光华管理学院 Guanghua School of Management

因思想 而光华

预测与 MoE
Forecasting and MoE

李丰

北京大学光华管理学院

https://feng.li/forecasting-with-ai

Mixture of Experts

为什么需要 Mixture of Experts（MoE）？

- 大模型越来越大，但不是所有任务都需要一个“巨无霸”
- 企业真实问题：需求预测、用户画像、营销投放、风险控制——场景差异巨大
- 一个模型难以擅长所有任务（“No Free Lunch”）
- MoE 的核心思想：
  - “把复杂问题分给更擅长的小专家，高效又精准。”

时间序列预测的 No Free Lunch 理论

- **No Free Lunch 理论（NFL）**
  - Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation, 1(1), 67–82.
  - 没有任何预测模型能在所有场景下表现最好。
  - 每一种模型都只对某些类型的数据/业务特别擅长。
- **为什么预测中会出现 No Free Lunch?**
  - 时间序列预测的世界很复杂：
    - 有些产品 → 稳定有趋势（如：咖啡豆）
    - 有些产品 → 明显季节性（如：空调）
    - 有些产品 → 大促跳点（如：电商快消）
    - 有些 → 完全随机（如：短信验证码）
    - 有些 → 长尾断货（intermittent demand）
  - 不同的模式 → 不同的最优模型
    - ARIMA 擅长趋势
    - ETS 擅长季节
    - Prophet 擅长节假日
    - XGBoost/LSTM 擅长复杂非线性
    - 大模型（LLM/TimeGPT）擅长跨序列迁移
  - 没有哪个模型可以全都擅长。

No Free Lunch 在预测中的形式化

- 如果你把所有可能的数据集平均起来：
  - 所有预测模型的**平均表现都是一样的**。
  - 不存在一个永远最优的“万能预测模型”。

- 在真实企业预测里：不能指望一个模型吃遍所有 SKU、区域、品类、促销场景。

- 很多 人 认为：“用 TimeGPT / DeepSeek / GPT-5 不就能解决了吗？”
  - 答案是：✖ 。大模型（LLM/FMs）也无法逃避 No Free Lunch
- 原因：
  - 虽然大模型非常强，但仍然不可能在所有类别的时间序列上同时最佳。
  - 趋势型序列、节季型序列、噪声型序列、促销型序列 → 需要不同的专长。

- MoE（混合专家）就是大模型用来应对 NFL 的策略之一。

MoE（Mixture of Experts）解决方案

- 不同时间序列 → 激活不同专家
  - 趋势专家
  - 季节专家
  - 促销专家
  - 外生变量专家
  - 异常专家
- Gating 让模型自动判断：“用谁做预测？”

- 这等于建立一个自动化的预测团队，而不是靠一个“巨无霸模型”，由多个小模型（仍然可以是LLM，但是规模和参数很小）组成。

MoE 从经典模型到现代深度学习

- Jacobs, R. A., Jordan, M. I., Nowlan, S. J., & **Hinton**, G. E. (1991). Adaptive mixtures of local experts. Neural Computation, 3(1), 79–87.
  - MoE 概念首次提出
  - 使用 Gating network 选择专家
  - 基础思想：不同专家处理不同输入区域
- Shazeer, N., Mirhoseini, Azalia, Maziarz, Krzysztof, Davis, A., Le, Q., Hinton, G., & Dean, J. (2017). Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. International Conference on Learning Representations.
  - Google 深度 MoE 的奠基之作
  - 引入 Top-K gating（稀疏激活）
  - 每次只激活部分专家 → 巨大规模 + 低成本
  - 后来 GPT-4、DeepSeek、LLaMA 都受它启发
- Fedus, W., Zoph, B., & Shazeer, N. (2022). Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. The Journal of Machine Learning Research, 23(1), 120:5232-120:5270.
  - Google 最著名的 MoE 模型
  - 每层只选 1 个专家（Switch gating）
  - 训练速度极快，参数上千亿，成本可控

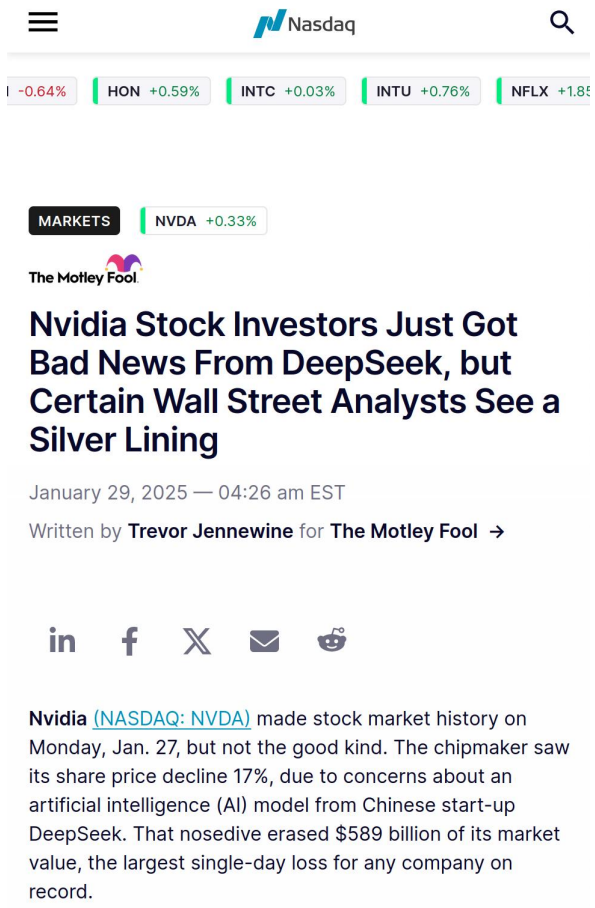
通用大模型（LLM）中的 MoE

- **GPT-4（推测）**
  - 公认为采用 Sparse MoE
  - 多层专家结构
  - 参数量大但激活参数少 → 性能高、成本低
- **DeepSeek MoE（DeepSeek-V2/V3/R1）**
  - Hybrid MoE（Dense + MoE）
  - 有共享专家 + 专家组
  - Token-level routing
  - 更稳定、更具工业化特点
  - 特别适合预测任务（heterogeneous time series）
  - 引发了 DeepSeek 做空英伟达
- **LLaMA-3.1 70B MoE（Meta）**
  - 最新开源 MoE LLM
  - Top-k MoE，每次激活两个专家
  - 非常高效
- **Mixtral (Mistral AI, 2023)**
  - 模型：Mixtral 8×7B
  - 8 个专家，每次激活 2 个
  - 开源性能极强（可媲美 GPT-3.5）
  - MoE 实用化的重要标志

小故事： DeepSeek 事件做空英伟达

事件概况

- 2025年1 月 27 日，DeepSeek 发布其新模型，引起市场强烈反应，英伟达股价单日下跌约 17%，市值蒸发数千亿美元。
- 短线做空英伟达的市场参与者据称从此次下跌中获利约 66 亿美元。
- 媒体将其称作“英伟达被 DeepSeek 恐慌性冲击”的一次典型事件。
- 市场情绪被触发。在 AI 热潮中，英伟达估值已经非常高。任何对其增长路径或基础设施模式的怀疑，都可能成为“做空”或风险重估的触发器。



做空英伟达的大致机制

- 虽然具体做空交易人的全部细节不可得，但从公开报道可以推断如下流程：
- 做空者观察到 DeepSeek 发布其模型，同时市场对 AI 基础设施依赖的假设受到挑战。
- 建立英伟达空仓或买入看跌期权，或通过衍生工具押注英伟达股价下跌。
- 随着英伟达股价因为 DeepSeek 的冲击而快速下跌，空头获利。
- 这一过程也伴随市场做空成本增加、监管介入（比如 “uptick rule” 等）的问题。

Mixture of Experts（MoE）自动化预测团队

现实团队	MoE 中的对应
每个分析师（专家）对某类数据很擅长	Expert（趋势、季节、促销、异常……）
团队负责人分配任务	Gating（路由器）
每个任务可能由不同人完成	Token-level / sample-level routing
最终由负责人综合意见	Expert 加权融合

MoE 的本质就是：

- 一个能自动分工协作的预测团队结构。
- 而不是一个“什么都试图做、但什么都不精”的巨无霸单模型。

预测（Forecasting）领域中的 MoE

- TimeGPT（Nixtla）
  - 使用 Mixture-of-Experts + Transformer 架构
  - 负责分配不同时间序列结构的专家
  - 特别擅长跨序列泛化
- Chronos-Bolt（Amazon, 2024）
  - Amazon 的大规模时间序列模型
  - MoE 结构 + Token routing
  - 为百万级 SKU 提供预测
- Google Moirai（2024, 世界最大 TS 模型）
  - 100B+ 参数 + Sparse MoE
  - 面向时序预测
  - Gating 自动为不同 regions / patterns 分派专家
  - 强调 scaling laws 与推理效率

MoE 的数学表示

MoE in Math

输入表示（时间序列 Embedding）

对时间序列（y_t）进行编码：

$$x_t = f_{\text{embed}}(y_{t-L:t}, \mathbf{s}_t, \mathbf{z}_t)$$

其中：

- $y_{t-L:t}$：过去 (L) 个历史点
- $\mathbf{s}_t$：时间特征（weekday, holiday, season）
- $\mathbf{z}_t$：外生变量（价格、天气、宏观等）
- x_t：Embedding

路由器（Gating Network）

- MoE 的核心是**给每个时间点选择不同专家**。
- 对每一个时间点 (t)，路由器给出对 (K) 个专家的权重分布：

$$\mathbf{g}_t = \text{softmax}(W_g \mathbf{x}_t + b_g), g_t^{(k)} = \frac{\exp(\alpha_t^{(k)})}{\sum_{j=1}^{K} \exp(\alpha_t^{(j)})}$$

其中：

- $g_t^{(k)}$：第 (k) 个专家在时间点 (t) 的权重
- W_g, b_g：Gating 参数
- $K < \infty$：专家数量

Sparse MoE（DeepSeek、Moirai、Mixtral 等）采用 Top-(m) 激活：

$$g_t^{(k)} = 0 \quad \text{if } k \notin \text{Top-}m(\mathbf{g}_t)$$

专家（Expert）模块

每个专家 ($\mathbf{E_k}$) 是一个预测子模型，例如：

- 小 Transformer block
- 小 MLP（MLP expert）
- 小 LSTM / Dilated CNN expert
- Domain expert（趋势 / 季节 / 促销 / 异常）

其形式为：

$$\mathbf{h}_t^{(k)} = E_k(\mathbf{x}_t)$$

其中：$\mathbf{h}_t^{(k)}$ 是第 (k) 个专家的隐藏输出。可以看成：

$$\mathbf{h}_t^{(k)} = E_k(\mathbf{x}_t) = \sigma(W_k\mathbf{x}_t + b_k)$$

或 Transformer block：

$$\mathbf{h}^{(k)} = \mathrm{FFN}_k(\mathrm{MHSA}_k(\mathbf{x}))$$

其中MHSA_k为多头自注意力 (Multi-Head Self-Attention)，FFN_k为前馈网络（Feed-Forward Network）

MoE 输出融合（Mixture Aggregation）

MoE 的融合是**加权专家输出**：

$$\mathbf{h}_t = \sum_{k=1}^{K} g_t^{(k)}, \mathbf{h}_t^{(k)}$$

如果是 Sparse MoE：

$$\mathbf{h}_t = \sum_{k \in \text{Top-}m(t)}^{K} g_t^{(k)}, \mathbf{h}_t^{(k)}$$

最终预测头（Forecasting Head）

输出未来 (H) 期预测：

$$\hat{y}_{t+1:t+H} = f_{\text{forecast}}(\mathbf{h}_t)$$

或逐步预测：

$$\hat{y}_{t+h} = W_o^{(h)}\mathbf{h}_t + b_o^{(h)}, \quad h = 1, \ldots, H$$

把所有步骤合并：

- $$\boxed{\hat{y}_{t+h} = f_{\text{forecast}}\left(\sum_{k=1}^{K} g_t^{(k)} E_k\left(f_{\text{embed}}(y_{t-L:t}, \mathbf{s}_t, \mathbf{z}_t)\right)\right)}$$

DeepSeek、TimeGPT、Moirai 的 MoE 特点

DeepSeekMoE（Hybrid MoE）

在 MoE 之前加入 dense FFN（稳定性）：

$$\mathbf{u}_t = \mathrm{FFN}_{dense}(\mathbf{x}_t),$$

$$\mathbf{h}_t = \mathbf{u}_t + \sum_{k \in \mathrm{Top}\text{-}m(t)}^{K} g_t^{(k)} E_k(\mathbf{x}_t)$$

TimeGPT / Moirai（Forecasting FM）

- 使用跨序列共享专家：$E_k(\mathbf{x}_t; \theta_k)$, 其中 θ_k 在不同 SKU / 城市 / 国家共享。

什么是大模型的“蒸馏”（Distillation）？

- 蒸馏就是让一个大模型（老师）教出一个更小、更快、更便宜的模型（学生），而且保留主要能力。
- 就像把一锅汤熬浓，把精华提取出来。
- 大模型（teacher）包含大量知识。
  - 在蒸馏过程中把不重要的部分“挥发掉”，把关键信息“提纯”
- 得到一个更轻、更高效的小模型（student）
- 它不是简单裁剪，而是 “知识迁移 + 模型压缩”。

蒸馏的三个核心步骤

老师模型生成大量高质量数据或答案

学生模型学习老师的“思考方式”、判断边界、输出概率分布。

- 学生模型模仿老师的行为
- 优化目标从模仿正确答案变成模仿老师的输出分布（更细腻）。
- 学生模型学习这种“软判断”。得到一个更小、更便宜能跑在边缘设备的模型
- 如：Teacher = 70B, Student = 7B
- 但保留了 Teacher 70%~90% 的能力。

蒸馏的主要类型

- **Logit Distillation（经典蒸馏）**
  - 学习 Teacher 的输出概率。
- **Response Distillation**（答案蒸馏）
  - 学生学习 Teacher 的回答、推理链。
- **Self-Distillation**（大模型自我蒸馏）
  - 如 DeepSeek-R1、OpenAI O1
  - 老师是自己生成的“更优解” 或 “延长推理”
  - 学生学习更短、更高效的推理方式。
- **Preference Distillation**（偏好蒸馏）
  - 把 RLHF 的偏好传给学生，提升对齐程度。

蒸馏与 MoE、预测有什么关系？

- 让 MoE 的专家更轻量
  - 大模型 MoE 专家往往 1–7B 规模，蒸馏让它们轻但保持能力。
  - 让推理（预测）更快
- 时间序列预测需要速度：
  - 存货盘点
  - 电商分钟级更新
  - 金融高频风险识别
  - 蒸馏后的模型能实现10 倍到 100 倍加速。
- 大模型界最重要的蒸馏应用
  - GPT-4 → GPT-4-mini / GPT-3.5
  - Mixtral MoE → Distilled Mixtral
  - BERT → DistilBERT（最早的成功案例）
  - LLM → LLM for Forecasting（TimeGPT 蒸馏小模型）

拆解 Chronos 2 的蒸馏过程

Knowledge Distillation for Chronos 2

蒸馏目标

- Chronos 2 是 HuggingFace 的 预训练时间序列大模型（TS-FM），本质是一个 tokenized 时序 Transformer。蒸馏目标通常有三类:

- **压缩模型**: 把 Chronos 2 的 decoder/encoder 层数、hidden size、num heads 降低一半或更多，把推理成本降到原来的 1/3–1/10。
- **领域迁移** (Domain Transfer): 让 student 模型更适合例如航空旅游预测、能源负荷、高频金融数据或者多任务预测。
- **多步预测更稳定**: 多步预测 (H > 1) 容易出现漂移(drift)，通过蒸馏可以让 student 的多步 distribution 更稳、时间一致性更好。

Teacher 模型

- Teacher = Chronos 2 预训练 checkpoint
  - `from transformers import ChronosForPrediction, ChronosTokenizer`
  - `teacher = ChronosForPrediction.from_pretrained("amazon/chronos-t5-base")`
  - `tokenizer = ChronosTokenizer.from_pretrained("amazon/chronos-t5-base")`

- Student 模型结构: 一般降一半即可:
  - `base: 12 layers → 6 layers`
  - `hidden = 768 → 384`
  - `heads = 12 → 6`

  - `from transformers import ChronosConfig, ChronosForPrediction`
  - `config = ChronosConfig.from_pretrained("amazon/chronos-t5-base")`
  - `config.num_layers = config.num_layers // 2`
  - `config.hidden_size = config.hidden_size // 2`
  - `config.num_heads  = config.num_heads // 2`
  - `student = ChronosForPrediction(config)`

蒸馏数据（关键）

- Chronos 的输入是 quantized time series tokens，即把连续的时间序列（real-valued time series）转换成离散 tokens，让时间序列可以像自然语言一样输入 Transformer。
- 你可以用：
  - (A) 真实时序数据（旅游、电力、航空）
  - (B) 让 teacher 对训练集做 self-training：
  - Teacher 生成未来 H=1,⋯,24 horizon 的分布（mean, variance, quantiles）

Token-level 蒸馏

- Teacher 输出 token 的概率分布
- Student 学这个分布
  - $Loss = KL(P_{Teacher}(\cdot\,|x)||P_{Student}(\cdot\,|x))$

Distribution-level 蒸馏

- Chronos 输出的预测是：quantiles, point

- 把 Teacher 的分布参数蒸馏到 student

  - $Loss = \sum_q ||\hat{y}_q^{(T)} - \hat{y}_q^{(S)}||$

蒸馏的其他技巧

- 数据增强会让 student 更“稳”。
  - random window crop
  - additive noise (Gaussian)
  - scaling (1.0–1.25 random)
  - time warping
  - seasonal shifting
- 温度调节：开始时使用较高温度（4-10），逐步降低到1
- 损失权重：软标签和硬标签的平衡很重要，通常α在0.5-0.9
- 批量大小：使用较大的批量大小以获得更稳定的软标签
- 学习率：通常比正常训练小1-2个数量级
- 验证策略：同时监控验证集准确率和蒸馏损失

常见问题解决

- 学生模型无法收敛
  - 解决方案：降低学习率，增加温度
- 性能差距过大
  - 解决方案：尝试特征蒸馏或关系蒸馏
- 过拟合问题
  - 解决方案：添加更多的数据增强，使用早停
- 内存不足
  - 解决方案：使用梯度累积，减少批量大小
- 蒸馏的关键是耐心调试超参数，特别是温度、损失权重和学习率。不同的任务和模型组合可能需要不同的配置。