**神经网络学习笔记**

1. **深度学习基本概念**

深度学习的核心是神经网络，神经网络主要是是**分类和回归**，**分类问题**是数据属于哪一个类别的问题。比如，区分图像中的人是男性还是女性的问题就是分类问题。而**回归问题**是根据某个输入预测一个（连续的）数值的问题。比如，根据一个人的图像预测这个人的体重的问题就是回归问题。回归问题的激活函数一般采用**恒等函数**，分类问题采用**softmax函数**或者**sigmoid函数**。

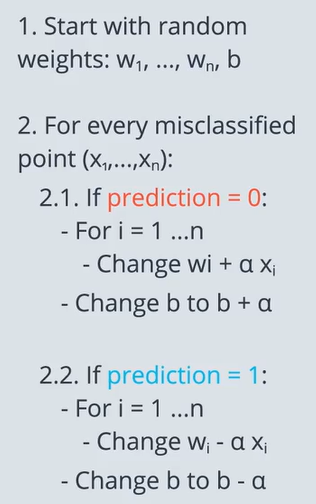
1. **感知器的介绍**

感知器类比人脑的神经元，输入的数据类比神经元树突获得的输入，神经元的作用是对这些冲动进行处理，然后判断是否通过轴突输出神经冲动。个人理解为，可以把感知器想象成一个黑盒子，数据进来盒子，盒子经过一定的运算输出结果，这个结果最好是0-1之间的数（连续）或者只有0和1（离散）。

**感知器有四类简单的感知器**，AND、OR和NOT感知器比较简单，跟电路里面的与或非门的功能一样，但是实现的过程感知器是找一条线性方程来对这四种输入数据进行分类，同时加上阶跃函数来实现二分类（只有0和1）。NOR感知器稍微复杂一点，但是可以用前面三种感知器组合成异或感知器，首先是与感知器和非感知器运算出第一个结果，然后第二个结果是或感知器，将这两个结果作为最后一个感知器（AND）的输入数据，最终出来的结果就是异或的结果。

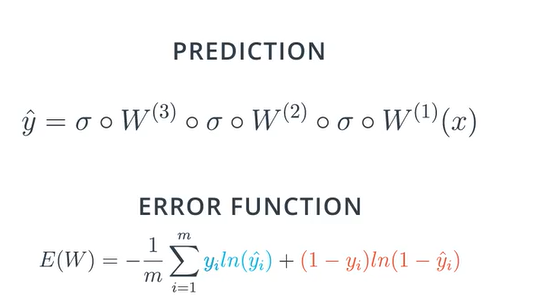
1. **感知器算法**

利用分界线两区域分类出错的点的坐标做参考点，调整分界线更接近它，从而不断修正分类函数。**整个数据集中的每一个点都会把分类的结果提供给感知器（分类函数），并调整感知器。**这就是计算机在神经网络算法中，找寻最优感知器的原理。



1. **误差函数和梯度下降**

对于有些问题无法用线性的界线来进行分类，可能是其他曲线，这个时候如何知道某个值距离界线的距离有多远，我们可以借助**误差函数**来进行衡量。误差函数可以告诉我们当前与正确答案之间的差别有多大。误差函数最好是连续的，可微分的。对于离散型的函数我们可以借助sigmoid函数来实现从离散到连续的转化。



1. **多类别分类和softmax函数**

在输出层添加更多的输出节点，每个节点告诉我们每个结果的概率，将结果运用softmax函数就可以得到每个类别的概率。

1. **One-hot编码**

对于多分类问题，可以把不同类别或者不是数字形式的输入转换成数字的形式，这就用到one-hot编码。比如有大象、河马和蚂蚁三类，可以编码为100，010，001，这样就可以方便后续的训练处理。

1. **最大似然率和交叉熵**

通过激活函数的处理的结果是处于0-1之间的数，也就是我们所说的概率。那么我们就可以用概率的形式来评判模型的好坏，把概率低的模型转换成概率高的模型的过程就叫最大似然法，也叫最大化概率。但是求概率的过程中我们运用了多次乘积的方法，在计算机中乘积的形式效率比较低，而求和的效率比较高，于是我们希望把求积转换成求和，这就需要借助log函数，因为log（a+b）=loga + logb 。

引入log后我们发现对概率取log结果是负值，因为概率是处于0-1之间的数，log在[0,1]上是小于零的，所以我们对log的值取负，将其转换成正数。这一系列转换后出来的结果就是我们所说的交叉熵，也就是我们可以把**最大似然率**转换成**最小化交叉熵**的形式。

其中，交叉熵的公式：

1. **Logistic回归**

1.获得数据

2.选择一个随机模型

3.计算误差

4.最小化误差，获得更好的模型

1. **梯度下降法**

我们得到误差函数后，需要最小化误差函数，实现这个功能的工具就是**梯度下降法**。推导梯度计算公式，运用到偏导数。

1. **非线性数据**

类比感知器的过程，我们可以把两个非线性模型组合成一个线性模型，只需要跟前面的线性过程一直把权重和偏差组合在一起，得出来的模型就是非线性的。那个模型的权重大一点它对最终组合后的非线性模型的结果影响就大一点。

1. **神经网络的层级结构**
2. 输入层：输入数据（n个点代表n维空间）
3. 隐藏层：针对输入层的一系列线性模型？
4. 输出层：多个线性模型组合成一个非线性模型
5. **前向反馈**

神经网络将输入变成输出的过程称为前向反馈。

预测值：，w1是隐藏层，w2是输出层

1. **反向传播**

1.进行前向反馈运算。

2.将模型的输出与期望的输出进行比较。

3.计算误差。

4.向后运行前向反馈运算（反向传播），将误差分散到每个权重上。

5.更新权重，并获得更好的模型。

6.继续此流程，直到获得很好的模型。

1. **线性回归和逻辑回归的区别**
2. 线性回归要求因变量（假设为Y)是连续数值变量，而logistic回归要求因变量是离散的类型变量，例如最常见的二分类问题，1代表正样本，0代表负样本；
3. 线性回归要求自变量服从正态分布，logistic回归对变量的分布则没有要求；
4. 线性回归要求自变量与因变量有线性关系，Logistic回归没有要求；
5. 线性回归是直接分析因变量与自变量的关系,logistic回归是分析因变量取某个值的概率与自变量的关系。

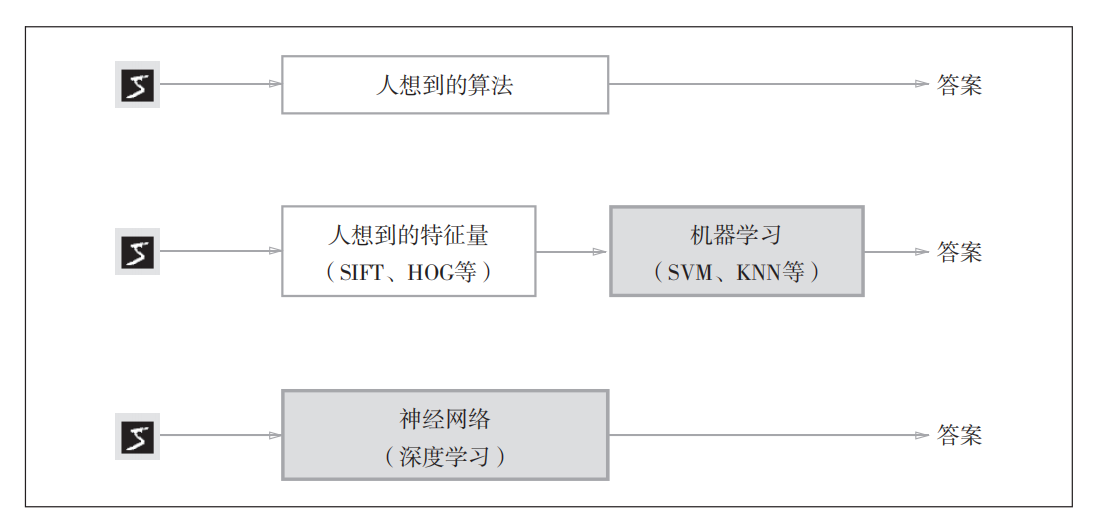
总之，logistic回归与线性回归实际上有很多相同之处，**最大的区别就在于他们的因变量不同**，其他的基本都差不多，正是因为如此，这两种回归可以归于同一个家族，即**广义线性模型**。这一家族中的模型形式基本上都差不多，不同的就是因变量不同，如果是连续的，就是多重线性回归，如果是二项分布，就是logistic回归。logistic回归的因变量可以是二分类的，也可以是多分类的，但是二分类的更为常用，也更加容易解释。所以实际中**最为常用的就是二分类的logistic回归**。线性回归可以看作一个Perceptron, 激活函数是identical, 即恒等函数将逻辑回归也可以看作一个Perceptron, 不同的是使用了sigmoid激活函数。逻辑回归的主体还是回归操作: 回归对象是sigmoid函数, 它将输入映射为一个处于0到1之间的小数. 得到这个0到1之间的小数之后人为将其解读成概率, 然后根据事先设定的阈值进行分类。

1. **线性空间和非线性空间的区别**

由直线分割而成的空间称为线性空间，由曲线分割而成的空间称为非线性空间。

1. **传统机器学习和深度学习的区别**

传统的机器学习需要先从图像中提取特征量，再学习这些特征量。而深度学习不需要人为参与，可以全自动进行。



1. **常见的激活函数（非线性）**

阶跃函数：感知器

sigmoid函数：应用于二分类

softmax函数：应用于多分类

恒等函数：回归问题

为什么采用这些函数来做激活函数？结合损失函数来理解，对于softmax函数采用交叉熵来作为损失函数，反向传播得到的输出值和标签值的差分，而回归问题采用的激活函数是恒等函数，损失函数是平方和误差函数，反向传播的结果也是输出值和标签值的差分。这个差分值就是用来衡量模型的好坏的标准，我们的目标是让这个差分值最小化，这就是为什么要用这些激活函数和损失函数。

1. **总结（还是有些概念理解的不够透彻，后续学习新知识还需要回头再次整理）**

**1.从问题的维度来梳理知识点:**

对于一堆数据，我们需要进行二分类，可以抽象定义为0或1这两类，那么我们需要找到一条线来讲这堆数据进行分类，我们可以规定线以上的区域是正（1），线以下的区域是负（0）。

那么问题就转化为如何找到这条分界线？简单情况下这条线是线性的，所以我们也用y=wx+b来表示这条线。这个线性方程中的w决定线的斜率，b决定线的截距。不同的w和b就代表不同的线。于是，我们把问题转化为寻找w，b。我们把w叫做**权重**，b叫做**偏差**。

既然我们需要找到这样一条线，我们就需要运用大量的数据进行训练来找到最佳的w和b。首先，我们可以随便画一条线，然后运用大量数据来判断当前这条线的分类效果如何，然后通过分类错误的定来不断修正我们这条线，也就是通过大量数据的训练来寻找最佳的w和b。

对于二分类问题，我们可以知道输入的特征是两个，也就是两个向量x1、x2。将x1和x2带入y=wx+b中可得y=w1\*x1+w2\*x2+b。对于这条线上的数据，y=0，如果y>0代表在线以上的区域，也就是属于1类，反之为0类。这里的y就是预测的值，我们写成y(预测)**。**预测的结果如果和实际的分类结果一直，代表这个点分类没问题，**不需要处理**。我们只需要对那些分类错误的点进行处理，如果点分类是1，但是标签（正确值）是0，那么我们可以减去一个值，如果分类是0，但是标签是1，我们可以加上一个值。其实就是把这条线不断靠近那些错误的点，直到最大程度分好类。那么这个值是多少呢？

在二分类问题中，这个值是 αp，αq和α，其中α是学习速度，p，q是x的坐标。我们需要w1- αp，w2-αq和b-α或者w1+αp，w2+αq和b+α。这就是简单的**误差函数**。通过不断地训练，最终我们可以找到最佳的w和b。这就是**感知器算法**。

对于无法用线性的线来进行划分时，我们需要引入**非线性**的概念。对于数据的分类可能不再是直线，而是其他形状的线。那么问题就转换为了如何找到这样的界线。我们可以把这样的界线叫做**概率函数**。概率函数的好坏决定了数据分类结果的好坏，因此我们需要类似线性情况下的训练过程来找到最佳的概率函数。这类概率函数的结果是概率，而不是离散的0和1，因此我们需要借助**sigmoid函数**来把离散的数转换为连续的数，范围在0-1之间，也就是我们说的概率。这也是我们所说的**逻辑回归**问题。

很显然，概率越大代表模型越好，反之模型越差。因为，寻找最佳概率函数转化为寻找**最大化概率**。然而计算概率的时候我们运用了乘积的形式，这种形式对于计算机来说效率太低，因此我们希望能够把乘积转换成求和的形式，于是我们引入了**log函数**，这个函数可以把乘积变成求和，因为log（a\*b）=loga+logb。最终我们可以用**交叉熵来**表示。从而问题就转换为了寻找**最小化交叉熵，**也就是**最小化误差函数**。

为了最小化误差函数，我们引入了**梯度下降算法**，这个算法跟感知机算法类似。感知器算法实现告诉我们，获取正确分类的方式，就是通过每一个错误分类的点，评估错误点位置与我们期望位置之间的差异，来慢慢的修正我们分类函数。而梯度下降跟感知器算法的区别在于，感知器算法对于正确的点不做处理，而**梯度下降向正确的点远离、错误的点靠近**。

对于**多分类**的问题，我们只需要把激活函数从sigmoid转换成softmax就可以实现。

**2.从感知器到神经网络的维度来梳理知识点：**

一开始人们希望模拟人脑的神经元，于是发明了感知器。但是感知器只局限于线性空间，其激活函数是阶跃函数。对于非线性的空间，人们想到了采用多层感知器的形式。其实就是激活函数采用了非线性的sigmoid函数的感知机。由多层感知器，不断加深层次，就演变成了神经网络。我们都知道，感知器需要人为设置权重，这比较麻烦，神经网络可以解决这个问题。神经网络可以自动地从数据中学习到合适的权重参数。神经网络可以分为**回归问题**和**分类问题**，两者的区别在于输出的结果。如果输出的结果是一个值（连续的），那么是回归问题，如果输出结果是数据属于哪个类别，那么是分类问题，而分类问题又分为**二分类**和**多分类**。一般来说，我们所说的**线性回归**就是回归问题，**逻辑回归**一般指二分类问题。除此之外，回归问题的输出层采用的激活函数是**恒等函数**，而二分类问题采取的是**sigmoid函数**，多分类问题采取的是**softmax函数**。

既然神经网络是不需要人为参与的，那么神经网络是如何获得之前需要人为设置的权重的呢？为了理解神经网络的工作过程，我们需要引入一些衡量标准，其中最重要的是**损失函数**。

**损失函数是表示神经网络性能的“恶劣程度”的指标，即当前的神经网络对监督数据在多大程度上不拟合，在多大程度上不一致。**以“性能的恶劣程度”为指标可能会使人感到不太自然，但是如果给损失函数乘上一个负值，就可以解释为“在多大程度上不坏”，即“性能有多好”。并且，“使性能的恶劣程度达到最小”和“使性能的优良程度达到最大”是等价的，不管是用“恶劣程度”还是“优良程度”，做的事情本质上都是一样的。损失函数一般采用**均方误差**和**交叉熵误差**。

**为什么不能使用识别精度来衡量呢？**我们知道对权重参数的损失函数求导，表示的是“如果稍微改变这个权重参数的值，损失函数的值会如何变化”。如果导数的值为负，通过使该权重参数向正方向改变，可以减小损失函数的值；反过来，如果导数的值为正，则通过使该权重参数向负方向改变，可以减小损失函数的值。不过，当导数的值为0时，无论权重参数向哪个方向变化，损失函数的值都不会改变，此时该权重参数的更新会停在此处。之所以不能用识别精度作为指标，是因为这样一来绝大多数地方的导数都会变为0，导致参数无法更新。这也是为什么神经网络激活函数不能采用阶跃函数，因为阶跃函数求导大部分为0，而sigmoid和softmax等函数不会出现这样的情况。

既然我们把损失函数作为衡量的标准，那么为了使模型更好，我们就应该想办法让损失函数达到最小的状态，也就是让预测的结果和实际的结果误差最小，这样的模型就是最好的模型。为了最小化损失函数，我们引入**数值微分**和**梯度**的概念**。，利用微小的差分求导数的过程称为数值微分，梯度是把数值微分所求的偏导数汇总而成的向量。**所谓的梯度其实就是前进的方向，可以想象成我这个界线需要向判断错误的点靠近，判断正确的点远离，这个移动的方向就是梯度。梯度表示的是各点处的函数值减小最多的方向，因此，在寻找函数的最小值（或者尽可能小的值）的位置的任务中，要以梯度的信息为线索，决定前进的方向。通过不断地沿梯度方向前进，逐渐减小函数值的过程就是**梯度下降法**。

我们知道通过数值微分的方法来计算梯度其实效率比较低，于是我们引入**误差方向传播法**来计算梯度。通过计算梯度，不断调整权重，从而找到最好的模型。这就是神经网络的工作过程。

对于这样的模型，宏观上是这样的，但有些细节我们还需要注意。比如权重的初始值如何设定？对于sigmoid或tanh函数，我们可以使用标准差为1/sqrt(n)的高斯分布作为权重初始值，这个叫做**Xavier初始值，**对于激活函数是ReLU函数时，权重初始值为sqrt（2/n），这个是**He初始值。**同**时要注意的是权重初始值不能为0**。

除此之外，一些超参数的初始值设置也是需要注意的，比如**学习速率**。学习速率是指沿着梯度的方向变化的幅度大小，如果过大可能会越过正确的点，如果过小可能永远接近不了正确的点。这就需要我们对**超参数进行最优化**，不断缩小超参数的范围，最终取一个比较合适的指。

当然，训练出来的模型还可能会存在**过拟合**的现象。过拟合指的是只能拟合训练数据，但不能很好地拟合不包含在训练数据中的其他数据的状态。**权值衰减**和**Dropout**是经常被使用的抑制过拟合的方法。