问题求解与实践　课程作业

518030910092 谢哲

摘要

在本作业中，使用 C++ 对 kaggle 中的共享自行车使用情况数据集进行了分析。首先，使用一个特定的类对数据进行读取和存储，并剔除了不符合条件的脏数据。使用 FLTK 对数据使用折线图、饼状图、比例图、柱状图等进行可视化。最后，使用**自己编写的神经网络框架**和多元线性回归方法，分别根据环境对自行车使用情况进行了预测，神经网络的预测方法效果良好。**报告的最后给出了 demo 的使用方法。**

1　数据读入

1.1　任务描述

该数据集存储了某城市 2011 和 2012 年的每小时、每天的共享自行车的使用量，和当时的日期、时间、季节、节假日、工作日、温度、湿度、天气、注册用户数、非注册用户数等数据。

1.2　解决方案

为了将数据读入进行分析与预测，首先进行如下三个步骤对数据进行读入与预处理。

(1) **数据读入**　在数据读入部分中，使用了一个 **Data_Reader** 类处理数据的读入。并将原始的.csv 数据使用 fstream 读入并以逗号分割后,将不同类别的数据以字符串方式存储,以字符串向量的方式存入一个 raw_data 向量容器中。

(2) **脏数据处理**　读入原始数据后先进行脏数据处理，脏数据处理的具体方法详见第 2 节。

(3) **数据存储**　在处理脏数据之后，进入数据读取步骤。在 Data_Reader 类中，定义了一个类 Bikie_Data 来存储不同格式的数据，包括日期、天气、温度、用户量等。并将所有剔除脏数据后的 raw_data 数据输入格式转换函数，将字符串类别转化为相应的特定类别数据，存入 Bike_Data 类变量，装入 Bike_Data 类别的 processed_data 向量容器中。以线性方式完成数据集的存储。

1.3　结果

数据集中的两个数据文件 data.csv 和 hour.csv 均能被正常读取并存入向量容器内。容器内的数据可以被快速、方便地访问和查找，为之后的可视化和数据预测提供了有力的支持。

2　脏数据处理

2.1　任务描述

该共享自行车数据集提供了包括日期、相对温度、相对湿度、季节、天气和使用量等多方面的信息。而在此之中可能包括各种各样的脏数据，包括格式不正确、数据范围不合理、数据自相矛盾和数据本身没有实际价值等

多方面的情况，需要将其进行脏数据处理之后，再进行可视化和数据预测的操作，避免脏数据带来的影响。

2.2 解决方案

对于数据集中的每一条记录，由于数据类型多种多样，而且表达的意义不同，所以不可能使用统一的方法来确定所有数据的脏数据标准。因此必须对于每一类别的数据分别定义脏数据标准。经过对数据的整体分析，确定的脏数据标准如下。

对于序号类别，由于其在可视化和预测中不起到任何作用，所以一律删除；对于日期类别，需确定其格式是否正确；对于季节、年份、月份、节假日、时间、工作日和天气状况类别，需保证其数据范围在数据集给定的范围之内；对于相对温度和体感相对温度，由于体感温度更具实际价值，仅保留体感相对温度；对于相对湿度，其正常范围均处于 0.1 ~ 1.0 之间，剔除之外的部分；对于用户总数、非注册用户数和注册用户数，需保证总数等于非注册和注册数量之和的客观规律。对于其余类别数据，需保证输入格式正确。对于任何非法数据（即不符合前述要求的数据），由于无法恢复，**应当直接剔除该条记录**。

2.3 结果

经过程序对源数据的格式筛选和格式化数据的数值筛选后，共剔除了 26 条脏数据。剔除了脏数据之后，数据被直接保存在 processed_data 向量容器中，为之后的可视化和数据预测的有效性和准确性提供了必不可少的前提条件。

3 数据可视化

3.1 任务描述

数据集一共包含了包含时间、用户数量、天气、温度等多个维度的近 20000 条数据，并且数据间可能存在一定的关联性。为了将这些数据以更直观的方式展现出来，可以使用 FLTK 库，采用数据可视化的方法来清晰地展现这些数据的某些特点。

3.2 解决方案

对于该数据集，存在多种数据可视化方法，包括条形图、折线图、饼状图、比例图等，而且每种类别的图形均被使用了多次。考虑使用面向对象的方式，将每种类别的图都写为一个类。为了使用方便，这些类都继承了课程中使用的 Graph.h 中的 Shape 类，可以使用统一的方式绑定到 Simple_Window.h 中的 Simple_Window 显示。再统一通过 Show_Data 类进行封装，方便用户调用。

在每个类中，重载了 draw_lines 函数，并且根据数据可视化方法的实际情况，增加了其它的一些函数进行重载。例如，在折线图中增加了设定坐标轴标签和范围的函数，在饼状图中增加了设定系列标签的函数等。可视化部分的继承和调用结构如图 1所示。相较于 FLTK 自带的 Fl_Chart 类，自定义的可视化图标类可以同时显示多个系列的折线图，并且有更为准确的数据标签，功能更为丰富，生成的图表更为美观。

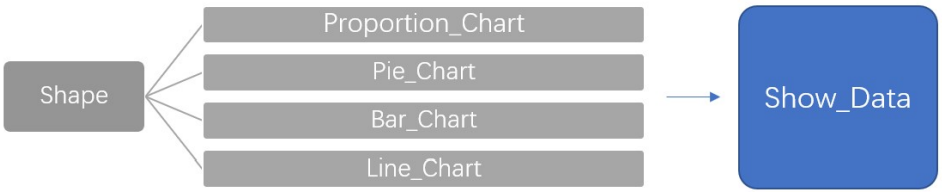
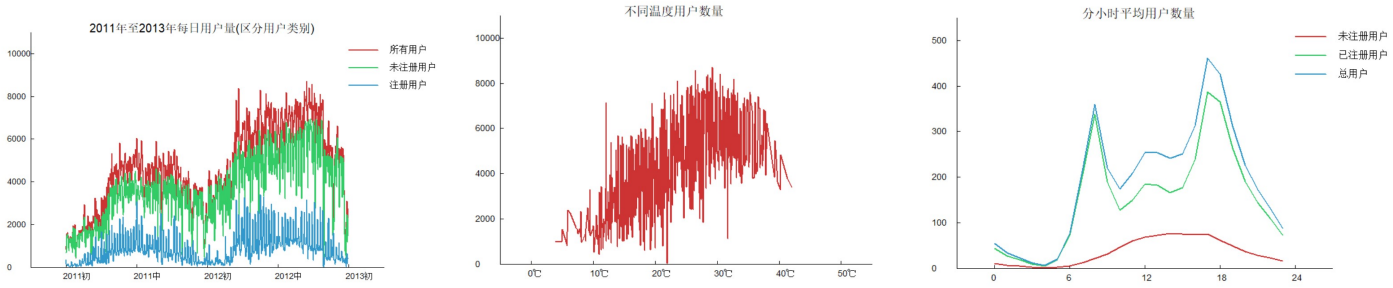


图 1: 可视化部分结构

对于每日的用户量的变化情况，使用折线图来展示，并使用不同的颜色来区分不同的系列（注册、未注册、总数量），如图 2所示。对于注册于未注册用户的比例情况，使用饼状图来反应，并使用比例图来反应其随着时间的比例变化情况。对于每天各时段的用户，使用折线图来反应其随时间的变化情况。对于不同气候条件下的用户数量，使用折线图和条形图进行可视化。

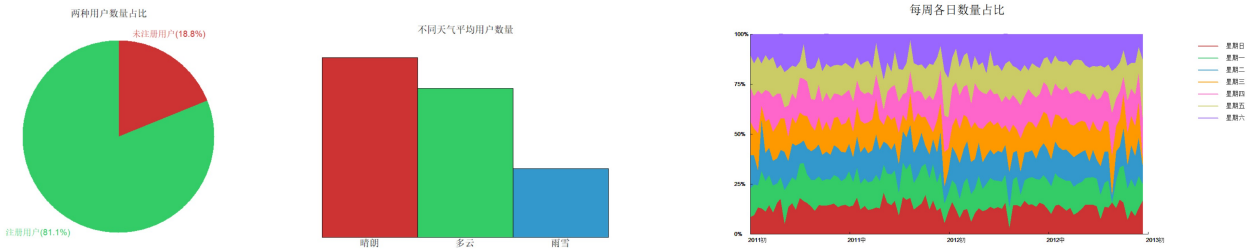
3.3 结果

由于篇幅有限，下图中仅展示部分可视化图片，全部可视化结果可以按照文末的说明运行 demo 查看。通过这些数据，不难分析出随着时间、温度变化，用户数量和比例的变化。



(a) 每日各类别用户量折线图　(b) 不同温度用户量折线图　(c) 每小时平均用户数量变化折线图

图 2: 折线图



(a) 用户比例饼状图　(b) 不同天气的用户数量条形图　(c) 每周各日用户比例变化比例图

图 3: 饼状图、条形图和比例图

4 数据预测

4.1 任务描述

本数据集的预测目标是共享自行车的总使用人数。预测的依赖因素包括年份、月份、天气、温度和湿度等信息。由于年份等与时间相关的因素是客观已知的，且天气、温度和湿度等信息是可以通过天气预报技术预测的，所以根据这些信息预测出当天的总单车使用量具有一定的可行性和实用价值。根据当初设定的目标，先根据随机划分的数据集与测试集，在预测模型生成之后，将对所有两年的每日使用数据与每小时使用数据进行预测。之后将前 7 个季度划分为数据集，后 1 个季度作为测试集，预测最后一个季度的每日单车使用情况。

4.2 解决方案

由于本任务存在多个因素，所以使用之前所学的一元线性回归等一元拟合方法是不可行的。于是考虑采用多元线性回归和神经网络的方法进行拟合。

(1) **多元线性回归** 多元线性回归与一元线性回归类似。对于其中的一个样本 $(x_1, x_2, \cdots, x_n)$，其对应的多元线性回归值为 $\hat{y} = b_0 + b_1x_1 + b_2x_2 + \cdots + b_nx_n$，其原理为优化参数 $\hat{b_i}$，最小化这些参数表述的回归超平面上的预测值与真实值的残差平方和 $Q(b_0, b_1, \cdots, b_n)$。通过数学方法，可以得到如下的线性方程组，可以求得参数的理论值。

$$
\begin{aligned}
b_0 &= \overline{y} - b_1\overline{x_1} - b_2\overline{x_2} - \cdots - b_n\overline{x_n} \\
s_{11}b_1 + s_{12}b_2 + \cdots + s_{1n}b_n &= s_{1y} \\
&\cdots \\
s_{n1}b_1 + s_{n2}b_2 + \cdots + s_{nn}b_n &= s_{ny}
\end{aligned}
$$

其中 $s_{jk} = \sum_{i=1}^{n}(x_{ji} - \bar{x_j})(x_{ki} - \bar{x_k}), s_{jy} = \sum_{i=1}^{n}(x_{ji} - \bar{x_j})(y_i - \bar{y})$。将实际值通过公式计算，代入线性方程组，使用高斯消元法求解，即可得到结果。

(2) **神经网络** 由于在本数据集中，各种因素与最终结果的关系不一定是线性关系，所以需要进行多元函数的高次拟合，神经网络便是这样一种能够拟合多元高次函数的简洁、高效的方法。直接调用一些线程的机器学习库进行训练不是本课程的初衷，于是考虑**从零实现一个简单易用的 C++ 神经网络框架**，对数据进行预测。

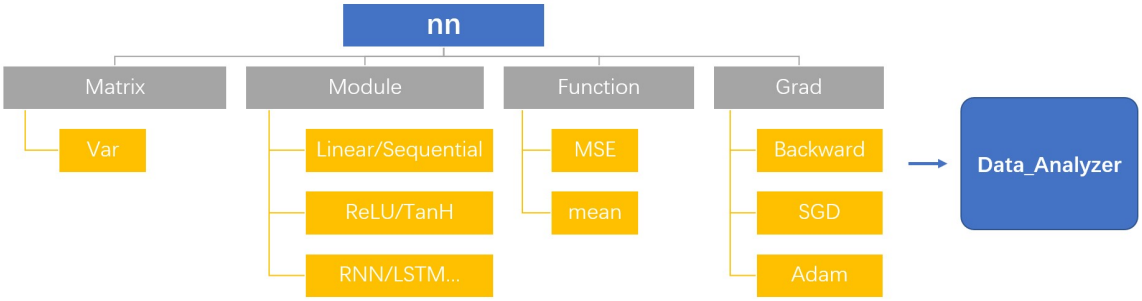


图 4: 神经网络框架结构

本人实现的神经网络框架为静态神经网络，它的结构如图 4所示。所有的类和函数都被包括在了一个名为 nn 的 namespace 中，这些类的结构全部声明在 nn.h 中方便查阅。由于神经网络运算本质是矩阵运算，所以定义了一个 Matrix 类完成矩阵的基本运算，这部分的内容定义在 nn_matrix.cpp 中。Var 类为神经网络计算图的基本单元，内部结合了 Matrix 类，记录了结点数据、梯度数据、优化器参数和计算图中的子节点的指针，并提供了 backward 函数和 optim 函数以进行反向传播和参数优化，这部分的内容定义在 nn_var.cpp 中。

为了方便用户实现计算图的快速搭建，神经网络框架充分使用 C++11 中的特性。例如，为了防止作用域内的结点在作用域外被析构，从而导致计算图失效，使用智能指针 std::shared_ptr 实现堆内存管理，保证所有计算图结点在运行过程中始终有效；为了使得多项式计算过程中，右值计算结果不失效，定义了移动赋值函数和右值引用类型的重载函数。

由于计算图的构建，前向传播和反向传播的过程是简单的，只需使用 DFS 遍历计算图，递归计算前向传播值和梯度值即可。Module 模块内置了 Linear、RNN、LSTM 等常用网络结构和 ReLU、TanH、Sigmoid

等激活函数，并提供了 Sequential 类来将这些结构顺序串联起来以快速搭建神经网络。Module 模块内置了一个 forward 函数，包括了前向传播的计算过程，不同的神经网络只需修改其前向传播过程，可以根据现有的相关资料和论文得到其前向传播计算公式。反向传播之后，使用 SGD 或 Adam 优化器即可优化权重参数，以训练神经网络。反向传播和优化器的部分定义在 nn_grad.cpp 中。

在训练每日用户数据预测过程中，使用的网络结构为 4 层全连接神经网络，输入层神经元个数为特征数（8 个或 9 个，视具体训练数据而定）; 第一隐藏层神经元个数为 32，激活函数为 ReLU；第二隐藏层神经元个数为 64，激活函数为 ReLU；输出层神经元个数为 1。在训练每小时数据预测过程中，为增强其学习能力，增加为 6 层神经网络，隐藏层神经元数分别为 16，16，32，4，激活函数是 ReLU。

经过实验发现 Adam 优化器收敛速度更快，效果更佳，于是在训练过程中使用 Adam 优化器进行优化。其简化后的伪代码如下：

算法 1 Adam 优化器

输入：学习率α, β_1, β_2**，函数**$f(\theta)$**，初始参数**θ_0

$m_0 \leftarrow 0, v_0 \leftarrow 0, t \leftarrow 0$
while θ_t 更新 **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$
    $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$
    $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$
end while
return θ_t

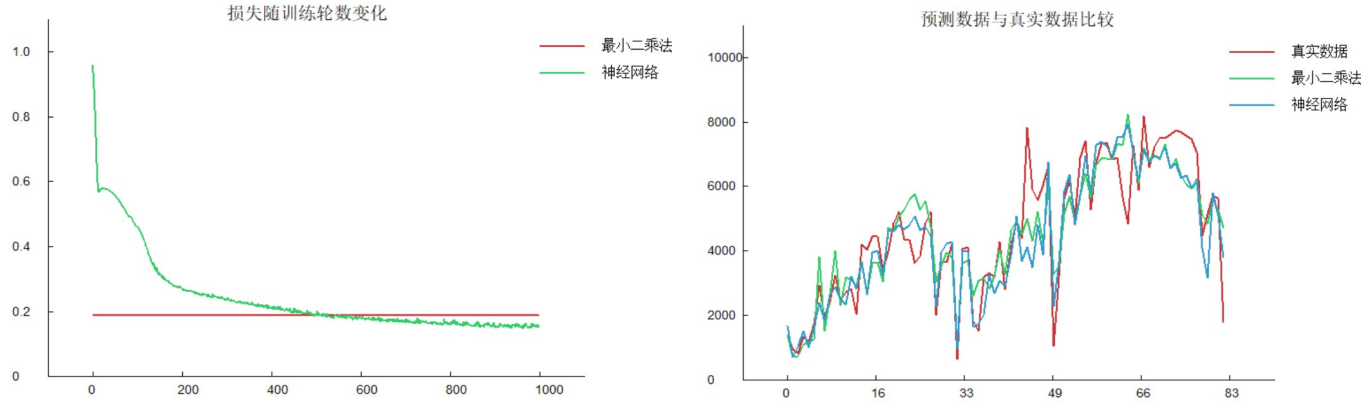
在得到预处理的数据之后，为了进行多元线性回归或神经网络训练，需要项将其先强制转化为 double 类型的向量。每小时的数据和每日的数据相比，要多出一个“时间”变量，因此预测时的输入需要多出一个维度。在确定输入、输出向量之后，根据任务描述中的要求划分训练集和测试集。随机划分时，按照 9:1 的比例划分训练集与测试集。将训练集放入定义好的神经网络内训练，训练集经过随机排序后，按照每批 100 个放入网络前向与反向传播。损失函数采用误差率，优化器采用 Adam，学习率设定为 0.005。最后评判在测试集上的误差率，（由于实际情况，将原任务描述中的准确率评判变为误差率评判，且为了防止某次误差率过大限制了其范围），误差率的定义如下：

$$\text{误差率} = \min\left(\frac{|\text{当日预测用车次数} - \text{当日实际用车次数}|}{\text{当日实际用车次数}}, 1\right) \times 100\%$$

4.3 结果

4.3.1 随机划分预测每日数据

使用最小二乘法预测每日总用车辆在测试集上的误差率为 18.87%；使用神经网络预测每日总用车辆在测试集上的误差率为 16.39%，预测结果优于最小二乘法的预测结果。为了体现其训练效果，图 5展示了随机划分训练集和测试集情况下，随着训练轮数的增加的误差率的变化情况和在预测集上两种不同方法的预测表现。可以发现，随着训练轮数的增加，神经网络的误差率逐渐减小，最终低于最小二乘法的误差率。而在预测集上的表现可以发现二者都能较好地拟合预测集上的数据，且神经网络在大多数情况下都要略胜一筹。



(a) 训练损失随轮数变化情况　　(b) 预测情况与真实情况比较

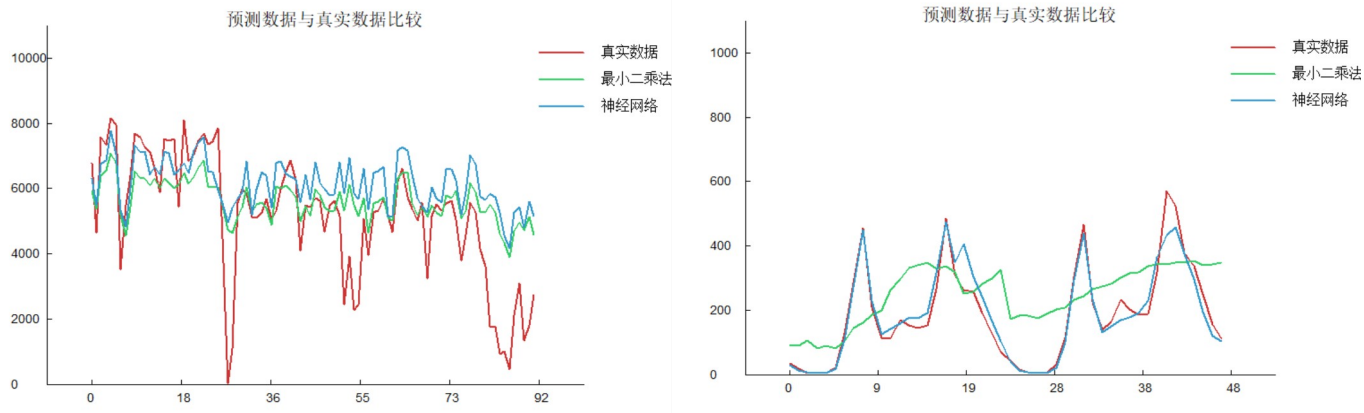
图 5: 随机训练集下的预测情况

4.3.2 预测最后一个季度的使用情况

预测最后一个季度的使用情况时，使用最小二乘法的误差率为 26.16%，使用神经网络的误差率为 29.61%，略高于最小二乘法的误差率。然而，根据训练时输出的损失率，可以发现其在测试集上的损失远小于该数字，这也说明神经网络预测时可能存在过拟合的情况。

4.3.3 预测每天内各小时的使用情况

使用最小二乘法的误差率为 59.98%，使用神经网络的误差率为 28.75%，远优于最小二乘法。从图 6(b) 中的预测结果与真实结果比较情况，可以发现最小二乘法的拟合曲线更趋向于线性，而神经网络的拟合曲线则是明显是非线性的，贴近于真实曲线，其学习能力较最小二乘法更强。



(a) 预测最后一个季度用户变化情况　　(b) 预测某天每小时用户变化情况

图 6: 特定训练集下的预测情况

4.3.4 结论

所有模型的最优误差率均低于 30%，达到了当初设定的目标，可以认为该模型具有较好的效果。

5 Demo 使用说明

5.1 本文作者的编译和运行环境

- 编译环境：Visual Studio 2019，C++14，fltk-1.3.5，x86，Release。
- 运行环境：Windows 10，Intel Core i7-8th，8GB RAM。
- 编码：所有包含汉字的文件均使用 UTF-8 with BOM 编码保存，请使用该编码打开文件，避免中文乱码。

5.2 运行步骤

1. 使用 Visual Studio 2019（推荐）打开根目录下的 PSP-Final.sln 文件，选择 x86，Release 模式，点击生成解决方案即可编译。如果使用其它编译工具，可以自行编译 PSP_Final 文件夹下：demo.cpp、data_analyzer.cpp、data_reader.cpp、show_data.cpp、./fltk/Graph.cpp、./fltk/GUI.cpp、./fltk/Simple_window.cpp、./fltk/Window.cpp、./nn/nn_functions.cpp、./nn/nn_grad.cpp、./nn/nn_matrix.cpp、./nn/nn_module.cpp、./nn/nn_tensor.cpp、./nn/nn_var.cpp，设定-std=c++14，并链接 FLTK 库。
2. 数据读入和脏数据处理：编译并运行程序后，即自动读入全部数据，并且立即自动完成脏数据的处理。
3. 可视化：按照提示，输入 1，按下回车，根据提示选择需要查看的可视化图像。查看完成后，点击窗口右上角的 Button，即可返回选择界面。
4. 数据预测：按照提示，输入 2，按下回车，根据提示选择需要预测的数据内容。等待一段时间后会自动弹出预测的误差率结果。

6 总结

通过实现数据集的读入、脏数据处理、可视化和数据预测，我学习了许多数据处理和可视化方面的实用的技巧。如今是大数据时代，对数据分析的能力非常重要。通过这次大作业，我加深了对数据分析的理解和认识。

此外，通过实现 FLTK 可视化和自己手动实现神经网络框架的过程，我认识到了一个好的程序框架对于程序的可维护性和可扩展性的重要意义。相比于重构的时间，在写代码之前认真思考程序的框架，能够起到事半功倍的神奇效果。

通过这次的作业，我还认识到神经网络并非万能的。当然在多数情况下，它比一些线性拟合方法得到的结果要更有，但是它也存在过拟合、训练速度慢等问题。例如在第 2 个任务中，神经网络的预测结果甚至不如最小二乘法的预测结果；但是在最后一个任务中，由于预测的曲线明显是非线性的，所以神经网络的预测结果远远优于最小二乘法的预测结果。这说明我们需要依据实际情况，选择合适的数据预测的方法。

7 致谢

感谢在问题求解与实践课上陈雨亭、沈艳艳老师的认真教学和助教课后的耐心辅导。

感谢 PyTorch 开源动态神经网络框架，为本人实现自己的 C++ 神经网络前端提供了思路。

8 参考文献与相关代码

[1] PyTorch 神经网络框架文档. https://pytorch.org/docs/stable/index.html.

[2] 本文中作者本人实现的神经网络框架 myNN. https://github.com/ligongzzz/myNN.