菜菜的scikit-learn课堂05

sklearn中的逻辑回归

小伙伴们晚上好~o(￣▽￣)ブ

我是菜菜，这里是我的sklearn课堂第五期，今晚的直播内容是逻辑回归~

我的开发环境是**Jupyter lab**，所用的库和版本大家参考：

Python 3.7.1（你的版本至少要3.4以上

Scikit-learn 0.20.1 （你的版本至少要0.20

Numpy 1.15.4, **Pandas** 0.23.4, **Matplotlib** 3.0.2, **SciPy** 1.1.0

请扫码进群领取课件和代码源文件，**扫描二维码后回复"K"就可以进群哦~**

菜菜的scikit-learn课堂05

sklearn中的逻辑回归

1 概述

1.1 名为“回归”的分类器

在过去的四周中，我们接触了不少带“回归”二字的算法，回归树，随机森林的回归，无一例外他们都是区别于分类算法们，用来处理和预测连续型标签的算法。然而逻辑回归，是一种名为“回归”的线性分类器，其本质是由线性回归变化而来的，一种广泛使用于分类问题中的广义回归算法。要理解逻辑回归从何而来，得要先理解线性回归。线性回归是机器学习中最简单的的回归算法，它写作一个几乎人人熟悉的方程：

$$z = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$$

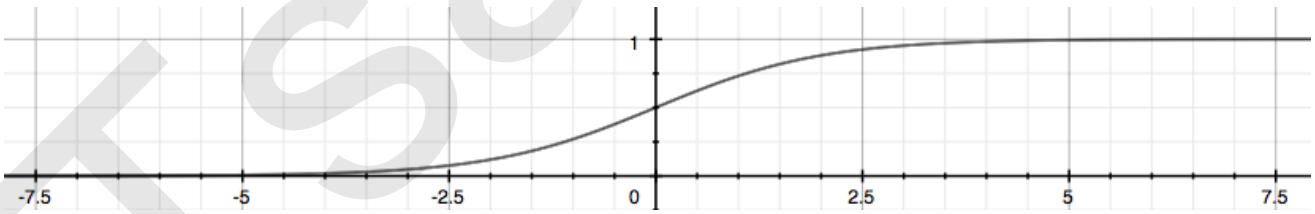
θ被统称为模型的参数，其中θ_0被称为截距(intercept)，θ_1~θ_n被称为系数(coefficient)，这个表达式，其实就和我们小学时就无比熟悉的$y = ax + b$是同样的性质。我们可以使用矩阵来表示这个方程，其中x和θ都可以被看做是一个列矩阵，则有：

$$z = [\theta_0, \theta_1, \theta_2 \ldots \theta_n] * \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \cdots \\ x_n \end{bmatrix} = \boldsymbol{\theta}^T \boldsymbol{x} \ \ (x_0 = 1)$$

线性回归的任务，就是构造一个预测函数z来映射输入的特征矩阵x和标签值y的线性关系，而构造预测函数的核心就是找出模型的参数：θ^T和θ_0，著名的最小二乘法就是用来求解线性回归中参数的数学方法。

通过函数z，线性回归使用输入的特征矩阵X来输出一组连续型的标签值y_pred，以完成各种预测连续型变量的任务（比如预测产品销量，预测股价等等）。那如果我们的标签是离散型变量，尤其是，如果是满足0-1分布的离散型变量，我们要怎么办呢？我们可以通过引入联系函数(link function)，将线性回归方程z变换为g(z)，并且令g(z)的值分布在(0,1)之间，且当g(z)接近0时样本的标签为类别0，当g(z)接近1时样本的标签为类别1，这样就得到了一个分类模型。而这个联系函数对于逻辑回归来说，就是Sigmoid函数：

$$g(z) = \frac{1}{1 + e^{-z}}$$



面试高危问题：Sigmoid函数的公式和性质

Sigmoid函数是一个S型的函数，当自变量z趋近正无穷时，因变量g(z)趋近于1，而当z趋近负无穷时，g(z)趋近于0，它能够将任何实数映射到(0,1)区间，使其可用于将任意值函数转换为更适合二分类的函数。
因为这个性质，Sigmoid函数也被当作是归一化的一种方法，与我们之前学过的MinMaxSclaer同理，是属于数据预处理中的“缩放”功能，可以将数据压缩到[0,1]之内。区别在于，MinMaxScaler归一化之后，是可以取到0和1的（最大值归一化后就是1，最小值归一化后就是0），但Sigmoid函数只是无限趋近于0和1。

线性回归中$z = \boldsymbol{\theta}^T \boldsymbol{x}$，于是我们将$z$带入，就得到了二元逻辑回归模型的一般形式：

$$g(z) = y(x) = \frac{1}{1 + e^{-\boldsymbol{\theta}^T \boldsymbol{x}}}$$

而g(z)就是我们逻辑回归返回的标签值。此时，$y(x)$的取值都在[0,1]之间，因此$y(x)$和$1-y(x)$相加必然为1。如果我们令$y(x)$除以$1-y(x)$可以得到形似几率(odds)的$\frac{y(x)}{1-y(x)}$，在此基础上取对数，可以很容易就得到：

$$
\begin{aligned}
ln\frac{y(x)}{1-y(x)} &= ln(\frac{\frac{1}{1+e^{-\theta^T x}}}{1-\frac{1}{1+e^{-\theta^T x}}}) \\
&= ln(\frac{\frac{1}{1+e^{-\theta^T x}}}{\frac{e^{-\theta^T x}}{1+e^{-\theta^T x}}}) \\
&= ln(\frac{1}{e^{-\boldsymbol{\theta}^T \boldsymbol{x}}}) \\
&= ln(e^{\boldsymbol{\theta}^T \boldsymbol{x}}) \\
&= \boldsymbol{\theta}^T \boldsymbol{x}
\end{aligned}
$$

不难发现，g(z)的形似几率取对数的本质其实就是我们的线性回归z，我们实际上是在对线性回归模型的预测结果取对数几率来让其的结果无限逼近0和1。因此，其对应的模型被称为"对数几率回归"（logistic Regression），也就是我们的逻辑回归，这个名为"回归"却是用来做分类工作的分类器。

思考：g(z)代表了样本为某一类标签的概率吗?

$ln\frac{y(x)}{1-y(x)}$是形似对数几率的一种变化。而几率odds的本质其实是$\frac{p}{1-p}$，其中p是事件A发生的概率，而1-p是事件A不会发生的概率，并且p+(1-p)=1。因此，很多人在理解逻辑回归时，都对g(z)做出如下的解释：

我们让线性回归结果逼近0和1，此时$y(x)$和$1-y(x)$之和为1，因此它们可以被我们看作是一对正反例发生的概率，即$y(x)$是某样本x的标签为1的概率，而$1-y(x)$是x的标签为0的概率，$\frac{y(x)}{1-y(x)}$就是样本x为1的相对概率。基于这种理解，我们使用最大似然法和概率分布函数推到出逻辑回归的损失函数，并且把返回样本在标签取值上的概率当成是逻辑回归的性质来使用，每当我们诉求概率的时候，我们都会使用逻辑回归。

然而这种理解是正确的吗？概率是度量偶然事件发生可能性的数值，尽管逻辑回归的取值在(0,1)之间，并且$y(x)$和$1-y(x)$之和的确为1，但光凭这个性质，我们就可以认为$y(x)$代表了样本x在标签上取值为1的概率吗？设想我们使用MaxMinScaler对特征进行归一化后，任意特征的取值也在[0,1]之间，并且任意特征的取值x_0和$1-x_0$也能够相加为1，但我们却不会认为0-1归一化后的特征是某种概率。**逻辑回归返回了概率**这个命题，这种说法严谨吗?

1.2 为什么需要逻辑回归

线性回归对数据的要求很严格，比如标签必须满足正态分布，特征之间的多重共线性需要消除等等，而现实中很多真实情景的数据无法满足这些要求，因此线性回归在很多现实情境的应用效果有限。逻辑回归是由线性回归变化而来，因此它对数据也有一些要求，而我们之前已经学过了强大的分类模型决策树和随机森林，它们的分类效力很强，并且不需要对数据做任何预处理。

何况，逻辑回归的原理其实并不简单。一个人要理解逻辑回归，必须要有一定的数学基础，必须理解损失函数，正则化，梯度下降，海森矩阵等等这些复杂的概念，才能够对逻辑回归进行调优。其涉及到的数学理念，不比支持向量机少多少。况且，要计算概率，朴素贝叶斯可以计算出真正意义上的概率，要进行分类，机器学习中能够完成二分类功能的模型简直多如牛毛。因此，在数据挖掘，人工智能所涉及到的医疗，教育，人脸识别，语音识别这些领域，逻辑回归没有太多的出场机会。

甚至，在我们的各种机器学习经典书目中，周志华的《机器学习》400页仅有一页纸是关于逻辑回归的（还是一页数学公式），《数据挖掘导论》和《Python数据科学手册》中完全没有逻辑回归相关的内容，sklearn中对比各种分类器的效应也不带逻辑回归玩，可见业界地位。

但是，无论机器学习领域如何折腾，逻辑回归依然是一个受工业商业热爱，使用广泛的模型，因为它有着不可替代的优点：

1. **逻辑回归对线性关系的拟合效果好到丧心病狂**，特征与标签之间的线性关系极强的数据，比如金融领域中的信用卡欺诈，评分卡制作，电商中的营销预测等等相关的数据，都是逻辑回归的强项。虽然现在有了梯度提升树GDBT，比逻辑回归效果更好，也被许多数据咨询公司启用，但逻辑回归在金融领域，尤其是银行业中的统治地位依然不可动摇（相对的，逻辑回归在非线性数据的效果很多时候比瞎猜还不如，所以如果你已经知道数据之间的联系是非线性的，千万不要迷信逻辑回归）
2. **逻辑回归计算快**：对于线性数据，逻辑回归的拟合和计算都非常快，计算效率优于SVM和随机森林，亲测表示在大型数据上尤其能够看得出区别
3. **逻辑回归返回的分类结果不是固定的0，1，而是以小数形式呈现的类概率数字**：我们因此可以把逻辑回归返回的结果当成连续型数据来利用。比如在评分卡制作时，我们不仅需要判断客户是否会违约，还需要给出确定的”信用分“，而这个信用分的计算就需要使用类概率计算出的对数几率，而决策树和随机森林这样的分类器，可以产出分类结果，却无法帮助我们计算分数（当然，在sklearn中，决策树也可以产生概率，使用接口predict_proba调用就好，但一般来说，正常的决策树没有这个功能）。

另外，逻辑回归还有抗噪能力强的优点。福布斯杂志在讨论逻辑回归的优点时，甚至有着”技术上来说，最佳模型的AUC面积低于0.8时，逻辑回归非常明显优于树模型”的说法。并且，逻辑回归在小数据集上表现更好，在大型的数据集上，树模型有着更好的表现。

由此，我们已经了解了逻辑回归的本质，它是一个返回对数几率的，在线性数据上表现优异的分类器，它主要被应用在金融领域。**其数学目的是求解能够让模型最优化的参数θ的值，并基于参数θ和特征矩阵计算出逻辑回归的结果y(x)。**注意：虽然我们熟悉的逻辑回归通常被用于处理二分类问题，但逻辑回归也可以做多分类。

1.3 sklearn中的逻辑回归

逻辑回归相关的类	说明
linear_model.LogisticRegression	逻辑回归回归分类器（又叫logit回归，最大熵分类器）
linear_model.LogisticRegressionCV	带交叉验证的逻辑回归分类器
linear_model.logistic_regression_path	计算Logistic回归模型以获得正则化参数的列表
linear_model.SGDClassifier	利用梯度下降求解的线性分类器（SVM，逻辑回归等等）
linear_model.SGDRegressor	利用梯度下降最小化正则化后的损失函数的线性回归模型
metrics.log_loss	对数损失，又称逻辑损失或交叉熵损失
【在sklearn0.21版本中即将被移除】	
linear_model.RandomizedLogisticRegression	随机的逻辑回归
其他会涉及的类	**说明**
metrics.confusion_matrix	混淆矩阵，模型评估指标之一
metrics.roc_auc_score	ROC曲线，模型评估指标之一
metrics.accuracy_score	精确性，模型评估指标之一

2 linear_model.LogisticRegression

class `sklearn.linear_model.LogisticRegression` (*penalty='l2'*, *dual=False*, *tol=0.0001*, *C=1.0*, *fit_intercept=True*, *intercept_scaling=1*, *class_weight=None*, *random_state=None*, *solver='warn'*, *max_iter=100*, *multi_class='warn'*, *verbose=0*, *warm_start=False*, *n_jobs=None*)

2.1 二元逻辑回归的损失函数

在学习决策树和随机森林时，我们曾经提到过两种模型表现：在训练集上的表现，和在测试集上的表现。我们建模，是追求模型在测试集上的表现最优，因此模型的评估指标往往是用来衡量模型在测试集上的表现的。然而，逻辑回归有着基于训练数据求解参数θ的需求，并且希望训练出来的模型能够尽可能地拟合训练数据，即模型在训练集上的预测准确率越靠近100%越好。

因此，我们使用"**损失函数**"这个评估指标，来**衡量参数θ的优劣，即这一组参数能否使模型在训练集上表现优异。**如果用一组参数建模后，模型在训练集上表现良好，那我们就说模型表现的规律与训练集数据的规律一致，拟合过程中的损失很小，损失函数的值很小，这一组参数就优秀；相反，如果模型在训练集上表现糟糕，损失函数就会很大，模型就训练不足，效果较差，这一组参数也就比较差。即是说，我们在求解参数θ时，追求损失函数最小，让模型在训练数据上的拟合效果最优，即预测准确率尽量靠近100%。

关键概念：损失函数
衡量参数θ的优劣的评估指标，用来求解最优参数的工具 损失函数小，模型在训练集上表现优异，拟合充分，参数优秀 损失函数大，模型在训练集上表现差劲，拟合不足，参数糟糕 **我们追求，能够让损失函数最小化的参数组合** 注意：没有"求解参数"需求的模型没有损失函数，比如KNN，决策树

逻辑回归的损失函数是由最大似然法来推导出来的，具体结果可以写作：

$$J(\theta) = -\sum_{i=1}^{m}(y_i * log(y_\theta(x_i)) \ + \ (1-y_i) * log(1-y_\theta(x_i)))$$

其中，θ表示求解出来的一组参数，m是样本的个数，y_i是样本i上真实的标签，$y_\theta(x_i)$是样本i上，基于参数θ计算出来的逻辑回归返回值，x_i是样本i的取值。我们的目标，就是求解出使$J(\theta)$最小的θ取值。

由于我们追求损失函数的最小值，让模型在训练集上表现最优，可能会引发另一个问题：如果模型在训练集上表示优秀，却在测试集上表现糟糕，模型就会过拟合。虽然逻辑回归和线性回归是天生欠拟合的模型，但我们还是需要控制过拟合的技术来帮助我们调整模型，**对逻辑回归中过拟合的控制，通过正则化来实现。**

2.2 正则化：重要参数penalty & C

正则化是用来防止模型过拟合的过程，常用的有L1正则化和L2正则化两种选项，分别通过在损失函数后加上参数向量θ的L1范式和L2范式的倍数来实现。这个增加的范式，被称为"正则项"，也被称为"惩罚项"。损失函数改变，基于损失函数的最优化来求解的参数取值必然改变，我们以此来调节模型拟合的程度。其中L1范数表现为参数向量中的每个参数的绝对值之和，L2范数表现为参数向量中的每个参数的平方和的开方值。

$$J(\theta)_{L1} = J(\theta) + \frac{1}{C}\sum_{j=1}^{n}|\theta_j| \ \ (j \geq 1)$$

$$J(\theta)_{L2} = J(\theta) + \frac{1}{C}\sqrt{\sum_{j=1}^{n}(\theta_j)^2} \ \ (j \geq 1)$$

其中$J(\theta)$是我们之前提过的损失函数，C是用来控制正则化程度的超参数，n是方程中特征的总数，也是方程中参数的总数，j代表每个参数。在这里，j要大于等于1，是因为我们的参数向量θ中，第一个参数是θ_0，是我们的截距，它通常是不参与正则化的。

参数	说明
penalty	可以输入"l1"或"l2"来指定使用哪一种正则化方式，不填写默认"l2"。 注意，若选择"l1"正则化，参数solver仅能够使用"liblinear"，若使用"l2"正则化，参数solver中所有的求解方式都可以使用。
C	C正则化强度的倒数，必须是一个大于0的浮点数，不填写默认1.0，即默认一倍正则项。 C越小，对损失函数的惩罚越重，正则化的效力越强，参数θ会逐渐被压缩得越来越小。在很多书籍和博客的原理讲解中，$\frac{1}{C}$被写作λ，为了大家便于理解sklearn中的参数，我将公式改写成$\frac{1}{C}$，更加直观。

L1正则化和L2正则化虽然都可以控制过拟合，但它们的效果并不相同。当正则化强度逐渐增大（即C逐渐变小），参数θ的取值会逐渐变小，但**L1正则化会将参数压缩为0，L2正则化只会让参数尽量小，不会取到0。**

在L1正则化在逐渐加强的过程中，携带信息量小的、对模型贡献不大的特征的参数，会比携带大量信息的、对模型有巨大贡献的特征的参数更快地变成0，所以L1正则化本质是一个特征选择的过程，掌管了参数的"稀疏性"。L1正则化越强，参数向量中就越多的参数为0，参数就越稀疏，选出来的特征就越少，以此来防止过拟合。因此，如果特征量很大，数据维度很高，我们会倾向于使用L1正则化。由于L1正则化的这个性质，逻辑回归的特征选择可以由Embedded嵌入法来完成。

相对的，L2正则化在加强的过程中，会尽量让每个特征对模型都有一些小的贡献，但携带信息少，对模型贡献不大的特征的参数会非常接近于0。通常来说，如果我们的主要目的只是为了防止过拟合，选择L2正则化就足够了。但是如果选择L2正则化后还是过拟合，模型在未知数据集上的效果表现很差，就可以考虑L1正则化。

而两种正则化下C的取值，都可以通过学习曲线来进行调整。

建立两个逻辑回归，L1正则化和L2正则化的差别就一目了然了：

```
from sklearn.linear_model import LogisticRegression as LR
from sklearn.datasets import load_breast_cancer
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

data = load_breast_cancer()
X = data.data
y = data.target

data.data.shape

lrl1 = LR(penalty="l1",solver="liblinear",C=0.5,max_iter=1000)

lrl2 = LR(penalty="l2",solver="liblinear",C=0.5,max_iter=1000)

#逻辑回归的重要属性coef_，查看每个特征所对应的参数
lrl1 = lrl1.fit(X,y)
lrl1.coef_

(lrl1.coef_ != 0).sum(axis=1)

lrl2 = lrl2.fit(X,y)
lrl2.coef_
```

可以看见，当我们选择L1正则化的时候，许多特征的参数都被设置为了0，这些特征在真正建模的时候，就不会出现在我们的模型当中了，而L2正则化则是对所有的特征都给出了参数。

究竟哪个正则化的效果更好呢？还是都差不多？

```
l1 = []
l2 = []
l1test = []
l2test = []
```

```
Xtrain, Xtest, Ytrain, Ytest = train_test_split(X,y,test_size=0.3,random_state=420)

for i in np.linspace(0.05,1,19):
    lrl1 = LR(penalty="l1",solver="liblinear",C=i,max_iter=1000)
    lrl2 = LR(penalty="l2",solver="liblinear",C=i,max_iter=1000)

    lrl1 = lrl1.fit(Xtrain,Ytrain)
    l1.append(accuracy_score(lrl1.predict(Xtrain),Ytrain))
    l1test.append(accuracy_score(lrl1.predict(Xtest),Ytest))

    lrl2 = lrl2.fit(Xtrain,Ytrain)
    l2.append(accuracy_score(lrl2.predict(Xtrain),Ytrain))
    l2test.append(accuracy_score(lrl2.predict(Xtest),Ytest))

graph = [l1,l2,l1test,l2test]
color = ["green","black","lightgreen","gray"]
label = ["L1","L2","L1test","L2test"]

plt.figure(figsize=(6,6))
for i in range(len(graph)):
    plt.plot(np.linspace(0.05,1,19),graph[i],color[i],label=label[i])
plt.legend(loc=4) #图例的位置在哪里?4表示，右下角
plt.show()
```

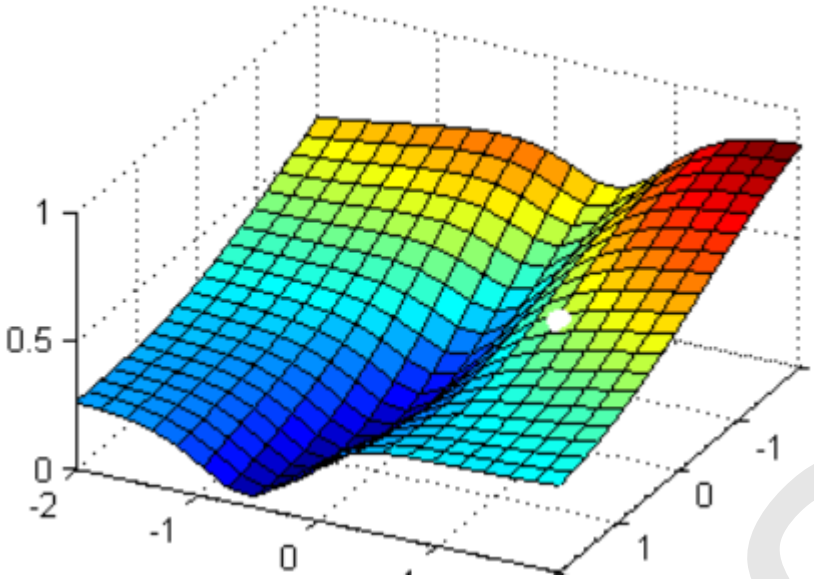
可见，至少在我们的乳腺癌数据集下，两种正则化的结果区别不大。但随着C的逐渐变大，正则化的强度越来越小，模型在训练集和测试集上的表现都呈上升趋势，直到C=0.8左右，训练集上的表现依然在走高，但模型在未知数据集上的表现开始下跌，这时候就是出现了过拟合。我们可以认为，C设定为0.9会比较好。在实际使用时，基本就默认使用l2正则化，如果感觉到模型的效果不好，那就换L1试试看。

【完整版】2.2 梯度下降：重要参数max_iter

之前提到过，逻辑回归的数学目的是求解能够让模型最优化的参数θ的值，即求解能够让损失函数最小化的θ的值，而这个求解过程，对于二元逻辑回归来说，有多种方法可以选择，最常见的有梯度下降法(Gradient Descent)，坐标轴下降法(Coordinate Descent)，牛顿法(Newton-Raphson method)等，每种方法都涉及复杂的数学原理，但这些计算在执行的任务其实是类似的。

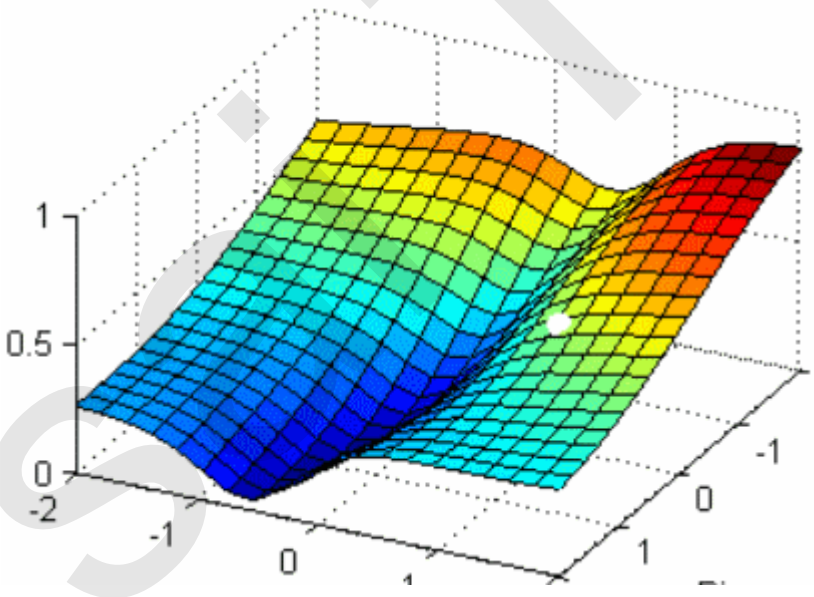
$$J(\theta) = -\sum_{i=1}^{m}(y_i * log(y_\theta(x_i)) + (1-y_i) * log(1-y_\theta(x_i)))$$

以梯度下降法为例，我们来看看求解过程是如何完成的。下面这个华丽的平面就是我们的损失函数在输入了一组特征矩阵和标签之后在三维立体坐标系中的图像。现在，我们寻求的是损失函数的最小值，也就是图像的最低点（看起来像是深蓝色区域的某处），一旦我们获取了图像在最低点的取值$J(\theta)_0$，在我们的损失函数公式中，唯一未知的就是我们的参数向量θ了。



现在，我在这个图像上随机放一个小球，当我松手，这个小球就会顺着这个华丽的平面滚落，直到滚到深蓝色的区域——损失函数的最低点。但是，小球不能够一次性滚动到最低处，它的能量不足，所以每次最多只能走距离G。为了严格监控这个小球的行为，我要求小球每次只能走0.05 * G，并且我要记下它每次走动的方向，直到它滚到图像上的最低点。

现在我松手，来看小球如何运动：



可以看见，小球从高处滑落，最终停在深蓝色的区域中的某个点上，而这个点就是我们在现有状况下可以获得的，图像的最低点。有了这个图像的最低点的数值，我们就可以计算出这个点所对应的参数向量θ了。

【完整版】2.3 二元回归与多元回归：重要参数solver

【完整版】2.4 逻辑回归中的特征选择

【完整版】2.5 样本不平衡与参数class_weight

【完整版】4 案例：用逻辑回归制作评分卡

【完整版】5 附录：

【完整版】5.1 逻辑回归的参数列表

【完整版】5.2 逻辑回归的属性列表

【完整版】5.3 逻辑回归的接口列表