基于 RNN(LSTM) 的刀具磨损量预报

Tool wear forecasting based on Long-Short Term Memory(Recurrent Neural Network)

kidozh

November 18, 2018

西北工业大学

导言 i

概述

概述

背景

研究背景 i

随着信息技术的发展，使得传感器等智能工具被嵌入到加工装备上，如何针对所采集的、实时的传感器数据进行有效分析，提前预知或者检测刀具状态，成为了新的发展趋势。

常见的实时数据

1. 运动轴状态：电流、位置、速度、温度等
2. 主轴状态：功率、扭矩、速度、温度
3. 机床运行状态数据：温度、震动、PLC、I/O、报警和故障信息
4. 机床操作状态数据：开机、关机、断电等
5. 加工程序数据：工件、刀具、加工时间、程序执行时间
6. 传感器数据：振动信号、声发射信号

但是面临着巨大的信号以及信噪比较低的信号，根据我们实验的保守估计，以 20000Hz 的采样频率对加工工况监测，平均 10 分钟产生加工过程的信号大小达到

了 **150MB**（约等于 50 本中文版的哈姆莱特），这个时候使用人工的方法训练或者预测必然带来很多问题，比如

人工方法的不足

- 需要手工提取特征，难以处理复杂并且庞大的信息，同时难以迁移到其他的工况
- 通常信号和输出之间的关系非常复杂
- 往往依赖先验知识和有效的预处理方法

概述

研究简况

本研究提出了一种依赖刀具退化轨迹的循环神经网络 (基于长短期记忆模型) 的刀具磨损量预报方法

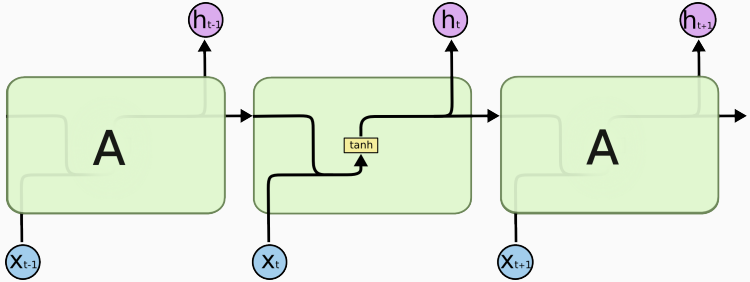


Figure 1: 基于 LSTM 的简单 RNN 模型

这种方法能有效的预报即将到来的刀具磨损，并且能通过自我指涉的方法把预测扩展到很长的一段时间。与基于 CNN 的事实监测结合并且针对 RNN 预报结果进行修正，得到最终综合结果。

优点

- 原生集针对信号进行处理，不依赖任何图像处理手段，不产生任何数据损失，保持了数据的原有结构
- 首个针对刀具磨损量进行预测的模型
- 能够通过自我指涉进行磨损量预报的延拓
- 预报时间和需要时间非常灵活，并且通用性极强
- 在短期结果预测较为精准，在较长的延拓空间内，自我指涉的结果仍然精准

理论基础

理论基础

RNN 网络基础

网络输入和输出 i

使用经典序列端到序列端模型针对 RNN 模型进行训练。

RNN 模型的输入和输出

入 t_{n-1} 和 t_n 两个时刻的磨损量

出 t_{n+1} 到 t_{n+5} 五个时刻的磨损量

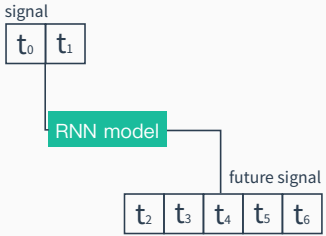


Figure 2: RNN 训练输入输出

在 RNN 模型中，我们使用基于LSTM（由 CUDA 加速）的所组成的 RNN 模型对样本进行训练。

网络输入和输出 iii

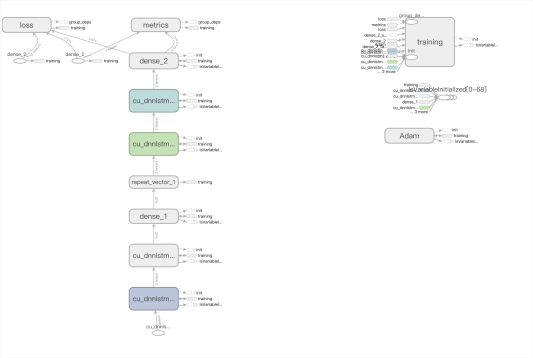


Figure 3: TensorFlow 可视化下的序列到序列（Seq2Seq）模型

LSTM 背景简介 i

长短期记忆（英语：Long Short-Term Memory，LSTM）是一种时间递归神经网络（RNN），论文首次发表于 1997 年。由于独特的设计结构，LSTM 适合于处理和预测时间序列中间隔和延迟非常长的重要事件。

LSTM 的表现通常比时间递归神经网络及隐马尔科夫模型（HMM）更好，比如用在不分段连续手写识别上。2009 年，用 LSTM 构建的人工神经网络模型赢得过 ICDAR 手写识别比赛冠军。LSTM 还普遍用于自主语音识别，2013 年運用 TIMIT 自然演講資料庫達成 17.7% 錯誤率的紀錄。作为非线性模型，LSTM 可作为复杂的非线性单元用于构造更大型深度神经网络。

理论基础

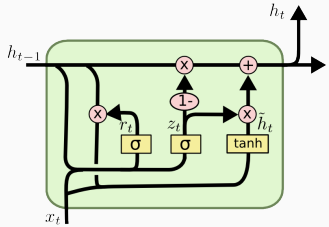
LSTM 原理

LSTM 原理 i

长短期记忆（英语：Long Short-Term Memory，LSTM）是一种时间递归神经网络（RNN），论文首次发表于 1997 年。由于独特的设计结构，LSTM 适合于处理和预测时间序列中间隔和延迟非常长的重要事件。

LSTM 的表现通常比时间递归神经网络及隐马尔科夫模型（HMM）更好，比如用在不分段连续手写识别上。2009 年，用 LSTM 构建的人工神经网络模型赢得过 ICDAR 手写识别比赛冠军。LSTM 还普遍用于自主语音识别，2013 年运用 TIMIT 自然演讲数据库达成 17.7% 错误率的纪录。作为非线性模型，LSTM 可作为复杂的非线性单元用于构造更大型深度神经网络。

图4显示了 LSTM 细胞内部的运算细节。

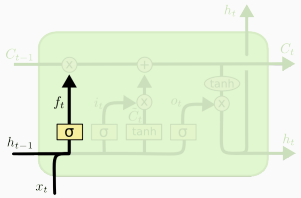


$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Figure 4: LSTM 内部原理图

LSTM 遗忘门

对于 LSTM 来说，最重要的就是决定在处理时序序列的时候要丢弃哪些信息。这也就是被遗忘门的 σ 激活层所决定的。其会被 h_{t-1} 和 x_t 所决定的，输出的是一个介于 0 到 1 之间的 C_{t-1} 值，其代表这个细胞是否遗忘上一个信息的依据。0 表示彻底遗忘这个信息，而 1 代表完全保留这个信息。

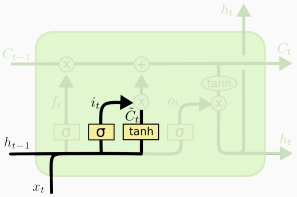


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Figure 5: 遗忘门

LSTM 储存机制 i

下一步就是确定 LSTM 将会怎样储存新的信息。其由两个机制决定。首先一个 σ 的输入门层决定 LSTM 将会更新的值，而另一个 *tanh* 层将会创建一个包含新候选的值 C_t，这个值有可能被添加到状态之中。接下来我们就会组合这两个值来更新 LSTM 的状态。



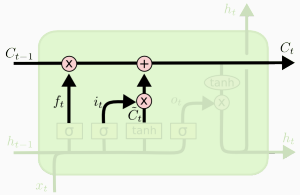
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Figure 6: 储存机制

LSTM 储存机制 ii

接下来就需要更新 LSTM 的状态值从 C_{t-1} 到 C_t。我们利用之前遗忘门所确定下来的丢弃之前信息的状态，我们通过将 $i_t * \tilde{C}_t$ 添加上，这样就可以得到新的 LSTM 状态。

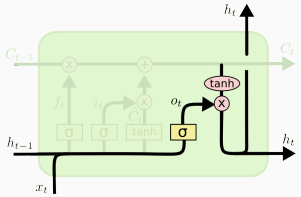


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Figure 7: LSTM 状态

LSTM 输出机制 i

最后我们需要确定输出的值，这个输出将会基于 LSTM 自身的状态。首先我们会应用一个 σ 层确定我们将会输出的 LSTM 的状态，之后我们将会应用一个 *tanh* 激活函数（将其映射到-1 和 1 之间）然后将其和 σ 激活后的结果相乘，这样就能得到最终的输出了。



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Figure 8: 输出状态

一个比较流行的版本就是由 Gers 和 Schmidhuber 在 2000 提出的添加 peephole 连接的变体，这样 LSTM 内部几个门之间都能参考细胞的状态。

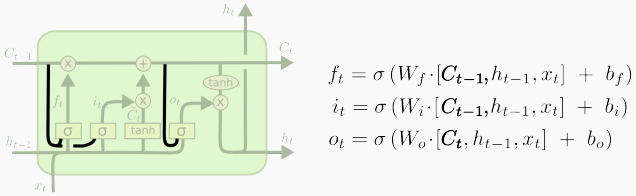
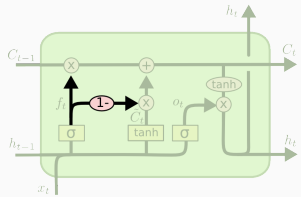


Figure 9: peepholes 连接

但是不是所有的论文都会将所有的 peephole 都加上的。

另一个变型的就是使用成对的遗忘和记忆门，这个方法将遗忘和添加信息综合考虑，我们只有在输入信息的时候才选择遗忘之前的信息，同时相应的当我们遗忘之前的信息的时候也会输入新的信息。

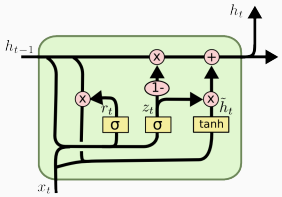


$$C_t = f_t * C_{t-1} + (1 - f_t) * \tilde{C}_t$$

Figure 10: 成对连接

LSTM 变体 iii

另一个比较有意思的变体就是 GRU（Gated Recurrent Unit，基于门机制的单元），其综合了遗忘和输出大一个更新的门，同事也能合并单元的状态和隐藏状态，并且也做了其他的改变。这个模型丢掉了储存单元，从而针对 LSTM 做了简化，在最近变得很流行了。



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Figure 11: GRU 内部细节

理论基础

RNN 重复出现构成一个新的层

通过将这些单元首尾相接，我们就可以得到 LSTM 链，也就是在网络中会使用的 RNN 层。

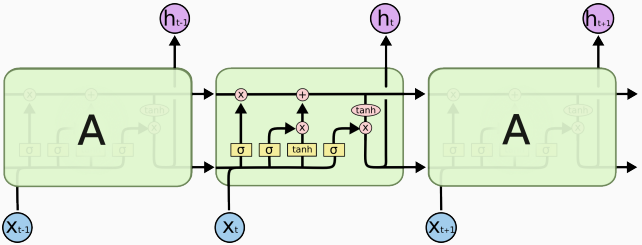


Figure 12: LSTM 成链组成 RNN 网络中的某层

这样我们观察输入输出，就可以得到一个输出状态与之前都具有很深刻的关系。

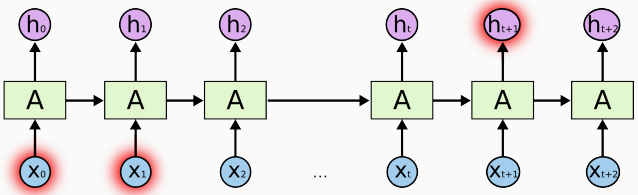


Figure 13: RNN 输入与输出之间的远期依赖关系

单元重复出现构成新层 iii

LSTM 明确旨在避免长期依赖性问题。长时间记住信息实际上是他们的默认行为，而不是他们难以学习的东西。同时，所有递归神经网络都具有神经网络重复模块链的形式。

理论基础

序列到序列模型（Sequence-to-sequence model）

seq2seq 原理 i

seq2seq 是一个 Encoder–Decoder 结构的网络，它的输入是一个序列，输出也是一个序列，Encoder 中将一个可变长度的信号序列变为固定长度的向量表达，Decoder 将这个固定长度的向量变成可变长度的目标的信号序列。

这个结构最重要的地方在于输入序列和输出序列的长度是可变的，可以用于翻译，聊天机器人，句法分析，文本摘要等。

基本的 seq2seq 模型包含了两个 RNN，解码器和编码器，如图14所示：

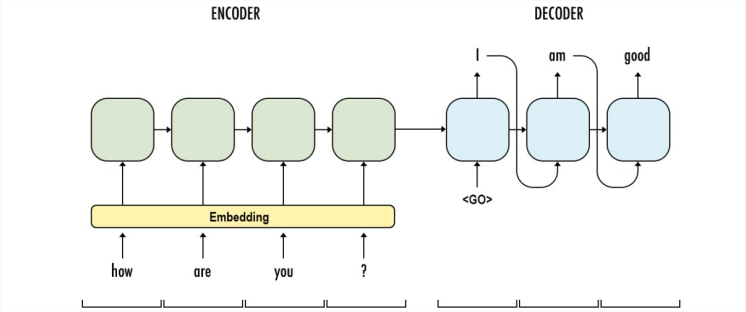


Figure 14: seq2seq 在翻译中的应用

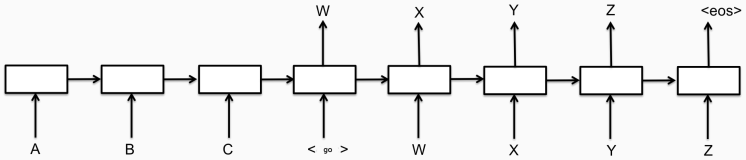


Figure 15: 基本 seq2seq 模型

图中每一个 box 代表了一个 RNN 单元，通常是 LSTM 或者 GRU。其实基础的 Seq2Seq 是有很多弊端的，首先 Encoder 将输入编码为固定大小状态向量的过程实际上是一个信息“信息有损压缩”的过程，如果信息量越大，那么这个转化向量的过程对信息的损失就越大，同时，随着 sequence length 的增加，意味着时间维度上的序列很长，RNN 模型也会出现梯度弥散。最后，基础的模型连接 Encoder 和

Decoder 模块的组件仅仅是一个固定大小的状态向量，这使得 Decoder 无法直接去关注到输入信息的更多细节。由于基础 Seq2Seq 的种种缺陷，随后引入了 Attention 的概念以及 Bi-directional encoder layer 等，由于我们的问题中的输入输出信号都是固定且长度较短，所以这些问题并不严重，对其他改进 tricks 先不做介绍。

基础的 Seq2Seq 主要包括 Encoder，Decoder，以及连接两者的固定大小的 State Vector。

理论基础

我们的模型

在 RNN 模型中，我们使用基于LSTM（由 CUDA 加速）的所组成的 RNN 模型对样本进行训练。

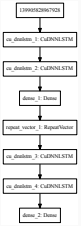


Figure 16: TCF 模型拓扑关系图

以 RepeatVector 为接线，靠近输入的所有 LSTM 构成一个编码器，靠近输出的所有 LSTM 构成一个解码器，RepeatVector 则作为保持状态的 StateVector。

通过这个构建方法，我们就完成了从已知数据到未来磨损量数据的端到端的 Seq2Seq 模型的建立。

实验

实验

采集样本

构建样本

使用 PHM 的针对刀具的磨损量作为样本来源，其包括了 3 把刀分别 315 个行程的刀具退化轨迹。我们将编号为 1,4 的刀退化轨迹作为训练集，6 作为验证集。

在每把刀的数据中，先取 $x_i \cdots x_{i+m}$ 作为模型输入，先取 $x_i \cdots x_{i+n}$ 作为模型输出，其中 $1 \leq i \leq 315 - n - 1$。这样就可以构建一个 Seq2Seq 模型。

在实验中，我们取输入的时间步长 m 为 2，输出的时间步长 n 为 3。这也就是说，通过了解过去连续两个时期的刀具磨损量的值，就可以得到未来三个时期的刀具磨损量的值。这也是一个经典的短期预报案例。

同时我们也试验了当 $m = 10, n = 20$ 的情况下的情况，即中远期预报结果。

实验

构建模型

构建模型

使用 Keras 构建之前所述的 Seq2seq 模型。编码器使用维度为 64 和 32 维度的 LSTM 层，而解码器则使用相对称的结果，最终连接一个全连接层输出结果。

需要注意的是，由于我们不使用 Masking 和嵌入层，故本模型不适用于变时间个数的信号处理。

因为较少存在孤立点，并且磨损量的结果是一个连续性的结果，使用均方误差（MSE）作为损失函数。使用 Adam 默认配置训练整个 seq2seq 模型。使用 Tensorboard 监测整个结果，并保留在验证集上损失最小的作为模型的结果。

模型结果

模型结果

短期预报结果

中短期预报损失函数随训练变化 i

在训练的过程中，Loss 下降呈现一个阶梯式下降，并且最终结果还是很理想的，loss 长期低于 1 并且仍有下降趋势，在训练过程中始终没有出现过拟合现象。模型随着训练轮数的增加，模型性能始终在好转，但是精度已经能够保障了。在训练中我们依然保留在验证集上效果最好的模型保留下来，作为最终的模型。

中短期预报损失函数随训练变化 ii

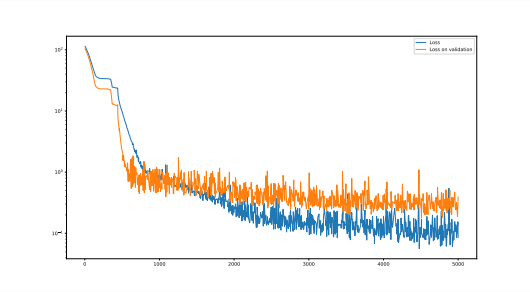


Figure 17: 中短期预报下的 RNN 模型 LOSS 下降结果（y 轴选用对数坐标）

中短期预报损失函数随训练变化 iii

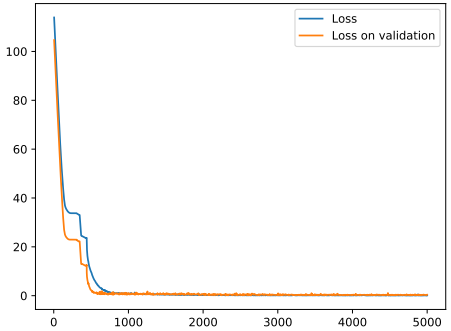


Figure 18: 中短期预报下的 RNN 模型 LOSS 下降结果

中短期预报结果 i

由于输出是一个 3 个连续时间磨损量的时间，我们相应的将每个磨损量按照编号命名为第 X 预报磨损量，X 为编号，针对每把刀，对应的第 X 预报磨损量如图所示。根据 Tensorboard 的监视结果，其均方误差和绝对误差均小于 1.

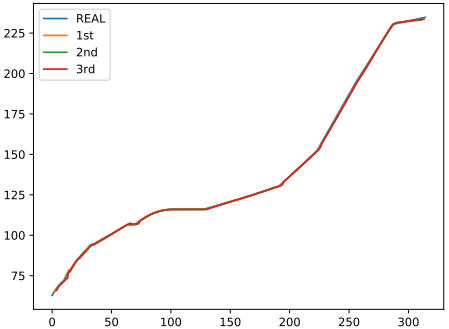


Figure 19: 第 1 把刀的预测结果

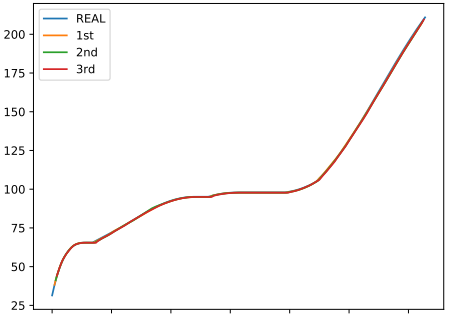


Figure 20: 第 4 把刀的预测结果

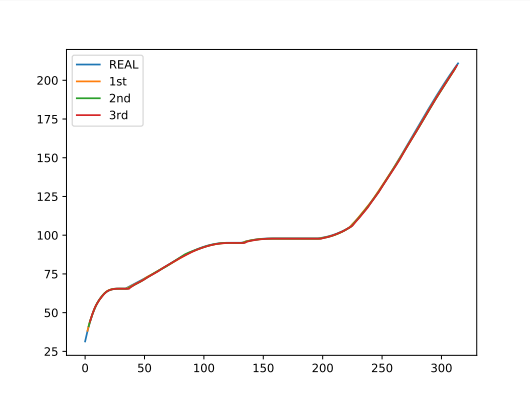


Figure 21: 第 6 把刀的预测结果（作为验证集）

中短期预报结果 v

在这里面 1,2,3 分别表示第 1,2,3 预报磨损量。可以看到在短期内都还是很精准的。换句话说，在有一个精确磨损量作为输入的情况下，输出结果总是很精准的，并且能够精准的预报整个阶段。

远期预报损失函数随训练变化 i

在训练的过程中，Loss 下降呈现一个阶梯式下降，并且最终结果还是很理想的，loss 长期低于 1 并且仍有下降趋势，在训练过程中始终没有出现过拟合现象。模型随着训练轮数的增加，模型性能始终在好转，但是精度已经能够保障了。在训练中我们依然保留在验证集上效果最好的模型保留下来，作为最终的模型。

远期预报损失函数随训练变化 ii

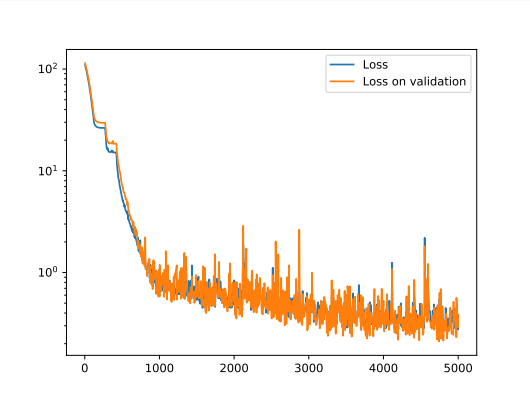


Figure 22: 远期预报下的 RNN 模型 LOSS 下降结果（y 轴选用对数坐标）

远期预报损失函数随训练变化 iii

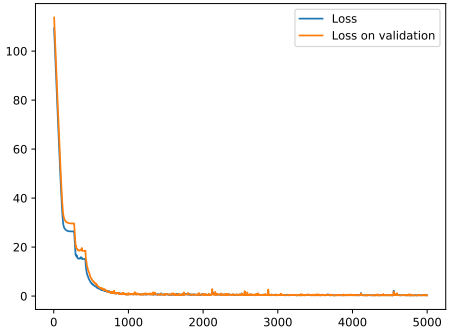


Figure 23: 远期预报下的 RNN 模型 LOSS 下降结果

自我指涉

自我指涉

介绍自我指涉

自我指涉背景 i

事实上，由于输出的预报磨损量长度大于输入预报长度，那么通过循环的将模型预报的输出作为下一个时刻的输入，我们就完成了对磨损量的延拓。

优点

1. 通过自我指涉，就可以将短期的预报拓展到长期的预报
2. 在中期进行自我指涉，可以累计之前预测的 N 种结果，修正出较为精确的值

缺点

1. 自我指涉容易引起误差的积累，并且积累往往呈现指数爆炸的效果
2. 自我指涉容易陷入鞍点和极值

自我指涉

自我指涉结果

短期预报的自我指涉 i

我们做了一个动画，介绍了在短期预报下每个点开始进行自我指涉的结果

长期预报的自我指涉 i

我们做了一个动画，介绍了在远期预报下每个点开始进行自我指涉的结果

自我指涉

自我指涉特征

自我指涉 i

在自我指涉中，很明显短期指涉精度不是很高，有效预报扩展范围最多扩展到 10-20 左右，而长期指涉精度和有效范围会长于短期的。

然而长期指涉因为可重复判断的值更多，所以在稳定性上表现更为突出，同时磨损量抑制表现也很明显。但是长期指涉也使得样本急速下降，在少样本的时候并不适合使用这种方法，同时由于输出的时间步变长，也使得误差提升。

但是可以看到的是，无论是短期和长期指涉都对平台期转向明显磨损的时候表现出极大的不稳定，将会出现使得远期预报结果呈现不变或者爆炸的结果。

无论怎么样说，由于自我指涉的误差富集效果和不确定因素，总使得在远期预报的结果与真实误差极大，所以修正是有必要的。

自我指涉

改进自我指涉方法

引入 CNN 监测值结果 i

在之前的研究中，我们获得了在单工况下较好的监测结果，而这个值就能作为预报结果的实时修正依据。而监测结果是较为精准的，所以可以借用之前带有真实值的修正图，可以事实修正结果。并且加上自我指涉的积累预报值，将会改进两个模型的结果。

具体的综合结果可以参考之前的第 X 预报值的结果，在监测值较好的情况下，能够得到比较好的结果

结论

结论

结论

结论 i

本方法提出了一个高精度的刀具磨损量预报结果，同时介绍了通过自我指涉的方法也扩展了预报范围的方法，并且也提出了一个综合提升检测和预测结果的方法来提高预测和监测的结果。

致谢

致谢

致谢

致谢

感谢你的关注!

在下面的 repo 上关注本工作的最新进展

https://github.com/kidozh/rnn_in_prediction_tool_wear

kidozh

西北工业大学