# Machine Learning---Backpropagation

## 引言

反向传播法(backpropagration)，是一个非常经典的监督学习方法。在前面已经介绍过LMS算法，所以对于一些公共部分可能不会再详细介绍。

## Backpropagation算法基本介绍

### 1.算法思想

反向传播算法是监督式学习最流行的方法。

它其中的算法思想就如它的名字一样。它顺序计算了输出值，在反序将error（计算值和正确值的差值）往回传，进行计算模型中的权值调整。

应用于多层感知器学习算法中，可以解决一些较复杂的问题。

### 2.权值调整公式

和前面介绍的LMS算法一样，这个算法通过一些给定的训练数据进行权值调整。

其中权值调整公式：

其中表示第i个“输入端”的权值，是学习参数，t是正确值，Y是计算值，是第i个“输入端”的输入值。

我们 这里选用这个处理函数是因为它的导数。

这个结果使得导数的计算简单化。

接下来部分是关于这个算法的一些说明：如果没有兴趣的读者可以直接跳过这一段：

首先我们顶一个差值的比方

其中t便是目标值（正确值），y便是计算值，这里去乘于1/2，主要为了之后求导之后的化简。

)，这里n便是输入端的数量，和便是第i个输入端的权值和输入值。

我们这里设。这样y = g()。

按照梯度下降法

其中是一个学习参数，剩下的我们就要求

我们将右边三个分式进行计算：

所以带入公式便是

其实不难发现这个公式在进行一次转换便是上面所提到的

这里我们对这个算法的核心部分进行了简单的数学介绍。关于其中提到的梯度下降算法的知识，笔者已经写过一篇《Machine Learning---LMS 算法数学说明》进行了介绍。

### 3.算法流程

对于这个算法的流程也是比较好理解：

1.首先给权值覆上随机初值；

2.在训练集中随机选择训练组；

3.按照当前计算模型进行计算输出值；

4.将error反向传播；

5.判断MSE是否满足条件，如果不满足继续步骤2;如果满足就跳出算法。

## 算法实现

在这里提供这个算法的实现代码：

struct TrainningSet

{

double inputs[NUM\_INPUTS];

double outputs[NUM\_OUTPUTS];

}TrainningSet;

TrainningSet tests[NUM\_TESTS];

double sigmoid(double val)

{

return ( 1.0 / (1.0 + exp(-val)) );

}

double sigmoid\_d(double val)

{

return val\*(1.0 - val);

}

//

void calculate\_output()

{

for (int i = 0 ; i < NUM\_HIDDENS; ++i)

{

hiddens[i] = 0.0;

for(int j = 0 ; j < NUM\_HIDDENS ; ++j)

{

hiddens[i] += inputs[j] \* w\_h\_i[i][j];

}

hiddens[i] = sigmoid(hiddens[i]);

}

for (int i = 0 ; i < NUM\_OUTPUTS ; ++i)

{

outputs[i] = 0.0;

for(int j = 0 ; j < NUM\_HIDDENS ; ++j)

{

outputs[i] += hiddens[j] \* w\_o\_h[i][j];

}

outputs[i] = sigmoid(outputs[i]);

}

}

void backpropagate\_error(int test)

{

for (int i = 0 ; i < NUM\_OUTPUTS ; ++i)

{

for (int j = 0 ; j < NUM\_HIDDENS ; ++j)

{

w\_o\_h[i][j] = w\_o\_h[i][j] -ALPHA \* (tests[test].outputs[i] - outputs[i]) \* sigmoid\_d(outputs[i]) \* hiddens[j];

}

}

//

for (int i = 0 ; i < NUM\_HIDDENS ; ++i)

{

for (int j = 0 ; j < NUM\_INPUTS ; ++j)

{

w\_h\_i[i][j] = w\_h\_i[i][j] -ALPHA \* (tests[test].hiddens[i] - hiddens[i]) \* sigmoid\_d(hiddens[i]) \* inputs[j];

}

}

}

## 总结

由于笔者不是专门研究人工智能方面，所以在写这些文章的时候，肯定会有一些错误，也请谅解，上面介绍中有什么错误或者不当地方，敬请指出，不甚欢迎。

如果有兴趣的可以留言，一起交流一下算法学习的心得。

声明：本文章是笔者整理资料所得原创文章，如转载需注明出处，谢谢。