时间序列分析与预测

第九讲

黄嘉平

深圳大学｜中国经济特区研究中心
粤海校区汇文楼办公楼 1510
课程网站 https://huangjp.com/TSAF/

1. 关于人工智能

1.1. 人工智能的发展历程

1950 年代： AI 研究的黎明期

- Turing, A. M. (1950). Computing Machinery and Intelligence. *Mind*, LIX(236), 433-460.

  Alan Turing 通常被认为是现代人工智能研究领域的奠基者。他在 Turing (1950) 中提出了著名的 imitation game（即图灵测试）以检验计算机是否拥有“思考”能力。

- 1956 年夏季，在美国达特茅斯学院召开的小型研讨会上确立了 artificial intelligence 一词，并开创了 AI 研究领域。该会议的组织者是 John McCarthy，参加者包括 Claude Shannon, Marvin Minsky, Arthur Samuel, Trenchard Moore, Ray Solomonoff, Oliver Selfridge, Allen Newell 以及 Herbert Simon。

- 1957 年，康奈尔大学的心理学和计算机科学家 Frank Rosenblatt 提出了神经网络模型 Perceptron。1958 年，他造出了基于神经网络理论的计算机 Mark 1 Perceptron。

1.1. 人工智能的发展历程

1980 年代：机器学习的兴起

- Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088), 533-536.

  这篇文章提出了反向传播算法，克服了多层神经网络模型训练中的诸多难题。此文为深度学习的兴起奠定了基础。

- Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Elsevier.

  Judea Pearl 在此书中提出了贝叶斯网络，随后发展为因果推断理论，是现代 AI 推理的理论基础。

- 1995 年，Stuart Russell 和 Peter Norvig 出版了第一版 *Artificial Intelligence: A Modern Approach*。这本书后来成为使用率最高的 AI 教科书。

1.1. 人工智能的发展历程

- 1997 年，IBM 的超级计算机 Deep Blue 第一次击败了人类国际象棋世界冠军 Garry Kasparov。

2000 年代：深度学习的时代

- Hinton, G. E. (2007). Learning multiple layers of representation. *Trends in Cognitive Sciences*, 11(10), 428-434.

  此篇文章提出了更加高效的多层神经网络模型的训练方法，同时也宣告了深度学习时代的来临。

- Raina, R., Madhavan, A., & Ng, A. Y. (2009). Large-scale deep unsupervised learning using graphics processors. *Proceedings of the 26th Annual International Conference on Machine Learning*, 873-880.

  展示了 GPU 在训练深度学习模型上优于传统的多核 CPU。

1.1. 人工智能的发展历程

- 2011 年，IBM 的超级计算机 Watson 在电视智力竞赛节目 Jeopardy! 中击败了人类选手 Ken Jennings 和 Brad Rutter。AI 首次在自然语言处理方面超过人类。
- 2016 年，DeepMind 的 AlphaGo 击败围棋世界冠军李世石。
- 2017 年，Google 团队发表论文 Attention is All You Need，提出了用于自然语言处理的神经网络模型 Transformer，后来成为大多数大语言模型的底层机制。例如 GPT = Generative Pre-trained Transformer。

2020 年代：生成式人工智能爆发

- 2020 年，OpenAI 推出了大语言模型 GPT-3，DeepMind 推出了用于预测蛋白质结构的 AlphaFold 2，Waymo 开始运营自动驾驶出租车业务。
- 2022 年，OpenAI 推出了 ChatGPT。此后生成式人工智能应用呈现爆发式增长。

1.2. 什么是人工智能

大英百科全书将 AI 定义为机器执行智能任务的能力。这里的机器指电子计算机或者由计算机控制的机器人。智能任务指通常只能由智能生物（例如人类）完成的任务，例如学习、推理、解题、感知、使用语言。

John McCarthy 认为 AI 是制造智能机器（特别是智能程序）的科学和工程。他所谓的智能是在世界上完成各种目标所需要的计算能力。

NASA 对 AI 的简略定义是能够完成通常只能由人类完成的复杂任务的计算机系统。

学术界自始至终都没有在 AI 的定义上达成共识，就像我们很难准确定义什么是哲学一样。同学们可以参考 Stanford Encyclopedia of Philosophy 中的 Artificial Intelligence 词条 [点击打开网页] 了解更多这方面的讨论。

1.3. 机器学习

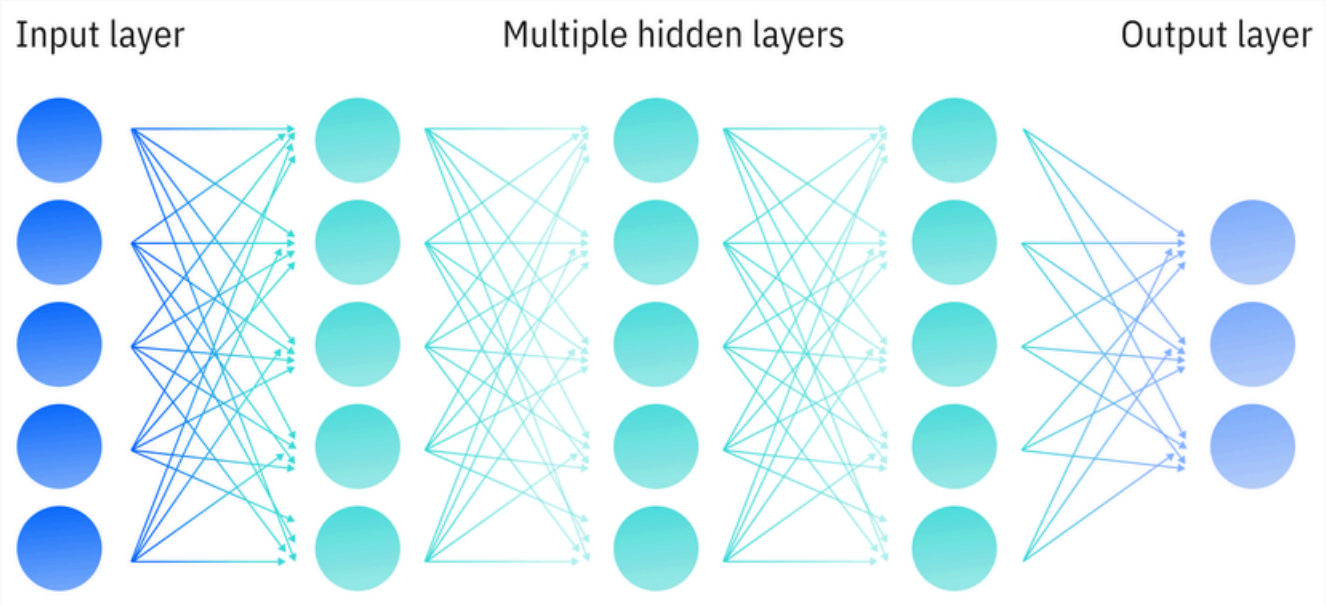
机器学习（machine learning）指通过发现数据中的特征而进行决策的算法。机器学习包含很多不同的方法，例如线性回归、逻辑回归、决策树、随机森林、支持向量机、k-近邻算法、聚类分析、神经网络等。

在众多方法中，**人工神经网络（artificial neural network，简称神经网络）**可以说是最流行的一个。神经网络算法的基本原理是模仿人类大脑的工作原理，通过众多节点（模仿神经元）构成的多层网络完成对数据特征的认知和决策。

机器学习方法主要可以分为**监督学习（supervised learning）**、**无监督学习（unsupervised learning）**和**强化学习（reinforcement learning）**。监督学习是在包含正确答案的训练数据中寻找变量间的关系，例如回归分析。无监督学习是在不包含正确答案的训练数据中寻找规律，例如聚类分析。强化学习是通过不断试错寻找合适的决策方法，例如棋牌类 AI。

1.4. 深度学习

深度学习（deep learning）是利用多层神经网络进行机器学习的方法的统称。深度学习方法在输入层和输出层以外，还包含至少三层隐藏层。GPT-4 中的神经网络模型共有 120 层[1]。



多层神经网络示意图，取自 www.ibm.com

[1] https://seifeur.com/gpt-4-layer-architecture/

1.4. 深度学习

Bilibili.com 上的 3Blue1Brown 账号中有一组关于深度学习的系列视频，非常直观的介绍了神经网络、back-propagation（反向传播法）、Transformer 等核心概念，感兴趣的同学可以自行观看。

一：神经网络的结构 👉 https://www.bilibili.com/video/BV1bx411M7Zx/

二：梯度下降法 👉 https://www.bilibili.com/video/BV1Ux411j7ri/

三：反向传播算法 👉 https://www.bilibili.com/video/BV16x411V7Qg/

四：大语言模型概述 👉 https://www.bilibili.com/video/BV1xmA2eMEFF/

五：Transformer 👉 https://www.bilibili.com/video/BV13z421U7cs/

六：注意力机制 👉 https://www.bilibili.com/video/BV1TZ421j7Ke/

七：大语言模型如何储存事实 👉 https://www.bilibili.com/video/BV1aTxMehEjK/

1.5. AI 的应用与风险

AI 的应用领域（不包含生成式 AI）：

- 智能驾驶（自动驾驶）：Waymo、萝卜快跑等
- 人形机器人：波士顿动力、宇树科技
- 自动规划与控制：NASA
- 机器翻译：Google Translate、DeepL
- 语音识别：Siri、Cortana 等
- 智能推荐：Amazon、Facebook、淘宝等
- 自动游戏：Deep Blue、AlphaGo、AlphaZero
- 图像识别：Google Cloud Vision、Amazon Rekognition 等
- 生物医药：AlphaFold
- 气象科学

AI 的风险：

- 用于决策时的偏误问题
- 对就业的冲击
- 安全并非第一考量
- AI 投资泡沫
- 对教育的影响

2. 机器学习和人工智能预测方法

2.1. Facebook 的 Prophet 模型

Facebook 的研究员 Sean Taylor 和 Benjamin Letham 在 2017 年发表了可用于大规模预测的时间序列模型 **Prophet**。他们对大规模（at scale）给出了独特的解读：1. 模型应该可以被大多数未受过专业训练的人使用；2. 模型应该适用于各式各样的预测问题；3. 模型应适用于大数据。

他们的核心贡献之一是完成了预测过程的自动化，但这并不罕见，我们学过的很多方法也多能实现一定程度的自动化。

Prophet 模型将时间序列分成四个成分，即

$$y_t = g(t) + s(t) + h(t) + \varepsilon_t$$

其中 $g(t)$ 是趋势项、$s(t)$ 是周期项、$h(t)$ 代表节假日虚拟变量。

Taylor, S. J., & Letham, B. (2018). Forecasting at Scale. *The American Statistician*, 72(1), 37-45.

2.1. Facebook 的 Prophet 模型

Prophet 模型的具体设定

- 趋势项：采用分段线性模型。Prophet 可以自动判断趋势变化的时间节点，在相邻的两个节点间采用线性趋势。
- 季节项：采用傅里叶级数模型，即一组不同频率的正弦和余弦函数之和。法国数学家傅里叶发现这种级数可以近似到任意一个（复杂的）周期函数。这里需要估计的参数是每个三角函数的系数。
- 节假日项：采用单纯的节假日虚拟变量。

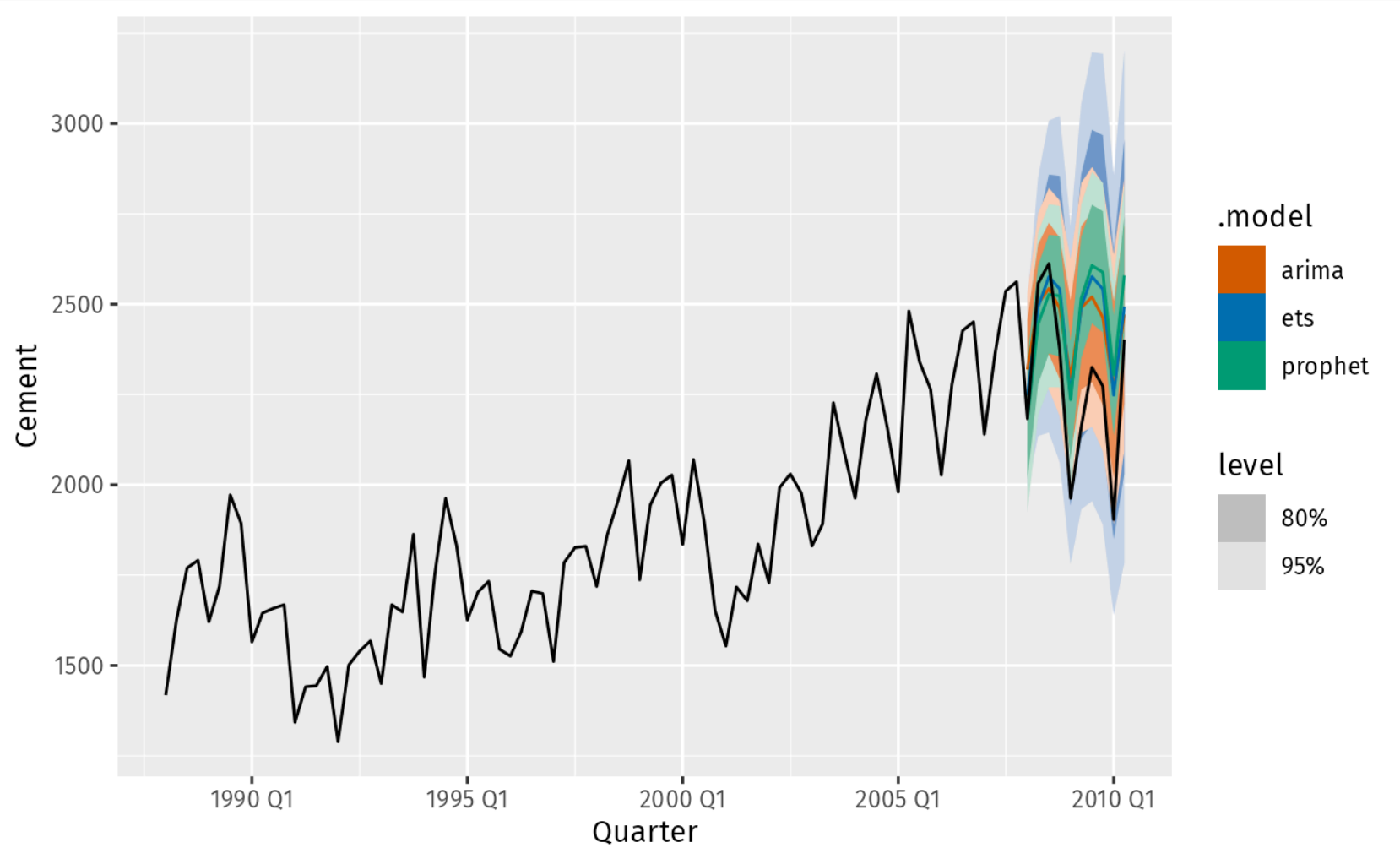
模型拟合采用贝叶斯方法，这可以自动完成模型选择（包括趋势项的变化节点和其他设定）。

综上可见，Prophet 模型和本门课程中重点介绍的其他模型在方法论上并没有显著区别（没有使用诸如神经网络之类的机器学习方法）。

2.1. Facebook 的 Prophet 模型

```
library(fable.prophet)
cement <- aus_production |>
  filter(year(Quarter) >= 1988)
train <- cement |>
  filter(year(Quarter) <= 2007)
fit <- train |>
  model(
    arima = ARIMA(Cement),
    ets = ETS(Cement),
    prophet = prophet(Cement ~ season(period = 4, order = 2,
                                      type = "multiplicative"))
)
fc <- fit |> forecast(h = "2 years 6 months")
fc |> autoplot(cement)
```

2.1. Facebook 的 Prophet 模型



2.1. Facebook 的 Prophet 模型

```
fc |> accuracy(cement)
```

```
#> # A tibble: 3 × 10
#>   .model   .type     ME   RMSE    MAE    MPE  MAPE  MASE RMSSE  ACF1
#>   <chr>    <chr>  <dbl>  <dbl>  <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl>
#> 1 arima    Test   -161.   216.   186. -7.71   8.68  1.27  1.26 0.387
#> 2 ets      Test   -171.   222.   191. -8.07   8.85  1.30  1.29 0.579
#> 3 prophet  Test   -176.   248.   215. -8.36   9.89  1.47  1.44 0.698
```

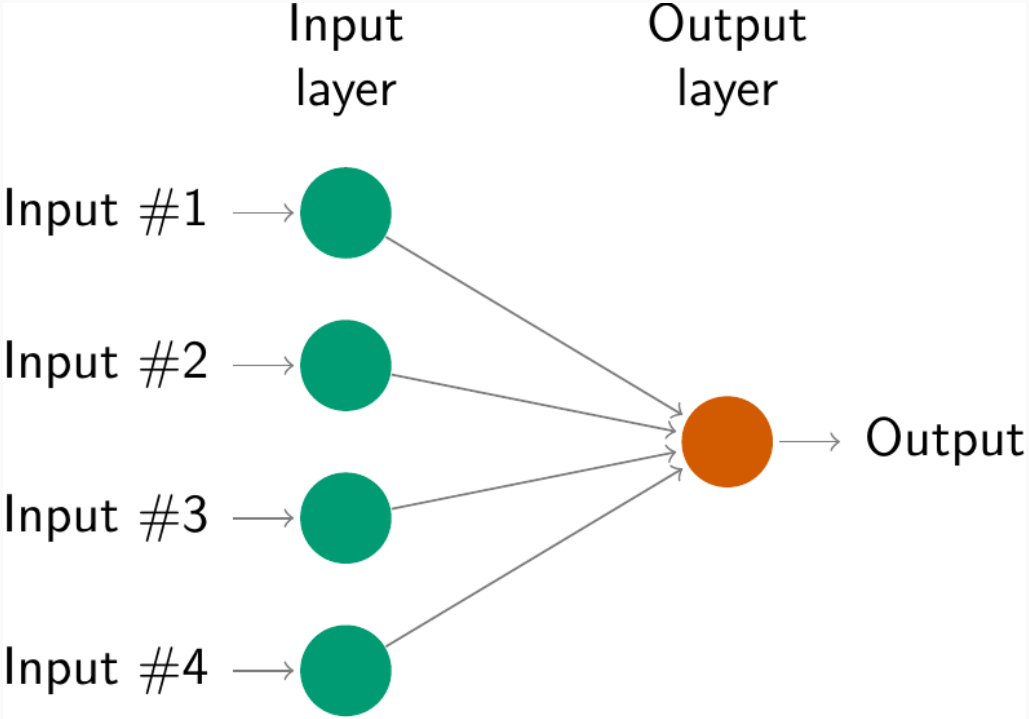
在这个例子中，Prophet 模型的表现不如 ARIMA 和 ETS，部分原因可能是因为数据是季度而非日度。一般来说，Prophet 的优势在于完全自动化以及拥有很快的计算速度，它可能更适用于日度商业数据。

2.2. 神经网络自回归模型

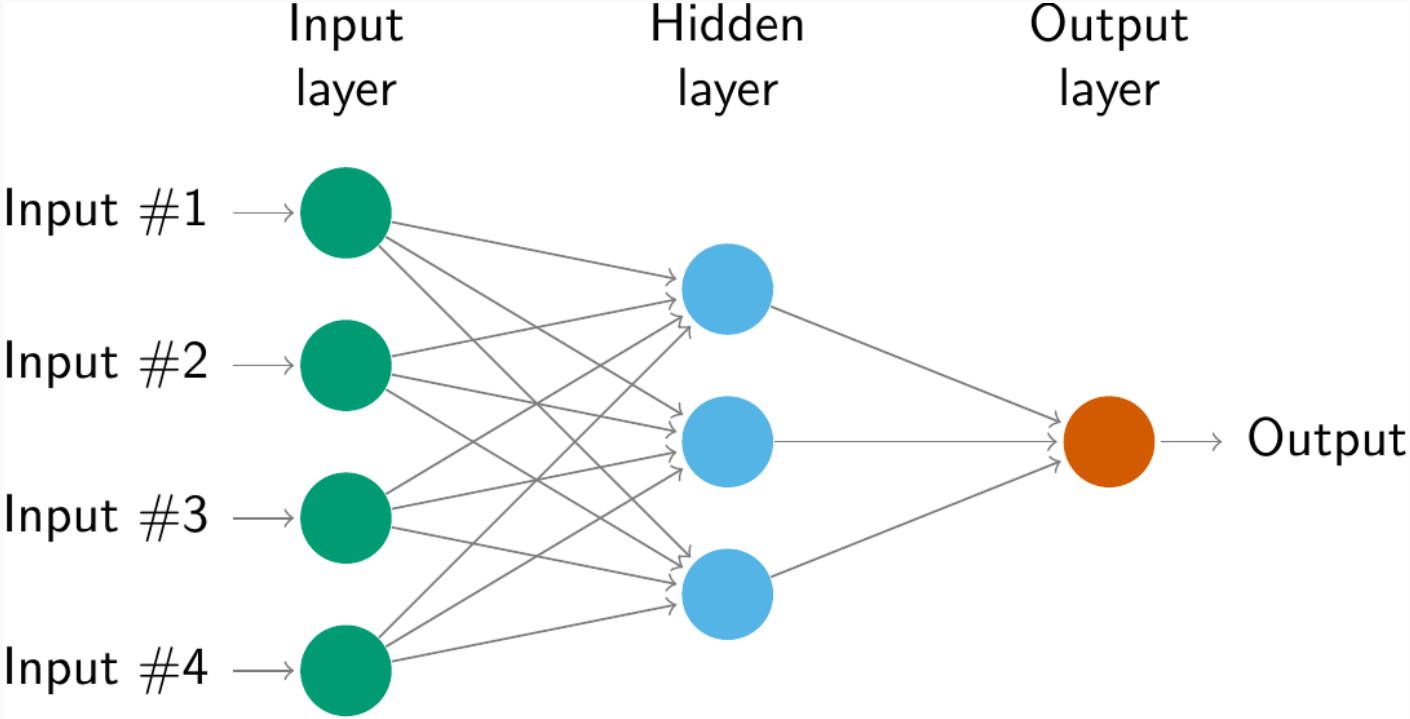
右图是一个最简单的神经网络，它仅包含一个输入层和一个输出层。输入层中每个节点（也叫神经元 neuron）代表一个变量，而输出层仅包含一个节点，它将所有输入变量的值进行加权平均，再经过某种函数变换后进行输出。

神经网络模型的目的是找到令输出值和数据中的观测值最接近的权重。

如果输出函数是 $f(z) = z$，那这个简单神经网络模型就等同于拥有四个预测变量的线性回归模型。



2.2. 神经网络自回归模型



这是加入了一个隐藏层的神经网络。隐藏层中的每个节点既是前一层的输出，又是后一层的输入。一个神经网络需要多少隐藏层，以及每一层需要多少节点是由分析者决定的。

2.2. 神经网络自回归模型

如果把 y_t 的滞后项作为输入变量加入神经网络，就得到了神经网络自回归模型（neural network autoregression, NNAR）。

我们只利用包含一个隐藏层的 NNAR 模型，写作 $\mathrm{NNAR}(p,k)$，p 代表滞后项的个数，k 代表隐藏层中的节点数。因此 $\mathrm{NNAR}(p,0) = \mathrm{ARIMA}(p,0,0)$。

对于季节性数据，我们可以添加 P 个季节滞后项，此时模型写作 $\mathrm{NNAR}(p,P,k)_m$，即包含 $y_{t-1},\dots,y_{t-p},y_{t-m},y_{t-2m},\dots,y_{t-Pm}$ 作为输入，m 为季节性周期。因此 $\mathrm{NNAR}(p,P,0)_m = \mathrm{ARIMA}(p,0,0)(P,0,0)_m$。

可以用 **`NNETAR()`** 函数对 $\mathrm{NNAR}(p,P,k)_m$ 模型进行拟合。如果不指定参数值，**`NNETAR()`** 也可以自动进行选择。

2.2. 神经网络自回归模型

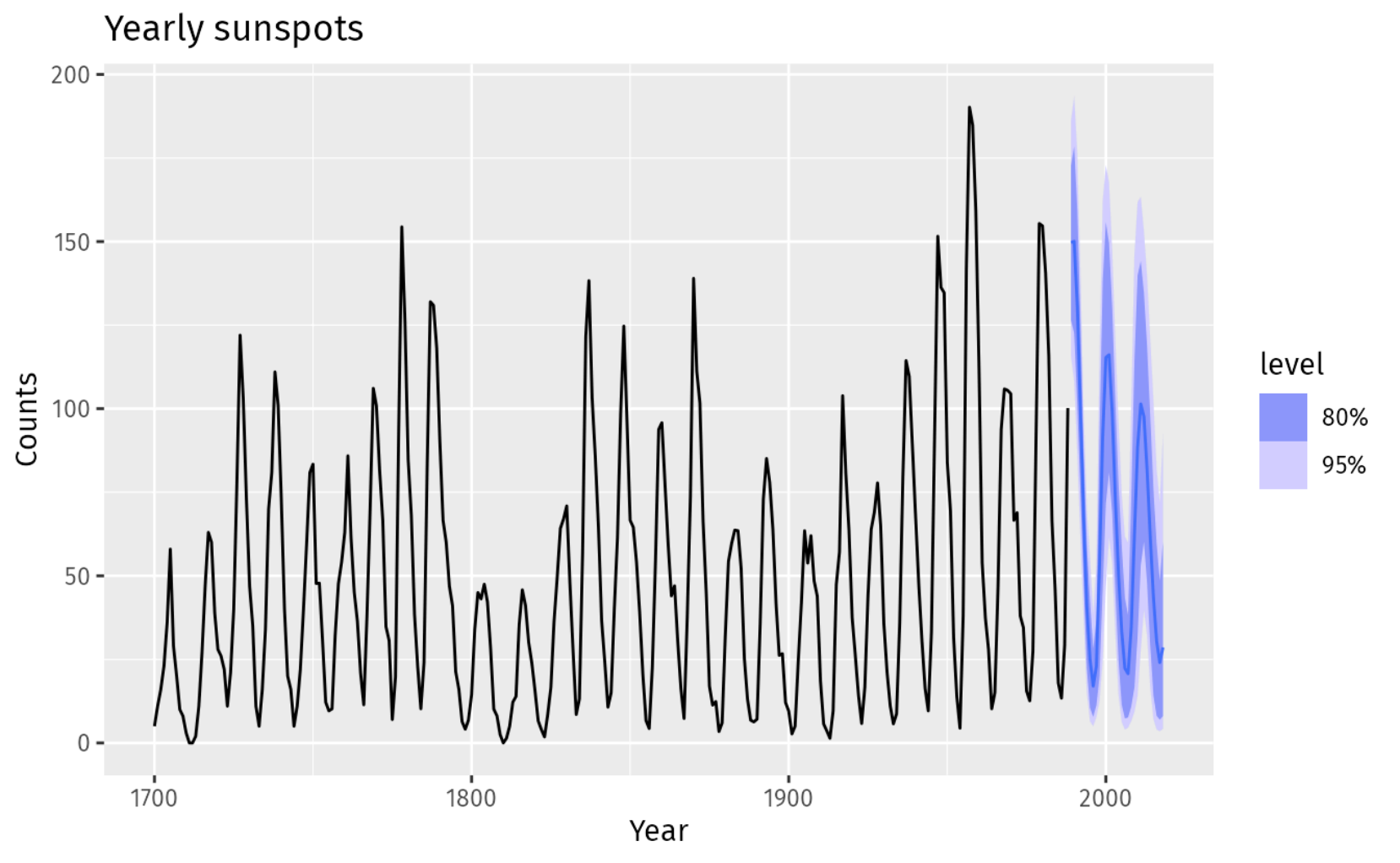
对太阳黑子数据应用 NNAR：

```
sunspots <- sunspot.year |> as_tsibble()
fit <- sunspots |>
  model(NNETAR(sqrt(value)))
fit |>
  forecast(h = 30) |>
  autoplot(sunspots) +
  labs(x = "Year", y = "Counts", title = "Yearly sunspots")
```

`NNETAR()` 函数选择了 $\mathrm{NNAR}(9,5)$，即滞后九阶输入加上五个节点的隐藏层。

在预测时，`NNETAR()` 需要利用仿真方法生成预测区间，因此相对于其他方法，此方法比较费时，需要耐心等待（几十秒至几分钟不等）。

2.2. 神经网络自回归模型



2.3. 时间序列基础模型

每一个生成式人工智能应用中都包含了**基础模型（foundation model）**，即基于超大规模数据集进行预训练的通用深度学习模型，并在此基础上进一步进行微调。例如 ChatGPT 的基础模型就是 GPT。

在时间序列预测领域，基础模型意味着以大量、多领域数据为基础进行预训练的通用模型。它们并不是针对某一特殊问题进行训练，因此具备高通用性，但也容易留下准确性差的印象。在进行预测时，即使预测对象没有被包含在训练集中，基础模型通常也不需要再次训练，这被称为零样本学习（zero-shot learning）。

比较著名的时间序列基础模型包括（但不限于）：

- TimeGPT (Nixtla)
- Tiny Time Mixer (IBM)
- Moirai (Salesforce)
- TimesFM (Google)
- Chronos (Amazon)

2.3. 时间序列基础模型

这里我们展示如何用 Nixtla 的 TimeGPT 进行预测。

利用基础模型时通常都要使用 API 调用保存在服务器上的程序。具体流程如下：

1. 访问 https://www.nixtla.io/，点击网页右上角的 free trial 注册账号。也可以用 GitHub 账号直接登录（跳到第二步）。免费试用期为一个月，一个月后需要付费升级才能继续使用。
2. 访问 https://www.nixtla.io/dashboard，登录后获取 API key。
3. 安装 R 程序包 `nixtlar`，调用并设定 API key 后即可使用 TimeGPT 的功能。

```
library(nixtlar)
nixtla_client_setup(api_key = "xxxxxxx")
  # 将你自己的 API key 复制到 xxxxxxx 处。
```

2.3. 时间序列基础模型

下面代码源自 https://nixtla.github.io/nixtlar/articles/get-started.html

这里使用的数据是 `nixtlar` 包中包含的 `electricity` 数据集。该数据集包含了欧洲和北美五个电力交易市场的每小时电力成交价数据，但每个市场的数据年份不同（观察区间长度基本相同）。

```
df <- electricity
```

```
glimpse(df)
```

```
Rows: 8,400
Columns: 3
$ unique_id <chr> "BE", "BE", "BE", "BE", "BE", "BE", "BE", "BE", "BE"…
$ ds        <chr> "2016-10-22 00:00:00", "2016-10-22 01:00:00", "2016-…
$ y         <dbl> 70.00, 37.10, 37.10, 44.75, 37.10, 35.61, 34.55, 50.…
```

注意：此数据保存为普通的数据框形式，时间信息也以字符形式保存在 `ds` 中。

2.3. 时间序列基础模型

预测仅需一行代码：

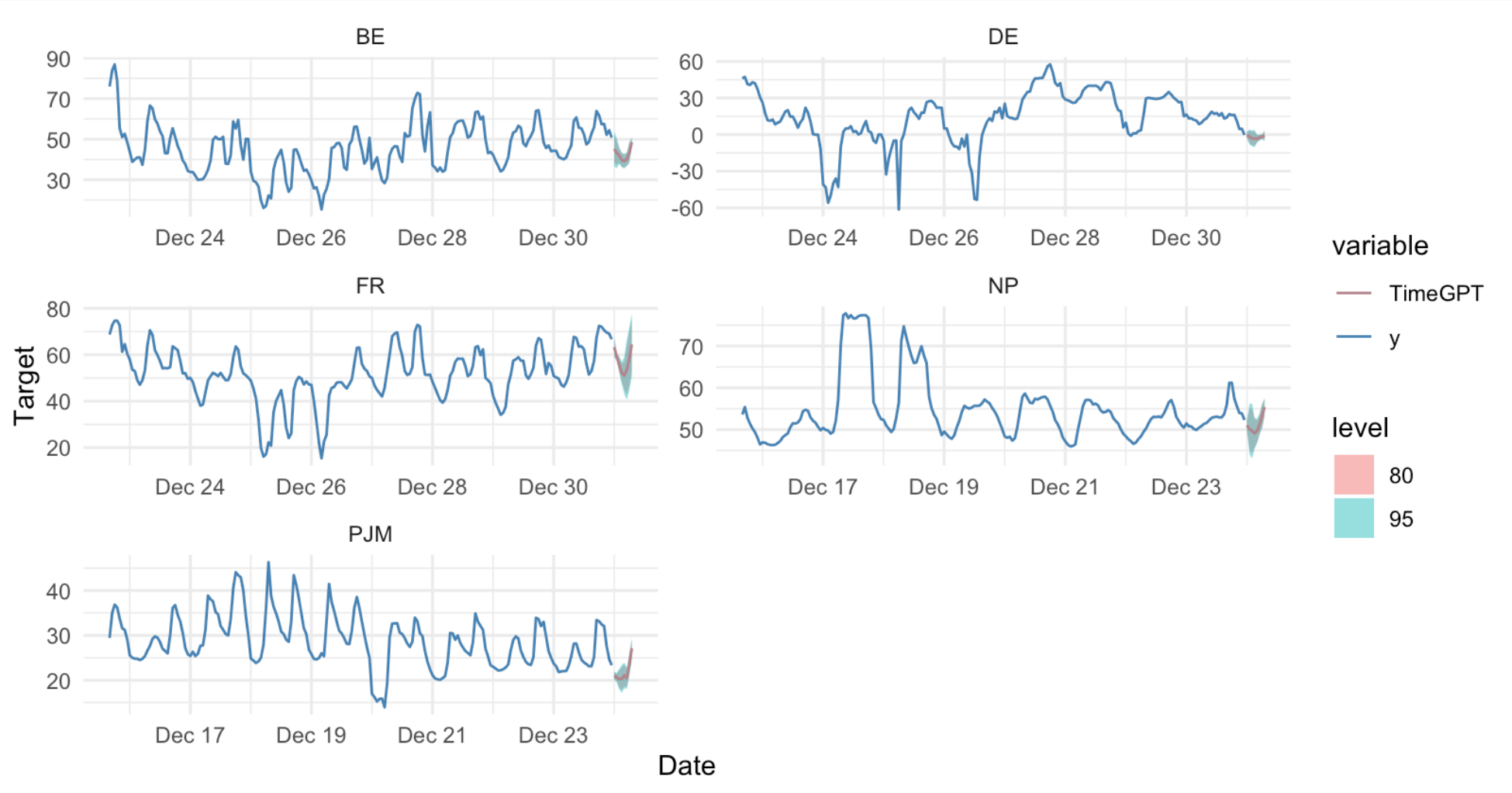
```
nixtla_client_fcst <- nixtla_client_forecast(df, h = 8, level =
c(80,95))
```

`nixtlar` 包还自带绘图函数：

```
nixtla_client_plot(df, nixtla_client_fcst, max_insample_length = 200)
```

这里仅绘制了历史数据中的最后 200 个观测值。

2.3. 时间序列基础模型


BE
DE
FR
NP
PJM
Target
Date
Dec 24
Dec 26
Dec 28
Dec 30
Dec 17
Dec 19
Dec 21
Dec 23
variable
TimeGPT
y
level
80
95

2.3. 时间序列基础模型

机器学习领域的常用编程语言是 python，多数模型都只能在 python 中实现。我们用的 **`nixtlar`** 包也是由 python 的程序包 **`nixtla`** 改编而来的。

大概是由于使用人数较少的缘故，**`nixtlar`** 包的维护并不好，版本号也较 python 的 **`nixtla`** 低。在近几天（2026 年 6 月 14-16 日）的尝试中，通过 R 调用 API 一直失败，但通过 python 却能成功，问题出在哪里不得而知。

如果同学们想尝试时间序列基础模型，最好学习一下 python 的基础用法，然后参照各个模型的说明文档进行操作。这超出了本门课程的覆盖范围，因此不再赘述。

扩展阅读

Bringsjord, S., & Govindarajulu, N.S. Artificial Intelligence. *The Stanford Encyclopedia of Philosophy*. https://plato.stanford.edu/entries/artificial-intelligence/

Caballar, R. D., & Stryker, C. What are foundation models? https://www.ibm.com/think/topics/foundation-models

Glassner, A. (2021). *Deep Learning: A Visual Approach*. No Starch Press.

Kottapalli, S. R. K., Hubli, K., Chandrashekhara, S., Jain, G., Hubli, S., Botla, G., & Doddaiah, R. (2025). Foundation Models for Time Series: A Survey (arXiv:2504.04011). arXiv. https://doi.org/10.48550/arXiv.2504.04011

Mucci, T. The history of AI. https://www.ibm.com/think/topics/history-of-artificial-intelligence

Nixtla Documentation, https://www.nixtla.io/docs

Russell, S., & Norvig, P. (2022). *Aritifical Intelligence: A Modern Approach*, 4th Edition. Pearson.

Stryker, C., & Kavlakoglu, E. What is artificial intelligence (AI)? https://www.ibm.com/think/topics/artificial-intelligence