### 第二十次2020年4月27日-5月10日

#### 1 机器学习

神经网络：

初始特征个数n很大时，将这些高阶多项式项数包括到特征里会使特征空间急剧膨胀。n很大时，增加特征来建立非线性分类器并不是好办法。如图计算机视觉领域一个简单的问题，识别图片是否为汽车。50\*50的像素图片对应2500个灰度像素，7500个RGB像素。2500个特征对应的二次特征多达三百万，我们之前讲过的logistic算法不能完美的解决这类问题，因此我们接下来了解神经网络是如何帮助我们解决这类问题。

手机屏幕截图

描述已自动生成

神经网络模型建立在很多神经元之上，每个神经元又是一个学习模型。这些神经元（也叫激活单元，activation unit）采纳一些特征作为输出，并且根据本身的模型提供一个输出。下图是一个以logistic回归模型作为自身学习模型的神经元示例，在神经网络中，参数又可被称为权重（weight）。x0是偏置单元（偏执神经元）恒等于1，是否画出视需要而定。

手机屏幕截图

描述已自动生成

a(j)i代表第j层的第i个激活单元。Θ(j)代表从第j层映射到第j+1层时的权重的矩阵。例如Θ(1)代表从第一层映射到第二层的权重矩阵。其尺寸为：以第j+1层的激活单元数量为行数，以第j层的激活单元数加1为列数的矩阵。神经网络中Θ(1)的尺寸为3\*4。Θ下标的第一个数字：代表第j+1层神经元的编号，第二个数字代表第j层神经元的编号。上标则代表神经元层数的编号。

我们可以知道：每一个a都是由上一层所有的x和每一个x所对应的参数决定的。我们把这样从左到右的算法称为前向传播算法(FORWARD PROPAGATION)

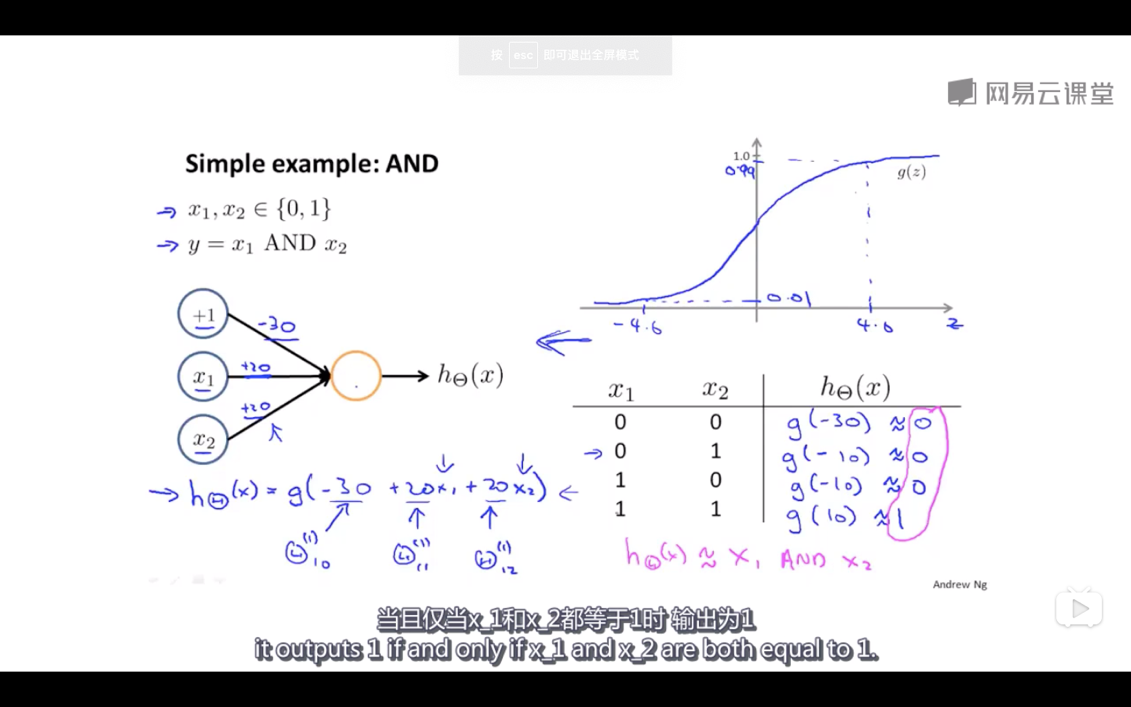
我们可以把a0,a1,a2,a3看成更为高级的特征值，也就是x0,x1,x2,x3的进化体，并且它们是由x决定的，因为是梯度下降的，所以a是变化的，并且变得越来越厉害，所以这些更高级的特征值远比仅仅将x次方厉害，也能更好的预测新数据。这就是神经网络相比于逻辑回归和线性回归的优势。

手机屏幕截图

描述已自动生成手机屏幕截图

描述已自动生成

单层神经元模拟逻辑与AND



单层神经元模拟逻辑或OR

手机屏幕的截图

描述已自动生成

单层神经元模拟逻辑非NOT

手机屏幕截图

描述已自动生成

单层神经元模拟XNOR

XNOR：输入的两个值必须一样，均为1或均为0

XNOR=(x1 AND x2) OR ((NOT x1) AND (NOT x2))

首先模拟(NOTx1)AND(NOTx2)同时为0时候为真，设三个权重分别为10，-20，-20。

一些文字和图片的手机截图

描述已自动生成

我们就得到了一个能实现XNOR运算符功能的神经网络。按这种方法我们可以逐渐构造出越来越复杂的函数，也能得到更加厉害的特征值。这就是神经网络的厉害之处。

面对多元分类问题，神经网络又是如何解决的呢？如图我们需要将图片识别为四种。

手机屏幕截图

描述已自动生成

原来我们用{1,2,3,4}来区别表示这四种分类结果，现在我们用一个四元列向量来表示，第一个值为1或0用于预测是否是行人，第二个值用于判断是否为汽车等等。x代表输入，y代表输出。许多（x,y）构成训练集。输入向量x有三个维度，两个中间层，输出层4个神经元分别用来表示4类，也就是每一个数据在输出层都会出现[abcd]T，且a,b,c,d中仅有一个为1，表示当前类。

手机屏幕截图

描述已自动生成

#### 2 论文学习

Simplifying CDCL Clause Database Reduction

**本文目标：**提出一种对生成的学习子句进行精简的有效新方法。

**背景：**

传统的精简学习子句方法是把学习子句分别储存在Core和Local中。Core中的子句在整个运行过程中保留，而Local中定期删除‘低质量’的子句。质量通常由子句大小，存在时间，文字块距离（LBD）和使用频率（活性）等评估。随着时间的推移，许多学习子句精简的方法融合在一起时的现在没有一个简洁明了的方法。关于质量度量的复杂性也使得我们难于去评估单个元素的贡献。我们期望一个很好的子句质量度量方法可以考虑到多种因素。

除了用一种策略来评估子句质量高低之外，我们还需要一种算法来删除低质量子句。1）堆方案：我们把子句质量度量Q和这些子句保留在堆中，子句数据库太大时可以删除最低质量的子句。然而使用堆的效率太低，花时间找到最低质量的子句也显得得不偿失。2）快速启发式：例如Delete-Half是定期对Local子句进行排序，并删除质量最低的一半。

**本文贡献：**

1.我们引入了一种新的“在线”子句删除方案，该策略易于实现并将Local的大小保持在所需的值。它不使用排序，在许多自然实例化中，每次冲突只需要花费常量时间。

2.我们将提出的策略实例化，此实例使用非常简单的机制考虑了子句用法和LBD。我们生成的求解器（Online-RU-T2Flag）与MapleLCMDistChronoBT（2018年SAT竞赛的第一名求解器）性能相当。

3.为了帮助理解特定方法在求解器性能中发挥作用的程度，我们提供了一些测量性能或其他属性的实验数据。

**MapleSAT删除方案：**

MapleLCMDistChronoBT使用COMiniSatPS引入的删除方案。

该方案具有三个子句数据库Core，Tier2和Local。将新学习的子句存储在何处的决定取决于其LBD：如果LBD≤3，则为Core；如果4≤LBD≤6，则为Tier2；如果6 <LBD，则为Local。如果在100,000次冲突之后，Core中没有足够的子句，则将核心阈值从3更改为5。

LBD或使用频率会影响子句在DB间的转移。每次冲突分析或子句简化过程之后，都会重新计算每个子句的LBD。如果子句的LBD充分减小，则将其从Local移到Tier2或Core，或从Tier2移到Core。每10,000个冲突后，在上30,000个冲突中未使用的Tier2子句都将移至Local。每发生15,000个冲突，Local中的所有子句均按其活性进行排序，并且删除了一半的子句。与当前分配有关的子句以及最近在LBD中有所改进的子句从删除中保存。

**本文提出的在线删除方案：**

我们用单向循环链表存储Local子句，索引Li标示‘删除候选者’，子句质量用Q表示，阀值用q表示。当需要在本地存储一个新的学习子句C时，我们在列表中选择一个“低质量”子句，以通过顺序搜索将其替换为C。 只要Q（Li）≥q，我们就增加i（“保存” 子句Li再进行一次“回合”）； 第一次Q（Li）<q时，我们将Li替换为C（删除“旧” Li）。 必须选择子句质量度量阈值，以使列表中始终有足够多的“低质量” 子句。 有算法方法可以确保这一点（例如，使用反馈控制机制），但是没有它们就很难获得良好的实用性能。

半删除和在线删除：考虑Delete-Half每k个冲突检查一次，如果其质量低于Local中当前子句的中位数，则将其删除。 如果实例化我们的在线方案S = 2k，并且使q足够接近中位数，我们希望每k个冲突检查每个子句，如果其质量低于Local中当前子句的中值，则将其删除。 从这个意义上讲，这两种方案可以做得很接近：我们在权衡取舍以动态估计中位数的情况下进行权衡。 这样，我们得到一个大小统一的子句数据库，而不是一个显着增长和收缩的子句数据库。

基于子句时间的删除：对每个子句C都假设Q（C）<q。这导致了一个纯粹的基于年龄的方案：每个新学习的子句都替换Local中最旧的子句。 这种非常低成本的方案效果很好。

#### 3 其他方面

老师对不起，实验结果我再整理一下尽快提交，现在的实验结果不太好。。