Lab: CudaVision – Learning Vision Systems on Graphics Cards (MA-INF 4308)

Assignment 8

25.7.2016

Prof. Sven Behnke
Dr. Seongyong Koo

Word2Vec

- **Goal: Understanding Skip-gram model and implementing Word2Vect model with TensorFlow**

- **Word Embeddings**

- **Word2Vec**

- **Skip-gram model**

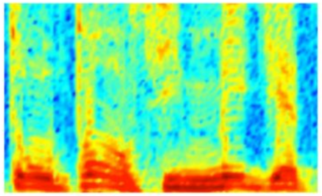
- **Theoretical reference**
    - "Distributed Representations of Words and Phrases and their Compositionality" by T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean (Google Inc.)
    - https://www.tensorflow.org/versions/r0.9/tutorials/word2vec/index.html#vector-representations-of-words

Word Embeddings

- Image and audio processing systems work with rich, high-dimensional datasets encoded as vectors of the individual raw pixel-intensities for image data, or e.g. power spectral density coefficients for audio data.

- Natural language processing systems traditionally treat words as discrete atomic symbols
  - These are arbitrary, and provide no useful information to the system regarding the relationships that may exist between the individual symbols. ('cats' and 'dogs' are both animals, four-legged, pets, etc.)
  - It leads to data sparsity and requires more data in order to successfully train statistical models.

AUDIO



Audio Spectrogram

DENSE

IMAGES

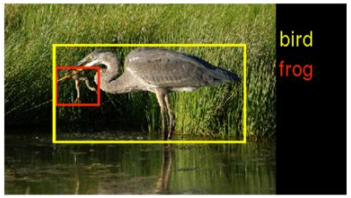


Image pixels

DENSE

TEXT

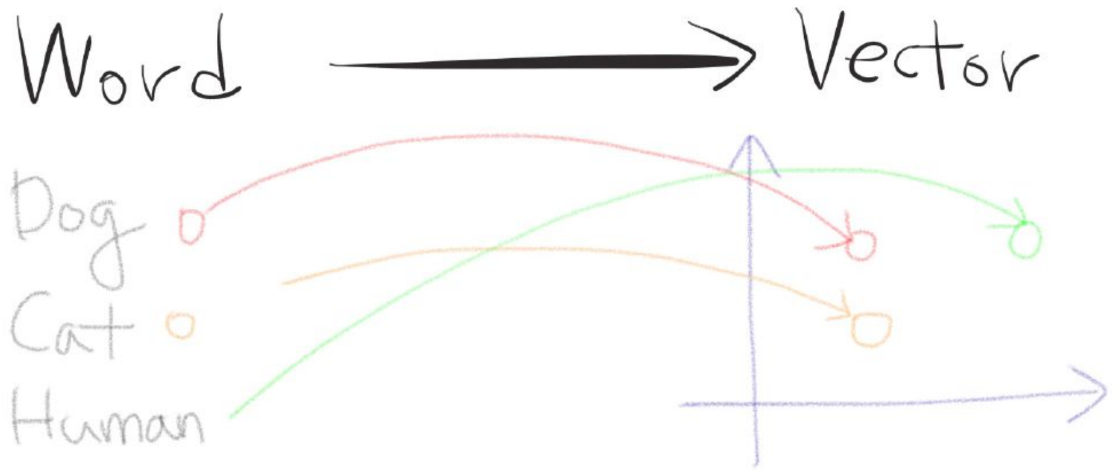
0	0	0	0.2	0	0.7	0	0	0	...	...

Word, context, or document vectors

SPARSE

Word Embeddings

- Vector space models (VSMs) represent (embed) words in a continuous vector space where semantically similar words are mapped to nearby points ('are embedded nearby each other').
- Word2vec is a particularly computationally-efficient predictive model for learning word embeddings from raw text.



Next Word Prediction

- N-Gram model
  - A type of probabilistic language model for predicting the next item in such a sequence in the form of a (n − 1)–order Markov model.
  - Two benefits of n-gram models (and algorithms that use them) are simplicity and scalability – with larger n, a model can store more context with a well-understood space–time tradeoff, enabling small experiments to scale up efficiently.

- Example
  - Source words: "I" "am" "a"
  - Target work: "body"

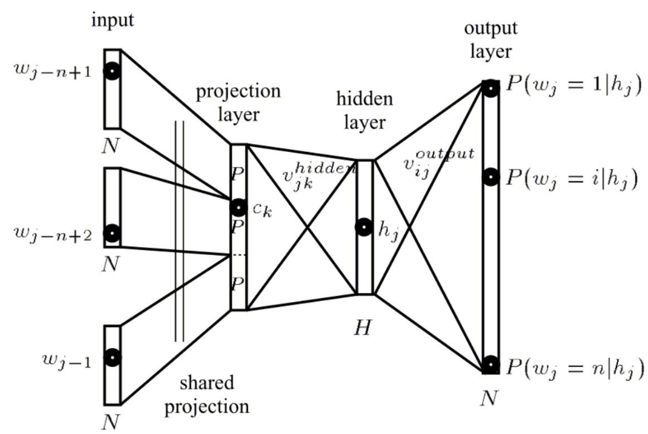
I am a boy

w_1 w_2 w_3 w_4

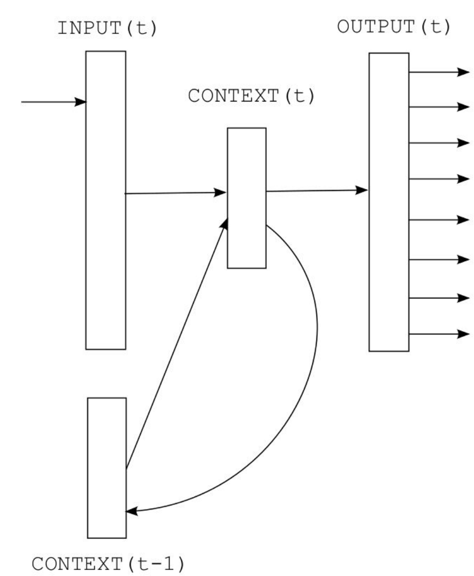
$$P(\text{boy} \mid \text{I am a}) = \frac{P(\text{I am a boy})}{P(\text{I am a})}$$

Neural Network Language Models

- Feedforward NNLM [Bengio et al. 2003]
  - A feedforward neural network with a linear projection layer and a non-linear hidden layer was used to learn jointly the word vector representation and a statistical language model.



- Recurrent NNLM [Mikolov et al. 2010]
  - Does not need to specify the context length and efficiently represent more complex patterns than NNLM.
  - Does not have a projection layer; only input, hidden and output layer.
  - Recurrent matrix that connects hidden layer to itself, using time-delayed connections (memory)



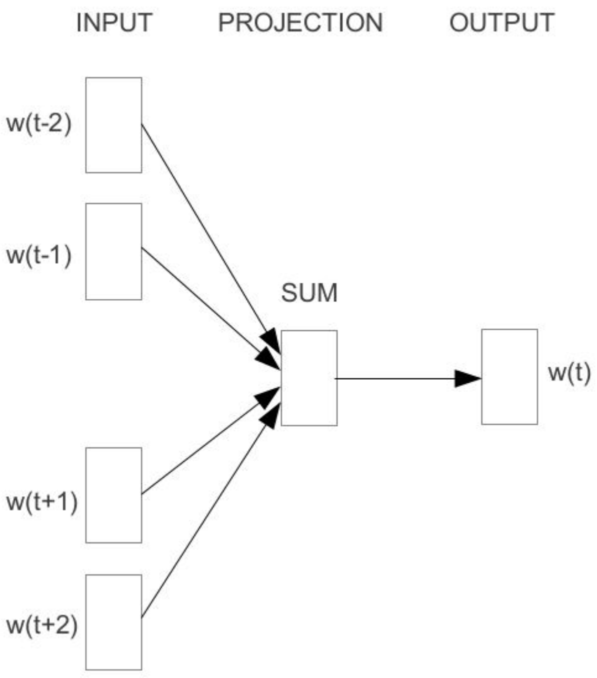
New Efficient NNLMs

- Goal
  - Learning distributed representations of words that try to minimize computational complexity.
  - Most of the complexity is caused by the non-linear hidden layer in the model.
  - Instead of representing data as precisely as neural networks, training simpler model on much more data efficiently.

- Idea
  - Continuous word vectors are learned using simple model, and then the N-gram NNLM is trained on top of these distributed representations of words.

New Efficient NNLMs

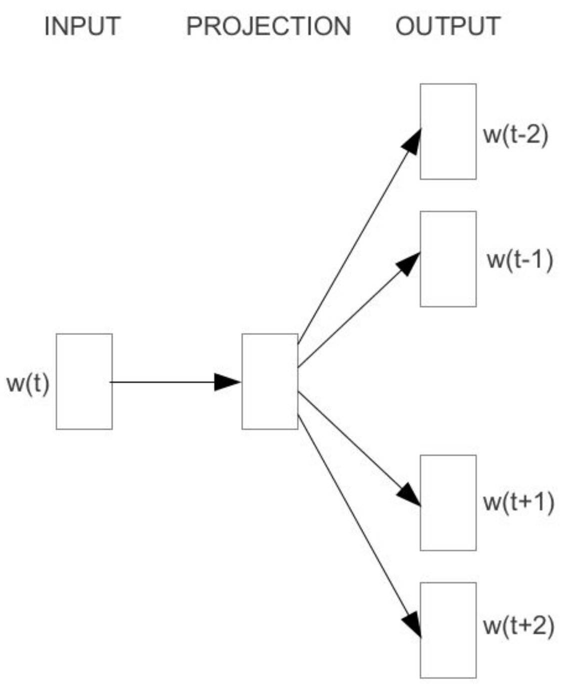
- Continuous Bag-of-Words Model (CBOW)
  - Similar to the feedforward NNLM, where the non-linear hidden layer is removed and the projection layer is shared for all words; thus, all words get projected into the same position (their vectors are averaged).
  - Building a log-linear classifier with future and history words at the input, where the training criterion is to correctly classify the current (middle) word.



CBOW

New Efficient NNLMs

- Skip-gram Model
  - Instead of predicting the current word based on the context, it tries to maximize classification of a word based on another word in the same sentence.
  - Use each current word as an input to a log-linear classifier with continuous projection layer, and predict words within a certain range before and after the current word.
  - Increasing the range improves quality of the resulting word vectors, but it also increases the computational complexity.



Skip-gram

New Efficient NNLMs

- Advantages of Skip-gram model compared to CBOW
  - Algorithmically, these models are similar, except that CBOW predicts target words (e.g. 'mat') from source context words ('the cat sits on the'), while the skip-gram does the inverse and predicts source context-words from the target words.
  - This inversion statistically has the effect that CBOW smoothes over a lot of the distributional information (by treating an entire context as one observation).
  - Useful thing for smaller datasets. Skip-gram treats each context-target pair as a new observation, and this tends to do better when we have larger datasets.

The Skip-gram Model

- The training objective
  - To find word representations that are useful for predicting the surrounding words in a sentence or a document.
  - Given a sequence of training words, maximize the average log probability

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t)$$

- Skip-gram formulation
  - v_w and v'_w are input and output vector representation of w

$$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^\top v_{w_I}\right)}{\sum_{w=1}^{W}\exp\left({v'_w}^\top v_{w_I}\right)}$$

  - Softmax evaluation for all words (W) is impractical!

The Skip-gram Model

- Noise Contrastive Estimation (NCE) [Mnih and Teh, 2012]
  - Basic assumption: a good model should be able to differentiate data from noise by means of logistic regression.
  - NCE can be shown to approximately maximize the log probability of the softmax

- Skip-gram model objective by NCE `tf.nn.nce_loss()`
  - Distinguish the target word from draws from the noise distribution Pn(w) using logistic regression, where there are k negative samples for each data sample.

$$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^\top v_{w_I}\right)}{\sum_{w=1}^{W} \exp\left({v'_w}^\top v_{w_I}\right)}$$

$$\Rightarrow \quad \log P(w_O|w_I) = \log \sigma({v'_{w_O}}^\top v_{w_I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[\log \sigma(-{v'_{w_i}}^\top v_{w_I})\right]$$

Assignment 8

- **Understanding Skip-gramm with a simple sentences**
  - Run 'word2vec_simple' and understand how to construct and train skip-gramm model

- **Training Skip-gramm model from a dictionary file**
  - Refer to a TensorFlow tutorial source file: https://github.com/tensorflow/tensorflow/blob/r0.9/tensorflow/examples/tutorials/word2vec/word2vec_basic.py

  - Download the text file and make a corpus
  - Make a dictionary with fixed length (using UNK token)
  - Make a batch generation function
  - Build a Skip-Gram Model
  - Train a Skip-Gram Model
  - Visualize the embeding using TSNE