时间序列分析
作业二

502022370071, 庄镇华, zhuangzh@lamda.nju.edu.cn

2022 年 11 月 10 日

作业提交注意事项

(1) 请严格参照教学立方网站所述提交作业，压缩包命名统一为学号 _ 姓名.zip;

(2) 未按照要求提交作业，或提交作业格式不正确，将会被扣除部分作业分数;

(3) 除非有特殊情况（如因病缓交），否则截止时间后不接收作业，本次作业记零分。

1 [100pts] 机器学习场景下的时序预测模型

在使用机器学习模型解决时序预测问题时，单条时间序列往往不能很好地衡量模型的能力，因此常见的做法是从一个更长的时间序列中构造若干个相似的子序列。本题中使用上次作业中的 ETTh1 的 OT 组分作为数据，在该场景下实现线性回归和指数平滑模型。数据集、代码分别保存在 data、code 文件夹下，请阅读代码并完成如下任务:

(1) 在 transforms.py 中实现归一化（Normalization），标准化（Standardization），平均归一化（Mean Normalization），Box-Cox 变换，需继承 Transform 类并实现其抽象方法;

(2) 在 metrics.py 中实现 MSE，MAE，MAPE，sMAPE，MASE。

(3) 在 models.py 中**设计并实现**线性回归模型以及指数平滑模型，要求模型以长度为 L 的历史序列为输入，并做出长度为 H 的预测。模型需继承 MLForecastModel 类并实现其抽象方法;（线性回归模型可参考第三章课件 39 页，指数平滑模型可参考第三章课件 19 页）

(4) 修改并运行 main.py，汇报 (3) 中不同方法，在 (1) 中不同变换下，用 (2) 中不同指标衡量的性能，以表格形式呈现。

注：相较上次作业，本次作业的 API 有所变化。最终需提交的文件为: 1. 修改后的代码，要求附加一个 markdown 格式的文件 README.md，说明如何复现报告中的结果。2. pdf 形式的报告，报告需描述模型的具体实现（例如以数学公式及算法的形式）并报告结果，写于 **solution** 部分即可。

Solution. 此处用于报告 *(*中英文均可*)*

(1) 数据变换即对数据进行规范化处理, 以便于后续的信息挖掘。常见的数据变换包括: 归一化 (*normalization*) , 标准化 (*Standardization*), 平均归一化 (*Mean Normalization*), *Box-Cox* 变换等。下面以数学公式的形式描述各种变换的实现方法。

归一化 (*normalization*): 将时序数据取值限制在 $[0, 1]$

$$y'_t = \frac{y_t - y_{min}}{y_{max} - y_{min}}$$

标准化 (*Standardization*) / *Z-Score*: 将时序数据变换为 *0* 均值以及标准方差

$$y'_t = \frac{y_t - \mu}{\sigma}$$

平均归一化 (*Mean Normalization*) *:*

$$y'_t = \frac{y_t - \mu}{y_{max} - y_{min}}$$

Box-Cox 变换用于分布"正态"程度矫正, 由于实验数据集含有负数, 因此使用二参数 *Box-Cox* 变换, 由于最小值大于*-5*, 因此选取 $\lambda_2 = 5$

$$y_t^{(\lambda)} = \begin{cases} \frac{((y_t+\lambda_2)^{\lambda_1}-1)}{\lambda_1} & \lambda_1 \neq 0 \\ \log(y_t + \lambda_2) & \lambda_1 = 0 \end{cases}$$

(2) 下面以数学公式的形式描述各种指标的计算方法。

MSE (Mean Square Error)

$$MSE = \frac{1}{H}\sum_{i=1}^{H}(y_{n+i} - y'_{n+i})^2$$

MAE (Mean Absolute Error)

$$MAE = \frac{1}{H}\sum_{i=1}^{H}|y_{n+i} - y'_{n+i}|$$

MAPE (Mean Absolute Percentage Error)

$$MAPE = \frac{100}{H}\sum_{i=1}^{H}\frac{|y_{n+i} - y'_{n+i}|}{|y_{n+i}|}$$

sMAPE (symmetric MAPE)

$$sMAPE = \frac{200}{H}\sum_{i=1}^{H}\frac{|y_{n+i} - y'_{n+i}|}{|y_{n+i}| + |y'_{n+i}|}$$

MASE (Mean Absolute Scaled Error)

$$MASE = \frac{1}{H}\sum_{i=1}^{H}\frac{|y_{n+i} - y'_{n+i}|}{\frac{1}{n+H-m}\sum_{j=m+1}^{n+H}|y_j - y_{j-m}|}$$

(3) a. 下面以公式的形式描述线性回归模型的实现：

基于长为 L 的历史数据，预测长为 T 的未来数据。假设历史数据为 $Y_{old} \in \mathbb{R}^{d\times L}$，增广后的历史数据为 $Y'_{old} = [\mathbf{1}_{d\times 1}|Y_{old}] \in \mathbb{R}^{d\times(L+1)}$，目标数据为 $Y_{new} \in \mathbb{R}^{d\times T}$，模型为 $W \in \mathbb{R}^{(L+1)\times T}$，则最终模型表示为（假设 $\boldsymbol{\varepsilon}$ 符合均值为 $\mathbf{0}$ 的多元高斯分布）：

$$Y_{new} = Y'_{old}W + \boldsymbol{\varepsilon}$$

最小化 $\varepsilon^T\varepsilon$，在 Y'_{old} 满秩的情况下求得闭式解：

$$W = (Y'^T_{old}Y'_{old})^{-1}Y'^T_{old}Y_{new}$$

b. 下面以公式和算法的形式描述指数平滑模型的实现：

指数平滑法思想（序列 $\hat{Y_n}$ 为指数平滑序列）

$$\hat{Y}_{L+t|L} = c\sum_{i=0}^{L-1}\lambda^i Y_{L-i} = c[Y_L, \lambda Y_{L-1} + ... + \lambda^{L-1}Y_1]$$

当 $c = \frac{1-\lambda}{1-\lambda^n}$ 时，$\hat{Y}$ 是无偏估计，并且当 $n \to \infty$，$c = 1-\lambda$，简便起见，取 $c = 1-\lambda$。可以得到如下递推公式：

$$\hat{Y}_{L+1} = (1-\lambda)\sum_{i=0}^{L-1}\lambda^i Y_{L-i} = (1-\lambda)Y_L + \lambda(1-\lambda)\sum_{i=0}^{L-2}\lambda^i Y_{L-1-i} = (1-\lambda)Y_L + \lambda\hat{Y}_L$$

令 $\alpha = 1-\lambda$，则

$$\hat{Y}_{L+1} = \alpha Y_L + (1-\alpha)\hat{Y}_L$$

平滑常数 α 决定最近的观测值如何影响预测，对于变化缓慢的序列，常取较小的 α 值；相反，对于变化迅速的序列，常取较大的 α 值。

确定平滑系数 α 主要有三种方法：第一种，经验判断法，画一下变量变化图，显示变化的快慢程度；第二种，试算评估法，对比评估不同 α 值的预测结果，选取预测结果最优的 α 值为最终值，一般用平均绝对百分比误差（$MAPE$）作为评定预测结果的指标；第三种，理论计算法，$\alpha = \frac{2}{(n+1)}$。理论计算法采用整个序列的 n 个时间序列来确定最佳的平滑系数 α 值。常用的确定 n 的方法，是将 n 按照移动平均的移动区间来对待，按移动平均区间的选择来确定 n。

本文我们选择第二种方式得出最终的 α 值，详细结果可见第 *(4)* 小问中的结果表格，可以看到当 $\alpha = 0.8$ 时，模型的平均绝对百分比误差（$MAPE$）最小，因此选择 $\alpha = 0.8$。

(4) 不同方法在不同变换下的性能指标如表*1*所示，针对二参数 *Box-Cox* 变换方式，我们选取 $\lambda_1 = 1$ 观察实验结果，$\lambda_2 = 5$ 为定值用于调整负值；针对指数平滑方法，我们选取 $\alpha = 0.2, 0.5, 0.8$ 观察结果，可以发现当 α 较小时，模型受历史数据影响较大，当 α 较大时，模型能够抓住短期变化趋势；复现此结果的方式请参照 *readme.md* 文件。

表 1: 不同方法在不同变换下的性能指标

Model	Transform	MAE	MSE	MAPE	sMAPE	MASE
AR	None	1.358007789	3.409546699	35.3837516	29.81411585	1.121469914
	Normalize	1.36803031	3.318040857	33.7773423	30.89265524	1.143146257
	Standardize	1.360842136	3.31792317	34.07053567	30.57256756	1.131426688
	MeanNormalize	1.360339888	3.314491135	34.06827655	30.53047105	1.131456024
	Box-Cox(1)	1.358007789	3.409546699	35.3837516	29.81411585	1.121469914
EMA (α=0.2)	None	1.492901456	3.947040015	36.16943852	32.79607163	1.254508032
	Normalize	1.492901457	3.947040015	36.16943852	32.79607164	1.254508033
	Standardize	1.492901456	3.947040016	36.16943853	32.79607161	1.254508032
	MeanNormalize	1.492901456	3.947040016	36.16943853	32.79607161	1.254508032
	Box-Cox(1)	1.492901457	3.947040015	36.16943852	32.79607164	1.254508033
EMA (α=0.5)	None	1.49245277	3.971514965	35.70747819	32.99877314	1.250634992
	Normalize	1.49245277	3.971514965	35.70747819	32.99877314	1.250634992
	Standardize	1.49245277	3.971514965	35.70747819	32.99877314	1.250634992
	MeanNormalize	1.49245277	3.971514965	35.70747819	32.99877314	1.250634992
	Box-Cox(1)	1.49245277	3.971514965	35.70747819	32.99877314	1.250634992
EMA (α=0.8)	None	1.494404806	4.005677583	35.67737469	33.20245878	1.249740162
	Normalize	1.494404806	4.005677583	35.67737469	33.20245878	1.249740162
	Standardize	1.494404806	4.005677583	35.67737469	33.20245878	1.249740162
	MeanNormalize	1.494404806	4.005677583	35.67737469	33.20245878	1.249740162
	Box-Cox(1)	1.494404806	4.005677583	35.67737469	33.20245878	1.249740162

2 [附加题 20pts] EMA 的应用

指数滑动平均 (EMA) 广泛应用于半监督，自监督等机器学习领域，常常被用作指标或模型的平滑。本题中学生需要实现一个在CIFAR10 数据集上的分类模型 (梯度下降方法优化)，并完成以下任务：

(1) 记录每次模型更新时的训练损失，绘制出使用 EMA 平滑前和平滑后**损失随时间变化曲线**。

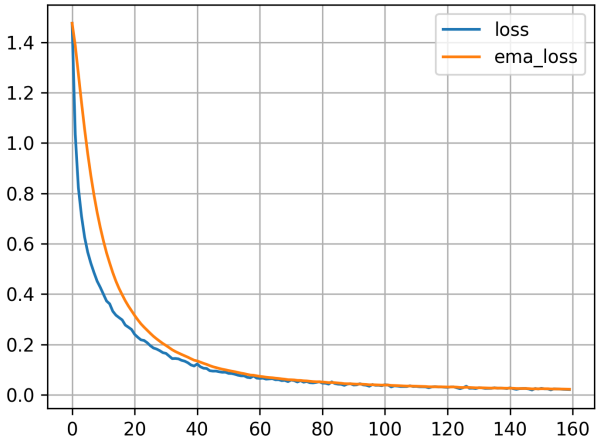
(2) 使用 EMA 对每次模型更新后的参数做平滑，绘制出使用 EMA 平滑前和平滑后的模型**测试准确率随时间变化曲线**。(注意 EMA 只对模型参数做平滑，并不参与模型训练，可参考第三章课件 60 页)

注：本题未提供模板代码，学生需自己实现，模型和训练方式任选，推荐使用 pytorch。最终需提交的文件为: 1. 名为 extra 的代码文件夹，无需包含数据集，要求附加一个 markdown 格式的文件 README.md，说明如何复现报告中的结果。2. pdf 形式的报告，按要求报告结果，写于 **solution** 部分即可。

Solution. *此处用于报告 (中英文均可)*

(1) 本题的代码实现可见 extra 文件夹，使用 pytorch 训练框架，采用 res_net-18 模型，模型训练 1000 轮，大约在 160 轮左右收敛，因此绘图仅绘制前 160 轮的数据，最终可以在测试集上达到 90%+ 的准确率。

图 1: 使用 EMA 平滑 loss 前后训练损失值随时间变化曲线



最终结果如图1所示，可以发现经过指数平滑后的损失值变化较为平缓（本次实验取平滑系数 $\alpha = 0.2$），实现代码如下图所示。

```python
loss_list = np.load('./loss_list_without.npy')[ : 160]
alpha = 0.2
ema_loss_list = loss_list.copy()
for i in range(1, loss_list.shape[0]):
    ema_loss_list[i] = alpha * loss_list[i] + (1 - alpha) * ema_loss_list[i - 1]

```

(2) EMA 代码参照第三章课件 60 页实现，代码如下图所示，最终结果如图2所示，可以发现经过指数平滑后的模型（类似于集成模型）收敛后更为稳健，在测试集上的表现略优于原始模型。

```python
class ModelEMA(object):
    def __init__(self, device, model, decay):
        self.ema = deepcopy(model)
        self.ema.to(device)
        self.ema.eval()
        self.decay = decay
        self.ema_has_module = hasattr(self.ema, 'module')
        self.param_keys = [k for k, _ in self.ema.named_parameters()]
        self.buffer_keys = [k for k, _ in self.ema.named_buffers()]
        for p in self.ema.parameters():
            p.requires_grad_(False)

    def update(self, model):
        needs_module = hasattr(model, 'module') and not self.ema_has_module
        with torch.no_grad():
            msd = model.state_dict()
            esd = self.ema.state_dict()
            for k in self.param_keys:
                if needs_module:
                    j = 'module.' + k
                else:
                    j = k
                model_v = msd[j].detach()
                ema_v = esd[k]
                esd[k].copy_(ema_v * self.decay + (1. - self.decay) * model_v)

            for k in self.buffer_keys:
                if needs_module:
                    j = 'module.' + k
                else:
                    j = k
                esd[k].copy_(msd[j])
```

图 2: 使用 EMA 平滑模型前后测试准确率随时间变化曲线