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 2.2.2 entwickelt wurde.

2 Liste der Algorithmen

2.1 EDH-Based GED

2.1.1 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.

2.1.2 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.

2.1.3 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.

2.2 HMM-Based GED

2.2.1 Beschreibung

[1, HMM-Based GED] *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. 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.

2.2.2 Implementierung

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