1. 数学分析基础知识

1.1 极限知识：两个重要极限

$$\lim_{x \to 0} \frac{\sin x}{x} = 1$$

$$\lim_{x \to \infty} \left(1 + \frac{1}{x}\right)^2 = e^x$$

1.2 导数

1.2.1 导数的定义

导数，也就是微分。微分是什么？最常用的两种是：

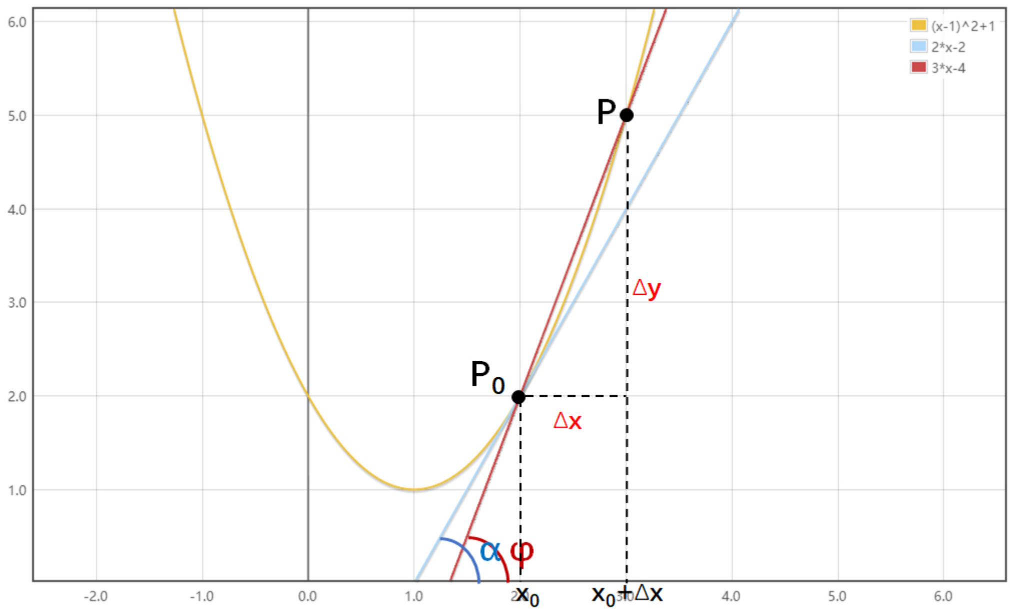
- 函数图像中，某点的切线的斜率；
- 函数的变化率；

虽然导数有很多含义，在物理场景中可能代表了瞬时速度，在函数曲线上可以代表切线的斜率，而归根到底，导数是函数增量Δy与自变量增量 Δx之比$\frac{\Delta y}{\Delta x}$的极限，这个增量比称之为函数关于自变量的平均变化率，而导数$f'(x_0)$则为 f 在 x_0 处关于 x 的变化率。公式如下：

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

在一元函数上，对应的图像如下所示：

图中，示例函数为：（$x-1$）$^2+1$，点 P（x,y）、P_0（$2,2$）为一元函数上的两点，从点 P_0 至函数任一点 $P(x,y)$，函数的增量为 Δy，自变量的增量为 Δx，增量之比为 $\frac{\Delta y}{\Delta x}$，函数在 P_0 点的导数 $f'(x_0)$ 即为函数在该点增量比的极限 $\lim_{\Delta x \to 0} \frac{\Delta y}{\Delta x}$，对应的导数为在该店切线的斜率 2，对应的导函数 $2x-2$。



常见的求导结果：

$$\frac{d\left(x^2\right)}{dx}=2x$$

$$\frac{d\left(-2y^5\right)}{dy}=-10y^4$$

$$\frac{d(5-\theta)^2}{d\theta}=-2(5-\theta)$$

不过由于是一元函数，因此也就只有一个方向变动。接下来，我们在1.2.3部分的偏导数中来讨论二元函数上不同方向的变化。

1.2.2 导数的基本计算

1. 基本求导法则

（1）$(u \pm v)'=u' \pm v'$

（2）$(uv)'=u'v+uv', (cu)'=cu'$（c为常数）

（3）$\left(\frac{u}{v}\right)'=\frac{u'v-uv'}{v^2}, \left(\frac{1}{v}\right)'=-\frac{v'}{v^2}$

（4）反函数导数：$\frac{dy}{dx}=\frac{1}{\frac{dx}{dy}}$

（5）复合函数导数：$\frac{dy}{dx}=\frac{dy}{du}\frac{du}{dx}$

2. 常用函数的导数

（1）$(c)'=0$（c为常数）

（2）$x^a = ax^{a-1}$（a为任意实数）

（3）$(\sin x)' = \cos x, (\cos x)' = -\sin x$

（4）$(\tan x)' = \sec^2 x, (\cot x)' = -\csc^2 x$

$(\sec x)' = \sec x \tan x, (\csc x) = -\csc^2 x$

（5）$(a^x)' = a^x \ln a, (e^x)' = e^x$

（6）$(\log_a^x) = \frac{1}{x \ln a}, (\ln x)' = \frac{1}{x}$

3. 一些应用

（1）问题：$f(x) = x^x$，求解$f'(x)$

解：令$t = x^x$，有$\ln t = x \ln x$

上式两边求导有：

$\frac{1}{t}t' = \ln x + 1t' = x^x(\ln x + 1)$

1.2.3 偏导数及方向导数

接下来，我们进入到二元函数领域，进一步介绍偏导数。

前面的例子都是单变量的微分，当一个函数有多个变量的时候，就有了多变量的微分，即分别对每个变量进行求微分。

$\frac{\partial}{\partial x}\left(x^2 y^2\right) = 2xy^2$

$\frac{\partial}{\partial y}\left(-2y^5 + z^2\right) = -10y^4$

$\frac{\partial}{\partial \theta_2}(5\theta_1 + 2\theta_2 - 12\theta_3) = 2$

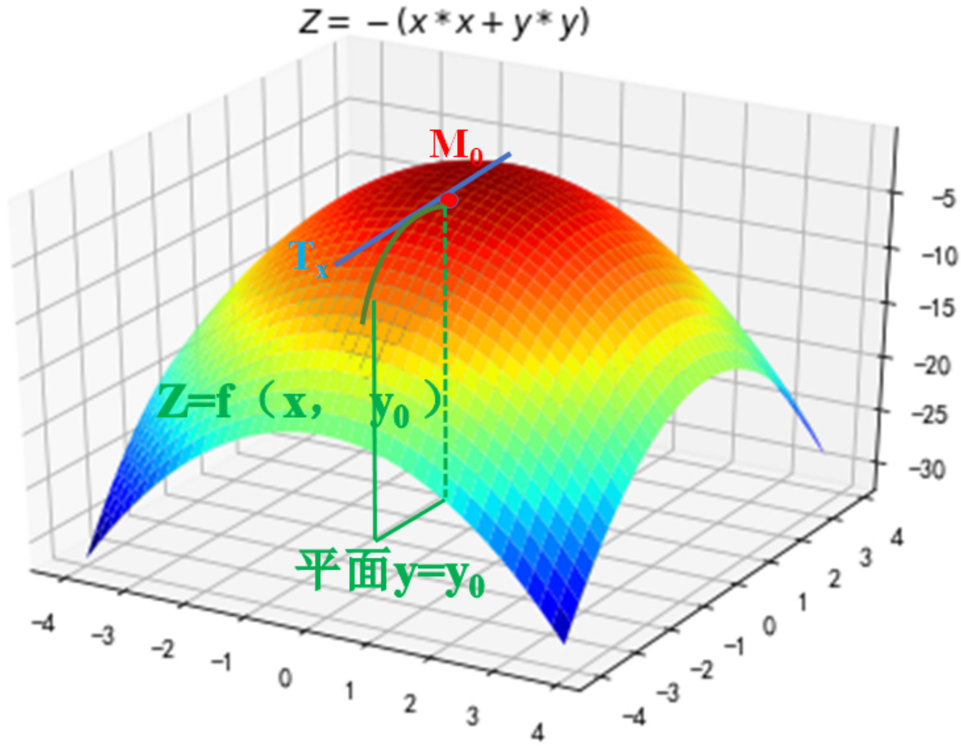
$\frac{\partial}{\partial \theta_2}(0.55 - (5\theta_1 + 2\theta_2 - 12\theta_3)) = -2$

有二元 $z = f(x, y)$，当下面极限存在时：

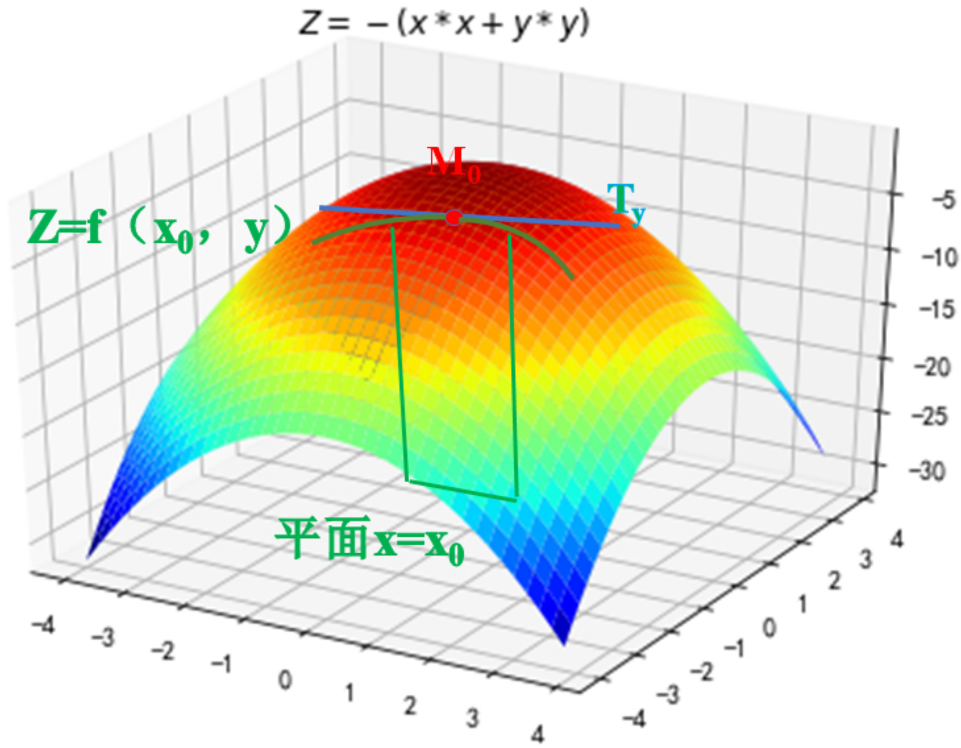
$$\lim_{\Delta x \to 0} \frac{\Delta_x f(x_0, y_0)}{\Delta x} = \lim_{\Delta x \to 0} \frac{f(x_0 + \Delta x, y_0) - f(x_0, y_0)}{\Delta x}$$

称此极限为函数f 在点(x_0, y_0)关于x的偏导数。

同样把偏导数转为图像有：



其中，偏导数$f_x(x_0,y_0)$可以理解为曲平面 $z=f(x,y)$ 与平面$y=y_0$的交线 $z=f(x,y_0)$ 在点M_0处的切线M_0T_x对x轴的斜率；



而偏导数$f_y(x_0,y_0)$可以理解为曲平面 $z=f(x,y)$ 与平面$x=x_0$的交线 $z=f(x_0,y)$ 在点M_0处的切线M_0T_y对y轴的斜率；

事实上，我们不难看出偏导数其实就是多元函数沿着坐标轴方向上的变化率。但在实际情况中，我们不仅要知道多元函数在坐标轴上的变化率，还希望知道在其他特定方向上的变化率，这就引出了下一个概念，方向导数。

设有二元函数 $z=f(x,y)$ 在点$M_0(x_0,y_0)$的某领域$U\left(M_0\right)\subset R^2$内有定义，$l$ 为从点M_0 出发的射线，$M(x+\rho\cos\theta,y+\rho\sin\theta)$ 为 上一点，ρ表示点M与M_0 两点间的距离，θ 表示 l 与 x 轴的夹角，当下面极限存在时：

$$
\begin{aligned}
f_l\left(M_0\right)=\lim_{\rho\rightarrow 0^+}\frac{f(M)-f\left(M_0\right)}{\rho}\\
=\frac{f\left(x_0+\rho\cos\theta,y_0+\rho\sin\theta\right)-f\left(x_0,y_0\right)}{\rho}
\end{aligned}
$$

则称其为函数 f 在点 M_0沿方向 l 的方向导数。

实际上，若函数f 在点 $M_0(x_0,y_0)$ 处可微，则 f 在点 M_0处沿任一方向 l 的方向导数都存在，可以证明有：

$f_l\left(M_0\right)=f_x\left(M_0\right)\cos\theta+f_y\left(M_0\right)\sin\theta$

证明：设$M(x,y)$ 为 l上任一点，有：

$x-x_0=\Delta x=\rho\cos\theta$

$y-y_0=\Delta y=\rho\sin\theta$

因为 f 在点 M_0 处可微，所以有：

$f(M)-f\left(M_0\right)=f_x\left(M_0\right)\Delta x+f_y\left(M_0\right)\Delta y+o(\rho)$

上式左右两边同时除以 ρ ，则有：

$\frac{f(M)-f(M_0)}{\rho}=\frac{f_x(M_0)\Delta x+f_y(M_0)\Delta y+o(\rho)}{\rho}$

$\qquad=f_x\left(M_0\right)\cos\theta+f_y\left(M_0\right)\sin\theta+\frac{o(\rho)}{\rho}$

当$\rho\rightarrow 0$ 时，有$\frac{o(\rho)}{\rho}\rightarrow 0$ ，所以有：

$f_l\left(M_0\right)=\lim_{\rho\rightarrow 0^+}\frac{f(M)-f(M_0)}{\rho}=f_x\left(M_0\right)\cos\theta+f_y\left(M_0\right)\sin\theta$

1.3 梯度及其应用

1.3.1 梯度下降法

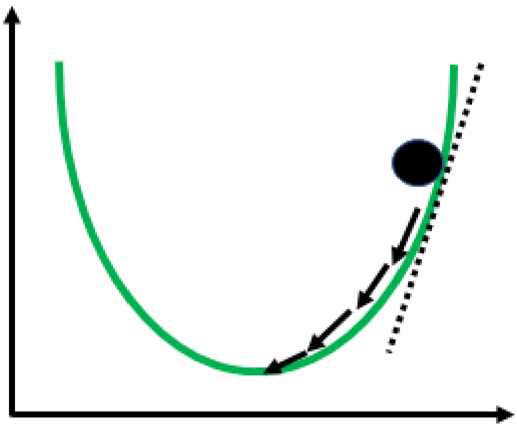
在前面，我们讨论了导数及方向导数。接下来，我们不禁要问在多元函数的曲平面上，哪个方向是函数变化最快的方向?

定义向量$g=\left(f_x(x,y),f_y(x,y)\right)$ 有方向 l 的单位向量 $l_0(\cos\theta,\sin\theta)$，那么方向导数的公式等于：$f_l\left(M_0\right)=g\left(M_0\right)\cdot l_0=\left|g\left(M_0\right)\right|\cdot\left|l_0\right|\cdot\cos\alpha=\left|g\left(M_0\right)\right|\cdot\cos\alpha$ ，其中 α 为向量 g 与单位向量 l_0 的夹角。

上述式中，当 $\alpha=0$ 时，$f_l(M_0)$ 能够取得最大值 $|g(M_0)|$ 。即当向量 l_0 与向量 g 的方向一致时，函数的增长最快，因此我们把向量 g 定义为函数 f 的梯度。进而我们说梯度方向是函数增长最快的方向，而负梯度方向则是函数降低最快的方向。正是由于梯度的这个性质，我们就有了梯度下降法，来对函数进行求解。

1. **梯度下降法**

我们可以把梯度下降法理解为一个下山（求解函数极值）的过程。假设这样一个场景：小张今天去登山，在登到一半的时候，出现了浓雾，因此需要谨记下山（假定目标就是山谷，即山的最低点）。但由于出现了浓雾，小张已经找不到回去的道路了。这个时候，他可以尝试利用梯度下降法找到下山的道路。由于没有导航可以给小张指示一条下山路径，他只能利用当前所处的位置信息来判断哪条道路适合下山。具体来看，小张是利用目前所在的位置，找到目前位置一个最陡峭的下降方向，然后朝着这个方向走。由于下山的道路是不断起伏的，所以小张发现，他需要每走一段距离后，再重复利用这个方法重新修订一下接下来的行走方向，通过不断重复这个过程，小张就能到达山谷。



我们把这个过程转换为数学表达，如下所示：

在机器学习算法中，我们在定义损失函数后，需要对损失函数最小化进行参数求解。而梯度下降法是在无约束优化问题时的经典方法。

定义问题如下：有目标函数$f(x)$，假设 $f(x)$ 是 R^n 连续可微函数，要求解的无约束最优化问题是：求得$\min_{x \in R^n} f(x)$ 的x^* 。

梯度下降法是一种迭代算法，目标是构造一个序列 $x^{(0)}, x^{(1)}, x^{(2)}, \ldots$ 有

$$f\left(x^{(t+1)}\right)<f\left(x^{(t)}\right), t=0,1,2, \ldots$$

只要上述过程不断执行下去总能收敛到局部极小值（为什么是局部极小值而不是全局最小值？下面再探讨），因此现在的问题就转化为沿着哪个方向更新 x ，能够使得函数值 $f(x)$ 下降得最快？答案是负梯度下降方向。

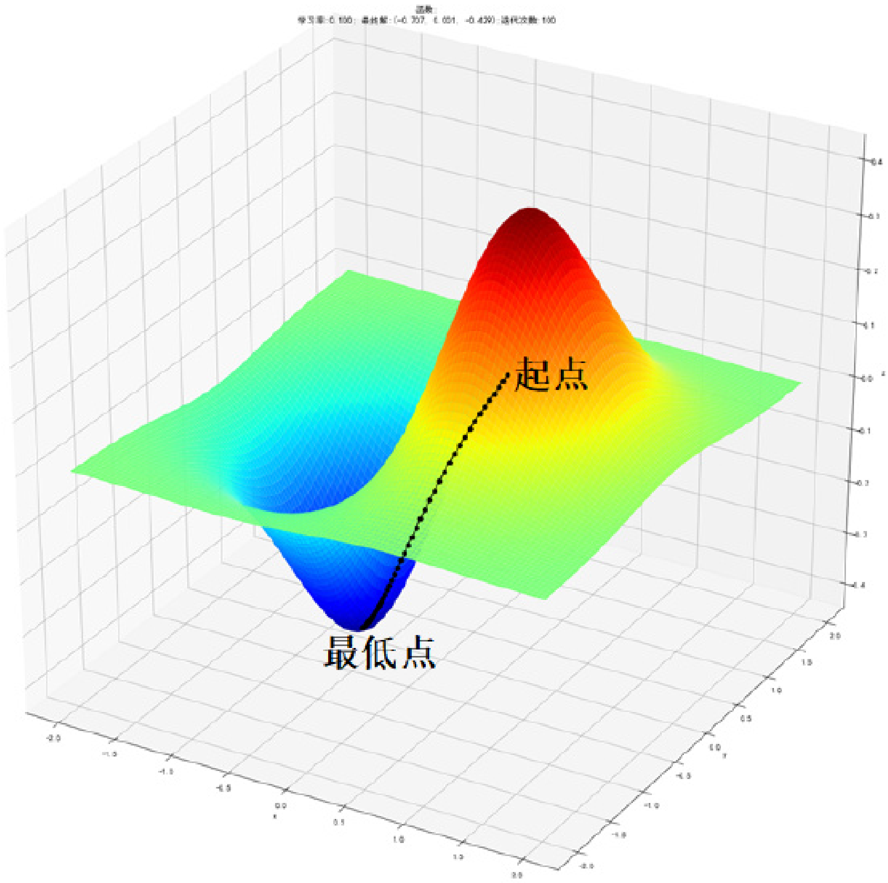
由于 $f(x)$ 是连续可微函数，函数 $f(x)$ 在 $x^{(t)}$ 处的一阶泰勒展开式有：

$$f\left(x^{(t)}+\Delta x\right) \approx f\left(x^{(t)}\right)+\Delta x^T \nabla f\left(x^{(t)}\right)$$

要保证 $f\left(x^{(t)}+\Delta x\right)<f\left(x^{(t)}\right)$ ，则可以选择 $\Delta x=-\lambda_t \nabla f\left(x^{(t)}\right)$ ，其中 $-\nabla f\left(x^{(t)}\right)$ 就是负梯度方向，λ_t 是每步步长，特别地， λ_t 可以由以下公式确定：

$$f\left(x^{(t)}-\lambda_t \nabla f\left(x^{(t)}\right)\right)=\min_{\lambda \geq 0} f\left(x^{(t)}-\lambda_t \nabla f\left(x^{(t)}\right)\right)$$

梯度下降过程示意图：



如上图，事实上梯度下降过程就好像从山上某一起点要下到山下的最低点。当然，从起点到最低点存在很多条路，那么我们应该怎么走呢？一个方法就是开始时，环顾四周，先找到一个下降最快的方向，之后沿着这个方向走一步。走一步后，重新环顾四周，再找到一个下降最快的方向，于是再走一步。之后不断重复这个过程，直到你发现去到某一个点，环顾四周，方向都比你高，则认为到达终点了。好了，到目前位置，我们看到了一个 λ_t ，事实上 λ_t 就是我们下山的步长。我们说过，我们每次测量一次方向想再走一步。但是这一步究竟多长呢？为了能够在尽可能短的时间内下山，我们应该减少测量方向的次数（因为每一次重新测量都需要耗费时间！），但是如果测量次数过少，那我们就很可能偏离了正确的下山方向。所以步长的选取，在梯度下降法中是一个关键，关于这点，我们在后面还会展开讨论。

好了，让我们把这个过程重写为一个操作步骤：

具体来说，梯度下降算法步骤如下：

输入：目标函数 $f(x)$ ，梯度函数 $g(x)=\nabla f(x)$ ，计算精度 ε .

输出：函数极小值$\min_{x\in R^n} f(x)$ 对应的x^* 。

（1） $t=0$ ，初始化 $x^0\in R^n$

（2） 计算 $f(x^{(t)})$

（3） 计算梯度函数 $\nabla f(x^{(t)})$ ，当梯度下降的距离 $\left\|\nabla f\left(x^{(t)}\right)\right\|<\varepsilon$ 时，停止迭代，令 $x^*=x^{(t)}$；否则求 λ_t， λ_t 满足 $f\left(x^{(t)}-\lambda_t\nabla f\left(x^{(t)}\right)\right)=\min_{\lambda\geq 0} f\left(x^{(t)}-\lambda_t\nabla f\left(x^{(t)}\right)\right)$ ；

（4） 令 $x^{(t+1)}=x^{(t)}-\lambda_t\nabla f\left(x^{(t)}\right)$ ，并计算 $f\left(x^{(t+1)}\right)$，当 $\left\|f\left(x^{(t+1)}\right)-f\left(x^{(t)}\right)\right\|\leq\varepsilon$ 或 $\left\|x^{(t+1)}-x^{(t)}\right\|\leq\varepsilon$ 时，停止迭代，令 $x^*=x^{(t+1)}$ ；

（5） 令 $t=t+1$ ，并返回步骤（3）；

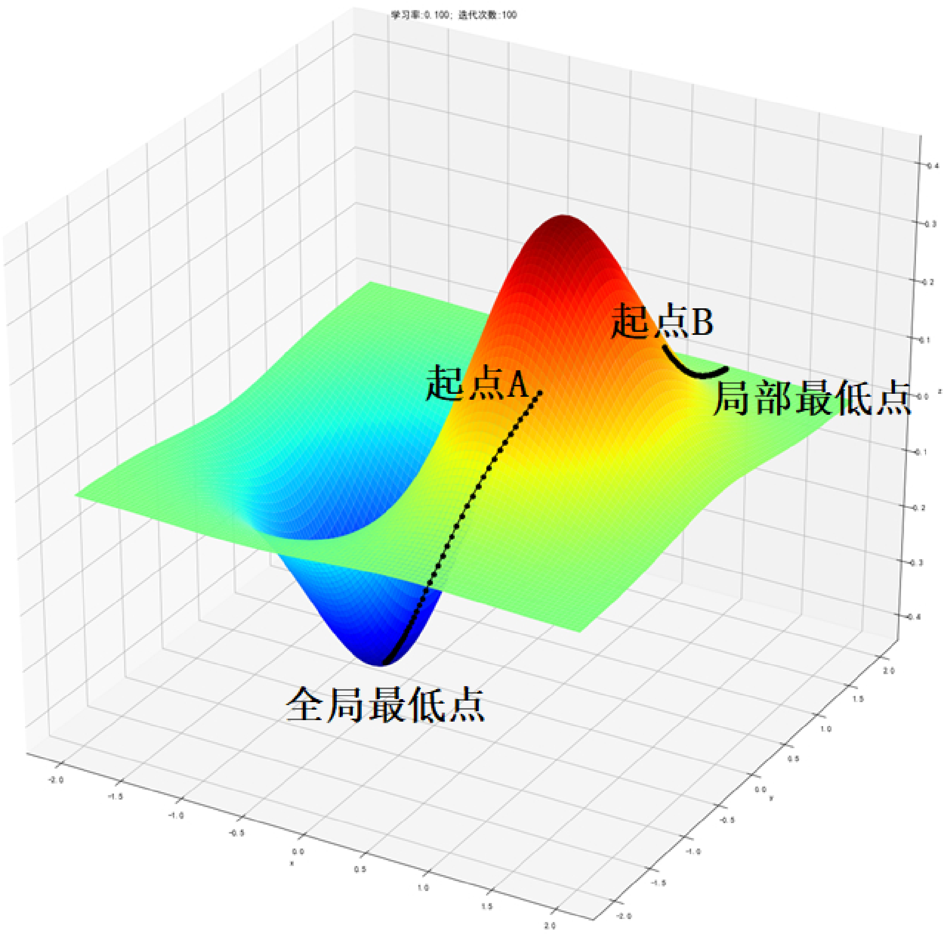
从上式中可以发现迭代的三个停止条件：

（1） 梯度小于指定阈值；

（2） 目标函数变化小于指定阈值；

（3） 迭代变量 x 小于指定阈值；

值得注意的是，当目标函数是凸函数时，梯度下降法的解是全局最优解，这是没有问题的。但是在很多情况下，我们通过梯度下降到达的不一定是全局最低点，而是局部最低点，如下图所示，分别从A点和B点出发，通过若干次迭代后，分别落于全局最低点和局部最低点。



在多变量情况下，梯度就是多元可微函数的偏导方向，这个方向就用梯度 $(grad = ai + bj)$ 这个向量来表示，其中 a 是函数在 x 方向上的偏导数，b 是函数在 y 方向上的偏导数，梯度的模就是这个最大方向导数的值。

梯度（Gradient）：

对于可微函数:

$$f : R^n \rightarrow R$$
$$\nabla f = \left(\frac{\partial f}{\partial x_1}\right), \ldots, \frac{\partial f}{\partial x_n}$$
$$\lim_{h \rightarrow 0} \frac{\|f(x+h) - f(x) - \nabla f(x) \cdot h\|}{\|h\|} = 0$$

实际上，梯度就是多变量微分的一般化。

$$
J(\Theta)=0.55-\left(5\theta_{1}+2\theta_{2}-12\theta_{3}\right)
$$
$$
\nabla J(\Theta)=\left\langle\frac{\partial J}{\partial\theta_{1}},\frac{\partial J}{\partial\theta_{2}},\frac{\partial J}{\partial\theta_{3}}\right\rangle=\langle-5,-2,12\rangle
$$

我们可以看到，梯度就是分别对每个变量进行微分，然后用逗号分割开，梯度用<>包括起来，说明其是一个向量。

2. **梯度下降实例**

我们已经基本了解了梯度下降算法的计算过程，那么我们就来看几个梯度下降算法的小实例，首先从单变量的函数开始。

（1）单变量函数的梯度下降

我们假设有一个单变量的函数：

$$
J(\theta)=\theta^{2}
$$

函数的微分:

$$
J^{\prime}(\theta)=2\theta
$$

初始化，起点为:

$$
\theta^{0}=1
$$

学习率为:

$$
\alpha=4
$$

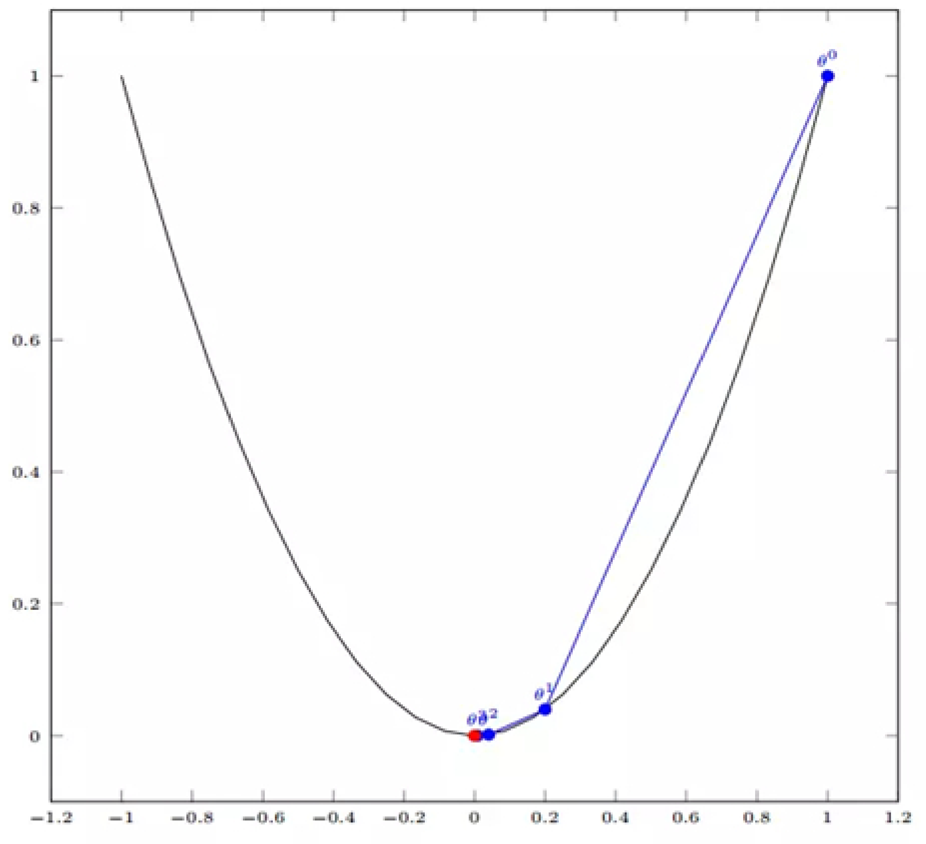
根据梯度下降的计算公式:

$$
\Theta^{1}=\Theta^{0}-\alpha\nabla J(\Theta)\cdot\text{ evaluated at }\Theta^{0}
$$

我们开始进行梯度下降的迭代计算过程：

$$
\begin{gathered}
\theta^{0}=1\\
\theta^{1}=\theta^{0}-\alpha^{*}J^{\prime}\left(\theta^{0}\right)=1-0.4^{*}2=0.2\\
\theta^{2}=\theta^{1}-\alpha^{*}J^{\prime}\left(\theta^{1}\right)=0.04\\
\theta^{3}=0.008\\
\theta^{4}=0.0016
\end{gathered}
$$

如图，经过四次的运算，也就是走了四步，基本就抵达了函数的最低点，也就是山底。



（2）多变量函数的梯度下降

我们假设有一个目标函数：

$$J\Theta = \theta_1^2 + \theta_2^2$$

现在要通过梯度下降法计算这个函数的最小值。我们通过观察就能发现最小值其实就是(0，0)点。但是接下来，我们会从梯度下降算法开始一步步计算到这个最小值。

函数的微分：

$$J'(\theta) = 2\theta_1 + 2\theta_2$$

我们假设初始的起点为：

$$\Theta^0 = (1,3)$$

初始学习率为：

$$\alpha = 0.1$$

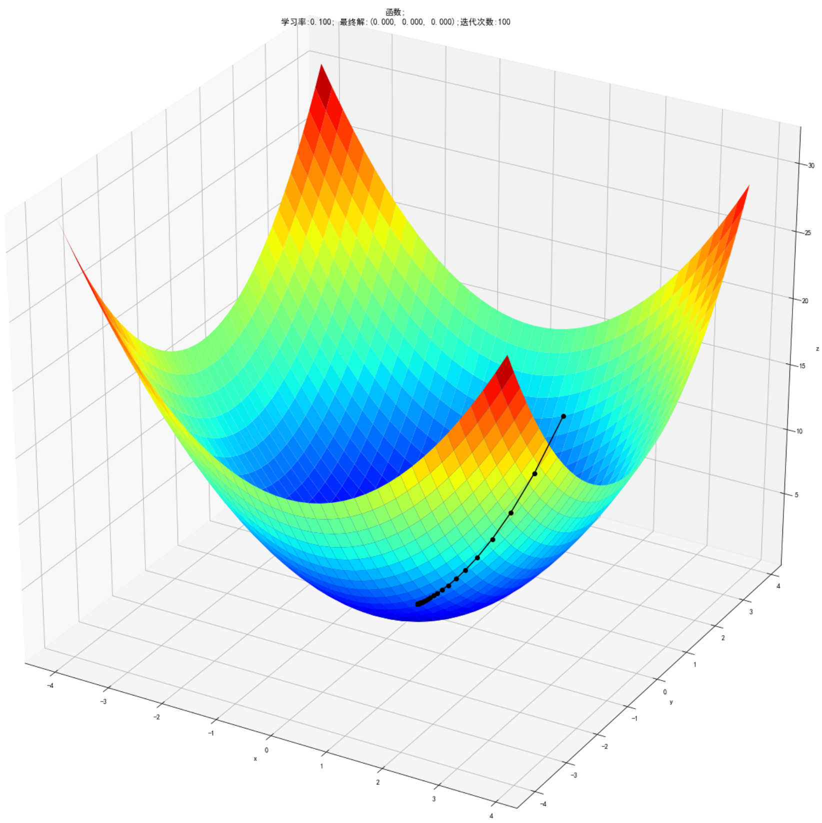
函数的梯度为：

$$\nabla J(\Theta) = \langle 2\theta_1, 2\theta_2 \rangle$$

我们开始进行梯度下降的迭代计算过程：

$$
\begin{gathered}
\Theta^0=(1,3) \\
\Theta^1=\Theta^0-\alpha \nabla J(\Theta)=(1,3)-0.1(2,6)=(0.8,2.4) \\
\Theta^2=(0.8,2.4)-0.1(1.6,4.8)=(0.64,1.92) \\
\Theta^3=(0.512,1.536) \\
\ldots \\
\ldots \\
\ldots \\
\Theta^{100}=\left(1.6296287810675902 e^{-10}, 4.8888886343202771 e^{-10}\right)
\end{gathered}
$$

我们发现，已经基本靠近函数的最小值点。



进一步，以回归分析为例：

对于多元回归分析，我们有回归公式如下所示：

$$
y=\theta_0+\theta_1 x_1+\theta_2 x_2+\ldots+\theta_m x_m
$$

其中我们的数据集 D 中一共有 n 个样本，包括 m 个变量进行描述。数据集 D 中的自变量表示成矩阵 X，第一列置为1，表示回归方程中的常数项：

$$
y=X \beta+\varepsilon
$$

其中:

$$
Y=\left[\begin{array}{c}
y_1 \\
y_2 \\
\vdots \\
y_n
\end{array}\right], \quad \theta=\left[\begin{array}{c}
\theta_0 \\
\theta_1 \\
\vdots \\
\theta_m
\end{array}\right], \quad X=\left[\begin{array}{ccccc}
1 & x_{11} & x_{12} & \ldots & x_{1 n} \\
1 & x_{21} & x_{22} & \ldots & x_{2 m} \\
\vdots & \vdots & \vdots & & \vdots \\
1 & x_{n 1} & x_{n 2} & \ldots & x_{n n}
\end{array}\right], \boldsymbol{\varepsilon}=\left[\begin{array}{c}
\varepsilon_1 \\
\varepsilon_2 \\
\vdots \\
\varepsilon_n
\end{array}\right]
$$

针对于该方程中的未知参数，我们可以利用最小二乘法进行估计，损失函数方程有：

$$L(\beta)=\frac{1}{2}\sum_{i=1}^{n}\left(\theta_0+\theta_1 x_1+\theta_2 x_{i2}+\cdots+\theta_m x_{in}-y_i\right)^2$$

有梯度下降公式如下：

$$\theta=\theta-\alpha\cdot\frac{\partial L(\theta)}{\partial(\theta)}$$

对于第 j 个变量，有：

$$\begin{aligned}
&\tfrac{\partial L(\theta)}{\partial(\theta_j)}=\tfrac{\partial}{\partial(\theta_j)}\tfrac{1}{2}\textstyle\sum_{i=1}^{n}\left(f_\theta\left(x_i\right)-y_i\right)^2\\
&=2\cdot\tfrac{1}{2}\textstyle\sum_{i=1}^{n}\left(\theta_0+\theta_1 x_{i1}+\theta_2 x_{i2}+\cdots+\theta_m x_{in}-y_i\right)\cdot x_{ij}\\
&=\textstyle\sum_{i=1}^{n}\left(\theta_0+\theta_1 x_{i1}+\theta_2 x_{i2}+\cdots+\theta_m x_{in}-y_i\right)\cdot x_{ij}\\
&=\textstyle\sum_{i=1}^{n}\left(f_\theta\left(x_i\right)-y_i\right)\cdot x_{ij}
\end{aligned}$$

所以有更新公式如下：

$$\theta_j=\theta_j-\alpha\sum_{i=1}^{n}\left(f_\theta\left(x_i\right)-y_i\right)\cdot x_{ij}$$

3. **梯度下降的应用方法**

在实际应用当中，梯度下降有三种方式分别是批量梯度下降（Batch Gradient Descent，BGD），随机梯度下降（Stochastic Gradient Descent，SGD）以及小批量梯度下降（Mini-batch Gradient Descent，MBGD）

（1） 批量梯度下降（Batch gradient descent）：最经典的梯度下降方法，就是上述例子所使用的计算公式，在计算时，每次更新都使用所有的样本进行计算，仍以回归分析为例，有：

$$\theta_j=\theta_j-\alpha\sum_{i=1}^{n}\left(f_\theta\left(x_i\right)-y_i\right)\cdot x_{ij}$$

（2） 随机梯度下降（Stochastic gradient descent, SGD），是批量梯度下降的一个变体，相对于批量梯度下降需要使用所有的样本进行更新，随机梯度下降每次更新只使用一个样本进行计算：

$$\theta_j=\theta_j-\alpha\left(f_\theta\left(x_i\right)-y_i\right)\cdot x_{ij}$$

由于随机梯度下降每次只使用一个样本进行迭代更新，因此训练速度要比批量梯度下降快得多，但是也由于每次只使用一个样本更新，需要的迭代次数将增加（但整体迭代时间还是要比批量梯度下降更快），而且得到的解可能并非最优解。

（3） 小批量梯度下降（Mini-batch Gradient Descent）：批量梯度下降和随机梯度下降实际上就是两个极端，一个使用所有样本，另一只使用一个样本，而小批量梯度下降则可以视为两种方法的折中，我们将使用 k 个样本进行更新：

$$\theta_j=\theta_j-\alpha\sum_{i=1}^{k}\left(f_\theta\left(x_i\right)-y_i\right)\cdot x_{ij}$$

梯度下降的其他问题：

（1） 初始点的选择：初始点的选择没有确定的方法。当初始点距离最优点较近的时候，可以获得更快的收敛速度。对于凸问题，任意初始点都能获得最优解，而对于非凸问题，则只有在最优点的邻域内才能获得最优解，否则只是局部最小。在实验中，可以尝试选择多个不同值作为初始点进行训练，选择能够获得损失函数最小值的初始点作为初始点。

（2） 学习率的选择：学习率既可以选择固定值，也可以选择自适应方法。一般来说，自适应方法迭代的早期会设定较大的步长，而随着迭代次数的增加，步长慢慢减小，以此来防止错过最优解。

存在问题：

- 确定合适的学习率非常困难

  ——大的学习率可保证收敛性，但收敛过程很慢

  ——大的学习率会导致损失函数优化过程在极值点附近波动甚至发散

- 学习率规划适应性差

  ——在训练过程中，可采用模拟退火等方法调整学习率

  ——或根据损失函数值的下降幅度大小调整学习率

  ——需手工预先设置，对训练集的自适应性差

- 不同参数采用相同的学习率

  ——稀疏数据、特征尺度不一致，不宜采用相同的学习率

- SGD难以跳出局部极小值点（或鞍点）

  ——神经网络对应的函数具有高度非线性

  ——众多局部较小点（或鞍点）

  ——在极小点或鞍点附近，目标函数值几乎不变，导致梯度近似为零。

4. **梯度下降的调整方法**

(1) **传统的梯度下降法：**

$$\theta_{k+1}=\theta_k-\eta\nabla J_\theta\left(\theta_k\right)$$

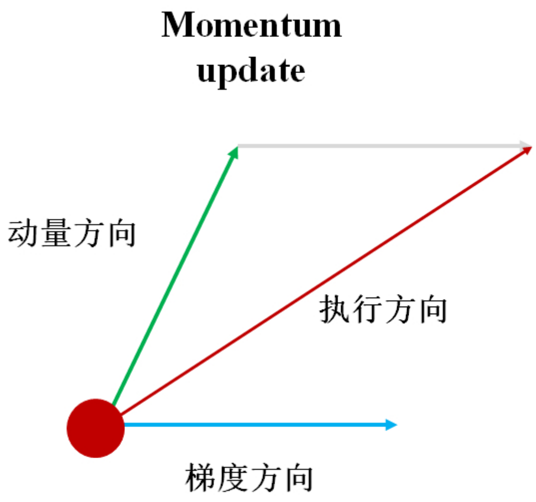
(2) **动量法（Momentmu，SGD+动量）：**

在经典力学中，动量表示为物体的质量和速度的乘积，物体的动量指的是这个物体在它运动方向上保持运动的趋势。动量法其实是在原有运动的基础上结合新的运动方向。因此动量法的核心思想是：在梯度方向一致的地方加速，在梯度方向不断改变的地方减速。其更新公式为(v 为每次的更新量)：

$$\begin{aligned}
&v_1=\eta\nabla J\left(\theta_1\right)\\
&v_k=\gamma v_{k-1}+\eta\nabla J\left(\theta_{k-1}\right),\quad \gamma\in(0,1)\\
&\theta_k=\theta_{k-1}-v_k
\end{aligned}$$

可以看到相比于传统梯度下降，动量法在更新项目里加上一个方向的趋势 γv_{k-1} 。其实有：

$$v_k=\eta\sum_{i=0}^{k-1}\gamma^i\nabla J\left(\theta_{k-1-i}\right)=\eta\gamma^0\nabla J\left(\theta_{k-1}\right)+\gamma^1\nabla J\left(\theta_{k-2}\right)+\cdots\gamma^{k-1}\nabla J\left(\theta_0\right)$$

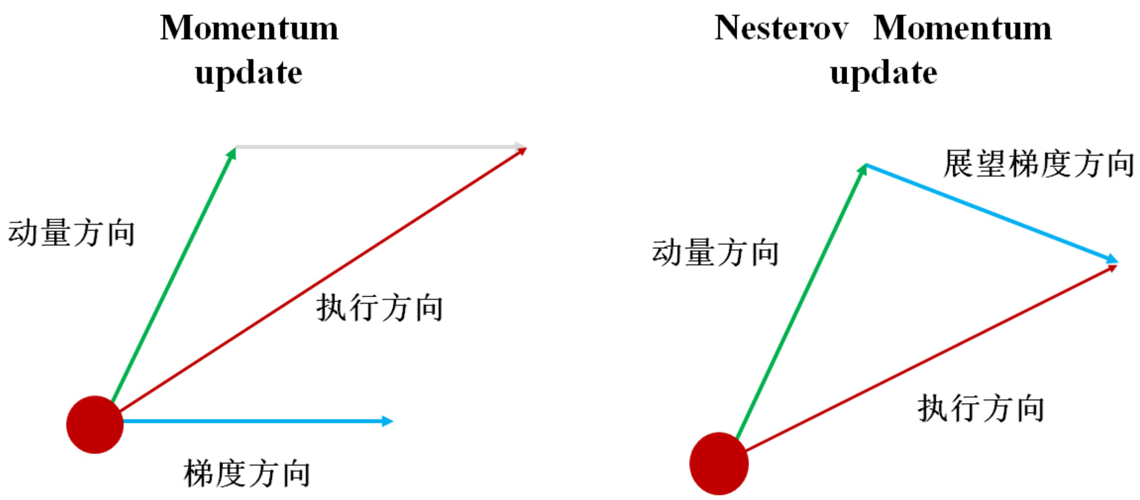


可以看到由于加入了动量，所以动量法比较平缓。

（3） **NAG: 涅斯捷罗夫梯度加速法(NAG, Nesterov accelerated gradient):**

从动量法可知，方向的更新包括两个部分，一个是上一时刻的更新值 γv_{k-1}，一个是基于当前位置的梯度 $\eta \nabla J\left(\theta_{k-1}\right)$。 NAG方法认为，既然本次更新已经决定有 γv_{k-1}，那为什么不先走 γv_{k-1}，再在这个新的位置上计算梯度进行。于是有了如下的公式：

$$\begin{aligned} & v_k=\gamma v_{k-1}+\eta \nabla J_\theta\left(\theta_{k-1}-\gamma v_{k-1}\right), \quad \gamma \in(0,1) \\ & \theta_k=\theta_{k-1}-v_k \end{aligned}$$



（4） **AdaGrad（Adaptive Gradient）：自适应学习率。**

上面的方法中，对于每一个参数都使用了相同的学习率。而AdaGrad的基本思想是对每个变量用不同的学习率，这个学习率在一开始比较大，用于快速梯度下降。随着优化过程的进行，对于已经下降很多的变量，则减缓学习率，对于还没怎么下降的变量，则保持一个较大的学习率。

设 $g_{t,i}$ 为第 t 轮第 i 个参数的梯度，即 $g_{t,i}=\nabla J\left(\theta_{t,i}\right)$，对于训练中的每个参数 θ_i 有更新公式：

$$\theta_{t+1,j}=\theta_{t,i}-\frac{\eta}{\sqrt{G_{t,ii}+\varepsilon}} \cdot g_{t,j}$$

其中 $G_t \in \mathbb{R}^{d \times d}$ 为对角矩阵，每个对角线位置 i, i 为对应参数 θ_i 从第1轮到第t轮的平方和。$G_{t,ii}$ 则是矩阵 G_t 在第i, i的位置。ε 是平滑项，用于避免分母为0，一般取值1e-8。

AdaGrad赋予高频出现的特征以较低的学习率，而赋予低频出现的特征以较高的学习率，从而有助于学习器发现具有高鉴别性的低频出现的特征。但Adagrad的缺点是在训练的中后期，分母上梯度平方的累加将会越来越大，从而梯度趋近于0，使得训练提前结束。

（5） **RMSprop**

RMSprop是Geoff Hinton提出的一种自适应学习率方法。（在课堂提出，没有正式发表。）Adagrad会累加之前所有的梯度平方，而RMSprop仅仅是计算对应的平均值。

$$
\begin{aligned}
v_{t,i} &= \rho v_{t-1,i} + (1-\rho) \times g_{t,i}^2 \\
\theta_{t+1,i} &= \theta_{t,i} - \frac{\eta}{\sqrt{v_{t,i}+\varepsilon}} \cdot g_{t,i}
\end{aligned}
$$

另一种写法：

$$
\begin{aligned}
\mathbf{E}\left[g^2\right]_t &= 0.9\mathbf{E}\left[g^2\right]_{t-1} + 0.1 g_t^2 \\
\theta_{t+1} &= \theta_t - \frac{\eta}{\sqrt{\mathbf{E}\left[g^2\right]_t+\varepsilon}} g_t
\end{aligned}
$$

其中 E 代表求期望.

（6） **Adadelta**

前面说到Adagrad不仅到了中后期，梯度会趋近于0，而且依然依赖于一个人工设置的全局学习率。所以AdaDelta作为一个改进版，他的改进之处在于：

- 计算历史梯度信息时引入时间窗，而非所有历史时间（通过引入衰减因子实现，类似于AdaGrad方法）
- 不用设置学习率参数

基本步骤：

- 计算时刻 t 的梯度 g_t；
- 累计梯度信息：$\mathbf{E}\left[g^2\right]_t = \rho\mathbf{E}\left[g^2\right]_{t-1} + (1-\rho) g_t^2$，其中 ρ 为衰减因子；
- 计算更新量：$\Delta x_t = -\frac{\sqrt{\mathbf{E}\left[\Delta x^2\right]_{t-1}+\varepsilon}}{\sqrt{\mathbf{E}\left[g^2\right]_t+\varepsilon}} g_t$；
- 累计更新量信息：$\mathbf{E}\left[\Delta x^2\right]_t = \rho\mathbf{E}\left[\Delta x^2\right]_{t-1} + (1-\rho)\Delta x_t^2$；
- 更新参数：$x_{t+1} = x_t + \Delta x_t$。

（7） **Adam**

Adam(Adaptive Moment Estimation)是另一种自适应学习率的方法。与RMSprop相比，其在梯度平方估计（二阶矩）的基础上增加了对梯度（一阶矩）的估计，Adam的优点主要在于经过偏置校正后，每一次迭代学习率都有个确定范围，使得参数比较平稳。

公式表达如下：

$$
\begin{aligned}
m_t &= \beta_1 m_{t-1} + \left(1-\beta_1\right) g_t \\
v_t &= \beta_2 v_{t-1} + \left(1-\beta_2\right) g_t^2 \\
&\downarrow \\
\hat{m}_t &= \frac{m_t}{1-\beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t} \\
&\downarrow \\
\theta_{t+1} &= \frac{\eta}{\sqrt{\hat{v}_t}+\varepsilon} \hat{m}_t
\end{aligned}
$$

上式中 m_t 和 v_t 分别是对梯度的一阶矩估计和二阶矩估计，可以看作对期望 $E(g_t)$， $E(g_t^2)$ 的近似；m_t' 和 v_t' 是对 m_t 和 v_t 的校正，这样可以近似为对期望的无偏估计。在数据比较稀疏的时候，adaptive的方法能得到更好的效果，例如Adagrad，RMSprop, Adam 等。Adam 方法也会比 RMSprop方法收敛的结果要好一些, 所以在实际应用中 ，Adam为最常用的方法，可以比较快地得到一个预估结果。

方法总结：

- SGD+Momentum方法最基本，调参较难；
- RMSprop和Adadelta是AdaGrad的改进方法；
- RMSprop、Adadelta和Adam方法性能相近；
- Adadelta方法无需设置学习率参数；
- NAG方法在RNN网络中效果显著；

一般建议可以使用Adam或者Adadelta。另外有一种建议是先使用Adam，当误差不再下降再改用SGD+Momentum。

各种方法比较动图可参考：（http://ruder.io/optimizing-gradient-descent/index.html）

1.3.2 海森矩阵

将一个一元函数 $f(x)$ 在 x_0 处进行泰勒展开，可以得到以下公式：

$$f\left(x_0+\Delta x\right)=f\left(x_0\right)+\frac{f'\left(x_0\right)}{1!}(\Delta x)+\frac{f''\left(x_0\right)}{2!}(\Delta x)^2+o\left((\Delta x)^2\right)$$

其中二阶导数的部分映射到二维以及多维空间就是Hessian Matrix。

在数学中，海森矩阵(Hessian matrix或Hessian)是一个自变量为向量的实值函数的二阶偏导数组成的方块矩阵，此函数如下：

$$f\left(x_1, x_2, \ldots, x_n\right)$$

如果 f 的所有二阶导数都存在，那么 f 的海森矩阵即：

$$H(f)_{ij}(x)=D_i D_j f(x)$$

其中 $\mathrm{x}=\left(x_1, x_2, \ldots, x_n\right)$ ，即 $H(f)$ 为：

$$\left[\begin{array}{cccc}
\frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\
\frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2}
\end{array}\right]$$

1.4 泰勒展开式

1.4.1 泰勒展开式

泰勒展开式如下：

$$f(x)=f\left(x_0\right)+\frac{f^{\prime}\left(x_0\right)}{1!}\left(x-x_0\right)+\frac{f^{\prime\prime}\left(x_0\right)}{2!}\left(x-x_0\right)^2+ \\ \ldots+\frac{f^{(n)}\left(x_0\right)}{n!}\left(x-x_0\right)^n+o\left(\left(x-x_0\right)^n\right)$$

令 $x_0=0$，得到麦克劳林公式:

$$f(x)=f(0)+\frac{f^{\prime}(0)}{1!} x+\frac{f^{\prime\prime}(0)}{2!} x^2+\ldots+\frac{f^{(n)}(0)}{n!} x^n+o\left(x^n\right)$$

另外一种形式，令 $x=x_0+\Delta x$,得到:

$$f\left(x_0+\Delta x\right)=f\left(x_0\right)+\frac{f^{\prime}\left(x_0\right)}{1!}(\Delta x)+\frac{f^{\prime\prime}\left(x_0\right)}{2!}(\Delta x)^2+ \\ \ldots+\frac{f^{(n)}\left(x_0\right)}{n!}(\Delta x)^n+o\left((\Delta x)^n\right)$$

1.4.2 泰勒展开的应用

1. 近似计算 e^x：

   令 $e^x=f(x)$，有：$e^x=1+x+\frac{x^2}{2!}+\frac{x^3}{3!}+\ldots+\frac{x^n}{n!}+o\left(x^n\right)$。

1.5 积分

加法的逆运算是减法，乘法的逆运算是除法，那么微分的逆运算就是积分。

设函数 f 与 F 在区间 I 上均有定义，有:

$$F^{\prime}(x)=f(x)$$

则称 F 为 f 在区间 I 上的原函数。

1.5.1 换元积分法

设 $g(u)$ 在 $[\alpha, \beta]$ 上有定义，$u=\varphi(x)$ 在 $[a, b]$ 上有定义，且 $\alpha \leq \varphi(x) \leq \beta$，$x \in[a, b]$，记: $f(x)=g(\varphi(x)) \varphi^{\prime}(x), x \in[a, b]$，

当 $\varphi(x) \neq 0, x \in[a, b]$，且 $f(x)$ 在 $[a, b]$ 上存在原函数 $F(x)$ 时，且 $g(u)$ 在 $[\alpha, \beta]$ 上也存在原函数 $G(u)$，有:

$$\int g(u) d u=\int g(\varphi(x)) \varphi^{\prime}(x) d x=\int f(x) d x=F(x)+C=F\left(\varphi^{-1}(u)\right)+C$$

例，求解 $\int \frac{1}{a^2+x^2} d x$ （令 $u=\frac{x}{a}$ ）

$$\int \frac{1}{a^2+x^2} d x=\int \frac{1}{1+\left(\frac{x}{a}\right)^2} \cdot \frac{1}{a^2} d x=\int \frac{1}{1+\left(\frac{x}{a}\right)^2} \cdot \frac{1}{a} d \frac{x}{a} \\ =\frac{1}{a} \int \frac{1}{1+\left(\frac{x}{a}\right)^2} d \frac{x}{a}=\frac{1}{a} \tan \frac{x}{a}+C$$

1.5.2 分部积分法

若 $u(x)$ 与 $v(x)$ 可导，不定积分 $\int u^{\prime}(x) v(x) d x$ 以及 $\int u(x) v^{\prime}(x) d x$ 也存在，则有:

$$\int u(x) v^{\prime}(x) d x=u(x) v(x)-\int u^{\prime}(x) v(x) d x$$

例，求解 $\int xcos(x)dx$ 。

解：令 $u=x,v'=cos(x)$ ，有 $u'=1,v=sinx$ ，得到：

$$\int x\cos xdx=x\sin x-\int \sin xdx=x\sin x+\cos x+C$$

1.5.3 定积分

若函数 f 在 $[a,b]$ 上连续，且存在原函数 F ，即 $F'(x)=f(x),x\in[a,b]$ ，则 f 在 $[a,b]$ 上可积，且：

$$\int_a^b f(x)dx=F(b)-F(a)$$

实际上，上式结果 A ，就是连续曲线 $f(x)\geq 0$ ，以及直线 $x=a$，$x=b$（$b>a$）和 x 轴围成的曲面梯形面积。上面的要求是保证结果为正。否则，如果结果为负数，A 就是负面积。

1.6 Γ 函数

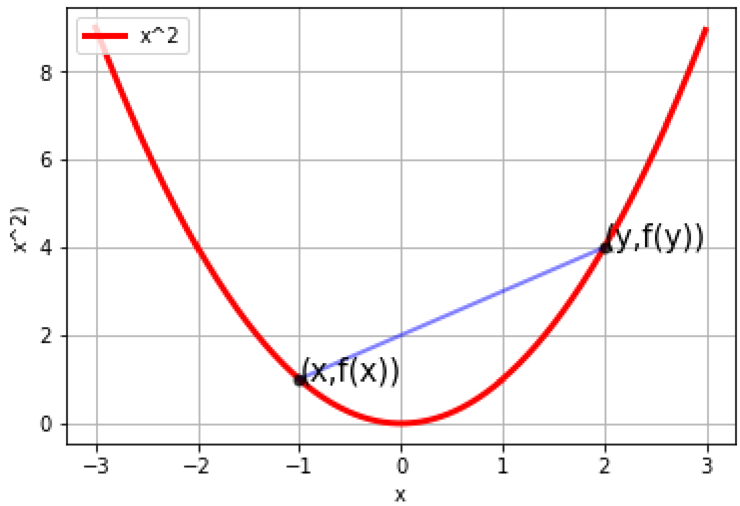
Γ 函数是阶乘在实数域上的推广，这样就能算小数的阶乘：

$$\Gamma(x)=\int_0^\infty t^{x-1}e^{-t}d(t)=(x-1)!$$

1.7 凸函数

1.7.1 凸函数定义

若函数 f 的定义域 $dom f$ 为凸集，且满足对于任意 $x,y\in dom f$ 和任意 $0\leq\theta\leq 1$ ，有 $f(\theta x+(1-\theta)y)\leq\theta f(x)+(1-\theta)f(y)$，则称函数 f 为凸函数。从几何角度去看凸函数的定义，凸函数就是任意两点之间的弦（即这两点构成的线段）都在该函数图像（此处是指这两点之间的函数图像，而非全部的函数图像）的上方。如下图所示：



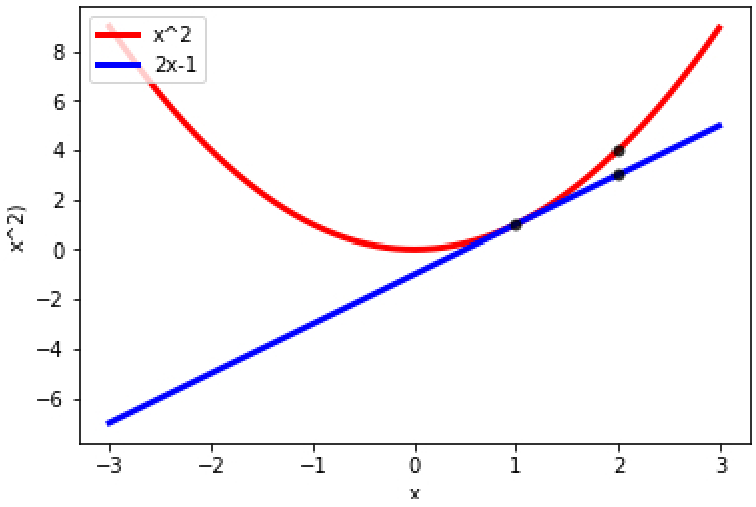
1.7.2 凸函数的判定

（1）对应的一阶可微条件

若 f 一阶可微，则函数 f 为凸函数的充分必要条件是：当且仅当 f 的定义域 $domf$ 为凸集，且满足：

$$\forall x, y \in \operatorname{dom} f, f(y) \geq f(x)+\nabla f(x)^{T}(y-x)$$

从几何角度看，其表示我们对凸函数上的任一点做切线，该切线总是在函数图像的下方，因其只需要一个点就可以判定函数是否为凸函数，因此，该方法又称为“一点法”。如下图所示：



（2）对应的二阶可微条件

若 f 二阶可微，则函数 f 为凸函数的充要条件是：当且仅当 f 的定义域 $domf$ 为凸集，且满足：

$$\nabla^{2} f(x) \succ= 0$$

说明：若 f 是一元函数，上式表示二阶导恒大于等于0。

若 f 是多元函数，上式表示二阶导Hessian矩阵半正定。

从几何角度看，Hessian矩阵半正定表示在点 x 处函数图像具有正（向上）的曲率。

1.7.3 凸函数举例

1. 指数函数：$f(x)=e^{ax}$
2. 幂函数：$f(x)=x^{a}, x \in R^{+}$，$a \geq 1$ 或 $a \leq 0$
3. 负对数函数：$f(x)=-ln(x)$
4. 负熵函数：$f(x)=xln(x)$
5. 范数函数：$f(x)=x$
6. 最大值函数：$f(x)=max(x_{1}, x_{2}, \ldots, x_{n})$
7. 指数线性函数：$f(\vec{x})=\log(e^{x_{1}}+e^{x_{2}}+\ldots+e^{x_{n}})$