Deep Learning I

10. 循环神经网络

WU Xiaokun 吴晓堃

xkun.wu [at] gmail

2022/04/29

序列模型

数据、结构

目前遇到的数据：表格、图像

- 图像数据：二维空间结构假设，用于**简化计算**

数据、结构

目前遇到的数据：表格、图像

- 图像数据：二维空间结构假设，用于**简化计算**

序列化数据：依赖某种**序关系**

- 文本：字（符）的先后顺序
- 视频、音频：时间顺序

数据、结构

目前遇到的数据：表格、图像

- 图像数据：二维空间结构假设，用于**简化计算**

序列化数据：依赖某种**序关系**

- 文本：字（符）的先后顺序
- 视频、音频：时间顺序

序关系假设：先验知识，**预测**序列的后续

- 文本生成
- 股价、天气预测

序列数据：难点

季节性统计规律：年后、毕业季的就业率，“新增高素质人才”

序列数据：难点

季节性统计规律：年后、毕业季的就业率，“新增高素质人才”

预测明天比预测过去更难：特别是股价，先见之明比事后解释难得多

- **外推法**、内插法

序列数据：难点

季节性统计规律：年后、毕业季的就业率，“新增高素质人才”

预测明天比预测过去更难：特别是股价，先见之明比事后解释难得多

- **外推法**、内插法

顺序重排导致意义丢失：文本、音乐、视频、语音

序列数据：难点

季节性统计规律：年后、毕业季的就业率，“新增高素质人才”

预测明天比预测过去更难：特别是股价，先见之明比事后解释难得多

- **外推法**、内插法

顺序重排导致意义丢失：文本、音乐、视频、语音

情感分析：例如影评，大众对电影的看法随时间推移而变化

情感分析：心理学提示

锚定效应：基于他人的意见做评价，“人云亦云”

- 电影得奖后关注度、评分骤升，持续到大众忘记这个奖项

情感分析：心理学提示

锚定效应：基于他人的意见做评价，“人云亦云”

- 电影得奖后关注度、评分骤升，持续到大众忘记这个奖项

享乐效应：适应较高（低）水平的电影并认为是常态，“拉高（低）了审美”

- 看过大量精彩电影之后，对其他电影的评分降低

情感分析：心理学提示

锚定效应：基于他人的意见做评价，“人云亦云”

- 电影得奖后关注度、评分骤升，持续到大众忘记这个奖项

享乐效应：适应较高（低）水平的电影并认为是常态，“拉高（低）了审美”

- 看过大量精彩电影之后，对其他电影的评分降低

其他相关因素

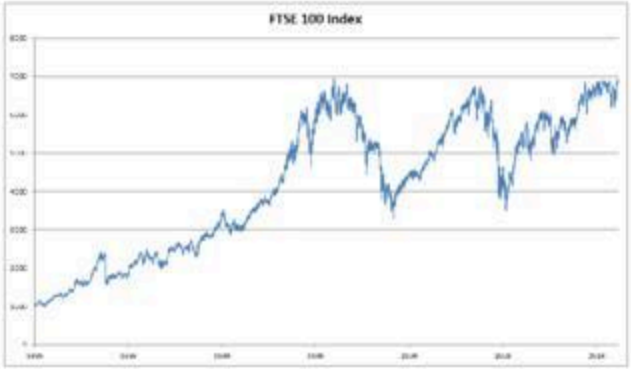
- 季节性：很少观众在暑假看贺岁片
- 道德高地：导演、演员的不当行为导致评分稍低

统计建模

将观测看成随机变量：时间t观测到x_t

- **序列化随机变量**的分布：$x_1, .., x_T \sim p(x_1, .., x_T) \equiv p(\mathbf{x})$

注意：序列化随机变量一般**不是相互独立**的；此处省略了模型参数

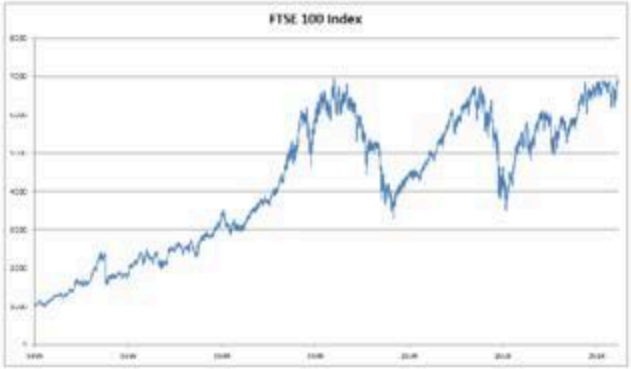


统计建模

将观测看成随机变量：时间t观测到x_t

- **序列化随机变量**的分布：$x_1, .., x_T \sim p(x_1, .., x_T) \equiv p(\mathbf{x})$

注意：序列化随机变量一般**不是相互独立**的；此处省略了模型参数



外推预测：$x_t \sim p(x_t|x_{t-1}, .., x_1) \equiv p(x_t|x_{1:t-1})$

统计建模：联合、条件概率

联合、条件概率公式

$$
\begin{aligned}
P(x_1, x_2, .., x_T) &= P(x_1)P(x_2|x_1)P(x_3|x_{1:2})...P(x_T|x_{1:T-1}) \\
&= \prod_{k=1}^{T} P(x_k|x_{1:k-1})
\end{aligned}
$$

统计建模：联合、条件概率

联合、条件概率公式

$$P(x_1, x_2, .., x_T) = P(x_1)P(x_2|x_1)P(x_3|x_{1:2})...P(x_T|x_{1:T-1})$$
$$= \prod_{k=1}^{T} P(x_k|x_{1:k-1})$$

逆序也可计算，但物理上不一定能够解释

$$P(x_1, x_2, .., x_T) = P(x_T)P(x_{T-1}|x_T)P(x_{T-2}|x_{T-1:T})...P(x_1|x_{2:T})$$

自回归模型

外推预测：$x_t \sim p(x_t|x_{1:t-1})$

- **自回归模型**：对见过的数据建模、对自己执行回归，即自监督学习

$$x_t \sim p(x_t|x_{1:t-1}) \equiv p(x_t|f(x_{1:t-1}))$$

自回归模型

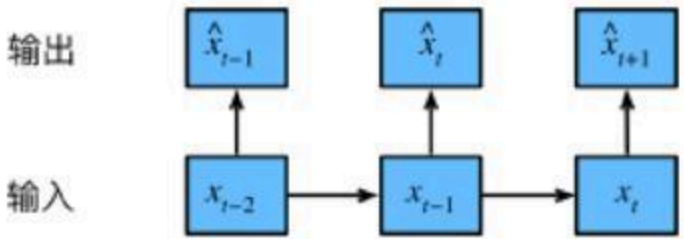
外推预测：$x_t \sim p(x_t|x_{1:t-1})$

- **自回归模型**：对见过的数据建模、对自己执行回归，即自监督学习

$$x_t \sim p(x_t|x_{1:t-1}) \equiv p(x_t|f(x_{1:t-1}))$$

Markov 假设：当前预测只取决于之前**最近的τ个观测**

- $\hat{x}_t \sim p(x_t|x_{t-\tau:t-1}) \equiv p(x_t|f(x_{t-\tau:t-1}))$

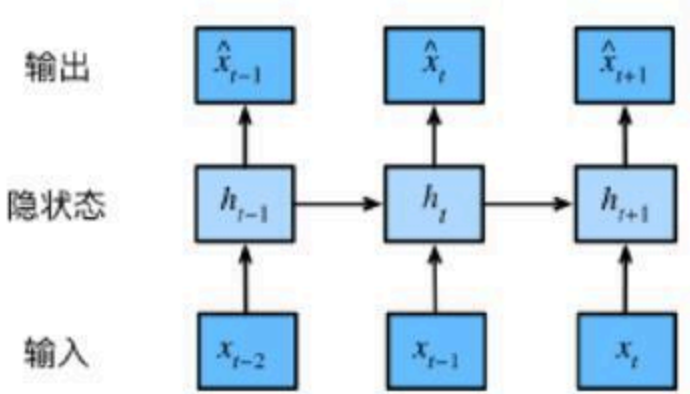


隐变量自回归模型

外推预测：$x_t \sim p(x_t|x_{1:t-1})$

- **隐变量自回归模型**：当前隐状态整合过去所有的观测信息

$$\hat{x}_t = p(x_t|h_t)$$
$$h_t = g(x_{t-1}, h_{t-1})$$



实验：序列模型

小结：序列模型

- 序列模型：当前预测与之前的观测数据相关
  - Markov假设：当前预测只取决于之前的最近几个观测数据
- 自回归模型：使用自身历史数据建模
- 隐变量模型：使用隐状态整合过去所有的观测信息

文本预处理

常见预处理步骤

- 将文本作为**字符串**加载到内存中。
- 将字符串拆分为**词元**（如单词、字符）。
- 建立一个**词表**，将拆分的**词元映射**到数字索引。
- 将文本转换为数字索引**（向量）序列**，方便模型操作。

实验：文本预处理

语言模型和数据集

语言模型

语言模型 language models (LMs)：对序列（中的词）赋予概率

- $P(W) = P(w_1, w_2, .., w_n)$
- $P(w_n|w_1, w_2, .., w_{n-1})$

有时也称为**语法、文法**

语言模型

语言模型 language models (LMs)：对序列（中的词）赋予概率

- $P(W) = P(w_1, w_2, .., w_n)$
- $P(w_n|w_1, w_2, .., w_{n-1})$

有时也称为**语法、文法**

N元语法 n-gram：连续N个词看作一个单元

- **二元语法 2-gram/bigram**：“your homework”，“我们”

N元语法：近似计算

本质是近似计算：

- **用最近的 N 个词替代**观察到的全部字符信息

例如**二元语法 bigram**：$P(w_n|w_{1:n-1}) \approx P(w_n|w_{n-1})$

N元语法：近似计算

本质是近似计算：

- **用最近的 N 个词替代**观察到的全部字符信息

例如**二元语法 bigram**：$P(w_n|w_{1:n-1}) \approx P(w_n|w_{n-1})$

$$
\begin{aligned}
P(\text{他下课后去了食堂}) &= P(\text{他下课后去了})P(\text{食堂}|\text{他下课后去了}) \\
&\approx P(\text{他下课后去了})P(\text{食堂}|\text{去了}) \\
&\approx P(\text{他})P(\text{下课后}|\text{他})P(\text{去了}|\text{下课后})P(\text{食堂}|\text{去了})
\end{aligned}
$$

N元语法：近似计算

本质是近似计算：

- **用最近的 N 个词替代**观察到的全部字符信息

例如**二元语法 bigram**：$P(w_n|w_{1:n-1}) \approx P(w_n|w_{n-1})$

$$
\begin{aligned}
P(\text{他下课后去了食堂}) &= P(\text{他下课后去了})P(\text{食堂}|\text{他下课后去了}) \\
&\approx P(\text{他下课后去了})P(\text{食堂}|\text{去了}) \\
&\approx P(\text{他})P(\text{下课后}|\text{他})P(\text{去了}|\text{下课后})P(\text{食堂}|\text{去了})
\end{aligned}
$$

Markov 假设：当前状态只取决于最近 N 个单元的信息

- **Markov 模型**：预测时不关注过于久远的历史信息

特殊情况

二元语法 bigram：“活在当下”

- 每个词只取决于前面的一个词

$$P(w_{1:n}) \approx \prod_{k=1}^{n} P(w_k|w_{k-1})$$

特殊情况

二元语法 bigram：“活在当下”

- 每个词只取决于前面的一个词

$$P(w_{1:n}) \approx \prod_{k=1}^{n} P(w_k|w_{k-1})$$

一元语法 unigram：随机行走

- 没有约束条件，每个词独立计算概率

$$P(w_{1:n}) \approx \prod_{k=1}^{n} P(w_k)$$

一般情况：N元语法

$$P(w_n|w_{1:n-1}) \approx P(w_n|w_{n-N+1:n-1})$$

$$P(w_{1:n}) \approx \prod_{k=1}^{n} P(w_k|w_{k-N+1})$$

一般情况：N元语法

$$P(w_n|w_{1:n-1}) \approx P(w_n|w_{n-N+1:n-1})$$

$$P(w_{1:n}) \approx \prod_{k=1}^{n} P(w_k|w_{k-N+1})$$

N元语法简单、易理解；但用于语言建模**通常不足**

- 语言中**长距离依赖**对语义理解很重要

一般情况：N元语法

$$P(w_n|w_{1:n-1}) \approx P(w_n|w_{n-N+1:n-1})$$

$$P(w_{1:n}) \approx \prod_{k=1}^{n} P(w_k|w_{k-N+1})$$

N元语法简单、易理解；但用于语言建模**通常不足**

- 语言中**长距离依赖**对语义理解很重要

最大似然估计

最大似然估计 Maximum Likelihood Estimation (MLE)

- 解决的问题：事件的可能性有多大?
  - 给定观测结果：统计出现的**频率**即可

最大似然估计

最大似然估计 Maximum Likelihood Estimation (MLE)

- 解决的问题：事件的可能性有多大?
  - 给定观测结果：统计出现的**频率**即可
- 从语料库统计N元词频，然后正则化为**可能性**

例如二元语法：

$$P(w_n|w_{n-1}) = \frac{\Gamma(w_{n-1}w_n)}{\sum_w \Gamma(w_{n-1}w)} = \frac{\Gamma(w_{n-1}w_n)}{\Gamma(w_{n-1})}$$

加一平滑

也称 **Laplace平滑**：灵感来源于（热）扩散过程

- （假装）每个词汇序列看到过比实际值多一次：所有词计数加一

加一平滑

也称 **Laplace平滑**：灵感来源于（热）扩散过程

- （假装）每个词汇序列看到过比实际值多一次：所有词计数加一

例如一元语法：$w_i \in w_{1:N}$，序列$W = w_{1:N}$有N个词

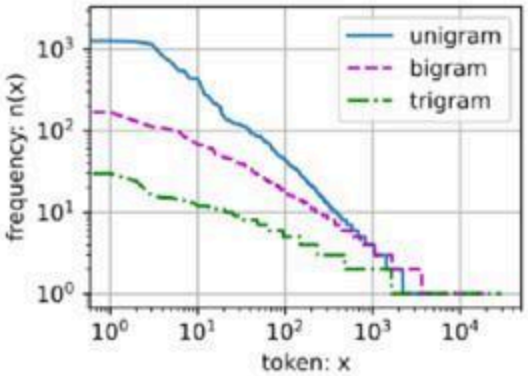
$$P_{MLE}(w_i) = \frac{c_i}{N}$$
$$P_{Add-1}(w_i) = \frac{c_i + 1}{N + |V|}$$

- 注意：分母需要对应增加总共$|V|$个词汇

N元模型词频比较

都遵循Zipf定律：$n_i \propto \frac{1}{i^\alpha}$

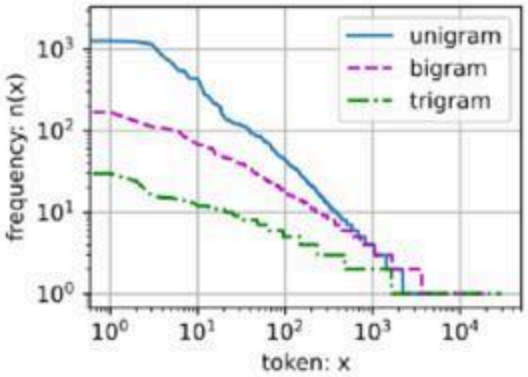
- **双对数空间**上近似直线：$\log n_i = -\alpha \log i + c$



N元模型词频比较

都遵循**Zipf定律**：$n_i \propto \frac{1}{i^\alpha}$

- **双对数空间**上近似直线：$\log n_i = -\alpha \log i + c$

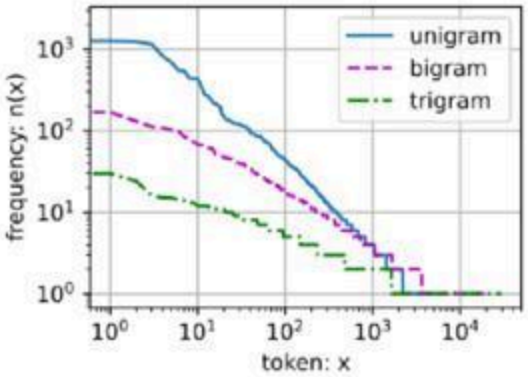


- 词表中N元组数量并不多，却可以表述非常复杂的语句
  - 说明语言中存在大量**结构**，可应用模型提取模式

N元模型词频比较

都遵循Zipf定律：$n_i \propto \frac{1}{i^\alpha}$

- **双对数空间**上近似直线：$\log n_i = -\alpha \log i + c$

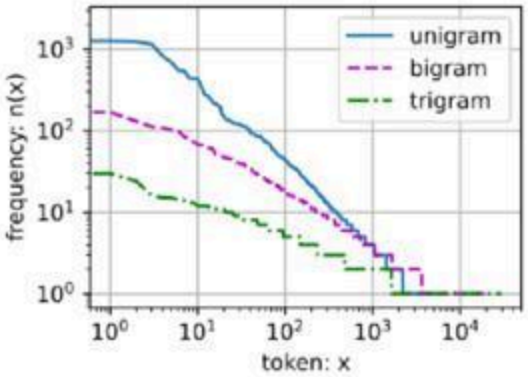


- 词表中N元组数量并不多，却可以表述非常复杂的语句
  - 说明语言中存在大量**结构**，可应用模型提取模式
- 很多N元组频率非常低：似然估计 + **平滑处理（如加一）不适合语言模型**
  - 信息非常稀疏：直接改公式对统计规律的破坏过于严重

N元模型词频比较

都遵循Zipf定律：$n_i \propto \frac{1}{i^\alpha}$

- **双对数空间**上近似直线：$\log n_i = -\alpha \log i + c$

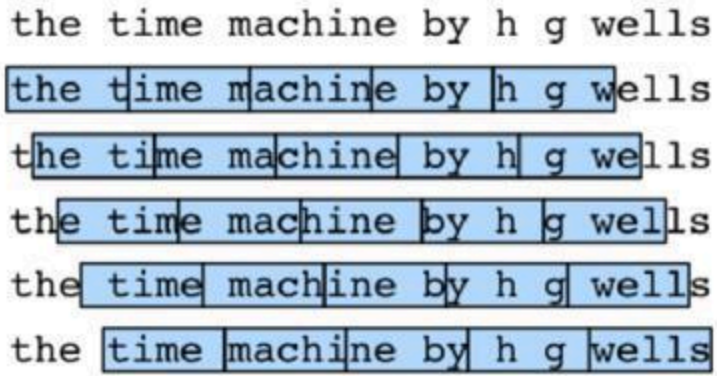


- 词表中N元组数量并不多，却可以表述非常复杂的语句
  - 说明语言中存在大量**结构**，可应用模型提取模式
- 很多N元组频率非常低：似然估计 + **平滑处理（如加一）不适合语言模型**
  - 信息非常稀疏：直接改公式对统计规律的破坏过于严重

推论：需要深度神经网络模型学习语言内在规律

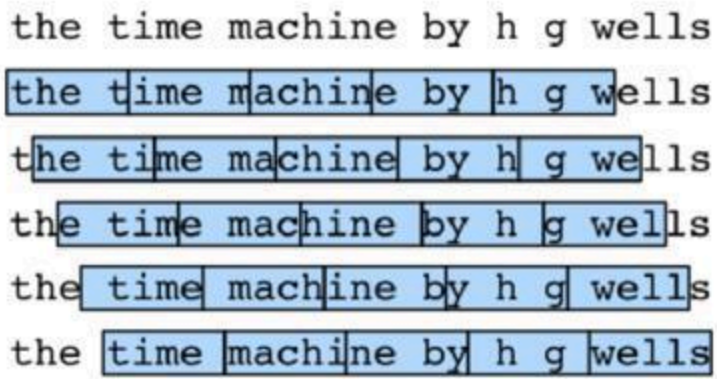
构造序列化数据集

序列太长、在线应用：拆分成**相同时间步**的子序列方便模型读取



构造序列化数据集

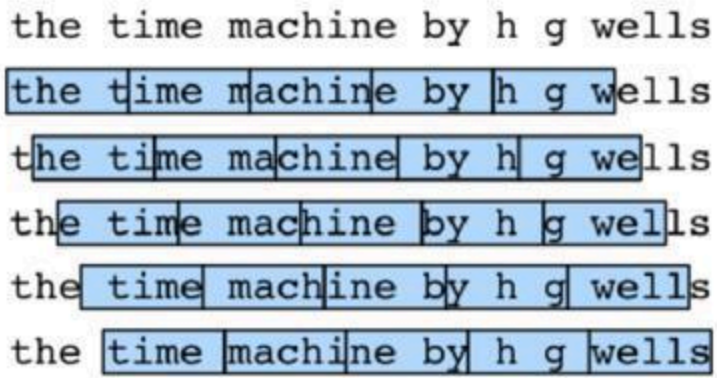
序列太长、在线应用：拆分成**相同时间步**的子序列方便模型读取



- 预测词元：真实标签对应原序列中**下一个位置**

构造序列化数据集

序列太长、在线应用：拆分成**相同时间步**的子序列方便模型读取



- 预测词元：真实标签对应原序列中**下一个位置**
- 起始位置：随机偏移量
- 采样策略：随机（生成文本更多变化）、顺序（生成文本更连贯）

实验：语言模型和数据集

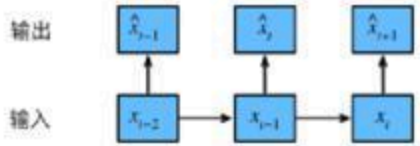
小结：语言模型和数据集

- 语言模型：预测句子或下一词的概率
  - 联合或条件概率
- N元语法：连续N个词看作一个单元
  - Markov假设

循环神经网络

回顾：自回归模型

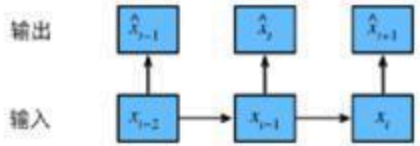
N元语法（Markov假设）：当前预测取决于前N个观测



- 长距离依赖？必须增加N（回忆德语例子）
  - 参数量呈**指数增长**：词表存储$|V|^N$，例如8000^{100}？

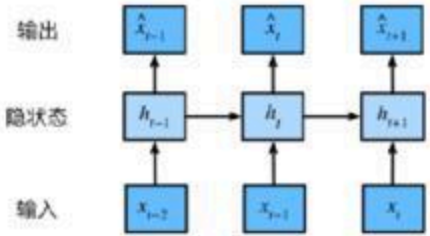
回顾：自回归模型

N元语法（Markov假设）：当前预测取决于前N个观测



- 长距离依赖？必须增加N（回忆德语例子）
  - 参数量呈**指数增长**：词表存储$|V|^N$，例如8000^{100}？

隐变量：（仅）**整合**到目前为止观测到的所有数据信息



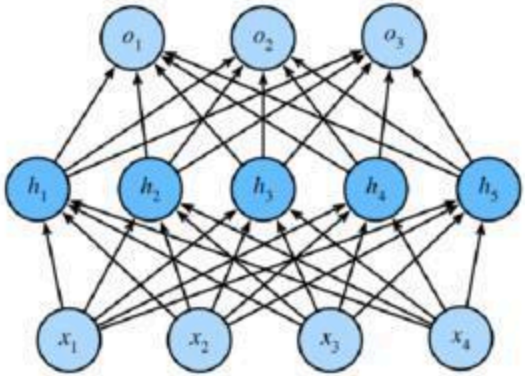
回顾：单隐藏层的MLP

隐藏层参数：提取特征，即对数据“不同角度的描述”

- 可学习、固定公式、自动计算：自动微分 + 梯度下降

$$\mathbf{H} = \sigma(\mathbf{X}\mathbf{W}^h + \mathbf{b}^h)$$
$$\mathbf{O} = \mathbf{H}\mathbf{W}^o + \mathbf{b}^o$$

- $\mathbf{X} \in \mathbb{R}^{B\times d}, \mathbf{W}^h \in \mathbb{R}^{d\times l}, \mathbf{b}^h \in \mathbb{R}^{1\times l}$
- $\mathbf{H} \in \mathbb{R}^{B\times l}, \mathbf{W}^o \in \mathbb{R}^{l\times C}, \mathbf{b}^o \in \mathbb{R}^{1\times C}$



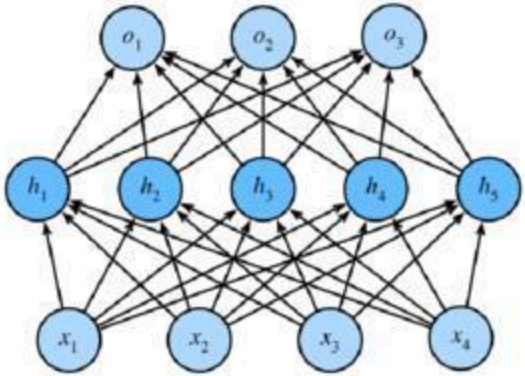
回顾：单隐藏层的MLP

隐藏层参数：提取特征，即对数据“不同角度的描述”

- 可学习、固定公式、自动计算：自动微分 + 梯度下降

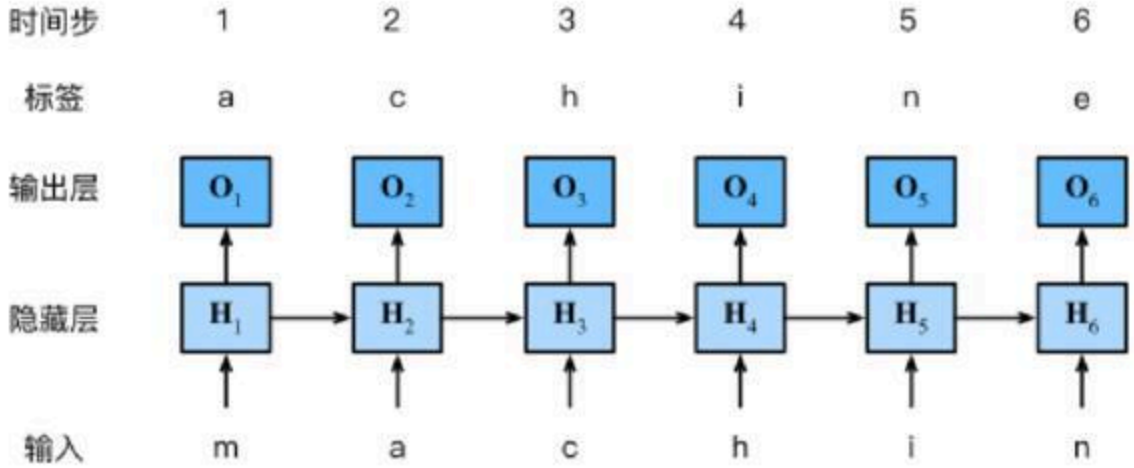
$$\mathbf{H} = \sigma(\mathbf{X}\mathbf{W}^h + \mathbf{b}^h)$$
$$\mathbf{O} = \mathbf{H}\mathbf{W}^o + \mathbf{b}^o$$

- $\mathbf{X} \in \mathbb{R}^{B\times d}, \mathbf{W}^h \in \mathbb{R}^{d\times l}, \mathbf{b}^h \in \mathbb{R}^{1\times l}$
- $\mathbf{H} \in \mathbb{R}^{B\times l}, \mathbf{W}^o \in \mathbb{R}^{l\times C}, \mathbf{b}^o \in \mathbb{R}^{1\times C}$



注意：不是隐变量，没有按时间步“阶段性整合信息”

基于循环神经网络的字符级语言模型



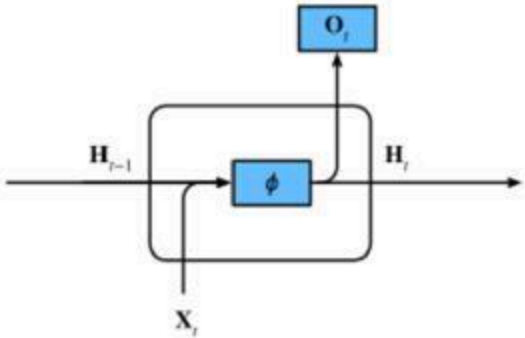
- 预测下一个词元：将原始序列移位一个词元作为标签

循环神经网络：时间步、隐变量

时间步t的更新：$h_t = f(x_{t-1}, h_{t-1})$

$$\mathbf{H}_t = \sigma(\mathbf{X}_t \mathbf{W}_x^h + \mathbf{H}_{t-1} \mathbf{W}_h^h + \mathbf{b}^h)$$
$$\mathbf{O}_t = \mathbf{H}_t \mathbf{W}_h^o + \mathbf{b}^o$$

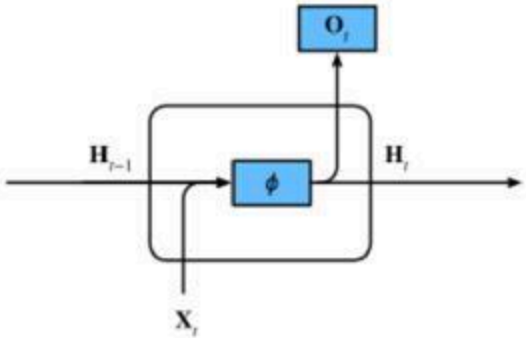
- $\mathbf{X}_t \in \mathbb{R}^{B \times d}$：时间步$t$的小批量样本
- $\mathbf{H}_t \in \mathbb{R}^{B \times l}$：时间步$t$的隐变量
- $\mathbf{W}_h^h \in \mathbb{R}^{l \times l}$：时间步之间隐变量的更新



循环神经网络：时间步、隐变量

时间步t的更新：$h_t = f(x_{t-1}, h_{t-1})$

$$\mathbf{H}_t = \sigma(\mathbf{X}_t \mathbf{W}_x^h + \mathbf{H}_{t-1} \mathbf{W}_h^h + \mathbf{b}^h)$$
$$\mathbf{O}_t = \mathbf{H}_t \mathbf{W}_h^o + \mathbf{b}^o$$



- $\mathbf{X}_t \in \mathbb{R}^{B \times d}$：时间步$t$的小批量样本
- $\mathbf{H}_t \in \mathbb{R}^{B \times l}$：时间步$t$的隐变量
- $\mathbf{W}_h^h \in \mathbb{R}^{l \times l}$：时间步之间隐变量的更新

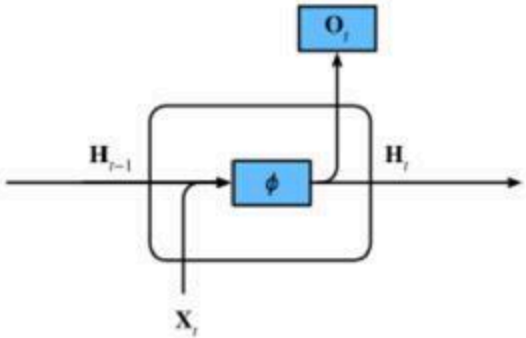
注意：不同时间步使用**相同参数**$\mathbf{W}_x^h, \mathbf{W}_h^h, \mathbf{W}_h^o, b^h, b^o$

- 参数量不会随时间推移而增长，计算开销恒定

循环神经网络：时间步、隐变量

时间步t的更新：$h_t = f(x_{t-1}, h_{t-1})$

$$\mathbf{H}_t = \sigma(\mathbf{X}_t \mathbf{W}_x^h + \mathbf{H}_{t-1} \mathbf{W}_h^h + \mathbf{b}^h)$$
$$\mathbf{O}_t = \mathbf{H}_t \mathbf{W}_h^o + \mathbf{b}^o$$

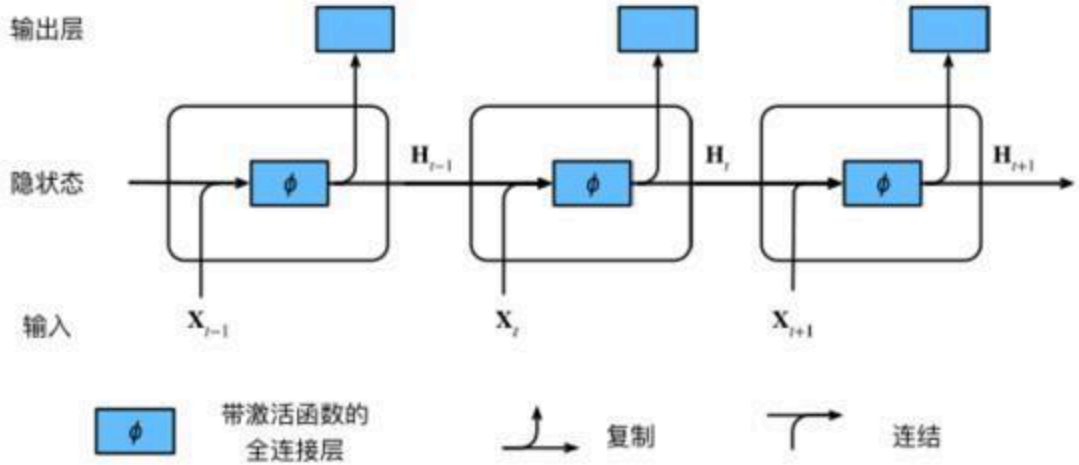


- $\mathbf{X}_t \in \mathbb{R}^{B \times d}$：时间步$t$的小批量样本
- $\mathbf{H}_t \in \mathbb{R}^{B \times l}$：时间步$t$的隐变量
- $\mathbf{W}_h^h \in \mathbb{R}^{l \times l}$：时间步之间隐变量的更新

注意：不同时间步使用**相同参数**$\mathbf{W}_x^h, \mathbf{W}_h^h, \mathbf{W}_h^o, b^h, b^o$

- 参数量不会随时间推移而增长，计算开销恒定
- “循环”：隐变量的当前时间步定义$\mathbf{H}_t$与前一时间步定义$\mathbf{H}_{t-1}$**模式相同**

循环神经网络：架构


输出层
隐状态
输入
$\mathbf{H}_{t-1}$
$\mathbf{H}_t$
$\mathbf{H}_{t+1}$
$\mathbf{X}_{t-1}$
$\mathbf{X}_t$
$\mathbf{X}_{t+1}$
ϕ
带激活函数的
全连接层
复制
连结

实验：循环神经网络

隐状态中$\mathbf{X}_t \mathbf{W}_x^h + \mathbf{H}_{t-1} \mathbf{W}_h^h$

```
X, W_xh = torch.normal(0, 1, (3, 1)), torch.normal(0, 1, (1, 4))
H, W_hh = torch.normal(0, 1, (3, 4)), torch.normal(0, 1, (4, 4))
torch.matmul(X, W_xh) + torch.matmul(H, W_hh)
```

```
tensor([[-0.5702, -0.4673, -0.1508, -0.0185],
        [-1.3559,  0.8042, -1.4394, -3.1735],
        [ 0.6559,  1.3140,  0.2949,  0.0781]])
```

实验：循环神经网络

隐状态中$\mathbf{X}_t \mathbf{W}_x^h + \mathbf{H}_{t-1} \mathbf{W}_h^h$

```
X, W_xh = torch.normal(0, 1, (3, 1)), torch.normal(0, 1, (1, 4))
H, W_hh = torch.normal(0, 1, (3, 4)), torch.normal(0, 1, (4, 4))
torch.matmul(X, W_xh) + torch.matmul(H, W_hh)
```

```
tensor([[-0.5702, -0.4673, -0.1508, -0.0185],
        [-1.3559,  0.8042, -1.4394, -3.1735],
        [ 0.6559,  1.3140,  0.2949,  0.0781]])
```

等价于$\mathbf{X}_t$和$\mathbf{H}_{t-1}$拼接与$\mathbf{W}_x^h$和$\mathbf{W}_h^h$**拼接的矩阵乘法**：并行计算

```
torch.matmul(torch.cat((X, H), 1), torch.cat((W_xh, W_hh), 0))
```

```
tensor([[-0.5702, -0.4673, -0.1508, -0.0185],
        [-1.3559,  0.8042, -1.4394, -3.1735],
        [ 0.6559,  1.3140,  0.2949,  0.0781]])
```

困惑度

困惑度 perplexity (PP)：模型在测试集上算得概率的倒数，再由词数归一化

$$
\begin{aligned}
PP(W) &= P(w_1 w_2 .. w_N)^{-\frac{1}{N}} \\
&= \sqrt[N]{\frac{1}{P(w_1 w_2 .. w_N)}} = \sqrt[N]{\prod_{k=1}^{N} \frac{1}{P(w_k | w_{1:k-1})}}
\end{aligned}
$$

困惑度

困惑度 perplexity (PP)：模型在测试集上算得概率的倒数，再由词数归一化

$$
\begin{aligned}
PP(W) &= P(w_1 w_2 .. w_N)^{-\frac{1}{N}} \\
&= \sqrt[N]{\frac{1}{P(w_1 w_2 .. w_N)}} = \sqrt[N]{\prod_{k=1}^{N} \frac{1}{P(w_k | w_{1:k-1})}}
\end{aligned}
$$

- 实际计算：$\exp(\log PP(W)) = \exp\left(-\frac{1}{N} \sum \log P(w_k | w_{1:k-1})\right)$

困惑度

困惑度 perplexity (PP)：模型在测试集上算得概率的倒数，再由词数归一化

$$
\begin{aligned}
PP(W) &= P(w_1 w_2 .. w_N)^{-\frac{1}{N}} \\
&= \sqrt[N]{\frac{1}{P(w_1 w_2 .. w_N)}} = \sqrt[N]{\prod_{k=1}^{N} \frac{1}{P(w_k | w_{1:k-1})}}
\end{aligned}
$$

- 实际计算：$\exp(\log PP(W)) = \exp\left(-\frac{1}{N} \sum \log P(w_k | w_{1:k-1})\right)$

信息论中**平均交叉熵**：$-\frac{1}{N} \sum \log P(w_k | w_{1:k-1})$

- 何为准确预测？压缩序列容易、存储开销低

困惑度：案例

困惑度 perplexity (PP)：模型在测试集上算得概率的倒数，再由词数归一化

$$
\begin{aligned}
PP(W) &= P(w_1 w_2 .. w_N)^{-\frac{1}{N}} \\
&= \sqrt[N]{\frac{1}{P(w_1 w_2 .. w_N)}} = \sqrt[N]{\prod_{k=1}^{N} \frac{1}{P(w_k | w_{1:k-1})}}
\end{aligned}
$$

下一位置可能出现的词数（即**分支因子**）的加权组合

困惑度：案例

困惑度 perplexity (PP)：模型在测试集上算得概率的倒数，再由词数归一化

$$
\begin{aligned}
PP(W) &= P(w_1 w_2 .. w_N)^{-\frac{1}{N}} \\
&= \sqrt[N]{\frac{1}{P(w_1 w_2 .. w_N)}} = \sqrt[N]{\prod_{k=1}^{N} \frac{1}{P(w_k | w_{1:k-1})}}
\end{aligned}
$$

下一位置可能出现的词数（即**分支因子**）的加权组合

- 完美预测：$PP = 1$
- 0命中：$PP = \infty$

困惑度：案例

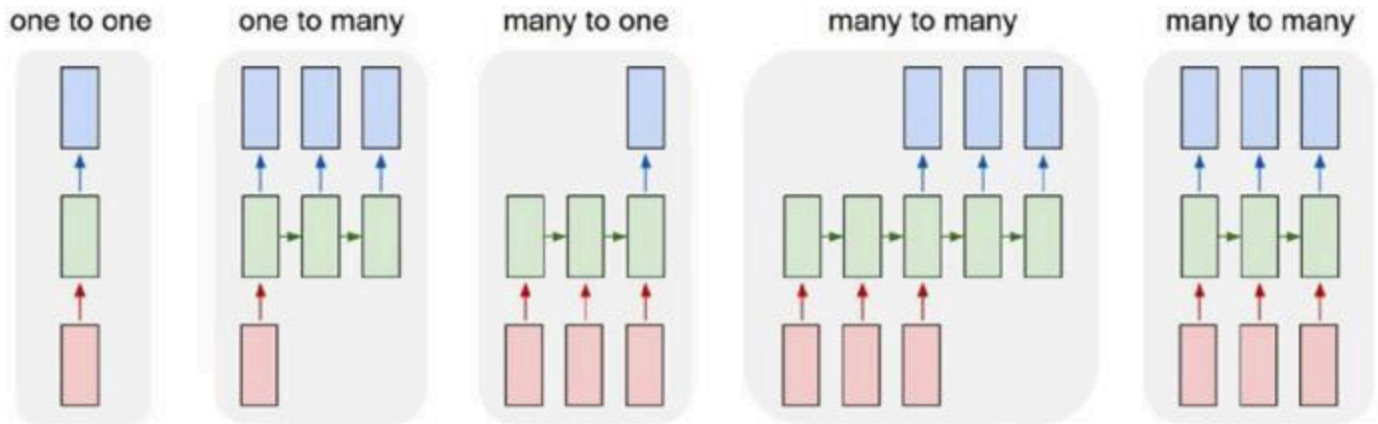
困惑度 perplexity (PP)：模型在测试集上算得概率的倒数，再由词数归一化

$$PP(W) = P(w_1 w_2 .. w_N)^{-\frac{1}{N}}$$
$$= \sqrt[N]{\frac{1}{P(w_1 w_2 .. w_N)}} = \sqrt[N]{\prod_{k=1}^{N} \frac{1}{P(w_k | w_{1:k-1})}}$$

下一位置可能出现的词数（即**分支因子**）的加权组合

- 完美预测：$PP = 1$
- 0命中：$PP = \infty$
- 基准：所有可用词元上等概率分布，PP = 词目数
  - 无压缩存储序列：等概率情况下的最优编码方式
  - 可以看成**实践上限**：经过设计的模型理应超过随机猜测（类比抛硬币）

RNN：应用


one to one
one to many
many to one
many to many
many to many
文本生成
文本分类
问答、机器翻译
Tag生成

小结：循环神经网络

- RNN：输出取决于当前输入、前一隐变量
- 语言模型应用：根据当前词预测下一时间步
- 困惑度：度量语言模型，加权的分支因子

实验：循环神经网络的从零开始实现

独热编码

将词元编码为相互独立的单位向量

预热

文本生成任务有时给定一小段“预热字串”

- 例如：“我永远不能忘记……”

预热

文本生成任务有时给定一小段“预热字串”

- 例如：“我永远不能忘记……”
- 可以用于优化模型参数，但不需要输出预测
  - 预热：**有监督的初始化**，获得更好的参数初始值

梯度剪裁

对长度T的序列：反向传播产生$O(T)$的矩阵乘法链

- 数值不稳定：梯度爆炸、消失

梯度剪裁

对长度T的序列：反向传播产生$O(T)$的矩阵乘法链

- 数值不稳定：梯度爆炸、消失

Lipschitz 假设：$|f(\mathbf{x}_1) - f(\mathbf{x}_2)| \leq L\|\mathbf{x}_1 - \mathbf{x}_2\|$

- 梯度更新：$|f(\mathbf{x}) - f(\mathbf{x} - \mathbf{g}\eta)| \leq L\eta\|\mathbf{g}\|$
  - 限制学习率：限制优化进展、数值问题

梯度剪裁

对长度T的序列：反向传播产生$O(T)$的矩阵乘法链

- 数值不稳定：梯度爆炸、消失

Lipschitz 假设：$|f(\mathbf{x}_1) - f(\mathbf{x}_2)| \leq L\|\mathbf{x}_1 - \mathbf{x}_2\|$

- 梯度更新：$|f(\mathbf{x}) - f(\mathbf{x} - \mathbf{g}\eta)| \leq L\eta\|\mathbf{g}\|$
  - 限制学习率：限制优化进展、数值问题

梯度剪裁：限制梯度，即$\mathbf{g} \leftarrow \min\left(1, \frac{\theta}{\|\mathbf{g}\|}\right)\mathbf{g}$

- 可以防止梯度爆炸，但不能应对梯度消失

实验：循环神经网络的简洁实现

Review

本章内容

序列模型。语言模型。循环神经网络。

重点：序列模型、自回归模型、隐变量模型；N元语法、最大似然估计、平滑处理；序列化数据集的构造；循环神经网络的架构，及其预测、评测、应用方法。

难点：困惑度的原理；循环神经网络的实现。

学习目标

- 理解序列模型、自回归模型、隐变量模型
- 理解N元语法、最大似然估计、平滑处理
- 理解序列化数据集的构造方法
- 理解循环神经网络的架构，及其预测、评测、应用方法

问题

简述序列模型、自回归模型、隐变量模型的建模方法。

简述N元语法、最大似然估计、平滑处理的原理。

简述序列化数据集的构造方法。

简述循环神经网络的架构，及其预测、评测、应用方法。