黄家曦 2022141460084
部分图和公式会在书和网上参考
部分图来源：知乎 《open mlsys》
部分公式来源：《机器学习》《open mlsys》

1. 向量是什么?在机器学习中向量可以用来表示什么?

向量是表示具有大小和方向的量。在几何学中，向量通常被可视化为带箭头的线段，其中箭头的方向代表向量的方向，而线段的长度代表向量的大小。同时向量可以表达二维坐标、三维坐标甚至更高维坐标系某个点的位置。向量支持多种运算，包括加法、数乘、点积（内积）、叉积等。
在机器学习中，向量几乎无处不在。比如：

- 一组具有不同特征的数据，比如房屋大小，房屋楼层、房屋价格，可以用一组3*1的向量表述
- 神经网络模型的权重，一般以矩阵的形式存在。
- Embedding中词在语义空间中对应的映射位置
- 神经网络输出的结果，比如LLM输出的token logits，分类任务的输出结果
- 具身智能中，机械臂关节对应在世界坐标系（world frame）中的位置和姿态

2. 如何度量不同向量的相似度?各自的优势如何?

在机器学习和数据科学中，度量不同向量之间的相似度是一项非常重要的任务。不同的相似度度量方法适用于不同的场景，选择合适的度量方法可以显著提升模型的性能。以下是几种常见的向量相似度度量方法及其优劣势分析：

1. 欧几里得距离（Euclidean Distance）

对于两个向量 $u = [u1, u2, \ldots, un]$ 和 $v = [v1, v2, \ldots, vn]$，欧几里得距离定义为：

$$d(u, v) = \sqrt{\sum_{i=1}^{n} (u_i - v_i)^2}$$

2. 余弦相似度（Cosine Similarity）

余弦相似度衡量两个向量之间的夹角余弦值，定义为：

$$Cosine\_Similarity(u, v) = \frac{u \cdot v}{\|u\| \cdot \|v\|}$$

其中，u·v 是向量的点积，‖u‖ 和 ‖v‖ 分别是向量的模长。

3. 曼哈顿距离（Manhattan Distance）

曼哈顿距离也称为城市街区距离，定义为：

$$d(u,v) = \sum_{i=1}^{n} |u_i - v_i|$$

4. 闵可夫斯基距离（Minkowski Distance）

闵可夫斯基距离是欧几里得距离和曼哈顿距离的泛化形式，定义为：

$$d(u,v) = (\sum_{i=1}^{n} |ui - vi|^p)^{1/p}$$

其中 p 是一个参数：

- 当 p=1 时，退化为曼哈顿距离。
- 当 p=2 时，退化为欧几里得距离。

5. 杰卡德相似系数（Jaccard Similarity）

对于二值向量（通常是集合的表示），杰卡德相似系数定义为：

$$Jaccard\_Similarity(u,v) = \frac{|A \wedge B|}{|A \vee B|}$$

其中 A 和 B 是两个向量对应的集合。

6. 皮尔逊相关系数（Pearson Correlation Coefficient）

皮尔逊相关系数衡量两个向量之间的线性相关性，定义为：

$$r = \frac{Cov(u,v)}{\sigma_u \sigma_v}$$

其中，Cov(u,v) 是协方差，σu 和 σv 是标准差。

7. 汉明距离（Hamming Distance）

汉明距离用于比较两个等长字符串或二值向量之间的差异，定义为：

$$d(u,v) = \sum_{i=1}^{n} \delta(ui, vi)$$

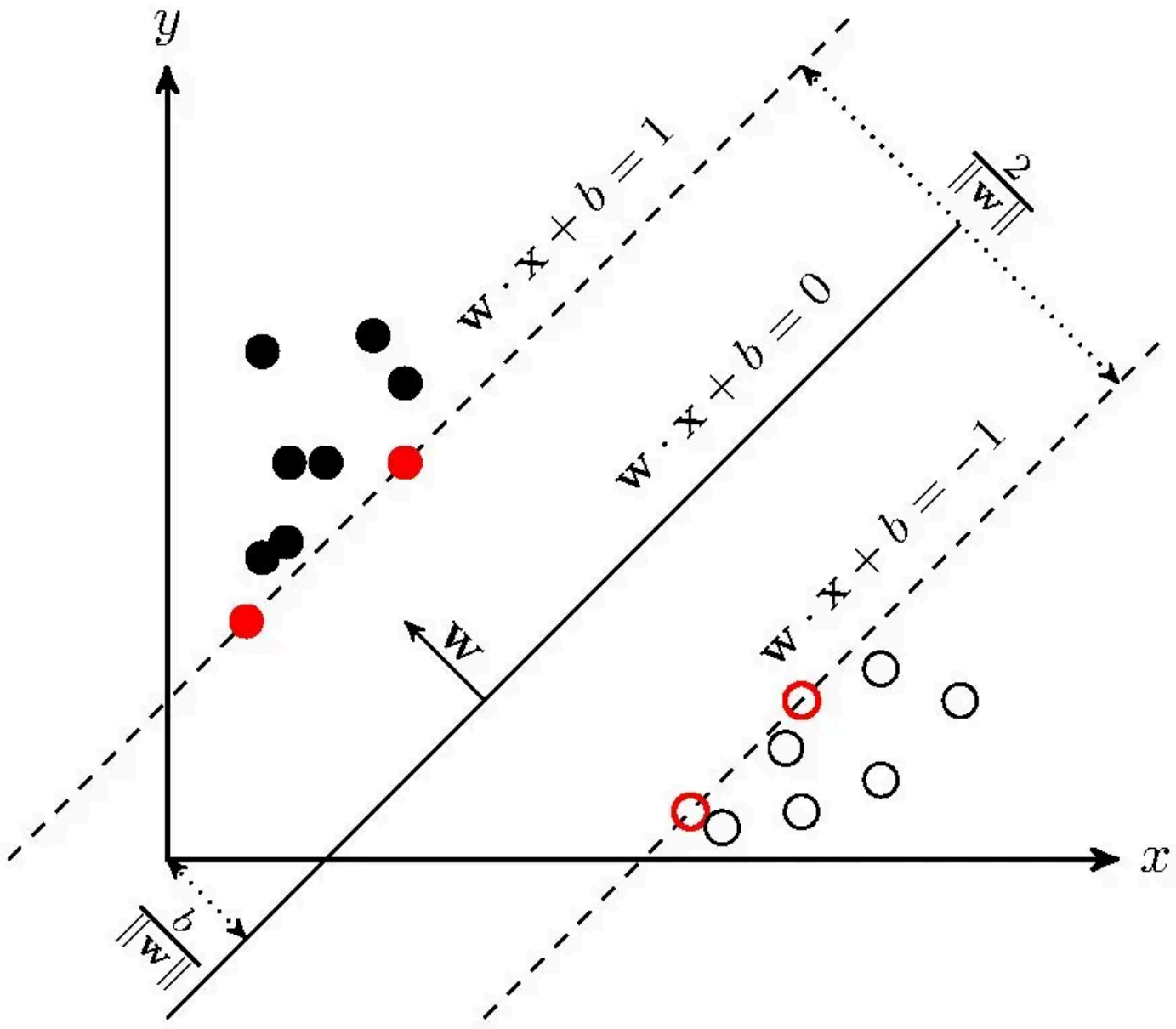
其中，δ(ui,vi) 是指示函数，当 ui!=vi 时为 1，否则为 0。

总结对比

方法	优点	缺点	适用场景
欧几里得距离	直观、易于实现	对高维数据敏感	图像处理、K-Means 聚类
余弦相似度	适合高维稀疏数据， 忽略大小	不考虑向量的绝对大小	文本分类、推荐系统
曼哈顿距离	计算简单，适合稀疏数据	不如欧几里得距离直观	网格搜索、路径规划
闵可夫斯基距离	泛化性强	高维数据中仍存在维度灾难	多种场景下的统一距离度量
杰卡德相似系数	适合稀疏数据和二值特征	不适用于连续型数据	文本分类、 用户行为相似性计算
皮尔逊相关系数	能捕捉线性相关性， 对平移和缩放鲁棒	仅限线性关系， 对异常值敏感	时间序列分析、 基因表达数据分析
汉明距离	适合二值特征，计算简单	仅适用于等长向量	编码理论、基因序列比对

3. 超平面是什么?它可以用来做什么?

对一个N维空间，超平面是这个N维空间对应的N-1维的子平面，可以将这N维空间分为两个空间。因此，超平面可以用来做分类问题，将数据集对应的多维空间通过若干个超平面分为若干个类。
SVM就是一个应用超平面的例子



给定训练样本集 $D = \{(X_1, Y_1)(X_2, Y_2), ..., (X_m, Y_m)\}, Yi \in \{-1, +1\}$。分类学习最基本的想法就是基于训练集 D 在样本空间中找到一个划分超平面、将不同类别的样本分开。但是超平面很多，怎么找到最好的那一个？SVM 的方法，直觉上就是找到在最“中间”的超平面，这样对样本的局部扰动容忍度最高。
在样本空间中，用以下线性方程划分描述超平面：

$$w^T x + b = 0 \quad w = (w_1; w_2; \cdots; w_d)$$

w为法向量，决定了超平面的方向，b决定超平面的位移。对于样本空间中的任意点x，到超平面的距离为

$$r = \frac{|w^T x + b|}{\|w\|}$$

对$w^T x + b = g$和$w^T x + b = -g$，我们认为大于g的y为1，小于$-g$的y为-1，若距离超平面最近的训练成本让这

两个条件成立，称它们为**support vector**，两个异类支持相连到超平面距离为:

$$r = \frac{2g}{\|w\|}$$

这里的r称为**间隔**
SVM的目的就是找到最大间隔，即$\max_{w,b} \frac{2g}{\|w\|}$
将最大化可以转为最小化，即$\min_{w,b} \frac{\|w\|^2}{2g}$

4. 什么是矩阵?你觉得矩阵在数据分析中的作用是什么?

矩阵是一个由数字、符号或表达式排列成的矩形阵列。可以把矩阵理解成多个向量的拼接。
矩阵的形式和思想被应用到了很多应用中，比如最常见的Excel，每一行代表一个观测值（样本），每一列代表一个特征（变量）。一个Excel表格就代表了一个矩阵。对于一个有多个特征的数据集也是如此，因此也可以利用矩阵运算，矩阵分析在机器学习中对数据集，对网络权重进行分析。

5. 什么是线性相关与线性无关?

- **线性相关** ：一组向量“共面”或“共线”。向量组成的方程组并不是“最小化”的，可以再化简。例如，在二维空间中，如果两个向量方向相同或相反（即成比例），它们就是线性相关的。这两个向量可以由同一个向量生成。
- **线性无关** ：一组向量彼此独立，不能通过其他向量的线性组合生成。例如，在三维空间中，三个不共面的向量是线性无关的。

6. 矩阵的秩描述了什么(即它的物理意义)?

矩阵的秩表示矩阵中线性无关行或列的最大数量。换句话说，矩阵的秩描述了矩阵的“信息量”或“自由度”的大小。对于一个 m×n 的矩阵 A，其秩记作 rank(A)，满足以下性质：

- rank(A)≤min(m,n)。
- 矩阵的秩等于其行空间或列空间的维数（两者相等）。
- 矩阵的秩可以通过高斯消元法、奇异值分解（SVD）或其他方法计算。
  矩阵秩的物理意义有很多, 比如：
- 如果矩阵的秩为 r，则该变换将输入空间压缩到一个 r-维子空间。矩阵的秩决定了线性变换后的输出空间的维度。例如，如果矩阵的秩小于其列数，则说明该变换会导致维度的“坍缩”。
- 矩阵的秩反映了数据的潜在结构和复杂度，PCA 通过计算协方差矩阵的秩，找到数据的主要方向（主成分），从而实现降维。
- 矩阵的秩反映了信号或图像中真正有意义的信息量：如果一个矩阵（如灰度图像）的秩较低，则说明该图像可以用较少的基向量近似表示。

7. 伪逆如何计算?为什么我们需要伪逆?

对于一个 m×n 的矩阵 A，其 Moore-Penrose 伪逆 A^+ 满足以下四个条件：

$$AA^+A = A$$
$$A^+AA^+ = A^+$$

$$(AA^+)^T = AA^+$$
$$(A^+A)^T = A^+A$$

对于一个矩阵A的SVD分解：
$A = U\sum V^T$

- U满足m*m的正交矩阵
- $\sum$是一个m*n的对角矩阵，对角线上为非负的奇异值
- V是一个n*n的正交矩阵
  伪逆可以通过以下式子计算：
  $A^+ = V\sum^+ U^T$
  $\sum^+$为$\sum$中非零奇异值取倒数后转置的矩阵。
  伪逆对数学问题和实际工程问题对有很大的作用：
- 对于非方阵和奇异矩阵，传统的逆矩阵不存在。
- 伪逆相对于逆矩阵，提供了一个更为广义的表达，可以让原本不可直接解的问题求得近似解。

8. 流形是什么?举几个现实生活中的流形的例子，并给出流形学习的定义。

流形的定义

流形（Manifold） 是数学中的一个重要概念，用于描述高维空间中具有低维结构的几何对象。简单来说，流形是一个局部看起来像欧几里得空间的拓扑空间。

数学定义

一个 d-维流形 M 是一个拓扑空间，满足以下条件：

1. **局部欧几里得性** ：对于流形上的每一点 $p \in M$，都存在一个邻域 U 和一个同胚映射 $\phi : U \to R^d$，使得该邻域与 R^d 的某个开集同胚。
2. **全局拓扑结构** ：流形可以通过多个局部坐标系（称为“图”或“chart”）拼接而成，且这些坐标系之间的转换是光滑的（在光滑流形的情况下）。

换句话说，尽管流形可能嵌入在高维空间中，但它的局部结构可以看作是一个低维的欧几里得空间。

现实生活中的流形例子

1. 地球表面（二维球面）

地球的表面是一个经典的二维流形。虽然它嵌入在三维空间中，但每个点的局部区域看起来像一个平面（即二维欧几里得空间）。例如，站在地球表面的某一点，你的视野范围内的地形可以近似看作平坦的。

2. 纸张的弯曲

一张纸在三维空间中可以被弯曲成各种形状（如卷成圆柱或折叠成复杂形状），但无论怎么弯曲，它的本质仍然是一个二维流形。因为纸张的每个局部区域都可以展平为二维平面。

3. 声波信号空间

在语音处理中，人类发出的声音信号通常位于一个高维空间中，但由于声音的生成机制（如声带振动和声道形状）受到物理限制，实际的信号分布往往集中在低维流形上。

4. 图像数据

在计算机视觉中，自然图像通常被认为是高维像素空间中的低维流形。例如，所有猫的图片可能分布在高维像素空间中的一个低维子空间上，这个子空间就是一个流形。

5. 机器人运动轨迹

机器人的关节角度空间可以看作一个流形。例如，一个机械臂的关节角度可能构成一个多维空间，但其运动轨迹通常受限于物理约束，从而形成一个低维流形。

流形学习的定义

流形学习（Manifold Learning） 是一种非线性降维技术，基于数据分布在一个低维流形上的假设。其目标是从高维数据中提取出低维表示，同时尽可能保留数据的内在结构。

核心思想

1. 高维数据通常具有冗余性，其本质信息可能集中在低维流形上。
2. 通过找到数据的低维流形结构，可以更好地理解数据的本质特征，并用于可视化、分类或回归等任务。

常见的流形学习算法

1. **主曲线与主曲面（Principal Curves and Surfaces）**
   找到一条或多条曲线/曲面来拟合数据分布。
2. **局部线性嵌入（Locally Linear Embedding, LLE）**
   假设每个数据点可以由其邻近点的线性组合表示，通过保持这种局部线性关系实现降维。
3. **等距映射（Isomap）**
   使用测地距离（数据点在流形上的最短路径）代替欧几里得距离，然后进行多维尺度分析（MDS）。
4. **拉普拉斯特征映射（Laplacian Eigenmaps）**
   构建数据的邻接图，并通过图拉普拉斯算子找到低维嵌入。
5. **t-SNE（t-Distributed Stochastic Neighbor Embedding）**
   通过概率分布匹配的方式，在低维空间中保留高维数据的局部结构。
6. **UMAP（Uniform Manifold Approximation and Projection）**
   一种高效的降维方法，结合了流形学习和拓扑数据分析的思想。

流形学习广泛应用在数据可视化、模式识别、图像处理、生物信息学、推荐系统中

9. 特征分解与奇异值分解的联系与不同

联系

- 两者都可以用于揭示矩阵的内在结构，对于对称正定矩阵 A，特征分解和奇异值分解的结果一致
- 在降维（PCA）、低秩近似、数据分析等领域有共同的应用

不同

- 特征分解要求矩阵是方阵，而奇异值分解适用于任意矩阵。
- 奇异值分解更通用、更稳定，尤其是在处理非方阵或病态矩阵时

10. F范数怎么计算?它可以用来度量什么?

对于一个 $m \times n$ 的矩阵 A，其 F范数定义为：

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}$$

即，F范数是矩阵所有元素平方和的平方根。
或者，也可以根据trace来计算

$$\|A\|_F = \sqrt{tr(A^TA)} = \sqrt{tr(AA^T)}$$

在许多优化问题中，F范数被用来衡量两个矩阵之间的差异。例如，在矩阵近似问题中，我们希望找到一个低秩矩阵 B 来近似原始矩阵 A，可以通过最小化 F范数误差来实现：

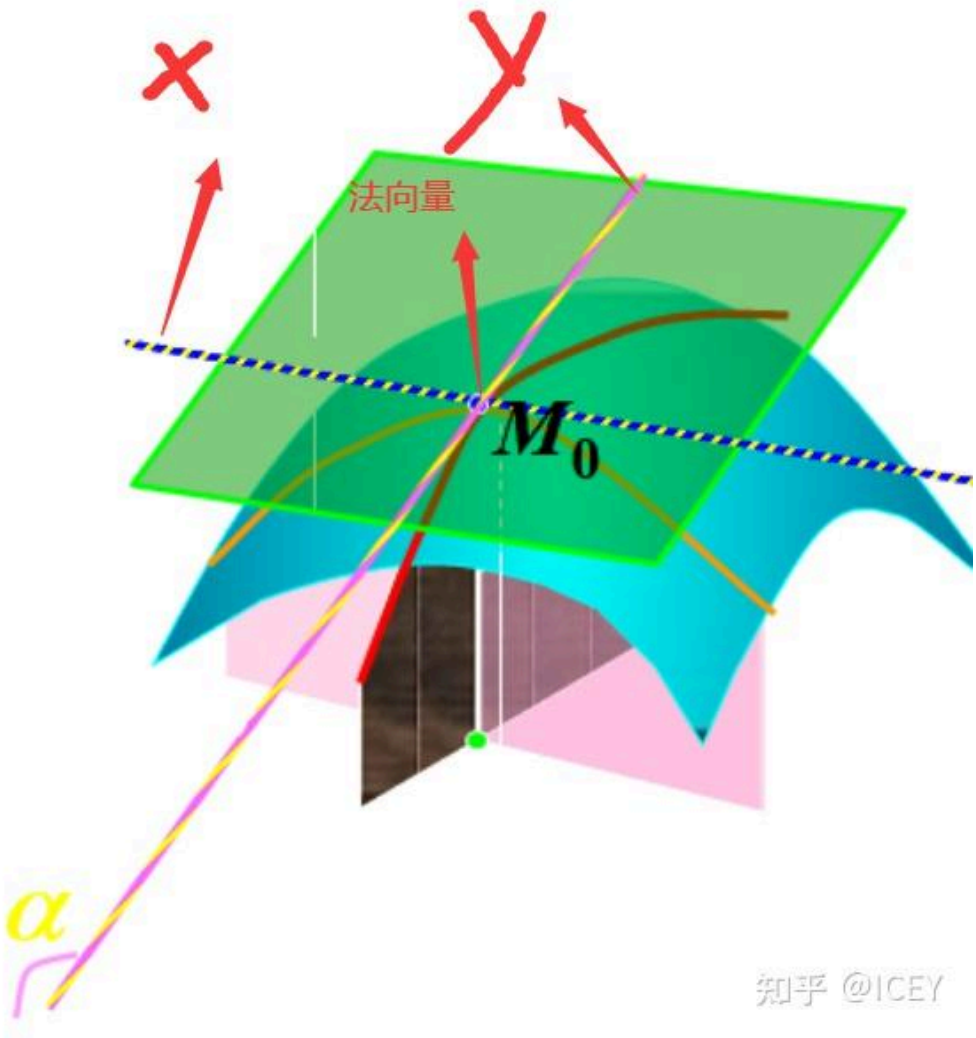
$$\min_B \|A - B\|_F$$

在主成分分析（PCA）中，F范数用于衡量数据投影后的误差。通过保留主要奇异值对应的奇异向量，可以最小化 F范数误差。
除此之外，F范数还可以用来正则化损失函数，防止过拟合，常用的有L1范数、L2范数

11. 给出微分的几何解释

一般的，对于多元函数$f(x_1, x_2, \cdots, x_n)$，其在点 $x_0 = (x_{1,0}, x_{2,0}, \cdots, x_{n,0})$ 处的微分定义为：

$$df = \frac{\partial f}{\partial x_1}dx_1 + \frac{\partial f}{\partial x_2}dx_2 + \cdots + \frac{\partial f}{\partial x_n}dx_n$$



在多元函数中，微分 df 描述了函数曲面在点 $x_0(M_0)$ 处的切平面，即曲面的局部线性近似。通过微分也可以求出梯度，从而示了函数值增加最快的方向。优化过程沿着负梯度方向移动，逐步逼近极小值。

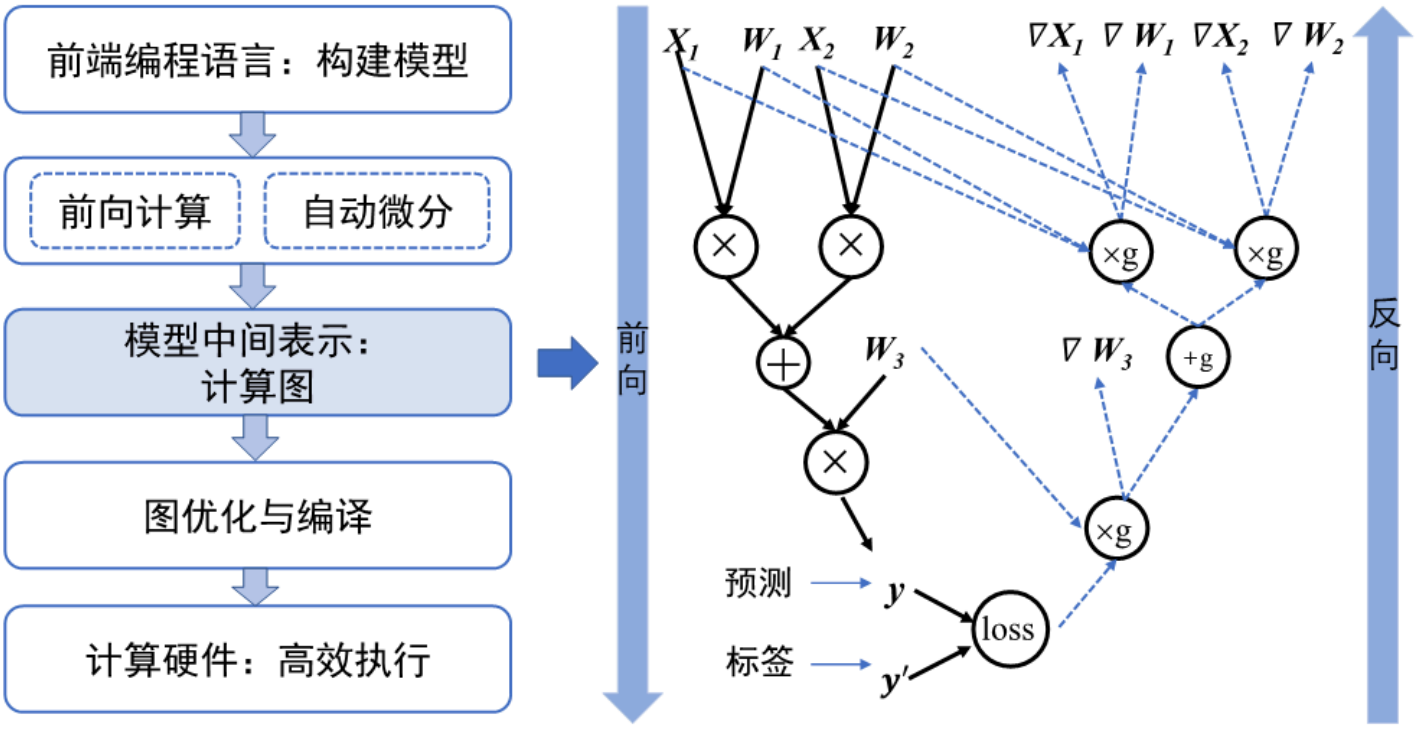
12. 梯度是如何计算的?

梯度的计算方法有很多，在传统的数学教学中梯度主要通过求导来计算。这里，我会结合机器学习系统中的计算图来介绍一下梯度下降和反向传播在实际框架中的大概的运行思路。
对于**简单的线性计算**：

$$Y_1 = a_1 \cdot x + b_1$$
$$Y = a_2 \cdot Y_1 + b_2$$

在计算梯度时，考虑应用链式法则即可。
对a_2,梯度为$loss * Y_1$
对a_1，梯度为$loss * Y_1 * x$
而考虑具有多个神经元和激活函数的神经网络，需要使用计算图来跟踪记录梯度反向传播中各层与各层应用链式法则的机制。

计算图



在实际的机器学习训练中，我们往往会借助机器学习框架，比如Torch、Tensorflow、MindSpore等，之所以这么做，一是这些框架提供了非常多的算子（Relu、卷积、Transformer等等），二是这些框架提供了静态或者动态计算图来自动构建计算流程，从而自动微分实现梯度下降，这在以往通过matlab或者纯c来实现是非常难的。而且更重要的是对于深度学习等非常消耗内存显存的领域，机器学习框架能够支持程序在GPU等并行化硬件上运行，大大提高运行效率，同时对于一些大模型训练，框架也可以通过支持分布式训练等特性在多个GPU集群上训练和推理模型。

对于梯度下降来说，在实际的神经网络中，输入数据张量往往不只一维，隐藏层会进行维度变换，最后一般会输出一维的张量。具体来说：

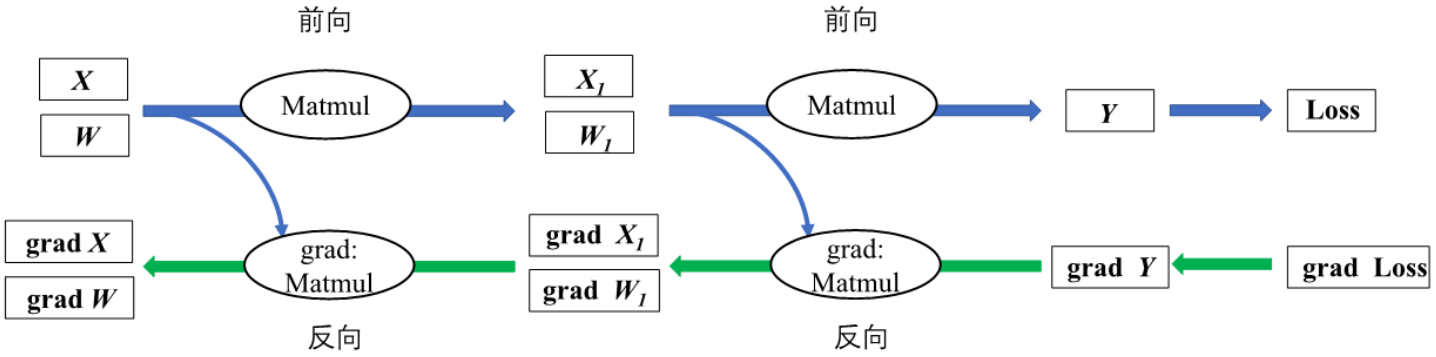
假设X是m维张量，Y为n维张量，z为一维向量，$Y=g(X)$并且$z=f(Y)$，则z关于X每一个元素的偏导数即为：

$$\frac{\partial_z}{\partial_{x_i}}=\sum_j \frac{\partial_z}{\partial_{y_i}}\frac{\partial_{y_i}}{\partial_{x_i}}$$

等价于：

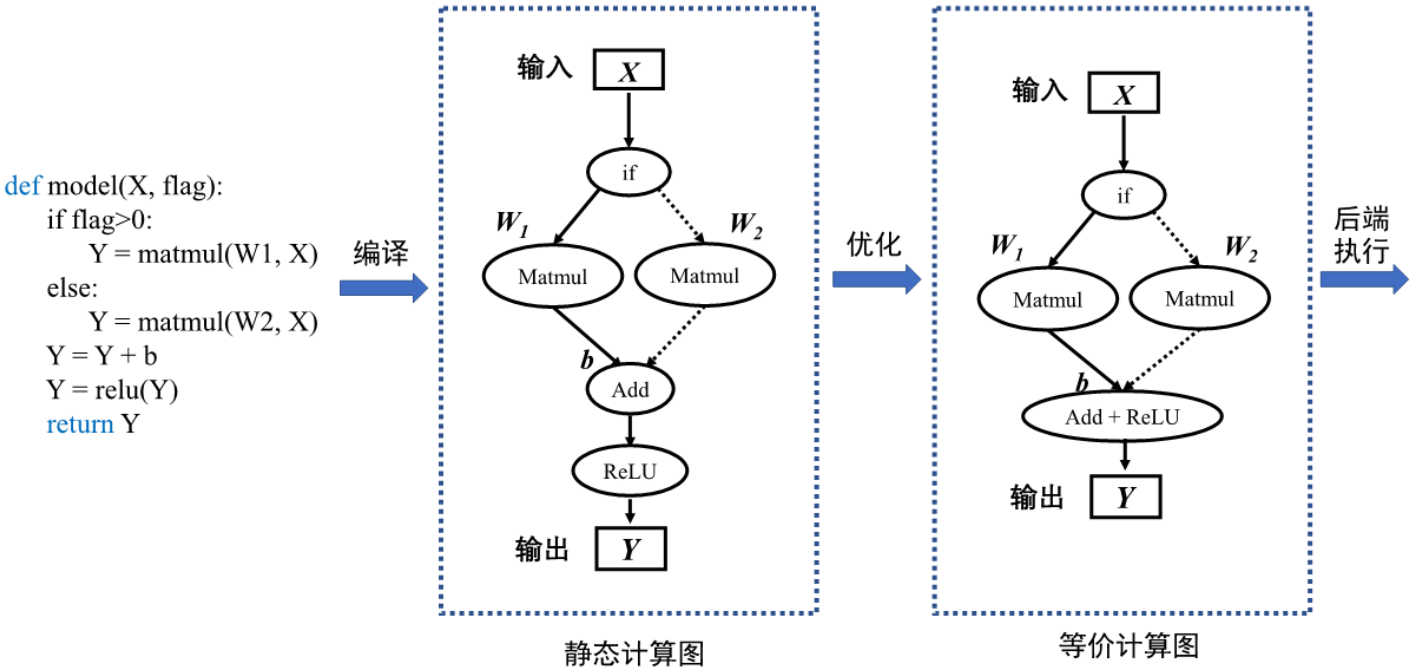
$$\nabla_x z=(\frac{\partial Y}{\partial X})^T\nabla_Y z$$

$\nabla_x z$是表示z关于X的梯度矩阵

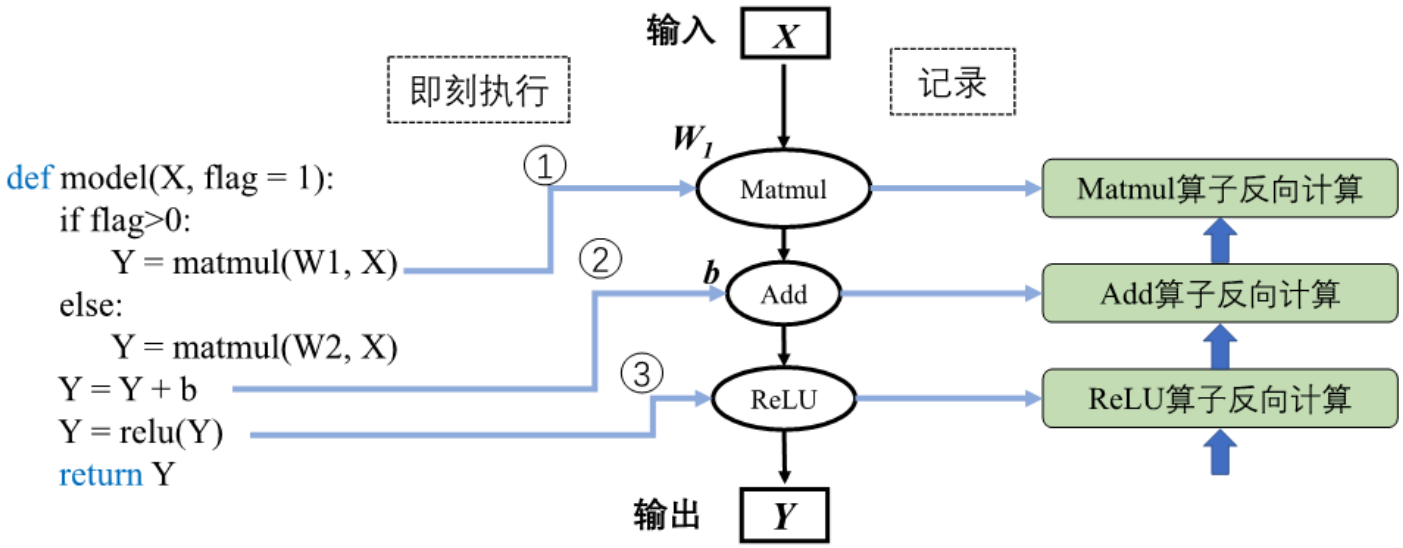


简单的正向计算和反向传播过程见上图，在具体计算过程中会保存和重复利用一些中间值，比如X_1、X通过生成计算图，可以普适性地应用上述计算过程。

对于静态图，使用前端语言定义模型形成完整的程序表达后，机器学习框架首先对神经网络模型进行分析，获取网络层之间的连接拓扑关系以及参数变量设置、损失函数等信息。然后机器学习框架会将完整的模型描述编译为可被后端计算硬件调用执行的固定代码文本，静态计算图直接接收数据并通过相应硬件调度执行图中的算子来完成任务。静态计算图可以通过优化策略转换成等价的更加高效的结构，提高后端硬件的计算效率。



而对于动态图,采用解析式的执行方式，其核心特点是编译与执行同时发生。动态图采用前端语言自身的解释器对代码进行解析，利用机器学习框架本身的算子分发功能，算子会即刻执行并输出结果。



神经网络前向计算按照模型声明定义的顺序进行执行。当模型接收输入数据X后，机器学习框架开始动态生成图拓扑结构，添加输入节点并准备将数据传输给后续节点。模型中存在条件控制时，动态图模式下会即刻得到逻辑判断结果并确定数据流向，对于上图的情形，每次执行完，都会根据flag或其他前段逻辑来动态生成不完整图。机器学习框架会在添加节点的同时完成算子分发计算并返回计算结果，同时做好准备向后续添加的节点传输数据。当模型再次进行前向计算时，动态生成的图结构则失效，并再次根据输入和控制条件生成新的图结构。相比于静态生成，可以发现动态生成的图结构并不能完整表示前端语言描述的模型结构，需要即时根据控制条件和数据流向产生图结构。由于机器学习框架无法通过动态生成获取完整的模型结构，因此动态图模式下难以进行模型优化以提高计算效率。

13. 局部最小值与全局最小值的关系。

- 全局最小值一定是局部最小值
- 局部最小值不一定是全局最小值

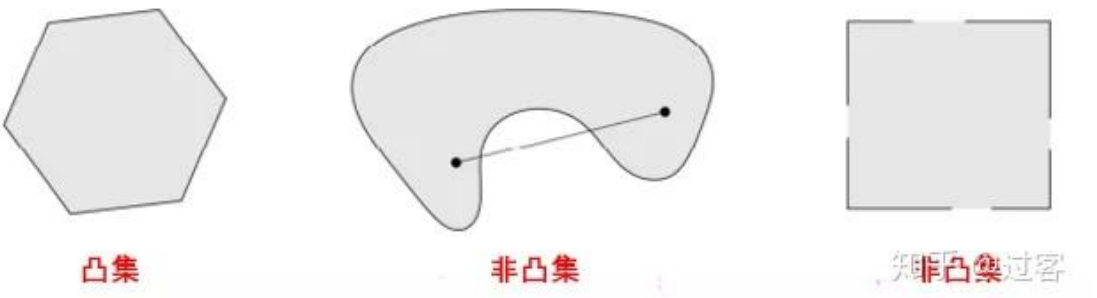
14. 什么是凸函数、凸优化?

凸集与凸函数

一个集合 $C \subseteq Rn$ 被称为凸集 ，如果对于任意 $x1, x2 \in C$ 和任意 $\lambda \in [0, 1]$，都有：

$$\forall x_1, x_2 \in C,\ \forall \lambda \in [0, 1], \quad \lambda x_1 + (1 - \lambda) x_2 \in C$$

在几何层面，就是说，凸集内任意两点的连线依然属于凸集范围



一个函数 $f：Rn \to R$ 被称为凸函数 ，如果对于其定义域中的任意两点 $x1, x2$，以及任意 $\lambda \in [0, 1]$，都有：

$$\forall x_1, x_2 \in \mathrm{dom}(f),\ x_1 \neq x_2,\ \forall \lambda_i \in (0,1), \quad f(\lambda_1 x_1 + \lambda_2 x_2) \leq \lambda_1 f(x_1) + \lambda_2 f(x_2) \quad \sum \lambda_i = 1, \lambda_i \geq 0$$

如果上述不等式在 $\lambda \in (0, 1)$ 时总是严格成立（即 $<$ 而不是 $\leq$)），则称 f 是严格凸函数 。公式如下：

$$\forall x_1, x_2 \in \mathrm{dom}(f),\ x_1 \neq x_2,\ \forall \lambda_i \in (0,1), \quad f(\lambda_1 x_1 + \lambda_2 x_2) < \lambda_1 f(x_1) + \lambda_2 f(x_2) \quad \sum \lambda_i = 1, \lambda_i \geq 0$$

凸函数在几何上，可以理解为：函数任意两点x_1和x_2之间的部分位于弦x_1x_2的下方或曲线任一点切线上方。

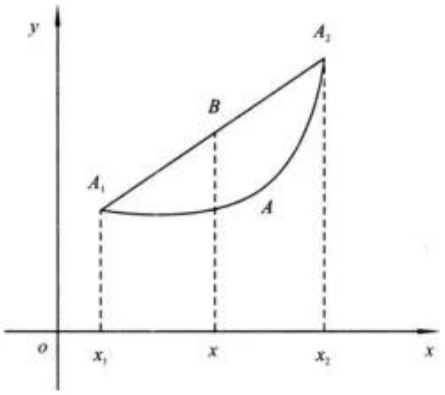


图 1 凸函数的几何性质

这里（$\lambda_1 x_1 + \lambda_2 x_2, \lambda_1 f(x_1)$ $+ \lambda_2 f(x_2)$）可以为A_1和A_2之间的线段上的任意一点，$f(\lambda_1 x_1 + \lambda_2 x_2)$可以看作图中的A点，$\lambda_1 f(x_1) + \lambda_2 f(x_2)$可以看作图中的B点

凸优化

凸优化是指将损失函数转为凸函数，从而可以保证在梯度下降过程中得到的最小值就是全局最小值，从而避免陷入局部最小值问题。
凸优化问题可以表示为：

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{subject to} \quad & g_i(x) \leq 0, \quad i = 1, \ldots, m \\ & h_j(x) = 0, \quad j = 1, \ldots, p \end{aligned}$$

其中：

- $f(x)$ 是凸函数；
- $g_i(x)$ 是凸函数；
- $h_j(x)$ 是仿射函数。

对于一个二次可微的函数 $f：R^n \to R$，其 Hessian 矩阵定义为：

$$\nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

- 其中，每个元素 $\frac{\alpha^2 f}{\alpha x \alpha y}$ 是函数 f 对变量 x_i 和 x_j 的二阶偏导数。
- Hessian 矩阵是一个对称矩阵（假设二阶偏导数连续）。
  如果 Hessian 矩阵半正定，特征值均为非负。则函数的曲率在所有方向上都是非负的，这意味着函数图像“向上弯曲”，不会出现凹陷。
  如果 Hessian 矩阵正定，则函数的曲率在所有方向上都是正的，这进一步保证了函数的严格凸性。
  如果函数 $f(x)$ 是二次可微的，则 $f(x)$ 是凸函数的充分必要条件是其 Hessian 矩阵 $\nabla^2 f(x)$ 半正定：

$$\nabla^2 f(x) \succeq 0, \quad \forall x \in \mathrm{dom}(f)$$

15. 什么是相关性、熵、KL散度、交叉熵、最大似然?给出它们的定义及公式。

1. 相关性

定义

相关性用于衡量两个随机变量之间的线性关系。最常用的相关性度量是**皮尔逊相关系数**，它描述了两个变量的协方差与它们标准差的比值。

公式

对于两个随机变量 X 和 Y，皮尔逊相关系数 $\rho(X, Y)$ 定义为：

$$\rho(X, Y) = \frac{\mathrm{Cov}(X, Y)}{\sigma_X \sigma_Y}$$

其中：

- $\mathrm{Cov}(X, Y)$ 是 X 和 Y 的协方差：

$$\mathrm{Cov}(X, Y) = \mathbb{E}[(X - \mu_X)(Y - \mu_Y)]$$

- σ_X 和 σ_Y 分别是 X 和 Y 的标准差：

$$\sigma_X = \sqrt{\mathbb{E}[(X - \mu_X)^2]}, \quad \sigma_Y = \sqrt{\mathbb{E}[(Y - \mu_Y)^2]}$$

2. 熵（Entropy）

定义

熵是信息论中的一个概念，用于衡量随机变量的不确定性或信息量。

公式

对于离散随机变量 X，其熵 $H(X)$ 定义为：

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log p(x)$$

对于连续随机变量 X，其微分熵 $h(X)$ 定义为：

$$h(X) = -\int_{-\infty}^{\infty} p(x) \log p(x) \, dx$$

其中：

- $p(x)$ 是随机变量 X 的概率分布。
- $\mathcal{X}$ 是 X 的取值集合。

3. KL散度（Kullback-Leibler Divergence）

定义

KL散度用于衡量两个概率分布 P 和 Q 之间的差异。它表示用分布 Q 近似分布 P 时的信息损失。

公式

对于离散分布 P 和 Q，KL散度定义为：

$$D_{\mathrm{KL}}(P \| Q) = \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)}$$

对于连续分布 P 和 Q，KL散度定义为：

$$D_{\mathrm{KL}}(P \| Q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} \, dx$$

其中：

- $p(x)$ 和 $q(x)$ 分别是分布 P 和 Q 的概率密度函数。

4. 交叉熵（Cross-Entropy）

定义

交叉熵用于衡量用分布 Q 编码分布 P 所需的平均信息量。

公式

对于离散分布 P 和 Q，交叉熵定义为：

$$H(P, Q) = -\sum_{x \in \mathcal{X}} p(x) \log q(x)$$

对于连续分布 P 和 Q，交叉熵定义为：

$$H(P, Q) = -\int_{-\infty}^{\infty} p(x) \log q(x) \, dx$$

其中：

- $p(x)$ 是真实分布。
- $q(x)$ 是近似分布。

5. 最大似然估计（Maximum Likelihood Estimation, MLE）

定义

最大似然估计是一种参数估计方法，通过最大化似然函数来找到使数据出现概率最大的模型参数。

公式

假设观测数据为 $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$，模型的概率分布为 $p(x|\theta)$，则似然函数为：

$$L(\theta) = \prod_{i=1}^{n} p(x_i|\theta)$$

对数似然函数为：

$$\ell(\theta) = \log L(\theta) = \sum_{i=1}^{n} \log p(x_i|\theta)$$

目标是找到使似然函数最大化的参数 θ：

$$\hat{\theta}_{\mathrm{MLE}} = \arg\max_{\theta} \ell(\theta)$$