Institut für Parallele und Verteilte Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit Nr. 3508

Zeitreihenanalyse auf dünnen Gittern

David Pfander

Studiengang:	Informatik
Prüfer/in:	Prof. Dr. Dirk Pflüger
Betreuer/in:	M.sc. Fabian Franzelin
Beginn am:	10. Juni 2013
Beendet am:	10. Dezember 2013
CR-Nummer:	H.2.8

Kurzfassung

Zeitreihen sind Mengen von zeitlich geordneten Beobachtungen und fallen bei nahezu allen messbaren Daten an. In dieser Arbeit wird das Vorhersageproblem für Zeitreihen untersucht, für das viele praktische Anwendungen existieren, darunter die Vorhersage von Börsendaten. Für die Untersuchung von Zeitreihen können Gitter-basierte Ansätze verwendet werden. Bei diesen treten jedoch bei hohen Problemdimensionen unpraktikabel große Rechenzeiten auf. In dieser Arbeit wird eine Methode zur Zeitreihenanalyse mit dünnen Gittern vorgestellt, die es erlaubt, Lösungen für Probleme mit höherer Dimensionalität zu berechnen. Die durchgeführten Experimente zeigen dabei, dass für einige Datensätze Vorhersagen mit sehr hoher Qualität berechnet werden. Gleichzeitig ist die benötigte Rechenzeit für viele zeitkritische Anwendungen bereits ausreichend. Um das Anwendungsspektrum der Methode weiter zu vergrößern, werden Optimierungen vorgestellt, mit denen die benötigte Rechenzeit weiter verringert wird.

Inhaltsverzeichnis

1. Einleitung **9**

2. Data Mining mit dünnen Gittern **13**
2.1. Was sind dünne Gitter? . . . 13
2.2. Knowledge Discovery und Data Mining . . . 14
2.3. Grundlagen der dünnen Gitter . . . 14
2.4. Klassifikation mittels der Methode der kleinsten Quadrate . . . 22
2.5. Klassifikation mittels Dichteschätzung . . . 24
2.6. Regressionaufgaben auf dünnen Gittern . . . 27
2.7. Adaptivität . . . 27

3. Zeitreihenanalyse mit dünnen Gittern **31**
3.1. Grundlagen der Zeitreihenanalyse . . . 31
3.2. Das Vorhersageproblem für Zeitreihen . . . 32
3.3. Zeitreihenanalyse als Data Mining Problem . . . 34
3.4. Regularisierung und Validierung . . . 35
3.5. Der gesamte Algorithmus . . . 37

4. Die Konstruktion von Attributräumen **39**
4.1. Das allgemeine Verfahren . . . 39
4.2. Attributkonstruktoren . . . 41
4.3. Über die Wahl der Datenpunkte . . . 42
4.4. Möglichkeiten der Datenvorverarbeitung . . . 43
4.5. Dimension des Dünngitterraums und Anzahl der Datenpunkte . . . 45

5. Datensätze und Vorhersagequalität **47**
5.1. Der synthetische Datensatz „Kurve“ . . . 48
5.2. Der synthetische Datensatz „Muster“ . . . 53
5.3. Experimente mit Finanzdatensätzen . . . 57
5.4. Interpretation der qualitativen Ergebnisse . . . 65

6. Beschleunigung der Zeitreihenanalyse **67**
6.1. Die verwendete Plattform . . . 67
6.2. Benötigte Rechenzeit ohne Optimierungen . . . 67
6.3. Ausgangspunkt und Methodik der Optimierungen . . . 70
6.4. Wiederverwertung des Koeffizientenvektors über mehrere Zeitschritte . . . 71
6.5. Levelreduktion durch Adaptivität . . . 72

6.6. Zoom-Ansatz 75

7. Zusammenfassung und Ausblick **79**

A. Anhang **81**
A.1. Evaluation des Dichte-basierten Ansatzes 81

Literaturverzeichnis **87**

Abbildungsverzeichnis

2.1. Eindimensionale Hutfunktion und nodale Basis. 16
2.2. Interpolation mit nodaler Basis in einer Dimension. 18
2.3. Konstruktion eines Dünngitterraums ohne Rand. 19
2.4. Konstruktion eines Dünngitteraums mit Rand. 20
2.5. Interpolation auf einem dünnen Gitter in einer Dimension. 22
2.6. Dünngitterfunktionen des Schachbrettdatensatzes als Heatmap. 26
2.7. Illustration des Prinzips der Verfeinerung in zwei Dimensionen. 28

3.1. Schema des iterierten Vorhersageproblems. 33

5.1. Verlauf der Zeitreihe des „Kurve“-Datensatzes. 49
5.2. Vorhersagequalität des „Kurve“-Datensatz. 50
5.3. Zeitreihe des „Kurve“-Datensatzes im Attributraum. 52
5.4. Illustration der Periodizität der Differenzen im „Kurve“-Datensatz. 53
5.5. Die Zeitreihe des „Muster“-Datensatzes. 54
5.6. Schema der Konstruktion des „Muster“-Datensatzes und berechnete Funktion. 55
5.7. Vorhersagequalität des „Muster“-Datensatz. 56
5.8. Die Zeitreihe des „DAX“-Datensatz. 58
5.9. Vorhersagequalität des „DAX“-Datensatz. 59
5.10. Die Zeitreihe des „Dow Jones“-Datensatz. 61
5.11. Vorhersagequalität des „Dow Jones“-Datensatz. 62
5.12. Verlauf des Dollar in Euro Kurses im Dezember 2012. 63
5.13. Vorhersagequalität des „FX“-Datensatz. 64

6.1. Benötigte Rechenzeit für hohe Trefferraten. 69
6.2. Koeffizienten-Wiederverwertung mit zwei Datensätzen. 71
6.3. Adaptivität bei niedrigerem Level angewendet auf den „Muster“-Datensatz. . 74
6.4. Adaptivität bei niedrigerem Level angewendet auf den „FX“-Datensatz. 74
6.5. Verfeinern der Umgebung mit zwei Datensätzen. Dabei wird jeweils die Schrittanzahl und damit die Dimension des Attributraums variiert. 76

A.1. Vorhersagequalität des „Kurve“-Datensatz mit dem Dichte-basierten Ansatz. . 82
A.2. Vorhersagequalität des „Muster“-Datensatz mit dem Dichte-basierten Ansatz. 83
A.3. Vorhersagequalität des „DAX“-Datensatz mit dem Dichte-basierten Ansatz. . . 84
A.4. Vorhersagequalität des „Dow Jones“-Datensatz mit dem Dichte-basierten Ansatz. 85
A.5. Vorhersagequalität des „FX“-Datensatz mit dem Dichte-basierten Ansatz. . . . 86

Verzeichnis der Algorithmen

3.1. Der gesamte Algorithmus 38

6.1. Verfeinerungs- und Vergröberungsschema 73

1. Einleitung

Eine Zeitreihe ist eine Menge von zeitlich geordneten Beobachtungen. Zeitreihenanalyse ist die Untersuchung von Zeitreihen, dabei sind unter anderem die folgenden drei Arten von Untersuchungen möglich [BJR08]:

1. Die Vorhersage zukünftiger Zeitschritte einer Zeitreihe.
2. Bestimmung einer Transferfunktion aus zwei gegebenen Zeitreihen. Dabei stellt eine Zeitreihe eine Eingabegröße dar. Über die andere Zeitreihe ist die Antwort eines dynamischen Systems auf die aufeinanderfolgenden Eingaben gegeben.
3. Die Untersuchung multivariater Zeitreihen. Hier werden mehrere Zeitreihen von korrelierten Größen zusammen untersucht. Ziel dieser Untersuchung ist vor allem die Verbesserung von Vorhersagen gegenüber der Betrachtung einer einzelnen Zeitreihe.

In dieser Arbeit werden alle drei vorgestellten Untersuchungen behandelt. Im Fokus steht dabei die Vorhersage zukünftiger Werte einer Zeitreihe. Dafür wird eine unbekannte Vorhersagefunktion mithilfe gegebener Zeitreihen approximiert, wobei die approximierte Vorhersagefunktion kann auch als Transferfunktion interpretiert werden kann. Außerdem werden die Auswirkungen auf die Vorhersage durch Verwendung korrelierter Zeitreihen betrachtet, was einer Untersuchung multivariater Zeitreihen entspricht.

Es existieren viele unterschiedliche Methoden zur Analyse von Zeitreihen [BJR08, SS10], darunter Support Vector Machines [HDO+98, CT03] und neuronale Netzwerke [Zha03]. Die Ansätze, die in dieser Arbeit betrachtet werden, sind Gitter-basierte Methoden. Dazu wird die Zeitreihe in einen Attributraum übersetzt, der es erlaubt Gitter-basierte Regressions- oder Klassifikationsalgorithmen zur Approximation einer unbekannten Vorhersagefunktion zu nutzen.

Gitter-basierte Methoden besitzen die problematische Eigenschaft, dass die benötigte Rechenzeit bei der Berechnung von höherdimensionalen Problemen schnell sehr groß wird. Dieses Phänomen wird als „Fluch der Dimensionalität“ bezeichnet [Bel61]. Durch die Verwendung von dünnen Gittern kann dieses Problem reduziert werden. Dünne Gitter verwenden eine hierarchische Basis im Gegensatz zur nodalen Basis, die bei vielen Gitter-basierten Ansatzen eingesetzt wird. Für dünne Gitter kann trotz einer deutlichen Reduktion der Anzahl der Gitterpunkte gezeigt werden, dass ein ähnlicher Interpolationsfehler wie bei der Verwendung herkömmlicher Gitter vorliegt. Aufgrund der reduzierten Anzahl an Gitterpunkten kann damit eine deutlich reduzierte Rechenzeit erreicht werden, besonders bei der Betrachtung höherdimensionaler Probleme [BG04, Pfl10, Gar04].

Um darzulegen, dass die Methode erfolgreich zum Vorhersagen von Zeitreihen genutzt werden kann, werden fünf Datensätze vorgestellt. Zwei der Datensätze wurden synthetisch generiert und dienen der grundsätzlichen Validierung des Ansatzes. Zusätzlich werden Experimente mit drei nichtsynthetischen Datensätzen durchgeführt, mit denen eine erfolgreiche Anwendung des Ansatzes auf realistischere Probleme gezeigt werden soll. Im Einzelnen wurden Daten der Aktienindizes DAX und Dow Jones verwendet, sowie zusätzlich der Kurs des Euro gegenüber dem Dollar.

Neben der Qualität der Vorhersagen, die mit dieser Methode erreichbar ist, wird angestrebt, diesen Ansatz auch für zeitkritische Anwendungen verfügbar zu machen. Dazu wird zunächst mittels einiger Experimente untersucht, wie hoch die Rechenzeitanforderungen durch die verwendete Methode sind. Anschließend werden einige Optimierungen vorgestellt, mit denen die benötigte Rechenzeit weiter verringert wird, damit noch stärker zeitkritische Datensätze mit dieser Methode untersucht werden können.

Gliederung

Die Arbeit ist in folgender Weise gegliedert:

Kapitel 2 – Data Mining mit dünnen Gittern: Hier wird in die zugrunde liegende Theorie der dünnen Gitter eingeführt. Weiterhin wird der Data Mining Prozess beschrieben, der auch in dieser Arbeit Verwendung findet. Anschließend wird ausgeführt, wie bestimmte Data Mining Probleme mithilfe der Theorie der dünnen Gitter gelöst werden können.

Kapitel 3 – Zeitreihenanalyse mit dünnen Gittern: In diesem Kapitel wird beschrieben, wie das Vorhersageproblem für Zeitreihen mit den im vorherigen Kapitel beschriebenen Data Mining Methoden gelöst werden kann. Da das beschriebene Verfahren durch Variation von Parametern eine Vielzahl von Modellen für einen gegebenen Datensatz erzeugen kann, wird zudem auf die Validierung passender Modelle eingegangen.

Kapitel 4 – Die Konstruktion von Attributräumen: Um das Vorhersageproblem für Zeitreihen als Data Mining Problem aufzufassen, müssen die Zeitreihen in einen Attributsraum überführt werden. Die Konstruktion geeigneter Attributsräume wird in diesem Kapitel beschrieben.

Kapitel 5 – Datensätze und Vorhersagequalität: Um zu zeigen, dass sich die beschriebene Methode zur Lösung des Vorhersageproblems eignet, werden in diesem Kapitel Experimente vorgestellt, die dies belegen. Dafür werden sowohl synthetische als auch nichtsynthetische Datensätze betrachtet.

Kapitel 6 – Beschleunigung der Zeitreihenanalyse: Da das zweite Ziel dieser Arbeit die Anwendung auf zeitkritische Vorhersageprobleme darstellt, wird in diesem Kapitel untersucht, wie das vorgestellte Verfahren weiter beschleunigt werden kann. Für die Untersuchungen werden die Datensätze aus dem letzten Kapitel verwendet.

Kapitel 7 – Zusammenfassung und Ausblick fasst die Ergebnisse der Arbeit zusammen und stellt Anknüpfungspunkte vor.

2. Data Mining mit dünnen Gittern

Der in dieser Arbeit vorgestellte Ansatz zur Zeitreihenanalyse basiert auf Data Mining Methoden, die ihrerseits wiederum auf dünnen Gittern basieren. Bevor also das Vorhersageproblem für Zeitreihen angegangen werden kann, müssen zunächst die benötigten Grundlagen ausgeführt werden. Als erster Schritt zur Zeitreihenanalyse wird daher in diesem Kapitel in die Theorie der dünnen Gitter eingeführt und die darauf basierenden Data Mining Methoden vorgestellt. Der Ansatz für die Zeitreihenanalyse selbst wird anschließend in Kapitel 3 vorgestellt.

2.1. Was sind dünne Gitter?

Auf gitterbasierte Ansätze mit äquidistanten Gitterpunkten in jeder Dimension kann bei Problemen mit Dimensionalität häufig nicht zurückgegriffen werden, da die Anzahl der Gitterpunkte mit der Dimension exponentiell steigt und die daraus resultierende Rechenzeit nicht mehr praktikabel ist. Es werden $O(N^d)$ Gitterpunkte für ein Problem in d Dimensionen benötigt, wenn in Richtung jeder Dimension N Gitterpunkte verwendet werden [Pfl10]. Dieses Phänomen ist als „Fluch der Dimensionalität“ bekannt [Bel61]. In dieser Arbeit werden gewöhnliche Gitter mit äquidistant verteilten Gitterpunkten in jeder Dimension als voll besetzte Gitter bezeichnet.

Bei einem Ansatz mit dünnen Gittern können häufig gegenüber einem voll besetzten Gitter weniger Gitterpunkte verwendet werden, ohne dabei eine schlechtere Lösung zu erhalten. Dies ist darauf zurückzuführen, dass es mit einem dünnen Gitter möglich ist, nur dort Gitterpunkte zu investieren, wo auch tatsächlich Datenpunkte vorliegen. Das heißt, durch die Verwendung von dünnen Gittern wird es möglich, ein Problem mit ähnlicher Genauigkeit, allerdings bei deutlich reduzierter Rechenzeit zu lösen.

Dünne Gitter wurden zuerst zur Interpolation und zum Lösen partieller Differenzialgleichungen verwendet [Zen91]. Seitdem wurden dünne Gitter insbesondere für Probleme aus dem Bereich der numerischen Quadratur [GG98] und Data Mining [HP13, PFPB13] eingesetzt. Ein Ansatz zur Zeitreihenanalyse mittels dünner Gitter wurde bereits in zwei anderen Arbeiten vorgestellt [GGG10, BG13]. Dort kam allerdings die Kombinationstechnik für dünne Gitter zum Einsatz, die in dieser Arbeit nicht verwendet wird[1].

[1] Die Kombinationstechnik wird von Jochen Garcke ausführlich dargestellt [Gar04].

2.2. Knowledge Discovery und Data Mining

Der Prozess der Wissensentdeckung in Datenbanken (Knowledge Discovery in Databases), aufgrund des englischsprachigen Ursprungs abgekürzt KDD, ist ein nichttrivialer Prozess zum Identifizieren von gültigen, neuen, potenziell nützlichen und verstehbaren Mustern in Daten. Das bedeutet, der KDD-Prozess erlaubt es Muster aus Daten zu extrahieren, die, durch korrekte Interpretation, neues und nützliches Wissen darstellen können [FPsS96]. KDD ist ein Prozess, der aus mehreren Schritten besteht [FPsS96]:

1. Auswahl der Daten: Es werden die Daten ausgewählt, aus der die Muster extrahiert werden sollen. Dafür ist es notwendig zu wissen, nach welchen Mustern gesucht wird und es wird zumindest eine Vermutung benötigt, welche Daten diese Muster enthalten könnten.
2. Vorverarbeitung: Die meisten Daten liegen in nicht direkt verwertbarer Form vor. Häufig muss eine Strategie zum Umgang mit fehlerhaften Daten gefunden werden. Falls außerdem Daten fehlen, muss eine Strategie entwickelt werden, um die Auswirkungen durch fehlende Daten zu minimieren.
3. Überführung der Daten: Die meisten Daten liegen nicht passend vor, um daraus die Muster mittels Data Mining extrahieren zu können. Häufig müssen zunächst als weitere Vorverarbeitung bestimmte Informationen aus den Daten gewonnen werden.
4. Data Mining: In diesem Arbeitsschritt werden aus den vorbereiteten Daten Muster errechnet. Dazu sind Klassifikations- und Regressionsmethoden üblich. In dieser Arbeit wird hierfür auf Methoden zurückgegriffen, die auf der Theorie dünner Gitter basieren.
5. Interpretation oder Evaluation der Daten: Die durch das Data Mining gewonnenen Muster werden interpretiert. Wurden die vorherigen Schritte korrekt durchgeführt, dann kann durch die richtige Interpretation der Muster neues Wissen gewonnen werden. Damit eine Interpretation möglich wird, können weitere Nachverarbeitungsschritte notwendig sein.

Die Vorgehensweise in dieser Arbeit folgt weitgehend diesem Muster, wobei die ersten drei Schritte nicht strikt getrennt werden, sondern als ein großer Vorverarbeitungsschritt betrachtet werden. Eine Aufteilung entsprechend dem beschriebenen Prozess ist jedoch grundsätzlich möglich. Auf die Data Mining Methoden wird in den Abschnitten 2.4 und 2.6 näher eingegangen. Die Vorverarbeitungsschritte werden vor allem in Kapitel 3 ausgeführt. Die Interpretation und Evaluation wird vor allem im Kontext konkreter Experimente in Kapitel 5 stattfinden.

2.3. Grundlagen der dünnen Gitter

Dünne Gitter wurden als Modifikation von voll besetzten Gittern entwickelt. Sie können elegant mittels einer Unterraumkonstruktion dargestellt werden. Damit der Schritt von voll

besetzten Gittern zu dünnen Gittern nachvollziehbar bleibt, wird zur Erläuterung auf volle Gitter zurückgegriffen. Das Interpolationsproblem auf einem voll besetzten Gitter mit einer Basis bestehend aus Hutfunktionen dient dabei als Ausgangspunkt. Zunächst werden jedoch noch einige mathematische Definitionen benötigt.

2.3.1. Mathematische Vorbemerkungen

Für die Darstellung von Vektoren wird eine Stich über dem Variablennamen verwendet. Des Weiteren wird sowohl für voll besetzte als auch für dünne Gitter üblicherweise gefordert, dass als Urbildmenge der d-dimensionale Hyperwürfel $[0,1]^d$ verwendet wird. Diese Konvention wird auch in dieser Arbeit beibehalten. Datenpunkte und Funktionen, die nicht passend vorliegen, werden daher auf $[0,1]^d$ normiert.

Um im Rahmen der Theorie der dünnen Gitter Multi-Indizes verwenden zu können, wird die folgende Relation in einem d-dimensionalen Vektorraum definiert wird:

$$\bar{l} \leq \bar{k} \iff \forall i \in \{1,\ldots,d\} : l_i \leq k_i \tag{2.1}$$

Zusätzlich werden die Summen- und Maximumsnormen für Multi-Indizes definiert mit

$$|\bar{l}|_1 := \sum_{j=1}^{d} l_j, |\bar{l}|_\infty := \max_{1\leq j\leq d} |l_j|. \tag{2.2}$$

Der Abstand zwischen zwei Gitterpunkten wird im Folgenden mit h_l bezeichnet, Er wird definiert als

$$h_l := 2^{-l}. \tag{2.3}$$

Der Parameter l stellt die Diskretisierungsstufe dar und wird auch als Level bezeichnet. In Fällen, in denen die genaue Breite des Intervalls keine Rolle spielt, wird bei h_l gelegentlich auf das Subskript l verzichtet. Im d-dimensionalen Fall wird der Level vektoriell mit $\bar{l}$ angegeben. Die zugehörige Breite der Intervalle ist dann gegeben durch

$$\bar{h_{\bar{l}}} := (2^{-l_1}, 2^{-l_2}, \ldots, 2^{-l_d}). \tag{2.4}$$

Hierdurch werden unterschiedliche feine Diskretisierungen in unterschiedliche Richtungen des verwendeten Raumes angegeben.

Des Weiteren wird das übliche Skalarprodukt des L_2-Raums definiert mit

$$(f,g)_{L_2} := \int_\Omega f(x)g(x)dx \qquad\qquad f,g \in L_2. \tag{2.5}$$

Außerdem wird die durch das Skalarprodukt induzierte Norm des L_2-Raums benötigt, diese ist definiert durch

$$||f||_{L_2}^2 := \int_\Omega f^2(x)dx. \tag{2.6}$$

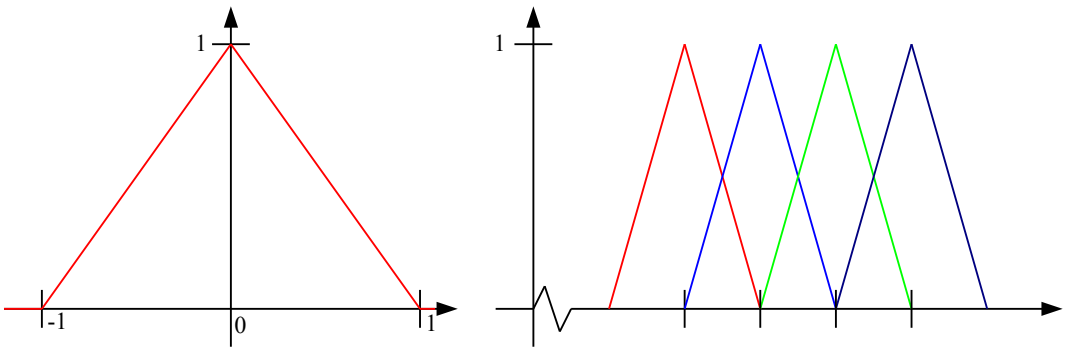


Abbildung 2.1.: Auf der linken Seite ist ein Graph der eindimensionalen Hutfunktion zu sehen. Die rechte Seite zeigt einen Ausschnitt der nodalen Basis in einer Dimension.

Der Support einer Funktion $f : A \to \mathbb{R}$, im Deutschen auch Träger genannt, ist gegeben durch

$$\text{supp}(f) := \overline{\{x \in A | f(x) \neq 0\}} \tag{2.7}$$

Der Support einer Funktion ist also die abgeschlossene Hülle der Menge der Punkte, an denen die Funktion nicht Null ist.

2.3.2. Das Interpolationsproblem in einer Dimension auf einem voll besetztem Gitter

Ein volles besetztes Gitter besitzt äquidistante Gitterpunkte in Richtung jeder Dimension. Voll besetzte Gitter werden häufig zusammen mit der nodalen Basis verwendet, die Hutfunktionen als Basisfunktionen verwendet. Dabei überlappt sich der Support der Basisfunktion an einem Gitterpunkt mit dem Support der Basisfunktionen an direkt benachbarten Gitterpunkten. Durch diese Überlappung werden Punkte zwischen den Gitterpunkten linear interpoliert. Ausgangspunkt für die Basisfunktionen ist die in Abbildung 2.1 auf der linken Seite dargestellte eindimensionale Hutfunktion

$$\phi(x) := \max(1 - |x|, 0). \tag{2.8}$$

Um die Hutfunktion in einem Gitter verwenden zu können, muss sie auf ein Intervall skaliert werden. Zu diesem Zweck wird die Funktion ϕ wie folgt angepasst:

$$\phi_j(x) := \max\left(1 - \left|\frac{x - jh}{h}\right|, 0\right) \tag{2.9}$$

Für eine gewählte Diskretisierung $h_l = 2^{-l}$ zählt der Parameter j die Hutfunktionen an den einzelnen Gitterpunkte auf. Die Menge der Hutfunktionen wiederum kann als Basis für einen Raum

$$V_l := \{\phi_1, \phi_2 \ldots, \phi_{2^l-1}\} \tag{2.10}$$

verwendet werden. V_l besitzt $2^l - 1$ Gitterpunkten mit $2^l - 1$ darauf zentrierten Hutfunktionen auf. Die Hutfunktion an einem der Punkte überlappt sich jeweils mit den direkt benachbarten Punkten. Dies ist in Abbildung 2.1 auf der rechten Seite dargestellt.

Alle Funktionen in V_n können als Linearkombinationen der Basisfunktionen dargestellt werden:

$$f_l(x) = \sum_{i=1}^{n} \alpha_i \phi_i(x) \tag{2.11}$$

Dabei ist n gewählt mit $n = 2^l - 1$. Ein Beispiel für eine einfache Anwendung dieses Funktionsraums ist das Interpolationsproblem. Eine Funktion $f : [0,1] \to \mathbb{R}$ soll in einem vollen Gitter mit Level l interpoliert werden. Dafür müssen initial n Auswertungen an den Gitterpunkten x_i durchgeführt werden, an denen genau eine der Hutfunktionen den Wert 1 annimmt. Diese Menge von Punkten kann als $S = \{(x_i, y_i)\}_{i=0}^{n}$ dargestellt werden. Zur Berechnung der Funktion, mit der die Datenpunkte in S interpoliert werden, muss lediglich für jeden Gitterpunkt der zugehörige Koeffizient α_i auf den Wert y_i gesetzt werden. Damit erhält man direkt die gewünschte interpolierende Funktion $f_n \in V_n$. In Abbildung 2.2 ist dies für ein Gitter mit Level 2 und daraus folgend 3 Gitterpunkten und Basisfunktionen veranschaulicht. Dabei liegt die zu interpolierende Funktion schwarz im Hintergrund. Die drei Basisfunktionen des Gitters sind grün eingefärbt. Die resultierende Funktion $f_2 \in V_2$ ist in roter Farbe dargestellt.

2.3.3. Hierarchische Räume

Dünne Gitter basieren auf einer hierarchischen Basis. Als Basisfunktionen werden, wie bei voll besetzten Gittern, üblicherweise Hutfunktionen verwendet. Da es sich bei Dünngitterräumen um hierarchische Räume handelt, die sich aus Unterräumen zusamensetzen, muss die grundlegende Hutfunktion erneut modifiziert werden. Dafür wird die Hutfunktion auf einem Unterraum eines vorgegebenen Levels l an einem Ort mit einem Index j definiert [Gar11]:

$$\phi_{l,j}(x) = \begin{cases} 1 - |(x - jh_l)/h_l| & x \in [(j-1)h_l, (j+1)h_l] \bigcap [0,1] \\ 0 & \text{sonst} \end{cases} \tag{2.12}$$

Wie bei voll besetzten Gittern ist jede Basisfunktion an einem Gitterpunkt zentriert. Über den Index werden die Gitterpunkte aufgezählt. Der Schnitt mit $[0,1]$ ist notwendig, damit der Rand des Gebiets korrekt berücksichtigt wird, falls ein Gitter mit Punkten auf dem Rand verwendet wird.

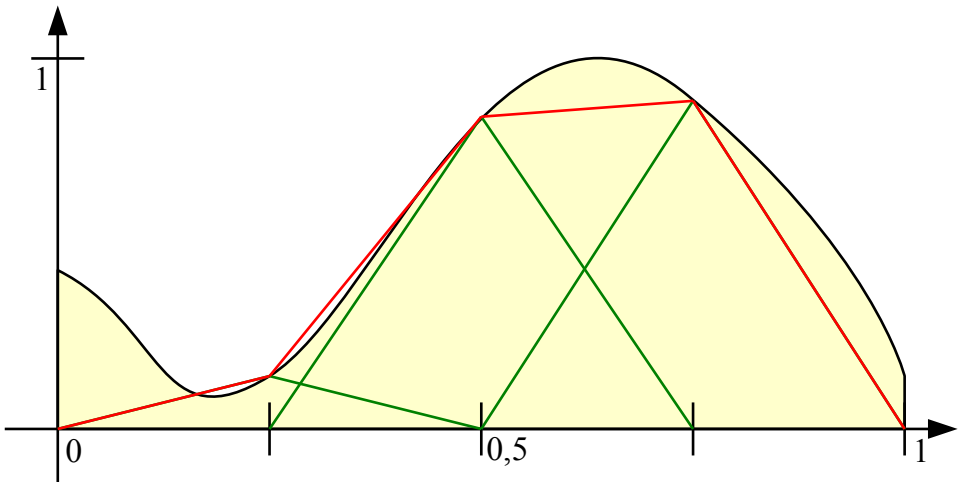


Abbildung 2.2.: Ein einfaches Beispiel für eine Interpolation auf einem voll besetzten Gitter in einer Dimension. Die Basisfunktionen sind in Grün dargestellt. Die rote interpolierende Funktion überlappt sich mit den Basisfunktionen an den Randpunkten.

Mit den Funktionen $\phi_{l,j}$ können die eigentlichen d-dimensionalen Basisfunktionen definiert werden als [Gar11]:

$$\phi_{\bar{l},\bar{j}}(\bar{x}) := \prod_{t=1}^{d} \phi_{l_t,j_t}(x_t) \tag{2.13}$$

Um zu beschreiben, wie sich die Gitterpunkte auf das Gebiet verteilen, wird die Indexmenge

$$I_{\bar{l}} := \{\bar{i} \in \mathbb{N} : 1 \leq i_j \leq 2^{l_j} - 1 \wedge i \text{ ungerade} \wedge 1 \leq j \leq d\} \tag{2.14}$$

benötigt. Die an dieser Stelle etwas arbiträr erscheinende Anforderung, dass i ungerade sein soll, wird später dazu dienen, während der Konstruktion des Dünngitterraums bereits hinzugefügte Gitterpunkte nicht ein zweites Mal hinzuzufügen. Mithilfe von $I_{\bar{l}}$ lässt sich ein Dünngitterraum als direkte Summe von hierarchischen Unterräumen darstellen. Zunächst werden hierzu die hierarchischen Unterräume mittels der oben definierten Basisfunktionen definiert als

$$W_{\bar{l}} := \{\phi_{\bar{l},\bar{i}}(\bar{x}) : \bar{i} \in I_{\bar{l}}\}. \tag{2.15}$$

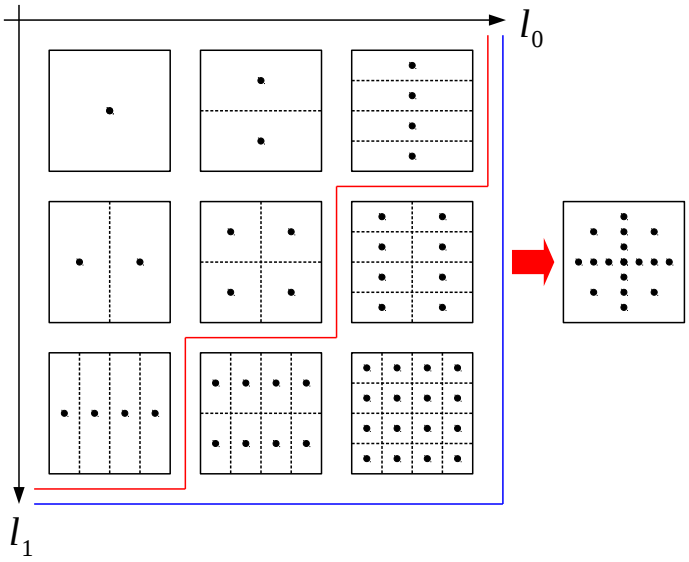


Abbildung 2.3.: Konstruktion eines Dünngitteraums ohne Rand (rot) in zwei Dimensionen mit Level $\bar{l} = (3,3)$. Die Unterräume des zugehörigen vollen Gitters sind blau abgegrenzt.

Dabei stellt $W_{\bar{l}}$ einen Unterraum dar, der genau die Gitterpunkte auf dem durch $\bar{l}$ spezifizierten Level enthält. Werden alle möglichen Unterräume mittels einer direkten Summe[2] kombiniert, erhält man ein volles Gitter durch den Vektorraum V_n^v [Pfl10]:

$$V_n^v := \bigoplus_{|\bar{l}|_\infty \leq n} W_{\bar{l}} \tag{2.16}$$

V_n^v besitzt jetzt eine hierarchische Basis anstatt der nodalen Basis. Voll besetzt ist das Gitter allerdings trotzdem, da die Gitterpunkte in Richtung jeder Dimension äquidistant über das Gebiet verteilt sind. In Abbildung 2.3 sind die hierarchischen Unterräume für ein zweidimensionales hierarchisches Gitter für alle Level bis inklusive $\bar{l} = (3,3)$ dargestellt. Werden alle dargestellten Unterräume summiert, ergibt sich das dargestellte voll besetzte Gitter. Im Unterschied zur nodalen Basis überlappen sich Gitterpunkte mit gleichem Level nicht, stattdessen besitzt die hierarchische Basis eine Baumstruktur. Dabei wird das Gebiet, auf dem eine der Basisfunktionen Support hat, im nächsten Level aufgeteilt. Es existieren dann mehrere nicht überlappende Basisfunktionen, die jeweils auf einer Teilmenge des Gebiets Support besitzen. Durch die gestrichelten Linien ist in Abbildung 2.3 der Support der Basisfunktion abgegrenzt.

[2]Die direkte Summe kann hier als lineare Hülle der Vereinigung der Basisfunktionen der Unterräume betrachtet werden.

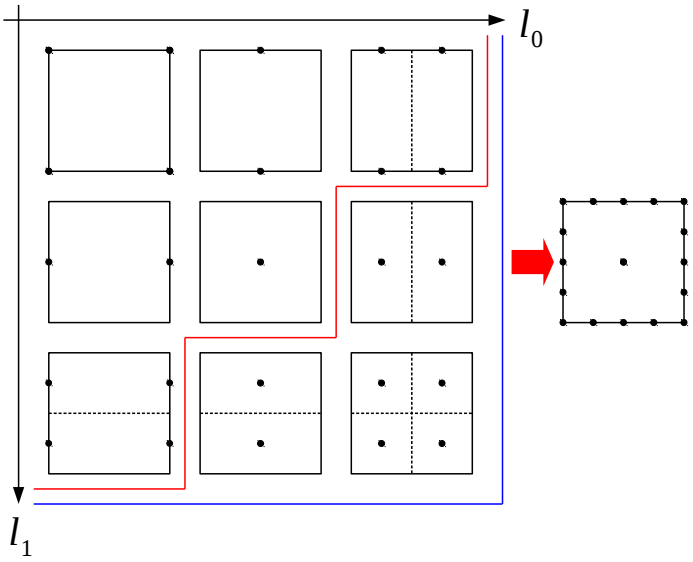


Abbildung 2.4.: Konstruktion eines Dünngitteraums mit Rand (rot) in zwei Dimensionen mit Level $\bar{l} = (3,3)$. Die Unterräume des zugehörigen vollen Gitters sind blau abgegrenzt.

Neben dünnen Gittern ohne Rand, wie sie in dieser Arbeit verwendet werden, existieren noch mehrere andere Varianten hierarchische Unterräume zu dünnen Gittern zu kombinieren [Pfl10]. Eine besonders interessante Variante stellen die dünnen Gitter mit Rand dar, die in Abbildung 2.4 für den zweidimensionalen Fall schematisch dargestellt sind. Während Randpunkte grundsätzlich erwünscht sind, besitzen gewöhnliche dünne Gitter mit Rand die Eigenschaft, dass mit zunehmendem Level die meisten Gitterpunkte am Rand liegen, und nicht mehr im Gebiet. Grundsätzlich gibt es zwei Strategien, um dieses Problem zu lösen. Zum einen können die Bedingungen für die Randpunkte modifiziert werden, wodurch deren Anzahl reduziert wird. Andererseits können die Datenpunkte durch eine geeignete Konstruktion so normalisiert werden, dass Gitterpunkte auf dem Rand nicht benötigt werden. In dieser Arbeit wird auf die zweite Strategie zurückgegriffen.

Durch das Auslassen bestimmter Unterräume erhält man einen Dünngitterraum

$$V_n := \bigoplus_{|\bar{l}|_1 \leq n+d-1} W_{\bar{l}}. \tag{2.17}$$

In Abbildung 2.3 ist die Konstruktion für den zweidimensionalen Fall veranschaulicht. Im Gegensatz zum Vollgitterfall werden jetzt nur die durch die rote, diagonal verlaufende Linie begrenzten Unterräume aufsummiert.

Durch die Konstruktion der Unterräume werden zwei unterschiedliche Levelbegriffe verwendet. Es existieren Unterräume eines bestimmten Levels $\bar{l}$. Gleichzeitig ergibt sich durch

die Unterraumkonstruktion der Level eines Gitter, der oben über das Subskript n angegeben wurde.

2.3.4. Interpolation in hierarchischen Räumen

Sei $\overline{\alpha}$ ein Koeffizientenvektor passender Dimension für den jeweiligen Raum. Dann kann eine Funktion $u \in V_n^v$ als Linearkombination der Basisfunktionen von V_n^v dargestellt werden:

$$u(x) = \sum_{|l|_\infty \leq n} \sum_{\bar{i} \in I_l} \alpha_{\bar{l},\bar{i}} \phi_{\bar{l},\bar{i}}(\overline{x}) \tag{2.18}$$

Analog ergibt sich eine Darstellung für beliebige Funktionen im Dünngitterraum V_n mit

$$u(\overline{x}) = \sum_{|\bar{l}|_1 \leq n+d-1} \sum_{i \in I_{\bar{l}}} \alpha_{\bar{l},\bar{i}} \phi_{\bar{l},\bar{i}}(\overline{x}). \tag{2.19}$$

Im Vergleich zur Berechnung der Koeffizienten bei der „flachen“ Basis aus 2.3.2 ist die Interpolation auf dünnen Gittern aufwendiger und kann effizient mittels einem als Hierarchisierung bezeichneten Verfahren durchgeführt werden [Pfl10]. Als einfachere, allerdings auch weniger effiziente Alternative kann ein lineares Gleichungssystem aufgestellt werden. Dazu wird zunächst die zu interpolierende Funktion wieder an jedem Gitterpunkt x_j ausgewertet, wodurch sich die Menge $S = \{(x_j, y_j)\}_{j=0}^n$ ergibt. Entsprechend der Definition der Funktion u ergibt sich damit das Gleichungssystem mit den Gleichungen

$$y_j = \sum_{|\bar{l}|_1 \leq n+d-1} \sum_{\bar{i} \in I_{\bar{l}}} \alpha_{\bar{l},\bar{i}} \phi_{\bar{l},\bar{i}}(\overline{x_j}). \tag{2.20}$$

für jeden der Gitterpunkte. In Abbildung 2.5 ist ein einfaches Beispiel für eine eindimensionale Dünngitterinterpolation mit Level 2 zu sehen. Die drei Basisfunktion sind in grün dargestellt. Die zu interpolierende Funktion ist schwarz eingefärbt, es handelt sich um dieselbe Funktion wie sie bei dem Vollgitterbeispiel 2.2 zu finden war. Die gestrichelten Pfeile symbolisieren die Werte und Vorzeichen der Koeffizienten. In Rot ist die resultierende interpolierende Funktion u dargestellt.

Für den asymptotischen Interpolationsfehler $||f(x) - u(x)||_{L_2}$ einer interpolierenden Funktion u kann für volle Gitter mit Level n und Intervallbreite $h_n = 2^{-n}$ gezeigt werden, dass der Fehler mit $O(h_n^2)$ abfällt. Dafür werden allerdings auch $O(2^{n^d})$ Gitterpunkte benötigt [BG04]. Wird mit einem dünnen Gitter interpoliert, dann ist der Fehler im Vergleich zum voll besetzten Gitter etwas größer mit

$$O(h_n^2 (log(h_n^{-1}))^{d-1}). \tag{2.21}$$

Dafür steigt die Anzahl der Gitterpunkte mit zunehmendem Level des Gitters deutlich langsamer [BG04]:

$$O(h_n^{-1} (log(h_n^{-1}))^{d-1}) = O(2^n (log(2^n)^{d-1}) \tag{2.22}$$

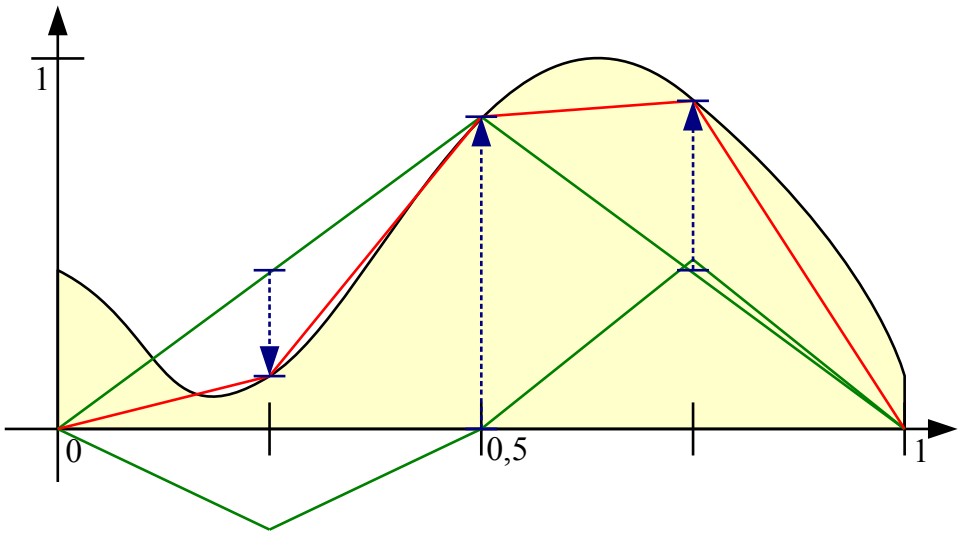


Abbildung 2.5.: Einfaches Beispiel einer Interpolation auf einem dünnen Gitter in einer Dimension. Die drei Basisfunktionen sind grün dargestellt, die blauen Pfeile stellen die Werte der Koeffizienten dar. Die interpolierende Funktion ist Rot dargestellt.

Das heißt, trotz der deutlich reduzierten Anzahl an Gitterpunkten ist bei einem dünnen Gitter immer noch ein ähnlicher Fehler wie bei einem voll besetzten Gitter zu erwarten. Aufgrund dieser Eigenschaft können mithilfe von dünnen Gittern Probleme mit höherer Dimension angegangen werden können, bei der ein volles Gitter bereits zu viel Rechenzeit benötigen würde, ohne dabei starke Einbußem hinsichtlich der Qualität der Lösung hinnehmen zu müssen.

2.4. Klassifikation mittels der Methode der kleinsten Quadrate

Klassifikation ist die Bestimmung einer Funktion, die Datenpunkten Klassenbezeichnungen zuordnet. Für die Bestimmung der Funktion wird eine Menge von Trainingsdaten verwendet. Sei

$$S := \{(\vec{x}_i, y_i) \in \mathbb{R}^d \times T\}_{i=1}^{m} \tag{2.23}$$

die Menge der Trainingsdaten, wobei y_i die Klasse des zugehörigen Datenpunkts $\vec{x}_i$ bezeichnet. Die Menge T ist die Menge der Klassen beziehungsweise Klassenbezeichnungen. Es wird gefordert, dass es sich bei T um eine endliche Menge handelt.

Auf den Trainingsdaten kann ein Klassifikationsproblem formuliert werden. Angenommen die Menge S besteht aus Auswertungen einer unbekannten Funktion $f : \mathbb{R}^d \to T$ aus dem

Funktionsraum V, die jedem Datenpunkt seine zugehörige Klasse zuordnet. Dann kann S genutzt werden, um f zu approximieren.

Zu beachten ist, dass die Trainingsdaten fehlerhaft oder unvollständig sein können, was die maximal erreichbare Approximationsqualität reduziert. Ebenso kann die Qualität eingeschränkt bleiben, falls die Trainingsdaten die unbekannte Funktion strukturell nicht ausreichend präzise wiedergeben, zum Beispiel weil alle Trainingsdaten nur aus einem kleinen Bereich der Urbildmenge der Funktion f stammen. Die Wahl der Trainingsdaten ist also entscheidend für eine gute Approximation der Klassifikationsfunktion.

Zur Lösung des Klassifikationsproblems mit der Methode der kleinsten Quadrate wird zunächst ein Raum V_n mit der Basis $\Psi = \{\phi_i(\overline{x})\}_{i=1}^{n}$ gewählt für den gilt: $V_n \subset V$. Eine approximierende Funktion $f_N(\overline{x})$ kann mithilfe der Basis als Linearkombination der Basisfunktionen und zusätzlichen Koeffizienten α_i geschrieben werden als

$$f_N(\overline{x}) = \sum_{i=1}^{N} \alpha_i \phi_i(\overline{x}). \tag{2.24}$$

Um die Koeffizienten zu bestimmen, mit denen die Trainingsmenge approximiert wird, kann der quadratische Fehler minimiert werden. Formal kann dies als Minimierungsproblem

$$f_N = \underset{f^* \in V_N}{\arg\min} \left(\frac{1}{m} \sum_{i=1}^{m} (y_i - f^*(\overline{x_i}))^2 + \lambda \sum_{i=1}^{N} \alpha_i^2 \right). \tag{2.25}$$

dargestellt werden [Pfl10]. Hierbei dient der Term $(y_i - f^*(\overline{x_i}))^2$ als zu minimierender Fehlerterm zwischen den Koeffizienten der Dünngitterfunktion und den vorliegenden Datenpunkten. Der Term $\sum_{i=1}^{N} \alpha_i^2$ stellt den Regularisierungsoperator dar, er dient also der Glättung der Funktion. Ziel der Regularisierung ist es, dass die Genauigkeit der Vorhersagen auf noch unbekannte Daten maximiert wird. Dafür ist eine geeignete Wahl des Regularisierungsparameters λ notwendig.

Es existieren auch andere Regularisierungsoperatoren, wobei sich der vorgestellte Operator für diese Arbeit als ausreichend erwies. Durch Zeitreihenanalyse als Anwendung des Data Mining Verfahrens ergeben sich einige spezielle Anforderungen für die Regularisierung, die verwendete Strategie zur Wahl des Regularisierungsparameters wird in Abschnitt 3.4 vorgestellt.

Als Lösung des obigen Minimierungsproblems ergibt sich ein lineares Gleichungssystem, welches in Matrixschreibweise notiert werden kann als [Pfl10]

$$\left(\frac{1}{m} B B^T + \lambda I \right) \overline{\alpha} = \frac{1}{m} B \overline{y}. \tag{2.26}$$

Dabei ist $\overline{\alpha}$ der Koeffizientenvektor der Dünngitterfunktion, die Matrix B ist gegeben durch

$$(B)_{ij} := \phi_i(\overline{x_j}). \tag{2.27}$$

Um f_N zu erhalten, muss das Gleichungssystem noch gelöst werden. Als Löser wurde hierfür das Konjugierte-Gradienten-Verfahren, im Folgenden als CG-Verfahren bezeichnet, verwendet [She94].

Durch die Methode der kleinsten Quadrate wird sichergestellt, dass an jeden Datenpunkt $\overline{x_i}$ die Dünngitterfunktion möglichst nahe am Wert y_i liegt. Um die Klassifikationsaufgabe zu lösen, müsste jedoch y_i eine der Klassen aus T sein. Um die Methode der kleinsten Quadrate verwenden zu können, muss zunächst mittels einer Abbildung $g : T \to \mathbb{R}$ jedes Element aus T auf eine reelle Zahl abgebildet werden. Da T endlich ist, stellt dies keine besondere Schwierigkeit dar. Bei zwei Klassen und der Bezeichnermenge $T = \{Rot, Grün\}$ kann den einzelnen Elementen zum Beispiel ein Element aus der Menge $\{0, 1\}$ zugeordnet werden. Damit alle Klassen unterscheidbar als Werte erhalten bleiben und keine neuen Klassenwerte unnötig eingeführt werden, muss die Abbildung bijektiv sein.

Um die berechnete Dünngitterfunktion f_N zur Klassifikation zu benutzen, ist eine Nachbehandlung des mit der Dünngitterfunktion berechneten Werts notwendig, durch die ein inverses Mapping $h : \mathbb{R} \to T$ realisiert wird. Ein derartige Funktion h wird benötigt, weil die Dünngitterfunktion nicht notwendigerweise Werte aus T zurückgibt. Für die Anwendungen im Rahmen dieser Arbeit ist jedoch ein einfaches Verfahren ausreichend. An einem zu klassifizierenden Datenpunkt $\overline{x}$ wird die Klasse gewählt, deren zugeordneter Wert am nächsten zu $f_N(\overline{x})$ liegt:

$$t = \arg\min_{i \in T} |(g(i) - f_N(\overline{x}))| \tag{2.28}$$

Im Rahmen dieser Arbeit werden nur binäre Klassifikationsprobleme betrachtet. Falls jedoch mehr Klassen benötigt werden, kann das vorgestellte Klassifikationsschema problematisch werden. Angenommen einer Menge mit drei Klassen werden die Werte 1, 2 und 3 zugeordnet. Dann kann es vorkommen, dass die Dünngitterfunktion in Bereichen, in denen ein Übergang von 3 nach 1 liegt, den Wert 2 einnimmt. Dies ist vor allem deswegen problematisch, weil dazu keinerlei Trainingsdaten aus der Klasse mit dem Wert 2 in der Nachbarschaft liegen müssen.

Dieses Problem kann durch ein anderes Klassifikationsschema gelöst werden. Anstatt eine einzelne Dünngitterfunktion für alle m Klassen zu erstellen, können auch m Dünngitterfunktionen für die einzelnen Klassen erstellt werden. Ein darauf aufbauendes Klassifikationsschema wird in Abschnitt 2.5 detaillierter ausgeführt, da die einfache obige Strategie für die Dichte-basierte Klassifikation nicht verwendet werden kann.

2.5. Klassifikation mittels Dichteschätzung

Ein Klassifikationsproblem kann auf ein Problem der Dichte-Schätzung zurückgeführt werden. Dazu wird für jede Klasse eine Funktion bestimmt, mittels der die Dichte an jedem Ort des betrachteten Hyperwürfels $[0, 1]^d$ abgeschätzt werden kann. Auf Basis der Dichteschätzungen jeder Klasse an einem bestimmten Punkt im Raum kann mithilfe eines

separat anzugebenden Entscheidungskriteriums die Klasse ausgewählt werden, die am besten zu den errechneten Dichten passt.

Als Ansatz für diese Vorgehensweise wurde von Hegland et al. ein Verfahren zur Dichteschätzung vorgestellt [HHR00], zu dem von Jochen Garcke als Anwendung das Lösen von Klassifikationsproblemen vorgeschlagen wurde [Gar04]. Dieser Ansatz wurde in anderen Arbeiten weiter verfolgt [Fra11, PFPB13].

Als Ausgangspunkt für die Dichteschätzung muss die Trainingsmenge $S = \{\overline{x_i} \in \mathbb{R}^d\}_{i=1}^n$ gegeben sein. Dann kann das folgende Variationsproblem als Ansatz für eine Dichteschätzung verwendet werden [Gar04]:

$$R(f) = ||(f(x)||_{L_2}^2 + \lambda ||Sf||^2 - \frac{1}{M}\sum_{i=1}^{M} f(\overline{x_i}) \tag{2.29}$$

$$R(f) \underset{f \in V}{\rightarrow} \min \tag{2.30}$$

S ist dabei der verwendete Regularisierungsoperator, der wie bei der Methode der kleinsten Quadrate mit der Identität gewählt wird. Unter Verwendung einer Dünngitterbasis kann das gegebene Variationsproblem damit als lineares Gleichungssystem dargestellt werden [Fra11]:

$$(A + \lambda I)\overline{\alpha} = \frac{1}{n} B^T \overline{1} \tag{2.31}$$

Dabei sind die einzelnen Matrizen wie folgt definiert:

$$A_{i,k} := (\phi_i, \phi_k)_{L_2} \qquad i,k \in \{1,2,\ldots N\} \tag{2.32}$$

$$B_{i,j} := \phi_j(x_i) \qquad i \in \{1,2,\ldots n\}, j \in \{1,2,\ldots N\} \tag{2.33}$$

Durch Auswertung der berechneten Dünngitterfunktion an einem Punkt $\overline{x}$, kann damit die Dichte der Trainingsdatenmenge S an diesem Punkt abgeschätzt werden. Damit allein kann jedoch keine Klassifikation durchgeführt werden. Zunächst muss der gegebene Trainingsdatensatz

$$S' := \{(\overline{x}_i, y_i) \in \mathbb{R}^d \times T\}_{i=1}^m \tag{2.34}$$

modifiziert werden, indem er nach Klassen aufgeteilt wird. Dabei entstehen mehrere Datensätze S wie oben angegeben. Für jeden der Trainingsdatensätze der Klassen wird eine Dichteschätzung mithilfe des obigen Verfahrens berechnet. Als Ergebnis erhält man eine Dünngitterfunktion für jede der Klassen. Diese Dünngitterfunktionen erlauben es abzuschätzen, wie dicht eine bestimmte Klasse an einem bestimmten Punkt im Raum liegt. Ein naheliegendes Kriterium zur Klassifikation ist die Wahl der Klasse, die am betrachteten Datenpunkt die höchste Dichte einnimmt [Fra11]. Dies lässt sich darstellen mit

$$t = \arg\max_{i \in T}(f_i(\overline{x})). \tag{2.35}$$

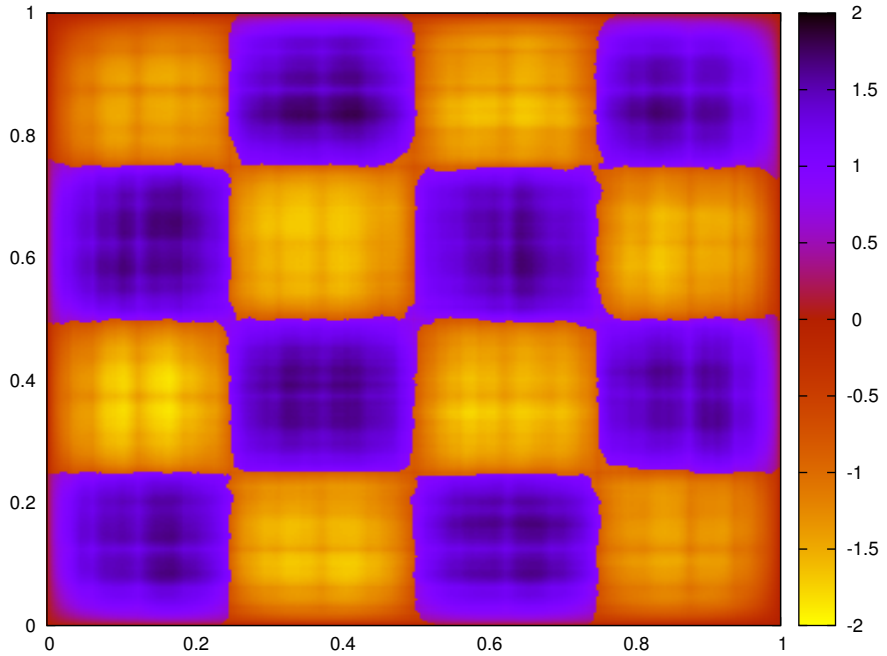


Abbildung 2.6.: Dünngitterfunktionen des Schachbrettdatensatzes als Heatmap dargestellt. Datenpunkte zweier Klassen wurden abwechselnd auf die einzelnen Felder eines Schachbrettmusters verteilt. Die Farbe codiert den Wert der größten Dünngitterfunktion am betrachteten Punkt, das Vorzeichen gibt die Klasse an.

Dabei ist t die Klasse, die dem Datenpunkt $\overline{x}$ zugeordnet wird.

Wie beim vorherigen Ansatz aus 2.4 muss auch hier für die Berechnung der Dünngitterfunktion auf ein Verfahren zum Lösen linearer Gleichungssysteme zurückgegriffen werden. Dafür wurde auch für diesen Ansatz das CG-Verfahren als Löser gewählt.

In Abbildung 2.6 ist ein Beispiel für die Klassifikation mittels Dichteschätzung zu sehen. Als Datensatz dient ein bereits normalisiertes Schachbrettmuster, wobei den Feldern des Schachbretts alternierende Klassen zugeordnet wurden. Die Abbildung zeigt die Werte der Dünngitterfunktionen vor Anwendung des eigentlichen Klassifikationskriteriums. Um beide Dünngitterfunktionen in der Abbildung darzustellen, wurde an jedem Punkt der Abbildung immer nur die Klasse dargestellt, deren Wert größer war. Zusätzlich wurde eine Funktion an der x-Achse gespiegelt, um die Klassen unterscheiden zu können. An der Abbildung kann man damit am Vorzeichen des Werts ablesen, mit welcher Klasse ein Punkt klassifiziert werden würde. Der positive oder negative Betrag zeigt den Verlauf der Funktionen.

2.6. Regressionaufgaben auf dünnen Gittern

Regressionsaufgaben sind eng mit Klassifikationsaufgaben verwandt. Bei Regressionsaufgaben soll eine unbekannte Funktion $f : \mathbb{R}^d \to \mathbb{R}$, wobei f aus dem Funktionsraum V stammt, approximiert werden. Wie bei der Klassifikationsaufgabe sei

$$S := \{(\overline{x}_i, y_i) \in \mathbb{R}^d \times \mathbb{R}\}_{i=1}^m \tag{2.36}$$

eine Menge an Trainingsdaten, die aus Auswertungen der unbekannten Funktion besteht. Im Gegensatz zur Klassifikation wird die Anforderung aufgegeben, dass die Menge der Klassen endlich ist. Auch hier kann die Trainingsdatenmenge fehlerhaft oder unvollständig sein.

Regressionsaufgaben können mittels der Methode der kleinsten Quadrate mit dem in Abschnitt 2.4 vorgestellten Algorithmus angegangen werden, wofür lediglich auf die Anwendung des Klassifikationsschemas verzichtet werden muss. Der Ansatz mittels Dichteschätzung steht für dieses Problem leider nicht zur Verfügung, da dieser darauf basiert, dass für jede Klasse eine eigene Dichteschätzfunktion ermittelt wird. Falls die Werte y_i einen kleinen Wertebereich einnehmen, können diese als Klassen aufgefasst werden. Im Allgemeinen stellen die Werte y_i jedoch keine Klassen dar und können auch nicht als solche betrachtet werden. Dadurch ist es nicht möglich, den Dichte-basierten Ansatz anzuwenden.

2.7. Adaptivität

Das Rechnen mit hohen Leveln oder hoher Dimensionalität führt zur Nutzung vieler Gitterpunkte. Dies kann ein Problem darstellen, da mehr Gitterpunkte bei dem Lösen eines assoziierten Gleichungssystems zu einem entsprechenden Mehraufwand führen. Besonders interessant ist der Fall, in dem ein interessanter Teil des Hyperwürfels $[0, 1]^d$ von zu wenig Gitterpunkten abgedeckt wird, um die unbekannte Funktion f lokal ausreichend gut zu approximieren.

In diesem Fall ist es nicht erwünscht, einfach nur auf höherem Level zu rechnen, denn dies würde dazu führen, dass auch andere Bereiche unnötig fein aufgelöst würden, was wiederum den Aufwand zum Lösen des Gleichungssystems unnötig vergrößert. Gerade bei Klassifikationsproblemen ergibt sich als zusätzliche Schwierigkeit, dass durch zu viele Gitterpunkte relativ zur Anzahl der Datenpunkte Overfitting auftritt. Durch Overfitting kann eine Lösung mit mehr Gitterpunkten schlechter werden, da die Dünngitterfunktion lokal zu schnell abfällt, wodurch zum Beispiel Bereiche zwischen zwei Datenpunkten der gleichen Klassen fälschlicherweise als isolierte Bereiche dieser Klasse durch die Dünngitterfunktion wiedergegeben werden.

Um die Performanceprobleme und das mögliche Overfitting durch einem höheren Level zu vermeiden und trotzdem eine Verbesserung der Lösung zu erzielen, ist es möglich eine bestehende Lösung adaptiv zu verfeinern. Dazu wird das zur unverfeinerten Lösung gehörige Gitter betrachtet. Zunächst wird bestimmt, an welchem oder welchen Gitterpunkten eine Verfeinerung den größten Nutzen bringen würde. Dies kann durch unterschiedliche Kriterien

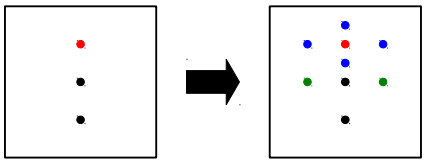


Abbildung 2.7.: Illustration des Prinzips der Verfeinerung in zwei Dimensionen. Der rote Punkt wird verfeinert, wodurch die blauen Punkte eingefügt werden. Die grünen Punkte müssen als hierarchische Vorgänger ebenfalls eingefügt werden.

festgelegt werden. Anschließend werden Punkte in die Umgebung des zu verfeinernden Punktes eingefügt. Eine Möglichkeit besteht darin, das Gebiet, auf der die Funktion des zu verfeinernden Gitterpunktes Support hat, in jede Richtung zu halbieren und auf halbem Weg einen neuen Gitterpunkt hinzuzufügen. Im zweidimensionalen Fall folgt daraus, dass das Gebiet geviertelt wird, im d-dimensionalen Fall werden $2d$ neue Gitterpunkte eingefügt.

Hierbei ergibt sich als zusätzliche Schwierigkeit, dass für die meisten Algorithmen alle hierarchischen Vorgänger eines Gitterpunktes verfügbar sein müssen [Pfl10]. Das bedeutet, dass es unter Umständen nicht ausreicht nur die Gitterpunkte einzufügen, die direkt der Verfeinerung dienen, sondern es müssen auch noch deren hierarchische Vorgänger eingefügt werden. Ein Beispiel, das dieses Phänomen zeigt, ist in Abbildung 2.7 zu sehen. Hier wird der rot markierte Gitterpunkt verfeinert, wodurch die blau markierten Punkte eingefügt werden sollen. Allerdings fehlen die grün markierten hierarchischen Vorgänger, die deswegen ebenfalls eingefügt werden müssen.

Noch zu klären ist, welche Gitterpunkte zu verfeinern sind. Grundsätzlich könnten Fehlerabschätzungen verwendet werden, die es erlauben würden, lokal den Fehler zu bestimmen. Ein empirisch nützliches und äußerst performantes Verfahren ergibt sich allerdings bereits durch die bloße Betrachtung der Koeffizienten. Betrachtet werden hierbei nur Punkte, deren hierarchische Nachfolger noch nicht eingefügt wurden. Das heißt, es werden die Blattknoten des Baumes betrachtet, der mit der hierarchischen Basis assoziiert ist. Bei Gitterpunkten, deren Koeffizienten hohe Werte einnehmen, kann vermutet werden, dass die lokale Approximation noch schlechter ist, als durch die verwendeten Datenpunkte eigentlich erreichbar. Denn sobald die Datenpunkte gut approximiert werden, sollten die Koeffizienten weiterer unnötiger Gitterpunkte nahe bei Null liegen [Pfl10].

Falls lokal eine gute Approximation vorliegt und die Koeffizienten gleichzeitig große Werte einnehmen, gilt, dass die Koeffizienten neu eingefügter Gitterpunkte kleine Werte einnehmen sollten. Das heißt, sollte im falschen Bereich verfeinert werden, wird dies zumindest höchstens einmal geschehen. Dieses Verfeinerungskriterium hat sich für die im Rahmen der in dieser Arbeit durchgeführten Experimente als gute Wahl erwiesen.

Analog zur Verfeinerung von Gitterpunkten kann es auch sinnvoll sein, Gitterpunkte wieder aus einem Gitter zu entfernen. Ein Anwendungsfall, der für diese Arbeit relevant ist, sind

Gitter, die mit ähnlichen Datensätzen über mehrere Zeitschritte wiederverwendet werden. Durch entfernte Datenpunkte können Bereiche, in denen im letzten Zeitschritt viele Gitterpunkte notwendig waren, jetzt durch weniger Gitterpunkte ausreichend approximiert werden. Analog zum obigen Fall können hier Gitterpunkte mit Koeffizienten mit einem Wert nahe Null entfernt werden, da diese Gitterpunkte, aufgrund der kleinen Koeffizientenwerte, nur wenig bei der Auswertung der Dünngitterfunktion beitragen.

3. Zeitreihenanalyse mit dünnen Gittern

In diesem Kapitel wird ein Ansatz zum Lösen des Vorhersageproblems für Zeitreihen vorgestellt. Dafür wird zunächst der Begriff der Zeitreihenanalyse näher ausgeführt. Wie gezeigt wird, lässt sich die Vorhersage von Zeitreihen als Klassifikations- oder Regressionsproblem auffassen, wodurch eine Anwendung der in Kapitel 2 vorgestellten Algorithmen möglich wird. Zuletzt wird ein Verfahren zur Validierung der berechneten Modellfunktionen vorgestellt.

3.1. Grundlagen der Zeitreihenanalyse

Eine Zeitreihe ist eine Menge von zeitlich geordneten Beobachtungen [BJR08]. Die Zeitpunkte der Beobachtungen können hierbei äquidistant gewählt sein, es sind allerdings auch unregelmäßige Zeitschritte möglich. Im einfachsten Fall wird die Entwicklung einer einzelnen Größe über die Zeit betrachtet, beispielsweise eine Zeitreihe, die eine Bevölkerungsentwicklung über die Zeit repräsentiert. Eine eindimensionale Zeitreihe wird in dieser Arbeit wie folgt formal dargestellt:

$$T_1^{(n)} = \{(t_i, y_i) \in \mathbb{R} \times \mathbb{R} | \forall k \in \{0, \ldots, i-1\} : t_k < t_i\}_{i=0}^{n} \tag{3.1}$$

Eindimensionale Zeitreihen werden auch univariate Zeitreihen genannt, höherdimensionale Zeitreihen werden auch als multivariate Zeitreihen bezeichnet. Bei multivariaten Zeitreihen wird nicht nur eine einzelne Größe über die Zeit verfolgt, stattdessen werden mehrere Größen verfolgt. Das heißt, jeder Zeitschritt ist nicht mit einem einzelnen Wert verknüpft, sondern mit einem Tupel an Werten [BJR08]. Bei multivariaten Zeitreihen werden also mehrere univariate Zeitreihen zusammen als eine einzige multivariate Zeitreihe aufgefasst:

$$T_d^{(n)} = \{(t_i, \overline{z}_i) \in \mathbb{R} \times \mathbb{R}^d \wedge \forall k \in \{0, \ldots, i-1\} : t_k < t_i\}_{i=0}^{n} \tag{3.2}$$

Multivariate Zeitreihen werden häufig verwendet, wenn die Datenpunkte einer univariaten Zeitreihe mit zusätzlichen Informationen verknüpft werden können. Ein Beispiel hierfür ist eine Abfolge von gemessenen Gesten einer Hand über die Zeit. Eine univariate Zeitreihe könnte hierbei nur aus den Namen der Gesten über die Zeit bestehen. In Form einer multivariaten Zeitreihe könnten zusätzliche Informationen über die Haltung der Finger gegeben werden. Durch diese zusätzlichen Informationen kann die Vorhersage zukünftiger Werte erleichtert werden, da mehr kontextuelle Informationen vorliegen. Für eine Verbesserung der

Vorhersage ist es erforderlich, dass die zusätzlichen Informationen für die vorherzusagende Zielgröße relevant sind.

Zeitreihen spielen immer dann eine Rolle, wenn zeitabhängige Prozesse untersucht werden. Dafür sind Anwendungsfälle in nahezu allen wissenschaftlichen und wirtschaftlichen Bereichen denkbar:

- In der Ökonomie werden unter anderem Aktienkurse und Unternehmensprofite über die Zeit betrachtet.
- Natürliche Phänomene können interessante Zeitreihen ergeben. Zum Beispiel kann durch Messung der Temperatur über die Zeit das Phänomen der globalen Erwärmung untersucht werden. Die Entwicklung von regelmäßigen Wetterphänomenen wie El Niño sind ebenfalls ausgesprochen interessant.
- In der Medizin kann unter anderem der Verlauf der Aktivität unterschiedlicher Hirnareale über die Zeit mit Untersuchungsmethoden wie der funktionellen Magnetresonanztomographie oder einem EEG verfolgt werden.
- In der Linguistik sticht besonders das Problem der Spracherkennung heraus. Hier wird der Verlauf von Schallwellen über die Zeit untersucht.

Eine Untersuchung einiger der beschriebenen Anwendungfälle wurde von Robert H. Shumway und David S. Stoffer durchgeführt [SS10].

3.2. Das Vorhersageproblem für Zeitreihen

Das Vorhersageproblem für univariate Zeitreihen kann wie folgt beschrieben werden. An einem Zeitschritt t_{n+1} ist eine Menge von Werten an vergangenen Zeitschritten

$$P_1^{(n)} = \{(t_i, y_{i+1}) \in \mathbb{R} \times \mathbb{R} | \forall k \in \{0 \dots i-1\} : t_k < t_i\}_{i=0}^{n} \tag{3.3}$$

gegeben und es soll der Wert y_{n+2} vorhergesagt werden.

Man beachte, dass im Gegensatz zur Definition einer Zeitreihe jetzt der Wert y_{i+1} am Zeitpunkt t_i Teil des zugehörigen Tupels ist, und nicht mehr y_i. Durch die Verknüpfung des nächsten Werts mit dem aktuellen Zeitpunkt wird ein Zusammenhang zum nächsten Wert hergestellt, da die Tupel als Auswertungen einer unbekannten Vorhersagefunktion aufgefasst werden können. Da die Vorhersage nicht immer korrekt ist, muss weiter zwischen einem vorhergesagten Wert y^*_{n+2} und dem tatsächlichen Wert y_{n+2}, der am nächsten Zeitschritt gemessen wird, unterschieden werden. Im besten Fall gilt $y^*_{n+2} = y_{n+2}$, häufig weicht die Vorhersage jedoch vom korrekten nächsten Wert ab.

Im multivariaten Fall wird die gegebene Zeitreihe der Vorhersageaufgabe analog zum univariaten Fall modifiziert:

$$p_{d+1}^{(n)} = \{(t_i, \overline{x}_i, y_{i+1}) \in \mathbb{R} \times \mathbb{R}^d \times \mathbb{R} | \forall k \in \{0 \dots i-1\} : t_k < t_i\}_{i=0}^{n} \tag{3.4}$$

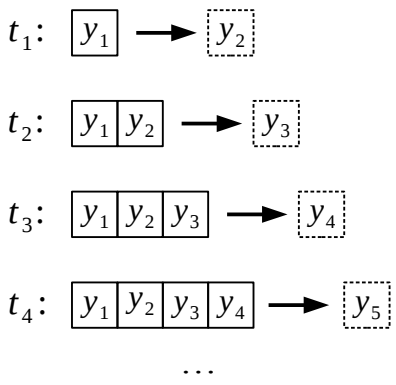


Abbildung 3.1.: Schema des iterierten Vorhersageproblems.

Dabei sind am Zeitpunkt t_{n+1} die weiteren Zeitreihen $\overline{x}$ gegeben und eine Vorhersage y^*_{n+2} für den Wert im nächsten Zeitschritt y_{n+2} wird gesucht. Gesucht ist also eine Vorhersage für den Wert einer bestimmten Zeitreihe.

Für das in dieser Arbeit betrachtete Szenario soll nicht nur ein einzelner Zeitschritt vorhergesagt werden, daher wird das iterierte Vorhersageproblem untersucht. Dabei fallen zu jedem Zeitschritt neue Werte $\overline{x}_{n+1}$ an und es wird eine Vorhersage y^*_{n+2} für den Wert y_{n+2} gesucht. Im nächsten Zeitschritt wird dann der tatsächliche Wert y_{n+2} gemessen und zusammen mit $\overline{x}_{n+1}$ als neuer Datenpunkt $(t_{n+1}, \overline{x}_{n+1}, y_{n+2})$ der Trainingsdatenmenge hinzugefügt:

$$p^{(n+1)}_{d+1} := P^{(n)}_{d+1} \cup \{(t_{n+1}, \overline{x}_{n+1}, y_{n+2})\} \tag{3.5}$$

Auf Basis der neuen Trainingsdatenmenge wird nun eine Vorhersage y^*_{n+3} für den noch unbekannten Wert y_{n+3} berechnet. Dieser Prozess kann wiederholt werden, solange weitere Beobachtungen möglich sind. Das in Abbildung 3.1 dargestellte Schema illustriert die Vorgehensweise für eine univariate Zeitreihe. Diese Vorgehensweise ist insbesondere bei Zeitreihen interessant, die kontinuierlich anfallen und in Echtzeit verarbeitet werden müssen. Beispiele hierfür sind medizinische Messdaten oder Börsendaten.

Gerade bei Börsendaten kann die Vorhersage zusätzlich mit Entscheidungsprozessen verknüpft sein. Ist dies der Fall, muss noch eine zweite zeitliche Komponente berücksichtigt werden. Es ist nicht nur notwendig y_{n+2} mit möglichst hoher Genauigkeit vorherzusagen, sondern der Wert für y_{n+2} muss schnell genug verfügbar sein, um eine Entscheidung noch rechtzeitig treffen zu können. Ein Beispiel für dieses Szenario ist die Vorhersage von Aktienkursen. Wenn ein bestimmter vorhergesagter Wert für eine bestimmte Aktie eine Einkaufs- oder Verkaufsaktion auslöst, dann sollte dieser Wert verfügbar sein, bevor die Aktie tatsächlich den Wert einnimmt, beispielsweise um Verluste zu vermeiden. Daher ist es sinnvoll, als obere Grenze für die erlaubte Rechenzeit der Vorhersage festzulegen, dass eine Rechenzeit von $\Delta t = t_{n+1} - t_n$ bei äquidistanten Zeitreihen nicht überschritten werden darf. Bei unregelmäßigen Zeitreihen sollte die Vorhersage spätestens eintreffen, bevor durchschnittlich ein

neuer Datenpunkt vorliegt[1]. Werden diese Kriterien eingehalten, kann von einer Vorhersage in Echtzeit gesprochen werden.

3.3. Zeitreihenanalyse als Data Mining Problem

Nachdem das Vorhersageproblem im letzten Abschnitt definiert wurde, kann jetzt eine Lösung des Problems vorgestellt werden. Es wurde bereits erwähnt, dass Mengen $P_{d+1}^{(n)}$ der Vorhersageprobleme als Auswertungen einer unbekannten Vorhersagefunktion $f : \mathbb{R} \to \mathbb{R}$ aufgefasst werden können, die einem Zeitpunkt eine Vorhersage y_{n+2}^* für den nächsten Zeitschritt zuordnet. Dies kann gleichzeitig als Regressionsproblem interpretiert werden, da bei Regressionsproblemen auf Basis einer Menge von Funktionauswertungen die zugrunde liegende Funktion rekonstruiert werden soll. Falls alle y_i aus einer endlichen Menge T stammen, kann das Vorhersageproblem weiter als Klassifikationsproblem betrachtet werden.

Eine erfolgreiche Vorhersage wird durch diese Vorgehensweise noch nicht erreicht, da die Zeitpunkte t_i nur eine Ordnung der Werte der Zeitreihe sicherstellen, allerdings selbst keine relevanten Informationen für den Verlauf der Zeitreihe darstellen. Damit prinzipiell erfolgreiche Vorhersagen berechnet werden können, muss t_i durch einen Wert oder ein Tupel von Werten ersetzt werden, das zum Zeitpunkt t_i verfügbar ist und gleichzeitig Werte beinhaltet, die für den nächsten Wert y_{i+1} relevant sind. Verfügbar sind hierfür alle vergangenen Werte der Zeitreihe und alle weiteren Zeitreihen, falls eine multivariate Zeitreihe betrachtet wird. Zu beachten ist, dass alte Werte der Zeitreihen nicht nur direkt verwendet werden können, sondern auch abgeleitete Werte, die sich aus Werten an vergangenen Zeitpunkten errechnen. Es bleibt zu betonen, dass die Wahl, welche Werte mit der Vorhersage zu einem Tupel des Regressionsproblems verknüpft werden, vom Problem abhängig ist und damit nicht allgemein geklärt werden kann. Es gibt jedoch einige Strategien, durch die bei vielen Zeitreihen eine erfolgreiche Vorhersage möglich wird.

Im einfachsten Fall kann als Tupel anstatt (t_i, y_{i+1}) das Tupel (y_i, y_{i+1}) verwendet werden. Wird das resultierende Regressionsproblem gelöst, ergibt sich eine Dünngitterfunktion, die auf Basis des aktuellen Wert y_n versucht, y_{n+1} vorherzusagen. Falls der Wert im letzten Zeitschritt eine zentrale Information für den Wert im nächsten Zeitschritt liefert, kann damit eine gute Vorhersage berechnet werden. Als Beispiel dient die folgende zyklische Zeitreihe:

$$5, 3, 7, 4, 9, 5, 3, 7, \ldots \tag{3.6}$$

Sind alle Werte des Zyklus Teil der Trainingsmenge, dann kann leicht eine Funktion angegeben werden, die korrekte Vorhersagen für zukünftige Zeitschritte berechnet. Allgemein können auch weiter in der Vergangenheit liegende Werte mit y_{n+1} verknüpft werden. Dabei ist es im allgemeinen sinnvoll, den nächsten Wert regelbasiert mit bestimmten, zum Beispiel m Schritte in der Vergangenheit liegenden Werten zu verknüpfen. Diese Strategie wird Delay-Embedding genannt [GGG10]. Einige konkrete Möglichkeiten vergangene Werte mit

[1] Für dieses Szenario sind auch strengere Kriterien denkbar.

dem nächsten Wert zum Zweck einer erfolgreichen Vorhersage zu verknüpfen, werden im nächsten Kapitel vorgestellt.

Durch Delay-Embedding entsteht die Trainingsmenge für das Regressionsproblem. Die neu eingefügten Komponenten der Tupel der Trainingsmenge liefern Informationen über den vorherzusagenden Wert der Zeitreihe und werden als Attribute bezeichnet. Die Abbildungen, die mit Werten aus den vergangenen Zeitschritten Attribute für das Regressionsproblem erzeugen, werden als Attributkonstruktoren bezeichnet. Der Raum, der durch Konstruktion der Attribute und der vorherzusagenden Zielgröße aufgespannt wird, wird Attributraum genannt.

Da die vorgestellten Verfahren für Regression und Klassifikation auf dünnen Gittern erfordern, dass die Attribute aus $[0,1]^d$ stammen, ist der Attributraum auf $[0,1]^d \times \mathbb{R}$ eingeschränkt. Dies ist keine wesentliche Einschränkung, da beliebige Attribute verwendet werden können, deren Werte reellen Zahlen zugeordnet werden können. Gegebenfalls müssen die Daten anschließend noch auf den passenden Hyperwürfel normalisiert werden.

Da in Abschnitt 2.3.3 ein Gitter ohne Rand gewählt wurde und generell zum Rand hin weniger Datenpunkte vorliegen, wäre eine direkte Normalisierung auf $[0,1]^d$ problematisch. Die Normalisierung würde sicherstellen, dass in Richtung jeder Dimension mindestens zwei Datenpunkte auf dem Rand liegen. Diese Datenpunkte könnten niemals optimal approximiert werden, da dies ein Gitter ohne Rand nicht erlaubt. Des Weiteren nimmt die Anzahl der Gitterpunkte zum Rand hin ab, und es wäre erforderlich einen sehr hohen Level zu wählen, falls viele Gitterpunkte in der Nähe des Rands liegen. Eine einfache Lösung für dieses Problem ist eine Normalisierung auf den inneren Würfel $[0.1, 0.9]^d$. Diese Wahl der Normalisierung hat sich im Rahmen dieser Arbeit bewährt.

3.4. Regularisierung und Validierung

Sowohl beim Data Mining mittels der Methode der kleinsten Quadrate als auch beim Dichtebasierten Data Mining ist es erforderlich, den der Regularisierung dienenden Parameter λ zu wählen. Dafür wird ein spezielles Verfahren vorgeschlagen, das es ausnützt, dass es sich bei der Vorhersage von Zeitreihen um ein iteriertes Problem handelt.

Jede Wahl von λ kann als ein neues Model für die gegebenen Beobachtungen betrachtet werden. Das heißt, für jeden Wert des Parameters λ muss getestet werden, ob dieser Wert eine gute Wahl im Verhältnis zu anderen möglichen Werten für λ darstellt. Um zu testen, ob ein Wert eine gute Wahl darstellt, bietet sich ein Validierungsschema an, dass als forward-chaining oder rolling-validation bekannt ist[2].

Dazu wird die Zeitreihe durchlaufen, wobei immer nur die vergangenen Beobachtungen bekannt sind und der nächste Wert vorhergesagt werden soll. Es werden also die vergangenen

[2]Der Ursprung des Verfahrens ist nicht vollständig klar, es wird unter anderem von Hu et al. [HZJP99] verwendet und ist im Data-Mining-Bereich häufig zu finden.

Vorhersagen aus obigem Algorithmus mit einem neuen λ nochmals nachvollzogen. Letztlich wird die Funktion für die Vorhersage ausgewählt, die im Fall der Klassifikation die höchste Trefferrate besitzt oder im Fall eines Regressionsproblems den niedrigsten durchschnittlichen quadratischen Fehler besitzt.

Dieses Verfahren hat das zentrale Problem, dass das Testen von neuen λ-Werten mit dieser Methode ausgesprochen teuer ist, da ein Testen mit allen Daten aus der Vergangenheit einen größeren Aufwand an Rechenzeit erfordert. Wird jedoch auf eine feste Menge von Lambdas zurückgegriffen, kann der benötigte Validierungsaufwand für jedes λ stark reduziert werden.

Es wird in jedem Zeitschritt für alle n Werte von λ eine Vorhersage für den nächsten Zeitschritt berechnet. Über alle Zeitschritte hinweg wird für jedes λ ein Zähler mitgeführt, der die Menge der erfolgreichen Vorhersagen im Klassifikationsfall zählt. Im Fall eines Regressionsproblems summiert der Zähler den Fehler an den einzelnen Zeitschritten auf. Auf Basis dieser Zähler kann für die Vorhersage im aktuellen Zeitschritt die Funktion ausgewählt werden, die in der Vergangenheit am meisten korrekte Vorhersagen ermöglichte, beziehungsweise die niedrigste Summe an Fehlertermen besitzt. In dieser Arbeit wurden folgenden Werte als Kandidaten für λ verwendet:

$$L = \{0, 0.1, 0.01, 0.001, 0.0001\} \tag{3.7}$$

Als zusätzliche Komplikation ergibt es sich, dass dieses Verfahren nur dann verwendet werden kann, wenn genügend Zeitschritte durchlaufen wurden, da die Zähler der einzelnen λ-Werte ansonsten nicht aussagekräftig sind. Diese Schwierigkeit kann mit einer Aufwärmphase gelöst werden. Hier ist zu bemerken, dass dieses Problem auch bei anderen Verfahren zur Wahl eines geeigneten Regularisierungsparameters existiert: Es spiegelt lediglich wieder, dass am Anfang des Durchlaufens einer Zeitreihe zu wenig Information zur Verfügung steht, um λ sinnvoll zu wählen. Für jede Art von aussagekräftiger Vorhersage am ersten Zeitschritt, müssen auch im ersten Zeitschritt bereits vergangene Beobachtungen vorliegen.

Durch dieses Verfahren ist es möglich, die Kosten für die Wahl des Regularisierungsparameters auf die einzelnen Zeitschritte zu verteilen. Die Menge der Testdaten für dieses Verfahren in jedem Zeitschritt ist jeweils nur ein Element, ähnlich wie bei einer Leave-One-Out Validierung [Biso6]. Allerdings bleibt die Regularisierung insgesamt relativ teuer, da immer nur für alle n Regularisierungsparameter in jedem Zeitschritt ein Data Mining Problem gelöst werden muss. Techniken zur weiteren Beschleunigung des Prozesses werden in Kapitel 6 besprochen. Durch die feste Wahl der Kandidaten für λ kann der Regularisierungsparameter nur dann optimal gewählt werden, wenn das optimale λ bereits in der Menge der Kandidaten enthalten ist. Da der optimale Wert schwierig im voraus zu bestimmen ist, bringt dies gewisse Nachteile in der Vorhersagequalität mit sich. In Kapitel 5 ist jedoch zu sehen, dass mit der gewählten Kandidatenmenge gute Vorhersagen möglich sind.

Ein weiteres Problem entsteht, wenn durch eine strukturelle Änderung des Attributraums für weitere erfolgreiche Vorhersagen ein anderer Wert für λ benötigt wird. Mit einer bestimmten Wahl des Regularisierungsparameters konnten bis zum aktuellen Zeitpunkt gute Vorhersagen getroffen werden, ab dem aktuellen Zeitschritt ist ein anderer Wert erforderlich.

Grundsätzlich werden die verwendeten Zähler für die Werte des Regularisierungsparameters nach einer gewissen Zeit die Änderungen im Attributraum widerspiegeln. Falls der Vorhersageprozess allerdings schon sehr lange läuft, kann es lange dauern, bis wieder gute Vorhersagen berechnet werden. Falls sich der Attributraum häufig strukturell ändert, können grundsätzlich keine guten Vorhersagen getroffen werden. Um dieses Problem abzumildern ist es notwendig eine feste Anzahl vergangener Vorhersagen für die Zähler zu berücksichtigen. Für Vorhersagen, die zu weit zurückliegen, kann dazu der auf den Zähler addierte Wert wieder abgezogen werden. Da bei den Experimenten in dieser Arbeit keine Vorhersagen über besonders große Zeiträume stattfanden, trat dieses Problem bei den durchgeführten Experimenten nicht auf.

3.5. Der gesamte Algorithmus

Mithilfe der Überlegungen auf diesem Kapitel kann der vollständige Algorithmus zur Vorhersage von Zeitreihen angegeben werden. In Algorithmus 3.1 ist dies für die Methode der kleinsten Quadrate dargestellt. Der Ablauf für den ersten Zeitschritt wird im Folgenden beschrieben.

Vor Beginn des eigentlichen Vorhersageprozesses muss die Trainingsdatenmenge aus den gegebenen Zeitreihen erzeugt werden. Dann beginnt der eigentliche Vorhersageprozess für die einzelnen Zeitschritte.

Dafür wird als Erstes der aktuelle Wert der Zeitreihen gemessen und in den Attributraum übersetzt. Anschließend wird mit der Methode der kleinsten Quadrate eine Dünngitterfunktion für jeden der gewählten Werte von λ berechnet. Mit jeder der berechneten Funktionen wird eine Vorhersage für den nächsten Zeitschritt berechnet und gespeichert. Als Vorhersage für den aktuellen Zeitschritt wird dabei die Vorhersage gewählt, deren Zähler den größten Wert einnimmt. An dieser Stelle kann die Vorhersage für beliebige Aufgaben verwendet werden.

Dann wird auf den nächsten Zeitschritt gewartet. Sobald eine neue Beobachtung gemacht werden kann, werden die Vorhersagen ausgewertet. Dafür wird der tatsächliche nächste Wert der Zielgröße gemessen und die Zähler für die Regularisierungsparameter erhöht, die eine korrekte Vorhersage berechnet haben. Zuletzt wird mit der gemessenen Klasse ein neues Trainingstupel erzeugt und der Trainingsmenge hinzugefügt. Dies wird für die gewünschte Anzahl von Zeitschritten wiederholt.

Der Algorithmus für das Dichte-basierte Verfahren ist dem Algorithmus 3.1 sehr ähnlich. Hier muss lediglich etwas zusätzlicher Verwaltungsaufwand betrieben werden, da nicht nur eine Dünngitterfunktion für jeden Wert von λ erzeugt wird, sondern so viele, wie Klassen vorliegen. Das heißt, die Funktion `CreateSparseGridFunction` wird mehrere Funktionen zurückgeben und die Funktion `EvaluateFunction` verwendet alle Dünngitterfunktionen als Eingabe und klassifiziert das Attributtupel wie in Abschnitt 2.5 beschrieben.

Algorithmus 3.1 Der gesamte Algorithmus

```
timeseries ← „Initiale Messungen“
procedure MakePredictions(timeseries, testSteps, lambdaSet)
    trainingSet ← CreateInitialTrainingSet(timeseries)
    ▷ Vorhersagen der nächsten testSteps Zeitschritte
    for t = 1 → testSteps do
        observation ← GetNextObservation()
        ▷ Beobachtung in Attributraum übersetzen
        attributeTuple ← MapToAttributeSpace(timeseries, observation)
        predictions ← MakeHashMap()
        ▷ Eine Vorhersage für jedes λ berechnen
        for λ ∈ lambdaSet do
            f ← CreateSparseGridFunction(trainingSet, λ)
            value ← EvaluateFunction(f, attributeTuple)
            class ← MapToClass(value)
            predictions[λ] ← class
        end for
        ▷ Das λ mit dem höchsten Zähler wählen
        bestLambda ← GetBestLambda(counters, lambdaSet)
        bestPrediction ← predictions[bestLambda]
        ▷ Verwenden der besten Vorhersage ...
        WaitUntilNextObservation()
        actualClass ← MeasureActualClass()
        ▷ Vorhersagen auswerten und Zähler anpassen
        for λ ∈ lambdaSet do
            if predictions[λ] = actualClass then
                counters[λ] ← counters[λ] + 1
            end if
        end for
        trainingSetTuple ← AddToTuple(attributeTuple, actualClass)
        trainingSet ← trainingSet ∪ {trainingSetTuple}
    end for
end procedure
```

4. Die Konstruktion von Attributräumen

Im letzten Kapitel wurde beschrieben, wie das Vorhersageproblem für Zeitreihen mit Data Mining Methoden gelöst werden kann. Dabei wurde als zentrale Strategie die Technik des Delay-Embeddings beschrieben. Die in dieser Arbeit verwendeten Attributkonstruktoren wurde jedoch offen gelassen. Dies wird in diesem Kapitel nachgeholt. Die Konstruktion von Attributräumen ist Teil der Vorverarbeitung des in Abschnitt 2.2 beschriebenen Knowledge Discovery Prozesses. Damit die Regressions- oder Klassifikationslösung erfolgreiche Vorhersagen erlaubt, sind neben der Wahl der Attributkonstruktoren noch weitere Aspekte der Datenvorverarbeitung zu berücksichtigen. Diese werden ebenfalls in diesem Kapitel behandelt.

4.1. Das allgemeine Verfahren

Alle in dieser Arbeit verwendeten Attributkonstruktoren sind Abbildungen, die an jedem Zeitschritt, an dem die Zielgröße existiert, den aktuellen Zeitschritt regelbasiert mit Werten aus der Vergangenheit verknüpfen. Dieses Vorgehen wird in diesem Abschnitt formalisiert.

Um einen Attributkonstruktor anwenden zu können, muss zunächst die Zielgröße an jedem Zeitschritt bestimmt werden. Es kann vorkommen, dass dies nicht an jedem Zeitschritt möglich ist. Wird als Zielgröße beispielsweise der Durchschnitt aus den letzten vier Werten und dem nächsten Wert gewählt, dann kann an den drei ersten Zeitschritten der Durchschnitt nicht berechnet werden. In dieser Arbeit wurde als Zielgröße nur der Trend im nächsten Zeitschritt verwendet, also die Information, ob der Wert der Zeitreihe steigt oder fällt. Da der Trend mit einem Rückgriff auf den letzten Zeitschritt berechnet wird, kann es auch hier vorkommen, dass in der Zeitreihe existierende Zeitschritte nicht als Teil des Data Mining Problems verwendet werden können.

Nachdem die Zeitreihe der Zielgrößen aus dem ursprünglichen Datensatz berechnet wurde, können die Attributkonstruktoren für jeden Zeitschritt angewendet werden. Alle Attributkonstruktoren, die in dieser Arbeit verwendet wurden, greifen auf eine festgelegte Anzahl von Schritten in die Vergangenheit zurück. Dabei müssen nicht alle vergangenen Zeitschritte in diesem Intervall für die Attributkonstruktion berücksichtigt werden. Im Rahmen dieser Arbeit wird die zusätzliche Anforderung gestellt, dass Attribute jeweils nur aus einer der Zeitreihen des Datensatzes erstellt werden können. Für Werte $z^{(j)}$ am Zeitschritt j aus einer Zeitreihe Z des Datensatzes, stellt beispielsweise die folgende Menge ein mögliches Attribut dar:

$$\{(t_j, (z^{(j)}, z^{(j-3)}, z^{(j-5)}, z^{(j-6)}))|\forall j \in \{7 \dots n\}\} \tag{4.1}$$

Um die Schemata geschickt angeben zu können, werden alle Zeitschritte, auf die Bezug genommen wird, relativ zum aktuell betrachteten Zeitschritt spezifiziert.

Um diese Überlegung zu formalisieren wird ein Indextupel

$$I_A \in \{(\bar{i} | \forall l \in \mathbb{N} : \forall k \in \{1, \ldots, l\} : i_k \in \{0, \ldots, n\}\} \tag{4.2}$$

benötigt, das die Differenzen zum aktuell betrachteten Zeitschritt beschreibt. Im obigen Beispiel wäre dies das Tupel $(0, 3, 5, 6)$. Um die Attribute zu erzeugen, können als Zwischenschritt die Wertetupel an den einzelnen Zeitschritten aufgeschrieben werden als

$$A' = \{(t_j, ((z^{(j-i_0)}, \ldots, z^{(j-i_m)}))) | \forall k \in \{1 \ldots m\} : i_k \in I_A \wedge z^{(j-i_k)} \in Z\}_{j=0}^{n}. \tag{4.3}$$

Mit den Wertetupeln an den Zeitschritten kann aus A' das eigentliche Attribut berechnet werden. Dafür muss noch eine Abbildung C mit $|I_A|$ als Dimension der Urbildmenge ausgewählt werden. Ist eine gewünschte Abbildung C gewählt, ergibt sich daraus das Attribut

$$A = \{(t_j, C(\overline{z}_j)) | (t_j, \overline{z}_j) \in A'\}_{j=0}^{n}. \tag{4.4}$$

Die Dimension des Attributs $|A|$ wird definiert durch die Dimension der Bildmenge der Abbildung C.

Die Zielgröße kann als ein weiteres Attribut betrachtet werden, wobei zur Berechnung der Zielgröße auch Werte am Zeitschritt $j + 1$ benutzt werden, das zugehörige Indextupel ist also

$$I_Y \in \{(\bar{i} | \forall k \in \{1, \ldots, l\} : i_k \in \{0, \ldots, n + 1\}\} \tag{4.5}$$

Außerdem ist der Bildraum des Attributs auf $\mathbb{R}$ eingeschränkt. Nach Auswahl der Zeitschritte folgt daraus

$$Y' = \{(t_j, (z^{(j-i_0)}, \ldots, z^{(j-i_m)})) | i_k \in I_Y\}_{j=0}^{n}. \tag{4.6}$$

Woraus durch eine weitere Abbildung C_Y die Zeitreihe der Zielgröße entsteht:

$$Y = \{(t_n, C_Y(\overline{z})) | (t_j, \overline{z}_j) \in Y'\} \tag{4.7}$$

Mit dem Konkatenationsoperator für Tupel $\circ$, den Attributen $A_1 \ldots A_m$ und der Zeitreihe der Zielgröße Y kann schließlich die Trainingsmenge für das Data Mining Problems angegeben werden:

$$P_{d+1}^{(n)} = \{a_2^{(1)} \circ \cdots \circ a_2^{(m)} \circ y_2 | \forall k \in \{1 \ldots m\} : a^{(k)} \in A_k \wedge a_1^{(k)} = t_j \wedge y \in Y \wedge y_1 = t_{j+1}\}_{j=0}^{n} \tag{4.8}$$

Über $a_1^{(j)}$ und $a_2^{(j)}$ werden auf den Zeitschritt t und das Wertetupel $\overline{z}$ der Tupel eines Attributs j zugegriffen. Die Gesamtdimension des Attributraums ergibt sich aus der Summe der

Dimensionen der Attribute, zuzüglich einer Dimension für die Zielgröße. Die Dimension der Attribute wird berechnet mit

$$d = |A_1| + |A_2| + \cdots + |A_m|. \tag{4.9}$$

Um also die Trainingsmenge aufzustellen, müssen die einzelnen Attribute festgelegt werden. Zusätzlich wird die Zielgröße aufgestellt, die, wie oben beschrieben, als spezielles Attribut behandelt werden kann. Jedes Attribut ist durch eine Indexmenge I und zugehörige Abbildung C gegeben. Mit der Indexmenge werden die Zeitschritte relativ zum aktuellen Zeitschritt aus einer Zeitreihe ausgewählt, aus denen ein Attribut errechnet wird. Die Abbildung berechnet aus den einzelnen Werten das resultierende Attribut. Zuletzt kann aus den Attributen und der Zeitreihe der Zielgröße wie oben beschrieben der Attributraum $P_{d+1}^{(n)}$ berechnet werden.

4.2. Attributkonstruktoren

Um ein Attribut zu erzeugen, muss, wie im letzten Abschnitt beschrieben wurde, zunächst eine Zeitreihe ausgewählt werden. Dann kann ein Attribut durch Vorgabe eines Indextupels und einer Abbildung angegeben werden. In diesem Abschnitt werden die im Rahmen der Experimente dieser Arbeit verwendeten Strategien zum erzeugen von Attributen ausgeführt.

4.2.1. Der Linear-Konstruktor und dessen quadratische Variante

Der einfachste Attributkonstruktor ist der Linear-Konstruktor. Dieser verwendet als Abbildung die Identität. Das heißt, er berechnet die über ein Indextupel vorgegebenen Werte an den vergangenen Zeitschritten und gibt diese als Tupel direkt zurück. Dieser Konstruktor kann verwendet werden, um zum Beispiel den Wert am aktuellen Zeitpunkt mit dem Wert am letzten Zeitpunkt zu verknüpfen. Chancen auf erfolgreiche Vorhersagen bestehen, wenn bekannte Kombinationen an Werten eine Aussage für den nächsten Wert erlauben. Zum Beispiel bei Zeitreihen mit unbekanntem, aber periodischem Verlauf kann dieser Ansatz erfolgsversprechend sein.

Für die Wahl der Indextupel wird eine weitere Restriktion eingeführt. Um Experimente zu vereinfachen können nicht beliebige Tupel ausgewählt werden, sondern es kann lediglich eine Zahl m gewählt werden, die bestimmt, wie viele vergangene Zeitschritte berücksichtigt werden. Ein Indextupel hat also die Form $(1, 2, 3, \ldots, m)$. Der Parameter m wird Schrittanzahl genannt.

Als Variante dieses Konstruktors kann dieselbe Abbildung mit einer quadratisch wachsenden Schrittweite verwendet werden. Das heißt, es werden Tupel $(t_i, t_{i-1}, t_{i-4}, t_{i-9})$ hinzugefügt. Wird die quadratische Variante verwendet, dann wird der Attributkonstruktor als Quad-Attributkonstruktor bezeichnet. Dieser Konstruktor ist nützlich, wenn sowohl kurz- als auch langfristige Strukturen in den Daten eine Rolle für die Vorhersage spielen.

Dieser Attributkonstrukor hat einen starken Nachteil, der ihn für viele Datensätze unbrauchbar macht. Wenn sich Werte der Datenpunkte nicht wiederholen, dann wird die Trefferquote mit diesem Konstruktor äußerst niedrig ausfallen. Zum Beispiel weil die unbekannte Funktion, die mit der Zeitreihe angenähert werden soll, einer Geraden mit Steigung ungleich null entspricht. Neue Werte in der Zeitreihe bedeuten dann, dass mit diesem Konstruktor eine Auswertung am aktuellen Zeitschritt an einem Datenpunkt x_n stattfindet, in dessen Nähe es keine oder nur wenige Datenpunkte geben kann, da ähnliche Werte bei einer monoton steigender oder fallenden Funktion nicht vorliegen können. Eine Abhilfe für diese Problematik stellt der nächste Konstruktor dar.

4.2.2. Der Differenz-Konstruktor und dessen quadratische Variante

Der Differenz-Konstruktor verwendet Differenzen zwischen dem aktuellen Werten einer Zeitreihe z_n und Werten aus der Vergangenheit. Damit können Tupel der Form

$$(z_n - z_{n-1}, z_n - z_{n-2}, \dots) \tag{4.10}$$

für äquidistante Zeitreihen erzeugt werden. Für unregelmäßige Zeitreihen muss der jeweilige Abstand der Zeitreihen berücksichtigt werden, was zu Tupeln der Form

$$\left(\frac{z_n - z_{n-1}}{t_n - t_{n-1}}, \frac{z_n - z_{n-2}}{t_n - t_{n-2}}, \dots\right) \tag{4.11}$$

führt. Durch diese Wahl der Konstruktion der Datenpunkte des Attributraums ist es möglich relativ betrachtet ähnliche Kombination von Werten zu berücksichtigen. Dies löst insbesondere das im letzten Abschnitt beschriebene Problem, dass vollständig neue Werte in der Zeitreihe auftreten, wodurch neue Bereiche im Attributraum erschlossen wurden. Mit diesem Konstruktor ist es nur noch erforderlich, dass ähnliche genäherte Ableitungswerte vorliegen.

Analog zum Linear-Konstruktor existiert auch bei diesem Konstruktor eine quadratische Variante, die Differenz-Quad-Konstruktor genannt wird. Auch hier wird wieder modelliert, dass Änderungen sowohl über kleine, als auch über größere zeitliche Abschnitte relevant sein können. Dieser Konstruktor erlaubt beides zu betrachten, ohne dass die Gesamtdimension zu groß wird. Tupel haben dann die folgende Form:

$$\left(\frac{z_n - z_{n-1}}{t_n - t_{n-1}}, \frac{z_n - z_{n-4}}{t_n - t_{n-4}}, \frac{z_n - z_{n-9}}{t_n - t_{n-9}}, \dots\right) \tag{4.12}$$

4.3. Über die Wahl der Datenpunkte

Durch die Attributkonstrukoren werden Informationen, die in der Zeitreihe versteckt sind, so dargestellt, dass sie für die Formulierung als Regressions- oder Klassifikationsproblem nützlich sind. Bestimmte Wertekombinationen der erzeugten Attribute signalisieren dann

über das Data Mining Problem, welcher Wert am nächsten Zeitpunkt zu vermuten ist. Dabei gibt es jedoch zwei Schwierigkeiten bei der Wahl der Datenpunkte.

Zum einen können leicht mehr Datenpunkte als eigentlich notwendig gewählt werden. Wenn bereits alle relevanten Bereiche des Dünngitterraums über die Wahl der aktuellen Datenpunkte der Zeitreihe für das gestellte Problem hinreichend genau modelliert wurden, dann sind zusätzliche Datenpunkte unnötig. Zudem führt die Verwendung von nicht benötigten Datepunkten dazu, dass sich der Bedarf an Rechenzeit erhöht. Bei Data Mining nach der Methode der kleinsten Quadrate benötigt das Aufstellen der Matrix $O(mn)$ Operationen, wobei m für die Anzahl der Datenpunkte und n für die Anzahl der Gitterpunkte steht. Beim Dichte-basierten Ansatz findet sich ebenfalls eine Komplexität von $O(mn)$ bei der Auswertung der rechten Seite des zugehörigen Gleichungssystems. Es muss allerdings betont werden, dass das Lösen des linearen Gleichungssystems den größten Teil der Zeit beansprucht.

Deutlich problematischer ist ein zweites Phänomen. Angenommen die aktuelle Wahl der Attributkonstrukoren erlaubt eine gute Vorhersage der Zeitreihe. Das heißt, die wichtigsten Aspekte der Zeitreihe werden durch die Datenpunkte korrekt wiedergegeben. Sollten jetzt neue Prozesse in der Zeitreihe auftreten, wird die Vorhersage stark an Genauigkeit verlieren. Durch das weitere kontinuierliche Hinzufügen neuer Datenpunkte in jedem Zeitschritt wird die Genauigkeit mit der Zeit wieder zunehmen. Allerdings kann es, ähnlich wie im Kontext der Regularisierung ausgeführt, lange dauern, bis die Datenpunkte mit jetzt veralteten Werten keine Rolle mehr für die Vorhersage spielen.

Um die beiden obigen Probleme zu minimieren, ist es notwendig die Anzahl der Datenpunkte in der Vergangenheit zu limitieren. Hierfür wird ein einfaches Verfahren gewählt, das als Sliding-Window bezeichnet wird. Dabei werden für die Vorhersage des nächstens Zeitschritts immer die letzten m Datenpunkte verwendet. Die Anzahl der zu berücksichtigenden Datenpunkte wird als Fenstergröße bezeichnet. Es muss an dieser Stelle angemerkt werden, dass nicht nur ein zu großes Fenster zu Problemen bei der Vorhersage führt. Werden zu wenig Datenpunkte verwendet, dann werden selbst bei der Wahl korrekter Attributkonstruktoren, die alle relevanten Informationen verfügbar machen, nicht alle benötigten Muster im Attributraum des Data Mining Problems repräsentiert. In diesem Fall sinkt die Qualität der Vorhersage ebenfalls stark.

4.4. Möglichkeiten der Datenvorverarbeitung

Durch die Darstellung der Zeitreihenanalyse als Data Mining Problem unterliegt diese Form der Zeitreihenanalyse dem üblichen Data-Mining-Prozess, wie er in Abschnitt 2.2 vorgestellt wurde. Das bedeutet, zur Anwendung des vorgestellten Verfahrens müssen die folgenden Schritte durchgeführt werden:

- Vorverarbeitung der Daten: Dieser Schritt ist Weiter unterteilt in Auswahl der Daten, Vorverarbeitung und Transformation der Daten. Nach Abschluss der Vorverarbeitung steht ein Attributraum für die Data-Mining-Algorithmen bereit.

- Dem eigentlichen algorithmischen Data Mining mit den vorgestellten Verfahren.
- Zuletzt werden die Ergebnisse interpretiert.

Dabei sind die drei Vorverarbeitungsschritte tendenziell manuelle Prozesse. Ebenso die Interpretation der Ergebnisse. Wie bei anderen Data Mining Ansätzen auch, ist die richtige Vorverarbeitung essenziell für die erreichbare Qualität der Ergebnisse. Außerdem erlaubt eine umsichtige Vorverarbeitung eine einfachere Interpretation der Ergebnisse. Auf der anderen Seite führt eine ungünstige Vorverarbeitung dazu, dass die Ergebnisse keine Aussagekraft besitzen. Falls schlechte Resulate gemessen werden, ist es zudem schwer zu sagen, ob die Vorverarbeitung der Daten zu schlecht gewählt war oder ob die Qualität der Daten ein Grund für eine unerwartet schlechte Qualität der Ergebnisse ist. Aus diesen Gründen ist es äußerst nützlich, wenn im Zuge der Vorverarbeitungsschritte bereits möglich viel Wissen über die Daten vorliegt, um möglichst gute Ergebnisse zu erreichen.

Im Rahmen des vorgestellten Ansatzes bestehen die folgenden Optionen für die Vorverarbeitung der Daten:

1. Wahl einer geeigneten Zielgröße
2. Entscheidung aus einem d-dimensionalen Datensatz für die Datenreihen, die relevant für die gewählte Zielgröße sind
3. Wahl eines oder mehrerer geeigneter Konstruktoren für jede gewählte Zeitreihe, um die Daten in für die Zielgröße relevante Informationen zu übersetzen
4. Konfiguration der ausgewählten Konstruktoren, zum Beispiel durch Vorgabe der Schrittanzahl
5. Normalisieren der finalen Datenpunkte des Attributraums
6. Wahl der Fenstergröße
7. Wahl geeigneter Kandidaten für den Regularisierungsparameter λ
8. Wahl der Anzahl der Iterationen (alternativ des Werts der Fehlernorm) des verwendeten iterativen Lösers
9. Wahl des Levels für das dünne Gitter, auf dem das Data Mining stattfindet
10. Optional: Auswählen, wie viele Verfeinerungsvorgänge und Vergröberungsvorgänge in jedem Zeitschritt durchgeführt werden sollen.

Die Wahl dieser Optionen beeinflusst die Qualität der Zeitreihenanalyse entscheidend, wie in Kapitel 5 an vielen Beispielen gezeigt wird. Auf zwei spezielle Schwierigkeiten, die speziell im Kontext einer Zeitreihenanalyse mit Gitter-basierten Ansätzen auftreten, muss noch separat eingegangen werden.

4.5. Dimension des Dünngitterraums und Anzahl der Datenpunkte

Die Gesamtdimension des Dünngitterraums entspricht der Summe der Dimensionen der einzelnen Attributkonstruktoren, zuzüglich einer weiteren Dimension für die vorherzusagende Zielgröße. Steigt die Dimension, dann steigt auch das Volumen des Dünngitterraums exponentiell. Wenn mit steigender Dimension nicht gleichzeitig die Anzahl der Datenpunkte erhöht wird, dann liegen die Datenpunkte immer dünner im Dünngitterraum. Im Allgemeinen werden die Punkte auch nicht gleichmäßig im Raum verteilt liegen, da die Werte der einzelnen Attribute nicht gleichmäßig verteilt auftreten. Dieses Phänomen hat mehrere Konsequenzen.

Zum einen motiviert dies die Verwendung eines dünnen Gitters anstatt eines voll besetzten Gitters, da bei voll besetzten Gittern die Anzahl der Gitterpunkte mit zunehmender Dimension deutlich schneller wächst (siehe 2.3.3). Das heißt, die maximal wählbare Anzahl an Attributen wird durch die Verwendung von dünnen Gittern deutlich verbessert. Zum anderen wird gerade bei höherer Dimension die Verwendung von adaptiver Verfeinerung und adaptiver Vergröberung wichtiger, da der Raum, bezogen auf das Volumen, durch weniger Datenpunkte überdeckt ist. Mittels adaptiver Verfeinerung kann sichergestellt werden, dass lokal eine genügend hohe Auflösung an Gitterpunkten vorliegt. Ohne die Verwendung von adaptiver Verfeinerung kann die Vorhersagegenauigkeit sonst selbst dann fallen, wenn die zusätzlich eingeführten Attribute tatsächlich relevant für die vorherzusagende Größe sind.

Eine weitere Schwierigkeit bei der Datenvorverarbeitung besteht darin, dass durch die Wahl unpassender Konstruktoren die Vorhersagegenauigkeit stark fallen kann. Naiv könnte vermutet werden, dass zusätzlich Attribute lediglich zusätzliche Informationen liefern würden und im schlechtesten Fall keine zusätzlichen Informationen liefern. Dann würde zwar das Problem in einer höheren Dimension als eigentlich notwendig gelöst und die Berechnung der Lösung würde mehr Rechenzeit in Anspruch nehmen, aber in Bezug auf die Vorhersagequalität würde sich jedoch kein Unterschied ergeben.

Diese Vermutung ist leider falsch, falls die Anzahl der Datenpunkte gleich bleibt. Man stelle sich ein zur Zielgröße unkorreliertes Attribut vor. Ein solches Attribut kann zum Beispiel durch eine Zufallsvariable modelliert werden. In Bezug auf die Datenpunkte des Dünngitterraums bedeutet dies, dass eine Koordinate in jedem Datenpunkt zufällig variiert ist. Wird nun eine konkrete Vorhersage berechnet und existiert ein ähnlicher Datenpunkt, dann sind alle Koordinaten des Datenpunktes ähnlich zu den aktuell gemessenen Daten. Die eine unkorrelierte Koordinate wird jedoch beliebig weit abweichen. Dadurch steigt die Distanz zwischen den aktuellen Daten und dem Datenpunkt beliebig weit an, wodurch die Qualität der Vorhersage grundsätzlich beliebig schlecht werden kann.

Damit trotz eines unkorrelierten Attributs noch sinnvolle Voraussagen errechnet werden können, müssen für alle Werte des unkorrelierten Attributs genügend Datenpunkte vorliegen, damit passende Datenpunkte, also Datenpunkte die in den anderen Attributen ähnlich sind, für beliebige Werte des unkorrelierten Attributs vorliegen. Falls die Anzahl der Datenpunkte also deutlich vergrößert werden kann oder falls das unkorrelierte Attribut nur eine kleine Wertemenge besitzt, wird die Vorhersagequalität nicht allzu stark negativ betroffen sein.

Falls allerdings die Anzahl der Datenpunkte gleich bleibt und die Wertemenge des Attributs groß ist, dann ist mit einer deutlichen Verschlechterung der Vorhersagen zu rechnen. Dieses Problem kann durch die Auswahl geeigneter Attributkonstrukoren umgangen werden.

5. Datensätze und Vorhersagequalität

In den letzten Kapiteln wurde ein Ansatz zur Zeitreihenanalyse auf dünnen Gittern vorgestellt. Um zu zeigen, dass diese Algorithmen auch tatsächlich für Vorhersagen genutzt werden können, wurde eine Untersuchung der Qualität der Vorhersagen mit mehreren Experimenten durchgeführt. In diesem Kapitel werden die einzelnen Datensätze der Experimente vorgestellt und die durchgeführten Experimente ausgewertet.

Insgesamt wurden fünf Datensätze betrachtet, von denen zwei synthetisch generiert wurden. Mit den synthetischen Datensätzen soll gezeigt werden, dass der vorgestellte Ansatz bei einer geeigneten Vorverarbeitung der Daten und der Verfügbarkeit von Daten mit hoher Qualität eine hohe Trefferrate korrekter Vorhersagen ermöglicht. Des Weiteren wurde der vorgestellte Ansatz mit drei nichtsynthetischen, auf Finanzdaten basierten Datensätzen untersucht. Durch die Experimente mit nichtsynthetischen Datensätzen soll gezeigt werden, dass auch bei realen Daten korrekte Vorhersagen möglich sind. Dabei erschweren die Wahl einer geeigneten Vorverarbeitung und eine geringere Qualität der Datensätze ähnlich gute Ergebnisse, wie sie bei den synthetischen Daten vorliegen.

Die vorherzusagende Zielgröße ist bei allen Experimenten dieselbe. Es wird vorhergesagt, ob die Zeitreihe im nächsten Schritt steigen wird oder ob sie fällt. Diese Art der Vorhersage wird Trendvorhersage genannt. Das Zuordnen des korrekten Trends zu den Werten an den Zeitpunkten ist dabei Teil der Vorverarbeitung. Zeitschritte, bei denen der Wert der Zeitreihe gleich bleibt, werden nicht als Teil des Vorhersageproblems betrachtet. Das heißt, wenn die Steigung der Zeitreihe positiv ist, dann wird versucht dies vorherzusagen. Wenn die Steigung der Zeitreihe negativ ist, dann wird ebenfalls versucht dies vorherzusagen. Wenn die Zeitreihe weder steigt noch fällt, ist die Vorhersage irrelevant. Dabei wird berücksichtigt, dass sich die Trefferrate aus dem Quotienten der korrekten Vorhersagen und den für die Vorhersage gültigen Zeitschritten zusammensetzt. Als weiterer Vorverarbeitungsschritt wurden alle Datenpunkte der Attributräume auf $[0.1, 0.9]^d \times \mathbb{R}$ normiert[1]. Diese Vorverarbeitungsschritte finden bei jedem Datensatz statt.

In Abschnitt 4.4 wurden die möglichen Optionen zur Datenvorverarbeitung beschrieben. Um die Anzahl der Experimente auf einen auswertbaren Rahmen zu reduzieren, wurden einige Parameter fest gewählt:

- Als Zielgröße dient immer der Trend.
- Experimente, bei denen kein Level angegeben wurde, wurden auf einem Gitter mit Level 6 berechnet.

[1] Die Zielgröße muss nach wie vor nicht normiert werden.

- Die Anzahl der Iterationen des Lösers wurde auf 15 Iterationen beschränkt. Falls die Fehlerschranke, hier festgelegt mit 10^{-5}, schneller erreicht wird, werden weniger Iterationen verwendet.
- Die Fenstergröße, das heißt die Anzahl der relevanten Daten aus der Vergangenheit, wurde mit 3000 Zeitschritten gewählt. Daraus folgt, dass für die Erstellung einer Dünngitterfunktion immer die letzten 3000 Datenpunkte berücksichtigt werden, jeweils vom aktuellen Zeitschritt aus betrachtet.
- Da die Ergebnisse des Dichte-basierten Ansatzes und die Ergebnisse des Ansatzes mit der Methode der kleinsten Quadrate sehr ähnlich sind, wird hier nur auf die Methode der kleinsten Quadrate eingegangen. Ergebnisse für den Dichte-basierten Ansatz sind jedoch in Anhang A.1 zu finden.

5.1. Der synthetische Datensatz „Kurve“

Die Durchführung synthetischer Experimente dient der experimentellen Überprüfung des in den letzten Kapiteln präsentierten Ansatzes. In Fällen, in denen alle relevanten Informationen über die Daten vorliegen und die Daten zudem so beschaffen sind, dass auffindbare Muster auch tatsächlich existieren und mit den vorgestellten Attributkonstruktoren extrahierbar sind, muss eine hohe Vorhersagequalität erreichbar sein. Dies wird mit diesem Datensatz und dem im nächsten Abschnitt folgenden Datensatz experimentell überprüft.

In diesem ersten synthetischen Datensatz wird eine bekannte Funktion an diskreten Zeitpunkten ausgewertet. Als einfaches Untersuchungsobjekt wurde eine Summe von mehreren Sinusfunktionen gewählt, wobei die einzelnen Funktionen kurz- und langfristige Zyklen in den Daten darstellen sollen:

$$f_{sin}(t) = \sum_{i=0}^{m} a_i sin\left(\frac{2\pi t}{b_i}\right) + ct \tag{5.1}$$

Durch die Multiplikation von t mit 2π werden die Sinusfunktionen auf eine Periode von 1 normiert. Durch den Parameter b_i werden beliebige Perioden erzeugt, mit den Parametern a_i werden die Amplituden gesteuert. Der Parameter c fügt eine zusätzliche Gerade hinzu, durch die eine vollständige Periodizität der Werte der Funktion verhindern soll, wenn c ungleich Null ist. Erhalten bleibt allerdings auch bei einer Wahl von c ungleich Null eine Periodizität in den Ableitungen, die interessant für die Vorhersage der Funktion ist.

Für die Untersuchung in diesem Abschnitt wird die Funktion f_{sin} mit den folgenden Parametern gewählt:

$$\overline{a} = (17, 11) \tag{5.2}$$

$$\overline{b} = (97, 29) \tag{5.3}$$

$$c = 0.05 \tag{5.4}$$

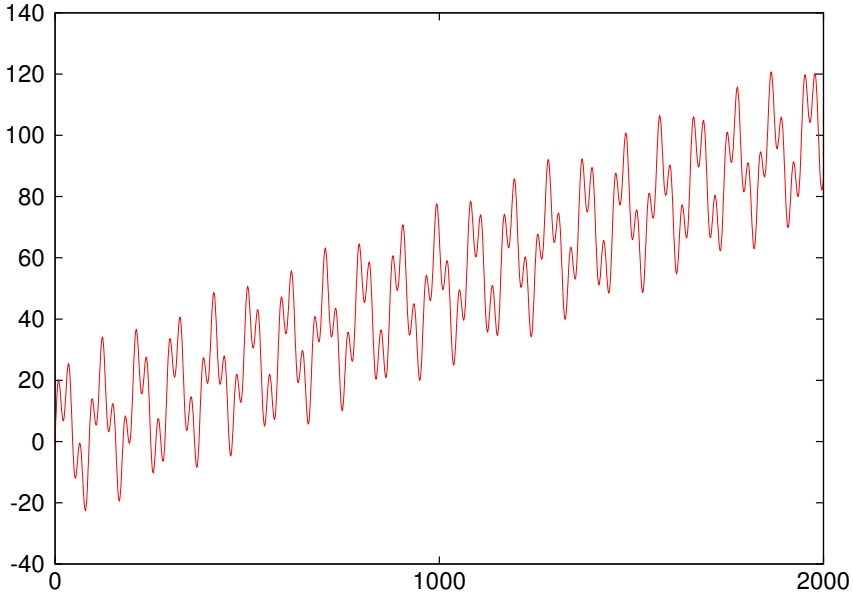


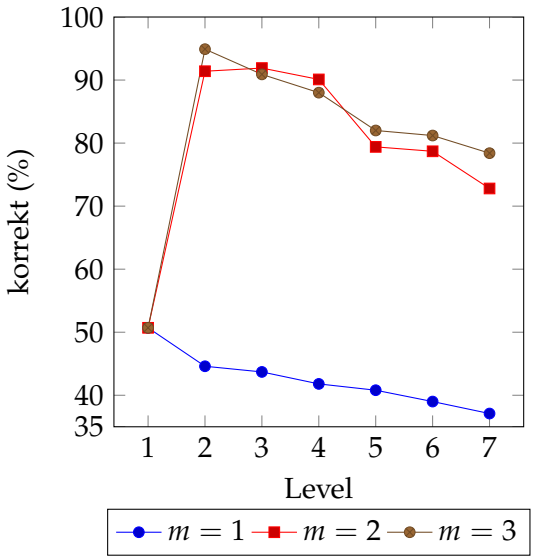
Abbildung 5.1.: Verlauf der Zeitreihe des „Kurve“-Datensatzes vor den Vorverarbeitungsschritten.

Mit diesen Parametern ergibt sich die in Abbildung 5.1 abgebildete Funktion. Die Gesamtperiode der Sinusfunktion ist das kleinste gemeinsame Vielfache der Einzelperioden, in diesem Fall also $97 \cdot 29 = 2813$, da beide Einzelperioden Primzahlen sind. Durch die Wahl einer geringen positiven Steigung wird das Data Mining etwas erschwert, da im Verlauf der Zeit immer neue, noch nie aufgetretene Werte auftreten. Trotzdem ist sichergestellt, dass f_{sin} klare Muster besitzt, da die Ableitung von f_{sin} periodisch ist. Da eine Zeitreihe über 5000 Zeitschritte generiert wurde und eine Fenstergröße mit 3000 Schritten gewählt wurde, ist die vorliegende Funktion bis auf die Gerade periodisch.

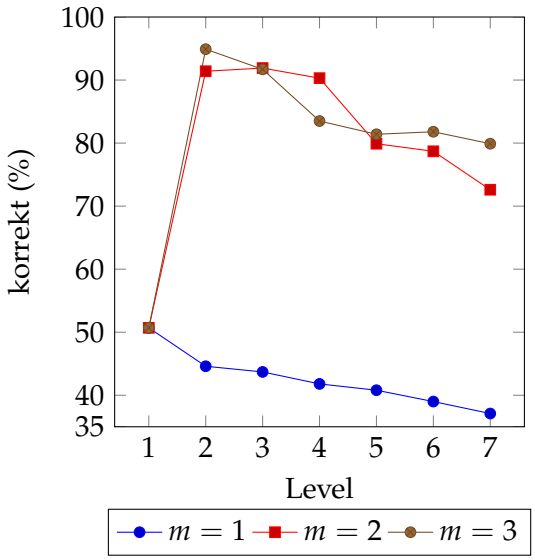
Die beschriebene Funktion hat die Eigenschaft, dass eine Trendvorhersage extrem einfach ist. Da die Anzahl der Extremstellen bezogen auf die 5000 Zeitschritte relativ gering ist, ist eine einfache Schätzung nach folgender Regel möglich: Wenn die Funktion im letzten Zeitschritt gestiegen ist, dann steigt sie auch in diesem Zeitschritt.

Wird diese einfache Regel verwendet, dann werden von den ersten 5000 Zeitschritten 346 falsch vorhergesagt, weil sich der Trend an den Extremstellen ändert. Das heißt, es ergibt sich eine Trefferrate von $93,1\%$. Daraus folgt, dass erst ab einer Trefferrate von mehr als $93,1\%$ auch einige der schwerer vorherzusagenden Trendwechsel garantiert erfolgreich vorhergesagt werden.

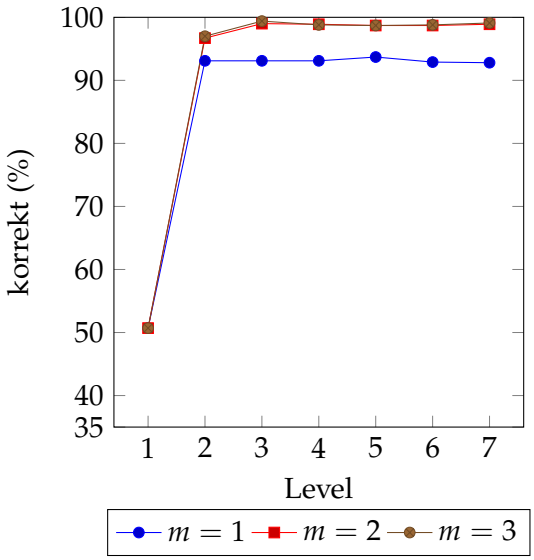
Bei den mit diesem Datensatz durchgeführten Experimenten wurde die Schrittanzahl m für jeden Konstruktor variiert. Wie in Abschnitt 4.2.1 beschrieben, steuert diese, wie viele Datenpunkte aus der Vergangenheit zur Konstruktion eines Tupels im Attributraum herangezogen werden. Daraus ergibt sich gleichzeitig die Dimension des Attributs.



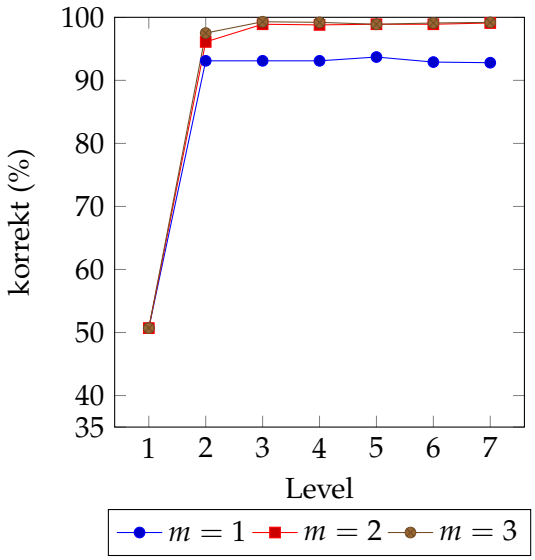
(a) Linear-Konstruktor



(b) Linear-Quad-Konstruktor



(c) Differenz-Konstruktor



(d) Differenz-Quad-Konstruktor

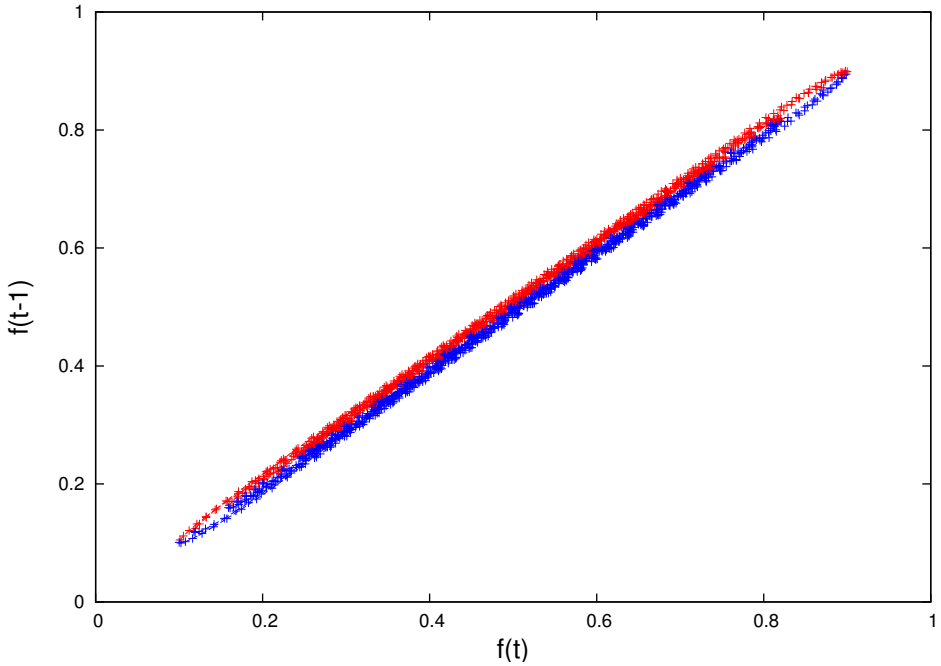
Abbildung 5.2.: Qualität der Vorhersagen verschiedener Attributkonstruktoren angewendet auf den „Kurve“-Datensatz. Dabei wird der Level des Gitters bei jedem Konstruktor variiert.

In Abbildung 5.2 sind die Ergebnisse der Experimente mit verschiedenen Attributkonstruktoren und dem „Kurve“-Datensatz zu sehen. Es kann beobachtet werden, dass die Genauigkeit der Vorhersage mit der Wahl des Konstruktors variiert. Da die Funktion f_{sin}, wie oben erwähnt, häufig zuvor noch nicht erreichte Werte einnimmt, ist die Vorhersagequalität der direkt auf den Werten operierenden Attributkonstruktoren relativ niedrig. Trotzdem ist die Trefferrate immer noch relativ hoch, da der Linear-Konstruktor bis zu $96,4\%$ korrekte Vorhersagen erreichen kann, der Linear-Quad-Konstruktor erreicht bis zu $96,2\%$. Die hohe Trefferrate kann dadurch erklärt werden, dass neue Werte nicht zu häufig erreicht werden, und zuvor erreichte Werte eine Aussage über den zukünftigen Verlauf erlauben.

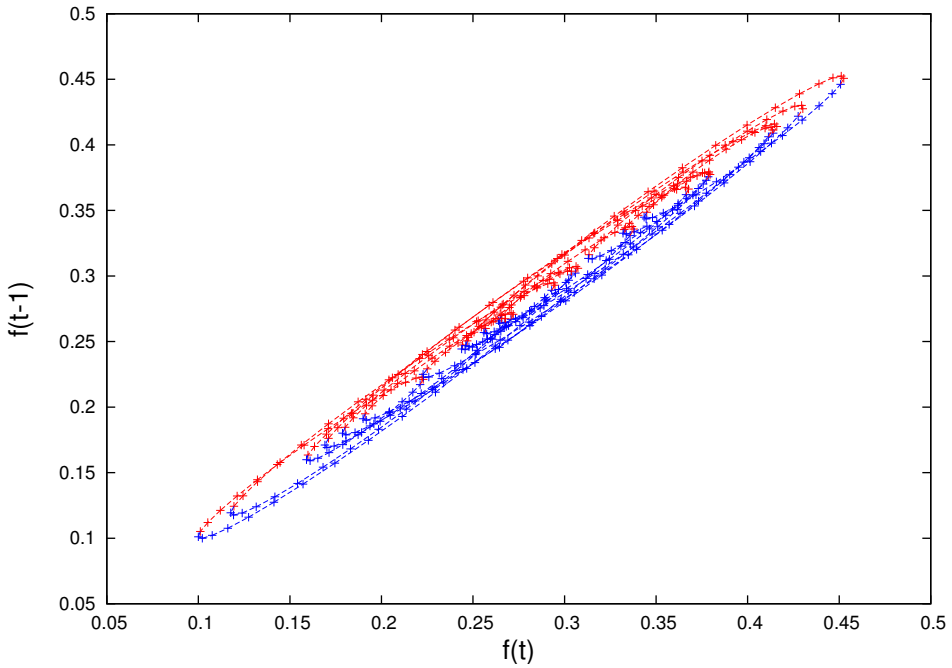
Abbildung 5.3 veranschaulicht dieses Phänomen. Diese Abbildung zeigt den Attributraum für den Linear-Konstruktor mit $m = 2$, wodurch ein zweidimensionaler Attributraum entsteht. Abbildung 5.3a zeigt dabei die Kurve über 2000 Zeitschritte, Abbildung 5.3b über einen kleineren Ausschnitt von 400 Zeitschritten. Datenpunkte, die zu unterschiedlichen Klassen gehören, sind farblich unterschiedlich markiert, wobei steigende Zeitschritte blau und fallende Zeitschritte rot eingefärbt sind. Die Kurve verläuft in Kreisbahnen von links unten nach rechts oben durch den Raum. Die Kreisbahnen sind auf die Sinusfunktionen zurückzuführen. Dadurch, dass die Werte im Verlauf der Zeit steigen, wandert die Kurve langsam von links unten nach rechts oben. Ein Punkt $(f(t), f(t-1))$ am Anfang des betrachteten Zeitfensters liegt eher links unten, weil die Werte der Zeitreihe klein (und normiert) sind, im Verlauf der Zeit jedoch größer werden. Die Kurve verläuft in der Nähe der Winkelhalbierenden, da die Differenz der Werte zwischen einzelnen Zeitschritten relativ klein ist. Da erkennbar ist, dass sich die Klassen nur wenig überlappen, lässt dies zunächst eine hohe Trefferrate erwarten. Dadurch, dass die Kurve durch den Attributraum wandert, finden die Auswertungen zur Vorhersage des nächsten Zeitschritts allerdings immer am Rand der aktuellen Menge an Datenpunkten im Attributraum statt. Dabei liegen einige wenige Punkte in der Nähe, wodurch die Anzahl der korrekten Vorhersagen immer noch relativ hoch ist. Das Auftreten von Mustern im Attributraum allein garantiert also noch keine optimale Trefferrate, da es vorkommen kann, dass die Muster, wie in diesem Beispiel zu sehen, für die Vorhersage nur eingeschränkt verwendet werden können.

Da der Datensatz bei Verwendung des Differenz-Konstruktors periodisch wird, ist eine höhere Trefferrate zu erwarten. Die Ergebnisse aus Abbildung 5.2 bestätigen diese Vermutung. Der Differenz-Konstruktor erreicht eine Trefferrate von bis zu $99,4\%$, der Differenz-Quad-Konstruktor kann sogar eine Trefferrate von bis zu $99,7\%$ erreichen.

In Abbildung 5.4 ist der berechnete Attributraum für den Differenz-Konstruktor zu sehen. Die Farbcodierung ist wieder Rot für „fallend“ und blau für „steigend“. Um die Periodizität der Kurve im Datenraum zu veranschaulichen, wurde eine modifizierte Funktion f_{sin} geplottet, bei die Perioden der Sinusfunktionen deutlich reduziert wurden. An dieser Abbildung kann deutlich gesehen werden, dass die meisten steigenden oder fallenden Datenpunkte klar räumlich getrennt sind. Die Korrektheit aller Vorhersagen kann vermutlich deswegen nicht erreicht werden, weil die Klassen sehr nahe beieinander liegen und eine exakte Trennung sehr viele Gitterpunkte erfordert.



(a) Attributraum von f_{sin} dargestellt als $f_{sin}(t)$ gegen $f_{sin}(t-1)$ über 2000 Zeitschritte.



(b) Attributraum von f_{sin} dargestellt als $f_{sin}(t)$ gegen $f_{sin}(t-1)$ über 400 Zeitschritte.

Abbildung 5.3.: Zeitreihe des „Kurve“-Datensatzes im 2D-Attributraum, transformiert mit dem Linear-Attributkonstruktor. Die Klassen sind farblich unterschieden. Die Kurve verläuft in Kreisbahnen von links unten nach rechts oben.

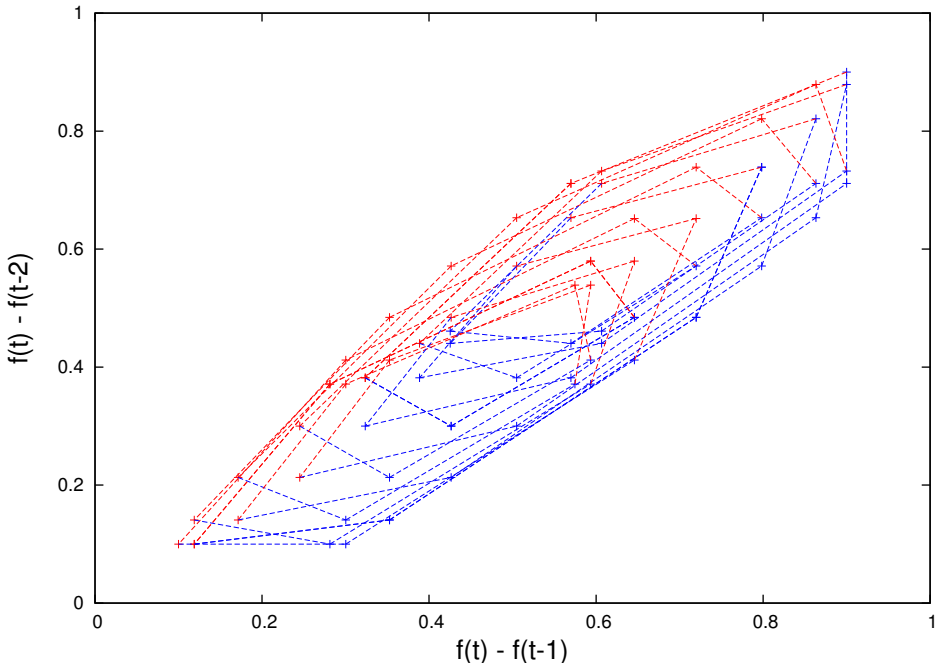


Abbildung 5.4.: Attributraum von f_{sin} dargestellt als $f_{sin}(t) - f_{sin}(t-1)$ gegen $f_{sin}(t) - f_{sin}(t-2)$ bei stark reduzierter Periode. Da mehr Zeitschritte dargestellt werden, als die Periode der Funktion lang ist, zeigt die Abbildung die vollständige Funktion. Die Klassen sind farblich markiert.

5.2. Der synthetische Datensatz „Muster“

Der zweite synthetische Datensatz besteht aus einem geometrischen Muster, das in eine Zeitreihe codiert wird. Ausgangspunkt dafür ist die folgende Funktion:

$$f(x_n, x_{n-1}) = \begin{cases} \text{rand}(x_n, 1) & x_n < 0,5 \wedge x_{n-1} < 0,5 \wedge \\ & (x_n - 0,25)^2 + (x_{n-1} - 0,25)^2 > 0,2^2 \\ \text{rand}(x_n, 1) & x_n > 0,5 \wedge x_{n-1} > 0,5 \wedge \\ & (x_n - 0,75)^2 + (x_{n-1} - 0,75)^2 < 0,2^2 \\ \text{rand}(x_n, 1) & x_n > 0,5 \wedge x_{n-1} < 0,5 \wedge 1 - x_{n-1} < x_n \\ \text{rand}(x_n, 1) & x_{n-1} > 0,5 \wedge x_n < 0,5 \wedge 1 - x_{n-1} < x_n \\ \text{rand}(0, x_n) & \text{sonst} \end{cases} \tag{5.5}$$

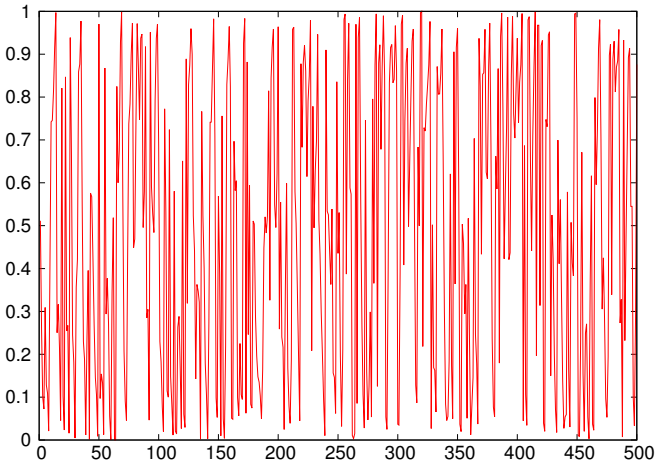


Abbildung 5.5.: Verlauf der Zeitreihe des „Muster"-Datensatzes vor den Vorverarbeitungsschritten.

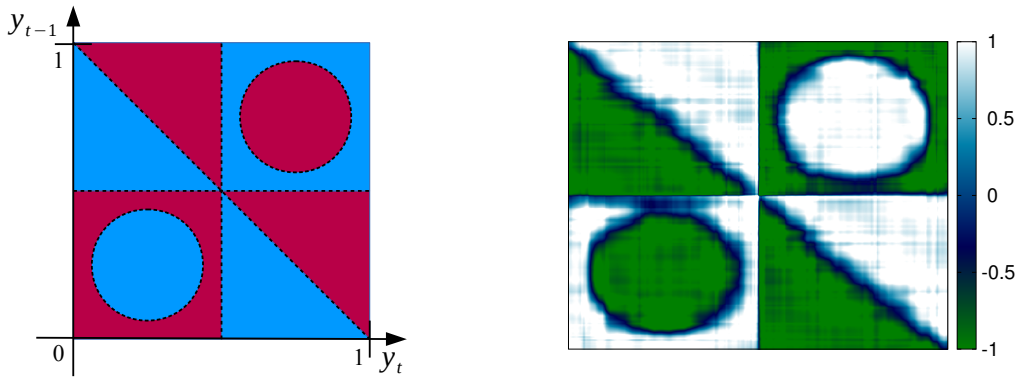
Die Funktion rand$(.,.)$ gibt einen zufälligen Wert aus dem durch die Argumente spezifizierten Intervall zurück. Die Zeitreihe wurde aus dem Datensatz durch die Rekursionsvorschrift

$$g(n+1) = f(g(n), g(n-1)) \tag{5.6}$$
$$g(1) = \text{rand}(0,1) \tag{5.7}$$
$$g(2) = \text{rand}(0,1) \tag{5.8}$$

erzeugt. Dabei wurden zusätzlich alle bereits erzeugten Werte von g zur Berechnung zukünftiger Zeitschritte wiederverwendet. Der Verlauf der Zeitreihe mit dieser Rekursionsvorschrift ist zufällig. Abbildung 5.5 zeigt einen möglichen Verlauf der Zeitreihe für einige Zeitschritte.

Wird dieser Datensatz auf eine bestimmte Weise betrachtet, ergibt sich ein deutlich erkennbares Muster, das in Abbildung 5.6a dargestellt ist. Die Fallunterscheidung in der Definition von f entspricht allen Bereichen, die in der Abbildung rot dargestellt sind. Dabei bedeutet es geometrisch, dass der nächste Wert der Zeitreihe aus dem Bereich zwischen dem aktuellen Wert und Eins gelost wird, wenn das Tupel bestehend aus dem aktuellen und dem letzten Wert eine Koordinate in einem roten Bereich darstellt. Entsprechend bedeutet es für ein Tupel, wenn es in einem blauen Bereich liegt, dass der nächste Wert aus dem Bereich zwischen null und dem aktuellen Wert gelost wird. In Abbildung 5.6b ist eine gute Näherung der Funktion zu sehen vor der Klassifikation zu sehen. Nach der Klassifikation würden alle Werte größer null der „roten" Klasse zugeordnet werden.

Wie beim Datensatz „Kurve" wurden auch hier wieder Experimente mit mehreren Attributkonstruktoren und Konfigurationen durchgeführt. Ein Überblick über die Ergebnisse ist in Abbildung 5.7 zu sehen. Da die Zeitreihe aufgrund ihrer Konstruktion durch den Linear-Konstruktor in zwei Dimensionen nahezu perfekt abgebildet werden sollte, ist hier die erreichte Vorhersagegenauigkeit am höchsten. Der Linear-Konstruktor erreichte eine



(a) Ideale Geometrie im Attributraum **(b)** Heatmap einer guten Näherung der Funktion

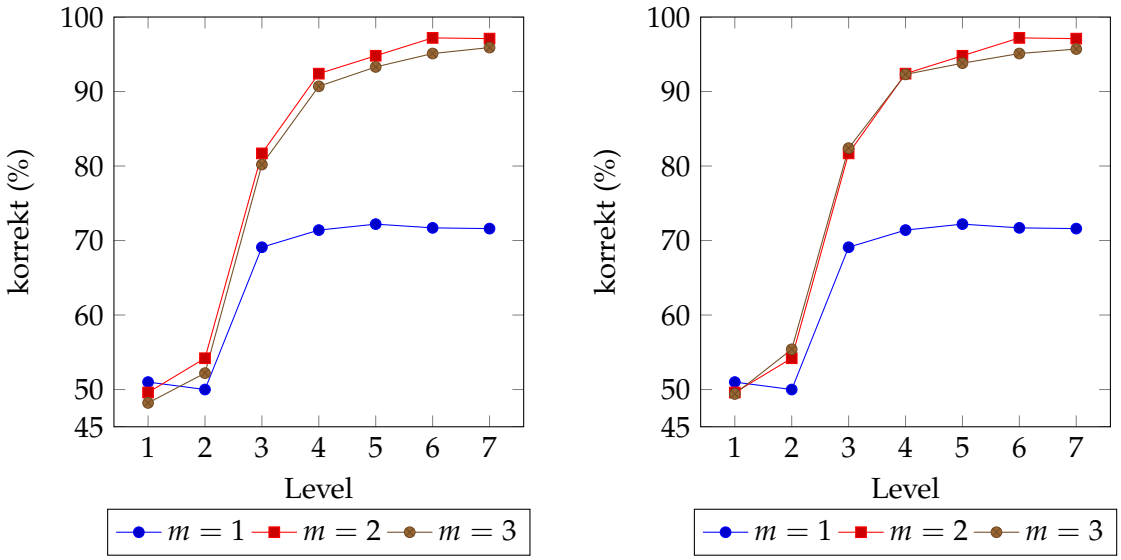
Abbildung 5.6.: Auf der linken Seite ist das Schema der Konstruktion des Datensatzes abgebildet. Rechts ist eine Dünngitterfunktion abgebildet, die eine sehr gute Annäherung an das Konstruktionsschema darstellt.

Trefferrate von 95,2%. Interessanterweise konnte dessen quadratische Variante dieselbe Genauigkeit erreichen.

In Abbildung 5.7 sind außerdem Ergebnisse für den Differenz-Konstruktor und den Differenz-Quad-Konstruktor zu sehen. Wie erwartet liegen die Ergebnisse deutlich unter den Ergebnissen des Linear-Konstruktors, da der Raum nicht entsprechend einem Differenzenschema aufgebaut wurde. Gerade dadurch, dass der nächste Wert vom Zufall abhängt, reduziert sich die Anzahl korrekter Vorhersagen, da die Differenz nahezu beliebige Werte einnehmen können. Dabei fällt insbesondere auf, dass die Vorhersage trotz des nicht optimalen Konstruktors immer noch deutlich über 50% liegt. Das heißt, auch bei einem schlecht gewählten Konstruktor können immer noch Muster im Attributraum auftreten.

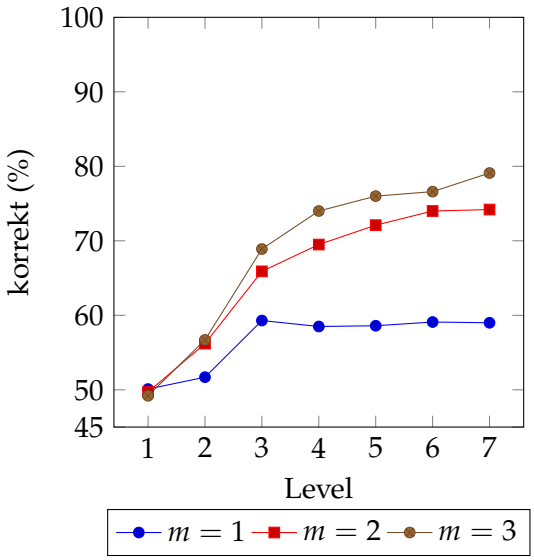
Aufgrund der Konstruktion des Raums mit zwei Kreisen und einer diagonal verlaufenden Grenze zwischen den Klassen werden, trotz der relativ einfachen Struktur des Problems, für eine hohe Vorhersagegenauigkeit sehr viele Gitterpunkte benötigt. Dies zeigt sich auch bei den gewählten Konstruktoren. Mit zunehmendem Level steigt die Qualität der Vorhersage weiter an. Eine perfekte Trefferrate wird mit den vorgestellten Experimenten noch nicht erreicht, obwohl aufgrund der Konstruktion keine Überlappung der Klassen vorliegt. Wird mit dem Linear-Konstruktor und noch höherem Level gerechnet, lässt sich die Trefferrate auf nahezu 100% verbessern. Eine perfekte Vorhersage ist allerdings schwierig, da Punkte auf den scharfen Klassengrenzen nur mit äußerst vielen Gitterpunkten immer korrekt den Klassen zugeordnet werden können.

Dieses Beispiel zeigt weiter, dass für eine gute Vorhersage keineswegs Periodizität benötigt wird. Trotz des zufälligen Verlaufs der Kurve durch den Attributraum ist der Verlauf der Kurve regelbasierend. Des Weiteren zeigen beide synthetischen Beispiele, dass die Wahl des

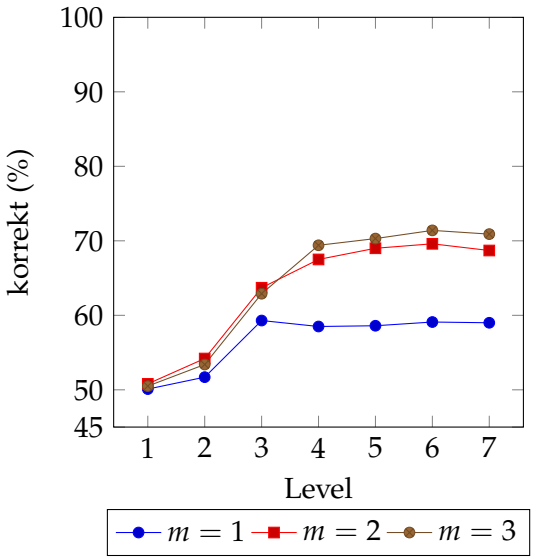


(a) Linear-Konstruktor

(b) Linear-Quad-Konstruktor



(c) Differenz-Konstruktor



(d) Differenz-Quad-Konstruktor

Abbildung 5.7.: Qualität der Vorhersagen verschiedener Attributkonstruktoren angewendet auf den „Muster“-Datensatz. Dabei wird der Level des Gitters bei jedem Konstruktor variiert.

Konstruktors äußerst wichtig ist. Wird dieser optimal gewählt und sind die Daten qualitativ ausreichend, dann können Vorhersagen mit hoher Präzision durchgeführt werden.

Dieser Datensatz ist auch ein gutes Beispiel dafür, dass die Betrachtung eines Problems in einer höheren Dimension nicht zwangsläufig zu einer Verbesserung der Vorhersage führt. Durch die Technik der Konstruktion der Zeitreihe handelt es sich inhärent um ein Problem in zwei Dimensionen. Die Trefferrate der Konstruktoren in Abbildung 5.7 bestätigt diese Hypothese.

5.3. Experimente mit Finanzdatensätzen

Durch Experimente mit real gemessenen Daten kann gezeigt werden, dass Zeitreihenanalyse mit dünnen Gittern auch abseits von synthetischen Experimenten erfolgreich zur Vorhersage verwendet werden kann. Dazu wurden drei Datensätze verwendet. Zwei davon bestehen aus Zeitschritten bekannter Aktienindizes, dem deutschen DAX und dem amerikanischen Dow Jones. Der dritte Datensatz beinhaltet Euro-Dollar Umrechnungskurse im Verlauf der Zeit. Ziel ist wie bisher die korrekte Vorhersage des Trends.

5.3.1. Vorhersage des Deutschen Aktienindex DAX

Als erstes Beispiel für nichtsynthetische Daten wird ein Ausschnitt des Kursverlaufs des Deutschen Aktienindex verwendet. Der DAX ist der wichtigste deutsche Aktienindex. Er setzt sich aus den 30 größten und umsatzstärksten Unternehmen zusammen, die an der Frankfurter Börse gelistet sind. Darunter sind Firmen wie Bayer, Siemens und BASF [Dax].

Die Daten der hier vorgestellten Experimente wurden aus Googles Finanzdienst „Finance“ manuell gewonnen [Goo]. Die verwendeten Daten stammen aus dem Zeitraum vom 30.10.2013 um 8:03 Uhr bis zum 19.11.2013 um 4:36 Uhr. Die Datenpunkte liegen in Minutenabständen vor, allerdings wird der DAX nur an Werktagen von 9:00 bis 17:45 gehandelt, weswegen große Lücken über die Nächte und über die Wochenenden entstehen.

In Abbildung 5.8 ist der Verlauf des DAX im verwendeten Zeitraum zu sehen. Deutlich erkennbar sind auch die Zeiträume, in denen der DAX tatsächlich gehandelt wird. Die kurzen Lücken werden durch die Nächte der Werktage hervorgerufen, die langen Lücken entstehen durch Wochenenden. Zusätzlich sind die Daten unvollständig, da für einige Zeitschritte keine Daten vorliegen. Für den beschriebenen Zeitraum liegen insgesamt 7573 Datenpunkte vor. Da die Werte minütlich vorliegen, kann errechnet werden, dass nicht für jede Minute ein Wert vorliegt. Da ebenfalls Daten für einige der DAX Unternehmen betrachtet werden sollen, wurden Zeitpunkte, an denen nicht alle Daten verfügbar waren, verworfen. Für die Vorhersagen mithilfe von assoziierten Daten stehen daher nur 5782 Datenpunkte zur Verfügung. Für die Experimente mit diesem Datensatz wurden jeweils 1000 Testdaten verwendet, das heißt, es wurden Vorhersagen für 1000 Zeitschritte berechnet.

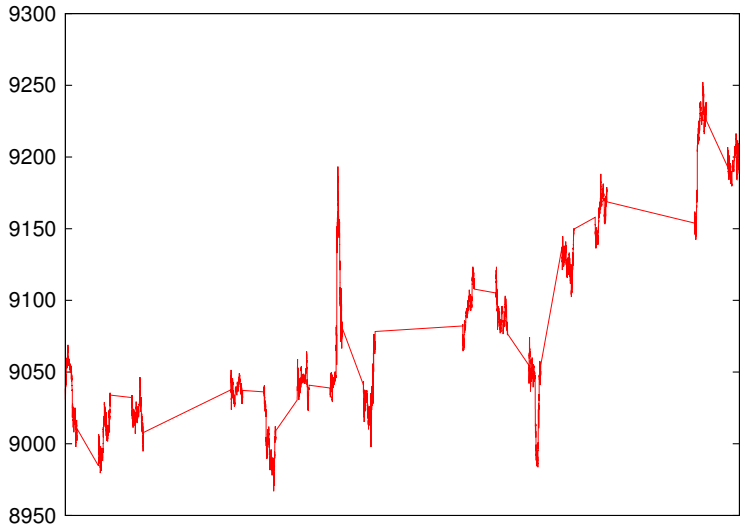


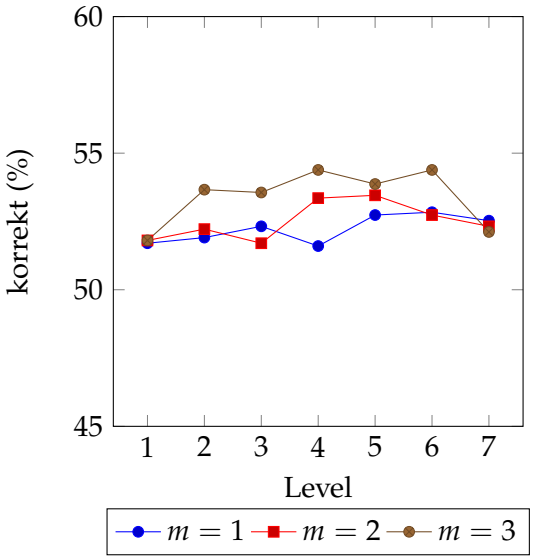
Abbildung 5.8.: Verlauf des DAX im Zeitraum zwischen dem 30.10.2013 um 8:03 Uhr und dem 19.11.2013 um 4:36 Uhr.

Zur Untersuchung der Qualität der Vorhersage wurden vier Experimente durchgeführt. Für die ersten beiden Experimente wurden der Linear-Konstruktor und der Differenz-Konstruktor ausgewählt. Als drittes Experiment wurden beide Konstruktoren der ersten beiden Experimente zusammen angewendet. Zuletzt wurden mehrere Zeitreihen betrachtet, neben dem Kurs des DAX selbst wurden dem Datensatz die Kurse von bis zu drei DAX-Unternehmen hinzugefügt.

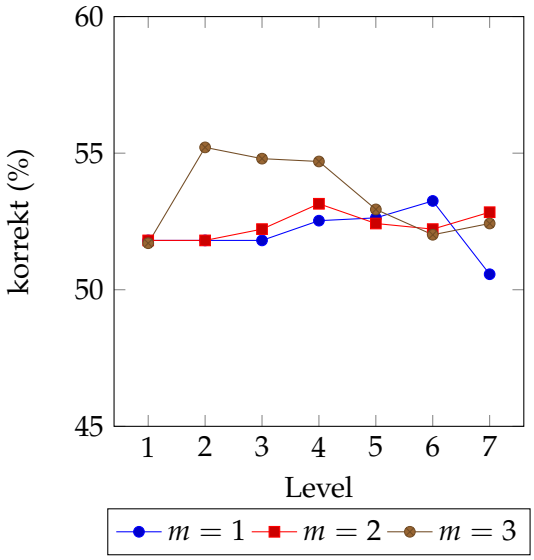
Auch für dieses Experiment gibt es eine minimal zu erreichende Trefferrate, die überschritten werden muss, damit Vorhersagen nicht als trivial gelten. Von den 1000 Elementen der Testdatenmenge sind insgesamt 969 Zeitschritte relevant. Bei 31 Zeitschritten liegt keine Änderung der Werte relativ zum Vorgänger vor. Von den relevanten Zeitschritten besitzen wiederum 502 Zeitschritte einen steigenden Trend. Da auch in den Daten der verwendeten Fenster mehr steigende als fallende Zeitschritte zu finden sind, wird die mindestens zu schlagende Basistrefferrate mit 51,8% angegeben.

In Abbildung 5.9 sind die Ergebnisse der durchgeführten Experimente zu sehen. Da für diese Experimente keine synthetischen Daten verwendet wurden, ist nicht die Trefferrate der synthetischen Experimente zu erwarten. Mit dem Linear-Konstruktor beträgt die beste erreichbare Trefferrate 54,4%, der Differenz-Konstruktor erreicht 55,2%. Die insgesamt beste Trefferrate wird durch die Kombination der beiden vorherigen Konstruktoren erreicht, sie beträgt 56,3%. Während also die generelle Trefferrate im Vergleich zu den synthetischen Daten deutlich niedriger ausfällt, wird zumindest die Basistrefferrate von 51,8% deutlich übertroffen.

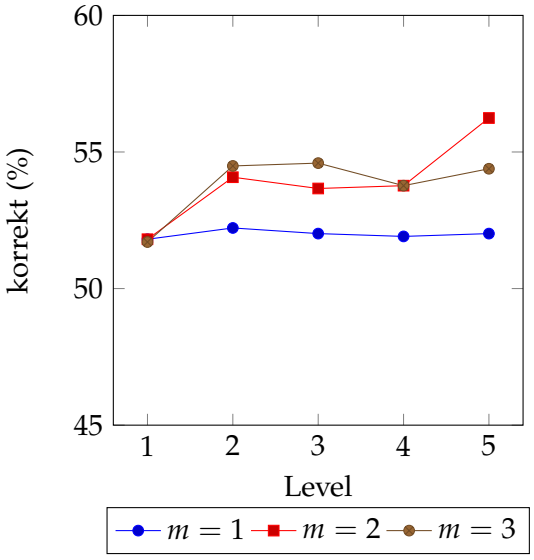
Bei der Kombination mehrerer Zeitreihen ist eine Basistrefferrate etwas niedriger, da der verwendete Datensatz, wie oben beschrieben, modifiziert wurde. Die Basistrefferrate liegt hier bei 50,6%. Da durch die Hinzunahme von zwei weiteren Zeitreihen eine Trefferrate



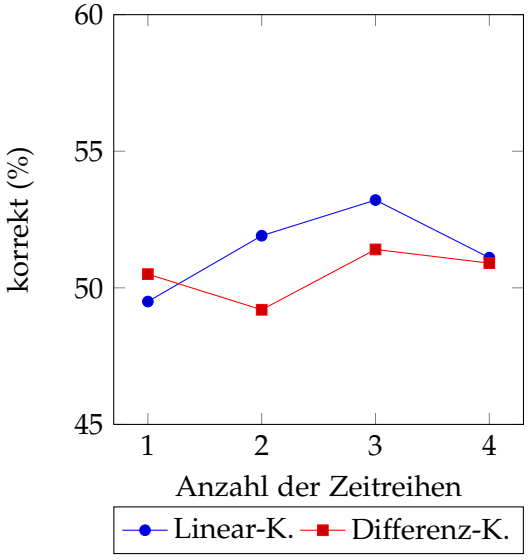
(a) Linear-Konstruktor



(b) Differenz-Konstruktor



(c) Linear-Konstruktor und Differenz-Konstruktor



(d) Verwendung mehrerer Zeitreihen.

Abbildung 5.9.: Qualität der Vorhersagen verschiedener Attributkonstruktoren angewendet auf den „DAX“-Datensatz. In den ersten drei Abbildungen wird der Level des Gitters zusätzlich variiert. Die vierte Abbildung zeigt die Trefferraten, wenn mehrere Zeitreihen zur Vorhersage verwendet werden.

von 53,2% erreicht werden kann, liegt eine Verbesserung um 2,6% vor. Da die drei anderen Experimente eine Verbesserung von im besten Fall 4,5% schaffen, ist zumindest bei diesem Vergleich eine korrekte Wahl der Konstruktor nützlicher als die Verwendung dieser weiteren Zeitreihen.

Im Allgemeinen steigt die Trefferrate zwischen Level 1 und Level 2 stark an. Dies deutet darauf hin, dass die erkannten Muster relativ grob sind. Ansonsten würde das Rechnen mit höheren Leveln zu weiteren Verbesserungen führen. Am deutlichsten ist dies bei Verwendung des Differenz-Konstruktors zu beobachten. Nach einer relativ hohen Trefferrate auf einem Gitter mit Level 2, fällt die Trefferrate mit zunehmendem Level tendenziell ab. Hier scheint ein grobes Muster durch wenige Gitterpunkte gut wiedergegeben zu werden, während es bei mehr Gitterpunkten schnell zu Overfitting kommt. Außerdem kann beobachtet werden, dass für akzeptable Vorhersagen mindestens ein zweidimensionaler Attributraum benötigt wird. Wird nur mit einer Dimension gerechnet, liegen die Trefferraten kaum über der Basistrefferrate.

5.3.2. Vorhersage des amerikanischen Aktienindex Dow Jones

Der Dow Jones ist ein wichtiger Aktienindex der New York Stock Exchange. Ähnlich wie der deutsche DAX ist der Dow Jones ein Aktienindex, dessen Wert sich aus dem gewichteten Wert großer Unternehmen zusammensetzt. Unternehmen mit starker Gewichtung sind unter anderem Visa, IBM, Goldman Sachs, Boeing und 3M [Dow]. In dieser Arbeit wird der Verlauf des Dow Jones und seiner assoziierten Unternehmen im Zeitraum zwischen dem 18.9.2013 um 1:31 Uhr und dem 8.10.2013 um 8:00 Uhr betrachtet. Dafür liegen 5711 Datenpunkte vor, die auf einen Zeitraum von etwa 20 Tagen verteilt sind. Wie die DAX-Daten wurde auch dieser Satz mithilfe von „Google Finance“ erstellt [Goo].

Der Verlauf des Dow Jones über den betrachteten Zeitraum ist in Abbildung 5.10 zu sehen. Ähnlich wie beim „DAX“-Datensatz gilt auch hier wieder, dass die zeitlichen Sprünge durch Nächte und Wochenenden erklärbar sind. Aufgrund von Wochenenden und der nicht ganztägigen Berechnungszeit des Dow Jones liegt bei diesem Datensatz pro Minute ein Wert vor. Auch hier ist der Datensatz nicht vollständig, weswegen Zeitschritte mit unvollständigen Daten verworfen wurden.

Da der Dow Jones dem DAX strukturell ähnelt, wurden dieselben Experimente durchgeführt. Es wurden auch hier der Linear-Konstruktor und der Differenz-Konstruktor einzeln und kombiniert angewendet. Ebenfalls wurde ein Experiment mit assoziierten Zeitreihen durchgeführt, wobei auch hier drei der Unternehmen gewählt wurden, aus denen sich der Dow Jones zusammensetzt.

Von den 1000 Elementen der Testdatenmengen, sind 951 für die Vorhersage relevant. Da 493 Zeitschritte mit steigenden Werten vorliegen, liegt die Basistrefferrate bei 51,8%. Durch eine entsprechende Rechnung liegt die Basistrefferrate beim modifizierten Datensatz für mehrere Zeitreihen bei 50,8%.

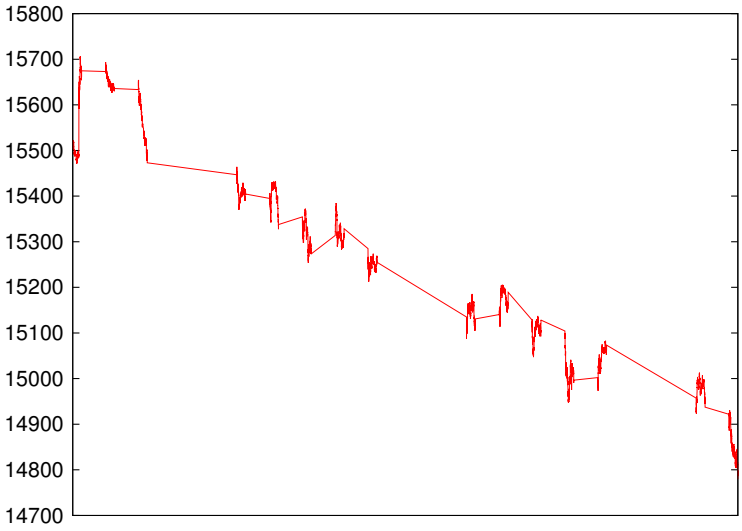


Abbildung 5.10.: Verlauf des Dow Jones im Zeitraum zwischen dem 18.9.2013 um 1:31 Uhr und dem 8.10.2013 um 8:00 Uhr.

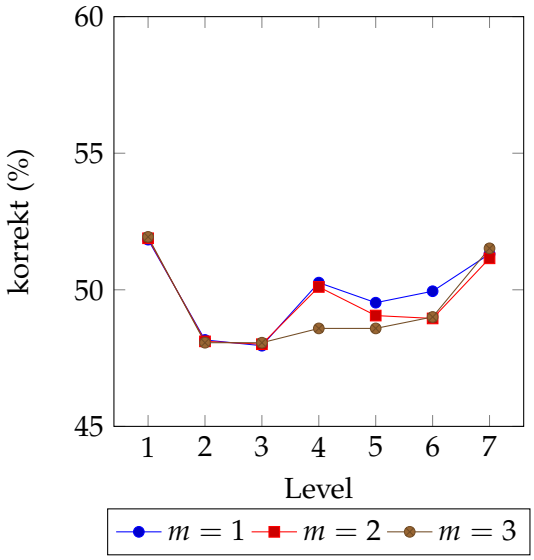
Wie die Ergebnisse der Experimente in Abbildung 5.11 zeigen, liegt die erzielte Trefferrate nur wenig über der Basistrefferrate. Das beste Ergebnis aus den einzelnen Konstruktoren und den kombinierten Konstruktoren beträgt 53,3% und liegt damit 1,5% über der Basistrefferrate von 51,8%. Eine Verbesserung kann durch die Verwendung einer zusätzlichen Zeitreihe erzielt werden. Dann steigt die Trefferrate auf 53,9%, wobei die Basistrefferrate hier 50,8% beträgt. Damit liegt das beste Experiment immerhin 3,1% über der Basistrefferrate.

Die erzielten Trefferraten bewegen sich damit auf einem ähnlichen Niveau wie die DAX-Daten. Da beide Datensätze eine ähnliche Struktur und eine ähnliche Qualität besitzen, ist ein solches Ergebnis plausibel. Eine Verbesserung der Trefferraten könnte vermutlich bei beiden Datensätzen erreicht werden, wenn weniger lückenhafte Daten vorliegen. Besonders über die langen Lücken durch die Nächte und Wochenenden sind zum Teil erhebliche Änderungen der Werte zu beobachten. Durch eine Verknüpfung mit weiteren Zeitreihen, deren Werte die nächtlichen Änderungen abschätzen lassen, sollte ebenfalls eine weitere Verbesserung erreichbar sein.

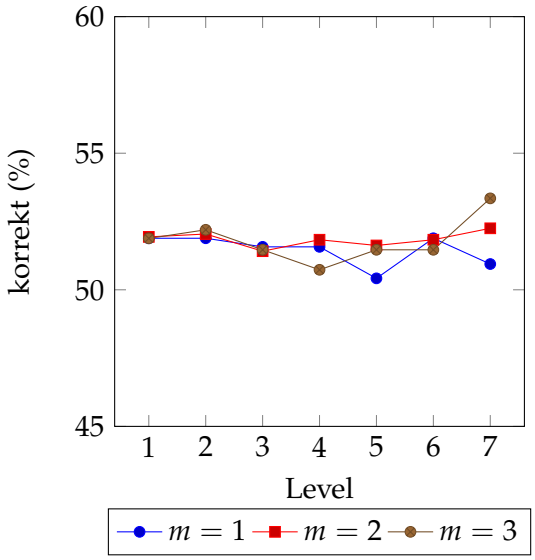
5.3.3. Vorhersage im Hochfrequenzhandel am Beispiel von Euro-Dollar Wechselkursen

Als dritter Datensatz zur Evaluierung des Ansatzes unter realistischeren Bedingungen wurden Wechselkurse des Euros in Dollar verwendet. Hierzu wurde ein Datensatz verwendet, der über den Datenanbieter TrueFX[2] bezogen wurde [Tru]. Bei den verwendeten Daten handelt es sich um Tickdaten. Das bedeutet, dass jede Änderung des Kurses verzeichnet ist.

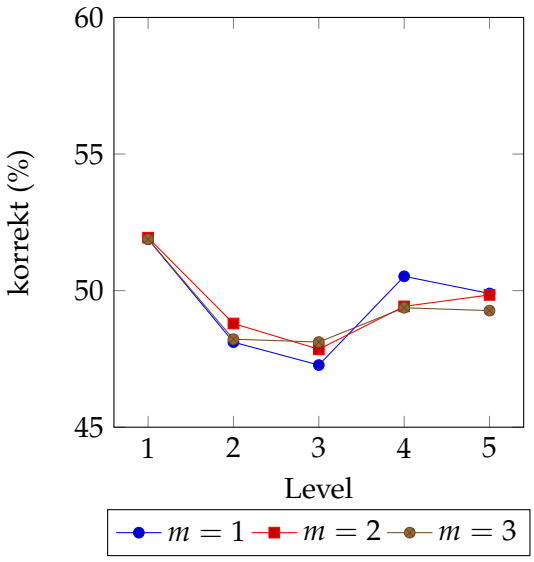
[2]TrueFX ist selbst ein Produkt der Integral Development Corporation.



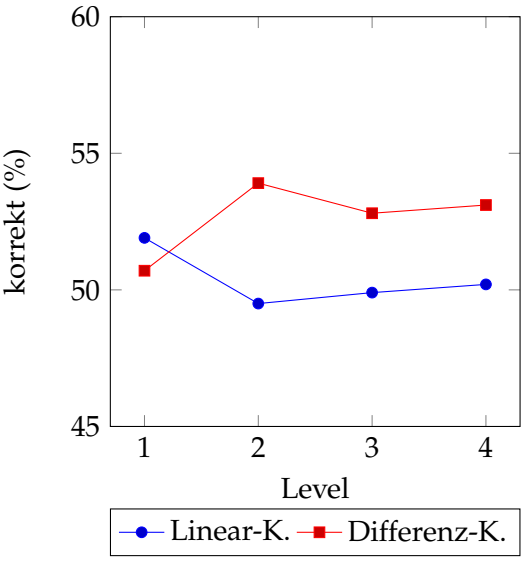
(a) Linear-Konstruktor



(b) Differenz-Konstruktor



(c) Linear-Konstruktor und Differenz-Konstruktor



(d) Trefferrate bei unterschiedlicher Anzahl an verwendeten Zeitreihen

Abbildung 5.11.: Qualität der Vorhersagen verschiedener Attributkonstruktoren angewendet auf den „Dow Jones“-Datensatz. In den ersten drei Abbildungen wird der Level des Gitters zusätzlich variiert. Die vierte Abbildung zeigt die Trefferraten, wenn mehrere Zeitreihen zur Vorhersage verwendet werden.

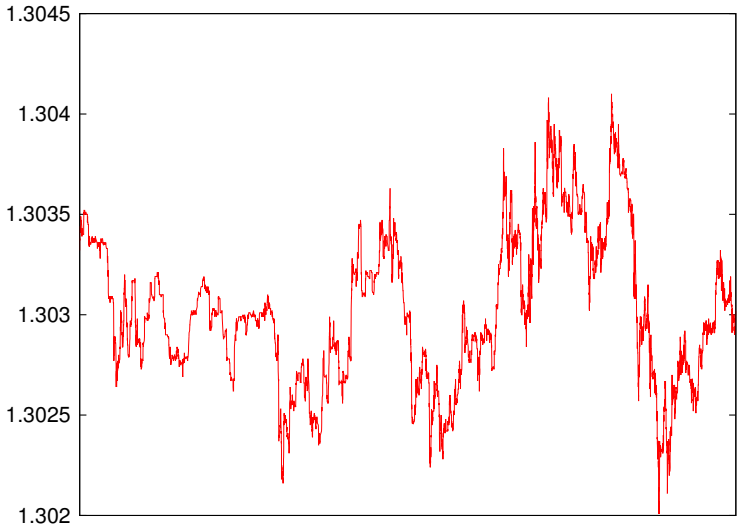


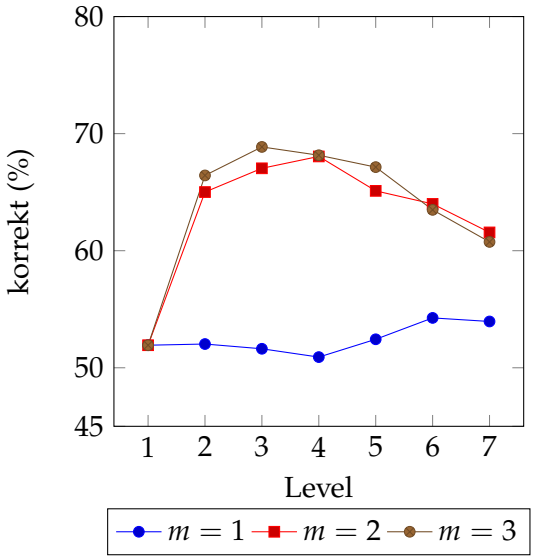
Abbildung 5.12.: Verlauf des Dollar in Euro Kurses im Dezember 2012.

Für die hier vorgestellten Experimente wurde der Datensatz so vorverarbeitet, dass Daten in Minutenintervallen vorliegen. Betrachtet wurde der Zeitraum vom 2.12.2012 um 11:00 Uhr bis zum 3.12.2012 um 10:33 Uhr. Daraus resultierten insgesamt 15430 Datenpunkte (aus den 6,7 Millionen Tickdaten für den gesamten Dezember 2012). Von den verbliebenen Datenpunkten wurden wiederum 2000 als Testdaten ausgewählt.

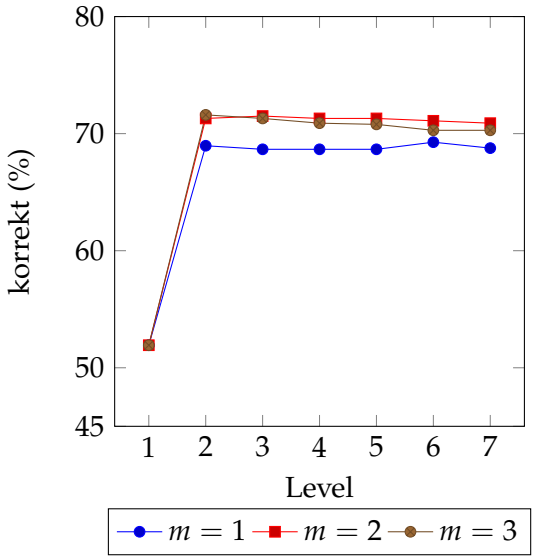
Um auch Experimente mit korrelierten Daten durchführen zu können, wurden zusätzlich Wechselkurse des Euros in britische Pfund verwendet. Diese wurden vom selben Datenanbieter bezogen und liegen ebenfalls als Tickdaten vor. Da Tickdaten jedoch schwer synchronisierbar sind, wurden die Daten modifiziert kombiniert. Dabei wurde immer der erste Sekundenwert als Teil des modifizierten Datensatzes ausgewählt, bei dem für beide Zeitreihen Daten vorliegen haben. Daraus entstand ein Datensatz mit 7652 Zeitschritten für den oben genannten Zeitabschnitt. Damit bestehen große Unterschiede zwischen dem allgemeinen Datensatz und dem Datensatz für Experimente mit korrelierten Zeitreihen. Letzterer Datensatz arbeitet auf festen Sekundenintervallen, der allgemeine Datensatz arbeitet dagegen auf den unregelmäßigen Tickdaten.

Auch mit diesem Datensatz wurden dieselben Experimente wie bei den beiden vorherigen Datensätzen durchgeführt. Da der Datensatz des Experiments mit mehreren Zeitreihen stark von den anderen Experimenten abweicht, sollte dieser allerdings als eigenständig betrachtet werden.

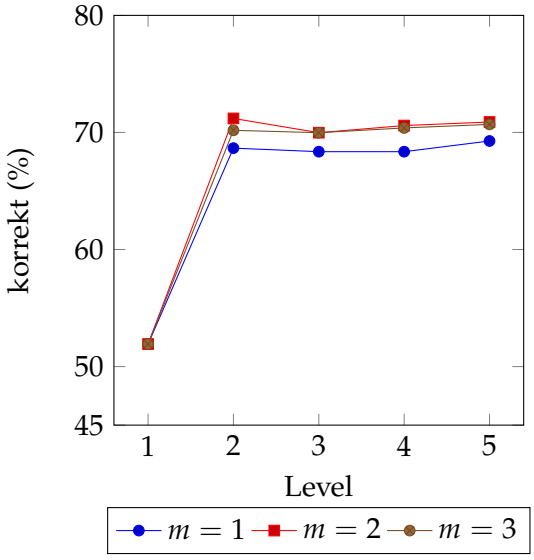
Für die ersten drei Experimente liegen 2000 Testzeitpunkte vor, darunter 692 steigende Zeitschritte und 676 fallende Zeitschritte. Die erwartete Basistrefferrate liegt damit bei 50,6%. Wie in Abbildung 5.13 zu sehen ist, kann die Basistrefferrate in den durchgeführten Experimenten deutlich überboten werden. Bereits der Linear-Konstruktor erreicht eine Trefferrate von bis zu 68,86%. Dies wird durch den Differenz-Konstruktor sogar noch weiter



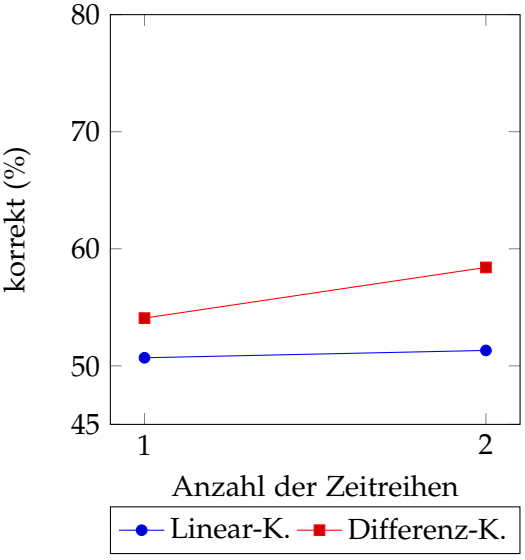
(a) Linear-Konstruktor



(b) Differenz-Konstruktor



(c) Linear-Konstruktor und Differenz-Konstruktor



(d) Trefferrate bei unterschiedlicher Anzahl an verwendeten Zeitreihen

Abbildung 5.13.: Qualität der Vorhersagen verschiedener Attributkonstruktoren angewendet auf den „FX“-Datensatz. In den ersten drei Abbildungen wird der Level des Gitters zusätzlich variiert. Die vierte Abbildung zeigt die Trefferraten, wenn mehrere Zeitreihen zur Vorhersage verwendet werden, dabei wird eine stark modifizierte Zeitreihe verwendet.

überboten, da dieser bis 71,6% korrekte Vorhersagen liefert. Der kombinierte Ansatz erreicht eine Trefferrate von 71,2%.

Es fällt auf, dass alle Experimente sehr hohe Trefferraten besitzen. Und dass für hohe Trefferraten bereits ein niedriger Level ausreicht. Gleichzeitig korreliert eine niedrigere Dimension tendenziell mit schlechteren Vorhersagen. All dies deutet darauf hin, dass der betrachtete Zeitraum einem verhältnismäßig einfachen Prozess unterliegt, der bereits mit wenigen Gitterpunkten gut abgebildet werden kann. Weitere Muster werden aber scheinbar nicht erkannt, da ein höherer Level nicht zu besseren Resultaten führt. Für diese Hypothese spricht besonders der Verlauf des Experiments, das den Linear-Konstruktor verwendet. Hier scheint bei höheren Levels Overfitting einzutreten. Des Weiteren wird mit dem Differenz-Konstruktor bereits auf Level 2 eine sehr hohe Trefferrate erreicht, das bedeutet, dass bereits 3 Gitterpunkte für gute Vorhersagen ausreichen. Mit 7 Gitterpunkten wird bereits die höchste Trefferrate erreicht.

Das Experiment mit mehreren Zeitreihen zeigt erwartungsgemäß einen anderen Verlauf. Während mit dem in diesem Datensatz bei Verwendung einer Zeitreihe eine Trefferrate von 54,1% erreicht wird, kann dies durch Verwendung einer weiteren Zeitreihe deutlich auf 58,3% gesteigert werden.

5.4. Interpretation der qualitativen Ergebnisse

Die mit den fünf Datensätzen durchgeführten Experimenten legen einige allgemeineren Schlüsse nahe. Die Trefferrate steigt mit zunehmendem Level und eine für das Problem passende Dimension erhöht ebenfalls die Trefferrate. Des Weiteren ist die Wahl der Konstruktoren äußerst wichtig.

Bei den synthetischen Experimenten sind diese Phänomene alle sehr deutlich zu beobachten, weil sehr gute Informationen über die Daten und zudem Daten von hoher Qualität vorliegen. Diese Informationen können genutzt werden, um geeignete Attributräume zu konstruieren. Damit konnten letztlich sehr hohe Trefferrate zu erreicht werden.

Die Trefferraten bei den Experimenten mit nichtsynthetischen Datensätzen sind erwartungsgemäß deutlich niedriger. Der Aufwand, der betrieben werden muss, um unter diesen Umständen geeignete Konstruktoren für reale Daten zu finden, ist ungleich größer, da die Beschreibung der Phänomene, die zu den Werten der Zeitreihe führen, ungleich komplizierter ist. Zudem besteht bei real gemessenen Daten das Problem, dass a priori unklar ist, ob die vorhandenen Daten auch tatsächlich für eine hohe Trefferrate bei den Vorhersagen ausreichen. Insbesondere der „DAX“- und der „Dow Jones“-Datensatz sind

Wird als Zeitreihe nur eine sehr einfache Zielgröße verwendet, wären gute Vorhersagen in vielen Fällen äußerst erstaunlich. Zum Beispiel stellt die Produktion einer Fabrik über die Zeit eine Zeitreihe dar. Und es kann auch durchaus der Fall sein, dass das Produktionsnivau bestimmten Regelmäßigkeiten folgt, die gute Vorhersagen erlauben. Trotzdem ist eine sehr hohe Trefferrate sehr unwahrscheinlich, da eine Vielzahl relevanter äußerer Umstände nicht einbezogen werden. Sollte aufgrund eines Unfalls die Produktion für einige Zeit ausfallen,

wird dies im Normfall nicht aus den Daten an vergangenen Zeitpunkten heraus vorhersagbar sein[3]. Wie bei allen Data Mining Aufgaben gilt auch hier, dass nur die Informationen aus den Daten gewonnen werden können, die im ursprünglichen Datensatz vorhanden sind.

Von Garcke et al. wurde ein ähnlicher Ansatz verfolgt, der ebenfalls auf dünnen Gittern basiert und die sogenannte Kombinationstechnik verwendet. Zudem wurden von Garcke et al. ebenfalls Börsendaten betrachtet. Unter ähnlichen Bedingungen wurde dort eine Trefferrate von knapp 53% erreicht [GGG10]. Unglücklicherweise standen die verwendeten Daten nicht zur Verfügung, da sie kommerziell bezogen wurden. Dies schränkt eine direkte Vergleichbarkeit deutlich ein.

[3]Natürlich sind auch Szenarien denkbar, wo Unfälle Regelmäßigkeiten folgen.

6. Beschleunigung der Zeitreihenanalyse

Data Mining auf großen Datensätzen kann sehr teuer sein. Bei Gitter-basierten Lösungen verschärft sich dieses Problem, wenn die Daten aus hochdimensionalen Räumen stammen. Nachdem im letzten Kapitel gezeigt wurde, dass mit dem beschriebenen Dünngitteransatz zur Zeitreiheanalyse erfolgreiche Vorhersagen möglich sind, wird in diesem Kapitel die benötigte Rechenzeit untersucht. Anschließend werden weitere Optimierungen vorgestellt und untersucht. Die Optimierungen wurden dabei nur für den auf der Methode der kleinsten Quadrate basierten Ansatz evaluiert.

6.1. Die verwendete Plattform

Um eine gewisse Vergleichbarkeit herzustellen, muss auf die verwendete Hard- und Software näher eingegangen werden. Das verwendete Dünngittertoolkit SG^{++} ist hochperformant implementiert und verwendet OpenMP zur Skalierung innerhalb eines Shared-Memory-Systems [Pfl10]. Dies wurde mit einem Intel-basierten 4-Sockel-System kombiniert, das als Prozessoren vier Intel Xeon E7540 nutzt. Diese Prozessoren besitzen je 6 Kerne und takten mit 2 Ghz[1]. Insgesamt liegen 24 Prozessorkerne vor, die dank Hyper-Threading 48 Threads gleichzeitig verarbeiten können. Jeder Prozessor besitzt außerdem 18MB Level-3 Cache und dem System stehen 512GB an Arbeitsspeicher zur Verfügung.

6.2. Benötigte Rechenzeit ohne Optimierungen

Da die Verwendung dünner Gitter im Vergleich zu voll besetzten Gittern bereits gute Rechenzeiten erwarten lässt, wird in diesem Abschnitt auf die benötigten Rechenzeiten ohne weitere Optimierungen eingegangen. In Abbildung 6.1 sind die Rechenzeiten für jeden der Datensätze aus dem letzten Kapitel dargestellt. Dabei wurde bei jedem Datensatz das Experiment ausgewählt, bei dem die höchste Trefferrate beobachtet werden konnte:

- „Kurve“: Die besten Ergebnisse wurden mit dem Differenz-Quad-Konstruktor mit Schrittanzahl $m = 3$ erreicht. Daraus folgt ein Attributraum mit Dimension 3.
- „Muster“: Der Linear-Konstruktor mit $m = 2$ ergab die höchste Trefferrate. Dies ergibt einen Attributraum mit Dimension 2.

[1] Durch den Turbomodus können die Prozessoren bis zu 2,26 Ghz erreichen.

- „DAX“: Hier wurden der Linear-Konstruktor und der Differenz-Konstruktor zusammen verwendet, wobei jeder Konstruktor mit Schrittanzahl $m = 2$ gewählt wurde. Das resultierende Problem hat damit 4 Dimensionen.
- „Dow Jones“: Bei diesem Datensatz konnte die höchste Trefferrate bei Verwendung zweier Zeitreihen beobachtet werden. Die Dimension entspricht der Anzahl der verwendeten Zeitreihen.
- „FX“: Der Differenz-Konstruktor mit $m = 3$ und daraus folgender Problemdimension 3 erreichte die höchste Trefferrate.

Abbildung 6.1 lässt den Schluss zu, dass bei allen Datensätzen eine Maximierung der Trefferrate mit relativ geringen Rechenzeitanforderungen zusammenfällt. Bei vier der fünf Datensätze wurden 1000 Testzeitpunkte verwendet, lediglich bei Experimenten mit dem „FX“-Datensatz bestand die Testdatenmenge aus 2000 Elementen. Da bei den synthetischen Experimenten kein Zeitmaßstab direkt vorgegeben ist, ist eine finale Einschätzung schwierig. Beim Datensatz „Kurve“ wurde die höchste Trefferrate bereits bei einem Gitter mit Level 3 erreicht, was aufgrund der niedrigen Anzahl an Gitterpunkten auf eine sehr geringe Rechenzeit für die einzelnen Testdatenpunkte hinweist.

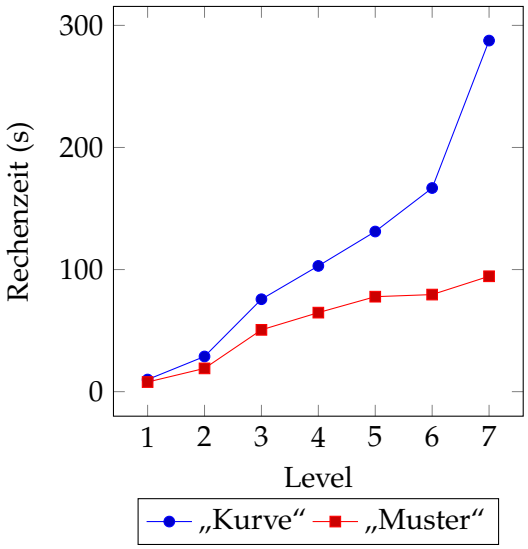
Interessanter ist der Datensatz „Muster“. Die beiden Klassen werden in diesem Datensatz durch scharfe Kanten begrenzt. Dadurch werden viele Gitterpunkte benötigt, um die Trefferrate zu maximieren. Entsprechend wird die höchste Trefferrate auch erst auf einem Gitter mit Level 6 erreicht. Dabei werden für die 1000 Testzeitpunkte insgesamt 80s Rechenzeit benötigt. Daraus folgt, dass 1 Sekunde Rechenzeit pro Zeitschritt bei einem Problem mit niedriger Dimension, das aber gleichzeitig ein feines Gitter benötigt, immer noch deutlich unterschritten wird.

Bei den nichtsynthetischen Datensätzen gelten ähnliche Aussagen. Beim Datensatz „DAX“ werden immerhin 4 Dimensionen benötigt und gleichzeitig ein Gitter mit Level 5. Bei diesem Datensatz wird damit am meisten Rechenzeit pro Zeitschritt aufgewendet, um die Trefferrate zu maximieren. Da die Daten bei diesem Datensatz minütlich vorliegen, kann der Datensatz trotzdem problemlos in Echtzeit verarbeitet werden.

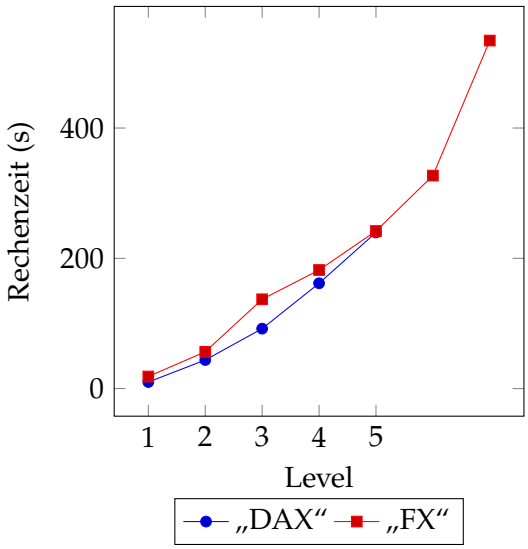
Für den „FX“-Datensatz gilt, dass die höchste Trefferrate bereits bei Level 2 erreicht wird, wodurch eine geringe Rechenzeit möglich wird, obwohl eine größere Testdatenmenge mit 2000 Zeitschritten verwendet wird. Für die gesamte Testdatenmenge werden 56s benötigt. Damit können 36 Zeitschritte pro Sekunde verarbeitet werden, wodurch auch bei diesem Datensatz, bei dem durchaus tatsächlich mehrere Zeitschritte pro Sekunde vorliegen, die Echtzeitanforderungen eingehalten werden können.

Ähnliches gilt für den „Dow Jones“-Datensatz. Für die beste Trefferrate wurden zwei Zeitreihen kombiniert. Da die höchste Trefferrate allerdings bei einem Gitter mit Level 2 erreicht wird und die Datenpunkte in Minutenabständen vorliegen, sind die Vorhersagen schnell genug verfügbar.

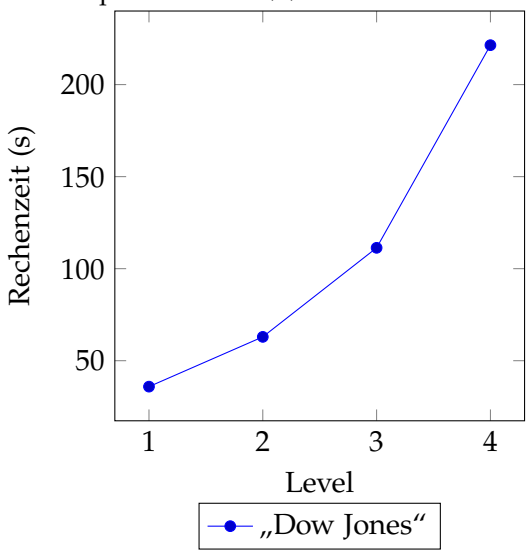
Generell ist bereits die unoptimierte Rechenzeit des Ansatzes gering genug, um einige Hundert bis wenige Tausend Gitterpunkte in Echtzeit verwenden zu können. Das vorgestellte „DAX“ Experiment benötigt zum Beispiel 769 Gitterpunkte. Durch die Verwendung



(a) Rechenzeit der synthetischen Experimente



(b) Rechenzeit der Datensätze „DAX“ und „FX“.



(c) Rechenzeit des „Dow Jones“-Datensatz

Abbildung 6.1.: Benötigte Rechenzeit für unterschiedliche Datensätze bei den Experimenten mit der höchsten Trefferrate.

stärkerer Hardware sollte noch eine deutliche Verbesserung erreicht werden können. Zudem werden durch das Regularisierungsverfahren aus Abschnitt 3.4 in jedem Zeitschritt mehrere Dünngitterfunktionen berechnet. Da jede Funktion unabhängig berechnet werden kann und nur die verwendeten Zählervariablen für die Vorhersage synchronisiert werden müssen, kann dieses Verfahren sehr einfach über mehrere Knoten parallelisiert werden.

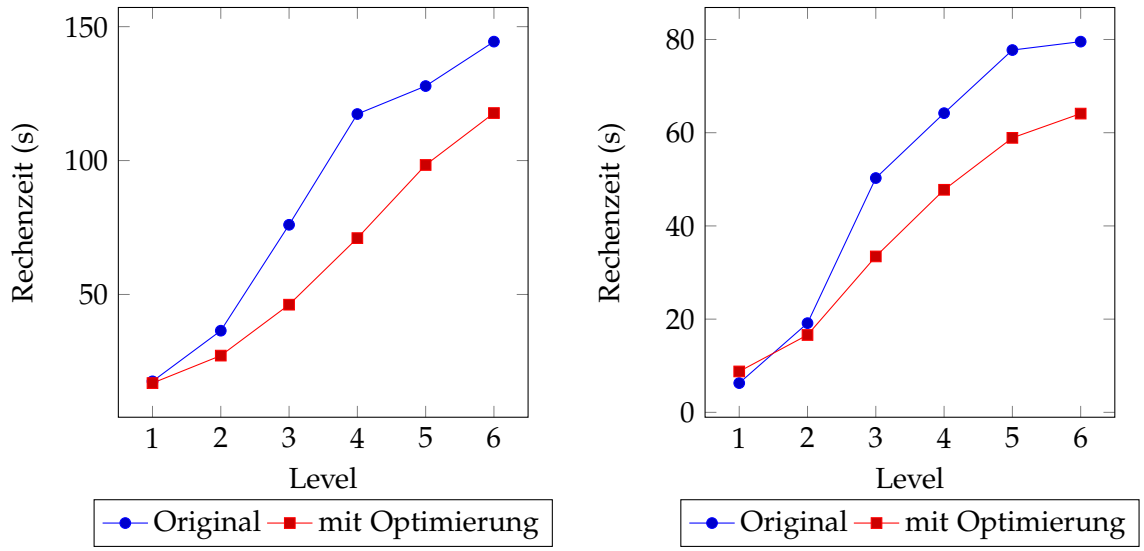
6.3. Ausgangspunkt und Methodik der Optimierungen

In vielen Anwendungsfällen kann eine längere Rechenzeit für die Berechnung der Dünngitterfunktion in Kauf genommen werden, da die Funktion anschließend zum Lösen eines nachgeschalteten Problems genutzt wird. Das heißt, die Berechnung der Dünngitterfunktion dominiert nicht die gesamte Rechenzeit. Der vorgestellte Algorithmus zur Zeitreihenanalyse dagegen erstellt eine Dünngitterfunktion pro Zeitschritt. Zudem wird diese Dünngitterfunktion in jedem Zeitschritt für lediglich eine einzige Auswertung verwendet: Es wird der nächste Wert vorhergesagt, anschließend wird auf den nächsten Zeitschritt gewartet.

Das hat zur Konsequenz, dass die Berechnung der Dünngitterfunktionen den Gesamtaufwand klar dominiert. Das heißt, die Minimierung der benötigten Zeit für die Berechnung der Dünngitterfunktion ist ein zentraler Ausgangspunkt zum Erreichen der Echtzeitfähigkeit in zeitlich restriktiveren Szenarien. Daneben spielt noch die Vorverarbeitung eine wichtige Rolle. Wird ein Attributraum mit zu hoher Dimension und damit zu vielen Gitterpunkte gewählt, kann dies die benötigte Rechenzeit ebenfalls erheblich erhöhen. An dieser Stelle wird allerdings versucht, Verbesserungen zu erzielen, die weitgehend unabhängig von einzelnen Datensätzen sind. Daher liegt der Fokus in den nächsten Abschnitten auf der Berechnung der Dünngitterfunktion.

Da effiziente Algorithmen für dünne Gitter und für das Data Mining auf dünnen Gittern bereits existieren [Pfl10, Fra11], werden nur Optimierungen betrachtet, die speziell für die Zeitreihenanalyse interessant sind. Ausgenutzt wird dabei, dass es sich bei der Zeitreihenanalyse um ein iteriertes Problem handelt, bei dem zwischen den Zeitschritten als einzige Änderung das Hinzufügen eines Datenpunktes und die Auswertung der berechneten Dünngitterfunktion an einer anderen Stelle besteht. Dies eröffnet mehrere Strategien zur Optimierung, die in den folgenden Abschnitten vorgestellt werden.

Als Datensätze zur Untersuchung der Optimierungen wurden der synthetische Datensatz „Muster“ und der nichtsynthetische Datensatz „FX“ ausgewählt. Wo sinnvoll möglich wurden Experimente ausgewählt, die gleichzeitig eine hohe Trefferrate besitzen, damit Auswirkungen der Optimierungen auf die Trefferrate ebenfalls berücksichtigt werden können.



(a) „FX“-Datensatz **(b)** „Muster“-Datensatz

Abbildung 6.2.: Koeffizienten-Wiederverwertung mit zwei Datensätzen. Dabei wird die Rechenzeit mit und ohne Wiederverwertung der Koeffizienten verglichen.

6.4. Wiederverwertung des Koeffizientenvektors über mehrere Zeitschritte

Dadurch, dass das zu lösende Vorhersageproblem zwischen zwei Zeitschritten sehr ähnlich ist, wird erwartet, dass auch die zum Lösen der Aufgabe berechneten Dünngitterfunktionen sehr ähnlich sind. Da mit dem Konjugierte-Gradienten-Verfahren ein iteratives Verfahren zum Lösen der Gleichungssysteme verwendet wird, kann die benötigte Rechenzeit verringert werden, indem ein guter Startvektor für das CG-Verfahren geraten wird, durch den die Fehlerschranke des Lösers mit möglichst wenig Iterationen erreicht wird. Aufgrund der Verwandtschaft der Probleme über die Zeit steht ein passender Kandidat als Startvektor bereit. Es wird der Koeffizientenvektor $\overline{\alpha}$ der Dünngitterfunktion des letzten Zeitschritts als Eingabe für den Löser verwendet. Damit diese Optimierung durchgeführt werden kann, ist es notwendig, das gleiche Gitter wie im letzten Zeitschritt zu verwenden.

Zur Evaluierung wurden bei beiden betrachteten Datensätzen die besten einzelnen Konstruktoren mit Schrittanzahl $m = 2$ verwendet. Als Testdatenmenge wurde beim „FX“-Datensatz wie üblich 2000 Zeitschritte verwendet, beim „Muster“-Datensatz bestand die Testdatenmenge aus 1000 Elementen. Die Ergebnisse sind in Abbildung 6.2 zu sehen, wobei die Dauer jeweils einmal mit und einmal ohne die Wiederverwertung des Koeffizientenvektors dargestellt wird.

Bei beiden Datensätzen kann eine deutliche Reduktion der benötigten Rechenzeit beobachtet werden, wenn der Koeffizientenvektor wiederverwertet wird und mehr als ein Gitterpunkt vorliegt. Die erzielte Reduktion der Rechenzeit kann dabei bis zu einem Drittel der unop-

timierten Rechenzeit betragen, wobei der Nutzen mit steigendem Level scheinbar wieder geringer wird.

Ein großer Vorteil dieses Ansatzes ist, dass durch die Wiederverwertung der Koeffizienten keine Verschlechterung der Vorhersagen zu befürchten ist. Zudem wurde kein relevantes Experiment beobachtet, bei dem die Wiederverwertung der Koeffizienten zu mehr Rechenzeit führt.

6.5. Levelreduktion durch Adaptivität

Der unmittelbare Grund für die höhere Rechenzeit auf einem Gitter mit höherem Level oder der Verwendung einer höheren Dimension ist die resultierende größere Anzahl an Gitterpunkten. Durch adaptive Verfeinerung kann die Gesamtanzahl an Gitterpunkten für ein gegebenes Problem häufig stark reduziert werden. Zusätzlich können bei dem vorliegenden iterierten Problem die Kosten für die Verfeinerungsschritte auf die Zeitschritte verteilt werden.

Durch die Ähnlichkeit der Probleme über die Zeit ist eine Wiederverwertung der Gitter möglich. Da sich die Datenpunkte nur langsam über die Zeit ändern, gilt dies auch für durch Verfeinerung weiter angepasste Gitter. Da es durch Verfeinerung möglich ist, nur in Regionen Gitterpunkte zu verwenden, wo diese auch tatsächlich benötigt werden, kann auf einem deutlich niedrigeren initialen Level gerechnet werden. In den ersten Zeitschritten ist dann ein etwas erhöhter Rechenbedarf notwendig, um einmalig ein passend verfeinertes Gitter zu erstellen. Diese initiale Verfeinerung wird hier als unproblematisch bewertet, da sie als Teil einer Aufwärmphase betrachtet werden kann. Anschließend wird ein Gitter verwendet, das ähnliche Trefferraten besitzt, wie sie bei einem unverfeinerten Gitter auf höherem Level zu messen sein sollten. Da die Gesamtanzahl der Gitterpunkte geringer ist, sind auch geringere Rechenzeiten zu erwarten.

Eine initiale Verfeinerung allein reicht nicht, um die Änderungen im Attributsraum beim Durchlaufen einer Zeitreihe korrekt widerzuspiegeln. Die neuen Punkte im Attributsraum erfordern weitere Verfeinerungsschritte. Da sich die Datenpunkte aber nur langsam ändern, reichen sehr wenig Verfeinerungsschritte in jedem betrachteten Zeitschritt aus. Eine Verfeinerung in jedem Zeitschritt hat die weitere negative Konsequenz für die Laufzeit, dass sich die Anzahl der Gitterpunkte beim Durchlaufen der Zeitreihe ständig erhöht. Damit der Vorteil der geringeren Anzahl an Gitterpunkten bei diesem Verfahren nicht verloren geht, ist es daher notwendig, Gitterpunkte wieder zu entfernen. Wie in Abschnitt 2.7 erklärt, können als einfache Vergröberungsstrategie Gitterpunkte mit niedrigen Koeffizientenwerten entfernt werden. Dies muss immer durchgeführt werden, wenn die Gesamtanzahl der Gitterpunkte zu hoch wird.

Der gesamte Algorithmus lässt sich damit wie folgt beschreiben. Es wird eine gewünschte Anzahl an Gitterpunkten t vorgegeben, die so gewählt sein muss, dass sich eine ausreichend gute Vorhersagequalität damit erreichen lässt. In der Aufwärmphase wird durch eine vorgegebene Anzahl an Modifikationsschritten s ein möglichst passendes Gitter aufgebaut.

Algorithmus 6.1 Verfeinerungs- und Vergröberungsschema

```
procedure CreateSparseGridFunction(trainingSet, λ, lastGrid)
    f ← SolveLeastSquareProblem(trainingSet, lastGrid)
    ᾱ ← GetCoefficients(f)
    gridPoints ← GetSize(lastGrid)
    for t = 1 → refinementSteps do
        if gridPoints ≤ targetGridPoints then
            newGrid ← RefineGrid(lastGrid, ᾱ)
        else
            newGrid ← EncoarsenGrid(lastGrid, ᾱ)
        end if
        f ← SolveLeastSquareProblem(trainingSet, newGrid)
    end for
end procedure
```

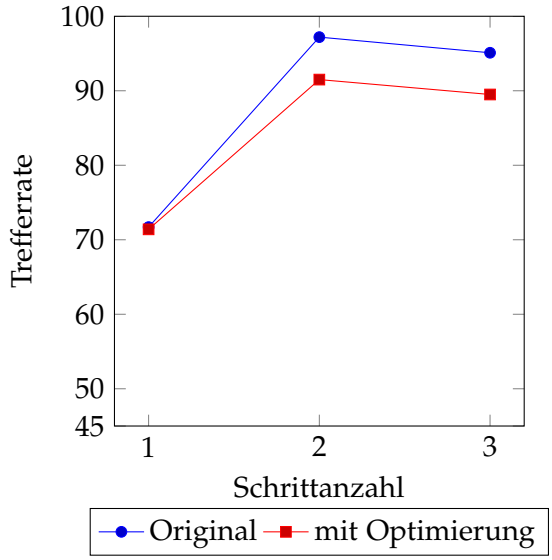
Dabei wird so lange verfeinert, bis die t Gitterpunkte überschritten werden. Dann wird vergröbert, bis wieder weniger als t Gitterpunkte vorliegen. Das resultierende Gitter sollte nach den vorgegebenen s Schritten gut auf die Datenpunkte passen. Während des Durchlaufens der Zeitreihe wird eine kleine Anzahl an Modifikationsschritten in jedem Zeitschritt durchgeführt, wodurch ein passendes Gitter in jedem Zeitschritt vorliegt.

Das kontinuierliche Verfeinern und Vergröbern an den einzelnen Zeitschritten ist in Algorithmus 6.1 dargestellt. Mit diesem Verfahren pendelt die Anzahl der Gitterpunkte um den vorgegebenen Wert, wobei in jedem Zeitschritt ein passend verfeinertes Gitter vorliegt.

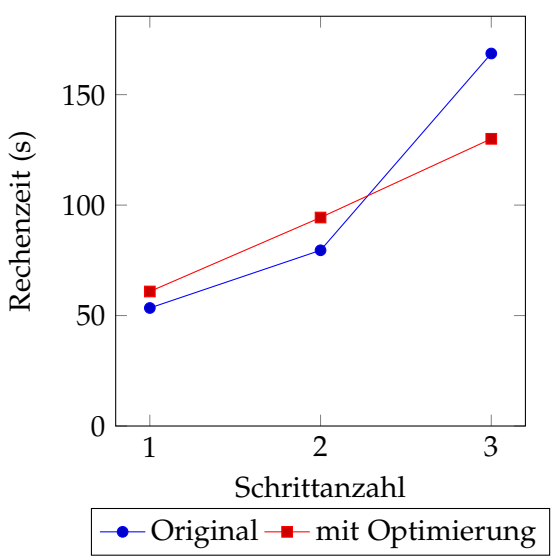
Für die hier vorgestellten Experimente wird lediglich ein Modifikationsschritt an jedem Zeitpunkt durchgeführt. Dabei werden jeweils 5 Gitterpunkte verfeinert. Falls die Anzahl der Gitterpunkte größer oder gleich der vorgegebenen Zahl an Gitterpunkten ist, wird versucht, die Differenz zur gewünschten Anzahl an Gitterpunkten zuzüglich 5 weiterer Gitterpunkte zu entfernen. Da nur Blattknoten der hierarchischen Basis entfernt werden können, wird die Anzahl der zu entfernenden Gitterpunkte bei Misserfolgen im nächsten Zeitschritt jeweils um zwei pro erfolglosem Versuch erhöht.

Bei diesem Ansatz müssen zwei Aspekte beachtet werden. Werden Gitterpunkte mittels adaptiver Verfeinerung hinzugefügt oder mittels adaptiver Vergröberung entfernt, dann müssen im Anschluss die Koeffizienten der zugehörigen Dünngitterfunktion neu berechnet werden. Das heißt, es muss erneut ein Gleichungssystem gelöst werden, was vergleichsweise teuer ist. Im Allgemeinen wird dieser Ansatz umso besser funktionieren, je mehr unnötige Gitterpunkte durch adaptive Verfeinerung eingespart werden können, da dann kleinere und schneller lösbare Gleichungssysteme zu lösen sind.

Des Weiteren erfordert es dieser Ansatz, dass eine passende Anzahl an Gitterpunkten für adaptiv angepasste Gitter gewählt wird. Werden zu wenig Gitterpunkte gewählt, dann nimmt die Vorhersagequalität ab. Werden zu viele Gitterpunkte verwendet, kann der Vorteil

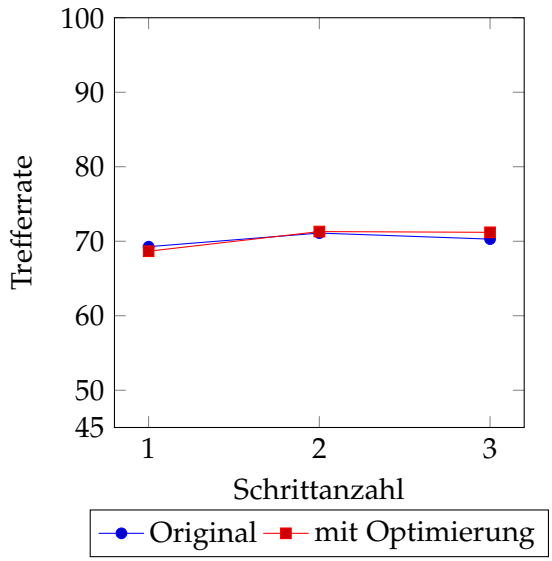


(a) Qualität der Vorhersage

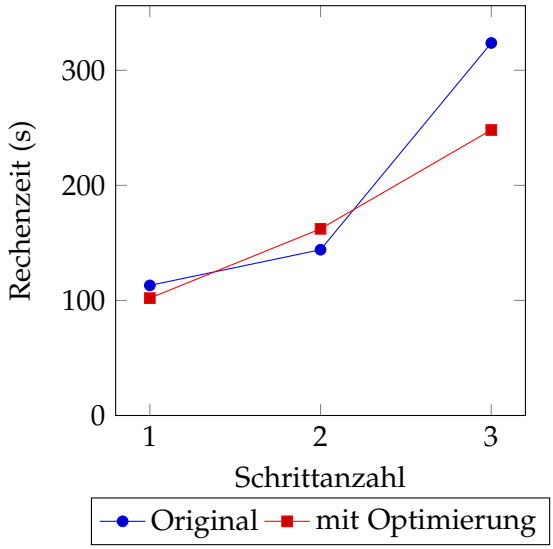


(b) Rechenzeit abhängig von der Dimension des Problems.

Abbildung 6.3.: Adaptivität bei niedrigerem Level angewendet auf den „Muster“-Datensatz.



(a) Qualität der Vorhersage



(b) Rechenzeit abhängig von der Dimension des Problems.

Abbildung 6.4.: Adaptivität bei niedrigerem Level angewendet auf den „FX“-Datensatz.

der geringeren Rechenzeit entfallen. Letzteres gilt dabei weniger wegen der Anzahl der Gitterpunkte, sondern vor allem aufgrund des Lösens weiterer Gleichungssysteme.

Aufgrund dieser Eigenheiten des Ansatzes wurden Experimente mit relativ hoher Dimension und Level durchgeführt. Variiert wurde die Schrittanzahl der Konstruktoren und damit die Dimension. Das Gitter wurde für die unoptimierten Durchläufe mit Level 6 gewählt. Für die Experimente mit Adaptivität wurde ein Gitter mit Level 3 gewählt, wobei als Ziel der Verfeinerungs- und Vergröberungsstrategie 500 Gitterpunkte vorgegeben wurden. Dieser Parameter wurde während der Experimente nicht variiert, was die Rechenzeit des adaptiven Ansatzes bei niedrigerer Dimension erhöht.

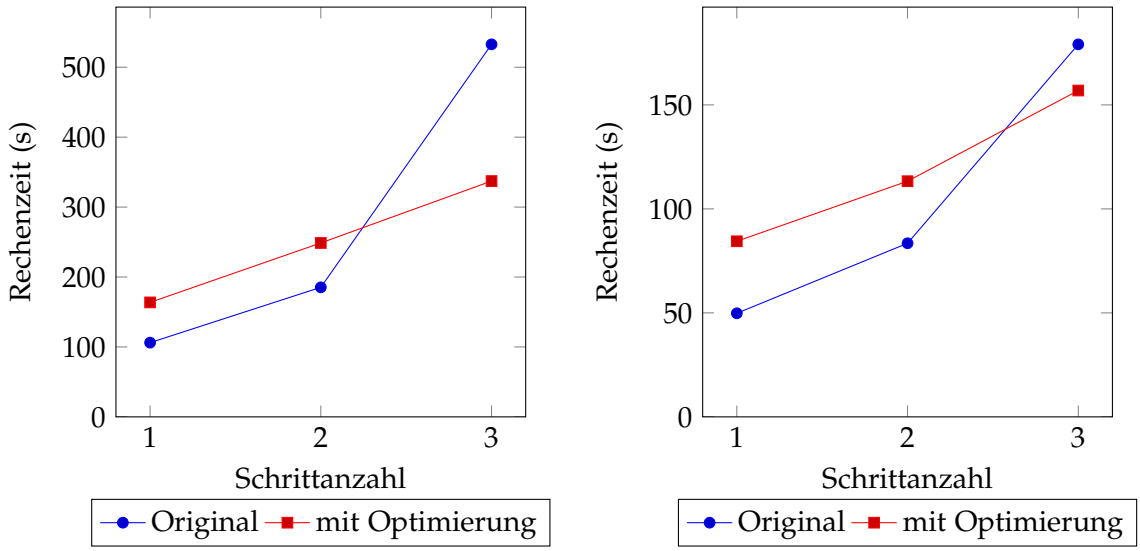
Die Ergebnisse dieser Experimente für den Datensatz „Muster“ sind in Abbildung 6.3 zu sehen. Bei niedriger Dimension ist die Verwendung des adaptiven Ansatzes etwas teurer, die Rechenzeit ist höher als beim unoptimierten Ansatz. Sobald drei Dimensionen vorliegen, dreht sich dies jedoch, da die Anzahl der Gitterpunkte auch auf einem dünnen Gitter mit zunehmender Dimension exponentiell steigt. Für noch höhere Dimensionen wird erwartet, dass noch mehr Rechenzeit vermieden werden kann. Gleichzeitig ist eine leichte Verringerung der Qualität der Vorhersagen zu beobachten. Es wird jedoch davon ausgegangen, dass dies durch eine bessere Wahl der gewünschten Anzahl an Gitterpunkten kompensiert werden kann. Es ist allerdings möglich, dass der Rechenzeitvorteil dann erst bei einer noch höheren Dimension sichtbar wird.

Ähnliche Ergebnisse zeigt der „FX“-Datensatz, wie in Abbildung 6.4 zu sehen ist. Auch hier zeigt die Optimierungen eine Verringerung der Rechenzeit bei einem Problem in drei Dimensionen. Die Qualität der Vorhersage ist dabei vergleichbar mit den unoptimierten Ergebnissen. Allerdings ist zu bedenken, dass zum Erreichen sehr guter Vorhersagen mit diesem Datensatz in Abschnitt 5.3.3 gezeigt wurde, dass hierfür bereits sehr wenig Gitterpunkte ausreichen.

Der vermutete Rechenzeitvorteil durch die Verwendung der adaptiven Verfeinerungsstrategie konnte in den durchgeführten Experimenten bestätigt werden. Die Wahl einer passenden Anzahl an Gitterpunkten hat sich in der Praxis allerdings als schwierig erwiesen.

6.6. Zoom-Ansatz

Eine Alternative zu den vorgeschlagenen Strategien entsteht durch eine spezielle Betrachtung des Attributraums. In jedem Zeitschritt ist bekannt, wo die zu berechnende Dünngitterfunktion ausgewertet werden soll. Außerdem findet genau eine Auswertung statt. Das heißt, es wird eigentlich nicht die vollständige Funktion benötigt, sondern eine möglichst gute Approximation der Funktion an dem Punkt, an dem ausgewertet wird. Da Datenpunkte, die weit entfernt vom Ort der Auswertung liegen, nur sehr geringen Einfluss auf die Auswertung am aktuellen Punkt besitzen, kann bei weiter entfernten Datenpunkten eine lediglich sehr grobe Approximation verwendet werden. Das wiederum bedeutet, dass weniger Gitterpunkte verwendet werden können, wodurch die Rechenzeit verringert wird.



(a) „FX“-Datensatz **(b)** „Muster“-Datensatz

Abbildung 6.5.: Verfeinern der Umgebung mit zwei Datensätzen. Dabei wird jeweils die Schrittanzahl und damit die Dimension des Attributraums variiert.

Eine Möglichkeit, dies zu realisieren, ist eine Verfeinerungsstrategie, die, wie im letzten Abschnitt beschrieben, abwechselnd Vergröberungs- und Verfeinerungsschritte verwendet. Dabei werden allerdings Gitterpunkte stärker berücksichtigt, die sich näher am Ort der Auswertung befinden. Konkret wird zum Punkt der Auswertung hin verfeinert, bei den durchgeführten Experimenten wurden konkret die 10 nächsten Gitterpunkte verfeinert, allerdings nur, falls sie Blattknoten darstellen.

Auch mit dieser Optimierungsstrategie wurden Experimente durchgeführt, deren Ergebnisse in Abbildung 6.5 vorliegen. Es kann dabei erneut eine Reduktion der benötigten Rechenzeit beobachtet werden, im Gegensatz zum zuvor vorgestellten Ansatz auch bereits bei zwei Dimensionen. Ähnlich wie beim letzten Ansatz ist ein leichtes Absinken der Vorhersagequalität zu beobachten. Beim „Muster“-Datensatz konnte in zwei Dimensionen keine Verschlechterung der Trefferrate beobachtet werden, sie lag mit und ohne Optimierung bei 97%. Bei zwei Dimensionen sank die Trefferrate von 95% auf 93%. Beim „FX“-Datensatz war die Trefferrate auf gleichem Niveau mit und ohne Verfeinerung der Umgebung. Sie lag bei 68% in einer Dimension und bei 71% in zwei Dimensionen.

Dieser Ansatz kann weiter ausgebaut werden. Bisher wurde argumentiert, dass Gitterpunkte nur in der Nähe des Ortes der Auswertung benötigt werden. Es ist naheliegend zu vermuten, dass dies auch auf die Datenpunkte zutrifft, da weiter entfernte Datenpunkte ebenfalls eine geringe Rolle für die aktuelle Auswertung spielen sollten. Um dies algorithmisch umzusetzen, wird eine Entfernung gewählt, ab der Datenpunkte nicht mehr als Teil der Trainingsmenge berücksichtigt werden. Hier wurde eine Distanz von $0,2$ im betrachteten Hyperwürfel gewählt, wobei als Metrik die euklidische Metrik zum Einsatz kam.

Bei Experimenten wurde festgestellt, dass die Rechenzeit deutlich verringert werden konnte. Da allerdings gleichzeitig die Trefferrate drastisch sank, wurde diese Idee nicht weiter verfolgt. Eine akzeptable Trefferrate konnte erst wieder erst wieder erreicht werden, als wieder nahezu alle Punkte berücksichtigt wurden.

Das Verfeinern der Umgebung um den Auswertungspunkt kann erfolgreich zur Reduktion der Rechenzeit eingesetzt werden. Zusätzlich kann dieser mit der zuvor vorgestellten Wiederverwertung des Koeffizientenvektors kombiniert werden. Dies führt zu weiter reduzierten Rechenzeiten. Wie beim allgemeinen adaptiven Ansatz aus Abschnitt 6.5 ist es auch hier wieder schwierig eine geeignete Anzahl an Gitterpunkten zu wählen, damit die Vorhersagen qualitativ vergleichbar bleiben.

Dieser Ansatz besitzt einige Nachteile. Dadurch, dass immer neue lokale Probleme betrachtet werden, sinkt die Qualität der Regularisierung mit dem in 3.4 vorgestellten Verfahren. Der Grund hierfür ist, dass die lokalen Probleme strukturell unterschiedlich beschaffen sein können und daher andere Werte für den Regularisierungsparameter benötigen.

7. Zusammenfassung und Ausblick

In dieser Arbeit wurde die Zeitreihenanalyse auf dünnen Gittern betrachtet. Es wurden auf dünnen Gittern basierte Algorithmen vorgestellt, die eine Lösung des Vorhersageproblems für Zeitreihen ermöglichen. Zudem sollte sichergestellt werden, dass die vorgestellte Lösung auch in zeitkritischen Anwendungen nutzbar ist, weshalb einige Optimierungen zur Verkürzung der benötigten Rechenzeit vorgestellt wurden.

Der vorgestellte Ansatz basiert auf dünnen Gittern, mit denen es möglich ist, Regressions- und Klassifikationsprobleme aus dem Bereich des Data Minings zu lösen. Auf diesen Verfahren aufbauend wurde ein Algorithmus vorgestellt, mit dem zukünftige Werte von Zeitreihen durch Formulierung als Regressions- oder Klassifikationsproblem berechnet werden können. Dafür wurden Datenpunkte des Vorhersageproblems als Auswertungen einer unbekannten Funktion interpretiert, die einem gegebenen Wertetupel den Wert im nächsten Zeitschritt zuordnet. Mit Methoden aus dem Bereich des Data Minings kann diese unbekannte Funktion approximiert werden.

Für eine erfolgreiche Approximation der Vorhersagefunktion ist es notwendig die Daten der Zeitreihen so vorzubereiten, dass durch die Data Mining Algorithmen auch tatsächlich eine passende Vorhersagefunktion approximiert wird.

Zur Validierung des Ansatzes wurden Experimente mit fünf Datensätzen durchgeführt, von denen zwei der Datensätze synthetische Daten beinhalten und drei der Datensätze aus nichtsynthetischen Daten bestehen. Die Vorhersagen für die synthetischen Daten entsprachen exakt den gestellten Erwartungen, wobei die Trefferrate korrekter Vorhersagen sehr hoch war. Bei den nichtsynthetischen Datensätzen wurden deutlich weniger, allerdings akzeptabel viele korrekte Vorhersagen gemessen. Dies ist vermutlich zum Teil auf die Qualität der Daten zurückzuführen und zum Teil auf das Problem, dass eine geeignete Vorverarbeitung von real gemessenen Daten äußerst schwierig ist. Die gemessene Vorhersagequalität war dabei mit ähnlichen Ansätzen vergleichbar.

Bei den durchgeführten Experimenten wurde festgestellt, dass die benötigte Rechenzeit für gute Vorhersagen bereits vor Anwendung der vorgestellten Optimierungen relativ gering war. Generell erfordern unterschiedliche Datensätze unterschiedliche Attributsräume und gerade Attributsräume mit hoher Dimension und vielen Gitterpunkten benötigen unter Umständen große Mengen an Rechenzeit. Da der vorgestellte Ansatz auf dünnen Gittern basiert, tritt dieses Problem im Vergleich zu vollen Gittern weniger stark zutage. Durch die zusätzlichen Optimierungen wurde eine weitere Verbesserung der Laufzeit beobachtet. Der zentrale Ansatzpunkt für die vorgestellten Optimierungen war dabei, dass sich bei iterierten Vorhersagen das Problem zwischen den Zeitschritten nur geringfügig ändert, wodurch in der Vergangenheit gewonnene Informationen weiter genutzt werden können.

Alles in allem besitzt der vorgestellte Ansatz eine gute Performance und insbesondere bei den Experimenten mit synthetischen Datensätzen eine sehr hohe Genauigkeit der Vorhersagen, die weitere Experimente mit nichtsynthetischen Daten wünschenswert erscheinen lassen. Die vorgestellten Optimierungen können dabei das Ziel erreichen, die Einsatzgebiete, bei denen Vorhersagen in Echtzeit berechnet werden können, zu vergrößern.

Ausblick

Auf dem vorgestellten Ansatz aufbauend, sind weitere Optimierungen möglich. Numerische Verfahren wie die Vorkonditionierung der Gleichungssysteme der einzelnen Verfahren können die bei einem gegebenen Verfahren benötigte Rechenzeit weiter reduzieren. Auch ist der Ansatz, Informationen über mehrere Zeitschritte hinweg zu nutzen, wahrscheinlich noch nicht ausgereizt, wodurch weitere Performanceverbesserungen möglich werden.

Während die Qualität der Vorhersagen insgesamt als gut zu bewerten ist, sollten dennoch weitere Experimente durchgeführt werden. Besonders Experimente mit nichtsynthetischen Datensätzen, die eine hohe Qualität besitzen und gleichzeitig gut verstanden sind, wären äußerst sinnvoll. Durch derartige Experimente könnte der Nutzen des Ansatzes deutlich besser abgeschätzt werden, da die für erfolgreiche Vorhersagen kritische Vorverarbeitung vereinfacht würde.

Bei der Durchführung der Experimente hat sich das Problem, eine geeignete Vorverarbeitung für die Daten zu finden, als zentral für erfolgreiche Vorhersagen herausgestellt. Zudem kann durch eine geeignete Vorverarbeitung eventuell ein Raum von niedriger Dimension gewählt werden, in dem unnötige Dimensionen nicht in den Datenraum eingebaut werden. Ebenfalls kann die Verwendung passender Attributskonstruktoren dazu führen, dass Muster im Attributsraum zu finden sind, die sich mit weniger Datenpunkten gut repräsentieren lassen. Das heißt, eine geeignete Vorverarbeitung kann sowohl zu besseren Vorhersagen als auch zu schnelleren Vorhersagen führen.

Die Vorverarbeitung von Zeitreihen kann als Optimierungsproblem aufgefasst werden, das durch die Wahl der Parameter bei der Konstruktion des Attributsraums gegeben ist. Ansätze zur Lösung dieses Problems würden vermutlich die größten Verbesserungen bei Verwendung der vorgestellten Methode oder verwandter Methoden erlauben. Allerdings gilt letztlich auch hier:

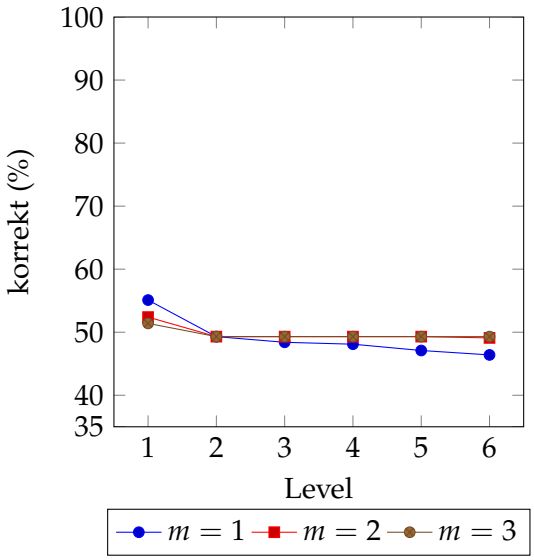
> „A lack of information cannot be remedied by any mathematical trickery.“
> - Cornelius Lanczos [Lan61]

A. Anhang

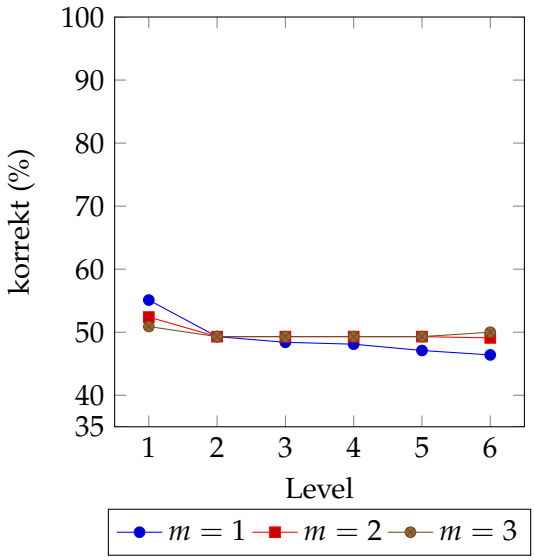
A.1. Evaluation des Dichte-basierten Ansatzes

Bei der Evaluation des Dichte-basierten Ansatzes wurden ähnliche Trefferraten beobachtet, wie sie auch bereits bei Verwendung der Methode der kleinsten Quadrate beobachtet wurden. Wie die Abbildungen auf den nächsten Seiten zeigen, werden bei den einzelnen Experimenten nahezu identische Trefferrate bei den einzelnen Datensätzen erreicht.

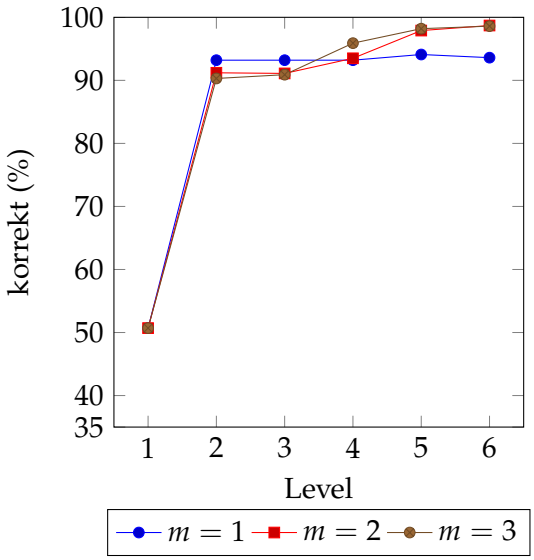
Für die Experimente bei diesem Ansatz etwa die doppelte Rechenzeit benötigt. Der Grund dafür ist, dass beim Dichte-basierten Ansatz eine Dünngitterfunktion pro Klasse des Problems berechnet werden muss. Wie in 2.4 beschrieben, gilt dies jedoch auch für den auf der Methode der kleinsten Quadrate basierenden Ansatz, da bei diesem bei mehr als zwei Klassen auch mehrere Dünngitterfunktionen verwendet werden sollten.



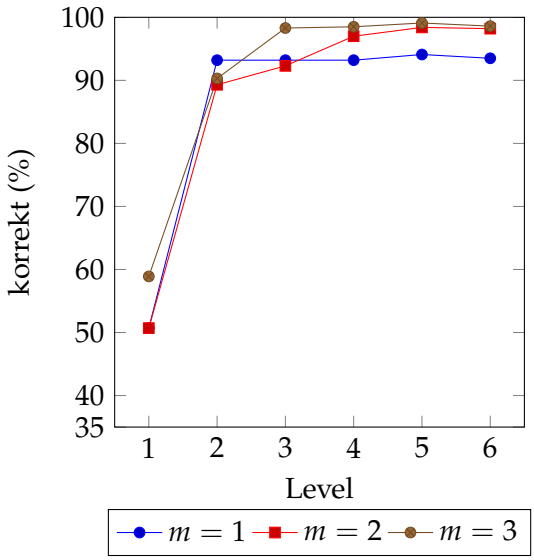
(a) Linear-Konstruktor



(b) Linear-Quad-Konstruktor

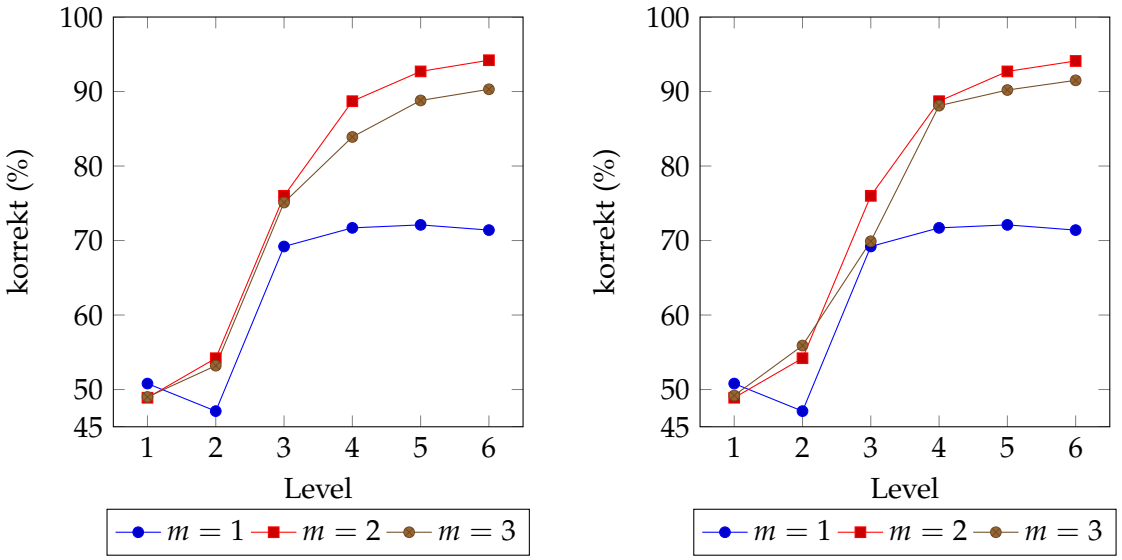


(c) Differenz-Konstruktor



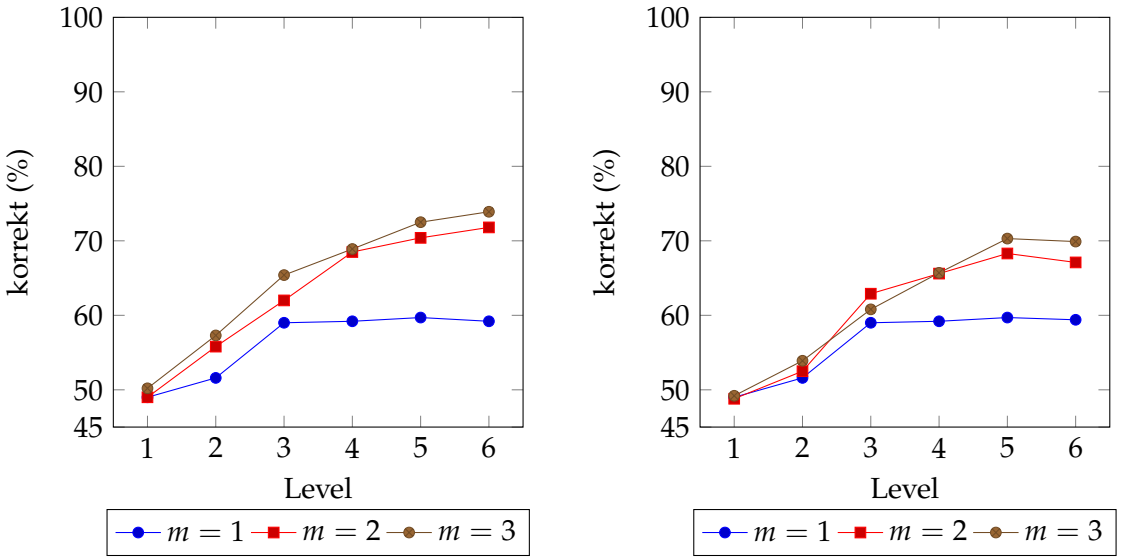
(d) Differenz-Quad-Konstruktor

Abbildung A.1.: Qualität der Vorhersagen bei Verwendung des Dichte-basierten Ansatzes bei verschiedener Attributkonstruktoren angewendet auf den „Kurve“-Datensatz. Dabei wird der Level des Gitters bei jedem Konstruktor variiert.



(a) Linear-Konstruktor

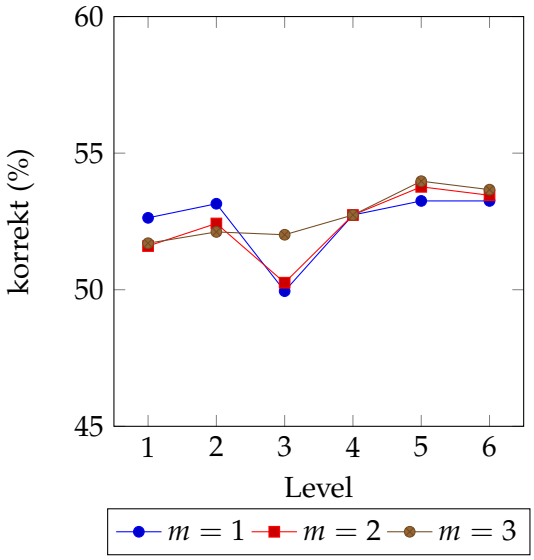
(b) Linear-Quad-Konstruktor



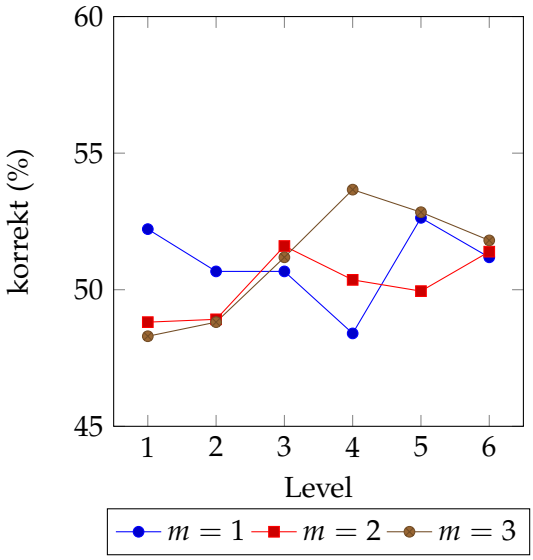
(c) Differenz-Konstruktor

(d) Differenz-Quad-Konstruktor

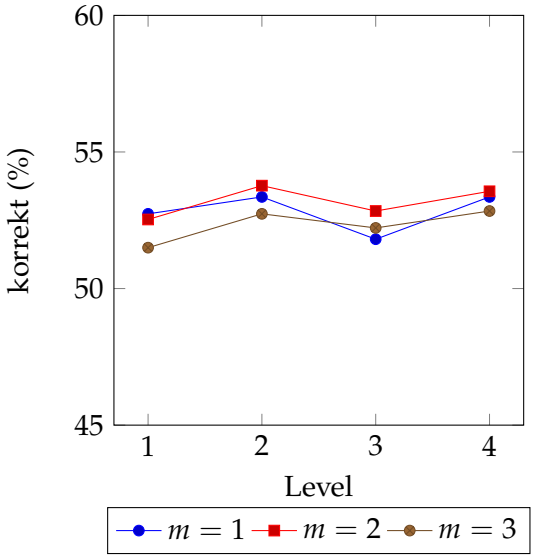
Abbildung A.2.: Qualität der Vorhersagen bei Verwendung des Dichte-basierten Ansatzes bei verschiedener Attributkonstruktoren angewendet auf den „Muster“-Datensatz. Dabei wird der Level des Gitters bei jedem Konstruktor variiert.



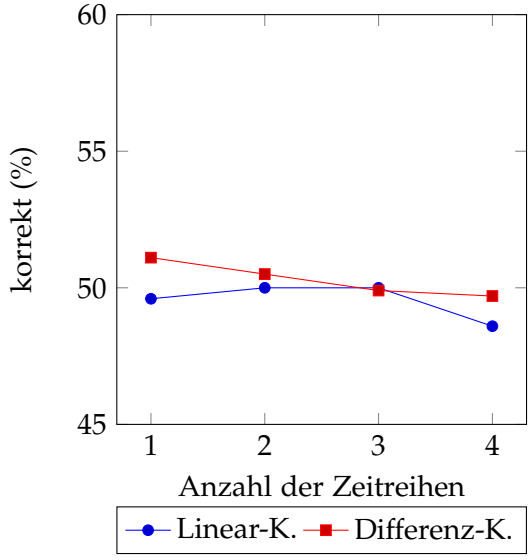
(a) Linear-Konstruktor



(b) Differenz-Konstruktor

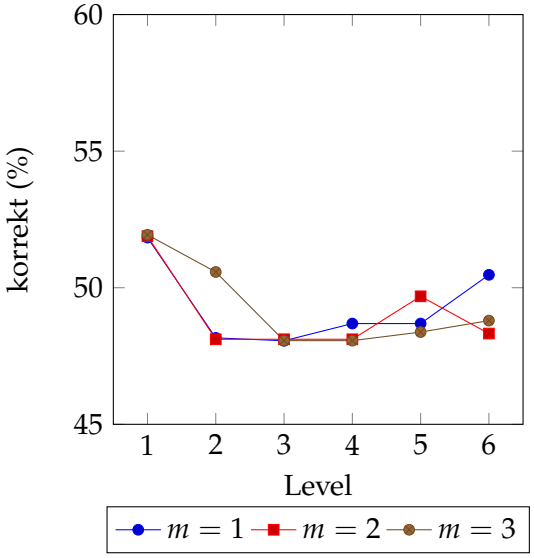


(c) Linear-Konstruktor und Differenz-Konstruktor

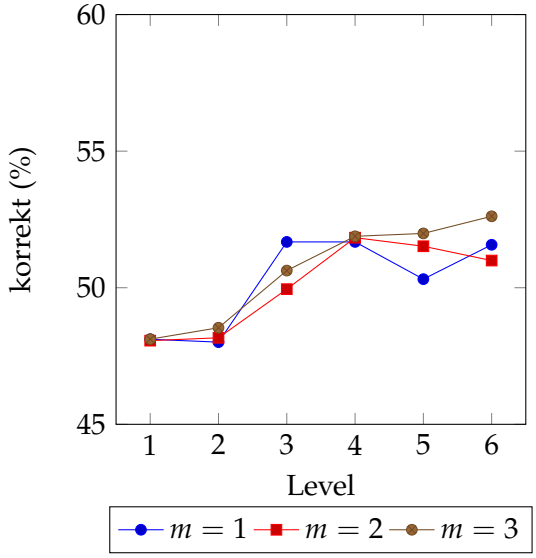


(d) Trefferrate bei unterschiedlicher Anzahl an verwendeten Zeitreihen

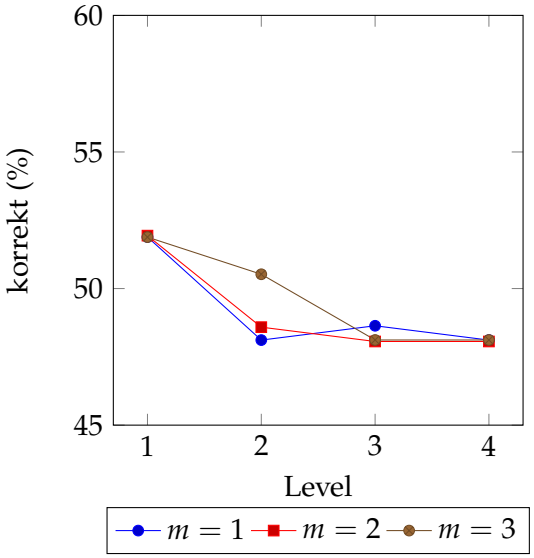
Abbildung A.3.: Qualität der Vorhersagen bei Verwendung des Dichte-basierten Ansatzes verschiedener Attributkonstruktoren angewendet auf den „DAX"-Datensatz. In den ersten drei Abbildungen wird der Level des Gitters zusätzlich variiert. Die vierte Abbildung zeigt die Trefferraten, wenn mehrere Zeitreihen zur Vorhersage verwendet werden.



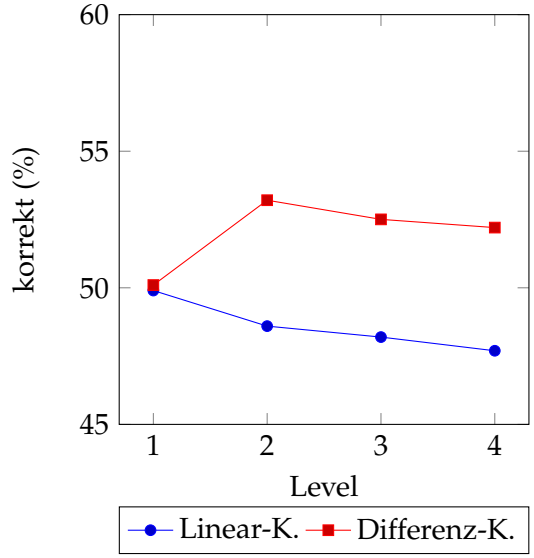
(a) Linear-Konstruktor



(b) Differenz-Konstruktor

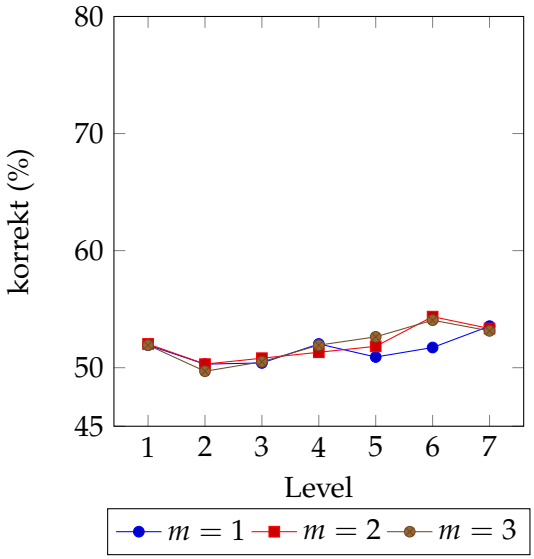


(c) Linear-Konstruktor und Differenz-Konstruktor

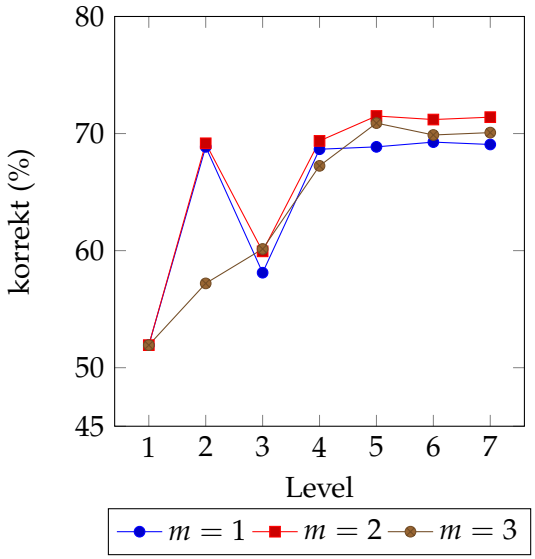


(d) Trefferrate bei unterschiedlicher Anzahl an verwendeten Zeitreihen

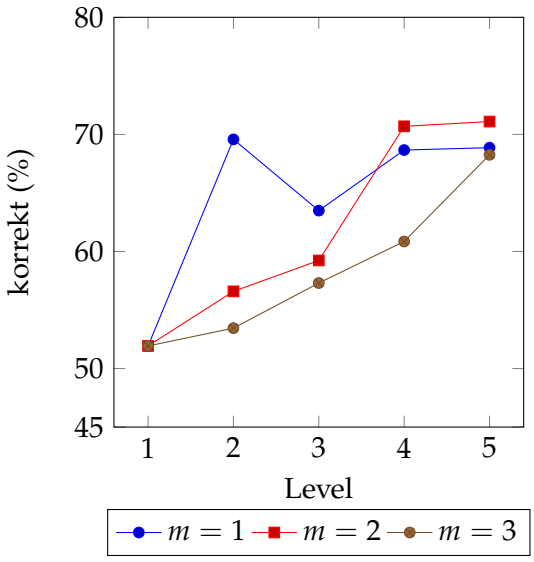
Abbildung A.4.: Qualität der Vorhersagen bei Verwendung des Dichte-basierten Ansatzes verschiedener Attributkonstruktoren angewendet auf den „Dow Jones"-Datensatz. In den ersten drei Abbildungen wird der Level des Gitters zusätzlich variiert. Die vierte Abbildung zeigt die Trefferraten, wenn mehrere Zeitreihen zur Vorhersage verwendet werden.



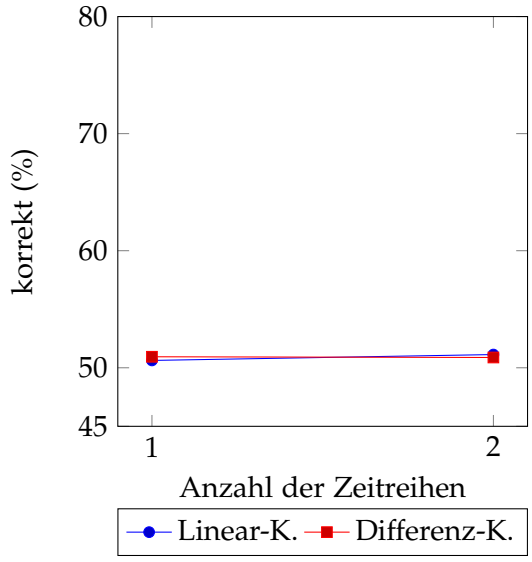
(a) Linear-Konstruktor



(b) Differenz-Konstruktor



(c) Linear-Konstruktor und Differenz-Konstruktor



(d) Trefferrate bei unterschiedlicher Anzahl an verwendeten Zeitreihen

Abbildung A.5.: Qualität der Vorhersagen bei Verwendung des Dichte-basierten Ansatzes verschiedener Attributkonstruktoren angewendet auf den „FX“-Datensatz. In den ersten drei Abbildungen wird der Level des Gitters zusätzlich variiert. Die vierte Abbildung zeigt die Trefferraten, wenn mehrere Zeitreihen zur Vorhersage verwendet werden, dabei wird eine stark modifizierte Zeitreihe verwendet.

Literaturverzeichnis

[Bel61] R. Bellman. *Adaptive Control Processes: A Guided Tour*. 'Rand Corporation. Research studies. Princeton University Press, 1961. (Zitiert auf den Seiten 9 und 13)

[BG04] H.-J. Bungartz, M. Griebel. Sparse grids. *Acta Numerica*, 13:1–123, 2004. (Zitiert auf den Seiten 9 und 21)

[BG13] B. Bohn, M. Griebel. An Adaptive Sparse Grid Approach for Time Series Prediction. In J. Garcke, M. Griebel, Herausgeber, *Sparse Grids and Applications*, Band 88 von *Lecture Notes in Computational Science and Engineering*, S. 1–30. Springer Berlin Heidelberg, 2013. (Zitiert auf Seite 13)

[Bis06] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., 2006. (Zitiert auf Seite 36)

[BJR08] G. E. Box, G. M. Jenkins, G. C. Reinsel. *Time Series Analysis: Forecasting and Control*. Wiley Series in Probability and Statistics. Wiley, 2008. (Zitiert auf den Seiten 9 und 31)

[CT03] L. J. Cao, F. Tay. Support vector machine with adaptive parameters in financial time series forecasting. *Neural Networks, IEEE Transactions on*, 14(6):1506–1518, 2003. (Zitiert auf Seite 9)

[Dax] Deutsche Börse Group Factsheet Dax. URL http://www.dax-indices.com/DE/MediaLibrary/Document/Factsheet_DAX_de.pdf. (Zitiert auf Seite 57)

[Dow] S&P Dow Jones Indices Factsheet and Methodology. URL http://www.djindexes.com/literature/. (Zitiert auf Seite 60)

[FPsS96] U. Fayyad, G. Piatetsky-shapiro, P. Smyth. From Data Mining to Knowledge Discovery in Databases. *AI Magazine*, 17:37–54, 1996. (Zitiert auf Seite 14)

[Fra11] F. Franzelin. *Classification with Estimated Densities on Sparse Grids*. Diplomarbeit, Technische Universität München, 2011. URL http://www5.in.tum.de/pub/franzelin_ma11.pdf. (Zitiert auf den Seiten 25 und 70)

[Gar04] J. Garcke. *Maschinelles Lernen durch Funktionsrekonstruktion mit verallgemeinerten dünnen Gittern.* Dissertation, University of Bonn, 2004. (Zitiert auf den Seiten 9, 13 und 25)

[Gar11] J. Garcke. Sparse Grid Tutorial. 2011. URL http://page.math.tu-berlin.de/~garcke/paper/sparseGridTutorial.pdf. (Zitiert auf den Seiten 17 und 18)

[GG98] T. Gerstner, M. Griebel. Numerical integration using sparse grids. *Numerical Algorithms*, 18(3-4):209–232, 1998. (Zitiert auf Seite 13)

[GGG10] J. Garcke, T. Gerstner, M. Griebel. Intraday Foreign Exchange Rate Forecasting using Sparse Grids, 2010. (Zitiert auf den Seiten 13, 34 und 66)

[Goo] Google Finance. URL http://www.google.com/finance. (Zitiert auf den Seiten 57 und 60)

[HDO+98] M. Hearst, S. Dumais, E. Osman, J. Platt, B. Scholkopf. Support vector machines. *Intelligent Systems and their Applications, IEEE*, 13(4):18–28, 1998. (Zitiert auf Seite 9)

[HHR00] M. Hegland, G. Hooker, S. Roberts. Finite Element Thin Plate Splines in Density Estimation, 2000. (Zitiert auf Seite 25)

[HP13] A. Heinecke, D. Pflüger. Emerging Architectures Enable to Boost Massively Parallel Data Mining using Adaptive Sparse Grids. *International Journal of Parallel Programming*, 41(3):357–399, 2013. (Zitiert auf Seite 13)

[HZJP99] M. Y. Hu, G. P. Zhang, C. X. Jiang, B. E. Patuwo. A Cross-Validation Analysis of Neural Network Out-of-Sample Performance in Exchange Rate Forecasting. *Decision Sciences*, 30(1):197–216, 1999. (Zitiert auf Seite 35)

[Lan61] C. Lanczos. *Linear differential operators*. Van Nostrand, London, 1961. (Zitiert auf Seite 80)

[Pfl10] D. Pflüger. *Spatially Adaptive Sparse Grids for High-Dimensional Problems*. Dissertation, Institut für Informatik, Technische Universität München, München, 2010. URL http://www5.in.tum.de/pub/pflueger10spatially.pdf. (Zitiert auf den Seiten 9, 13, 19, 20, 21, 23, 28, 67 und 70)

[PFPB13] B. Peherstorfer, F. Franzelin, D. Pflüger, H.-J. Bungartz. Classification with Probability Density Estimation on Sparse Grids. In *Sparse Grids and Applications*. 2013. Submitted. (Zitiert auf den Seiten 13 und 25)

[She94] J. R. Shewchuk. An Introduction to the Conjugate Gradient Method Without the Agonizing Pain. 1994. (Zitiert auf Seite 24)

[SS10] R. Shumway, D. Stoffer. *Time Series Analysis and Its Applications: With R Examples*. Springer texts in statistics. Springer, 2010. (Zitiert auf den Seiten 9 und 32)

[Tru] TrueFX. URL http://www.truefx.com/. (Zitiert auf Seite 61)

[Zen91] C. Zenger. Sparse Grids. *Notes on Numerical Fluid Mechanics*, 31:241–251, 1991. (Zitiert auf Seite 13)

[Zha03] G. Zhang. Time series forecasting using a hybrid {ARIMA} and neural network model. *Neurocomputing*, 50(0):159 – 175, 2003. (Zitiert auf Seite 9)

Alle URLs wurden zuletzt am 9. 12. 2013 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift