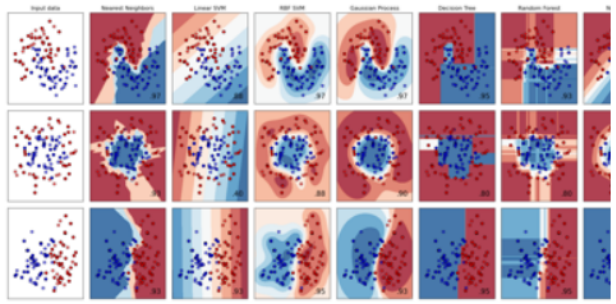
sklearn介绍

sklearn官网：https://scikit-learn.org/stable/

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.
Algorithms: SVM, nearest neighbors, random forest, and more...

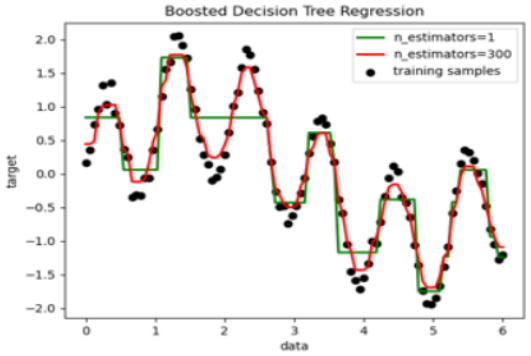


Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: SVR, nearest neighbors, random forest, and more...

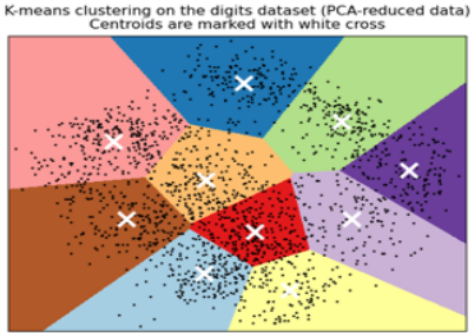


Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, and more...



Examples

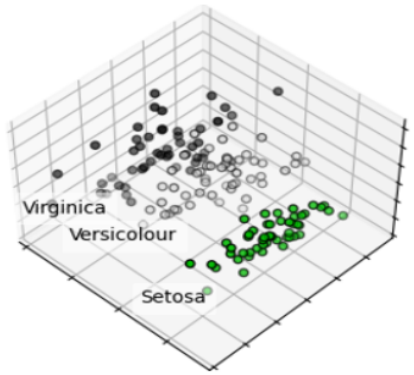
sklearn介绍

sklearn官网：https://scikit-learn.org/stable/

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency
Algorithms: k-Means, feature selection, non-negative matrix factorization, and more...

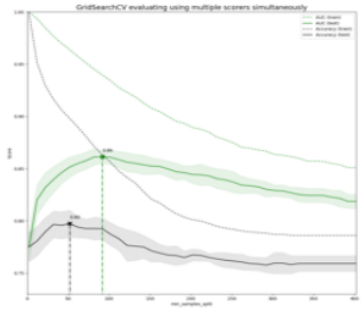


Examples

Model selection

Comparing, validating and choosing parameters and models.

Applications: Improved accuracy via parameter tuning
Algorithms: grid search, cross validation, metrics, and more...

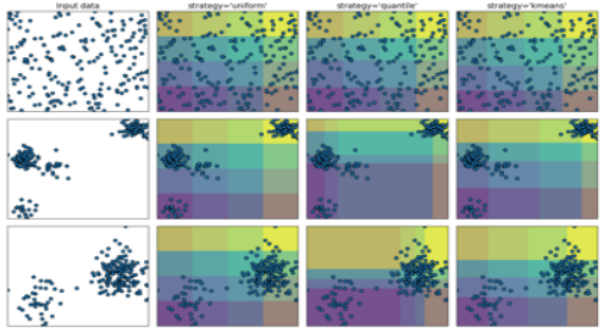


Examples

Preprocessing

Feature extraction and normalization.

Applications: Transforming input data such as text for use with machine learning algorithms.
Algorithms: preprocessing, feature extraction, and more...



Examples

sklearn介绍

scikit-learn 是一个基于 Python 语言的机器学习工具

- 一个简单高效的数据挖掘和数据分析工具
- 构建在 NumPy， SciPy 和 matplotlib 上
- 可供大家在各种环境中重复使用
- 开源，可商业使用 - BSD许可证

安装sklearn

- 安装命令：pip install scikit-learn

- 使用国内镜像源安装：
- pip install scikit-learn -i https://pypi.tuna.tsinghua.edu.cn/simple

sklearn中常用接口

1. 数据集相关
2. 数据预处理相关
3. 常见模型导入
4. 建模、调参相关
5. 模型评估、预测相关

数据集相关

```
#鸢尾花数据集
from sklearn.datasets import load_iris
#乳腺癌数据集
from sklearn.datasets import load_breast_cancer
#人脸识别数据集
from sklearn.datasets import fetch_lfw_people
#随机产生聚类数据集
from sklearn.datasets import make_blobs
```

数据预处理相关

```
#拆分数据集
from sklearn.model_selection import train_test_split
#数据标准化：均值是0，方差为1
from sklearn.preprocessing import StandardScaler
#数据归一化：[0,1]
from sklearn.preprocessing import MinMaxScaler
#数据降维：主成分分析
from sklearn.decomposition import PCA
```

常见模型导入

```
#线性回归
from sklearn.linear_model import LinearRegression
#逻辑回归
from sklearn.linear_model import LogisticRegression
#决策树
from sklearn.tree import DecisionTreeClassifier
#支持向量机
from sklearn.svm import SVC
#随机森林
from sklearn.ensemble import RandomForestClassifier
```

建模、调参相关

```
#拟合训练集
model=LogisticRegression(random_state=0)
model.fit(X_train,y_train)
#交叉验证
from sklearn.model_selection import cross_val_score
#带有交叉验证的网格搜索
from sklearn.model_selection import GridSearchCV
param_grid={'C':[0.001,0.01,0.1,1,10],
            'gamma':[0.01,0.1,1,10,100,]}
grid_search=GridSearchCV(SVC(),param_grid,cv=5)
```

模型评估、预测相关

```
#求精度
knn.score(X_test,y_test)
#绘制混淆矩阵
from sklearn.metrics import confusion_matrix
#绘制ROC曲线、获取AUC值
from sklearn.metrics import roc_curve,roc_auc_score
#预测
y_pred=knn.predict(X_test)
```

案例：应用K近邻算法预测乳腺癌肿瘤

- 样本数据：一些乳腺肿瘤的临床测量数据，样本个数为569个。
- 每个肿瘤有以下10个特征：
  1. 半径（从中心到周边点的距离的平均值）
  2. 纹理（灰度值的标准偏差）
  3. 周边
  4. 面积
  5. 平滑度（半径长度的局部变化）
  6. 紧密度（周长^ 2 /面积-1.0）
  7. 凹度（轮廓的凹入部分的严重程度）
  8. 凹点（轮廓的凹入部分的数量）
  9. 对称性
  10. 分形维数（“海岸线近似”-1）

案例：应用K近邻算法预测乳腺癌肿瘤

- 计算出肿瘤的每个特征的均值（mean）、标准差（standard error）和最差的（worst）
- 也就是说每个特征对应三个数值型的值，所以每个肿瘤有30个特征，均为数值型。
- 最差的表示使肿瘤最糟的某个特征的最大值/最小值的前3个的均值，该特征的值越大/越小，表示肿瘤越糟糕。

- 每个肿瘤有一个类别标签，恶性（Malignant）或者良性（Benign）。

- 任务目标：预测肿瘤是良性还是恶性

代码实现

代码中要做以下几件事情：

1. 数据集导入
2. 将数据集拆分成训练集和测试集
3. 用训练集去训练K近邻算法
4. 根据测试集的精度评估模型
5. 如何选择K值？

线性回归

线性回归

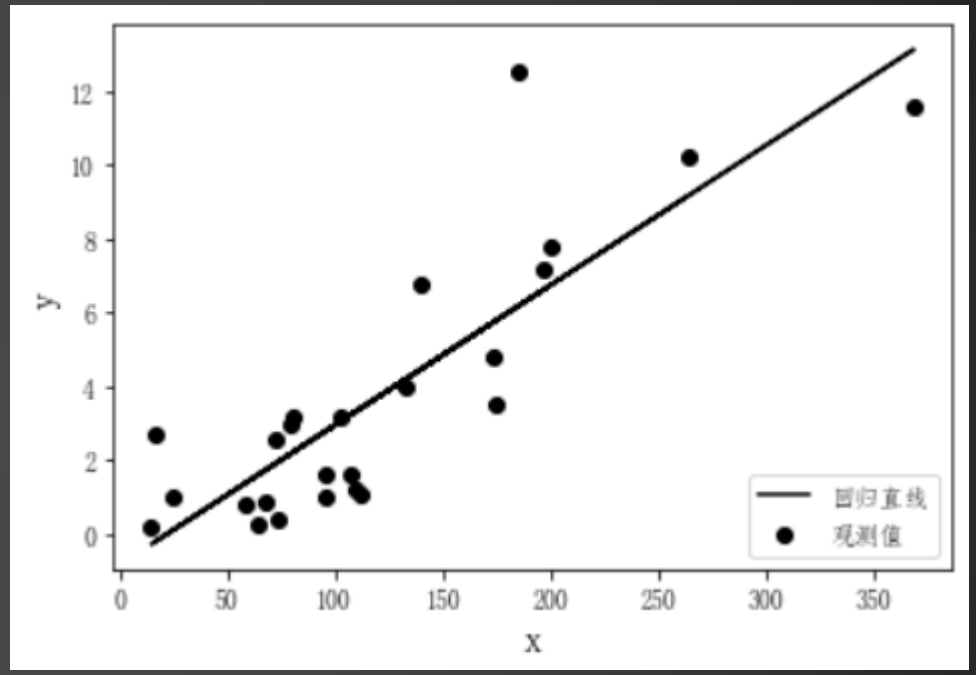
- 线性回归是研究自变量X和因变量y之间的线性关系。
- 自变量：x1，x2，x3，...，均为数值型。
- 因变量：y，为数值型。

- 根据自变量X的个数分为**一元线性回归（一个自变量）**和**多元线性回归（多个自变量）**。

一元线性回归

- 特点：只涉及一个自变量x
- 回归方程：y= $wx + b$
- 一次函数：y=kx+b

- 任务目标：估计参数w, b



一元线性回归示意图

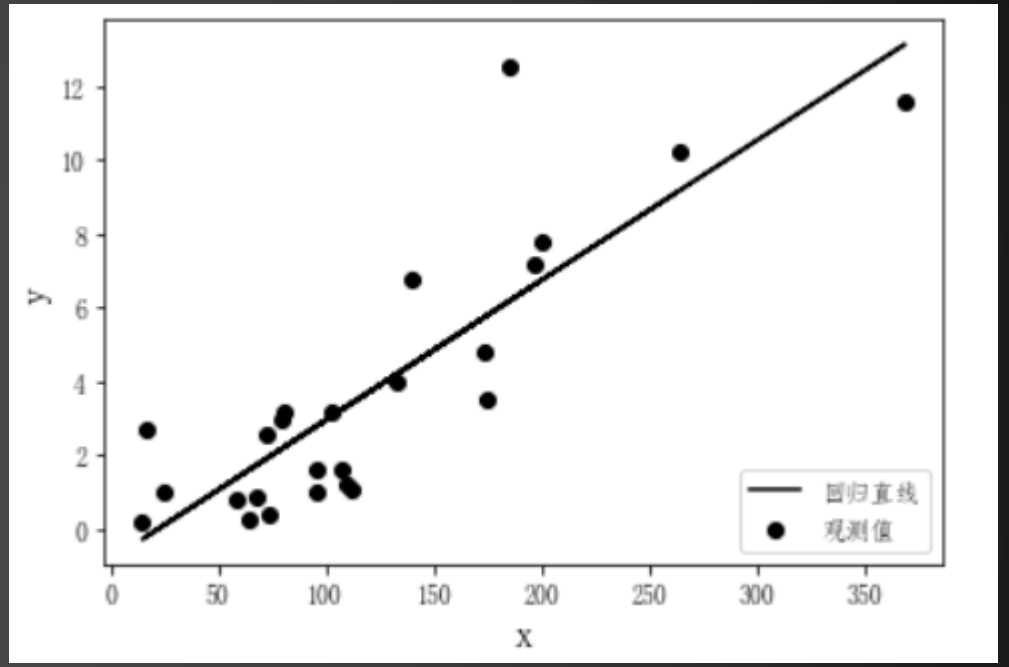
最小二乘法

假定有m个样本，$(x_1, y_1), (x_2, y_2), \cdots, (x_m, y_m)$

最小二乘法的基本思想是通过最小化这些点到直线的总误差来估计参数w和b。

根据最小二乘法，使以下这个式子最小。

$$\sum_{i=1}^{m}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{m}(y_i - wx_i - b)^2$$



这个式子采用的是误差的平方和，加上平方是为了避免正负相抵。

在机器学习中，一般将最小化的目标函数称为损失函数（loss function）或者代价函数（cost function）。

最小二乘法

损失函数（loss function）：

$$l(w,b)=\sum_{i=1}^{m}(y_i-wx_i-b)^2$$

在给定样本数据后，l是w和b的函数，且最小值总是存在。

根据微积分的极值定理，对l求相应于w和b的偏导数，并令其等于0，便可求出w和b,即

$$\begin{cases}\frac{\partial l}{\partial w}=-2\sum_{i=1}^{m}x_i(y_i-wx_i-b)=0\\ \frac{\partial l}{\partial b}=-2\sum_{i=1}^{m}(y_i-wx_i-b)=0\end{cases}$$

解上述方程组得

$$\begin{cases}w=\frac{m\sum_{i=1}^{m}x_iy_i-\sum_{i=1}^{m}x_i\sum_{i=1}^{m}y_i}{m\sum_{i=1}^{m}x_i^2-(\sum_{i=1}^{m}x_i)^2}\\ b=\frac{1}{m}\sum_{i=1}^{m}(y_i-wx_i)\end{cases}$$

大家只要了解原理，一般借助于计算机来计算！！！

评估回归效果

将由回归方程计算出来的y值记为$\hat{y}$，对于每一个实际观测值y_i，其误差的大小用$y_i - \hat{y}_i$来表示，所有观测值的总误差可以通过每个观测值的误差的平方和来表示，即

$$\sum(y_i - \hat{y}_i)^2$$

以上式子可以称为误差平方和，或者残差平方和。

接着对误差做一下变形。

$$y_i - \hat{y}_i = y_i - \overline{y} - (\hat{y}_i - \overline{y})$$

移项得

$$y_i - \overline{y} = (y_i - \hat{y}_i) + (\hat{y}_i - \overline{y})$$

上式本质上是误差分解，不妨将其称为**误差分解式**。

接着，将误差分解式两边平方，并求和，得

$$\sum(y_i-\overline{y})^2=\sum[(y_i-\hat{y}_i)+(\hat{y}_i-\overline{y})]^2$$

将右边平方展开，得

$$\sum(y_i-\overline{y})^2=\sum(y_i-\hat{y}_i)^2+\sum(\hat{y}_i-\overline{y})^2+2\sum(y_i-\hat{y}_i)(\hat{y}_i-\overline{y})$$

上式中，可以证明$\sum(y_i-\hat{y}_i)(\hat{y}_i-\overline{y})=0$

说明：证明时，需要将$\hat{y}_i=wx+b$代入上式中，前面已经求得参数w和b的值。

于是，有

$$\sum(y_i-\overline{y})^2=\sum(y_i-\hat{y}_i)^2+\sum(\hat{y}_i-\overline{y})^2$$

评估回归效果

$$\sum(y_i-\overline{y})^2=\sum(y_i-\hat{y}_i)^2+\sum(\hat{y}_i-\overline{y})^2$$

在上式中，左边称为**总平方和**，右边第一项称为**误差平方和**，右边第二项称为**回归平方和**，我们的目标是希望**误差平方和**尽可能小。

在总平方和一定的情况下，回归平方和越大，误差平方和就越小，所以可以借助于回归平方和占总平方和的比例来评估回归方程的好坏，我们将这个比例称之为**判定系数**，记为R^2，其表达式为：

$$R^2=\frac{\sum(y_i-\hat{y}_i)^2}{\sum(y_i-\overline{y})^2}=1-\frac{\sum(\hat{y}_i-\overline{y})^2}{\sum(y_i-\overline{y})^2}$$

可以看出，判定系数是一个介于0到1之间的数，判定系数越接近于1，说明回归方程拟合效果越好。

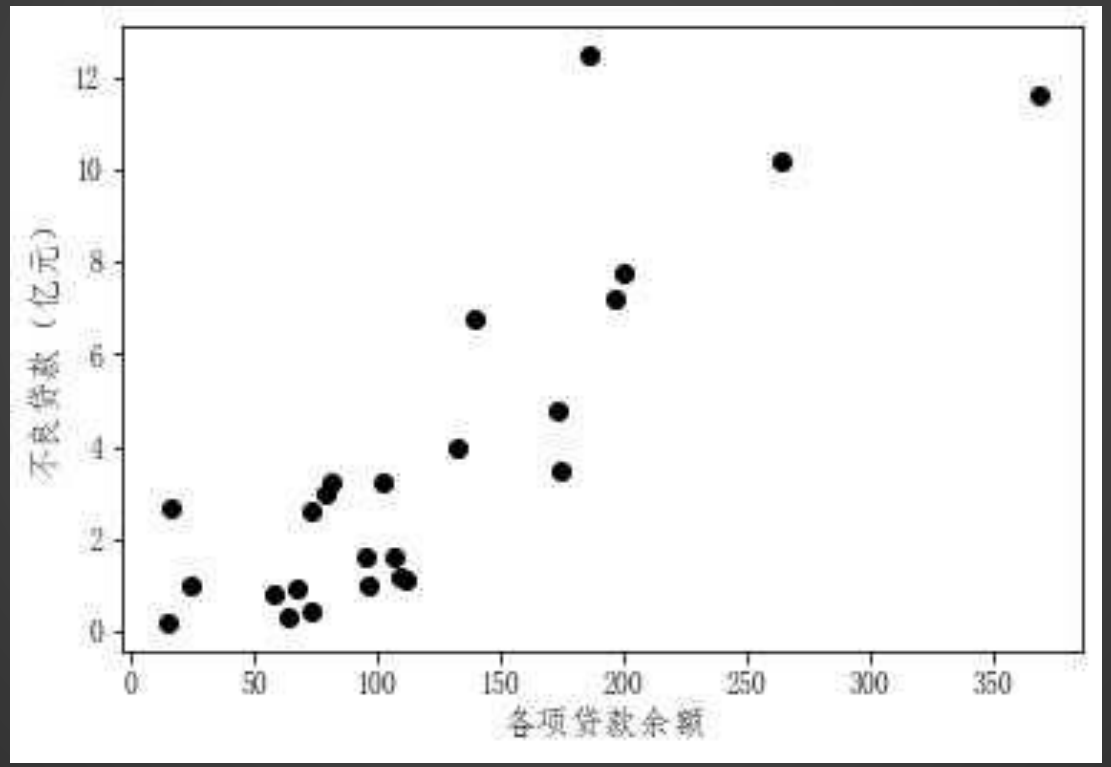
案例：分析不良贷款形成的原因

- 一家大型商业银行在多个地区有25家分行，其业务主要是进行基础设施建设、国家重点项目建设、固定资产投资等项目的贷款。
- 下面是该银行所属的25家分行2002年的有关业务数据（前10条）。

分行编号	不良贷款（亿元）	各项贷款余额	本年累计应收贷款（亿元）	贷款项目个数	本年固定资产投资额（亿元）
1	0.9	67.3	6.8	5	51.9
2	1.1	111.3	19.8	16	90.9
3	4.8	173.0	7.7	17	73.7
4	3.2	80.8	7.2	10	14.5
5	7.8	199.7	16.5	19	63.2
6	2.7	16.2	2.2	1	2.2
7	1.6	107.4	10.7	17	20.2
8	12.5	185.4	27.1	18	43.8
9	1.0	96.1	1.7	10	55.9
10	2.6	72.8	9.1	14	64.3

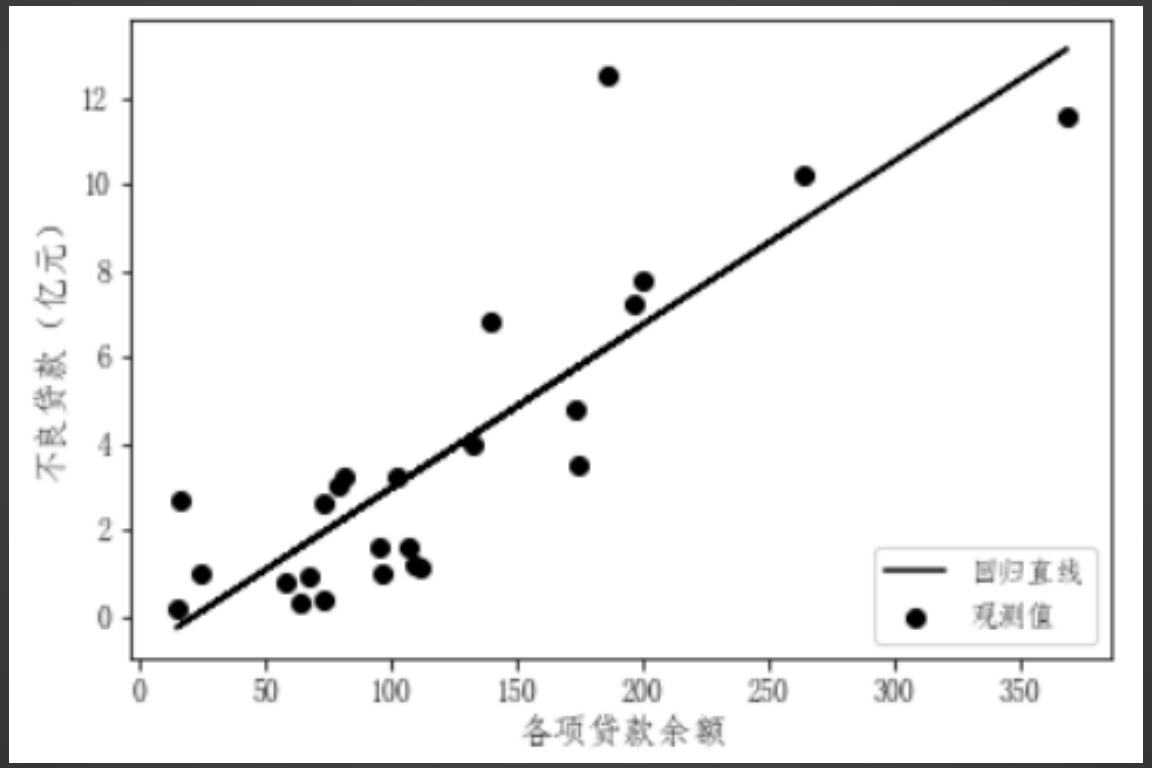
案例：分析不良贷款形成的原因

- 选择各项贷款余额为自变量x，不良贷款为因变量y，绘制x与y的散点图。



拟合直线

- 接下来需要找到一条直线（回归直线）来拟合这些数据点。



假设回归直线方程为：y= $wx + b$，估计参数w, b。

一元线性回归：Python实现

1. 数据读取
2. 绘制散点图
3. 利用sklearn建立回归模型
4. 得到回归方程，并进行预测
5. 模型评估：判定系数

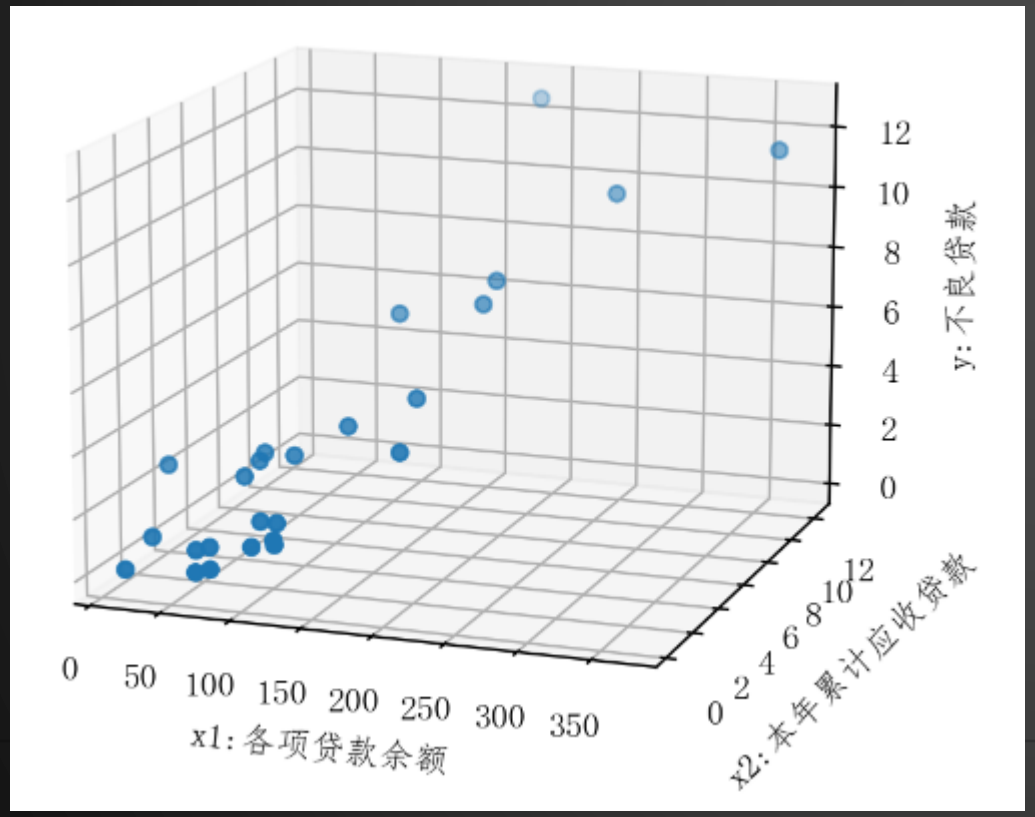
多元线性回归

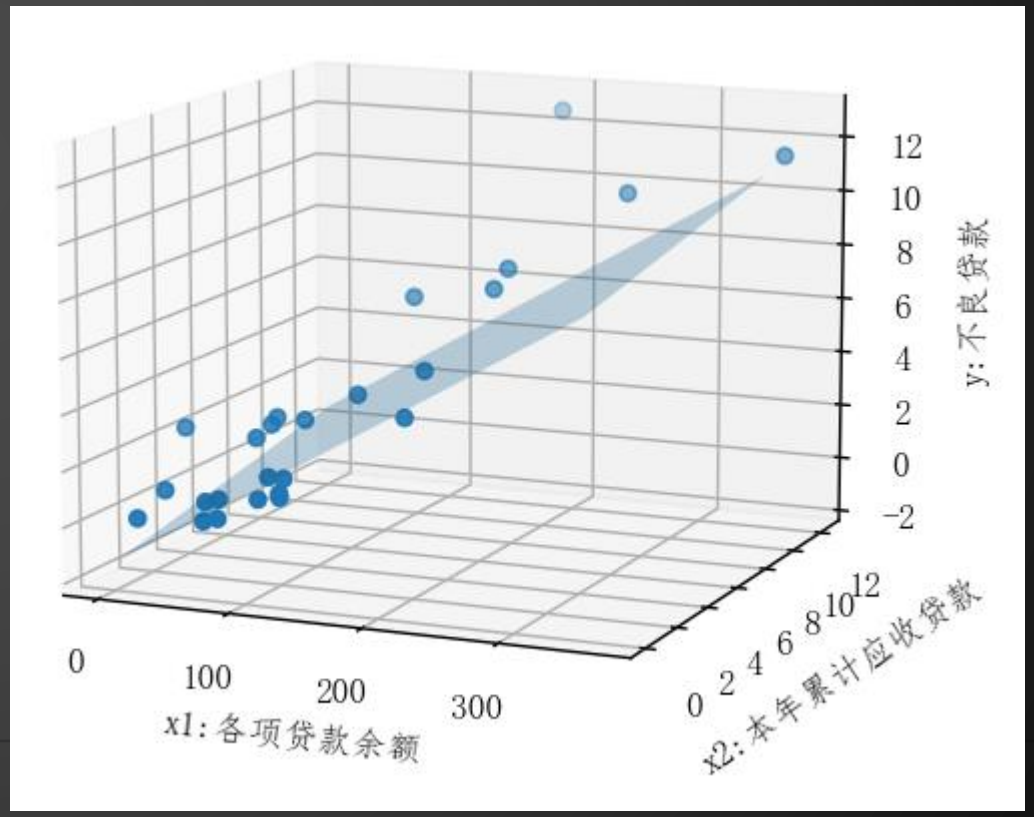
- 特点：涉及多个自变量，x1，x2，x3，.....
- 回归方程：$y = w_1x_1 + w_2x_2$+...+ w_nx_n+b

- 任务目标：估计参数$w_1, w_2, \cdots, w_n$, b

二元线性回归示意图

- 假设有两个自变量，x1，x2，加上一个因变量y，可以绘制一个三维散点图。
- 此时，回归方程：$y = w_1 x_1 + w_2 x_2 + b$，是一个平面，用这个平面去拟合这些样本点。





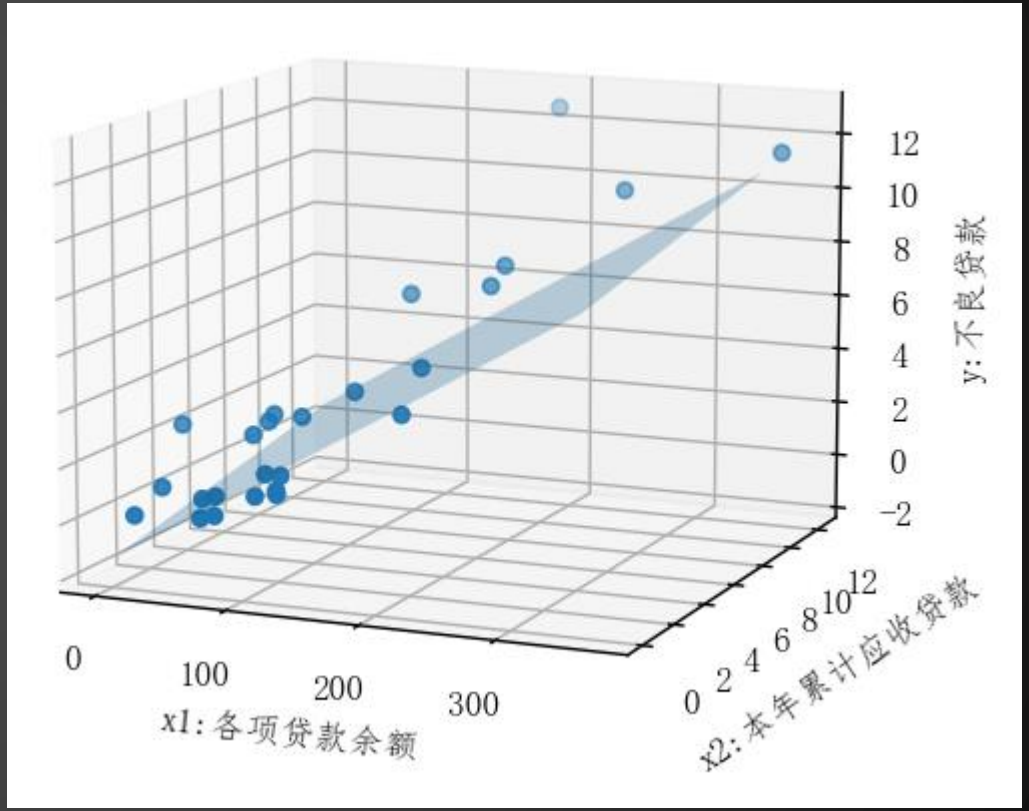
二元线性回归的最小二乘法

假定有m个样本，$\left(x_1^{(1)}, x_1^{(2)}, y_1\right), \left(x_2^{(1)}, x_2^{(2)}, y_2\right), \cdots, \left(x_m^{(1)}, x_m^{(2)}, y_m\right)$

最小二乘法的基本思想是通过最小化这些点到拟合平面的总误差来估计参数w_1, w_2, b

根据最小二乘法，使以下这个式子最小。

$$\sum_{i=1}^{m}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{m}(y_i - w_1 x_i^{(1)} - w_2 x_i^{(2)} - b)^2$$



在机器学习中，一般将最小化的目标函数称为损失函数（loss function）或者代价函数（cost function）。

多元线性回归的最小二乘法

假定有m个样本，$(x_1^{(1)}, x_1^{(2)}, \cdots, x_1^{(n)}, y_1), (x_2^{(1)}, x_2^{(2)}, \cdots, x_2^{(n)}, y_2), \cdots, (x_m^{(1)}, x_m^{(2)}, \cdots, x_m^{(n)}, y_m)$。

损失函数：

$$l(w_1, w_2, ..., w_n, b) = \sum_{i=1}^{m}(y_i - w_1x_1 - w_2x_2 - ... - w_nx_n - b)^2$$

注意：n个参数，m个样本。

根据微积分的极值定理，对l求相应于$w_1, w_2, ..., w_n$和b的偏导数，并令其等于0，即

$$\begin{cases} \frac{\partial l}{\partial w_i} = -2\sum_{i=1}^{m} x_i(y_i - w_1x_i^{(1)} - w_2x_i^{(2)} - \cdots - w_nx_i^{(n)} - b) = 0, i = 1, 2, ..., n \\ \frac{\partial l}{\partial b} = -2\sum_{i=1}^{m}(y_i - w_1x_i^{(1)} - w_2x_i^{(2)} - \cdots - w_nx_i^{(n)} - b) = 0 \end{cases}$$

解上述方程组可以得到各参数的值。

大家只要了解原理，一般借助于计算机来计算！！！

评估回归效果

在一元线性回归中，我们用回归平方和占总平方和的比例来评估回归方程的好坏，我们将这个比例称之为**判定系数**，记为R^2，其表达式为：

$$R^2 = \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \overline{y})^2} = 1 - \frac{\sum(\hat{y}_i - \overline{y})^2}{\sum(y_i - \overline{y})^2}$$

在多元线性回归中，也可以推导出一个类似的公式，我们将其称为**多重判定系数。**

多重判定系数也是一个介于0到1之间的数，判定系数越接近于1，说明回归方程拟合效果越好。

案例：分析不良贷款形成的原因

- 一家大型商业银行在多个地区有25家分行，其业务主要是进行基础设施建设、国家重点项目建设、固定资产投资等项目的贷款。
- 下面是该银行所属的25家分行2002年的有关业务数据（前10条）。

分行编号	不良贷款（亿元）	各项贷款余额	本年累计应收贷款（亿元）	贷款项目个数	本年固定资产投资额（亿元）
1	0.9	67.3	6.8	5	51.9
2	1.1	111.3	19.8	16	90.9
3	4.8	173.0	7.7	17	73.7
4	3.2	80.8	7.2	10	14.5
5	7.8	199.7	16.5	19	63.2
6	2.7	16.2	2.2	1	2.2
7	1.6	107.4	10.7	17	20.2
8	12.5	185.4	27.1	18	43.8
9	1.0	96.1	1.7	10	55.9
10	2.6	72.8	9.1	14	64.3

案例：分析不良贷款形成的原因

- 自变量
  - x1：各项贷款余额
  - x2：本年累计应收贷款
  - x3：贷款项目个数
  - x4：本年固定资产投资额
- 因变量
  - y：不良贷款

多元线性回归：Python实现

1. 数据读取
2. 建立回归模型
3. 得到回归方程，并进行预测
4. 模型评估：多重判定系数

梯度下降

从最小二乘法到梯度下降

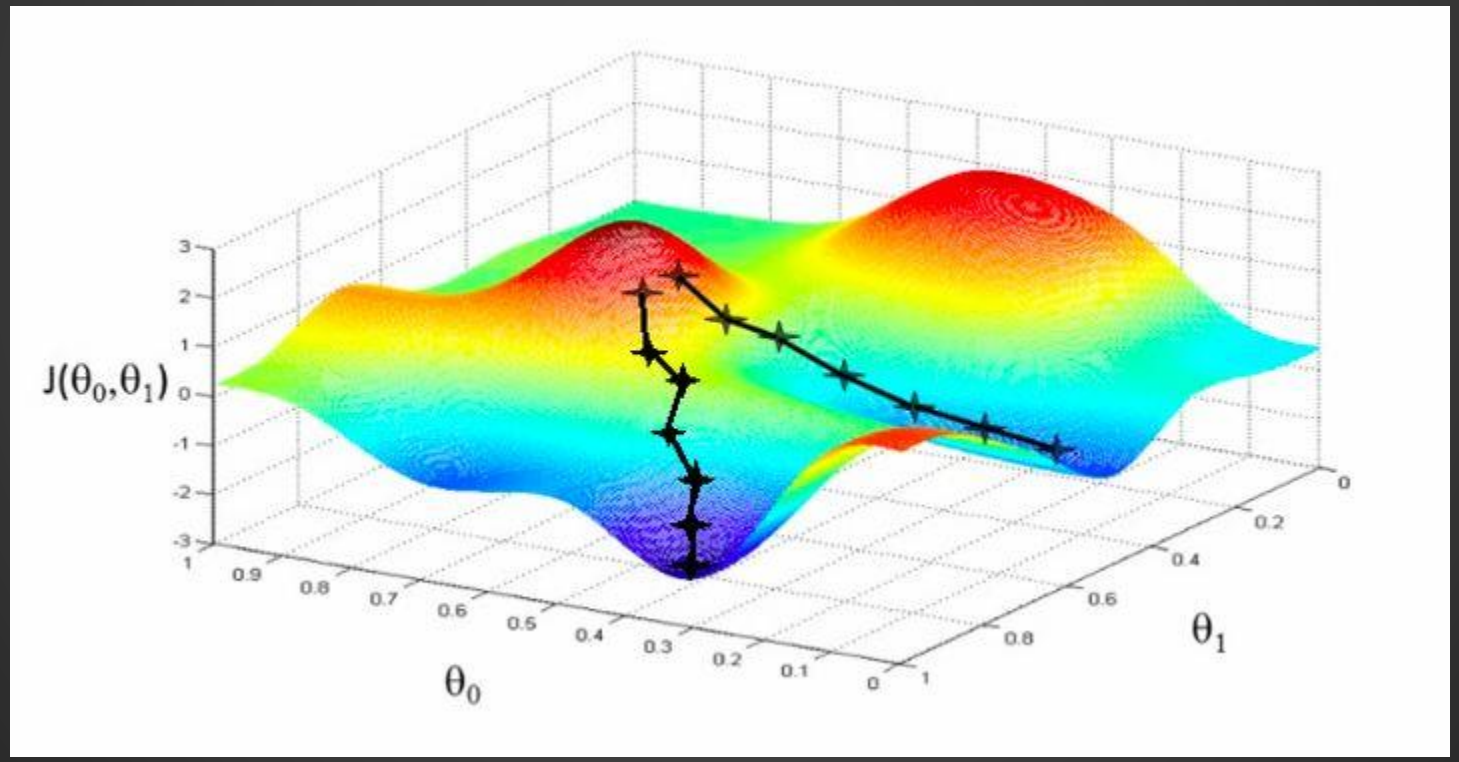
- 回顾前面的线性回归，我们采用最小二乘法，通过对参数求偏导数，列出方程组，解方程组，得到参数的值。这种通过方程直接计算出来的解被称为解析解。

- 在数据量不多，且样本的特征不多的情况下，可以直接用最小二乘法求最小值的解析解。但对于很多实际问题来说，数据量大、样本特征多，用最小二乘法求解，计算量会非常大，只能通过迭代的方式去求一个接近于实际最小值的点。

- 而梯度下降就是这种迭代式求最小值的方法中应用最为广泛的一种方法。

梯度下降法的基本思想

- 梯度下降法的基本思想可以类比为一个下山的过程。



下山类比梯度下降

下山	梯度下降
山	一个函数的图像
山上的某个位置	函数上的一个点
某个位置最陡峭的方向（山的海拔增加最快）	函数的梯度（函数值增加最快）
朝着陡峭的方向下降的方向	梯度的反方向，梯度下降的方向
山的最低点	函数的最小值

什么是梯度?

设函数z=f(x,y)在区域D上具有连续的一阶偏导数，则有梯度的计算公式：

$$\mathbf{grad}f(x,y)=\frac{\partial f}{\partial x}i+\frac{\partial f}{\partial y}j \qquad 或 \qquad \nabla f(x,y)=\frac{\partial f}{\partial x}i+\frac{\partial f}{\partial y}j$$

其中，i是沿x轴的单位向量(1,0)，j是沿y轴的单位向量(0,1)。

还可以将梯度写成如下形式：

$$\mathbf{grad}f(x,y)=(\frac{\partial f}{\partial x},\frac{\partial f}{\partial y})$$

梯度本质上是一个向量。

什么是梯度?

梯度的数学表达式:

$$\mathbf{grad}f(x,y) = (\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y})$$

- 梯度：函数增加（上升）最快的方向。沿着梯度的方向，可以找到这个函数的最大值。
- 在机器学习中，损失函数（或者代价函数）的值代表误差，误差肯定是越小越好，也就是要求损失函数的最小值。
- 梯度下降：函数减少（下降）最快的方向。沿着梯度下降的方向，就可以找到一个函数的最小值。

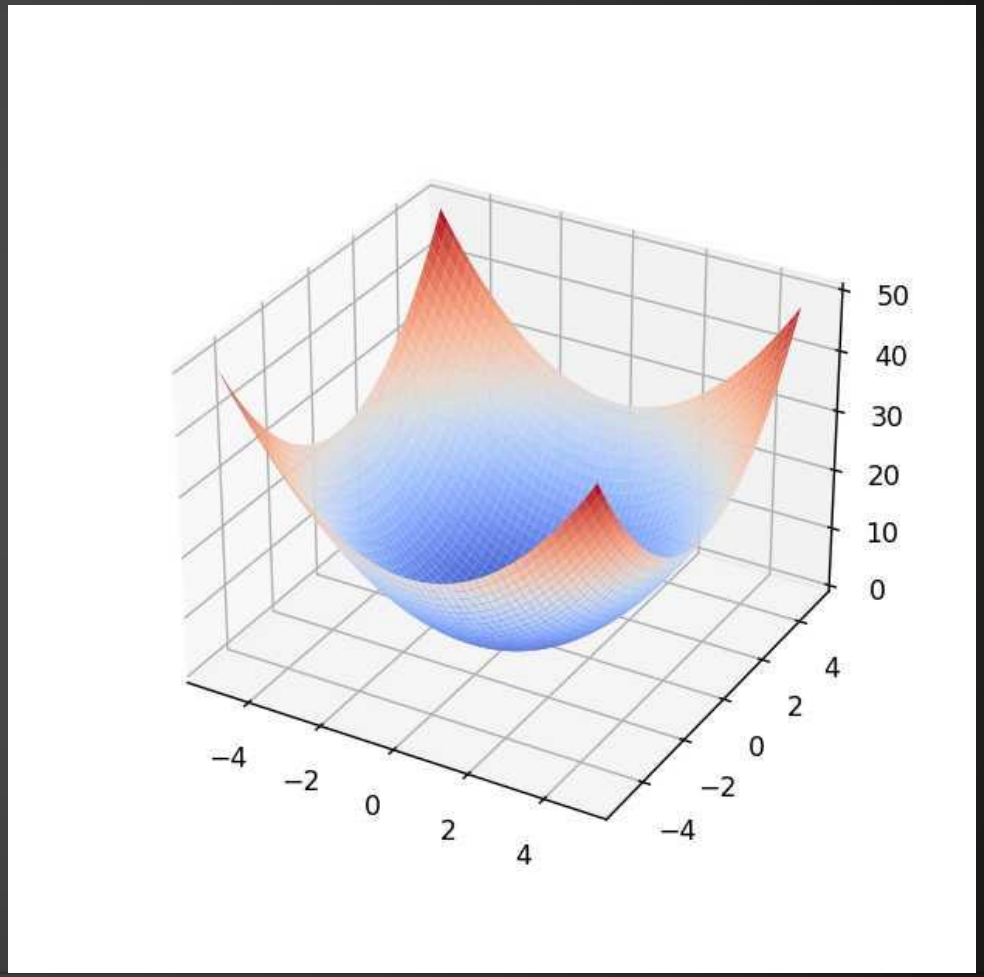
举个例子：二元函数的梯度

- 例如，给定一个二元函数，

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

- 该函数的梯度函数为：

$$\mathbf{grad} f(x,y) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right) = (2x, 2y)$$



梯度下降基本步骤

1. 首先，我们有一个可导函数（损失函数），现在的目标就是找到这个函数的最小值，也就是山底。
2. 根据之前的场景假设，最快的下山的方式就是找到当前位置最陡峭的方向，然后沿着此方向向下走(梯度下降)，就能让函数值下降得最快！
3. 所以，我们重复利用这个方法，反复求取梯度，最后就能到山底，即获得函数的最小值，这就类似于我们下山的过程。

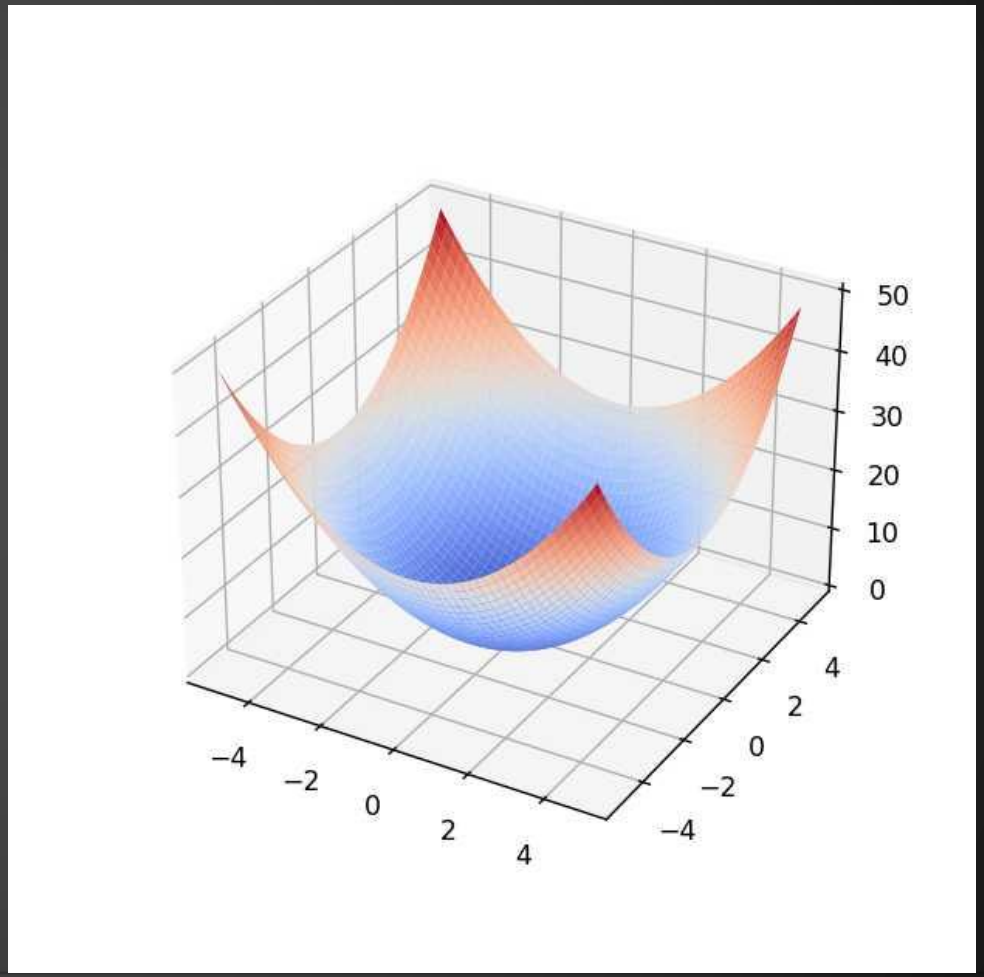
梯度下降实例

- 给定目标函数（损失函数）：

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

- 该函数的梯度函数为：

$$\mathbf{grad}f(x,y)=\left(\frac{\partial f}{\partial x},\frac{\partial f}{\partial y}\right)=(2x,2y)$$



梯度下降实例

- 给定一个初始点、学习率（也叫步长）
- 初始点：梯度下降开始的点；学习率：用来控制每一步下降的距离，每一步下降的距离为 $0.01 \times \mathbf{grad}\ f(x,y)$
- 假设初始点选为$X_0 = (2,4)$，学习率：0.01
- 在初始点，函数值$f(2,4) = 20$，梯度为：$\mathbf{grad}\ f(2,4) = (4,8)$

- 接着，沿着梯度下降的方向走。
- 第1步：$X_1 = X_0 - 0.01 \times \mathbf{grad}\ f(2,4) = X_0 - 0.01 \times (4,8) = (1.96,3.92)$
- 说明：以上负号（减号）表示梯度下降。
- 此时，函数值$f(1.96,3.92) = 19.21$

梯度下降实例

- 第2步：$X_2 = X_1 - 0.01 \times \mathbf{grad}\, f(1.96,3.92) = X_1 - 0.01 \times (3.92,7.84) = (1.92,3.84)$

此时，函数值$f(1.92,3.84) = 18.45$

- 第3步：$X_3 = X_2 - 0.01 \times \mathbf{grad}\, f(1.92,3.84) = X_2 - 0.01 \times (3.84,7.68) = (1.88,3.76)$
- 此时，函数值$f(1.88,3.76) = 17.72$

继续……

- 第100步：，此时，已经下降到了点$(0.27, 0.53)$，此时，函数值$f(0.27, 0.53) = 0.35$

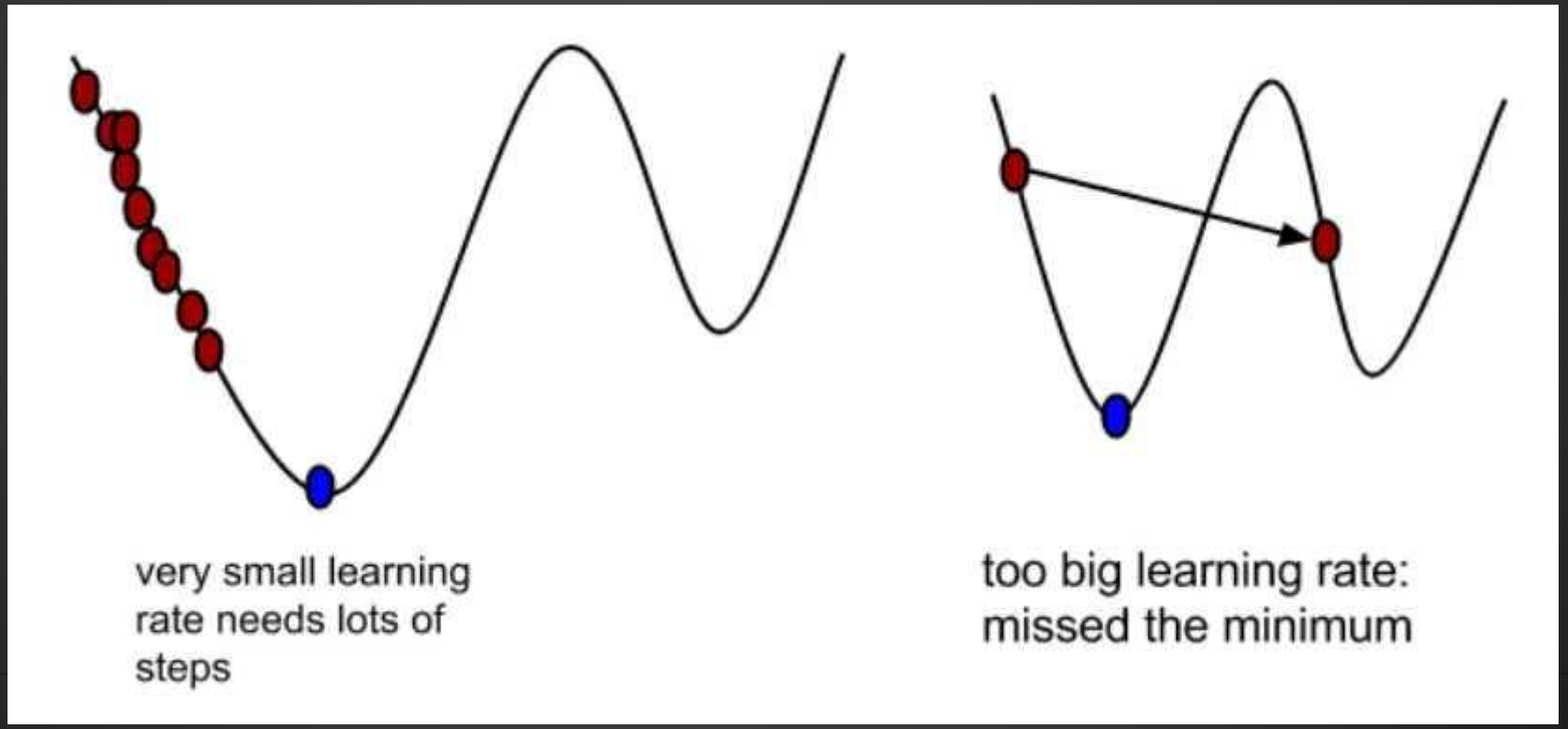
可以看到，当下降100步的时候，函数值已经很小了。

- 第200步：函数值为0.006，已经非常接近真实最小值0了，此时误差小于0.01。

这就是用梯度下降求解这个二元函数的一个基本过程。

关于学习率的说明

- 学习率对于下降的速度有着很大的影响，有人可能会说，把学习率设置得大一点，不就能很快下降到最小值了吗？俗话说，“步子太大，容易扯着蛋”，如果学习率过大，可能会错过最低点，见下图。



梯度下降：Python实现

1. 定义目标函数（损失函数）
2. 绘制函数图像
3. 定义梯度函数
4. 给定初始值、学习率，并进行3次梯度下降
5. 梯度下降：多次迭代
6. 绘制损失曲线图

梯度下降数学推导

对于多元线性回归，假设函数为

$$h_w(x_1, x_2, ..., x_n) = w_1x_1 + w_2x_2 + ... + w_nx_n + b$$

假设有m个样本，$(x_1^{(0)}, x_2^{(0)}, ...x_n^{(0)}, y_0), (x_1^{(1)}, x_2^{(1)}, ...x_n^{(1)}, y_1), ...(x_1^{(m)}, x_2^{(m)}, ...x_n^{(m)}, y_m)$

对于上方的假设函数，损失函数为：

$$l(w_0, w_1, ..., w_n) = \frac{1}{2m}\sum_{j=1}^{m}(h_w(x_0^{(j)}, x_1^{(j)}, ..., x_n^{(j)}) - y_i)^2$$

说明：这里除以m表示平均损失，又除了一个2是为了方便后面的计算，并不影响对于目标函数的优化。

接下来要求这个损失函数的最小值。

梯度下降算法过程：

初始化$w_1, w_2, ..., w_n, b$，步长α。

1. 求损失函数的梯度，对于参数w_i，其梯度表达式为：

   $\frac{\partial}{\partial w_i} l(w_1, w_2, ..., w_n, b) = \frac{1}{m} \sum_{j=0}^{m} (h_w(x_0^{(j)}, x_1^{(j)}, ..., x_n^{(j)}) - y_i) x_i^{(j)}$

2. 用步长乘以损失函数的梯度，得到当前位置下降的距离

   $\alpha \frac{\partial}{\partial w_i} l(w_0, w_1, ..., w_n)$

3. 更新参数，对于参数w_i，其更新表达式为：

   $w_i = w_i - \alpha \frac{\partial}{\partial w_i} l(w_1, w_2, ..., w_n, b) = w_i - \alpha \frac{1}{m} \sum_{j=0}^{m} (h_w(x_0^{(j)}, x_1^{(j)}, ..., x_n^{(j)}) - y_i) x_i^{(j)}$

   说明：因为是梯度下降，所以这里是减号，如果是梯度上升，那就是加号了！

4. 确定是否所有的参数w_i梯度下降的距离都小于ϵ，如果小于ϵ则算法终止，否则进入步骤3

梯度下降算法家族成员

1、批量梯度下降

- 批量梯度下降法是梯度下降法最常用的形式，具体做法也就是在更新参数时使用所有的样本来进行更新，这个方法对应于前面说的线性回归的梯度下降算法。

$$w_i = w_i - \alpha \frac{1}{m} \sum_{j=0}^{m} (h_w(x_0^{(j)}, x_1^{(j)}, ..., x_n^{(j)}) - y_i) x_i^{(j)}$$

2、随机梯度下降

- 随机梯度下降法和批量梯度下降法原理类似，区别在于求梯度时没有用所有的m个样本的数据，而是选取一个样本来求梯度，对应的参数更新公式为：

$$w_i = w_i - \alpha (h_w(x_0^{(j)}, x_1^{(j)}, ..., x_n^{(j)}) - y_i) x_i^{(j)}$$

- 随机梯度下降法和批量梯度下降法是两个极端，一个采用所有数据来梯度下降，一个用一个样本来梯度下降。自然各自的优缺点都非常突出。

梯度下降算法家族成员

3、小批量梯度下降

- 小批量梯度下降法是批量梯度下降法和随机梯度下降法的折衷，也就是对于m个样本，采用x个样子来迭代，1<x<m。一般可以取x=10，当然根据样本的数据，可以调整这个x的值。对应的参数更新公式是：

$$w_i = w_i - \alpha \sum_{j=t}^{t+x-1} (h_w(x_0^{(j)}, x_1^{(j)}, ..., x_n^{(j)}) - y_i) x_i^{(j)}$$