# BP_NETWORK

**18307130251 蒋晓雯**

**文档内容（包括但不限于）：**

- **代码基本架构**
- **不同网络结构、网络参数的比较**
- **对反向传播算法的理解**

# 手写体识别

**训练集：12 x 600， 开发集：12 x 20**

## 数据结构

每张图片28 x 28个像素点，输入层需要784个神经元，每一个神经元的输入为0或1，0表示该像素点为白色，1表示该像素点为黑色。

一共有12个汉字，输出层需要12个分类器，输出为0-11是对应的汉字的编号。

除第一层输入层和最后一层输出层，中间层每一层的输出为前一层的输出的加权和（weighted_sum）再过一层sigmoid函数（S形函数）得到sigmoided_weighted_sum；输出层的输出即为前一层的加权和再过一层softmax，无需再过一次sigmoid函数，会弱化分类结果。

$$f(\sigma)=\frac{1}{1+e^{-\sigma}},\ \sigma=\sum w_i x_i+b$$

$$f'(\sigma)=f(\sigma)(1-f(\sigma))$$

**公式1 - sigmoid函数及其导数**

神经网络一共有n层（layer_size = n），一共有1层输入层，一层输出层和n-2层的中间层。

一共需要n-1层的**权重**（weight），第i层权重是神经网络的第i层的输出神经元对应第i+1层每一个神经元的权重，每一层权重是[第i层神经元个数（行）x第i+1层神经元个数（列）]这样的一个矩阵，第n行第m列的值表示第i层的第n个输出（output）到第i+1层的第m个输出需要乘的权重。

**偏移量**（biases）也是n-1层，第i层偏移量是神经网络第i-1层的输出神经元加权后对应第i层每一个神经元需要再附加的一个偏移，每一层是[1x第i+1层神经元个数（列）]这样的一个行向量。

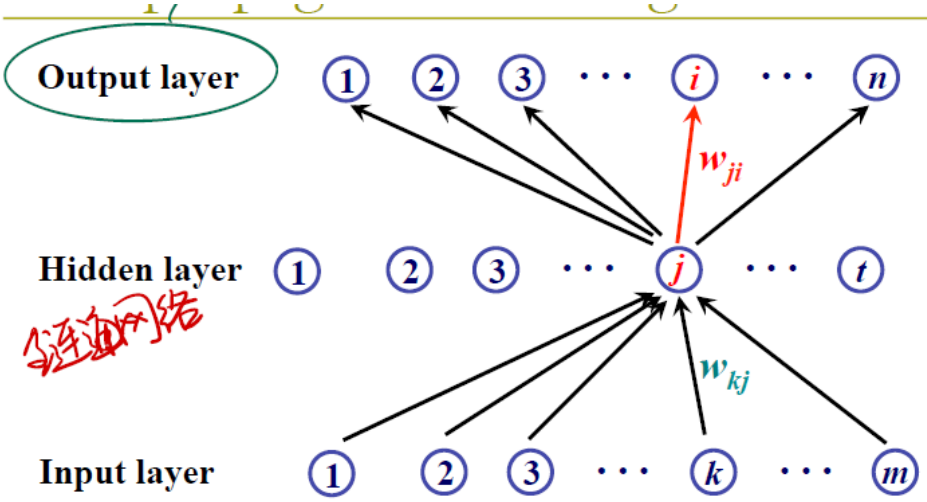
初次的权重和偏移量全部随机生成，正态分布在（-0.05,0.05）区间之内。

第一遍先用当前的权重和偏移把第1层到最后一层，每一层神经元的输出全部存下来。weight_sums中会有n层，第i行的第j个数值对应第i层第j个神经元的输出。

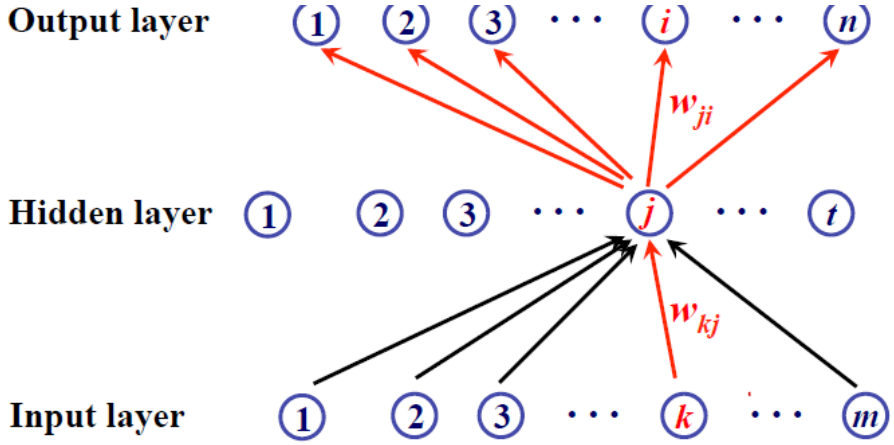
## 反向传播

从输出层一层层向前调整前一层到这一层进行线性加权和的**权重**和附加的**偏移量**。

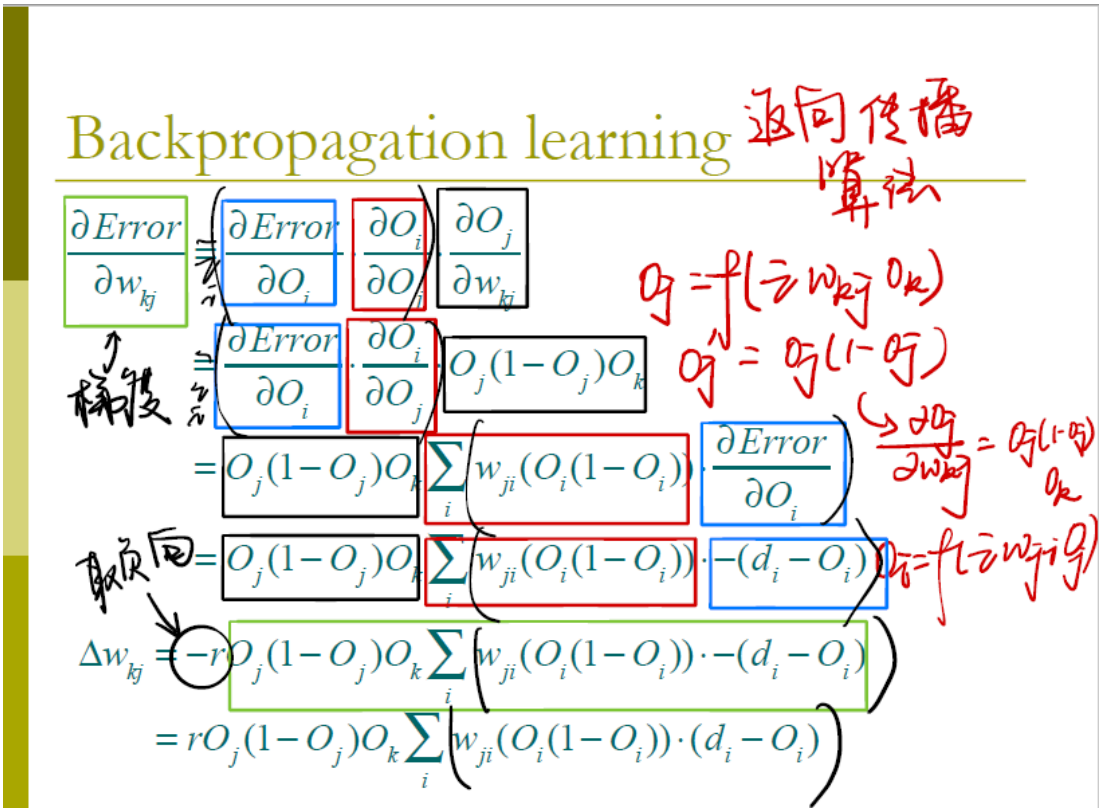
要调整weighted_sums的第i-1层的weighted_sum，即神经网络第i-1层到第i层的weighted_sum，




$$\Delta w_{ji} = r(d_i - O_i)O_i(1-O_i)O_j$$




$$\Delta w_{kj} = rO_j(1-O_j)O_k\sum_i w_{ji}(O_i(1-O_i))\cdot(d_i - O_i)$$




根据梯度下降的原理，可以整理得到每一层的权重调整值都与后面的层权重和神经元的输出（sigmoided_weighted_sum）相关。

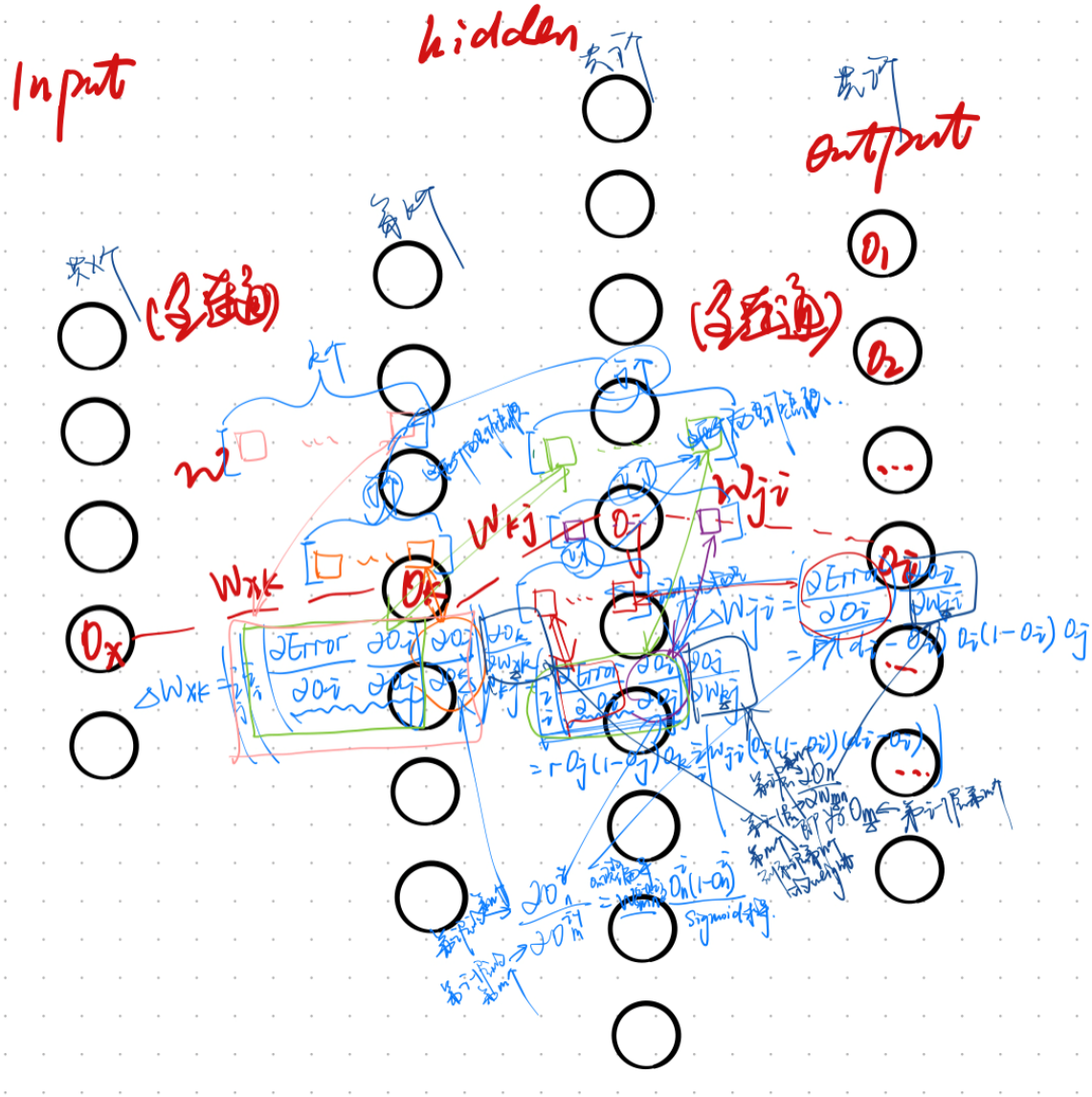



**图1 - 大概的权重调整思路**

一层一层的向前调整，当前一层的调整过程的中间量可以给再前一层使用。

$$Error = -\frac{1}{n}\sum_{j=1}^{n}\sum_{i=0}^{11}\left[G_i^j \ln\left(P_i^j\right) + \left(1-G_i^j\right)\ln\left(1-P_i^j\right)\right] + \frac{\lambda}{2n}\sum_{w}||w||^2$$

$G_i^j$：第$j$个样本第$i$种分类的理想概率，$P_i^j$：第$j$个样本第$i$种分类的实际概率

$\lambda$：正则项系数，$n$：样本个数，$k$：分类的种数（这里是$12$）

**公式2 - 交叉熵和正则项结合的损失函数**

$$\frac{\partial Error}{\partial output^j} = P^j - G^j$$

$G^j$：第$j$个样本理想分类输出，$P^j$：第$j$个样本实际分类输出

**公式3 - 损失函数对输出层求导的结果**

根据梯度下降法可以得到**权重**和**偏移量**的调整量。调整方向为梯度的负向，使得损失函数尽可能减小。其中**正则项系数**控制过拟合程度，使得`scale`尽可能不要太大，产生过拟合现象，把噪声也全都拟合进来不好。

$$\Delta w = -\left(r\frac{\partial ErrorCross}{\partial w} + \frac{r\lambda}{n}w\right)$$

$$\Delta b = -\left(r\frac{\partial ErrorCross}{\partial b}\right)$$

**公式4 - 权重和偏移量的调整量**

## batch

并不是每一个样本输入进来就要调整**权重**和**偏移量**，会使得weight的上下调整幅度很大，不稳定。一个batch个样本得到的**权重**的调整量求平均后对**权重**进行调整后，再进行下一个batch。

## epoch

每把**训练集**（train_set）中的所有样本跑完一次获得了新的**权重**和偏移量，用**开发集**(develop_set)来测试当前**权重**和**偏移量**是否能得到正确的分类，用得到正确分类的次数除以开发集中样本的总个数得到正确率，输出出来看看情况好不好，最后可以用正确率和error画曲线。

## right_rate

分类的正确率，输出层为[1xk]的行向量，每列的值代表该种分类的概率（在[0, 1]之间），分类结果取概率最大的那一列的下标，即为最终分类，如果和理想的分类输出相同即正确分类。

## 实验结果

**蓝线：训练集，红线：开发集**

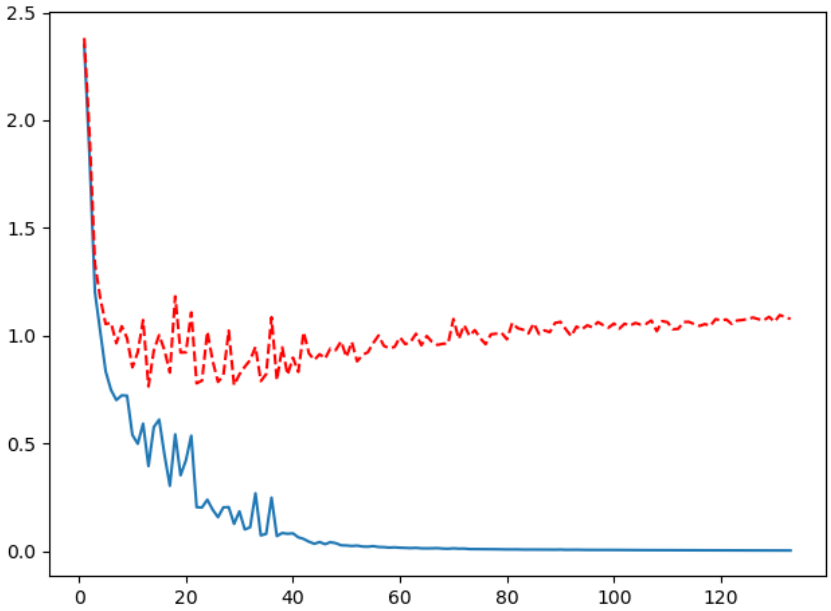



**图2 - 误差**

- 随着迭代次数的增加，**训练集**的误差会逐渐收敛到0；而开发集的误差，在不停迭代后会先下降再缓步上升。
- 可以发现在前40次迭代中，**训练集**一开始误差下降的很快，之后开始进行较为剧烈的波动，之后的error才逐渐趋向平稳；而**开发集**的误差波动直到最后都是保持逐渐减小，但仍不光滑的一个状态。

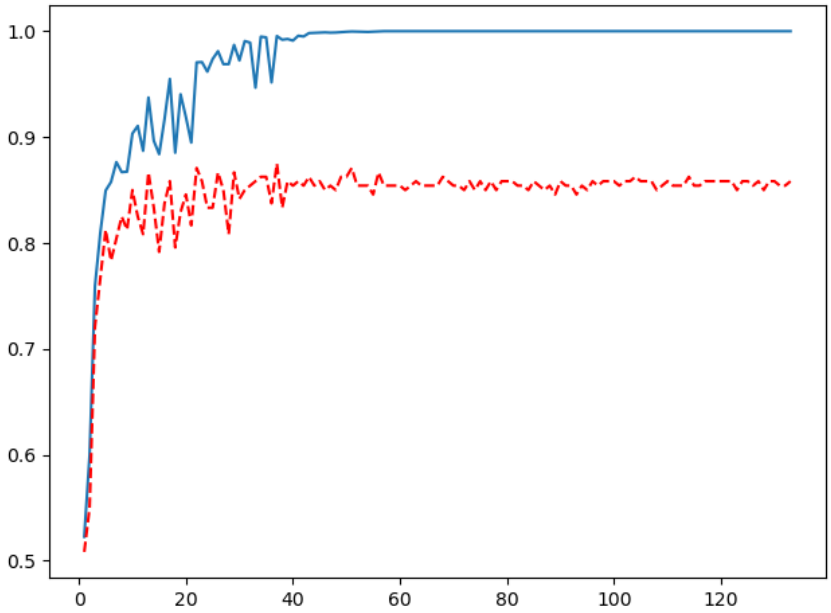



**图3 - 分类正确率**

- 随着迭代次数的增加，**训练集**的正确率会逐渐收敛到100%，而开发集则是收敛到一个低于100%的值，我这里做出的最好结果是正确率最高可达88.75%，但大部分可能的组合，这个正确率都是收敛到87.5%左右。
- 正确率的波动状况和误差的波动状况是相似的。

| index | train_size | develop_size | r | lmda | batch_size | initial | hide_1 | hide_2 | epoch_count | max_develop_right_rate | epoch_index |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 12*500 | 12*120 | 0.01 | 0.001 | 1 | randn(-0.05, 0.05) | 128 | 0 | 83 | 87.08% | 37 |
| 2 | 12*600 | 12*20 | 0.01 | 0.001 | 1 | randn(-0.05, 0.05) | 128 | 0 | 79 | 87.50% | 24 |
| 3 | 12*600 | 12*20 | 0.01 | 0.001 | 1 | randn(-0.05, 0.05) | 90 | 30 | 152 | 88.33% | 69 |
| 4 | 12*600 | 12*20 | 0.01 | 0.001 | 1 | randn(-0.02, 0.02) | 90 | 30 | 300 | 87.92% | 68 |
| 5 | 12*600 | 12*20 | 0.01 | 0.001 | 1 | randn(-0.1, 0.1) | 90 | 30 | 113 | 87.50% | 60 |
| 6 | 12*600 | 12*20 | 0.01 | 0.001 | 1 | randn(-0.05, 0.05) | 99 | 0 | 136 | 88.33% | 52 |
| 7 | 12*600 | 12*20 | 0.01 | 0.001 | 5 | randn(-0.05, 0.05) | 128 | 0 | 300 | 86.25% | 59 |
| 8 | 12*600 | 12*20 | 0.01 | 0.002 | 1 | randn(-0.05, 0.05) | 128 | 0 | 130 | 87.92% | 27 |
| 9 | 12*600 | 12*20 | 0.01 | 0.001 | 1 | randn(-0.05, 0.05) | 18 | 0 | 300 | 86.67% | 33 |
| 10 | 12*600 | 12*20 | 0.01 | 0.002 | 1 | randn(-0.05, 0.05) | 99 | 0 | 143 | 87.50% | 28 |
| 11 | 12*600 | 12*20 | 0.01 | 0.002 | 1 | randn(-0.05, 0.05) | 90 | 30 | 300 | 88.33% | 45 |
| 12 | 12*600 | 12*20 | 0.01 | 0.003 | 1 | randn(-0.05, 0.05) | 99 | 0 | 145 | 87.50% | 34 |
| 13 | 12*600 | 12*20 | 0.05 | 0.001 | 1 | randn(-0.05, 0.05) | 99 | 0 | 300 | 75.00% | 175 |
| 14 | 12*600 | 12*20 | 0.05 | 0.001 | 5 | randn(-0.05, 0.05) | 99 | 0 | 138 | 88.75% | 94 |
| 15 | 12*600 | 12*20 | 0.05 | 0.001 | 5 | randn(-0.05, 0.05) | 90 | 30 | 300 | 87.50% | 66 |
| 16 | 12*600 | 12*20 | 0.05 | 0.001 | 5 | randn(-0.05, 0.05) | 128 | 0 | 125 | 87.08% | 29 |
| 17 | 12*600 | 12*20 | 0.05 | 0.001 | 5 | randn(-0.05, 0.05) | 99 | 0 | 133 | 87.50% | 37 |

**表1 - 手写体识别的实验数据**

**实验记录了总迭代次数，开发集的最高正确率和第一次达到改峰值时是第几次迭代。**

**迭代终止的条件为：最多进行300次迭代，在300次迭代中如果训练集的误差 `<0.005` 时不再进行迭代。**

---

- 只有一层隐藏层，该层神经元的个数过多或是过少，**训练集**误差都无法收敛的很快，需要调整神经元的个数不能过多也不能过少来使得收敛速度达到一个比较快的水平。
- 两层隐藏层和一层隐藏层，总神经元个数相似，两层时**训练集**误差收敛总是更慢。
- **batch_size**增大

1. **学习率**很小时，增大**batch_size**会也会使得收敛变慢，但会减小曲线的波动。最终的正确率并无不同。
2. 增大**学习率**后，如果不增大**batch_size**，收敛速度会减慢；增大**batch_size**可以加快收敛速度。

- 增大**正则项系数**也会使得收敛速度下降。
- 初始化**权重**和**偏移量**距0过近也会使得收敛速度下降。
- **训练集**越小，则需要更多的迭代次数达到收敛。

# sin函数拟合

数据结构和分类器的相同，损失函数不同。

$$Error = \frac{1}{n}\sum_{j=1}^{n}\left|P^j - G^j\right|$$

$n$：总样本个数，$P^j$：第$j$个样本的实际输出，$G^j$：第$j$个样本的理论输出

**公式5 - 损失函数**

权重和**偏移量**的调整量也随之改变。

$$\Delta w = -r\frac{\partial Error}{\partial w}$$

$$\Delta b = -r\frac{\partial Error}{\partial b}$$

**公式6 - 权重和偏移量的调整量**

并且不用再考虑正确率，只要看error就行了。

## 实验结果

**蓝线：训练集，红线：测试集**

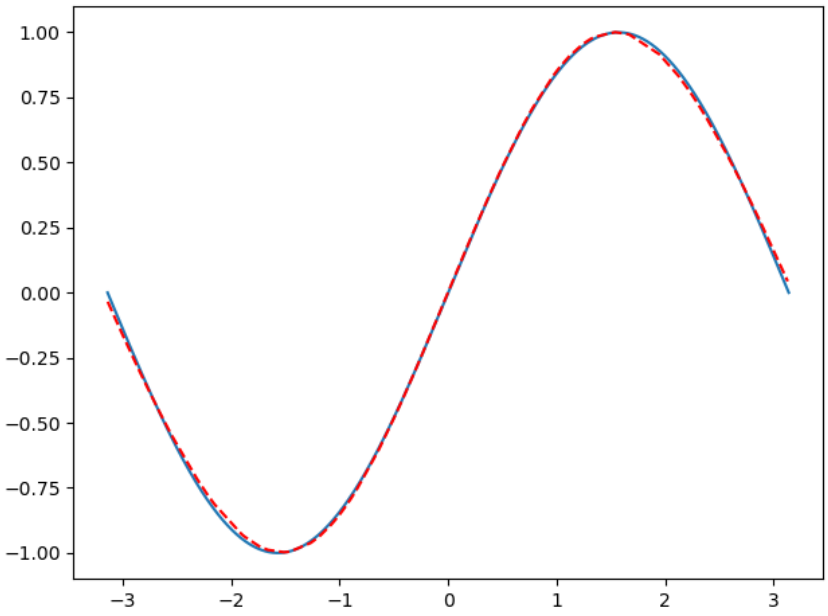



**图3 - 拟合结果**

可以看到拟合的很好。

| index | train_size | test_size | r | batch_size | initial | hide_1 | hide_2 | epoch_count |
|---|---|---|---|---|---|---|---|---|
| 1 | 1000 | 99 | 0.001 | 1 | random(-.0.0005, 0.0005) | 11 | 0 | 1290 |
| 2 | 1000 | 99 | 0.001 | 1 | random(-.0.0005, 0.0005) | 13 | 0 | 708 |
| 3 | 1000 | 99 | 0.001 | 1 | random(-.0.0005, 0.0005) | 18 | 0 | 759 |
| 4 | 1000 | 99 | 0.005 | 1 | random(-.0.0005, 0.0005) | 13 | 0 | 2267 |
| 5 | 1000 | 99 | 0.005 | 5 | random(-.0.0005, 0.0005) | 13 | 0 | 5000+ |
| 6 | 1000 | 99 | 0.0005 | 1 | random(-.0.0005, 0.0005) | 13 | 0 | 5000+ |
| 7 | 1000 | 99 | 0.001 | 1 | random(-.0.0005, 0.0005) | 8 | 3 | 3000+ |
| 8 | 1000 | 99 | 0.001 | 1 | random(-.0.0005, 0.0005) | 4 | 3 | 3000+ |
| 9 | 10000 | 99 | 0.001 | 1 | random(-.0.0005, 0.0005) | 13 | 0 | 920 |

**表2 - sin拟合的实验数据**

**实验记录了总迭代次数。**

**迭代终止的条件为：最多进行3000次迭代，在3000次迭代中如果训练集的误差`<0.001`时不再进行迭代。**

**每一次的测试数据都是随机的，所以相同参数做出来的结果也可能会有差异。**

---

- **学习率**很小，且**batch_size**为1时可以较快收敛，增大**学习率**，或者增大**batch_size**收敛都会变慢。
- 增加隐藏层的层数也会使得误差收敛减慢。
- 改编神经元个数对于收敛的速度的改变是细微的，但可以看出，最好不要过多或者过少，控制在10左右可以达到比较快速的收敛。

# 对反向传播算法的理解

反向传播的思路在上面已经很详细的写了。

每一层权重和偏移量的调整值是根据梯度下降的原理，为了使得误差不断减小而向梯度的负向调整。每一层的调整量和后面每一层的输出相关，可以通过调整使得网络的最终输出更加接近实际输出。