# 高级量化交易技术

**闫涛** 科技有限公司 北京 2021.05.08 {yt7589}@qq.com 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()
2
          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
11
          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)
14
          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
17
     (i / (cmd_args.lr_warmup / cmd_args.batch_size), 1.0))
          if cmd_args.continue_train:
18
              e, model_dict, optimizer_dict = self.load_ckpt(self.
19
     ckpt_file)
              model.load_state_dict(model_dict)
20
              opt.load_state_dict(optimizer_dict)
          # training loop
22
          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
28
          non_acc_up_epochs = 0 # 目前多少个epoch精度未提高
```

```
max no acc up epochs = 50 # 如果精度在这些epoch后还没提高
30
      则终止训练过程
           for epoch in range (cmd_args.num_epochs):
               print(f'\n epoch {epoch}')
32
               model.train(True)
33
               for batch in tgdm.tgdm(train iter):
34
                   opt.zero grad()
35
                   X, y = self.get stock batch sample(batch,
36
      batch size, cmd args.embedding size)
                   y hat = model(X)
37
                   loss = F. nll loss (y hat, y)
                   loss.backward()
39
                   # clip gradients
40
                   # - If the total gradient vector has a length > 1,
41
      we clip it back down to 1.
                   if cmd_args.gradient_clipping > 0.0:
42
                       nn.utils.clip_grad_norm_(model.parameters(),
43
      cmd args.gradient clipping)
                   opt.step()
                   sch.step()
45
                   seen += X. \operatorname{size}(0)
46
               with torch.no grad():
47
                   model.train(False)
48
                   tot, cor=0.0, 0.0
49
                   for batch in tqdm.tqdm(test iter):
50
                       X, y = self.get_stock_batch_sample(batch,
51
      batch_size , cmd_args.embedding_size)
                       y \text{ hat} = \text{model}(X) . \operatorname{argmax}(\dim = 1)
                       tot += float(X. size(0))
                       cor += float ((y == y_hat).sum().item())
                   acc = cor / tot
                   # 获取当前最佳测试集精度,并保存对应的模型
56
                   if best acc < acc:
57
                       acc up = acc - best acc
58
                       if acc_up > min_acc_up:
59
                            best acc = acc
60
                           non_acc_up_epochs = 0
61
                            print ('保存模型参数')
62
                            self.save_ckpt(self.ckpt_file, epoch,
63
     model, opt)
                   else:
64
                       non_acc_up_epochs += 1
65
                       if non_acc_up_epochs > max_no_acc_up_epochs:
66
                            print ('模型已经处于饱合状态,停止训练过程:
67
```

```
break

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;

2 总结

- 第行:;
- 1.2 预测过程
- 2 总结

3 附录 X 7

3 附录 X