机器学习导论
作业三

151250189, 翟道京, zhadj@smail.nju.edu.cn

2018 年 5 月 8 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) Implement decision tree, we can realize the $f(x, y, z)$ function. See the decision tree below,

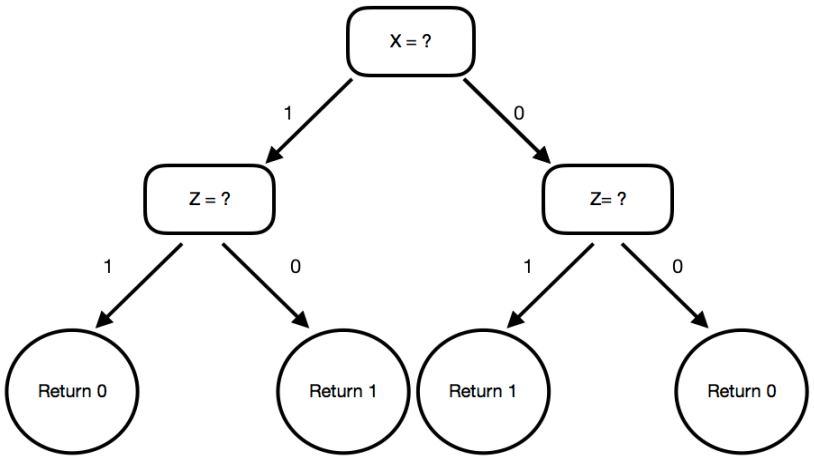

图 1: H: XOR decision tree

(2) Based on information gain, the decision tree of above dataset is shown below,

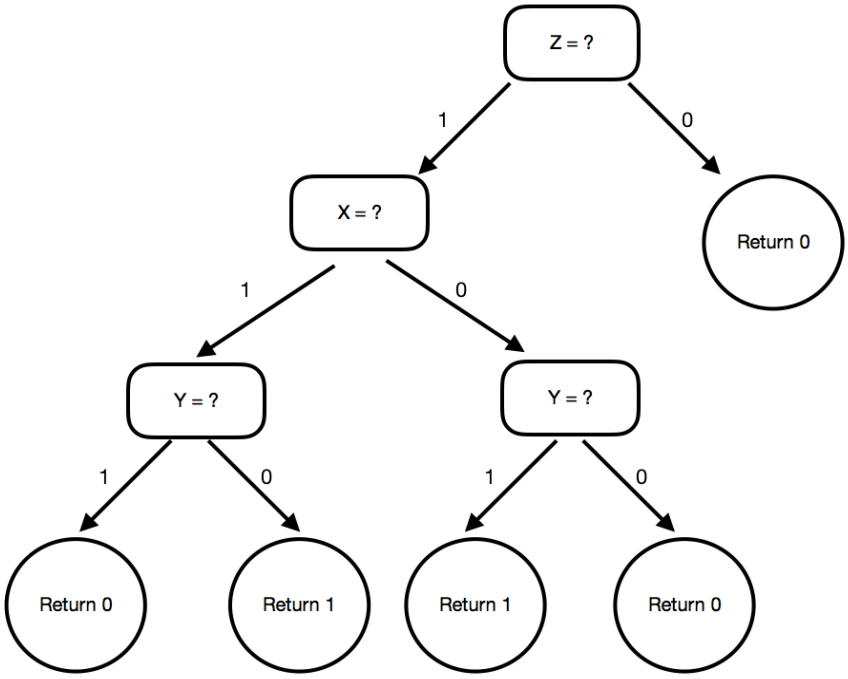

图 2: Decision tree

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) The entropy of the root is

$$Ent(D) = -\sum_{k=1}^{|\mathbf{y}|} p_k \log_2 p_k \tag{2.1}$$

$$= -(\frac{1}{2}\log_2 \frac{1}{2} + \frac{1}{2}\log_2 \frac{1}{2}) \tag{2.2}$$

$$= 1 \tag{2.3}$$

(2) The best division should purify the class of child node. Please so the following decision tree, its holds optimization division.

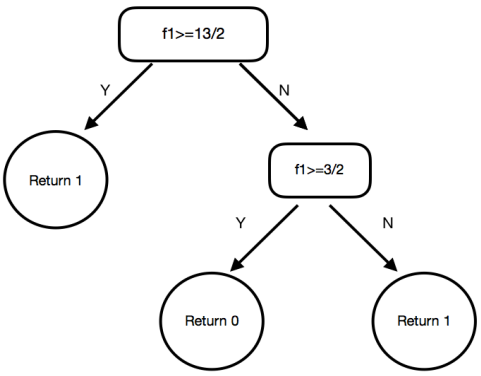

图 3: Decision tree

(3) It happens when someone choose the divisions $f_1 \geq \frac{9}{2}$, $f_2 \geq \frac{5}{2}$ *or* $f_2 \geq \frac{9}{2}$.

3 [25pts] Universal Approximator

已知函数 $f:[-1,1]^n \mapsto [-1,1]$ 满足 ρ-Lipschiz 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) 以 $f(\mathbf{x})$ 参数 $\mathbf{x}$ 是二维为例，我们的拟合思路是构造出一个个梯形体，每个梯形体的高度为权重，以此来逼近函数 $f(x)$，下图我们给出来二维情况下的示意图[1]

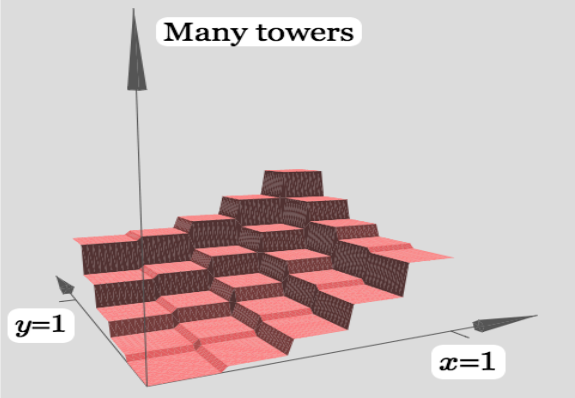

图 4: 二维情况下的拟合

从二维情况开始类比，我们将原函数定义域 $[-1,1]^n$ 划分为一个个边长为 Δx 的小超立方体，之后在每个立方体中，取中心点的函数值为构建的阶梯函数的取值。

而根据 ρ-*Lipschiz* 性质，

$$|f(\mathbf{x}) - f(\mathbf{x} + \mathbf{x})| \leq \rho||\Delta\mathbf{x}||_2 < \epsilon$$

对 n 维 $\mathbf{x}$，我们每个维度的的 Δx_i 在满足下列条件时，我们可以使用该阶梯函数逼近 $f(\mathbf{x})$。

$$|\Delta\mathbf{x}_i| < \frac{1}{n}\frac{\epsilon}{\rho}$$

我们通过这种方式，将定义域 $[-1,1]^n$ 分为了 $\frac{2}{|\Delta x_i|}$ 的超立方体，在每个超立方体中，我们取中心点 $\mathbf{x_0}$ 的值为该超立方体内部的取值，即 $f(\mathbf{x_0})$，我们的思路是先寻找 $\mathbf{x}$ 所在的超立方体，之后以该立方体内 $f(\mathbf{x_0})$ 来逼近 $f(\mathbf{x})$。

[1]Reference: A visual proof that neural nets can compute any function

以 $n=3$ 为例，我们给出了一个神经网络示意图

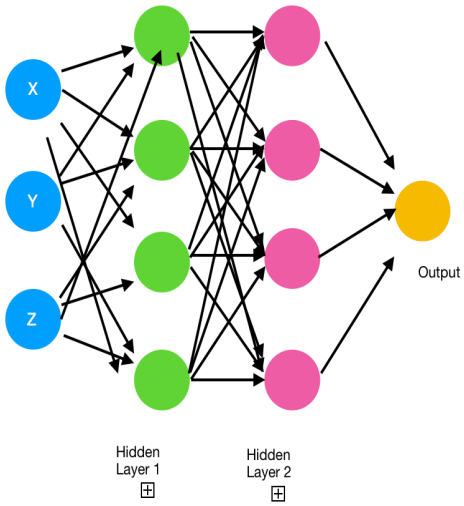

图 5: n = 3 时神经网络

(2) 第一隐层 $(\frac{n}{\Delta x}$ 个神经元*)*，判断某方向 x_i 位于哪个小方格中

$$sgn(-1<x_1<-1+\Delta x), sgn(-1+\Delta x<x_1<-1+2\Delta x), \cdots, sgn(1-\Delta x<x_1<1)\cdots$$

$$sgn(-1<x_2<-1+\Delta x), sgn(-1+\Delta x<x_2<-1+2\Delta x), \cdots, sgn(1-\Delta x<x_2<1)\cdots$$

$$sgn(-1<x_3<-1+\Delta x), sgn(-1+\Delta x<x_3<-1+2\Delta x), \cdots, sgn(1-\Delta x<x_3<1)\cdots$$

$$\cdots$$

$$sgn(x_n<-1+\Delta x), sgn(x_n<-1+2\Delta x), sgn(x_n<-1+3\Delta x), \cdots, sgn(x_n<1)$$

第一隐层实际上判断了 $(x_1, x_2, \cdots, x_n)$ 每一纬度上可以出现的区间范围，当成立时即输出 *1*，下面我们需要用第二隐层构成的区间。

第二隐层 *(*$(\frac{1}{\Delta x})^n$ 个神经元*)* 的作用将第一层的结果取交，输出最终的立方体区间位置（输出 *1*)。第二隐层中有 $(\frac{1}{\Delta x})^n$ 个神经元，每个神经元代表一个维度。最终我们根据输出结果，计算 $f(\mathbf{x_0})$ 来逼近 $f(\mathbf{x})$。

(3) 对于任意的 $\mathbf{x}\in[-1,1]^n$，我们输出 $\mathbf{x}$ 所在立方体中心取值 $\mathbf{x}_0$，则

$$||\mathbf{x}_0-\mathbf{x}||_2\le||\frac{\mathbf{x}}{2}||_2<||\Delta\mathbf{x}||_2$$

进一步的

$$||\Delta\mathbf{x}||_2<||\Delta\mathbf{x}||_1<\sum_{i=1}^{n}|\Delta x_i|<\frac{\epsilon}{\rho}$$

使用前面给的 *ρ-Lipschiz* 性质，得到 $|f(\mathbf{x})-\mathcal{N}(\mathbf{x})|\le\epsilon$，因此得证满足误差要求。

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7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；

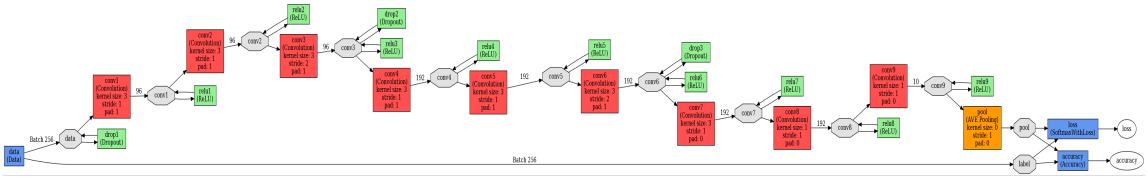

图 6: 待实现网络结构

(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)

1. In machine learning, a convolutional neural network (CNN) is a class of deep, feed-forward artificial neural networks that has successfully been applied to analyzing visual imagery. CNNs use a variation of multilayer perceptrons designed to require minimal preprocessing.

   After reading some online materials[2][3][4], and visually gain insight into the CNN from Adam Harley's Model: 3D Visualization of a Convolutional Neural Network, I have a general understanding of CNN Architecture.

   A CNN consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of convolutional layers, pooling layers, fully connected layers and normalization layers.

   - **Convolutional layers**

     The convolutional layer is the core building block of a CNN. They apply a convolution operation to the input, passing the result to the next layer. The convolution emulates the response of an individual neuron to visual stimuli. Each convolutional neuron processes data only for its receptive field. The layer's parameters consist of a set of learnable filters (or kernels), which have a small receptive field, but extend through the full depth of the input volume. During the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product between the entries of the filter and the input and producing a 2-dimensional activation map of that filter. As a result, the network learns filters that activate when it detects some specific type of feature at some spatial position in the input.

     Stacking the activation maps for all filters along the depth dimension forms the full output volume of the convolution layer. Every entry in the output volume can thus also be interpreted as an output of a neuron that looks at a small region in the input and shares parameters with neurons in the same activation map.

   - **Pooling layer**

     Pooling layer is a form of non-linear down-sampling. There are several non-linear functions to implement pooling including max pooling, average pooling, etc, and among tham max pooling is the most common. It partitions the input image into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum. The intuition is that the exact location of a feature is less important than its rough

[2]Reference: Wikipedia. Convolutional Neural Network

[3]Reference: CS231n Convolutional Neural Networks for Visual Recognition. Github

[4]Reference: zhihu

location relative to other features. The pooling layer serves to progressively reduce the spatial size of the representation, to reduce the number of parameters and amount of computation in the network, and hence to also control overfitting. It is common to periodically insert a pooling layer between successive convolutional layers in a CNN architecture. The pooling operation provides another form of translation invariance.

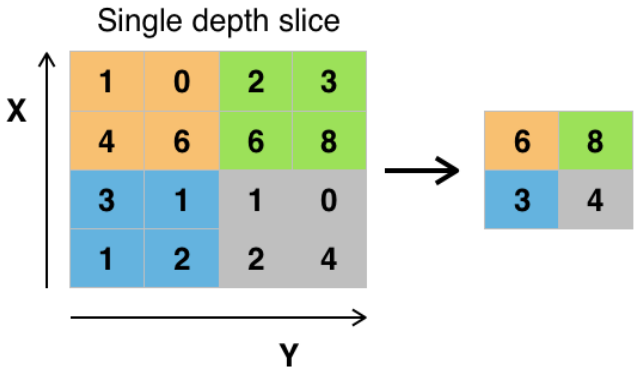

图 7: Max pooling layer sample (From Wikipedia)

- **ReLU layer**

  ReLU(Rectified Linear Units) layer applies the non-saturating activation function $f(x) = \max(0, x)$. It increases the nonlinear properties of the decision function and of the overall network without affecting the receptive fields of the convolution layer.

- **Fully connected layer**

  CNNs share weights in convolutional layers, which means that the same filter is used for each receptive field in the layer; this reduces memory footprint and improves performance.

- **Loss layer**

  The loss layer specifies how training penalizes the deviation between the predicted and true labels and is normally the final layer. Various loss functions appropriate for different tasks may be used there, including Softmax loss, Sigmoid cross-entropy loss and Euclidean loss.

2. I implement the `cifar10_tutorial.ipynb` program under the guidance of `PyTorch` tutorial. See the accuracy, loss of trained classifier in the following form.

表 2: cifar10_tutorial results

Epoch	Learning Rate	Momentum	Loss	Accuracy
2	0.001	0.9	1.280	55%
2	0.002	0.9	1.307	53%
10	0.001	0.9	0.873	63%
10	0.002	0.9	1.034	57%

As the `simple_cnn.py` is a simple program, the changes of Epoch and Learning Rate doesn't have a large improvement on the accuracy. In the next step, I build a more advanced program base on the literature.

(2) Refer to the literature, I build a convolutional neural network (CNN) in the program `complex_cnn.py`. I trained the classifier on GPU under different parameters, and the results are listed below.

表 3: CNN results

Epoch	Learning Rate	Weight decay	Loss	Accuracy
100	0.001	0.0	0.551	72%
100	0.002	0.0	0.203	79%
100	0.003	0.0	2.302	10%

We can see the new CNN significantly improved the classifier performance. However, when learning rate is a bit large(0.003), the accuracy decreases greatly. Actually, the loss function overshoot the minimum. It may fail to converge, or even diverge. In the next table we can see the details.

表 4: Results when learning rate = 0.003

Item	Accuracy	Item	Accuracy
plane	100%	dog	0%
car	0%	frog	0%
bird	0%	horse	0%
cat	0%	ship	0%
deer	0%	truck	0%

(3) From (2), we obtain different results with the literature. The reason is we don't use same number epochs due to the limited time. Also, we use different optimizer (Adam), as well as some parameters. Sometimes, stochastic parts of the model make differences.

In conclusion, in this project, I implement two CNN models and investigate the effect of different parameters on the training results. I find that to avoid overfitting and underfitting, to determine the learning rate, epoch and other parameters is one important part of Machine Learning. We could use former experience, have a careful try (when the parameter number is not large, we even could do grid search or cross validation). Other advanced methods are to learn.