Google机器学习速成课程笔记

1. **机器学习概念**
   1. **机器学习简介**
   2. **框架处理**
2. **机器学习术语**

什么是（监督式）机器学习？简单来说，它的定义如下：机器学习系统通过学习如何组合输入信息来对从未见过的数据做出有用的预测。下面我们来了解一下机器学习的基本术语。

**标签** 是我们要预测的事物，即简单线性回归中的 y 变量。标签可以是小麦未来的价格、图片中显示的动物品种、音频剪辑的含义或任何事物。

**特征** 是输入变量，即简单线性回归中的 x 变量。简单的机器学习项目可能会使用单个特征，而比较复杂的机器学习项目可能会使用数百万个特征，按如下方式指定：

{x\_1,x\_2,...x\_N}

在垃圾邮件检测器示例中，特征可能包括：

电子邮件文本中的字词

发件人的地址

发送电子邮件的时段

电子邮件中包含“一种奇怪的把戏”这样的短语。

**样本** 是指数据的特定实例：**x**。（我们采用粗体 **x** 表示它是一个矢量。）我们将样本分为以下两类：

有标签样本

无标签样本

**有标签样本**同时包含特征和标签。即：

labeled examples: {features, label}: (x, y)

我们使用有标签样本来训练模型。在我们的垃圾邮件检测器示例中，有标签样本是用户明确标记为“垃圾邮件”或“非垃圾邮件”的各个电子邮件。

例如，下表显示了从包含加利福尼亚州房价信息的[数据集](https://developers.google.cn/machine-learning/crash-course/california-housing-data-description)中抽取的 5 个有标签样本：

|  |  |  |  |
| --- | --- | --- | --- |
| housingMedianAge （特征） | totalRooms （特征） | totalBedrooms （特征） | medianHouseValue （标签） |
| 15 | 5612 | 1283 | 66900 |
| 19 | 7650 | 1901 | 80100 |
| 17 | 720 | 174 | 85700 |
| 14 | 1501 | 337 | 73400 |
| 20 | 1454 | 326 | 65500 |

**无标签样本**包含特征，但不包含标签。即：

unlabeled examples: {features, ?}: (x, ?)

在使用有标签样本训练了我们的模型之后，我们会使用该模型来预测无标签样本的标签。在垃圾邮件检测器示例中，无标签样本是用户尚未添加标签的新电子邮件。

**模型** 定义了特征与标签之间的关系。例如，垃圾邮件检测模型可能会将某些特征与“垃圾邮件”紧密联系起来。我们来重点介绍一下模型生命周期的两个阶段：

**训练** 表示创建或**学习**模型。也就是说，您向模型展示有标签样本，让模型逐渐学习特征与标签之间的关系。

**推断** 表示将训练后的模型应用于无标签样本。也就是说，您使用训练后的模型来做出有用的预测 (y')。例如，在推断期间，您可以针对新的无标签样本预测 medianHouseValue。

回归与分类

**回归** 模型可预测连续值。例如，回归模型做出的预测可回答如下问题：

加利福尼亚州一栋房产的价值是多少？

用户点击此广告的概率是多少？

**分类** 模型可预测离散值。例如，分类模型做出的预测可回答如下问题：

某个指定电子邮件是垃圾邮件还是非垃圾邮件？

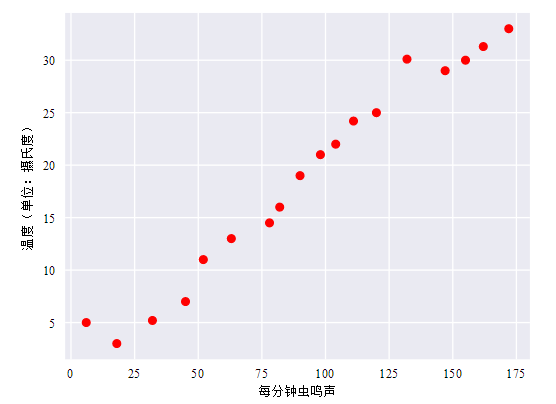
这是一张狗、猫还是仓鼠图片？

* 1. **深入了解机器学习**

1. **线性回归**

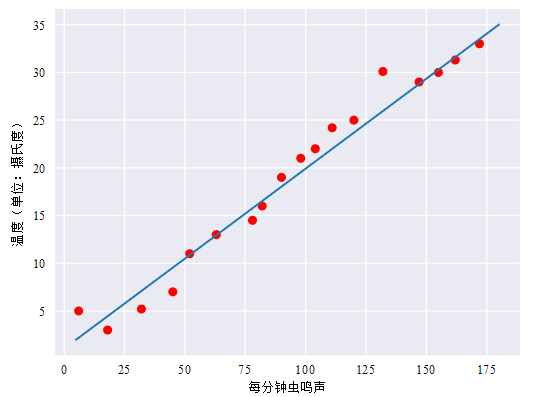
人们[早就知晓](https://wikipedia.org/wiki/Dolbear's_law)，相比凉爽的天气，蟋蟀在较为炎热的天气里鸣叫更为频繁。数十年来，专业和业余昆虫学者已将每分钟的鸣叫声和温度方面的数据编入目录。Ruth 阿姨将她喜爱的蟋蟀数据库作为生日礼物送给您，并邀请您自己利用该数据库训练一个模型，从而预测鸣叫声与温度的关系。

首先建议您将数据绘制成图表，了解下数据的分布情况：



**图 1. 每分钟的鸣叫声与温度（摄氏度）的关系**

毫无疑问，此曲线图表明温度随着鸣叫声次数的增加而上升。鸣叫声与温度之间的关系是线性关系吗？是的，您可以绘制一条直线来近似地表示这种关系，如下所示：



**图 2. 线性关系**

事实上，虽然该直线并未精确无误地经过每个点，但针对我们拥有的数据，清楚地显示了鸣叫声与温度之间的关系。只需运用一点代数知识，您就可以将这种关系写下来，如下所示：

y=mx+b

其中：

y 指的是温度（以摄氏度表示），即我们试图预测的值。

m 指的是直线的斜率。

x 指的是每分钟的鸣叫声次数，即输入特征的值。

b 指的是 y 轴截距。

按照机器学习的惯例，您需要写一个存在细微差别的模型方程式：

y′=b+w1x1

其中：

y′ 指的是预测[标签](https://developers.google.cn/machine-learning/crash-course/framing/ml-terminology#labels)（理想输出值）。

b 指的是偏差（y 轴截距）。而在一些机器学习文档中，它称为 w0。

w1 指的是特征 1 的权重。权重与上文中用 m 表示的“斜率”的概念相同。

x1 指的是[特征](https://developers.google.cn/machine-learning/crash-course/framing/ml-terminology#features)（已知输入项）。

要根据新的每分钟的鸣叫声值 x1 **推断**（预测）温度 y′，只需将 x1 值代入此模型即可。

下标（例如 w1 和 x1）预示着可以用多个特征来表示更复杂的模型。例如，具有三个特征的模型可以采用以下方程式：

y′=b+w1x1+w2x2+w3x3

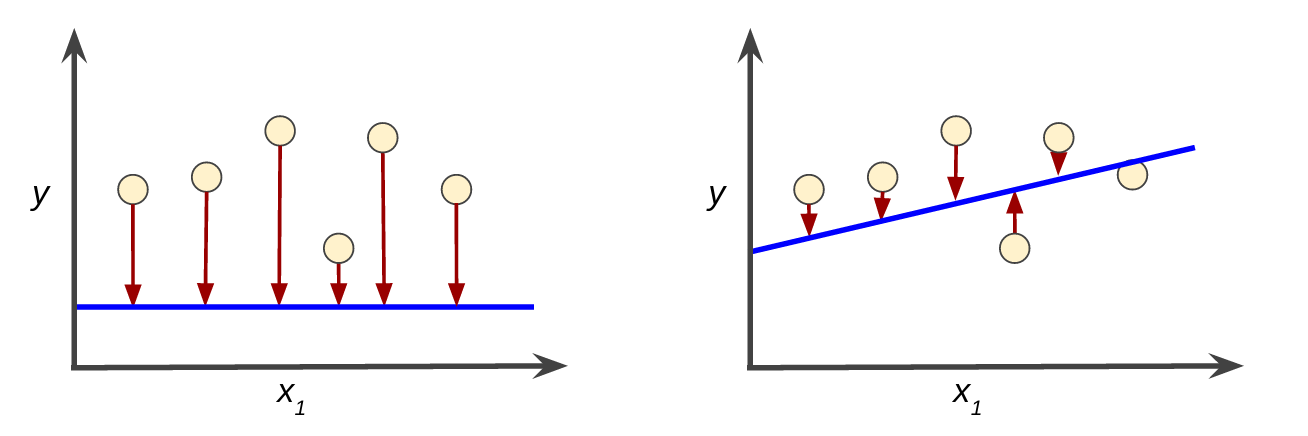
1. **训练与损失**

简单来说，**训练**模型表示通过有标签样本来学习（确定）所有权重和偏差的理想值。在监督式学习中，机器学习算法通过以下方式构建模型：检查多个样本并尝试找出可最大限度地减少损失的模型；这一过程称为**经验风险最小化**。

损失是对糟糕预测的惩罚。也就是说，**损失**是一个数值，表示对于单个样本而言模型预测的准确程度。如果模型的预测完全准确，则损失为零，否则损失会较大。训练模型的目标是从所有样本中找到一组平均损失“较小”的权重和偏差。例如，图 3 左侧显示的是损失较大的模型，右侧显示的是损失较小的模型。关于此图，请注意以下几点：

*红色箭头表示损失*

*蓝线表示预测*



**图 3. 左侧模型的损失较大；右侧模型的损失较小**

请注意，左侧曲线图中的红色箭头比右侧曲线图中的对应红色箭头长得多。显然，相较于左侧曲线图中的蓝线，右侧曲线图中的蓝线代表的是预测效果更好的模型。

您可能想知道自己能否创建一个数学函数（损失函数），以有意义的方式汇总各个损失。

平方损失：一种常见的损失函数

接下来我们要看的线性回归模型使用的是一种称为**平方损失**（又称为 **L2 损失**）的损失函数。单个样本的平方损失如下：

= the square of the difference between the label and the prediction

= (observation - prediction(x))2

= (y - y')2

**均方误差** (**MSE**) 指的是每个样本的平均平方损失。要计算 MSE，请求出各个样本的所有平方损失之和，然后除以样本数量：

C:\Users\Administrator\AppData\Roaming\Tencent\Users\188512936\TIM\WinTemp\RichOle\PR1P]%E~4V}0`[3~5I]4OOU.png

其中：

(x,y) 指的是样本，其中

x 指的是模型进行预测时使用的特征集（例如，温度、年龄和交配成功率）。

y 指的是样本的标签（例如，每分钟的鸣叫次数）。

prediction(x) 指的是权重和偏差与特征集 x 结合的函数。

D 指的是包含多个有标签样本（即 (x,y)）的数据集。

N 指的是 D 中的样本数量。

虽然 MSE 常用于机器学习，但它既不是唯一实用的损失函数，也不是适用于所有情形的最佳损失函数。

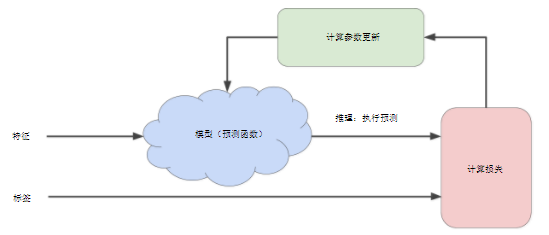
* 1. **降低损失**

1. **迭代方法**

[上一单元](https://developers.google.cn/machine-learning/crash-course/descending-into-ml)介绍了损失的概念。在本单元中，您将了解机器学习模型如何以迭代方式降低损失。

迭代学习可能会让您想到“[Hot and Cold](http://www.howcast.com/videos/258352-how-to-play-hot-and-cold/)”这种寻找隐藏物品（如顶针）的儿童游戏。在我们的游戏中，“隐藏的物品”就是最佳模型。刚开始，您会胡乱猜测（“w1 的值为 0。”），等待系统告诉您损失是多少。然后，您再尝试另一种猜测（“w1 的值为 0.5。”），看看损失是多少。哎呀，这次更接近目标了。实际上，如果您以正确方式玩这个游戏，通常会越来越接近目标。这个游戏真正棘手的地方在于尽可能高效地找到最佳模型。

下图显示了机器学习算法用于训练模型的迭代试错过程：



**图 1. 用于训练模型的迭代方法**

我们将在整个机器学习速成课程中使用相同的迭代方法详细说明各种复杂情况，尤其是处于暴风雨中的蓝云区域。迭代策略在机器学习中的应用非常普遍，这主要是因为它们可以很好地扩展到大型数据集。

“模型”部分将一个或多个特征作为输入，然后返回一个预测 (y') 作为输出。为了进行简化，不妨考虑一种采用一个特征并返回一个预测的模型：

y′=b+w1x1

我们应该为 b 和 w1 设置哪些初始值？对于线性回归问题，事实证明初始值并不重要。我们可以随机选择值，不过我们还是选择采用以下这些无关紧要的值：

b = 0

w1 = 0

假设第一个特征值是 10。将该特征值代入预测函数会得到以下结果：

 y' = 0 + 0(10)  
 y' = 0

图中的“计算损失”部分是模型将要使用的[损失函数](https://developers.google.cn/machine-learning/crash-course/descending-into-ml/training-and-loss)。假设我们使用平方损失函数。损失函数将采用两个输入值：

y'：模型对特征 x 的预测

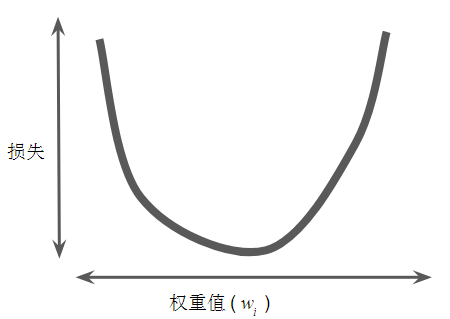
y：特征 x 对应的正确标签。

最后，我们来看图的“计算参数更新”部分。机器学习系统就是在此部分检查损失函数的值，并为 b 和 w1 生成新值。现在，假设这个神秘的绿色框会产生新值，然后机器学习系统将根据所有标签重新评估所有特征，为损失函数生成一个新值，而该值又产生新的参数值。这种学习过程会持续迭代，直到该算法发现损失可能最低的模型参数。通常，您可以不断迭代，直到总体损失不再变化或至少变化极其缓慢为止。这时候，我们可以说该模型已**收敛**。

1. **梯度下降**

迭代方法图（[图 1](https://developers.google.cn/machine-learning/crash-course/reducing-loss/an-iterative-approach#ml-block-diagram)）包含一个标题为“计算参数更新”的华而不实的绿框。现在，我们将用更实质的方法代替这种华而不实的算法。

假设我们有时间和计算资源来计算 w1 的所有可能值的损失。对于我们一直在研究的回归问题，所产生的损失与 w1 的图形始终是凸形。换言之，图形始终是碗状图，如下所示：

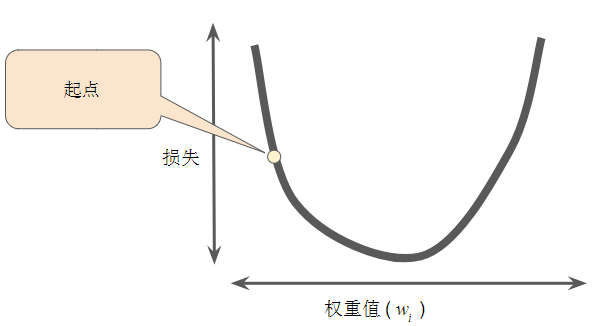


**图 2.回归问题产生的损失与权重图为凸形**

 凸形问题只有一个最低点；即只存在一个斜率正好为 0 的位置。这个最小值就是损失函数收敛之处。

通过计算整个数据集中 w1 每个可能值的损失函数来找到收敛点这种方法效率太低。我们来研究一种更好的机制，这种机制在机器学习领域非常热门，称为**梯度下降法**。

梯度下降法的第一个阶段是为 w1 选择一个起始值（起点）。起点并不重要；因此很多算法就直接将 w1 设为 0 或随机选择一个值。下图显示的是我们选择了一个稍大于 0 的起点：



**图 3. 梯度下降法的起点**

然后，梯度下降法算法会计算损失曲线在起点处的梯度。简而言之，**梯度**是偏导数的矢量；它可以让您了解哪个方向距离目标“更近”或“更远”。请注意，损失相对于单个权重的梯度（如图 3 所示）就等于导数。

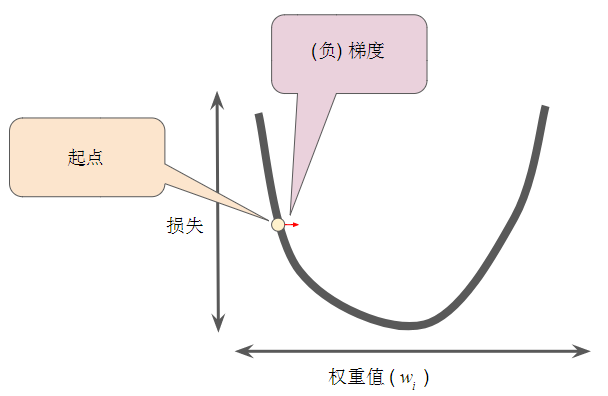
详细了解偏导数和梯度。

请注意，梯度是一个矢量，因此具有以下两个特征：

方向

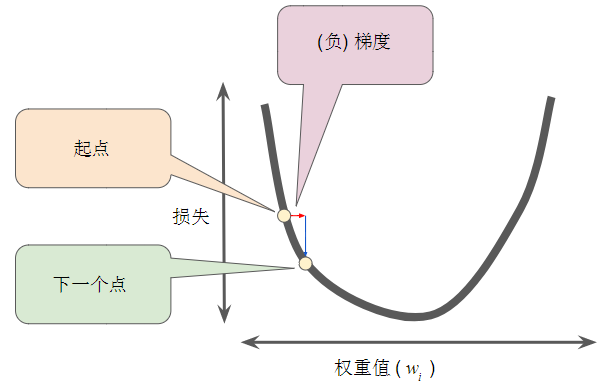
大小

梯度始终指向损失函数中增长最为迅猛的方向。梯度下降法算法会沿着负梯度的方向走一步，以便尽快降低损失。



**图 4. 梯度下降法依赖于负梯度**

为了确定损失函数曲线上的下一个点，梯度下降法算法会将梯度大小的一部分与起点相加，如下图所示：



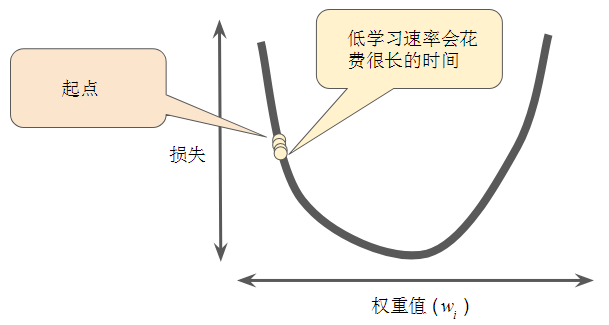
**图 5. 一个梯度步长将我们移动到损失曲线上的下一个点**

然后，梯度下降法会重复此过程，逐渐接近最低点。

1. **学习速率**

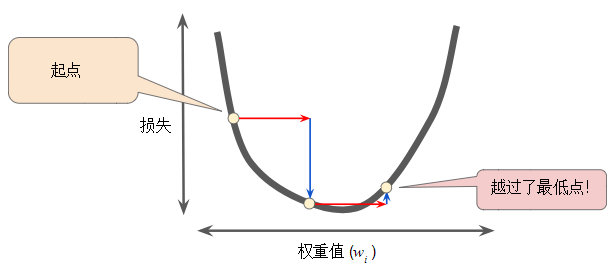
正如之前所述，梯度矢量具有方向和大小。梯度下降法算法用梯度乘以一个称为**学习速率**（有时也称为**步长**）的标量，以确定下一个点的位置。例如，如果梯度大小为 2.5，学习速率为 0.01，则梯度下降法算法会选择距离前一个点 0.025 的位置作为下一个点。

**超参数**是编程人员在机器学习算法中用于调整的旋钮。大多数机器学习编程人员会花费相当多的时间来调整学习速率。如果您选择的学习速率过小，就会花费太长的学习时间：



**图 6. 学习速率过小**

相反，如果您指定的学习速率过大，下一个点将永远在 U 形曲线的底部随意弹跳，就好像量子力学实验出现了严重错误一样：



**图 7. 学习速率过大**

每个回归问题都存在一个[金发姑娘](https://wikipedia.org/wiki/Goldilocks_principle)学习速率。“金发姑娘”值与损失函数的平坦程度相关。如果您知道损失函数的梯度较小，则可以放心地试着采用更大的学习速率，以补偿较小的梯度并获得更大的步长。

![C:\Users\Administrator\AppData\Roaming\Tencent\Users\188512936\TIM\WinTemp\RichOle\SZUCU8OBLQ((DCTBDH1C](Q.png](data:image/png;base64,)

**图 8. 学习速率恰恰好**

1. **随机梯度下降法**

在梯度下降法中，批量指的是用于在单次迭代中计算梯度的样本总数。到目前为止，我们一直假定批量是指整个数据集。就 Google 的规模而言，数据集通常包含数十亿甚至数千亿个样本。此外，Google 数据集通常包含海量特征。因此，一个批量可能相当巨大。如果是超大批量，则单次迭代就可能要花费很长时间进行计算。

包含随机抽样样本的大型数据集可能包含冗余数据。实际上，批量大小越大，出现冗余的可能性就越高。一些冗余可能有助于消除杂乱的梯度，但超大批量所具备的预测价值往往并不比大型批量高。

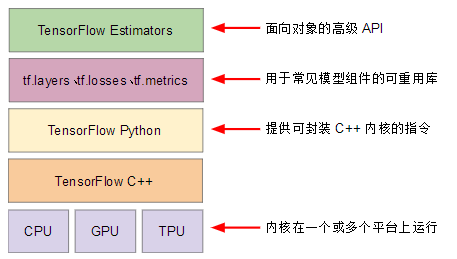
如果我们可以通过更少的计算量得出正确的平均梯度，会怎么样？通过从我们的数据集中随机选择样本，我们可以通过小得多的数据集估算（尽管过程非常杂乱）出较大的平均值。 随机梯度下降法 (SGD) 将这种想法运用到极致，它每次迭代只使用一个样本（批量大小为 1）。如果进行足够的迭代，SGD 也可以发挥作用，但过程会非常杂乱。“随机”这一术语表示构成各个批量的一个样本都是随机选择的。

小批量随机梯度下降法（小批量 SGD）是介于全批量迭代与 SGD 之间的折衷方案。小批量通常包含 10-1000 个随机选择的样本。小批量 SGD 可以减少 SGD 中的杂乱样本数量，但仍然比全批量更高效。

为了简化说明，我们只针对单个特征重点介绍了梯度下降法。请放心，梯度下降法也适用于包含多个特征的特征集。

* 1. **使用TF的基本步骤**

下图显示了 TensorFlow 工具包的当前层次结构：



**图 1. TensorFlow 工具包层次结构**

下表总结了不同层的用途：

| 工具包 | 说明 |
| --- | --- |
| Estimator (tf.estimator) | 高级 OOP API。 |
| tf.layers/tf.losses/tf.metrics | 用于常见模型组件的库。 |
| TensorFlow | 低级 API |

TensorFlow 由以下两个组件组成：

[图协议缓冲区](https://tensorflow.google.cn/extend/tool_developers/#protocol_buffers)

执行（分布式）图的运行时

这两个组件类似于 Java 编译器和 JVM。正如 JVM 会实施在多个硬件平台（CPU 和 GPU）上一样，TensorFlow 也是如此。

您应该使用哪个 API？您应该使用能够解决问题的最高级抽象层。较高级别的抽象层更易于使用，但（设计方面）不够灵活。我们建议您先从最高级 API 入手，让所有组件正常运作起来。如果您希望在某些特殊建模方面能够更加灵活一些，则可以降低一个级别。请注意，每个级别都是使用低级 API 构建的，因此降低层次结构级别应该比较直观。

**tf.estimator API**

我们将使用 tf.estimator 来完成机器学习速成课程中的大部分练习。您在练习中所做的一切都可以在较低级别（原始）的 TensorFlow 中完成，但使用 tf.estimator 会大大减少代码行数。

tf.estimator 与 scikit-learn API 兼容。 [scikit-learn](http://scikit-learn.org/) 是极其热门的 Python 开放源代码机器学习库，拥有超过 10 万名用户，其中包括许多 Google 员工。

概括而言，以下是在 tf.estimator 中实现的线性回归程序的格式：

import tensorflow as tf  
  
# Set up a linear classifier.  
classifier = tf.estimator.LinearClassifier()  
  
# Train the model on some example data.  
classifier.train(input\_fn=train\_input\_fn, steps=2000)  
  
# Use it to predict.  
predictions = classifier.predict(input\_fn=predict\_input\_fn)

* 1. **泛华**

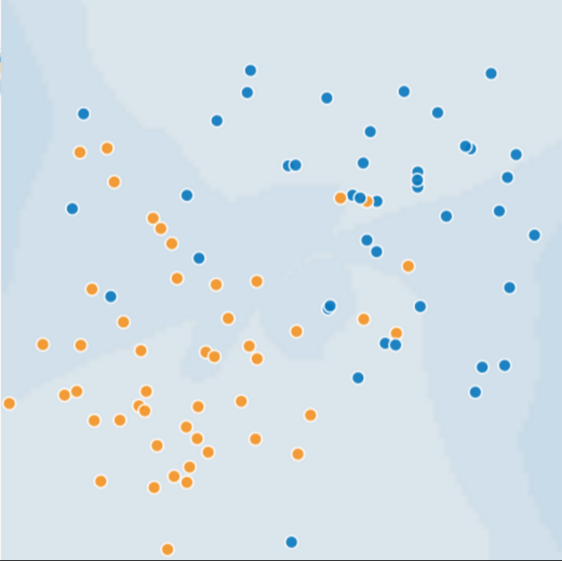
**过拟合风险**

本单元将重点介绍泛化。为了让您直观地理解这一概念，我们将展示 3 张图。假设这些图中的每个点代表一棵树在森林中的位置。图中的两种颜色分别代表以下含义：

*蓝点代表生病的树*

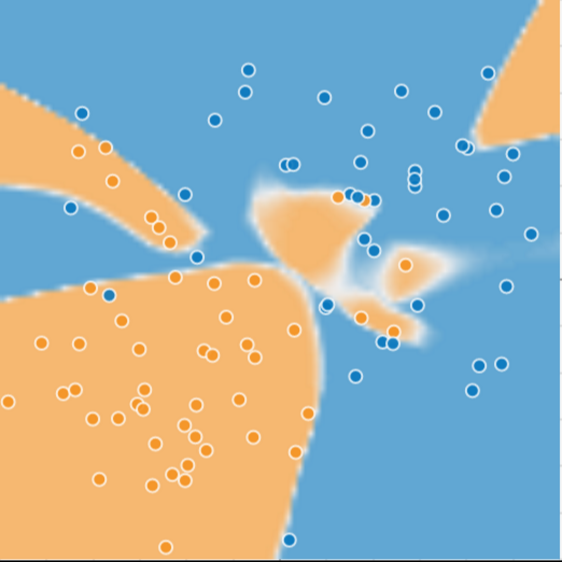
*橙点代表健康的树*

接下来，我们来看看图 1。



**图 1. 生病（蓝色）和健康（橙色）的树**

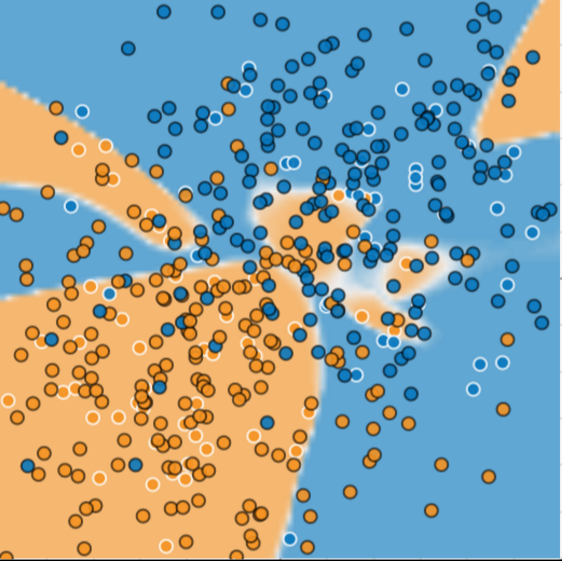
您能设想出一个有效的模型来预测以后的生病或健康的树吗？花点时间在脑海里绘制一条弧线将蓝点与橙点分开，或者在脑海中圈住一些橙点或蓝点。然后再看看图 2，它显示某种机器学习模型如何将生病的树与健康的树区分开。请注意，该模型产生的损失非常低。



**图 2. 用于区分生病的树与健康的树的复杂模型**

损失很低，但仍然是糟糕的模型？

图 3 显示我们向该模型中添加了新数据后所发生的情况。结果表明，该模型在处理新数据方面表现非常糟糕。请注意，该模型对大部分新数据的分类都不正确。



**图 3. 该模型在预测新数据方面表现非常糟糕**

图 2 和图 3 所示的模型**过拟合**了训练数据的特性。过拟合模型在训练过程中产生的损失很低，但在预测新数据方面的表现却非常糟糕。如果某个模型在拟合当前样本方面表现良好，那么我们如何相信该模型会对新数据做出良好的预测呢？正如您[稍后](https://developers.google.cn/machine-learning/crash-course/regularization-for-simplicity/l2-regularization)将看到的，过拟合是由于模型的复杂程度超出所需程度而造成的。机器学习的基本冲突是适当拟合我们的数据，但也要尽可能简单地拟合数据。

机器学习的目标是对从真实概率分布（已隐藏）中抽取的新数据做出良好预测。遗憾的是，模型无法查看整体情况；模型只能从训练数据集中取样。如果某个模型在拟合当前样本方面表现良好，那么您如何相信该模型也会对从未见过的样本做出良好预测呢？

奥卡姆的威廉是 14 世纪一位崇尚简单的修士和哲学家。他认为科学家应该优先采用更简单（而非更复杂）的公式或理论。奥卡姆剃刀定律在机器学习方面的运用如下：

机器学习模型越简单，良好的实证结果就越有可能不仅仅基于样本的特性。

现今，我们已将奥卡姆剃刀定律正式应用于**统计学习理论**和**计算学习理论**领域。这些领域已经形成了**泛化边界**，即统计化描述模型根据以下因素泛化到新数据的能力：

模型的复杂程度

模型在处理训练数据方面的表现

虽然理论分析在理想化假设下可提供正式保证，但在实践中却很难应用。机器学习速成课程则侧重于实证评估，以评判模型泛化到新数据的能力。

机器学习模型旨在根据以前未见过的新数据做出良好预测。但是，如果您要根据数据集构建模型，如何获得以前未见过的数据呢？一种方法是将您的数据集分成两个子集：

**训练集** - 用于训练模型的子集

**测试集** - 用于测试模型的子集

一般来说，在测试集上表现是否良好是衡量能否在新数据上表现良好的有用指标，前提是：

测试集足够大。

您不会反复使用相同的测试集来作假。

机器学习细则

以下三项基本假设阐明了泛化：

我们从分布中随机抽取**独立同分布** (**i.i.d**) 的样本。换言之，样本之间不会互相影响。（另一种解释：i.i.d. 是表示变量随机性的一种方式）。

分布是**平稳的**；即分布在数据集内不会发生变化。

我们从同一分布的数据划分中抽取样本。

在实践中，我们有时会违背这些假设。例如：

想象有一个选择要展示的广告的模型。如果该模型在某种程度上根据用户以前看过的广告选择广告，则会违背 i.i.d. 假设。

想象有一个包含一年零售信息的数据集。用户的购买行为会出现季节性变化，这会违反平稳性。

总结：

如果违背了上述三项基本假设中的任何一项，那么我们就必须密切注意指标。

如果某个模型尝试紧密拟合训练数据，但却不能很好地泛化到新数据，就会发生过拟合。

如果不符合监督式机器学习的关键假设，那么我们将失去对新数据进行预测这项能力的重要理论保证。

* 1. **训练集和测试集**

**拆分数据**

上一单元介绍了将数据集分为两个子集的概念：

训练集 - 用于训练模型的子集。

测试集 - 用于测试训练后模型的子集。

您可以想象按如下方式拆分单个数据集：



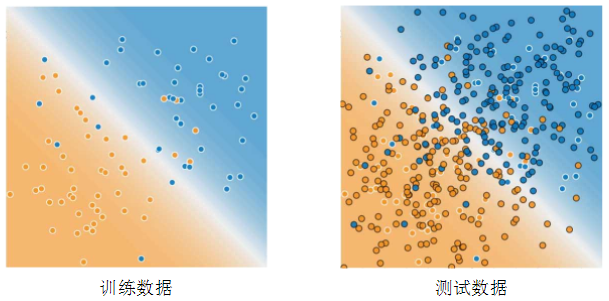
**图 1. 将单个数据集拆分为一个训练集和一个测试集**

确保您的测试集满足以下两个条件：

规模足够大，可产生具有统计意义的结果。

能代表整个数据集。换言之，挑选的测试集的特征应该与训练集的特征相同。

假设您的测试集满足上述两个条件，您的目标是创建一个能够很好地泛化到新数据的模型。我们的测试集充当新数据的代理。以下图为例。请注意，从训练数据中学习的模型非常简单。该模型的表现并不完美，出现了一些错误的预测。不过，该模型在测试数据上的表现与在训练数据上的表现一致。也就是说，这个简单的模型没有过拟合训练数据。



**图 2. 对照测试数据验证训练后的模型**

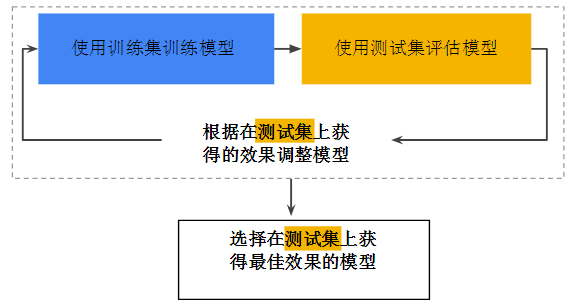
**请勿对测试数据进行训练。** 如果您的评估指标取得了意外的好结果，则可能表明您不小心对测试集进行了训练。例如，高准确率可能表明测试数据泄露到了训练集。

例如，假设一个模型要预测某封电子邮件是否是垃圾邮件，它使用主题行、邮件正文和发件人的电子邮件地址作为特征。我们按照 80-20 的拆分比例将数据拆分为训练集和测试集。在训练之后，该模型在训练集和测试集上均达到了 99% 的精确率。我们原本预计测试集上的精确率会低于此结果，因此再次查看数据后发现，测试集中的很多样本与训练集中的样本是重复的（由于疏忽，我们在拆分数据之前，没有将输入数据库中的相同垃圾邮件重复条目清理掉）。我们无意中对一些测试数据进行了训练，因此无法再准确衡量该模型泛化到新数据的效果。

* 1. **验证**

**另一个划分**

[上一单元](https://developers.google.cn/machine-learning/crash-course/training-and-test-sets/video-lecture)介绍了如何将数据集划分为训练集和测试集。借助这种划分，您可以对一个样本集进行训练，然后使用不同的样本集测试模型。采用两种分类之后，工作流程可能如下所示：



**图 1. 可能的工作流程？**

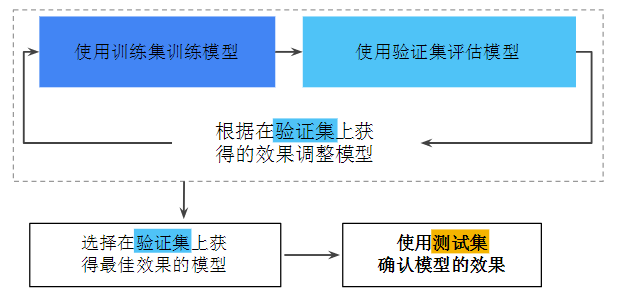
在图中，“调整模型”指的是调整您可以想到的关于模型的任何方面，从更改学习速率、添加或移除特征，到从头开始设计全新模型。该工作流程结束时，您可以选择在测试集上获得最佳效果的模型。

将数据集划分为两个子集是个不错的想法，但不是万能良方。通过将数据集划分为三个子集（如下图所示），您可以大幅降低过拟合的发生几率：



**图 2. 将单个数据集划分为三个子集**

使用**验证集**评估训练集的效果。然后，在模型“通过”验证集之后，使用测试集再次检查评估结果。下图展示了这一新工作流程：



**图 3. 更好的工作流程**

在这一经过改进的工作流程中：

1.选择在验证集上获得最佳效果的模型。

2.使用测试集再次检查该模型。

该工作流程之所以更好，原因在于它暴露给测试集的信息更少。

*提示*

*不断使用测试集和验证集会使其逐渐失去效果。也就是说，您使用相同数据来决定超参数设置或其他模型改进的次数越多，您对于这些结果能够真正泛化到未见过的新数据的信心就越低。请注意，验证集的失效速度通常比测试集缓慢。*

*如果可能的话，建议您收集更多数据来“刷新”测试集和验证集。重新开始是一种很好的重置方式。*

* 1. **表示法**

1. **特征工程**

传统编程的关注点是代码。在机器学习项目中，关注点变成了表示。也就是说，开发者通过添加和改善特征来调整模型。

将原始数据映射到特征

图 1 左侧表示来自输入数据源的原始数据，右侧表示**特征矢量**，也就是组成数据集中样本的浮点值集。 **特征工程**指的是将原始数据转换为特征矢量。进行特征工程预计需要大量时间。

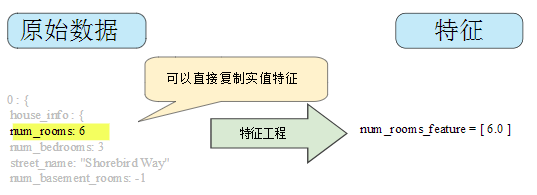
机器学习模型通常期望样本表示为实数矢量。这种矢量的构建方法如下：为每个字段衍生特征，然后将它们全部连接到一起。

![C:\Users\Administrator\AppData\Roaming\Tencent\Users\188512936\TIM\WinTemp\RichOle\G)O0S2YDI6](5P6ST3A41KV.png](data:image/png;base64,)

**图 1. 程序工程将原始数据映射到机器学习特征**

映射数值

机器学习模型根据浮点值进行训练，因此整数和浮点原始数据不需要特殊编码。正如图 2 所示，将原始整数值 6 转换为特征值 6.0 是没有意义的：



**图 2. 将整数值映射到浮点值**

映射字符串值

模型无法通过字符串值学习规律，因此您需要进行一些特征工程来将这些值转换为数字形式：

1）首先，为您要表示的所有特征的字符串值定义一个**词汇表**。对于 street\_name特征，该词汇表中将包含您知道的所有街道。

***注意****：所有其他街道都可以归入一个笼统的“其他”类别，该类别称为****OOV****（未收录到词汇表中）桶。*

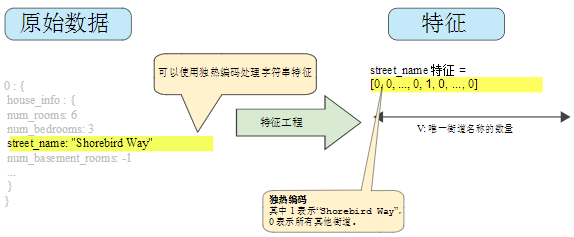
2）然后，使用该词汇表创建一个**独热编码**，用于将指定字符串值表示为二元矢量。在该矢量（与指定的字符串值对应）中：

只有一个元素设为 1。

其他所有元素均设为 0。

该矢量的长度等于词汇表中的元素数。

图 3 显示了某条特定街道 (Shorebird Way) 的独热编码。在此二元矢量中，代表 Shorebird Way 的元素的值为 1，而代表所有其他街道的元素的值为 0。



**图 3. 通过独热编码映射字符串值**

映射分类（枚举）值

分类特征具有一组离散的可能值。例如，名为 Lowland Countries 的特征只包含 3 个可能值：

{'Netherlands', 'Belgium', 'Luxembourg'}

您可能会将分类特征（如 Lowland Countries）编码为枚举类型或表示不同值的整数离散集。例如：

将荷兰表示为 0

将比利时表示为 1

将卢森堡表示为 2

不过，机器学习模型通常将每个分类特征表示为单独的布尔值。例如，Lowland Countries 在模型中可以表示为 3 个单独的布尔值特征：

x1：是荷兰吗？

x2：是比利时吗？

x3：是卢森堡吗？

采用这种方法编码还可以简化某个值可能属于多个分类这种情况（例如，“与法国接壤”对于比利时和卢森堡来说都是 True）。

1. **良好特征的特点**

我们探索了将原始数据映射到合适特征矢量的方法，但这只是工作的一部分。现在，我们必须探索什么样的值才算这些特征矢量中良好的特征。

避免很少使用的离散特征值

良好的特征值应该在数据集中出现大约 5 次以上。这样一来，模型就可以学习该特征值与标签是如何关联的。也就是说，大量离散值相同的样本可让模型有机会了解不同设置中的特征，从而判断何时可以对标签很好地做出预测。例如，house\_type 特征可能包含大量样本，其中它的值为 victorian：

house\_type: victorian

相反，如果某个特征的值仅出现一次或者很少出现，则模型就无法根据该特征进行预测。例如，unique\_house\_id 就不适合作为特征，因为每个值只使用一次，模型无法从中学习任何规律：

unique\_house\_id: 8SK982ZZ1242Z

最好具有清晰明确的含义

每个特征对于项目中的任何人来说都应该具有清晰明确的含义。例如，下面的房龄适合作为特征，可立即识别为年龄：

value.house\_age: 27

相反，对于下方特征值的含义，除了创建它的工程师，其他人恐怕辨识不出：

house\_age: 851472000

在某些情况下，混乱的数据（而不是糟糕的工程选择）会导致含义不清晰的值。例如，以下 user\_age 的来源没有检查值恰当与否：

user\_age: 277

不要将“神奇”的值与实际数据混为一谈

良好的浮点特征不包含超出范围的异常断点或“神奇”的值。例如，假设一个特征具有 0 到 1 之间的浮点值。那么，如下值是可以接受的：

quality\_rating: 0.82

quality\_rating: 0.37

不过，如果用户没有输入 quality\_rating，则数据集可能使用如下神奇值来表示不存在该值：

quality\_rating: -1

为解决神奇值的问题，需将该特征转换为两个特征：

一个特征只存储质量评分，不含神奇值。

一个特征存储布尔值，表示是否提供了 quality\_rating。为该布尔值特征指定一个名称，例如 is\_quality\_rating\_defined。

考虑上游不稳定性

特征的定义不应随时间发生变化。例如，下列值是有用的，因为城市名称一般不会改变。（注意，我们仍然需要将“br/sao\_paulo”这样的字符串转换为独热矢量。）

city\_id: "br/sao\_paulo"

但收集由其他模型推理的值会产生额外成本。可能值“219”目前代表圣保罗，但这种表示在未来运行其他模型时可能轻易发生变化：

inferred\_city\_cluster: "219"

1. **数据清理**

苹果树结出的果子有品相上乘的，也有虫蛀坏果。而高端便利店出售的苹果是 100% 完美的水果。从果园到水果店之间，专门有人花费大量时间将坏苹果剔除或给可以挽救的苹果涂上一层薄薄的蜡。作为一名机器学习工程师，您将花费大量的时间挑出坏样本并加工可以挽救的样本。即使是非常少量的“坏苹果”也会破坏掉一个大规模数据集。

缩放特征值

**缩放**是指将浮点特征值从自然范围（例如 100 到 900）转换为标准范围（例如 0 到 1 或 -1 到 +1）。如果某个特征集只包含一个特征，则缩放可以提供的实际好处微乎其微或根本没有。不过，如果特征集包含多个特征，则缩放特征可以带来以下优势：

帮助梯度下降法更快速地收敛。

帮助避免“NaN 陷阱”。在这种陷阱中，模型中的一个数值变成 [NaN](https://wikipedia.org/wiki/NaN)（例如，当某个值在训练期间超出浮点精确率限制时），并且模型中的所有其他数值最终也会因数学运算而变成 NaN。

帮助模型为每个特征确定合适的权重。如果没有进行特征缩放，则模型会对范围较大的特征投入过多精力。

您不需要对每个浮点特征进行完全相同的缩放。即使特征 A 的范围是 -1 到 +1，同时特征 B 的范围是 -3 到 +3，也不会产生什么恶劣的影响。不过，如果特征 B 的范围是 5000 到 100000，您的模型会出现糟糕的响应。

*要缩放数字数据，一种显而易见的方法是将 [最小值，最大值] 以线性方式映射到较小的范围，例如 [-1，+1]。*

*另一种热门的缩放策略是计算每个值的 Z 得分。Z 得分与距离均值的标准偏差数相关。换而言之：*

*scaledvalue=(value−mean)/stddev.*

*例如，给定以下条件：*

*均值 = 100*

*标准偏差 = 20*

*原始值 = 130*

*则：*

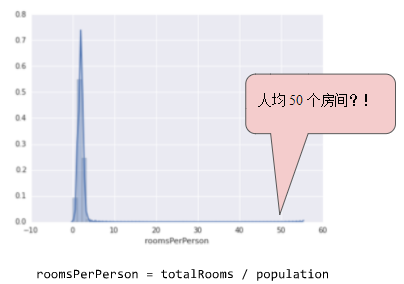
*scaled\_value = (130 - 100) / 20*

*scaled\_value = 1.5*

*使用 Z 得分进行缩放意味着，大多数缩放后的值将介于 -3 和 +3 之间，而少量值将略高于或低于该范围。*

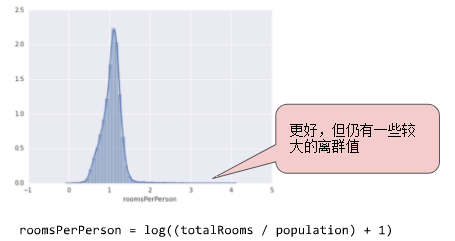
处理极端离群值

下面的曲线图表示的是[加利福尼亚州住房数据集](https://developers.google.cn/machine-learning/crash-course/california-housing-data-description)中称为 roomsPerPerson 的特征。roomsPerPerson 值的计算方法是相应地区的房间总数除以相应地区的人口总数。该曲线图显示，在加利福尼亚州的绝大部分地区，人均房间数为 1 到 2 间。不过，请看一下 x 轴。



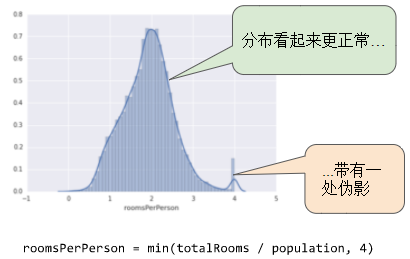
**图 4. 一个非常非常长的尾巴**

如何最大限度降低这些极端离群值的影响？一种方法是对每个值取对数：



**图 5. 对数缩放仍然留有尾巴**

对数缩放可稍稍缓解这种影响，但仍然存在离群值这个大尾巴。我们来采用另一种方法。如果我们只是简单地将 roomsPerPerson 的最大值“限制”为某个任意值（比如 4.0），会发生什么情况呢？

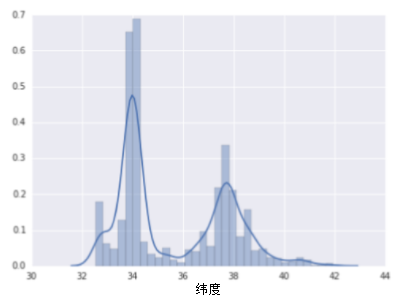


**图 6. 将特征值限制到 4.0**

将特征值限制到 4.0 并不意味着我们会忽略所有大于 4.0 的值。而是说，所有大于 4.0 的值都将变成 4.0。这就解释了 4.0 处的那个有趣的小峰值。尽管存在这个小峰值，但是缩放后的特征集现在依然比原始数据有用。

分箱

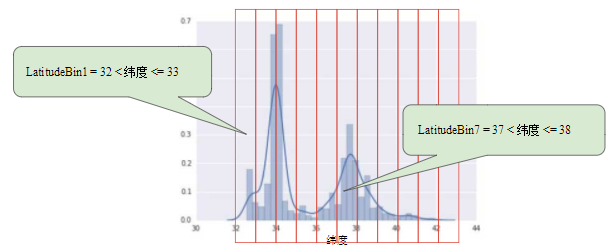
下面的曲线图显示了加利福尼亚州不同纬度的房屋相对普及率。注意集群 - 洛杉矶大致在纬度 34 处，旧金山大致在纬度 38 处。



**图 7. 每个纬度的房屋数**

在数据集中，latitude 是一个浮点值。不过，在我们的模型中将 latitude 表示为浮点特征没有意义。这是因为纬度和房屋价值之间不存在线性关系。例如，纬度 35 处的房屋并不比纬度 34 处的房屋贵 35/34（或更便宜）。但是，纬度或许能很好地预测房屋价值。

为了将纬度变为一项实用的预测指标，我们对纬度“分箱”，如下图所示：



**图 8. 分箱值**

我们现在拥有 11 个不同的布尔值特征（LatitudeBin1、LatitudeBin2、…、LatitudeBin11），而不是一个浮点特征。拥有 11 个不同的特征有点不方便，因此我们将它们统一成一个 11 元素矢量。这样做之后，我们可以将纬度 37.4 表示为：

[0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0]

分箱之后，我们的模型现在可以为每个纬度学习完全不同的权重。

清查

截至目前，我们假定用于训练和测试的所有数据都是值得信赖的。在现实生活中，数据集中的很多样本是不可靠的，原因有以下一种或多种：

**遗漏值。** 例如，有人忘记为某个房屋的年龄输入值。

**重复样本。** 例如，服务器错误地将同一条记录上传了两次。

**不良标签。** 例如，有人错误地将一颗橡树的图片标记为枫树。

**不良特征值。** 例如，有人输入了多余的位数，或者温度计被遗落在太阳底下。

一旦检测到存在这些问题，您通常需要将相应样本从数据集中移除，从而“修正”不良样本。要检测遗漏值或重复样本，您可以编写一个简单的程序。检测不良特征值或标签可能会比较棘手。

除了检测各个不良样本之外，您还必须检测集合中的不良数据。直方图是一种用于可视化集合中数据的很好机制。此外，收集如下统计信息也会有所帮助：

最大值和最小值

均值和中间值

标准偏差

考虑生成离散特征的最常见值列表。例如，country:uk 的样本数是否符合您的预期？language:jp 是否真的应该作为您数据集中的最常用语言？

了解数据

遵循以下规则：

记住您预期的数据状态。

确认数据是否满足这些预期（或者您可以解释为何数据不满足预期）。

仔细检查训练数据是否与其他来源（例如信息中心）的数据一致。

像处理任何任务关键型代码一样谨慎处理您的数据。良好的机器学习依赖于良好的数据。

* 1. **特征组合**

1. **对非线性规律进行编码**

在图 1 和图 2 中，我们做出如下假设：（蓝点代表生病的树，橙点代表健康的树）



**图 1. 这是线性问题吗？**

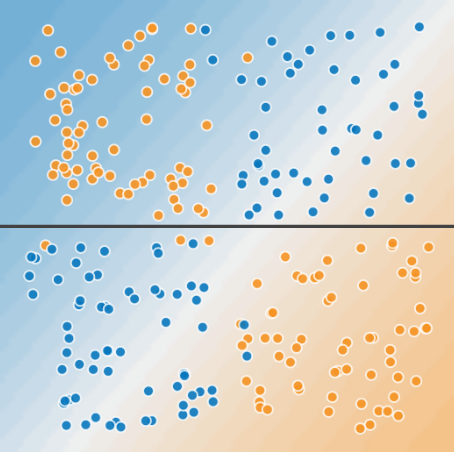
您可以画一条线将生病的树与健康的树清晰地分开吗？当然可以。这是个线性问题。这条线并不完美。有一两棵生病的树可能位于“健康”一侧，但您画的这条线可以很好地做出预测。

现在，我们来看看下图：



**图 2. 这是线性问题吗？**

您可以画一条直线将生病的树与健康的树清晰地分开吗？不，您做不到。这是个非线性问题。您画的任何一条线都不能很好地预测树的健康状况。



**图 3. 一条线无法分开两类数据。**

 要解决图 2 所示的非线性问题，可以创建一个特征组合。**特征组合**是指通过将两个或多个输入特征相乘来对特征空间中的非线性规律进行编码的合成特征。“cross”（组合）这一术语来自 [*cross product*](https://wikipedia.org/wiki/Cross_product)（向量积）。我们通过将 x1 与 x2 组合来创建一个名为 x3 的特征组合：

x3=x1x2

我们像处理任何其他特征一样来处理这个新建的 x3 特征组合。线性公式变为：

y=b+w1x1+w2x2+w3x3

线性算法可以算出 w3 的权重，就像算出 w1 和 w2 的权重一样。换言之，虽然 w3 表示非线性信息，但您不需要改变线性模型的训练方式来确定 w3 的值。

特征组合的种类

我们可以创建很多不同种类的特征组合。例如：

[A X B]：将两个特征的值相乘形成的特征组合。

[A x B x C x D x E]：将五个特征的值相乘形成的特征组合。

[A x A]：对单个特征的值求平方形成的特征组合。

通过采用[随机梯度下降法](https://developers.google.cn/machine-learning/crash-course/reducing-loss/stochastic-gradient-descent)，可以有效地训练线性模型。因此，在使用扩展的线性模型时辅以特征组合一直都是训练大规模数据集的有效方法。

1. **组合独热矢量**

到目前为止，我们已经重点介绍了如何对两个单独的浮点特征进行特征组合。在实践中，机器学习模型很少会组合连续特征。不过，机器学习模型却经常组合独热特征矢量，将独热特征矢量的特征组合视为逻辑连接。例如，假设我们具有以下两个特征：国家/地区和语言。对每个特征进行独热编码会生成具有二元特征的矢量，这些二元特征可解读为 country=USA, country=France 或 language=English, language=Spanish。然后，如果您对这些独热编码进行特征组合，则会得到可解读为逻辑连接的二元特征，如下所示：

  country:usa AND language:spanish  
再举一个例子，假设您对纬度和经度进行分箱，获得单独的独热 5 元素特征矢量。例如，指定的纬度和经度可以表示如下：

binned\_latitude = [0, 0, 0, 1, 0]

binned\_longitude = [0, 1, 0, 0, 0]

假设您对这两个特征矢量创建了特征组合：

binned\_latitude X binned\_longitude

此特征组合是一个 25 元素独热矢量（24 个 0 和 1 个 1）。该组合中的单个 1 表示纬度与经度的特定连接。然后，您的模型就可以了解到有关这种连接的特定关联性。

假设我们更粗略地对纬度和经度进行分箱，如下所示：

binned\_latitude(lat) = [

0 < lat <= 10

10 < lat <= 20

20 < lat <= 30

]

binned\_longitude(lon) = [

0 < lon <= 15

15 < lon <= 30

]

针对这些粗略分箱创建特征组合会生成具有以下含义的合成特征：

binned\_latitude\_X\_longitude(lat, lon) = [

0 < lat <= 10 AND 0 < lon <= 15

0 < lat <= 10 AND 15 < lon <= 30

10 < lat <= 20 AND 0 < lon <= 15

10 < lat <= 20 AND 15 < lon <= 30

20 < lat <= 30 AND 0 < lon <= 15

20 < lat <= 30 AND 15 < lon <= 30

]

现在，假设我们的模型需要根据以下两个特征来预测狗主人对狗狗的满意程度：

行为类型（吠叫、叫、偎依等）

时段

如果我们根据这两个特征构建以下特征组合：

[behavior type X time of day]

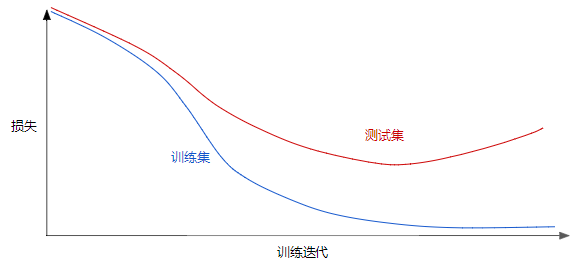
我们最终获得的预测能力将远远超过任一特征单独的预测能力。例如，如果狗狗在下午 5 点主人下班回来时（快乐地）叫喊，可能表示对主人满意度的正面预测结果。如果狗狗在凌晨 3 点主人熟睡时（也许痛苦地）哀叫，可能表示对主人满意度的强烈负面预测结果。

线性学习器可以很好地扩展到大量数据。对大规模数据集使用特征组合是学习高度复杂模型的一种有效策略。[神经网络](https://developers.google.cn/machine-learning/crash-course/introduction-to-neural-networks)可提供另一种策略。

* 1. **正则化：简单性**

1. **L2正则化**

请查看以下**泛化曲线**，该曲线显示的是训练集和验证集相对于训练迭代次数的损失。



**图 1. 训练集和验证集损失**

图 1 显示的是某个模型的训练损失逐渐减少，但验证损失最终增加。换言之，该泛化曲线显示该模型与训练集中的数据[过拟合](https://developers.google.cn/machine-learning/crash-course/generalization/peril-of-overfitting)。根据[奥卡姆剃刀定律](https://developers.google.cn/machine-learning/crash-course/generalization/peril-of-overfitting#occam)，或许我们可以通过降低复杂模型的复杂度来防止过拟合，这种原则称为**正则化**。

也就是说，并非只是以最小化损失（经验风险最小化）为目标：

minimize(Loss(Data|Model))

而是以最小化损失和复杂度为目标，这称为**结构风险最小化**：

minimize(Loss(Data|Model) + complexity(Model))

现在，我们的训练优化算法是一个由两项内容组成的函数：一个是**损失项**，用于衡量模型与数据的拟合度，另一个是**正则化项**，用于衡量模型复杂度。

机器学习速成课程重点介绍了两种衡量模型复杂度的常见方式（这两种方式有些相关）：

将模型复杂度作为模型中所有特征的权重的函数。

将模型复杂度作为具有非零权重的特征总数的函数。（[后面的一个单元](https://developers.google.cn/machine-learning/crash-course/regularization-for-sparsity/l1-regularization)介绍了这种方法。）

如果模型复杂度是权重的函数，则特征权重的绝对值越高，对模型复杂度的贡献就越大。

我们可以使用 **L2 正则化**公式来量化复杂度，该公式将正则化项定义为所有特征权重的平方和：

L2 regularization term=||w||22=w12+w22+...+wn2

在这个公式中，接近于 0 的权重对模型复杂度几乎没有影响，而离群值权重则可能会产生巨大的影响。

例如，某个线性模型具有以下权重：

{w1=0.2,w2=0.5,w3=5,w4=1,w5=0.25,w6=0.75}

L2 正则化项为 26.915：

w12+w22+w32+w42+w52+w62

=0.22+0.52+52+12+0.252+0.752

=0.04+0.25+25+1+0.0625+0.5625

=26.915

但是 w3（上述加粗内容）的平方值为 25，几乎贡献了全部的复杂度。所有 5 个其他权重的平方和对 L2 正则化项的贡献仅为 1.915。

1. **Lambda**

模型开发者通过以下方式来调整正则化项的整体影响：用正则化项的值乘以名为 lambda（又称为正则化率）的标量。也就是说，模型开发者会执行以下运算：

minimize(Loss(Data|Model)+λ complexity(Model))

执行 L2 正则化对模型具有以下影响

使权重值接近于 0（但并非正好为 0）

使权重的平均值接近于 0，且呈正态（钟形曲线或高斯曲线）分布。

增加 lambda 值将增强正则化效果。 例如，lambda 值较高的权重直方图可能会如图 2 所示。

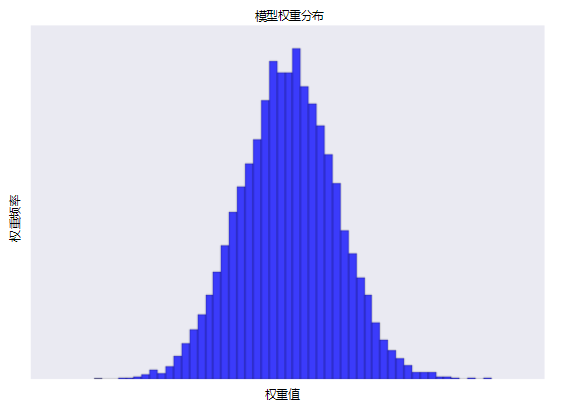


图 2. 权重直方图。

降低 lambda 的值往往会得出比较平缓的直方图，如图 3 所示。

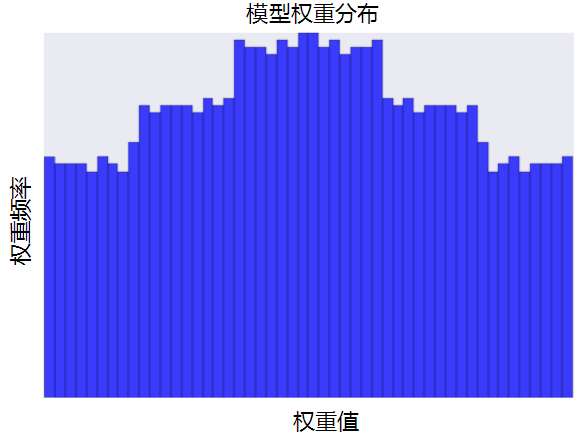


图 3. 较低的 lambda 值得出的权重直方图。

在选择 lambda 值时，目标是在简单化和训练数据拟合之间达到适当的平衡：

如果您的 lambda 值过高，则模型会非常简单，但是您将面临数据欠拟合的风险。您的模型将无法从训练数据中获得足够的信息来做出有用的预测。

如果您的 lambda 值过低，则模型会比较复杂，并且您将面临数据过拟合的风险。您的模型将因获得过多训练数据特点方面的信息而无法泛化到新数据。

*注意：将 lambda 设为 0 可彻底取消正则化。 在这种情况下，训练的唯一目的将是最小化损失，而这样做会使过拟合的风险达到最高。*

理想的 lambda 值生成的模型可以很好地泛化到以前未见过的新数据。 遗憾的是，理想的 lambda 值取决于数据，因此您需要手动或自动进行一些调整。

*学习速率和 lambda 之间存在密切关联。强 L2 正则化值往往会使特征权重更接近于 0。较低的学习速率（使用早停法）通常会产生相同的效果，因为与 0 的距离并不是很远。 因此，同时调整学习速率和 lambda 可能会产生令人混淆的效果。*

*早停法指的是在模块完全收敛之前就结束训练。在实际操作中，我们经常在以*[*在线*](https://developers.google.cn/machine-learning/crash-course/production_training)*（连续）方式进行训练时采取一些隐式早停法。也就是说，一些新趋势的数据尚不足以收敛。*

*如上所述，更改正则化参数产生的效果可能会与更改学习速率或迭代次数产生的效果相混淆。一种有用的做法（在训练一批固定的数据时）是执行足够多次迭代，这样早停法便不会起作用。*

* 1. **逻辑回归**

1. **计算概率**

许多问题需要将概率估算值作为输出。逻辑回归是一种极其高效的概率计算机制。实际上，您可以通过下两种方式之一使用返回的概率：

“按原样”

转换成二元类别。

我们来了解一下如何“按原样”使用概率。假设我们创建一个逻辑回归模型来预测狗在半夜发出叫声的概率。我们将此概率称为：

p(bark | night)

如果逻辑回归模型预测 p(bark | night) 的值为 0.05，那么一年内，狗的主人应该被惊醒约 18 次：

startled = p(bark | night) \* nights

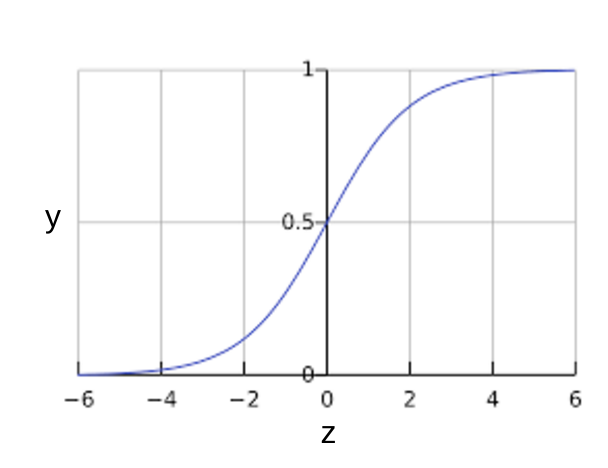
18 ~= 0.05 \* 365

在很多情况下，您会将逻辑回归输出映射到二元分类问题的解决方案，该二元分类问题的目标是正确预测两个可能的标签（例如，“垃圾邮件”或“非垃圾邮件”）中的一个。之后的[单元](https://developers.google.cn/machine-learning/crash-course/classification/video-lecture)会重点介绍这一内容。

您可能想知道逻辑回归模型如何确保输出值始终落在 0 和 1 之间。巧合的是，**S 型函数**生成的输出值正好具有这些特性，其定义如下：

C:\Users\Administrator\AppData\Roaming\Tencent\Users\188512936\TIM\WinTemp\RichOle\J_C0K(JRJELB}~LM)]ID~IT.png

S 型函数会产生以下曲线图：



**图 1：S 型函数**

如果 z 表示使用逻辑回归训练的模型的线性层的输出，则 S 型(z) 函数会生成一个介于 0 和 1 之间的值（概率）。用数学方法表示为：

C:\Users\Administrator\AppData\Roaming\Tencent\Users\188512936\TIM\WinTemp\RichOle\}L57R%}67))0$JBP4QSUE}T.png

其中：

y' 是逻辑回归模型针对特定样本的输出。

z 是 b + w1x1 + w2x2 + … wNxN

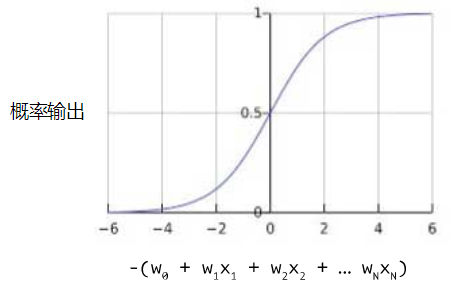
“w”值是该模型学习的权重和偏差。

“x”值是特定样本的特征值。

请注意，z 也称为对数几率，因为 S 型函数的反函数表明，z 可定义为标签“1”（例如“狗叫”）的概率除以标签“0”（例如“狗不叫”）的概率得出的值的对数：

C:\Users\Administrator\AppData\Roaming\Tencent\Users\188512936\TIM\WinTemp\RichOle\`54JU~A71PVUP{()FBEZDB7.png

以下是具有机器学习标签的 S 型函数：



**图 2：逻辑回归输出。**

|  |
| --- |
| 假设我们的逻辑回归模型具有学习了下列偏差和权重的三个特征：  b = 1  w1 = 2  w2 = -1  w3 = 5  进一步假设给定样本具有以下特征值：  x1 = 0  x2 = 10  x3 = 2  因此，对数几率：  b+w1x1+w2x2+w3x3  将是：  (1) + (2)(0) + (-1)(10) + (5)(2) = 1  因此，此特定样本的逻辑回归预测值将是 0.731：  C:\Users\Administrator\AppData\Roaming\Tencent\Users\188512936\TIM\WinTemp\RichOle\1)Q@TNQC5GL}PZEW{ZZ4WUW.png  C:\Users\Administrator\AppData\Roaming\Tencent\Users\188512936\TIM\WinTemp\RichOle\]}G}`@PX2}I(12K)XP51V4P.png  **图 3：73.1% 的概率** |

1. **模型训练**

逻辑回归的损失函数

线性回归的损失函数是平方损失。逻辑回归的损失函数是**对数损失函数**，定义如下：

![C:\Users\Administrator\AppData\Roaming\Tencent\Users\188512936\TIM\WinTemp\RichOle\{BU[BBD8RLM%]X8T1W](HSI.png](data:image/png;base64,)

其中：

(*xy*)ϵ*D* 是包含很多有标签样本 (*x*,*y*) 的数据集。

“y”是有标签样本中的标签。由于这是逻辑回归，因此“y”的每个值必须是 0 或 1。

“y'”是对于特征集“x”的预测值（介于 0 和 1 之间）。

对数损失函数的方程式与 [Shannon 信息论中的熵测量](https://wikipedia.org/wiki/Entropy_(information_theory))密切相关。它也是[似然函数](https://wikipedia.org/wiki/Likelihood_function)的负对数（假设“y”属于[伯努利分布](https://wikipedia.org/wiki/Bernoulli_distribution)）。实际上，最大限度地降低损失函数的值会生成最大的似然估计值。

逻辑回归中的正则化

[正则化](https://developers.google.cn/machine-learning/crash-course/regularization-for-simplicity/video-lecture)在逻辑回归建模中极其重要。如果没有正则化，逻辑回归的渐近性会不断促使损失在高维度空间内达到 0。因此，大多数逻辑回归模型会使用以下两个策略之一来降低模型复杂性：

L2 正则化。

早停法，即，限制训练步数或学习速率。

（我们会在[之后的单元](https://developers.google.cn/machine-learning/crash-course/regularization-for-sparsity/video-lecture)中讨论第三个策略，即 L1 正则化。）

假设您向每个样本分配一个唯一 ID，且将每个 ID 映射到其自己的特征。如果您未指定正则化函数，模型会变得完全过拟合。这是因为模型会尝试促使所有样本的损失达到 0 但始终达不到，从而使每个指示器特征的权重接近正无穷或负无穷。当有大量罕见的特征组合且每个样本中仅一个时，包含特征组合的高维度数据会出现这种情况。

幸运的是，使用 L2 或早停法可以防止出现此类问题。

*总结:*

*逻辑回归模型会生成概率。*

*对数损失函数是逻辑回归的损失函数。*

*逻辑回归被很多从业者广泛使用。*

* 1. **分类**

1. **阈值**

逻辑回归返回的是概率。您可以“原样”使用返回的概率（例如，用户点击此广告的概率为 0.00023），也可以将返回的概率转换成二元值（例如，这封电子邮件是垃圾邮件）。

如果某个逻辑回归模型对某封电子邮件进行预测时返回的概率为 0.9995，则表示该模型预测这封邮件非常可能是垃圾邮件。相反，在同一个逻辑回归模型中预测分数为 0.0003 的另一封电子邮件很可能不是垃圾邮件。可如果某封电子邮件的预测分数为 0.6 呢？为了将逻辑回归值映射到二元类别，您必须指定分类阈值（也称为判定阈值）。如果值高于该阈值，则表示“垃圾邮件”；如果值低于该阈值，则表示“非垃圾邮件”。人们往往会认为分类阈值应始终为 0.5，但阈值取决于具体问题，因此您必须对其进行调整。

我们将在后面的部分中详细介绍可用于对分类模型的预测进行评估的指标，以及更改分类阈值对这些预测的影响。

注意：“调整”逻辑回归的阈值不同于调整学习速率等超参数。在选择阈值时，需要评估您将因犯错而承担多大的后果。例如，将非垃圾邮件误标记为垃圾邮件会非常糟糕。不过，虽然将垃圾邮件误标记为非垃圾邮件会令人不快，但应该不会让您丢掉工作。

1. **阳性与阴性；正分类与负分类**

在本部分，我们将定义用于评估分类模型的指标的主要组成部分。不过，我们先来看一则寓言故事：

**伊索寓言：狼来了（精简版）**

有一位牧童要照看镇上的羊群，但是他开始厌烦这份工作。为了找点乐子，他大喊道：“狼来了！”其实根本一头狼也没有出现。村民们迅速跑来保护羊群，但他们发现这个牧童是在开玩笑后非常生气。

[这样的情形重复出现了很多次。]

一天晚上，牧童看到真的有一头狼靠近羊群，他大声喊道：“狼来了！”村民们不想再被他捉弄，都待在家里不出来。这头饥饿的狼对羊群大开杀戒，美美饱餐了一顿。这下子，整个镇子都揭不开锅了。恐慌也随之而来。

我们做出以下定义：

“狼来了”是**正类别**。

“没有狼”是**负类别**。

我们可以使用一个 2x2 [混淆矩阵](https://developers.google.cn/machine-learning/crash-course/glossary#confusion_matrix)来总结我们的“狼预测”模型，该矩阵描述了所有可能出现的结果（共四种）：

|  |  |
| --- | --- |
| 真正例 (TP)：   * 真实情况：受到狼的威胁。 * 牧童说：“狼来了。” * 结果：牧童是个英雄。 | 假正例 (FP)：   * 真实情况：没受到狼的威胁。 * 牧童说：“狼来了。” * 结果：村民们因牧童吵醒他们而感到非常生气。 |
| 假负例 (FN)：   * 真实情况：受到狼的威胁。 * 牧童说：“没有狼”。 * 结果：狼吃掉了所有的羊。 | 真负例 (TN)：   * 真实情况：没受到狼的威胁。 * 牧童说：“没有狼”。 * 结果：大家都没事。 |

**真正例**是指模型将正类别样本正确地预测为正类别。同样，**真负例**是指模型将负类别样本正确地预测为负类别。

**假正例**是指模型将负类别样本错误地预测为正类别，而**假负例**是指模型将正类别样本错误地预测为负类别。

在后面的部分中，我们将介绍如何使用从这四种结果中衍生出的指标来评估分类模型。

1. **准确率**

准确率是一个用于评估分类模型的指标。通俗来说，**准确率**是指我们的模型预测正确的结果所占的比例。正式点说，准确率的定义如下：

Accuracy=Number of correct predictions/Total number of predictions

对于二元分类，也可以根据正类别和负类别按如下方式计算准确率：

Accuracy=(TP+TN)/(TP+TN+FP+FN)

其中，TP = 真正例，TN = 真负例，FP = 假正例，FN = 假负例。

让我们来试着计算一下以下模型的准确率，该模型将 100 个肿瘤分为[恶性](https://wikipedia.org/wiki/Malignancy) （正类别）或[良性](https://wikipedia.org/wiki/Benign_tumor)（负类别）：

|  |  |
| --- | --- |
| 真正例 (TP)：   * 真实情况：恶性 * 机器学习模型预测的结果：恶性 * **TP 结果数：1** | 假正例 (FP)：   * 真实情况：良性 * 机器学习模型预测的结果：恶性 * **FP 结果数：1** |
| 假负例 (FN)：   * 真实情况：恶性 * 机器学习模型预测的结果：良性 * **FN 结果数：8** | 真负例 (TN)：   * 真实情况：良性 * 机器学习模型预测的结果：良性 * **TN 结果数：90** |

Accuracy=(TP+TN)/(TP+TN+FP+FN)=(1+90)/(1+90+1+8)=0.91

准确率为 0.91，即 91%（总共 100 个样本中有 91 个预测正确）。这表示我们的肿瘤分类器在识别恶性肿瘤方面表现得非常出色，对吧？

实际上，只要我们仔细分析一下正类别和负类别，就可以更好地了解我们模型的效果。

在 100 个肿瘤样本中，91 个为良性（90 个 TN 和 1 个 FP），9 个为恶性（1 个 TP 和 8 个 FN）。

在 91 个良性肿瘤中，该模型将 90 个正确识别为良性。这很好。不过，在 9 个恶性肿瘤中，该模型仅将 1 个正确识别为恶性。这是多么可怕的结果！9 个恶性肿瘤中有 8 个未被诊断出来！

虽然 91% 的准确率可能乍一看还不错，但如果另一个肿瘤分类器模型总是预测良性，那么这个模型使用我们的样本进行预测也会实现相同的准确率（100 个中有 91 个预测正确）。换言之，我们的模型与那些没有预测能力来区分恶性肿瘤和良性肿瘤的模型差不多。

当您使用**分类不平衡的数据集**（比如正类别标签和负类别标签的数量之间存在明显差异）时，单单准确率一项并不能反映全面情况。

在下一部分中，我们将介绍两个能够更好地评估分类不平衡问题的指标：精确率和召回率。

1. **精确率和召回率**

精确率

**精确率**指标尝试回答以下问题：

在被识别为正类别的样本中，确实为正类别的比例是多少？

精确率的定义如下：

Precision=TP/(TP+FP)

**注意**：如果模型的预测结果中没有假正例，则模型的精确率为 1.0。

让我们来计算一下[上一部分](https://developers.google.cn/machine-learning/crash-course/classification/accuracy)中用于分析肿瘤的机器学习模型的精确率：

|  |  |
| --- | --- |
| 真正例 (TP)：1 | 假正例 (FP)：1 |
| 假负例 (FN)：8 | 真负例 (TN)：90 |

精确率精确率=TP/(TP+FP)=1/(1+1)=0.5

该模型的精确率为 0.5，也就是说，该模型在预测恶性肿瘤方面的正确率是 50%。

召回率

**召回率**尝试回答以下问题：在所有正类别样本中，被正确识别为正类别的比例是多少？

从数学上讲，召回率的定义如下：

召回率召回率=TP/(TP+FN)

**注意**：如果模型的预测结果中没有假负例，则模型的召回率为 1.0。

让我们来计算一下肿瘤分类器的召回率：

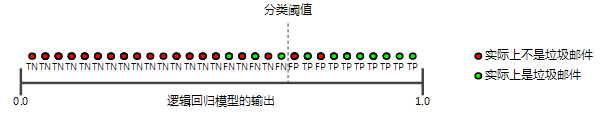
|  |  |
| --- | --- |
| 真正例 (TP)：1 | 假正例 (FP)：1 |
| 假负例 (FN)：8 | 真负例 (TN)：90 |

召回率召回率=TP/(TP+FN)=1/(1+8)=0.11

该模型的召回率是 0.11，也就是说，该模型能够正确识别出所有恶性肿瘤的百分比是 11%。

精确率和召回率：一场拔河比赛

要全面评估模型的有效性，必须**同时**检查精确率和召回率。遗憾的是，精确率和召回率往往是此消彼长的情况。也就是说，提高精确率通常会降低召回率值，反之亦然。请观察下图来了解这一概念，该图显示了电子邮件分类模型做出的 30 项预测。分类阈值右侧的被归类为“垃圾邮件”，左侧的则被归类为“非垃圾邮件”。



**图 1. 将电子邮件归类为垃圾邮件或非垃圾邮件**

我们根据图 1 所示的结果来计算精确率和召回率值：

|  |  |
| --- | --- |
| 真正例 (TP)：8 | 假正例 (FP)：2 |
| 假负例 (FN)：3 | 真负例 (TN)：17 |

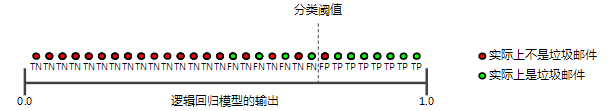
精确率指的是**被标记为垃圾邮件的电子邮件**中正确分类的电子邮件所占的百分比，即图 1 中阈值线右侧的绿点所占的百分比：

Precision=TP/(TP+FP)=8/(8+2)=0.8

召回率指的是**实际垃圾邮件**中正确分类的电子邮件所占的百分比，即图 1 中阈值线右侧的绿点所占的百分比：

Recall=TP/(TP+FN)=8/(8+3)=0.73

图 2 显示了提高分类阈值产生的效果。



**图 2. 提高分类阈值**

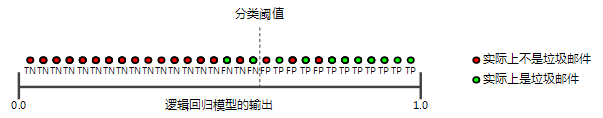
假正例数量会减少，但假负例数量会相应地增加。结果，精确率有所提高，而召回率则有所降低：

|  |  |
| --- | --- |
| 真正例 (TP)：7 | 假正例 (FP)：1 |
| 假负例 (FN)：4 | 真负例 (TN)：18 |

Precision=TP/(TP+FP)=7/(7+1)=0.88

Recall=TP/(TP+FN)=7/(7+4)=0.64

相反，图 3 显示了降低分类阈值（从图 1 中的初始位置开始）产生的效果。



**图 3. 降低分类阈值**

假正例数量会增加，而假负例数量会减少。结果这一次，精确率有所降低，而召回率则有所提高：

|  |  |
| --- | --- |
| 真正例 (TP)：9 | 假正例 (FP)：3 |
| 假负例 (FN)：2 | 真负例 (TN)：16 |

Precision=TPTP+FP=99+3=0.75

Recall=TPTP+FN=99+2=0.82

我们已根据精确率和召回率指标制定了各种指标。有关示例，请参阅 [F1 值](https://wikipedia.org/wiki/F1_score)。

1. **ROC和曲线下面积**

**ROC 曲线**（**接收者操作特征曲线**）是一种显示分类模型在所有分类阈值下的效果的图表。该曲线绘制了以下两个参数：

真正例率

假正例率

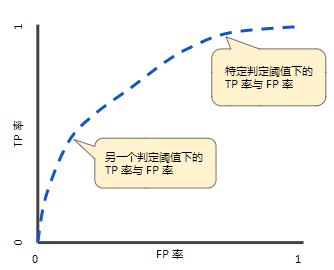
**真正例率** (**TPR**) 是召回率的同义词，因此定义如下：

TPR=TP/(TP+FN)

**假正例率** (**FPR**) 的定义如下：

FPR=FP/(FP+TN)

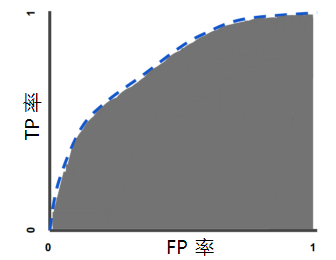
ROC 曲线用于绘制采用不同分类阈值时的 TPR 与 FPR。降低分类阈值会导致将更多样本归为正类别，从而增加假正例和真正例的个数。下图显示了一个典型的 ROC 曲线。



**图 4. 不同分类阈值下的 TP 率与 FP 率**

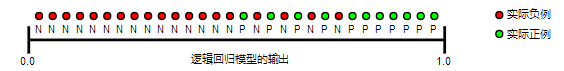
为了计算 ROC 曲线上的点，我们可以使用不同的分类阈值多次评估逻辑回归模型，但这样做效率非常低。幸运的是，有一种基于排序的高效算法可以为我们提供此类信息，这种算法称为曲线下面积。

**曲线下面积**表示“ROC 曲线下面积”。也就是说，曲线下面积测量的是从 (0,0) 到 (1,1) 之间整个 ROC 曲线以下的整个二维面积（参考积分学）。



**图 5. 曲线下面积（ROC 曲线下面积）**

曲线下面积对所有可能的分类阈值的效果进行综合衡量。曲线下面积的一种解读方式是看作模型将某个随机正类别样本排列在某个随机负类别样本之上的概率。以下面的样本为例，逻辑回归预测从左到右以升序排列：



**图 6. 预测按逻辑回归分数以升序排列**

曲线下面积表示随机正类别（绿色）样本位于随机负类别（红色）样本右侧的概率。

曲线下面积的取值范围为 0-1。预测结果 100% 错误的模型的曲线下面积为 0.0；而预测结果 100% 正确的模型的曲线下面积为 1.0。

曲线下面积因以下两个原因而比较实用：

曲线下面积的**尺度不变**。它测量预测的排名情况，而不是测量其绝对值。

曲线下面积的**分类阈值不变**。它测量模型预测的质量，而不考虑所选的分类阈值。

不过，这两个原因都有各自的局限性，这可能会导致曲线下面积在某些用例中不太实用：

**并非总是希望尺度不变。** 例如，有时我们非常需要被良好校准的概率输出，而曲线下面积无法告诉我们这一结果。

**并非总是希望分类阈值不变。** 在假负例与假正例的代价存在较大差异的情况下，尽量减少一种类型的分类错误可能至关重要。例如，在进行垃圾邮件检测时，您可能希望优先考虑尽量减少假正例（即使这会导致假负例大幅增加）。对于此类优化，曲线下面积并非一个实用的指标。

1. **预测偏差**

逻辑回归预测应当无偏差。即：“预测平均值”应当约等于“观察平均值”

**预测偏差**指的是这两个平均值之间的差值。即：

预测偏差=预测平均值−数据集中相应标签的平均值

***注意****：“预测偏差”与*[*偏差*](https://developers.google.cn/machine-learning/crash-course/descending-into-ml)*（“wx + b”中的“b”）不是一回事。*

如果出现非常高的非零预测偏差，则说明模型某处存在错误，因为这表明模型对正类别标签的出现频率预测有误。

例如，假设我们知道，所有电子邮件中平均有 1% 的邮件是垃圾邮件。如果我们对某一封给定电子邮件一无所知，则预测它是垃圾邮件的可能性为 1%。同样，一个出色的垃圾邮件模型应该预测到电子邮件平均有 1% 的可能性是垃圾邮件。（换言之，如果我们计算单个电子邮件是垃圾邮件的预测可能性的平均值，则结果应该是 1%。）然而，如果该模型预测电子邮件是垃圾邮件的平均可能性为 20%，那么我们可以得出结论，该模型出现了预测偏差。

造成预测偏差的可能原因包括：

特征集不完整

数据集混乱

模型实现流水线中有错误？

训练样本有偏差

正则化过强

您可能会通过对学习模型进行后期处理来纠正预测偏差，即通过添加**校准层**来调整模型的输出，从而减小预测偏差。例如，如果您的模型存在 3% 以上的偏差，则可以添加一个校准层，将平均预测偏差降低 3%。但是，添加校准层并非良策，具体原因如下：

您修复的是症状，而不是原因。

您建立了一个更脆弱的系统，并且必须持续更新。

如果可能的话，请避免添加校准层。使用校准层的项目往往会对其产生依赖 - 使用校准层来修复模型的所有错误。最终，维护校准层可能会令人苦不堪言。

***注意****：出色模型的偏差通常接近于零。即便如此，预测偏差低并不能证明您的模型比较出色。特别糟糕的模型的预测偏差也有可能为零。例如，只能预测所有样本平均值的模型是糟糕的模型，尽管其预测偏差为零。*

分桶偏差和预测偏差

逻辑回归可预测 0 到 1 之间的值。不过，所有带标签样本都正好是 0（例如，0 表示“非垃圾邮件”）或 1（例如，1 表示“垃圾邮件”）。因此，在检查预测偏差时，您无法仅根据一个样本准确地确定预测偏差；您必须在“一大桶”样本中检查预测偏差。也就是说，只有将足够的样本组合在一起以便能够比较预测值（例如 0.392）与观察值（例如 0.394），逻辑回归的预测偏差才有意义。

您可以通过以下方式构建桶：

以线性方式分解目标预测。

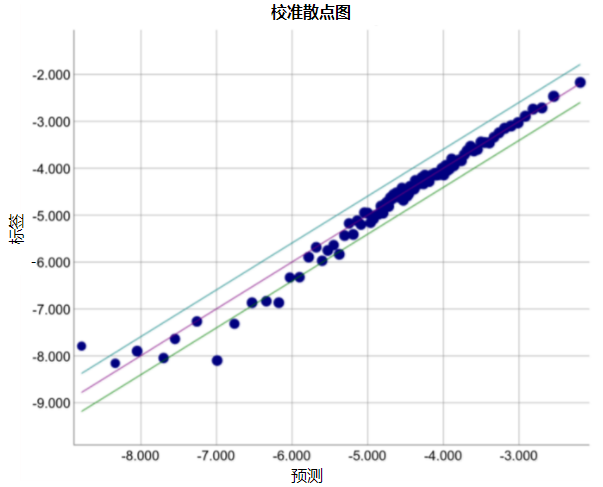
构建分位数。

请查看以下某个特定模型的校准曲线。每个点表示包含 1000 个值的分桶。两个轴具有以下含义：

x 轴表示模型针对该桶预测的平均值。

y 轴表示该桶的数据集中的实际平均值。

两个轴均采用对数尺度。



**图 8. 预测偏差曲线（对数尺度）**

为什么只有模型的某些部分所做的预测如此糟糕？以下是几种可能性：

训练集不能充分表示数据空间的某些子集。

数据集的某些子集比其他子集更混乱。

该模型过于[正则化](https://developers.google.cn/machine-learning/crash-course/regularization-for-simplicity/video-lecture)。（不妨减小 [lambda](https://developers.google.cn/machine-learning/crash-course/glossary#lambda) 的值。）

* 1. **正则化：稀疏性**

**L1正则化**

稀疏矢量通常包含许多维度。创建[特征组合](https://developers.google.cn/machine-learning/crash-course/feature-crosses/video-lecture)会导致包含更多维度。由于使用此类高维度特征矢量，因此模型可能会非常庞大，并且需要大量的 RAM。

在高维度稀疏矢量中，最好尽可能使权重正好降至 0。正好为 0 的权重基本上会使相应特征从模型中移除。 将特征设为 0 可节省 RAM 空间，且可以减少模型中的噪点。

以一个涵盖全球地区（不仅仅只是涵盖加利福尼亚州）的住房数据集为例。如果按分（每度为 60 分）对全球纬度进行分桶，则在一次稀疏编码过程中会产生大约 1 万个维度；如果按分对全球经度进行分桶，则在一次稀疏编码过程中会产生大约 2 万个维度。这两种特征的特征组合会产生大约 2 亿个维度。这 2 亿个维度中的很多维度代表非常有限的居住区域（例如海洋里），很难使用这些数据进行有效泛化。 若为这些不需要的维度支付 RAM 存储费用就太不明智了。 因此，最好是使无意义维度的权重正好降至 0，这样我们就可以避免在推理时支付这些模型系数的存储费用。

我们或许可以添加适当选择的正则化项，将这种想法变成在训练期间解决的优化问题。

L2 正则化能完成此任务吗？遗憾的是，不能。 L2 正则化可以使权重变小，但是并不能使它们正好为 0.0。

另一种方法是尝试创建一个正则化项，减少模型中的非零系数值的计数。只有在模型能够与数据拟合时增加此计数才有意义。 遗憾的是，虽然这种基于计数的方法看起来很有吸引力，但它会将我们的凸优化问题变为非凸优化问题，即 [NP 困难](https://wikipedia.org/wiki/NP-hardness)。 （如果您仔细观察，便会发现它与背包问题关联。） 因此，L0 正则化这种想法在实践中并不是一种有效的方法。

不过，L1 正则化这种正则化项的作用类似 L0，但它具有凸优化的优势，可有效进行计算。因此，我们可以使用 L1 正则化使模型中很多信息缺乏的系数正好为 0，从而在推理时节省 RAM。

L1 和 L2 正则化。

L2 和 L1 采用不同的方式降低权重：

L2 会降低权重2。

L1 会降低 |权重|。

因此，L2 和 L1 具有不同的导数：

L2 的导数为 2 \* 权重。

L1 的导数为 k（一个常数，其值与权重无关）。

您可以将 L2 的导数的作用理解为每次移除权重的 x%。如 [Zeno](https://wikipedia.org/wiki/Zeno%27s_paradoxes#Dichotomy_paradox) 所知，对于任意数字，即使按每次减去 x% 的幅度执行数十亿次减法计算，最后得出的值也绝不会正好为 0。（Zeno 不太熟悉浮点精度限制，它可能会使结果正好为 0。）总而言之，L2 通常不会使权重变为 0。

您可以将 L1 的导数的作用理解为每次从权重中减去一个常数。不过，由于减去的是绝对值，L1 在 0 处具有不连续性，这会导致与 0 相交的减法结果变为 0。例如，如果减法使权重从 +0.1 变为 -0.2，L1 便会将权重设为 0。就这样，L1 使权重变为 0 了。

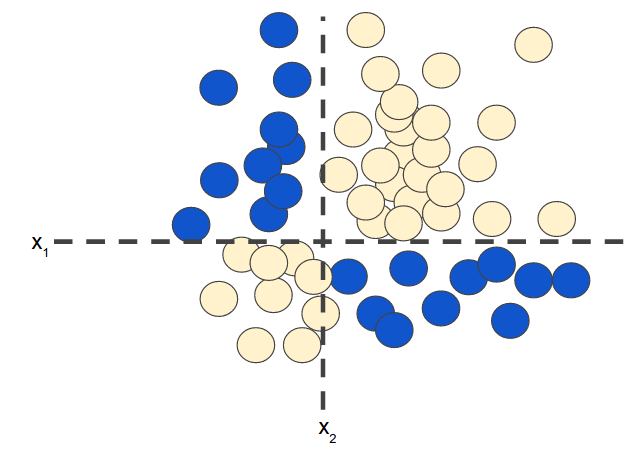
L1 正则化 - 减少所有权重的绝对值 - 证明对宽度模型非常有效。

请注意，该说明适用于一维模型。

* 1. **神经网络简介**

**剖析**

如果您还记得[特征组合](https://developers.google.cn/machine-learning/crash-course/feature-crosses/video-lecture)这一单元的话，就会发现以下分类问题属于非线性问题：



**图 1. 非线性分类问题**

“非线性”意味着您无法使用形式为

b+w1x1+w2x2

的模型准确预测标签。也就是说，“决策面”不是直线。之前，我们了解了对非线性问题进行建模的一种可行方法 - [特征组合](https://developers.google.cn/machine-learning/crash-course/feature-crosses/video-lecture)。

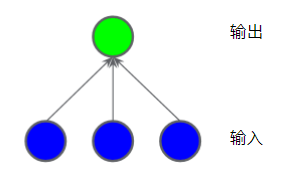
现在，请考虑以下数据集：



**图 2. 更难的非线性分类问题**

图 2 所示的数据集问题无法用线性模型解决。

为了了解神经网络可以如何帮助解决非线性问题，我们首先用图表呈现一个线性模型：



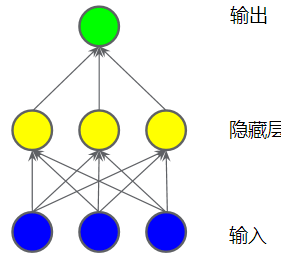
**图 3. 用图表呈现的线性模型**

每个蓝色圆圈均表示一个输入特征，绿色圆圈表示各个输入的加权和。

要提高此模型处理非线性问题的能力，我们可以如何更改它？

隐藏层

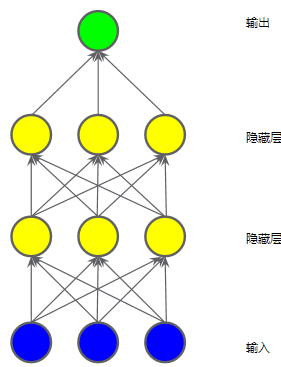
在下图所示的模型中，我们添加了一个表示中间值的“隐藏层”。隐藏层中的每个黄色节点均是蓝色输入节点值的加权和。输出是黄色节点的加权和。



**图 4. 两层模型的图表**

此模型是线性的吗？是的，其输出仍是其输入的线性组合。

在下图所示的模型中，我们又添加了一个表示加权和的“隐藏层”。



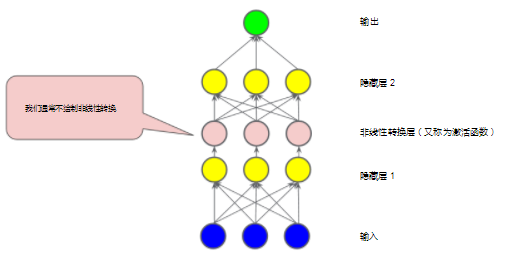
**图 5. 三层模型的图表**

此模型仍是线性的吗？是的，没错。当您将输出表示为输入的函数并进行简化时，您只是获得输入的另一个加权和而已。该加权和无法对图 2 中的非线性问题进行有效建模。

激活函数

要对非线性问题进行建模，我们可以直接引入非线性函数。我们可以用非线性函数将每个隐藏层节点像管道一样连接起来。

在下图所示的模型中，在隐藏层 1 中的各个节点的值传递到下一层进行加权求和之前，我们采用一个非线性函数对其进行了转换。这种非线性函数称为激活函数。



**图 6. 包含激活函数的三层模型的图表**

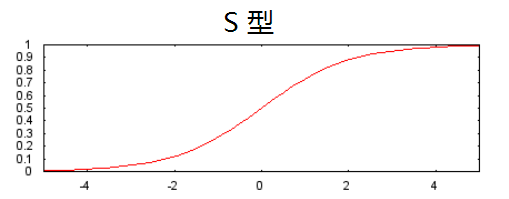
现在，我们已经添加了激活函数，如果添加层，将会产生更多影响。通过在非线性上堆叠非线性，我们能够对输入和预测输出之间极其复杂的关系进行建模。简而言之，每一层均可通过原始输入有效学习更复杂、更高级别的函数。如果您想更直观地了解这一过程的工作原理，请参阅 [Chris Olah 的精彩博文](http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/)。

常见激活函数

以下 **S 型**激活函数将加权和转换为介于 0 和 1 之间的值。

C:\Users\Administrator\AppData\Roaming\Tencent\Users\188512936\TIM\WinTemp\RichOle\U7112)HFIW{C73LOWH6`S_B.png

曲线图如下：

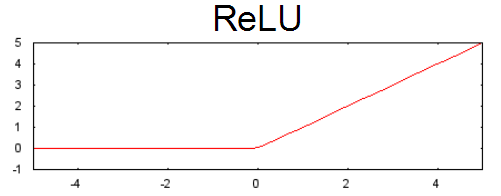


**图 7. S 型激活函数**

相较于 S 型函数等平滑函数，以下**修正线性单元**激活函数（简称为 **ReLU**）的效果通常要好一点，同时还非常易于计算。

F(x)=max(0,x)

ReLU 的优势在于它基于实证发现（可能由 ReLU 驱动），拥有更实用的响应范围。S 型函数的响应性在两端相对较快地减少。



**图 8. ReLU 激活函数**

实际上，所有数学函数均可作为激活函数。假设 σ 表示我们的激活函数（ReLU、S 型函数等等）。因此，网络中节点的值由以下公式指定：

σ(w⋅x+b)

TensorFlow 为[各种激活函数](https://tensorflow.google.cn/api_docs/python/nn.html)提供开箱即用型支持。但是，我们仍建议从 ReLU 着手。

**总结**

现在，我们的模型拥有了人们通常所说的“神经网络”的所有标准组件：

一组节点，类似于神经元，位于层中。

一组权重，表示每个神经网络层与其下方的层之间的关系。下方的层可能是另一个神经网络层，也可能是其他类型的层。

一组偏差，每个节点一个偏差。

一个激活函数，对层中每个节点的输出进行转换。不同的层可能拥有不同的激活函数。

警告：神经网络不一定始终比特征组合好，但它确实可以提供适用于很多情形的灵活替代方案。

* 1. **训练神经网络**

**最佳做法**

本部分介绍了反向传播算法的失败案例，以及正则化神经网络的常见方法。

失败案例

很多常见情况都会导致反向传播算法出错。

梯度消失

较低层（更接近输入）的梯度可能会变得非常小。在深度网络中，计算这些梯度时，可能涉及许多小项的乘积。

当较低层的梯度逐渐消失到 0 时，这些层的训练速度会非常缓慢，甚至不再训练。

ReLU 激活函数有助于防止梯度消失。

梯度爆炸

如果网络中的权重过大，则较低层的梯度会涉及许多大项的乘积。在这种情况下，梯度就会爆炸：梯度过大导致难以收敛。

批标准化可以降低学习速率，因而有助于防止梯度爆炸。

ReLU 单元消失

一旦 ReLU 单元的加权和低于 0，ReLU 单元就可能会停滞。它会输出对网络输出没有任何贡献的 0 激活，而梯度在反向传播算法期间将无法再从中流过。由于梯度的来源被切断，ReLU 的输入可能无法作出足够的改变来使加权和恢复到 0 以上。

降低学习速率有助于防止 ReLU 单元消失。

丢弃正则化

这是称为**丢弃**的另一种形式的正则化，可用于神经网络。其工作原理是，在梯度下降法的每一步中随机丢弃一些网络单元。丢弃得越多，正则化效果就越强：

0.0 = 无丢弃正则化。

1.0 = 丢弃所有内容。模型学不到任何规律。

0.0 和 1.0 之间的值更有用。

* 1. **多类别神经网络**

1. **一对多**

一对多提供了一种利用二元分类的方法。鉴于一个分类问题会有 N 个可行的解决方案，一对多解决方案包括 N 个单独的二元分类器，每个可能的结果对应一个二元分类器。在训练期间，模型会训练一系列二元分类器，使每个分类器都能回答单独的分类问题。以一张狗狗的照片为例，可能需要训练五个不同的识别器，其中四个将图片看作负样本（不是狗狗），一个将图片看作正样本（是狗狗）。即：

这是一张苹果的图片吗？不是。

这是一张熊的图片吗？不是。

这是一张糖果的图片吗？不是。

这是一张狗狗的图片吗？是。

这是一张鸡蛋的图片吗？不是。

当类别总数较少时，这种方法比较合理，但随着类别数量的增加，其效率会变得越来越低下。

我们可以借助深度神经网络（在该网络中，每个输出节点表示一个不同的类别）创建明显更加高效的一对多模型。下图展示了这种方法：

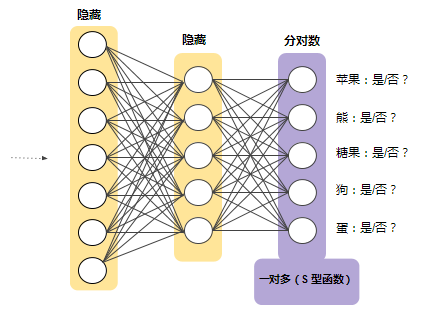


图 1. 一对多神经网络

1. **Softmax**

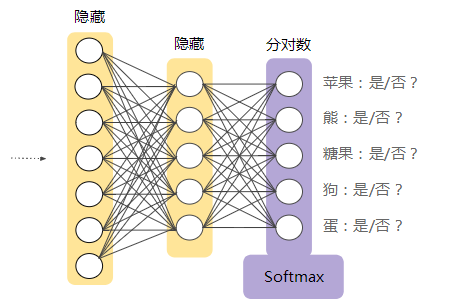
我们已经知道，[逻辑回归](https://developers.google.cn/machine-learning/crash-course/logistic-regression)可生成介于 0 和 1.0 之间的小数。例如，某电子邮件分类器的逻辑回归输出值为 0.8，表明电子邮件是垃圾邮件的概率为 80%，不是垃圾邮件的概率为 20%。很明显，一封电子邮件是垃圾邮件或非垃圾邮件的概率之和为 1.0。

**Softmax** 将这一想法延伸到多类别领域。也就是说，在多类别问题中，Softmax 会为每个类别分配一个用小数表示的概率。这些用小数表示的概率相加之和必须是 1.0。与其他方式相比，这种附加限制有助于让训练过程更快速地收敛。

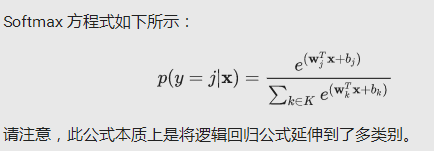
例如，回到我们在图 1 中看到的图片分析示例，Softmax 可能会得出图片属于某一特定类别的以下概率：

| 类别 | 概率 |
| --- | --- |
| 苹果 | 0.001 |
| 熊 | 0.04 |
| 糖果 | 0.008 |
| 狗狗 | 0.95 |
| 鸡蛋 | 0.001 |

Softmax 层是紧挨着输出层之前的神经网络层。Softmax 层必须和输出层拥有一样的节点数。



**图 2. 神经网络中的 Softmax 层**



Softmax 选项

请查看以下 Softmax 变体：

**完整 Softmax** 是我们一直以来讨论的 Softmax；也就是说，Softmax 针对每个可能的类别计算概率。

**候选采样**指 Softmax 针对所有正类别标签计算概率，但仅针对负类别标签的随机样本计算概率。例如，如果我们想要确定某个输入图片是小猎犬还是寻血猎犬图片，则不必针对每个非狗狗样本提供概率。

类别数量较少时，完整 Softmax 代价很小，但随着类别数量的增加，它的代价会变得极其高昂。候选采样可以提高处理具有大量类别的问题的效率。

一个标签与多个标签

Softmax 假设每个样本只是一个类别的成员。但是，一些样本可以同时是多个类别的成员。对于此类示例：

您不能使用 Softmax。

您必须依赖多个逻辑回归。

例如，假设您的样本是只包含一项内容（一块水果）的图片。Softmax 可以确定该内容是梨、橙子、苹果等的概率。如果您的样本是包含各种各样内容（几碗不同种类的水果）的图片，您必须改用多个逻辑回归。

* 1. **嵌入**

1. **协同过滤的目的**

协同过滤是一项可以预测用户兴趣（根据很多其他用户的兴趣）的任务。以影片推荐的任务为例，假设我们有 100 万个用户，以及每位用户观看过的影片的列表（可供观看的影片共有 50 万部）。 我们的目标是向用户推荐影片。

要解决这个问题，我们需要使用某种方法来确定哪些影片是相似的。我们可以通过将影片嵌套到低维空间（使得相似的影片彼此邻近）来实现这个目标。

在介绍如何学习嵌套之前，我们先来了解一下我们希望嵌套具备的特质类型，以及我们将如何表示训练数据以供学习嵌套。

在一维数轴上排列影片

为了更直观地了解嵌套过程，请准备一张纸，试着在一维数轴上排列以下影片，让越相关的影片靠得越近：

| 影片 | [分级](https://wikipedia.org/wiki/Motion_Picture_Association_of_America_film_rating_system#MPAA_film_ratings) | 说明 |
| --- | --- | --- |
| [《蓝》](http://www.imdb.com/title/tt0108394/) | R | 一位法国妇人在丈夫与爱女丧命于一场车祸后悲痛欲绝。 |
| [《蝙蝠侠：黑暗骑士崛起》](http://www.imdb.com/title/tt1345836) | PG-13 | 这部影片是[《黑暗骑士》](http://www.imdb.com/title/tt0468569/)的续集，以 DC 漫画的宇宙空间为背景，讲述蝙蝠侠尽力保护高谭市免遭核毁灭的故事。 |
| [《哈利·波特与魔法石》](http://www.imdb.com/title/tt0241527/) | PG | 一个失去双亲的男孩发现自己会巫术，于是前去霍格沃茨魔法学校学习魔法，在这里他与邪恶的伏地魔展开了第一场激斗。 |
| [《超人总动员》](http://www.imdb.com/title/tt0317705/) | PG | 被迫在郊区过着平民生活的超人一家重出江湖，拯救超人家族免遭辛拉登及其杀手机器人的迫害。 |
| [《怪物史莱克》](http://www.imdb.com/title/tt0126029/) | PG | 可爱的怪物史莱克和他的伙伴驴子，启程营救被火龙囚禁在城堡的菲奥娜公主。 |
| [《星球大战》](http://www.imdb.com/title/tt0076759) | PG | 卢克·天行者和汉·索洛与两个义军机器人结成一队，共同拯救莱娅公主并保卫星球。 |
| [《疯狂约会美丽都》](http://www.imdb.com/title/tt0286244) | PG-13 | 专业骑行者查宾在环法自行车大赛期间被挟持，他的奶奶带着他家的胖狗漂洋过海，并在爵士歌手三姐妹的帮助下救出了他。 |
| [《记忆碎片》](http://www.imdb.com/title/tt0209144/) | R | 一位短期记忆丧失症患者将线索纹在身上，竭尽全力寻找杀害自己妻子的凶手。 |

一个可行（但极不完善）的解决方案。



图 1. 一种可行的一维排列

虽然这种嵌套有助于捕获影片的适宜观赏年龄段（儿童或成人），但在推荐影片时还需要考虑影片的许多其他方面。我们进一步分析此示例，再添加一个嵌套维度。

在二维空间中排列影片

再次进行之前的练习，但这次要在二维空间中排列影片。

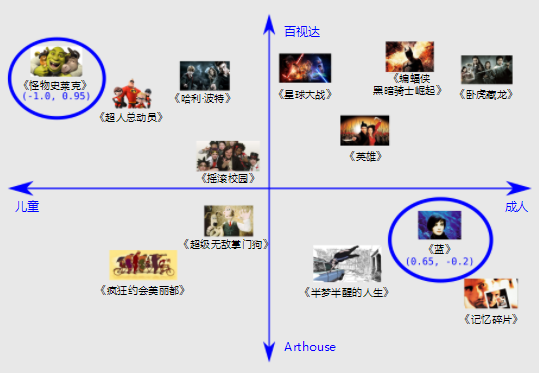
另一可行的解决方案。

图 2. 一种可行的二维排列

利用这种二维嵌套，我们可以定义影片之间的距离，从而使在适宜儿童或成人的程度上相近的影片以及属于大片或艺术影片的程度上相近的影片位于相近的位置。当然，这只是影片诸多重要特征中的两个。

更笼统地来说，我们所做的是将这些影片映射到一个嵌套空间，其中的每个字词都由一组二维坐标来表示。例如，在这个空间中，《怪物史莱克》映射到了 (-1.0, 0.95)，而《蓝》则映射到了 (0.65, -0.2)。通常情况下，在学习 d 维嵌套时，每部影片都由 d 个实值数字表示，其中每个数字都分别表示在一个维度中的坐标。

在此示例中，我们为每个维度指定了名称。在学习嵌套时，每个维度的学习跟它们的名字无关。有时我们可以查看嵌套并为维度赋予语义，但有时则无法做到这一点。通常，每个此类维度都称为一个潜在维度，因为它代表的特征没有明确显示在数据中，而是要根据数据推断得出。

最终，真正有意义的是嵌套空间中各个影片之间的距离，而不是单个影片在任意指定维度上的坐标。

1. **分类输入数据**

**分类数据**是指用于表示一组有限选项中的一个或多个离散项的输入特征。例如，它可以是某用户观看过的一组影片，某文档中使用的一系列单词，或某人从事的职业。

分类数据的最高效表示方式是使用稀疏张量（一种含有极少非零元素的张量）。例如，如果要构建一个影片推荐模型，可以为每部可能的影片分别分配一个唯一的 ID，然后通过用户已观看影片的稀疏张量来表示每位用户，如图 3 所示。

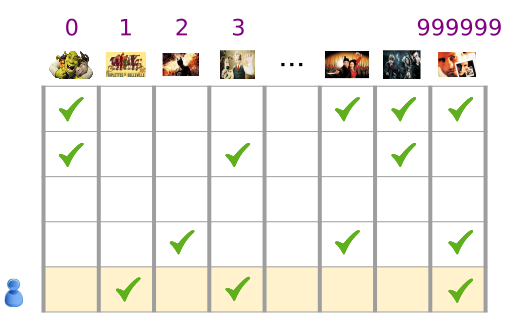


图 3. 影片推荐问题的数据

在图 3 的矩阵中，每一行都是一个显示用户的影片观看记录的样本，并以稀疏张量的形式表示，因为每个用户只会观看所有可能的影片中的一小部分。根据影片图标上方所示的索引，最后一行对应于稀疏张量 [1, 3, 999999]。

类似地，我们还可将字词、句子和文档表示为稀疏矢量 - 在这种情况下，词汇表内每个字词所扮演的角色类似于推荐示例中的影片。

为了能够在机器学习系统中使用这类[表示法](https://developers.google.cn/machine-learning/crash-course/representation/video-lecture)，我们需要将每个稀疏矢量表示为数字矢量，从而使语义上相似的项（影片或字词）在矢量空间中具有相似的距离。但如何将字词表示为数字矢量呢？

最简单的方法是：定义一个巨型输入层，并在其中为词汇表内的每个字词设定一个节点，或者至少为您数据中出现的每个字词设定一个节点。如果您的数据中出现了 50 万个独一无二的单词，您可以使用长度为 50 万的矢量来表示每个单词，并将每个字词分配到相应矢量中对应的索引位置。

如果为“马”分配的索引是 1247，那么为了将“马”馈入到您的网络中，可以将第 1247 个输入节点设成 1，其余节点设成 0。这种表示法称为**独热编码** (one-hot encoding)，因为只有一个索引具有非零值。

更常见的是，使用一个包含各个单词在大块文本中出现次数的向量。这被称为“词袋”(bag of words) 表示法。在一个词袋矢量中，50 万个节点中的若干个节点将会具有非零值。

不过，无论您如何确定非零值，若将节点与字词一一对应，您得到的输入矢量就会比较稀疏 - 即：矢量很大，但非零值相对较少。稀疏表示法存在多项问题（如下所述），这些问题可能会致使模型很难高效地学习。

网络的规模

巨型输入矢量意味着神经网络的对应权重数目会极其庞大。如果您的词汇表内有 M 个字词，而神经网络输入层上方的第一层内有 N 个节点，您便需要为该层训练 MxN 个权重。权重数目过大会进一步引发以下问题：

**数据量**：模型中的权重越多，高效训练所需的数据就越多。

**计算量**：权重越多，训练和使用模型所需的计算就越多。这很容易就会超出您硬件的能力范围。

矢量之间缺乏有意义的联系

如果您已将 RGB 通道的像素值馈入到图片分类器中，分析“邻近”值便行得通。不管是从语义上来看，还是从矢量之间的几何距离来看，红蓝色与纯蓝色都是邻近的。不过，对于在索引 1247 处设为 1 以表示“马”的矢量而言，如果说它与在索引 238 处设为 1 以表示“电视机”的矢量不够邻近，那么它与在索引 50430 处设为 1 以表示“羚羊”的矢量亦然。

解决方案：嵌套

上述问题的解决方案就是使用**嵌套**，也就是将大型稀疏矢量映射到一个保留语义关系的低维空间。在此模块的随后几个部分中，我们将从直观角度、概念角度和编程角度来详细探讨嵌套。

1. **转换到低纬度空间**

要解决稀疏输入数据的核心问题，您可以将高维度数据映射到低维度空间。

通过纸上练习您已了解，即便是小型多维空间，也能自由地将语义上相似的项归到一起，并将相异项分开。矢量空间中的位置（距离和方向）可对良好的嵌套中的语义进行编码。例如，下面的真实嵌套可视化图所展示的几何关系图捕获了国家与其首都之间的语义关系。

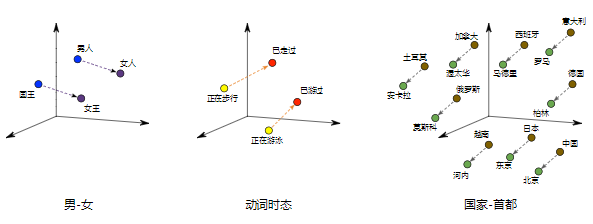


图 4. 嵌套可产生精彩的模拟

借助这种有意义的空间，机器学习系统能够检测出对学习任务可能有帮助的模式。

收缩网络

尽管我们需要足够的维度来编码丰富的语义关系，但我们也需要足够小的嵌套空间来更快速地训练我们的系统。实用嵌套的量级大致有数百个维度。这可能比您在自然语言任务中使用的词汇规模要小好几个数量级。

嵌套充当查询表

嵌套是一个矩阵，每列表示您词汇中的一项所对应的矢量。要获得某个词汇项的密集矢量，您可以检索该项所对应的列。

但是，如何转换字词矢量的稀疏包呢？要获得表示多个词汇项（例如，一句或一段中的所有字词）的稀疏矢量的密集矢量，您可以检索各项的嵌套，然后将它们相加。

如果稀疏矢量包含词汇项的计数，则您可以将每项嵌套与其对应项的计数相乘，然后再求和。

这些运算可能看起来很眼熟吧。

嵌套查询充当矩阵乘法

我们刚刚阐述的查询、乘法和加法程序等效于矩阵乘法。假设有一个 1 X N 的稀疏表示 S 和一个 N X M 的嵌套表 E，矩阵乘法 S X E 可以得出密集矢量 1 X M。

但首要问题是，如何获取 E 呢？我们将在下一部分介绍如何获取嵌套。

1. **获取嵌入**

您可以通过多种方式来获取嵌套，包括 Google 研发的世界一流算法。

标准降维技术

目前有很多在低维空间捕获高维空间重要结构的数学技术。理论上，这些技术都可以用来创建用于机器学习系统的嵌套。

例如，[主成分分析](https://wikipedia.org/wiki/Principal_component_analysis) (PCA) 已用于创建字词嵌套。在给定一组实例的情况下，例如字词矢量包，PCA 会尝试查找高度相关且可以合并的维度。

Word2vec

Word2vec 是 Google 为了训练字词嵌套而研发的一种算法。Word2vec 基于**分布假设**，将语义上相似的字词映射到在几何图形上邻近的嵌套矢量。

分布假设指出经常具有相同相邻字词的字词往往在语义上相似。“狗”和“猫”这两个字词经常靠近“兽医”一词出现，这就可以说明这两个字词在语义上相似。正如语言学家约翰·弗斯 (John Firth) 在 1957 年所言：“观其伴而知其意”。

Word2Vec 通过训练神经网络来区分实际共同出现的多组字词与随机出现在一起的字词，从而充分利用此类上下文信息。输入层采用一种稀疏表示法用于组合一个目标字词与一个或多个上下文字词。这一输入层会连接到一个较小的隐藏层。

在其中一版算法中，系统通过用随机噪点字词替代目标字词来举出反面示例。在给出正面示例“the plane flies”的情况下，系统可能会换成“jogging”来创建对比鲜明的反面示例“the jogging flies”。

另一版算法通过将真实的目标字词与随机选择的上下文字词配对来创建反面示例。因此，系统可能会举出正面示例（(the, plane)、(flies, plane)）和反面示例（(compiled, plane)、(who, plane)），然后通过学习分辨哪几对真正地在文字中一起出现。

不过，分类器不是上述任何一版算法的真正用途。在训练模型后，你得到的是一组嵌套。借助将输入层连接到隐藏层的权重，您可以将字词的稀疏表示映射到小型矢量。这类嵌套可在其他分类器中重复利用。

要详细了解 word2vec，请参阅 [tensorflow.org 上的教程](https://tensorflow.google.cn/tutorials/word2vec/index.html)

将嵌套训练为大型模型的一部分

您也可以将嵌套作为目标任务的神经网络的一部分进行学习。通过这个方法，您可以为自己的特定系统量身定制嵌套，不过耗费的时间可能要比单独训练嵌套的时间长。

一般来说，当您具有稀疏数据（或您想要嵌套的密集数据）时，您可以创建一个嵌套单元，这个嵌套单元其实是大小为 d 的一个特殊类型的隐藏单元。此嵌套层可与任何其他特征和隐藏层组合。和任何 DNN 中一样，最终层将是要进行优化的损失函数。例如，假设我们正在执行协同过滤，目标是根据其他用户的兴趣预测某位用户的兴趣。我们可以将这个问题作为监督式学习问题进行建模，具体做法是随机选取（或留出）用户观看过的一小部分影片作为正类别标签，然后再优化 Softmax 损失。

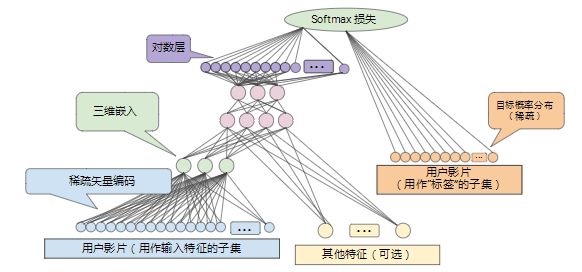


图 5. 根据协同过滤数据学习影片嵌套的 DNN 架构示例

再举一个例子，如果您想在 DNN 中针对房地产广告词创建嵌套层来预测房价，则您可以将训练数据中的已知房屋售价用作标签来优化 L*2* 损失。

在学习 d 维嵌套时，每一项都会映射到 d 维空间中的一个点，这样相似项就会在该空间内彼此邻近。图 6 说明了在嵌套层中学到的权重与几何视图之间的关系。输入节点与 d 维嵌套层中的节点之间的边的权重对应于 d 维坐标轴中每一维的坐标值。

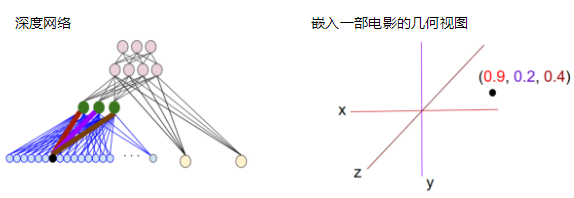


图 6. 嵌套层权重的几何视图

1. **机器学习工程**
   1. **生产环境机器学习系统**
   2. **静态训练与动态训练**

静态模型-离线训练

易于构建和测试：使用批量训练和测试，对其进行迭代，直到达到良好效果。

仍然需要对摄入进行监控。

模型容易过时。

动态模型-在线模型

随着时间推移不断为训练数据注入新数据，定期同步更新版本。

使用渐进式验证，而不是批量训练和测试。

需要监控，模型回滚和数据隔离功能。

* 1. **静态推理与动态推理**

您可以选择以下任一推理策略：

**离线推理**，指的是使用 MapReduce 或类似方法批量进行所有可能的预测。然后，将预测记录到 SSTable 或 Bigtable 中，并将它们提供给一个缓存/查询表。

*优点：可以使用批量方法；可以在推送之前对数据预测执行后期验证。*

*缺点：只能对我们知晓的数据进行预测，不适用于存在长尾的情况；更新可能延迟数小时或数天。*

**在线推理**，指的是使用服务器根据需要进行预测。

优点：可在新项目加入时对其进行预测，非常适合存在长尾的情况。

缺点：计算量非常大，对延迟较为敏感，可能会限制模型的复杂度；监控需求更多。

* 1. **数据依赖关系**

特征管理：

输入数据（特征）决定机器学习系统的行为。

选择输入信号时要谨慎。

对输入数据提出的问题：

可靠性，信号不可用时会出现什么情况？您知道吗？

版本控制，计算此信号的系统是否发生过变化？多久一次？会出现什么情况？

必要性，信号的实用性是否能证明值得添加此信号？

相关性，是否有任何输入信号密不可分，以至于需要采取额外策略来梳理它们？

反馈环，哪个输入信号可能会受到我的模型输出的影响？

1. **机器学习现实世界应用实例**
   1. **癌症预测**

模型经过训练后可以根据病历来预测“病人患有癌症的概率”

特征包括病人年龄、性别、之前的病史、医院名称、生命体征、检验结果

模型在处理预留检验数据方面表现出色

但模型在针对新病人进行预测时表现却很糟糕，这是为什么呢？有医院名称这个特征。

* 1. **18世纪文字**
  2. **现实世界应用准则**

确保第一个模型简单易用

着重确保数据管道的正确性

使用简单且可观察的指标进行训练和评估

拥有并监控您的输入特征

将您的模型配置视为代码：进行审核并记录在案

记下所有实验的结果，尤其是“失败”的结果

**总结**  
要继续机器学习培训，进一步巩固您的 TensorFlow 技能，请查看以下资源：

机器学习实践课程

查看下列有关 Google 如何在其产品中运用机器学习技术的真实案例研究，以及相关视频和实际动手编码练习：

* [**图片分类**](https://developers.google.cn/machine-learning/practica/image-classification/)：了解 Google 如何开发用于在 Google 照片中为搜索提供支持的图片分类模型，然后构建您自己的图片分类器。

其他机器学习资源

* [深度学习](https://www.udacity.com/course/deep-learning--ud730)：关于神经网络的机器学习高级课程，对图片和文字模型进行了广泛的介绍
* [机器学习规则](https://developers.google.cn/machine-learning/rules-of-ml)：关于机器学习工程的最佳做法
* [TensorFlow.js](https://js.tensorflow.org/)：采用 WebGL 加速技术且基于浏览器的 JavaScript 库，用于训练和部署机器学习模型

TensorFlow

* [安装 TensorFlow](https://tensorflow.google.cn/install/)：关于在 Mac OS X、Ubuntu 和 Windows 计算机上设置 TensorFlow 的说明
* [tf.contrib.learn 快速入门](https://tensorflow.google.cn/get_started/tflearn)：关于使用高级 TensorFlow API 构建神经网络分类器的指南
* [TensorFlow 编程人员指南](https://tensorflow.google.cn/programmers_guide/)：有关 TensorFlow 主要功能（包括变量、线程和调试）的详细指南
* [2017 年 TensorFlow 开发者峰会](https://www.youtube.com/playlist?list=PLOU2XLYxmsIKGc_NBoIhTn2Qhraji53cv)：一系列技术讲座和演示，重点介绍 TensorFlow API 和实际应用

参加 Kaggle 大赛

已准备好开始运用新掌握的机器学习技能来应对数据科学领域面临的现实挑战了吗？ 您可以在 [Kaggle](https://www.kaggle.com/) 上的众多比赛中一试身手！