BP神经网络研究与实现

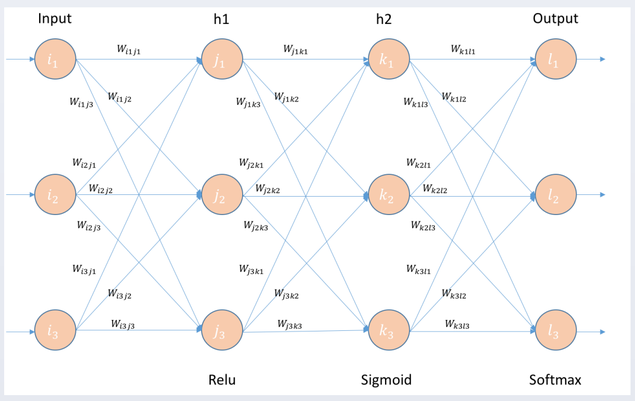
岳家瑞
下一代互联网互联设备国家工程实验室
18120168@bjtu.edu.cn

介绍

人工神经网络（Artificial Neural Network，即ANN ），是20世纪80 年代以来人工智能领域兴起的研究热点。 它从信息处理角度对人脑神经元网络进行抽象， 建立某种简单模型，按不同的连接方式组成不同的网络。

随着近来人工智能行业的火爆越来越多的诸如TensorFlow、Keras等开源库趋近完善、方便。使得越来越少的人关注神经网络底层的算法实现。本研究将深入神经网络最底层，带你领略神经网络的“魅力”

前向传播



一个神经网络包含多个层，每个层又包含多个神经元，前一层的神经元到下一层的神经元之间的连接包含一个权重w，每个神经元又包含一个偏执b。前向传播是输入在神经元中正向传播的过程，就是向量点乘，也就是加权求和，然后经过一个激活函数，例如sigmoid、relu，得到输出作为下一层的输入，逐层迭代，这样一个过程即为前向传播。

后向传播

后向传播过程主要是为了通过计算得到的结果误差反馈给前面的神经网络以减小误差。为了定量化误差，我们需要对误差进性定义，由于此研究为二分类过程，因此使用交叉熵作为误差函数（后称损失函数）。

$$\text{Crossentropy} = \sum_{i=1}^{2} -(y_i) \times \log(O_{outi}) - (1 - y_i) \times \log(1 - O_{outi})$$

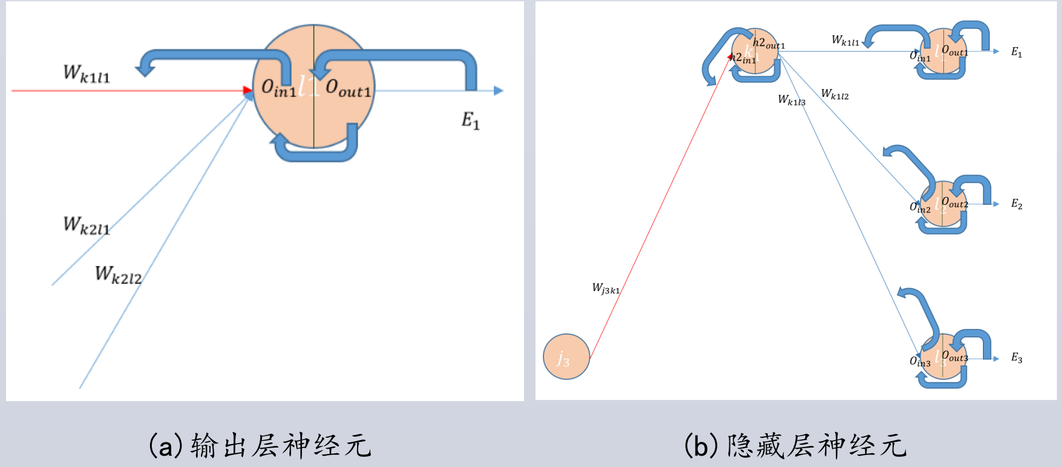
后向传播的目标是调整网络中的参数w、b使得顺势函数更小，对于输出层我们可以通过下面的公式和链式法则：

$$\frac{\partial \text{E1}}{\partial \text{O}_{\text{out1}}} = \frac{\partial(-(y_i)\times\log(O_{outi})-(1-y_i)\times\log(1-O_{outi}))}{\partial \text{O}_{\text{out1}}}$$

$$\frac{\partial \text{O}_{out1}}{\partial \text{O}_{in1}} = \text{O}_{out1} \times (1 - \text{O}_{out1}) \quad \text{(for sigmoid)}$$

$$\frac{\partial \text{O}_{in1}}{\partial W_{k1l1}} = \frac{\partial((hpre_{out1} \times W_{k1l1}) + (hpre_{out1} \times W_{k1l1}) + (hpre_{out1} \times W_{k1l1}))}{\partial W_{k1l1}}$$

$$\frac{\partial \text{E1}}{\partial Wk1l1} = \frac{\partial \text{E1}}{\partial \text{O}_{\text{out1}}} \times \frac{\partial \text{O}_{out1}}{\partial \text{O}_{in1}} \times \frac{\partial \text{O}_{in1}}{\partial W_{k1l1}}$$



(a)输出层神经元　　(b)隐藏层神经元

通过上式即可计算输出层的权重更新值，以使损失函数更小。对于中间的隐藏层还需计算：

$$\frac{\partial \text{E1}}{\partial W_{k1l1}} = \frac{\partial \text{E1}}{\partial \text{O}_{\text{out1}}} \times \frac{\partial \text{O}_{out1}}{\partial \text{O}_{in1}} \times \frac{\partial \text{O}_{in1}}{\partial W_{k1l1}}$$

$$\frac{\partial \text{E1}}{\partial \text{O}_{\text{out1}}} = \frac{\partial \text{E1}}{\partial O_{out1}} \times \frac{\partial O_{out1}}{\partial O_{in1}} \times \frac{\partial O_{in1}}{\partial \text{O}_{\text{out1}}}$$

$$\frac{\partial \text{E2}}{\partial \text{O}_{\text{out1}}} = \frac{\partial \text{E2}}{\partial O_{out2}} \times \frac{\partial O_{out2}}{\partial O_{in2}} \times \frac{\partial O_{in2}}{\partial \text{O}_{\text{out1}}}$$

……

由上式可知可以通过后一层来求得上一层所需要的变量，通过迭代即可求得每一层的W和b的变化量。

结果

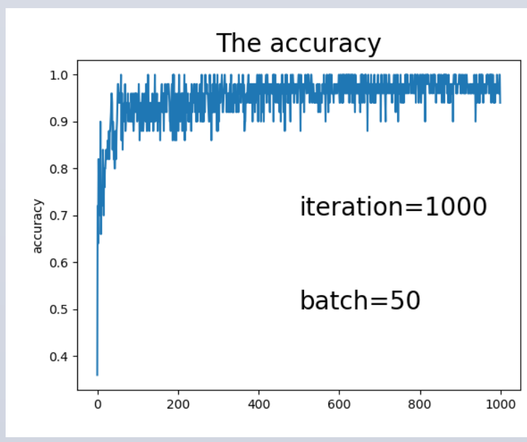
通过上述理论使用c语言进行实现，并创建包含一个输入层、一个隐藏层、一个输出层的神经网络架构。创建数据集进行测试，使用多项式x^2-2 × y^2 + z^2进行二元分类，输入x、y、z，当大于0时取类0，小于0时取类1。

创建50000条数据进行训练，每次训练50条数据并测试准确率和损失，结果如下图：

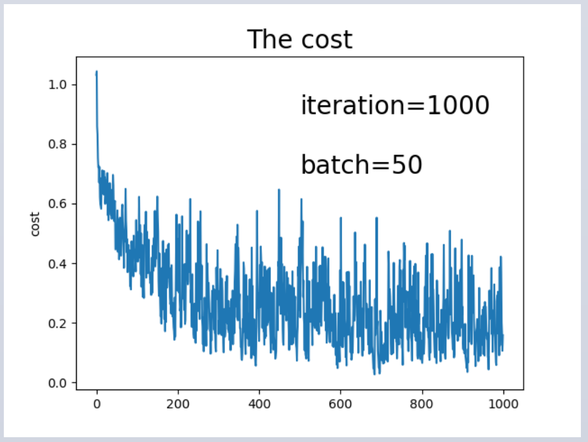
```
**********accuracy**********0.460000 1.367770
**********accuracy**********0.700000 1.369997
**********accuracy**********0.620000 1.369422
**********accuracy**********0.780000 1.426213
**********accuracy**********0.720000 1.314150
**********accuracy**********0.740000 1.299778
**********accuracy**********0.740000 1.429263
**********accuracy**********0.680000 1.235190
**********accuracy**********0.900000 1.492150
**********accuracy**********0.660000 1.085650
**********accuracy**********0.720000 1.042013
**********accuracy**********0.720000 1.137237
**********accuracy**********0.740000 1.130108
**********accuracy**********0.740000 1.568353
**********accuracy**********0.840000 1.410534
**********accuracy**********0.800000 1.494342
**********accuracy**********0.700000 1.454570
**********accuracy**********0.820000 1.231173
**********accuracy**********0.740000 1.147875
**********accuracy**********0.800000 1.537779
**********accuracy**********0.820000 1.305524
**********accuracy**********0.820000 1.373842
**********accuracy**********0.840000 1.050965
**********accuracy**********0.840000 1.204806
**********accuracy**********0.820000 1.489297
**********accuracy**********0.860000 1.187458
**********accuracy**********0.880000 1.195881
**********accuracy**********0.840000 1.561580
**********accuracy**********0.840000 0.921605
**********accuracy**********0.880000 1.125074
**********accuracy**********0.820000 1.010257
**********accuracy**********0.860000 1.164178
**********accuracy**********0.880000 1.254840
**********accuracy**********0.900000 1.490925
**********accuracy**********0.940000 1.150475
**********accuracy**********0.960000 1.173027
```

（c）使用c语言实现神经网络并训练

可以看到准确率有明显上升，损失明显下降，将数据保存并使用python进行绘图，其损失图像使用三点平均使曲线平滑：



（d）准确率图像



（e）损失图像

从结果可以看出本研究有效的实现了BP神经网络，并进行测试，理论严谨，成果合理，为入门神经网络必不可少的一步。

本研究全部代码已上传Github，欢迎访问，仅做学习交流使用，谢谢　https://github.com/saberrey