人工智能

跟开发 APP 和后台服务器相比，人工智能需要大量的数学知识

需要哪些数学知识？？？

微积分

线性代数

概率论

最优化

关于书籍，特别说明一下，除非你是数学知识遗忘的特别厉害了，或者是本科的时候没有学过相关数学知识，否则不建议大家抱着书去学习，会浪费大家大量的精力和时间

微积分

导数与求导公式

一阶导数与函数的单调性

一元函数极值判定法则

高阶导数

二阶导数与函数的凹凸性

一元导数泰勒展开

先说微积分/高等数学。在机器学习中，微积分主要用到了微分部分，作用是求函数

的极值，就是很多机器学习库中的求解器（solver）所实现的功能。在机器学习里会用到微积分中的以下知识点：

- 导数和偏导数的定义与计算方法
- 梯度向量的定义
- 极值定理，可导函数在极值点处导数或梯度必须为 0
- 雅克比矩阵，这是向量到向量映射函数的偏导数构成的矩阵，在求导推导中会用到
- Hessian 矩阵，这是 2 阶导数对多元函数的推广，与函数的极值有密切的联系
- 凸函数的定义与判断方法
- 泰勒展开公式
- 拉格朗日乘数法，用于求解带等式约束的极值问题

其中最核心的是记住多元函数的泰勒展开公式，根据它我们可以推导出机器学习中常用的梯度下降法，牛顿法，拟牛顿法等一系列最优化方法，泰勒公式：

微积分和线性代数，微积分中会用到大量线性代数的知识，线性代数中也会用到微积分的知识

线性代数

向量及其运算
矩阵及其运算
张量
行列式
二次型
特征值与特征向量

相比之下，线性代数用的更多。在机器学习的几乎所有地方都有使用，具体用到的知识点有：

- 向量和它的各种运算，包括加法，减法，数乘，转置，内积
- 向量和矩阵的范数，L1 范数和 L2 范数
- 矩阵和它的各种运算，包括加法，减法，乘法，数乘
- 逆矩阵的定义与性质
- 行列式的定义与计算方法
- 二次型的定义
- 矩阵的正定性
- 矩阵的特征值与特征向量
- 矩阵的奇异值分解
- 线性方程组的数值解法，尤其是共轭梯度法

机器学习算法处理的数据一般都是向量、矩阵或者张量。经典的机器学习算法输入的数据都是特征向量，深度学习算法在处理图像时输入的 2 维的矩阵或者 3 维的张量。掌握这些知识会使你游刃有余。

多元函数微分学
高阶偏导数
雅克比矩阵
Hessian 矩阵
多元函数泰勒展开
多元函数极值判定法则

回到线性代数
奇异值分解 SVD
常用的矩阵和向量求导公式

概率论

随机事件与概率
条件概率和贝叶斯公式
随机变量
随机变量的期望和方差
常用概率分布（正太分布、均匀分布、伯努利二项分布）
随机向量（联合概率密度函数等）
协方差与协方差矩阵
最大似然估计

如果把机器学习所处理的样本数据看作随机变量/向量，我们就可以用概率论的观点对问题进行建模，这代表了机器学习中很大一类方法。在机器学习里用到的概率论知识点有:

- 随机事件的概念，概率的定义与计算方法
- 随机变量与概率分布，尤其是连续型随机变量的概率密度函数和分布函数
- 条件概率与贝叶斯公式
- 常用的概率分布，包括正态分布，伯努利二项分布，均匀分布
- 随机变量的均值与方差，协方差

- 随机变量的独立性
- 最大似然估计

最优化

最后要说的是最优化，因为几乎所有机器学习算法归根到底都是在求解最优化问题。求解最优化问题的指导思想是在极值点出函数的导数/梯度必须为 0。因此你必须理解梯度下降法，牛顿法这两种常用的算法，它们的迭代公式都可以从泰勒展开公式中得到。如果能知道坐标下降法、拟牛顿法就更好了。

凸优化是机器学习中经常会提及的一个概念，这是一类特殊的优化问题，它的优化变量的可行域是凸集，目标函数是凸函数。凸优化最好的性质是它的所有局部最优解就是全局最优解，因此求解时不会陷入局部最优解。如果一个问题被证明为是凸优化问题，基本上已经宣告此问题得到了解决。在机器学习中，线性回归、岭回归、支持向量机、logistic 回归等很多算法求解的都是凸优化问题。

拉格朗日对偶为带等式和不等式约束条件的优化问题构造拉格朗日函数，将其变为原问题，这两个问题是等价的。通过这一步变换，将带约束条件的问题转换成不带约束条件的问题。通过变换原始优化变量和拉格朗日乘子的优化次序，进一步将原问题转换为对偶问题，如果满足某种条件，原问题和对偶问题是等价的。这种方法的意义在于可以将一个不易于求解的问题转换成更容易求解的问题。在支持向量机

中有拉格朗日对偶的应用。

KKT 条件是拉格朗日乘数法对带不等式约束问题的推广，它给出了带等式和不等式约束的优化问题在极值点处所必须满足的条件。在支持向量机中也有它的应用。

如果你没有学过最优化方法这门课也不用担心，这些方法根据微积分和线性代数的基础知识可以很容易推导出来。如果需要系统的学习这方面的知识，可以阅读《凸优化》，《非线性规划》两本经典教材。

算法或理论	**用到的数学知识点**
贝叶斯分类器	随机变量，贝叶斯公式，随机变量独立性，正态分布，最大似然估计
决策树	概率，熵，Gini 系数
KNN 算法	距离函数

主成分分析	协方差矩阵，散布矩阵，拉格朗日乘数法，特征值与特征向量
流形学习	流形，最优化，测地线，测地距离，图，特征值与特征向量
线性判别分析	散度矩阵，逆矩阵，拉格朗日乘数法，特征值与特征向量
支持向量机	点到平面的距离，Slater 条件，强对偶，拉格朗日对偶，KKT 条件，凸优化，核函数，Mercer 条件
Logistic	概率，随机变量，最大似然估计，梯度下降法，凸优化，牛顿法
随机森林	抽样，方差
AdaBoost 算法	概率，随机变量，最大似然估计，梯度下降法，凸优化，牛顿法
隐马尔可夫模型	概率，离散型随机变量，条件概率，随机变量独立性，拉格朗日乘数法，最大似然估计
条件随机场	条件概率，数学期望，最大似然估计
高斯混合模型	正态分布，最大似然估计，Jensen 不等式
人工神经网络	梯度下降法，链式法则
卷积神经网络	梯度下降法，链式法则
循环神经网络	梯度下降法，链式法则
生成对抗网络	梯度下降法，链式法则，极值定理，Kullback-Leibler 散度，Jensen-Shannon 散度，测地距离，条件分布，互信息
K-means 算法	距离函数
贝叶斯网络	条件概率，贝叶斯公式，图
VC 维	Hoeffding 不等式

出现频率最高的是优化方法，拉格朗日乘数法，梯度下降法，牛顿法，凸优化

第二类概率论知识，随机变量，贝叶斯公式，随机变量独立性，正太分布，最大似然估计

第三类线性代数知识，几乎所有都会涉及到向量、矩阵、张量的计算，包括特征值和特征向量，很多算法都会最终变成求解特征值和特征向量问题。

微积分的知识比如链式法则。

除了主体这些数学知识，会用到微分几何中的流行、测地线、测地距离的概念。

支持向量机会用到 Mercer 条件、核函数，涉及到泛函分析和识别函数的范畴。

再比如说人工神经网络的证明，万能逼近定理会用到泛函分析和识别函数的内容，用来证明这样一个函数可以来逼近任何形式的函数。

离散数学的知识比如图论、树在机器学习里面也会用到，但是用的都是比较简单的。

所以说我们只有掌握好微积分、线性代数、概率论还有一些优化的算法，我们就能看懂所有

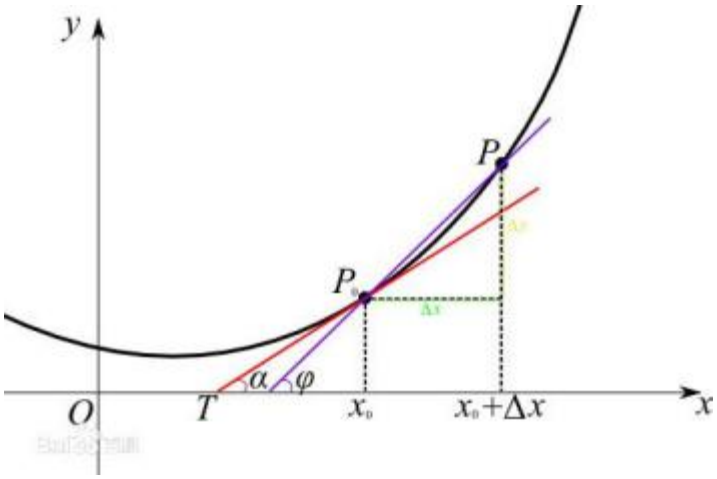
的机器学习算法了。像刚才说的一些相对高深的微分几何、泛函分析和识别函数，它们主要用在一些基础理论证明上面，说白了就是证明一些算法的合理性，你即使看不懂这些证明，它也不影响你理解这些算法的推导、思想和使用。

微积分

导数的定义

$$f'(x_0)=\lim_{\Delta x\to 0}\frac{f(x_0+\Delta x)-f(x_0)}{\Delta x}$$

导数：这是微积分里面最核心的概念，当函数的自变量Δx 趋近于 0 的时候，如果存在式子里面的值，那么函数可导



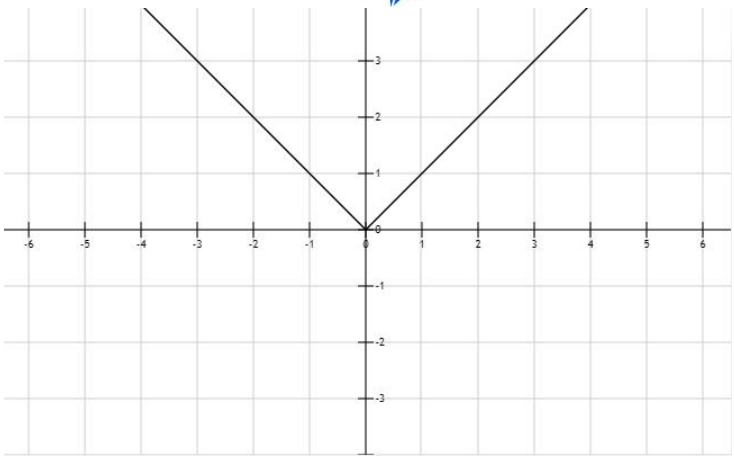
极限 limit 认为是高等数学和初等数学的分界线

左导数与右导数、可导函数

趋近于 0 有两个方向，从左边趋向于 0 是左导数，反之是右导数

下面的绝对值函数的左导数和右导数不相同，一个-1 一个+1，0 位置不可导

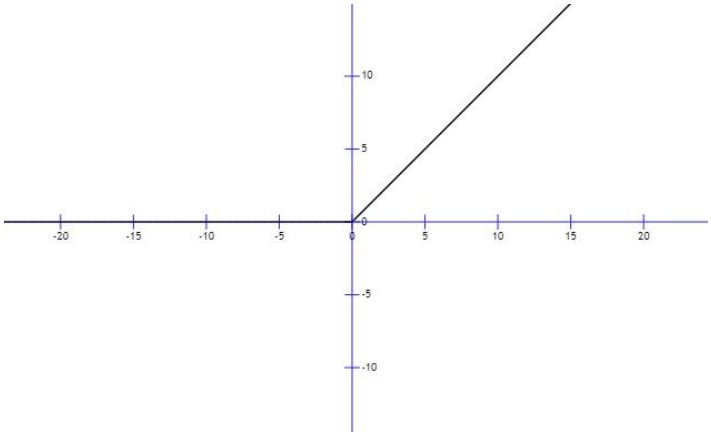
f(x)=|x|



Relu 函数

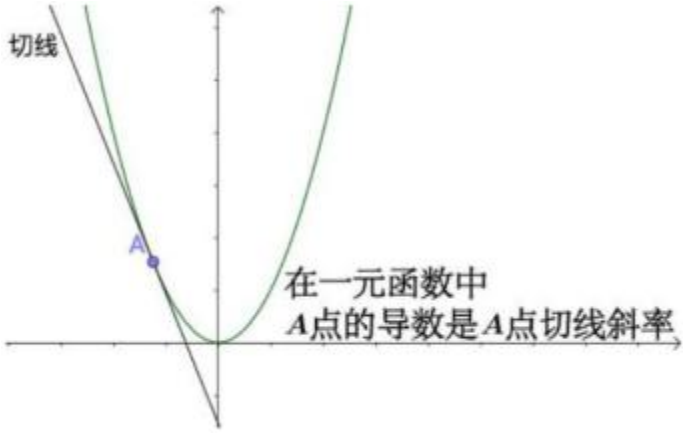
max(0,x)

$$f(x)=\begin{cases}x & x\geq 0\\ 0 & x<0\end{cases}$$



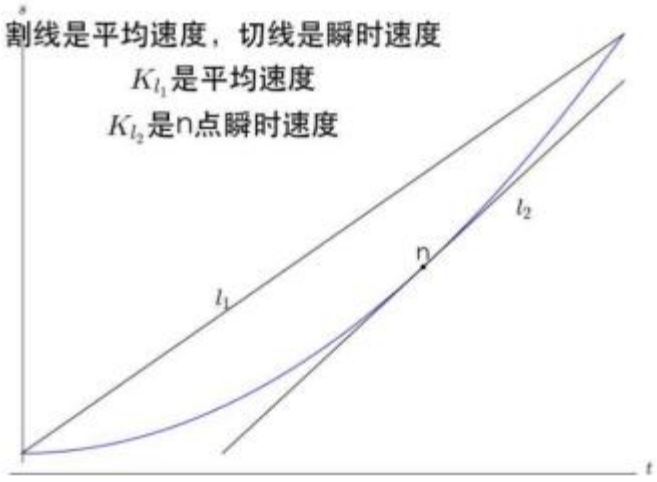
导数的几何意义与物理意义

几何意义，切线的斜率



物理意义，瞬时速度

$$f'(t)=\lim_{\Delta t\to 0}\frac{\Delta s}{\Delta t}$$



瞬时位移距离 除以 瞬时需要时间

求导公式

1. 基本函数
2. 四则运算
3. 复合函数

根据三种公式组合就可以求出任何公式的导数值

基本函数

幂函数

$$(x^a)'=ax^{a-1}$$

指数函数

$$(e^x)'=e^x$$

以 a 为底的指数函数

$$(a^x)'=a^x\ln a$$

对数函数

$$(\ln x)'=\frac{1}{x}$$

以任意为底的对数函数

$$(\log_a x)'=\left(\frac{\ln x}{\ln a}\right)'=\frac{1}{\ln a}\frac{1}{x}$$

导数的公式都可以根据下面的式子推导出来

定义

$$\lim_{n\to+\infty}(1+\frac{1}{n})^n=e$$

$$\lim_{x\to0}\frac{\sin x}{x}=1$$

三角函数的导数不要管，我们很少会用到它，而且三角函数很讨厌，是周期性函数，而我们的机器学习中很多时候要求是单调的函数，单调增也好，单调减也好，最好不要周期性函数

四则运算法则

导数加减乘除

$$(f(x)+g(x))'=f'(x)+g'(x)$$

$$(f(x)g(x))'=f'(x)g(x)+f(x)g'(x)$$

$$\left(\frac{f(x)}{g(x)}\right)'=\frac{f'(x)g(x)-f(x)g'(x)}{g^2(x)}$$

复合函数求导法则

$$(f(g(x)))'=f'(g)g'(x)$$

导数的用途

1. 求极值，往往设导数为 0，这里函数的导函数形式肯定得求
2. 神经网络里面激活函数会用到，其实还是求导数为 0 的情况，只不过是复合函数形式

例子

$$f(x)=\log(1+x^2+e^{2x})$$

高阶导数

前面学的是一阶导数，对导数再次求导就是高阶导数，二阶和二阶以上的导数统称为高阶导数。

$f''(x)$

$(f'(x))'$

$(5x^4)' = 20x^3$

$(5x^4)'' = (20x^3)' = 60x^2$

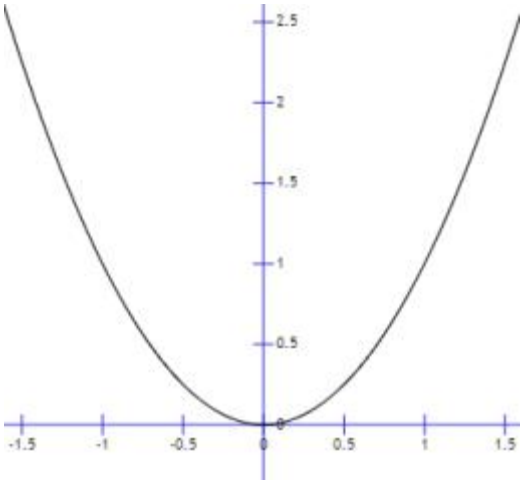
$f^{(n)}(x)$

导数与函数单调性的关系

$f'(x) > 0 \uparrow$ $f'(x) < 0 \downarrow$

函数的导数大于 0，函数是单调增的。函数的导数小于 0，函数是单调减的。

$f'(x) = 2x$



上图函数的导数是 2X，那么就是 x<0 的时候函数单调减，x>0 的时候函数单调增。

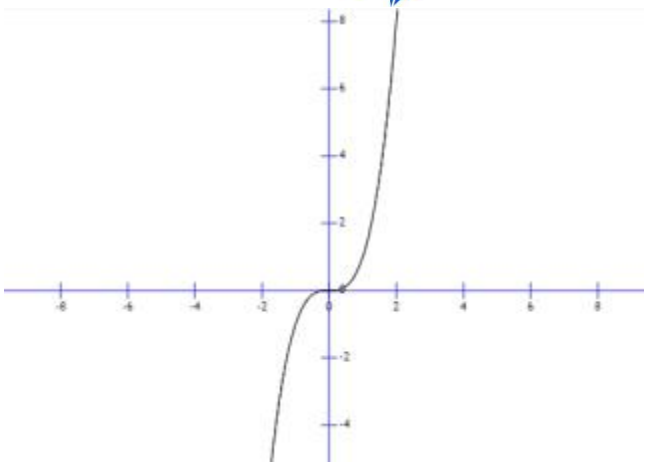
极值定理

导数为我们寻找极值提供依据，对于可导函数而言，因为在极值位置必然有函数的导数等于 0。

$$f'(x) = 0$$

极值处函数的导数等于 0，这是必要条件，但不是充分条件，因为极值处的导数必然等于 0，但是导数等于 0 处不代表一定是极值。

比如 X 的三次方：



导数与函数凹凸性的关系

函数的二阶导数是和函数的凹凸性是有关系的，凹凸性怎么定义的？
先来做简单的回顾，更多的会在最优化方法里面给大家讲，这里先记住凸函数是向下凸的，反正就是凹的，是否是凸函数可以通过二阶导数，如果二阶导数是大于 0 就是凸函数，

$$f''(x)>0$$

拿 X 的平方举例子，它的二阶导数是 2，大于 0 所以是凸函数。

$$f'(x)=0$$

称之为驻点，驻点是函数增减性的交替点，一侧增一侧减或一侧减一侧增

$$f''(x)=0$$

称之为拐点，拐点是凹凸性的，一侧凹一侧凸或一侧凸一侧凹
拿 X 的三次方举例子，一阶导是 3X 的平方，二阶导是 6X，这样当 X 小于 0 就是凹函数，X 大于 0 就是凸函数。

一元函数泰勒展开

$$f(x)=\frac{f(x_0)}{0!}+\frac{f'(x_0)}{1!}(x-x_0)+\frac{f''(x_0)}{2!}(x-x_0)^2+\ldots+\frac{f^{(n)}(x_0)}{n!}(x-x_0)^n+R_n(x)$$

$$f(x)=f(x_k)+(x-x_k)f'(x_k)+\frac{1}{2}(x-x_k)^2f''(x_k)+o^n$$

我们通过讲一元函数泰勒展开，把前面讲的导数给串起来，关于多元函数泰勒展开咱们后面讲。

泰勒展开是通过多项式函数来近似一个可导函数 f(x)，在 x=x0 处进行泰勒展开，如果函数 f(x)是 n 阶可导的。常数项+一阶项+二阶项 一直加到 n 的阶乘分之一乘以 n 阶导数。

泰勒展开在以前我们学高等数学的时候是非常有用的，它可以用来研究函数某些性质完成很多任务，在机器学习里面，它用来求函数的极值用的，很多时候函数 f(x)可能会非常复杂，我们去用泰勒展开做一个近似，梯度下降法怎么做的呢？是做一个近似，只保留泰勒展开一阶项，还有牛顿法，牛顿法是保留泰勒展开二阶项，忽略二阶以上的项，用二次函数来进行函数 f(x)。

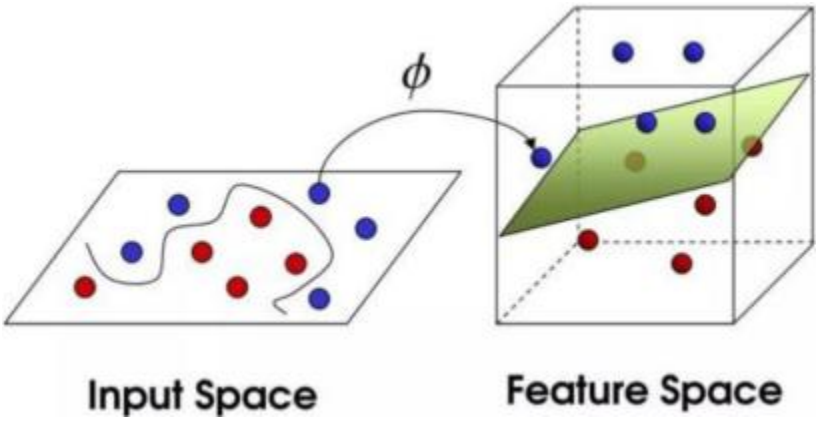
线性代数

向量与其运算

向量是线性代数里面最基本的概念，它其实就是一维数组，由 N 个数构成的，

$$X = (X1 \quad X2 \quad . \quad . \quad Xn)$$

向量的几何意义就是空间中的点，物理意义速度或者力这样的矢量，



向量的分量我们称之为维度，n 维向量集合的全体就构成了 n 维欧式空间，R^n

行向量和列向量

行向量是按行把向量排开，列向量是按列把向量排开

$$(X1 \quad X2 \quad . \quad . \quad Xn) \qquad \begin{matrix} X1 \\ X2 \\ . \\ . \\ Xn \end{matrix}$$

在数学中我们更多的把数据写成列向量，在编程语言中更多的把数据存成行向量

向量的运算 加法，数乘，减法，内积，转置

光上面定义向量是没有用的，肯定还要定义向量的一些运算，它有加减乘除一些运算，下面我们一一列举：

向量 X+Y

等于它们的分量分别相加，显然两个向量的长度得是相等的，减法我们在这里不列举，很容易举一反三

$$\begin{pmatrix}1\\2\\3\end{pmatrix}+\begin{pmatrix}4\\5\\6\end{pmatrix}=\begin{pmatrix}5\\7\\9\end{pmatrix}$$

数乘运算

它是一个数和这个向量每个分量相乘

$$2\times\begin{pmatrix}1\\2\\3\end{pmatrix}=\begin{pmatrix}2\\4\\6\end{pmatrix}$$

转置

把列向量变成行向量，把行向量变成列向量

$$\begin{pmatrix}1\\2\\3\end{pmatrix}^T=1\quad 2\quad 3 \qquad (1\quad 2\quad 3)^T=\begin{pmatrix}1\\2\\3\end{pmatrix}$$

运算法则

A+B+C=A+(B+C)

K*(X+Y)=KX+KY

向量的内积

两个列向量， X^TY ，等于对应位置相乘再相加

两个向量的内积的本质是变成一个标量

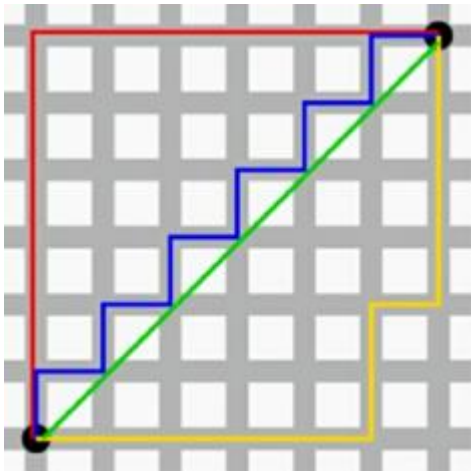
$$\begin{pmatrix}1 & 2 & 3\end{pmatrix}\begin{pmatrix}1\\1\\2\end{pmatrix}=1\times1+2\times1+3\times2=9$$

向量的范数

范数的公式是向量每个分量 绝对值 P 次方 再用幂函数计算 P 分之一，这里 P 肯定是整数 1，2，3...到正无穷都是可以的

向量的范数就是把向量变成一个标量，范数的表示就是两个竖线来表示，然后右下角写上 P

$$\|x\|_p=\left(\sum_{i=1}^{n}|x_i|^p\right)^{\frac{1}{p}}$$



1 范数是绝对值加和，1 范数写成 L1

$$\|x\|_1=\sum_{i=1}^{n}|x_i|$$

2 范数是平方加和开根号，其实代表的是向量的长度，高中时候学的向量的模，2 范数写成 L2

$$\|x\|_2=\sqrt{\sum_{i=1}^{n}(x_i)^2}$$

范数在后面是非常有用的，在后面讲正则项的时候会用到

特殊的向量

0 向量

就是分量全为 0 的向量

$$\begin{pmatrix}0 & 0 & . & . & 0\end{pmatrix}$$

单位向量

就是 L2 范数/模/长度为 1 的向量

矩阵与其运算

$$\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}$$

矩阵就是二维数组，上面是一个 m 乘 n 的矩阵，它有 m 行，n 列，每行每列上面都有一个元素，每个元素都有行标 i 和列标 j， a_{ij}

方阵，对称矩阵，单位矩阵，对角线

下面介绍几种特殊的矩阵，如果 m 等于 n，那就称为方阵

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}$$

对称矩阵

定义是 a_{ij} 等于 a_{ji} 那么就是对称矩阵，肯定是个方阵

$$\begin{bmatrix} 1 & 1 & 2 \\ 1 & 2 & 3 \\ 2 & 3 & 3 \end{bmatrix}$$

单位矩阵

主对角线都是 1，其它位置是 0，这称之为单位阵，单位矩阵写为 I，一定是方阵，等同于数字里面的 1

$$\begin{bmatrix} 1 & \ldots & 0 & 0 & 0 \\ \ldots & 1 & \ldots & 0 & 0 \\ 0 & \ldots & \ldots & \ldots & 0 \\ 0 & 0 & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & \ldots & 1 \end{bmatrix}$$

对角阵，就是主对角线非 0，其它位置是 0

$$\begin{bmatrix} \lambda_1 & \ldots & 0 & 0 & 0 \\ \ldots & \lambda_2 & \ldots & 0 & 0 \\ 0 & \ldots & \ldots & \ldots & 0 \\ 0 & 0 & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & \ldots & \lambda_n \end{bmatrix}$$

矩阵的运算 加法，数乘，减法，转置

矩阵的加减

矩阵的加法就是矩阵的对应位置相加，减法也是一样就是对应位置相减

$$\begin{bmatrix}1 & 2 & 3\\0 & 0 & 0\end{bmatrix}+\begin{bmatrix}4 & 5 & 6\\0 & 0 & 0\end{bmatrix}=\begin{bmatrix}5 & 7 & 9\\0 & 0 & 0\end{bmatrix}$$

数乘

$$5\times\begin{bmatrix}1 & 2 & 3\\0 & 0 & 0\end{bmatrix}=\begin{bmatrix}5 & 10 & 15\\0 & 0 & 0\end{bmatrix}$$

矩阵还有一种非常特殊的操作

转置

转置的操作和向量是一样的，就是把 aij 变成 aji，把行和列互换一下

$$a_{ij}\Rightarrow a_{ji}$$

$$\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix}^{T}=\begin{bmatrix}1 & 4\\2 & 5\\3 & 6\end{bmatrix}$$

矩阵的乘法

矩阵的乘法和一般的乘法是不太一样的

它是把第一个矩阵的每一行，和第二个矩阵的每一列拿过来做内积得到结果

$$\begin{bmatrix}1 & 2 & 3\\0 & 0 & 0\end{bmatrix}\begin{bmatrix}1 & 0\\1 & 0\\1 & 0\end{bmatrix}=\begin{bmatrix}6 & 0\\0 & 0\end{bmatrix}$$

$$m\times n \quad n\times k \quad m\times k$$

满足分配律，结合律，和交换律

A+B+C=A+(B+C)

加法肯定是满足，重点看乘法

首先乘法是满足结合律的

(AB)C=A(BC)

满足分配律的，这里是左分配律，和右分配律

(A+B)C=AC+BC

A(B+C)=AB+AC

特别强调的是矩阵是不满足交换律的，不一定相等，甚至 AB 的尺寸和 BA 的尺寸是不同的

AB≠BA

还有一个特殊的转置的公式

$$(AB)^{T}=B^{T}A^{T}$$

逆矩阵

矩阵有 AB，但是没有 A/B 这么一说，只有逆矩阵
逆矩阵怎么定义的？
假设有个矩阵 A，注意它一定是方阵，乘以矩阵 B 等于 I
AB=I
或者
BA=I
I 为单位矩阵，那么我们称这里的 B 为 A 的右逆矩阵，和左逆矩阵
有个很重要的结论就是，如果这样的 B 存在的话，它的左逆和右逆一定相等，统称为 A 的 -1
矩阵求逆有什么用呢？它可以帮助我们解线性方程组，比如 AZ=B
两边同时乘以 A 的逆，那么 Z=A 的-1 乘以 B，它发明的目的也是干这样的事情用的
从这里我们也可以看出来单位矩阵像我们乘法里面的 1
下面我们看一下公式：

$$(AB)^{-1} = B^{-1}A^{-1}$$

$$(A^{-1})^{-1} = A$$

$$(A^T)^{-1} = (A^{-1})^T$$

行列式

行列式其实在机器学习中用的并不多，一个矩阵必须是方阵，才能计算它的行列式
行列式是把矩阵变成一个标量

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

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32} - a_{13}a_{22}a_{31} - a_{12}a_{21}a_{33} - a_{11}a_{23}a_{32}$$

前面讲的是行列式的计算方式，下面我们看下行列式的性质，当然这都是方阵而言的

$$|AB| = |A||B|$$

$$|A^{-1}| = |A|^{-1}$$

$$|\alpha A| = \alpha^n |A|$$

数乘α，相当于α的 n 次方乘以 A 的行列式，因为刚才我们看计算方式的时候，相当于每一列都乘上了α所以是 n 阶的嘛

$$|\alpha A| = \begin{vmatrix} \alpha \cdot a11 & \alpha \cdot a12 & \alpha \cdot a13 \\ \alpha \cdot a21 & \alpha \cdot a22 & \alpha \cdot a23 \\ \alpha \cdot a31 & \alpha \cdot a32 & \alpha \cdot a33 \end{vmatrix}$$

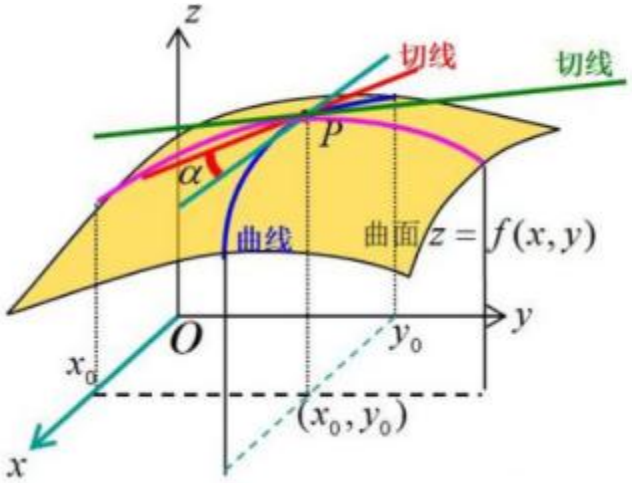
$$= \alpha \cdot a_{11}\alpha \cdot a_{22}\alpha \cdot a_{33} + \alpha \cdot a_{12}\alpha \cdot a_{23}\alpha \cdot a_{31} + \alpha \cdot a_{13}\alpha \cdot a_{21}\alpha \cdot a_{32}$$
$$- \alpha \cdot a_{13}\alpha \cdot a_{22}\alpha \cdot a_{31} - \alpha \cdot a_{12}\alpha \cdot a_{21}\alpha \cdot a_{33} - \alpha \cdot a_{11}\alpha \cdot a_{23}\alpha \cdot a_{32}$$
$$= \alpha^3 |A|$$

多元函数的微分学

偏导数

$$\frac{\partial f}{\partial x_i} = \lim_{\Delta x_i \to 0} \frac{f(x_1,...,x_i + \Delta x_i,...,x_n) - f(x_1,...,x_i,...,x_n)}{\Delta x_i}$$

偏导数，可以看作是导数的推广，对于多元函数来说，我们把其它的自变量固定不动，看成是常量，我们对其中的某一个变量求导数的话，那就是偏导数了，偏偏对一个变量求导数！



几何意义上面来说就是在某个方向上对原函数来切一下，再去求导，就是偏导数

$$f(x,y) = x^2 + xy - y^2$$

$$\frac{\partial f}{\partial x} = 2x + y$$

$$\frac{\partial f}{\partial y} = x - 2y$$

有些时候我们也可以更简洁的写

$f'x \quad f'y$

高阶偏导数

有高阶导数的话，同样我们有高阶偏导数，它的情况比高阶导数要复杂一些，因为它的求导变量有多个，比如说

$$\frac{\partial^2 f}{\partial x \partial y}$$

它对 x,y 求高阶偏导数的话，就是先对 x 求偏导，再对 y 求偏导，其实跟一元函数的高阶导数是一样的，依次对每个变量反复求导呗，我们还是以上面的公式为例

$$f(x,y)=x^2+xy-y^2$$

对 x 求偏导，然后再对 x 求偏导就等于 2 了

$$\frac{\partial^2 f}{\partial^2 x}=2$$

$$\frac{\partial^2 f}{\partial x \partial y}=1$$

$$\frac{\partial^2 f}{\partial y \partial x}=1$$

$$\frac{\partial^2 f}{\partial^2 y}=-2$$

有个重要的结论，就是高阶导数和求导次序无关

$$\frac{\partial^2 f}{\partial x \partial y}=\frac{\partial^2 f}{\partial y \partial x}$$

梯度

机器学习中的梯度下降法，和牛顿法很多地方都会用到这个概念的

$$\nabla f(x)=\left(\frac{\partial f}{\partial x_1},\frac{\partial f}{\partial x_2},\ldots,\frac{\partial f}{\partial x_n}\right)^T$$

梯度可以看成是一元函数它的导数，对于多元函数的推广

对于多元函数如果它的自变量有 N 个

$x_1 \quad x_2 \quad . \quad . \quad x_n$

它的梯度是个向量，是由对 X1 X2 等的偏导数构成的这样一个向量，称之为梯度

梯度我们用倒三角这个符号来表示作用于 f(x)得到这样一个向量，式子里面的 T 表示往往我们把它转置一下，看成是列向量

雅可比矩阵

这个可能很多同学学高等代数的时候可能没有学过，但是这个也比较好理解，就是由一阶偏导数构成的矩阵，发明它的目的主要是为了简化求导公式，对多元的复合函数求导，如果我们用雅可比矩阵来计算的话，它会写起来非常简洁，这在我们的人工神经网络反向推导的过程中往往会看到的

$$y = f(x)$$
$$\uparrow \quad \uparrow$$
$$k \quad n$$

假设有这样一个函数可以把 n 维 x 向量映射为 k 维的向量 y

$$y_i = f(x_i)$$

其中每个 xi 和每个 yi 都相关的，也就是每个 yi 是单独从 xi 映射过来的函数

它的雅可比矩阵就是每个 yi 分别对每个 xi 求偏导，然后构成的矩阵叫做雅可比矩阵

第一行就是 y1 对 X1 X2 到 Xn 求偏导，第二行就是 y2 对 X1 X2 到 Xn 求偏导，

第 k 行就是 yk 对 X1 X2 到 Xn 求偏导，

$$\begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_2}{\partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial y_k}{\partial x_1} & \frac{\partial y_k}{\partial x_2} & \cdots & \frac{\partial y_k}{\partial x_n} \end{bmatrix}$$

如果 xi 是 n 维向量，y 是 k 个值的结果，那么雅可比矩阵就是 k*n 的矩阵

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \quad \begin{matrix} y_1 = x_1^{\,2} + 2x_1x_2 + x_3 \\ y_2 = x_1 - x_2^{\,2} + x_3^{\,2} \end{matrix} \quad \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

如果 x1,x2,x3 会映射成为 y1,y2，y1 是 x1,x2,x3 的函数，y2 也是 x1,x2,x3 的函数，那么它的雅可比矩阵是怎么构成的呢？

$$\begin{bmatrix} 2x_1 + 2x_2 & 2x_1 & 1 \\ 1 & -2x_2 & 2x_3 \end{bmatrix}$$

Hessian 矩阵

它是对于一个多元函数来说的，它就相当于一元函数的二阶导数

怎么定义的呢？有一个 n 元函数，比方说 X1，X2 一直到 Xn

$$\begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

它的 hessian 矩阵是一个 n*n 的矩阵，矩阵里面的元素是什么呢？

它的所有的元素是二阶偏导数构成的，第一个元素是对 X1 求二阶偏导数，第二个元素是对 X1X2 求偏导数，因为咱们前面讲过，多元函数高阶偏导数和顺序无关，所以 hessian 矩阵是对称矩阵

$f(x,y,z) = 2x^2 - xy + y^2 - 3z^2$

下面这个例子先看一下它的一阶偏导数

$f'x = 4x - y$

$f'y = -x + 2y$

$f'z = -6z$

然后把 hessian 矩阵求出来

$$\begin{bmatrix} 4 & -1 & 0 \\ -1 & 2 & 0 \\ 0 & 0 & -6 \end{bmatrix}$$

Hessian 矩阵和函数的凹凸性是有密切关系的，如果 hessian 矩阵正定，可以说函数 f(x)是凸函数，如果是负定，它就是凹函数，矩阵正定怎么定义的呢？

极值判别法则

对于一元函数，我们前面讲过，f(x)的一阶导数等于 0 处有极值，当 f(x)的二阶导数大于 0 时是极小值，当 f(x)的二阶导数小于 0 时是极大值，可以参考 X 的平方这个函数

多元函数的极值判别法则

首先 f(x)的一阶导数等于 0，这点是驻点的话，那它就可能是极值点，它是极大值还是极小值或者不是极值怎么判定的？

看 hessian 矩阵，在 f(x)的一阶导数等于 0 处，就是驻点处，

如果 hessian 矩阵是正定的话，函数在该点有极小值

如果 hessian 矩阵是负定的话，函数在该点有极大值

如果 hessian 矩阵不定，还需要看更高阶的导数

对于任意向量 X≠0，都有 $x^T Ax > 0$，那就是正定矩阵，如果是≥的话，那就是半正定矩阵

怎么判断矩阵是正定的呢？就是拿这个式子去证明，

X^TX半正定：对于任意的非零向量u

$$uX^TXu=(Xu)^TXu \xrightarrow{令v=Xu} v^Tv\geq 0$$

但是这样不太容易，有时候我们会根据几个原则去判断：

矩阵的特征值全部大于 0（矩阵的特征值和特征向量我们会讲到）

矩阵的所有顺序主子式都大于 0（顺序主子式这个我们用的比较少）

矩阵合同于单位阵

线性代数

二次型

二次型就是纯二次项构成的一个函数

因为二次函数（方程）的二次部分最重要，为了方便研究，我们把含有 n 个变量的二次齐次函数：

$$f(x_1,x_2,\cdot,x_n)=a_{11}x_1^2+a_{22}x_2^2+\cdots+a_{nn}x_n^2+2a_{12}x_1x_2+2a_{13}x_1x_3+\cdots+2a_{n-1,n}x_{n-1}x_n$$

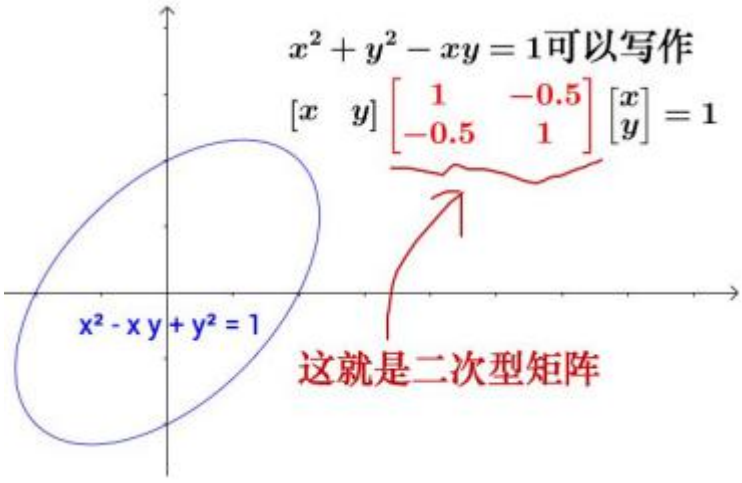
称为二次型。

它就是一个数，它是怎么做的呢？

它就是一个向量和一个矩阵相乘的结果，X 是列向量，转置就是行向量

$$x^TAx$$

$$\begin{pmatrix}x_1 & \dots & \dots & x_n\end{pmatrix}\begin{bmatrix}a_{11} & a_{12} & \dots & a_{1n}\\ a_{21} & a_{22} & \dots & a_{2n}\\ \dots & \dots & \dots & \dots\\ a_{n1} & a_{n2} & \dots & a_{nn}\end{bmatrix}\begin{pmatrix}x_1\\ \dots\\ \dots\\ x_n\end{pmatrix}$$



$ax^2+2bxy+cy^2$可以写作

$$[x \quad y]\begin{bmatrix}a & b\\ b & c\end{bmatrix}\begin{bmatrix}x\\ y\end{bmatrix}=1$$

$$\left.\begin{array}{r}[x \quad y]\begin{bmatrix}a & b\\ b & c\end{bmatrix}\begin{bmatrix}x\\ y\end{bmatrix}=1\\ X=\begin{bmatrix}x\\ y\end{bmatrix}\\ A=\begin{bmatrix}a & b\\ b & c\end{bmatrix}\end{array}\right\}\Longrightarrow X^TAX$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n}a_{ij}x_ix_j$$

当然这是对方阵而言的，aij 有 n 的平方这么多项，这种形式在机器学习中会见到，比如是一次型的

$$f(x;w)=w^Tx+b$$

或者二次型的

$$f(x;w)=x^Twx+b$$

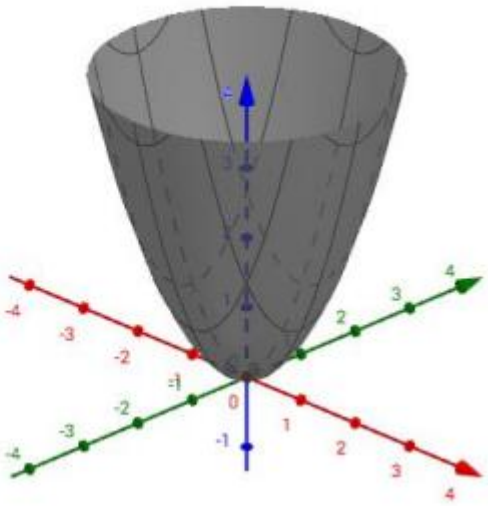
例如该场景的数据分布是一次型的，那我们就可以选择 logistic regression、SVM 等分界面为一次型的模型；如果场景的数据分布是二次型的，我们可以选择 naive bayes；如果场景的数据分布既不是一次型也不是二次型，那我们可以选择基于决策树的模型，例如 gbdt、random forest 等，或者 DNN，这些模型都高度非线性，表达能力极强，理论上可以拟合任意曲线

回看 Hessian 矩阵：

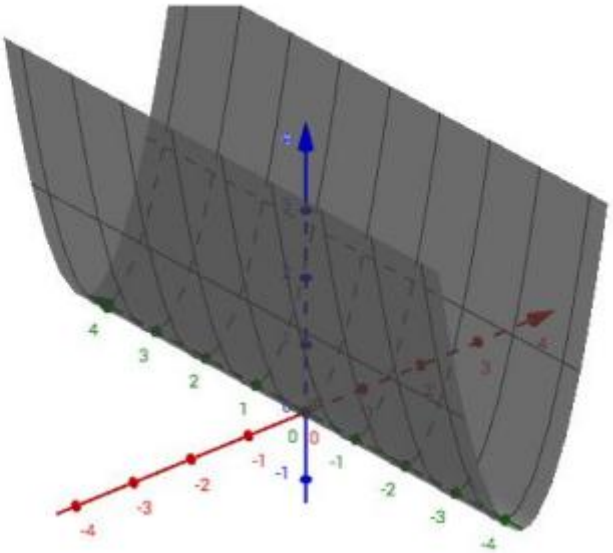
对于二次型函数，$f(x)=x^TAx$：

- $f(x)>0, x\neq 0, x\in\mathbb{R}$，则 f 为正定二次型，A 为正定矩阵
- $f(x)\geq 0, x\neq 0, x\in\mathbb{R}$，则 f 为半正定二次型，A 为半正定矩阵
- $f(x)<0, x\neq 0, x\in\mathbb{R}$，则 f 为负定二次型，A 为负定矩阵
- $f(x)\leq 0, x\neq 0, x\in\mathbb{R}$，则 f 为半负定二次型，A 为半负定矩阵
- 以上皆不是，就叫做不定

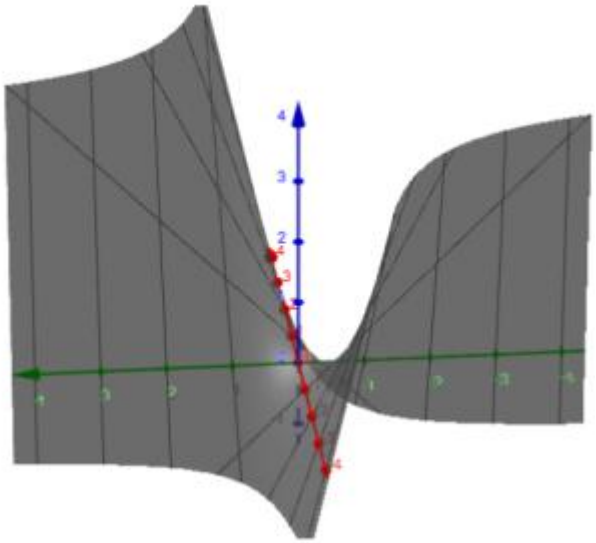
从图像上看，这是正定：

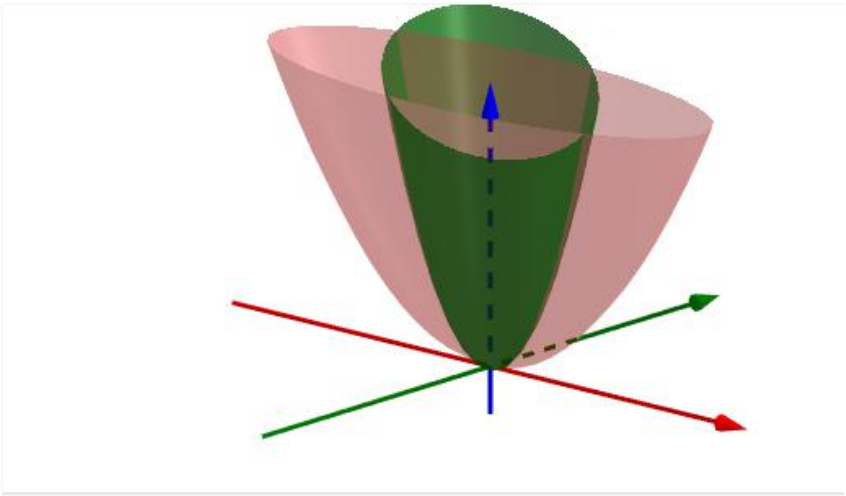


半正定：



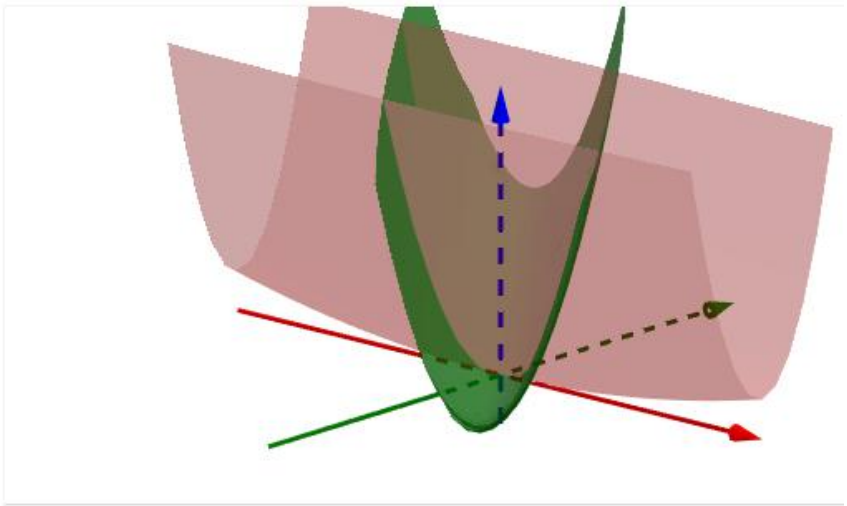
不定:



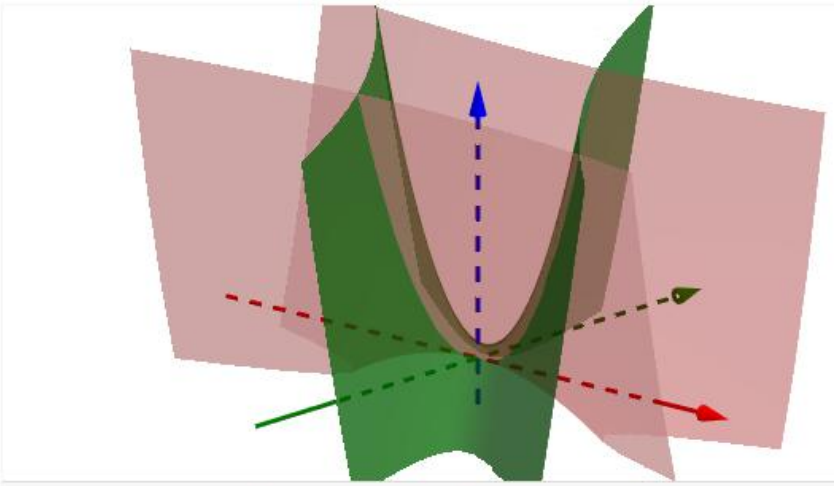


$$\begin{pmatrix} 1.15 & 0.9 \\ 0.9 & 1.25 \end{pmatrix}\begin{pmatrix} 0.3 & 0 \\ 0 & 2.1 \end{pmatrix}$$

a = 1.15 b = 0.9 c = 1.25



$$\begin{pmatrix} 1.15 & 0.9 \\ 0.9 & 0.75 \end{pmatrix}\begin{pmatrix} 0.03 & 0 \\ 0 & 1.87 \end{pmatrix}$$

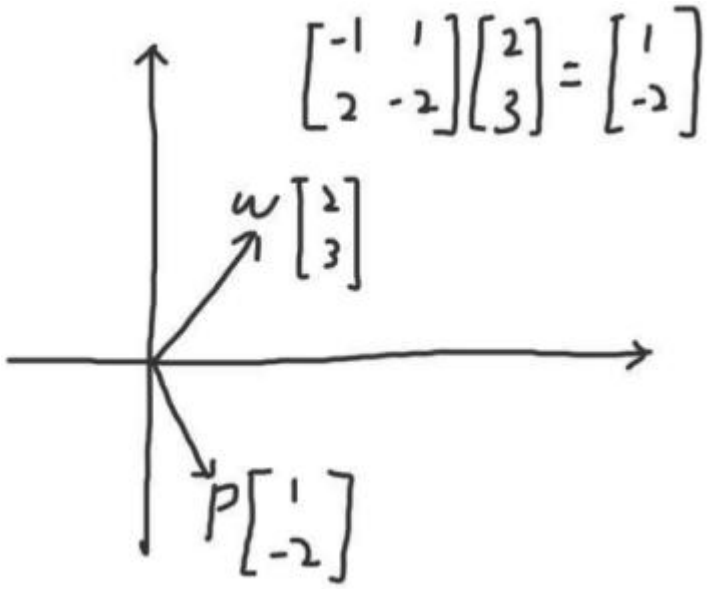


$$\begin{pmatrix} 1.15 & 0.9 \\ 0.9 & 0.5 \end{pmatrix}\begin{pmatrix} -0.13 & 0 \\ 0 & 1.78 \end{pmatrix}$$

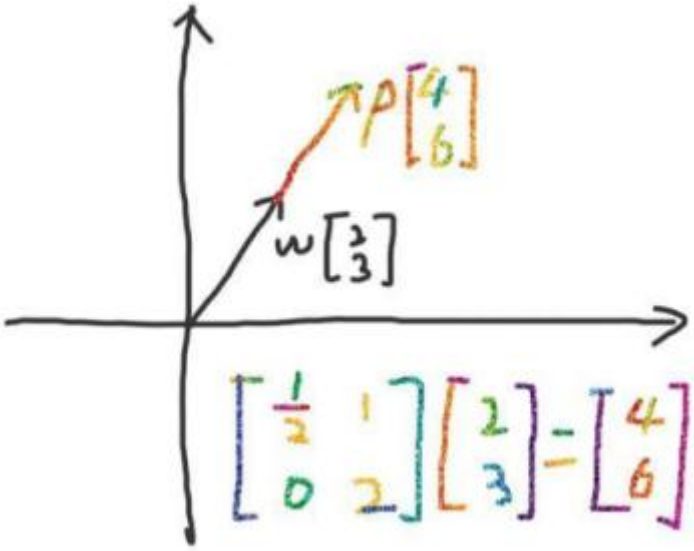
a = 1.15 b = 0.9 c = 0.5

特征值和特征向量

我们已经知道，矩阵和向量的乘法就相当于对该向量做了一个线性变换。在这个变换中，大部分的向量都发生了偏移，脱离了原“轨道”。举个例子：



如上图所示，向量 w（2,3）在矩阵[-1,1;2,-2]的作用下，线性变换为另一个向量 p（1，-2）。p 和 w 明显不在一条直线上，发生了偏移。那有没有一个矩阵，恰好使线性变换之后新向量 p 和原来的向量 w 在同一条直线上呢？我们再看一个例子：



如上图，还是向量 w（2,3），此时线性变换的矩阵变成了[1/2,1;0,2]，产生的新向量 p（4,6）跟 w 还在同一条直线上。

怎么定义的呢？

设 A 是 n 阶方阵，如果存在数λ和非零 n 维列向量 x，使得 Ax=λx 成立，则称 λ 是矩阵 A 的一个特征值（characteristic value)或本征值（eigenvalue)。

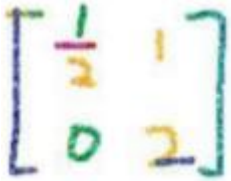
$$Ax = \lambda x$$

式 Ax=λx 也可写成$(A - \lambda \mathrm{I})x = 0$。这是 n 个未知数 n 个方程的齐次线性方程组，它有非零解的充分必要条件是系数行列式$|A - \lambda \mathrm{I}| = 0$。

$$|\lambda E - A| = \begin{vmatrix} \lambda - a_{11} & -a_{12} & \cdots & -a_{1n} \\ -a_{21} & \lambda - a_{22} & \cdots & -a_{2} \\ \cdots & \cdots & \cdots & \cdots \\ -a_{n1} & -a_{2n} & \cdots & \lambda - a_{nn} \end{vmatrix}$$

$$= \lambda^n + \alpha_1\lambda^{n-1} + \alpha_2\lambda^{n-2} + \cdots + \alpha_{n-1}\lambda + \alpha_n$$

展开之后是一个关于λ的一个 n 次方程，直接可以通过方程的求根来算



$$\lambda^2 - 2.5\lambda + 1 = (\lambda - 0.5)(\lambda - 2)$$

但是当 n 的规模很大，就是矩阵的阶数大于等于 5，这个是没法求的，因为我们数学里面有个重要的结论，5 次和 5 次以上的代数方程它是没有求根公式的，所以这么做是非常困难的在工程上面我们去计算矩阵的特征值的话，一般是用 QR 算法，

$A^{(0)} = A$
for k = 1, 2, ...
  $Q^{(k)}R^{(k)} = A^{(k-1)}$ QR factorization of $A^{(k-1)}$
  $A^{(k)} = R^{(k)}Q^{(k)}$ Recombine factors in reverse order

http://madrury.github.io/jekyll/update/statistics/2017/10/04/qr-algorithm.html

对于方程的求根，有个重要的定理叫做韦达定理

设一元二次方程 $ax^2 + bx + c = 0(a, b, c \in R, a \neq 0)$ 中，两根x_1、x_2有如下关系：

$$x_1 + x_2 = -\frac{b}{a}$$

$$x_1 x_2 = \frac{c}{a}$$

就是初中时候学过，二次函数求根公式导出的韦达定理，

由一元二次方程求根公式知：$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

则有：

$$x_1 + x_2 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} + \frac{-b - \sqrt{b^2 - 4ac}}{2a} = -\frac{b}{a}$$

$$x_1 \cdot x_2 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \times \frac{-b - \sqrt{b^2 - 4ac}}{2a} = \frac{c}{a}$$

对于 n 次方程同样有这样一个结论，trace 就是 A 矩阵的主对角线的值加和等于所有特征值λ的和

$$tr(A)=\sum_{i}^{n}a_{ii}=\lambda_1+\lambda_2+...+\lambda_n=\sum_{i}^{n}\lambda_i$$

所有特征值的乘积等于 A 的行列式的

$$\prod_{i=1}^{n}\lambda_i=|A|$$

特征值和特征向量在机器学习中会被用到，像 PCA 主成分分析，LDA 线性判别分析，以及其它算法里面都会用到它的理论和方法

特征值分解

特征值和特征向量的定义如下：

$$Ax=\lambda x$$

其中 A 是一个 n×n 的矩阵，x 是一个 n 维向量，则我们说λ是矩阵 A 的一个特征值，而 x 是矩阵 A 的特征值λ所对应的特征向量。

求出特征值和特征向量有什么好处呢？ 就是我们可以将矩阵 A 特征分解。如果我们求出了矩阵 A 的 n 个特征值

$$\lambda_1\le\lambda_2\le...\le\lambda_n$$

以及这 n 个特征值所对应的特征向量

$$(w_1 \quad w_2 \quad ... \quad w_n)$$

那么矩阵 A 就可以用下式的特征分解表示：

$$A=W\sum W^{-1}$$

特征值分解怎么来的？

我们以前在学线性代数的时候都学过我们怎么把一个矩阵变化成一个对角矩阵

定理1 令 $\boldsymbol{M}$ 为n×n矩阵，其特征值为 $\lambda_1,\lambda_2,...,\lambda_n$ ，特征向量为 $\boldsymbol{V}_1,\boldsymbol{V}_1,...,\boldsymbol{V}_n$ ，形成线性无关集合，以每个特征向量为列构成矩阵 $\boldsymbol{A}$ ，如下所示。

$$\boldsymbol{A}=[\boldsymbol{V}_1 \quad \boldsymbol{V}_2 \quad \cdots \quad \boldsymbol{V}_n].$$

矩阵 $\boldsymbol{A}$ 可以将矩阵 $\boldsymbol{M}$ 对角化，乘积矩阵 $\boldsymbol{A}^{-1}\boldsymbol{M}\boldsymbol{A}$ 的主对角元素是矩阵 $\boldsymbol{M}$ 的特征值：

$$\boldsymbol{A}^{-1}\boldsymbol{M}\boldsymbol{A}=\begin{pmatrix}\lambda_1 & 0 & \cdots & 0\\ 0 & \lambda_2 & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \cdots & \lambda_n\end{pmatrix}.$$

反之，如果存在可逆矩阵 $\boldsymbol{A}$ ，使 $\boldsymbol{A}^{-1}\boldsymbol{M}\boldsymbol{A}$ 为对角矩阵，则矩阵 $\boldsymbol{A}$ 的列等于矩阵 $\boldsymbol{M}$ 的特征向量， $\boldsymbol{A}^{-1}\boldsymbol{M}\boldsymbol{A}$ 的主对角元素为矩阵 $\boldsymbol{M}$ 的特征值 [1] 。

证明：首先计算矩阵乘积 $\boldsymbol{MA}$。由于矩阵 $\boldsymbol{A}$ 的第j列对应特征向量 $\boldsymbol{V}_j$，则 $\boldsymbol{MA}$ 的第j列等于 $\boldsymbol{MV}_j$。由于 $\boldsymbol{V}_j$ 为特征向量，则 $\boldsymbol{MV}_j=\lambda_j\boldsymbol{V}_j$，矩阵乘积 $\boldsymbol{MA}$ 可写为

$$\boldsymbol{MA}=[\lambda_1\boldsymbol{V}_1 \quad \lambda_2\boldsymbol{V}_2 \quad \cdots \quad \lambda_n\boldsymbol{V}_n]$$

$$=[\boldsymbol{V}_1 \quad \boldsymbol{V}_2 \quad \cdots \quad \boldsymbol{V}_n]\begin{pmatrix}\lambda_1 & 0 & \cdots & 0\\ 0 & \lambda_2 & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \cdots & \lambda_n\end{pmatrix}.$$

$$=\boldsymbol{A}\begin{pmatrix}\lambda_1 & 0 & \cdots & 0\\ 0 & \lambda_2 & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \cdots & \lambda_n\end{pmatrix}.$$

由于特征向量 $\boldsymbol{V}_j$ 线性无关，矩阵 $\boldsymbol{A}$ 可逆，$\boldsymbol{A}^{-1}\boldsymbol{MA}$ 的表达式可写为

$$\boldsymbol{A}^{-1}\boldsymbol{MA}=\boldsymbol{A}^{-1}\boldsymbol{A}\begin{pmatrix}\lambda_1 & 0 & \cdots & 0\\ 0 & \lambda_2 & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \cdots & \lambda_n\end{pmatrix}=\begin{pmatrix}\lambda_1 & 0 & \cdots & 0\\ 0 & \lambda_2 & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \cdots & \lambda_n\end{pmatrix}.$$

通过一个正交变换做到的，$P^{-1}AP=\Lambda$

Λ 是对角矩阵，它就是对角线的值是矩阵所有特征值构成的矩阵

$$\begin{bmatrix}\lambda_1 & & & & \\ & \lambda_2 & & & \\ & & \cdots & & \\ & & & \cdots & \\ & & & & \lambda_n\end{bmatrix}$$

P 是正交矩阵，正交矩阵的定义是 P 的逆等于 P 的转置

$P^{-1}=P^T$

正交矩阵的性质，是行和列相互之间是正交的，一个向量组 $(x_1,x_2,..,x_n)$，其中 Xi 和 Xj 内积是 0 当 i 不等于 j 时，如果 i=j，那么向量内积就是 1，其实它就是我们几何里面垂直的概念的抽象，

$P\Lambda P^{-1}=A$

我们可以把一个矩阵拆分成，一个正交阵和Λ还有正交阵的逆的乘积的，这就是我们的特征值分解

多元函数的泰勒展开

一元函数的泰勒展开

$$f(x)=\frac{f(x_0)}{0!}+\frac{f'(x_0)}{1!}(x-x_0)+\frac{f''(x_0)}{2!}(x-x_0)^2+...+\frac{f^{(n)}(x_0)}{n!}(x-x_0)^n+R_n(x)$$

$$f(x)=f(x_k)+(x-x_k)f^{'}(x_k)+\frac{1}{2}(x-x_k)^2f^{''}(x_k)+o^n$$

多元函数的泰勒展开

f(xk)是标量，而且是个常数，一次项，不过注意(x-xk)是个向量，然后 T 转置这里是把梯度和(x-xk)做内积，二次项的内积，是和 hessian 矩阵做内积，

$$f(x)=f(x_k)+[\nabla f(x_k)]^T(x-x_k)+\frac{1}{2}[x-x_k]^T H(x_k)[x-x_k]+o^n$$

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

$$x^T Ax \Rightarrow ax^2$$

多元函数泰勒展开是非常有用的，后面我们在推导梯度下降法，牛顿法，还有拟牛顿法的时候都是用的到的，所以同学们需要把它记住，同学们可以跟一元函数泰勒展开公式类比来记忆它

矩阵和向量的求导公式

$$\nabla w^T x = w$$

证明:

$$\nabla w^T x = w \Rightarrow \sum_{i=1}^{n} w_i x_i$$

对 Xi 求偏导得到的就是 Wi，所以对 x 求导就是 w

$$\begin{matrix}x_1\\x_2\\\vdots\\x_n\end{matrix}\begin{pmatrix}w_1\\w_2\\\vdots\\w_n\end{pmatrix}$$

$$\nabla x^T Ax = (A+A^T)x$$

二阶导就是再一次对 x 求导

$$\nabla^2 x^T Ax = A + A^T$$

$$\begin{aligned}\beta' V \beta &= \begin{pmatrix}\beta_1 & \beta_2 & \beta_3\end{pmatrix}\begin{pmatrix}V_{11} & V_{12} & V_{13}\\ V_{21} & V_{22} & V_{23}\\ V_{31} & V_{32} & V_{33}\end{pmatrix}\begin{pmatrix}\beta_1\\ \beta_2\\ \beta_3\end{pmatrix}\\ &= V_{11}\beta_1^2 + V_{22}\beta_2^2 + V_{33}\beta_3^2 + (V_{12}+V_{21})\beta_1\beta_2 + (V_{13}+V_{31})\beta_1\beta_3 + (V_{23}+V_{32})\beta_2\beta_3.\end{aligned}$$

$$\begin{aligned}
\frac{\partial(\beta' V\beta)}{\partial\beta} &= \begin{pmatrix} \frac{\partial(\beta' V\beta)}{\partial\beta_1} \\ \frac{\partial(\beta' V\beta)}{\partial\beta_2} \\ \frac{\partial(\beta' V\beta)}{\partial\beta_3} \end{pmatrix} \\
&= \begin{pmatrix} 2V_{11}\beta_1 + (V_{12}+V_{21})\beta_2 + (V_{13}+V_{31})\beta_3 \\ 2V_{22}\beta_2 + (V_{12}+V_{21})\beta_1 + (V_{23}+V_{32})\beta_3 \\ 2V_{33}\beta_3 + (V_{13}+V_{31})\beta_1 + (V_{23}+V_{32})\beta_2 \end{pmatrix} \\
&= \begin{pmatrix} 2V_{11} & V_{12}+V_{21} & V_{13}+V_{31} \\ V_{12}+V_{21} & 2V_{22} & V_{23}+V_{32} \\ V_{13}+V_{31} & V_{23}+V_{32} & 2V_{33} \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{pmatrix} \\
&= \left(\begin{pmatrix} V_{11} & V_{12} & V_{13} \\ V_{21} & V_{22} & V_{23} \\ V_{31} & V_{32} & V_{33} \end{pmatrix} + \begin{pmatrix} V_{11} & V_{21} & V_{31} \\ V_{12} & V_{22} & V_{32} \\ V_{13} & V_{23} & V_{33} \end{pmatrix} \right) \begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{pmatrix} \\
&= (V + V')\beta.
\end{aligned}$$

奇异值分解

微积分和线性代数最后一个知识点

$A = U\sum V^T$

这个比较抽象，前面我们知道一个矩阵可以进行分解，但这是对一个 n 阶方阵而言的，但如果 A 不是一个方阵这种分解（对角化）将是无效的

$A = P\Lambda P^{-1}$

有没有类似的分解呢？

有，就是我们的奇异值分解，而且奇异值分解比较神奇，可以应用于任意形状矩阵，如果 A 是一个 m*n 的矩阵，它一样可以分解成为三个矩阵的乘积，U V 和 Σ，其中 U V 都是正交矩阵，Σ是对角矩阵，注意这里Σ并不是一个方阵，而是 m*n 的矩阵.

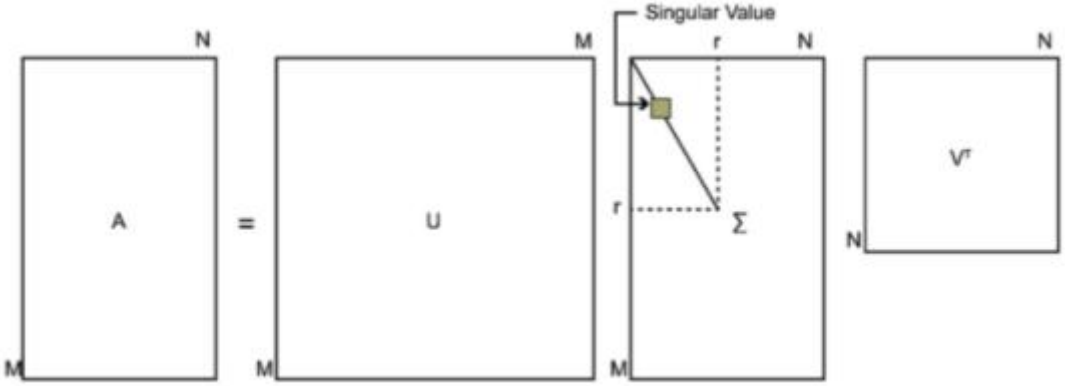
SVD 也是对矩阵进行分解，但是和特征分解不同，SVD 并不要求要分解的矩阵为方阵。假设我们的矩阵 A 是一个 m×n 的矩阵，那么我们定义矩阵 A 的 SVD 为： $A = U\sum V^T$

$$\begin{bmatrix} \lambda_1 & & & & 0 \\ & \lambda_2 & & & \\ & & \ddots & & \\ & & & \ddots & \\ 0 & & & & \lambda_n \end{bmatrix}$$

U 是 AA^T 的矩阵，所以是 m*m 的矩阵，AA^T 正交化特征向量构成的矩阵，$A^T A$ 是正交化特征向量所构成的矩阵，n*n 的矩阵。

其中 U 是一个 m×m 的矩阵，Σ是一个 m×n 的矩阵，除了主对角线上的元素以外全为 0，主对角线上的每个元素都称为奇异值，V 是一个 n×n 的矩阵。U 和 V 都是酉矩阵，即满足

$U^TU=\mathrm{I}, V^TV=\mathrm{I}$。下图可以很形象的看出上面 SVD 的定义：



奇异值分解如何求解

那么我们如何求出 SVD 分解后的 U,Σ,V 这三个矩阵呢？

如果我们将 A 的转置和 A 做矩阵乘法，那么会得到 n×n 的一个方阵 A^TA。既然 A^TA 是方阵，那么我们就可以进行特征分解，得到的特征值和特征向量满足：

$$(A^TA)v_i=\lambda_i v_i$$

这样我们就可以得到矩阵 A^TA 的 n 个特征值和对应的 n 个特征向量 v 了。将 A^TA 的所有特征向量张成一个 n×n 的矩阵 V，就是我们 SVD 公式里面的 V 矩阵了。一般我们将 V 中的每个特征向量叫做 A 的右奇异向量。

如果我们将 A 和 A 的转置做矩阵乘法，那么会得到 m×m 的一个方阵 AA^T。既然 AA^T 是方阵，那么我们就可以进行特征分解，得到的特征值和特征向量满足：$(AA^T)u_i=\lambda_i u_i$

这样我们就可以得到矩阵 AA^T 的 m 个特征值和对应的 m 个特征向量 u 了。将 AA^T 的所有特征向量张成一个 m×m 的矩阵 U，就是我们 SVD 公式里面的 U 矩阵了。一般我们将 U 中的每个特征向量叫做 A 的左奇异向量。

U 和 V 都求出来了，现在就剩下奇异值矩阵Σ没有求出了。由于Σ除了对角线上是奇异值其他位置都是 0，那我们只需要求出每个奇异值σ就可以了。我们注意到:

$$A=U\sum V^T$$
$$AV=U\sum V^TV$$
$$AV=U\sum$$
$$Av_i=\sigma_i u_i$$
$$\sigma_i=Av_i/u_i$$

这样我们可以求出我们的每个奇异值，进而求出奇异值矩阵Σ。

上面还有一个问题没有讲，就是我们说 A^TA 的特征向量组成的就是我们 SVD 中的 V 矩阵，而 AA^T 的特征向量组成的就是我们 SVD 中的 U 矩阵，这有什么根据吗？这个其实很容易证明，我们以 V 矩阵的证明为例。

$$A = U\sum V^T$$
$$A^T = V\sum^T U^T$$
$$A^T A = V\sum^T U^T U\sum V^T$$
$$= V\sum^2 V^T$$

上式证明使用了:$U^T U = \mathrm{I}, \sum^T \sum = \sum^2$。可以看出 $A^T A$ 的特征向量组成的就是 SVD 中的 V 矩阵。类似的方法可以得到 AA^T 的特征向量组成的就是 SVD 中的 U 矩阵。

进一步我们还可以看出我们的特征值矩阵等于奇异值矩阵的平方，也就是说特征值和奇异值满足如下关系：$\sigma_i = \sqrt{\lambda_i}$

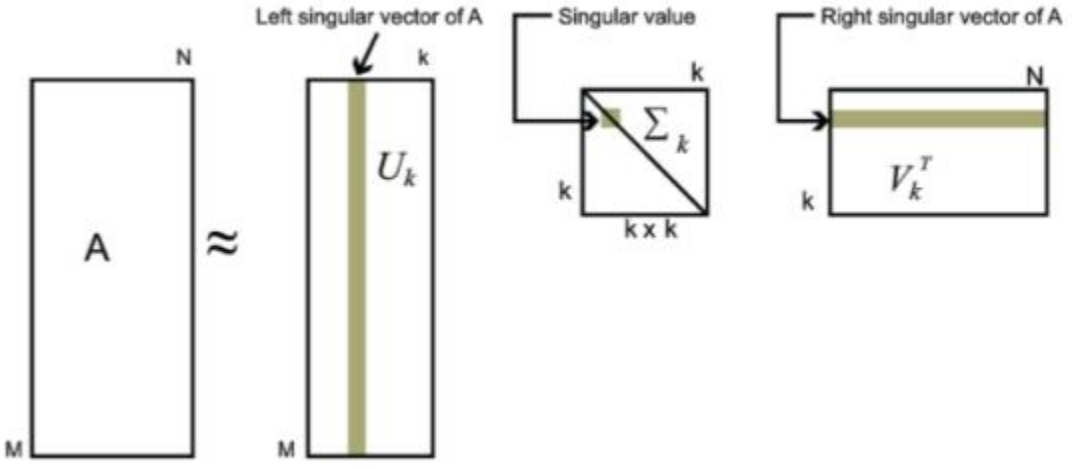
这样也就是说，我们可以不用 $\sigma_i = Av_i / u_i$ 来计算奇异值，也可以通过求出 $A^T A$ 的特征值取平方根来求奇异值。

奇异值分解性质

上面我们对 SVD 的定义和计算做了详细的描述，似乎看不出我们费这么大的力气做 SVD 有什么好处。那么 SVD 有什么重要的性质值得我们注意呢？

对于奇异值,它跟我们特征分解中的特征值类似，在奇异值矩阵中也是按照从大到小排列，而且奇异值的减少特别的快，在很多情况下，前 10%甚至 1%的奇异值的和就占了全部的奇异值之和的 99%以上的比例。也就是说，我们也可以用最大的 k 个的奇异值和对应的左右奇异向量来近似描述矩阵。也就是说：

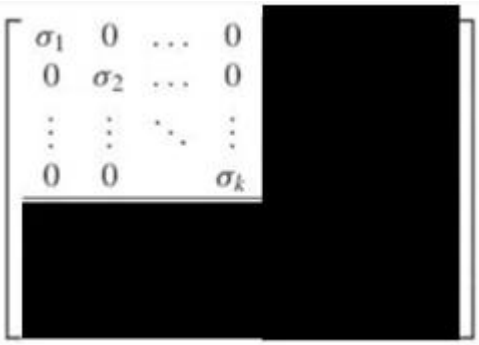
$A_{m\times n} = U_{m\times m}\sum_{m\times n} V^T_{n\times n} \approx U_{m\times k}\sum_{k\times k} V^T_{k\times n}$。其中 k 要比 n 小很多，也就是一个大的矩阵 A 可以用三个小的矩阵 $U_{m\times k}, \sum_{k\times k}, V^T_{k\times n}$ 来表示。如下图所示，现在我们的矩阵 A 只需要三个小矩阵就可以近似描述了。



U 的每一列都是标准正交的（orthonormal），V 的转置都是每一行都是标准正交的，Σ是呈对角线的（diagonal），只有对角线有值，而且一定都是非负的，意味着对角线一定有值，但是也有可能值是 0，而且神奇的地方，如果左上角有值σ1，第二个左上角值是σ2，那么σ1 一定大于等于σ2，并且σ2 大于等于σ3，以此类推。

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_k > 0$$

所以Σ中有 k 个非 0 的值，依次从左上角往右下角方向排列，我们可以把Σ中等于 0 的行列去掉就保留 k*k 个值，那么 U 和 V 中也分别只保留 k 列和 k 行，我们会发现一样乘回来可以得到 A 仍然不变。



如果我们保留的小于 k 个，那么乘回去的矩阵就不等于原来的 A 了。如果去掉的是第 k 个，那么乘回来的矩阵就是 A+，是最接近 A 的矩阵

还比如可以求解对称矩阵的逆矩阵，可以用于降维算法中的特征分解，还可以用于推荐系统，以及自然语言里面的主题分析里面，会用到这个算法的

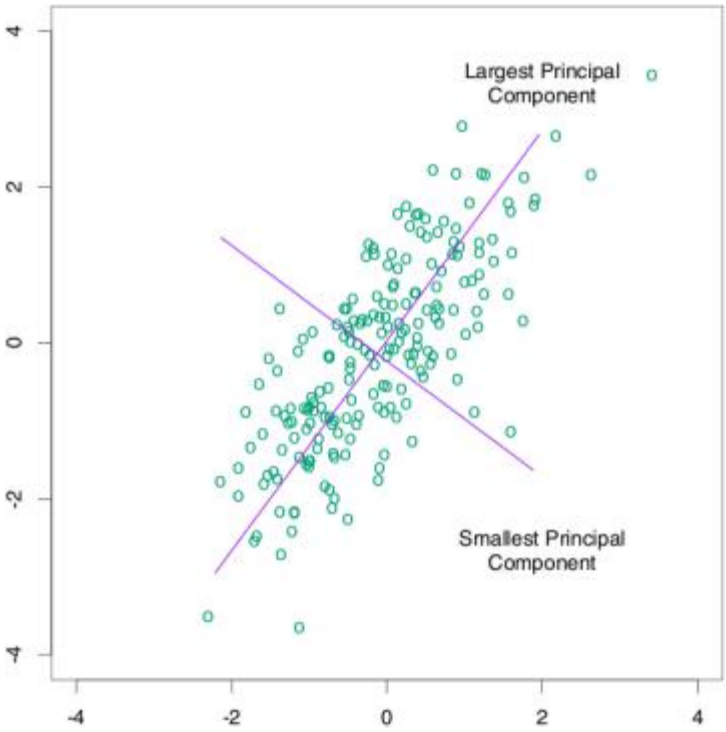
SVD 用于数据压缩

奇异值分解在数值计算是非常有用的，首先可以做矩阵的压缩


```
import numpy as np
 arr = np.array([[0, 0, 0, 2, 2],
                 [0, 0, 0, 3, 3],
                 [0, 0, 0, 1, 1],
                 [1, 1, 1, 0, 0],
                 [2, 2, 2, 0, 0],
                 [5, 5, 5, 0, 0],
```

```
                    [1, 1, 1, 0, 0]])
 # 1. 分解
u, sigma, v = np.linalg.svd(arr)
 # 2. 重构
new_arr = np.mat(u[:, 0:2]) * np.mat(np.diag(sigma[0:2])) * np.mat(v[0:2, :])
```

SVD 用于 PCA 降维



在中心化后，由于特征的均值变为 0，所以数据的协方差矩阵 C 可以用 $E\left(XX^T\right)$或者 $\frac{1}{m}XX^T$ 来表示，X^T 为矩阵的转置。这里 X 每一行为一个特征。当然也可以表示为 $E\left(X^TX\right)$或者 $\frac{1}{m}X^TX$，这时每一行为一个样本。

$$cov(X,Y)=\frac{\sum_{i=1}^{n}(X_i-\bar{X})(Y_i-\bar{Y})}{(n-1)}$$

vs.

$$var(X)=\frac{\sum_{i=1}^{n}(X_i-\bar{X})(X_i-\bar{X})}{(n-1)}$$

SVD 分解得到的三个矩阵分别称为：左奇异向量，奇异值矩阵，右奇异向量。左奇异向量用于压缩行，右奇异向量压缩列。压缩方法均是取奇异值较大的左奇异向量或者右奇异向量与原数据 C 相乘。

SVD 用于协调过滤

	鳗鱼饭	日式炸鸡排	寿司	烤牛肉	手撕猪肉
Jim	2	0	0	4	4
John	5	5	5	3	3
sally	2	4	2	1	2
Tom	1	1	1	5	5

可以直接计算 Jim 与其余三个用户的相似度，然后选最相似的样本来为 Jim 的两个空位打分。但是这样，如果一旦样本、特征过多，计算量就猛增。而事实上，我们不一定需要那么多特征，因此可以使用 SVD 分解把样本映射到低维空间。（事实上，容易能从数据中看出来映射 2 维空间，左边三个和右边两个明显不一样）

```
food = np.mat([[2, 0, 0, 4, 4], [5, 5, 5, 3, 3], [2, 4, 2, 1, 2], [1, 1, 1, 5, 4]])
u, sigma, v = np.linalg.svd(food)

simple_food = np.mat(u[:, 0:2]) * np.mat(np.diag(sigma[0:2])) * np.mat(v[0:2, :])
```

SVD 用于矩阵求逆

在矩阵求逆过程中，矩阵通过 SVD 转换到正交空间，不同得奇异值和奇异值向量代表了系统矩阵中不同的线性无关（或独立）项。对矩阵进行 SVD 分解，形式如下所示：

$$[H]_{n\times v}=[U]_{n\times n}[\Sigma]_{n\times v}[V]^T_{v\times v}=\sum_{i=1}^{n}\{U_i\}_n\sigma_i\{V_i\}^T_v$$

奇异值矩阵为：

$$[\Sigma]=\begin{bmatrix}\sigma_1 & 0 & \cdots & \cdots\\ 0 & \sigma_1 & \cdots & \cdots\\ 0 & \cdots & \cdots & \cdots\\ 0 & \cdots & \cdots & \sigma_n\end{bmatrix}$$

当用 SVD 方法进行求逆时，会使得求逆运算变得非常简单，这是因为通过 SVD 求逆，只需要对奇异值求倒数即可，而正交阵 B，有 $B^{-1}=B^T$。因此，其求逆形式为：

$$\begin{aligned}[H^{-1}]_{n\times v}&=\left([V]^T_{v\times v}\right)^{-1}\left([\Sigma]_{n\times v}\right)^{-1}\left(U_{n\times n}\right)^{-1}\\&=[V]_{v\times v}\left[\Sigma^{-1}\right]_{v\times n}[U]^T_{n\times n}\\&=\sum_{i=1}^{n}\{V_i\}_v\sigma_i^{-1}\{U_i\}^T_n\end{aligned}$$

奇异值矩阵为：

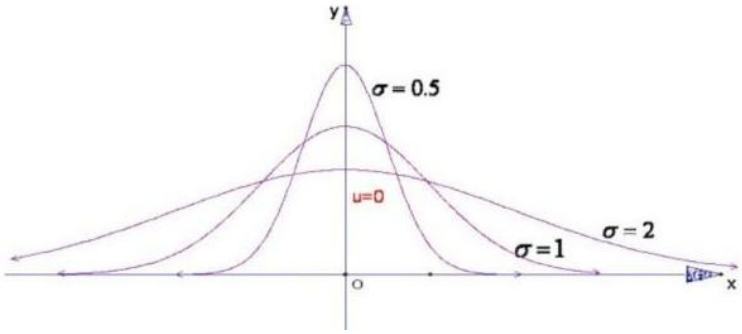
$$\left[\sum^{-1}\right]_{v\times n}=\begin{bmatrix}\sigma_1^{-1} & 0 & \cdots & 0\\ 0 & \sigma_2^{-1} & \cdots & 0\\ \cdots & \cdots & \cdots & \cdots\\ 0 & 0 & \cdots & \sigma_n^{-1}\end{bmatrix}$$

从上面可以看出，SVD 求逆是原始奇异值的倒数，这就使得通过 SVD 对矩阵求逆变得非常简单：奇异值求倒数，奇异矩阵转置。

概率论

为什么会涉及到概率论？

在机器学习和深度学习中是用的比较多的，为什么会涉及到概率论？
因为 X 以及如果是有监督机器学习的话，它的标签 y，如果我们把它当作是随机变量的话，那我们就可以用概率论的观点对它进行建模，假设它服从某种概率分布，比如说人的身高它大体是服从正太分布的，像姚明身高非常高的非常少，像郭敬明一样矮也是非常少的，比如中国的男性平均身高 1.75 左右，画出来就是我们学概率论和数理统计时候的一个正太分布这样一个东西



我们要是对数据进行分类的话，根据他的身高、体重等等，那我们就可以对他的身高进行建模来计算它服从某种分布，然后计算他的概率，这就是我们要学习概率论的原因

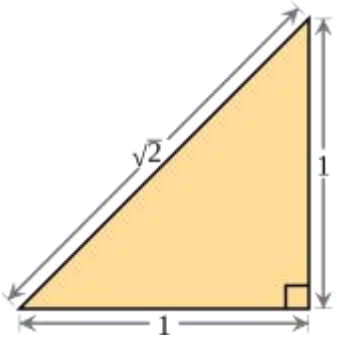
随机事件和随机事件概率

什么是随机事件呢？
就是可能发生，也可能不发生的事件
比如你抛硬币，它正面朝上，这就是一个随机事件，生孩子生男生女这也是一个随机事件
如果一定发生的话，这种称为必然事件，比如说太阳明天会升起，这肯定是必然事件
反正不可能发生的事件，我们称之为不可能事件，比如水往高处流，这就是不可能事件，因为水是往低处流的，否则它会违反自然规律的
我们一般把随机事件用大写字母 A B 这样来表示，每一个随机事件它关联有一个发生的概

率，记作 P(A)，像抛硬币它正面朝上的概率是 0.5，反面朝上的概率也是 0.5，0≤P(A)≤1 如果等于 1 那就是必然事件，如果等于 0 那就是不可能事件

以前学概率论的时候，老师交了我们各种计算概率的方法，比如抽各种颜色的球等等这样的问题，一般都是用排列组合来算的

还有另外一种是那种连续的事件，比如在第一象限扔一个石头进去，它落在什么位置，这就是连续性的事件，那么落在一半的区域里面的概率就是 1/2，这是通过面积来算的，这是另外一种套路

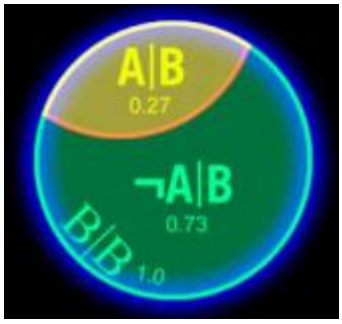


条件概率和贝叶斯公式

条件概率是针对于两个或更多个有相关关系/因果关系的随机事件而言的

对于两个随机事件 A 和 B 而言，在 A 发生的情况下 B 发生的概率，那记住 P(B|A)，AB 同时发生的概率除以 A 发生的概率

$$p(b|a)=\frac{p(a,b)}{p(a)}$$



$$p(a|b)=\frac{p(a,b)}{p(b)}$$

贝叶斯公式

$$p(a|b)=\frac{p(a)p(b|a)}{p(b)}$$

这里称 a 为因，b 是果，我们称知道原因后结果发生的概率是先验概率，贝叶斯公式得到的结果是后验概率

$$p(b)p(a|b)=p(a,b)=p(a)p(b|a)$$

贝叶斯公式在整个机器学习和深度学习中是非常有用的，因为很多时候我们要用一种叫做

最大化后验概率的思想 MAP

随机事件的独立性

说白了就是两个事情是不相关的，b 在 a 发生的条件下发生的概率是等于 b 本身发生的概率

$$p(b|a)=p(b)$$

$$p(a,b)=p(a)p(b)$$

我们可以给它推广到 n 个事件相互独立的情况上面去，就是等于各自发生概率的乘积

$$p(a_1,...,a_n)=\prod_{i=1}^{n}p(a_i)$$

随机变量

整个概率论的核心

变量是什么呢？

我们中学的时候就学习过变量了，它是取值可以变化的量，比如可以取 0 到 1 区间上所有的实数，或者取从 1 到 100 之间的整数

随机变量是什么呢？

就是变量取值都有一个概率

第一种情况是离散型的随机变量，比如前面说的抛硬币正面朝上还是反面朝上，这是两个事件，我们可以把这两个事件编号，得到

$$x:\begin{cases}=0 & 反\\ =1 & 正\end{cases}$$

这就是随机变量，X 取值有两种情况，0 或 1，取每种值的概率都是 0.5，离散型的随机变量它的取值只可能是有限个可能，或者是无穷可列个，比如是从 0 到+∞，但是一定是能用整数编号编出来

另一种是连续型的随机变量，理解起来抽象一些，它的取值是无限不可列个，比如 0 到 1 之间的所有的实数，首先它肯定是无限个，而它比无限可列个更高级，它不可列，比如其中的 0.001 到 0.002 之间它还是有无限个，不管怎么细分，a 和 b 之间还是有无限个，这就是连续型的随机变量，比如说抛石子在 0 到 1 的矩形范围内，它可能落在区域内任何一个位置，那么石子落在的位置 x,y 就是连续型随机变量，说白了就是它坐标取 0 到 1 之间任何一个值都是有可能的

对于离散型随机变量，写成如下：

$$P(x=x_i)=p_i$$

$$p_i\geq 0$$

$$\sum p_i=1$$

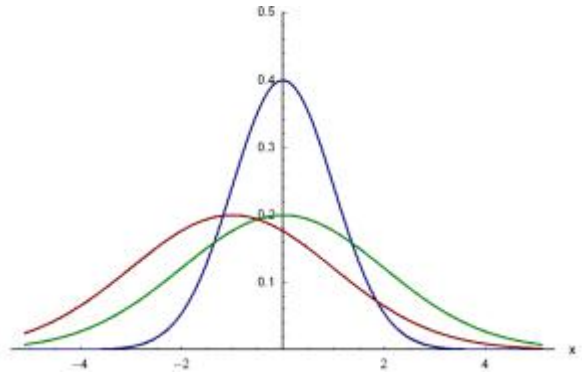
对于连续型随机变量我们是这么定义的，利用它的概率密度函数来定义

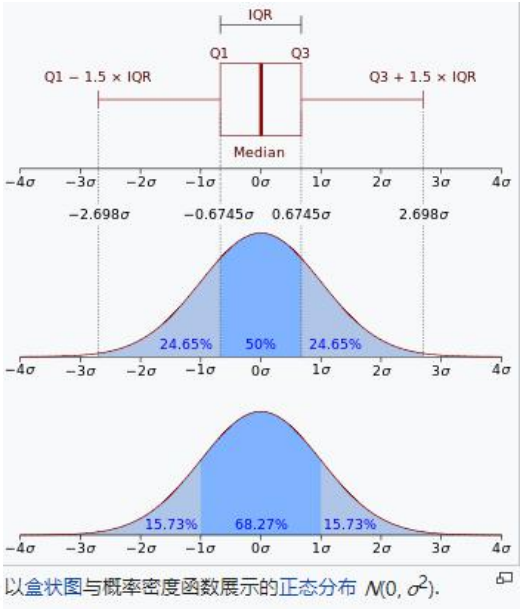
$$f(x) \geq 0$$

$$\int_{-\infty}^{+\infty} f(x)dx = 1$$

概率密度函数算出来的 f(x)可以认为是近似等同于事件发生的概率 P(x)，但是它还不是概率，但是他们之间是正相关的

$$F(y) = p(x \leq y) = \int_{-\infty}^{y} f(x)dx$$





需要注意的是，连续性随机变量它取某个具体值的概率是 0 的， $p(x = x_i) = 0$

但是它落在某个区间范围的概率是有值的，因为算的是面积，就好像前面提到的石子落到区域内的面积

$$\int_{x_1}^{x_2} f(x)dx = F(x_2) - F(x_1)$$

数学期望和方差

这个在学概率论的时候同学们都是学过的，这是核心概念之一

什么是数学期望，从均值开始看起

$$n: x_1 \quad x_2 \quad \dots \quad x_n : \frac{1}{n}\sum_{i=1}^{n} x_i \rightarrow \sum_{i=1}^{n} \frac{1}{n} x_i \rightarrow \sum_{i=1}^{n} p_i x_i$$

这里的 1/n 可以看作是每个 Xi 的权重，或者叫概率，如果把它替换称概率 P，就得到了我们的数学期望

举个例子，比如说买彩票有 0.2 的概率中 500 万，0.3 的概率中 1 万，0.5 的概率中 0.1 万，那可能的收益不可能用 500+1+0.1 来平均一下，肯定要考虑概率值

$$500 \times 0.2 + 1 \times 0.3 + 0.1 \times 0.5$$

说白了，数学期望就是概率意义的平均值，这是对于离散型的随机变量而言的

对于连续型的随机变量，把它推广一下变成定积分，求一个广义积分就是数学期望

$$E(x)=\sum x_i p(x_i)$$

$$E(x)=\int_{-\infty}^{+\infty} xf(x)dx$$

方差

反应的数据的波动程度的，就是它和均值，我们的数学期望偏离程度的平均

这里每个数据减去期望的平方，不平方的话正负抵消掉了，然后再乘以 P 概率值

$$D(x)=\sum (x_i-E(x))^2 p(x_i)$$

$$D(x)=\int_{-\infty}^{+\infty} (x-E(x))^2 f(x)dx$$

这里（x-E(x)）的平方等于如下

利用数学期望的线性性质：

$$E(a+bX)=a+bEX$$

$$E(X-\mu)^2=E(X^2-2\mu X+\mu^2)=EX^2-2\mu EX+\mu^2$$

$$\mu=EX$$

$$VarX=E(X^2)-(E(X))^2$$

这是求方差时非常常用的一个等式

常用随机变量服从的分布

正太分布

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

均匀分布

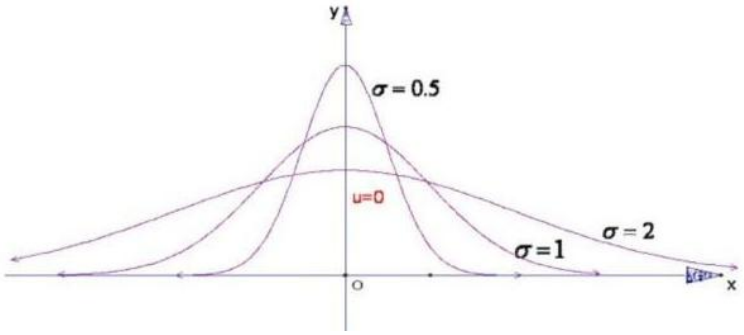
$$f(x)=\begin{cases}\frac{1}{b-a} & a\le x\le b\\ 0 & x<a,x>b\end{cases}$$

二项分布

$$p(x=1)=p$$

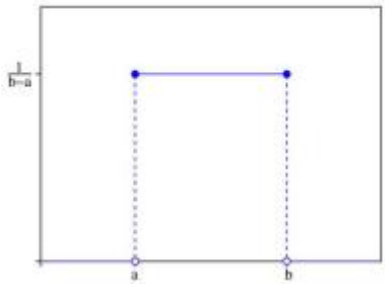
$$p(x=0)=1-p$$

自然界和生活中很多事件都服从正太分布的，或者近似的服从正太分布的，比如人的身高和体重，大部分人都是平均值，小部分人比较胖，或比较瘦，比较高，或比较矮，还有考试成绩啊，人的收入啊这些都近似服从正太分布



均匀分布，很好理解，a 和 b 这个区间内取任意一个值的概率相等，

rand()



二项分布拿我们抛硬币的例子来说，比如 x=0 是背面朝下的概率，x=1 是正面朝上的概率，那么它取值只有 0 或 1 两种情况，当然不用 0 或 1，你用-1 和+1 也是可以的。都是二项分布，取每个值都有一个概率值。在我们机器学习中，主要用的就是这几种概率分布。

随机向量

线性代数中，我们把标量 x 推广到向量，就是它有多个分量。

同样我们把单个随机变量可以推广到随机向量，就是它有多个分量，这样就有了随机向量的概念了，这是很自然的延申。

离散型的随机向量

向量 X 取某一个具体的值为向量 Xi，然后取每一个向量值的概率都大于等于 0，所有的概率加起来要等于 1，符合这两个约束条件就可以了

$p(x = x_i)$

连续型的随机向量

它是用 0 和概率密度函数来描述的，n 重积分等于 1，相当于体积等于 1

$f(x) \geq 0$

$\iiint f(x)dx = 1$

下面是二维的随机向量

$f(x_1, x_2) \geq 0$

$\int_{-\infty}^{+\infty} f(x_1, x_2)dx_1 dx_2 = 1$

随机变量的独立性

两个随机变量如果相互独立的话，它们的联合概率密度函数等于它们的分别的概率密度函数乘积

推广到多个随机变量相互独立

$$f(x_1,x_2,..,x_n)=f(x_1)f(x_2)...f(x_n)$$

这和随机事件的形式上是统一的，f 换成符合 P 就可以了

协方差

协方差是对于方差的推广，对于两个随机变量，它们的协方差是反应它们两个之间的线性相关程度的，把 x2 换成 x1 那就是方差了，展开之后就是 x1 和 x2 的期望等于它们期望的乘积

$$\operatorname{cov}(x_1,x_2)=E((x_1-E(x_1))(x_2-E(x_2)))$$

$$\operatorname{cov}(x_1,x_2)=E(x_1x_2)-E(x_1)E(x_2)$$

对于 n 维的向量 X，它的协方差就构成了一个协方差矩阵，第一个是 x1 和 x1 的协方差，第一行第二个是 x1 和 x2 的协方差，第一行第 n 个是 x1 和 xn 的协方差

$$\begin{bmatrix} x_1x_1 & x_1x_2 & \cdots & x_1x_n \\ x_2x_1 & \ddots & \cdots & \\ \vdots & \vdots & \ddots & \vdots \\ x_nx_1 & & & x_nx_n \end{bmatrix}$$

显然这是一个对称阵，这在我们机器学习里面会经常使用的

随机向量的常见分布

和随机变量一样，我们的随机向量也服从一些典型的分布，比如说正太分布和均匀分布，正太分布是对于一元的随机向量的一个推广

$$f(x)=\frac{1}{(2\pi)^{\frac{n}{2}}\left|\sum\right|^{\frac{1}{2}}}e^{(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu))} \qquad x\in R^n$$

Σ是协方差矩阵，然后不是绝对值，是行列式的开根号，μ是均值向量，然后Σ的逆矩阵

这时我们再去理解一元的正太分布概率密度函数就好理解了，形式是一致的

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

$$X^TAX=aX^2$$

以上就是多维的正太分布

最大似然估计

它是解决估计一个概率密度函数中参数问题的

比如有个向量 X，θ是它的参数，比如正太分布中的μ和σ这都是需要估计的参数

$p(x;\theta)$

那我们怎么估计这组参数呢？肯定是根据一组样本来学习，假设我们有 l 个样本它们是独立同分布的，也就是说它们服从同样一个概率分布，并且它们之间相互独立的，抽样出来的

$x_i, i = 1,...,l$

那么所有变量发生的概率就可以写成它们乘积的形式，因为它们之间是相互独立的嘛，这时 L 是似然函数，这里 x 一个个的已经取了具体样本的值了，然后θ是我们要估计的参数

$$L(\theta) = \prod_{i=1}^{l} p(x_i;\theta)$$

既然这组样本是已经抽样抽出来的，是已经发生的，我们肯定要把它发生的概率最大化，也就是说要最大化这样一个似然函数

$$\max \prod_{i=1}^{l} p(x_i;\theta)$$

前面我们说了，求解一个函数的极值，就是要求解它的导数，也就是梯度等于 0

$$\nabla L(\theta) = 0$$

而这样的乘积形式求导是不容易的，因为前面讲过公式

$$(fg)' = f'g + fg'$$

如果更多项展开是非常麻烦的，所以我们可以对函数取个对数，因为对数函数是单调增函数，所以求原函数的极值，也等于求它的对数形式的极值，所以我们两边取对数的话，就把连乘的形式转化为了连加的形式，因为连加的形式的导数，就是等于导数的连加

$$\ln ab = \ln a + \ln b$$

$$\ln L(\theta) = \ln \prod_{i=1}^{l} p(x_i;\theta) = \sum_{i=1}^{l} \ln p(x_i;\theta)$$

所以我们要解决的问题就是求这个函数的极大值，这个可以对θ求导让它等于 0 得到，带 log 的是对数似然函数，这就是最大似然函数最基本的思想

$$\max \sum_{i=1}^{l} \ln p(x_i;\theta)$$

最优化

基本概念

最优化问题就是求 f(x)的极大值或者极小值

$f(x) \quad x \in R^n$

我们往往求极小值，在这里 x 是优化变量，就是自变量，f(x)称为目标函数，可能还带有约束条件，有些优化问题既有等式约束，又有不等式约束

$$\max f(x) \Leftrightarrow \min f(x)$$

$$g_i(x) = 0 \quad i = 1..m$$

$$h_j(x) \le 0 \quad j = 1..n$$

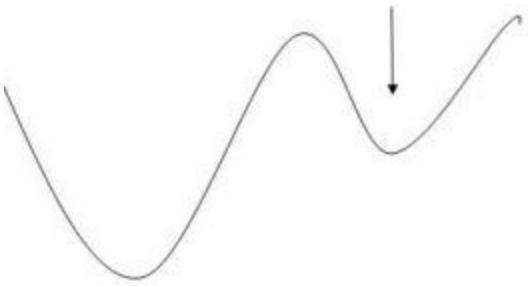
满足这种约束条件，并且还在 f(x)定义域之内的那些 X 构成的集合叫可行域 D

局部极小值

任何在 X0 的邻域存在

$$f(x) \ge f(x_0)$$

$$\forall x \in \delta(x_0)$$



全局极小值

其实初中我们就学求极值的问题了，比如二次函数抛物线的顶点就是极大值或者极小值，高中我们也会有求极值的题目，算一个非常复杂的函数的极值，借用初等数学一些的技巧来完成的。而真正的飞跃是发生在大学里面，微积分这门课为求解极值提供了一个统一的方法，就是找它导数等于 0 的点，如果是多元函数的话，我们说它的梯度等于 0，我们去解方程组就可以了，在机器学习里面我们遇到的几乎所有的函数都是可导的。光滑即可导

$$f'(x) = 0$$

局部最小值（local minimum）， 通过大量实践发现在高维度的优化问题中，局部最小值和全局最小值通常没有太大的区别，甚至在有些情况下比局部最小值有更好的归纳能力（泛化能力）。

为什么要迭代求解

前面咱们说了求极值就是导数或梯度等于 0，找到疑似的极值，就是驻点

$$\left.\begin{array}{l} f'(x) = 0 \\ \nabla f(X) = 0 \end{array}\right\} \text{驻点}$$

有了微积分里面这样的手段，只要函数可导，我们求解不就行了？

因为很多时候去求方程等于 0 或方程组等于 0 的根并不是那么容易，比如

$$f(x,y) = x^3 - 2x^2 + e^{xy} - y^3 + 10y^2 + 100\sin(xy)$$

分别求对 X 和 Y 的偏导数

$$\begin{cases} 3x^2 - 4x + ye^{xy} + 100y\cos(xy) = 0 \\ xe^{xy} - 3y^2 + 20y + x\cos(xy) = 0 \end{cases}$$

对于求下面的一些函数的方程叫做超越方程，别说超越方程了，对于 5 次或以上的代数方程它都没有求根公式了，所以这个思路看上去很美，但是实际上是行不通的，我们得想其它的办法

$e^x \quad \ln x \quad \sin x \quad arc\sin x$

近似求解，我们先给个初始的值，看看它是不是极值点，当然得满足等式或者不等式约束，它如果不是，我们想办法再前进一步，只要我们得 Xn 它越来越接近我们的极值点，那这个问题不就解决了嘛

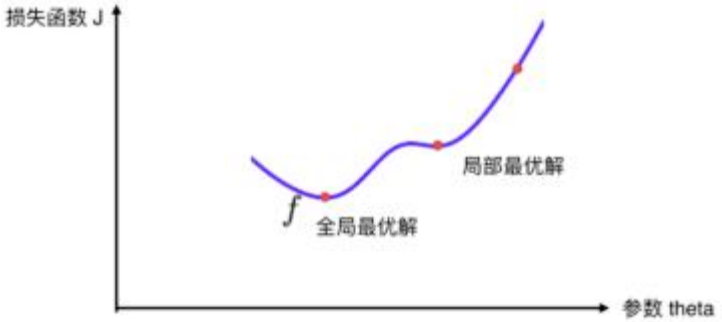
$x_0 \to x_1 \to x_2 \to \ldots \to x_n$

我们要构造一个序列，满足这样一个极限，当 k 趋近于 + ∞的时候，f(Xk)这点的梯度值极限趋近于 0 的话，那我们就找到了函数的驻点

$\lim_{k\to+\infty} \nabla f(x_k) = 0$

由此我们要解决的核心问题就变成了，怎么从当前一个点移动到下一个点上面去，也就是怎么从 x_k 到 x_{k+1} ，迭代法是我们计算数学中经常采用的一种方法，它不单单可以求极值，还可以用来求方程组的解，包括线性方程，非线性方程，一个矩阵的特征值等等

$x_{k+1} = h(x_k)$



$\nabla f(X) = 0$

梯度下降法

数值优化算法，是求近似解，而不是理论上面的精确解

$x_{k+1} = x_k - \gamma\nabla f(x_k)$

下面我们来推导这个公式，需要用到多元函数的泰勒展开公式，最后加上一个高阶无穷小

$f(x) = f(x_0) + [\nabla f(x_0)]^T (x - x_0) + o(x - x_0)$

如果我们忽略高阶无穷小，也就是说 x 它是属于 x0 的δ邻域里面的话，那么等号就会变成约等于，那么我们把 f(x0)放到等号左边

$f(x) - f(x_0) \approx [\nabla f(x_0)]^T (x - x_0)$

$X^T Y = \|X\| \cdot \|Y\| \cdot \cos\theta$

≥ 0

这时要想下降的更快，就要让等式右边得到最小值，即当 cosθ=-1，下降的幅度最大！

单位向量代表的是方向，长度为 1 的向量就是模为 1 的向量称为单位向量。

$$\left\|\vec{v}\right\| = \sqrt{v_1^2 + v_2^2 + \ldots + v_n^2}$$

向量乘以标量不会改变它的方向，只会改变它的幅值。这就是缩放向量的方法。

所以一个向量可以表示为一个标量乘以一个单位向量。

为了使得下降幅度最大，那么向量(x-x0)的方向和梯度的方向相反：

$$v = -\frac{\nabla f(\theta_0)}{\left\|\nabla f(\theta_0)\right\|}$$

$$\theta = \theta_0 - \eta \frac{\nabla f(\theta_0)}{\left\|\nabla f(\theta_0)\right\|}$$

分母是个标量，可以并入到η中，简化为：

$$\theta = \theta_0 - \eta \nabla f(\theta_0)$$

需要注意的是，x 和 x0 离的充分近，也就是在它的δ邻域里面，才能忽略掉泰勒展开里面的一次以上的项，否则就不能忽略它了，它就不是高阶无穷小了，约等于的条件就不成立了，所以η步长不能够太大，由此我们就得到了梯度下降法的公式。

$$x = x_0$$

$$for(\ldots)$$

$$x_{k+1} = x_k - \gamma \nabla f(x_k)$$

实现细节问题

初始值的设定，随机或者设定满足约束条件的就 OKAY

步长的设定，一个充分接近于 0 的数就可以了，也可以动态的调整

循环终止的判定，max_iteration，上一次减下一次的 f(x)

牛顿法

$$x_{k+1} = x_k - H_k^{-1} g_k$$

这个方法是用伟大的物理学家和数学家命名的，莱布尼茨和牛顿都被普遍认为是独立的微积分发明者。牛顿最先将微积分应用到普通物理当中，而莱布尼茨创作了不少今天在微积分所使用的符号。牛顿利用了微积分的技巧，由万有引力及运动定律出发说明了他的宇宙体系，解决天体运动，流体旋转的表面，地球的扁率，摆线上重物的运动等问题。牛顿在解决数学

物理问题时，使用了其独特的符号来进行计算，并提出了乘积法则、链式法则、高阶导数、泰勒级数。

它是基于这样一个思想，我们不是要找梯度等于 0 的点嘛，迭代 x 让梯度值尽可能等于 0

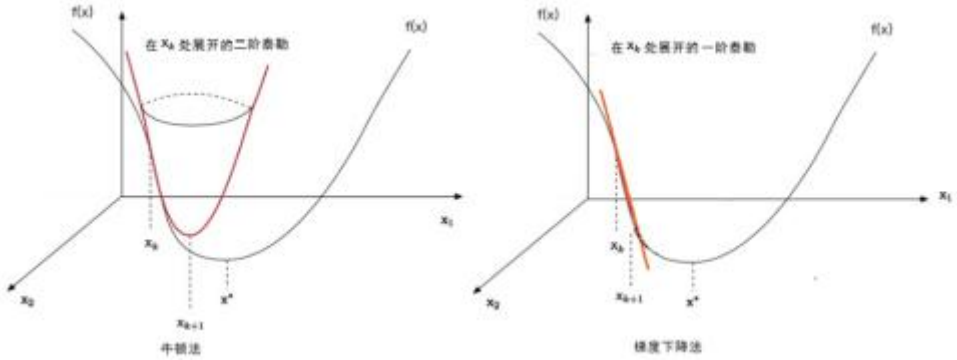
$$\nabla f(X)=0$$

$$x_0 \to x_1 \to x_2 \to \ldots \to x_n$$

推导牛顿法的公式，多元函数的泰勒展开公式，这里我们把它展开成二次以上的项

$$f(x)=f(x_0)+[\nabla f(x_0)]^T(x-x_0)+\frac{1}{2}(x-x_0)^T H(x_0)(x-x_0)+o(x-x_0)$$

同样当 x 在 x0 的δ邻域里面的话，我们可以把后面的高阶无穷小忽略掉，从而建立一个近似的等式，你现在不是要找梯度等于 0 的 x 嘛，我们现在用二次函数来代替原来的函数 f(x)，在 x0 处用一个抛物线来近似它，但是离 x0 越远误差会越大，离 x0 越近误差会越小



如果是二次函数的话，我们找它梯度等于 0 的点是非常好找的，基于约等于的式子，两边同时求导，取梯度的话，就是下面

$$f(x)=f(x_0)+[\nabla f(x_0)]^T(x-x_0)+\frac{1}{2}(x-x_0)^T H(x_0)(x-x_0)+o(x-x_0)$$

$$f(x)\approx f(x_0)+[\nabla f(x_0)]^T(x-x_0)+\frac{1}{2}(x-x_0)^T H(x_0)(x-x_0)$$

因为，

$$(w^T x)'=w$$

$$(x^T Ax)'=(A+A^T)x$$

所以，

$$\nabla f(x)\approx \nabla f(x_0)+H(x_0)(x-x_0)=0$$

下面就可以解一次方程，如果 hessian 矩阵可逆，就可以两边乘上 H 的逆矩阵

$$g+H(x-x_0)=0$$

$$x-x_0=-H^{-1}g$$

前提是可逆的，同学们对比一下和梯度下降法的公式，注意一下 H 和 g 都是 x=x0 点处取得的

$$x_{k+1}=x_k-\eta\cdot g_k$$

有同学肯定会说，那我们去求解方程组，那不是一次就可以求得梯度等于 0 的点 x 了嘛，但是因为我们忽略了后面的高阶无穷小，所以我们只是近似的找到了梯度等于 0 的点，但是它并没有真正找到，所以我们下次要继续去用这个公式去迭代。

$x_{k+1} = x_k - \eta \cdot H_k^{-1} \cdot g_k$

实现细节问题

初始值的设定，随机或者设定满足约束条件的就 OKAY

步长的设定，η步长的设定和梯度下降法不同，不是迭代就一定使得 f(x)下降，所以设置不好就有可能不收敛，求不到最优解的值，牛顿法一般会用 line search 的技术，就是选择一些10^{-4} 10^{-6} 等等，看看哪个步长使得 $f(x_{k+1})$ 更小就反过来选择哪个作为步长

$x_{k+1} = x_k - \eta \cdot H_k^{-1} \cdot g_k$

循环终止的判定，max_iteration，上一次减下一次的 f(x)

如果我们的 x 是二次函数的话，牛顿法一步就可以收敛，前提是步长不做设置，等于 1 的时候就可以了，这是因为如果原函数 f(x)是二次函数的话，泰勒展开里面就不是约等于，而是直接等于

在工程里面我们不是直接求解 hessian 矩阵的逆矩阵，而是求一个方程组 AX=b，可以用共轭梯度法

梯度下降法只用到了一阶导数的信息，牛顿法既用到了一阶导数的信息，也用到了二阶导数的信息。梯度下降法是用线性函数来代替目标函数，牛顿法是用二次函数来代替目标函数，所以说牛顿法的收敛速度是更快的。

但是牛顿法也有它的局限，hessian 矩阵不一定可逆，第二个即使存在，我们来解一个线性方程组，当 hessian 矩阵规模很大，解线性方程组是非常耗时的，因此出现了一种改进的算法叫拟牛顿法

坐标下降法

分治法的思想

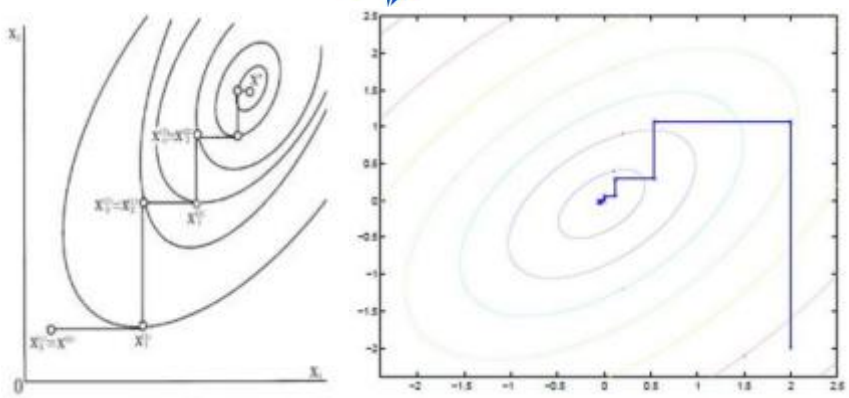
$\min f(x), x = (x_1, x_2, \ldots, x_n)$

它的思想是按住其它的不动，只优化其中一个比如 x_1，那就把多元函数求极值问题变成了一元函数求极值问题，这样优化的难度就小了很多，紧接着我们把其它的按住不动，再优化 x_2，一直到 x_n，完了之后再回来优化 x_1，至于一个变量怎么解，可以用牛顿法或者公式来求解了，看具体问题的形式了

SVM 中的 SMO 算法就是把其中两个拿出来优化，其它的固定不动

liblinear 库也大量的使用坐标下降法来求解的

直观的看就好比



我们把所有的从 x_1 x_2 到 x_n 优化一遍，就好比梯度下降迭代一次，这种方法它也有很大好处，就是计算的工作量小很多

数值优化算法面临的问题

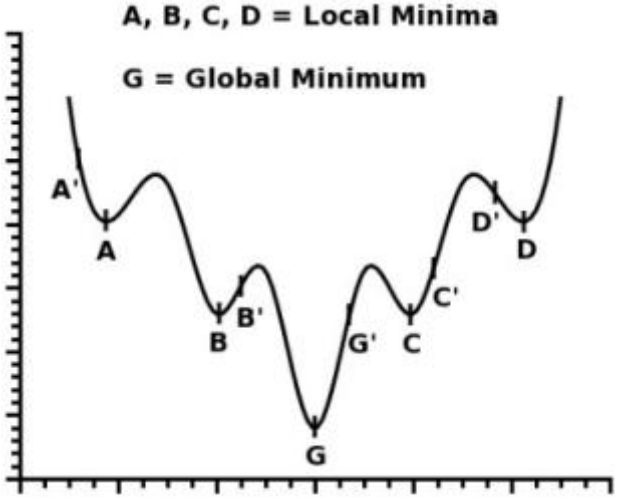
前面梯度下降法，还有牛顿法求极值的依据都是 $\nabla f(x_k)=0$ ，而前面我们也说了函数在某一点的导数或梯度等于 0 就是驻点，它只是函数取得极值的必要条件，不是充分条件，也就是说无法推导出

$$\nabla f(x_k)=0 \Rightarrow x_k \min$$

所有我们前面介绍的数值优化算法，都会面临两个问题

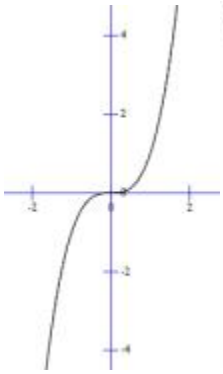
局部极值问题

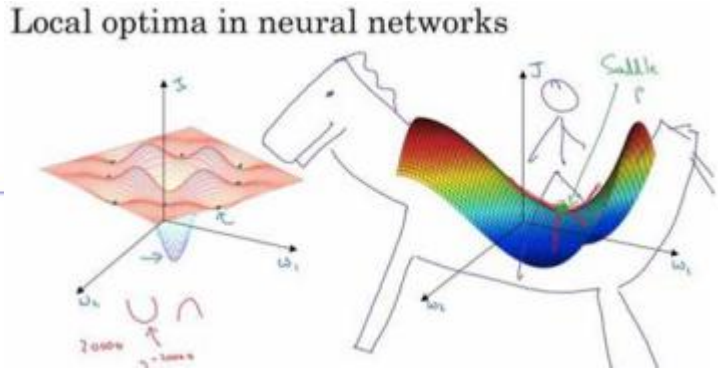
这个问题好歹是个局部极值，只不过不是全局极值，理论上我们要求全局最优解的话，我们要把所有极小值找出来，你可能要不断的去设置不同的初始迭代点，反复的求解让它收敛到不同的局部最小值，然后来比较谁最小来找到一个全局最小值。



鞍点问题

前面我们说过一元函数 x^3 函数，在坐标为 0 处它是驻点，但是它连局部最小值点都不是，对应多元函数来说，我们称之为鞍点，严格定义是，在这一点它的 hessian 矩阵是不定的，既不正定也不负定，这样它就即不是极小值的点也不是极大值的点。





拉格朗日乘数法

高等数学和微积分的时候我相信大家都学过，用来求解等式约束下的极值问题的

$\min f(x)$

$h_i(x)=0, i=1,..,k$

拉格朗日乘子函数，把对 x 带约束条件的优化问题，转化为不带约束条件的优化问题

$$L(x,\alpha)=f(x)+\sum_{i=1}^{k}\alpha_i h_i(x)$$

下面两个式子是分布对 x 和α求梯度，然后去解下面方程组就可以了

$$\nabla_x f+\sum_{i=1}^{k}\alpha_i\nabla_x h_i=0$$

$$h_i(x)=0$$

凸优化问题

前面我们说过数值优化面临两个问题，一个是局部极值问题，和鞍点问题，我们能不能避免这两个问题呢？

只要我们对优化问题进行限定就可以，这类问题有两个限定条件

1，优化变量的可行域必须是凸集

2，优化函数必须是个凸函数

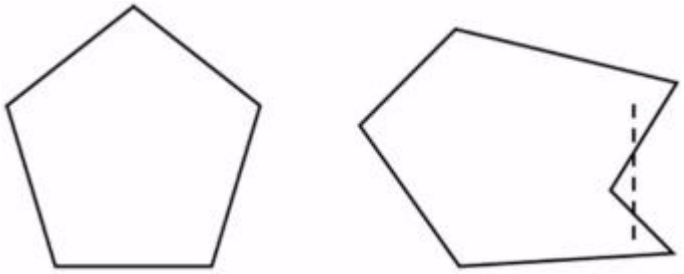
同时满足这两个限定条件的问题，叫做凸优化问题，从而才能说局部极小值就是全局极小值

凸集

凸集的定义：对于一个点的集合 C，有 x,y 它都是属于 C 里面的两个点，它们两点的连线中任何一点也是属于集合 C 的，

$$\theta x+(1-\theta)y\in C$$

$$0\le\theta\le 1$$



典型的凸集

欧式空间 R^n

$x, y \in R^n \Rightarrow \theta x + (1-\theta) y \in R^n$

它的意义在于，很多时候可行域就是欧式空间，那肯定是凸集

仿射子空间 $\{x \in R^n : Ax = b\}$

x 是 n 维欧式空间里的向量，满足 Ax=b 这个方程组的解，它构成的集合就是仿射子空间

$x, y \in C$

$$\begin{aligned} A(\theta x + (1-\theta) y) &= \theta Ax + (1-\theta) Ay \\ &= \theta b + (1-\theta) b \\ &= b \end{aligned}$$

它的意义在于，所有的等式约束构成的集合是凸集，很多时候等式约束就是这样的线性方程，所以一般我们不会构建处非线性的等式约束来。

多面体 $\{x \in R^n : Ax \le b\}$

线性不等式的解

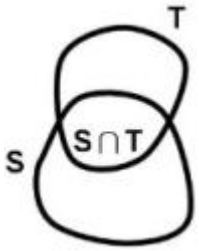
$x, y \in C$

$$\begin{aligned} A(\theta x + (1-\theta) y) &= \theta Ax + (1-\theta) Ay \\ &\le \theta b + (1-\theta) b \\ &\le b \end{aligned}$$

这样中间的点它还是不等式的解，所以说这样一个集合它还是凸集，它的现实意义在于，如果有一组线性不等式约束的话，那它构成的可行域也一定是凸集。

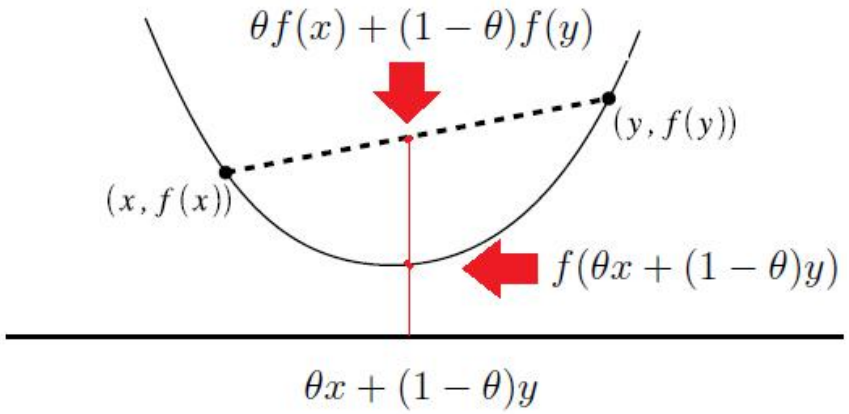
凸集的交集也是凸集 $\bigcap_{i=1}^{k} C_i$

如果有 k 个集合 $C_1\ C_2 .. C_k$，它们是凸集，那它们的交集也一定是凸集，它们的并集不是凸集，会有凹的地方了



凸函数

$f(\theta x+(1-\theta)y)<\theta f(x)+(1-\theta)f(y)$



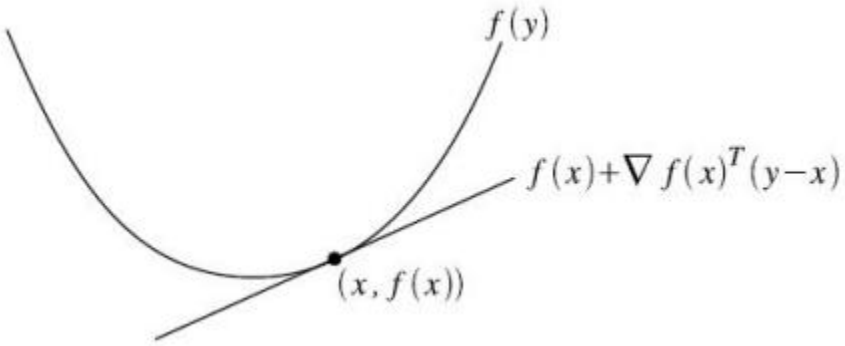
凸函数在函数上任意找两点它们的连续就是割线上的值比对应的 f(x)的值要大

证明函数是凸函数，第一种当然是利用上面定义去证明，也可以利用一阶导数信息，和二阶导数信息去证明。

先说对于一元函数来说，怎么用一阶导数来判断是凸函数还是不是凸函数呢？

$f(y)\geq f(x)+f^{'}(x)(y-x)$ x 是切线和抛物线相交的点，y 是变化的

多元函数 $f(y)\geq f(x)+\nabla f(x)^{T}(y-x)$



二阶判别法

如果一元函数，那么 $f^{''}(x)\geq 0$ 就是凸函数，多元函数 Hessian 矩阵是半正定的，它就是凸函数，>0 就是严格的凸函数，=0 就是有平的地方，就是有线性变化的地方

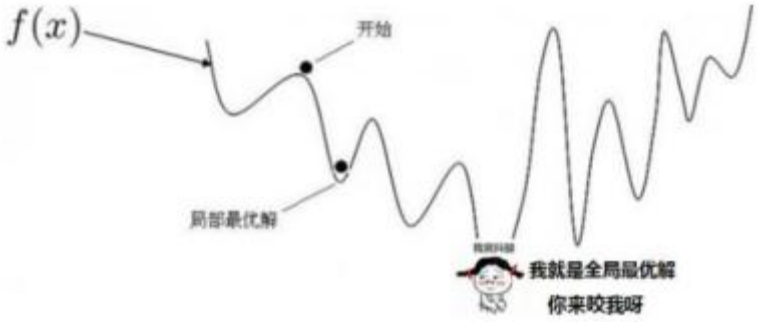
比如 ax+b 求二阶的时候就是=0.

如果每个函数 $f_i(x)$ 都是凸函数，那么它们的非负线性组合 $f(x)=\sum_{i=1}^{k}w_i f_i(x)$ if $w_i\geq 0$ 也是凸函数

凸优化的性质

凸优化的定义是目标函数是凸函数，可行域是个凸集，如果有这两个限定条件的话，局部最

优解一定是全局最优解，下面证明对于凸优化问题，它的局部最优解一定是全局最优解：
反证法，假设有一点 x 是局部最小值，但不是全局最小值，那样就存在另外一个点 y 是全局最小值位置，这时 f(y)<f(x)



如果可以证明 x 的邻域有个点 z 是比 x 的值还小的，f(z)<f(x)，那么 x 肯定不是最优解了

$z = \theta y + (1-\theta)x$

$\theta = \frac{\delta}{2\|x-y\|_2}$ δ是邻域半径$\|x-y\|_2$是 L2 范数，表示 x 和 y 两点之间的距离，把θ带进来

证明 z 是在 x 邻域的，也就是小于等于δ的

$$\|x-z\|_2 = \left\| x - \left(\frac{\delta}{2\|x-y\|_2} y + \left(1 - \frac{\delta}{2\|x-y\|_2}\right)x \right) \right\|_2$$

$$= \left\| \frac{\delta}{2\|x-y\|_2}(x-y) \right\|_2$$

$$= \frac{\delta}{2} < \delta$$

根据凸函数的性质，同时 f(y)<f(x)，θ>0

$f(z) = f(\theta y + (1-\theta)x) \le \theta f(y) + (1-\theta) f(x) = \theta(f(y) - f(x)) + f(x) < f(x)$

所以这和前面的假设是矛盾的，所以说前面的性质是成立的

凸优化一般的表述形式

$\min f(x)$
$x \in C$ C 是凸集

$\min f(x)$
$c_i(x) \le 0, i = 1,..,m$ 由不等式等式约束条件构成的可行域也是凸集
$h_j(x) = 0, j = 1,..,k$

往往我们需要去证明的一些机器学习算法它们都是凸优化问题，比如逻辑回归，SVM，线性回归等，证明它的可行域是凸集，目标函数是凸函数就可以了，凸优化规避了局部最小值问题，而且也规避了鞍点问题，f(x)是凸函数，它的 hessian 矩阵是半正定的，当是半正定的也就规避了鞍点问题

拉格朗日对偶

凸优化，非线性规划问题，甚至是运筹学里面的线性规划问题都会涉及这个概念，它的意义是把原始问题转化为另外一个问题来求解，但是转化之后的问题要容易求解一点，
拉格朗日乘数法的扩展，用来解决既带等式约束条件，又带不等式约束条件的一种方法
通过把原问题转换为对偶问题来求解，很多时候对偶问题比原问题更容易求解

$$\min f(x)$$
$$g_i(x) \le 0 \quad i = 1,..,m$$
$$h_j(x) = 0 \quad j = 1,..,k$$

构建一个广义的拉格朗日函数，所谓广义就是还包括不等式约束条件
在下面式子中你会发现对 x 的约束没有了，虽然有个对α的约束

$$L(x,\alpha,\beta) = f(x) + \sum_{i=1}^{m} \alpha_i g_i(x) + \sum_{j}^{k} \beta_j h_j(x)$$

原始问题

$$p^* = \min_x \max_{\alpha,\beta,\alpha_i \ge 0} L(x,\alpha,\beta)$$
$$= \min_x \theta_p(x)$$

原始问题等价于我们要求解的问题！

KKT 条件

拉格朗日乘数法的扩展，用来解决既带等式约束条件，又带不等式约束条件的一种理论结果

$$\min f(x)$$
$$g_i(x) \le 0 \quad i = 1,...,q$$
$$h_j(x) = 0 \quad j = 1,...,p$$
$$L(x,\lambda,\mu) = f(x) + \sum_{j}^{p} \lambda_j h_j(x) + \sum_{i=1}^{q} \mu_i g_i(x)$$

$$\nabla_x L(x^*) = 0$$
$$\mu_i \ge 0$$
$$\mu_i g_i(x^*) = 0$$
$$h_j(x^*) = 0$$
$$g_i(x^*) \le 0$$