基于 Pytorch 的深度学习实践与作业——指导手册

李传艺

2023.07.03

一、实践

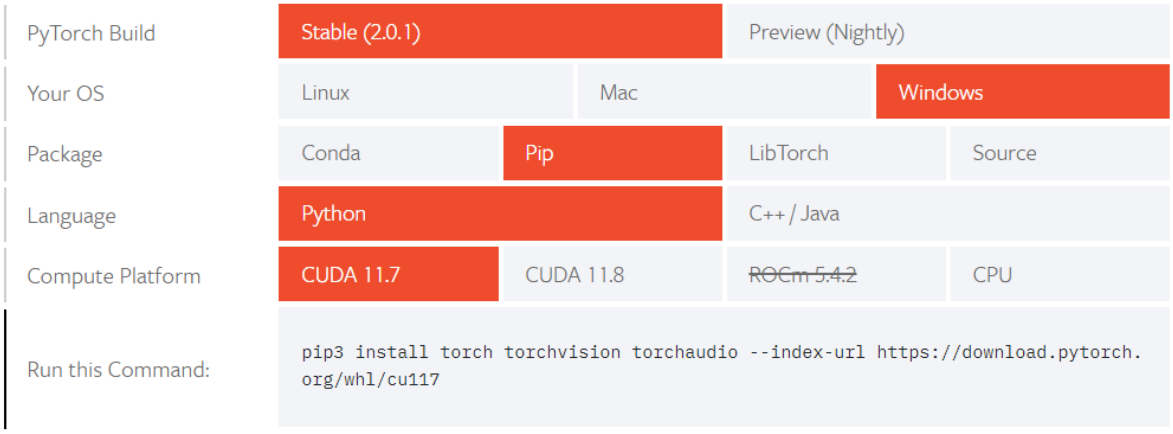
本部分首先介绍 Pytorch 官方提供的示例程序，作为作业程序的基础。

首先需要配置环境，包括安装 python、pytorch、Pycharm。建议使用 Anaconda 管理 python 环境，保证设备上不同 python 项目的 python 环境互不干扰，特别是不会有依赖冲突的问题。环境配置具体包括以下几个步骤：

1. 安装 Anaconda：https://docs.anaconda.com/free/anaconda/install/index.html
2. 创建虚拟 python 环境：https://docs.anaconda.com/free/anaconda/configurations/switch-environment/

   创建环境命令：conda create –name {取个名字} python={版本号}　%这里我们可以使用 python 3.10

   启动刚才的 python 环境：activate {取的名字}
3. 在这个环境中安装 pytorch 相关的库：首先查看页面：https://pytorch.org/get-started/locally/ ，这里会自动检测你系统的信息，示例如下：



最后的“Run this Command”提供的是可以用来安装 Pytorch 的命令，直接拷贝下来，在刚才启动的环境中运行这个命令，就可以开始安装了，如下（这里我的虚拟环境名字为 summer23）：

```
C:\Users\lcynju>conda env list
# conda environments:
#
base                  *  C:\Users\lcynju\Anaconda3
kongli                   C:\Users\lcynju\Anaconda3\envs\kongli
summer23                 C:\Users\lcynju\Anaconda3\envs\summer23
tfpy36                   C:\Users\lcynju\Anaconda3\envs\tfpy36


C:\Users\lcynju>activate summer23

C:\Users\lcynju>conda.bat activate summer23

(summer23) C:\Users\lcynju>pip3 install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu117
```

4. 安装 pycharm，一个对 python 支持非常强大的集成开发环境（IDE）：https://www.jetbrains.com/pycharm/

安装好之后（可能需要注册、激活，好像可以用学校邮箱免费激活）打开，创建 Python 项目，在项目下新建 python 文件（随意命名，例如 SimpleNN.py），将官方提供的 pytorch 示例程序拷贝到该文件中，也就是下面这个文件：

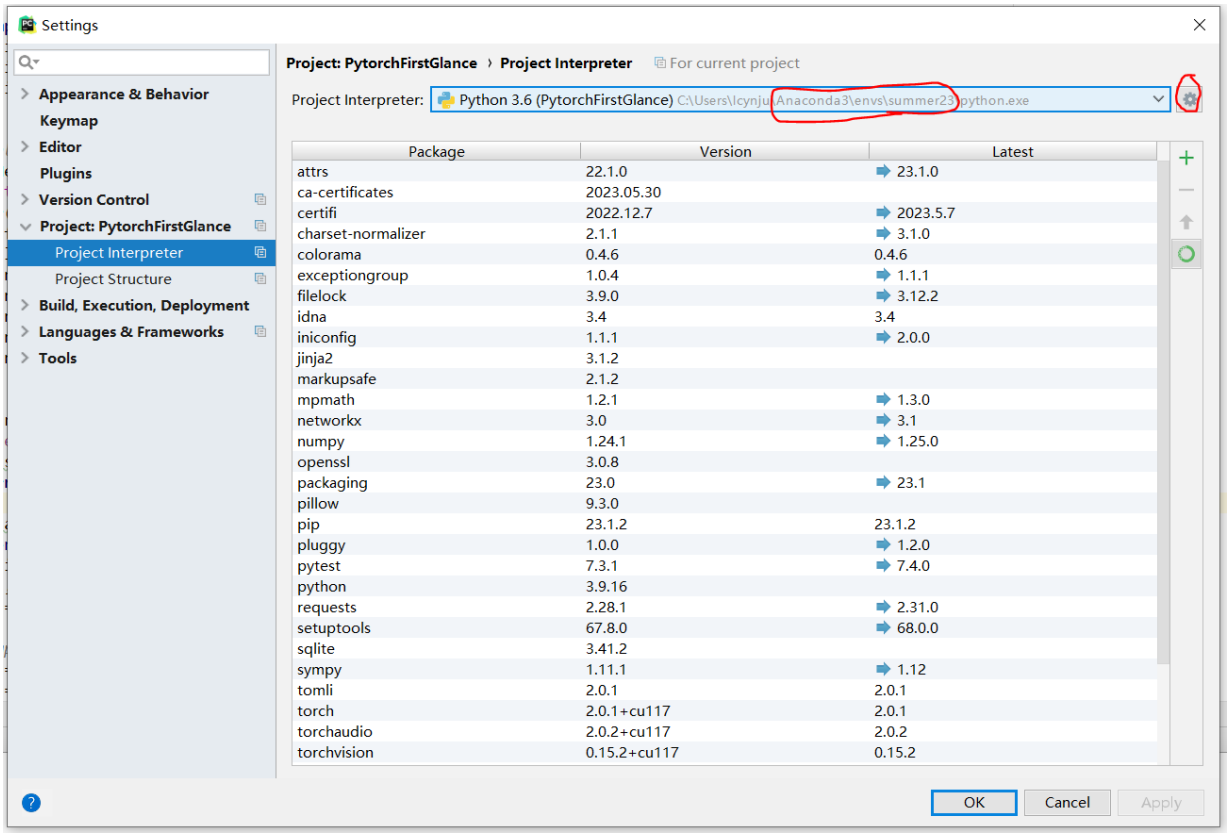
```python
import torch
from torch import nn
from torch.utils.data import DataLoader
from torchvision import datasets
from torchvision.transforms import ToTensor
```

```python
# Define model
#! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! !
class NeuralNetwork(nn.Module):
    def __init__(self):
        super().__init__()
        self.flatten = nn.Flatten()
        self.linear_relu_stack = nn.Sequential(   #非常简单的网络结构
            nn.Linear(28*28, 512),    #第一层是将输入层 28*28 的向量长度（例子中图像的像素点数量）映射到 512 维
            nn.ReLU(),    #这里定义了激活函数是 ReLU
            nn.Linear(512, 512),    #第二层的输入是第一层输出的 512 维向量，再映射到 512 维
            nn.ReLU(),
            nn.Linear(512, 10)   #最后将第二层输出的 512 维向量映射到 10 个神经元，每一个代表一个目标类型
        )

    def forward(self, x):   #不可以自己显式调用，pytorch 内部自带调用机制
        x = self.flatten(x)
        logits = self.linear_relu_stack(x)  #这里是最后一层 10 个神经元的输出；在下面计算 loss 的时候，会用到
        return logits

def train(dataloader, model, loss_fn, optimizer): #模型训练过程的定义；这个可以看作是模板，以后写 pytorch 程序基本都这样
    size = len(dataloader.dataset)
    model.train()
    for batch, (X, y) in enumerate(dataloader):
        X, y = X.to(device), y.to(device)

        # Compute prediction error
        pred = model(X)
        loss = loss_fn(pred, y)

        # Backpropagation
        loss.backward()
        optimizer.step()
        optimizer.zero_grad()

        if batch % 100 == 0:
            loss, current = loss.item(), (batch + 1) * len(X)
            print(f"loss: {loss:>7f}  [{current:>5d}/{size:>5d}]")

def test(dataloader, model, loss_fn): #模型测试过程的定义，这个也是模板，以后可以借鉴
    size = len(dataloader.dataset)
    num_batches = len(dataloader)
    model.eval()
    test_loss, correct = 0, 0
    with torch.no_grad():
        for X, y in dataloader:
            X, y = X.to(device), y.to(device)
            pred = model(X)
            test_loss += loss_fn(pred, y).item()
            correct += (pred.argmax(1) == y).type(torch.float).sum().item()
    test_loss /= num_batches
    correct /= size
    print(f"Test Error: \n Accuracy: {(100*correct):>0.1f}%, Avg loss: {test_loss:>8f} \n")


if __name__ == '__main__':
    # Download training data from open datasets.
    #这是官方提供的示例数据，这个类是会自动到网上下载数据的；如果我们用自己的数据，就是要自定义一个 dataset
    #! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! !
    training_data = datasets.FashionMNIST(
        root="data",
        train=True,
        download=True,
        transform=ToTensor(),
    )

    # Download test data from open datasets.
    test_data = datasets.FashionMNIST(
        root="data",
```

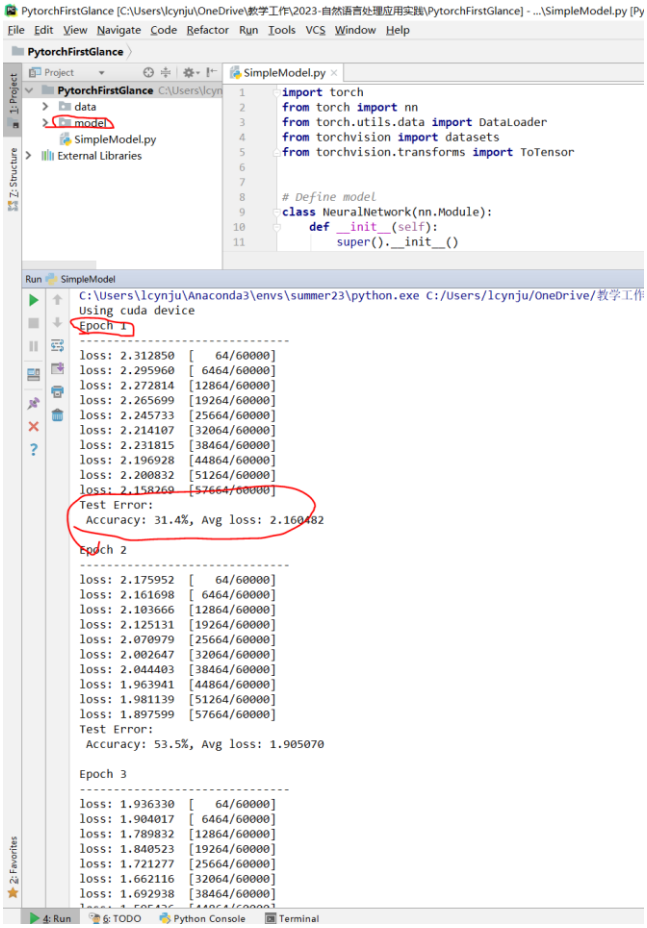
```
    train=False,
    download=True,
    transform=ToTensor(),
)

batch_size = 64 #这里可以自定义

# Create data loaders.
# 这个也是标准用法，只要按照要求自定义数据集，就可以用标准的dataLoader 加载数据
train_dataloader = DataLoader(training_data, batch_size=batch_size)
test_dataloader = DataLoader(test_data, batch_size=batch_size)

device = (
    "cuda"
    if torch.cuda.is_available()
    else "mps"
    if torch.backends.mps.is_available()
    else "cpu"
)
print(f"Using {device} device")

model = NeuralNetwork().to(device)

loss_fn = nn.CrossEntropyLoss()  #课上我们说过，loss 类型是可以选择的
optimizer = torch.optim.SGD(model.parameters(), lr=1e-3)  #这里的优化器也是可以选择的

epochs = 5  #这个训练的轮数也可以设置

#下面这个训练和测试的过程也是标准形式，我们用自己的数据也还是这样去写
for t in range(epochs):
    print(f"Epoch {t+1}\n-------------------------------")
    train(train_dataloader, model, loss_fn, optimizer)
    test(test_dataloader, model, loss_fn)
print("Done!")

torch.save(model.state_dict(), "model/haha")  #模型可以保存下来，这里 model 文件夹要和当前 py 文件在同一个目录下
print("Saved PyTorch Model State to the project root folder!")

classes = [
    "T-shirt/top",
    "Trouser",
    "Pullover",
    "Dress",
    "Coat",
    "Sandal",
    "Shirt",
    "Sneaker",
    "Bag",
    "Ankle boot",
]

model.eval()
x, y = test_data[0][0], test_data[0][1]
with torch.no_grad():
    x = x.to(device)
    pred = model(x)
    predicted, actual = classes[pred[0].argmax(0)], classes[y]
    print(f'Predicted: "{predicted}", Actual: "{actual}"')
```

请仔细阅读代码中添加的注释。这个代码基本可以当作是用 pytorch 编写神经网络模型的模板，我们只需要定义模型、定义数据集即可。

如果想要运行这段代码，还需要为当前的 python 项目配置 python 解释器（interpreter），具体如下图：（点击菜单栏 File—>选择 Settings，弹出对话框，选择左边的 Project：XXXX，点击 Project Interpreter 出现可选的下拉框，可能需要我们点击小齿轮，Add Local…，将通过 Anaconda 创建的 python 环境加到这里，然后在这个界面选择）



然后就可以运行这个 py 文件了，结果如下（如果报错看看是不是模型保存目录"model"文件夹没创建）：



到此为止，我们已经能够在自己配置的环境下运行官方的示例代码了。接下来就是修改官方代码，完成我们的作业。

二、作业

1. 作业描述：根据给定的 IMDB 数据集（csv 文件，未划分训练集、验证集和测试集），训练一个基于 LSTM 的神经网络分类模型，为数据集中的每一个电影评论确定其情感极性。数据集中每一条数据是一个电影评论，以及这个评论的情感极性，即积极（positive）和消极（negative），如下图：

review	sentiment
One of the other reviewers has mentioned that after watching just 1 Oz episode you'll be hooked. They are right, as this is exactly what happened with me. The first thing that struck me about Oz was its brutality and unflinching scenes of violence, which set in right from the word GO. Trust me, this is not a show for the faint hearted or timid. This show pulls no punches with regards to drugs, sex or violence. Its is hardcore, in the classic use of the word. It is called OZ as that is the nickname given to the Oswald Maximum Security State Penitentary. It focuses mainly on Emerald City, an experimental section of the prison where all the cells have glass fronts and face inwards, so privacy is not high on the agenda. Em City is home to many..Aryans, Muslims, gangstas, Latinos, Christians, Italians, Irish and more....so scuffles, death stares, dodgy dealings and shady agreements are never far away. I would say the main appeal of the show is due to the fact that it goes where other shows wouldn't dare. Forget pretty pictures painted for mainstream audiences, forget charm, forget romance...OZ doesn't mess around. The first episode I ever saw struck me as so nasty it was surreal, I couldn't say I was ready for it, but as I watched more, I developed a taste for Oz, and got accustomed to the high levels of graphic violence. Not just violence, but injustice (crooked guards who'll be sold out for a nickel, inmates who'll kill on order and get away with it, well mannered, middle class inmates being turned into prison bitches due to their lack of street skills or prison experience) Watching Oz, you may become comfortable with what is uncomfortable viewing....thats if you can get in touch with your darker side.	positive
A wonderful little production. The filming technique is very unassuming- very old-time-BBC fashion and gives a comforting, and sometimes discomforting, sense of realism to the entire piece. The actors are extremely well chosen- Michael Sheen not only "has got all the polari" but he has all the voices down pat too! You can truly see the seamless editing guided by the references to Williams' diary entries, not only is it well worth the watching but it is a terrificly written and performed piece. A masterful production about one of the great master's of comedy and his life. The realism really comes home with the little things: the fantasy of the guard which, rather than use the traditional 'dream' techniques remains solid then disappears. It plays on our knowledge and our senses, particularly with the scenes concerning Orton and Halliwell and the sets (particularly of their flat with Halliwell's murals decorating every surface) are terribly well done.	positive
I thought this was a wonderful way to spend time on a too hot summer weekend, sitting in the air conditioned theater and watching a light-hearted comedy. The plot is simplistic, but the dialogue is witty and the characters are likable (even the well bread suspected serial killer). While some may be disappointed when they realize this is not Match Point 2: Risk Addiction, I thought it was proof that Woody Allen is still fully in control of the style many of us have grown to love. This was the most I'd laughed at one of Woody's comedies in years (dare I say a decade?). While I've never been impressed with Scarlet Johanson, in this she managed to tone down her "sexy" image and jumped right into a average, but spirited young woman. This may not be the crown jewel of his career, but it was wittier than "Devil Wears Prada" and more interesting than "Superman" a great comedy to go see with friends.	positive

2. 过程指导：

（1）自行划分训练集（70%）、验证集（10%）、测试集（20%）。分别统计训练集、验证集、测试集中 positive 和 negative 数据的规模，并统计整个数据集中评论的平均长度（词语个数）、最大和最小长度，在实验报告中汇报。注意：训练集用来训练模型；验证集用来为模型选择超参数，例如 batch size、epoch、loss function、activation function、input length、optimizer 等（分别设置不同的选择，训练后在验证集上看效果，如果验证集达到最好的效果，则选定这些超参数）；测试集用来评估使用选定的超参数训练的模型的效果。

（2）评价指标就使用示例代码中的 accuracy 即可。

（3）【可选】对所有文本过滤停用词。自行选择停用词表（Stop words），将评论中的这些词删掉，形成新的数据集，基于新的数据集训练和评估模型。

（3）使用 word2vec 为数据集构建词语的向量表示，将每一个评论转换为词向量的数组，和极性标签。Word2vec 的用法参考：https://radimrehurek.com/gensim/models/word2vec.html

其关键是使用所有的文本构建 Word2vec 模型，然后再用模型将每一个评论变成词语的向量表示：

```
##首先需要在我们的虚拟 python 环境中安装 gensim 库，先 activate 我们创建的环境，
##再使用 pip2 install gensim 安装 gensim
from gensim.test.utils import common_texts
```

```
from gensim.models import Word2Vec

##下面这句话是构建 Word2Vec 模型，自行查看官方文档，了解每一个参数的含义
model = Word2Vec(sentences=common_texts, vector_size=100, window=5, min_count=1, workers=4)

##然后保存模型
model.save("word2vec.model")

##可以用下面的方式将词语'computer'变成向量，可以 print(vector)试试看是什么
vector = model.wv['computer']
```

可以将给大家的 csv 文件直接拷贝到 txt 文件中，当作文本文件处理，可以可以使用 pandas 库处理 csv 文件（同样在我们的 python 环境中用 pip3 install pandas 命令安装）。pandas 用法参考：https://pandas.pydata.org/docs/getting_started/index.html

转换成向量的数据文件中，每一个评论应该是一个向量数组和一个标签（positive 或 negative）。接下来就是使用这个文件构建我们的数据集。

（4）第一部分实践中，我们给出的示例代码用的是官方的 FashionMNIST()数据集，这里我们需要为我们的评论数据自定义一个数据集，包括训练、验证和测试三个子集，可以分别定义数据集对象。在 pytorch 中自定义数据集的方法可参考：https://pytorch.org/tutorials/beginner/basics/data_tutorial.html#creating-a-custom-dataset-for-your-files （请仔细阅读这个页面的示例文件，最好自己能够理解）

这里我们简单解释一下这个例子：

```python
import os
import pandas as pd
from torchvision.io import read_image

#注意这里是继承了一个 Dataset 基础类，很多功能是这个基础类就有的，
#我们只需要按照这个示例实现__init__和__getitem__就行了
class CustomImageDataset(Dataset):
    def __init__(self, annotations_file, img_dir, transform=None, target_transform=None):
        #这里使用 pandas 读取 csv 文件，参数应该是一个文件路径，这个文件里面放的是每一张图片的类型，图片是另外存放的
        #我们的数据不同，我们是评论的向量和标签放在一起的
        self.img_labels = pd.read_csv(annotations_file)
        self.img_dir = img_dir#这里定义了图片的根目录
        self.transform = transform
        self.target_transform = target_transform

    def __len__(self):
        return len(self.img_labels)

    #这个方法是关键，需要我们根据 idx 返回每一条对应序号的数据，包括 x 和 y，这个例子中 x 是 image,y 是 label
    #我们的评论数据中，x 是转换之后的 word vector 数组，y 是极性标签
    #我们的数据集比这个示例要简单，可以直接在__init__中将所有数据以数组读出来，然后在__getitem__中用 idx 获取就行了
    def __getitem__(self, idx):
        img_path = os.path.join(self.img_dir, self.img_labels.iloc[idx, 0])
        image = read_image(img_path)
        label = self.img_labels.iloc[idx, 1]
        if self.transform:
            image = self.transform(image)
        if self.target_transform:
            label = self.target_transform(label)
        return image, label
```

根据上述的例子，我们可以实现一个训练数据集（请仔细阅读 pandas 的 read_csv 方法的说明）：

```
import os
import pandas as pd
from torchvision.io import read_image

class IMDBTrainingDataset(Dataset):
    def __init__(self, train_file, transform=None, target_transform=None):
        self.vectors_labels = pd.read_csv(train_file) #第一列是 vector，第二列是 label

    def __len__(self):
        return len(self.img_labels)

    def __getitem__(self, idx):
        vector = self.vectors_labels.iloc[idx, 0]
        label = self.vectors_labels.iloc[idx, 1]
        return vector, label
```

有了数据集，就可以用 DataLoader 将数据导入了，然后将 dataloader 传递给训练模型的 train 函数了。

（5）定义 lstm 模型，上述示例程序用的是全连接神经网络，本次作业要求大家使用 LSTM 网络。这个可以直接使用 Pytorch 中已经封装好的 LSTM：https://pytorch.org/docs/stable/generated/torch.nn.LSTM.html 。例如，我们作业中可以这样定义网络结构（不完整，需要添加二分类的层，参考第一部分的教程）：

```
class MyLSTM(nn.Module):
    def __init__(self, input_size, hidden_dim):
        super(MyLSTM,self).__init__()
        self.input_dim = input_size
        self.hidden_dim = hidden_dim
        #此处 input_size 是我们 word2vec 的词向量的维度；
        #这里设置了输入的第一个维度为 batchsize，那么在后面构造输入的时候，需要保证第一个维度是 batch size 数量
        self.lstm = nn.LSTM(input_size, hidden_dim, batch_first=True)

    def init_hidden(self, batch_size):#初始化两个隐藏向量 h0 和 c0
        return (Variable(torch.zeros(1, batch_size, self.hidden_dim)),
                Variable(torch.zeros(1, batch_size, self.hidden_dim)))

    def forward(self, input):#不可以自己显式调用，pytorch 内部自带调用机制
        #input 是传递给 lstm 的输入，它的 shape 应该是（每一个文本的词语数量，batch size，词向量维度）
        #输入的时候需要将 input 构造成
        self.hidden = self.init_hidden(input.size(0))   #input.size(0)得到 batch_size
        lstm_out, self.hidden = self.lstm(seq, self.hidden)
        return lstm_out  #查看文档，了解 lstm_out 到底是什么
```

注意：使用 dataloader 导入数据的时候，需要配上 batch_size，但是这样构造的 input 的 shape 需要我们手动查看一下，看看是不是 (batch_size, sequence_length, input_size)，如果不是的话，需要调整 input 的 shape 或者在模型定义中调整 batch_first=True 的设置，并修改 forward 函数。

3. 作业提交内容

（1）完整的实验代码，包括 word2vec 数据构造、数据集读取、训练模型、验证模型和测试模型的代码。

（2）实验报告：

（2.1）数据集划分与统计：划分方式描述，以及上述数据集的特征统计。

（2.2）实验结果：使用了哪些不同的超参数设置，哪一种超参数设置在验证集上效果最好，其最终训练 loss 和测试 loss 分别是多少，测试集上的分类 accuracy 是多少。