7.3 Skip-gram Word Embeddings

Edwin Simpson

Department of Computer Science,

University of Bristol, UK.

Dense Vectors

- Term-document matrices are sparse with large numbers of dimensions and many near-zero values;
- A dense vector representation of 50-1000 values has advantages:
  - Fewer dimensions means models for tasks such as classification and sequence labelling need fewer parameters;
  - This leads to less **over-fitting** and better generalisation;
  - Relations such as synonymy can be better represented.
- Dense vectors can be learned using the **skip-gram** model,
  - Implemented by the software **word2vec;**
  - Alternatives: continuous bag of words; GloVe.

Skip-gram: Core Ideas

- Use the **contextual** view of meaning
  - Distributional hypothesis
  - Determine a word's meaning from its neighbouring words
- Represent the context that a target word *t* occurs in:
  - Term-document matrix: whole document
  - Skip-gram: **context** window of ± k words either side of the target word

```
... lemon,  a [tablespoon of apricot jam,     a] pinch ...
               c1          c2    t      c3      c4
```

Section 6.8, Speech and Language Processing, 3rd edition draft, Jurafsky & Martin (2021).

Skip-gram: Core Ideas

- Embeddings as a by-product of a classifier:
  - Learn a classifier to distinguish real and fake contexts for any given *t*
  - The parameters of this classifier form an **embedding vector**
- **Self-supervised learning:**
  - No training labels, learn to predict part of the text itself
  - Positive examples: the real context words in a set of training documents
  - Negative examples: randomly sampled words from the vocabulary.

Prediction with Skip-gram

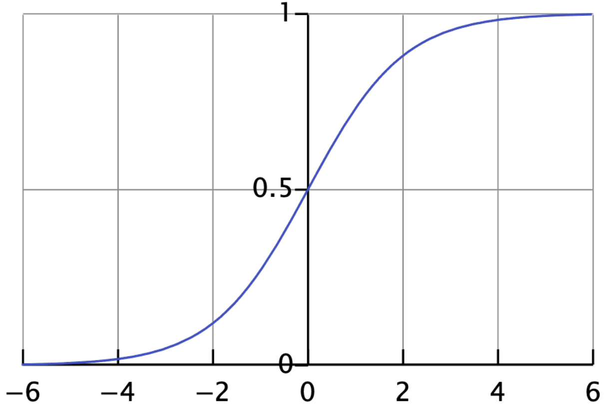
$$P(+ \mid \boldsymbol{c}, \boldsymbol{t}) = \prod_{i=1}^{k} \frac{1}{1+e^{-\boldsymbol{t} \cdot \boldsymbol{c_i}}}$$

Prediction with Skip-gram

- Recall that cosine similarity is a normalised dot product;
- We combine the embedding vectors for target word *t* and context word c_i using the dot product, $\boldsymbol{t} \cdot \boldsymbol{c_i}$.
- The value of $\boldsymbol{t} \cdot \boldsymbol{c_i}$ ranges from $-\infty$ to ∞.

Prediction with Skip-gram

- **Sigmoid function** maps real values to numbers between 0 and 1.
- $\sigma(\boldsymbol{t} \cdot \boldsymbol{c_i}) = \frac{1}{1+e^{-\boldsymbol{t} \cdot \boldsymbol{c_i}}}$
- This means we have a **logistic regression** classifier for each target word with weights $\boldsymbol{t}$
- It maps input vectors $\boldsymbol{c_i}$ to probabilities.



Prediction with Skip-gram

- If a context ***c*** is true (positive), the occurrences of all its individual words must be positive;
- Assume the truth values of context words are **conditionally independent:**

$$P(+ \mid \boldsymbol{c}, \boldsymbol{t}) = \prod_{i=1}^{k} \frac{1}{1+e^{-\boldsymbol{t} \cdot \boldsymbol{c}_{\boldsymbol{i}}}}$$

- Summary:
  - Combine embeddings of target and context words using dot product;
  - Apply sigmoid function to obtain a probability;
  - Take a product of independent probabilities.

Choosing Negative Examples

- Noise words are selected randomly according to their weighted frequency:

$$P_{\alpha(w)} = \frac{count(w)^{\alpha}}{\sum_{w'} count(w')^{\alpha}}$$

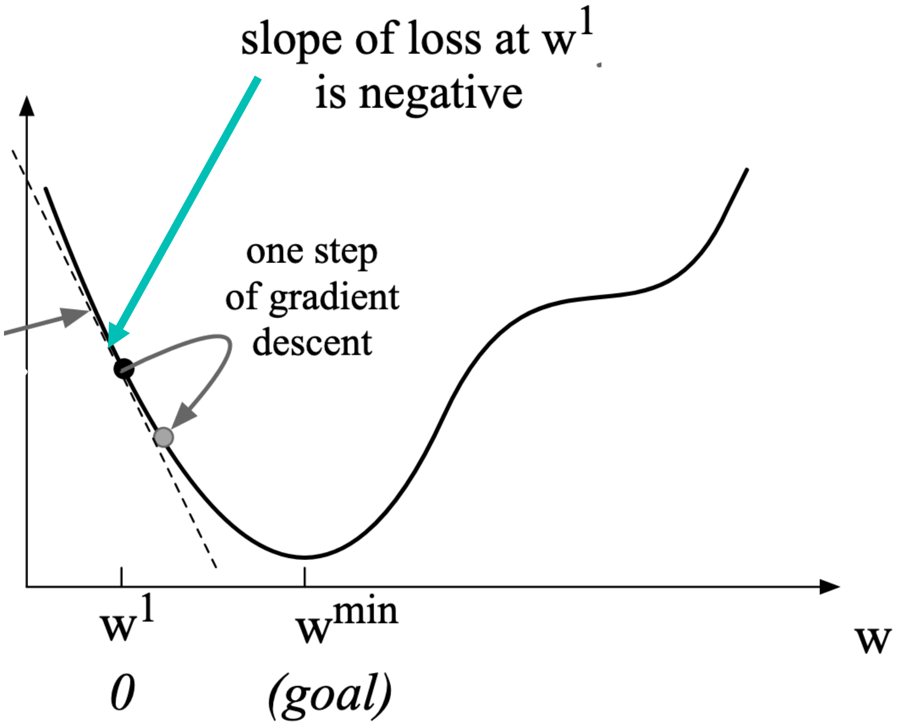
- Selecting by frequency avoids comparing with rare negative words that are very easy to spot as fake;
- Weighting the frequencies by 0.75 is a heuristic that prevents selecting too many very frequent words like ‘the’.

Learning Objective

- Goal: choose $\boldsymbol{t}$ and $\boldsymbol{c_i}$ to minimise prediction error on the training set
- $L(\boldsymbol{T}, \boldsymbol{C}) = -\sum_{(\boldsymbol{t},\boldsymbol{c}) \in +} \log P(+|\boldsymbol{t}, \boldsymbol{c}) - \sum_{(\boldsymbol{t},\boldsymbol{c}) \in -} \log P(-|\boldsymbol{t}, \boldsymbol{c})$
- Two sets of embeddings for each word!
  - Target word embeddings;
  - Context word embeddings.
  - Typically, only the target embeddings ***T*** are taken from the model for use as embeddings in downstream tasks.

Gradient Descent

- Consider each dimension of each target and context vector as a ‘weight’, *w.*
- Starting from random initial values of *w…*
- Increase or decrease *w* in the opposite direction to the gradient of $L(\boldsymbol{T}, \boldsymbol{C})$:
- Thereby reduce the loss.



Mini-batch Stochastic Gradient Descent

$$w^{t+1} = w^t - \frac{\eta}{N}\,\nabla_w L(\boldsymbol{T}, \boldsymbol{C})$$

- $L(\boldsymbol{T}, \boldsymbol{C})$ is a sum over contexts, so is $\eta\,\nabla_w L(\boldsymbol{T}, \boldsymbol{C})$.
- So, split the training data into **batches, compute** the gradient for one batch at a time:

$$w^{t+1} = w^t - \frac{\eta}{B}\,\nabla_w L(\boldsymbol{T}_i, \boldsymbol{C}_i)$$

- Faster updates, parallelisation

∇_w: the gradient with respect to weight *w.*

N: number of training contexts.

η: learning rate that controls the size of each step.

B: batch size.

Mini-batch Stochastic Gradient Descent

- To learn the skipgram embeddings:
  - Iterate over batches;
  - Within each batch, iterate over the ‘weights’, i.e., the parameters in the target word and context vectors;
  - Output target word vectors as embeddings.
- Same algorithm is used for:
  - **Logistic regression**
  - **Neural networks**

Summary

- Skip-gram applies the distributional hypothesis to learn dense vector representations (embeddings) of words from their contexts;
- We learn a binary classifier that distinguishes real (+ve) and fake (-ve) contexts for any given target word;
- Stochastic gradient descent trains the classifier by iteratively making adjustments to the weights to reduce the loss
- The weights of the classifier for a particular target word are used as its embedding.
- Words with similar contexts will have similar embeddings.