



(10) **DE 10 2021 213 112 A1** 2023.05.25

(12) **Offenlegungsschrift**

(21) Aktenzeichen: **10 2021 213 112.3**
(22) Anmeldetag: **22.11.2021**
(43) Offenlegungstag: **25.05.2023**

(51) Int Cl.: **G05B 13/02 (2006.01)**
G05D 1/00 (2006.01)

(71) Anmelder:
**Robert Bosch Gesellschaft mit beschränkter
Haftung, 70469 Stuttgart, DE**

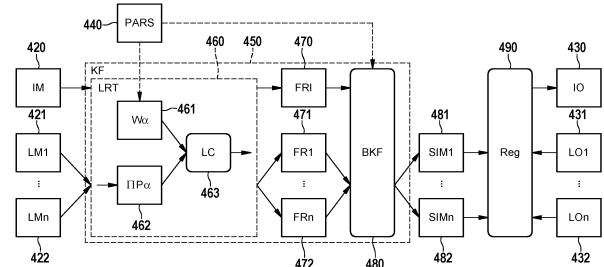
(72) Erfinder:
Kirstein, Max, 31249 Hohenhameln, DE

Rechercheantrag gemäß § 43 PatG ist gestellt.

Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen.

(54) Bezeichnung: **STEUERUNGSSYSTEM DURCH VERWENDEN EINER GAUSSSCHEN PROZESSREGRESSION**

(57) Zusammenfassung: Die Erfindung betrifft ein computerimplementiertes Steuerungsverfahren (700). Messungen von physikalischen Eingabegrößen werden erhalten, und die Gaußsche Prozessregression wird verwendet, um einen Wert einer Ausgabegröße basierend auf den erhaltenen Messungen zu folgern, der dann verwendet wird, um das computergesteuerte System zu steuern. Die GP-Regression bezieht das Anwenden einer Kernelfunktion ein, um jeweilige Ähnlichkeiten der erhaltenen Messungen mit jeweiligen gekennzeichneten Instanzen zu berechnen. Die Kernelfunktion wird angewendet, indem ein trainiertes Tensor-Train-Netzwerk auf die physikalischen Eingabegrößen angewendet wird, und eine Basis-Kernel-Funktion auf die resultierenden Merkmalsdarstellungen angewendet wird. Der Wert der Ausgabegröße wird von den Ähnlichkeiten und jeweiligen Werten der Ausgabegröße der gekennzeichneten Instanzen gefolgert.



Beschreibung

Feld der Erfindung

[0001] Die Erfindung betrifft ein computerimplementiertes Steuerungsverfahren zum Steuern eines computergesteuerten Systems, das mit einer Umgebung interagiert, und ein entsprechendes System. Die Erfindung betrifft ferner ein computerimplementiertes Trainingsverfahren zum Trainieren einer Gaußschen Prozessregression zur Verwendung beim Steuern eines computergesteuerten Systems, und ein entsprechendes System. Die Erfindung betrifft ferner ein computerlesbares Medium.

Hintergrund der Erfindung

[0002] Steuerungssysteme empfangen wachsende Mengen von Sensordaten von einem weiten Bereich unterschiedlicher Sensoren, und es wird von ihnen erwartet, immer weitreichendere Steuerungsentscheidungen zu treffen. Zum Beispiel können (halb-)autonome Fahrzeuge Eingaben von Kameras, Radarsensoren und Lidar-Sensoren verwenden, um Entscheidungen zu treffen, wann sie bremsen, die Fahrspuren wechseln, usw. Diese Eingaben müssen kombiniert werden, um eine umfassende Steuerungsentscheidung zu treffen, wofür eine hohe Sicherheit erforderlich ist.

[0003] Maschinelles Lernen ist vielversprechend für die generelle Gewinnung von Wissen von solchen Sensordaten. Jedoch stellt die Anwendbarkeit von Techniken für maschinelles Lernen in Hoch-Risiko-Bereichen, wie zum Beispiel automatisierten Steuerungssystemen, aber auch medizinischen Anwendungen, eine Anzahl spezieller Anforderungen, für deren Handhabung nicht alle Techniken für maschinelles Lernen gut ausgerüstet sind. Insbesondere in anspruchsvollen Bereichen ist es wichtig, Techniken für maschinelles Lernen zu verwenden, die gut mit Unsicherheit umgehen können, die zum Beispiel ihre Vorhersageunsicherheit zuverlässig quantifizieren können, sodass diese Quantifizierung bei den durch das Steuerungssystem getroffenen Steuerungsentscheidungen berücksichtigt werden kann. Andere Anliegen beinhalten das Ermöglichen der Integration von Vorwissen in den Modellierungsprozess, und das Bereitstellen von Interpretierbarkeit durch strenge mathematische Grundlagen, sogar auf Kosten einer reduzierten Darstellungsleistung.

[0004] Eine bekannte Technik zum Umgang mit Vorhersageunsicherheit ist die Verwendung von Gaußschen Prozessen. Eine wichtige Aufgabe in Steuerungssystemen, für die Techniken für maschinelles Lernen eingesetzt werden können, ist die Regression, insbesondere der Rückschluss auf einen Wert einer physikalischen Größe basierend auf Messungen anderer physikalischer Größen. Zum Beispiel kann in einem (halb-)automatisierten Fahrzeug eine Entfernung, eine Geschwindigkeit oder eine Beschleunigung von Sensormessungen abgeleitet werden. Ein Gaußscher Prozess folgert in diesem Zusammenhang eine Ausgabegröße von physikalischen Eingabegrößen, indem er die physikalischen Eingabegrößen mit gekennzeichneten Instanzen gemäß einer Kernelfunktion vergleicht. Die Ausgabegröße wird von ihren Werten für die gekennzeichneten Instanzen gemäß den berechneten Ähnlichkeiten gefolgert. Die Variabilität zwischen den Ausgaben für ähnliche Eingaben kann verwendet werden, um eine Unsicherheit des gefolgerten Werts zu schätzen.

[0005] Um zu ermöglichen, dass die Gaußsche Prozessregression mit komplexeren Eingaben, wie zum Beispiel hochdimensionalen Daten und Bildern arbeitet, ist es bekannt, Deep-Kernel-Learning zu verwenden. Beim Deep-Kernel-Learning wird ein neuronales Netzwerk verwendet, um Eingaben auf Punkte in einem Zwischenmerkmalsraum abzubilden, der dann als der Eingaberaum für den Gaußschen Prozess verwendet wird. Mit anderen Worten, es wird eine Kernelfunktion verwendet, die das neuronale Netzwerk als einen Merkmalsextrahierer beinhaltet. Dies ist zum Beispiel in S. Ober et al., „The Promises and Pitfalls of Deep Kernel Learning“ (verfügbar unter <https://arxiv.org/abs/2102.12108> und hierin unter Bezugnahme eingegliedert) beschrieben. Diese Referenz weist jedoch auch darauf hin, dass dieser Ansatz unter dem Problem der Überanpassung leidet. Dieses Problem kann nur teilweise durch Regularisierung behoben werden. Insbesondere ist es mit den bestehenden Techniken schwierig, Expertenwissen in die Modellstruktur aufzunehmen.

Kurzdarstellung der Erfindung

[0006] Es wäre wünschenswert, Steuerungssysteme zu haben, die ihre Steuerungsentscheidungen basierend auf verbesserten Rückschlüssen auf physikalische Größen basieren können.

[0007] Gemäß einem ersten Aspekt der Erfindung werden ein computerimplementiertes Steuerungsverfahren und ein entsprechendes Steuerungssystem zum Steuern eines computergesteuerten Systems, das mit einer Umgebung interagiert, beschrieben, wie durch die Ansprüche 1 bzw. 13 definiert. Gemäß einem ande-

ren Aspekt der Erfindung werden ein computergesteuertes Trainingsverfahren und ein entsprechendes Trainingssystem zum Trainieren einer Gaußschen Prozessregression zur Verwendung bei einer solchen Steuerung bereitgestellt, wie in den Ansprüchen 7 bzw. 14 definiert. Gemäß einem Aspekt der Erfindung wird ein computerlesbares Medium beschrieben, wie in Anspruch 15 definiert.

[0008] Verschiedene Ausführungsformen beziehen sich auf die Verwendung von Regression in einem Steuerungssystem. Das Steuerungssystem kann Messungen einer oder mehrerer physikalischer Eingabegrößen des computergesteuerten Systems und/oder seiner Umgebung von entsprechenden Sensoren, wie zum Beispiel Bildsensoren, Lidar-Sensoren und/oder Radarsensoren, erhalten. Die Regression kann verwendet werden, um einen Wert einer Ausgabegröße basierend auf den erhaltenen Messungen zu folgern, optional in Kombination mit einer zugehörigen Unsicherheitsschätzung. Zum Beispiel kann das computergesteuerte System ein autonomes oder halbautonomes Fahrzeug sein, wobei die physikalischen Eingabegrößen zum Beispiel Bild-, Radar- und/oder Lidar-Daten oder von solchen Daten extrahierte physikalische Größen umfassen können, und die Ausgabegröße kann eine gefolgerte Entfernung, Geschwindigkeit oder Beschleunigung sein. Der gefolgerte Wert der Ausgabegröße und/oder dessen Unsicherheit können verwendet werden, um Steuerungsdaten auf verschiedene Arten zu bestimmen, die an sich bekannt sind, wie zum Beispiel durch Auslösen eines Alarms, wenn der gefolgerte Wert über oder unter einem Schwellenwert liegt, oder Anwenden eines weiteren Modells für maschinelles Lernen, das den gefolgerten Wert und/oder dessen Unsicherheit als Eingabe(n) hat. Die Ausgabegröße ist typischerweise eine physikalische Größe, deren Wert geschätzt wird. Dies ist jedoch nicht erforderlich. Als ein Beispiel kann die Ausgabegröße auch die Wertefunktion eines Steuerungsproblems sein, die an sich keine physikalische Größe ist, sondern die zum Beispiel verwendet werden kann, um ein optimales Rückmeldungsgesetz abzuleiten, und dadurch eine Online-Steuerung des computergesteuerten Systems auszuführen.

[0009] Der Wert der Ausgabegröße kann durch Verwenden der Gaußschen Prozess (GP) -Regression gefolgert werden. Wie an sich bekannt, kann bei der GP-Regression eine Kernelfunktion verwendet werden, um jeweilige Ähnlichkeiten erhaltener Messungen mit entsprechenden gekennzeichneten Instanzen zu berechnen, die vorherige Messungen oder typischer synthetische Instanzen sein können, die für die GP-Regression (zum Beispiel eine spärliche GP-Regression) erzeugt wurden. Der Wert der Ausgabegröße kann von den jeweiligen berechneten Ähnlichkeiten und jeweiligen Werten der Ausgabegröße der gekennzeichneten Instanzen gefolgert werden. Unterschiedliche Arten des Ausführens einer GP-Regression basierend auf einer Kernelfunktion, einschließlich einer spärlichen GP-Regression, sind dem Fachmann an sich bekannt, und können hierin wie zum Beispiel in CE Rasmussen und CKI Williams, „Gaussian Processes for Machine Learning“, The MIT-Press, 2006 beschrieben verwendet werden.

[0010] Um die Kernelfunktion für die GP-Regression anzuwenden, beabsichtigten die Erfinder interessanterweise, ein trainiertes niedrigrangiges Tensor-Netzwerkmodell als einen Merkmalsextrahierer zu verwenden, vorzugsweise ein Tensor-Train-Netzwerk. Wie an sich bekannt ist, kann ein niedrigrangiges Tensor-Netzwerkmodell ein Merkmal gemäß einem Gewichts-Tensor, der in niedrigrangigere Unter-Tensoren zerlegt wird, berechnen. Tensor-Netzwerkmodelle sind auch als Tensor-Zerlegungen oder Summen-Produkt-Netzwerke, oder als ein Typ einer eingeschränkten Boltzmann-Maschine bekannt. Im Allgemeinen kann das Berechnen eines Merkmals gemäß einem niedrigrangigen Tensor-Netzwerkmodell das Berechnen einer Linearkombination von Produkten jeweiliger typischerweise nichtlinearer Basisfunktionen umfassen, die auf jeweilige Eingabegrößen angewendet werden, wobei die Koeffizienten der Linearkombination durch einen Gewichts-Tensor definiert werden. Dass das trainierte Tensor-Netzwerkmodell „niedrigrangig“ ist, kann sich darauf beziehen, dass dieser Gewichts-Tensor in einer komprimierten Form dargestellt wird, nämlich durch eine Anzahl trainierter Parameter, die kleiner ist als die Anzahl der Summanden der Linearkombination. Insbesondere kann die Anzahl der Parameter in der Anzahl der Eingabegrößen subexponentiell sein, während die Anzahl der Produkte von Basisfunktionen im Allgemeinen exponentiell skaliert.

[0011] Um das niedrigrangige Tensor-Netzwerkmodell in der GP-Regressions-Kernelfunktion zu verwenden, kann das Tensor-Netzwerkmodell auf die Messungen der einen oder mehreren physikalischen Eingabegrößen und auf Werte der physikalischen Eingabegrößen von gekennzeichneten Instanzen der GP-Regression angewendet werden, was in entsprechenden Merkmalsdarstellungen resultiert. Eine Basis-Kernelfunktion kann dann auf die Merkmalsdarstellungen angewendet werden, um Ähnlichkeiten der erhaltenen Messungen mit den gekennzeichneten Instanzen zu erhalten, auf deren Basis die GP-Regression angewendet werden kann, um den Wert der Ausgabegröße zu folgern.

[0012] Insbesondere kann das niedrigrangige Tensor-Netzwerkmodell ein hierarchisches Tensor-Modell umfassen, vorzugsweise ein Tensor-Train-Netzwerk. Hierarchische Tensor-Modelle und insbesondere Baum-

Tensor-Modelle stellen einen guten Kompromiss zwischen dem Bereitstellen einer ausdrucksstarken Funktionsdarstellung und einer begrenzten Speicherungs- und Rechenkomplexität bereit. Insbesondere haben Experimente gezeigt, dass ein Tensor-Train-Netzwerk, das ein bestimmtes Baum-Tensor-Modell ist, besonders gut funktioniert. Die Komponenten des Tensor-Train können dazu ausgelegt sein, einen einheitlichen Rang zu haben. Dies begrenzt das Ausmaß der Hyperparameteroptimierung, und es stellte sich heraus, dass es gute Ergebnisse liefert, insbesondere wenn es mit dem alternierenden linearen Schema wie hierin beschrieben kombiniert wird. Anstelle eines Tensor-Train kann auch ein hierarchisches Tucker-Modell, ein Multi-Scale-Entanglement-Renormalization-Ansatz (MERA) -Modell oder ein Correlator-Product-States (CPS) -Modell verwendet werden. Alle diese verschiedenen Modelle haben Eigenschaften, die für spezifische Problembereiche vorteilhaft sein können.

[0013] Die beschriebene Verwendung eines Tensor-Train-Netzwerks oder eines anderen Typs eines niedrig-rangigen Tensor-Netzwerkmodells für die GP-Regression in dem Zusammenhang mit Steuerungssystemen hat verschiedene Vorteile. Im Vergleich zu der GP-Regression durch Verwenden anderer Typen von Modellen für maschinelles Lernen, zum Beispiel durch Verwenden eines tiefen neuronalen Netzwerks, erleichtert der bereitgestellte Ansatz das Lernen deutlich ausdrucksstärkerer Funktionen. Gleichzeitig sind die Berechnungskosten aufgrund der durch das niedrigrangige Tensor-Netzwerkmodell bereitgestellten Komprimierung geringer. Zusätzlich wird eine Überanpassung mit dem bereitgestellten Kompositionsmodell vermieden, indem seine inhärenten Regularisierungseigenschaften, insbesondere die Eigenschaften der Basisfunktionen und der niedrigrangigen Struktur, ausgenutzt werden. Daraus resultieren Rückschlüsse, die zuverlässiger sind, und/oder deren Zuverlässigkeit besser geschätzt werden kann, was zu einer zuverlässigeren Steuerung des autonomen Fahrzeugs oder anderer computergesteuerter Systeme führt.

[0014] Insbesondere durch Auswählen von Basisfunktionen von einem spezifischen Funktionsraum basierend auf der Kenntnis der Typen der dargestellten physikalischen Größen, ermöglichen die Techniken die explizite Einführung von Vorwissen über den Typ der verwendeten Sensordaten, und dadurch eine explizite Regularisierung und ein verringertes Risiko von Überanpassung. Die niedrigrangige Struktur des Tensor-Netzwerkmodells stellt auch eine inhärente Regularisierung bereit. Die Regularisierung ist ein direktes Merkmal des Modells selbst, anstelle eines Merkmals, das (teilweise) obendrauf enthalten sein kann. Zusätzlich führt die niedrigrangige Struktur eine implizite Merkmalsauswahl an den Daten aus, wodurch aussagekräftigere Merkmale für die GP-Regression bereitgestellt werden. Daher verbessert die Verwendung eines niedrig-rangigen Tensor-Netzwerkmodells die GP-Regression, insbesondere für hochdimensionale Daten, wie zum Beispiel Bilddaten oder Zeitreihen. Darüber hinaus stellen die strengen mathematischen Grundlagen des GP-Regressionsmodells einerseits und des Tensor-Netzwerkmodells andererseits auch eine verbesserte Interpretierbarkeit bereit.

[0015] Um die Leistung der bereitgestellten Techniken zu bewerten, haben die Erfinder Schätzungen ihres Generalisierungsfehlers mit einer Anzahl von Basismodellen auf mehreren synthetischen und realen Datensätzen verglichen. Die experimentellen Ergebnisse zeigen, dass die eingebrachten Tensor-Netzwerke eine hochgenaue GP-Regression ermöglichen. Während sie signifikant geringere Speicherkomplexität und geringeren Rechenaufwand aufweist, ist die beobachtete Leistung den untersuchten Standardmodellen, insbesondere tiefen neuronalen Netzwerken mit einer viel größeren Anzahl von Parametern, gewöhnlich um eine Größenordnung im mittleren quadratischen Fehler, klar überlegen.

[0016] Vergleicht man den beschriebenen Ansatz mit bekannten Verwendungen des Tensor-Netzwerkmodells in Steuerungssystemen, zum Beispiel, um Wertefunktionen für optimale Steuerungssysteme zu berechnen, verwendet der beschriebene Ansatz ein niedrigrangiges Tensor-Netzwerkmodell auf eine spezifische Weise, nämlich als Merkmalsextrahierer in einer GP-Regression. Dies hat die oben diskutierten Vorteile, einschließlich einer genauen Zuverlässigkeitsschätzung aufgrund des Vergleichs mit gekennzeichneten Instanzen. Die beschriebenen Techniken unterscheiden sich von direkten Bayesschen Ansätzen für Tensor-Netzwerkmodelle, indem ein Tensor-Netzwerkmodell verwendet wird, das selbst nicht Bayesisch ist, und dieses nicht Bayessche Modell in einen Gaußschen Prozess integriert wird, um Zuverlässigkeitsschätzungen zu erhalten. Dadurch stellen die beschriebenen Techniken Modelle bereit, die einfacher zu trainieren sind und weniger Trainingsdaten benötigen.

[0017] Verschiedene Ausführungsformen betreffen das Training der beschriebenen GP-Regression. Die GP-Regression kann durch eine Anzahl von Hyperparametern, die insbesondere die trainierbaren Parameter des Tensor-Netzwerkmodells beinhalten können, parametrisiert werden. Das Training kann auf eine Ende-zu-Ende-Weise ausgeführt werden, indem die Gaußsche Prozessregression an den gemessenen physikalischen Eingabegrößen von Trainingsinstanzen ausgeführt wird, um Vorhersagen der Ausgabegröße zu erhalten.

ten, wobei diese Vorhersagen mit Grundwahrheitswerten der Ausgabegröße verglichen werden, um ein Trainingssignal zu erhalten, und indem trainierbare Parameter der Gaußschen Prozessregression basierend auf dem Trainingssignal aktualisiert werden.

[0018] Die bereitgestellten Techniken können in einem weiten Bereich von Steuerungssystemen angewendet werden. Insbesondere können die bereitgestellten Techniken verwendet werden, um Werte einer oder mehrerer physikalischer Größen in Bezug auf das gesteuerte System und/oder Objekte in seiner Umgebung zu bestimmen. In einer Ausführungsform können die physikalischen Größen eine Entfernung, eine Geschwindigkeit, eine Beschleunigung und/oder eine verfolgte Position eines Objekts umfassen. Diese Werte können zum Beispiel in ein Steuerungsmodell eingegeben werden, um Steuerungsdaten durch Verwenden von Techniken abzuleiten, die an sich bekannt sind. In einer Ausführungsform kann das computergesteuerte System ein autonomes oder halb-autonomes Fahrzeug oder eine andere anspruchsvolle Anwendung sein, wie zum Beispiel ein Roboter, eine Fertigungsmaschine, ein Haushaltsgerät, ein persönlicher Assistent, ein Gebäude, usw.

[0019] Optional kann die vorhergesagte Ausgabegröße einen Fortschritt eines Fertigungsroboters, zum Beispiel in der Halbleiterproduktion, darstellen. Der Fortschritt kann auf einer Längenskala, zum Beispiel in Zentimeter oder Nanometer, dargestellt werden. Der gefolgerte Fortschritt kann verwendet werden, um den Fertigungsroboter und/oder andere Komponenten des Fertigungsprozesses zu steuern. Zum Beispiel kann ein Steuerungsparameter des Fertigungsprozesses, wie zum Beispiel eine Geschwindigkeit, ein angewendeter Druck, eine zu verwendende Materialmenge, usw., angepasst werden. Die Eingabemessungen können an dem zu fertigenden Produkt ausgeführte Messungen und/oder Messungen des Fertigungssystems beinhalten, wie zum Beispiel eine Geschwindigkeit, einen angewendeten Druck, eine Temperatur, eine Dicke, usw.

[0020] Optional kann die vorhergesagte Ausgabegröße eine Zeit bis zum Ausfall einer Komponente in einem komplexen System, wie zum Beispiel einem Motor, einem Robotergelenk oder einer Montagelinie, darstellen. Die vorhergesagte Ausgabegröße kann zur automatischen Steuerung verwendet werden, zum Beispiel durch Umschalten in einen ausfallsicheren Modus, und/oder um einen Alarm auszulösen, um den vorhergesagten Ausfall der Komponente zu melden. Die Eingaben zum Treffen der Vorhersage können in diesem Fall Messungen der Komponente und/oder anderer Komponenten des Systems umfassen, optional kombiniert mit Nutzungsdaten. Zum Beispiel können Messungen von Vibration und/oder Schmiermitteln und/oder Druck und/oder Geräusch und/oder Temperatur verwendet werden.

[0021] Optional kann die Regressionsaufgabe die Lokalisierung eines Roboters basierend auf Sensormessungen, zum Beispiel von einem Kilometerzähler, einem Infrarotsensor, einem Laserscanner und/oder einem Ultraschallsensor, sein. Der gefolgerte Standort, zum Beispiel x- und y-Koordinaten, kann zum Beispiel zur Steuerung verwendet werden, und/oder einem Benutzer angezeigt, zum Beispiel auf einer Karte visualisiert werden.

[0022] Optional kann die Gaußsche Prozessregression ferner einen Unsicherheitswert bestimmen, der eine Unsicherheit des gefolgerten Werts angibt. Auch dieser Unsicherheitswert kann durch das Steuerungssystem verwendet werden, um die Steuerungsdaten abzuleiten. Durch Verwenden des Tensor-Netzwerkmodells können besonders genaue Ähnlichkeiten von Messungen mit gekennzeichneten Instanzen berechnet werden, und dadurch kann ein besonders genauer Unsicherheitswert abgeleitet werden. Solche Unsicherheitswerte sind wichtig, um eine robuste Steuerung computergesteuerter Systeme auszuführen, insbesondere bei anspruchsvollen Anwendungen, wie zum Beispiel dem autonomen Fahren.

[0023] Optional kann das niedrigrangige Tensor-Netzwerkmodell eine Linearkombination von Produkten jeweiliger Basisfunktionen, die auf jeweilige physikalische Eingabegrößen angewendet werden, darstellen. Die verwendeten Basisfunktionen können von einer L2-Orthonom-Polynombasis, einer Fourier-Basis und/oder einer trigonometrischen Basis ausgewählt werden. Obwohl diese Basen unendlichdimensional sind, wird typischerweise eine endliche Teilmenge von Basisfunktionen ausgewählt und für das niedrigrangige Tensor-Netzwerkmodell verwendet. Interessanterweise kann die Basis, von der die Basisfunktionen ausgewählt werden, so gewählt werden, wie für die vorliegenden Sensordaten geeignet. Zum Beispiel können die Eingabegrößen eine oder mehrere geometrische Eingabegrößen, für die eine Fourier-Basis und/oder eine trigonometrische Basis verwendet werden kann, umfassen.

[0024] Beim Ausführen des Ende-zu-Ende-Trainings der GP-Regression können die trainierbaren Parameter, die aktualisiert werden, mindestens die trainierbaren Parameter des niedrigrangigen Tensor-Netzwerkmodells beinhalten. Die trainierbaren Parameter können optional ferner trainierbare Parameter der Basis-Kernel-

funktion beinhalten, obwohl es auch möglich ist, eine Basis-Kernelfunktion zu verwenden, die keine trainierbaren Parameter hat. Die trainierbaren Parameter können optional ferner Parameter der GP-Regression beinhalten. Insbesondere können die Parameter der GP-Regression Werte der Eingabe- und Ausgabegrößen von induzierenden gekennzeichneten Instanzen beinhalten. Diese induzierenden gekennzeichneten Instanzen können dann als synthetische gekennzeichnete Instanzen zum Ausführen der GP-Regression verwendet werden, im Gegensatz zum Verwenden tatsächlicher Messungen; zum Beispiel kann die GP-Regression eine sogenannte spärliche GP-Regression sein.

[0025] Optional kann das Training der GP-Regression das Verwenden separater Lernraten für die Parameter des niedrigrangigen Tensor-Netzwerkmodells, die Parameter der Basis-Kernelfunktion (falls vorhanden) und die induzierten gekennzeichneten Instanzen (falls vorhanden) umfassen. Das niedrigrangige Tensor-Netzwerkmodell kann ein Baum-Tensor-Netzwerk umfassen, das mehrere Unterkomponenten umfasst, wobei in diesem Fall auch separate Lernraten für die jeweiligen Unterkomponenten verwendet werden können. Es stellte sich heraus, dass die Verwendung separater Lernraten die Qualität des trainierten Modells erheblich verbessert.

[0026] Optional kann vor dem Ausführen des Ende-zu-Ende-Trainings das niedrigrangige Tensor-Netzwerkmodell trainiert werden, um die Ausgabegröße von den physikalischen Eingabegrößen zu folgern. Auf diese Weise können Anfangsparameter für das Tensor-Netzwerkmodell erhalten werden, die durch das Ende-zu-Ende-Training weiter verfeinert werden können. Das Tensor-Netzwerkmodell kann trainiert werden, um eine einzelne Ausgabe zu berechnen, die der Ausgabegröße entspricht. Das vollständige Modell oder ein Teil davon, zum Beispiel der Ausgabekern, kann dann repliziert werden, um ein Tensor-Netzwerkmodell zu erhalten, das mehrere Ausgaben bereitstellt, die der Größe der Merkmalsdarstellung entsprechen. Das Ausführen dieses Vortrainings resultiert in einer genaueren GP-Regression. Da außerdem das Vortraining die wohlverstandene mathematische Struktur des niedrigrangigen Tensor-Netzwerkmodells nutzen kann, kann es durch Verwenden speziell auf diese mathematische Struktur zugeschnittener Optimierungstechniken ausgeführt werden. Dies kann das Training besonders effizient machen, zum Beispiel kann das Hinzufügen von Vortraining die Gesamttrainingseffizienz erhöhen; und es kann bessere Garantien bereitstellen, dass verglichen mit der Anwendung von generischen Optimierungstechniken eine optimale Lösung erreicht wird. Experimentell wurde ein starker Einfluss der Einbeziehung des Vortrainings auf die Qualität der erhaltenen Modelle festgestellt.

[0027] Optional kann dieses Training, wenn das Vortraining des niedrigrangigen Tensor-Netzwerkmodells ausgeführt wird, das wiederholte Trainieren jeweiliger Unterkomponenten des Netzwerks umfassen, während andere Unterkomponenten fixiert bleiben. Dies ist auch als alternierendes lineares Schema bekannt. Insbesondere kann das niedrigrangige Tensor-Netzwerk ein Baum-Tensor-Netzwerk umfassen, wie zum Beispiel einen Tensor-Train, der eine solche Aufteilung in separat trainierbare Unterkomponenten zulassen kann. Insbesondere in dem Fall von Baum-Tensor-Netzwerken kann eine Unterkomponente durch Verwenden der Kleinste-Quadrate-Optimierung effizient optimiert werden, wodurch es ermöglicht wird, dass ein Optimum für die Unterkomponente genau und effizient berechnet werden kann. Aufgrund der Struktur des Baum-Tensor-Netzwerkmodells kann eine solche wiederholte lokale Optimierung der Unterkomponenten zu einem gut funktionierenden Gesamt-Tensor-Netzwerkmodell führen. Insbesondere kann die Unterkomponente effizient trainiert werden, indem ein lineares System in einem Satz von Parametern der Unterkomponente gelöst wird, indem eine LU-Faktorisierung, ein Krylov-Unterraumverfahren usw. ausgeführt wird.

[0028] Optional kann das Training der GP-Regression das Ausführen einer Hyperparameteroptimierung des Gaußschen Prozesses umfassen. Die Hyperparameter können Tensor-Train-Ränge (getrennt oder einheitlich) und/oder Anzahl von Basispolynomen und/oder Größe der Merkmalsraumdarstellung und/oder eine Anzahl von induzierenden Instanzen und/oder Lernrate(n) (vorzugsweise getrennt für Basis-Kernel, induzierende Instanzen und/oder Tensor-Netzwerkmodellkomponenten) umfassen. Die Hyperparameteroptimierung kann durch Zufallssuche ausgeführt werden. Dies hat den Vorteil, dass es im Gegensatz zu alternativen Methoden, wie zum Beispiel der Rastersuche, nicht dem Fluch der Dimensionalität unterliegt; und insbesondere weist es die vorteilhafte Eigenschaft auf, dass es dazu in der Lage ist, Unterräume gleichmäßig abzudecken. Dies ist besonders wichtig, da die meisten Funktionsapproximatoren nicht gleich stark von allen Hyperparametern abhängen, was bedeutet, dass einige Hyperparameter wichtiger sind als andere, und daher diese Unterräume sorgfältiger abgedeckt werden sollten. Die Zufallssuche wird aufgrund ihrer begrenzten Berechnungskosten bevorzugt, aber es ist auch möglich, eine verschachtelte Kreuzvalidierung auszuführen, in Fällen, in denen es eine besondere Notwendigkeit gibt, eine Überanpassung zu vermeiden.

[0029] Im Allgemeinen kann das Steuerungssystem verschiedene Typen von physikalischen Eingabegrößen messen, auf die die beschriebene GP-Regression angewendet werden kann. Die Eingabegrößen können einzelne Sensormesswerte umfassen, zum Beispiel skalare physikalische Größen, zum Beispiel höchstens oder mindestens 5 oder höchstens oder mindestens 10; oder Zeitreihen von solchen individuellen Sensormesswerten, zum Beispiel Zeitreihen von oder mindestens 20 Einträgen, höchstens oder mindestens 50 Einträgen, oder höchstens oder mindestens 100 Einträgen. Die eingegebenen physikalischen Größen können auch Bilddaten, zum Beispiel Videodaten, Radardaten, LiDAR-Daten, Ultraschalldaten, Bewegungsdaten, Wärmebilddaten oder Sonardaten, umfassen. Ein für den jeweiligen Typ von Eingabedaten geeignetes Tensor-Netzwerkmodell kann wie an sich bekannt verwendet werden, zum Beispiel ein hierarchisches Tensor-Netzwerkmodell für Bilder, usw.

[0030] Fachleute werden anerkennen, dass zwei oder mehr der oben erwähnten Ausführungsformen, Implementierungen und/oder optionalen Aspekte der Erfindung auf jegliche als nützlich erachtete Weise kombiniert werden können.

[0031] Modifikationen und Variationen jeglichen Systems und/oder jeglichen computerlesbaren Mediums, die den beschriebenen Modifikationen und Variationen eines entsprechenden computerimplementierten Verfahrens entsprechen, können durch einen Fachmann auf der Basis der vorliegenden Beschreibung durchgeführt werden.

Figurenliste

[0032] Diese und andere Aspekte der Erfindung werden von den beispielhaft beschriebenen Ausführungsformen in der folgenden Beschreibung und mit Bezug auf die beigefügten Zeichnungen ersichtlich und weiter verdeutlicht, in denen gilt:

Fig. 1 zeigt ein Trainingssystem zum Trainieren einer Gaußschen Prozessregression;

Fig. 2 zeigt ein Steuerungssystem zum Steuern eines computergesteuerten Systems;

Fig. 3 zeigt ein (halb-)autonomes Fahrzeug, das ein Steuerungssystem umfasst;

Fig. 4 zeigt ein detailliertes Beispiel dafür, wie eine GP-Regression ausgeführt wird;

Fig. 5 zeigt ein detailliertes Beispiel dafür, wie eine GP-Regression trainiert wird;

Fig. 6a zeigt ein detailliertes Beispiel trainierbarer Parameter eines niedrigrangigen Tensor-Netzwerkmodells;

Fig. 6b zeigt ein detailliertes Beispiel dafür, wie ein niedrigrangiges Tensor-Netzwerkmodell vortrainiert wird;

Fig. 7 zeigt ein computerimplementiertes Steuerungsverfahren zum Steuern eines computergesteuerten Systems;

Fig. 8 zeigt ein computerimplementiertes Trainingsverfahren zum Trainieren einer Gaußschen Prozessregression zur Verwendung beim Steuern eines computergesteuerten Systems;

Fig. 9 zeigt ein computerlesbares Medium, das Daten umfasst.

[0033] Es ist zu beachten, dass die Figuren rein schematisch und nicht maßstabsgetreu gezeichnet sind. In den Figuren können Elemente, die bereits beschriebenen Elementen entsprechen, dieselben Bezugszeichen haben.

Detaillierte Beschreibung von Ausführungsformen

[0034] **Fig. 1** zeigt Trainingssystem 100 zum Trainieren einer Gaußschen Prozess (GP) -Regression zur Verwendung beim Steuern eines computergesteuerten Systems, wie hierin beschrieben, zum Beispiel durch System 200 von **Fig. 2** oder **Fig. 3**. Die Gaußsche Prozessregression kann dazu dienen, einen Wert einer Ausgabegröße des computergesteuerten Systems und/oder seiner Umgebung basierend auf Messungen einer oder mehrerer physikalischer Eingabegrößen des computergesteuerten Systems und/oder seiner Umgebung zu folgern.

[0035] Das System 100 kann Datenschnittstelle 120 zum Zugreifen auf Trainingsdatensatz 030 von Trainingsinstanzen umfassen. Eine Trainingsinstanz des Trainingsdatensatzes 030 kann Werte der Eingabe-

und Ausgabegrößen umfassen. Zum Beispiel kann der Trainingsdatensatz höchstens oder mindestens 1.000, höchstens oder mindestens 10.000 oder höchstens oder mindestens 100.000 Trainingsinstanzen umfassen. Die Datenschnittstelle 120 kann auch dazu dienen, auf Regressionsdaten 040 zuzugreifen, um die Gaußsche Prozessregression, insbesondere trainierbare Parameter der GP-Regression, auszuführen. Zum Beispiel kann die Gesamtanzahl trainierbarer Parameter der Regressionsdaten 040 höchstens oder mindestens 1.000, höchstens oder mindestens 100.000 oder höchstens oder mindestens 1.000.000 betragen.

[0036] Die Regressionsdaten 040 können insbesondere Kernel-Hyperparameter einer Kernelfunktion und mehrere gekennzeichnete Instanzen umfassen. Die Kernel-Hyperparameter können Parameter eines niedrig-rangigen Tensor-Netzwerkmodells, zum Beispiel eines Tensor-Train-Netzwerks, umfassen. Diese Anzahl von Parametern kann zum Beispiel höchstens oder mindestens 1.000, höchstens oder mindestens 10.000 oder höchstens oder mindestens 100.000 betragen. Eine gekennzeichnete Instanz kann Werte der Eingabe- und Ausgabegrößen umfassen. Zum Beispiel kann die Anzahl gekennzeichneteter Instanzen höchstens oder mindestens 50, höchstens oder mindestens 100 oder höchstens oder mindestens 250 betragen. Die gekennzeichneten Instanzen der Regressionsdaten 040 können dem Trainingsdatensatz 030 entsprechen, oder ein Untersatz davon sein, sind jedoch typischer synthetische Instanzen, die durch das Trainingssystem 100 bestimmt werden. Die Regressionsdaten 040 können durch ein Steuerungssystem verwendet werden, um eine GP-Regression auszuführen, und ein computergesteuertes System basierend auf seinem Ergebnis zu steuern, zum Beispiel durch das System 200 von **Fig. 2** oder **Fig. 3**.

[0037] Zum Beispiel kann, wie auch in **Fig. 1** veranschaulicht, die Eingabeschnittstelle durch eine Datenspeicherungsschnittstelle 120 gebildet werden, die auf die Daten 030, 040 von Datenspeicherung 021 zugreifen kann. Zum Beispiel kann die Datenspeicherungsschnittstelle 120 eine Speicherschnittstelle oder eine beständige Speicherungsschnittstelle sein, zum Beispiel eine Festplatte oder eine SSD-Schnittstelle, aber auch eine persönliche, lokale oder Wide-Area-Network-Schnittstelle, wie zum Beispiel eine Bluetooth-, Zigbee- oder Wi-Fi Schnittstelle oder eine Ethernet- oder Glasfaserschnittstelle. Die Datenspeicherung 021 kann eine interne Datenspeicherung des Systems 100, wie zum Beispiel eine Festplatte oder SSD, aber auch eine externe Datenspeicherung, zum Beispiel eine netzwerkzugängliche Datenspeicherung, sein. In einigen Ausführungsformen kann auf die Daten 030, 040 jeweils von einer unterschiedlichen Datenspeicherung zugegriffen werden, zum Beispiel über ein unterschiedliches Subsystem der Datenspeicherungsschnittstelle 120. Jedes Subsystem kann von einem Typ sein, wie er oben für die Datenspeicherungsschnittstelle 120 beschrieben ist.

[0038] Das System 100 kann ferner Prozessorsubsystem 140 umfassen, das dazu ausgelegt sein kann, während des Betriebs des Systems 100 ein Ende-zu-Ende-Training der Gaußschen Prozessregression auszuführen. Das Ende-zu-Ende-Training kann das Ausführen der Gaußschen Prozessregression an physikalischen Eingabegrößen einer Trainingsinstanz des Trainingsdatensatzes 030 gemäß den Regressionsdaten 040, zum Beispiel wie mit Bezug auf **Fig. 2** beschrieben, umfassen, um eine Vorhersage der Ausgabegröße zu erhalten. Das Ende-zu-Ende-Training kann das Vergleichen der Vorhersage der Ausgabegröße mit einem Wert der Ausgabegröße in dem Trainingsdatensatz 030 umfassen, um ein Trainingssignal zu erhalten. Das Ende-zu-Ende-Training kann das Aktualisieren trainierbarer Parameter 040 der Gaußschen Prozessregression basierend auf dem Trainingssignal umfassen.

[0039] Das System 100 kann ferner eine Ausgabeschnittstelle zum Ausgeben von trainierten Regressionsdaten 040 umfassen, die das Training der GP-Regression darstellen. Zum Beispiel kann, wie auch in **Fig. 1** dargestellt, die Ausgabeschnittstelle durch die Datenschnittstelle 120 gebildet werden, wobei die Schnittstelle in diesen Ausführungsformen eine Eingabe/Ausgabe (Input/Output - IO)-Schnittstelle ist, über die die trainierten Regressionsdaten 040 in der Datenspeicherung 021 gespeichert werden können. Zum Beispiel können 'untrainierte' Regressionsdaten während oder nach dem Training mindestens teilweise durch 'trainierte' Regressionsdaten ersetzt werden, insofern als trainierbare Parameter der GP-Regression, wie zum Beispiel Parameter des niedrig-rangigen Tensor-Netzwerkmodells, angepasst werden können, um das Training in den Trainingsdaten 030 widerzuspiegeln. Dies ist auch in **Fig. 1** durch die Bezugszeichen 040 veranschaulicht, die sich sowohl auf die untrainierten Regressionsdaten als auch auf die trainierten Regressionsdaten beziehen. In anderen Ausführungsformen kann die trainierte Regression getrennt von den untrainierten Regressionsdaten gespeichert werden. In einigen Ausführungsformen kann die Ausgabeschnittstelle von der Datenspeicherungsschnittstelle 120 getrennt sein, kann jedoch im Allgemeinen von einem Typ wie oben für die Datenspeicherungsschnittstelle 120 beschrieben sein.

[0040] **Fig. 2** zeigt ein Steuerungssystem 200 zum Steuern eines computergesteuerten Systems, das mit Umgebung 082 interagiert.

[0041] Das System 200 kann Datenschnittstelle 220 zum Zugreifen auf Regressionsdaten 040 zum Ausführen einer Gaußschen Prozess (GP) -Regression umfassen. Die GP-Regression kann dazu dienen, einen Wert einer Ausgabegröße des computergesteuerten Systems und/oder seiner Umgebung basierend auf Messungen einer oder mehrerer physikalischer Eingabegrößen des computergesteuerten Systems und/oder seiner Umgebung zu folgern. Die Regressionsdaten 040 können Kernel-Hyperparameter einer Kernelfunktion und mehrere gekennzeichnete Instanzen umfassen. Eine gekennzeichnete Instanz kann Werte der Eingabe- und Ausgabegrößen umfassen. Die Regressionsdaten wurden zuvor gemäß einem Trainingsverfahren wie hierin beschrieben, zum Beispiel durch das System 100 von **Fig. 1**, bestimmt. Das System 200 kann die Regressionsdaten 040 zusätzlich zu ihrer Anwendung trainieren, zum Beispiel kann das System 200 mit dem System 100 von **Fig. 1** kombiniert werden.

[0042] Zum Beispiel kann, wie auch in **Fig. 2** veranschaulicht, die Datenschnittstelle durch eine Datenspeicherungsschnittstelle 220 gebildet werden, die auf die Daten 040 von Datenspeicherung 022 zugreifen kann. Im Allgemeinen können die Datenschnittstelle 220 und die Datenspeicherung 022 von einem selben Typ sein, wie unter Bezugnahme auf **Fig. 1** für die Datenschnittstelle 120 und die Datenspeicherung 021 beschrieben.

[0043] Das System 200 kann ferner Prozessor-Subsystem 240 umfassen, das dazu ausgelegt sein kann, während des Betriebs des Systems 200 Messungen 223 der einen oder mehreren physikalischen Eingabegrößen zu erhalten. Das Prozessor-Subsystem 240 kann ferner dazu ausgelegt sein, die Gaußsche Prozessregression auszuführen, um einen Wert der Ausgabegröße basierend auf den erhaltenen Messungen zu folgern. Das Prozessor-Subsystem 240 kann ferner dazu ausgelegt sein, Steuerdaten 225 basierend auf dem Wert der Ausgabegröße abzuleiten. Das Prozessor-Subsystem 240 kann ferner dazu ausgelegt sein, die Steuerdaten 225 an den Aktuator bereitzustellen.

[0044] Die GP-Regression kann das Anwenden der Kernelfunktion umfassen, um jeweilige Ähnlichkeiten der erhaltenen Messungen 223 mit den jeweiligen gekennzeichneten Instanzen der Regressionsdaten 040 zu berechnen. Um die Kernelfunktion anzuwenden, kann das Prozessor-Subsystem 240 ein trainiertes niedrig-rangiges Tensor-Netzwerkmodell, wie zum Beispiel ein durch Regressionsdaten 040 parametrisiertes Tensor-Train-Netzwerk, auf die Messungen 223 der einen oder mehreren physikalischen Eingabegrößen und auf Werte der einen oder mehreren physikalischen Eingabegrößen einer gekennzeichneten Instanz 040 anwenden, um Merkmalsdarstellungen der Messungen und der gekennzeichneten Instanz zu erhalten. Um die Kernelfunktion anzuwenden, kann das Prozessor-Subsystem 240 ferner eine Basis-Kernelfunktion auf die Merkmalsdarstellungen anwenden, um Ähnlichkeiten der erhaltenen Messungen mit den gekennzeichneten Instanzen zu erhalten. Um die GP-Regression auszuführen, kann das Prozessor-Subsystem 240 ferner den Wert der Ausgabegröße von den jeweiligen berechneten Ähnlichkeiten und von jeweiligen Werten der Ausgabegröße der gekennzeichneten Instanzen 040 folgern.

[0045] Es versteht sich, dass für das Prozessor-Subsystem 240 dieselben Überlegungen und Implementierungsoptionen gelten, wie für das Prozessor-Subsystem 140 von **Fig. 1**. Es versteht sich ferner, dass im Allgemeinen dieselben Überlegungen und Implementierungsoptionen für das System 200 gelten können, wie für das System 100 von **Fig. 1**, sofern nicht anders angegeben.

[0046] **Fig. 2** zeigt ferner verschiedene optionale Komponenten des Systems 200. Zum Beispiel kann in einigen Ausführungsformen das System 200 Sensorschnittstelle 260 zum direkten Zugreifen auf Sensordaten 224 umfassen, die durch einen oder mehrere Sensoren 072 in einer Umgebung 082 erfasst werden. Der/die Sensor(en) kann/können in der Umgebung 082 angeordnet sein, kann/können aber auch entfernt von der Umgebung 082 angeordnet sein, zum Beispiel wenn die Größe(n) entfernt gemessen werden kann/können. Der Sensor 072 kann, muss aber nicht, Teil des Systems 200 sein.

[0047] Der Sensor 072 kann jegliche geeignete Form haben, wie zum Beispiel ein Bildsensor, ein Lidarsensor, ein Radarsensor, ein Drucksensor, ein Temperatursensor usw. In einigen Ausführungsformen können die Sensordaten 072 Sensormessungen von unterschiedlichen physikalischen Größen umfassen, insofern, als sie von zwei oder mehr unterschiedlichen Sensoren, die unterschiedliche physikalische Größen erfühlen, erhalten werden können. Die Sensordatenschnittstelle 260 kann jegliche geeignete Form haben, die in ihrem Typ dem Typ des Sensors entspricht, einschließlich, aber nicht beschränkt auf eine Low-Level-Kommunikationsschnittstelle, zum Beispiel basierend auf I2C- oder SPI-Datenkommunikation, oder eine Datenspeicherungsschnittstelle eines Typs, wie oben für die Datenschnittstelle 220 beschrieben.

[0048] In einigen Ausführungsformen kann das System 200 Aktuatorschnittstelle 280 zum Bereitstellen von Steuerungsdaten 226 an Aktuator 092 in der Umgebung 082 umfassen. Solche Steuerungsdaten 226 können

durch das Prozessor-Subsystem 240 abgeleitet werden, um den Aktuator basierend auf dem gefolgerten Wert der Ausgabegröße zu steuern. Zum Beispiel kann die gefolgerte Ausgabegröße als Steuerungsdaten 226 verwendet werden, die eine physikalische Zielgröße für den Aktuator 092, zum Beispiel eine Zieltemperatur, einen Zieldruck, eine Zielgeschwindigkeit usw., darstellen, oder die Ausgabegröße kann durch ein weiteres Modell für maschinelles Lernen verwendet werden, um die Steuerungsdaten 226 abzuleiten. Der Aktuator kann Teil des Systems 200 sein. Zum Beispiel kann der Aktuator ein elektrischer, hydraulischer, pneumatischer, thermischer, magnetischer und/oder mechanischer Aktuator sein. Spezifische, jedoch nicht einschränkende Beispiele beinhalten Elektromotoren, elektroaktive Polymere, Hydraulikzylinder, piezoelektrische Aktuatoren, pneumatische Aktuatoren, Servomechanismen, Magnete, Schrittmotoren, usw. Ein solcher Typ von Steuerung wird unter Bezugnahme auf **Fig. 3** für ein (halb-)autonomes Fahrzeug beschrieben.

[0049] In anderen Ausführungsformen (in **Fig. 2** nicht gezeigt) kann das System 200 eine Ausgabeschnittstelle zu einem Wiedergabegerät, wie zum Beispiel einer Anzeige, einer Lichtquelle, einem Lautsprecher, einem Vibrationsmotor usw., umfassen, die verwendet werden können, um ein sensorisch wahrnehmbares Ausgabesignal zu erzeugen, das basierend auf dem gefolgerten Wert der Ausgabegröße erzeugt werden kann. Das sensorisch wahrnehmbare Ausgabesignal kann den gefolgerten Wert der Ausgabegröße direkt angeben, kann aber auch ein abgeleitetes sensorisch wahrnehmbares Ausgabesignal darstellen, zum Beispiel zur Verwendung bei der Führung, Navigation oder einem anderen Typ von Steuerung des computergesteuerten Systems.

[0050] Im Allgemeinen kann jedes in dieser Spezifikation beschriebene System, einschließlich, aber nicht beschränkt auf das System 100 von **Fig. 1** und das System 200 von **Fig. 2**, als ein einzelnes Gerät oder eine einzelne Vorrichtung, wie zum Beispiel eine Workstation oder ein Server, verkörpert werden, oder in diesen enthalten sein. Das Gerät kann ein eingebettetes Gerät sein. Das Gerät oder die Vorrichtung kann einen oder mehrere Mikroprozessoren umfassen, die geeignete Software ausführen. Zum Beispiel kann das Prozessor-Subsystem des jeweiligen Systems durch eine einzelne Zentraleinheit (Central-Processing-Unit - CPU), aber auch durch eine Kombination oder ein System solcher CPU und/oder anderer Typen von Verarbeitungseinheiten verkörpert werden. Die Software kann heruntergeladen und/oder in einem entsprechenden Speicher, zum Beispiel einem flüchtigen Speicher, wie zum Beispiel RAM, oder einem nichtflüchtigen Speicher, wie zum Beispiel Flash, gespeichert worden sein. Alternativ kann das Prozessor-Subsystem des jeweiligen Systems in dem Gerät oder der Vorrichtung in der Form einer programmierbaren Logik, zum Beispiel Field-Programmable-Gate-Array (FPGA) implementiert werden. Im Allgemeinen kann jede Funktionseinheit des jeweiligen Systems in der Form einer Schaltung implementiert werden. Das jeweilige System kann auch auf eine verteilte Weise implementiert werden, zum Beispiel durch Einbeziehen unterschiedlicher Geräte oder Vorrichtungen, wie zum Beispiel verteilter lokaler oder Cloud-basierter Server. In einigen Ausführungsformen kann das System 200 Teil eines Fahrzeugs, Roboters oder einer ähnlichen physischen Einheit sein, und/oder kann ein Steuerungssystem darstellen, das dazu ausgelegt ist, die physikalische Einheit zu steuern.

[0051] **Fig. 3** zeigt ein Beispiel des Obigen, insofern, als das Steuerungssystem 200 als ein Steuerungssystem von (halb)autonomen Fahrzeug 62 gezeigt wird, das in Umgebung 50 betrieben wird. Das autonome Fahrzeug 62 kann autonom sein, insofern, als es ein autonomes Fahrsystem oder ein Fahrassistenzsystem umfassen kann, wobei das Letztere auch als ein halbautonomes System bezeichnet wird. Das autonome Fahrzeug 62 kann zum Beispiel das System 200 beinhalten, um das Lenken und Bremsen des autonomen Fahrzeugs basierend auf Sensordaten, die von in das Fahrzeug 62 integrierter Videokamera 22 erhalten werden, zu steuern. Zum Beispiel kann das System 200 Elektromotor 42 steuern, um (regeneratives) Bremsen auszuführen, falls erwartet wird, dass das autonome Fahrzeug 62 mit einem Verkehrsteilnehmer kollidiert. Das System 200 kann das Lenken und/oder Bremsen steuern, um eine Kollision mit dem Verkehrsteilnehmer zu vermeiden. Zu diesem Zweck kann das System 200 die GP-Regression verwenden, um einen Wert einer physikalischen Größe des Fahrzeugs 62, zum Beispiel eine Entfernung, eine Geschwindigkeit oder eine Beschleunigung, zu folgern, und Steuerungsdaten basierend auf diesem gefolgerten Wert abzuleiten. Wenn zum Beispiel erwartet wird, dass der zukünftige Status des Fahrzeugs, zum Beispiel seine zukünftige Position relativ zu dem Verkehrsteilnehmer, in einer Kollision resultiert, kann das System 200 entsprechende Maßnahmen ergreifen.

[0052] **Fig. 4** zeigt ein detailliertes, jedoch nicht einschränkendes Beispiel dafür, wie eine Gaußsche Prozess (GP) -Regression ausgeführt wird;

[0053] Die GP-Regression kann dazu dienen, Ausgabewert IO 430 einer Ausgabegröße eines computergesteuerten Systems und/oder seiner Umgebung, basierend auf Eingabemessungen IM 420 einer oder mehrerer physikalischer Eingabegrößen des computergesteuerten Systems und/oder seiner Umgebung zu folgern.

Der Ausgabewert IO ist typischerweise ein Skalar, insbesondere ein reellwertiger Skalar. Die Eingabemessungen IM sind typischerweise auch Skalare, zum Beispiel reellwertige Skalare, und/oder können als ein reellwertiger Vektor dargestellt werden. Zum Beispiel können die Eingabemessungen IM durch höchstens oder mindestens 10, höchstens oder mindestens 20, oder höchstens oder mindestens 50 Skalarwerte dargestellt werden. In dem Beispiel dieser Figur wird die GP-Regression verwendet, um einen einzelnen Ausgabewert IO für einen einzelnen Satz von Eingabemessungen IM vorherzusagen, aber es ist auch möglich, die GP-Regression anzuwenden, um mehrere Ausgabewerte entsprechend den jeweiligen Sätzen von Eingabemessungen vorherzusagen

[0054] Die GP-Regression kann einen Satz gekennzeichneten Instanzen verwenden. Eine gekennzeichnete Instanz kann Werte der Eingabe- und Ausgabegrößen umfassen, zum Beispiel zeigt die Figur Eingabewerte LM1 421 und den entsprechenden Ausgabewert LO1 431 bis zu den Eingabewerten LMn 422 und dem entsprechenden Ausgabewert LOn 432. Eine gekennzeichnete Instanz LMi, LOi, ist im Allgemeinen von demselben Typ und derselben Dimensionen wie die Eingabemesswerte IM und der Ausgabewert IO. Zum Beispiel kann die Anzahl der in einer GP-Regression gekennzeichneten Instanzen höchstens oder mindestens 10, höchstens oder mindestens 100, oder höchstens oder mindestens 1.000 betragen.

[0055] Im Allgemeinen kann die GP-Regression den Wert IO der Ausgabegröße von jeweiligen Ähnlichkeiten SIM1 481, ..., SIMn 482 der erhaltenen Messungen IM zu den jeweiligen gekennzeichneten Instanzen LMi und von den jeweiligen Ausgabewerten LOi folgern. Insbesondere können die Ähnlichkeiten der Messungen IM mit den Eingabewerten LMi verwendet werden, um Gewichte zum Bestimmen des gefolgerten Ausgabewerts IO als eine Linearkombination der jeweiligen Ausgabewerte LOi abzuleiten. Diese Folgerung kann durch Regressionsoperation Reg 490 ausgeführt werden, von der Beispiele an sich bekannt sind.

[0056] Interessanterweise können die Ähnlichkeiten SIMi durch Anwenden von Kernel-funktion KF 450, die die Anwendung von trainiertem niedrigrangigem Tensor-Netzwerkmodell LRT 460, vorzugsweise eines Tensor-Train-Netzwerks, beinhaltet, berechnet werden. Die Figur zeigt Satz von Parametern PAR 440 der Kernel-funktion, einschließlich Parametern des trainierten Tensor-Netzwerkmodells. In dem Zusammenhang der GP-Regression sind Kernel auch als Kovarianzfunktionen bekannt. Die Parameter PAR der Kernelfunktion werden auch als Kernel-Hyperparameter bezeichnet.

[0057] Wie in der Figur gezeigt, kann das niedrigrangige Tensor-Netzwerkmodell LRT eine Merkmalsdarstellung seiner Eingabewerte berechnen, indem Produkte ΠPa 462 von jeweiligen Basisfunktionen, die auf jeweilige Eingabewerte angewendet werden, berechnet werden, zum Beispiel $\prod_{i=1}^d P_{\alpha_i}(x_i)$. Für jeden Eingabewert x_i , $i = 1, \dots, d$, einer Eingabemessung kann ein endlicher Satz $\{P_{\alpha_i}\}$ von Basisfunktionen, zum Beispiel ein Abschneiden einer unendlich großen Funktionsbasis, definiert werden. Zum Beispiel kann ein gemeinsamer Satz von Basisfunktionen verwendet werden, oder es können entsprechende Sätze von Basisfunktionen für die jeweiligen Eingabewerte verwendet werden. Zum Beispiel kann die Anzahl von Funktionen in einer Basis höchstens oder mindestens 2, höchstens oder mindestens 10, oder höchstens oder mindestens 25 betragen.

[0058] Die Merkmalsdarstellung FR^* kann durch Berechnen von Linearkombination LC 463 der Produkte berechnet werden, wobei Gewichte Wa 461 der Linearkombination von den Parametern PAR der Kernelfunktion abgeleitet werden, zum Beispiel kann ein Merkmal berechnet werden als

$$f(x) = \sum_{\alpha_1=1}^{J_1} \dots \sum_{\alpha_d=1}^{J_d} W_{\alpha_1, \dots, \alpha_d} \prod_{i=1}^d P_{\alpha_i}(x_i).$$
 Die Gewichte Wa können einen Koeffizienten-Tensor der Ordnung d bilden, zum Beispiel $W \in \otimes_{i=1}^d \mathbb{R}^{J_i}$, wobei d die Eingabedimension des Tensor-Netzwerks ist. Dass das Tensor-Netzwerk „niederrangig“ ist, bezieht sich auf die Anzahl der Parameter PAR, die verwendet werden, um den Gewichts-Tensor W zu definieren, der kleiner ist als die Anzahl der Summanden der Linearkombination, mit anderen Worten, kleiner als die Dimension des Tensors. Während insbesondere die Größe des Tensors in der Dimension d exponentiell wachsen kann, kann die Anzahl der Parameter PAR, die zum Definieren des Tensors verwendet werden, subexponentiell sein, wofür hierin verschiedene Möglichkeiten beschrieben werden. Wie an sich bekannt, beinhaltet die Berechnung der Linearkombination LC typischerweise auch keine Berechnung der separaten Gewichte Wa und/oder Summanden der Linearkombination, zum Beispiel kann auch die Berechnung der Linearkombination eine Komplexität haben, die in der Dimension d subexponentiell ist.

[0059] Eine durch das Tensor-Netzwerkmodell LRT berechnete Merkmalsdarstellung FR^* umfasst typischerweise mehrere Merkmalswerte, zum Beispiel jeder berechnet als eine Linearkombination gemäß einem jeweiligen Gewichts-Tensor Wa . Typischerweise überlappen sich die Parameter PAR, die verwendet werden,

um die jeweiligen Gewichts-Tensoren zu definieren, miteinander, was in einer geringeren Speicherkomplexität und einem effizienteren Training resultiert.

[0060] Mathematisch kann die Berechnung eines Merkmals durch das Tensor-Netzwerkmodell LRT als eine Auswertung einer Zufallsvariablen Y betrachtet werden, die von den Werten der Eingabegrößen abhängt, und eine d -dimensionale Zufallsvariable Y bildet. Das Tensor-Netzwerkmodell kann die Eingabegrößen X_i als unabhängige $(B(X_i), X_i)$ Zufallsvariablen mit $X_i \subset \mathbb{R}$ und einem Wahrscheinlichkeitsmaß ρ_i für $i = 1, \dots, d$ behandeln. Dann kann $X := (X_1, \dots, X_d)$ als eine $(B(X), X)$ Zufallsvariable mit $\mathcal{X} := \prod_{i=1}^d \mathcal{X}_i$ und $\rho := \otimes_{i=1}^d \rho_i$ definiert werden. Die Berechnung kann durch Beziehung $Y = \Phi(X)$ dargestellt werden, wobei Φ durch das Tensor-Netzwerkmodell f angenähert wird. Es sei $f \in F(X)$, dann kann der F Funktionsraum mit einer Tensor-Produktstruktur, zum Beispiel $\mathcal{F} = \otimes_{i=1}^d \mathcal{F}_i(X_i)$, ausgestattet sein. Ohne Verlust der Allgemeingültigkeit kann davon ausgegangen werden, dass $\mathcal{F} \subset L^2_\rho(\mathcal{X})$. Die Funktion kann als eine Linearkombination von Produkten von Basisfunktionen wie folgt dargestellt werden. Es sei $\{P_j\} \mid j = 1, 2, 3, \dots\} \in F_i(X_i)$ für $i = 1, \dots, d$ eine Folge von orthonormalen Basisfunktionen, die eine eindeutige Expansion jeglicher Funktion in F_i ergeben. Ein solcher Satz von Basisfunktionen kann auf endlichdimensionale Unterräume beschränkt werden, indem Abschneiden angewendet wird, zum Beispiel:

$$\hat{\mathcal{F}}_i := \text{span}\left(P_i^1, \dots, P_i^{J_i}\right), \quad i = 1, \dots, d, \quad J_i \in \mathbb{N}.$$

[0061] Die Produkte der Basisfunktionen können einen endlichdimensionalen Tensor-Produktraum bilden, zum Beispiel:

$$\hat{\mathcal{F}}_A \otimes_{i=1}^d \hat{\mathcal{F}}_i \subseteq \mathcal{F}(\hat{\mathcal{X}}),$$

mit $A_{i=1}^d \{1, \dots, J_i\}$, einem Satz von d -Tupeln von Indizes $\alpha_i = 1, \dots, J_i$. Durch Verwenden von Multiindex-Notation kann eine endliche Basis von $\hat{\mathcal{F}}_A$ definiert werden durch $\{P_{\alpha}\}_{\alpha \in A}$, wobei

$$P_{\alpha} \otimes_{i=1}^d P_{\alpha_i}$$

für $\alpha = (\alpha_1, \dots, \alpha_d) \in A$. Auf diese Weise kann eine Darstellung von f erhalten werden als eine Linearkombination von Produkten der Basisfunktionen:

$$f(x) = \sum_{\alpha_1=1}^{J_1} \dots \sum_{\alpha_d=1}^{J_d} W_{\alpha_1, \dots, \alpha_d} \prod_{i=1}^d P_{\alpha_i}(x_i)$$

für Proben $x = (x_1, \dots, x_d) \in X$, zum Beispiel Eingabemessungen IM oder Eingabewerte LMi von einer gekennzeichneten Instanz. Hier ist W der Gewichts-Tensor W_a , in diesem Beispiel ein Koeffizienten-Tensor der Ordnung d , zum Beispiel: $W \in \otimes_{i=1}^d \mathbb{R}^{J_i}$.

[0062] Von der obigen Beschreibung ist anzumerken, dass im Allgemeinen eine Summe von Produkten von Basisfunktionen eine exponentielle Anzahl von Interaktionen zwischen Datenmerkmalen modellieren kann, was in einer Komplexität von $O(\max(J_1, \dots, J_d)^d)$ resultiert. Während dies eine große Ausdruckskraft bedeutet, stellt es jedoch gleichzeitig eine erhebliche Berechnungsbelastung aufgrund des exponentiellen Komplexitätswachstums in der Dimensionalität d dar, bekannt als der Fluch der Dimensionalität.

[0063] In einem niedrigrangigen Tensor-Netzwerkmodell LRT wird dieser Fluch der Dimensionalität interessanterweise gemildert, indem der Gewichts-Tensor W_a basierend auf einem Satz von Parametern PAR definiert wird, der in seiner Größe kleiner ist als die Anzahl der Summanden der Linearkombination. Hierfür sind mehrere Möglichkeiten an sich bekannt, die hierin verwendet werden können. Gute Ergebnisse wurden durch Verwenden eines Tensor-Train (TT) -Modells erhalten, wie detaillierter mit Bezug auf die **Fig. 6a** und **Fig. 6b** diskutiert. Allgemeiner kann ein Baum-Tensor-Modell oder ein anderer Typ einer hierarchischen Tensor-Darstellung verwendet werden, von denen ein TT-Modell ein Sonderfall ist. Tensor-Train-Modelle sind auch als Matrix-Product-State (MPS) -Modelle bekannt. Im Allgemeinen sind niedrigrangige Tensor-Netzwerkmodelle auch als Tensor-Zerlegungen bekannt. Andere Typen von niedrigrangigen Tensor-Netzwerkmodellen, die verwendet werden können, beinhalten Hierarchical-Tucker (HT) -Modelle, Multi-Scale-Entanglement-Renormalization-Ansatz (MERA) -Modelle und Correlator-Product-States (CPS) -Modelle, von denen

alle Eigenschaften haben, die für spezifische Problembereiche vorteilhaft sein können. Tensor-Netzwerke sind auch als Summen-Produkt-Netzwerke bekannt, und können als ein Spezialfall eingeschränkter Boltzmann-Maschinen betrachtet werden.

[0064] Wie in der Figur gezeigt, können die Daten IM, LMi zuerst durch das Tensor-Netzwerkmodell LRT, zum Beispiel ein Tensor-Train-Modell, verbreitet werden, um jeweilige Merkmalsdarstellungen FRI 470, FR1 471 ..., FRn 472 zu erhalten. Anschließend kann die GP-Regression an den resultierenden Ausgaben FR* ausgeführt werden. Somit kann das Tensor-Netzwerkmodell LRT effektiv als ein Merkmalsextrahierer für die GP dienen. Mathematisch wird das Tensor-Netzwerkmodell LRT als $g_\gamma \in M_r$ mit $g_\gamma: X \rightarrow Z$ bezeichnet, und es sei $h \in K$, BKF 480 eine Kernelfunktion, die als die Basis-Kernelfunktion bezeichnet wird, und auf die Darstellungen FR* angewendet wird, um die Ähnlichkeiten SIMi zu erhalten. Zum Beispiel mathematisch, $h: Z \rightarrow Y$, wobei K der reproduzierende Kernel-Hilbert-Raum ist, der durch die Kovarianzfunktion BKF, $\hat{k}: Z \times Z \rightarrow \mathbb{R}$ der GP bestimmt wird. Das niedrigrangige Tensor-Netzwerkmodell LRT und die Basis-Kernelfunktion BKF können zusammen eine zusammengesetzte Kernelfunktion KF bilden, die wie folgt bezeichnet wird:

$$k(x, x') := \hat{k}(g_\gamma(x), g_\gamma(x') | \theta),$$

wobei $\theta = \{\gamma, \theta\}$ die Parameter der Kernelfunktion KF sind, einschließlich trainierter Parameter γ des Tensor-Netzwerkmodells LRT und trainierter Parameter θ der Basis-Kernelfunktion BKF, falls vorhanden. In Bezug auf diesen Kernel kann die gesamte GP-Regression bezeichnet werden als

$$f(x) = (g_\gamma \circ h)(x) = \sum_{n=1}^N k(x, x^{(n)} | \theta) c_n,$$

wobei f ein Element des reproduzierenden Kernel-Hilbert-Raums H ist, der durch den zusammengesetzten Kernel k , KF bestimmt wird.

[0065] Für die Basis-Kernelfunktion $\hat{k}$, BKF, sind verschiedene Optionen möglich. Die Basis-Kernelfunktion hängt normalerweise von einem Satz von Hyperparametern θ ab, die in den Parametern PARS der Kernelfunktion enthalten sind, zum Beispiel:

$$\hat{k}(x, x') = \hat{k}(x, x' | \theta).$$

[0066] Zum Beispiel kann der RBF-Kernel verwendet werden:

$$k_{RBF}(x, x') = \sigma_f^2 \exp\left(-\frac{1}{2\ell} \|x - x'\|^2\right)$$

in diesem Fall die vorherige Standardabweichung σ_f , auch als die Signalvarianz bekannt; die Längenskala ℓ , die die Rate des Korrelationsabfalls mit zunehmendem Abstand zwischen den Eingaben bestimmt; und/oder die Rauschvarianz (zur Vereinfachung der Darstellung nicht in der obigen Formel enthalten) kann trainierte Hyperparameter sein, zum Beispiel $\theta = \{\sigma_f, \ell, \sigma_n\}$.

[0067] Zurückkommend auf die Regressionsoperation Reg, wird nun ein detailliertes mathematisches Beispiel für das Ausführen einer GP-Regression gegeben. Mathematisch kann die GP-Regression als eine Technik zum Konstruieren einer Funktion betrachtet werden, die beobachtete Daten approximiert. Konkret kann eine Zufallsvariable Y geschätzt werden, die von einer d -dimensionalen Zufallsvariablen X abhängt, durch Proben LMi, LOi des Pairs (X, Y) . Hier kann $X := (X_1, \dots, X_d)$ als eine $(B(X), \cdot)$ Zufallsvariable definiert werden. Unter der Annahme $Y = \Phi(X)$ kann die GP-Regression Φ durch eine Funktion f approximieren. Insbesondere können die Proben LMi, LOi Realisierungen der Zufallsvariablen X, Y sein, mit $X = [x^{(1)}, \dots, x^{(N)}]$ gezeichnet gemäß p und $y = [y^{(1)}, \dots, y^{(N)}]$ mit $y^{(n)} = \Phi(x^{(n)})$ für alle $n = 1, \dots, N$. Bei der GP-Regression kann, um Φ durch f zu approximieren, eine vorherige Gaußsche Prozesses (GP) -Verteilung f_0 zugeordnet werden zu f . Die vorherige GP f_0 kann charakterisiert werden durch eine Mittelwertfunktion $m_0: X \rightarrow \mathbb{R}$ und die Kernelfunktion KF, zum Beispiel eine symmetrische positivendliche Funktion, $k_0: X \times X \rightarrow \mathbb{R}$, zum Beispiel:

$$f_0(x) \sim \mathcal{GP}(m_0(x), k_0(x, x')).$$

[0068] In diesem Fall können für eine endliche Anzahl von Eingaben die entsprechenden Funktionswerte der GP eine gemeinsame Gaußsche Verteilung haben:

$$f_0(\mathbf{X}) = \left[f_0(x^{(1)}), \dots, f_0(x^{(N)}) \right] \sim \mathcal{N}(m_0(\mathbf{X}), k_0(\mathbf{X}, \mathbf{X})),$$

wobei der Mittelwertvektor $m_0(\mathbf{X}) = [m_0(x^{(1)}), \dots, m_0(x^{(N)})]$ und die Kovarianzmatrix $k_0(\mathbf{X}, \mathbf{X}) = k_0(x^{(i)}, x^{(j)})$ für alle $i, j = 1, \dots, N$ durch die Mittelwertfunktion bzw. die Kovarianzfunktion definiert werden. Um Vorhersagen IO an neuen Punkten $\mathbf{IM}, \mathbf{X}_* = (x_*^{(n+1)}, \dots, x_*^{(n+m)})$ zu bestimmen, kann die GP auf die Daten LMi, LOi konditioniert werden, was den hinteren Prozess ergibt

$$f_*(x) \sim \mathcal{GP}(m_*(x), k_*(x, x')).$$

[0069] Insbesondere können aktualisierte Mittelwert- und Kovarianzfunktionen m_* , k_* mit dem entsprechenden Mittelwertvektor $m_*(X_*) = m_0(X_*) - k_0(X_*, X)k_0(X, X)^{-1}(m_0(X) - y)$ und der Kovarianzmatrix $k_*(X_*, Y_*) = k_0(X_*, X_*) - k_0(X_*, X)k_0(X, X)^{-1}k_0(X, X_*)$ erhalten werden.

[0070] Der Wert der Ausgabegröße kann durch einen Linearprädiktor, auch bekannt als ein Kernelprädiktor, gefolgert werden. Insbesondere kann die vorherige GP als Mittelwert Null angenommen werden, wobei in diesem Fall der Wert der Ausgabegröße IO gefolgert werden kann als:

$$m_*(x) = \sum_{n=1}^N c_n k_0(x, x^{(n)}) \quad \forall x \in \mathcal{X},$$

mit $c = [c_1, \dots, c_N]$ und $c = k_0(X, X)^{-1}y$. Die vorherige Kovarianzfunktion kann den Funktionsraum des hinteren Mittelwerts als einen reproduzierenden Kernel-Hilbert-Raum identifizieren, zum Beispiel $m_* \in K = \text{span}\{k_0(\cdot, x), x \in \mathcal{X}\}$.

[0071] Zusätzlich zu dem Folgern des Ausgabewerts IO kann die GP-Regression Reg einen Unsicherheitswert (nicht gezeigt) bestimmen, der eine Unsicherheit des gefolgerten Werts IO angibt, zum Beispiel hinsichtlich einer Varianz oder eine Standardabweichung des gefolgerten Werts IO, oder hinsichtlich einer daraus abgeleiteten diskreten Punktzahl (unsicher/sicher, auf einer Skala von 1-5, usw.). Wie an sich bekannt, kann der Unsicherheitswert von der aktualisierten Kovarianzfunktion abgeleitet werden.

[0072] Fig. 5 zeigt ein detailliertes, jedoch nicht einschränkendes Beispiel dafür, wie eine Gaußsche Prozess-Regression trainiert wird.

[0073] Das Beispiel zeigt Kernel-Funktion KF 550, parametrisiert durch Satz von Parametern PAR 540. Zum Beispiel kann die Kernelfunktion KF wie in Bezug auf Fig. 4 diskutiert sein. Insbesondere kann das Anwenden der Kernelfunktion KF das Anwenden eines trainierten niedrigrangigen Tensor-Netzwerkmodells, wie zum Beispiel eines Tensor-Train-Netzwerks, umfassen, um Merkmalsdarstellungen von zwei jeweiligen Sätzen von Werten der Eingabegrößen zu erhalten, und das Anwenden einer Basis-Kernelfunktion auf die Merkmalsdarstellungen, um eine Ähnlichkeit zu bestimmen. Die Parameter PAR der Kernelfunktion können trainierbare Parameter des niedrigrangigen Tensor-Netzwerkmodells und trainierbare Parameter der Basis-Kernelfunktion (falls vorhanden) beinhalten.

[0074] Dieses Beispiel demonstriert ein Ende-zu-Ende-Training basierend auf einem Trainingsdatensatz von Trainingsinstanzen. Insbesondere zeigt die Figur Werte von Eingabegrößen TIM 520 und einen Wert der Ausgabegröße TOM 539 einer Trainingsinstanz. In diesem Beispiel wird ein Trainingsschritt für eine einzelne Instanz ausgeführt, es ist jedoch auch möglich, ein Ende-zu-Ende-Training auszuführen, wobei die GP-Regression einen Stapel mehrerer Ausgaben folgert.

[0075] Das Training kann das Ausführen der Gaußschen Prozessregression an den physikalischen Eingabegrößen TIM der Trainingsinstanz einbeziehen, um Vorhersage IO 530, der Ausgabegröße zu erhalten. Dies kann wie in Bezug auf Fig. 4 diskutiert durchgeführt werden, insbesondere durch Anwenden der Kernelfunktion KF, um jeweilige Ähnlichkeiten SIM1 581, ..., SIMn 582 der Trainingsinstanz zu jeweiligen gekennzeichneten Instanzen LM1 521, ... LMn 522 gemäß der Kernelfunktion zu berechnen, und durch Folgern des Werts IO 530 der Ausgabegröße von den jeweiligen berechneten Ähnlichkeiten SIMi und jeweiligen Werten LO1 531, ..., LOn 532 der Ausgabegröße der gekennzeichneten Instanzen. Die gekennzeichneten Instanzen können von dem Trainingsdatensatz ausgewählt werden, aber typischer sind sie induzierende gekennzeichnete

nete Instanzen, die als Teil des Trainings optimiert werden, zum Beispiel kann die GP-Regression eine sogenannte spärliche GP-Regression sein.

[0076] Das Ende-zu-Ende-Training kann ferner Trainingsoperation Train 510 beinhalten, in der die Vorhersage IO der Ausgabegröße mit dem bekannten Wert TOM der Ausgabegröße verglichen wird, um ein Trainingssignal zu erhalten, und trainierbare Parameter der Gaußschen Prozessregression werden basierend auf dem Trainingssignal aktualisiert. Insbesondere kann die marginale Wahrscheinlichkeit der GP in Bezug auf die trainierbaren Parameter maximiert werden. Die trainierbaren Parameter können die trainierbaren Parameter PARS der Kernelfunktion KF und die induzierenden Eingaben LMi und Ausgaben Loi, falls verwendet, beinhalten. Insbesondere können die Tensor-Netzwerkparameter als Kernel-Hyperparameter der Kernelfunktion KF behandelt und durch die marginale Wahrscheinlichkeit der GP optimiert werden.

[0077] Zum Beispiel kann die marginale Wahrscheinlichkeit der Ziele y, TOM maximiert werden, abhängig von den Daten X, TIM und den Hyperparametern θ , PARS, mit Bezug auf θ :

$$\log p(\mathbf{y}|\mathbf{X},\theta) \propto -\mathbf{y}^T \mathbf{K}_\theta^{-1} \mathbf{y} - \log|\mathbf{K}_\theta|.$$

[0078] Hier bezeichnet $\mathbf{K}_\theta$ die Kovarianzmatrix der Ziele y, bei gegebenen Hyperparametern θ , zum Beispiel $\mathbf{K}_\theta = k_0(\mathbf{X}, \mathbf{X})$. Der erste Term gibt ein Maß an, wie gut der gewählte Kernel die Daten darstellt. Wie durch den zweiten Term in diesem Beispiel demonstriert, kann ein Regularisierungsterm angewendet werden, der Komplexität benachteiligt. Ein solcher Regularisierungsterm kann eine Überanpassung verhindern, da die marginale Wahrscheinlichkeit naturgemäß Modelle mittlerer Komplexität bevorzugt. Interessanterweise haben die Erfinder herausgefunden, dass die obige Optimierung eine ausreichende Regularisierung bereitstellen kann, wenn ein niedrigrangiges Tensor-Netzwerkmodell verwendet wird, während selbst mit einem solchen Regularisierungsterm eine Kernelfunktion, die ein tiefes neuronales Netzwerk verwendet, keine ausreichende Regularisierung bereitstellen kann.

[0079] Insbesondere kann das Training Train spärliche Variational-Inference (VI) anwenden. Eine Gaußsche Dichte im mittleren Feld kann verwendet werden. Die Erfinder haben herausgefunden, dass es vorteilhaft ist, jeweiligen Parametersätzen individuelle Lernraten zuzuweisen, insbesondere: unterschiedlichen Unterkomponenten des Tensor-Netzwerkmodells, zum Beispiel. Tensor-Train-Kernen; der Basiskernelfunktion, zum Beispiel einer Radial-Basis-Funktion (RBF); und VIbezogenen Parametern, wie zum Beispiel den induzierenden Punkten, Parametern der Variationsdichte, wie zum Beispiel einem Mittelwert und/oder einer Kovarianz usw. Dies kann durch Einstellen entsprechender anfänglicher Lernraten erfolgen, zum Beispiel durch Spezifizieren von Parametergruppen in der Optimiererklasse der PyTorch-Software. Das Training Train kann gradientenbasiert sein, zum Beispiel durch Verwenden eines stochastischen Gradientenabstiegs. Zum Beispiel kann der Adam-Optimierer verwendet werden, wie offenbart in Kingma und Ba, „Adam: A Method for Stochastic Optimization“ (verfügbar unter <https://arxiv.org/abs/1412.6980> und hierin durch Bezugnahme aufgenommen). Wie bekannt ist, können solche Optimierungsverfahren heuristisch sein, und/oder zu einem lokalen Optimum gelangen. Das Training kann auf einer Instanz-für-Instanz-Basis oder in Stapeln, zum Beispiel von höchstens oder mindestens 64 oder höchstens oder mindestens 256 Instanzen, ausgeführt werden.

[0080] Das Training Train kann ferner das Ausführen einer Hyperparameter-Optimierung der Gaußschen Prozessregression umfassen. Die Hyperparameter, die durch diese Hyperparameteroptimierung optimiert werden, sind nicht die oben beschriebenen trainierbaren Parameter, sie beinhalten zum Beispiel nicht die trainierbaren Parameter des niedrigrangigen Tensor-Netzwerkmodells oder der Basis-Kernelfunktion oder die induzierenden Punkte selbst, sondern sie sind Hyperparameter der GP-Regression, zum Beispiel die Anzahl von induzierenden Punkten und/oder Größenparametern des niedrigrangigen Tensor-Netzwerkmodells, die seine Anzahl trainierbarer Parameter definieren. Als methodisches Werkzeug zur Hyperparameter-Optimierung kann die Zufallssuche, zum Beispiel zusammen mit Advanced-Early-Stopping, verwendet werden.

[0081] Zum Beispiel kann ein Datensatz in einen Trainingsdatensatz, einen Validierungsdatensatz und/oder einen Testdatensatz unterteilt werden. Durch Verwenden des Trainingssatzes in Verbindung mit Early-Stopping des Validierungssatzes kann ein optimaler Funktionsapproximator abhängig von Hyperparametern $\lambda \in \Lambda$ erhalten werden:

$$f^*(\lambda) \in \arg \min_{f \in \mathcal{M}} \int \chi^\ell(f(\lambda); \mathbf{x}) \rho(d\mathbf{x}),$$

für eine Modellklasse M und Verlustfunktion -e. Anschließend können optimale Hyperparameter

$$\lambda^* \in \arg \min_{\lambda \in \Lambda} \int \mathcal{X}^\ell(f^*(\lambda); x) \rho(dx),$$

erhalten werden, indem das Modell mit der besten Bewertung auf dem Validierungssatz von einer Zufallssuche ausgewählt wird, zum Beispiel wenn sich eine Zufallssuche mit 100 Proben über den potenziell hochdimensionalen Hyperparameter-Raum A als erfolgreich herausgestellt hat. Das Risiko des Modells kann geschätzt werden als:

$$L(f^*) \int \mathcal{X}^\ell(f^*(\lambda^*); x) \rho(dx) = \mathbb{E}_\rho [\ell(f^*(\lambda^*); \cdot)]$$

wobei der resultierende Mittelwert als Leistungsindikator verwendet wird.

[0082] Interessanterweise ist die Zufallssuche weniger anfällig für den Fluch der Dimensionalität als Alternativen, wie zum Beispiel die Rastersuche. Darüber hinaus weist sie die vorteilhafte Eigenschaft auf, dass sie dazu in der Lage ist, eine einigermaßen gleichmäßige Abdeckung jedes Unterraums von A bereitzustellen. Dies ist besonders wichtig, da die meisten Funktionsapproximatoren nicht gleich stark von allen Hyperparametern abhängen, was bedeutet, dass einige Hyperparameter wichtiger sind als andere, und daher diese Unterräume von A sorgfältiger abgedeckt werden sollten. Obwohl eine Überanpassung ein Risiko für diese Art der Hyperparameter-Optimierung sein kann, stellt sich heraus, dass dies durch den Vorteil aufgewogen werden kann, dass sie viel berechnungseffizienter ist als andere Techniken wie die verschachtelte Kreuzvalidierung.

[0083] Fig. 6a zeigt ein detailliertes, jedoch nicht beschränkendes Beispiel trainierbarer Parameter PARS 640, eines niedrigrangigen Tensor-Netzwerkmodells. Zum Beispiel kann dieses Beispiel verwendet werden, um das niedrigrangige Tensor-Netzwerkmodell LRT 460 von Fig. 4 zu implementieren.

[0084] Das niedrigrangige Tensor-Netzwerkmodell in diesem Beispiel ist ein Tensor-Train (TT). Ein Tensor-Train kann als ein niedrigrangiges Tensor-Netzwerkmodell definiert werden, in dem der Gewichts-Tensor in entsprechende Unterkomponenten für die jeweiligen Dimensionen der Eingabe zerlegt wird: einen Tensor zweiter Ordnung für die erste und letzte Dimension, und Tensoren dritter Ordnung für die verbleibenden Dimensionen. Zu Zwecken der Veranschaulichung zeigt die Figur Gewichts-Tensor vierter Ordnung W_a 661, der in vier Unterkomponenten zerlegt ist: Tensor zweiter Ordnung $V(1)$ 641 für die erste Dimension; Tensor dritter Ordnung $V(2)$ 642, für die zweite Dimension; Tensor dritter Ordnung $V(3)$ 643, für die dritte Dimension, und Tensor zweiter Ordnung $V(4)$ 644, für die vierte Dimension. Ein Gewicht des Gewichts-Tensors W_a , indiziert durch Indizes für die jeweiligen Dimensionen, kann als eine Summe von Produkten von Elementen der jeweiligen Unterkomponenten-Tensoren definiert werden. Ein Index des Unterkomponenten-Tensors kann dem Index des Gewichts-Tensors entsprechen; die Summation kann sich über die verbleibenden Indizes summieren. Die oberen Summationsgrenzen für die verbleibenden Indizes können als die Ränge des Tensor-Train bezeichnet werden. Zum Beispiel kann ein Rang auf höchstens oder mindestens 2, höchstens oder mindestens 15, oder höchstens oder mindestens 30 gesetzt werden. Die Ränge können einheitlich auf denselben Wert gesetzt werden.

[0085] Mathematisch kann eine niedrigrangige Tensor-Train-Kompression des Koeffizienten-Tensors W , W_a bezeichnet werden als

$$W_{\alpha_1, \dots, \alpha_d} \approx \sum_{k_0=1}^{r_0} \dots \sum_{k_d=1}^{r_d} \prod_{i=1}^d V_{k_{i-1}, \alpha_i, k_i}^{(i)}$$

mit $r_0 = r_d = 1$. Hier sind die Komponenten $V^{(i)}$, $i = 2, \dots, d-1$ Tensoren dritter Ordnung. Die verbleibenden Komponenten $V^{(1)}$, $V^{(d)}$ sind zweiter Ordnung. Die Tensoren $V^{(1)}, \dots, V^{(d)}$ können trainierbare Parameter des Tensor-Train sein. Die oberen Summationsgrenzen r_i , $i = 1, \dots, d-1$ sind die TT-Ränge, die als Hyperparameter der GP-Regression trainiert werden können.

[0086] Der Approximationsfehler zwischen dem wahren Tensor und seiner TT-Form kann durch eine Singulärwertzerlegung höherer Ordnung bestimmt werden. Diese Singulärwertzerlegung kann auch verwendet werden, um die TT-Darstellung zu konstruieren. Durch Schwellenwertbildung der singulären Werte mit einer vorgeschriebenen Genauigkeit kann der Fehler kontrolliert und die Ränge begrenzt werden.

[0087] Interessanterweise hat das TT-Format eine Speicherkomplexität, mit anderen Worten, eine Anzahl trainierbarer Parameter, die in der Anzahl der Merkmale x_i linear skaliert, und somit wesentlich effizienter ist, als das Speichern und Verarbeiten separater Gewichte des Gewichts-Tensors, insbesondere

beim Vermeiden des Fluchs der Dimensionalität. Insbesondere kann eine Speicherkomplexität von $O(\max(J_1, \dots, J_d) \cdot d \cdot \max(r_1, \dots, r_{d-1})^2)$ erreicht werden. Interessanterweise ermöglicht dies, praktische Berechnungen in einem sehr hochdimensionalen Raum auszuführen, und gleichzeitig trainierbar zu bleiben.

[0088] Effektiv kann das Auferlegen einer niedrigrangigen Struktur auf den Koeffizienten-Tensor W in einer eingebetteten Mannigfaltigkeit von Funktionen mit niedrigrangigen Tensoren vom Rang höchstens $r = (r_1, \dots, r_{d-1})$ in F resultieren, somit ergibt sich

$$\mathcal{M}_r := \{f \in \mathcal{F}(\mathcal{X}) \mid f \text{ with rank-}r \text{ TT coefficient in } (*)\},$$

wobei sich $(*)$ auf den allgemeinen Ausdruck für f als eine Linearkombination von Produkten von Basisfunktionen bezieht, die durch den Gewichts-Tensor $f(x) = \sum_{\alpha_1=1}^{J_1} \dots \sum_{\alpha_d=1}^{J_d} W_{\alpha_1, \dots, \alpha_d} \prod_{i=1}^d P_{\alpha_i}(x_i)$ definiert werden. Effektiv können die Tensor-Train-Darstellung oder andere niedrigrangige Darstellungen falsche Merkmale vernachlässigen, indem sie die Rangparameter $r_1, \dots, r_{d-1}$ in dem Sinne einer impliziten Merkmalsauswahl steuern. Analytische Grenzen, die dieses Verhalten rigoros quantifizieren, sind bekannt.

[0089] Durch Verwenden eines TT-Formats für den Koeffizienten-Tensor W kann der folgende Ausdruck verwendet werden, um ein Merkmal einer Merkmalsdarstellungsausgabe durch das Tensor-Netzwerk zu bestimmen:

$$f(x) = \sum_{\alpha \in A} \sum_{k_0=1}^{r_0} \dots \sum_{k_d=1}^{r_d} \prod_{i=1}^d V_{k_{i-1}, \alpha_i, k_i}^{(i)} P_{\alpha_i}(x_i).$$

[0090] **Fig. 6b** zeigt ein detailliertes, jedoch nicht einschränkendes Beispiel dafür, wie niedrigrangiges Tensor-Netzwerkmodell LRT 660, zum Beispiel ein Tensor-Train-Netzwerk, trainiert wird. Dieses Beispiel kann verwendet werden, um ein Vortraining des niedrigrangigen Tensor-Netzwerkmodells auszuführen, bevor das Ende-zu-Ende-Training der GP-Regression, wie zum Beispiel in Bezug auf **Fig. 5** diskutiert, ausgeführt wird.

[0091] Wie auch in Bezug auf **Fig. 4** diskutiert, kann das niedrigrangige Tensor-Netzwerkmodell LRT dazu ausgelegt sein, eine Merkmalsdarstellung zu bestimmen, die mehrere Merkmale umfasst. Das Ende-zu-Ende-Training kann das Tensor-Netzwerkmodell LRT auf eine solche Weise optimieren, dass es diese Merkmale auf eine Weise bestimmt, die für die GP-Regression optimal ist.

[0092] Interessanterweise kann ein Vortraining des Tensor-Netzwerkmodells ausgeführt werden, bei dem das Modell trainiert werden kann, um die Ausgabegröße zu folgern, die auch die GP-Regression folgert. Die Figur zeigt nämlich eine gekennzeichnete Instanz, die Trainingseingabemessungen TIM 620 und entsprechenden Trainingsausgabemengenwert TOM 639 umfasst. Wie in der Figur gezeigt, kann das niedrigrangige Tensor-Netzwerkmodell LRT in Trainingsoperation Train 610 trainiert werden, um die Ausgabegröße TOM von den Eingabegrößen TIM zu folgern. Zum Beispiel kann das Tensor-Netzwerkmodell LRT auf die Trainingseingabemessungen TIM angewendet werden, um gefolgerten Wert IO 630 für die Ausgabegröße und Parameter PAR 640 des niedrigrangigen Tensor-Netzwerkmodells LRT 660 zu erhalten, kann optimiert werden, um eine Differenz zwischen dem Trainingsausgabemengenwert TOM und dem gefolgerten Ausgabewert IO zu minimieren.

[0093] Somit kann das Vortraining in einem Tensor-Netzwerkmodell LRT resultieren, das eine einzelne Ausgabe bereitstellt, die sich der Ausgabegröße TOM approximiert. Dieses Modell kann erweitert werden, um dieselbe Ausgabe mehrmals bereitzustellen, entsprechend der gewünschten Größe der zukünftigen Darstellung, die durch das Tensor-Netzwerk extrahiert wird. Zum Beispiel kann ein Teil des Modells ersetzt werden, und/oder ein Untersatz der Parameter kann über Ausgabewerte hinweg geteilt werden. Zum Beispiel kann die latente Dimension des Modells von Vortraining zu Ende-zu-Ende-Training angepasst werden, indem der erste übereinstimmende Kern des Modells dupliziert wird. Das resultierende Tensor-Netzwerkmodell kann dann als das Tensor-Netzwerkmodell zum Beispiel von **Fig. 4** und/oder **Fig. 5** verwendet werden. Durch das Verwenden des Vortrainings werden verbesserte Anfangswerte für das anschließende Ende-zu-Ende-Training der GP-Regression erhalten.

[0094] Insbesondere veranschaulicht diese Figur den Fall, in dem das niedrigrangige Tensor-Netzwerkmodell ein Baum-Tensor-Netzwerk umfasst. In einem solchen Baum-Tensor-Netzwerk kann der Gewichts-Tensor W 661, der verwendet wird, um Linearkombination LC 663 von Produkten von Basisfunktionen P 662 zu berechnen, in mehrere Unterkomponenten zerlegt werden. Als eine Veranschaulichung zeigt die Figur

Unterkomponenten, die die Tensoren $V(1)$ 641, $V(2)$ 642, $V(3)$ 643, $V(4)$ 644 eines Tensor-Train-Modells darstellen.

[0095] Das Trainieren eines solchen Baum-Tensor-Netzwerks mit mehreren Unterkomponenten kann durch wiederholtes Trainieren einer Unterkomponente $V(i)$ des Baum-Tensor-Netzwerks ausgeführt werden, während andere Unterkomponenten $V(j)$ des Baum-Tensor-Netzwerks fixiert bleiben. Interessanterweise kann dies die Verwendung effizienter Optimierungstechniken ermöglichen, die die spezifische, einfachere Struktur der Unterkomponente verwenden, und die nicht zum Trainieren des Modells als Ganzes verwendet werden können, zum Beispiel kann ein stochastischer Gradientenabstieg vermieden werden. Insbesondere kann für das Training ein alternierendes lineares Schema verwendet werden, das optional regularisiert werden kann. Das Training der Unterkomponente kann einem linearen Kleinste-Quadrate-Problem entsprechen, das in ein lineares System umgeschrieben und zum Beispiel durch Verwenden einer LU-Faktorisierung, einer Krylov-Unterraummethode oder ähnlichem gelöst werden kann.

[0096] Insbesondere in dem Zusammenhang einer regulierten empirischen Risikominimierung kann ein Vortraining durch Verwenden eines alternierenden linearen Schemas wie folgt ausgeführt werden. Das Problem des Trainierens des Tensor-Baum-Modells, um den Ausgabewert von den Eingabemessungen zu folgern, kann als ein Risikominimierungsproblem wie folgt formuliert werden:

$$\min_{f \in \mathcal{M}_r} L(f) \min_{f \in \mathcal{M}_r} \int \mathcal{X}^\ell(f; x) \rho(dx),$$

wobei $\ell(f; x) = (\Phi(x) - f(x))^2$. Das empirische Risiko kann für einen Satz von Realisierungen $(x^{(n)}, y^{(n)} = \Phi(x^{(n)}))$ für $n = 1, \dots, N$ wie folgt berechnet werden:

$$L(f) \approx \frac{1}{N} \sum_{n=1}^N \left(y^{(n)} - f(x^{(n)}) \right)^2.$$

[0097] Optional kann eine Regularisierung verwendet werden, zum Beispiel um das folgende zu minimierende Ziel zu erhalten:

$$\hat{L}(f) = \sum_{n=1}^N \left(y^{(n)} - f(x^{(n)}) \right)^2 + \delta \|f\|_{\mathcal{F}}^2,$$

wobei δ ein Hyperparameter ist, und der Typ der Regularisierung von dem Funktionsraum $\mathcal{F}$ abhängig ist.

[0098] In Baum-Tensor-Netzwerken, wie zum Beispiel Tensor-Trains, kann eine Funktion $f \in \hat{\mathcal{F}}_A$ durch eine Liste von Tensoren $(V^{(i)})_{i=1}^d$ parametrisiert werden. Von der Tensor-Struktur und Parsevals-Identität kann

abgeleitet werden, dass $\left\| f\left(; \left(V^{(i)} \right)_i \right) \right\|_{\mathcal{F}}^2 = \left\| \left(V^{(i)} \right)_i \right\|_{\mathcal{F}}^2$ wobei $\left\| \left(V^{(i)} \right)_i \right\|_{\mathcal{F}}$ die Frobenius-Norm des vollen Tensors bezeichnet, der $(V^i)_i$ entspricht. Dementsprechend kann das folgende endlichdimensionale Minimierungsproblem erhalten werden:

$$\min_{(V^{(i)})_i} \hat{L}\left(\left(V^{(i)} \right)_i \right) = \min_{(V^{(i)})_i} \frac{1}{N} \sum_{n=1}^N \left(y^{(n)} - f\left(x^{(n)} ; \left(V^{(i)} \right)_i \right) \right)^2 + \delta \left\| \left(V^{(i)} \right)_i \right\|_{\mathcal{F}}^2,$$

über alle Unterkomponenten-Tensoren $V^{(i)}$. Diese multilineare Modellstruktur kann ausgenutzt werden, indem ein alternierendes Optimierungsverfahren verwendet wird, bei dem alle bis auf einen Unterkomponenten-Tensor $V^{(i)}$ fixiert sind, und für das ein Unterkomponenten-Tensor $V^{(i)}$ optimiert wird, während über den Index i iteriert wird:

$$\min_{V^{(j)}} \frac{1}{N} \sum_{n=1}^N \left(y^{(n)} - f\left(x^{(n)} ; \left(V^{(i)} \right)_i \right) \right)^2 + \delta \left\| V^{(j)} \right\|_{\mathcal{F}}^2,$$

wobei $V^{(i)}$ für alle $i \neq j$ fixiert ist. Ein solches alternierendes Training ist für sich als ein Alternating-Linear-Scheme (ALS) bekannt. Durch Verwenden von ALS kann das nichtkonvexe Lernproblem effektiv dahingehend gelockert werden, dass eine Folge von linearen Systemen gelöst wird. Interessanterweise kann die

Regularisierung bei dieser Technik durch geeignete Wahl der orthonormalen Basisfunktionen und Poenalisierung der Frobenius-Norm auf natürliche Weise ausgeführt werden.

[0099] Unten werden mehrere Implementierungsoptionen diskutiert, die bei einer Implementierung der bereitgestellten Techniken verwendet werden können. Als Budgets für Vortraining, Hyperparameter-Optimierung und Testsatzbewertung können 20, 50 bzw. 100 Epochen verwendet werden. Early-Stopping an der Bewertungseinstellung kann verwendet werden.

[0100] Die Folgenden sind mögliche Hyperparameter der bereitgestellten Trainingstechniken und beispielhafte Verteilungen, von denen diese Hyperparameter ausgewählt oder bemustert werden können (zum Beispiel bei einer Hyperparameter-Optimierung):

- Tensor-Train-Ränge: $r \sim U(2, 15)$,
- L^2 Polynomgrad und Anzahl der Basisfunktionen: $J \sim U(2, 14)$,
- ALS Regularisierungskoeffizient: $\delta_1 = 10^b$, $b \sim U(\log(e - 10), \log(e - 1))$
- Orthogonalisierung des TT nach Vortraining mit gleicher Wahrscheinlichkeit
- Dimensionalität der latenten Merkmalsdarstellung: $L \sim U(1, d)$
- Anzahl der induzierenden Punkte: $M \sim U(10, 1000)$
- TT-Regularisierung während des Ende-zu-Ende-Trainings mit gleicher Wahrscheinlichkeit
- TT-Regularisierungskoeffizient während des Ende-zu-Ende-Trainings: $\delta_2 = 10^b$, $b \sim U(\log(e - 10), \log(e - 1))$
- anfängliche TT-Lernrate: $\eta_1 = 10^b$, $b \sim U(\log(e - 5), \log(e - 1))$
- anfängliche RBF-Basiskernel-Hyperparameter-Lernrate: $\eta_2 = 10^b$, $b \sim U(\log(e - 4), \log(e - 1))$
- anfängliche VI-bezogene Hyperparameter-Lernrate: $\eta_3 = 10^b$, $b \sim U(\log(e - 3), \log(e - 1))$
- Datenstapelgröße: $S \sim U(4, 1024)$

[0101] Die Erfinder verglichen die bereitgestellten Techniken, die unten als Tensor-Kernel-Learning (TKL) bezeichnet werden, mit mehreren Ausgangsbasen. Die erste Ausgangsbasis ist ein Deep-Learning-Modell (DNN), genauer gesagt ein ResNet vor der Aktivierung. Die zweite Ausgangsbasis ist eine hochdimensionale Tensor-Regression durch Verwenden der TT-Zerlegung. Die dritte Ausgangsbasis ist ein reguläres GP-Modell. Die vierte Ausgangsbasis ist eine GP-Regression durch Verwenden von Deep-Kernel-Learning (DKL) durch Verwenden des ResNet vor der Aktivierung. Experimente wurden mit drei synthetischen und drei realen Datensätzen ausgeführt, nämlich den physikalisch-chemischen Eigenschaften der Protein-Tertiärstruktur, dem KEGG-Metabolic-Relation-Network (gerichtet) und den Kin40K-Datensätzen.

[0102] Es stellte sich heraus, dass die bereitgestellten Techniken alle Ausgangsbasen in jedem der realen Datensätze um ca. eine Größenordnung übertreffen. Bei den synthetischen Datensätzen übertrifft TKL die GP-, DNN- und DKL-Modelle, während nur die TT-Regression bessere Ergebnisse erreicht. Dies ist vermutlich auf die Struktur der synthetischen Datensätze zurückzuführen, die nicht auf reale Daten übertragen werden kann. Verglichen mit einer regulären GP stellte sich heraus, dass die Verwendung eines niedrigrangigen Tensor-Netzwerkmodells insbesondere bei den höherdimensionalen Datensätzen Verbesserungen bereitstellt. Die bessere praktische Leistung von TKL verglichen mit DKL beweist seine besseren Regularisierungseigenschaften und das daraus resultierende geringere Risiko einer Überanpassung. Es ist auch anzumerken, dass das TKL-Modell verglichen mit dem Verwenden eines vergleichbaren Merkmalsextraktors für neuronale Netzwerke in DKL eine viel geringere Anzahl trainierbarer Parameter hat.

[0103] Hinsichtlich der Konvergenz von TKL auf die synthetischen Datensätze bezüglich der Anzahl von Datenpunkten ist ein konsistenter Abwärtstrend des mittleren quadratischen Fehlers für alle drei Datensätze zu beobachten. Bei einem Datensatz ist die Konvergenz langsamer, was auf die Verwendung der GP-Regression zurückzuführen sein kann, die unter dem Fluch der Dimensionalität leidet.

[0104] Hinsichtlich des mittleren quadratischen Fehlers zeigen Ablationsexperimente, dass das Verwenden von Trainingsmethoden, die für die Struktur von in dem Vortraining verwendeten niedrigrangigen Funktionen spezifisch sind, zum Beispiel durch Verwenden des alternierenden linearen Schemas, einen großen Vorteil

bereitstellt. Eine kernspezifische Gradientenoptimierung durch Verwenden separater Lernraten ist ebenfalls von großem Vorteil, insbesondere für niedrigerdimensionale Daten.

[0105] Fig. 7 zeigt ein Blockdiagramm von computerimplementiertem Steuerungsverfahren 700 zum Steuern eines computergesteuerten Systems, das mit einer Umgebung interagiert. Das Verfahren 700 kann einem Betrieb des Systems 200 von Fig. 2 oder Fig. 3 entsprechen. Dies ist jedoch keine Einschränkung, insofern, als das Verfahren 700 auch durch Verwenden eines anderen Systems, einer anderen Vorrichtung oder eines anderen Geräts ausgeführt werden kann.

[0106] Das Verfahren 700 kann in einer Operation mit dem Titel „ZUGREIFEN AUF REGRESSIONSDATEN“ das Zugreifen 710 auf Regressionsdaten zum Ausführen einer Gaußschen Prozessregression umfassen. Die Gaußsche Prozessregression kann dazu dienen, einen Wert einer Ausgabegröße des computergesteuerten Systems und/oder seiner Umgebung basierend auf Messungen einer oder mehrerer physikalischer Eingabegrößen des computergesteuerten Systems und/oder seiner Umgebung zu folgern. Die Regressionsdaten können Kernel-Hyperparameter einer Kernelfunktion und mehrere gekennzeichnete Instanzen umfassen. Eine gekennzeichnete Instanz kann Werte der Eingabe- und Ausgabegrößen umfassen.

[0107] Das Verfahren 700 kann in einer Operation mit dem Titel „ERHALTEN VON MESSUNGEN“ das Erhalten 720 von Messungen der einen oder mehreren physikalischen Eingabegrößen umfassen.

[0108] Das Verfahren 700 kann in einer Operation mit dem Titel „AUSFÜHREN EINER GP-REGRESSION“ das Ausführen 730 der Gaußschen Prozessregression umfassen, um einen Wert der Ausgabegröße basierend auf den erhaltenen Messwerten zu folgern. Die GP-Regression 730 kann in einer Operation mit dem Titel „ANWENDEN EINER KERNELFUNKTION EINSCHLIESSLICH EINES NIEDRIGRANGIGEN TENSOR-NETZWERKMODELLS“ das Anwenden 732 der Kernel-Funktion umfassen, um jeweilige Ähnlichkeiten der erhaltenen Messungen mit den jeweiligen gekennzeichneten Instanzen zu berechnen. Das Anwenden der Kernelfunktion kann das Anwenden eines trainierten niedrigrangigen Tensor-Netzwerkmodells auf die Messungen der einen oder mehreren physikalischen Eingabegrößen und auf Werte der einen oder mehreren physikalischen Eingabegrößen einer gekennzeichneten Instanz umfassen, um Merkmalsdarstellungen der Messungen und der gekennzeichneten Instanz zu erhalten. Das Anwenden der Kernelfunktion kann ferner das Anwenden einer Basis-Kernelfunktion auf die Merkmalsdarstellungen umfassen, um Ähnlichkeiten der erhaltenen Messungen mit den gekennzeichneten Instanzen zu erhalten. Die GP-Regression 730 kann ferner in einer Operation mit dem Titel „FOLGERN EINER AUSGABE“ das Folgern 734 des Werts der Ausgabegröße von den jeweiligen berechneten Ähnlichkeiten und jeweiligen Werten der Ausgabegröße der gekennzeichneten Instanzen umfassen.

[0109] Das Verfahren 700 kann in einer Operation mit dem Titel „ABLEITEN VON STEUERUNGSDATEN“ das Ableiten 740 von Steuerungsdaten basierend auf dem Wert der Ausgabegröße umfassen. Das Verfahren 700 kann in einer Operation mit dem Titel „STEUERN DES SYSTEMS“ das Steuern 750 des computergesteuerten Systems gemäß den abgeleiteten Steuerungsdaten umfassen.

[0110] Fig. 8 zeigt ein Blockdiagramm von computerimplementiertem Trainingsverfahren 800 zum Trainieren einer Gaußschen Prozessregression zur Verwendung beim Steuern eines computergesteuerten Systems, wie hierin zum Beispiel gemäß dem Verfahren 700 von Fig. 7 beschrieben. Das Verfahren 800 kann einem Betrieb des Systems 100 von Fig. 1 entsprechen. Dies ist jedoch keine Einschränkung, da das Verfahren 800 auch durch Verwenden eines anderen Systems, einer anderen Vorrichtung oder eines anderen Geräts ausgeführt werden kann.

[0111] Das Verfahren 800 kann in einer Operation mit dem Titel „ZUGREIFEN AUF TRAININGSDATEN“ das Zugreifen 810 auf einen Trainingsdatensatz von Trainingsinstanzen umfassen. Eine Trainingsinstanz des Trainingsdatensatzes kann Werte der Eingabe- und Ausgabegrößen umfassen.

[0112] Das Verfahren 800 kann in einer Operation mit dem Titel „ENDE-ZU-ENDE-TRAINIEREN“ das Ausführen 820 eines Ende-zu-Ende-Trainings der Gaußschen Prozessregression umfassen. Das Ende-zu-Ende-Training 820 kann in einer Operation mit dem Titel „GP-REGRESSION“ das Ausführen 830 der Gaußschen Prozessregression an den physikalischen Eingabegrößen der Trainingsinstanz umfassen, um eine Vorhersage der Ausgabegröße zu erhalten. Das Ende-zu-Ende-Training 820 kann in einer Operation mit dem Titel „VERGLEICHEN DER ERGEBNISSE“ das Vergleichen 840 der Vorhersage der Ausgabegröße mit dem Wert der Ausgabegröße umfassen, um ein Trainingssignal zu erhalten. Das Ende-zu-Ende-Training 820

kann in einer Operation mit dem Titel „AKTUALISIEREN DER PARAMETER“ das Aktualisieren 850 trainierbarer Parameter der Gaußschen Prozessregression basierend auf dem Trainingssignal umfassen.

[0113] Es versteht sich, dass im Allgemeinen die Operationen des Verfahrens 700 von **Fig. 7** und des Verfahrens 800 von **Fig. 8** in jeglicher geeigneten Reihenfolge ausgeführt werden können, zum Beispiel nacheinander, gleichzeitig oder in einer Kombination davon, vorbehaltlich dessen, dass, wo anwendbar, eine bestimmte Reihenfolge, zum Beispiel durch Eingabe/Ausgabe-Beziehungen notwendig ist. Einige oder alle der Verfahren können auch kombiniert werden, zum Beispiel kann das Verfahren 700 zum Steuern eines Systems durch Verwenden der GP-Regression anschließend darauf angewendet werden, dass dieses GP-Regressionsmodell gemäß dem Verfahren 800 trainiert wird.

[0114] Das/die Verfahren kann/können auf einem Computer als ein computerimplementiertes Verfahren, als fest zugeordnete Hardware oder als eine Kombination von beiden implementiert werden. Wie auch in **Fig. 9** veranschaulicht, können Anweisungen für den Computer, zum Beispiel ausführbarer Code, auf computerlesbarem Medium 900 gespeichert sein, zum Beispiel in der Form von Serie 910 von maschinenlesbaren physikalischen Markierungen, und/oder als eine Serie von Elementen, die unterschiedliche elektrische, zum Beispiel magnetische, oder optische Eigenschaften oder Werte haben. Das Medium 900 kann flüchtig oder nichtflüchtig sein. Beispiele für computerlesbare Medien beinhalten Speicherungsgeräte, optische Speicherungsgeräte, integrierte Schaltungen, Server, Online-Software, usw. **Fig. 9** zeigt eine optische Platte 900. Alternativ kann das computerlesbare Medium 900 Regressionsdaten 910 zum Ausführen einer Gaußschen Prozessregression wie hierin beschrieben umfassen.

[0115] Beispiele, Ausführungsformen oder optionale Merkmale, ob als nicht einschränkend angegeben oder nicht, sind nicht als die beanspruchte Erfindung einschränkend zu verstehen.

[0116] Es ist zu beachten, dass die oben erwähnten Ausführungsformen die Erfindung eher veranschaulichen als einschränken, und dass Fachleute in der Lage sein werden, viele alternative Ausführungsbeispiele zu entwerfen, ohne von dem Umfang der beigefügten Ansprüche abzuweichen. In den Ansprüchen sollen jegliche zwischen Klammern platzierte Bezugszeichen nicht als den Anspruch einschränkend ausgelegt werden. Die Verwendung des Verbs „umfassen“ und seiner Konjugationen schließt das Vorhandensein anderer als der in einem Anspruch angegebenen Elemente oder Phasen nicht aus. Der Artikel „ein“ oder „eine“ vor einem Element schließt das Vorhandensein mehrerer solcher Elemente nicht aus. Ausdrücke, wie zum Beispiel „mindestens eines von“, wenn sie einer Liste oder Gruppe von Elementen vorangestellt werden, stellen eine Auswahl aller oder jeglicher Teilmenge von Elementen von der Liste oder Gruppe dar. Zum Beispiel sollte der Ausdruck „mindestens eines von A, B und C“ so verstanden werden, dass er nur A, nur B, nur C, sowohl A als auch B, sowohl A als auch C, sowohl B als auch C oder alle von A, B und C beinhaltet. Die Erfindung kann mittels Hardware, die mehrere verschiedene Elemente umfasst, und mittels eines geeignet programmierten Computers implementiert werden. In dem einige Mittel aufzählenden Geräteanspruch können einige dieser Mittel durch ein und dasselbe Hardware-Exemplar verkörpert werden. Die bloße Tatsache, dass bestimmte Maßnahmen in voneinander unterschiedlichen abhängigen Ansprüchen genannt werden, gibt nicht an, dass eine Kombination dieser Maßnahmen nicht vorteilhaft verwendet werden kann.

Patentansprüche

1. Computerimplementiertes Steuerungsverfahren (700) zum Steuern eines computergesteuerten Systems, das mit einer Umgebung interagiert, wobei das Verfahren Folgendes umfasst:

- Zugreifen (710) auf Regressionsdaten zum Ausführen einer Gaußschen Prozessregression, wobei die Gaußsche Prozessregression dazu dient, einen Wert einer Ausgabegröße des computergesteuerten Systems und/oder seiner Umgebung basierend auf Messungen einer oder mehrerer physikalischer Eingabegrößen des computergesteuerten Systems und/oder seiner Umgebung zu folgern, wobei die Regressionsdaten Kernel-Hyperparameter einer Kernelfunktion und mehrere gekennzeichnete Instanzen umfassen, wobei eine gekennzeichnete Instanz Werte der Eingabe- und Ausgabegrößen umfasst;
- Erhalten (720) von Messungen der einen oder mehreren physikalischen Eingabegrößen;
- Ausführen (730) der Gaußschen Prozessregression, um einen Wert der Ausgabegröße basierend auf den erhaltenen Messungen zu folgern.
- Anwenden (732) der Kernelfunktion, um jeweilige Ähnlichkeiten der erhaltenen Messungen mit den jeweiligen gekennzeichneten Instanzen zu berechnen, umfassend Anwenden eines trainierten Tensor-Train-Netzwerks auf die Messungen der einen oder mehreren physikalischen Eingabegrößen und auf Werte der einen oder mehreren physikalischen Eingabegrößen einer gekennzeichneten Instanz, um Merkmalsdarstellungen der Messungen und der gekennzeichneten Instanz zu erhalten, und Anwenden einer Basis-Kernel-

funktion auf die Merkmalsdarstellungen, um Ähnlichkeiten der erhaltenen Messungen mit den gekennzeichneten Instanzen zu erhalten;

- Folgern (734) des Werts der Ausgabegröße von den jeweiligen berechneten Ähnlichkeiten und jeweiligen Werten der Ausgabegröße der gekennzeichneten Instanzen;
- Ableiten (740) von Steuerungsdaten basierend auf dem Wert der Ausgabegröße, und Steuern (750) des computergesteuerten Systems gemäß den abgeleiteten Steuerungsdaten.

2. Verfahren (700) nach einem der vorhergehenden Ansprüche, wobei das Anwenden des Tensor-Train-Netzwerks das Berechnen eines Merkmals einer Merkmalsdarstellung durch Berechnen einer Linearkombination von Produkten jeweiliger Basisfunktionen umfasst, die auf jeweilige physikalische Eingabegrößen angewendet werden, und wobei die Anzahl trainierbarer Parameter des Tensor-Train-Netzwerks kleiner ist als die Anzahl der Summanden der Linearkombination.

3. Verfahren (700) nach Anspruch 2, das das Anwenden von Basisfunktionen umfasst, die von einer L2-Orthonom-Polynombasis, einer Fourier-Basis oder einer trigonometrischen Basis ausgewählt wurden.

4. Verfahren (700) nach einem der vorhergehenden Ansprüche, wobei das Verfahren ferner das Bestimmen eines Unsicherheitswerts umfasst, der eine Unsicherheit des gefolgerten Werts angibt, und wobei die Steuerungsdaten ferner basierend auf dem Unsicherheitswert abgeleitet werden.

5. Verfahren (700) nach einem der vorhergehenden Ansprüche, das das Folgern eines Fortschritts eines Fertigungsroboters und das Steuern des Fertigungsroboters basierend auf dem gefolgerten Fortschritt umfasst.

6. Verfahren (700) nach einem der Ansprüche 1-5, das das Folgern einer vorhergesagten Zeit bis zum Ausfall eines Motors, eines Robotergelenks oder einer Montagelinie umfasst.

7. Computerimplementiertes Trainingsverfahren (800) zum Trainieren einer Gaußschen Prozessregression zur Verwendung beim Steuern eines computergesteuerten Systems nach einem der vorhergehenden Ansprüche, wobei das Verfahren Folgendes umfasst:

- Zugreifen (810) auf einen Trainingsdatensatz von Trainingsinstanzen, wobei eine Trainingsinstanz des Trainingsdatensatzes Werte der Eingabe- und Ausgabegrößen umfasst; und
- Ausführen (820) eines Ende-zu-Ende-Trainings der Gaußschen Prozessregression durch: Ausführen (830) der Gaußschen Prozessregression an den physikalischen Eingabegrößen der Trainingsinstanz, um eine Vorhersage der Ausgabegröße zu erhalten, durch Verwenden eines Tensor-Train-Netzwerks als einen Merkmalsextrahierer für eine Basis-Kernelfunktion; Vergleichen (840) der Vorhersage der Ausgabegröße mit dem Wert der Ausgabegröße, um ein Trainingssignal zu erhalten; und Aktualisieren (850) trainierbarer Parameter der Gaußschen Prozessregression basierend auf dem Trainingssignal.

8. Verfahren (800) nach Anspruch 7, wobei das Tensor-Train-Netzwerk mehrere Unterkomponenten umfasst, und wobei das Training separate Lernraten zum Trainieren der Basis-Kernelfunktion, der mehreren Unterkomponenten und/oder der Gaußschen Prozessregression verwendet.

9. Verfahren (800) nach Anspruch 7 oder 8, das ferner vor dem Ausführen des Ende-zu-Ende-Trainings das Trainieren des Tensor-Train-Netzwerks umfasst, um die Ausgabegröße von den physikalischen Eingabegrößen zu folgern.

10. Verfahren (800) nach Anspruch 9, wobei das Tensor-Train-Netzwerk mehrere Unterkomponenten umfasst, und wobei das Trainieren des Tensor-Train-Netzwerks wiederholtes Trainieren einer Unterkomponente des Tensor-Train-Netzwerks umfasst, während andere Unterkomponenten des Tensor-Train-Netzwerks fixiert bleiben.

11. Verfahren (800) nach Anspruch 10, wobei das Trainieren der Unterkomponente das Lösen eines linearen Systems in einem Satz von Parametern der Unterkomponente durch Ausführen einer LU-Faktorisierung umfasst.

12. Verfahren (800) nach einem der Ansprüche 7-11, wobei das Training ferner das Ausführen einer Hyperparameter-Optimierung der Gaußschen Prozessregression durch Zufallssuche umfasst.

13. Steuerungssystem (200) zum Steuern eines computergesteuerten Systems, das mit einer Umgebung interagiert, wobei das Verfahren Folgendes umfasst:

- eine Datenschnittstelle (220) zum Zugreifen auf Regressionsdaten zum Ausführen einer Gaußschen Prozessregression, wobei die Gaußsche Prozessregression dazu dient, einen Wert einer Ausgabegröße des computergesteuerten Systems und/oder seiner Umgebung basierend auf Messungen einer oder mehrerer physikalischer Eingabegrößen des computergesteuerten Systems und/oder seiner Umgebung zu folgern, wobei die Regressionsdaten Kernel-Hyperparameter einer Kernelfunktion und mehrere gekennzeichnete Instanzen umfassen, wobei eine gekennzeichnete Instanz Werte der Eingabe- und Ausgabegrößen umfasst;
- eine Sensorschnittstelle (260) zum Erhalten von Messungen der einen oder mehreren physikalischen Eingabegrößen;
- eine Aktuatorschnittstelle (280) zum Bereitstellen von Steuerungsdaten an einen Aktuator zum Steuern des computergesteuerten Systems;
- ein Prozessor-Subsystem, das dazu ausgelegt ist: über die Sensorschnittstelle die Messwerte der einen oder mehreren physikalischen Eingabegrößen zu erhalten; die Gaußsche Prozessregression auszuführen, um einen Wert der Ausgabegröße basierend auf den erhaltenen Messwerten zu folgern; Steuerungsdaten basierend auf dem Wert der Ausgabegröße abzuleiten; und über die Aktuatorschnittstelle die Steuerungsdaten an den Aktuator bereitzustellen, wobei das Ausführen der Gaußschen Prozessregression Folgendes umfasst:
 - Anwenden der Kernelfunktion, um jeweilige Ähnlichkeiten der erhaltenen Messungen mit den jeweiligen gekennzeichneten Instanzen zu berechnen, umfassend Anwenden eines trainierten Tensor-Train-Netzwerks auf die Messungen der einen oder mehreren physikalischen Eingabegrößen und auf Werte der einen oder mehreren physikalischen Eingabegrößen einer gekennzeichneten Instanz, um Merkmalsdarstellungen der Messungen und der gekennzeichneten Instanz zu erhalten, und Anwenden einer Basis-Kernelfunktion auf die Merkmalsdarstellungen, um Ähnlichkeiten der erhaltenen Messungen mit den gekennzeichneten Instanzen zu erhalten; und
 - Folgern des Werts der Ausgabegröße von den jeweiligen berechneten Ähnlichkeiten und jeweiligen Werten der Ausgabegröße der gekennzeichneten Instanzen.

14. Trainingssystem (100) zum Trainieren einer Gaußschen Prozessregression zur Verwendung beim Steuern eines computergesteuerten Systems nach einem der Ansprüche 1-6, wobei das System Folgendes umfasst:

- eine Datenschnittstelle (120) zum Zugreifen auf einen Trainingsdatensatz (030) von Trainingsinstanzen, wobei eine Trainingsinstanz des Trainingsdatensatzes Werte der Eingabe- und Ausgabegrößen umfasst; und
- ein Prozessorsystem (140), das dazu ausgelegt ist, ein Ende-zu-Ende-Training der Gaußschen Prozessregression auszuführen durch: Ausführen der Gaußschen Prozessregression an den physikalischen Eingabegrößen der Trainingsinstanz, um eine Vorhersage der Ausgabegröße zu erhalten, Verwenden eines Tensor-Train-Netzwerks als einen Merkmalsextrahierer für eine Basis-Kernelfunktion; Vergleichen der Vorhersage der Ausgabegröße mit dem Wert der Ausgabegröße, um ein Trainingssignal zu erhalten; und Aktualisieren trainierbarer Parameter der Gaußschen Prozessregression basierend auf dem Trainingssignal.

15. Flüchtliges oder nichtflüchtliges computerlesbares Medium (900), das Daten (910) umfasst, die Folgendes darstellen:

- Anweisungen, die, wenn durch ein Prozessorsystem ausgeführt, das Prozessorsystem dazu veranlassen, das computerimplementierte Verfahren nach einem der Ansprüche 1-6 und/oder einem der Ansprüche 7-12 auszuführen; und/oder
- Regressionsdaten zum Ausführen einer Gaußschen Prozessregression nach einem der Ansprüche 1-6, wobei die Gaußsche Prozessregression dazu dient, einen Wert einer Ausgabegröße des computergesteuerten Systems und/oder seiner Umgebung basierend auf Messungen einer oder mehrerer physikalischer Eingabegrößen des computergesteuerten Systems und/oder seiner Umgebung zu folgern, wobei die Regressionsdaten Kernel-Hyperparameter einer Kernelfunktion und mehrere gekennzeichnete Instanzen umfassen, wobei die Kernel-Hyperparameter Parameter eines Tensor-Train-Netzwerks umfassen, und wobei eine gekennzeichnete Instanz Werte der Eingabe- und Ausgabegrößen umfasst.

Es folgen 6 Seiten Zeichnungen

Anhängende Zeichnungen

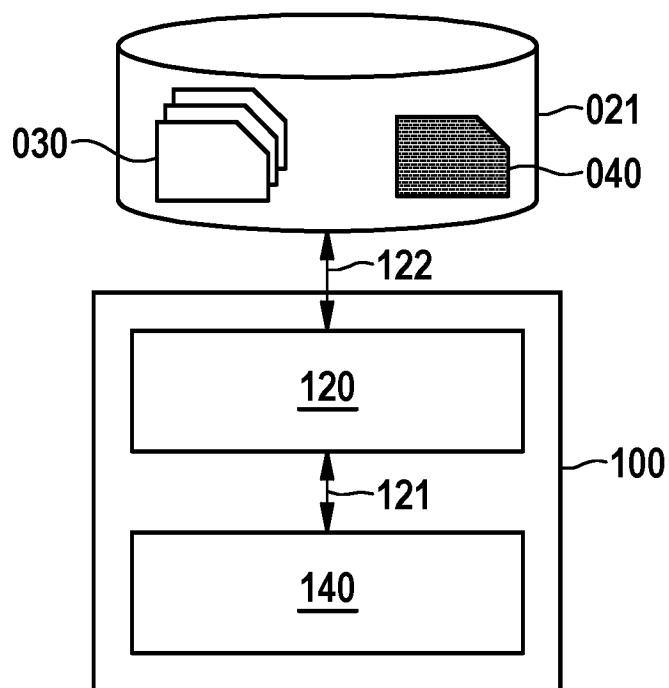


Fig. 1

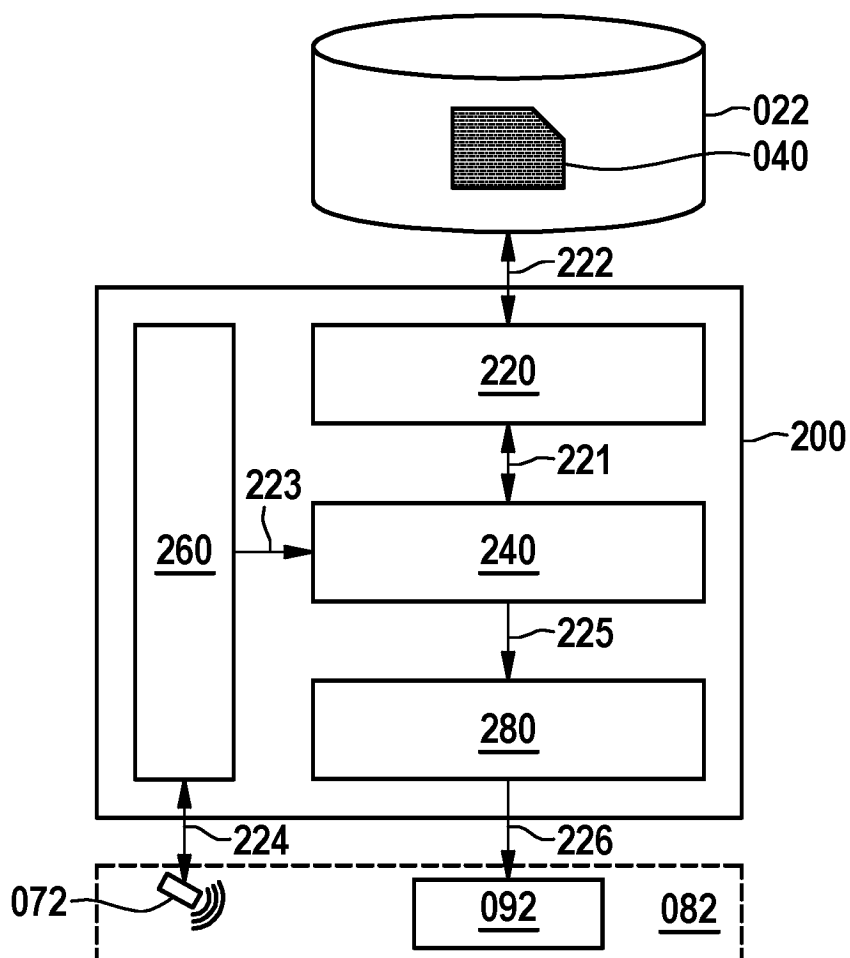


Fig. 2

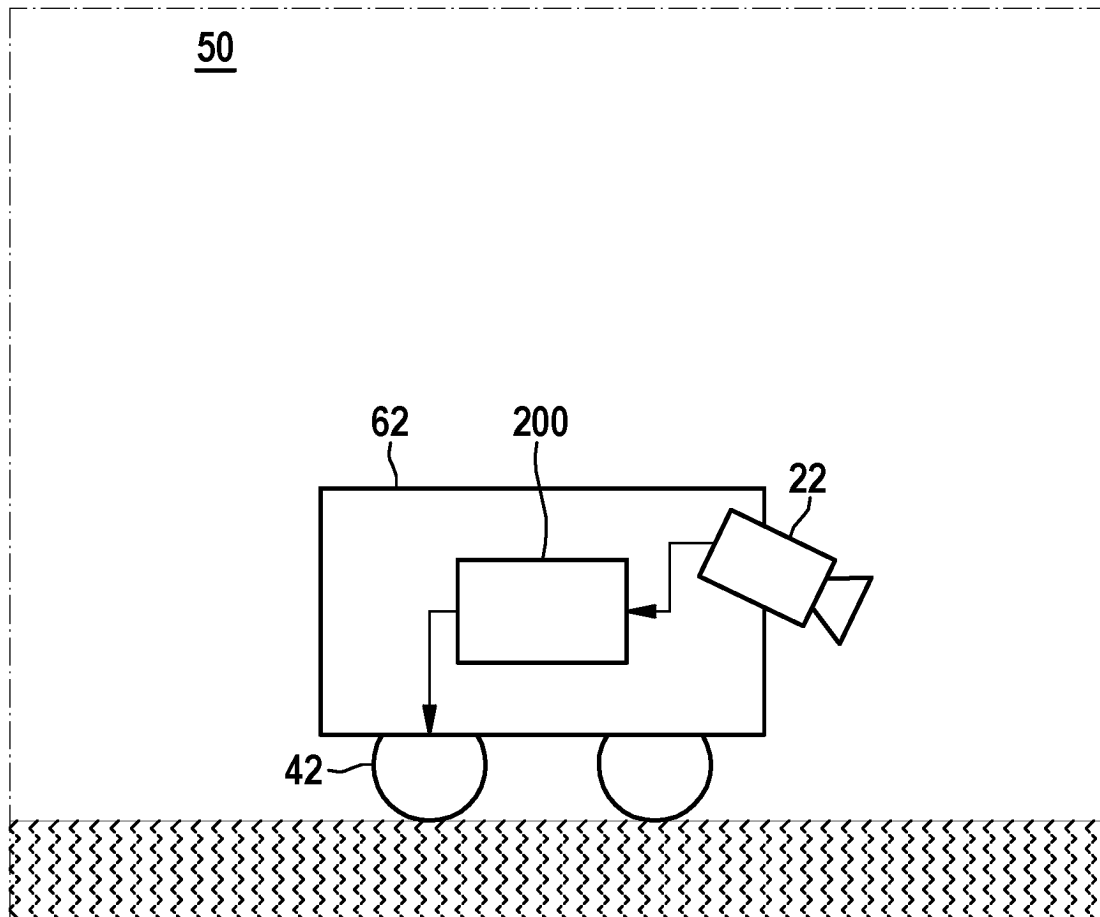


Fig. 3

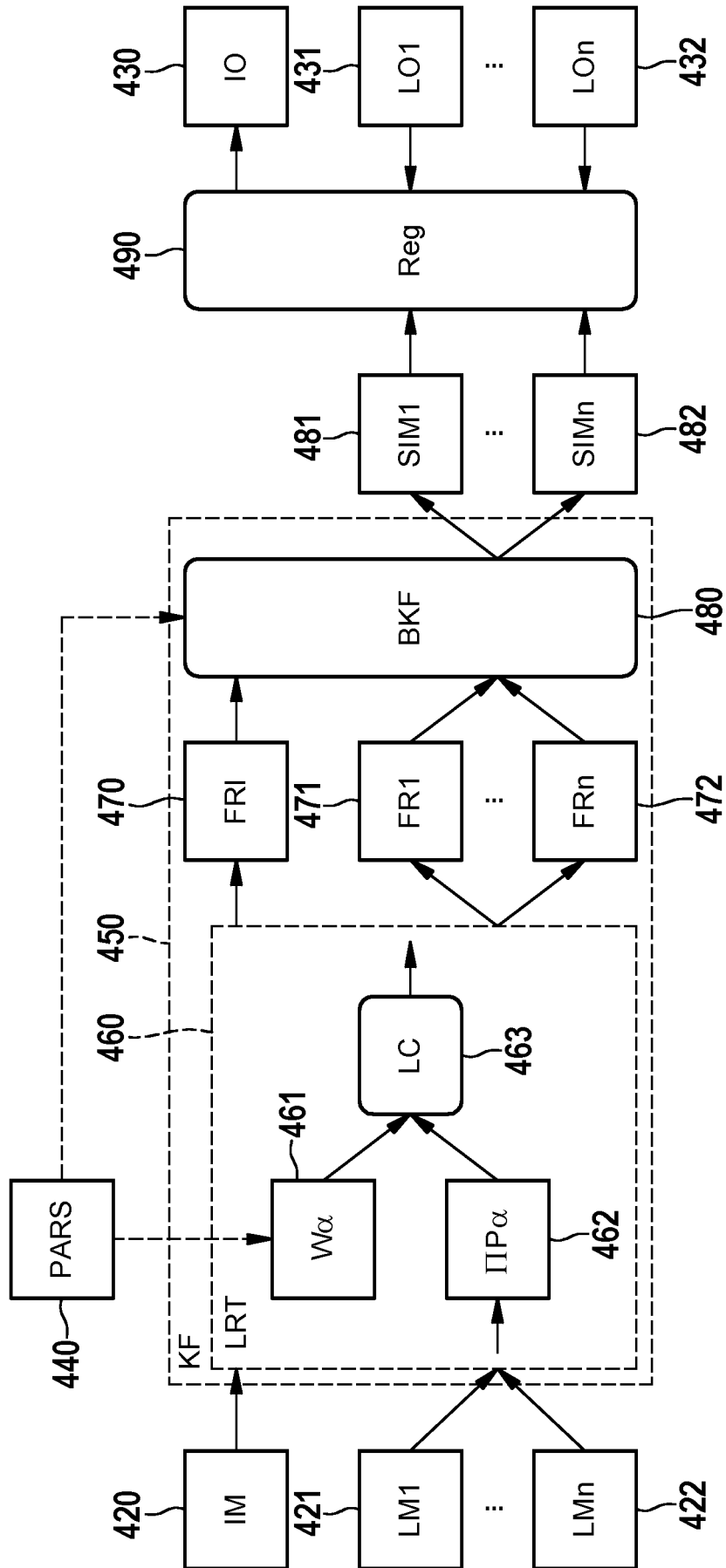


Fig. 4

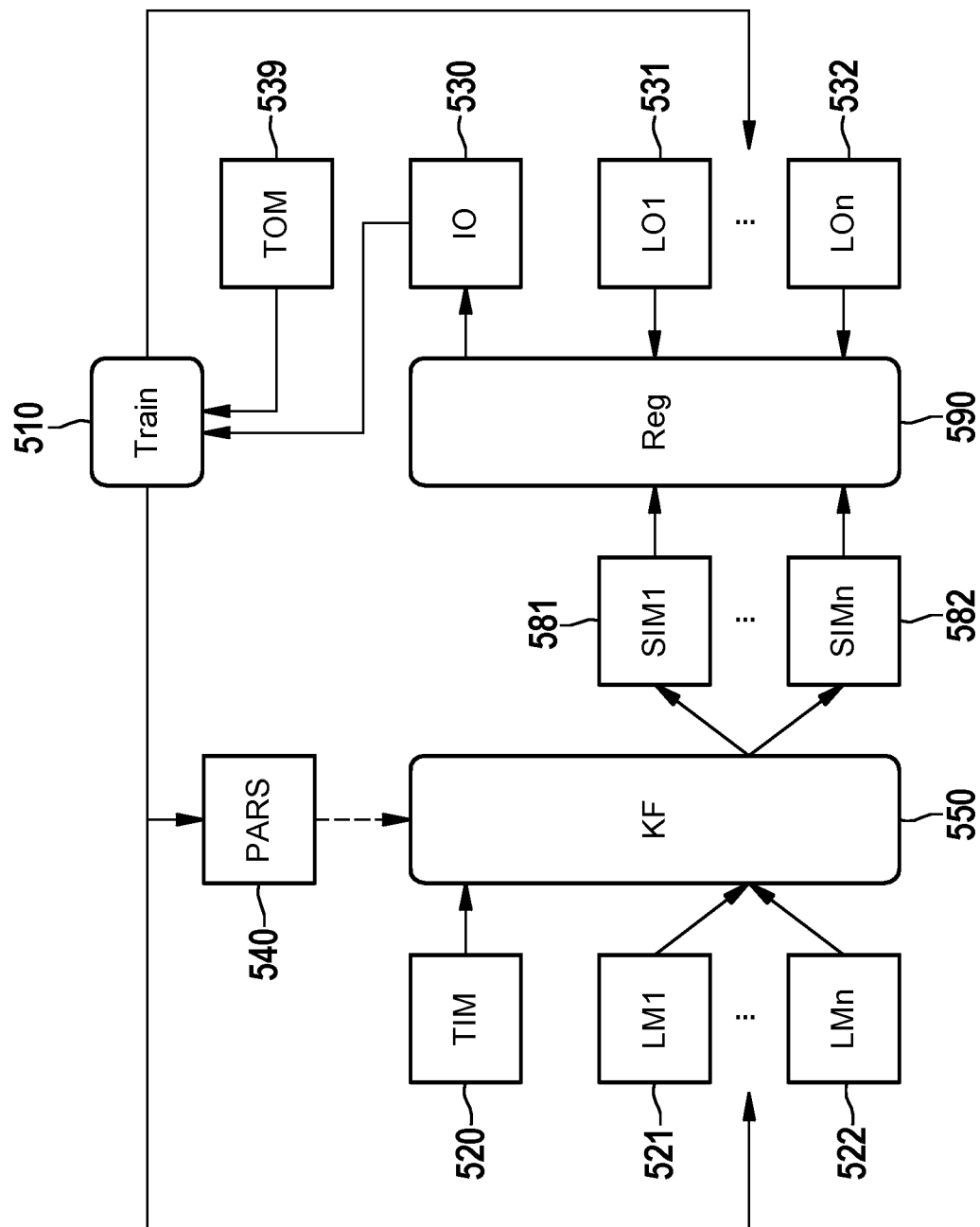


Fig. 5

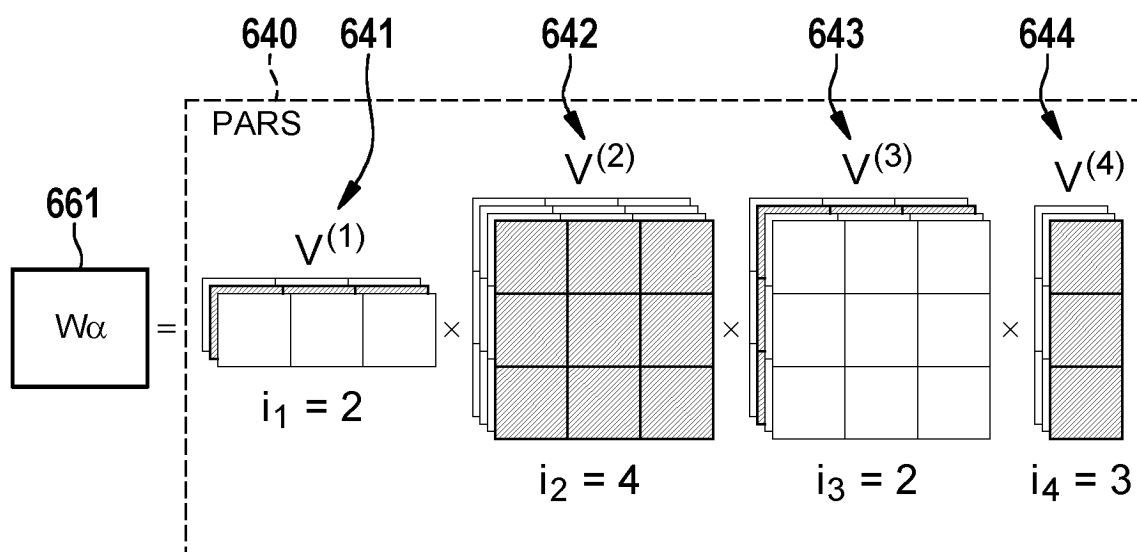


Fig. 6a

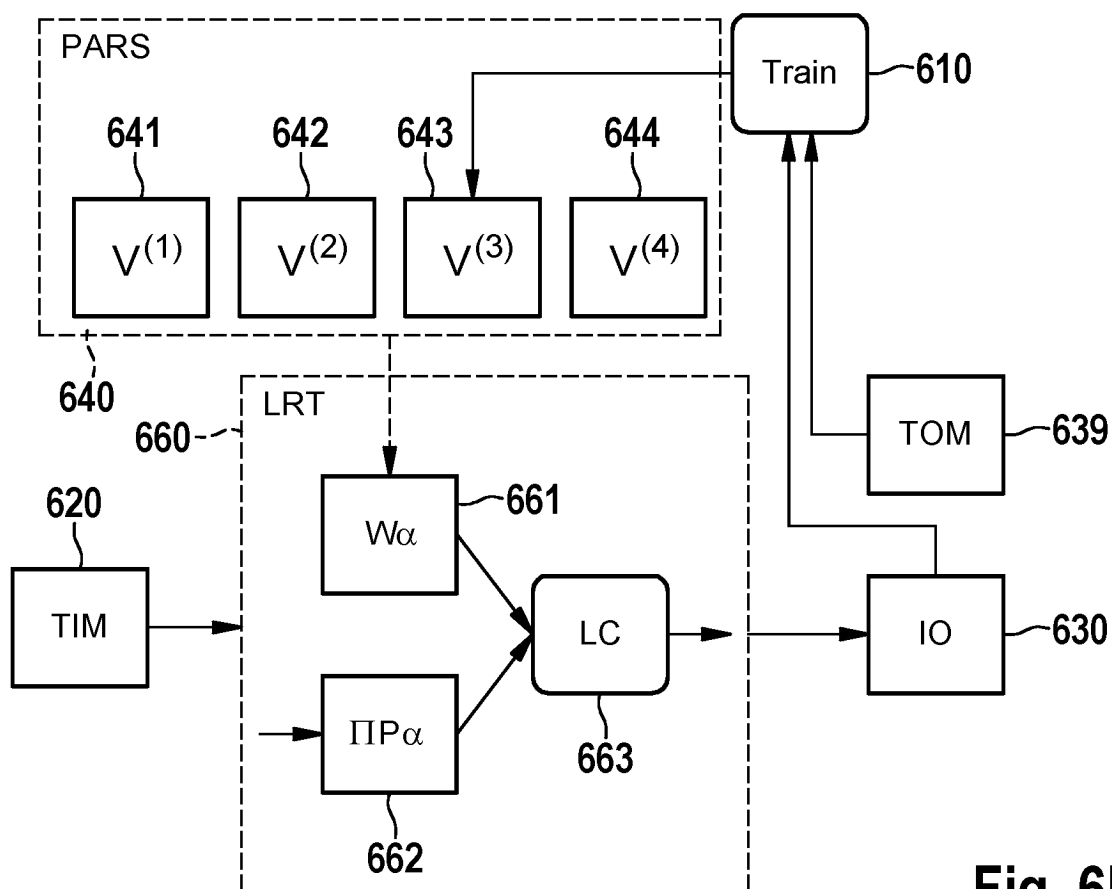


Fig. 6b

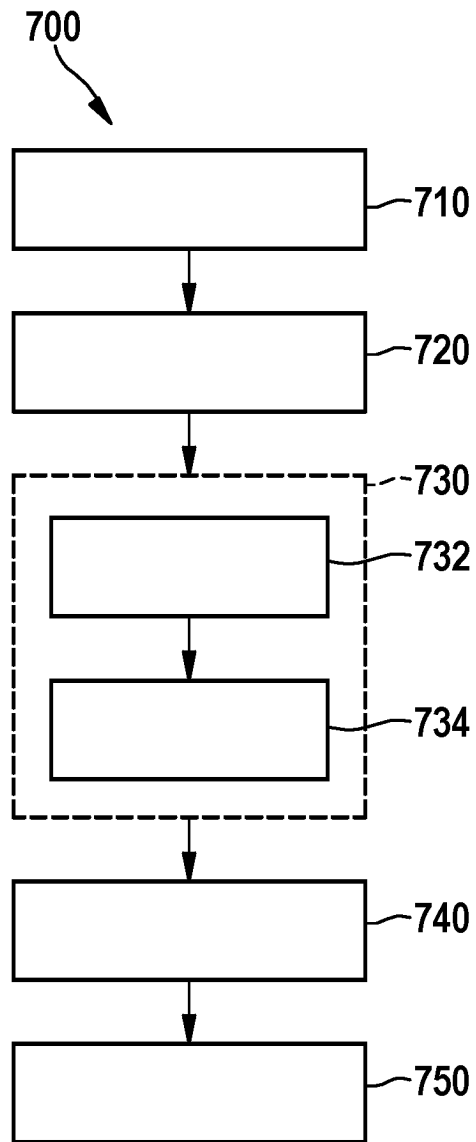


Fig. 7

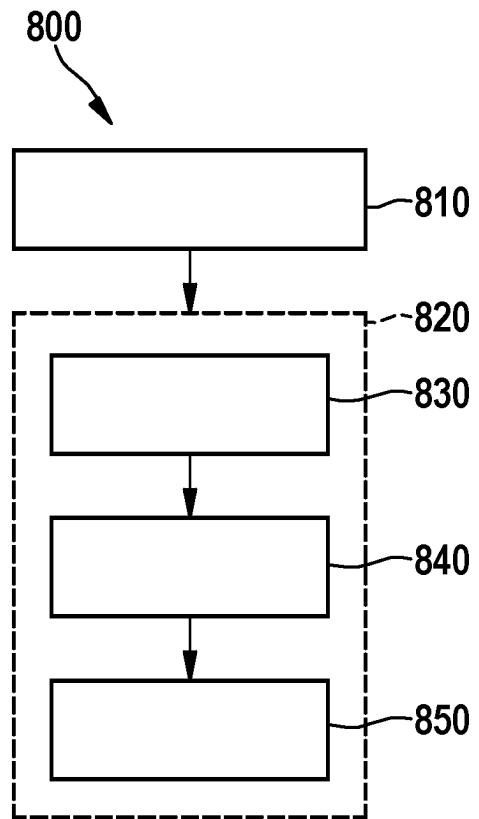


Fig. 8

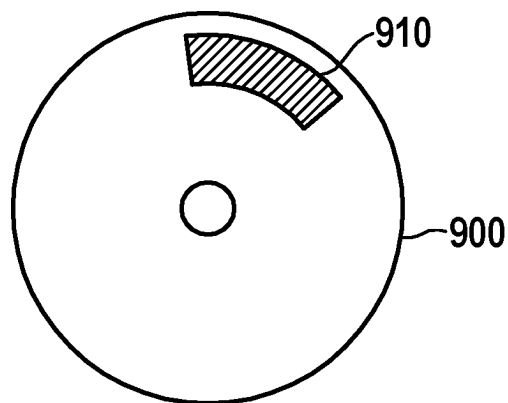


Fig. 9