

データサイエンス講座

Level 01-7

第7章：教師なし学習と次元削除テクニック

第7章：教師なし学習と次元削除テクニック

教師なし学習の基礎

重要性とメリット

教師なし学習は、ラベル付けされていないデータからパターンや構造を自動的に抽出する技術で、従来の分析手法が前提条件や人間の仮説に依存するのに対し、未知のデータの中に隠れた関係性やグループを発見することが可能です。これにより、データの次元削減やクラスタリングを活用して、高次元データの処理効率を向上させるだけでなく、顧客セグメンテーションや異常検知など、ビジネスや研究の新たな可能性を切り開きます。

クラスタリング基礎

クラスタリングとは

ラベルが付いていないデータを「似た特徴を持つグループ」に分け、データの構造やパターンを発見する教師なし学習の手法で、顧客セグメント化や異常検知など幅広い分野で活用されます。

K-means

データをあらかじめ指定したクラスタ数に分けるアルゴリズムで、各クラスタの中心点を計算しながらデータを割り当て直す手法です。
マーケティングにおける顧客セグメント化や、画像圧縮で似た色をまとめる場面で応用されます。

改装型

データを階層的に分けることで、クラスタの構造を可視化するアルゴリズムです。
木構造（デンドログラム）を利用し、柔軟にクラスタ数を選べるため、遺伝子データ解析や文書のトピック分類など、階層的関係が重要な場面で使われます。

DBSCAN

データの密度に基づいてクラスタを形成し、ノイズや外れ値を除外できるアルゴリズムです。
非球状のクラスタやノイズの多いデータにも対応できるため、地理データを用いた領域分析や異常検知（例: 不正取引の検出）で応用されます。

クラスタ数決定方法

クラスタ数決定意義

クラスタ数の決定は、クラスタリングの結果に直接影響を与える重要な要素で、適切なクラスタ数を選ぶことで、データの特徴を正確に捉え、過剰分割や不十分な分割を防ぐことができます。例えば、エルボー法やシルエットスコアを使うことで、データ構造に最も適したクラスタ数を見つけることが可能です。

エルボー法

クラスタ数を増やすと、誤差（二乗和）が減少するが、減少率が急激に低下する点（エルボーポイント）をクラスタ数に選ぶ。

シルエットスコア

クラスタリングの結果の良さを評価する指標。値が1に近いほど良好。

次元削除基礎

DS Lv.1-7

次元削減は、高次元データの冗長性を排除し、計算効率を向上させるとともに、データの構造やパターンを視覚的に理解しやすくする重要な手法であり、モデルの精度向上や過学習の防止にも寄与します。

PCA（主成分分析）

データの分散が最大になる方向（主成分）を計算し、高次元データを少数の次元に射影する線形次元削減手法で、計算効率が高く、データ構造を簡潔に把握できます。
応用例: 顧客セグメントの分析や特徴量選択。

t-SNE

t-SNEは、高次元データの局所的な構造を維持しながら、非線形次元削減を行う手法で、視覚化に特化しています。ただし、計算コストが高く、大規模データには不向きです。
応用例: 画像データや遺伝子データのクラスタ可視化。

UMAP

UMAPは、非線形次元削減を効率的に行う手法で、t-SNEより高速かつ大規模データに適用可能です。局所的・大域的構造の両方を保持できるため、視覚化とクラスタリングに適しています。
応用例: 医療データや自然言語処理データの可視化。

LDA

LDAは、クラス間の分離を最大化する線形次元削減手法で、分類タスクに特化しています。教師あり学習に使用され、データの識別力を高めます。
応用例: 文書分類や顔認識。

演習：顧客セグメンテーション



学習のポイント

1. 教師なし学習の理解: 正解ラベルがないデータから、パターンやグループを見つける面白さを体感。
2. クラスタリングの実務応用: 顧客セグメンテーションやパーソナライズ戦略への応用可能性を学ぶ。
3. 次元削減の役割: 高次元データを視覚化して、クラスタの分布を直感的に把握する。
4. データ解釈の重要性: 各クラスタの特性を分析し、ビジネスへの具体的な応用アイデアを考察。



動作の流れ

1. データの作成-顧客の年齢、収入、消費スコアをランダム生成し、クラスタリング対象のデータセットを作成。
2. データの標準化-各特徴量を標準化して、クラスタリングに適したスケールに調整。
3. クラスタリングの実行-k-meansクラスタリングでデータを4つのクラスタに分割。
4. 次元削減と可視化-PCAでデータを2次元に削減し、クラスタリング結果を散布図で視覚化。
5. クラスタごとの分析-各クラスタの平均特性を計算し、セグメントごとの特徴を明らかにする。

7-01.py

```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt

# 1. ダミーデータの作成（顧客データ）
np.random.seed(42)
data = {
    "Age": np.random.randint(18, 65, 200),
    "Income": np.random.randint(20000, 120000, 200),
    "Spending_Score": np.random.randint(1, 100, 200)
}
df = pd.DataFrame(data)

# 2. データの標準化
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df)
...

```


演習：センサーデータ異常検知



学習のポイント

1. DBSCANの動作原理を理解: 密度に基づくクラスタリングのメリットと、異常値検出への応用可能性を学ぶ。
2. 異常検知の実践力を養う: ノイズを効果的に検出するDBSCANの特性を確認。
3. クラスタリング手法の選択スキル: 非球状のクラスターやノイズを含むデータへのDBSCANの適用性を理解。



動作の流れ

1. センサーデータの作成-正常値を基にセンサーデータを生成し、ランダムに異常値を追加。
2. DBSCANの適用 -データ密度を基にクラスターを形成し、異常値（ノイズ）を検出。
3. クラスタリング結果の可視化 - 各クラスターを散布図でプロットし、異常値をハイライト。
4. 異常値のリストアップ - 異常値を含むデータを抽出し、出力。

7-02.py

```
import numpy as np
import pandas as pd
from sklearn.cluster import DBSCAN
import matplotlib.pyplot as plt

# 1. ダミーセンサーデータの作成
np.random.seed(42)
sensor_data = {
    "Sensor_1": np.random.normal(50, 5, 300),
    "Sensor_2": np.random.normal(60, 5, 300)
}

# Adding anomalies
anomalies = {
    "Sensor_1": np.random.uniform(80, 100, 10),
    "Sensor_2": np.random.uniform(80, 100, 10)
}

sensor_data["Sensor_1"] = np.concatenate([sensor_data["Sensor_1"],
anomalies["Sensor_1"]])
sensor_data["Sensor_2"] = np.concatenate([sensor_data["Sensor_2"],
anomalies["Sensor_2"]])
. . .
```


演習：遺伝子データ分析



学習のポイント

1. 遺伝子データの理解: 遺伝子間の類似性を視覚化し、階層構造を発見する能力。
2. デンドログラムの解釈スキルを習得: クラスタ間の距離やグループ化の階層的な関係を読み取る方法を学ぶ。
3. クラスタリング結果を応用: 各クラスタの生物学的意味を解釈し、実際の研究や応用に結びつける。
4. 階層型クラスタリングの柔軟性: 遺伝子データだけでなく、その他のデータセット（テキスト、音声など）にも応用可能な手法であることを理解。



動作の流れ

1. データの準備 -各遺伝子の発現量を含むデータセットを読み込み。
2. データの前処理 -データを標準化し、スケールの違いを調整。
3. 階層型クラスタリングの適用 -遺伝子間の相関や距離を計算し、Ward法で階層型クラスタリングを実行。
4. デンドログラムの作成 -遺伝子間の類似度に基づくデンドログラムを作成し、グループ化の構造を可視化。
5. クラスタの分割 -デンドログラムを基に適切なクラスタ数を選択し、遺伝子をグループに分割。
6. 各クラスタの特徴分析 -グループごとの遺伝子機能や役割を調べ、病気や症状との関連を考察。

7-03.py

```
import pandas as pd
import numpy as np
from scipy.cluster.hierarchy import dendrogram, linkage, fcluster
import matplotlib.pyplot as plt

# 1. ダミー遺伝子発現データの作成
np.random.seed(42)
genes = [f"Gene_{i}" for i in range(1, 51)] # 50遺伝子
samples = [f"Sample_{i}" for i in range(1, 21)] # 20サンプル

data = np.random.rand(20, 50) * 100 # 遺伝子発現量データ（ランダム値）
df = pd.DataFrame(data, index=samples, columns=genes)

# 2. 階層型クラスタリングの適用
linked = linkage(df.T, method='ward') # 遺伝子間でクラスタリング

# 3. デンドログラムのプロット
plt.figure(figsize=(12, 8))
dendrogram(linked, labels=genes, leaf_rotation=90, leaf_font_size=10)
plt.title('Dendrogram of Gene Expression Data')
plt.xlabel('Genes')
...

```


DS Lv.1-7

お疲れ様でした

info@mjin.space

mjin.space