深层学习为何要“Deep”（上）

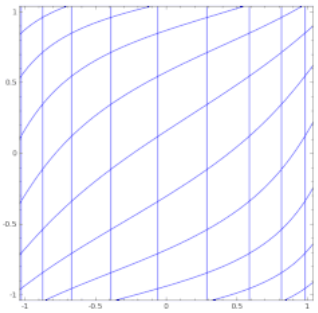
介绍

为了研究神经网络，我们必须要对什么网络是什么有一个更直观的认识。

一、基本变换：层

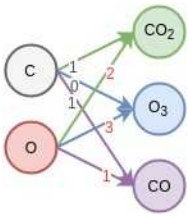
神经网络是由一层一层构建的，那么每**层**究竟在做什么？

- **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)。 **注：**用“空间”二字的原因是被分类的并不是单个事物，而是一**类**事物。空间是指这类事物所有个体的集合。
    - **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。**

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



- **1.** 如果权重W的数值如（1），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]。 $\begin{bmatrix} CO_2 \\ O_3 \\ CO \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 0 & 3 \\ 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} C \\ O \end{bmatrix}$ （1）
- **2.** 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，$[O_{0.3}; CO_{1.5}]$。 $\begin{bmatrix} O_{0.3} \\ CO_{1.5} \end{bmatrix} = \begin{bmatrix} 0 & 0.3 \\ 1 & 1.5 \end{bmatrix} \cdot \begin{bmatrix} C \\ O \end{bmatrix}$ （2）
- **3.** 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。
- **4.** 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。

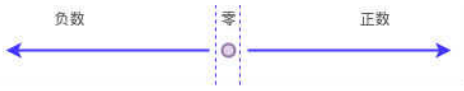
> 每层神经网络的物理理解：**通过现有的不同物质的组合形成新物质。**

二、理解视角：

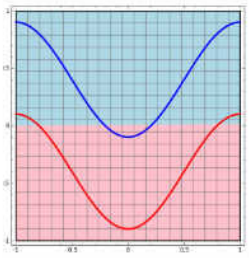
现在我们知道了每一层的行为，但这种行为又是如何完成识别任务的呢？

数学视角：“线性可分”

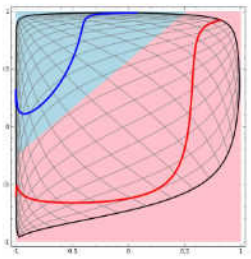
- **一维情景**：以分类为例，当要分类正数、负数、零，三类的时候，一维空间的直线可以找到两个超平面（比当前空间低一维的子空间。当前空间是直线的话，超平面就是点）分割这三类。但面对像分类奇数和偶数无法找到可以区分它们的点的时候，我们借助 x % 2（取余）的转变，把x变换到另一个空间下来比较，从而分割。



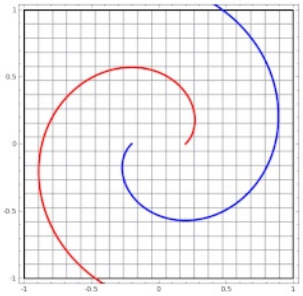
- **二维情景**：平面的四个象限也是线性可分。但下图的红蓝两条线就无法找到一超平面去分割。



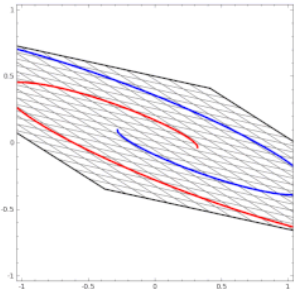
神经网络的解决方法依旧是转换到另外一个空间下，用的是所说的**5种空间变换操作**。比如下图就是经过放大、平移、旋转、扭曲原二维空间后，在三维空间下就可以成功找到一个超平面分割红蓝两线 (同SVM的思路一样)。



上面是一层神经网络可以做到的，如果把 $\vec{y}$ 当做新的输入再次用这5种操作进行第二遍空间变换的话，网络也就变为了二层。最终输出是 $\vec{y} = a_2(W_2 \cdot (a_1(W_1 \cdot \vec{x} + b_1)) + b_2)$。 设想网络拥有很多层时，对原始输入空间的“扭曲力”会大幅增加，如下图，最终我们可以轻松找到一个超平面分割空间。



当然也有如下图失败的时候，关键在于“如何扭曲空间”。所谓监督学习就是给予神经网络网络大量的训练例子，让网络从训练例子中学会如何变换空间。每一层的权重W就**控制着如何变换空间**，我们最终需要的也就是训练好的神经网络的所有层的权重矩阵。

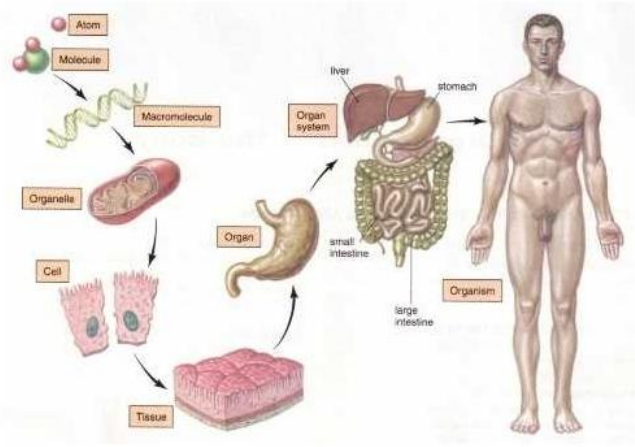


这里有非常棒的可视化空间变换demo，一定要打开尝试并感受这种扭曲过程。更多内容请看Neural Networks, Manifolds, and Topology。

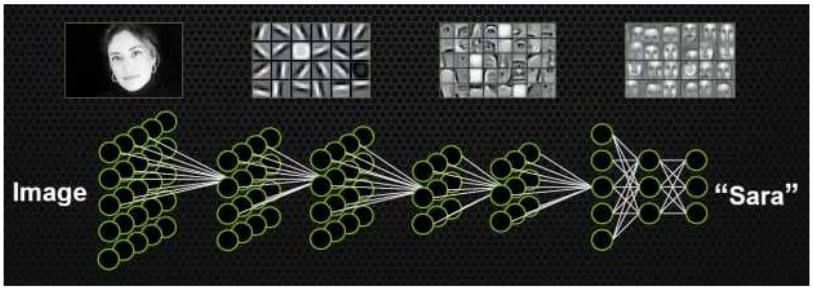
> 线性可分视角：神经网络的学习就是**学习如何利用矩阵的线性变换加激活函数的非线性变换，将原始输入空间投向线性可分/稀疏的空间去分类/回归。**
> **增加节点数：增加维度，即增加线性转换能力。**
> **增加层数：增加激活函数的次数，即增加非线性转换次数。**

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。 附：Tensorflow playground展示了数据是如何“流动”的。



> 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。**

三、神经网络的训练

知道了神经网络的学习过程就是**学习**控制着空间变换方式（物质组成方式）的**权重矩阵**后，接下来的问题就是**如何学习**每一层的权重矩阵 W 。

如何训练：

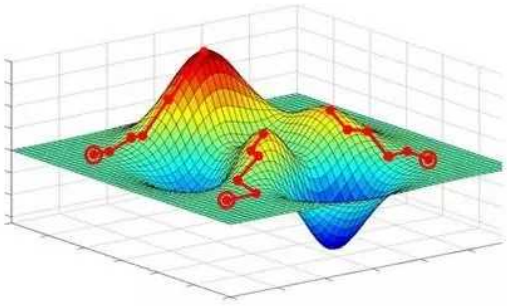
既然我们希望网络的输出尽可能的接近真正想要预测的值。那么就可以通过**比较**当前网络的**预测值**和我们真正想要的**目标值**，再根据两者的差异情况来更新每一层的权重矩阵（比如，如果网络的预测值高了，就调整权重让它预测低一些，不断调整，直到能够预测出目标值）。因此就需要先**定义“如何比较**预测值和目标值的**差异**”，这便是**损失函数或目标函数（loss function or objective function）**，用于衡量预测值和目标值的差异的方程。loss function的输出值（loss）越高表示差异性越大。那神经网络的训练就变成了尽可能的缩小loss的过程。 所用的方法是**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。

梯度下降的问题：

然而使用梯度下降训练神经网络拥有两个主要难题。

1、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：

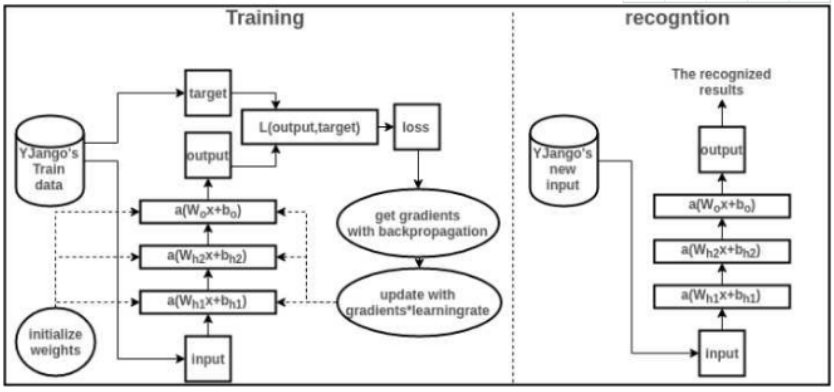
- **调节步伐：**调节学习速率，使每一次的更新“步伐”不同。常用方法有：
- 随机梯度下降（Stochastic Gradient Descent (SGD)：每次只更新一个样本所计算的梯度
- 小批量梯度下降（Mini-batch gradient descent）：每次更新若干样本所计算的梯度的平均值
- 动量（Momentum）：不仅仅考虑当前样本所计算的梯度；Nesterov动量（Nesterov Momentum）：Momentum的改进
- Adagrad、RMSProp、Adadelta、Adam：这些方法都是训练过程中依照规则降低学习速率，部分也综合动量
- **优化起点**：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）

2、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** 这里的解释留给非常棒的Computational Graphs: Backpropagation 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。**要结合图解阅读以下内容。但手机显示的图过小，最好用电脑打开。**

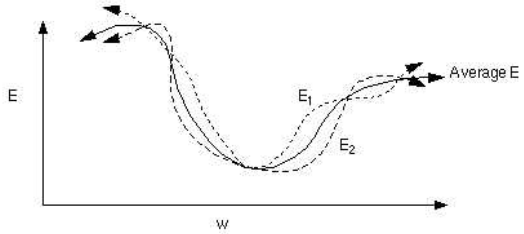


- **收集训练集（train data）**：也就是同时有input以及对应label的数据。每个数据叫做训练样本（sample）。label也叫target，也是机器学习中最贵的部分。上图表示的是我的数据库。假设input本别是x的维度是39，label的维度是48。

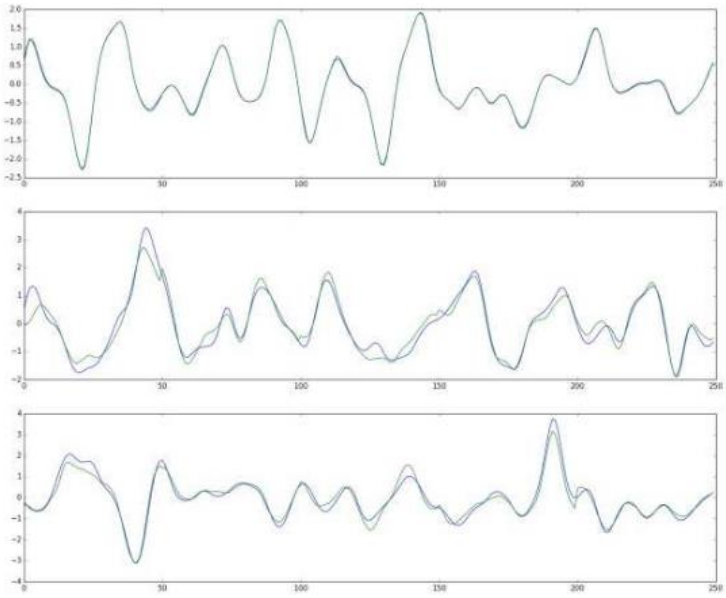
- **设计网络结构（architecture）**：确定层数、每一隐藏层的节点数和激活函数，以及输出层的激活函数和损失函数。上图用的是两层隐藏层（最后一层是输出层）。隐藏层所用激活函数a( )是ReLu，输出层的激活函数是线性linear（也可看成是没有激活函数）。隐藏层都是1000节点。损失函数L( )是用于比较距离MSE：mean((output - target)^2)。MSE越小表示预测效果越好。训练过程就是不断减小MSE的过程。到此所有数据的维度都已确定：
  - 训练数据：$input \in R^{39}; label \in R^{48}$
  - 权重矩阵：$W_{h1} \in R^{1000x39}; W_{h2} \in R^{1000x1000}; W_o \in R^{48x1000}$
  - 偏移向量：$b_{h1} \in R^{1000}; b_{h2} \in R^{1000}; b_o \in R^{48}$
  - 网络输出：$output \in R^{48}$

- **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（normalization）、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。上图用均匀分布初始权重（Uniform distribution）。
- **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。传统的方法Gradient Descent是正向传递所有样本来计算梯度。SGD的方法来计算梯度的话，loss function的形状如下图所示会有变化，这样在更新中就有可能“跳出”局部最小值。



  - 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。输出数据是48维，这里只取1个维度来画图。蓝色的是实际值，绿色的是实际值。最上方的是训练数据的对比图，而下方的两行是神经网络模型**从未见过**的数据预测对比图。（不过这里用的是RNN，主要是为了让大家感受一下效果）



注：此部分内容**不是**这篇文章的**重点**，但为了理解**深层**神经网络，需要明白最基本的训练过程。 若能理解训练过程是通过梯度下降尽可能缩小loss的过程即可。 若有理解障碍，可以用python实践一下从零开始训练一个神经网络，体会整个训练过程。若有时间则可以再体会一下计算图自动求梯度的方便利用TensorFlow。

结合Tensorflow playground理解5种空间操作和物质组成视角

打开网页后，总体来说，蓝色代表正值，黄色代表负值。拿**分类**任务来分析。

- 数据：在二维平面内，若干点被标记成了两种颜色。黄色，蓝色，表示想要区分的两类。你可以把平面内的任意点标记成任意颜色。网页给你提供了4种规律。神经网络会根据你给的数据训练，再分类相同规律的点。

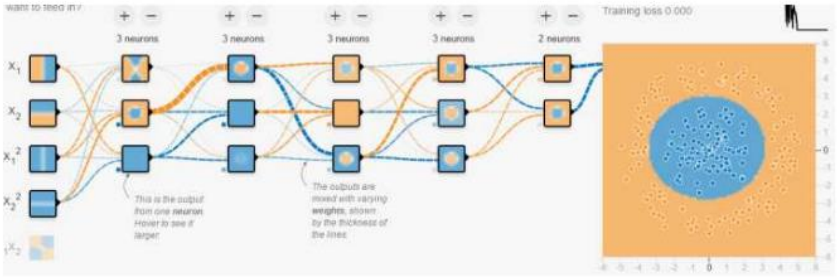
- 输入：在二维平面内，你想给网络多少关于“点”的信息。从颜色就可以看出来，x_1 左边是负，右边是正，x_1 表示此点的横坐标值。同理，x_2 表示此点的纵坐标值。x_1^2 是关于横坐标值的“抛物线”信息。你也可以给更多关于这个点的信息。给的越多，越容易被分开。

- 连接线：表示权重，蓝色表示用神经元的原始输出，黄色表示用负输出。深浅表示权重的绝对值大小。鼠标放在线上可以看到具体值。也可以更改。在（1）中，当把 x_2 输出的一个权重改为-1时，x_2 的形状直接倒置了。不过还需要考虑激活函数。（1）中用的是linear。在（2）中，当换成sigmoid时，你会发现没有黄色区域了。因为sigmoid的值域是(0,1)

（1）

（2）

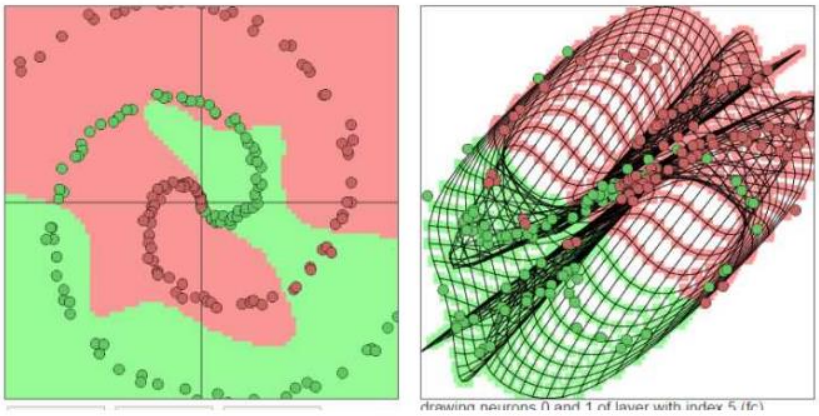
- 输出：黄色背景颜色都被归为黄点类，蓝色背景颜色都被归为蓝点类。深浅表示可能性的强弱。



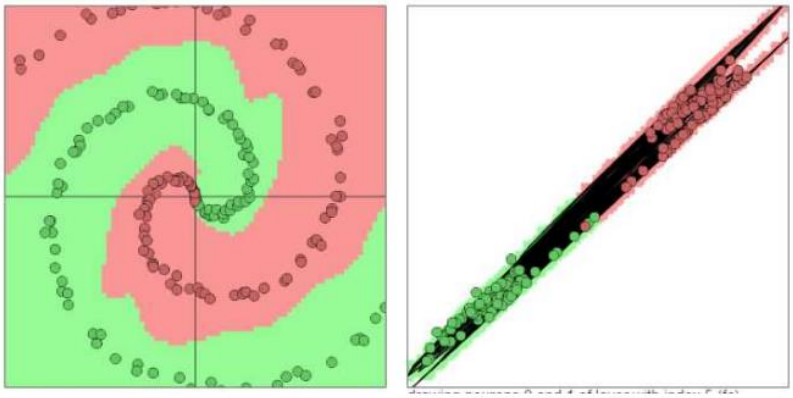
上图中所有在黄色背景颜色的点都会被分类为“黄点 ”，同理，蓝色区域被分成蓝点。在上面的分类分布图中你可以看到每一层通过上一层信息的组合所形成的。权重（那些连接线）控制了“如何组合”。神经网络的学习也就是从数据中学习那些权重。Tensorflow playground所表现出来的现象就是“在我文章里所写的“物质组成思想”，这也是为什么我把Tensorflow playground放在了那一部分。

不过你要是把Tensorflow的个名字拆开来看的话，是tensor（张量）的flow（流动）。Tensorflow playground的作者想要阐述的侧重点是“**张量如何流动**”的。

5种空间变换的理解：Tensorflow playground下没有体现5种空间变换的理解。需要打开这个网站尝试：ConvNetJS demo: Classify toy 2D data



左侧是原始输入空间下的分类图，右侧是转换后的高维空间下的扭曲图。



最终的扭曲效果是所有绿点都被扭曲到了一侧，而所有红点都被扭曲到了另一侧。这样就可以线性分割（用超平面（这里是一个平面）在中间分开两类）

四、表现原因

文章的最后稍微提一下深层神经网络。深层神经网络就是拥有更多层数的神经网络。

按照上文在理解视角中所述的观点，可以想出下面两条理由关于为什么更深的网络会更加容易识别，增加容纳变异体（variation）（红苹果、绿苹果）的能力、鲁棒性（robust）。

数学视角：变异体（variation）很多的分类的任务需要高度非线性的分割曲线。不断的利用那5种空间变换操作将原始输入空间像“捏橡皮泥一样”在高维空间下捏成更为线性可分/稀疏的形状。

物理视角：通过对“**抽象概念**”的判断来识别物体，而非细节。比如对“飞机”的判断，即便人类自己也无法用语言或者若干条规则来解释自己如何判断一个飞机。因为人脑中真正判断的不是是否“有机翼”、“能飞行”等细节现象，而是一个抽象概念。层数越深，这种概念就越抽象，所能**涵盖的变异体**就越多，就可以容纳战斗机，客机等很多种不同种类的飞机。