Abstract

Reale Anwendungen der Anomalieerkennung in nicht-stationären, hochdimensionalen und multivariaten Zeitreihen stellen hohe Anforderungen an statistische Modellierungsansätze. Klassische Verfahren stoßen hierbei insbesondere bei sich verändernden Datenverteilungen und selten auftretenden Anomalien an ihre Grenzen.

In dieser Arbeit wird ein Boosted-Gaussian-Mixture-Ensemble (BGME) entwickelt, das eine rückwirkende Anpassung an zuvor ungesehene Daten ermöglicht. Der Ansatz kombiniert ein offline trainiertes Ensemble mit Boosting-Mechanismen zur differenzierten Modellierung von Normalzuständen und einer kontrollierten Online-Adaption auf Basis verzögerter Feedback-Informationen.

Die Evaluation erfolgt auf einem realen Datensatz, dem ServerMachineDataset. Um das Langzeitverhalten von BGMEs zu testen, wurde zusätzlich ein synthetischer Datenstrom erzeugt, der mithilfe eines eigens entwickelten Datensimulators generiert wird. Dadurch können sowohl praxisnahe Szenarien als auch gezielte Langzeitmuster untersucht werden.

Die Auswertung der Ergebnisse zeigt, dass die betrachteten Boosted-Gaussian-Mixture-Ensembles über unterschiedliche Maschinen hinweg eine gleichmäßigere Detektionsleistung aufweisen als ein einzelnes Gaussian-Mixture-Modell. Insbesondere im Vergleich zur Baseline ergibt sich eine geringere relative Streuung der F1-Scores, was auf eine höhere Robustheit gegenüber variationsreicher Datencharakteristik hindeutet. Dieser Effekt bleibt auch bei veränderten Hyperparameterkonfigurationen konsistent.

Der Ansatz stellt keine universelle Lösung dar, sondern eignet sich insbesondere für Anwendungen mit überwiegend stabilen Normalzuständen und strukturierten, wiederkehrenden Anomalien.

Inhaltsverzeichnis

1. Einleitung

Die Anomalieerkennung in hochdimensionalen, kontinuierlichen Datenströmen stellt eine zentrale Herausforderung moderner Systeme dar. Insbesondere in industriellen Anwendungen, wie der Zustandsüberwachung von Maschinen oder der Analyse sensornaher Prozessdaten, können Abweichungen vom Normalverhalten auf Defekte, Sicherheitsrisiken oder Leistungseinbrüche hinweisen. Da solche Anomalietypen eher selten auftreten, sich vielfältig ausprägen und Veränderungen im zeitlichen Verlauf aufweisen, stoßen klassische Methoden oft an ihre Grenzen [1].

Überwachte und teilüberwachte Ansätze des Machine Learnings haben sich daher als praktikable Methoden etabliert, weil sie primär Normalverhalten modellieren und Abweichungen identifizieren. Eine weit verbreitete Methode in diesem Kontext sind Gaussian-Mixture-Modelle (GMMs), welche komplexe Datenverteilungen durch eine gewichtete Kombination mehrerer multivariater Normalverteilungen approximieren. GMMs bieten eine probabilistische Interpretation, sind mathematisch nachvollziehbar und lassen sich effizient trainieren. Allerdings zeigen sie in dynamischen Umgebungen mit nicht-stationären Datenverteilungen Einschränkungen, insbesondere bei langfristigen Datenströmen [2].

Um diesen Limitationen zu begegnen, werden Ensemble-Methoden zunehmend untersucht, da sie durch die Kombination mehrerer Modelle eine höhere Robustheit und Anpassungsfähigkeit versprechen. Insbesondere ermöglichen Boosting-basierte Ansätze, die Schwächen einzelner Modelle gezielt auszugleichen und schrittweise komplexere Datenstrukturen zu erfassen [3]. In Verbindung mit Gaussian-Mixture-Modellen entsteht so das Konzept von Boosted-Gaussian-Mixture-Ensembles (BGMEs), die das Normalverhalten in einem kontinuierlichen Datenstrom inkrementell und adaptiv modellieren können.

1.1. Ziel und Beitrag

Ziel dieser Arbeit ist es, ein solches Boosted-Gaussian-Mixture-Ensemble für die Anomalieerkennung in sequentiellen Datenströmen zu entwickeln, zu analysieren und empirisch zu evaluieren. Dabei liegt der Fokus auf der Untersuchung, inwiefern ein ensemblebasierter, teilweise rückwirkend lernender Ansatz gegenüber klassischen Einzel-GMMs Vorteile hinsichtlich Erkennungsleistung, Stabilität und Fehlalarmrate bietet. Die Evaluation erfolgt anhand simulierter sowie aufgezeichneter Daten aus realen Einsatzfeldern, wobei verschiedene Anomalietypen und Konfigurationsparameter systematisch betrachtet werden.

Hierfür wird ein Datensimulator implementiert, der möglichst realistische Bedingungen durch Rauschen, Quantisierung und Clipping bietet und Daten in zwei unterschiedlichen Betriebsmodi generiert. Ein Modus, bei dem nur die Normalzustände und die Übergangszustände modelliert werden, und ein anderer Modus, bei dem Normalzustände kontinuierlich generiert werden und wiederkehrende Anomaliemuster auftreten. Der Simulator bietet verschiedene Konfigurationsmöglichkeiten, um gezielt unterschiedliche Zustände zu simulieren.

Ein realer Datensatz wird für die Hauptevaluation hinzugezogen. Das ServerMachineDataset von NetManAIOps [4] ist ein zeitreihenbasierter Datensatz, der auf realen Servern über einen Zeitraum von fünf Wochen Daten gesammelt hat. Hierbei werden Maschinen in drei Gruppen unterteilt, wodurch ein Datensatz bestehend aus 28 unterschiedlichen Zeitreihen entsteht. Jede einzelne dieser Maschinen hat einen gleich lang aufgeteilten Training- und Testsplit, inklusive der Kennzeichnung von Anomalien.

Zudem wird eine Boosted-Gaussian-Mixture-Ensemble-Architektur erstellt, die auch eine Online-Learning-Komponente bietet. In dieser Architektur werden Normaldaten gezielt durch Gewichtung gelernt und somit eine angepasstere Abbildung der vorliegenden Datenstrukturen geboten. Dabei gibt es verschiedene Modelle, die sich alle auf unterschiedliche Anteile der Normaldaten spezialisieren. In Kombination mit der Online-Learning-Komponente kann ein neues Gaussian-Mixture-Modelle in die Ensembles aufgenommen werden. Dies wird durch die Nutzung von nachträglich verfügbarem Expertenwissen, ob die Einschätzung des zugrundeliegenden Normalmodells korrekt war, ermöglicht. Während ein Teil des Ensembles auf die bestmögliche Abbildung der Normaldaten spezialisiert ist, deckt der andere Teil Anomaliemuster ab und kann diese potenziell wiedererkennen.

Abschließend wird die genannte Architektur auf dem ServerMachineDataset evaluiert und gegen ein einzelnes Gaussian-Mixture-Modell als Baseline gestellt. Die Online-Learning-Komponente ermöglicht zusätzlich die Unterscheidung zwischen Lernen aus Fehlern und dem Lernen aus allen aufgetretenen Datenpunkten. Beide Strategien werden in dieser Auswertung berücksichtigt.

1.2. Aufbau der Arbeit

Im folgenden Kapitel 2 wird die relevante Theorie aufgelistet. Das Boosted-Gaussian-Mixture-Ensemble, sowohl im Aufbau als auch in der Betrachtung der unterschiedlichen Lernstrategien, wird in Kapitel 3 aufgeführt. In Kapitel 4 wird die methodische Vorgehensweise der Arbeit, inklusive des entwickelten Datensimulators, dessen Konfigurationsmöglichkeiten und der reale Datensatz

ServerMachineDataset beschrieben. Die Evaluation der vorgestellten Modelle und Ergebnisse wird in Kapitel 5 aufgezeigt. Die Diskussion aus Kapitel 6 ordnet die Ergebnisse kritisch ein, Stärken und Schwächen des Ansatzes werden diskutiert. Im Fazit und Ausblick aus Kapitel 7 werden die wichtigsten Erkenntnisse der Arbeit zusammengefasst und mögliche Erweiterungen aufgelistet.

2. Theorie

In diesem Kapitel wird die relevante Theorie aufgelistet und beschrieben. Um die folgenden Kapitel im Detail nachzuvollziehen, werden bestimmte Vorkenntnisse benötigt. Aus der Kombination dieser Theorien entsteht dann die Grundidee der Arbeit, das Boosted-Gaussian-Mixture-Ensemble.

2.1. Anomalieerkennung in kontinuierlichen Datenströmen

Um die Anomalieerkennung in kontinuierlichen Datenströmen zu verstehen, wird eine Definition von Anomalien benötigt. Anomalien werden als Unstimmigkeiten gegenüber dem erwarteten Normalverhalten der Daten definiert [1]. Nach Z. Zamanzadeh Darban et al. [5] werden aktuell Transformer-basierte Ansätze, sowie beispielsweise Variational Auto-Encoder für Anomaliedetektion erforscht und getestet. Auch Gaussian-Mixture Ensemble-Ansätze in Kombination mit tiefen Encodern sind aktuell vertreten [6]. Gaussian-Mixture-Modelle sind daher als Baseline vertretbar und aktuell präsent. Wichtig in der Anomaliedetektion ist jedoch, zwischen Anomalien, Ausreißern und Novelty zu unterscheiden. Ausreißer sind Datenpunkte, die stark von der Mehrheit abweichen, sind deswegen aber nicht zwingend interessant oder eine Anomalie. Neue, zuvor nicht beobachtete Muster werden als Novelty beschrieben. Diese sind nicht zwingend anomal, sondern neu. Anomalien können in verschiedenen Mustern auftreten, grundsätzlich werden drei Arten unterschieden: Punktanomalien, kontextbasierte Anomalien und kollektive Anomalien. Punktanomalien sind einzelne anomale Datenpunkte, während kontextbasierte Anomalien nur in Bezug auf andere Datenpunkte anomal sind, zum Beispiel bei Temperaturvergleichen in verschiedenen Jahreszeiten. Kollektive Anomalien sind ganze Gruppierungen von Datenpunkten, die gemeinsam dieselbe Anomalie beschreiben, wobei jeder einzelne Datenpunkt für sich nicht anomal sein muss [1]. Ein Beispiel hierfür sind Sequenzen von Datenpunkten, die über mehrere Zeitschritte hinweg identische Werte annehmen, obwohl zuvor ein schwankendes Normalverhalten vorlag. Die zuverlässige Erkennung von Anomalien ist wichtig, da fehlerhafte Einordnungen erhebliche Auswirkungen haben. Falsch-positive Erkennungen (False Positives) führen zu unnötigen Alarmierungen, während falsch-negative Entscheidungen (False Negatives) zu übersehenen Anomalien führen. In kontinuierlichen Datenströmen mit selten auftretenden Anomalien stellt ein ausgeglichenes Verhältnis zwischen False Positives und False Negatives eine zentrale Herausforderung dar, wobei die Gewichtung stark vom Use-Case abhängig ist.

Aufbauend auf diesem Verständnis von Anomalien muss noch zwischen den verschiedenen Lernarten für einen kontinuierlichen Datenstrom unterschieden werden. Dabei lassen sich drei Arten von Trainingsstrategien unterscheiden, um Anomalien zu erkennen [1]. Beim überwachten Lernen werden Modelle auf annotierten Normal- und Anomaliedaten trainiert. Wenn nur die Normalzustände der Daten bekannt sind, werden Anomalien als Abweichungen von diesen eingeordnet, dabei wird vom teilüberwachten Lernen gesprochen. Unüberwachtes Lernen beschreibt den Fall, in dem keinerlei Annotationen vorhanden sind. Je nach Problemstellung eignet sich einer dieser Ansätze besser als die anderen. In der Regel wird mit allen bestehenden Informationen gearbeitet. Kontinuierliche Datenströme sind im Vergleich zu statischen Datensätzen besonders anspruchsvoll. Solche Datenströme weisen zeitliche Veränderungen auf, sind teilweise nicht-stationär, haben Concept Drift durch Verschiebung der Normalzustände und müssen trotzdem in Echtzeit eingeordnet werden [7].

In dieser Arbeit wird der Fokus auf die Erkennung kollektiver Anomalien in kontinuierlichen Datenströmen, sowohl synthetischen als auch realen, gelegt. Dafür werden teilüberwachte und überwachte Lernverfahren verwendet. Dabei wird jeder Datenpunkt einzeln betrachtet, jedoch steht dieser meist im Zusammenhang mit anderen anomalen Datenpunkten.

2.2. Gaussian-Mixture-Modelle (GMM)

Gaussian-Mixture-Modelle beschreiben Wahrscheinlichkeitsdichten von Datenpunkten als gewichtete Summe mehrerer Normalverteilungen.

2.2.1. Grundidee probabilistischer Dichteschätzung

Kontinuierliche Zufallsvariablen werden durch Wahrscheinlichkeitsdichtefunktionen beschrieben. Eine Wahrscheinlichkeitsdichte $p(x)$ beschreibt, wie dicht Werte in der Umgebung von x auftreten. Das Integral der Dichtefunktion über diesem Intervall ergibt die Wahrscheinlichkeit, dass eine Zufallsvariable x in (a, b) liegt. Die Dichtefunktion selbst dient nicht direkt als Wahrscheinlichkeit, sondern als Grundlage zur Berechnung von Intervallwahrscheinlichkeiten [8].

$$P(x \in (a,b)) = \int_a^b p(x)\,dx \tag{2.1}$$

Um zu bewerten, wie gut ein Datenpunkt x durch ein gegebenes Modell beschrieben wird, wird die Likelihood $p(x \mid \Theta)$ herangezogen, wobei Θ die Menge aller

Modellparameter bezeichnet. Likelihood-Werte können für hochdimensionale Daten sehr klein werden, weshalb in der Praxis meist die Log-Likelihood verwendet wird. Dies vereinfacht die Berechnung, da Produkte von Wahrscheinlichkeiten in Summen überführt werden. Für einen einzelnen Datenpunkt ist diese definiert als

$$\log p(x \mid \Theta) = \log \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(x \mid \mu_k, \Sigma_k) \right), \tag{2.2}$$

wobei π_k die Gewichte, μ_k die Mittelwerte und Σ_k die Kovarianzmatrizen der einzelnen Komponenten bezeichnen [8].

Auch wenn Gaussian-Mixture-Modelle interpretierbar und probabilistisch sind, weisen diese beim Einsatz in realen Datenströmen Einschränkungen auf. Das größte Problem ist die Annahme stationärer Datenverteilungen. Treten Änderungen der Normaldaten im Datenstrom auf, kann dies dazu führen, dass ein einmal trainiertes Modell zunehmend schlechtere Dichteschätzungen liefert. Zudem erfordert ein GMM die Festlegung der Anzahl der Komponenten K vor dem Training. Zu viele Komponenten führen zu Überanpassung, während zu wenig Komponenten die Daten unzureichend modellieren. Die Anpassung der Komponenten ist in dynamischen Szenarien oft nicht ohne Weiteres möglich. Klassische GMMs reagieren ebenfalls empfindlich auf seltene oder neu auftretende Strukturen in den Daten [1].

2.2.2. Definition eines Gaussian-Mixture-Modells

Ein Gaussian-Mixture-Modell (GMM) ist ein probabilistisches Modell, das komplexe Datenverteilungen beschreibt. Die Wahrscheinlichkeitsdichte wird als gewichtete Mischung mehrerer multivariater Normalverteilungen dargestellt [2]. Im Gegensatz zu einer einzelnen Normalverteilung kann ein GMM mehrmodale Datenstrukturen modellieren. Formal wird die Dichtefunktion eines GMMs definiert als

$$p(x \mid \Theta) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x \mid \mu_k, \Sigma_k) \tag{2.3}$$

Wobei K die Anzahl der Komponenten bezeichnet. Die Gewichte π_k erfüllen die Bedingungen $\pi_k \geq 0$ sowie $\sum_{k=1}^{K} \pi_k = 1$. Jede Komponente besitzt einen Mittelwertvektor μ_k und eine Kovarianzmatrix Σ_k. Die Menge aller Modellparameter wird mit $\Theta = \{\pi_k, \mu_k, \Sigma_k\}_{k=1}^{K}$ zusammengefasst [8].

2.2.3. Training mit dem Expectation-Maximization (EM) Algorithmus

Ein Gaussian-Mixture-Modell wird in der Regel mithilfe des Expectation-Maximization-Algorithmus trainiert, der die Parameter schätzt. Der EM-Algorithmus ist ein iterativer Ansatz und dient zur Maximierung der Log-Likelihood von Modellen mit latenten Variablen [2, 8].

Latente Variablen entsprechen im Kontext von Gaussian-Mixture-Modellen unbekannten Zuordnungen der Datenpunkte zu einzelnen Komponenten. Der EM-Algorithmus maximiert nicht direkt die Log-Likelihood der beobachteten Daten, sondern passt die sogenannte erwartete vollständige Log-Likelihood iterativ an.

$$Q(\Theta, \Theta^{(t)}) = \mathbb{E}_{Z|X,\Theta^{(t)}} \left[\log p(X, Z \mid \Theta)\right] \tag{2.4}$$

wobei X die beobachteten Daten, Z die latenten Variablen und $\Theta^{(t)}$ die Parameter der vorherigen Iteration bezeichnen.

Im Expectation-Schritt (E-Step) wird der Erwartungswert der vollständigen Log-Likelihood unter Verwendung der aktuellen Parameter berechnet. Im Maximization-Schritt (M-Step) werden die Modellparameter anschließend so aktualisiert, dass diese Erwartung für die vorliegenden Datenpunkte maximiert. Durch den abwechselnden Ablauf von E- und M-Step konvergiert der Algorithmus gegen ein lokales Maximum der Log-Likelihood [8].

2.2.4. Anomalieerkennung mit GMMs

Gaussian-Mixture-Modelle werden häufig in der Anomalieerkennung eingesetzt, um das Normalverhalten eines Systems zu modellieren. Ein GMM wird dabei ausschließlich auf Daten trainiert, die zuvor als normal klassifiziert wurden. Anhand der Likelihood neuer Datenpunkte kann das gelernte Modell bewerten, ob diese anomal sind [1].

In der Praxis wird häufig die Log-Likelihood eines Datenpunktes als Maß für seine Übereinstimmung mit dem Normalmodell verwendet. Geringe Log-Likelihood-Werte, die durch unzureichende Abdeckung des Modells entstehen, werden erkannt, um Datenpunkte als potenziell anomal zu klassifizieren. Die Entscheidung, ob ein Datenpunkt anomal ist, erfolgt in der Regel durch feste Schwellwerte auf Basis der Log-Likelihood.

Die Wahl des Schwellwertes hat direkten Einfluss auf die vom Modell produzierten False Positives und False Negatives. Ein zu hoher Schwellwert führt zu vielen

falsch klassifizierten Normaldaten, während ein zu niedriger Schwellwert echte Anomalien übersieht.

Gaussian-Mixture-Modelle bieten sowohl Vorteile als auch Nachteile für die Anomalieerkennung. Durch ihre probabilistische Natur und die damit einhergehenden interpretierbaren Log-Likelihood-Werte ermöglicht dies eine nachvollziehbare Bewertung von Abweichungen. Klassische GMMs weisen gleichzeitig Einschränkungen auf, insbesondere in kontinuierlichen Datenströmen mit nicht-stationären Datenverteilungen. Änderungen, beispielsweise durch Concept Drifts, können dazu führen, dass ein statistisches Modell immer mehr ungeeignete Dichteschätzungen liefert.

Zudem ist die vorher bestimmte Anzahl der Komponenten nicht adaptiv und kann sich nicht veränderten Datenstrukturen anpassen [7].

2.3. Online-Lernen und nicht-stationäre Daten

Online-Lernen ist ein wichtiger Bestandteil der Anomalieerkennung in nicht-stationären Datenströmen. Beim Online-Lernen werden inkrementelle Updates an Modellen vorgenommen. Dabei ist der Speicher meist begrenzt und Updates müssen ohne manuelle Überprüfung der Datenqualität erfolgen. Im Gegensatz dazu erfolgt klassisches Training auf einem zuvor bestimmten Datensatz, der in einem einmaligen Training verarbeitet wird [7].

Nicht-stationäre Datenströme werden durch Veränderungen des statistischen Normalverhaltens im Zeitverlauf gekennzeichnet. Diese Veränderungen werden als Concept Drift beschrieben [1]. Concept Drift kann jederzeit in realen Szenarien auftreten und sich schrittweise entwickeln oder periodisch wiederkehren. Für eine Anwendung mit dem Ziel der Anomalieerkennung stellt dies eine besondere Herausforderung dar, da Änderungen im Normalverhalten nicht mit echten Anomalien verwechselt werden dürfen.

Probabilistische Modelle zur Dichteschätzung, wie beispielsweise Gaussian-Mixture-Modelle, reagieren sensibel auf Concept Drift. Einmalig trainierte Modelle können mit zunehmendem Drift immer schlechtere Approximationen der aktuellen Datenverteilung liefern. Dadurch kann die Anzahl falsch-positiver Erkennungen steigen. Umgekehrt kann eine zu schnelle Anpassung an neue Daten zu einem Normalmodell führen, das Anomalien zunehmend schlechter erkennt.

Die Gewichtung von Plastizität, also die Anpassungsfähigkeit an neue Daten, und der Stabilität bereits gelernter Strukturen wird als Stability-Plasticity-Dilemma bezeichnet [7]. Um Anomalieerkennung in nicht-stationären Datenströmen effektiv zu bewältigen, müssen Verfahren sowohl vergangenes Wissen bewahren als auch

flexibel auf neue Datenverteilungen reagieren. Reines Online-Lernen ist nicht automatisch robust gegenüber Concept Drift.

2.4. Ensemble-Methoden

Einzelne Modelle stoßen in komplexen und nicht-stationären Umgebungen an ihre Grenzen. Änderungen der Datenverteilungen, wie in Abschnitt 2.3 beschrieben, können dazu führen, dass ein einmalig trainiertes Modell zunehmend ungeeignete Approximationen liefert. Einzelne Modelle sind zudem häufig sensibel gegenüber Rauschen, Ausreißern oder fehlerhaften Annahmen über die bestehende Datenverteilung.

2.4.1. Motivation und Grundprinzip des Ensemble-Lernens

Durch das Kombinieren mehrerer Modelle zu einem sogenannten Ensemble, bei dem mehrere Basislerner gemeinsam Entscheidungen treffen, kann besser mit den Einschränkungen einzelner Modelle umgegangen werden. Die zentrale Idee besteht darin, dass unterschiedliche Modelle verschiedene Aspekte der Daten erfassen und sich ihre individuellen Schwächen gegenseitig ausgleichen. Dadurch lassen sich sowohl die Stabilität als auch die Generalisierungsfähigkeit eines Systems erhöhen [9].

Insbesondere in dynamischen Szenarien, wie kontinuierlichen Datenströmen, bieten Ensembles den Vorteil, dass einzelne Modelle auf unterschiedliche Zeiträume oder Teilbereiche der Daten spezialisiert werden können. Auf diese Weise lassen sich lang- und kurzfristige Muster parallel berücksichtigen, ohne dass das gesamte Modell bei veränderten Datenverteilungen neu trainiert werden muss.

Das Ensemble-Lernen beschreibt allgemein die Kombination mehrerer Basislerner zu einem gemeinsamen Vorhersagemodell. Die einzelnen Modelle können unabhängig oder teilweise abhängig voneinander trainiert werden, während die finale Entscheidung durch eine Aggregation der individuellen Modellausgaben erfolgt. Übliche Aggregationsstrategien umfassen beispielsweise Mittelwertbildungen von Scores, gewichtete Kombinationen oder Mehrheitsentscheidungen. Voraussetzung für den Erfolg eines Ensembles ist dabei eine ausreichende Diversität der Basislerner, da nur unkorrelierte Fehler zu einer Verbesserung der Gesamtleistung führen.

Klassische Ensemble-Methoden behandeln alle Basislerner gleich, während Boosting-Verfahren einen adaptiven Ansatz verfolgen. Dabei werden insbesondere schwer zu modellierende oder fehleranfällige Datenpunkte im Trainingsprozess gezielt

stärker gewichtet, um die Leistungsfähigkeit des Gesamtsystems iterativ zu verbessern [3, 9]. Im folgenden Abschnitt wird dieses Prinzip näher erläutert.

2.5. Grundidee des Boostings

Boosting bezeichnet eine spezielle Art von Ensemble-Verfahren, bei der mehrere Basislerner sequenziell trainiert und zu einem Modell beziehungsweise Ensemble kombiniert werden. Schwer zu modellierende Datenpunkte werden gezielt in einem adaptiven Boosting-Verfahren stärker berücksichtigt, während bei klassischen Ensembles alle Modelle gleich behandelt werden [3].

Die Grundidee besteht darin, dass neue Basislerner den Fokus auf Datenpunkte legen, die von vorherigen Modellen des Ensembles schlecht abgedeckt wurden. Dazu werden den Datenpunkten Gewichte zugewiesen, die iterativ während des Lernprozesses angepasst werden. Datenpunkte mit unzureichender Abdeckung oder hoher Modellunsicherheit erhalten ein höheres Gewicht als Datenpunkte, die bereits ausreichend abgedeckt sind. Damit können nachfolgende Modelle verstärkt auf den unzureichend abgedeckten Anteil der Datenverteilung trainiert werden.

Abstrakt lässt sich dieses Prinzip durch eine iterative Gewichtsanpassung der Trainingsdaten beschreiben:

$$w_i^{(m+1)} \propto w_i^{(m)} \cdot \exp\left(\ell_i^{(m)}\right), \tag{2.5}$$

wobei $w_i^{(m)}$ das Gewicht des Datenpunktes i in Iteration m bezeichnet und $\ell_i^{(m)}$ ein Maß für den Modellfehler oder die unzureichende Abbildung dieses Datenpunktes durch den aktuellen Basislerner darstellt. Das Proportionalitätszeichen $\propto$ veranschaulicht, dass die Gewichtsaktualisierung zunächst unnormiert erfolgt und anschließend über alle Datenpunkte hinweg normalisiert wird.

In dieser Arbeit entspricht $\ell_i^{(m)}$ einem indikatorbasierten Fehlermaß, das schwer modellierbare Datenpunkte kennzeichnet. Datenpunkte mit besonders niedriger Log-Likelihood, beispielsweise unterhalb eines festgelegten Quantils der durch den aktuellen Basislerner berechneten Log-Likelihood-Verteilung, werden als *schwer modellierbar* markiert. Anschließend erhalten diese Datenpunkte durch $\exp\left(\ell_i^{(m)}\right)$ ein erhöhtes Gewicht und werden dadurch in nachfolgenden Lernschritten verstärkt berücksichtigt.

Normalerweise entspricht $\ell_i^{(m)}$ einer Fehlklassifikation oder einer darauf basierenden Verlustfunktion in Boosting-Verfahren. $\ell_i^{(m)}$ kann jedoch auch kontinuierliche

Fehlermaße repräsentieren, zum Beispiel geringe Likelihood- oder Log-Likelihood-Werte in probabilistischen Modellen. Durch diese Abstraktion lässt sich dieses Boosting-Prinzip auch jenseits von überwachten Klassifikationsproblemen anwenden.

AdaBoost ist ein bekanntes Beispiel für einen Boosting-Algorithmus. Hierbei werden schwache Lerner zu einem starken Klassifikator kombiniert. Die Vorhersage wird aus einer gewichteten Kombination der einzelnen Basislerner gebildet, bei der bessere Modelle stärker zur Gesamtentscheidung beitragen [3]. Gerade bei komplexen Datenstrukturen, bei denen einzelne Modelle Schwierigkeiten haben, alle relevanten Muster abzubilden, eignet sich eine fokussierte Lernstrategie wie Boosting.

Boosting erfordert jedoch gleichzeitig eine sorgfältige Gestaltung, da eine zu starke Fokussierung auf fehlerhafte Datenpunkte zu Überanpassung oder Verstärkung von Rauschen führen kann. Gerade in realen, verrauschten Datenströmen ist eine ausgeglichene Gewichtung von zentraler Bedeutung.

Dieses abstrakte Boosting-Prinzip bildet die konzeptionelle Grundlage für das in dieser Arbeit entwickelte Boosted-Gaussian-Mixture-Ensemble.

2.6. Einordnung von GMM-Ensembles in der Anomalieerkennung

Probabilistische Dichteschätzungen, wie sie bei Gaussian-Mixture-Modellen eingesetzt werden, können Datenpunkte mit einer gewissen Wahrscheinlichkeit einzelnen Komponenten zuweisen. Diese sind jedoch an Trainingsdaten gebunden und nicht anpassungsfähig. Ensemble-Lernen kombiniert mehrere Modelle zu einer gewichteten Vorhersage, um Datenstrukturen besser abzubilden. Mit Boosting lassen sich schwierige Datenstrukturen der Normaldaten besser in ein Ensemble integrieren.

Mit einer Kombination aus probabilistischer Dichteschätzung, Ensemble-Lernen und adaptivem Boosting entsteht ein Ansatz, der sowohl Modellierungen komplexer Normalverteilungen als auch die kontinuierliche Anpassung an nicht-stationäre Datenströme ermöglicht. Diese Überlegungen sind die theoretische Grundlage für den in dieser Arbeit vorgestellten Ansatz eines Boosted-Gaussian-Mixture-Ensembles.

3. Boosted-Gaussian-Mixture-Ensemble

In diesem Kapitel wird die Funktionsweise des Boosted-Gaussian-Mixture-Ensembles (BGMEs) beschrieben. Die Evaluation der Resultate erfolgt in Kapitel 5.

3.1. Überblick und Notation

Das BGME dient der Modellierung komplexer Normalzustandsverteilungen in hochdimensionalen Daten. Gerade in multivariaten realen Umgebungen reichen statische Modelle ohne spezifisches Boosting häufig nicht aus. Anomalien werden als Datenpunkte mit geringer Log-Likelihood unter dem gelernten Normalmodell interpretiert. Sei $\mathbf{X} \in \mathbb{R}^{n \times d}$ ein Datensatz mit n Beobachtungen und d Dimensionen sowie $\mathbf{x} \in \mathbb{R}^d$ ein einzelner Datenpunkt. Das Ensemble besteht aus M Gaussian-Mixture-Modellen, deren Log-Likelihoods zur Bewertung neuer Datenpunkte kombiniert werden.

3.2. Konfigurationsmöglichkeiten des BGME

Ein BGME besteht aus jeweils zwei Konfigurationen. Eine dieser Konfigurationen bestimmt das Verhalten der Basislerner, die Kombination der Scores, das Boosting sowie das Modellmanagement. Die andere Konfiguration wird im Online-Streaming-Szenario verwendet und bestimmt Parameter wie Quantile, Refit-Windows und den finalen Scoring-Modus. Mit der `BGMEConfig` lässt sich die erstgenannte Konfiguration implementieren. Wobei *int* für eine ganze Zahl steht, *float* für eine reellwertige Zahl und *str* für eine beliebig lange Zeichenkette. In der Spalte `Optional` steht N für nicht optional und J für optional.

Einige dieser Parameter werden benötigt, um ein Gaussian-Mixture-Modell von sklearn [10] zu trainieren. Im Rahmen dieser Arbeit werden ausschließlich Gaussian-Mixture-Modelle von sklearn integriert.

Tabelle 3.1. BGME-Konfigurationsmöglichkeiten

Parameter	Format	Optional
`n_estimators`	int	N
`n_components`	int	N
`covariance_type`	str	N
`reg_covar`	float	N
`max_iter`	int	N
`tol`	float	N
`random_state`	str	J
`learning_rate`	float	N
`hard_quantile`	float	N
`min_weight`	float	N
`max_weight`	float	N
`combine`	str	N
`alpha_decay_rate`	float	J
`max_models_per_ensemble`	int	J

`n_estimators` steht für die Anzahl der sklearn Gaussian-Mixture-Modelle als Basislerner im Normal-Ensemble, welche direkt höher gewichtete Daten lernen und beim Übergang in den Online-Modus bereits bereitstehen.

`n_components` ist die Anzahl der einzelnen Komponenten eines Basislerners sowie der neuen Gaussian-Mixture-Modelle, die im späteren Verlauf in das Ensemble aufgenommen werden.

`covariance_type` gibt an, welche Art von Kovarianzparameter genutzt wird. ”full” gibt jeder Komponente eine eigene Kovarianzmatrix, ”tied” gibt allen Komponenten dieselbe Kovarianzmatrix, ”diag” gibt jeder Komponente eine eigene diagonale Kovarianzmatrix und ”spherical” gibt jeder Komponente eine einzelne Varianz.

`reg_covar` ist eine Regularisierung die auf der diagonalen Kovarianzmatrix hinzugefügt wird. Dadurch wird sichergestellt, dass alle Kovarianzmatrizen positiv definit sind.

`tol` gibt den Konvergenzschwellwert an, ab welchem Zeitpunkt der EM-Algorithmus terminiert.

`random_state` ist der benutzte Zufallsseed, welcher die initialen Modellparameter festlegt. Bei gleichen Parametern entstehen die gleichen Ergebnisse.

Alle bisherig genannten Parameter haben direkten Einfluss auf die Modellperformance und darauf, wie Modelle die Normalzustände abbilden, unabhängig von dem späteren Einfluss des erweiterten Ensembles. Alle Parameter werden beim Trainieren eines sklearn Gaussian-Mixture-Modells gemäß dieser Konfiguration so übergeben.

`learning_rate` ist ein zusätzlicher Skalierungsfaktor beim Gewichten der geboosteten Daten.

`hard_quantile` gibt an, wie viel Prozent der am schlechtesten abgebildeten Datenpunkte ein höheres Gewicht erhalten.

`min_weight` gibt das minimale Gewicht an, das einem Datenpunkt zugewiesen werden kann.

`max_weight` gibt das maximale Gewicht an, das einem Datenpunkt zugewiesen werden kann.

`combine` gibt an, nach welcher Regel die verschiedenen Scores des Ensembles zusammengerechnet werden. ”mean_loglik” nutzt den Mittelwert aller Log-Likelihoods und ”logmeanexp” nutzt die Wahrscheinlichkeiten, um die echte Misch-Dichte als Score zu verwenden.

`alpha_decay_rate` bestimmt, falls angegeben, mit welchem Faktor Modelle im Zeitverlauf an Gewicht verlieren, bis sie letztendlich ersetzt werden. Der Wert muss zwischen 0 und 1 liegen. Je geringer der Wert, desto schneller werden Modelle vergessen. Hierbei wird lediglich nach Zeit der Faktor einberechnet und nicht nach der tatsächlichen Performance einzelner Modelle. Somit wird jedes Modell nach einer gewissen Zeit vergessen, unabhängig von der Modellperformance.

`max_models_per_ensemble` gibt die maximale Anzahl an Modellen pro Ensemble an. Falls zusätzlich `alpha_decay_rate` einen Wert hat, werden Modelle nach Erreichen der Maximalgrenze mit dem geringsten Alpha ersetzt. Ohne `alpha_decay_rate`-Wert werden Modelle nicht ersetzt und erreichen einen Zeitpunkt, nach dem keine Updates mehr stattfinden werden. Ohne `max_models_per_ensemble`-Wert gibt es keine maximale Anzahl an Modellen.

Mit diesen Parametern lässt sich ein BGME initialisieren und anschließend auf Normalzustände trainieren. Im Folgenden wird beschrieben, wie dieser Prozess erfolgt.

3.3. Offline BGME: Training

Im Offline-Training wird das BGME ausschließlich auf Daten des Normalzustands trainiert. Damit soll eine möglichst präzise Modellierung der zugrunde liegenden Normalverteilung erreicht werden, bevor das Modell im Anschluss im Online-Streaming eingesetzt wird. Die Trainingsdaten bestehen aus n Datenpunkten, wobei jedem Datenpunkt ein Gewicht w_i zugewiesen wird, mit $w_i^{(1)} = \frac{1}{n}$. Das erste Modell entspricht der klassischen Maximum-Likelihood-Schätzung.

Die Anzahl der zu trainierenden Basislerner im Offline-Modus ist durch `n_estimators` festgelegt. Jeder Basislerner wird unabhängig initialisiert, wobei der Zufallsseed `random_state` die Reproduzierbarkeit des Trainings sicherstellt.

3.3.1. Training der Basislerner

Jeder Basislerner ist ein Gaussian-Mixture-Modell mit K Komponenten, dessen Parameter in der m-ten Boosting-Iteration geschätzt werden. Die Modellanpassung erfolgt unter expliziter Berücksichtigung der aus der vorherigen Boosting-Iteration resultierenden Datengewichte $w_i^{(m)}$.

Das Training erfolgt mittels Expectation-Maximization-(EM)-Algorithmus, wobei die gewichtete Log-Likelihood $\sum_{i=1}^{n} w_i^{(m)} \log p(\mathbf{x}_i \mid \Theta^{(m)})$ maximiert wird. Datenpunkte, die von vorherigen Modellen nur unzureichend beschrieben wurden und daher ein erhöhtes Gewicht aufweisen, beeinflussen die Parameterschätzung entsprechend stärker. Dadurch fokussiert sich jeder neue Basislerner zunehmend auf die vom vorherigen Basislerner als schwer zu modellierend eingestuften Datenpunkte. Auf diese Weise entsteht eine feinere Approximation der Normalzustandsverteilung.

3.3.2. Boosting-Mechanismus und Gewichtsanpassung

Nach dem Training des m-ten Basislerners wird für jeden Datenpunkt die Log-Likelihood gemäß der Formel 2.2 berechnet.

Um das Boosting zu realisieren, wird ein festes Quantil τ bestimmt, das durch den Parameter `hard_quantile` definiert wird. Dieses Quantil identifiziert den Anteil der Datenpunkte, die vom aktuellen Modell am schlechtesten beschrieben sind. Nur diese Datenpunkte erhalten im nächsten Trainingsschritt ein höheres Gewicht.

Die Gewichtsanpassung erfolgt gemäß der Formel 2.5, wobei die Lernrate η innerhalb $\exp\left(\ell_i^{(m)}\right)$ zusätzlich ergänzt wird und dementsprechend $\exp\left(\eta \cdot \ell_i^{(m)}\right)$ ergibt. Anschließend werden die Gewichte auf den Wertebereich zwischen `min_weight` und `max_weight` beschränkt.

3.3.3. Iterativer Ensemble-Aufbau

Die beschriebenen Schritte werden iterativ für alle Basislerner durchgeführt. Jeder neue Basislerner wird dabei auf einer veränderten Gewichtung der Trainingsdaten trainiert und fokussiert sich zunehmend auf Datenbereiche, die von vorherigen Modellen nur unzureichend abgebildet wurden.

Durch diesen sequentiellen Aufbau entsteht ein Ensemble aus komplementären Gaussian-Mixture-Modellen, die gemeinsam eine komplexe Normalzustandsverteilung approximieren. Der Boosting-Ansatz verhindert dabei, dass einzelne Modelle dominante Strukturen übermäßig stark anpassen, während seltene oder schwierige Muster ignoriert werden.

3.3.4. Ergebnis des Offline-Trainings

Das Ergebnis des Offline-Trainings ist ein vollständig initialisiertes Ensemble bestehend aus M Gaussian-Mixture-Modellen inklusive ihrer zugehörigen Parameter und internen Gewichtungen. Dieses Ensemble dient als Ausgangspunkt für die Online-Phase, in der neue Datenpunkte bewertet, Anomaliescores berechnet und bei Bedarf zusätzliche Modelle ergänzt oder bestehende ersetzt werden.

3.4. Ensemble Scoring

Nach dem Offline-Training steht ein Ensemble aus M Gaussian-Mixture-Modellen zur Verfügung, welches zur Bewertung neuer Datenpunkte eingesetzt wird. Ziel des Ensemble-Scorings ist es, aus den individuellen Modellbewertungen einen einzelnen, konsistenten Anomaliescore zu berechnen, der die Unsicherheit und Diversität der einzelnen Basislerner berücksichtigt.

Jedes Modell liefert eine Log-Likelihood, welche angibt, wie gut der Datenpunkt durch das jeweilige Normalmodell beschrieben wird. Diese individuellen Scores bilden die Grundlage der Ensemble-Aggregation.

3.4.1. Ungewichtete Score-Aggregation

Die einfachste Form der Aggregation erfolgt über den arithmetischen Mittelwert der Log-Likelihoods aller Basislerner. Dieser Aggregationsmodus wird durch die Konfiguration `combine = ''mean_loglik''` realisiert und ergibt sich zu

$$s(\mathbf{x}) = \frac{1}{M} \sum_{m=1}^{M} \ell^{(m)}(\mathbf{x}). \tag{3.1}$$

Dieser Ansatz gewichtet alle Basislerner gleich und erzeugt einen robusten Anomaliescore. Einzelne fehlerhafte Modellschätzungen wirken sich dadurch weniger stark auf das Gesamtergebnis aus.

3.4.2. Wahrscheinlichkeitsbasierte Aggregation

Alternativ kann das Ensemble als explizit formulierte Mischung der einzelnen Basislerner interpretiert werden. In diesem Fall wird die Log-Likelihood der gemischten Dichte berechnet, die durch die Konfiguration `combine = ''logmeanexp''` umgesetzt wird. Der Ensemble-Score ergibt sich zu

$$s(\mathbf{x}) = \log\left(\sum_{m=1}^{M} \alpha_m \exp\left(\ell^{(m)}(\mathbf{x})\right)\right), \tag{3.2}$$

wobei α_m die normierten Ensemblegewichte der einzelnen Modelle bezeichnet.

Diese Form der Aggregation entspricht der Log-Likelihood einer Mischdichte [8] und besitzt eine klare probabilistische Interpretation. Modelle mit höherer Wahrscheinlichkeit für den gegebenen Datenpunkt tragen stärker zum Gesamtscore bei, während schlecht passende Modelle automatisch abgeschwächt werden.

3.4.3. Modellgewichtung und zeitlicher Verfall

Sofern der Parameter `alpha_decay_rate` gesetzt ist, unterliegen die Ensemblegewichte α_m einem zeitlichen Verfall. Dadurch wird der Einfluss älterer Modelle sukzessive reduziert, um das Ensemble an veränderliche Datenverteilungen anzupassen. Der Gewichtungsfaktor eines Modells nimmt dabei exponentiell ab und wird bei der Score-Aggregation berücksichtigt.

Dieser Mechanismus ermöglicht eine kontrollierte Form des Vergessens, ohne dass Modelle abrupt entfernt werden müssen. Insbesondere im Online-Streaming-Szenario trägt dies zur Stabilität der Architektur bei, etwa bei Datenströmen, die Concept Drift beinhalten.

3.4.4. Interpretation des Ensemble-Scores

Der resultierende Ensemble-Score $s(\mathbf{x})$ beschreibt die Übereinstimmung des Datenpunktes mit dem gelernten Normalzustandsmodell. Niedrige Werte deuten auf eine geringe Wahrscheinlichkeit gemäß dem Ensemble und werden als potenzielle Anomalien interpretiert.

Der Ensemble-Score selbst stellt noch keine binäre Anomalieentscheidung dar, sondern bildet die Grundlage für nachgelagerte Entscheidungsmechanismen, etwa quantilbasierte Schwellwerte oder adaptive Thresholds im Online-Modus.

3.5. Konfigurationsmöglichkeiten des Online-BGME

Die hier aufgeführten Parameter dienen zusätzlich zu den Parametern aus Tabelle 3.1 zur Konfiguration der Online-Komponente des BGME.

Tabelle 3.2. Konfigurationsmöglichkeiten der BGME-Online-Komponente

Parameter	**Format**	**Optional**
`ll_threshold`	float	J
`threshold_quantile`	float	N
`refit_window_size`	int	N
`include_correct_predictions`	bool	N
`min_samples_for_refit`	int	N
`scoring_mode`	str	N
`separation_threshold`	float	N

wobei *float* für reellwertige Zahlen, *int* für ganze Zahlen, *str* für beliebig lange Zeichenketten und *bool* für einen binären Wert steht. In der Spalte *Optional* steht *J* für optional und *N* für nicht optional.

`ll_threshold` gibt den Schwellwert der Log-Likelihood an, der verwendet werden soll. Falls dieser nicht angegeben ist, wird der Schwellwert automatisch beim ersten Training ermittelt und gesetzt.

`threshold_quantile` gibt das Quantil der Schwellenwerte an, die als anomal gelten.

`refit_window_size` ist die Anzahl der beobachteten Datenpunkte, bis der nächste periodische Refit stattfindet.

`include_correct_predictions` bestimmt, ob das Modell nur aus Fehlklassifikationen mit einem Wert *False* lernt, oder mit dem Wert *True* alle gesehenen Datenpunkte in einen Refit mit aufnimmt.

`min_samples_for_refit` bestimmt die Mindestanzahl an Datenpunkten, die benötigt werden, um ein neues Modell in ein Ensemble zu integrieren. Speziell bei `include_correct_predictions = False` können häufig weniger Datenpunkte als dieser Schwellwert auftreten, was dazu führt, dass ein BGME, welches aus allen Datenpunkten lernt, früher die maximale Modellanzahl erreicht.

`scoring_mode` bietet die Konfiguration des Scoring-Modus im Online-Betrieb, `''normal_only''` ignoriert das Anomalie-Ensemble für die Entscheidungen und `''separation''` vergleicht die Log-Likelihoods beider Ensembles.

`separation_threshold` ermöglicht einen zusätzlichen Bias für eines der beiden Ensembles mit $(ll_{anomaly} - ll_{normal}) > separation\_threshold$. Ist diese Ungleichung erfüllt, wird eine Anomalie vorhergesagt.

3.6. Online-BGME: Streaming und Auto-Refit

Mit den zuvor beschriebenen Konfigurationen lässt sich ein BGME in einen zuvor ungesehenen Online-Datenstrom integrieren. Jeder Datenpunkt wird einzeln verarbeitet, bewertet und optional zur Anpassung des Ensembles herangezogen. Ziel ist es, Anomalien frühzeitig zu erkennen und gleichzeitig auf sich ändernde Datenverteilungen reagieren zu können, ohne das Modell vollständig neu zu trainieren.

3.6.1. Sequentielle Verarbeitung von Datenpunkten

Für jeden Datenpunkt wird zunächst der Ensemble-Score $s(\mathbf{x}_t)$ gemäß dem in Gleichung 3.1 beziehungsweise 3.2 beschriebenen Aggregationsmechanismus berechnet. Der resultierende Score beschreibt, ob ein Datenpunkt mit dem aktuell gelernten Normalzustand vergleichbar ist, und bildet die Grundlage für alle nachfolgenden Entscheidungen im Online-Modus.

3.6.2. Anomalieentscheidung im Online-Modus

Die Entscheidung, ob ein Datenpunkt als anomal eingestuft wird, erfolgt anhand eines Schwellwerts auf dem Ensemble-Score. Dieser Schwellwert kann entweder explizit über den Parameter `ll_threshold` oder implizit über das Quantil `threshold_quantile` bestimmt werden.

Im quantilbasierten Fall wird der Schwellwert adaptiv aus der Score-Verteilung der bislang beobachteten Datenpunkte berechnet. Ein Datenpunkt gilt als anomal, wenn $s(\mathbf{x}_t) < \theta$, wobei θ den aktuellen Schwellenwert bezeichnet. Dieser Ansatz ermöglicht eine robuste Anpassung an unterschiedliche Skalen der Log-Likelihoods und ist insbesondere in hochdimensionalen Szenarien vorteilhaft.

3.6.3. Scoring-Modi im Online-Betrieb

Der Parameter `scoring_mode` steuert, wie die Anomalieentscheidung im Online-Betrieb erfolgt. Im Modus `normal_only` basiert die Entscheidung ausschließlich auf dem Normal-Ensemble. Alternativ kann der Modus `separation` verwendet werden, bei dem ein zusätzliches Anomalie-Ensemble aufgebaut wird. In diesem Fall werden die Log-Likelihoods beider Ensembles verglichen, und eine Anomalie wird erkannt, wenn die Ungleichung aus 3.5 erfüllt ist. Dieser Ansatz erlaubt eine explizite Trennung zwischen Normal- und Anomaliezuständen und erhöht die Robustheit in komplexen Grenzbereichen zwischen Normal- und Anomaliezuständen.

3.6.4. Pufferung und Refit-Trigger

Unabhängig von der finalen Entscheidung werden ausgewählte Datenpunkte in einem Refit-Puffer gesammelt. Die Auswahl erfolgt gemäß dem Parameter `include_correct_predictions`. Ist dieser deaktiviert, werden ausschließlich Fehlklassifikationen in den Puffer aufgenommen, andernfalls fließen alle beobachteten Datenpunkte in den Puffer ein.

Falls die Anzahl der gepufferten Datenpunkte den Wert `min_samples_for_refit` erreicht und zusätzlich das Refit-Fenster `refit_window_size` erfüllt ist, wird ein Refit initiiert.

3.6.5. Auto-Refit und Ensemble-Erweiterung

Beim Auto-Refit wird auf Basis der gepufferten Daten ein neues Gaussian-Mixture-Modell trainiert. Dieses Modell repräsentiert eine aktuelle Approximation der Datenverteilung und wird anschließend in das Ensemble integriert.

Abhängig von den Parametern `max_models_per_ensemble` und `alpha_decay_rate` kann dabei entweder das Ensemble erweitert oder ein bestehendes Modell mit geringem Gewicht ersetzt werden. Dadurch bleibt die Ensemblegröße kontrolliert, während gleichzeitig eine kontinuierliche Anpassung an neue Datenstrukturen ermöglicht wird.

3.6.6. Stabilität und Adaptivität im Streaming

Die Kombination aus quantilbasierter Schwellenwertbildung, kontrollierter Modellgewichtung und periodischem Auto-Refit erlaubt es dem BGME, sowohl stabile Normalzustände als auch schleichende Veränderungen der Datenverteilung zuverlässig abzubilden.

Insbesondere bei nicht-stationären Datenströmen mit Concept Drift stellt dieser Mechanismus sicher, dass das Ensemble nicht veraltet, ohne durch kurzfristige Ausreißer destabilisiert zu werden.

3.7. Ensemble-Management

Das Ensemble-Management stellt eine eigenständige Meta-Ebene des BGME dar. Dabei ist es klar von Modelltraining, Score-Aggregation und Online-Entscheidungen getrennt. Während Offline- und Online-Training die Parameterschätzung einzelner Gaussian-Mixture-Modelle übernehmen und die Score-Aggregation deren

Bewertungen zu einem Ensemble-Score kombiniert, kontrolliert das Ensemble-Management die Zusammensetzung der Modellpopulation. Dazu zählen insbesondere die Begrenzung der Ensemblegröße, die Gewichtung einzelner Modelle sowie deren gezielte Ergänzung oder Ersetzung im Streaming-Betrieb.

Auf diese Weise adressiert das Ensemble-Management den Zielkonflikt zwischen Stabilität und Adaptivität im Online-Betrieb. Im Gegensatz zu naivem Online-Retraining erfolgt die Anpassung des BGME inkrementell und kontrolliert. Modellwechsel erfolgen nicht abrupt, sondern über Gewichtung und selektives Ersetzen.

3.8. Visualisierung eines BGME

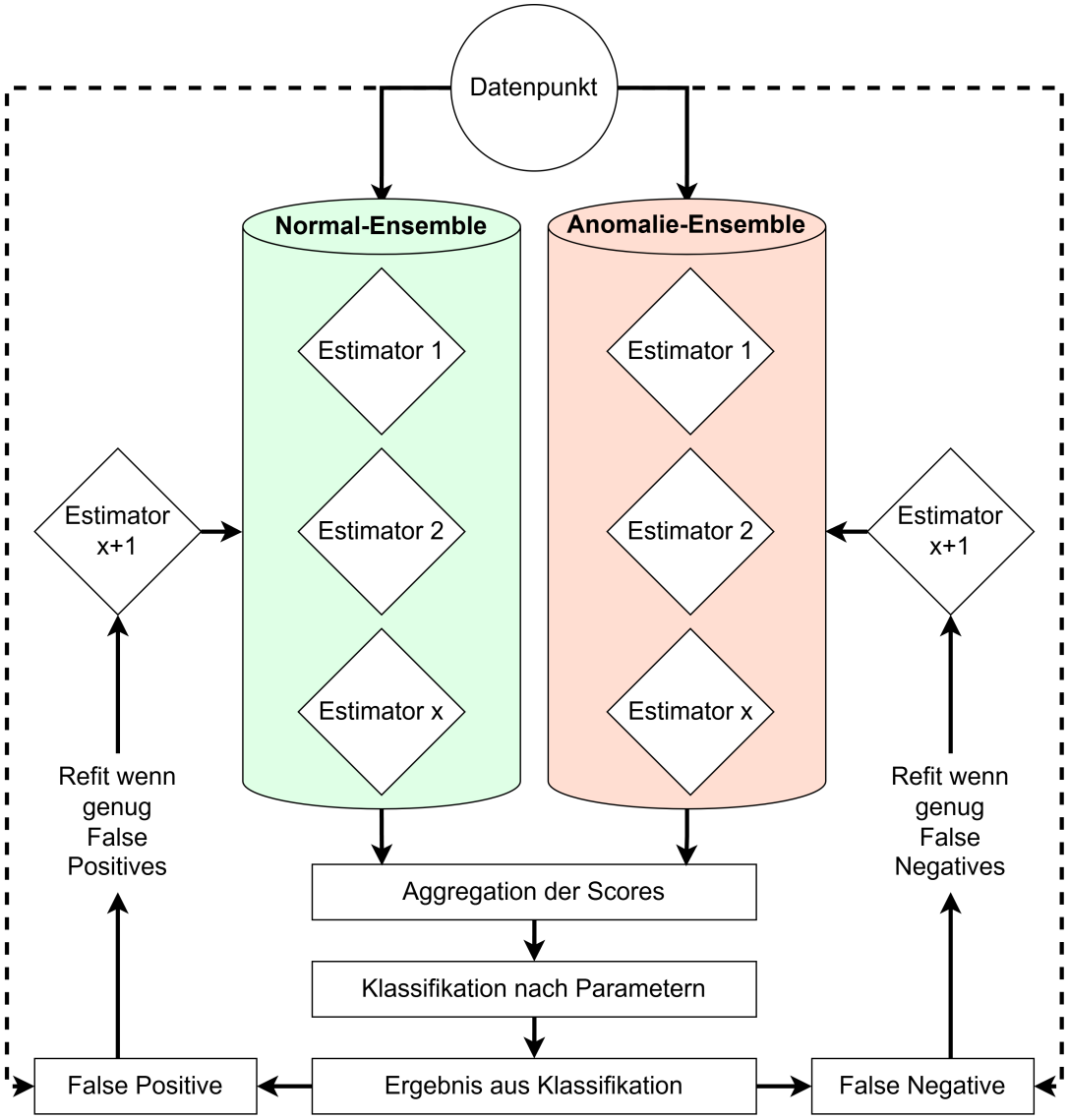

Abbildung 3.1. Visualisierung eines BGME

Jeder Datenpunkt wird rückblickend mit der Klassifikation des Ensembles verglichen, um die False Positives und False Negatives zu extrahieren. Aus diesen werden neue Modelle, hier `Estimator` genannt, in den jeweiligen Ensembles mit $x+1$ veranschaulicht und integriert. Das rote Ensemble ist bei der Aufnahme in den Online-Betrieb vorerst ohne `Estimator`, während das grüne Ensemble mit den Basislernern startet. Neue Modelle werden von beiden Ensembles im Verlauf integriert. Hierbei wird ein BGME abgebildet, das nur aus Fehlern lernt.

4. Methodik und Datenbasis

4.1. Methodischer Überblick

Die Evaluation des in dieser Arbeit entwickelten Boosted-Gaussian-Mixture-Ensembles (BGME) erfolgt in einem kontinuierlichen Datenstrom. Dabei wird zwischen drei verschiedenen Modellen unterschieden. Als Baseline wird ein einzelnes Gaussian-Mixture-Modell mit erhöhter Anzahl an Komponenten im Vergleich zu den anderen beiden Modellen verwendet [10]. Diese Baseline wird je zwei Boosted-Gaussian-Mixture-Ensembles gegenübergestellt. Eines dieser Modelle erhält rückwirkend Informationen über gemachte Fehler bei Falschklassifikationen, während das andere mit allen Datenpunkten unabhängig von der Klassifikation arbeitet. Dabei gelten nahezu identische Rahmenbedingungen hinsichtlich der Konfigurationsparameter der Modelle.

In der Evaluation werden sowohl reale als auch synthetisch erzeugte Datenströme analysiert. Während synthetische Daten eine gezielte Kontrolle über Anomalietypen und Verteilungsänderungen ermöglichen, zeigt ein realer Datensatz die praktische Anwendbarkeit des Ansatzes.

4.2. Annahmen

Damit dieser Ansatz funktioniert, müssen einige Vorbedingungen gegeben sein. Das Normalverhalten der Daten sollte eine stabile statistische Struktur aufweisen und ausschließlich aus Normaldaten ohne Anomalien bestehen. Anomalien treten erst im Online-Betrieb auf und sind nicht überrepräsentiert. Dabei können sowohl einzelne Anomalieausreißer als auch kollektive Anomalietypen auftreten. Die Label, also die Markierung der Datenpunkte, die als normal beziehungsweise anomal klassifiziert sind, müssen auch rückwirkend im Online-Betrieb verfügbar sein, spätestens wenn ein Refit stattfindet. Die Boosted-Gaussian-Mixture-Ensembles erhalten in dieser Arbeit beim Vorhersagen jedes Datenpunktes direkt das jeweilige Label, verwenden jenes aber erst bei erlaubten Refits. Dadurch haben die Label keinen Einfluss auf die Vorhersage der Modelle.

4.3. Realer Datensatz - ServerMachineDataset

Für die Hauptevaluation wird ein realer Datensatz herangezogen. Das ServerMachineDataset (SMD) [4] ist ein multivariater, hochdimensionaler Datenstrom, der

in einem Zeitraum von fünf Wochen auf realen Servern erhoben wurde. In diesem Datensatz wurden Server beziehungsweise Maschinen überwacht. Dabei werden drei unterschiedliche Gruppierungen von Maschinen betrachtet, die jeweils in mehrere unabhängige Sequenzen unterteilt sind. Durch die Unterteilung verschiedener Maschinen weisen die einzelnen Zeitreihen in etwa eine Länge von 27.500 Messpunkten auf. Jede Sequenz besteht aus einer festen Anzahl von Merkmalen, die kontinuierlich in diskreten Zeitintervallen aufgezeichnet werden. Insgesamt entstehen dadurch 28 unterschiedliche Zeitreihen. Jede dieser Zeitreihen umfasst eine Trainings- und eine Testzeitreihe, die in etwa gleich lang sind.

Anomalien sind zeitlich annotiert und treten vergleichsweise selten auf. In den unterschiedlichen Sequenzen gibt es einen Anomalieanteil von etwa 0.01 % bis 16.00 %. Auftretende Anomalien manifestieren sich häufig nicht als Ausreißer, sondern als zusammenhängende Muster über mehrere aufeinanderfolgende Zeitpunkte. Die Annotationen sind teilweise gröber gewählt und nicht unbedingt für jeden Datenpunkt repräsentativ. Durch diese Anomalietypen und die vorhandenen Label-Eigenschaften eignet sich dieser Datensatz insbesondere für die Untersuchung von Anomalien, wie sie in der Praxis häufig in kontinuierlichen Datenströmen auftreten.

Der SMD stellt ein realitätsnahes Evaluationsszenario dar, da die Daten sowohl Rauschen als auch nicht-stationäres Verhalten aufweisen. Zudem sind Annotationen gegeben, die eine quantitative Bewertung der Anomalieerkennung ermöglichen. Durch die zeitlich ausgedehnten Anomalien sowie sich verändernde Normalzustände stellt dieser Datensatz hohe Anforderungen an die eingesetzten Modelle. Gerade bei Modellen, die ausschließlich auf einem statistischen Normalmodell basieren, stoßen diese Ansätze schnell an ihre Grenzen.

4.4. Synthetischer Datensimulator

4.4.1. Zielsetzung und Motivation

Der SMD Datensatz ist hochdimensional und anspruchsvoll. Dies liegt zum Teil an den kürzeren Datenströmen. Um längere Zeitreihen evaluieren zu können, wird ein synthetischer Datengenerator benötigt. Dabei liegt der Fokus stärker auf der Veranschaulichung des Grundprinzips von Boosted-Gaussian-Mixture-Ensembles und multivariaten Zeitreihen und weniger auf dem Heranziehen dieses Datensatzes als Hauptevaluationsgrundlage für die Modellperformance. Der Datensimulator soll zudem ermöglichen, längere Zeitreihen zu imitieren, um eine simple Langzeitanalyse der Architektur durchzuführen und einen Vergleich mit dem Baseline-GMM zu erlauben.

4.4.2. Beispielhafte Generierung

Mit den Konfigurationen aus A.3.1 und A.3.2 im *Anhang* A.3 erstellt der Datensimulator folgende Datensimulation:

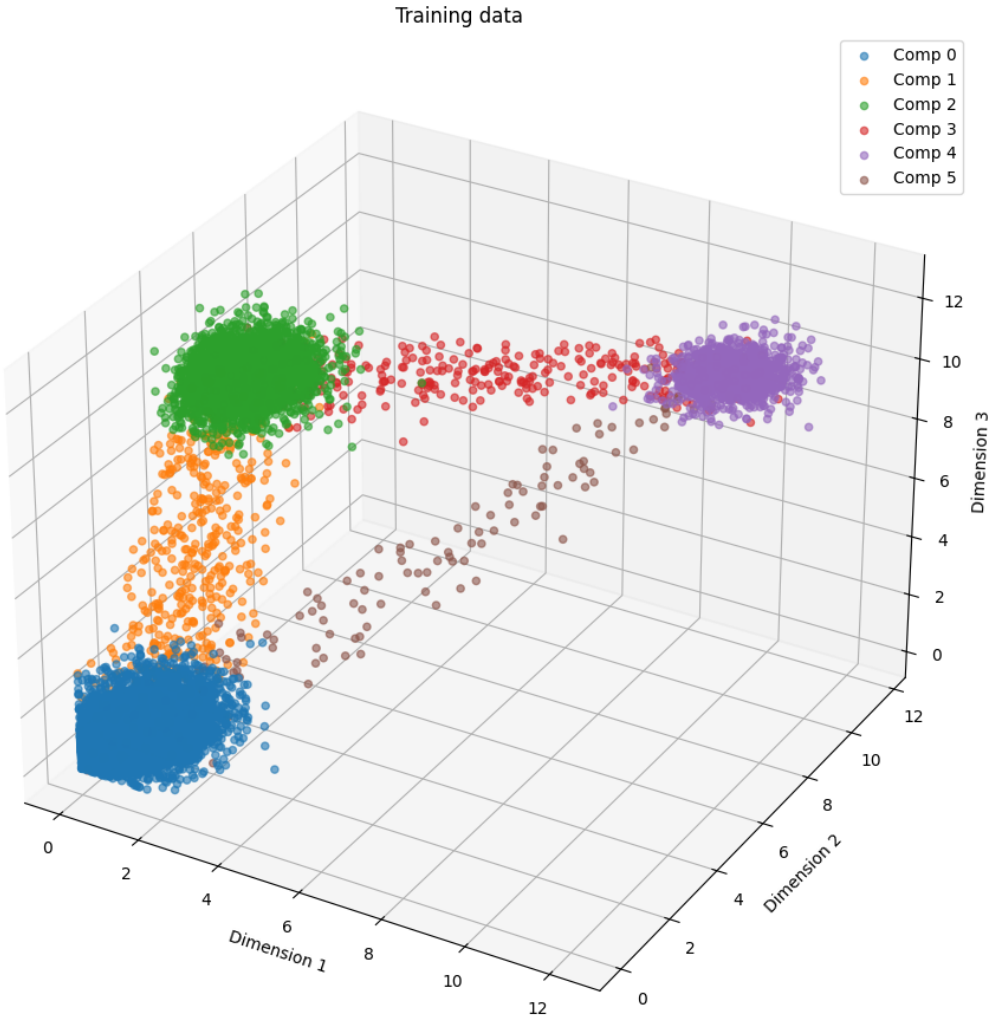

Abbildung 4.1. Beispiel von Normalzuständen im Datensimulator

Dabei steht `Training data` für die Normalzustandsdaten und *Comp* x für die verschiedenen Modi. *Comp* 0, *Comp* 2 und *Comp* 4 sind Betriebsmodi, während *Comp* 1, *Comp* 3 und *Comp* 5 Übergangsmodi sind.

Die generierten Daten im Online Betrieb mit identischer Konfiguration sind wie folgt:

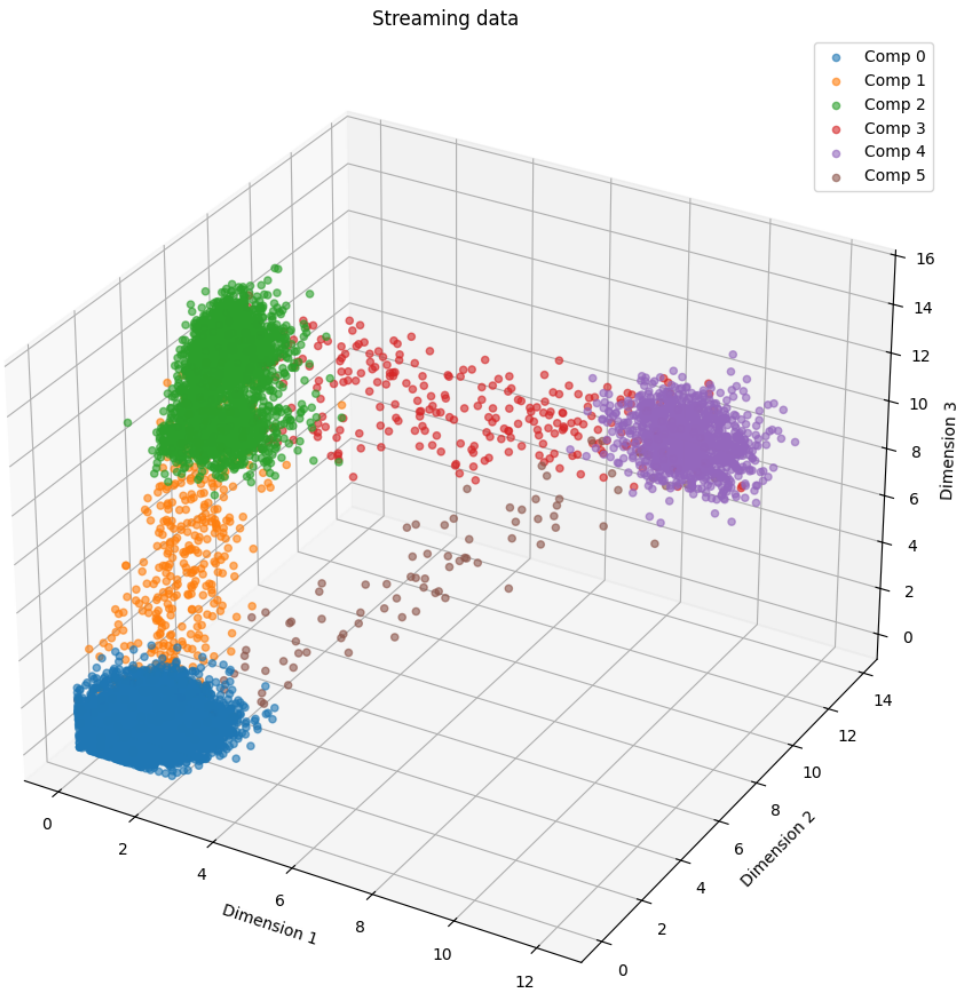

Abbildung 4.2. Beispiel eines anomalieinjizierten Datenstroms im Datensimulator

Dabei steht `Streaming data` für die kontinuierlichen Datenströme, die nicht im Modelltraining mit eingeschlossen sind.

Bei genauerer Betrachtung beider Grafiken werden deutliche Unterschiede bei *Comp* 2, *Comp* 3 und *Comp* 4 sichtbar, während die restlichen Modi relativ ähnlich zu den Normalzuständen sind. Bei *Comp* 2 trat mindestens eine Mittelwertverschiebung und bei *Comp* 4 mindestens eine Varianzskalierung auf. Der Übergangsmodus *Comp* 3 dazwischen ist ebenfalls von den Auswirkungen der Anomaliemuster betroffen.

4.4.3. Konfigurationsmöglichkeiten

Der Datensimulator bietet verschiedene Konfigurationsmöglichkeiten für die Generierung der Daten sowie für die Anomalieinjektion im Online-Betrieb. Die folgende Tabelle zeigt die Parameternamen und deren Format sowie eine anschließende Erklärung ihrer Funktionen bei der Generierung von Normaldaten.

Tabelle 4.1. Konfigurationsmöglichkeiten des Datensimulators

Parameter	Format	Optional
`num_modes`	int	N
`num_dimensions`	int	N
`random_seed`	int	N
`means_list`	list[tuple[float, float]]	B
`vars_list`	list[tuple[float, float]]	B
`mean_range`	tuple[float, float]	B
`var_range`	tuple[float, float]	B
`mode_weights`	list[float]	J
`samples_per_iteration`	int	N
`iterations_during_training`	int	N
`iterations_for_generator`	int	N
`noise_student_t_df`	float	J
`noise_student_t_scale`	float	B
`quant_step`	float	J
`clip`	tuple[float, float]	J

In der Format-Spalte steht *tuple[ ]* für Paare der inneren Formate und *list[ ]* für mehrere Werte dieser Inhalte. *int* steht für eine ganze Zahl, während *float* eine reellwertige Zahl darstellt. In der Spalte Optional steht *N* für nicht optional, *J* für optional und *B* für unter bestimmten Voraussetzungen optional.

`num_modes` steht für die Anzahl der generierten normalen Betriebsmodi. Diese Zahl wird aufgrund der implementierten Übergangsmodi, die nicht explizit angegeben werden müssen, verdoppelt.

`num_dimensions` gibt an, wie viele Dimensionen ein Datenpunkt pro Zeitpunkt hat.

`random_seed` ist der Seed, der für Zufallsgenerierungen genutzt wird und somit reproduzierbare Ergebnisse ermöglicht.

`means_list` und `mean_range` geben die Mittelwerte der generierten Datenverteilungen an. Nur eines der beiden muss angegeben sein. Bei `means_list` müssen exakt die Mittelwerte für jede Kombination aus `num_modes` und `num_dimensions` angegeben werden. Alternativ generiert `mean_range` zufällige Mittelwerte im Rahmen der `num_modes` und `num_dimensions` innerhalb des angegebenen Wertebereichs.

`vars_list` und `var_range` geben die jeweiligen Varianzen der angegebenen Mittelwerte an. Ähnlich wie bei den Mittelwerten muss nur eines von beiden angegeben sein, je nachdem, ob vorgegebene oder zufällige Verteilungen generiert werden sollen. Die Regeln für `vars_list` sind identisch mit denen von `means_list`, ebenso gelten für `vars_range` die gleichen Regeln wie für `mean_range`.

`mode_weights` gibt die prozentuale Gewichtung der jeweiligen Modi an. Hierbei müssen exakt `num_modes` $\cdot 2$ für normale Betriebs- und Übergangsmodi angegeben

werden. Die ungeraden Stellen dieser Aufzählung repräsentieren die Gewichtung der Betriebsmodi und die geraden Stellen der Übergangsmodi. Diese müssen als reellwertige Zahlen angegeben werden und summieren sich auf 1. Ist `mode_weights` nicht angegeben, so werden gleichverteilte Gewichtungen an alle Modi verteilt. Bei 10 Betriebsmodi hat beispielsweise jeder Betriebs- und Übergangsmodus einen Anteil von 0.05 oder 5 %.

`samples_per_iteration` gibt die Anzahl der Datenpunkte pro Iteration an.

`iterations_during_training` gibt die Anzahl der Iterationen mit jeweils `samples_per_iteration` Datenpunkten für die Generierung der Normalzustände an.

`iterations_for_generator` gibt die Anzahl der Iterationen mit je `samples_per_iteration` Datenpunkten des Online-Betriebs an. Dies entspricht den Testdaten mit Normalzuständen und Anomalieinjektionen.

`noise_student_t_df` erzeugt eine Student-t-Verrauschung der Daten um realitätsnähere Bedingungen zu schaffen. Hohe reellwertige Zahlen erzeugen stärkeres, jedoch weniger extremes Rauschen, während geringere Werte selteneres, dafür intensiveres Rauschen erzeugen. Die Angabe ist optional. Wird nichts angegeben, findet keine Verrauschung statt.

`noise_student_t_scale` ergibt einen zusätzlichen Parameter für die Steuerung der Student-t-Verrauschung. Geringe reellwertige Zahlen erhalten ähnliches Verhalten der Verrauschung, während hohe mehr Rauschen erzeugen. Die Angabe ist bedingt optional, wird `noise_student_t_df` angegeben, so muss `noise_student_t_scale` auch angegeben sein.

`quant_step` gibt an, wie viele Nachkommastellen Werte haben können. Die Angabe erfolgt mit der kleinstmöglichen Rundungsanpassung. Bei einer Angabe von 0.01 werden Werte auf zwei Nachkommastellen gerundet, um ebenfalls realitätsnahe Bedingungen zu schaffen, falls Sensoren beispielsweise nur gewisse Genauigkeiten haben. Bei keiner Angabe haben Werte die vollständige numerische Genauigkeit der Generierung.

`clip` besteht aus einem Paar ganzer Zahlen, welches die Werte auf einen bestimmten Wertebereich begrenzt. Der erste Wert steht für den kleinstmöglichen Wert und der zweite für den größtmöglichen. Bei *[0, 20]* werden Werte unter 0 auf 0 gesetzt und dementsprechend Werte über 20 auf 20. Die Angabe ist optional, ohne Angabe werden Werte nicht eingegrenzt.

Damit hat der Datensimulator alle nötigen Informationen gegeben, um Normaldaten sowohl im Normalzustand als auch im Online-Betrieb zu generieren. Die Anomalieinjektionen im Online-Betrieb lassen sich ebenfalls konfigurieren. In der

folgenden Tabelle werden alle Konfigurationsmöglichkeiten der Anomalieinjektion aufgezählt und erklärt.

Tabelle 4.2. Konfigurationsmöglichkeiten der Anomalieinjektion

Parameter	**Format**
`modes`	list[str]
`stream_anomaly_prob`	float
`stream_use_seen_template_prob`	float
`max_stream_anomaly_template`	int
`mean_shift_prob`	float
`var_shift_prob`	float
`new_mode_prob`	float
`mean_shift_scale_min`	float
`mean_shift_scale_max`	float
`var_scale_min`	float
`var_scale_max`	float

In der Format-Spalte steht *list[ ]* für einen oder mehrere Werte. *int* steht für eine ganze Zahl, während *float* eine reellwertige Zahl darstellt. *str* stellt beliebig lange Zeichenketten dar.

`modes` steht für die Anomalietypen, die bei einer Injektion stattfinden. Diese lassen sich als eine beliebige Kombination der folgenden Anomalietypen angeben:

Tabelle 4.3. Anomalietypen der Injektion

Anomalietyp	**Beschreibung**
`mean_shift`	Verschiebung der Mittelwerte
`var_shift`	Skalierung der Varianzen
`new_mode`	Neuer Modus kombiniert aus `mean_shift` und `var_shift`

`stream_anomaly_prob` gibt an, wie hoch die Wahrscheinlichkeit ist, dass ein erneut auftretender Betriebsmodus mit einem Anomalietyp injiziert wird.

`stream_use_seen_template_prob` gibt an, wie hoch die Wahrscheinlichkeit ist, dass ein injiziertes Anomaliemuster erneut verwendet wird. Dafür muss erst eine komplett neue injizierte Anomalie gesehen werden.

`max_stream_anomaly_template` gibt an, wie viele Anomaliemuster gespeichert werden. Ist der Speicher voll, dann werden neu auftretende Anomaliemuster nicht erneut genutzt, wenn auf zuvor gesehenen Anomalien zurückgegriffen wird. Dadurch wird die Vielfalt der Anomaliemuster begrenzt, während dennoch einmaliges, nicht-wiederkehrendes anomales Verhalten abgebildet werden kann.

`mean_shift_prob` gibt die Wahrscheinlichkeit an, mit der die `mean_shift` Anomalie auftritt, wenn diese in `modes` angegeben ist.

`var_shift_prob` gibt die Wahrscheinlichkeit an, mit der die `var_shift` Anomalie auftritt, wenn diese in `modes` angegeben ist.

`new_mode_prob` gibt die Wahrscheinlichkeit an, mit der die `new_mode` Anomalie auftritt, wenn diese in `modes` angegeben ist.

Die Summe aller Wahrscheinlichkeiten der in `modes` angegebenen Anomalietypen von `mean_shift_prob`, `var_shift_prob` und `new_mode_prob` muss 1 ergeben.

`mean_shift_scale_min` ist der minimale Faktor, der mit einer Einheitslänge multipliziert wird, um den Mittelwert zu verschieben.

`mean_shift_scale_max` ist der maximale Faktor, der mit einer Einheitslänge multipliziert wird, um den Mittelwert zu verschieben.

Die tatsächliche Mittelwertverschiebung wird wie folgt berechnet

$$\mu_{\text{neu}} = \mu_{\text{alt}} + \texttt{direction} \cdot \texttt{mean\_scale} \tag{4.1}$$

wobei μ_{neu} den neuen Mittelwert und μ_{alt} den alten Mittelwert darstellen. `direction` ist ein zufälliger Richtungsvektor im Raum mit einer Länge von exakt 1. `mean_scale` ist der zufällig gewählte Faktor zwischen `mean_shift_scale_min` und `mean_shift_scale_max`.

`var_scale_min` ist der minimale Faktor, mit dem die bisherige Varianz skaliert wird.

`var_scale_max` ist der maximale Faktor, mit dem die bisherige Varianz skaliert wird.

Mit diesen beiden Konfigurationsmöglichkeiten hat der Datensimulator alle nötigen Informationen gegeben, um eine synthetische, hochdimensionale, multivariate Zeitreihe zu generieren. Dies gilt sowohl für reine Normalzustände zum Training als auch für Normalzustände mit Anomalieinjektionen im Online-Betrieb.

4.4.4. Generierung der Normalzustände

Für die Generierung der Normalzustände wird im Generator des Datensimulators jeweils ein einzelner Datenpunkt generiert. Die Funktion `simulate_data` des `DataSim` (Datensimulatorklasse) generiert jedoch die zuvor konfigurierte Datenmenge auf einmal und gibt diese anschließend zurück. Bei der Generierung von Datenpunkten sind mehrere Voraussetzungen zu beachten. In einer vollen Iteration wird zwischen Betriebs- und Übergangsmodi mehrfach hin und her gewechselt. Bei der Generierung von Betriebsmodi wird ein zufälliger Datenpunkt aus den angegebenen Verteilungen gezogen. Der zugehörige Mittelwert sowie die

Kovarianzmatrix, deren Varianzen zufällig bestimmt werden, ergeben sich gemäß der folgenden Formel.

$$\mathbf{\Sigma} = Q \,\mathrm{diag}(v_1, \ldots, v_d)\, Q^\top \tag{4.2}$$

Dabei bezeichnet $\mathbf{v} = (v_1, \ldots, v_d)$ die vorgegebenen Varianzen, $Q \in \mathbb{R}^{d \times d}$ eine orthogonale Matrix, die aus der QR-Zerlegung einer zufälligen Matrix $A \in \mathbb{R}^{d \times d}$ erzeugt wird, und $\mathrm{diag}(v_1, \ldots, v_d)$ eine diagonale Matrix mit den Varianzen auf der Diagonalen.

Danach wird der Datenpunkt mit den angegebenen Verrauschungsparametern `noise_student_t_df`, `noise_student_t_scale`, `quant_step` und `clip` verrauscht. Während Quantisierung und Clipping ohne weitere Erläuterung erfolgen können, erfordert die Student-t-Verrauschung weitergehendes Verständnis der folgenden Formel:

$$\varepsilon \mid G \sim \mathcal{N}\left(0, \frac{t_{\text{scale}}^2}{G}\right), \qquad G \sim \mathrm{Gamma}\left(\frac{\nu}{2}, \frac{\nu}{2}\right), \tag{4.3}$$

Bedingt auf die latente Variable G ist das Rauschen normalverteilt mit einer durch G^{-1} skalierten Varianz. Die Zufallsvariable G folgt einer Gamma-Verteilung mit Parametern $\nu/2$ und $\nu/2$, wodurch sich für die resultierende Student-t-Verteilung effektiv ν ergibt.

Nach Integration über die latente Variable G ergibt sich eine Student-t-Verteilung, die im Vergleich zur Normalverteilung schwerere Randbereiche und somit eine höhere Wahrscheinlichkeit für Abweichungen aufweist.

Der Übergang zwischen zwei Betriebsmodi A und B wird durch lineare Interpolation der zugehörigen Verteilungsparameter beschrieben. Wenn n die Anzahl der Samples im Übergangsmodus und $i \in \{0, \ldots, n-1\}$ der Index des aktuellen Samples sind, dann gilt:

$$x_i \sim \mathcal{N}\Big(\underbrace{(1 - \frac{i+1}{n})\,\mu_A + \frac{i+1}{n}\,\mu_B}_{\text{Interpolierter Mittelwert } \mu_i}, \underbrace{(1 - \frac{i+1}{n})\,\Sigma_A + \frac{i+1}{n}\,\Sigma_B}_{\text{Interpolierte Kovarianz } \Sigma_i} \Big). \tag{4.4}$$

Dabei entsteht ein kontinuierlicher Übergang zwischen den Betriebsmodi A und B, bei dem der Einfluss von A stetig abnimmt.

Mit diesen Funktionen und Methoden werden Normalzustände mit verschiedenen Konfigurationsmöglichkeiten generiert.

4.4.5. Online Modus und Anomalieinjektion

Beim Generieren von Daten im Online-Betrieb unterscheidet sich lediglich die Anomalieinjektion von der Generierung der Normalzustände. Mit der Funktion `simulate_continuous_data` des `DataSim` lässt sich hier über einen Python-Generator jeder Wert einzeln generieren. Dabei tritt mit der Wahrscheinlichkeit `stream_anomaly_prob` ein Anomalietyp in einem Betriebsmodus auf. Der gesamte generierte anomale Betriebsmodus wird als anomal markiert und erzeugt Daten entsprechend dem Muster des jeweiligen Anomalietyps. Unabhängig davon, ob eine neue oder bereits bekannte Anomalie generiert wurde, wird beim Auftreten einer Anomalie nur die Mittelwerte und Kovarianzen des Betriebsmodus gezielt mit der Anomalie vertauscht, um so anomale Daten zu generieren. Wichtig zu erwähnen ist, dass der gesamte Betriebsmodus als Anomalie markiert wird, obwohl es theoretisch unter bestimmten Konfigurationen möglich ist, dass Anomalien vollständig im Bereich der Normalzustände liegen.

4.5. Evaluationspipeline

Die Evaluationspipeline bietet eine konsistente und vergleichbare Bewertung der betrachteten Modelle unter identischen Streaming-Bedingungen. Die Pipeline umfasst den vollständigen Ablauf von der Initialisierung der Normalzustände über den Online-Betrieb bis hin zur quantitativen Auswertung der Anomalieerkennung der verschiedenen Modelle.

4.5.1. Initialisierung der Modelle

Sowohl im synthetischen als auch im realen Datensatz sind Trainingsdaten enthalten, die für die Initialisierung der Modelle benötigt werden. Im SMD sind für jede Maschine Trainingsdateien gegeben, die per Funktion geladen werden können. Im Datensimulator funktioniert dies wie bereits erläutert über die `simulate_data`-Funktion. Das Baseline-Gaussian-Mixture-Modell wird gegen ein BGME mit Feedback der gemachten Fehler sowie einem BGME mit Feedback aller Daten verglichen. Dabei gelten identische Rahmenbedingungen. Die Daten haben die gleiche Dimensionalität und die Modelle bekommen eine vergleichbare Komponentenanzahl.

4.5.2. Online-Phase

Jeder Datenpunkt wird sequentiell verarbeitet. Für jeden Zeitpunkt berechnen die Modelle einen Anomaliescore, der sich aus der negativen Log-Likelihood

bildet. Die Entscheidung, ob ein Datenpunkt anomal ist, erfolgt über einen Schwellwert. Das Label ist für die BGME verfügbar, wird jedoch nicht direkt in der Online-Entscheidung genutzt, sondern ausschließlich für die verzögerte Modellanpassung im Rahmen des Feedback-Mechanismus. Das Baseline-Modell verwendet kein Boosting und keine klassischen Updates. Damit ist das Modell statisch und einmalig auf den Normalzuständen trainiert. Das BGME mit Feedback der Fehler erhält rückwirkend die Informationen, bei welchen Datenpunkten Fehlklassifikationen aufgetreten sind. Aus diesen Fehlern werden neue Modelle in ein Ensemble aufgenommen. Das BGME mit vollem Feedback aller Daten trainiert rückwirkend unabhängig davon, ob Fehler gemacht wurden oder nicht, mit allen gesehenen Datenpunkten.

4.5.3. Refit-Strategie und Update-Mechanismus

Ein Refit der BGMEs erfolgt periodisch nach einer festgelegten Anzahl von n_{total} Datenpunkten sowie einer n_{min} Mindestanzahl an falschklassifizierter Daten für das BGME mit Feedback der Fehler. Die Datenpunkte mit Label werden intern im Datenpuffer gehalten und anschließend bei einem Refit-Zeitpunkt über ein neues Modell in das Ensemble aufgenommen. Das Ziel des Feedback-Mechanismus der BGME ist die Stabilisierung bei veränderten Normalzuständen. Die Architektur bietet zusätzliche Optionen, um auf Concept Drifts angemessen zu reagieren.

4.5.4. Anomalieentscheidung, Schwellwert und Klassifikationsmetriken

Die Entscheidung, ob ein Datenpunkt anomal ist, erfolgt über einen festen, konfigurierbaren Quantil-Schwellenwert, der vor der Evaluation festgelegt wird. Liegen Punkte mit dem Anomaliescore unterhalb des Schwellwerts des jeweiligen Modells, so werden diese als anomal markiert. Der Quantil-Schwellwert ist für alle Modelle identisch, um eine faire Evaluation durchzuführen und direkte Vergleiche zwischen Modellen zu ermöglichen. Dabei werden Datenpunkte individuell bewertet und nach folgender Tabelle eingeordnet [11]:

Tabelle 4.4. Konfusionsmatrix Erläuterung

Interpretation	**Kürzel**	**Modellausgabe**	**Wahrheit**
`False Positive`	FP	1	0
`False Negative`	FN	0	1
`True Positive`	TP	1	1
`True Negative`	TN	0	0

wobei 1 für die Klassifizierung einer Anomalie steht und 0 für die Klassifizierung von Normaldaten.

Daraus können Kennzahlen für eine quantitative Bewertung abgeleitet werden, beispielsweise der Recall, welcher aus der folgenden Formel gebildet wird:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4.5}$$

Der Recall wird verwendet, um zu zeigen, wie hoch der Anteil der tatsächlich erkannten Anomalien ist und steht für die Detektionsfähigkeit.

Die Precision wird verwendet, um den Anteil der erkannten Anomalien an allen als anomal klassifizierten Punkten zu messen und steht dafür, ob ein Modell viele Falschklassifikationen produziert.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{4.6}$$

Um zu bewerten, wie Modelle abschneiden, wird häufig der F1-Score benutzt, dabei ist es von Bedeutung eine hohe Precision und einen hohen Recall zu kombinieren um einen hohen F1-Score zu erlangen. Damit misst dieser die ausgewogene Qualität einer Anomalieerkennung, indem Fehlalarme und verpasste Anomalien gleichzeitig berücksichtigt werden. Der F1-Score bildet sich wie folgt:

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2\,\text{TP}}{2\,\text{TP} + \text{FP} + \text{FN}} \tag{4.7}$$

Zusammenfassend besteht die Evaluationspipeline aus einer initialen Trainingsphase aller Modelle, die anschließend in einer sequentiellen Online-Phase mit verzögerter Modellanpassung eine quantitative Bewertung der Anomalieerkennung anhand standardisierter Metriken erhalten.

5. Evaluation und Ergebnisse

5.1. Einordnung der Evaluation

Für die Hauptevaluation der Arbeit wird das ServerMachineDataset [4] verwendet. Dabei wird auf allen 28 verschiedenen Maschinen des Datensatzes segmentweise evaluiert, deren Anomalieanteil zwischen etwa 0.01 % und 16.00 % liegt. Die Trainingsdaten, welche die Normalzustände repräsentieren und von den Modellen abgebildet werden sollen, umfassen jeweils etwa 27.500 Zeitpunkte mit 38 Variablen pro Zeitpunkt. Die Testdaten sind ähnlich lang und enthalten zusätzlich annotierte Anomalien. Somit können verschiedene Verhaltensweisen der Modelle basierend auf der Datengrundlage analysiert werden. Es werden drei Modelle miteinander verglichen: ein Baseline-GMM, welches mit einer erhöhten Komponentenzahl die Trainingsdaten lernt, sowie ein BGME, bei dem nur Falschklassifikationen (BGME + Fehler) und ein BGME, bei dem alle gesehenen Datenpunkte in die automatischen Updates aufgenommen werden (BGME + alle Daten).

Die drei Hauptevaluationskriterien sind die bereits beschriebenen Formeln Recall, siehe Gleichung 4.5, Precision, siehe Gleichung 4.6 und F1-Score, siehe Gleichung 4.7. Ein Recall nahe 1 steht dafür, dass ein hoher Anteil der tatsächlichen Anomalien erkannt wurde. Eine Precision nahe 1 beschreibt, dass ein großer Anteil der als anomal klassifizierten Datenpunkte tatsächlich Anomalien sind. Diese beiden Metriken werden in einen F1-Score kombiniert.

Zur besseren Vergleichbarkeit der Modelle werden aggregierte Kennzahlen sowie deren Streuung betrachtet. Die für die Evaluation verwendeten Konfigurationstabellen des Baseline-GMM und der BGMEs sind in Tabelle 5.1 zu finden. Zudem wurde das Modellverhalten auf dem Datensimulator mit zwei verschiedenen Konfigurationen über längere Zeitreihen getestet, was in Kapitel 5.8 dargestellt wird.

5.2. Konfigurationen für die Hauptevaluation

Die folgenden Konfigurationswerte werden für die Hauptevaluation der Modelle verwendet.

Tabelle 5.1. Evaluationskonfiguration der BGMEs

Parameter	**Wert 1**	**Wert 2**
`n_estimators`	6	6
`n_components`	10	10
`covariance_type`	”diag”	”diag”
`reg_covar`	0.0002	0.0001
`max_iter`	500	250
`tol`	0.001	0.002
`random_state`	42	42
`learning_rate`	0.7	0.4
`hard_quantile`	0.9	0.95
`min_weight`	0.000001	0.0001
`max_weight`	1000	200
`combine`	”logmeanexp”	”logmeanexp”
`alpha_decay_rate`	None	None
`max_models_per_ensemble`	30	20

Tabelle 5.2. Evaluationskonfiguration der BGME-Online-Komponenten

Parameter	**Wert 1**	**Wert 2**
`ll_threshold`	None	None
`threshold_quantile`	0.001	0.0005
`refit_window_size`	500	1000
`include_correct_predictions`	True/False	True/False
`min_samples_for_refit`	20	100
`scoring_mode`	”separation”	”separation”
`separation_threshold`	0.0	0.0

Dabei steht True bei `include_correct_predictions` für das *BGME + alle Daten* und False für das *BGME + Fehler*. Die Konfiguration des Baseline-GMMs ist bis auf die Parameter `n_components` und `n_estimators` identisch und verwendet $(n\_components \cdot n\_estimators) \div 2$ als Komponentenanzahl, damit das Baseline-GMM eine realistische Chance hat die Normaldaten ausreichend abzubilden. Für die Hauptevaluation werden Werte der Spalte `Wert 1` verwendet. Die Ergebnisse aus `Wert 2` werden in Kapitel 5.6 mit denen aus `Wert 1` verglichen.

5.3. Segmentweise Performance auf Maschinen-Ebene

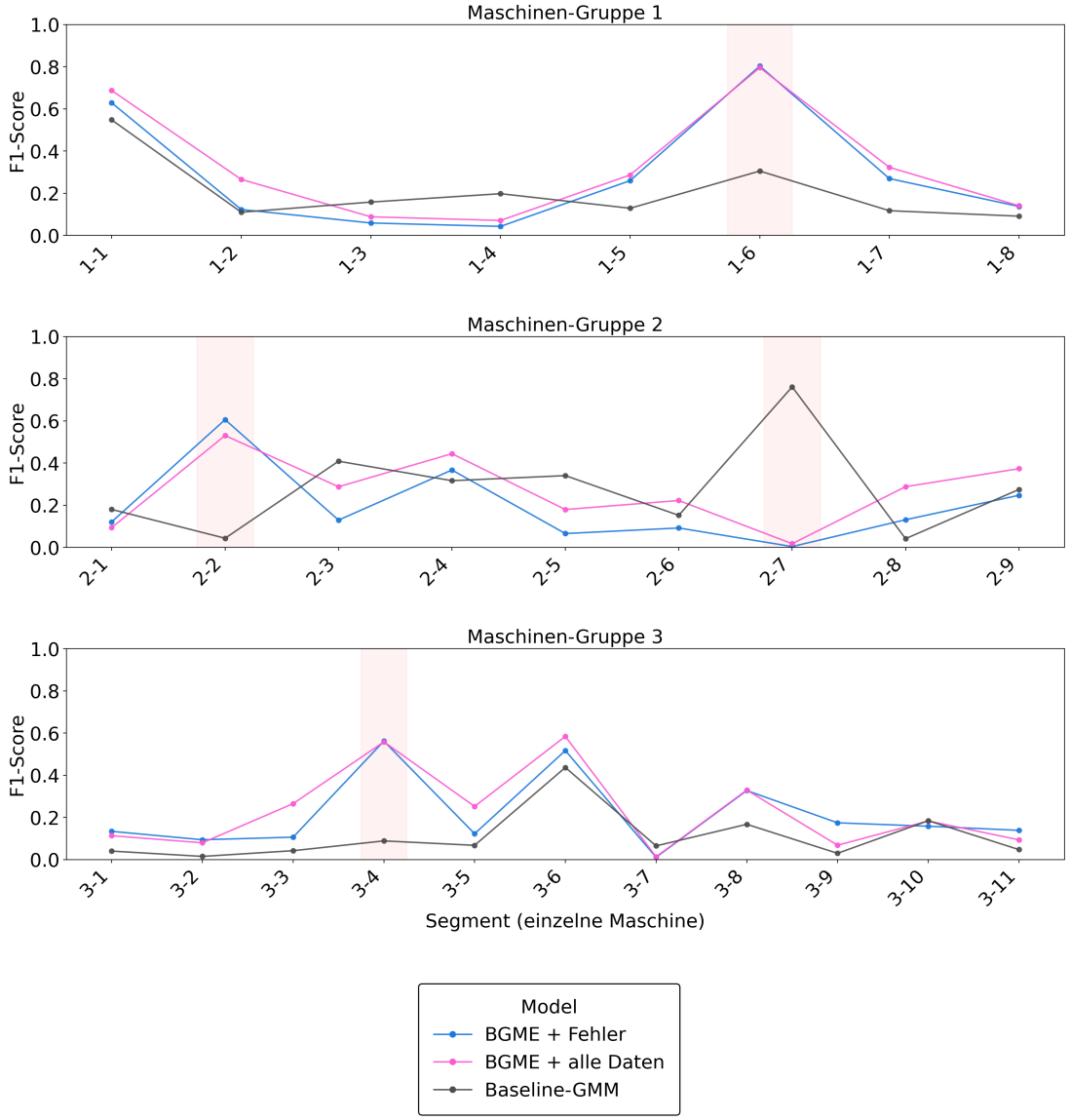

Abbildung 5.1. F1-Scores pro Maschine, große Unterschiede sind rot markiert

Abbildung 5.1 zeigt F1-Scores der einzelnen Modelle pro Maschine. Jeder Datenpunkt repräsentiert die Performance eines Modells auf einer Maschine und entspricht einem einzelnen F1-Score. Je höher der Datenpunkt auf der y-Achse ist, desto besser schneiden die Modelle auf den jeweiligen Segmenten ab. Die zusätzlich rot markierten Maschinen stellen große Unterschiede zwischen dem Baseline-GMM und den BGMEs dar.

Über alle 28 Maschinen hinweg erzielt das BGME + alle Daten in 20 Segmenten einen höheren F1-Score als das Baseline-GMM, während das BGME + Fehler in 18 Segmenten überlegen ist. In Kapitel 5.6 wird ebenfalls festgestellt, dass BGME tendenziell besser abschneiden.

5.3.1. Analyse der Maschinen 1-6, 2-2 und 3-4

Bei den Maschinen 1-6, 2-2 und 3-4 erzielen beide BGMEs einen deutlich höheren F1-Score. Durch das Aufteilen der Zeitreihen in kleinere Segmente sowie die Unterscheidung von Normal- und Anomalieanteilen wird erkenntlich, dass bei allen drei Maschinen die Anomalien überwiegend in klar abgegrenzten, zeitlich zusammenhängenden Segmenten auftreten, während der verbleibende Großteil der Zeitreihen stabile Normalzustände aufweist. Dabei liegt der Anomalieanteil der Zeitreihen zwischen 4.12 % und 15.65 %. Zusätzlich unterscheiden sich die Maschinen 1-6 und 3-4 bereits nach etwa 5.000 Zeitpunkten im Testsatz deutlich von den Normalzuständen. Bei den Maschinen 1-6 und 2-2 treten zusätzlich mehrere räumlich getrennte Anomaliesegmente mit zeitlichem Abstand auf.

Zudem zeigen diese Maschinen deutliche und konsistente Verschiebungen in einzelnen Features zwischen Normal- und Anomaliephasen. Zusammenfassend zeichnen sich die Maschinen 1-6, 2-2 und 3-4 durch eine günstige Kombination aus stabilen Normalzuständen und strukturierten, persistenten Anomalien aus.

5.3.2. Analyse der Maschine 2-7

Bei Maschine 2-7 erzielt das Baseline-GMM einen deutlich höheren F1-Score als beide BGMEs. Auffällig ist der Anomalieanteil von 1.76% in diesem Segment. Dabei treten 305 der insgesamt 417 Anomalien in einem zusammenhängenden Segment auf. Insgesamt gibt es 20 Segmente, einschließlich des 305 Datenpunkte langen Segments. Durch eine detaillierte Analyse der Featureräume fällt auf, dass Feature 28 während des Trainings praktisch konstant ist, im Test jedoch dynamisch schwankt. Diese Dynamik existiert nicht in den Trainingsdaten.

Dadurch, dass diese Anomalie innerhalb der Testzeitreihe nicht erneut auftritt, kann keine Aussage darüber getroffen werden, ob die BGMEs diese in Zukunft hätten erkennen können.

5.4. Globaler FP-FN-Trade-off auf allen Maschinen

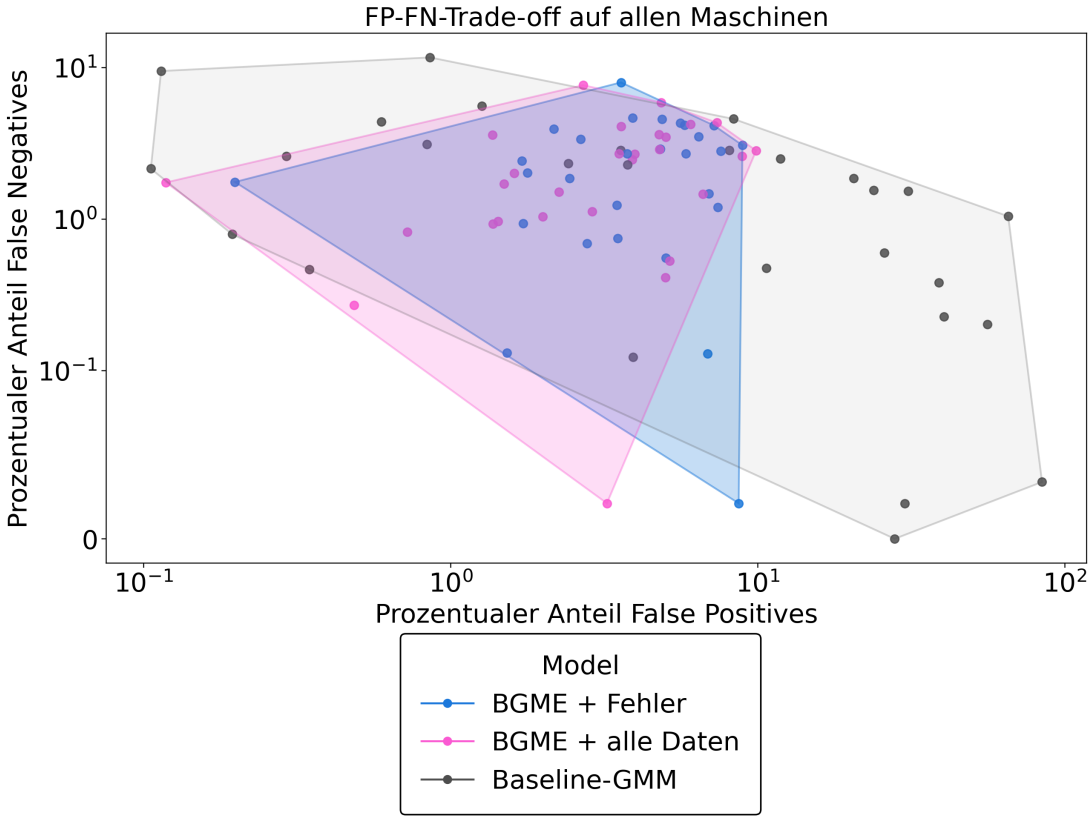

Abbildung 5.2. FP-FN-Trade-off BGME

In der Grafik steht ein hoher y-Wert für viele erzeugte False Negatives (FN), während ein hoher x-Wert für viele erzeugte False Positives (FP) steht. Die Achsen sind logarithmisch skaliert, wodurch relative Abstände zunehmend stärkere Auswirkungen haben. Für jedes Modell steht jeder Punkt für das Ergebnis einer einzelnen Maschine. Die Grafik beschreibt, wie die Modelle mit Fehlalarmen und verpassten Anomalien umgehen. Je kleiner der Bereich ist, den die jeweilige Modellhülle abdeckt, desto geringer ist die Anzahl der gemachten Fehler. Die in Abbildung 5.2 gezeigten Hüllen dienen ausschließlich der Visualisierung der Streuung und stellen keine statistischen Konfidenzregionen dar.

5.5. Einfluss des Anomalieanteils

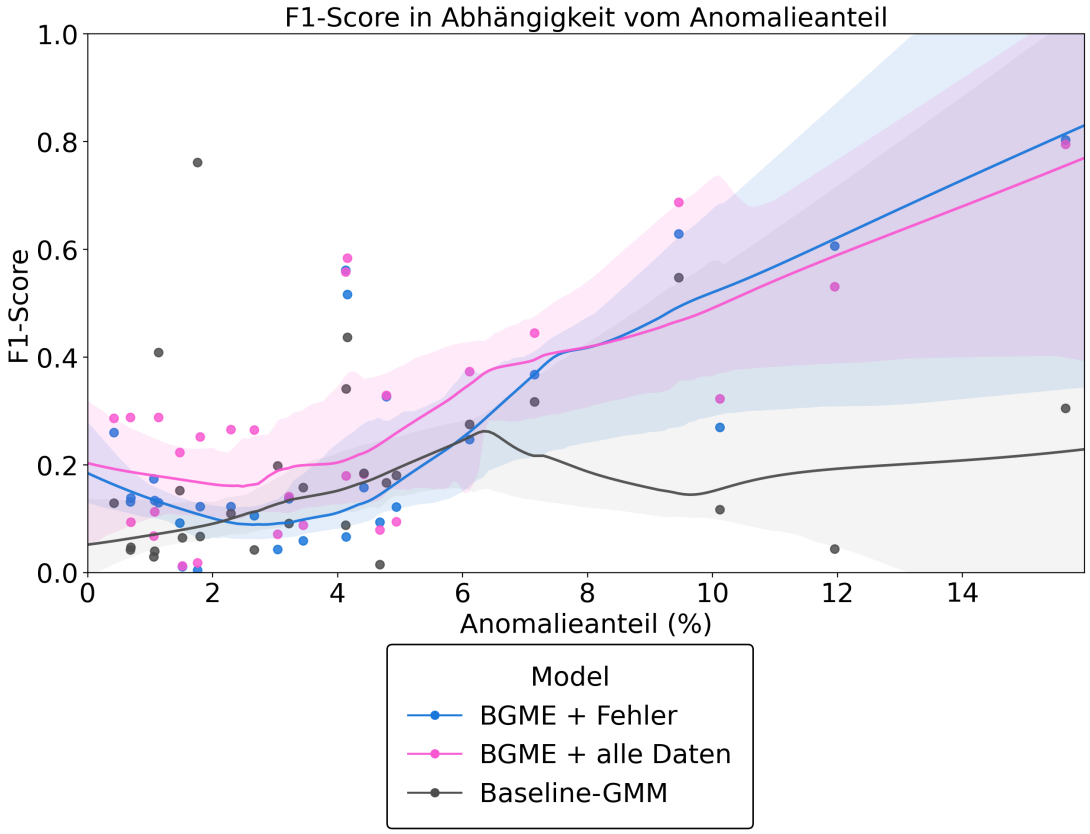

Abbildung 5.3. F1-Score in Abhängigkeit vom Anomalieanteil

Dabei stellt die y-Achse den F1-Score der einzelnen Modelle pro Maschine dar, während die x-Achse den prozentualen Anteil an anomalen Datenpunkten in der Testzeitreihe zeigt. Dadurch wird der Zusammenhang zwischen dem Anteil der Anomaliedatenpunkte und deren direkter Auswirkung auf die Modellleistung sichtbar. Ein BGME benötigt ein Mindestmaß an vorkommenden Anomalien, um durch die strukturellen Vorteile der Architektur zu profitieren. Die dargestellten Unsicherheitsbänder in Abbildung 5.3 geben die Streuung der F1-Scores über verschiedene Maschinen wieder und stellen keine Konfidenzintervalle im inferenzstatistischen Sinn dar, sondern dienen ausschließlich der Visualisierung der Modellvarianz.

5.6. Einfluss der Hyperparameter

Auf Basis der Konfigurationswerte aus der Spalte `Wert 1` der Tabellen 5.1 und 5.2 ergeben sich die folgenden mittleren F1-Scores mit Standardabweichung und Variationskoeffizient. Ein geringerer Variationskoeffizient deutet auf eine gleichmäßigere Modellperformance über verschiedene Maschinen hinweg hin und wird in dieser Arbeit als Maß für Robustheit interpretiert.

Tabelle 5.3. Mittlerer F1-Score und Streuung auf dem ServerMachineDataset

Modell	**F1-Score Mittelwert ± Standardabweichung**	**Variationskoeffizient**
Baseline-GMM	0.191 ± 0.178	0.930
BGME + Fehler	0.230 ± 0.210	0.912
BGME + alle Daten	0.273 ± 0.206	0.757

Auf Basis der Konfigurationswerte aus der Spalte `Wert 2` ergeben sich die folgenden mittleren F1-Scores mit Standardabweichungen und Variationskoeffizienten in der darunter liegenden Tabelle.

Tabelle 5.4. Mittlerer F1-Score und Streuung mit anderen Hyperparametern

Modell	**F1-Score Mittelwert ± Standardabweichung**	**Variationskoeffizient**
Baseline-GMM	0.184 ± 0.179	0.972
BGME + Fehler	0.262 ± 0.226	0.866
BGME + alle Daten	0.290 ± 0.225	0.773

Die Ergebnisse aus den Tabellen 5.3 und 5.4 verdeutlichen, dass sowohl die absolute Streuung der F1-Scores als auch deren relative Variabilität von den gewählten Hyperparametern abhängen. Die mittleren F1-Scores mit Standardabweichungen und Variationskoeffizienten der beiden Konfigurationen unterscheiden sich teilweise deutlich. Trotz dieser quantitativen Unterschiede bleibt die qualitative Bewertung der Modelle stabil.

Die hier verwendeten Hyperparameterkonfigurationen stellen keine optimierten Einstellungen dar, sondern wurden bewusst exemplarisch gewählt und basieren auf in Vorversuchen erprobten Parameterbereichen. Ziel der Evaluation ist die Untersuchung der Robustheit der verschiedenen Modelle, nicht die Maximierung der F1-Scores.

5.7. Zwischenfazit der Ergebnisse

Die Evaluation zeigt deutliche Performanceunterschiede der betrachteten Modelle über die verschiedenen Maschinen hinweg. Das Baseline-GMM weist dabei eine hohe Streuung der Ergebnisse auf, während die beiden BGME-Varianten insgesamt konsistentere F1-Scores über die Maschinen erzielen. BGMEs erlauben durch Online-Refits eine kontinuierliche Anpassung an die Datenstrukturen, anstatt lediglich eine statische Approximation wie beim Baseline-GMM zu verwenden.

Die segmentweise Betrachtung verdeutlicht, dass die erzielbare Erkennungsleistung stark zwischen einzelnen Maschinen variiert. Maschinen mit klar strukturierten Anomaliesegmenten erreichen durchweg höhere F1-Scores als Segmente mit sehr geringem Anomalieanteil oder stark dominierenden Normalzuständen.

Der Zusammenhang zwischen Anomalieanteil und F1-Score zeigt, dass höhere Anomalieanteile tendenziell mit besseren Erkennungsleistungen einhergehen. Die Ergebnisse unterstreichen damit die Relevanz einer differenzierten Auswertung auf Maschinen- und Segmentebene.

5.8. Test in längeren synthetischen Zeitreihen

Mit zwei unterschiedlichen Konfigurationen wurde das Verhalten der BGMEs auf längere Zeitreihen getestet. Einmal wurden seltenere, wenig intensivere Anomalien getestet, einmal häufigere, stärkere Anomalien in einem längeren Datenstrom. Diese Ergebnisse sind nicht Teil der Hauptevaluation, sondern stellen eine explorative Langzeitanalyse dar. Aufgrund der beschriebenen Limitationen, insbesondere der unscharfen Trennung zwischen Normal- und Anomaliedaten im Datensimulator aus Kapitel 4.4.5, lässt sich erwarten, dass klassische Metriken verzerrt sind.

Im ersten Test wird eine ähnliche Konfiguration wie bei den Abbildungen 4.1 und 4.2 verwendet. Die zugehörigen Konfigurationstabellen und Ergebnisse sind wie folgt.

Tabelle 5.5. Erste Langzeittest Datensimulatorkonfiguration

Parameter	**Wert**
Datensimulator	
`num_modes`	3
`num_dimensions`	3
`random_seed`	42
`means_list`	[(1, 1, 1), (1, 5, 10), (10, 10, 10)]
`vars_list`	[(0.5, 0.2, 0.7), (0.1, 0.3, 0.5), (0.2, 0.1, 0.5)]
`mean_range`	None
`var_range`	None
`mode_weights`	[0.5, 0.05, 0.3, 0.03, 0.11, 0.01]
`samples_per_iteration`	1000
`iterations_during_training`	20
`iterations_for_generator`	500
`noise_student_t_df`	5.0
`noise_student_t_scale`	0.2
`quant_step`	0.001
`clip`	(0, 20)
Anomalieinjektion	
`modes`	[”`mean_shift`”, ”`var_shift`”]
`stream_anomaly_prob`	0.08
`stream_use_seen_template_prob`	0.85
`max_stream_anomaly_templates`	8
`mean_shift_prob`	0.5
`var_shift_prob`	0.5
`new_mode_prob`	0
`mean_shift_scale_min`	0.75
`mean_shift_scale_max`	1.5
`var_scale_min`	0.75
`var_scale_max`	1.5

Tabelle 5.6. Erste Langzeittest Modellkonfigurationen

Parameter	Wert
Modellkonfigurationen	
`Baseline-GMM n_components`	12
`BGME n_estimators`	5
`BGME n_components`	5
`covariance_type`	”full”
`reg_covar`	0.0001
`max_iter`	300
`tol`	0.001
`random_state`	42
`learning_rate`	0.15
`hard_quantile`	0.975
`min_weight`	0.000001
`max_weight`	200
`combine`	”logmeanexp”
`alpha_decay_rate`	0.99
`max_models_per_ensemble`	30
BGME Online-Komponente	
`ll_threshold`	None
`threshold_quantile`	0.0005
`refit_window_size`	2000
`include_correct_predictions`	True/False
`min_samples_for_refit`	200
`scoring_mode`	”separation”
`separation_threshold`	0.0

Wobei True bei `include_correct_predictions` für das *BGME + alle Daten* und False für das *BGME + Fehler* steht.

Tabelle 5.7. Erste synthetische Langzeitergebnisse des BGME

Modell	Precision	Recall	F1-Score
Baseline-GMM	0.06	0.49	0.11
BGME + Fehler	0.07	0.43	0.12
BGME + alle Daten	0.07	0.48	0.12

Zusätzlich ist die Konfusionsmatrix dargestellt, bestehend aus True Positives (TP), True Negatives (TN), False Positives (FP) und False Negatives (FN).

Tabelle 5.8. Erste synthetische Langzeit-Klassifikationstabelle des BGME

Modell	TP	TN	FP	FN
Baseline-GMM	15712	231803	236271	16214
BGME + Fehler	13766	281056	187018	18160
BGME + alle Daten	15235	250521	217553	16691

Im zweiten Test werden inklusive Übergangsmodi zehn anstelle von sechs Modi verwendet. Zudem liefert jeder Datenpunkt fünf Variablen pro Zeitpunkt anstelle von drei. Die Anomaliekonfiguration erzeugt stärkere Abweichungen, während die Datensimulatorkonfiguration vermehrt stärkere Ausreißer in den Normaldaten generiert. Die Konfigurationen und Ergebnisse sind wie folgt.

Tabelle 5.9. Zweite Langzeittest Datensimulatorkonfiguration

Parameter	**Wert**
Datensimulator	
num_modes	5
num_dimensions	5
random_seed	42
means_list	[(1, 1, 1, 1, 1), (1, 5, 10, 5, 10), (10, 10, 10, 10, 10), (6, 2, 12, 2, 12), (12, 6, 3, 6, 3)]
vars_list	[(0.5, 0.2, 0.7, 0.2, 0.4), (0.1, 0.3, 0.5, 0.3, 0.5), (0.2, 0.1, 0.5, 0.1, 0.5), (0.4, 0.4, 0.2, 0.4, 0.2), (0.15, 0.25, 0.35, 0.25, 0.35)]
mean_range	None
var_range	None
mode_weights	[0.22, 0.02, 0.2, 0.02, 0.18, 0.02, 0.16, 0.02, 0.14, 0.02]
samples_per_iteration	2000
iterations_during_training	20
iterations_for_generator	400
noise_student_t_df	4.0
noise_student_t_scale	0.3
quant_step	0.001
clip	(0, 20)
Anomalieinjektion	
modes	[”mean_shift”, ”var_shift”, ”new_mode”]
stream_anomaly_prob	0.1
stream_use_seen_template_prob	0.75
max_stream_anomaly_templates	10
mean_shift_prob	0.45
var_shift_prob	0.45
new_mode_prob	0.1
mean_shift_scale_min	0.8
mean_shift_scale_max	2.2
var_scale_min	0.7
var_scale_max	2.0

Tabelle 5.10. Zweite Langzeittest Modellkonfigurationen

Parameter	Wert
Modellkonfigurationen	
`Baseline-GMM n_components`	40
`BGME n_estimators`	8
`BGME n_components`	10
`covariance_type`	”full”
`reg_covar`	0.0003
`max_iter`	300
`tol`	0.001
`random_state`	42
`learning_rate`	0.15
`hard_quantile`	0.98
`min_weight`	0.00001
`max_weight`	50
`combine`	”logmeanexp”
`alpha_decay_rate`	0.99
`max_models_per_ensemble`	30
BGME Online-Komponente	
`ll_threshold`	None
`threshold_quantile`	0.0001
`refit_window_size`	10000
`include_correct_predictions`	True/False
`min_samples_for_refit`	1000
`scoring_mode`	”separation”
`separation_threshold`	0.75

Wobei True bei `include_correct_predictions` für das *BGME + alle Daten* und False für das *BGME + Fehler* steht.

Tabelle 5.11. Zweite synthetische Langzeitergebnisse des BGME

Modell	Precision	Recall	F1-Score
Baseline-GMM	0.09	0.67	0.17
BGME + Fehler	0.09	0.12	0.10
BGME + alle Daten	0.09	0.28	0.14

Bei insgesamt 800.000 Datenpunkten im Testsatz entsteht die folgende Konfusionsmatrix des zweiten Tests.

Tabelle 5.12. Zweite synthetische Langzeit-Klassifikationstabelle des BGME

Modell	TP	TN	FP	FN
Baseline-GMM	52914	238050	483072	25964
BGME + Fehler	9816	620559	100563	69062
BGME + alle Daten	22008	505975	215147	56870

Der Fokus der Ergebnisse liegt auf den relativen und strukturellen Effekten und nicht auf den absoluten Werten. Vor allem in Tabelle 5.12 wird ersichtlich, dass die

BGMEs mehr True Negatives erkennen und damit die Anzahl der False Positives im Vergleich zum Baseline-GMM deutlich reduzieren. Je nach Modellkonfiguration können BGMEs über längere Zeiträume hinweg ein stabileres Normalitätsmodell bereitstellen.

6. Diskussion

6.1. Erkenntnisse der Evaluation

Neben den mittleren F1-Scores liefert die Streuung der Ergebnisse über die einzelnen Maschinen wichtige Hinweise auf die Robustheit der Modelle. Tabelle 5.3 zeigt, dass das Baseline-GMM zwar geringere mittlere F1-Scores aufweist als das BGME + alle Daten, gleichzeitig jedoch den höchsten Variationskoeffizienten besitzt. Dies deutet darauf hin, dass die Modellleistung des Baseline-GMM stark von der jeweiligen Maschine abhängt und in relativer Betrachtung weniger stabil ist.

Das BGME + alle Daten weist trotz höheren mittleren F1-Scores einen deutlich geringeren Variationskoeffizienten auf, was auf eine gleichmäßigere Performance über unterschiedliche Datencharakteristiken hindeutet. Dieser Effekt bleibt auch bei veränderten Hyperparametern bestehen, wie Tabelle 5.4 zeigt, wodurch sich die Beobachtung als robuste Eigenschaft der Architektur interpretieren lässt.

Die Ergebnisse zeigen, dass Boosted-Gaussian-Mixture-Ensembles insbesondere im Hinblick auf die Stabilität der Detektionsleistung Vorteile gegenüber einem einzelnen Gaussian-Mixture-Modell bieten, auch wenn sie nicht in jedem Segment die höchste absolute Performance erreichen.

6.2. Wann funktionieren BGMEs besonders gut

Ein BGME funktioniert insbesondere dann gut, wenn klare Normalzustände in den Trainingsdaten vorhanden sind. Wie in den Annahmen aus Kapitel 4.2 beschrieben, müssen Annotationen sowohl im Training als auch im Online-Betrieb gegeben sein und zeitlich zusammenhängen. Im Online-Datenstrom sind persistente, strukturierte und wiederkehrende Anomaliemuster präsent. Durch die Boosting-Mechanismen lassen sich auch schlecht modellierte Regionen vermutlich besser abbilden, was zu einer tieferen Modellstruktur führen kann. Dadurch entsteht ein Ensemble aus Basislernern, die unterschiedliche Regionen des Normalzustands abdecken und gemeinsam robustere Dichteschätzungen ermöglichen. Gerade bei Maschinen wie 1-6, 2-2 oder 3-4 aus Kapitel 5.3 sind diese Eigenschaften gegeben.

6.3. Wann funktionieren BGMEs nicht gut

BGMEs zeigen eine reduzierte Erkennungsleistung, wenn Normaldaten zu dominant abgebildet werden und ähnlich wie bei Maschine 2-7 wenig Anomalien in den Online-Daten präsent sind. Dadurch kann sich das Ensemble nicht ausreichend anpassen, um zukünftige Daten besser zu modellieren. Einzelne Feature-Shifts, die nicht im Training existieren, können negative Auswirkungen auf die Vorhersagen eines BGME haben. Das BGME ist zusätzlich abhängig von den Annotationen der Anomalien. Sind diese zu grob segmentiert oder ohne klare Abgrenzung zu tatsächlichen Normaldaten, ist dies kontraproduktiv für die Adaption des BGME. Ein BGME eignet sich wahrscheinlich für strukturierte Wiedererkennungsansätze und vermutlich nicht als Drift-Detektor, dafür müssten verschiedene neue Methoden aus dem Ausblick 7.2 implementiert werden.

6.4. Fehlerlernen vs. Lernen aus allen Daten

Ein BGME, das nur mit Fehlklassifikationen lernt, ist konservativer als ein BGME, das mit allen Daten lernt. Beim Lernen mit allen Daten ergibt sich eine schnelle Adaption an die Daten, jedoch besteht die Gefahr der Überspezialisierung. Andererseits ist das Lernen mit Fehlklassifikationen robuster gegenüber Rauschen und passt sich langsamer an Drifts an. Diese Effekte sind im Langzeittest aus Kapitel 5.8 zu erkennen, wobei diese Effekte vermutlich durch das Anpassen der Hyperparameter zusätzlich verstärkt werden können. Damit spiegelt sich in den beiden Strategien der Zielkonflikt zwischen Stabilität und Anpassungsfähigkeit wider, der für Online-Lernverfahren charakteristisch ist. Durch Anpassen der Hyperparameter lässt sich in dieser Architektur spezifischer auf bestimmte Use-Cases anpassen.

6.5. Einfluss des Anomalieanteils

Ein höherer Anomalieanteil erscheint vorteilhaft für ein BGME, da dadurch eine stärkere Adaption an komplexere Anomalien stattfinden kann, was in mehr Detektionen im späteren Verlauf der Daten resultiert. Dies wird in der Abbildung 5.3 verdeutlicht. Bei einem zu geringen Anteil an Anomaliedaten fehlt der Architektur die notwendige Adaptionsfähigkeit. In diesen Fällen wird der Anomaliescore primär durch das initiale Normalmodell bestimmt, wodurch zusätzliche Ensemble-Updates kaum Einfluss auf die Entscheidungsgrenze haben. Aufgrund der fehlenden Adaptionsfähigkeit wirkt das Baseline-GMM bei niedrigem Anomalieanteil stabiler. Bei steigendem Anomalieanteil und einer insgesamt höheren

Anzahl an Anomalien adaptieren BGMEs deutlich besser als das Baseline-GMM. Gerade im ServerMachineDataset stellen kurze Zeitreihen mit wenig Anomalien für BGMEs eine größere Herausforderung dar. Dadurch werden weniger Refits durchgeführt, sodass sich die Vorteile der BGME-Architektur nur eingeschränkt entfalten können, welche bei längeren Zeitreihen ersichtlich werden.

6.6. Limitationen

Die Evaluation weist mehrere Limitationen auf. Die Ergebnisse sind extrem abhängig von den Hyperparametern der verschiedenen Komponenten. In dieser Arbeit werden keine Hyperparameter an die jeweilige Anwendung optimiert, sondern angemessene Hyperparameter verwendet. Ein festes Quantil wird für die gesamte Evaluation benutzt, Log-Likelihood-Schwellwerte werden ebenfalls nicht adaptiv im Prozess optimiert, sondern sind nach dem Trainingsprozess statisch. Es gibt keine echte Online-Labelverzögerung. Labels werden direkt intern gespeichert und anschließend bei einem Refit benutzt. Um eine realitätsnahe Anwendung darstellen zu können, müssten Labels rückwirkend beim Refit übergeben werden. Die Annotation der Anomalien im Datensimulator ist grob und ohne Differenzierung zu echten Normalzuständen. Der Unterschied in der benötigten Rechenzeit der verschiedenen Modelle wird nicht berücksichtigt, jedoch steigt diese bei BGMEs mit mehr Modellen sowohl bei Vorhersagen als auch bei Refits.

6.7. Praktische Implementierung

Sofern die Limitationen ausgebessert sind, eignen sich BGMEs im praktischen Ansatz vor allem dann, wenn ein langlaufendes System gegeben ist. Zusätzlich sind wiederkehrende Muster gegeben. Labels sind verfügbar, das System ist dauerhaft im Einsatz und ausschließlich bei potenziellen Refits pausiert. Ein BGME ist nicht für eine Implementation geeignet, bei der Anomalien extrem selten und ohne Wiederholungsmöglichkeit auftreten. Geeignete Einsatzszenarien sind beispielsweise Zustandsüberwachungssysteme in industriellen oder IT-nahen Umgebungen, bei denen wiederkehrende Abweichungen vom Normalbetrieb auftreten und Labels mit zeitlicher Verzögerung verfügbar sind.

7. Fazit und Ausblick

7.1. Fazit

Ziel dieser Arbeit war die Entwicklung einer Architektur, die eine rückwirkende Anpassung an Daten in hochdimensionalen, multivariaten Zeitreihen ermöglicht. Zu diesem Zweck wurde ein synthetischer Datensimulator entwickelt, um die Problemstellung zu veranschaulichen und eine Langzeitanalyse des Systems zu ermöglichen. Die entwickelten Boosted-Gaussian-Mixture-Ensembles ermöglichen eine konfigurierbare Implementierung eines Anomaliedetektionssystems. Durch rückwirkendes Lernen lassen sich zuvor ungesehene Daten auch mit zeitlichem Abstand in die Modellabbildung integrieren. Die Modelle sind im Rahmen dieser Arbeit gezielt auf Robustheit ausgelegt, jedoch nicht auf die maximale Performance optimiert.

Die Evaluation zeigt, dass Boosted-Gaussian-Mixture-Ensembles in mehreren Maschinen eine stabilere Detektionsleistung als ein einzelnes Gaussian-Mixture-Modell erreichen können, vor allem bei strukturierten und wiederkehrenden Anomalien.

Der Ansatz eignet sich daher insbesondere für langlaufende Systeme mit überwiegend stabilen Normalzuständen und strukturierten, wiederkehrenden Anomalien.

Die beobachteten Leistungsunterschiede beruhen auf empirischen Befunden; eine abschließende Erklärung oder eine grundsätzlich überlegene Modellierungsstrategie lässt sich daraus jedoch nicht ohne Weiteres ableiten. Vielmehr verdeutlichen die Ergebnisse die starke Abhängigkeit der Modellleistung von Datencharakteristik, Anomalieanteil und Feedback-Strategie.

7.2. Ausblick

Diese Methode bietet eine vielversprechende Implementierung der Anomalieerkennung in hochdimensionalen, multivariaten Zeitreihen. Insbesondere durch gezielte Verbesserungen könnte diese Architektur das Potenzial haben, sich zu etablieren.

Log-Likelihood-Schwellwerte könnten im Online-Modus adaptiv bestimmt werden, anstatt nach dem Trainingsprozess statisch zu bleiben. Darüber hinaus könnten Bayesian-Gaussian-Mixture-Modelle, die strukturell an klassische Gaussian-Mixture-Modelle angelehnt sind, eingesetzt werden, um die Anpassungsfähigkeit einzelner Basislerner gegenüber Unsicherheiten und sich verändernden Datenverteilungen weiter zu erhöhen.

Der zeitliche Alphaverfall könnte um explizite Persistenzparameter erweitert werden, um einzelne Basislerner gezielt vom zeitlichen Gewichtungsverlust auszunehmen. Ebenso stellt die aktuell verwendete Refit-Strategie lediglich ein *Proof-of-Concept* dar und müsste für reale Anwendungen um asynchrone oder verzögerte Annotationen ergänzt werden.

Zusätzlich bieten sich automatisierte Verfahren zur Hyperparameteroptimierung an, um die starke Abhängigkeit der Modellleistung von Konfigurationsparametern zu reduzieren. Eine systematische Analyse der Online-Laufzeit sowie des Speicherverbrauchs könnte zudem Aufschluss über die praktische Einsatzfähigkeit des Ansatzes in ressourcenbeschränkten Umgebungen liefern.

Literaturverzeichnis

[1] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, Jul. 2009. [Online]. Available: https://doi.org/10.1145/1541880.1541882

[2] D. Reynolds, *Gaussian Mixture Models*. Boston, MA: Springer US, 2009, pp. 659–663. [Online]. Available: https://doi.org/10.1007/978-0-387-73003-5_196

[3] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S002200009791504X

[4] NetManAIOps, "Smd: Server machine dataset from omnianomaly; anomaly detection," https://www.kaggle.com/datasets/mgusat/smd-onmiad, Kaggle, 2019, commit/Version vom 22. Dezember 2025.

[5] Z. Zamanzadeh Darban, G. I. Webb, S. Pan, C. Aggarwal, and M. Salehi, "Deep learning for time series anomaly detection: A survey," *ACM Comput. Surv.*, vol. 57, no. 1, Oct. 2024. [Online]. Available: https://doi.org/10.1145/3691338

[6] Y. El-Laham, N. Dalmasso, E. Fons, and S. Vyetrenko, "Deep Gaussian mixture ensembles," in *Proceedings of the Thirty-Ninth Conference on Uncertainty in Artificial Intelligence*, ser. Proceedings of Machine Learning Research, R. J. Evans and I. Shpitser, Eds., vol. 216. PMLR, 31 Jul–04 Aug 2023, pp. 549–559. [Online]. Available: https://proceedings.mlr.press/v216/el-laham23a.html

[7] M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy, "Mining data streams: a review," *SIGMOD Rec.*, vol. 34, no. 2, p. 18–26, Jun. 2005. [Online]. Available: https://doi.org/10.1145/1083784.1083789

[8] C. M. Bishop, *Pattern Recognition and Machine Learning*, ser. Information Science and Statistics. Springer, 2006.

[9] T. G. Dietterich, "Ensemble methods in machine learning," in *Multiple Classifier Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 1–15.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn:

Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[11] C. Goutte and E. Gaussier, "A probabilistic interpretation of precision, recall and f-score, with implication for evaluation," vol. 3408, 04 2005, pp. 345–359.

Verwendete Tools

Zur sprachlichen Unterstützung und Überarbeitung einzelner Textpassagen wurde ein Large Language Model (ChatGPT, OpenAI) verwendet. Die inhaltliche Ausarbeitung der Arbeit erfolgte vollständig eigenständig.

Zur Unterstützung bei der Softwareentwicklung wurde ein KI-basierter Codeassistent (GitHub Copilot) eingesetzt. Copilot diente ausschließlich zur Codevervollständigung und zur Generierung syntaktischer Vorschläge; die Konzeption, Implementierung, Überprüfung und Validierung des Codes lagen vollständig beim Autor.

Formelverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Anhang A.

Anhang

Im Anhang sind weitere für diese Arbeit relevante Informationen zusammengefasst, die im Hauptteil keinen unmittelbaren Mehrwert bieten.

A.1. Repository

Der im Rahmen dieser Arbeit entwickelte Quellcode ist in einem Git-Repository abgelegt. Das Repository enthält die vollständige Implementierung des Datensimulators, des Boosted-Gaussian-Mixture-Ensembles (BGME) sowie die zugehörige Evaluations- und Visualisierungspipeline.

Zusätzlich sind Skripte zur Durchführung der Experimente sowie zur Erzeugung der in Kapitel 5 dargestellten Abbildungen enthalten. Das Repository ist so gestaltet, dass sämtliche Experimente mit identischen Zufallsseeds und Parametereinstellungen reproduziert werden können.

Das Repository ist unter folgender URL öffentlich zugänglich: `https://github.com/robinkorn/bgme`

A.2. Danksagung

"Falls du wen zum Gegenlesen brauchst, frag bitte nicht mich - ich werde es eh nicht machen." - Fynn, *vermutlich sarkastisch gemeint*

Danke, Fynn, für die LaTeX-Strukturen der Arbeit.

Danke Anna, Jonas, Filip und David [1] für das Korrekturlesen sowie das wertvolle Feedback.

Danke, Stefan, für den Themenvorschlag, die teils sehr ambitionierten Meilensteintermine zur Themenfindung und Eingrenzung sowie das entgegengebrachte Vertrauen in meine experimentelle Herangehensweise.

[1] Die Reihenfolge der Namensnennung wurde in einem 3-Round-Match-Schere-Stein-Papier-Blind-Mini-Round-Robin-Turnier bestimmt.

A.3. Zusätzliche Konfigurationstabellen

Tabelle A.3.1. Konfiguration des Datensimulators für Abbildung 4.1 und 4.2

Parameter	Wert
num_modes	3
num_dimensions	3
random_seed	123
means_list	[(1, 1, 1), (1, 5, 10), (10, 10, 10)]
vars_list	[(0.5, 0.2, 0.7), (0.1, 0.3, 0.5), (0.2, 0.1, 0.5)]
mean_range	None
var_range	None
mode_weights	[0.5, 0.05, 0.3, 0.03, 0.11, 0.01]
samples_per_iteration	1000
iterations_during_training	10
iterations_for_generator	10
noise_student_t_df	5.0
noise_student_t_scale	0.2
quant_step	0.001
clip	None

Die Visualisierung in Abbildung 4.1 und 4.2 werden ohne die negativen Teile der Dimensionen generiert, auch wenn Datenpunkte potenziell unter 0 liegen können werden diese nicht angezeigt.

Tabelle A.3.2. Konfiguration der Anomalieinjektion für Abbildung 4.2

Parameter	Wert
modes	[”mean_shift”, ”var_shift”, ”new_mode”]
stream_anomaly_prob	0.1
stream_use_seen_template_prob	0.5
max_stream_anomaly_templates	32
mean_shift_prob	0.4
var_shift_prob	0.4
new_mode_prob	0.2
mean_shift_scale_min	0.5
mean_shift_scale_max	2.0
var_scale_min	1.1
var_scale_max	3.0