关于深度神经网络算法的报告

赵时 物理学院 1900011395

【摘要】：作为机器学习领域最热门的研究方向，近年来深度学习发展迅猛、应用广泛，将人工智能推向了新的高潮。深度学习中相关核心算法、优化算法也渐渐进入大众视角。本文初步介绍深度学习中的深度神经网络算法，分析算法在机器学习领域的优势所在，并对深度学习应用和前景进行展望。

【关键词】：深度学习，DNN 网络，前向传播和反向传播算法，目标函数，优化器

【正文】：

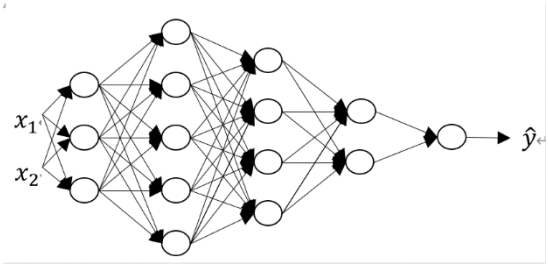
一、机器学习算法和深度学习算法简介

机器学习灵感的源泉来源于图灵的一个问题：通用计算机能否进行创新和学习。在传统的符号主义人工智能中（包括经典的计算机程序设计），人们输入规则和数据，得到答案。监督型的机器学习则接受数据和答案，试图通过训练掌握其中联系的规则。机器学习算法通过对输入数据利用核方法等相关算法进行运算，得到输入的标签，并与给定标签进行比对，利用反馈算法对计算过程进行优化，以达到学习的目的。但是机器学习存在映射空间变换简单、特征工程复杂等问题，于是简单的、端到端的深度学习模型应运而生。

普通的深度学习算法由许多数据变换层堆叠而成，每层对输入数据所做的具体操作保存在该层的权重之中。可以把这样的模型类比于一个蒸馏器，随着网络的深入，数据与原始数据的差别越来越大，而关于最终结果的信息却越来越丰富，即数据更“纯”了。人们按照特定的规则给这些权重找到初始值，之后让网络计算出结果，通过定义损失函数的方式衡量计算出的结果和预期结果之间的距离，利用优化器实现反向传播算法，以实现权重的更新学习。多次训练循环之后，得到训练较好的网络。通过增加网络的深度和提高网络更新权重的自主性的，深度学习克服了机器学习的一些弊端，得到了更好的应用。

二、深度神经网络算法模型

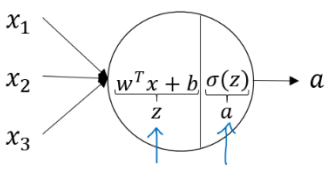
1. 深度神经网络模型框架



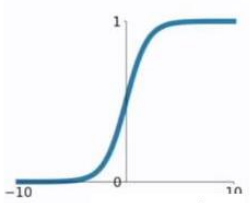
右图为深度学习网络框架的典型模型，主要用来处理数据的分类、回归等数字问题。模型由输入层（输入x_1和x_2两个特征），输出层（输出希望得到的标签）和隐藏层（Hidden layer）构成。隐藏层中有多个节点（Neuron），每一个节点都与前一层输出的全部激活值进行运算，使得每层网络的特征紧密联系，更有利于掌握输入的核心特征。输出层也可以有多个节点，以解决多分类问题。

2. 深度学习前向传播算法

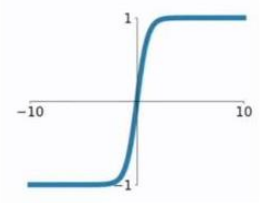
给定x_1和x_2，深度神经网络利用前向传播计算最后的标签 y。具体的计算过程是，在每一个节点处，都对由前一层输入的全部激活值进行如下运算得到下一组激活值。计算过程是将输入的激活值和这个节点的一组权重向量进行点积运算，将得到的值加上偏差（bias），最后再对得到的值用激活函数进行计算，得到这个节点的激活值。



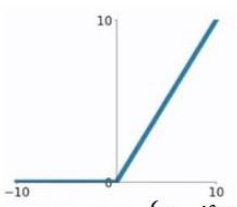
激活函数包括 sigmoid，tanh 和 relu 等函数，主要目的是给网络赋予非线性化成分，使得网络更加丰富。由于之前的运算是一个线性变换，如果不添加非线性成分，由线性运



sigmoid: $g(z) = \frac{1}{1+e^{-z}}$



tanh: $g(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$



relu: $g(z) = \begin{cases} z, & \text{if } z > 0 \\ 0, & \text{if } z < 0 \end{cases}$

算的性质可知，多次线性变换的结果可以由一次线性变换完全等效，多个隐藏层将完全没有意义。具体的激活函数的值和图像见下图，其中的 relu 函数由于其在正半轴恒定的导数特性，便于梯度下降算法进行。在具体的神经网络中使用较多。

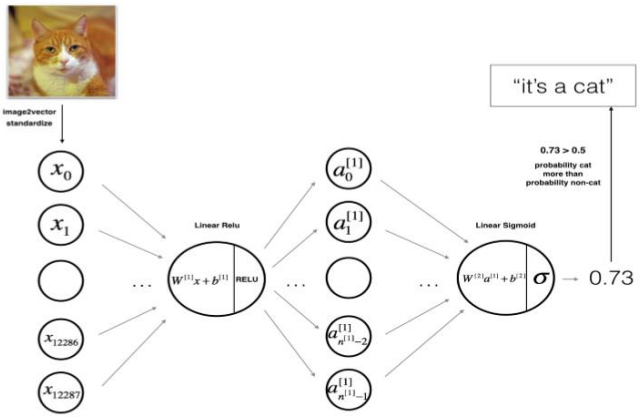
总之，每一层都进行如下运算，逐层运算得到全新的、不同维度的特征向量，直到最后的输出层。

$$
\begin{aligned}
Z^{[1]} &= W^{[1]}X + b^{[1]} \\
A^{[1]} &= g^{[1]}(Z^{[1]}) \\
Z^{[2]} &= W^{[2]}A^{[1]} + b^{[2]} \\
A^{[2]} &= g^{[2]}(Z^{[2]}) \\
&\vdots \\
A^{[L]} &= g^{[L]}(Z^{[L]}) = \hat{Y}
\end{aligned}
$$

为了数学上的简便，我们还可以把每一层不同节点的运算合到一起，得到一个权重矩阵，该矩阵的大小等于输入向量大小（上一层的节点数）乘以输出向量大小（即这一层的节点数）。输入向量和权重矩阵通过矩阵乘法得到新的向量。

$$
\begin{aligned}
z_1^{[1]} &= w_1^{[1]T}x + b_1^{[1]},\ a_1^{[1]} = \sigma(z_1^{[1]}) \\
z_2^{[1]} &= w_2^{[1]T}x + b_2^{[1]},\ a_2^{[1]} = \sigma(z_2^{[1]}) \\
z_3^{[1]} &= w_3^{[1]T}x + b_3^{[1]},\ a_3^{[1]} = \sigma(z_3^{[1]}) \\
z_4^{[1]} &= w_4^{[1]T}x + b_4^{[1]},\ a_4^{[1]} = \sigma(z_4^{[1]})
\end{aligned}
$$

输出层依旧对每个节点先进行一次线性变换，之后根据输出类型的不同选择不同的激活函数。在线性回归问题中输出层可以不采用激活函数，得到的数据是线性变换后任意范围的数据。在二分类问题中，如检测一个图片中是否包含一只小猫咪的图像检测问题（见图），一般采用 sigmoid 函数。由上图 sigmoid 函数可知其返回 0-1 之间的数，我们即可把他看成图片是猫的概率，并规定如果返回值大于 0.5 就将其看作是一只猫，这样经过训练的深度神经网络就可以利用这样的结构判断一张图是否是猫咪。



在多分类问题（比如判断一张小动物的图是猫，是狗还是笔者）中，我们常常采用 softmax 激活函

数作为最后一层。多分类问题最后一层往往输出的是一个向量，我们要即对线性变换后的每一个节点的值都求其指数 e，之后进行归一化，从而得到一个和为 1 的向量，对应于每一种分类的概率。选择概率最高的作为结果，就得到了类别的最终判断。

$$h_\theta(x^{(i)}) = \begin{bmatrix} p(y^{(i)}=1|x^{(i)};\theta) \\ p(y^{(i)}=2|x^{(i)};\theta) \\ \vdots \\ p(y^{(i)}=k|x^{(i)};\theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^{k} e^{\theta_j^T x^{(i)}}} \begin{bmatrix} e^{\theta_1^T x^{(i)}} \\ e^{\theta_2^T x^{(i)}} \\ \vdots \\ e^{\theta_k^T x^{(i)}} \end{bmatrix}$$

3. 深度学习反向传播算法

如果我们有合适的每一层每一个节点权重的取值，就可以通过上述前向传播算法得到结果，而调整和寻找最佳权重就要依靠反向传播算法，因此反向传播算法也是深度学习方法的核心。其核心思路在于，对于不同类型的问题，定义不同的衡量算出的标签和给定的最佳标签的差距的目标函数，这个目标函数是所有权重值的多元函数，通过数学上求梯度的方式调整权重取值，使得目标函数达到全局最小值。

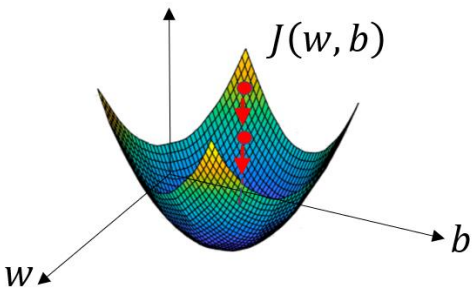
以两分类问题为例，我们定义损失函数如下：

$$J = -\frac{1}{m}\sum_{i=0}^{m}\left(y^{(i)}\log\left(a^{[2](i)}\right) + (1-y^{(i)})\log\left(1-a^{[2](i)}\right)\right)$$

称为二元交叉熵。不难从数学上证明，以向量和矩阵为整体求 J 关于权重值的导数，有如下结果（采用 python numpy 语法）：

$$\begin{aligned} dZ^{[L]} &= A^{[L]} - Y \\ dW^{[L]} &= \frac{1}{m} dZ^{[L]} A^{[L]T} \\ db^{[L]} &= \frac{1}{m} np.\mathrm{sum}(\mathrm{d}Z^{[L]}, axis = 1, keepdims = True) \\ dZ^{[L-1]} &= dW^{[L]T} dZ^{[L]} g'^{[L]}(Z^{[L-1]}) \\ &\vdots \\ dZ^{[1]} &= dW^{[L]T} dZ^{[2]} g'^{[1]}(Z^{[1]}) \\ dW^{[1]} &= \frac{1}{m} dZ^{[1]} A^{[1]T} \\ db^{[1]} &= \frac{1}{m} np.\mathrm{sum}(\mathrm{d}Z^{[1]}, axis = 1, keepdims = True) \end{aligned}$$

其中dX表示$\frac{\partial J}{\partial x}$，[L]表示第 L 层，$Z^{[L]}$表示第 L 层经过线性运算后的结果，$A^{[L]}$表示激活函数作用后的结果，m 表示输入模型的总数据数量。按这种方法进行计算，就可以得到 J 关于每一层的权重 W 和偏差 b 的偏导数。



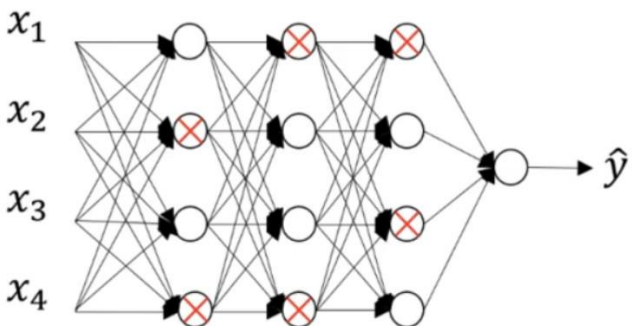
从数学上我们可知，多元函数梯度方向就是每一元偏导数的取值乘以该元对应的方向向量，对应函数下降最快的方向。因此我们可以在梯度方向上更新权重，即：

$$W^{[L]} = W^{[L]} - \alpha dW^{[L]}$$
$$b^{[L]} = b^{[L]} - \alpha db^{[L]}$$

α称为学习率，表示每一次梯度下降的步幅。注意α对于每一个参数都应该是相同的，以保证整个损失函数是按照梯度方向逐渐的下降。

4. 对深度学习算法的一些优化

（1）过拟合问题的解决



普通的深度学习问题的给定数据一般分成训练集和测试集。训练集占很大比例，我们将它们输入搭建好的模型，让模型在数据上进行多次学习，更新权重，之后用测试集进行测试，得到准确率等评估训练结果的数据。有一个问题在于，如果模型在训练集训练很多，有可能导致过拟合问题，即模型参数过分适应训练集的数据，导致其在训练集上表现很棒，可是对于新的数据并不能很好得到结果。一般情况下，我们将数据集的一部分划分为验证集，来验证是否在训练集上产生过拟合问题。

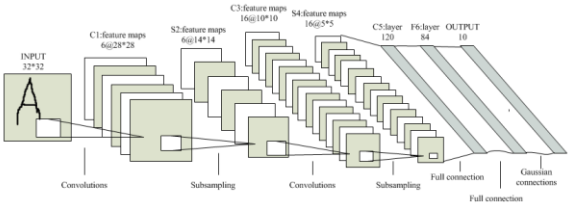
数学上可以证明，过拟合问题一般源于参数的过分复杂，因此我们可以用正则化方法来削弱参数的作用，减少过拟合问题，常用 Dropout 正则化或者 L2 正则化方法。Dropout 正则化指对于每一次训练，删去一些层的一些节点，使得每一次训练的网络变得相对简单。L2 正则化则是给目标函数加上所有权重值的平方的和的某个倍数。由于我们要最小化目标函数，因此也把权重值的大小一定程度缩小了，也起到简化网络的目的。

（2）超参数选择问题

权重成为参数值，而如算法模型层数多少，每层节点的多少，学习率的大小等参数是提前规定好的，不会被机器运行算法而修改，这些参数称为超参数。

对于超参数的选择，一般依据相关专业论文和专家的建议来选。超参数的细微调节，则需要利用验证集。把训练好的模型在验证集上验证，之后依据得到的值修改超参数，直到选出较佳的超参数达到预设目标。

三、深度学习应用和前景



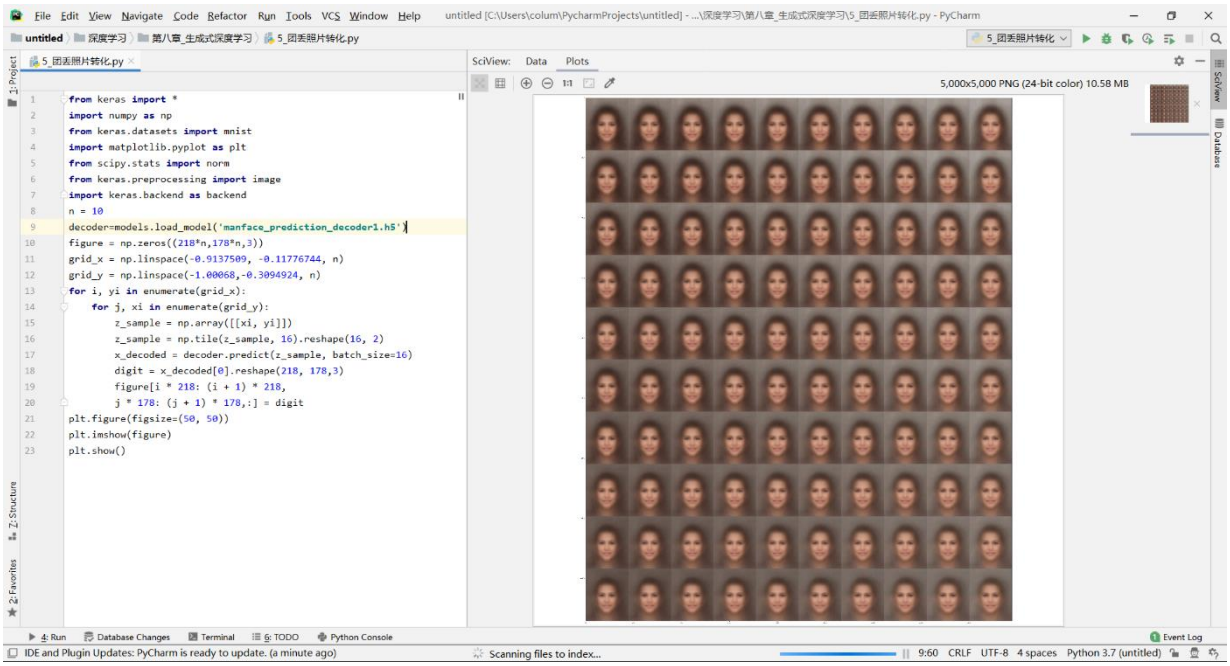
深度学习算法还包括更适合处理图片信息的卷积神经算法（CNN），处理序列 问题的循环神经网络算法、可以自动生成文字或者图像的生成式深度学习算法等等。涉及面极广，应用丰富。

笔者认为，深度学习的深度之美，就在于既可以用其完成宏大的工程，如无人车、无人机、智能模拟，也可以很简单的用它完成生活中的小应用，产生生活中的小美好。

比如，通过简单的 CNN 或深度神经网络可以用 ImageNet 上的数据训练一个动物分类器，将其拷贝到带摄像头和传感器的树莓派上，就可以利用它实现动物园的动物识别

又比如，利用生成时深度学习算法，可以通过把人的脸与高维空间中的点一一对应的形式，剖析脸的深度特征，从而构造出一系列类似于你的机器脸，甚至可以找到从你到你女朋

友的脸部特征过度:



给柯南图片加上星空的特征:

给美丽的景色加上恐怖的元素

总之，利用深度学习算法甚至仅仅利用深度神经网络算法就可以实现许许多多有趣的应用。然而，深度学习算法的发展还远远没有到头，更快更好的深度学习算法、更宏大的人工智能应用正亟待发掘，未来的深度学习算法定将渗透入人们生活的方方面面，人工智能的实现也将 不再遥远!

【参考文献】:

1. 弗朗索瓦·肖莱 Python 深度学习[M] 人民邮电出版社 2018 年 8 月第一版
2. 吴恩达深度学习课程 吴恩达 https://www.coursera.org/learn/machine-learning-projects