动手学深度学习v2

第一周

# 动手学深度学习 v2

<https://zhuanlan.zhihu.com/p/38431213>

## [01 课程安排](file:///d:/BaiduSyncdisk/foam-template-master/研究生/深度学习/ppt/part-0_1.pdf::%PDF-1.4)

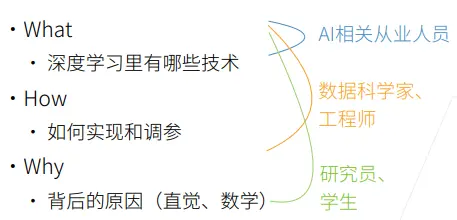
### 目标

介绍深度学习经典模型和最新模型：LeNet、ResNet、LSTM、BERT。。。 机器学习基础：损失函数、目标函数、过拟合、优化 实践：使用 pytorch 实现介绍的知识点；在真实数据上体验算法效果

### 内容

深度学习基础：线性神经网络、多层感知机 卷积神经网络：LeNet、AlexNet、VGG、Inception、ResNet 循环神经网络：RNN、GRU、LSTM、seq2seq 注意力机制：Attention、Transformer 优化算法：SGD、Momentum、Adam 高性能计算：并行、多 GPU、分布式 计算机视觉：目标检测、语义分割 自然语言处理：词嵌入、BERT

### 将会学到什么？

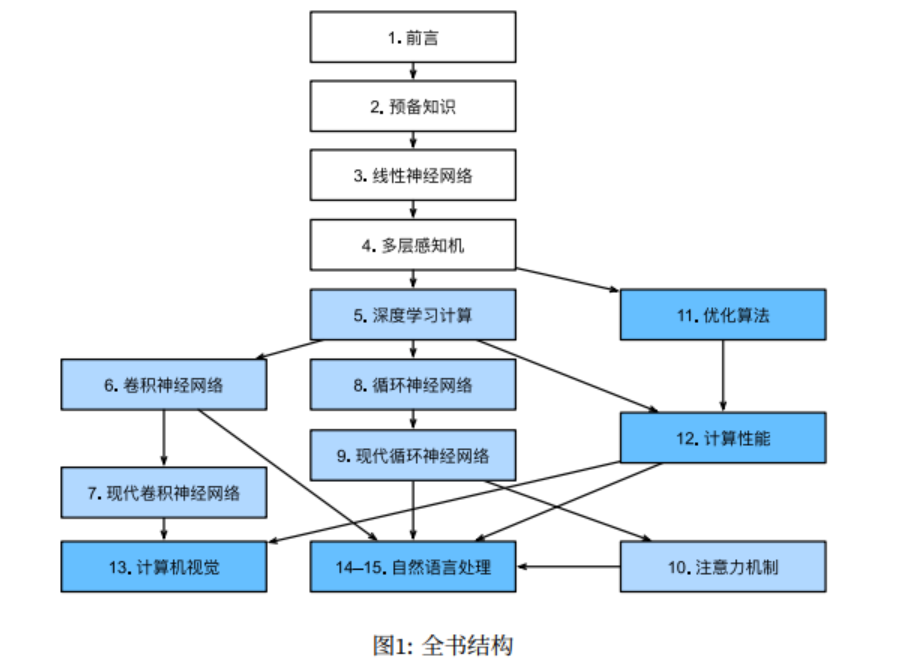


### 资源

END: 课程主页：https://courses.d2l.ai/zh-v2 教材：https://zh-v2.d2l.ai/ 课程论坛讨论：https://discuss.d2l.ai/c/16 Pytorch 论坛：https://discuss.pytorch.org/

## DONE 前言

### 图 1: 全书结构



### 02 [深度学习的介绍](file:///d:/BaiduSyncdisk/foam-template-master/研究生/深度学习/ppt/part-0_2.pdf)

1 图片分类 2 物体检测和分割 3 样式迁移 4 人脸合成 5 文字生成图片 6 文字生成 7 无人驾驶

预测结果：数据->特征提取->训练模型->输出结果 模型获取：以往数据+已有输出->模型

## 03 [安装](file:///d:/BaiduSyncdisk/foam-template-master/研究生/深度学习/ppt/part-0_3.pdf)

[深度学习软件的安装conda、jupyter notebook](id:2023-11-19T120056)

### 版本

$ python --version  
Python 3.9.18  
$ conda list scikit-learn  
# packages in environment at /home/jzdxdd/miniconda3/envs/d2l:  
#  
# Name Version Build Channel  
scikit-learn 1.0.2 py39h51133e4\_1

conda install scipy==1.11.3  
conda install numpy==1.22.2  
conda install pandas==1.4.0  
conda install matplotlib==3.5.1  
conda install scikit-learn==1.0.2  
  
conda install -c conda-forge xgboost # 从conda-forge渠道安装  
conda install -c conda-forge lightgbm

## 符号

## 引言

### 日常生活中的机器学习

### 机器学习中的关键组件

1. 数据
2. 模型
3. 目标函数
4. 优化算法

### 各种机器学习问题

1. 监督学习
   1. 回归
   2. 分类
   3. 标记问题
   4. 搜索
   5. 推荐系统
   6. 序列学习
2. 无监督学习
3. 与环境互动
4. 强化学习

### 起源

### 深度学习的发展

### 深度学习的成功案例

### 特点

## 预备知识

### DONE [04 数据操作](file:///d:/BaiduSyncdisk/foam-template-master/研究生/深度学习/ppt/part-0_4.pdf)

[ipynb](file:///d:/BaiduSyncdisk/foam-template-master/研究生/深度学习/d2l-zh/pytorch/chapter_preliminaries/ndarray.ipynb)

1. 入门

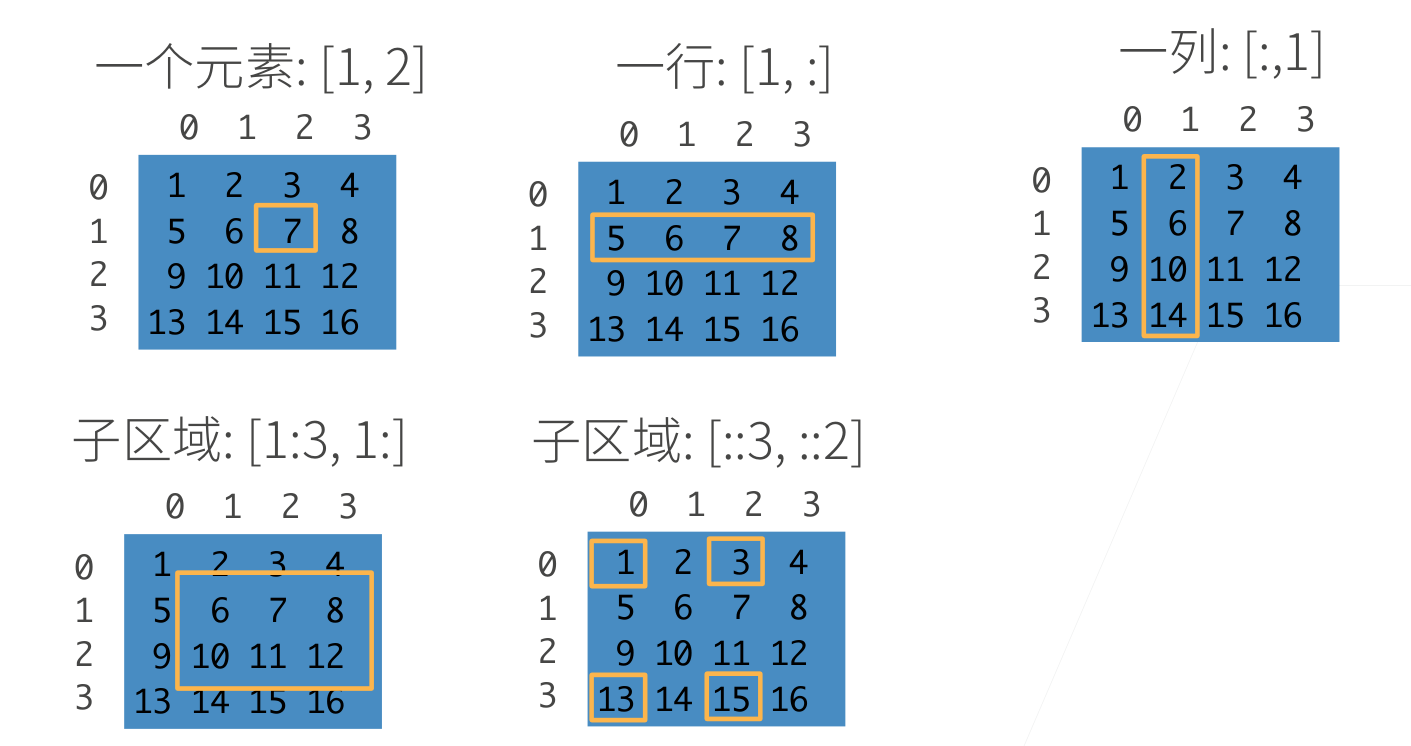
|  |  |
| --- | --- |
| touch.arrange(num) | 创建行向量 |
| .shape | 访问形状 |
| .numel() | 返回大小 |
| .reshape(num1,num2) | 改变形状不改变元素数量和值 |
| torch.zeros((num1,num2,num3)) | 全 0 |
| torch.ones((num1,num2,num3)) | 全 1 |
| torch.randn(3,4) | 正态随机 |

1. 运算符

* 按元素（elementwise）运算
  1. 连结（concatenate）
  + 轴 0⇒行拼 轴 1⇒列拼
  + torch.cat((X, Y), dim=0), torch.cat((X, Y), dim=1)
  1. 比较
  2. .sum()求和

1. 广播机制

* 矩阵若缺少则复制另外矩阵的相同位置

1. 索引和切片
   1. 访问元素:与[)
   * 
   1. n 维数组
      1. n 维数组，也称为张量（tensor）
      2. 创建数组需要
      * 形状、每个元素数据类型、每个元素的值
2. 节省内存

* id()函数：提供了内存中引用对象的确切地址。 用切片表示法执行原地操作：
* Y[:]= <expression>

1. 转换为其他 Python 对象

* torch 张量和 numpy 数组将共享它们的底层内存 .item

### DONE 数据预处理

pandas 软件包

1. 读取数据集

* import os  
  os.makedirs(os.path.join('..', 'data'), exist\_ok=True)  
  data\_file = os.path.join('..', 'data', 'house\_tiny.csv')  
  with open(data\_file, 'w') as f:  
   f.write('NumRooms,Alley,Price\n') # 列名  
    
  # !pip install pandas  
  import pandas as pd  
    
  data = pd.read\_csv(data\_file)  
  print(data)

1. 处理缺失值
   1. 方法：插值法、删除法
   2. 位置索引 iloc
   3. inputs.mean()的版本问题
   * data.iloc[:, 0:2] 当使用 Pandas2.\*版本的时候，将无法通过 inputs.mean()直接进行填充。需要选中相应的列，计算均值，然后相应处理。 其次，在 Pandas2.\*版本下，pd.getdummies()返回值将是 True 与 False 的形式，但与 1, 0 是通用的。 如果遇到 inputs = inputs.fillna(inputs.mean())报错：TypeError: can only concatenate str (not “int”) to str 可以采用：inputs = inputs.fillna(inputs.selectdtypes(include='number').mean())
2. 转换为张量格式

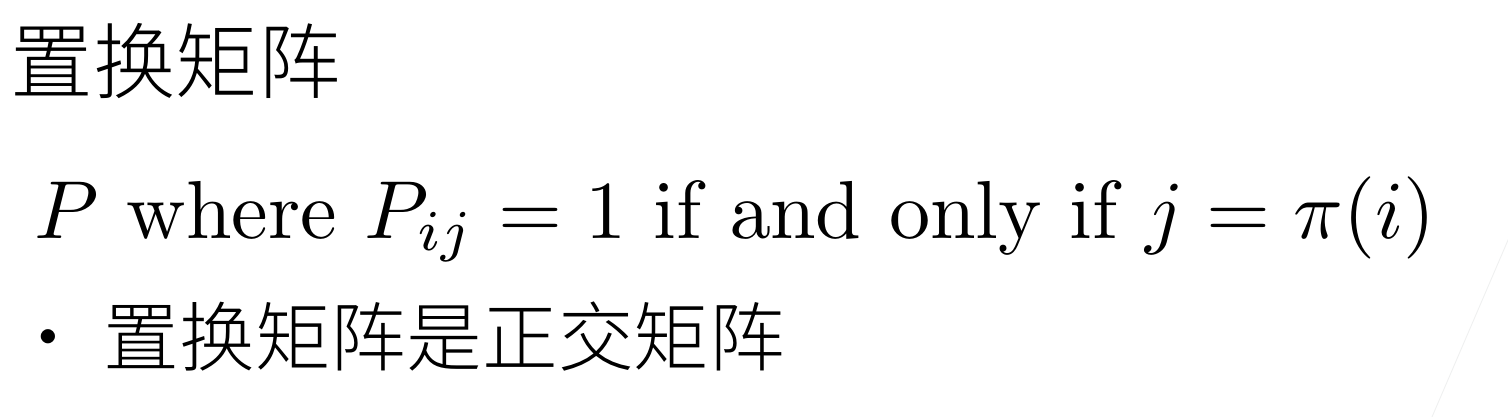
* .tonumpy

### TODO 线性代数

1. 标量

* 指数 x\*\*y

1. 向量
   1. 长度、维度和形状
   * .arrange  
     len(x)  
     x.shape
2. 矩阵

* 矩阵转置 A.T
  1. 特殊矩阵
     1. 正交矩阵
     2. TODO 置换矩阵
     + 置换矩阵就是重新排列后的单位矩阵 

1. 张量
2. 张量算法的基本性质
   1. 矩阵复制
   * B = A.clone()
   1. 矩阵乘法
   * 两个矩阵的按元素乘法称为 Hadamard 积（Hadamard product） A\*B
3. 降维

* A\_sum\_axis0 = A.sum(axis=0)  
  A\_sum\_axis1 = A.sum(axis=1)  
  A.sum(axis=[0, 1]) # 结果和A.sum()相同  
    
  A.mean(), A.sum() / A.numel()  
  A.mean(axis=0), A.sum(axis=0) / A.shape[0]
  1. 非降维求和
  + sum\_A = A.sum(axis=1, keepdims=True)
  + 沿某个轴计算 A 元素的累积总和，比如 axis=0（按行计算），可以调用 cumsum 函数
  + A.cumsum(axis=0)

1. 点积（Dot Product）

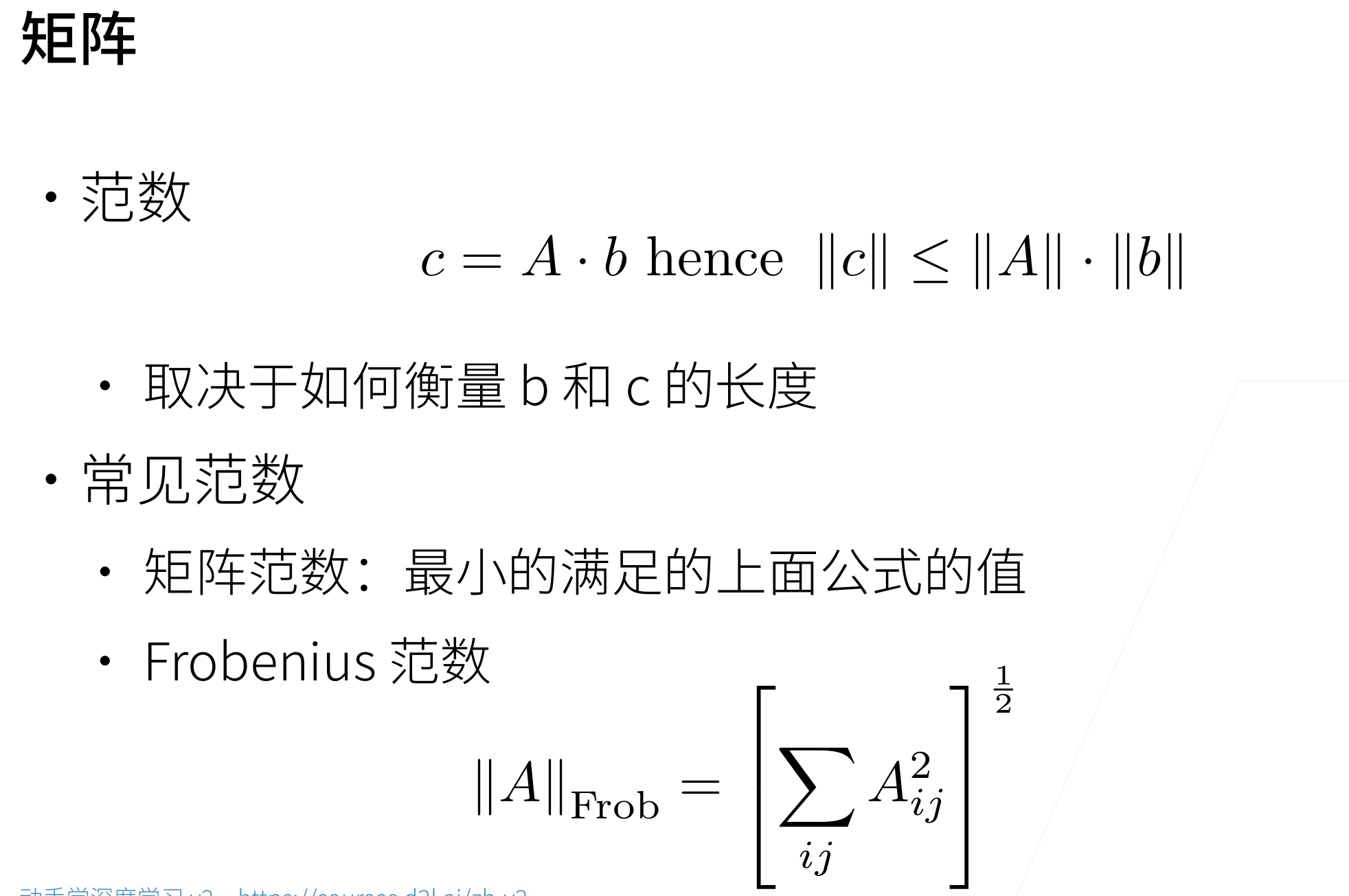
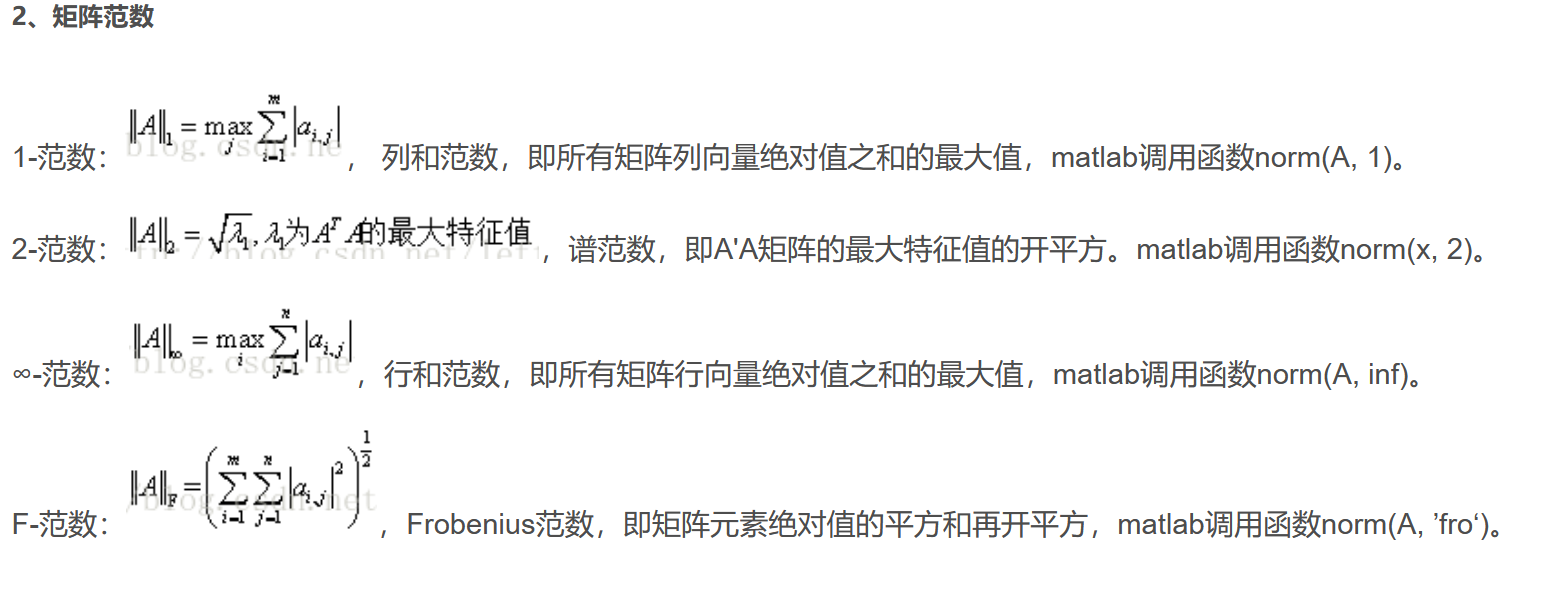
* torch.dot(x, y)
* 当权重为非负数且和为 1（即$ ( ∑ \_ { i = 1 } ^ { d } w \_ { i } = 1 ) $）时，点积表示加权平均（weighted average）。

1. TODO 矩阵-向量积

* torch.mv(A, x)
* <https://blog.csdn.net/zhinanpolang/article/details/104957513>

1. 矩阵-矩阵乘法

* torch.mm(A, B)

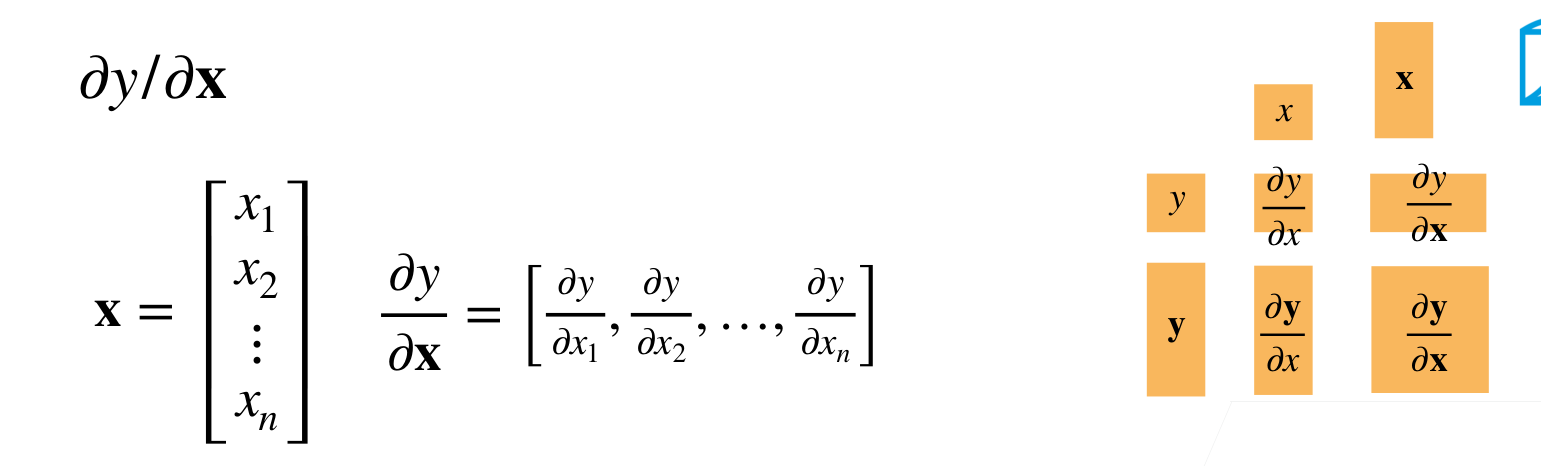
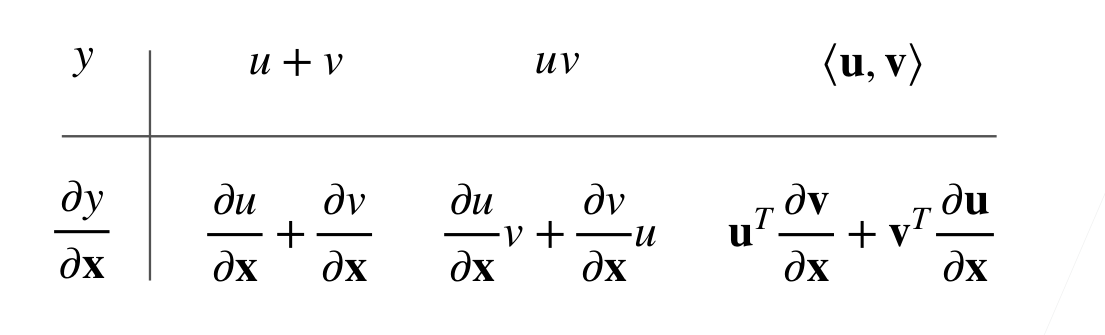
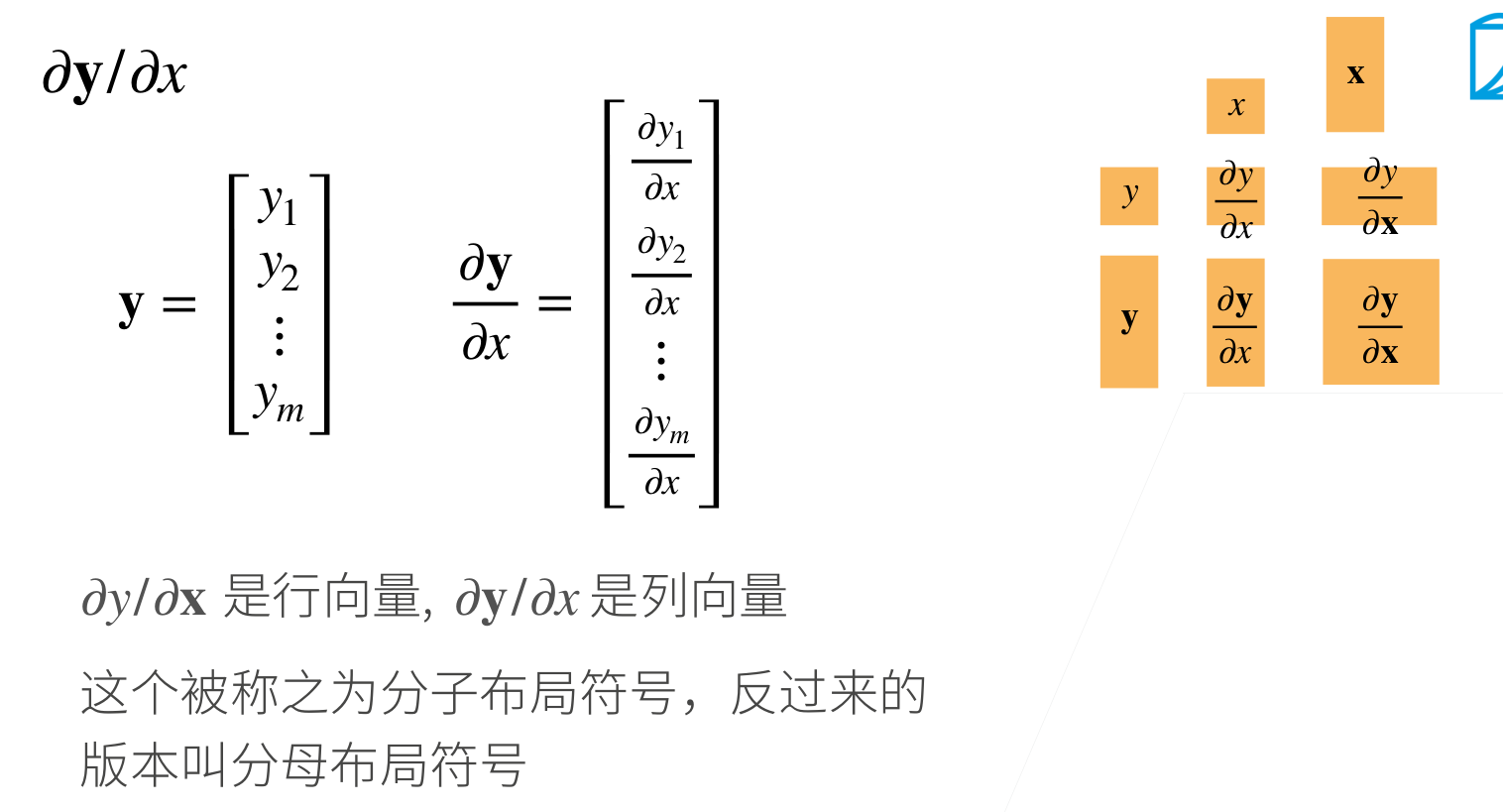
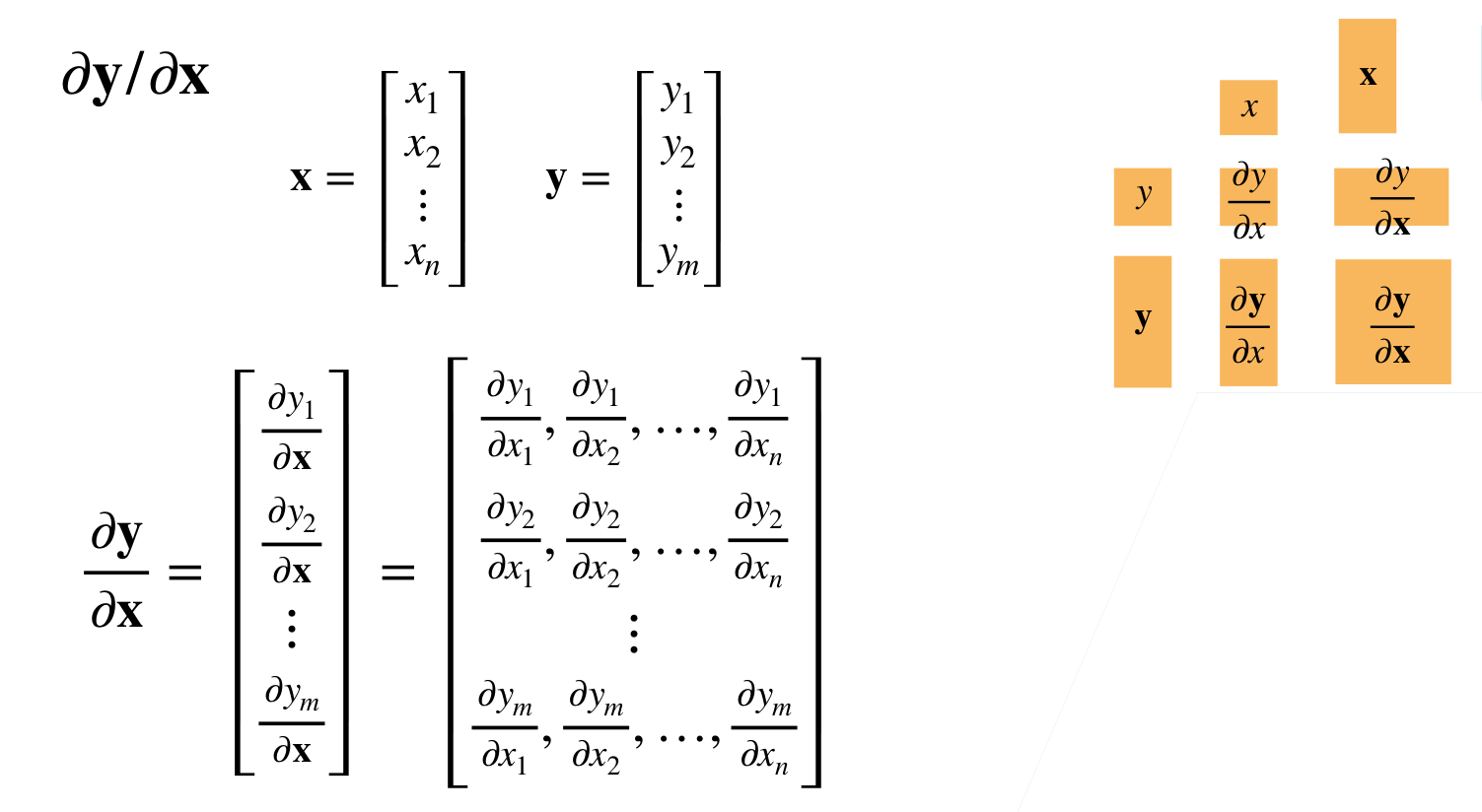
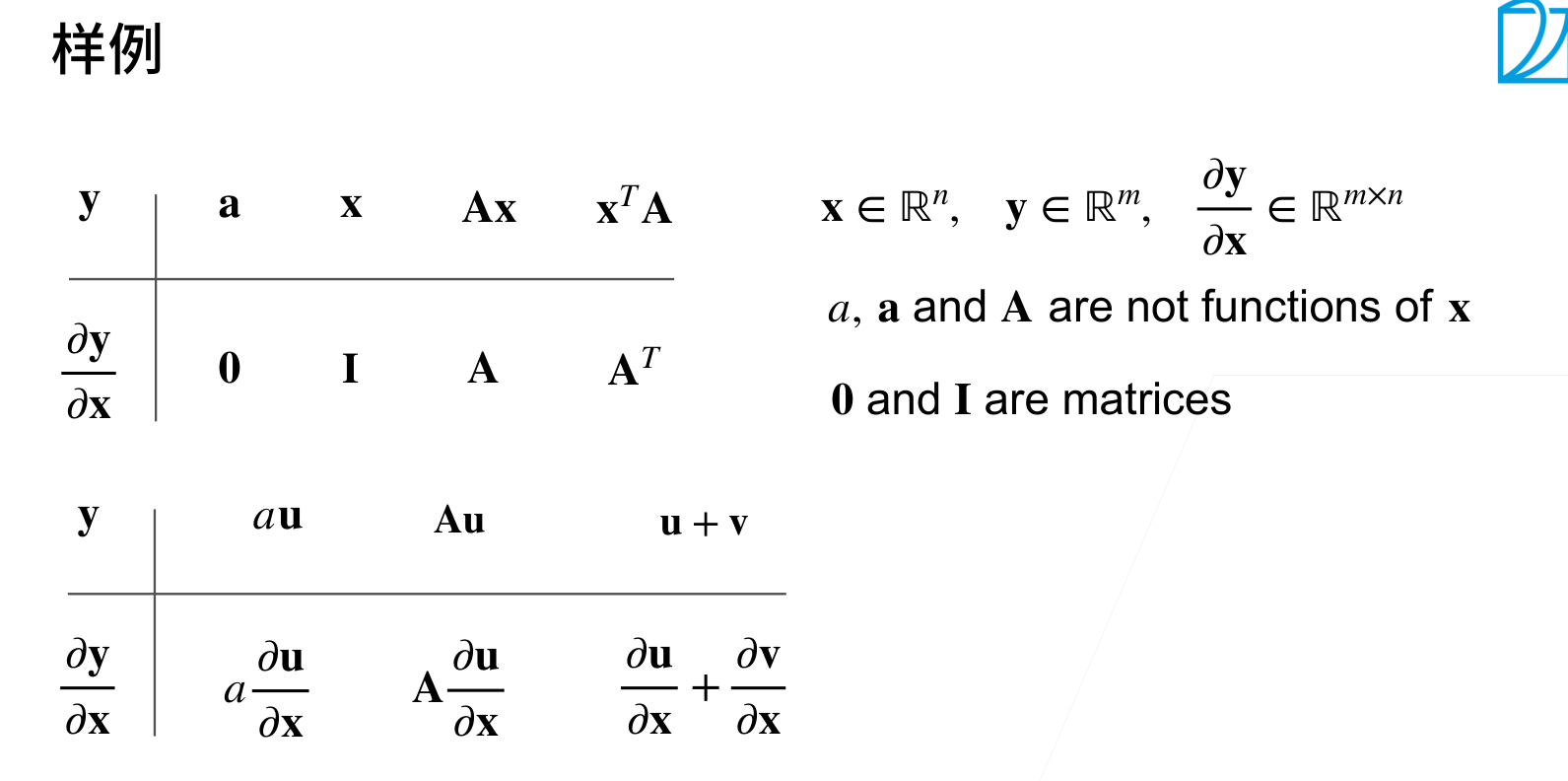
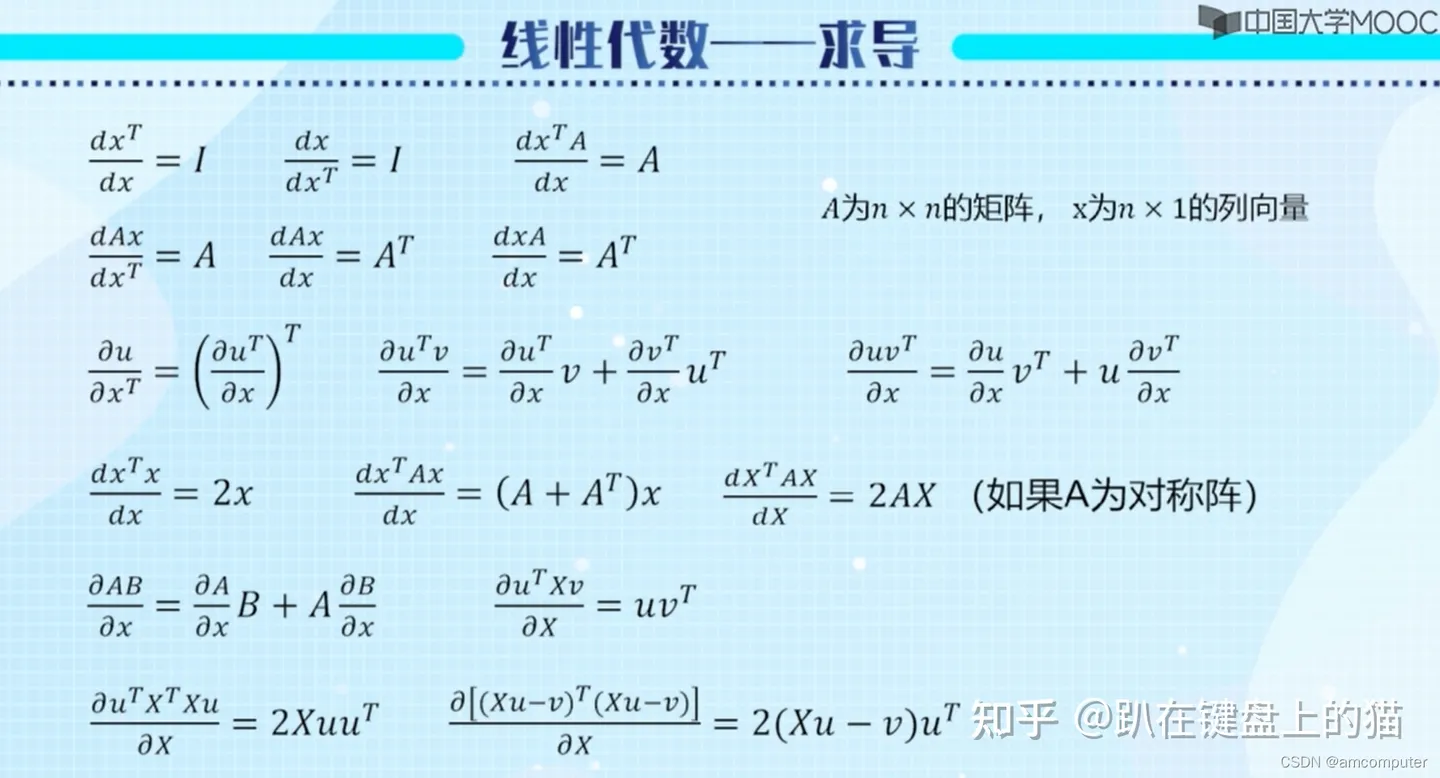
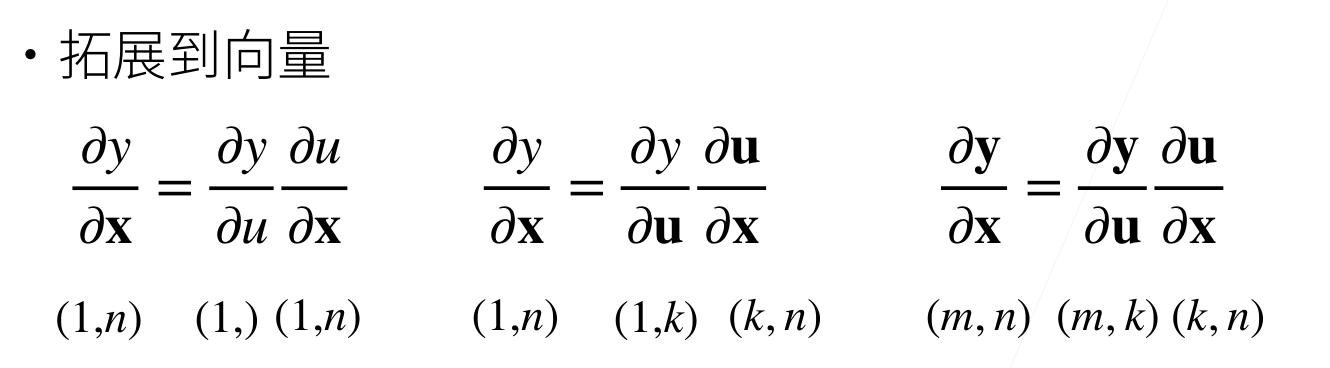
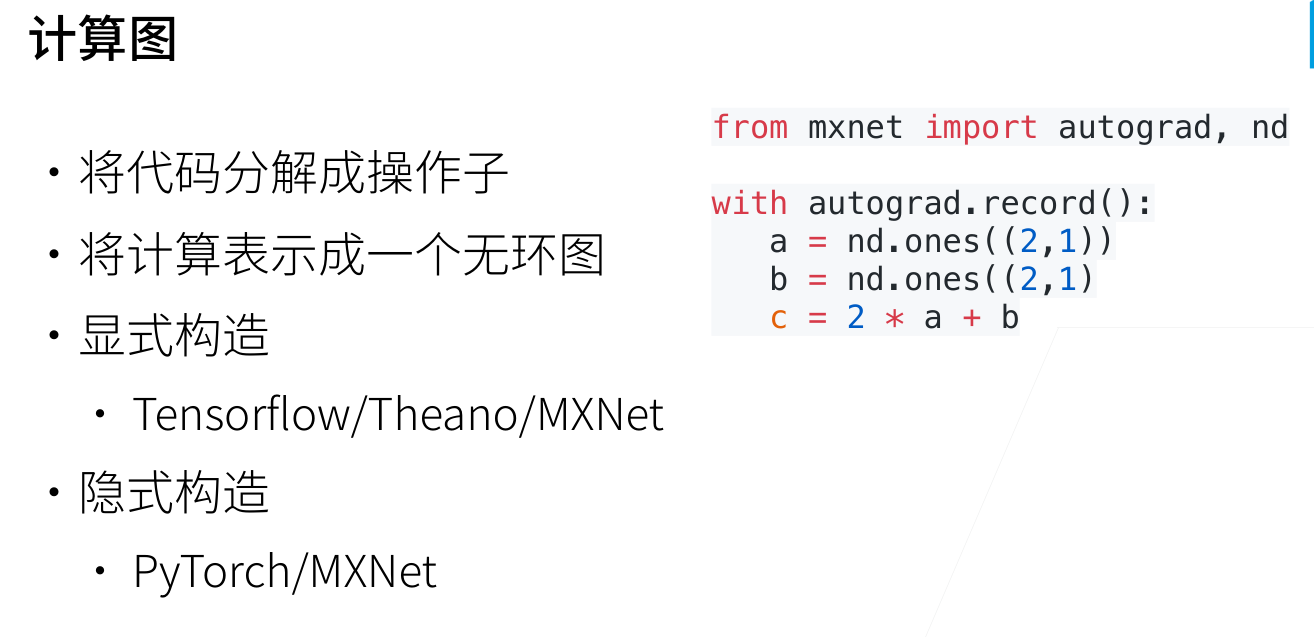
1. 范数（norm）
   1. 向量范数的性质
   * 向量范数是将向量映射到标量的函数 f $ f(**x**+**y**) ≤ f(**x**)+f(**y**)$ $f(**x**) ≥ 0 $
   1. $L2$范数
   * torch.norm(u)
   1. $L1$范数
   * torch.abs(u).sum()
   1. $Lp$范数
   2. 矩阵的范数
      1. TODO 矩阵范数
      * <https://zhuanlan.zhihu.com/p/507328276>  
      1. Frobenius 范数
      * 矩阵 的 Frobenius 范数 (Frobenius norm) 是矩阵元素平方和的平方根: $\|**X**\|F= $ **Frobenius** 范数满足向量范数的所有性质，它就像是矩阵形向量的 L2 范数
      * torch.norm(torch.ones((4, 9)))
2. 范数和目标

* 试图解决优化问题：最大化分配给观测数据的概率; 最小化预测和真实观测之间的距离。

1. 关于线性代数的更多信息

* <https://d2l.ai/chapter_appendix-mathematics-for-deep-learning/geometry-linear-algebraic-ops.html>

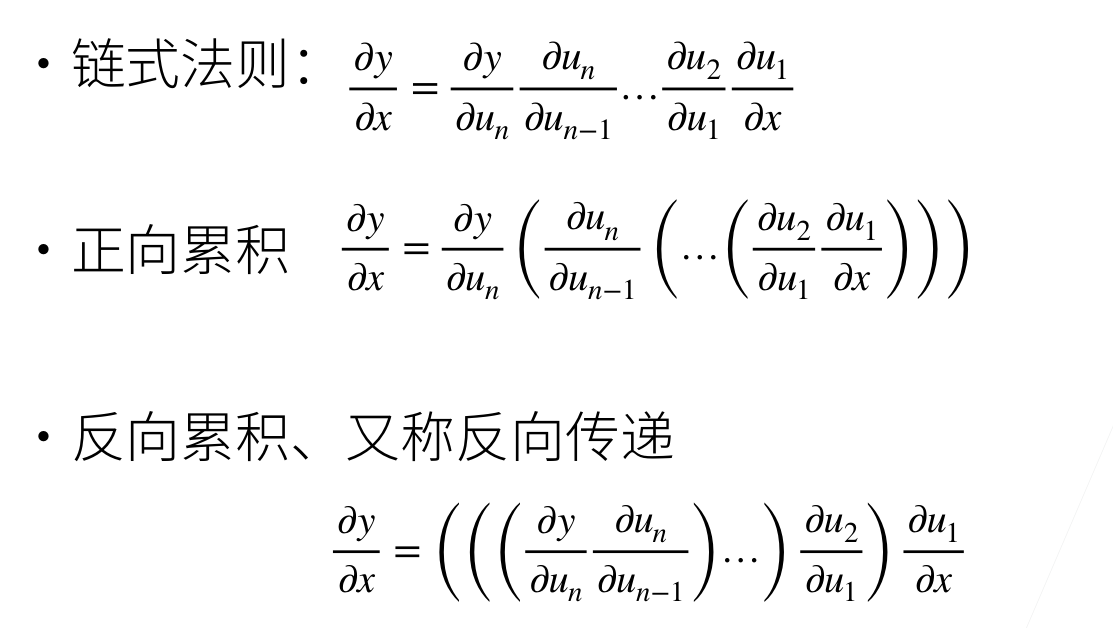
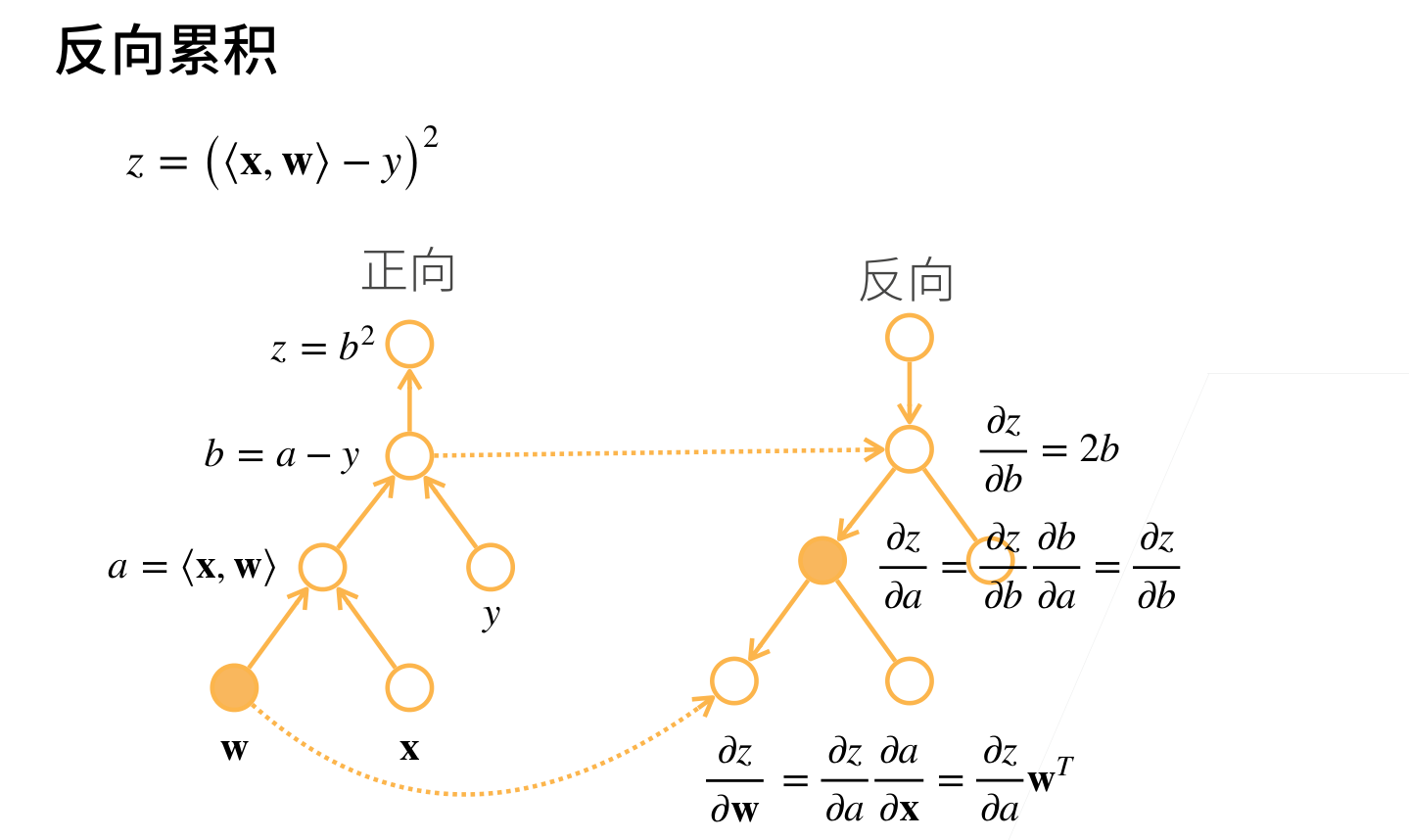
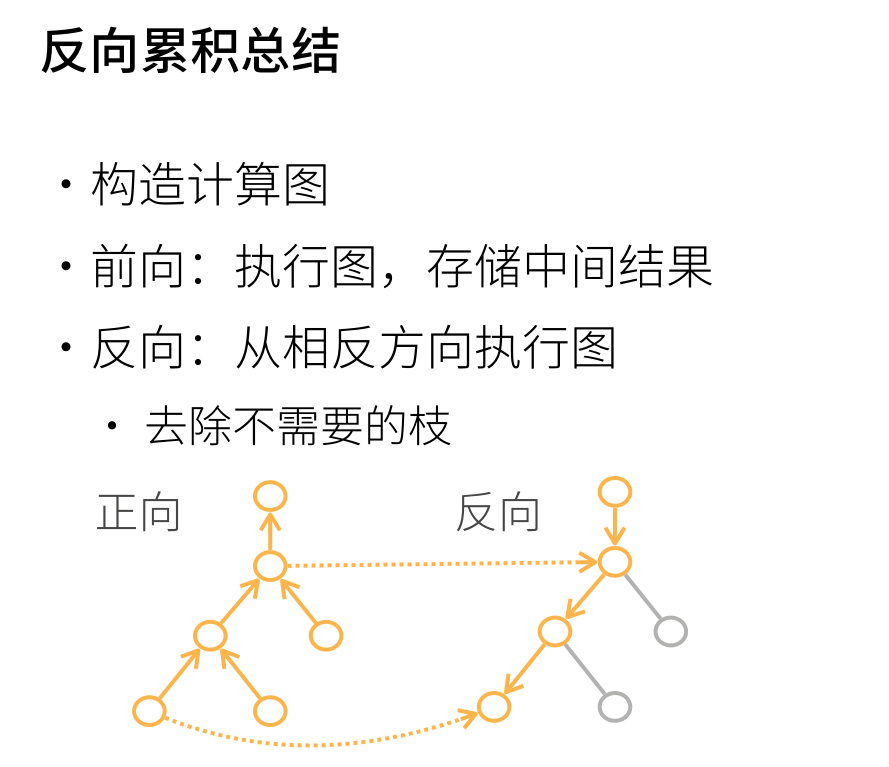
### TODO 微积分

1. 将拟合模型的任务分解为两个关键问题：
   1. 优化（optimization）：
   * 用模型拟合观测数据的过程；
   1. 泛化（generalization）：
   * 数学原理和实践者的智慧，能够指导我们生成出有效性超出用于训练的数据集本身的模型。
2. 导数和微分
   1. %matplotlib inline
   * 是在 Jupyter Notebook 中使用的魔术命令（magic command），它用于在 notebook 中嵌入 matplotlib 绘图，并将图像直接显示在 notebook 中而不是弹出一个新窗口。
   1. f-string 来格式化输出
   2. 注释#@save 是一个特殊的标记，会将对应的函数、类或语句保存在 d2l 包
   * #@save  
     # X和Y轴的数据，x轴和y轴的标签，图例，坐标轴的范围等  
     def plot(X, Y=None, xlabel=None, ylabel=None, legend=None, xlim=None,  
      ylim=None, xscale='linear', yscale='linear',  
      fmts=('-', 'm--', 'g-.', 'r:'), figsize=(3.5, 2.5), axes=None):  
      """绘制数据点"""  
      # 如果没有传入图例参数，则将图例设置为空列表  
      if legend is None:  
      legend = []  
       
      # 调用set\_figsize函数设置图形的大小  
      set\_figsize(figsize)  
      # 如果没有传入axes参数，则获取当前活动的图形对象。  
      axes = axes if axes else d2l.plt.gca()  
       
      # 如果X有一个轴，输出True  
      def has\_one\_axis(X):  
      return (hasattr(X, "ndim") and X.ndim == 1 or isinstance(X, list)  
      and not hasattr(X[0], "\_\_len\_\_"))  
       
      # 如果X只有一个轴，则将其转换为列表  
      if has\_one\_axis(X):  
      X = [X]  
       
      # 如果Y为空，则将X和Y都设置为长度为X长度的空列表。如果Y只有一个轴，则将其转换为列表。  
      if Y is None:  
      X, Y = [[]] \* len(X), X  
      elif has\_one\_axis(Y):  
      Y = [Y]  
       
      # 如果X和Y的长度不相等，则将X复制为与Y相同长度的列表  
      if len(X) != len(Y):  
      X = X \* len(Y)  
       
      # 清除当前图形对象的所有轴  
      axes.cla()  
       
      for x, y, fmt in zip(X, Y, fmts):  
      if len(x):  
      axes.plot(x, y, fmt)  
      else:  
      axes.plot(y, fmt)  
       
      # 使用zip函数将X、Y和fmts进行迭代，根据fmts中的格式绘制数据点。  
      set\_axes(axes, xlabel, ylabel, xlim, ylim, xscale, yscale, legend)
   1. 绘制图形
   * x = np.arange(0, 3, 0.1)  
     plot(x, [f(x), 2 \* x - 3], 'x', 'f(x)', legend=['f(x)', 'Tangent line (x=1)'])  
       
     import numpy as np  
     from d2l import torch as d2l  
     import os  
     os.environ["KMP\_DUPLICATE\_LIB\_OK"]="TRUE"  
     x = np.arange(0.5, 3, 0.2)  
     d2l.plot(x, [x \*\* 3 - 1 / x, 4 \* x - 4], 'x', 'f(x)', legend=['f(x)', 'Tangent line (x=1)'])  
     d2l.plt.show();
3. 偏导数
4. TODO [06梯度](file:///mnt/d/BaiduSyncdisk/foam-template-master/研究生/深度学习/ppt/part-0_6.pdf)
   * 导数拓展到向量  
   * 
   *  
   1. TODO 矩阵求导
   * 假设$**x**$为$n$维向量，在微分多元函数时经常使用以下规则:
   * 对于所有$**A** ∈ ℝm × n$，都有 对于所有$**A** ∈ ℝn × m$，都有 对于所有$**A** ∈ ℝn × n$，都有
   * 同样，对于任何矩阵$**X**$，都有$∇**X** \|**X** \|F2 = 2**X**$。 正如我们之后将看到的，梯度对于设计深度学习中的优化算法有很大用处。
   1. TODO 矩阵公式参考
      1. 分子布局与分母布局
      * 视频里采用的都是分子布局 在套公式的时候注意布局，一般是分母布局 “直观上讲，分子布局，就是分子是列向量形式，分母是行向量形式。分母布局则相反”。
      1. “分子布局的本质：
      * 分子是标量、列向量、矩阵向量化后的列向量 分母是标量、列向量转置后的行向量、矩阵的转置矩阵、矩阵向量化后的列向量转置后的行向量。
      * 那么，第一个分子标量，分母列向量的形式。因为是分子布局，分子标量不用管，分母是列向量，根据第二句话，列向量需要转置成行向量。标量对行向量求偏导，也就是行向量。
      * 第二个分子列向量，分子标量。符合分子布局，直接求偏导，所以为列向量。
      * 第三个分子列向量，分母列向量。根据第二句话，分母应该转置成行向量。之后就根据沐神在视频里的解释，将分子的一个个元素分别对分母求偏导，得到多个行向量，而这些行向量以列向量的方式组合在一起，形成矩阵。
      * 而最后拓展到矩阵，因为是分子布局，总体来说就是对分子矩阵不变，分母矩阵转置。 [matric-cook-book](file:///mnt/d/BaiduSyncdisk/foam-template-master/研究生/深度学习/matrix-cook-book.pdf)  <https://zhuanlan.zhihu.com/p/629989192> <https://zhuanlan.zhihu.com/p/661750788> <https://www.zhihu.com/question/528230648> <https://www.zhihu.com/question/25399811> <https://www.zhihu.com/question/264192195>
      * <https://blog.csdn.net/daaikuaichuan/article/details/80620518>
      * 矩阵求导的本质与分子布局、分母布局的本质（矩阵求导——本质篇）：https://zhuanlan.zhihu.com/p/263777564 矩阵求导公式的数学推导（矩阵求导——基础篇）：https://zhuanlan.zhihu.com/p/273729929 矩阵求导公式的数学推导（矩阵求导——进阶篇）：https://zhuanlan.zhihu.com/p/288541909
5. 07 [链式法则](file:///mnt/d/BaiduSyncdisk/foam-template-master/研究生/深度学习/ppt/part-0_7.pdf)
   1. 向量链式法则
   * 
   1. TODO 计算图
   * 

### DONE 自动微分（automatic differentiation）

自动微分使系统能够随后反向传播梯度。 这里，反向传播（backpropagate）意味着跟踪整个计算图，填充关于每个参数的偏导数。

1. 一个简单的例子
   1. 一个标量函数关于向量 x 的梯度是向量，并且与 x 具有相同的形状
   2. .grad
   * x.requires\_grad\_(True) # 等价于x=torch.arange(4.0,requires\_grad=True)  
     x.grad # 默认值是None
   * 这段代码使用了 PyTorch 库来创建一个张量 x，并对其进行了一些操作。让我一步步解释：
   * x.requiresgrad(True)：通过调用 requiresgrad方法，将张量 x 设置为需要计算梯度。这意味着当我们对 x 进行操作和计算时，PyTorch 将会自动追踪并计算其梯度。 x = torch.arange(4.0, requiresgrad=True)：这是另一种创建具有梯度追踪功能的张量 x 的方式。torch.arange 函数用于创建一个从 0 到 3 的张量，步长为 1，并将 requiresgrad 参数设置为 True，表示该张量需要计算梯度。 x.grad：这是张量 x 的 grad 属性，用于获取计算得到的梯度值。在刚创建的时候，默认情况下，x.grad 的值是 None，表示尚未进行任何计算或者还未通过反向传播计算梯度。
   * 综合起来，这段代码创建了一个需要计算梯度的张量 x，并且展示了 x.grad 属性的默认值为 None。在进行实际计算和反向传播之后，才能得到 x.grad 的具体值。
   1. tensor(28., gradfn=<MulBackward0>)：
   * 这是一个 PyTorch 张量的输出。其中 28.是计算得到的结果值， 而 gradfn=<MulBackward0>表示该张量是通过乘法操作进行计算得到的， 并且具有梯度函数（gradfn）用于反向传播计算梯度
   1. 清除累计梯度
   * # 在默认情况下，PyTorch会累积梯度，我们需要清除之前的值  
     x.grad.zero\_()  
     y = x.sum()  
     y.backward()  
     x.grad
2. 非标量变量的反向传播

*   

1. 分离计算

* 将某些计算移动到记录的计算图之外 梯度不会向后流经 u 到 x .detach

1. Python 控制流的梯度计算

* 即使构建函数的计算图需要通过 Python 控制流（例如，条件、循环或任意函数调用），我们仍然可以计算得到的变量的梯度

### TODO 概率

1. 基本概率论
   1. 概率论公理
   2. 随机变量
2. 处理多个随机变量
   1. 联合概率
   2. 条件概率
   3. 贝叶斯定理
   4. 边际化
   5. 独立性
   6. 应用
3. 期望和方差

### 查阅文档

1. 查找模块中的所有函数和类
2. 查找特定函数和类的用法

## 线性神经网络

### 线性回归

1. 线性回归的基本元素
   1. 线性模型
   2. 损失函数
   3. 解析解
   4. 随机梯度下降
   5. 用模型进行预测
2. 矢量化加速
3. 正态分布与平方损失
4. 从线性回归到深度网络
   1. 神经网络图
   2. 生物学

### 线性回归的从零开始实现

1. 生成数据集
2. 读取数据集
3. 初始化模型参数
4. 定义模型
5. 定义损失函数
6. 定义优化算法
7. 训练