Master in Human Centered Computing
„Formale Methoden“

SS2015

Prof. Dr.-Ing. habil. Natividad Martínez

- Abschlussausarbeitung -

Das Hidden Markov Model im Machine Learning Kontext

---

vorgelegt von:

Paul Pasler

1. Semester

Contact addresses:
paul.pasler@student.reutlingen-university.de

Submitted on: 20.06.2015

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einführung

Das Hidden Markov Model hat im Bereich des machinellen Lernens viele Anwendungsfälle. In der vorgelegten Arbeit werden diese und die Funktionsweise des Hidden Markov Model vorgestellt (Kapitel 3). Die dafür notwendigen Grundlagen werden in den nachfolgenden Abschnitten und im Kapitel 2 beleuchtet. Kapitel 4 zeigt den Vergleich mit anderen Ansätzen im Machine Learning Kontext. Ein abschließendes Fazit wird im letzten Kapitel (5) gezogen.

1.1 Machine Learning

Machine Learning befasst sich mit der modellierung des Lernvorgangs auf einem Computer [Mar09]. Es wird versucht ein "künstliche" Generierung von Wissen aus Erfahrung zu erzeugen. Dabei wird anhand von Beispielen "gelernt", sodass nicht nur die selben Daten wieder erkannt werden können, sondern auch ähnliche bzw. unbekannte Daten klassifiziert werden. Diese Transferleistung nennt man Generalisierung und ist auch beim Menschen eine wichtige Eigenschaft im Lernvorgang.

So können wir Äpfel von Birnen (Siehe Abbildung 1.1 [1]) unterscheiden, egal, ob wir genau diese Frucht schon einmal gesehen haben. Wir entscheiden anhand gelernter Merkmale, um welche Frucht es sich vermutlich handelt. Merkmale sind bspw. Größe, Form, Farbe, Geruch etc.

Abb. 1.1: *Birne und Apfel unterscheiden sich durch Farbe, Form etc. - einen Stiel haben jedoch beide*

Die Extraktion signifikanter Merkmale ist ein wichtiger Teil von Machine Learning. Viele Eigenschaften eines Objektes sind nicht geeignet, es von anderen

[1] Quelle: `http://www.lifeline.de/img/abnehmen/origs76797/7656955923-w830-h830/Birne-und-Apfel.jpg`

zu unterscheiden. Im Apfel-Birnen Beispiel würde das Merkmal “Stiel” nicht zu einer Unterscheidung führen.

Der nächste Schritt ist das Training des Systems. Hierbei wird zwischen drei algorithmischen Ansätzen unterschieden:

- Überwachtes Lernen
- Unüberwachtes Lernen
- Bestärkendes Lernen

Die häufigste menschliche Lernform, ist das bestärkende Lernen, hier wird mit “Belohnung” und “Bestrafung” gearbeitet. Im Machine Learning Bereich ist jedoch Überwachtes und Unüberwachtes Lernen sehr viel häufiger zu finden. Beim überwachten Lernen, werden mehrere Eingaben und Lösungen an den Algorithmus überreicht und nach einigen Durchgängen, sollte er in der Lage sein Assoziationen herzustellen. Je nach Algorithmus werden hierzu Funktionen und Gewichtungen angepasst. Das Hidden Markov Model ist an ehesten im Bereich des “Unüberwachten Lernens” beheimatet.

Aus der Menge der Eingaben wird ein Modell erzeugt, das Vorhersagen ermöglichen soll. Dazu wird eine Form des Expectation-Maximization-Algorithmus (EM-Algorithmus [AD77]) genutzt. Bei einem EM-Algorithmus wird versucht, die vorliegenden Daten in Kategorien einzuteilen, sodass die Daten optimal erklärt werden. Hierbei wird versucht in einem zweistufigen Verfahren (Expectation und Maximization), das Modell besser zu machen.

Weitere Machine Learning Ansätze sind (nicht vollständig)

- Neuronale Netze
- Support Vector Machine
- K-Means

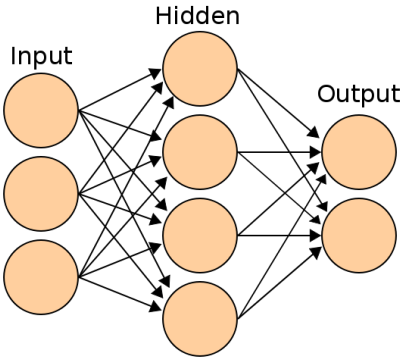

Abb. 1.2: *Vereinfachte Darstellung eines neuronalen Netzwerkes mit 3 Merkmaldimensionen.*

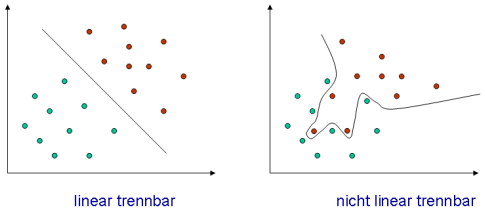

Abb. 1.3: *Support Vector Machine: Daten lassen sich nicht immer linear trennen.*

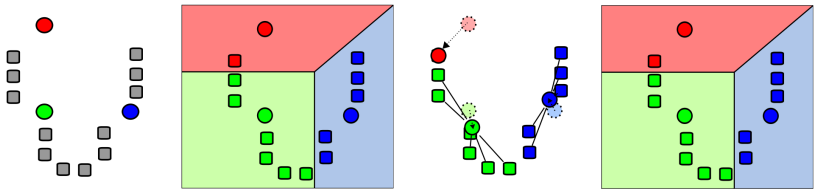

Abb. 1.4: *Vereinfachter Ablauf einer Iteration des k-Means Algorithmus. 1) Clusterzentren zufällig wählen. 2) Datenpunkte den Clustern zuordnen. 3) Clusterzentren verschieben. 4) Datenpunkt den neuen Clustern zuordnen.*

Neuronale Netze versuchen das menschliche Gehirn mit seinen Neuronen und Synapsen nachzubauen [MP88]. Für jede Dimension des Merkmalsvektors sind Neuronen vorhanden, welche wiederum mit anderen Neuronen verschaltet sind. Beim Training werden die Gewichtungen der einzelnen Verschaltungen verändert. Abbildung 1.2 [2] zeigt ein vereinfachtes neuronales Netz mit drei Inputs, vier weiteren Neuronen und zwei Outputs.

Die Support Vector Machine (Stützvektormaschine) versucht die Daten durch lineare Trennung zu Klassifizieren (Siehe Abbildung 1.3 [3]) [BGV92]. Es wird versucht, den Stützvektor möglichst weit von den beiden Klassen entfernt zu erstellen (Large-Margin-Classifier).

Der k-Means Ansatz versucht die gegebenen Daten zu gruppieren bzw. zu clustern und geht auf den polnischen Mathematiker Hugo Steinhaus zurück [Ste57]. Zu beginn wird die Anzahl der Cluster k definiert und diese per Zufall im Koordinatensystem verteilt (Siehe Abbildung 1.4). Nun wird in jeder Iteration versucht, das Zentrum der Cluster zu verbessern und näher an die Daten zu bringen. k-Means ist ebenfalls ein EM-Ansatz.

Im Abschnitt 4 werden die vorgestellten Ansätze Neuronale Netze (4.1), SVM (4.2) und k-Means (4.3) mit dem Hidden Markov Model verglichen.

1.2 Mathematische Grundlagen

Im folgenden Abschnitt werden grundlegende mathematische Grundlagen und Definition vorgestellt.

1.2.1 Zeit-diskreter Prozess

Ein zeit-diskreter Prozess beschreibt die Eigenschaft, dass sich die Zustände eines Systems immer im selben zeitlichen Abstand ändern (Äquidistante Zeitabstände). Beispielsweise wird jede Sekunde ein Tick ausgelöst, welcher zu einer Zustandsänderung führt. Ist dies nicht der Fall, dann spricht man von einem zeit-kontinuierlichen Prozess.

[2] Quelle: `http://en.wikipedia.org/wiki/Artificial_neural_network#/media/File:Artificial_neural_network.svg`

[3] Quelle: `http://upload.wikimedia.org/wikipedia/de/a/a0/Diskriminanzfunktion.png`

1.2.2 Wahrscheinlichkeit

Die Wahrscheinlichkeit (Probabilität) beschäftigt sich mit der mathematischen Untersuchung von Zufallsgeschehen. Gemeinsam mit der Statistik, bildet sie das mathematische Teilgebiet der Stochastik.

Dafür wird von einem Zufallsexperiment ausgegangen und dieses analysiert. Alle möglichen Ergebnisse dieses Experiments werden in der Ergebnismenge Ω zusammengefasst. Meistens ist nicht das genaue Ergebnis $\omega \in \Omega$ von Interesse, sondern lediglich, ob das Ergebnis in einer bestimmten Teilmenge von Ω liegt. Weiterhin wird mit dem Ereignisraum Σ definiert und enthält alle möglichen Ereignisse. Die Wahrscheinlichkeit wird mit P bezeichnet und bewegt sich im Intervall $[0, 1]$.

Im Beispiel "Münzwurf" ergeben sich dann folgende Definitionen [4].

- Ergebnismenge $\Omega = \{Zahl, Kopf\}$
- Ereignisraum $\Sigma = \{\emptyset, \{Zahl\}, \{Kopf\}, \Omega\}$
- Wahrscheinlichkeiten
  - $P(\emptyset) = 0$
  - $P(\{Zahl\}) = 1 - P(\{Kopf\})$
  - $P(\Omega) = 1$

Bei den Wahrscheinlichkeiten ist $P(\emptyset) = 0$ und $P(\Omega) = 1$, da es ein Ergbenis geben muss und dieses aus Ω stammen muss. Die Wahrscheinlichkeit von $\{Zahl\}$ und $\{Kopf\}$ addieren sich zu 1. Bei einer fairen idealen Münze haben Kopf und Zahl jeweils eine Wahrscheinlichkeit von 0.5 (50%).

1.2.3 Bedingte Wahrscheinlichkeit

Die bedingte Wahrscheinlichkeit beschreibt die Wahrscheinlichekit für ein das Eintreten eines Ereignisses A unter der Bedingung, dass vorher ein Ereigniss B eingetreten ist. $P(A|B)$ drückt diesen Zusammenhang aus, der $|$ ist als "unter der Bedingung" zu lesen.

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

Definition 1.1 zeigt eine weitere Schreibweise der bedingten Wahrscheinlichkeit, bei der die Wahrscheinlichkeit, dass A und B gemeinsam auftreten, durch die Wahrscheinlichkeit, dass B eintritt geteilt wird.

[4] Quelle: `http://de.wikipedia.org/wiki/Wahrscheinlichkeitstheorie`

2 Markov Kette

Grundlage des Hidden Markov Model war die vom russischen Mathematiker Andrej Andrejewitsch Markov (1856 - 1922, siehe [Mar13]) entwickelte Markov Kette. Zu Beginn des 20. Jahrhunderts beschäftigte er sich als erster mit einer statistischen Beschreibung von Zustands- und Symbolfolgen. Er führte eine statistische Analyse der Buchstabenfolge des Textes "Eugene Onegin" von Alexander Pushkin durch. Die erstellte Markov-Kette erlaubte es, ausgehend von einem bestimmten Buchstaben, die Wahrscheinlichkeit (Siehe Abschnitt 1.2.2) für den folgenden Buchstaben zu bestimmen.

Veranschaulichen lässt sich eine Markov Kette als gerichtetes Zustandsdiagramm (Abb. 2.1 [1]) mit einer Menge von Zuständen S und mit den Übergangswahrscheinlichkeiten X_i an den Kanten

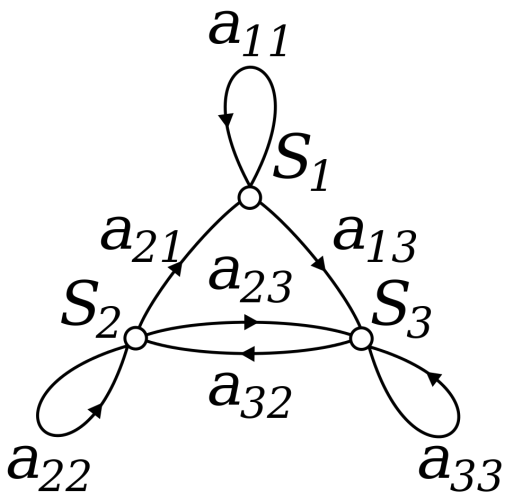

Abb. 2.1: *Einfaches Zustandsdiagramm einer Markov Kette*

2.1 Beispiel

Markov Kette für das Wetter [2]

Im folgenden Beispiel soll aufgrund des aktuellen Wetters auf das Wetter der folgenden Tage geschlossen werden. Das Wetter kann entweder "sunny" oder "rainy" sein, zu Beginn (Tag 0, $t = 0$) des Experiments ist es "sunny". Die Wahrscheinlichkeit, dass auf "sunny" wieder "sunny" folgt, liegt bei 90% ("rainy"

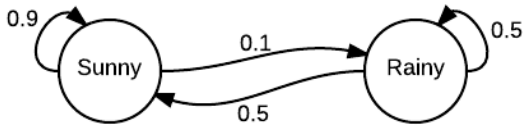

Abb. 2.2: *Gerichteter Zustandsgraph der modellierten Wetter-Markov Kette*

= 1 - "sunny" = 10%). Nach "rainy" liegt die Wahrscheinlichkeit jeweils bei 50% (siehe Abb. 2.2).

$$Zustände : S = [\text{“}sunny''\text{,“}rainy'']$$

$$Anfangszustand : \Pi = X_0 = [1,0]$$

$$Übergangsmatrix : A = \begin{bmatrix} 0.9 & 0.1 \\ 0.5 & 0.5 \end{bmatrix}$$

Nun kann die Wahrscheinlichkeit für das Wetter an Tag 1 berechnet werden über:

$$X_1 = X_0 * A = [1,0] \begin{bmatrix} 0.9 & 0.1 \\ 0.5 & 0.5 \end{bmatrix} = [0.9, 0.1]$$

Für Tag 2:

$$X_2 = X_1 A = X_0 A^2 = [1,0] \begin{bmatrix} 0.9 & 0.1 \\ 0.5 & 0.5 \end{bmatrix}^2 = [0.86, 0.14]$$

Verallgemeinert für Tag k bedeutet das:

$$X_k = X_{k-1} A = X_0 A^k = [1,0] \begin{bmatrix} 0.9 & 0.1 \\ 0.5 & 0.5 \end{bmatrix}^k$$

2.2 Definition

Eine Markov Kette beschreibt einen zeit-diskreten Prozess $(X_t)_{t \in \mathbb{N}_0}$ mit m abzählbaren Zuständen S [KHW13]. Weiterhin wird sie als stationär bezeichnet, wenn alle Wahrscheinlichkeiten unabhängig von der Zeit sind und sich nicht

[1] Quelle: `de.wikipedia.org/wiki/Markow-Kette`
[2] Quelle: `en.wikipedia.org/wiki/Examples_of_Markov_chains`

mehr verändern. Da die Verteilung der Zufallsvariablen nur von den vergangenen Zuständen abhängt, gilt eine Markov Kette als kausal [Fin03, 48]. Definition 2.1 zeigt die allgemeine Berechnung der Übergangswahrscheinlichkeiten einer Markov Kette. Als Übergangswahrscheinlichkeit bezeichnte man die bedingte Wahrscheinlichkeit (Siehe Abschnitt **??**) $P(X_{t+1} = s_{t+1}|X_t = s_t)$, dass auf den aktuellen Zustand s_t der Nachfolgezustand s_{t+1} folgt.

$$P(X_{t+1} = s_{t+1}|X_0 = s_0, \ldots, X_{t-1} = s_{t-1}, X_t = s_t) \tag{2.1}$$

Wichtig für eine Markov Kette ist die sogenannte Markov-Eigenschaft (Siehe Definition 2.2). Im Unterschied zu Definition 2.1 hängt der nächste Zustand nur vom aktuellen Zustand ab. Anders ausgedrückt beschreibt die Markov-Eigenschaft die Gedächtnislosigkeit des Prozesses, da der Folgezustand nur vom direkten Vorgänger abhängt.

$$P(X_{t+1} = s_{t+1}|X_t = s_t) \tag{2.2}$$

Genügt eine Markov Kette dieser Eigenschaft, wird sie als “einfach” oder Markov Kette 1. Ordnung bezeichnet.

Die Übergangswahrscheinlichkeiten werden üblicherweise zu einer Übergangsmatrix zusammengefasst (Definition 2.3)

$$A = [a_{ij}] = \begin{bmatrix} a_{00} & \cdots & a_{0m} \\ \vdots & \ddots & \vdots \\ a_{m0} & \cdots & a_{mm} \end{bmatrix} \forall i, j \in S \tag{2.3}$$

Da es sich um Wahrscheinlichkeiten handelt, muss sich die Summe jeder Reihe zu Eins addieren.

Weiterhin benötigt der Prozess einen Vektor für den Anfangzustand $t = 0$ (Definiton 2.4)

$$\Pi = [\pi_i] = [P(X_0 = i)], i \in S \tag{2.4}$$

So lässt sich eine Markov-Kette durch Zustandsraum S, den Übergangsmatrix A und einen Anfangszustand Π definieren.

Die Wahrscheinlichkeit für k-Schritte lässt sich so ausrechnen:

$$X_k = X_{k-1}A = X_0A^k$$

3 Hidden Markov Model

Das Hidden Markov Model ist ein stochastisches Modell für sequentielle Daten und wird vor allem in der Spracherkennung und in der Bioinformatik eingesetzt.

Der amerikanischen Mathematiker Leonard E. Baum (* 1931) und andere Autoren entwickelten auf Basis der Markov Kette Ende der sechziger Jahre das Hidden Markov Model [BP66]. Erste Hidden Markov Model-Applikationen wurden zur Spracherkennung und später auch in der Bioinformatik zur Analyse von Nukleotid- und Proteinsequenzen eingesetzt.

Mit dem Hidden Markov Model können Erkentnisse über ein System gewonnen werden, ohne genau das Aussehen bzw. die Zustände ("Hidden states") sehen zu können. Es wird lediglich eine Wahrschinlichkeit anhand von Beobachtungen (Emissionen) berechnet.

Abbildung 3.1 zeigt die möglichen Zustände x, die Übergangswahrscheinlichkeiten a, sowie die möglichen Beobachtungen y und die Emissionswahrscheinlichkeiten b. x und a sind Teil einer gewöhnlichen Markovkette (Siehe Abschnitt 2). Beobachtbar (sichtbar) sind jedoch nur die Emissionen y. Mit dem Wissen der Emissionswahrscheinlichkeiten b lassen sich trotzdem Rückschlüsse auf die Zustände x ziehen.

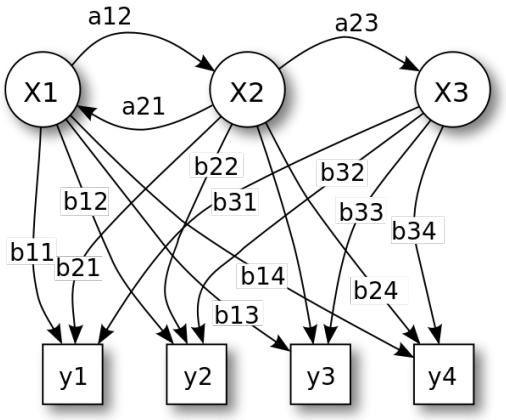

Abb. 3.1: *Zustandsgraph eines Hidden Markov Models*

3.1 Beispiele

Gefangener im Verlies [1]

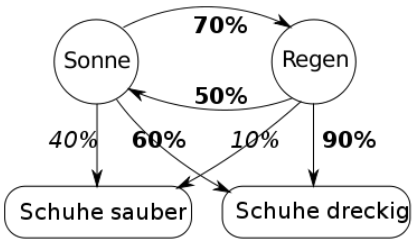

Abb. 3.2: *Hidden Markov Model für das Beipiel des Gefangen im Verlies*

Ein Gefangener im Kerkerverlies möchte das aktuelle Wetter herausfinden. Er weiß, dass auf einen sonnigen Tag zu 70 % ein Regentag folgt und dass auf einen Regentag zu 50 % ein Sonnentag folgt. Weiß er zusätzlich, dass die Schuhe der Wärter bei Regen zu 90 % dreckig, bei sonnigem Wetter aber nur zu 60 % dreckig sind, so kann er durch Beobachtung der Wärterschuhe Rückschlüsse über das Wetter ziehen (das heißt, er kann die Wahrscheinlichkeit für Regenwetter gegenüber sonnigem Wetter abschätzen).

Sonne und Regen sind in diesem Fall die versteckten Zustände x. Die Emissionen bzw. die Observation die der Gefangene machen kann sind nur der Verschmutzungsgrad der Schuhe der Wärter y. Dabei ist es völlig egal, welches Wetter vor 2 Tage geherrscht hat (Wegen der Markov-Eigenschaft, siehe 2.2).

Spracherkennung
In der automatischen Spracherkennung entsprechen bspw. die gesprochenen Laute (Phoneme) den versteckten Zuständen und die Silben bzw. Worte den Emissionen eines Hidden Markov Models (Siehe [Eul05]). Das Model hat somit die Aufgabe, aus einer Sequenz von Lauten auf eine Sequenz von Silben oder Wörtern zu schließen. Das Hidden Markov Model passt sehr gut zur Vorstellung von Sprachsignalen als Abfolge einzelner Ereignisse.

[1] Quelle: `http://de.wikipedia.org/wiki/Hidden_Markov_Model`

3.2 Definition

Ein Hidden Markov Model erweitert eine Markov Kette um eine weiteren Zufallsprozess und ist somit ein zweistufiger stochastischer Prozess [Fin03, 67]. Hierfür wird jedem Zustand der Markov Kette eine Ausgabe bzw. Emission zugeordnet dessen Wahrscheinlichkeitsverteilung einzig vom aktuellen Zustand abhängig ist. Die Emissionen sind die einzigen beobachtbaren Zustände des Hidden Markov Model. Der Rest ist sozusagen 'versteckt' woher sich auch der Name des Models ableitet. Eine Folge von Emissionen wird auch Observationsfolge genannt (Siehe Definition 3.1).

$$o \in Y; o = y_a, \ldots, y_z \tag{3.1}$$

Das Hidden Markov Model wird in 3.2 definiert [Fin03, 68].

$$\lambda = (X; Y; A; B; \pi) \tag{3.2}$$

- Endlich Menge von Zuständen
  $X = \{x | 1 <= x <= N\}$

- Alphabeth der Emissionen
  $Y = \{y | 1 <= y <= M\}$

- Matrix der Zustandsübergangs-wahrscheinlichkeiten
  $A = \{a_{ij} | a_{ij} = P(X_t = j | X_{t-1} = i)\}$

- Matrix der Emissionsverteilung
  $B = \{b_{jk} | b_{jk} = P(Y_t = o_k | X_t = j)\}$ bzw.
  $B = \{b_j(x) | b_j(x) = p(x | X_t = j)\}$

- Vektor von Zustandsstartwahrscheinlichkeiten
  $\pi = \{\pi_i | \pi_i = P(X_1 = i)\}$

Die Emissionsmodellierung ist hierbei vom Kontext der Problemstellung abhängig. Wird das Hidden Markov Model zum Beispiel bei der Analyse von biologischen Sequenzen, spricht von einem diskreten Symbolinventar und es wird ein diskretes Emissionsmodel genutzt (Siehe Definition 3.3). Man spricht hierbei auch von einem diskreten Hidden Markov Model. Wenn dieses Model

zur Verarbeitung von Signalen verwendet werden soll, erfordert dies in der Vorverarbeitung der Daten einen Quantisierer der die kontinuerlichen Merkmale in eine diskrete Observationsfolge überführt.

$$B = \{b_{jk} | b_{jk} = P(Y_t = o_k | X_t = j)\} \tag{3.3}$$

Gängiger ist es hierfür kontinuierliche Hidden Markov Model's zu nutzen. Hierbei wird eine Emissionsmodelierung auf Basis kontinuierlicher Dichtefunktionen genutzt die kontinuierliche Observationen im $\mathbb{R}^n$ verarbeitet (Siehe Definition 3.4).

$$B = \{b_j(x) | b_j(x) = p(x | S_t = j)\} \tag{3.4}$$

Zur Behandlung kontinuerlicher Verteilungen mit mehreren komplexen Häufigkeitsgebieten werden approximatische Verfahren genutzt. Die verbreiteste Technik besteht aus der Verwendung von Mischverteilungen auf der Basis von Gauß-Dichten (Gaussian Mixture Model). Es lässt sich zeigen, dass sich jede allgemeine kontinuierliche Verteilung $p(x)$ durch eine Linearkombination von i.a. unendlich vielen Basis-Normalverteilungen beliebig genau approximieren lässt [Fin03, 69] (Siehe Definition 3.5)

$$p(x) \hat{=} \sum_{k=1}^{\infty} c_k N(x | \mu_k, K_k) \approx \sum_{k=1}^{M} c_k N(x | \mu_k, K_k) \tag{3.5}$$

Der Approximationsgfehler lässt sich hierbei über eine geeignete Anzahl von M Basisverteilungen klein halten. Somit ergibt sich für die Beschreibung der Emissionsverteilung eines Zustands des Hidden Markov Model folgende Formel:

$$b_j(x) = \sum_{k=1}^{M} c_{jk} g_{jk}(x) \tag{3.6}$$

Die Anzahl der Basisverteilungen eines Gaussian Mixture Model kann hierbei für die einzelnen Zustände des HMM variieren.

3.3 Funktionsweise

Das Konzept des Hidden Markov Model kann laut [Rab89] in drei Problemstellungen eingeteilt werden:

- Evaluierungsproblem: Bestimme die Wahrscheinlichkeit für ein Model, mit der dieses eine gegebene Observationsfolge (o) erzeugt. (Siehe Abschnitt 3.3.1)

- Dekodierungsprobem: Finde interne Abläufe für eine gegebene Observationsfolge (Siehe Abschnitt 3.3.2)

- Trainingsproblem: Finde Modellparameter für gegebene Beispieldaten (Siehe Abschnitt 3.3.3)

3.3.1 Evaluierung

In der Evaluierung soll die Wahrscheinlichkeit bestimmt werden, mit der eine betrachtete Observationsfolge o in einer beliebigen Zustandsfolge von einem gegebenen Hidden Markov Model λ generiert wird. Diese Wahrscheinlichkeit wird auch Produktionswahrscheinlichkeit genannt.

Die Produktionswahrscheinlichkeit wird mit dem Forward-Algorithmus berechnet. Der Algorithmus nutzt hierfür die geltende Markov Eigenschaft (Siehe Abschnitt 2] aus, das nur die Speicherung eines (des letzten) internen Zustandes erlaubt. Hierfür definiert man als Vorwärtsvariable $\alpha_t(i)$ die Wahrscheinlichkeit, bei gegebenem Model λ den Anfang der betrachteten Observationsfolge O_t zu erzeugen und zum Zeitpunkt t den Zustand i zu erreichen (Siehe Definition 3.7).

$$\alpha_t(i) = P(O_1, O_2, \ldots, O_t, s_t = i|\lambda) \tag{3.7}$$

Die Vorwährtsvariable lässt sich nun mit den folgenden Schritten rekursiv berechnen, um dann die Gesamtwahrscheinlichkeit des Models zu erhalten.

1. Initialisierung
   $\alpha_1(i) := \pi_i bi(O_1)$

2. Rekursion
   für alle Zeitpunkte $t, t = 1 \ldots T-1$
   $\alpha_{t+1}(j) := \sum_i \{\alpha_t(i)\alpha_{ij}\} b_j(O_{t+1})$

3. Rekursionsabschluss
   $P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$

3.3.2 Dekodierung

Bei der Dekodierung soll, für ein gegebenes λ und eine Observationsfolge o, die optimale bzw. wahrscheinlichste Zustandsfolge s^* aus der Menge der Zustände ermittelt werden. Zur Ermittlung der optimalen Zustandsfolge bedient man sich des Viterbi-Algorithmus, einem induktiven Verfahren, dass dem Forward Algorithmus sehr ähnlich ist. Zu Begin werden erneut die Wahrscheinlichkeiten $\delta_t(i)$ für partiell optimale Pfade definiert, die das Anfangssegment der Observationsfolge bis O_t mit maximaler Wahrscheinlichkeit erzeugen und in Zustand i enden.

$$\delta_t(i) = \max_{s_1,s_2,\ldots,s_t} P(O_1,\ldots,O_t,s_1,\ldots,s_t = i|\lambda) \tag{3.8}$$

Der Algorithmus entspricht weitgehend dem Forward-Algorithmus jedoch werden anstatt der Summe im Rekursionsabschluss, die Maximalen über die in den Vorgängerzuständen vorliegenden Wahrscheinlichkeiten gebildet.

1. Initialisierung
   $\delta_1(i) := \pi_i bi(O_1)$
   $\phi_1(i) := 0$
2. Rekursion
   für alle Zeitpunkte $t, t = 1 \ldots T-1$
   $\delta_{t+1}(j) := \max_i\{\delta_t(i)\alpha_{ij}\}b_j(O_{t+1})$
   $\phi_{t+1}(j) := \arg\max_i\{\lambda_t(i)\alpha_{ij}$
3. Rekursionsabschluss
   $P^*(O|\lambda) = (P(O,s^*|\lambda) = \max_i \lambda_T(i)\alpha_T(i)$
4. Rückverfolgung des Pfades
   für alle Zeitpunkte $t, t = 1 \ldots T-1$
   $s_t^* = \phi_{t+1}(s_{t+1}^*)$

Mit $\phi_t(j)$ wird ein "Rückwärtszeiger" definiert, der für jedes entsprechende $\delta_t(j)$ entlang der partiellen Pfade den jeweils optimalen Vorgängerzustand speichert.

Viterbi Algorithmus: Code Beispiel [2]

Das Code-Beispiel implementiert das Vorgehen eines Arztes, der nur anhand von Aussagen der Dorfbewohner, über ihren Gesundheitszustand entscheidet, ob der einzelne Dorfbewohner Gesund ist oder Fieber hat (Siehe Abbildung 3.3).

Codeblock 3.1 zeigt die Initialiserung, bei der das HMM implementiert wird.

[2] Quelle: `https://en.wikipedia.org/wiki/Viterbi_algorithm`

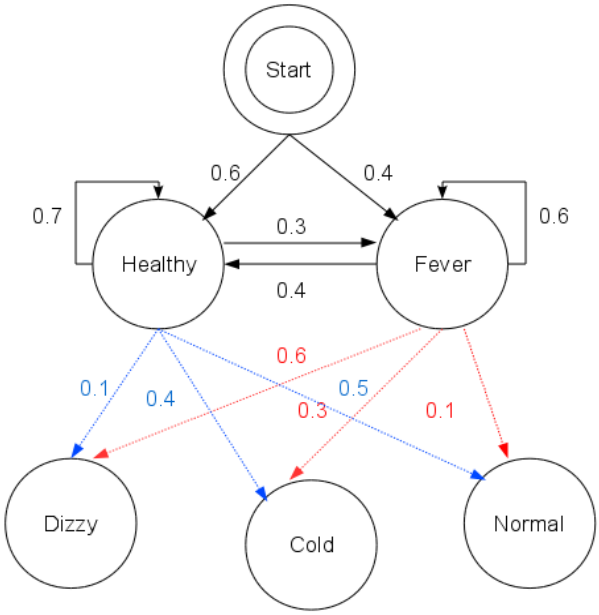

Abb. 3.3: *HMM für das Code-Beispiel, das den Gesundheitszustand von Dorfbewohnern darstellt.*

```
states = ('Healthy', 'Fever') # X
observations = ('normal', 'cold', 'dizzy') # Y
start_probability = {'Healthy': 0.6, 'Fever': 0.4}#pi
transition_probability = { # A
 'Healthy' : {'Healthy': 0.7, 'Fever': 0.3},
 'Fever' : {'Healthy': 0.4, 'Fever': 0.6}
}
emission_probability = { # B
  'Healthy' : {'normal': 0.5, 'cold': 0.4, 'dizzy':
     0.1},
  'Fever' : {'normal': 0.1, 'cold': 0.3, 'dizzy':
     0.6}
}
```

Codeauszug 3.1: *Initialisierung*

Codeblock 3.2 zeigt die einzelnen Schritte des Viterbi-Algorithmus (Iterativ und nicht Rekursiv). "V" ist in diesem Fall die definierte Vorwärtsvariable δ und "path" die Speicherung des Pfades ϕ. Zeile 6-8 zeigt die Initialisierung der Zustände zum Zeitpunkt 0. Nun wird für jeden Zeitpunkt der Weg in den Zuständen mit der maximalen Wahrscheinlichkeit bestimmt

```
def viterbi(obs, states, start_p, trans_p, emit_p):
    V = [{}]
    path = {}

    # Initialize base cases (t == 0)
    for y in states:
        V[0][y] = start_p[y] * emit_p[y][obs[0]]
        path[y] = [y]

    # Run Viterbi for t > 0
    for t in range(1, len(obs)):
        V.append({})
        newpath = {}

        for y in states:
            (prob, state) = max((V[t-1][y0] * trans_p
               [y0][y] * emit_p[y][obs[t]], y0) for y0
                in states)
            V[t][y] = prob
            newpath[y] = path[state] + [y]

        # Don't need to remember the old paths
        path = newpath
    n = 0  # if only one element is observed max is
       sought in the initialization values
    if len(obs) != 1:
        n = t
    print_dptable(V)
    (prob, state) = max((V[n][y], y) for y in states)
    return (prob, path[state])
```

Codeauszug 3.2: *Viterbi-Algorithmus*

3.3.3 Training

Gegeben sei eine Observationsfolge o, aus dem ein passendes Hidden Markov Model erzeugt werden soll.

Je nach Problemstellung, müssen unterschiedliche Modelle eines HMM's gewählt werden. Es ist bisher kein Verfahren bekannt das aufgrund einer Stichprobe ein optimales Modell generieren kann. Die Anzahl der Zustände, die Wahl der Emissionsverteilungen, sowie deren initiale Parameterwerte müssen nach eigenen Erfahrungen gewählt werden. Wenn dies geschehen ist kann das Modell in einem iterativen Prozess trainiert werden. Hierbei werden die Parameter einer Wachstumstransformation unterworfen. Ziel ist es das die Modellparameter so verändert werden, dass die Bewertung des veränderten Models besser, als die des Ausgangsmodels ist.

Zum trainieren eines Hidden Markov Model existieren diverse Algorithmen. Sie unterscheiden sich im wesentlichen durch die Verwendeten Qualitätsmaße zur Bewertung der Modelierungsgüte. Beim Baum-Welch-Algorithmus [Rab89] wird die Produktionswahrscheinlichkeit $P(O|\lambda)$ zur Bewertung genutzt. Beim Viterbi-Algorithmus [Vit06] und dem eng verwandten Segmental-k-means Algorithmus [JR90] nur die Wahrscheinlichkeit ($P(O, s^*|\lambda)$ der jeweils optimalen Zustandsfolge betrachtet [Fin03].

4 Vergleich mit anderen Machine Learning Ansätzen

Es exisitieren zahllose Ansätze, um maschinelles Lernen abzubilden. Jedes Verfahren hat seinen Ursprung und wurde oft für einen speziellen Anwendungsfall entwickelt. Für jedes Verfahren gibt es zudem wieder verschiedene Erweiterungen, Algorithmen und Implementierungen, die spezielle Probleme lösen. Darum fällt es schwer zu sagen, was der beste Ansatz ist - ohne die genauen Anforderungen zu kennen.

Dennoch werden in den folgenden Abschnitten kurze Gegenüberstellungen, sowie Vor- und Nachteile vorgestellt.

4.1 Neuronale Netze

Das Neuronale Netz mit allen seinen Ablegern, sind heute wieder eine wichtige Größe im Machine Learning Bereich, nachdem sie Ende der 1960er Jahre wegen einiger Probleme nicht mehr fokusiert wurden. So konnte das Ursprünglich Neuronale Netz kein XOR abbilden konnte und Probleme in der linearen Separierbarkeit hatte. Zudem ist nicht ganz klar, was in einem neuronalen Netz vorgeht und was die einzelnen Parameter im Detail bedeuten [Mar09, 167] - dennoch funktionieren sie. Dies ist im Hidden Markov Model anders, da es sich bei den Parametern, nicht um abstrakte Gewichte, sondern um Wahrscheinlichkeiten handelt.

Rekurrente (tief vorwärtsgerichtete) Neuronale Netze können dank schnellerer CPU oder gar GPU-Nutzung, immer bessere Ergebnisse mit kurzer Rechenzeit liefern. Neuronale Netz benötigen sehr viele Daten und Rechenzeit für das Training. Weiterhin können sie nicht mit sequentiellen Daten umgehen, was eine Datenvorverarbeitung nötig macht. Dies gilt ebenso für beiden folgenden Ansätze.

Dennoch bleibt das neuronal Netz eines der beliebsten Machine Learning Ansätzen. Darum exisitert für die Spracherkennung ein hybrider Ansatz, wie ihn [BM94] beschreibt, bei dem jeder Zustand einen Eingang in einem Neuronalen Netz bedient.

4.2 Support Vector Machine

Support Vector Machines eigenen sich für die Erkennung von Handschriften, zur klassifizierung von Bildern oder in der Bioinformatik zur klassifizierung von Proteinen.

Probleme ergeben sich, wenn sich die Daten nicht linear Separieren lassen. Hier wendet die SVM den sog. Kernel-Trick an. Dieser überfürht die Daten in einen höher-dimensionalen Raum, sodass sie wieder "eindeutig" Trennbar sind (Ausreißer werden vernachlässigt). Die SVM ist ebenfalls ein populärer und sehr guter Machine Learning Algorithmus, obwohl er, wegen aufwendiger Berechnungen, nicht für große Datenmengen geeignet ist [Mar09, 119]. Hier kann das HMM mit seiner besseren Performance punkten. Dennoch hat die SVM stärken in der benötigten Anzahl von Trainingsdaten.

Auch hier existieren Ansätze, die HMMs und SVMs bei der Spracherkennung verbinden. Hierbei wird die zeitliche Variabiliät (sequentielle Daten) mit dem HMM modeliert und die akustische Komponente mit einer SVM, die hier deutlich robuster funktioniert (Siehe [Stu08]).

4.3 k-Means

Der k-Means Algorithmus wird häufig in der Bildverarbeitung zur Segementierung eingesetzt. Oftmals wird reicht die euklidische Distanz nicht aus und es müssen weitere Merkmale hinzugezogen werden. Beispielsweise nutzt OpenCV eine Form des k-Means-Algorithmus, um Vordergrund von Hintergrund zu trennen oder Objekte zu erkennen.

Der k-Means-Algorithmus findet aber nicht unbedingt die beste "globale" Lösung, sondern findet je nach Wahl der Startpunkte nur lokale Lösungen. Weiterhin wird im Vorfeld der Anzahl der Cluster bestimmt, welche das Ergebnis stark beeinflussen kann.

Der k-Means Algorithmus ist, wie das Hidden Markov Model, ein Vertreter des unüberwachten Lernens - wohin gegen die beiden vorherigen Ansätze überwachtes Lernen umsetzen.

Auch für k-Means und das HMM existieren gemeinschaftliche Anwendungen, bspw. werden sie in einem hybriden schreiberunabhängigen Schrifterkennungssystem genutzt, bei dem der k-Means Algorithmus das Clustering im Vorfeld übernimmt und das HMM für die Erkennung zuständig ist [PC00].

5 Zusammenfassung

Wir haben in der vorgelegte Arbeit Grundlagen des Machine Learnings und der Wahrscheinlichekitsrechnung kennen gelernt. Weiterhin wurde mit Hilfe der Statistik die Markov Kette erläutert, die dann im Hidden Markov Model mündet und die Brücke zum Machine Learning schlägt.

Der Vegleich des HMM mit anderen Ansätzen gestaltet sich schwierig, da es für alle Anwendungen und vorliegenden Daten, Experten gibt und diese oftmals nur durch ausprobieren eindeutig bestimmt werden können.

Im folgenden Abschnitt werde ich ein kurzes Fazit ziehen.

5.1 Fazit

Das Hidden Markov Model eigent sich besonders für die Verarbeitung von sequentiellen Daten (Bspw. Sprache). Hier kann es, durch das Verweilen in einem internen Zustand, auch mit Verlängerungen eines Phonems in der gesprochenen Sprache sehr gut umgehen. Das mathematische Konzept hinter dem HMM ist sehr gut erforscht und formal definiert. Weiterhin sind Wahrscheinlichkeiten Größen, die ohne weiteres verstanden werden können. Dennoch benötigt man für das Verständnis von statistischen Verfahren, wie dem Hidden Markov Model, Kenntnisse von Statistik (Siehe Abschnit 1.2.2), was den Einstieg unter Umständen erschwert.

Für die Spracherkennung, vor allem der sequentiellen Komponente, eigent sich das Hidden Markov Model hervorragend. Dennoch exisitieren es viele hybride Ansätze 4, die sich die Vorteile von verschiedenen Algorithmen zu Nutze machen.

Literatur

[AD77] D.B. Rubin A.P. Dempster, N.M. Laird. Maximum-Likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 1977.

[BGV92] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 144–152, New York, NY, USA, 1992. ACM.

[BM94] H. Bourlard and N. Morgan. Connectionist speech recognition. a hybrid approach. *Kluwer international series in engineering and computer science*, 1994.

[BP66] Leonard E. Baum and Ted Petrie. Statistical inference for probabilistic functions of finite state markov chains. *Ann. Math. Statist.*, 37(6):1554–1563, 12 1966.

[Eul05] S. Euler. *Grundkurs Spracherkennung*. Friedr. Vieweg und Sohn Verlag - GWV Fachverlage GmbH, Wiesbaden, 2005.

[Fin03] Gernot A. Fink. *Mustererkennung mit Markov-Modellen*. B. G. Teubner Verlag, 2003.

[JR90] Biing-Hwang Juang and L. R. Rabiner. The segmental K-means algorithm for estimating parameters of hidden Markov models. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38:1639–1641, 1990.

[KHW13] U.M. Stocker K.-H. Waldmann. *Stochastische Modelle - Eine anwendungsorientierte Einführung*. Springer-Verlag, Berlin Heidelberg, 2013.

[Mar13] A.A. Markov. Example of a statistical investigation of the text of "eugene onegin" illustrating the dependence between samples in chain., 1913.

[Mar09] Stephen Marsland. *Machine Learning - An Algorithmic Perspective.* Chapman I& Hall, 2009.

[MP88] Warren S. McCulloch and Walter Pitts. Neurocomputing: Foundations of research. chapter A Logical Calculus of the Ideas Immanent in Nervous Activity, pages 15–27. MIT Press, Cambridge, MA, USA, 1988.

[PC00] Michael P. Perrone and Scott D. Connella. K-means clustering for hidden markov models. 2000.

[Rab89] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, pages 257–286, 1989.

[Ste57] Hugo Steinhaus. Sur la division des corps matériels en parties. *Bull. Acad. Polon. Sci.*, 12:801–804, 1957.

[Stu08] André Stuhlsatz. *Hybride Spracherkennung: Eine HMM/SVM-Systemintegration.* VDM Verlag Dr. Müller, 2008.

[Vit06] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Inf. Theor.*, 13(2):260–269, September 2006.