

Skript zur Vorlesung
Managing and Mining Multiplayer Online Games
im Sommersemester 2014

Kapitel 8: Zeitliche Verhaltensmodellierung

Skript © 2012 Matthias Schubert

http://www.dbs.ifi.lmu.de/cms/VO_Managing_Massive_Multiplayer_Online_Games

Kapitelüberblick

- Verhalten und Sequenzen
- Vergleiche zwischen Sequenzen
- Finden häufiger Teilsequenzen
- Markow-Ketten
- Hidden Markow-Ketten
- Zeitreihen und Feature-Transformationen
- Vergleich zwischen Zeitreihen
- Poisson-Prozesse

Spielerverhalten

Beispiele für Spielerverhalten

- Abfolge der Züge im Schach
 - Folge von Bewegungen, Aktionen und Interaktionen in einem MMORPG
 - Folge von Aufträgen an die eigenen Einheiten in RTS Games
-
- Abstrakt besteht Verhalten aus einer Folge von möglichen Aktionen
 - Einfachstes Modell für Verhalten sind Strings oder Sequenzen.

Definition: Sei $A = \{A_1, \dots, A_n\}$ ein endliches Alphabet von n möglichen Spieleraktionen. Dann heißt der l -Tupel $(a_1, \dots, a_l) \in A \times \dots \times A$ Sequenz der Länge l über A .

Bemerkung:

- Modell beschreibt erstmal nur Beobachtung und unterscheidet nicht zwischen möglichen und unmöglichen Sequenzen.
- Modell vernachlässigt die Zeit zwischen den Aktionen.

3

Beispiel: SC II Zerg Rushes

Aggressive Pool First - Liguipedia Starcraft 2 Wiki - Mozilla Firefox

While the build is a cheese build, in that it needs to do damage or else you will be behind, it is made to transition into a standard game after the first 6 or 8 Zerglings. However it can be turned into all in simply by continuing to build Zerglings.

Basic Build Order

All the variations of this opening are shown below:

- 6 Pool**
 - 6 Spawning Pool
 - 5 Drone
 - 6 Drone
 - 100% Spawning Pool, 3 pairs of Zerglings
 - 10 Zergling (Extractor Trick)
 - 11 Overlord
- 7 Pool**
 - 7 Spawning Pool
 - 6 Drone
 - 7 Drone
 - 8 Overlord
 - 100% Spawning Pool, 3 pairs of Zerglings
 - 4th pair of Zerglings when the larva becomes available
- 8 Pool**
 - 8 Spawning Pool
 - 7 Drone
 - 8 Drone
 - 9 Overlord
 - 100% Spawning Pool, 3 pairs of Zerglings
 - 4th pair of Zerglings when the larva becomes available
- 9 Pool**
 - 9 Spawning Pool
 - 8 Drone
 - 9 Drone
 - 10 Drone (Extractor Trick)
 - 11 Overlord
 - 100% Spawning Pool, 3 pairs of Zerglings
 - 4th pair of Zerglings when the larva becomes available
- 10 Pool**
 - 10 Spawning Pool
 - 9 Drone
 - 10 Overlord
 - 10 Drone (Extractor Trick)
 - 100% Spawning Pool, 3 pairs of Zerglings
 - 4th pair of Zerglings when the larva becomes available

Notes

The only key difference in the variations, besides the number of drones and timing of Zerglings, is that a 6 Pool doesn't allow for an Overlord to be made before the Zerglings are ready.

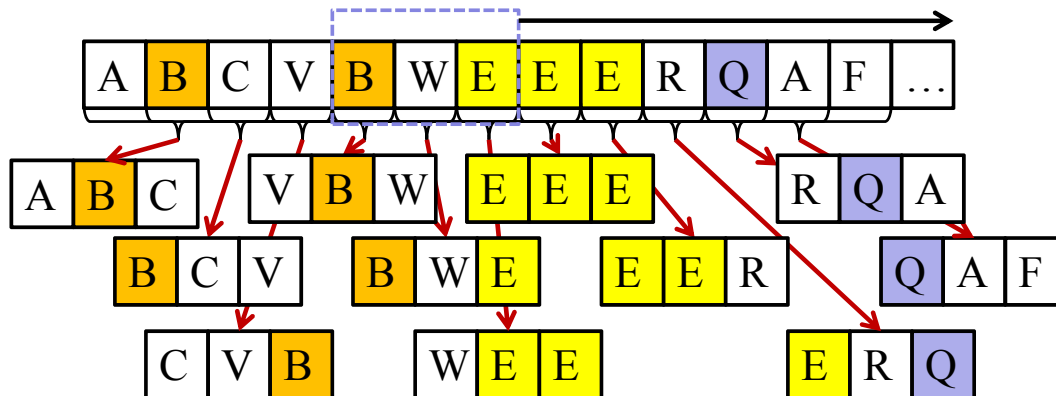
10 Pool allows you to start your Queen almost immediately after your initial 6 Zerglings are made.

Spawning Pool timings

4

Subsequenzen und Aufteilung der Beobachtung

- Wann wird welcher Spieler wie lange beobachtet?
- Je länger ein Spieler beobachtet wird, desto unwahrscheinlicher wird es, das ein anderer Spieler sich sehr ähnlich verhält.
- Zum Finden typischer Verhaltensmuster wird eine Sequenz in der Regel in Subsequenzen aufgeteilt.
- Aufteilen durch Windowing.
Schiebe ein Fenster der Länge k über die Sequenz und betrachte alle Teilsequenzen: im Beispiel ist $k = 3$



5

Subsequenzen und Aufteilung der Beobachtung

Problem: Eine Sequenz der Länge l hat $l - (k-1)$ k -elementige Subsequenzen und viele davon sind irrelevant.

Idee: Nur Sequenzen, die in einer bestimmten Häufigkeit auftreten sind interessant.

Frequent Subsequenz Mining

Finde alle Teilsequenzen einer Sequenz-Datenbank, die häufiger als *minsup* vorkommen. (vgl. frequent Itemset Mining)

⇒ Länge der Sequenz ist beliebig.

⇒ Suchraum ist größer als beim Itemset Mining.

(mehrfaches Auftreten eines Elements und Reihenfolge)

6

Frequent Subsequence Mining

- Häufigkeit $fr(S, G)$ von S in der Sequence G :
Zähle Vorkommen von S in G .
- relative Häufigkeit von S :

$$\varphi(S, G) = \frac{fr(S, G)}{|G| - |S| - 1}$$

- Sequenzbeschreibung von G :
 $\delta(G) = \{(S, \varphi(S, G)) \mid S \in G\}$
- Mining Sequential Patterns ist gut erforscht
=> viele Lösungsansätze und Algorithmen

7

Suffix Bäume

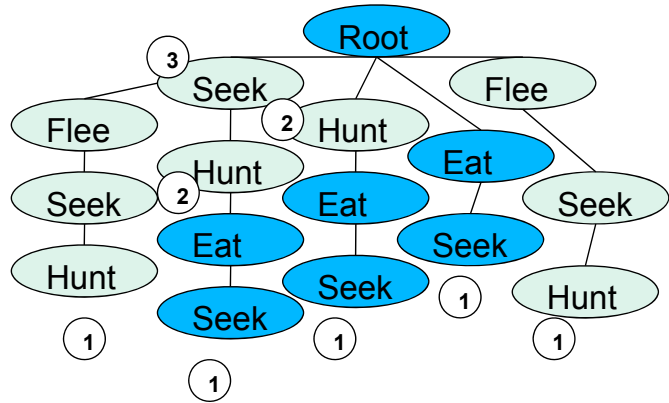
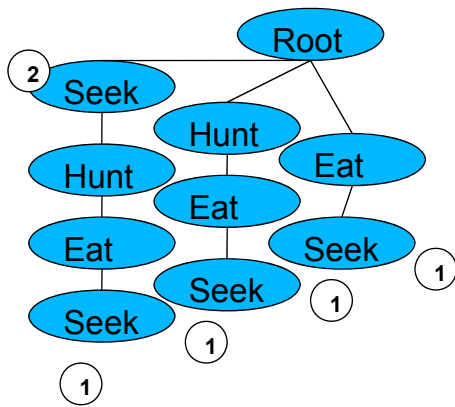
Eigenschaften eines Suffix Baums ST über dem Alphabet A
für Sequenz G mit $|G| = n$:

- ST hat genau $n+1$ Blattknoten, die von 0 bis n nummeriert werden.
- Kanten repräsentieren nicht-leere Teilsequenzen.
- Alle Kanten mit dem gleichen Startknoten beginnen mit einem anderem Element von A .
- Für jeden Blattknoten i ist die Konkatenation der Kanten des Pfads von der Wurzel zu i die Subsequenz $[i, \dots, n-1]$.

8

Suffix Trees

- Beispiel: Alphabet $A = \{\text{eat, hunt, seek, flee}\}$
- Einfügen:
 $S_1 = (\text{seek, hunt, eat, seek})$,
 $S_2 = (\text{seek, flee, seek, hunt})$



9

Das Interessante an Teilsequenzen

- interessant \neq häufig
- **Beispiel:** select drones, collect crystals, train drone, ...
Aber: Die ersten Aktionen in SC II sind fast immer identisch.
- die Anzahl häufiger Teilsequenzen kann sehr groß sein.
- die meisten davon beschreiben Standard-Game-Plays.
- Interessantheit sollte immer im Bezug auf ein weiteres Merkmal untersucht werden:
 - Karte (Bezug auf eine Örtlichkeit)
 - Spieler (Bezug auf das Individuum)
 - Strategie (Bezug auf Situation)
 - Kombination mehrerer Bezüge (Karte und Strategie ...)

10

Maß für die Interessantheit

- Verwendung von Maßen für die Korrelation:
- Finde eine Zielvariable: z.B. player_id
- Finde interessante Ereignisse: z.B. Boss-Kämpfe, Flaggenträger, ...
- Finde Orte, die ähnliches Verhalten auslösen: Spawn-Punkte für Monster, Flaggenabgabepunkte, Ort einer Bossbegegnung ...
- Berechnung durch zum Beispiel:
 - **Mutual Information**

$$MI(S, \text{Player_ID}) = \sum_{P \in \text{Players}} \sum_{S \in \{S_1, \bar{S}_1\}} \Pr[S, P] \cdot \log \frac{\Pr[S, P]}{\Pr[S] \cdot \Pr[P]}$$

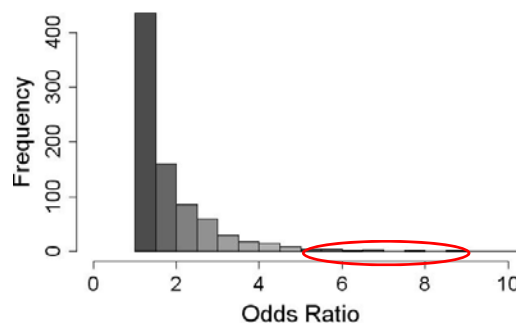
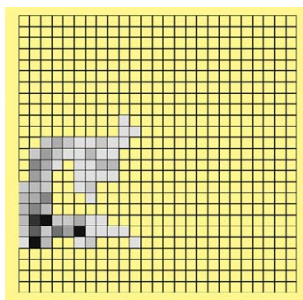
- **Odds Ratio**

$$\text{oddsR}_S(G_1, G_2) = \frac{\varphi(S, G_1)}{\varphi(S, G_2)}$$

11

Verwendung von Frequent Subsequences

- **Identifikation von Spielern:** Verwende das Vorkommen der k-„interessantesten“ Teilsequenzen als Dimensionen eines Vektorraums.
(hier Interessantheit = höchste MI mit der Player_id)
=> Beschreibe Spieler als Vektor der beobachteten Teilsequenzen.
- **Suche ortsspezifisches Verhalten:** Vergleiche die Häufigkeit von Handlungen auf der gesamten Karte mit der Anzahl der Handlungen in einem bestimmten Gebiet. (Odds-Ratio)



12

Vergleiche zwischen 2 Sequenzen

Gegeben: Alphabet A und eine Sequenzdatenbank

$$DB \subseteq \{(x_1, \dots, x_k) \mid k \in \mathbb{N} \wedge x_i \in A \text{ für } 1 \leq i \leq k\}.$$

Gesucht: Die Ähnlichkeit von $S1, S2 \in DB$.

- **Hamming Distanz:** Anzahl der unterschiedlichen Einträge aller Positionen.

Für 2 Sequenzen mit $|S1|=|S2|=k$:

$$Sim_{Ham}(S1, S2) = \sum_{i=0}^k \begin{cases} 0 & \text{if } s_{1,i} = s_{2,i} \\ 1 & \text{else} \end{cases}$$

Anmerkung: Bei unterschiedlich langen Sequenzen auffüllen der kürzeren Sequenz mit dem Gap Symbol „-“.

Beispiel: $S1 = (A,B,B,A,B)$ und $S2 = (A,A,A,A,A)$

$$\left. \begin{array}{l} (A,B,B,A,B) \\ (A,A,A,A,A) \end{array} \right\} Sim_{Ham}(S1, S2) = 2$$

13

Levenshtein Distanz

- **Hamming Distanz:** Berechnet minimale Kosten, um S1 in S2 zu verwandeln. Dabei sind nur Substitutionen der einzelnen Elemente erlaubt. (Aus B mache A.)
- **Idee:** Erweitere die erlaubten Änderungsoperationen durch Löschen und Einfügen von Symbolen.
- **Levenshtein Distanz:** Minimale Kosten, um S1 in S2 mit den 3 Operationen **Löschen**, **Einfügen** und **Substitution** zu verwandeln.

$$\left. \begin{array}{l} (A,B,B,A,B) \\ (A,A,B) \end{array} \right\} \left. \begin{array}{l} (A,B,B,A,B) \\ (A,-,-,A,B) \end{array} \right\} Sim_{Lev}(S1, S2) = 3$$

14

Berechnung der Levenshtein Distanz

Gegeben: Zwei Sequenzen S_1, S_2 über dem Alphabet A mit $|S_1| = n$ und $|S_2| = m$.

Gesucht: $Dist_{Lev}(S_1, S_2)$

Berechnung der Levenshtein Distanz mit dynamischer Programmierung:

Sei D eine $n \times m$ -Matrix über \mathbb{N} mit:

$$D_{0,0} = 0$$

$$D_{0,i} = i, \quad 0 \leq i \leq n$$

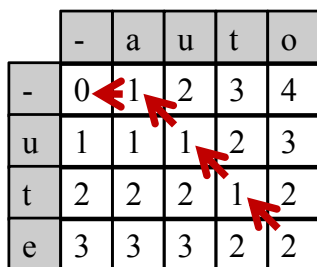
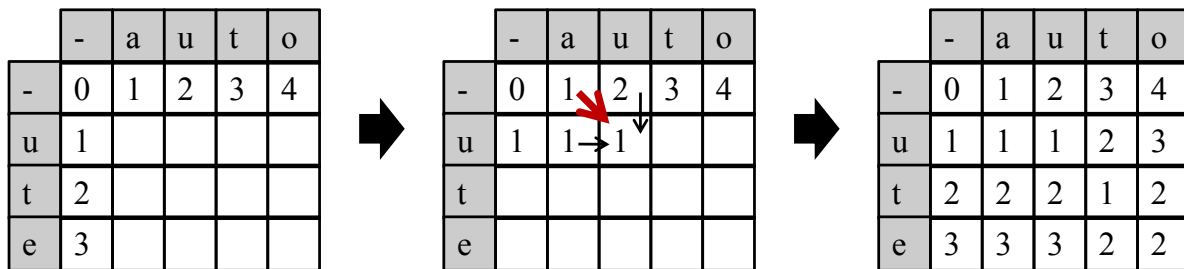
$$D_{j,0} = j, \quad 0 \leq j \leq m$$

$$D_{i,j} = \min \begin{cases} D_{i-1,j-1} + 0, \text{ falls } s_{1i} = s_{2j} \\ D_{i-1,j-1} + 1, (\text{Ersetzung}) \\ D_{i,j-1} + 1, (\text{Einfügung}) \\ D_{i-1,j} + 1, (\text{Löschung}) \end{cases} \quad \text{für } 1 \leq i \leq n, \quad 1 \leq j \leq m$$

Nach Aufbau der Matrix enthält der Eintrag $D_{n-1,m-1}$ die Levenshtein-Distanz.

Beispiel Levenshtein Distanz

$S_1 = \text{auto}, S_2 = \text{ute}$



$$\left. \begin{matrix} (a,u,t,o) \\ (-,u,t,e) \end{matrix} \right\} Dist_{Lev}(S_1, S_2) = 2$$

Edit Distanzen

- Verallgemeinerung der Levenshtein-Distanzen:
 - andere Kosten Matrix: Substitution kostet 4, Löschen 1, Einfügen 2..
 - weitere Operationen:
 - Reihenfolge vertauschen

$$\left. \begin{array}{l} (A, B, B, A, B) \\ (A, B, A, B, B) \end{array} \right\} 1 \text{ Vertauschung}$$

- vervielfältigen, ...

$$\left. \begin{array}{l} (A, B, B, B, B) \\ (A, B,) \end{array} \right\} 3 \text{ Vervielfältigungen von } B$$

- Kosten können für Werte unterschiedlich sein:
 $Subst.(A, B) \neq Subst.(A, Z)$
- Funktioniert auch bei Sequenzen über reellwertige Alphabeten,
zum Beispiel: für $A = IR$: $Subst(5, 1) = |5-1|$

17

Markow-Ketten und Sequenzen

- Sequenzen aus Aktionen unterliegen bestimmten Regeln
- Modellierung über endliche Automaten
(Test auf Gültigkeit der Sequenz)
- Markow-Ketten sind probabilistische Automaten:
Modellieren nicht nur erlaubte Zustandsübergänge, sondern auch
Wahrscheinlichkeitsverteilungen über Zustandsübergänge.
- **Markow-Annahme 1. Ordnung**: Der Zustand zum Zeitpunkt $t+1$ ist nur vom Zustand zum Zeitpunkt t abhängig.
- Ordnung einer Markow-Kette ist die Anzahl der
Vorgängerzustände, von denen die Auswahl des nächsten
Zustands abhängig ist.

18

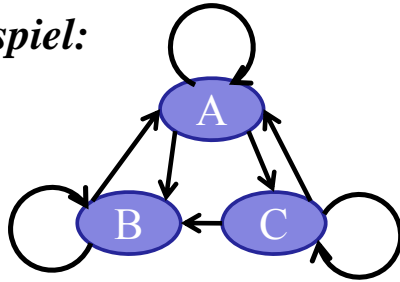
Markow-Ketten 1. Ordnung

Definition: Eine Markow-Kette M ist definiert über eine Zustandsmenge A und eine stochastische $|A| \times |A|$ Übergangsmatrix D .

Erläuterungen:

- A kann einen Start- und einen Absorbtionszustand enthalten (Modellierung von Start und Ende)
- Stochastische Matrix: Zeilen addieren sich auf 1 auf.
(Zeile i enthält die Verteilung der Nachfolger vom Zustand i)

Beispiel:



| | - | A | B | C |
|---|-----|------|-----|------|
| - | 0.0 | 0.3 | 0.3 | 0.4 |
| A | 0.1 | 0.25 | 0.5 | 0.15 |
| B | 0.1 | 0.5 | 0.4 | 0.0 |
| C | 0.1 | 0.1 | 0.7 | 0.1 |

$$p(ACBB) = P(A | -) \cdot P(C | A) \cdot P(B | C) \cdot P(B | B) \cdot P(- | B) \\ = 0.3 \cdot 0.15 \cdot 0.4 \cdot 0.7 \cdot 0.4 \cdot 0.1$$

19

Hidden Markow Modelle

Training einer Markow Kette:

- Zerlegen der Trainingssequenzen in 2-Gramme und bestimme die relative Häufigkeit.

(Wie häufig folgt auf A ein B?)
$$P(B | A) = \frac{fr(AB)}{fr(A)}$$

Problem:

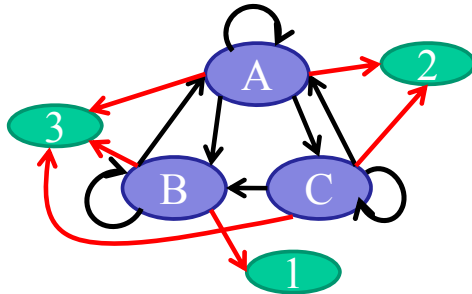
- Beobachtungen entsprechen häufig nicht direkt dem beobachteten Verhalten:
 - Action-Log ist verfügbar, aber Spielgeschehen soll analysiert werden
 - Fehlerhafte Ausführung verschleiert die tatsächliche Intention
 - Analyse der Zustandswechsel einer KI
(beobachtete Aktionen können in verschiedenen Zuständen angewendet werden)

20

Hidden Markow Modelle

Definition: Ein Hidden Markow Model M ist definiert über eine Zustandsmenge A , eine stochastische $|A| \times |A|$ Übergangsmatrix D , eine Beobachtungsmenge B und eine stochastische $|A| \times |B|$ Output-Matrix F .

Beispiel: $A = \{A, B, C\}$, $B = \{1, 2, 3\}$



| D | - | A | B | C | F | 1 | 2 | 3 |
|---|-----|------|-----|------|---|-----|-----|-----|
| - | 0.0 | 0.3 | 0.3 | 0.4 | A | 0.0 | 0.2 | 0.8 |
| A | 0.1 | 0.25 | 0.5 | 0.15 | B | 0.5 | 0.0 | 0.5 |
| B | 0.1 | 0.5 | 0.4 | 0.0 | C | 0.0 | 0.5 | 0.5 |
| C | 0.1 | 0.1 | 0.7 | 0.1 | | | | |

$P(122)$: Bestimme alle Zustandstripel, die 122 erzeugen können:

BAA, BAC

$$P(122) = P(BAA) \cdot P(122 | BAA) + P(BAC) \cdot P(122 | BAC)$$

Verwendung von HMMs

- **Evaluierung:** Wie wahrscheinlich ist eine Beobachtung $O = (o_1, \dots, o_k)$ mit $o_i \in B$ für das HMM (A, B, D, F) ?
(*Forward Estimation*)
- **Erkennung:** Gegeben ist die Beobachtung $O = (o_1, \dots, o_k)$ und das HMM (A, B, D, F) . Welche Sequenz (s_1, \dots, s_k) mit $s_i \in A$ erklärt O am besten? (*Viterbi-Algorithmus*)
- **Training:** Gegeben ist die Beobachtung $O = (o_1, \dots, o_k)$, wie kann man D und F anpassen, um $P(O | (A, B, D, F))$ zu maximieren?
(*Baum-Welch Estimation*)

Evaluierung: Forward Variablen

Gegeben: $O=(o_1, \dots, o_k)$ und (A,B,D,F)

Gesucht: $P(O|(A,B,D,F))$

Naive Lösung: Berechne $P(O|S)$ für alle k -elementigen Sequenzen S über A .
(Anzahl steigt exponentiell mit k)

Bessere Lösung: Ausnutzen der Markow-Annahme

Definiere Forward-Variable $\alpha_j(t)$ als

$$\alpha_j(t) = P(o_1, o_2, \dots, o_t, s_t = a_j | (ABDF))$$

Berechnung durch Induktion:

$$\alpha_j(1) = d_{-,j} \cdot f_{j,o_1}, \quad 1 \leq j \leq |A|$$

$$\alpha_j(t+1) = \left(\sum_{i=1}^{|A|} \alpha_i(t) \cdot d_{i,j} \right) \cdot f_{j,o_{t+1}}, \quad 1 \leq t \leq k-1$$

Berechnung mit $|A|^2 \cdot k$ Operationen:

$$P(O | (A, B, D, F)) = \sum_{i=1}^{|A|} P(O, s_t = a_i | (A, B, D, F)) = \sum_{i=1}^{|A|} \alpha_i(k)$$

23

Erkennung: Viterbi Algorithmus

Gegeben: $O=(o_1, \dots, o_k)$, und Model (A, B,D,F) .

Gesucht: $S=(s_1, \dots, s_k)$, die $P(O|S,(A, B,D,F))$ maximiert.

- Definiere $\delta(t)$ als höchste Wahrscheinlichkeit einer einzelnen Sequenz über A der Länge t für die Beobachtung O .

$$\delta_j(t) = \max_{s_1, \dots, s_{t-1}} P(s_1, \dots, s_{t-1}, O | (A, B, D, F))$$

- Berechnung durch Induktion

$$\delta_j(1) = d_{-,j} \cdot f_{j,o_1}, \quad 1 \leq j \leq |A|$$

$$\delta_j(t+1) = \left(\max_{1 \leq i \leq |A|} (\delta_i(t) d_{i,j}) \right) \cdot f_{j,o_{t+1}}, \quad 1 \leq j \leq k-1$$

$$\psi_j(1) = 0, \quad 1 \leq j \leq |A|$$

$$\psi_j(t+1) = \arg \max_{1 \leq i \leq |A|} (\delta_i(t) d_{i,j}), \quad 1 \leq j \leq k-1$$

- Ähnlich dem Forward Algorithmus, aber effizienter, da immer nur die beste Lösung verfolgt wird.

24

Backward Variablen

Analog zur Forward-Variable lässt sich auch eine Backward-Variable definieren, die für das Training des HMM verwendet wird.

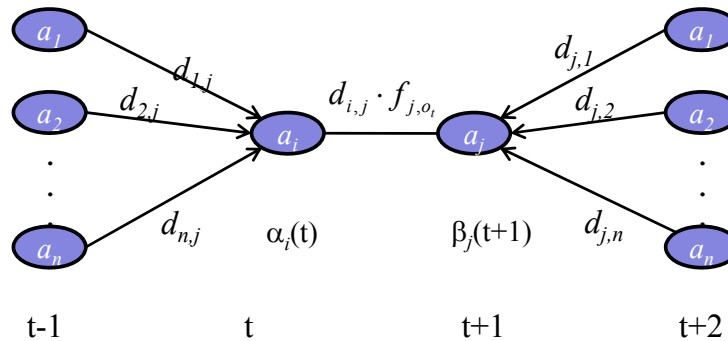
Definiere Backward-Variable $\beta_j(t)$ als

$$\beta_j(t) = P(o_{t+1}, \dots, o_k | s_t = a_j, (ABDF))$$

Berechnung durch Induktion:

$$\beta_i(k) = 1 \quad , 1 \leq i \leq |A|$$

$$\beta_i(t-1) = \sum_{j=1}^{|A|} d_{i,j} \cdot f_{j,o_t} \cdot \beta_j(t) \quad , 2 \leq t \leq k$$



25

Training: Baum-Welch Estimation

Gegeben: $O=(o_1, \dots, o_k)$, A und B .

Gesucht: D, F , die $P(O|(A,B,D,F))$ maximieren.

- Lokal optimierende Lösung durch *Expectation Maximization (EM)*
- Definiere $\xi_{i,j}(t)$ als Wahrscheinlichkeit, dass man zum Zeitpunkt t in Zustand a_i ist und zum Zeitpunkt $t+1$ in Zustand a_j :

$$\begin{aligned} \xi_{i,j}(t) &= P(s_t = a_i, s_{t+1} = a_j | O, (A, B, D, F)) \\ &= \frac{\alpha_i(t) \cdot d_{i,j} \cdot f_{j,o_{t+1}} \beta_j(t+1)}{P(O | (A, B, D, F))} \\ &= \frac{\alpha_i(t) \cdot d_{i,j} \cdot f_{j,o_{t+1}} \beta_j(t+1)}{\sum_{k=1}^{|A|} \sum_{l=1}^{|A|} \alpha_k(t) \cdot d_{k,l} \cdot f_{l,o_{t+1}} \beta_l(t+1)} \end{aligned}$$

- Definiere $\gamma_i(t)$ als die Wahrscheinlichkeit, dass man zum Zeitpunkt t in State a_i ist:

$$\gamma_i(t) = \sum_{j=0}^{|A|} \xi_{i,j}(t)$$

26

Training: Baum-Welch Estimation

- $\sum_{t=1}^{k-1} \xi_{i,j}(t)$ entspricht der erwarteten Anzahl von Zustandsübergängen von a_i auf a_j .
- $\sum_{t=1}^{k-1} \gamma_i(t)$ entspricht der erwarteten Anzahl an Zustandsübergängen von a_i auf andere Zustände.

- Parameter werden jetzt wie folgt neu abgeschätzt:

$$d_{-,a_i} = \gamma_i(1) \quad , \quad d_{i,j} = \frac{\sum_{t=1}^{k-1} \xi_{i,j}(t)}{\sum_{t=1}^{k-1} \gamma_i(t)} \quad , \quad f_{j,b_l} = \frac{\sum_{t \in \{t | o_t = b_l\}} \gamma_i(t)}{\sum_{t=1}^{k-1} \gamma_i(t)}$$

- Training erfolgt durch abwechselndes
 - Berechnen von $\gamma_i(t)$, $\xi_{i,j}(t)$ und $P(O|(A,B,D,F))$
 - Updates von D und F (Neuabschätzung wie oben)
- Algorithmus terminiert, wenn $P(O|(A,B,D,F))$ weniger als ε wächst.

27

Reellwertige Sequenzen

- **Bisher:** Alphabet ist eine diskrete Domäne
- Sequenzen können aber auch über reellwertige Domänen, zum Beispiel \mathbb{R}^d , gebildet werden.
- Frequent Pattern Mining auf reellwertigen Domänen ist i.d.R. nicht möglich.
- Vergleich von 2 reellwertigen Sequenzen über der Domäne D mittels einer Distanzfunktion $dist: D \times D \rightarrow \mathbb{R}_0^+$.

- Analog zur Hamming Distanz kann man die Summe der Distanzen an jeder Position der Sequenz bestimmen.

$$dist_{sequ}(S_1, S_2) = \sum_{i=1}^{|S_1|} dist(s_{1,i}, s_{2,i}) + (|S_2| - |S_1|) \cdot \varphi, \quad \text{für } |S_2| \geq |S_1|, \varphi \in \mathbb{R}^+$$

- Erweitern der Edit-Distanz ist ebenfalls möglich: Kosten für Substitution von v und u entsprechen dann $dist(v,u)$.

(Genauerer kommt später beim Dynamic Time Warping)

28

Zeitreihen

- **Bisher:** Sequenzen modellieren nur die Reihenfolge, nicht aber die Zeitpunkte der Handlungen.

Aber: In Echtzeitspielen ist Timing entscheidend.

⇒ RTS Spiele: Build-Order sind nur dann effektiv, wenn sie in minimaler Zeit realisiert werden.

⇒ In MMORPGs hängt der ausgeteilte Schaden von der Anzahl der Aktionen pro Zeiteinheit ab.

⇒ Schach mit Schachuhr: Ein Zug wird auch daran gemessen, wie viel Zeit man zum Nachdenken gebraucht hat.

- **Zeitreihe:** Sei T eine Domäne zur Darstellung der Zeit und sei F eine Objektdarstellung, dann heißt:

$Z = ((x_1, t_1), \dots, (x_l, t_l)) \in (F \times T) \times \dots \times (F \times T)$ Zeitreihe der Länge l über F .

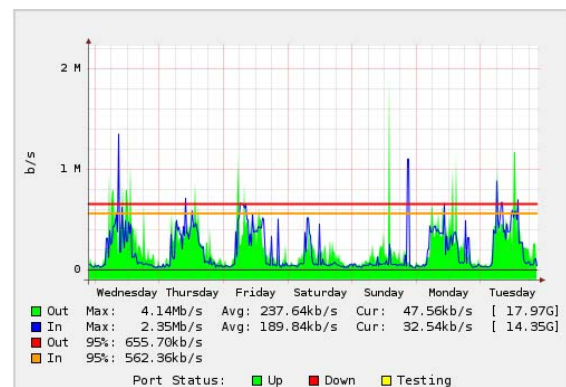
29

Beispiele für Zeitreihen

- SC2-Logs: Zeitreihe über diskrete Handlungen

```
0:00 TSLHyuN Select Hatchery (10230)
0:00 TSLHyuN Select Larva x3 (1027c,10280,10284), Deselect all
0:00 TSLHyuN Train Drone
0:01 TSLHyuN Train Drone
0:01 TSLHyuN Select Drone x6 (10234,10238,1023c,10240,10244,10248), Deselect all
0:01 TSLHyuN Right click; target: Mineral Field (10114)
0:01 TSLHyuN Deselect 6 units
0:02 TSLHyuN Right click; target: Mineral Field (10170)
....
```

- Netzwerk-Traffic:
 - Verwendung in der Bot-Erkennung
 - Abschätzung der Spielintensität



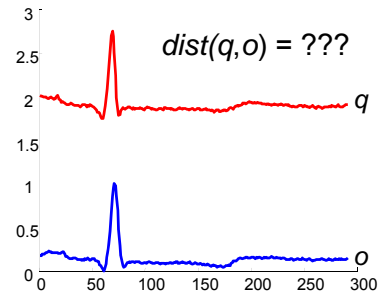
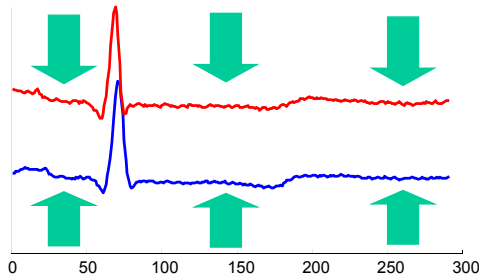
30

Vorverarbeitung von Zeitreihen (1)

Offset Translation

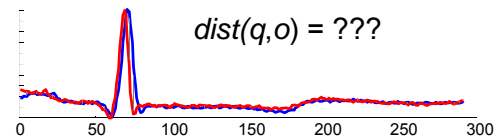
- Ähnliche Zeitreihen mit unterschiedlichen Offsets
- Verschiebung aller Zeitreihen um den Mittelwert MW :

$$\forall 1 \leq i \leq |o|: o_i = o_i - MW(o)$$



$$q = q - MW(q)$$

$$o = o - MW(o)$$

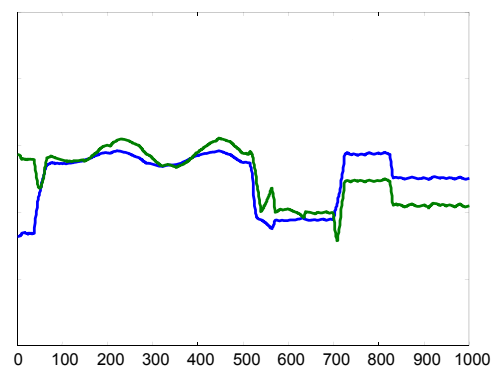
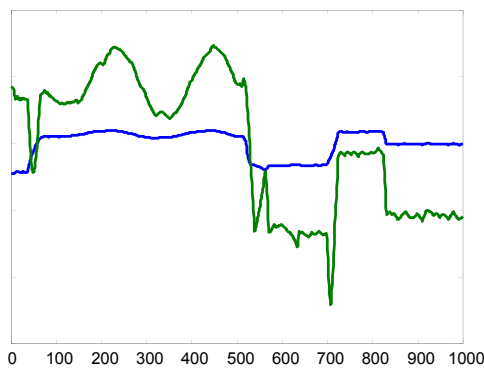


Vorverarbeitung von Zeitreihen(2)

Amplituden Skalierung

- Zeitreihen mit ähnlichem Verlauf, aber unterschiedlichen Amplituden
- Verschiebung der Zeitreihen um den Mittelwert (MW) und Normierung der Amplitude mittels der Standardabweichung (StD):

$$\forall 1 \leq i \leq |o|: o_i = (o_i - MW(o)) / StD(o)$$



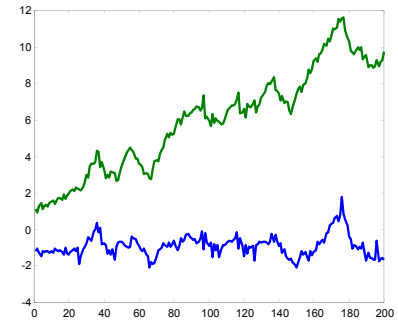
$$q = (q - MW(q)) / StD(q)$$

$$o = (o - MW(o)) / StD(o)$$

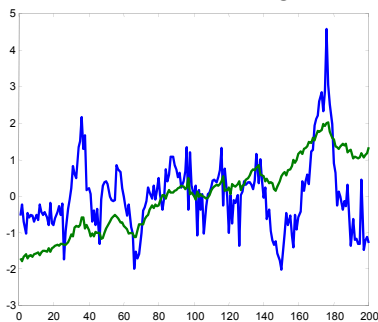
Vorverarbeitung von Zeitreihen (3)

Lineare Trends

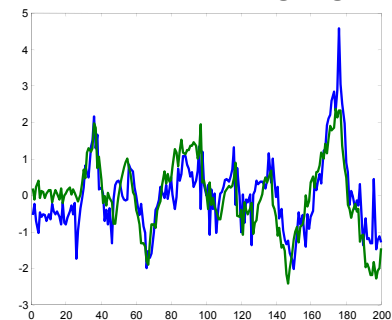
- Ähnliche Zeitreihen mit unterschiedlichen Trends
- Intuition:
 - Bestimme Regressionslinie
 - Verschiebe Zeitreihe anhand dieser Linie



Offset Translation + Amplituden
Skalierung



Offset Translation + Amplituden Skalierung
+ **Lineare Trend-Beseitigung**

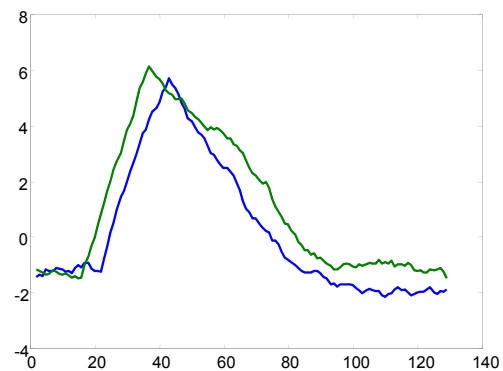
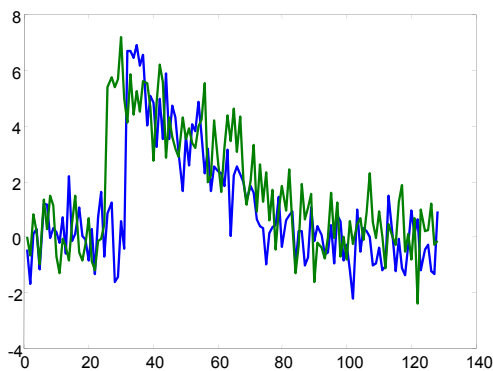


33

Vorverarbeitung von Zeitreihen (4)

Bereinigung von Rauschen

- Ähnliche Zeitreihen mit hohem Rauschanteil
- Glättung: Bilde für jeden Wert o_i den Mittelwert über alle Werte $[o_{i-k}, \dots, o_i, \dots, o_{i+k}]$ für ein gegebenes k .

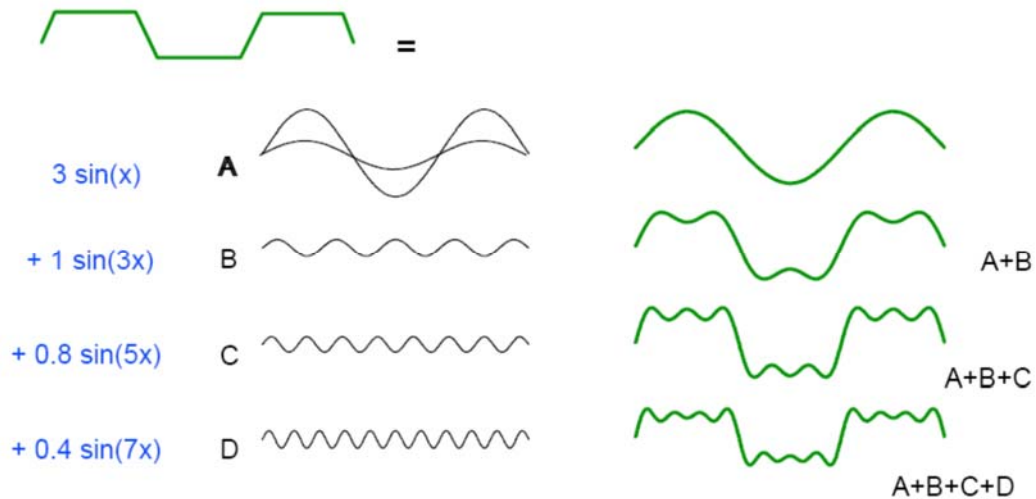


34

Diskrete Fourier Transformation (DFT)

Idee:

- Beschreibe beliebige periodische Funktionen als gewichtete Summe periodischer Grundfunktionen (Basisfunktionen) mit unterschiedlicher Frequenz. Aus einer Zeitreihe wird ein Vektor mit fester Länge.
- Basisfunktionen: sin und cos



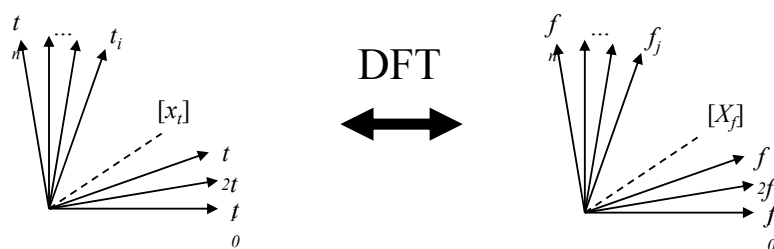
35

Diskrete Fourier Transformation (DFT)

Fouriers Theorem: Jede beliebige periodische Funktion lässt sich als Summe von Kosinus- und Sinus-Funktionen unterschiedlicher Frequenzen darstellen.

Eigenschaften:

- Transformation verändert eine Funktion nicht, sondern stellt sie nur anders dar
- Transformation ist umkehrbar => inverse DFT
- Analogie: Basiswechsel in der Vektorrechnung



36

Diskrete Fourier Transformation (DFT)

Formal:

- Gegeben sei eine Zeitreihe der Länge n : $x = [x_t]$, $t = 0, \dots, n - 1$
- Die DFT von x ist eine Sequenz $X = [X_f]$ von n komplexen Zahlen für die Frequenzen $f = 0, \dots, n - 1$ mit

$$X_f = \frac{1}{\sqrt{n}} \sum_{t=0}^{n-1} x_t \cdot e^{-i \cdot 2\pi \cdot f \cdot t / n} =$$
$$\underbrace{\frac{1}{\sqrt{n}} \sum_{t=0}^{n-1} x_t \cos\left(\frac{2 \cdot \pi \cdot f \cdot t}{n}\right)}_{\text{Realteil}} - i \cdot \underbrace{\frac{1}{\sqrt{n}} \sum_{t=0}^{n-1} x_t \sin\left(\frac{2 \cdot \pi \cdot f \cdot t}{n}\right)}_{\text{Imaginärteil}}$$

wobei i die imaginäre Einheit bezeichnet, d.h. $i^2 = -1$.

- Der Realteil gibt den Anteil der Kosinusfunktionen und der Imaginärteil den Anteil der Sinusfunktionen in der jeweiligen Frequenz f an.

37

Diskrete Fourier Transformation (DFT)

- Durch die inverse DFT wird das ursprüngliche Signal x wieder hergestellt:

$$x_t = \frac{1}{\sqrt{n}} \sum_{f=0}^{n-1} X_f \cdot e^{i \cdot 2\pi \cdot f \cdot t / n}$$

$t = 0, \dots, n - 1$ (t : Zeitpunkte)

$[x_t] \leftrightarrow [X_f]$ bezeichnet ein **Fourier-Paar**,

d.h. $\text{DFT}([x_t]) = [X_f]$ und $\text{DFT}^{-1}([X_f]) = [x_t]$.

- Die DFT ist eine **lineare Abbildung**, d.h. mit $[x_t] \leftrightarrow [X_f]$ und $[y_t] \leftrightarrow [Y_f]$ gilt auch:
 - $[x_t + y_t] \leftrightarrow [X_f + Y_f]$ und
 - $[ax_t] \leftrightarrow [aX_f]$ für ein Skalar $a \in \mathbb{R}$
- **Energie einer Sequenz**
 - Die Energie $E(c)$ von c ist das Quadrat der Amplitude: $E(c) = |c|^2$.
 - Die Energie $E(x)$ einer Sequenz x ist die Summe aller Energien über die Sequenz: $E(x) = \|x\|^2 = \sum_{t=0}^{n-1} |x_t|^2$

38

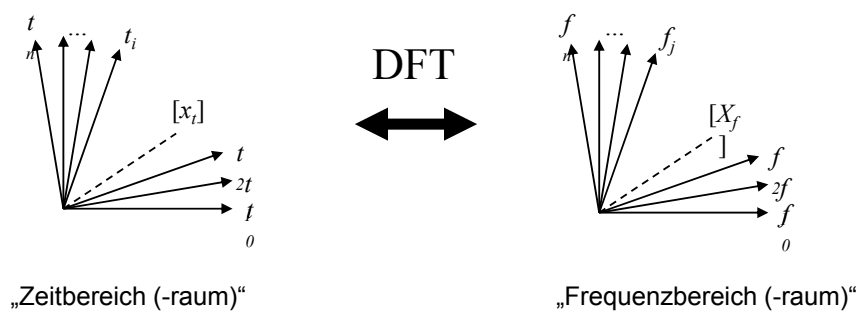
Diskrete Fourier Transformation (DFT)

Satz von Parseval: Die Energie eines Signals im Zeitbereich ist gleich der Energie im Frequenzbereich.

Formal: Sei X die DFT von x , dann gilt:

$$\sum_{t=0}^{n-1} |x_t|^2 = \sum_{f=0}^{n-1} |X_f|^2$$

- Folge aus Parsevals Satz und der Linearität der DFT: Die Euklidische Distanz zweier Signale x und y stimmt im Zeit- und im Frequenzbereich überein: $\|x - y\|^2 = \|X - Y\|^2$



39

Diskrete Fourier Transformation (DFT)

Grundidee der Anfragebearbeitung:

- Als Ähnlichkeitsfunktion für Sequenzen wird die Euklidische Distanz verwendet:

$$dist(x, y) = \|x - y\| = \sqrt{\sum_{t=0}^{n-1} |x_t - y_t|^2}$$

- Der Satz von Parseval ermöglicht nun, die Distanzen im Frequenz- statt im Zeitbereich zu berechnen: $dist(x, y) = dist(X, Y)$
- In der Praxis haben die tiefsten Frequenzen die größte Bedeutung.
- Die ersten Frequenz-Koeffizienten enthalten also die wichtigste Information.
- Für den Aufbau eines Index werden die transformierten Sequenzen gekürzt, d.h. von $[X_f], f = 0, 1, \dots, n - 1$ werden nur die ersten c Koeffizienten $[X_f < c]$, $c < n$, indiziert.

$$dist_c(x, y) = \sqrt{\sum_{f=0}^{c-1} |x_f - y_f|^2} \leq \sqrt{\sum_{f=0}^{n-1} |x_f - y_f|^2} = dist(x, y)$$

- Im Index kann dann eine untere Schranke der echten Distanz berechnet werden: Filter-Refinement:

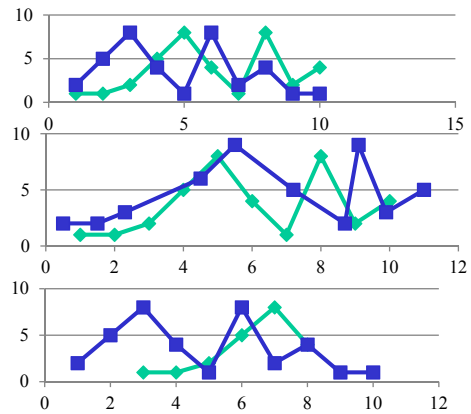
- Filterschritt auf gekürzten Zeitreihen (mit Indexunterstützung),
- Refinement auf kompletten Zeitreihen

40

Distanzen auf Zeitreihen

Probleme: Welche Zeitpunkte sollen verglichen werden?

- Offset am Anfang: S2 ist zu S1 zeitlich verschoben.
- Taktung der Messwerte: Zeitpunkte der Messungen sind unterschiedlich.
- Länge der Zeitreihe: Dauer der Messung ist unterschiedlich.



- Bei gleicher Taktung und Länge können Zeitreihen als Vektoren verglichen werden. (Dimension = Zeitpunkt)

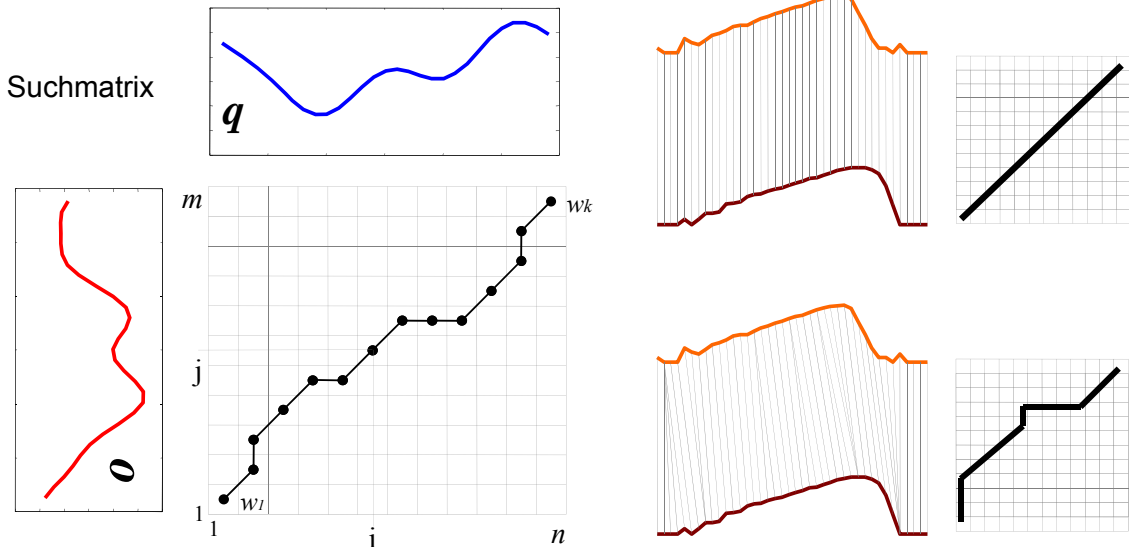
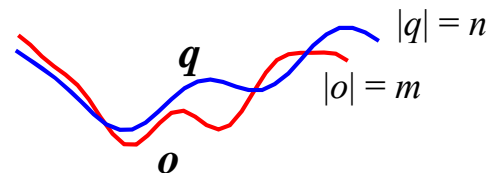
$$Dist_{timeseries}(S1, S2) = \sum_{t=1}^T dist_{obj}(s_{1t}, s_{2t})$$

- Bei Variabler Länge, Taktung und bei Offsets, Adaption der Edit-Distanz für Sequenzen => **Dynamic Time Warping**

Dynamic Time Warping Distanz

Berechnung:

- Gegeben: Zeitreihen q und o unterschiedlicher Länge
- Finde Mapping von allen q_i auf o mit minimalen Kosten



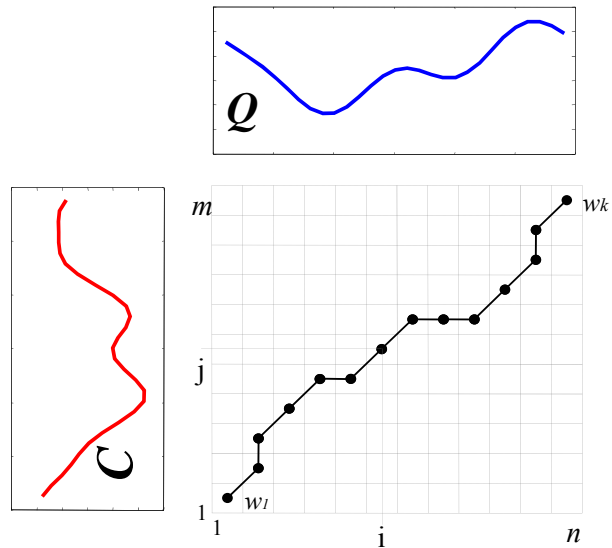
Dynamic Time Warping Distanz

Suchmatrix

- Alle möglichen Mappings von q auf o können als „warping“ Pfad in der Suchmatrix aufgefasst werden
- Von all diesen Mappings suchen wir den Pfad mit den niedrigsten Kosten

$$DTW(q, o) = \min \left\{ \sqrt{\sum_{k=1}^K w_k} / K \right.$$

- Dynamisches Programmieren
=> Laufzeit ($n \cdot m$)
(vgl. Edit Distanz)

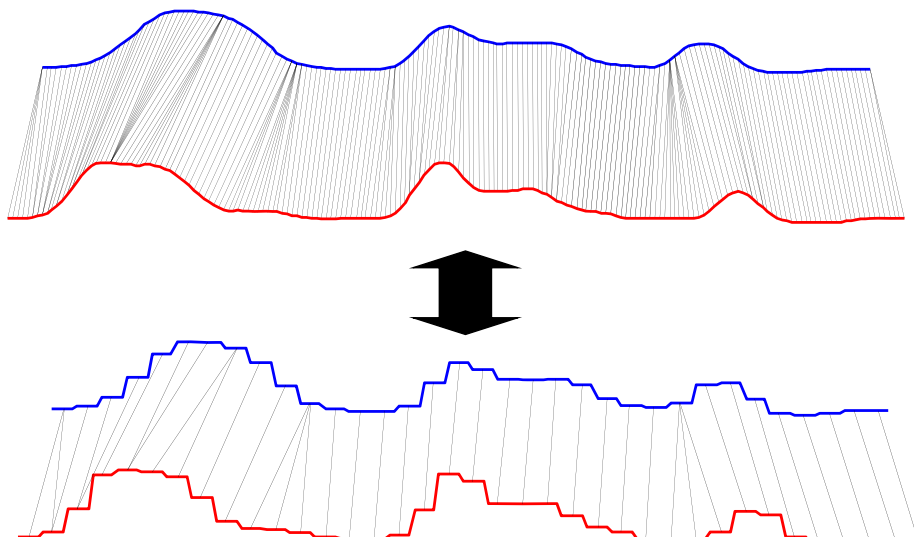


43

Approximative Dynamic Time Warping Distanz

Idee:

- Approximiere die Zeitreihen
(komprimierte Repräsentation, Sampling, ...)
- Berechne DTW auf den Approximationen



44

Statische Modelle für die Zeit

Problem:

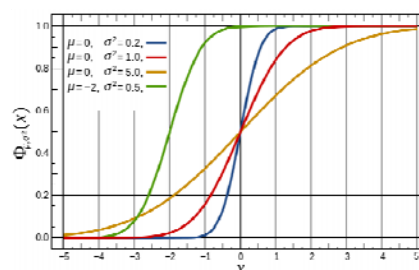
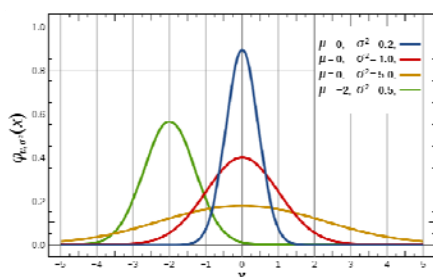
Wie kann man den Zeitpunkt der nächsten Aktion modellieren?

⇒ Statistische Modelle für die Zeit zwischen 2 Ereignissen werden benötigt.

⇒ Zeit ist eine kontinuierliche Variable ⇒ Wahrscheinlichkeitsdichtefunktion

⇒ Gesucht: Wahrscheinlichkeit, dass das nächste Ereignis e im Zeitraum $t+\Delta t$ stattfindet.

⇒ Beschreibung über kumulierte Wahrscheinlichkeitsdichtefunktion

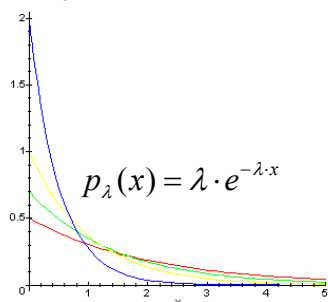


45

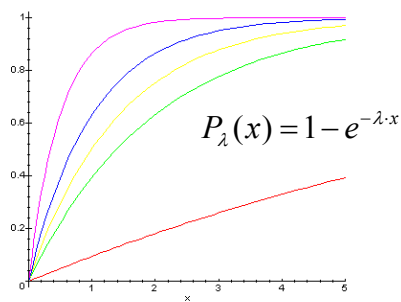
Homogene Poisson Prozesse

- Einfachster Prozess zur Modellierung von Zeit.
- Zeitpunkte zwischen 2 Ereignissen sind exponentiell verteilt.
Wahrscheinlichkeitsdichte der Exponentialverteilung: $p_\lambda(x) = \lambda \cdot e^{-\lambda \cdot x}$
- Durch Integration erhält man die kumulierte Dichtefunktion, die die Wahrscheinlichkeit beschreibt, dass die nächste Aktion im Zeitintervall zwischen 0 ... x stattfinden wird.

$$P_\lambda(x) = \int_0^x p_\lambda(t) dt = 1 - e^{-\lambda \cdot x}$$



Dichtefunktion der Exponential-Verteilung



kumulierte Dichtefunktion der Exponential-Verteilung

46

Parameterschätzung

Gegeben: Eine Trainingsmenge aus Zeitpunkten $X = \{x_1, \dots, x_n\}$, die exponentialverteilt sind.

Gesucht: Der wahrscheinlichste Wert für den Intensitätsparameter λ .

Abschätzung über Maximum Likelihood

=> Suche den Wert für λ , bei dem die Wahrscheinlichkeit der Generierung von X am höchsten ist. Likelihood-Funktion L für Sample X :

$$L_X(\lambda) = \prod_{i=1}^n \lambda \cdot e^{-\lambda \cdot x_i} = \lambda^n \cdot e^{-\lambda \cdot \sum_{i=1}^n x_i} = \lambda^n \cdot e^{-\lambda \cdot n \cdot E(X)} \quad \text{mit } E(X) = \frac{\sum_{i=1}^n x_i}{n}$$

Ableiten der logarithmischen Likelihood nach λ und bestimmen der Nullstellen:

$$\frac{d}{d\lambda} \ln L(\lambda) = \frac{d}{d\lambda} (n \cdot \ln L(\lambda) - \lambda \cdot n \cdot E(X)) = \frac{n}{\lambda} - n \cdot E(X)$$
$$\Rightarrow \lambda^* = \frac{1}{E(X)}$$

47

Lernziele

- Sequenzen und Zeitreihen
- Frequent Subsequenz Mining mit Suffix-Bäumen
- Distanzmaße auf Sequenzen
 - Hamming Distanz
 - Levenshtein Distanz
- Markow-Ketten
- Hidden Markow Ketten
- Zeitreihen und Vorverarbeitungsschritte
- Dynamic Time Warping
- Poisson Prozesse

48

Literatur

- Kyong Jin Shim, Jaideep Srivastava: **Sequence Alignment Based Analysis of Player Behavior in Massively Multiplayer Online Role-Playing Games (MMORPGs)**, in Proceedings of the 2010 IEEE International Conference on Data Mining Workshops, 2010.
- Ben G. Weber, Michael Mateas: **A data mining approach to strategy prediction**, in Proceedings of the 5th International Conference on Computational Intelligence and Games, 2009.
- K.T. Chen, J.W. Jiang, P. Huang, H.H. Chu, C.L. Lei, W.C. Chen: **Identifying MMORPG bots: A traffic analysis approach**, In Proceedings of the 2006 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology, 2006.