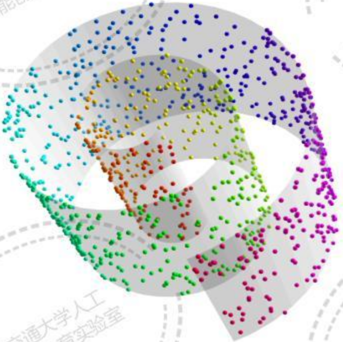
人工智能基础与进阶

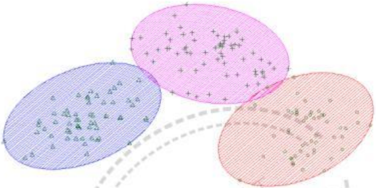
聚类和寻优算法

上海交通大学

目录 content



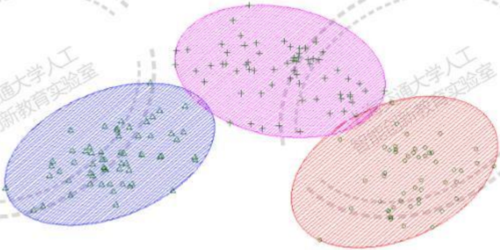
第一节　K-means聚类



K-means聚类—聚类与分类

聚类和分类的区别

- 分类（classification）：监督学习任务，利用已知的样本标记训练学习器预测未知样本的类别
- 聚类（clustering）：无监督学习任务，不知道真实的样本标记，只把相似度高的样本聚合在一起。



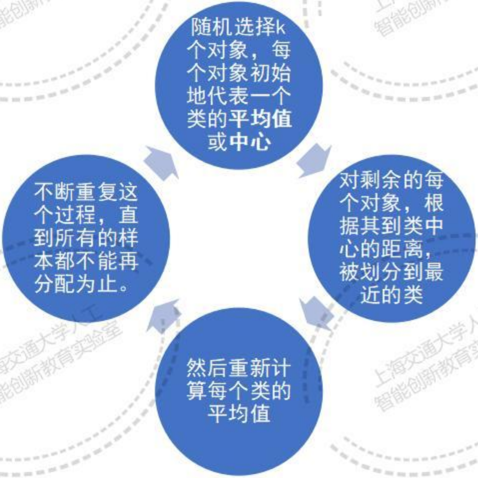
K-means聚类—聚类的定义

聚类的定义

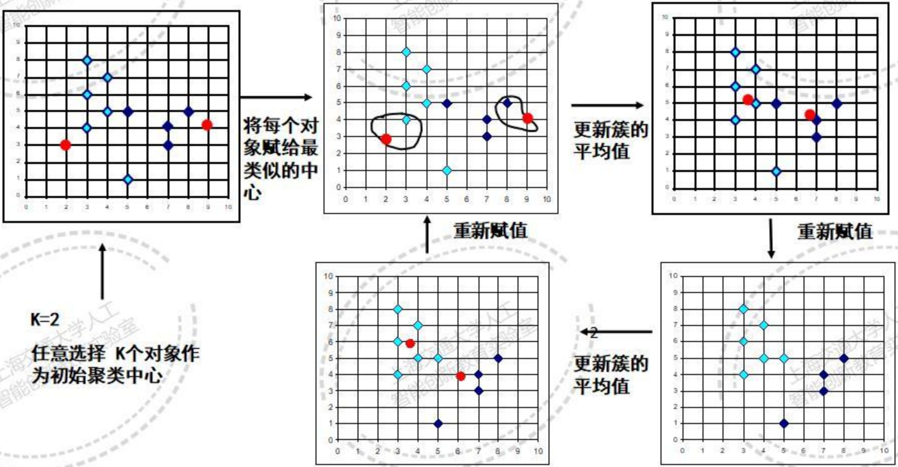
利用样本在多维空间的相对位置，将样本分成两个或多个集团的算法。

按照事务的某些属性，把事务数据分组成为多个类，在同一个类内对象之间具有较高的相似度，不同类之间的对象差别较大。即，使类间的相似性尽量小，类内相似性尽量大。

聚类实施过程



K-means聚类—算法实现



K-means聚类—性能分析

优点：

简单直观，易于理解与实现；

复杂度相对较低，在K不是很大的情况下，K-means的计算时间相对较短；

会产生密度比较高的簇。

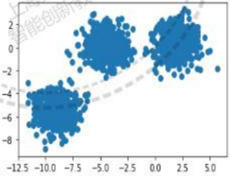
缺点：

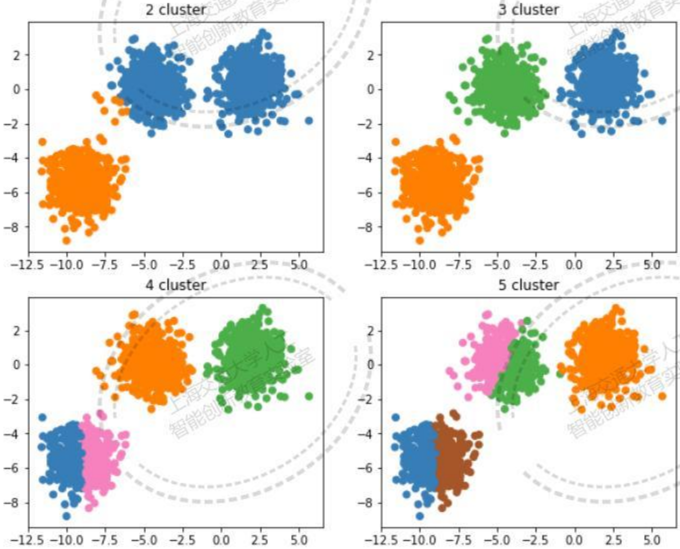
K需要预先得知，很难预测到准确值；

对初始设置很敏感；

对噪声和离异点非常敏感。

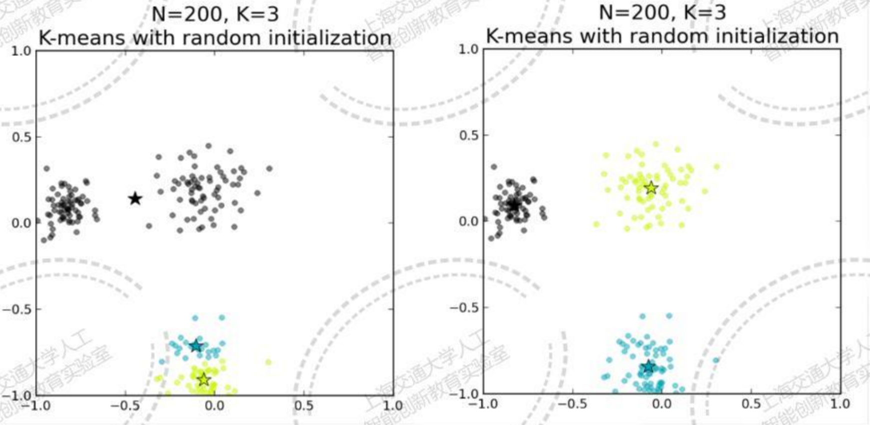
K-means聚类—k值影响





K值的影响

K-means聚类—初始值影响

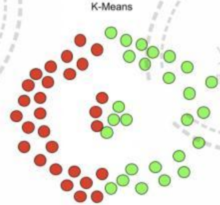


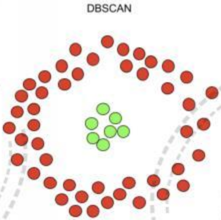
初始值的影响

为什么要引入DBSCAN聚类算法?

- 基于距离的聚类算法例如kmeans，在处理非球状结构的数据集时，聚类效果不好。

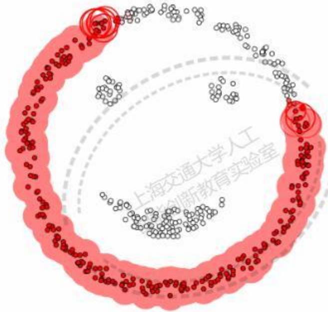
- 与之不同的是，在基于密度的聚类算法例如DBSCAN中，通过在数据集中寻找被低密度区域分离的高密度区域，将分离出的高密度区域作为一个独立的类别，以此发现任意形状的聚类，





DBSCAN 聚类

- DBSCAN 全称为 Density-Based Spatial Clustering Algorithm with Noise，基于密度的噪声空间聚类算法。
- DBSCAN 算法是基于密度对数据点进行处理的，主要是将特征空间中足够密集的点划分为同一个簇，簇的形状可以是任意的，而且数据点中有噪声点的话，不会将这些点划分给某个簇。

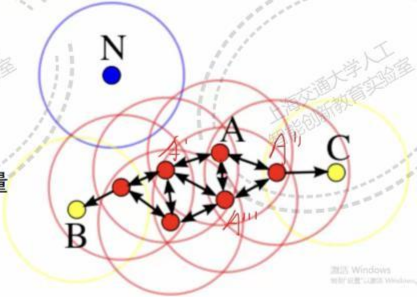


www.naftaliharris.com/blog/visualizing-dbscan-clustering/

DBSCAN 聚类

基本概念

- 核心点：在半径为r的邻域内，含有超过minPts数目的点，成为核心点
- 边界点：在半径为r的邻域内，含有点的数量小于minPts个，但与核心点直接相连
- 离群点：既不是核心点也不是边界点的点

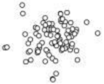


算法步骤

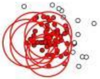
1. 任选一个未被访问过的点p，若点p的r邻域包含多minPts个点，则创建一个以p为核心点的簇；否则，设p为离群点；
2. 访问该点r邻域内的其他点，若还没有被分配一个簇，则将其纳入上一步创建的簇中，若为核心样本，则依次访问其r邻域的其他点，以此类推，直到簇内r邻域内没有更多核心点为止；
3. 选取另一个尚未被访问过的点，重复1,2步骤。

DBSCAN 聚类——半径的影响

半径r：1
阈值minPts：4



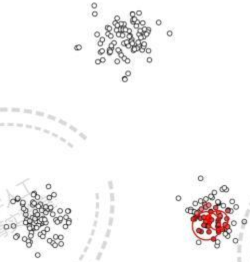
半径r：1.7
阈值minPts：4



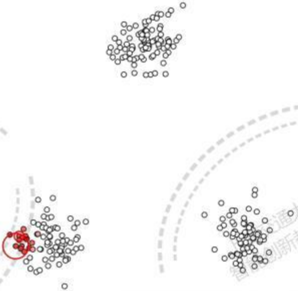
www.naftaliharris.com/blog/visualizing-dbscan-clustering/

DBSCAN 聚类——阈值的影响

半径r：1
阈值minPts：4



半径r：1
阈值minPts：2



www.naftaliharris.com/blog/visualizing-dbscan-clustering/

DBSCAN 聚类

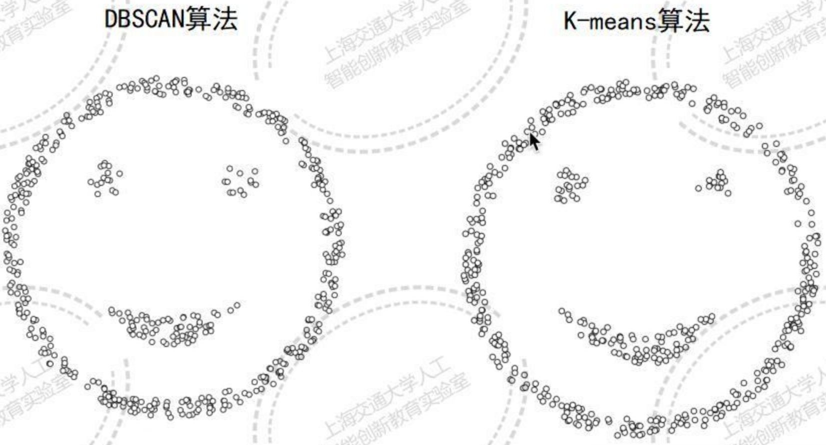
优点：

- 不需要我们指定数据集中簇的个数K。
- 前面说K均值聚类通常只在球形分布的数据集上分类效果比较好，而 DBSCAN 聚类则可以用于各种复杂形状的数据集。
- 可以识别出不属于任何簇的离群点，非常适合用来检测离群点。

缺点：

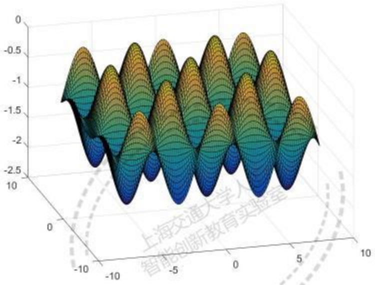
- DBSCAN 算法的运行速度要比 KMeans 算法慢一些。
- DBSCAN 算法的两个参数也是要根据具体的数据情况进行多次尝试。
- 对于具有不同密度的簇，DBSCAN 算法的效果可能不是很好。

DBSCAN与K-means算法

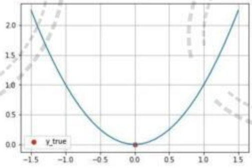


www.naftaliharris.com/blog/visualizing-dbscan-clustering/

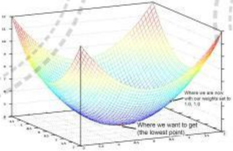
第二节 寻优算法



寻优算法—什么是寻优？



一维寻优



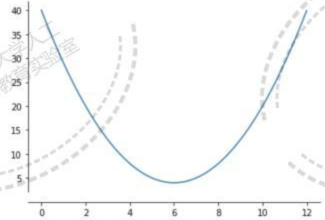
二维寻优

寻优算法的任务，就是找到一个函数的最小值

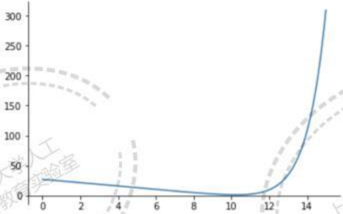
计算最小成本、最短路径、最小损失……

寻优算法—为什么要算法？

$$min\, f(x): x^2 - 6x + 10$$



$$\min f(x): \frac{e^x}{1000} + \frac{x^2}{10} - 3x + 27$$

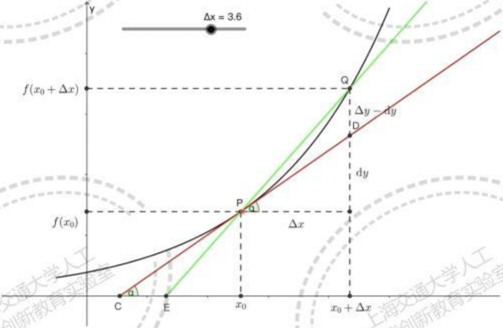


如何找到一个函数的最小值？

寻优算法——什么是梯度？

导数定义

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



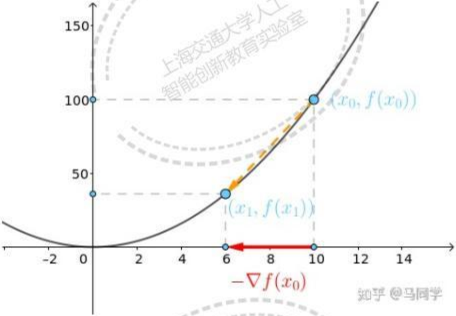
梯度定义

$$\nabla f(x, y)$$

$$= \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right)$$

$$= \frac{\partial f}{\partial x} \cdot i + \frac{\partial f}{\partial y} \cdot j$$

寻优算法—梯度下降法



梯度下降法就是通过沿着负梯度方向不断迭代，找到梯度为0的位置

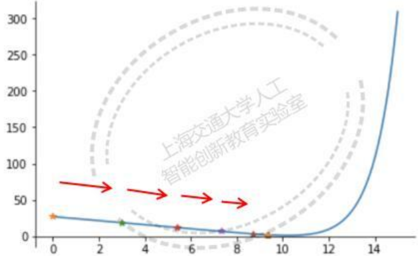
迭代公式$x_{n+1} = x_n - f'(x_n)$

寻优算法—梯度下降法

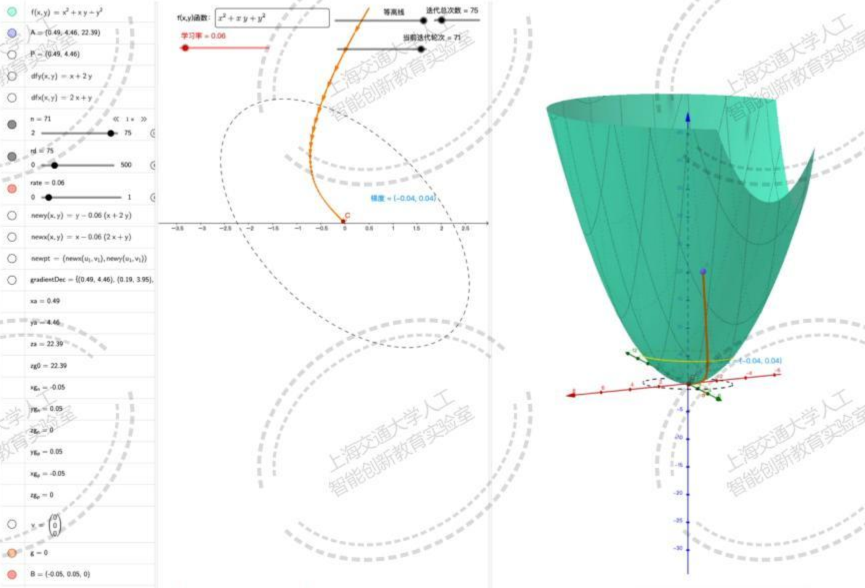
$\min f(x): \frac{e^x}{1000} + \frac{x^2}{10} - 3x + 27$

梯度下降算法步骤：

1. 给定起始点x_0计算导数$f'(x_0)$
2. 迭代：$x_{n+1} = x_n - f'(x_n)$
3. 重复2直至$f'(x_n)$的绝对值足够小



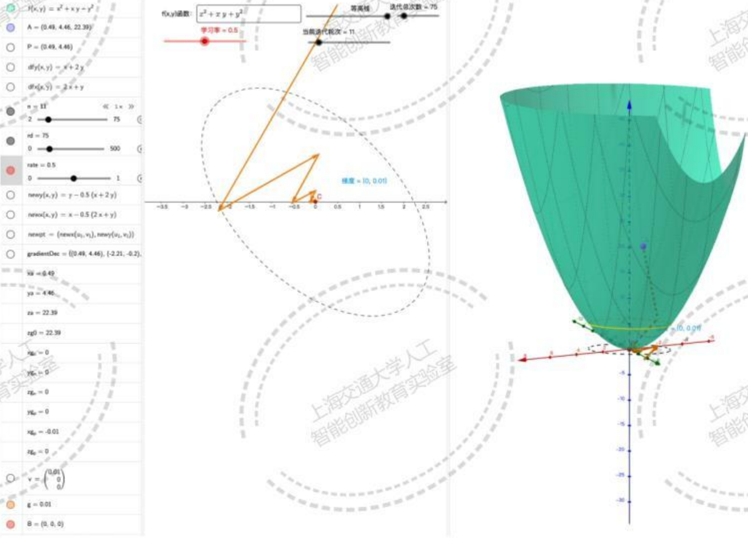
寻优算法—梯度下降法



www.geogebra.org/m/pmcmahyd

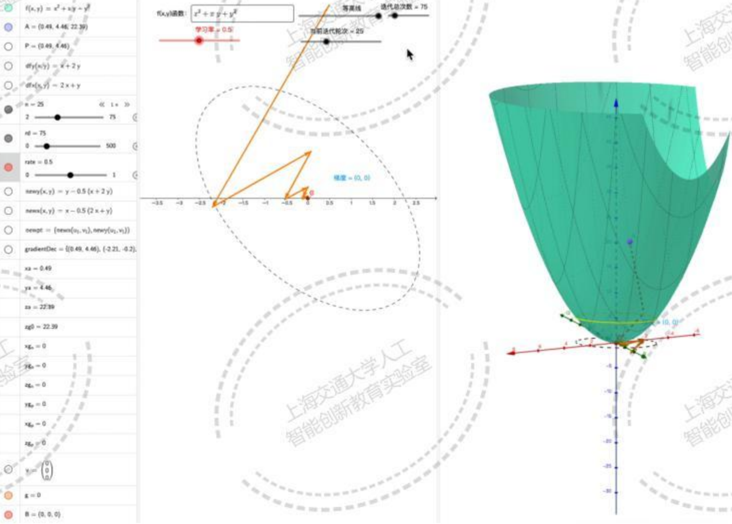
寻优算法—梯度下降法

增大步长（学习率）的影响：震荡且收敛至最低点



寻优算法—梯度下降法

增大步长（学习率）的影响：震荡且无法收敛

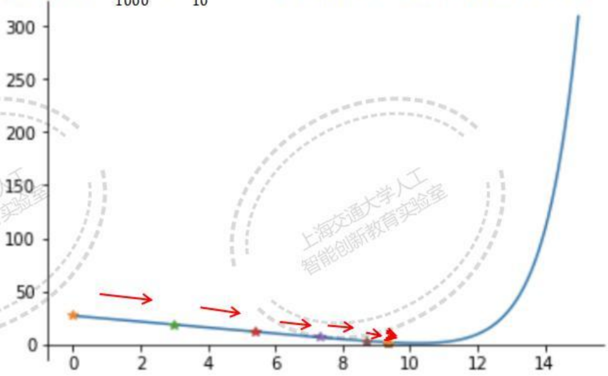


寻优算法—梯度下降法

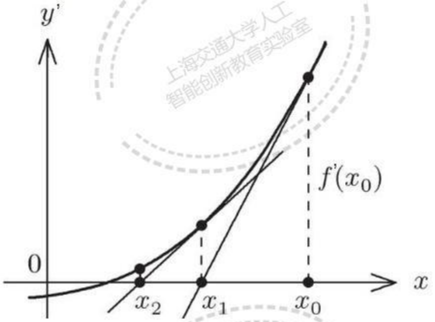
梯度下降算法的**优点**：计算简单

梯度下降算法的**缺点**：梯度很小时，迭代过于缓慢！

- min $f(x)$：$\frac{e^x}{1000} + \frac{x^2}{10} - 3x + 27$ 需要迭代21次到误差小于10^{-9}
- min $(f(x)$：$\frac{e^x}{1000} + \frac{x^2}{10} - 3x + 27)$ / 100 需要迭代1684次到误差小于10^{-9}！



寻优算法—牛顿法



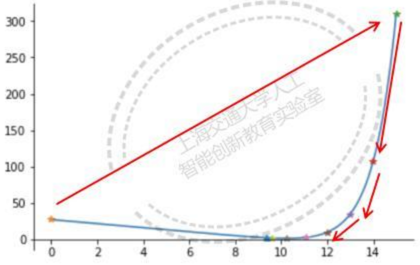
- 牛顿法就把求解函数最小值问题，转化为求解导数零点的问题
- 通过对导函数不断做切线，并找到切线与x轴交点，找到下一个迭代位置
- 迭代公式$x_{n+1} = x_n - f'(x_n)/f''(x_n)$

寻优算法—牛顿法

$$\min f(x): \frac{e^x}{1000} + \frac{x^2}{10} - 3x + 27$$

牛顿法步骤：

1. 给定起始点x_0计算导数$f'(x_0)$，二阶导$f''(x_0)$
2. 迭代$x_{n+1} = x_n - f'(x_n)/f''(x_n)$
3. 重复2直至$f'(x_n)$的绝对值足够小

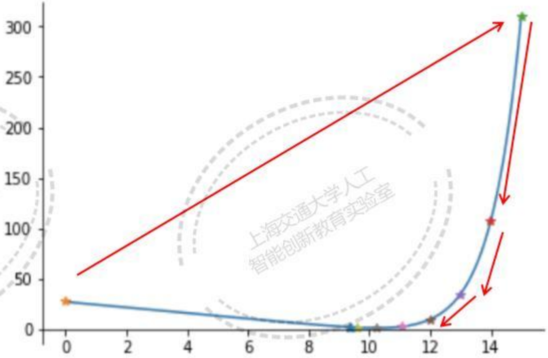


寻优算法—牛顿法

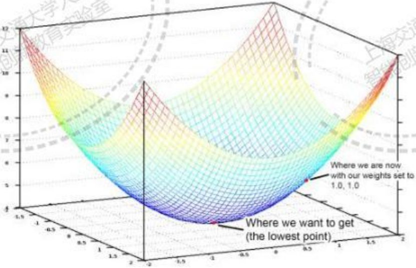
牛顿法的**缺点**：计算比较麻烦，需要算二阶导甚至H矩阵

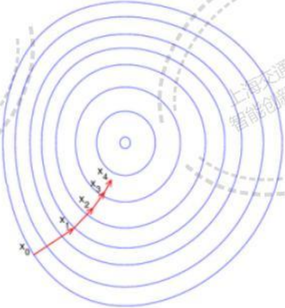
牛顿法的**优点**：能快速通过梯度平稳区域！

- min $f(x): \frac{e^x}{1000} + \frac{x^2}{10} - 3x + 27$ 需要迭代10次到误差小于10^{-9}
- min $(f(x): \frac{e^x}{1000} + \frac{x^2}{10} - 3x + 27)/100$ 需要迭代9次到误差小于10^{-9}！



寻优算法—多维寻优





- 梯度下降法：$X_{n+1} = X_n - \nabla f(X_n)$ ；
- 牛顿法：$X_{n+1} = X_n - H\left(f(X_n)\right)^{-1}\nabla f(X_n)$
- 多维寻优问题，寻优算法不但可以提供迭代的距离，也能提供前进的方向！


谢谢聆听
THANKS FOR YOUR ATTENTION