Analyse des UMAP Verfahrens

Christopher Reiners

Geboren am 9. April 1998 in Detmold

6. August 2019

Bachelorarbeit Mathematik

Betreuer: Prof. Dr. Jochen Garcke

Zweitgutachter: Dr. Bastian Bohn

Mathematisches Institut für Numerische Simulation

Mathematisch-Naturwissenschaftliche Fakultät der

Rheinischen Friedrich-Wilhelms-Universität Bonn

Danksagung

An dieser Stelle möchte ich gerne Prof. Dr. Jochen Garcke, für die Vergabe dieses sehr interessanten und spannenden Themas, danken. Besonderer Dank gilt Leland McInnes für das persönliche Gespräch in Ottawa, Kanada. So konnte ich die Motivation, welche er für das UMAP Verfahren hatte, aus erster Hand erfahren.

Gerne möchte ich auch meinen Freunden danken welche mich im Laufe der Studienzeit, besonders während der Bachelorarbeit, begleitet haben. Anna für die vielen Eispause und die lustige Unterhaltung auch spät in der Nacht, Tobi, Hendrik und Lennard für die Motivation und den mathematischen Austausch und Lukas für die hilfreichen Gespräche. Ohne euch wäre diese Arbeit nicht entstanden.

Zum Schluss meiner Familie für Ihre bedingungslose Unterstützung, trotz Einsilbigkeit meiner Antworten in den letzten Wochen meiner Arbeit.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Wer eine Tageszeitung aufschlägt oder sie beim Frühstücken auf seinem Tablett liest, wird nicht vermeiden können, mindestens einen Artikel über die „Algorithmen der Zukunft“, „Daten hungrigen Konzerne“, ... zu lesen. Fakt ist heutzutage werden mehr Daten generiert und gespeichert als je zuvor in der Geschichte der Menschheit und wir benötigen aussagekräftige, schnelle und zuverlässige Verfahren um die gesammelten Daten auszuwerten, zu strukturieren und einen Mehrwert für die Gesellschaft zu generieren.

Wir werden uns in dieser Arbeit mit einer Klasse an Verfahren befassen, welche hochdimensionale Daten visualisiert. Insbesondere werden wir das UMAP (Uniform Manifold Approximation and Projection) Verfahren

Datenanalyse

Eine neuere Form der Datenanalyse - die topologische Datenanalyse (*kurz: TDA*) - nutzt mathematische Instrumente des Teilgebiets der Topologie (*griechisch: Lehre vom Ort/Platz*) um Daten zu beschreiben und zu strukturieren.

Wir beschreiben die Situation einen Datensatz X zu analysieren wie folgt. Wir betrachten einen D-dimensionalen Raum und ein Objekt K. In unserem Fall werden wir uns größtenteils mit dem euklidischen Raum $\mathbb{R}^D$ versehen mit der euklidischen Norm $\|\cdot\|$ beschäftigen.

K kann beispielsweise eine geschlossene Menge sein. Die genaue Struktur von K bleibt uns allerdings unbekannt. Später werden wir argumentieren, dass K gewisse Regularitätseigenschaften erfüllt und in vielen Fällen lokal einem niedrigdimensionalen Raum $\mathbb{R}^d, (d \ll D)$ gleicht.

Statt K ist uns eine endliche Menge an N Punkten $X = \{x_i\}_{i=1}^N,\ (x_i \in \mathbb{R}^D)$, gegeben. X wird als *Punktwolke* bezeichnet und beschreibt in einem Experiment gemessene Daten, beispielsweise Sensormessdaten, biologische Informationen oder Bilddatensätze.

In Kapitel **??** werden wir uns mit einem Bilddatensatz mit 100 000 Farbbildern mit einer Auflösung von 300×300 Pixeln beschäftigen. Wir können diesen Datensatz als $N = 100\,000$-elementige Punktwolke im $\mathbb{R}^{300\times300\times4}$ auffassen, wobei wir die vier Farbkanäle der Bilder berücksichtigen.

Wir gehen dabei davon aus, dass K und X in einem gewissen Sinne „ähnlich“ sind. Nun ist es Ziel der TDA mittels Methoden der Topologie Aussagen über die Struktur von X zu treffen, welche, aufgrund der Ähnlichkeit, auch für K gelten sollen. Die K zugrundeliegende Struktur kann uns dann dabei helfen Aussagen über weitere gemessene Daten zu treffen, da diese auch in der uns unbekannten Menge K liegen sollten. Zusätzlich hilft

Dimensionsreduktion / Problemstellung

Sei $X = \{x_i\}_{i=1}^N$, $(x_i \in \mathbb{R}^D)$ wir bezeichnen D als die Dimension unserer Daten, beziehungsweise als die Anzahl der gemessenen Eigenschaften. N ist die Anzahl der verfügbaren Datenpunkte. In der Praxis können D und N sehr groß sein. So gilt für den Bilddatensatz welchen wir in Kapitel **??** betrachten werden, $N = 100\,000, D = 360\,000$.

Eine Herangehensweise die d-dimensionale Repräsentation zu finden ist es eine $N \times N$ Matrix M aufzustellen, die ersten d Eigenvektoren von M, wobei die Eigenvektoren absteigend nach der Größe der zugehörigen Eigenvektoren geordnet sind, geben

Hier sollen die zwei Arten an DR Algorithmen vorgestellt werden (Matrix-Faktorisierung und Graph Layout). Zusätzlich sollen PCA, Isomap, Laplacian Eigenmaps und t-SNE vorgestellt werden.

Die meisten Verfahren zur Dimensionsreduktion beruhen auf der Annahme, das reale hochdimensionale Daten sich in der Umgebung einer niedrigdimensionalen Mannigfaltigkeit konzentrieren. In der Literatur ist diese Annahme als Mannigfaltigkeit-Hypothese (*engl.: manifold hypothesis*) bekannt [**Mitter**, **Rifai**]. Ansätze um einen gegebenen Datensatz auf diese Annahme zu testen finden sich in [**Fefferman**].

Ziele der Arbeit

Eigene Beiträge

Wir möchten nun die Ziele dieser Arbeit formulieren.

- Einführung in die grundlegenden Werkzeuge der topologischen Datenanalyse
- Mögliche Schritte wie die Lücke zwischen Theorie und Praxis des UMAP Verfahrens geschlossen werden kann
- UMAP anhand sinnvoller Datensätze mit anderen Dimensionsreduktionsverfahren vergleichen
-

Insbesondere werden wir die in Kapitel 2 von [**UMAP**] beschriebene Theorie ausführlicher beschreiben und in einen allgemeineren Kontext setzen.

Aufbau der Arbeit

In Kapitel **??** werden wir die theoretische Grundlage des UMAP Verfahrens erklären. Dies wird einige Definitionen und Grundlagen aus der Kategorientheorie und der (Algebraischen-)Topologie erfordern. Wir haben mich bemüht diese möglichst vollständig darzustellen. Für zusätzliche Informationen empfiehlt sich [**Brandenburg**, **Barr**, **Munkres**].

Kapitel **??** wird die Theorie des UMAP Verfahrens darstellen. Dabei werden wir auf die in **??** gelegten Grundlagen zurückgreifen. Zusätzlich soll argumentiert werden, dass eine praktische Implementierung wie sie in der Theorie beschrieben ist zu rechenaufwendig ist und somit wenig praktischen nutzen hat. Deshalb

Vergleich mit Original Paper

Kapitel 2

Grundlagen

Um die Geometrie „mathematischer Räume“ zu beschreiben, ist es aus topologischer Sicht ausreichend, die „Löcher“ der Räume zu charakterisieren. Um diese Aussage zu formalisieren, und damit die Grundlage für das UMAP-Verfahren zu legen, werden wir einige Begriffe aus der (algebraischen) Topologie benötigen.

Die grundlegenden Definitionen *topologischer und metrischer Räume* werden uns helfen *(riemannsche) Mannigfaltigkeiten* einzuführen. Der Begriff der Mannigfaltigkeit formalisiert den niedrigdimensionalen Raum auf welchem unsere Daten liegen.

Die geometrische Struktur dieser Räume werden wir mit Hilfe der *Homologie* einführen und sehen wie diese visuelle Eigenschaften beschreibt. Um diese strukturelle Darstellung auf eine endliche Punktwolke anwenden zu können, werden wir den Begriff der *persistenten Homologie* einführen müssen.

Die Begriffe und benötigten Aussagen werden vollständig und anschaulich eingeführt. An einigen Stellen werden wir den Leser auf geeignete ergänzende Literatur verweisen.

2.1 Topologische Räume

Der Grundlegende Begriff der Topologie ist der des topologischen Raumes.

Definition 2.1 (Topologischer Raum)**.** Eine Topologie ist ein Mengensystem T ...

Man kann den Begriff erweitern indem man einen Abstandsbegriff auf der Menge X definiert. Dies führt uns zum metrischen Raum.

Definition 2.2 (Metrischer Raum)**.** Eine Metrik ist eine Abbildung,...

In der Einleitung bereits erwähnt, werden wir Annehmen, dass unsere Daten $X = \{x_i\}_{i=1}^N, (x_i \in \mathbb{R}^D)$ mit D gemessenen Eigenschaften mittels $d, (d \ll D)$ Eigenschaften dargestellt werden können. Dies werden wir formalisieren indem wir Mannigfaltigkeiten einführen. Anschaulich ist eine d-dimensionale Mannigfaltigkeit ein topologischer Raum welcher lokal dem euklidischen Raum $\mathbb{R}^d$ gleicht.

Definition 2.3 (Mannigfaltigkeit)**.** Sei $\mathcal{M}$ ein topologischer Raum,...

Bemerkung. Eine *differenzierbare* Mannigfaltigkeit ist eine...

Beispiel 2.1. S^n ist eine n-dimensionale Mannigfaltigkeit im $\mathbb{R}^{n+1}$.

Unter der Annahme, dass wir nur d Eigenschaften benötigen um unsere Daten X darzustellen, können wir sagen, das X eine d-dimensionale Mannigfaltigkeit im $\mathbb{R}^D$ ist. Wir können eine Mannigfaltigkeit $\mathcal{M}$ mit einem Abstandbegriff erweitern. Diese zusätzliche Regularitätseigenschaft ermöglicht es uns von Distanzen, Winkeln und Kurven auf $\mathcal{M}$ zu sprechen. Diesen Abstandsbegriff nennen wir *riemannsche Metrik* und er ist wie folgt definiert:

Definition 2.4 (Riemannsche Mannigfaltigkeit)**.** Sei g,... Ein Tupel $(\mathcal{M}, g)$, wobei $\mathcal{M}$ eine Mannigfaltigkeit und g eine riemannsche Metrik ist heißt *riemannsche Mannigfaltigkeit.*

Bemerkung. Eine riemannsche Mannigfaltigkeit ist stets metrisierbar im Sinne, das ...

Bemerkung. Es ist sinnvoll in unserem Fall X als riemannsche Mannigfaltigkeit zu betrachten, da wir erwarten, wenn x_i Nahe bei x_j und x_k liegt, dass x_j auch Nahe bei x_j liegt. Wenn X einen Bilddatensatz beschreibt, nehmen wir an, dass eine kleine Veränderung im Bild den Inhalt nicht wesentlich ändert.

Die Riemannmetrik beschreibt eine Distanz auf der Mannigfaltigkeit. Die Länge eines kürzesten Weges auf $\mathcal{M}$ zwischen zwei Punkten $p, q \in \mathcal{M}$ wird als Geodäte bezeichnet und ist definiert als

Definition 2.5 (Geodäte)**.** Seien $p, q \in \mathcal{M}$ ...

Der Begriff der riemannschen Mannigfaltigkeit ermöglicht es uns unsere zentrale Annahme, das $X \subseteq \mathbb{R}^D$ einer niedrigdimensionalen Struktur entnommen ist, zu formalisieren. Wir möchten nun diese niedrigdimensionale Struktur genauer beschreiben.

Eine bekannte Aussage, welche der Topologie entstammt, ist, dass eine Kaffeetasse das Gleiche topologische Objekt wie ein Donut ist. Dies ist dadurch begründet, dass man eine Kaffeetasse durch strecken und stauchen (ohne reißen) in einen Donut transformieren kann. Diese „Gleichheit“ lässt sich wie folgt mathematisch darstellen:

Definition 2.6 (Stetige Abbildung)**.** Sei ...

Definition 2.7 (Homöomorphismus)**.** Sei ...

Nun werden wir den Begriff der Homologie einführen. Dafür werden wir den Begriff der simplizialen Komplexe benötigen um zuerst die simpliziale Homologie einzuführen.

Definition 2.8 (Simplizialer Komplex)**.** Sei ...

Definition 2.9 (Abstrakter simplizialer Komplex)**.** Sei ...

Definition 2.10 (Homologie)**.** Sei ...

Definition 2.11 (Homologiegruppe)**.** Sei ...

Beispiel 2.2. Die Homologiegruppen eines Balls, Donuts, Graphen, ...

Definition 2.12 (Filtration)**.** Sei $T \subseteq \mathbb{R}$, eine *Filtration* $\mathcal{X}$ über T ist eine Familie von topologischen Räumen $\{X_i\}_{i \in T}$, so dass $X_i \subseteq X_j$ für $i \leq j \in T$.

Bemerkung. Eine Filtration $\mathcal{X}$ wird meist nicht als Familie verschiedener topologischer Räume X_i angesehen, sondern als ein einziger Raum, welcher sich im Laufe der Zeit „transformiert“. Siehe dazu [**Oudot**]. Dies stimmt mit unserem Bild überein, den ϵ-Parameter eines Čech-Komplexes immer größer werden zu lassen. Wir werden *persistente Homologie* nutzen um die homologischen Eigenschaften zu beschreiben, welche für alle $i \in T$ erhalten (bzw. persistent) bleiben.

Beispiel 2.3. Insbesondere ist somit der Čech-Komplex eine Filtration, da ...

Beispiel 2.4. Eine weitere Filtration ist der Vietoris-Rips-Komplex

Bemerkung. Der VR-Komplex ist ein Beispiel eines Clique-Komplexes und wird als solcher eindeutig durch sein 1-Skelet identifiziert. Sei V ein VR-Komplex und V^1 das zugehörige 1-Skelet. Dann erhält man V aus V^1, indem man alle Cliquen im V^1 zugrundeliegenden Graphen bestimmt. Eine k-Clique ist ein vollständiger Subgraph mit k Knoten.

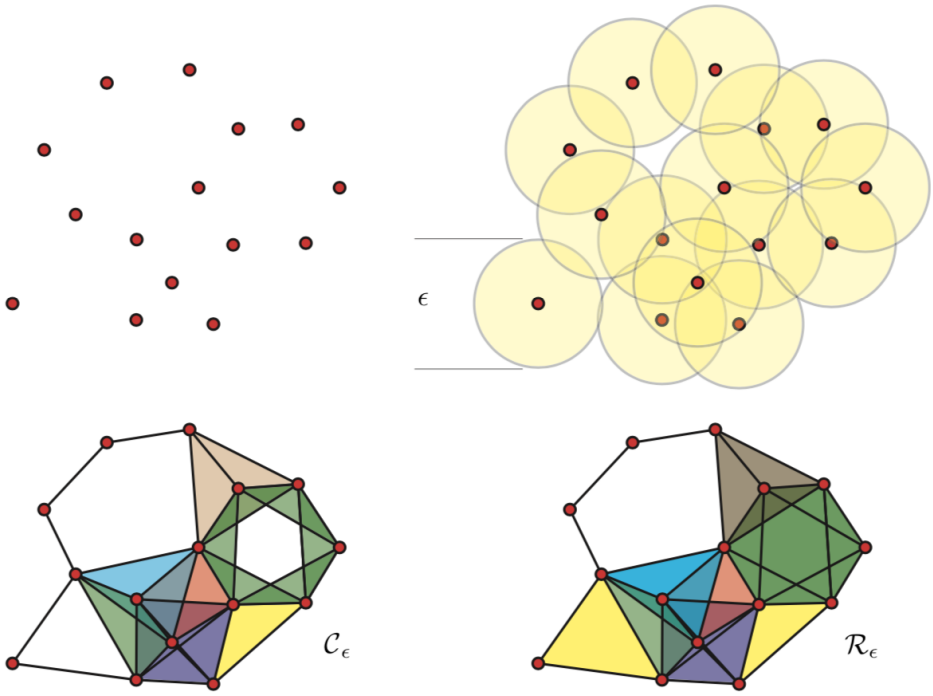

ABBILDUNG 2.1: Eine Punktwolke [oben links] kann zu einem Čech-Komplex [unten links] oder einem VR-Komplex [unten rechts] basierend auf dem Parameter ϵ konstruiert werden. Die Homotopietypen der $\epsilon/2$ Überdeckung vom Čech-Komplex $(S^1 \vee S^1 \vee S^1)$, während das VR-Komplex die Homotopietypen $(S^1 \vee S^2)$ besitzt. (*Quelle:* [**Barcodes**])

Es gibt weitere Formen von Komplexen, die bekanntesten sind die CW-Komplexe und Zellkomplexe. Ziel ist es stets mit Hilfe einfacher Konstrukte die Topologie eines Raums zu beschreiben.

Die vom Čech-Komplex und vom Vietoris-Rips-Komplex beschriebene Homologie kann sich unterscheiden.

Nerv Theorem. *Sei ...*

Wie wir sehen ist die Konstruktion stark abhängig vom ϵ-Parameter. Um dem entgegenzuwirken führt man die Idee der *persistenten Homologie* ein. Dabei betrachtet man wie sich die Homologie eines Raumes für eine monoton wachsende Folge $(\epsilon_i)_{i\in\mathbb{N}}$ verhält.

2.2 Kategorientheorie

Die für die mathematischen Grundlagen des UMAP Verfahren benötigten Definitionen werde ich mithilfe der Kategorientheorie einführen. Diese sehr abstrakte Form mathematische Objekte und Zusammenhänge zu formalisieren wurde erstmals in den 1940ern von Samuel Eilenberg und Saunders Mac Lane eingeführt.
Die Definitionen sind dem Buch von Brandenberg [**Brandenburg**] entnommen.

Definition 2.13 (Kategorie)**.** Eine Kategorie $\mathcal{C}$ besteht aus folgenden Daten:

1. Eine Klasse $Ob(\mathcal{C})$, deren Elemente wir *Objekte* nennen

2. zu je zwei Objekten $A, B \in Ob(\mathcal{C})$ einer Menge $\mathsf{Hom}_{\mathcal{C}}(A, B)$, deren Elemente wir mit $f : A \to B$ notieren und *Morphismen* von A nach B nennen,

3. zu je drei Objekten $A, B, C \in Ob(\mathcal{C})$ einer Abbildung

$$\mathsf{Hom}_{\mathcal{C}}(A, B) \times \mathsf{Hom}_{\mathcal{C}}(B, C) \to \mathsf{Hom}_{\mathcal{C}}(A, C)$$

die wir mit $(f, g) \mapsto g \circ f$ notieren und *Komposition von Morphismen* nennen,

4. zu jedem Objekt $A \in Ob(\mathcal{C})$ einen ausgezeichneten Morphismus

$$id_A \in \mathsf{Hom}_{\mathcal{C}}(A, A),$$

welchen wir die *Identität* von A nennen.

Diese Daten müssen den folgenden Regeln genügen:

1. Die Komposition von Morphismen ist *assoziativ*: Für drei Morphismen der Form $f : A \to B,\ g : B \to C,\ h : C \to D$ in $\mathcal{C}$ gilt

$$h \circ (g \circ f) = (h \circ g) \circ f$$

als Morphismen $A \to D$.

2. Die Identität sind *beidseitig neutral* bezüglich der Komposition: Für jeden Morphismus $f : A \to B$ in $\mathcal{C}$ gilt

$$f \circ id_A = f = id_A \circ f$$

Bemerkung. Anstelle von $A \in Ob(\mathcal{C})$ schreibt man meistens $A \in \mathcal{C}$. Falls die Kategorie $\mathcal{C}$ aus dem Kontext bekannt ist, werden wir $\mathsf{Hom}_{\mathcal{C}}(A, B)$ mit $\mathsf{Hom}(A, B)$ abkürzen.

Bemerkung. Eine Klasse ist eine Menge, welche zu groß ist um eine Menge zu sein. Für eine Definition einer Klasse verweisen wir auf [**Levy**]. In unseren Beispielen genügt die Vorstellung einer Menge. Meist ist $Ob(\mathcal{C})$ sogar eine Menge. Dann spricht man formal von einer strikten kleinen Kategorie.

Insbesondere kann man für eine Kategorie $\mathcal{C}$ und Objekte $A, B, C \in \mathcal{C}$ den *Hom-Funktor* definieren, indem man

$$\mathsf{Hom}(-, B) : \mathcal{C} \to \mathbf{Set} \tag{2.1}$$

betrachtet. Der Hom-Funktor bildet ein Objekt $A \in \mathcal{C}$ auf die Menge der Morphismen $\mathsf{Hom}(A, B)$ ab, und einen Morphismus $h : A \to C$ auf die Funktion

$$\mathsf{Hom}(h, B) : \mathsf{Hom}(C, B) \to \mathsf{Hom}(A, B), \text{wobei } g \mapsto g \circ h \text{ für } g \in \mathsf{Hom}(C, B) \tag{2.2}$$

Beispiel 2.5. *Passende Beispiele von Kategorien, welche später wieder genutzt werden.* **Top**, **Set**

Ein weiterer für die folgenden Definitionen wichtiger Begriff ist der der dualen Kategorie.

Definition 2.14 (Duale Kategorie)**.** Es sei $\mathcal{C}$ eine Kategorie. Dann können wir eine neue Kategorie $\mathcal{C}^{op}$ konstruieren: Sie besitzt dieselben Objekte wie $\mathcal{C}$, allerdings werden die Morphismen „umgedreht“: Für $A, B \in \mathcal{C}$ sei

$$\mathsf{Hom}_{\mathcal{C}^{op}}(A, B) \coloneqq \mathsf{Hom}_{\mathcal{C}}(B, A).$$

Die Identitäten verändern sich nicht. Die Komposition

$$\circ^{op} : \mathsf{Hom}_{\mathcal{C}^{op}}(A,B) \times \mathsf{Hom}_{\mathcal{C}^{op}}(B,C) \to \mathsf{Hom}_{\mathcal{C}^{op}}(A,C)$$

ist durch

$$\mathsf{Hom}_{\mathcal{C}}(B,A) \times \mathsf{Hom}_{\mathcal{C}}(C,B) \cong \mathsf{Hom}_{\mathcal{C}}(C,B) \times \mathsf{Hom}_{\mathcal{C}}(C,B) \times \mathsf{Hom}_{\mathcal{C}}(B,A) \xrightarrow{\circ} \mathsf{Hom}_{\mathcal{C}}(C,A)$$

definiert, d.h. $f \circ^{op} g \coloneqq g \circ f$. Auf diese Weise ist $\mathcal{C}^{op}$ tatsächlich eine Kategorie und heißt die *zu* $\mathcal{C}$ *duale Kategorie*.

Bemerkung. Eine Eigenschaft der dualen Kategorie ist, dass Aussagen, welche für alle Kategorien bewiesen wurden, auch für alle dualen Kategorien gelten.

Wir möchten nun den Begriff des Funktors zwischen zwei Kategorien einführen. Ein Funktor ordnet Objekte einer Kategorie $\mathcal{C}$ Objekten einer Kategorie $\mathcal{D}$ zu, und entsprechend für Morphismen. Insbesondere bleibt die Eigenschaft der Isomorphie zwischen zwei Objekten erhalten.

Definition 2.15 (Funktor)**.** Es seien $\mathcal{C}$ und $\mathcal{D}$ zwei Kategorien. Ein *Funktor*

$$F : \mathcal{C} \to \mathcal{D}$$

von $\mathcal{C}$ nach $\mathcal{D}$ besteht aus folgenden Daten:

1. für jedes Objekt $A \in \mathcal{C}$ ein Objekt $F(A) \in \mathcal{D}$,
2. für jeden Morphismus $f : A \to B$ in $\mathcal{C}$ einen Morphismus

$$F(f) : F(A) \to F(B)$$

in $\mathcal{D}$.

Dabei soll gelten:

1. Für jedes Objekt $A \in \mathcal{C}$ ist $F(id_A) = id_{F(A)}$.
2. Für je zwei Morphismen $f : A \to B,\ g : B \to C$ in $\mathcal{C}$ gilt in $\mathcal{D}$:

$$F(g \circ_{\mathcal{C}} f) = F(g) \circ_{\mathcal{D}} F(f)$$

Bemerkung. Bezüglich der Kategorie $\mathcal{C}$ ist ein Funktor $F : \mathcal{C} \to \mathcal{D}$ kovariant, während $F : \mathcal{C}^{op} \to \mathcal{D}$ kontravariant (bzgl. $\mathcal{C}$) ist.

Eine häufig verwendete Form eines kontravarianten Funktors ist die Prägarbe (*engl.: presheaf*). Wir werden diesen Funktor später verwenden um *simpliziale Mengen* einzuführen.

Definition 2.16 (Prägarbe)**.** Eine Prägarbe auf einer kleinen Kategorie $\mathcal{C}$ ist ein Funktor

$$F : \mathcal{C}^{op} \to \mathbf{Set}$$

von der dualen Kategorie $\mathcal{C}^{op}$ von $\mathcal{C}$ in die Kategorie **Set** von Mengen.

Definition 2.17 (Prägarbenkategorie)**.** Sei $\widehat{\mathcal{C}}$ die Prägarbenkategorie einer Kategorie $\mathcal{C}$: Objekte sind die Funktoren $F : \mathcal{C}^{op} \to \mathbf{Set}$, und Morphismen sind natürliche Transformationen der Funktoren.

Bemerkung. Allgemeiner kann man auch die Kategorie $[\mathcal{C}, \mathcal{D}]$ einführen, deren Objekte Funktoren $F : \mathcal{C} \to \mathcal{D}$ sind und deren Morphismen ebenfalls natürliche Transformationen der Funktoren sind.

2.3 Simpliziale Mengen

Simpliziale Mengen,...

Definition 2.18 (Simplexkategorie)**.** Die Objekte der *Simplexkategorie* Δ sind die Mengen $[n] \coloneqq \{0, 1, \dots, n\}$ für $n \in \mathbb{N}$, und Morphismen sind monoton wachsende Abbildungen.

Wir können die n-elementigen Mengen in Δ als geometrische n-Simplizes auffassen. Dazu betrachten wir den Funktor $|\cdot| : \Delta \to \mathbf{Top}$, gegeben durch

$$|\cdot| : [n] \mapsto |\Delta^n| \coloneqq \left\{ (t_0, \dots, t_n) \in \mathbb{R}^{n+1} \middle| \sum_{i=0}^{n} t_i = 1, t_i \geq 0 \right\}$$

Definition 2.19. Eine *simpliziale Menge* ist ein Funktor $X : \Delta^{op} \to \mathbf{Set}$. Üblicherweise wird $X([n])$ als X_n geschrieben, und wir bezeichnen die Elemente $x \in X_n$ als n-Simplizes. Der n-dimensionale *Standardsimplex* ist

$$\Delta^n \coloneqq \mathsf{Hom}(-, [n]).$$

Der Standardsimplex ist somit ein Hom-Funktor (siehe **??**).

Kapitel 3

UMAP

In diesem Kapitel werden wir die aus **??** erlangten Grundlagen verwenden um eine geeignete topologische Repräsentation unserer Daten $X = \{x_i\}_{i=1}^N,\ (x_i \in \mathbb{R}^D)$ zu erlangen. Eine ähnliche Repräsentation werden wir von einem d-dimensionalen Raum ($d \ll D$) betrachten, um dann mit einem geeigneten Begriff des Abstandes der beiden Repräsentationen die niedrigdimensionale Repräsentation der hochdimensionalen anzupassen. Dies wird uns zum UMAP Verfahren führen.

3.1 Topologische Repräsentation

Blabla bla

Fluch der Dimensionen

Ein kritischer Punkt beim analysieren hochdimensionaler Daten ist der sogenannte Fluch der Dimensionen. Was das bedeutet und wie das UMAP Verfahren macht um diesen zu vermeiden werde ich im Folgenden kurz erklären. Unter dem Fluch der Dimensionen versteht man.

Abbildung **??** kann man zwei interessante Aussagen über den Fluch der Dimensionen entnehmen.

1. Die paarweisen euklidischen Distanzen steigen mit zunehmender Größe der Dimensionen.

2. Die paarweisen Distanzen sind ungefähr normalverteilt, wobei die Varianz der Normalverteilung für höhere Dimensionen abnimmt. Dadurch sind die Distanzen eines Punktes zu seinen ersten, zweiten, ... k-ten Nachbarn im hochdimensionalen Raum annähernd gleich.

Um dieses Phänomen zu vermeiden nutzt das UMAP Verfahren eine Normalisierung der Distanzen mit dem ersten Nachbarn. So werden die relativen Distanzen zwischen den k-nächsten-Nachbarn größer, siehe

Um die niedrigdimensionale Repräsentation der hochdimensionalen anzupassen wird für das UMAP Verfahren die Kreuzentropie zwischen zwei unsicheren (simplizialen) Mengen vorgeschlagen.

$$C((A,\mu),(A,\nu)) \coloneqq \sum_{a\in A}\left(\mu(a)\log\left(\frac{\mu(a)}{\nu(a)}\right) + (1-\mu(a))\log\left(\frac{1-\mu(a)}{1-\nu(a)}\right)\right) \tag{3.1}$$

In Abschnitt **??** werden wir die Kreuzentropie genauer betrachten.

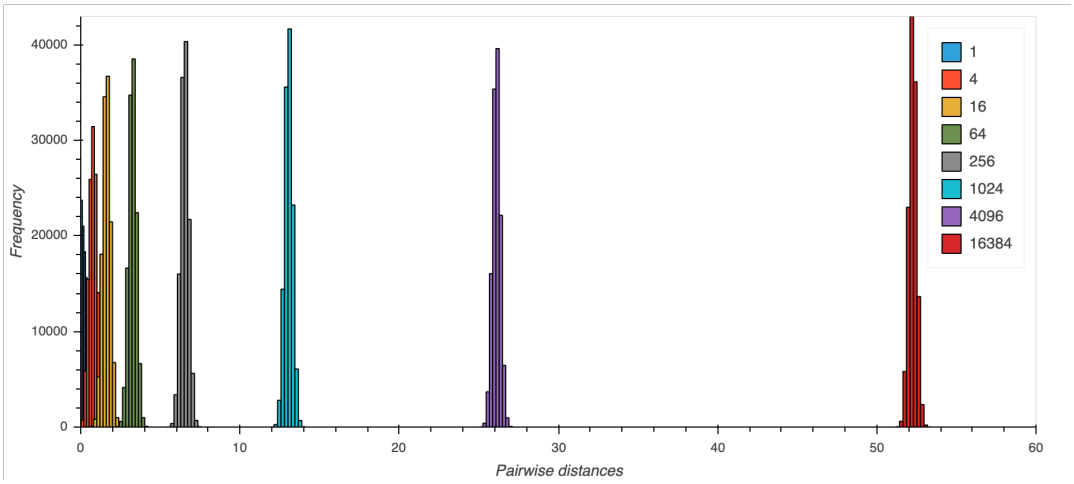

ABBILDUNG 3.1: Paarweise Distanzen von $N = 500$ zufällig gleichverteilten Punkten im R^D.

3.2 Ergänzungen

Kapitel 4

Implementierung

In diesem Kapitel möchten wir die Implementierung des UMAP Verfahren beschreiben. Die vollständige Berechnung aller Simplizes hat eine exponentielle Laufzeit, da hierfür alle 2^N Teilmengen unseres N-elementigen Datensatzes betrachtet werden müssten. In aktuellen Implementierungen werden hingegen nur alle zweielementigen Teilmengen betrachtet. Wie wir in Kapitel **??** sehen werden, liefert uns diese Approximation des Čech-Komplexes sehr gute visuelle Ergebnisse. In Kapitel **??** werden wir Ansätze erwähnen um Simplizes höherer Ordnungen effizient zu finden.

Zunächst werden wir die Notation aus Kapitel **??** anpassen, da wir nur das 1-Skelett der topologischen Repräsentationen betrachten. Danach werden wir die Zielfunktion explizit angeben und den Gradienten herleiten. In Abschnitt **??** werden wir den UMAP Algorithmus im Pseudo-Code angeben. Die darauf folgenden Abschnitte identifizieren rechenintensive Schritte der von uns verwendeten Implementierung [**cpu**] und beschreiben Alternativen. Abschnitt **??** gibt eine Beschreibung der Hyperparameter an.

4.1 Numerische Formulierung der Optimierung

Um eine Unterscheidung zwischen der theoretischen Sichtweise auf das UMAP Verfahren, welche alle k-Simplizes ($1 \leq k \leq N$) berücksichtigt, und der praktischen Implementierung zu verdeutlichen, werden wir die verwendete Notation anpassen.

Die unsicheren Mengen $(A, \mu), (A, \nu)$ aus Gleichung (**??**) lassen sich als gewichtete Graphen, in Form einer Adjazenzmatrix darstellen. Das 1-Skelett der hochdimensionalen Repräsentation notieren wir als Adjazenzmatrix V mit $v_{ij} = \mu(a)$, für ein 1-Simplex a, mit Facetten i und j.

Die Wahl des Zugehörigkeitsgrades für 1-Simplizes der niedrigdimensionalen Repräsentation der Daten ist wie folgt gegeben:

$$w_{ij} = (1 + a(\|\mathbf{y}_i - \mathbf{y}_j\|_2^2)^b)^{-1}, \tag{4.1}$$

wobei $\mathbf{y}_i, 1 \leq i \leq N$ die Vektoren der Einbettung sind. Die Wahl der Werte a, b werden wir in Abschnitt **??** beschreiben.

Somit erhalten wir folgende Kreuzentropie zwischen den Graphen der hoch- und niedrigdimensionalen Darstellung der Daten:

$$C(V,W) = \sum_{i,j=1}^{N} v_{ij} \log\left(\frac{v_{ij}}{w_{ij}}\right) + (1-v_{ij}) \log\left(\frac{1-v_{ij}}{1-w_{ij}}\right) \tag{4.2}$$

$$= \sum_{i,j=1}^{N} (v_{ij} \log(v_{ij}) + (1-v_{ij}) \log(1-v_{ij})) \tag{4.3}$$

$$- \sum_{i,j=1}^{N} (v_{ij} \log(w_{ij})) - \sum_{i,j=1}^{N} ((1-v_{ij}) \log(1-w_{ij}))$$

$$= C_v - \sum_{i,j=1}^{N} (v_{ij} \log(w_{ij}) + (1-v_{ij}) \log(1-w_{ij})) \tag{4.4}$$

Der Term C_v bleibt während der Minimierung der Funktion bezüglich der $\mathbf{y}_i$ konstant.

Aufgrund der Wahl einer differenzierbaren Funktion für den Zugehörigkeitsgrad der 1-Simplizes, bzw. der Gewichte des niedrigdimensionalen Graphen, lässt sich nun der Gradient der Zielfunktion (**??**) herleiten.

Mittels der Kettenregel ergibt sich mit $d_{ij} \coloneqq \|\mathbf{y}_i - \mathbf{y}_j\|_2$:

$$\frac{\partial C}{\partial \mathbf{y}_i} = \sum_{k,l=1}^{N} \frac{\partial C}{\partial w_{kl}} \sum_{m,n=1}^{N} \frac{\partial w_{kl}}{\partial d_{mn}^2} \sum_{p,q=1}^{N} \frac{\partial d_{mn}^2}{\partial d_{pq}} \frac{\partial d_{pq}}{\partial \mathbf{y}_i} \tag{4.5}$$

$$= \sum_{k,l=1}^{N} \frac{\partial C}{\partial w_{kl}} \sum_{m,n=1}^{N} \frac{\partial w_{kl}}{\partial d_{mn}^2} \frac{\partial d_{mn}^2}{\partial d_{mn}} \frac{\partial d_{mn}}{\partial \mathbf{y}_i} \tag{4.6}$$

$$= 2 \sum_{k,l=1}^{N} \frac{\partial C}{\partial w_{kl}} \sum_{m=1}^{N} \frac{\partial w_{kl}}{\partial d_{mi}^2} \frac{\partial d_{mi}^2}{\partial d_{mi}} \frac{\partial d_{mi}}{\partial \mathbf{y}_i} \tag{4.7}$$

$$= 2 \sum_{m=1}^{N} \left(\sum_{k,l=1}^{N} \frac{\partial C}{\partial w_{kl}} \frac{\partial w_{kl}}{\partial d_{mi}^2} \right) \frac{\partial d_{mi}^2}{\partial d_{mi}} \frac{\partial d_{mi}}{\partial \mathbf{y}_i} \tag{4.8}$$

$$= 4 \sum_{m=1}^{N} \left(\sum_{k,l=1}^{N} \frac{\partial C}{\partial w_{kl}} \frac{\partial w_{kl}}{\partial d_{mi}^2} \right) (\mathbf{y}_i - \mathbf{y}_m) \tag{4.9}$$

Bei der Umformung von (**??**) nach (**??**) haben wir verwendet, dass d_{mi} die euklidische Norm ist. Für Einbettungen in andere metrische Räume müsste der Gradient an dieser Stelle entsprechend angepasst werden. Da wir das UMAP Verfahren im wesentlichen zur Visualisierung im $\mathbb{R}^2$ benutzen ist die Wahl der euklidischen Norm gerechtfertigt. Die übrigen Umformungen ergeben sich aus umordnen und wegfallen der Terme.

(**??**) lässt sich weiter umformen, dazu nutzen wir:

$$\frac{\partial C}{\partial w_{ij}} = -\frac{v_{ij}}{w_{ij}} + \frac{1-v_{ij}}{1-w_{ij}} = \frac{w_{ij} - v_{ij}}{w_{ij}(1-w_{ij})} \tag{4.10}$$

$$\frac{\partial w_{ij}}{\partial d_{ij}^2} = -bad_{ij}^{2(b-1)} w_{ij}^2 = -\frac{b}{d_{ij}^2} w_{ij}(1-w_{ij}) \tag{4.11}$$

Der UMAP Gradient ist also gegeben durch:

$$\frac{\partial C}{\partial \mathbf{y}_i} = 4 \sum_{j=1}^{N} (w_{ij} - v_{ij}) \frac{b}{d_{ij}^2} (\mathbf{y}_i - \mathbf{y}_j) \tag{4.12}$$

4.2 Pseudo-Code

Nun können wir das UMAP Verfahren für die 1-Skelette der topologischen Repräsentation angeben.

Algorithm 1 UMAP Algorithmus

1: **function** UMAP($X, N, D, d, min\_dist, n\_epochs$)
2: **for** $x \in X$ **do**
3: $knn(x) \leftarrow k$-nächste-Nachbarn(x)
4: $graph(x) \leftarrow \perp_{y \in knn(x)}(\{x, y\}, exp(-d_{x,y})) \perp \perp_{y \in X \setminus \{x\}}(\{x, y\}, 0)$
5: $V \leftarrow$ gewichtete Adjazenzmatrix($\bigcup_{x \in X} graph(x)$)
6: $D \leftarrow$ Grad-Matrix des Graphen V
7: $L \leftarrow D^{1/2}(D - V)D^{1/2}$ → Symmetrische normalisierte Laplace-Matrix
8: $evec \leftarrow$ sortierte Eigenvektoren von L
9: $Y \leftarrow evec[1, \ldots, d{+}1]$
10: $Y \leftarrow$ OptimiereEinbettung(Y, min_dist, n_epochs) → siehe Algorithmus **??**
11: **return** Y

In Abschnitt **??** werden zwei effiziente Verfahren für die k-nächste-Nachbarn Suche (Zeile **??**) angeben.

Der in Zeile **??** verwendete $\perp$ Operator verdeutlicht, dass wir die Kantengewichte mittels wahrscheinlichkeitstheoretischer t-Conorm vereinigen. In den von uns verwendeten Implementierung des UMAP Verfahrens [**cpu**, **GPUUMAP**] wird ein zusätzlicher Hyperparameter verwendet, welcher eine Verallgemeinerung der Vereinigung darstellt (siehe: `set_op_mix_ratio` in Abschnitt **??**). Zusätzlich ist zu beachten, dass die Vereinigung in Zeile **??** ungerichtete Kanten betrachtet. Der so erhaltene ungerichtete Graph besitzt aufgrund der Symmetrie der t-Conorm wohldefinierte Kantengewichte.

Die Grad-Matrix D (Zeile **??**) ist eine Diagonalmatrix, wobei $d_{ii} \coloneqq \sum_{1 \leq k \leq N} v_{ik}, (1 \leq i \leq N)$. Für den Spezialfall, der ungewichteten Adjazenzmatrix, beschreibt d_{ii} also den Grad des Knoten i.

Die Initialisierung der niedrigdimensionalen Repräsentation Y in Zeile **??** ist die spektrale Einbettung des Graphen. Eine genauere Beschreibung warum die so gewählte Initialisierung sinnvoll ist findet sich in Abschnitt **??**. Dabei ist zu beachten, dass eine Lösung des Eigenwertproblems nur von der Größe der Einbettungsdimension abhängig ist. Da wird stets $d \ll D$ betrachten ist eine effiziente Implementierung mittels sukzessiver Eigenwertsuche möglich.

4.3 Spektrale Einbettung

Der Laplace Operator ist eine diskrete Approximation des Laplace-Beltrami Operators. Spektrale Einbettung ...

D	Laufzeit NN-Descent	Laufzeit der Optimierung
100	9%	75,3%
500	12%	73,8%
1000	14%	72,9%
5000	30,4%	58%
10000	44%	45,1%
50000	78,8%	14,8%

TABELLE 4.1: D beschreibt die Größe der Umgebungsdimension. Abhängig von D haben wir die Laufzeit des UMAP Verfahrens profiliert. Die zweite und dritte Spalte beziehen sich auf die relativen Laufzeiten des kNN Verfahrens und der Optimierung der Einbettung.

4.4 Profiling

In Kapitel **??** werden wir genauer auf die tatsächliche Laufzeit des UMAP Algorithmus eingehen. An dieser Stelle möchten wir die rechenintensiven Subroutinen des Verfahrens ausmachen. Dafür haben wir den Python cProfiler verwendet, dieser misst die Laufzeit der aufgerufenen Funktionen. Um für verschiedene Umgebungsdimensionen vergleichbare Ergebnisse zu erhalten, haben wir $N = 10\,000$ Datenpunkte in 10 unterschiedlichen $D = [100, 500, 1000, 5000, 10000, 50000]$-dimensionalen Gaußverteilten Datenwolken gewählt. Diese Daten wurden dann in den zweidimensionalen Raum eingebettet.

Dabei ist uns aufgefallen, dass besonders der k-nächste-Nachbarn-Algorithmus und die Optimierung mittels stochastischem Gradienten Verfahren einen großen Teil der Laufzeit des Verfahrens beanspruchen. In Tabelle **??** sind die Ergebnisse der Profilierung zusammengefasst. Insbesondere scheint die k-nächste-Nachbarn Suche die Laufzeit des UMAP Verfahren für hochdimensionale Daten stark zu beeinflussen. Wir werden beide rechenintensiven Subroutinen im folgenden betrachten und geeignete Verbesserungen diskutieren.

4.5 Gradientenverfahren

Um die Zielfunktion (**??**) zu optimieren bietet sich die Wahl eines Gradientenverfahrens an, da eine differenzierbare Approximation des Zugehörigkeitsgrades (siehe Gleichung **??**) gegeben ist.

In den vergangenen Jahren gab es viele Weiterentwicklungen, insbesondere bezüglich der Konvergenzgeschwindigkeit, von Gradientenverfahren. Diese werden unter anderem für das trainieren neuronaler Netzwerke bei der Backpropgation genutzt. In [**SGDGPU**] werden verschiedene Implementierungen verglichen.

Um den Rechenaufwand im Gradientenverfahren zu verringern, wird der Gradient in zwei Summanden aufgeteilt. Dabei wird folgende Beobachtung genutzt: Für Kanten $\{i, j\}$ mit einem hohen Zugehörigkeitsgrad ($v_{ij} \approx 1$) ist der Term $(1 - v_{ij}) \log(1 - w_{ij})$ aus Gleichung (**??**) nahe Null, deshalb ist es sinnvoll nur den Gradienten des Terms $v_{ij} \log(w_{ij})$ zu betrachten. Für Kanten mit $v_{ij} \approx 0$) sollte hingegen der Gradient des Terms $(1 - v_{ij}) \log(1 - w_{ij})$ betrachtet werden. Für das UMAP Verfahren wird deswegen folgende Implementierung vorgeschlagen:

Die in Zeile **??** beschriebene Ziehung der Stichprobe dient dazu in Zeile **??** keine zusätzliche Multiplikation mit v_{ij} zu machen. Diese Idee entstammt [**LINE**]. Dadurch wird der Gradient nicht durch den Wert von v_{ij} beeinflusst.

Algorithm 2 Optimiere die Einbettung mittels modifiziertem SGD

```
1: function OptimiereEinbettung(Y, V, W, n_epochs)
2:     α ← 1.0
3:     for e ← 1, ..., n_epochs do
4:         for all v_ij do
5:             if Random() ≤ v_ij then
6:                 y_i ← y_i + α · ∇(log(w_ij))
7:                 for l ← 1, ..., n_neg_samples do
8:                     m ← Unif((0, N))
9:                     y_i ← y_i + αγ · ∇(log(1 − w_im))
10:    α ← 1.0 − e/n_epochs
11:    return Y
```

Das in jedem Durchlauf des modifizierten SGD mehrere *negative samples* betrachtet werden geht auch [**Mikolov**] zurück. Im wesentlichen verhindert dies, das sich die Vektoren der niedrigdimensionalen Darstellung häufen. Die dabei $(0, N)$-gleichverteilt gezogene Stichprobe ist eine Modifizierung von [**Tang**].

Der Gradient in Zeile **??** ist gegeben durch:

$$\nabla(\log(w_{ij})) = \frac{-2ab\|\mathbf{y}_i - \mathbf{y}_j\|_2^{2(b-1)}}{(1 + a(\|\mathbf{y}_i - \mathbf{y}_j\|_2^2)^b)^{-1}}(\mathbf{y}_i - \mathbf{y}_j) \tag{4.13}$$

und der Gradient in Zeile **??** durch:

$$\nabla(1 - \log(w_{ij})) = \frac{2b}{(\epsilon + \|\mathbf{y}_i - \mathbf{y}_j\|_2^2)(1 + a\|\mathbf{y}_i - \mathbf{y}_j\|_2^{2b})}(\mathbf{y}_i - \mathbf{y}_j), \tag{4.14}$$

wobei der ϵ-Parameter eine Division mit Null vermeidet.

4.6 Nächste-Nachbarn-Klassifikation

Zum effizienten finden der 1-Simplizes der topologischen Repräsentation unserer Daten, benötigen wir einen k-nächste-Nachbarn-Algorithmus (kurz: *kNN-Algorithmus*).

Das Ergebnis eines kNN-Algorithmus wird meist in einem ungerichteten Graph – dem kNN-Graph – dargestellt, wobei die Knoten den Datenpunkten entsprechen und die Kanten den Nachbarschaftsbeziehungen, somit besitzt jeder Knoten Grad k.

Bei einer naiven Implementierung beträgt die Laufzeit $\mathcal{O}(N^2 D)$ (wobei N die Anzahl der Datenpunkte und D die Dimension der Datenpunkte ist). Mit einer effizienten Implementierung ist in der Praxis eine annähernd in N lineare Laufzeit möglich. Die Herangehensweisen lassen sich nach [**Tang**] in drei Kategorien einteilen. (1) Baum basierte Verfahren auf Partitionen des Raumes, (2) Hashfunktionen auf lokalen Teilgebieten des Raumes (3) Nachbarschafts-Erkundungen.

Wir möchten nun zwei Verfahren vorstellen.

NN-Descent

Der NN-Descent Algorithmus [**k-NNG**] beruht auf dem Prinzip der Nachbarschafts-Erkundungen. Dabei wird ein initialer kNN-Graph iterativ verbessert, unter der Annahme, dass die Nachbarschaftsbeziehung transitiv ist, für zwei vorhandene Nachbarschaftspaare $(x, y), (y, z)$ also mit hoher Wahrscheinlichkeit auch ein Nachbarschaftspaar (x, z) im kNN-Graph existiert. Der initiale Graph im NN-Descent Verfahren wird dabei zufällig gewählt. Dies kann dazu führen, dass nur lokal optimale k-NN-Graphen gefunden werden. Dies könnte laut [**EFANNA**] dadurch verbessert werden, indem für die Initialisierung „random projection trees“, wie in [**Tang**], verwendet werden.

Ein Vorteil des NN-Descent Verfahren ist, dass kein globaler Index der verwaltet werden muss. Somit ist eine Anwendung auf großen Datensätzen möglich welche nicht komplett in den Arbeitsspeicher (RAM) des verwendeten Rechners geladen werden können.

Nachteil des NN-Descent Algorithmus ist die Speicherplatzkomplexität, diese ist durch $\mathcal{O}(N^2)$ beschränkt. Im wesentlichen ist dies dadurch begründet, dass paarweise die Ähnlichkeit, welche im Falle von UMAP durch die Metrik des Umgebungsraums gegeben ist, gespeichert wird. Aufgrund dessen, dass nur lokale Optima garantiert sind, ist das Ergebnis des NN-Descent Verfahren approximativ. In [**UMAP**] wird jedoch argumentiert, dass dies wegen des Informationsverlust bei Dimensionsreduktionen kaum Auswirkungen auf die resultierende Einbettung hat.

FAISS

Das FAISS Bibliothek [**FAISS**] nutzt die Architektur einer GPU aus. Dabei baut FAISS eine effiziente Datenstruktur, welche für die Vektoren die nächsten Nachbarn speichert. Somit ist eine sehr schnelle Implementierung für das aufstellen des k-NN-Graphen möglich.

Der RAM der meisten GPUs ist stark begrenzt. Um dennoch mit großen Datensätzen zu arbeiten werden komprimierte Darstellungen der Vektoren genutzt. Für FAISS werden „product quantization codes“ genutzt.

Vorteile der FAISS Datenstruktur sind die effiziente Implementierung auf GPUs und das sowohl exakte Ergebnisse sowie Approximationen für die nächsten Nachbarn angegeben werden können. Die Rückgabe approximativer Ergebnisse erhält Laufzeit sowie Speicherplatz Vorteile.

Nachteil des FAISS Verfahren ist, dass zurzeit nur die euklidische Distanz unterstützt wird.

4.7 Hyperparameter

- `n_neighbors`: beschreibender Text
- `metric` Text
- `n_epochs` Text
- `set_op_mix_ratio`
- `local_connectivity`
- `repulsion_strength`
- `a, b`: Erwähne wie a und b standardmäßig gewählt werden, zeige Plot von a, b Erwähne `min_dist` und `spread`

- `negative_sample_weight`

Kapitel 5

Experimente

Nach ausführlicher Darstellung der Theorie des UMAP Verfahrens, möchten wir nun UMAP auf drei Datensätzen mit alternativen Verfahren empirisch testen. Wir werden eine möglichst vollständige Darstellung der Ergebnisse in dieser Arbeit präsentieren. Allerdings ist es zu empfehlen die Ergebnisse in einem interaktiven Jupyter notebook zu betrachten. Dieses befindet sich auf der beigelegten CD oder auf GitHub [1].

5.1 Alternative Verfahren

Wir haben uns dazu entschieden UMAP mit folgenden Verfahren zu vergleichen:

Da die Implementierungen des Laplacian Eigenmaps Verfahrens und Isomap Verfahrens schlecht für große Datensätze skalieren, haben wir uns bewusst dazu entschieden, diese nicht mit in die Analyse aufzunehmen.

5.1.1 t-SNE

Das t-SNE Verfahren [**tSNE**] ist zur Zeit eines der bekanntesten und meistgenutzten nicht-linearen Dimensionsreduktionsverfahren. Dabei wird UMAP fast ausschließlich zur Visualisierung genutzt, da die Laufzeit für höhere Einbettungsdimensionen schlecht ist.

t-SNE konstruiert zuerst eine Wahrscheinlichkeitsverteilung P auf Paaren (i, j) der hochdimensionalen Datenpunkten. Diese ist so gewählt, dass Paare ähnlicher Objekte eine höhere Wahrscheinlichkeit zugeordnet bekommen, wohingegen sehr unterschiedliche Datenpunkte eine Wahrscheinlichkeit nahe 0 haben. Die Ähnlichkeit der Punkte wird dabei meist mittels der euklidischen Distanz gemessen, kann aber ähnlich wie im UMAP Algorithmus durch andere Metriken ersetzt werden. Um P zu konstruieren wird eine Gaußverteilung genutzt, wobei die Varianz abhängig vom `perplexity` Parameter ist. Die so erhaltenen Wahrscheinlichkeiten $p_{i|j}$ sind im Allgemeinen nicht symmetrisch. Die Symmetrie wird durch mitteln der Daten erhalten.

Ähnlich wird eine Wahrscheinlichkeitsverteilung Q im niedrigdimensionalen Raum mithilfe der studentschen t-Verteilung konstruiert. Ursprünglich wurde Q ebenfalls durch eine Gausverteilung konstruiert, das so erhaltene Verfahren (SNE [**tSNEvar1**]) ist allerdings aufgrund einer schwierig zu optimierenden Zielfunktion und dem „crowding problem“ wenig praktikabel.

Um die d-dimensionale Repräsentation der Daten zu optimieren wir die Kullback-Leiber Divergenz von zwischen P und Q bezüglich der y_i minimiert.

Seit der Veröffentlichung des Verfahrens wurden zahlreiche Verbesserungen, insbesondere für die Laufzeit, vorgeschlagen [**tSNEmod1**, **tSNEmod3**]. Dabei ist besonders Barnes-Hut-SNE [**tSNEmod2**] zu erwähnen, allerdings sollte hier beachtet

[1] https://github.com/reinerschristopher/bachelorthesis

werden, dass aufgrund der Konstruktion einer speziellen Datenstruktur die Laufzeit für $d > 3$ sehr schlecht ist.

Die von t-SNE produzierte Repräsentation der Daten ist vom `perplexity` Parameter abhängig. Dabei kann man festhalten, je größer die `perplexity` ist, desto größer ist die Varianz des Gaußverteilung. Somit werden für große `perplexity` Werte globalere Strukturen erfasst, da der Gaußkern sehr breit ist. Wenn der `perplexity` Parameter in der Größenordnung der Anzahl an Datenpunkten N liegt gleicht t-SNE dem MDS Verfahren.

Der zweite wichtige Hyperparameter, welchen wir beschreiben möchten ist die `exaggeration`. meistens wird hier zwischen `early-` und `late-exaggeration` unterschieden. Im wesentlichen verbessert der Parameter die Optimierung des Gradienten und sorgt dafür, dass Punkte desselben Clusters möglichst schnell in der niedrigdimensionalen Repräsentation gruppiert werden [**tSNEcluster**]. Der `late-exaggeration` wie in [**tSNEmod3**] beschrieben kontrahiert gefundene Cluster, so lassen sich in einer 2- oder 3-dimensionalen Darstellung leichter Cluster bestimmen - entweder visuell oder mittels Clustering-Verfahren.

Wir werden reale Datensätze analysieren und das Verhalten der Hyperparameter beschreiben, um ein zusätzliches Verständnis für die von t-SNE genutzten Hyperparameter zu bekommen empfiehlt sich [**tSNEparam**] - zeigt interaktiv auf künstlich erzeugten Datensätzen die Auswirkung der Parameter.

Für unsere Experimente haben wir die scikit Implementierung des t-SNE Verfahren genutzt [**scikit-learn**]. Zusätzlich haben wir die openTSNE [**opentSNE**] Implementierung genutzt. Diese beschleunigt die Laufzeit des t-SNE Algorithmus durch eine zusätzliche Fouriertransformation [**tSNEmod3**]. Die openTSNE Implementierung besitzt im Vergleich zur scikit Implementierung die Möglichkeit den `late-exaggeration` Parameter zu spezifizieren.

5.1.2 TriMap

Das TriMap Verfahren [**TriMap**] soll eine globalere Repräsentation der Daten finden als beispielsweise t-SNE, das nicht nur paarweise die Ähnlichkeit zweier Objekte i, j betrachtet wird, sondern stets Triple i, j, k. Die so erhaltene globale Struktur der Daten soll die Cluster-Abstände der Daten repräsentieren.

Wir haben diesem Algorithmus gewühlt, da die Tripletts Ähnlichkeiten mit 2-Simplizes des UMAP Verfahren haben. Die Tripletts sind vergleichbar mit den 2-Simplizes des UMAP Verfahrens. Insbesondere können die Ansätze eine lineare Teilmenge ($O(N)$) an Tripletts zu finden weitere Entwicklungen des UMAP Verfahren motivieren.

5.2 Bewertung der Ergebnisse

Um die d-dimensionalen Repräsentationen der Daten zu bewerten gibt es verschiedene Ansätze. Diese sollen nun vorgestellt und verglichen werden.

Ein DR-Verfahren, welches die statistischen Eigenschaften der zugrundeliegenden Daten erfasst, sollte invariant bezüglich Rauschen der Daten sein. Hingegen sollte eine schlechte Einbettung wenig stabil sein, wenn zusätzliches Rauschen in den Daten auftritt.

Eine weitere Methodik um die Qualität der Einbettung zu erfassen, erhält man dadurch, dass

In [**Harmeling**] werden zwei Methoden zur Auswahl eines geeigneten DR-Verfahrens verglichen.

Abbildung 5.1: Sechs zufällig gewählte Gesichter des Cartoon Set.

Um die lokale Qualität der Algorithmen zu analysieren haben wir uns die *Cluster* in der 2-dimensionalen Repräsentation angeschaut, wobei wir als Cluster eine Teilmenge der Daten bezeichnen, welche deutlich von den anderen Datenpunkten getrennt ist. Insbesondere bei der Analyse des Cartoon Set **??** konnten wir gut lokale Strukturen erkennen, da jeder Datenpunkt mehrere Eigenschaften gegeben hat – im Vergleich zum MNIST und FMNIST Datensatz, wo uns nur ein *Label* pro Datenpunkt gegeben ist.

Die globale Struktur der Repräsentation qualitativ zu bewerten ist subjektiv. Dabei ist insbesondere die Frage – „Wie *stark* unterscheiden sich die Cluster?“ – zu beantworten. Beispielsweise sollten die Cluster welche die Ziffern 1 und 7 darstellen näher zueinander liegen als die Cluster der Ziffern 1 und 6.

Für die qualitative Analyse wird die Fähigkeit des Gehirns genutzt Strukturen zu erkennen. Nachteile der qualitativen Analyse sind, (1) die Subjektivität und somit Abhängigkeit vom Betrachter, (2) dass sie nur im d-dimensionalen ($d \leq 3$) möglich ist.

In den folgenden Experimenten haben wir uns auf eine qualitative Analyse der Daten beschränkt. Einerseits ist dies unserer Erfahrung nach (und Einträgen in Internetforen, ...) die meistgenutzte Art die Repräsentation in der Praxis zu bewerten. Zusätzlich bietet die Wahl der Datensätze eine gute Gelegenheit die Repräsentationen visuell zu bewerten.

5.3 Cartoon Set

In diesem Abschnitt werden wir den *Cartoon Set* Datensatz analysieren [**cartoon**]. Dabei werden wir:

- sehen, dass UMAP eine vergleichbare Laufzeit mit der von FIT-SNE hat
- das Verhalten der niedrigdimensionalen Darstellung unter verschiedenen Hyperparametern betrachten
- eine exemplarische Beschreibung der Hyperparameter geben
- UMAP mit anderen Dimensionsreduktionsverfahren vergleichen
- sehen, dass UMAP zugrundeliegende Mannigfaltigkeiten erkennt und darstellt

Beschreibung des Datensatzes

Der Cartoon Datensatz enthält 100 000 unterschiedliche Bilder von gezeichneten Gesichtern (siehe Abbildung **??**).

Die Bilder wurden aus 16 Komponenten zusammengesetzt (u.a. Gesichtsform, Gesichtsfarbe, Frisur, Haarfarbe), dabei variiert die Anzahl der Möglichkeiten pro Komponente zwischen zwei (Augenlid, Wimpern,...) und 111 (Anzahl mögliche Frisuren).

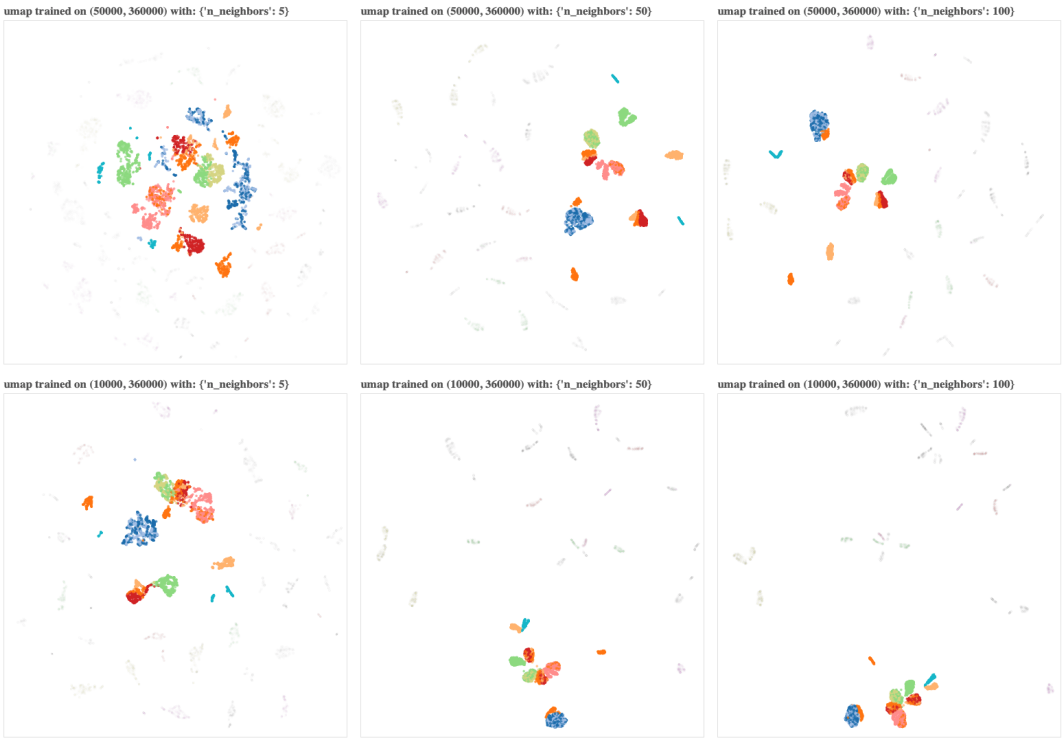

Abbildung 5.2: TODO: Beschreibung des Bildes

Die Farben der Komponenten wurden aus einem diskreten RGB Raum gewählt. Insgesamt ergibt sich eine mögliche Anzahl von 10^{13} Gesichtern.

Für die Analyse haben wir verschiedene Eigenschaften zusammengefasst um einen besseren Überblick zu haben. Beispielsweise haben wir die 111 Frisuren, nach qualitativer Analyse, zu 19 Frisurformen zusammengefasst.

Wir haben uns für diesen Datensatz entschieden um UMAP auf Daten mit einer komplexeren Struktur zu testen als dies in [**UMAP**] gemacht wird. Dabei ist auch zu beachten, dass es aufgrund der 16 Komponenten aus welchen die Gesichter bestehen kein richtige oder falsche Einbettung der Daten gibt.

Wir haben die Bewertung der Einbettung unter der Annahme gemacht, dass *ähnliche* Gesichter *ähnliche* Hautfarben, Frisuren, Haarfarben, Brillen und Bärte haben. Diese fünf Eigenschaften möchten wir besonders hervorheben, da sie die dominantesten Merkmale des Gesichts beschreiben.

Die ursprüngliche Größe eines Bildes betrug 500×500 Pixel mit vier Farbkanälen (CYMK-Darstellung der Farben). Aufgrund des großen Randes haben wir uns dazu entschieden die Größe der Bilder auf 300×300 ohne nennenswerten Informationsverlust zu verringern. Weiterhin haben wir uns für die Bewertung der Einbettung auf 10 000 Bilder beschränkt. Somit beträgt die Dimension des Cartoon Set $D = 360\,000$ und die Anzahl an Beispielen $N = 100\,000$.

Qualitative Analyse der Ergebnisse

5.4 MNIST

FMNIST

5.5 Laufzeitanalyse

Die praktischen Tests der Verfahren wurden auf Rechnern mit einer Linux-Architektur ausgeführt. Die CPU Tests haben wir auf Intel Xeon 6136 CPUs mit 48 Kernen und 384 GB RAM ausgeführt. Für die Verfahren welche mittels Berechnungen auf einer Graphikkarte verbessert wurden, haben wir Intel Xeon Gold 6136 CPUs mit 188 GB RAM und Nvidia V100 GPUs genutzt. Insgesamt haben wir über 100 Experimente gemacht um genauere Aussagen über die Laufzeit der Verfahren zu treffen und diese in Abhängigkeit der wichtigen Parameter zu setzen.

5.6 Stabilität unter sub-sampling

5.7 Zusammenfassung der Ergebnisse

Kapitel 6

Zusammenfassung und Ausblick