机器学习导论
作业三

学号, 作者姓名, 邮箱

2018 年 5 月 5 日

1 [15pts] Decision Tree I

(1) **[5pts]** 假设一个包含三个布尔属性X, Y, Z的空间，并且目标函数是$f(x, y, z) = x$ **XOR** z，其中**XOR**为异或运算符。令H为基于这三个属性的决策树，请问：目标函数f可实现吗？如果可实现，画出相应的决策树以证明；如果不可实现，请论证原因；

(2) **[10pts]** 现有如表 1所示数据集：

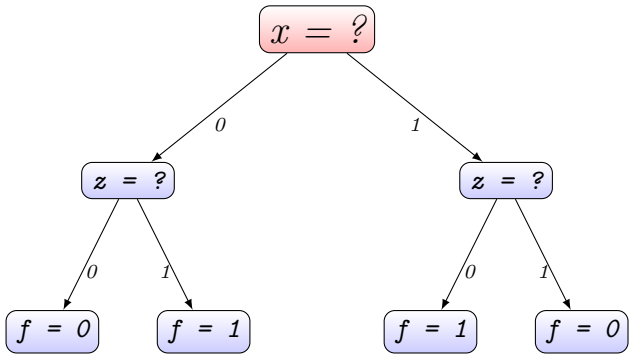
表 1: 样例表

X	Y	Z	f
1	0	1	1
1	1	0	0
0	0	0	0
0	1	1	1
1	0	1	1
0	0	1	0
0	1	1	1
1	1	1	0

请画出由该数据集生成的决策树。划分属性时要求以信息增益 (information gain)为准则。当信息增益 (information gain)相同时，依据字母顺序选择属性即可。

Solution.

(1) 目标函数f是可以实现的，决策树如下所示：



(2) 按照信息增益为准则，划分依据为：

第一层根结点：

如果选择X划分：

$$\begin{aligned} Gain(D,X) =& Ent(D) - \sum_{v=1}^{2} \frac{|D^v|}{|D|} Ent(D^v) \\ =& 1 - 1 = 0 \end{aligned} \tag{1.1}$$

如果选择Y划分：

$$Gain(D,Y) = 1 - 1 = 0 \tag{1.2}$$

如果选择Z划分：

$$Gain(D,Y) = 1 - \frac{3}{4} \times 0.918 = 0.6885 \tag{1.3}$$

所以选择Z划分。

第二层第一个结点都是同一类，现在考虑第二个结点的划分：

如果选择X划分：

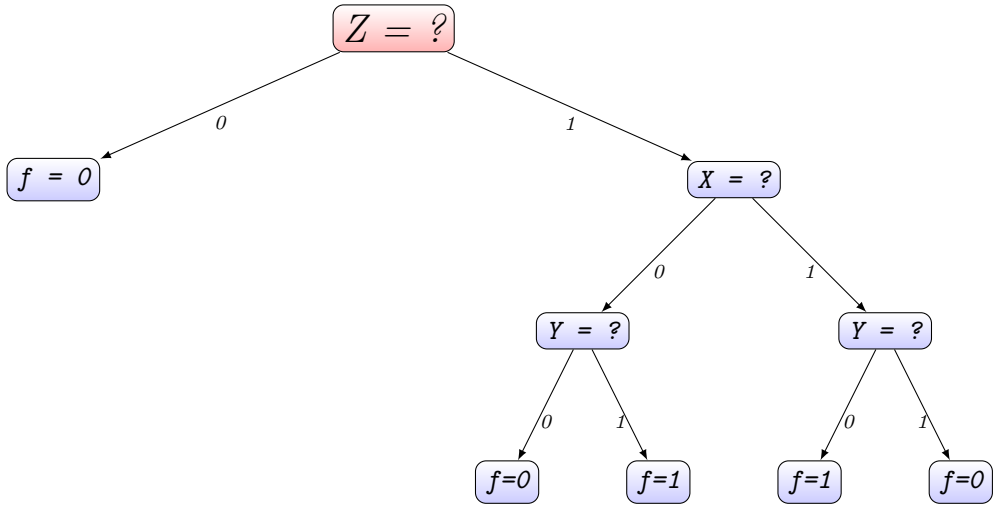
$$Gain(D,X) = 0.918 - 0.918 = 0 \tag{1.4}$$

如果选择Y划分：

$$Gain(D,Y) = 0.918 - 0.918 = 0 \tag{1.5}$$

所以按照字母顺序选择X划分。

根据数据集生成的决策树如下：


Z = ?
0
1
f = 0
X = ?
0
1
Y = ?
Y = ?
0
1
0
1
f=0
f=1
f=1
f=0

2 [20pts] Decision Tree II

考虑如下矩阵：

$$\begin{bmatrix} 4 & 6 & 9 & 1 & 7 & 5 \\ 1 & 6 & 5 & 2 & 3 & 4 \end{bmatrix}^T$$

该矩阵代表了6个样本数据，每个样本都包含2个特征f_1和f_2。这6个样本数据对应的标签如下：

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}^T$$

在这个问题中，我们要构造一个深度为2的树进行分类任务。

(1) **[5pts]** 请计算根结点 (root) 的熵值 (entropy)；

(2) **[10pts]** 请给出第一次划分的规则，例如$f_1 \geq 4, f_2 \geq 3$。对于第一次划分后产生的两个结点，请给出下一次划分的规则；

提示：可以直观判断，不必计算熵。

(3) **[5pts]** 现在回到根结点 (root)，并且假设我们是建树的新手。是否存在一种划分使得根结点 (root) 的信息增益 (information gain) 为0?

Solution.

(1) 根结点的熵值为

$$Ent(D) = -(\frac{3}{6}\log_2\frac{3}{6} + \frac{3}{6}\log_2\frac{3}{6}) = 1.000 \tag{2.1}$$

(2) 规则如图*1*所示。第一次划分的规则为$f_1 \leq 6$。不满足$f_1 \leq 6$的结点的所有样本的标签都是1，所以这一部分不需要再次划分。第一次划分之后，满足$f_1 \leq 6$的节点的第二次划分的规则为$f_2 \leq 1$，其中满足$f_2 \leq 1$的结点的样本的标签都都是1，不满足的都是0，所以不需要再次划分。

(3) 规则如图*2*所示。按照$f_2 \leq 2$来划分，将D分成了D^1和D^2两个部分。其中D^1包含第一个和第四个样例。由此可得

$$Ent(D^1) = -(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}) = 1.000 \tag{2.2}$$

$$Ent(D^2) = -(\frac{2}{4}\log_2\frac{2}{4} + \frac{2}{4}\log_2\frac{2}{4}) = 1.000 \tag{2.3}$$

所以可得信息增益为

$$\begin{aligned} Gain(D, f_2 \leq 2) &= Ent(D) - \sum_{v=1}^{2}\frac{|D^v|}{|D|}Ent(D^v) \\ &= 1 - (\frac{2}{6}\times 1 + \frac{4}{6}\times 1) \\ &= 0 \end{aligned} \tag{2.4}$$

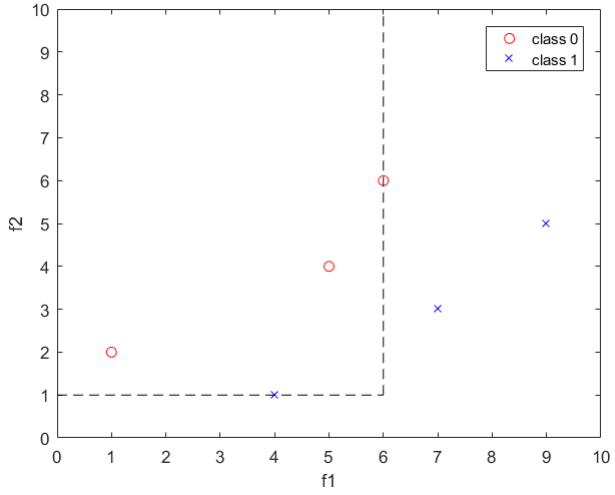


图 1: 划分规则

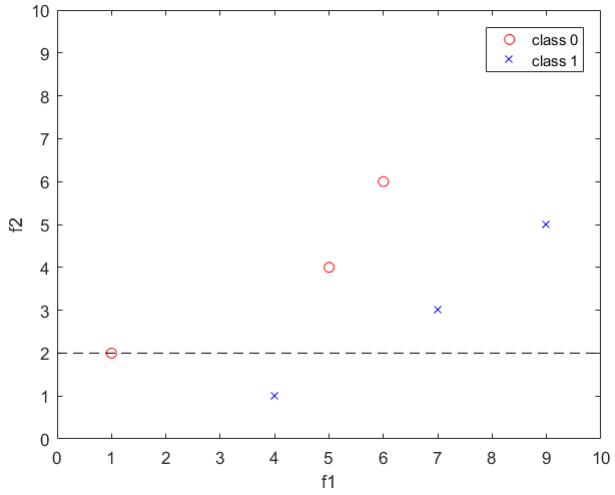


图 2: 划分规则

3 [25pts] Universal Approximator

已知函数$f:[-1,1]^n \mapsto [-1,1]$满足$\rho$-Lipschiz性质。给定误差$\epsilon > 0$，请构造一个激活函数为 sgn($\mathbf{x}$) 的神经网络$\mathcal{N}:[-1,1]^n \mapsto [-1,1]$，使得对于任意的输入样本$\mathbf{x} \in [-1,1]^n$，有$|f(\mathbf{x}) - \mathcal{N}(\mathbf{x})| \leq \epsilon$。
(Lipschitz条件参见Wikipedia，其中 sgn($\mathbf{x}$) 的定义参见《机器学习》第98页。)

(1) **[5pts]** 请画出构造的神经网络$\mathcal{N}$的示意图；

(2) **[10pts]** 请对构造的神经网络进行简要的说明(写清每一层的线性组合形式，也就是结点间的连接方式和对应的权重)；

(3) **[10pts]** 证明自己构造的神经网络的拟合误差满足要求。

Solution. *(1)* 神经网络$\mathcal{N}$的示意图如下所示：

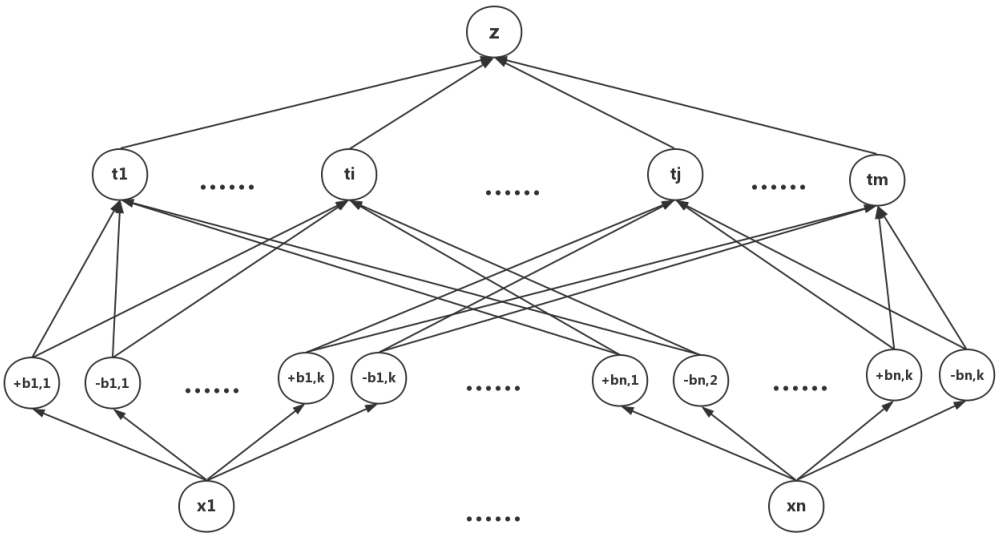


图 3: 神经网络$\mathcal{N}$

(2) 神经网络说明：

结构：

神经网络分为4层，其中包括输入层，输出层和两个隐层。其中输入层有n个输入单元（每个单元对应了一个$\mathbf{x}$的维度的数值），第一个隐层（以后称为B层）有$2nk$个单元，第二个隐层（以后称为T层）有m个单元，输出层只有一个单元也就是整个神经网络的输出值。

符号表示： 见下表：

表 2: 符号表

x_i	输入单元接受的输入值，也就是$\mathbf{x}$向量的各个维度的数值
$b_{i,j}^+, b_{i,j}^-$	B层的单元，注意到$b_{i,j}^+$和$b_{i,j}^-$是成对出现的
$\alpha_{i,j}^+, \alpha_{i,j}^-$	分别是$b_{i,j}^+$和$b_{i,j}^-$的输入
$\beta_{i,j}^+, \beta i, j^-$	分别是$b_{i,j}^+$和$b_{i,j}^-$的输出
$\theta_{i,j}^+, \theta i, j^-$	分别是$b_{i,j}^+$和$b_{i,j}^-$的阈值
t_i	T层的单元
γ_i	t_i的输入
δ_i	t_i的输出
λ_i	t_i的阈值
z	输出层的单元的输出值
K	函数f的Lipschiz常数
m	神经网络的超参数，用于控制B层的神经元的数目

线性组合形式：

输入层神经元数目：n

B层神经元数目：$2m$

T层神经元数目：m^n

输出层神经元数目：1

从输入层到B层：

$$\alpha_{i,j}^{*} = x_i \ , \quad * \in \{+,-\} \tag{3.1}$$

B层的神经元计算：

$$\begin{aligned} \theta_{i,j}^{+} &= -1 + (j-1)L \\ \theta_{i,j}^{-} &= -1 + jL \\ \beta_{i,j}^{*} &= sgn(\alpha_{i,j}^{*} - \theta_{i,j}^{*}) \ , \quad * \in \{+,-\} \end{aligned} \tag{3.2}$$

从B层到T层，映射较为复杂：B层的神经元$b_{1,index_1}^{+}, b_{2,index_2}^{+}, ..., b_{n,index_n}^{+}$的输出乘以权重1以及$b_{1,index_1}^{-}, b_{2,index_2}^{-}, ..., b_{n,index_n}^{-}$的输出乘以权重$-1$之后作为$t_s$的输入值，其中$s = 1 + \sum_{i=1}^{n}(index_i - 1)m^{(i-1)}$，所以为了根据$s$得到$index_i$，给出以下算法：

Algorithm 1 IndexTranslate

Require: T层的神经元编号s

Ensure: 输入到t_s的B层神经元的编号$(index_1, index_2, ..., index_n)$

1: $s = s - 1$
2: **for** i from 1 to n **do**
3: $index_i = s \mod m^i$
4: $s = s - index_i$
5: $index_i = index_i + 1$
6: **end for**

现在给出从B层到T层的传播：

$$(index_1, index_2, ..., index_n) = IndexTranslate(s) \tag{3.3}$$

$$\gamma_s = \sum_{i=1}^{n} \beta_{i,index_i}^{+} - \sum_{i=1}^{n} \beta_{i,index_i}^{-} \tag{3.4}$$

T层的神经元计算：

$$\begin{aligned} \lambda_s &= n - \frac{1}{2} \\ \delta_s &= sgn(\gamma_s - \lambda_s) \end{aligned} \tag{3.5}$$

(3) ***[10pts]*** 证明自己构造的神经网络的拟合误差满足要求。

4 [40pts] Neural Network in Practice

通过《机器学习》课本第5章的学习，相信大家已经对神经网络有了初步的理解。深度神经网络在某些现实机器学习问题，如图像、自然语言处理等表现优异。本次作业旨在引导大家学习使用一种深度神经网络工具，快速搭建、训练深度神经网络，完成分类任务。

我们选取PyTorch为本次实验的深度神经网络工具，有了基础工具，我们就能如同搭积木一样构建深度神经网络。PyTorch是Facebook开发的一种开源深度学习框架，有安装方便、文档齐全、构架方便、训练效率高等特点。本次作业的首要任务就是安装PyTorch。

目前PyTorch仅支持Linux和MacOS操作系统，所以Window用户需要装一个Linux虚拟机或者直接安装Linux系统。PyTorch安装很方便，只需要在其主页中的Get Start一栏选择对应的环境设置，便能够一键安装。有GPU的同学也可以尝试安装GPU版本的PyTorch。为保证此次作业的公平性，只要求使用CPU进行网络训练，当然有条件的同学也可以尝试使用GPU进行训练。在批改作业时，助教会提供Python 2.7、3.5、3.6三种环境进行实验验证。

我们选取CIFAR10作为本次作业的训练任务。CIFAR10是一个经典的图片分类数据集，数据集中总共有60000张32×32的彩色图片，总共有10类，每类6000张图片，其中50000张图片构成训练集，10000张图片构成测试集。PyTorch通过torchvision给用户提供了获取CIFAR10的方法，详细信息可见PyTorch的教程。此外关于CIFAR10分类准确率排行可见此链接。

下面我们将尝试使用PyTorch来解决实际问题：

(1) **[15pts]** 首先我们跟随PyTorch的教程，用一个简单的卷积神经网络（Convolutional Neural Network, CNN），完成CIFAR10上的分类任务，具体要求如下：

- **[7pts]** 在代码实现之前，大家可能需要对CNN网络进行一定的了解，请大家自行查阅资料（PyTorch的教程中也有部分介绍CNN网络），并在实验报告中给出对CNN的见解：主要回答什么是卷积层，什么是Pooling层，以及两者的作用分别是什么；
- **[8pts]** 接下来就是具体的代码实现和训练。教程会手把手教你完成一次训练过程，其中使用SGD作为优化方法，请同学们自行调整epoch的大小和学习率，完成此次训练。另外，请在实验报告中给出必要的参数设置，以及训练结果如最终的loss、在测试集上的准确率等；

(2) **[20pts]** 显然，这样一个简单的网络在CIFAR10上并不能取得令人满意的结果，我们需要选取一个更为复杂的网络来提升训练效果。在此小题中，我们选取了CIFAR10准确率排行榜上排名第二的结构，具体参见论文链接。为了方便大家实现，我们直接给出了网络结构如图4所示。请大家搭建完成此网络结构，并选择Adam为优化器，自行调整相关参数完成训练和预测，实验结果报告内容同第（1）小题；

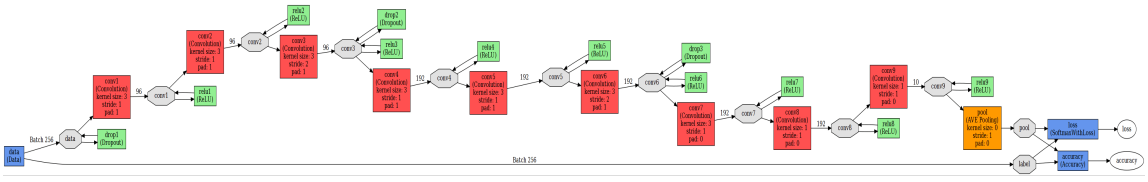


图 4: 待实现网络结构

(3) **[5pts]** 通过上一题实验我们可以发现，即使使用现成的网络结构也不一定能达到与其相同的训练效果。请大家分析其中的原因，并谈谈本次实验的感想，以及对深度学习调参的体会。

实验报告.

(1)

对CNN的见解：卷积神经网络是一种非常强大的，适合用于图像，视频识别以及自然语言处理的神经网络的。卷积神经网络本身就是一种特殊的神经网络，其训练的流程为：输入层接受训练集数据，通过隐层一层层传递到输出层，输出与真实标签进行比较，得到损失函数，然后再通过反向传播来调整隐层的参数，进而降低损失函数，来使得预测结果逼近与真实标签。CNN中比较特殊的隐层结构是卷积层和pooling层，其中卷积层主要承担了对特定模式的识别工作，pooling层主要起到了采样的作用。

卷积层的概念：是卷积神经网络的核心，承担了卷积神经网络大部分的工作量。

卷积层的作用：卷积层的作用主要在于提取特征，这个操作类似于信号处理中的滤波，和人类大脑认知世界也有几分相似。这个操作的实现用到了卷积的方法。具体的，如果输入的数据大小为$H \times W \times D$(此处先假设batch size为1，其中D表示channel数量)，那么卷积层就会有诸多大小为$H' \times W' \times D$的filter(其中$H' < H$而且$W' < W$)。通过将一个filter“覆盖”在输入数据上，会得到输入数据上的一个和filter相同大小的数据块，把数据块和filter对应位置的元素相乘然后求和，可以得到一个标量值，把filter“覆盖”到数据的不同位置可以得到很多标量值，将这些标量值按照顺序排列起来可以得到一个activation map，每一个filter针对每一个输入数据都会得到一个activation map，把多个filter对应的activation map堆叠在一起就可以得到下一个隐层的输入数据。每一个卷积层得到的activation map其实就是对某种特征的提取，如果map中某个元素值很大，就说明这个元素对应的输入数据的特定位置有可能存在某种特征。所以activation map实际上可以反应出特征在输入数据集中的分布。卷积层可以识别的特征多种多样的，一般第一卷积层只能识别一些简单的特征，例如边缘，曲线，角，更多层的卷积层可以识别到更加复杂的特征。

Pooling层的概念：pooling层也是卷积神经网络中一个重要的结构，他的主要功能在于对数据进行采样。

Pooling层的作用：总体来讲pooling层的作用在于减少数据大小，减少参数，降低运算量，减低训练时间，防止过拟合，提高泛化能力，同时还可以保持特征的不变形（包括平移，旋转，尺度等方面）。具体的，pooling层就是按照一定的比例对输入数据进行采样，常见的有max-pooling和average-pooling。常见的做法是将数据分成2×2的数据块然后从每个数据块中选择一个最大值（或者平均值）来代替整个数据块。其中max-pooling采样操作还能够保持输入数据的特征不变型，比如说某个数据块中出现了某个特征，反映在输入数据上就是某个元素的值非常大。当我们在对这个数据块采样的时候，为了在下一层维持这个特征的表现，我们选择了最大的数值作为采样后的结果，这背后蕴含了即便数据缩小了，我们仍然尽可能保持原有的特征这一思想，所以实践中往往max-pooling表现会比较好。实践中会在卷积层之间周期性的插入pooling层来提升整体的效果。

(2)