

Abbildung von Steuerungslogiken durch maschinelles Lernen für die Simulation von Produktionssystemen

Emulation of Control Strategies through Machine Learning in Manufacturing Simulations

Sören Bergmann, Niclas Feldkamp, Ulrich Hinze, Steffen Straßburger, TU Ilmenau,
Ilmenau (Germany), soeren.bergmann@tu-ilmenau.de, niclas.feldkamp@tu-
ilmenau.de, ulrich.hinze@tu-ilmenau.de, steffen.strassburger@tu-ilmenau.de.

Abstract: In the context of discrete-event simulation of production and logistics systems, modelling an exact representation of the real system is needed for obtaining sound and reliable results. The automatic generation of simulation models is an approach for saving time and effort for creating models and, therefore, it is a recurring issue in modelling methodology research. In automatic model generation, the approximation of dynamic behaviour is a challenging problem. This is for example the case when the dispatching and scheduling of production jobs needs to be adequately emulated, but the underlying rules are unknown. In previous work, we presented an approach for approximating dynamic behaviour through artificial neural networks. In this paper, we propose an improved approach and investigate its suitability again with artificial neuronal networks as well as with other data mining and supervised machine learning methods.

1 Einführung

Die automatische Generierung von Simulationsmodellen wird immer wieder als ein probates Mittel genannt, um zum einen Simulationsstudien zu beschleunigen, sowie zum anderen neue Nutzergruppen für die Simulation zu gewinnen (Selke 2005, S.23; Bergmann und Straßburger 2010, S.545f; Bergmann 2013, S.25ff). Eine spezielle Herausforderung bzw. Lücke innerhalb der automatischen Modellgenerierung (AMG) stellt die Abbildung von Steuerungsregeln dar (Selke 2005, S.30; Bracht et al. 2008, S.30; Bergmann und Straßburger 2010, S.545f; Bergmann 2013, S.71ff).

Steuerungsregeln können hierbei in vier verschiedene Klassen eingeordnet werden, konkret in Strategien zur Auftragsfreigabe, zur Losgrößenbestimmung, zur Ressourcenbelegung und Reihenfolgegestaltung (Selke 2005, S.45), wobei gerade Entscheidungen der letztgenannten Klasse in vielen Systemen von enormer Bedeutung sind und stellvertretend im Folgenden betrachtet werden. In der Praxis werden in allen

Klassen häufig Entscheidungsregeln, z. B. First In First Out (FIFO) oder Shortest Processing Time (SPT), genutzt (Pinedo 2012).

Zur Behandlung der skizzierten Lücke im Prozess der AMG sind drei Klassen von Ansätzen mit je eigenen Vor- und Nachteilen bzw. Restriktionen denkbar.

Die Trivallösung und erste Klasse besteht in dem Modellieren der Regeln durch einen Experten, entweder direkt in der Simulationsumgebung oder in einer systemneutralen Sprache, z. B. SysML (Schönherr und Rose 2009).

Die zweite Klasse versucht, bestehende Regeln bzw. eine Kombination von Basisregeln zu erkennen (Wiedemann 1999; Selke 2005; Reinhart und Gyger 2008).

Der in die dritte Klasse fallende im Folgenden aufgegriffene Ansatz erkennt keine vorgefertigten Steuerungsregeln, sondern approximiert das Verhalten der (Realwelt-)Regeln. Basis können hier beispielsweise Daten der Betriebsdatenerfassung (BDE-Daten) sein, welche direkt zum Erzeugen bzw. Trainieren der Regel genutzt werden. Dieser Ansatz wurde bereits vereinzelt in Publikationen aufgegriffen. So wurden bspw. künstliche neuronale Netze (KNN) zur Approximation von Reihenfolgeregeln prototypisch eingesetzt (Bergmann 2013; Bergmann et al. 2014). Das genannte approximierende Verfahren zeigte das Potential solcher Ansätze auf, unterlag aber verschiedensten Restriktionen bezüglich Datenqualität oder Skalierbarkeit; so wurde beispielsweise eine fixe Puffergröße unterstellt.

Dieser Beitrag beschreibt eine Erweiterung bzw. Generalisierung des bestehenden Ansatzes. Es wird ein an den KDD-Prozess (Knowledge Discovery in Databases) (Fayyad 1996) angelehntes Framework zur Approximation von Steuerungsregeln aus BDE-Daten mittels Methoden des maschinellen Lernens vorgestellt. Hierbei werden mittels Data-Mining-Methoden auf Basis von BDE-Daten Entscheidungsmodelle erzeugt, welche im Folgenden aus der Simulation heraus genutzt werden können.

Hierzu wird in Kapitel 2 das Framework zur Datenaufbereitung und der Abbildung von Steuerungsregeln durch maschinelles Lernen vorgestellt. Ebenfalls werden die aktuell genutzten Methoden des maschinellen Lernens / Data-Mining-Methoden und eine Auswahl ihrer wichtigsten Parameter kurz vorgestellt. Im 3. Kapitel wird zunächst ein kleines Testszenario definiert, bevor die verschiedenen vorgestellten Data-Mining-Methoden unter unterschiedlichen Randbedingungen und mit verschiedenen Parametrisierungen der Methoden sowie unter Nutzung verschiedener Datenaufbereitungsverfahren des Frameworks hinsichtlich ihrer Abbildungsgenauigkeit validiert werden. Die Ergebnisse werden anschließend miteinander verglichen und diskutiert. Abschließend wird ein kurzes Fazit gezogen und ein Ausblick auf mögliche Erweiterungen gegeben.

2 Konzept eines Frameworks zur Abbildung von Steuerungsregeln

Ziel des Frameworks ist es, ein direkt nutzbares Entscheidungsmodell auf Basis eines bestehenden Datenbestands zu erzeugen. Das finale Entscheidungsmodell kann nach Abschluss des Erzeugungsprozesses direkt genutzt werden, z.B. im Kontext der AMG direkt aus der Simulation heraus.

Der Basisdatenbestand umfasst hierbei zunächst je einen Datensatz pro Entscheidungsvorgang, welcher den aktuellen Systemzustand und die getroffene Entscheidung beinhaltet. Aus Gründen der Skalierbarkeit und Robustheit werden alle Entscheidungen auf binäre Entscheidungen zurückgeführt und entsprechende Datensätze erzeugt. Muss zum Beispiel eine Auswahl zwischen 3 Aufträgen (A, B, C) getroffen werden, wird zunächst ein Datensatz für den Vergleich zwischen A und B und anschließend zwischen dem „Gewinner“ und C durchgeführt.

Um gute Ergebnisse zu erreichen, sind zwei wesentliche Teilbereiche zu untersuchen (Abb. 1). Zum einen müssen geeignete Verfahren des maschinellen Lernens ausgewählt und deren Parameter geeignet gesetzt werden, zum anderen müssen die Daten für gute Ergebnisse meist vorab aufgearbeitet werden (Datentransformation).

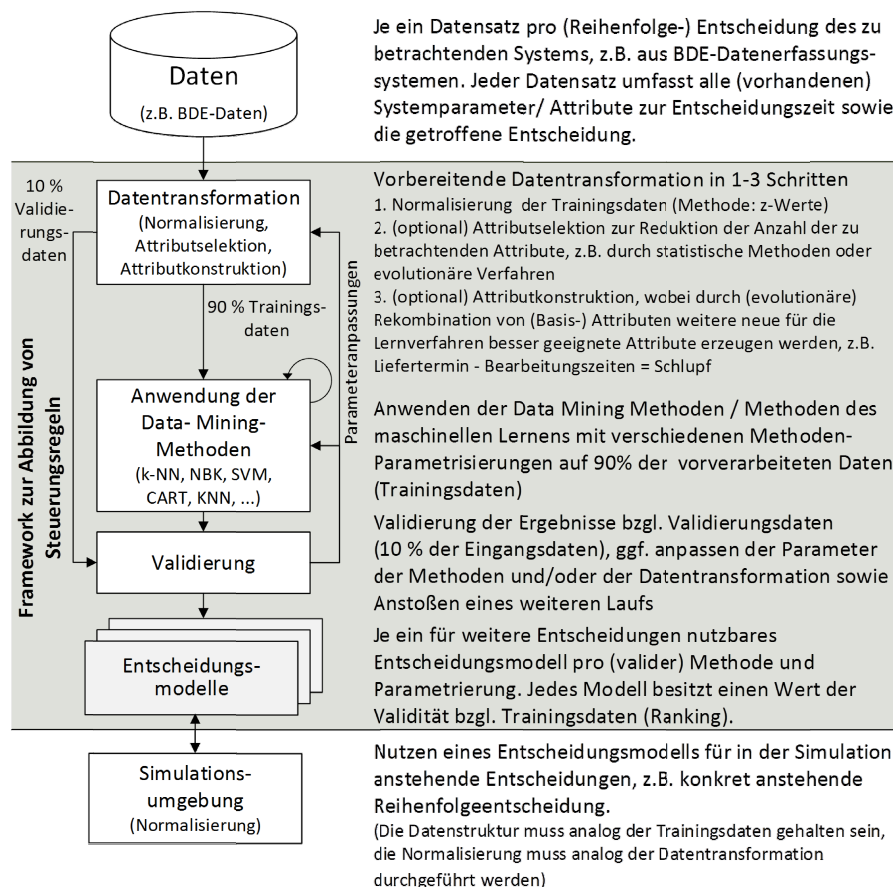


Abbildung 1: Schematischer Ablauf bei der Abbildung von Steuerungsregeln

Das Framework bietet für die Datentransformation verschiedene Methoden zur Datenaufbereitung, z. B. die konstruktive Induktion von Attributen, die Attributselektion und die Normalisierung von Attributwerten. Die Normalisierung von Attributwerten wird obligatorisch durchgeführt, das aktuell genutzte Verfahren

basiert auf dem statistischen Standardverfahren der z-Transformation (Bortz und Schuster 2001).

Die Attributselektion und -konstruktion sind optionale Schritte. Im aktuellen Entwicklungsstand werden u.a. evolutionär mögliche Varianten zur Auswahl von Attributen und zur Rekombination von Attributen durchlaufen. Jede Variante wird anschließend angewendet und validiert.

Allgemein sind beliebige Methoden des maschinellen Lernens im Framework einsetzbar und aktuell sind die fünf bekanntesten Vertreter implementiert (vgl. Kapitel 3). Jede Methode hat wiederum eine Vielzahl von Parametern, die das Ergebnis beeinflussen können. Viele dieser Parameter können im Framework gesetzt und entsprechend angepasst werden.

3 Methoden des maschinellen Lernens

Unter Methoden des maschinellen Lernens bzw. den im Folgenden synonym verwendeten Data-Mining-Methoden werden allgemein Methoden verstanden, welche aus heterogenen und großen Datenmengen Muster erkennen, um diese für zukünftige Entscheidungen nutzbar zu machen (Fayyad 1996). Für die Nachbildung der Steuerungsregeln wurden fünf verschiedene Methoden genutzt, und zwar k-nächste-Nachbarn (k-NN), Naiver-Bayes-Klassifikator (NBK), Support-Vektor-Maschine (SVM), künstliche neuronale Netze (KNN) und der CART-Algorithmus zur Erzeugung von Entscheidungsbäumen, jeweils mit verschiedenen Parametervariationen. Die prinzipielle Funktionsweise der jeweiligen Methode soll im Folgenden kurz besprochen werden. Um einen Vergleich zur vorherigen Arbeit und dem Ansatz mit neuronalen Netzen zu ziehen, wird auch die Funktionsweise der neuronalen Netze (Multilayer Perceptrons mit Backpropagation) kurz erläutert.

3.1 k-nächste-Nachbarn-Algorithmus

Dieser Algorithmus ordnet unklassifizierte Datensätze jener Klasse zu, die der Mehrheit seiner k-nächsten Nachbarn entspricht. Die Nachbarschaft von Datensätzen wird durch deren Ähnlichkeit bestimmt. Für k=1 sucht der Algorithmus beispielsweise den ähnlichsten Datensatz in der Trainingsdatenmenge und weist dem zu klassifizierenden Datensatz dieselbe Klasse entsprechend seines Nachbarn zu. Die Ähnlichkeit von zwei Datensätzen wird über ein Distanzmaß berechnet.

$$d(p, q) = \sqrt{\sum_{i=1}^n (q_i - p_i)^2} \quad (1)$$

Gleichung 1 zeigt, wie sich die Distanz d zwischen zwei Datensätzen p und q mit Hilfe des euklidischen Distanzmaßes berechnen würde. Die Wahl des k-Parameters hat großen Einfluss auf die Qualität der Ergebnisse und hängt von der Beschaffenheit der zu untersuchenden Datenmenge ab. Größere k-Werte machen den Algorithmus üblicherweise robuster gegen Rauschen, während kleinere k-Werte eine schärfere Abgrenzung zwischen den Klassen herstellen. Tatsächlich generiert der k-NN-Algorithmus kein konkretes Entscheidungsmodell oder -funktion, sondern vergleicht lediglich paarweise den zu klassifizierenden Datensatz mit den Trainingsdaten und berechnet jeweils die Distanz (Altman 1992).

3.2 Naiver-Bayes-Klassifikator

Dieser Algorithmus ist ein wahrscheinlichkeitsbasierter Klassifikator. Das Entscheidungsmodell weist demnach einem unklassifizierten Datensatz diejenige Klasse mit der höchsten Wahrscheinlichkeit zu.

Während der Trainingsphase, d.h. der Generierung des Entscheidungsmodells, werden Wahrscheinlichkeitsverteilungen für die Parametervektoren jeder Klasse gebildet. Soll nun ein neuer Datensatz klassifiziert werden, wird aus den Wahrscheinlichkeiten jedes Parameters eine Gesamtwahrscheinlichkeit für jede Klasse gebildet. Abschließend wird nun also jene Klasse zugewiesen, die die höchste Wahrscheinlichkeit aufweist, dass der zu klassifizierende Datensatz dieser Klasse angehört. Ein großer Nachteil hierbei ist, dass die Verteilungsfunktion für kontinuierliche Parameter im Vorhinein bekannt sein muss. Eine falsche Verteilungsfunktion führt eventuell zu falschen Wahrscheinlichkeiten und damit zu falschen Klassifizierungen. Durch die Anwendung von komplexeren Verteilungsfunktionen wie der multivariaten Normalverteilung kann dieses Problem abgeschwächt werden (Rennie et al. 2003).

3.3 Support-Vektor-Maschine

Im Gegensatz zu den anderen Methoden teilt dieser Algorithmus die Trainingsdaten immer in zwei Klassen auf. Da wir in unserem Ansatz hinsichtlich der Reihenfolgeplanung immer zwei Aufträge (und damit zwei Klassen) paarweise miteinander vergleichen, ist dies aber keine hinderliche Einschränkung. Ordnet man alle Trainingsdatensätze im Raum an, spannt der Algorithmus eine Ebene bzw. bei mehr als 3 Parametern eine Hyperebene auf, die als Grenze zwischen den zwei Klassen dient. Neue Datensätze können dann relativ einfach anhand ihrer Position im Raum klassifiziert werden. Wenn die Trainingsdaten nicht linear trennbar sind, lässt sich der sogenannte Kernel-Trick anwenden, welcher die Daten in einen Raum mit höherer Dimension übersetzt, in welcher diese dann separierbar sind. Die dafür benötigte Kernelfunktion ist ein zusätzlicher Stellparameter der SVM und kann deren Leistung beeinflussen (Han et al. 2012; Boser et al. 1992).

3.4 Entscheidungsbäume

Für die Generierung von Entscheidungsmodellen sind Entscheidungsbäume sicherlich die bekannteste Methode. Ein Entscheidungsbaum besteht aus Knoten und Blättern. Von oben nach unten werden dabei entlang der Baumstruktur an den Knoten jeweils Entscheidungen über bestimmte Parameterwerte getroffen, sodass am Ende eine an den Blättern hinterlegte Klasse zugewiesen werden kann. Für den Aufbau von Entscheidungsbäumen existiert eine Vielzahl an verschiedenen Algorithmen. In dieser Arbeit wurde für die Induktion von Bäumen der Algorithmus CART (Classification and Regression Trees) verwendet. CART verwendet binäre Bäume, was bedeutet, dass jeder Knoten im Baum jeweils exakt zwei Kinder hat. Welcher Parameter an welchem Knoten im Baum abgefragt wird, hängt von der Reinheit der zu erwartenden Trennung ab, also dem Gewinn an Informationsgehalt (Entropie). Dies wird durch das sog. Splitkriterium, wie beispielsweise der Gini-Index (Breiman 1984), berechnet.

3.5 Künstliche neuronale Netze

Künstliche neuronale Netze (KNN) sind vielfältig einsetzbar und in verschiedenen Varianten bzw. Topologien zu finden. Allgemein kann gesagt werden, dass KNN als hoch flexible Funktionsapproximierer genutzt werden können (Zell 1997). Der Grundansatz ist inspiriert vom menschlichen Gehirn. Die Basiskomponenten sind einfache Entitäten, die Neuronen, welche untereinander durch gewichtete Verbindungen verbunden sind. Im einfachsten Fall besitzt ein Neuron ein Aktivierungslevel, welches sich aus der Aktivierung der verbundenen Neuronen ergibt, d.h. die Summe der gewichteten Eingangssignale darstellt. Durch eine im Neuron definierte Aktivierungsfunktion wird dessen Ausgangssignal auf Basis des Aktivierungslevels berechnet. In den hier genutzten Multilayer Perceptrons sind die Neuronen in Schichten angeordnet, wobei nur Verbindungen von Neuronen einer Schicht zur nächst höheren erlaubt sind. Alle Eingaben an das Netz werden über die Aktivierung von Neuronen der ersten Schicht realisiert. Die Ergebnisse sind in der Aktivierung der Neuronen der letzten Schicht abzulesen. Das Trainieren der Netze kann mit verschiedenen Verfahren erfolgen, wobei die Gewichte von Verbindungen und die Parameter der Aktivierungsfunktionen angepasst werden. Das verbreitetste Verfahren für überwachtes Lernen ist das Back-Propagation-Verfahren (Zell 1997).

4 Testszenario und Ergebnisse der Versuche

Der Validierung des Ansatzes bzw. des Frameworks erfolgt in drei Phasen. In Phase 1 werden die (BDE-) Daten erzeugt. Da die Untersuchungen zunächst darauf abzielen, Methoden, Methodenparametrierungen und Einstellungen der Datentransformation zu testen und zu bewerten, wurde bewusst ein überschaubares Simulationsmodell zu Datengenerierung herangezogen. Die Simulation ermöglicht hierbei, schnell verschiedene Daten mit unterschiedlichen Eigenschaften zu erzeugen; so können u. a. schnell Daten für verschiedene Reihenfolgeregeln, welche dann approximiert werden sollen, generiert werden. Als Simulationssystem wurde Tecnomatix Plant Simulation gewählt. Im Modell (Abb. 2) wurden fünf Auftrags-typen mit unterschiedlichen Attributen (z.B. Bearbeitungszeiten, Lieferzeiten, Rüstzuständen) definiert.

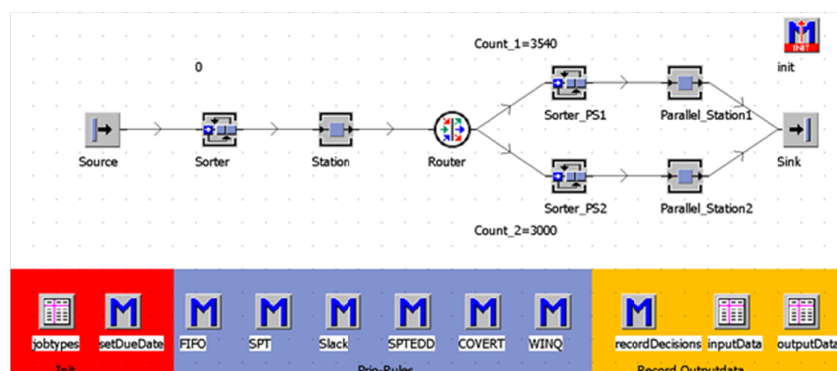


Abbildung 2: Screenshot des Plant-Simulation-Modells zur Datengenerierung

Die Reihenfolgeentscheidung erfolgt in einem Sortierpuffer vor der ersten Bearbeitungsstation. Im Sortierpuffer können verschiedene Reihenfolgeregeln (z.B. FIFO, SPT, EDD) hinterlegt werden, wobei je Datenerzeugungsphase nur eine sich nicht ändernde Regel zum Einsatz kommt. Des Weiteren wurde eine BDE-Daten-Erfassung emuliert.

Die 2. Phase umfasst den eigentlichen Prozess der Erzeugung der Entscheidungsmodelle wie in Kapitel 2 (vgl. Abb. 1) beschrieben. Hierbei kommen für jedes Trainingsdatenset aus Phase 1 alle vorgestellten Data-Mining-Methoden zum Einsatz, wobei je Methode verschiedene Parametrierungen getestet werden. Zusätzlich werden Versuche mit unterschiedlichen Datentransformationen durchgeführt. Der Großteil der prototypischen Implementierung des Frameworks erfolgte in MathWork MATLAB unter Nutzung der enthaltenen Toolboxes.

In der letzten Phase (Phase 3) werden die in Phase 2 erzeugten Modelle für die Validierungsdaten getestet, hierbei wird der Anteil der korrekten (binären) Entscheidungen an der Menge aller (binären) Entscheidungen gespeichert.

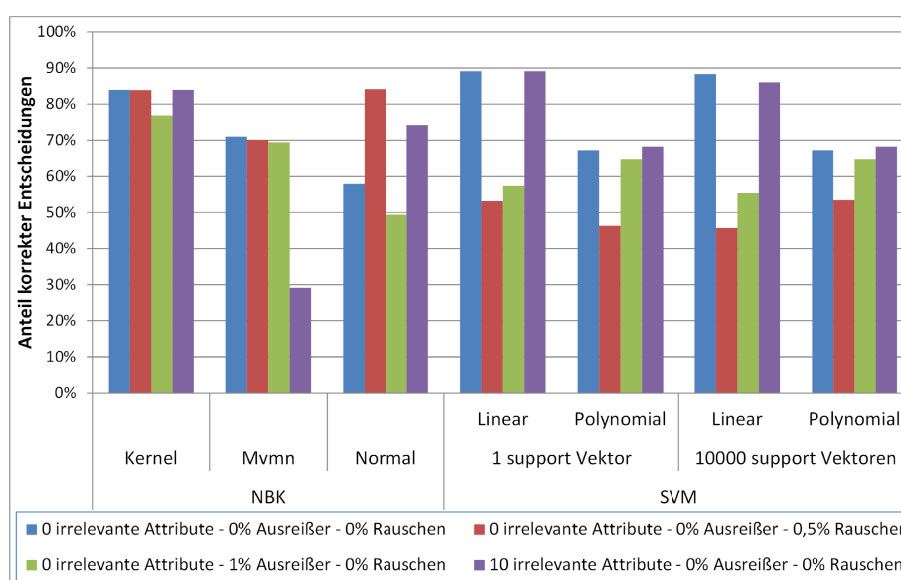


Abbildung 3: Vergleich der Data-Mining-Methoden NBK und SVM für verschiedene Parametrierungen

Erste Versuche und Auswertungen wurden für jede Data-Mining-Methode durchgeführt, um zum einen deren prinzipielle Eignung und zum anderen eine, für das Szenario, gute Parametrierung zu ermitteln. Hierbei wurde zugleich die Robustheit bzgl. der Datenqualität (Stufe A – blau; B – violett; C – grün; D – rot) untersucht. Beispielhaft sind die gemittelten Resultate für verschiedene Systemkonfigurationen (wie verschiedene Reihenfolgeregeln im Generatormodell) für die Methoden NBK und SVM in Abbildung 3 dargestellt.

Gut zu erkennen ist der hohe Einfluss der Parametrierung auf den Anteil der korrekten Entscheidungen. Des Weiteren zeigten die Versuche, dass einzelne Methoden

erhebliche Unterschiede bzgl. ihrer Robustheit gegenüber niedrigerer Datenqualität aufweisen. Auffällig ist, dass einzelne Methoden bei bestimmten Parametrierungen besonders bei bestimmten Datenverfälschungen deutlich unterdurchschnittlich abschneiden, z. B. reagiert der NBK mit multivariater Multinomialverteilung (Mvnm) besonders schlecht auf zusätzliche Attribute. Wird anstatt dessen eine Kernelfunktion genutzt, ist der SVM robust gegenüber sinkender Datenqualität. Der SVM hingegen schneidet bei Rauschen und Ausreißern prinzipiell unterdurchschnittlich ab. Die skizzierten Fakten sind auch bei allen weiteren Methoden entsprechend beobachtbar. Nebenergebnis dieser ersten Untersuchungen war die Erkenntnis, dass die Methoden sich deutlich in ihrer Performance des Lernalgorithmus unterscheiden. So wurden Zeiten gemessen, die je nach Datenqualität, -menge und Parametrierung von wenigen Sekunden für beispielsweise den k-NN bis hin zu mehreren Stunden für SVM und KNN lagen. Bei der späteren Nutzung der Entscheidungsmodelle hingegen waren kaum Performanceunterschiede erkennbar.

Neben der Ermittlung der besten Parametrierung für einzelne Methoden ist natürlich die Auswahl der besten Methode sowie Datentransformation Ziel der Versuche. Alle weiteren Versuche und Vergleiche nutzen die für jede Data-Mining-Methode ermittelte beste Parametrierung.

Abbildung 4 zeigt den Vergleich der fünf Data-Mining-Methoden. Gut ersichtlich ist, dass Entscheidungsbäume unter Anwendung des CART-Algorithmus und die Methode k-NN prinzipiell zu sehr guten Ergebnissen führen. Für den Anwendungsfall der AMG sind Werte von >95% korrekte Entscheidungen hervorragend.

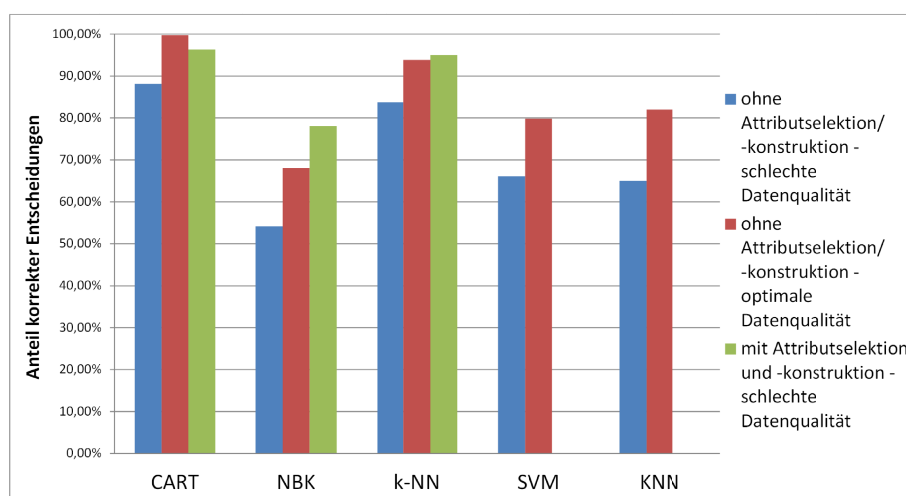


Abbildung 4: Vergleich der verschiedenen Data-Mining-Methoden (je beste Parametrierung) für verschiedene Datenqualitäten und bei Einsatz der Attributselektion und -konstruktion

Des Weiteren ist ersichtlich, dass Datentransformation, mit Attributselektion und -konstruktion, schlechte Datenqualität kompensieren kann. Für die Methoden SVM und KNN sind keine Daten bzgl. der Korrektheit bei aktivierter Attributselektion und -konstruktion sinnvoll zu erheben, da aufgrund der Laufzeit der Algorithmen

der im Framework vorgesehene evolutionärer Ansatz, welcher eine große Zahl von Lernvorgängen benötigt, versagt. Alle anderen Methoden profitieren aber deutlich von der Datentransformation.

Generell hat sich gezeigt, dass die Methode der Entscheidungsbäume unter Nutzung des CART-Algorithmus zu den besten Ergebnissen (bis zu 99% korrekte Entscheidungen) führt. Erstaunlich ist aber, dass der sehr simple k-NN, besonders bei Nutzung entsprechender Datentransformation, zu ähnlich guten Ergebnissen führt. Der NBK hingegen scheint eher ungeeignet für die Abbildung von Steuerungsregeln; in keinem Versuch, auch bei bester Datenqualität, konnten Ergebnisse mit 80% oder mehr korrekten Entscheidungen beobachtet werden.

5 Zusammenfassung und Ausblick

Die Ergebnisse der Untersuchung zeigen, dass ausgewählte Methoden des maschinellen Lernens selbst für komplexe Steuerungsregeln prinzipiell gute Approximationen bieten. Somit wurde veranschaulicht, dass die Anwendung von Data-Mining-Methoden eine effektive Lösung für das Abbilden von dynamischem Verhalten im Kontext der automatischen Modellgenerierung darstellt. In den bisherigen Untersuchungen wurden keinerlei Einschränkungen bzgl. möglicher abbildbarer Regeln sichtbar, weitere Untersuchungen in komplexeren Realwelt-Szenarien stehen aber noch aus. Auch lassen sich keine Aussagen darüber machen, wie gut und wie schnell sich die Algorithmen an ein sich änderndes Systemverhalten während der Trainingsphase anpassen können. Dies wäre in weiteren Forschungsarbeiten zu überprüfen.

Unser Ansatz, Reihenfolgeentscheidungen auf binäre Vergleiche zwischen zwei Aufträgen zurückzuführen, bringt dabei eine große Flexibilität mit sich. Auf der anderen Seite steigt damit jedoch der Rechenaufwand mit der Anzahl der zu vergleichenden Aufträge im System. Die Qualität der Entscheidungsmodelle schwankt stark in Abhängigkeit von der gewählten Methode und deren Konfiguration und Parametrisierung. Hier besteht im Rahmen weiterer Forschungsarbeit noch weiteres Optimierungspotenzial. Durch vorbereitende Datentransformationsschritte konnte zudem die Entscheidungsqualität weiter erhöht werden. Selbst bei schlechter Datenqualität der Trainingsdaten (d. h. Rauschen und absichtliche Fehlklassifizierungen) konnte hierbei mit Hilfe des CART-Algorithmus eine Korrektheit von bis zu 99% erreicht werden.

Allerdings richtet sich das von uns verwendete Kriterium zur Bewertung der Entscheidungsqualität rein nach der Anzahl der korrekt getroffenen Entscheidungen. Eine interessante Erweiterungsmöglichkeit zur Bewertung der Verfahren wäre hierbei, die generierten Entscheidungsmodelle wieder zurück in das Ausgangsmodell zu implementieren und somit zusätzlich zu der Korrektheit der Entscheidung zu messen, ob und wie stark sich eine falsch getroffene Entscheidung überhaupt auf das System auswirkt. Eine faktisch falsch getroffene Entscheidung muss sich nicht zwangsläufig in veränderten Leistungskennzahlen des Systems niederschlagen. Des Weiteren ist zu überprüfen, ob die Leistungsfähigkeit der untersuchten Algorithmen in komplexeren Realszenarios bestätigt werden kann.

Literatur

- Altman, N. S.: An introduction to Kernel and Nearest-Neighbor Nonparametric regression. *The American Statistician* (1992) 46, S. 175–185.
- Bergmann, S.: Automatische Generierung adaptiver Modelle zur Simulation von Produktionssystemen. Ilmenau: Univ.-Verl. Ilmenau 2013.
- Bergmann, S.; Strassburger, S.: Challenges for the automatic generation of simulation models for production systems. In: *Proceedings of the International Summer Simulation Multiconference*. Ottawa (Canada), 12.-14. Juli 2010, S. 545–549.
- Bergmann, S.; Stelzer, S.; Strassburger, S.: On the use of artificial neural networks in simulation-based manufacturing control. *Journal of Simulation* 8 (2014) 1, S. 76–90.
- Bortz, J.; Schuster, S.: Statistik für Human- und Sozialwissenschaftler, 7. Auflage. Berlin: Springer 2001.
- Boser, B. E.; Guyon, I. M.; Vapnik, V. N.: A training algorithm for optimal margin classifiers. In: Haussler, D. (Hrsg.): *Proceedings of the fifth annual workshop on Computational learning theory*, Pittsburgh (USA), 27 -29 July 1992, S. 144–152.
- Bracht, U.; Rooks, T.; Adrian, R.: Virtuelle Logistikplanung für die Montage im Rahmen der Digitalen Fabrik. In: Rabe, M. (Hrsg.): *Advances in Simulation for Production and Logistics Applications*. Stuttgart: Fraunhofer IRB 2008.
- Breiman, L.: Classification and regression trees. New York: Chapman & Hall 1984.
- Fayyad, U. M.: Advances in knowledge discovery and data mining. Menlo Park, Calif.: American Association for Artificial Intelligence; AAAI 1996.
- Han, J.; Kamber, M.; Pei J.: Data mining concepts and techniques, 3. Auflage. Amsterdam: Elsevier/Morgan Kaufmann 2012.
- Pinedo, M.: Scheduling. Theory, algorithms, and systems, 4. Auflage. New York: Springer 2012.
- Reinhart, G.; Gyger, T.: Identification of implicit strategies in production control. In: *Proceedings of the Industrial Engineering and Engineering Management 2008 (IEEM 2008)*. Singapore, 8.-11. Dezember 2008, S. 302–306.
- Rennie, J. D.; Shih, L.; Teevan, J.; Karger, D. R.: Tackling the poor assumptions of naive bayes text classifiers. In: Fawcett, T.; Mishra N. (Hrsg.): *Proceedings of the Twentieth International Conference on Machine Learning*, Washington, DC (USA), 21-24 August 2003, S. 616–623.
- Schönherr, O.; Rose, O.: First steps towards a general SysML model for discrete processes in production systems. In: Rossetti, M. D.; Hill, R. R.; Johansson, B.; Dunkin A.; Ingalls, R. G. (Hrsg.): *Proceedings of the 2009 Winter Simulation Conference (WSC2009)*. Austin (USA), 13.-16. Dezember 2009, S. 1711–1718.
- Selke, C.: Entwicklung von Methoden zur automatischen Simulationsmodellgenerierung. München: Utz 2005.
- Wiedemann, T.: Database oriented modeling with simulation microfunctions. In: Farrington, P. A.; Nembhard, H. B.; Sturrock, D. T.; Evans, G. W. (Hrsg.): *Proceedings of the 1999 Winter Simulation Conference (WSC1999)*. Phoenix (USA), 5.-8. Dezember 1999, S. 586–590.
- Zell, A.: Simulation Neuronaler Netze. München: Oldenbourg 1997.