Models for computing continuous vector representations of words

--Andy 张安迪

Curse of dimensionality

- Many NLP systems treat words as atomic units
- 10 different values of each of n variables(n could easily be thousands)

  10^n

- Complexity of the target function to be learned

- Fight against it by **learning a distributed representation for words**
- **10000+ → 30, 50, 100**

Models

- Neural network language model(NNLM)
- RNNLM
- Continuous bag-of-words model(CBOW)
- Continuous skip-gram model

NNLM

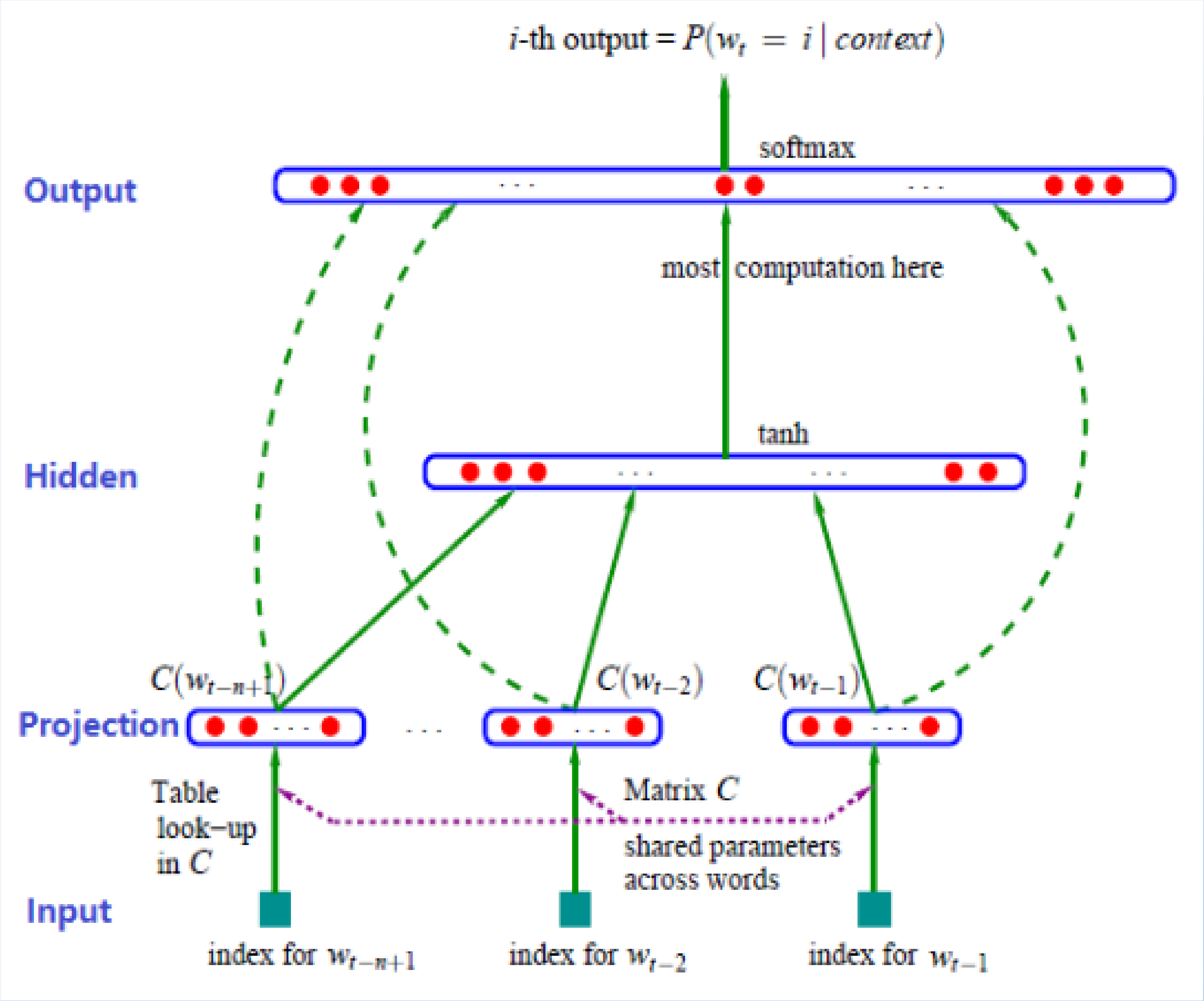
- Training set: V
- Input: previous N-1 words

  Using 1-of-V coding
- Projection layer:

$$C(w_t) = w_t * C$$
$$x = \left(C(w_{t-1}), C(w_{t-2}), \dots, C(w_{t-N+1})\right)$$

- C, a $|\boldsymbol{V}| \times \boldsymbol{m}$ matrix

  row i of C is the **word vector** for word i



NNLM(2)

- $x = \left(C(w_{t-1}), C(w_{t-2}), \dots, C(w_{t-N+1})\right)$
- Hidden layer:

$$\tanh(d + Hx)$$

- output layer:

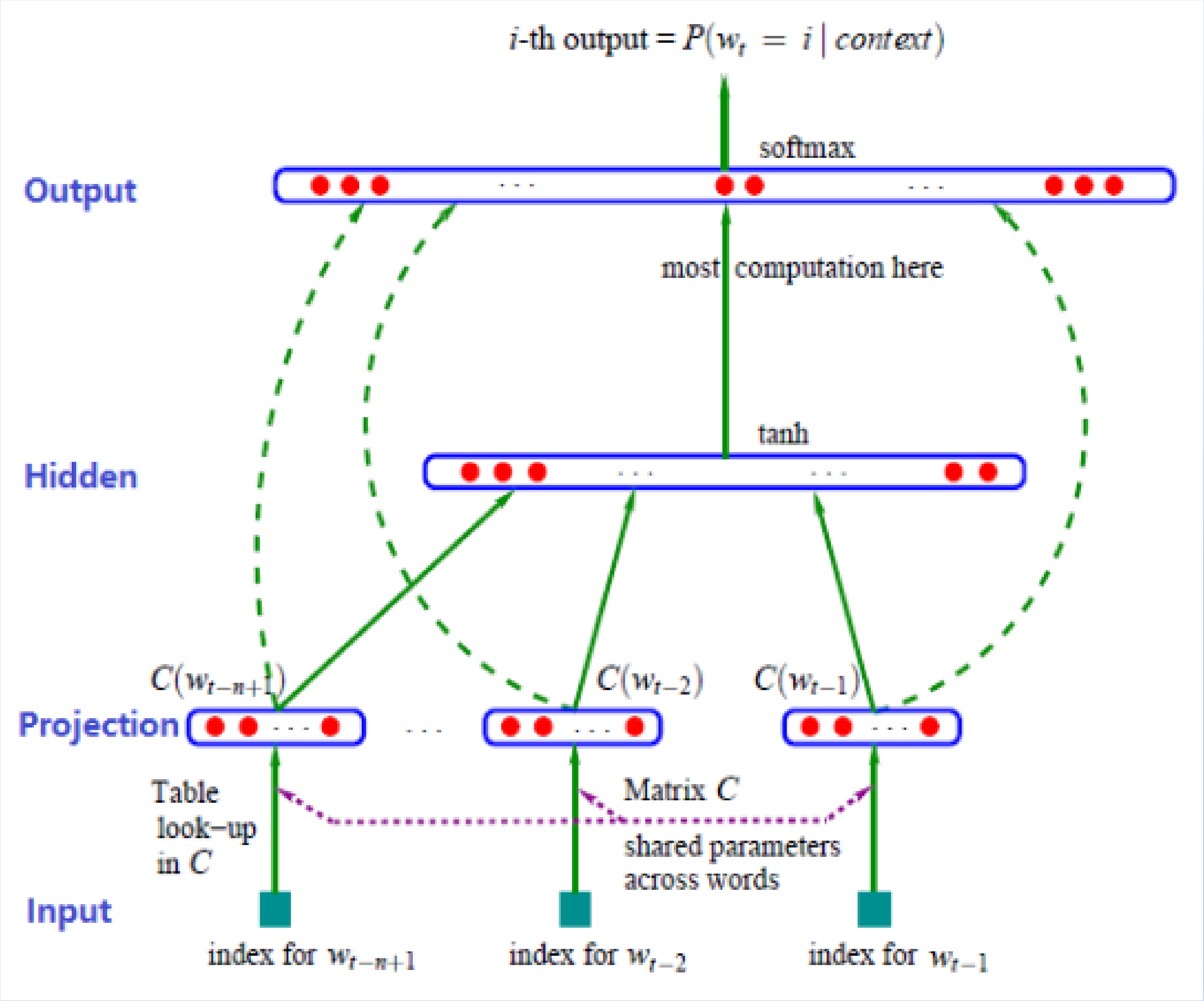
$$y = b + Wx + U\tanh(d + Hx)$$

$$\hat{P}(w_t | w_{t-n+1}^{t-1}; \theta) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}}$$

- Objective function :

$$L = \frac{1}{T} \sum_t log\hat{P}(w_t | w_{t-n+1}^{t-1}; \theta) + R(\theta)$$

(L^2)



NNLM(3)

- BP: $y = b + U\tanh(d + Hx)$

  Let $\boldsymbol{o} = \boldsymbol{d} + \boldsymbol{Hx}, \boldsymbol{a} = \boldsymbol{\tanh(o)}$

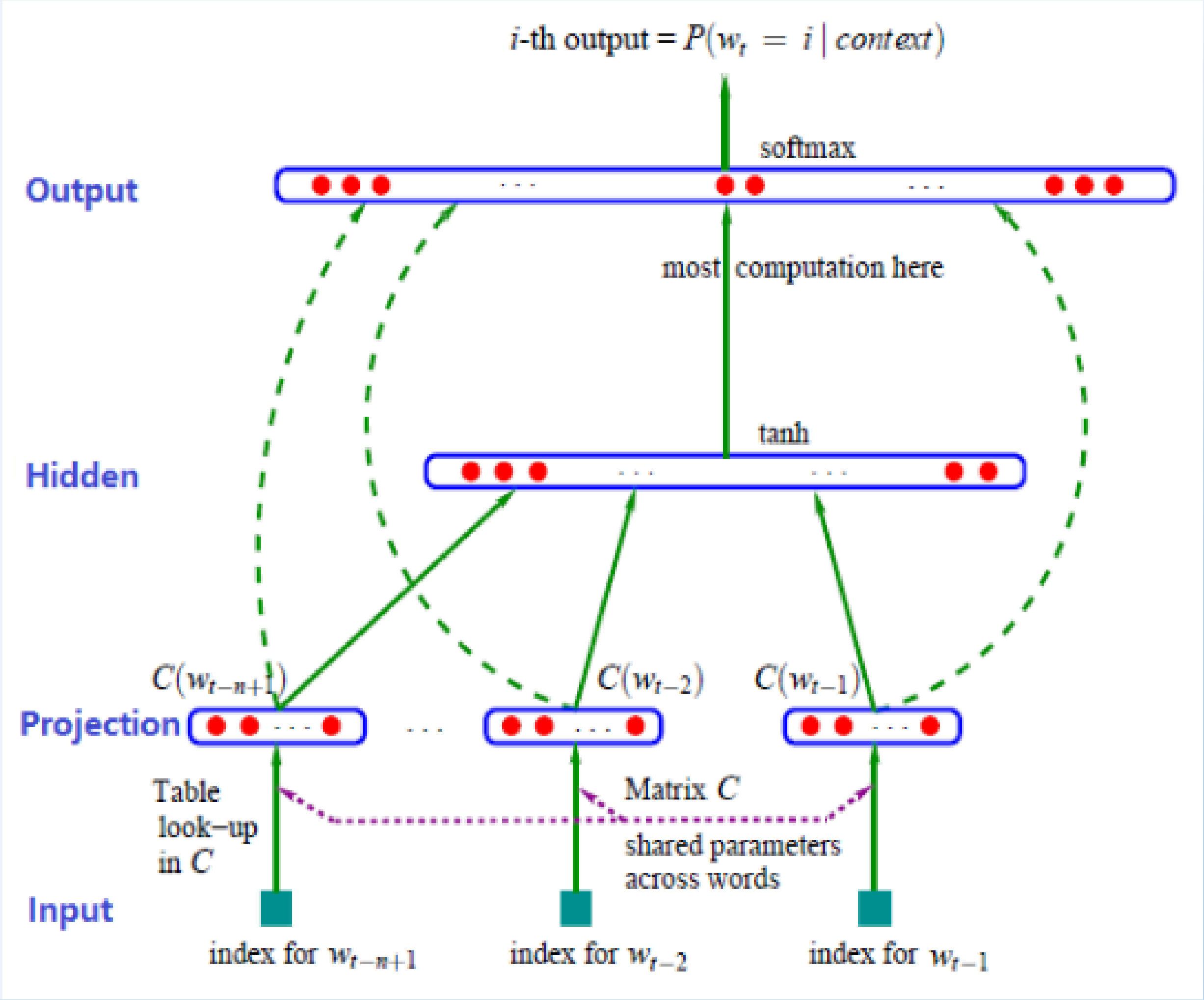
- i in V

$$\frac{\partial L}{\partial y_i} \leftarrow 1_{i==w_t} - p_i$$

$$b_i \leftarrow b_i + \varepsilon \frac{\partial L}{\partial y_i}$$

$$\frac{\partial L}{\partial a} \leftarrow \frac{\partial L}{\partial a} + \frac{\partial L}{\partial y_i} U_i$$

$$U_i \leftarrow U_i + \varepsilon \frac{\partial L}{\partial y_i} a$$



NNLM(4)

- BP: $y = b + U tanh(d + Hx)$

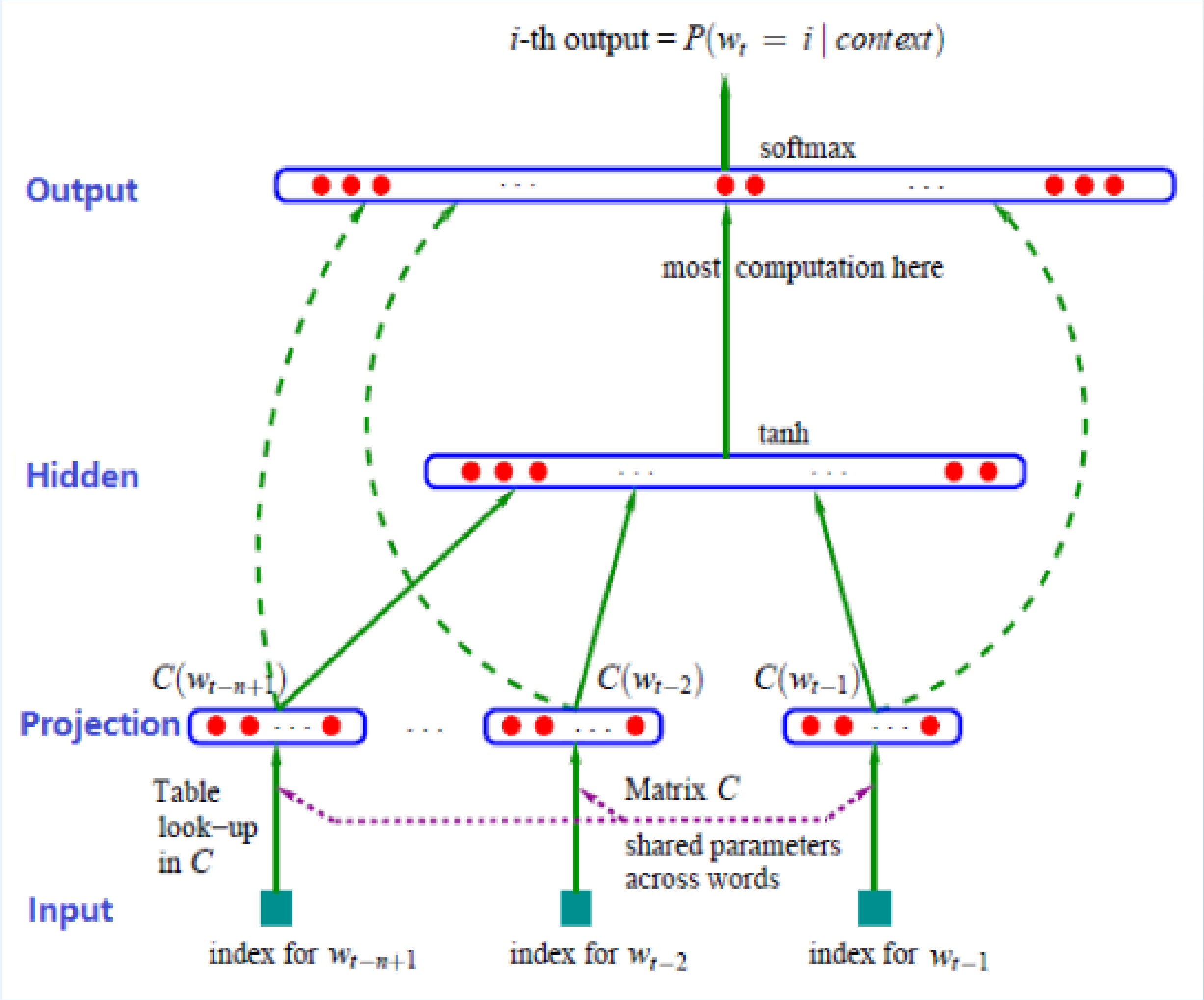
  Let $\boldsymbol{o} = \boldsymbol{d} + \boldsymbol{Hx}, \boldsymbol{a} = \boldsymbol{tanh}(\boldsymbol{o})$
- k=0..h-1

$$\frac{\partial L}{\partial o_k} \leftarrow (1 - a_k^2) \frac{\partial L}{\partial a_k}$$

$$\frac{\partial L}{\partial x} \leftarrow H^T \frac{\partial L}{\partial o}$$

$$d \leftarrow d + \varepsilon \frac{\partial L}{\partial o}$$

$$H \leftarrow H + \varepsilon \frac{\partial L}{\partial o} x^T$$



NNLM(5)

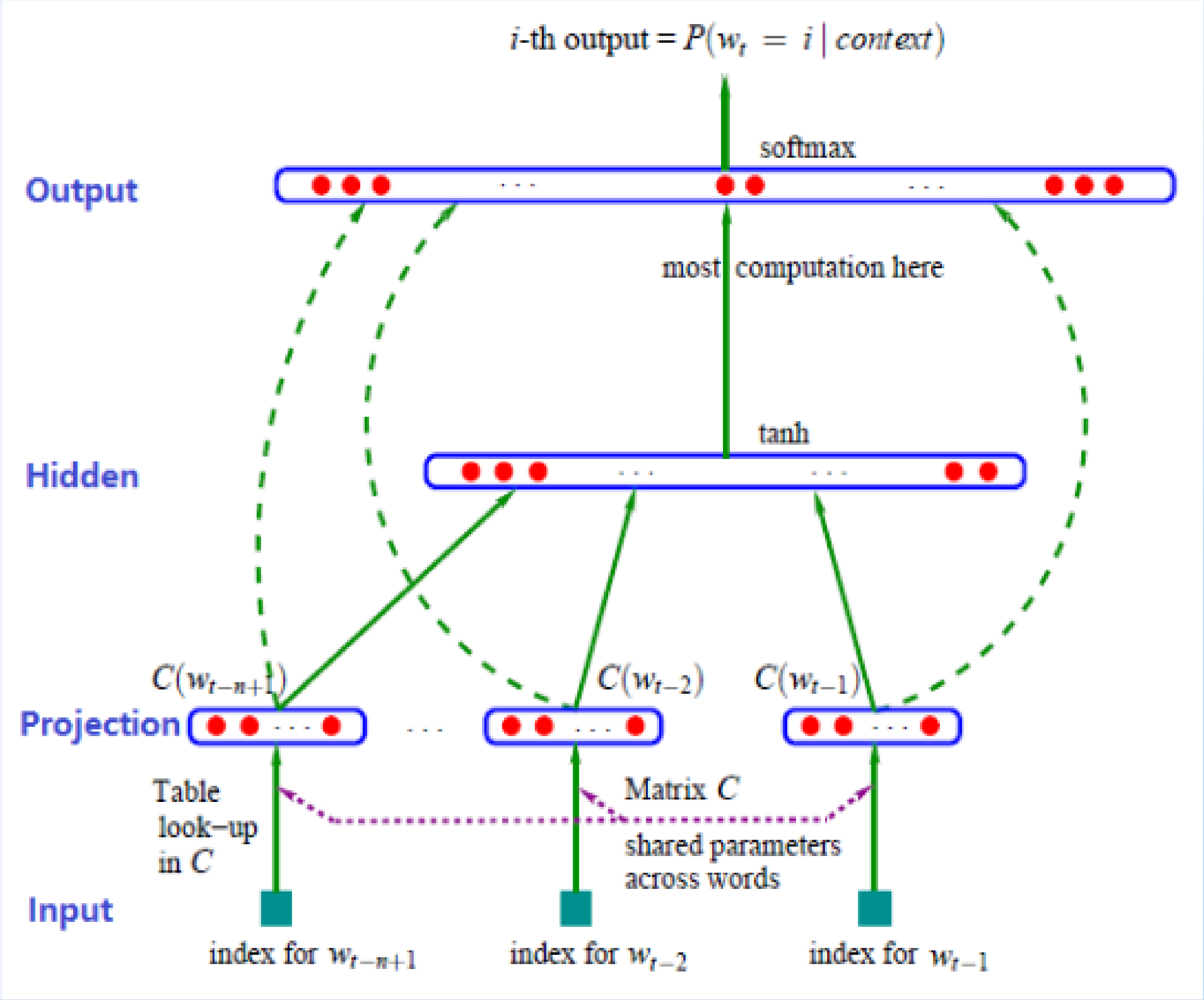
- BP: $y = b + U tanh(d + Hx)$

  $\boldsymbol{Let\ o = d + Hx, a = tanh(o)}$

- k=1..n-1

$$C(w_{t-k}) \leftarrow C(w_{t-k}) + \varepsilon \frac{\partial L}{\partial x(k)}$$

$\frac{\partial L}{\partial x(k)}$ is the k-th block(of length m) of the vector $\frac{\partial L}{\partial x}$



RNNLM

- Major drawbacks of NNLM:
- The model ‘see’ only 5-10 previous words
- Need to specify the length of the context before training

RNNLM(2)

$$x(t) = Uw(t) + Ws(t-1)$$

(concatenation)

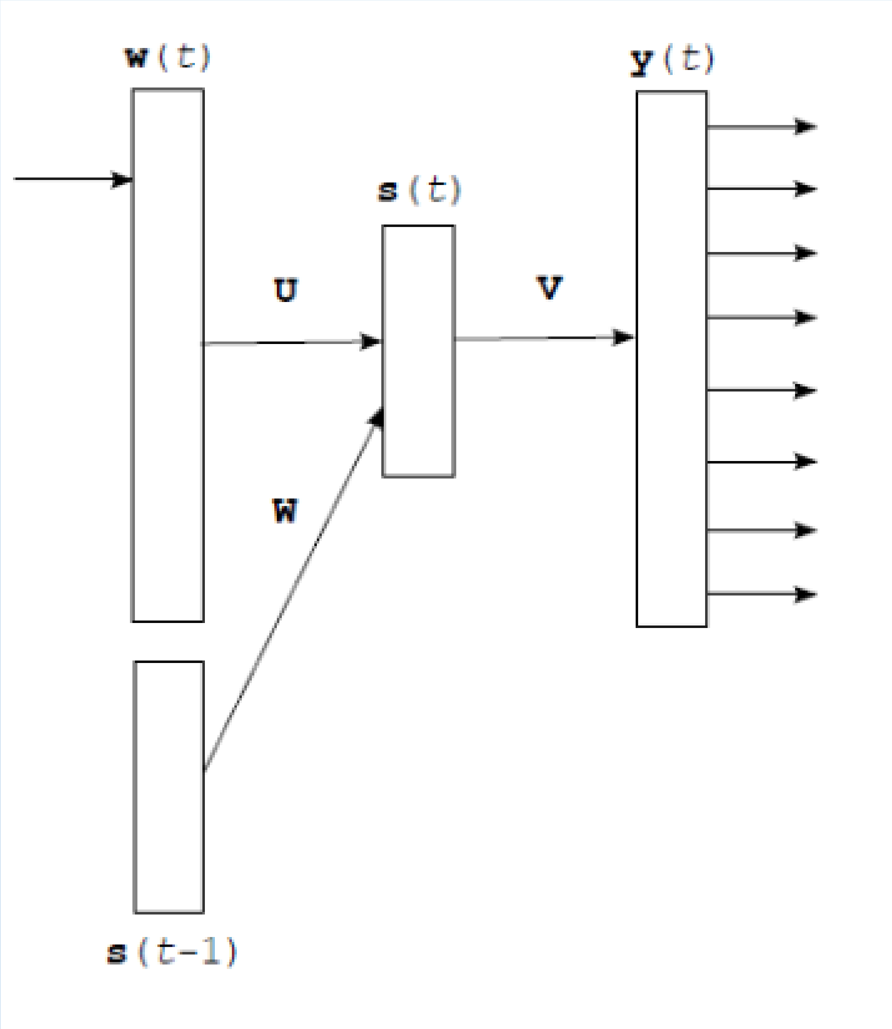
--no projection layer but still projects

$$s_j(t) = f(x(t))$$

(sigmoid)

$$y_k(t) = g(Vs(t))$$

(softmax)

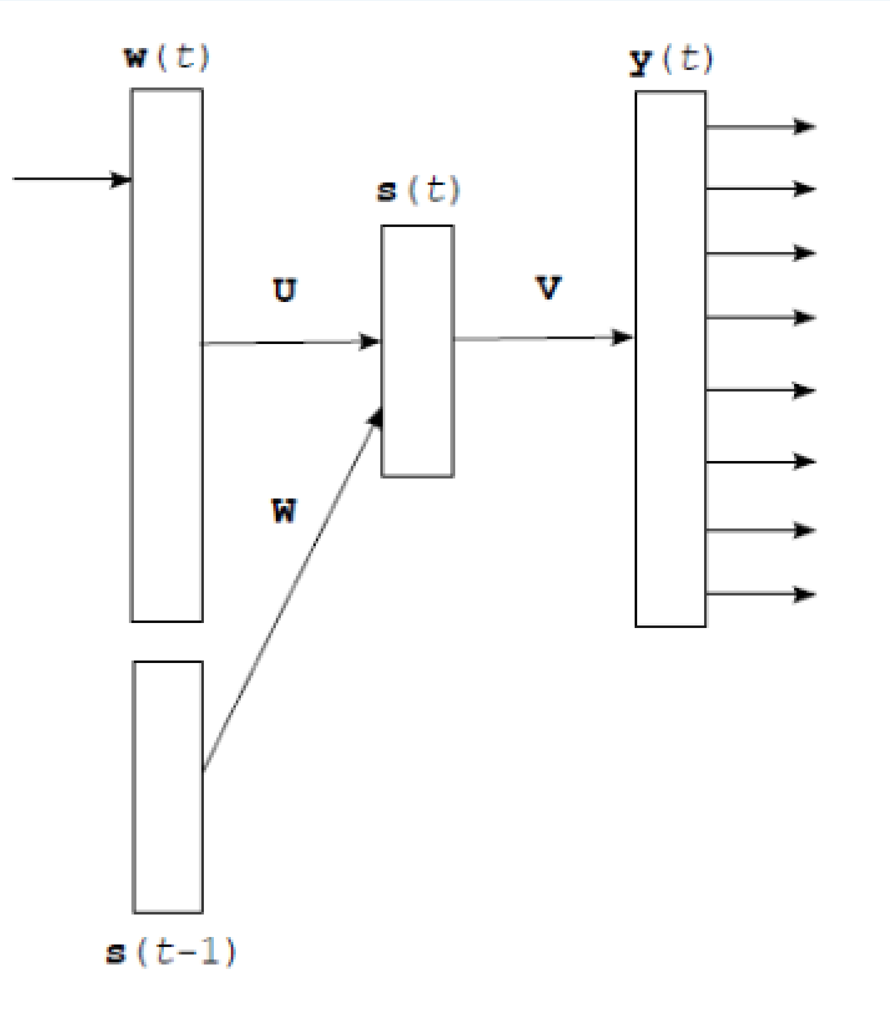


RNNLM(3)

- Train several epochs(usually 5-10)
- Error： $\mathbf{e}_o(t) = \mathbf{d}(t) - \mathbf{y}(t)$

  d(t): 1-of-V coding

$$\mathbf{V}(t+1) = \mathbf{V}(t) + \mathbf{s}(t)\mathbf{e}_o(t)^T \alpha,$$



RNNLM(4)

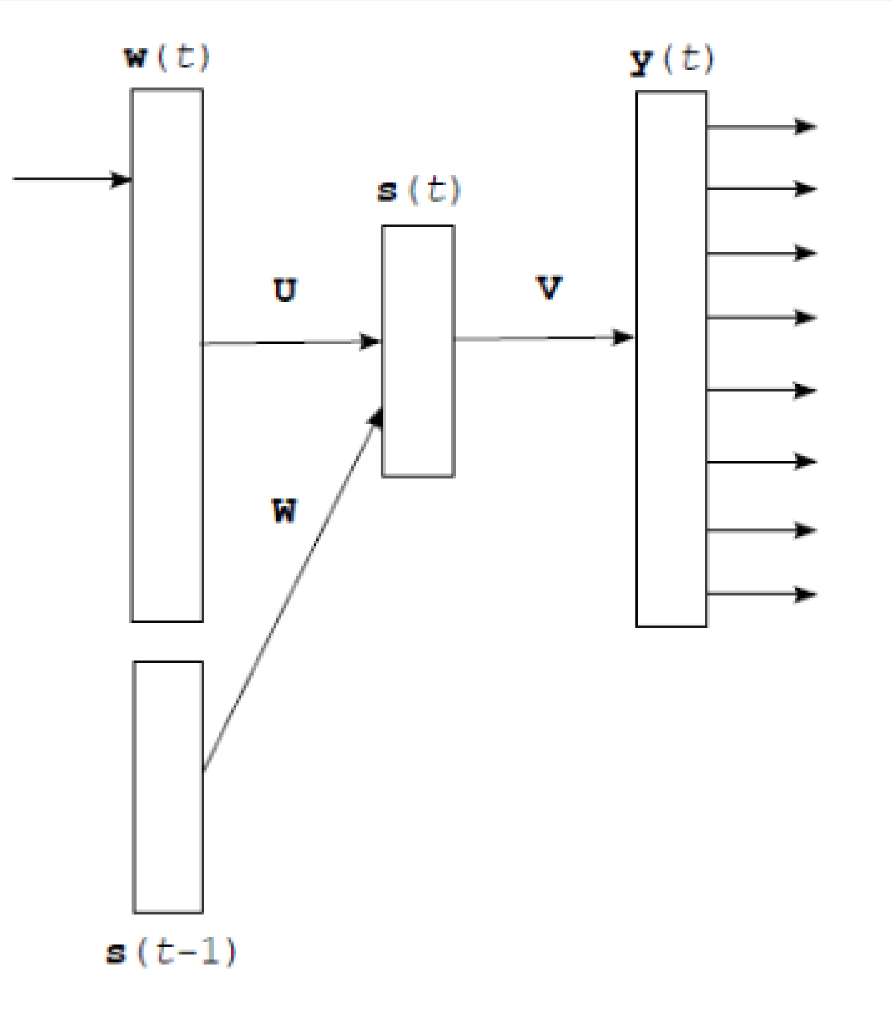
- Gradients of errors propagates from output layer to hidden layer

$$\mathbf{e}_h(t) = d_h\left(\mathbf{e}_o(t)^T\mathbf{V}, t\right),$$

- Where $d_h()$ is :

$$d_{hj}(x, t) = x s_j(t)(1 - s_j(t)).$$

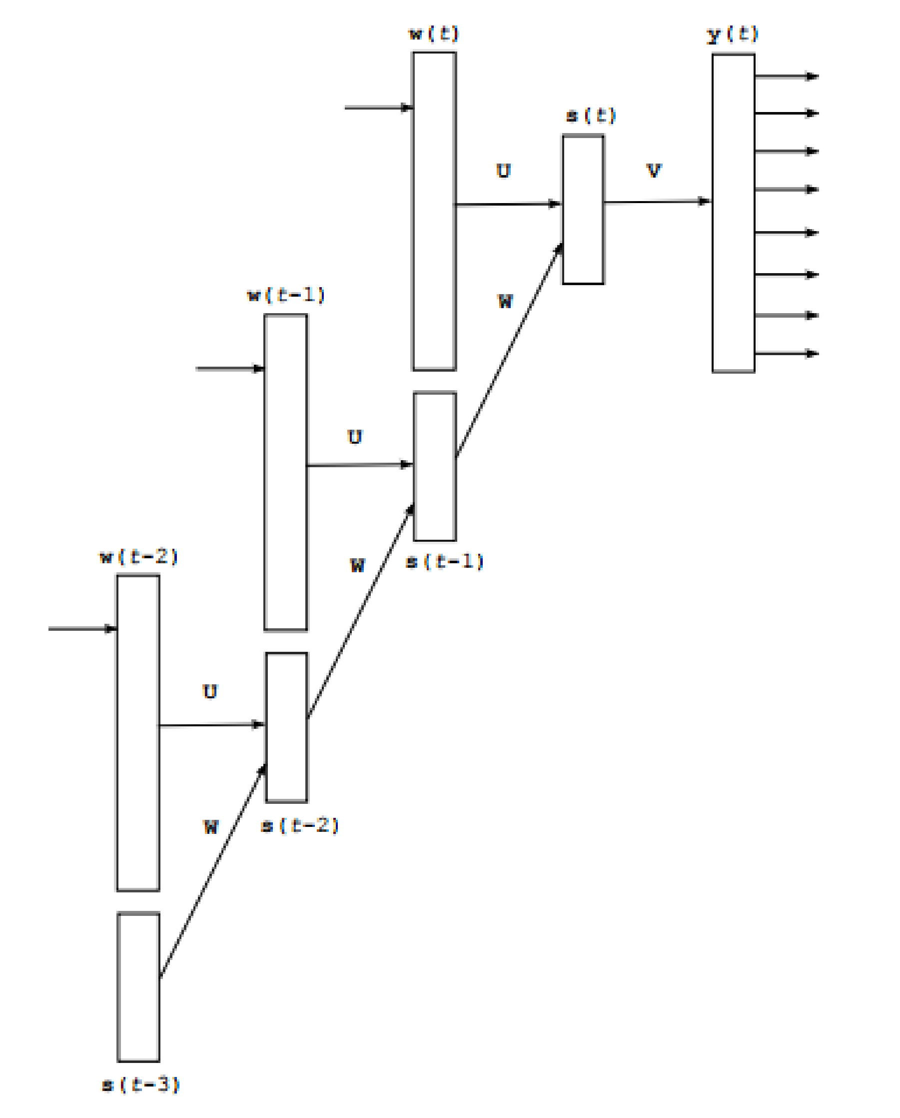
$$\mathbf{U}(t{+}1) = \mathbf{U}(t) + \mathbf{w}(t)\mathbf{e}_h(t)^T\alpha.$$



RNNLM(5)

$$\mathbf{e}_h(t-\tau-1) = d_h\left(\mathbf{e}_h(t-\tau)^T\mathbf{W}, t-\tau-1\right).$$

$$\mathbf{W}(t+1) = \mathbf{W}(t) + \sum_{z=0}^{T} \mathbf{s}(t-z-1)\mathbf{e}_h(t-z)^T\alpha.$$



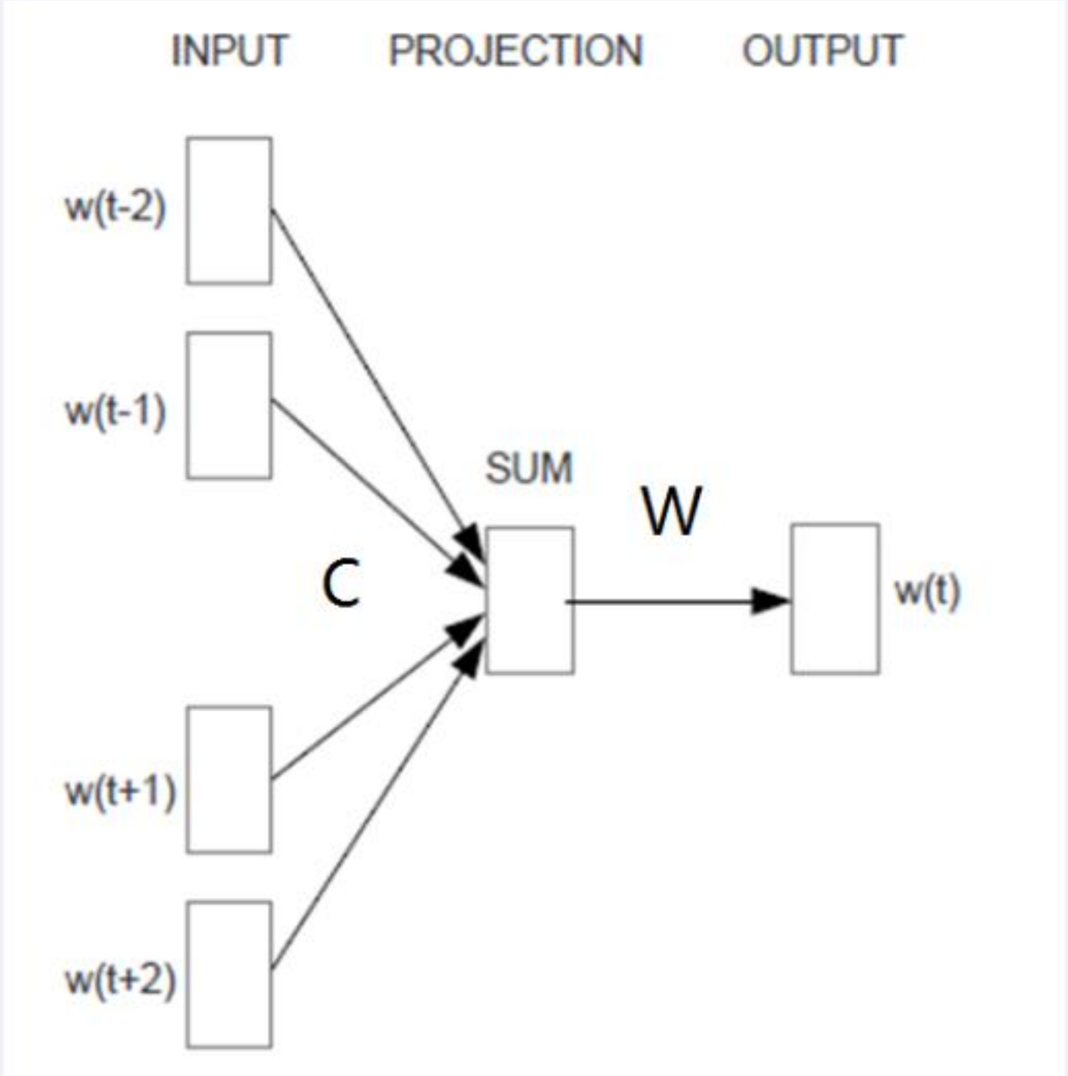
- NNLM and RNNLM are computationally expensive
- Define complexity as number of parameters
- Per training example(per training step):

NNLM: $Q = N \times D + N \times D \times H + H \times log_2(V)$

RNNLM: $Q = H \times H + H \times log_2(V)$

CBOW

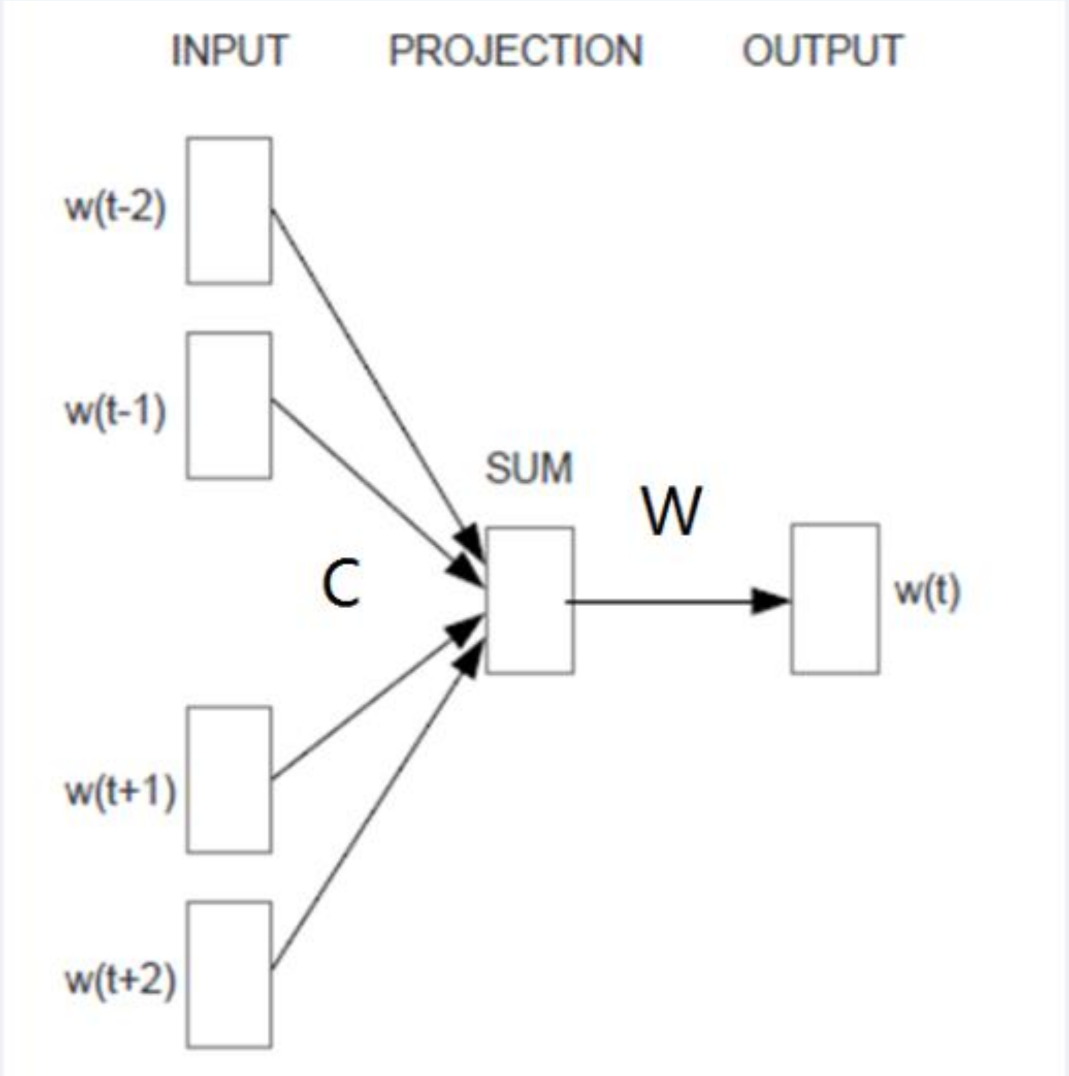
- No hidden layer
- Shared C
- $Q = N \times D + D \times V$



CBOW(2)

- $h = \frac{1}{N} C^T (w_1 + \cdots + w_N)$
- $u_j = W_j h$
- softmax

$$p(w_j|w_I) = y_j = \frac{\exp(u_j)}{\sum_{j'=1}^{V} \exp(u_{j'})},$$



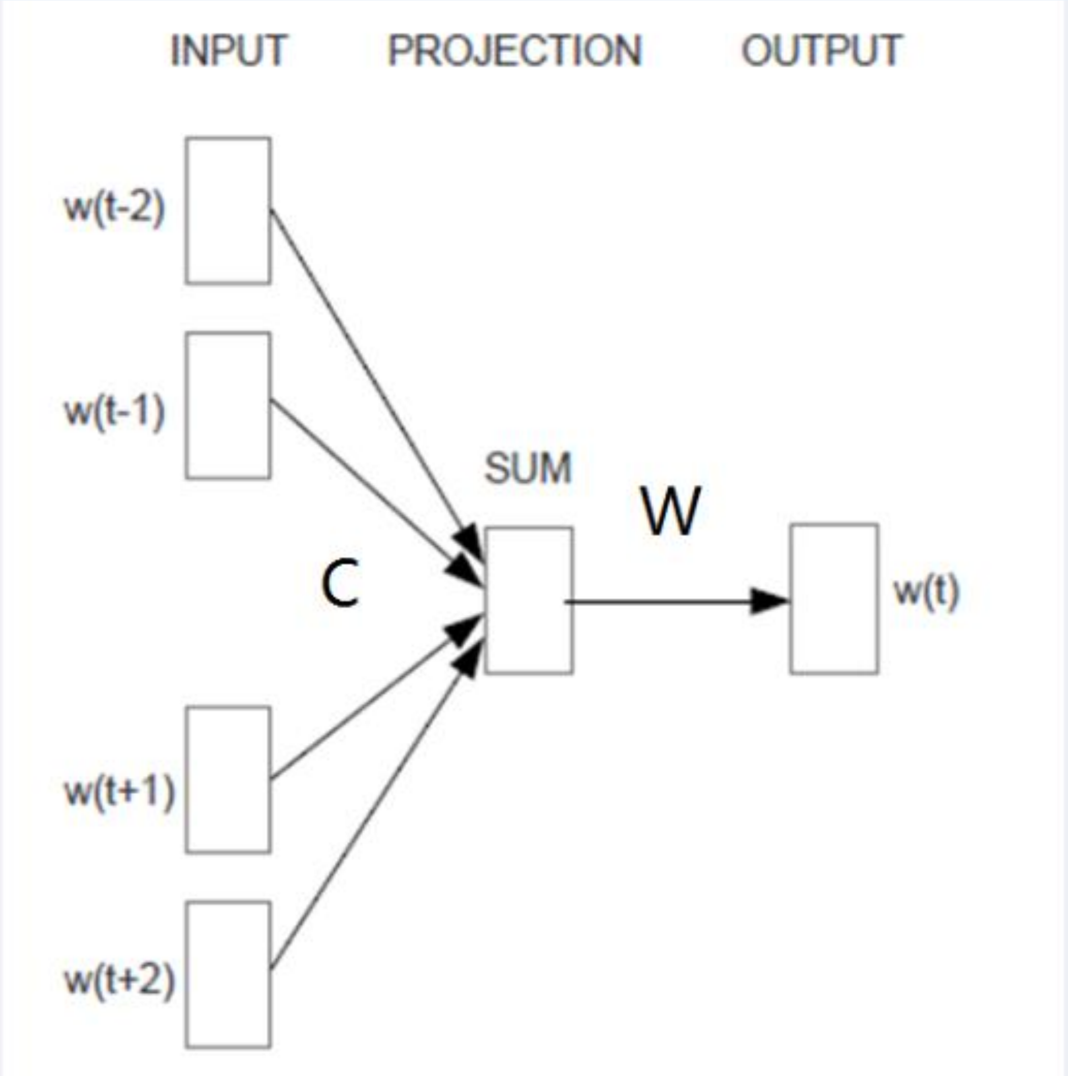
CBOW(3)

- BP
- $E = -\log p(w_O | w_{I,1}, \dots, w_{I,N})$

$$\frac{\partial E}{\partial u_j} = y_j - {y'}_j := e_j$$

$$\frac{\partial E}{\partial w'_{ij}} = \frac{\partial E}{\partial u_j} \cdot \frac{\partial u_j}{\partial w'_{ij}} = e_j \cdot h_i$$

$$\frac{\partial E}{\partial h_i} = \sum_{j=1}^{V} \frac{\partial E}{\partial u_j} \cdot \frac{\partial u_j}{\partial h_i} = \sum_{j=1}^{V} e_j \cdot w'_{ij} := \mathrm{EH}_i$$

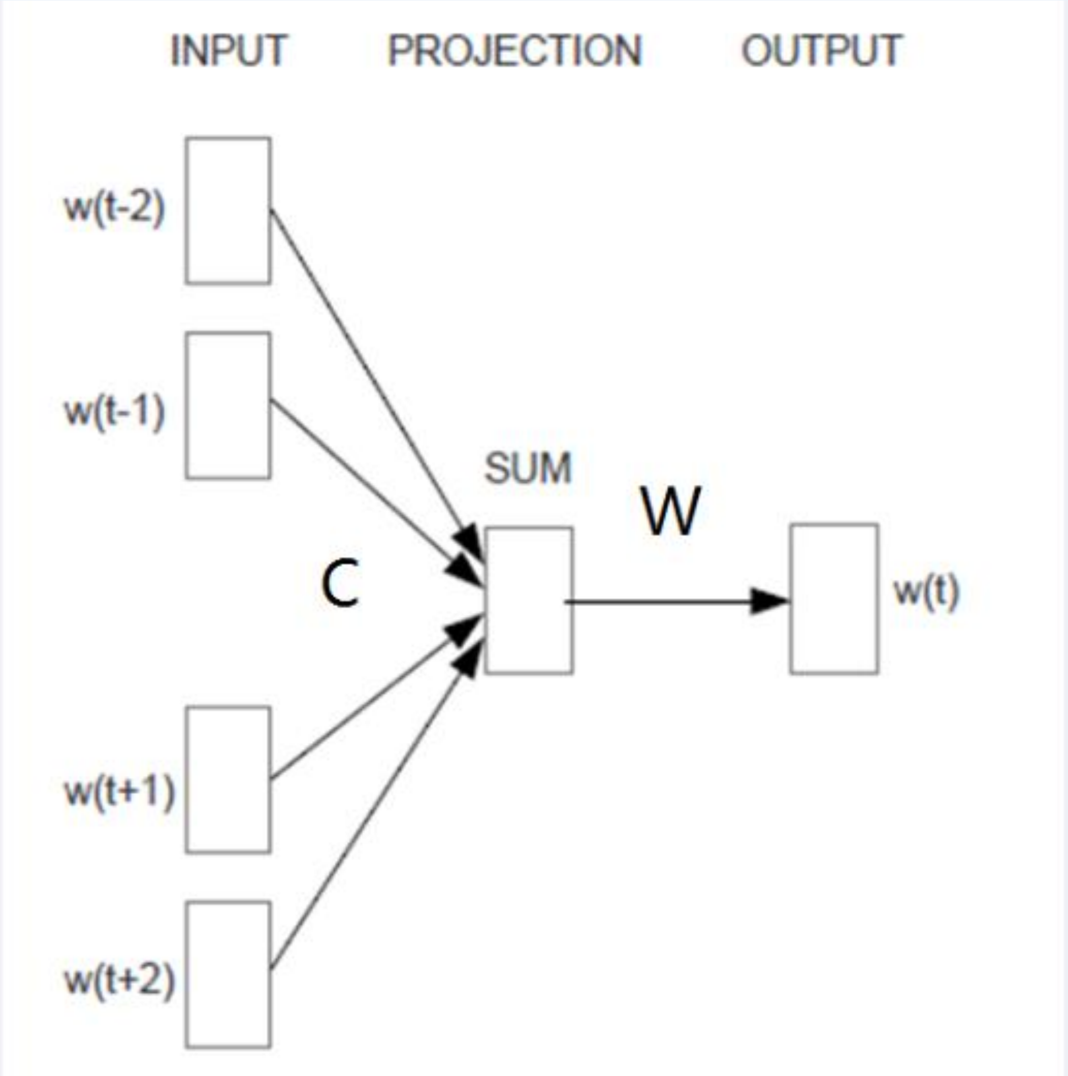


CBOW(4)

- BP
- X: 1-of-V coded input

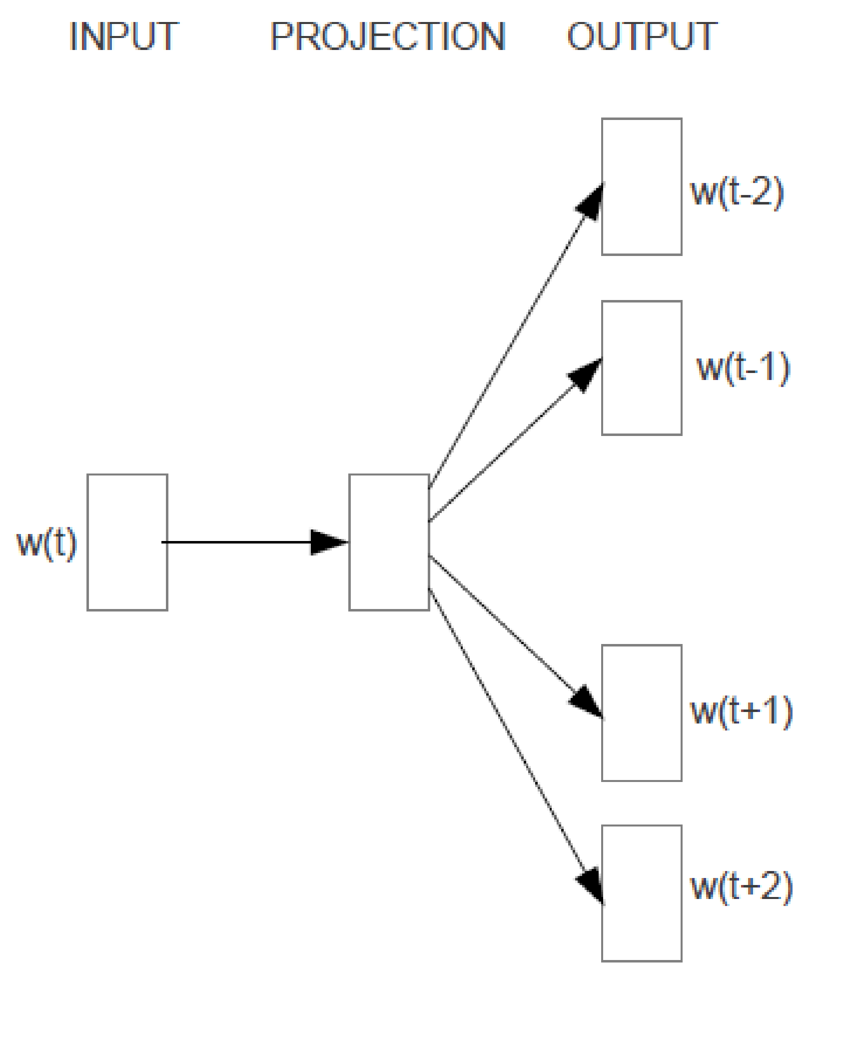
$$\frac{\partial E}{\partial h_i} = \sum_{j=1}^{V} \frac{\partial E}{\partial u_j} \cdot \frac{\partial u_j}{\partial h_i} = \sum_{j=1}^{V} e_j \cdot w'_{ij} := \mathrm{EH}_i$$

$$\mathbf{v}_{w_I,n}^{(\text{new})} = \mathbf{v}_{w_I,n}^{(\text{old})} - \frac{1}{N} \cdot \eta \cdot \mathrm{EH}_i^T$$



Skip-gram

- Use current word to predict context
- N multinomial distributions
- $Q = C \times (D + D \times V)$

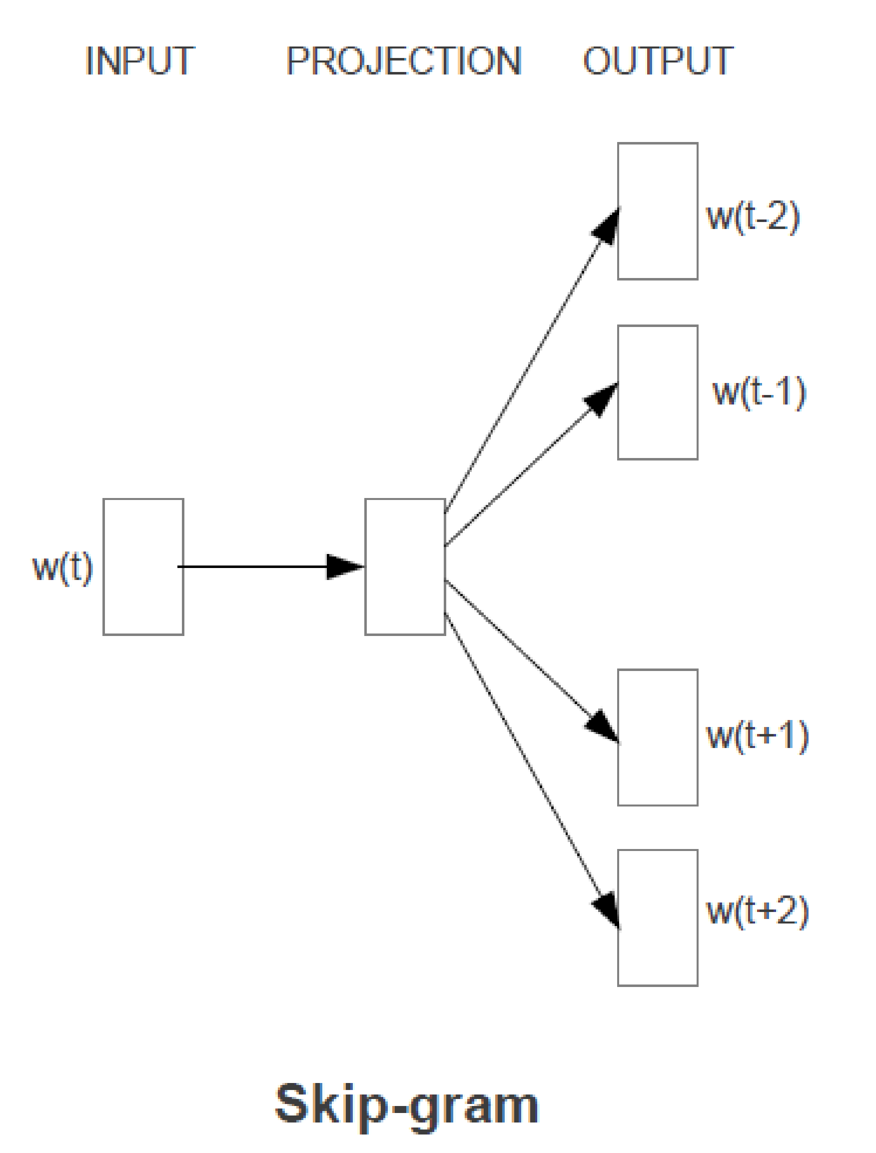


Skip-gram

Skip-gram(2)

- $p(w_{n,j} = w_O | w_I) = y_{n,j} = \frac{\exp(u_{n,j})}{\sum_{j'=1}^{V} \exp(u_{j'})}$

$$\begin{aligned} E &= -\log p(w_{O,1}, w_{O,2}, \cdots, w_{O,C} | w_I) \\ &= -\log \prod_{c=1}^{C} \frac{\exp(u_{c,j_c^*})}{\sum_{j'=1}^{V} \exp(u_{j'})} \\ &= -\sum_{c=1}^{C} u_{j_c^*} + C \cdot \log \sum_{j'=1}^{V} \exp(u_{j'}) \end{aligned}$$



Skip-gram(3)

- Errors over all context words

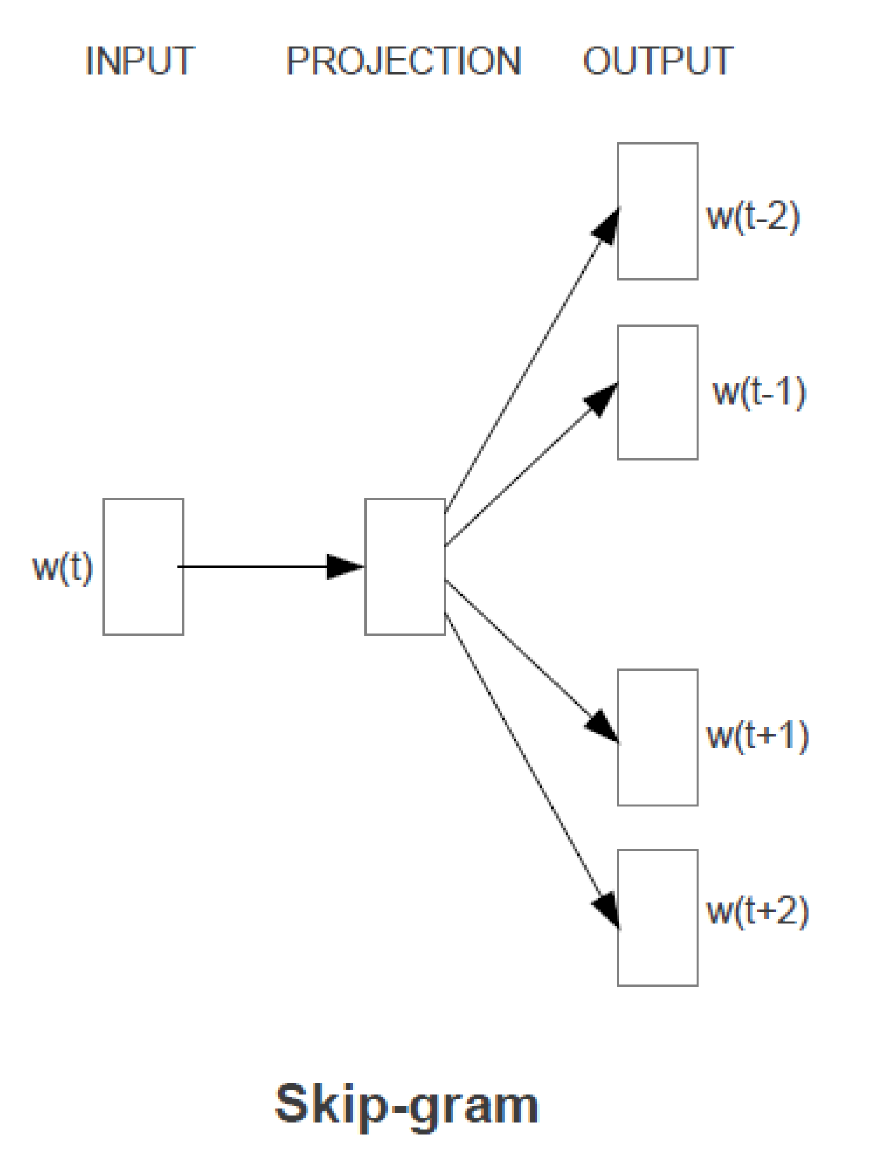
$$\mathrm{EI}_j = \sum_{c=1}^{C} e_{c,j}$$

$$\frac{\partial E}{\partial w'_{ij}} = \sum_{c=1}^{C} \frac{\partial E}{\partial u_{c,j}} \cdot \frac{\partial u_{c,j}}{\partial w'_{ij}} = \mathrm{EI}_j \cdot h_i$$

- Input to projection matrix

$$\mathrm{EH}_i = \sum_{j=1}^{V} \mathrm{EI}_j \cdot w'_{ij}.$$

$$\mathbf{v}_{w_I}^{(\text{new})} = \mathbf{v}_{w_I}^{(\text{old})} - \eta \cdot \mathrm{EH}_i^T$$



Optimize the computation

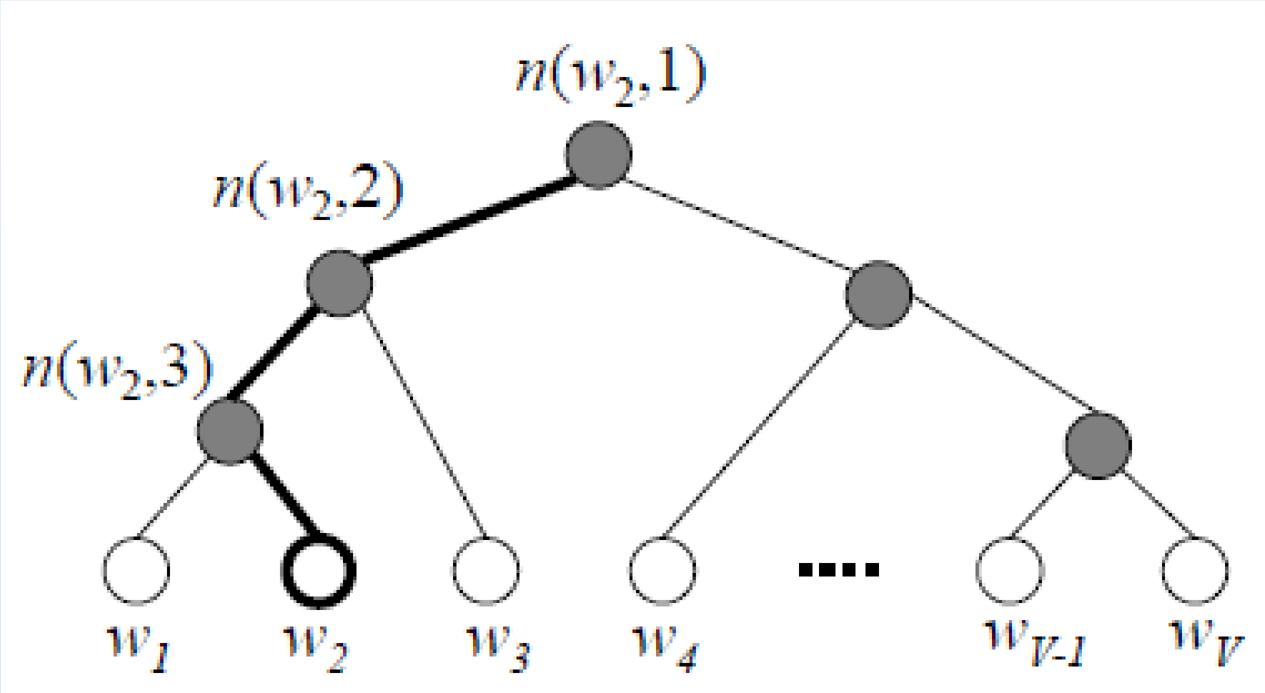
- **Hierarchical softmax**
- Define:

$ch(x)$ (left child)

$n(w, j)$the jth node to w

$$[\![x]\!] = \begin{cases} 1 & \text{if } x \text{ is true;} \\ -1 & \text{otherwise.} \end{cases}$$

v'_j ouput vector of inner node j



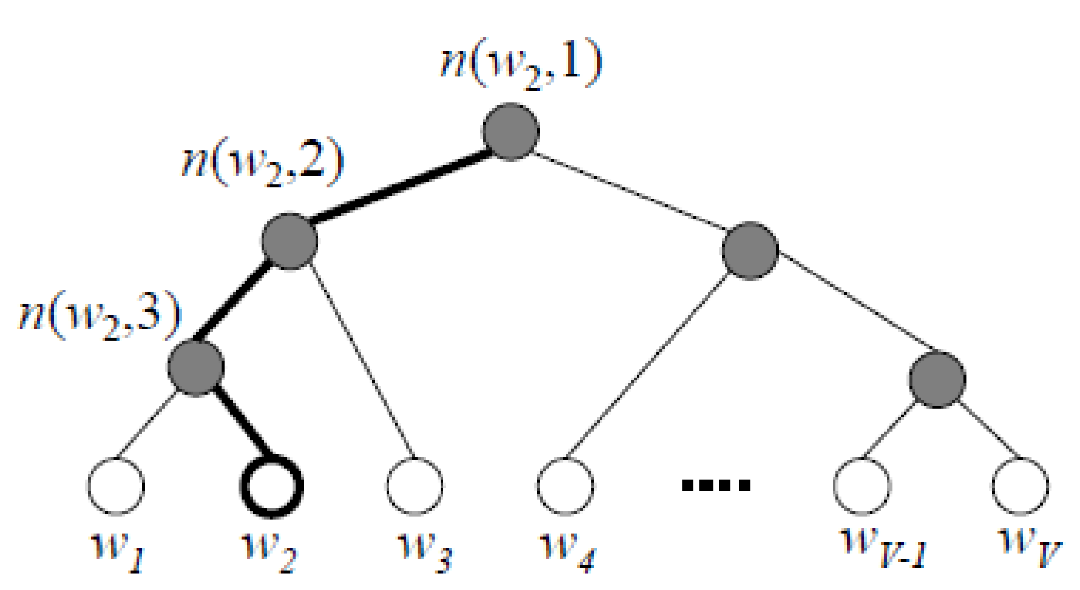
Methods to reduce the complexity

- **Hierarchical softmax**

$$p(w = w_O) =$$

$$\prod_{j=1}^{L(w)-1} \sigma\left([\![n(w, j+1) = \text{ch}(n(w,j))]\!] \cdot {\mathbf{v}'_{n(w,j)}}^T \mathbf{h} \right)$$

$$E = -\log p(w = w_O | w_I)$$



Methods to reduce the complexity

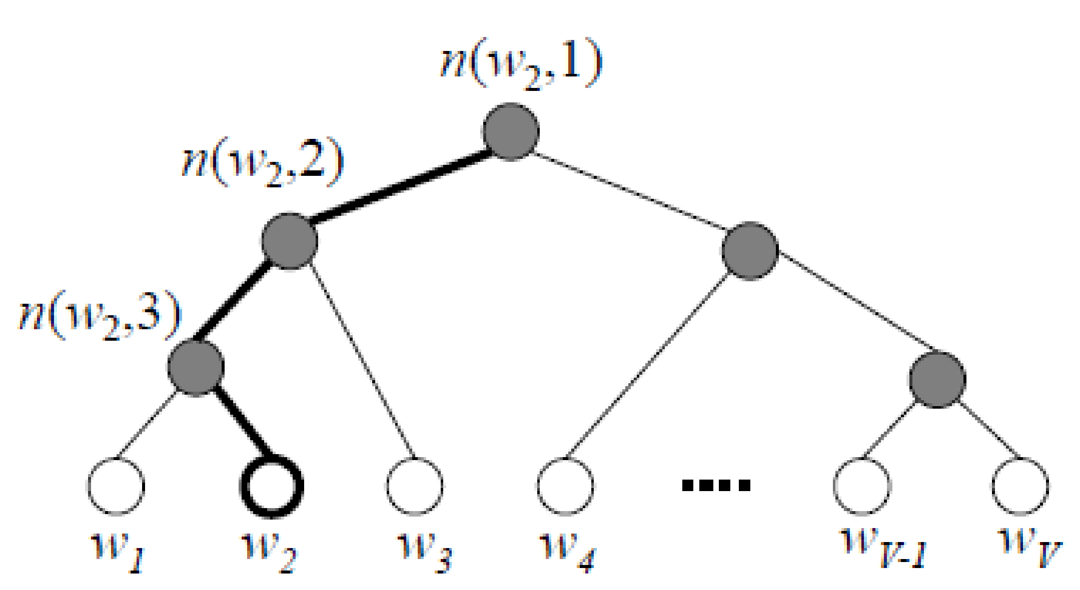
- **Hierarchical softmax**

$$\frac{\partial E}{\partial \mathbf{v}'_j \mathbf{h}} = \left(\sigma([\![\cdot]\!]\mathbf{v}'^T_j \mathbf{h}) - 1\right)[\![\cdot]\!]$$

$$= \begin{cases} \sigma(\mathbf{v}'^T_j \mathbf{h}) - 1 & ([\![\cdot]\!] = 1) \\ \sigma(\mathbf{v}'^T_j \mathbf{h}) & ([\![\cdot]\!] = -1) \end{cases}$$

$$= \sigma(\mathbf{v}'^T_j \mathbf{h}) - t_j$$

$$\frac{\partial E}{\partial \mathbf{v}'_j} = \frac{\partial E}{\partial \mathbf{v}'_j \mathbf{h}} \cdot \frac{\partial \mathbf{v}'_j \mathbf{h}}{\partial \mathbf{v}'_j} = \left(\sigma(\mathbf{v}'^T_j \mathbf{h}) - t_j\right) \cdot \mathbf{h}$$



Methods to reduce the complexity

- **Negative sampling**

$$E = -\log \sigma({\mathbf{v}'_{w_O}}^T \mathbf{h}) - \sum_{w_j \in \mathcal{W}_{\text{neg}}} \log \sigma(-{\mathbf{v}'_{w_j}}^T \mathbf{h})$$

$$\begin{aligned} \frac{\partial E}{\partial {\mathbf{v}'_{w_j}}^T \mathbf{h}} &= \begin{cases} \sigma({\mathbf{v}'_{w_j}}^T \mathbf{h}) - 1 & \text{if } w_j = w_O \\ \sigma({\mathbf{v}'_{w_j}}^T \mathbf{h}) & \text{if } w_j \in \mathcal{W}_{\text{neg}} \end{cases} \\ &= \sigma({\mathbf{v}'_{w_j}}^T \mathbf{h}) - t_j \end{aligned}$$

$$\begin{aligned} \frac{\partial E}{\partial \mathbf{h}} &= \sum_{w_j \in \{w_O\} \cup \mathcal{W}_{\text{neg}}} \frac{\partial E}{\partial {\mathbf{v}'_{w_j}}^T \mathbf{h}} \cdot \frac{\partial {\mathbf{v}'_{w_j}}^T \mathbf{h}}{\partial \mathbf{h}} \\ &= \sum_{w_j \in \{w_O\} \cup \mathcal{W}_{\text{neg}}} \left(\sigma({\mathbf{v}'_{w_j}}^T \mathbf{h}) - t_j\right) \mathbf{v}'_{w_j} := \text{EH} \end{aligned}$$

Methods to reduce the complexity

- **Negative sampling**
- unigram distribution U(w) raised to the 3/4rd power (i.e., U(w) $U(w)^{3/4}$/Z)

$$len(w) = \frac{[\text{counter}(w)]^{\frac{3}{4}}}{\sum_{u \in \mathcal{D}} [\text{counter}(u)]^{\frac{3}{4}}}.$$