机器学习导论
作业三

2018 年 6 月 30 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 2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. *此处用于写解答(中英文均可)*

(1) 可实现，相应决策树如图 *1*所示。

(2) 生成的决策树如图*2*所示。

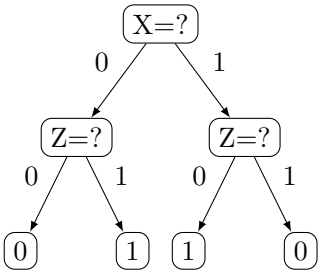


图 1: decision tree 1

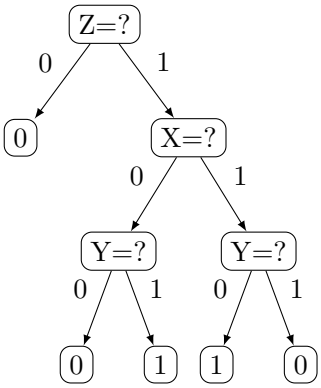


图 2: decision tree 2

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. *此处用于写解答(中英文均可)*

(1)

$$\begin{aligned} Ent(D) &= -\sum_{k=1}^{2} p_k \log_2 p_k \\ &= -(\frac{1}{2}\log_2 \frac{1}{2} + \frac{1}{2}\log_2 \frac{1}{2}) \\ &= 1 \end{aligned} \tag{2.1}$$

(2) 第一次划分：$f_1 \geq 7$.

第二次划分：$f_2 \geq 2$.

(3) 存在。例如选第一次划分为：$f_2 \geq 3$.

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(**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$|f(\mathbf{x}) - \mathcal{N}(\mathbf{x})| \leq \epsilon$。
(Lipschiz条件为：$\forall \mathbf{x}, \mathbf{y} \in [-1,1]^n$，$\exists \rho > 0$， s.t. $|f(\mathbf{x}) - f(\mathbf{y})| \leq \rho \|\mathbf{x} - \mathbf{y}\|_2$，其中 sgn(**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. *此处用于写解答(中英文均可)*

(1) diagram

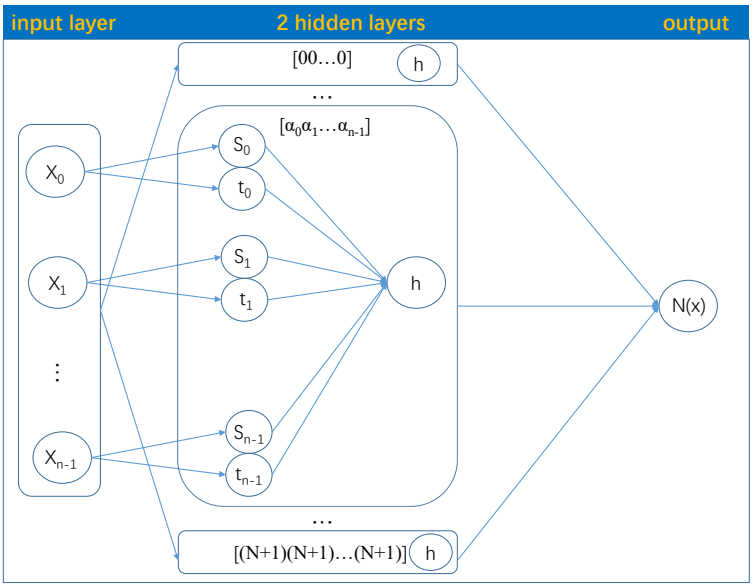


图 3: Neural Net Structure

(2) explanation

- 共有*1*个输入层$(x_0 x_1 \ldots x_{n-1})$，*1*个输出层，中间*2*层隐层构成$(N+1)^n$个单元，单元的索引为$[\alpha_0 \alpha_1 \ldots \alpha_{n-1}]$，*N*的取值为：

$$N = \lceil \frac{2}{(\frac{\varepsilon}{\sqrt{n}L})} \rceil \tag{3.1}$$

其中，*L*为*Lipschitz constant.*

- 输入层的x_i第$[\alpha_0\alpha_1\ldots\alpha_{n-1}]$个双隐层单元的中的$s_i, t_i$相连，其对应的权重和激活的阈值为：

$$W[s_i^{\alpha_0\alpha_1\ldots\alpha_{n-1}}]_{in} = 1 \tag{3.2}$$

$$W[t_i^{\alpha_0\alpha_1\ldots\alpha_{n-1}}]_{in} = -1 \tag{3.3}$$

$$\theta[s_i^{\alpha_0\alpha_1\ldots\alpha_{n-1}}] = \alpha_i\delta + 1 \tag{3.4}$$

$$\theta[t_i^{\alpha_0\alpha_1\ldots\alpha_{n-1}}] = -[(\alpha_i + 1)\delta + 1] \tag{3.5}$$

其中，

$$\alpha_i = \lfloor \frac{x_i}{N} \rfloor \tag{3.6}$$

- 第$[\alpha_0\alpha_1\ldots\alpha_{n-1}]$个双隐层单元的中的$s_i, t_i$相连，其对应的权重和激活的阈值为：

$$W[s_i^{\alpha_0\alpha_1\ldots\alpha_{n-1}}]_{out} = \frac{f(\alpha_0\delta + 1, \alpha_1\delta + 1, \ldots, \alpha_{n-1}\delta + 1)}{n} \tag{3.7}$$

$$W[t_i^{\alpha_0\alpha_1\ldots\alpha_{n-1}}]_{out} = -\frac{f(\alpha_0\delta + 1, \alpha_1\delta + 1, \ldots, \alpha_{n-1}\delta + 1)}{n} \tag{3.8}$$

$$\theta[h^{\alpha_0\alpha_1\ldots\alpha_{n-1}}]_{out} = f(\alpha_0\delta + 1, \alpha_1\delta + 1, \ldots, \alpha_{n-1}\delta + 1) \tag{3.9}$$

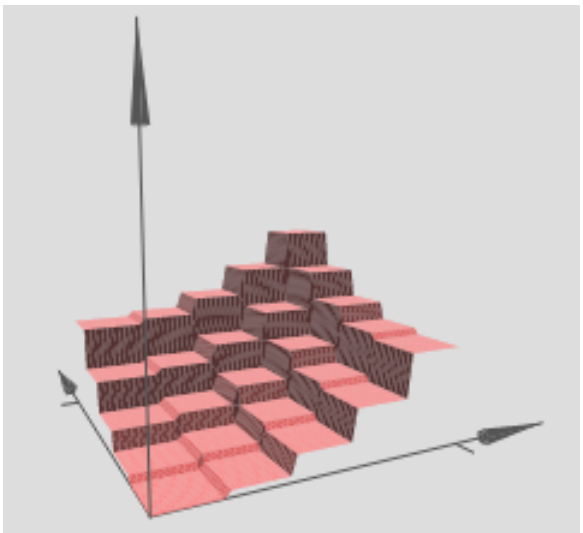
其中，

$$\delta = \frac{2}{N} \tag{3.10}$$

- 每个双隐层单元h到最终输出节点的权重都是*1.*

(3) proof

此网络结构用用逼近的方法去模拟函数$f(x)$(如下图所示).将输入空间划分为边长为δ的共计N^2个格子，每个格子对应第$[\alpha_0\alpha_1\ldots\alpha_{n-1}]$各双隐层单元，对于任意输入$x$,只有距离$(norm2)x$最近的一个格子被激活，产生接近$f(x)$的输出。 由此可以计算拟合误差：



$$|f(\mathbf{x}) - \mathcal{N}(\mathbf{x})| \leq L(\sqrt{n}\delta) = L(\sqrt{n}\frac{2}{N}) \leq \epsilon \tag{3.11}$$

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；

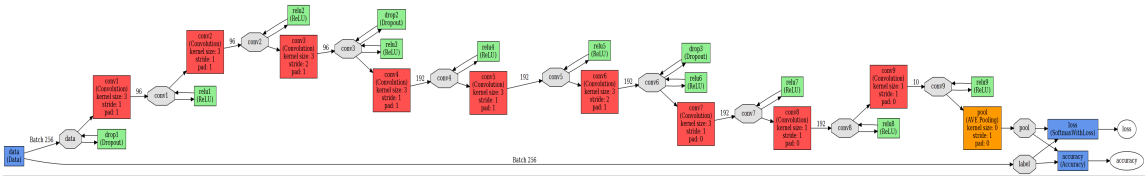


图 4: 待实现网络结构

(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)
- 卷积层用一个三维的kernel按照一定顺序去卷积输入层的每一个位置，对每个可能的位置遍历一遍完成一层卷积，此过程中每个位置的kernel共享权重。作用是提取可能的局部特征比如特定的形状（直角，边界）或者颜色等。
- 池化操作(Pooling)在尽可能保持图像特征的同时减少特征图像的维度，因此池化操作是一种“下采样”手段。常见的有Max Pooling，Mean Pooling等。作用主要是降维和防止过拟合。
- 调参过程中的几个主要参数及结果如表2： 可见，当$epoch = 20, lr = 0.0005$时模型测试准确率最高。此时如果再提高epoch则产生过拟合，提高lr则容易越过最优值。

表 2: 调参表

epoch	*lr*	*loss*	*accuracy*
2	0.001	1.306	53%
5	0.001	1.025	59%
5	0.002	1.1685	56%
10	0.001	0.842	61%
20	0.001	0.689	61%
20	0.0005	0.674	64%
25	0.0005	0.574	61%
50	0.0005	0.424	59%

(2) 实现了图4所示的网络结构，即原论文中的ALL-CNN-C. 论文中运用的优化方法是SGD，而本实验要求使用Adam优化器，所以两者的参数有所不同，主要设置的对比如表3：

表 3: 实验设置对比

	原论文设置	本实验设置
optimizer	SGD	Adam
training epochs	350	350
base learning rate	0.05	0.001
lr adaptive multiplier	0.1	0.1
lr adaptive milestones	[200,250,300]	[200,250,300]
wight decay	0.001	0.0001
momentum	0.9	0.9
dropout	[0.2,0.5,0.5]	[0.2,0.5,0.5]

其中，学习率(learning rate)大小的设置参考Adam论文，Adam中未列出的参数也都是按照论文中推荐值设置。实验在测试集的结果(分类准确率)如下表：实验过程中，每5个epoch做

表 4: 测试结果：分类准确率

Total Accuracy	85%
Accuracy of plane	81%
Accuracy of car	92%
Accuracy of bird	100%
Accuracy of cat	77%
Accuracy of deer	76%
Accuracy of dog	80%
Accuracy of frog	66%
Accuracy of horse	75%
Accuracy of ship	85%
Accuracy of truck	82%

一次测试，每20个iteration输出一个loss，所得loss和accuracy在训练过程中的变化曲线如图6和5所示：

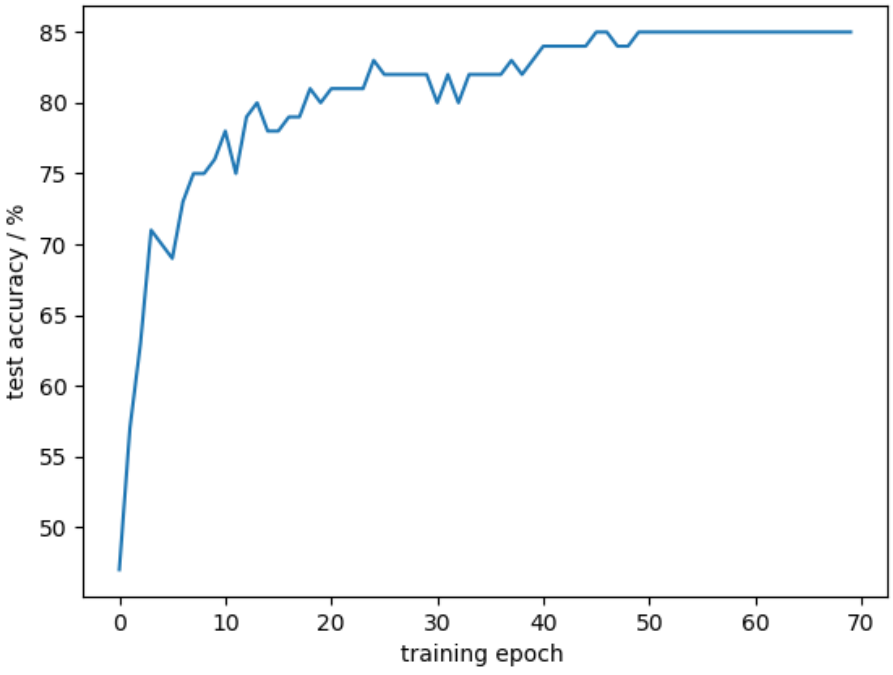


图 5: accuracy

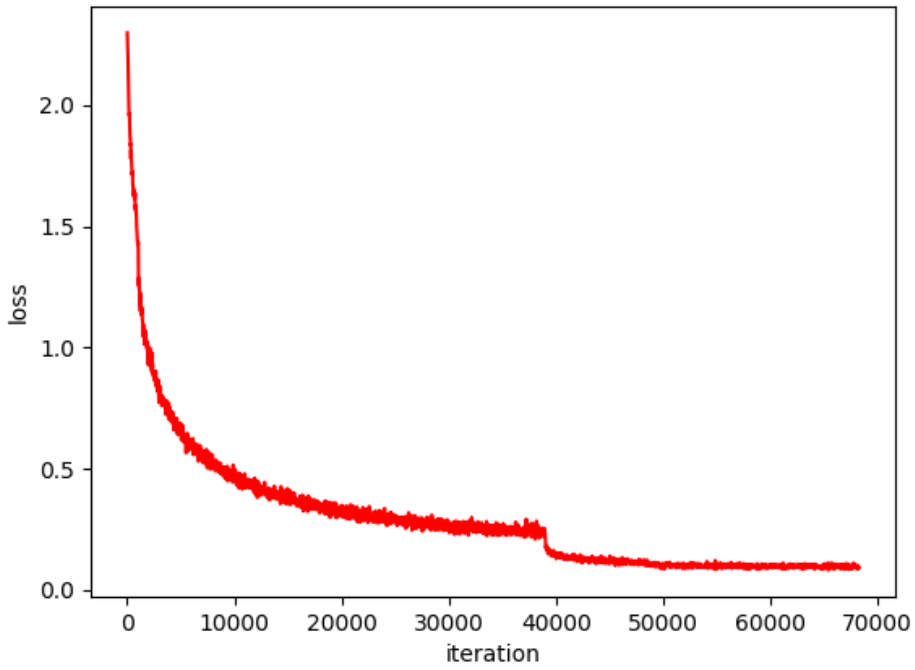


图 6: loss

(3) 调参过程中主要尝试调整weight decay的大小，以及在卷积层之间加入batchnorm，各种调整对模型的影响如表5.

表 5: 实验设置对比

模型调整	测试准确率
weight decay设为0.001	81%
weight decay设为0.0004	85%
卷积层之间加入batchnorm同时取消中间两层dropout	83%
卷积层之间加入batchnorm同时保留中间两层dropout	81%
learning rate 设为0.05	模型不收敛
weight decay 设为0.00001	模型不收敛

当学习率和l2正则化参数较大时，模型不收敛；当学习率调小后，模型收敛较快，此时影响模型泛化性能的主要是正则化参数，因此在结构一定的情况下主要对正则化参数进行调节(取值0.0004时也能取得85%的结果)。当尝试batch norm发现并未提升模型效果，可能的原因是模型中已经用dropout以及weight decay对模型结构进行了优化，因此猜想batch norm 成为冗余结构。当去加入batch norm时去除中间层的dropout，发现模型效果比直接加入batch norm更好，进一步验证了猜想。