Word Vectors (Part 2)

CS114B Lab 7

Kenneth Lai

March 17, 2022

Distributed Representations of Words

- Representations of (contexts of) words as embeddings in some vector space
- Two approaches to distributed, distributional representations (Baroni et al. 2014):
  - Count-based
    - Count occurrences of words in contexts, optionally followed by some mathematical transformation (e.g. tf-idf, PPMI, SVD)
  - Prediction-based
    - Given some context vector(s) $\mathbf{c}$, predict some word $\mathbf{x}$ (or vice versa)
    - a.k.a. language modeling-based

Language Models

- Given some context vector(s) $\mathbf{c}$, predict some word $\mathbf{x}$ (or vice versa)

Language Models

- Given some context vector(s) $\mathbf{c}$, predict some word $\mathbf{x}$ (or vice versa)
- Two approaches to language models:

Language Models

- Given some context vector(s) $\mathbf{c}$, predict some word $\mathbf{x}$ (or vice versa)
- Two approaches to language models:
  - Generative models

Language Models

- Given some context vector(s) $\mathbf{c}$, predict some word $\mathbf{x}$ (or vice versa)
- Two approaches to language models:
  - Generative models
    - Model the joint probability distribution $P(\mathbf{x}, \mathbf{c})$

Language Models

- Given some context vector(s) $\mathbf{c}$, predict some word $\mathbf{x}$ (or vice versa)
- Two approaches to language models:
  - Generative models
    - Model the joint probability distribution $P(\mathbf{x}, \mathbf{c})$
    - Examples: n-gram language models

Language Models

- Given some context vector(s) $\mathbf{c}$, predict some word $\mathbf{x}$ (or vice versa)
- Two approaches to language models:
  - Generative models
    - Model the joint probability distribution $P(\mathbf{x}, \mathbf{c})$
    - Examples: n-gram language models
      - Unigram: predict $P(\mathbf{x}_i)$
      - Bigram: predict $P(\mathbf{x}_i|\mathbf{x}_{i-1})$
      - Trigram: predict $P(\mathbf{x}_i|\mathbf{x}_{i-2}, \mathbf{x}_{i-1})$

Language Models

- Given some context vector(s) $\mathbf{c}$, predict some word $\mathbf{x}$ (or vice versa)
- Two approaches to language models:
  - Discriminative models

Language Models

- Given some context vector(s) $\mathbf{c}$, predict some word $\mathbf{x}$ (or vice versa)
- Two approaches to language models:
  - Discriminative models
    - Predict the conditional probability $P(\mathbf{x}|\mathbf{c})$ (or $P(\mathbf{c}|\mathbf{x})$) directly

Language Models

- Given some context vector(s) $\mathbf{c}$, predict some word $\mathbf{x}$ (or vice versa)
- Two approaches to language models:
  - Discriminative models
    - Predict the conditional probability $P(\mathbf{x}|\mathbf{c})$ (or $P(\mathbf{c}|\mathbf{x})$) directly
    - Examples: neural network language models

Language Models

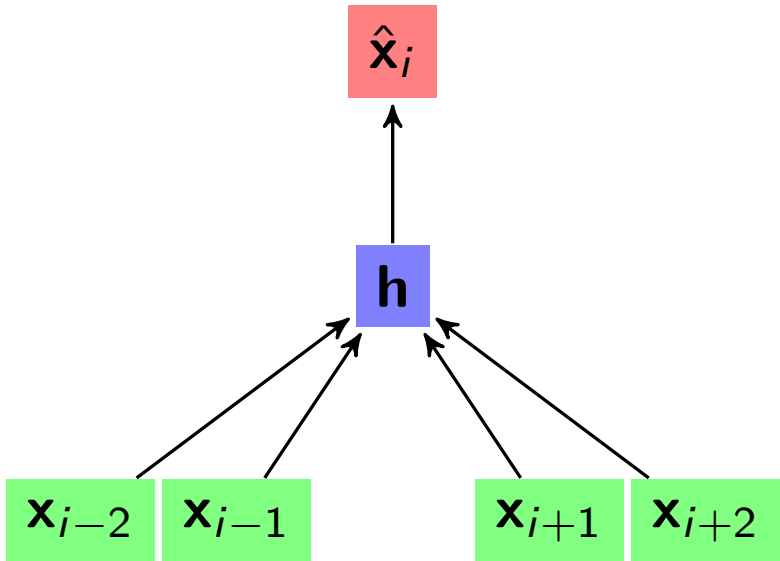
- Given some context vector(s) $\mathbf{c}$, predict some word $\mathbf{x}$ (or vice versa)
- Two approaches to language models:
  - Discriminative models
    - Predict the conditional probability $P(\mathbf{x}|\mathbf{c})$ (or $P(\mathbf{c}|\mathbf{x})$) directly
    - Examples: neural network language models
      - Feedforward: word2vec (Mikolov et al. 2013a, 2013b)
      - Recurrent:  (Peters et al. 2018)
      - Transformer:  (Devlin et al. 2019)

word2vec

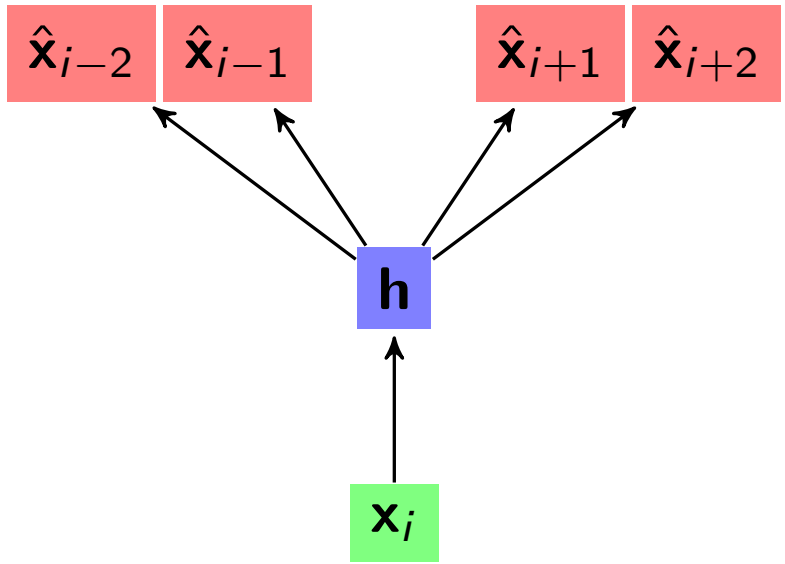
- Based on a feedforward neural network language model

word2vec

- Based on a feedforward neural network language model



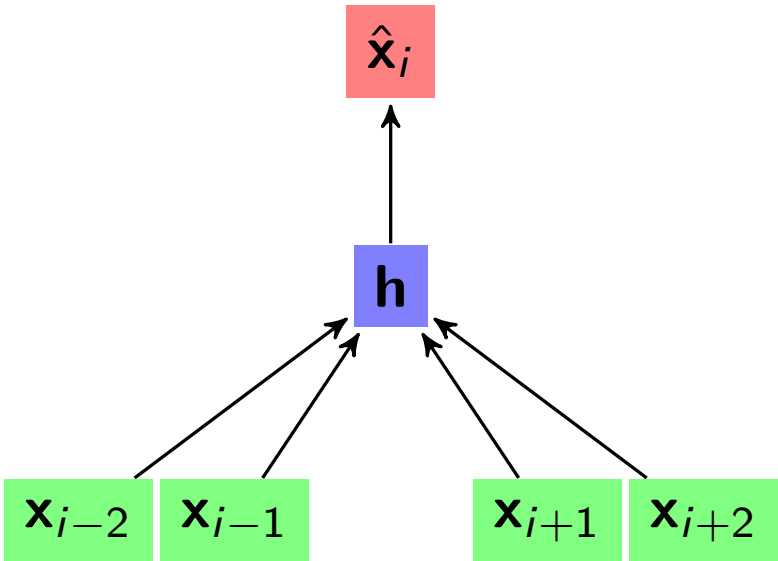
CBOW



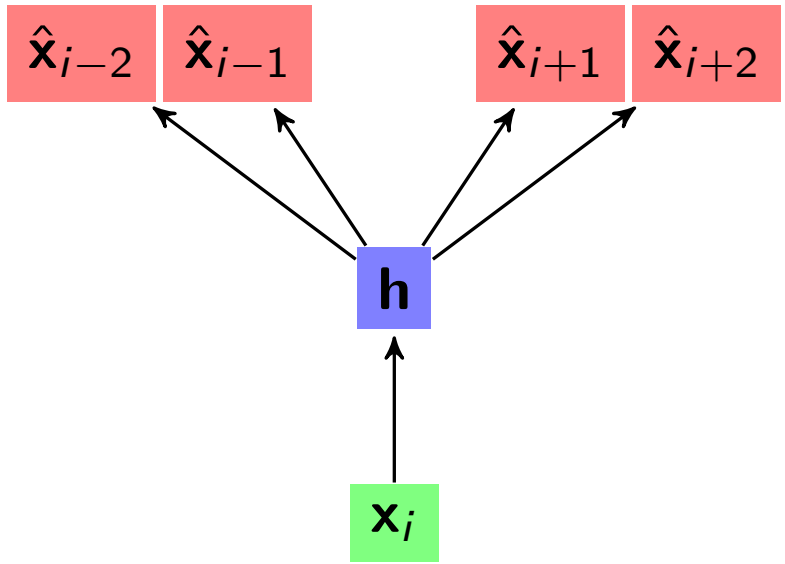
Skip-gram

word2vec

- Based on a feedforward neural network language model



CBOW

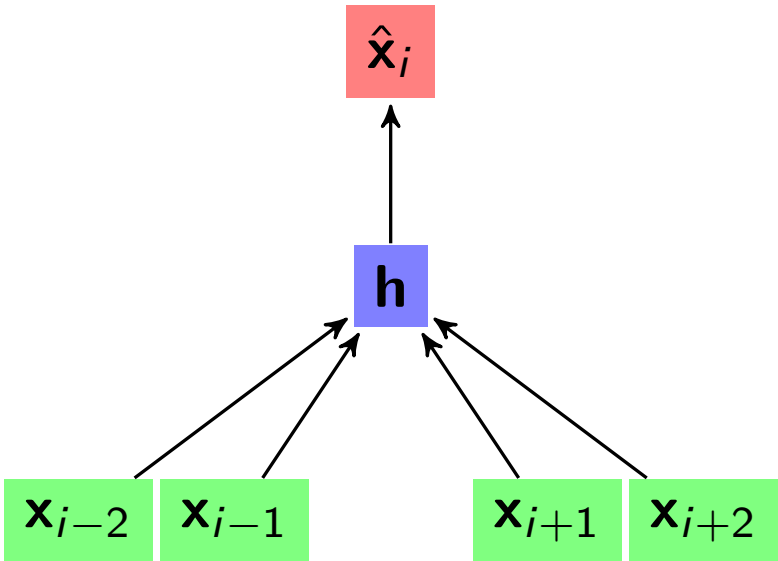


Skip-gram

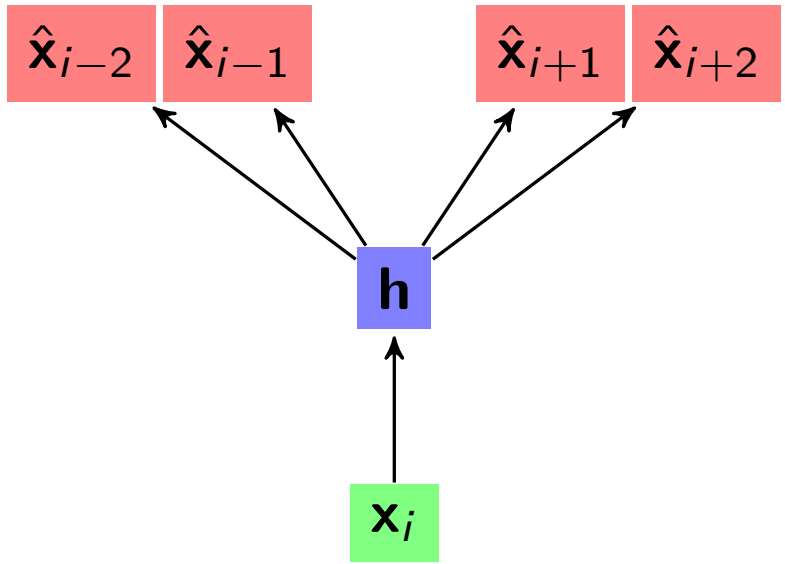
- Continuous bag of words (CBOW): use context to predict current word

word2vec

- Based on a feedforward neural network language model



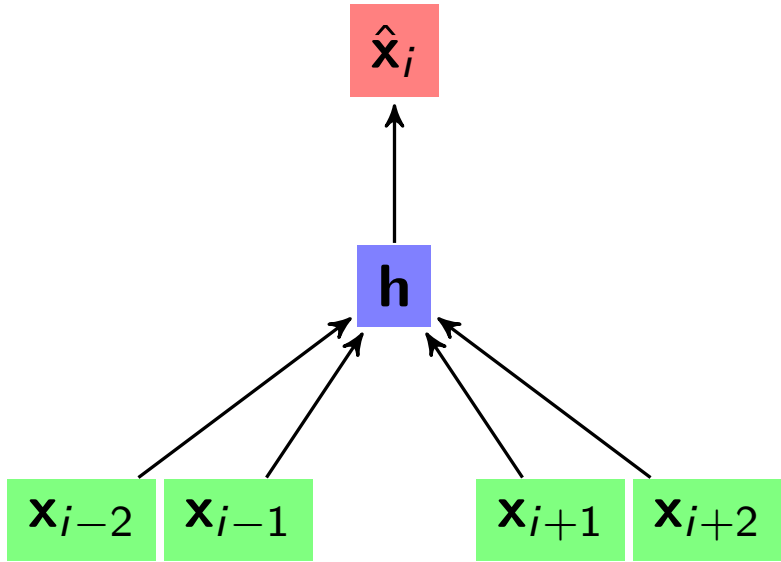
CBOW



Skip-gram

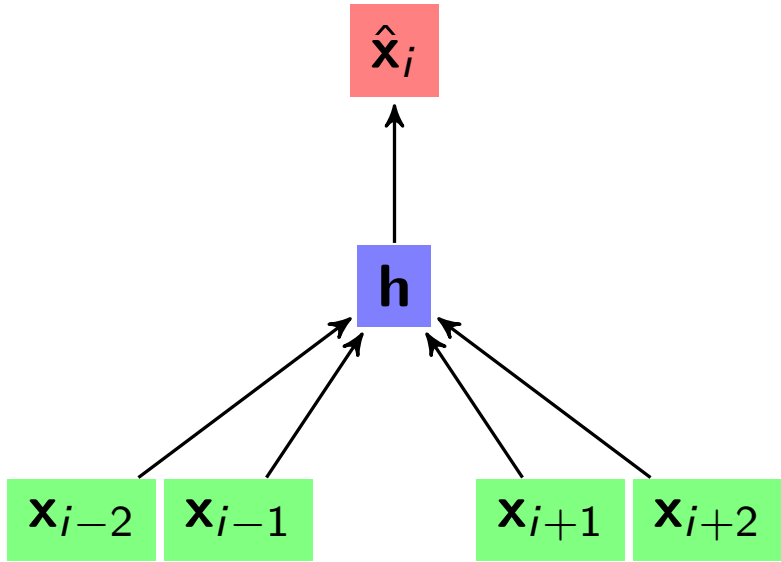
- Continuous bag of words (CBOW): use context to predict current word
- Skip-gram: use current word to predict context

CBOW



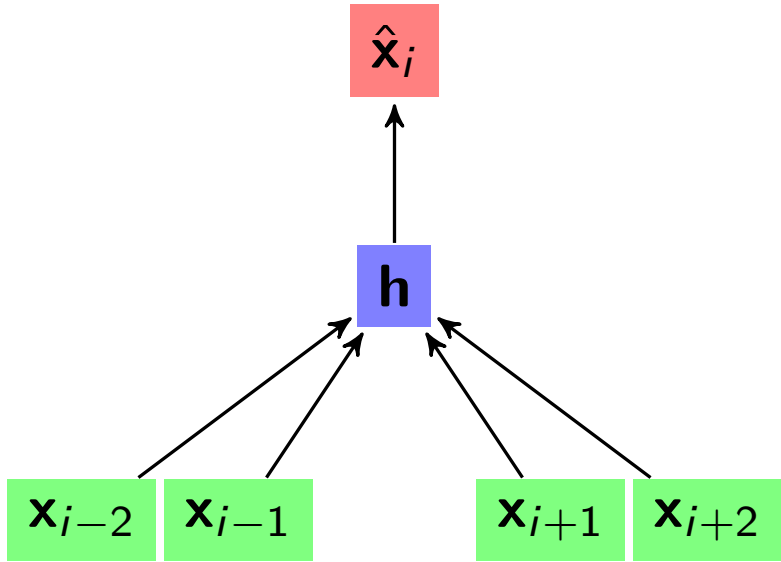
- Input layer: one-hot word vectors
  - $\begin{bmatrix} 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \end{bmatrix}$

CBOW



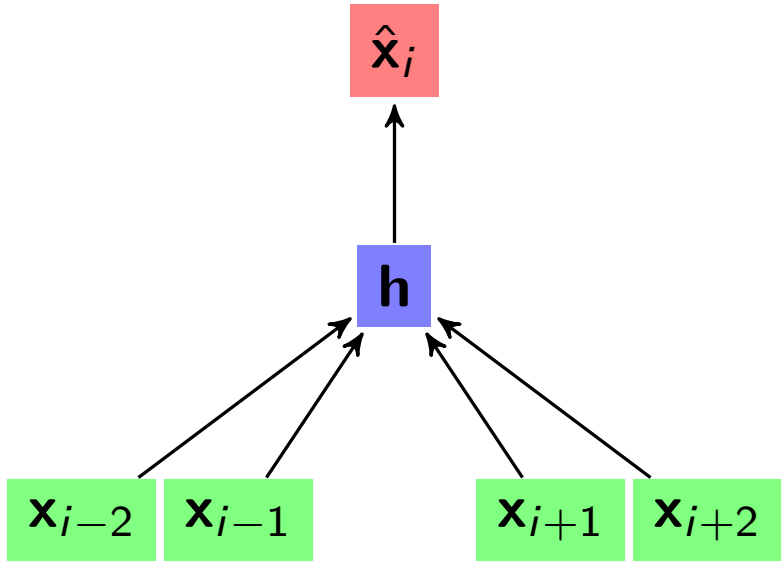
- Input layer: one-hot word vectors
  - $\begin{bmatrix} 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \end{bmatrix}$
  - Context words within some window

CBOW



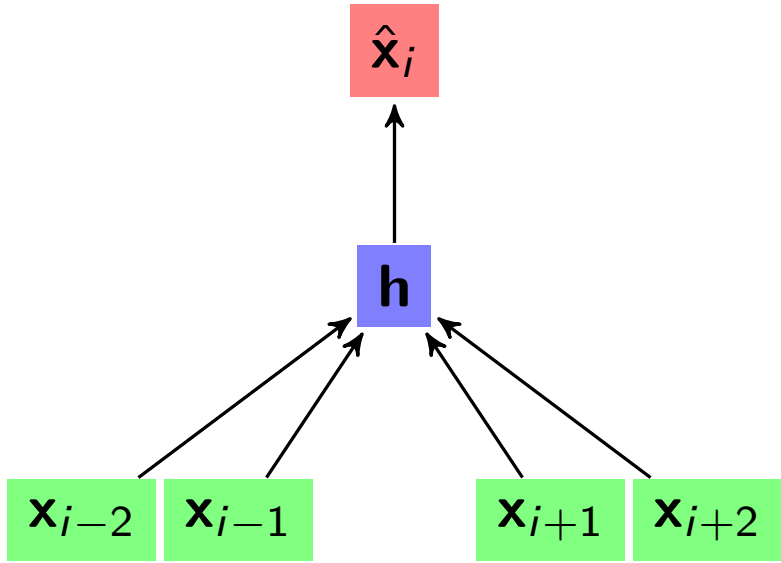
- Hidden (projection) layer: identity activation function, no bias
  - Weight matrix shared for all context words

CBOW



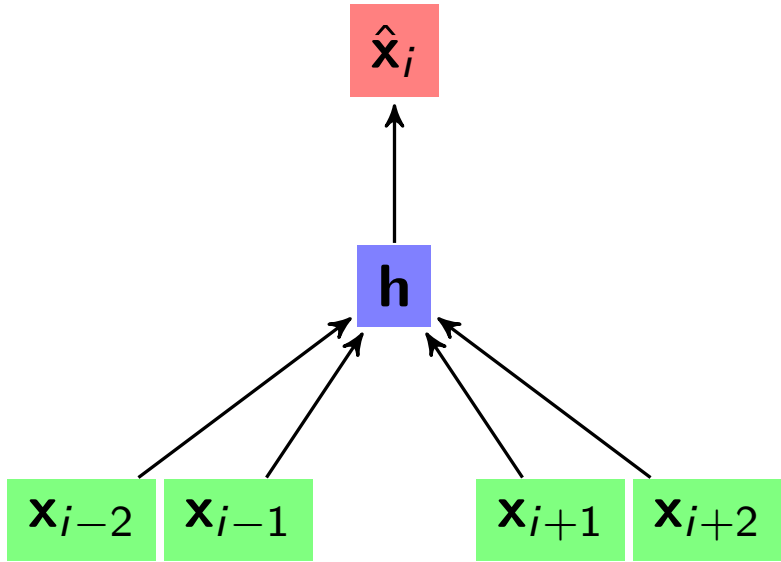
- Hidden (projection) layer: identity activation function, no bias
  - Weight matrix shared for all context words
  - Input $\rightarrow$ hidden $=$ table lookup (in weight matrix)

CBOW



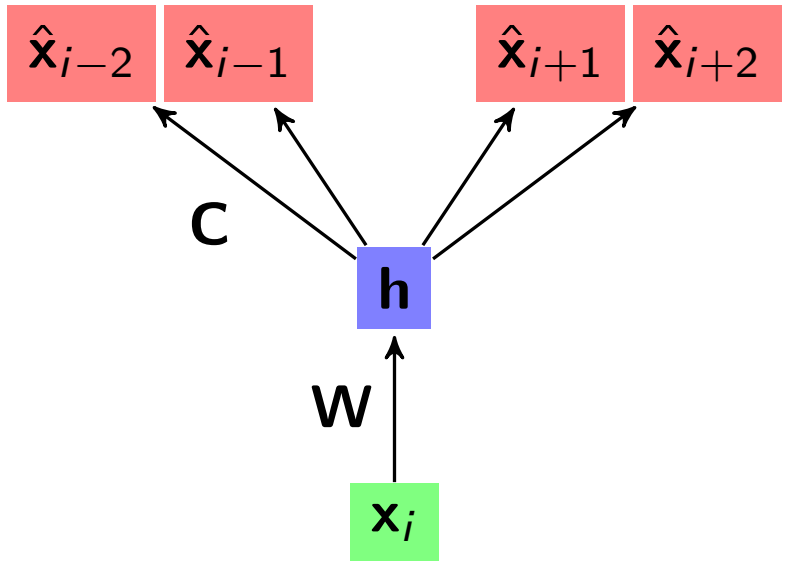
- Hidden (projection) layer: identity activation function, no bias
  - Weight matrix shared for all context words
  - Input $\rightarrow$ hidden $=$ table lookup (in weight matrix)
  - Context word vectors are averaged

CBOW



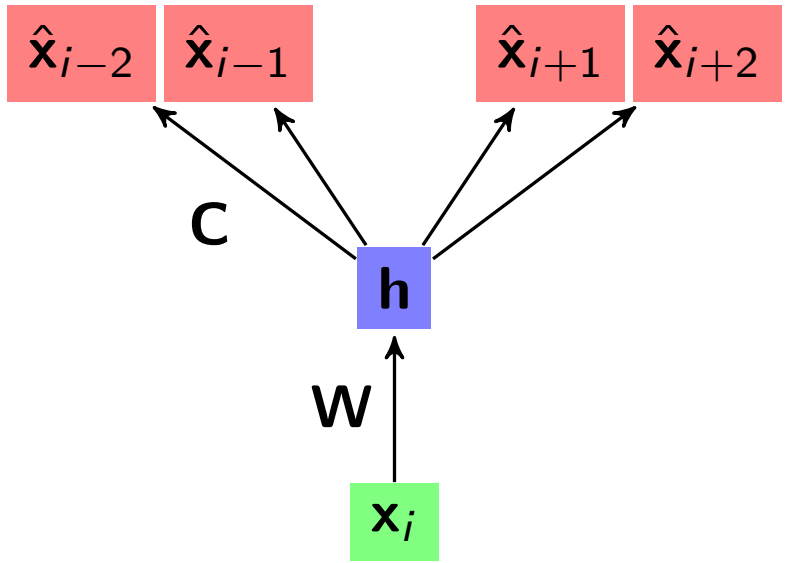
- Output layer: softmax activation function
  - Numbers $\rightarrow$ probabilities

Skip-gram



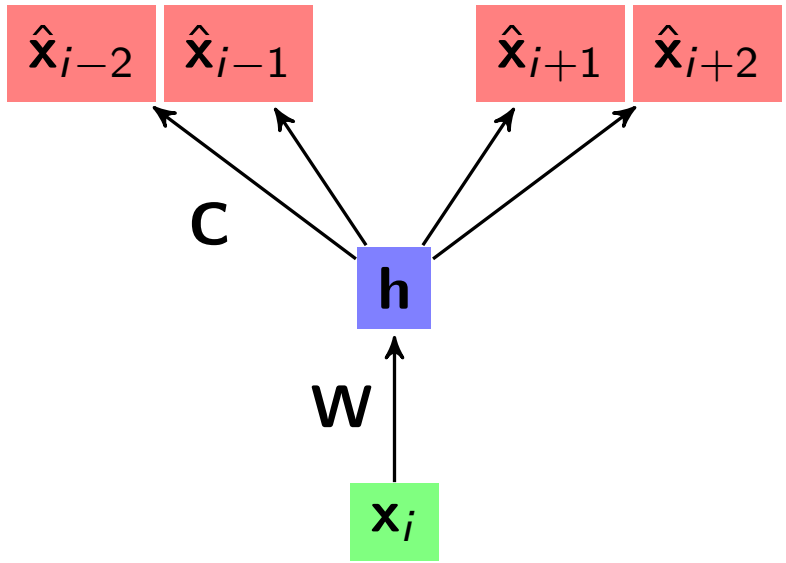
- Input layer: one-hot word vectors
  - $\begin{bmatrix} 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \end{bmatrix}$

Skip-gram



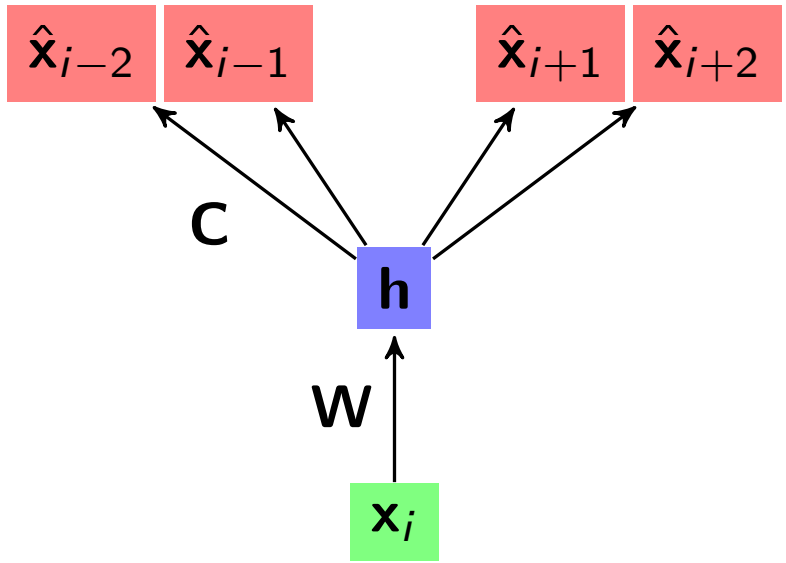
- Hidden (projection) layer: identity activation function, no bias
  - Input $\rightarrow$ hidden $=$ table lookup (in weight matrix)

Skip-gram



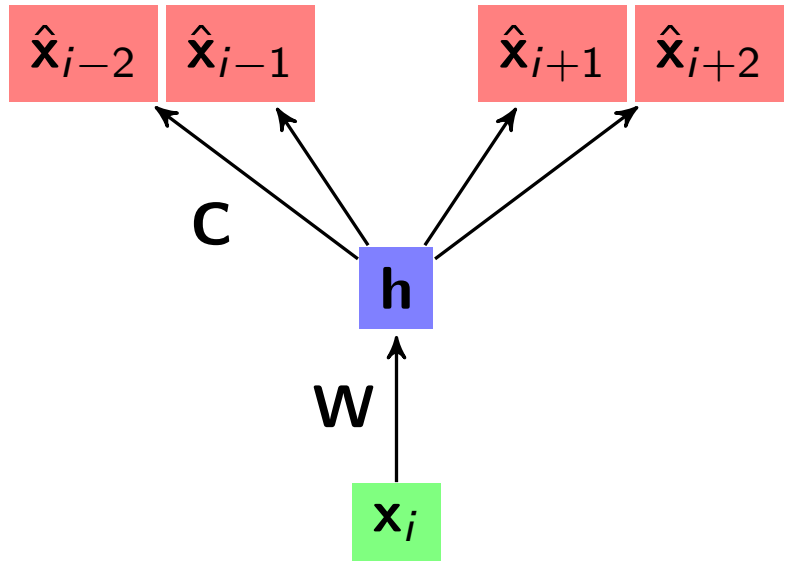
- Output layer: softmax activation function
  - Predict context words within some window

Skip-gram



- Output layer: softmax activation function
  - Predict context words within some window
  - Separate classification for each context word

Skip-gram



- Output layer: softmax activation function
  - Predict context words within some window
  - Separate classification for each context word
  - Closer context words sampled more than distant context words

Negative Sampling

- Skip-gram model: let $\mathbf{w}$ be a target word vector and $\mathbf{c}$ be a context word vector

Negative Sampling

- Skip-gram model: let $\mathbf{w}$ be a target word vector and $\mathbf{c}$ be a context word vector
  - $\mathbf{w}$ = row of $\mathbf{W}$ corresponding to input word (= output of hidden layer)

Negative Sampling

- Skip-gram model: let $\mathbf{w}$ be a target word vector and $\mathbf{c}$ be a context word vector
  - $\mathbf{w}$ = row of $\mathbf{W}$ corresponding to input word (= output of hidden layer)
  - $\mathbf{c}$ = column of $\mathbf{C}$ corresponding to output word

Negative Sampling

- Skip-gram model: let $\mathbf{w}$ be a target word vector and $\mathbf{c}$ be a context word vector
  - $\mathbf{w}$ = row of $\mathbf{W}$ corresponding to input word (= output of hidden layer)
  - $\mathbf{c}$ = column of $\mathbf{C}$ corresponding to output word
- Negative sampling loss function

$$L = -\left[\log \sigma(\mathbf{w} \cdot \mathbf{c}_{pos}) + \sum_{i=1}^{k} \log \sigma(\mathbf{w} \cdot -\mathbf{c}_{neg})\right]$$

Negative Sampling

- Skip-gram model: let $\mathbf{w}$ be a target word vector and $\mathbf{c}$ be a context word vector
  - $\mathbf{w}$ = row of $\mathbf{W}$ corresponding to input word (= output of hidden layer)
  - $\mathbf{c}$ = column of $\mathbf{C}$ corresponding to output word
- Negative sampling loss function

$$L = -\left[\log \sigma(\mathbf{w} \cdot \mathbf{c}_{pos}) + \sum_{i=1}^{k} \log \sigma(\mathbf{w} \cdot -\mathbf{c}_{neg})\right]$$

  - Regular cross-entropy loss

Negative Sampling

- Skip-gram model: let $\mathbf{w}$ be a target word vector and $\mathbf{c}$ be a context word vector
  - $\mathbf{w}$ = row of $\mathbf{W}$ corresponding to input word (= output of hidden layer)
  - $\mathbf{c}$ = column of $\mathbf{C}$ corresponding to output word
- Negative sampling loss function

$$L = -\left[\log \sigma(\mathbf{w} \cdot \mathbf{c}_{pos}) + \sum_{i=1}^{k} \log \sigma(\mathbf{w} \cdot -\mathbf{c}_{neg})\right]$$

  - Regular cross-entropy loss
  - Maximize the (log-)probability of positive samples (true context words)

Negative Sampling

- Skip-gram model: let $\mathbf{w}$ be a target word vector and $\mathbf{c}$ be a context word vector
  - $\mathbf{w}$ = row of $\mathbf{W}$ corresponding to input word (= output of hidden layer)
  - $\mathbf{c}$ = column of $\mathbf{C}$ corresponding to output word
- Negative sampling loss function

$$L = -\left[\log \sigma(\mathbf{w} \cdot \mathbf{c}_{pos}) + \sum_{i=1}^{k} \log \sigma(\mathbf{w} \cdot -\mathbf{c}_{neg})\right]$$

  - Minimize the (log-)probability of k negative samples (random non-context words)

word2vec

- Skip-gram model: for each word, word2vec learns two word embeddings

word2vec

- Skip-gram model: for each word, word2vec learns two word embeddings
  - Target word vector $\mathbf{w}$ (row of $\mathbf{W}$, = output of hidden layer)
  - Context word vector $\mathbf{c}$ (column of $\mathbf{C}$)

word2vec

- Skip-gram model: for each word, word2vec learns two word embeddings
  - Target word vector $\mathbf{w}$ (row of $\mathbf{W}$, = output of hidden layer)
  - Context word vector $\mathbf{c}$ (column of $\mathbf{C}$)
- Common final word embeddings
  - Add $\mathbf{w} + \mathbf{c}$
  - Just $\mathbf{w}$ (throw away $\mathbf{c}$)

References

- Baroni, Marco, Georgiana Dinu, and Germán Kruszewski. "Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors." Proceedings of ACL. 2014.
- Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." Proceedings of NAACL-HLT. 2019.
- Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. "Efficient estimation of word representations in vector space." Proceedings of ICLR. 2013.
- Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. "Distributed representations of words and phrases and their compositionality." Proceedings of NeurIPS. 2013.
- Peters, Matthew E., Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. "Deep Contextualized Word Representations." Proceedings of NAACL-HLT. 2018.