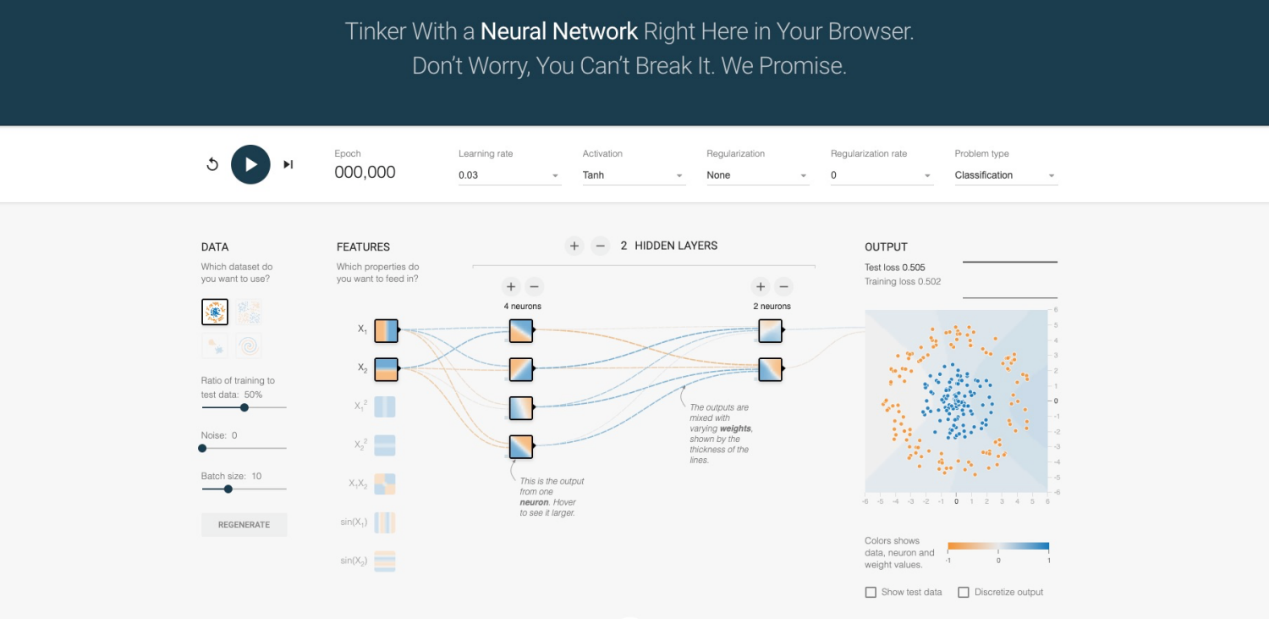
# Tensowflow Playground试用报告

## 概要

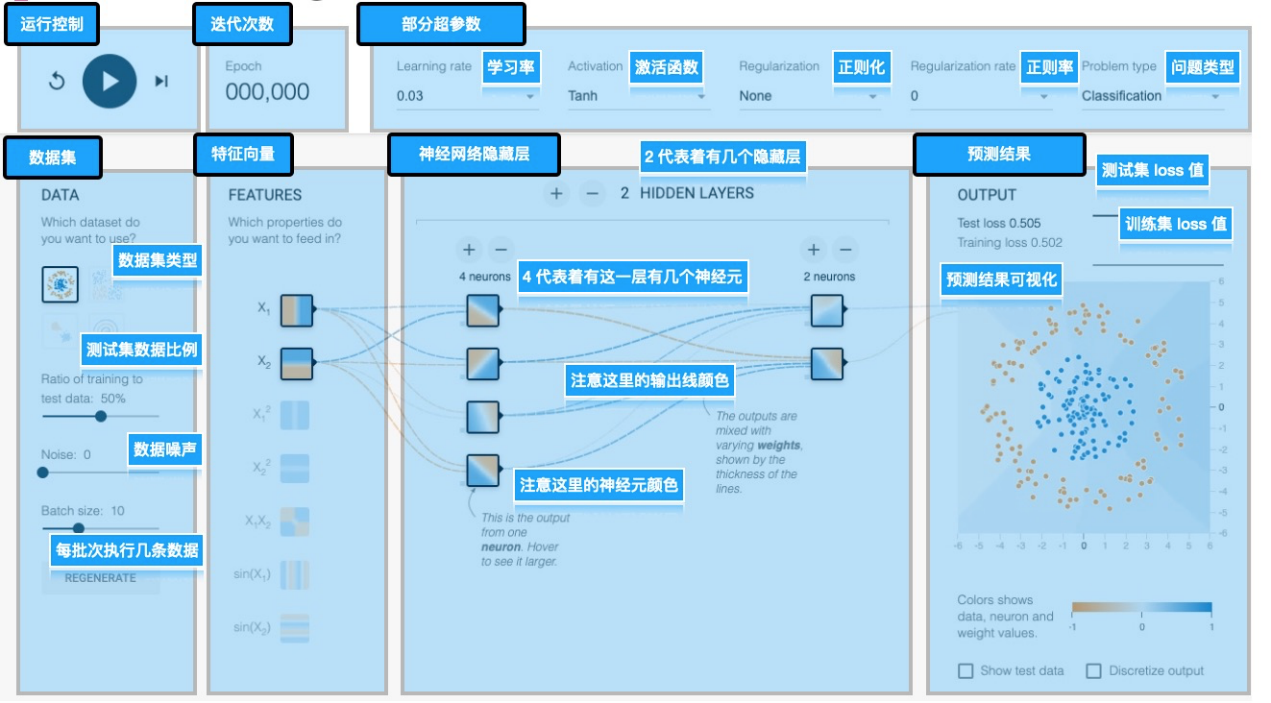
[TenforFlow Playground](https://link.juejin.cn?target=https%3A%2F%2Fgithub.com%2Ftensorflow%2Fplayground) 又名 TensorFlow 游乐场，是一个用来图形化教学的简单神经网络在线演示和实验的平台，非常强大且极其易用。如果我已经有一部分神经网络知识，在这个超级易懂的 demo 里，可以快速体验一个算法工程师的调参工作。



Tensorflow Playground

我对Tensorflow Playground 进行一个细致的布局划分，总体上有如下区域：

* 运行控制区，这里主要对算法执行进行控制，可以启动、暂停和重置
* 迭代次数展示区，这里展示当前算法执行到了哪一次迭代
* 超参数选择区，这里可以调整算法的一些超参数，不同的超参能解决不同的算法问题，得到不同的效果
* 数据集调整区，数据集定义了我们要解决怎样的问题，数据集是机器学习最为重要的一环，
* 特征向量选择，从数据集中攫取出的可以用来被训练的特征值
* 神经网络区域，算法工程师构建的用于数据拟合的网络
* 预测结果区，展示此算法的预测结果



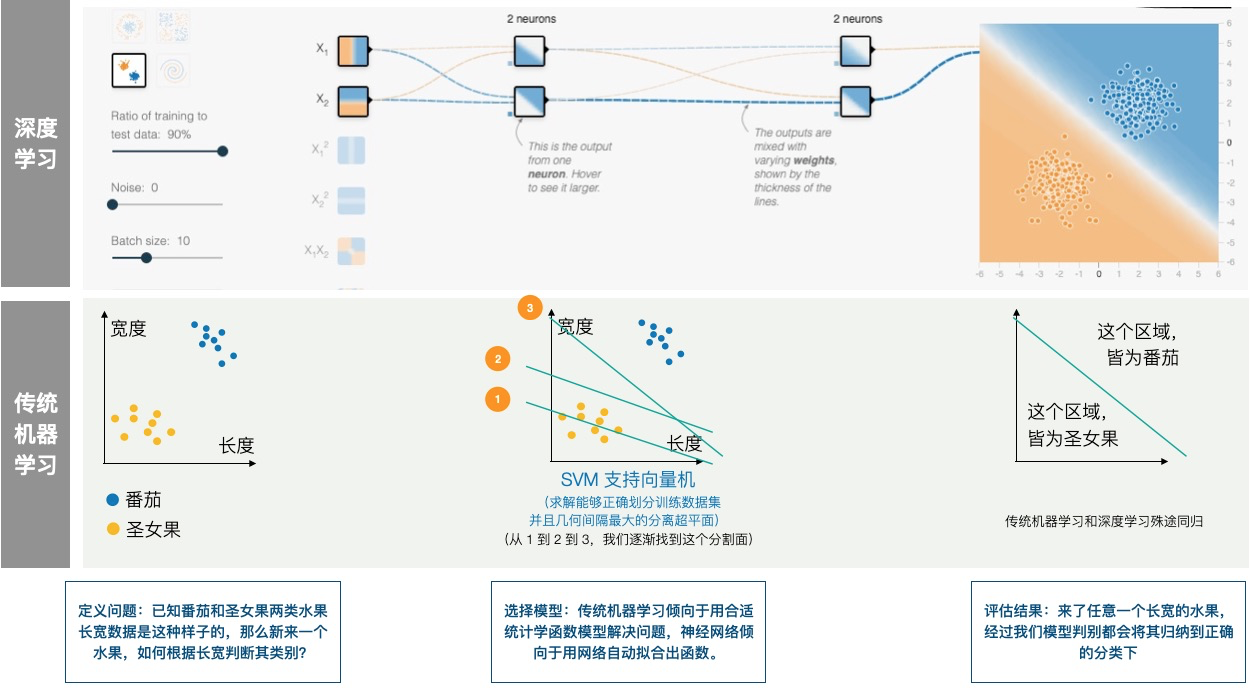
Playground 区域划分

接下来我们会对深度学习做一个总体的介绍，然后对每个区域所涉及的深度学习知识详解。

结合 Playground 自身极强的可视化信息，相信每个读完本文的同学都能畅游其中。

## 1．深度学习介绍

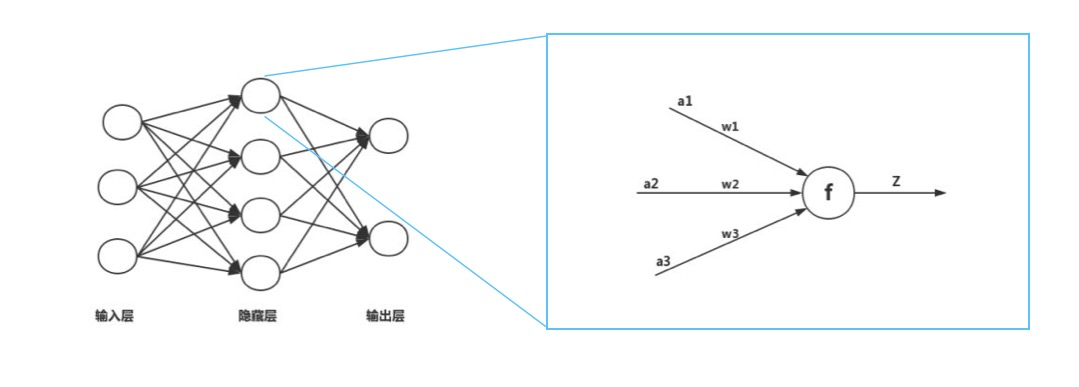
在 TensorFlow Playground 里，我就是一个初级算法工程师。需要选择一个想解决的数据集分类问题，依次调整数据集、超参数、特征向量和隐藏层，点击运行并查看预测结果。回归到深度学习本身，我们要剥离其中晦涩的概念，给出一个零基础的同学也能看懂的说明。机器学习并没有什么魔法，它其实就是一个找出过往数据的内在规律，并对未来新数据进行预测的过程，所有的机器学习，包括深度学习，都是找到从输入到输出的最佳拟合函数的过程，传统机器学习可能用的是一些从统计学中传承的方法，而作为机器学习的一个新领域的深度学习，则是从生物神经网络原理中得到的灵感，用网状结构逐步调整各神经元权重的方法来拟合函数。以 Playground 里第三个数据集为例，我们选择这个数据集，设定任意参数，运行就可以看到 output 区域里的结果是对角划分蓝黄区域的这么一个图。我们需要把节点和颜色分开来看，节点是我们的训练数据，可以看见因为 x,y 值的不同，我们人为的将数据分为了黄蓝两类。现实中这种场景比比皆是，比如番茄和圣女果就可以根据宽高不同划分，假定我们有番茄和圣女果两类数据，一黄一蓝，通过深度或传统机器学习我们得到了一个可以判别水果类型的方法，这个方法可视化之后就是一个区域划分图，坐标系中长宽较大的右上角区域偏蓝，一个新的数据进来，根据其宽高进行判断，落在这里就会被判别为番茄，反之则是圣女果。



传统机器学习和深度学习解题步骤

上图中，传统机器学习的解题核心路径来自统计学，比如 SVM 向量机的基本想法是求解能够正确划分训练数据集并且几何间隔最大的分离超平面。如下图所示， 即为分离超平面，对于线性可分的数据集来说，这样的超平面有无穷多个（即感知机），但是几何间隔最大的分离超平面却是唯一的。

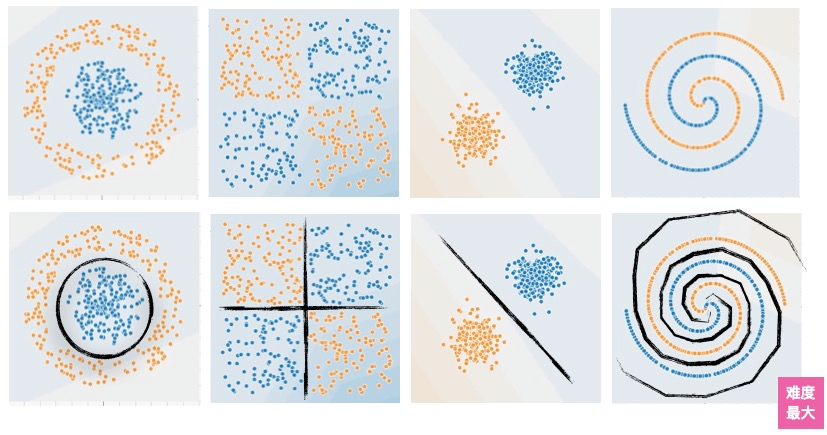
深度学习核心是神经网络，通常一个简单的神经网络包含3部分：输入层、隐含层和输出层，图中的圆圈代表神经元，箭头代表数据的流向，每根连线上面都对应一个不同的权重，权重是网络通过学习自己得到的。网络中每一个神经元都包含了输入、输出以及计算模型3个部分，可以把神经元看成一个计算与存储单元，计算是神经元对其的输入进行计算功能。存储是神经元会暂存计算结果，并传递到下一层。



神经网络原理

**2．数据集**

首先数据集区域，我可以在这里选择四种数据集，每个数据集都有两个类别，这两个类别分别可以用圆形、两条正交直线、对角线和螺旋线的函数来划分。相较于前三个数据集，最后一个数据集函数拟合最困难，是 playground 里调试起来最有难度的数据集。



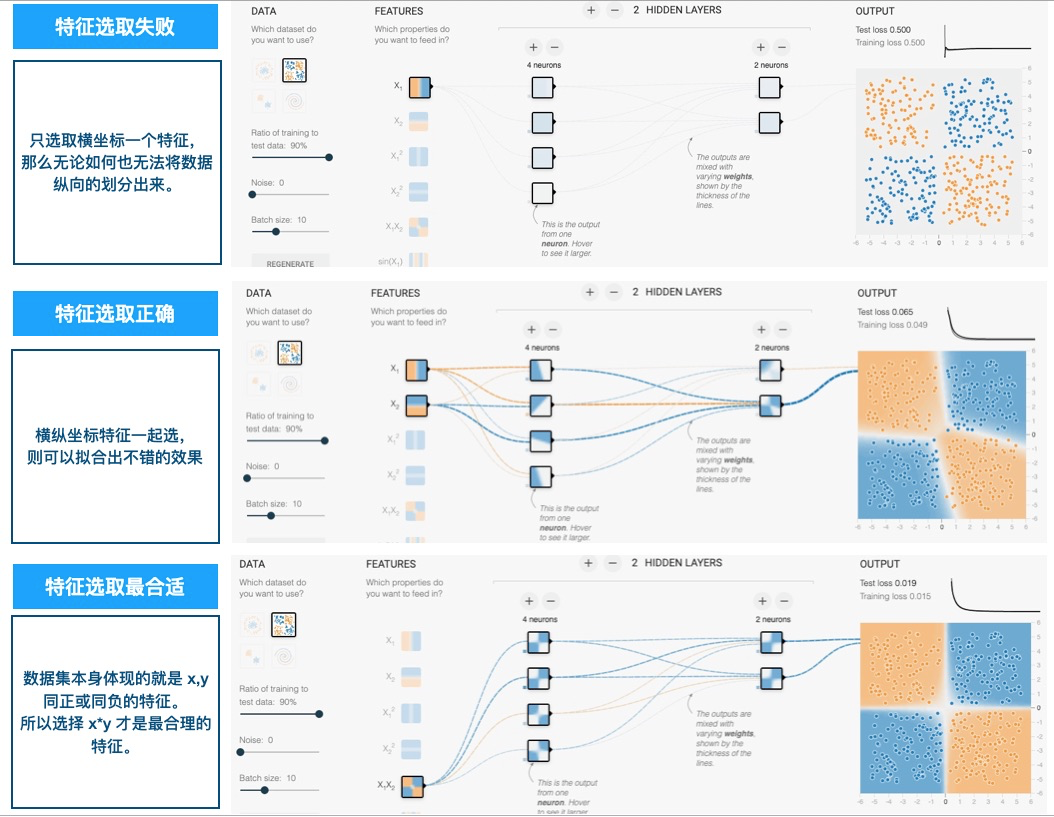
数据集类型

我可以在 Ratio of training to test data 下调整训练与测试数据的比率，一般来说完整的机器学习有训练集、验证集和测试集三类数据集。训练集是程序寻找特征的数据集，一般来说数据量最大。验证集是每个迭代用于在训练过程中检验模型的状态和收敛情况的数据集。测试集用来评价模型泛化能力，就是训练好了一个模型后需要在测试集上验证真实的结果如何。测试集和验证集数量可以稍微少点，吴恩达的深度学习课程里对不同数据量级别的各数据集占比给出了不同的推荐比例，在百万甚至更多的数据量下，验证集和测试集可以占百分之一甚至千分之一的比例，而在数据量偏少的情况下时，验证集和测试集需要避免过小，推荐训练集、验证集、测试集可以按 7：2： 1 的比例分配。tensorflow playground 里面没有验证集概念，所以训练与测试数据比例设为 80%+ 的情况下训练效果最好。

noise 是数据噪声，我可以视情况决定是否启用噪声，噪声越小，训练效果越好。batch size 是每批进入神经网络数据点的个数，我们一个迭代里需要跑完整个数据集，大数据量的情况下一次性的把数据全部塞进网络里显然是不合理的，所以我们可以设定一个 batch size 大小，每次只把 size 大小的数据放进网络里进行训练。

**3．特征**

Feature 一栏包含了可供选择的 7 种特征，这里写的是 x1, x2, 我们不妨直接把 X2 看做是 y 坐标。对于 playground 里的这些用 (x, y) 描述的二维平面数据集，我们可以从 x,y 入手，抽离出很多特征，比如单纯的 x,y 的大小，x,y 平方，sin(x)，sin(y) 正弦分布等。在这里，特征的小方块中同样有 黄蓝两种颜色，这个颜色是数值大小的意思，蓝色代表大，黄色代表小，基本可以认为蓝色是正数，黄色是负数。方块本身代表着整个的取值空间，可以认为中心点就是坐标原点，那么以 x1x2 特征来看，一三象限内值都是蓝色，也就是正数，二四象限内都是负数，很形象的可视化表述了我们的数据特征。



选择合适的特征很重要

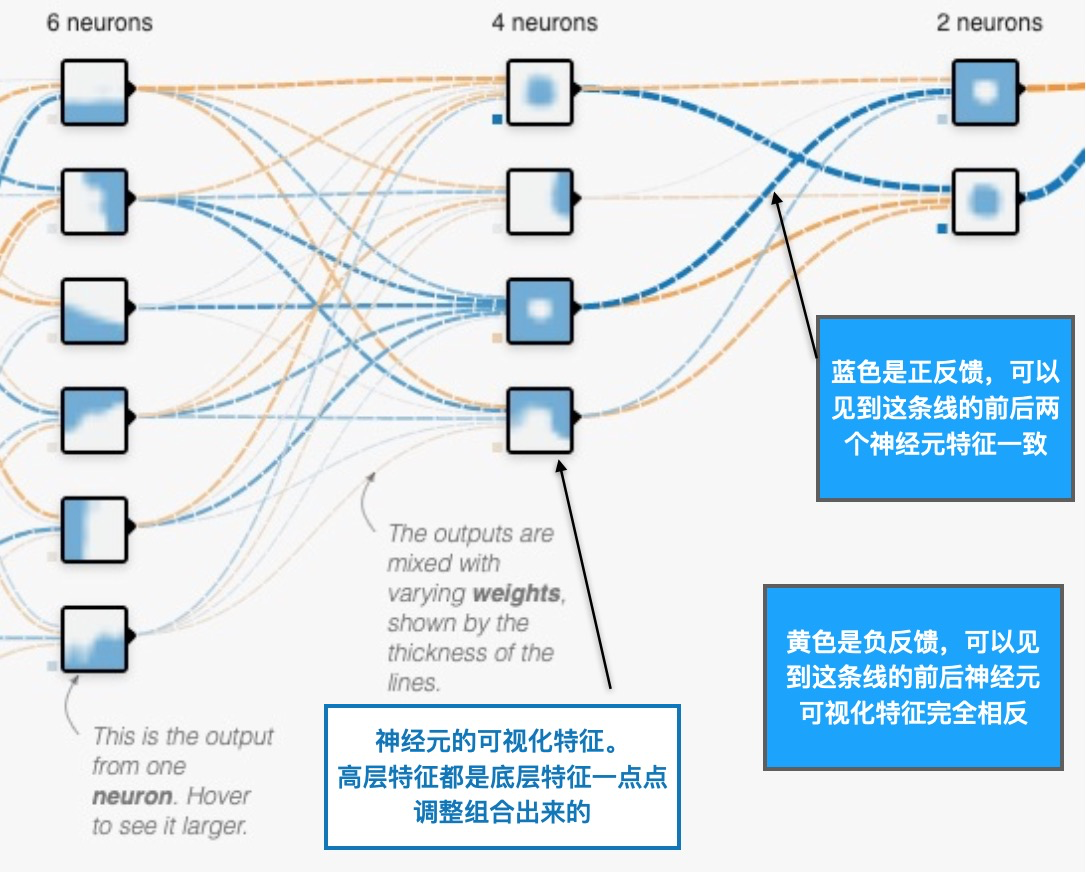
上图用一个例子形象描述了选择合适特征的重要性，一个需要横纵坐标数据一起作用才能预测的数据集，只凭横坐标一个特征不够的，因为数据中和分类真正有关的有效信息没有传进网络，那么无论多少隐藏层多少神经元都无法实现对数据的有效划分。

上图我们只用横坐标尝试失败了，添加了纵坐标特征就成功了，然后我们选择最为合适的 x\*y 的特征，则效果出乎意料的好。

**4．隐藏层 hidden layers**

一般来说，TensorFlow 、keras 等框架的模型的训练结果包含了模型自身结构信息和隐藏层权重信息两部分，所以神经网络的训练其实就是隐藏层中各个节点权重值的训练。我们可以通过解释 playground 里面的颜色来感知隐藏层权重的意义，他们的颜色和数据点蓝黄颜色的意义还稍稍不一样：

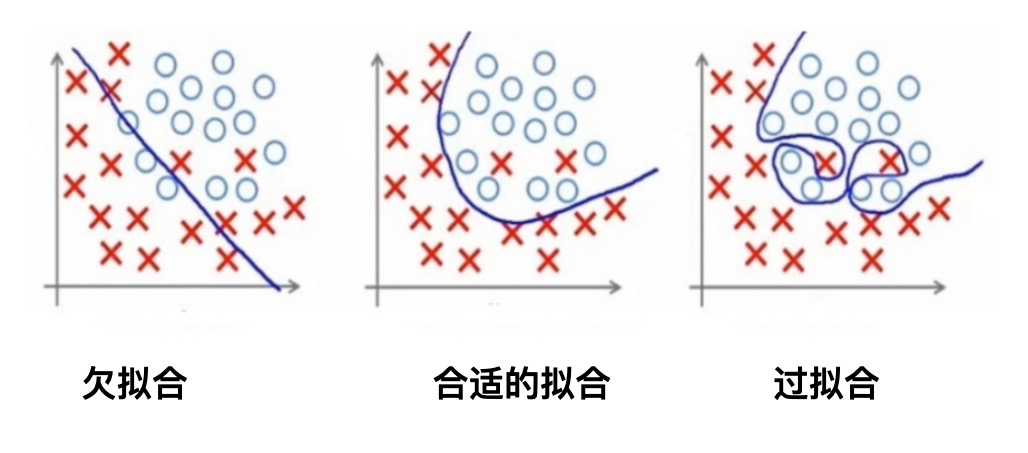
* 每个神经元只有蓝白两色，蓝色部分是此神经元的敏感区域
* 每个神经元输出的线有蓝黄两色，蓝色是正反馈，黄色是负反馈。越粗则说明下游节点眼里这个特征越重要



神经元和权重的可视化

hidden layers 一栏可以设置很多隐藏层。一般来说，隐藏层越多，衍生出的特征类型也就越丰富，分类的效果也会越好。但是层数多了训练速度会变慢，也可能不会有效收敛，容易过拟合。

简单来说，我们要用神经网络拟合出一个可以划分两个数据边界的线，隐藏层的深度和大小就是这个线的细致程度，如果隐藏层越多那么线就会越细致，就会有越多的点被准确划分。但是过于细致也不行，因为如果数据存在噪声，比如一个数据错误的跑到了对面去了，那我这条线就可能要为这个数据绕很大一个圈，这样虽然训练集的准确率高了，每个训练数据的特征都被完美的学到了，但是其实它的划分线并不合理，新来一个数据很有可能就判断出错，如下图所示：



分类 classification 问题中的三种拟合状态

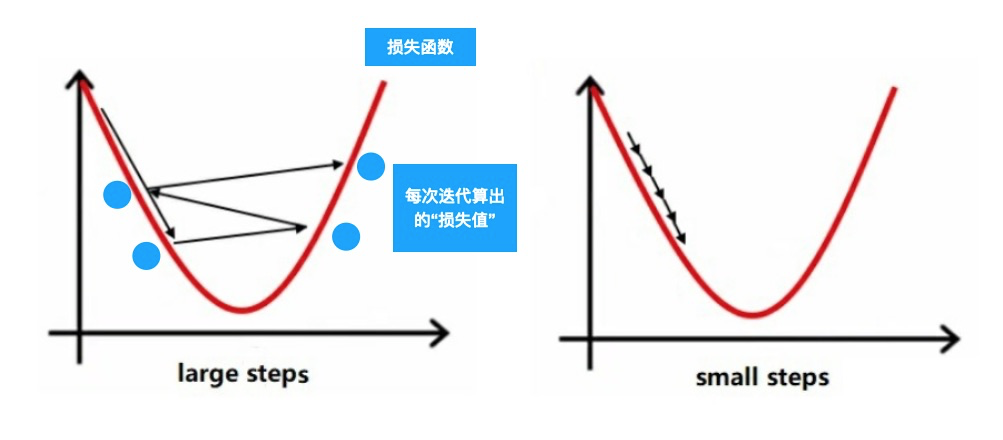
所以，机器学习中一个很常见的问题就是解决过拟合问题，解决它，就要求我们的网络或是其中的一些步骤具有一定的“模糊”效果。只要模糊了，拟合出的函数就会张弛有度，不会吹毛求疵。

**5．迭代 Epoch**

迭代理念是机器学习不同于传统统计学分析的最大的点之一，首先我们要知道，机器学习是基于统计学的。机器学习建立在统计框架之上，作为一门处理数据的学科，不经过统计框架直接描述数据是不可能的。除此之外，机器学习也利用了大量其他数据和计算机科学知识。比如理论层面来自数学和统计学等领域，算法层面来自优化、矩阵代数、微积分等领域，而实现层又来自计算机科学与工程学概念。虽然最终你使用 tensorflow 来做算法工作时很多概念都不会有感知，但是究其来源才是掌握一门知识的正确方法。

回到我们“迭代”这个概念上来，我们以监督学习为例，我们已经有了一些数据，目标是找到将 x 值映射到 y 值的函数，可以描述此映射的所有可能函数集合我们成为假设空间，神经网络中每个节点的权重组成的权重网络就是假设空间中的每个“函数集合”。要找到这个最准确的函数，或是这个最准确的权重网络，我们就需要让算法有一些逐步逼近并找出最优解的最佳实践。

这个最佳实践就叫损失函数的求解过程。我们对于最终结果建立起一套“风险函数评估机制”，每次在假设空间中检索的时候都会将这个风险函数计算出的数据拿出来实时查看，始终往风险低的方向走。如果我们在假设空间里是往正确的方向前进，则我们的风险函数（损失函数）也会越来越低，最后趋于一个最低值。



损失函数求解过程

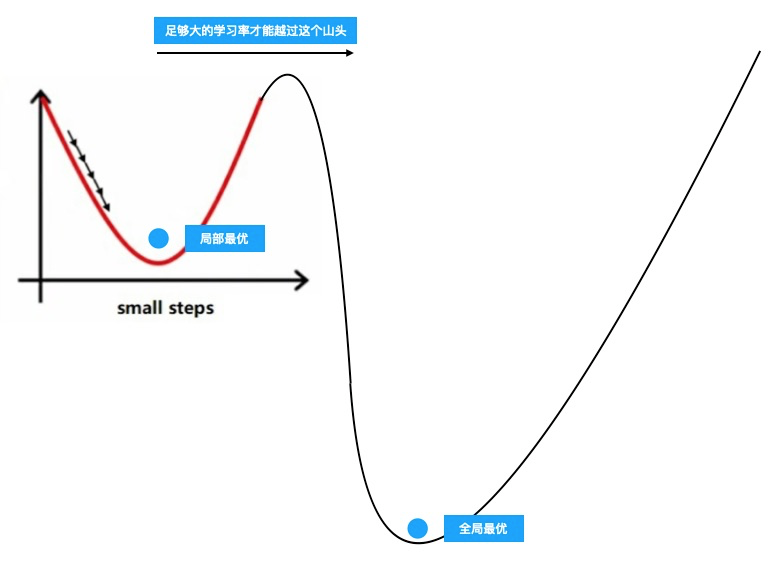
这里计算损失函数的步骤都是在全量的数据跑完之后进行的，全量的数据在网络里跑完一次后计算出当前的损失函数大小，然后往下个方向继续行进。这个过程就是一个迭代（Epoch），我们在 Tensorflow Playground 里面运行算法时，随着 Epoch 的增加，Output 里 loss（损失函数值）也在随之变化，直到 N个迭代后到达损失函数最优点。这时候我们可以说，已经找到了当前数据样本的最佳拟合函数，我们的模型已经训练完成。

**6．超参数**

除了隐藏层的个数和神经元个数，我们还有很多左右训练结果的初始参数，他们都叫超参数。

**6.1学习率 Learning rete**

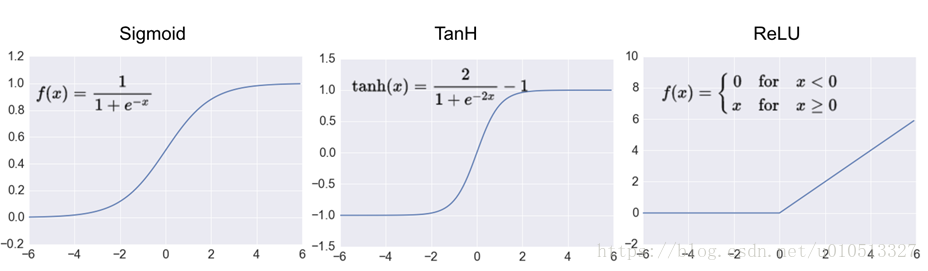
还是看上面的损失函数求解过程，学习率就是每个迭代完之后的行进步长，步长过大则有可能越过了最优解，反复横跳不能得到最优效果。步长过小则不仅收敛效果慢，还有可能如下图一样陷入局部最优，这时候加大学习率才能越过山头找到真正的最优解。总之，选用合适的学习率很重要，不同的模型也有不同的合适学习率可选择。甚至你可以用一些动态学习率的方法来在运算的过程中动态调整学习率，loss 下降明显时加大学习率，loss 趋缓时减小学习率。



陷入局部最优

**6.2激活函数 Activation**

激活函数是神经网络独有的概念，可以这么说，没有激活函数，那么神经网络是画不出那一条细致的分类线的，再多的隐藏层和神经元都只能复合出线性运算，只能画直线拟合线性函数。激活函数在 Playground 里面供我们选择的有 Relu、Tanh、Sigmoid 等，他们样子如下：



激活函数对比

* sigmoid 左端趋近于 0，右端趋近于 1，两端趋于饱和。饱和就容易出现差别过小，从而梯度消失。
* TanH 函数和 sigmoid 趋势类似，但是比起 sigmoid 是中心对称的
* ReLU 函数是个分段线性函数，是目前用的最多的激活函数。因为正值区间是线性的，所以很好的解决了梯度消失的问题。计算速度也快，收敛速度远快于 sigmoid 和 tanh。Relu 的模型可以做的很深，也带来了一批模型的突破

**6.3正则化 Regularization 和 正则化率 Regularization Rate**

正则化是解决过拟合的手段之一，使用正则化可以降低模型的复杂度，增强模型的泛化能力。正则化常用的几种方法有 L1 和 L2。

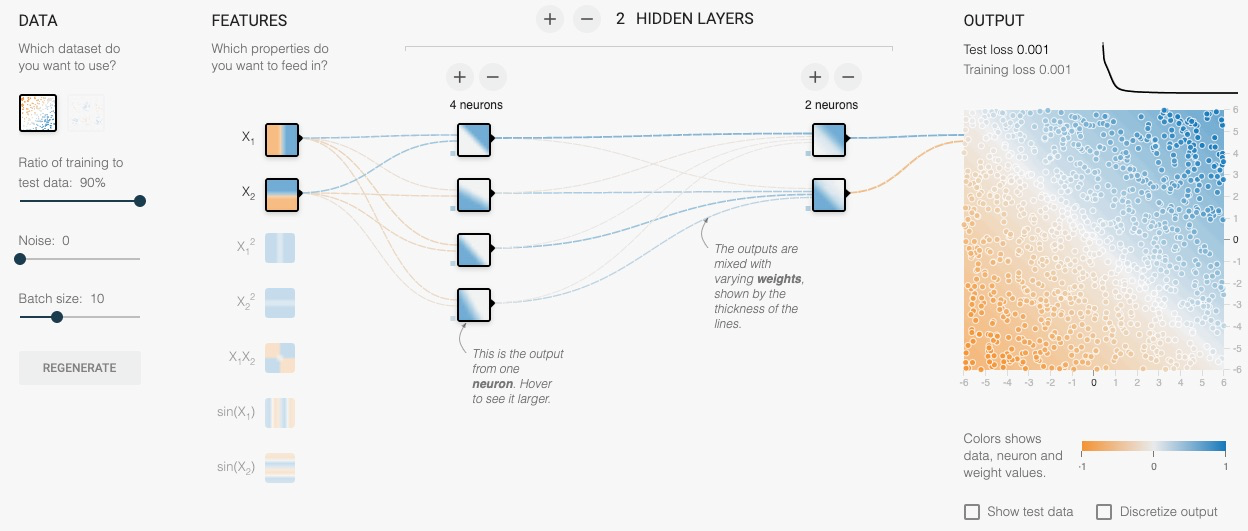
* L1 范数：权值向量 w 中各个元素的绝对值之和。
* L2 范数：权值向量 w 中各个元素的平方和求平方根

正则化率就是正则化的程度，提高正则化率可以减少过拟合，但是也要注意不要过高导致欠拟合。

**6.4Problem Type**

可以更换问题类型，本文介绍的都是分类问题，目的是给定一个数据，我们猜测他的分类。

另一种问题类型叫回归问题，给定一个数据，猜测的结果不需要是分类，而是一个数值。



**7.模型评估**

最后我们可以在 Playground 的右侧查看本次训练的模型的好坏。

Loss 值本质上可以认为就是模型的好坏，loss 越小，说明我们模型当前估算出的函数越贴近数据集。所以每个算法的 loss 函数的计算方式都是算法核心之一，定义了 loss 函数等于定义了问题解题路径。

