Visualisierung von Musikdaten mittels t-SNE und PCA

Programmentwurf

im Studiengang Informatik

an der Dualen Hochschule Baden-Württemberg

Stuttgart

von

Robert Orbach

18.03.2025

Bearbeitungszeitraum	März 2025 bis April 2025
Matrikelnummer, Kurs	8489365, INF22A
Bei	Andreas Buckenhofer

Inhaltsverzeichnis

Abkürzungsverzeichnis

t-SNE t-distributed Stochastic Neighbor Embedding

PCA Principal Component Analysis

1 Einleitung

1.1 Hinführung zum Thema

In der heutigen datenlastigen Welt stehen wir immer häufiger vor der Herausforderung, hochdimensionale Daten zu analysieren und zu verstehen. Ein gutes Beispiel dafür ist der Bereich der Musikanalyse, in dem wir Datensätzen begegnen, die durch zahlreiche Merkmale wie Tanzbarkeit, Tempo, Lautstärke und viele weitere Attribute charakterisiert sind und darüber vorschläge erstellt werden. Diese multidimensionalen Merkmalsräume entziehen sich jedoch unserer direkten visuellen Wahrnehmung, die auf zwei oder maximal drei Dimensionen beschränkt ist.

Dimensionalitätsreduktionsmethoden wie t-distributed Stochastic Neighbor Embedding (t-SNE) [4] und Principal Component Analysis (PCA) [3] bieten einen Ausweg aus diesem Dilemma. Sie ermöglichen es uns, die wesentlichen Strukturen und Muster hochdimensionaler Datensätze in einem niedrigdimensionalen Raum darzustellen, ohne dabei kritische Informationen zu verlieren. Diese Techniken haben sich als unverzichtbare Werkzeuge in der explorativen Datenanalyse etabliert und finden besonders in der Musikinformatik breite Anwendung.awdawdawdawd

Die moderne Musikindustrie nutzt solche Techniken bereits intensiv für Empfehlungssysteme, Genreklassifikationen und die Entdeckung musikalischer Trends. Streaming-Dienste wie Spotify verwenden ähnliche Ansätze, um Benutzern personalisierte Playlists zu erstellen und neue Musik zu empfehlen, die ihren Vorlieben entspricht.

1.2 Principal Component Analysis (PCA)

Principal Component Analysis ist eine der ältesten und am häufigsten verwendeten Techniken zur Dimensionalitätsreduktion. Sie wurde bereits Anfang des 20. Jahrhunderts entwickelt und hat sich seither als Standard-Methode etabliert [5].

Mathematische Grundlage

PCA basiert auf einer linearen Transformation der Daten in ein neues Koordinatensystem, bei dem die Achsen (Hauptkomponenten) nach absteigender Varianz geordnet sind. Die erste Hauptkomponente erklärt den größten Teil der Varianz in den Daten, die zweite den zweitgrößten und so weiter.

Mathematisch gesehen sucht PCA nach den Eigenvektoren der Kovarianzmatrix der Daten:

$$\text{Cov}(X) = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})(x_i - \bar{x})^T$$

Die Eigenvektoren bilden die Hauptkomponenten, und die zugehörigen Eigenwerte geben an, wie viel Varianz durch jede Komponente erklärt wird.[5, 7]

Eigenschaften und Anwendung

PCA eignet sich besonders gut für Datensätze mit linearen Strukturen und wird oft als erster Schritt in einer Datenanalyse verwendet. Ihre Stärken liegen in der Einfachheit, Interpretierbarkeit und Skalierbarkeit:

- **Einfachheit**: PCA ist konzeptionell leicht zu verstehen und effizient zu berechnen.
- **Interpretierbarkeit**: Die Hauptkomponenten können oft semantisch interpretiert werden.
- **Rauschreduktion**: Durch Fokussierung auf die Komponenten mit der größten Varianz wird Rauschen reduziert.
- **Datenkompression**: PCA eignet sich hervorragend zur Datenkompression mit

minimalem Informationsverlust.

Im Kontext der Musikdatenanalyse kann PCA helfen, Zusammenhänge zwischen verschiedenen Merkmalen wie Tempo, Lautstärke und Tanzbarkeit zu identifizieren und Musikstücke in einem zweidimensionalen Raum zu visualisieren.

1.3 t-distributed Stochastic Neighbor Embedding (t-SNE)

t-SNE ist eine neuere, nichtlineare Dimensionalitätsreduktionstechnik, die 2008 von Laurens van der Maaten und Geoffrey Hinton entwickelt wurde [4]. Im Gegensatz zu PCA konzentriert sich t-SNE darauf, die lokale Struktur der Daten zu bewahren, was sie besonders wertvoll für die Visualisierung macht.

1.3.1 Mathematischer Hintergrund

t-SNE arbeitet in zwei Hauptschritten:

1. **Modellierung der Ähnlichkeiten im hochdimensionalen Raum**: t-SNE konstruiert eine Wahrscheinlichkeitsverteilung über Paare von Datenpunkten, sodass ähnliche Objekte eine hohe Wahrscheinlichkeit haben, als Nachbarn ausgewählt zu werden.

2. **Minimierung der Kullback-Leibler-Divergenz**: t-SNE erzeugt dann eine entsprechende Wahrscheinlichkeitsverteilung im niedrigdimensionalen Raum und versucht, die Kullback-Leibler-Divergenz zwischen den beiden Verteilungen zu minimieren:

$$\mathrm{KL}(P||Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

   Dabei verwendet t-SNE eine t-Verteilung im niedrigdimensionalen Raum, um das "Crowding Problem" zu mildern, das bei anderen Methoden auftreten kann.

[4]

1.3.2 Eigenschaften und Anwendung

t-SNE zeichnet sich durch folgende Eigenschaften aus:

- **Bewahrt lokale Strukturen**: t-SNE ist besonders gut darin, lokale Nachbarschaftsbeziehungen zu erhalten.
- **Gruppierungsfähigkeit**: t-SNE tendiert dazu, ähnliche Datenpunkte zu Clustern zusammenzufassen.
- **Nichtlinearität**: Kann komplexe, nichtlineare Muster erfassen, die PCA möglicherweise verpasst.
- **Perplexitätsparameter**: Erlaubt eine Feinabstimmung der Balance zwischen lokalen und globalen Aspekten der Daten.

In der Musikanalyse kann t-SNE verwendet werden, um Musikstücke basierend auf ihrer akustischen Ähnlichkeit zu visualisieren, was oft zu intuitiv verständlichen Clustern führt, die verschiedene Genres oder Stile repräsentieren.[2, 8]

1.4 Vektordatenbanken und pgvector

Die Verwaltung und Abfrage hochdimensionaler Vektordaten stellt besondere Anforderungen an Datenbanksysteme. Herkömmliche relationale Datenbanken sind nicht optimal für die effiziente Ähnlichkeitssuche in Vektorräumen konzipiert. Hier kommen spezialisierte Vektordatenbanken ins Spiel.

Was ist pgvector?

pgvector ist eine Erweiterung für das populäre PostgreSQL-Datenbanksystem, die es

ermöglicht, Vektoren effizient zu speichern und Ähnlichkeitsabfragen durchzuführen [6]. Es unterstützt verschiedene Distanzmetriken wie euklidische Distanz, Kosinus-Ähnlichkeit und Taxicab-Distanz.

Die Hauptfunktionen von pgvector umfassen:

- Speicherung von dichten Vektoren unterschiedlicher Dimension.
- Effiziente Ähnlichkeitssuche mit verschiedenen Metriken.
- Indizierungsmethoden für beschleunigte Abfragen.
- Integration in das SQL-Ökosystem von PostgreSQL.

Die Kombination von pgvector mit Dimensionalitätsreduktionstechniken wie PCA und t-SNE bietet leistungsstarke Werkzeuge für die Analyse und Visualisierung von Musikdaten.

2 Installation

2.1 Setup der PostgreSQL-Datenbank mit pgvector

Für die Speicherung und Analyse von Musikdaten verwenden wir eine PostgreSQL-Datenbank mit der `pgvector`-Erweiterung[6]. Diese ermöglicht die effiziente Speicherung und Abfrage hochdimensionaler Vektoren. Im Folgenden wird beschrieben, wie die Datenbank mit Docker eingerichtet wird.

2.1.1 Docker-Setup

Wir verwenden Docker[1], um die PostgreSQL-Datenbank mit der `pgvector`-Erweiterung zu starten. Die Konfiguration erfolgt über die folgende `docker-compose.yml`-Datei:

```
version: '3'
services:
  postgres:
    image: ankane/pgvector:latest
    container_name: pgvector_spotify
    environment:
      POSTGRES_DB: music_vectors_db
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: postgres123
    ports:
      - "5432:5432"
    volumes:
      - pgdata:/var/lib/postgresql/data
    command: >
      postgres -c max_connections=100 -c shared_buffers=256MB
    restart: unless-stopped

volumes:
```

```
  pgdata:
```

Skript 2.1: Unkomplizierte docker-compose.yml Datei

Schritte zur Ausführung:

1. Speichern Sie die obige Konfiguration in einer Datei mit dem Namen `docker-compose.yml`.

2. Starten Sie den PostgreSQL-Container mit dem Befehl:

   ```
   docker-compose up -d
   ```

3. Überprüfen Sie, ob der Container läuft:

   ```
   docker ps
   ```

2.1.2 Initialisierung der Datenbank

Nach dem Start der PostgreSQL-Datenbank muss die `pgvector`-Erweiterung[6] aktiviert, die Datenbanktabelle und ein Index auf dieser zur beschleunigung von Anfragen eingerichtet werden. Dafür wird das `init.sql` Skript verwendet:

```
-- Tabelle fuer Musikdaten erstellen
CREATE TABLE music_vectors (
    id SERIAL PRIMARY KEY,
    features VECTOR(14)  -- 14-dimensionaler Vektor
);

-- IVFFLAT-Index fuer schnelle Nearest-Neighbor-Suchen erstellen
CREATE INDEX music_vector_idx ON music_vectors
USING ivfflat (features vector_l2_ops) WITH (lists = 100);
```

Skript 2.2: Ausschnitt aus init.sql

Dieses Skript erstellt eine Tabelle `music_vectors` mit einer Spalte `vector` mit einer Kapazität von 14 Dimensionalen Vektoren, der Anzahl Merkmale im Datensatz. Um spätere abfragen zu beschleunigen erstellen wir außerdem einen IVFFLAT-Index um die Ähnlichkeitssuche zu beschleunigen.

Speichern Sie die angehängte Vollversion der SQL-Datei als `init.sql`, die Ausführung wird durch das Hautprogramm, dem angehängten `jupyter notebook`, durchgeführt.

2.2 Herunterladen der genutzten Musikdaten

Die Musikdaten, die in diesem Projekt verwendet werden, sind in der `spotify_dataset.csv`-Datei enthalten, welche von Kaggle heruntergeladen wurde. Diese Datei enthält Eigenschaften von verschiedenen Musikstücken, die Informationen wie die Lautstärke, die Energie, die Tanzbarkeit und die Dauer enthalten.

2.3 Installation der Python-Bibliotheken

Eine python Installation ist notwendig um die Daten zu verarbeiten und die Analyse durchzuführen. Die benötigten Bibliotheken sind in der `requirements.txt`-Datei aufgelistet und können mit dem folgenden Befehl installiert werden:

```
pip install -r requirements.txt
```

2.4 Ergebnis der Installation

Nachdem diese Schritte abgeschlossen sind, sollte das Dateisystem wie folgt aussehen:

```
D:.
  |
  +-- compose.yml
  +-- init.sql
  +-- requirements.txt
```

```
+-- spotify_dataset.csv
+-- Visualization.ipynb
```

Skript 2.3: Dateisystem nach Installation

3 Beispielumsetzung

Nun, da alles bereit ist, könen wir die PCA und t-SNE Ansätze zur Dimensionsreduktion von Musikdaten vergleichen. Zuerst werden wir den Datensatz in die Datenbank laden und die Daten normalisieren. Dann wenden wir zuerst die PCA und dann t-SNE auf die Musikdaten an und analysieren die Ergebnisse.

3.1 Datenvorbereitung

Bevor die Dimensionsreduktion durchgeführt werden kann, müssen die Musikdaten zunächst in die Datenbank geladen und anschließend normalisiert werden.

Normalisierung der Daten

Die Merkmale der Musikdaten, wie Lautstärke, Energie, Tanzbarkeit und Dauer, liegen auf unterschiedlichen Skalen vor. Um eine konsistente Analyse zu ermöglichen, ist eine Normalisierung der Daten erforderlich. Hierfür wird der StandardScaler aus der Python-Bibliothek sklearn verwendet, der die Daten so transformiert, dass sie eine Standardnormalverteilung mit einem Mittelwert von 0 und einer Standardabweichung von 1 aufweisen.

Datenimport in die Datenbank

Die Musikdaten aus der Datei `spotify_dataset.csv` werden in die PostgreSQL-Datenbank importiert. Dazu wird die Tabelle `music_vectors` mittels der `init.sql` Datei erstellt, die die Merkmale der Musikstücke als Vektoren speichert. Der folgende Python-Code zeigt, wie die Daten in die Datenbank eingefügt werden:

```
conn = connect_to_postgres()
execute_values(
    cursor,
    "INSERT INTO music_vectors (features) VALUES %s",
    data_to_insert,
```

```
    template="(%s)"
)
conn.commit()
```

Skript 3.1: Insert der Musikdaten in die Datenbank

Nach dem erfolgreichen Import stehen die Daten für die weitere Verarbeitung und Analyse bereit.

3.2 Dimensionalitätsreduktion mit PCA

Die Principal Component Analysis (PCA) ist eine Methode zur Dimensionsreduktion, die darauf abzielt, die Hauptvariationsrichtungen in den Daten zu identifizieren. Sie reduziert die Anzahl der Dimensionen, indem sie die Daten auf eine kleinere Anzahl von Hauptkomponenten projiziert, die die maximale Varianz erklären.

Anwendung von PCA

Das folgende Python-Skript zeigt, wie PCA mit hilfe der sklearn lib auf die normalisierten Daten angewendet wird:

```
from sklearn.decomposition import PCA
# Perform PCA
pca = PCA(n_components=2)
pca_result = pca.fit_transform(features_standardized)
```

Skript 3.2: PCA-Analyse der Musikdaten

Dabei ergibt sich die `pca_result` als zweidimensionales Array, das die reduzierten Daten enthält. Bei dem genutzten Beispieldatensatz ergibt sich eine Varianz von 31% für die erste und 13% für die zweite Hauptkomponente. Damit konnten rund 45% der Varianz im vorliegenden Datensatz abgedeckt werden.

Dies zeigt sich in der Visualisierung der Dimensionsreduktion in der die Datenpunkte der Klassen zwar in zwei Regionen liegen, diese jedoch aneinander grenzen.

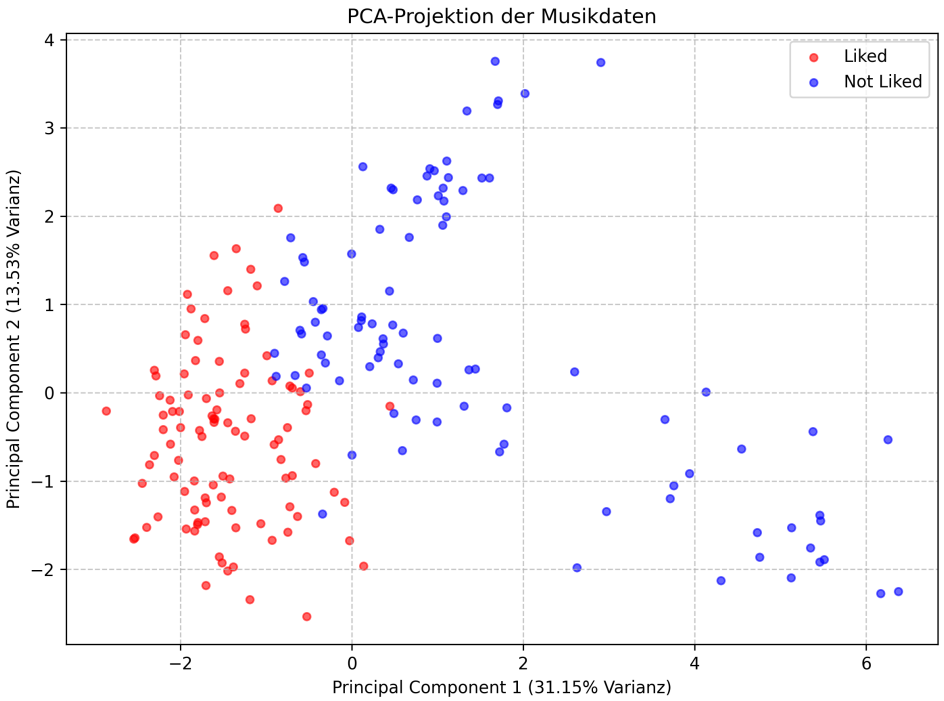

Abbildung 3.1: PCA der Musikdaten

Ohne die Einfärbung wäre die Unterteilung in Cluster nicht eindeutig zu erkennen. Damit wäre eine Klassifizierung neuer Datenpunkte abhängig von der Position nicht zuverlässig möglich. Für Songs, die stark in einem roten/blauen Bereich liegen, wäre eine Klassifizierung jedoch möglich.

Was der Grund für eine solche Unterteilung ist, kann durch die Analyse der Hauptkomponenten ermittelt werden. Die PCA-Loadings geben an, wie stark jedes Feature zu den Hauptkomponenten beiträgt. Sie sind ein entscheidender Bestandteil der PCA-Analyse, da sie helfen, die Bedeutung der einzelnen Features für die Hauptkomponenten zu interpretieren. Die folgende Abbildung zeigt die Beiträge der Features zu den beiden Hauptkomponenten:

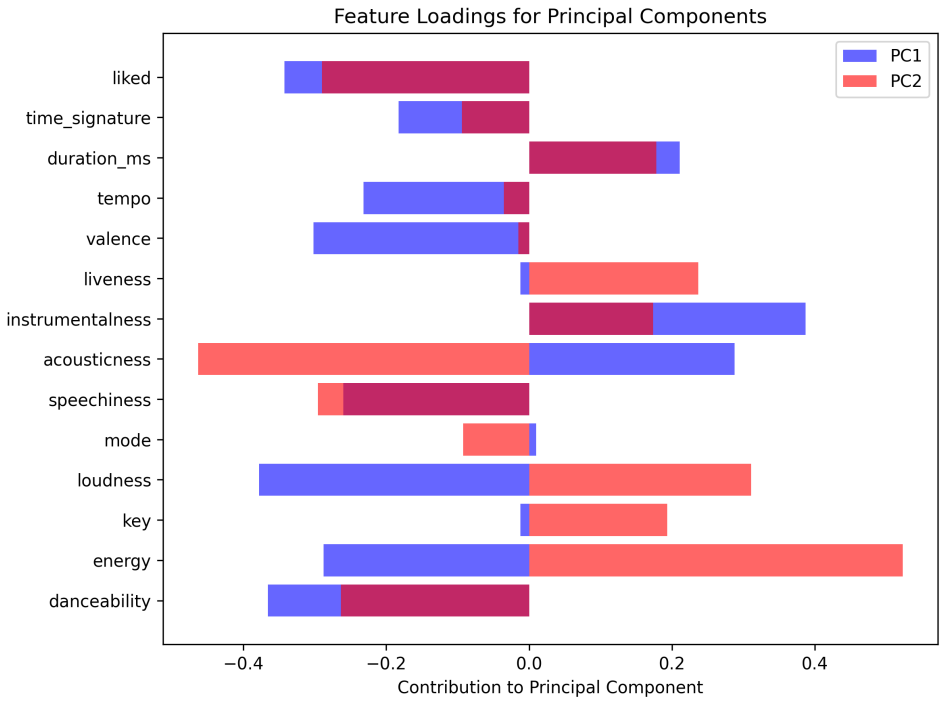

Abbildung 3.2: Erklärte Varianz der Hauptkomponenten

Interpretation der PCA-Loadings

Die Balkendiagramme zeigen die Richtung und Stärke der Beiträge der einzelnen Features zu den Hauptkomponenten. Positive/negative Werte geben an, ob ein Feature positiv/negativ mit der jeweiligen Hauptkomponente korreliert. Die Länge der Balken zeigt die Stärke des Beitrags an.

In PC1 sind somit besonders die Features `instrumentalness` und `acousticness` stark positiv vertreten. In PC2 sind es `energy`, `loudness` und `lieveness`.

3.3 Dimensionalitätsreduktion mit t-SNE

Da t-SNE eine weitere Methode zur Dimensionsreduktion mit dem fokus auf die Erhaltung von lokalen Strukturen ist, wenden wir diese auf die normalisierten Daten an.

Anwendung von t-SNE

```
from sklearn.decomposition import PCA
# Perform t-SNE
tsne = TSNE(n_components=2, perplexity=perplexity, max_iter=1000)
tsne_result = tsne.fit_transform(features_standardized)
```

Skript 3.3: t-SNE-Analyse der Musikdaten

Bei der t-SNE-Analyse wird die Perplexität als neuen Parameter eingeführt. Dieser Wert gibt an, wie viele Nachbarn für die Berechnung der Wahrscheinlichkeitsverteilung berücksichtigt werden sollen. Ein hoher Wert führt zu einer stärkeren Betonung globaler Strukturen, während ein niedriger Wert lokale Strukturen bevorzugt.

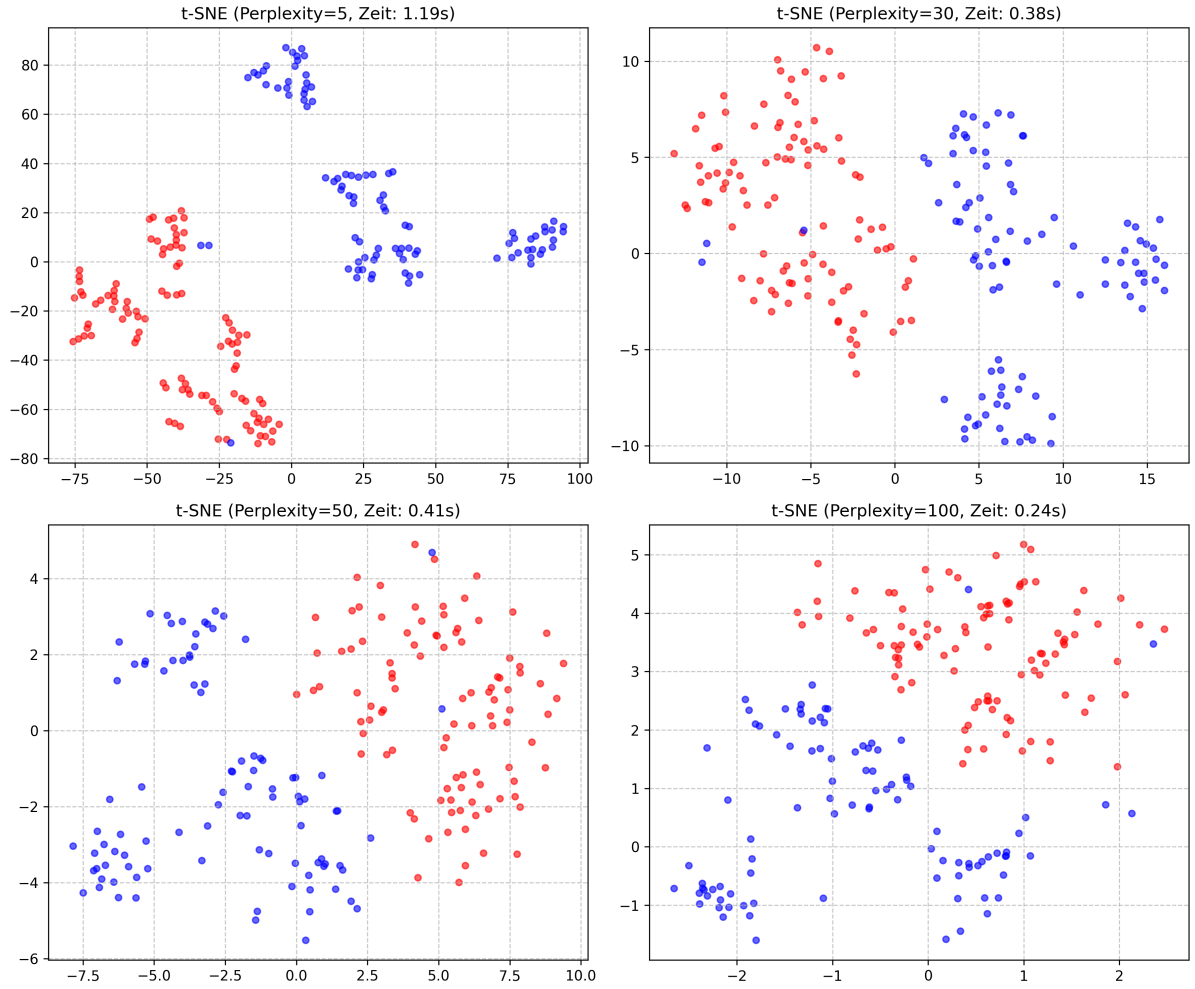

Abbildung 3.3: Verschiedene Perplexitätswerte für t-SNE

Als optimalen Wert für die Perplexität wird 30 gewählt, da dieser relativ gut getrennte Cluster zeigt, und Probleme mit hohen Werten wie das Zusammenfassen von Clustern verhindert.

3.4 Vergleich von PCA und t-SNE

Nach der Anwendung beider Dimensionsreduktionstechniken auf unseren Musikdatensatz können wir nun einen fundierten Vergleich zwischen PCA und t-SNE ziehen, sowohl aus theoretischer als auch aus praktischer Sicht.

Unterschiede in der Methodik

Die grundlegenden methodischen Unterschiede zwischen PCA und t-SNE spiegeln sich deutlich in den Ergebnissen wider:

- **PCA** ist ein lineares Verfahren, das auf die Maximierung der Varianz abzielt. Es identifiziert Richtungen im Datenraum, in denen die Daten am stärksten variieren, und projiziert die Daten auf diese Hauptkomponenten. In unserem Fall konnten durch die ersten beiden Hauptkomponenten etwa 45% der Gesamtvarianz erklärt werden.

- **t-SNE** hingegen ist ein nichtlineares Verfahren, das lokale Strukturen und Nachbarschaftsbeziehungen in den Daten bewahrt. Es modelliert die Ähnlichkeiten zwischen Datenpunkten und versucht, diese im niedrigdimensionalen Raum zu erhalten. Die Perplexität ist dabei ein kritischer Parameter, der die Anzahl der berücksichtigten Nachbarn steuert.

Visuelle Interpretation der Ergebnisse

Die Visualisierungen zeigen deutliche Unterschiede:

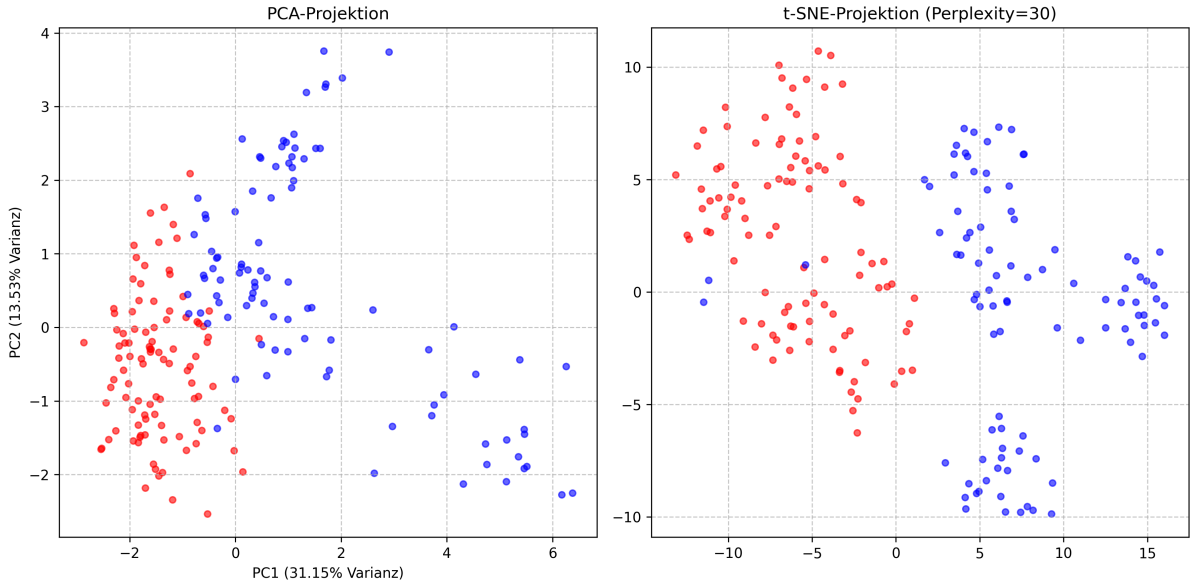

Abbildung 3.4: Vergleich der Dimensionsreduktionsmethoden

Bei der **PCA-Visualisierung** sehen wir:

- Eine graduelle Trennung der Musikgenres, jedoch mit Überlappungen
- Die Datenpunkte folgen einer linearen Struktur, die durch die Hauptkomponenten vorgegeben wird
- Die Interpretierbarkeit der Achsen ist durch die Loadings gegeben – wir können erkennen, dass PC1 stark mit Instrumentalität und Akustik korreliert, während PC2 mit Energie und Lautstärke zusammenhängt

Bei der **t-SNE-Visualisierung** mit optimaler Perplexität (30) zeigt sich:

- Eine deutlichere Clusterbildung der Musikgenres
- Stärkere lokale Strukturen, die in der PCA nicht sichtbar sind
- Keine direkte Interpretierbarkeit der Achsen, da t-SNE keine direkten linearen

Transformationen der ursprünglichen Features darstellt

3.5 Anwendungsszenarien der Verfahren

Basierend auf unserer Analyse eignen sich die beiden Methoden für unterschiedliche Szenarien:

PCA ist vorzuziehen, wenn:

- Die interpretierbare Darstellung der Dimensionen wichtig ist
- Der Fokus auf der Erhaltung der globalen Datenstruktur liegt
- Eine schnelle Berechnung erforderlich ist (PCA ist deutlich weniger recheninten-siv)
- Die Daten auf lineare Beziehungen untersucht werden sollen
- Neue Datenpunkte einfach in den reduzierten Raum projiziert werden müssen, ohne die gesamte Transformation neu zu berechnen

t-SNE ist vorzuziehen, wenn:

- Die Visualisierung und Entdeckung von Clustern im Vordergrund steht
- Lokale Strukturen und nichtlineare Beziehungen erhalten werden sollen
- Die Interpretierbarkeit der Dimensionen weniger wichtig ist als die Qualität der Visualisierung
- Ausreichend Rechenkapazität für die iterative Optimierung zur Verfügung steht

Auswirkungen auf das Musikempfehlungssystem

Für ein Musikempfehlungssystem haben beide Methoden Vor- und Nachteile:

1. **Mit PCA** könnten wir ein interpretierbares Modell erstellen, das beispielsweise erklären kann, warum bestimmte Songs empfohlen werden (z.B. "dieser Song hat ähnliche akustische Eigenschaften wie die, die Sie mögen"). Die lineare Transformation ermöglicht zudem eine effiziente Integration neuer Songs in das bestehende System.

2. **Mit t-SNE** könnten wir feinere Unterschiede zwischen Musikstücken erfassen und besser abgegrenzte Genre-Cluster identifizieren. Dies könnte zu präziseren Empfehlungen führen, insbesondere für Nutzer mit spezifischen Musikvorlieben.

Eine optimale Lösung könnte die Kombination beider Methoden sein: PCA zur initialen Dimensionsreduktion und Entfernung von Rauschen in den Daten, gefolgt von t-SNE zur Feinabstimmung der Visualisierung und Clusteridentifikation.

Effizienz und Skalierbarkeit

Ein wichtiger praktischer Aspekt ist die Effizienz und Skalierbarkeit der Methoden:

- **PCA** hat eine Zeitkomplexität von $O(\min(n^2p, np^2))$, wobei n die Anzahl der Datenpunkte und p die Anzahl der Dimensionen ist. In unserem Fall mit etwa 200 Songs und weniger als 20 Features ist die Berechnung in Sekundenbruchteilen abgeschlossen.

- **t-SNE** hat eine deutlich höhere Zeitkomplexität von $O(n^2)$ und benötigt in unserem Fall bereits auf dem keinen Datensatz mit 1000 Iterationen mehrere Sekunden zur Berechnung.

Für ein praktisches Musikempfehlungssystem mit Millionen von Songs wäre PCA daher möglicherweise die effizientere Wahl für die Echtzeit-Verarbeitung, während t-SNE für Offline-Analysen und tiefere Einblicke in die Datenstruktur verwendet werden könnte.

Fazit

Die Wahl zwischen PCA und t-SNE hängt stark vom spezifischen Anwendungsfall und

den Prioritäten des Projekts ab. Für ein Musikempfehlungssystem ergibt sich aus der Analyse, dass PCA durch seine Interpretierbarkeit und Effizienz Vorteile für die Basis-Infrastruktur bietet, während t-SNE wertvolle Einblicke für die Feinabstimmung und Visualisierung liefern kann.

In der praktischen Umsetzung könnte ein hybrider Ansatz, der die Stärken beider Methoden kombiniert, die vielversprechendste Lösung darstellen, um sowohl effiziente als auch hochwertige Musikempfehlungen zu generieren.

Literaturverzeichnis

[1] *Docker - Build, Share, and Run Any App, Anywhere*. o.J. URL: https://www.docker.com/ (besucht am 22. 03. 2015).

[2] *How to Use t-SNE Effectively*. o.J. URL: https://distill.pub/2016/misread-tsne/ (besucht am 22. 03. 2015).

[3] Ian T. Jolliffe. *Principal Component Analysis*. Springer, 2002.

[4] Laurens van der Maaten und Geoffrey Hinton. „Visualizing data using t-SNE“. In: *Journal of Machine Learning Research* 9.Nov (2008), S. 2579–2605.

[5] Karl Pearson. „On lines and planes of closest fit to systems of points in space“. In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11 (1901), S. 559–572.

[6] *Pgvector - Open-source vector similarity search for Postgres*. o.J. URL: https://github.com/pgvector/pgvector (besucht am 22. 03. 2015).

[7] Jonathon Shlens. „Tutorial on Principal Component Analysis“. In: (2005). URL: https://www.cs.cmu.edu/~elaw/papers/pca.pdf.

[8] *t-SNE: The effect of various perplexity values on the shape*. o.J. URL: https://scikit-learn.org/stable/auto_examples/manifold/plot_t_sne_perplexity.html (besucht am 22. 03. 2015).