大模型原理与应用

第 10 课：行业应用实践 | 时间序列 Foundation 模型

以 TimesFM 与 Chronos 为例，看预测任务如何进入大模型时代

这一课看什么

- 为什么时间序列预测也开始走向 foundation model
- TimesFM 和 Chronos 分别代表什么路线
- 时间序列 token 化、zero-shot forecasting 和概率预测在说什么
- 这些模型真正适合什么场景，又有什么边界

本课主线

1. 先回答：为什么 forecasting 也会走向“预训练模型”
2. 再看 TimesFM 和 Chronos 的两条典型路线
3. 然后把它们放进真实系统场景里理解价值
4. 最后回到能力边界与工程判断

为什么时间序列也会走向 foundation model

传统时间序列预测常见做法是：

- 为每个数据集单独选模型
- 反复调参
- 在新场景上重新训练

这样的问题是：

- 迁移成本高
- 新任务冷启动慢
- 很难形成统一基线

所以大家自然会问：

> 能不能先在大量时序数据上预训练一个模型，再把它迁移到新任务？

什么是时间序列 Foundation 模型

它通常有几个共同特点：

- 在大规模、多领域、多频率时间序列上预训练
- 学习可迁移的时序模式
- 尽量减少新任务上的专门训练
- 强调 zero-shot 或 few-shot forecasting

一句话理解：

- 从“一个数据集一个模型”
- 走向“一个预训练模型适配很多数据集”

先把任务地图打开

时间序列任务不只有一个“预测明天值多少”。

常见任务包括：

- 单变量预测
- 多变量预测
- 概率预测
- 异常检测
- 事件预测
- 负荷 / 流量 / 传感器趋势预测

但目前最成熟、最容易进入 foundation model 叙事的，还是：

- forecasting

为什么 forecasting 特别适合先走起来

因为它有几个天然优势：

- 任务定义清楚
- 历史序列到未来序列的接口统一
- 评价指标成熟
- 很多行业都需要

所以它很像 NLP 里的语言建模：

- 输入是一段上下文
- 输出是未来的延续

时间序列 Foundation 模型在想什么

它们想解决的核心问题并不复杂：

- 能不能把时序模式学成一种可迁移先验？

也就是说：

- 趋势
- 周期
- 波动
- 局部异常
- 跨尺度变化

这些模式，能不能像语言模式一样，被预训练模型反复复用？

一个快速全景

近几年有代表性的时间序列 foundation model 包括：

- TimesFM
- Chronos / Chronos-Bolt / Chronos-2
- Moirai
- MOMENT
- TimeGPT

这一课重点只抓两条路线：

- TimesFM
- Chronos

因为它们最适合课堂入门，也最适合做最小 demo。

TimesFM：forecasting-native 路线

TimesFM 可以理解成：

- 专门为 forecasting 设计的 foundation model

它的关键词是：

- decoder-only
- zero-shot forecasting
- 大规模时序预训练

它不是把时间序列硬套成“语言”，而是更直接地把 forecasting 本身当成主任务。

TimesFM 为什么值得讲

因为它代表了一种很清楚的态度：

- 预测任务足够重要，值得直接为它设计 foundation model

教学上它有几个优点：

- 叙事简单
- 任务边界清楚
- 很适合和传统 forecasting baseline 对比

所以 TimesFM 更像：

- 为 forecasting 原生设计的大模型

TimesFM 的公开版本说明了什么

目前公开版本里，TimesFM 的主模型规模大约在：

- 200M

这说明它并不是一个“特别轻”的课堂玩具模型，而更像：

- 一个中等规模的统一 forecasting 基础模型

你可以把它理解成：

- 不追求最小
- 更强调作为统一强基线

Chronos：把时间序列写成“语言”

Chronos 的思路非常适合课堂讲，因为它很直白：

1. 先对数值序列做缩放
2. 再把数值量化成 token
3. 把时间序列变成 token 序列
4. 用类似语言模型的方式做训练和生成

一句话理解：

- Chronos 在尝试“学习时间序列的语言”

Chronos 为什么有代表性

因为它把一个很重要的问题讲清楚了：

- 如果 LLM 可以建模 token 序列
- 那时间序列能不能先变成 token，再交给类似框架处理?

这就是它最有启发性的地方：

- 它不是为 forecasting 重新发明全部结构
- 而是借用了语言模型式建模思路

Chronos 在优化什么

Chronos 不只是做一个点预测。

它更强调：

- 概率预测
- 多条未来轨迹采样
- 预测区间

这意味着它给出的不是：

- “未来一定是这个值”

而更像是：

- “未来大概落在这个范围里”

为什么概率预测很重要

因为真实系统里，决策很少只依赖一个点预测。

例如：

- 电力负荷调度
- 网络流量预估
- 设备维护预警
- 传感器异常监控

真正关心的往往是：

- 不确定性有多大
- 风险上界在哪里
- 这个预测稳不稳

所以 Chronos 的“采样未来轨迹”非常有工程意义。

Chronos 家族为什么适合课堂

Chronos 不是单一一个模型，而是一组不同尺寸的家族：

- `chronos-t5-tiny`
- `chronos-t5-mini`
- `chronos-t5-small`
- `chronos-t5-base`
- `chronos-t5-large`

以及：

- Chronos-Bolt
- Chronos-2

这很适合课堂，因为它让学生立刻看到：

- 同一思想可以有 tiny 到 large 的多种落地方式

Chronos-Bolt / Chronos-2 在说明什么

这两条后续路线说明了一件事：

- 时间序列 foundation model 也会像文本模型一样继续分化

大家关心的维度包括：

- 精度
- 推理速度
- 内存占用
- 多变量支持
- 外生变量支持

也就是说，这条路线已经不只是“能不能做”，而是进入了：

- 怎样做得更快、更稳、更通用

只做 zero-shot 还不够时，下一步是什么

讲到这里，学生通常会自然问：

- 如果 zero-shot 已经能跑，下一步怎么继续提升？

最直接的答案就是：

- fine-tune

也就是：

- 保留预训练模型作为底座
- 再用当前任务的数据做一轮更贴近本场景的训练

为什么 Chronos 特别适合讲 fine-tune

因为 Chronos 的一个巨大教学优势是：

- 它底层并不是一套特别封闭、特别神秘的新结构

对当前 `chronos-t5-tiny` 来说，更准确的说法是：

- 它底层就是一个带 Chronos 配置的 `T5ForConditionalGeneration`

这意味着：

- 预训练部分负责给出时序先验
- fine-tune 部分可以直接复用 Hugging Face / PyTorch 的标准训练套路

Chronos fine-tune 在工程上为什么方便

因为你真正要补的，只剩下三层东西：

1. 数据怎么切成 `history -> future`
2. 数值怎么经过 Chronos tokenizer 变成 token
3. 训练循环怎么把 `input_ids / attention_mask / labels` 喂给 T5

也就是说：

- 模型结构不用重写
- loss 不用自己发明
- 训练接口也和常见 seq2seq 模型高度一致

Chronos fine-tune 的最小闭环长什么样

我们现在写的 demo，本质上就是这 5 步：

1. 从 ETTh1 里切很多滑动窗口
2. 每个窗口拆成 `context` 和 `future`
3. 用 Chronos tokenizer 把数值序列转成 token
4. 用 `T5ForConditionalGeneration` 做 teacher-forcing 训练
5. 用 `ChronosPipeline` 比较微调前后预测

它看起来像“时间序列脚本”，但核心骨架其实很像文本 seq2seq 微调。

为什么说它“基于 T5，所以 fine-tune 很方便”

这里的“方便”不是说：

- 一定效果最好

而是说：

- 训练接口成熟
- 模型类现成可用
- 参数保存和加载都标准化
- 很容易接进现有 PyTorch / Transformers 工作流

对于教学来说，这一点尤其重要，因为它让学生能把：

- 文本大模型微调经验

迁移到：

- 时间序列 foundation model 微调

Chronos fine-tune 真正需要学生理解什么

不是“把代码跑完”而已，而是理解这三件事：

1. 为什么先做滑动窗口切片
2. 为什么数值序列不能直接硬喂给 T5，而要先 token 化
3. 为什么最后一定要比较 fine-tune 前后的预测图和误差

换句话说：

- Chronos 的难点不在训练循环本身
- 而在“时间序列如何变成语言模型可处理的对象”

如果把 Chronos 看成一条路线

那它的 fine-tune 思路可以概括成：

- 先把时间序列翻译成 token 语言
- 再用标准 seq2seq 模型继续学习

所以它的优势是：

- 实现路径清楚
- 教学闭环很短
- 很容易做最小可运行实验

代价则是：

- tokenization 设计本身会影响表现
- prediction length、量化方式和配置耦合较强

TimesFM 和 Chronos 的真正差别

最核心的差别不是“谁分数高一点”，而是问题 framing 不一样。

TimesFM 更像：

- forecasting-native foundation model

Chronos 更像：

- language-model-inspired time series model

也就是：

- 一个更像“为预测而生”
- 一个更像“把时间序列翻译成 token 语言”

参数规模能说明什么，又说明不了什么

Chronos 家族从 tiny 到 large，尺度更灵活。
TimesFM 当前公开主版本更像一个固定主力版本。

所以一个简单结论是：

- 如果看最小模型，Chronos 更容易做课堂 demo

但要强调：

- 参数更小，不等于一定更好
- 参数更多，也不自动意味着更适合所有行业任务

真正要看的是：

- 任务类型
- 数据分布
- 预测跨度
- 资源约束

这些模型为什么和行业应用直接相关

因为行业里最常见的问题之一就是：

- 新场景来了，但没时间重做一整套 forecasting pipeline

这时 foundation model 的吸引力很强：

- 先给一个 zero-shot baseline
- 快速看有没有可迁移模式
- 再决定要不要继续做专门微调或系统改造

哪些场景特别适合先试 TS foundation model

- 电力负荷预测
- 流量 / 带宽趋势预测
- 传感器序列预测
- 设备监测信号趋势预测
- 工业过程控制中的短期预测

这些场景的共同点是：

- 序列很多
- 分布异构
- 需要快速建立可用 baseline

Chronos + ETTh

因为它满足几个条件：

- 数据公开
- 任务简单
- 预测图直观
- 很容易解释“历史 -> 未来”的接口

演示：

- foundation model 不是只会聊天
- 它也可以进入电力、流量、传感器这类行业序列问题

Chronos demo 里的预测图怎么读

课堂里那张图，至少要看懂四件事：

- `history`
  - 模型真正看到的历史窗口
- `ground truth`
  - 真实发生的未来序列
- `median forecast`
  - 多次采样后得到的中位预测
- `prediction interval`
  - 未来可能落入的一个预测范围

这张图最关键的意义不是“红线有没有完全压住黑线”，而是：

- 模型有没有抓住趋势
- 区间宽度是否合理
- 未来不确定性是否被表达出来

TS foundation model 最吸引人的地方

- 统一建模视角
- 强 zero-shot baseline
- 更低的跨数据集迁移成本
- 更快的冷启动

尤其在标注有限或时间紧张的系统里，它非常有吸引力。

但一定要讲清楚边界

这类模型不是一上来就会替代所有传统 forecasting pipeline。

几个关键限制包括：

- zero-shot 不一定最优
- domain shift 仍然很关键
- 多变量和外生变量处理仍有难点
- 长预测跨度会明显变难
- 可解释性和效率还在持续改进

它们最可能在哪些地方失败

- 数据分布和预训练语料差得太远
- 任务特别依赖强领域约束
- 预测跨度太长
- 多变量依赖关系很强
- 决策环节对误差极其敏感

所以教学上一定要让学生形成一个判断：

- foundation model 很强，但它首先是 baseline，不是魔法

这一课和整个课程怎么接起来

它其实在重复整门课的一条总逻辑：

- 先有预训练
- 再有迁移
- 再进入行业场景

只不过这里的对象换成了：

- 时间序列

这说明 foundation model 这套思想，并不只属于文本和图像。

这节课你该带走什么

1. 时间序列预测也正在走向 foundation model
2. TimesFM 代表 forecasting-native 路线
3. Chronos 代表 tokenized time series 路线
4. 概率预测和预测区间是时间序列场景里的关键价值
5. 这些模型很适合做跨场景 zero-shot baseline
6. 但真正落地时，仍然要面对 domain shift、预测跨度和系统约束

第 5 课收束

时间序列 Foundation 模型的关键，不只是“能预测”，而是把可迁移时序先验带进真实系统

- 先学跨数据集的时序模式
- 再把它迁移到新的行业序列
- 最后再用工程判断决定它是否真的值得落地