人工智能数学基础

前言

- 本章主要介绍用于AI中所用到的基本数学知识，包括线性代数、概率论和最优化问题模块。

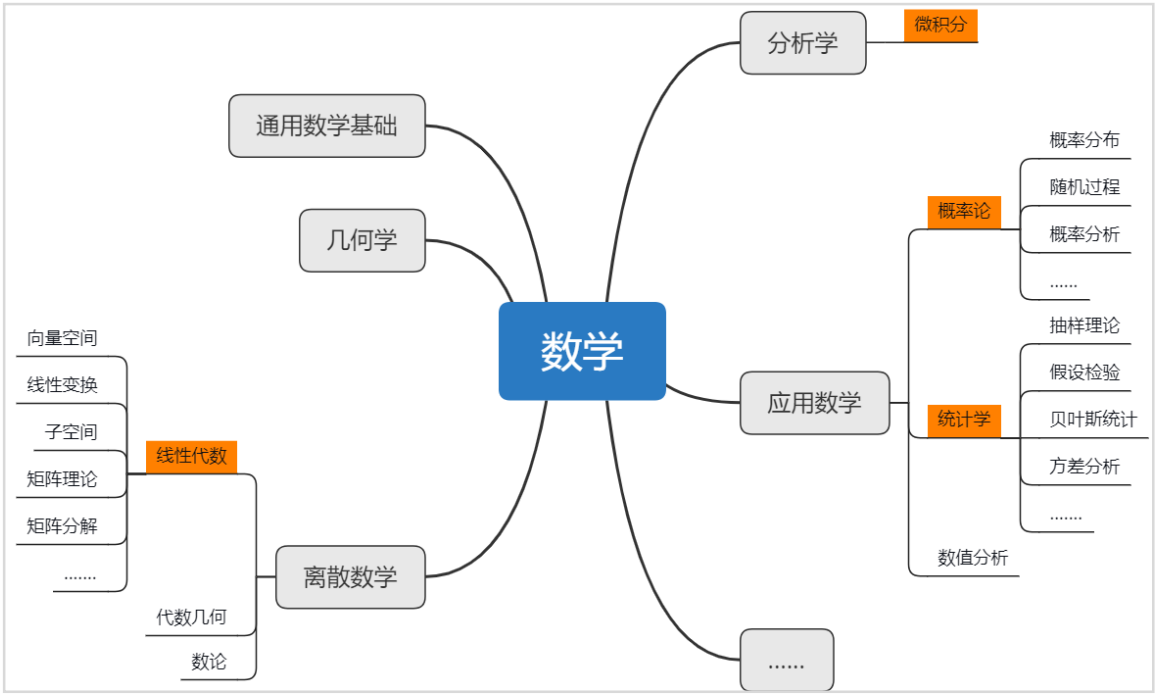
目标

- 学完本课程后，您将能够：
  - 掌握线性代数的基础知识及应用
  - 掌握概率论的基础知识及应用
  - 掌握优化问题的分类与解决方法

目录

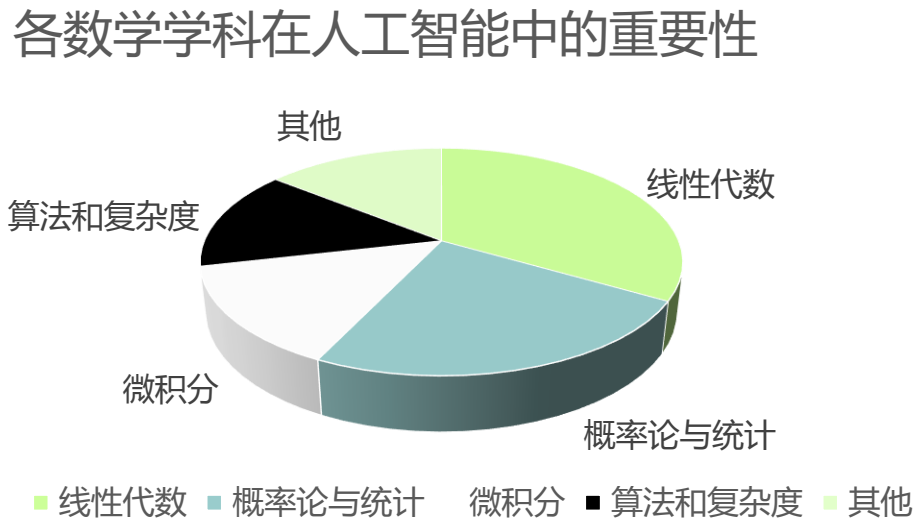
数学与人工智能 (1)

- 人工智能是一个交叉学科，应用的领域也非常广阔。不同的应用领域所要求的数学背景知识也不尽相同。但是线性代数、概率论、微积分和统计学是人工智能用于表述的“语言”。学习数学知识将有助于深入理解底层算法机制，便于开发新算法。



数学与人工智能 (2)

- 线性代数：描述深度学习算法的基础也是核心。它通过矩阵表示法来实现深度学习方法。待处理的非结构化数据转都换成离散的矩阵或向量形式。比如一张图像可以表示为按顺序排列的像素数组形式，声音数据可以表示为向量形式、神经网络就是无数的矩阵运算和非线性变换的结合。
- 概率论与统计学：研究数据分布与如何处理数据。深度学习算法所做的绝大多数事情就是预测，预测源于不确定性，而概率论与统计就是讨论不确定性的学科。
- 微积分：数学分析的基础。



目录

线性代数

- **线性代数**是代数学的一个分支，主要处理线性问题。**线性问题**是指数学对象之间的关系是以一次形式来表达的。线性代数诞生于求解**线性方程组**。**行列式**、**矩阵和向量是**处理线性问题的有力工具。
- 线性代数与人工智能：
  - 神经网络中的所有参数都被存储在矩阵中；线性代数使矩阵运算变得更加快捷简便，尤其是在GPU上训练模型时，因为GPU可以并行地以向量和矩阵运算。
  - 图像在计算中被表示为按序排列的像素数组。
  - 视频游戏使用庞大的矩阵来产生令人炫目的游戏体验。

标量、向量与矩阵

- **标量（Scalar）**：一个数，例如：$x = 3$。
- **向量（Vector）**：一个有序排列的列数，例如：

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

- **矩阵（Matrix）**：由$m \times n$个数$a_{ij}(i = 1，2，\cdots，m；j = 1，2，\cdots，n)$排成$m$行$n$列的数表，记作：

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

特殊地，行数与列数都等于n的矩阵称为n阶矩阵或n阶方阵。

向量与矩阵应用实例

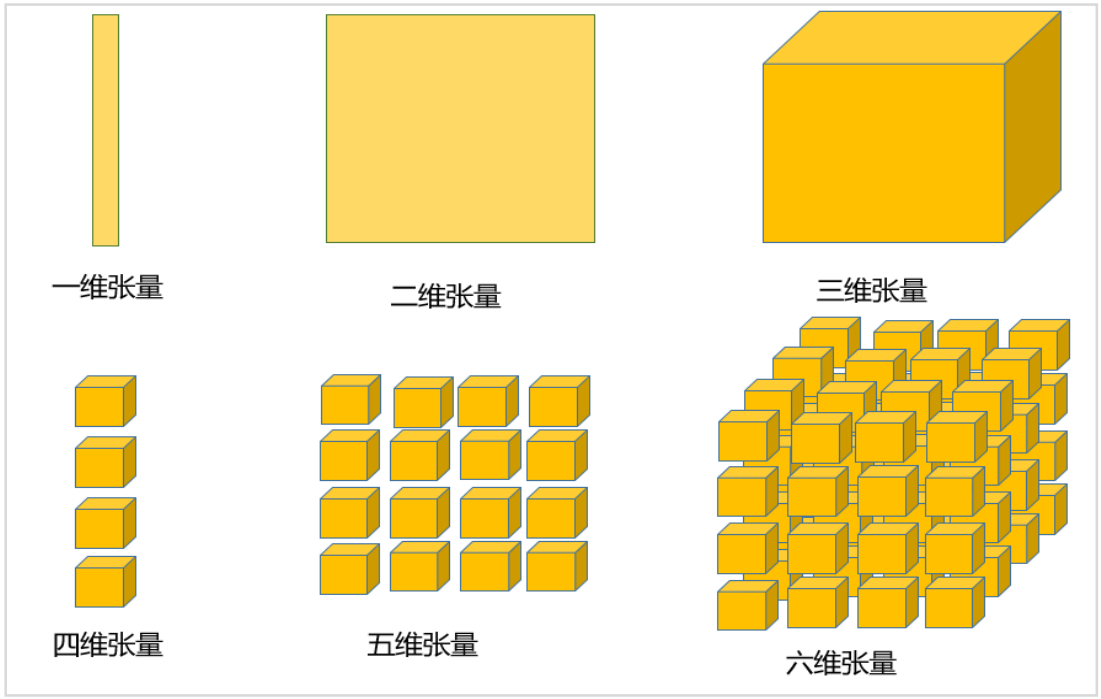
- 在计算机视觉中，一张图片会以矩阵的形式进行存储。在处理图像时，常常会把图像矩阵转换为一个向量来处理。

$$A_{m*n} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \ddots & a_{3n} \\ \vdots & \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{bmatrix}$$

$$A_{m*n} = \begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \\ \vdots \\ a_{21} \\ \vdots \\ a_{mn} \end{bmatrix}$$

张量

- **张量（Tensor）**是深度学习中的一个重要概念，是TensorFlow、Pytorch等很多深度学习框架重要组成部分。深度学习中的很多运算与模型优化过程都是基于tensor完成。
- 张量定义：一个多维数组。
  - 零阶张量：标量；一阶张量：向量；二阶张量：矩阵。



矩阵的运算

- **矩阵加法**：$C_{m\times n}=A_{m\times n}+B_{m\times n}$，即矩阵$A$，$B$对应元素相加，$c_{ij}=a_{ij}+b_{ij}$。

  注：只有矩阵A、B的行列数一样，两矩阵才可以相加。

- **标量和矩阵乘法**：设$A=(a_{ij})_{m\times n}$，$k\in K$，k与矩阵A的乘积定义为$kA=(ka_{ij})_{m\times n}$。

- **矩阵乘法**：若矩阵$A=(a_{ij})_{m\times n}$，$B=(b_{ij})_{n\times p}$，则

$$C=AB=(c_{ij})_{m\times p}$$

  注：矩阵A的列数必须和矩阵B的行数相等，AB才有意义。

$$A_{n\times m}=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix} \quad B_{m\times p}=\begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \cdots & b_{mp} \end{bmatrix}$$

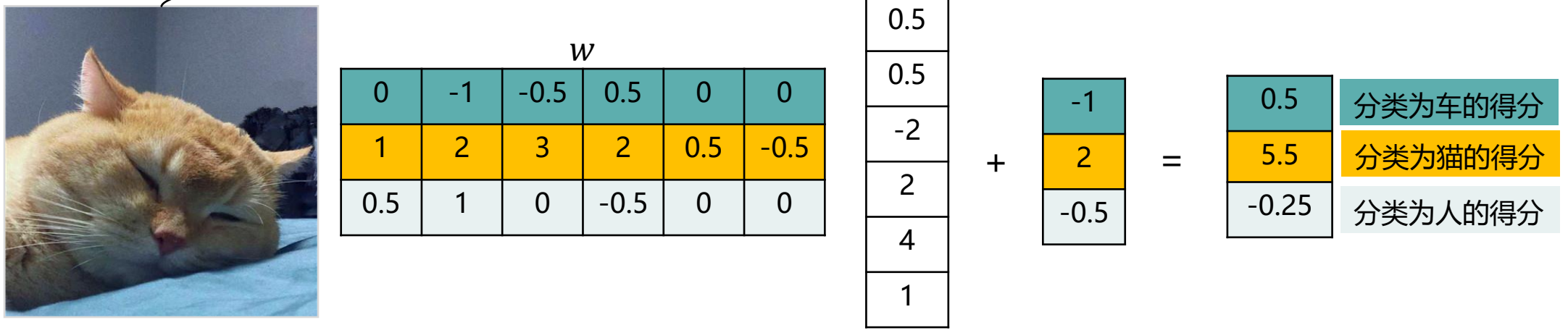
a_i 指向 c_{ij}，b_j 指向 c_{ij}

$$c_{ij}=a_ib_j=\sum_k a_{ik}b_{kj}$$

线性分类器

- 总共3类（猫、车和人），对一张图片（通过图像处理将图像处理为只由6像素点组成）进行分类，以最简单的线性分类器为例：

$$y = wx + b$$



目录

引入案例 - 矩阵与运动 (1)

- 以下矩阵对平面上的任意一点P，由$P' = A_i \mathrm{P}$ $(i = 1,2,3,\dots)$确定了什么样的变换?

(1) $A_1 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$　　(2) $A_2 = \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}$　　(3) $A_3 = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$

- 解：对于任意一点$\mathrm{P} = \begin{pmatrix} x \\ y \end{pmatrix}$，有0

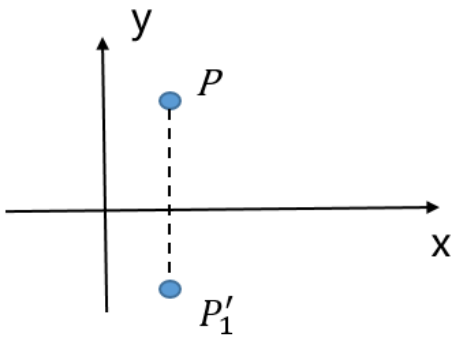
(1) $P_1{}' = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x \\ -y \end{pmatrix}$

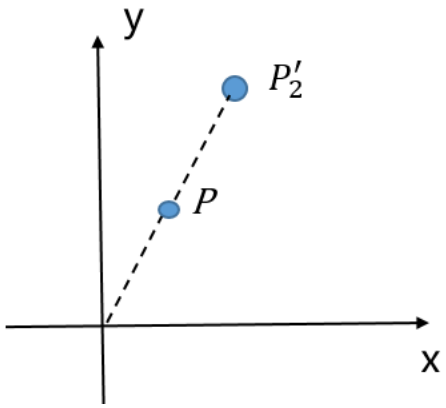
(2) $P_2{}' = \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \lambda x \\ \lambda y \end{pmatrix}$

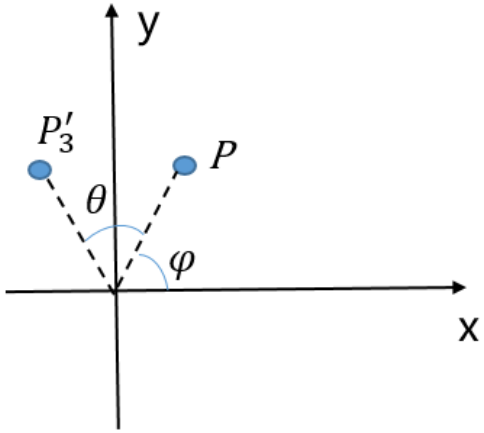
(3) $P_3{}' = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{pmatrix} r\cos(\rho) \\ r\sin(\rho) \end{pmatrix} = \begin{bmatrix} r\cos(\theta+\rho) \\ r\sin(\theta+\rho) \end{bmatrix}$

引入案例 - 矩阵与运动 (2)

- A_1确定的变换为将P变换到它关于x轴对称的点P_1'。
- A_2确定的变换为将P变换到它与原点连线上，λ>0为伸缩倍数。
- A_3确定的变换为将P绕原点旋转了角度ϑ。







- 在上述的讨论中，变换由矩阵A确定，因此称A为变换矩阵。A_1确定的变换称为反射或镜像变换，A_2确定的变换称为相似变换（λ为相似比），A_3确定的变换称为旋转变换。
- 在线性空间中，一个矩阵就对应一个线性变换，通过矩阵乘法实现。这些变换包括对于向量的旋转、缩放和映射。

线性变换

- 定义：设V_n，U_m分别是n维和m维的线性空间，T是一个从V_n 到 U_m的映射，如果映射满足：

（1）任一元素α_1，$\alpha_2 \in V_n$（从而$\alpha_1 + \alpha_2 \in V_n$），有

$$T\ (\alpha_1 + \alpha_2)\ = T\ (\alpha_1)\ + T\ (\alpha_2)$$

（2）任一元素$\alpha \in V_n$，$\lambda \in R$（从而 $\lambda\alpha \in V_n$），有

$$T(\lambda\alpha) = \lambda T(\alpha)$$

那么，T就称为从V_n到 U_m的线性映射，或称为线性变换。

- 可以将线性代数看作是讨论空间变换与向量运动的科学，而空间变换与向量运动都是由线性变换实现的。
- 线性变换的应用：线性变换在深度学习中最直观的应用为通过矩阵乘法对图像或语音数据集进行增强。如将图像沿着某个方向平移、对图像进行旋转或缩放等于产生新的图像。

目录

矩阵的转置、单位矩阵

- **转置矩阵**：把矩阵$A = (a_{ij})_{s\times n}$的行与列相互交换产生的矩阵称为A的转置，记作A'或A^T。
- 例如：

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix} \longrightarrow A^T = \begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \end{bmatrix}$$

- **单位矩阵**：所有沿主对角线的元素都是1，而其他位置的所有元素都是0的矩阵。任意矩阵与单位矩阵相乘，都不会改变。

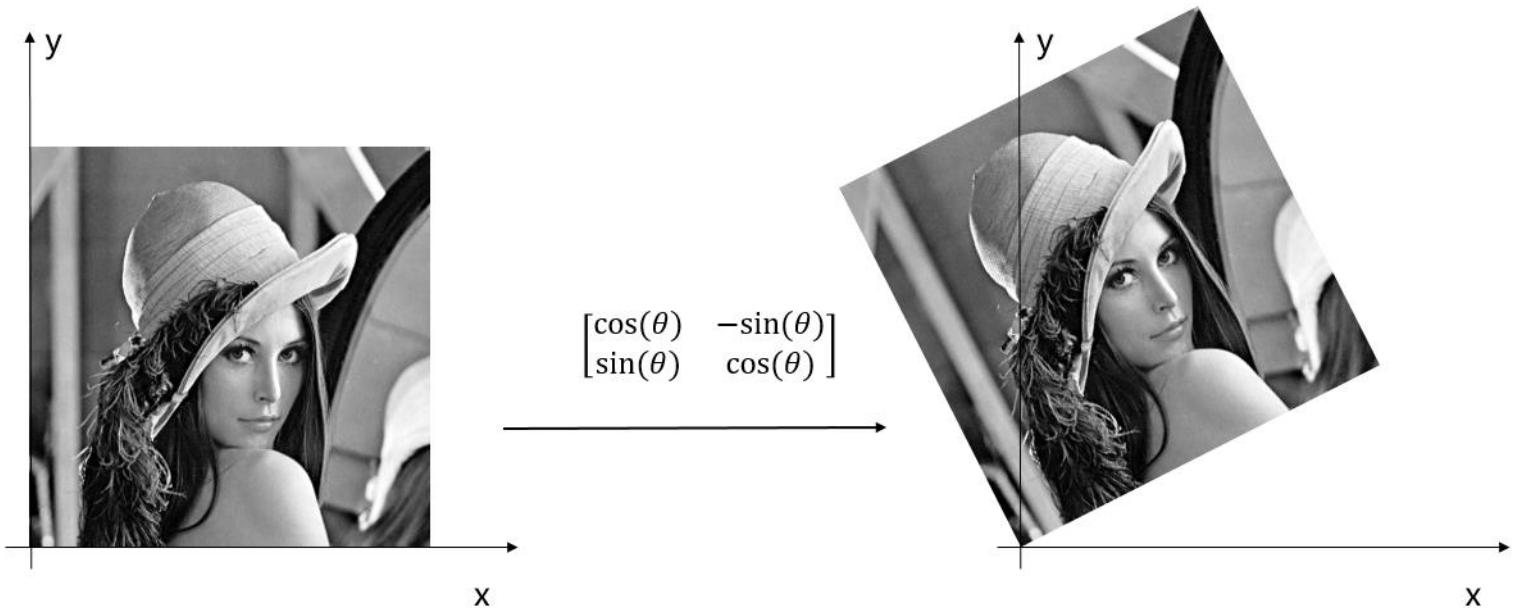
$$I_n = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix}$$

逆矩阵、正交矩阵

- **逆矩阵**：**方阵**A的逆矩阵记作A^{-1}，其满足$A^{-1}A = I_n$。
  - 逆矩阵在深度学习中的应用：牛顿法优化神经网络。
  - 在深度学习中，经常需要求逆矩阵，但是由于求逆矩阵的计算开销巨大，因此通常会将矩阵转换成其他特殊矩阵的形式以避免或简化矩阵求逆。
- **正交矩阵**：设n阶**方阵**$A = (a_{ij})_{n\times n}$，满足$AA^T = A^TA = I_n$，则称A为正交矩阵，即$A^{-1} = A^T$。
  - 正交矩阵的行向量之间与列向量之间都是两两正交（向量点积为0）的单位向量。
  - n阶正交矩阵可以看作n维空间中任意相互垂直（正交）坐标基。
  - 向量乘以一个正交矩阵：可以看作是对向量只进行旋转，而没有伸缩和空间映射作用。
- 正交矩阵的应用：
  - RNN中防止梯度消失和维度爆炸的方法：正交初始化。
  - 对于正交矩阵，可以将求逆矩阵的过程转化为求矩阵转置，大大减小了计算量。
  - 矩阵分解……

正交矩阵与图像旋转

- 之前我们提到的旋转矩阵 $A = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$ 也是一个正交矩阵。作用于图像，即与图像所对应的矩阵相乘，只图像进行旋转而保持图像的形状和大小不变。



对角矩阵

- **对角矩阵**：主对角线之外的元素皆为0的矩阵。常写为$diag(\lambda_1, \lambda_2, \cdots, \lambda_n)$，即

$$D = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_N \end{bmatrix}$$

- 对角矩阵的性质：
  - 对角矩阵的和、差、积、方幂为主对角线上元素的和、差、积、方幂。
  - 其逆矩阵为：$D^{-1} = \begin{bmatrix} {\lambda_1}^{-1} & 0 & \dots & 0 \\ 0 & {\lambda_2}^{-1} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & {\lambda_3}^{-1} \end{bmatrix}$
- 对对角矩阵进行乘法操作或逆矩阵求解（仅方阵）十分高效、计算量小。因此在机器学习中，我们会将某些矩阵限制为对角矩阵以降低计算开销。

对称矩阵

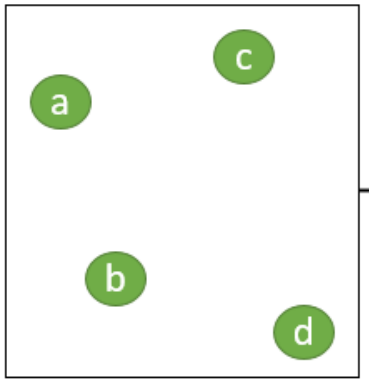
- 对称矩阵：设方阵$A=(a_{ij})_{n\times n}$，满足$A^T=A$，即$a_{ij}=a_{ji}$，则称A为对称矩阵。

$$A=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

例如：距离矩阵和协方差矩阵都是对称矩阵。

$$\Sigma=\begin{bmatrix} Var(X) & Cov(X,Y) \\ Cov(X,Y) & Var(Y) \end{bmatrix}$$

协方差矩阵



	a	b	c	d
a	0	1.2	1.1	2.8
b	1.2	0	2.2	1.3
c	1.1	2.2	0	2.5
d	2.8	1.3	2.5	0

距离矩阵

行列式

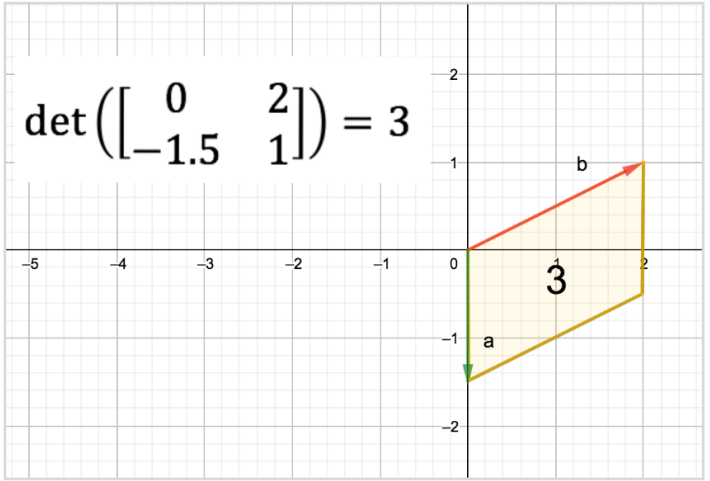
- 行列式是一个将方阵映射到一个标量的函数，记作det(A)或|A|。行列式可以看作是矩阵有向面积或体积的推广。或者说是在n维欧几里得空间中，行列式描述了一个线性变换对“体积”所造成的影响。
- 行列式的意义
  - 行列式等于矩阵特征值的乘积。
  - 行列式的绝对值可以用来衡量矩阵参与矩阵乘法后空间扩大或缩小了多少。
    - 如正交矩阵的行列式大小都为1或-1。即用正交矩阵进行线性变换后的矩阵在空间中的有向面积或体积保持不变。
  - 行列式的正负表示空间的定向。
- 行列式的应用：求矩阵特征值，求解线性方程等。

行列式

- 二阶行列式的计算：

$$\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = \mathrm{a}_{11}\mathrm{a}_{22} - \mathrm{a}_{12}\mathrm{a}_{21}$$

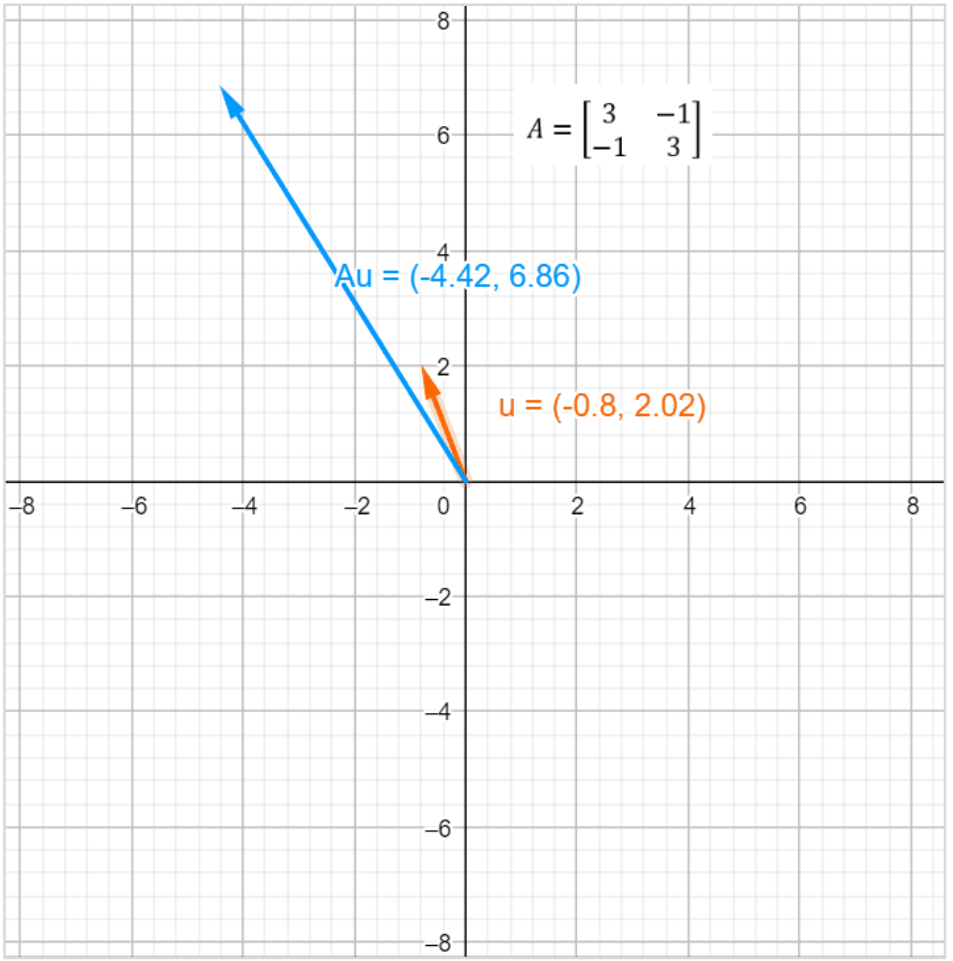
- 二阶行列式 $D = \det\left(\begin{bmatrix} 0 & 2 \\ -1.5 & 1.0 \end{bmatrix}\right)$ 的值表示二维平面上向量$a = (0, -1.5)^T$，$b = (2,1)^T$ 围成平行四边形的有向面积。



目录

特征值与特征向量

- 定义：设A是数域K上的n阶**方阵**，如果K^n中有非零列向量α使得$A\alpha = \lambda\alpha$，且$\lambda \epsilon K$，则称λ是A的一个特征值，称α是A的属于特征值λ的一个 特征向量。
- 一个矩阵对应着一种线性变换，通过矩阵乘法实现对向量的旋转、压缩和映射。如图所示，如果矩阵作用于某一个向量或某些向量使这些向量只发生伸缩变换，而不产生旋转及投影的效果，那么这些向量就称为这个矩阵的特征向量，伸缩的比例就是特征值。



特征值与特征向量的计算

- 怎样求矩阵A的特征值与特征向量：

$$A\alpha = \lambda\alpha$$

$$\Leftrightarrow A\alpha - \lambda\alpha = 0$$

$$\Leftrightarrow (A - \lambda I)\alpha = 0$$

$$\underset{\Leftrightarrow}{\alpha \neq 0} \ |A - \lambda I| = 0$$

$$\Leftrightarrow \begin{bmatrix} a_{11}-\lambda & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22}-\lambda & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn}-\lambda \end{bmatrix} = 0$$

其中，$|A - \lambda I| = 0$称为矩阵A的特征方程，λ为特征方程的解，即特征根，将特征根λ代入$A\alpha = \lambda\alpha$即可求得特征向量α。

- 特征值与特征向量的计算过程涉及到求解行列式，由此也可看出只有方阵才能求解其特征值与特征向量。

特征分解

- 设**方阵**A有n个线性无关的特征向量α_1，α_2，$\cdots$，α_n，相对应的特征值为λ_1，λ_2，$\cdots$，λ_n，则A的特征分解为：

$$A = Pdiag(\lambda)P^{-1},$$

其中$P = \{\alpha_1, \alpha_2, \cdots, \alpha_n\}$，$\lambda = \{\lambda_1, \lambda_2, \cdots, \lambda_n\}$。

- 把矩阵分解为一组特征向量和特征值，是使用最广的矩阵分解方法之一。
- 从**线性空间**的角度看，特征值越大，则矩阵在对应的特征向量上的方差越大，信息量越多。
- 在**最优化**中，矩阵特征值的大小与函数值的变化快慢有关，在最大特征值所对应的特征方向上函数值变化最大，也就是该方向上的方向导数最大。
- 应用：用于降维的PCA（Principle Component Analysis）、最优化问题、用于处理模型过拟合的正则化。

奇异值分解

- 奇异值分解：将矩阵分解为奇异向量和奇异值。可以将矩阵$A = (a_{ij})_{m\times n}$分解为三个矩阵的乘积：

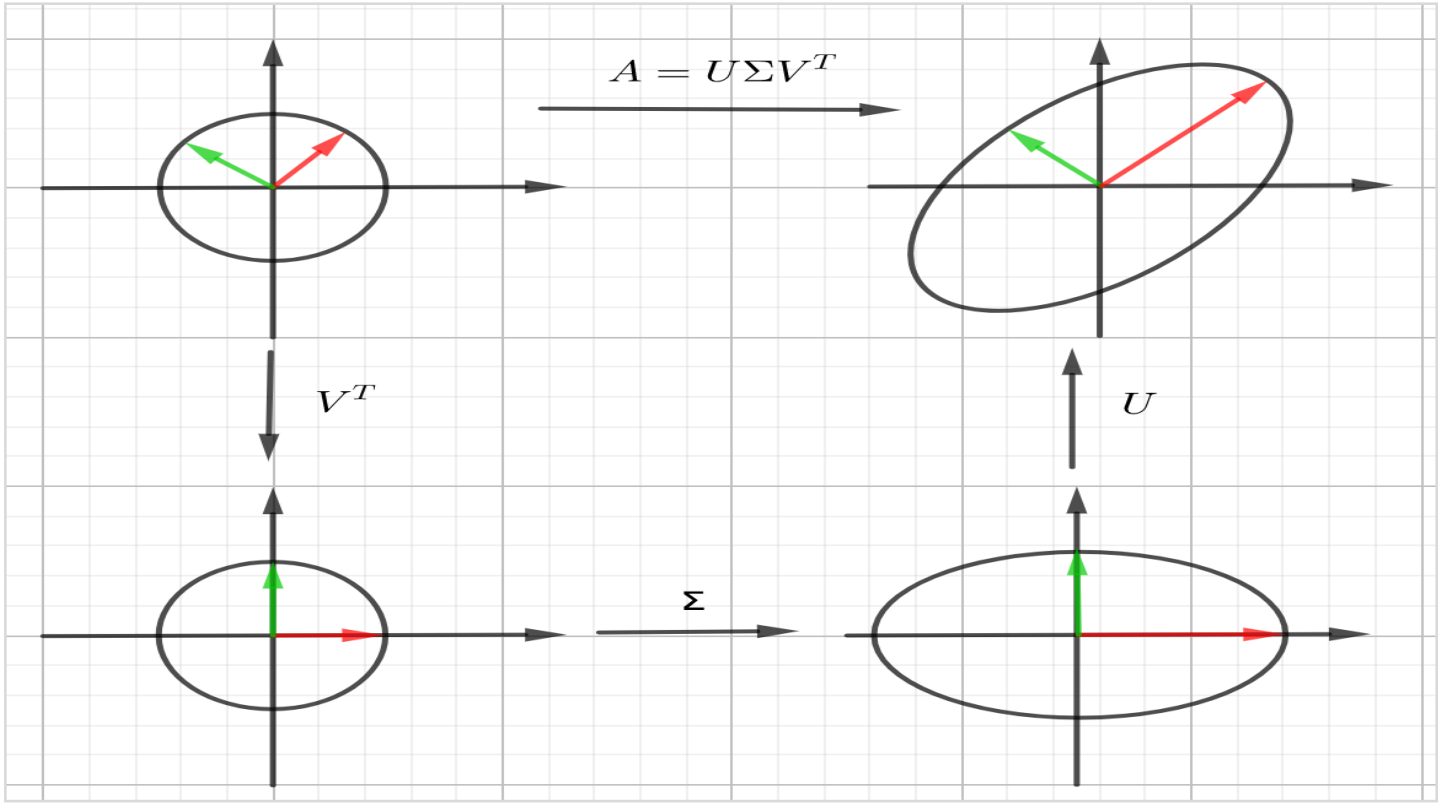
$$A = U\Sigma V^T,$$

其中$U = (b_{ij})_{m\times m}$，$\Sigma = (c_{ij})_{m\times n}$，$V^T = (d_{ij})_{n\times n}$。矩阵$U$和$V$都为正交矩阵，矩阵$U$的列向量称为左奇异向量，矩阵$V$的列向量称为右奇异向量，$\Sigma$为对角矩阵（不一定为方阵），$\Sigma$对角线上的元素称为矩阵$A$的奇异值，奇异值按从大到小的顺序排列。

- 奇异值分解的应用：PCA，数据压缩（以图像压缩为代表）算法，特征提取、数字水印和LSI（Latent semantic analysis，潜在语义分析）。

奇异值分解的几何意义

- 奇异值分解可以理解为在原空间内找到一组正交基v_i通过矩阵乘法将这组正交基映射到像空间中，其中奇异值对应伸缩系数。
- 奇异值分解将矩阵原本混合在一起的旋转、缩放和投影的三种作用效果分解出来了。



奇异值分解应用实例 – 图像压缩 (1)

- 奇异值分解：

$$A = U\Sigma V^T$$

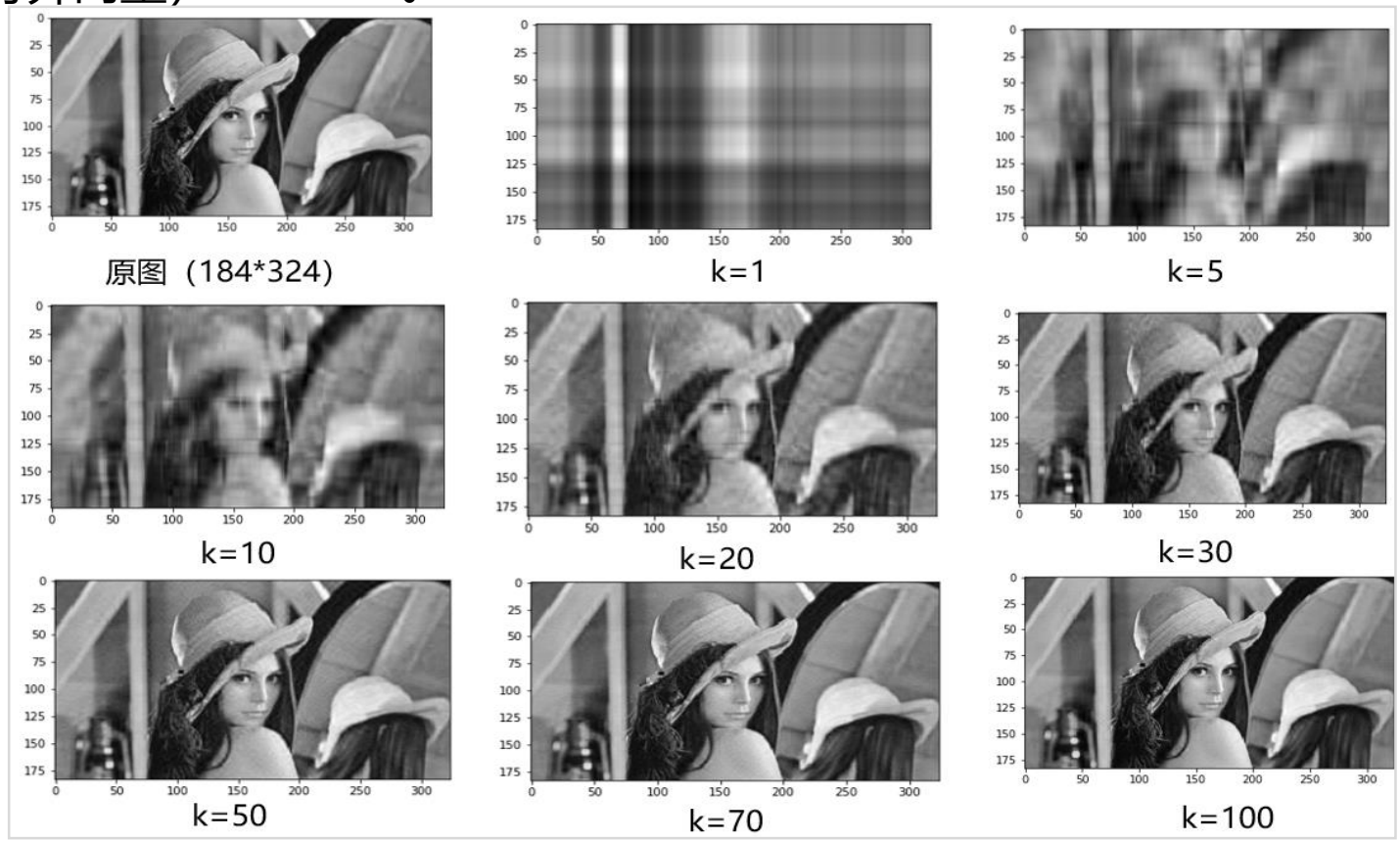
$$A=[\vec{u}_1 \quad \vec{u}_2 \quad \cdots \quad \vec{u}_m]\begin{pmatrix}\sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma n\end{pmatrix}\begin{bmatrix}\vec{v}_1^T \\ \vec{v}_2^T \\ \vdots \\ \vec{v}_n^T\end{bmatrix}$$

$$A = u_1\sigma_1\vec{v}_1^T + u_2\sigma_2\vec{v}_2^T + u_1\sigma_1\vec{v}_1^T + \cdots + u_n\sigma_n\vec{v}_n^T$$

- 将上式中的A看作一个m*n个像素组成的图像矩阵，式中奇异值矩阵中的奇异值按从大到小排列。数值越大，说明其对应的奇异向量越重要；值越小，则不是重要组成部分可以舍去。若只取前k项即能基本看清图像，则可以达到图像压缩的效果。

奇异值分解应用实例 – 图像压缩 (2)

- 原图为一张184*324的灰度图像，原图数据量为184*324=59616。用SVD实现图像压缩，取图像的前70个分量来表示图像，则图像现在数据量变为184*70（左奇异向量）+70（奇异值）+70*324（右奇异向量）= 35660。



目录

概率论与人工智能

- 概率论是研究不确定的学科。
- 概率论是现有许多人工智能算法的基础。现阶段的很多人工智能算法都是数据驱动的，且目的大多为了做预测或是作出更好的决策。如：
  - 机器翻译中，如何检测你输入的语言种类。一种简单的方法就是把你输入的词或句子进行分解，计算各语言模型的概率，然后概率最高的是最后确定的语言模型。
  - 用神经网络进行图像分类，网络的输出是衡量分类结果可信程度的概率值，即分类的置信度，我们选择置信度最高的作为图像分类结果。
  - 混合高斯模型、隐马尔科夫模型等传统语音处理模型都是以概率论为基础的。

目录

随机试验

- 满足以下三个特点的试验称为**随机试验**：
  - 可以在相同的条件下重复进行。
  - 每次试验的可能结果不止一个，并且能事先明确试验的所有可能结果。
  - 进行一次试验之前不能确定哪一个结果会出现。
- 举例：
  - E_1:抛两枚硬币，出现正面H、反面T的情况。
  - E_2:抛一枚骰子，观察可能出现的点数情况。

样本点、样本空间、随机事件

- **样本点**：一个随机试验所有可能结果的集合是**样本空间**，而随机试验中的每个可能结果称为**样本点**。
- **随机事件**：随机试验的某些样本点组成的集合，常用大写字母表示。
- 举例：
  - 随机试验E_1:扔一次骰子，观察可能出现的点数情况。
  - 扔一次硬币点数出现情况样本空间为：$S=\{1，2，3，4，5，6\}$。
  - 扔一次硬币样本点为：$e_i=1，2，3，4，5，6$。
  - 随机事件A_1：“扔一次硬币骰子出现的点数为5”，即$A_1=\{x|x=5\}$。

随机变量

- **随机变量：**本质是一个函数，是从样本空间的子集到实数的映射，将事件转换成一个数值。一些随机试验的结果可能不是数，因此很难进行描述和研究，比如$S=\{$正面，反面$\}$。因此将随机试验的每一个结果与实数对应起来，从而引入了随机变量的概念。随机变量用大写字母表示，其取值用小写字母表示。
- 举例1：随机试验E_4:抛两枚骰子，观察可能出现的点数的和。试验的样本空间是$S=\{e\}=\{(i，j)|i，j=1，2，3，4，5，6\}$，$i$，$j$分别是第1次，第2次出现的点数，以$X$记为两球号码之和，则$X$是一个随机变量：

$$X=X(e)=X(i，j)=i+j，i，j=1，2，\cdots，6。$$

- 按照随机变量的可能取值，可分为：
  - 离散随机变量：随机变量的全部可能取到的值是有限个或可列无限多个。如：某年某地的出生人数。
  - 连续随机变量：随机变量的全部可能取到的值有无限个，或数值无法一一列举。如：奶牛每天挤出奶的量，可能是一个区间中的任意值。

分布律

- 对于离散随机变量，我们通常分布律来描述其取值规律。
- **分布律，又叫概率质量函数（Probability Mass Function，PMF）**：设离散型随机变量X的所有可能取值为$x_k(k=1，2，\cdots)$，X取各个可能值的概率，即事件$\{X=x_k\}$的概率，为

$$f_X(x_k)=P\{X=x_k\}=p_k，\ k=1，2，\cdots。$$

由概率的定义，p_k满足如下两个条件：

(1) $p_k \geq 0$，$k=1$，2，$\cdots$。

(2) $\sum_{k=1}^{\infty} p_k = 1$。

- 分布律也可以用表格的形式来表示：

X	x_1	x_2	…	x_n	…
p_k	p_1	p_2	…	p_n	…

特殊离散分布 -伯努利分布

- 伯努利分布（0-1分布，两点分布，a-b分布）：设随机变量X只可能取0与1两个值，它的分布律是:

$$P\{X=k\}=p^k(1-p)^{1-k},\ k=0,\ 1\ (0<p<1),$$

则称X服从以p为参数的伯努利分布。

- 伯努利分布的分布律也可以写成:

X	0	1
p_k	$1-p$	p

其中，$E(X)=p$，$\mathrm{V}ar(X)=p(1-p)$。

- 伯努利分布主要用于二分类问题，可以用伯努利朴素贝叶斯进行文本分类或垃圾邮件分类。伯努利模型中每个特征的取值为1和0，即某个单词在文档中是否出现过，或是否为垃圾邮件。
- 为防止模型过拟合，常会用dropout方法随机丢弃神经元，每个神经元都被建模为伯努利随机变量，被抛弃的概率为p，成功输出的比例为$1-p$。

特殊离散分布 - 二项分布

- 二项分布是重复n次伯努利试验满足的分布。
- 若用X表示n重伯努利试验中事件A发生的次数，则n次试验中事件A发生k次的概率为：

$$P(X = k) = C_n^k p^k (1-p)^{n-k}，\ k = 0，\ 1，\ 2，\ \cdots n，$$

此时称X服从参数为n，p的**二项分布**，记为$X \sim B(n，\ p)$。其中$E(X) = np$，$Var(x) = np(1-p)$。

- 二项分布在NLP中使用得非常广泛，例如估计文本中含有“的”字的句子所占百分比，或者确定一个动词在语言中常被用于及物动词还是非及物动词。
- 如在Dropout方法中，对于某一层的n个神经元在每个训练步骤中可以被看作是n个伯努利实验的集合，即被丢弃的神经元总数服从参数为n，p的二项分布。

特殊离散分布 - 泊松分布

- **泊松分布：**若随机变量所有可能的取值为0，1，2，…，而取每个值的概率为：

$$P\{X=k\}=\frac{\lambda^k e^{-\lambda}}{k!}，k=0，1，2，\cdots，$$

则称X服从参数为λ的泊松分布，记为：$X\sim P(\lambda)$.其中，$E(X)=\lambda$，$D(X)=\lambda$。参数λ是单位时间或单位面积内随机事件的平均发生率。

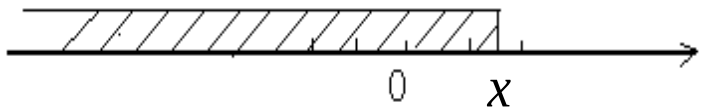
- 泊松分布是二项分布当n很大p很小时的近似计算。
- 泊松分布用于描述单位时间内随机事件发生的次数。如一段时间内某一客服电话受到的服务请求的次数、汽车站台的候客人数、机器出现的故障数、自然灾害发生的次数、DNA序列的变异数等。
- 图像处理中，图像会因为观点显示仪器测量造成的不确定性而出现服从泊松分布的泊松噪声，我们经常会给图像加泊松噪声用于图像的数据增强。

分布函数

- 实际生活中，我们通常不太关心取到某一点的概率，而是取到某一区间的概率。所以我们需要研究分布函数。
- **分布函数，又叫累计分布函数（Cumulative Distribution Function，CDF）**：设X是一个随机变量，x是任意实数，函数$F(x)$称为X的分布函数，

$$F(x) = P\{X \leq x\}, \quad -\infty < x < \infty。$$

- 分布函数$F(x)$的意义：如果将X看成是数轴上的随机点的坐标，那么分布函数$F(x)$在x处的函数值就表示X落在区间$(-\infty, x]$上的概率，即随机变量X小于等于x的概率。

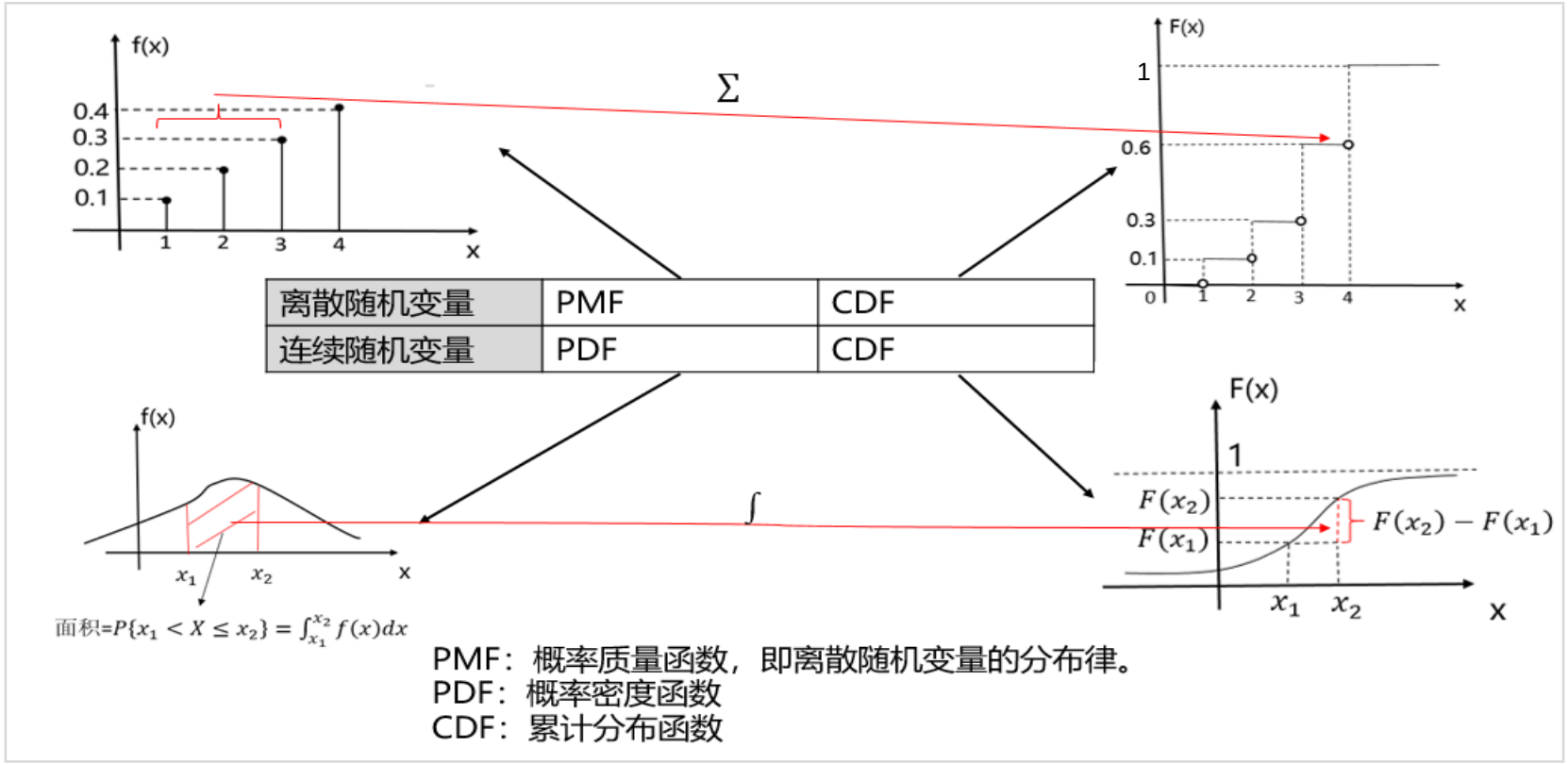


连续型随机变量与概率密度函数

- 如果对于连续随机变量X的分布函数$F(x)$，存在非负函数$f(x)$，使对于任意实数x有

$$F(x)=\int_{-\infty}^{x} f(t)dt,$$

则称函数$f(x)$称为X的**概率密度函数（Probability Density Function，PDF）**，简称概率密度。



PMF：概率质量函数，即离散随机变量的分布律。
PDF：概率密度函数
CDF：累计分布函数

特殊分布 - 正态分布

- 若连续型随机变量X的概率密度函数为

$$f(x)=\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(x-\mu)^2}{2\sigma^2}},\quad -\infty<x<\infty,$$

其中μ，$\sigma(\sigma>0)$为常数，则称X服从参数为μ，σ的正态分布或高斯分布，记为$X\sim N(\mu,\sigma^2)$。当$\mu=0$，$\sigma=1$时称随机变量X服从标准正态分布，记为$X\sim N(0,1)$。

- 在自然现象和社会现象中，大量随机变量都服从或近似服从正态分布。高斯分布是机器学习中最常用的分布，如：
  - 图像处理中，我们可以给图像添加高斯噪声用于图像增强等任务。也可以用高斯滤波器去除噪声并平滑图像。还可以用混合高斯模型进行图像的前景目标检测。
  - 在传统语音识别模型GMM-HMM（高斯混合模型-隐马尔科夫）中，高斯混合模型就是由多个高斯分布混合起来的模型。

图像的泊松噪声与高斯噪声

原图

加入$\mu = 0$，$\sigma = 10$的高斯噪声

加入$\lambda = 15$的泊松噪声

目录

随机向量

- 在实际应用中，经常需要对所考虑的问题用多个变量来描述。我们把多个随机变量放在一起组成向量，称为多维随机变量或者随机向量。
- **定义**：如果$X_1(\omega)$，$X_2(\omega)$，…，$X_n(\omega)$是定义在同一个样本空间$\Omega = \{\omega\}$上的n个随机变量，则称

$$X\ (\omega)\ =\ (X_1(\omega),\ X_2(\omega),\ \ldots,\ X_n(\omega))\ ,$$

为**n维（或n元）随机变量或随机向量。**

- 如我们通过人脸判断人的年龄，可能需要结合多个特征（随机变量），如脸形、脸部纹理、面部斑点、皮肤松弛度、发际线等，将这些特征结合映射为一个实数，即年龄。

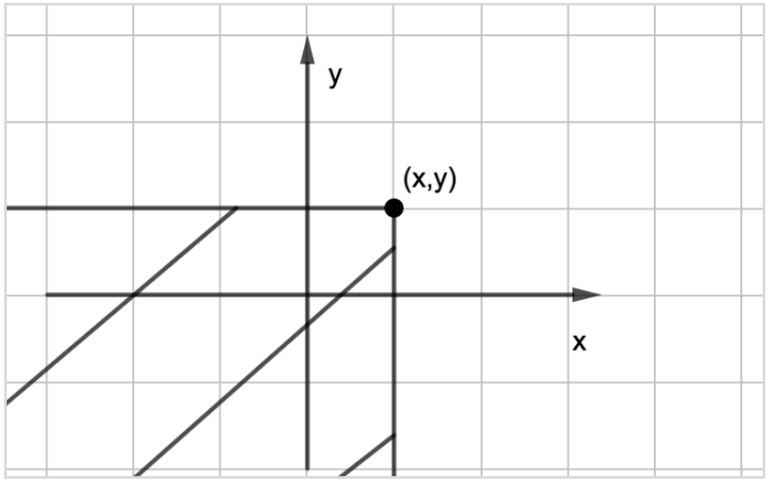
联合分布函数

- 对应随机变量的分布函数，随机向量有对应的联合分布函数。
- 定义：对任意的n个实数$x_1, x_2, \ldots, x_n$，则**n个事件$\{X_1 \le x_1\}$,$\{X_2 \le x_2\}$,...,$\{X_n \le x_n\}$同时发生的概率**为

$$F\left(x_1, x_2, \ldots, x_n\right) = P\left(X_1 \le x_1, X_2 \le x_2, \ldots, X_n \le x_n\right)$$

称为n维随机变量的联合分布函数。

- 二维联合分布函数：$F(x,y) = P(X \le x, Y \le y)$，表示随机点（X,Y）落在以$(x,y)$为顶点的左下方无穷矩形区域的概率。



联合概率密度

- 对应一维随机变量的概率密度函数，随机向量有对应的联合概率密度。
- 定义：如果存在二元非负函数$p(x,y)$，使得二维随机变量(X,Y)的分布函数可表示为

$$F(x,y)=\int_{-\infty}^{x}\int_{-\infty}^{y}p(u,v)dudv,$$

则称（X,Y）为二维连续随机变量，称$p(u,v)$为(X,Y)的联合概率密度。

目录

条件概率、贝叶斯公式 (1)

- 已知原因求解事件发生的概率通常被叫做**条件概率也叫后验概率**：

$$P(Y|X) = \frac{P(YX)}{P(X)}$$

- 我们经常需要在已知事件发生的情况下计算$P(X|Y)$，即事件已经发生了，再分析原因。此时若还知道先验概率$P(X)$，我们就可以用**贝叶斯公式**来计算：

$$P(X|Y) = \frac{P(XY)}{P(Y)} = \frac{P(Y|X)P(X)}{P(Y)}$$

条件概率、贝叶斯公式 (2)

- 假设X是由相互独立的事件组成的概率空间$\{X_1, X_2, \ldots, X_n\}$，则$P(Y)$可以用**全概率公式**展开：$P(Y) = P(Y|X_1)P(X_1) + P(Y|X_2)P(X_2) + \cdots + P(Y|X_n)P(X_n)$，此时**贝叶斯公式**可表示为：

$$P(X_i|Y) = \frac{P(Y|X_i)P(X_i)}{\sum_{i=1}^{n} P(Y|X_i)P(X_i)}$$

- 贝叶斯公式应用：中文分词、统计机器翻译、深度贝叶斯网络等。

贝叶斯定理应用 – 中文分词

- 如何对这个句子进行分词（词串）才最靠谱？
  - 杭州|西湖、杭|州西湖、杭州西|湖
- 令 Y 为字串（句子），X 为词串（一种特定的分词假设）。我们就是需要寻找使得 P(X|Y) 最大的 X，使用贝叶斯公式有得：

$$P(\mathrm{X}|Y) = \frac{P(XY)}{P(Y)} = \frac{P(Y|X)P(X)}{P(Y)}$$

- 若已知P(Y)：对于每种分词假设都不变、P(X)：这种分词方式（词串）的可能性和P(Y|X)：这个词串生成我们的句子的可能性，我们就可以成功分词。

期望、方差

- **数学期望（或均值，亦简称期望）**:是试验中每次可能结果的概率乘以其结果的总和，是概率分布最基本的数学特征之一。它反映随机变量平均取值的大小。
  - 对于离散型随机变量：$E(X)=\sum_{k=1}^{\infty}x_k\,p_k$，$k=1$，$2$，$\cdots$。
  - 对于连续型随机变量：$E(X)=\int_{-\infty}^{\infty}xf(x)dx$。
- **方差：**是衡量随机变量或一组数据离散程度的度量，即随机变量和其数学期望之间的偏离程度。

$$D(X)=Var(X)=E\{[X-E(X)]^2\}$$

另外，$\sqrt{D(X)}$，记为$\sigma(X)$，称为标准差或均方差。$X^*=\frac{X-E(X)}{\sigma(X)}$称为$X$的标准化变量。

协方差、相关系数、协方差矩阵

- **协方差**：在某种意义上给出了两个随机变量线性相关性的强度。

$$Cov(X，Y) = E\left[\left(X - E(X)\right)\left(Y - E(Y)\right)\right]$$

- **相关系数**又叫线性相关系数，用来度量两个变量间的线性关系。

$$\rho_{XY} = \frac{Cov(X，Y)}{\sqrt{D(X)}\sqrt{D(Y)}}$$

- 随机变量$(X_1，X_2)$的**协方差矩阵**：

$$C = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix}$$

其中，$c_{ij} = Cov\left(X_i，X_j\right) = E\left\{\left[X_i - E(X_i)\right]\left[X_j - E\left(X_j\right)\right]\right\}$，$i，j = 1，2，\cdots，n$。协方差矩阵对角线上的元素分别是$X_1$，$X_2$的方差，其余元素为$X_1$，$X_2$的协方差。

目录

最优化问题

- **最优化问题：**指的是改变x以最小化或最大化某个函数$f(x)$的任务。可以表示为：

$$min(max)\ f(x) \qquad \text{目标函数的极小（极大）}$$

$$s.t.\ \ g_i(x) \geq 0, i = 1,\ 2,\ \cdots,\ m, \qquad \text{不等式约束}$$

$$h_j(x) = 0,\ j = 1,\ 2,\ \cdots,\ p, \qquad \text{等式约束}$$

其中$x = (x_1,\ x_2,\ \cdots,\ x_n)^T \in R^n$，我们将$f(x)$称为目标函数或准则；当对其进行最小化时，也把它称为**代价函数**、**损失函数**或**误差函数**。

- 如果除目标函数以外，对参与优化的各变量没有其他约束，则称为无约束最优化问题。反之，称为有约束最优化问题。

最优化问题的分类

- **无约束最优化**可以写为

$$minf(x)$$

- **约束优化**：是优化问题的分支。实际生活中的优化问题大多都是带约束条件的，我们可能希望在x的某些集合s中找$f(x)$的最大值或最小值。集合s内的点称为**可行点**。
- **等式约束最优化**可以写为

$$minf(x)$$
$$s.t.\ \ g_i(x)=0,\ \ i=1,\ \ 2,\ \ \cdots,\ \ n$$

- **不等式约束最优化**可以写为

$$minf(x)$$
$$s.t.\ \ g_i(x)=0,\ \ i=1,\ \ 2,\ \ \cdots,\ \ n,$$
$$h(x)\leq 0,\ \ j=1,\ \ 2,\ \ \cdots,\ \ m$$

最优化问题求解

- **无约束优化求解**：主要有解析法和直接法。
  - 直接法通常用于当目标函数表达式十分复杂或写不出具体表达式时的情况。通过数值计算，经过一系列迭代过程产生点列，在其中搜索最优点。
  - 解析法，即间接法，是根据无约束最优化问题的目标函数的解析表达式给出一种求最优解的方法，主要有梯度下降法、牛顿法、拟牛顿法、共轭方向法和共轭梯度法等。
- **约束最优化求解**：解决约束最优化问题最常用的方法是引用拉格朗日乘子（等式约束）或者KKT（Kuhn-Kuhn-Tucker）条件（不等式约束）将含有n个变量和k个约束条件的约束优化问题转化为含有（n+k）个变量的无约束优化问题进行求解。
- 在此我们重点讨论在深度学习最常用的无约束优化求解方法，即梯度下降算法。

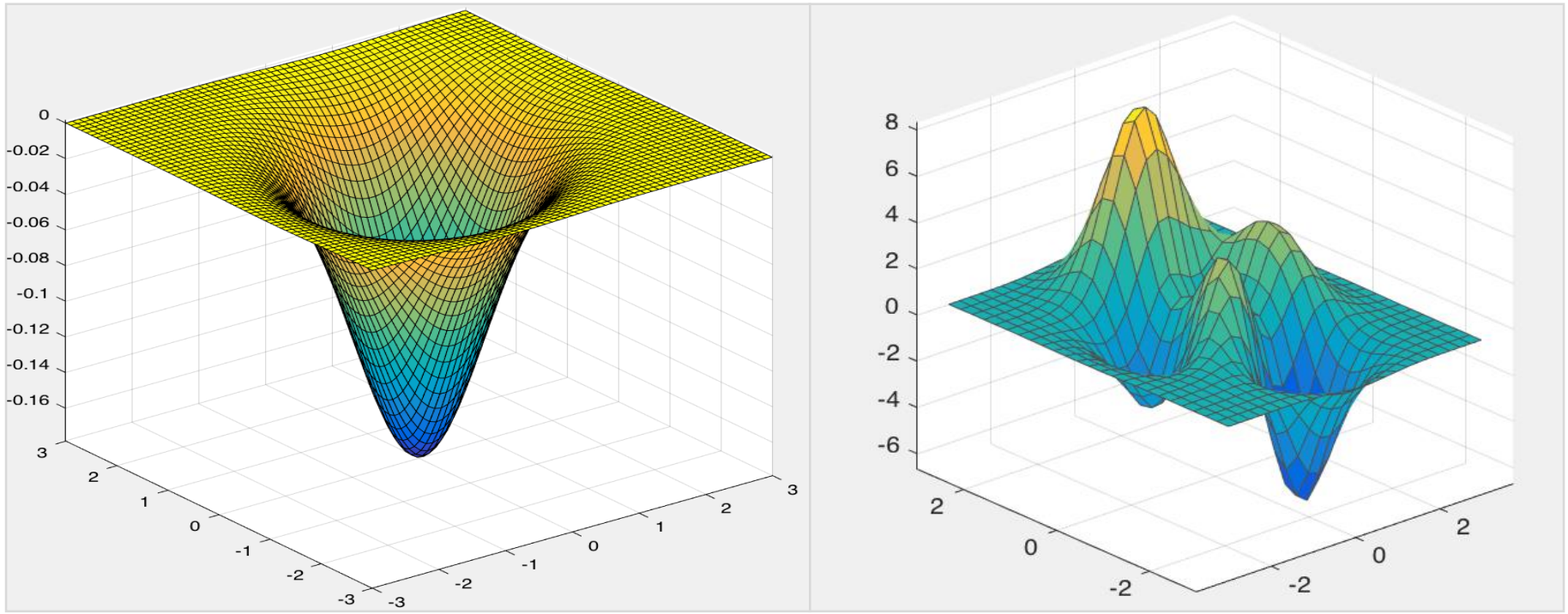
目录

梯度下降法 (1)

- 凸函数：对于$\lambda \in (0, 1)$，任意x_1，$x_2 \in R$，都有

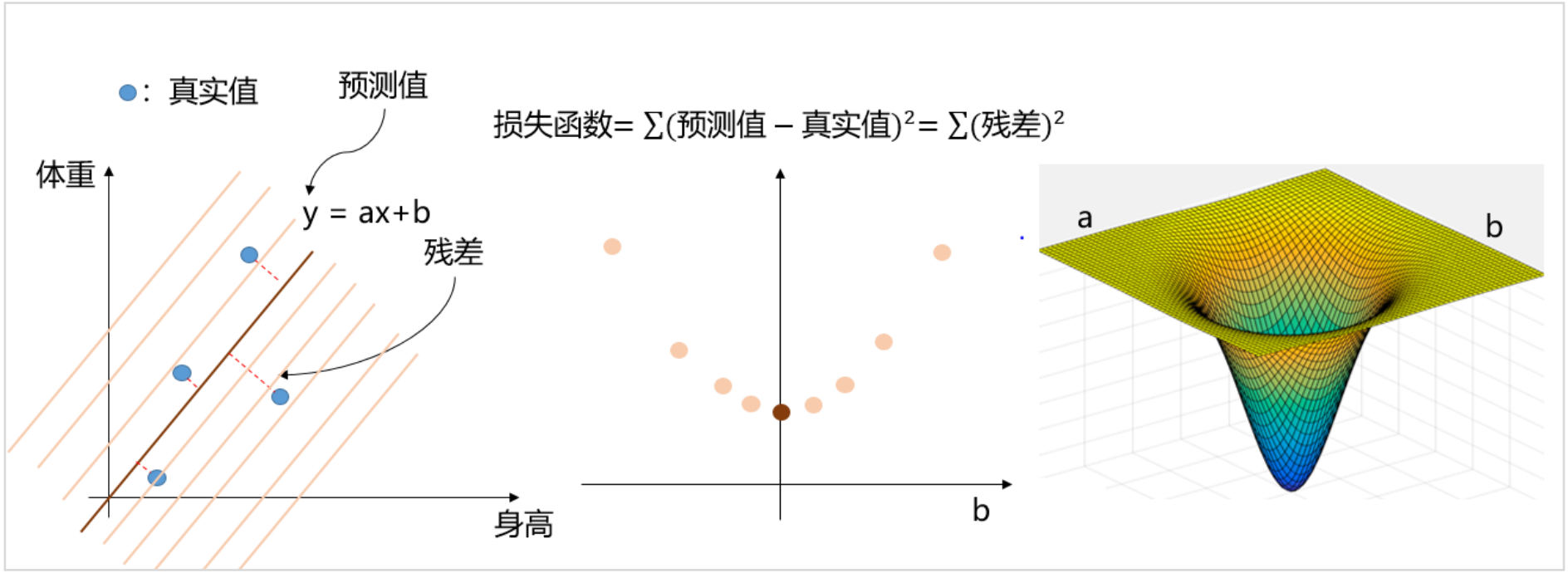
$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2),$$

则称$f(x)$是一个凸函数。凸函数的极值点出现在驻点处。



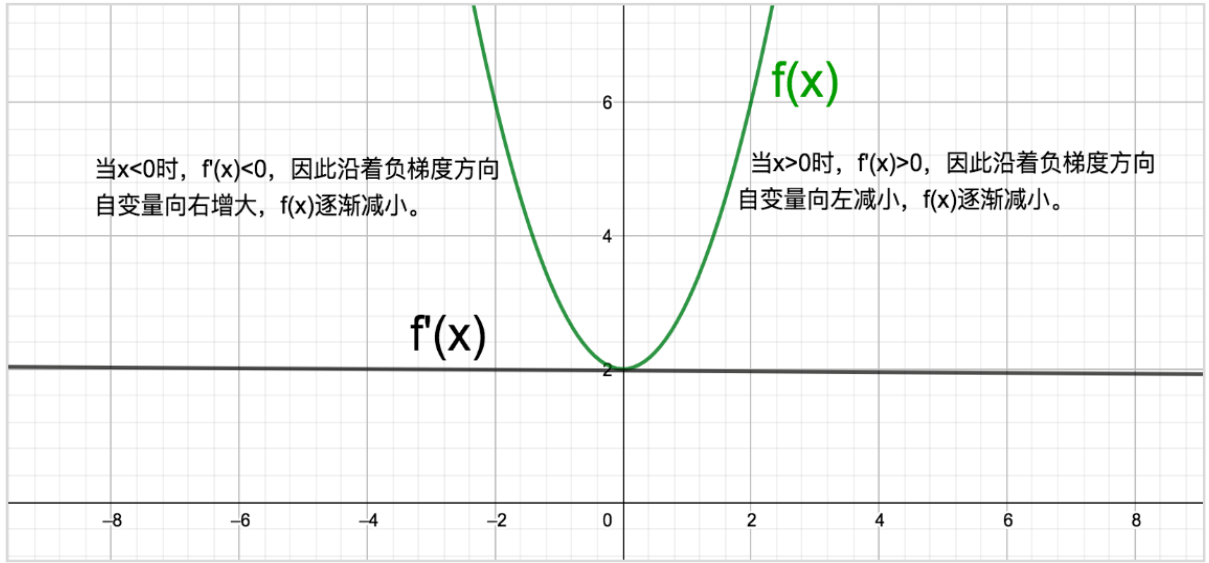
梯度下降法 (2)

- 优化的过程是逐渐调整模型或函数参数使其输出的预测值与真实值越来越接近的过程。
- 损失函数通常是用来描绘模型的预测值与真实值的一致程度。
- 优化的目标是寻找损失函数最小（大）值的过程。
- 若只优化一个变量，损失函数是一元的。若优化的变量有两个以上，则损失函数是二元及以上的。



梯度下降法 (2)

- 一元函数的极值问题：
  - 函数极值存在于$f'(x) = 0$的点处。
  - $f'(x) = 0$的点称为临界点或驻点。
  - 函数的极值点一定是驻点，反之未必。
- 推广至多维函数的情形，用偏导数描述函数相对于各自变量的变化程度。



梯度下降法 (3)

- 如何从初始位置到函数极值点?
  - 每一步都朝着函数值下降最快的方向移动一段距离。
- 函数下降（上升）最快的方向：函数值变化率最大的方向。
  - **导数**：函数沿坐标轴正方向的变化率。
  - **方向导数**：函数沿着任意方向的变化率。（多个）
  - **最大方向导数：梯度**。梯度方向即函数值变化率最大的方向，也就是我们需要沿着梯度方向寻找函数极值。
- 一段距离：步长，由经验获取的，可以在尝试过程中不断调整。

梯度下降法 (4)

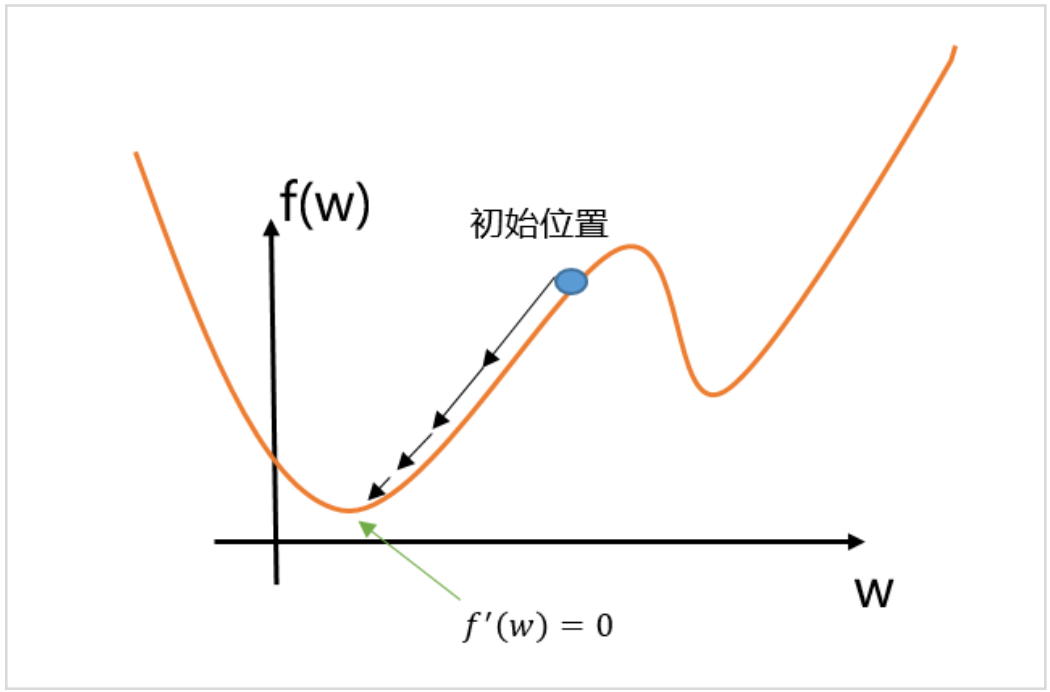
- 正梯度向量指向上坡，负梯度向量指向下坡。我们在负梯度方向上移动可以最快地减小函数值，这被称为**最速下降法**（method of steepest descent）或**梯度下降**（gradient descent）。
- 在梯度下降法中，每一步更新的方式为：

$$x' = x - \varepsilon \nabla_x f(x),$$

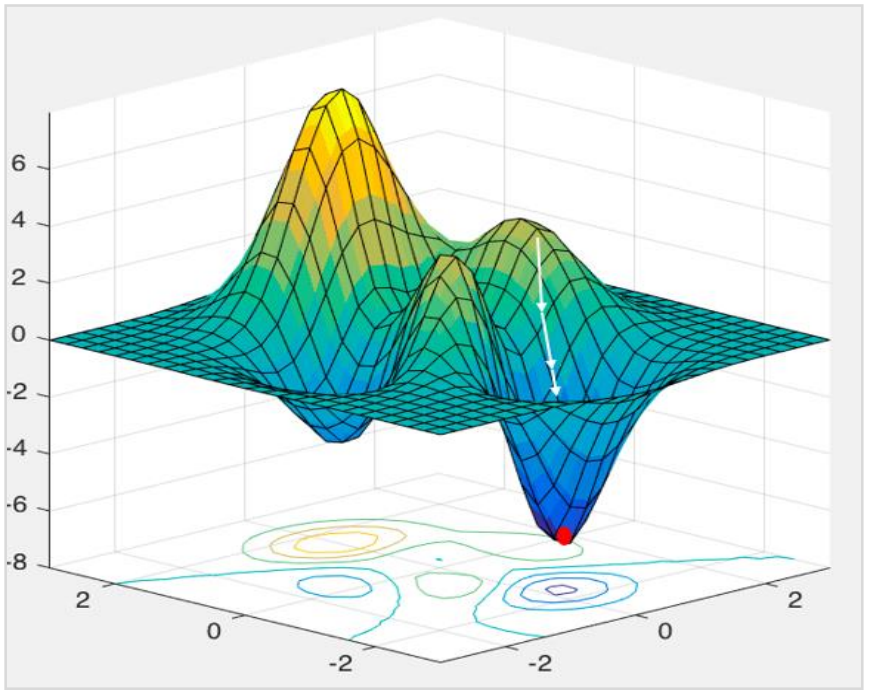
其中ε为学习率（learning rate），是一个确定步长的正标量。

- 迭代在梯度为零或趋近于零的时候收敛。

梯度下降法 (5)



二维空间



三维空间

本章总结

- 本章主要介绍了AI的数学基础知识，其中包括线性代数、概率论与最优化问题，为后续学习奠定基础。

更多信息

- 华为Learning网站
  - http://support.huawei.com/learning/Index!toTrainIndex
- 华为Support案例库
  - http://support.huawei.com/enterprise/servicecenter?lang=zh


谢 谢
www.huawei.com