

18 序列模型

概要

➤ 序列模型

- 马尔可夫 (Markov)
- 隐马尔可夫模型 (HMM)

内容提要

- 什么是语言模型？
 - 核心任务、关键应用与评估标准
- 经典基石：N-gram 统计语言模型
 - 马尔可夫假设：一种务实的简化。
 - 构建与训练：简单有效的“数数”
- N-gram 模型的根本性危机
 - 数据稀疏、维度灾难与泛化无能
 - 为什么统计方法会走到尽头？

相关的随机变量

为什么序列模型是必需的？

- 传统机器学习的基石：i.i.d. 假设
 - 独立同分布 (Independently and Identically Distributed)
 - 样本之间无关联，顺序可任意打乱
 - 例如：图像分类任务中，数据集中图像的顺序无关紧要
- 序列数据的颠覆性特质：顺序蕴含结构与意义
 - 文本：“狗咬人” vs. “人咬狗” —— 词汇相同，顺序定义了完全不同的语义
 - 时间序列：股票价格、气温变化 —— 过去的值直接影响未来的预测
 - 音频/视频：语音波形、电影帧 —— 信息的表达依赖于时序连续性
- 对于序列数据，顺序本身就是一种至关重要的信息。我们的模型必须有能力捕捉和利用这种顺序依赖性

序列建模的根本任务：估计联合概率分布

- 给定一个序列 $x_1, x_2, \dots, x_T$ ，我们的目标是建模其联合概率 $P(x_1, x_2, \dots, x_T)$
- 根据概率的链式法则，这个联合概率可以精确分解为：

$$P(x_1, \dots, x_T) = P(x_1) \prod_{t=2}^T P(x_t | x_1, \dots, x_{t-1})$$

- 自回归 (Autoregressive) 的形式：每一项的概率都以其所有历史为条件
 - 评估序列: 计算一个给定序列的似然度（例如，判断一句话是否通顺）
 - 预测/生成: 给定历史 $x_1, \dots, x_{t-1}$ ，预测下一个最可能的 x_t

核心挑战：不可控的历史与参数的诅咒

- 链式法则 $P(x_t|x_1, \dots, x_{t-1})$ 在理论上完美，但在实践中是灾难性的
- 挑战1：可变且无限的条件长度
 - 条件 $x_1, \dots, x_{t-1}$ 的长度 $t-1$ 随时间步变化
 - 当序列很长时，历史会变得无限长。如何用一个模型处理任意长度的输入？
- 挑战2：参数空间的指数级爆炸（维度灾难）
 - 假设我们处理的是离散符号（如单词），词汇表大小为 $|V|$
 - 要对 $P(x_t|x_1, \dots, x_{t-1})$ 建模，理论上需要为每一种可能的历史前缀都存储一个概率分布
 - 参数数量会随着历史长度 $t-1$ 呈 $|V|^{t-1}$ 级别增长，这在计算和统计上都是不可行的
- 面对无限的历史，我们能做的第一个、也是最简单的妥协是什么？

序列模型

统计工具

- 在时间 t 观察到 x_t , 那么得到 T 个不独立的随机变量

$$(x_1, \dots, x_T) \sim p(\mathbf{x})$$

- 使用条件概率展开

$$p(a, b) = p(a)p(b | a) = p(b)p(a | b)$$

统计工具

- 在时间 t 观察到 x_t , 那么得到 T 个不独立的随机变量 $(x_1, \dots, x_T) \sim p(\mathbf{x})$
 - 使用条件概率展开

$$p(a, b) = p(a)p(b | a) = p(b)p(a | b)$$

- 前序【推理】

$$p(x) = p(x_1) \cdot p(x_2 | x_1) \cdot p(x_3 | x_1, x_2) \cdot \dots p(x_T | x_1, \dots, x_{T-1})$$



- 反序【追溯原因】

$$p(x) = p(x_T) \cdot p(x_{T-1} | x_T) \cdot p(x_{T-2} | x_{T-1}, x_T) \cdot \dots p(x_1 | x_2, \dots, x_T)$$



- 因果关系?

- 阻止反方向, 模型的“错误”方向通常要复杂得多

序列模型

➤ $p(x) = p(x_1) \cdot p(x_2|x_1) \cdot p(x_3|x_1, x_2) \cdot \dots p(x_T|x_1, \dots x_{T-1})$



➤ 对条件概率建模

$$p(x_t|x_1, \dots x_{t-1}) = p(x_t|f(x_1, \dots x_{t-1}))$$

➤ 自回归模型

➤ 对见过的数据建模

计划A - 马尔可夫 (Markov) 假设 - 只记着最近发生的事

- 假设当前数据只和 τ 个过去数据相关

$$p(x_t | x_1, \dots, x_{t-1}) = p(x_t | x_{t-\tau}, \dots, x_{t-1}) = p(x_t | f(x_{t-\tau}, \dots, x_{t-1}))$$

- 称为 $(\tau - 1)$ 阶马尔可夫假设
- 可以在过去数据上训练一个MLP

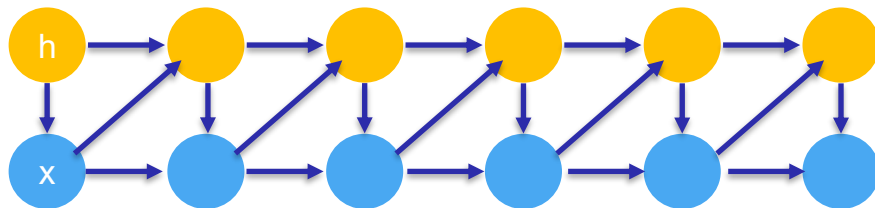
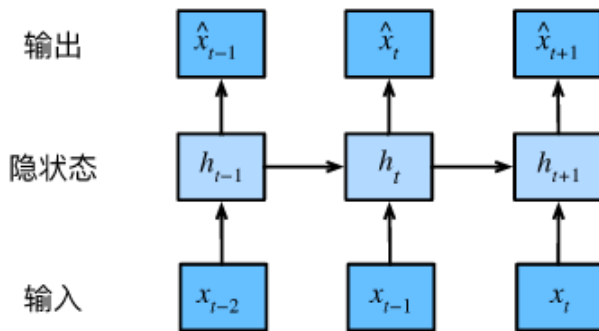
➤ 优点

- 极大地简化了问题。条件部分的长度从可变的 $t - 1$ 变成了固定的 $\tau - 1$
- 使得参数化和统计估计成为可能
- 这是一种有损压缩，通过“遗忘”遥远历史来换取模型的可行性。

计划B – 潜变量模型

➤ 引入潜变量 h_t 来表示过去信息 $h_t = f(x_1, \dots, x_{t-1})$

➤ 这样 $x_t = p(x_t | h_t)$



总结

- 时序模型中，当前数据跟之前观察到的数据相关
- 自回归模型使用自身过去数据来预测未来
- 马尔科夫模型假设当前只跟最近少数数据相关，从而简化模型
- 潜变量模型使用潜变量来概括历史信息

什么是语言模型

语言模型的核心任务：为文本序列分配概率

- 给定一个由 T 个词（或符号）组成的文本序列 $W = (w_1, w_2, \dots, w_T)$ ，语言模型 (LM) 的目标是计算这个序列出现的联合概率 $P(W)$
- 根据概率的链式法则，这个联合概率可以精确分解为

$$P(W) = P(w_1, \dots, w_T) = P(w_1) \prod_{t=2}^T P(w_t | w_1, \dots, w_{t-1})$$

- 本质：语言模型是一个自回归 (Autoregressive) 模型，它在每个时间步 t 预测下一个词 w_t 的概率分布，条件是所有已经出现的历史词序列 $w_1, \dots, w_{t-1}$

语言模型的广泛应用：无处不在的NLP基石

- 语言模型的能力（评估和生成文本）使其成为众多NLP任务的核心
- 评估序列似然度 (判断哪个更“自然”)
 - 机器翻译: $P(\text{"the cat is cute"}) > P(\text{"cute is the cat"})$
 - 语音识别: 区分同音异义词 $P(\text{"to recognize speech"}) > P(\text{"to wreck a nice beach"})$
 - 拼写纠错: $P(\text{"I am happy"}) > P(\text{"I ma happy"})$
- 生成文本 (Text Generation)
 - 给定一个前缀 (prompt)，通过反复采样下一个最可能的词来生成连贯的文本
 - 应用: 对话系统、故事生成、代码自动补全
 - 示例: Input: "The best thing about AI is" -> Output: "its ability to..."
- 作为预训练模型 (Pre-trained Models)
 - 在海量无标签文本上训练一个强大的语言模型，其学到的“语言知识”可以被迁移到各种下游任务（如分类、问答）
 - 这是现代NLP的基石，例如 BERT 和 GPT 系列模型

评估标准：困惑度 (Perplexity)

- 一个好的语言模型，应该对真实的、自然的测试文本给予高概率
- 在测试集 $W = (w_1, \dots, w_N)$ 上，困惑度 (Perplexity, PPL) 定义为测试集逆几何平均概率

$$\text{PPL}(W) = P(w_1, \dots, w_N)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1, \dots, w_N)}}$$

- 其中 N 是测试集的总词数
- 等价形式 (使用交叉熵)

$$\text{PPL}(W) = \exp\left(-\frac{1}{N} \sum_{i=1}^N \log P(w_i | w_1, \dots, w_{i-1})\right) = \exp(\text{Cross-Entropy})$$

➤ 解读

- 困惑度越低，模型性能越好
- 困惑度可以被直观地理解为“模型在预测下一个词时，平均有多少个等可能性的选择”
 - 如果 $\text{PPL}=100$ ，意味着模型在每个位置上，其不确定性等价于从100个词中均匀随机选择一个

基于马尔可夫假设的实践：N-gram 语言模型

➤ 将马尔可夫假设应用于语言，我们得到 N-gram 模型

➤ 一元语法 (Unigram, $n=1$): 词与词之间相互独立

$$P(x_1, \dots, x_T) \approx \prod_{t=1}^T P(x_t)$$

➤ 二元语法 (Bigram, $n=2$): 每个词只依赖于前一个词

$$P(x_1, \dots, x_T) \approx P(x_1) \prod_{t=2}^T P(x_t | x_{t-1})$$

➤ 三元语法 (Trigram, $n=3$): 每个词依赖于前两个词

$$P(x_1, \dots, x_T) \approx P(x_1, x_2) \prod_{t=3}^T P(x_t | x_{t-2}, x_{t-1})$$

➤ 参数估计：通过最大似然估计（即“数数”）来计算概率

$$P(w_t | w_{t-n+1}, \dots, w_{t-1}) = \frac{\text{Count}(w_{t-n+1}, \dots, w_{t-1}, w_t)}{\text{Count}(w_{t-n+1}, \dots, w_{t-1})}$$

N-gram 的根本缺陷 (1): 数据稀疏性与零概率问题

- 如果一个N-gram在训练语料中未出现过，其计数为0，导致概率估计为0
 - 如，在训练集中没见过 "to wreck a nice beach"，模型会认为这句话不可能出现。这显然是错误的
- 经典对策：平滑 (Smoothing)
 - 拉普拉斯平滑：给所有可能的 N-gram 计数加一个小的平滑值 α

$$P(w_t | \dots) = \frac{\text{Count}(\dots, w_t) + \alpha}{\text{Count}(\dots) + \alpha|V|}$$

- 问题：拉普拉斯平滑过于粗暴，它会从高频事件中“偷走”太多概率，分配给海量的未见事件。对于语言的长尾分布（Zipf's Law），效果很差
- 更复杂的平滑技术（如Kneser-Ney）效果更好，但无法根治问题

N-gram 的根本缺陷 (2): 无法摆脱的维度灾难

- 模型缺陷的根源: N-gram 模型的参数空间会随着 n 的增加而指数级增长
- 一个大小为 $|V|$ 的词汇表, N-gram 模型的参数数量级为 $O(|V|^n)$
- 假设 $|V| = 20000$ (一个中等大小的词汇表)
 - Bigram ($n = 2$): $20000^2 = 4 \times 10^8$
 - Trigram ($n = 3$): $20000^3 = 8 \times 10^{12}$
 - 4-gram ($n = 4$): $20000^4 = 1.6 \times 10^{17}$
- 后果:
 - 存储灾难: 无法存储如此庞大的参数表
 - 统计灾难: 即使能存储, 也绝无可能在有限数据上可靠地估计这些参数
- 被迫的妥协: 在实践中, 我们只能使用非常小的 n (通常 $n \leq 5$)。这意味着模型只能看到非常短的上下文, 无法捕捉句子中的长程依赖 (Long-term Dependencies)
 - 例: "The boy who lives in that big house on the hill ... is happy."
 - 主语 "boy" 和动词 "is" 之间的单复数一致性, 需要跨越很长的距离, 这是 N-gram 模型无法捕捉

N-gram 的根本缺陷 (3): 语义的缺失与泛化无能

- N-gram 模型将每个单词视为一个独立的、离散的符号 (one-hot vector)
- 它无法理解单词之间的语义相似性
 - 假设模型在语料中见过 "the cat is walking"
 - 计算 "the dog is walking" 的概率时, 无法利用 "cat" 和 "dog" 都是动物、都可作为主语这一语义信息
 - 在模型眼中, 历史 (the, cat) 和 (the, dog) 是两个完全不相关的、正交的实体。它们的知识无法共享
- 根本原因: 基于计数的离散表示法, 无法学习到一个平滑、泛化的语义空间。模型只能“记忆”见过的模式, 而不能“理解”和“泛化”到未见的、但语义相似的模式

文本预处理

Tokenization

➤ 将原始文本字符串转化为模型可以处理的整数ID序列

➤ 策略对比

➤ 词级 (Word-level): 简单, 但词汇表爆炸, 且无法处理未登录词 (OOV)

➤ 字符级 (Character-level): 无OOV问题, 但序列过长, 模型学习效率低

➤ 子词级 (Subword-level): 黄金标准

➤ 算法: BPE, WordPiece, SentencePiece

➤ 原理: 通过数据驱动的方式, 将文本切分为最优的子词单元组合。高频词是单个token, 低频词和未见词由多个子词token组合表示

➤ 优势: 在词汇表大小、序列长度和OOV处理之间取得了最佳平衡

读取长序列数据

- 整个语料库（如一本书）太大，无法一次性载入内存
- 解决方案： 将长序列切分为固定长度的小批次 (mini-batches)
 - 随机采样 (Random Sampling)
 - 每次从语料库中随机抽取一段序列
 - 适用场景： 无状态模型（如前馈神经LM）， 批次间无需传递信息
 - 顺序分区 (Sequential Partitioning)
 - 将整个语料库按顺序切分。后一个批次的数据在原文中紧接着前一个批次
 - 适用场景： 有状态模型（如RNN）， 需要在批次间传递“记忆”（隐状态）
 - 这是训练高性能RNN语言模型的标准做法

The Time Machine by H. G. Wells
The Time Machine by H. G. Wells
The Time Machine by H. G. Wells
The Time Machine by H. G. Wells
The Time Machine by H. G. Wells
The Time Machine by H. G. Wells

总结

- 定义了语言模型的核心任务（为序列分配概率）及其重要应用和评估标准（困惑度）
- 深入剖析了经典的N-gram模型，理解了其基于马尔可夫假设的构建方式。
- 揭示了N-gram模型面临的三大根本危机：数据稀疏、维度灾难和泛化无能，论证了其局限性
- 引入神经语言模型的思想，特别是分布式表示（词嵌入）作为解决泛化问题的关键
- 一个更强大模型的诉求——处理任意长度上下文，从而自然地引出了下一章的主题：循环神经网络 (RNN)

Thanks