**神经网络对比**

**DNN：**

1. **应用场景：全连接的前馈深度神经网络(Fully Connected Feed Forward Neural Networks)，也就是DNN适用于大部分分类(Classification)任务，比如数字识别等。**但一般的现实场景中我们很少有那么大的数据量来支持DNN，所以纯粹的全连接网络应用性并不是很强。

**CNN：**

1）**应用场景：**虽然我们一般都把CNN和图片联系在一起，但事实上CNN可以处理大部分格状结构化数据(Grid-like Data)。举个例子，**图片**的像素是二维的格状数据，时间序列在等时间上抽取相当于一维的的格状数据，而**视频数据**可以理解为对应视频帧宽度、高度、时间的三维数据。

**RNN：**

1. **应用场景：语音分析，文字分析，时间序列分析。主要的重点就是数据之间存在前后依赖关系，有序列关系。**一般首选LSTM，如果预测对象同时取决于过去和未来，可以选择双向结构，如双向LSTM。

**自编码器(Auto-encoder)：**

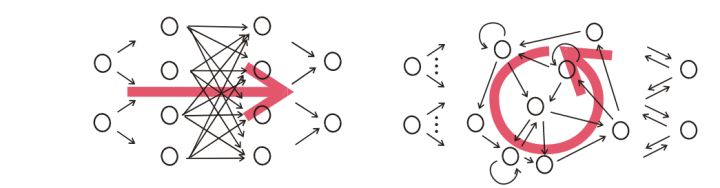
1. **应用场景：：主要用于降维(Dimension Reduction)，这点和PCA比较类似。同时也有专门用于去除噪音还原原始数据的去噪编码器(Denoising Auto-encoder)**。

**GCN：**

1）**应用场景：社交网络非常适合用图数据来表达，电商场景中的推荐知识图谱**。

**从神经元的角度来讲解。MLP是最朴素的DNN，CNNs是encode了空间相关性的DNN，RNNs是encode进了时间相关性的DNN。**

**在序列信号的应用上，CNN是只响应预先设定的信号长度（输入向量的长度），RNN的响应长度是学习出来的。**那么CNN似乎也有类似的功能？ 那么CNN是不是也可以当做RNN来用呢? 答案是否定的，RNN的重要特性是可以处理不定长的输入，得到一定的输出。当你的输入可长可短， 比如训练翻译模型的时候， 你的句子长度都不固定，你是无法像一个训练固定像素的图像那样用CNN搞定的。而利用RNN的循环特性可以轻松搞定。



**CNN对特征的响应是线性的，RNN在这个递进方向上是非线性响应的。这也带来了很大的差别。**

**CNN vs DNN**

**共性：**每个hidden neuron对应的weights|filter都在提炼学习特征，越深层filter提炼越抽象复杂的特征

**不同点：**

**CNN重视空间信息的保留（DNN，否）**

**CNN眼里只有局部小特征（DNN，只有全局大特征）**

**CNN：理解图片，是一步一步，一块一块，比对小特征的过程（DNN：全局性大特征比对）**

卷积神经网络与普通神经网络的区别在于，卷积神经网络包含了一个由卷积层和子采样层构成的特征抽取器。在卷积神经网络的卷积层中，一个神经元只与部分邻层神经元连接。在CNN的一个卷积层中，通常包含若干个特征平面(featureMap)，每个特征平面由一些矩形排列的的神经元组成，同一特征平面的神经元共享权值，这里共享的权值就是卷积核。卷积核一般以随机小数矩阵的形式初始化，在网络的训练过程中卷积核将学习得到合理的权值。共享权值（卷积核）带来的直接好处是减少网络各层之间的连接，同时又降低了过拟合的风险。子采样也叫做池化（pooling），通常有均值子采样（mean pooling）和最大值子采样（max pooling）两种形式。子采样可以看作一种特殊的卷积过程。卷积和子采样大大简化了模型复杂度，减少了模型的参数。

**DNN vs GCN**

如果只考虑节点分类任务，将图卷积的本质理解为特征传播即可。相比于一般的DNN，GCN只是每层在节点特征上做了一次变换，其他并没有什么不一样。

GCN只是在特征上做了一个变换，而这个变换的实质就是特征通过拓扑结构进行了传播。每个节点的特征不再是自身的特征，而是自身和其邻居节点的特征加权求和。

**DNN vs RNN**

在深度学习领域，传统的多层感知机为基础的上述各网络结构具有出色的表现，取得了许多成功，它曾在许多不同的任务上——包括手写数字识别和目标分类上创造了记录。但是，他们也存在一定的问题，上述模型都无法分析输入信息之间的整体逻辑序列。这些信息序列富含有大量的内容，信息彼此间有着复杂的时间关联性，并且信息长度各种各样。这是以上模型所无法解决的，递归神经网络正是为了解决这种序列问题应运而生，其关键之处在于当前网络的隐藏状态会保留先前的输入信息，用来作当前网络的输出。

全连接的DNN存在着一个无法解决的问题：无法对时间序列上的变化进行建模。为了应对这种需求，业内提出了上文中提到的递归神经网络RNN。

在普通的全连接网络中，DNN的隐层只能够接受到当前时刻上一层的输入，而在RNN中，神经元的输出可以在下一时间段直接作用到本身。换句话说，就是递归神经网络它的隐层不但可以接收到上一层的输入，也可以得到上一时刻当前隐层的输入。

这一个变化的重要意义就在于使得神经网络具备了历史记忆的功能，原则上它可以看到无穷长的历史信息，这非常适合于像语音语言这种具有长时相关性的任务。

**CNN vs RNN**

1）one to one：表示的是CNN网络的场景，从固定的输入到固定的输出

2）one to many：RNN的场景，序列输出，有点像看图说话，例如固定了输入的图片，然后输出一段序列描述这个图的意义

3）many to one：RNN的场景，序列输入，比如我们做语义情感分析，输入一串不定长度的话，返回情绪

4）many to many：RNN的场景，常见的sequence to sequence，比如之前的一个文章到的，通过周杰伦的歌词数据，模仿写出一首周杰伦风格的歌词，这种场景的输入和输出的长度都是不定的。

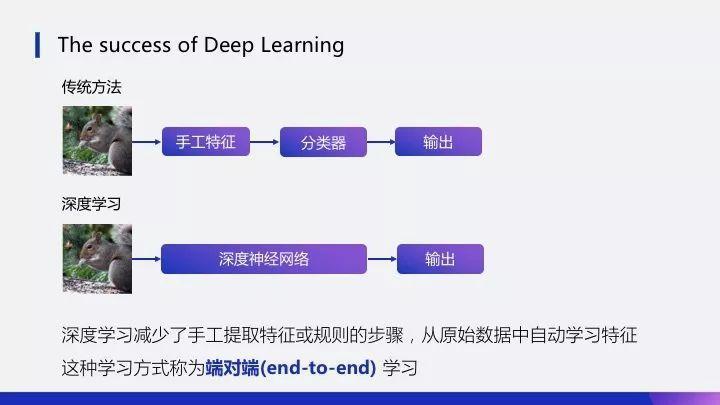
# 浅析图卷积神经网络PPT

今天想和大家分享的是图卷积神经网络。随着人工智能发展，很多人都听说过机器学习、深度学习、卷积神经网络这些概念。但图卷积神经网络，却不多人提起。那什么是图卷积神经网络呢？简单的来说就是其研究的对象是图数据（Graph），研究的模型是卷积神经网络。

**为什么有图卷积神经网络**

自2012年以来，深度学习在计算机视觉以及自然语言处理两个领域取得了巨大的成功。和传统方法相比，它好在哪里呢？

假设有一张图，要做分类，传统方法需要手动提取一些特征，比如纹理啊，颜色啊，或者一些更高级的特征。然后再把这些特征放到像随机森林等分类器，给到一个输出标签，告诉它是哪个类别。而深度学习是输入一张图，经过神经网络，直接输出一个标签。特征提取和分类一步到位，避免了手工提取特征或者人工规则，从原始数据中自动化地去提取特征，是一种端到端（end-to-end）的学习。相较于传统的方法，深度学习能够学习到更高效的特征与模式。

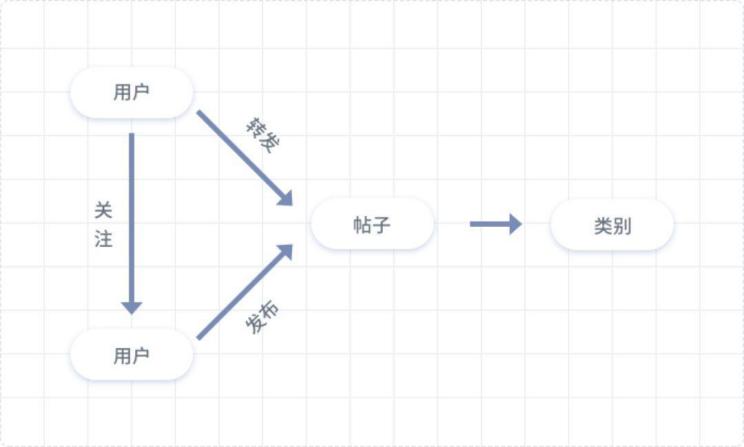


卷积神经网络很好，但是它研究的对象还是限制在Euclidean domains的数据。什么是Euclidean data？   Euclidean data最显著的特征就是有规则的空间结构，比如图片是规则的正方形栅格，比如语音是规则的一维序列。而这些数据结构能够用一维、二维的矩阵表示，卷积神经网络处理起来很高效。



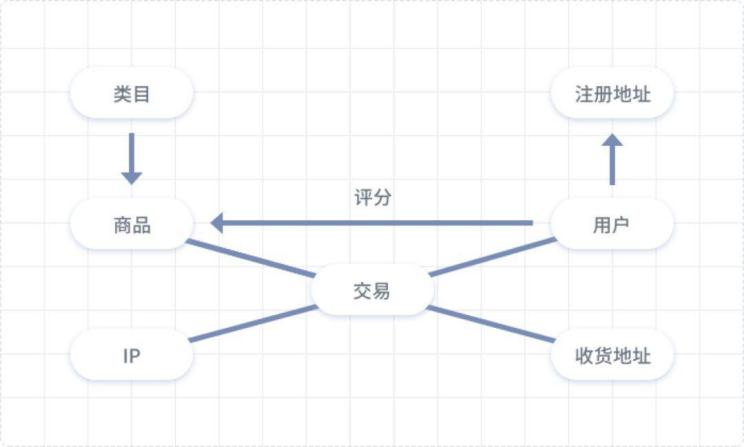
但是，我们的现实生活中有很多数据并不具备规则的空间结构，称为Non Euclidean data。比如推荐系统、电子交易、计算几何、脑信号、分子结构等抽象出的图谱。这些图谱结构每个节点连接都不尽相同，有的节点有三个连接，有的节点有两个连接，是不规则的数据结构。

下面结合两个典型的业务场景来说明什么是图：



社交网络非常适合用图数据来表达

上面的图谱刻画社交网络中各个节点以及它们之间的关系，用户A、用户B、帖子都是节点，用户与用户之间的关系是关注，用户与帖子之间的关系可能是发布或者转发。通过这样一个图谱，可以分析用户对什么人、什么事感兴趣，进一步实现推荐机制。



电商场景中的图谱

在电商中，我们首先可以想到的关键节点就是，用户、交易和商品。用户关联的节点比如会有注册地址、收获地址等；交易会关联到商品、收货地址、交易IP等、商品会关联类目等。这些节点之间的关系，比如用户除了可以通过交易购买商品，还可以对商品进行评分。这样的图数据我们可以用来做两件事情，一是推荐、二是反欺诈。

通过上面两个例子，可以很明显的感受到，图有两个基本的特性：

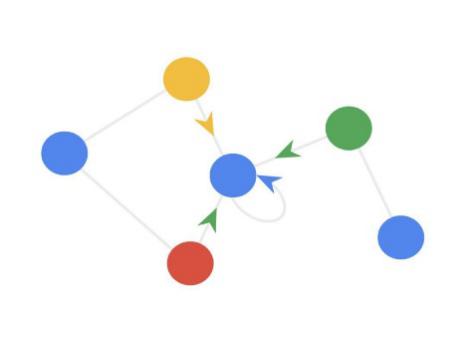
**一是每个节点都有自己的特征信息。**比如针对上图，我们建立一个风控规则，要看这个用户的注册地址、IP地址、交易的收货地址是否一样，如果这些特征信息不匹配，那么系统就会判定这个用户就存在一定的欺诈风险。这是对图节点特征信息的应用。

**二是图谱中的每个节点还具有结构信息。**如果某段时间某个IP节点连接的交易节点非常多，也就是说从某个IP节点延伸出来的边非常多，那么风控系统会判定这个IP地址存在风险。这是对图节点结构信息的应用。

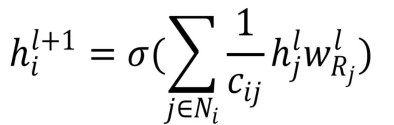
总的来说，在图数据里面，我们要同时考虑到节点的特征信息以及结构信息，如果靠手工规则来提取，必将失去很多隐蔽和复杂的模式，那么有没有一种方法能自动化地同时学到图的特征信息与结构信息呢？——图卷积神经网络

**什么是图卷积神经网络**

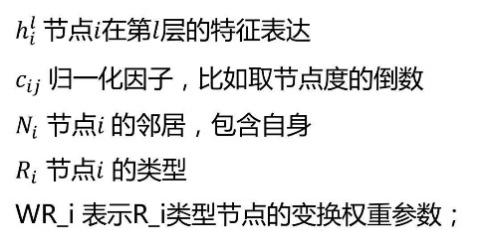
图卷积神经网络（Graph Convolutional Network）是一种能对图数据进行深度学习的方法。



**图卷积算子：**

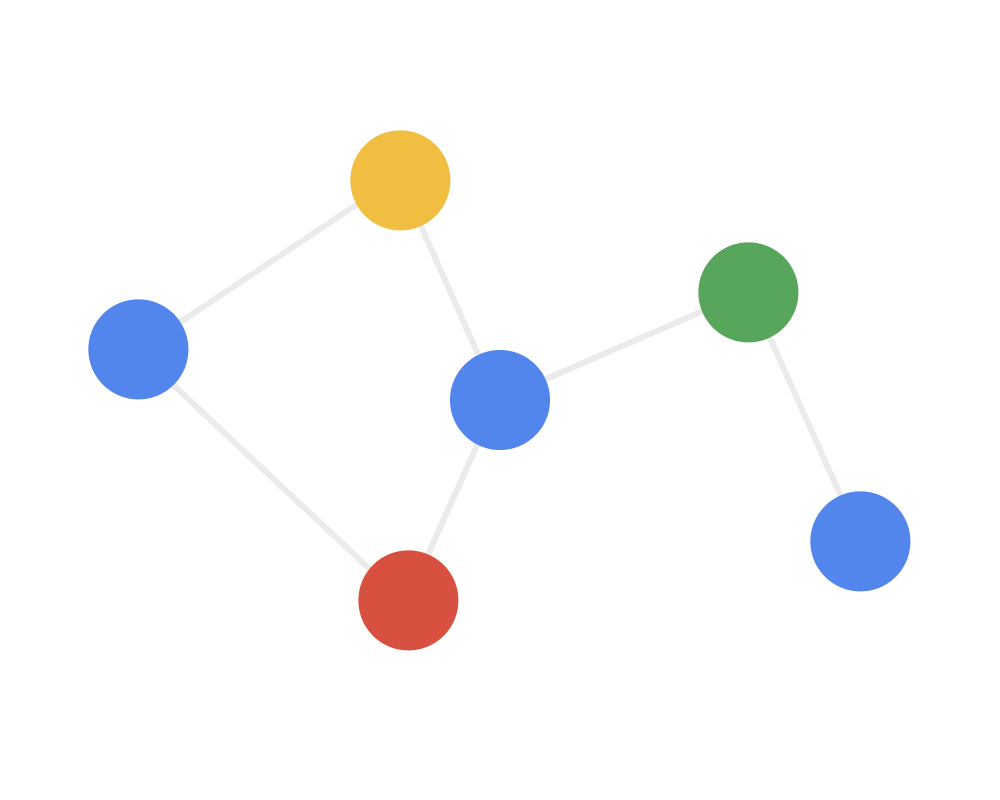


上面给出的是图卷积算子的计算公式，设中心节点为i；

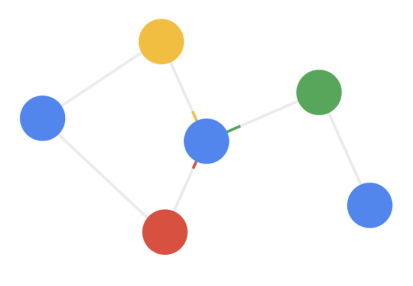


**如何理解图卷积算法？我们看动图分三步去理解（注意不同颜色代表不同的权重）：**

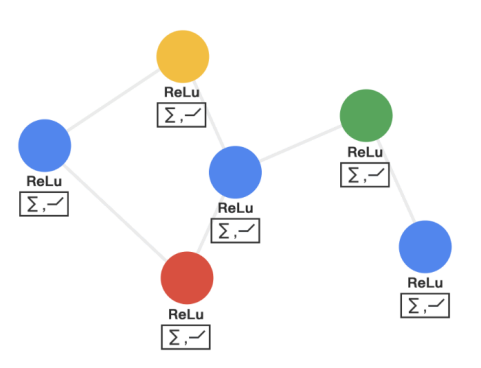
第一步：发射（send）每一个节点将自身的特征信息经过变换后发送给邻居节点。这一步是在对节点的特征信息进行抽取变换。



第二步：接收（receive）每个节点将邻居节点的特征信息聚集起来。这一步是在对节点的局部结构信息进行融合。



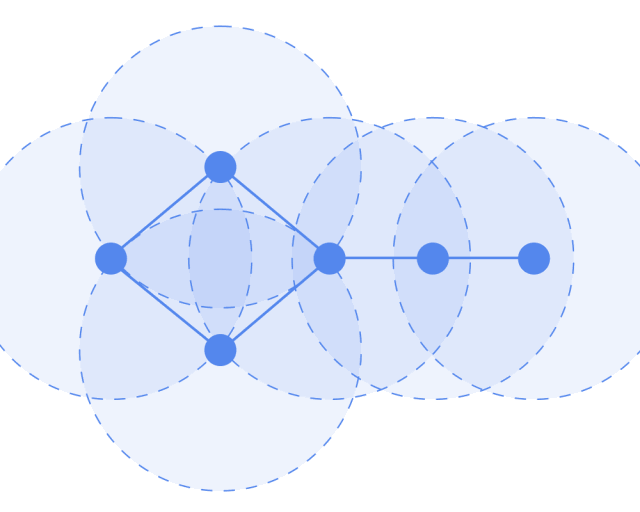
第三步：变换（transform）把前面的信息聚集之后做非线性变换，增加模型的表达能力。



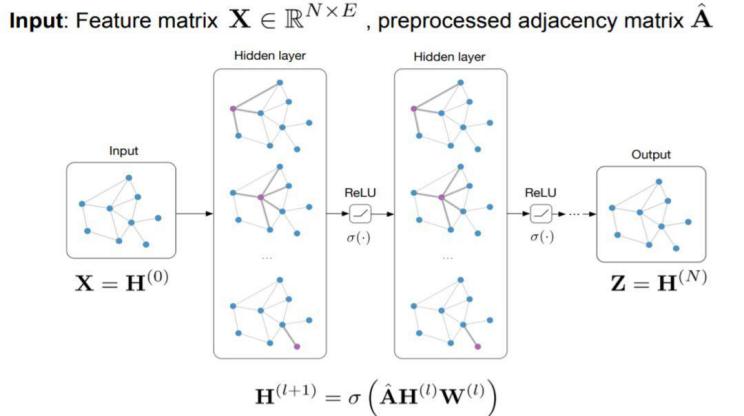
**图卷积神经网络具有卷积神经网络的以下性质：**

1、局部参数共享，算子是适用于每个节点（圆圈代表算子），处处共享。

2、感受域正比于层数，最开始的时候，每个节点包含了直接邻居的信息，再计算第二层时就能把邻居的邻居的信息包含进来，这样参与运算的信息就更多更充分。层数越多，感受域就更广，参与运算的信息就更多。



我们来看GCN这个模型框架，输入是一张图，经过一层一层计算变换，最后输出一张图。



**GCN模型同样具备深度学习的三种性质：**

1、层级结构（特征一层一层抽取，一层比一层更抽象，更高级）；

2、非线性变换 （增加模型的表达能力）；

3、端对端训练（不需要再去定义任何规则，只需要给图的节点一个标记，让模型自己学习，融合特征信息和结构信息。）

**GCN四个特征：**

1、GCN 是对卷积神经网络在 graph domain 上的自然推广。

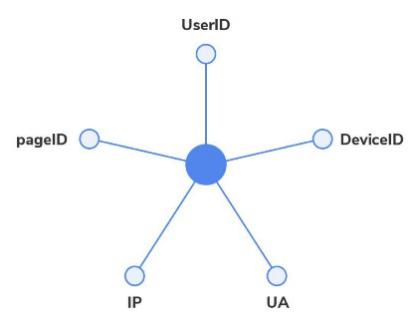
2、它能同时对节点特征信息与结构信息进行端对端学习，是目前对图数据学习任务的最佳选择。

3、图卷积适用性极广，适用于任意拓扑结构的节点与图。

4、在节点分类与边预测等任务上，在公开数据集上效果要远远优于其他方法。

**我们怎么用图卷积神经网络**

下面分享一个我们在实际应用场景中的实验：



实验输入是一个验证数据构成的图数据，节点是验证事件以及事件相关的属性节点。如IP，DeviceID，UA等节点。（我们总计用了30天的验证数据，每两个小时的数据构成一张图，共360张图。）

实验输出是对事件节点进行人机分类，正常或者异常。

**实验细节**

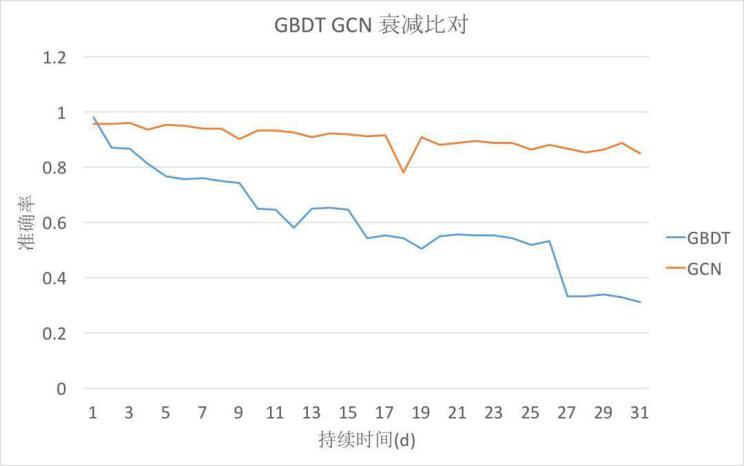
**网络结构：**

GCN(128)->GCN(64)->GCN(64)->Linear(2)

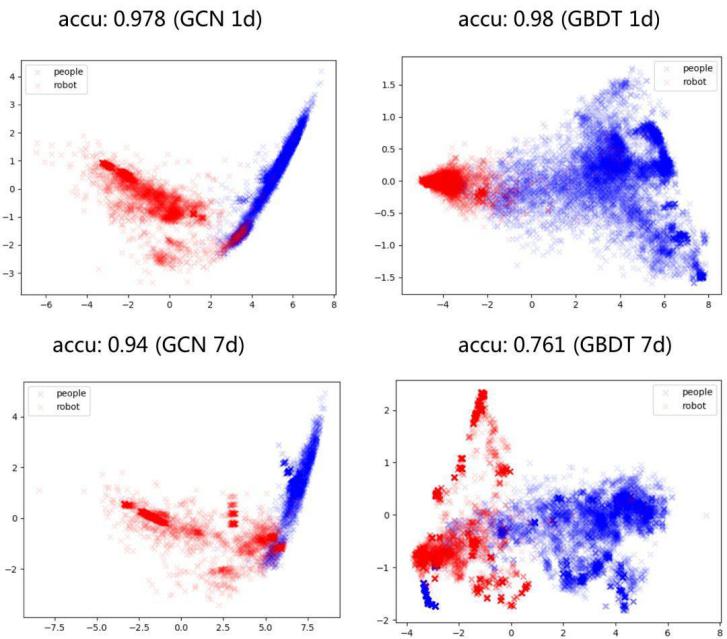
**训练:** Adam优化器, lr=0.001

**参照基准:** 以只能学习特征信息的GBDT做为基准, grid\_search 搜索超参数，GBDT是目前最流行的浅层分类器。

我们用第一天的数据做训练，持续30天预测结果如下：



GCN模型的准确率衰减比较小，而GBDT的衰减很严重。可见，GCN模型的人机判别效果要好，鲁棒性好。



7d评估效果可视化，（用第一天的数据训练模型，第七天观察其预测效果及最后一层输出的tsne可视化结果)。上图可以看出，GCN在第七天时对样本判别的分界面仍很明显，但是GBDT对样本判别的分界面已经很模糊类了。综上，GCN学到的结构信息在人机判别中不仅效果很好，也具有更好的鲁棒性。

**写在最后**

由于时间有限，很多问题浅尝辄止，关于GCN还有很多有趣的东西。我们将开设专栏《Graph Learning》，作者会分享给大家

**有监督学习和无监督学习中的损失函数**

监督学习就是在假设空间 F 中选取一个模型 f 作为决策函数，对于给定的输入X，给出相应的输出f(X)，这个输出的预测值f(X)与真实值Y可能一致也可能不一致，**用一个损失函数(lossfunction)或代价函数(cost function)来度量预测错误的程度，刻画了模型和训练样本的匹配程度**。**损失函数是f(X)与Y的非负实值函数，记作L(Y,f(X))。**

**回归是平方损失函数，分类是交叉熵。**

**无监督学习并没有给出label，得到的信息是标量(有/无),**

**这个类别不是由label给的而是由反馈反映的。**

**神经网络中的loss function，最外层神经元识别的label是人为标定，通过反馈，神经元学到，每一个神经元都是一个二分类。**

无监督学习的特点是，模型学习的数据没有标签，因此无监督学习的目标是通过对这些无标签样本的学习来揭示数据的内在特性及规律，其代表就是聚类。与监督学习相比，监督学习是按照给定的标准进行学习（这里的标准指标签），而无监督学习则是按照数据的相对标准进行学习（数据之间存在差异）。通常无监督学习是指不需要人为注释的样本中抽取信息。例如word2vec。聚类有着自己的性能度量，这和监督学习的损失函数类似。考虑聚类的任务的目的，容易想到就是”物以类聚“，即达到”簇内相似度高“，”簇间相似度低“的性能效果。具体的性能度量有两类，一类是外部指标，与某个专家给定的参考模型进行比对，另一类是内部指标，只考虑自己聚类之后的结果。

**监督学习问题和规则化的理解**

监督机器学习问题无非就是“minimizeyour error while regularizing your parameters”，也就是在规则化参数的同时最小化误差。最小化误差是为了让我们的模型拟合我们的训练数据，而规则化参数是防止我们的模型过分拟合我们的训练数据。因为参数太多，会导致我们的模型复杂度上升，容易过拟合，也就是我们的训练误差会很小。但训练误差小并不是我们的最终目标，我们的目标是希望模型的测试误差小，也就是能准确的预测新的样本。所以，我们需要保证模型“简单”的基础上最小化训练误差，这样得到的参数才具有好的泛化性能（也就是测试误差也小），而模型“简单”就是通过规则函数来实现的。另外，规则项的使用还可以约束我们的模型的特性。这样就可以将人对这个模型的先验知识融入到模型的学习当中.

还有几种角度来看待规则化的。规则化符合奥卡姆剃刀(Occam’s razor)原理：在所有可能选择的模型中，我们应该选择能够很好地解释已知数据并且十分简单的模型。从贝叶斯估计的角度来看，规则化项对应于模型的先验概率。民间还有个说法就是，规则化是结构风险最小化策略的实现，是在经验风险上加一个正则化项(regularizer)或惩罚项(penalty term)。

**DNN、RNN、CNN**

这3个名词其实是第三代神经网络里运用非常多3大算法：DNN（深度神经网络）、RNN（递归神经网络）、CNN（卷积神经网络）。

**三代神经网络的发展**

在正式开讲这3者的区别之前，我们先简单做个回顾，第一代和第二代神经网络到底是什么？

**第一代神经网络又称为感知器，在1950年左右被提出来**，它的算法只有两层，输入层输出层，主要是线性结构。它不能解决线性不可分的问题，对稍微复杂一些的函数都无能为力，如异或操作。

为了解决第一代神经网络的缺陷，**在1980年左右Rumelhart、Williams等人提出第二代神经网络多层感知器（MLP）。**和第一代神经网络相比，第二代在输入层之间有多个隐含层的感知机，可以引入一些非线性的结构，解决了之前无法模拟异或逻辑的缺陷。

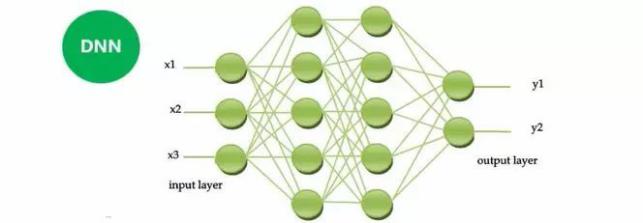
第二代神经网络让科学家们发现神经网络的层数直接决定了它对现实的表达能力，但是随着层数的增加，优化函数愈发容易出现局部最优解的现象，由于存在梯度消失的问题，深层网络往往难以训练，效果还不如浅层网络。

**2006年Hinton采取无监督预训练**（Pre-Training）的方法解决了梯度消失的问题，使得深度神经网络变得可训练，将隐含层发展到7层，神经网络真正意义上有了“深度”，由此揭开了深度学习的浪潮，第三代神经网络开始正式兴起。

**DNN：深度神经网络**

从结构上来说，DNN和传统意义上的NN（神经网络）并无太大区别，最大的不同是层数增多了，并解决了模型可训练的问题。

简言之，**DNN比NN多了一些隐层，但这些隐层的作用是巨大的**，带来的效果是非常显著和神奇的。



当然第三代神经网络能够带来神奇的效果，并不仅仅是因为它的模型结构和训练方法更为优化、算法更加先进，**最重要的是随着移动互联网的普及海量数据的产生和机器计算能力的增强。**

DNN中的“deep”意为深度，但深度学习中深度没有固定的定义或者衡量标准，不同问题的解决所需要的隐含层数自然也是不相同的，就大家比较熟识的语音识别来说，解决问题可能4层就够了，但一般图像识别需要达到20多层才能够解决问题。

DNN最大的问题是只能看到预先设定的长度的数据，对于语音和语言等前后相关的时序信号的表达能力还是有限的，基于此提出了RNN模型，即递归神经网络。

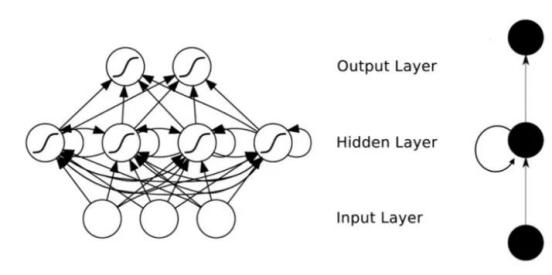
**RNN：递归神经网络**

全连接的DNN存在着一个无法解决的问题：**无法对时间序列上的变化进行建模。**

为了应对这种需求，业内提出了上文中提到的递归神经网络RNN。

在普通的全连接网络中，DNN的隐层只能够接受到当前时刻上一层的输入，而在RNN中，神经元的输出可以在下一时间段直接作用到本身。**换句话说，就是递归神经网络它的隐层不但可以接收到上一层的输入，也可以得到上一时刻当前隐层的输入。**

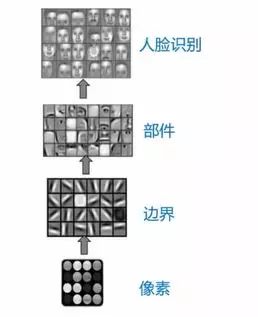
这一个变化的重要意义就在于使得神经网络具备了历史记忆的功能，原则上它可以看到无穷长的历史信息，这非常适合于像语音语言这种具有长时相关性的任务。



**CNN：卷积神经网络**

**卷积神经网络主要是模拟人的视觉神经系统提出来的。**

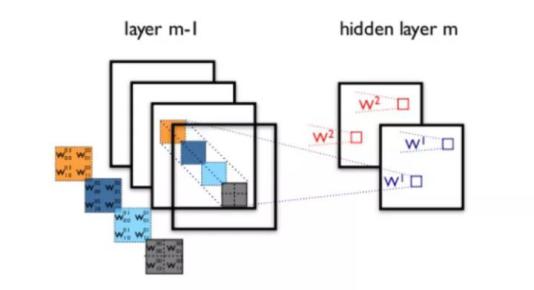
以CNN做人脸识别任务为例，先得到一些像素信息，再往上层得到一些边界信息，然后再往上提取就是一些人脸的部件信息，包括眼睛、耳朵、眉毛嘴巴等，最后是人脸识别，这整个过程和人的视觉神经系统是非常相似的。



卷积神经网络的结构依旧包括输入层、隐藏层和输出层，**其中卷积神经网络的隐含层包含卷积层、池化层和全联接层3类常见构筑，**接下来我们着重讲解下卷积和池化的相关知识点。

卷积层的功能是对输入数据进行特征提取，其内部包含多个卷积核，一个卷积核覆盖的原始图像的范围叫做感受野（权值共享）。

一次卷积运算(哪怕是多个卷积核)提取的特征往往是局部的，难以提取出比较全局的特征，因此需要在一层卷积基础上继续做卷积计算，这就是多层卷积。



在卷积层进行特征提取后，输出的特征图会被传递至池化层进行特征选择和信息过滤。池化层包含预设定的池化函数，其功能是将特征图中单个点的结果替换为其相邻区域的特征图统计量。

**通过这种池化的操作，能够一定程度上克服图像的一些旋转和局部的细微变化，**从而使得特征的表达更加稳定。



CNN 专门解决图像问题的，可用把它看作特征提取层，放在输入层上，最后用MLP 做分类。

RNN 专门解决时间序列问题的，用来提取时间序列信息，放在特征提取层（如CNN）之后。

DNN 说白了就是 多层网络，只是用了很多技巧，让它能够 deep 。

**CNN和GCN：**

1 什么是离散卷积？CNN中卷积发挥什么作用？

了解GCN之前必须对离散卷积（或者说CNN中的卷积）有一个明确的认识：

如何通俗易懂地解释卷积？这个链接的内容已经讲得很清楚了，离散卷积本质就是一种加权求和。

如图1所示，CNN中的卷积本质上就是利用一个共享参数的过滤器（kernel），通过计算中心像素点以及相邻像素点的加权和来构成feature map实现空间特征的提取，当然加权系数就是卷积核的权重系数。

那么卷积核的系数如何确定的呢？是随机化初值，然后根据误差函数通过反向传播梯度下降进行迭代优化。这是一个关键点，卷积核的参数通过优化求出才能实现特征提取的作用，GCN的理论很大一部分工作就是为了引入可以优化的卷积参数。

2 GCN中的Graph指什么？为什么要研究GCN？

CNN是Computer Vision里的大法宝，效果为什么好呢？原因在上面已经分析过了，可以很有效地提取空间特征。但是有一点需要注意：CNN处理的图像或者视频数据中像素点（pixel）是排列成成很整齐的矩阵（如图2所示，也就是很多论文中所提到的Euclidean Structure）。

与之相对应，科学研究中还有很多Non Euclidean Structure的数据，如图3所示。社交网络、信息网络中有很多类似的结构。

实际上，这样的网络结构（Non Euclidean Structure）就是图论中抽象意义上的拓扑图。

所以，Graph Convolutional Network中的Graph是指数学（图论）中的用顶点和边建立相应关系的拓扑图。

**那么为什么要研究GCN？原因有三：**

(1)CNN无法处理Non Euclidean Structure的数据，学术上的表达是传统的离散卷积（如问题1中所述）在Non Euclidean Structure的数据上无法保持平移不变性。通俗理解就是在拓扑图中每个顶点的相邻顶点数目都可能不同，那么当然无法用一个同样尺寸的卷积核来进行卷积运算。

(2)由于CNN无法处理Non Euclidean Structure的数据，又希望在这样的数据结构（拓扑图）上有效地提取空间特征来进行机器学习，所以GCN成为了研究的重点。

(3)读到这里大家可能会想，自己的研究问题中没有拓扑结构的网络，那是不是根本就不会用到GCN呢？其实不然，广义上来讲任何数据在赋范空间内都可以建立拓扑关联，谱聚类就是应用了这样的思想（谱聚类（spectral clustering）原理总结）。所以说拓扑连接是一种广义的数据结构，GCN有很大的应用空间。

综上所述，GCN是要为除CV、NLP之外的任务提供一种处理、研究的模型。

**DNN🡪CNN🡪RNN🡪GCN**

深度学习一直都是被几大经典模型给统治着，如CNN、RNN等等，它们无论再CV还是NLP领域都取得了优异的效果，那这个GCN是怎么跑出来的？是因为我们发现了很多CNN、RNN无法解决或者效果不好的问题——**图结构**的数据。我们做图像识别，对象是图片，是一个二维的结构，于是人们发明了CNN这种神奇的模型来提取图片的特征。CNN的核心在于它的kernel，kernel是一个个小窗口，在图片上平移，通过卷积的方式来提取特征。这里的关键在于图片结构上的平移不变性：一个小窗口无论移动到图片的哪一个位置，其内部的结构都是一模一样的，因此CNN可以实现参数共享。这就是CNN的精髓所在。再回忆一下RNN系列，它的对象是自然语言这样的序列信息，是一个一维的结构，RNN就是专门针对这些序列的结构而设计的，通过各种门的操作，使得序列前后的信息互相影响，从而很好地捕捉序列的特征。

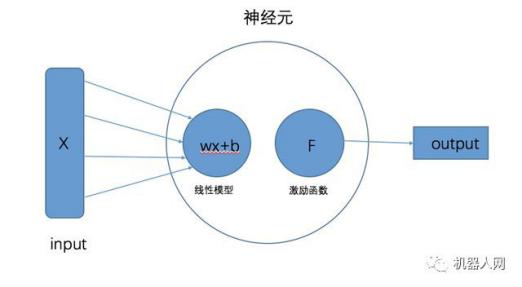
图卷积网络是把CNN拓展到非欧几里德数据。我们知道。

积神经网络之所以效果非常好，是因为利用了数据的局部结构特性（locality），所以在GCN里必须考虑的问题是如何利用数据的局部性。根据局部性不同，可以分成两类方法：（1）频域角度（2）空域角度。以空域角度为例子，直观地说，我们需要解决两个问题：其一是CNN里，神经元除了相邻之外还具有空间位置关系。而图里面没有空间位置关系，本质上缺少区分不同相邻节点的能力。其二是图里，每个节点的相邻节点数目不固定，导致无法固定卷积核的大小，从而无法共享参数。gcn设计了一些机制来解决这些问题。

**一、神经元的构成**

神经元可以说是深度学习中最基本的单位元素，几乎所有深度学习的网络都是由神经元通过不同的方式组合起来。

**一个完整的神经元由两部分构成，分别是“线性模型”与“激励函数”。**如果看过之前的文章，相信可以回忆起其中“线性回归”和“激励函数”的概念



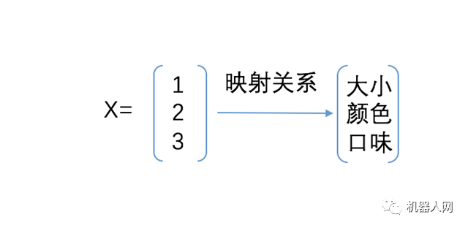
**1.线性模型**

（1）构成

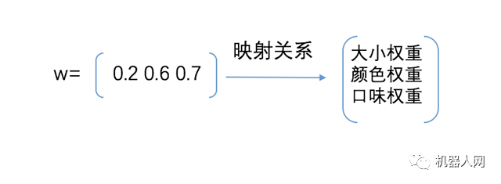
假设这个线性模型的函数： y=wx+b（有木有很熟悉），其中x是一个1xn的向量矩阵，矩阵中的每个向量值即代表样本一个特征的值，w为nx1的权重矩阵（对应向量的所占的比重），b为偏置项。

（2）工作流程

以判定一个苹果的品质为例，我们假定y代表品质变量，x为1×3矩阵，w为3×1矩阵（偏置忽略为0的情况下），具体如下



**x矩阵里的向量值“1、2、3”分别代表一个数据中提取出来的特征向量的值。**



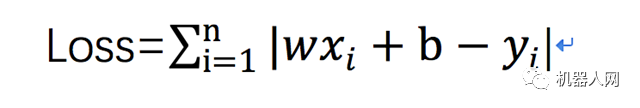
**w矩阵里的“0.2、0.6、0.7”分别代表每个特征向量的权重取值大小。**

**这两个矩阵相乘，最终会得到一个实数（涉及到数学矩阵运算，并非所有都会是实数哦~）**

**1X0.2+2X0.6+3X0.7=3.5**

得到的3.5即我们拟合出来的一个苹果的品质假定为y1，用这个值与已经标定好的真实品质y0做差，就可以得到一个数据的拟合值与真实值的误差，当然真实的计算这可是海量数据计算

用到我第一章分享“线性回归”中对全局性误差函数的定义



**通过这个函数来描述所有数据拟合值与真实值之间的关系，目的也是和机器学习一样，最终是要找到一个符合要求的Loss与w,b之间的映射关系**

以上单个神经元中“线性模型”的运算流程，本质和机器学习中的“线性回归”过程是没有区别的

**2.激励函数**

（1）激励函数的作用

激励函数位于一个神经元线性模型之后，也有翻译成激活函数。它的作用有两个：

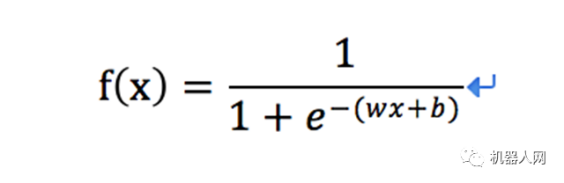
**加入“非线性”因素**

**根据不同训练目的的需要，进行数学函数映射**

为什么要加入“非线性”因素，那是因为“现实世界”的数据不可能都是线性的，如果你强行用“线性模型”去拟合非线性数据，最后得到的结果肯定是“欠拟合”

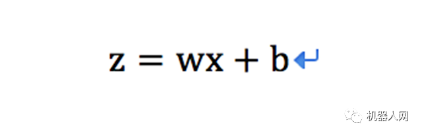
**怎么理解数学函数映射呢，在这里拿最常用的Sigmoid函数举例**

Sigmoid函数定义：

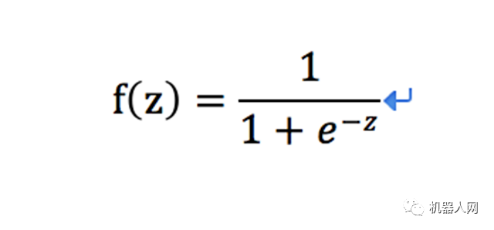


激励函数之前的线性模型“y=wx+b”已经经过运算得到了一个实数（即前面的3.5）

可以作如下的推导



则激励函数sigmoid变为



**下图为Sigmoid函数图形，由图像可以看出，最初的x经过“线性模型”映射为z（z理论上可以为任意大小的实数），而z经过激励函数的再一次映射，最后的输出必然为了【0，1】区间的实数，这就实现了一次数学函数的映射。**

它可以实现一个简单的概率分类判断，假定“0”和“1”各代表一个概念，那么最后的输出在区间【0，1】，更接近“1”，就代表它是更可能是“1”所代表的概念

（2）激励函数的种类

激励函数的种类实在很多，应用的场景也各不相同，**比较常见的除了上面提到的Sigmoid函数外，还有多用于RNN（循环神经网络）的Tanh函数，大部分用于CNN（卷积神经网络）的ReLU函数，以及Linear函数等。**

在这里就不一一列举他们的公式和函数图像了，总之每个激励函数都有自己的“个性”（特性），根据不同的算法模型和应用场景，会搭配使用不同的激励函数，当然最终的目的只有一个，就是让算法模型收敛的越快，拟合的越好

**二、神经网络**

**1.神经网络的构成**

神经网络，其实就是多个神经元横向与纵向的堆叠，最为简单和基础的神经网络可以用下图来表示

通常分为以下三层：

**输入层：负责直接接受输入的向量，通常情况下不对数据做处理，也不计入神经网络的层数。**

**隐含层：是整个神经网络最为重要的部分，它可以是一层，也可是N层，隐含层的每个神经元都会对数据进行处理。**

**输出层：用来输出整个网络处理的值，这个值可能是一个分类向量值，也可能是一个类似线性回归那样产生的连续的值。**

**2.神经网络工作流程**

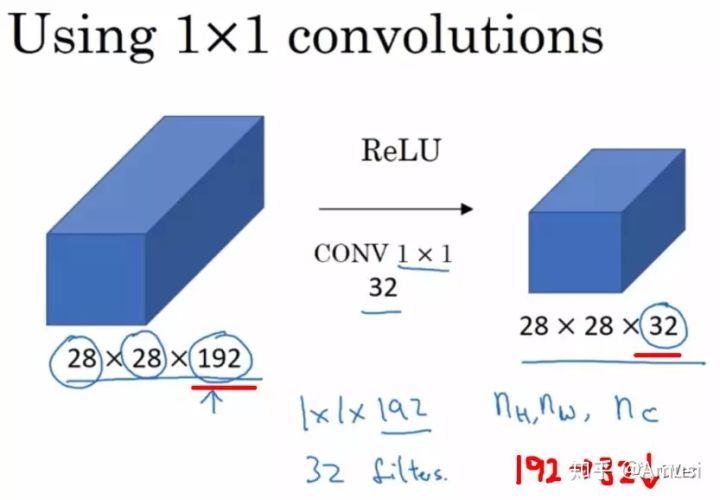
**初步的理解，神经元就是通过图中首尾相连的方式进行连接并实现数据传递，上一个神经元的输出，会成为下一层神经元的输入。**对于一个x向量中的任意一个维度的分量，都会在整个神经网络进行一层一层地处理。

**神经网络的厉害之处就在于，我们可以调节神经网络的层数，网络的拓扑结构以及神经元的参数，去改变对一个输入向量x的不同数学维度上的处理方式，进而达成不同的训练目的。这也是后来像DNN、RNN、CNN成为当下人工智能炙手可热的一大原因。（其实DNN，但从结构上来说，可以简单理解为层数的增加，进而带来对特征的提取和抽象能力的增强）**

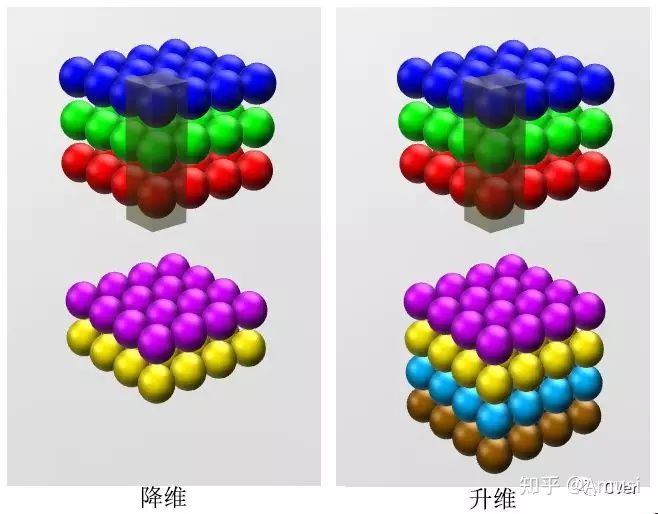
当然，随着网络层数的增加，拓扑结构的复杂，随之而来也会带来整个神经网络的副作用和难题，比如容易陷入局部最优解，梯度消失严重等问题。这也是后续需要探讨和深化了解的东西。

1x1卷积核作用

**降维/升维**



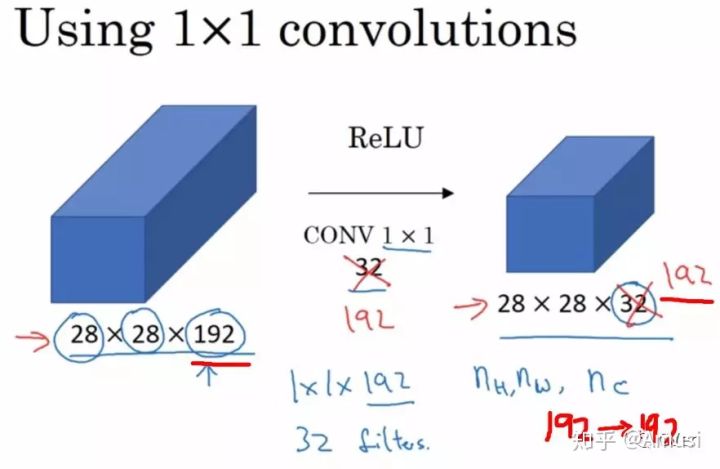
由于 1×1 并不会改变 height 和 width，改变通道的第一个最直观的结果，就是可以将原本的数据量进行增加或者减少。这里看其他文章或者博客中都称之为升维、降维。但我觉得维度并没有改变，改变的只是 height × width × channels 中的 channels 这一个维度的大小而已[5]。



**增加非线性**

1\*1卷积核，可以在保持feature map尺度不变的（即不损失分辨率）的前提下大幅增加非线性特性（利用后接的非线性激活函数），把网络做的很deep。

备注：一个filter对应卷积后得到一个feature map，不同的filter(不同的weight和bias)，卷积以后得到不同的feature map，提取不同的特征，得到对应的specialized neuron[7]。



**跨通道信息交互（channal 的变换）**

例子：使用1x1卷积核，实现降维和升维的操作其实就是channel间信息的线性组合变化，3x3，64channels的卷积核后面添加一个1x1，28channels的卷积核，就变成了3x3，28channels的卷积核，原来的64个channels就可以理解为跨通道线性组合变成了28channels，这就是通道间的信息交互[7]。

注意：只是在channel维度上做线性组合，W和H上是共享权值的sliding window