<https://developers.google.com/machine-learning/crash-course/>

机器学习简介

**框架处理**

监督式学习，通过学习如何组合输入信息来对从未见过的数据进行分类。

机器学习术语：

标签

特征：输入变量

样本：数据的特定实例

有标签样本：labeled examples: {features, label}: (x, y)

无标签样本：unlabeled examples: {features, ?}: (x, ?)

模型：定义了特征与标签之间的关系

回归与分类

回归模型可预测连续值

分类模型可预测离散值

Note：合适的特征应该是具体可量化的

**深入了解机器学习**

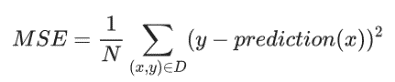
线性回归：

训练和损失：（损失是基于反向传播算法的一个很重要的因子，损失函数的设计直接影响模型的效果和收敛程度，所以在Hiton的capsule理论中就开始尝试抛弃反向传播算法。）

训练模型表示通过有标签样本来学习所有权重和偏差的理想值。

经验风险最小化：检查多个样本并尝试找出可最大限度的减少损失的模型。

训练模型的目标是从所有的样本中找到一组平均损失最小的权重和偏差。

线性回归模型使用的是一种平方损失，又称L2损失的损失函数：

当然可以手动设计损失函数

**降低损失**

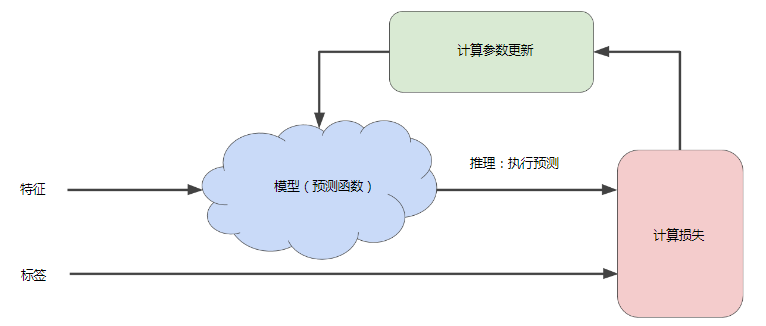
**视频：**

随机梯度下降法

1.首先确保误差沿着梯度（误差函数的导数）的方向前进，这样误差下降的速度才最快，其次没有必要计算整体的样本的梯度，理论上，只有计算所有的所有样本数据的梯度值，才能确保我们计算的正确的梯度方向，但计算速度太慢，根据经验随机选取一部分样本计算梯度即可，对于梯度前进步伐的lr。Lr过大可能会越过最优点。

2.出发点重要吗？

凸优化问题，如果在碗底大概位置出发，最终会到达碗底，但是神经网络并不是凸形的，初始值很重要。

**迭代方法：**机器学习模型如何以迭代方式降低损失。

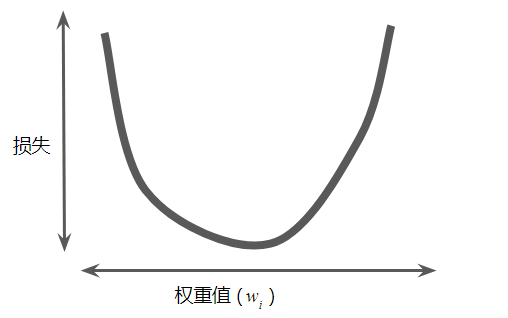
迭代策略在机器学习中的应用非常普遍，这主要是因为他们可以很好的扩展到大型数据集。

计算损失部分：模型将要使用的损失函数

计算参数更新部分：机器学习就是在此部分检查损失函数的值，并生成新的权重和参数，反向传播和链式求导。假设这个神秘的绿色框会产生新值，然后机器学习系统将会根据所有的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，直到算法发现可能最低的模型参数，一般到总体损失不在变化或者变化及其缓慢为止，说明模型已经收敛。

**梯度下降法**

计算参数更新：

假设我们有时间和计算资源来计算w的所有可能的损害，就回归问题而言，所产生的损失和w图始终是凸形。

凸形问题只有一个最低点，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的w每个可能值的损失函数来找到收敛点这种方法效率太低，所以采用梯度下降法。

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

**学习速率**

梯度具有方向和大小，梯度下降算法乘以一个称为学习速率的变量，以确定下一个点的位置。

Lr的选择是纯粹的超参问题，实际上也就是try，不过你如果直到损失函数的梯度较小，你可以放心的试着采用更大的学习速率，已补偿较小的梯度并获得更大的步长。当然可以尝试网格搜索法。

降低学习率可以减少损失，但损失仍然会收敛于不可接受的高值。

**优化学习速率**

<https://developers.google.com/machine-learning/crash-course/fitter/graph>

**随机梯度下降法**

批量：单次迭代中计算梯度的样本总数。决定了单次迭代所用的时间，如果该批量定义为整个数据集的话，那么单次运行的时间也是相当长的。

大型数据集可能出现冗余数据的可能很高，冗余数据集是有利于消除杂乱的梯度，但是超大批量所具备的预测价值也不必大型批量高。

随机梯度下降（SGD）每次只取一个样本进行迭代，迭代次数足够高的话，也能发挥作用

小批量随机梯度下降介于全批量和SGD之间。

Note：在小批量或甚至包含一个样本的批量上执行梯度下降法通常比全批量更高效，毕竟，计算一个样本的梯度要比计算数百万个样本的梯度成本低得多。

**使用TF的基本步骤**

视频：

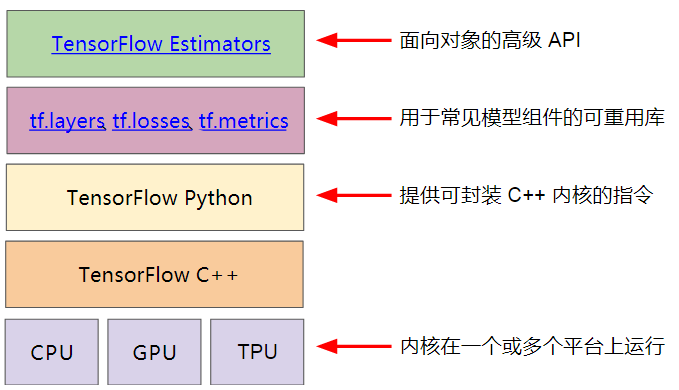
tf.estimator 高级OOP API

tf.layers/tf.losses/tf.metrics 用于常见模型组件的库

tensorflow 低级API

Pandas：是一种列存数据分析API、

DataFrame：关系从数据表格，其中包含多个行和已命名的列。

Series：单一列，DataFrame中有多个Series。

使用tensorflow的基础步骤：

**泛化：**

**三项基本假设阐明泛化理论：**

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。

分布是平稳的，不会随时间发生变化。

我们从同一分部的数据划分中抽取样本。

过拟合是由于模型的复杂程度超过了所需程度造成的。机器学习的基本冲突是适当拟合我们的数据，但也要尽可能简单的拟合数据。

机器学习的目标是从真实概率分布中抽取的新数据作出良好的预测。

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。

**验证集和测试集**

测试集需要满足下面两个条件：

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。

创建一个能够很好泛化到新数据的模型。测试集充当新数据的代理。

当学习速率设为3时，测试损失明显高于训练损失。

通过降低学习率（例如，降至0.001），测试损失会下降到非常接近训练损失的值。在大多数运行中，增加批量大小不会显著影响训练损失或测试损失。然而，在一小部分运行中，将批量大小增加到20或者更高会导致测试损失略低于训练损失。

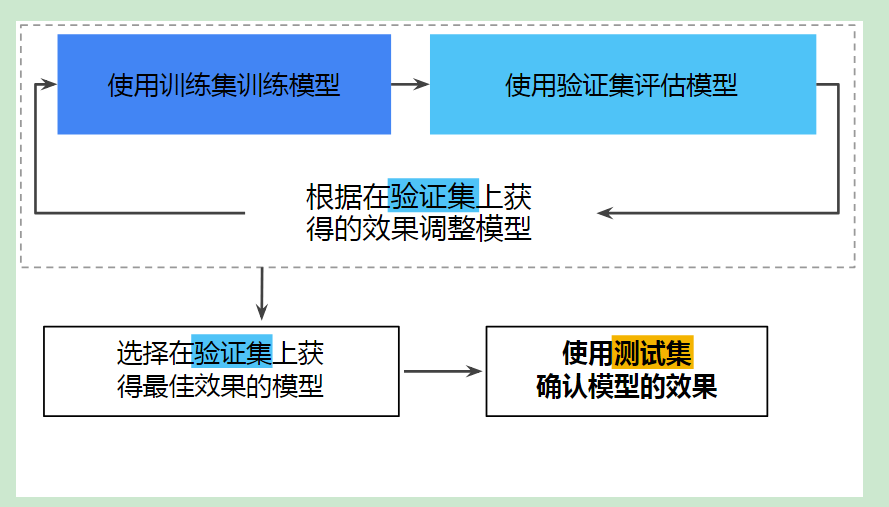
将训练数据和测试数据之比从50%降至10%大幅度降低了训练集中数据点的个数。由于数据太少，较高的批次大小和学习速度会导致训练模型沿着曲线无规律的跳动（在最低点上方反复跳跃）

**验证**

使用测试集和训练集来推动模型开发迭代的流程。在每次迭代时，我们都会对训练诗句进行训练并评估测试数据，兵以基于测试数据的评估结果为指导来选择和更改各种模型超参数，例如学习速率和特征，这种方法是否存在问题？

存在，我们基于给定测试集执行评估的次数越多，不知不觉的过拟合该测试集的风险就越高。

调整模型指的是调整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。

将数据集划分为三个自己可以大幅度降低过拟合的发生几率。

该工作流程之所以更好，是因为它暴露给测试集的信息更少。

**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。**

特征这块深度学习就是由卷积来完成了，但传统的机器学习还是要做特征工程

**表示法 (Representation)**

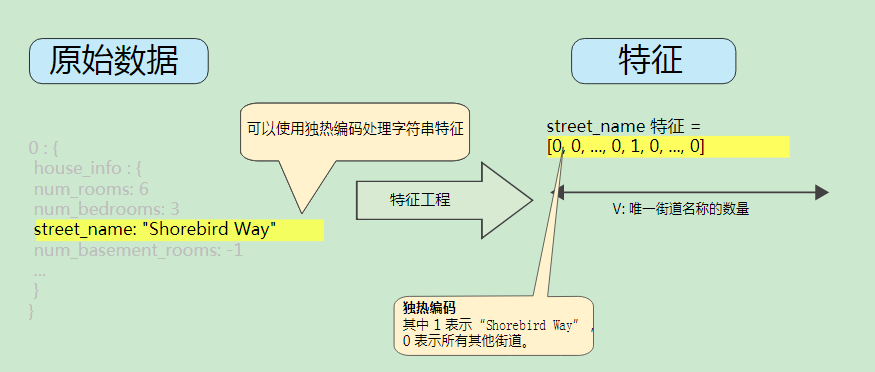
创建数据表示，为模型提供有用的信号来了解数据的关键特性，也就是说，为了训练模型，必须选择最能代表数据的特征集。

**特征工程**

将原始数据映射成特征

映射数值：机器学习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。

映射字符串：字符串要转化成数字形式

1. 首先为表示所有特征的字符串定义以一个词汇表。
2. 然后使用词汇表创建一个独-热编码。

映射分类（枚举）值

分类特征可能被编码为枚举类型或表示不同值的整数离散集。可以将每个分类特征表示为单独的布尔值，有一个最好的好处可能简化某个值属于多分类问题。

**良好特征的特点**

第一：将原始数据映射到合适特征矢量的方法，第二，什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的作出预测。

最好具有清晰明确的含义

不要将“神奇”的值与实际数据混为一谈

考虑上游不稳定性

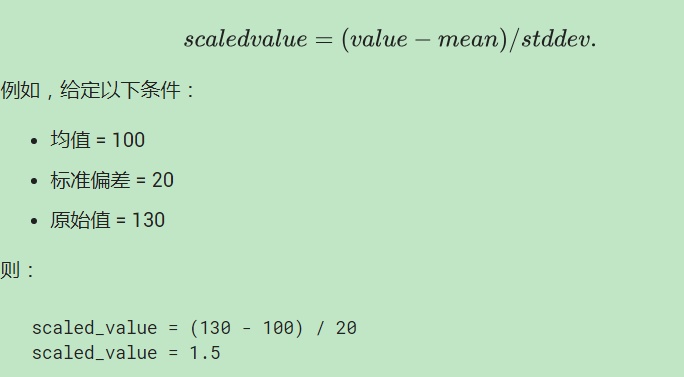
特征的定义不应随时间发生变化

**清洗数据**

**缩放特征值** 如果有多个特征（深度学习中同理），有如下优势：

帮助梯度下降法更快的收敛

帮助避免“NAN陷阱”。在这种陷阱中，模型中的一个数值变成NAN（例如某个值在训练期间超过浮点精度精确率限制时），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，（我在计算OA时曾经出现过0的情况，是不是这个原因？）

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，则规模会对范围较大的特征投入较大精力。

**处理极端离群值**

1. 取对数
2. 将极大值限制为某个任意值

**分箱**

**清查**

数据集并不是都是可靠的

遗漏值、重复样本、不良标签、不良特征值

统计 最大值和最小值，均值和中间值，标准偏差也是有意义的

**特征组合**

**对非线性规律进行编码**

解决非线性问题，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待空间中的非线性规律进行编码的合成特征。

我们通过将 x1 与 x2 组合来创建一个名为 x3 的特征组合：x3=x1\*x2

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：

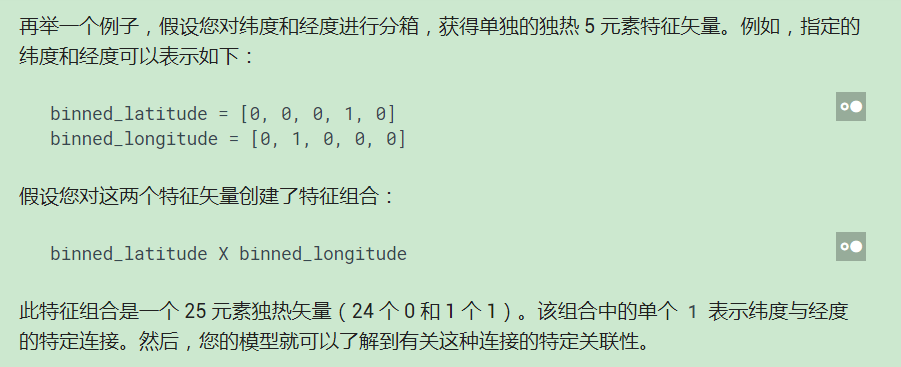
y=b+w1x1+w2x2+w3x3

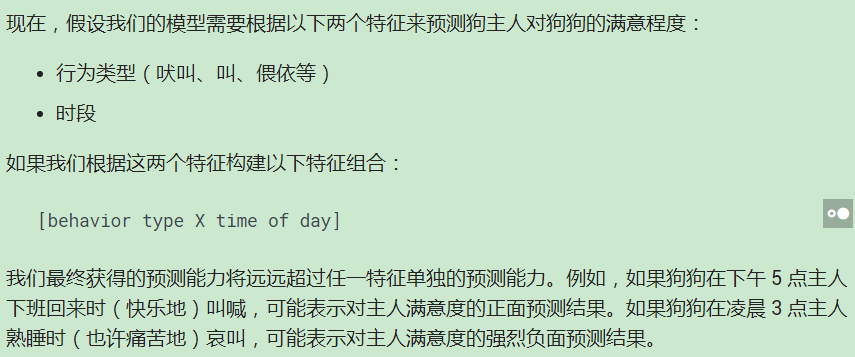
线性算法可以算出 w3 的权重，就像算出 w1 和 w2 的权重一样。换言之，虽然 w3 表示非线性信息，但您不需要改变线性模型的训练方式来确定 w3 的值。

通过随机梯度下降法，可以有效的训练线性模型。因此，在使用扩展的线性模型时辅以特征组合一周都是训练大规模数据集的有效方法。

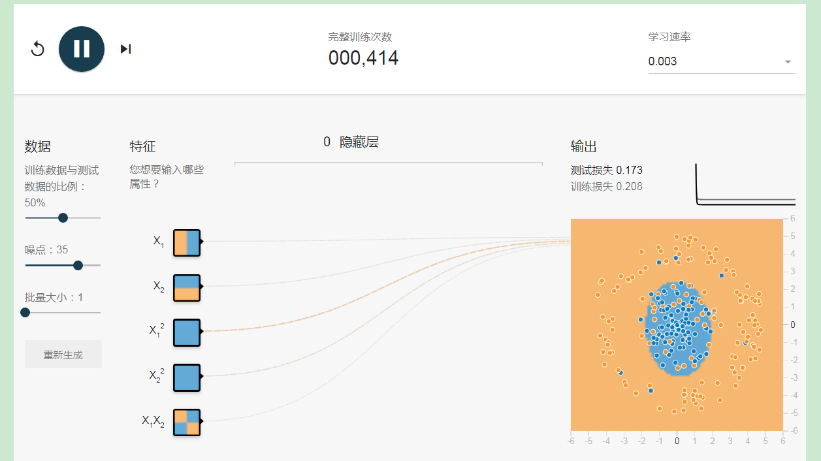
**组合独热矢量**

很有意义，因为单独的特征之间存在某种关联，组合特征有的时候比单独一个特征要更加有效。

机器学习模型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。

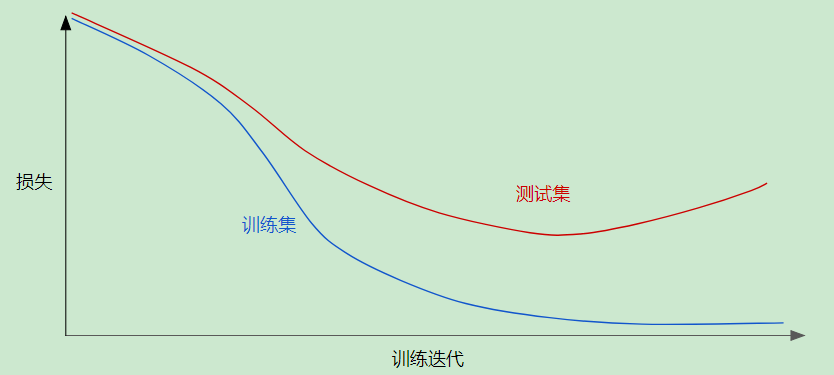
线性学习器可以很好地扩展到大量数据。对于大规模数据集使用特征组合是学习高度复杂模型的一种有效的策略。

Playgroud

<https://developers.google.com/machine-learning/crash-course/feature-crosses/playground-exercises>

输出结果看起来不像线性模型

**正则化**

过度拟合，

这是一个泛化图，明显的模型过拟合，测试集的loss相比训练集的loss先降后升，acc肯定也是比训练集的低了。

怎么办？ 正则化（通过降低复杂模型的复杂度来防止过拟合，称之为正则化）

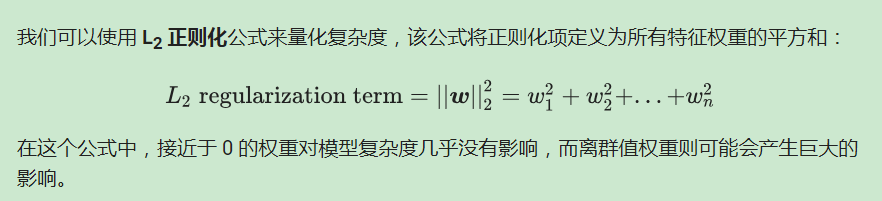
并非只是以最小化损失为目标（经验风险最小化）

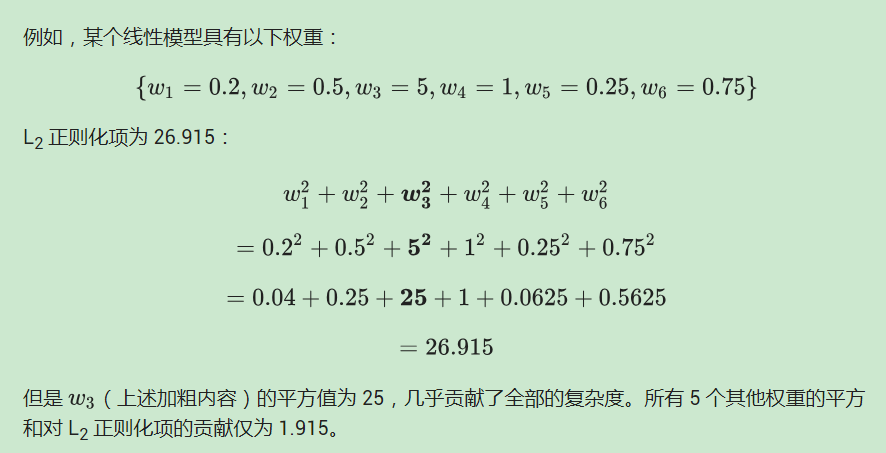
而是以最小化损失和复杂为目标（结构风险最小化）

**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。**

两种衡量模型复杂度的常见方式：

1. 将模型复杂度作为模型中所有特征的权重的函数
2. 将模型复杂度作为具有非零权重的特征总数的函数。

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。



**简化正则化**

用正则化项的值乘以名为lambda的标量

L2正则化对模型具有以下影响：

1. 使权重值接近于0
2. 使权重平均值接近于0，且呈正态分布。

增加lambda值将增强正则化效果。

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lambda值过高，则模型会非常简单，面临数据欠拟合的风险。模型将无法从训练数据中获得足够的信息来作出有用的预测。

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

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.com/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。

由于在模型中添加了正则化，其实每个特征贡献的权重也会发生变化，整个模型训练的权重参数会随之产生变化。

正则化率从0增加到0.3

测试损失明显减少。（虽然测试损失明显减少，训练损失实际上是有所增加。但是这属于正常现象，因为向损失函数中添加了另一项来降低复杂度，所以训练的loss是有所增加的。但是最重要的是测试损失，因为他才是用于衡量模型是否针对新数据做出良好预测的标准）

测试损失和训练损失之间的差值明显减少

特征和某些特征组合的权重的绝对值较低，这表明模型复杂度有所降低。整个模型的权重是发生变化的。

假设某个线性模型具有两个密切相关的特征；也就是说，这两个特征几乎是彼此的副本，但其中一个特征包含少量的随机噪点。如果我们使用 L2 正则化训练该模型，这两个特征的权重将出现什么情况？

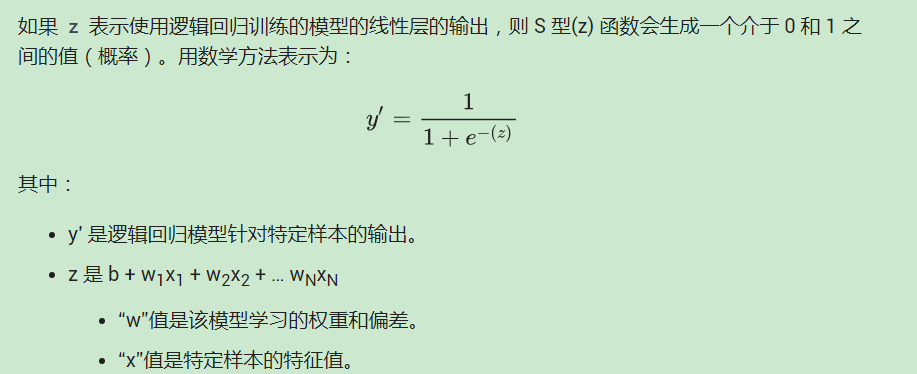
这两个特征将拥有几乎相同的适中权重。L2正则化会使特征的权重几乎相同，大约为模型中只有两个特征之一时权重的一半。

L2正则化降低较大权重的程度高于降低较小权重的程度。因此，即使某个权重降低的速度比另一个块，L2正则化也往往会使较大权重降低的速度快于较小的权重。

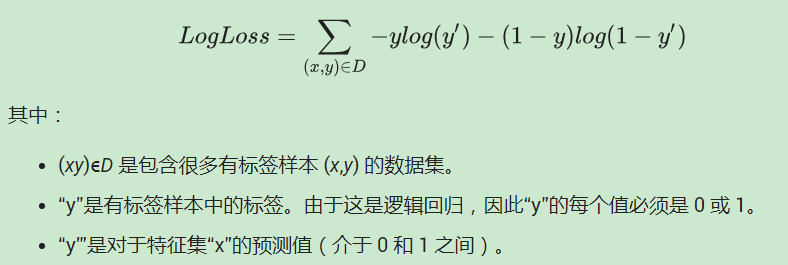
**逻辑回归**

许多问题需要将概率估算值作为输出。逻辑回归是一种及其高效的概率计算机制。

S型函数（sigmoid function）

使用S函数来确保输出值始终在0-1之间。

**模型训练**

逻辑回归的损失函数是对数损失函数

**逻辑回归中的正则化**

正则化在逻辑回归建模中及其重要。如果没有正则化，逻辑回归的渐进性会不断促使损失在高维空间内达到0.

L2正则化；早停法（限制训练步数或学习速率）

**分类：指定阈值**

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

调整逻辑回归的阈值不用于调整学习速率等超参数。在选择阈值时，需要评估你将因犯错而承担多大的后果。

**分类：真与假以及正类别与负类别**

真正例(TP) 是指模型将正类别样本正确地预测为正类别。

真负例(TN) 是指模型将负类别样本正确地预测为负类别

假正例(FP) 是指模型将负类别样本错误地预测为正类别

假负例(FN) 是指模型将正类别样本错误地预测为负类别

**准确率**

分类不平衡问题的指标：

精确率： 在所有的正例中，我预测为正确的样本数

召回率： 在所有正样本中，被正确识别的类别的比例

其实三个指标都高才是最有效的。

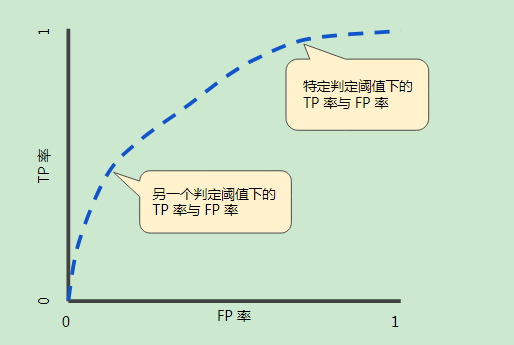
提高分类阈值：假正例数量会减少，但假负例数量会相应的增加。结果，精确率会有所增加，而召回率会有所降低。

降低分类阈值：假正例数量会增加，而加福利数量会减少，精度率有所降低，召回率有所上升。

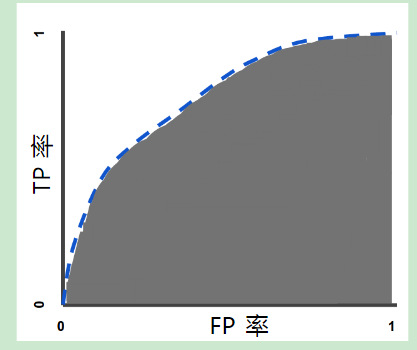
**ROC和曲线下面积**(我在高光谱影像分类中用的很少)

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



ROC曲线下面积



对所有可能的分类阈值的效果进行综合衡量。解读，模型将某个随机正类别样本排列在某个随机负样本之上的概率。

曲线面积因以下两个原因比较实用：

曲线下面积的尺度不变。

曲线下面积的分类阈值不变。

**预测偏差**

预测偏差 = 预测平均值 – 数据集中相应标签的平均值

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

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

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

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，不妨减小lambda的值。

**稀疏正则化：L1正则化**

稀疏矢量通常包含很多维度。创建特征组合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。

高维稀疏矢量中一些组合特征信息是没有意义的，能使其权重刚好降至0.就相当于在模型中移除了相应的特征，将特征设为0可节省RAM空间，且可以减少模型中的噪点。

但是L2正则解决不了这个问题，L2正则可以使权重变小，但是不能使他们正好变为0.

两一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。遗憾的是这种基于计数的方法会将我们的凸优化问题变为非凸优化问题，即NP困难。L0正则化并不是一种有效的方法。

L1正则化这种正则化像的作用类似Lo，但它具有凸优化的优势，可以有效的进行计算。因此，我们可以使用L1正则化使模型中很多信息缺乏的稀疏正好为0，从而在推理时节省RAM。

L1和L2正则化

L2降低权重的平方，L1降低权重的绝对值。

L2的导数为2\*权重，L1的倒数为k，是一个常数，其值与权重无关。

你可以理解L2的导数的作用是每次移除权重的x%。L1每次从权重中减去一个常数

**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**

有些混乱的小型训练数据集

1. 从L2正则化转换到L1正则化之后，测试损失与训练损失之间的插值明显减少。
2. 从L2正则化转换到L1正则化之后，所有的权重都有所减少。
3. 增加L1正则化率一般会减少已知权重，不过如果正则化率过高，该模型便不能收敛，损失也会很高。

**神经网络简介**<https://developers.google.com/machine-learning/crash-course/introduction-to-neural-networks/playground-exercises>

非线性问题的解决，采用特征组合。

Relu带来了非线性转化能力，包括sigmoid，relu等

混乱复杂的数据集即使特征组合也很难处理的。

神经元越多越好，就效果而言

减少神经元个数，减少层数，减少复杂度

神经网络可能很简单，但是通过更改激活函数，正则化等措施依然可以达到很好的效果。

**训练神经网络**

反向传播失败的案例：

梯度消失

当较低层（更接近输入）的梯度可能会变得非常小，使用Relu有助于防止梯度消失

梯度爆炸 网络的权重过大，则较低层的梯度会设计许多大项的乘积。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的输入可能无法做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

Dropout

**多类别神经网络（Multi-Class Neural Networks）: 一对多**

**Softmax（面试热点）**

**嵌套 (Embedding)**

**协同过滤的目的**

很像one-hot representation到distributed representation的转变，将信息放到一个高维的向量空间中，那么相似的特征就会类聚在一起。特征之间的距离是很有用的指标。

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。

独热编码：

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

若将节点和字词一一对应，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。

巨型输入矢量意味着神经网络的对应权重数目会及其庞大。如果词汇表内有M个字词，神经网络输入层上方的第一层内有N个节点，需要为该层训练M\*N个权重。权重数目过大会进一步引发一下问题。

数据量：模型中的权重越多，高效训练所需数据量就越多。

计算量：权重越多，训练和使用模型所需的计算就越多。

矢量之间缺乏有意义的联系

**嵌套：将大型稀疏矢量映射到一个保留语义关系的低维空间。**

**转换到低维度空间**

要解决稀疏输入数据的和新问题，可以将高维数据映射到低维空间。

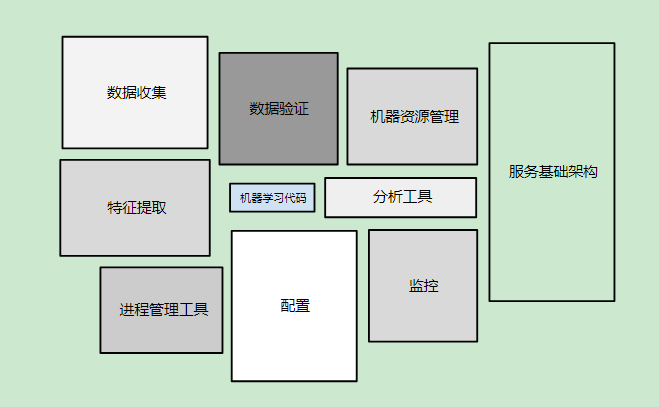
**获取Embeddings**

PCA

Word2vec

**机器学习工程**

前面主要介绍了如何构建机器学习模型，下图是机器学习系统是大型生态系统，模型只是其中的一部分。



静态训练和动态训练

静态模型采用离线训练的方式。也就是说，我们只训练模型一次，然后使用训练后的模型一段时间。

动态模型采用在线训练方式。也就是说，数据会不断进入系统，我们通过不断的更新系统将这些数据整合到模型中。

很多数据源都会随着时间而变化，即便是静态训练，也必须监控输入数据是否发生变化。

**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。

**在线推理：**指的是使用服务器根据需要进行预测。

以下是离线推理的优缺点：

优点：不需要过多担心推理成本。

优点：可以使用批量方法或某些巨型 MapReduce 方法。

优点：可以在推送之前对预测执行后期验证。

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。

缺点：更新可能延迟数小时或数天。

以下是在线推理的优缺点：

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。

缺点：监控需求更多。

**数据依赖关系**

尽量减少数据依赖关系

数据可靠

是否需要添加新特征

特征之间的相关性

反馈环

标签泄露

随机化划分数据集，训练，验证和测试集