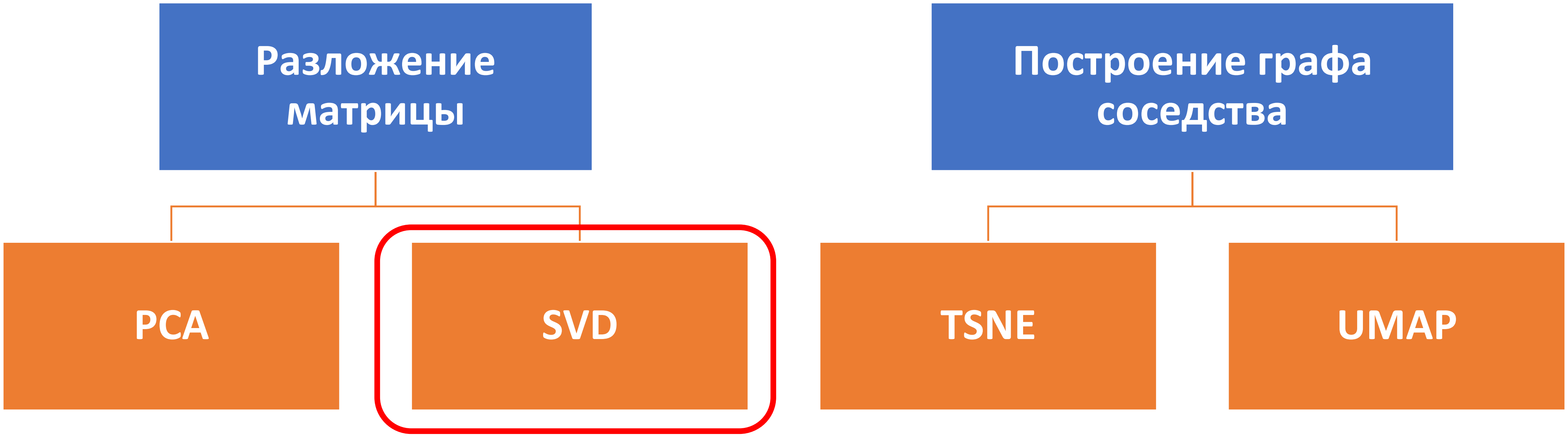
Анализ данных и машинное обучение, ч. 2

Лекция 4. Обучение без учителя. Факторный анализ. Метод главных компонент. Диагностика и оценка результатов

Киреев В.С.,
к.т.н., доцент

Москва, 2025

Обучение без учителя. Методы сокращения размерности. Извлечение признаков



Метод сингулярного разложения (SVD)

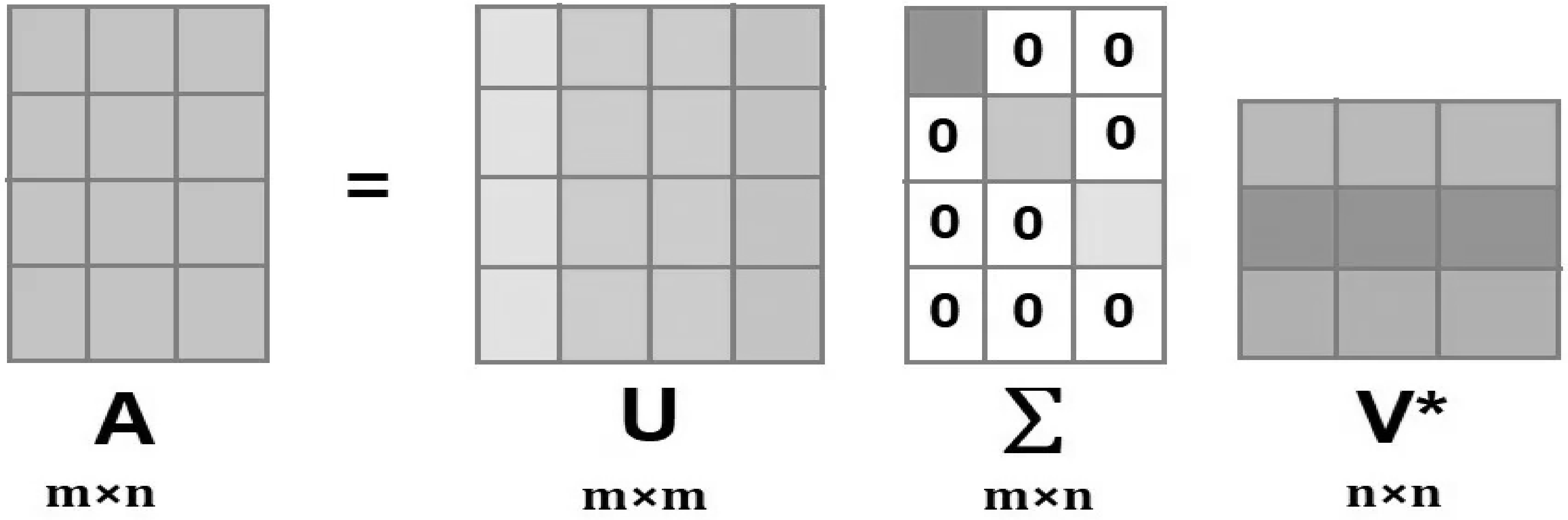
Метод сингулярного разложения (SVD)— это подход к разложению матриц, который помогает сократить матрицу путем обобщения собственного разложения квадратной матрицы (с одинаковым количеством столбцов и строк) на любую матрицу.

SVD широко используется как при вычислении других матричных операций, таких как обращение матрицы, так и в качестве метода сокращения данных в машинном обучении. SVD также можно использовать для линейной регрессии методом наименьших квадратов, сжатия изображений и шумоподавления данных.

SVD. Пример на Python

```
import numpy as np
from scipy.linalg import svd
# Создадим случайную матрицу
np.random.seed(42)
A = np.random.randn(5, 3) * 10
print("Исходная матрица A:\n", A)
# Вычисляем SVD
U, s, Vt = svd(A)
print("\nЛевые сингулярные векторы U:\n", U)
print("\nСингулярные значения s:\n", s)
print("\nПравые сингулярные векторы V^T:\n", Vt)
# Проверка восстановления
Sigma = np.zeros((A.shape[0], A.shape[1]))
np.fill_diagonal(Sigma, s)
A_reconstructed = U @ Sigma @ Vt
print("\nВосстановленная матрица:\n", A_reconstructed)
print("\nОшибка восстановления:", np.linalg.norm(A - A_reconstructed))
```

Метод сингулярного разложения (SVD)



$$\mathbf{A} = \mathbf{USV}^{\top}$$

Матрица U

Столбцы U являются ортонормированными собственными векторами AAT и называются левыми сингулярными векторами матрицы A.

$$U = A*A^T$$

Матрица V

Столбцы V являются ортонормированными собственными векторами ATA и называются правыми сингулярными векторами матрицы A.

$$V = A^T A$$

Матрица Sigma

Матрица Σ – это диагональная матрица, значения которой являются квадратными корнями собственных значений матрицы U или V в порядке убывания. Эти диагональные элементы матрицы Σ называются сингулярными значениями.

Метод Грэма-Шмидта

Метод Грэма-Шмидта — это способ ортогонализации системы линейно-независимых векторов. Суть метода Грэма-Шмидта состоит во взятии первого ортогонального вектора равным первому исходному вектору и построении каждого нового ортогонального вектора равным текущему исходному вектору, скорректированному на величины проекций текущего вектора на предыдущие ортогональные векторы.

Метод Грэма-Шмидта. Процесс

Первый вектор строящегося базиса выбираем так:

1. $\mathbf{b}_0 = \mathbf{a}_0$ - так выбираем первый вектор строящегося базиса.
2. $\mathbf{e}_{b_0} = \dfrac{\mathbf{b}_0}{\|\mathbf{b}_0\|}$ - нормируем вектор $\mathbf{b_0}$
3. $\mathbf{b}_1 = \mathbf{a}_1 - \langle \mathbf{a_1}, \mathbf{e_{b_0}} \rangle \cdot \mathbf{e_{b_0}}$, где $\langle \mathbf{a_1}, \mathbf{e_{b_0}} \rangle \cdot \mathbf{e_{b_0}}$ - проекция вектора $\mathbf{a_1}$ на вектор $\mathbf{e_{b_0}}$, вдоль нормированного вектора $\mathbf{e_{b_0}}$.
4. $\mathbf{b}_2 = \mathbf{a}_2 - \langle \mathbf{a}_2, \mathbf{e_{b_0}} \rangle \cdot \mathbf{e_{b_0}} - \langle \mathbf{a}_2, \mathbf{e_{b_1}} \rangle \cdot \mathbf{e_{b_1}}$
5. $\mathbf{b}_i = \mathbf{a}_i - \sum\limits_{k=0}^{i-1} \langle \mathbf{a}_i, \mathbf{e_{b_k}} \rangle \cdot \mathbf{e_{b_k}}$ - в общем виде.

Двумерные диаграммы, биплоты (biplots)

Двумерные диаграммы, биплоты (biplots) — это тип исследовательского графика, используемый в статистике, обобщение простой диаграммы рассеяния с двумя переменными. Биплот накладывает график оценки на график загрузки. Биплот позволяет графически отображать информацию как о выборках, так и о переменных матрицы данных.

Объекты выборки отображаются в виде точек, а переменные — в виде векторов, линейных осей или нелинейных траекторий. В случае категориальных переменных, точки уровня категории могут использоваться для представления уровней категориальной переменной. Обобщенный биплот отображает информацию как о непрерывных, так и о категориальных переменных.

Двумерные диаграммы, биплоты (biplots). Отображение главных компонент

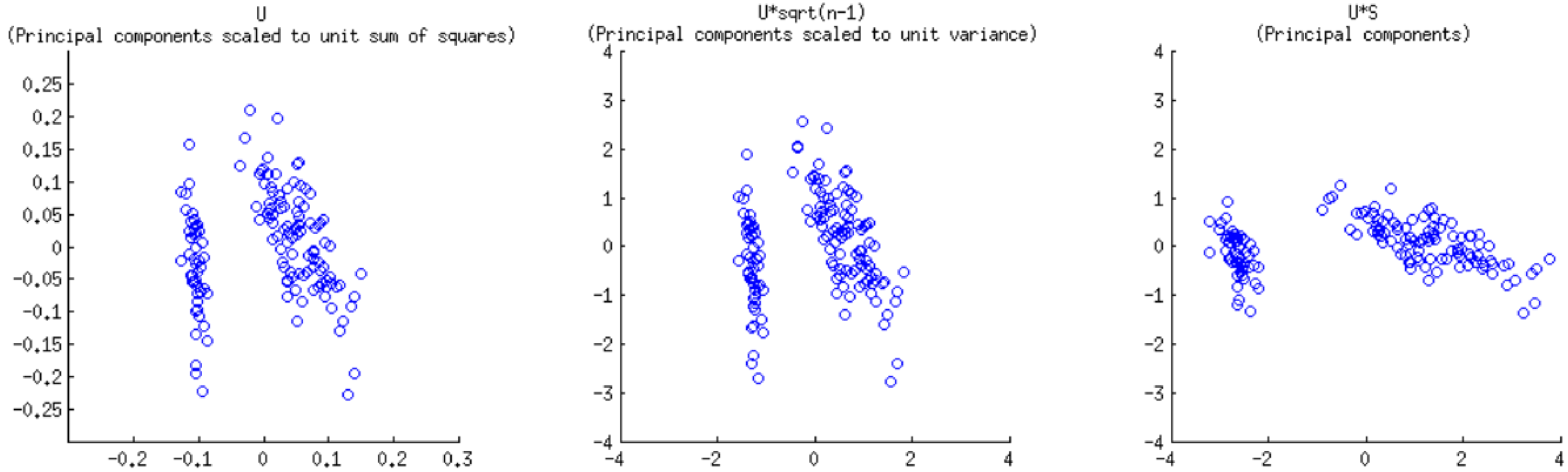
Предполагая, что исходная матрица может быть представлена в форме:

$$\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^{\top}$$

На биплоте PCA две первые главные компоненты отображаются в виде графика рассеяния, т. Е. Первый столбец U строится против его второго столбца. Но нормализация может быть разной; например можно использовать:

- Столбцы U - это главные компоненты, приведенные к единице суммы квадратов;
- Столбцы $\sqrt{n-1}\mathbf{U}$ - это стандартизированные основные компоненты (дисперсия);
- Столбцы $\mathbf{US}$ - это «сырые» главные компоненты (проекции на главные направления).

Двумерные диаграммы, биплоты (biplots). Отображение главных компонент. Пример

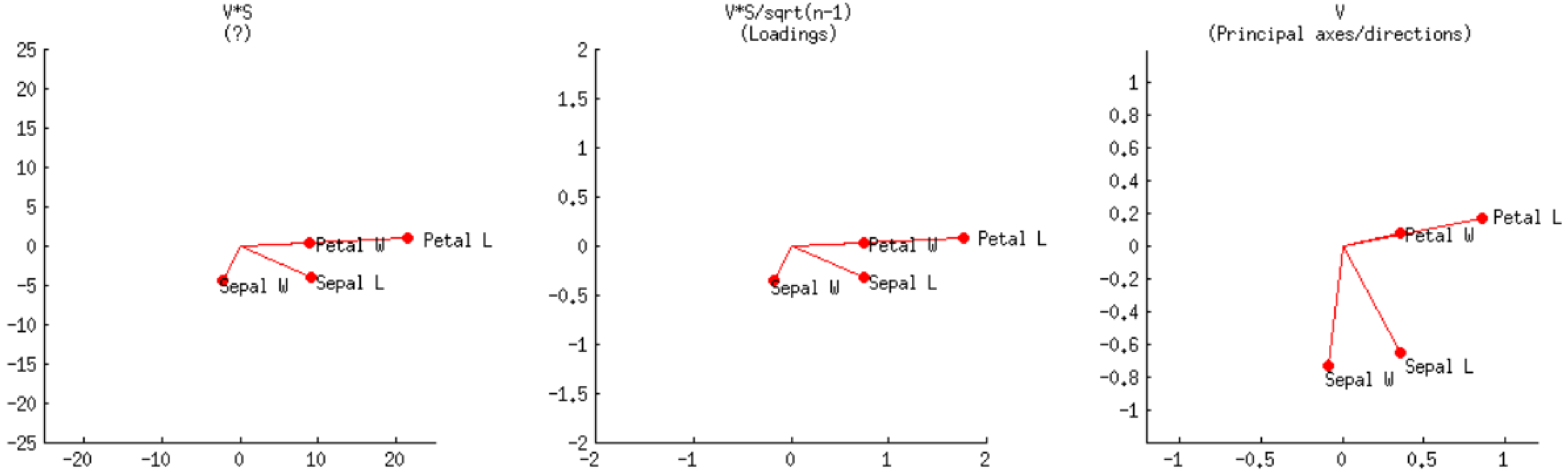


Двумерные диаграммы, биплоты (biplots). Отображение исходных переменных

Далее исходные переменные изображаются стрелками; то есть (x, y) координаты направления для i-той стрелки задается i-ыми значениями в первом и втором столбце V. Для этих координат также можно выбрать виды нормализации:

- Столбцы $\mathbf{VS}$;
- Столбцы $\mathbf{VS}/\sqrt{n-1}$ - это нагрузки;
- Столбцы $\mathbf{V}$ - это главные оси (также известные как главные направления, также известные как собственные векторы).

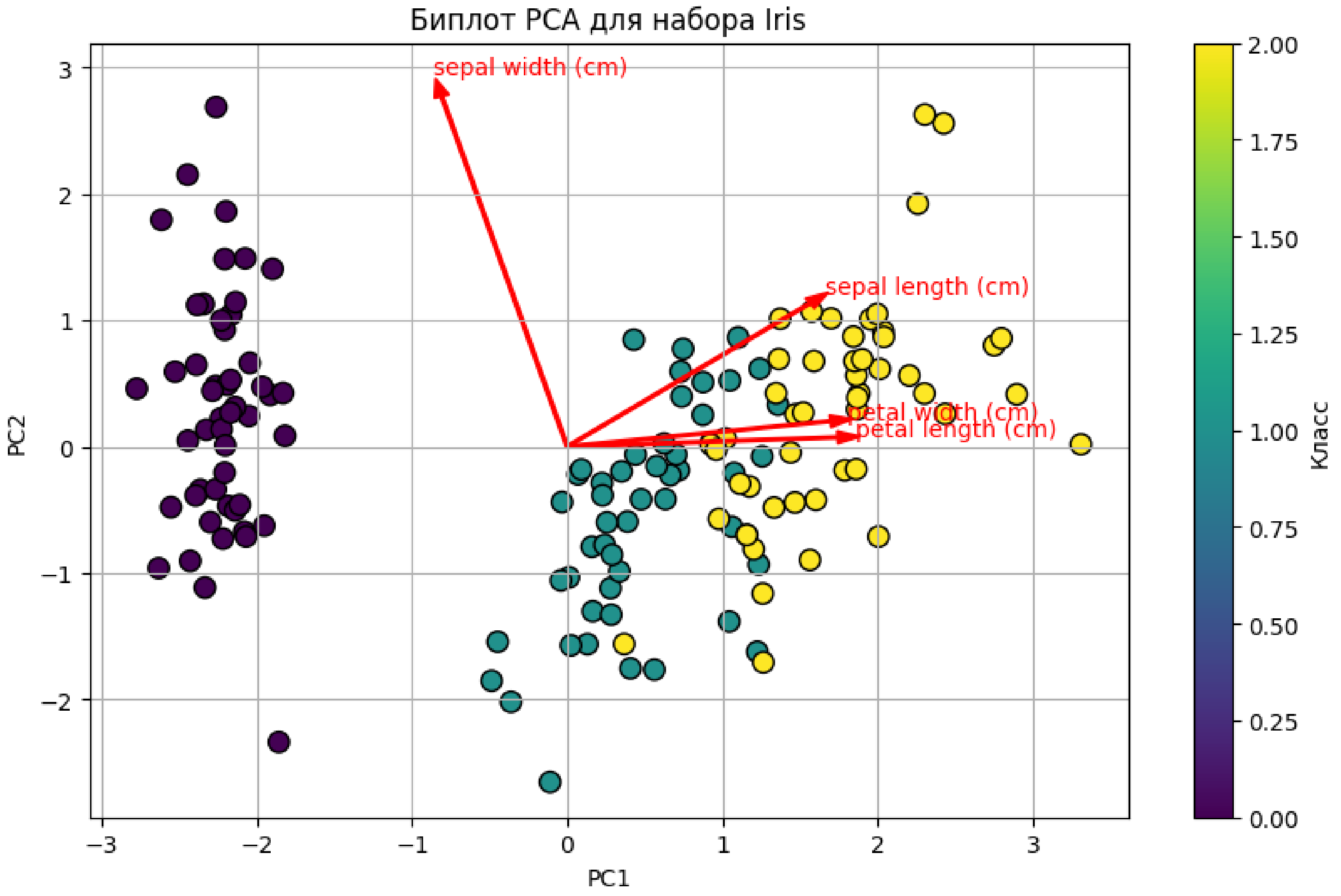
Двумерные диаграммы, биплоты (biplots). Отображение исходных переменных. Пример



Биплот для набора Ирисы Фишера. Python

```
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
# Данные Iris для примера
from sklearn.datasets import load_iris
data = load_iris()
X = data.data
y = data.target
# Нормализация
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
# PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)
# Визуализация объектов
plt.figure(figsize=(10, 6))
scatter = plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y, cmap='viridis', edgecolor='k', s=80)
# Визуализация переменных (стрелки)
for i, feature in enumerate(data.feature_names):
    plt.arrow(0, 0, pca.components_[0, i]*3, pca.components_[1, i]*3,
              color='red', width=0.02, head_width=0.1)
    plt.text(pca.components_[0, i]*3.2, pca.components_[1, i]*3.2, feature,
             color='red', fontsize=10)
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.title('Биплот PCA для набора Iris')
plt.colorbar(scatter, label='Класс')
plt.grid(True)
plt.show()
```

Биплот для набора Ирисы Фишера



SVD. Пример выбора разных рангов

```python
U, S, VT = np.linalg.svd(X, full_matrices = False)
S = np.diag(S)

fig,ax=plt.subplots(1,3)
i = 0
for k in (5, 20, 100):
    X_Ap = U[:,:k] @ S[0:k,:k] @ VT[:k,:]
    i += 1
    img = ax.flatten()[i-1].imshow(X_Ap)
    img.set_cmap('gray')
    ax.flatten()[i-1].axis('off')
    ax.flatten()[i-1].set_title('k = ' + str(k))
```

Применение SVD. Эффективная инверсия матриц

Разложение по сингулярным числам может значительно упростить инверсию матриц для некоторых конкретных матричных структур. Это полезно в алгоритмах, где требуется повторяющаяся инверсия с небольшими изменениями в матричной структуре.

Применение SVD. Сжатие данных

Разложение по сингулярным числам может выявить наиболее значимые линейные компоненты (направления, подпространства и т. д.), содержащие наибольшее количество информации. Это имеет очевидное применение в сжатии данных. При использовании PCA в сжатии изображений, PCA использует SVD внутри для расчета компонентов с наибольшей дисперсией.

Применение SVD. Шумоподавление данных

Подобно сжатию данных, для удаления шума из данных можно использовать разложение по сингулярным числам с удалением шума.

Применение SVD. Обработка естественного языка

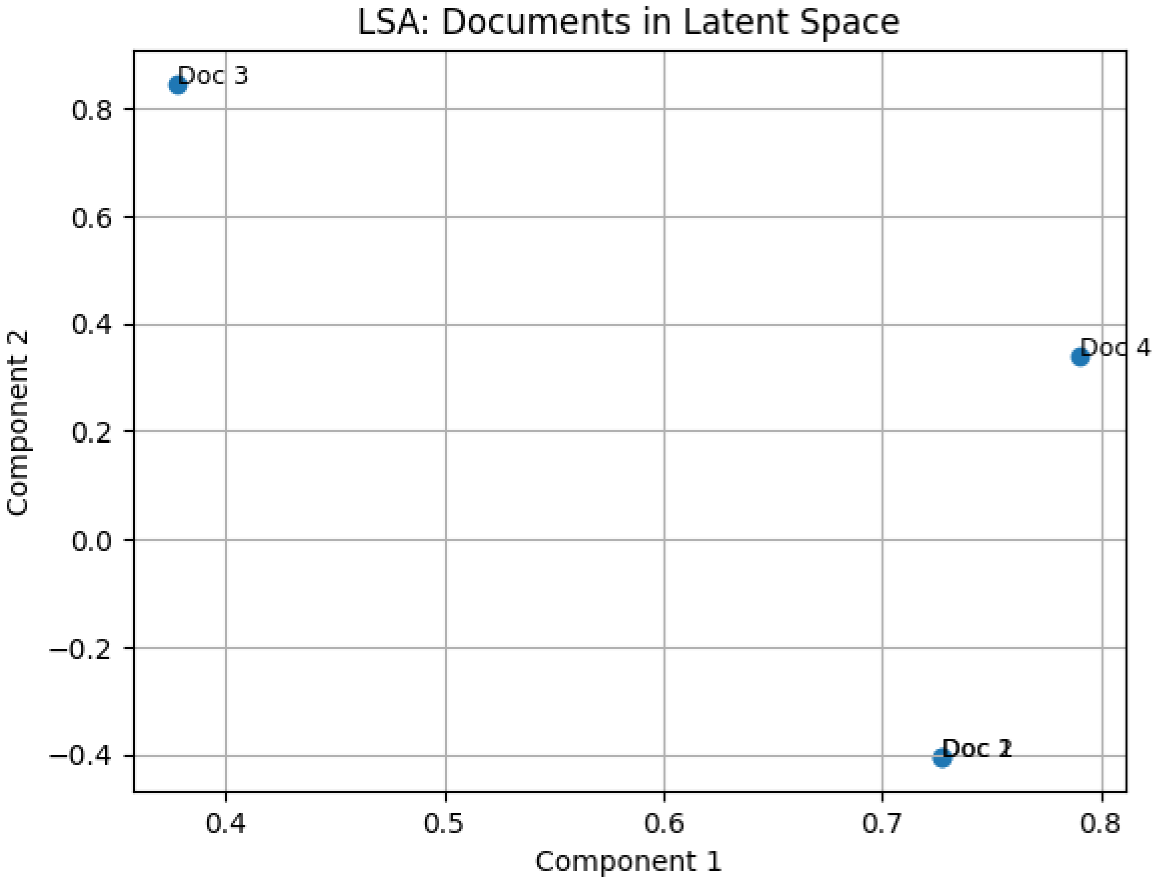
SVD применяется во многих процессах с большим количеством линейной алгебры, таких как искусственный интеллект, машинное обучение, LSA, совместная фильтрация и обработка естественного языка.

Применение SVD. Обработка естественного языка

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import TruncatedSVD
documents = [
    "I love machine learning",
    "Machine learning is fun",
    "Python is great for data science",
    "Data science and machine learning are related"
]
# TF-IDF
vectorizer = TfidfVectorizer(stop_words='english')
X_tfidf = vectorizer.fit_transform(documents)
```

```
# SVD (TruncatedSVD — для больших матриц)
svd = TruncatedSVD(n_components=2, random_state=42)
X_svd = svd.fit_transform(X_tfidf)
# Визуализация
plt.scatter(X_svd[:, 0], X_svd[:, 1])
for i, doc in enumerate(documents):
    plt.text(X_svd[i, 0], X_svd[i, 1], f'Doc {i+1}', fontsize=9)
plt.xlabel('Component 1')
plt.ylabel('Component 2')
plt.title('LSA: Documents in Latent Space')
plt.grid(True)
plt.show()
```

Применение SVD. Обработка естественного языка



Связь SVD и PCA

Аспект	**PCA**	**SVD**
Вход	Центрированная матрица	Любая матрица
Основа	Ковариационная матрица	Разложение матрицы
Главные компоненты	Собственные векторы	Столбцы U × \Sigma
Нагрузки	Собственные векторы	Столбцы V

Спасибо за внимание!