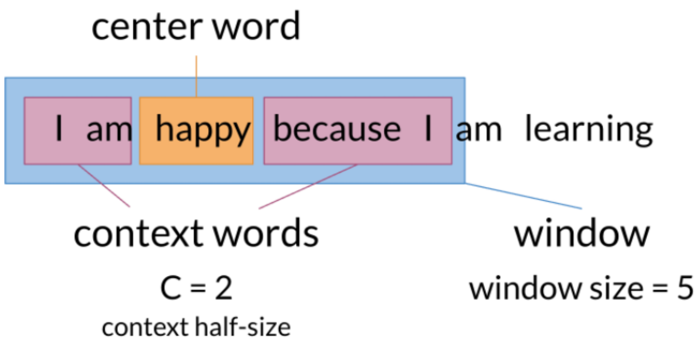
Word2Vec: Continuous Bag of Words (CBOW)

- References:
  - https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture01-wordvecs1-public.pdf
  - https://web.stanford.edu/class/cs224n/readings/cs224n_winter2023_lecture1_notes_draft.pdf

- Continuous Bag of Words (CBOW): Use the outside words (o) to predict the center word (c).

- Skip-gram: Use the **center word (c)** to predict the distribution of **outside words (o)**.

$\{v_1, v_2, \ldots, v_{|V|}\}$: context word Embedding



$$\begin{aligned}
\text{minimize } J &= -\log P(w_c \mid w_{c-m}, \ldots, w_{c-1}, w_{c+1}, \ldots, w_{c+m}) \\
&= -\log P(u_c \mid \hat{v}) \\
&= -\log \frac{\exp(u_c^T \hat{v})}{\sum_{j=1}^{|V|} \exp(u_j^T \hat{v})} \\
&= -u_c^T \hat{v} + \log \sum_{j=1}^{|V|} \exp(u_j^T \hat{v})
\end{aligned}$$

$\hat{v} = \frac{1}{2m} \sum_{j=-m,\, j\neq 0}^{m} v_{w_j}$

$\text{Loss} = \sum_{c \in \text{Corpus}} \left(-u_c^T \hat{v} + \log \sum_{j=1}^{|V|} \exp(u_j^T \hat{v}) \right)$

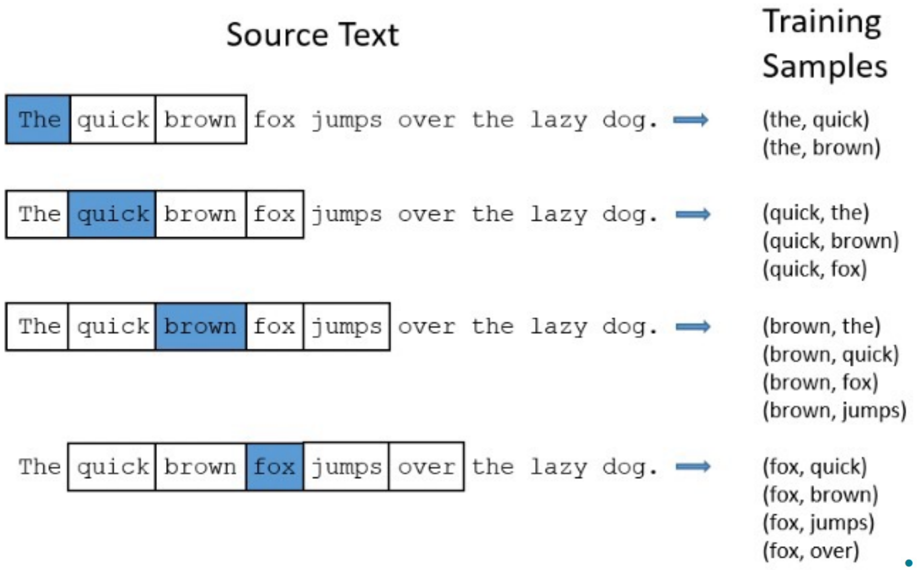
$\{u_1, u_2, \ldots, u_{|V|}\}$ $\quad Pr(W = c \mid \text{context})$

v^* as the word embedding.

SGD/Adam to find (u^*, v^*)

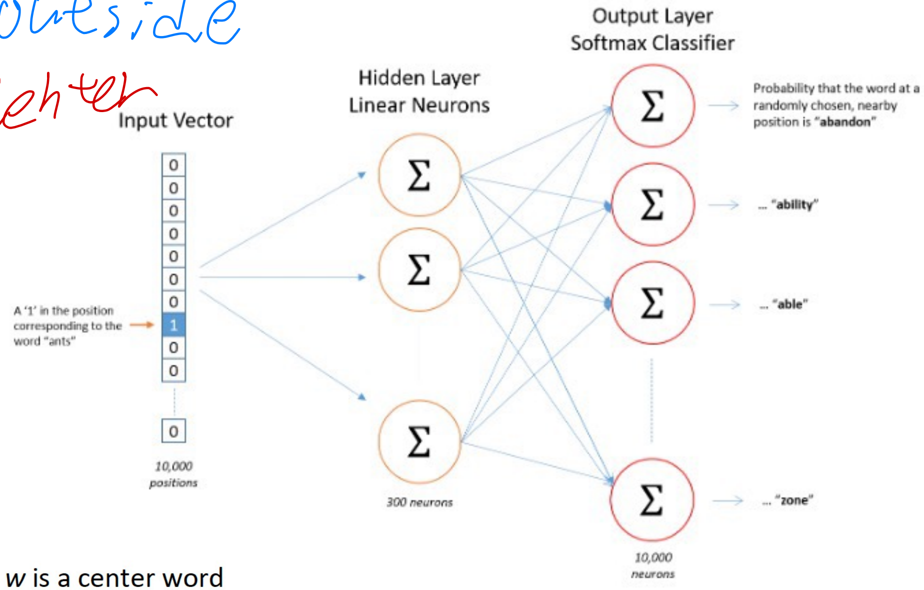
Word2Vec: Skip-Gram

- References:
  - https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture01-wordvecs1-public.pdf
  - https://web.stanford.edu/class/cs224n/readings/cs224n_winter2023_lecture1_notes_draft.pdf



u: outside

v: center



- v_w when w is a center word
- u_w when w is a context word

We want to minimize the objective function:

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\le j\le m \\ j\ne 0}} \log P(w_{t+j} \mid w_t; \theta)$$

Then for a center word c and a context word o:

$$\boxed{P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w\in V}\exp(u_w^T v_c)}}$$

too computational costly

$\sum_{w \in V_{neg}} \exp(u_w^T v_c)$

Negative Sampling

Word2Vec: GloVe

- Model: 2 Embeddings : $U \in \mathbb{R}^{|V| \times d}$, $V \in \mathbb{R}^{|V| \times d}$

$$Q_{ij} = \frac{\exp(\vec{u}_j^T \vec{v}_i)}{\sum_{w=1}^{W} \exp(\vec{u}_w^T \vec{v}_i)}$$

- Adjusted Square loss: $\hat{J} = \sum_{i=1}^{W} \sum_{j=1}^{W} X_i (\hat{P}_{ij} - \hat{Q}_{ij})^2$

where $\hat{P}_{ij} = X_{ij}$ and $\hat{Q}_{ij} = \exp(\vec{u}_j^T \vec{v}_i)$

- Some approximations:

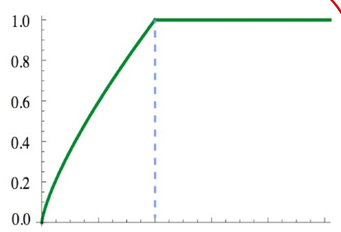
$$\hat{J} = \sum_{i=1}^{W} \sum_{j=1}^{W} X_i (\log(\hat{P})_{ij} - \log(\hat{Q}_{ij}))^2$$

$$= \sum_{i=1}^{W} \sum_{j=1}^{W} X_i (\vec{u}_j^T \vec{v}_i - \log X_{ij})^2$$

Final Loss Function to Minimize:

Loss: $$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right) \left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2$$

- Fast training
- Scalable to huge corpora

$f \sim$ 

Recurrent Neural Network (RNN)

→ is the distribution of $x^{(t+1)}$

Reference: Stanford CS224N, Lecture 5:
https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture05-rnnlm.pdf

$U \in \mathbb{R}^{|V| \times d_h}$

output distribution

$$\hat{\boldsymbol{y}}^{(t)} = \text{softmax}\left(\boldsymbol{U}\boldsymbol{h}^{(t)} + \boldsymbol{b}_2\right) \in \mathbb{R}^{|V|}$$

$= \dfrac{\exp(z_i^{(t)})}{\sum_j \exp(z_j^{(t)})}$

where $z^{(t)} = U \cdot h^{(t)} + b_2$

$\mathbb{R}^{d_h \times d_h}$

$\mathbb{R}^{d_h \times d_e}$

hidden states

$$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_e\boldsymbol{e}^{(t)} + \boldsymbol{b}_1\right)$$

$\boldsymbol{h}^{(0)}$ is the initial hidden state

activation function

elementwise operator

word embeddings

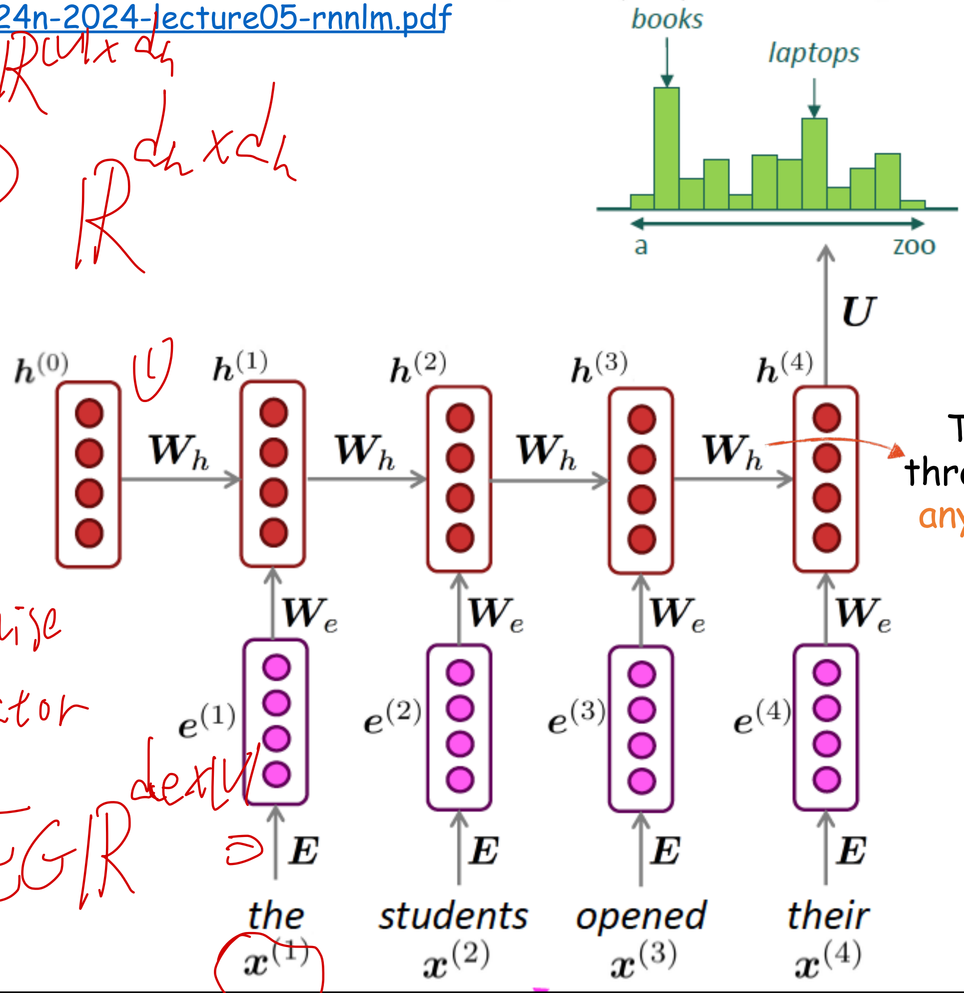
$$\boldsymbol{e}^{(t)} = \boldsymbol{E}\boldsymbol{x}^{(t)}$$

$E \in \mathbb{R}^{d_e \times |V|}$

words / one-hot vectors

$$\boldsymbol{x}^{(t)} \in \mathbb{R}^{|V|}$$

$$\hat{\boldsymbol{y}}^{(4)} = P(\boldsymbol{x}^{(5)} | \text{the students opened their})$$



The same W_h is applied throughout, so it can handle any input sequence length.

Training RNN

Reference: Stanford CS224N, Lecture 5:
https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture05-rnnlm.pdf

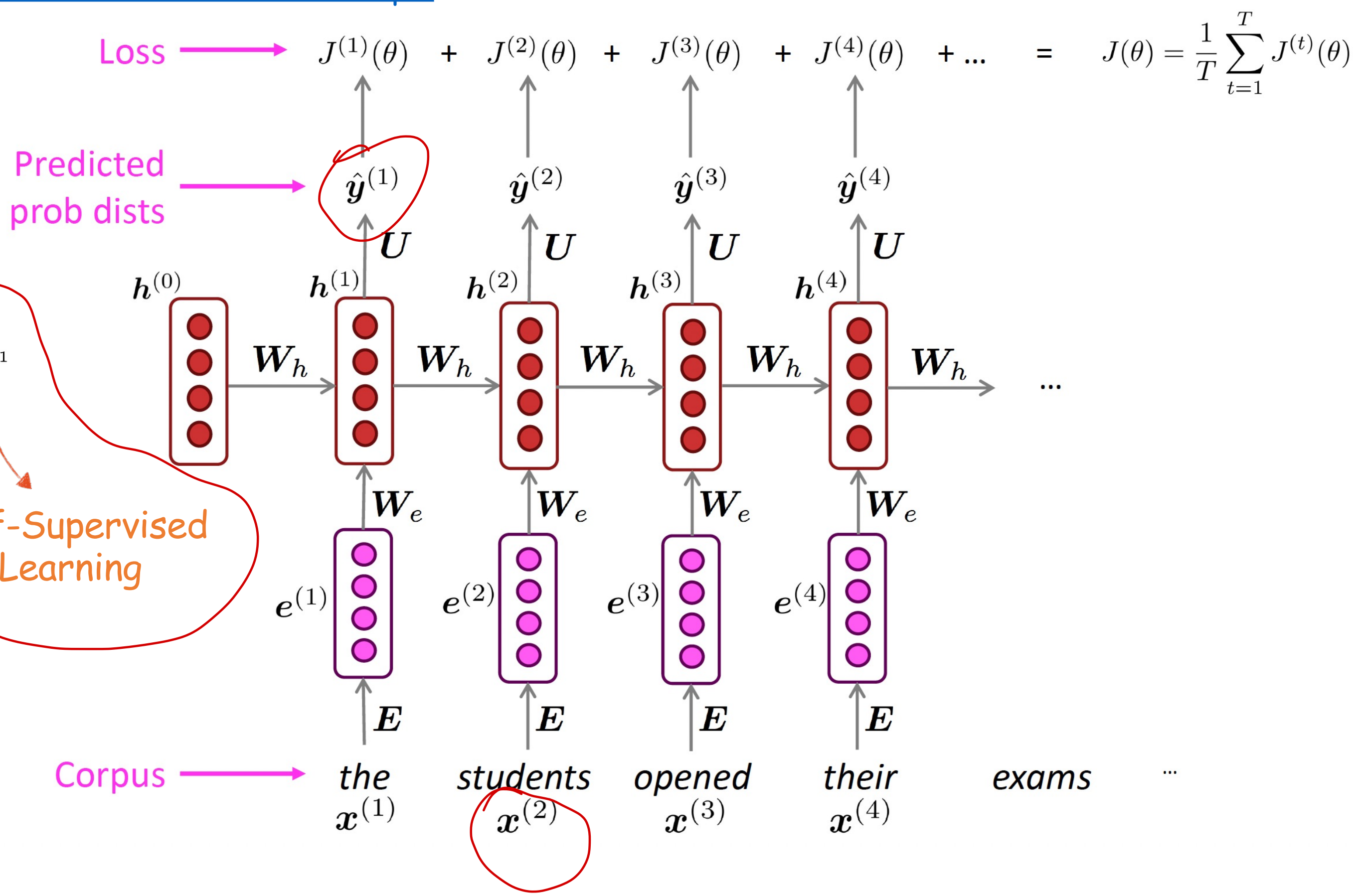
- Loss functions in step t is the cross-entropy between the true 1-hot and the predicted distribution:

$$J^{(t)}(\theta) = CE(\boldsymbol{y}^{(t)}, \hat{\boldsymbol{y}}^{(t)}) = -\sum_{w \in V} \boldsymbol{y}_w^{(t)} \log \hat{\boldsymbol{y}}_w^{(t)} = -\log \hat{\boldsymbol{y}}_{\boldsymbol{x}_{t+1}}^{(t)}$$

Self-Supervised Learning

- So, the overall loss for the entire training corpus is:

$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T} J^{(t)}(\theta) = \frac{1}{T}\sum_{t=1}^{T} -\log \hat{\boldsymbol{y}}_{\boldsymbol{x}_{t+1}}^{(t)}$$



Vanishing (and Exploding) Gradient in RNN

Reference: Stanford CS224N, Lecture 5:
https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture05-rnnlm.pdf

Backpropagation through time

loss

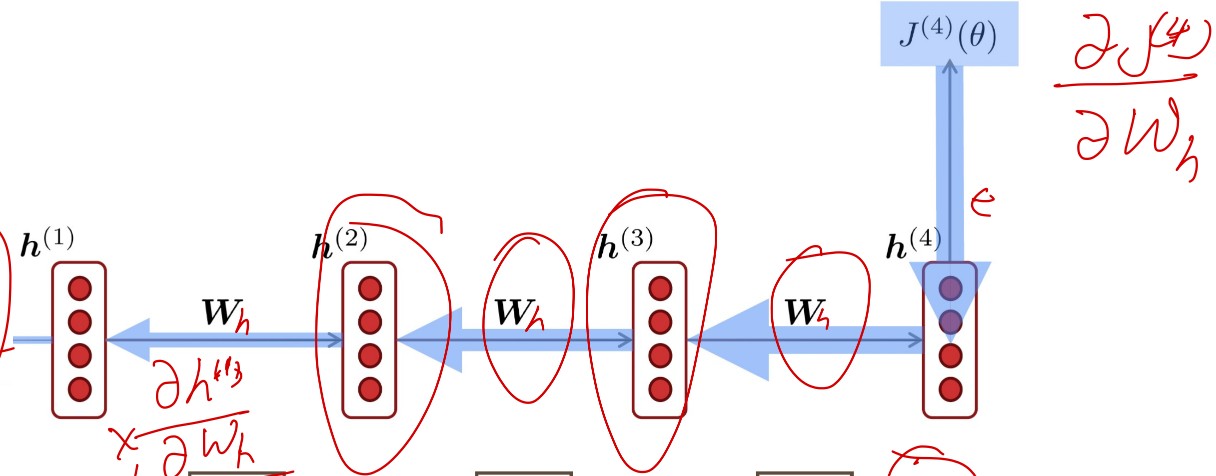
$$\frac{\partial L}{\partial W_h} \propto \sum_{1\le k\le t} \boxed{\left(\boxed{\prod_{t\ge i>k} \frac{\partial h_i}{\partial h_{i-1}}}\right) \frac{\partial h_k}{\partial W_h}} \longrightarrow \text{Contribution of hidden state } k$$

Length of the product proportional to how far k is from t

$$\boxed{\frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial h_{t-2}} \frac{\partial h_{t-2}}{\partial h_{t-3}} \frac{\partial h_{t-3}}{\partial h_{t-4}} \frac{\partial h_{t-4}}{\partial h_{t-5}} \frac{\partial h_{t-5}}{\partial h_{t-6}} \frac{\partial h_{t-6}}{\partial h_{t-7}} \frac{\partial h_{t-7}}{\partial h_{t-8}} \frac{\partial h_{t-8}}{\partial h_{t-9}} \frac{\partial h_{t-9}}{\partial h_{t-10}} \frac{\partial h_{t-10}}{\partial W_h}}$$

Contribution of hidden state *t-10*

Vanishing vs. Exploding Gradient



$$\frac{\partial J^{(4)}}{\partial W_h} = \frac{\partial J^{(4)}}{\partial h^{(1)}} \times \frac{\partial h^{(1)}}{\partial W_h}$$

$$\frac{\partial J^{(4)}}{\partial h^{(1)}} = \boxed{\frac{\partial h^{(2)}}{\partial h^{(1)}}} \times \boxed{\frac{\partial h^{(3)}}{\partial h^{(2)}}} \times \boxed{\frac{\partial h^{(4)}}{\partial h^{(3)}}} \times \frac{\partial J^{(4)}}{\partial h^{(4)}}$$

What happens if these are small?

Vanishing gradient problem:
When these are small, the gradient signal gets smaller and smaller as it backpropagates further

Gradient signals from far away will be lost!
The weights W_h only capture near effects.

On the difficulty of training recurrent neural networks
R Pascanu, T Mikolov, Y Bengio
International conference on machine learning, 2013 · proceedings.mlr.press

Abstract
There are two widely known issues with properly training recurrent neural networks, the vanishing and the exploding gradient problems detailed in Bengio et al.(1994). In this paper we attempt to improve the understanding of the underlying issues by exploring these problems from an analytical, a geometric and a dynamical systems perspective. Our analysis is used to justify a simple yet effective solution. We propose a gradient norm clipping strategy to deal with exploding gradients and a soft constraint for the vanishing

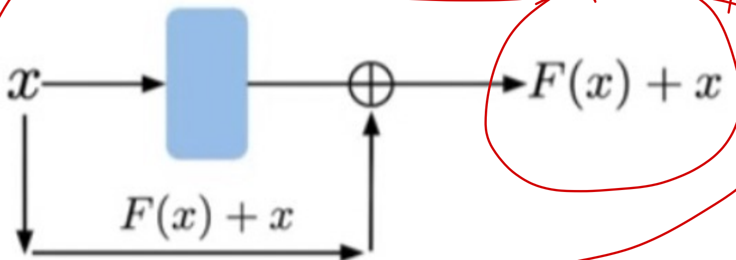
Gradient Clipping and Skip Connection

Reference: Stanford CS224N, Lecture 5:
https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture05-rnnlm.pdf

- The exploding gradient problem is relatively easier to address: Gradient Clipping.
- Intuition: Take a smaller step in the same direction.
- One idea to address vanishing gradient is to create direct and linear pass-through connections in the model: Residual/skip connections, attention, etc.

Algorithm 1 Pseudo-code for norm clipping

$\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$
if $\|\hat{\mathbf{g}}\| \geq threshold$ **then**
  $\hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|}\hat{\mathbf{g}}$
end if



$= F'(x) + 1$

$F'(x) < 1$

Deep residual learning for image recognition
K He, X Zhang, S Ren, J Sun - … and pattern recognition, 2016 - openaccess.thecvf.com
… Deeper neural networks are more difficult to train. We present a residual learning framework to ease the training of networks that are substantially deeper than those used previously. …
★ 保存 99 引用 被引用次数：196717 相关文章 所有 76 个版本 »