高级量化交易技术

闫涛
科技有限公司
北京 2021.05.08
{yt7589}@qq.com

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

第零篇深-度学习

第 3 章模型训练

Abstract

在本章中我们将详细讲解用于金融交易的 Transformer 网络的模型训练和预测过程。

1 模型训练与预测概述概述

1.1 训练过程

下面我们来看模型的训练过程，训练入口程序如下所示：

```
    def train(self):
        cmd_args = self.parse_args()
        stock_symbol = 'sh600260'
        batch_size = cmd_args.batch_size
        NUM_CLS = 3
        cmd_args.embedding_size = 5
        seq_length = 11
        cmd_args.num_heads = 4
        cmd_args.depth = 6 # 原始值为2
        train_iter, test_iter = self.load_stock_dataset(stock_symbol, batch_size)
        cmd_args.num_heads = 8
        model = FmtsTransformer(emb=cmd_args.embedding_size, heads=cmd_args.num_heads, depth=cmd_args.depth, \
                    seq_length=seq_length, num_tokens=cmd_args.vocab_size, num_classes=NUM_CLS, \
                    max_pool=cmd_args.max_pool)
        model.to(self.device)
        opt = torch.optim.Adam(lr=cmd_args.lr, params=model.parameters())
        sch = torch.optim.lr_scheduler.LambdaLR(opt, lambda i: min(i / (cmd_args.lr_warmup / cmd_args.batch_size), 1.0))
        if cmd_args.continue_train:
            e, model_dict, optimizer_dict = self.load_ckpt(self.ckpt_file)
            model.load_state_dict(model_dict)
            opt.load_state_dict(optimizer_dict)
        # training loop
        cmd_args.num_epochs = 3
        seen = 0
        # early stopping参数
        best_acc = -1
        acc_up = 0.0
        min_acc_up = 0.000001 # 识别为精度提高的最小阈值
        non_acc_up_epochs = 0 # 目前多少个epoch精度未提高
```

```
30          max_no_acc_up_epochs = 50 # 如果精度在这些epoch后还没提高
    则终止训练过程
31          for epoch in range(cmd_args.num_epochs):
32              print(f'\n epoch {epoch}')
33              model.train(True)
34              for batch in tqdm.tqdm(train_iter):
35                  opt.zero_grad()
36                  X, y = self.get_stock_batch_sample(batch,
    batch_size, cmd_args.embedding_size)
37                  y_hat = model(X)
38                  loss = F.nll_loss(y_hat, y)
39                  loss.backward()
40                  # clip gradients
41                  # - If the total gradient vector has a length > 1,
     we clip it back down to 1.
42                  if cmd_args.gradient_clipping > 0.0:
43                      nn.utils.clip_grad_norm_(model.parameters(),
    cmd_args.gradient_clipping)
44                  opt.step()
45                  sch.step()
46                  seen += X.size(0)
47              with torch.no_grad():
48                  model.train(False)
49                  tot, cor= 0.0, 0.0
50                  for batch in tqdm.tqdm(test_iter):
51                      X, y = self.get_stock_batch_sample(batch,
    batch_size, cmd_args.embedding_size)
52                      y_hat = model(X).argmax(dim=1)
53                      tot += float(X.size(0))
54                      cor += float((y == y_hat).sum().item())
55                  acc = cor / tot
56                  # 获取当前最佳测试集精度, 并保存对应的模型
57                  if best_acc < acc:
58                      acc_up = acc - best_acc
59                      if acc_up > min_acc_up:
60                          best_acc = acc
61                          non_acc_up_epochs = 0
62                          print('保存模型参数')
63                          self.save_ckpt(self.ckpt_file, epoch,
    model, opt)
64                  else:
65                      non_acc_up_epochs += 1
66                      if non_acc_up_epochs > max_no_acc_up_epochs:
67                          print('模型已经处于饱合状态, 停止训练过程'
    )
```

```
68                         break
69                 print(f'-- {"test" if cmd_args.final else "
   validation"} accuracy {acc:.3}')
```

Listing 1: 模型训练入口

代码解读如下所示：

- 第 5 行：Transformer 网络输出三种市场状态：上涨、下跌、震荡；
- 第 6 行：我们将每个时刻等价为一个单词，每个时刻可以用开盘、最高、最低、收盘、交易量来表示，因此相当于每个单词是 5 维向量；
- 第 7 行：我们通常考虑当前时刻，同时向前看 10 个时刻，因此每个样本包括 11 个时刻，相当于每个句子有 11 个单词，所以序列长度为 11；
- 第 8 行：共有 6 层结构；
- 第 10 行：共有 8 个自注意头；
- 第 12~15 行：初始化模型，其中 vocab_size=50000 为缺省值，max_pool 代表使用最大池化，并将其放入 GPU 中；
- 第 16 行：使用 adam 优化算法；
- 第 17 行：使用带有 warmup 的学习调整计划算法：...；
- 第 18~21 行：在之前训练的基础上继续训练；
- 第 22 行：训练集训练遍数；
- 第 23 行：共训练了多少个样本；
- 第 26 行：当前所取得的最佳精度；
- 第 27 行：当前精度提高的数值；
- 第 28 行：视为精度有提高的最小阈值，用于避免拢动情况；
- 第 29 行：目前已经有多少个 epoch 精度没有提高，如果达到最大允许的精度没有提高的 epoch 数，则停止训练过程；
- 第 30 行：允许连续多少个 epoch 精度没有显著提高；
- 第 31 行：训练指定个 epoch；
- 第 33 行：将模型置为训练状态；
- 第 34 行：读出训练集的每个迷你批次，循环进行处理；
- 第 35 行：所有参数的梯度设置为 0；
- 第 36 行：获取样本集和标签集，具体实现细节将在后面详细讲解；
- 第 37 行：调用 FmtsTransformer 模型计算网络输出结果；
- 第 38 行：使用负对数似然函数为代价函数，计算代价函数的值；
- 第 39 行：将代价函数值反向传播，求出梯度值；
- 第 42、43 行：当梯度所组成的向量长度大于 1 时，调整梯度值，使长度变为 1；
- 第 44 行：调用优化器对参数进行优化；
- 第 45 行：调用学习率变化方法，按需调整学习率；
- 第 46 行：统计当前学习了多少个样本；

- 第 47、48 行：将其变为验证过程；
- 第 49 行：定义 tot 代表总样本数，cor 为结果正确的样本数；
- 第 50 行：按批次循环处理测试数据集；
- 第 51 行：取出一个批次的样本集和标签集；
- 第 52 行：调用 FmtsTransformer 模型的前向传播过程，求出网络层输出 logits，然后在每个样本结果向量中取出最大值的索引值（取值范围为 0~2），作为预测值；
- 第 53 行：将批次数加入到样本总数中；
- 第 54 行：y=y_hat 的运算结果为一个新的同维度 ndarray，对于第 i 个元素，如果 $y_i == \hat{y}_i$，则该元素为 1，否则为 0。对新数据取 sum 就可以计算出在本批次中正确的样本数量；
- 第 55 行：当处理完所有测试集后，求出在测试集上的精度；
- 第 57~63 行：如果当前精度高于最佳精度，求出精度的提升量，如果提升量高于认为有提升的阈值，则更新最佳精度至当前精度，将连续没有精度提高的 epoch 数置为 0，保存模型至文件；
- 第 64~68 行：如果没有精度提升，则连续没有精度提升的 epoch 数加 1，如果该值大于允许连续没有精度提升 epoch 的最大值，则退出训练过程；

1.2 预测过程

下面来看预测过程，代码如下所示：

```
1  def predict(self):
2      stock_symbol = 'sh600260'
3      cmd_args = self.parse_args()
4      batch_size = cmd_args.batch_size
5      NUM_CLS = 3
6      cmd_args.embedding_size = 5
7      seq_length = 11
8      cmd_args.depth = 6 # 原始值为2
9      cmd_args.num_heads = 8
10     model = FmtsTransformer(emb=cmd_args.embedding_size, heads=
       cmd_args.num_heads, depth=cmd_args.depth, \
11                     seq_length=seq_length, num_tokens=cmd_args.
       vocab_size, num_classes=NUM_CLS, \
12                     max_pool=cmd_args.max_pool)
13     model.to(self.device)
14     e, model_dict, optimizer_dict = self.load_ckpt(self.ckpt_file)
15     model.load_state_dict(model_dict)
16     # 获取测试样本
17     train_iter, test_iter = self.load_stock_dataset(stock_symbol,
       batch_size)
18     batch = iter(test_iter).next()
19     batch_X, batch_y = self.get_stock_batch_sample(batch,
       batch_size, cmd_args.embedding_size)
20     X = batch_X[:1, :, :]
```

```
21      y = batch_y[:1]
22      print('##### X: {0}; y: {1};'.format(X.shape, y.shape))
23      y_hat = model(X).argmax(dim=1)
24      print('y_hat: {0}; y: {1};'.format(y_hat.item(), y.item()))
```

Listing 2: 模型预测入口

代码解读如下所示：

- 第 4 行：批次大小 batch_size 的缺省值为 4；
- 第 5 行：输出层大小，代表市场状态：上涨、下跌、震荡；
- 第 6 行：每个时刻行情数据：开盘、最高、最低、收盘、交易量，可以视为 NLP 中的单词，设定其维度为 5 维；
- 第 7 行：包括当前时刻，再向前取 10 个时刻，共 11 个时刻，相当于 NLP 中规定每个句子的长度为 11；
- 第 8 行：共有 6 层 Encoder 层；
- 第 9 行：共有 8 个自注意力头；
- 第 10~ 12 行：初始化模型，其中 vocab_size=50000 为缺省值，max_pool 代表使用最大池化；
- 第 13 行：将其放入 GPU 中；
- 第 14、15 行：载入预训练好的模型；
- 第 17 行：载入数据集；
- 第 18 行：取出第一个批次；
- 第 19 行：取出这个批次中的样本集和标签集；
- 第 20 行：取出样本集中第 1 个样本；
- 第 21 行：取出标签集中对应的标签；
- 第 23 行：调用 FmtsTransformer 模型求出输出信号 $\hat{y} \in R^3$，通过 argmax 求出最大值元素的索引号作为输出值；

2 总结

3 附录 X