

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 $f(x, y, z) = x \text{ XOR } z$, 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:

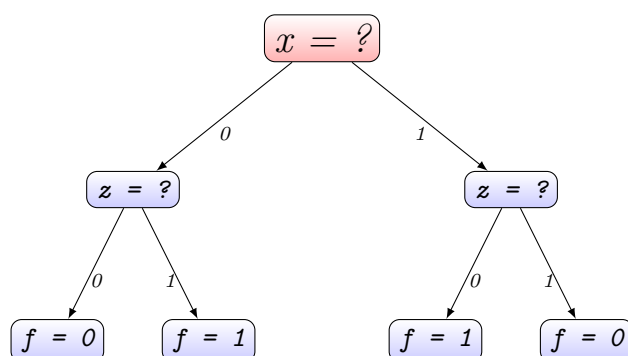
表 1: 样例表

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：

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

如果选择 Y 划分：

$$Gain(D, Y) = 1 - 1 = 0 \quad (1.2)$$

如果选择 Z 划分：

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

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：

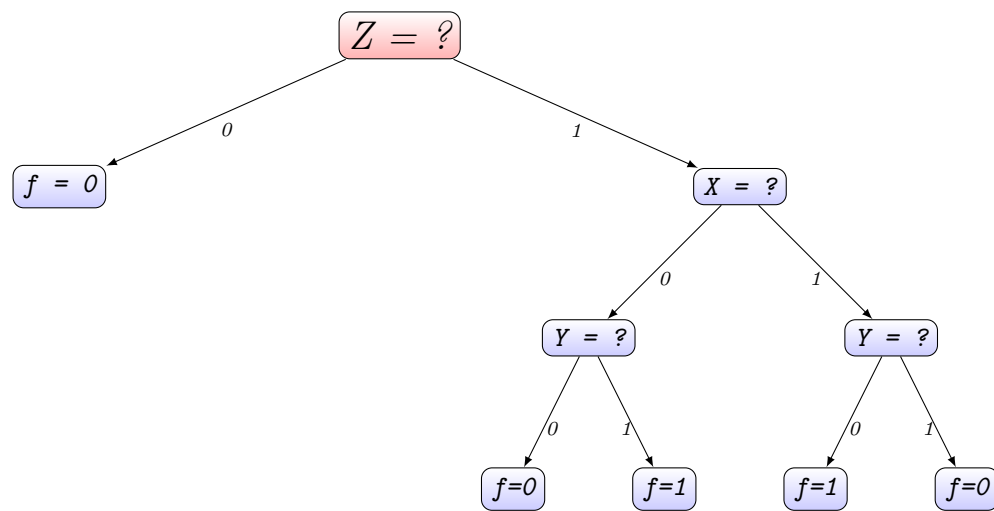
$$Gain(D, X) = 0.918 - 0.918 = 0 \quad (1.4)$$

如果选择 Y 划分：

$$Gain(D, Y) = 0.918 - 0.918 = 0 \quad (1.5)$$

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：



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

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

(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

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

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

所以可得信息增益为

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

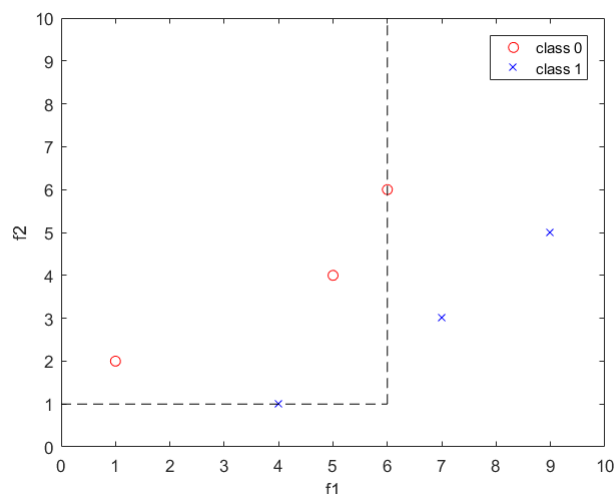


图 1: 划分规则

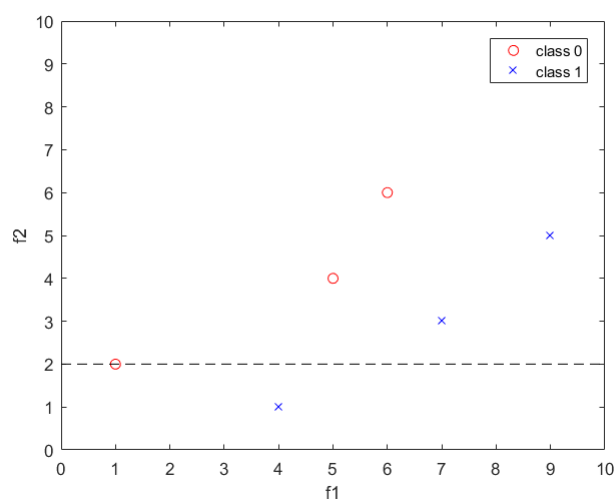


图 2: 划分规则

3 [25pts] Universal Approximator

已知函数 $f : [-1, 1]^n \mapsto [-1, 1]$ 满足 ρ -Lipschitz 性质。给定误差 $\epsilon > 0$ ，请构造一个激活函数为 $\text{sgn}(\mathbf{x})$ 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 $\text{sgn}(\mathbf{x})$ 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:

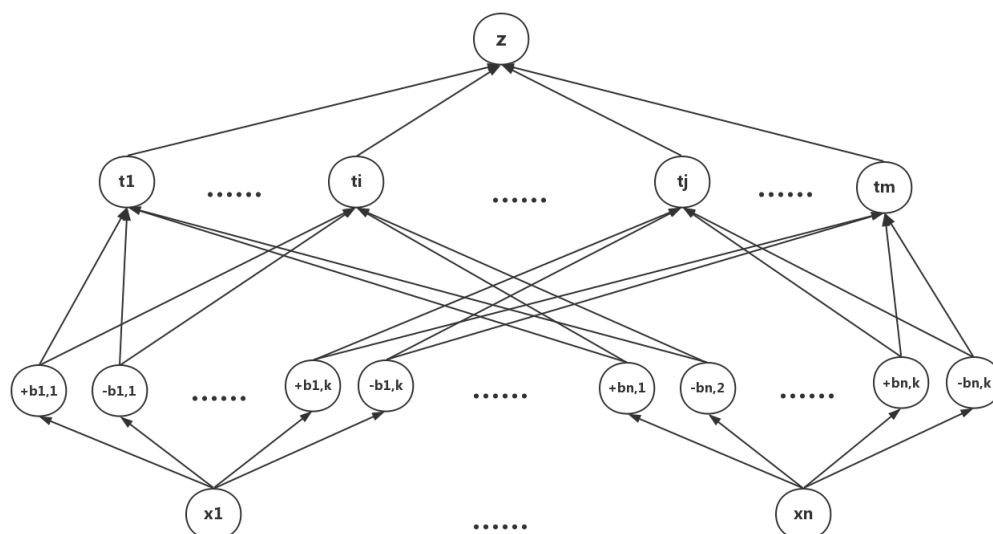


图 3: 神经网络 \mathcal{N}

(2) 神经网络说明:

\mathcal{N} 的结构:

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。

\mathcal{N} 的符号表示：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Lipschitz常数
m	神经网络的超参数，用于控制B层的神经元的数目
σ	$\sigma > 0$ ，加在 $\theta_{i,m}^-$ 上，确保 $x_i = 1$ 时也能正确处理

\mathcal{N} 的各层的线性组合形式:

输入层神经元数目: n , 即 x_1, x_2, \dots, x_n

B 层神经元数目: $2mn$, 即 $b_{1,1}^*, \dots, b_{1,m}^*, \dots, b_{n,1}^*, \dots, b_{n,m}^*$, $*$ $\in \{+, -\}$

T 层神经元数目: m^n , 即 t_1, t_2, \dots, t_{m^n}

输出层神经元数目: 1, 即 z

从输入层到 B 层的传播:

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

B 层的神经元计算:

$$\theta_{i,j}^+ = -1 + (j-1)\frac{2}{m} \quad (3.2)$$

$$\theta_{i,j}^- = \begin{cases} -1 + j\frac{2}{m} & 1 \leq i \leq m-1; \\ -1 + j\frac{2}{m} + \sigma & i = m; \end{cases} \quad (3.3)$$

$$\beta_{i,j}^* = \text{sgn}(\alpha_{i,j}^* - \theta_{i,j}^*) , \quad * \in \{+, -\} \quad (3.4)$$

从 B 层到 T 层, 映射较为复杂: B 层的神经元 $b_{1, \text{index}_1}^+, b_{2, \text{index}_2}^+, \dots, b_{n, \text{index}_n}^+$ 的输出乘以权重1以及 $b_{1, \text{index}_1}^-, b_{2, \text{index}_2}^-, \dots, b_{n, \text{index}_n}^-$ 的输出乘以权重-1之后作为 t_s 的输入值, 其中 $s = 1 + \sum_{i=1}^n (\text{index}_i - 1)m^{(i-1)}$, 所以为了根据 s 得到 index_i , 给出以下算法:

Algorithm 1 IndexTranslate

Require: s : T 层的神经元编号

Ensure: $(\text{index}_1, \text{index}_2, \dots, \text{index}_n)$: 输入到 t_s 的 B 层神经元的编号

```

1:  $s = s - 1$ 
2: for  $i$  from 1 to  $n$  do
3:    $\text{index}_i = s \bmod m^i$ 
4:    $s = s - \text{index}_i$ 
5:    $\text{index}_i = \text{index}_i + 1$ 
6: end for
```

现在给出从 B 层到 T 层的传播:

$$(\text{index}_1, \text{index}_2, \dots, \text{index}_n) = \text{IndexTranslate}(s) \quad (3.5)$$

$$\gamma_s = \sum_{i=1}^n \beta_{i, \text{index}_i}^+ - \sum_{i=1}^n \beta_{i, \text{index}_i}^- \quad (3.6)$$

T 层的神经元计算:

$$\lambda_s = n - \frac{1}{2} \quad (3.7)$$

$$\delta_s = \text{sgn}(\gamma_s - \lambda_s) \quad (3.8)$$

从 T 层到输出层的边的权值其实是函数 f 在某些点的值, 下面给出算法通过 t_s 的编号 s 来计算 δ_s 对应的权值

Algorithm 2 GetWeightFromIndex**Require:** s : T层的神经元编号**Ensure:** w_s : 从T层到输出层的边中 δ_s 对应的边的权重

```

1:  $(index_1, index_2, \dots, index_n) = IndexTranslate(s)$ 
2: for  $i$  from 1 to  $n$  do
3:    $xt_i = \frac{2index_i - 1 - m}{m}$ 
4: end for
5:  $\mathbf{xt} = [xt_1, xt_2, \dots, xt_n]$ 
6:  $w_s = f(\mathbf{xt})$ 

```

通过上述算法计算对于每一个 δ_i 的权值 $w_i = GetWeightFromIndex(i)$

输出层的最终输出:

$$z = \sum_{i=1}^{m^n} w_i \delta_i \quad (3.9)$$

(3) 证明:

在进行具体的证明之前, 首先对神经网络 \mathcal{N} 的设计思路进行说明: 由于需要逼近的函数 f 具有Lipschitz性质, 即:

$$|f(\mathbf{x}) - f(\mathbf{y})| \leq K \|\mathbf{x} - \mathbf{y}\| \quad (3.10)$$

所以 \mathcal{N} 的大致思想就是将 f 的定义域分割成很多个足够小的子空间, 根据Lipschitz性质, 子空间内的任意两个点 \mathbf{x} 和 \mathbf{y} 对应的函数值 $f(\mathbf{x})$ 和 $f(\mathbf{y})$ 相差会有一个上界即 $K \|\mathbf{x} - \mathbf{y}\|$ 。只要确定了分割的子空间的大小, 那么 $\|\mathbf{x} - \mathbf{y}\|$ 也会随之确定。所以只要确保将定义域分割的足够细使得 $K \|\mathbf{x} - \mathbf{y}\| \leq \epsilon$ 即可得到符合题意的神经网络。考虑到有限维向量范数的等价性, 为了方便之后的证明, 在这里取2-范数 (即便取其他范数, 我们的结论仍然不会改变, 这个将在之后证明), 即:

$$|f(\mathbf{x}) - f(\mathbf{y})| \leq K \|\mathbf{x} - \mathbf{y}\|_2 \quad (3.11)$$

设置 $m = \frac{k\sqrt{n}}{\epsilon}$, 下证: 对于任意的输入样本 $\mathbf{x} \in [-1, 1]^n$, 有 $|f(\mathbf{x}) - \mathcal{N}(\mathbf{x})| \leq \epsilon$ 。

- (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。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)