Visualisierung von Musikdaten mittels t-SNE und PCA am Beispiel pgvector

Jannis Gehring
INF22B, Data Warehouse
Duale Hochschule Baden-Württemberg (DHBW)
Stuttgart, Deutschland
inf22115@lehre.dhbw-stuttgart.de

Contents

I. Einleitung/Abstract

In modernen Datawarehouse-systemen handelt es sich um große und vielfältige Datenmengen. Um aus diesen Daten sinnvolle Schlussfolgerungen zu ziehen, müssen sie meist aufs wesentliche reduziert werden. Eine Methode ist hierbei die Dimensionsreduktion, bei der hochdimensionale Daten (bis zu mehrere tausend Dimensionen) in niedrig-dimensionale Daten (meist 2/3-dimensional) umgewandelt werden. Im Folgenden werden die Dimensionsreduktionsverfahren Principal Component Analysis (PCA) und t-distributed SNE (t-SNE) beschrieben und die Implementierung in scikit-learn vorgestellt. Die Datenspeicherung erfolgt in einer PostgreSQL Datenbank. Als Datengrundlage dient ein kaggle-Datensatz mit Musikdaten aus Spotify.

II. Theoretische Grundlagen

A. Principal Component Analysis (PCA)

Principal Component Analysis (PCA) ist ein lineares Verfahren zur Dimensionsreduktion. Es wurde in der ersten Hälfte des zwanzigsten Jahrhunderts entwickelt, fand aber aufgrund seiner Berechnungsanforderungen erst in den 60ern breite Anwendung. [1]

Mathematisch liegt PCA eine Eigenvektorberechnung zugrunde. Vereinfacht wird zu Beginn die Kovarianz-Matrix $\boldsymbol{S}$ des Datensatzes berechnet. Dann werden die Eigenvektoren dieser Matrix berechnet und nach dem Betrag ihrer jeweiligen Eigenwerte sortiert. Von diesen n Eigenvektoren werden nun, je nach Anwendungsfall, die k ersten gewählt ($k \leq n$). Mit einer weiteren Matrix $\boldsymbol{W}$, die jene gewählten Eigenvektoren als Zeilenvektoren hat, wird nun die Transformation der Ausgangstupel in den (meist niedriger-dimensionalen) Raum durchgeführt. [2]

Das folgende Beispiel gibt dieser mathematischen Definition eine handfeste Intuition. Hier entspricht PCA dem iterativen Auswählen von zueinander orthogonalen Linien durch den Datensatz, die diesen am besten teilen. *Am besten* bedeutet hier, dass die Varianz der Projektionen auf diese Linie maximal sind. [2]

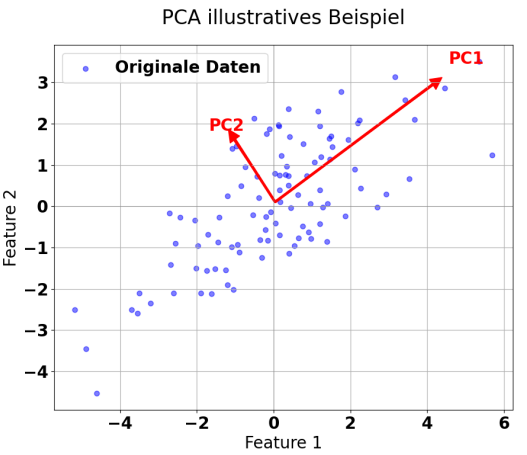

Fig. 1: Veranschaulichung von PCA mit $n = k = 2$

B. t-distributed SNE (t-SNE)

t-distributed SNE (t-SNE) ist ein nicht-lineares Verfahren zur Dimensionsreduktion. Es wurde 2008 als Weiterentwicklung des Verfahrens Stochastic Neighbour Embedding (SNE) vorgestellt.

Zentral für SNE ist, dass Nachbarschaftsbeziehungen des hochdimensionalen Raums so gut wie möglich im niedrig-dimensionalen Raum erhalten bleiben müssen. Hier zu wird im Ausgangsraum X die Wahrscheinlichkeit $p_{j|i}$ definiert. Wenn vom Punkt $x_i \in X$ ein zufälliger Nachbar gewählt wird, wobei nähere Punkte anhand einer Gauß-verteilung wahrscheinlicher sind als fernere, dann bezeichnet $p_{j|i}$ die Wahrscheinlichkeit, dass hierbei der Punkt x_j gewählt wird.

Genau die gleichen Beziehungen werden im niedrig-dimensionalen Raum $Y \supset \{y_i, y_j\}$ definiert, wobei die Wahrscheinlichkeit hier mit $q_{j|i}$ bezeichnet wird.

Ziel von SNE ist nun, die Punkte aus X so auf Y abzubilden, dass $q_{j|i}$ für alle i und j möglichst nahe an $p_{j|i}$ ist. Dies wird mittels eines Gradient-Descent Algorithmus durchgeführt.

t-distributed SNE (t-SNE) erweitert SNE in zwei Weisen. Zum Einen werden p und q so neu definiert, dass $p_{ij} = p_{ji}$ und $p_{ij} = p_{ji}$ $\forall i, j$. Zum Anderen nutzt es im niedrig-dimensionalen Raum nicht die Gaußverteilung zur Ermittlung des Nachbarn, sondern die t-Verteilung nach Student.

C. Vergleich von Principal Component Analysis (PCA) und t-SNE

Im Folgenden werden PCA und t-SNE anhand unterschiedlicher Kriterien verglichen [3]:

Kriterium	PCA	t-SNE
Ziel der Dimensionsreduktion	Maximieren der Varianz	Erhalten von lokaler Struktur
Linearität	Linear	Nicht-Linear
Iterativ?	Nein	Ja
Berechnungskomplexität	$O(d^2n + n^3)$*	$O(n^2)$

* d ist die Anzahl der Dimensionen des Ursprungsraumes, n die Anzahl der Datenpunkte

III. Verwendete Technologien

A. Datenbank: PostgreSQL mit pgvector und psycopg

PostgreSQL ist ein opensource relationales Datenbankmanagementsystem. Es geht zurück auf das POSTGRES Projekt von 1986 an der University of California, wird aber immer noch weiterentwickelt und ist weit verbreitet. [4] PostgreSQL kann sowohl mit einem cli als auch über einen Webclient, pgadmin, bedient werden.

Pgvector ist eine Erweiterung von PostgreSQL, die den Datentyp *vector* einführt und es so erlaubt, Vektoren in SQL zu speichern. Desweiteren verfügt sie über Funktionen wie nearest-neighbour-search und Abstandsberechnungen (cosine-distance, L1/L2 distance, etc.). [5]

Psycopg ist ein PostgreSQL Adapter für Python, der es ermöglicht, `INSERT`, `UPDATE`, `SELECT` sowie viele weitere SQL-Statements direkt aus Python heraus auszuführen. Da Psycopg auch Pgvector unterstützt, erlaubt es, alle für dieses Projekt relevanten Datenbankoperationen direkt in Python zu schreiben. [6]

B. Algorithmen: scikit-learn

scikit-learn ist eine Python Bibliothek für Machine Learning. Neben Klassifikations-, Regressions-, Clustering- und weiteren Algorithmen implementiert scikit-learn auch Dimensionsreduktionsverfahren wie PCA und t-SNE. [7]

C. Visualisierung: plotly

plotly ist eine opensource Visualisierungsbibliothek für Python, die mehr als 40 unterschiedliche Graphen unterstützt. Plotly ist für dieses Projekt besonders gut geeignet, weil die Visualisierungen interaktiv sind, dass heißt es kann beispielsweise gezoomt werden und Informationen zu einzelnen Datenpunkten können *on-hover* angezeigt werden. [8]

D. Datensatz:

Als Datengrundlage dient ein Datensatz von kaggle namens "Spotify Tracks Genre". Dieser verfügt über 89741 Songs aus 114 Genres. Zu jedem Song werden neben der track_id 19 Werte gespeichert, von denen die Folgenden für dieses Projekt von Interesse sind [9]:

1) **Meta-informationen**
   - Name des Songs
   - Name des Künstlers
   - Name des Genres
2) **Audio-parameter**

   Diese Parameter werden von Spotify ermittelt und beziehen sich auf den musikalischen Charakter des Songs.

Parameter	Interval/Einheit
Danceability	[0;1]
Energy	[0;1]
Loudness	db
Speechiness	[0;1]
Acousticness	[0;1]
Instrumentalness	[0;1]
Liveness	[0;1]
Valence*	[0;1]

* *Valence* beschreibt, wie musikalisch positiv ein Song ist.

IV. Umsetzungsbeispiel

A. Codeablauf

Folgendes Flowchart beschreibt den Ablauf des Programmes:

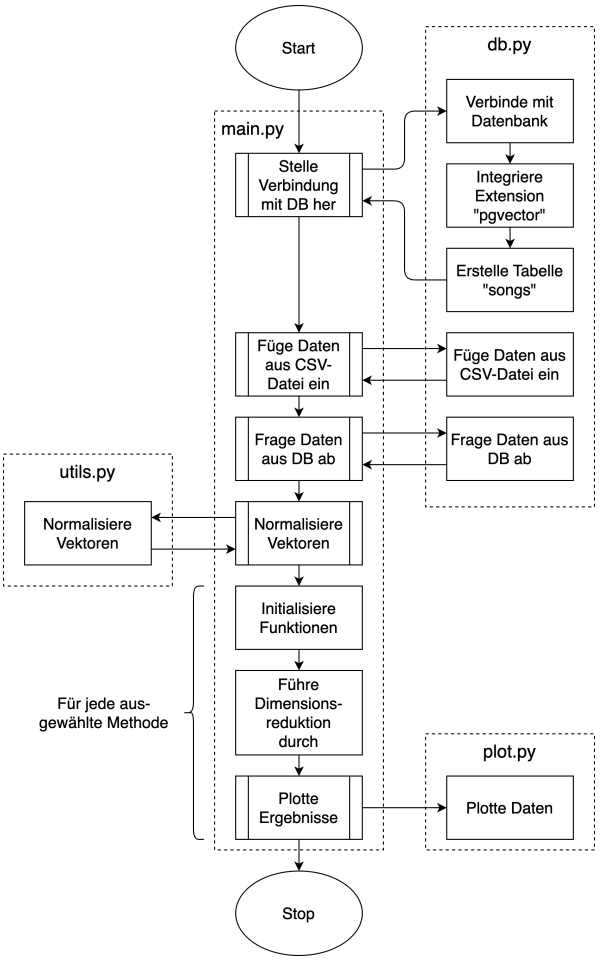

Fig. 2: Flowchart der Implementierung

B. How-to: Setupprozess

Es folgt eine Schritt für Schritt Anleitung für das Aufsetzen der Umgebung. Annahme ist, dass alle Terminalcommands aus dem obersten Verzeichnis des git-repositories ausgeführt werden und Docker und Python inklusive venv installiert sind.

1) **Klonen des Git-Repositories**
   `git clone https://github.com/GehriJan/music-vector-visualization.git && cd music-vector-visualization`
1) **Starten von Docker**
   macOS: `open -a docker`
2) **Starten der Docker-container**
   `docker-compose up -d`
3) **Aufsetzen einer venv**
   `python3 -m venv venv`
4) **Starten der venv**
   zsh/macOS: `source venv/bin/activate`
5) **Installieren der pip-packages**
   `pip install -r requirements.txt`

C. How-to: Nutzungsprozess

Das Programm kann vollständig aus dem Terminal genutzt und konfiguriert werden. Hierzu muss über

```
python3 src/main.py
```

die Hauptdatei aufgerufen werden.

Folgende Parameter sind notwendig:

1) `-m` / `--method`
   Hiermit werden die zur Dimensionsreduktion genutzten Verfahren ausgewählt. Aufgrund des einheitlichen Interfaces in scikit-learn sind neben t-SNE und klassischem PCA noch diverse Abwandlungen von PCA und FastICA wählbar. Die Argumente mappen wie folgt auf die Funktionen:

Argument	Funktion
pca	PCA
kernelpca	Kernel PCA
sparsepca	Sparse PCA
incrementalpca	Incremental PCA
tsne	t-SNE
fastica	Fast ICA

   Ein falsches Argument bringt das Programm mit einer darauf hinweisenden Exception zum Absturz.
2) `-g` / `--genre`
   Hier kann eine beliebige Teilmenge der im Datensatz vorliegenden Genres zur Visualisierung ausgewählt werden. Genrenamen, die nicht im Datensatz vorkommen, werden ignoriert.

Die Argumente werden immer durch Leerzeichen voneinander getrennt.

V. Ergebnisse

Die folgenden Charts demonstrieren das Plotting mit diversen Konfigurationen. Die Plots werden Stück für Stück im Browser geöffnet.

Folgende Features kennzeichnen die Plots:

1) An-/Abwählen einzelner Genres in der Legende
2) Zooming
3) Labels mit Titel und Künstler beim Hovern über die Daten
4) Weitere Funktionen im oberen rechten Eck

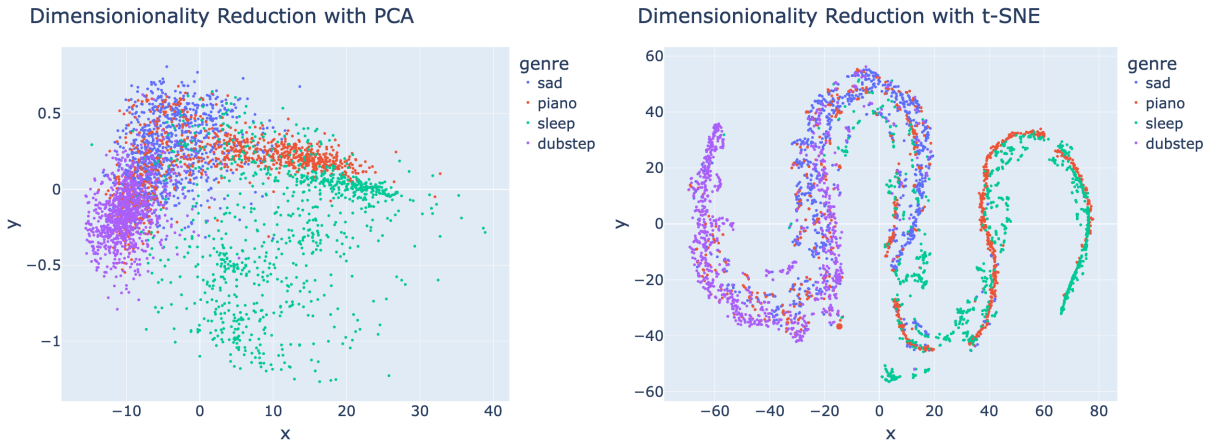

Fig. 3: Command: python3 `src/main.py -m pca tsne -g sad sleep piano dubstep`
Die Genres sleep und dubstep bilden insbesondere bei t-SNE zwei Pole, die von sad verbunden werden. Piano umspannt das ganze Spektrum.

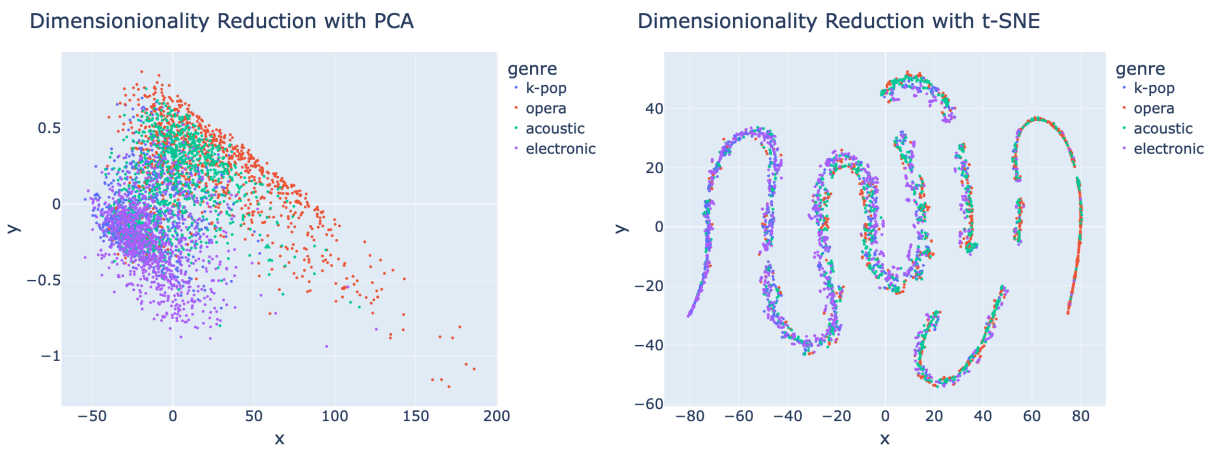

Fig. 4: Command: python3 `src/main.py -m pca tsne -g electronic acoustic opera k-pop`
Hier ist grob erkennbar, dass das Genre acoustic eine Brücke zwischen opera und electronic bildet.

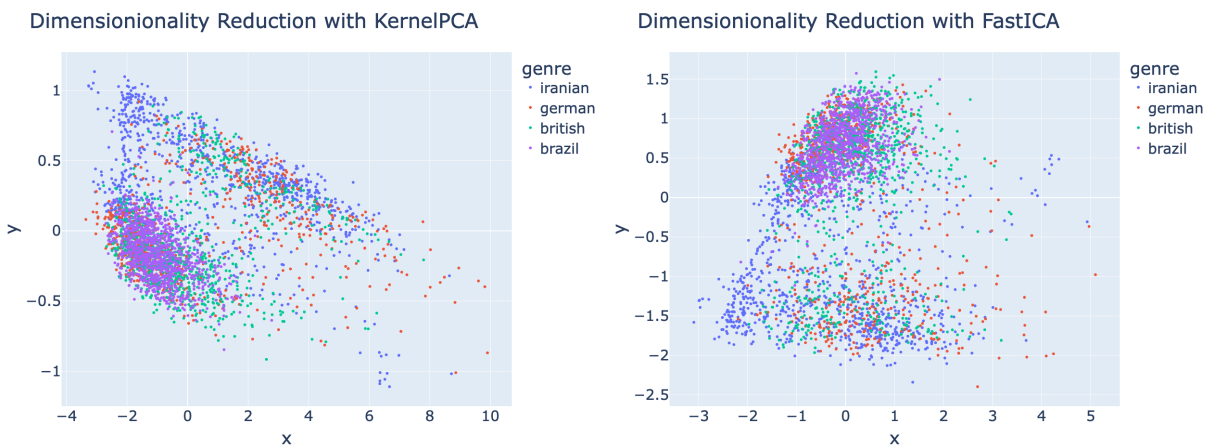

Fig. 5: Command: python3 `src/main.py -m kernelpca fastica -g iranian german brazil british`

References

[1] A. Maćkiewicz and W. Ratajczak, "Principal components analysis (PCA)," *Computers & Geosciences*, vol. 19, no. 3, pp. 303–342, 1993, doi: https://doi.org/10.1016/0098-3004(93)90090-R.

[2] M. E. Tipping and C. M. Bishop, "Mixtures of probabilistic principal component analysers," *Neural Computation 11(2)*, pp. 443–482, 1999, [Online]. Available: http://www.miketipping.com/papers/met-mppca.pdf

[3] F. Anowar, S. Sadaoui, and B. Selim, "Conceptual and empirical comparison of dimensionality reduction algorithms (PCA, KPCA, LDA, MDS, SVD, LLE, ISOMAP, LE, ICA, t-SNE)," *Computer Science Review*, vol. 40, p. 100378, 2021, doi: https://doi.org/10.1016/j.cosrev.2021.100378.

[4] "PostgreSQL - About," [Online]. Available: https://www.postgresql.org/about/

[5] "GitHub - pgvector," [Online]. Available: https://github.com/pgvector/pgvector

[6] "Psycopg 3 – PostgreSQL database adapter for Python," [Online]. Available: https://www.psycopg.org/psycopg3/docs/

[7] "scikit-learn - Machine Learning in Python," [Online]. Available: https://scikit-learn.org/stable/

[8] "Getting Started with Plotly in Python," [Online]. Available: https://plotly.com/python/getting-started/

[9] "spotify-tracks-dataset (name of original on hugging face) / Spotify Tracks Genre (name on kaggle)," [Online]. Available: original: https://huggingface.co/datasets/maharshipandya/spotify-tracks-dataset; Accessed from: https://www.kaggle.com/datasets/thedevastator/spotify-tracks-genre-dataset