Introduction to Machine Learning
Homework 3

181860155 朱晓晴
heloize@126.com

2020 年 11 月 16 日

1 [20pts] Decision Tree

(1) [10pts] Assume there is a space contains three binary features X, Y, Z and the objective function is $f(x, y, z) = \neg(x \text{ XOR } y)$. Let H denotes the decision tree constructed by these three features. Please answer the following question:

- Is function f realizable?
- If the answer is yes, please draw the decision tree H otherwise please give the reason.

(2) [10pts] Consider the following matrix:

$$\begin{bmatrix} 24 & 53 & 23 & 25 & 32 & 52 & 22 & 43 & 52 & 48 \\ 40 & 52 & 25 & 77 & 48 & 110 & 38 & 44 & 27 & 65 \end{bmatrix}$$

which contains 10 examples and each example contains two features x_1 and x_2. The corresponding label of these 10 examples as follows:

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

In this problem, we want to build a decision tree to do the classification task.

- Calculate the entropy of the root node.

- Building your decision tree. What is your split rule and the classification error?

解：(1) f 可实现，决策树如下图所示：

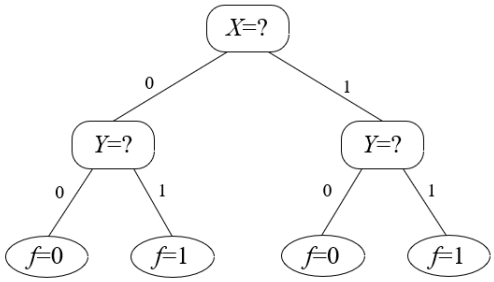

图 1: 基于函数 f 生成的决策树

(2) 根结点的信息熵为

$$\mathrm{Ent}(D) = -\sum_{k=1}^{2} p_k log_2 p_k = -(\frac{4}{10} log_2 \frac{4}{10} + \frac{6}{10} log_2 \frac{6}{10}) = 0.97095$$

对属性 x_1，将样本中 x_1 的取值从小到大排序，将 $\frac{x_1^i + x_1^{i+1}}{2}$ 作为侯选划分点，得到

$$T_{x_1} = \{22.5, 23.5, 24.5, 28.5, 37.5, 45.5, 50, 52, 52.5\}$$

依次计算

$$\mathrm{Gain}(D, x_1, x_1^i) = \mathrm{Ent}(D) - \sum_{v=1}^{2} \frac{|D^v|}{|D|} \mathrm{Ent}(D^v)$$

得到

$$\mathrm{Gain}(D, x_1, 22.5) = 0.07898, \mathrm{Gain}(D, x_1, 23.5) = 0.00740, \mathrm{Gain}(D, x_1, 24.5) = 0.00580$$
$$\mathrm{Gain}(D, x_1, 28.5) = 0.04644, \mathrm{Gain}(D, x_1, 37.5) = 0.12451, \mathrm{Gain}(D, x_1, 45.5) = 0.01997$$
$$\mathrm{Gain}(D, x_1, 50) = 0.09128, \mathrm{Gain}(D, x_1, 52) = 0.00740, \mathrm{Gain}(D, x_1, 52.5) = 0.14448$$

x_1 信息增益为 0.14448，对应划分点 52.5。

对属性 $x2$，有

$$T_{x_2} = \{26, 32.5, 39, 42, 46, 50, 58.5, 71, 93.5\}$$

$$\text{Gain}(D, x_1, 26) = 0.14448, \text{Gain}(D, x_1, 32.5) = 0.32193, \text{Gain}(D, x_1, 39) = 0.09128$$
$$\text{Gain}(D, x_1, 42) = 0.019973, \text{Gain}(D, x_1, 46) = 0.12451, \text{Gain}(D, x_1, 50) = 0.04644$$
$$\text{Gain}(D, x_1, 58.5) = 0.28129, \text{Gain}(D, x_1, 71) = 0.17095, \text{Gain}(D, x_1, 93.5) = 0.07898$$

x_2 信息增益为 0.32193，对应划分点 32.5。因此，以 x_2 为根节点划分属性。

对于下一层的左结点，划分出的样本标记都为 0，无需继续划分。对于右结点，可按照上述计算出 x_1 信息增益为 0.31128，对应划分点 37.5；x_2 信息增益为 0.20443，对应划分点 58.5。因此，以 x_1 为该节点划分属性。

对于下一层的左结点，划分出的样本标记都为 1，无需继续划分。对于右结点，可按照上述计算出 x_1 信息增益为 0.31128，对应划分点 45.5 和 52.5；x_2 信息增益为 1，对应划分点 58.5。因此，以 x_2 为该节点划分属性。

对于下一层的左结点，划分出的样本标记都为 0，无需继续划分。对于右结点，划分出的样本标记都为 1，无需继续划分。

决策树如下图所示：

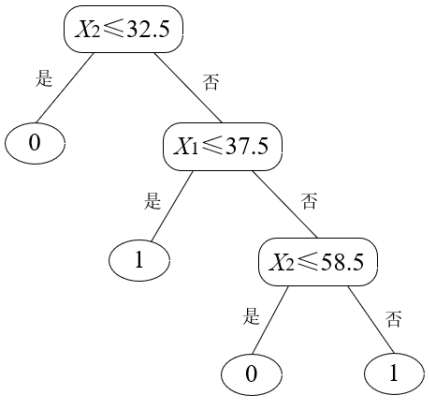

图 2: 基于信息增益生成的决策树

该决策树的分类误差为 0。

2 [20pts] Neural Network

Consider the following neural network, consisting of two input units, a single hidden layer containing two units, and one output unit:

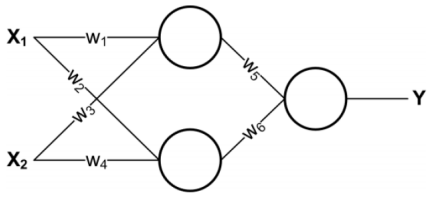

图 3: 神经网络

(1) [5pts] Assume that the network is using linear units: that is, for a given unit U, A is the vector of activations of units that send their output to U and W is the weight vector corresponding to these outputs, then the output of the unit is $W^{\mathrm{T}}A$. Let the weight values w_i be fixed, re-design the neural network to compute the same function without using any hidden units. Express the new weights in terms of the old weights.

(2) [5pts] Is it always possible to express a neural network made up of only linear units without a hidden layer?

(3) [10pts] Choose an activation function for each unit in the network which will cause this network to learn the same function that logistic regression would learn. Each unit should use a logistic, linear, or threshold activation function, with no constraints on the weights.

解：(1) 新的神经网络如下图所示：

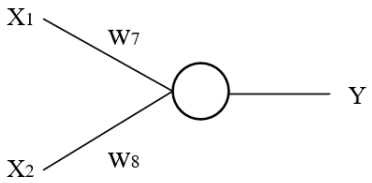

图 4: 仅使用线性单元的神经网络

其中 $w_7 = w_1w_5 + w_2w_6$，$w_8 = w_3w_5 + w_4w_6$。
(2) 总是能以没有隐藏层的形式表示一个仅由线性单元构成的神经网络。因为由线性单元构成的神经网络的输出总是能够用矩阵乘法表示，以题中的

神经网络为例，输出可以表示为：

$$\begin{aligned} Y &= w_5(w_1x_1 + w_3x_2) + w_6(w_2x_1 + w_4x_2) \\ &= (w_1w_5 + w_2w_6)x_1 + (w_3w_5 + w_4w_6)x_2 \\ &= \begin{bmatrix} w_5 & w_6 \end{bmatrix} \begin{bmatrix} w_1 & w_3 \\ w_2 & w_4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \end{aligned}$$

因此，如果网络有 n 个输入，分别为 $x_1, x_2, \ldots, x_n$，输出可以表示为：

$$Y = B_m B_{m-1} \ldots B_1 \begin{bmatrix} x_1 & x_2 & \ldots & x_n \end{bmatrix}^{\mathrm{T}}$$

其中 B_j 为每一层的权重构成的矩阵。那么，将各个输入 x_i 直接与输出层神经元相连，令其权重为 $B_m B_{m-1} \ldots B_1$ 乘积中第 i 列元素的值，即可得到符合题意的不含隐藏层的神经网络。

(3) 令隐藏层上下两个神经元分别为 E 和 F，输出层神经元为 G。

E 的激活函数为 $f_1(x) = x$，输出为 $f_1(W_1^{\mathrm{T}}A) = w_1x_1 + w_3x_2$，其中

$$W_1 = \begin{bmatrix} w_1 \\ w_3 \end{bmatrix}, A = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

F 的激活函数为 $f_2(x) = x$，输出为 $f_2(W_2^{\mathrm{T}}A) = w_2x_1 + w_4x_2$，其中

$$W_1 = \begin{bmatrix} w_2 \\ w_4 \end{bmatrix}$$

G 的激活函数为 $f_3(x) = (1 + e^{-x})^{-1}$，输出为

$$f_3(W_3^{\mathrm{T}}A_1) = (1 + e^{-(w_1w_5 + w_2w_6)x_1 + (w_3w_5 + w_4w_6)x_2})^{-1}$$

其中

$$W_3 = \begin{bmatrix} w_5 \\ w_6 \end{bmatrix}, A_1 = \begin{bmatrix} w_1x_1 + w_3x_2 \\ w_2x_1 + w_4x_2 \end{bmatrix} = \begin{bmatrix} W_1^{\mathrm{T}} \\ W_2^{\mathrm{T}} \end{bmatrix} A$$

3 [60 pts] Neural Network in Practice

In this task, you are asked to build a Convolutional Neural Networks (CNNs) from scratch and examine performance of the network you just build on **MNIST** dataset. Fortunately, there are some out-of-the-box deep learning tools that can help you get started very quickly. For this task, we would like to ask you to work with the **Pytorch** deep learning framework. Additionally, Pytorch comes with a built-in dataset class for MNIST digit classification task in the **torchvision** package, including a training set and a validation set. You may find a pytorch introduction at here. Note that, you can use CPU or GPU for training at your choice.

Please find the detailed requirements below.

(1) [5 pts] You are encouraged to implement the code using *Python3*, implementations in any other programming language will not be judged. Please name the source file (which contains the main function) as *CNN_main.py*. Finally, your code needs to print the performance on the provided validation set once executed.

(2) [20 pts] Use any type of CNNs as you want and draw graphs to show your network architecture in the submitted report. You are encouraged to try more architectures.

(3) [25 pts] During training, you may want to try some different optimization algorithms, such as SGD, Adam. Also, you need to study the effect of learning rate and the number of epoch, on the performance (accuracy).

(4) [10 pts] Plot graphs (learning curves) to demonstrate the change of training loss as well as the validation loss during training.

实验报告

1 实验环境

本次实验的环境如下：

Python 3.8.3

PyTorch 1.7.0

CUDA 11.0

2 卷积神经网络

本次实验采用了 Gradient-Based Learning Applied to Document Recognition 提出的 LeNet-5 卷积神经网络，其结构如下图所示：

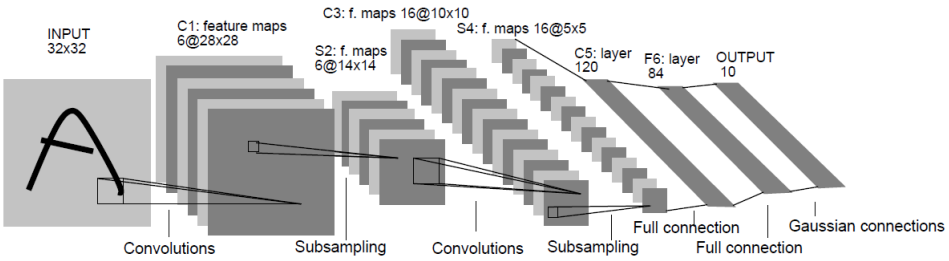

图 5: LeNet-5 结构

由于 MNSIT 数据集提供的样本数据以 $28 \times 28 \times 1$ 的形式组织，因此需要对 LeNet-5 结构进行调整，调整后的结构如下：

输入：$28 \times 28 \times 1$ 的图像

C1：卷积层，使用 6 个 3×3 的滤波器，步长为 1，padding 采用 Same 方式，输出 $28 \times 28 \times 6$ 的图像。

S2：池化层，使用 2×2 的滤波器，步长为 2，输出 $14 \times 14 \times 6$ 的图像。

C3：卷积层，使用 16 个 3×3 的滤波器，步长为 1，padding 采用 Valid 方式，输出 $10 \times 10 \times 16$ 的图像。

S4：池化层，使用 2×2 的滤波器，步长为 2，输出 $5 \times 5 \times 16$ 的图像。

C5 和 F6：全连接层，分别有 120 个和 84 个神经元。

输出层：10 个神经元

3 代码实现

基于上一节中的理论，本次实验设计了如下数据结构和模块划分：

LeNet5 类：用于构建 LeNet-5 卷积神经网络，卷积层和全连接层使用 ReLu 作为激活函数，代码实现见图 6。

Train 函数：通过学习 60000 个样本来训练 CNN。

Test 函数：用训练出的 CNN 测试 10000 个样本

```
# LeNet-5: CONV-->POOL-->CONV-->POOL-->FC-->FC-->OUTPUT
self.conv1 = nn.Conv2d(in_channels=1, out_channels=6,
                       kernel_size=5, stride=1, padding=2)
self.pool1 = nn.MaxPool2d(kernel_size=2, stride=2)
self.conv2 = nn.Conv2d(in_channels=6, out_channels=16,
                       kernel_size=5, stride=1, padding=0)
self.pool2 = nn.MaxPool2d(kernel_size=2, stride=2)
self.fc1 = nn.Linear(5 * 5 * 16, 120)
self.fc2 = nn.Linear(120, 84)
self.fc3 = nn.Linear(84, 10)
```

图 6: LeNet-5 代码实现

代码的执行流程大致如下：

1. 读取 MNIST 训练集和测试集。
2. 定义 LeNet5 类，并创建 CNN 实体 net。
3. 确定损失函数和优化算法。
4. 训练卷积神经网络。
5. 测试并输出 CNN 的性能数据。

4 参数选取

实验鼓励尝试不同的优化算法和超参数，本节将对以上 2 个方面进行讨论，并选出相对更优的配置。激活函数和损失函数则采用神经网络中常见的 ReLU 和交叉熵损失。

4.1 优化算法

给定以下参数

LR=0.01

BATCH_SIZE=50

分别使用 SGD 和 Adam 优化算法训练模型，得到的测试准确率如下表所示：

Accuracy	epoch=1	epoch=2	epoch=5
SGD	97.66%	98.22%	98.90%
Adam	97.81%	97.61%	97.90%

表 1: 使用 SGD 和 Adam 优化的准确率

根据上表数据，易知在 LeNet-5 结构下，SGD 优化算法的表现较 Adam 优化算法更好。同时，根据 traning loss 和 validation loss 的图像（未给出），采用 Adam 算法进行优化时，模型更易、更早出现过拟合。因此，本次实验选用 SGD 作为优化算法，算法参数 momentum=0.9。

4.2 超参数

采用 LeNet-5 结构，令 BATCH_SIZE=50，对不同的学习率和 epoch 进行研究，得到的测试准确率如下表所示：

Accuracy	epoch=1	epoch=2	epoch=3	epoch=4	epoch=5	epoch=8
lr=0.005	96.70%	97.86%	98.20%	98.51%	98.53%	98.72%
lr=0.01	97.66%	98.22%	98.82%	98.91%	98.90%	98.84%
lr=0.02	97.82%	98.05%	98.73%	98.44%	98.87%	99.04%
lr=0.05	98.07%	98.42%	97.82%	98.62%	98.24%	98.75%

表 2: 不同学习率和 epoch 下的准确率

根据上表中的数据，易发现当学习率一定时，准确率基本随着 epoch 的增大而增大。考虑到 epoch 越大时，准确率增加得越缓慢，本次实验选用的参数为

LR=0.01

EPOCH=5

BATCH_SIZE=50

5 实验结果

由第 4 节的讨论结果，本次实验最终采用 LeNet-5 结构、SGD 优化算法和以下参数，并在本节给出训练出的模型的性能指标。

LR=0.01

EPOCH=5

BATCH_SIZE=50

模型在 10000 个样本的 MNIST 数据集上的准确率为 98.90%（见表 2）。

训练时，每学习完 1 个 mini-batch（即 50 个样本），会输出这个 batch 的训练损失。同时，会选取测试集中的 100 个样本，由当前模型进行预测，得到验证损失。最终，得到如下的 training loss 和 validation loss 曲线。

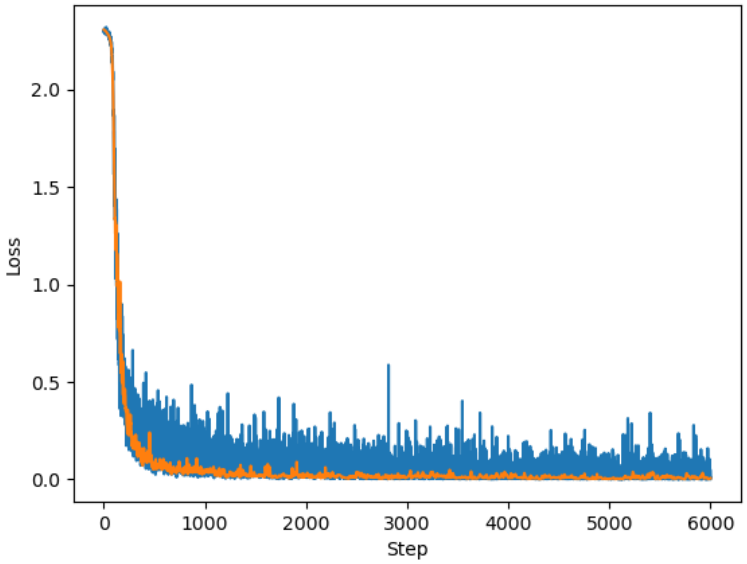

图 7: 学习曲线

参考文献

[1] 周志华. 机器学习 [M]. 清华大学出版社, 2016.

[2] 卷积神经网络
https://www.jianshu.com/p/c0215d26d20a

[3] 常见卷积神经网络
https://www.cnblogs.com/guoyaohua/p/8534077.html

[4] LeNet-5 实现 MNIST 分类（Tensorflow2 实现）
https://www.cnblogs.com/nickhan-cs/p/13340869.html

[5] 随机梯度下降算法
https://blog.csdn.net/qq_38150441/article/details/80533891

[6] cifar10_tutorial
https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html

[7] 根据损失曲线诊断神经网络
https://blog.csdn.net/sjwzdh/article/details/103523465