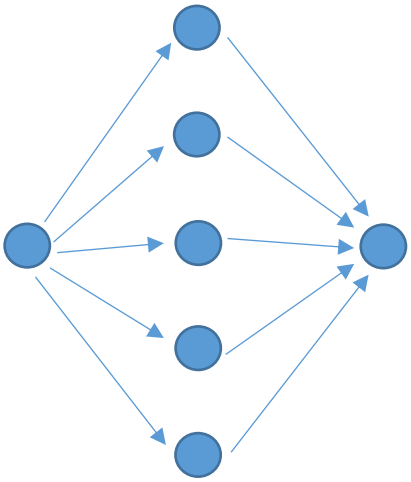
训练一个 1-5-1 的神经网络来逼近函数 sin(x)：(1)由 sin 函数产生 100 个输入-输出对，训练该神经网络使其能根据 x 预测 sin(x)，报告其精度（误差）；(2)报告输入-隐藏层权值及隐藏-输出层权值；(3) 报告作为 x 的函数的隐藏神经元输出 y，报告输出神经元的输出 z，找到每一个隐藏神经元输出函数的分界点，讨论参与产生输出的隐藏神经元。

神经网络 1-5-1 结构设计如下：



input hidden output

隐层神经元采用 sigmoid 作为激活函数。第一层链接的权值为W_{1i} $i = 1,2,..,5$；第二层权值为W_{2i} $i = 1,2,..,5$。

算法主要分为两步，forward 步和 backward 步，分别写在函数 updateOutput 和函数 updategradient.

根据我们以前的推导，有如下权重更新公式：

$$\Delta W_{1i} = (output - target) * W_{2i} * \sigma(h_i) * \big(1 - \sigma(h_i)\big) * input$$

$$\Delta W_{2i} = (output - target) * \sigma(h_i) * input$$

$$W_{1i} = W_{1i} - \alpha \Delta W_{1i}$$

$$W_{2i} = W_{2i} - \alpha \Delta W_{2i}$$

算法参数：

训练集样本： 600

测试集样本： 100

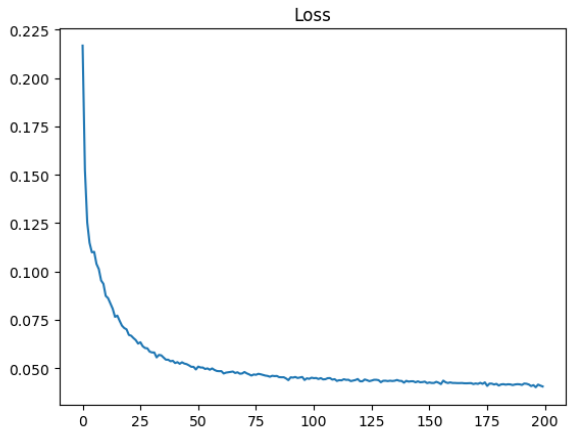
训练轮数： 200

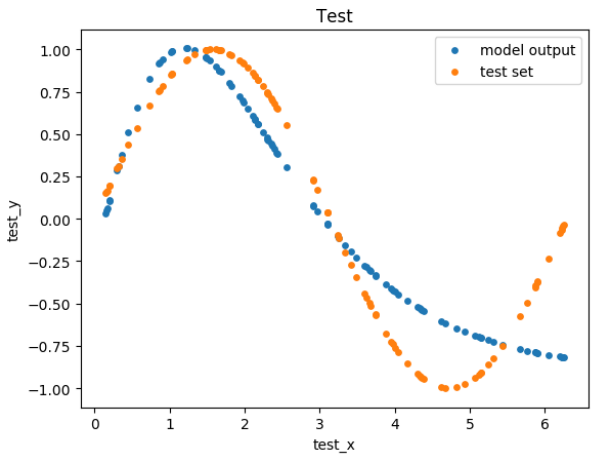
学习率： 0.03

权重随机初始化

采用随机梯度下降，单个样本更新一次

结果如下：





从结果可以看出，经过一定轮数的训练，在(0,2pi)间较好地拟合了 sin(x)函数。

实验期间感悟出一个道理，**参数真的很重要，调参技术也很重要。**

代码见 backpropagation.py。