神经网络与深度学习

张翼鹏

December 2, 2019

目录

1 简述神经网络

思想：人工神经网络是一种模仿动物神经网络行为特征，进行分布式并行信息处理的数学模型。这种网络依靠系统的复杂程度，通过调整内部大量节点之间相互连接的关系，从而达到处理信息的目的。下面以最基本的前馈神经网络为例，对神经网络的工作流程进行讲解。

步骤：1、构造神经网络结构：输入层-隐藏层-输出层。

2、初始化权重和偏置，对于每个输入x，使用前向传播计算每个神经元的激活值a。

$$a^l = \sigma(w^l a^{l-1} + b^l)$$

3、通过反向传播计算代价函数对权重和偏置的偏导。

$$\delta^L = \nabla_a C \odot \sigma'(z^L)$$

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$$

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l$$

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l$$

4、利用小批量梯度下降法更新权重和偏置，直到达到理想效果。

$$w'_k = w_k - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial w_k}$$

$$b'_l = b_l - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial b_l}$$

优点：具有很强的非线性拟合能力；上限很高，如果数据量充足且超参数选择得当，将达到非常惊艳的效果。

缺点：需要大量数据；算法较为复杂，需要确定的超参数较多；无法解释其推理过程和推理依据。

2 反向传播

2.1 什么是反向传播法

首先回顾一下用梯度下降法调整神经网络权重和偏置的公式：

$$w'_k = w_k - \eta \frac{\partial C}{\partial w_k}$$

$$b_l' = b_l - \eta \frac{\partial C}{\partial b_l}$$

而反向传播的核心就是求一个$\frac{\partial C}{\partial w}$ （或$\frac{\partial C}{\partial b}$）的表达式，它让我们细致领悟到如何通过改变权重和偏置来改变整个网络的行为。

2.2 推导反向传播法前的准备

2.2.1 使用矩阵快速计算输出

当我们有了第$l-1$层的所有激活值、第l层的所有权重和偏置，就可以用下式将两层的激活值联系起来（即前向传播）：

$$a_j^l = \sigma(\sum_k w_{jk}^l a_k^{l-1} + b_j^l)$$

我们用矩阵的形式改写上式，引入权重矩阵w^l、偏置向量b^l以及向量化函数σ，上式可以被改写为：

$$a^l = \sigma(w^l a^{l-1} + b^l)$$

矩阵化的表达式省略了索引下标，形式上更加美观简洁；且在实际应用中可以利用编程软件的矩阵库中提供的矩阵乘法、向量加法等快速方法，提高了算法的运算速度。

另外值得注意的是，在上式中，我们计算了中间量z^l，称为第l层神经元的带权输入：

$$z^l \equiv w^l a^{l-1} + b^l \qquad a^l = \sigma(z^l)$$

在接下来的讨论中，我们会充分利用带权输入z^l

2.2.2 关于代价函数的两个假设

代价函数形式如下：

$$C = \frac{1}{2n} \sum_x \left\| y(x) - a^L(x) \right\|^2$$

L表示神经网络的层数，$a^L(x)$表示对于单个样本x，网络最后输出的激活值的向量。

假设1: 代价函数可以被写成每个训练样本x的代价函数C_x的均值，即$C = \frac{1}{n}\sum_x C_x$。对于目前我们所用的二次代价函数，$C_x = \frac{1}{2}\left\| y - a^L \right\|^2$。

原因: 反向传播实际上是对一个独立的训练样本x计算$\frac{\partial C_x}{\partial w}$和$\frac{\partial C_x}{\partial b}$，然后再通过在所有训练样本上取平均值得到$\frac{\partial C}{\partial w}$和$\frac{\partial C}{\partial b}$。实际上，有了这个假设我们可以认为训练样本$x$已经固定住了，可以丢掉其下标，将代价函数$C_x$看作$C$。

假设2: 代价可以写成关于神经网络输出的函数（即C是关于a^L的函数）。

原因: 推导反向传播算法时，我们要以a^L为起点，推出我们想要的表达式。

2.2.3 Hadamard乘积

反向传播算法会用到一种运算，$s \odot t$。例如：

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix} \odot \begin{bmatrix} 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 1*3 \\ 2*4 \end{bmatrix} = \begin{bmatrix} 3 \\ 8 \end{bmatrix}$$

2.3 反向传播的四个基本方程

2.3.1 误差的概念

在开始推导反向传播之前，首先引入一个中间量δ_j^l，称为第l层第j个神经元上的误差。反向传播将给出计算误差δ_j^l的方法，然后将其关联到$\frac{\partial C}{\partial w_{jk}^l}$和$\frac{\partial C}{\partial b_j^l}$上。

在第l层第j个神经元上，当输入a^{l-1}进来时，如果我们给带权输入z_j^l增加一个很小的变化Δz_j^l，会使该神经元输出由$\sigma(z_j^l)$变为$\sigma(z_j^l + \Delta z_j^l)$。这个变化会向网络后边的层进行传播，最终导致整个代价产生$\frac{\partial C}{\partial z_j^l}\Delta z_j^l$的改变。

于是，我们可以根据$\frac{\partial C}{\partial z_j^l}$的值来找到能够优化代价的$\Delta z_j^l$：若$\frac{\partial C}{\partial z_j^l}$的绝对值很大，则可以选择与其符号相反的$\Delta z_j^l$来降低代价；若$\frac{\partial C}{\partial z_j^l}$的值接近0，那么并不能通过改变$\Delta z_j^l$来改善多少代价，我们可以认为这时神经元已经接近最优了。所以这里有一种启发性的认识，称$\frac{\partial C}{\partial z_j^l}$是神经元的误差的度量。

按照上述描述，我们定义第l层第j个神经元上的误差δ_j^l为：

$$\delta_j^l = \frac{\partial C}{\partial z_j^l}$$

2.3.2 四个基本方程

方程1: 输出层误差δ^L的方程：

$$\delta_j^L = \frac{\partial C}{\partial a_j^L}\sigma'(z_j^L)$$

写成矩阵形式：

$$\delta^L = \nabla_a C \odot \sigma'(z^L)$$

由于二次代价函数$C = \frac{1}{2}\sum_j (y_j - a_j)^2$，所以$\frac{\partial C}{\partial a_j^L} = (a_j - y_j)$，写成矩阵形式有$\nabla_a C = (a^L - y)$，于是方程1最终的矩阵形式为：

$$\delta^L = (a^L - y) \odot \sigma'(z^L)$$

解释: 右式第一项$\frac{\partial C}{\partial a_j^L}$表示代价随第$j$个输出激活值的变化而变化的速度；右式第二项$\sigma'(z_j^L)$表示激活函数$\sigma$在$z_j^L$处变化的速度。

方程2: 用下一层的误差δ^{l+1}来表示当前层的误差δ^l：

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$$

解释: 我们把转置后的权重矩阵$(w^{l+1})^T$作用到第$l+1$层的误差δ^{l+1}上，可以凭直觉把它看作是在沿着神经网络反向移动误差。然后我们进行Hadamard乘积运算$\odot\sigma'(z^l)$，让第$l+1$层的误差通过第l层的激活函数反向传递回来得到第l层的误差。

方程3: 代价函数关于任意偏置的改变率：

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l$$

简记为（其中δ和b都是针对同一个神经元）：

$$\frac{\partial C}{\partial b} = \delta$$

方程4: 代价函数关于任意权重的改变率：

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1}\delta_j^l$$

简记为：

$$\frac{\partial C}{\partial w} = a_{\text{in}}\delta_{\text{out}}$$

解释: 当a_{in}很小时，$\frac{\partial C}{\partial w}$也会趋向很小，结果就是作用在低激活值上的权重学习会非常缓慢。

我们将这四个方程组合在一起，还可以总结出一个规律：当$\sigma(z_j^l)$近似为0或1时，$\sigma'(z_j^l) \approx 0$。所以如果某神经元的输出处于低激活值（≈ 0）或高激活值（≈ 1）时，我们称该神经元已经饱和了，对应的权重和偏置的学习将会非常缓慢（甚至终止）。　　我们可以将反向传播看成是一种系统性地应用多元微积分中的链式法则来计算代价函数梯度的方法。更深层地说，反向传播其实是一种巧妙地追踪权重（和偏置）微小变化如何传播并抵达输出层影响代价函数的技术。

3　简述深度学习

深度学习是学习样本数据的内在规律和表示层次，这些学习过程中获得的信息对诸如文字，图像和声音等数据的解释有很大的帮助。它的最终目标是让机器能够像人一样具有分析学习能力，能够识别文字、图像和声音等数据。深度学习是一个复杂的机器学习算法，在语音和图像识别方面取得的效果，远远超过先前相关技术。

4 卷积神经网络CNN

4.1 卷积神经网络的介绍

卷积神经网络是深度学习中最常见的模型。对于图像识别问题，普通的前馈神经网络没有考虑图像的空间结构，而是在以完全相同的方式去对待相距很远和相距很近的输入元素。这种情况下神经网络想获得图像的空间结构信息只能从训练数据中去逐渐推断。

而卷积神经网络在初始架构时就会利用图像的空间结构，所以它非常适用于图像分类问题。卷积神经网络采用了以下三种基本概念，依然以手写数字识别为例展开介绍。

4.2 局部感受野

在卷积神经网络中，输入层是由$28*28$的方形排列的神经元组成，第一个隐藏层的神经元不会与所有输入相连接，而是分别只连接输入层的一个小区域，比如$5*5$的区域，即25个输入神经元。这个区域被称为隐藏神经元的局部感受野。

与普通神经网络类似，隐藏神经元与局部感受野内的每个输入神经元的连接上有一个权重，隐藏神经元自身有一个偏置，可以认为隐藏神经元学习分析的只是它的局部感受野。

我们在输入层移动这个$5*5$的小区域，每在某个方向上移动1个像素（移动的距离称为跨距，跨距在不同网络中可调），便是一个不同的局部感受野，对应一个不同的隐藏神经元。在本例中，第一个隐藏层拥有$24*24$个神经元（隐藏层依然保留了图像的二维结构）。

4.3 共享权重和偏置

对于第一个隐藏层中每一个神经元，我们使用相同的权重和偏置。这意味着第一个隐藏层的所有神经元检测完全相同的特征，只是检测的位置不同。这种设定能很好地适应图像的平移不变性。

我们把输入层到第一个隐藏层的映射称为一个特征映射，把定义特征映射的权重和偏置称为共享权重和共享偏置，二者共同称为一个卷积核或滤波器。一个特征映射只能检测一种局部特征，为了完成图像识别，我们需要多个（记为k）不同的特征映射，它们共同构成了一个卷积层。

共享权重和偏置的一个优点是大大减少了卷积网络的参数，加快了训练速度，有助于我们最终建立卷积深度网络。

4.4 混合层

除了上述的卷积层，卷积神经网络中还包括混合层。混合层通常紧接着卷积层使用，功能是简化卷积层输出的信息。

混合层的每个神经元在卷积层的每一个特征映射上对应一小块互不重叠的区域（比如$2*2$），它以某种方式将这一小块区域的所有激活值混合。比如最大值混合，即输出区域内最大的激活值；或者$L2$混合，即输出区域内所有激活值的平方和的平方根。

5 其他深度学习的模型

5.1 递归神经网络RNN

在卷积神经网络（CNN）中，各样本之间是独立的，且输入层的输入值完全决定了最后的输出。

而RNN是某种体现出随时间动态变化的特性的神经网络，所以RNN尤其适用于语音识别和自然语言处理等需要处理时序数据或过程的领域中。RNN中每个输出不仅受本次输入影响，还与网络之前的输出有关，即每一刻的输出都是带着之前输出值加权之后的结果。

但是，RNN对于短期记忆的模型效果很好，却无法进行长期记忆的输出，因为权重累加过于庞大，可能导致结果失真、运算效率低下。所以$LSTM$应运而生。

5.2 长短期网络$LSTM$

$LSTM$中的神经元可以判断当前哪些值需要被遗忘，哪些值需要被长期记忆。事实上，$LSTM$就是在RNN的基础上，增加了对过去状态的过滤，从而可以选择哪些状态对当前更有影响，而不是简单地选择最近的状态。

5.3 深度信念网络DBN

DBN是一种生成式模型，即在DBN中，我们有时可以指定某些特征神经元的值，然后反向运行，生成输入值。举例来说，DBN在手写数字图像上的训练同样可以用来生成类似的手写数字图像，换句话说DBN可以学习写数字的能力。

从概率分布的角度考虑，每个样本均有特征x_i对应分类标记y_i，我们可以将所有分类模型分为两种：

生成式模型：学习得到联合概率分布$P(x,y)$，然后求条件概率分布。此类模型能够学习到数据生成的机制。当样本数足够多时，算法能够更快地收敛，但计算

量较大。

判别式模型：学习得到条件概率分布$P(y|x)$。此类模型比较节省计算资源，由于我们不需要输入的分布来计算条件概率，所以我们可以对输入进行抽象（如降维），从而简化学习问题。

DBN的另一个特点是可以进行无监督或半监督的学习。例如在学习图像数据时，即使训练数据是无标签的，DBN也可以从中学习有用的特征用来理解其他图像。

5.4 强化学习

强化学习模型由两部分组成，一个是智能体（agent），另一个是环境（environment）。

智能体指的是强化学习算法，环境则表示智能体执行动作时所处的场景。环境首先向智能体发送一个状态，然后智能体基于其知识采取动作来响应该状态。之后，环境发送下一个状态，并把奖励返回给智能体。智能体用环境所返回的奖励来更新其知识，对上一个动作进行评估，然后响应下一个状态。如果智能体的某个行为策略导致环境正的奖赏(强化信号)，那么它以后产生这个行为策略的趋势便会加强。智能体的目标是在每个离散状态发现最优策略以使期望的奖赏和最大。