Machine Learning for Text Mining -introduction-

Antske Fokkens & Pia Sommerauer

CBS course

Supervised Machine Learning

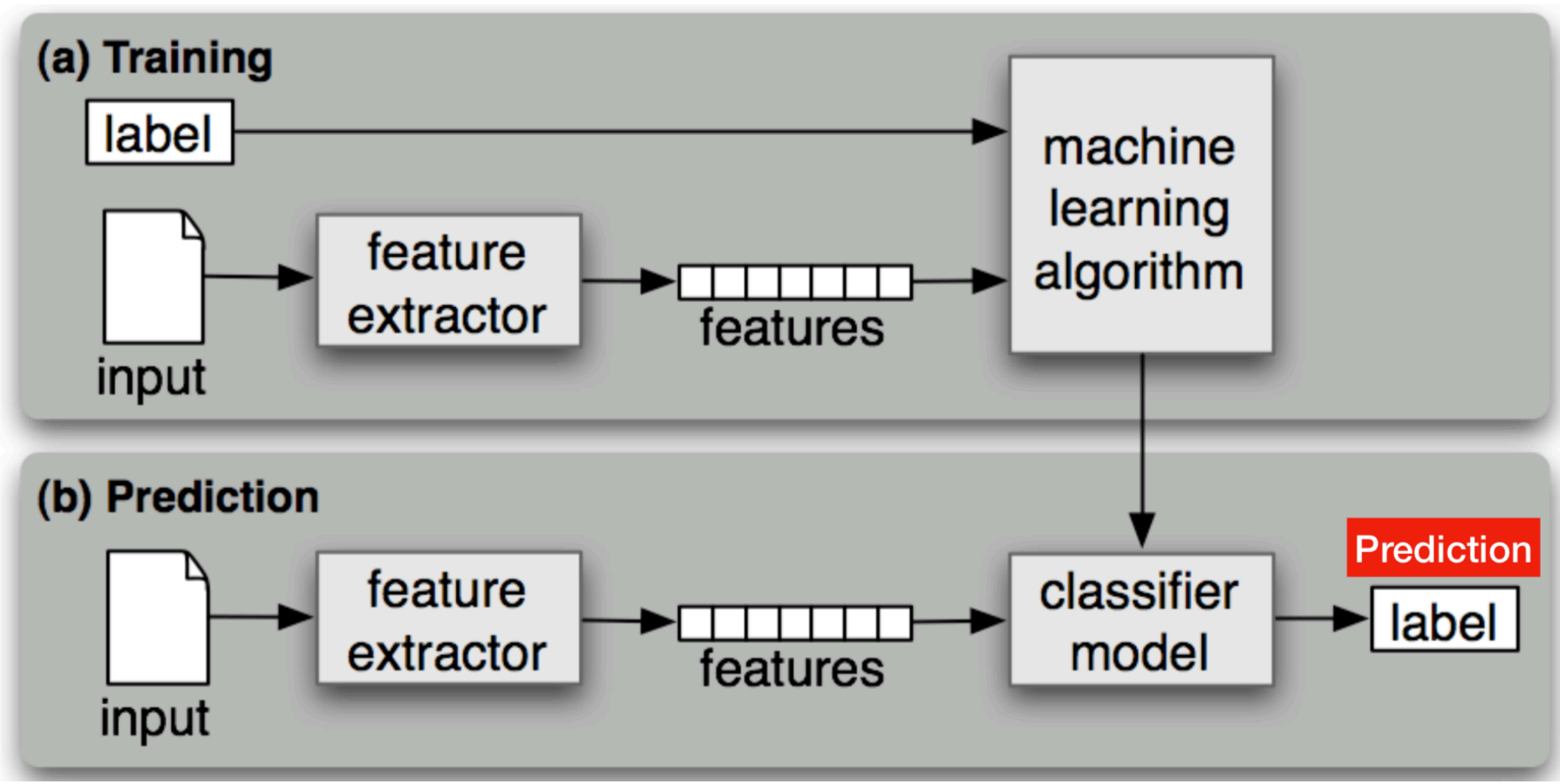
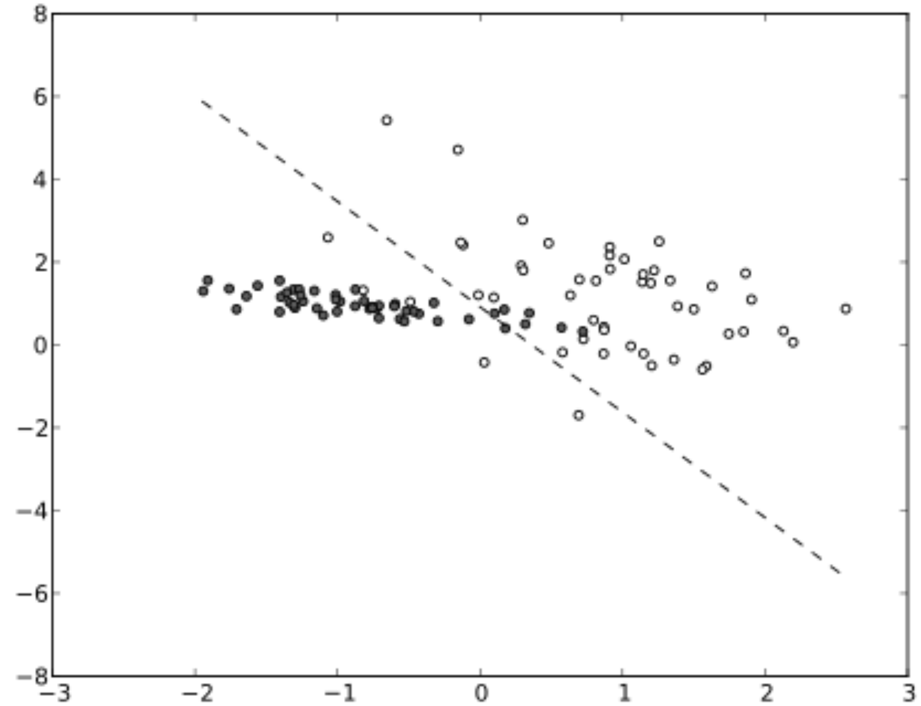


Image taken from Piek Vossen's slides (see lecture machine learning)

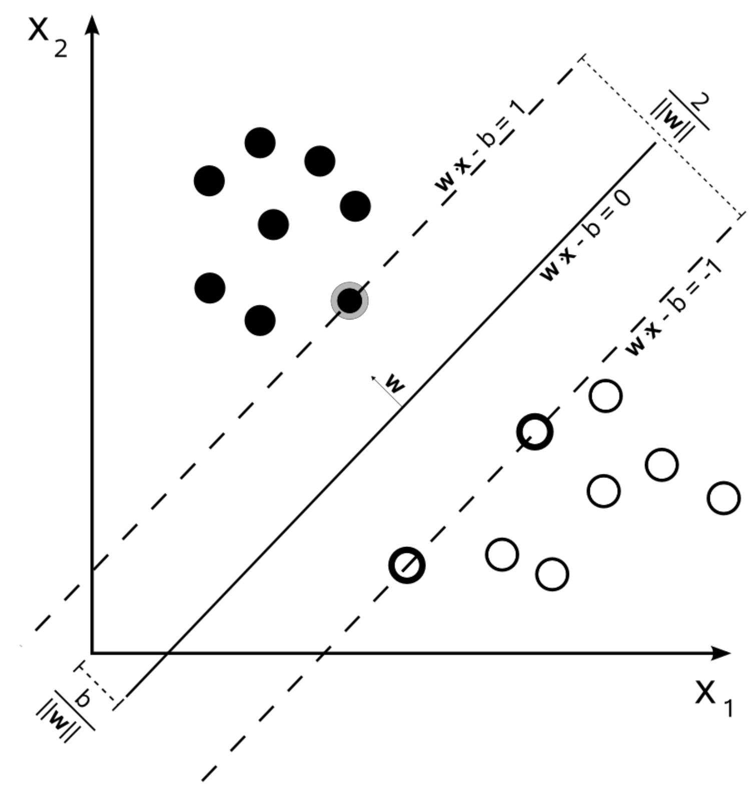
Translating to numbers

- Methods directly related to statistics (distribution & correlation)
- Many other forms: geometric representation & identify lines

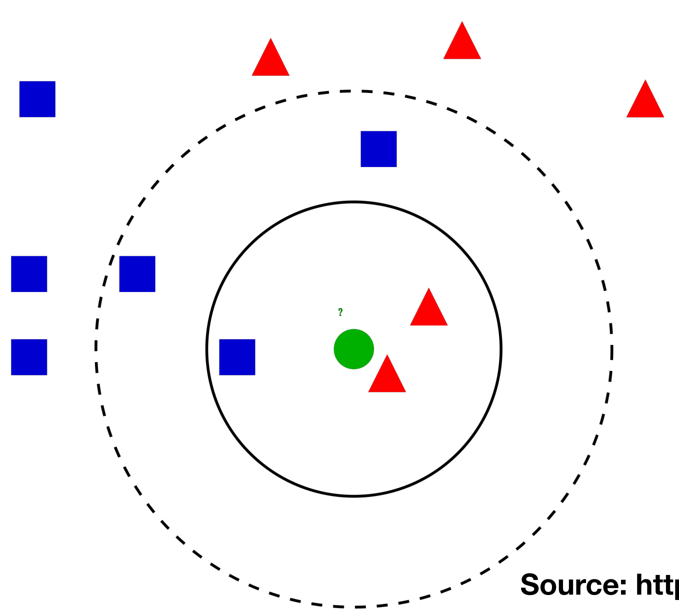
ML methods (examples)



Source: https://commons.wikimedia.org/wiki/File:Linear-svm-scatterplot.svg



Source: https://en.m.wikipedia.org/wiki/File:Svm_max_sep_hyperplane_with_margin.png



Source: https://commons.wikimedia.org/wiki/File:KnnClassification.svg

ML: basic overview

- Many approaches:
  1. represent features as vectors with numerical values
  2. predict class based on feature vector
- For example:
  - K-nearest neighbor: pick majority class of k-nearest points in space
  - Logistic regression: find hyperplane that separates the data best (minimizing loss)

Features

- Common features (for many tasks):
  - POS-tag
  - Word
  - Lemma
  - ngrams
- More advanced:
  - Chunks
  - Syntactic dependencies
  - Word sense

Features

- Common features (for many tasks):
  - POS-tag
  - Word
  - Lemma
  - ngrams

Highly Informative for many tasks!!!

- More advanced:
  - Chunks
  - Syntactic dependencies
  - Word sense

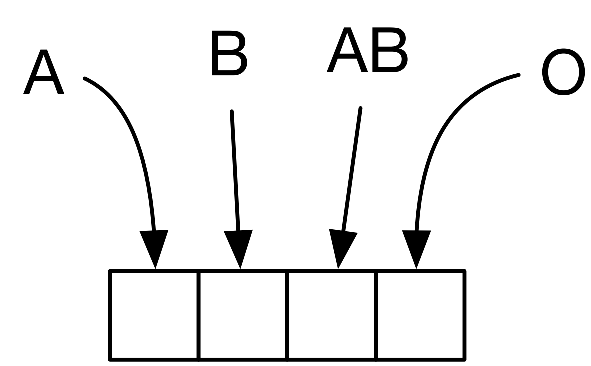
Input representation

- Many Machine Learning Classifiers use **input representations** that correspond to vectors
- How can we translate linguistic information to space?

Representations

- A number in one dimension, e.g. size of houses: represent size as number (e.g. square meters)

- Different dimensions allocated to various values, e.g. blood type of a person: represent value in a 4-dimensional vector



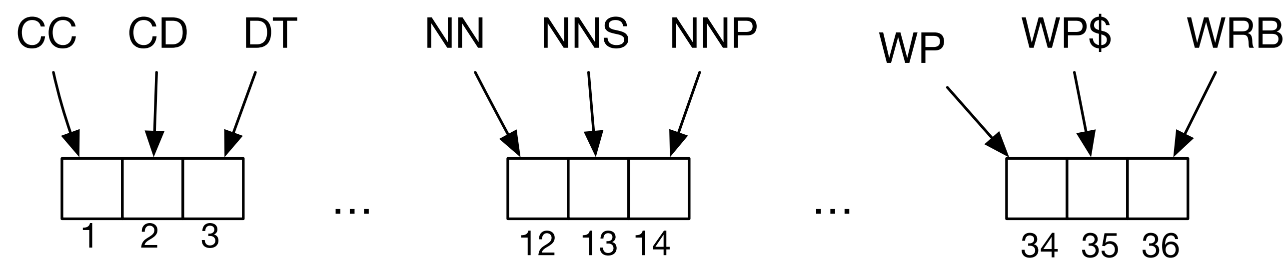
Linguistic features

How to represent?

- POS-tags
- Words/lemmas
- Chunks
- Dependencies

Penn Treebank Pos-tags

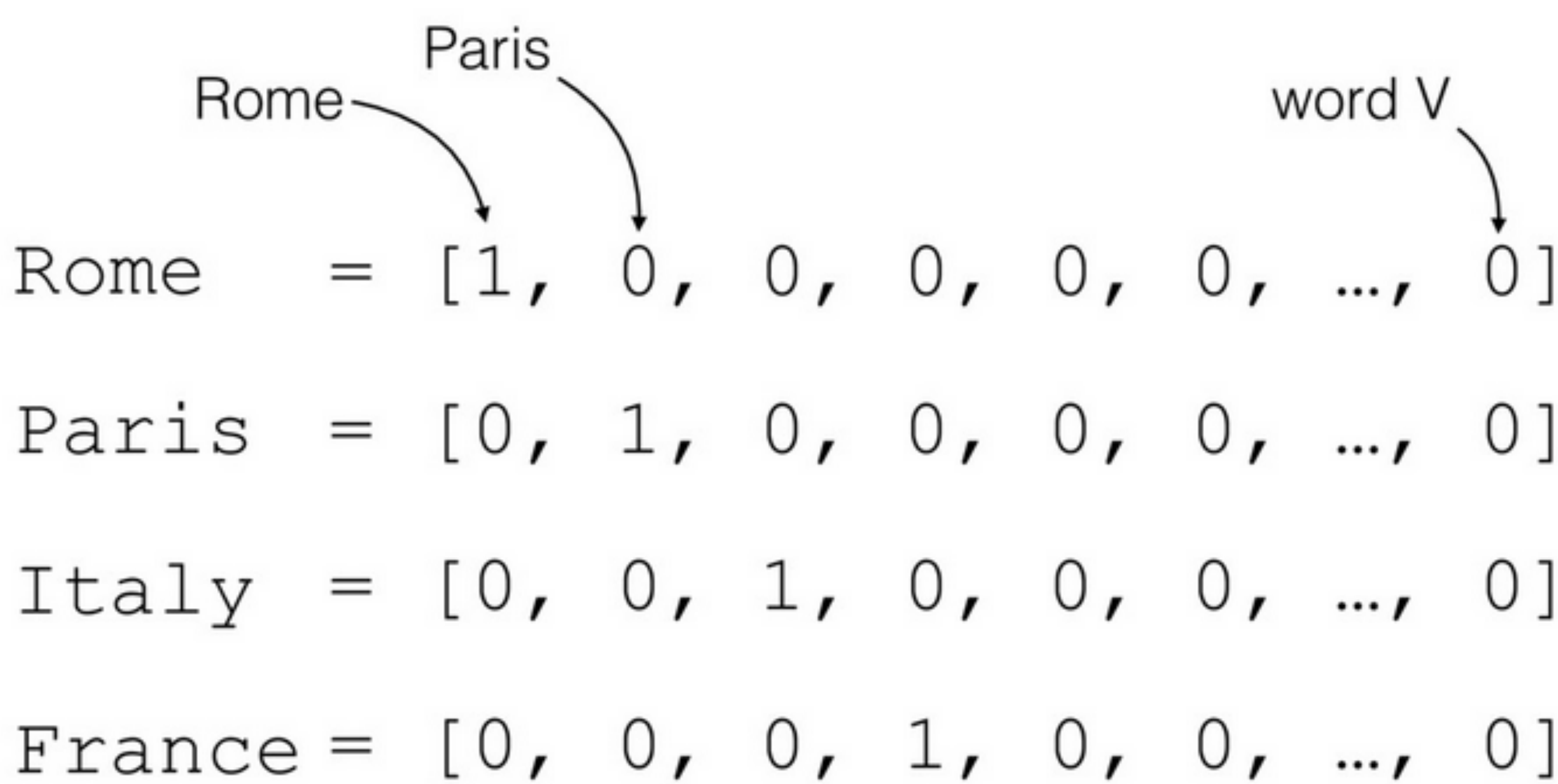
- 36 tags: 36 dimensions (one-hot representation)



CC	<1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0>
CD	<0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0>
DT	<0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0>
NN	<0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0>
NNS	<0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0>
NNP	<0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0>
WP	<0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0>
WP$	<0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0>
WRB	<0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1>

Word - Lemmas

- One-hot representations: vocabulary-size dimensions



Source image: Shaffy, Athif (2017) Vector Representation of Text for Machine Learning.
https://medium.com/@athif.shaffy/one-hot-encoding-of-text-b69124bef0a7

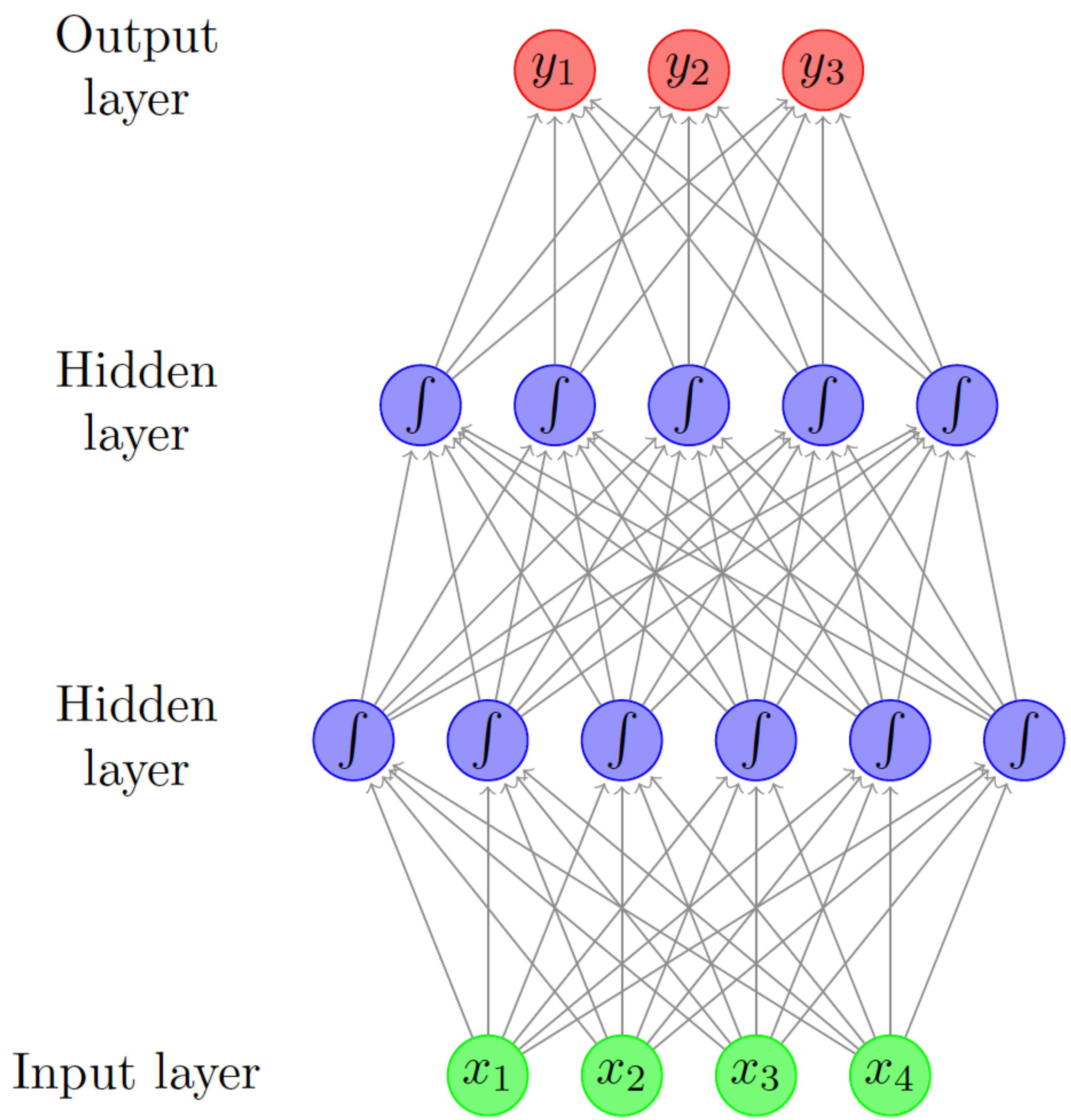
Word - Lemmas

- Word embeddings
- Lower-dimensional (~50-1,000), higher density representations
- Presentation based on context a word occurs in

  => words in similar context have similar representation
  => algorithm can gather evidence from similar words in training data

MORE ON WORD EMBEDDINGS LATER !

A Basic Neural network



from Goldberg (2015)

Brain Analogy

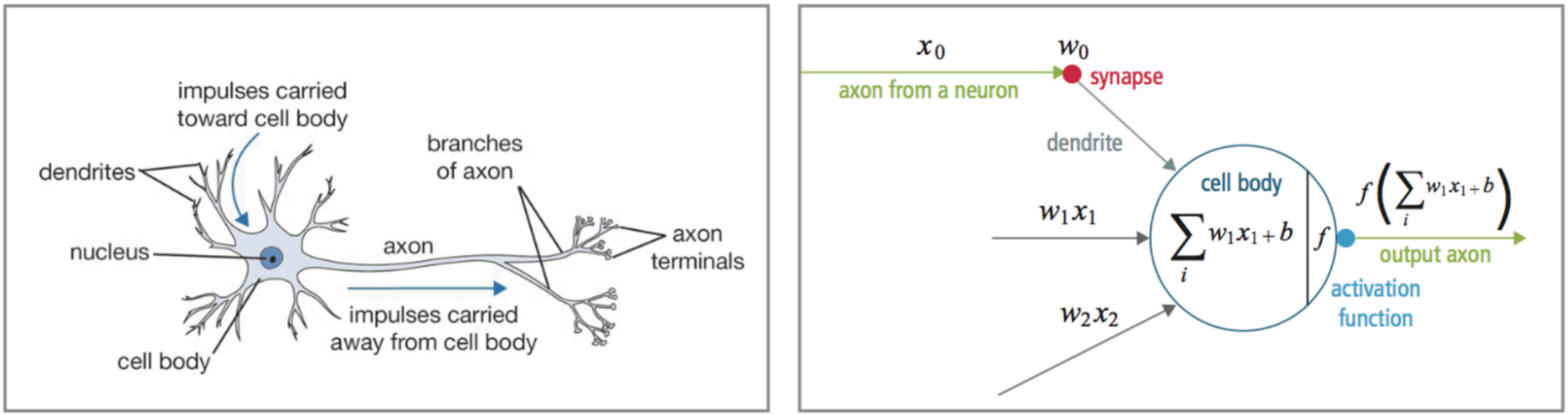
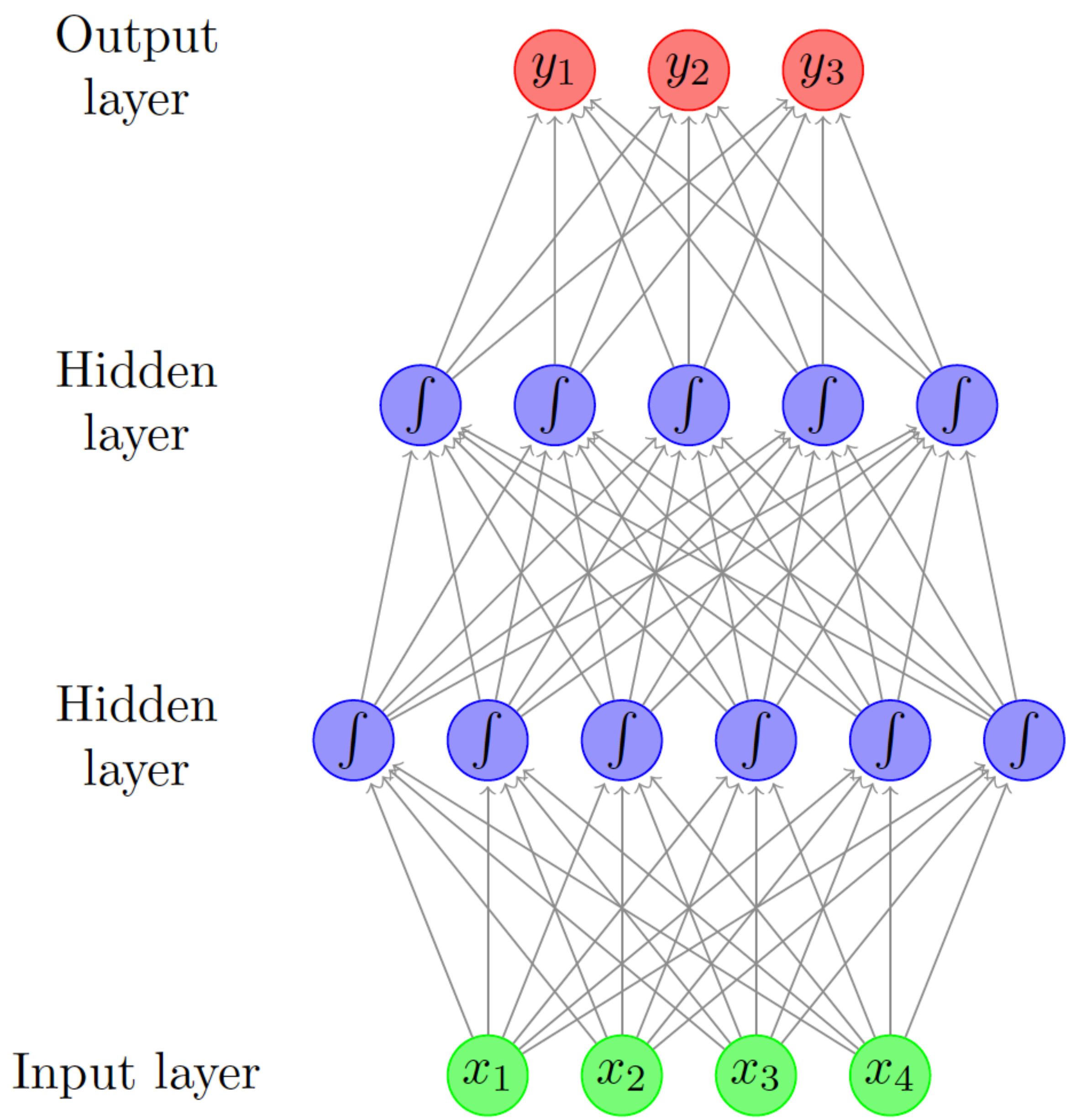


Figure 3: Illustration of a biological neuron (left) and its mathematical model (right) [2].

+ non-linear function

By: Andrej Karapthy 2015

A Basic Neural network



from Goldberg (2015)

Requirements

- workable size of feature space:
  - limited feature space

    => new representations of words
    => using research on lexical semantics

Word Embbedings

PPMI

- PPMI = $argmax(0, log(\frac{P(w1,w2)}{P(w1)P(w2)}))$

PPMI

- High dimensional (size of vocabulary)
- Low density (zeros for all context-words occurring less than by chance)
- Relatively high impact of low frequency words

Singular Value Decomposition (SVD)

- Method to reduce the number of dimensions:

  given a m x n matrix, construct a m x k matrix,
  where $k << n$

- Uses linear algebra to reduce the number of dimensions, preserving most of the variance of the original matrix

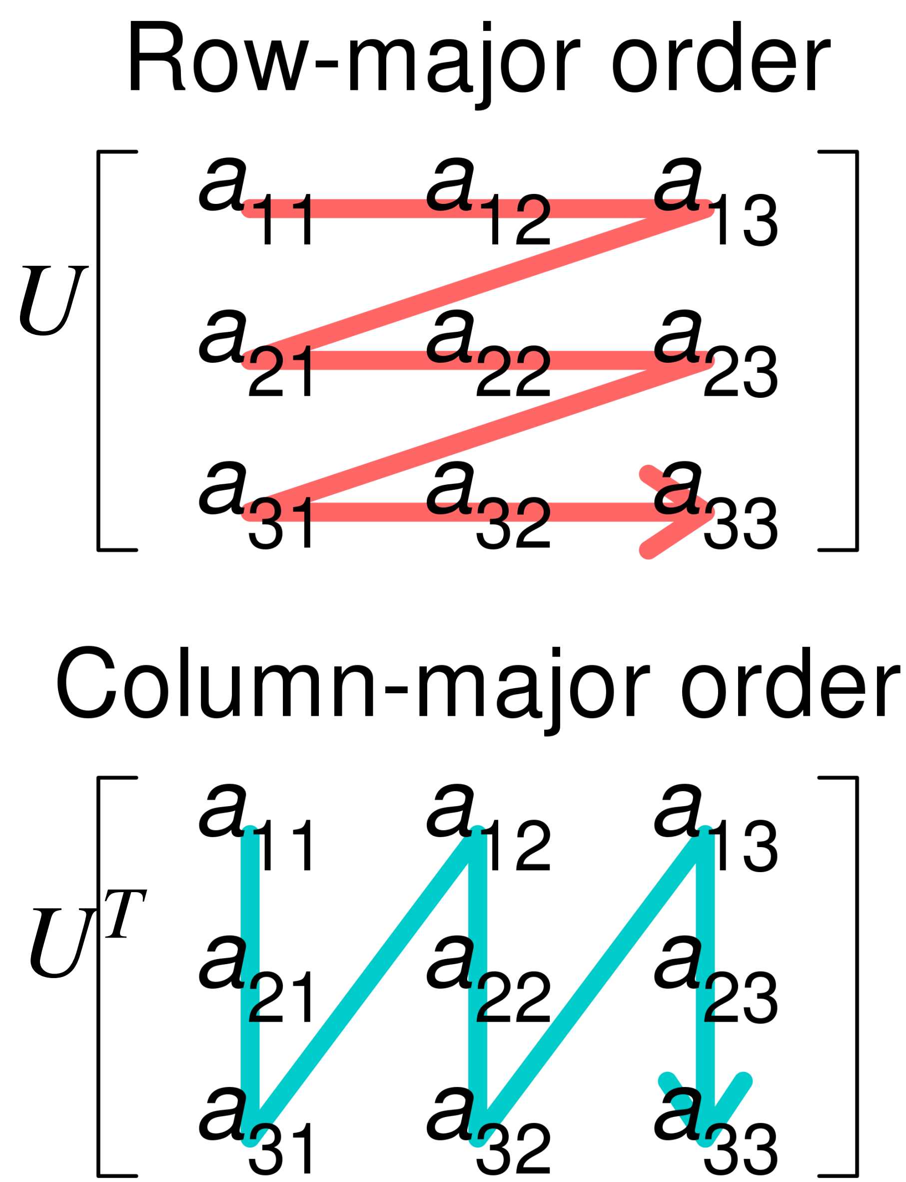
SVD

- A matrix *A* can be broken down (decomposed) into the product of three matrices:

$$A = U\Sigma V^T$$

- where *U* and *V* are orthogonal
- The columns of *U* are orthonormal eigenvectors of AA^T
- The columns of *V* are orthonormal eigenvectors of A^TA
- Σ is a diagonal matrix containing square roots of eigenvalues from *U* or *V* in descending order

U and V are orthogonal

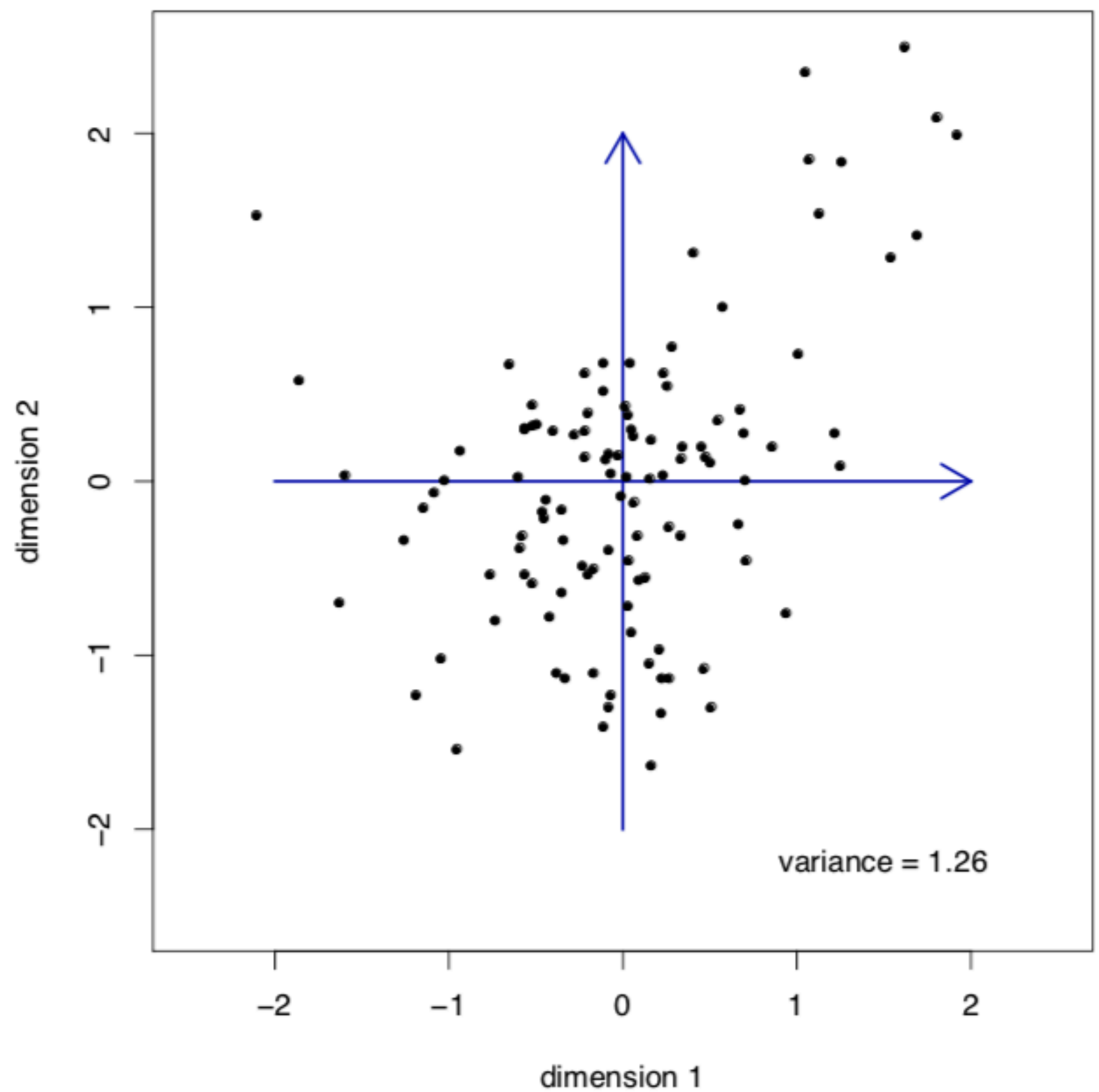


$$UU^T = I$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

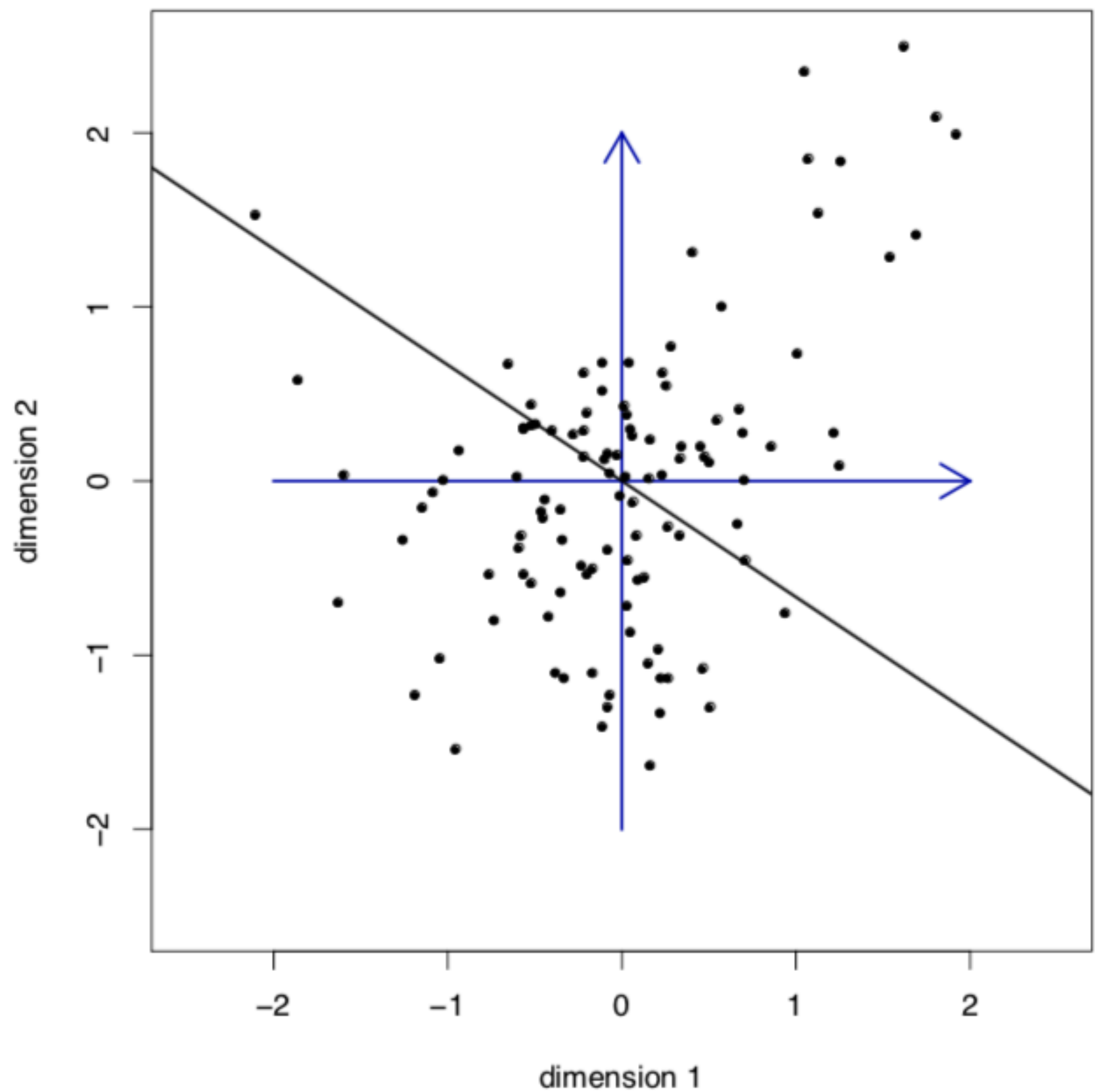
Left image from: https://en.wikipedia.org/wiki/Matrix_representation

Capturing most variance



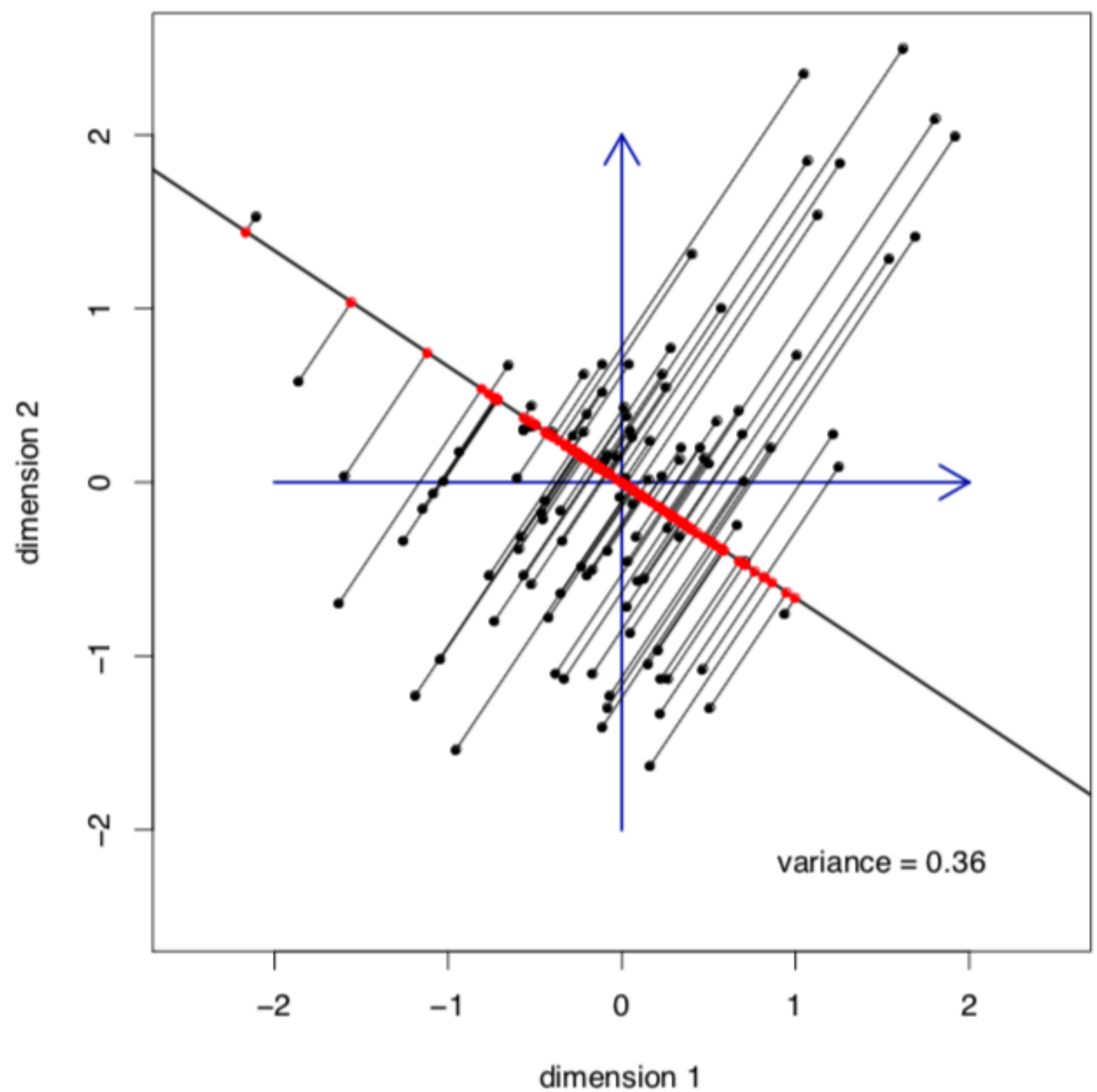
from Baroni & Boleda

Capturing most variance



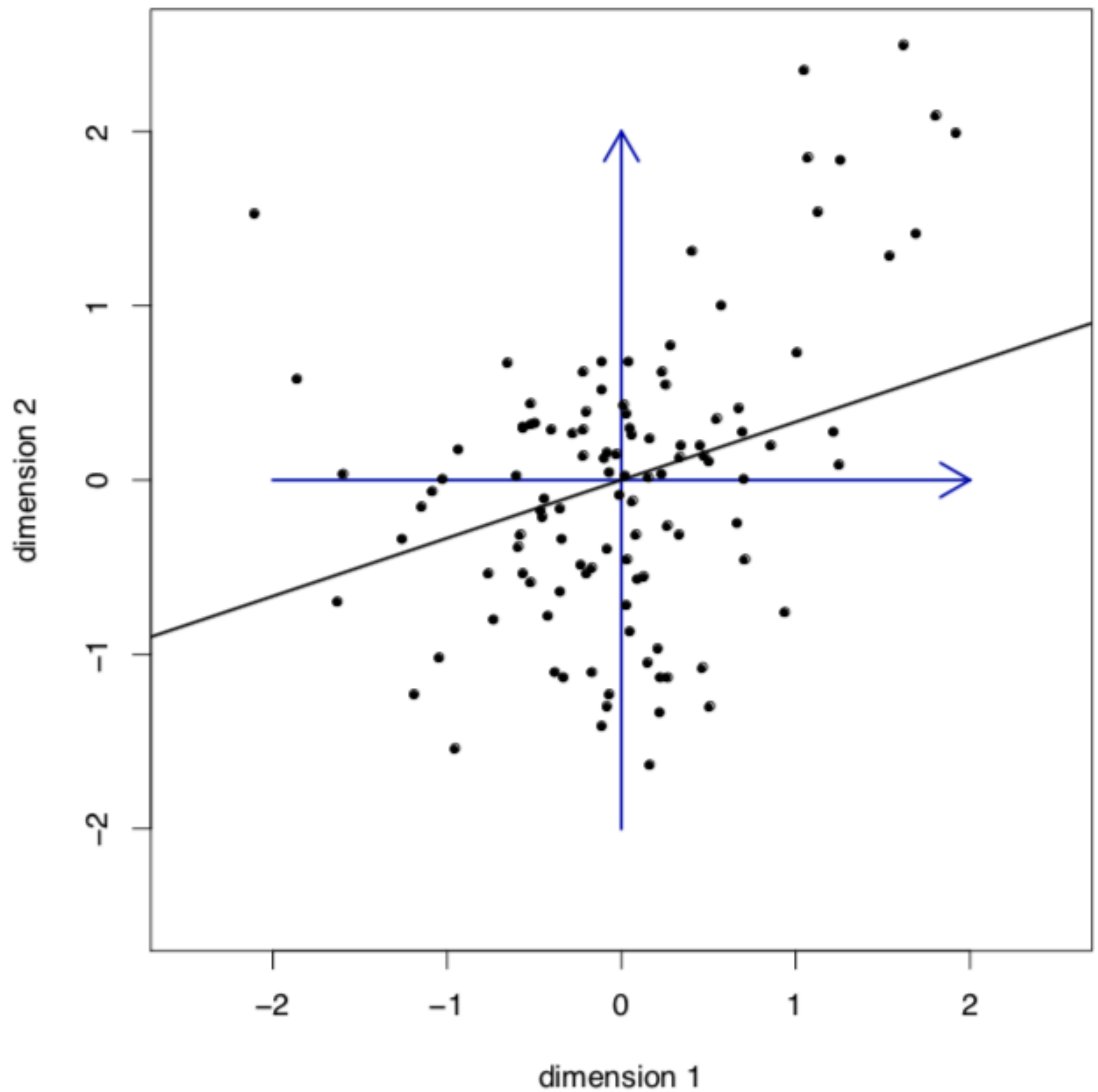
from Baroni & Boleda

Capturing most variance



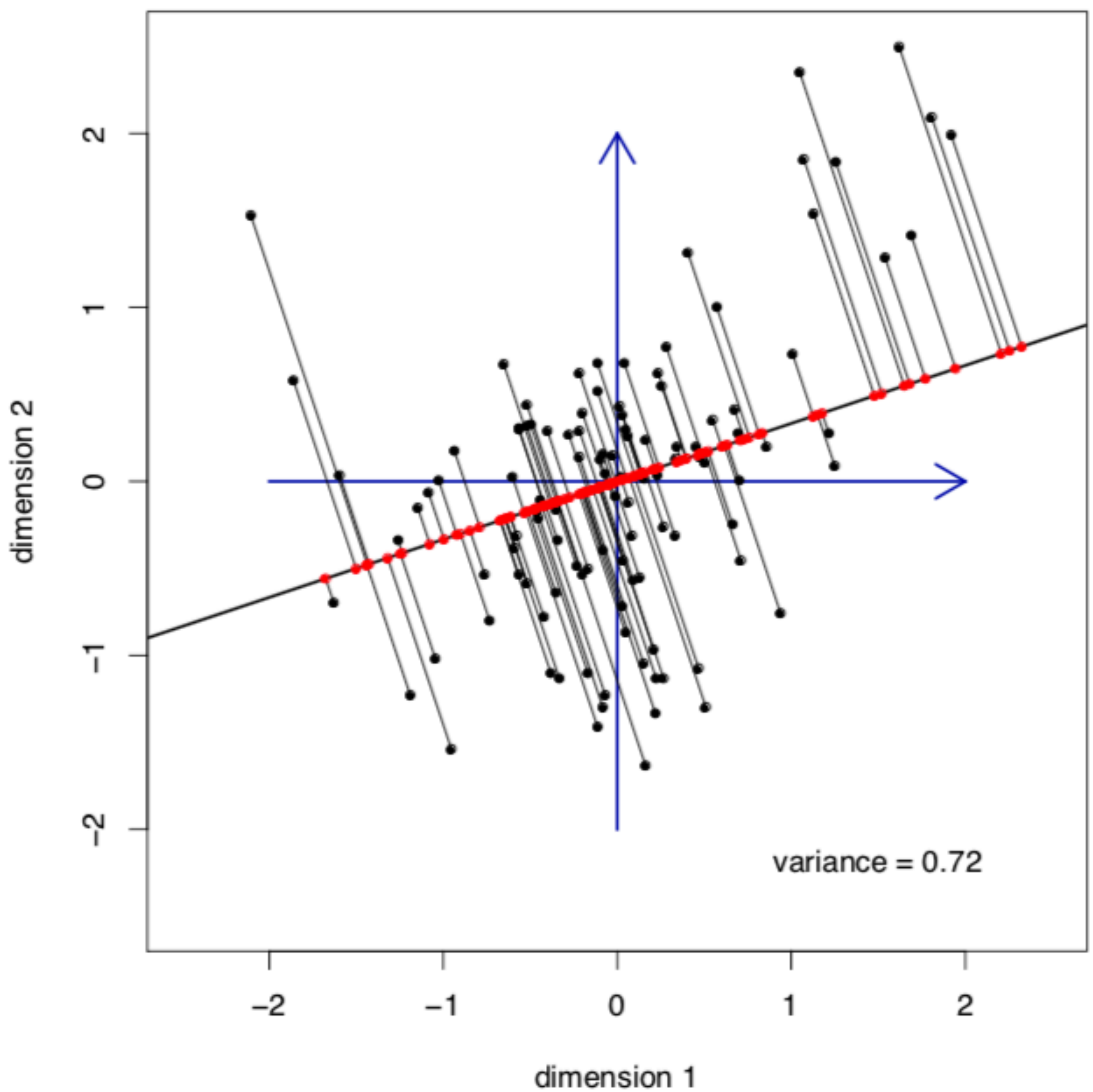
from Baroni & Boleda

Capturing most variance



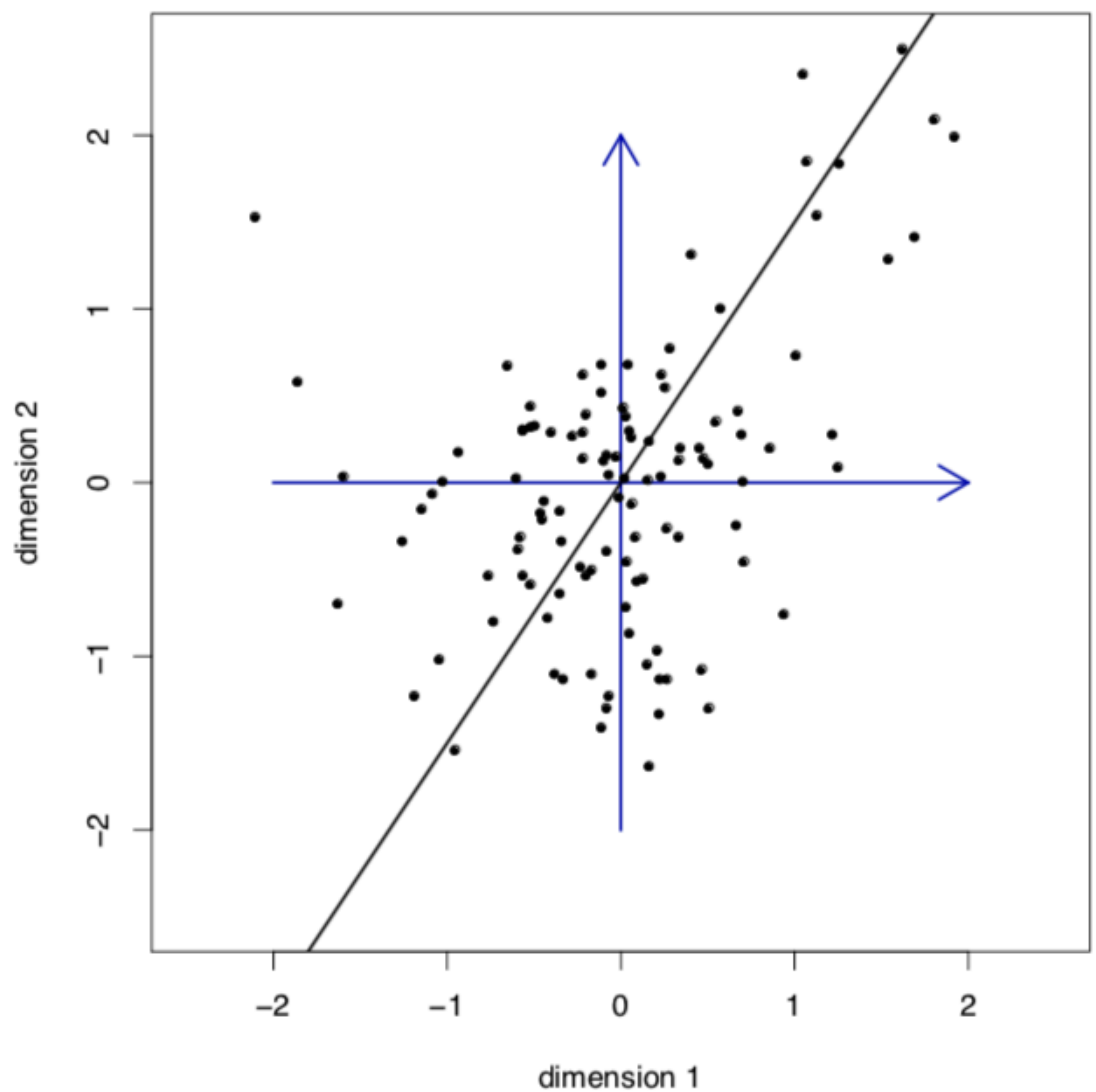
from Baroni & Boleda

Capturing most variance



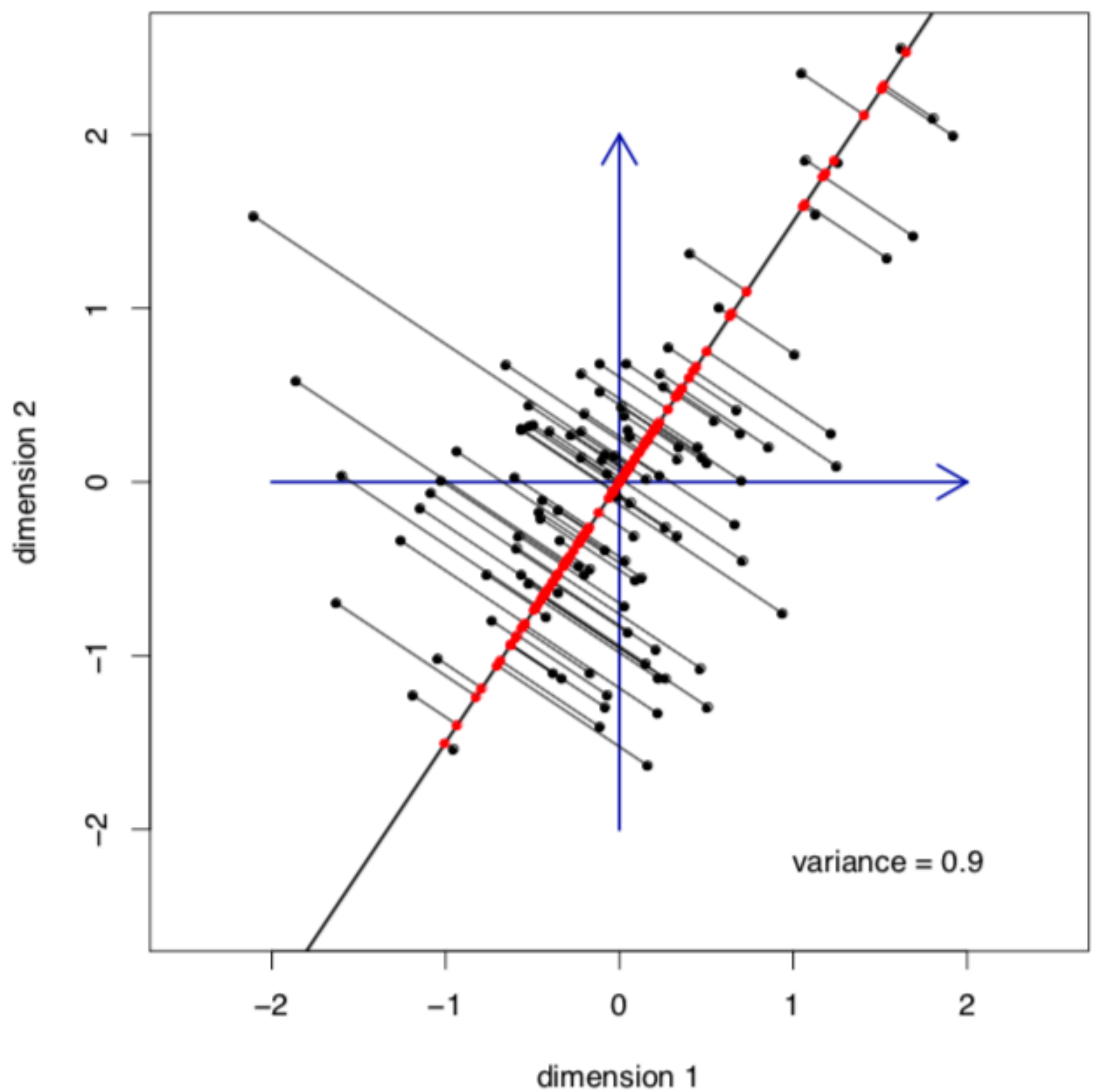
from Baroni & Boleda

Capturing most variance



from Baroni & Boleda

Capturing most variance



from Baroni & Boleda

Reducing Dimensions

- Columns of *U* and *V* are ordered according to highest associated eigenvalue

- Diagonal values of Σ are ordered starting with highest (root of) eigenvalues

  => Using the first *d* rows of *U,* the first *d* columns of V^T and *d*x*d* rows and columns of Σ guarantees that we end up with those values that provide the highest variance.

Predictive Models

- PPMI: captures desirable properties, but is still sparse and high dimensional

- SVD: generalizes better is high density and lower dimensions, but is inefficient to obtain (and does not perform perfectly)

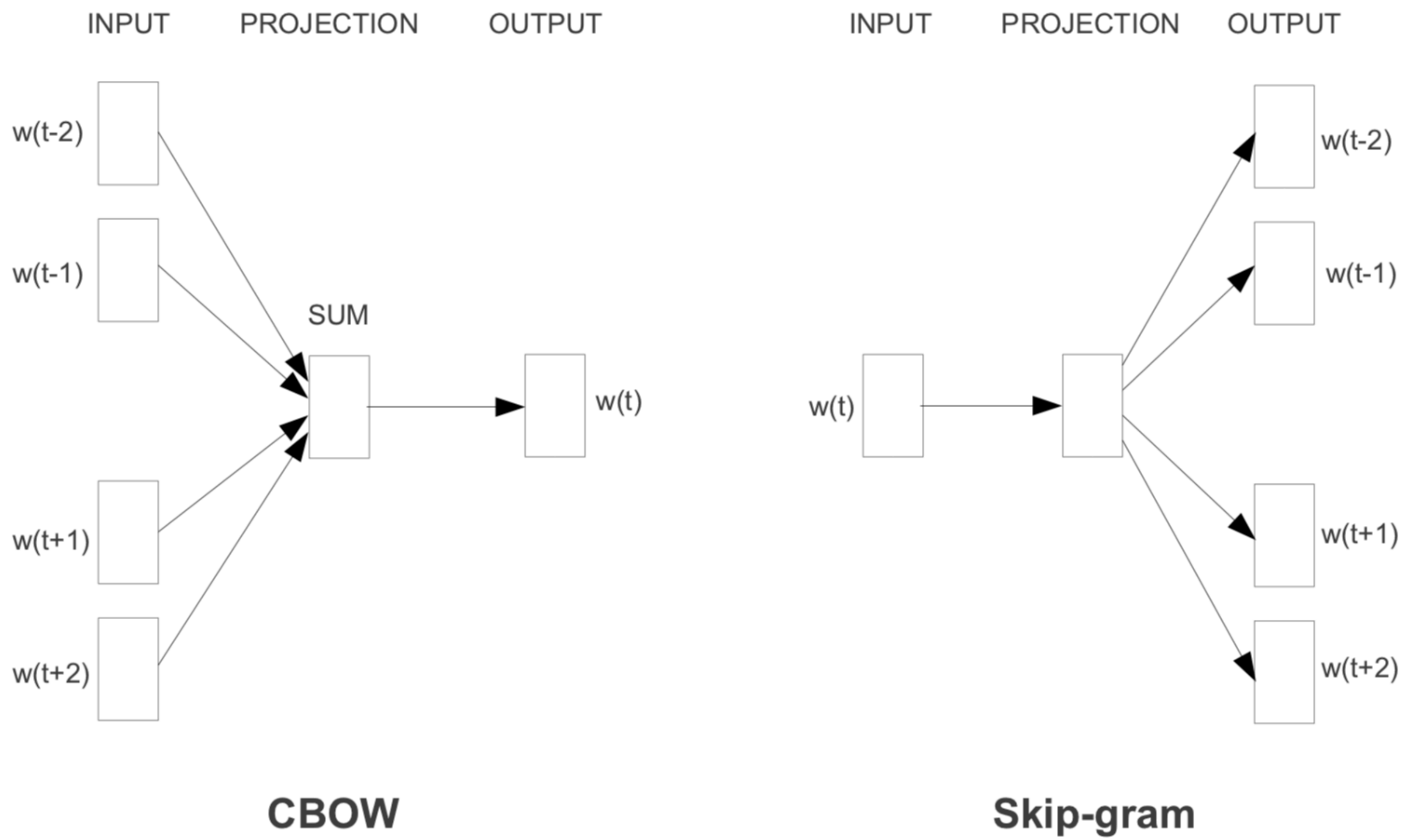
- Alternative idea: use language modeling as an auxiliary task for creating word embeddings

  => machine learning to **predict** which words occur in each other's context

Predictive Models

- word2vec (Mikolov et al. 2013a,b):
  - Several methods for creating embeddings
  - All start from randomly initiated vectors with a preset number of dimensions
  - Two vocabulary matrices: one for the words, one for contexts
    - Models: CBOW & Skipgram
    - Training: hierarchical softmax & negative sampling
    - Preprocessing: dynamic context windows, subsampling, delete rare words

CBOW vs Skipgram



from Mikolov et al. (2013)

Training

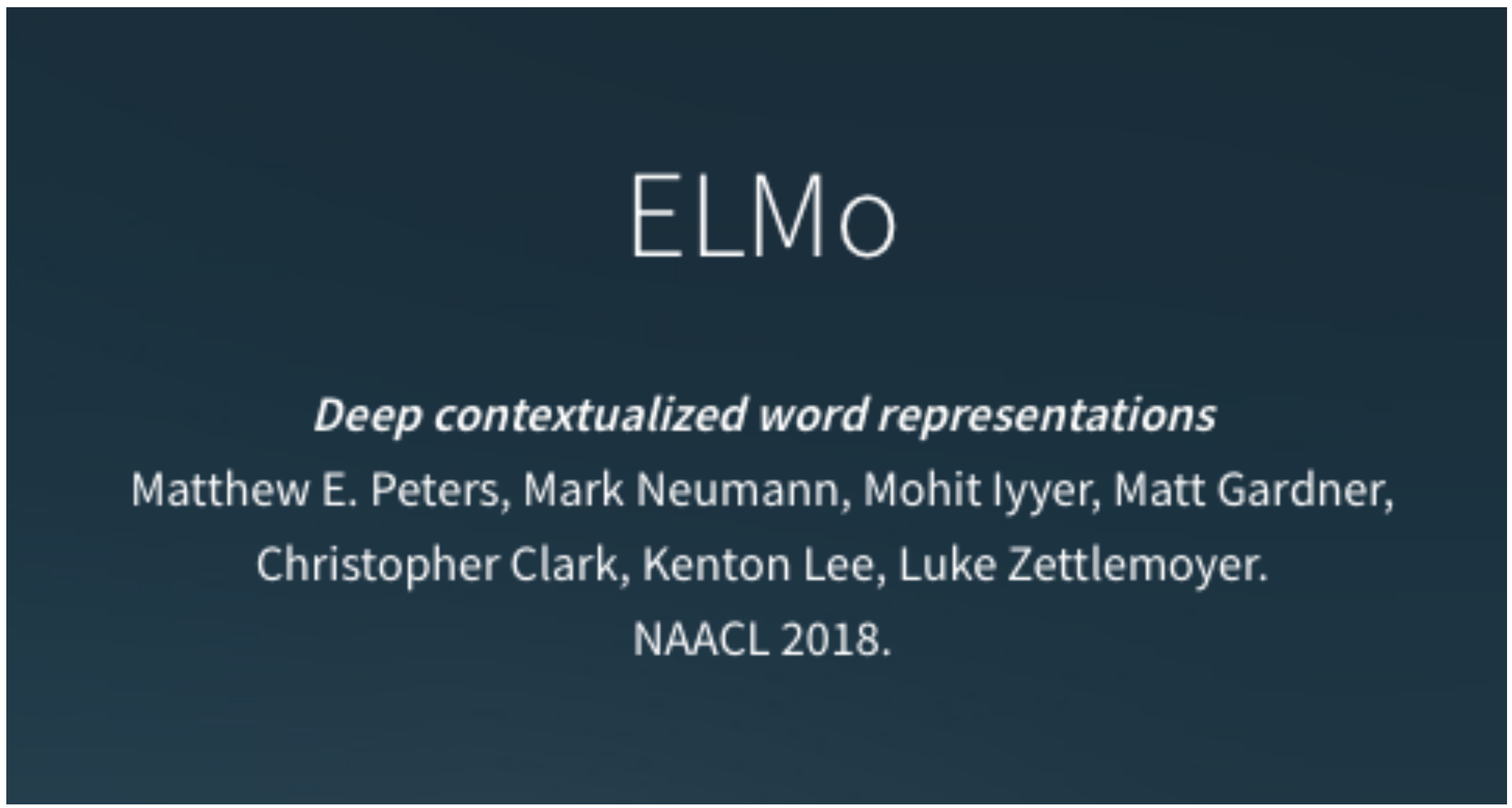
- Hierarchical softmax: efficient way to determine most probable context given a word (or vice versa) over the whole model

- Negative sampling: distinguish the actual context words from *k* other words (randomly chosen)

Result

- *d*-dimensional word & context embeddings
- distributional model: the vectors representing words are kept

Latest Developments



https://allennlp.org/elmo

- Easy to use (with advanced context selection & neural network for learning)
- State-of-the-art for many NLP tasks

Word Embeddings as features

- Suitable to be used as input in neural networks
- Generalize better => enhance many tasks, also when used with other machine learning approaches
- Ongoing research: which embeddings (how created) for which task?

  => dependency parsing
  => sentiment analysis

Sources

- Baroni & Boleda. Distributional Semantic Models https://www.cs.utexas.edu/~mooney/cs388/slides/dist-sem-intro-NLP-class-UT.pdf

- Goldberg, Yoav (2015) A primer on neural network models for Natural Language Processing

- Mikolov, T., K. Chen, G. Corrado and J. Dean (2013) Efficient estimation of word representations in vector space. https://arxiv.org/pdf/1301.3781.pdf

- Shaffy, Athif (2017) Vector Representation of Text for Machine Learning. https://medium.com/@athif.shaffy/one-hot-encoding-of-text-b69124bef0a7

- Pia Sommerauer. What is in a word embedding vector?

- All reported sources were accessed between January 14-16 2019