Natural Language Processing and Its Applications

#L3

Neural Network based Language Model

袁彩霞

yuancx@bupt.edu.cn

智能科学与技术中心

Outline

- Motivation
- Word Representation
- Neural network based language models
  - Feedforward NNLM
  - Recurrent NNLM

N-grams

- Standard approach to language modeling

- Task: compute probability of a sentence W

$$P(w_1,w_2,w_3, \ldots,w_n) = \prod_i P(w_i | w_1,w_2, \ldots w_{i-1})$$

- Often simplified to trigrams:

$$P(w_1,w_2,w_3, \ldots,w_n) = \prod_i P(w_i | w_{i-2}, w_{i-1})$$

N-grams: example

- $P(\text{“}this\ is\ a\ sentence\text{“}) = P(this) \times P(is|this) \times P(a|this, is) \times P(sentence|is, a)$

- The probabilities are estimated from counts:

$$P(a|this, is) = \frac{C(this\ is\ a)}{C(this\ is)}$$

- Smoothing is used to redistribute probability to unseen events (this avoids zero probabilities)

n-gram模型的局限

- 问题1：数据稀疏问题
  - 理论上，模型阶数越高越好，但由于数据稀疏，N-gram模型中n达到一定值后，n越大性能反而越差(e.g., <6)，有没有可以算高阶的模型？
  - 同样，由于数据稀疏问题，平滑很重要，有没有不需要平滑就可以直接用的模型？

n-gram模型的局限

- 问题2：没有考虑词的含义
  - The cat is walking in the bedroom的训练样本对 A dog was running in a room的句子概率无贡献
    - 没有相同的bigram
  - 更细节：p(eat|cat)和p(eat|dog)无关
    - cat 和 dog无关，所以两个概率无关，因此，在某些语料中，这两个值可能差别很大
    - 词相似，概率有理由相似：p(eat|cat) ~ p(eat|dog)

- 基于符号的词表示方法做不到！
- 词的符号表示？
- 词的语义表示？
  - How to represent word meaning?

How do we represent the meaning of a word?

- Definition: **Meaning (Webster dictionary)**
  - the idea that is represented by a word, phrase, etc.
  - the idea that a person wants to express by using words, signs, etc.
  - the idea that is expressed in a work of writing, art, etc.
  - **词义：词的含义。**(也有词典解释为：**词语的意义**)

How to represent meaning in a computer?

- Common answer: Use a taxonomy like WordNet that has hypernyms (is-a) relationships and synonym sets

[Synset('procyonid.n.01'),
Synset('carnivore.n.01'),
Synset('placental.n.01'),
Synset('mammal.n.01'),
Synset('vertebrate.n.01'),
Synset('chordate.n.01'),
Synset('animal.n.01'),
Synset('organism.n.01'),
Synset('living_thing.n.01'),
Synset('whole.n.02'),
Synset('object.n.01'),
Synset('physical_en4ty.n.01'),
Synset('en4ty.n.01')]

S: (adj) full, good
S: (adj) es4mable, good, honorable, respectable
S: (adj) beneficial, good
S: (adj) good, just, upright
S: (adj) adept, expert, good, prac4ced, proficient, skillful
S: (adj) dear, good, near
S: (adj) good, right, ripe
…
S: (adv) well, good
S: (adv) thoroughly, soundly, good
S: (n) good, goodness
S: (n) commodity, trade good, good

Problems with this discrete representation

- Great as resource but missing nuances, e.g.
  - **synonyms:** adept, expert, good, practiced, proficient, skillful?
- Missing new words (impossible to keep up to date): wicked, badass, crack, ace, wizard, genius, ninjia
- Subjective
- Requires human labor to create and adapt
- Hard to compute accurate word similarity

Problems with this discrete representation

- The vast majority of rule-based **and statistical NLP work regards**
- words as atomic symbols: hotel, conference, walk
- In vector space terms, this is a vector with one 1 and a lot of zeroes

  [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]

- Dimensionality:

  20K (speech) – 50K (PTB) – 500K (big vocabulary) – 13M (Google 1T)

- We call this a "one-hot" representation. Its problem:

  motel [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0] AND

  hotel [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0] = 0

Distributional similarity based representations

- You can get a lot of value by representing a word by means of its neighbors

"You shall know a word by the company it keeps"

(J. R. Firth 1957: 11)

- One of the most successful ideas of modern statistical NLP

government debt problems turning into *banking* crises as has happened in

Europe needs unified *banking* regulation to replace the hodgepodge

- These words will represent *banking*

How to make neighbors represent words?

- Answer: With a cooccurrence matrix X
- 2 options: full document vs windows
  - Word - document cooccurrence matrix: give general topics (all sports terms will have similar entries) leading to “Latent Semantic Analysis”
  - Window around each word: captures both syntactic and semantic information

Window based cooccurence matrix

- Window length l (e.g., 1, more common: 5 - 10)
- Symmetric (irrelevant whether left or right context)
- Example corpus:
  - I like deep learning.
  - I like NLP.
  - I enjoy flying.

Window based cooccurence matrix

- Example corpus:
  - I like deep learning.
  - I like NLP.
  - I enjoy flying.

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

Problems with simple cooccurrence vectors

- Increase in size with vocabulary
- Very high dimensional: require a lot of storage
- Sparsity issues → Models are less robust

Solution: Low dimensional vectors

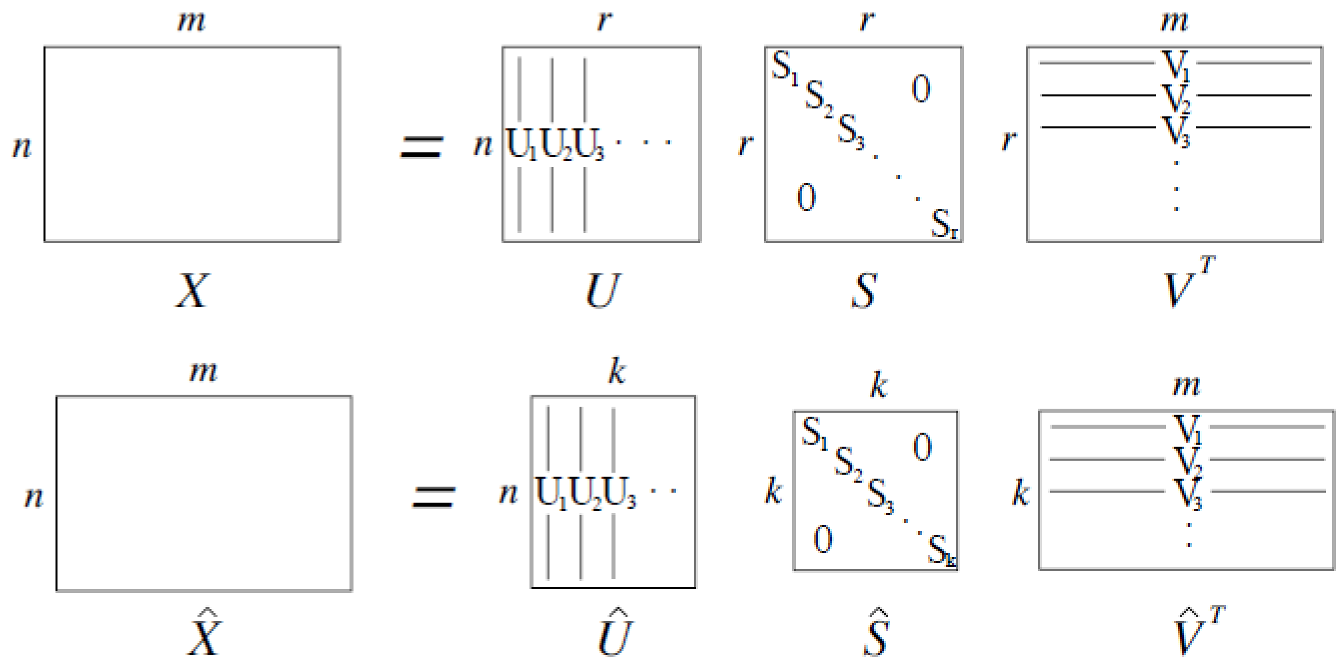
- Idea: store "most" of the important information in a fixed, small number of dimensions: a dense vector

- Usually around 25 – 1000 dimensions

- How to reduce the dimensionality?

Method 1: Dimensionality Reduction on X

- Singular Value Decomposition of cooccurrence matrix *X*



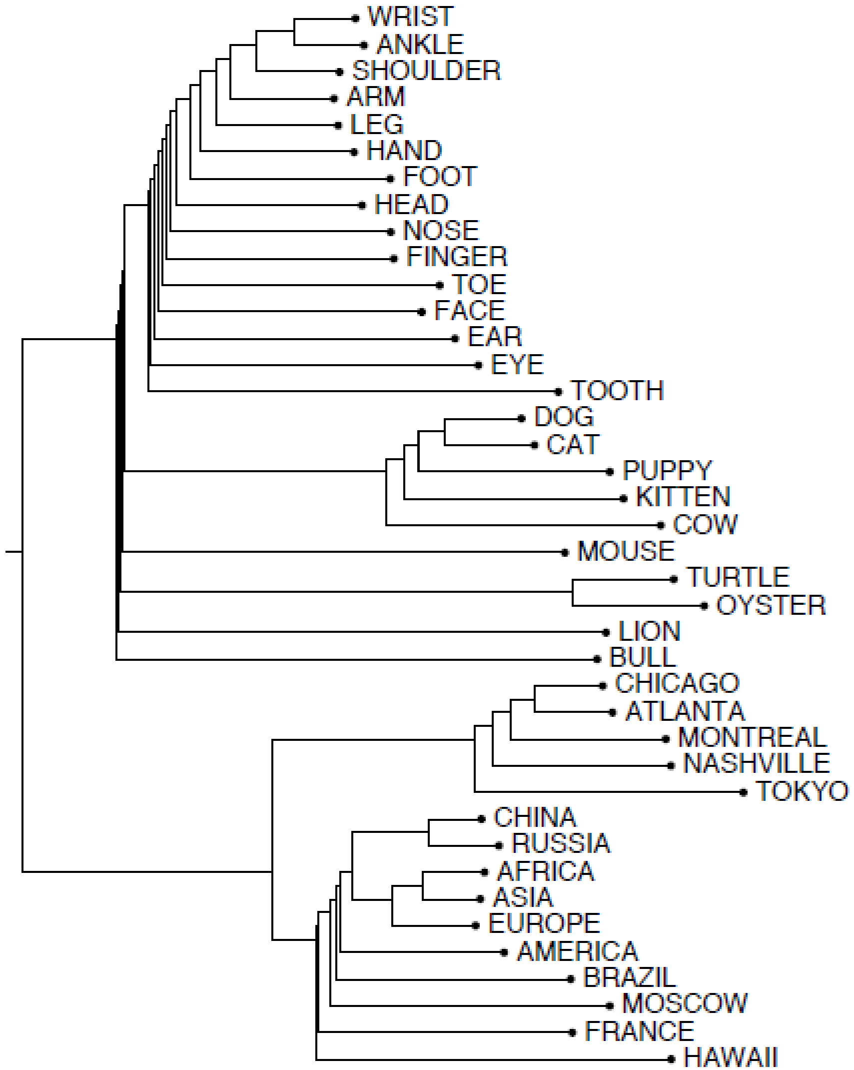
- X^ is the best rank k approximation to X, in terms of least squares.

Word meaning is defined in terms of vectors

- A word is represented as a dense vector

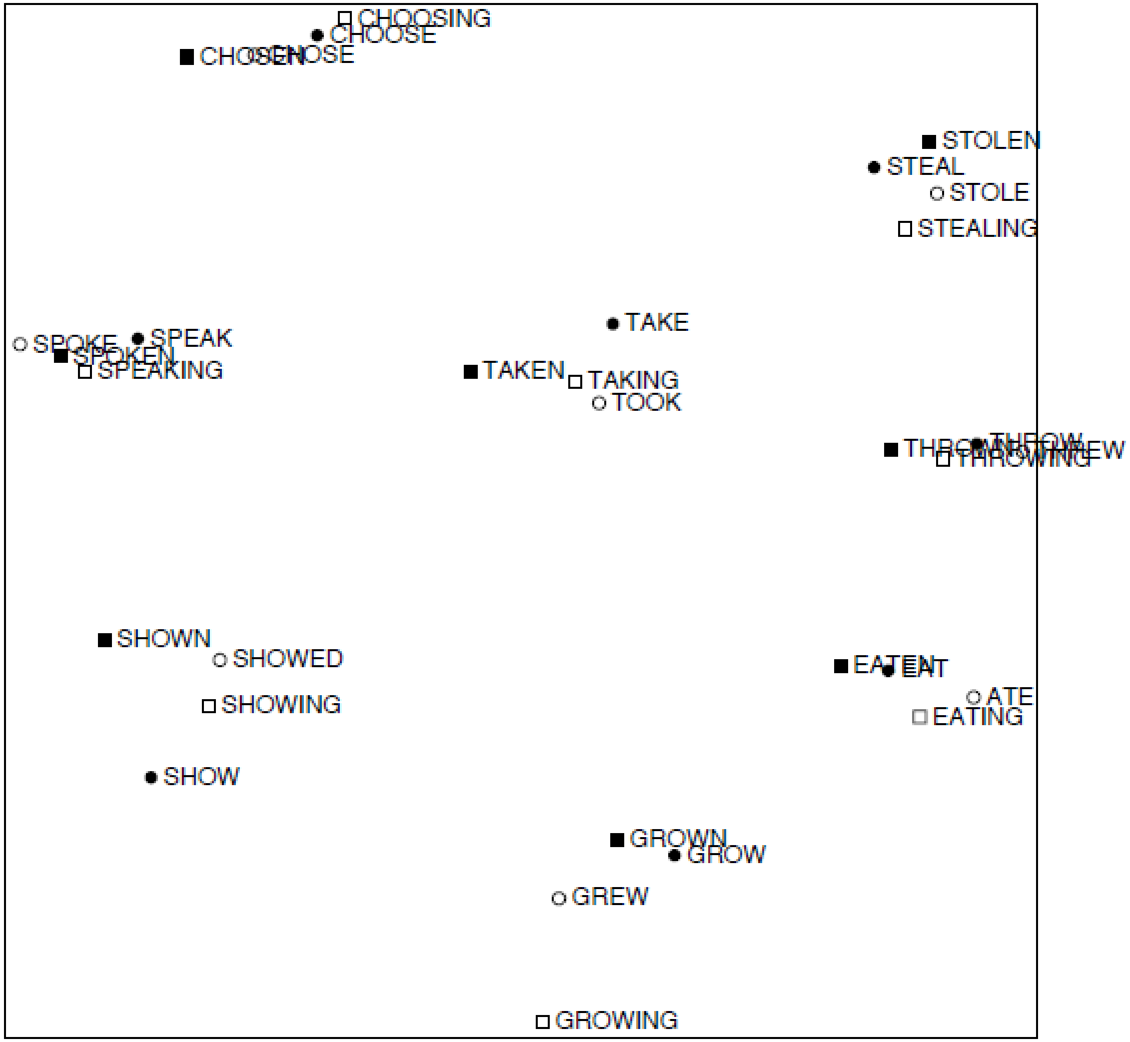
$$
\textit{linguistics} = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix}
$$

Interesting semantic patterns emerge in the vectors



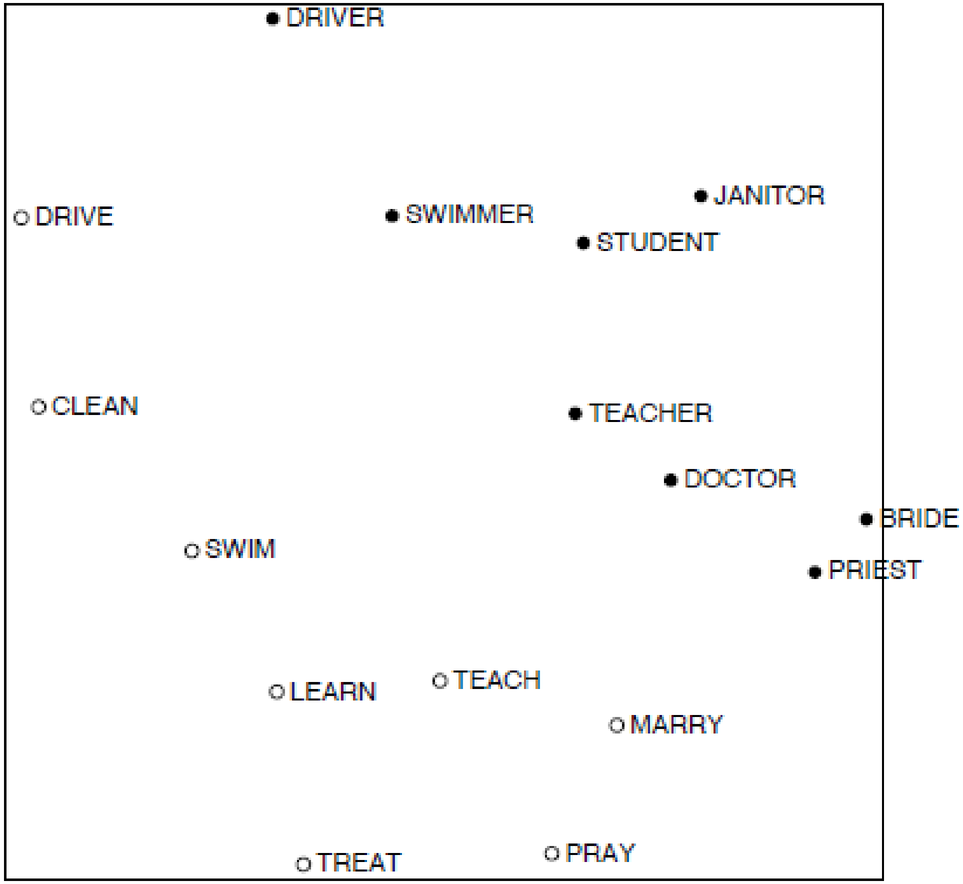
Rohde et al. An Improved Model of Semantic Similarity Based on Lexical Co-Occurrence. 2005.

Interesting syntactic patterns emerge in the vectors



Rohde et al. An Improved Model of Semantic Similarity Based on Lexical Co-Occurrence. 2005.

Interesting semantic patterns emerge in the vectors



Rohde et al. An Improved Model of Semantic Similarity Based on Lexical Co-Occurrence. 2005.

Problems with SVD

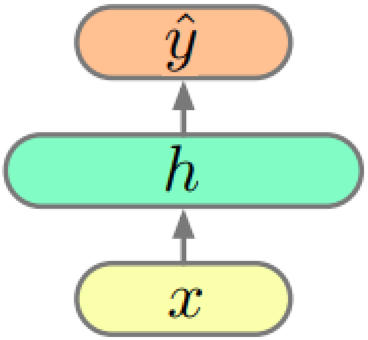
- Computational cost scales quadratically for n x m matrix: O(mn^2) *flops (when n<m)*
- → Bad for millions of words or documents
- Hard to incorporate new words or documents
- Different learning regime than other deep learning models

Idea: Directly learn low-dimensional word vectors

- Learning representations by back-propagating errors. (Rumelhart et al., 1986)
- A neural probabilistic language model (Bengio et al., 2003)
- NLP (almost) from Scratch (Collobert & Weston, 2008)
- Distributed Representations of Words and Phrases and their Compositionality (A recent, even simpler and faster model: word2vec, Mikolov et al. 2013)
- *Glove: Global Vectors for Word Representation* (Pennington et al., 2014 and Levy and Goldberg, 2014)
- … more later

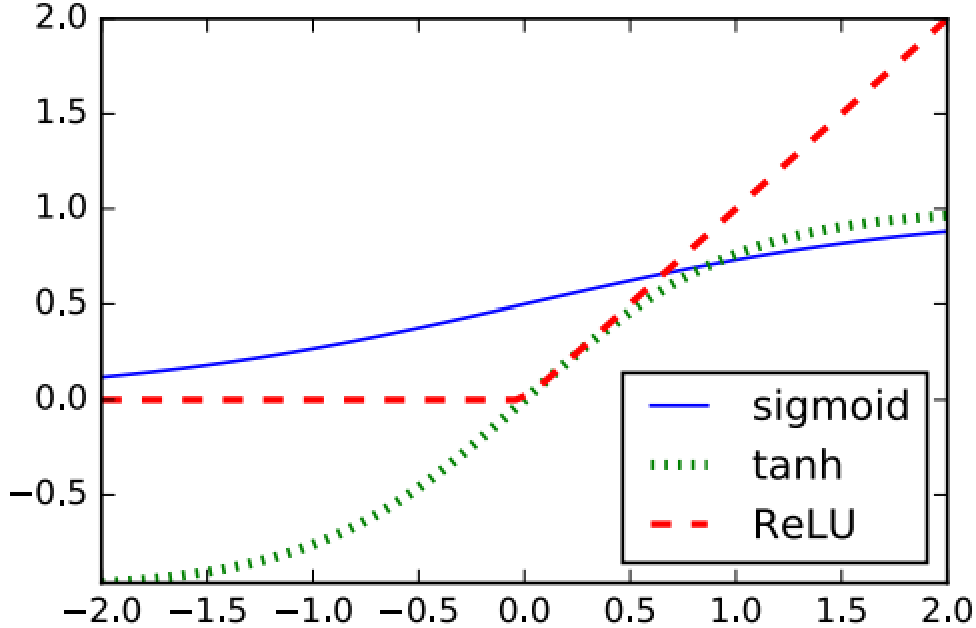
A Simple Feed forward Neural Network

- Feed forward network



$$
\begin{aligned}
h &= g(Vx + c) \\
\hat{y} &= Wh + b
\end{aligned}
$$

Nonlinear activation functions



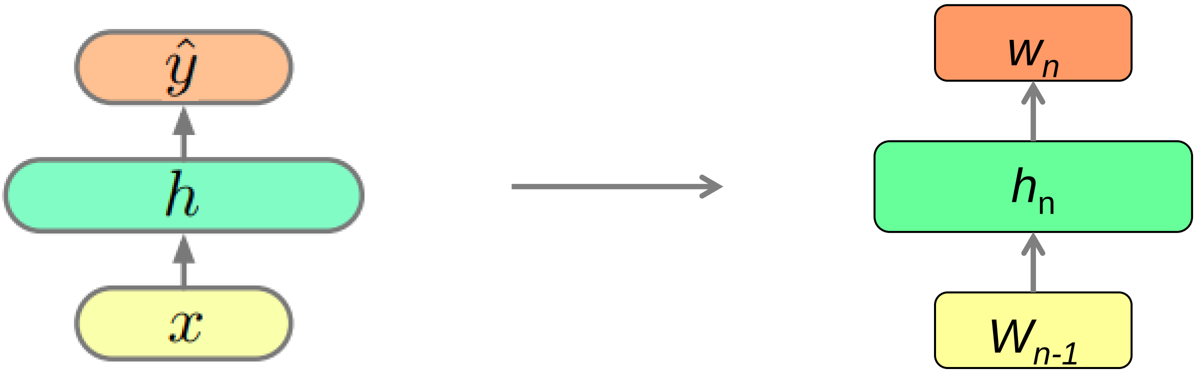
$$\mathrm{sigmoid}(x) = \frac{e^x}{1+e^x}$$

$$\tanh(x) = 2 \times \mathrm{sgm}(x) - 1$$

$$(x)_+ = \max(0, x)$$

a.k.a. "ReLU"

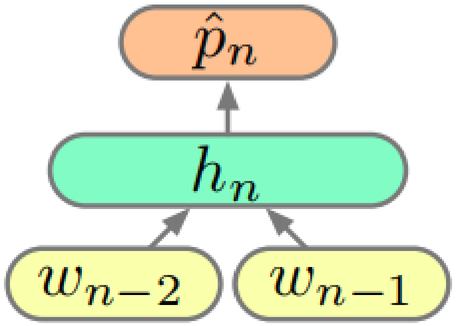
A (Very) Simple NN for Language Model



- 模拟了一个bigram语言模型
- 前一个词通过隐藏层的映射，来预测后一个词(输出层对应于|V|个分类器)

Trigram NN language model

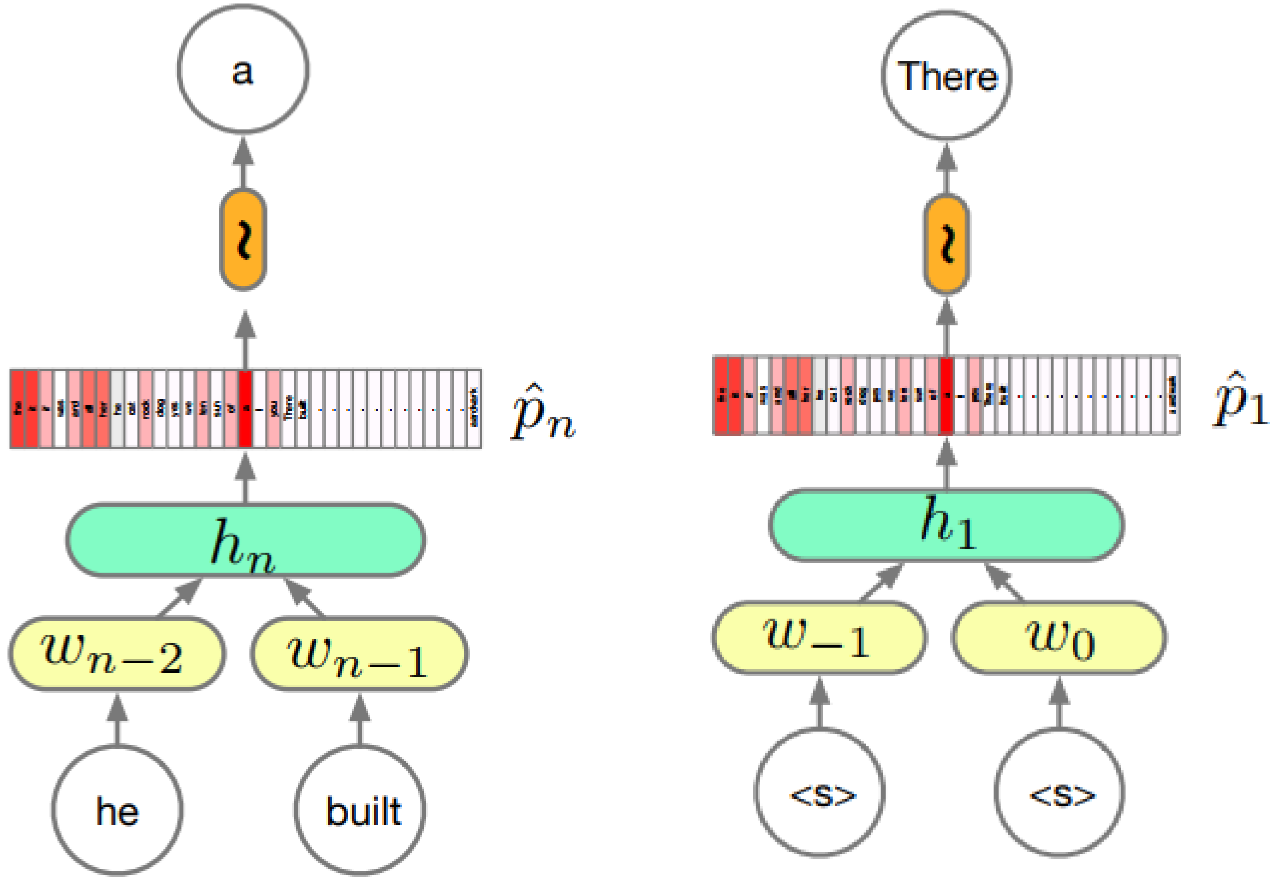
$$
\begin{aligned}
h_n &= g(V[w_{n-1}; w_{n-2}] + c) \\
\hat{p}_n &= \text{softmax}(Wh_n + b) \\
\text{softmax}(u)_i &= \frac{\exp u_i}{\sum_j \exp u_j}
\end{aligned}
$$



- w_i are one hot vetors and $p\hat{}_i$ are distributions
- $|wi| = |p\hat{}_i| = V$ (words in the vocabulary)
- V is usually very large > 1e5

- Sampling

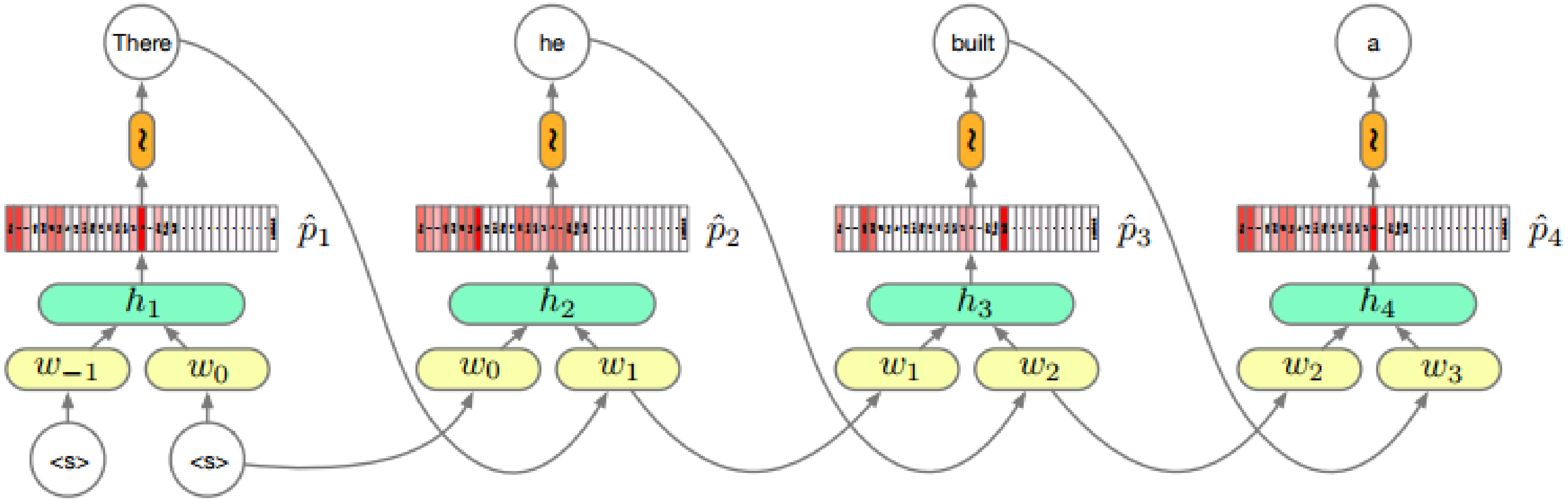
$$w_n | w_{n-1}, w_{n-2} \quad \sim \quad \hat{p}_n$$



Trigram NN language model

- Sampling

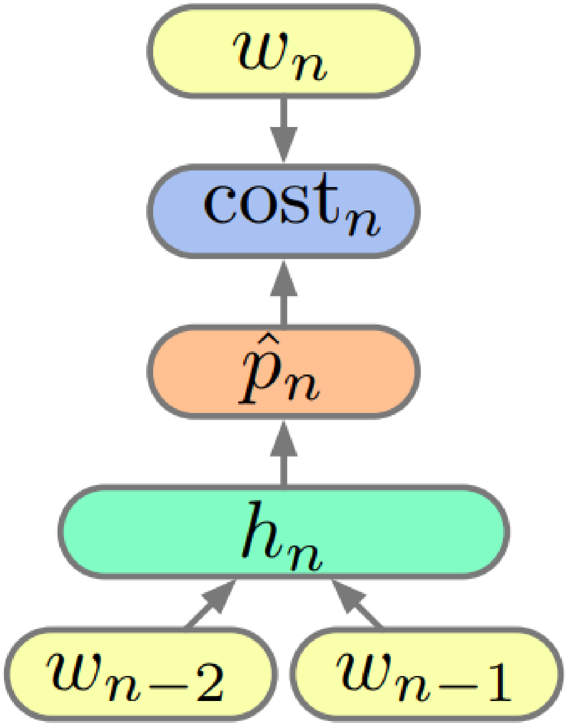
$$w_n | w_{n-1}, w_{n-2} \sim \hat{p}_n$$



Trigram NN language model

- Training: the usual training objective is the cross entropy of the data given the model

$$\mathcal{F} = -\frac{1}{N}\sum_{n} \mathrm{cost}_n(w_n, \hat{p}_n)$$

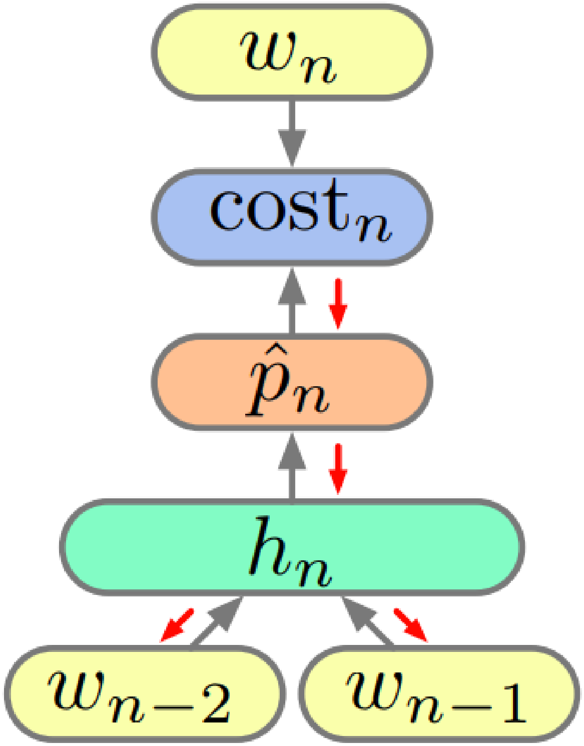


Trigram NN language model

- Training: Calculating the gradients is straightforward with back propagation

$$\frac{\partial \mathcal{F}}{\partial W} = -\frac{1}{N}\sum_n \frac{\partial \mathrm{cost}_n}{\partial \hat{p}_n}\frac{\partial \hat{p}_n}{\partial W}$$

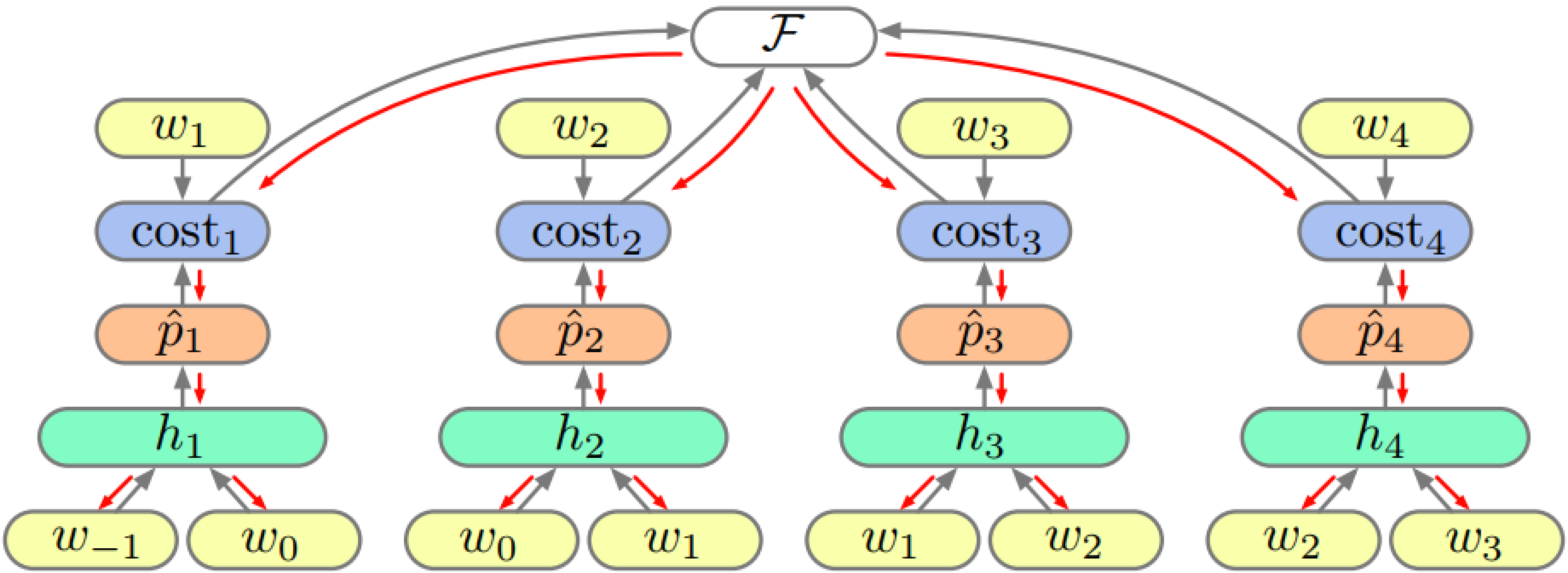
$$\frac{\partial \mathcal{F}}{\partial V} = -\frac{1}{N}\sum_n \frac{\partial \mathrm{cost}_n}{\partial \hat{p}_n}\frac{\partial \hat{p}_n}{\partial h_n}\frac{\partial h_n}{\partial V}$$



Trigram NN language model

- Training: Calculating the gradients is straightforward with back propagation:

$$\frac{\partial \mathcal{F}}{\partial W} = -\frac{1}{4}\sum_{n=1}^{4}\frac{\partial \mathrm{cost}_n}{\partial \hat{p}_n}\frac{\partial \hat{p}_n}{\partial W} \qquad \frac{\partial \mathcal{F}}{\partial V} = -\frac{1}{4}\sum_{n=1}^{4}\frac{\partial \mathrm{cost}_n}{\partial \hat{p}_n}\frac{\partial \hat{p}_n}{\partial h_n}\frac{\partial h_n}{\partial V}$$



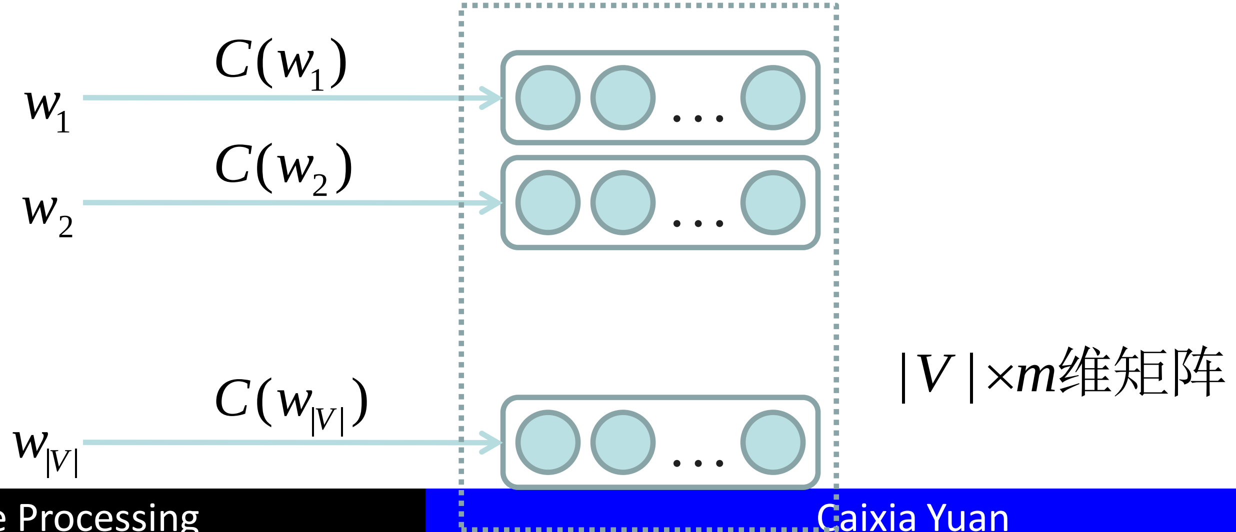
NNLM (Bengio et. al 2003)

- Bengio et. al (2003)模型贡献
  - 得到分布式词表示
  - 得到基于分布式词表示的语言模型
    - 高阶(例如6阶)，无需平滑
  - 实验表明比基于符号的语言模型更好
    - PP值评测
- 符号约定
  - x：变量
  - w_i：具体词
  - v'：v的转置
  - A_j：A的第j行

NNLM

- 模型结构：
  - 1.词表映射
    - 目标：对词表V 中的词(w_1,…w_i ,.. $w_{|V|}$) 得到其m维向量表示
    - 实现方式
      - 查表映射C：将任意词映射为一个m维向量

      - 对于V中所有词：将V中第i词w_i映射为C(w_i)，简记为C(i)



NNLM

- 2.神经网络模型
  - 模型目标：训练一个映射g来建模n元语言模型，即

$$g(C(x_t),C(x_{t-1}),..,C(x_{t-n+1});\omega)=P(C(x_t)\,|\,C(x_{t-1}),..,C(x_{t-n+1}))$$

    - 其中ω为神经网络参数
  - 训练的目标是使得该n元模型对于测试词序列$x_1,x_2,\ldots x_T$ (x_i均为词表V中的词)具有最小PP值，即极小化：

$$PP(C(x_1),\ldots,C(x_T))=P(C(x_1),\ldots,C(x_T))^{-\frac{1}{T}}$$
$$=(\prod_{t=1}^{T}P(C(x_t)\,|\,C(x_{t-1}),..,C(x_{t-n+1})))^{-\frac{1}{T}}$$

NNLM

- 即极大化：

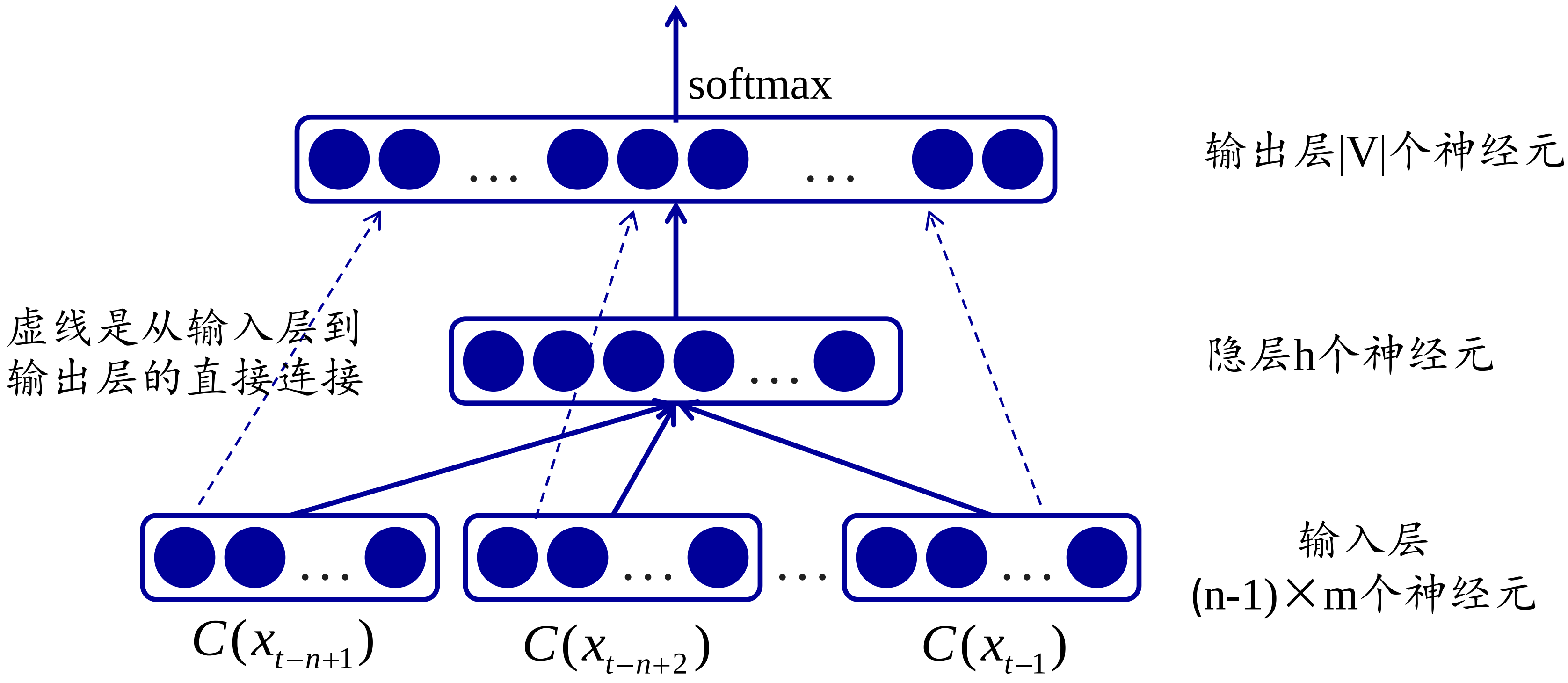
$$L=\frac{1}{T}\sum_{t=1}^{T}\log P(C(x_t)\,|\,C(x_{t-1}),..,C(x_{t-n+1}))$$

$$=\frac{1}{T}\sum_{t=1}^{T}\log g(C(x_t),C(x_{t-1}),..,C(x_{t-n+1});\omega)$$

NNLM

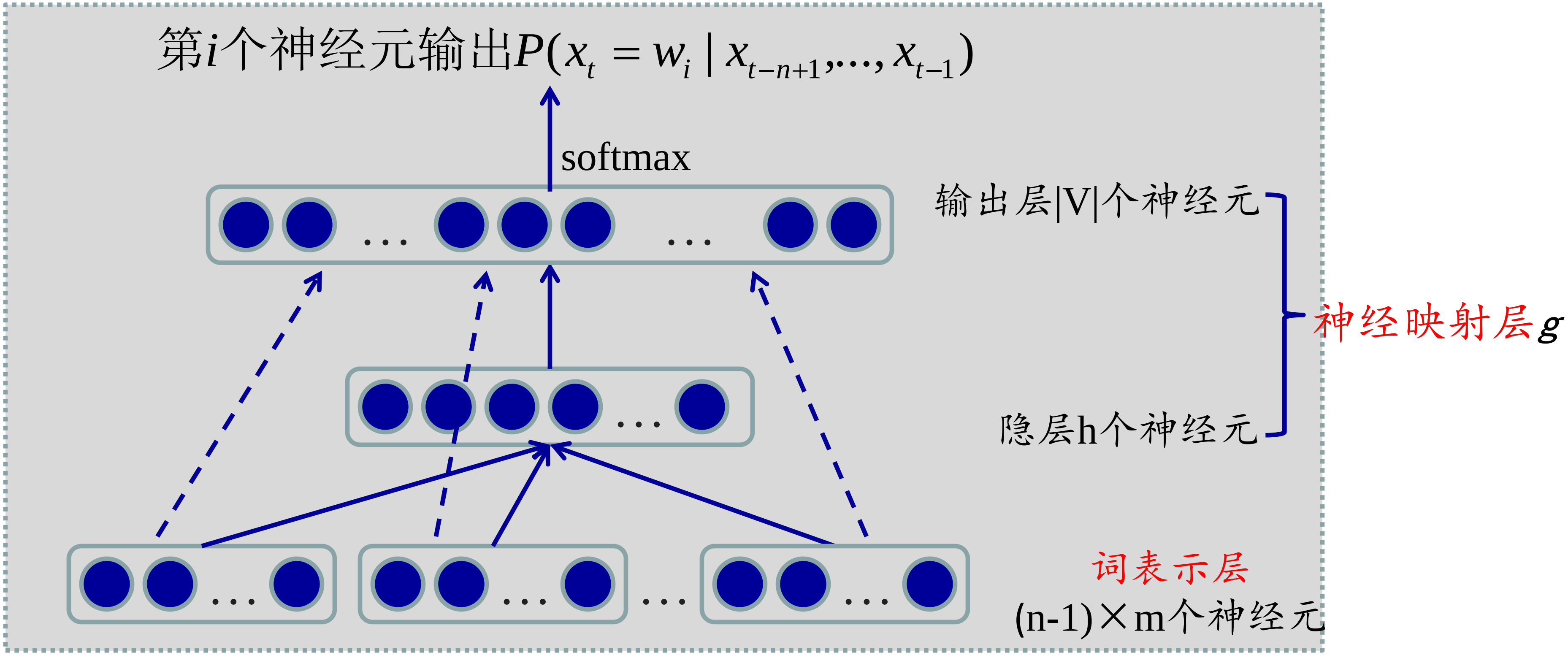
- 神经网络结构

第i个神经元输出为$P(C(x_t)=C(w_i)\,|\,C(x_{t-n+1}),..,C(x_{t-1}))$



NNLM

- 神经网络结构

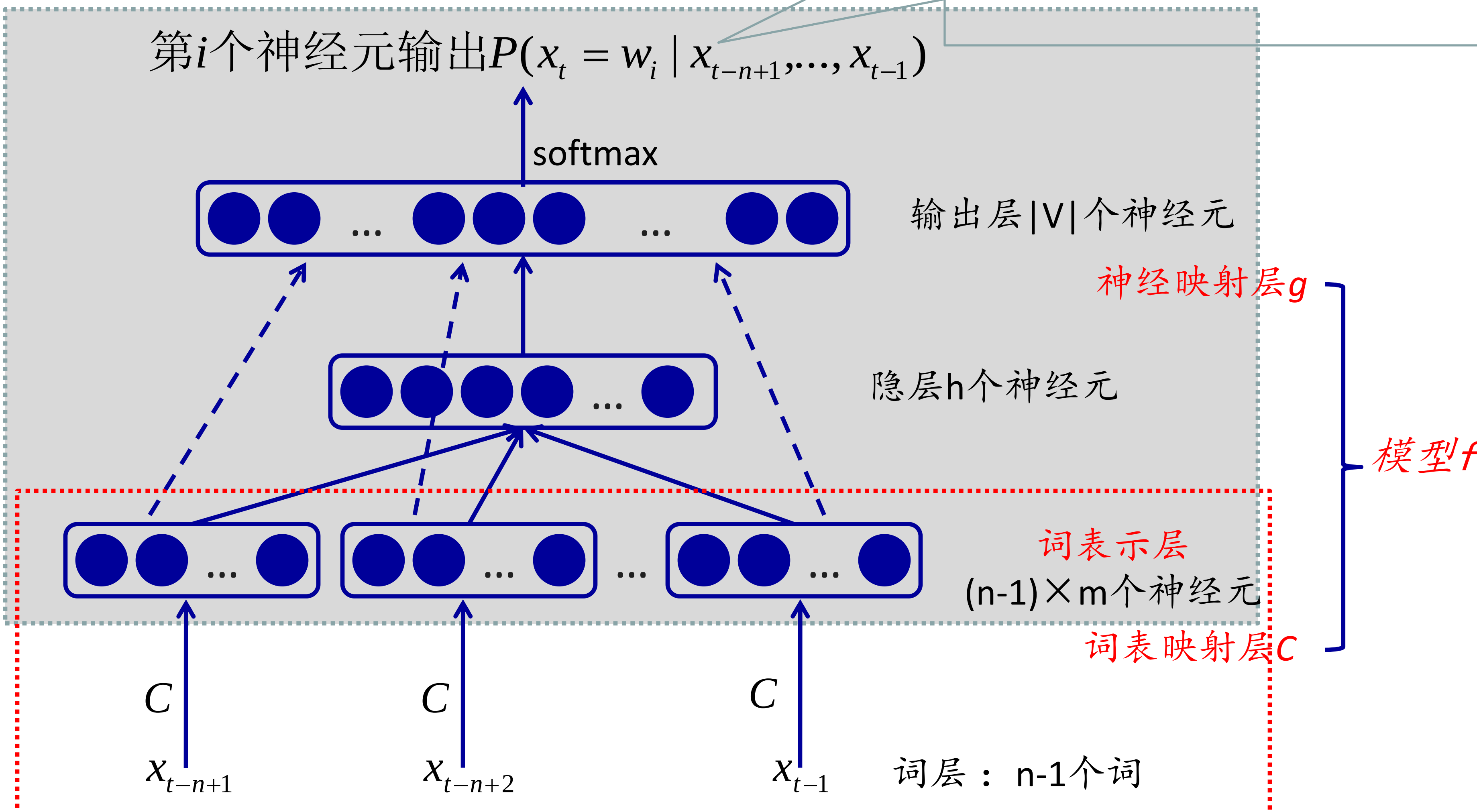


第i个神经元输出$P(x_t = w_i \mid x_{t-n+1},\ldots,x_{t-1})$
softmax
输出层|V|个神经元
神经映射层g
隐层h个神经元
词表示层
(n-1)×m个神经元

- 合并词表映射与神经网络模型

给定C时有：

$$P(C(x_t) \mid C(x_{t-n+1}),..,C(x_{t-1})) = P(x_t \mid x_{t-n+1},..,x_{t-1})$$



NNLM

- 整体模型的训练目标中把C也纳入，则为极大化：

$$L=\frac{1}{T}\sum_{t=1}^{T}\log g(C(x_t),C(x_{t-1}),..,C(x_{t-n+1});\omega)$$

$$=\frac{1}{T}\sum_{t=1}^{T}\log f(x_t,x_{t-1},..,x_{t-n+1};C,\omega)$$

- 加上正则化项，则为：

$$L=\frac{1}{T}\sum_{t=1}^{T}\log f(x_t,x_{t-1},..,x_{t-n+1};C,\omega)+R(C,\omega)$$

NNLM

- 模型参数
  - 各层
    - 词层n-1个节点（ n-gram语法的n-1个历史词）
    - 词表示层(n-1)×m个节点，每个词用m维向量表示
    - 隐层h个节点，阈值为d，h维
    - 输出层|V|个节点，阈值为b，|V|维
  - 层间
    - 词层到表示层：每一个词都对应一个向量表示，得到C=|V|×m矩阵
    - 表示层到隐层：权重H，(n-1)m×h矩阵
    - 表示层到输出层：权重W，(n-1)m×|V|矩阵
    - 隐层到输出层：权值U，h×|V|矩阵
- 总参数个数
  - |V|* (1+mn+h)+h*(1+ (n-1)m)

NNLM

- 模型计算：对每一个输入的n元串
  - 前向计算
    - 隐层输入为：y=b+W*C(x)+Utanh(d+HC(x))
    - 隐层输出为：

$$P(x_t \mid x_{t-1},\ldots,w_{t-n+1}) = \frac{e^{y_{x_t}}}{\sum_i e^{y_{x_i}}}$$

    - 其中：C(x)是词x的向量表示
    - 参数集：$\theta=(b,W,C,U,d,H)$
  - 反向随机梯度下降

$$\theta \leftarrow \theta + \varepsilon \frac{\partial \log P(x_t \mid x_{t-1},\ldots,w_{t-n+1})}{\partial \theta}$$

    - ε为学习率
    - 不在输入窗口中的词向量值不需要调整

Next lecture

- Language Model Evaluation
- 复习/自学：
  - 信息熵