# **常用算法简介**

接下来我们将介绍八个常用的机器学习算法.

## C4.5

C4.5作为一种决策树算法, 前身是著名的ID3算法, 二者的区别在于划分属性的准则.

决策树是一类常用的机器学习算法, 顾名思义, 决策树是基于树结构进行决策的, 这一点恰是人类面对决策问题的处理机制. 算法通过把实例从根结点排列到某个叶子结点实现分类, 其核心问题在于选择分裂属性和决策树的剪枝. 常用的算法有ID3、C4.5、CART等, 这些算法均采用自顶而下的贪婪算法, 每个结点选择分类效果最好的属性将结点分裂成2个或多个子结点, 下图为根据天气情况判断“周六是否适合打网球”的决策树模型.

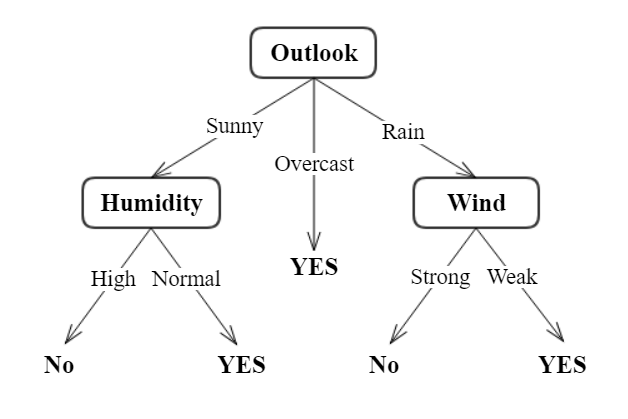


图3 决策树示例图

**Fig.**3 **Decision tree example diagram**

一般的, 一棵决策树包含一个根结点、若干个内部结点和若干个叶结点, 叶结点对应着决策结果, 其他每个结点对应一个属性测试. 由根结点到叶结点的过程即为决策过程, 决策树学习的目的是为产生一种泛化能力较强的分类模型. 通常先选取一个根结点, 根绝其属性生成各个分支, 再以每个分支为根结点或叶结点进一步划分, 具体流程如下

|  |
| --- |
| **输入**：训练集  属性集  **过程**：函数TreeGenerate  1：生成结点node；  2：**if** *D*中样本全属于同一类别C **then**  3: 将node标记为*C*类叶结点；**return**  4：**end if**  5：**if**  **OR** *D*中样本在*A*上取值相同**then**  6： 将node标记为叶结点, 其类别标记为*D*中样本数最多的类；**return**  7：**end if**  8：从*A*中选择最优划分属性  9：**for** 的每一个值**do**  10： 为node生成一个分支；令表示中在上取值为的样本子集  11： **if** 为空 **then**  12： 将分支结点标记为叶结点, 其类别标记为中样本最多的类；**return**  13： **else**  14： 以TreeGenerate  15： **end if**  16：**end for**  **输出：**以node为根结点的一棵决策树 |

图4 决策树学习基本算法

**Fig.**4 **Basic learning algorithm of decision tree**

决策树的生成是一个递归过程, 有三种情形会导致递归返回：①当前结点包含的样本全属于同一类别, 无需划分；②当前属性即为空, 或是所有样本在所有属性上的取值相同, 无法划分；③当前结点包含的样本集合为空, 不能划分.

对于第②种, 直接将当前结点标记为叶结点, 且类别设定为该结点所包含样本最多的类别；对于第③种, 将当前结点标记为叶结点, 将其类别设定为其父结点所包含样本最多的类别.

算法最核心的部分为第8行选择最优划分属性, 不同的划分方法对应着不同的决策树算法.

我们先给出“信息熵”的定义：

假定当前样本集合*D*中第*k*类样本所占的比例为, 则*D*的信息熵定义为

若使用某离散属性*a*对样本集*D*进行划分, 设其有*V*个可能的取值, 则会产生*V*个分支结点, 第*v*个分支结点包含了*D*中所有属性*a*取值为的样本, 即为. 由此可计算的信息熵, 考虑各分支结点样本数量不同, 赋予第*v*个分支结点权重, 于是可计算用属性*a*对样本进行划分所得到的“信息增益”如下

若在算法流程第8行选取使信息增益最大的属性作为最优划分属性, 即可得到ID3决策树学习算法.

由于信息增益偏好取之数目较多的属性, 因此可考虑采用“增益率”来选择最优划分属性, 以减少此偏好带来的不利影响. 增益率定义如下

其中

称为属性*a*的“固有值”

C4.5算法不直接采用增益率最大的划分属性, 而是从候选划分属性中找出信息增益高于平均水平的属性, 再从中选择增益率最高的.

## *k*-Means

在先前介绍的“无监督学习”方法中, 研究最多、应用最广的一大领域是“聚类”. 聚类试图将数据集划分为若干个互不相交的子集, 每个子集称为一个“簇”. 聚类算法可以解释数据的内在性质及规律, 获取数据的一些潜在的概念, 需注意这样的概念是事先未知的, 在聚类完成后需要由使用者把握及解释.

通俗地讲, 聚类是将相似的事物聚集在一起, 而将不相似的事物划分到不同的类别的过程. 那么, 如何衡量两个事物之间的相似度, 我们可通过定义“样本间距离”及“类间距离”进行衡量.

首先给定两样本与, 常用的“距离度量”有

1. **欧氏距离**
2. **曼哈顿距离**
3. **切比雪夫距离**
4. **闵可夫斯基距离**

* 时, 是曼哈顿距离；
* 时, 是欧式距离；
* 时, 是切比雪夫距离.

1. **马氏距离**

有*M*个样本向量, 协方差矩阵记为, 均值记为向量, 则其中样本向量到的马氏距离表示为：

1. **汉明距离**

两个等长字符串与之间的汉明距离定义为两个（相同长度）字对应位不同的数量. 例如字符串“1111”与“1001”之间的汉明距离为2.

1. **Pearson相关系数**
2. **夹角余弦**

需要说明的是, 用距离作为亲疏程度的度量值时, 距离越小, 样品之间的关联性越大；用相似系数作为亲疏程度的度量值时, 相似系数的绝对值越大, 意味着指标之间的关联性越大.

*k-*Means算法需事先给定聚类个数*k*, 聚类目标为最小化*m*个数据与其对应聚类中心点距离平方和

其中

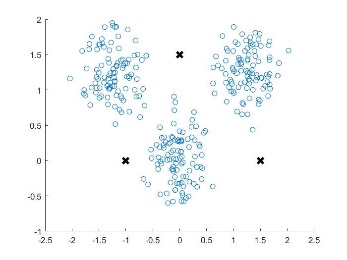
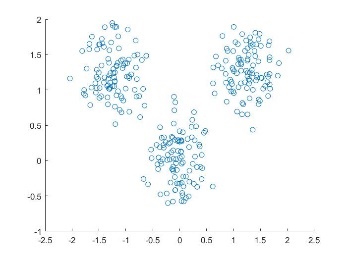
数学上已经证明这是一个NP难问题, 因此*k-*Means算法采用了贪心策略, 通过迭代优化来近似求解式(3.12), 具体流程如下

|  |
| --- |
| **输入**：样本集  聚类簇数*k.*  **过程**：  1：从样本集*D*中随机选择*k*个样本作为初始均指向量  2：**repeat**  3: 令  4： **fordo**  5： 计算样本与各均值向量的距离：;  6： 根据到最近的均指向量确定的簇标记：  7： 将样本划入相应的簇：;  8： **end for**  9： **for** **do**  10： 计算新均值向量：  11： **if then**  12： 将当前均值向量更新为  13： **else**  14： 保持当前均值向量不变  15： **end if**  16： **end for**  17：**until** 当前均值向量均未更新  **输出：**簇划分 |

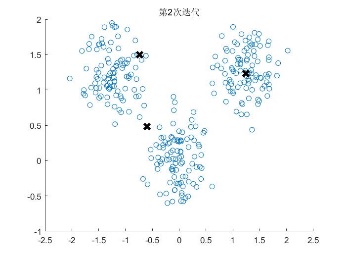
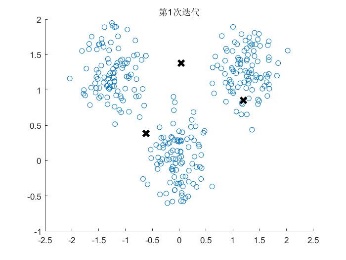
图5 *k*-Means算法

**Fig.**5 ***k*-Means algorithm**

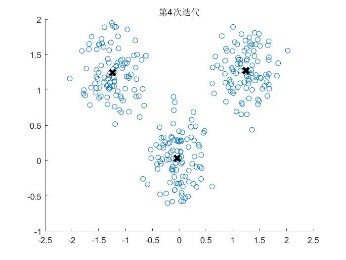
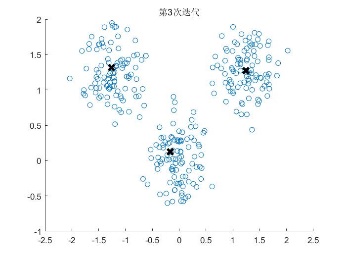
下面演示了*k-*Means算法的迭代过程：



|  |  |
| --- | --- |
| 图6 原始数据分布图 | 图7 初始化聚类中心 |
| **Fig.**6 **Original data distribution diagram** | **Fig.**7 **Initialize the clustering centers** |



|  |  |
| --- | --- |
| 图8 第一次迭代结果 | 图9 第二次迭代结果 |
| **Fig.**8 **The first iteration result** | **Fig.**9 **The second iteration result** |



|  |  |
| --- | --- |
| 图10 第一次迭代结果 | 图11 第二次迭代结果 |
| **Fig.**10 **The third iteration result** | **Fig.**11 **The fourth iteration result** |

## *k*NN

*k*NN算法是一种常用的监督学习算法, 算法工作机制非常简单：给定测试集, 基于某种距离度量找出与训练样本最近的*k*个样本, 基于这*k*个“邻居”的信息来进行预测. 通常, 在分类任务中, 可采用“投票法”, 即选取这*k*个样本中出现最多的类别作为预测结果；在回归任务中, 可采用“平均法”, 即将这*k*个样本的输出平均值作为预测结果；此外, 还可以根据距离远近做加权投票或加权平均, 距离越近权重越大.

同*k-*Means, *k*NN算法有两个重要参数, 一是参数*k*, 当*k*取不同值的时候, 分类结果会显著不同, 另一个是距离计算方式, 若采用不同的距离度量, 可能找到的*k*个“近邻”会有显著差别, 导致分类结果的显著不同.

*k*NN算法是一类典型的“懒惰学习”算法, 即在训练阶段仅保存样本, 训练时间开销为零, 待收到测试样本后再进行处理；与之对应的, 例如上述两个学习方法, 都属于“急切学习”的范畴.

## SVM

SVM, 全称为“支持向量机”(Support Vectory Machine), 是统计学习中一类非常重要的机器学习算法, 具有较好的适应能力及较高的区分率.

给定训练数据集, 找到一个直接划分两个不同的样本类的超平面是一个自然的思路. 假设存在这样一个超平面划分两个样本类, 则称这一问题为“线性可分的”, 但这样的超平面可能有许多个, 如下图, 如何找到一个“最合理”的超平面是我们需要思考的问题.

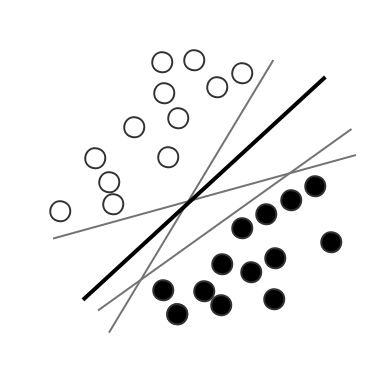


图12 存在多个超平面可将两类样本划分开

**Fig.**12 **Multiple hyperplanes can divide two kinds of samples**

直观上, 我们希望找到一个位于两类训练样本“正中间”的超平面, 即到两类样本距离较为接近的超平面, 如上图纯黑色的直线. 这样的划分超平面对噪声的“容忍性”最强, 即具有最好的泛化性能.

通常, 一个划分超平面可表示为一线性方程：

其中为法向量, 决定超平面的方向；参数*b*为位移量, 决定超平面到原点距离, 可得任意点到该超平面的距离为：

假设该超平面可正确划分训练样本集, 且对任一样本对, 该超平面满足

且距离该超平面最近的样本点满足使等号成立, 这样的样本点被称为“支持向量”. 根据分析可知这样的和总是存在的, 且有两个异类支持向量到超平面的距离和为

称之为“间隔”.

则“线性可分支持向量机”算法的学习目的为找到具有“最大间隔”的划分超平面, 即

等价于

此为SVM的基本型, 该问题为等式约束的图二次规划问题, 可采用拉格朗日乘子法求解, 可写出拉格朗日函数为

其中, , 令对和的偏导为零可得

将式(3.22)代入(3.21)得(3.20)的“对偶问题”如下

注意到式(3.20)有不等式约束, 可知上述过程需满足KKT条件, 即要求

于是, 由第三式可知, 对任意训练样本, 总有或. 若, 则在式(3.22)中不会出现；若, 则必有, 该样本为支持向量. 因此最终模型只与支持向量有关, 故模型称之为“支持向量机”.

式(3.24)为一个二次规划问题, 可采用通用的二次规划算法求解, 问题规模正比于训练样本数. 若考虑问题本身的特性, 则存在一些高效的求解算法如SMO等.

若某样本集不存在线性划分超平面, 例如“异或”, 则上述的“线性可分支持向量机”模型无法求解. 幸运的是, 如果原始空间为有限维（属性有限）, 则必存在一个非线性映射将原样本映射到一高维特征空间且满足样本集线性可分.

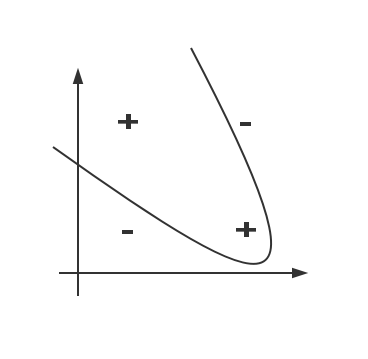


图13 异或问题

**Fig.**13 **XOR problem**

原模型可表示为

其中和为模型参数, 类似的有

对偶问题为

注意到式(3.28)中为样本与样本映射到特征空间后的内积, 有时直接计算很困难, 可设想一个函数：

式(3.28)可改写为

求解得

该方法称之为“核方法”. 由于不知道特征映射的形式, 无法确定选择哪个核函数是最合适的, 因此核函数的选择也是一重要变量.

下表列出了常用的核函数：

表2 常用核函数

**Table** 2 **Common kernel function**

|  |  |  |
| --- | --- | --- |
| 名称 | 表达式 | 参数 |
| 线性核 |  |  |
| 多项式核 |  |  |
| 高斯核 |  |  |
| 拉普拉斯核 |  |  |
| Sigmoid核 |  |  |

## Naive Bayes

假设共有*N*个可能的分类分类, 即记作是将一个真实标记为的样本分类为的损失, 由此基于后验概率我们可以计算出样本被分类为的期望损失, 即在样本上的条件风险

我们希望找到一个判定准则以最小化总体风险

由此定义贝叶斯判定准则：对每个样本, 选择可使条件风险最小的类别标记, 即可最小化总体风险, 称该判别准则为贝叶斯最优分类器

若分类的最优目标为最小化分类错误率, 则判别损失可写作

此时条件风险

最小化分类错误率的贝叶斯最优分类器为

由贝叶斯定理知, 可写作

基于贝叶斯公式估计后验概率有一大难点为类条件概率是所有属性上的联合概率, 难以从有限的样本中直接估计, 故提出“属性条件独立性假设”, 即对已知类别, 假设所有属性相互独立, 式(3.38)可写作

其中*d*为属性数目, 为在第*i*个属性上的取值.

对所有类别来说相同, 则贝叶斯最优分类器可写作

此即朴素贝叶斯分类器的表达式. 因此, 朴素贝叶斯分类器的训练过程需要基于训练集*D*来估计类先验概率, 并为每个属性估计其条件概率.

令为训练集中类别为第类的全部样本, 可估算出类先验概率

若属性为离散属性, 令表示中在第*i*个属性上的取值为的样本组成的集合, 则条件概率可估计为

对于连续属性, 可假设服从正态分布, 即假定和分别为*c*类样本在第*i*个属性上取值的均值与方差, 条件概率密度可估计为

## BP

日前最热门的机器学习研究领域为深度学习方向, 其基础为神经网络算法. 前文介绍到神经网络于二十世纪五十年代便诞生萌芽, 而机器学习真正迈入“深度学习”时代源于2006年Hinton提出并解决了训练过程的深度置信网络(DBN), 成名于2012年ImageNet比赛利用AlexNet以高于第二名十几个百分点的准确率夺得冠军.

首先介绍最基本的神经元模型. 考虑生物神经元, 每个神经元与若干其他神经元连接, 当它“兴奋”时则向其他神经元传递化学物质, 并改变这些神经元的电位, 若某个神经元电位超过一定“阈值”, 则被激活, 即“兴奋”起来, 向其他神经元传递信息. McCulloch于1943年提出的“M-P神经元模型”便是抽象了上述情形, 该模型中, 神经元接受*n*个其他神经元传递来的信号, 并采用加权和作为总输入值, 若大于阈值则被激活并通过“激活函数”处理后输出.

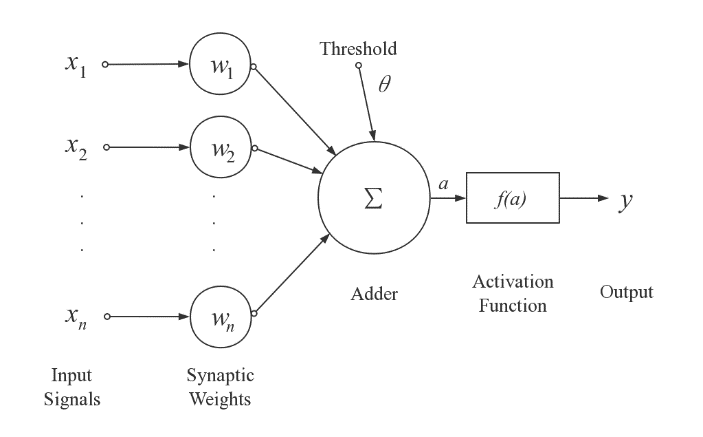


图14 M-P神经元模型

**Fig.**14 **M-P neuron model**

模型需给定激活函数, 常用的有Sigmoid函数等, 模型的训练过程就是调整模型内部参数, 包括权值和阈值.

进一步的, 将许多个这样的神经元按一定的层次结构连接起来, 就得到了神经网络

最简单的神经网络模型为感知机, 可实现逻辑与、或、非运算. 该模型由两层神经网络组成, 输入层接收外界输入信号后传递给输出层, 输出层为M-P神经元. 对于有个输入单输出的感知机, 其输出可表示为, 可将阈值视为输入固定为的结点所对应的权重.

设激活函数为单位阶跃函数, 给定训练样本, 设其输出为, 则感知机的权重可以如此调整：

其中称为学习率, 实际是输出关于参数的偏导数值

对于大于两层的神经网络, 规定第一层为输入层, 最后一层为输出层, 其余为隐含层. 称具有隐含层的网络为“多层网络”, 利用多层网络学习便称为“深度学习”.

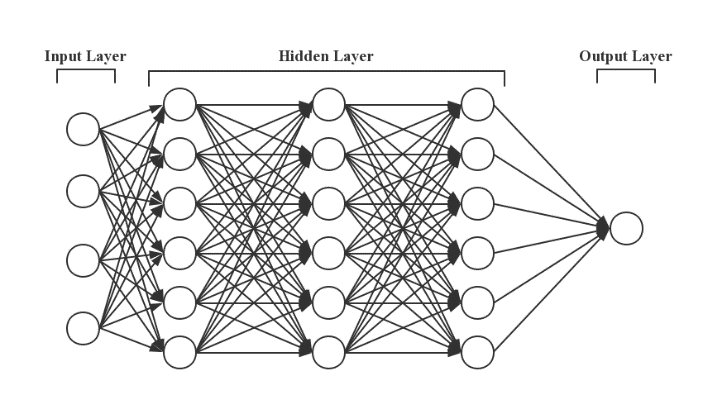


图15 多层神经网络

**Fig.**15 **Multi-layer neural network**

多层网络的学习能力要远强于单层感知机, 而训练难度也远复杂于单层感知机, 直到误差逆传播(简称BP)算法的出现才解决了这一问题. 给定一多层神经网络, 设激活函数为Sigmoid函数, 输出误差为均方误差, 算法流程如下

|  |
| --- |
| **输入**：样本集  学习率  **过程**：  1：在范围内随机初始化网络中所有连接权和阈值  2：**repeat**  3： **for all**  **do**  4：计算每个隐含层结点的输入：  5：计算每个隐含层结点的输出：  6： 计算每个输出层结点的输入：  7： 计算每个输出层结点的输出：  8： 计算每个输出层结点的误差：  9： 计算每个隐含层结点的误差：  10： 更新输入层到隐含层的权值：  11： 更新隐含层到输出层的权值：  12： 更新每个隐含层结点的偏置：  13： 更新每个输出层结点的偏置：  14：**until** 满足终止条件  **输出：**训练完成的多层神经网络 |

图16 BP算法

**Fig.**16 **Error back propagation algorithm**

## CNN

CNN为目前应用最广泛的神经网络模型. 1962年, Hubel和Wiesel通过对猫视觉皮层细胞的研究(局部敏感和方向选择的神经元具有独特的网络结构), 提出了“感受野”(receptive field)的概念. 感受野以某种方式覆盖整个视觉域, 它在输入空间中起局部作用, 因而能够更好地挖掘出存在于自然图像中强烈的局部空间相关性.

1984年, 日本学者Fukushima基于感受野概念提出的神经认知机, 是感受野概念在人工神经网络领域的首次应用.

随后, LeCun等人基于Fukushinma的研究工作并使用BP算法设计并训练了LeNet-5, 并在手写数字识别等模式识别领域取得较大的成功. LeNet-5是一经典的CNN(Convolutional Neural Network, 卷积神经网络)模型, 后续很多工作便是基于此进行改进, 例如前先介绍的AlexNet便是改进LeNet-5一举夺得2012年ImageNet竞赛第一名.

较传统MLP神经网络, CNN具有局部连接、权值共享、池化操作及多层结构等优良特性, 其前馈过程更类似于真实生物视觉过程. CNN通过权值共享减少了需要训练的权值个数、降低了网络的计算复杂度, 同时通过池化操作使得网络对输入的局部变换具有一定的不变性如平移不变性、缩放不变性等, 提升了网络的泛化能力. CNN将原始数据直接输入到网络中, 然后隐性地从训练数据中进行网络学习, 避免了手工提取特征、从而导致误差累积的缺点, 其整个分类过程是自动的.

CNN的基本结构由输入层、卷积层(Convolut-ional layer)、池化层(Pooling layer)、全连接层及输出层构成. 卷积层和池化层一般会取若干个, 采用卷积层和池化层交替设置. 整体网络结构如下所示

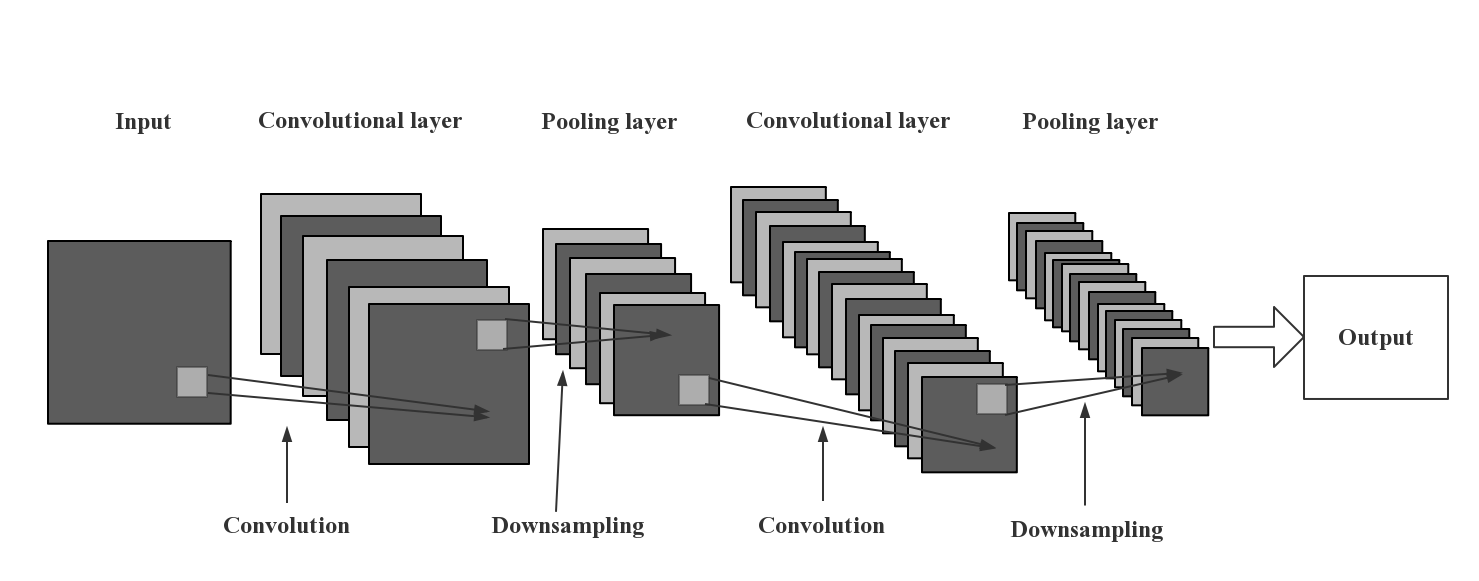


图17 卷积神经网络

**Fig.**17 **Convolutional neural network**

1. **卷积层**

卷积层由多个特征图(Feature Map)组成, 每个特征面由多个神经元组成, 每一个神经元通过卷积核与上一层特征面的局部区域相连. 卷积核是一个权值矩阵, 其作用为提取特征, 第一层提取如边缘、线条、角落等初级特征, 后几层可提取更高级的特征. 卷积阶段的输入为个大小的二维特征图构成的三维数组. 每个特征图记为. 该阶段的输出也是个三维数组, 由个大小的特征图构成. 在卷积阶段, 连接输入特征图和输出特征图的权值记为, 即可训练的卷积核, 卷积核的大小为. 输出特征图为

其中, 为二维离散卷积运算, 示意图如下所示

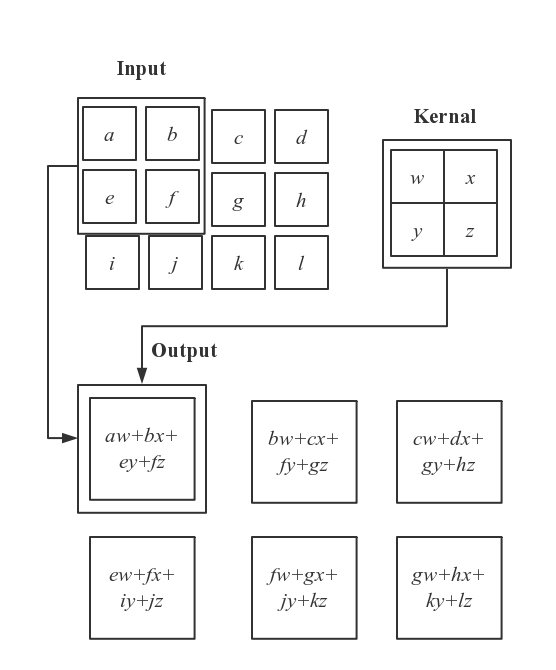


图18 卷积运算

**Fig.**18 **Convolution operation**

对卷积阶段得到的特征作非线性变换, 即“激活”操作, 常用激活函数有

sigmoid

tanh

softsign

ReLU

1. **池化层**

池化层即对输入的特征图作下采样操作, 通常采用平均池化(Average pooling)或最大池化(Max pooling)操作. 平均池化依据定义的邻域窗口计算特定范围内像素的均值, 邻域窗口平移步长大于1(小于等于池化窗口的大小); 最大池化则将均值替换为最值输出到下个阶段. 该层主要目的为降低模型复杂度, 减少运算时间.

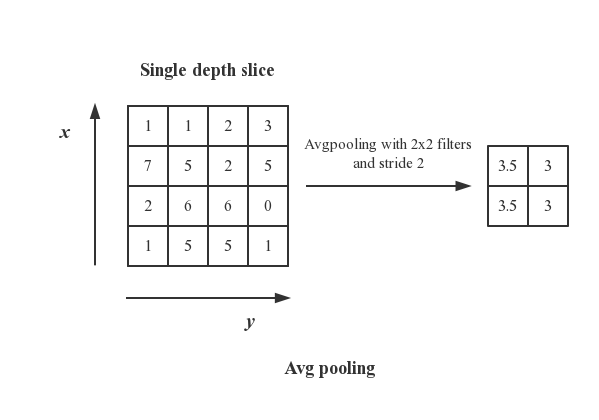


图19 平均池化

**Fig.**19 **Average pooling**

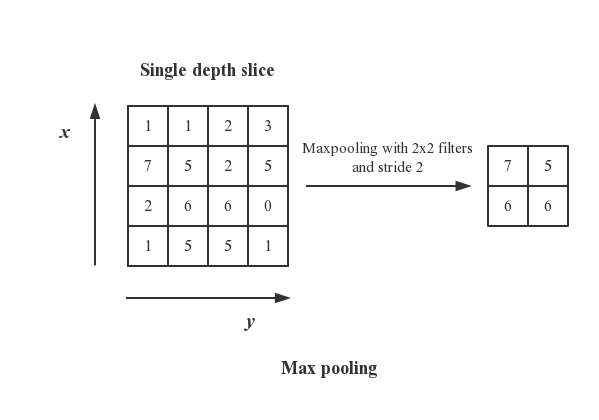


图20 最大池化

**Fig.**20 **Max pooling**

1. **全连接层**

在CNN结构中, 经多个卷积层和池化层后, 连接着1个或1个以上的全连接层. 为了提升CNN网络性能, 全连接层每个神经元的激励函数一般采用ReLU函数. 最后一层全连接层的输出值被传递给一个输出层, 可以采用softmax逻辑回归(softmax regression)进行分类. 通常, CNN的全连接层与MLP的结构一样, CNN的训练算法也多采用BP算法.