen verwendet werden.

[2]Bildquelle: INDEX-Werke GmbH & Co. KG, Esslingen unter: http://www.index-werke.de/de/traub/dreh-fraeszentren/tnx6542 [Stand: August 2016]
[3]drehbare Werkzeugbatterien von bis zu zehn Werkzeugen, wie in Abb. 1.2

Um einen automatisierten Erkennungsmechanismus zu erstellen, der bewertet, welche und wie viele Modelle oder Modellinstanzen wann zu verwenden sind, nähert sich die Arbeit der vorgestellten Problematik zunächst theoretisch.

1.2 Aufbau der Arbeit

Die Bachelorarbeit stellt Concept Drift vor und klärt Definitionen und Begrifflichkeiten. Es folgt eine Erklärung, wie Datenstromanalyse grundsätzlich funktioniert und wie Vorhersagen zu Daten getroffen werden können. Eine Übersicht der Problematik Concept Drift wird gegeben und gängige und verwendete Behandlungsmethoden dafür vorgestellt.
Anschließend erfolgt eine Erläuterung warum von Zyklen ausgegangen wird und wie sich welche Besonderheiten daraus ergeben. Dafür wird ein Zyklus definiert und es wird analysiert, wie und unter welchen Umständen Zyklizität nützlich sein kann.
Im Anschluss werden erste theoretische Ansätze vorgestellt darauf möglichst gut zu reagieren, indem die Vorhersagemodelle zu den richtigen Zeitpunkten gelernt und gewechselt werden. Zudem wird erläutert, wie sich die Auswahl der Trainingsmengen und die Einsatzzeitpunkte verschiedener Modellinstanzen auf die Qualität der Ergebnisse auswirken kann.
Als nächstes wird eine Praxis bezogene Systematik entwickelt und umgesetzt, die es erlaubt, große Mengen an Konfigurationen gegeneinander zu testen und die die Abfolge der Anwendung und des Trainings von Modellinstanzen an die Daten anpasst.
Durch genaue Dokumentation während des Ermittelns der optimalen zeitlichen Abstimmung für das Trainieren und Anwenden der Modellinstanzen kann nach ausgewählten Experimenten bewertet werden, ob und wie stark sich die Vorhersagequalität verbessern ließ. Die Ergebnisse werden dargestellt und bewertet.
Zum Abschluss werden die Erkenntnisse der Bachelorarbeit zusammengefasst, die Testsystematik kritisch bewertet und es wird einen Ausblick auf offene Fragen geben.

Kapitel 2

Concept Drift

Wie in der Einleitung dargelegt, unterliegen Produktionsprozesse vielen Veränderungen. Diese sind in manchen Fällen vorhersehbar: beispielsweise nimmt die Jahres- und Tageszeiten bedingte Umgebungstemperatur Einfluss auf Produktionsmaschinen und Werkstoffe (Klebstoff, Chemikalien, etc.), der Austausch eines Schneidwerkzeugs an einer Werkzeugmaschine reduziert Vibrationen, durch Anpassungen der Peripherie (Förderbänder etc.) ändert sich der Produktionsablauf, um auf Verschleiß der Maschinen und Sensorik zu reagieren wird die Bedienung der Produktionsmaschinen den Gegebenheiten angepasst.
In anderen Fällen kann keine Aussage über das Auftreten von Einflussfaktoren gemacht werden: Beispielsweise durch unregelmäßig geöffnete Wartungsluken strömt Luft in eine Anlage, Defekte bringen die Produktion vorübergehend zum erliegen (Komponenten akklimatisieren sich, Maschinen müssen wieder angefahren werden), durch unerwartet reibungslosen Dauerbetrieb verhalten sich Komponenten plötzlich anders.
Ein Produktionsprozess ist ein hochdynamischer Vorgang, bei dem komplizierte Kausalitäten das Prozessergebnis bestimmen.

2.1 Begriffe und Definitionen

Jede Art von Ablauf wird durch seine Umwelt beeinflusst; damit ist jede Beobachtung von einer erzeugenden Umgebung abhängig. Wie diese Umgebung, die Messungen von Daten und deren Verarbeitung formal beschrieben werden kann wird in diesem Kapitel festgelegt und erläutert.

2.1.1 Source

Die Umwelt ist enorm facettenreich und unmöglich zu erfassen oder darzustellen. Dennoch bestimmt die Umgebung (eines Prozesses), bestehend aus allen nötigen Akteuren, Sensoren und anderen Entitäten das Produkt und die Produktionsdaten. Diese nicht messbare Umwelt wird in der Literatur *Source* genannt[15, 23] (mitunter auch *hidden Context*[18, 19]),

da sie als Quelle der Produktionsdaten die Wahrscheinlichkeiten festlegt, mit der Produkte eine bestimmte Ausprägung y aus einer Menge aller möglichen Ausprägungen Y besitzen (beispielsweise ist es in manchen Anwendungen gefordert Daten zu klassifizieren). Die Source wird als eine Wahrscheinlichkeitsverteilung interpretiert und behandelt. Sie *erzeugt* Daten X einer gewissen Verteilung.

$$S := \big\{\{p(X), p(X|y)\}|y \in Y\big\}$$

Wir definieren S_t als Source des Datums X_t [23] (siehe Abb. 2.1).

2.1.2 Data Item

Ein Datum wird umgesetzt durch ein Data Item X_t, eine Menge von Schlüssel-Werte-Paaren, die über einen Zeitstempel t identifiziert werden kann. Das Data Item enthält Attribute, in denen die Produktionsdaten abgelegt sind. Daten werden von der Sensorik des Prozesses erfasst und in einem Data Item gesammelt. Zusätzlich können Metadaten an ein Data Item angehängt werden. Diese „Annotations“ können Informationen zur Prozesskontrolle oder zum Produkt selbst sein[1].

2.1.3 Stream

Ein Datenstrom, d.h. eine Folge von Data Items $(X_t)_{t\in[0...n]}$, wird *Stream* genannt. Die Daten werden *nacheinander* verarbeitet und es ist nicht festgelegt, wann (oder ob) der Datenstrom endet. Natürlich kann ein endlicher Stream auch aus einer Datei erstellt werden, indem diese Schrittweise dem Bearbeitungsprogramm, in diesem Fall einem Lernalgorithmus, vorgelegt wird.

2.1.4 Concept

Ein Concept C_A definieren wir als eine Teilmenge des Streams mit Data Items, welche die gleiche erzeugende Source S_A haben $C_A = \{X_t|S_t = S_A\}$. Anschaulich ist die Source die (latente) Ursache, das Concept die (beobachtbare) Wirkung. Ein Concept kann als eine Gruppe g von l Daten verstanden werden, die unter gleichen Umständen generiert wurden. Die Source der Daten lässt sich nicht modellieren, die Verteilung hingegen schon: $\vec{p}$ beschreibt den Zusammenhang zwischen Merkmalen X und Ausprägung $y \in Y$ eines Datums[2]. Damit ist ein Concept

$$g(\vec{p}, l). \tag{2.1}$$

[1] Ist ein Data Item einer Entität fest zugewiesen, zum Beispiel einem produzierten Bauteil, kann möglicherweise eine Klassifizierung in Gruppen (Label) erfolgen (z.B. gute-, mittlere-, und schlechte Qualität) oder ein unbekannter Wert vorhergesagt werden (Regression). Diese Information kann dann als Annotation dem Data Item angefügt werden und zur späteren Evaluation beitragen.

[2] Beispielsweise kann $\vec{p}$ als Parameter eines abstrakten Ablaufs gesehen werden, welcher zu Merkmalen eines Datums die Ausprägung festlegt.

Üblicherweise treten in einem Stream mehrere Concepts auf, also verschiedene Verteilungen, die zur Verarbeitung vorliegen.

2.2 Verarbeitung von Daten

Daten als Stream zu verarbeiten bietet den großen Vorteil, dass zu keinem Zeitpunkt alle Daten gleichzeitig vorliegen müssen; So können immense Datenmengen Resourcen schonend analysiert werden. Ein Data Item wird erfasst, verarbeitet und nicht gespeichert. Weitergearbeitet wird nur mit einem durch das Datum veränderten Zustand des verarbeitenden Programms. Dies wird Online-Verarbeitung genannt, weil nur das aktuell vorliegende Data Item betrachtet wird und nicht erheblich ist, wie viele Daten schon gelesen wurden oder noch folgen [2, 7, 13]. Die Daten zu speichern ist entweder nicht möglich[3] oder nicht erwünscht, weil für ein Datum ohnehin nur eine einmalige Bearbeitung vorgesehen ist.
Werden die Daten gespeichert oder einer permanent vorhandenen Datei entnommen, so heißt das Verfahren Offline-Lernen. Lernt ein Algorithmus auf einer Menge von Daten gleichzeitig, nennt sich dieses Verfahren Batch-Lernen[7]. Jedes dieser Verfahren wird in der Industrie verwendet, doch Online-Algorithmen sind im Hinblick auf Echtzeitanforderungen unumgänglich.

2.3 Modelle zur Vorhersage

In der Industrie werden zur Vorhersage von Informationen (Online wie auch Offline) statistische Modelle verwendet. Instanzen dieser Modelle werden vorab mit Daten trainiert, zu denen die Informationen bereits bekannt sind, die normalerweise im Betrieb vorhergesagt werden sollen. Nach Abschluss des Trainings trifft die Modellinstanz Vorhersagen zu neuen Data Items, indem es das gesammelte Wissen über die Wahrscheinlichkeiten zu den Trainingsdaten anwendet. Ein gängiges Vorgehen, ist es Instanzen erst *offline* mit gespeicherten Daten zu trainieren und dann *online* für einen Datenstrom anzuwenden.

Am Beispiel des naiven Bayes-Klassifikators lässt sich die Vorgehensweise eines solchen Modells gut beschreiben: Das Modell speichert während des Trainings die Merkmale X und die Ausprägung $y \in Y$ der Daten. Aus diesen Informationen lässt sich die Wahrscheinlichkeit für eine Ausprägung zu einem Merkmal $p(y|X) = \frac{p(X|y)p(y)}{p(X)}$ errechnen. Die Vorhersage für ein Datum X ist nun dasjenige y mit der größten Wahrscheinlichkeit $p(y|X)$, eine anschauliche Erläuterung dieses Beispiels findet sich in [22].
Für gute Vorhersagen ist es essentiell, dass sich Trainingsdaten und Testdaten ähneln, oder

[3] Datenmengen von Radioteleskopen beispielsweise sind mitunter zu groß um sie (ökonomisch vertretbar) speichern zu können [6].

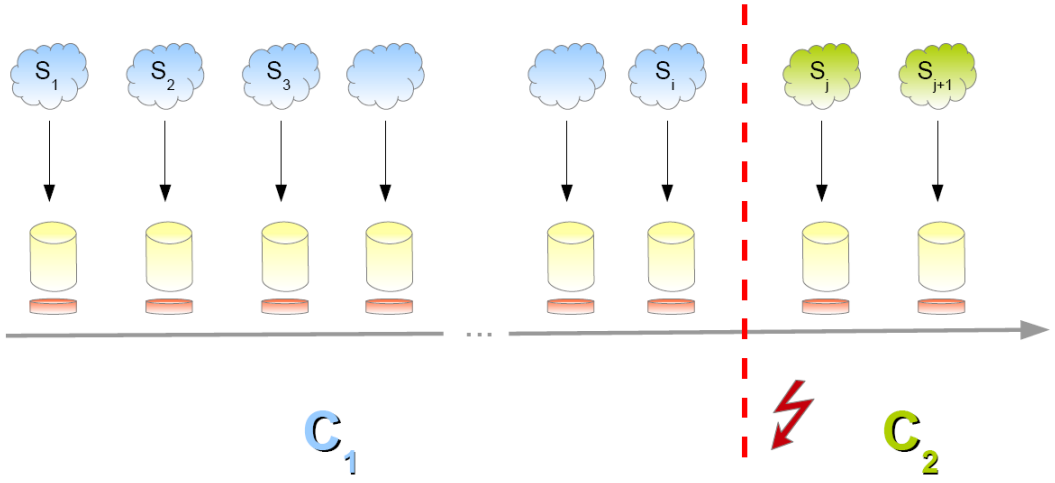

Abbildung 2.1: Concept Drift [15]

einer ähnlichen Verteilung zu Grunde liegen (siehe 4.3): Die beiden Datenmengen sollten aus dem gleichen Concept stammen.

2.4 Concept Drift

Die Produktionsumgebung vieler industrieller Prozesse unterliegt wie eingangs erläutert Veränderungen [15]. Die Source ändert sich *latent*, also ungreifbar[4], sichtbar sind die Auswirkungen: Durch verschiedene Sources werden Daten unterschiedlicher Concepts aufgezeichnet, sodass eine Abfolge f einer Conceptmenge $\hat{C}$ entsteht.

$$f(\hat{C}) = C_1, C_2, C_3, \ldots \tag{2.2}$$

Der Wechsel zwischen Concepts kann auf drei unterschiedliche Arten erfolgen (siehe auch Abb. 2.2):

- **abrupt:** Von einem Datum des Concepts C_A auf das nächste Datum wechselt das Concept zu C_B. Da es sich bei Concepts um sehr schwer zu fassende Größen handelt, wird diese Driftart erst zeitverzögert erkannt. In einigen Quellen wird hier nicht von Concept Drift gesprochen, sondern von Concept- (oder Level-) Change, da es sich weniger um eine Driftphase als um einen harten Wechsel handelt.

- **graduell:** Während Items des einen Concepts C_A zunehmend seltener auftreten, häufen sich Data Items des Concepts C_B. Bei einem rein graduellen Drift gehören die Items fest zu ihrem jeweiligen Concept. Anders als beim abrupten Drift dauert dieser Wechsel einige Data Items an.

[4] vgl. „hidden context“ in [18]

- **inkrementell:** Die Source des einen Concepts C_A geht langsam in die von C_B über. D.h. die Verteilung die Daten von C_A hervorgebracht hat ändert sich langsam zu einer, die Daten des Concepts C_B produziert. Zur Zeit des Drifts lässt sich nicht feststellen, zu welchem Concept ein Datum gehört, weil Data Items eines sich kontinuierlich verändernden Misch-Concept gemessen werden.

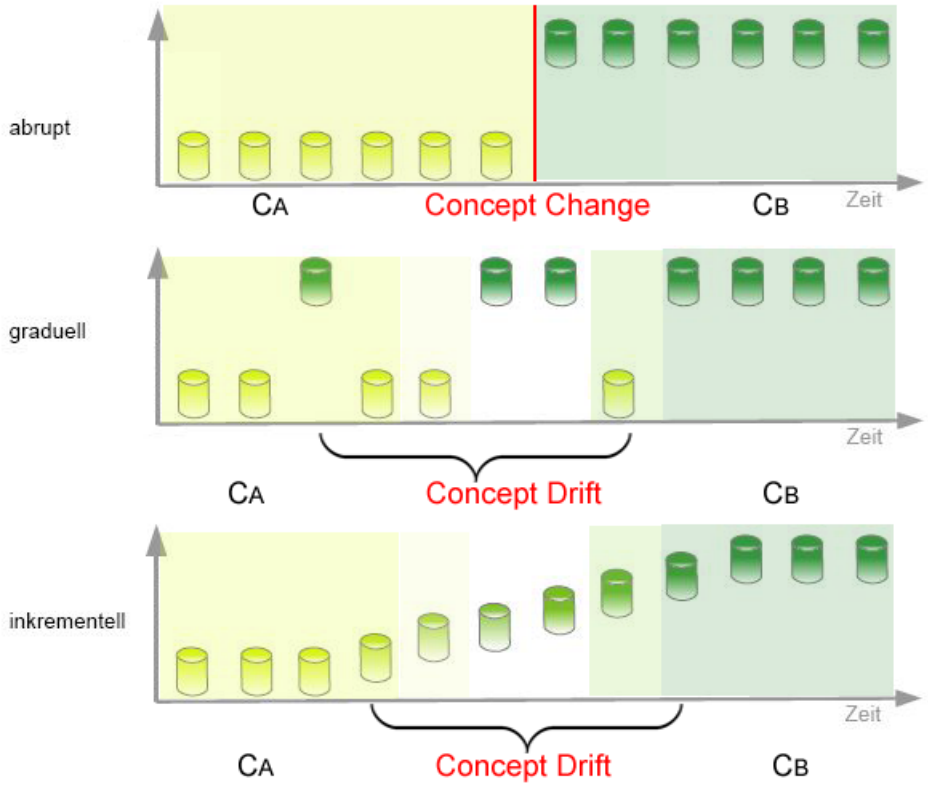

Abbildung 2.2: Driftarten nach [15] und [23]

Es ist durchaus denkbar, dass diese Driftarten in der Praxis weder klar voneinander abgegrenzt noch gleichmäßig auftreten. So kann ein Datenstrom z.B. erst inkrementell auf ein Concept C_A zu driften und noch während des Übergangs C_A zu C_B Items enthalten, die ganz dem Ziel-Concepts C_B angehören.

2.5 Behandlungsmethoden für Concept Drift

Solche Concept Drifts sind also *strukturelle* Änderungen in den Daten und verschlechtern somit die Vorhersagequalität der Modellinstanzen, da diese mit Daten eines veralteten Concepts trainiert wurden.
Es existieren bereits einige Ansätze, auf Concept Drift zu reagieren um die Qualität der Vorhersagen nach einem Drift zu verbessern. Dabei unterscheiden sie sich in Komplexität

und Aufwand enorm.

Die einfachste Methode ist regelmäßig ein neues Modell zu lernen. Dieser **naive Ansatz** ignoriert jeden Drift und geht nur davon aus, dass sich die Daten ändern. Das funktioniert dann sehr gut, wenn der Beginn des Neu-Lernens mit dem Ende der Driftphase (siehe Abb. 2.2) zusammen fällt. Das ist üblicherweise sehr selten und es erfordert großes Geschick für einen Stream einen geeigneten Abstand zu wählen. Da hier keine Intelligenz für eine Drifterkennung oder mehrere Modelle notwendig sind, ist die naive Driftbehandlung relativ resourcensparend, solange nicht in unnötig kurzen Abständen neu gelernt wird. Sind die Abstände zu groß, wird zu lange eine veraltete Modellinstanz angewendet und die Vorhersagequalität sinkt [15].
Driftdetektoren sind Verfahren, die auf Basis der Verteilung der Daten Rückschlüsse ziehen, ob ein Drift stattfindet, bevorsteht oder ausbleibt. Steht ein Drift kurz bevor, wird eine Warnung ausgegeben, sodass sich Lerner oder andere beteiligte Programme auf einen kommenden Drift einstellen können. Wird ein Drift erkannt gibt der Detektor eine Bestätigung aus und setzt die Warnung zurück [3, 9] (siehe Abb. 2.3).
Eine ganze Gruppe an Behandlern bilden die **Fensterverfahren** [15, 12]. Hier werden auf einer Teilmenge der zuletzt gelesenen Items statistische Verfahren angewendet, die erkennen sollen, ob sich die Verteilung der vorherzusagenden Informationen (z.B. *Labels*) oder die der Items zu diesen Informationen (sprich die Source) ändert. Einige Implementierungen benutzen adaptive Fenstergrößen: Um nicht Daten der Driftphase (Abb. 2.2) zu benutzen, trainiert die Instanz kurz nach einer geänderten Verteilung auf einem kleineren Fenster. Das Fenster wird vergrößert, wenn sich die Verteilungen nicht mehr ändern [12]. Es gibt unterschiedliche Strategien nach denen die Fenstergröße angepasst wird, sie wirken sich stark auf die Performanz der Driftbehandlung aus. Auf diese Strategien soll hier nicht näher eingegangen werden. Der Aufwand besteht beim Fensterverfahren darin, das Fenster zu verwalten und dessen Größe zu berechnen.
Ensemble-Lernen beschreibt die Methodik, viele verschiedene Lerner zu halten und zu trainieren. Entweder werden Instanzen des gleichen Modells auf unterschiedlichen Datenmengen trainiert [13] oder verschiedene Modelltypen, die beispielsweise auf bestimmte Concepte spezialisiert sind. Alle trainierten Lerner treffen nun Vorhersagen zu einem Datum. Die Entscheidung, wessen Vorhersage verwendet wird kann auch hier nach mannigfaltigen Strategien getroffen werden: Beispielsweise durch einen Mehrheitsentscheid zwischen den Lernern, oder es kann für jeden Lerner die Performanz geschätzt werden, dann wird die Vorhersage desjenigen Lerners mit der voraussichtlich besten Genauigkeit stärker gewichtet, als die der anderen. Die Möglichkeiten Ensembles zu konfigurieren um eine gute Qualität zu erreichen sind vielseitig und komplex [13]. Kombinationen dieser Methoden sind üblich: Fenster- und Ensembleverfahren verwenden meist Driftdetektoren.

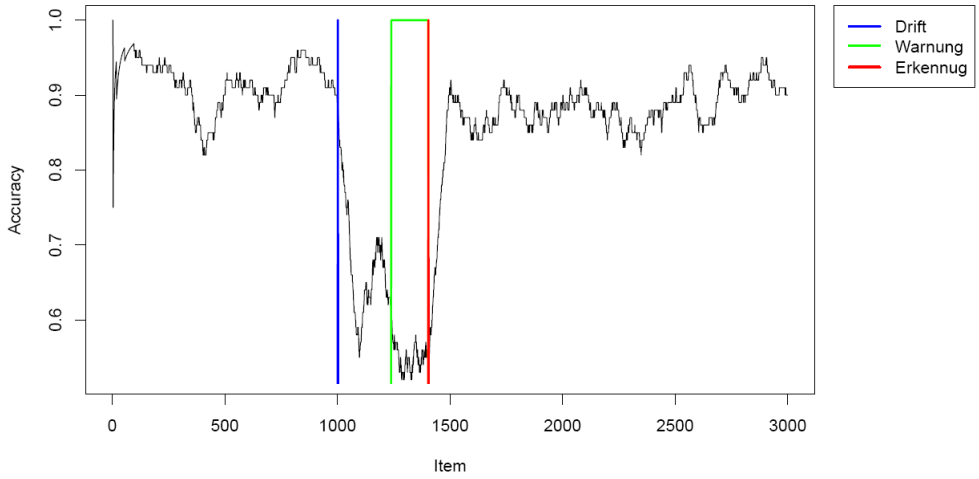

Abbildung 2.3: Level des Driftdetektors DDM(Drift Detection Method) [15]

2.5.1 Metriken für Drifterkenner

Um Driftdetektoren bewerten zu können und zu entscheiden, welcher Detektor für einen bestimmten Zweck am besten geeignet ist, wurden in [3, 15] eine Reihe von Metriken vorgestellt. Diese Maßstäbe für Driftdetektoren geben über jeweils andere Aspekte der Nützlichkeit eines Detektors Auskunft, beispielsweise über die Verlässlichkeit einer Warnung, Zeit die bis zur Drifterkennung vergeht[5] oder die Anzahl der Fehldetektionen. Hier werden nun die wichtigsten vorgestellt:

- **Mean Time between False Alarms (MTFA):** Wenn ein Drift erkannt wurde, obwohl keiner stattgefunden hat, wirkt sich das schlecht auf die Effizienz und Performanz des Lernalgorithmus aus. Auch bei einem falschen Alarm i (zum Zeitpunkt t_{F_i}) wird neu gelernt, das kostet Resourcen und bedeutet u.U. Verwaltungsaufwand für die Software. Die Menge der Fehlalarme I erzeugt Mehraufwand, daher sollte die Zeit zwischen diesen möglichst groß sein.

  $$\text{MTFA} = \frac{1}{I}\sum_{i=1}^{I}(t_{F_i} - t_{F_{i-1}})$$

  Die **False Alarm Rate (FAR)** lässt sich direkt als 1/MTFA ableiten [3].

- Die **Mean Time to Detection (MTD)** beschreibt die Zeit zwischen Drift c_i und Erkennung, dabei muss die Erkennung in der Umgebung des Drifts liegen ($D_{u(c_i)}$).

  $$\text{MTD} = \frac{1}{C}\sum_{i=1}^{C}(t_{D_i} - t_{c_i})$$

[5] In diesem Zusammenhang wird die Zeit in Data Items gemessen. Das Lesen eines Items entspricht einer Zeiteinheit.

- Die **Missed Detection Rate (MDR)**, d.h. die Anzahl der Drifts in deren Umgebung keine Erkennung stattfindet ist eine für diese Arbeit enorm wichtige Größe; bleibt ein Drift unerkannt, kann nicht auf ihn reagiert werden.

$$\text{MDR} = |\{c_t | \nexists D_{u(t)}\}|$$

- **Average Number of Items to Learn (ANIL):** Beschreibt die durchschnittliche Anzahl der Items zwischen Warnung W_i und Erkennung D_i.

$$\text{ANIL} = \frac{1}{I} \sum_{i=0}^{I} |\{t | \in [W_i, \ldots, D_i]\}|$$

  Weil diese Items i.d.R. (auch in dieser Arbeit) zum trainieren des nächsten Modells verwendet werden, darf die Anzahl weder zu klein (Modell wird nicht gut) noch zu groß sein (Resourcenverschwendung, Overfitting ([11]), möglicherweise werden Daten eines veralteten Concepts verwendet).

- Mit der **Detection Warning Ratio (DWR)** lassen sich Rückschlüsse ziehen, wie „empfindlich“ der Detektor konfiguriert ist. Diese Größe misst die Anzahl der Alarme W pro Drifterkennung D.

$$\text{DWR} = \frac{W}{D}$$

Concept Drift ist deshalb eine Herausforderung, weil nicht bekannt ist, welches Concept als nächstes folgt. Es lässt sich nicht entscheiden, welche Modellinstanz nach einem zukünftigen Drift angewandt werden oder ob eine neue Instanz trainiert werden muss. Das liegt in der Unvorhersehbarkeit der Conceptabfolge begründet: Der Abfolge fehlt vorhersehbare Struktur.

Kapitel 3

Zyklische Sensordaten

Im Allgemeinen ist Struktur nützlich für viele Anwendungen, da sich aus der Struktur selbst Schlüsse über die Anwendung ziehen lassen; Beispielsweise können Menschen sich in ihrer komplexen Umwelt nur so effizient zurechtfinden, weil sie sich der Struktur ihrer Umgebung bedienen um Informationen daraus zu gewinnen. Unwichtige Dinge werden erkannt und ausgeblendet, andersherum kann ein bekanntes Muster, welches verändert auftritt, sehr schnell bewertet werden[1]. Je mehr Meta-Informationen wir zur Verfügung haben, desto besser und schneller finden wir uns zurecht. Ein Beispiel, dass auch Algorithmen von solchen Informationen profitieren können liefert [14].
Vieles im (Arbeits-) Leben muss effizient und schnell funktionieren, deshalb ist es streng organisiert und folgt einem gut strukturierten Ablauf.

3.1 Ursachen für Zyklen

In Produktionsprozessen findet vieles nicht nur strukturiert, sondern darüber hinaus zyklisch statt. Im kleinen Maßstab wirkt sich möglicherweise der Tag-/Nacht-Zyklus auf die Produktionsmenge/ -geschwindigkeit aus: Oft stehen die Maschinen zu Pausenzeiten still, auch diese Zeiten hängen von der Tageszeit ab und treten daher zyklisch auf. Arbeiten die Maschinen hingegen durchgängig, erfolgt der Schichtwechsel des Bedienpersonals zyklisch. Unabhängig vom Tagesrythmus gibt es strikte Wartungspläne, Zertifizierungen und Vorgaben zum Austausch von Komponenten[2]. Auch zeitlich variable Zyklen treten auf: Werkzeuge müssen aussortiert werden (erst, wenn sie defekt sind), Leitungen erneuert werden (wenn sie zugesetzt sind).
Um solche Strukturen einzugrenzen muss zunächst festgelegt werden, wie Streams beschaffen sind um als zyklisch zu gelten.

[1] Ein Beispiel einer solchen Strukturierung ist die StVO: Wird die Struktur erfüllt, ist ein Verkehrsteilnehmer nicht überrascht(, wenn das blinkende Fahrzeug kurz darauf abbiegt). Bricht etwas die Struktur, erregt es sofort Aufmerksamkeit (Fremdkörper auf der Fahrbahn etc.).

[2] Beispielsweise haben Hydraulikschläuche eine feste Lebensdauer, die nicht überschritten werden darf.

Abbildung 3.1: Einige mögliche Conceptverteilungen im Arbeitsrythmus Abb. 1.1

3.2 Definition: Zyklus

Ein Zyklus z_i ist eine geordnete Abfolgen von Concepts, die sich direkt im Anschluss mindestens einmal in gleicher Form wiederholt. Abfolgen z von Concepts werden mit einem Distanzmaß $\Delta_Z \in [0,1]$ verglichen, beispielsweise mit dem „*strikten*“ Distanzmaß.

$$\Delta_Z(z_m, z_n) = \begin{cases} 0, & \text{falls } (|z_m| = |z_n|) \bigwedge (\forall j \; C_{m,j} = C_{n,j}) \\ 1, & \text{sonst} \end{cases} \tag{3.1}$$

Damit sind die Abfolgen z_m und z_n gleich, g.d.w. sie die gleiche Anzahl Concepts haben und jedes Concepts aus der Abfolge z_m an der gleichen Position j in $z_n$auftritt. D.h. die Sequenz z_i ist ein Zyklus, g.d.w. $\Delta(z_i, z_{i+1}) < \epsilon$. [3]
Die Folge aller I gleichen Conceptabfolgen wird im Folgenden Epoche genannt, dabei ist I die Anzahl, der Iterationen der Zyklen.

$$E := (z_i)_{i=1,\ldots,I+1}$$

3.3 Mögliche Szenarien

Das ideale zyklische Szenario (Abb. 3.1) sieht so aus, dass eine Menge z von Concepts C_j sich in gleicher Weise immer wiederholt; dabei bleibt jedes Concept über alle Zyklen gleich und gleich lang. Die Zyklen einer Epoche gleichen sich in diesem Szenario auch strukturell: die Concepts treten in immer gleicher Reihenfolge auf, die Menge z der Concepts bleibt über alle Perioden gleich, d.h. nach dem ersten Zyklus treten keine neuen (nie gesehenen) Concepts mehr auf und es treten immer alle bisher gelernten Concepts in jedem neuen Zyklus auf.
Es ist möglich und sogar wahrscheinlich, dass dieser Idealfall nicht immer eintritt und die

[3] Die Längen der Concepts müssen ausdrücklich *nicht* übereinstimmen, auch weitere Lockerungen der hier vorgestellten Distanz sind möglicherweise sinnvoll: z.B. kann es hilfreich sein kleine Ungleichheiten in der Reihenfolge zuzulassen, um bei sich leicht ändernden Zyklen nicht neu lernen zu müssen (siehe 3.3 und 4.4).

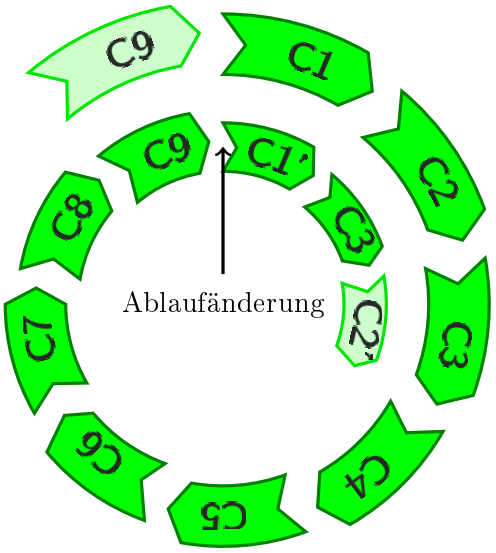

Abbildung 3.2: Concept C1 und C2 treten leicht abgewandelt im nächsten Zyklus auf, die Abfolge von C2 und C3 ist verändert.

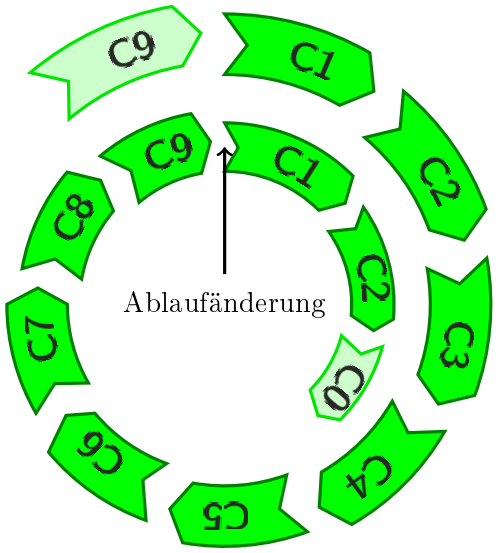

Abbildung 3.3: Concept C1 und C2 treten mit veränderter Länge im nächsten Zyklus auf, ein neues Concept C0 ist vorhanden.

Zyklen und/oder Concepts sich unterscheiden (siehe Abb. 3.2 und 3.3): Durch geänderte Abläufe in der Produktion kann sich ein Concept C_j über mehrere Zyklen hinweg leicht ändern (C'_j) ohne, dass es sinnvoll wäre C_j und C'_j vom Lerner unterscheiden zu lassen, da sich die beiden nur um wenige Nuancen unterscheiden. Die Veränderung kann sowohl die Verteilung $\vec{p}$ im Concept betreffen, als auch dessen Länge$l$ (siehe 2.1.4).
Das kann verschiedenste Ursachen haben: Produktionsprozesse verändern sich stetig, der Produzent ist ständig daran interessiert, die Qualität, die Effizienz oder das Produktionsvolumen zu verbessern. In diesem Zusammenhang werden Prozessschritte verändert, zusammengelegt oder neu angeordnet.
Auch kann es passieren, dass solche Optimierungen zu veränderter Concept-Reihenfolge, oder sogar zu leicht geänderten Concepts führen. Beispielsweise wurde eine Werkzeugmaschine bisher gewartet, kalibriert, eingesetzt, gereinigt, kalibriert und wieder eingesetzt, jetzt wurde die Reinigung so verändert, dass der Kalibrierungsschritt danach wegfällt. Dafür wird jetzt gereinigt, gewartet, kalibriert, eingesetzt, gereinigt, eingesetzt usw.
Nur Idealtypische Zyklen zu berücksichtigen ist in der Realität zu unflexibel und daher wenig sinnvoll (siehe 4.4).

3.4 Realität und Anwendbarkeit

Echte Produktionsdaten unterliegen gleichzeitig vielen verschiedenen zyklischen Einflüssen, die sich überschneiden, verschiedene Frequenzen und Amplituden haben, die nicht die gleiche Auswirkung auf die gemessenen Daten haben. So wird sich möglicherweise eine zyklische Änderung der Jahreszeiten bedingten Raumtemperatur weniger in den Daten widerspiegeln, als die montägliche Anlaufphase aller Maschinen.
Es ist denkbar, dass sich zyklische Einflüsse gegenseitig verstärken oder aufheben wie in Abb. 3.8 dargestellt. Ein Beispiel:
An einer unter Überdruck arbeitenden Produktionsmaschine sei regelmäßig der Drucksensor defekt (misst immer zu niedrigen Druck): Abb. 3.4 und zusätzlich mit einer anderen Regelmäßigkeit ein Ventil (schlägt Leck): Abb. 3.5. Fallen *zufällig* beide Defekte zusammen, wird der für den Druck verantwortliche Kompressor angesteuert einen stärkeren Betriebsdruck zu schaffen. Auf diese Weise kompensiert der defekte Sensor *zufällig* das defekte Ventil: Abb. 3.6.
Der umgekehrte Fall tritt ein, wenn ein defekter Drucksensor immer zu hohen Druck misst und gleichzeitig ein Ventil Leck schlägt: Der Kompressor wird zu früh abgeschaltet und zusätzlich entweicht Luft aus dem defekten Ventil. So verstärken sich *zufällig* beide Defekte: Abb. 3.7.

In welchem Umfang solche Effekte real relevant sind oder Concepts sich in der Realität tatsächlich so verhalten lässt sich an dieser Stelle nicht beurteilen. Es könnte sein, dass die Vielzahl der periodischen Einflüsse ein Feststellen von Zyklen unmöglich machen. Es ist allerdings genauso gut vorstellbar, dass einige wenige Faktoren durch eine große Differenz in ihrer Einflussnahme auf die Daten ganz klar Zyklen entstehen lassen (siehe Kapitel 7).

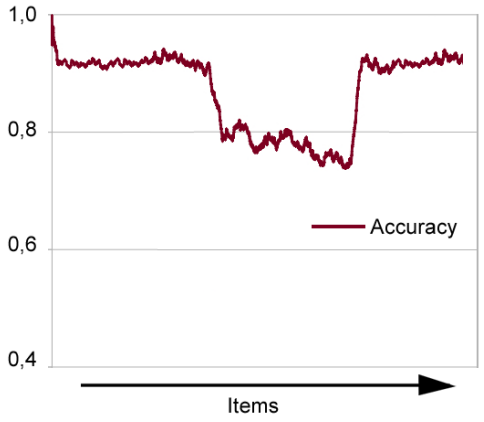

Abbildung 3.4: Concept Drift 1

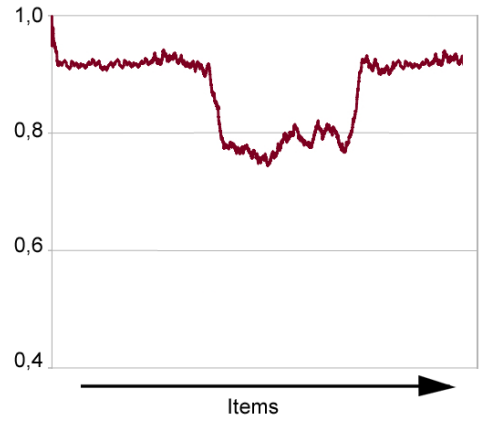

Abbildung 3.5: Concept Drift 2

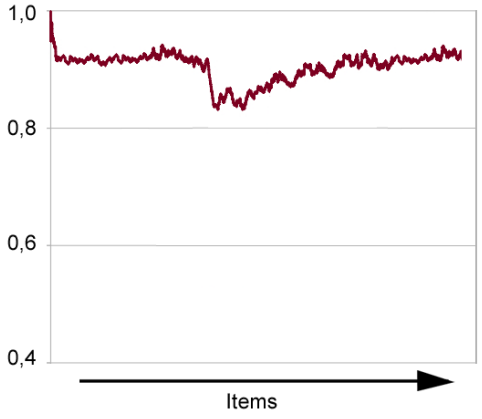

Abbildung 3.6: Concept Drift 1 und 2 heben sich gegenseitig auf.

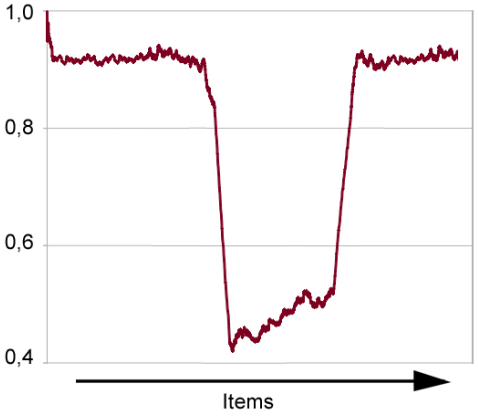

Abbildung 3.7: Concept Drifts 1 und 2 verstärken sich gegenseitig.

Abbildung 3.8: Verschiedene Effekte mehrerer sich überschneidend driftender Faktoren

Kapitel 4

Modellverwaltung

Um zyklische Strukturen der Conceptabfolgen auszunutzen sind eine Reihe von Schritten notwendig: wiederkehrende Strukturen müssen erkannt, Wissen über diese Strukturen angewandt und die Wissensbasis erweitert werden. Um dies zu gewährleisten wird eine Modellverwaltung eingeführt, welche die genannten Anforderungen umsetzt:

4.1 Concepts erkennen

Offensichtlich ist es nötig, um Conceptzyklen erkennen zu können, zunächst Concepts zu identifizieren. Das ist insofern schwierig, weil bei realen Daten nicht direkt erkennbar ist, um welches Concept es sich handelt, da ein Concept von der (latenten) Source abhängt. Durch Drifterkennung (siehe 2.5) ist es bereits möglich festzustellen, dass ein Drift stattgefunden hat. Dadurch lässt sich der Datenstrom in Concepts sequenzieren. Um Gleichheit auf Concepts zu definieren muss jedem Concept eine messbare Größe zugewiesen werden. Der hier gewählte Ansatz ist, zwei Concepts C_A, C_B anhand der Performanzen der für sie trainierten Modellinstanzen zu vergleichen. Beispielsweise kann dafür eine Verlustfunktion [8] verwendet werden: $L(Y, \hat{f}_A(X)) = \big(Y - \hat{f}_A(X)\big)^2$. [1] Nicht nur die Verlustfunktion kann Auskunft über die Performanz einer Modellinstanz geben, jedes Qualitätsmaß ist dafür (in unterschiedlichem Umfang) geeignet. Hier sei die Distanz δ zweier Concepts definiert als Differenz ihrer Performanzen. Sie sind gleich[2], wenn gilt:

$$\delta\big(\hat{f}_A(X), \hat{f}_B(X)\big) < \lambda, \text{[3]} \tag{4.1}$$

Diesem Vorgehen liegt die Annahme zugrunde, „dass ein trainierter Lerner auf einem einzelnen Concept, eine in etwa konstante und verhältnismäßig geringe Fehlklassifkationsrate

[1] $\hat{f}_A$ ist die Instanz des Modells, welche sich für C_A als beste gezeigt hat (i.d.R. weil sie dafür trainiert wurde, siehe 4.3).

[2] Diese Gleichheit ist eine Ähnlichkeit, sie ist nicht transitiv. (siehe auch 4.4)

[3] z.B. $\delta(\hat{f}_A(X), \hat{f}_B(X)) = |(L(Y, \hat{f}_A(X)) - L(Y, \hat{f}_B(X))|$

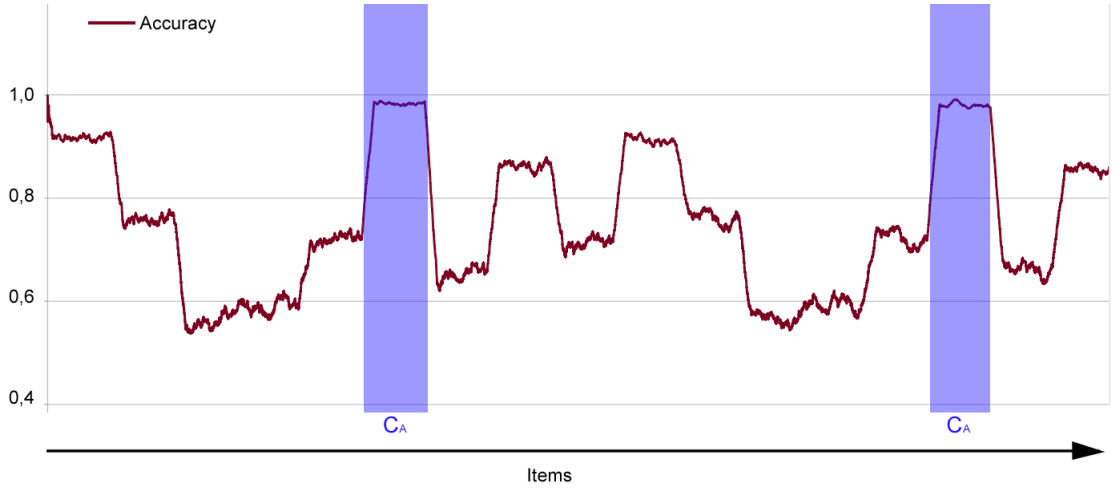

Abbildung 4.1: Performanz eines auf C_A spezialisierten Modells auf Daten mit Concept Drift.

aufweist. Auf einem anderen, ebenfalls konstanten Concept wird der Lerner auch eine ungefähr konstante, jedoch geringere Klassifkationsgenauigkeit aufweisen."([15], S.18) (Abb. 4.1)

Die Performanz (oder wie hier im Speziellen die Vorhersagegenauigkeit) ist die Abbildung der Qualität einer Modellinstanz auf einen Wert, diesen Wert *aussagekräftig* und *aktuell* zu bestimmen ist nicht trivial.

4.1.1 Performanz der Modellinstanzen berechnen

In der Literatur werden einige Verfahren vorgestellt, die Qualität eines Modells zu messen [16], hier soll die Schwierigkeit der Messung einmal am Beispiel der Genauigkeit über eine Teilmenge W der zuletzt verarbeiteten Daten

$$\text{Accuracy}_W(\hat{f}) = \frac{|\text{von } \hat{f} \text{ korrekt klassifizierte Items in } W|}{|W|}$$

verdeutlicht werden. Das Genauigkeitsfenster hat enormen Einfluss auf die Conceptidentifizierung, denn es bestimmt, welche Data Items in die Performanz eingerechnet und insbesondere welche nicht eingerechnet werden.

Um dies zu verdeutlichen folgt hier ein Beispiel: Sei ein inkrementeller Drift von einem Concept C_A zu C_B (Abb. 4.2), nach oder während dem Drift wird die Drifterkennung allein abhängig von der Daten- und Merkmalverteilung [3] eine Warnung auslösen und etwas später den Drift erkennen. Zum Zeitpunkt der Drifterkennung ist es nun von hoher Wichtigkeit eine belastbare Aussage über die Qualität jeder Modellinstanz für das *aktuelle* Concept C_B zu machen[4].

[4] wg. 4.2

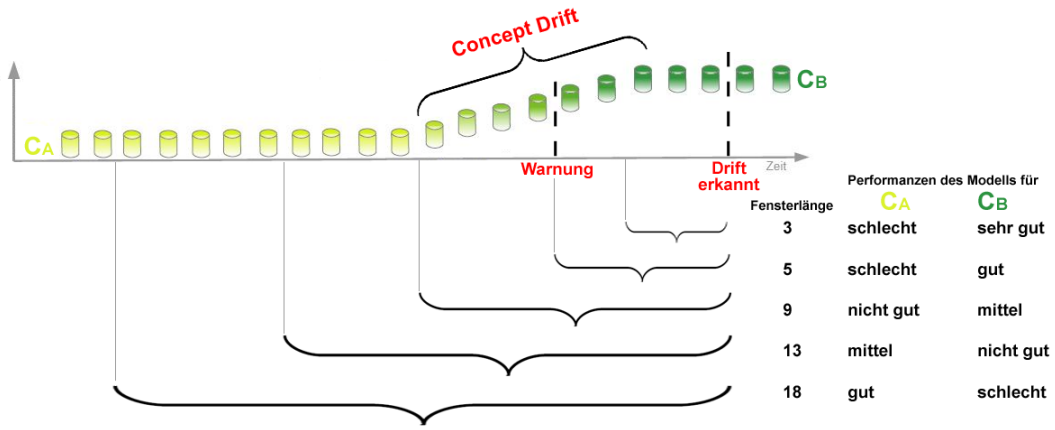

Abbildung 4.2: Auswirkung der Fenstergröße zur Genauigkeitsberechnung für verschiedene Modellinstanzen. Die Dimensionen sind zur Illustration kleiner gewählt, als sie für einen realen Datenstrom sinnvoll sind.

Einerseits ist es erstrebenswert, nur wenige der zuletzt gesehenen Daten in die Berechnung einfließen zu lassen, um einen sehr aktuellen Wert der Qualität verwenden zu können. Andererseits muss sichergestellt werden, dass die errechnete Genauigkeit eine sinnvolle und gegen Outlier und Rauschen robuste Projektion der Qualität einer Modellinstanz ist. (Letzteres ist in Abb. 4.2 mit den beiden kleinsten Fenstern nicht sichergestellt.) Fließen zu viele Items von C_A ein, sticht dessen Modellinstanz möglicherweise die eigentlich bessere für C_B aus (Abb. 4.2, Fenster der Länge 13 und 18 (mit steigender Länge wahrscheinlicher)). Enthält das Fenster vornehmlich Items der Driftphase, ist von den Modellen keine besonders hohe Leistung zu erwarten, unerheblich davon, für welches Concept die Instanzen spezialisiert wurden; schlimmer noch: durch Zufall kann es passieren, dass eine völlig unbeteiligte Instanz, welche z.B. für C_C trainiert wurde, eine gute Qualität auf den Daten dieser Driftphase liefert (Abb. 4.2, Fenster der Länge 9). Von Interesse ist aber, welche Modellinstanz für das *aktuell vorliegende Concept* (C_B) die höchste Performanz bieten wird. Auf dieser Grundlage kann dann entschieden werden, welches das aktuelle Concept *ist*.

4.2 Struktur erkennen und reagieren

Ist ein Concept identifiziert, kann eine *Modellverwaltung* diese Information speichern und so nach jedem Drift das Wissen über die Conceptabfolge des bisherigen Streams aktualisieren. Mit 4.1 wird überprüft, ob das aktuelle Concept einem vorherigen gleicht[5]. Wiederholt sich eine Teilmenge von Concepts nach 3.1, ist ein Zyklus erkannt. Die Modellverwaltung geht nun davon aus, dass sich diese Struktur genauso (3.2) oder so ähnlich (3.3) wiederholt.

[5]Dafür ist es nötig, für jedes Concept eine spezialisierte Modellinstanz (siehe 4.3) zu verwalten und kontinuierlich anzuwenden um die Performanzen vergleichen(4.1) zu können.

Die Reihenfolge der Concepts ist nun bekannt. Beim nächsten Drift wird die Modellinstanz verwendet, die im Zyklus zuvor für das nun folgende Concept trainiert und verwandt wurde. Wann und mit welchen Daten eine Instanz trainiert wird entscheidet die Modellverwaltung unter Berücksichtigung folgender Gesichtspunkte:

4.3 Trainingsmenge und -zeitpunkte

Entscheidend für die Qualität einer Vorhersage ist, dass die zugrundeliegende Modellinstanz mit möglichst rausch- und ausreißerfreien Daten desjenigen Concepts trainiert wurde, zu welchem die Vorhersage getroffen werden soll.
Die Trainingsmenge sollte genügend Items enthalten um das Concept umfangreich und angemessen zu repräsentieren. Sehr viele Trainingsdaten zu verwenden ist unkritisch, zu wenige hingegen birkt das Risiko eine schlecht angepasstes Modellinstanz zu verwenden[6].
Dadurch allerdings, dass jedes Concept (vom ersten Concept abgesehen) nach einem Drift das erste mal auftritt und es erstrebenswert ist, dass unmittelbar ein trainiertes Modell dafür vorliegt, sollte der Zeitpunkt das Training abzuschließen nach diesem Gesichtspunkt nicht allzu spät sein, d.h. die Trainingsmenge darf nicht zu groß sein.
Die passende Balance zu finden ist auch Thema dieser Bachelorarbeit: Da nicht bekannt ist, wie lang ein Concept bereits vorliegt[7], muss während der Laufzeit empirisch festgestellt werden, mit welchen Daten trainiert wird und wann das Training abgeschlossen sein soll.

Der ideale Einsatzzeitpunkt einer Modellinstanz $\hat{f}_i$ ist bei Beginn des Concepts, für das $\hat{f}_i$ trainiert wurde. D.h. sofern es bereits trainiert ist, liegt dieser Zeitpunkt mitten im Drift (siehe Abb. 2.2). An der Stelle, an der sich die Performanzen der Instanzen schneiden: Beispielsweise wenn zum ersten mal gilt: $\text{Accuracy}_W(\hat{f}_i) > \text{Accuracy}_W(\hat{f}_{i-1})$. Dieser Zeitpunkt wird in der Praxis nie erreicht werden können, da erstens die Drifterkennung einen Drift erst *nach* einem solchen erkennen kann und i.d.R. für das Training „saubere“ Daten (s.o.) eines Concepts benötigt werden.
Das Risiko, ein nicht ausreichend trainiertes Modell zur Anwendung zu bringen, muss also gegen den Performanzverlust abgewogen werden, der auftritt, wenn ein veraltetes Modell für die Vorhersagen genutzt wird.
Das *kontinuierliche* Anpassen einer Modellinstanz an deren Concept ist wichtig, wenn sich Concepts über mehrere Zyklen hinweg weiterentwickeln.

[6] Durch *zu viele* Trainingsdaten kann es unter anderen Umständen zu Overfitting [11] kommen. Dieses Problem ist hier nicht relevant, da es aus den nachfolgend im Text genannten Gründen nötig ist das Training zu beenden, deutlich bevor Overfitting ein Problem darstellt.

[7] Abhängig von den Daten brauchen Driftdetektoren mehr oder weniger Zeit um Drifts erkennen zu können.

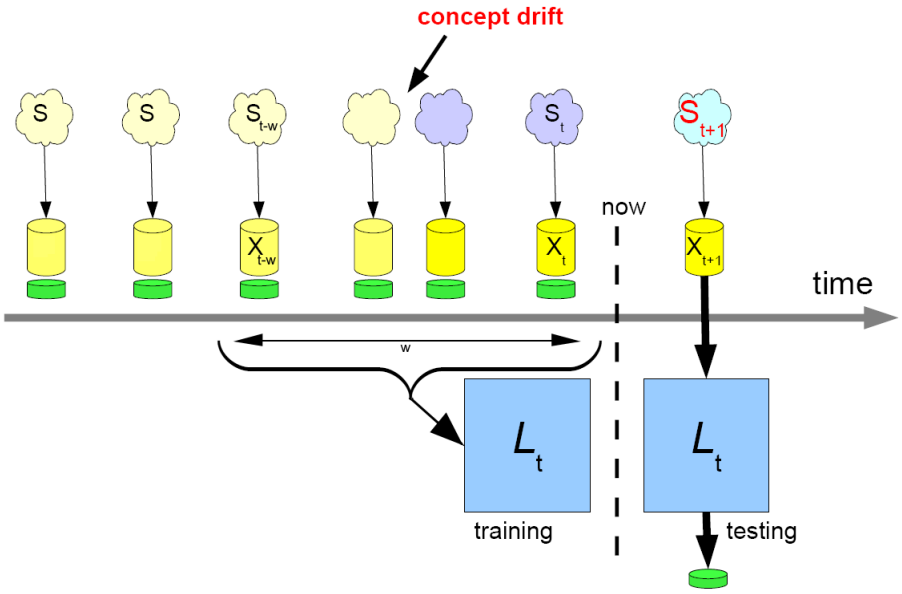

Abbildung 4.3: Veranschaulichung der Trainingsmenge [23]: Nach dem Item X_t wurde ein Drift erkannt und der neu trainierte Lerner L_t soll das Label von X_{t+1} vorhersagen. L_t wurde auf einem Fenster der Länge w trainiert; wie viele Items in diesem Fenster von welchem Concept stammen lässt sich nicht feststellen.

4.4 Wissen über Concepts aktualisieren

Es wäre naiv anzunehmen, die Concepts seien von Zyklus zu Zyklus immer exakt gleich oder träten immer in der genau gleichen Reihenfolge auf. Durch das Zusammenspiel vieler unterschiedlicher (phasenverschiedener) Faktoren (siehe 3.4) und der sich ständig ändernden Umgebung ist es gut denkbar, dass sich Concepts ändern[8] oder im Zyklus die Position tauschen (siehe z.B. Abb. 3.2). Daher ist es wichtig, dass die Modellverwaltung die beste Modellinstanz jedes Concepts so aktuell wie möglich hält. Das kann durch Training mit aktuellen Daten des Concepts erreicht werden.
Auch wahrscheinlich ist, dass über die Zeit neue, nie gesehene Concepts auftreten oder alte verschwinden (siehe z.B. Abb. 3.3). Deshalb muss eine Modellverwaltung diese Veränderungen erkennen und mit entsprechenden neuen Modellen oder Modellinstanzen reagieren bzw. unbrauchbare verwerfen. Das Wissen über die Conceptsabfolge (und darin enthaltene Zyklen) muss immer aktuell gehalten werden um auf Verschiebungen in der Reihenfolge reagieren zu können.
Eine *praktisch* implementierte Modellverwaltung muss diese Flexibilität zuverlässig umsetzen.

[8] Kleine Änderungen ($< \lambda$) lässt die Definition der Gleichheit von Concepts zu (siehe 4.1). Auch langsame inkrementelle Veränderungen eines Concepts an der Position j über mehrere Zyklen $(\ldots z_i, z_{i+1}, z_{i+2} \ldots)$ hinweg sind zulässig: $(C_{i,j} = C_{i+1,j}) \wedge (C_{i+1,j} = C_{i+2,j}) \not\Rightarrow C_{i,j} = C_{i+2,j}$

Kapitel 5

Implementierung

Es sind bereits nützliche Mechanismen zur Datenstromanalyse vorhanden; Mit **streams** existiert eine anpassbare Umgebung für Online-Lernverfahren und -Algorithmen. Auch Modelle, Lerner und Evaluatoren sind zum Teil vorgefertigt und/oder anpassbar (5.2).
Das Hauptaugenmerk liegt im Folgenden auf den Anpassungen des Datengenerators (5.3) und des **streams** Frameworks, sowie auf der Implementierung eines Zyklengenerators und der Modellverwaltung.

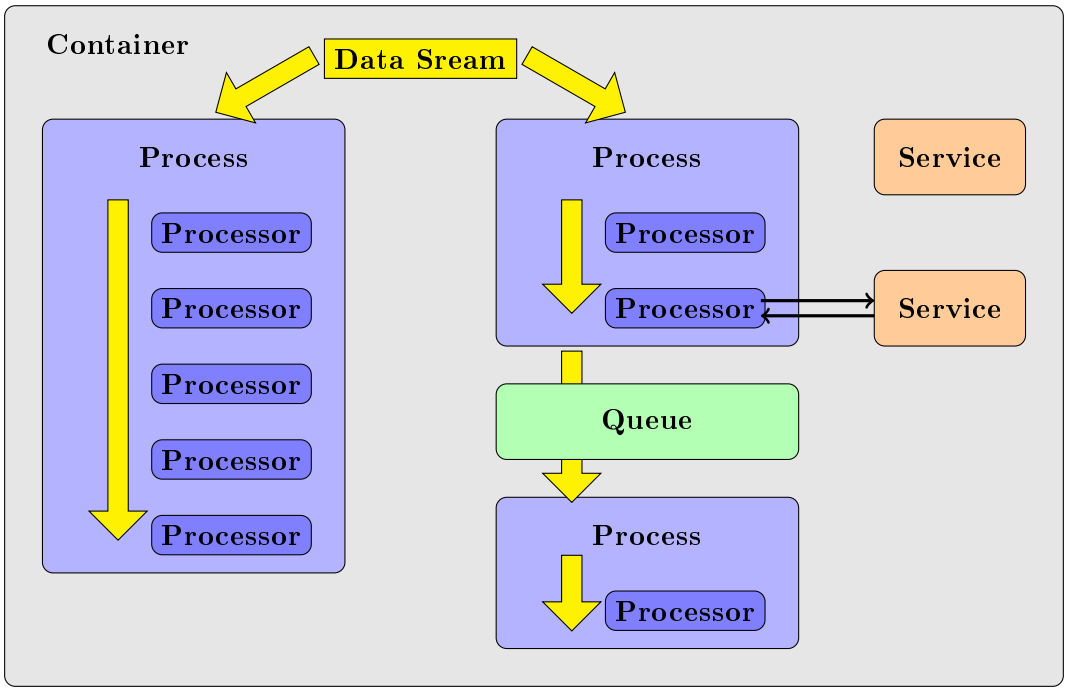

Abbildung 5.1: Schematischer Ablauf in einem **streams**-Container [5]

```
<container>
        <stream  id="stream"
                class="stream.generator.ExampleStream"
                limit="1000"/>

        <process>
                <stream.learner.Prediction learner="lerner1" />
                <stream.learner.evaluation.PredictionError />
                <stream.classifier.MyClassifier id="lerner1" alpha="0.01" />
        </process>
</container>
```

Abbildung 5.2: XML Definition eines Containers [15]

5.1 Streams

Bei der Online Analyse von Datenströmen existieren viele Aufgabenbereiche nebeneinander: Daten müssen aufbereitet, auf unterschiedliche Arten verrechnet, zum Trainieren verwandt, bewertet und visualisiert werden. Ergebnisse, Vorhersagen und Metadaten zu all diesen Verfahren müssen erstellt und verarbeitet werden.
Das **streams** Framework [5] verwaltet solche Aufgaben und ermöglicht es dem Anwender eine Vielzahl eigener hochkomplexer Analyseabläufe zu kontrollieren. Es ist eine Umgebung zur modularen Konkatenation mehrerer Einheiten, welche je eine der o.g. Aufgaben bewältigen. Zum Erstellen von Daten kann eine Quelle angegeben werden, aus der dann ein Stream erzeugt wird, dieser durchläuft dann einen oder mehrere **streams**-Prozesse (siehe Abb. 5.1). Prozesse enthalten zum einen *Processors*, die hintereinander Daten verarbeiten (siehe Abb. 5.3), und zum anderen *Services*, die als flexible Einheiten in Prozessoren oder (ungebunden an lineare Ausführung) im Prozess selbst verwandt werden können. Jeder Prozessor stellt eine Datenverarbeitungseinheit dar. Das kann Aufbereitung der Daten sein, das Anwenden oder Trainieren von Modellinstanzen oder die Analyse und Bewertung der Ergebnisse. Ein Service dient zur Verwaltung des Datenflusses; Entscheidungen, welche Prozessoren in welcher Reihenfolge mit einem Datum ausgeführt werden, treffen Services. Dies führt zu einer klaren Trennung von Kontroll- und Datenfluss [5].
Um Daten zwischen Prozessen und Prozessoren zu puffern sind *Queues* vorhanden.
Für die übersichtliche Verwaltung so vieler Aufgaben lassen sich *Container* definieren, in welchen die Parametrisierung und Abfolge von Streams, Prozessen, Prozessoren und Services definiert wird (siehe Abb. 5.1). Praktisch ist ein Container ein XML Dokument, das den gesamten Ablauf der Datenstromanalyse spezifiziert (Abb. 5.2).

Die Daten werden durch eine Map wie in 2.1.2 angedeutet repräsentiert [4]. Das ermöglicht den Prozessoren Informationen in Datenflussrichtung auszutauschen, indem sie diese an ein Data Item annotieren (siehe Abb. 5.3). Die Funktionen der Prozessoren können frei

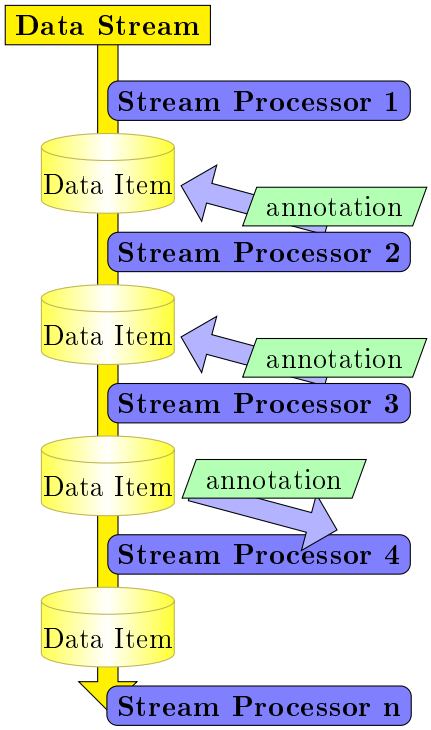

Abbildung 5.3: Verarbeitungsablauf eines **Streams** Prozesses nach [4, 15]

gewählt werden: entweder definiert der Anwender eigene Funktionalität oder wählt aus einer Menge flexibel einsetzbarer Funktionen. Ein Beispiel hierfür kann ein Vorhersagemodell aus dem MOA Framework sein.

5.2 MOA, WEKA

Das Massive Online Analysis (MOA) Framework ist eine Umgebung für Analyse- und Lernverfahren auf Datenströmen mit dem Fokus auf Klassifikation, Regression und Clustering. MOA bietet vorgefertigte Modelle zur Vorhersage[1][15, 5]. Weiterhin existieren Mechanismen um Outlier zu erkennen. Concept Drift-Detektoren und Behandlungsmethoden sind umgesetzt und erweiterbar. Es ist möglich (und vorgesehen) individuelle Lerner, Detektoren und Behandler umzusetzen um die Analyse Bedürfnissen der Anwendung anzupassen.

Die Waikato Environment for Knowledge Analysis (WEKA)-Bibliothek lässt Anwender auf eine Vielzahl bewährter Datenfilter, Modelle für maschinelles Lernen und Data Mining zugreifen. Es sind vorgefertigte Programme zur Klassifikation und Regression vorhanden[2, 10], die mit einem Wrapper wie MOA-Verfahren konfiguriert und benutzt

[1] z.B. Naives Bayes-Lernen, Entscheidungsbäume und verschiedene Ensemble dieser und anderer Modelle.

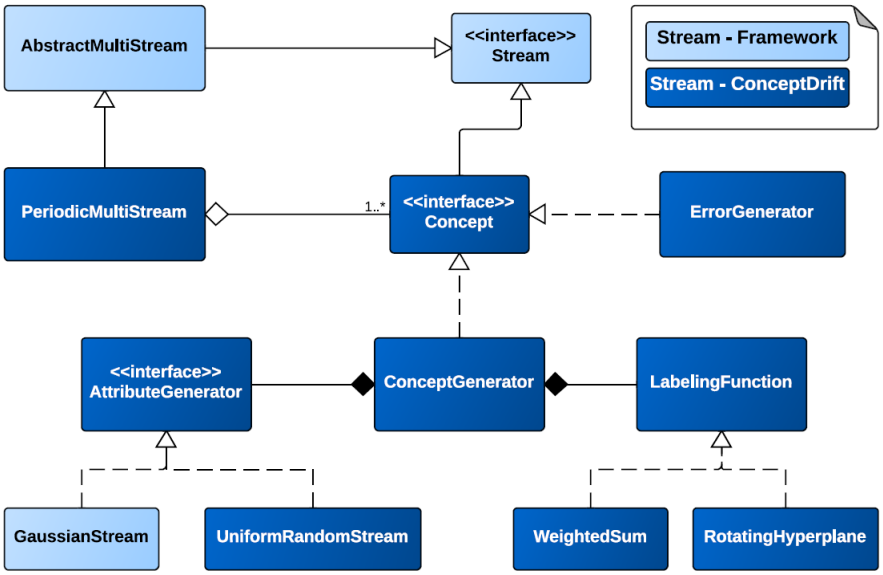

Abbildung 5.4: Vereinfachtes Klassendiagramm des Generator-Frameworks in [15]

werden können[15].
Diese Lernfervahren benötigen Daten zum Training und als Eingabe für eine Vorhersage.

5.3 Generator

Mit dem Conceptgenerator von Stefan Rötner ([15]) lassen sich Data Items verschiedener Concepts von anpassbarer Länge erzeugen. Diese Items werden durch den *rotating Hyperplane*- Algorithmus gelabelt, variabel mit Rauschen versetzt und als Stream ausgegeben (siehe Abb. 5.4).

Um Abwandlungen zwischen wiederkehrenden Concepts umzusetzen wurde auf dieser Basis ein Szenario Generator umgesetzt, der auf Aufforderung Gleichheiten zwischen Concepts aufeinanderfolgender Perioden in definierter Weise verwischt. Außerdem kann der Generator, wenn gefordert, die Reihenfolge und das Ausbleiben von Concepts beeinflussen (siehe 3.3 und Abb. 3.2 und 3.3). Mit dem Szenario Generator lassen sich die Concepts $g(\vec{p}, l)$ (2.1) kontrolliert verändern, indem die Gewichtung der *Label Funktion* (s.o.) mit einer definierten Wahrscheinlichkeit und um einen einstellbaren Wert angepasst wird.
Auch die Reihenfolge der Concepts im Zyklus $f(\hat{C})$ (2.2) kann veränderlich gestaltet werden: Zum einen durch Angeben von Distanz, um die ein Concept im Zyklus „verschoben“ wird und die Wahrscheinlichkeit mit der dies eintritt, zum anderen durch Festlegen von Wahrscheinlichkeiten für das Ausbleiben eines bekannten bzw. Vorkommen eines neuen Concepts.
Auf diese Weise lassen sich reproduzierbare Datensätze und Szenarien, in denen Lerner und

Modellverwaltung mit verschiedenen (aber zyklisch auftretenden) Concepts und Driftarten konfrontiert sind, erstellen.

5.4 Modellverwaltung

Der Vorgang, passende Modellinstanzen zu trainieren und über deren Anwendung zu entscheiden ist in verschiedene **streams**- Prozessoren aufgeteilt (siehe Abb. 5.5).

Zuerst wird von einem Prediction-Prozessor nach dem Vorbild in [15] für jede trainierte Modellinstanz[2] eine Vorhersage für das aktuelle Datum angefordert und an das Data Item annotiert.

Ein weiterer Prozessor berechnet aus dem (vom Szenario Generator) annotierten Label und der Vorhersage den Fehler jeder Modellinstanz und schreibt auch diesen als Annotation in das Datum.

Der nächste Prozessor hält für jedes Modell die gemittelte Genauigkeit über ein gewisses Fenster[3] vor, rechnet den aktuellen Fehler mit ein und fügt dem Datum diese Genauigkeit per Annotation an.

Jetzt folgt der Kern der Modellverwaltung, der Modellmanager, dieser setzt einen Prozessor um, der die gesehene Conceptfolge speichert und in dem ein Driftdetektor für den Datenstrom angewandt wird[4]. Bei jedem Item wird der Status des Driftdetektors überprüft und dem weiteren Prozess per Annotation mitgeteilt, welche Modellinstanz zu verwenden und welche zu trainieren ist:

- **Kein Drift:** Die aktuelle Modellinstanz wird beibehalten und weiter trainiert. Zusätzlich wird deren Genauigkeit (abhängig von erwähntem Fenster) im Modellmanager gespeichert.
- **Warnung:** Die aktuelle Instanz wird weiter verwendet. Zusätzlich wird eine neue auf den aktuellen Daten trainiert. Wenn für das vorliegende Concept ein Nachfolger bekannt ist, wird die Modellinstanz für diesen Nachfolger trainiert. Die Performanz

[2] Im Unterschied zu [15], dort wird nur eine Instanz benutzt. Hier handelt es sich beim Start des Experiments um eine einzelne Instanz, es kommen mit aktualisieren der Conceptliste nach 4.4 mehr Modellinstanzen dazu.

[3] Die Fenstergröße spielt hier eine entscheidende Rolle. siehe 4.1.1

[4] In dieser Implementierung wird EDDM (Early Drift Detection Method) [3, 1] als Detektor gewählt. Diese Methode hat sich im Laufe der Arbeit empirisch als die beste erwiesen.

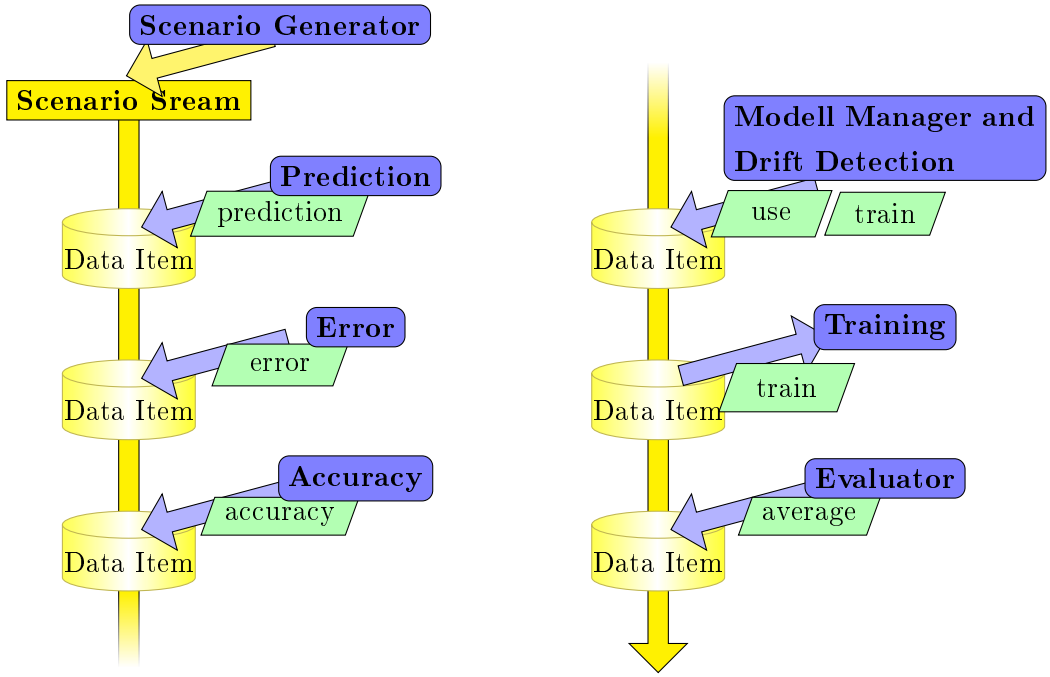

Abbildung 5.5: Verarbeitungsablauf der Modellverwaltung. Um die Lesbarkeit zu verbessern wurden einige Lesezugriffe nicht berücksichtigt.

der Instanzen wird *nicht* gespeichert, da beide in der Driftphase unterdurchschnittlich gute Performanzen aufweisen.

- **Drift erkannt:** Der Übergang zwischen beiden Concepts ist abgeschlossen, jetzt kann überprüft werden, ob es sich bei dem neuen um das vermutete Concept handelt (siehe 4.1). Die Modellinstanz für entsprechend erkanntes Concept wird angewandt und weiter trainiert. Die Drifterkennung wird zurückgesetzt.

Der folgende Prozessor trainiert die Modellinstanz, welche vom Modellmanager zum Training ausgewählt ist.
Zuletzt ist ein Prozessor als Evaluator eingereiht, der die Entscheidung der Modellverwaltung umsetzt und die ID der zur Verwendung ausgewählten Modellinstanz für eine mögliche spätere Visualisierung annotiert[5]. Außerdem berechnet dieser Prozessor die durchschnittliche Performanz aller Modellinstanzen über den gesamten Datenstrom.

Nach der Implementierung des Generators und der Modellverwaltung können nun Experimente mit zyklischen Daten durchgeführt werden.

[5] **Streams** bietet Schnittstellen zum Speichern von Ergebnissen und Plotter zur Visualisierung an. Solche Aufgaben können einfach als Prozessor in den Kontrollfluss integriert werden, sind aber in Abb. 5.5 nicht aufgeführt.

Kapitel 6

Experimente

Bei den Experimenten gilt es nun zu überprüfen, ob sich die Vorhersagequalität der Modellverwaltung von der herkömmlicher Concept Drift Behandlungen unterscheidet.
Verglichen werden:

- Der naive Ansatz (**naiv**): in regelmäßigen Abständen (1000 Items) wird die angewandte Modellinstanz neu trainiert (die Trainingsmenge umfasst 120 Items).
- Drifterkennung (**DDM**): Die *Drift Detection Method* wird auf den Stream angewandt; ist ein Drift erkannt, wird die Modellinstanz auf den aktuellen Daten trainiert (mit 50 Items ab Drift).
- Die Modellverwaltung (**MVStrategie**) funktioniert wie in Kapitel 4 beschrieben.
- (**MVBest**): Die Conceptliste der Modellverwaltung wird zum Erkennen eines Concepts nicht verwendet, zum Driftzeitpunkt wird die Instanz mit der besten Performanz angewandt, bis erneut ein Concept Drift stattfindet. Allerdings wird das beschriebene Vorgehen (inklusive Conceptliste) zur Auswahl der zu trainierenden Instanzen benutzt.
- (**MVEnsemble**): Wieder wird die Liste der Conceptabfolge zur Auswahl der zu trainierenden Modellinstanzen genutzt, auch wird nur bei Warnung des Driftdetektors trainiert. Im Unterschied zu MVBest allerdings wird zu jeder Zeit (nicht nur bei Drifterkennung) die Instanz mit bester Performanz zur Vorhersage genutzt.

Damit die Modellverwaltung die bestmöglichen Ergebnisse liefert müssen zunächst wichtige Einstellungen bekannt sein.

6.1 Finden geeigneter Parameter

Um die Konfiguration der Modellverwaltung (4) zu verbessern wurden vorab empirische Tests durchgeführt, in denen optimal zyklische Daten mit unterschiedlichen Einstellungen

verarbeitet wurden. Auf diese Weise konnte ein guter Bereich für das Genauigkeitsfenster (4.1.1) festgelegt werden[1], die benötigte Wartezeit bis das neue Modell identifiziert wird[2] und einige interne Schwellwerte, die zur Identifikation von Concepts unabdingbar sind[3]. Nach der Feinabstimmung der Modellverwaltung können verschiedene Szenarien getestet werden. Da unbekannt ist, in welcher Weise Zyklen in Datenströmen auftreten (siehe 3.4), werden alle folgenden Experimente zusätzlich zum einen mit veränderten Conceptreihenfolgen $f(\hat{C})$, zum anderen mit evolvierenden Concepts $g(\vec{p}, l)$ durchgeführt.

6.2 Szenarien

Um die Experimente vergleichbar zu machen, werden immer zehn Concepts der Länge 200 in sechs Zyklen getestet. Als Lerner wird der naive Bayes- Klassifikator genutzt. Dabei werden systematisch Kombinationen der folgenden Unterscheidungen für f, $\vec{p}$, und l für den Stream benutzt:

- Ideal: Keine Veränderungen werden vorgenommen.
- Vertauschung: Pro Zyklus werden Concepts bzgl. ihrer Position vertauscht. Dabei variiert die Tauschdistanz von nah (2 Concepts überspringen) bis weit (7 überspringen) und die Anzahl der Vertauschungen von wenig (1 mal tauschen) bis viel (4 Vertauschungen).
- Ausbleiben/Erscheinen: In jedem Zyklus kommen neue Concepts mit vorher spezifizierter Wahrscheinlichkeit vor oder bekannte Concepts bleiben aus. Es wird in zwei Szenarien unterschieden: wenig Wechsel (0,1 Wahrscheinlichkeit für das Ausbleiben, 0,2 Wahrscheinlichkeit für das Auftreten) und viele Wechsel (0,3 Wahrscheinlichkeit für das Ausbleiben, 0,5 Wahrscheinlichkeit für das Auftreten).
- geänderte Conceptlänge: Mit einer festgelegten Wahrscheinlichkeit ändert sich von einem Zyklus zum nächsten die Länge eines Concepts um maximal die Hälfte. Es wird länger oder kürzer.
- geänderte Labelverteilung im Concept: Die Gewichte der *Label Funktion* (siehe 5.3) um Maximal 0,01 ; 0,1 ; 0,3 oder 0,6.

[1] ca. 80-110 Items
[2] ca. 5-10 Items (auch abhängig von der Fenstergröße)
[3] z.B. λ aus 4.1

6.3 Ergebnisse

Für jedes Experiment wurde beobachtet, wie gut der Driftdetektor funktioniert hat: 60 Concepts ergeben 59 tatsächliche Drifts, weicht die Zahl der **Detektionen** weit ab, wird die Erkennung einer Struktur in diesem Experiment nicht optimal funktioniert haben. Eine Ausnahme bilden Experimente, in denen der Zyklus so verändert wurde, dass die Anzahl der Concepts variabel ist.
Im direkten Zusammenhang dazu steht die **Anzahl der Modelle**, in diesem Fall gleichbedeutend mit der Anzahl der Modellinstanzen, welche Auskunft darüber gibt, wie viele Concepts ein erkannter Zyklus enthält, sprich, wie exakt die Concepterkennung funktioniert hat. Wird kein Zyklus entdeckt, nutzt die Modellverwaltung bei jedem Drift eine neue Modellinstanz, die Anzahl ist aus praktischen und Performanz-Gründen auf zehn limitiert worden.
Zu jedem Zeitpunkt wird *eine* dieser Instanzen zur Vorhersage verwendet; Wie gut die Vorhersage der angewandten Instanzen ist, drückt die **Accuracy** aus[4].
Der **Durchschnitt** der Accuracies aller *verfügbaren* Modellinstanzen gibt einen Überblick über deren Spezialisierung: Ist der gemessene Wert nah der Accuracy der verwendeten Instanz, sind sich die Instanzen entweder sehr ähnlich oder Auswahl der Instanz war suboptimal.
Nicht aufgeführt aber gemessen wurde die Dauer jedes Experimentes (siehe 6.4).

6.3.1 Exakt gleiche Concepts

Um die genannten Werte in Relation setzen zu können, wurde zunächst eine Experimentreihe mit unveränderten Concepts durchgeführt.

Vorgehen	Detektionen	Anzahl Modelle	Accuracy	Durchschnitt
naiv	0	1	0.7684700135	0.7684700135
DDM	6	1	0.8485614602	0.8485614602
MVStrategie	60	10	0.8835620734	0.8678067344
MVBest	59	10	0.8620293929	0.8515467847
MVEnsemble	22	5	0.8728772495	0.8509859219

Tabelle 6.1: Ideal zyklische Conceptfolge, keine Änderungen der Labelverteilung oder Conceptlänge

[4]Das Fenster ist die Länge des gesamten Streams.

Vorgehen	Detektionen	Anzahl Modelle	Accuracy	Durchschnitt
naiv	0	1	0.7631878103	0.7631878103
DDM	103	1	0.8480217413	0.8480217413
MVStrategie	60	10	0.8715345043	0.8596405324
MVBest	60	10	0.8514973446	0.8405215357
MVEnsemble	60	10	0.8684244125	0.8516164592

Tabelle 6.2: Vertauschte Concepts (Distanz 2, Anzahl 1), keine Änderungen der Labelverteilung oder Conceptlänge

Vorgehen	Detektionen	Anzahl Modelle	Accuracy	Durchschnitt
naiv	0	1	0.7574648368	0.7574648368
DDM	10	1	0.8465906306	0.8465906306
MVStrategie	60	10	0.8856142833	0.8627474739
MVBest	60	10	0.8595285116	0.8484042886
MVEnsemble	33	8	0.8717231162	0.8523444684

Tabelle 6.3: Vertauschte Concepts (Distanz 7, Anzahl 1), keine Änderungen der Labelverteilung oder Conceptlänge

Vorgehen	Detektionen	Anzahl Modelle	Accuracy	Durchschnitt
naiv	0	1	0.7643048467	0.7643048467
DDM	34	1	0.8481444687	0.8481444687
MVStrategie	26	5	0.8856529113	0.8644592749
MVBest	60	10	0.8565319564	0.8526004927
MVEnsemble	38	9	0.8683131298	0.8497113995

Tabelle 6.4: Vertauschte Concepts (Distanz 2, Anzahl 4), keine Änderungen der Labelverteilung oder Conceptlänge

Vorgehen	Detektionen	Anzahl Modelle	Accuracy	Durchschnitt
naiv	0	1	0.7643205016	0.7643205016
DDM	5	1	0.788906548	0.788906548
MVStrategie	48	10	0.8779744438	0.8647216671
MVBest	44	10	0.860712222	0.8452700126
MVEnsemble	60	10	0.8635393444	0.8573729021

Tabelle 6.5: Vertauschte Concepts (Distanz 7, Anzahl 4), keine Änderungen der Labelverteilung oder Conceptlänge

Vorgehen	Detektionen	Anzahl Modelle	Accuracy	Durchschnitt
naiv	0	1	0.758480428	0.758480428
DDM	11	1	0.8479169123	0.8479169123
MVStrategie	42	10	0.8748312123	0.8623931541
MVBest	49	10	0.850752481	0.8390630634
MVEnsemble	4	2	0.8719697859	0.8673044092

Tabelle 6.6: Ausbleibende-/Erscheinende Concepts (Wahrscheinlichkeiten 0,1 ; 0,2), keine Änderungen der Labelverteilung oder Conceptlänge

Vorgehen	Detektionen	Anzahl Modelle	Accuracy	Durchschnitt
naiv	0	1	0.7679783696	0.7678719979
DDM	32	1	0.8552608513	0.8552608513
MVStrategie	52	10	0.8764981265	0.8662385511
MVBest	38	10	0.8644456572	0.851285045
MVEnsemble	44	2	0.8662966519	0.8696019792

Tabelle 6.7: Ausbleibende-/Erscheinende Concepts (Wahrscheinlichkeiten 0,3 ; 0,5), keine Änderungen der Labelverteilung oder Conceptlänge

6.3.2 Concepts variabler Länge

In den folgenden Experimenten ändert sich zwischen den Zyklen die Länge eines Concepts mit der Wahrscheinlichkeit 0,5. Dabei variiert sie (gaußverteilt) zwischen 100 und 300 Data Items.

Vorgehen	Detektionen	Anzahl Modelle	Accuracy	Durchschnitt
naiv	0	1	0.7254506166	0.7254506166
DDM	4	1	0.8037589185	0.8037589185
MVStrategie	19	2	0.8769647917	0.8410820595
MVBest	2	2	0.8309337878	0.8426522853
MVEnsemble	29	8	0.8693065629	0.8459801375

Tabelle 6.8: Ideal zyklische Conceptfolge, keine Änderungen der Labelverteilung, Conceptlänge variabel

Vorgehen	Detektionen	Anzahl Modelle	Accuracy	Durchschnitt
naiv	0	1	0.7716871561	0.7716871561
DDM	9	1	0.8316242877	0.8316242877
MVStrategie	14	6	0.8707890669	0.8643904707
MVBest	16	7	0.8478434549	0.8311771176
MVEnsemble	14	8	0.8569196697	0.8558181438

Tabelle 6.9: Vertauschte Concepts (Distanz 2, Anzahl 1), keine Änderungen der Labelverteilung, Conceptlänge variabel

Vorgehen	Detektionen	Anzahl Modelle	Accuracy	Durchschnitt
naiv	0	1	0.7620336131	0.7620336131
DDM	5	1	0.822897884	0.822897884
MVStrategie	25	8	0.8610434631	0.8551911584
MVBest	25	8	0.8492123488	0.8301934
MVEnsemble	35	9	0.8689180457	0.8572096492

Tabelle 6.10: Vertauschte Concepts (Distanz 7, Anzahl 1), keine Änderungen der Labelverteilung, Conceptlänge variabel

Vorgehen	Detektionen	Anzahl Modelle	Accuracy	Durchschnitt
naiv	0	1	0.7502419797	0.7502419797
DDM	4	1	0.8415387084	0.8415387084
MVStrategie	34	5	0.8796036629	0.8660652616
MVBest	15	7	0.8589108386	0.8450989825
MVEnsemble	31	5	0.8659443748	0.861436216

Tabelle 6.11: Vertauschte Concepts (Distanz 2, Anzahl 4), keine Änderungen der Labelverteilung, Conceptlänge variabel

Vorgehen	Detektionen	Anzahl Modelle	Accuracy	Durchschnitt
naiv	0	1	0.7680206832	0.7680206832
DDM	8	1	0.8366300943	0.8366300943
MVStrategie	32	6	0.8717807664	0.8253462937
MVBest	47	8	0.8575135083	0.8475507855
MVEnsemble	25	6	0.8670234214	0.8509388397

Tabelle 6.12: Vertauschte Concepts (Distanz 7, Anzahl 4), keine Änderungen der Labelverteilung, Conceptlänge variabel

Vorgehen	Detektionen	Anzahl Modelle	Accuracy	Durchschnitt
naiv	0	1	0.7793896688	0.7793896688
DDM	21	1	0.8133976824	0.8133976824
MVStrategie	27	7	0.8775450681	0.8585679352
MVBest	23	6	0.8578587697	0.8471794832
MVEnsemble	52	7	0.8503044584	0.8421267604

Tabelle 6.13: Ausbleibende-/Erscheinende Concepts (Wahrscheinlichkeiten 0,1 ; 0,2), keine Änderungen der Labelverteilung, Conceptlänge variabel

Vorgehen	Detektionen	Anzahl Modelle	Accuracy	Durchschnitt
naiv	0	1	0.7474747927	0.7474747927
DDM	4	1	0.8652524506	0.8652524506
MVStrategie	14	7	0.873104039	0.8558965414
MVBest	11	3	0.8614318578	0.8571280215
MVEnsemble	13	9	0.8519889981	0.8542847268

Tabelle 6.14: Ausbleibende-/Erscheinende Concepts (Wahrscheinlichkeiten 0,3 ; 0,5), keine Änderungen der Labelverteilung, Conceptlänge variabel

6.3.3 Veränderte Concepts (1)

Um langsam abwandelnde Concepts zu erstellen, wurde wie in 6.2 beschrieben die Labelverteilung angepasst. Die Gewichtung zur Bestimmung des Labels ändert sich für jedes Merkmal der Daten um 0,01. Diese Änderung erfolgt nicht während das Concept auftritt (exakt das wäre ein Concept Drift), sondern *ausschließlich von einem Zyklus zum nächsten.*

Vorgehen	Detektionen	Anzahl Modelle	Accuracy	Durchschnitt
naiv	0	1	0.7578878451	0.7578878451
DDM	10	1	0.8486660681	0.8486660681
MVStrategie	37	10	0.8760494856	0.8594394814
MVBest	60	10	0.8597831026	0.850162384
MVEnsemble	60	10	0.8629089629	0.8523000191

Tabelle 6.15: Ideal zyklische Conceptfolge, Änderungen der Labelverteilung durch anders-Gewichtung der Merkmale um maximal 0,01; konstante Conceptlänge

Vorgehen	Detektionen	Anzahl Modelle	Accuracy	Durchschnitt
naiv	0	1	0.7648390705	0.7648390705
DDM	107	1	0.8508085324	0.8508085324
MVStrategie	60	10	0.8801552144	0.8657175495
MVBest	60	10	0.8595237055	0.8398465383
MVEnsemble	60	10	0.8754056102	0.8587105236

Tabelle 6.16: Vertauschte Concepts (Distanz 2, Anzahl 1), Änderungen der Labelverteilung durch anders-Gewichtung der Merkmale um maximal 0,01; konstante Conceptlänge

Vorgehen	Detektionen	Anzahl Modelle	Accuracy	Durchschnitt
naiv	0	1	0.7612334485	0.7612334485
DDM	10	1	0.8480428144	0.8480428144
MVStrategie	60	10	0.8721528871	0.8628801684
MVBest	59	10	0.855933425	0.84577798
MVEnsemble	24	8	0.87748877	0.8657964271

Tabelle 6.17: Vertauschte Concepts (Distanz 7, Anzahl 1), Änderungen der Labelverteilung durch anders-Gewichtung der Merkmale um maximal 0,01; konstante Conceptlänge

Vorgehen	Detektionen	Anzahl Modelle	Accuracy	Durchschnitt
naiv	0	1	0.7569133335	0.7569133335
DDM	46	1	0.8494538833	0.8494538833
MVStrategie	16	4	0.8777254373	0.863547095
MVBest	23	10	0.8691663785	0.8505671593
MVEnsemble	0	1	0.8724678298	0.8696672696

Tabelle 6.18: Vertauschte Concepts (Distanz 2, Anzahl 4), Änderungen der Labelverteilung durch anders-Gewichtung der Merkmale um maximal 0,01; konstante Conceptlänge

Vorgehen	Detektionen	Anzahl Modelle	Accuracy	Durchschnitt
naiv	0	1	0.7534968585	0.7534968585
DDM	4	1	0.8441880152	0.8441880152
MVStrategie	25	9	0.8843120109	0.86404995
MVBest	60	4	0.8574713905	0.8606011693
MVEnsemble	12	6	0.8748918998	0.8645209046

Tabelle 6.19: Vertauschte Concepts (Distanz 7, Anzahl 4), Änderungen der Labelverteilung durch anders-Gewichtung der Merkmale um maximal 0,01; konstante Conceptlänge

Vorgehen	Detektionen	Anzahl Modelle	Accuracy	Durchschnitt
naiv	0	1	0.7542401567	0.7542401567
DDM	97	1	0.848277225	0.848277225
MVStrategie	50	10	0.870650065	0.8626367503
MVBest	3	3	0.8411738705	0.8379843409
MVEnsemble	42	10	0.8599611854	0.8433568035

Tabelle 6.20: Ausbleibende-/Erscheinende Concepts (Wahrscheinlichkeiten 0,1 ; 0,2), Änderungen der Labelverteilung durch anders-Gewichtung der Merkmale um maximal 0,01; konstante Conceptlänge

Vorgehen	Detektionen	Anzahl Modelle	Accuracy	Durchschnitt
naiv	0	1	0.7608524127	0.7608524127
DDM	5	1	0.7859405403	0.7859405403
MVStrategie	33	10	0.8898987368	0.8673419073
MVBest	15	4	0.8691782552	0.8442507271
MVEnsemble	47	10	0.8746364232	0.8625509348

Tabelle 6.21: Ausbleibende-/Erscheinende Concepts (Wahrscheinlichkeiten 0,3 ; 0,5), Änderungen der Labelverteilung durch anders-Gewichtung der Merkmale um maximal 0,01; konstante Conceptlänge

6.3.4 Veränderte Concepts (2)

Wie die vorherige Testreihe, aber mit einer Änderung der Gewichtung um 0,1.

Vorgehen	Detektionen	Anzahl Modelle	Accuracy	Durchschnitt
naiv	0	1	0.7596429573	0.7596429573
DDM	5	1	0.8446662387	0.8446662387
MVStrategie	60	10	0.8762150666	0.8656462346
MVBest	12	4	0.8310895708	0.8366915246
MVEnsemble	24	6	0.8731934727	0.849801502

Tabelle 6.22: Ideal zyklische Conceptfolge, Änderungen der Labelverteilung durch anders-Gewichtung der Merkmale um maximal 0,1; konstante Conceptlänge

Vorgehen	Detektionen	Anzahl Modelle	Accuracy	Durchschnitt
naiv	0	1	0.7484816143	0.7484816143
DDM	4	1	0.8370400288	0.8370400288
MVStrategie	63	5	0.8859914062	0.8634416187
MVBest	60	10	0.8554953647	0.8368453028
MVEnsemble	39	7	0.8713059519	0.8621287433

Tabelle 6.23: Vertauschte Concepts (Distanz 2, Anzahl 1), Änderungen der Labelverteilung durch anders-Gewichtung der Merkmale um maximal 0,1; konstante Conceptlänge

Vorgehen	Detektionen	Anzahl Modelle	Accuracy	Durchschnitt
naiv	0	1	0.7578079689	0.7578079689
DDM	112	1	0.845808949	0.845808949
MVStrategie	61	3	0.8726915507	0.8491562104
MVBest	60	10	0.8570928506	0.8386779143
MVEnsemble	65	9	0.8631304427	0.8404558687

Tabelle 6.24: Vertauschte Concepts (Distanz 7, Anzahl 1), Änderungen der Labelverteilung durch anders-Gewichtung der Merkmale um maximal 0,1; konstante Conceptlänge

Vorgehen	Detektionen	Anzahl Modelle	Accuracy	Durchschnitt
naiv	0	1	0.7569374395	0.7568541131
DDM	30	1	0.8446140248	0.8446140248
MVStrategie	60	10	0.8759661593	0.8623780931
MVBest	60	10	0.8687037878	0.8408957032
MVEnsemble	60	10	0.8733638792	0.8594146722

Tabelle 6.25: Vertauschte Concepts (Distanz 2, Anzahl 4), Änderungen der Labelverteilung durch anders-Gewichtung der Merkmale um maximal 0,1; konstante Conceptlänge

Vorgehen	Detektionen	Anzahl Modelle	Accuracy	Durchschnitt
naiv	0	1	0.737028164	0.7369448376
DDM	71	1	0.8494157913	0.8494157913
MVStrategie	20	3	0.8764793303	0.8571419735
MVBest	60	10	0.8545636628	0.8531461843
MVEnsemble	5	4	0.8488127379	0.8502130577

Tabelle 6.26: Vertauschte Concepts (Distanz 7, Anzahl 4), Änderungen der Labelverteilung durch anders-Gewichtung der Merkmale um maximal 0,1; konstante Conceptlänge

Vorgehen	Detektionen	Anzahl Modelle	Accuracy	Durchschnitt
naiv	0	1	0.7566923366	0.7566923366
DDM	6	1	0.8337307013	0.8337307013
MVStrategie	41	10	0.8746006428	0.8608996472
MVBest	18	10	0.8407334445	0.843258398
MVEnsemble	21	6	0.8742392771	0.8596251815

Tabelle 6.27: Ausbleibende-/Erscheinende Concepts (Wahrscheinlichkeiten 0,1 ; 0,2), Änderungen der Labelverteilung durch anders-Gewichtung der Merkmale um maximal 0,1; konstante Conceptlänge

Vorgehen	Detektionen	Anzahl Modelle	Accuracy	Durchschnitt
naiv	0	1	0.75353126	0.75353126
DDM	4	1	0.83489486	0.83489486
MVStrategie	51	10	0.8680890295	0.8587910334
MVBest	8	2	0.8617097865	0.8321290385
MVEnsemble	43	10	0.8631008168	0.8541264218

Tabelle 6.28: Ausbleibende-/Erscheinende Concepts (Wahrscheinlichkeiten 0,3 ; 0,5), Änderungen der Labelverteilung durch anders-Gewichtung der Merkmale um maximal 0,1; konstante Conceptlänge

6.3.5 Veränderte Concepts (3)

Wie die vorherigen Experimente, aber mit einer Änderung der Gewichtung um 0,3. Man kann nun nicht mehr von einer Gleichheit der Concepts $C_{i,j}$ und $C_{i+1,j}$ sprechen. Ob die Modellverwaltung trotzdem Zyklen erkennt, soll dieser Durchlauf zeigen.

Vorgehen	Detektionen	Anzahl Modelle	Accuracy	Durchschnitt
naiv	0	1	0.7345825643	0.7345825643
DDM	26	1	0.8284640328	0.8283807064
MVStrategie	46	10	0.8845358018	0.8720117968
MVBest	36	10	0.8571023736	0.8381549344
MVEnsemble	14	4	0.8673933229	0.8537649745

Tabelle 6.29: Ideal zyklische Conceptfolge, Änderungen der Labelverteilung durch anders-Gewichtung der Merkmale um maximal 0,3; konstante Conceptlänge

Vorgehen	Detektionen	Anzahl Modelle	Accuracy	Durchschnitt
naiv	0	1	0.7502478277	0.7502478277
DDM	4	1	0.8523845915	0.8523845915
MVStrategie	51	10	0.887359376	0.87181653
MVBest	24	8	0.8479522751	0.8475149523
MVEnsemble	59	10	0.8510976745	0.8443811509

Tabelle 6.30: Vertauschte Concepts (Distanz 2, Anzahl 1), Änderungen der Labelverteilung durch anders-Gewichtung der Merkmale um maximal 0,3; konstante Conceptlänge

Vorgehen	Detektionen	Anzahl Modelle	Accuracy	Durchschnitt
naiv	0	1	0.738458651	0.738458651
DDM	9	1	0.84593553	0.84593553
MVStrategie	60	10	0.8810153911	0.8636337253
MVBest	60	10	0.859604546	0.8465607281
MVEnsemble	60	10	0.8656183306	0.8508216343

Tabelle 6.31: Vertauschte Concepts (Distanz 7, Anzahl 1), Änderungen der Labelverteilung durch anders-Gewichtung der Merkmale um maximal 0,3; konstante Conceptlänge

Vorgehen	Detektionen	Anzahl Modelle	Accuracy	Durchschnitt
naiv	0	1	0.7501958522	0.7501958522
DDM	8	1	0.8553255724	0.8553255724
MVStrategie	43	7	0.8626169447	0.8601164444
MVBest	6	2	0.8644065268	0.8155060384
MVEnsemble	59	10	0.8503545827	0.8406612835

Tabelle 6.32: Vertauschte Concepts (Distanz 2, Anzahl 4), Änderungen der Labelverteilung durch anders-Gewichtung der Merkmale um maximal 0,3; konstante Conceptlänge

Vorgehen	Detektionen	Anzahl Modelle	Accuracy	Durchschnitt
naiv	0	1	0.7584861106	0.7584861106
DDM	114	1	0.8476778409	0.8476778409
MVStrategie	59	10	0.8562000678	0.8545049616
MVBest	24	8	0.8685401482	0.8576947859
MVEnsemble	33	7	0.8719082826	0.8532824878

Tabelle 6.33: Vertauschte Concepts (Distanz 7, Anzahl 4), Änderungen der Labelverteilung durch anders-Gewichtung der Merkmale um maximal 0,3; konstante Conceptlänge

Vorgehen	Detektionen	Anzahl Modelle	Accuracy	Durchschnitt
naiv	0	1	0.7853923547	0.7852859831
DDM	15	1	0.8465512115	0.8465512115
MVStrategie	45	10	0.8848481028	0.8635338082
MVBest	6	5	0.8202264941	0.8263902402
MVEnsemble	42	10	0.863421006	0.8553228281

Tabelle 6.34: Ausbleibende-/Erscheinende Concepts (Wahrscheinlichkeiten 0,1 ; 0,2), Änderungen der Labelverteilung durch anders-Gewichtung der Merkmale um maximal 0,3; konstante Conceptlänge

Vorgehen	Detektionen	Anzahl Modelle	Accuracy	Durchschnitt
naiv	0	1	0.7267741141	0.7267741141
DDM	29	1	0.8456889087	0.8456889087
MVStrategie	14	3	0.8816018719	0.8794991954
MVBest	48	10	0.8484175621	0.8515593305
MVEnsemble	40	10	0.8483772059	0.8402378629

Tabelle 6.35: Ausbleibende-/Erscheinende Concepts (Wahrscheinlichkeiten 0,3 ; 0,5), Änderungen der Labelverteilung durch anders-Gewichtung der Merkmale um maximal 0,3; konstante Conceptlänge

6.3.6 Veränderte Concepts (4)

Wie die vorherige Testreihe, aber mit einer Änderung der Gewichtung um 0,6. Concepts gleichen sich nicht mehr. Durch Zufall kann eine Ähnlichkeit der Concepts entstehen, zyklisches Auftreten ist jedoch extrem unwahrscheinlich.

Vorgehen	Detektionen	Anzahl Modelle	Accuracy	Durchschnitt
naiv	0	1	0.7362311782	0.7362311782
DDM	111	1	0.8370596781	0.8370596781
MVStrategie	59	9	0.8545122648	0.8506753339
MVBest	21	7	0.8329394887	0.8064014111
MVEnsemble	61	10	0.7987505402	0.7793705395

Tabelle 6.36: Ideal zyklische Conceptfolge, Änderungen der Labelverteilung durch anders-Gewichtung der Merkmale um maximal 0,6; konstante Conceptlänge

Vorgehen	Detektionen	Anzahl Modelle	Accuracy	Durchschnitt
naiv	0	1	0.7502478277	0.7502478277
DDM	4	1	0.8523845915	0.8523845915
MVStrategie	51	10	0.887359376	0.87181653
MVBest	24	8	0.8479522751	0.8475149523
MVEnsemble	59	10	0.8510976745	0.8443811509

Tabelle 6.37: Vertauschte Concepts (Distanz 2, Anzahl 1), Änderungen der Labelverteilung durch anders-Gewichtung der Merkmale um maximal 0,6; konstante Conceptlänge

Vorgehen	Detektionen	Anzahl Modelle	Accuracy	Durchschnitt
naiv	0	1	0.7454505392	0.7454505392
DDM	11	1	0.8191235151	0.8191235151
MVStrategie	60	10	0.8590498304	0.8465375787
MVBest	56	10	0.8319654625	0.8196844374
MVEnsemble	60	10	0.838363459	0.8140633741

Tabelle 6.38: Vertauschte Concepts (Distanz 7, Anzahl 1), Änderungen der Labelverteilung durch anders-Gewichtung der Merkmale um maximal 0,6; konstante Conceptlänge

Vorgehen	Detektionen	Anzahl Modelle	Accuracy	Durchschnitt
naiv	0	1	0.729700649	0.729700649
DDM	49	1	0.8227806624	0.8227806624
MVStrategie	61	10	0.8586093909	0.8343189685
MVBest	36	8	0.8464212688	0.831879921
MVEnsemble	37	10	0.8460056793	0.8369311307

Tabelle 6.39: Vertauschte Concepts (Distanz 2, Anzahl 4), Änderungen der Labelverteilung durch anders-Gewichtung der Merkmale um maximal 0,6; konstante Conceptlänge

Vorgehen	Detektionen	Anzahl Modelle	Accuracy	Durchschnitt
naiv	0	1	0.7854434436	0.7854434436
DDM	5	1	0.8056009811	0.8056009811
MVStrategie	36	8	0.8600963214	0.8410479957
MVBest	32	10	0.8166486019	0.7894420581
MVEnsemble	60	10	0.8439413061	0.8244882189

Tabelle 6.40: Vertauschte Concepts (Distanz 7, Anzahl 4), Änderungen der Labelverteilung durch anders-Gewichtung der Merkmale um maximal 0,6; konstante Conceptlänge

Vorgehen	Detektionen	Anzahl Modelle	Accuracy	Durchschnitt
naiv	0	1	0.7668409754	0.7667190391
DDM	12	1	0.8645629801	0.8645629801
MVStrategie	47	10	0.8905569356	0.8707440864
MVBest	47	10	0.8610238916	0.8524623419
MVEnsemble	40	10	0.8801863892	0.8596061042

Tabelle 6.41: Ausbleibende-/Erscheinende Concepts (Wahrscheinlichkeiten 0,1 ; 0,2), Änderungen der Labelverteilung durch anders-Gewichtung der Merkmale um maximal 0,6; konstante Conceptlänge

Vorgehen	Detektionen	Anzahl Modelle	Accuracy	Durchschnitt
naiv	0	1	0.7514224339	0.7514224339
DDM	10	1	0.8350226543	0.8350226543
MVStrategie	36	10	0.8874176917	0.8613397275
MVBest	21	7	0.8600704922	0.8462187314
MVEnsemble	45	9	0.8620637343	0.8433137977

Tabelle 6.42: Ausbleibende-/Erscheinende Concepts (Wahrscheinlichkeiten 0,3 ; 0,5), Änderungen der Labelverteilung durch anders-Gewichtung der Merkmale um maximal 0,6; konstante Conceptlänge

MVStrategie ist mit einer über alle Szenarien gemittelten Accuracy von 87,46% in den meisten Fällen den anderen Verfahren überlegen. Dabei ist die Accuracy durchschnittlich 11,89% besser als der naive Ansatz, 3,64% als DDM, 2,26% als MVBest und 1,23% als MVEnsemble. MVStrategie trainiert die Modellinstanzen im Durchschnitt zu einer Accuracy von 85,84% (MVBest erreicht 83,96% und MVEnsemble 84,97%).

6.4 Analyse und Bewertung

Die Modellverwaltung bietet einen deutlichen Genauigkeitsvorteil gegenüber DDM und dem naiven Verfahren. Die Modellverwaltung als Ensemble oder nur zum Training mehrerer Modelle zu verwenden macht keinen signifikant großen Unterschied und kann aber aus Gründen der (minimalen[5]) Zeitersparnis trotzdem für eine Anwendung in Frage kommen.

Der Durchschnitt aller Modellinstanzen lässt Rückschlüsse darauf zu, wie die ähnlich die Instanzen trainiert wurden: ein geringer Wert spricht für einen hohen Grad der Spezia-

[5]MVStrategie benötigte im Durchschnitt 1,552 s für ein Szenario, MVBest war in diesen Tests 0,014 s schneller, MVEnsemble 0,028 s. Ein *deutlicher* Unterschied zu MVStrategie bzgl. der Laufzeit ließ sich bei den Verfahren naiv und DDM beobachten: der naive Ansatz war mit 0,647 s mehr als doppelt so schnell, wie die Modellverwaltung. Die DDM benötigte 0,755 s.

lisierung. Der Unterschied zwischen der Performanz der benutzten Instanz und der Durchschnittsperformanz ist bei MVStrategie 1,61%; bei MVBest 1,22% und bei MVEnsemble 1,25%.

Kapitel 7

Fazit und Ausblick

Als erste Arbeit, die sich *praktisch* und in diesem Umfang mit zyklischem Concept Drift befasst[1], schafft sie eine Grundlage für weitere Erforschung von Lernverfahren auf zyklisch strukturierten Daten. Als Bachelorarbeit war die Annäherung an diese Aufgabenstellung nur auf Basis der umfangreichen theoretischen und praktischen Vorarbeit in [15] möglich. Sie stellt ein weitgehend *automatisiertes* Verfahren vor, Zyklizität in Datenströmen zu begegnen. Dabei ist die Qualität der Vorhersagen gegenüber herkömmlichen Concept Drift Behandlungen erkennbar besser.
Selbst basierend auf anpassbarer und flexibler Software zur Datenstromanalyse ist die Methodik der Modellverwaltung erweiterbar und Ausgangspunkt für die Beantwortung offener Fragen an die zukünftige Forschung.

7.1 Offene Fragen

Welche anderen Verfahren zur Verwaltung von Modellinstanzen möglich, effizienter oder performanter sind bleibt offen und ist eine interessante Fragestellung für weitere Forschung. Dafür beschreibt 6.4 bereits eine Methode, indirekt auf die Spezialisierung der verwendeten Instanzen zu schließen. Die hier gesammelte Erfahrung zeigt, dass sich wahrscheinlich durch andere Verfahren noch weiter angepasste Modellinstanzen trainieren lassen. Auch die Erfassung von Zyklen kann noch weiter verbessert werden um flexibel auf unterschiedliche Gegebenheiten zu reagieren.
Ob und inwieweit die getesteten Szenarien eine gute Repräsentation der Wirklichkeit sind, müssen Daten von echten Anwendungen zeigen.

[1] In [18] und anderen Papieren wird erwähnt, dass Concepts wiederholt vorkommen können, eine spezielle Behandlung für Zyklen findet nicht statt.

7.2 Kritik

Auch wenn viele Aspekte der Modellverwaltung eine Vereinfachung und möglicherweise nicht für einige reale Daten geeignet sind[2], konnten die Experimente zeigen, dass sich die Vorhersagequalität von Modellen verbessern ließ, wenn eine Auswahl der Instanzen auf der Struktur der Conceptfolge basierte. Dabei ist die gute Auswahl der Trainingsdaten eine der wichtigsten Voraussetzungen für die spätere Bewertung der Modellinstanzen.

Die wertvollste Erkenntnis dieser Arbeit ist, dass vorhandene, speziell angepasste Modellinstanzen wiederverwendet und weiter trainiert werden sollten. Sie funktionieren besser im Laufe eines wiederkehrenden Concepts als neue Instanzen. Das gilt auch für leicht verändert wiederkehrende Concepts und deren Abfolgen.

[2]Kommt ein Concept im Zyklus mehrmals vor, funktioniert die vorgestellte Methodik nicht und trainiert die falsche Modellinstanz.

Abbildungsverzeichnis

Literaturverzeichnis

[1] Baena-Garcia, Manuel, José del Campo-Ávila, Raúl Fidalgo, Albert Bifet, R Gavalda und R Morales-Bueno: *Early drift detection method.* In: *Fourth international workshop on knowledge discovery from data streams*, Band 6, Seiten 77–86, 2006.

[2] Bifet, Albert und Richard Kirkby: *Data stream mining a practical approach.* 2009.

[3] Bifet, Albert, Jesse Read, Bernhard Pfahringer, Geoff Holmes und Indrė Žliobaitė: *Cd-moa: Change detection framework for massive online analysis.* In: *International Symposium on Intelligent Data Analysis*, Seiten 92–103. Springer, 2013.

[4] Bockermann, Christian: *Mining big data streams for multiple concepts.* Doktorarbeit, 2015.

[5] Bockermann, Christian und Hendrik Blom: *The streams framework.* TU Dortmund University, Tech. Rep, 5:12, 2012.

[6] Bockermann, Christian, Kai Brügge, Jens Buss, Alexey Egorov, Katharina Morik, Wolfgang Rhode und Tim Ruhe: *Online Analysis of High-Volume Data Streams in Astroparticle Physics.* In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Seiten 100–115. Springer, 2015.

[7] Fontenla-Romero, Óscar, Bertha Guijarro-Berdiñas, David Martinez-Rego, Beatriz Pérez-Sánchez und Diego Peteiro-Barral: *Online machine learning.* Efficiency and Scalability Methods for Computational Intellect, Seiten 27–54, 2013.

[8] Friedman, Jerome, Trevor Hastie und Robert Tibshirani: *The elements of statistical learning*, Band 1. Springer series in statistics Springer, Berlin, 2001.

[9] Gama, Joao, Pedro Medas, Gladys Castillo und Pedro Rodrigues: *Learning with drift detection.* In: *Brazilian Symposium on Artificial Intelligence*, Seiten 286–295. Springer, 2004.

[10] Hall, Mark, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann und Ian H Witten: *The WEKA data mining software: an update.* ACM SIGKDD explorations newsletter, 11(1):10–18, 2009.

[11] Hawkins, Douglas M: *The problem of overfitting.* Journal of chemical information and computer sciences, 44(1):1–12, 2004.

[12] Klinkenberg, Ralf und Thorsten Joachims: *Detecting Concept Drift with Support Vector Machines.* In: *ICML*, Seiten 487–494, 2000.

[13] Minku, Leandro L, Allan P White und Xin Yao: *The impact of diversity on online ensemble learning in the presence of concept drift.* IEEE Transactions on Knowledge and Data Engineering, 22(5):730–742, 2010.

[14] Moore, Darnell J, Irfan A Essa und Monson H Hayes: *Exploiting human actions and object context for recognition tasks.* In: *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, Band 1, Seiten 80–86. IEEE, 1999.

[15] Rötner, Stefan und Katharina Morik: *Behandlung von Concept Drift in zyklischen Prozessen.*

[16] Sokolova, Marina, Nathalie Japkowicz und Stan Szpakowicz: *Beyond accuracy, F-score and ROC: a family of discriminant measures for performance evaluation.* In: *Australasian Joint Conference on Artificial Intelligence*, Seiten 1015–1021. Springer, 2006.

[17] Stolpe, Marco, Katharina Morik, Benedikt Konrad, Daniel Lieber und Jochen Deuse: *Challenges for Data Mining on Sensor Data of Interlinked Processes.* In: *Proceedings of the Next Generation Data Mining Summit (NGDM) 2011*, 2011. http://www.kd2u.org/NGDM11/schedule.html.

[18] Tsymbal, Alexey: *The problem of concept drift: definitions and related work.* Computer Science Department, Trinity College Dublin, 106:2, 2004.

[19] Widmer, Gerhard und Miroslav Kubat: *Learning in the presence of concept drift and hidden contexts.* Machine learning, 23(1):69–101, 1996.

[20] Wieland, Dipl-Wirt-Inf BA Uwe, Dipl-Wirt-Inf Marco Fischer und Andreas Hilbert: *Prozessverbesserung im Kontext von Industrie 4.0—ein Geschäftsmodellansatz für IT-Unternehmen.* HMD Praxis der Wirtschaftsinformatik, 50(4):63–72, 2013.

[21] Zaki, M.J., J.X. Yu, B. Ravindran und V. Pudi: *Advances in Knowledge Discovery and Data Mining, Part II: 14th Pacific-Asia Conference, PAKDD 2010, Hyderabad, India, June 21-24, 2010, Proceedings*. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010.

[22] Zhang, Harry: *The optimality of naive Bayes*. AA, 1(2):3, 2004.

[23] Žliobaitė, Indrė: *Learning under concept drift: an overview*. arXiv preprint arXiv:1010.4784, 2010.

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet sowie Zitate kenntlich gemacht habe.

Dortmund, den 11. August 2016

Tobias Rickhoff