1.DNN反向传播算法简介

回顾我们前面学到的监督问题，通常会遇到这种情况，假如有m个训练样本，分别为$\{(x_1,y_1),(x_2,y_2),(x_3,y_3),\ldots,(x_m,y_m)\}$，其中$x$为输入变量，特征维度为n_in，y为输出向量，特征维度为n_out。现在我们利用这m个训练样本来训练模型，当有测试样本$(x_{test},?)$时，需要我们能够预测出y_{test}向量的输出。

现在对应到我们的DNN模型之中，即输入层有n_in个神经元，输出层有n_out个神经元，再加上一些含有若干个神经元的隐含层。此时我们需要找到所有隐含层和输出层所对应的线性系数矩阵W、偏倚向量b，希望通过DNN对所有的训练样本计算后，计算结果能够等于或很接近样本输出，当有新的测试样本数据时，能够有效预测样本输出。但怎样找到合适的线形系数矩阵W和偏倚变量b呢?

回顾我们前面学习的机器学习之Logistic回归、机器学习之SVM支持向量机等机器学习算法，很容易联想到，我们可以用一个合适的损失函数来度量训练样本的输出损失。然后对损失函数优化，求损失函数最小化的极值，此时对应的线性系数矩阵W，偏倚变量b便是我们希望得到的结果。深度神经网络中，损失函数优化极值求解的过程，通常是利用梯度下降法迭代完成的。当然也可以利用其他的迭代方法，比如牛顿法或拟牛顿法。梯度下降算法以前在机器学习之线形回归中有过详细介绍，有兴趣可以回顾一下。

对DNN损失函数用梯度下降法进行迭代优化求极小值的过程，便是我们的**反向传播算法(Back Propagation,BP)**。

2.DNN反向传播算法数学推导

进行DNN反向传播算法之前，我们需要选择一个损失函数，来度量计算样本的输出和真实样本之间的损失。但训练时的计算样本输出怎么得到呢?

初始时，我们会随机选择一系列W,b，然后利用神经网络之前向传播算法中介绍到的$a^l=\sigma(z^l)=\sigma(W^la^{l-1}+b^l)$，计算输出层所对应的$a^L$，此时的$a^L$便是DNN计算样本的输出。为专注DNN反向传播算法的推导，我们选择较为简单的损失函数，为此我们使用最常见的均方差来度量损失。

即对于每个样本，我们期望能够最小化下式，其中a^L和y为特征维度的n_out的向量，$||S||_2$为S的L2范数。

$$J(W,b,x,y)=\frac{1}{2}||a^L-y||_2^2$$

通过损失函数，我们能够用梯度下降法来迭代求解每一层的W，b。首先计算的是输出层，其中输出层的W，b满足下式

$$a^L=\sigma(z^L)=\sigma(W^La^{L-1}+b^L)$$

$$J(W,b,x,y)=\frac{1}{2}||a^L-y||_2^2=\frac{1}{2}||\sigma(W^La^{L-1}+b^L)-y||_2^2$$

然后对W^L,b^L分别求偏导，其中符号⊙表示Hadamard积，对于两个维度的向量$A(a_1,a_2,a_3,\ldots,a_n)^T$和$B(b_1,b_2,b_3,\ldots,b_n)^T$，那么$A\odot B=(a_1b_1,a_2b_2,a_3b_3,\ldots,a_nb_n)^T$。之所以使用Hadamard积，是因为我们不了解激活函数的形式，所以用Hadamard积来乘激活函数的导数。另外补充矩阵求导的知识点，其中$\frac{\partial AB}{\partial B}=A^T$。

$$\frac{\partial J(W,b,x,y)}{\partial W^L}=\frac{\partial J(W,b,x,y)}{\partial z^L}\frac{\partial z^L}{\partial W^L}=(a^L-y)\odot\sigma'(z^L)(a^{L-1})^T$$

$$\frac{\partial J(W,b,x,y)}{\partial b^L}=\frac{\partial J(W,b,x,y)}{\partial z^L}\frac{\partial z^L)}{\partial b^L}=(a^L-y)\odot\sigma'(z^L)$$

注意到在求解输出层W，b的时候，有公共部分$\frac{\partial J(W,b,x,y)}{\partial z^L}$，因此我们可以把公共部分先算出来，记为

$$\delta^L=\frac{\partial J(W,b,x,y)}{\partial z^L}=(a^L-y)\odot\sigma'(z^L)$$

现在我们已经把输出层的梯度算出来了，那么如何求解L-1、L-2...层的梯度呢？这里我们需要进一步递推，对于第l层的δ^l可以表示为

$$\delta^l=\frac{\partial J(W,b,x,y)}{\partial z^l}=\frac{\partial J(W,b,x,y)}{\partial z^L}\frac{\partial z^L}{\partial z^{L-1}}\frac{\partial z^{L-1}}{\partial z^{L-2}}\cdots\frac{\partial z^{l+1}}{\partial z^l}$$

如果我们能够计算出第l层的δ^l，那么对于该层的W^l,b^l也会很容易计算。为什么呢?注意到前向传播算法，我们有

$$z^l=W^la^{l-1}+b^l$$

所以根据上式我们可以很方便的计算第l层的W^l,b^l

$$\frac{\partial J(W,b,x,y)}{\partial W^l}=\frac{\partial J(W,b,x,y)}{\partial z^l}\frac{\partial z^l}{\partial W^l}=\delta^l(a^{l-1})^T$$

$$\frac{\partial J(W,b,x,y)}{\partial b^l}=\frac{\partial J(W,b,x,y)}{\partial z^l}\frac{\partial z^l)}{\partial b^l}=\delta^l$$

现在问题关键便是如何求解δ^l。假设我们已经得到第$l+1$层的δ^{l+1}，那么如何得到第l层的δ^l呢？我们注意到

$$\delta^l=\frac{\partial J(W,b,x,y)}{\partial z^l}=\frac{\partial J(W,b,x,y)}{\partial z^{l+1}}\frac{\partial z^{l+1}}{\partial z^l}=\delta^{l+1}\frac{\partial z^{l+1}}{\partial z^l}=$$

$$\frac{\partial(\delta^{l+1})^Tz^{l+1}}{\partial z^l}=\frac{\partial(\delta^{l+1})^T(W^{l+1}\sigma(z^l)+b^{l+1})}{\partial z^l}=\frac{\partial(\delta^{l+1})^TW^{l+1}\sigma(z^l)}{\partial z^l}=$$

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

现在我们已经得到δ^l的递推式，只要我们求出当前隐含层的δ^l，便能够得到W^l,b^l。

3.DNN反向传播算法过程

梯度下降算法有批量(Batch)，小批量(Mini-Batch)，随机三种方式，采用哪种方式取决于我们的问题而定。为简化描述，这里采用最基本的批量梯度下降法来描述反向传播算法。

输入：总层数L、各隐含层与输出层的神经元个数、激活函数、损失函数、迭代步长α、最大迭代次数Max、停止迭代阈值ϵ、输入的m个训练样本$(x_1,y_1),(x_2,y_2),\ldots,(x_m,y_m)$。

输出：各隐含层与输出层的线性关系系数W和偏倚变量b。

- 初始化各隐藏层与输出层的线性关系系数矩阵W和偏倚向量b为随机值。
- $for\ iter=1\ to\ Max$

- $for\ i\ = 1\ to\ m$
  - 将a^1输入值设置为x_i
  - $for\ l = 2\ to\ L$，进行前向传播算法，计算$a^{i,l} = \sigma(z^{i,l}) = \sigma(W^l a^{i,l-1} + b^l)$
  - 通过损失函数计算输出层$\delta^{i,L}$
  - $for\ l = L\ to\ 2$，进行反向传播算法，计算$\delta^{i,l} = (W^{l+1})^T \delta^{i,l+1} \odot \sigma'(z^{i,l})$
- $for\ l = 2\ to\ L$，更新第l层的W^l, b^l
  - $W^l = W^l - \alpha \sum_{i=1}^{m} \delta^{i,l} (a^{i,l-1})^T$
  - $b^l = b^l - \alpha \sum_{i=1}^{m} \delta^{i,l}$
- 如果所有的W,b的变化值都小于停止迭代阈值ε，跳出循环。

- 输出各隐含层和输出层的线形关系系数矩阵W和偏倚向量b。

通过深度神经网络之中的前向传播算法和反向传播算法的结合，我们能够利用DNN模型去解决各种分类或回归问题，但对于不同问题，效果如何呢？是否会过拟合呢？我们将在下次文章中详细介绍损失函数和激活函数的选择、正则化方面的知识点，来让深度神经网络能更精确的解决我们的问题。

参考

> 刘建平Pinard_深度神经网络(DNN)反向传播算法(BP)

4.推广

更多内容请关注公众号**谓之小一**，若有疑问可在公众号后台提问，随时回答，欢迎关注，内容转载请注明出处。

长按关注微信公众号