Daten sind Zeichen plus zugehörige Syntax.

Daten werden zu Informationen, wenn sie eine Bedeutung haben.

Informationsverarbeitung verarbeitet Daten und ihre Bedeutung.

Wissen ist Fähigkeit Informationen zu nutzen.

Wissensverarbeitung ist Zwölferlei:

* Erkunden,
* Suchen,
* Erkennen,
* Identifizieren,
* Untersuchen,
* Analysieren,
* Bewusstmachen,
* Entscheiden,
* Verbessern,
* Restrukturieren,
* Behalten
* Informieren.

Merkmale der Wissensverarbeitung lassen sich in intelligenten Systemen:

* beliebigen Symbolen
* inneren Modells der äußeren Welt
* Schlussfolgerungen aus dem Wissen ziehen
* Verallgemeinerung (Abstraktion)
* Spezialisierung: Anwendung allgemeiner Zusammenhänge
* Generalisierung
* Planung
* Anpassungsfähigkeit
* Lernfähigkeit
* unscharf oder unvollständig
* Mustererkennung und zur aktiven Auseinandersetzungen
* Kommunikationsmittel

Unter Wissen (IT) werden auch Situationen verstanden, bei denen es darum geht, dass jemand etwas glaubt oder meint, oder etwas kennt oder kann.

Wissensrepräsentation ist die Formalisierung von Wissen



Prozedurale Wissensdarstellungen:

* Verfahren zur Konstruktion,
* Verknüpfung von Wissen
* Anwendung von Wissen
* durch Programmiersprachen darstellen

Deklarative Darstellungen:

* Beschreibungen von Sachverhalten
* „Die Summe aus 3 und 4 ist 7.“

Metawissen

* Steuerung deklarativer und prozeduraler Wissensbeschreibungen

Folgende Wissensarten werden repräsentiert:

* Objekte
* Ereignisse
* Handlungswissen
* Metawissen (Kontrollwissen)

Wissensart Objekte, ist die Kenntnis von Fakten über Objekte (Vogel hat Flügel). Man muss kategorisieren, beschreiben und verknüpfen können.

Wissensart Ereignisse, repräsentiert die Darstellung, zeitlichen Ablauf und Ursache-Wirkungs-Beziehung von Ereignisse.

Wissensart Handlungswissen, wie man Handlungen ausführt (Bildung von Sätzen).

Wissensart Metawissen, beschreibt den Umfang und die Herkunft des Wissens, die Verlässlichkeit Informationen und die relative Wichtigkeit spezifischer Fakten.

In Semantische Netze, repräsentieren Knoten die Objekte, Kanten die Relation zwischen den Objekten.

Objekt-Attribut-Wert-Tripel. Das Objekt beschreibt die physische Entität, das Attribut beschreibt dessen Eigenschaften und der Wert beschreibt die Beschaffenheit (Bsp.: Apfel-Farbe-rot | Apfel-Herkunft-Israel). OAW kann auch als Baumstruktur aufgespannt werden.

Ein Frame beschreibt ein Objekt, Akt oder Ereignis anhand einer Bündelung von Knoten und Attribut-Wert Paaren in einem semantischen Netz.

Regalbasierte Systeme, Produktionsregeln bestehen aus Prämissenteil (Ausführungsbedingung) und einem Aktionsteil (Ausführung bei erfüllter Bedingung).

Supervised Learning

* Lernen aus Beispielen
* Voraussetzung: Gegebene Ein- / Ausgabe-Beziehung (Trainingsbeispiele)
* Ziel: Modellbildung und Generalisierung
* Anwendung: Mustererkennung, Klassifikation, Prognose

Unsupervised Learning:

* Strukturierung von Daten
* Vorrausetzung: Trainingsdaten sind Beschreibungen von Situationen/Objekten
* Ziel: Lerne was typisch, ähnlich, interessant und besonders ist
* Anwendung: Clustering (Gruppieren)

Reinforcement Learning:

* Selbständiges und motivierendes (aufgrund von Belohnungen) Lernen
* Trail/Error
* Voraussetzung: Trainingsdaten bestehen aus einer Sequenz von Zuständen und ausgeführten Aktionen und deren Belohnungen
* Ziel: Erlerne die optimalste Strategie
* Anwendung: Definiert ein Lerntypus. Maximierung der Belohnungen bzw. Minimierung der Kosten.
* Direkte und zeitlich verzögerte Belohnungen bzw. Kosten.

Klassifikation bezeichnet die Klassenzugehörigkeit (Bsp. Guter oder Schlechter Kunde)

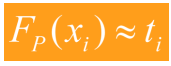
Regression versucht eine Vorhersage einer Erwartung zu treffen (Bsp. Erwartender gewinn o. Verlust)

Positive Belohnung, Agent erhält eine positive Rückmeldung nach einer Transition. Neutrale Belohnung, Agent erhält keine Belohnung. Negative Belohnung, Agent erhält eine negative Belohnung/Strafe/Kosten.

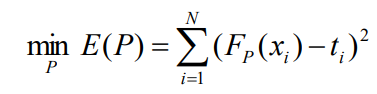
Merkmale sind reellwertige oder diskrete Attribute die Eigenschaften von Situationen oder Objekte bezeichnen:

* Reelwertig 🡪 Temperatur
* Diskret 🡪 Geschlecht, Monat

Ein Eingabemuster ist ein Vektor von Merkmalen die ein Objekt oder eine Situation beschreibt. Features and Labels.

Ziel beim überwachtem Lernen eines Modells ist es ein Modell in Form einer parametrisierten Funktion zu lernen die die Eingabemuster auf Ausgaben abbildet. Wir lernen also die Features um die Labels (so gut wie möglich) abbilden zu können.

Durch variieren des Parametervektor P soll der durchschnittliche quadrierte Fehler aller Trainingsmuster minimiert werden:



Da wir zu Beginn nicht wissen welcher Parametersatz P den Fehler E minimiert haben wir ein Optimierungsproblem. Durch Annährungsverfahren wie Gradientabstiegsverfahren kann P minimiert werden:



Machine Learning Algorithmen: lineare Modelle, neuronale Modelle, Entscheidungsbäume, SVM, dynamisches programmieren, speicherbasiertes Lernen. Zutaten: Gegeben/Gesucht, Repräsentation des gelernten Wissens, Lösungsverfahren.

Generalisierung um Overfitting zu vermeiden und vernünftige Aussagen über unbekannte Situationen treffen zu können.

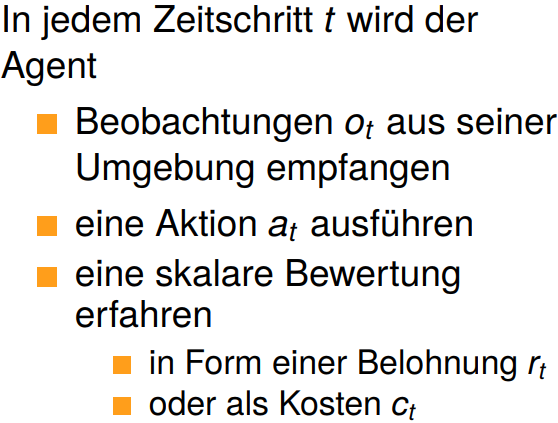
Zeitlich verzögertes RL, Belohnung tritt erst später ein (Aktion, Aktion… 🡪 Bewertung). Problem: Gewichtung der Aktion.

Mehrstufige Entscheidungsprobleme:

* Ziel: Aktionen wählen die die zukünftigen Belohnungen maximiert
* Problem: Aktion kann Konsequenzen in der Zukunft verursachen

Actor-Critic-System, Verteilt externe Bewertung auf einzelne Aktionen.

Vorgehensweise bei RL dynamisches programmieren: Optimierungsaufgabe formulieren, dann lernen Optimierungsverfahren. Approximative Verfahren, da Zustandsraum groß und Prozessverhalten unbekannt.

Agent und seine Umgebung:   


Die Historie des Agenten beinhaltet alle Beobachtungen, Aktionen und Bewertungen bis zum Zeitschritt t.

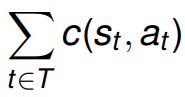
Der Zustand beschreibt die Aktion der Agent aufgrund der gegebenen Informationen wählt:

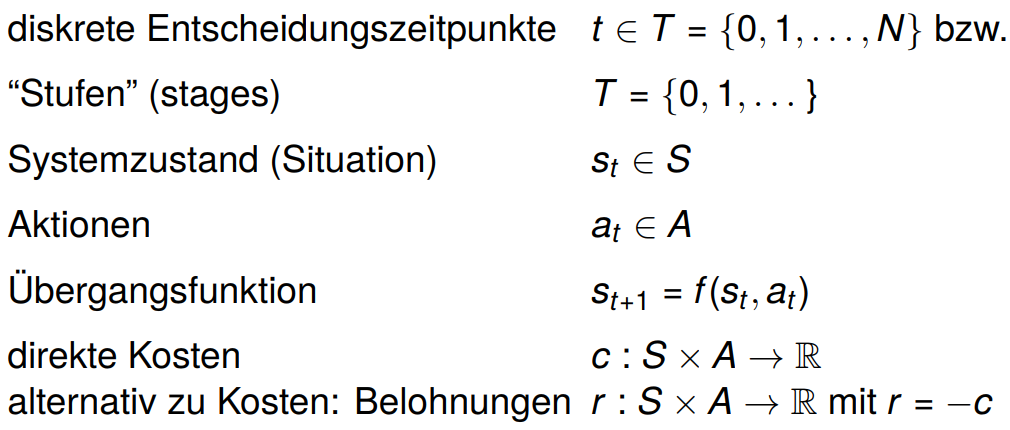
Zustand der Umgebung:   
Zustand des Agenten:   
Volle Beobachtbarkeit: =

Anforderungen an die Beschreibung der Umgebung:

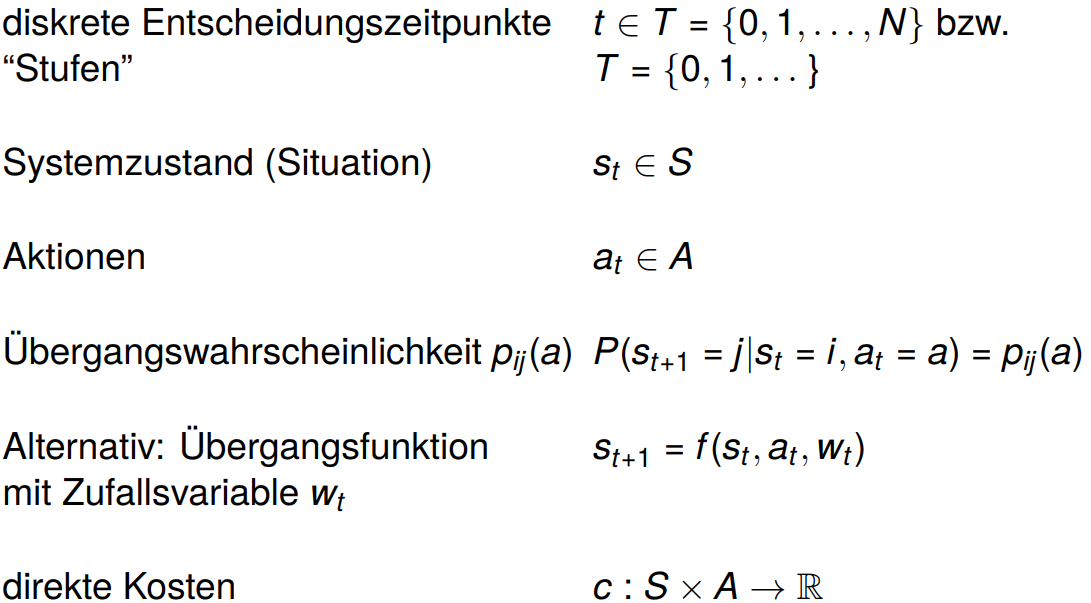
* Aktuelle Situation
* Zeitbedingte Eingriffe möglich
* Zielspezifikation durch Kosten/Belohnungen
* Störung und Rauschen

Kostenfunktion:

Kosten für jede Entscheidung:   
Gesamtkosten:

Deterministischer-Kürzester-Pfad (DKP):  
Zusammengefasst: 5-Tupel **(T, S, A, f, c (s, a))**

Stochastischer-Kürzester-Pfad (SKP), Störung und Rauschen wird berücksichtigt. Wird durch MDP modelliert.

Markov’sche Entscheidungsprozesse (MDP):  
  
Zusammengefasst: 5-Tupel **(T, S, A, pij (a), c (s, a))**

Ein MDP kann auch ein DKP-Problem abbilden, in dem die Übergangsfunktion durch eine Wahrscheinlichkeit von 100% abgebildet wird. Im Gegenzug, kann die von der DKP verwendete Übergangsfunktion in MDP dargestellt werden, indem man eine Zufallsvariable mit gegebener Wahrscheinlichkeitsverteilung einfügt.

Die Markov Eigenschaft fasst Informationen über die Historie im aktuellen Zustand zusammen.

Auswahlfunktion, beschreibt welche Aktion aus der Aktionsmenge der Agent zu einem gegebenen Zeitpunkt ausführt:  


Situationsabhängige Auswahlaktion, beschreibt welche Aktion aus der verfügbaren Aktionsmenge der Agent zu einem gegebenen Zeitpunkt ausführt:

Neben der deterministischen Auswahlfunktion gibt es auch stochastische die die Aktion des Agenten mit einer gewissen Wahrscheinlichkeit ausführen ([0, 1], weil alle Wahrscheinlichkeiten am Ende 100 ergeben müssen):



Die Strategie bestimmt das Handeln eines Agenten und damit implizit auch seine Belohnung. Eine Strategie (policy) πˆ besteht aus N Auswahlfunktionen (mit N als Anzahl der Entscheidungszeitpunkte):   
Ziel: Bestimmen der idealen Strategie, durch lösen des Optimierungsproblems mithilfe optimierenden Lernens.

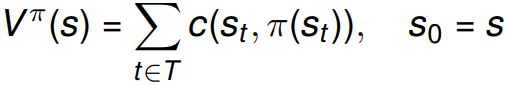
Nichtstationäre Strategien, wenn sich die Auswahlfunktion für einzelne Zeitpunkte unterscheiden. Der Agent kann für dieselbe Situation zu einem unterschiedlichen Zeitpunkt eine andere Entscheidung treffen. Beispiel: Fußball: Anfangs Rückpass, Gegen Ende Torschuss.

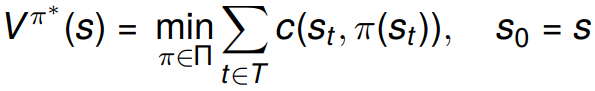
Stationäre Strategien, wenn die Auswahlfunktionen für alle Zeitpunkte identisch sind, also Strategie = Auswahlfunktion.

Wertfunktion V: S → R, bewertet die Güte von Zuständen und liefert eine reelwertige Abschätzung zukünftiger bzw. erwartender erhaltender Belohnungen.

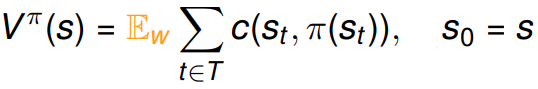


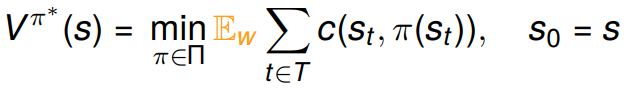
Pfadkosten besitzt dieselbe Eigenschaft wie die Wertfunktion, jedoch werden hier die Kosten betrachtet und nicht die Belohnungen. Diese beziehen sich auf einem gegebenen Zustand und einer Strategie:

  
Gesucht ist die optimale Strategie, so dass für alle s die Strategie mit den minimalsten Kosten gewählt werden:

  
Nebenbedingung: deterministische Übergangsfunktion.

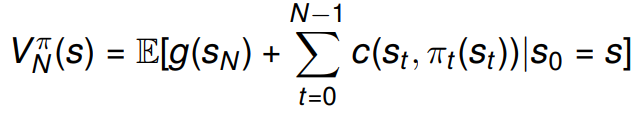
Pfadkosten im MDP (stochastisches System), hier werden die erwarteten Pfadkosten betrachtet bei gegebenem Zustand mit fester Strategie:

  
Gesucht ist die optimale Strategie, so dass für alle s die Strategie mit den minimalsten erwartenden Kosten gewählt werden:

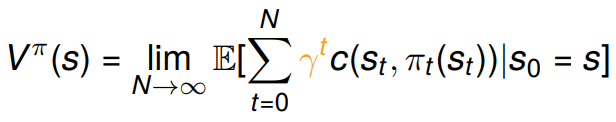
  
Nebenbedingung: stochastische Übergangsfunktion.

Der Horizont N eines Problems bezeichnet die Anzahl der zu durchlaufenden Entscheidungsstufen.

Endlicher Horizont, beschreibt Probleme mit vorgegeben Abbruchzeitpunkt.

* N-Stufiges Entscheidungsproblem
* Terminalkosten g(i), die anfallen, wenn System nach N Stufen in Endzustand endet.
* Kosten für eine Strategie
* Weil das Ende absehbar ist wird hier typischerweise nicht-stationäre Strategie verwendet, weil in jedem Zeitschritt eine andere Auswahlfunktion zur Aktionswahl eingesetzt werden kann bzw. sollte.

Unendlicher Horizont, Approximation für langandauernde Vorgänge bzw. Vorgänge mit unbestimmten Ende (Reglungsaufgabe):

* Aufgrund von unendlichen Kosten wird Diskontierung eingeführt
* Es gibt keine Finalkosten g(i)
* Diskontierung reduziert die Kosten die den Agenten erst in der Zukunft treffen

Das Modell sagt was die Umgebung als nächstes tun wird, dabei sagt pij(a) mit welcher Wahrscheinlichkeit welche Folgezustand eintritt und c(s,a) die nächsten Kosten voraus. Ein Modell kann unvollständig oder ungenau sein.

Um ein MDP-Problem modellieren zu können benötigen wir die Zustände, Aktionen, Übergangswahrscheinlichkeiten, Entscheidungskosten, Horizont/Diskontierungsparameter.

Kategorien von RL-Agenten:

1. Wertfunktion/Pfadkostenvektor Methoden
   1. Strategie ergibt sich aus der Funktion
2. Strategiebasierte Methoden
   1. Explizite Strategie
   2. Kommen ohne Wertfunktion/Pfadkostenvektor
3. Actor-Critic-Methoden
   1. Kombination aus 1. und 2.

Kategorien von Modellverwendung:

1. Modellfreie Methoden
   1. Verwendet eine Wertfunktion/Pfadkostenvektor
   2. Verwendet kein Modell der Umgebung
2. Modellbasierte Methoden
   1. Verwendet eine Wertfunktion/Pfadkostenvektor
   2. Verwendet ein Modell der Umgebung

Planen versus Lernen, bei mehrstufigen Entscheidungsproblemen unterscheiden man zwischen zwei Problemtypen:

1. Planung (Bsp. GVGAI):
   1. Modell der Umgebung ist bekannt
   2. Agent führt Berechnungen mit Modell durch
   3. Agent verbessert seine Strategie
2. Optimierendes Lernen (RL) (Bsp. PROJEKT)
   1. Umgebung ist zu Beginn unbekannt
   2. Agent interagiert mit Umgebung (lernt diese)
   3. Agent verbessert seine Strategie

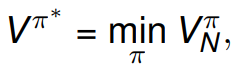
Da wir auf Basis von Trial/Error arbeiten und der Agent eine gute Strategie finden soll muss der Agent neugierig gestaltet werden. Die optimale Strategie definiert sich aus den gesammelten Erfahrungen des Agenten während diese so wenig Kosten wie möglich erfährt. Um dies zu erreichen muss der Agent sowohl Ausbeutung als auch Exploration beherrschen.

Ausbeutung ist die Ausnutzung bereits gesammelter Erfahrungen.

Exploration beschreibt die Auswahl unbekannter oder wenig bekannten Aktionen mit dem Zeil, um möglichst viele neue Informationen über die Umgebung zu sammeln und ggf. eine neue bessere Strategie zu finden.

Lösung dynamischer Optimierungsprobleme, durch dynamisches programmieren: Backward Dynamic Programming, Value Iteration oder Policy Iteration

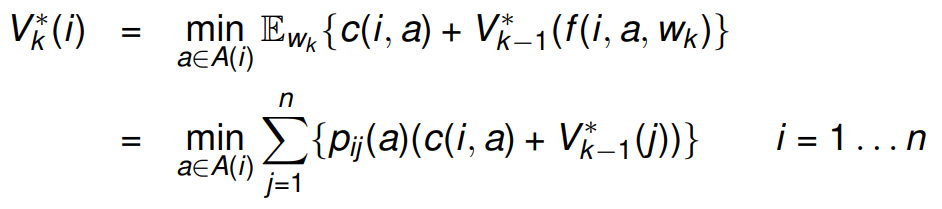
Backward Dynamic Programming

* Problem: stochastische oder mehrstufige Entscheidungsprobleme mit endlichem Horizont
* Idee: Berechne die Kosten von der letzten Stufe bis hin zur ersten.
* Gesucht: Ist eine optimale Strategie die die Kosten minimiert, also  für die gilt
* Hinweis: π\* wird als optimale Pfadkosten bezeichnet: 
* Vorgehensweise: 1. Berechnung der Pfadkosten für alle Zustände (n-dimensionaler Vektor). 2. Aus ergibt sich die optimale Strategie für das k-Schritt-Problem (k-schritte bis Prozess terminiert)
* Deterministisch O(N \* n) | stochastisch O(N\*²)

Bellman’sches Optimalitätsprinzip

Behauptung: Optimale Kosten für einen Zustand minimale Erwartungswert der Summe aus direkte Übergangskosten + optimale Pfadkosten des Folgezustands. Minimierung berücksichtigt alle verfügbaren Aktionen.

Bemerkung: Das Prinzip gilt aufgrund der Markov-Eigenschaft des Prozesses. BDP Algorithmus.

Formalisierung: Rekursive Berechnung der optimalen Kosten des N-stufigen Optimierungsproblems mit k=0:  


Weiteriterationsverfahren ist ein verfahren vom dynamischen Programmieren. Planung für ein MDP. Gute Konvergenzeigenschaft bei endlichen Zuständen. Benötigt unendlich viele Iterationen.

Wertiterationsverfahren bei unendlichem Horizont. Diskontierung wird dem MDP hinzugefügt:

2 Problemtypen: 1. SKP-Problem, 2. Diskontiertes-Problem

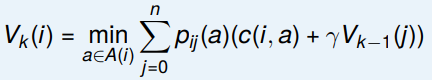
Deterministischer Weiteriterationsverfahren:

Man berechnet für alle Zustände den/die optimalen Aktionen mit folgender Formel:



Solange bis eine Konvergenz entsteht, also keine Änderungen entstehen: Vk = Vk-1

Value Iteration, Ähnlich wie deterministisch, nur dass hier Wahrscheinlichkeiten berücksichtigt wird, also:



Erfüllende Strategie, eine Strategie ist erfüllend, wenn von jedem Startzustand der Terminalzustand in N-Schritten mit positiver Wahrscheinlichkeit erreicht werden kann, damit sind die kosten endlich.

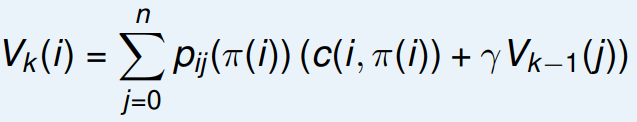
Konvergenzvorrausetzung bei Value Iteration für SKP. Es muss mind. 1 erfüllende Strategie geben (SKP-V1). Für jede nicht erfüllende Strategie muss ein Zustand der Strategie unendliche Pfadkosten besitzen (SKP-V2).

Konvergenzvorrausetzung bei Value Iteration für diskontierte Probleme. Umformulierung von diskontiert zu SKP, durch die Einführung eines Terminalzustands „0“ und ändern von Übergangsfunktionen. Aus jedem Zustand kann mit jeder Aktion mit Wahrscheinlichkeit von   
1 - Gamma in den Terminalzustand übergegangen werden.

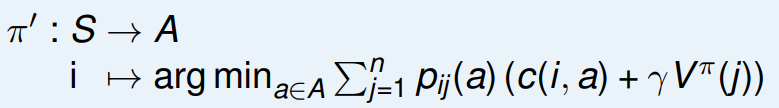
Strategieiterationsverfahren,

* Iteration über die Menge möglicher Strategien.
* Starte mit einer beliebigen erfüllenden Strategie
* Ermittle die dazugehörige Bewertungsfunktion
  + Strategiebewertung
* Werte diese „gierig aus und wähle neue Strategie
  + Strategieverbesserung
* Wiederhole bis die optimale Strategie gefunden

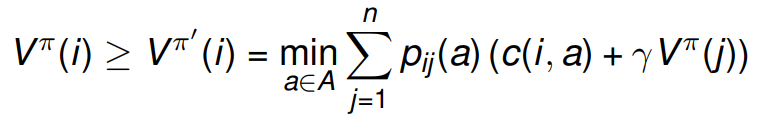
Strategiebewertung,

* Problem: Bewerte eine Strategie
* Lösung: Iterative Anwendung der Bellmann-Gleichung
* Eine Folge von Wertfunktionen bis Konvergenz eintritt
* Funktion: 

Strategieverbesserung, Strategie gegeben und eine zugehörige Wertfunktion. Über gierige Auswertung soll eine bessere Strategie gewonnen werden.

Gierige Strategie: 

Verbesserung (Kosten kleiner):



Strategieiteration, Wiederholung von Strategiebewertung + Strategieverbesserung, solange bis keine Änderung. Die Strategie verbessert sich im jeden Schritt.

Problemlösung dynamisches Programmieren:

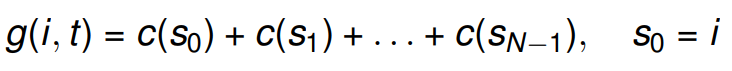
* Entscheidungsproblem
* Formulierung als MDP
  + Zustände / Aktion
  + Problemtyp
    - En- unendlicher Horizont
    - SKP, Diskontierung
* Lösen des MDPs

TD Verfahren wenn Modell unbekannt, zu viele o. unendliche Zustände. Lernen durch Interaktion mit Umgebung. Trajektorie (Rollout) liefert Pfadkosten gemäß aktueller Strategie. Mithilfe der Rollouts sollen die erwarteten Kosten geschätzt werden.

Real Time Dynamic Programmierung, Initialisiere Startzustand. Solange nicht im Terminalzustand, führe Aktionen durch, exploriere die Umgebung und Anwendung auf Prozess (Bewerten).

Es gibt 2 TD-Methoden Ansätze:

1. Generiere Rollouts von Startzustand bis zum Terminalzustand (MC und TD(1)
2. Benutze auch Informationen zwischen den Rollouts für die Zwischenzustände (übergangsbasierter Ansatz, TD(Lambda) und TD(0))

Monte-Carlo-Methode TD(1), wir betrachten die Pfadkosten einer Trajektorie, startend im Zustand i:

Dann gilt: 

Kann auch abgeschätzt werden: 

Danach kann man die MC Strategieevaluierung ausführen:  


Bei stochastischer Approximation muss Alpha gegen null gehen.

Zeitliche Differenzmethode. Benutzt auch die Informationen zwischen den Trajektorien. anpassen nach jedem Zustandsübergang auf Basis der Diskrepanz zwischen den Zustandsbewertungen, mithilfe Zeitlicher Differenzfehler:

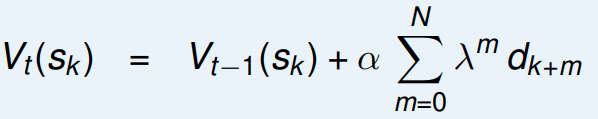


Der Aktualisierungsschritt der Wertfunktion V sieht dann so aus:



Übergangsbasiert, Aktualisierung gemäß Bellmann nach jedem Übergang durch das Verwenden von mehreren Übergangskosten (I) im Aktualisierungsschritt (z.B. 3-step-ahead).

TD(Lambda), Berücksichtigung mehrere Folgeschritte bei der Aktualisierung mit exponentiell abfallende Gewichtung, wobei Lambda ein Korrekturterm ist:



Mit zeitlichem Differenzfehler:

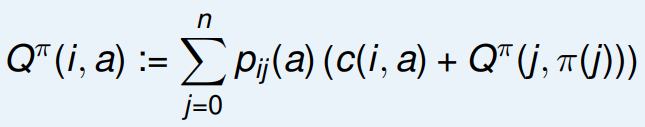


Online-Version: Anpassung nach jedem Übergang gemacht werden innerhalb eines Rollouts.

Offline-Version: Anpassung durch durchlaufen eines kompletten Rollouts.

Konvergenz, wenn jeder Zustand unendlich oft besucht wird und die Schrittrate der Lernrate vernünftig abnimmt.

Zustands-Aktions-Wertfunktion. Die Q-Funktion, schätzt die erwartenden Kosten bzw. Belohnungen für einen Agenten ab, wenn dieser in Zustand i die Aktion a gemäß der Strategie ausführt. Die Q-Funktion betrachtet also Zustand und Aktion und bewertet im Gegensatz zur V-Funktion jede mögliche Aktion eines Zustands.



Aktionsauswahl, hier betrachteten wir e-greedy für Exploration und Explotation. Mit einer Wahrscheinlichkeit lassen wir den Agent Aktionen ausführen die von dem Zeitpunkt bekannte, bestmögliche abweiche.

Optimistische Strategieiteration, Episoden gesteuerter Lernprozess (wie Trajektorien bei MC) mit endlicher Anzahl k. Strategiebewertung mit MC. Lösung zwar nicht exakt aber verzichten auf unendliche viele Trajektorien.

Modellfreie Strategieverbesserung:  


On-Policy, der Agent interagiert mit der zu bewertenden Zielstrategie mit der Umwelt.

Off-Policy, der Agent verhält sich gemäß einer anderen Strategie will aber trotzdem etwas über Pi lernen (also Pi bewerten).

Q-Learning, off-policy:

SARSA, on-policy:

GLIE, Konvergenz für Exploration. Epsilion läuft gegen 0. Wenn t gegen unendlich läuft steigt die Wahrscheinlichkeit die beste Aktion ausführen statt zu explorieren.

Boltzman-Exploration, T ist ein Temperaturparameter, wenn T klein ist wir der geringste Q-Wert genommen.

Approximation der Wertfunktion, wenn n sehr groß oder kontinuierlich ist, durch:

* Diskretisieren
* Regression

Diskretisieren, effizient, transparent und gut analysierbar, aber hoher Speicherbedarf, Generalisierungsprobleme und für praktische Probleme nicht einsetzbar.

* Regelmäßige: Zustandsraum wird regelmäßig diskretisiert
* Unregelmäßig: Individuelle Auflösung
* Adaptive: Verfeinerung in wichtige Bereiche

Regression, anhand einer gegebenen Mustermenge soll ein Modell, d.h. Funktion abgebildet werden die die gegebenen Datenpunkte möglichst genau approximiert.

Regressionsbäume, diskretisieren anhand eines Baums. Blätter werden mit Zielwert verknüpft. Spalten eines Blatts, wenn Bsp. zu viel Varianz. Vorteile: adaptiert auf Komplexität, informativ und leistungsfähig. Nachteile: gute Splitting-Kriterien schwer zu finden, Baumtiefe beschränkt und ist auch nur eine Form der Diskretisierung.