PYTHON神经网络编程笔记:

1.TIP1当我们不能精确知道一些事情如何运作时，我们可以尝试使用模型来估计其运作方式，在 模型中，包括了我们可以调整的参数。如果我们不知道如何将千米转换为英里，那么我们 可以使用线性函数作为模型，并使用可调节的梯度值作为参数。

2.TIP2改进这些模型的一种好方法是，基于模型和已知真实示例之间的比较，得到模型偏移的误 差值，调整参数

3，接受了一个输入，并做出应有的预测，输出结果，所以我们将其称为预测器。根据结果与已知真实示例进行比较所得到的误差，调整内部参数，使预测更加精确。分类器！！！

4，误差值E为 误差值=（期望目标值-实际输出值）E = (ΔA)x。根据误差值E，我们希望知道需要将A调整多少，才能改进直线的斜 率，得到一台更好的分类器。要做到这一点，我们只要重新调整上一个方 程，将 ΔA算出： ΔA= E / x我们可以使用误 差值E，将所得到的ΔA作为调整分界线斜率A的量。新的A值=ΔA+A旧。但这样，最终改进的直线不会顾及所有先前的训练样本，而是抛弃了所有先前训练样本的学习结果，只是对最近的一个实例进行了学 习。

5.=》

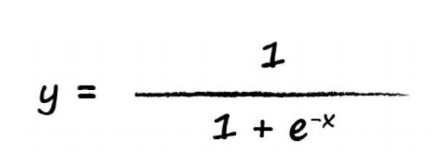
6，TIP3我们使用简单的数学，理解了线性分类器输出误差值和可调节斜率参数之间的关系。也就 是说，我们知道了在何种程度上调整斜率，可以消除输出误差值。 使用朴素的调整方法会出现一个问题，即改进后的模型只与最后一次训练样本最匹 配，“有效地”忽略了所有以前的训练样本。解决这个问题的一种好方法是使用学习率，调 节改进速率，这样单一的训练样本就不能主导整个学习过程。 来自真实世界的训练样本可能充满噪声或包含错误。适度更新有助于限制这些错误样本的 影响。

7，布尔逻辑函数通常需要两个输入，并输出一个答案。（AND&OR）数字1表示真，使用数字0表示假。布尔函数称为XOR，这是eXclusive OR（异或）的缩写， 这种函数只有在A或B仅有一个为真但两个输入不同时为真的情况下，才输出为真

8，我们可以使用多个分类器一起工 作。这是神经网络的核心思想。你可以想象，多条直线可以分离出异常形 状的区域，对各个区域进行分类

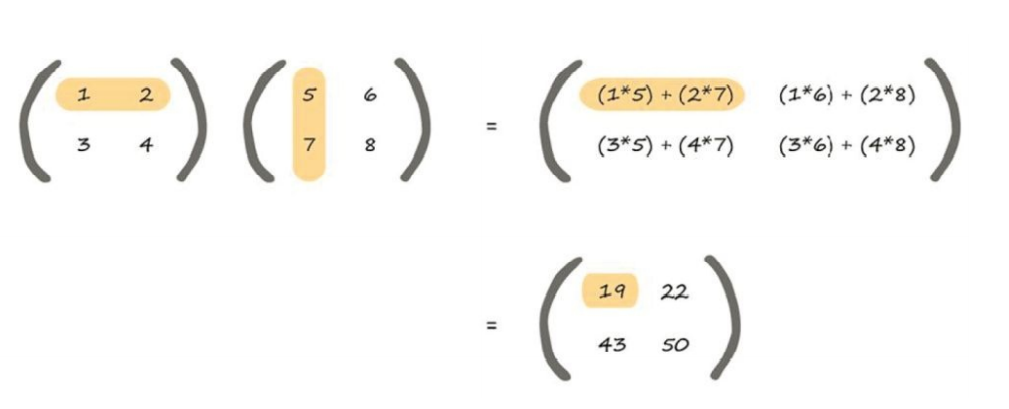
9，TIP4如果数据本身不是由单一线性过程支配，那么一个简单的线性分类器不能对数据进行划 分。例如，由逻辑XOR运算符支配的数据说明了这一点。 但是解决方案很容易，你只需要使用多个线性分类器来划分由单一直线无法分离的数据。

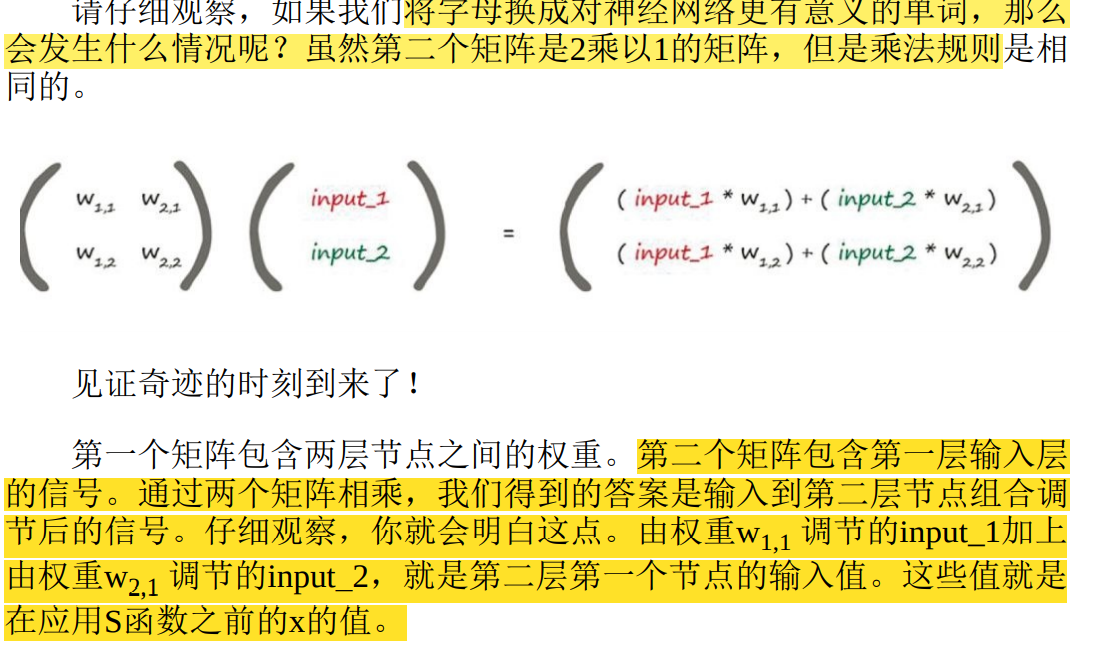
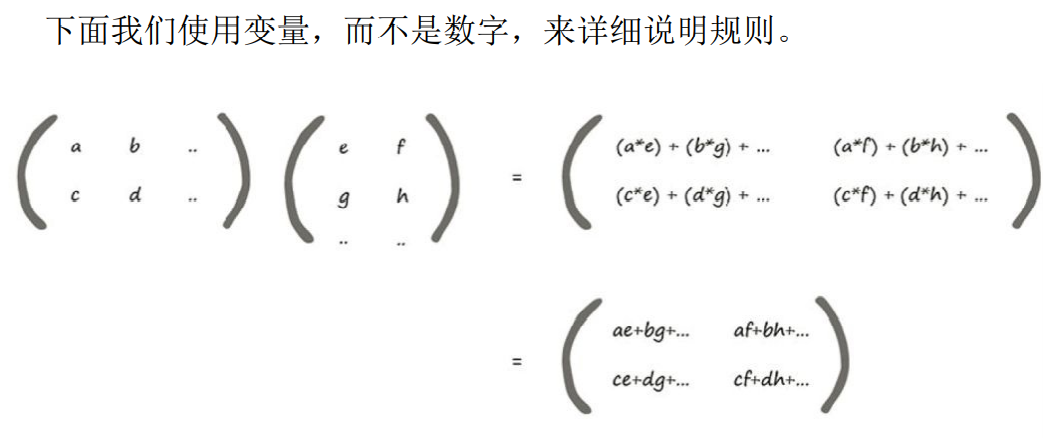
10，S函数：



11，神经网络通过调整优化网络内部的链接权重改进输出，一些权重 可能会变为零或接近于零。零或几乎为零的权重意味着这些链接对网络的 贡献为零，因为没有传递信号。零权重意味着信号乘以零，结果得到零， 因此这个链接实际上是被断开了。

12，得S函数y = 1 /（1 + e -x ）吗？这个函数中的x 表示一个节点的组合输入。此处组合的是所连接的前一层中的原始输出， 但是这些输出得到了链接权重的调节。

13，矩阵允许我们简洁、方便地表示我们所需的工作，同时计算机 可以快速高效地完成计算。第一个数字代表行，第二个数字代表列，这是约定



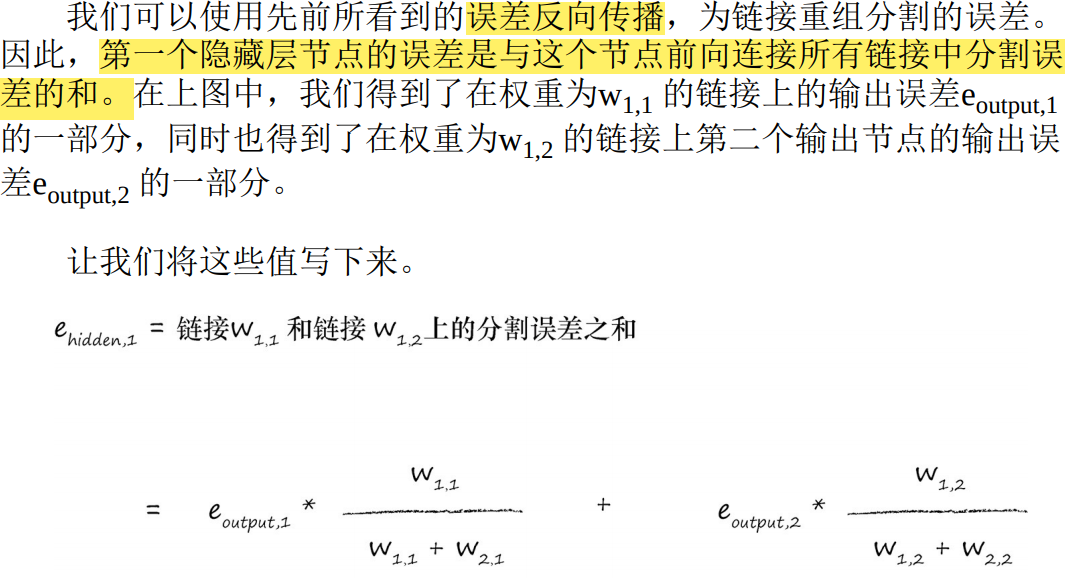
14，我们可以使用矩阵乘法表示所有计算，计算出组合调 节后的信号x ，输入到第二层的节点中。我们可以使用下式，非常简洁地 表示： X = W •I 此处，W 是权重矩阵，I 是输入矩阵，X 是组合调节后的信号，即输 入到第二层的结果矩阵。矩阵通常使用斜体显示，表示它们是矩阵，而不 是单个数字。

15，来自第二层的最终输出是： O = S( X ) 斜体的O 代表矩阵，这个矩阵包含了来自神经网络的最后一层中的所 有输出。

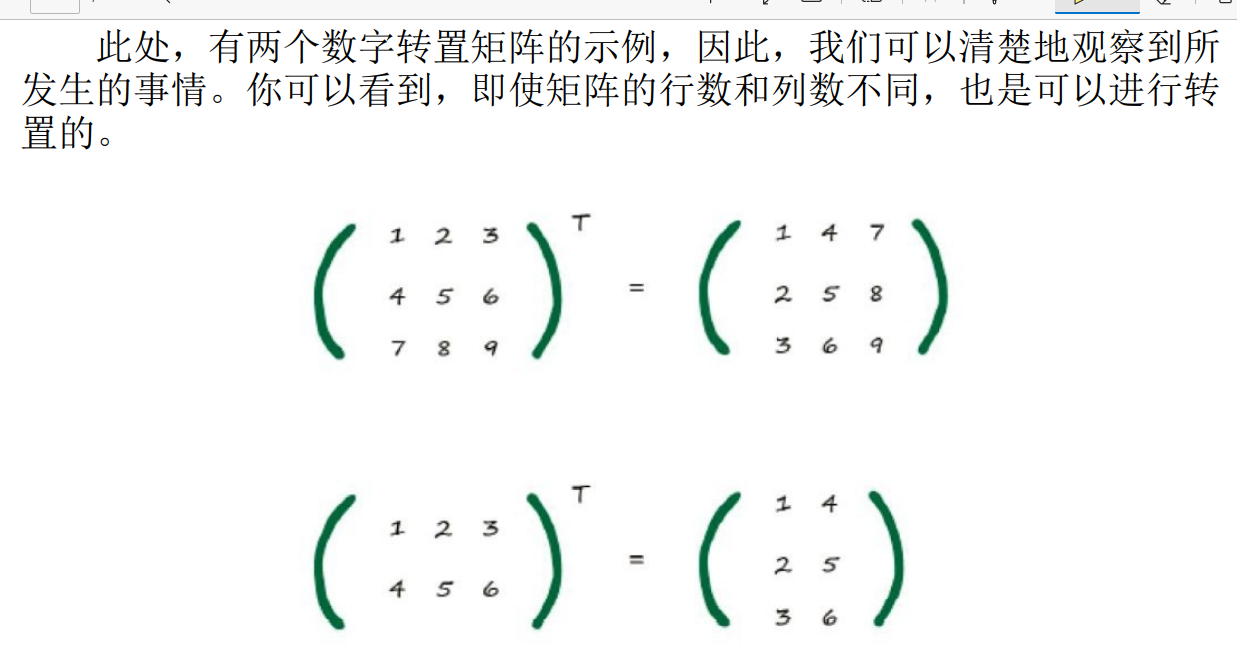
**16，**第一 层为输入层，最后一层为输出层，中间层我们称之为隐藏层。W input\_hidden 是输入层和隐藏层之间的权重。我们需要另一个权重矩阵来 表示隐藏层和输出层之间的链接，这个矩阵我们称之为W hidden\_output 。

17，下一步，将神经网络的输出值与训练样本中的输出值进行比较，计算 出误差。我们需要使用这个误差值来调整神经网络本身，进而改进神经网 络的输出值。不等分误差。与前一种思想相反，我们为较大链接权重 的连接分配更多的误差。

18，我们需要隐藏层节点的误差，这样我们就可以使用这个误 差更新前一层中的权重。我们称这个误差为ehidden 。但是，我们不需要明 确地回答这些误差是什么。我们的训练样本数据只给出了最终输出节点的 目标值，因此不能说这个误差等于中间层节点所需目标输出值与实际输出 值之间的差。

19

20，TIPS：神经网络通过调整链接权重进行学习。这种方法由误差引导，误差就是训练数据所给出正 确答案和实际输出之间的差值。 简单地说，在输出节点处的误差等于所需值与实际值之间的差值。 然而，与内部节点相关联的误差并不显而易见。一种方法是按照链路权重的比例来分割输 出层的误差，然后在每个内部节点处重组这些误差

21，较大的权重就意味着携带较多的输出误差给隐藏层。 这是非常重要的一点。这些分数的分母是一种归一化因子。如果我们忽略 了这个因子，那么我们仅仅失去后馈误差的大小。也就是说，我们使用简 单得多的e1 \* w1,1 来代替e1 \* w1,1 / ( w1,1 + w2,1 )。是这个矩阵沿对角线进 行了翻转，因此现在右上方的元素变成了左下方的元素，左下方的元素变 成了左上方的元素。我们称此为转置矩阵，记为wT 。

22，TIP：反向传播误差可以表示为矩阵乘法。 无论网络规模大小，这使我们能够简洁地表达反向传播误差，同时也允许理解矩阵计算的 计算机语言更高效、更快速地完成工作。 这意味着前向馈送信号和反向传播误差都可以使用矩阵计算而变得高效。

**23，**我们使用误差来指导如何调整链接权重，从而改 进神经网络输出的总体答案。这基本上就是我们使用本书先前所讨论的线 性分类器所做的事情。

24，向真实的 最小值靠近了一些。我们可以继续这样操作，直到几乎不能改进为止，这 样我们就确信已经到达了最小值。 我们要改变步子大小，避免超调，这样就会避免在最小值的地方来回 反弹，这是一个必要的优化。你可以想象，如果我们距离真正的最小值只 有0.5米，但是采用2米的步长，那么由于向最小值的方向走的每一步都超 过了最小值，我们就会错过最小值。如果我们调节步长，与梯度的大小成 比例，那么在接近最小值时，我们就可以采用小步长。这一假设的基础 是，当我们接近最小值时，斜率也变得平缓了。对于大多数光滑的连续函 数，这个假设是合适的。但是对于有时突然一跃而起、有时突然急剧下降 的锯齿函数而言，也就是说存在数学家所说的间断点，这不是一个合适的 假设。

**25,** 为了避免终止于错误的山谷或错误的函数最小值，我们从山上的不同 点开始，多次训练神经网络，确保并不总是终止于错误的山谷。不同的起 始点意味着选择不同的起始参数，在神经网络的情况下，这意味着选择不 同的起始链接权重

26,TIP: 梯度下降法是求解函数最小值的一种很好的办法，当函数非常复杂困难，并且不能轻易使 用数学代数求解函数时，这种方法却发挥了很好的作用。 更重要的是，当函数有很多参数，一些其他方法不切实际，或者会得出错误答案，这种方 法依然可以适用。 这种方法也具有弹性，可以容忍不完善的数据，如果我们不能完美地描述函数，或我们偶 尔意外地走错了一步，也不会错得离谱。

27, 第三种选择是差的平方，即（目标值-实际值）2 。我们更喜欢使用第 三种误差函数，而不喜欢使用第二种误差函数，原因有以下几点： 使用误差的平方，我们可以很容易使用代数计算出梯度下降的斜率。 误差函数平滑连续，这使得梯度下降法很好地发挥作用——没有间 断，也没有突然的跳跃。 越接近最小值，梯度越小，这意味着，如果我们使用这个函数调节步 长，超调的风险就会变得较小。

28, 我们可以在一个节点传入链接数量平方根倒数的大致范围内 随机采样，初始化权重。因此，如果每个节点具有3条传入链接，那么初始 权重的范围应该在从 到 ，即±0.577之间。如果每个节点具有 100条传入链接，那么权重的范围应该在 至 ，即±0.1之 间。

29，不管你做什么，禁止将初始权重设定为相同的恒定值，特别是禁止将 初始权重设定为0。要不然，事情会变得很糟糕。由于正确训练的网络应该具有不等的权重（对于几乎所有的问题， 这是极有可能的情况），那么由于这种对称性，你将永远得不到这种网 络，因此这是一种很糟糕的情况。由于0权重，输入信号归零，取决于输入信号的权重更新函数也因此归 零，这种情况更糟糕。网络完全丧失了更新权重的能力。

30，TIPS：一个常见的问题是饱和。在这个时候，大信号（这有时候是由大权重带来的）导致了应用 在信号上的激活函数的斜率变得非常平缓。这降低了神经网络学习到更好权重的能力。 另一个问题是零值信号或零值权重。这也可以使网络丧失学习更好权重的能力。 内部链接的权重应该是随机的，值较小，但要避免零值。如果节点的传入链接较多，有一 些人会使用相对复杂的规则，如减小这些权重的大小。 输入应该调整到较小值，但不能为零。一个常见的范围为0.01～0.99，或-1.0～1.0，使用哪 个范围，取决于是否匹配了问题。 输出应该在激活函数能够生成的值的范围内。逻辑S函数是不可能生成小于等于0或大于等 于1的值。将训练目标值设置在有效的范围之外，将会驱使产生越来越大的权重，导致网 络饱和。一个合适的范围为0.01～0.99

PYTHON：

１，使用数组来表述输入信号，权重和输出信号的矩阵。神经系统网络内部的新号前馈或误差在神经网络中反向传播的时候，使用数组来表示这些信号和误差