18

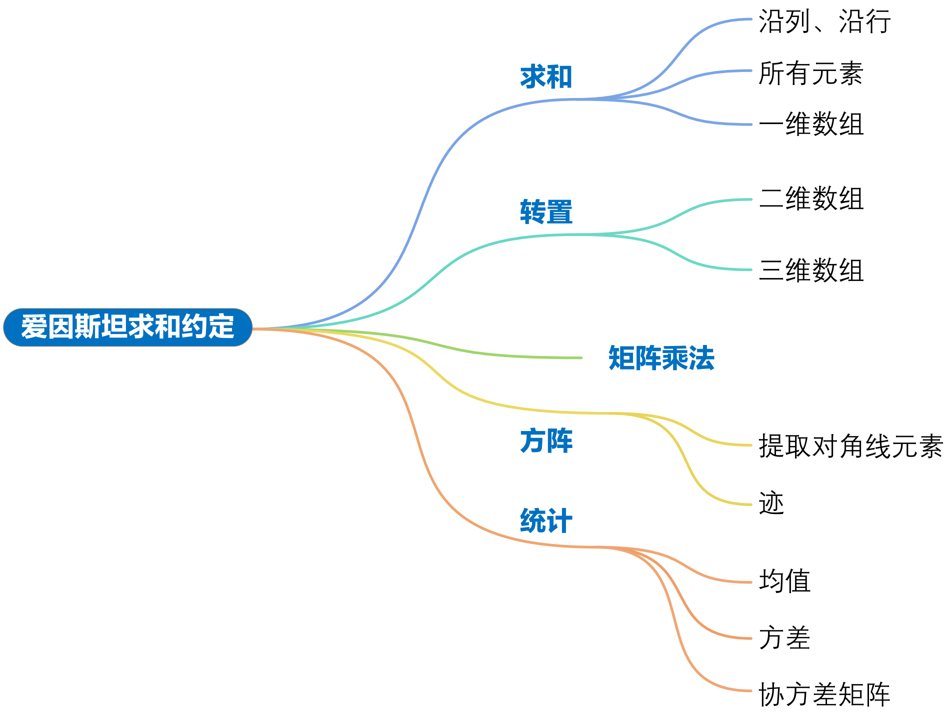
Einstein Summation in NumPy

NumPy 爱因斯坦求和约定

简化线性代数和张量计算，可以蜻蜓点水扫读

> 我不能教任何人任何东西。我只能让他们思考。
>
> ***I cannot teach anybody anything. I can only make them think.***
>
> —— 苏格拉底 (Socrates) | 古希腊哲学家 | 470 ~ 399 BC

- `numpy.average()` 计算平均值
- `numpy.cov()` 计算协方差矩阵
- `numpy.diag()` 以一维数组的形式返回方阵的对角线元素，或将一维数组转换成对角阵
- `numpy.einsum()` 爱因斯坦求和约定
- `numpy.stack()` 将矩阵叠加
- `numpy.sum()` 求和



18.1 什么是爱因斯坦求和约定?

NumPy 中还有一个非常强大的函数 numpy.einsum()，它完成的是**爱因斯坦求和约定** (Einstein summation convention 或 Einstein notation)。

爱因斯坦求和约定，由**阿尔伯特 · 爱因斯坦** (Albert Einstein) 于 1916 年提出，是一种数学表示法，用于简化线性代数和张量计算中的表达式。

使用 numpy.einsum() 完成绝大部分有关线性代数运算时，大家记住一个要点——输入中重复的索引代表元素相乘，输出中消去的索引意味着相加。

举个例子，矩阵 A 和 B 相乘用 numpy.einsum()函数可以写成：

```
C = numpy.einsum('ij,jk->ik', A, B)
```

如图 1 所示，“->” 之前分别为矩阵 A 和 B 的索引，它们用逗号隔开。矩阵 A 行索引为 i，列索引为 j。矩阵 B 行索引为 j，列索引为 k。j 为重复索引，因此在这个方向上元素相乘。

“->” 之后为输出结果的索引。输出结果索引为 ik，消去 j，因此在 j 索引方向上存在相乘再求和的运算。

注意，当然根据爱因斯坦求和运算的具体定义 (本章不展开讨论)，我们也会遇到输入中存在不重复索引，但是这些索引在输出中也消去的情况。本章配套代码会给出几个例子。

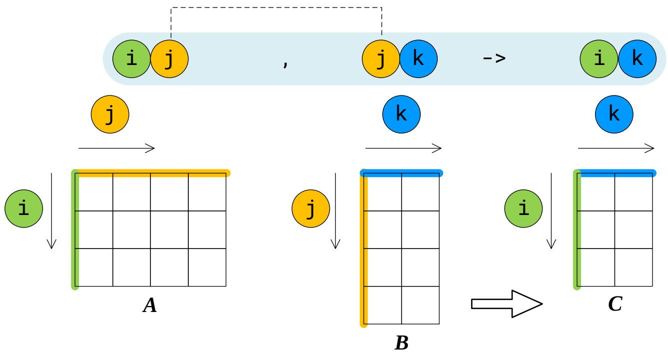


图 1. 利用爱因斯坦求和约定计算矩阵乘法

表 1 总结如何使用 numpy.einsum() 完成常见线性代数运算。下面我们选取其中重要的运算配合鸢尾花数据展开讲解。

为了方便大家理解，我们在本章中不会介绍爱因斯坦求和约定的具体数学表达，而是通过图解和 Python 实例方式让大家理解这个数学工具。

大家如果之前没有学过线性代数，这一章可以跳过不读；用到的时候再回来参考。

表 1. 使用 numpy.einsum()完成常见线性代数运算

运算	使用 numpy.einsum()完成运算
向量 ***a*** 所有元素求和（结果为标量）	numpy.einsum('ij->',a) numpy.einsum('i->',a_1D)
等行数列向量 ***a*** 和 ***b*** 的逐项积	numpy.einsum('ij,ij->ij',a,b) numpy.einsum('i,i->i',a_1D,b_1D)
等行数列向量 ***a*** 和 ***b*** 的向量内积（结果为标量）	numpy.einsum('ij,ij->',a,b) numpy.einsum('i,i->',a_1D,b_1D)
向量 ***a*** 和自身的张量积	numpy.einsum('ij,ji->ij',a,a) numpy.einsum('i,j->ij',a_1D,a_1D)
向量 ***a*** 和 ***b*** 的张量积	numpy.einsum('ij,ji->ij',a,b) numpy.einsum('i,j->ij',a_1D,b_1D)
矩阵 ***A*** 的转置	numpy.einsum('ji',A) numpy.einsum('ij->ji',A)
矩阵 ***A*** 所有元素求和（结果为标量）	numpy.einsum('ij->',A)
矩阵 ***A*** 对每一列元素求和	numpy.einsum('ij->j',A)
矩阵 ***A*** 对每一行元素求和	numpy.einsum('ij->i',A)
提取方阵 ***A*** 的对角元素（结果为向量）	numpy.einsum('ii->i',A)
计算方阵 ***A*** 的迹 trace(***A***)（结果为标量）	numpy.einsum('ii->',A)
计算矩阵 ***A*** 和 ***B*** 乘积	numpy.einsum('ij,jk->ik', A, B)
乘积 ***AB*** 结果所有元素求和（结果为标量）	numpy.einsum('ij,jk->', A, B)
矩阵 ***A*** 和 ***B*** 相乘后再转置，即 $(\boldsymbol{AB})^{\mathrm{T}}$	numpy.einsum('ij,jk->ki', A, B)
形状相同矩阵 ***A*** 和 ***B*** 逐项积	numpy.einsum('ij,ij->ij', A, B)

本节配套的 Jupyter Notebook 文件是 Bk1_Ch18_01.ipynb，请大家一边阅读本章一边实践。

18.2 二维数组求和

本节介绍二维数组求和。

代码 1 ⓐ 导入鸢尾花数据矩阵。

ⓑ 提取四个特征样本数据，保存在 X，结果二维数组。

ⓒ提取标签数据。

每一列求和

代码 1ⓓ中 `np.einsum('ij->j', X)` 的含义是对输入数组 X 进行一个特定的操作，其中 `'ij->j'` 是一个描述操作的字符串。下面，让我们来分解这个字符串。

如图 2 所示，`'ij'` 表示输入数组 X 的维度索引。`'i'` 和 `'j'` 是分别表示二维数组的行和列。

`'->j'` 表示输出的维度索引。在这里，`'->'` 表示输出；`'j'` 是输出数组的维度索引，表示最终结果的维度。也就是说，`'i'` 这个索引被压缩、折叠。

所以，`np.einsum('ij->j', X)` 的操作是将输入二维数组 X 沿着 `'i'` 维度求和，然后返回一维数组，其维度只有 `'j'`。

总结来说，图 2 执行了列求和操作，将二维数组的每一列相加。相当于 `np.sum(X, axis = 0)`。

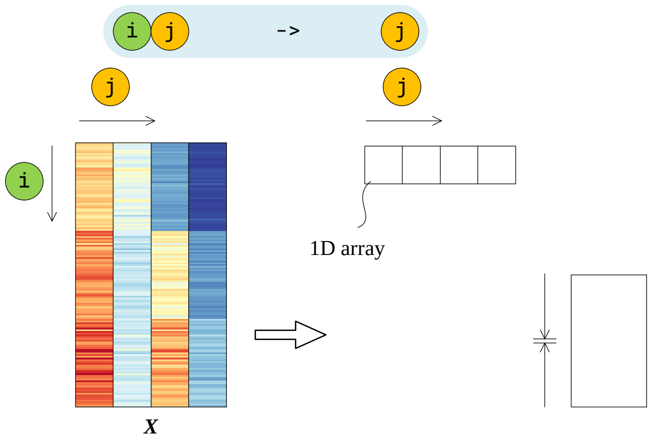


图 2. 利用爱因斯坦求和约定计算每一列求和

每一行求和

类似地，代码 1ⓔ将输入二维数组 X 沿着 `'j'` 维度求和，然后返回一维数组，其维度只有 `'i'`。也就是说，这行代码完成了行求和操作，将二维数组的每一行相加。

如图 3 所示，`'ij'` 表示输入数组 X 的维度索引。`'i'` 和 `'j'` 是两个维度索引。`'->i'` 表示输出的维度索引。相当于 `np.sum(X, axis = 1)`

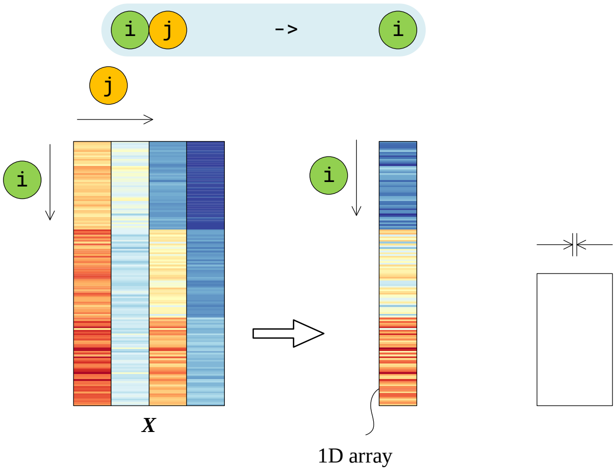


图 3. 利用爱因斯坦求和约定计算每一行求和

所有元素求和

代码 1 f 中 np.einsum('ij->', X) 的操作是对整个输入二维数组 X 进行汇总求和。

具体操作是将矩阵中的所有元素相加，最终返回一个标量值，表示所有元素的总和。相当于 np.sum(X, axis =(0,1))。

如图 4 所示，'i' 和 'j' 这两个维度索引都被折叠。

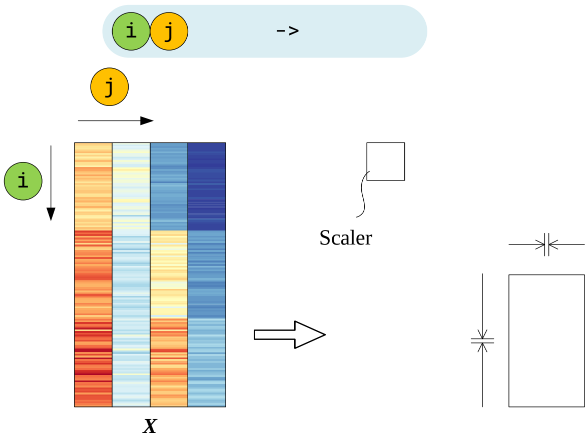


图 4. 利用爱因斯坦求和约定计算矩阵所有元素之和

```python
# 导入包
import numpy as np
from sklearn.datasets import load_iris

# 从sklearn导入鸢尾花数据
iris = load_iris()

X = iris.data
y = iris.target

# 每一列求和
np.einsum('ij->j',X)
# np.sum(X, axis = 0)

# 每一行求和
np.einsum('ij->i',X)
# np.sum(X, axis = 1)

# 矩阵所有元素求和
np.einsum('ij->',X)
# np.sum(X, axis = (0,1))
```

代码 1. 爱因斯坦求和约定代码，求和 | Bk1_Ch18_01.ipynb

18.3 转置

本节介绍如何用爱因斯坦求和约定完成二维、三维数组转置。

二维数组

如图 5 所示，对于二维数组，用爱因斯坦求和约定完成转置很容易。我们只需要调换维度索引，请大家参考代码 2 中 a。

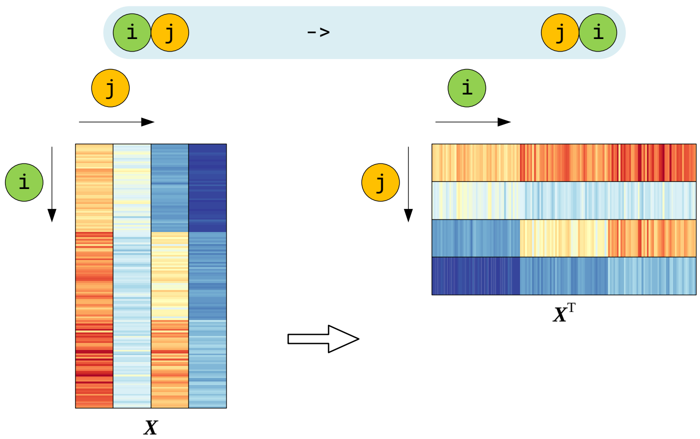


图 5. 利用爱因斯坦求和约定计算二维数组转置

三维数组

对于三维数组，我们也可以在指定轴上完成转置。如图 6 所示，对于这个三维数组，我们保持 i 不变，通过调换 j 和 k 维度索引，完成这两个方向的转置。

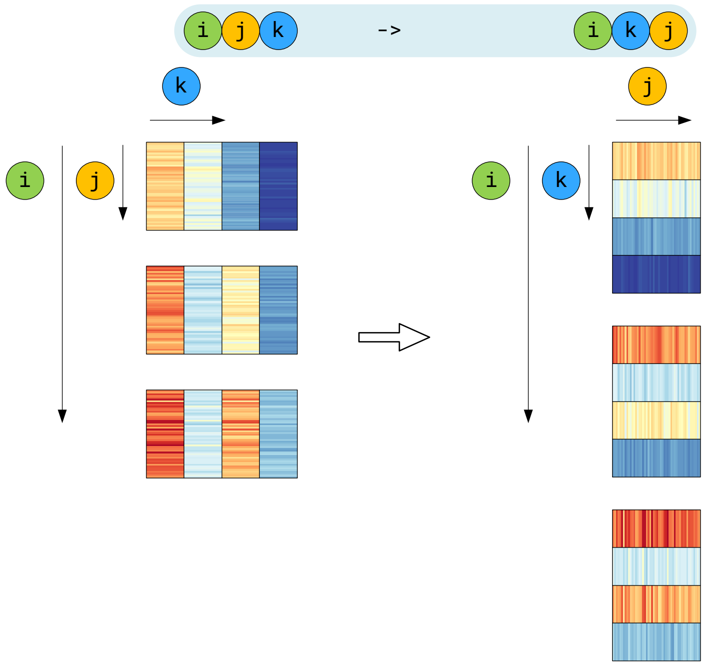


图 6. 利用爱因斯坦求和约定计算三维数组转置

代码 2 中 b 首先利用 np.stack()创建了一个三维数组。X[y == 0] 利用布尔切片提取 y 为 0 位置对应的 X 中元素，即鸢尾花数据中标签为 setosa 的样本数据。

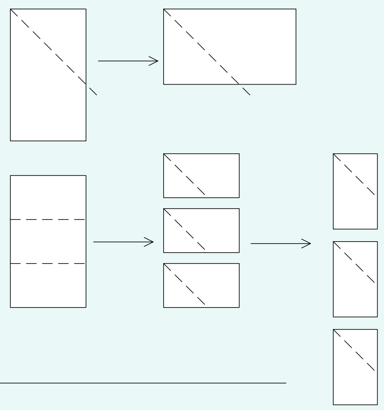
c 利用 numpy.einsum()完成三维数组转置，i 索引维度保持不变，j 和 k 调换。

这一句下面还给出如何用 numpy.transpose()完成相同的转置运算，这句被注释掉。

```
# 二维数组转置
np.einsum('ij->ji',X)
# X.T
# np.transpose(X)

# 三维数组
X3D = np.stack([X[y == 0],
                X[y == 1],
                X[y == 2]], axis=0)

# 三维数组转置
X3D_T = np.einsum('ijk->ikj', X3D)
# np.transpose(X3D,(0, 2, 1))
```



代码 2. 爱因斯坦求和约定代码，转置；使用时请配合前文代码 | Bk1_Ch18_01.ipynb

18.4 矩阵乘法

本节用三个例子介绍如何用爱因斯坦求和约定完成矩阵乘法运算。

格拉姆矩阵

如图 7 所示，在计算格拉姆矩阵 G 时，我们指定第一个矩阵 X 的维度索引为 i、j，第二个 X 维度索引为 i、k。利用爱因斯坦求和约定，维度索引 i 被折叠。

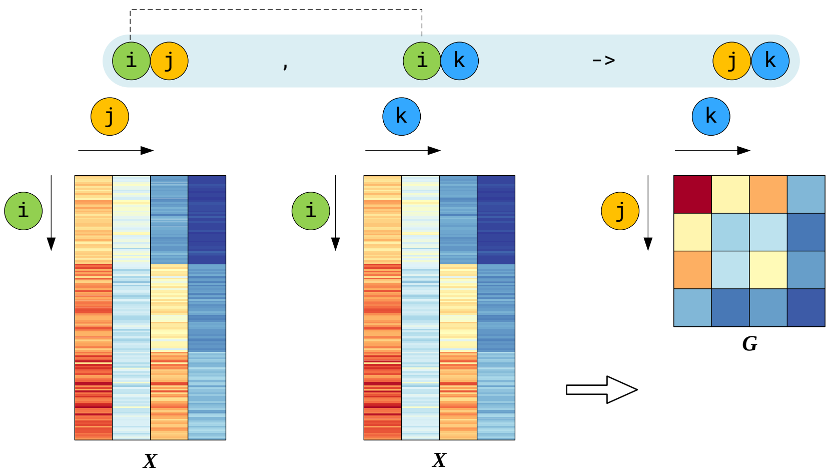


图 7．利用爱因斯坦求和约定计算格拉姆矩阵 G

图 8 总结了在格拉姆矩阵中如何计算对角线元素和非对角线元素。

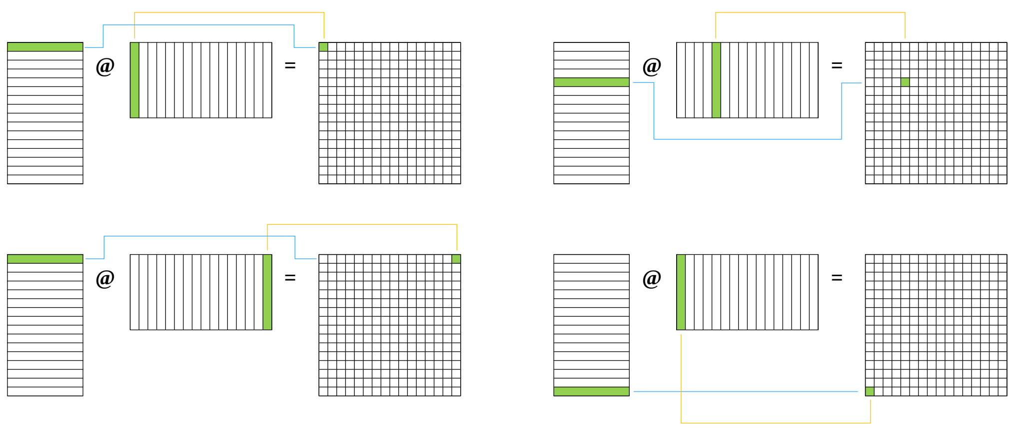


图 8．格拉姆矩阵对角线元素和非对角元素

类似地，如图 9 所示，在计算格拉姆矩阵 H 时，我们指定第一个矩阵 X 的维度索引为 i、j，第二个 X 维度索引为 k、j。利用爱因斯坦求和约定，维度索引 j 被折叠。

请大家自行分析代码 3 中 a 和 b 两句。

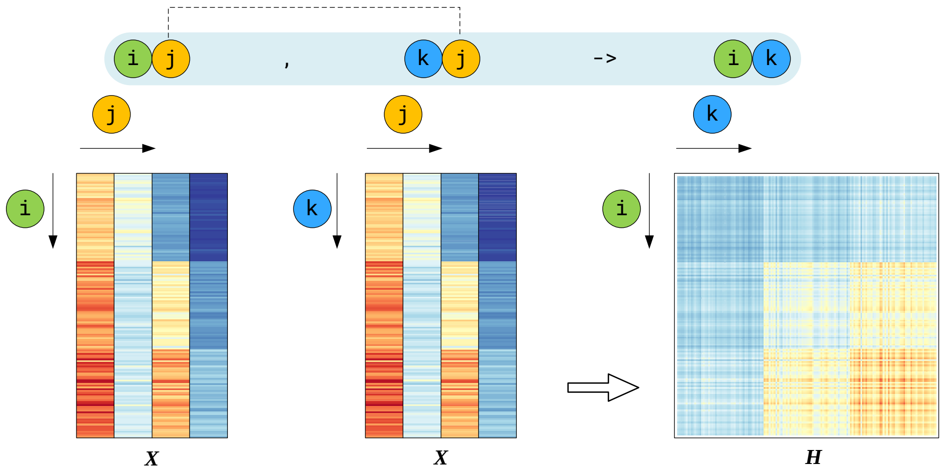


图 9. 利用爱因斯坦求和约定计算格拉姆矩阵 ***H***

分类矩阵乘法

如图 10 所示，我们还可以用爱因斯坦求和约定完成更为复杂的矩阵乘法。在计算格拉姆矩阵时，我们考虑不同鸢尾花类别。也就是说，每一类鸢尾花标签对应一个样本数据切片的格拉姆矩阵。

请大家自行分析代码 3 中 c 。

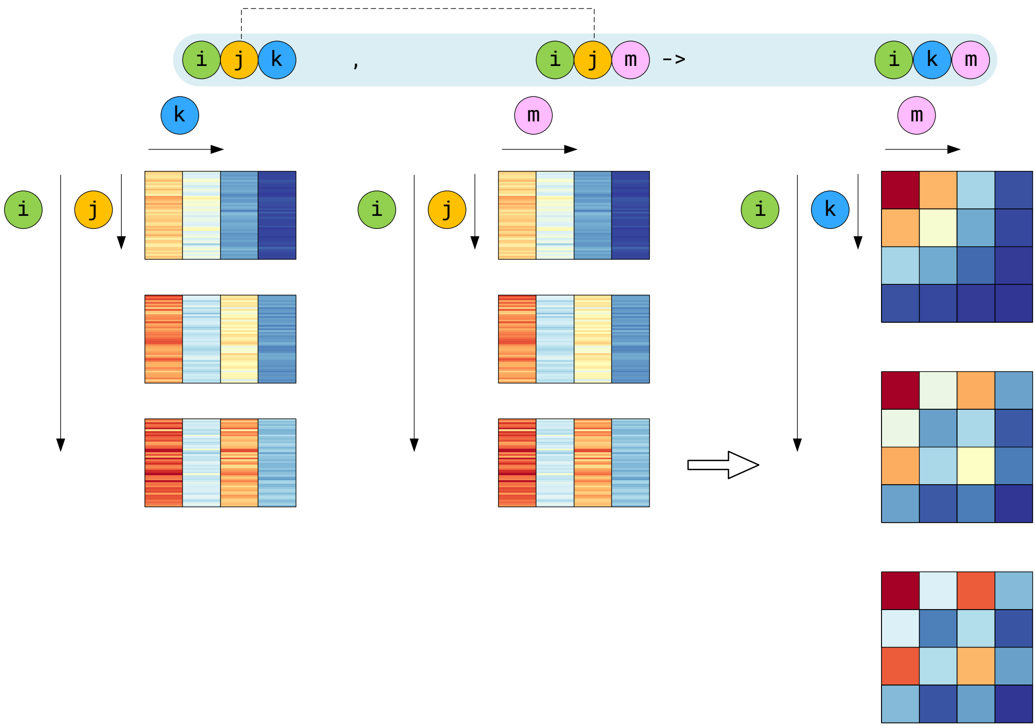


图 10. 利用爱因斯坦求和约定计算格拉姆矩阵，考虑不同鸢尾花类别

请大家思考如何用爱因斯坦求和约定完成多个矩阵连乘。

```python
# 计算矩阵乘法 X @ X.T
np.einsum('ij,kj->ik', X, X)
# np.einsum('ij,jk->ik', X, X.T)
# X @ X.T

# 计算矩阵乘法 X.T @ X
G = np.einsum('ij,ik->jk', X, X)
# np.einsum('ij,jk->ik', X.T, X)
# X.T @ X

# 三维矩阵乘法
G_3D = np.einsum('ijk,ijm->ikm', X3D, X3D)
# np.einsum('mij,mjk->mik', X3D_T, X3D)
```

代码 3. 爱因斯坦求和约定代码，矩阵乘法；使用时请配合前文代码 | Bk1_Ch18_01.ipynb

18.5 一维数组

有了本章之前的内容做铺垫，用爱因斯坦求和约定完成一维数组相关操作就很容易理解了。图 11 所示为利用 numpy.einsum()完成一维数组求和。

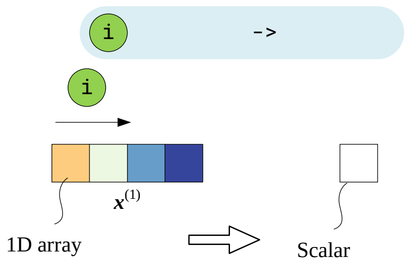


图 11. 利用爱因斯坦求和约定计算一维数组求和

图 12 所示为利用 numpy.einsum()计算一维数逐项积。

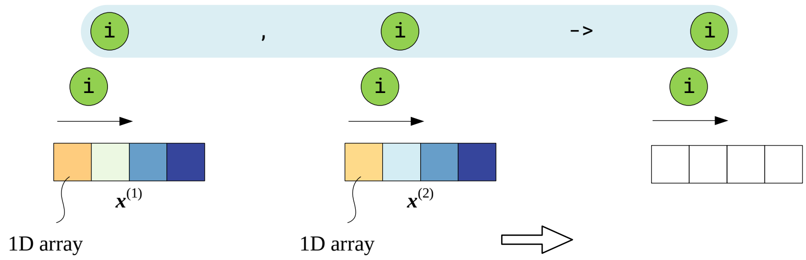


图 12. 利用爱因斯坦求和约定计算一维数组向量逐项积

图 13 所示为利用 `numpy.einsum()`计算一维数向量内积，即标量积。

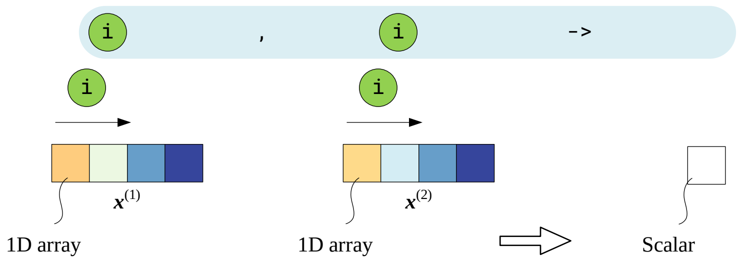


图 13. 利用爱因斯坦求和约定计算一维数组向量内积（标量积）

图 14 所示为利用 `numpy.einsum()`计算一维数向量外积，即张量积。

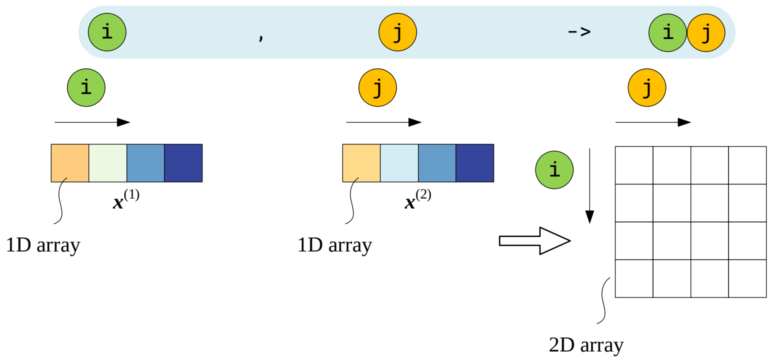


图 14. 利用爱因斯坦求和约定计算一维数组向量外积（张量积）

请大家自行分析代码 4。

```
# 提取两个行向量
a_1D = X[0]
b_1D = X[1]

# 一维向量求和
a np.einsum('i->',a_1D)

# 一维向量逐项积
b np.einsum('i,i->i',a_1D,b_1D)

# 一维向量内积
c np.einsum('i,i->',a_1D,b_1D)

# 一维向量外积
d np.einsum('i,j->ij',a_1D,b_1D)
```

代码 4. 爱因斯坦求和约定代码，一维数组；使用时请配合前文代码 | Bk1_Ch18_01.ipynb

18.6 方阵

本节介绍两个和方阵有关的爱因斯坦求和约定操作。图 15 所示为利用 numpy.einsum()提取方阵对角元素。

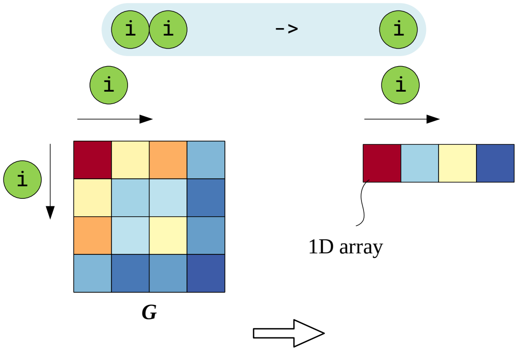


图 15. 利用爱因斯坦求和约定提取对角元素

图 16 所示为利用 numpy.einsum()计算方阵迹。本书前文提过，迹是指方阵主对角线上元素的总和。再次注意，迹只对方阵有定义。

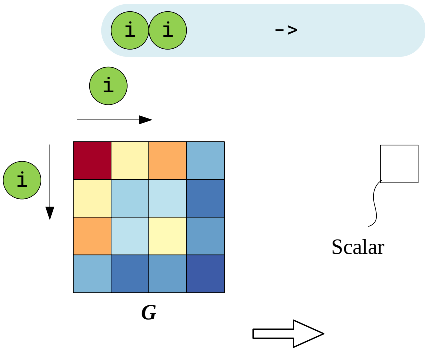


图 16. 利用爱因斯坦求和约定计算方阵迹

请大家自行分析代码 5。

```
#%% 取出方阵对角
np.einsum('ii->i',G)   # a
# np.diag(G)

#%% 计算方阵迹
np.einsum('ii->',G)   # b
# np.trace(G)
```

代码 5. 爱因斯坦求和约定代码，方阵；使用时请配合前文代码 | Bk1_Ch18_01.ipynb

18.7 统计运算

爱因斯坦求和约定也可以用来完成统计运算，比如均值（图 17）、方差（图 18）、协方差（图 19）。

图 18 中 X_c 代表中心化矩阵，即数据的每一列减去其均值。

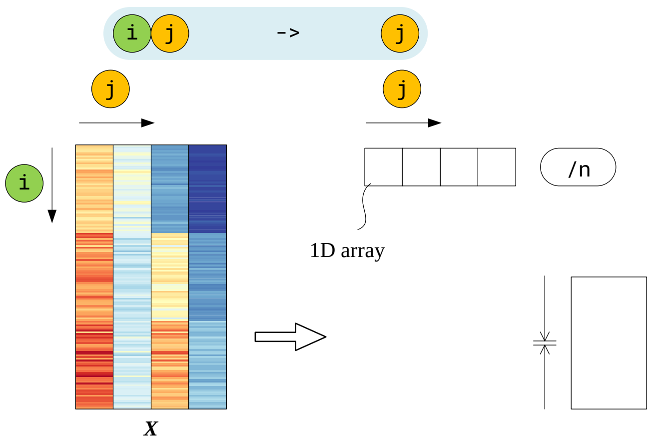


图 17. 利用爱因斯坦求和约定计算每一列均值

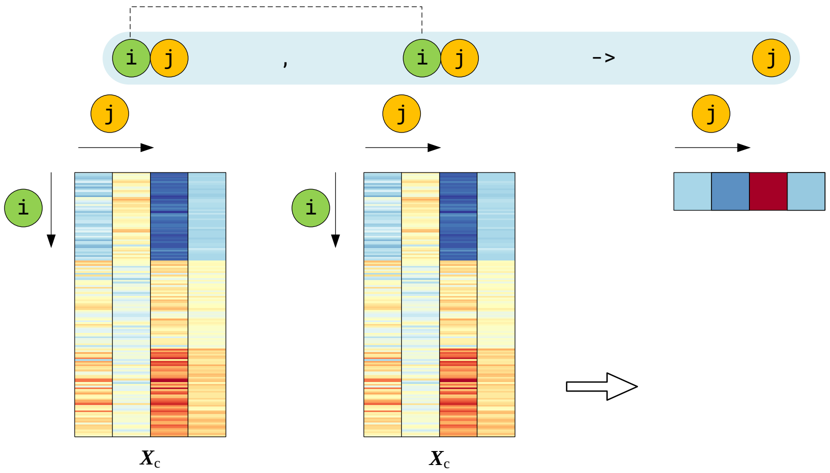


图 18. 利用爱因斯坦求和约定计算方差

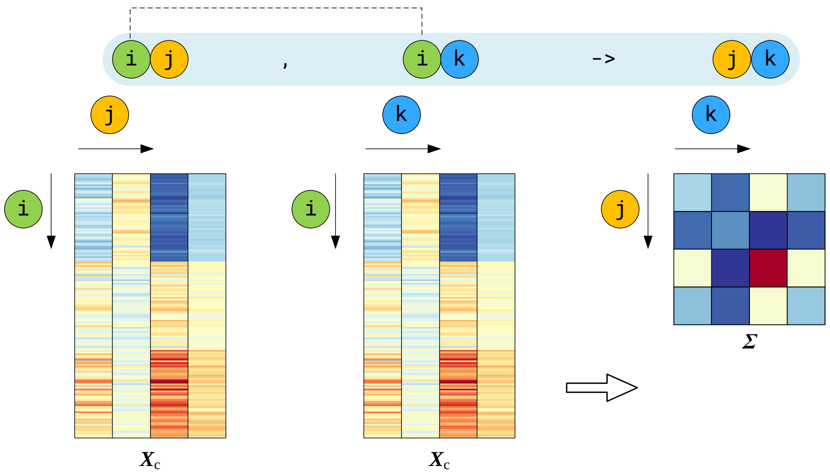


图 19. 利用爱因斯坦求和约定计算协方差矩阵

请大家自行分析代码 6。

```python
# 计算列均值，质心
n = X.shape[0]   # 样本数量
mean_X = np.einsum('ij->j', X) / n
# np.mean(X, axis = 0)

# 计算方差
X_c = X - mean_X   # 中心化数据
variance = np.einsum('ij,ij->j', X_c, X_c) / (n - 1)
# np.var(X, axis = 0, ddof = 1)

# 计算协方差矩阵
cov_matrix = np.einsum('ij,ik->jk', X_c, X_c) / (n - 1)
# np.cov(X.T, ddof = 1)
```

代码 6. 爱因斯坦求和约定代码，统计运算 | Bk1_Ch18_01.ipynb

请大家完成如下题目。

Q1. 唯一题目就是请大家在 JupyterLab 中自己复刻一遍本章所有爱因斯坦求和约定运算。

* 题目很基础，本书不给答案。

本章介绍了爱因斯坦求和约定，这个运算法则可以极大简化很多线性代数运算。这一章一方面介绍了这种全新的运算，另一方面我们还借此机会回顾了常见线性代数、概率统计运算。

下一章开始，我们正式进入“数据”板块，学习有关 Pandas 库的各种操作。