**UNIVERSITÄT DUISBURG-ESSEN**
FAKULTÄT FÜR INGENIEURWISSENSCHAFTEN
ABTEILUNG INFORMATIK UND ANGEWANDTE KOGNITIONSWISSENSCHAFT

**Bachelorarbeit**

# Fancy thesis template

Vorname Nachname
Matrikelnummer: 0123456
Angewandte Informatik (Bachelor)

UNIVERSITÄT
DUISBURG
ESSEN

Fachgebiet Verteilte Systeme, Abteilung Informatik
Fakultät für Ingenieurwissenschaften
Universität Duisburg-Essen

3. März 2023

**Erstgutachter:** Prof. Dr-Ing. Torben Weis
**Zweitgutachter:** Hier Zweitgutachter eintragen
**Zeitraum:** 1. September 2042 - 1. Januar 2043

# Abstract

An abstract is a brief summary of a research article, thesis, review, conference proceeding, or any in-depth analysis of a particular subject and is often used to help the reader quickly ascertain the paper's purpose. When used, an abstract always appears at the beginning of a manuscript or typescript, acting as the point-of-entry for any given academic paper or patent application. Abstracting and indexing services for various academic disciplines are aimed at compiling a body of literature for that particular subject. [1]

[1] Wikipedia: `https://en.wikipedia.org/wiki/Abstract_(summary)`

# Contents

# Chapter 1

# Grundlagen

## 1.1 Grundbegriffe der Stochastik

### 1.1.1 Zufallsexperimente und Elementarereignisse

Der Grundbaustein der Stochastik sind sogenannte **Zufallsexperimente**. Klassische Beispiele für Zufallsexperimente sind das werfen eines fairen 6-seitigen Wüfels oder das ziehen von Kugeln aus einer Urne. Das Ergebnis eines Zufallsexperimentes nennt man **Elementarereignis** $w$. Die Menge aller Möglichen Elementarereignisse ist $\Omega$. Wenn unser Zufallsexperiment das einmalige Werfen eines fairen Würfels ist, dann gilt $\Omega = \{1, 2, 3, 4, 5, 6\}$ und für das einmalige Werfen zweier fairer Würfel gilt $\Omega = \{1, 2, 3, 4, 5, 6\} \times \{1, 2, 3, 4, 5, 6\}$.

### 1.1.2 Zufallsereignisse

Ein **Ereignis** $E$ ist eine Teilmenge von $\Omega$. Zum Beispiel ist $E = \{6\}$ das Ereigniss, dass die Zahl 6 gewürfelt wird und $E = \{2, 4, 6\}$ das Ereignis, dass eine gerade Zahl fällt. Die Menge aller möglichen Ereignisse auf einer Menge von Elementarereignissen ist die **Ereignisalgebra** $\mathcal{E}$. Sie entspricht der Potenzmenge der Elementarereignisse $\mathcal{E} = \mathcal{P}(\Omega)$.

### 1.1.3 Wahrscheinlichkeiten

Die Wahrscheinlichkeit eines Ereignisses $E \in \mathcal{E}$ ist gegeben durch ein **Wahrscheinlichkeitsmaß** $P : \mathcal{E} \to [0, 1]$. Zufallsereignisse können durch **Mengenoperationen** kombiniert werden. So ist die Wahrscheilichkeit für das gemeinsame Eintreten von $A$ und $B$ gegeben durch die Vereinigung der Mengen $P(A \cap B)$ und die Wahrscheinlichkeit

für das Eintreten von $A$ oder $B$ gegeben durch die Vereinigung $P(A \cup B)$. Zwei Zufallsereignisse $A$ und $B$ sind voneinander **unabhängig** wenn gilt

$$P(A \cap B) = P(A) \cdot P(B) \tag{1.1}$$

Für Zufallsereignisse die nicht unabhängig sind können wir die **bedingte Wahrscheinlichket** berechnen. Die Wahrscheinlichkeit von $A$ abhängig von $B$ wird notiert mit $P(A \mid B)$ und ist definiert als:

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)} \tag{1.2}$$

Die Wahrscheinlichkeit von $A$ lässt sich nur berechnen falls die Wahrscheinlichkeit von $B$ **wohldefiniert** ist, also $P(B) > 0$.

### 1.1.4 Wahrscheinlichkeitsräume

Eine Menge an Elementarereignissen $\Omega$, die auf $\Omega$ definierte Ereignisalgebra $\mathcal{E}$ und das Wahrscheinlichkeitsmaß $P : \mathcal{E} \to [0, 1]$ bilden zusammen einen **Wahrscheinlichkeitsraum** $(\Omega, \mathcal{E}, P)$. Im folgenden beschränken wir uns auf diskrete Wahrscheinlichkeitsräume, welche dadurch gekennzeichnet sind, dass $\Omega$ abzählbar endlich ist [1].

### 1.1.5 Zufallsvariablen

Eine **Zufallsvariable** $X$ ist eine Abbildung von $\Omega$ in eine beliebige Menge $C$ [1]. Sei zum Beispiel $\Omega$ die Menge aller möglichen Ergebnisse des Werfens zweier fairer Würfel. Dann ist die Augensumme der Würfel gegeben durch die Abbildung $X : \Omega \to [0, 1]$ mit $X(i, j) := i + j$ für alle $(i, j) = w \in \Omega$. Die Wahrscheinlichkeit, dass die Augensumme 3 ergibt lässt sich dann berechnen mit

$$P(X = 3) = P(\{(1, 2), (2, 1)\}) = P((1, 2)) + P((2, 1)) = \frac{1}{36} + \frac{1}{36} = \frac{1}{18}$$

Analog zu Zufallsereignissen sind Zufallsvariablen unabhängig wenn gilt

$$P(X_1 = x_1 \mid X_2 = x_2, \dots X_n = x_n) = P(X_1 = x_1) \cdot P(X_2 = x_2) \cdot \dots \cdot P(X_n = x_n) \tag{1.3}$$

### 1.1.6 Stochastische Prozesse

Ein Stochastischer Prozess beschreibt ein System welches sich zu einem gegebenen Zeitpunk $t$ in einem von endlich vielen Zuständen $S = \{s_1, s_2, \dots, s_n\}$ befinden kann, wobei das Verhalten des Systems durch Zufallsereignisse bestimmt wird. Formaler ausgedrückt ist ein stochastischer Prozess ist eine Menge an Zufallsvariablen $\{X_t\}_{t \in T}$, indiziert durch $T$ [12]. Die Zufallsvariablen sind eine Abbildung in den Zustandsraum $S$ [4].$X_t -> S = \{s_1, s_2, \dots, s_n\}$

## 1.2 Markovprozesse und Markov Modelle

### 1.2.1 Markovketten

Eine Markovkette ist ein stochastischer Prozess in welchem der nächste Zustand einzig vom gegenwärtigen Zustand abhängt. Diese Eigenschaft ist bekannt als Markov-Eigneschaft [12].

**Definition 1 (Markovkette)** *Ein Stochastischer Prozess $\{X_t\}_{t\in T}$, welcher Werte aus einer Menge an Zuständen $S = \{s_1, s_2, \ldots, s_n\}$ annehmen kann ist ein Markovprozess, falls gilt*

$$P(X_{n+1} = s_{n+1} \mid X_n = i_n, \ldots, X_2 = i_2, X_1 = i_1) = P(X_{n+1} = i_{n+1} \mid X_n = i_n) \quad (1.4)$$

*Für alle $n \in N_0$ und alle $i_k \in S$ unter der Vorraussetzung, dass alle bedingten Wahrscheinlichkeiten Wohldefiniert sind, also $P(X_n = i_n, \ldots, X_2 = i_2, X_1 = i_1) > 0$.*

Für solch eine Markovkette definieren wir nun die zwei folgenden Variablen.

**Definition 2 (Startwahrscheinlichkeitsvektor *pi*)** *$\pi$ ist ein Vektor mit Länge $n = |S|$. $\pi_i$ gibt die Wahrscheinlichkeit an, in Zustand $s_i$ zu starten. Also die Wahrscheinlichkeit, dass sich der Prozess in Zeitpunkt $t = 0$ in Zustand $s_i$ befindet.*

$$\pi_i = P(X_0 = s_i) \quad (1.5)$$

**Definition 3 (Transitionsmatrix *A*)** *Die Transitionsmatrix A beschreibt die bedingten Wahrscheinlichkeiten, mit denen ein Zustandsübergang geschieht. $a_i(j)$ ist die Wahrscheinlichkeit dass sich der Prozess in Zeitpunkt $t+1$ in Zustand $s_j$ befindet, unter der Bedingung das er sich in Zeitpunk $t$ in Zustand $s_i$ befindet.*

$$a_{ij} = P(X_{t+1} = s_j \mid X_t = s_i) \quad (1.6)$$

Eine Markovkette ist **zeithomogen** wenn die Transitionswahrscheinlichkeiten $a_{ij}$ unabhängig von der Zeit $t$ sind [5].

$$a_{ij} = P(X_{t+1} = s_j \mid X_t = s_i) = P(X_t = s_j \mid X_{t-1} = s_i) \quad (1.7)$$

In dieser Arbeit werden ausschließlich zeithomogene Markovketten betrachtet.

Das Ereignis in irgendeinem Zustand zu starten, so wie das Ereignis von einem gegebenen Zustand in irgendeinen anderen Zustand zu wechseln sind sichere Ereignisse. Somit gelten die Bedingungen, dass die Werte des Startwahrscheinlichkeitsvektors $\pi$ und die

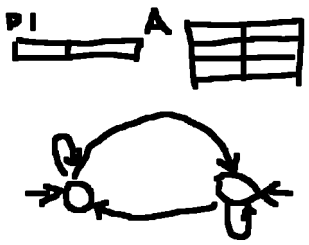
Werte der ausgehenden Transitionen für jeden Zustand $A_i$ in Summe jeweils 1 ergeben müssen. Diese Bedingung nennt man **Reihenstochastizität**

$$\sum_{i=0}^{N} \pi_i = 1 \qquad\qquad \sum_{j=0}^{N} a_{ij} = 1 \tag{1.8}$$

Als Beispiel für eine zeitdiskrete homogene Markovkette werden wir die durchschnittliche jährliche Temperatur modellieren. Seien die Zustände des Modells gegeben durch $S = \{H, K\}$, wobei $H$ für heiß und $K$ für kalt steht. Der Startvektor $\pi$ und die Transitionsmatrix $A$ seien gegeben durch

$$\pi = (0.7, 0.3) \qquad\qquad A = \begin{bmatrix} 0.8 & 0.2 \\ 0.4 & 0.6 \end{bmatrix} \tag{1.9}$$

Wir können diese Markovkette anschaulich als gerichteten Transitionsgraphen representieren.




Mit unserem Temperaturmodell können wir nun zum Beispiel berechnen, was die Wahrscheinlichkeit ist die Temperaturabfolge $O = \{H, K, K, H, H\}$ zu beobachten.

$$\begin{aligned} P(O) &= P(O_1 = H) \cdot P(O_2 = K \mid O_1 = H) \cdot P(O_3 = K \mid O_2 = K) \\ &\quad \cdot P(O_4 = H \mid O_3 = K) \cdot P(O_5 = H \mid O_4 = H) \\ &= \pi_1 \cdot a_{12} \cdot a_{22} \cdot a_{21} \cdot a_{11} \\ &= 0.7 \cdot 0.2 \cdot 0.6 \cdot 0.4 \cdot 0.8 = 0.02688 \end{aligned}$$

Eine Markovkette gibt uns also die möglichkeit probabilistische Systeme, in denen die Zustände zu den Observationen korrespondieren zu modellieren.

### 1.2.2 Hidden Markov Modelle

Bei einem Hidden Markov Modell ist die Observationssequenz nicht identisch zur Zustandssequenz, sondern die Observationssymbole sind das Ergebnis einer stochastischen Funktion der Zustände. Die Zustände selbst können wir nicht beobachten denn sie sind

"hidden". Ein HMM besteht also aus zwei stochastischen Prozessen. Ein versteckter Prozess, der die Transitionen in Abhängigkeit des vorherigen Zustandes bestimmt und ein zweiter Prozess, der die Observationssymbole in Abhängigkeit des gegenwärtigen Zustandes bestimmt. Dieser zweite Prozess wird beschrieben durch eine Emissionsmatrix $B$.

**Definition 4 (Emissionsmatrix $B = \{b_j(k)\}$)** *Die Emissionsmatrix $B$ gibt die Wahrscheinlichkeit eines Observationssymbols $v_k$ unter der Bedingung, dass sich der Prozess im gegenwärtigen Zeitpunkt $t$ in Zustand $s_j$ befindet an.*

$$b_j(k) = P(O_t = v_k \mid q_t = S_j) \qquad 1 \leq j \leq N \qquad 1 \leq k \leq M \tag{1.10}$$

Wie ein solches HMM nun eine Observationssequenz $O$ von Länge $T$ erzeugen kann möchte ich durch folgendes Gedankenexperiment verdeutlichen.

Stellen wir uns einen Raum vor, in dem $N$ Urnen stehen und in welchem sich eine Person befindet. Wir können in diesen Raum nicht hineinsehen. Die Person wird nun insgesamt $T$ Kugeln aus den Urnen ziehen und zurücklegen, wobei sie sich abhängig von einem Zufallsprozess von einer Urne zur nächsten bewegt. Um ethische Bedenken über dieses Gedankenexperiment zu minimieren sei angemerkt, dass der Raum klimatisiert ist und der Person Kakao und Kekse zur Verfügung stehen.

- Zunächst wird die initiale Urne anhand des Startwahrscheinlichkeitsvektor $\pi$ ausgewählt.
- Dann zieht die Person eine Kugel aus der Urne, notiert deren Farbe als $O_1$ und legt die Kugel zurück.
- Danach wird eine neue Urne $q_2 = S_j$ mit Wahrscheinlichkeit $a_{ij} = P(q_2 = S_j \mid q_1 = S_1)$ gewählt. Aus dieser neuen Urne wird die nächste Kugel $O_2$ der Farbe $v_k$ mit Wahrscheinlichkeit $b_j(k)$ gezogen.

Diese Prozedur wird so oft wiederholt bis der Zeitpunkt $t = T$ erreicht ist. Wir erhalten somit eine Observationssequenz von Farben $O = \{O_1, O_2, \ldots, O_T\}$. Abbildung 1.1 zeigt eine Visualisierung dieses Gedankenexperiments für $N = 3$

## 1.3 Die drei klassichen Probleme

Für ein gegebenes Hidden Markov Model $\lambda = (A, B, \pi)$ und eine Observationssequenz $O$ gibt es drei Probleme, welche von besonderem Interesse sind.

- **Problem 1** Was ist die Wahrscheinlichkeit, dass die Observationssequenz $O$ von Modell $\lambda$ erzeugt wurde $P(O \mid \lambda)$, und wie können wir diese effizient berechnen?

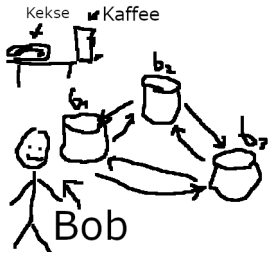



Figure 1.1: Urn Ball Model

- **Problem 2** Welche Zustandssequenz $Q = q_1, q_2, \ldots, q_T$ hat die größte Wahrscheinlichkeit die Observationssequenz $O = O_1, O_2, \ldots, O_T$ zu erzeugen $\text{argmax}_Q\ P(Q \mid O, \lambda)$?
- **Problem 3** Wie können wir die Parameter des Modells $\lambda$ anpassen um die Wahrscheilichkeit der Observationssequenz $O$ zu maximieren? Anders formuliert suchen wir ein $\lambda'$, so dass $P(O \mid \lambda') > P(O \mid \lambda)$

Auf diese Probleme wird jeweils in den folgenden drei Abschnitten eingegangen.

### 1.3.1 Berechnen der Wahrscheinlichkeit einer Observationssequenz

Bevor wir das Problem angehen wie man $P(O \mid \lambda)$ berechnet befassen wir uns zunächst damit wie man $P(O \mid Q, \lambda)$, also die Wahrscheinlichkeit von $O$ wenn die Zustandssequenz $Q = q_1, q_2, \ldots, q_T$ gegeben ist, berechnen kann. Da wir in diesem Fall für jeden Zeitpunkt $t$ das beobachtete Symbol $O_t$ und den emittierenden Zustand $q_t$ kennen ergibt sich die Berechnung von $P(O \mid Q, \lambda)$ aus der Definition von $B$.

$$P(O \mid Q, \lambda) = \prod_{t=1}^{T} b_{q_t}(O_t)$$

Ein intuitiver Ansatz $P(O \mid \lambda)$ zu Berechnen wäre nun alle Möglichen Zustandssequenzen $Q$ aufzuzählen und die Wahrscheinlichkeiten $P(O \mid Q, \lambda)$ zu summieren.

$$P(O \mid \lambda) = \sum_{\text{alle Q}} P(O \mid Q, \lambda)$$

Es ist zwar möglich $P(O \mid \lambda)$ so zu berechnen, jedoch wächst die Anzahl der Möglichen Zustandsabfolgen $Q$ exponentiell in Abhängigkeit zur Länge $T$ von $O$. Insgesamt wären $2T \cdot N^T$ Berechnungen notwendig, was selbst bei kleinen Werten für $N$ und $T$ einen untragbaren Rechenaufwand darstellt. Für N=5 und T=100 wären es $2 \cdot 100 \cdot 5^{100} \approx 10^{72}$ Berechnungen. Um diese Zahl in Relation zu stellen, $10^{72}$ ist mindestens 3 mal mehr als 1000 und 1000 ist schon ziemlich groß. Zum Glück gibt es aber eine effiziente Methode um $P(O \mid \lambda)$ zu berechnen. Die Forwärtsvariable.

#### Der Forwärtsalgorithmus

Sei $\alpha$, die Forwärtsvariable folgendermaßen definiert

$$\alpha_t(i) = P(O_1 O_2 \dots O_t \wedge q_t = S_i \mid \lambda)$$

$\alpha_t(i)$ Beschreibt also die Wahrscheinlichkeit, die partielle Observationssequenz $O_1 O_2 \dots O_t$ zu beobachten und in Zeitpunkt $t$ in in Zustand $i$ zu sein. $\alpha_t(i)$ kann folgendermaßen induktiv berechnet werden

**Für** $t = 0$ lässt sich $alpha_t(i)$ aus $\pi$ und $B$ berechnen, da noch keine Transition stattgefunden hat.

$$\alpha_0(i) = \pi_i \cdot b_i(O_0)$$

**Für** $t > 0$ gilt

$$\alpha_{t+1}(j) = \left[ \sum_{i=1}^{N} \alpha_t(i) \cdot a_{i,j} \right] \cdot b_j(O_{t+1})$$

Der Casus Knacksus, welcher eine effiziente Berechung von $\alpha$ ermöglicht ist die Markov-Eigenschaft. Diese besagt, dass der Zustand in Zeitpunkt $t + 1$ nur vom Zustand in Zeitpunkt $t$ abhängt. Für jeden Zeitpunkt und jeden Zustand gibt es also genau $N$ Zustände in denen sich der Prozess zuvor befunden haben kann. Da der Rechenaufwand für jeden Zeitpunkt gleich ist hängt die Komplexität von *alpha* mit $N^2 \cdot T$ nur noch linear von $T$ ab und nicht exponentiell wie in dem vorherigen Ansatz.

### 1.3.2 Der Viterbi Algorithmus

Um die wahrscheinlichste Zustandssequenz zu berechnen müssen wir zunächst eine weitere Variable einführen, die Rückwärtsvariable $\beta$.

**Definition 5 (Rückwärtsvariable $\beta$)** *Die Rückwärtsvariable beta funktioniert analog zu $\alpha$ und ist wie folgt definiert.*

$$\beta_t(i) = P(O_{t+1}, O_{t+2} \dots O_T \mid q_t = S_i, \lambda)$$

*Somit ist $\beta_t(i)$ die Wahrscheinlichkeit in Zeitpunkt t in Zustand $S_i$ zu sein und außgehend von $S_i$ die partielle Observationssequenz $O_{t+1}, O_{t+2} \dots O_T$ zu beobachten.*

***Für*** $t = T$ *gilt*

$$\forall_i \; 1 \leq i \leq N \mid \beta_T(i) = 1$$

***Für*** $t < T$ *gilt*

$$\beta_t(i) = \sum_{j=1}^{N} a_{i,j} \cdot b_j(O_{t+1}) \cdot \beta_{t+1}(j)$$

Mit den Variablen $\alpha$ und $\beta$ gewappnet können wir nun eine weitere Variable definieren.

$$\gamma_t(i) = P(q_t = S_i \mid O, \lambda)$$

Die Variable $\gamma$ beschreibt die Wahrscheinlichkeit beim Beobachten von O in Zeitpunkt $t$ in Zustand $i$ zu sein. $\gamma_t(i)$ wird berechnet durch

$$\gamma_t(i) = \frac{\alpha_t(i) \cdot \beta_t(i)}{P(O \mid \lambda)} = \frac{\alpha_t(i) \cdot \beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i) \cdot \beta_t(i)}$$

Der Viterbi Algorithmus ähnelt dem Forwärtsalgorithmus sehr stark. Der maßgebliche Unterschied liegt jedoch darin, dass der Viterbi Algorithmus das **Maximum** der vorherigen Pfadwahrscheinlichkeiten betrachtet, wohingegen der Forwärtsalgorithmus die **Summe** betrachtet. Um die Wahrscheinlichkeit einer Observationssequenz zu berechnen ist es egal welcher Pfad

- Wir müssen uns nur den Pfad angucken, welcher am Wahrscheinlichsten in einen Zustand führt

Sei $\delta_t(i)$ die maximale Wahrscheinlichkeit, aller zusammenhängenden Pfade welche die ersten $t$ Observationen erklären und in Zustand $s_i$ enden.

$$\begin{aligned}
\delta_t(i) &= \max_{q_1, q_2, \ldots, q_{t-1}} P(q_1 q_2 \ldots q_{t-1} \wedge q_t = i \wedge O_1 O_2 \ldots O_t \mid \lambda) \\
\delta_{t+1}(i) &= (\max_i \delta_t(i) \cdot a_{ij}) \cdot b_j(O_{t+1})
\end{aligned}$$

Damit wir die wahrscheinlichste Zustandssequenz ausgeben können müssen wir uns für jeden Zeitpunkt $t$ und jeden Zustand $s_i$ das Argument, $j$ welches $\delta_{t-1}(j)aij$ maximiert merken. Dies geschieht mittels des Arrays $\psi_t(i)$. Der ganze Lachs kann nun wie folgt gebutter werden.

- 1) Initialisierung

$$\begin{aligned}
\delta_1(i) &= \pi_i b_i(O_1) && 1 \leq i \leq N \\
\psi_1(i) &= 0
\end{aligned}$$

- 2) Rekursion

$$\begin{aligned}
\delta_t(j) &= \max_{1 \leq i \leq N} (\sigma_{t-1}(i) a_{ij}) \cdot b_j(O_t) && 2 \leq t \leq T, 1 \leq j \leq N \\
\psi_t(j) &= \operatorname*{argmax}_{1 \leq i \leq N} (\delta_{t-1}(i) a_{ij}) && 2 \leq t \leq T, 1 \leq j \leq N
\end{aligned}$$

- 3) Terminierung

$$P^* = \max_{1 \leq i \leq N} [\delta_T(i)]$$
$$q_T^* = \operatorname*{argmax}_{1 \leq i \leq N} [\delta_T(i)]$$

- 4) Zustandssequenzrückverfolgung

$$q_t^* = \psi_{t+1}(q_{t+1}^*) \qquad t = T-1, T-2, \ldots, 1$$

### Baum-Welch Verfahren

Die letzte Variable, welche wir für den Baum-Welch Algorithmus benötigen ist $\xi_t(i,j)$, die vorwärts-rückwarts-Variable. Diese gibt die Wahrscheinlichkeit an in Zeitpunkt $t$ in Zustand $S_i$ zu sein und im nächsten Zeitpunk $t+1$ in Zustand $S_j$ zu sein.

$$\xi_t(i,j) = \frac{\alpha_t(i) \cdot a_{i,j} \cdot b_j(O_{t+1}) \cdot \beta_{t+1}(j)}{P(O \mid \lambda)} = \frac{\alpha_t(i) \cdot a_{i,j} \cdot b_j(O_{t+1}) \cdot \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) \cdot a_{i,j} \cdot b_j(O_{t+1}) \cdot \beta_{t+1}(j)} \tag{1.11}$$

## 1.4 Baum-Welch Algorithmus

Der Baum Welch Algorithmus ist ein erwartungsmaximierender Ansatz. Er basiert auf dem Prinzip, dass wir aus den Variablen $\xi$ und $\gamma$ Erwartungswerte für die Anzahl an Transitionsübergängen und Symbolemissionen berechnen können. Mit dem Wissen wie oft diese Ereignesse erwartungsgemäß eintreffen können wir die Parameter des Models $\lambda = \pi, B, A$ neu berechnen. Wir erinnern uns

$\gamma_t(i)$ = Wahrscheinlichkeit in Zeitpunkt $t$ in Zustand $S_i$ zu sein

$\xi_t(i,j)$ = Wahrscheinlichkeit in Zeitpunkt $t$ in Zustand $S_i$ und in $t+1$ in $S_j$ zu sein

Wenn wir $gamma_t(i)$ über alle $t$ exklusive $t = T$ summieren erhalten wir die zu erwartende Anzahl an ausgehenden Transitionen von $S_i$

$\sum_{t=1}^{T-1} \gamma_t(i)$ = Zu erwartende Anzahl der von $S_i$ ausgehenden Transitionen

Summieren wir $gamma_t(i)$ über alle $t$ inklusive $t = T$ erhalten wir die zu erwartende Anzahl an Aufenthalten in Zustand $S_i$

$\sum_{t=1}^{T} \gamma_t(i)$ = Zu erwartende Anzahl an Aufenthalten in $S_i$

Die zu erwartende Anzahl der Transitionen von $S_i$ zu einem bestimmten Zustand $S_j$ erhalten wir ähnlich, durch summieren von $\xi_t(i,j)$ über alle $0 < t < T$.

$\sum_{t=1}^{T-1} \xi_t(i,j)$ = Zu erwartende Anzahl an Transitionen von $S_i$ nach $S_j$

Mit den Erwartungswerten für Anzahlen an Zustandsübergängen, Zustandsaufenthalten und Zustandsemissionen können wir neue Werte für die Parameter des HMM berechnen.

$$\pi_i = \gamma_1(i) b_{i,j} = \frac{\sum_{t=1 s.d. O_t = V_k}^{T} \gamma_t(i)}{\sum_{t=1}^{T} \gamma_t(i)} a_{i,j} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

Eine Neuberechnung der Parameter kann beliebig oft durchgeführt werden. In der Praxis wird als Abbruchbedingung ein Mindestwert für die Verbesserung des Models festgelegt.

## 1.5 Optimierungsverfahren

Im vorherigen Abschnitt haben wir uns unter anderem damit befasst wie die Parameter eines HMMs mittels des Baum-Welch Algorithmus optimiert werden können. In diesem Kapitel treten wir einen Schritt zurück und beantworten die Fragen was es überhaupt bedeutet Dinge zu optimieren und welche verschiedenen Optimierungsansätze existieren.

### 1.5.1 Was ist Optimierung

Wir sprechen von einem **Optimierungsproblem** wenn wir optimale Parameter eines Systems bestimmen wollen. Als Optimale Parameter bezeichnen wir solche, die eine Problemspezifische **Zielfunktion** $f$ minimieren (oder maximieren). Die Domäne der Zielfunktion nennt man einen **Suchraum** $S$. Der Wert der Zielfunktion für eine Kombination von Parametern aus dem Suchraum gibt die **Qualität** dieser Parameter an. Oft gibt es Einschränkungen für die Parameter, so dass nicht alle Werte des Suchraumes mögliche Lösungen sind. Den Raum aller möglichen Lösungen nennen wir **zulässige Region** [10].

Um diese Konzepte zu verdeutlichen betrachten wir das bekannte Problem des Handlungsreisenden (traveling salesman problem), in welchem es darum geht den Besuch mehrerer Städte optimal zu planen, so dass jede Stadt außer dem Starpunkt nur einmal besucht wird und der Endpunkt equivalent zu dem Startpunkt ist. Der Suchraum ist in diesem Fall gegeben durch alle möglichen Routen (Kombinationen der zu besuchenden Städte). Die Zielfunktion, welche wir minimieren wollen gibt die gesamte Länge einer Reiseroute an. In der zulässigen Region des Suchraumes befinden sich ausschließlich Routen in welchen jede Stadt, bis auf den Startpunkt, einmal besucht wird und der Endpunkt equivalent zum Startpunkt ist. Da Teleportation noch nicht erfunden wurde ist die zulässige Region zusätzlich eingeschränkt auf alle Routen in welchen nacheinander besuchte Städte durch eine Strecke verbunden sind.

### 1.5.2 Kategorien von Optimierungsverfahren

Eine Optimierungsmethode, welche stets die global optimale Lösung findet nennt man eine **exakte Methode**. Für viele Probleme gibt es jedoch keine exakten Algorithmen die eine Lösung in polynomieller Zeit finden. In solch einem Fall kann man zu einer **Heuristik** greifen. Eine Heuristik liefert eine Lösung die "gut genug" ist in "annehmbarer Zeit". Es handelt sich also um ein Verfahren, welches man umgangssprachlich als Faustregel bezeichnen würde. Eine Heuristik ist nicht das selbe wie ein **Approximationsalgorithmus**. Denn ein Approximationsalgorithmus garantiert eine untere Schranke für die Qualität einer gefundenen Lösung. Bei einer Heuristik verhält es sich ähnlich wie mit Privatkäufen über Kleinanzeigenportale: Es besteht keine Garantie. Die gefundenen Lösungen eines heuristischen Verfahrens können also beliebig schlecht sein. Heuristiken werden unterteilt in **spezifische Heuristiken** und **Metaheuristiken**. Spezifische Heuristiken sind, wie der Name bereits vermuten lässt zugeschnitten auf ein spezifisches Problem, wohingegen Metaheuristiken sehr allgemein sind und auf fast alle Optimierungsprobleme angewendet werden können [10]. Metaheuristiken kann man weiter unterteilen in

Die wahrscheinlich bekannteste Heuristik ist eine **lokale Suche**. Der Baum-Welch Algorithmus zum Beispiel ist eine lokale Suche. Genetische Algorithmen, mit welchen wir uns später im Detail beschäftigen zählen zu den bekanntesten und ältesten Metaheuristiken.

Figur 1.2 zeigt die zuvor beschriebene Unterteilung der Optimierungsverfahren. Es sei angemerkt, dass die unternommene Unterteilung keineswegs Anspruch auf Vollständigkeit erhebt, sondern primär zur Einordnung des Baum-Welch Algorithmus und des genetischen Algorithmus im Kontext der Optimierungsverfahren dient.

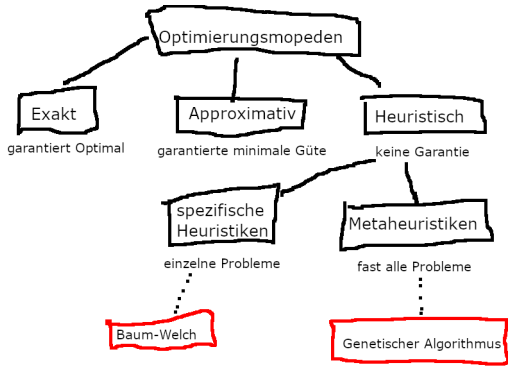



Figure 1.2: Unterteilung der Optimierungsverfahren

### 1.5.3 Metaheuristiken

Bei einer Metaheuristik handelt es sich nicht um einen bestimmten Algorithmus sondern vielmehr eine Ansammlung von Ideen, Konzepten und Operatoren, welche verwendet werden können um einen spezifischen heuristischen Algorithmus zu erstellen [8]. Es gibt also zum Beispiel nicht *den* genetischen Algorithmus sondern es existiert ein Genetischer Algorithmus "Bauplan" an welchem man sich orientieren kann.

Um eine Metaheuristik zu erstellen gilt es zwei konfliktierende Kriterien zu balancieren. Zum einen die **Diversifizierung** (exploration): das entdecken neuer Lösungen und zum anderen die **Intensivierung** (exploitation): das verbessern einer bekannten Lösung [10]. Reine Diversifizierung ist das selbe wie eine zufällige Suche und reine Intensivierung ist equivalent zu einer lokalen Suche.

Metaheuristiken sind oft inspiriert durch natürliche Prozesse. Bekannte Metaheuristiken aus dem Gebiet der Biologie sind **Genetische Algorithmen (GA)**, welche die Evolution einer Population durch natürliche Selektion nachahmen und **Ant Colony Optimization (ACO)** Algorithmen, welche der Pheromon-basierten Kommunikation von Ameisen innerhalb einer Kolonie nachempfunden sind. Zu den bekanntesten Beispielen aus der Physik zählt das **Simulated Annealing (SA)** Verfahren, welches angelehnt ist an den Abkühlungsprozess eines Metalls nach dem erhitzen [9].

### 1.5.4 Hybride Heuristiken

Eine kombination mehrerer verschiedener Algorithmen zu einem neuen nennt man Hybridisierung. Das kombinieren von metaheuristischen Algorithmen ist ein aktives Forschungsfeld und es existiert mittlerweile eine beachtliche Anzahl hybrider Metaheuristiken. Das kombinieren von Konzepten erlaubt es Wissenschaftlern außerhalb der Grenzen einer bestimmten Metaheuristik zu denken und ihrer Kreativität freien Lauf zu lassen [2].

Es gibt verschiedene möglichkeiten der Hybridisierung. Wenn man verschiedene Heuristiken sequentiell hintereinander ausführt spricht man von einer **Relay Hybridization (RH)**. Eine Hybridisierung in der verschiedene Heuristiken kooperativ zusammenarbeiten nennt man eine **Teamwork Hybridization (TH)** [10].

Die Hauptmotivation hinter der hybridisierung ist es komplementäre Eigenschaften verschiedener Algorithmen auszunutzen [2]. Populationsbasierte Metaheuristiken, wie zum Beispiel genetische Algorithmen sind gut im diversifizieren aber schlecht im intensivieren. Eine lokale Suche wiederum ist gut im intensivieren einer Lösung, bietet aber keine Diversifizierung. Es bietet sich also an Eine populationsbasierte Metaheuristik mit einer lokalen Suche zu kombinieren [10].

### 1.5.5 Parameter Tuning

Ein großer Nachteil vom Metaheuristiken ist, dass sie neue Parameter einführen, welche selbst optimiert werden müssen. Beispiele für solche Parameter sind die Mutationsrate eines Genetischen Algorithmus, die initiale Temperatur beim simulated annealing oder auch die Pheromonpersistenz eines Ant Colony Algorithmus. Eine Optimale Belegung dieser Parameter ist Problemabhängig. Daher gibt es keine Metaheuristik für welche universal optimale Parameter existieren. Man unterscheidet zwischen dem **Off-Line** und dem **On-Line** Parameter Tuning. In einem Off-Line Ansatz werden Werte für die Parameter vor der Ausführung des Algorithmus fixiert. Beim On-Line Parameter Tuning werden die Parameter während der Ausführung dynamisch angepasst [10].

### 1.5.6 Kritik an metapher basierten Metaheuristiken

In den letzten Jahren wurde das Feld der Optimierung regelrecht überschwemmt mit "neuen" Metapher basierten Algorithmen. Ob Mikrofledermaus, Jazz-Musiker, Schwarze Löcher oder auch intelligente Wassertropfen. Jedes erdenkliche Konzept kann in einen Optimierungsalgorithmus verwandelt werden. Oft stellt sich jedoch heraus dass das einzige was diese "neuen" Algorithmen zum Feld beitragen eine Umbenennung eines bereits etablierten Algorithmus ist. [3] So ist zum Beispiel Harmony search, ein Suchalgorithmus der auf dem Prinzip von Jazz-Musikern funktioniert nichts weiteres als eine Umbenennung eines speziellen Falles des Genetischen Algorithmus. [11] Trotz mangelnder Innovation verzeichnet eine Suche nach "harmony search" auf Google Scholar laut Weyland im Jahre 2010 586 Einträge. Im Jahre 2023 ist diese Zahl auf stolze 57.500 Einträge gestiegen, wovon 7.840 Einträge nach 2022 erschienen. Die Flut solcher vermeintlich "neuen" Algorithmen ist sehr nachteilig für das Feld der Optimierung, denn die elaborierten Metaphern für bereits existierende Konzepte führen zu Verwirrung und tatsächlich innovative Ansätze werden übersehen. [8]

## 1.6 Genetische Algorithmen

**Evolutionäre Algorithmen** sind stochastische populationsbasierte Metaheuristiken, welche dem Prozess der natürlichen Selektion nachempfunden sind. Die bekanntesten Paradigmen für Evolutionäre Algorithmen sind genetische Algorithmen (GA), evolution strategies (ES), evolutionary programming (EP), und genetic programming (GP) [10]. Im folgenden werde ich zunächst die Begrifflichkeiten, welche mit genetischen Algorithmen assoziiert sind erklären, dann einen Überblick über den Ablauf eines genetischen Algorithmus verschaffen und schließlich auf die einzelnen genetischen Operatoren eingehen und häufige Implementationen dieser aufzeigen.

### genetische Algorithmen

- Wurden erstmals entwickelt durch J. Holland in den 1970 Jahren [missing quote EGT 384]

Ein genetischer Algorithmus arbeitet mit einer genetischen Representation der zu optimierenden Parameter [7]. Die genetische Representation werden wir im folgenden als **Chromosom (Genotyp)** und die originale Representation als **Phenotyp** bezeichnen. Ein Chromosom besteht aus mehreren **Genen** welche zu den Parametern der Zielfunktion korrespondieren. Eine Menge an Chromosomen bezeichnet man als **Population**. Abbildung 1.3 verdeutlicht diesen Sachverhalt.

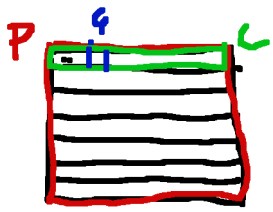



Figure 1.3: Gen, Chromosom und Population

Die **fitness** eines Chromosoms ist der Wert der Zielfunktion für den Phenotyp des Chromosoms. Also der Wert der Zielfunktion für die Parameter welche die Gene des Chromosoms representieren. Der Ablauf eines typischen genetischen Algorithmus besteht aus 4 Schritten.

### 1.6.1 Ablauf eines genetischen Algorithmus

Der Ausgangspunkt eines genetischen Algorithmus ist eine Population an Chromosomen. Das erstellen dieser Population nennt man **Initialisierung**. Gewöhnlicherweise wird die initiale Population zufällig gewählt. Ein Iterationsschritt des klassischen genetischen Algorithmus besteht aus Vier Schritten [6].

- Selektion: Aus der Population wird eine Menge an Eltern gewählt
- Crossover: Die Eltern werden zu einer Menge an Kindern Kindern kombiniert. Typischerweise erzeugen zwei Eltern-Chromosome ein Kind-Chromosom
- Mutation: Die Nachkommen werden mutiert, also zufällig verändert
- Evaluation: Jedem Kind wird ein fitness-Wert durch die fitness-Funktion zugewiesen
- Ersetzung: Die Alte Generation wird durch die neue ersetzt

Der Algorithmus terminiert wenn die vorher spezifizierte Anzahl an Generationen erreicht wurde. Alternative Terminierungskriterien sind zum Beispiel maximale Laufzeit des Algorithmus oder maximale Anzahl an Generationen ohne Verbesserung [7]. In Figur 1.4 wird der beschriebene Ablauf als Flussdiagramm dargestellt.

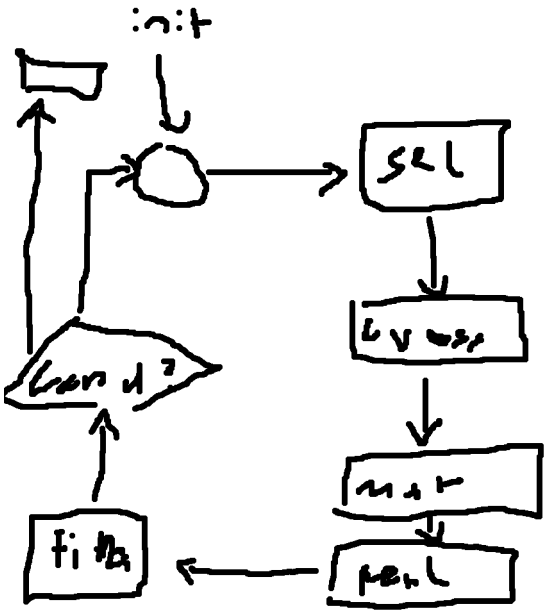


Figure 1.4: Flussdiagram eines genetischen Algorithmus

### 1.6.2 Selektionsoperator

Der Selektionsoperator wählt eine Menge von Eltern aus der Population, aus welchen Kinder für die nächste Generation erzeugt werden. Typischerweise hat jedes Kind-Chromosom zwei Eltern-Chromosome. Es gibt jedoch auch Crossover-Operatoren welche mit mehr als zwei Eltern arbeiten. Ganz nach der Evolutionstheorie sollen hier Chromosome mit einer höheren Fitness auch eine höhere Chance haben als Elternteil gewählt zu werden. Wie stark Chromosome mit einer höheren Fitness bevorzugt werden wird als Selektionsdruck bezeichnet. Selektionsschemata können in zwei Klassen unterteilt werden, proportionale Selektion und ordinale Selektion. [7] Eine proportionale Selektion gewichtet Chromosome anhand ihrer Fitness. Ordinale Selektion gewichtet Chromosome anhand ihres Ranges. Der Selektionsdruck Bei einer proportionalen Selektion ist der Selektionsdruck hoch und es besteht das Risiko einer verfrühten Konvergenz. Denn wenn es ein Chromosom gibt, welches weitaus fitter als der Rest der Population ist wird dieses einen proportionalen Selektionsprozess dominieren und somit die genetische Diversität

der Population senken. Andererseits führt ein geringer Selektionsdruck zu langsamer konvergenz. [7] Die Auswahl des Selektionsoperators sollte also wohlüberdacht sein.

### Roulette Rad Selektion

Roulette Rad Selektion ist einer der traditionellen proportionalen Selektionsoperatoren. Stellen wir uns ein Roulette Rad vor, welches in $N$ Segmente unterteilt ist, wobei $N$ die Anzahl der Chromosome in einer Population ist. Die Länge eines Segmentes $s_i$ ist proportional zu der normalisierten Fitness des korrespondierenden Chromosoms.

$$s_i = 2\pi \cdot \frac{fitness(i)}{\sum_{j=1}^{N} fitness(j)} \tag{1.12}$$

Nun wird das Roulette Rad gedreht und das Chromosom auf wessen Feld man landet wird in die Menge der Eltern aufgenommen. Diese Prozedur wird so oft wiederholt bis man die gewünschte Anzahl an Eltern gesammelt hat.

### Zufällige Selektion

Der zufällige Selektionsoperator ist der simpelste. Jedes Chromosom hat die gleiche Wahrscheinlichkeit in die Elternmenge aufgenommen zu werden.

### Rang Selektion

Die Rang Selektion ist eine Abwandlung der Roulette Rad Selektion. Die länge eines Segmentes ist jedoch nicht proportional zu der fitness sondern proportional zu dem Rang des Chromosoms.

$$s_i = 2\pi \cdot \frac{N - rank(i)}{\sum_{j=1}^{N} rank(j)} \tag{1.13}$$

### Elitismus

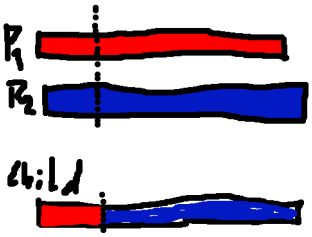
Die Selektionsoperatoren garantieren nicht, dass die besten Chromosome ausgewählt werden. Darüber hinaus kann es vorkommen, dass die besten Chromosome durch Crossover und Mutation verschlechtert werden, sodass die Fitness der folgenden Generation geringer als die vorherige ist. Um eine Abnahme der Fitness zu verhindern kann wird Elitismus angewendet. Die n besten Chromosome einer Population, die sogenannten Eliten werden auf jeden Fall in die nächste Generation aufgenommen ohne durch Crossover und Mutation verändert zu werden.

### 1.6.3 Crossoveroperator

Crossover bezeichnet das Rekombinieren von typischerweise zwei Eltern-Chromosomen zu einem Kind-Chromosom. Der vollständigkeit halber sei angeführt, dass auch Crossoveroperatoren existieren, welche mit mehr als zwei Eltern akzeptieren oder mehr als ein Kind produzieren. In dieser Arbeit beschränken wir uns jedoch auf Crossoveroperatoren von zwei Eltern zu einem Kind. Beim Crossover werden keine neuen Informationen in den Genpool eingeführt sondern es werden nur die vorhandenen Gene der Eltern rekombiniert. Es folg eine Aufzählung gängiger Crossoveroperatoren.

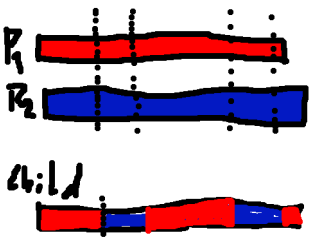
#### Single Point Crossover

Beim Single Point Crossover wird ein Cutpoint entlang der länge der Eltern gewählt. Beide Eltern werden dann an diesem Cutpoint geschnitten und das Kind-Chromosom setzt sich zusammen aus der ersten Hälfte des ersten Elternteils und der zweiten Hälfte des zweiten Elternteils.



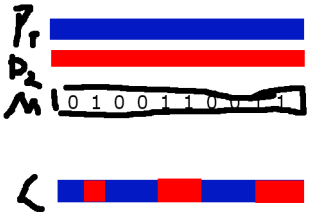
#### N-Point Crossover

Der N-Point Crossover ist eine Generallisierung des Single Point Crossovers. Es werden $n$ Crossover Points entlang der Länge der Eltern gewählt. Anhand dieser Crossover Points werden die Eltern in $n + 1$ Segmente unterteilt. Das Kind erhält alle Segmente mit geradem Index vom ersten Elternteil und alle Segmente mit ungeradem Index vom zweiten Elternteil. Im Allgemeinen führen mehr cutpoints jedoch zu einer geringeren Effizienz des genetischen Algorithmus. [7]



#### Uniform Crossover

Bei einem uniformen Crossover wird eine Crossovermaske $m$ mit gleicher Länge zu den Eltern erstellt. Das Kind erhält Gene der Eltern nach dieser Crossover Maske, wobei $m_i$ angibt, von welchem Elternteil das $i$-te Gen bezogen wird. Für jedes Elternpaar wird eine neue Crossovermaske erstellt. Typischerweise gilt $P(m_i = 1) = P(m_1 = 0) = 0.5$. Die Wahrscheinlichkeit, dass ein Gen von einem Elternteil bezogen wird kann jedoch auch gewichtet werden anhand der Fitness oder Ränge, so dass $P(m_i = 0) = w$ und $P(m_i = 1) = 1 - w$




### 1.6.4 Mutationsoperator

Der Crossoverschritt verringert unweigerlich die genetische Diversität der Population. Um dem entgegenzuwirken muss es einen Mechanismus geben, welcher neues genetisches Material hinzufügt. Dieser Mechanismus ist die Mutation, welche ein Chromosom zufällig verändert.

TODO: Include Examples for Real valued GA crossover

#### Mutationschance

Die Mutationschance bestimmt wie viele Gene eines Chromosoms durch die Mutation verändert werden und wie viele Unverändert bleiben. Bei einer Mutationschance von 100% werden alle Gene eines Chromosoms verändert und der genetische Algorithmus ist equivalent zu einer zufälligen Suche. [7]

### 1.6.5 Ersetzen

TODO: Ersetzen moped schmoped

Population Size Wählt man diese Zu gering kann es zu einer verfrühten Konvergenz und somit auch einer schlechten Lösung kommen Gleichzeitig ist die Population size ein

Maßgebender Faktor in der Rechenzeit und eine zu große Population könnte Rechenzeit verschwenden [GAs]

Im folgenden werde ich bekannte implementationen der genetischen Operatoren eingehen und beschreiben worauf man bei der implementation achten muss

### Mutationsoperatoren

Random Uniform Mutation: Random Unifor mutation

### Parameter eines genetischen Algorithmus

Wie im Abschnitt [Abschnitt Label hier] erwähnt haben Metaheuristiken den Nachteil, dass sie neue Parameter einführen. Ein genetischer Algorithmus, wie er zuvor beschrieben wurde hat folgende Parameter.

1. Populationsgröße
2. Anzahl der Generationen (oder Abbruchbedingung)
3. Mutationsfunktion und Mutationsrate
4. Crossoverfunktion und Crossoverrate
5. Selektionsfunktion
6. Anzahl der Eliten
7. Ersetzungsstrategie

Die Belegung dieser Parameter ist keinesfalls trivial denn die Parameter beeinflussen die Konvergenzrate des Algorithmus maßgeblich. Die Populationsgröße zum Beispiel beeinflusst die globale Suchkapazität, ein größerer Wert ist hier von Vorteil. Jedoch benötigt eine große Populationi auch mehr Rechenleistung, Speicher und Zeit [7]. Die Mutationsrate beeinflusst ebenfalls die Konvergenzrate. Eine zu hohe Mutationsrate führt zu einer langsamen Konvergenz und einem ineffizienten Algorithmus. Eine zu niedrige Mutationsrate führt wiederum zu einer verfrühten Konvergenz und somit zu weniger optimierten Lösungen. Die Auswahl der genetischen Operatoren wirkt sich unter anderem auf die Rechenlast aus. Ein Single-Point-Crossover zum Beispiel generiert eine zufällige Zahl, und macht einen Vergleich. Die Anzahl der benötigten Zufallszahlen und Vergleiche bei einem Uniform-Crossover hingegen sind equivalent zu der Anzahl an Genen. Bei mehreren Hunderten von Genen macht sich dieser Unterschied durchaus bemerkbar.

# Chapter 2

# Trainieren von HMMs mit einem GA

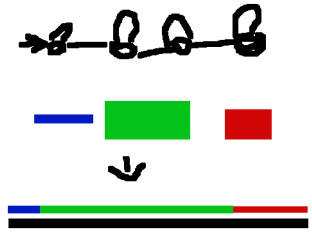
## 2.1 Konstruktion eines GAHMM

In diesem Kapitel werde ich beschreiben wie man einen Genetischen Algorithmus für Hidden Markov Modelle erstellen kann und dann auf die in meiner Arbeit verwendeten konkreten Implementationen der genetischen Operatoren eingehen.

### Representation

Ähnlich wie in den Papers [hier papers einfügen] Ist die Chromosonale Representation eines HMMs ein Vektor welcher die Reihen aller Matrizen hintereinander enthält.

Um die Struktur von nicht ergodischen Hidden Markov Modellen im Mutations-schritt nicht zu Verändern muss man die Gene Welche Startwahrscheinlichkeiten und Transitionswahrscheinlichkeiten beschreiben von der Mutation ausschließen oder man kann eine Maske definieren welche eine Mutation Gene welche initial 0 oder 1 sind verhindert.



### Fitness Operator

Die Fitness eines Chromosoms ist die durchschnittliche Log Wahrscheinlichkeit des Hidden Markov Models, welchen das Chromosom representiert. Als Observations Sequenzen werden Äußerungen der Zahl 0 aus dem Free Spoken Digit Dataset verwendet

### Mutationsoperatoren

Mutationsoperatoren wird dies das annanas gewählt

Belegung der Parameter:

Hidden Markov Parameter Bei den Hidden Markov Modellen welche von den genetischen Algorithmus optimiert werden handelt es sich ausschließlich um Left-right Modelle mit 4 Zuständen und 128 Observationssymbolen. Diese Werte sind relativ Arbiträr gewählt und orientieren sich an existierender Literatur zu Spracherkennung mit Hidden Markov modellen.

Als Observations-Sequenzen werden Äußerungen der Zahl 0 aus der Free Spoken Digit Database gewählt. Welche mittels eines k-means algorithmus mit k=128 quantisiert wurden.

Genetischer Algorithmus Parameter:

Die Populationsgröße wird auf 50 beschränkt und die Anzahl der Generationen auf 100. Das Rational für diese Entscheidung ist, dass ein Ablauf des Genetischen Algorithmus mit diesen Parametern auf meinem Klapprechner in unter 5 minuten abläuft und 100 Generationen meißt ausreichen um den Genetischen Algorithmus konvergieren zu lassen Die Anzahl der Observationssequenzen ist 10

### Erweiterung des GA

- Der genetische Algorithmus muss um einen Normalisierungsschritt erweitert werden, falls die Crossover oder Mutationsoperatoren keine Reihenstochastizität garantieren.

Da die Mutations und Crossoverfunktionen Lösungen erzeugen, welche außerhalb der zulässigen Region liegen müssen wir die Werte normalisieren. Das transformieren einer unzulässigen Lösung in eine zulässige ist bekannt als Reparatur Strategie [10]

## 2.2 Literaturübersicht

TODO: in diesem Chapter gebe ich eine übersicht über die unternommenen Ansätze das Training von Hidden Markov Modellen mit hybriden metaheuristischen Algorithmen.

Wir können drei Kategorien für die verwendung von genetischen Algorithmen für das trainieren von Hidden Markov Modellen erkennen. - Genetischer Algorithmus ohne Baum-Welch (Chau 1997) - Genetischer Algorithmus vor Baum-Welch fürs finden solider initialer Belegungen der Parameter (Slimane 1996)

- GA wird hybrid mit BW angewendet, so dass alle n Generation BW k mal angewendet wird.

Optimization of HMM By A Genetic Algorithm: - Genetischer Algorithmus ohne Baum Welch - 20000 GA Iterationen - population size - crossover-rate 0.01 - mutation rate 0.0001 - crossover type single point crossover - mutation type N-Point

Baum-welch - max 200 iterationen - 1 HMM

## 2.3 Implementierung

Die ursprüngliche Idee war es bereits existierende Python Libraries für HMMs und Genetische Algorithmen zu kombinieren. Da jedoch keine Libraries gefunden habe welche meinen hohen Ansprüchen an Erweiterbarkeit genügten habe ich den Code von Grund auf konstruiert. Meine Erfahrung, des aus dieser Entscheidung resultierenden Schaffensprozesses, lässt sich adäquat durch folgendes Zitat beschreiben. "The basic idea behind the from scratch-oriented approach is the apparent simplicity of metaheuristic code. Programmers are tempted to develop themselves their codes. Therefore, they are faced with several problems: the development requires time and energy, and it is error prone and difficult to maintain and evolve" [10]

# Chapter 3

# Evaluation

## 3.1 Vergleich genetischer Operatoren

## 3.2 Berechnungskost eines GA

- Wallah Teuer - Lohnt net

Für gewöhnlich ist bei Metaheuristiken die Evaluation der Objektiven Funktion am Rechenintensivsten. [10].

=¿ Ist das hier auch der Fall?

# Bibliography


[1] Ehrhard Behrends. *Elementare Stochastik*. 2012.

[2] Nen paar dullis. *Paper mit nem tittel*. 2011.

[3] *Grey Wolf Firefly and Bat Algorithms Three Widespread Algorithms that Do Not Contain Any Novelty*. 2020.

[4] Dominik Wied Karsten Webel. *Stochastische Prozesse*. 2016.

[5] Christian Kohlschein. *An introduction to Hidden Markov Models*.

[6] Sivanandam. *Genetic Algorithms. In: Introduction to Genetic Algorithms. Springer*. 2008.

[7] Sivanandam. *Terminologies and Operators of GA. In: Introduction to Genetic Algorithms. Springer*. 2008.

[8] Kenneth Sorensen. *Metaheuristics—the metaphor exposed*. 2012.

[9] Springer. *Metaheuristics*. 2008.

[10] El-Ghazali Talbi. *Metaheuristics - From design to implementation*. 1965.

[11] Dennis Weyland. *A rigoruous analysis vong dem harmony moped*. 2010.

[12] Gordan Žitković. *Introduction to Stochastic Processes - Lecture Notes*. 2010.

**Eigenständigkeitserklärung**

Hiermit versichere ich, dass ich diese Arbeit bzw. im Fall einer Gruppenarbeit den von mir entsprechend gekennzeichneten Anteil an der Arbeit selbständig verfasst habe. Ich habe keine unzulässige Hilfe Dritter in Anspruch genommen. Zudem habe ich keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und alle Ausführungen (insbesondere Zitate), die anderen Quellen wörtlich oder sinngemäß entnommen wurden, kenntlich gemacht. Ich versichere, dass die von mir in elektronischer Form eingereichte Version dieser Arbeit mit den eingereichten gedruckten Exemplaren übereinstimmt. Mir ist bekannt, dass im Falle eines Täuschungsversuches die betreffende Leistung als mit "nicht ausreichend" (5,0) bewertet gilt. Zudem kann ein Täuschungsversuch als Ordnungswidrigkeit mit einer Geldbuße von bis zu 50.000 Euro geahndet werden. Im Falle eines mehrfachen oder sonstigen schwerwiegenden Täuschungsversuchs kann ich zudem exmatrikuliert werden. Mir ist bekannt, dass sich die Prüferin oder der Prüfer bzw. der Prüfungsausschuss zur Feststellung der Täuschung des Einsatzes einer entsprechenden Software oder sonstiger elektronischer Hilfsmittel bedienen kann.

Duisburg, 3. März 2023 _____

(Ort, Datum) (Vorname Nachname)