智能之门

神经网络和深度学习入门

（基于Python的实现）

STEP 5 非线性分类

第 10 章

多入单出的双层神经网络

非线性二分类

在本部分中，我们将学习更复杂的分类问题。很多年前，两位著名的学者证明了感知机无法解决逻辑中的异或问题，从而使感知机这个研究领域陷入了长期的停滞。我们将使用双层网络解决异或问题，并理解其工作原理。

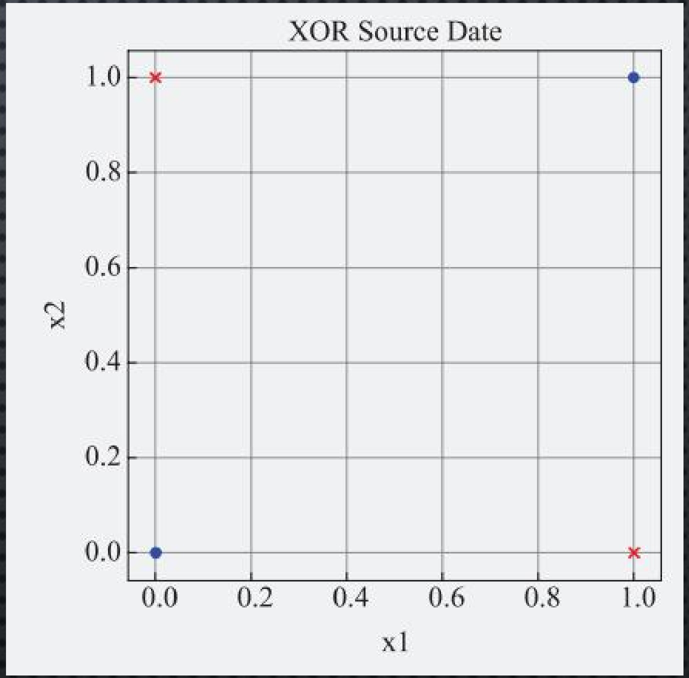
然后我们将会用一个稍微复杂些的二分类例子，来说明在二维平面上，神经网络是怎样通过线性变换加激活函数压缩，把线性不可分问题转化为线性可分问题的。

10.1 非线性二分类问题

➢ 问题一：异或问题

在1969年，一本著名的书《Perceptrons》（感知器，Minsky, Papert,1969）证明了无法使用单层网络（当时称为感知器）来表示最基本的异或逻辑功能，如右图。

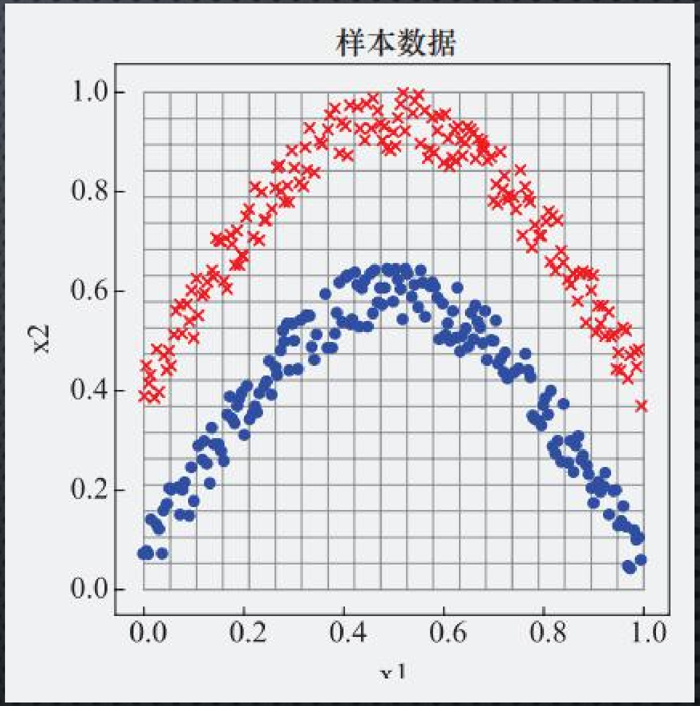
这本书带来了毁灭性的影响，对于感知机这一新生领域的资金支持及兴趣都消失了。



10.1 非线性二分类问题

➢ 问题二：双弧形问题

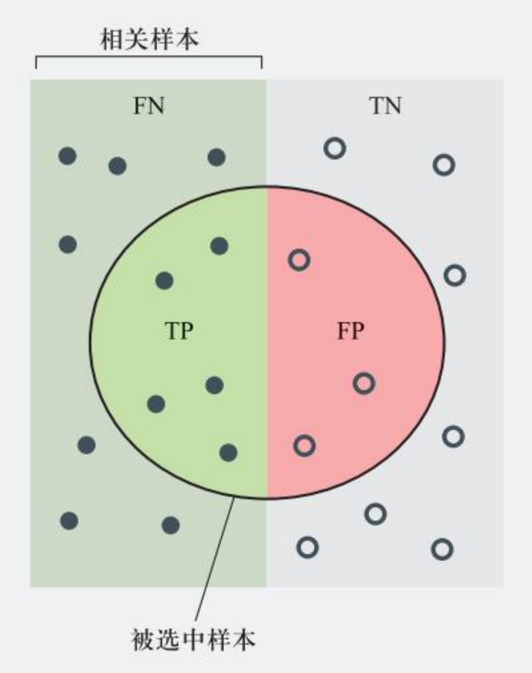
右图展示了一个比异或问题复杂的问题。平面上有两类样本数据，都成弧形分布，由于弧度的存在，使得无法使用一根直线来分开红蓝两种样本点，那么神经网络能用一条曲线来分开它们吗？



10.1 非线性二分类问题

➢ 二分类模型的评估标准

- **准确率**：也可以称之为精度，各类中判别正确的样本数除以总样本数，即为准确率。
- **混淆矩阵**：具体深入到每个类别上，会分成以下4部分来评估。
  - ✓ 正例中被判断为正类的样本数（TP-True Positive）
  - ✓ 正例中被判断为负类的样本数（FN-False Negative）
  - ✓ 负例中被判断为负类的样本数（TN-True Negative）
  - ✓ 负例中被判断为正类的样本数（FP-False Positive）



预测值	被判断为正类	被判断为负类	总和
样本实际为正例	TP	FN	Actual Positive=TP+FN
样本实际为负例	FP	TN	Actual Negative=FP+TN
总和	Predicated Postivie=TP+FP	Predicated Negative=FN+TN	

10.1 非线性二分类问题

- **混淆矩阵可提供的信息**
  - 准确率（Accuracy）

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

  - 精确率/查准率（Precision）

$$Precision = \frac{TP}{TP + FP}$$

  - 召回率/查全率（Recall）

$$Recall = \frac{TP}{TP + FN}$$

- 真正例率（TPR）

$$TPR = \frac{TP}{TP + FN} = Recall$$

- 假正例率（FPR）

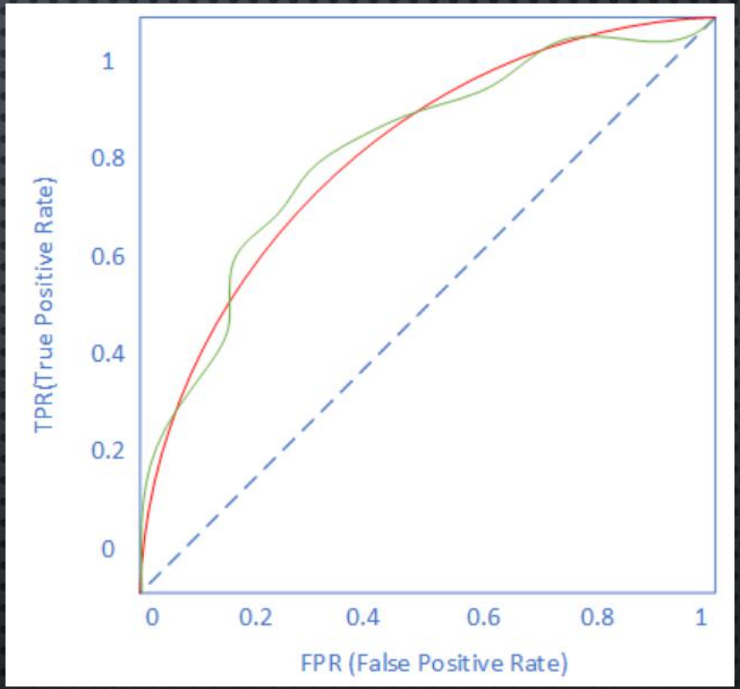
$$FPR = \frac{FP}{FP + TN}$$

- 调和平均值 F1-Score

$$F1 = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}}$$

10.1 非线性二分类问题

- **ROC曲线与AUC值**：接收者操作特征，又称为感受曲线。图中红色的曲线就是ROC曲线，曲线下的面积就是AUC值，区间 [0.5,1.0] 。
  - ✓ ROC曲线越靠近左上角，该分类器的性能越好。
  - ✓ 对角线表示一个随机猜测分类器。
  - ✓ 若一个学习器的ROC曲线被另一个学习器的曲线完全包住，则可判断后者性能优于前者。
  - ✓ 若两个学习器的ROC曲线没有包含关系，则可以判断ROC曲线下的面积，即AUC，谁大谁好。



10.1 非线性二分类问题

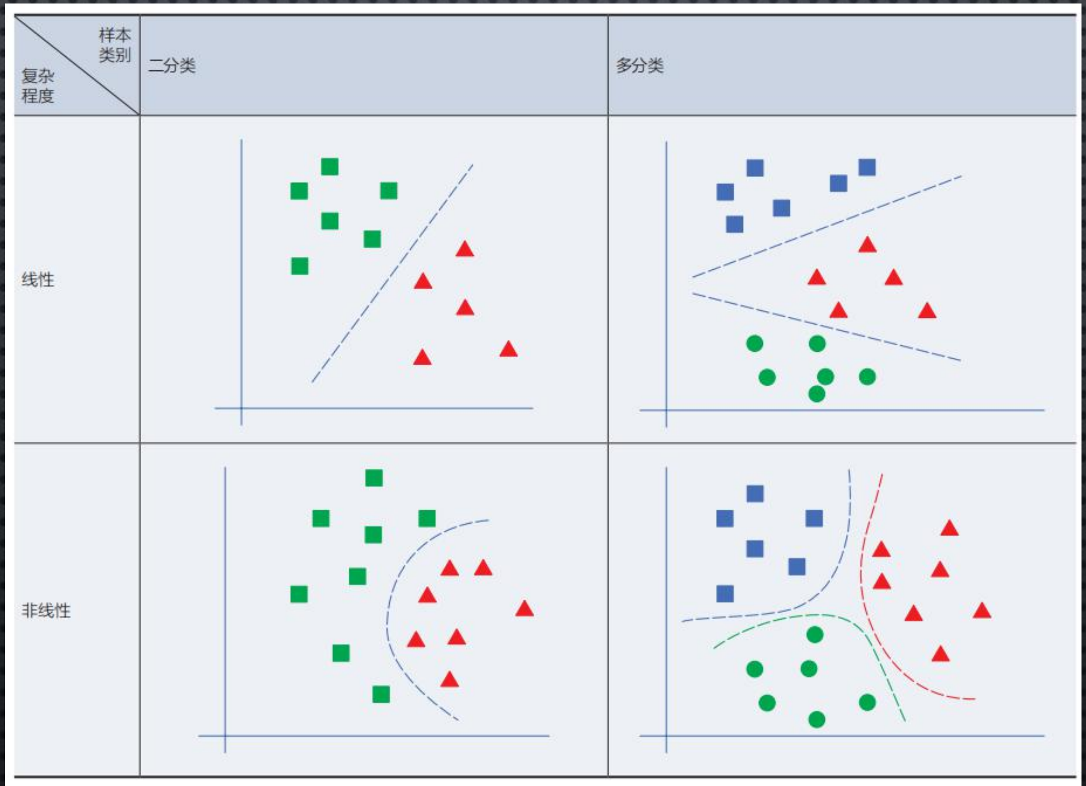
- **Kappa系数**
  - 内部一致性系数，是评价判断的一致性程度的重要指标，取值在 [0,1] 之间。公式如下：

$$Kappa = \frac{p_0 - p_e}{1 - p_e}$$

  - Kappa值大于0.75时，两者一致性较好；
  - Kappa值大于0.4，小于0.75时，两者一致性一般；
  - Kappa值小于0.4时，两者一致性较差。
- **平均绝对误差和均方根误差**
- **相对绝对误差和相对均方根误差**

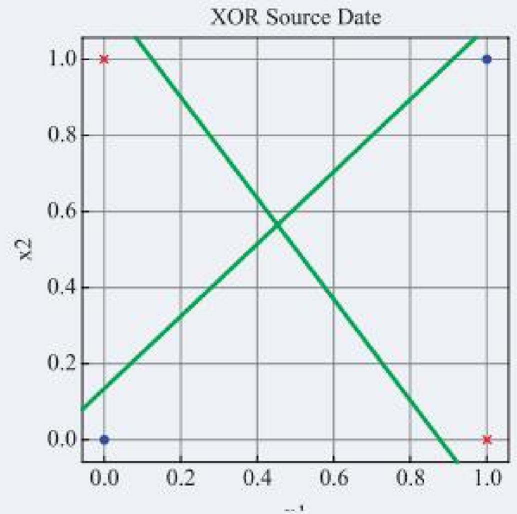
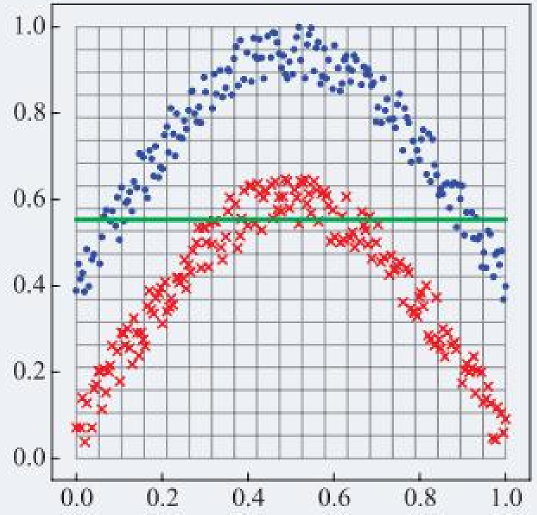
10.2 双层神经网络的意义

- **各种分类任务图示**



10.2 双层神经网络的意义

➢ 在两个任务中应用线性分类

<table>
<tr><th>XOR 问题</th><th>弧形问题</th></tr>
<tr><td>
</td><td>
</td></tr>
<tr><td>图中两根直线中的任何一根，都不可能把蓝色点分到一侧，同时红色点在另一侧</td><td>对于线性技术来说，它已经尽力了，使得两类样本尽可能地分布在直线的两侧</td></tr>
</table>

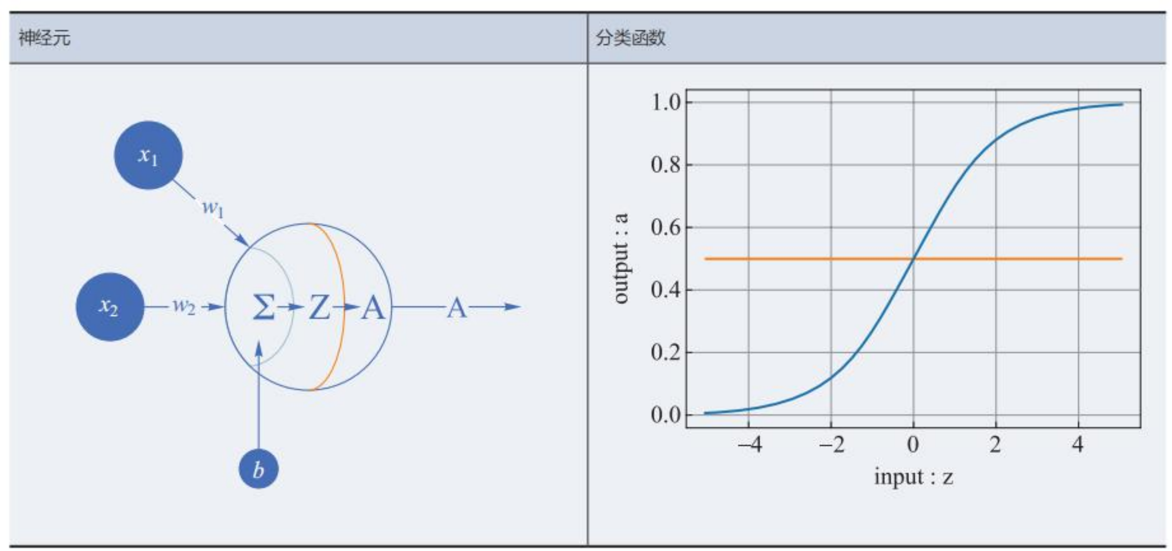
10.2 双层神经网络的意义

- **单层神经网络解决异或问题?**
  - 前向计算公式

    $$z = x_1 w_1 + x_2 w_2 + b$$

    $$a = Logistic(z)$$
  - 四个样本代入上式后化简
    - ✓ $b < 0$
    - ✓ $w_2 + b > 0$
    - ✓ $w_1 + b > 0$
    - ✓ $w_1 + w_2 + b < 0$
  - 以上四式不可能同时成立

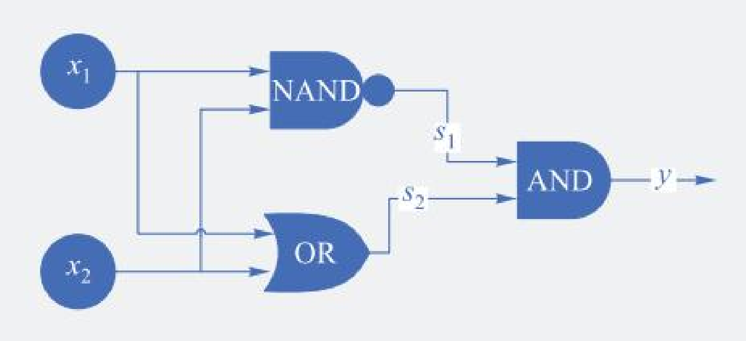
样本	x_1	x_2	y
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	0



10.2 双层神经网络的意义

➢ 非线性的可能性

- 前边讲解过如何实现与、与非、或、或非，用已有的逻辑搭建异或门，如右图所示。
- 实践证明两层逻辑电路可以解决异或问题。



样本与计算	1	2	3	4
x_1	0	0	1	1
x_2	0	1	0	1
$s_1=x_1$ NAND x_2	1	1	1	0
$s_2=x_1$ OR x_1	0	1	1	1
$y=s_1$ AND s_2	0	1	1	0

10.3 非线性二分类实现

➢ 神经网络结构

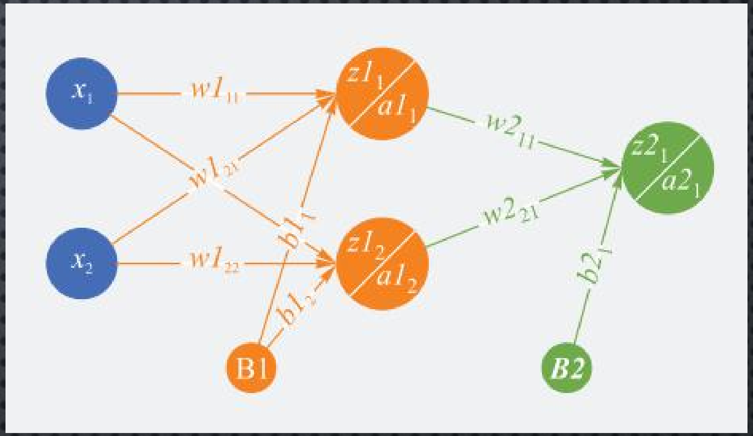
- 输入层：$X = (x_1 \quad x_2)$。
- 隐层权重和偏置：

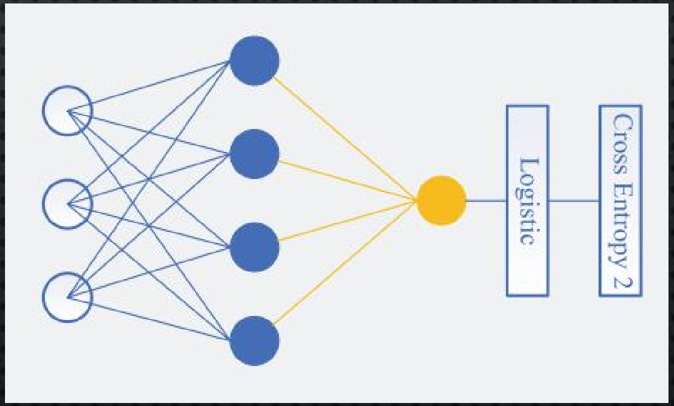
$$W1 = \begin{pmatrix} w1_{11} & w1_{12} \\ w1_{21} & w1_{22} \end{pmatrix}, \qquad B1 = (b1_1 \quad b1_2)$$

- 隐层：$Z1 = (z1_1 \quad z1_2)$，$A1 = (a1_1 \quad a1_2)$。
- 输出层权重和偏置：

$$W2 = \begin{pmatrix} w2_{11} \\ w2_{21} \end{pmatrix}, \qquad B2 = (b2_1)$$

- 输出层：$Z2 = (z2_1)$，$A2 = (a2_1)$。





10.3 非线性二分类实现

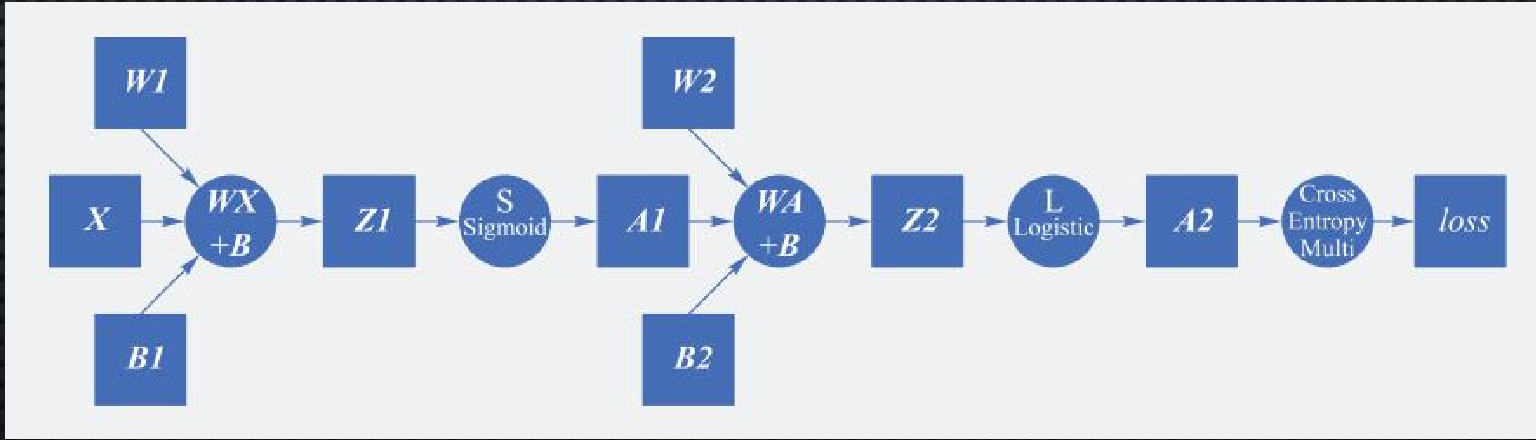
➢ 前向计算

- 层间计算

$$Z1 = X \cdot W1 + B1, \qquad A1 = Sigmoid(Z1), \qquad Z2 = A1 \cdot W2 + B2, \qquad A2 = Logistic(Z2)$$

- 损失函数

$$loss = -[Y \ln A2 + (1 - Y) \ln(1 - A2)]$$



10.3 非线性二分类实现

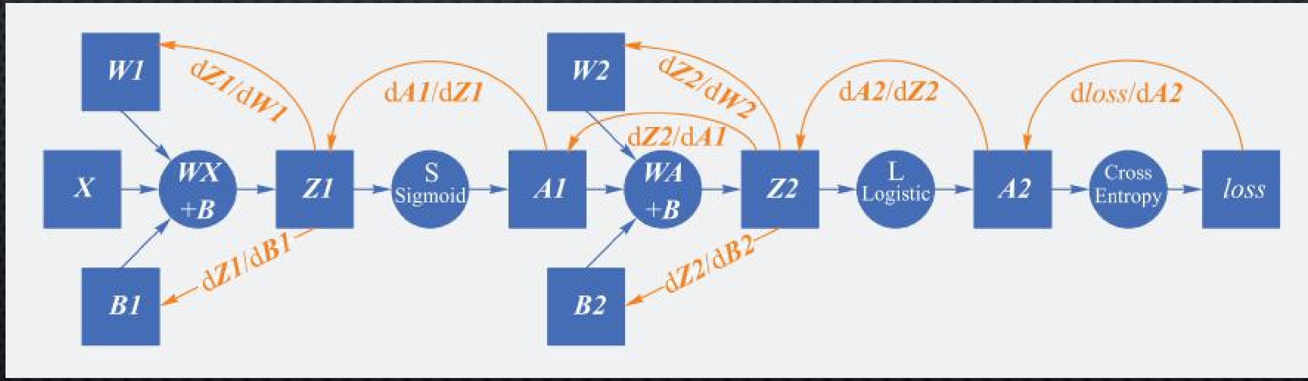
➢ 反向传播

- 链式法则求导，结果类似于此前章节的推导结果：

$$dZ2 = \frac{\partial loss}{\partial Z2} = A2 - Y, \qquad dW2 = \frac{\partial loss}{\partial W2} = A1^T \cdot dZ2, \qquad dB2 = \frac{\partial loss}{\partial B2} = dZ2$$

$$dA1 = \frac{\partial loss}{\partial A1} = dZ2 \cdot W2^T, \qquad dZ1 = \frac{\partial loss}{\partial Z1} = dZ2 \cdot W2^T \odot dA1,$$

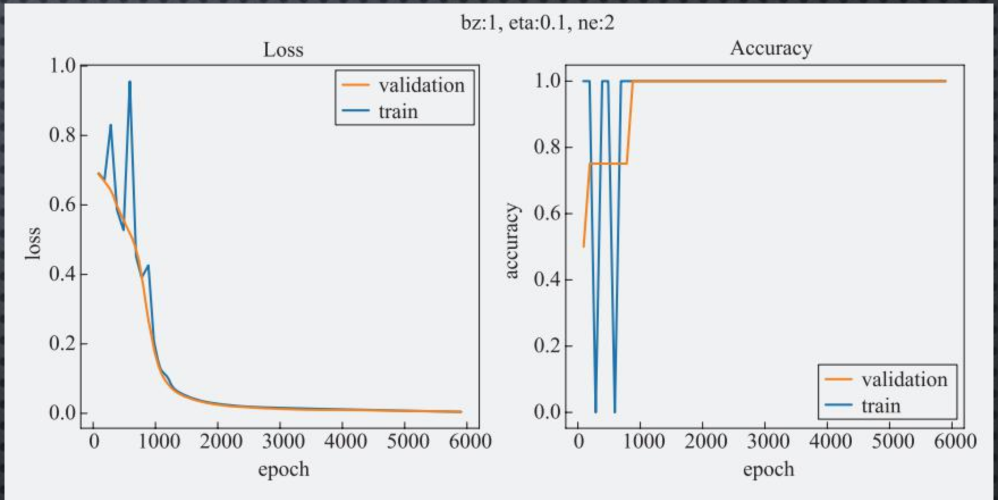
$$dW1 = X^T \cdot dZ1, \qquad dB1 = dZ1$$



10.4 逻辑异或门

➢ 迭代训练结果

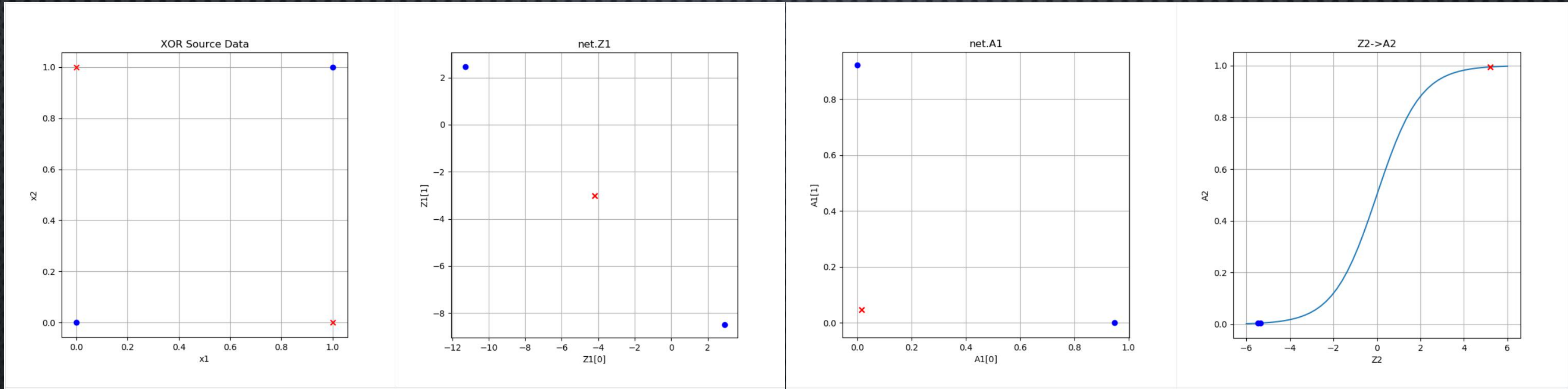
- 表中第四列的推理值与第三列的XOR结果非常的接近，继续训练的话还可以得到更高的精度，但是一般没这个必要了。由此我们再一次认识到，神经网络只可以得到无限接近真实值的近似解。



x_1	x_2	XOR	Inference	diff
0	0	0	0.0041	0.0041
0	1	1	0.9945	0.0055
1	0	1	0.9945	0.0055
1	1	0	0.0047	0.0047

10.4 逻辑异或门

➢ 推理过程可视化



原始样本

Z1是第一层网络线性计算结果

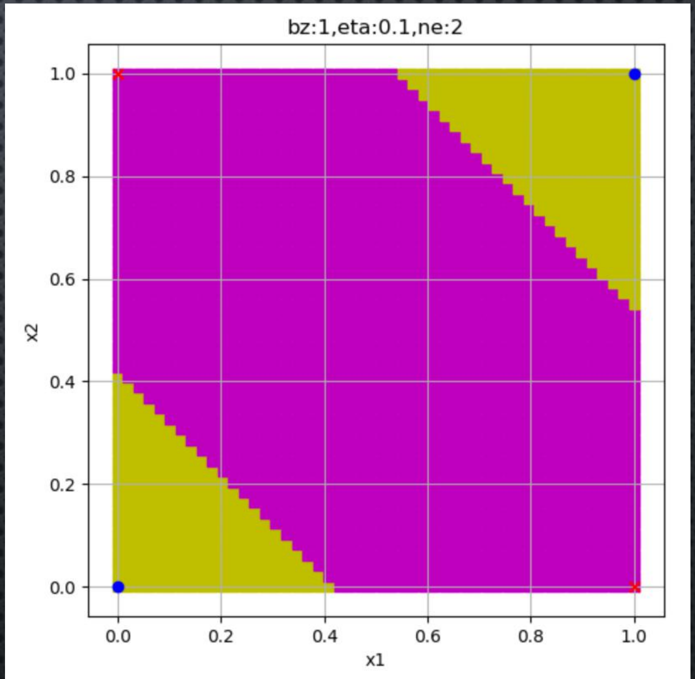
A1是Z1的激活函数计算结果

Z2是第二层线性计算结果，A2是二分类结果

10.4 逻辑异或门

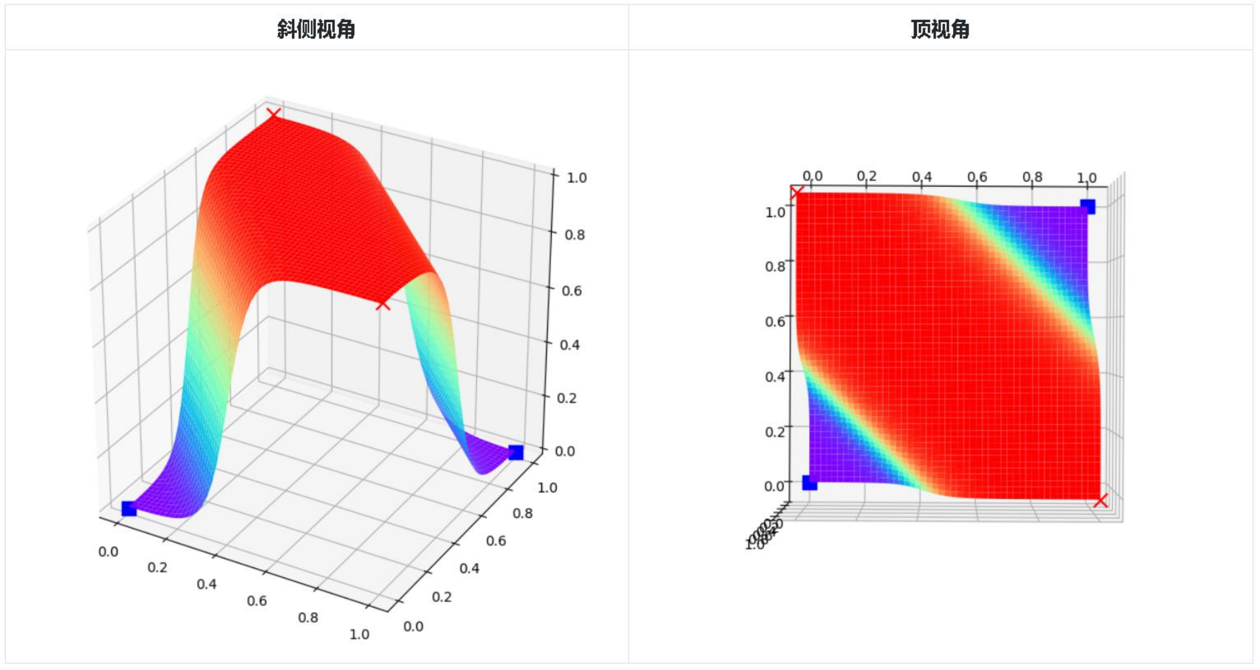
➢ **分类结果可视化**

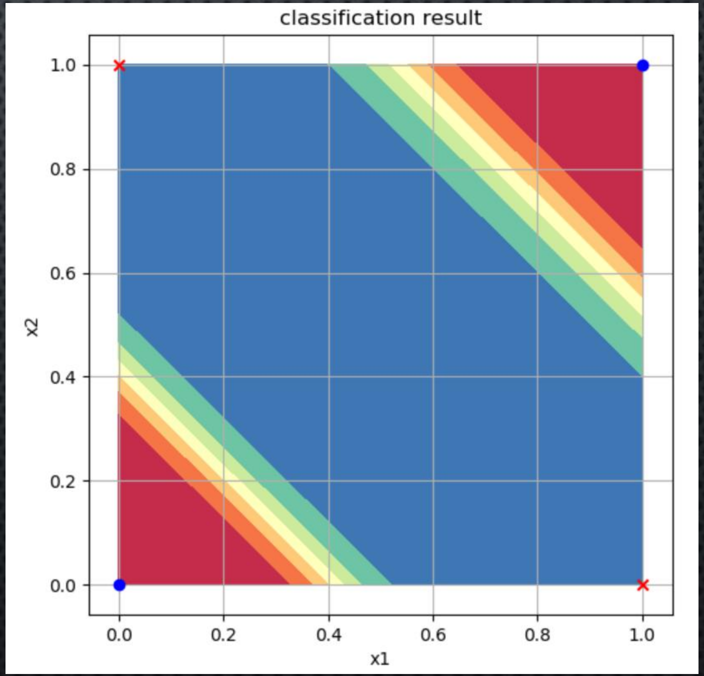
- 请忽略图中的锯齿，因为我们取了50 × 50的网格，所以会有马赛克，如果取更密集的网格点，会缓解这个问题，但是计算速度要慢很多倍。
- 可以看到，两类样本点被分在了不同颜色的区域内，说明神经网络可以同时画两条分割线，更准确的说法是“可以画出两个分类区域”。



10.4 逻辑异或门

- **更直观的可视化**
  - 左图为3D可视化，右图为2.5D可视化。

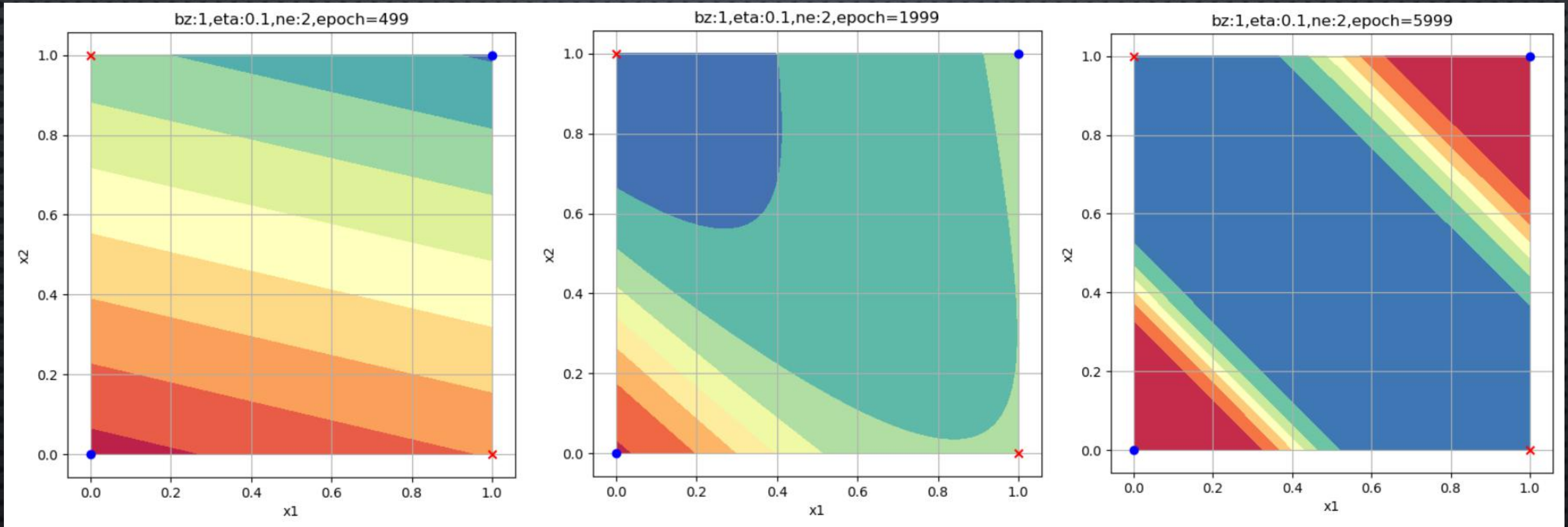




10.4 逻辑异或门

➢ 分类结果的演变

- 以下三图分别展示了迭代次数进行500、2000、6000次之后的结果。



10.4 逻辑异或门

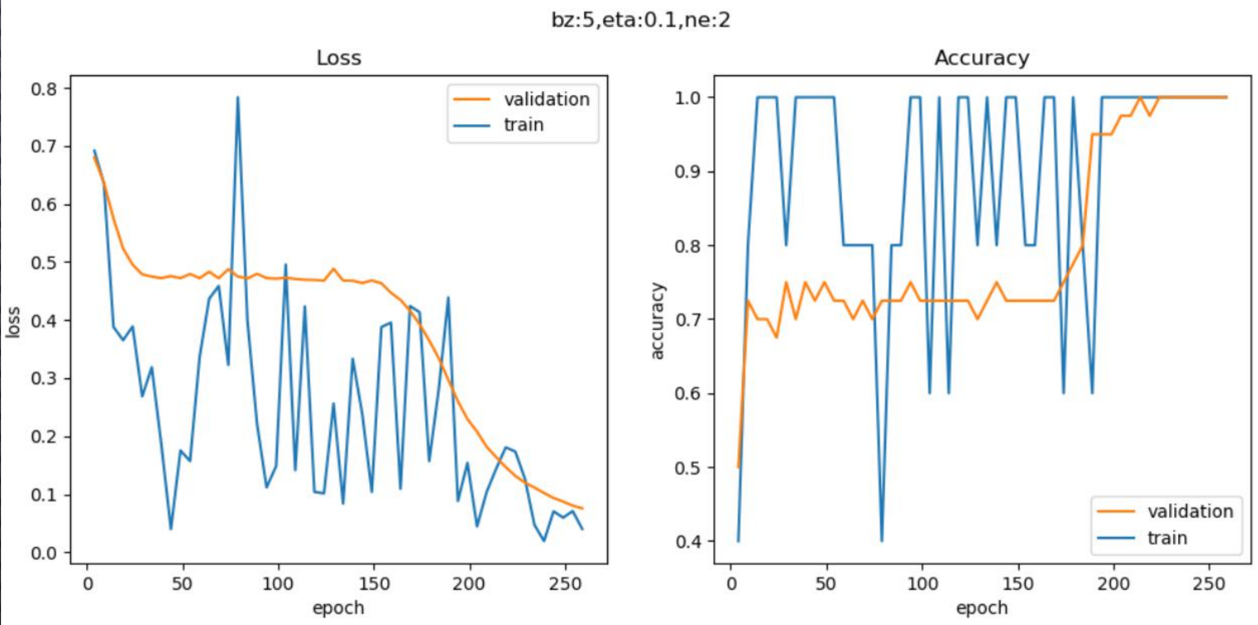
➢ **隐层神经元数量的影响**

- 2个神经元肯定是足够的。
- 4个神经元肯定要轻松一些，用的迭代次数最少。
- 而更多的神经元也并不是更轻松，比如8个神经元，杀鸡用牛刀，由于功能过于强大，出现了曲线的分类边界。
- 而16个神经元更是事倍功半地把4个样本分到了4个区域上，当然这也给了我们一些暗示：神经网络可以做更强大的事情！

10.5 双弧形二分类

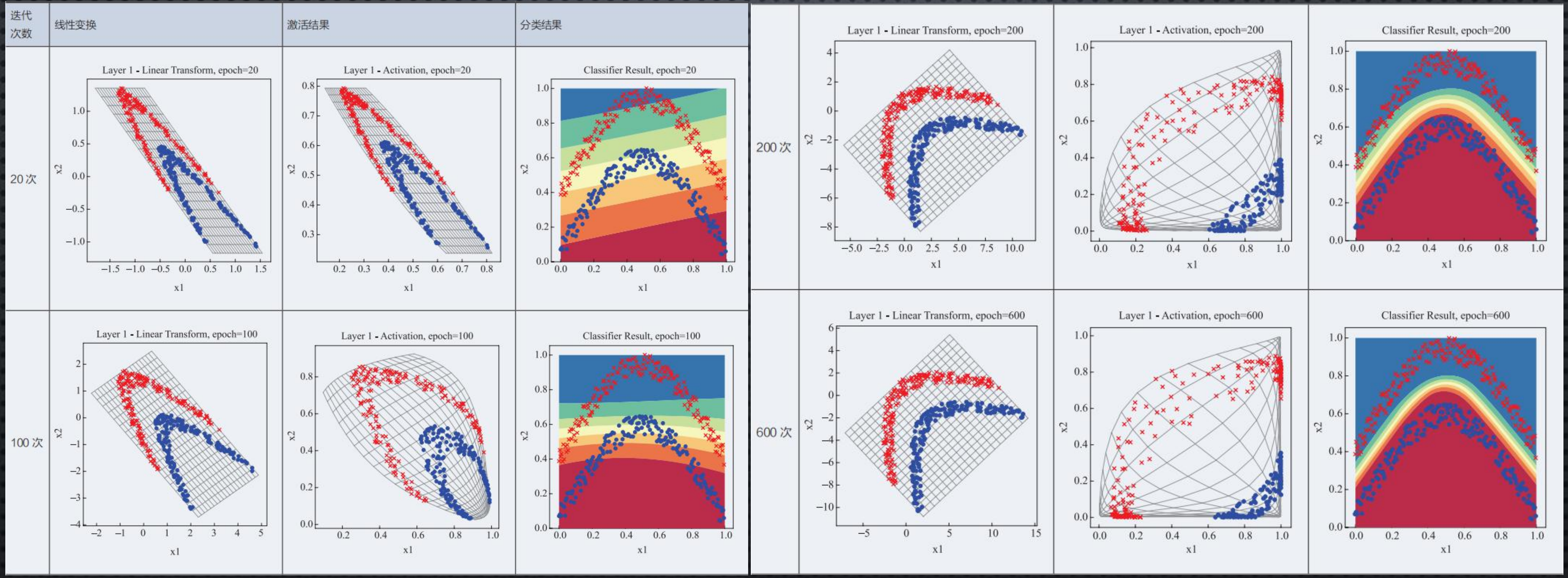
➤ 迭代训练结果

- 蓝色的线条是小批量训练样本的曲线，波动相对较大，不必理会，因为批量小势必会造成波动。红色曲线是验证集的走势，可以看到二者的走势很理想，经过一小段时间的磨合后，从第200个epoch开始，两条曲线都突然找到了突破的方向，然后只用了50个epoch，就迅速达到指定精度。



10.5 双弧形二分类

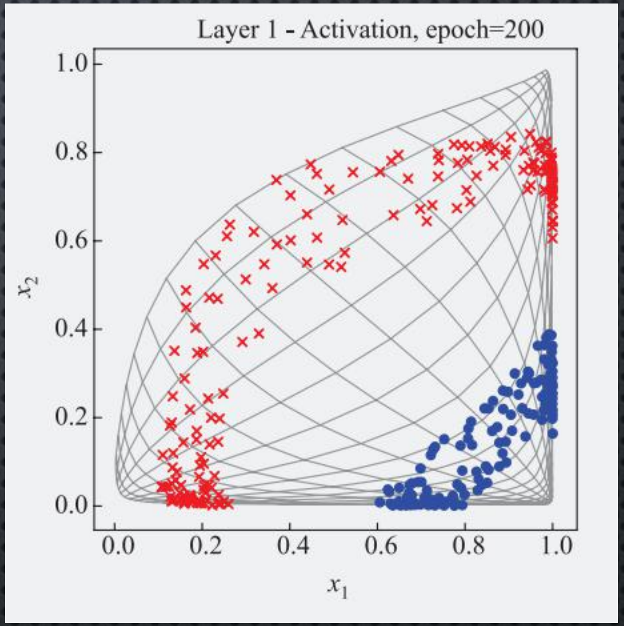
➢ 训练过程可视化



10.5 双弧形二分类

➢ 结果分析

- 在第一层的线性变换中，原始样本被斜侧拉伸，角度渐渐左倾到40度，并且样本间距也逐渐拉大，原始样本归一化后在[0,1]之间，最后已经拉到了[−5,15]的范围。这种侧向拉伸实际上是为激活函数做准备。
- 在激活函数计算中，由于激活函数的非线性，所以空间逐渐扭曲变形，使得红色样本点逐步向右下角移动，并变得稠密；而蓝色样本点逐步向左上方扩撒，相信它的极限一定是[0,1]空间的左边界和上边界；另外一个值得重点说明的就是，通过空间扭曲，红蓝两类之间可以用一条直线分割了！这是一件非常神奇的事情。
- 最后的分类结果，从毫无头绪到慢慢向上拱起，然后是宽而模糊的分类边界，最后形成非常锋利的边界。



➢ 神经网络通过空间变换的方式，把线性不可分的样本变成了线性可分的样本，从而让最后的分类变得很容易。

THE END

谢谢！