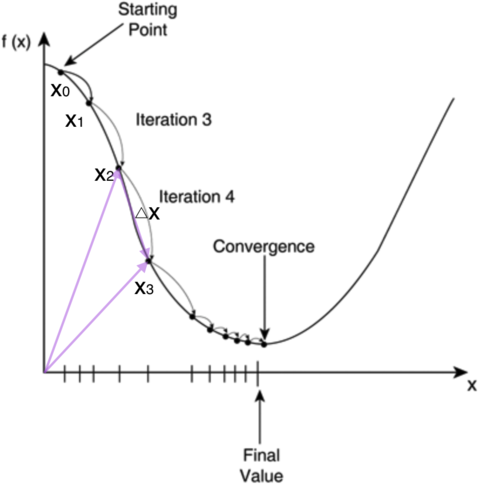
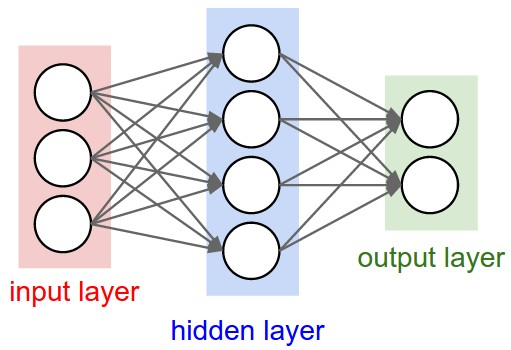
开始接触深度学习，首先应对于感知器有一个初步的了解。感知器代表了神经网络中的基本单元——神经元。我们单独使用一个感知器的时候，用到的激活函数是阶跃函数，而神经元在神经网络中一般使用sigmoid函数或tanh函数来作为激活函数。那么要实现感知器，我们其实就需要获得感知器的权重项ω和偏置项b，这个获取的过程我们称之为训练。通过一定的算法来使感知器可以实现目标函数。每次从训练数据中取出一个样本的输入向量x，使用感知器计算其输出y，再根据感知器规则来调整权重。每处理一次样本就调整一次权重。进过多轮迭代后，就可以训练出感知器的权重。

感知器有一个问题，当面对的数据集不是线性可分的时候，“感知器规则”可能无法收敛，这意味着我们永远也无法完成一个感知器的训练。为了解决这个问题，我们使用一个可导的线性函数来替代感知器的阶跃函数，这种感知器就叫做线性单元。线性单元在面对线性不可分的数据集时，会收敛到一个最佳的近似上。重要的是，线性单元是对于某个事物的基本特征做一个概括，用不同的权值来拟合各项特征。这其实是一个非常大的进步，有时候实现算法没有那么难，关键是如何确定事物的特征，这一点在之后的CNN，RNNs中尤为重要。实现了结构，我们就要来确定如何训练。在机器学习中，一般有两类学习方法，第一种叫监督学习，就是我的样本里输入输出都存在，从输入和输出中寻找规律，只要样本量足够大，它就能找到一些规律来预测答案了。很多时候，样本只有输入没有输出，这种情况下使用第二种学习方法，无监督学习，这样模型可以总结出输入的一些规律，但无法知道其对应的答案。通过数学推导可知，线性单元的目标函数其实是一个关于参数 ω的函数，就是求取合适的 ω使目标函数取最小值。这在数学上称为优化问题。那么，做这个优化问题的方法，就是所谓的梯度下降优化算法。



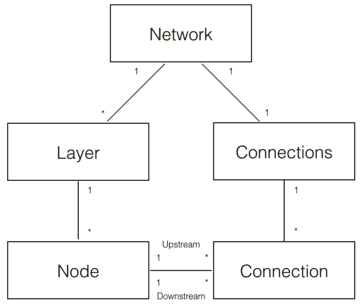
根据高数知识，梯度下降的方向指向函数值下降最快的方向，只要合理地设计步长，就可以一步一步地接近我们要的函数最小值点。



神经网络其实就是按照一定规则连接起来的多个神经元。上图展示了一个全连接(full connected, FC)神经网络，通过观察上面的图，我们可以发现它的规则包括：

* 神经元按照层来布局。最左边的层叫做输入层，负责接收输入数据；最右边的层叫输出层，我们可以从这层获取神经网络输出数据。输入层和输出层之间的层叫做隐藏层，因为它们对于外部来说是不可见的。
* 同一层的神经元之间没有连接。
* 第N层的每个神经元和第N-1层的所有神经元相连(这就是full connected的含义)，第N-1层神经元的输出就是第N层神经元的输入。
* 每个连接都有一个权值。

要训练神经网络，我们采用反向传播算法。要在代码中实现，我们可以把神经网络拆分成几个对象



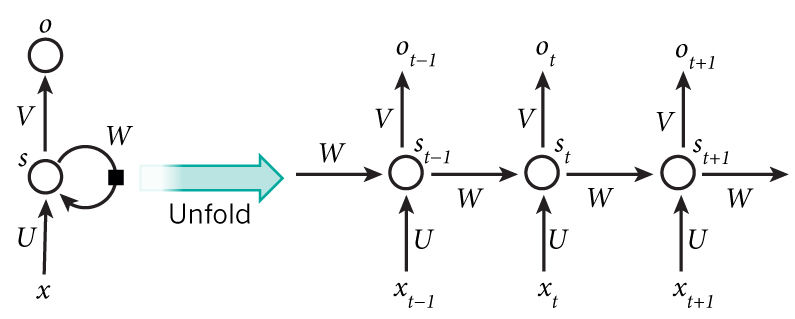
如上图，可以分解出5个领域对象来实现神经网络：

* *Network* 神经网络对象，提供API接口。它由若干层对象组成以及连接对象组成。
* *Layer* 层对象，由多个节点组成。
* *Node* 节点对象计算和记录节点自身的信息(比如输出值、误差项等)，以及与这个节点相关的上下游的连接。
* *Connection* 每个连接对象都要记录该连接的权重。
* *Connections* 仅仅作为Connection的集合对象，提供一些集合操作。

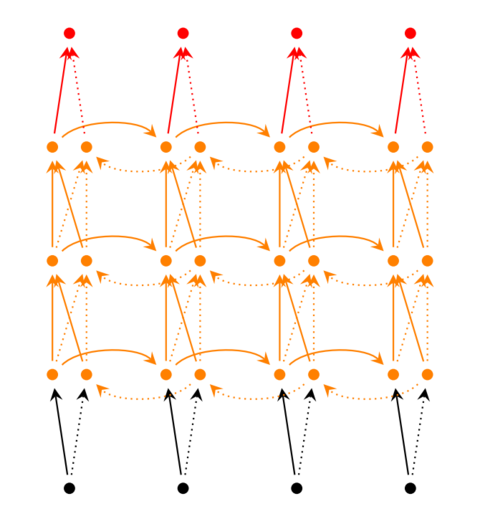
当然还可以用向量化编程，那样更为简洁。

RNN，这是一个比较宽泛的概念，最早RNN是在自然语言处理领域被用起来的，也就是用RNN来为语言模型建模。语言模型可以在前言的基础上预测后语，要实现这样的功能，就不能独立地分析前言里的某一个词汇，而应该分析这些词连起来的整个语句，这就要用到循环神经网络RNN。

最基本的循环神经网络图为下图左侧所示，仅有输入层、一个隐藏层和一个输出层组成。如果我们把左边的图展开，循环神经网络也可以画成右边这个样子。



深度循环神经网络



如图所示通过堆叠隐藏层，我们可以得到深度循环神经网络，而且它实现了双向循环，值得一提的是，在双向计算中，正向计算和反向计算不共享权重。

关于RNN的训练，我们采用BPTT算法，首先我们应该了解什么是BP算法。

BP算法

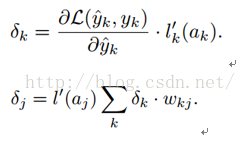
（1）BP算法通过链式法则计算出每一层参数的梯度，使用梯度下降的算法进行损失函数的优化；

（2）由于损失函数是非凸的，所以梯度下降算法并不能保证得到最优解，然而在实际应用中，使用随机梯度下降（stochastic gradient descent，SGD）算法可以得到很好的效果；这里的随机指的是每次进行参数迭代时，随机选取部分样本（也就是一个batch）进行优化计算；

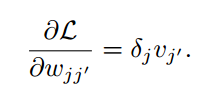
917132141411.png

Fi是单个样本（xi，yi）的损失函数，参数表示学习率；

（3）BP算法中梯度的计算是基于残差进行计算的，残差的公式如下：



（4）有了残差，就可以计算出梯度了：



PS：l 表示激活函数，v表示激活值；

RNN可以按时间展开，从而进行前向以及后向计算，从而可以将传统的BP算法应用于RNN模型的训练，由于是按照时间展开的BP算法，所以术语就叫做BPTT（backpropagation through time）；然而用BPTT训练RNN具有一个致命的缺陷，就是梯度弥散问题，下面为了分析简便，只考虑一个输入节点，一个隐藏节点以及一个输出节点的网络，如下图1，按时间展开为图2：

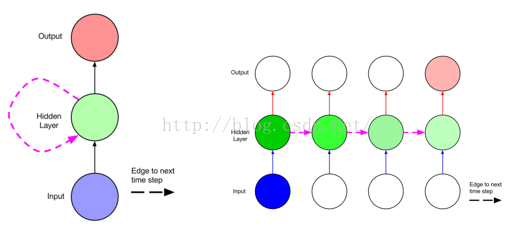


               图1                                                                       图2

假设只考虑从时间节点t1到时间节点t2之间的序列，在节点t1给一个输入，t1与t2之间的节点输入都为0；那么节点t1的输入传到了节点t2时的输出随着t2-t1的增大而指数下降；与此同时，下降的速度也与时间点之间的传输权重以及激活函数有关；因此RNN并不能保持长时间的上下文信息；为了解决这个问题，我们最常用的方法就是使用其他结构的RNNs，比如LSTM和GRU。