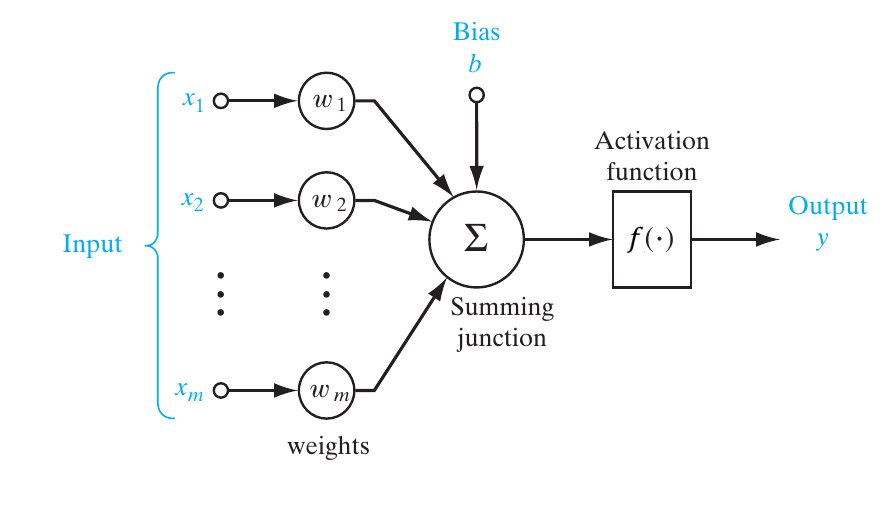
**1. 传统神经网络**

**1.1 感知器/神经元（Perceptron）**

**感知器结构如下：**



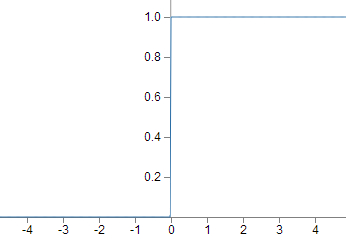
**对应公式：**

**其中，x为输入，y为输出，w为权值，b为偏置项**

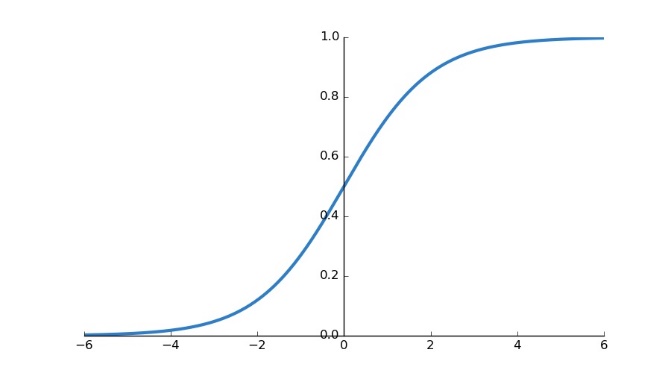
**,**

**f为激活函数（Activation Function），常用的激活函数如下：**

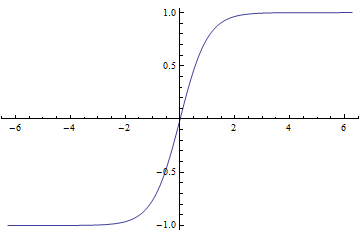
* **阶跃函数（Step Function）**



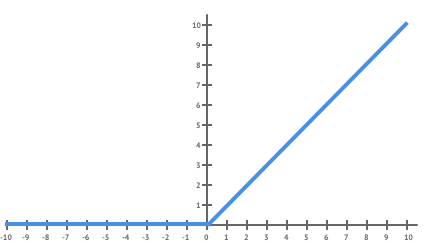
* **Sigmoid函数**



* **tanh函数**



* **ReLU函数（Rectified Linear Unit, ReLU）**



**感知器可以拟合任何的线性函数，如布尔运算等，任何线性分类/线性回归问题都可以用感知器来解决。**

**如果没有激励函数，神经网络不管叠加多少层，输出都仅仅是输入的线性组合，因此引入非线性函数作为激励函数。**

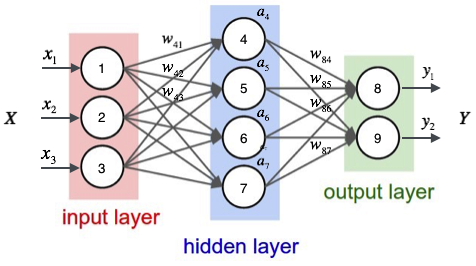
**Sigmoid或tanh作为激励函数，因为是指数运算，计算量较大，而且反向传播时，容易出现梯度消失问题，这主要是由于sigmoid或tanh接近饱和区时，导数趋于0，容易丢失信息。**

**ReLU相对计算量小很多，计算速度和收敛速度远快于sigmoid和tanh，同时ReLU是分段线性函数，更容易学习优化，会使一部分神经元的输出为0，形成网络稀疏，减少了参数的相互依存，一定程度缓解过拟合，ReLU分段线性存在的问题是可能导致某些神经元永远不会被激活，相应的参数也永远不被更新。**

**1.2 神经网络（Neural Network, NN）**

**输入向量的维度和输入层神经元个数相同**

**输出向量的维度和输出层神经元个数相同**



**上图网络对应公式如下：**

**输入层：**

**隐藏层：**

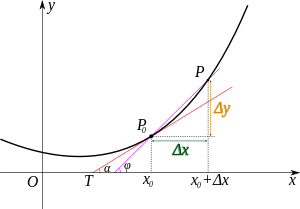
**输出层：**

**矩阵表示为：**

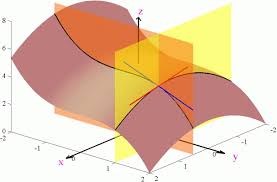
**1.3 反向传播算法（Back Propagation, BP）**

**1.3.1 梯度**

**导数（Derivative）：**



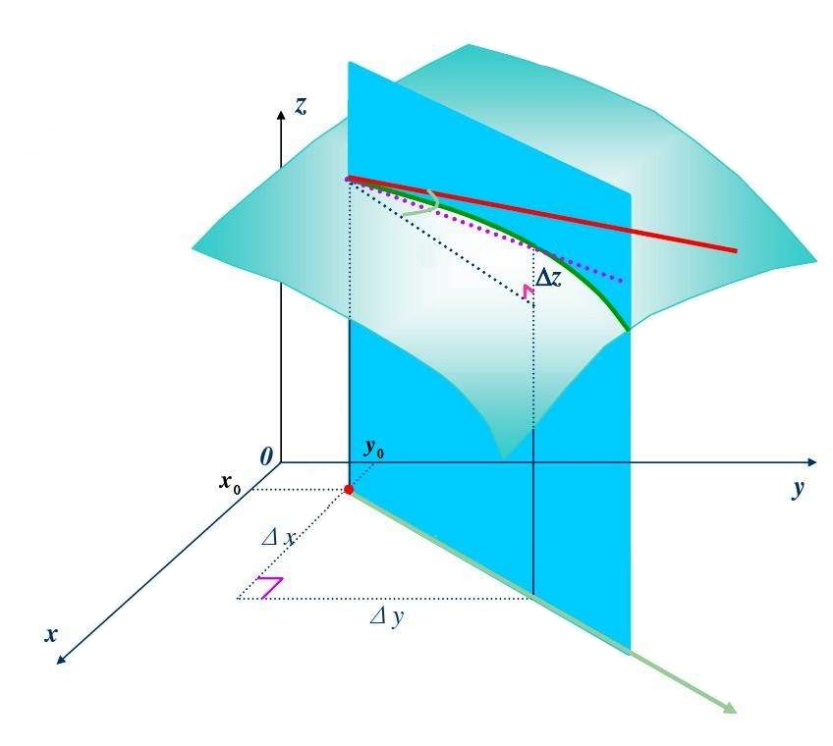
**偏导数（**Partial Derivative**）：**



**方向导数（Directional Derivative）：**

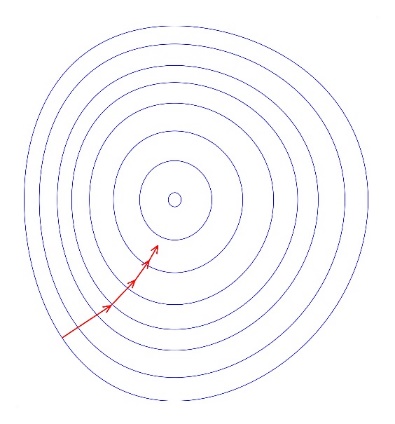
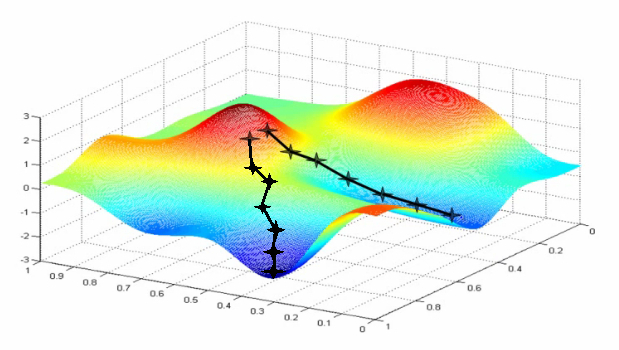
**若函数具有一阶连续偏导数，计算在点沿单位向量 的方向导数，为向量u与坐标轴x, y的方向角**

**一般的**



**梯度（Gradient）：**

**梯度是一个矢量，表示函数在某一点处，沿着该方向增长最快，梯度大小也即为该点上最大的方向导数。梯度亦是等值线的垂线方向。**



**1.3.2 损失函数（Loss Function）**

**损失函数是用来估量模型的预测值**y与真实值t的不一定程度，其中n为样本数量

* 0-1损失函数
* 绝对损失函数（Absolute Loss Function）
* 平方损失函数（Quadratic Loss Function）

计算方便，欧式距离是一种很好的相似性度量标准

* 交叉熵损失函数（Cross Entropy）

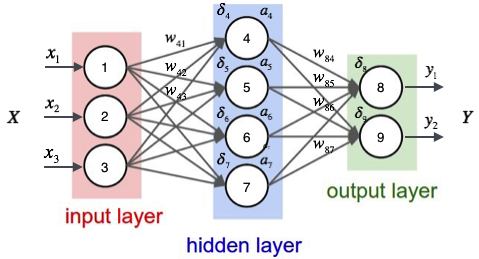
**Sigmoid激活函数求导时，在两端区域导数趋于零，变化平缓，导致学习速度下降。交叉熵损失函数可以有效解决学习速度下降问题，当网络为二分类0-1，且实际值t和预测值y较接近时，交叉熵损失函数将趋于零。交叉熵损失函数具有非负性，即。**

* **极大似然损失函数（log-likelihood）**

**表示最后一层为Softmax层时，预测类别为y的概率，，当预测准确，趋于1，趋近于0，当预测不准确，趋于0，趋近于正无穷大。log-likelihood损失函数常与Softmax层组合，亦可有效解决学习速度下降问题。**

神经网络的训练过程即为，在训练集的输入x，输出y确定的情况下，修改网络权值w和b，使得实际值t和网络的输出值y之间的差异尽可能小，即求解损失函数的最小值点。

梯度的相反方向即为Loss下降最快的方向，训练时，将权重和偏置项初始化为0，然后利用损失函数的梯度迭代来修改w和b，直到训练完成，其中，为学习率。



**若激活函数用Sigmoid，损失函数用平方损失函数，其中**表示节点i的输入，表示节点i的输出，表示节点i的误差项，即输入sigmoid的误差。

对于输出层节点：

对于隐藏层节点：

**若激活函数用Sigmoid，损失函数用交叉熵损失函数，其中**表示节点i的输入，表示节点i的输出，表示节点i的误差项，即输入sigmoid的误差。由于交叉熵损失函数的偏导相比于平方损失函数，少了这一项，即sigmoid激活函数的导数的影响，因此可以解决sigmoid函数的梯度消失学习减速问题。仅有输出层的sigmoid偏导会被消除，对于隐藏层的偏导，sigmoid仍然会产生影响。

对于输出层节点：

**1.3.3 正则化（Regularization）**

**降低过拟合的主要方法有：增加训练数据集，降低模型复杂度。但复杂的模型具有更好的表达能力。正则项可以描述模型的复杂程度。在损失函数中加入正则项可有效降低过拟合，训练的过程即尽量使损失loss减少的同时不能让正则项变大。**

* **L1正则（L1 Regularization）**

为网络的权重w之和，为正则化参数，调整正则项的相对重要程度，为本次迭代反向求导所用训练样本batch个数。

符号函数，注意，在时不可导，需要事先令

* **L2正则（L2 Regularization）**

正则项偏向于让网络学习比较小的权重w，正则项通常不包括偏置b，因此对偏置b没有影响。

神经网络中，对于较小的权重，输入数据x的变化不会对网络造成太大影响，因此降低了数据噪声的影响。也防止了过拟合。正则化的网络通常要比未正则化的网络泛化能力更好。

L1正则学习的模型参数非常稀疏，即存在较多的参数w为0，这使得模型的预测输出仅和少量的输入特征相关，即仅部分输入对结果起到决定性作用，符合一些现实情况，L1可用来做特征选择。

L2正则会使得较多的参数w趋于0，并且加入L2正则使得模型求最优解变得稳定快速。神经网络中多用L2。

**1.3.4 Softmax层**

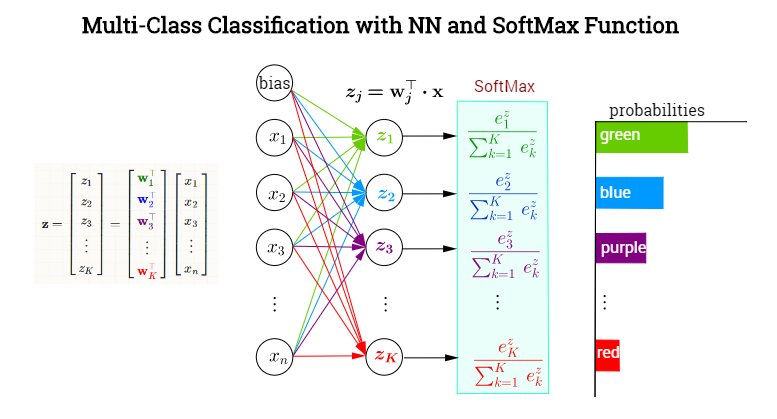
**将Softmax函数作为网络的输出层激活函数。**

其中i表示输出层节点i，k为输出层所有节点。

Softmax函数输出的取值范围为(0, 1)，并且所有输出的和为1，因此常常将Softmax层的输出看作一个概率分布。即将输出的值看作是神经网络认为分类结果是i的概率。

**若最后一层选择Softmax，采用log-likelihood损失函数，其中**表示节点i的输入，表示节点i的输出，表示节点i的误差项。

对于输出层节点：



**1.3.5 梯度检查（Gradient Checking）**

**梯度更新公式**

，

关键在于对w,b的偏导要正确，导数亦可依据极限算出，有

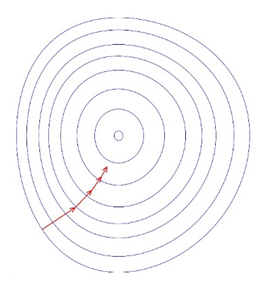
**若两者差别非常小，说明反向求导正确**

**1.3.6 随机梯度下降（Stochastic Gradient Descent, SGD）**

**随机梯度下降，如果目标函数是凸函数，梯度下降可以得到全局最优解，对于非凸函数，容易得到局部最优解**

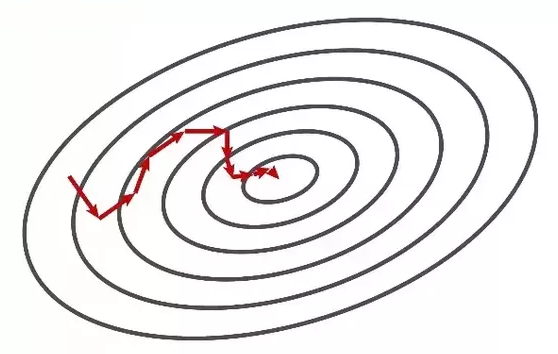
* **批量梯度下降（BGD）**

**对于w，b的更新，需要计算所有样本的损失和，再计算得到梯度， w, b一次更新的幅度比较大。对于大规模样本效率低下**

****

* **随机梯度下降（SGD）**

**每次随机从训练集中选取一个样本，并用这个样本计算梯度来更新w, b。虽然不是每次迭代得到的梯度都向着全局最优方向，但大体的方向是朝向全局最优，最终的解往往是在全局最优解附近，适用于大规模训练样本情况。**



* **小批量梯度下降（Mini-batch SGD）**

**每次从训练集中选取小部分（mini-batch）进行训练，并计算梯度更新w, b。是批量梯度下降和随机梯度下降的折中。**



* **RMSprop**
* **Adam**
* **Momentum**

其中，为上一次更新时的梯度，为动量因子

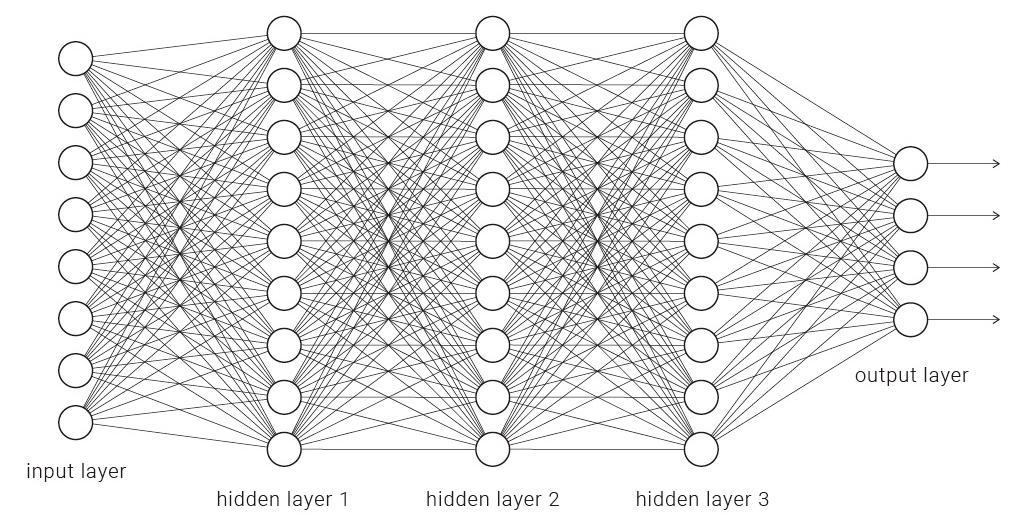
Momentum模拟物理中动量的概念，对于本次梯度更新，一定程度保留之前更新的方向，在一定程度上保证梯度更新的稳定性，并具有一定程度拜托局部最优的能力。若本次计算梯度与历史梯度方向一致，则可加速梯度更新的速度。在梯度下降陷入局部极小值来回震荡的时候，梯度趋于零，使得更新幅度增大，跳出局部极小值。在计算梯度与历史梯度不一致时，momentum项能够减缓更新幅度。

* **Nesterov**

**实践表明，Nesterov往往比Momentum效果好。**

* **Adagrad**
* **Adadelta**

**1.4 前馈神经网络**



**2. 卷积神经网络（Convolutional Neural Network, CNN）**

**2.1** **卷积层（Convolution Layer）**

**线性处理，提取特征**

**2.2** **Batch Normalization Layer**

**训练过程中，对于网络某神经元节点，若输入的分布不断发生变化，则网络会调整参数，以适应新的分布，并且这样的调整往往波动较大，而使得学习速率下降，为使各batch在该神经元节点具有相似分布，加入Batch Normalization层对数据做归一化，相当于对数据做一个约束，可以加快网络训练收敛速度。**

**Batch Normalization借鉴了数据处理中的白化预处理，数据经过白化预处理后满足两个条件：1. 特征之间的相关性降低，2. 数据均值/标准差归一化，即使得特征的均值为0，标准差为1。为了简化计算Batch Normalization仅处理数据使其满足第2条。同时，由于归一化会破坏特征分布，因此，引入变换重构。**

**Batch Normalization公式如下：**

**,**

**其中，为输入数据，m为样本个数，表示一个Batch，****为输入数据的期望，为输入数据的方差，为经过归一化处理后的数据，为变换重构操作，为重构参数，在训练过程中该参数需要被学习。当为下述值时，可恢复原始网络所要学习的特征分布。这其实是对归一化做了一个逆操作。**

**，**

**在训练阶段，根据每批训练数据进行计算，并将其保存，在预测阶段，由下式计算，即根据训练阶段保存的求整体的均值和方差。**

**,**

**这里是因为求时用的每个Batch的均值，而与整体的均值是有出入的，这导致所有Batch方差的平均会比总体的方差略小，因此加入将方差放大。这种称为方差的无偏估计。**

**预测阶段，Batch Normalization公式如下：**

**对于网络某一层，Batch Normalization归一化该层的输入数据，提高网络泛化能力。放松了网络对于学习率，学习速率衰减等的初始化条件，在CNN网络中Batch Normalization一般加到卷积层和ReLU层中间。**

**2.3 激活函数层（ReLU）**

**激活函数，引入非线性，使网络可以学习非线性分类**

**2.4 池化层（Pooling Layer）**

**缩小图像，降维，较少数据量**

**2.5 Concat Layer**

**拼接多个输入**

**2.6 全连接（Full Connect Layer）**

**参数较多，一般放到网络的最后，现在被大多数主流网络所舍弃**

**2.7 Dropout Layer**

**为防止过拟合及使网络更加具有鲁棒性，在训练过程中丢弃一些节点。**

**2.8 调参**

**选择损失函数**

**选择梯度下降方法**

**设置学习率，学习率衰减系数**

**参数初始化**

**Dropout比例**

**2.9 常见CNN网络**

**AlexNet**

**VGG-16**

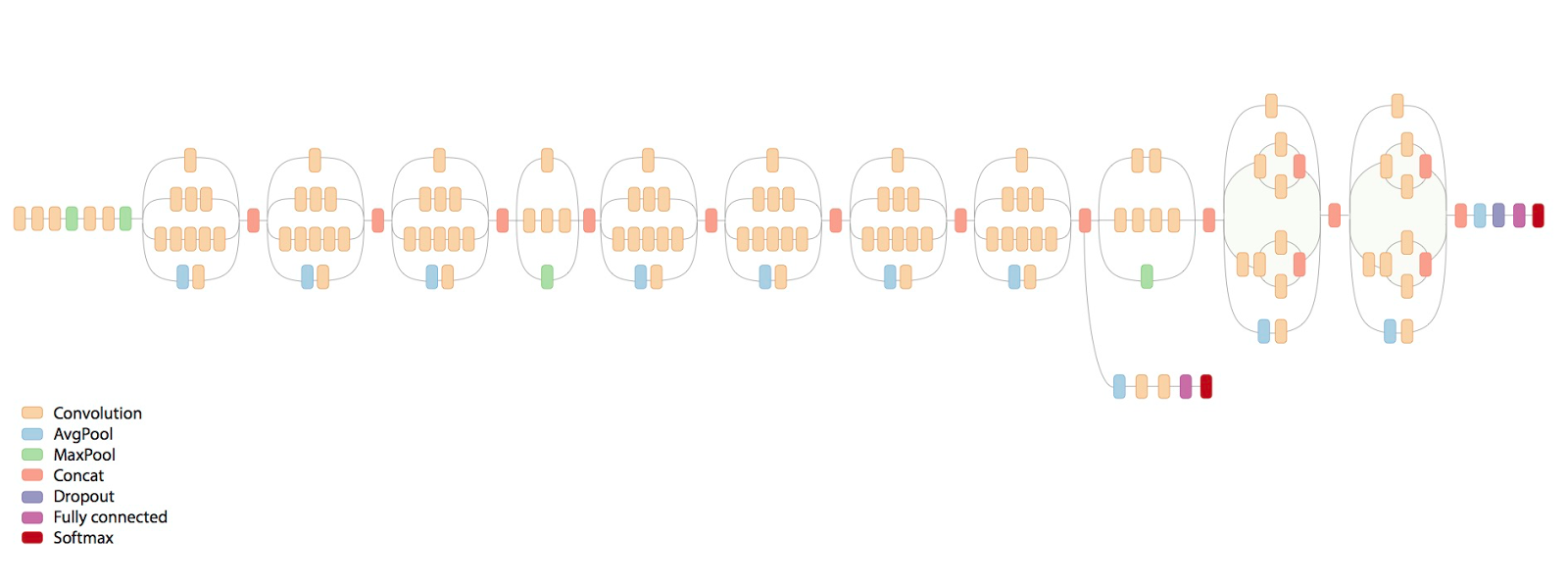
**VGG-19**

**Google Inception v1**

**Google Inception v2**

**Google Inception v3**

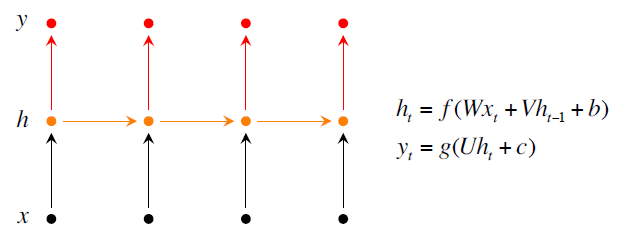
**Google Inception v4**

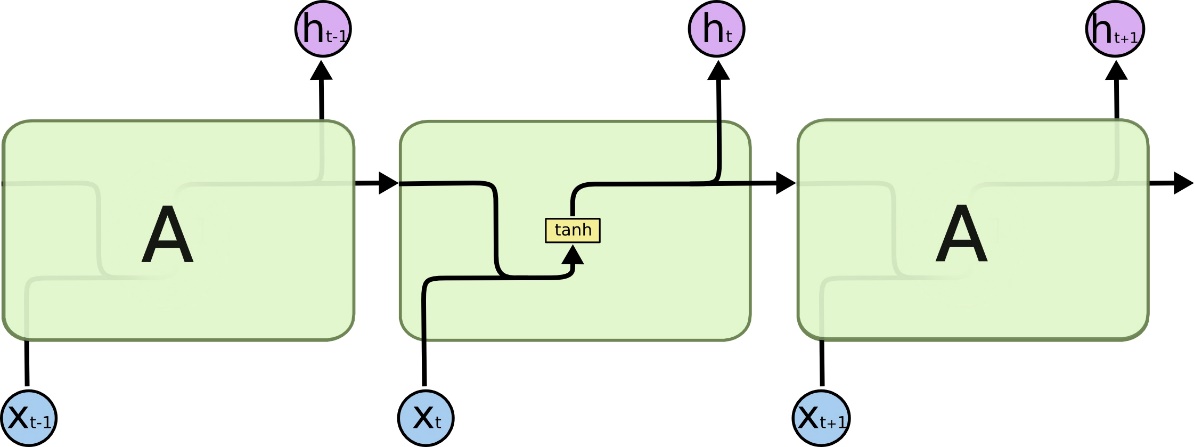


**3. 循环神经网络（Recurrent Neural Network, RNN）**

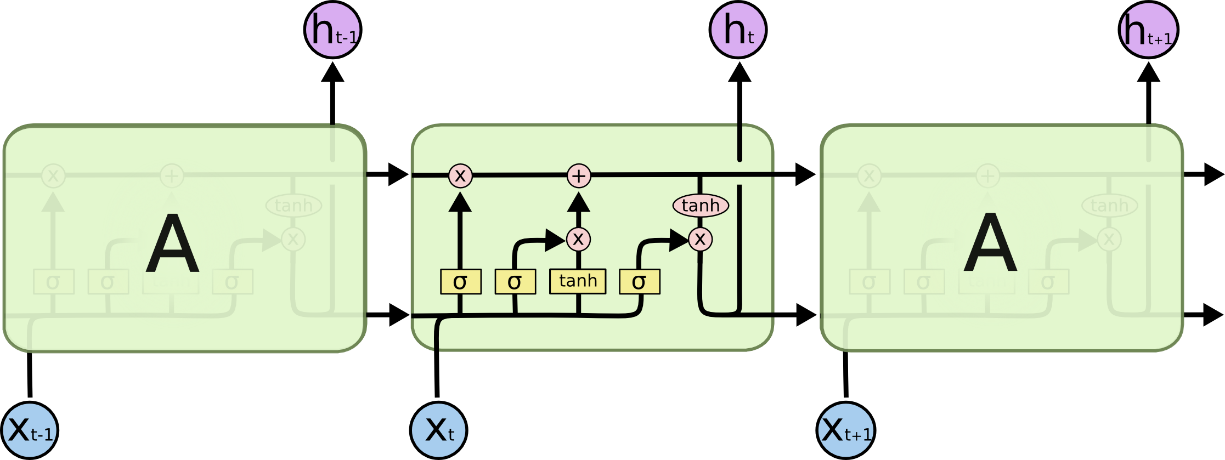
**3.1 循环神经网络（Recurrent Neural Network, RNN）**

**点**





**3.2 长短时记忆网络（Long-short Term Memory, LSTM）**



**LSTM相对于RNN可以很好解决梯度消失问题**

**梯度消失是因为反向求导时导数趋于0**

**梯度爆炸是因为有奇异点**

**3.3 门限递归单元（Gated Recurrent Unit, GRU）**

**可以**