Praktikumsbericht

Buote Xu

18. April 2012

Zusammenfassung

Eine Zusammenfassung der implementierten Methoden im Bereich "Graph-Edit-Distance", ihre Anwendung auf generierten sowie echten Daten und ein Ausblick für weitere Entwicklungsmöglichkeiten.

1 Einführung

Aufgabe des Praktikums war, ein oder mehrere Algorithmen aus [1] zu implementieren, um ihre Tauglichkeit auf einigen Graphtypen zu überprüfen, dabei wurde vor allem die Methode basierend auf *Hidden Markov Models* evaluiert. Es hat sich herausgestellt, dass die vorgeschlagene Distanz sich sehr instabil gegenüber kleinen Veränderungen an den Daten zeigt, weswegen auf Basis von [3] und [4] eine neue Methode 3.2.1 entwickelt wurde. Die beiden implementierten Algorithmen werden im Folgenden beschrieben und evaluiert. Die verwendete Programmiersprache ist C++.

2 EDH-Based GED

2.1 Beschreibung

[1, EDH-Based GED] Dieser Algorithmus ist in zwei Phasen aufgeteilt und dient dazu, Graphen anhand ihrer Kanten zu beschreiben.

1. Erstelle ein Histogramm, um die verschiedenen Kanten eines Graphen zu kategorisieren. Die von den Autoren verwendete Methode nutzt hierfür die bekannten kartesischen Raumwinkel. Dabei ist es prinzipiell egal (und so auch implementiert), ob der Winkel ein Label der Kante an sich ist oder über die Position ihrer Knoten bestimmt werden kann.

2. Finde die günstigste Zuordnung zweier Histogramme zueinander; hierfür wurde nur der zweidimensionale Fall, beschrieben in [8, Appendix A] implementiert, da durch die Periodizität der Winkel ($360° = 0°$) ein sinnvolles, rotations-invariantes Matching berechenbar ist.

2.2 Implementierung

Um eine möglichst generische Benutzung zu ermöglichen, wurde die Boost Graph Library [9] verwendet, dabei ist ein sehr generischer und bedauerlicherweise auch sehr schwer lesbarer Code entstanden. Features:

- Unterstützung für beliebige Boost-Graphs, wenn für die Knoten jeweils eine "location" definiert wird.
- In der Lage, die Ähnlichkeit/Distanz zwischen zwei eindimensionalen Histogrammen (also zweidimensionalen Daten) zu berechnen
- Es ist möglich, auch andere Methoden zur Berechnung von Winkeln hinzuzufügen (z.B. für Geodaten), wenn man die passende Klasse schreibt.

2.3 Kommentar

Der Algorithmus hat einige gute Eigenschaften, so ist er invariant gegenüber Rotation, wegen der Normalisierung auch gegen Streckung, dies könnte theoretisch bei der Analyse von Buchstaben/Schrift von Vorteil sein, dazu ist er zumindest in zwei Dimensionen sehr flott: Die Laufzeit beträgt (O(E)), also linear in der Kantenmenge.

Eine Spiegelung der Daten hingegen führt dazu, dass sich das Histogram umdreht - genau hier existiert ein Problem des Ansatzes; ein matching zwischen zwei Histogrammen wie vorgeschlagen hat wenige Freiheiten.

So ist es nur möglich, Winkel linear einander zuzuordnen - wenn also z.B. die Winkel ($0^\circ_A = 30^\circ_B$) der Graphen A und B zueinander passen, so wird damit automatisch auch festgelegt, dass ($40^\circ_A = 70^\circ_B$) gilt - die Histogramme werden also quasi bestmöglich zueinander *verschoben*, was bei der genannten Spiegelung oder auch einer Zerrung eben nicht erfolgreich sein kann.

Ein weiteres Problem ist die Skalierbarkeit in höhere Dimensionen: Schon bei 3 Dimensionen hat man es mit 2-dimensionalen Histogrammen zu tun, die größtenteils leer sind (je nach Auflösung der Histogramme), so dass sinnvolle Zuordnungen schwer gefunden werden können. Aus diesen Gründen wurde das Hauptaugenmerk auf den nächsten Algorithmus gelegt.

3 HMM-Based GED

[1, HMM-Based GED] Auch dies ist ein zweigeteilter Algorithmus, HMM zur Modellierung eines Graphen und Distanzen um diese Modelle miteinander zu vergleichen.

3.1 HMM

3.1.1 Beschreibung

HMMs oder genauer *Hidden Markov Models* werden oft in der Sprach- und Bildverarbeitung genutzt, dennoch eine kurze Erklärung, für Details ist [2] zu empfehlen.

Für einen Informatiker ist die naheliegendste Struktur wohl eine *Finite State Machine*, die in jedem *Zeit*-schritt mit einer gewissen Wahrscheinlichkeit ein Symbol ausgibt.

Es ist möglich, damit mehrere Problemstellungen zu bewältigen; in dem vorliegenden Fall ist das Ziel folgendes:

Angenommen, ein *Hidden Markov Model* hat eine Folge von Symbolen (also z.B. Atome in einem Protein, oder abstrakter, Punkte im $\mathbb{R}^3$ ausgegeben -

welches waren die wahrscheinlichsten Parameter? (Mit welchem *State* wurde begonnen, wie sind die Transitionswahrscheinlichkeiten, mit welcher Wahrscheinlichkeit gibt *State* 3 das Symbol x aus?

Da es im $\mathbb{R}^3$ eine unendliche Anzahl an möglichen Symbolen gibt (vgl. hierzu ein Model, das nur die Symbole *Sonne* und *Regen* kennt, und das Wetter in Mannheim grob beschreiben soll), ist es notwendig für jeden State eine *Wahrscheinlichkeitsverteilung* zu finden.

Dies geschieht über *Gaussian Mixture Models*, die eine gewichtete Ansammlung von *Gaussian Models*, also Glockenkurven-Verteilungen sind. So wird also jedem *State* genau ein *Gaussian Mixture Model* zugewiesen. Um vernünftige Startwerte für ein mögliches Modell zu erzeugen, wird gemäß [1] ein [5] *Expectation Maximization*-Algorithmus angewandt, der für eine gegebene Punktmenge das wahrscheinlichste *Gaussian Mixture Model* berechnet - dabei ist zu bemerken, dass das perfekte Modell nicht gefunden werden kann und somit mögliche Lösungen iterativ erzeugt werden bis sie sich nicht mehr merklich verbessern.

Nach Erzeugung von HMMs für zwei Graphen gibt es verschiedene [4], [1] Ähnlichkeits-/Distanzmaße, die später vorgestellt werden.

Fragen:

- **Welches sind die States in einem Graphen?** Die Autoren in [1] schlagen vor, einen *KMeans* anzuwenden, um dann jedem einzelnen Cluster einen State zuzuordnen. Dies ist ein sinnvoller Ansatz, um die *GMMs* lokal einzuschränken.

- **Inwiefern spielen Graphkanten eine Rolle?** Die einzige Möglichkeit, diese mit einzubeziehen, ist während des *Clustering*; so kann z.B. der [10] *Chinese Whisper*-Algorithmus verwendet werden, der vor allem auf dicht besetzten Graphen zu sinnvollen Ergebnissen führt.

- **Wie ist die zeitliche Ordnung von stationären, also z.B. Proteindaten?** Hier kommen wir zu einem Problem des ursprünglichen Algorithmus in [1]: Durch Vertauschung der Reihenfolge von wenigen Punkten erhält man unendliche Distanzen, da *HMMs* primär für die Beschreibung von zeitabhängigen Daten geeignet sind, es gibt aber auch für stationäre Daten bessere Distanzmaße die stabiler gegenüber solchen Änderungen sind.

3.1.2 Implementierung

Da es sich vornehmlich um Matrix-Operationen handelt, wurde *Armadillo* [6] eingesetzt, eine hinreichend schnelle Bibliothek, die zusammen mit *Atlas* oder *Lapack* die nötige Lineare Algebra bereitstellt. Daten-Punkte werden hierbei als Spalten von Armadillo-Matrizen übergeben, für *pdb*-Dateien ist unter den *examples* ein Parser beigelegt, der die Atome aus ebendiesen extrahiert und passende Matrizen erzeugt.

Aufgrund mangelnder Open-Source-Implementierungen von *HMMs*, die auf *GMMs* basieren, wurden diese im Zuge des Praktikums in Anlehnung an [5] und [2] geschrieben und über *Doxygen* dokumentiert.

Features:

- *Geschwindigkeit:* Im Laufe der Entwicklung haben sich einige Stellen zur Optimierung aufgetan, sodass nun für die Parametersuche eine HMM über den *Baum-Welch*-Algorithmus zusätzlich eine Methode angeboten wird, die durch Caching einiger Wahrscheinlichkeiten sowie günstiger Matrix-Multiplikationen sich 5-10x schneller als die naive, wenn auch speichergünstigere Variante zeigt. Die Laufzeitkomplexität ist, aufgrund eines sehr schnellen Konvergenzverhaltens der iterativen Methoden (10-20 Schritte) als linear in der Anzahl der Datenpunkte zu betrachten.
- *Kompatibilität:* Er werden Objekte im $\mathbb{R}^n$ unterstützt, da die unterliegenden GMMs nur hierfür sinnvoll definiert sind. Es wäre denkbar, für diskrete Symbole (z.B. Angebot eines Online-Versenders) eine passende Template-Version anzubieten, dies ist aber nicht Inhalt dieser Arbeit.
- *Numerische Stabilität:* Wie in [2] beschrieben, wurden in dieser Implementierung normierte Wahrscheinlichkeiten verwendet, da ansonsten die Multiplikation von kleinen Zahlen zur völligen Auslöschung führen würde. Dabei ist zu beachten, dass im ursprünglichen Werk Fehler enthalten waren welche in [11] behoben wurden.

3.2 Distanzen und Ähnlichkeiten

Um eine Angabe über die Ähnlichkeit zweier berechneter HMMs zu machen, wird ein Maß verwendet, um die enthaltenen Wahrscheinlichkeitsverteilungen der beiden miteinander zu vergleichen - wenn sich diese ähneln, so auch die damit beschriebenen Daten.

3.2.1 KL-Divergenz

Ein häufig verwendetes Maß für zwei Verteilungen P, Q ist die *Kullback-Leibler*-Divergenz:

$$KL(P,Q) = \int P(x) \log \frac{P(x)}{Q(x)} \, \mathrm{d}x$$

Diese hat jedoch im konkreten Fall einige Probleme, da sie über ein Integral definiert ist, welches für kompliziertere Verteilungen nur näherungsweise berechnet werden kann. Hier ist zu bemerken, dass bei der Modellierung eines Datensatzes aufgrund der vorherigen Einteilung in wenig-überlappende Cluster (mit KMeans z.B.) *GMMs* entstehen, die sich stark voneinander unterscheiden. Ein Datenpunkt kann somit oftmals nur von einer einzigen GMM (und damit einem *State*, vgl. 3.1.1) erzeugt werden, was bei der Distanzberechnung zu Problemen führt.

Bemerkung: Die KLD zwischen zwei unabhängigen Verteilungen ist quasi unendlich aufgrund des log, wenn $\frac{P(x)}{Q(x)} \approx 0$

KLD-Berechnung nach [1] Die dort vorgeschlagene Näherung hat genau das Problem, dass nicht alle States $(S_i)_i$, $(T_j)_j$ zweier HMMs miteinander verglichen werden, sondern nur S_1 mit T_1, S_2 mit T_2 etc., so kann es bei einer ungünstigen Nummerierung dazu kommen, dass die resultierende Distanz unendlich ist - es ist eher erforderlich, die jeweils nächstliegenden States einander

zuzuordnen. Weiterhin zeigt sich der Algorithmus absolut instabil gegenüber kleinen Änderungen an der Inputreihenfolge, was ihn z.B. für Proteinanalyse vollkommen untauglich macht.

Überhaupt kriegt er Schwierigkeiten, wenn "zu viele" Daten vorhanden sind, so dass die enstehenden Verteilungen zu einseitig sind (z.B. wenn die Wahrscheinlichkeit, im State 4 zu starten bei 100% liegt).

Es hat sich herausgestellt, dass der umgekehrte Ansatz, die Ähnlichkeit statt der Distanz/Divergenz zu berechnen, vielversprechender ist, wie sich im nächsten Algorithmus zeigt.

RandomWalk [4] Dieser Ansatz führt prinzipiell sehr ähnliche Berechnungen durch - da aber ein entgegengesetztes Maß berechnet werden soll (also Ähnlichkeit statt Distanz), kann e^{-d} für eine Distanz d berechnet werden, um zusammen mit der KLD-Näherung in [3] Werte zu erhalten, die zwischen 0 und 1 liegen.

Somit kann man für die States $(S_i)_i$, $(T_j)_j$ eine Matrix mit $i * j$ Einträgen erstellen, die die jeweiligen Ähnlichkeiten beschreibt.

Das resultierende Ähnlichkeitsmaß zweier HMMs hat folgende Eigenschaften:

- *Wertebereich:* Ursprünglich zwischen 0 und 1, um aber sinnvollere Ergebnisse zu erhalten fließen im Gegensatz zu [4] nun die absoluten Ähnlichkeiten zwischen den States mit ein, was bei extrem ähnlichen HMMs (also z.B. wenn sie übereinstimmen) zu geringfügig höheren Ergebnissen wie 1.3 führen können.

- *Geschwindigkeit:* Linear bezüglich der Anzahl an verwendeten Gaussian Models, also unerheblich im Vergleich zur HMM-Berechnung

- *Interpretierbarkeit:* Aus den Zwischenergebnissen kann man z.B. über die *Ungarische Methode* ein bestmögliches Mapping zwischen den States erhalten, dies wurde aus Zeitgründen jedoch nicht mehr implementiert.

- *Invarianz:* Basierend auf der Idee von stationären Wahrscheinlichkeiten in [4] (diese geben an, welche States eher besucht werden) ist es möglich und sinnvoll, etwaige Veränderungen der Ausgangsdaten zu *reverse engineeren*. So ist eine Methode enthalten, die mit Hilfe von Clustermittelpunkten lineare Transformationen durchführt, um zwei HMMs soweit wie möglich aneinander anzupassen. Dadurch wird der Algorithmus stabil (also 100% Übereinstimmung zwischen den veränderten Daten) gegenüber Spiegelung, Streckung und Drehung, jedoch nicht Verschiebungen - Daten sollten deshalb vor der HMM-Berechnung *zentriert* werden.

- *Abhängigkeit von der Inputreihenfolge:* Diese ist immernoch gegeben, jedoch nicht so stark ausgeprägt wie beim ursprünglichen Ansatz, sodass z.B. zwischen einem Protein und dessen gemischten Äquivalent noch eine wesentlich höhere Ähnlichkeit besteht als einem anderem Protein.

3.3 Kommentar

Am Ende ist ein schneller, stabiler, interpretierbarer Algorithmus entstanden, der so natürlich auch für ein Klassifizierungen verwendet werden kann. Wegen

dem Einsatz verschiedener Wahrscheinlichkeitsverteilungen und statistischen Vorgängen ist er für kleine Graphen (also in der Größenordnung bis 100 Knoten) ungeeignet, dafür skaliert er aber auch noch gut mit größeren Mengen. Weitere Anwendungsideen wären Text- und Sprachanalysen, wo HMMs häufig verwendet werden, dafür müssten allerdings noch die unterliegenden GMMs angepasst werden.

Literatur

[1] X.-B. Gao, B. Xiao et. al. A Comparative Study of Three Graph Edit Distance Algorithms. Foundations on Computational Intelligence (Edited by A. Abraham, Aboul-Ella Hassanien et al.), ISBN: 978-3-642-01535-9, Springer, Vol. 5, SCI 205, pp. 223-242, 2009.

[2] L. R. Rabiner (1989). Ä Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition,"Proceedings of the IEEE, vol 77, no 2, 257–287.

[3] Goldberger, Gordon, Greenspan. An Efficient Image Similarity Measure Based on Approximations of KL-Divergence Between Two Gaussian Mixtures, Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on In Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on (2003), pp. 487-493 vol.1, doi:10.1109/ICCV.2003.1238387

[4] A Novel Low-Complexity HMM Similarity Measure. Sayed Mohammad Ebrahim Sahraeian, Student Member, IEEE, and Byung-Jun Yoon, Member, IEEE

[5] Unsupersived Learning of Finite Mixture Models. Mario A.T. Figueiredo and Anil K. Jain

[6] Conrad Sanderson. Armadillo: An Open Source C++ Linear Algebra Library for Fast Prototyping and Computationally Intensive Experiments. Technical Report, NICTA, 2010.

[7] Loriot S, Cazals F, Bernauer J: ESBTL: efficient PDB parser and data structure for the structural and geometric analysis of biological macromolecules. Bioinformatics 2010, 26(8):1127-1128. PMID: 20185407

[8] A Linear Time Histogram Metric for Improved SIFT Matching. Ofir Pele and Michael Werman.

[9] The Boost Graph Library. Jeremy Siek, Lie-Quan Lee, Andrew Lumsdaine.

[10] Chinese Whispers - an Efficient Graph Clustering Algorithm and its Application to Natural Language Processing Problems. Christian Biemann

[11] An Erratum for “A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition”. Ali Rahimi