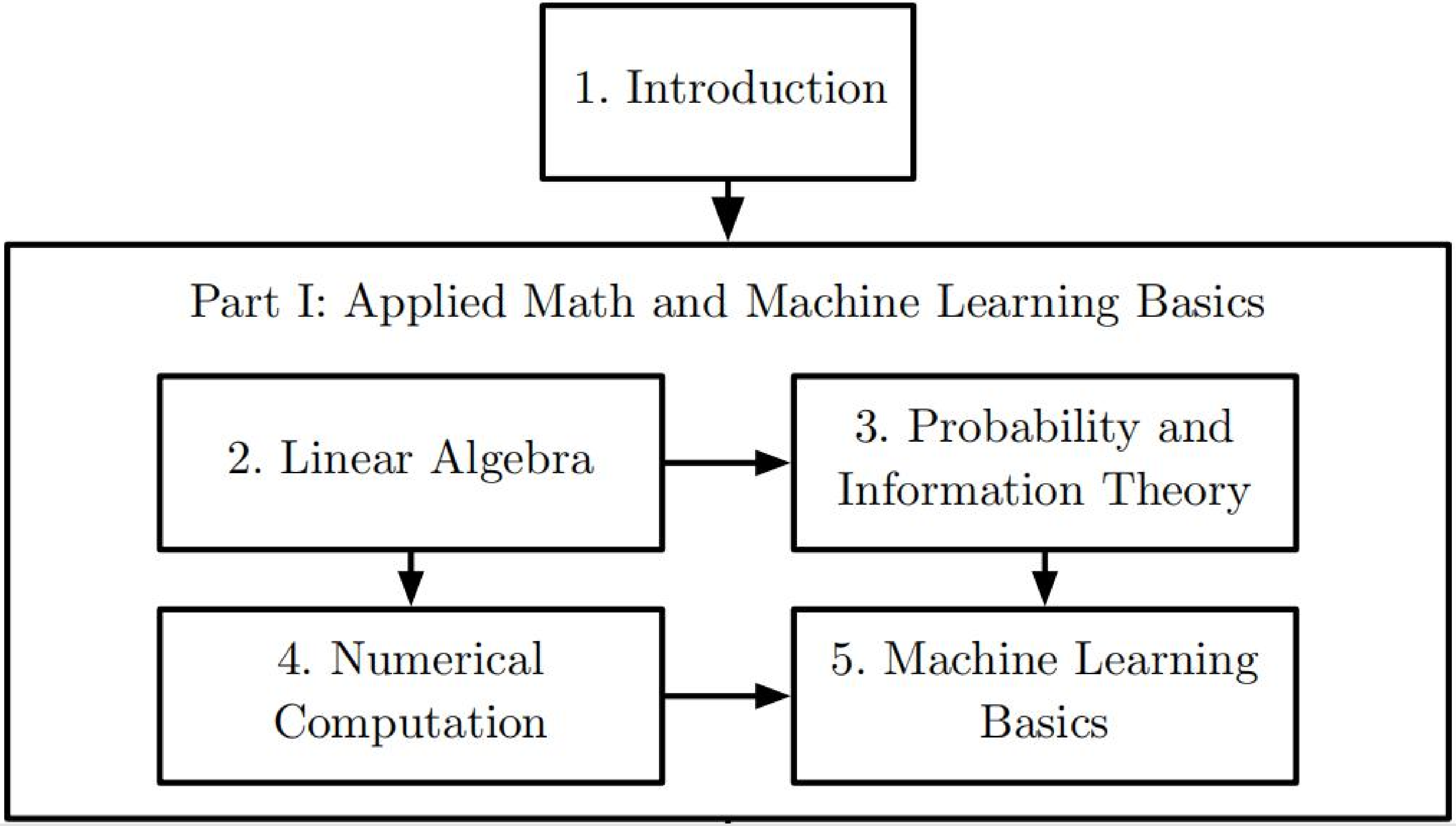
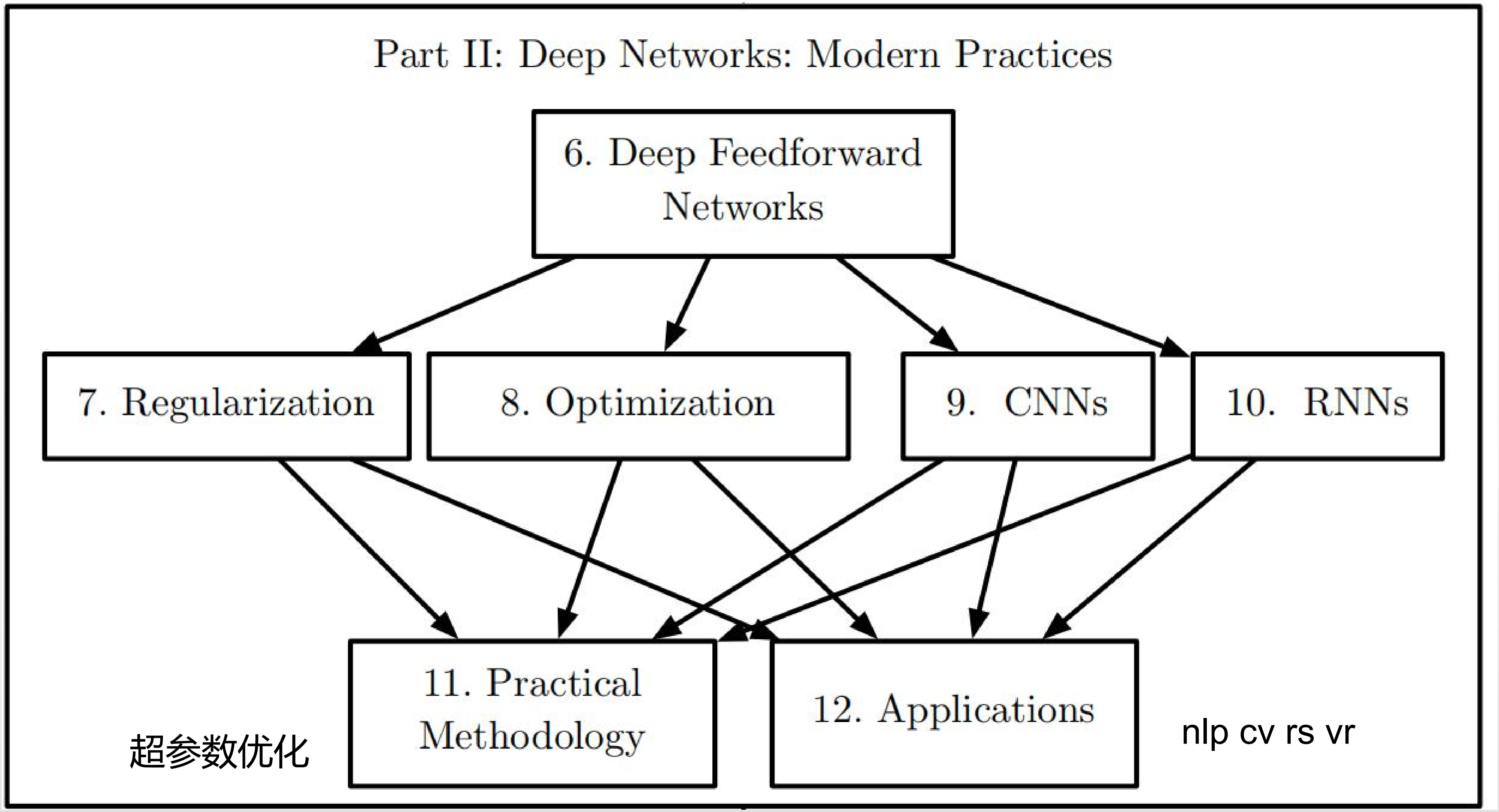
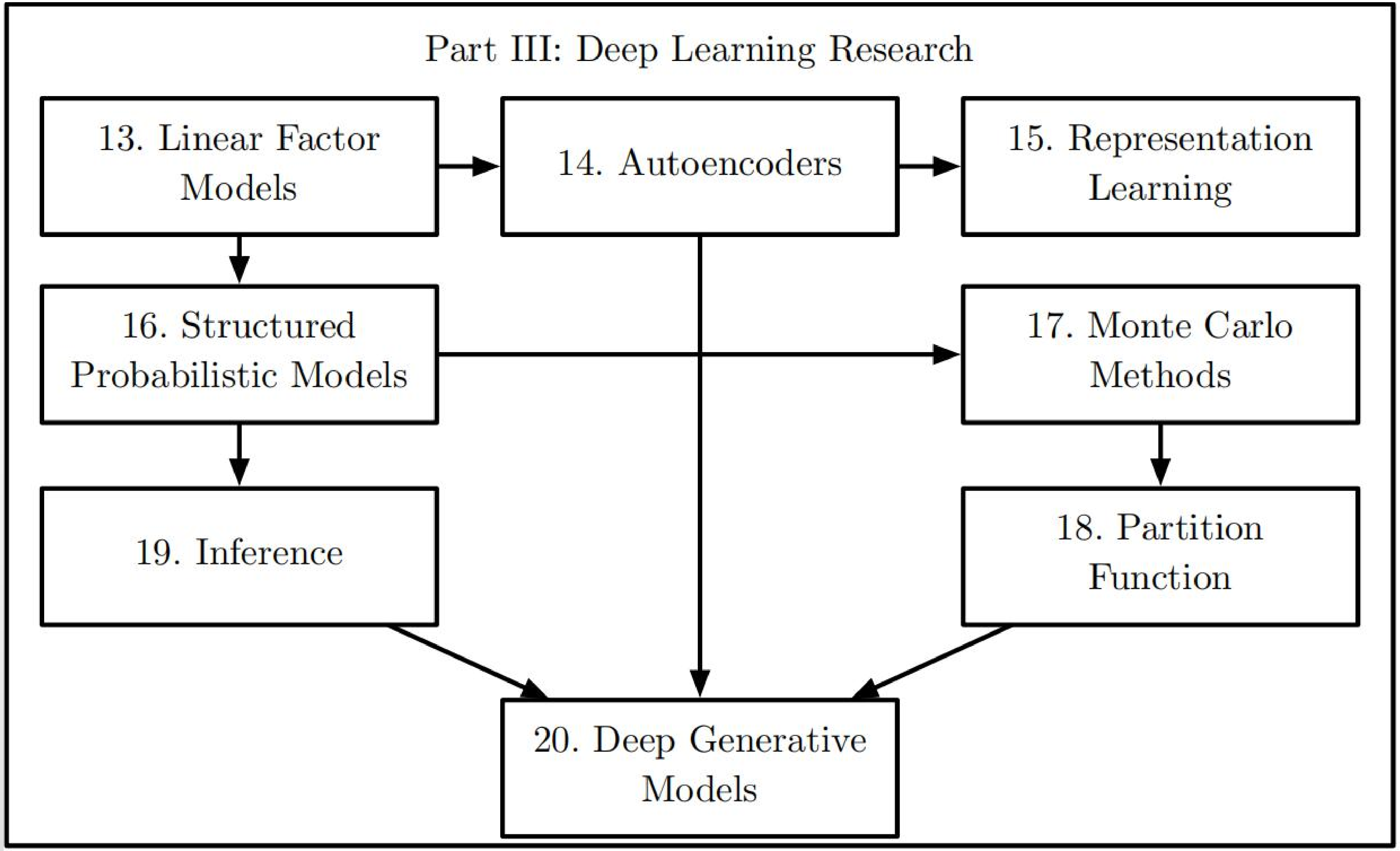
深度学习

2017.09.04


1. Introduction
Part I: Applied Math and Machine Learning Basics
2. Linear Algebra
3. Probability and Information Theory
4. Numerical Computation
5. Machine Learning Basics


Part II: Deep Networks: Modern Practices
6. Deep Feedforward Networks
7. Regularization
8. Optimization
9. CNNs
10. RNNs
11. Practical Methodology
12. Applications
超参数优化
nlp cv rs vr

PGM


Part III: Deep Learning Research
13. Linear Factor Models
14. Autoencoders
15. Representation Learning
16. Structured Probabilistic Models
17. Monte Carlo Methods
19. Inference
18. Partition Function
20. Deep Generative Models

第二章 线性代数：

2.1 标量、向量、矩阵和张量

```
[1]: import numpy as np
```

```
[2]: # 标量
s = 5
# 向量
v = np.array([1,2])
# 矩阵
m = np.array([[1,2], [3,4]])
# 张量
t = np.array([
  [[1,2,3],[4,5,6],[7,8,9]],
  [[11,12,13],[14,15,16],[17,18,19]],
  [[21,22,23],[24,25,26],[27,28,29]],
  ])
print("标量: " + str(s))
print("向量: " + str(v))
print("矩阵: " + str(m))
print("张量: " + str(t))
```

```
标量: 5
向量: [1 2]
矩阵: [[1 2]
 [3 4]]
张量: [[[ 1  2  3]
  [ 4  5  6]
  [ 7  8  9]]

 [[11 12 13]
  [14 15 16]
  [17 18 19]]
```

2.2 矩阵的转置、加法与乘法

矩阵转置 (Transpose) 相当于**沿着对角线翻转**，定义如下：

$$A_{i,j}^{\top} = A_{i,j}$$

矩阵转置的转置等于矩阵本身：

$$\left(\boldsymbol{A}^{\top}\right)^{\top} = \boldsymbol{A}$$

转置将矩阵的形状从 $m \times n$ 变成了 $n \times m$ 。

向量可以看成是**只有一列的矩阵**，为了方便，我们可以使用行向量加转置的操作，如：$\boldsymbol{x} = [x_1, x_2, x_3]^{\top}$。

标量也可以看成是**一行一列的矩阵**，其转置等于它自身：$a^{\top} = a$ 。

A.T 适用于一、二维

[3]:
```
A = np.array([[1.0,2.0],[1.0,0.0],[2.0,3.0]])
A_t = A.transpose()
print("A:", A)
print("A 的转置:", A_t)
```

```
A: [[1. 2.]
 [1. 0.]
 [2. 3.]]
A 的转置: [[1. 1. 2.]
 [2. 0. 3.]]
```

加法即**对应元素相加**，要求两个矩阵的形状一样：

$$\boldsymbol{C} = \boldsymbol{A} + \boldsymbol{B}, C_{i,j} = A_{i,j} + B_{i,j} \tag{5}$$

数乘即**一个标量**与**矩阵每个元素相乘**：

$$\boldsymbol{D} = a \cdot \boldsymbol{B} + c, D_{i,j} = a \cdot B_{i,j} + c \tag{6}$$

有时我们允许矩阵和向量相加的，得到一个矩阵，把 $\boldsymbol{b}$ 加到了 A 的每一行上，本质上是构造了一个将 $\boldsymbol{b}$ 按行复制的一个新矩阵，这种机制叫做广播 (`Broadcasting`)：

$$\boldsymbol{C} = \boldsymbol{A} + \boldsymbol{b}, C_{i,j} = A_{i,j} + b_j \tag{7}$$

```
[5]: m1 = np.array([[1.0,3.0],[1.0,0.0]])
m2 = np.array([[1.0,2.0],[5.0,0.0]])
print("按矩阵乘法规则: ", np.dot(m1, m2))
print("按逐元素相乘: ", np.multiply(m1, m2))
print("按逐元素相乘: ", m1*m2)
v1 = np.array([1.0,2.0])
v2 = np.array([4.0,5.0])
print("向量内积: ", np.dot(v1, v2))
```

```
按矩阵乘法规则:  [[16.  2.]
 [ 1.  2.]]
按逐元素相乘:  [[1. 6.]
 [5. 0.]]
按逐元素相乘:  [[1. 6.]
 [5. 0.]]
向量内积:  14.0
```

两个矩阵中对应元素的乘积，Hadamard 乘积（Hadamard product），记为 A⊙ B。

2.3 单位矩阵和逆矩阵

```
[6]: np.identity(3)
```

```
[6]: array([[1., 0., 0.],
            [0., 1., 0.],
            [0., 0., 1.]])
```

矩阵 $\boldsymbol{A}$ 的逆 (Inversion) 记作 $\boldsymbol{A}^{-1}$，定义为一个矩阵使得

$$\boldsymbol{A}^{-1}\boldsymbol{A} = \boldsymbol{I}_n$$

如果 $\boldsymbol{A}^{-1}$ 存在，那么线性方程组 $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ 的解为:

$$\boldsymbol{A}^{-1}\boldsymbol{A}\boldsymbol{x} = \boldsymbol{I}_n\boldsymbol{x} = \boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$$

```
[7]: A = [[1.0,2.0],[3.0,4.0]]
     A_inv = np.linalg.inv(A)
     print("A 的逆矩阵", A_inv)
```

```
A 的逆矩阵 [[-2.   1. ]
 [ 1.5 -0.5]]
```

linear algebra

2.4 线性相关和生成子空间

$$\sum_i c_i \boldsymbol{v}^{(i)}.$$

一组向量的生成子空间：由这组向量的线性组合所构成的空间

确定 Ax = b 是否有解相当于确定向量 b 是否在 A 列向量的生成子空间中。这个特殊的生成子空间被称为 A 的 列空间（column space）或者 A 的 值域（range）。（把A的每个列看成一个向量）

2.5 范数

范数就是将向量映射到非负值的函数。

作用：衡量向量的大小

更严格地说，范数是满足下列性质的任意函数：

- $f(\boldsymbol{x}) = 0 \Rightarrow \boldsymbol{x} = \mathbf{0}$
- $f(\boldsymbol{x} + \boldsymbol{y}) \leq f(\boldsymbol{x}) + f(\boldsymbol{y})$ （三角不等式（triangle inequality））
- $\forall \alpha \in \mathbb{R},\ f(\alpha \boldsymbol{x}) = |\alpha| f(\boldsymbol{x})$

距离：第三条是对称性

$$\|\boldsymbol{x}\|_p = \left(\sum_i |x_i|^p \right)^{\frac{1}{p}}$$

当 p = 2 时，L2 范数被称为 欧几里得范数（Euclidean norm）。它表示从原点出发到向量 x 确定的点的欧几里得距离。||X||，略去下标 2。平方 L2 范数， 计算：$x^\top x$。

L1 范数。L1 范数可以简化如下：

$$\|\boldsymbol{x}\|_1 = \sum_i |x_i|.$$

为什么L1正则能产生稀疏解？

为什么稀疏解能抑制过拟合？

百面 7.7

L∞ 范数：

$$\|\boldsymbol{x}\|_\infty = \max_i |x_i|.$$

衡量矩阵的大小。在深度学习中，最常见的做法是使用 Frobenius 范数（Frobenius norm），

$$\|\boldsymbol{A}\|_F = \sqrt{\sum_{i,j} A_{i,j}^2},$$

```
[8]: a = np.array([1.0,3.0])
     print("向量 2 范数", np.linalg.norm(a,ord=2))
     print("向量 1 范数", np.linalg.norm(a,ord=1))
     print("向量无穷范数", np.linalg.norm(a,ord=np.inf))
```

```
向量 2 范数 3.1622776601683795
向量 1 范数 4.0
向量无穷范数 3.0
```

```
[9]: a = np.array([[1.0,3.0],[2.0,1.0]])
     print("矩阵 F 范数", np.linalg.norm(a,ord="fro"))
```

```
矩阵 F 范数 3.872983346207417
```

2.6 特殊类型的矩阵和向量—— 正交矩阵

正交矩阵（orthogonal matrix）是指行向量和列向量是分别标准正交的方阵：

$$\boldsymbol{A}^{\top}\boldsymbol{A} = \boldsymbol{A}\boldsymbol{A}^{\top} = \boldsymbol{I}. \tag{2.37}$$

这意味着

$$\boldsymbol{A}^{-1} = \boldsymbol{A}^{\top}, \tag{2.38}$$

2.7 特征分解：

特征分解（eigendecomposition）是使用最广的矩阵分解之一，即我们将矩阵分解成一组特征向量和特征值。

方阵 $\boldsymbol{A}$ 的**特征向量**（eigenvector）是指与 $\boldsymbol{A}$ 相乘后相当于对该向量进行缩放的非零向量 $\boldsymbol{v}$:

$$\boldsymbol{A}\boldsymbol{v} = \lambda \boldsymbol{v}. \tag{2.39}$$

标量 λ 被称为这个特征向量对应的**特征值**（eigenvalue）。（类似地，我们也可以定义**左特征向量**（left eigenvector）$\boldsymbol{v}^\top \boldsymbol{A} = \lambda \boldsymbol{v}^\top$，但是通常我们更关注**右特征向量**（right eigenvector））。

$$\boldsymbol{A} = \boldsymbol{V}\operatorname{diag}(\boldsymbol{\lambda})\boldsymbol{V}^{-1}.$$

每个实对称矩阵都可以分解成实特征向量和实特征值：（Q为正交阵）

$$\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top.$$

```python
A = np.array([[1.0,2.0,3.0],
        [4.0,5.0,6.0],
        [7.0,8.0,9.0]])
# 计算特征值
print("特征值:", np.linalg.eigvals(A))
# 计算特征值和特征向量
eigvals,eigvectors = np.linalg.eig(A)
print("特征值:", eigvals)
print("特征向量:", eigvectors)
```

```
特征值: [ 1.61168440e+01 -1.11684397e+00 -3.73313677e-16]
特征值: [ 1.61168440e+01 -1.11684397e+00 -3.73313677e-16]
特征向量: [[-0.23197069 -0.78583024  0.40824829]
 [-0.52532209 -0.08675134 -0.81649658]
 [-0.8186735   0.61232756  0.40824829]]
```

2.8 奇异值分解

$$A = UDV^{\top}.$$

A 是一个 m × n 的矩阵，那么 U 是一个 m × m 的正交矩阵，D 是一个 m × n 的矩阵，V 是一个 n × n 正交矩阵。

A 的 左奇异向量（left singular vector）是 $AA^{\top}$ 的特征向量。A 的 右奇异向量（right singularvector）是 $A^{\top}A$ 的特征向量。A 的非零奇异值是 $A^{\top}A$ 特征值的平方根，同时也是$AA^{\top}$ 特征值的平方根。

```
A = np.array([[1.0,2.0,3.0],
        [4.0,5.0,6.0]])
U,D,V = np.linalg.svd(A)
print("U:", U)
print("D:", D)
print("V:", V)
```

```
U: [[-0.3863177  -0.92236578]
 [-0.92236578  0.3863177 ]]
D: [9.508032   0.77286964]
V: [[-0.42866713 -0.56630692 -0.7039467 ]
 [ 0.80596391  0.11238241 -0.58119908]
 [ 0.40824829 -0.81649658  0.40824829]]
```

2.9 Moore-Penrose 伪逆

$$A^{+} = VD^{+}U^{\top}.$$

2.10 迹运算

迹运算返回的是矩阵对角元素的和：

$$\mathrm{Tr}(\boldsymbol{A}) = \sum_i \boldsymbol{A}_{i,i}.$$

$$\|\boldsymbol{A}\|_F = \sqrt{\sum_{i,j} A_{i,j}^2},$$

$$\|A\|_F = \sqrt{\mathrm{Tr}(\boldsymbol{A}\boldsymbol{A}^\top)}.$$

$$\mathrm{Tr}(\boldsymbol{A}) = \mathrm{Tr}(\boldsymbol{A}^\top).$$

$$\mathrm{Tr}(\boldsymbol{A}\boldsymbol{B}\boldsymbol{C}) = \mathrm{Tr}(\boldsymbol{C}\boldsymbol{A}\boldsymbol{B}) = \mathrm{Tr}(\boldsymbol{B}\boldsymbol{C}\boldsymbol{A}).$$

更一般的：

$$\mathrm{Tr}(\prod_{i=1}^{n} \boldsymbol{F}^{(i)}) = \mathrm{Tr}(\boldsymbol{F}^{(n)} \prod_{i=1}^{n-1} \boldsymbol{F}^{(i)}).$$

实例：主成分分析 （PCA）

$$
\begin{tikzcd}
 & c \in Rl \arrow[dr, "g(c) = Dc"] & \\
x \in Rn \arrow[ur, "f(x) = c"] & & r=g(c) \in Rn
\end{tikzcd}
$$

target： $x \approx g(f(x))$

$L(x; g(f(x)))$

PCA 由我们选择的解码函数而定

对于给定的X，我们希望找到能够另重构误差最小的编码c*，
用数学语言描述就是:

$$c^\star = \arg\min_c \|x - g(c)\|_2 = \arg\min_c \|x - g(c)\|_2^2$$

将平方L2范数展开:

$$\|x - g(c)\|_2^2 = (x - g(c))^\top (x - g(c)) = x^\top x - 2x^\top g(c) + g(c)^\top g(c)$$

$$\begin{aligned} c^\star &= \arg\min_c -2x^\top Dc + c^\top D^\top Dc \\ &= \arg\min_c -2x^\top Dc + c^\top I_l c \\ &= \arg\min_c -2x^\top Dc + c^\top c \end{aligned}$$

D只是列正交，非严格正交阵

优化:

$$\begin{aligned} \nabla_c(-2x^\top Dc + c^\top c) &= 0 \\ -2D^\top x + 2c &= 0 \\ c &= D^\top x \end{aligned}$$

得到编码函数:

$$f(x) = D^\top x.$$

所以，整个PCA重构操作可以这样定义：

$$r(\boldsymbol{x}) = g(f(\boldsymbol{x})) = \boldsymbol{D}\boldsymbol{D}^\top \boldsymbol{x}.$$

接下来，我们需要挑选编码矩阵 D：（因为要用相同的矩阵 D 对所有点进行解码，所以算重构损失的时候也要考虑到所有点，假设在 Rn 空间中我们有 m 个点 {x(1), . . . , x(m)}）
前面我们是用平方L2范数来衡量X与重构之后的X之间的误差的，考虑到所有点：

$$\boldsymbol{D}^* = \arg\min_{\boldsymbol{D}} \sqrt{\sum_{i,j} \left(x_j^{(i)} - r(\boldsymbol{x}^{(i)})_j \right)^2} \text{ subject to } \boldsymbol{D}^\top \boldsymbol{D} = \boldsymbol{I}_l.$$

$$\boldsymbol{d}^* = \arg\min_{\boldsymbol{d}} \sum_i \left\| \boldsymbol{x}^{(i)} - \boldsymbol{d}\boldsymbol{d}^\top \boldsymbol{x}^{(i)} \right\|_2^2 \text{ subject to } \|\boldsymbol{d}\|_2 = 1.$$

$$\boldsymbol{d}^* = \arg\min_{\boldsymbol{d}} \left\| \boldsymbol{X} - \boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top \right\|_F^2 \text{ subject to } \boldsymbol{d}^\top \boldsymbol{d} = 1.$$

$$\arg\min_{\boldsymbol{d}} \left\| \boldsymbol{X} - \boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top \right\|_F^2$$

$$\|A\|_F = \sqrt{\mathrm{Tr}(\boldsymbol{A}\boldsymbol{A}^\top)}.$$

$$= \arg\min_{\boldsymbol{d}} \mathrm{Tr}\left(\left(\boldsymbol{X} - \boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top \right)^\top \left(\boldsymbol{X} - \boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top \right) \right)$$

（式 (2.49) ）

$$= \arg\min_{\boldsymbol{d}} \mathrm{Tr}\left(\boldsymbol{X}^\top\boldsymbol{X} - \boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top - \boldsymbol{d}\boldsymbol{d}^\top\boldsymbol{X}^\top\boldsymbol{X} + \boldsymbol{d}\boldsymbol{d}^\top\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top \right)$$

$$= \arg\min_{\boldsymbol{d}} \mathrm{Tr}(\boldsymbol{X}^\top\boldsymbol{X}) - \mathrm{Tr}(\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top) - \mathrm{Tr}(\boldsymbol{d}\boldsymbol{d}^\top\boldsymbol{X}^\top\boldsymbol{X}) + \mathrm{Tr}(\boldsymbol{d}\boldsymbol{d}^\top\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top)$$

$$= \arg\min_{\boldsymbol{d}} - \mathrm{Tr}(\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top) - \mathrm{Tr}(\boldsymbol{d}\boldsymbol{d}^\top\boldsymbol{X}^\top\boldsymbol{X}) + \mathrm{Tr}(\boldsymbol{d}\boldsymbol{d}^\top\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top)$$

$$= \arg\min_{\boldsymbol{d}} - 2\mathrm{Tr}(\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top) + \mathrm{Tr}(\boldsymbol{d}\boldsymbol{d}^\top\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top)$$

$$\arg\min_{\boldsymbol{d}} - 2\mathrm{Tr}(\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top) + \mathrm{Tr}(\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top\boldsymbol{d}\boldsymbol{d}^\top) \text{ subject to } \boldsymbol{d}^\top\boldsymbol{d} = 1$$

$$= \arg\min_{\boldsymbol{d}} - 2\mathrm{Tr}(\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top) + \mathrm{Tr}(\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top) \text{ subject to } \boldsymbol{d}^\top\boldsymbol{d} = 1$$

$$
\begin{aligned}
&= \arg\min_{\boldsymbol{d}} \; -\operatorname{Tr}(\boldsymbol{X}^\top \boldsymbol{X} \boldsymbol{d} \boldsymbol{d}^\top) \text{ subject to } \boldsymbol{d}^\top \boldsymbol{d} = 1 \\
&= \arg\max_{\boldsymbol{d}} \operatorname{Tr}(\boldsymbol{X}^\top \boldsymbol{X} \boldsymbol{d} \boldsymbol{d}^\top) \text{ subject to } \boldsymbol{d}^\top \boldsymbol{d} = 1 \\
&= \arg\max_{\boldsymbol{d}} \operatorname{Tr}(\boldsymbol{d}^\top \boldsymbol{X}^\top \boldsymbol{X} \boldsymbol{d}) \text{ subject to } \boldsymbol{d}^\top \boldsymbol{d} = 1.
\end{aligned}
$$