Natural Language Processing

Learning word embeddings: The skip-gram model

Marco Kuhlmann
Department of Computer and Information Science

LINKÖPING UNIVERSITY

The skip-gram model

Eisenstein § 14.5.2

- The **skip-gram model** is one of two word embedding models implemented in Google's word2vec software.
- In the context of this model, a **skip-gram** is a pair of words from a text that are separated by at most *k* other words.
- The word embeddings are obtained as a by-product of the task to predict one word in the skip-gram from the other word.

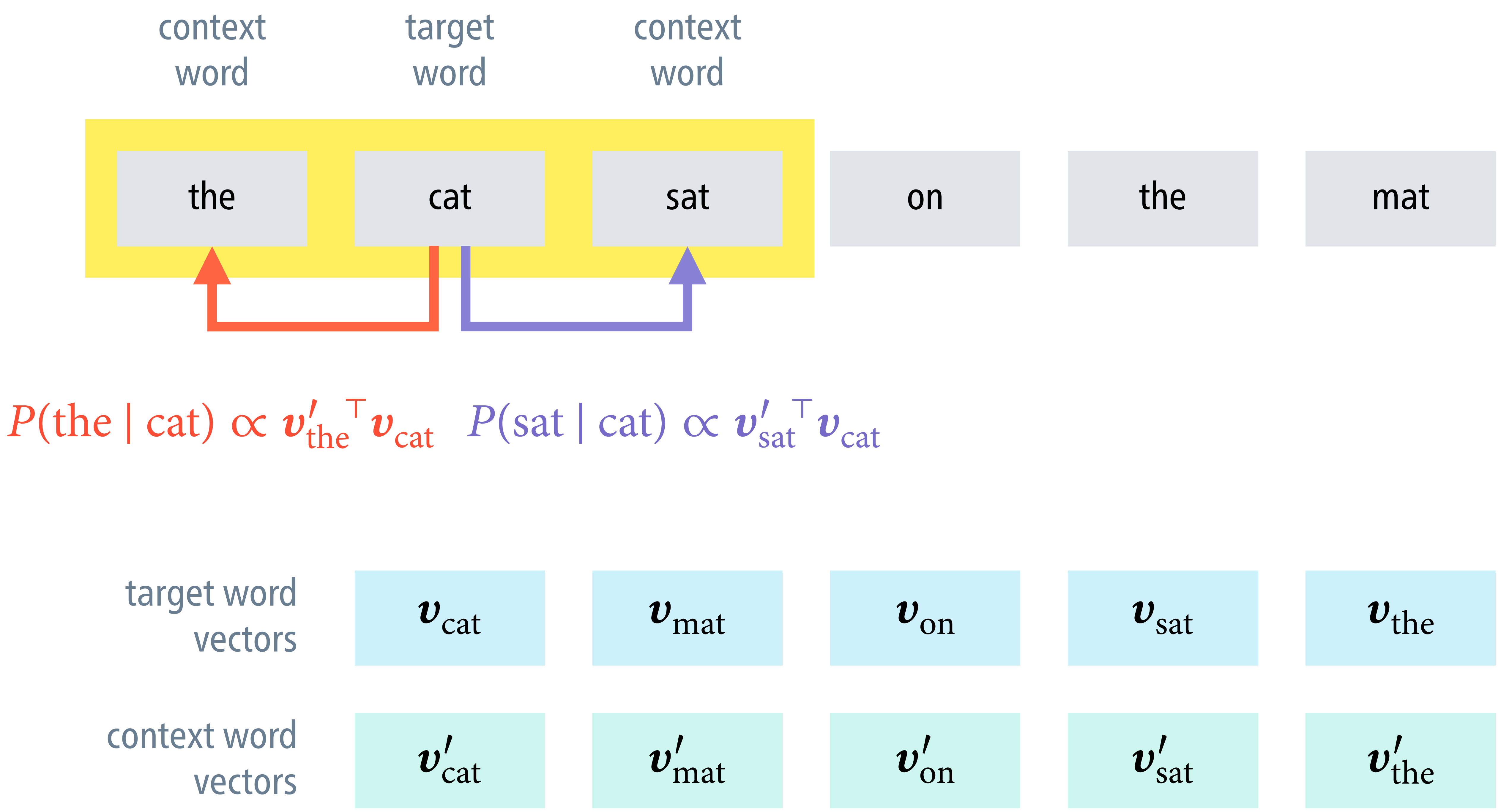
Mikolov et al. (2013)

Training the skip-gram model

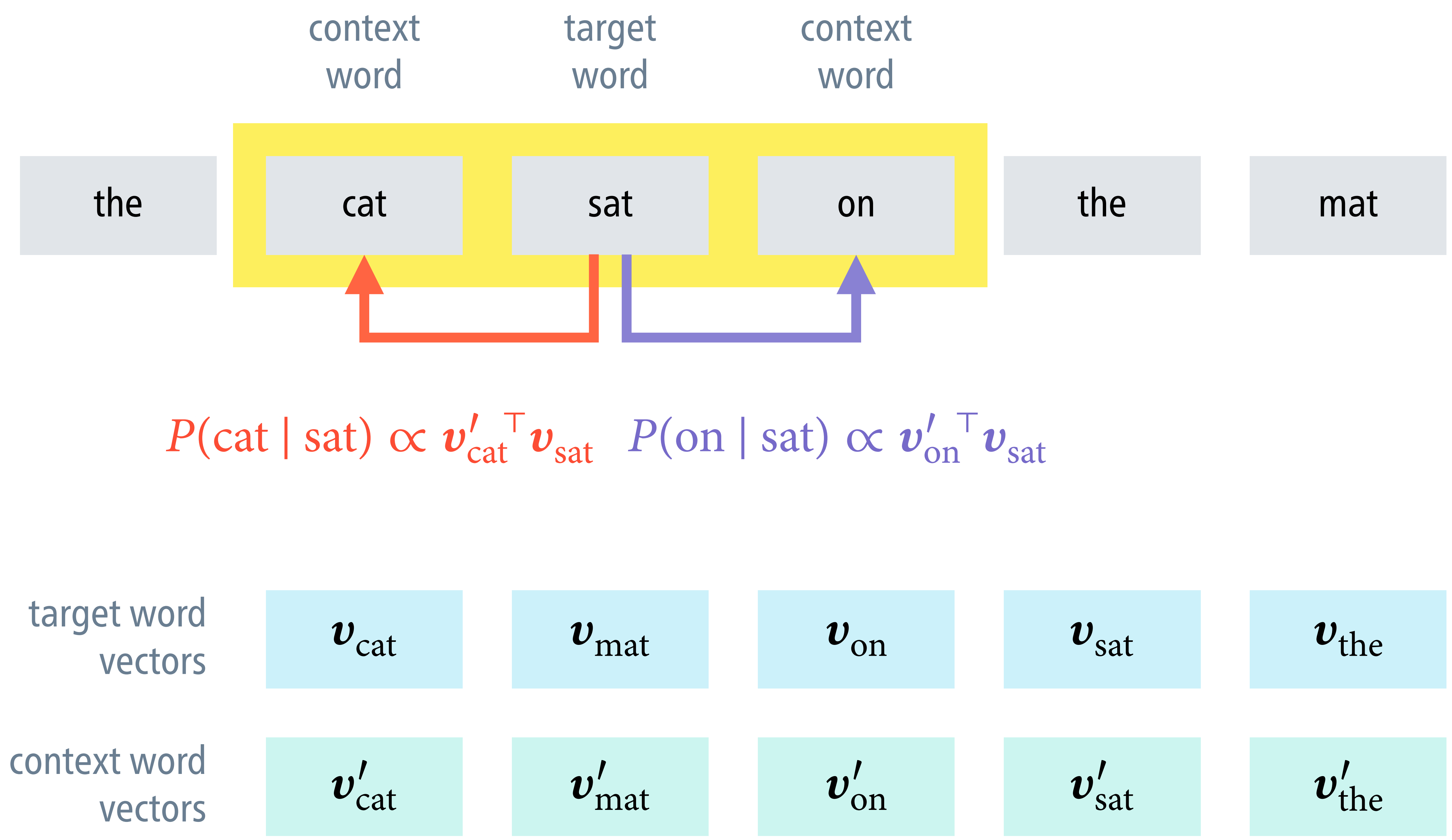
- Start with random word vectors.
- Move a small, symmetric window over the words in a text. Each window contains a target word w and context words c.
- For each window, use the similarity of the current word vectors for w and c to define a conditional probability $P(c|w)$.
- Tweak the word vectors to maximise this probability.

Mikolov et al. (2013)

Training the skip-gram model



Training the skip-gram model



The skip-gram model in detail (1)

- We maintain two separate vector representations: one for target words and one for context words. Initially, they are random.
- The probability of a context word c given a target word w is defined using the softmax function:

vector representation for context words

vector representation for target words

$$P(c \mid w\,;\boldsymbol{\theta}) = \frac{\exp(\boldsymbol{v}_c'^{\top}\boldsymbol{v}_w)}{\sum_{x \in V} \exp(\boldsymbol{v}_x'^{\top}\boldsymbol{v}_w)}$$

all parameters of the model

The skip-gram model in detail (2)

To *maximise* the conditional probabilities, we *minimise* the cross-entropy loss on the training data:

$$J(\boldsymbol{\theta}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{\substack{-m\leq j\leq m\\ j\neq 0}} \log P(w_{i+j} \mid w_i\,;\boldsymbol{\theta})$$

all parameters of the model

length of the text

size of each window

Computational complexity

Eisenstein § 14.5.3

Computing the softmax is expensive: For each position in the text, we need to sum over the complete vocabulary.

- **Solution 1:** Decompose the standard softmax computation into a tree-like structure of simpler computations.

  hierarchical softmax

- **Solution 2:** Instead of maximising the conditional probabilities directly, maximise simpler quantities that approximate them.

  negative sampling

Skip-gram with negative sampling

Eisenstein § 14.5.3

- Maximise the probability of observed word–context pairs, while minimising the probability of randomly drawn samples.

$$J(\boldsymbol{\theta}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{\substack{-m\leq j\leq m\\ j\neq 0}} \log\sigma(\boldsymbol{v}'^{\top}_{w_{i+j}}\boldsymbol{v}_{w_i}) + \sum_{c\sim D}\log\sigma(-\boldsymbol{v}'^{\top}_{c}\boldsymbol{v}_{w_i})$$

length of the text — logistic function — negative samples

- The negative samples are drawn from $D(c) \propto \#(c)^{\alpha}$, where α is a hyperparameter (default value: 0.75).

Skip-gram with negative sampling in detail

- Subsampling: To reduce the influence of very frequent words (and speed up learning), discard a token w with probability

$$P(w) = \max\left(0, 1 - \sqrt{tN/\#(w)}\right)$$

count of the word w

where t is a chosen threshold (default value: 0.001).

- Do not use a constant window size; instead, sample window sizes up to the maximum size m with uniform probability.

As a consequence, far-away context words will get less influence.

The SGNS model as a neural network