神经网络

https://www.zhihu.com/question/22553761/answer/126474394

- 理解神经网络的工作原理；
- 训练什么网络的流程；
- 一步步实现神经网络，并通过神经网络拟合出函数；
- 方向传播 back propagation

1. 神经网络的层

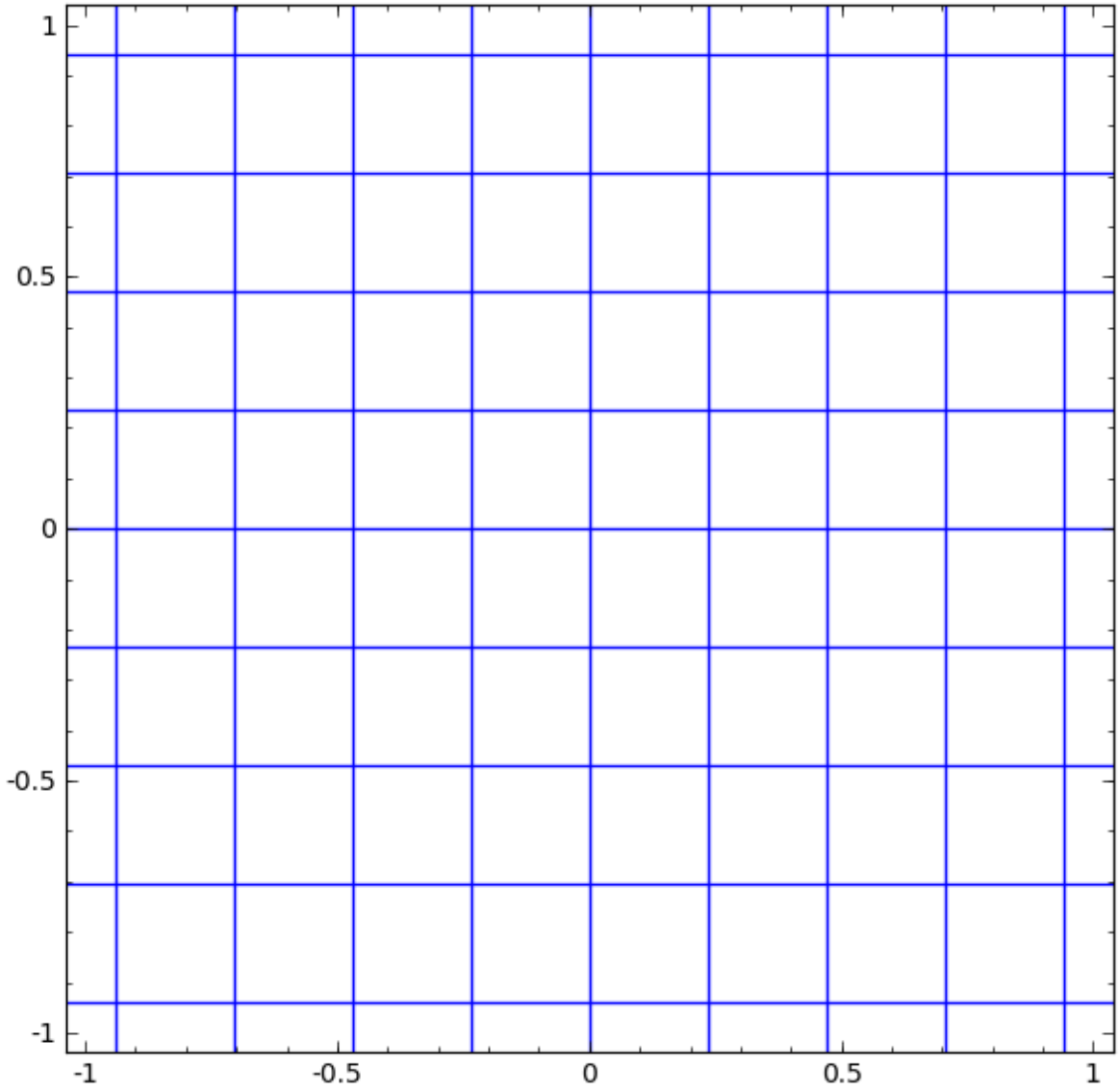
（1）数学角度解释

数学式子：$\vec{y} = a(W \cdot \vec{x} + b)$，其中$\vec{x}$是输入向量，$\vec{y}$是输出向量，$\vec{b}$是偏移向量，$W$是权重矩阵，$a()$是激活函数。每一层仅仅是把输入$\vec{x}$经过如此简单的操作得到$\vec{y}$。

数学理解：通过如下5种对输入空间（输入向量的集合）的操作，完成 **输入空间 ——> 输出空间** 的变换 (矩阵的行空间到列空间)。

- **1.** 升维/降维
- **2.** 放大/缩小
- **3.** 旋转
- **4.** 平移
- **5.** “弯曲”

  这5种操作中，1,2,3的操作由$W \cdot \vec{x}$完成，4的操作是由$+\vec{b}$完成，5的操作则是由$a()$来实现

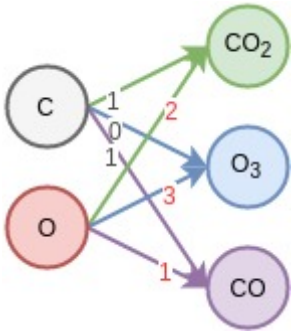


> 每层神经网络的数学理解：**用线性变换跟随着非线性变化，将输入空间投向另一个空间**。

（2）物理角度解释

物理理解：对 $\boldsymbol{W}\cdot\vec{\boldsymbol{x}}$ 的理解就是**通过组合形成新物质**。$\boldsymbol{a}()$又符合了我们所处的世界都是非线性的特点。

- **情景：**$\vec{\boldsymbol{x}}$是二维向量，维度是碳原子和氧原子的数量$[C;O]$，数值且定为$[1;1]$，若确定$\vec{\boldsymbol{y}}$是三维向量，就会形成如下网络的形状 (神经网络的每个节点表示一个维度)。通过改变权重的值，可以获得若干个不同物质。右侧的节点数决定了想要获得多少种不同的新物质。（矩阵的行数）



- 如果权重W的数值如上图，那么网络的输出$\vec{y}$ 就会是三个新物质，[二氧化碳，臭氧，一氧化碳]。
$$\begin{bmatrix} CO_2 \\ O_3 \\ CO \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 0 & 3 \\ 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} C \\ O \end{bmatrix}$$
- 也可以减少右侧的一个节点，并改变权重W至（2），那么输出$\vec{y}$ 就会是两个新物质，$[O_{0.3}; CO_{1.5}]$。
$$\begin{bmatrix} O_{0.3} \\ CO_{1.5} \end{bmatrix} = \begin{bmatrix} 0 & 0.3 \\ 1 & 1.5 \end{bmatrix} \cdot \begin{bmatrix} C \\ O \end{bmatrix} \quad (2)$$
- 如果希望通过层网络能够从[C, O]空间转变到$[CO_2; O_3; CO]$空间的话，那么网络的学习过程就是将W的数值变成尽可能接近(1)的过程 。如果再加一层，就是通过组合$[CO_2; O_3; CO]$这三种基础物质，形成若干更高层的物质。
- 重要的是这种组合思想，组合成的东西在神经网络中并不需要有物理意义。

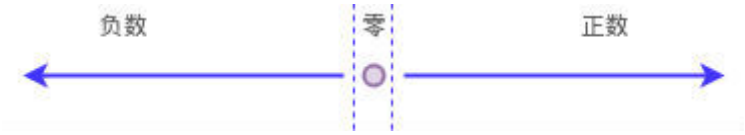
> 每层神经网络的物理理解：**通过现有的不同物质的组合形成新物质**。

2. 神经网络的工作原理

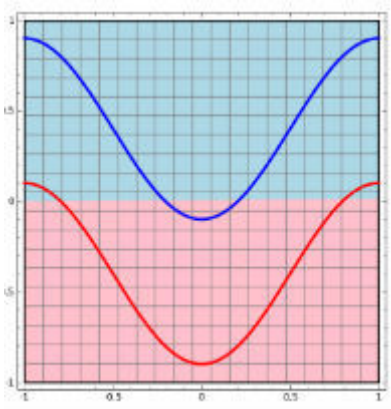
现在我们知道了每一层的行为，但这种行为又是如何完成识别任务的呢？

（1）数学视角："线性可分"

- **一维情景**：以分类为例，当要分类正数、负数、零，三类的时候，一维空间的直线可以找到两个超平面（比当前空间低一维的子空间。当前空间是直线的话，超平面就是点）分割这三类。但面对像分类奇数和偶数无法找到可以区分它们的点的时候，我们借助 x % 2（取余）的转变，把x变换到另一个空间下来比较，从而分割。

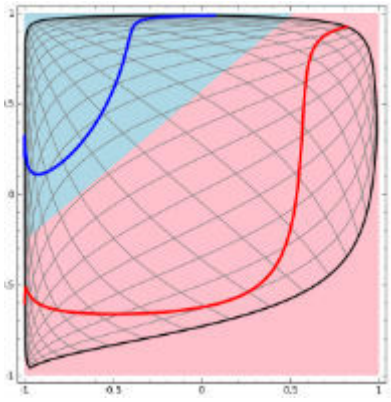


- **二维情景**：平面的四个象限也是线性可分。但下图的红蓝两条线就无法找到一超平面去分割。



神经网络的解决方法依旧是转换到另外一个空间下，用的是所说的**5种空间变换操作**。

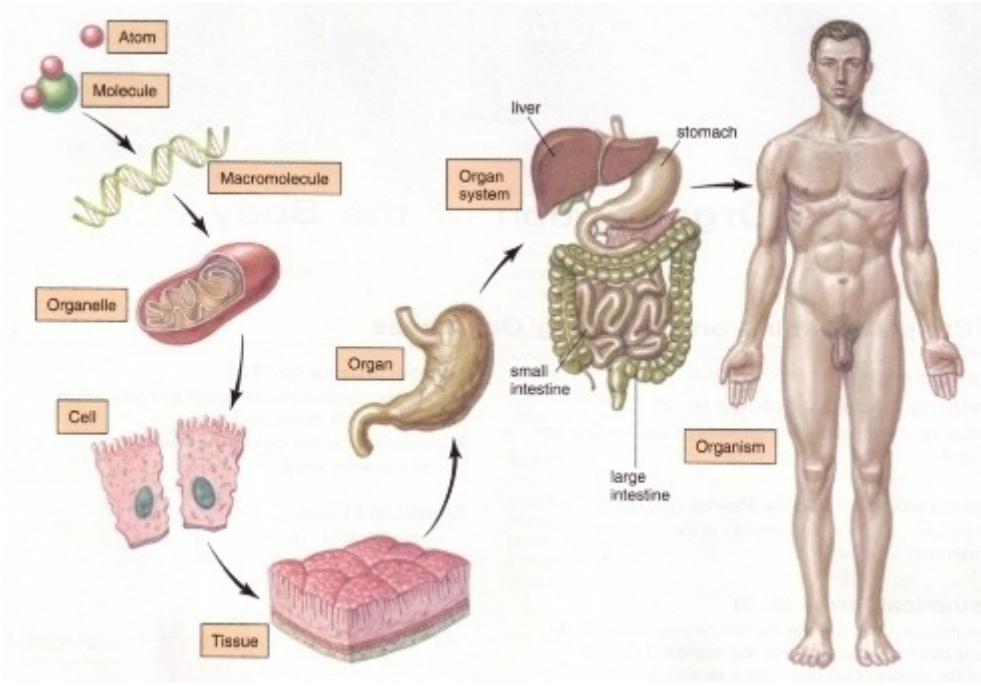
比如下图就是经过放大、平移、旋转、扭曲原二维空间后，在三维空间下就可以成功找到一个超平面分割红蓝两线 。



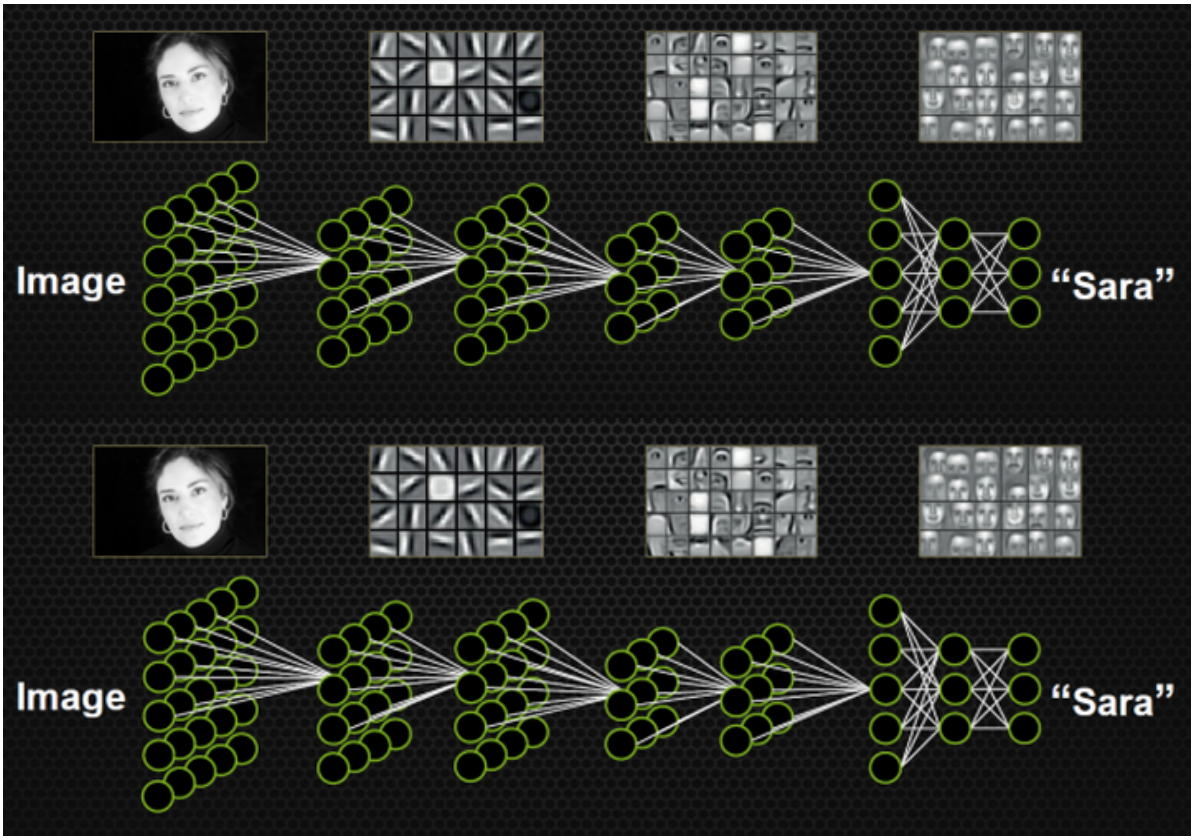
> 线性可分视角：神经网络的学习就是**学习如何利用矩阵的线性变换加激活函数的非线性变换，将原始输入空间投向线性可分/稀疏的空间去分类/回归。**
>
> **增加节点数：增加维度，即增加线性转换能力。**
>
> **增加层数：增加激活函数的次数，即增加非线性转换次数。**

(2) 物理视角："物质组成"

- **类比**：回想上文由碳氧原子通过不同组合形成若干分子的例子。从分子层面继续迭代这种组合思想，可以形成DNA，细胞，组织，器官，最终可以形成一个完整的人。继续迭代还会有家庭，公司，国家等。这种现象在身边随处可见。并且原子的内部结构与太阳系又惊人的相似。不同层级之间都是以类似的几种规则再不断形成新物质。你也可能听过**分形学**这三个字。可通过观看从1米到150亿光年来感受自然界这种层级现象的普遍性。



- **人脸识别情景**：我们可以模拟这种思想并应用在画面识别上。由像素组成菱角再组成五官最后到不同的人脸。每一层代表不同的不同的物质层面 (如分子层)。而每层的W**存储着如何组合上一层的物质从而形成新物质**。
  如果我们完全掌握一架飞机是如何从分子开始一层一层形成的，拿到一堆分子后，我们就可以判断他们是否可以以此形成方式，形成一架飞机。



> 物质组成视角：神经网络的学习过程就是**学习物质组成方式的过程。**
>
> **增加节点数：增加同一层物质的种类，比如118个元素的原子层就有118个节点。**
>
> **增加层数：增加更多层级，比如分子层，原子层，器官层，并通过判断更抽象的概念来识别物体。**

3. 训练神经网络

神经网络学习到控制空间变换的权重矩阵。如何学习到的呢？

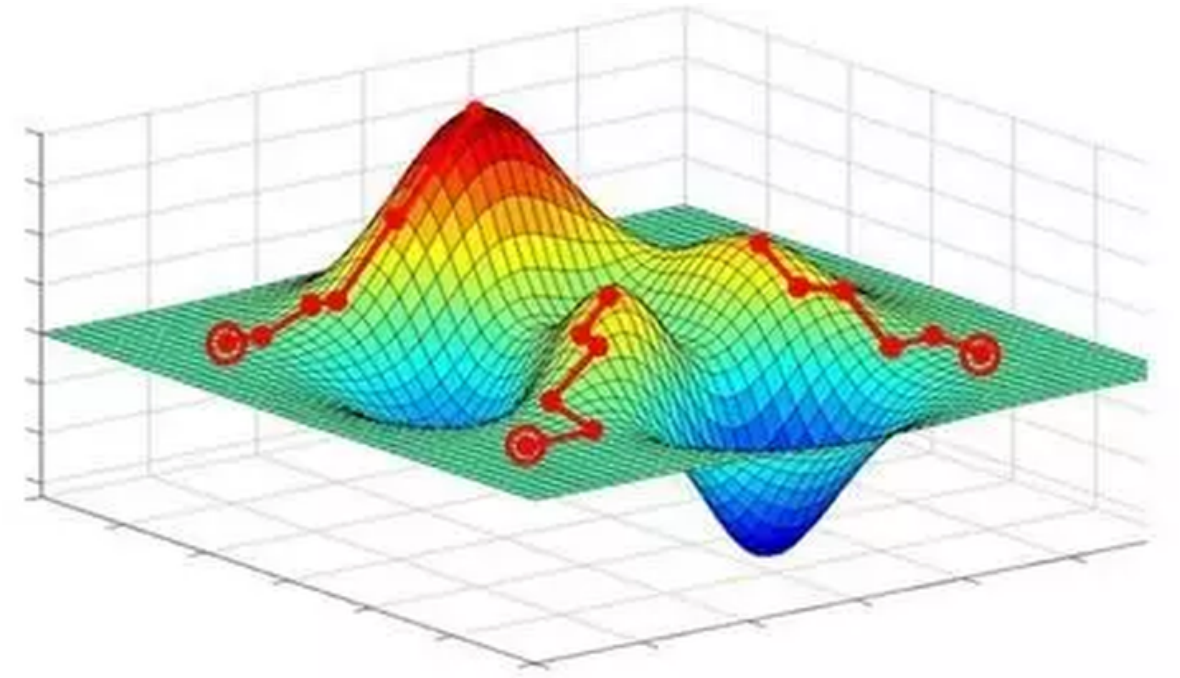
（1）如何训练：

既然我们希望网络的输出尽可能的接近真正想要预测的值。那么就可以通过**比较**当前网络的**预测值**和我们真正想要的**目标值**再根据两者的差异情况来更新每一层的权重矩阵比如，如果网络的预测值高了，就调整权重让它预测低一些，不断调整，直到能够预测出目标值）。因此就需要先**定义“如何比较**预测值和目标值的**差异**”，**损失函数或目标函数（loss function or objective function）**用于衡量预测值和目标值的差异的方程。loss function的输出值（loss）越高表示差异性越大。那神经网络的训练就变成了尽可能的缩小loss的过程。所用的方法是**梯度下降（Gradient descent）**：通过使loss值向当前点对应梯度的反方向不断移动，来降低loss。一次移动多少是由**学习速率（learning rate）**来控制的。

（2）梯度下降的问题：

- **局部极小值**

梯度下降寻找的是loss function的局部极小值，而我们想要全局最小值。如下图所示，我们希望loss值可以降低到右侧深蓝色的最低点，但loss有可能“卡”在左侧的局部极小值中。



试图解决“卡在局部极小值”问题的方法分两大类：

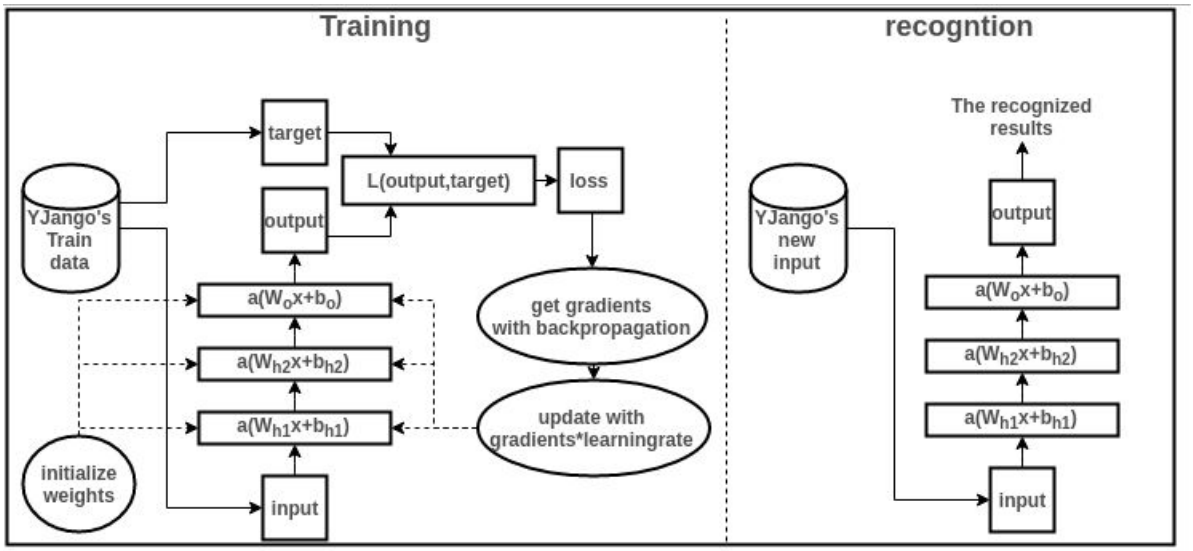
- **调节步伐：**调节学习速率，使每一次的更新“步伐”不同。常用方法有：
  - 随机梯度下降（Stochastic Gradient Descent (SGD)：每次只更新一个样本所计算的梯度
  - 小批量梯度下降（Mini-batch gradient descent）：每次更新若干样本所计算的梯度的平均值
  - 动量（Momentum）：不仅仅考虑当前样本所计算的梯度；Nesterov动量（Nesterov Momentum）：Momentum的改进
    - Adagrad、RMSProp、Adadelta、**Adam**：这些方法都是训练过程中依照规则降低学习速率，部分也综合动量
- **优化起点**：合理初始化权重（weights initialization）、预训练网络（pre-train），使网络获得一个较好的“起始点”，如最右侧的起始点就比最左侧的起始点要好。常用方法有：高斯分布初始权重（Gaussian distribution）、均匀分布初始权重（Uniform distribution）、Glorot 初始权重、He 初始权、稀疏矩阵初始权重（sparse matrix）

（3）梯度的计算

机器学习所处理的数据都是高维数据，该**如何快速计算梯度**、而不是以年来计算。
其次如何更新**隐藏层**的权重？
解决方法是：计算图：**反向传播算法**
需要知道的是，**反向传播算法是求梯度的一种方法**。如同快速傅里叶变换（FFT）的贡献。
而计算图的概念又使梯度的计算更加合理方便。

基本流程图：

下面就结合图简单浏览一下训练和识别过程，并描述各个部分的作用。



- **收集训练集（train data）**：也就是同时有input以及对应label的数据。每个数据叫做训练样本（sample）。label也叫target，也是机器学习中最贵的部分。上图表示的是我的数据库。
- **设计网络结构（architecture）**：确定层数、每一隐藏层的**节点数**和**激活函数**，以及**输出层的激活函数**和**损失函数**。上图用的是两层隐藏层（最后一层是输出层）。隐藏层所用激活函数a( )是ReLu，输出层的激活函数是线性linear（也可看成是没有激活函数）。隐藏层都是1000节点。损失函数L( )是用于比较距离**MSE：mean((output - target)^2)**。MSE越小表示预测效果越好。训练过程就是不断减小MSE的过程。
- **数据预处理（preprocessing）**：将所有样本的input和label处理成能够使用神经网络的数据，label的值域符合激活函数的值域。并简单优化数据以便让训练易于收敛。比如中心化（mean subtraction）、归一化（normlization）、主成分分析（PCA）、白化（whitening）。假设上图的input和output全都经过了中心化和归一化。
- **权重初始化（weights initialization）**：W_{h1}, W_{h2}, W_0在训练前不能为空，要初始化才能够计算loss从而来降低。W_{h1}, W_{h2}, W_0初始化决定了loss在loss function中从哪个点开始作为起点训练网络。
- **训练网络（training）**：训练过程就是用训练数据的input经过网络计算出output，再和label计算出loss，再计算出gradients来更新weights的过程。
- - 正向传递：，算当前网络的预测值
    $$output = linear(W_o \cdot Relu(W_{h2} \cdot Relu(W_{h1} \cdot input + b_{h1}) + b_{h2}) + b_o)$$
  - 计算loss：$loss = mean((output - target)^2)$
  - 计算梯度：从loss开始反向传播计算每个参数（parameters）对应的梯度（gradients）。这里用Stochastic Gradient Descent (SGD) 来计算梯度，即每次更新所计算的梯度都是从一个样本计算出来的。
- - 更新权重：这里用最简单的方法来更新，即所有参数都
    $$W = W - learningrate * gradient$$
  - 预测新值：训练过所有样本后，打乱样本顺序再次训练若干次。训练完毕后，当再来新的数据input，就可以利用训练的网络来预测了。这时的output就是效果很好的预测值了。下图是一张**实际值**和**预测值**的三组对比图。