National Technical University of Athens
School of Electrical and Computer Engineering
Division of Signals, Control and Robotics

Development of Multidimensional Scaling Algorithms with Application to Natural Language Representation Problems

Manifold Learning technique

Diploma Thesis

of

LENA K.FOTAKI

Supervisor: Alexandros Potamianos
Associate Professor

Athens, October 2023

National Technical University of Athens
School of Electrical and Computer Engineering
Division of Signals, Control and Robotics

Development of Multidimensional Scaling Algorithms with Application to Natural Language Representation Problems

Manifold Learning technique

Diploma Thesis

of

LENA K.FOTAKI

Supervisor: Alexandros Potamianos
Associate Professor

Approved by the examination committee on 10th October 2023.

(Signature)	*(Signature)*	*(Signature)*
Alexandros Potamianos	Konstantinos Tzafestas	Dimitrios Fotakis
Associate Professor	Associate Professor	Associate Professor

Athens, October 2023

National Technical University of Athens
School of Electrical and Computer Engineering
Division of Signals, Control and Robotics

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism.

(Signature)

. . . . . .
LENA K.FOTAKI

NTUA Electrical and Computer Engineering Graduate
10th October 2023

Περίληψη

Στην παρούσα διπλωματική εργασία εξετάζουμε και αναπτύσσουμε αλγορίθμους Συνολοθεωρητικής Πολυδιάστατης Κλιμάκωσης και μείωσης διαστατικότητας. Εστιάζουμε σε αλγορίθμους που μπορούν να απλοποιήσουν δεδομένα με πολύπλοκη και μη γραμμική δομή, έτσι ώστε να γίνουν καλύτερα αντιληπτές από τον άνθρωπο και να αποκαλύψουν πιθανές συσχετίσεις μεταξύ των δεδομένων.

Η συνεισφορά της διπλωματικής αποτελείται από δύο μέρη. Στο πρώτο μέρος της διπλωματικής βελτιστοποιούμε μια επέκταση της κλασικής μεθόδου πολυδιάστατης κλιμάκωσης (Multidimensional Scaling) χρησιμοποιώντας τεχνικές βελτιστοποίησης χωρίς παραγώγους. Η αξιολόγηση του αλγορίθμου μας έγινε σε συνθετικά και πραγματικά δεδομένα και το συμπέρασμα μας είναι ότι ο αλγόριθμος μπορεί να εκτιμήσει με ακρίβεια την γεωμετρία των δεδομένων που είναι ενσωματωμένα σε χώρους με υψηλές διαστάσεις.

Στο δεύτερο μέρος, ορίζουμε το πρόβλημα του Set-MDS όπου η ιδιαιτερότητα είναι ότι τα δεδομένα μας αποτελούνται από σύνολα σημείων και όχι από μεμωνομένα σημεία, και στην συνέχεια προτείνουμε μία προσεγγιστική λύση μέσω του αλγορίθμου COSMOS. Αφού αναλύσουμε τον αλγόριθμο, παρουσιάζουμε τα αποτελέσματα των πειραμάτων μας.

Λέξεις Κλειδιά

Μείωση Διαστατικότητας, Μηχανική Μάθηση, Αλγόριθμοι Πολυδιάστατατης Κλιμάκωσης

Abstract

In this thesis, we explore and develop Manifold Learning and Dimensionality Reduction algorithms. We focus on algorithms capable of simplifying data with complex and non-linear structures to make them more comprehensible for humans and to reveal possible correlations among the data.

The contribution of this thesis consists of two parts. In the first part, we optimize an extension of the classical Multidimensional Scaling (MDS) method using derivative-free optimization techniques. We evaluate our algorithm on synthetic and real-world data, and our conclusion is that the algorithm accurately estimates the geometry of data embedded in high-dimensional spaces.

In the second part, we define the Set-MDS problem, where the peculiarity is that our data consists of sets of points rather than individual points. Subsequently, we propose an approximate solution through the COSMOS algorithm. After analyzing the algorithm, we present the results of our experiments.

Keywords

Dimensionality Reduction, Manifold Learning, Multidimensional Scaling, Set-MDS, Non-linear Algorithms, Knot theory

to my parents

Ευχαριστίες

Πρώτα από όλα θα ήθελα να εκφράσω την ευγνωμοσύνη μου στον επιβλέποντα καθηγητή αυτής της διπλωματικής τον κ. Α. Ποταμιάνο, ο οποίος μου έδωσε την ευκαιρία να ασχοληθώ και να πραγματοποιήσω αυτήν την εργασία που ήταν πρόκληση για εμένα. Κύριε Ποταμιάνε, σας ευχαριστώ.

Ακόμα θα ήθελα να ευχαριστήσω θερμά τον υποψήφιο διδάκτορα, Γιώργο Παρασκευόπουλο για την συνεργασία μας, και κυρίως για την υπομονή, την υποστήριξή και τις συμβουλές του. Ο Γιώργος, πέρα από άριστος μηχανικός, είναι πάντα πρόθυμος να βοηθήσει, να λύσει απορίες και να μοιραστεί τις γνώσεις του.

Επιπλέον, πολύ μεγάλη ευγνωμοσύνη νιώθω για τον Μανώλη Βλατάκη-Γκαραγκούνη, τον οποίο αισθάνομαι μέντορα, συνεργάτη, φίλο. Είναι τιμή μου που συνεργάστηκα μαζί του. Η αγάπη του για την έρευνα αποτέλεσε πηγή έμπνευσης για εμένα και έπαιξε καθοριστικό ρόλο στο να επιμείνω μέχρι τέλους σε αυτήν διπλωματική.

Θέλω να πω ένα μεγάλο ευχαριστώ στους φίλους μου, με τους οποίους πέρασα τις πιο όμορφες στιγμές στα φοιτητικά μου χρόνια. Μαζί τους μοιράστηκα αμέτρητες στιγμές γέλιου, ξεγνοιασιάς αλλά και ατελείωτες ώρες διαβάσματος και σκληρής δουλειάς. Το ταξίδι αυτό, που πλέον φτάνει στο τέλος του, δεν θα ήταν το ίδιο χωρίς αυτούς.

Κλείνοντας την λίστα ευχαριστιών, δεν θα μπορούσα να μην αναφέρω τους γονείς μου, οι οποίοι πάντα είναι δίπλα μου σε οτιδήποτε και αν επιλέξω.

Athens, October 2023

LENA K.FOTAKI

Table of Contents

Περίληψη 1

Abstract 3

Ευχαριστίες 7

Εκτεταμένη Ελληνική Περίληψη 19
- 0.1 Εισαγωγή . . . 19
- 0.2 Landmarks Pattern Seach MDS - (LPS-MDS) . . . 19
  - 0.2.1 Σχετική Βιβλιογραφία . . . 19
  - 0.2.2 Προτεινόμενη Μέθοδος Βελτιστοποίησης Πολυπλοκότητας . . . 20
  - 0.2.3 Αποτελέσματα . . . 22
    - 0.2.3.1 Συνθετικά δεδομένα . . . 22
    - 0.2.3.2 MNIST Σύνολο Δεδομένων . . . 23
- 0.3 Πρόβλημα Set-MDS . . . 25
  - 0.3.1 Προτεινόμενη Μέθοδος - Ο αλγόριθμος COSMOS . . . 25
  - 0.3.2 Αποτελέσματα . . . 26
  - 0.3.3 Συνθετικά δεδομένα . . . 26
  - 0.3.4 Ομοιότητα Λέξεων - Men & Simlex . . . 27

1 Introduction 29
- 1.1 Motivation . . . 29
- 1.2 Thesis Contributions . . . 30
- 1.3 Chapter Outline . . . 31

2 Machine Learning and Dimensionality Reduction 33
- 2.1 Defining Machine Learning and Artificial Intelligence . . . 33
  - 2.1.1 Unraveling Artificial Intelligence (AI) . . . 33
  - 2.1.2 The Role of Machine Learning (ML) . . . 33
- 2.2 Types of Machine Learning . . . 34
  - 2.2.1 Supervised Learning . . . 34
  - 2.2.2 Unsupervised Learning . . . 34
  - 2.2.3 Semi-Supervised Learning . . . 34
  - 2.2.4 Reinforcement Learning . . . 34
- 2.3 Dimensionality Reduction & Manifold learning . . . 35
- 2.4 Manifolds . . . 36

2.5 Dimensionality Reduction Algorithms . . . 37
2.6 Linear Methods . . . 38
2.6.1 SVD . . . 39
2.6.2 PCA . . . 41
2.7 Non Linear Algorithms - Manifold Learning Technique . . . 42
2.7.1 Isometric Feauture Mapping (Isomap) . . . 43
2.7.2 Isomap with landmark points . . . 43
2.7.3 LLE . . . 44

3 Multidimensional Scaling (MDS) 47
3.1 Classical MDS . . . 47
3.2 Metric MDS . . . 48
3.3 Non-metric MDS . . . 48
3.4 Pattern Search MDS . . . 48
3.4.1 General Pattern Search (GPS) methods . . . 49
3.4.2 GPS Convergence . . . 50
3.4.3 The PSMDS algorithm . . . 51
3.4.4 GPS Formulation of PSMDS . . . 54
3.4.5 Updating the Current Dissimilarity Matrix . . . 56
3.4.6 Complexity . . . 56

4 Pattern Search MDS with Landmarks (LPS-MDS) 57
4.1 The LPS-MDS Algorithm . . . 57
4.1.1 Complexity . . . 59
4.2 Experiments with Synthetic Dataset . . . 59
4.2.1 Mnist Dataset . . . 63
4.2.1.1 Visualizing Linear Embeddings . . . 63
4.2.1.2 Image Classification on the Low-Dimensional Space . . . 64
4.2.2 Classification on Microbiome Data Set . . . 65

5 The COSMOS Algorithm 67
5.1 Notation . . . 67
5.1.1 The definition of Set-MDS Problem . . . 67
5.1.2 The violation of triangle inequality in Power Sets . . . 68
5.2 The COSMOS Algorithm . . . 69
5.2.1 Runtime Complexity . . . 71
5.2.2 Experiments . . . 71
5.2.2.1 Synthetic Dataset . . . 71
5.2.3 Semantic Similarity - MEN & SimLex-999 . . . 72
5.2.4 Semantic Similarity - Wordnet . . . 73

6 Conclusions 79
6.1 Future Work . . . 80

Bibliography **85**

List of Figures

1 Απλοποίηση του τρισδιάστατου Swissroll σε δύο διαστάσεις 22
2 Σύγκρισεις ανάμεσα στους αλγορίθμους PSMDS και LPS-MDS. Το LPS-MDS έχει εξαιρετικά αποτελέσματα σε όλες τις περιπτώσεις και για μεγάλο αριθμό σημείων 23
3 Landmark MDS 24

2.1 A one-dimensional manifold embedded in three dimensions. 36
2.2 Earth is an example of a manifold. 36
2.3 The Singular Value Decomposition 39
2.4 PCA minimizes the sum of squared perpendicular distances or maximizes the variance of the projections of the data 41

3.1 Sphere of radius r around point $x_i^{(k)}$ and possible search directions 53

4.1 Original Swissroll in 3D 60
4.2 Visual Swissroll unrolling in 2 dimensions 61
4.3 Original Swissroll in 2D, with Landmarks 62
4.4 Comparison between PSMDS and LPS-MDS. LPS-MDS has remarkable results in all cases 63

5.1 Example of non metric power set space 68
5.2 Example of the structure of WordNet hierarchy. 73
5.3 ChatGPT example usage 74
5.4 The algorithm correctly splits the word "book" and selects positions close to "read" and "hotel", proving both meanings. 75
5.5 The algorithm correctly splits the word "mouse" and selects positions close to "trap" and close to "computer", proving both meanings. 76
5.6 This graph is a combination of the previous words. We ask the algorithm to execute two splits, and it splits correctly the words "mouse" and "book". 77

List of Images

List of Tables

1 Σύγκριση τεχνικών μείωσης διαστατικότητας για το σύνολο δεδομένων MNIST με διάσταση ενσωμάτωσης ίση με 10. Για την ταξινόμηση, χρησιμοποιείται ο αλγόριθμος KNN με διαφορετικό αριθμό πλησιέστερων γειτόνων (παράμετρος K). Η στήλη "Χρόνος" καθορίζει το χρονικό διάστημα που χρειάστηκε κάθε μέθοδος για να υπολογίσει όλα τα διανύσματα ενσωμάτωσης. Το "Αρχικό" αντιστοιχεί στην εκτέλεση του KNN στο αρχικό διανύσμα υψηλής διαστατικότητας χωρίς καμία μείωση διαστατικότητας. 25

2 Ο πίνακας συνοψίζει τα αποτελέσματα πολλαπλών πειραμάτων. Δείχνει το ποσοστό επιτυχημένων διαιρέσεων σε πειράματα με διάφορο αριθμό σημείων μ και διαιρέσεις k 27

3 Σύγκριση τεχνικών μείωσης διαστάσεων σε ομοιότητα λέξεων με χρήση των συνόλων δεδομένων MEN και SimLex-999. 27

4.1 In this table, we display results from experiments using the Swissroll dataset of 1,000 to 20,000 points with varying numbers of landmarks. It is shown that 300 landmarks resulted in less than a 1.5% deviation from the original error, in all cases. 60

4.2 Results from experiment with 10.000 Swissroll. MDS with Landmarks executes in less time, with fewer epochs and less memory. 61

4.3 Comparison of dimensionality reduction techniques for the MNIST dataset with embedding dimension equal to 10. For the classification KNN algorithm is with different number of nearest neighbors (parameter K) is used. “Time" column specifies the amount of time that each method took to compute all embedding vectors. “Initial" corresponds to performing KNN on the original high-dimensional pixel without any dimensionality reduction. . 65

4.4 Comparison of dimensionality reduction techniques for the Micro-biame Data Set with embedding dimension equal to 3. For the classification KNN algorithm with K=9 is used. “Initial" corresponds to performing KNN on the original high-dimensional pixel without any dimensionality reduction. The columns Diet, Source, Donor, CollectionMet, Sex are the different categories of the classification. 66

5.1 The table summarizes the results of multiple experiments. Is shows the percentage of successful splits in experiments with different number of points μ and splits $\mathbf{k}$ 72

5.2 Comparison of dimensionality reduction techniques for the semantic similarity task for MEN and SimLex-999 datasets. 72

Εκτεταμένη Ελληνική Περίληψη

0.1 Εισαγωγή

Σε μια εποχή όπου περιβαλλόμαστε από υπερβολικά πολύπλοκα και πολυδιάστατα δεδομένα, είναι ζωτικής σημασίας να τα απλοποιούμε ώστε να εξάγουμε τις πληροφορίες και να τις χρησιμοποιούμε εύκολα. Στην παρούσα διπλωματική εργασία ασχολούμαστε με τεχνικές που δίνουν λύση σε αυτό το προβλημα και αναπτύσσουμε αλγορίθμους Συνολοθεωρητικής Πολυδιάστατης Κλιμάκωσης και μείωσης διαστατικότητας, οι οποίοι επιτυγχάνουν να απλοποιήσουν δεδομένα με πολύπλοκή δομή.

Η συνεισφορά της διπλωματικής αποτελείται από δύο μέρη. Στο πρώτο μέρος, επιχειρούμε να βελτιώσουμε την πολυπλοκότητα του αλγορίθμου Pattern Search MDS, όπως περιγράφεται στο [1] χρησιμοποιώντας την μέθοδο των Landmark Points όπως περιγράφεται στα [2, 3]. Τα αποτελέσματα μας μειώνουν δραματικά τον χρόνο εκτέλεσης του αλγορίθμου, χωρίς να μειώνουν την ποιότητα του αποτελέσματος. Στο δεύτερο μέρος, ορίζουμε το πρόβλημα του Set-MDS και ασχολούμαστε με την ανάπτυξη του αλγόριθμου COSMOS που δίνει μια ευριστική λύση στο προβλημα αυτό, και που μάλιστα κάνει χρήση του αλγορίθμου Pattern Search MDS που αναλύουμε στο πρώτο μέρος. Εδώ η διαιτερότητα είναι ότι τα δεδομένα μας βρίσκονται σε sets και ο χώρος είναι μη μετρικός, καθώς στις αποστάσεις ανάμεσα σε sets σημείων παραβιάζεται η τριγωνική ανισότητα.

0.2 Landmarks Pattern Seach MDS - (LPS-MDS)

Πιο αναλυτικά, για το πρώτο μέρος της διπλωματικής βελτιστοποιούμε μια επέκταση της κλασικής μεθόδου πολυδιάστατης κλιμάκωσης (Multidimensional Scaling) χρησιμοποιώντας τεχνικές βελτιστοποίησης χωρίς παραγώγους. Η αξιολόγηση του αλγορίθμου μας έγινε σε συνθετικά και πραγματικά δεδομένα και το συμπέρασμα μας είναι ότι ο αλγόριθμος μπορεί να εκτιμήσει με ακρίβεια την γεωμετρία των δεδομένων που είναι ενσωματωμένα σε χώρους με υψηλές διαστάσεις.

0.2.1 Σχετική Βιβλιογραφία

Στο άρθρο [1] περιγράφεται η επέκταση της κλασικής μεθόδου πολυδιάστατης κλιμάκωσης (Multidimensional Scaling) χρησιμοποιώντας τεχνικές βελτιστοποίησης χωρίς παραγώγους. Η μέθοδος που προτείνει η εργασία βασίζεται σε πιθανή κίνηση κάθε σημείου πάνω σε μία σφαίρα σταθερής ακτίνας.

Η είσοδος στον αλγόριθμο είναι ένας $N \times N$ πίνακας ο οποίος περιέχει τις επιθυμητές αποστάσεις ανάμεσα στα σημεία. Στόχος του αλγορίθμου είναι να παράξει τον ίδιο αριθμό

σημείων σε μικρότερες διαστάσεις, πρεσβεύοντας πάντα τις ίδιες αποστάσεις που είχαν μπει στην αρχή του αλγορίθμου. Αυτό που πετυχαίνουμε με άλλα λόγια, είναι στην έξοδο του αλγορίθμου να έχουμε δεδομένα με μικρότερες διαστάσεις, τα οποία διατηρούν τις αρχικές αποστάσεις μεταξύ των σημείων.

Η διαδικασία αρχικοποίησης του αλγορίθμου περιλαμβάνει τα εξής βήματα:

- Δημιουργία N τυχαίων σημείων στον ενσωματωμένο χώρο με τις τελικές -επιθυμητές L διαστάσεις. $X^{(0)} = [x_1^{(0)}, x_2^{(0)}, ..., x_N^{(0)}] \in \mathbb{R}^{N \times L}$
- Υπολογισμός του πίνακα αποστάσεων μεταξύ των σημείων. Το στοιχείο $d_{ij}^{(0)}$ είναι η Ευκλείδεια απόσταση μεταξύ των διανυσμάτων $x_i^{(0)}$ και $x_j^{(0)}$ του $X^{(0)}$
- Υπολογισμός του αρχικού σφάλματος προσέγγισης $e^{(0)} = ||\mathbf{\Delta} - \mathbf{D}||_F^2$, όπου το e αντιπροσωπεύει το σφάλμα μέσου τετραγωνικού σφάλματος (MSE) μεταξύ των δύο πινάκων. Η συνάρτηση στόχου που βελτιστοποιείται για όλα τα PSMDS φαίνεται παρακάτω και είναι η συνάρτηση τάσης για το MDS:

$$f(\mathbf{\Delta}, \mathbf{D}) = ||\mathbf{\Delta} - \mathbf{D}||_F^2 = \sum_{i=1}^{N} \sum_{j=1}^{N} (\delta_{ij} - d_{ij})^2, \quad \text{όπου } \mathbf{\Delta}, \mathbf{D} \in \mathbb{R}^{N \times N} \tag{1}$$

Σε συνέχεια των βημάτων αρχικοποίησης, σε κάθε εποχή, λαμβάνουμε υπόψη την επιφάνεια μιας σφαίρας ακτίνας r γύρω από κάθε σημείο $x_i^{(k)}$. Σκοπός είναι το κάθε σημείο να κινηθεί σε μία από τις πιθανές κατευθύνσεις αναζήτησης οι βρίσκονται στην επιφάνεια αυτής της σφαίρας κατά μήκος της ορθογώνιας βάσης του χώρου. Για παράδειγμα, στην περίπτωση του 3-διάστατου χώρου κατά τις κατευθύνσεις $\pm x, \pm y, \pm z$ στη σφαίρα που φαίνεται στο Σχ. 1. Αυτό δημιουργεί τον πίνακα κατευθύνσεων αναζήτησης S και περιλαμβάνεται στον Αλγ. 3.3.

Ο αλγόριθμος σε κάθε εποχή εξετάζει όλες τις κατευθύνσεις και επιλέγει να μετακινήσει το κάθε σημείο προς εκείνη την κατεύθυνση που παράγει το ελάχιστο σφάλμα. Σκοπός λοιπόν είναι να υπολογιστεί το τελικό error e^* ελαχιστοποιώντας την συνάρτηση σφάλματος 1 , $e(k) = f(T, D(k))$, για κάθε σημείο σε κάθε εποχή. Εαν η μείωση του σφάλματος φτάσει σε ένα πλατό, το οποίο για μία συγκεκριμένη πολύ μικρή σταθερά $\epsilon > 0$ εκφράζεται ως, $e^{(k)} - e^* < \epsilon \cdot e^{(k)}$ τότε μειώνουμε την ακτίνα αναζήτησης στο μισό και συνεχίζουμε στην επόμενη εποχή. Η διαδικασία τελικά θα σταματήσει όταν η ακτίνα r γίνει πολύ μικρή, δηλαδή $r < \delta$ όπου δ είναι μια μικρή σταθερά, όπως φαίνεται στον Αλγόριθμο 2.

0.2.2 Προτεινόμενη Μέθοδος Βελτιστοποίησης Πολυπλοκότητας

Η ιδέα των σημείων Landmark [2, 3] φάνηκε πολύ υποσχόμενη όταν προσπαθούσαμε να βρούμε τρόπους για να βελτιώσουμε την πολυπλοκότητα του αλγορίθμου PSMDS. Η κεντρική ιδέα της μεθόδου Landmarks είναι ότι δεν είναι απαραίτητο να υπολογίζουμε αποστάσεις μεταξύ όλων των σημείων στον πίνακα αποστάσεων. Τουλάχιστον $n \geq L+1$ σημεία (σημεία Landmark) είναι επαρκή, από τα οποία μπορούμε να υπολογίσουμε τις αποστάσεις μας από όλα τα άλλα σημεία.

Εάν γνωρίζουμε τις θέσεις των σημείων Landmark, τότε όλα τα υπόλοιπα σημεία μπορούν να τοποθετηθούν μοναδικά στο χώρο σε σχέση με αυτά. Αυτός ο ισχυρισμός είναι ανάλογος

με τη μέθοδο GPS, όπου για να βρούμε την ακριβή θέση ενός σημείου, είναι αρκετό να γνωρίζουμε την απόστασή του από έναν πεπερασμένο αριθμό σημείων. Τα σημεία Landmark μπορούν να είναι τυχαία όσο ο αριθμός τους είναι μεγαλύτερος από το $L+1$ (όπου L είναι η διάσταση της ενσωμάτωσης).

Στην είσοδο του MDS (Multidimensional Scaling), έχουμε τον πίνακα αποστάσεων $N \times N$ με τις αποστάσεις κάθε σημείου από όλα τα άλλα. Κατά τη διάρκεια του αλγορίθμου, προσπαθούμε να βρούμε διανύσματα διάστασης L έτσι ώστε ο τελικός πίνακας αποστάσεων $N \times N$ να είναι όσο το δυνατόν πιο κοντά στον αρχικό. Ο πίνακας αποστάσεων γίνεται $n \times N$, όπου $n \ll N$.

Στην αρχή, εκτελούμε το MDS κανονικά σαν να υπήρχαν μόνο τα σημεία Landmark, τα οποία θα τοποθετηθούν κανονικά στην χώρο με μειωμένες διαστάσεις. Στη συνέχεια, τα υπόλοιπα σημεία $N-n$ θα τοποθετηθούν στον χώρο με βάση τις αποστάσεις τους από αυτά τα σημεία Landmark.Η διαφορά στον αλγόριθμο είναι ότι ο πίνακας αποστάσεων εισόδου P και ο πίνακας αποστάσεων D, έχουν διάσταση $N \times n$ αντί για $N \times N$, όπου $n \ll N$. Αυτό ειδικά όταν το N είναι πολύ μεγάλο.

Η διαδικασία αρχικοποίησης του αλγορίθμου περιλαμβάνει τα εξής βήματα:

- Δημιουργία n τυχαίων σημείων με τις τελικές επιθυμητές διαστάσεις L. $X_a^{(0)} = [x_1^{(0)}, x_2^{(0)}, ..., x_n^{(0)}] \in \mathbb{R}^{n \times L}$. Ως n αναφερόμαστε στον αριθμό των σημείων Landmarks.

- Υπολογισμός του πίνακα αποστάσεων μεταξύ των σημείων Landmark. Το στοιχείο $d_{ij}^{(0)}$ είναι η Ευκλείδεια απόσταση μεταξύ των διανυσμάτων $x_i^{(0)}$ και $x_j^{(0)}$ του $X_a^{(0)}$.

- Υπολογισμός του αρχικού σφάλματος προσέγγισης $e^{(0)} = ||\mathbf{P} - \mathbf{D}||F^2$, όπου το e αντιπροσωπεύει το μέσο τετραγωνικό σφάλμα (MSE) ανά στοιχείο μεταξύ των δύο πινάκων. Η συνάρτηση στόχου που βελτιστοποιείται για όλα τα PSMDS παρουσιάζεται παρακάτω και είναι η συνάρτηση stress για το MDS:

$$f(\mathbf{P}, \mathbf{D}) = ||\mathbf{P} - \mathbf{D}||F^2 = \sum_{i=1}^{n} \sum_{j=1}^{n} (t_{ij} - d_{ij})^2, \quad ; \text{όπου}; \mathbf{P}, \mathbf{D} \in \mathbb{R}^{n \times n} \tag{2}$$

Σε συνέχεια των βημάτων αρχικοποίησης, ο αλγόριθμος LPS-MDS, όπως και ο PSMDS, σε κάθε εποχή εξετάζει όλες τις κατευθύνσεις και επιλέγει να μετακινήσει το κάθε σημείο προς εκείνη την κατεύθυνση που παράγει το ελάχιστο σφάλμα. Σκοπός λοιπόν είναι να υπολογιστεί το τελικό error e^* ελαχιστοποιώντας την συνάρτηση σφάλματος 1 , $e(k) = f(T, D(k))$, για κάθε σημείο σε κάθε εποχή. Εαν η μείωση του σφάλματος φτάσει σε ένα πλατό, το οποίο για μία συγκεκριμένη πολύ μικρή σταθερά $\epsilon > 0$ εκφράζεται ως, $e^{(k)} - e^* < \epsilon \cdot e^{(k)}$ τότε μειώνουμε την ακτίνα αναζήτησης στο μισό και συνεχίζουμε στην επόμενη εποχή. Η διαδικασία τελικά θα σταματήσει όταν η ακτίνα r γίνει πολύ μικρή, δηλαδή $r < \delta$ όπου δ είναι μια μικρή σταθερά, όπως φαίνεται στον Αλγόριθμο 2.

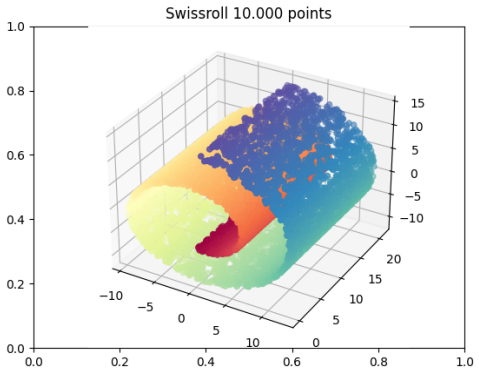
Τώρα όλα τα σημεία Landmark $X^{(a)}$ έχουν τοποθετηθεί στον ενσωματωμένο χώρο. Θα επαναλάβουμε την ίδια διαδικασία με τα υπόλοιπα σημεία $N-n$. Τώρα όλες οι αποστάσεις θα γίνονται ανάμεσα στα n Landmarks και $N-n$ υπολοιπόμενα σημεία.

0.2.3 Αποτελέσματα

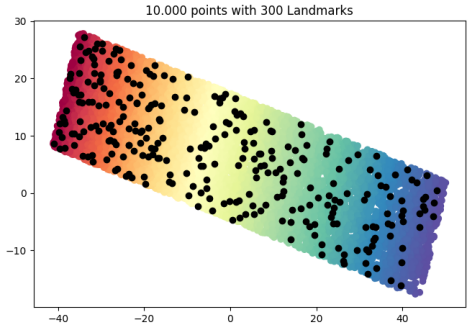
Σε αυτήν την υποενότητα παραθέτουμε κάποια από τα αποτελέσματα των πειραμάτων μας.

0.2.3.1 Συνθετικά δεδομένα

Σε αυτή την πειραματική διάταξη πραγματοποιούμε μη γραμμική μείωση διαστατικότητας σε ένα 3D Swissroll (όπως φαίνεται στο Σχήμα 1a) χρησιμοποιώντας το Landmarks MDS. Η υπόθεση εδώ είναι ότι τα δεδομένα βρίσκονται σε έναν πολυτοπισμό που είναι τοπικά Ευκλείδιος και έτσι οι γεωδεσικές αποστάσεις θα διατηρούσαν την αληθινή δομή των δεδομένων κατά την εκτέλεση του MDS. Στο Σχήμα 1b φαίνεται το ξετύλιγμα του Swissroll σε 2 διαστάσεις, καθως και τα landmarks που έχει επιλέξει ο αλγόριθμος.



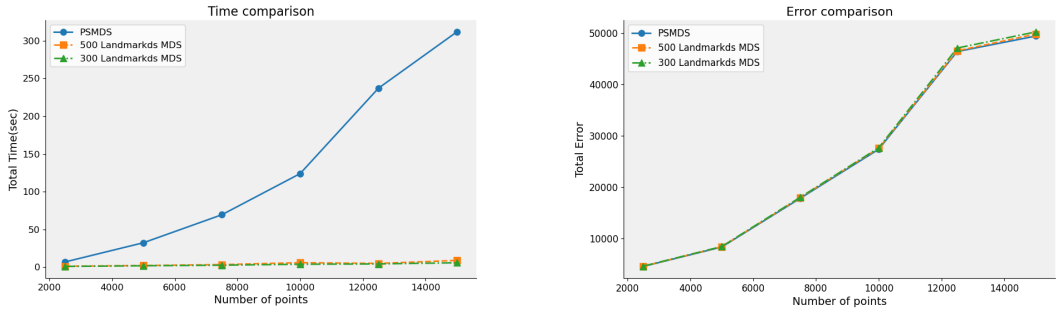
(a) *Πρωτότυπο τρισδιάστατο Swissroll*



(b) *Ξετύλιγμα του Swissroll σε 2 διαστάσεις με χρήση του LPS-MDS. Τα μαύρα σημεία είναι τα Landmarks*

Figure 1. *Απλοποίηση του τρισδιάστατου Swissroll σε δύο διαστάσεις*

Στα ακόλουθα γραφήματα, παρουσιάζουμε μια συγκριτική ανάλυση μεταξύ του Landmarks MDS και του Pattern Search MDS όσον αφορά τον Χρόνο και το Σφάλμα για σύνολα δεδομένων με διαφορετικό αριθμό σημείων (2500, 5000, 7500, 10000, 12500, και 15000). Οι παρατηρήσεις μας αποκαλύπτουν ότι το σφάλμα παραμένει σταθερό μεταξύ των δύο αλγορίθμων, δείχνοντας παρόμοια ποιότητα αποτελεσμάτων. Όπως ήταν αναμενόμενο, υπάρχουν σημαντικές βελτιώσεις στους χρόνους εκτέλεσης όταν χρησιμοποιούμε το Landmarks MDS, ιδιαίτερα καθώς το μέγεθος του συνόλου δεδομένων αυξάνεται. Η προσέγγιση του Landmark MDS δείχνει συνεχώς ταχύτερη σύγκλιση, μειώνοντας τον απαιτούμενο χρόνο για να επιτευχθούν τα επιθυμητά αποτελέσματα. Αυτά τα ευρήματα τονίζουν την αποτελεσματικότητα και την απόδοση του Landmarks MDS στη διαχείριση μεγαλύτερων συνόλων δεδομένων, καθιστώντας το μια υποσχόμενη επιλογή για εργασίες μείωσης διαστατικότητας.



(a) *Σύγκριση στον χρόνο.* **(b)** *Σύγκριση στο σφάλμα.*

Figure 2. *Σύγκρισεις ανάμεσα στους αλγορίθμους PSMDS και LPS-MDS. Το LPS-MDS έχει εξαιρετικά αποτελέσματα σε όλες τις περιπτώσεις και για μεγάλο αριθμό σημείων*

0.2.3.2 MNIST Σύνολο Δεδομένων

Το σύνολο Δεδομένων MNIST, είναι μια βάση δεδομένων χειρόγραφων ψηφίων που έχει χρησιμοποιηθεί ευρέως για εκπαίδευση και δοκιμασία στον τομέα της μηχανικής μάθησης. Επιλέξαμε συγκεκριμένα να εργαστούμε με ένα υποσύνολο χειρόγραφων ψηφίων, περιορίζοντας την επιλογή μας σε τέσσερα από τα δέκα διαθέσιμα ψηφία: 0, 1, 4, 8. Από αυτές τις τέσσερις κατηγορίες, επιλέξαμε τυχαία 1000 εικόνες για την ανάλυσή μας. Το αρχικό μας βήμα περιλάμβανε τον υπολογισμό του πίνακα Ευκλείδειων αποστάσεων, ο οποίος απεικονίζει τις δυαδικές αποστάσεις μεταξύ όλων των εικόνων στον χώρο των pixel. Κατόπιν, πραγματοποιήσαμε μείωση διαστατικότητας, με στόχο την απεικόνιση των δεδομένων σε δύο διαστάσεις. Χρησιμοποιήσαμε και τα δύο αλγορίθμους, το Pattern Search MDS και το MDS με Ορόσημα για αυτό το έργο. Εξαιρετικά, και οι δύο αλγόριθμοι πέτυχαν παρόμοιο επίπεδο ακρίβειας, όπως δείχνουν τα σφάλματα Stress τους. Για να βοηθήσουμε στην οπτικοποίηση των αποτελεσμάτων μας, ανατίθεσαμε διακριτά χρώματα στις διαφορετικές κατηγορίες χειρόγραφων ψηφίων, αντιστοιχίζοντας τα 0, 1, 4, 8 στα κόκκινο, πορτοκαλί, κίτρινο, μωβ αντίστοιχα. Οι προκύπτουσες ενσωματώσεις αποκάλυψαν ένα ενδιαφέρον αποτέλεσμα. Οι αλγόριθμοι MDS διατηρούσαν επιτυχώς τις γεωμετρικές σχέσεις μεταξύ των χειρόγραφων ψηφίων. Ειδικότερα, τα ψηφία που ανήκαν στην ίδια κατηγορία είχαν την τάση να συσσωρεύονται μαζί σε συμπαγείς υποχώρους 2D Ευκλείδειας, παρουσιάζοντας την αποτελεσματικότητα της προσέγγισής μας.

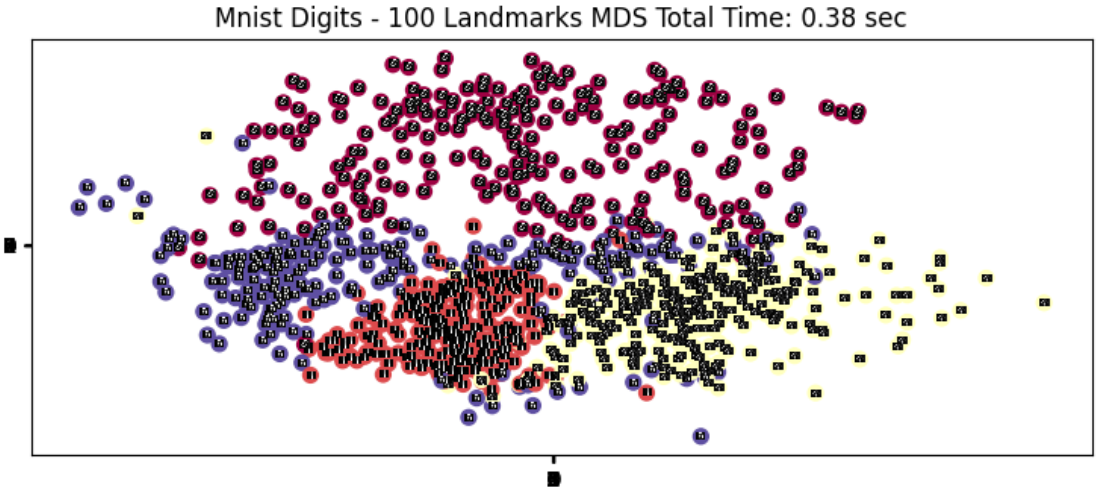


Figure 3. *Landmark MDS*

Για να συνεχίσουμε, χρησιμοποιούμε ένα υποσύνολο από 3000 εικόνες και πραγματοποιούμε γραμμική μείωση διάστασης χρησιμοποιώντας το Pattern Search MDS και το Landmark PSMDS με 100 και 300 σημεία αναφοράς για να λάβουμε ενσωματώσεις 10 διαστάσεων. Για την αξιολόγηση της απόδοσης των αλγορίθμων, εκτελείται ταξινόμηση K-Nearest Neighbor (KNN) χρησιμοποιώντας διαφορετικές παραμέτρους κοντινότερων γειτόνων. Επίσης, διεξάγω ταξινόμηση KNN στον αρχικό χώρο χωρίς να πραγματοποιώ οποιαδήποτε μείωση διάστασης για να αποκτήσω καλύτερη εποπτεία της ακρίβειας. Χρησιμοποιούνται 2700 εικόνες ως σύνολο εκπαίδευσης και 300 ως σύνολο δοκιμής. Ο χρόνος εκτέλεσης βελτιώνεται με τα σημεία αναφοράς, ενώ η χρήση 300 σημείων αναφοράς αντί για 100 οδηγεί σε καλύτερες βαθμολογίες ακρίβειας.

Για να εμβαθύνουμε περαιτέρω στην ανάλυσή μας, δουλέψαμε με ένα υποσύνολο που αποτελείται από 3000 εικόνες και πραγματοποιήσαμε μείωση διάστασης σε 10 διαστάσεις με και τις δύο τεχνικές. Στη διαδικασία αξιολόγησής μας, χρησιμοποιήσαμε την ταξινόμηση K-Nearest Neighbor (KNN), πειραματιζόμενοι με διάφορες ρυθμίσεις κοντινότερων γειτόνων. Επιπλέον, διεξάγαμε ταξινόμηση KNN στα αρχικά δεδομένα, παρακάμπτοντας οποιαδήποτε μείωση διάστασης. Αυτή η προσέγγιση παρείχε πολύτιμες πληροφορίες σχετικά με την ακρίβεια των αποτελεσμάτων μας. Για τα πειράματά μας, χωρίσαμε τα δεδομένα σε 2700 εικόνες για εκπαίδευση και 300 εικόνες για δοκιμαστικούς σκοπούς. Είναι αξιοσημείωτο ότι η χρήση σημείων αναφοράς βελτίωσε σημαντικά τον χρόνο εκτέλεσης. Επιπλέον, η χρήση 300 σημείων αναφοράς, αντί για 100, οδήγησε σε καλύτερες βαθμολογίες ακρίβειας, υπογραμμίζοντας την αποτελεσματικότητα της μεθοδολογίας μας.

Method	Dimensions	Time(sec)	K=3	K=5	K=7	K=9
Initial	784	-	0.90	0.89	0.91	0.92
PSMDS	10	140.54	0.90	0.87	0.89	0.89
100LandmarksPSMDS	10	**11.53**	0.83	0.86	0.82	0.81
300LandmarksPSMDS	10	**19.55**	**0.89**	**0.89**	0.88	0.86

Table 1. *Σύγκριση τεχνικών μείωσης διαστατικότητας για το σύνολο δεδομένων MNIST με διάσταση ενσωμάτωσης ίση με 10. Για την ταξινόμηση, χρησιμοποιείται ο αλγόριθμος KNN με διαφορετικό αριθμό πλησιέστερων γειτόνων (παράμετρος K). Η στήλη "Χρόνος" καθορίζει το χρονικό διάστημα που χρειάστηκε κάθε μέθοδος για να υπολογίσει όλα τα διανύσματα ενσωμάτωσης. Το "Αρχικό" αντιστοιχεί στην εκτέλεση του KNN στο αρχικό διανύσμα υψηλής διαστατικότητας χωρίς καμία μείωση διαστατικότητας.*

0.3 Πρόβλημα Set-MDS

Στο δεύτερο μέρος της διπλωματικής, δίνουμε τον ορισμό του προβλήματος Set-MDS και προσπαθούμε να δώσουμε μία προσεγγιστική λύση αναπτύσσοντας τον αλγόριθμο COSMOS. Το πρόβλημα που προσπαθούμε να επιλύσουμε είναι παρόμοιο με το MDS, όμως τώρα τα δεδομένα είναι οργανωμένα σε σύνολα. Ο χώρος μας πλέον δεν είναι μετρικός, καθώς οι αποστάσεις ανάμεσα σε σύνολα σημειων παραβιάζουν την τριγωνική ανισότητα. Δεδομένου λοιπόν ενός μετρικού χώρου και ενός πίνακα αποστάσεων συνόλων $\mathbf{\Delta}$, ο οποίος δεν είναι απαραίτητα μετρικός, κατασκευάζουμε έναν πίνακα αναπαράστασης $\mathbf{X}$ με σημεία L- διαστάσεων, καθώς και μια διαίρεση των σημείων σε σύνολα, ώστε ο πίνακας πίνακας που προκύπτει από το $\mathbf{X}$ να είναι όσο το δυνατόν πιο κοντά στο $\mathbf{\Delta}$.

Το πρόβλημα Set-MDS, όπως ορίζεται, αποτελεί μια υπολογιστική πρόκληση καθώς κατατάσσεται στα προβλήματα NP-complete. Αποτελεί πρόβλημα βελτιστοποίησης όπου ο στόχος είναι να ελαχιστοποιηθεί η διαφορά μεταξύ ενός δοσμένου πίνακα αποστάσεων και του πίνακα αποστάσεων που προκύπτει από μια προτεινόμενη αναπαράσταση σημείων. Αυτή η διαδικασία είναι περίπλοκη και οδηγεί σε έναν τεράστιο χώρο αναζήτησης λύσεων με πολλά δυνητικά τοπικά ελάχιστα. Επιπλέον, η οργάνωση των σημείων σε ξεχωριστές ομάδες (σύνολα) προσθέτει έναν επιπλέον βαθμό πολυπλοκότητας, καθώς ο αριθμός των τρόπων για να γίνει ο διαχωρισμός αυτός αυξάνεται γρήγορα με περισσότερα σημεία.

Ο αλγόριθμος COSMOS είναι, για την ακρίβεια, μια επέκταση του αλγορίθμου Pattern Search MDS και έχει σχεδιαστεί ειδικά για να λειτουργεί με σύνολα αντικειμένων ή σημείων δεδομένων, αντί για μεμονωμένα σημεία, προκειμένου να επιλύσει το πρόβλημα Set-MDS. Αυτή η ευρηστική προσέγγιση έχει σχεδιαστεί για να παρέχει αποδεκτές λύσεις με ευκολία εντός πολυωνυμικού χρόνου, μια διαπίστωση που επιβεβαιώνεται από τα πειραματικά μας ευρήματα.

0.3.1 Προτεινόμενη Μέθοδος - Ο αλγόριθμος COSMOS

Η είσοδος στον αλγόριθμό είναι ένας πίνακας απόστάσεων $\mathbf{\Delta}$, ο οποίος περιέχει αποστάσεις ανάμεσα σε sets, η διάσταση L του χώρου καθώς και ο αριθμός των διασπάσεων k. Ο αλγόριθμος ξεκινάει παράγωντας σημεία διαστάσεων L και τρέχοντας τον LPS-MDS αλ-

γόριθμο έτσι ώστε να αποκτήσει μια καλύτερη αρχικοποίηση των σημείων. Στην συνέχεια ο αλγόριθμος θα αρχίσει να διασπά k σημεία.

- Βρείτε τη νέα διαίρεση δημιουργώντας $2 * L$ τυχαία σημεία με διαστάσεις L στον ενσωματωμένο χώρο και αναθέτοντας καθένα από αυτά σε ένα σύνολο. Επιλέξτε τον συνδυασμό του σημείου και του συνόλου που ελαχιστοποιεί το σφάλμα περισσότερο. Το $\mathbf{X}$ έχει τώρα ένα ακόμη σημείο.

- Υπολογίστε τις Ευκλείδειες αποστάσεις μεταξύ του νεοσυσταθέντος σημείου διαίρεσης και όλων των άλλων σημείων. Στη συνέχεια, προσθέστε μια νέα γραμμή και στήλη στον πίνακα D_{points}, ο οποίος αποθηκεύει τις Ευκλείδειες αποστάσεις μεταξύ όλων των σημείων. Αυτός ο πίνακας αποτελεί τη βάση για την κατασκευή του πίνακα απόστασης συνόλων, που συμβολίζεται ως D_{sets}, όπου το στοιχείο d_{ij} αντιπροσωπεύει την κοινή απόσταση συνόλων, όπως καθορίζεται στην Εξίσωση 5.1.

- Μετακινήστε κάθε σημείο x_i σε έναν σφαίρα ακτίνας r γύρω από το σημείο x_i, όπως στο Σχήμα 3.1. Επιλέξτε την κατεύθυνση που ελαχιστοποιεί το σφάλμα περισσότερο. Το σφάλμα είναι η ακόλουθη εξίσωση: όπου το e αντιπροσωπεύει το μέσο τετραγωνικό σφάλμα (MSE) στοιχείο προς στοιχείο μεταξύ των δύο πινάκων αποστάσεων συνόλων. Η συνάρτηση στόχου είναι παρόμοια με τη συνάρτηση MDS Stress, αλλά τα πίνακες τώρα είναι πίνακες συνόλων.

$$f(\mathbf{D}, \mathbf{\Delta}) = ||\mathbf{D} - \mathbf{\Delta}||F^2 = \sum i = 1^n \sum_{j=1}^{n} (d_{ij} - p_{ij})^2, \quad ; \text{όπου; } \mathbf{\Delta}, \mathbf{D} \in \mathbb{R}^{\nu \times \nu} \tag{3}$$

0.3.2 Αποτελέσματα

Σε αυτήν την υποενότητα συγκεντρώνουμε αποτελέσματα των πειραμάτων μας που αφορούν στην αξιολόγηση του αλγορίθμου COSMOS.

0.3.3 Συνθετικά δεδομένα

Για τη δημιουργία των Συνθετικών Δεδομένων για αυτό το πείραμα, ξεκινήσαμε τη διαδικασία δημιουργώντας μ τυχαία σημεία, καθένα με L διαστάσεις, όπου οι τιμές κυμαίνονται από -1 έως 1. Στη συνέχεια, αυτά τα μ σημεία ανατέθηκαν τυχαία σε ν σύνολα. Το αποτέλεσμα περιλαμβάνει έναν πίνακα αποστάσεων, τον $\mathbf{\Delta} \in \mathbb{R}^{\nu \times \nu}$, και έναν πίνακα δεικτών, που συμβολίζεται ως $M_{sets} \in \mathbb{R}^{1 \times \mu}$, λειτουργώντας ως αντιστοίχιση μεταξύ των διανυσμάτων που παράγουν τον $\mathbf{\Delta}$ και των ν συνόλων. Η προσδοκία μας είναι ότι ο αλγόριθμος COSMOS, δεδομένου του πίνακα $\mathbf{\Delta}$ και του αριθμού k στην είσοδο, θα σπάσει ακριβώς τα σωστά σημεία.

Στον πίνακα 2 παρακάτω συνοψίζονται τα αποτελέσματα πολλαπλών πειραμάτων. Διενεργήσαμε πειράματα με διάφορο αριθμό σημείων (1000, 500, 100) και διάφορο αριθμό διαιρέσεων k (1, 5, 10, 20, 50, 100, 150, 200, 300, 400, 500), πραγματοποιώντας ταυτόχρονα μείωση διαστατικότητας από 3 διαστάσεις σε 2. Στόχος μας είναι να αξιολογήσουμε εάν ο αλγόριθμος COSMOS επιλέγει να κάνει τις σωστές διαιρέσεις. Συγκρίνουμε τα αναμενόμενα

τελικά σύνολα M_{sets} που προκύπτουν από τα συνθετικά δεδομένα με την έξοδο του αλγορίθμου. Τα αποτελέσματα δείχνουν ότι ο αλγόριθμος COSMOS μπορεί να αναγνωρίσει με ποσοστό άνω του 65% τις σωστές διαιρέσεις.

Διαιρέσεις / Σημεία	1	10	20	50	100	150	200	300	400	Μέσος Όρος
1000	1.00	0.80	0.75	0.72	0.64	0.65	0.65	0.65	0.62	0.70
500	1.00	0.80	0.65	0.68	0.60	0.71	0.58	0.62	0.62	0.66
100	1.00	0.60	0.60	0.70						0.67

Table 2. *Ο πίνακας συνοψίζει τα αποτελέσματα πολλαπλών πειραμάτων. Δείχνει το ποσοστό επιτυχημένων διαιρέσεων σε πειράματα με διάφορο αριθμό σημείων μ και διαιρέσεις k*

0.3.4 Ομοιότητα Λέξεων - Men & Simlex

Για αυτό το πείραμα χρησιμοποιούμε τα σύνολα δεδομένων MEN και SimLex-999. Και τα δύο σύνολα δεδομένων αποτελούνται από λίστες ζευγαριών λέξεων, όπου κάθε ζευγάρι συσχετίζεται με ένα βαθμό ομοιότητας. Αυτός ο βαθμός υπολογίστηκε παίρνοντας τον μέσο όρο των ομοιοτήτων που παρείχαν ανθρώπινοι αξιολογητές.

Στο πείραμα χρησιμοποιούμε τα 300-διάστατα διανύσματα GloVe, ως είσοδο στον αλγόριθμο. Μειώνουμε τη διαστατικότητα των διανυσμάτων στην επιθυμητή διάσταση L και υπολογίζουμε το συντελεστή συσχέτισης Spearman μεταξύ των ανθρώπινα παρεχόμενων και των αυτόματα υπολογισμένων βαθμών ομοιότητας. Τα αποτελέσματα συνοψίζονται στον πίνακα 2 για $L = 10$. Παρατηρούμε ότι οι PSMDS και Landmark PSMDS παράγουν τα καλύτερα αποτελέσματα για το MEN, η UMAP είναι αποτελεσματική για το SimLex-999, ενώ ο αλγόριθμος COSMOS έχει επίσης καλά αποτελέσματα.

Μέθοδος	Διαστάσεις	MEN	SimLex-999
-	300	0.74	0.37
PSMDS	10	**0.68**	0.30
300 Landmark PSMDS	10	**0.67**	0.23
100 Landmark PSMDS	10	**0.61**	0.21
SMACOF	10	0.56	0.23
UMAP	10	0.29	0.44
COSMOS(2 διαιρέσεις)	10	0.59	0.25
COSMOS(20 διαιρέσεις)	10	-	0.25

Table 3. *Σύγκριση τεχνικών μείωσης διαστάσεων σε ομοιότητα λέξεων με χρήση των συνόλων δεδομένων MEN και SimLex-999.*

Chapter 1

Introduction

1.1 Motivation

In a time when we're surrounded by more data than ever before, it's crucial to be able to simplify complicated information into something we can easily understand and use. Data visualization turns data into graphics, helping people make sense of it. It's like building a bridge from a sea of numbers to our human minds. From the days of Florence Nightingale, who used diagrams to show why soldiers were dying in the Crimean War, to today's dynamic charts that help with important business choices, the skill of turning data into visuals has evolved greatly.

Data visualization relies critically on the process of dimensionality reduction, especially in today's era where data is exceedingly complex and multidimensional. To effectively visualize and interpret this data, it's crucial to distill it down to its most significant aspects. Numerous established techniques are employed across various fields to achieve this, enabling the extraction of key insights from the high-dimensional data. These methods not only simplify the data for analysis but also uncover underlying patterns and relationships, making complex information accessible and comprehensible. For example, Uniform Manifold Approximation and Projection (UMAP) is used in medical data analysis to visualize patient data based on various health metrics. This can help in identifying subgroups of patients with similar health profiles or responses to treatment. Also, researchers use Isomap in the analysis of brain imaging data to uncover patterns related to brain activity, connectivity, and neurological diseases.

In this thesis we are having a insight to Multidimensional Scaling (MDS) that is used across various fields to analyze and visualize complex, high-dimensional data. In psychology, MDS helps in understanding human perception by visualizing how individuals perceive similarities or differences among various stimuli. It's used in areas like psychometrics to analyze questionnaire data. In genomics, MDS is used to visualize genetic distance or similarity among different species or individuals. This is crucial for understanding genetic relationships, evolution, and population genetics. In health sciences, MDS can help visualize patient data, similarities in symptoms or diseases, and other medical data for better understanding of health patterns and diagnosis. MDS assists in exploring linguistic data, such as phonetic data, to uncover patterns and relationships in languages and dialects.

Furthermore in this thesis, we explore a very interesting phenomenon that we are facing

in multiple fields in our world; the violation of triangle inequality across the data. We are going to give an example in word embeddings, and more specifically we have worked on the phenomenon of 'semantic ambiguity' or 'polysemy' of the words. We have developed an innovative algorithm as an extension of MDS that identifies the dual meaning of the word 'book' and visualizes it two times; once close to 'travel' and once close to 'library'. These two words are semantically far away from each other, and this phenomenon violates the triangle inequality.

Beyond the realm of linguistics, similar complexities are observed in various fields. For instance, in social network analysis, a person may have different types of relationships with others —*friends, colleagues, family*— each representing different social contexts. Similarly, in recommender systems, a single item (like a movie or product) may bridge disparate user groups with unique tastes, lacking mutual resemblance. In genomics, a gene's involvement across multiple biological pathways or diseases illustrates its proximity to diverse gene clusters, which may not exhibit a close relationship amongst themselves. The healthcare sector presents another example, wherein a symptom could signify various diseases, each proximate in the symptom-disease matrix, yet the diseases themselves remain unalike. Lastly, in robotics, a robot's action may serve multiple tasks, such as 'grabbing', which is relevant to both 'cooking' and 'assembly', despite these tasks diverging in the context of a robot's operational paradigm.

1.2 Thesis Contributions

The contribution of this thesis consists of two parts. In the first part, we attempt to improve the complexity of the Pattern Search MDS algorithm, as described in [1], using the Landmark Points method as outlined in [2, 3]. Pattern Search MDS is an innovative algorithm that is an extension to classical MDS and uses derivative-free techniques, so no gradient descent is applied. In addition to this approach, in this thesis we are using the Landmark Points in order to improve the complexity of the algorithm. Our results are exceptional and they dramatically reduce the execution time of the algorithm without compromising the quality of the outcome.

In the second part of our thesis, we are developing an extension of MDS, the COSMOS (COmmon Sense Multidimensional Optimization Splitter) algorithm, a heuristic algorithm that is trying to solve the Set-MDS problem. Below we will give an informal definition of the Set-MDS Problem:

Informal Definition of Set-MDS Problem

Given a metric space $\mathbf{X} \subseteq \mathbf{R}^n$ and $\nu, \mu, d \in \mathbf{N}_+$ such that $\nu \leq \mu$ and a dissimilarity matrix (non- necessarily metric) $\mathbf{D}_{\nu \times \nu}$ of ν-sets of μ points, construct a representation matrix $\mathbf{X}_{\mu \times d}$ and a separation of μ points $\mathbf{S} = \{S_1, \cdots, S_\nu\} : S_i \cap S_j = \phi$ and $| \bigcup S_i | = \mu$ such that the induced dissimilarity matrix of $\mathbf{X}$ is as much as possible close to $\mathbf{D}$.

The Set-MDS problem, as defined, presents a computational challenge indicative of NP-complete problems. Primarily, it involves an optimization task where the goal is to minimize the discrepancy between a given dissimilarity matrix and the induced dissimilarity matrix of a proposed representation. This task is complicated by its non-linear

and non-convex nature, leading to a vast search space with many potential local minima. Additionally, organizing points into separate groups (sets) adds another layer of complexity, as the number of ways to do this increases rapidly with more points. The problem's involvement with high-dimensional spaces further adds to its complexity.

COSMOS algorithm, is in fact an extension of the Pattern Search MDS and is specifically designed to work with sets of objects or data points, rather than individual points, in order to solve the Set-MDS problem. This heuristic approach is engineered to efficiently provide approximate solutions within polynomial time, a claim substantiated by our experimental findings.

1.3 Chapter Outline

The thesis is structured as follows:

Chapter 2 provides an overview of machine learning fundamentals and the process of dimensionality reduction. Initially, the first chapters cover key ideas in machine learning, such as Supervised, Semi-Supervised, Reinforcement, and Unsupervised learning. This thesis focuses on the application of unsupervised learning, specifically using the PSMDS and COSMOS algorithms. Later parts of the chapter discuss briefly the importance of dimensionality reduction, introduce manifold learning, and present some widely recognized algorithms from existing literature.

Chapter 3 focuses solely on one dimensionality reduction method, MultiDimensional Scaling (MDS). We choose to discuss MDS separately, as it is central to our research. We start by discussing the established versions: Classical, Metric, and non-Metric MDS, as found in existing literature. Section 3.4 introduces a new method called Pattern Search MDS (PSMDS). We briefly explain this method as it is prior to our work. This innovative technique approaches nonlinear manifold learning without relying on gradient descent optimization. It builds on Classical MDS, aiming to optimize the placement of data points in a new space. Rather than using gradients, it uses 'moves' within a set radius to assess and adjust the positions of these points.

Chapter 4 discusses an improvement to the PSMDS algorithm by incorporating Landmark points. We highlight this addition as a significant contribution of the thesis, noting its role in increasing the complexity of the algorithm and yielding notable results in our experiments.

Chapter 5 introduces our proposed COSMOS algorithm. We start by mathematically defining the Set-MDS problem and then briefly explain the development process of our algorithm. The chapter concludes with a presentation of some experimental results.

Finally, Chapter 6 includes conclusions inferred from the thesis and provides an outlook into the future.

Chapter 2

Machine Learning and Dimensionality Reduction

2.1 Defining Machine Learning and Artificial Intelligence

Artificial Intelligence (AI) and Machine Learning (ML) are two different yet closely related terms that are often mistakenly used interchangeably. While all machine learning is a form of AI, not all AI relies on machine learning. Together, both AI and ML are reshaping industries by offering data-driven solutions and augmenting human capabilities.

2.1.1 Unraveling Artificial Intelligence (AI)

Artificial Intelligence refers to the simulation of human intelligence in machines that are programmed to think and act like humans. It encompasses a broad range of technologies and methods that allow machines to perform tasks that typically require human cognitive functions, such as problem-solving, pattern recognition, understanding natural language, and decision-making. AI can be rule-based, where it operates under a set of explicit instructions, or it can use machine learning algorithms to adapt to data and improve its operations over time. The ultimate goal of AI research is to create systems that can perform tasks that, when done by humans, are considered to require intelligence. This includes tasks like visual perception, speech recognition, strategic planning, and even artistic endeavors. AI applications are pervasive in today's world, impacting sectors from healthcare to finance to entertainment.

2.1.2 The Role of Machine Learning (ML)

Machine learning is a subfield of AI that plays a pivotal role in realizing the AI dream. ML focuses on developing algorithms that allow computers to learn from and make decisions based on data. Instead of being explicitly programmed to perform a task, a machine learning model uses patterns and inference to improve its performance over time. The learning process involves feeding large amounts of data to the algorithm and allowing it to adjust its internal parameters to make accurate predictions or classifications. ML encompasses various techniques including supervised learning, where the model is trained with labeled data; unsupervised learning, where the data isn't labeled; and reinforcement learning, where an agent learns by interacting with its environment and receiving rewards or penalties for its actions. Machine learning applications are diverse and range from email

filtering and recommendation systems to image recognition and autonomous driving.

2.2 Types of Machine Learning

Machine Learning methods are typically segmented into three main categories based on the type of feedback provided to the learning algorithm: Supervised, Unsupervised, and Reinforcement Learning. Additionally, some systems employ a combination of methods, as seen in Semi-supervised Learning that combines both Supervised and Unsupervised methods.

2.2.1 Supervised Learning

In supervised learning, the objective is to derive a function based on provided input-output pairs. The system receives sample inputs alongside their corresponding desired outputs, known as labeled training data, provided by an instructor. The goal is to learn a mapping or a function that can successfully associate inputs with outputs. By examining the labeled data, a supervised learning technique can deduce a function. Ideally, this function will accurately predict outputs for new, previously unseen input examples.

2.2.2 Unsupervised Learning

Unsupervised Learning deals with unlabeled data, where there are no predefined output labels. The algorithm tries to find inherent patterns or structures in the data. Also known as self-organization, unsupervised learning allows for modeling of probability densities over inputs. Clustering and dimensionality reduction are common tasks in unsupervised learning.

2.2.3 Semi-Supervised Learning

Semi-supervised Learning approach combines elements of both unsupervised learning (without labeled training data) and supervised learning (exclusively using labeled training data). Utilizing unlabeled data alongside a limited set of labeled data can enhance the learning outcome. Gathering labeled data is a costly procedure usually demands expertise or specific experiments. For this reason fully labeled datasets can be often impractical. In contrast, procuring unlabeled data is more cost-effective, making semi-supervised learning highly beneficial in practical scenarios.

2.2.4 Reinforcement Learning

In Reinforcement Learning approach an agent learns how to behave in an environment by taking actions and receiving rewards or penalties in return. Instead of being explicitly taught, the agent learns from trial and error, aiming to discover the best strategy, called a policy, to obtain the maximum cumulative reward over time. Unlike supervised learning, it doesn't require labeled input/output pairs and doesn't demand immediate corrections for actions that aren't optimal. The problem framework is often expressed using a Markov

decision process. Many reinforcement learning strategies employ dynamic programming methods. However, these algorithms stand apart from traditional dynamic programming techniques as they don't rely on a precise mathematical model of the Markov decision process and are tailored for scenarios where exact methods are unfeasible.

2.3 Dimensionality Reduction & Manifold learning

Dimensionality reduction is a technique used in machine learning and statistics to reduce the number of input variables or features in a dataset without losing essential information. It aims to capture the most important information contained in the data using fewer features, making the data more manageable, reducing computation time, and potentially improving model performance.

Perhaps the most obvious usage of dimensionality reduction is visualization. By reducing dimensions, data that lies in a high dimensional space can be visualized in 2D or 3D spaces, helping in understanding patterns, clusters, or relationships within the data. Beyond visualization, dimensionality reduction enhances storage since less space is needed when data dimensions are reduced. It also optimizes computational performance, as processing lower-dimensional data is faster and allows many algorithms to converge more quickly on simplified data. Furthermore, it can lead to better model performance as keeping the most important information can be equivalent with removing noise or redundant features.

For example, a field that Bioinformatics. The whole human genome consists of around 3 billion base pairs. When we consider sequencing data from thousands of individuals for genome-wide association studies, the datasets become massive. By applying dimensionality reduction techniques, researchers can pinpoint the specific regions of the genome (or specific Single Nucleotide Polymorphisms, SNPs) that are of interest and store only that reduced data. This significantly cuts down on the storage requirements, allowing for efficient use of storage resources and faster data retrieval. Consider a surveillance system that processes video feeds to detect unauthorized intruders. The raw video data is high-dimensional, consisting of pixel values for each frame. If we were to process this high-dimensional data in real-time, it would require immense computational power and might not deliver results quickly enough. By using dimensionality reduction techniques, the system can focus on the essential features of the video feed that pertain to object detection, drastically reducing the computational burden. This enables the system to process video feeds faster, making real-time detection feasible and more responsive.

Numerous dimensionality reduction strategies exist, all stemming from different assumptions and contexts. We're interested in a specific approach that identifies high-dimensional data as potentially being simpler than it seems. Often, a dataset with many features might have several that all relate to a single root cause. This is common, for instance, in videos taken of an object from multiple viewpoints at once. Such datasets might have redundant information, and it would be beneficial to have a more streamlined version that corresponds to the primary elements guiding the data. This concept is structured around the idea of a "manifold", suggesting that data exists on a low-dimensional

surface within a high-dimensional space. This low-dimensional space mirrors the primary elements, while the high-dimensional one represents feature space. The process to discover this structure is called manifold learning.

2.4 Manifolds

A manifold is a fundamental concept in mathematics, particularly in geometry and topology. In intuitive terms, a manifold is a space that is "locally" similar to Euclidean space. That means that if you zoom in closely enough to any point on a manifold, it will look like a flat, n-dimensional space. A manifold is a topological space that, for every point, has a neighborhood that is homeomorphic (i.e., topologically equivalent) to an open subset of Euclidean space R^n for some n.

Consider the curve shown in figure 2.1 Note that the curve is in R^3, yet it has zero volume, and in fact zero area. The extrinsic dimensionality – three – is some-what misleading since the curve can be parameterized by a single variable. One way of formalizing this intuition is via the idea of a manifold: the curve is a one-dimensional manifold because it locally "looks like" a copy of R^1. Another good example of manifold is the earth. A good example of a manifold is the Earth (Figure 2.2). Locally, at each point on the surface of the Earth, we have a 3-D coordinate system: two for location and the last one for the altitude. Globally, it is a 2-D sphere in a 3-D space.

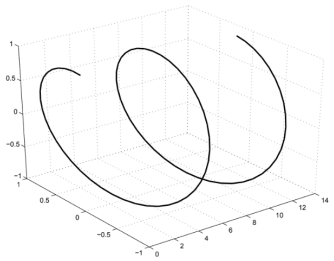


Figure 2.1. *A one-dimensional manifold embedded in three dimensions.*

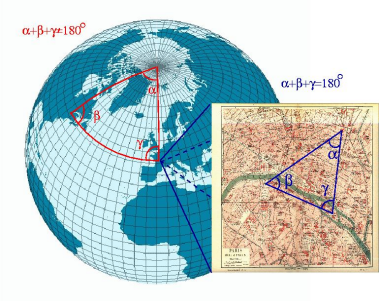


Figure 2.2. *Earth is an example of a manifold.*

2.5 Dimensionality Reduction Algorithms

In the ever-expanding landscape of data-driven research, the burgeoning volume and complexity of high-dimensional datasets pose formidable challenges. As datasets grow in dimensions, so does the intricacy of analysis and interpretation. The field of dimensionality reduction algorithms emerges as a pivotal player in this narrative, seeking to distill the essence of data by transforming it into a more manageable form without sacrificing critical information. Through the lens of these algorithms, the pursuit of meaningful patterns and relationships within vast datasets becomes an attainable endeavor. This thesis navigates the realms of dimensionality reduction, with a primary focus on the venerable Multidimensional Scaling (MDS) technique and its extended variant designed to accommodate non-metric distance matrices. By unraveling the complexities of high-dimensional spaces, this research endeavors to contribute to the foundational understanding of dimensionality reduction and its far-reaching implications in diverse scientific and practical domains.

High-dimensional datasets can be very difficult to visualize. While data in two or three dimensions can be plotted to show the inherent structure of the data, equivalent high-dimensional plots are much less intuitive. To aid visualization of the structure of a dataset, the dimension must be reduced in some way. The simplest way to accomplish this dimensionality reduction is by taking a random projection of the data. Though this allows some degree of visualization of the data structure, the randomness of the choice leaves much to be desired. In a random projection, it is likely that the more interesting structure within the data will be lost.

As explained from [4] a manifold is a topological space that locally resembles a Euclidean space. The purpose of Multidimensional Scaling (MDS) is to infer data representations on a low-dimensional manifold while simultaneously preserving the distances of the high-dimensional data points. When data lies on or close to a linear subspace, low-dimensional representations of data can be obtained using linear dimensionality reduction techniques like Principle Components Analysis (PCA) [5] and classical MDS. In real data applications, such a linearity assumption may be too strong and can lead to meaningless results. Thus a significant effort has been made by the machine learning community to apply manifold learning in non-linear domains. Representative manifold learning algorithms include Isometric Feature Mapping (ISOMAP) [6, 7, 8, 9, 10], Landmark ISOMAP [3], [11],Locally Linear Embedding (LLE) [12, 13, 14, 15, 16], Modified LLE [17] Hessian LLE [18, 19], Semidefinite Embedding [20, 21, 22, 23] Laplacian Eigenmaps (LE) [12, 16, 17], Local Tangent Space Alignment (LTSA) [18],etc.

ISOMAP uses a geodesic distance to measure the geometric information within a manifold. LLE assumes that a manifold can be approximated in a Euclidean space and the reconstruction coefficients of neighbors can be preserved in the low-dimensional space. LE uses an undirected weighted graph to preserve local neighbor relationships. Hessian LLE obtains low- dimensional representations through applying eigenanalysis on a Hessian coefficient matrix. LTSA utilizes local tangent information to represent the manifold geometry and extends this to global coordinates. Finally, SDE attempts to maximize the distance between points that don't belong in a local neighborhood. Also, a common nonlinear method

for dimensionality reduction is the kernel extension of PCA [24]. A wide class of derivative-free algorithms for nonlinear optimization has been studied and analyzed in [25] and [26]. GPS methods consist a subset of the aforementioned algorithms which do not require the explicit computation of the gradient in each iteration-step. Some GPS algorithms are: the original Hooke and Jeeves pattern search algorithm [27], the evolutionary operation by utilizing factorial design [28] and the multi-directional search algorithm [29], [30]. In [31], a unified theoretical formulation of GPS algorithms under a common notation model has been presented as well as an extensive analysis of their global convergence properties. Local convergence properties have been studied later by [32]. Notably, the theoretical framework as well as the convergence properties of GPS methods have been extended in cases with linear constrains [33], boundary constrains [34] and general Lagrangian formulation [35].

2.6 Linear Methods

Linear dimensionality reduction is a technique used in data analysis and machine learning to reduce the dimensionality of a dataset while preserving relevant information and relationships among the data points. It operates under the assumption that the most important patterns in the data can be captured through linear combinations of the original features.

The key characteristics and principles of linear dimensionality reduction are:

- Linearity: Linear dimensionality reduction methods assume that relationships between variables are linear. This means that the new features created during the dimensionality reduction process are linear combinations of the original features.

- Preservation of Variance: Linear dimensionality reduction techniques aim to retain as much of the variance in the data as possible while reducing the dimensionality. High-variance directions in the data are considered to be the most informative.

- Orthogonality: Principal components or linear combinations created by these methods are orthogonal to each other. This means they are uncorrelated, which can simplify further analysis.

- Eigenvalue Decomposition: Many linear dimensionality reduction techniques rely on eigenvalue decomposition of the covariance matrix of the data. This decomposition yields eigenvectors (principal components) and eigenvalues, where the eigenvectors represent the directions in which the data varies the most and the eigenvalues represent the amount of variance explained by each component.

- Feature Ranking: These techniques often involve selecting a subset of the principal components or linear combinations based on their corresponding eigenvalues. Components with higher eigenvalues are considered more important.

2.6.1 SVD

Singular Value Decomposition (SVD) is one of the fundamental techniques in linear algebra and data analysis. It is a powerful matrix factorization technique that decomposes a given matrix into three matrices. These matrices provide insights into the underlying structure and essential characteristics of the original matrix. SVD has applications in numerous fields, including data analysis, image compression, natural language processing, and recommendation systems.

As described in [36] SVD of an $m \times n$ matrix A expresses the matrix as the product of three "simple" matrices:

$$A = U\Sigma V^T = \sum_{i=1}^{r} \sigma_i u_i v_i^T \tag{2.1}$$

where

- A is the original matrix of dimension $m \times n$
- U is an $m \times m$ orthogonal matrix containing the left singular vectors
- V is an $n \times n$ orthogonal matrix containing the right singular vectors
- Σ is an $m \times n$ diagonal matrix with nonnegative entries, and with the diagonal entries sorted from high to low.

The singular values in Σ are non-negative and represent the magnitude or importance of the corresponding singular vectors in U and V^T. They are typically ordered in decreasing order, which allows us to capture the most significant patterns and reduce the dimensionality of the data.

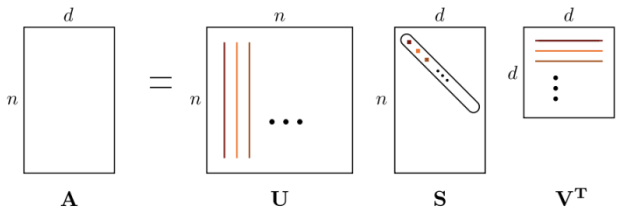


Figure 2.3. *The Singular Value Decomposition*

Consider each row of an $n \times d$ matrix A as a point in d-dimensional space. The SVD finds the best-fitting k-dimensional subspace for $k = 1, 2, 3, \cdots$, for the set of n data points. Here, "best" means minimizing the sum of the squares of the perpendicular distances of the points to the subspace, or equivalently, maximizing the sum of squares of the lengths of the projections of the points onto this subspace.

As described in [37] Let A be an $n \times d$ matrix and think of the rows of A as n points in d-dimensional space. Let

$$A = \sum_{i=1}^{r} \sigma_i u_i v_i^T \tag{2.2}$$

be the SVD of A. For $\{k \in 1, 2, \cdots, r\}$, let

$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T \tag{2.3}$$

be the truncated after k terms. It is clear that A_k has rank k. We show that A_k is the best rank k approximation to A, where, error is measured in the Frobenius norm. Geometrically, this says that $v_1, \cdots, v_k$ define the k-dimensional space minimizing the sum of squared distances of the points to the space. To see why, we need the following lemma.

Lemma *The rows of A_k are the projections of the rows of A onto the subspace V_k spanned by the first k singular vectors of A.*

Proof Let **a** be an arbitrary row vector. Since the v_i are orthonormal, the projection of the vector **a** onto V_k is given by $\sum_{i=1}^{k}(a \cdot v_i)v_i^T$. Thus, the matrix whose rows are the projections of the rows of A onto V_k is given by $\sum_{i=1}^{k} Av_i v_i^T$. This last expression simplifies to

$$\sum_{i=1}^{k} Av_i v_i^T = \sum_{i=1}^{k} \sigma_i u_i v_i^T = A_k \tag{2.4}$$

1. Dimensionality Reduction: By truncating the singular values and their corresponding singular vectors, SVD can be used to reduce the dimensionality of data while retaining the most important information. This is particularly useful in data compression and feature selection.

2. Image Compression: In image processing, SVD is used to compress images efficiently. By retaining only the most significant singular values and their associated vectors, one can reconstruct images with minimal loss of quality.

3. Collaborative Filtering: In recommendation systems, SVD is applied to user-item matrices to uncover latent factors that explain user preferences and item characteristics. This information is then used to make personalized recommendations.

4. Natural Language Processing: SVD plays a crucial role in techniques like Latent Semantic Analysis (LSA), which is used for text and document analysis. It helps discover the underlying semantic structure of textual data.

5. Principal Component Analysis (PCA): SVD is at the core of PCA, a popular technique for data analysis and visualization. It helps find the principal components that capture the most variation in high-dimensional data.

Solving SVD numerically can be computationally intensive, especially for large matrices. Various algorithms, such as the Golub-Reinsch algorithm and randomized SVD, have been developed to make the computation more efficient. These methods are essential for practical applications of SVD.

Singular Value Decomposition is a versatile mathematical tool with a wide range of applications in data analysis, signal processing, and machine learning. Understanding its principles and applications is invaluable for researchers and practitioners seeking to gain insights from data, reduce dimensionality, and make more informed decisions.

2.6.2 PCA

PCA stands for Principal Component Analysis, and it is a widely used linear dimensionality reduction technique in data analysis and machine learning. Its primary purpose is to reduce the dimensionality of a dataset while preserving as much of the relevant information as possible. PCA achieves this by transforming the original features into a new set of uncorrelated variables called principal components.

The definition of the method is, for a given data set and parameter k, to compute the k-dimensional subspace(through the origin) that minimizes the average squared distance between the points and the subspace, or equivalently that maximizes the variance of the projections of the data points onto the subspace, as in shown in the Figure 3.1.

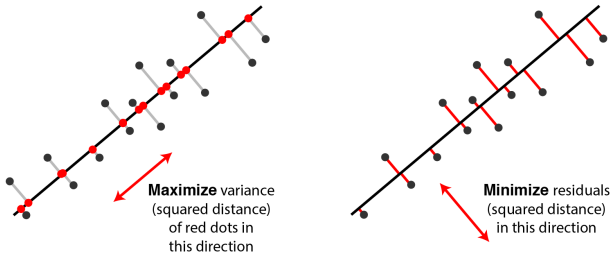


Figure 2.4. *PCA minimizes the sum of squared perpendicular distances or maximizes the variance of the projections of the data*

As described in [38] the goal of PCA is to approximately express each of m n-dimensional vectors $x_1, ..., x_m \in \mathbb{R}^n$ as linear combinations of k-dimensional vectors $v_1, ..., v_k \in \mathbb{R}^n$ so that

$$x_i \approx \sum_{j=1}^{k} a_{ij} v_j \tag{2.5}$$

The vectors $v_1, ..., v_k \in \mathbb{R}^n$ are computed in order to maximize the sum of squared projection length.

$$\frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{k} < x_i, v_j >^2 \tag{2.6}$$

The resulting k orthonormal vectors are called the top k principal components of the data. Before using PCA, it's important to pre-process the data.

- Data Centering: PCA starts by centering the data, which means subtracting the mean of each feature from the data points $x_1, \cdots, x_m$. This is easy to enforce by subtracting (i.e., shifting) each point by the "sample mean" $\overline{x} = \frac{1}{m} \sum_{i=1}^{m} x_i$. This ensures that the new coordinate system is centered at the origin.

Covariance Matrix: Next, PCA computes the covariance matrix of the centered data. The covariance matrix describes how the features in the dataset vary together. It contains information about the relationships between the features.

Eigenvalue Decomposition: PCA then performs an eigenvalue decomposition of the covariance matrix. This decomposition results in a set of eigenvalues and corresponding eigenvectors. The eigenvectors represent the directions (principal components) along which the data varies the most, and the eigenvalues represent the amount of variance explained by each principal component.

Selecting Principal Components: To reduce dimensionality, you can choose a subset of the principal components that explains a sufficiently high percentage of the total variance. Typically, you rank the eigenvalues in decreasing order, and the top 'k' eigenvectors (principal components) are selected.

Projection: Finally, the original data is projected onto the selected principal components to obtain the lower-dimensional representation. This transformation results in a new dataset where each data point is described using only the selected principal components.

PCA is often used for the following purposes:

Dimensionality Reduction: By selecting a smaller number of principal components, you can reduce the dimensionality of your data while retaining most of the information.

Data Visualization: PCA can be used to visualize high-dimensional data in two or three dimensions by plotting data points along the first few principal components. In order to do so, the top few principal components (k is 1,2 or 3) can be used as a new coordinate axis and plot the projections of all of the data points in this new coordinate system.

Noise Reduction: In some cases, PCA can help in reducing noise and improving the signal-to-noise ratio in data.

Feature Engineering: PCA can be used as a feature engineering technique to create new features that capture the most important patterns in the data.

PCA technique is a linear technique, which means it's most effective at capturing linear relationships between variables. Non-linear relationships may not be well-preserved by PCA, and for such cases, non-linear dimensionality reduction techniques like t-SNE or Isomap may be more appropriate.

Lastly, the PCA has some weaknesses. It can get tricked by high-variance noise, and the orthogonality constraints on the principal components mean that principal components after the first few can be difficult to interpret. Lastly, as PCA is a linear technique, it fails to discover nonlinear structure. For such cases non-linear dimensionality reduction techniques like Isomap may be more appropriate. Some non-linear algorithms will be explained in the next chapters.

2.7 Non Linear Algorithms - Manifold Learning Technique

Non-linear algorithms in ML are designed to handle data that doesn't adhere to a linear relationship, meaning the relationship between input and output is more intricate and multifaceted. Unlike linear algorithms, which might plot a straight line when trying to make predictions, non-linear algorithms can curve, twist, and turn to adjust to more complex patterns in the data.

The algorithms that try to solve the problem of nonlinear dimensionality reduction have many approaches, some of them are local - they present the local geometry of the data

(nearby points to nearby points), and some of them are global- they preserve geometry at all scales(nearby points to nearby points and faraway points to faraway points). Global approaches give more faithful representation of the data's global structure, but they are not as computational efficient as local approaches.

2.7.1 Isometric Feauture Mapping (Isomap)

Isomap was one of the first algorithms introduced for manifold learning. The algorithm is perhaps the best known and most applied among the multitude of procedures now available for the problem at hand. It may be viewed as an extension to Multidimensional Scaling (MDS), a classical method for embedding dissimilarity information into Euclidean space, that will be explained extensively in the next Chapter. Isomap has a global approach of solving the problem of nonlinear dimensionality reduction and it consists of three stages as descripted in [8]:

1. Determine a neighbourhood graph G of the observed data x_i in a suitable way. For example, G might contain $x_i x_j$ iff x_j is one of the k nearest neighbours of x_i (and vice versa). Alternatively, G might contain the edge $x_i x_j$ iff $|x_i - x_j| < e$, for some e.

2. Estimate the geodesic distances by computing shortest paths in the graph for all pairs of data points. Each edge $x_i x_j$ in the graph is weighted by its Euclidean length $|x_i - x_j|$, or by some other useful metric. The shortest path can be computed via a shortest-path al- gorithm (Dijkstra's or Floyd's).

3. Apply MDS to the resulting shortest-path distance matrix D to find a new embedding of the data in Euclidean space.

The problem that Isomap tries to addresses is as follows: when provided with a matrix $D \in \mathbb{R}^{N\times}$ representing dissimilarities, the goal is to create a group of points whose mutual Euclidean distances align closely with those in D. Multiple cost functions exist for this purpose, along with several algorithms to reduce these costs. Our attention is centered on classical MDS (cMDS) because it's the technique employed by Isomap. Classical MDS is extensively explained in Chapter 4.

2.7.2 Isomap with landmark points

In the paper [2], two main computational challenges associated with the Isomap algorithm are highlighted. The initial challenge revolves around the calculation of the $N \times N$ shortest-paths distance matrix, denoted D_N. Although Floyd's method results in a complexity of $\mathcal{O}(N^3)$, adopting Dijkstra's algorithm with Fibonacci heaps can refine it to $\mathcal{O}(kN^2 \log N)$, where k stands for the neighbourhood size. The subsequent challenge is the MDS eigenvalue computation which deals with a full $N \times N$ matrix, bringing along a complexity of $\mathcal{O}(N^3)$.

L-Isomap offers solutions to both these computational constraints. Some of the the data points are defined as landmark points, where $n << N$. Rather than determining D_n,

the algorithm computes the $n \times N$ matrix, $D_{n,N}$, representing distances from every data point exclusively to the landmark points. Using a new procedure LMDS (Landmark MDS), we find a Euclidean embedding of the data using $D_{n,N}$ instead of D_N . This leads to an enormous savings when is much less than , since $D_{n,N}$ can be computed using Dijkstra in $\mathcal{O}(knN \log N)$ time, and LMDS runs in $\mathcal{O}(n^2 N)$.

LMDS is feasible precisely because we expect the data to have a low-dimensional embedding. The initial step involves applying classical MDS exclusively to the landmark points, allowing their accurate embedding in $\mathbb{R}^{<}$. Subsequent data points are then positioned in $\mathbb{R}^{<}$, leveraging their known distances from the landmark points as guiding constraints. This approach mirrors the methodology used in the Global Positioning System, where a limited set of distance measurements are utilized to pinpoint a specific geographical position. As long as $n/geql + 1$ and the landmarks maintain a general position, sufficient constraints are available to distinctly determine x. The selection of landmark points can be random, with the stipulation that the number of landmarks should exceed the minimum of $l + 1$ to guarantee stability.

2.7.3 LLE

Locally Linear Embedding (LLE) is a dimensionality reduction technique and is an unsupervised algorithm that aims to preserve the local linear structures within the data.

LLE is based on the assumption that the relationships between data points are locally linear. In other words, for each data point, there exists a set of its nearest neighbors for which it can be linearly represented as a weighted sum of those neighbors.

- Step 1: Constructing the Neighborhood Graph: LLE begins by constructing a neighborhood graph, where each data point is connected to its k nearest neighbors. The choice of k is a hyperparameter that determines the local neighborhood size. Common distance metrics, such as Euclidean distance, can be used to define proximity.
- Step 2: Weight Matrix Calculation: For each data point, LLE finds the optimal weights that best approximate it as a linear combination of its neighbors. These weights are computed in such a way that minimizes the reconstruction error. This can be formulated as a minimization problem with constraints. The weights are typically found by solving a system of linear equations.
- Step 3: Constructing the Lower-Dimensional Embedding: After determining the weights for each data point, LLE creates a lower-dimensional representation of the data by minimizing the reconstruction error. This is done by finding a set of coordinates for each data point in the lower-dimensional space that preserves the pairwise linear relationships between points in the high-dimensional space. The lower-dimensional coordinates are optimized using techniques like eigenvalue decomposition.

Locally Linear Embedding (LLE) stands out as a valuable tool for dimensionality reduction and data visualization, particularly in scenarios characterized by nonlinear data

structures. Its versatility extends across various domains, including image processing, computer vision, and biology, providing a robust means to analyze high-dimensional datasets and unveil meaningful patterns.

The effectiveness of Locally Linear Smoothing (LLS) lies in its adeptness at capturing the intrinsic structure of data, rendering it invaluable for tasks such as manifold learning and the preservation of local distances. Its capability to handle complex, nonlinear relationships between data points further enhances its utility.

However, it is essential to acknowledge certain limitations. LLE's performance can exhibit sensitivity to the choice of the neighborhood size (k), necessitating meticulous tuning for optimal results. Additionally, challenges may arise when dealing with data featuring gaps or disconnected components, as LLE heavily relies on local neighborhood relationships for its computations. These considerations highlight the need for thoughtful parameter selection and the recognition of specific data characteristics to maximize the efficacy of LLE in diverse applications. LLE is summarized by [39]

- Compute the k nearest neighbors of each point X_i.
- Compute the weights W_{ij} of a convex combination of the k nearest neighbors that best represent the point X_i.
- Find a low-dimensional projection Y_i such that the above local representations are best preserved.

Chapter 3

Multidimensional Scaling (MDS)

3.1 Classical MDS

The concept of Classical Multidimensional Scaling (MDS) was initially introduced by Torgerson in [40]. MDS can be formally defined as follows: Given the matrix $\mathbf{\Delta}$ containing pairwise distances or dissimilarities $\{\delta_{ij}\}_{1\leq i,j\leq N}$ between N points in a high-dimensional space, the objective is to determine a set of points $\{\mathbf{x}_i\}_{i=1}^{N}$ that reside on a manifold $\mathcal{M} \in \mathbb{R}^L$ while preserving the given dissimilarities $\{\delta_{ij}\}_{1\leq i,j\leq N}$ as faithfully as possible. Here, $\mathbf{X}^T \in \mathbb{R}^{L\times N}$ represents the data array containing all points $\mathbf{x}_i \in \mathbb{R}^L,\ 1 \leq i \leq N$ as its columns.

The ultimate goal is to obtain an embedding dimension L that is as small as possible in order to reduce the dimensionality of the resulting manifold $\mathcal{M}$ while minimizing deviation from the given dissimilarities δ_{ij}. Typically, Euclidean distances are utilized as $d_{ij}(\mathbf{X}) = ||\mathbf{x}_i - \mathbf{x}_j||_2 = \sqrt{\sum_{k=1}^{L}(x_{ik} - x_{jk})^2}$ in the embedded space $\mathbb{R}^L$.

Classical MDS incorporates a centering matrix $\mathbf{H} = \mathbf{I}_N - \frac{1}{N}\mathbf{1}_N^T\mathbf{1}_N$, which effectively subtracts the mean of the columns and rows for each element. Here, $\mathbf{1}_N = [1, 1, ..., 1]$ is a vector of ones in $\mathbb{R}^{\mathbb{N}}$ space. Applying double centering to the Hadamard product of the given dissimilarities yields the construction of the Gram matrix $\mathbf{B}$ as follows:

$$\mathbf{B} = -\frac{1}{2}\mathbf{H}^T(\Delta \odot \Delta)\mathbf{H} \tag{3.1}$$

As Chapter 12 in [41] illustrates, Classical MDS aims to minimize the Strain algebraic criterion, denoted as:

$$||\mathbf{X}\mathbf{X}^T - \mathbf{B}||_F^2 \tag{3.2}$$

The matrix $\mathbf{B}$ can be factorized as $\mathbf{B} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T$ where $\mathbf{\Lambda}$ is a symmetric matrix.The embedded points in $\mathbb{R}^L$ are given by the first L positive eigenvalues of $\mathbf{\Lambda}$, namely $\mathbf{X} = \mathbf{V}_L$. As mentioned in [42], it is worth noting that the solution to Classical MDS yields the same results as Principal Component Analysis (PCA) when PCA is applied in the high-dimensional space.

Originally, Classical MDS was designed for dissimilarity matrices Δ that can be closely approximated in a low-dimensional Euclidean space, as is considered in this study. Nevertheless, matrices that correspond to various spaces, such as Euclidean sub-spaces [43],

Poincare disks [44], and constant-curvature Riemannian spaces [45], have also been subject to investigation.

3.2 Metric MDS

Metric MDS describes contains Classical MDS. Shepard has introduced heuristic methods to enable transformations of the given dissimilarities δ_{ij} [46], [47] but did not provide any loss function in order to model them [48]. Kruskal in [49] and [50] formalized the metric MDS as a least squares optimization problem of minimizing the non-convex Stress-1 function defined below:

$$\sigma_1(\mathbf{X}, \hat{\mathbf{D}}) = \sqrt{\frac{\sum_{i=1}^{N}\sum_{j=1}^{N}(\hat{d_{ij}} - d_{ij}(\mathbf{X}))}{\sum_{i=1}^{N}\sum_{j=1}^{N} d_{ij}^2(\mathbf{X})}} \tag{3.3}$$

where matrix $\hat{\mathbf{D}}$ with elements $\hat{d_{ij}}$ represents all the pairs of the transformed dissimilarities δ_{ij} that are used to fit the embedded distance pairs $d_{ij}(\mathbf{X})$.

A weighted MDS raw Stress function is defined as:

$$\sigma_{raw}^2(\mathbf{X}, \hat{\mathbf{D}}) = \sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}(\hat{d_{ij}} - d_{ij}(\mathbf{X}))^2 \tag{3.4}$$

where the weights w_{ij} are restricted to be non-negative; for missing data the weights are set equal to zero. In our work, we consider always $w_{ij} = 1, \forall 1 \leq i, j \leq N$ where we assume an equal contribution to the Metric-MDS solution for all the elements.

3.3 Non-metric MDS

In non-metric MDS, rather than using a distance metric, $d_y(x_i, x_j)$, for the distances between point sin the embedding space, we use $f(d_y(x_i, x_j))$ where $f(.)$, is a non-parametric monotonic function. In other words, only the order of dissimilarities is important rathen than the amount of dissimilarities.

$d_y(x_i, x_j) \leq d_y(y_k, y_l)$, for the distances between points in the embedding space, we use f(dy(xi , xj)) where f(.) is a nonparametric monotonic function. In other words, only the order of dissimilarities is important rather than the amount of dissimilarities (Agarwal et al., 2007; Jung, 2013):

3.4 Pattern Search MDS

In this section, we provide an overview of nonlinear manifold learning using derivative-free optimization techniques. Specifically, the Classical MDS algorithm is extended in a way that no gradient descent is applied. Instead, the points are evaluated in the embedded space by considering possible "moves" within a fixed-radius sphere. We can establish a fixed-point convergence guarantee by formulating the proposed algorithm as an instance of

the General Pattern Search (GPS) framework. Additionally, we illustrate the significance of Landmark points, which can greatly enhance algorithm complexity and yield impressive results. The contribution of the thesis focuses on the extension of the MDS algorithm with the inclusion of these Landmark points, and we support our findings with experimental results.

3.4.1 General Pattern Search (GPS) methods

A broad range of derivative-free optimization techniques for nonlinear problems has been extensively investigated and discussed in [51] and [52]. Within this category, there exists a subset known as GPS methods, which distinguish themselves by not requiring the explicit calculation of gradients during each iteration. Notable GPS algorithms include the original Hooke and Jeeves direct coordinate search approach [53], an approach involving evolutionary operations through factorial design [54] and the multi-directional search algorithm [55], [56]. Furthermore, in [57], a comprehensive theoretical framework unifying GPS algorithms using a standardized notation model was introduced, accompanied by an extensive analysis of their overall convergence properties.

Next we present a short description of iterative GPS minimization as described in [4]. First we have to define the following components:

- A basis matrix that could be any nonsingular matrix $B \in \mathbb{R}^{n\times n}$
- A matrix $C^{(k)}$ for generating all the possible moves for the kth iteration of the minimization algorithm

$$C^{(k)} = [\ M^{(k)} - M^{(k)}L^{(k)}]\ = [\ \Gamma^{(k)}L^{(k)}] \tag{3.5}$$

  where the columns of $M^{(k)} \in \mathbb{Z}^{n\times n}$ form a positive span of $\mathbb{R}^n$ and $L^{(k)}$ contains at least the zero column of the search space $\mathbb{R}^n$.

- A pattern matrix $P^{(k)}$ defined as

$$P^{(k)} = BC^{(k)} = [\ BM^{(k)} - BM^{(k)}BL^{(k)}] \tag{3.6}$$

  where the submatrix $BM^{(k)}$ forms a basis $\mathbb{R}^n$.

In each iteration k, the set of steps $\{\mathbf{s}_i^{(k)}\}_{i=1}^m$ are generated by the pattern matrix $\mathbf{P}^{(k)}$:

$$\mathbf{s}_i^{(k)} = \Delta^{(k)}\mathbf{p}_i^{(k)},\ \ \mathbf{P}^{(k)} = [\mathbf{p}_1^{(k)}, ..., \mathbf{p}_m^{(k)}] \in \mathbb{R}^{n\times m} \tag{3.7}$$

where $\mathbf{p}_i^{(k)}$ is the ith column of defines the direction of the new step, while $\Delta^{(k)}$ configures the length towards this direction. If pattern matrix $\mathbf{P}^{(k)}$ contains m columns, then $m \geq n+1$ in order to positively span the search space $\mathbb{R}^n$. A new trial point of GPS algorithm would be $\mathbf{x}_i^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{s}_i^{(k)}$ where we evaluate the value of the function f which we seek to minimize. A successful step would mean a further minimization of the objective function would mean f, i.e., $f(\mathbf{x}^{(k)}+\mathbf{s}_i^{(k)}) > f(\mathbf{x}^{(k+1)})$. A pseudo-code for all GPS methods is presented in Algorithm 3.1.

ALGORITHM 3.1: *General Pattern Search (GPS)*

1: **procedure** GPS_SOLVER($\mathbf{x}^{(0)}$, $\Delta^{(0)}$, $\mathbf{C}^{(0)}$, $\mathbf{B}$)
2: $k = -1$
3: **while**
4: **do**$k = k + 1$
5: $\mathbf{s}^{(k)}$ = EXPLORE_MOVES($\mathbf{BC}^{(k)}$, $\mathbf{x}^{(k)}$, $\Delta^{(k)}$)
6: $\rho^{(k)} = f(\mathbf{x}^{(k)} + \mathbf{s}^{(k)}) - f(\mathbf{x}^{(k)})$
7: **if** $\rho^{(k)} < 0$ **then**
8: $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{s}^{(k)}$ ▷ Successful iteration
9: **else**
10: $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)}$ ▷ Unsuccessful iteration
11: **end if**
12: $\Delta^{(k+1)}, \mathbf{C}^{(k+1)}$ = UPDATE($\mathbf{C}^{(k)}$, $\Delta^{(k)}$, $\rho^{(k)}$)
13: **end while**convergence criterion is not met
14: **end procedure**

Initially, we select $\mathbf{x}^{(0)} \in \mathbb{R}^n$ and a positive step length parameter $\Delta^{(0)} > 0$. In each iteration k, we explore a set of possible steps defined by the `EXPLORE_MOVES()` subroutine at line 5 of the algorithm. Pattern search methods mainly differ on the heuristics used for the selection of exploratory moves that they evaluate the function f on. If a new exploratory point lowers the value of the function f, iteration k is considered successful and the starting point of the next iteration is updated $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{s}^{(k)}$. Otherwise, if at a certain iteration we cannot obtain any successful step then the algorithm can produce the zero-step point. The step length parameter $\Delta^{(k)}$ is modified by the `UPDATE()` subroutine. For successful iterations, i.e., $\rho^{(k)} < 0$, the step length is forced to increase in a way that can be described as follows:

$$\begin{aligned} &\Delta^{(k+1)} = \Lambda^{(k)}\Delta^{(k)}, \;\; \Lambda^{(k)} \in \Lambda = \{\tau^{w_1}, ..., \tau^{w_{|\Lambda|}}\} \\ &\tau > 1, \;\; \{w_1, ..., w_{|\Lambda|}\} \subset \mathbb{N}, \;\; |\Lambda| < +\infty \end{aligned} \tag{3.8}$$

where τ and w_i are predefined constants that are used for the ith successive successful iteration. For unsuccessful iterations the step length parameter is decreased, i.e., $\Delta^{(k+1)} \leq \Delta^{(k)}$ as follows:

$$\Delta^{(k+1)} = \theta\Delta^{(k)}, \;\; \theta = \tau^{w_0}, \;\; \tau > 1, \;\; w_0 < 0, \tag{3.9}$$

where τ and the negative integer w_0 determine the fixed ratio of step reduction. Generating matrix $\mathbf{C}^{(k+1)}$ could be also updated for unsuccessful/successful iterations in order to contain more/less search directions, respectively.

3.4.2 GPS Convergence

Important convergence properties have been shown in [57, 58, 59, 33, 60] for any GPS method that can be described by the previously defined framework. -[Weak Hypothesis on Exploratory Moves] The subroutine `EXPLORE_MOVES()` as defined in Algorithm 3.1, line 5 guarantees the following:

- The exploratory step direction for iteration k is selected from the columns of the pattern matrix $\mathbf{P}^{(k)}$ as defined in Equation 3.7 and the exploratory step length is $\Delta^{(k)}$ as defined in Equations 3.8, 3.9.

- If among the exploratory moves $\mathbf{a}^{(k)}$ at iteration k selected from the columns of the matrix $\Delta^{(k)}\mathbf{B}[\mathbf{M}^{(k)}-\mathbf{M}^{(k)}]$ exist at least one move that leads to a successful iteration, i.e., $f(\mathbf{x}^{(k)}+\mathbf{a}) < f(\mathbf{x}^{(k)})$, then the `EXPLORE_MOVES()` subroutine will return a move $\mathbf{s}^{(k)}$ such that $f(\mathbf{x}^{(k)}+\mathbf{s}^{(k)}) < f(\mathbf{x}^{(k)})$.

Hypothesis 3.4.2 enforces some mild constraints on how the exploratory moves would be produced by Algorithm 3.1, line 5. Essentially, the suggested step $\mathbf{s}^{(k)}$ must be derived from the pattern matrix $\mathbf{P}^{(k)}$. Moreover, the algorithm needs to provide a simple decrease for the objective function f at every step. Specifically, the only way to accept an unsuccessful iteration would be if none of the steps from the columns of the matrix $\Delta^{(k)}\mathbf{B}[\mathbf{M}^{(k)}-\mathbf{M}^{(k)}]$ lead to a decrease of the objective function f.

Based on the aforementioned Hypothesis, a GPS method can enjoy some theoretical convergence guarantees which are stated rigorously in Theorem. 1 as follows:

Theorem 1. *Let $L(\mathbf{x}^{(0)}) = \{\mathbf{x} : f(\mathbf{x}) \leq f(\mathbf{x}^{(0)})\}$ be closed and bounded and f continuously differentiable on a neighborhood of $L(\mathbf{x}^{(0)})$, namely on the union of the open balls $\bigcup_{\mathbf{a}\in L(\mathbf{x}^{(0)})} B(\mathbf{a},\eta)$ where $\eta > 0$. If a GPS method is formulated as described in Section 3.4.4 and Hypothesis 3.4.2 holds then for the sequence of iterations $\{\mathbf{x}^{(k)}\}$ produced by Algorithm 3.1*

$$\lim_{k\to+\infty} \inf \ ||\nabla f(\mathbf{x}^{(k)})|| = 0$$

For the proof of this Theorem we refer the reader to [57].

However, as shown in [61] one can construct a continuously differentiable objective function and a GPS method with infinite many limit points with non-zero gradients and thus even Theorem. 1 holds, the convergence of $||\nabla f(\mathbf{x}^{(k)})||$ is not assured.

3.4.3 The PSMDS algorithm

As introduced by [1], the primary concept underlying the PSMDS algorithm is to approach Multidimensional Scaling (MDS) as a problem that doesn't rely on derivatives. Instead, it employs a modified version of general pattern search optimization to minimize a loss function. The PSMDS algorithm takes as input an $N \times N$ target dissimilarity matrix denoted as T and the desired target dimension L for the embedding space. An overview of the algorithm is presented in Algorithm 3.2.

The algorithm's initialization process includes the following steps:

- Generate N random points in the embedded space with the final -desired L dimensions. $X^{(0)} = [x_1^{(0)}, x_2^{(0)}, ..., x_N^{(0)}] \in \mathbb{R}^{N\times L}$ $\mathbf{X} = [x_1, x_2, ..., x_N] \in \mathbb{R}^{N\times L}$

- Compute the distance matrix between points. The element $d_{ij}^{(0)}$ is the Euclidean distance between vectors $x_i^{(0)}$ and $x_j^{(0)}$ of $X^{(0)}$

- Calculate the initial approximation error $e^{(0)} = ||\mathbf{\Delta} - \mathbf{D}||_F^2$, where e represents the element-wise mean squared error (MSE) between the two matrices. The objective function which is optimized for all PSMDS is shown below and is the stress function for MDS:

$$f(\mathbf{\Delta}, \mathbf{D}) = ||\mathbf{\Delta} - \mathbf{D}||_F^2 = \sum_{i=1}^{N}\sum_{j=1}^{N}(\delta_{ij} - d_{ij})^2, \quad \text{where } \mathbf{\Delta}, \mathbf{D} \in \mathbb{R}^{N\times N} \tag{3.10}$$

Algorithm 3.2: *Pattern Search MDS*

procedure PSMDS($\mathbf{\Delta}$, L, $r^{(0)}$)
  $k \leftarrow 0$ ▷ k is the number of epochs
  $\mathbf{X}^{(k)} \leftarrow$ UNIFORM($N \times L$)
  $\mathbf{D}^{(k)} \leftarrow$ DISTANCE_MATRIX($\mathbf{X}^{(k)}$)
  $e^{(k)} \leftarrow ||\mathbf{\Delta} - \mathbf{D}||_F^2$
  $e^{(k-1)} \leftarrow +\infty$
  $r^{(k)} \leftarrow r^{(0)}$
  while $r^{(k)} > \delta$ **do**
    if $e^{(k-1)} - e^{(k)} \leq \epsilon \cdot e^{(k)}$ **then**
      $r^{(k)} \leftarrow \frac{r^{(k)}}{2}$
    end if
    $\mathcal{S} \leftarrow$ SEARCH_COORDINATES($r^{(k)}$, L)
    for all $x \in \mathbf{X}^{(k)}$ **do**
      $\mathbf{X}^*, e^* \leftarrow$ OPTIMAL_MOVE($\mathbf{D}^{(k)}$, $\mathbf{X}^{(k)}$, x, $\mathcal{S}$, $e^{(k)}$)
      $e^{(k-1)} \leftarrow e^{(k)}$
      $e^{(k)} \leftarrow e^*$
      $\mathbf{X}^{(k)} \leftarrow \mathbf{X}^*$
      $\mathbf{D}^{(k)} \leftarrow$ UPDATE_DISTANCE_MATRIX ($\mathbf{X}^*$)
    end for
    $k = k + 1$
  end while
end procedure

Following the initialization steps, in each epoch (iteration), we consider the surface of a hypersphere of radius r around each point $x_i^{(k)}$. The possible search directions lie on the surface of a hypersphere along the orthogonal basis of the space, e.g., in the case of 3-dimensional space along the directions $\pm x, \pm y, \pm z$ on the sphere shown in Fig. 1. This creates the search directions matrix S and is summarized in Alg. 3.3

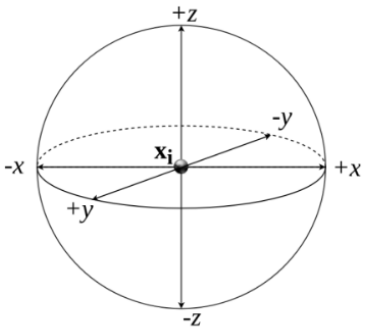


Figure 3.1. *Sphere of radius r around point* $x_i^{(k)}$ *and possible search directions*

Algorithm 3.3: *Define search directions*

```
1: function SEARCH_COORDINATES(r, L)
2:     S+ ← r · I_L
3:     S− ← −r · I_L
4:     S ← [ S+/S− ]
5:     return S
6: end function
```

Each point is moved greedily along the dimension that produces the minimum error. At this stage we only consider moves that yield a monotonic decrease in the error function. Alg. 3.4 finds the optimal move that minimizes $e(k) = f(T, D(k))$ for each new point x and moves X in that direction. Note that when writing s ∈ S, the matrix S is considered to be a set of row vectors.

Algorithm 3.4: *Find optimal move for a point*

```
1: function OPTIMAL_MOVE(D^(k), X^(k), x, S, e^(k))
2:     X̃ ← X^(k)
3:     e* ← e^(k)
4:     for all s ∈ S do
5:         X̃ ← X(x) + s                ▷ Update x row of X̃ with the new step
6:         D̃ ← DISTANCE_MATRIX(X̃)
7:         ẽ ← ||T − D̃||²_F
8:         if ẽ < e* then               ▷ If step s produces lower error then update
9:             X* ← X̃
10:            e* ← ẽ
11:        end if
12:    end for
13:    return X*, e*
14: end function
```

The resulting error e is computed after performing the optimal move for each point in X(k). If the error decrease hits a plateau, we halve the search radius and proceed to the next epoch. This is expressed as e(k)e < e (k), where is a small positive constant, namely

the error decrease becomes very small in relation to e (k). The process stops when the search radius r becomes very small, namely r $<$ δ, where δ is a small constant, as shown in Alg. 3.2.

3.4.4 GPS Formulation of PSMDS

PSMDS can be expressed by using the unified GPS formulation introduced in Section 3.4.2. Next, we express our proposed algorithm and associated objective function under this formalism.

To begin with, the problem of MDS is restated in a vectorized form which spans all the possible coordinates considered for all points. We use matrix $\mathbf{\Delta}$ with elements $\{\delta_{ij}\}_{1\leq i,j\leq N}$ as the dissimilarities between N points in the high dimensional space. The set of points $\{\mathbf{x}_i\}_{i=1}^{N}$ lie on the low dimensional manifold $\mathcal{M} \in \mathbb{R}^L$ and form the column set of matrix $\mathbf{X}^T$. The embedded data matrix $\mathbf{X} \in \mathbb{R}^{N\times L}$ will be now vectorized as an one column vector as shown next:

$$\begin{aligned} \mathbf{x}_i &= [x_{i1}, ..., x_{iL}]^T \in \mathbb{R}^L, 1 \leq i \leq N \\ \mathbf{z} &= vec(\mathbf{X}^T) = [x_{11}, ..., x_{1L}, ..., x_{N1}, ..., x_{NL}]^T \end{aligned} \tag{3.11}$$

Now our new variable $\mathbf{z}$ lies in the search space $\mathbb{R}^{N\cdot L}$. The distance between any two points $\mathbf{x}_i$ and $\mathbf{x}_j$ of the manifold $\mathcal{M}$ remains the same but is now expressed as a function of the vectorized variable $\mathbf{z}$. Namely, $d_{ij}(\mathbf{X}) = ||\mathbf{x}_i - \mathbf{x}_j|| = \sqrt{\sum_{k=1}^{L}(x_{ik} - x_{jk})^2} = d_{ij}(\mathbf{z})$. The equivalent objective function to minimize g is the MSE between the given dissimilarities δ_{ij} and the euclidean distances d_{ij} in the low dimensional manifold $\mathcal{M}$ as defined next:

$$g(\mathbf{z}) = ||\mathbf{D}_\mathbf{z} - \mathbf{\Delta}||_F^2 = \sum_{i=1}^{N}\sum_{j=1}^{N}(d_{ij}(\mathbf{z}) - \delta_{ij})^2, \;\; \mathbf{z} \in \mathbb{R}^{N\cdot L} \tag{3.12}$$

Consequently, the initial MDS is now expressed as an unconstrained non-convex optimization problem which that tries to mimize the objective function g over the search space of $\mathbb{R}^{N\cdot L}$ (Equation 3.13). Thus, the degrees of freedom for our solution are $L{\cdot}N$ corresponding to the L coordinates for all N points on the manifold $\mathcal{M}$.

$$\mathbf{z}^* = \min_{\mathbf{z}\in\mathbb{R}^{N\cdot L}} g(\mathbf{z}) \tag{3.13}$$

CSMDS is now expressed in an equivalent way by utilizing the auxiliary variable $\mathbf{z}$. Next, we match each epoch of our initial algorithm with an iteration of a GPS method. Therefore, the steps which are produced by CSMDS would form a sequence of points $\{\mathbf{z}^{(k)}\}$ in the new search space $\mathbb{R}^{N\cdot L}$. Moreover, we define the matrices $\mathbf{B}, \mathbf{C}^{(k)}, \mathbf{P}^{(k)}$ for the general framework of CSMDS in an equivalent way to Equations 3.5, 3.6. The choice of our basis matrix $\mathbf{B}$ is the corresponding identity matrix of our search space as shown in Equation 3.15.

$$\mathbf{e}_i = [0, .., \underbrace{1}_{index\;\; i}, ..., 0]^T, 1 \leq i \leq N \cdot L \tag{3.14}$$

$$\mathbf{B} = \mathbf{I}_{N \cdot L} = [\mathbf{e}_1, ..., \mathbf{e}_{N \cdot L}] \tag{3.15}$$

While the identity matrix is non singular and its columns span positively the search space $\mathbf{R}^{NL}$, we also define $\mathbf{M}^{(k)}$ as the identity matrix. In Eq. 3.16 matrix $\Gamma^{(k)}$) represents the movement alongside the unit coordinate vectors of $\mathbf{R}^{NL}$. Nevertheless, our generating matrix $\tilde{\mathbf{C}}$ also comprises of all the remaining possible directions which are generated by the set {-1, 0, 1}. In total, we have $3^{NL} - 2NL$ extra direction vectors inside the corresponding matrix $\mathbf{L}^{(\tilde{\mathbf{k}})}$ as it is shown in Eq. 3.17.

$$\begin{aligned} \tilde{\mathbf{M}}^{(\mathbf{k})} &= \tilde{\mathbf{M}} = \mathbf{I}_{NL} \in \mathbf{Z}^{NL \times NL} \\ \boldsymbol{\Gamma}^{(\tilde{\mathbf{k}})} &= \tilde{\boldsymbol{\Gamma}} = [\tilde{\mathbf{M}} - \tilde{\mathbf{M}}] \\ \tilde{S} &= \{-1, 0, 1\} \end{aligned} \tag{3.16}$$

$$\begin{aligned} &\mathbf{L}^{(\mathbf{k})} = \tilde{\mathbf{L}} \\ &\tilde{L} = \tilde{u} : \tilde{u} \in \underbrace{\tilde{S} \times \cdots \times \tilde{S}}_{NL} \wedge \tilde{u} \notin e_1, \ldots, e_{NL} \end{aligned} \tag{3.17}$$

According to Eqs. 3.16, 3.17, we construct the full pattern matrix $P^{(k)}$ in Eq. 3.18 in a similar way to Eq. 3.6. For our algorithm the pattern matrix is is equal to our generating matrix $\mathbf{C}^{(\mathbf{k})} = \tilde{\mathbf{C}}$ which is also fixed for all iterations. Conceptually, the generating matrix $\tilde{\mathbf{C}}$ contains all the possible exploratory moves while a heuristic is utilized for evaluating the objective function g only for a subset of them.

$$\begin{aligned} \mathbf{C}^{(\mathbf{k})} &= \tilde{\mathbf{C}} = [\tilde{\boldsymbol{\Gamma}}\tilde{\mathbf{L}}] = [\tilde{\mathbf{M}} - \tilde{\mathbf{M}}\tilde{\mathbf{L}}] \\ \mathbf{P}^{(\mathbf{k})} &= \tilde{\mathbf{P}} \equiv \mathbf{B}\tilde{\mathbf{C}} \equiv \tilde{\mathbf{C}} \end{aligned} \tag{3.18}$$

Finally, we configure the updates of the step length parameter for each class of both successful and unsuccessful iterations as they were previously described in Eqs. 3.8,3.9 respectively.

A short description of our algorithm as a GPS method for solving the problem stated in Eq. 3.13 follows: In each iteration, we fix the optimal coordinate direction for each one of the points lying on the low dimensional manifold $x_i \in M, 1 \leq i \leq N$. For each internal iteration of Alg. 3.4, if the optimal direction produces a lower value for our objective function g we accumulate this direction and move alongside this coordinate of the $\mathbf{R}NL$. Otherwise, we remain at the same position. As a result, the exploration of coordinates for the new point x_{i+1} begins from this temporary position. This greedy approach provides a potential one-hot vector as described in Eq. 3.14 if the iterate is successful or otherwise, the zero vector $0 \in R_{NL}$. The final direction vector $\tilde{s}^{(k)}$ for kth iteration is computed by summing these one-hot or zero vectors. At the kth iteration, the movement would be given by a scalar multiplication of the step length parameter $\Delta^{(k)}$ with the final direction vector in a similar way as defined in Eq.3.7. This provides a simple decrease for the

objective function g or in the worst case represent a zero movement in the search space R_{NL}. Regarding the movement across $\tilde{s}^{(k)}$, it is trivial to show that this reduction of the objective function g is an associative operation. In other words, accumulating all best coordinate steps for each point $xi_{i=1}^{N}$ and performing the movement at the end of the kth iteration (as GPS method formulation requires) produces the same result as taking each coordinate step individually. Finally, pattern search MDS terminates when the step length parameter $\Delta^{(k)}$ becomes smaller than a predefined threshold.

3.4.5 Updating the Current Dissimilarity Matrix

In line 6 of Algorithm 3.4 we observe that we recompute the dissimilarity matrix after we make a move for each point. This can be avoided because each move modifies only one point $\mathbf{x}_i^{(k)}$, therefore only the row $\mathbf{d}_{i,:}^{(k)}$ and column $\mathbf{d}_{:,i}^{(k)}$ of the dissimilarity matrix $\mathbf{D}^{(k)}$ are affected. Furthermore only one dimension l of the vector $\mathbf{x}_i^{(k)}$ is modified by the move, i.e., only element $x_{i,l}^{(k)}$ of matrix $\mathbf{X}^{(k)}$. In detail, the element $d_{i,j}$ that stores the dissimilarity between points $\mathbf{x}_i$ and $\mathbf{x}_j$ should be updated as follows for the move from $x_{i,l}^{(k)}$ to $x_{i,l}^{(k+1)}$ for $i \neq j$:

$$d_{i,j}^{(k+1)} = \sqrt{\left(d_{i,j}^{(k)}\right)^2 - \left(x_{i,l}^{(k)} - x_{j,l}^{(k)}\right)^2 + \left(x_{i,l}^{(k+1)} - x_{j,l}^{(k+1)}\right)^2} \tag{3.19}$$

3.4.6 Complexity

For each epoch we search across $2L$ dimensions for N points. In each search we also need $\mathcal{O}(N)$ operations to update the distance matrix. Thus, the per epoch computational complexity of the algorithm is $\mathcal{O}(N^2 L)$.

Chapter 4

Pattern Search MDS with Landmarks (LPS-MDS)

4.1 The LPS-MDS Algorithm

The idea of Landmark points [2, 3] sounded very promising when trying to find ways to improve the algorithm complexity of PSMDS algorithm, as described in Chapter 4. The key idea of Landmarks method is that it is unnecessary to compute distances between all points in the distance matrix. At least $n \geq L + 1$ points (Landmark points) are sufficient, from which we can calculate our distances from all the other points.

If we know the positions of the Landmark points, then all the remaining points can be uniquely placed in space relative to them. This claim is analogous to the GPS method, where to find the exact location of a point, it is enough to know its distance from a finite number of points. The Landmark points can be random as long as their number is greater than $L + 1$ (where L is the dimension of the embedding).

In the entrance of the MDS (Multidimensional Scaling), we have the distance matrix $N \times N$ with the distances of each point from all the others. During the algorithm, we try to find vectors of dimension L such that the final distance matrix $N \times N$ is as close as possible to the original one. The distance matrix becomes $n \times N$, where $n \ll N$.

At the beginning, we execute MDS normally as if only the Landmark points existed, which will be placed normally in the embedding. Then, the remaining $N - n$ points will be placed in the embedding based on their distances from these Landmark points. The difference in the algorithm is that the input distance matrix P and the distance matrix D, dimension is $N \times n$ instead of $N \times N$, where $n \ll N$. It is not so computationally heavy.

The algorithm's initialization process includes the following steps:

- Generate n random points in the embedded space with the final -desired L dimensions. $X_a^{(0)} = [x_1^{(0)}, x_2^{(0)}, ..., x_n^{(0)}] \in \mathbb{R}^{n \times L}$. As n we refer in the number of Landmarks.

- Compute the distance matrix between landmark points. The element $d_{ij}^{(0)}$ is the Euclidean distance between vectors $x_i^{(0)}$ and $x_j^{(0)}$ of $X_a^{(0)}$.

- Calculate the initial approximation error $e^{(0)} = ||\mathbf{\Delta} - \mathbf{D}||_F^2$, where e represents the element-wise mean squared error (MSE) between the two matrices. The objective function which is optimized for all PSMDS is shown below and is the stress function

for MDS:

$$f(\mathbf{\Delta}, \mathbf{D}) = ||\mathbf{\Delta} - \mathbf{D}||_F^2 = \sum_{i=1}^{n}\sum_{j=1}^{n}(\delta_{ij} - d_{ij})^2, \quad \text{where } \mathbf{\Delta}, \mathbf{D} \in \mathbb{R}^{n\times n} \tag{4.1}$$

Now all the landmark $X^{(a)}$ points have been positioned in the embedded space. We will repeat the same procedure with the rest $N - n$ points.

- Generate $N-n$ random points in the embedded space with the final -desired L dimensions. $X_b^{(0)} = [x_{n+1}^{(0)}, x_{n+2}^{(0)}, ..., x_N^{(0)}] \in \mathbb{R}^{(N-n)\times L}$. The total points of the embedded space are $X^{(0)} = X_a \cup X_b^{(0)} \in \mathbb{R}^{(N)\times L}$
- Compute the distance matrix between X_a and $X_b^{(k)}$. The element $d_{ij}^{(0)}$ is the Euclidean distance between vectors $x_i^{(0)}$ and $x_j^{(0)}$ of $X^{(0)}$.
- Calculate the initial approximation error $e^{(0)} = ||\mathbf{\Delta} - \mathbf{D}||_F^2$, where e represents the element-wise mean squared error (MSE) between the two matrices. The objective function which is optimized for all PSMDS is shown below and is the stress function for MDS:

$$f(\mathbf{\Delta}, \mathbf{D}) = ||\mathbf{\Delta} - \mathbf{D}||_F^2 = \sum_{i=n}^{N}\sum_{j=n}^{N}(\delta_{ij} - d_{ij})^2, \quad \text{where } \mathbf{\Delta}, \mathbf{D} \in \mathbb{R}^{n\times n} \tag{4.2}$$

ALGORITHM 4.1: *Pattern Search MDS with Landmarks*

Input: The matrix $\mathbf{\Delta} \in \mathbb{R}^{N\times N}$ with proximities, the dimension L and the number of landmark points n.

Output: The matrix $\mathbf{X} \in \mathbb{R}^{N\times N}$, with the coordinates of the N objects, in the new space of M dimensions.

procedure LANDMARK-PSMDS($\mathbf{\Delta}$, L, $r^{(0)}$,n)
  $k \leftarrow 0$ ▷ k is the number of epochs
  $\mathbf{X}^{(k)} \leftarrow$ UNIFORM($n \times M$)
  $\mathbf{D}^{(k)} \leftarrow$ DISTANCE_MATRIX($\mathbf{X}^{(k)}$) ▷ $D \in \mathbb{R}^{n\times n}$
  $e^{(k)} \leftarrow ||\mathbf{\Delta}_{n\times n} - \mathbf{D}||_F^2$ ▷ First n rows in $\mathbf{\Delta}$ are considered as distances between Landmarks
  $e^{(k-1)} \leftarrow +\infty$
  $r^{(k)} \leftarrow r^{(0)}$
  e^*,$X^* \leftarrow$ MDS($r^{(k)}$,$\mathbf{X}^{(k)}$,$e^{(k-1)}$,$e^{(k)}$,M) ▷ positioning only the landmark points
  $k \leftarrow 0$
  $\mathbf{X}^{(k)} \leftarrow X^* \cup$ [UNIFORM($(N-n) \times M$)]
  $\mathbf{D}^{(k)} \leftarrow$ DISTANCE_FROM_LANDMARKS_MATRIX($\mathbf{X}^{(k)}$,n) ▷ $D \in \mathbb{R}^{N\times n}$
  $e^{(k)} \leftarrow ||\mathbf{\Delta}_{(N-n)\times n} - \mathbf{D}||_F^2 + e^*$
  $e^{(k-1)} \leftarrow +\infty$
  $r^{(k)} \leftarrow r^{(0)}$
  _,$X^{(k)} \leftarrow$ MDS($r^{(k)}$,$\mathbf{X}^{(k)}[n:N]$,$e^{(k-1)}$,$e^{(k)}$,M)
end procedure

ALGORITHM 4.2: *MDS - positioning points*

function MDS($r^{(k)}$,$\mathbf{X}^{(k)}$,$e^{(k-1)}$,$e^{(k)}$,M)
  while $r^{(k)} > \delta$ **do**
    if $e^{(k-1)} - e^{(k)} \leq \epsilon \cdot e^{(k)}$ **then**
      $r^{(k)} \leftarrow \frac{r^{(k)}}{2}$
    end if
    $\mathcal{S} \leftarrow$ SEARCH_COORDINATES($r^{(k)}$, L)
    for all $x \in \mathbf{X}^{(k)}$ **do**
      $\mathbf{X}^*, e^* \leftarrow$ OPTIMAL_MOVE($\mathbf{D}^{(k)}$, $\mathbf{X}^{(k)}$, x, $\mathcal{S}$, $e^{(k)}$)
      $e^{(k-1)} \leftarrow e^{(k)}$
      $e^{(k)} \leftarrow e^*$
      $\mathbf{X}^{(k)} \leftarrow \mathbf{X}^*$
      $\mathbf{D}^{(k)} \leftarrow$ UPDATE_DISTANCE_MATRIX ($\mathbf{X}^*$)
    end for
    $k = k + 1$
  end while
  return $e^{(k)}$, $\mathbf{X}^{(k)}$
end function

The functions $SEARCH\_COORDINATES()$ and $OPTIMAL\_MOVE()$ are similar as descripted in Pattern Search MDS, as in Algorithms 3.3 and 3.4 correspondingly.

4.1.1 Complexity

For each epoch we search across $2L$ dimensions for N points. In each search we also need $\mathcal{O}(n)$ operations to update the distance matrix. Thus, the per epoch computational complexity of the algorithm is $\mathcal{O}(nNL)$, where $n \ll N$.

4.2 Experiments with Synthetic Dataset

In this experimental setup we perform non-linear dimensionality reduction on a 3D Swissroll (as in Figure 4.1) using Landmarks MDS. The assumption here is that data lie on a manifold which is locally Euclidean and thus geodesic distances would preserve the true structure of the data when performing MDS.

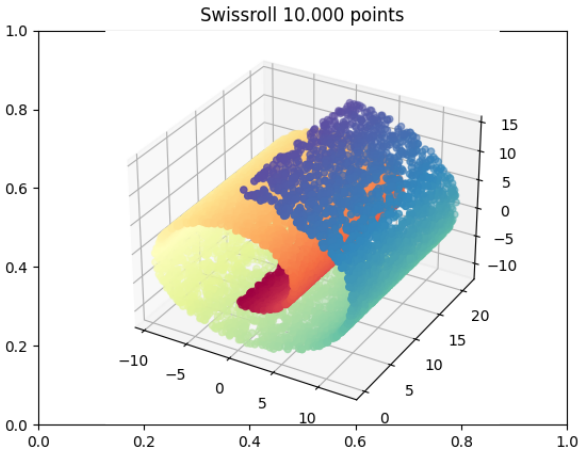


Figure 4.1. *Original Swissroll in 3D*

In our initial experiment, our goal was to identify the ideal number of landmarks necessary to effectively unroll the Swissroll in two dimensions, all while maintaining a consistent level of error comparable to the results obtained without the use of landmarks. Remarkably, we found that utilizing only 300 landmarks proves to be more than sufficient for this purpose. Our experimentation was carried out on Swissroll datasets consisting of 1,000, 10,000, and 20,000 data points, each examined with varying quantities of landmarks. Notably, across all scenarios, the employment of landmarks consistently yielded highly satisfactory results. This is further substantiated by the data presented in 4.1, where it is evident that with 300 landmarks, we achieved a deviation of less than 1.5% from the original error.

Landmarks	1000 points	10000 points	20000 points
-	error - %relative change	error - %relative change	error - %relative change
10	3532380.7 (64424,85%)	22347409.89 (81626,62%)	1796591175.21 (3606595,19%)
100	5642.89 (3,08%)	28561.27 (4,45%)	129743825.55 (260363,50%)
200	5607.63 (2,43%)	28399.87 (3,85%)	50749.89 (1,88%)
300	**5550.87 (1,40%)**	**27626.71(1,03%)**	**50489.65 (1,36%)**
500	5508.3 (0,62%)	27533.46 (0,69%)	50169.29 (0,72%)
1000	**5474.45 (0,00%)**	27435.7 (0,33%)	49952.37 (0,28%)
5000	-	27357.04 (0,05%)	49826.26 (0,03%)
10000	-	**27344.1 (0,00%)**	49811.5 (0,00%)
15000	-	-	49829.69 (0,03%)
20000	-	-	**49812.67(0,00%)**

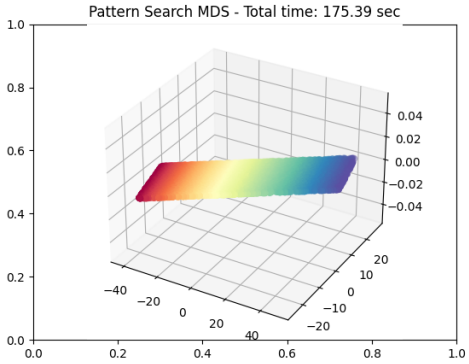
Table 4.1. *In this table, we display results from experiments using the Swissroll dataset of 1,000 to 20,000 points with varying numbers of landmarks. It is shown that 300 landmarks resulted in less than a 1.5% deviation from the original error, in all cases.*

To continue with, in Table 4.2, we provide a summary of the results when unrolling a 10,000 points Swissroll, with Pattern Search MDS and MDS with Landmarks. Firstly, we calculate the execution time in seconds. Notably, the standard MDS algorithm takes 171.21

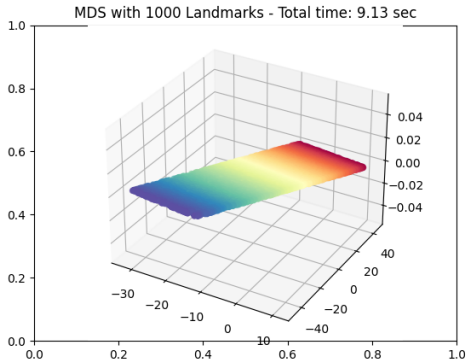
seconds to complete, whereas the MDS with 300 landmark points significantly reduces the runtime to just 3.58 seconds. Moreover, it's evident that, with a sufficient number of landmarks (at least 300), the algorithm requires fewer epochs to converge. Lastly, we have calculated the RAM usage in gigabytes (GB), and as expected, the Landmark-based approach consumes less memory. A Visual Swissroll unrolling in 2 dimensions is given in Figures 4.2.

Algorithm	Time(Sec)	Epochs	Final Error	RAM(GB)
MDS	**171.21**	245	**27344.1**	1.15
100 Landmarks MDS	4.04	202	28561.27	0.67
300 Landmarks MDS	**3.58**	96	**27626.71**	0.86
1000 Landmarks MDS	8.53	94	27435.7	0.92

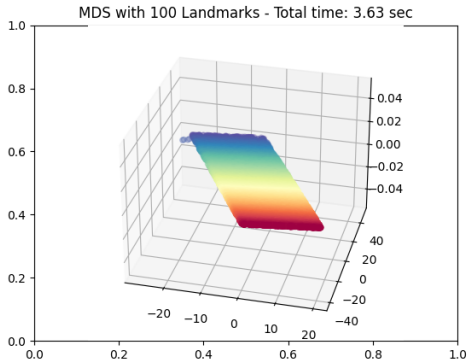
Table 4.2. *Results from experiment with 10.000 Swissroll. MDS with Landmarks executes in less time, with fewer epochs and less memory.*



(a) *Pattern Search MDS*



(b) *MDS with 1000 Landmarks*



(c) *MDS with 100 Landmarks*

Figure 4.2. *Visual Swissroll unrolling in 2 dimensions*

In all the above executions of the algorithm, the landmark are chosen randomly. In the following graph you may see with black color the 300 Landmark points in a 10.000 points experiment.

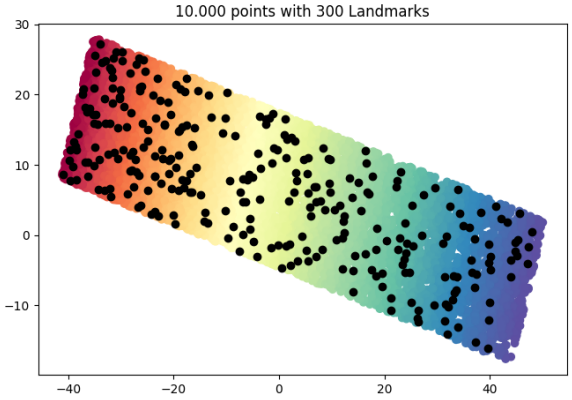
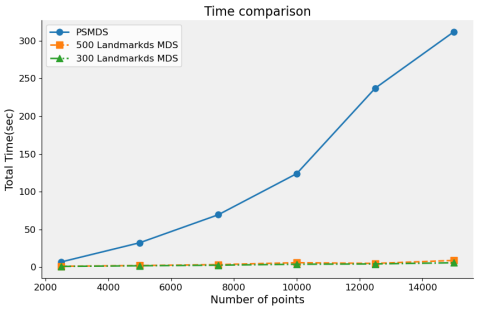
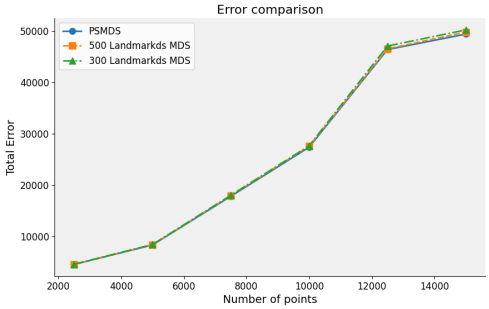


Figure 4.3. *Original Swissroll in 2D, with Landmarks*

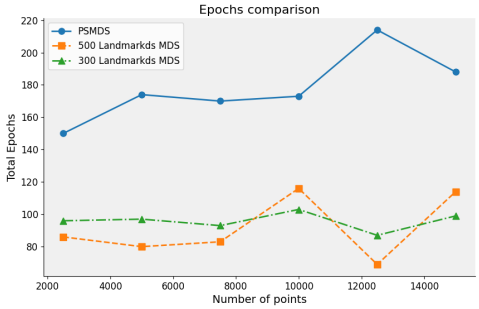
In the following graphs, we present a comparative analysis between Landmarks MDS and Pattern Search MDS in terms of Time, Error, and Epochs for datasets with varying numbers of points (2500, 5000, 7500, 10000, 12500, and 15000). This experiment provides valuable insights into the performance of these algorithms across different data sizes.
Notably, our observations reveal that the error remains consistent between the two algorithms, indicating similar quality of results. However, there are significant improvements in terms of computation time when using Landmarks MDS, especially as the dataset size increases. The Landmark MDS approach consistently demonstrates quicker convergence, reducing the time required to achieve the desired results.
These findings highlight the efficiency and effectiveness of Landmarks MDS in handling larger datasets, making it a promising choice for dimensionality reduction tasks. Overall, the graphical comparisons offer valuable insights into the performance trade-offs between these two approaches across various data sizes.



(a) *Time comparison*



(b) *Error comparison*



(c) *Epochs comparison*

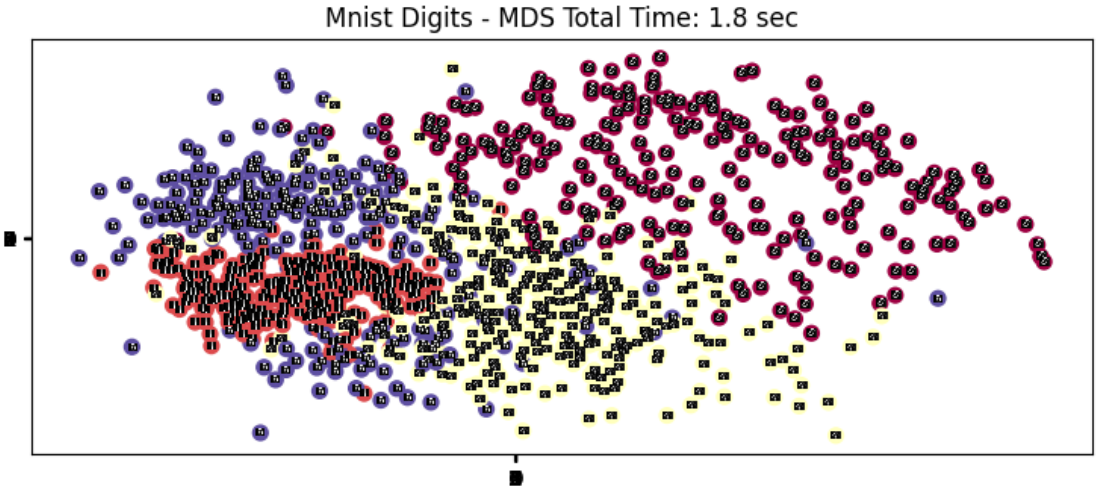
Figure 4.4. *Comparison between PSMDS and LPS-MDS. LPS-MDS has remarkable results in all cases*

4.2.1 Mnist Dataset

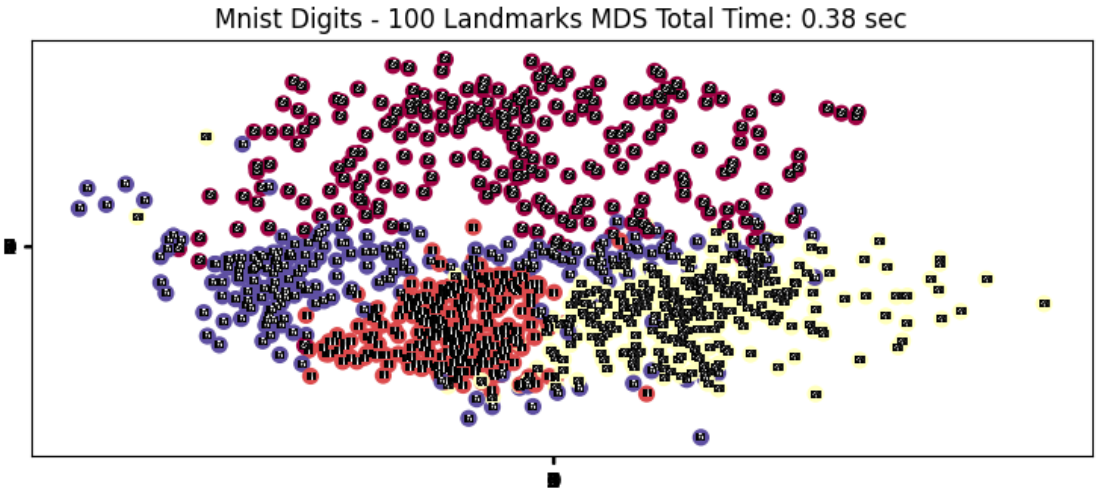
In this experimental setup we perform dimensionality reduction using as target dissimilarity matrices the Euclidean ones from the high-dimensional pixel space $\mathbb{R}^{784}$ of MNIST dataset [62].

4.2.1.1 Visualizing Linear Embeddings

We specifically chose to work with a subset of handwritten digits, limiting our selection to four out of the ten available digits: 0, 1, 4, 8. From these four classes, we randomly selected 1000 images for our analysis. Our initial step involved computing the Euclidean distance matrix, which captures the pairwise distances between all the images in the pixel space. Following this, we performed dimensionality reduction, aiming to represent the data in two dimensions. We employed both Pattern Search MDS and MDS with Landmarks for this task. Remarkably, both algorithms achieved a similar level of accuracy, as indicated by their Stress errors. To aid in visualizing our results, we assigned distinct colors to the different handwritten digit classes, mapping 0, 1, 4, 8 to red, orange, yellow, purple respectively. The resulting embeddings revealed a compelling outcome. The MDS algorithms successfully preserved the geometric relationships among the handwritten digits. Notably, digits belonging to the same class exhibited a tendency to cluster together in compact 2D Euclidean subspaces, showcasing the effectiveness of our approach.



(a) *Pattern Search MDS*



(b) *Landmark MDS*

4.2.1.2 Image Classification on the Low-Dimensional Space

To continue with, we use a subset of 3000 images and we perform linear-dimensionality reduction using Pattern Search MDS and Landmark PSMDS with 100 and 300 landmark points in order to obtain 10-dimensional embeddings. To evaluate the performance of the algorithms, K-Nearest Neighbor (KNN) classification is performed using different parameters of nearest neighbors. I also perform KNN on the original space without performing any dimensionality reduction in order to get a better insight of the accuracy. 2700 images are used as training set and 300 as test set. The execution time is improved with landmark points, while using 300 landmark points instead of 100 leads to better accuracy scores.

To further our analysis, we worked with a subset comprising 3000 images, and we performed dimensionality reduction to 10 dimensions with both techniques. In our evaluation process, we utilized K-Nearest Neighbor (KNN) classification, experimenting with various nearest neighbor parameter settings. Additionally, we conducted KNN classification on the original data space, bypassing any dimensionality reduction. This approach provided

valuable insights into the accuracy of our results. For our experiments, we partitioned the data into 2700 images for training and 300 images for testing purposes. Notably, our use of landmark points significantly improved execution time. Furthermore, employing 300 landmark points, as opposed to 100, resulted in enhanced accuracy scores, underlining the effectiveness of our methodology.

Method	Dimensions	Time(sec)	K=3	K=5	K=7	K=9
Initial	784	-	0.90	0.89	0.91	0.92
PSMDS	10	140.54	0.90	0.87	0.89	0.89
100LandmarksPSMDS	10	**11.53**	0.83	0.86	0.82	0.81
300LandmarksPSMDS	10	**19.55**	**0.89**	**0.89**	0.88	0.86

Table 4.3. *Comparison of dimensionality reduction techniques for the MNIST dataset with embedding dimension equal to 10. For the classification KNN algorithm is with different number of nearest neighbors (parameter K) is used. "Time" column specifies the amount of time that each method took to compute all embedding vectors. "Initial" corresponds to performing KNN on the original high-dimensional pixel without any dimensionality reduction.*

4.2.2 Classification on Microbiome Data Set

The microbiome dataset of Turnbaugh [63] refers to a collection of data generated by researchers that characterizes the microbial communities present in various biological samples. These samples were collected from different sources within the human or mouse gastrointestinal tract, and possibly other locations, to study the composition and diversity of microorganisms in those environments. Within this dataset, we find 675 unique donors, each of which is characterized by 6,686 Operational Taxonomic Units (OTUs).[1] Each donor in this dataset is further categorized across five distinct attributes, each possessing varying values, as descripted below:

1. **Sex:** Male, Female
2. **Diet:** LFPP, Western, CARBR, FATR, Suckling, Human
3. **Source:** Cecum1, Cecum2, Colon1, Colon2, ,SI1, SI13, SI15, SI2, SI5, SI9, Stomach, Cecum
4. **Donor:** HMouseLFPP, CONVR, Human, Fresh, Frozen, HMouseWestern, CONVD
5. **CollectionMet:** Contents,Scraping

In our research, we conducted dimensionality reduction from the original 6,686 dimensions down to only 3 dimensions. For the classification process we utilized K-Nearest

[1] In the realm of microbiology and microbial ecology, an OTU signifies a cluster of closely related microorganisms, such as bacteria or archaea. OTUs are typically defined based on the sequence similarity of specific genes, such as the 16S ribosomal RNA (rRNA) gene for bacteria or the 18S rRNA gene for eukaryotic microorganisms.

Neighbor (KNN) classification with K=9. To facilitate classification, we employed the K-Nearest Neighbor (KNN) classification technique with a parameter value of K=9. Additionally, we conducted KNN classification on the initial dataset, prior to dimensionality reduction, for comparative purposes. We run the K-Nearest Neighbor multiple times to find independently the accuracy score for Sex, Diet, Source, Donor and CollectionMet. The results revealed that categories with only two distinct values, such as Sex and CollectionMet, exhibited impressive performance, with KNN achieving classification accuracy scores nearing 90%. Diet and Source also yielded promising results, demonstrating consistent scores across various methodologies. Notably, the challenge emerged in classifying Donor, with low accuracy rates around 30%. This suggests that certain microorganisms are shared between animals and humans, making them indistinguishable through any algorithmic means.

Method	Dimensions	Diet	Source	Donor	CollectionMet	Sex
Initial	6686	0.79	0.64	0.29	0.85	0.9
PSMDS	3	0.78	0.63	0.29	0.85	0.9
100LandmarksPSMDS	3	0.74	**0.64**	0.23	**0.85**	**0.9**
300LandmarksPSMDS	3	**0.81**	**0.64**	**0.28**	**0.85**	**0.9**

Table 4.4. *Comparison of dimensionality reduction techniques for the Micro-biame Data Set with embedding dimension equal to 3. For the classification KNN algorithm with K=9 is used. "Initial" corresponds to performing KNN on the original high-dimensional pixel without any dimensionality reduction. The columns Diet, Source, Donor, CollectionMet, Sex are the different categories of the classification.*

Chapter 5

The COSMOS Algorithm

5.1 Notation

Let $\mathbb{M} \subseteq \mathbf{R}^n$ represent the ambient space for our manifold, where ν denotes the total number of sets, μ denotes the number of points, and L signifies the dimension of the embedded space. Each point of the collection of points $\tilde{x} = \{x_1, \cdots, x_\mu\}$ corresponds to a row of the matrix $\mathbf{X} \in \mathbf{R}^{\mu \times L}$. Also, we index the sets $\mathbf{S} = \{S_1, \cdots, S_\nu\}$. Notably, each set S_i may include one or more elements from the collection $\tilde{x}$, ensuring that $S_i \cap S_j = \phi$ and $|\bigcup S_i| = \mu$. We also denote a mapping matrix $M_{sets} \in \mathbf{N}^{1 \times \mu}$ that maps the μ points into the sets $\mathbf{S}$.

Additionally, we denote the function $D_{points}(x_i, x_j)$ as the euclidean distance between two points and the function $D_{sets}(S_i, S_j)$ as the common sense (equation 5.1) distance between sets in (pseudo)-metric space. The set distance matrix $\mathbf{\Delta}$ is induced by set of points with dimension M, where $M > L$.

For any $S_1, S_2 \in \mathbf{S}$ the common sense set distance is defined as:

$$D_{sets}(S_1, S_2) = \min_{x_i \in S_1, x_j \in S_2} d(x_i, x_j) \tag{5.1}$$

5.1.1 The definition of Set-MDS Problem

The Set-MDS problem is defined in a given metric space $\mathbb{M}$ and $\nu, \mu, L \in \mathbf{N}_+$ such that $\nu \leq \mu$ and a dissimilarity matrix $\mathbf{\Delta}_{\nu \times \nu}$ of ν-sets of μ M-dimensional points, the output is the matrix $\mathbf{X} \in \mathbf{R}^{\nu \times L}$ and the sets $\mathbf{S}$ such that the objective function:

$$f(\mathbf{D}, \mathbf{\Delta}) = ||\mathbf{D} - \mathbf{\Delta}||_F^2 = \sum_{i=1}^{n} \sum_{j=1}^{n} (d_{ij} - \delta_{ij})^2, \quad \text{where } \mathbf{\Delta}, \mathbf{D} \in \mathbb{R}^{\nu \times \nu} \tag{5.2}$$

is minimized, where $\mathbf{D} = d_{ij}$ is the induced set distance matrix from the $\mathbf{X}$ and the function D_{sets}.

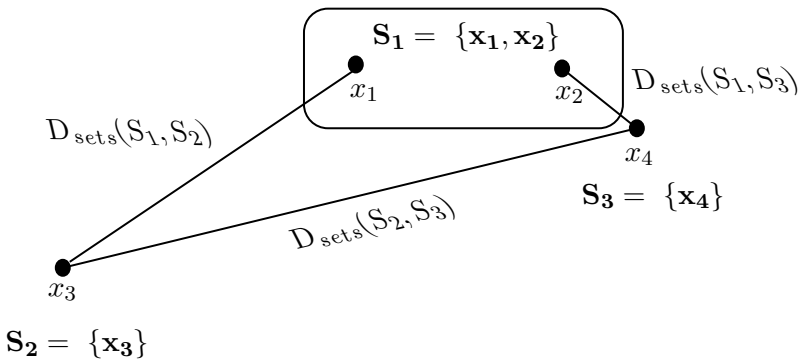


Figure 5.1. *Example of non metric power set space*

5.1.2 The violation of triangle inequality in Power Sets

Consider a finite metric space $(\mathbb{M}, L)$ and the power set of $\mathbb{M}$, $P(\mathbb{M})$, which is also finite. It is easy to see from the figure 5.1 that we may find triplets $S_1, S_2, S_3 \in P(\mathbb{M})$ where the triangle inequality does not hold.

Thus, under the "common sense" set distance the power set space $P(\mathbb{M})$ is not metric, and $(\mathbb{M}, L)$ is not a metric space, hence, the nice convergence properties of metric spaces and the resulting notion of neighborhood are not satisfied. Although these notions might not exist globally in the power set space, they are satisfied locally under some assumptions. For instance, if $\mathbb{M}$ has n elements, $x_1, x_2, \ldots, x_n$, then the set space $Q \triangleq \{\{x_1\}, \{x_2\}, \ldots, \{x_n\}\} \subset P(\mathbb{M})$ is a metric space under D_{sets}.

To demonstrate that the power set space, $(P(M), D_{sets})$, is not metric in general, consider the set $Y \triangleq \{\{x_1, x_2\}, \{x_3\}, \ldots, \{x_n\}\} \subset P(\mathbb{M})$, i.e., Y consists of a single set with two elements $A = \{x_1, x_2\}$ and the single element sets of all the remaining members of $\mathbb{M}$, i.e., $\{x_3\}, \{x_4\}, \ldots, \{x_n\}$. Then, the triangle inequality is satisfied for all triplets of elements of Y, with the possible exception of triplets containing set A, where it may not holds.
For instance, an illustrative example is depicted in Fig. 5.1 with $n = 4, A = \{x_1, x_2\}$, $B = x_3$, and $C = x_4$, where we have:

$$D_{sets}(S_2, S_3) > D_{sets}(S_1, S_2) + D_{sets}(S_1, S_3) \tag{5.3}$$

The probability of violating the triangle inequality is higher when the two members of S_1 are far away in the underlying metric space $(\mathbb{M}, L)$, i.e., $D_{points}(x_1, x_2) > 2\delta$ and in addition, x_3 and x_4 are in the δ-neighborhood of x_1 and x_2 , respectively (or vice versa), i.e., $D_{points}(x_1, x_3) < \delta$ and $D_{points}(x_2, x_4) < \delta$.

In the context of word semantics, we are going to give an example. Let's assume that in the Fig 5.1 let the word W_{S_1} represented by the set S_1, consisting of just two word senses, x_1 and x_2, while words W_{S_2} and W_{S_3}, two monosemous words with a single word sense each, x_3 and x_4, respectively. For example, let the polysemous word W_{S_1} ='book' with corresponding word senses x_1 ='book with the sense of reservation' and x_2 ='book with the sense of reading' and the monosemous words W_{S_2}='travel' and W_{S_3}='author'. Then,

apparently the triangle inequality is violated for the pair ('travel','author').

The above example reveals that the space $W = (W, 1 - S_W)$ does not form a metric space because triangle inequality is not satisfied.

5.2 The COSMOS Algorithm

In this section, the COSMOS (COmmon Sense Multidimensional Optimization Splitter) algorithm is described. COSMOS is a dimensionality reduction algorithm that projects a set of μ objects into a vector space of L dimensions, and gives a heuristic solution to the Set-MDS problem. In this thesis, the algorithm is tested in words, but the algorithm could easily be applied to an arbitrary set of objects.

The inputs to the algorithm are the dissimilarity or semantic distance matrix $\mathbf{\Delta} \in \mathbb{R}^{\nu \times \nu}$ where each element $\Delta(i, j)$ encodes the degree of dissimilarity between points δ_i and δ_j, the projection dimension L and the number of splits $\mathbf{k}$. The output of the algorithm is the matrix $\mathbf{X} \in \mathbf{R}^{\mu \times L}$ with $\mu = \nu + \mathbf{k}$ points in the embedded L-dimensional space, the ν sets and the $\mathbf{k}$ splits.

COSMOS algorithm starts by generating μ random points and running the LPS-MDS, as described in Chapter 4. This preliminary step is crucial for ensuring a better starting point before the points, $\mathbf{X} \in \mathbb{R}^{\nu \times L}$ start to split in sets. After the initialization process, we start splitting the $\mathbf{k}$ points. The algorithm is structured to execute exactly $\mathbf{k}$ iterations, during each of which a new point (split) is generated.

Below, we provide a detailed explanation of these iterations:

- Find the new split by generating $2 * L$ random points with L dimensions in the embedded space and assign each one of them in a set. Choose the combination of the point and the set that minimizes the error most. $\mathbf{X}$ now has one more point.

- Calculate the Euclidean distances between the newly introduced split point and all other points. Subsequently, we append a new row and column to the D_{points} matrix, which stores the Euclidean distances among all points. This matrix forms the basis for constructing the set distance matrix, denoted as D_{sets}, where the element d_{ij} represents the common set distance, as defined in Equation 5.1.

- Move each point x_i in a sphere of radius r around the point x_i, as in figure 3.1. Choose the direction that minimizes the error most. The error is the following equation: where e represents the element-wise mean squared error (MSE) between the two set distance matrices. The objective function which is similar with the MDS Stress function, but the matrices now are set matrices.

$$f(\mathbf{D}, \mathbf{\Delta}) = ||\mathbf{D} - \mathbf{\Delta}||_F^2 = \sum_{i=1}^{n} \sum_{j=1}^{n} (d_{ij} - \delta_{ij})^2, \quad \text{where } \mathbf{\Delta}, \mathbf{D} \in \mathbb{R}^{\nu \times \nu} \tag{5.4}$$

ALGORITHM 5.1: *COSMOS(COmmon Sense Multidimensional Optimization Splitter)*

Input: The matrix $\mathbf{\Delta} \in \mathbb{R}^{\nu \times \nu}$ with dissimilarities and the dimension L.

Output: The matrix $\mathbf{X} \in \mathbb{R}^{\mu \times L}$ with the coordinates of the μ elements in $L-$dimensional space, where $\mu = \nu + k$, and a list of sets $\mathbf{S}$.

procedure SET - MDS($\mathbf{\Delta}$, L, $r^{(0)}$)
  x_1 to x_ν,$\mathbf{D} = MDS(\mathbf{\Delta})$ ▷ As described in Chapter 3
  for $j = 1, ..., k$ **do** ▷ k is the number of splits
    $\mathbf{G}$ = GENERATE_RANDOM_POINTS(2L)
    for all $g \in \mathbf{G}$ **do**
      $\mathbf{D}^*_{points}$ = COMPUTE_TEMP_Dpoints_MATRIX(g, $\mathbf{X}$)
      for all $x \in \mathbf{X}_\nu$ **do**
        $\mathbf{S}^* = split\_set(x, g)$; ▷ x, g are in the same set
        $\mathbf{D}^*_{sets}$ = COMPUTE_TEMP_Dsets_MATRIX($\mathbf{S}^*$,$\mathbf{D}^*_{points}$)
        e = COMPUTE_ERROR($\mathbf{\Delta}$, $\mathbf{D}^*_{sets}$)
        Save error e for combination g and x;
      end for
    end for
    Choose the combination of g and x that minimizes the error most;
    $\mathbf{S} \leftarrow \mathbf{S}^*$
    $\mathbf{D}_{points} \leftarrow \mathbf{D}^*_{points}$
    $\mathbf{D}_{sets} \leftarrow \mathbf{D}^*_{sets}$
    $k \leftarrow 0$
    while $r^{(k)} > \delta$ **do**
      if $e^{(k-1)} - e^{(k)} \leq \epsilon \cdot e^{(k)}$ **then**
        $r^{(k)} \leftarrow \frac{r^{(k)}}{2}$
        $\mathcal{S} \leftarrow$ SEARCH_COORDINATES($r^{(k)}$, $2L$)
      end if
      for all $x \in x_1$ to x_{n+j} **do**
        $\mathbf{X}^*, s^*_l, e^* \leftarrow$ OPTIMAL_MOVE($\mathbf{X}^{(k)}$, x, $\mathcal{S}$, $e^{(k)}$)
        $e^{(k-1)} \leftarrow e^{(k)}$
        $e^{(k)} \leftarrow e^*$
        $\mathbf{X}^{(k)} \leftarrow \mathbf{X}^*$
        $\mathbf{D}_{points} \leftarrow$ UPDATE_Dpoints_MATRIX($\mathbf{X}^*$)
        $\mathbf{D}_{sets} \leftarrow$ UPDATE_Dsets_MATRIX($\mathbf{D}_{points}$, $\mathbf{S}$)
      end for
      $k = k + 1$
    end while
  end for
end procedure

5.2.1 Runtime Complexity

Firstly, we execute the LPS-MDS whose complexity for ν points and n landmarks and L dimensions is $\mathcal{O}(n\nu L^2)$ per round. We execute $\mathbf{k}$ epochs, one for every split of the following . We generate $2L$ random points, and we assign each generated point to all ν sets and we compute the error for all possible combinations of random points and sets. To compute the error we need $\mathcal{O}(\mu L)$ time. Until now the complexity of the algorithm is $\mathcal{O}(2Lk\mu L) = \mathcal{O}(2L^2k\mu)$.

Then apply a variation of MDS and we start moving all the points μ in $2L$ possible dimensions, and we need to choose the best possible step. The time for this is $\mathcal{O}(2L\mu)$. Then we need $\mathcal{O}(\mu)$ time to update the D_{points} and $\mathcal{O}(\mu)$ to update the D_{sets}. So for updating the distance matrices we need $\mathcal{O}(\mu + \mu)$=$\mathcal{O}(2\mu)$

The final complexity is $\mathcal{O}(2L^2k\mu 2L\mu 2\mu) = \mathcal{O}(4kL^3\mu^3)$

5.2.2 Experiments

First, we will present the algorithm's performance on a synthetic dataset that we have constructed. Subsequently, we will provide some results tested on semantic similarity between words.

5.2.2.1 Synthetic Dataset

To generate the Synthetic Dataset for this experiment, we initiated the process by creating μ random points, each with L dimensions, where the values range from -1 to 1. Subsequently, these μ points were randomly assigned to ν sets. The outcome comprises a distance matrix, denoted as $\mathbf{\Delta} \in \mathbb{R}^{\nu\times\nu}$, and an index matrix, designated as $M_{sets} \in \mathbb{R}^{1\times\mu}$, serving as a mapping between the vectors from $\mathbf{\Delta}$ to the ν sets. Our expectation is that the COSMOS algorithm, given the matrix $\mathbf{\Delta}$ and the number k as input, will accurately split the correct points.

The following routines show in detail our techniques.

Algorithm 5.2: *Synthetic Dataset Creation*

Input: The number of sets ν, the number of splits k and the number of dimensions L.
Output: The distance matrix $\mathbf{\Delta} \in \mathbb{R}^{\nu\times\nu}$ with the distances between sets.

1: **procedure** SYNTHETIC_DATASET_CREATION(μ,$\mathbf{k}$,L)
2: $g \leftarrow generate\_random\_points(\mu,L,\text{-}1,1)$
3: $D_{points} \leftarrow Distance\_matrix\_between\_points(g)$
4: $D_{sets} \leftarrow Generate\_random\_integers(\nu)$
5: $D_{sets} \leftarrow Distance\_matrix\_between\_sets(D_{points})$
return D_{sets}, $\mathbf{S}$
6: **end procedure**

In the table 5.1 below the results of multiple experiments are summarized. We executed experiments with different number of points (1000,500,100) with different number of splits k (1,5,10,20,50,100,150,200,300,400,500), while performing dimensionality reduction from 3 dimensions to 2. Our goal is to evaluate whether the COSMOS algorithm chooses to make the correct splits. We compare the expected final sets generated from the synthetic dataset, with the output of the algorithm. The results show that COSMOS algorithm can recognize with percentage above 65% the correct splits.

Points \ Splits	1	5	10	20	50	100	150	200	300	400	500	Average
1000	1.00	0.80	0.80	0.75	0.72	0.64	0.65	0.65	0.65	0.62	0.63	0.70
500	1.00	0.60	0.80	0.65	0.68	0.60	0.71	0.58	0.62	0.62		0.66
100	1.00	0.60	0.60	0.60	0.70							0.67

Table 5.1. *The table summarizes the results of multiple experiments. Is shows the percentage of successful splits in experiments with different number of points μ and splits* ***k***

5.2.3 Semantic Similarity - MEN & SimLex-999

For this experiment we use MEN and SimLex-999 semantic datasets as ground truth. Both datasets are provided in the form of lists of word pairs, where each pair is associated with a similarity score. This score was computed by averaging the similarities provided by human annotators. As the high-dimensional semantic word vectors, we use the 300-dimensional GloVe vectors. We reduce the dimensionality of the vectors to the target dimension L and calculate the Spearman correlation coefficient between the human provided and the automatically computed similarity scores. Results are summarized in Table 5.2 for L = 10. We observe PSMDS and Landmark PSMDS with 300 landmark points yields the best results for MEN, UMAP performs best SimLex-999, while COSMOS algorithm has also good results.

Method	Dimensions	MEN	SimLex-999
-	300	0.74	0.37
PSMDS	10	**0.68**	0.30
300 Landamark PSMDS	10	**0.67**	0.23
100 Landamark PSMDS	10	**0.61**	0.21
SMACOF	10	0.56	0.23
UMAP	10	0.29	0.44
COSMOS(2splits)	10	0.59	0.25
COSMOS(20splits)	10	-	0.25

Table 5.2. *Comparison of dimensionality reduction techniques for the semantic similarity task for MEN and SimLex-999 datasets.*

5.2.4 Semantic Similarity - Wordnet

For the following experiment, we use the Wordnet hierarchy. As explained in [64] WordNet is a large lexical database of English words, where nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. In Fig.5.2 a toy example of hierarchy structure provided by WordNet is depicted. Actually, it corresponds to a tree-like structure where the top category of everything is the 'entity' and thereafter the tree expands to lower levels into other sub categories and words. For example, in the figure two sub-categories are the 'physical entity' and 'abstract entity' which are also expanded to lower levels. Note that, ambiguous words will be assigned to different parts (or and different levels) of the tree structure for example the word 'state' is mapped under the 'region' and also under the 'attribute'.

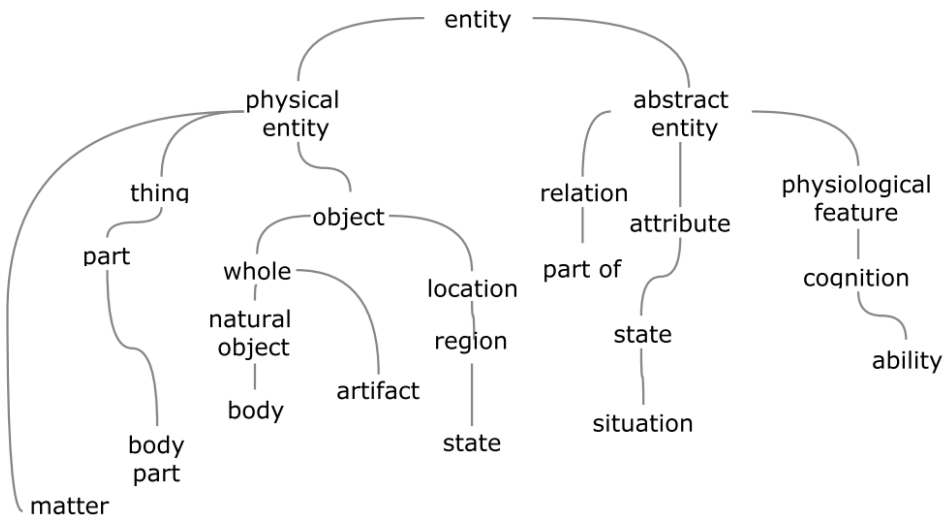


Figure 5.2. *Example of the structure of WordNet hierarchy.*

We firstly asked ChatGPT 3.5, to generate bags of words that they share one common word, but the other words in each bag have different meanings. We give an example of the output of chatGPT in Figure 5.3.

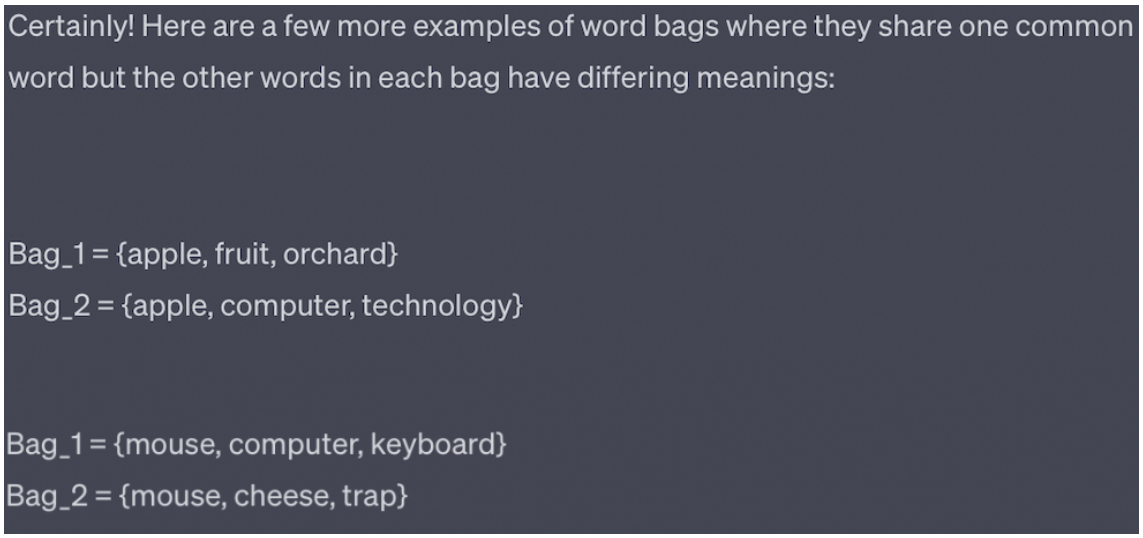


Figure 5.3. *ChatGPT example usage*

Subsequently, we use WordNet Similarity in order to construct the input matrix $\mathbf{\Delta}$ in the COSMOS algorithm. As described in [65] WordNet::Similarity is a freely available software package that makes it possible to measure the semantic similarity and relatedness between a pair of concepts (or synsets). The measures take as input two concepts, and return a numeric value that represents the degree to which they are similar or related.

We use the NLP | WuPalmer – WordNet Similarity [66] that it calculates relatedness by considering the depths of the two synsets in the WordNet taxonomies, along with the depth of the LCS (Least Common Subsumer).

$$S(x_1, x_2) = 2\frac{depth(lcs(x_1, x_2))}{depth(x_1) + depth(x_2)} \tag{5.5}$$

The score can be $0 < score <= 1$. The score can never be zero because the depth of the LCS is never zero (the depth of the root of taxonomy is one). It calculates the similarity based on how similar the word senses are and where the Synsets occur relative to each other in the hypernym tree.

Since the COSMOS algorithm uses as input a dissimilarity or semantic distance matrix, the pairwise word similarity matrix $W \in R^{\nu \times \nu}$ is transformed to a semantic distance (or dissimilarity) matrix $\mathbf{\Delta}_k \in \mathbb{R}^{\nu \times \nu}$ as:

$$\mathbf{\Delta}_k(i, j) = c_1 \cdot e^{-c_2 \cdot S(i,j)} \tag{5.6}$$

where $c_1, c_2 \in R$ are constants and the i, j indexes run from 1 to n. In this work, we experimentally chosen $c_1 = c_2 = 0.1$. The transformation defined by 5.6 was selected in order to non-linearly scale and increase the relative distance of dissimilar words compared to similar ones. Except from the exponential nonlinear scaling function from similarity to distance, similar function can be found in the literature, e.g., [41]

as the Logarithmic Scaling:

$$\mathbf{\Delta}_k(i, j) = -c \cdot \log(S(i, j)) \tag{5.7}$$

and the Inverse Scaling:

$$\mathbf{\Delta}_k(i,j) = \frac{c}{S(i,j)} \tag{5.8}$$

In our experiments the equation 5.6 seemed to have the best results. The below graphics prove that COSMOS algorithm choose to split the correct points and position them in a correct place.

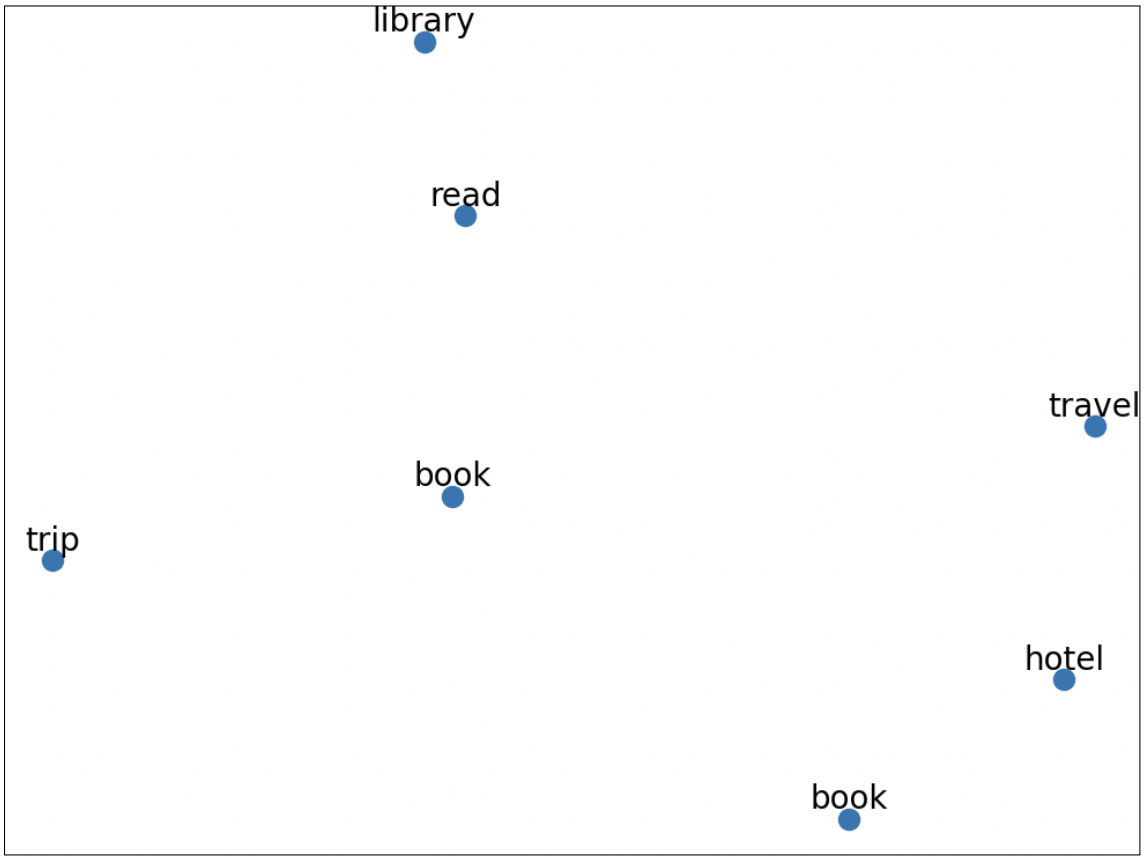


Figure 5.4. *The algorithm correctly splits the word "book" and selects positions close to "read" and "hotel", proving both meanings.*

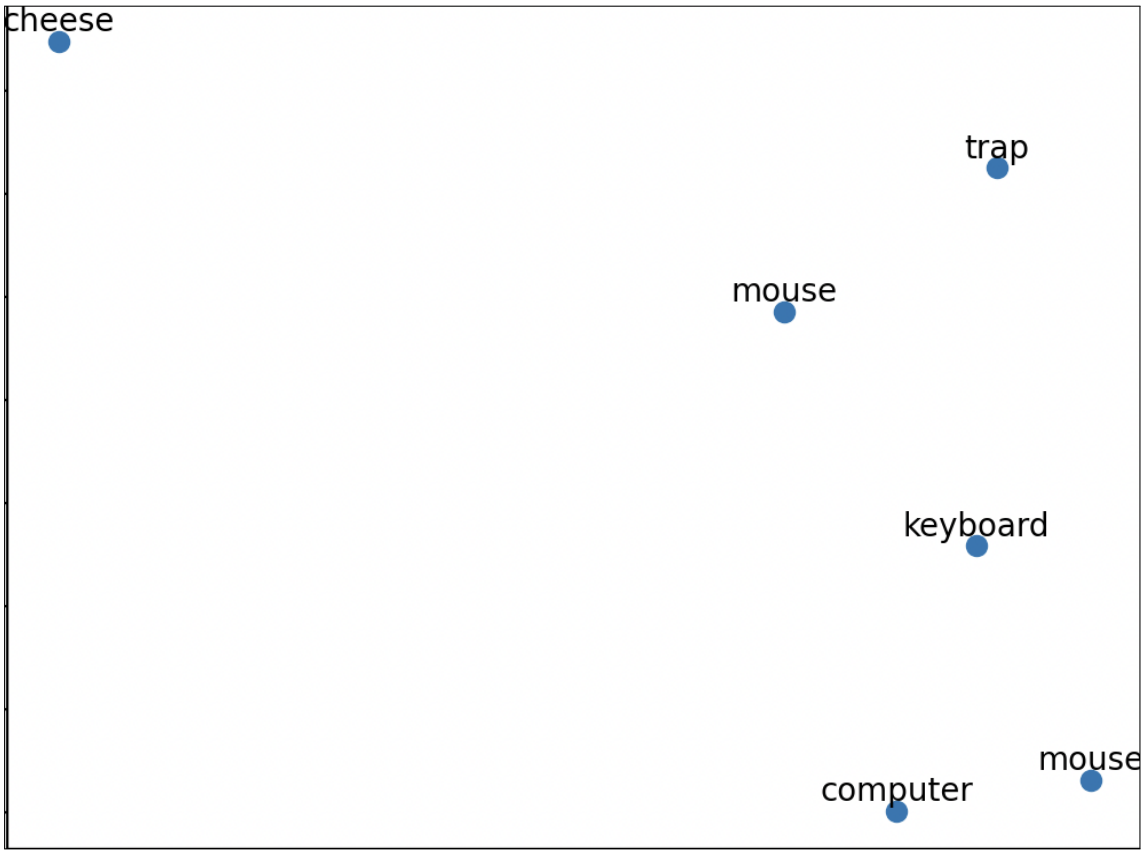
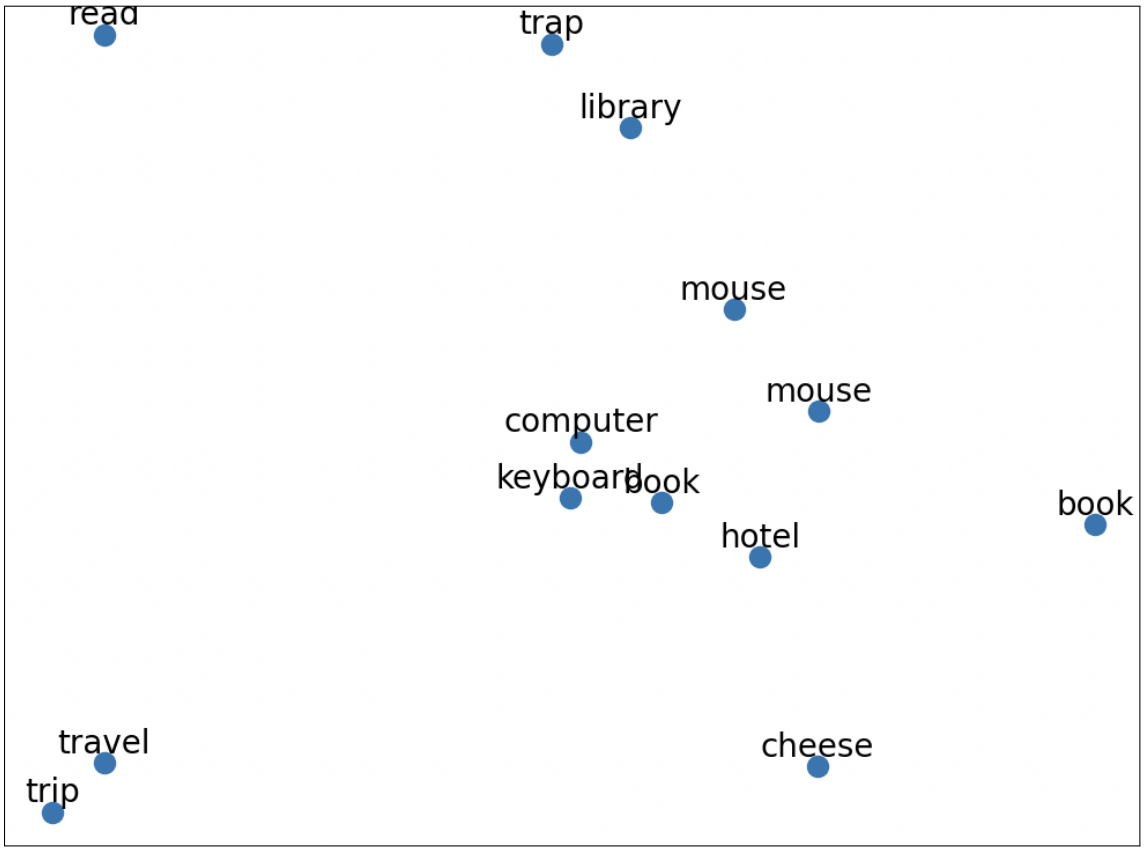


Figure 5.5. *The algorithm correctly splits the word "mouse" and selects positions close to "trap" and close to "computer", proving both meanings.*



(a) *Time comparison*

Figure 5.6. *This graph is a combination of the previous words. We ask the algorithm to execute two splits, and it splits correctly the words "mouse" and "book".*

Chapter 6

Conclusions

The dimensionality reduction and data visualization of complex and high-dimensional data is a challenging problem with multiple real-world applications. There are a lot of dimensionality reduction techniques that achieve effectively to embed the data in a lower-dimensional space while simultaneously preserving the relationship between them.

This thesis concentrates on Multi-Dimensional Scaling (MDS), specifically introducing an extension of Classical MDS known as Pattern Search MDS (PSMDS), employing derivative-free optimization techniques. Our aim was to enhance the complexity of the existing PSMDS algorithm, which, in addressing the MDS problem, computes distances between all points to establish the distance matrix in the embedded space. The proposed Landmark Pattern Search algorithm introduces "Landmark" points, calculating distances only from these points instead of between all points.

Our experimental findings with LPS-MDS demonstrate a significant improvement in algorithmic complexity when employing Landmark points, while maintaining result quality comparable to the initial PSMDS algorithm. We conducted multiple experiments using synthetic datasets, the Mnist Dataset featuring images of integers, and the Microbiome Data Set containing genes from diverse biological samples. Across all experiments, LPS-MDS consistently delivered outstanding results.

To continue with, in the second part of the thesis we propose another extension of MDS, that we call COSMOS(COmmon Sense Multidimensional Optimization Splitter) algorithm. The COSMOS algorithm, has all the characteristics of the MDS algorithm, so it preserves the distances between objects and applies stress minimization during the dimensionality reduction process. A noteworthy feature of the COSMOS algorithm is its adaptation to operate on sets of points, rather than individual sets. This is the unique characteristic of our algorithm that adds one more level of complexity in the classical MDS and it gives the flexibility of understanding ambiguity in objects.

In order to evaluate the performance of our algorithm, we firstly executed experiments on synthetic datasets, yielding promising results. Subsequently, we devised a semantic similarity task, specifically addressing word ambiguity, where a single word may have multiple meanings. Our objective was to examine the algorithm's capability to resolve such ambiguity. For the semantic similarity task, we utilized word pairs from MEN and SimLex, employing Glove as our word vectors. Our experimental outcomes prove that the COSMOS algorithm adeptly captures word ambiguity. The algorithm demonstrates

highly encouraging results, underscoring its potential for practical applications and further exploration within NLP-related contexts.

6.1 Future Work

Our COSMOS algorithm is taking its first steps to solve the Set-MDS problem, helping untangle data that looks like knots. As future work we plan to further refine the algorithm, to optimize its computational efficiency and scalability, ensuring its applicability to larger datasets.

Another very interesting research is the integration with real-world applications. Currently the proposed representation has been only tested in semantic similarity, in order to capture the ambiguity of the words, but there are so much more we can explore. In fact there are a lot of fields that the data is ambiguous or its structure contains knots, like in genes or medical images.

Taking the gene ambiguity as an example, when working with genomic data, the same genes can do different things depending on the context. It becomes crucial to disambiguate between genes, ensuring that their unique functions are appropriately considered in the analysis. Furthermore, in the medical image processing field, the interpretation of medical imaging results, such as X-rays or MRIs, can be complex. Radiological findings may have multiple possible explanations, requiring collaboration between clinicians and radiologists to arrive at accurate diagnoses.

The examples discussed earlier revolve around our key areas of interest in biology and medicine, but someone could find applications in other fields such as Social Media, Economy, Art or Recommendation Systems.

Bibliography

[1] I. Liaperdos, L. Dermentzoglou, A. Arapoyanni και Y. Tsiatouhas. *Fault Detection in RF Mixers Combining Defect-Oriented and Alternate Test Strategies*. *26th Conference on Design of Circuits and Integrated Systems (DCIS)*, San Sebastian, Spain, 2011.

[2] V. D. Silva και J. B. Tenenbaum. *Global versus local methods in nonlinear dimensionality reduction*. *Advances in Neural Information Processing Systems 15*, σελίδες 705–712, 2003.

[3] Vin Silva και Joshua Tenenbaum. *Sparse Multidimensional Scaling using Landmark Points*. *Technology*, 2004.

[4] G. Paraskevopoulos, E. Tzinis, E. V. Vlatakis-Gkaragkounis και A. Potamianos. *Pattern Search Multidimensional Scaling*. *ArXiv:1806.00416, 2018. arXiv: 1806.00416 [quant-ph]*, 2018.

[5] K. F. Pearson. *Liii. on lines and planes of closest fit to systems of points in space*. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[6] J. B. Tenenbaum, V.d. Silva και J. C. Langford. *A global geometric framework for nonlinear dimensionality reduction*. *Science*, 290(5500):2319–2323, 2000.

[7] M. Bernstein, V. D. Silva, J. C. Langford και J. B. Tenenbaum. *Graph approximations to geodesics on embedded manifolds*. 2000.

[8] H. Zha και Z. Zhang. *Isometric embedding and continuum isomap*. *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, σελίδες 864–871, 2003.

[9] D. L. Donoho και C. Grimes. *Image manifolds which are isometric to Euclidean space*. *Journal of Mathematical Imaging and Vision*, 23(1):5–24, 2005.

[10] R. Pless. *Image spaces and video trajectories: Using isomap to explore video sequences*. *ICCV*, τόμος 3, σελίδες 1433–1440, 2003.

[11] V. D. Silva και J. B. Tenenbaum. *Global versus local methods in nonlinear dimensionality reduction*. *Advances in Neural Information Processing Systems 15*, σελίδες 705–712. MIT Press, 2003.

[12] M. Belkin και P. Niyogi. *Laplacian eigenmaps for dimensionality reduction and data representation*. *Neural Comput.*, 15(6):1373–1396, 2003.

[13] L. Cayton και S. Dasgupta. *Robust Euclidean embedding*. *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, σελίδες 169–176, 2006.

[14] L. K. Saul και S. T. Roweis. *Think globally, fit locally: Unsupervised learning of low-dimensional manifolds*. *J. Mach. Learn. Res.*, 4:119–155, 2003.

[15] F. Sha και L. K. Saul. *Analysis and extension of spectral methods for nonlinear dimensionality reduction*. *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, σελίδες 784–791, 2005.

[16] M. Belkin και P. Niyogi. *Laplacian eigenmaps and spectral techniques for embedding and clustering*. *Advances in Neural Information Processing Systems 14*, σελίδες 585–591. MIT Press, 2002.

[17] Z. Zhang και J. Wang. *MLLE: Modified locally linear embedding using multiple weights*. *Advances in Neural Information Processing Systems*, σελίδες 1593–1600, 2007.

[18] Z. Zhang και H. Zha. *Principal manifolds and nonlinear dimensionality reduction via tangent space alignment*. *SIAM journal on scientific computing*, 26(1):313–338, 2004.

[19] D. L. Donoho και C. Grimes. *Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data*. *Proceedings of the National Academy of Sciences*, 100(10):5591–5596, 2003.

[20] K. Q. Weinberger και L. K. Saul. *Unsupervised learning of image manifolds by semidefinite programming*. *International journal of computer vision*, 70(1):77–90, 2006.

[21] K. Q. Weinberger, B. Packer και L. K. Saul. *Nonlinear dimensionality reduction by semidefinite programming and kernel matrix factorization*. *AISTATS*, 2005.

[22] L. Vandenberghe και S. Boyd. *Semidefinite programming*. *SIAM Rev.*, 38(1):49–95, 1996.

[23] D. P. Bertsekas. *Nonlinear programming*. Athena scientific Belmont, 1999.

[24] Bernhard Schölkopf, Alexander Smola και Klaus Robert Müller. *Nonlinear component analysis as a kernel eigenvalue problem*. *Neural computation*, 10(5):1299–1319, 1998.

[25] Luis M Rios και Nikolaos V Sahinidis. *Derivative-free optimization: A review of algorithms and comparison of software implementations*. *Journal of Global Optimization*, 56(3):1247–1293, 2013.

[26] Mordecai Avriel. *Nonlinear programming: Analysis and methods*. Courier Corporation, 2003.

[27] R Hooke και TA Jeeves. *"Direct search" solution of numerical and statistical problems*. *J. ACM*, 8(2):212–229, 1961.

[28] George EP Box. *Evolutionary operation: A method for increasing industrial productivity*. *Applied statistics*, σελίδες 81–101, 1957.

[29] Virginia J Torczon. *Multidirectional search: A direct search algorithm for parallel machines*. Διδακτορική Διατριβή, Rice University, 1989.

[30] John J Dennis, Jr και Virginia Torczon. *Direct search methods on parallel machines*. *SIAM Journal on Optimization*, 1(4):448–474, 1991.

[31] Virginia Torczon. *On the convergence of pattern search algorithms*. *SIAM Journal on optimization*, 7(1):1–25, 1997.

[32] E D Dolan, R M Lewis και V Torczon. *On the local convergence of pattern search*. *SIAM Journal on Optimization*, 14(2):567–583, 2003.

[33] R. M. Lewis και V. Torczon. *Pattern Search Methods for Linearly Constrained Minimization*. *SIAM Journal on Optimization*, 10(3):917–941, 2000.

[34] Robert M Lewis και Virginia Torczon. *Pattern search algorithms for bound constrained minimization*. *SIAM Journal on Optimization*, 9(4):1082–1099, 1999.

[35] Andrew R Conn, Nicholas I M Gould και Philippe Toint. *A globally convergent augmented lagrangian algorithm for optimization with general constraints and simple bounds*. *SIAM Journal on Numerical Analysis*, 28(2):545–572, 1991.

[36] Tim Roughgarden και Gregory Valiant. *The Modern Algorithmic Toolbox: Lecture 9 - The Singular Value Decomposition (SVD) and Low-Rank Matrix Approximations*, 2017.

[37] Avrim Blum, John Hopcroft και Ravindran Kannan. *Foundations of Data Science*. 2016. Pages 44-50.

[38] Tim Roughgarden και Gregory Valiant. *Understanding and Using Principal Component Analysis (PCA)*. *CS168: The Modern Algorithmic Toolbox*, 2016.

[39] Xiaoming Huo, Xuelei (Sherry) Ni και Andrew K. Smith. *A SURVEY OF MANIFOLD BASED LEARNING METHODS, Chapter 1*. Διδακτορική Διατριβή, Georgia Institute of Technology, 1989.

[40] W. S. Torgerson. *Multidimensional scaling: I. theory and method*. *Psychometrika*, 17(4):401–419, 1952.

[41] Ingwer Borg. *Modern Multidimensional Scaling: Theory and Applications*. Springer, 2005.

[42] J. C. Gower. *Some distance properties of latent root and vector methods used in multivariate analysis*. *Biometrika*, 53(3-4):325–338, 1966.

[43] M. A. A. Cox και T. F. Cox. *Multidimensional Scaling on the Sphere*. *Compstat*D. Edwards και N. E. Raun, επιμελητές, σελίδες 323–328. Physica-Verlag HD, 1988.

[44] A. Cvetkovski και M. Crovella. *Low-stress Data Embedding in the Hyperbolic Plane using Multidimensional Scaling*. *Appl. Math*, 11(1):5–12, 2017.

[45] H. Lindman και T. Caelli. *Constant Curvature Riemannian Scaling*. *Journal of Mathematical Psychology*, 17(2):89–109, 1978.

[46] R. N. Shepard. *The Analysis of Proximities: Multidimensional Scaling with an Unknown Distance Function. I.* *Psychometrika*, 27(2):125–140, 1962.

[47] R. N. Shepard. *The Analysis of Proximities: Multidimensional Scaling with an Unknown Distance Function. II.* *Psychometrika*, 27(3):219–246, 1962.

[48] P. Groenen και I. Borg. *Past, Present, and Future of Multidimensional Scaling*. *Visualization and Verbalization of Data*, σελίδες 95–117. 2014.

[49] J. B. Kruskal. *Multidimensional Scaling by Optimizing Goodness of Fit to a Nonmetric Hypothesis*. *Psychometrika*, 29(1):1–27, 1964.

[50] J. B. Kruskal. *Nonmetric Multidimensional Scaling: A Numerical Method*. *Psychometrika*, 29(2):115–129, 1964.

[51] L. M. Rios και N. V. Sahinidis. *Derivative-free optimization: A review of algorithms and comparison of software implementations*. *Journal of Global Optimization*, 56(3):1247–1293, 2013.

[52] M. Avriel. *Nonlinear programming: Analysis and methods*. Courier Corporation, 2003.

[53] R. Hooke και T. A. Jeeves. *"Direct search" solution of numerical and statistical problems*. *J. ACM*, 8(2):212–229, 1961.

[54] G. E. Box. *Evolutionary operation: A method for increasing industrial productivity*. *Applied statistics*, σελίδες 81–101, 1957.

[55] V. J. Torczon. *Multidirectional Search: A Direct Search Algorithm for Parallel Machines*. Διδακτορική Διατριβή, Rice University, 1989.

[56] J. J. E. Dennis και V. Torczon. *Direct Search Methods on Parallel Machines*. *SIAM Journal on Optimization*, 1(4):448–474, 1991.

[57] V. Torczon. *On the Convergence of Pattern Search Algorithms*. *SIAM Journal on Optimization*, 7(1):1–25, 1997.

[58] E. D. Dolan, R. M. Lewis και V. Torczon. *On the local convergence of pattern search*. *SIAM Journal on Optimization*, 14(2):567–583, 2003.

[59] R. M. Lewis και V. Torczon. *Pattern search methods for bound constrained minimization*. *SIAM Journal on Optimization*, 9(4):1082–1099, 1999.

[60] A. R. Conn, N. I. M. Gould και P. Toint. *A globally convergent augmented lagrangian algorithm for optimization with general constraints and simple bounds*. *SIAM Journal on Numerical Analysis*, 28(2):545–572, 1991.

[61] C. Audet. *Convergence Results for Generalized Pattern Search Algorithms are Tight*. *Optimization and Engineering*, 5(2):101–122, 2004.

[62] Yann LeCun. *The MNIST Database of Handwritten Digits*. http://yann.lecun.com/exdb/mnist/, 1998.

[63] Peter J. Turnbaugh, Vanessa K. Ridaura, Jeremiah J. Faith, Federico E. Rey, Rob Knight και Jeffrey I. Gordon. *The Effect of Diet on the Human Gut Microbiome: A Metagenomic Analysis in Humanized Gnotobiotic Mice*. *Science Translational Medicine*, 1:6ra14, 2009.

[64] Georgia Athanasopoulou. *Semantic Similarity Computation and Word Sense Induction using Hidden Sets Multidimensional Scaling*. Πτυχιακή εργασία, TECHNICAL UNIVERSITY OF CRETE SCHOOL OF ELECTRONIC AND COMPUTER ENGINEERING, 2016.

[65] Siddharth Patwardhan Ted Pedersen και Jason Michelizzi. *WordNet::Similarity - Measuring the Relatedness of Concepts*.

[66] GeeksforGeeks. *NLP | Wu&Palmer (WordNet Similarity)*. https://www.geeksforgeeks.org/nlp-wupalmer-wordnet-similarity/.