Embeddings

(Feature Learning)

28.5.2020

Motivation

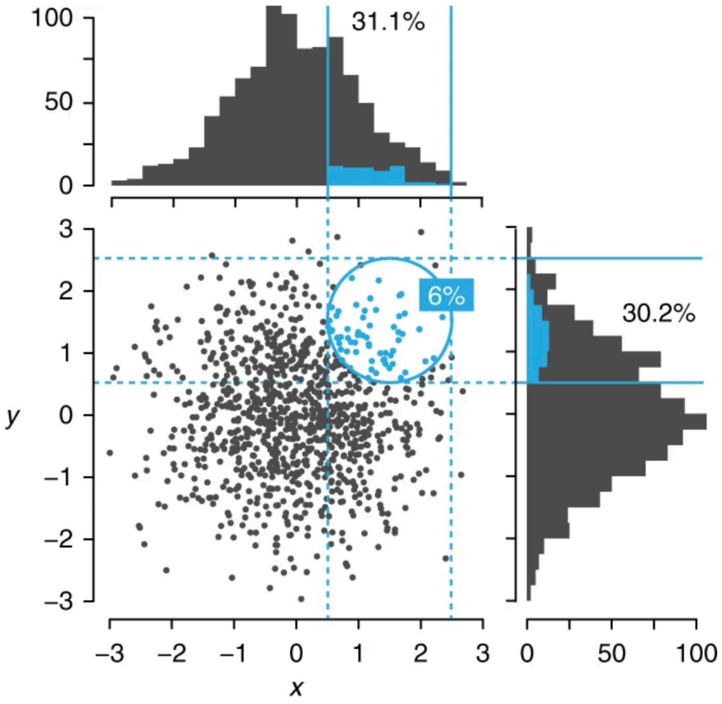
- Previously
  - Token-based models (eg. n-grams)
  - Discrete (small) vocabulary (eg. [a-z0-9], …)
  - More complex models used feature vector ($x \in \mathbb{R}^n$)
- $x \in \mathbb{R}^n$ is straight-forward for real-valued data (audio, video, …)
  - What about discrete (and large!) vocabularies?
  - Eg. Natural language (= words)?

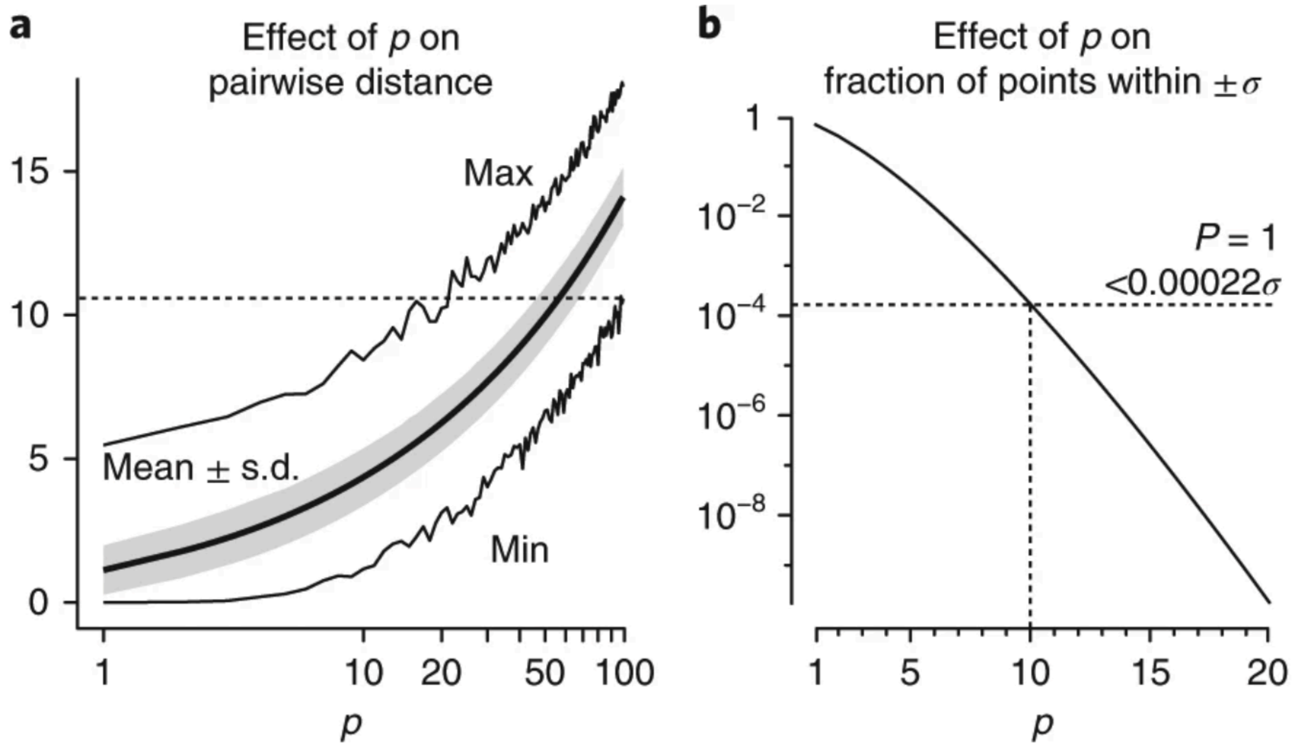
One-Hot Encoding

- Given fixed vocabulary $V = \{w_1, w_2, \dots, w_n\}$
- Set $x \in \mathbb{R}^{|V|}$ with $x_i = 1$ and $x_{j \neq i} = 0$ for word w_i
- *aka* word vector
- Drawbacks
  - Curse of dimensionality
  - Euclidean distance between points not necessarily semantic
  - Isolated words → loss of context

Curse of Dimensionality [1]

[1] Bellman, R. E. *Adaptive Control Processes: A Guided Tour, Ch. 5.16* (Princeton Univ. Press, Princeton, NJ, 1961)





[2] Altman, N., Krzywinski, M. The curse(s) of dimensionality. *Nat Methods* **15,** 399–400 (2018).
https://doi.org/10.1038/s41592-018-0019-x

Wanted: A mapping that…

- Can handle a large vocabulary
- Has a rather small output dimension
- Ideally…
  - Produces output values where (Euclidean) distances correlate with semantic distances
  - Incorporates the context of each token

Latent Semantic Indexing (1990)

- Key idea:
  Terms that occur in the same document should relate to each other
- Construct a term-occurrence matrix
- Find principal components using singular value decomposition
- Apply rank-reduction (ie. discard dimensions relating to smaller singular values)
- Use resulting matrix to map term vectors to lower-dim space
  - Works reasonably well for spam/ham, etc.
  - Context modelling limited to plain co-occurrence

Scott Deerwester, Susan Dumais, George Furnas, Thomas Landauer, Richard Harshman: *Indexing by Latent Semantic Analysis.* In: *Journal of the American society for information science.* 1990.

Word Embeddings (2003)

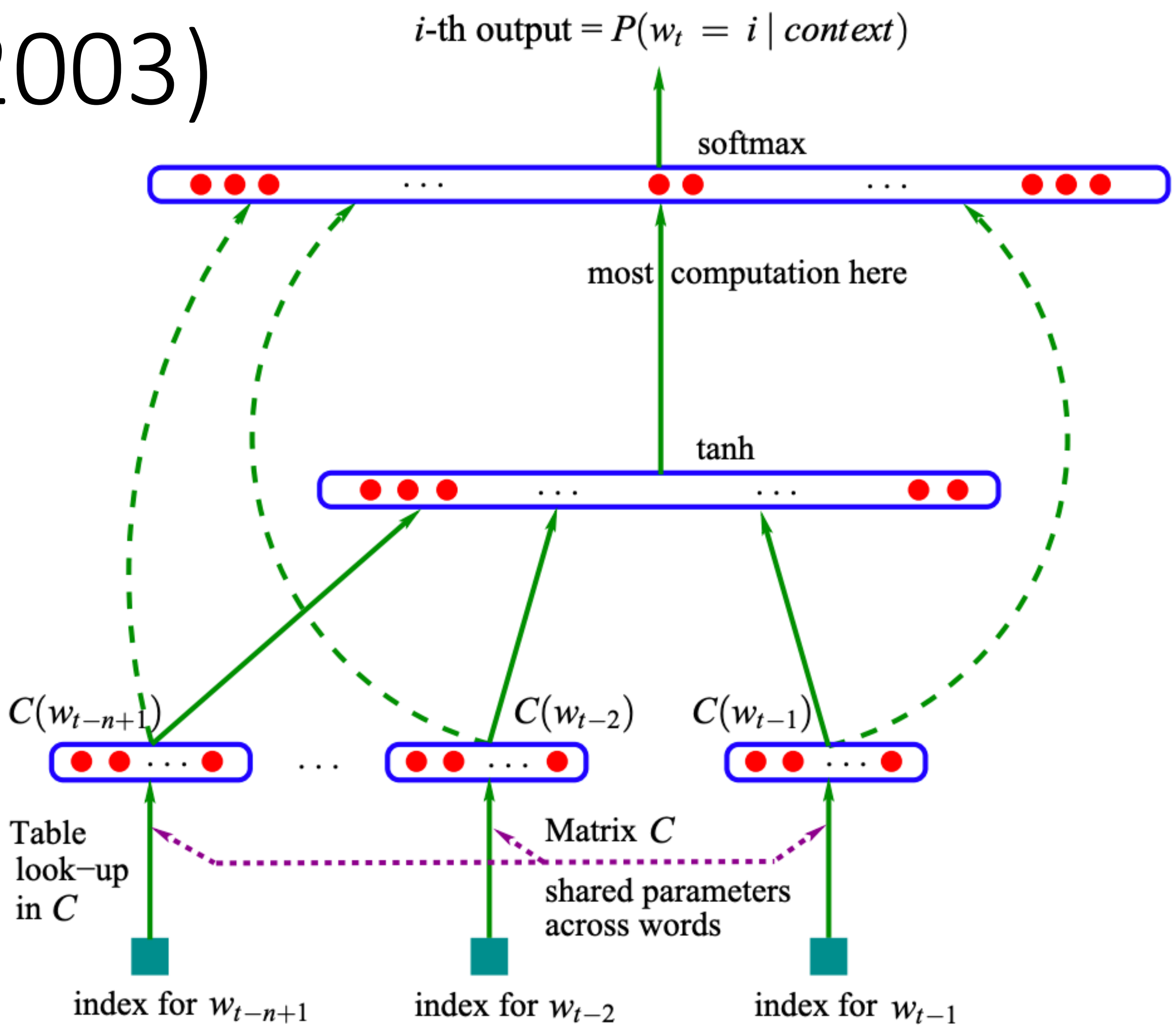
- Key idea:
  Use NN to predict next word

$$\hat{P}(w_t|w_{t-1},\cdots w_{t-n+1}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}}$$

$$y = b + Wx + U\tanh(d + Hx)$$

- Use shared "embedding" layer

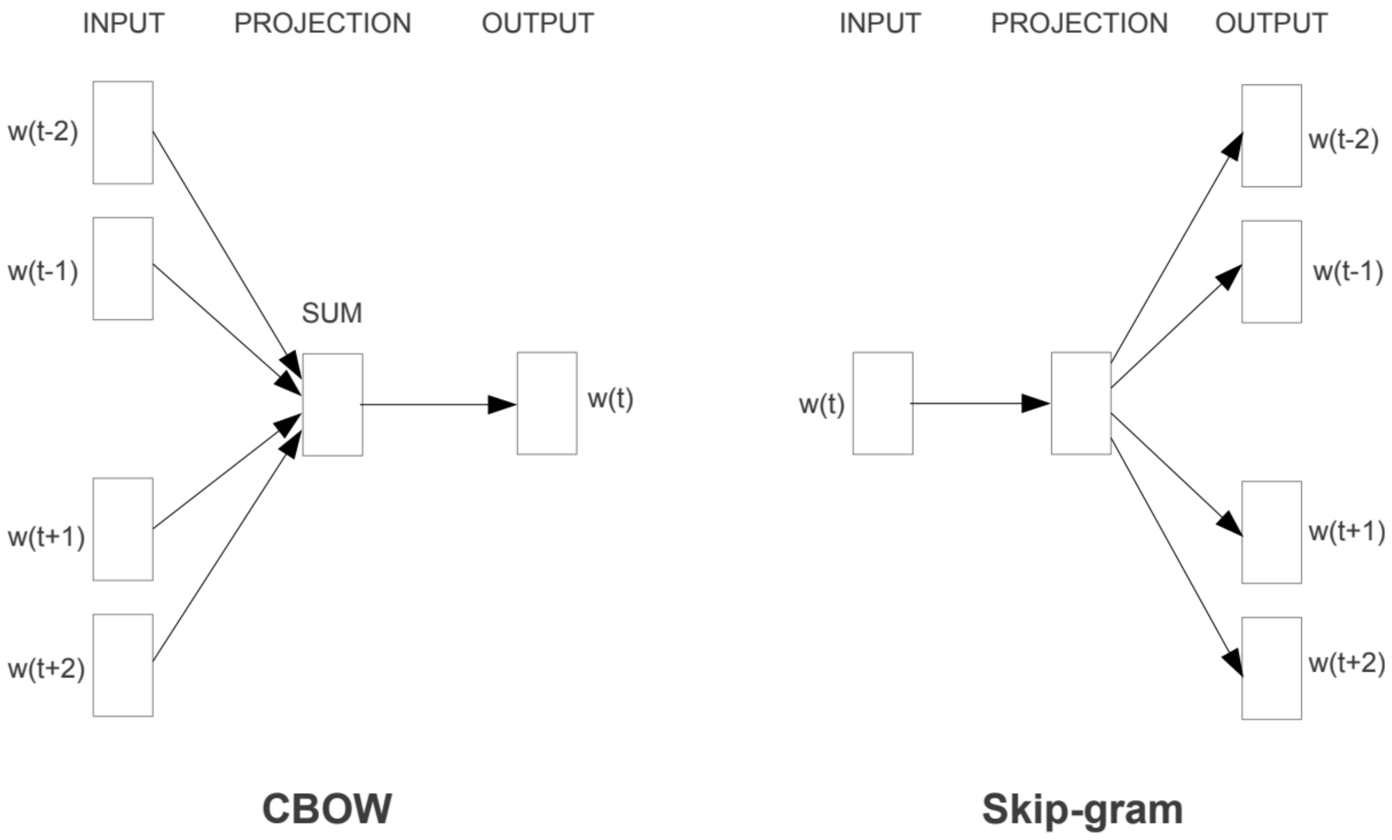
$$x = (C(w_{t-1}), C(w_{t-2}), \cdots, C(w_{t-n+1})).$$



Bengio, Y., Ducharme, R., Vincent, P., & Janvin, C. (2003). A Neural Probabilistic Language Model. The Journal of Machine Learning Research, 3, 1137–1155.

Word2Vec (2013)

- Avoid costly hidden layer
- Allow for more context
- Continuous Bag-of-Words (CBOW) uses context to predict center word
- Skip-gram predicts context from center word



Mikolov, T., Corrado, G., Chen, K., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. Proceedings of the International Conference on Learning Representations (ICLR 2013), 1–12.

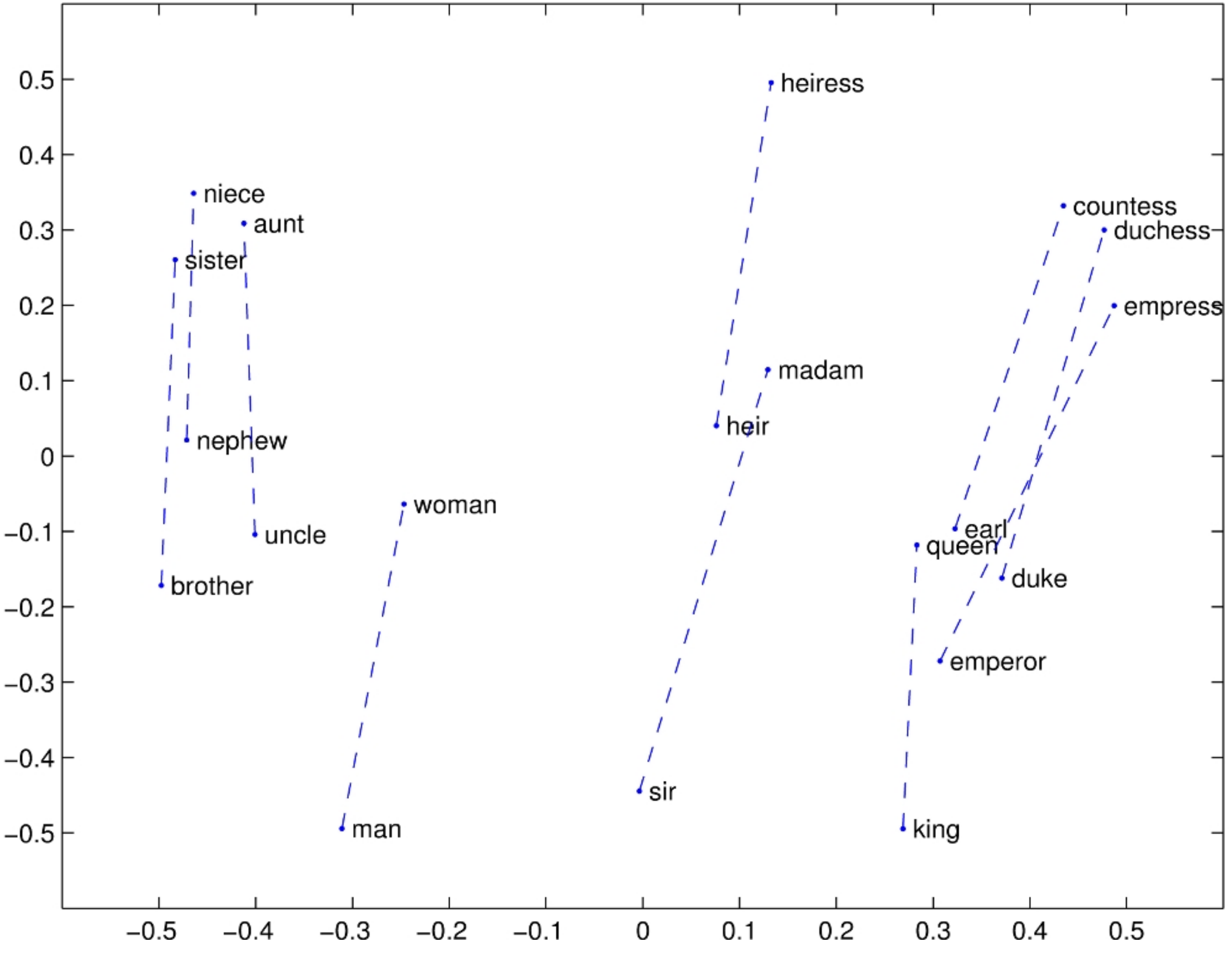
GloVe (2014)

- Based on word-word co-occurrence
- Minimize

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

word vectors

co-occurrence count

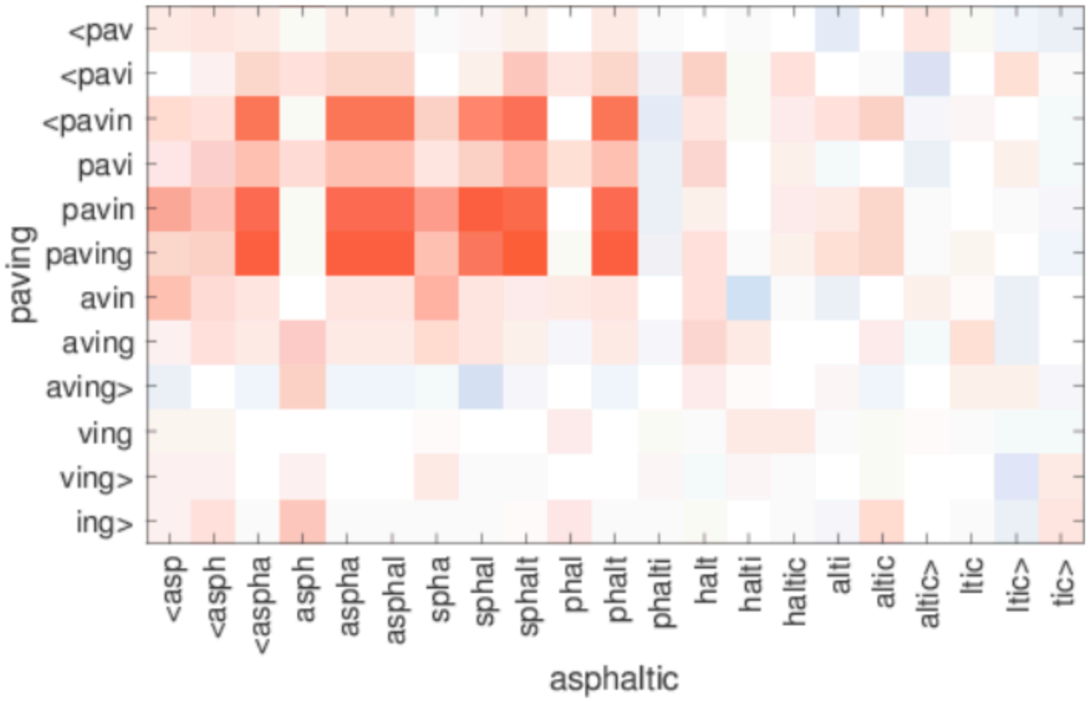


Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global Vectors for Word Representation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, 1532–1543.

FastText

- Previous word-based models struggle with OOV
  What to do, if an observed word is not in the vocabulary?

- Alternative:
  - Train on character n-grams instead
  - Use skip-gram approach
- Can handle OOV by averaging over known n-grams



Piotr Bojanowski, Edouard Grave, Armand Joulin and Tomas Mikolov. *[Enriching Word Vectors with Subword Information](#)*. 2016

Transfer Learning vs. Deep Learning

- Word2Vec, FastText, etc. can be trained on large amounts of unlabeled data
  - Ready-to-go models avaliable to map Words to feature vectors
  - Statistics can be updated using more (in-domain) data
- Most approaches can be modeled as computational graph
  → integrate models into training routines (with backprop)
- Most basic form: (single) embedding layer to map one-hot to smaller dimension (eg. Sparse layer in pytorch: https://pytorch.org/docs/stable/nn.html#embedding)