Text Analysis in R

Lecture 3: Analysis of text data, pt. II

Helge Liebert

Outline

Vector semantics and distributional models of meaning

Recall: Term-document-matrix

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	5	117	0	0

Figure 15.1 The term-document matrix for four words in four Shakespeare plays. Each cell contains the number of times the (row) word occurs in the (column) document.

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	5	117	0	0

Figure 15.2 The term-document matrix for four words in four Shakespeare plays. The red boxes show that each document is represented as a column vector of length four.

Information retrieval

- Term-document-matrices initially defined as a means of finding similar documents for information retrieval.
- Finding similar documents is a major use case in industry (e.g. product recommendations).
- Example: Literature Map, Global Network of Discovery.
- Retrieving similar documents/observations also useful for research.
- In other contexts, retrieval of words similar to other words is desired (e.g. translation, thesaurus).

Recall: TDM vector space representation

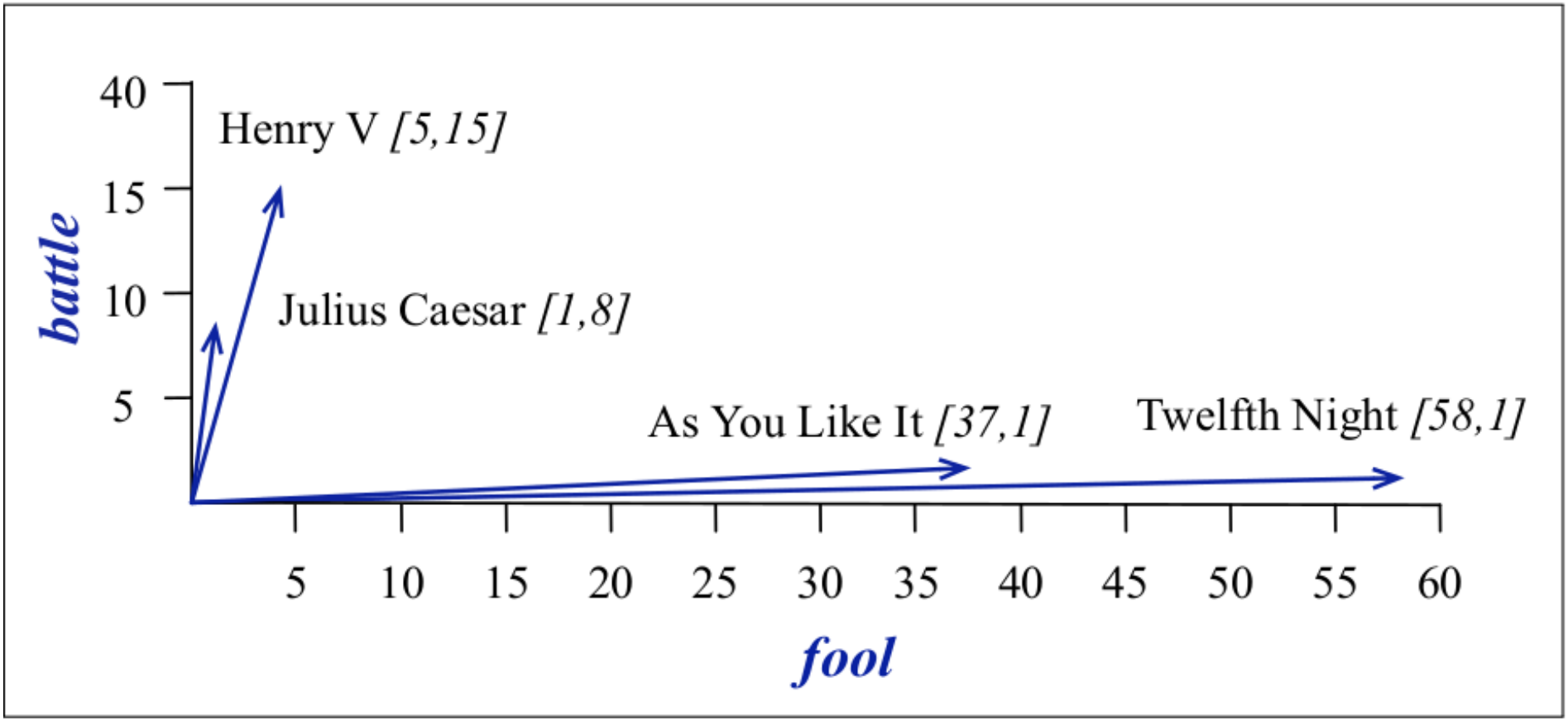


Figure 15.3 A spatial visualization of the document vectors for the four Shakespeare play documents, showing just two of the dimensions, corresponding to the words *battle* and *fool*. The comedies have high values for the *fool* dimension and low values for the *battle* dimension.

Vector space representations

- Documents can be represented in a vector space.
- Using a 1-gram representation, the dimension of the vector space will be the length of the corpus vocabulary, $|\mathcal{V}|$.
- Similar documents will have similar words, i.e. similar column vectors.
- The reverse holds true: Similar words occur in similar documents, i.e. similar row vectors.

Vector semantics

- Words can be encoded as vectors as well.
- Much of the recent advances in NLP applications can be attributed to better vector representations of words.

  "You shall know a word by the company it keeps."

  (Firth 1957)

- Distributional semantics: A word's meaning is given by the words that frequently appear close-by.

Vector semantics

- Words can be encoded as vectors as well.
- Much of the recent advances in NLP applications can be attributed to better vector representations of words.

  "You shall know a word by the company it keeps."

  (Firth 1957)

- Distributional semantics: A word's meaning is given by the words that frequently appear close-by.

➥ Shift the focus for now: How do we best encode the meaning of *words*?

(Most methods in this chapter can be applied to both document or word-representations. Circle back to document representations later.)

Distributional models of meaning

A bottle of tesgüino is on the table.
Everybody likes tesgüino.
Tesgüino makes you drunk.
We make tesgüino out of corn.

Distributional models of meaning

```
A bottle of tesgüino is on the table.
Everybody likes tesgüino.
Tesgüino makes you drunk.
We make tesgüino out of corn.
```

- Tesgüino means a fermented alcoholic drink made from corn.
- The words in the text and other similar context words also occur around the words ‘beer’, ‘liquor’ or ‘tequila’.
- Helps us discover the similarity between these words and tesgüino.
- More sophisticated syntactic features: ‘occurs before drunk’, ‘occurs after bottle’, ‘is the direct object of like’.

Distributional models of meaning

```
A bottle of tesgüino is on the table.
Everybody likes tesgüino.
Tesgüino makes you drunk.
We make tesgüino out of corn.
```

- Tesgüino means a fermented alcoholic drink made from corn.
- The words in the text and other similar context words also occur around the words ‘beer’, ‘liquor’ or ‘tequila’.
- Helps us discover the similarity between these words and tesgüino.
- More sophisticated syntactic features: ‘occurs before drunk’, ‘occurs after bottle’, ‘is the direct object of like’.

➥ Meaning of a word is related to the distribution of words around it.

➥ Distributional methods represent words as vectors of numbers (‘word embeddings’). Generally based on co-occurence matrices or predictions.

Vector semantics

- Rather than considering word similarity at the document level, we will consider the context of words.
- Use term-context (term-term, word-word) matrices of dim. $|\mathcal{V}| \times |\mathcal{V}|$ or $|\mathcal{V}_w| \times |\mathcal{V}_c|$.
- Represent a word as being embedded in a vector space.
- Each cell measures the co-occurence of the row and columns words in some context.
- Context can be the document, but typically smaller contexts are used (+/- 3/4 words).
- Techniques (e.g. tf-idf, cosine similarity, SVD ...) can be applied to both term-document and word-word matrices equally.

Context example

is traditionally followed by	cherry	pie, a traditional dessert
often mixed, such as	strawberry	rhubarb pie. Apple pie
computer peripherals and personal	digital	assistants. These devices usually
a computer. This includes	information	available on the internet

Term-context matrix

	aardvark	...	computer	data	result	pie	sugar	...
cherry	0	...	2	8	9	442	25	
strawberry	0	...	0	0	1	60	19	
digital	0	...	1670	1683	85	5	4	
information	0	...	3325	3982	378	5	13	

Figure 6.5 Co-occurrence vectors for four words in the Wikipedia corpus, showing six of the dimensions (hand-picked for pedagogical purposes). The vector for *digital* is outlined in red. Note that a real vector would have vastly more dimensions and thus be much sparser.

Vector representation

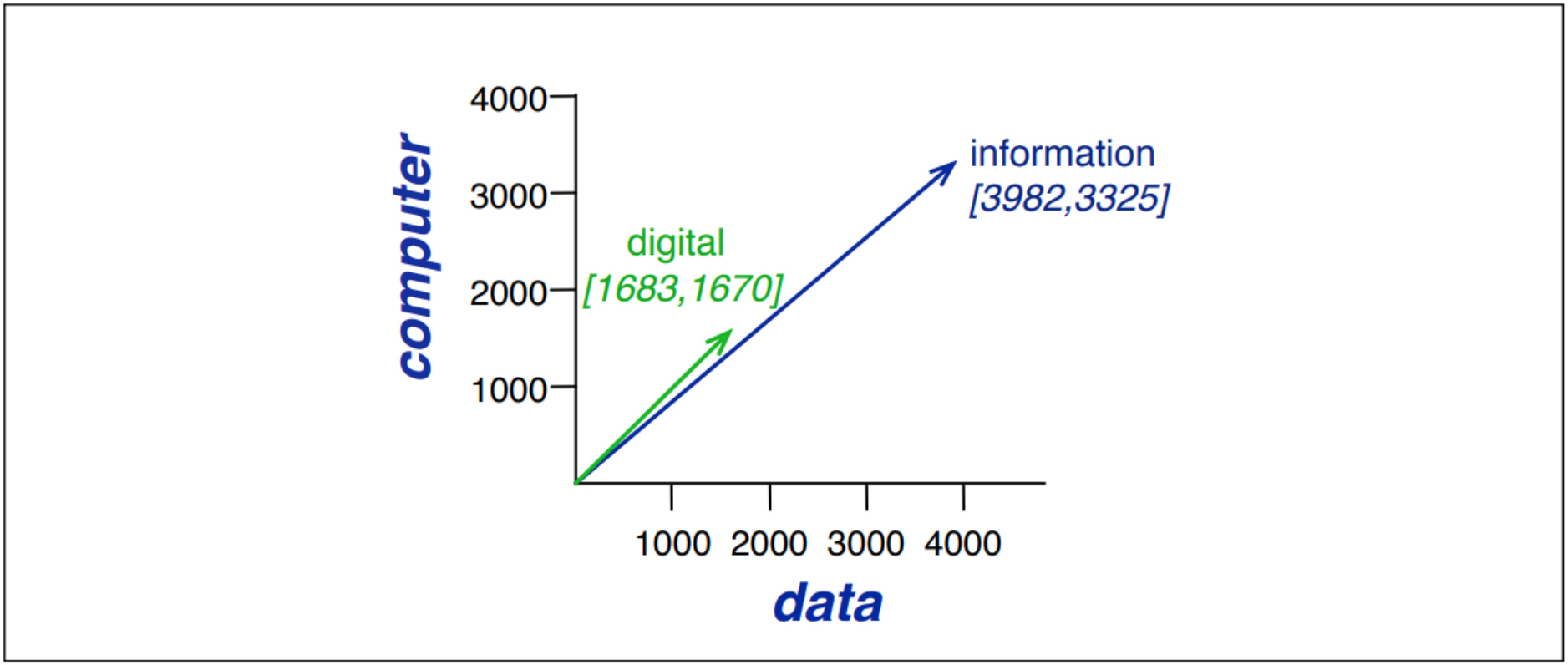


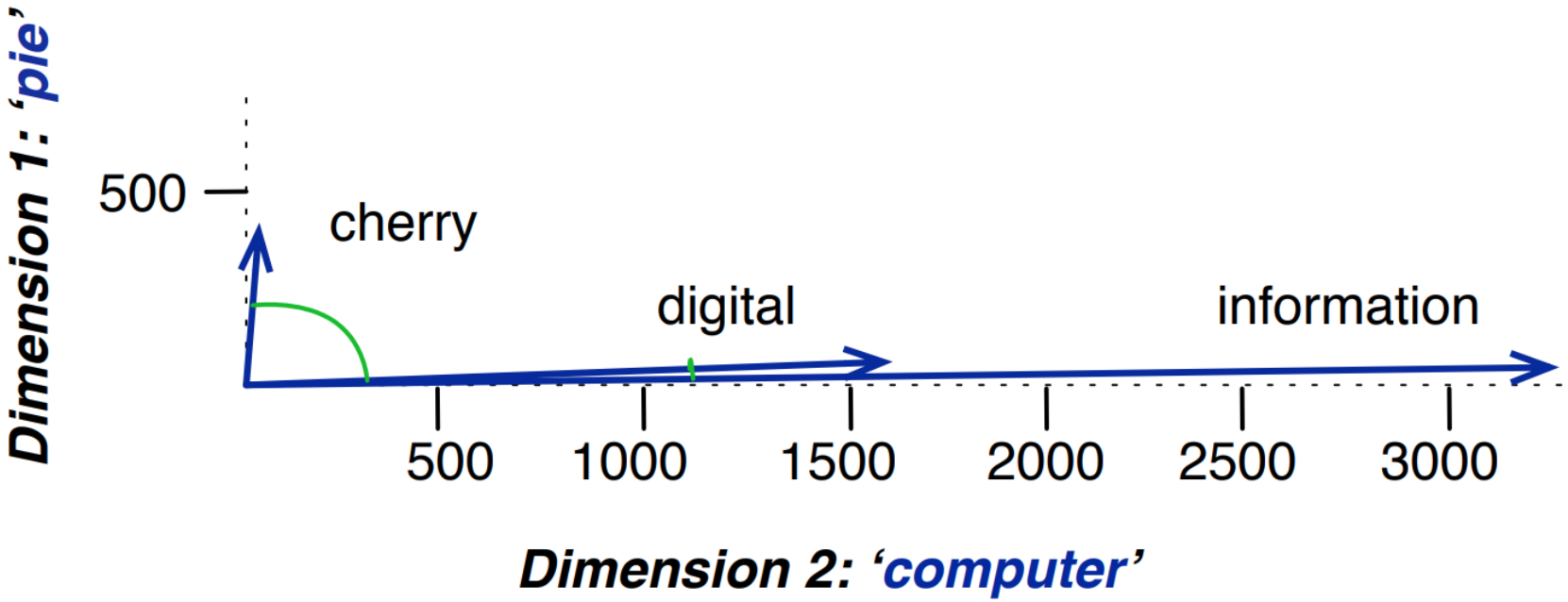
Figure 6.6 A spatial visualization of word vectors for *digital* and *information*, showing just two of the dimensions, corresponding to the words *data* and *computer*.

Vector semantics

- Similarity and relatedness between words can be measured.
  dot product$(\boldsymbol{v}, \boldsymbol{w}) = \boldsymbol{v} \cdot \boldsymbol{w} = \sum_{i=1}^{N} v_i w_i = v_1 w_1 + v_2 w_2 + \cdots + v_N w_N$
- High if large values in the same dimensions, 0 if orthogonal.
- Problem: Favors long vectors, raw dot product higher for frequent words (recall: vector length $\boldsymbol{v} = \sqrt{\sum_{i=1}^{N} v_i^2}$).
- More frequent words have longer vectors, since they tend to co-occur with more words. Similarity metric should be independent of frequency!

➥ Normalize by the length of both vectors. Normalized dot product is the same as the cosine of the angle between the vectors.

Cosine similarity



$$\text{cosine}(\boldsymbol{v}, \boldsymbol{w}) = \frac{\boldsymbol{v} \cdot \boldsymbol{w}}{|\boldsymbol{v}||\boldsymbol{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2} \sqrt{\sum_{i=1}^{N} w_i^2}}$$

$\{\cos(\boldsymbol{v}, \boldsymbol{w}) \in \mathcal{R} | 0 \leq \cos(\boldsymbol{v}, \boldsymbol{w}) \leq 1\}$ since frequencies are positive.

Other ways to measure similarity: tf-idf, PPMI

- tf-idf weighting (now applied to the word-word matrix.)
- An important concept in NLP is pointwise mutual information (Fano 1963).
- Alternative similarity measure: *Positive Pointwise Mutual Information* (PPMI).

Pointwise Mutual Information

- Measures how often two events occur, compared with what would be expected if they were independent.

$$\text{PMI}(x, y) \equiv \log \frac{p(x, y)}{p(x)p(y)} = \log \frac{p(x|y)}{p(x)} = \log \frac{p(y|x)}{p(y)}$$

- PMI between a target word w and a context word w:

$$\text{PMI}(w, c) = \log \frac{p(w, c)}{p(w)p(c)}$$

- How much more do the words co-occur with each other than we would expect by chance?
- Ranges between $(-\infty, +\infty)$.

Positive Pointwise Mutual Information

- Large negative values (indicating things are co-occuring less often than expected) are unreliable unless we have an extremely large corpus.
- Distinguishing whether two words with individual probabilities 10^{-6} occur together less often than by chance requires the probability of them occurring together to be significantly different than 10^{-12}.
- Positive PMI more commonly used.

$$\text{PPMI}(w, c) = \max\left(\log \frac{p(w, c)}{p(w)p(c)}, 0\right)$$

Sparse vs. dense vectors

- So far: Each word represented with a vector that is long (length $|\mathcal{V}|$, commonly 20,000–50,000) and sparse. Difficult to work with.
- Find an alternative word representation using vectors that are short (of length 50–300) and dense.
- Better to use as inputs for statistical modeling: less computationally intensive, more likely to generalize and to avoid overfitting.
- Capture synonymy better: `car` and `automobile` will be distinct dimensions in a sparse representation (vector product is zero).

Semantics with dense vectors: Intuition

- Suppose this sentence occured in our training set:
  "`I have to make sure when I get home to feed the cat.`"
- We are trying to predict what comes after:
  "`I forgot when I got home to feed the`"
- Suppose we've never seen the word "dog" after the words "feed the".
- Traditional N-gram model will predict cat and won't expect "dog".
- A word embedding model will assign a reasonably high probability to "dog" as well as "cat", merely because they have similar vectors.

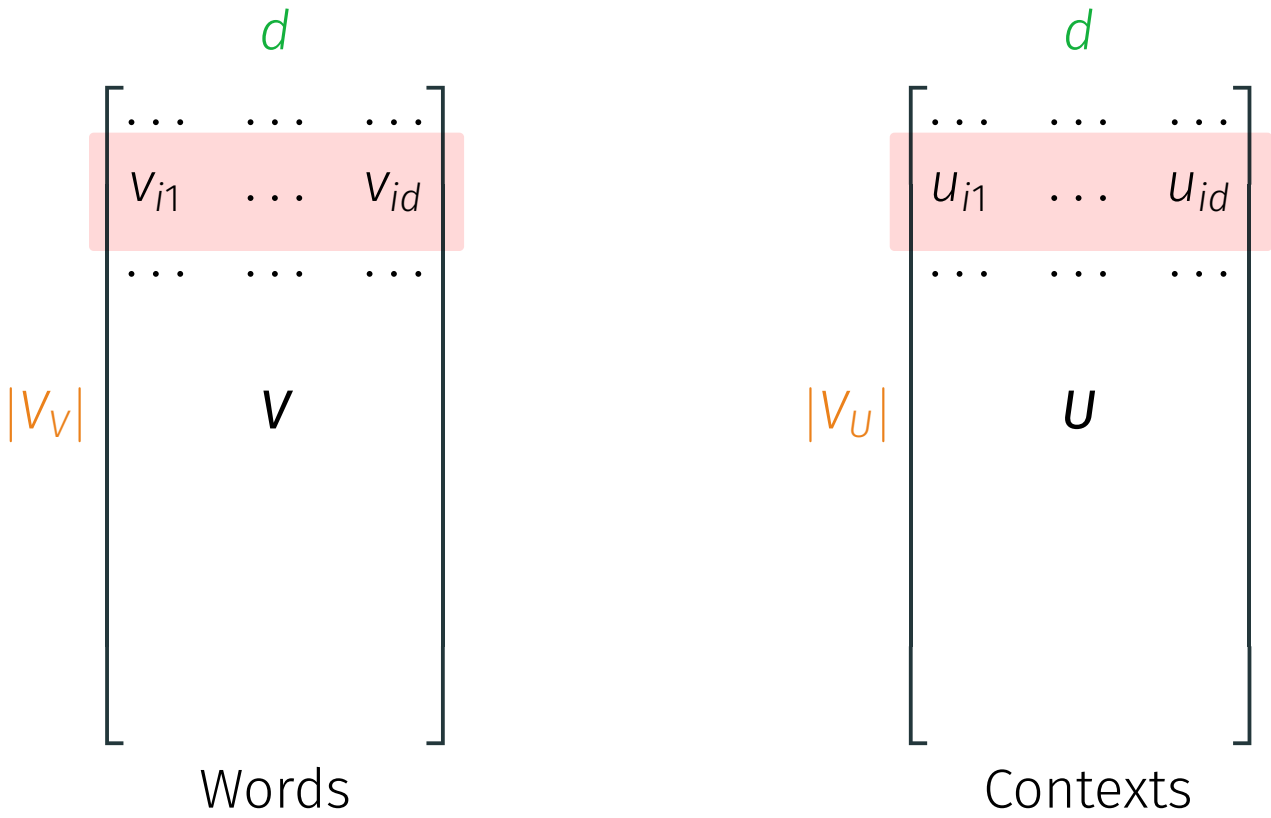
Methods to learn dense embeddings

- Dense embeddings are a powerful representation: words that have similar meanings or grammatical properties have similar vectors.
- They also allow for meaningful arithmetic operations: vector(king) - vector(man) + vector(woman) $\approx$ vector(queen).
- Popular methods: SVD, word2vec, GloVe, fasttext. Approaches differ: matrix factorization of count statistics vs. context prediction.

➥ How do we arrive at a shorter, denser representation of words?

Dense embeddings: Notation

- In practice, learn *two* dense representations (easier optimization).
- Typically this will be a *word* and a *context* representation.
- Represent each word/context as a d-dimensional vector.
- Arrange vectors in two matrices, U and V.



Information retrieval

Information retrieval

Principal components analysis

Principal components analysis

- PCA uses an orthogonal transformation to convert a set of possibly correlated variables into a set of linearly uncorrelated variables called principal components.
- PCA finds a lower-dimensional representation of the data that captures as much as possible of its variation.
- Data may have a high number of dimensions, but not all are equally interesting, i.e. exhibit a lot of variation.
- PCA provides a reduced set of dimensions along which the observations vary the most.
- Scale dependent, so variables should be standardized.
- Developed by Pearson (1901) and Hotelling (1933).

Principal components analysis

- Suppose we have a set of p features $x_1, x_2, \ldots, x_p$. The *first principal component* is the normalized linear combination of features that has the largest variance.

$$z_1 = \phi_{11} x_1 + \phi_{21} x_2 + \cdots + \phi_{p1} x_p$$

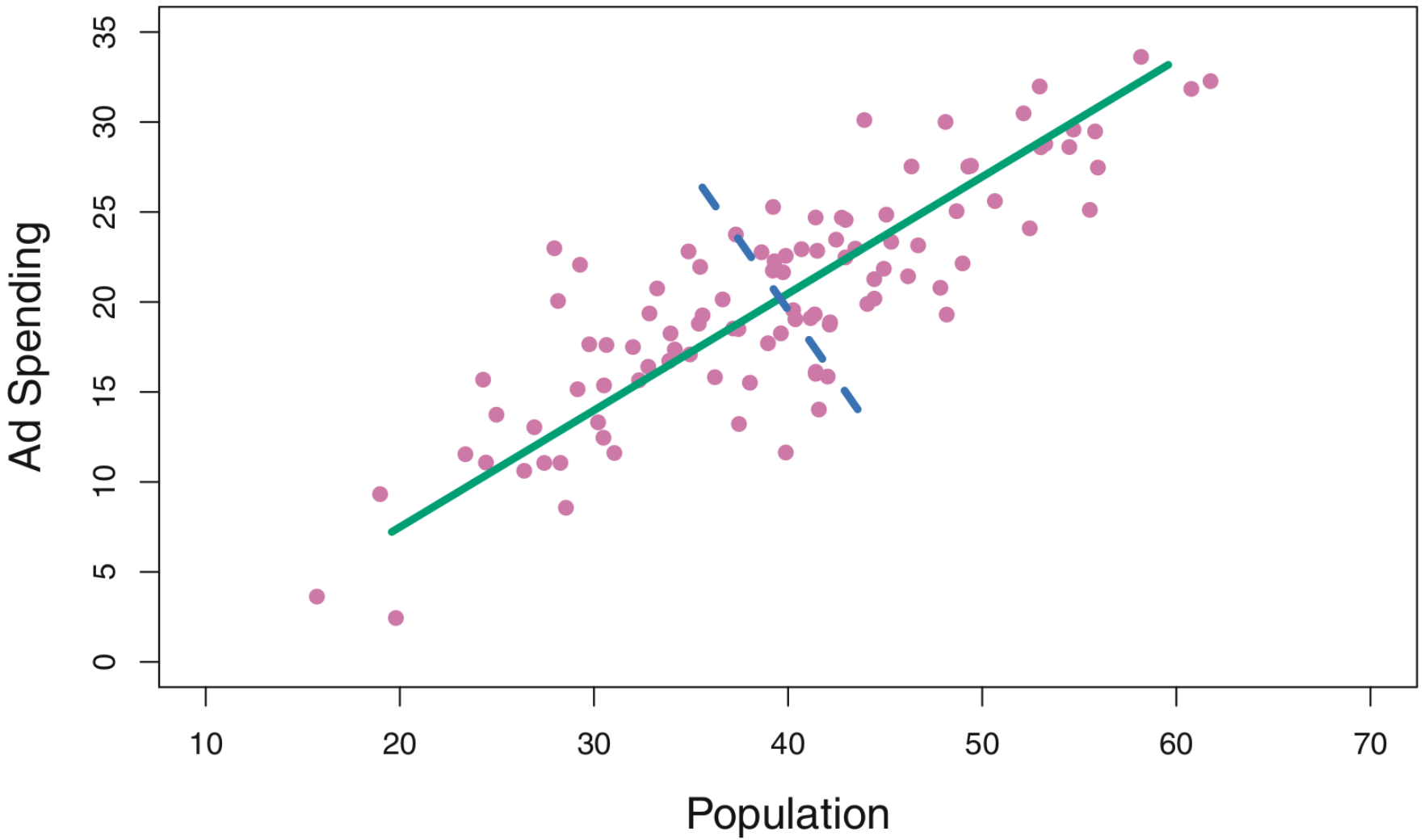
- Normalized: $\sum_{j=1}^{p} \phi_{j1}^2 = 1$.
- Elements $\phi_{11}, \ldots, \phi_{p1}$ are the *loadings* of the first principal component, the principal component direction or loading vector is $\boldsymbol{\phi_1} = (\phi_{11} \ldots \phi_{p1})^T$.

Principal components analysis

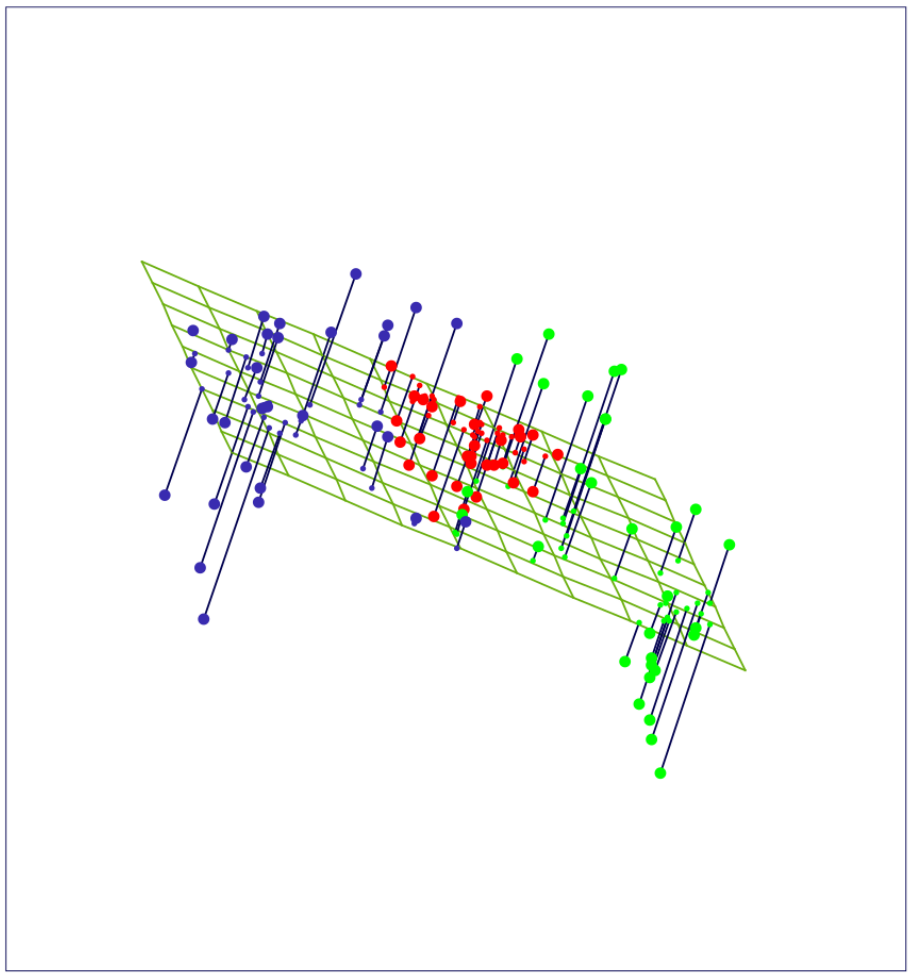
$$\max_{\phi_{11}\ldots\phi_{p1}} \left\{ \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{p}\phi_{j1}x_{ij}\right)^2 \right\} \text{ s.t. } \sum_{j=1}^{p}\phi_{j1}^2 = 1$$

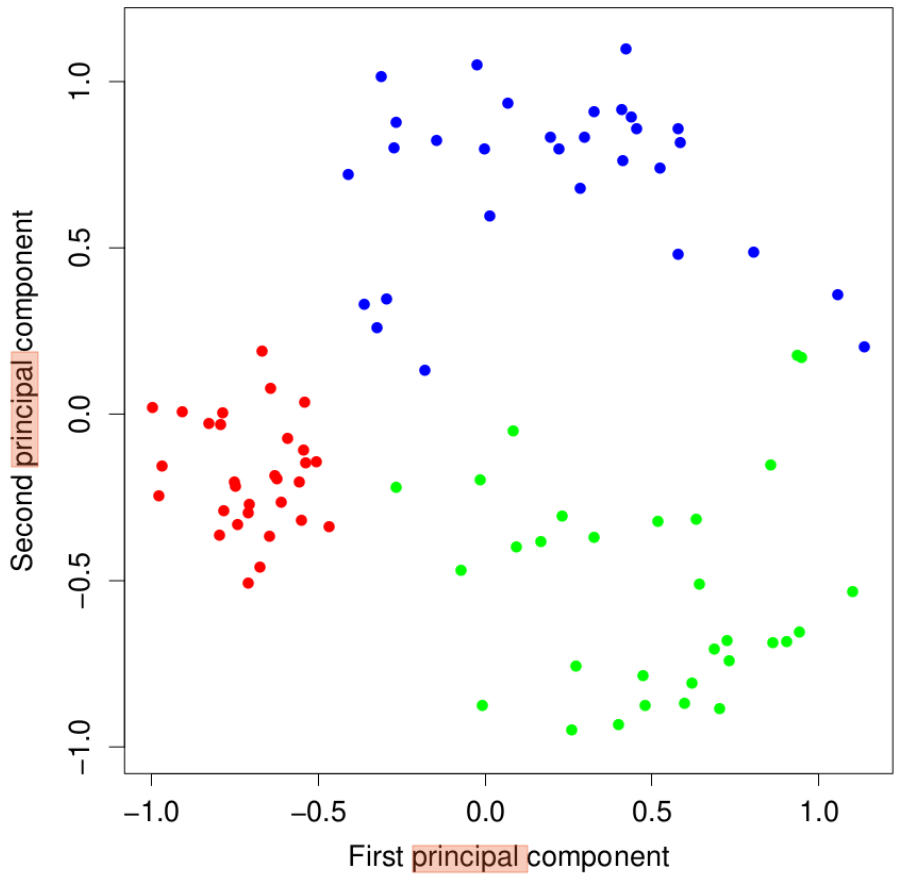
- Further principal components found in the same way, with the additional constraint that the next principal component should be uncorrelated to the previous ones (ϕ_2 orthogonal to ϕ_1).
- Second principal component z_2 is the linear combination of $x_1, \ldots, x_p$ that has maximal variance out of all linear combinations that are uncorrelated with z_1.

PCA: Graphical example



PCA: Graphical example





Principal Components Analysis

- End up with the derived input columns $z_m = Xu_m$ for $m = 1, \ldots, d$, where d is the number of principal components.
- Stacking for all possible principal components: $Z_{n \times d} = X_{n \times p} U_{p \times d}$, with $d_{max} = \min(n - 1, p)$.
- Substitute X with a word-word matrix M, find the first d principal components to get a reduced representation.
- Applying PCS (truncated SVD) to the term-document matrix is also called Latent Semantic Analysis (Deerwester et al. 1990).
- Principal components are typically obtained via singular value decomposition:
  $Z = MU \Leftrightarrow V\Sigma = MU \Leftrightarrow M = V\Sigma U^T$

Principal Components Analysis

- End up with the derived input columns $z_m = Xu_m$ for $m = 1, \ldots, d$, where d is the number of principal components.
- Stacking for all possible principal components: $Z_{n \times d} = X_{n \times p} U_{p \times d}$, with $d_{max} = \mathbf{min}(n - 1, p)$.
- Substitute X with a word-word matrix M, find the first d principal components to get a reduced representation.
- Applying PCS (truncated SVD) to the term-document matrix is also called Latent Semantic Analysis (Deerwester et al. 1990).
- Principal components are typically obtained via singular value decomposition:
  $Z = MU \Leftrightarrow V\Sigma = MU \Leftrightarrow M = V\Sigma U^T$
  $Z_{n \times d} = M_{n \times p} U_{p \times d} \Leftrightarrow V_{n \times d} \Sigma_{d \times d} = M_{n \times p} U_{p \times d} \Leftrightarrow M_{n \times p} = V_{n \times d} \Sigma_{d \times d} U^T_{d \times p}$

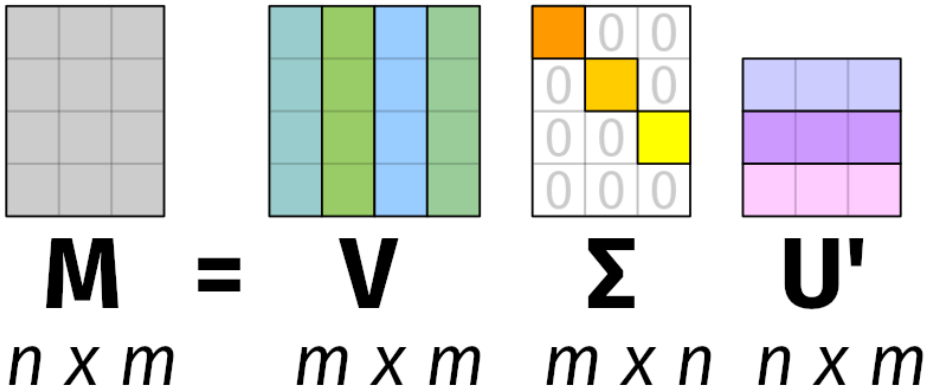
Singular value decomposition

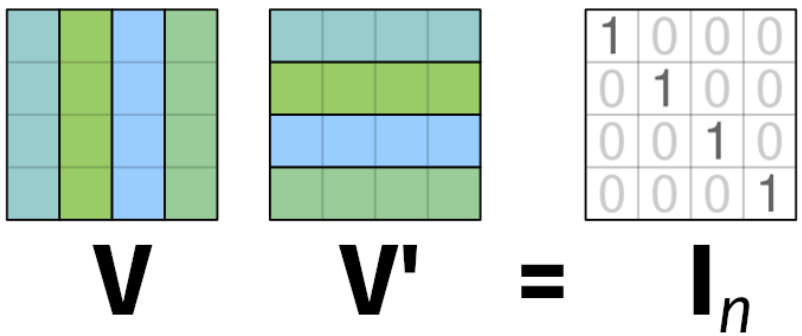
$$M_{n\times m} = V_{n\times n}\Sigma_{n\times m}U^T_{m\times m}$$

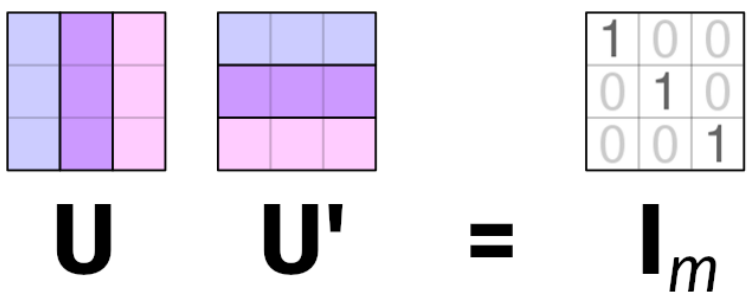
- Any rectangular matrix can be expressed as the product of 3 matrices.
- The columns of V and the rows of U^T are the left-/right-singular vectors of M. Column/row vectors of V/U^T are orthogonal.
- Σ is a diagonal matrix. Diagonal entries σ_{ii} of Σ are the singular values of M. Number of non-zero singular values equals the rank of M.
- SVD is not unique. Choose the decomposition s.t. the singular values σ_{ii} are in descending order (then, Σ is uniquely determined by M).
- $MU = V\Sigma$: Columns of $V\Sigma$ are the principal components of M.

Singular value decomposition

$$M_{n\times m} = V_{n\times n}\Sigma_{n\times m}U^T_{m\times m}$$







Singular value decomposition

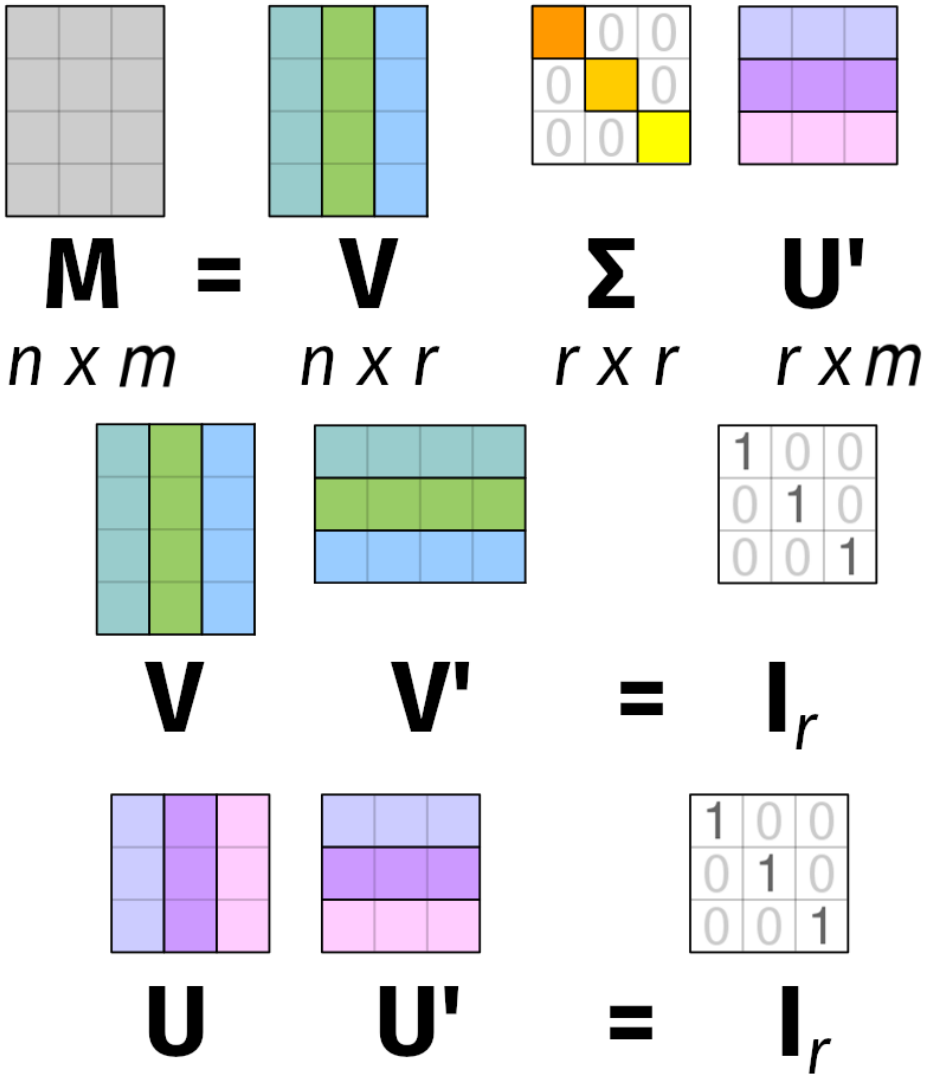
$$M_{n\times m} = V_{n\times n}\Sigma_{n\times m}U^T_{m\times m}$$

- V: Rows correspond to original but n columns represents a dimension in a new latent space (s.t. column vectors are orthogonal to each other and columns are ordered by the amount of variance in the dataset each new dimension accounts for).
- Σ: Diagonal $n \times m$ matrix of singular values expressing the importance of each dimension.
- U^T: Columns correspond to original, but m rows correspond to singular values (as with V).
- Less general: Eigendecomposition $M_{n\times n} = V_{n\times n}\Sigma_{n\times n}V^T_{n\times n}$ if M is a square matrix and has full rank.

Compact SVD

$$M_{n\times m} = V_{n\times r}\Sigma_{r\times r}U^T_{r\times m}$$

where $r = rank(M) \leq \min(n, m)$.
(Dispense with the σ_i which are zero.)



SVD of the word-word matrix

$$M_{|V|\times|V|} = V_{|V|\times|V|}\Sigma_{|V|\times|V|}U^T_{|V|\times|V|}$$

where M is a word-word matrix and assuming $\mathbf{rank}(M) = |V|$.

$$\underset{|V|\times|V|}{\begin{bmatrix} \\ M \\ \\ \\ \end{bmatrix}} = \underset{|V|\times|V|}{\begin{bmatrix} \\ V \\ \\ \\ \end{bmatrix}} \underset{|V|\times|V|}{\begin{bmatrix} \sigma_1 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & \dots & 0 \\ 0 & 0 & \sigma_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 0 & \sigma_{|V|} \end{bmatrix}} \underset{|V|\times|V|}{\begin{bmatrix} \\ U^T \\ \\ \\ \end{bmatrix}}$$

Truncated SVD of the word-word matrix

- Keep only the first d latent dimensions.
- These are the dimensions that absorb most of the variance between words.

$$\underset{|V| \times |V|}{\begin{bmatrix} \\ M \\ \\ \\ \end{bmatrix}} = \underset{|V| \times |V|}{\left[\begin{array}{c|c} & \\ V & \\ & \end{array}\right]} \left[\begin{array}{ccc|cc} \sigma_1 & 0 & \dots & \dots & 0 \\ 0 & \ddots & \dots & \dots & 0 \\ \vdots & \vdots & \sigma_d & \dots & 0 \\ \hline \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 0 & \sigma_{|V|} \end{array}\right]_{|V| \times |V|} \underset{|V| \times |V|}{\left[\begin{array}{c} \\ U^T \\ \\ \hline \\ \end{array}\right]}$$

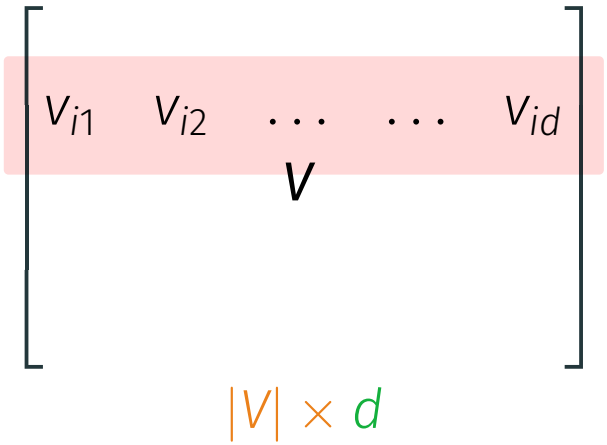
Truncated SVD of the word-word matrix

- Decomposition no longer exact (but useful). Faster to compute.
- Approximate matrix $\tilde{M}$ is the closest approximation to M that can be achieved by a matrix of rank d.

$$\underbrace{\begin{bmatrix} \\ \tilde{M} \\ \\ \end{bmatrix}}_{|V| \times |V|} = \underbrace{\begin{bmatrix} \\ V \\ \\ \end{bmatrix}}_{|V| \times d} \underbrace{\begin{bmatrix} \sigma_1 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & \dots & 0 \\ 0 & 0 & \sigma_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 0 & \sigma_d \end{bmatrix}}_{d \times d} \underbrace{\begin{bmatrix} \\ U^T \\ \\ \end{bmatrix}}_{d \times |V|}$$

Truncated SVD of the word-word matrix

Embedding for word i



- Each row of V is a d-dimensional representation of each word w.
- d may typically range from 50 to 300, rarely up to 1000.
- Generally keep only the top dimensions, but some studies suggest that getting rid of some of the top dimensions may be helpful.
- SVD is expensive and does not scale well with large corpuses, although less relevant today with sparse matrix representations.
- PCA can be difficult to tune.

Vector semantics

- Interpretation of semantic relations non-intuitive with PCA.
- Embeddings do not capture semantic relations.
- PCA does dimension reduction and captures variation, but does not encode meaning well.
- Often better to apply to document-term matrices for dimension reduction rather than word-word matrices.
- Alternative models better suited to capture meaning (GloVe, word2vec).

Distributional language models

Distributional language models

word2vec

Vector semantics

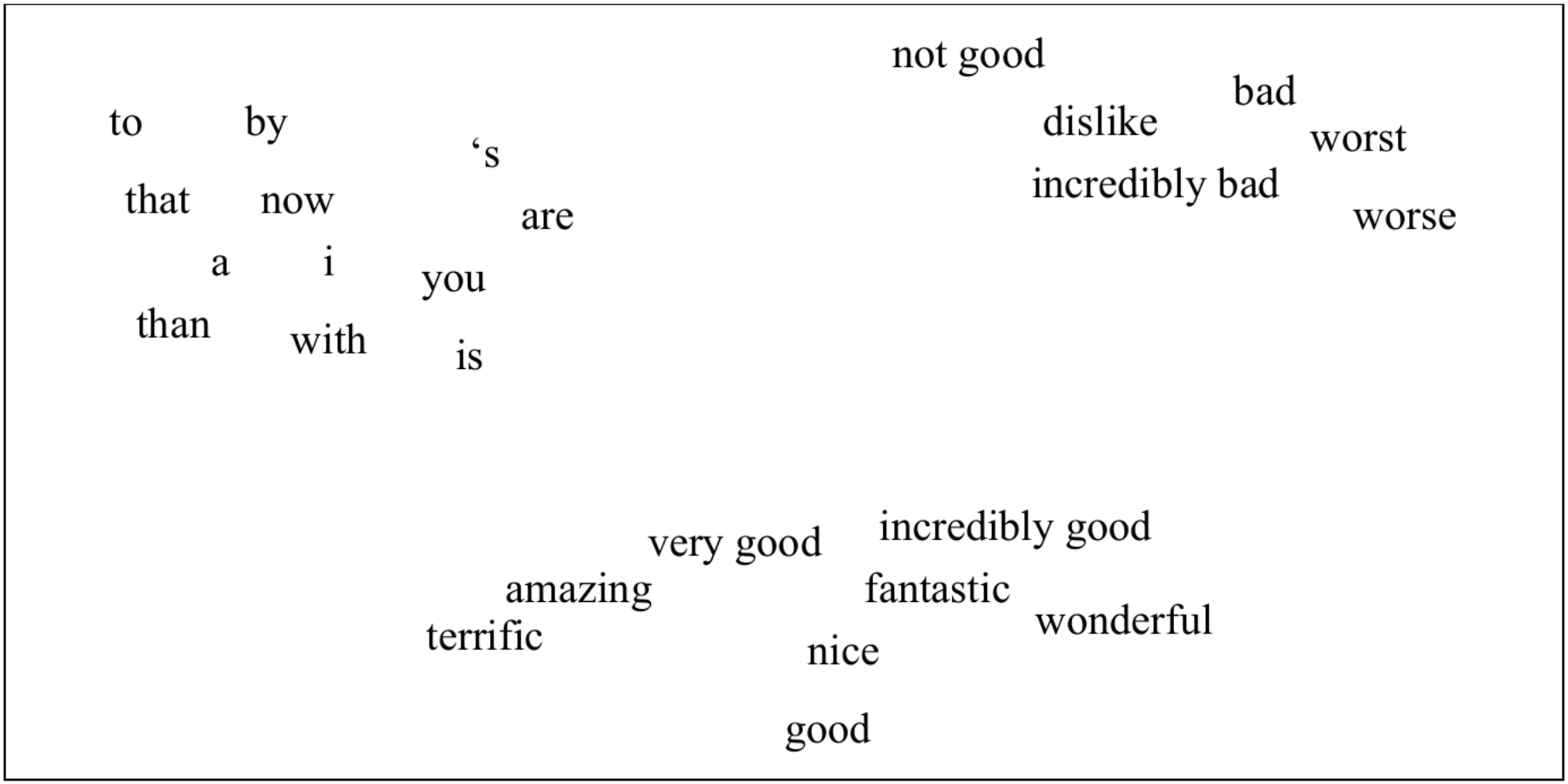


Figure 6.1 A two-dimensional (t-SNE) projection of embeddings for some words and phrases, showing that words with similar meanings are nearby in space. The original 60-dimensional embeddings were trained for a sentiment analysis task. Simplified from Li et al. (2015).

Vector semantics: Relational properties

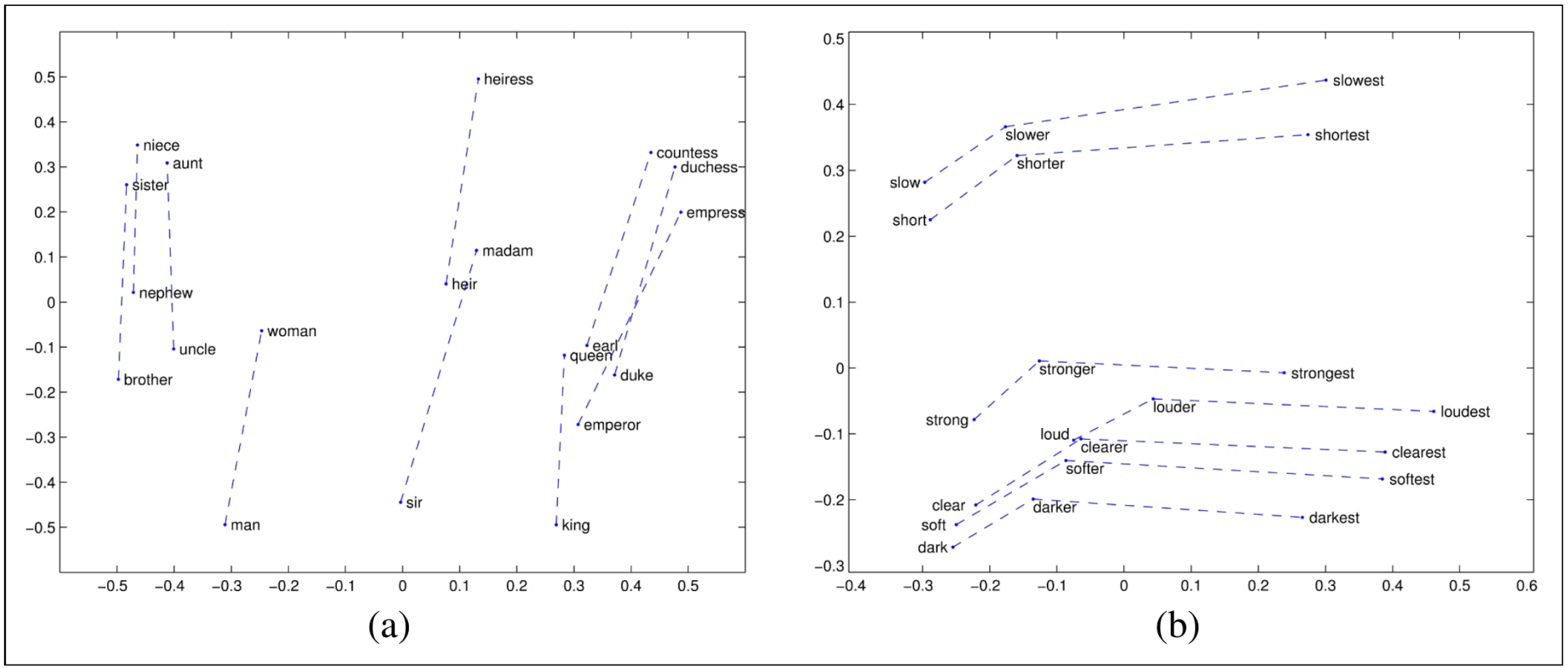


Figure 6.14 Relational properties of the vector space, shown by projecting vectors onto two dimensions. (a) 'king' - 'man' + 'woman' is close to 'queen' (b) offsets seem to capture comparative and superlative morphology (Pennington et al., 2014).

Vector semantics: Meaning over time

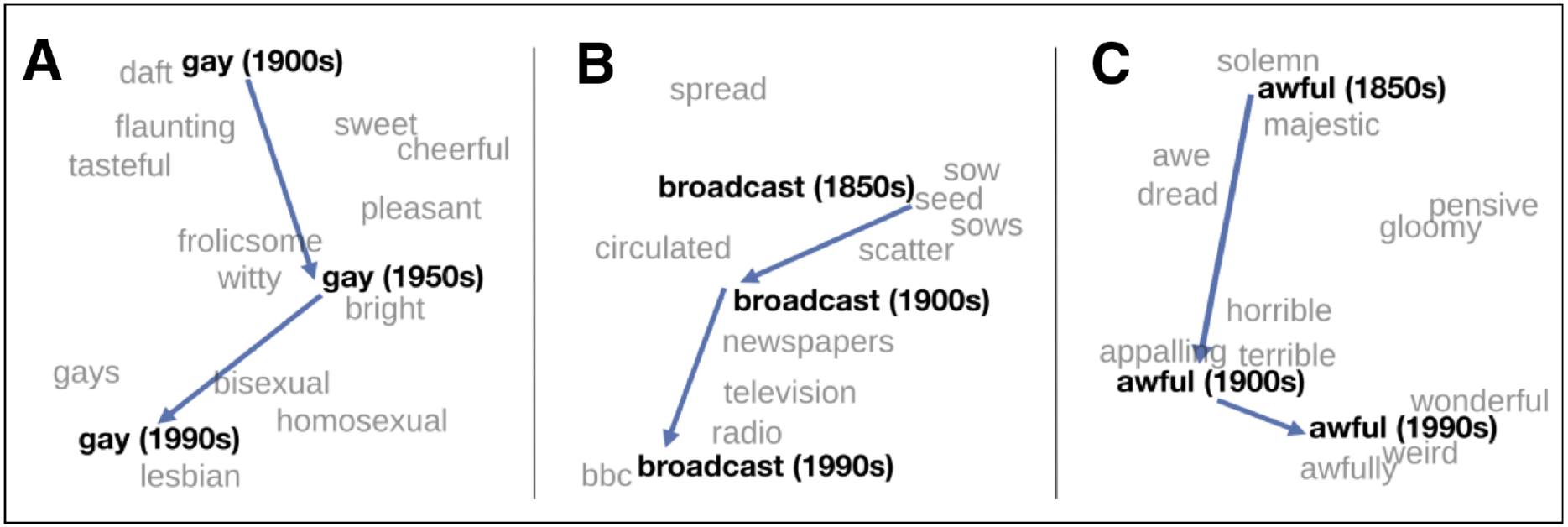


Figure 6.15 A t-SNE visualization of the semantic change of 3 words in English using word2vec vectors. The modern sense of each word, and the grey context words, are computed from the most recent (modern) time-point embedding space. Earlier points are computed from earlier historical embedding spaces. The visualizations show the changes in the word *gay* from meanings related to "cheerful" or "frolicsome" to referring to homosexuality, the development of the modern "transmission" sense of *broadcast* from its original sense of sowing seeds, and the pejoration of the word *awful* as it shifted from meaning "full of awe" to meaning "terrible or appalling" (Hamilton et al., 2016b).

word2vec

- word2vec: Mikolov et al. (2013b). *Efficient Estimation of Word Representations in Vector Space*.
- One of the most popular embedding models.
- Fast:

  > *"[...] an optimized single-machine implementation can train on more than 100 billion words in one day."*
  >
  > *Mikolov et al. 2013a*

- Basic idea: word2vec learns the embeddings by predicting word contexts.
- Unsupervised, but the contexts can be considered *implicitly labeled* training data.

word2vec

- Exploit the fact that we can observe whether words occur in the vicinity of others.
- *Predict* rather than *count* co-occurence.
- Build a dense vector for each word s.t. it is good at predicting other words in its context (which are also represented by vectors).
- Trains word embeddings by optimizing a loss function with gradient descent. Loss function measures how words predict their surrounding.

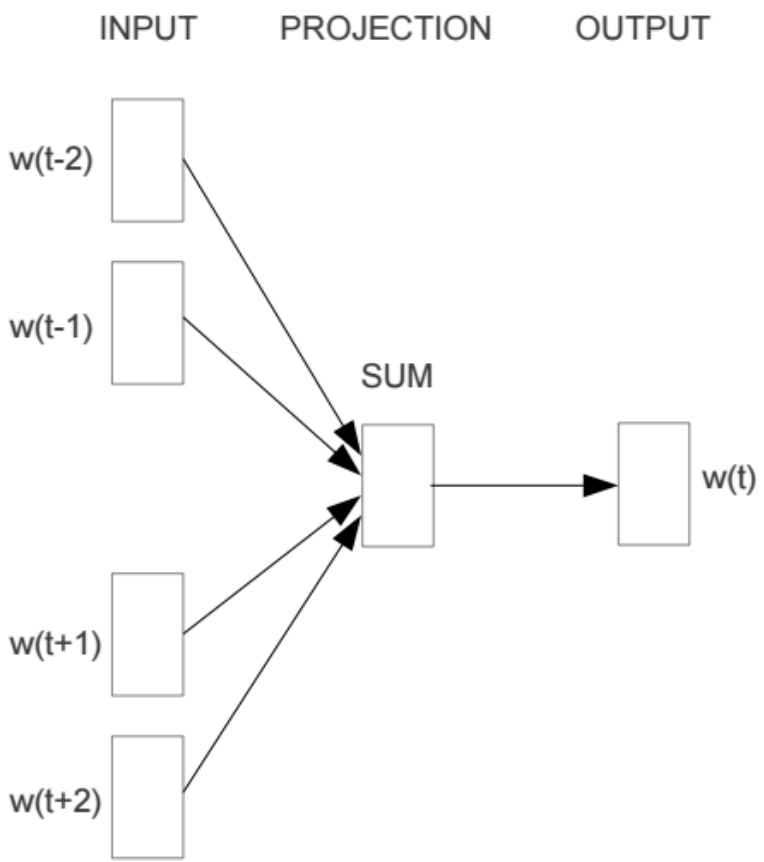
word2vec

1. We have a large corpus of text.
2. Every word in a fixed vocabulary is represented by a vector.
3. Go through each position in the text, which has a target (center) word w and context (outside) words c.
4. Use the similarity of the word vectors for w and c to calculate the probability of c given w (or vice versa).
5. Keep adjusting the word vectors to maximize this probability.

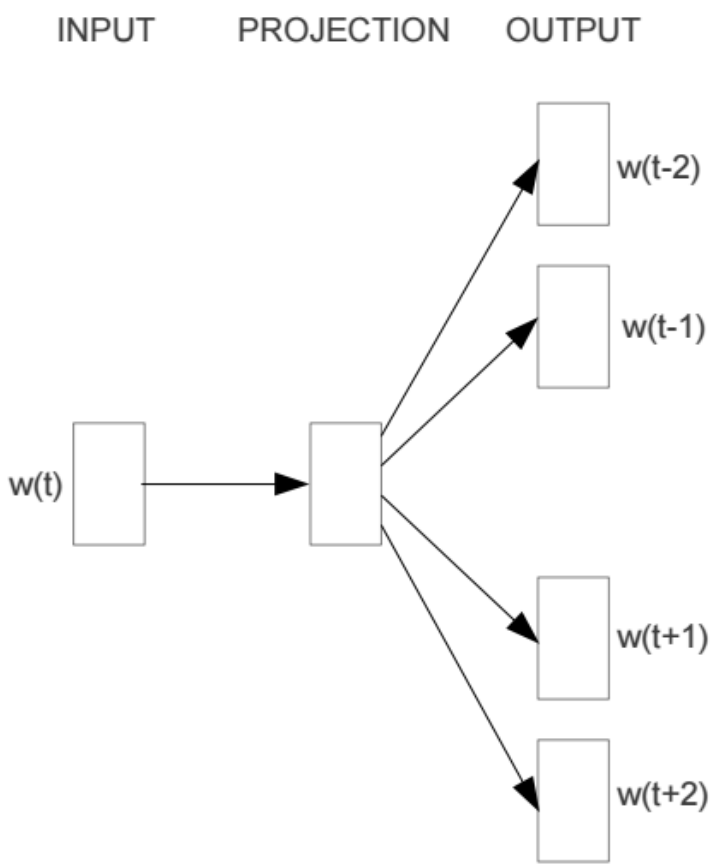
word2vec

- *Predict* rather than *count* co-occurence. Two approaches:
  - Predict the word in focus from context words (continuous-bag-of-words).
  - Predict context words using the word in focus (skip-gram model).
- Different training methods:
  - Naive softmax.
  - Negative sampling.
  - Hierarchical softmax.

CBOW/Skip-grams



CBOW



Skip-gram

word2vec: Skip-grams

Consider a +/- 2 word context window. Training sentence:

...lemon, a tablespoon of apricot jam, a pinch of ...

word2vec: Skip-grams

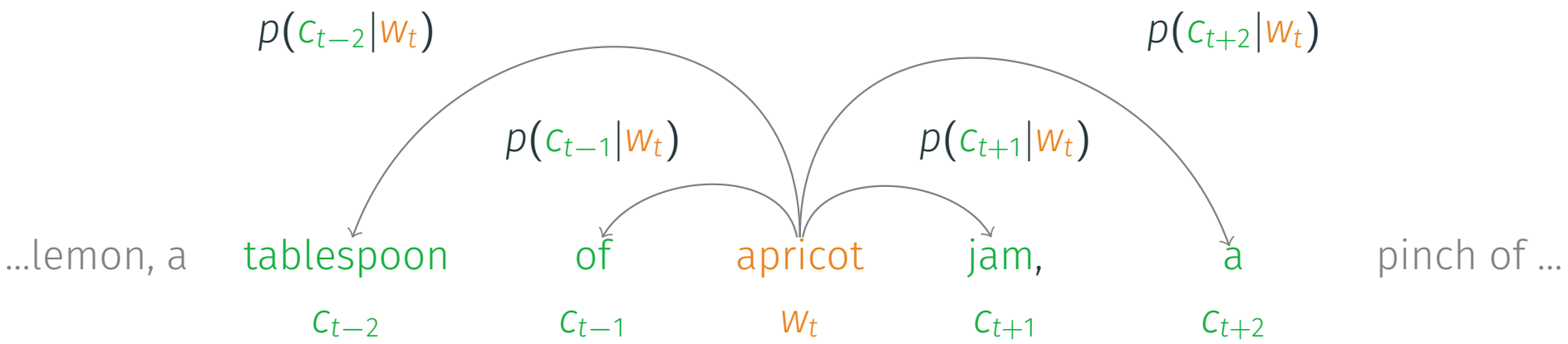
Consider a +/- 2 word context window. Training sentence:

...lemon, a tablespoon of apricot jam, a pinch of ...

c_{t-2} c_{t-1} w_t c_{t+1} c_{t+2}

word2vec: Skip-grams

Consider a +/- 2 word context window. Training sentence:



word2vec: Skip-grams

- Stack word/context embeddings in a single parameter vector.

$$\boldsymbol{\theta} = \begin{bmatrix} \dots & \dots & \dots \\ v_{w1} & \dots & v_{wd} \\ \dots & \dots & \dots \\ \\ \dots & \dots & \dots \\ u_{c1} & \dots & u_{cd} \\ \dots & \dots & \dots \\ \\ \end{bmatrix} \in \mathcal{R}^{d2V}$$

- Predict the probability that a word c_{t+j} is a real context (outside) word for target (center) word w_t.

$$p(c_{t+j}|w_t, \boldsymbol{\theta})$$

word2vec: Loss function

- For each position in the text $t = 1, \ldots, T$, predict context words within a window of fixed size m, given center word w_t. Likelihood:

$$L(\boldsymbol{\theta}) = \prod_{t=1}^{T} \prod_{\substack{-m \leq j \leq m \\ j \neq 0}} p(c_{t+j} | w_t, \boldsymbol{\theta})$$

- Minimize negative log-likelihood:

$$J(\boldsymbol{\theta}) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \log p(c_{t+j} | w_t, \boldsymbol{\theta})$$

- Alternatively, just sum over all `(word, context)` tuples and minimize

$$J(\boldsymbol{\theta}) = \sum_{(w,c) \in D} \log p(c | w, \boldsymbol{\theta})$$

word2vec: Loss function

$$J(\boldsymbol{\theta}) = \sum_{(w,c)\in D} \log p(c|w,\boldsymbol{\theta})$$

- How do we calculate $p(c|w,\boldsymbol{\theta})$?
- Rely on the word embedding vectors, i.e. rows of $\boldsymbol{V}$ and $\boldsymbol{U}^T$.
  Row v_t contains embeddings when word t is a center/target word.
  Row u_t contains embeddings when word t is an outside/context word.
- Recall: Two vectors are similar if their dot product is large.
- Apply the softmax (normalized exponential) function to map dot product into a probability ($\sigma : \mathcal{R}^n \mapsto (0,1)^n$).

word2vec: Loss function

- For any target word w and context word c:

$$p(c|w) = \frac{\exp u_c^T v_w}{\sum_{c' \in C} \exp(u_{c'}^T v_w)}$$

- Dot product measures similarity.
- Exponentiation makes it positive.
- Normalization over entire vocabulary gives a probability.
- Softmax is a generalized logistic/sigmoid function (as in multinomial logistic regression).
- Simplifying assumption: assume context words are independent.

word2vec: Loss function

$$\arg\max_{\boldsymbol{\theta}} \sum_{(w,c)\in D} \left(\log \exp(u_c^T v_w) - \log \sum_{c'\in C} \exp(u_{c'}^T v_w) \right)$$

- Minimize cost function $J(\boldsymbol{\theta})$ using gradient descent.
- Derive the gradient $\nabla_{\boldsymbol{\theta}} J(\theta) = \begin{bmatrix} \nabla_{v_w} J(\theta) \\ \nabla_{u_c} J(\theta) \end{bmatrix}$.

word2vec: Gradient descent

More

- Basic idea: for current value of $\boldsymbol{\theta}$, calculate gradient of $J(\boldsymbol{\theta})$, then update $\boldsymbol{\theta}$ by a small step in the direction of the negative gradient. Repeat.
- Update equation: $\boldsymbol{\theta}^{new} = \boldsymbol{\theta}^{old} - \alpha \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$.
- Problem: $\nabla_{\boldsymbol{\theta}} J(\theta)$ is expensive to compute (function of all windows in the corpus). Inner summation is costly, dimension of all possible contexts can be large.
- Need to make computation more tractable.
- Alternatives: Negative sampling or hierarchical softmax.

word2vec: Skip-grams with negative sampling

Algorithm:

1. Treat the target word and a neighboring context word as positive examples.
2. Randomly sample other words in the lexicon to get negative samples.
3. Use logistic regression to train a classifier to distinguish those two cases.
4. Use the weights as the embeddings.

word2vec: Skip-grams with negative sampling

- Consider a +/- 2 word context window. Training sentence:

  ...lemon, a tablespoon of apricot jam, a pinch of ...

  w_{t-2} w_{t-1} w_t w_{t+1}, w_{t+2}

- Given a (word, context) tuple (w, c) , e.g.

  (apricot, jam)
  (apricot, aardvark),

  find the probability that c is a real context word.

- Probability that (w, c) is/is not from the corpus data:

$$p(D = 1|w, c)$$
$$p(D = 0|w, c) = 1 - p(D = 1|w, c)$$

- Again, parameters $\boldsymbol{\theta}$ control the distribution: $p(D = 1|w, c, \boldsymbol{\theta})$.

word2vec: Skip-grams with negative sampling

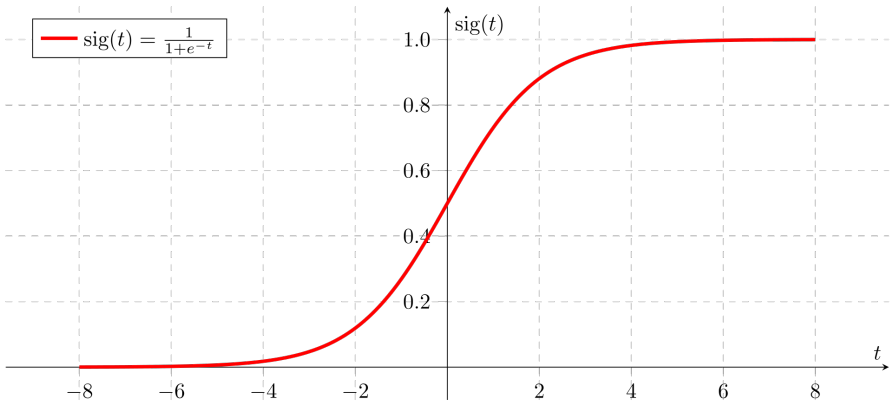
- Find parameters to maximize the probabilities that all observations came from the data:

$$\arg\max_{\boldsymbol{\theta}} \prod_{(w,c)\in D} p(D=1|w,c,\boldsymbol{\theta})$$
$$= \arg\max_{\boldsymbol{\theta}} \sum_{(w,c)\in D} \log p(D=1|w,c,\boldsymbol{\theta})$$

- Use the sigmoid (logistic) function to map dot product into a probability.

$$p(D=1|w,c,\boldsymbol{\theta}) = \sigma(v_w, u_c^T)$$
$$= \frac{\exp(v_w u_c^T)}{1+\exp(v_w u_c^T)}$$

$$p(D=0|w,c,\boldsymbol{\theta}) = 1-\sigma(v_w, u_c^T)$$
$$= \frac{1}{1+\exp(v_w u_c^T)}$$



word2vec: Skip-grams with negative sampling

- New objective:

$$\arg\max_{\boldsymbol{\theta}} \sum_{(w,c)\in D} \log \frac{\exp(v_w u_c^T)}{1+\exp(v_w u_c^T)}$$

- Trivial solution: Set $\boldsymbol{\theta}$ such that $p(D=1|w,c,\boldsymbol{\theta})=1$ for every pair (w,c).
- Achieved by setting $\boldsymbol{\theta}$ s.t. $v_w = u_c^T$ and $v_w \cdot u_c^T = K$, where K is a large enough number.

➥ Need a mechanism that prevents all the vectors from having the same value by penalizing some (w,c) combinations.

word2vec: Skip-grams with negative sampling

- Negative sampling: generate a set D' of negative example (w, c) pairs.
- In practice, randomly sample context words for negative example and assume they are incorrect.
- The text provides positive examples.
- Negative examples need to be sampled.

positive examples +	
t	c
apricot	tablespoon
apricot	of
apricot	jam
apricot	a

negative examples -			
t	c	t	c
apricot	aardvark	apricot	seven
apricot	my	apricot	forever
apricot	where	apricot	dear
apricot	coaxial	apricot	if

word2vec: Skip-grams with negative sampling

- New, augmented objective:

$$\arg\max_{\boldsymbol{\theta}} \prod_{(w,c)\in D} p(D=1|w,c,\boldsymbol{\theta}) \prod_{(w,c)\in D'} (1-p(D=1|w,c,\boldsymbol{\theta}))$$

$$= \arg\max_{\boldsymbol{\theta}} \sum_{(w,c)\in D} \log \frac{\exp(v_w u_c^T)}{1+\exp(v_w u_c^T)} + \sum_{(w,c)\in D'} \log \frac{1}{1+\exp(v_w u_c^T)}$$

- Given the set of positive/negative training instances, adjust s.t.
  1. maximize similarity of target/context word pairs from the positive examples
  2. minimize similarity of pairs drawn from the negative examples.

word2vec: Skip-grams with negative sampling

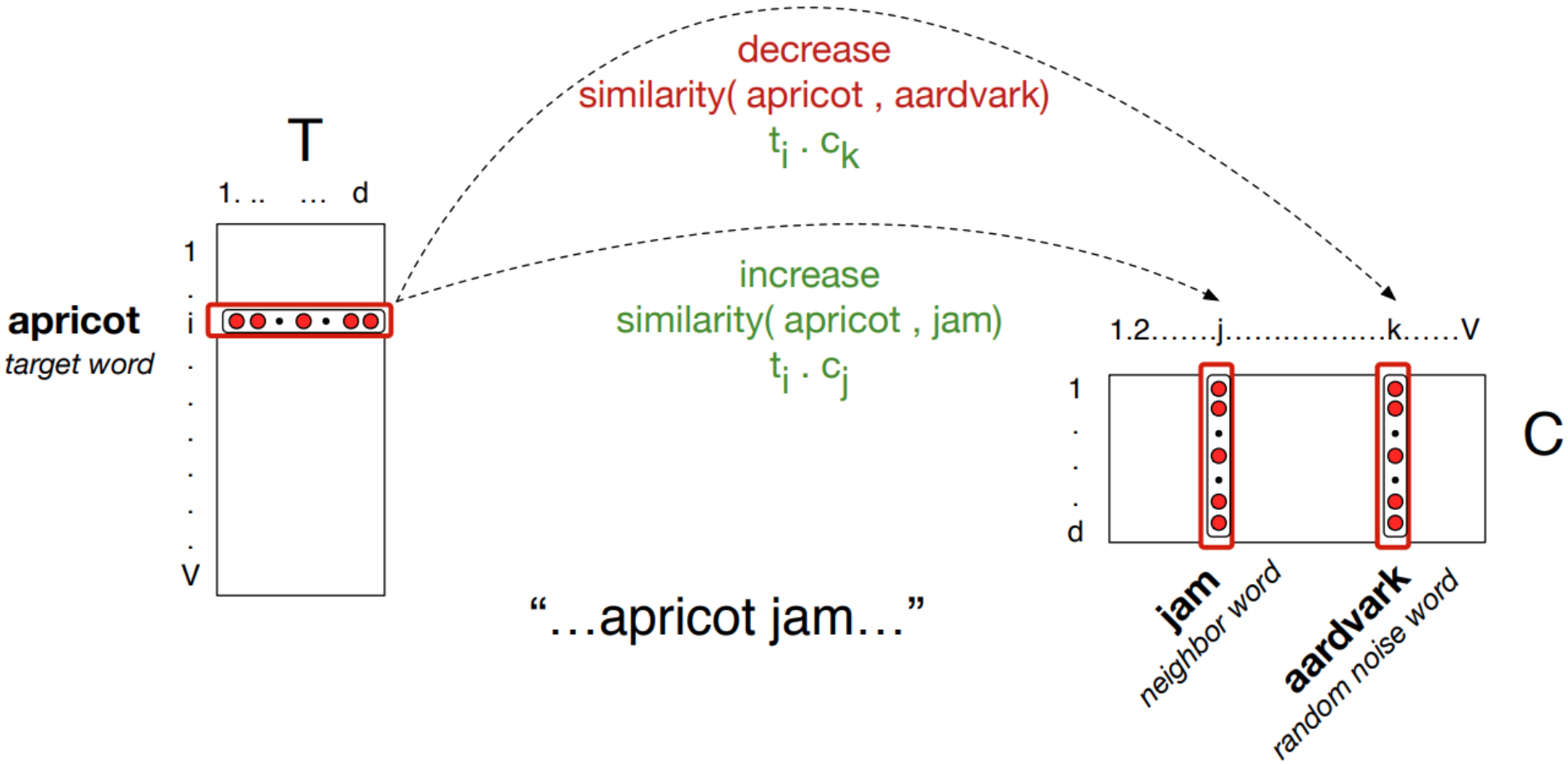
- Sampling ratio negative:positive $k > 1$, tuning parameter.
- Sample negative examples according to their (weighted) unigram frequency.

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

- $\alpha = 0.75$ gives better performance as it gives rare words slightly higher probabilities of being sampled.

word2vec: Skip-grams with negative sampling

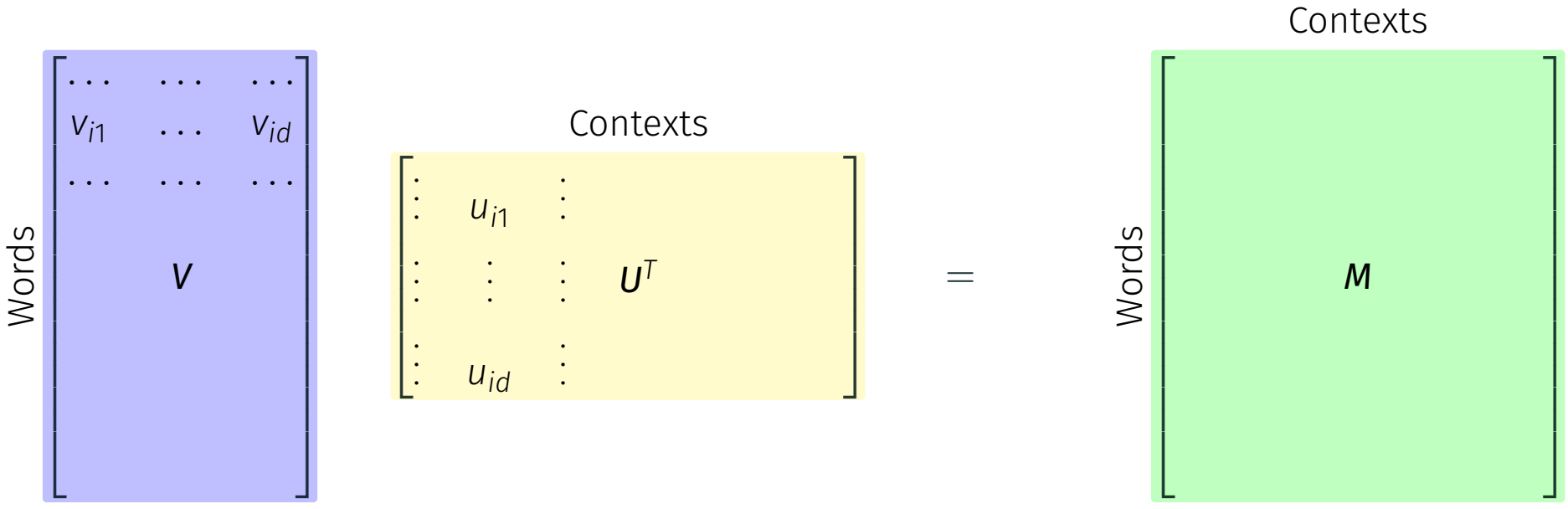
1. Initialize a set of random embeddings *V* and *U*.
2. Learn context and target embeddings by walking through the training corpus using stochastic gradient descent.



SGNS: Remarks

- Skip-gram with negative sampling trains a probabilistic classifier that, given a target word w and its context window of $2j$ words $c_{-j:j}$, assigns a probability based on how similar this context window is to the target word.
- Learn embeddings by iteratively shifting an initial set of random or zero embedding vectors for each word to be more like nearby words, and less like those that are not nearby.
- Probability is based on applying the logistic function to the dot product of the embeddings of the target word with each context word.
- SGNS does not model $p(c|w)$ directly, but instead models a quantity related to the joint distribution of w and c.

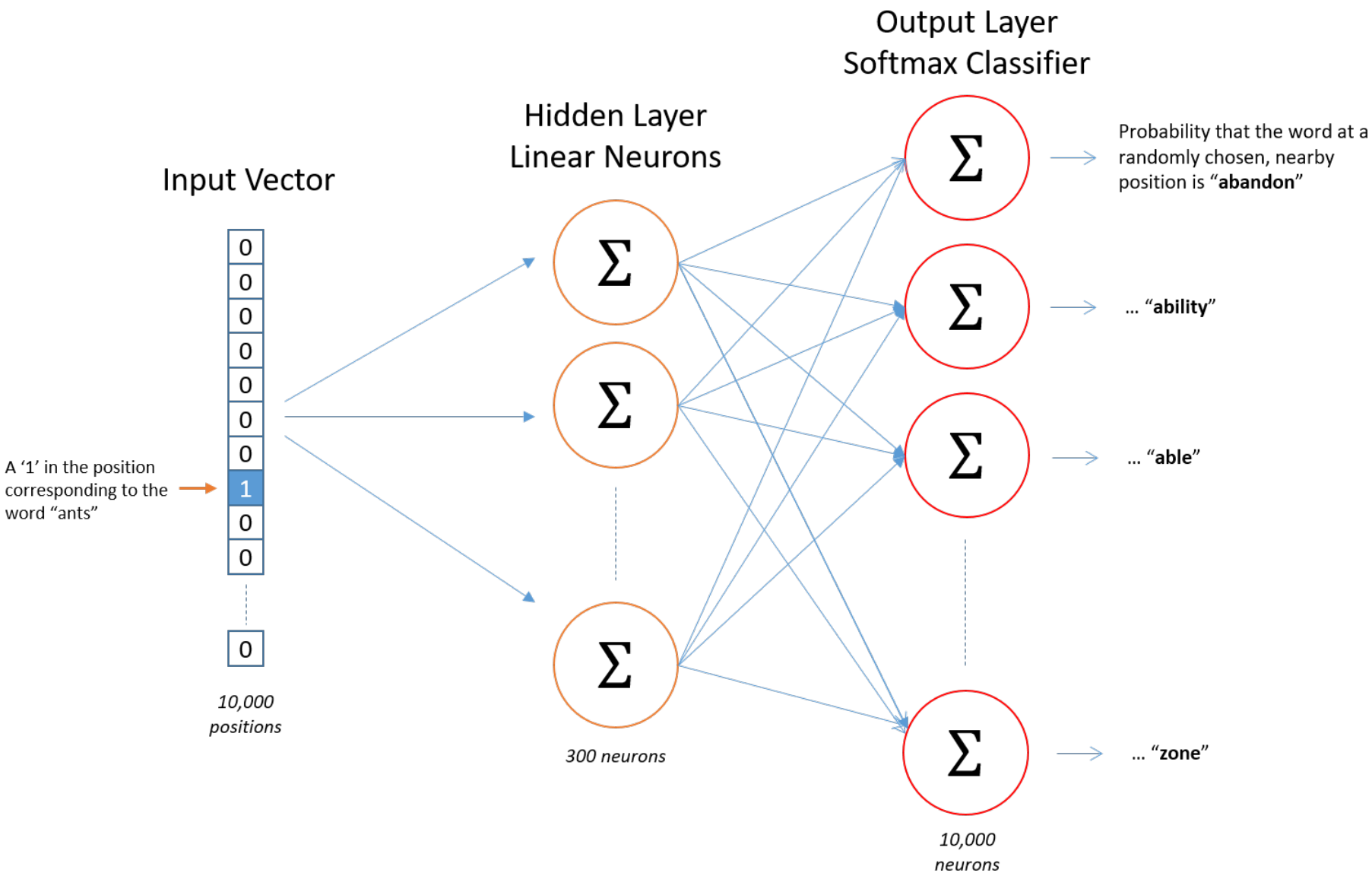
SGNS: Reinterpretation



- Similar to SVD (without using matrix factorization).
- Formal relation: Levy and Goldberg (2014).

word2vec: Neural network weights

Live demo



word2vec

- Embeddings are sensitive to the size of the context window.
- Implementations often use neural nets.
- Not strictly required: SGNS works with logistic regression.
- Dense embeddings work extremely well for many tasks, especially those were word analogies matter (e.g. translation, similarity).
- Not (formally) fully clear why.

Word embeddings

- Embeddings capture synonymy and polysemy better, but interpretability is lost.
- Differences can still be interpreted. Allows investigating e.g. gender stereotypes or measure implicit associations.
- Bolukbasi et al. 2016 look at embeddings from news texts.
- Difference in embedding distance between gender-specific terms and gender-neutral terms as a measure of bias.
- Closest occupation to ‘computer programmer’ - ‘man’ + ‘woman’ is ‘homemaker’. ‘Father’ is to ‘doctor’ as ‘mother’ is to ‘nurse’.

Gender stereotypes

Extreme *she*	**Extreme *he***
1. homemaker	1. maestro
2. nurse	2. skipper
3. receptionist	3. protege
4. librarian	4. philosopher
5. socialite	5. captain
6. hairdresser	6. architect
7. nanny	7. financier
8. bookkeeper	8. warrior
9. stylist	9. broadcaster
10. housekeeper	10. magician

Gender stereotype *she-he* analogies

sewing-carpentry	registered nurse-physician	housewife-shopkeeper
nurse-surgeon	interior designer-architect	softball-baseball
blond-burly	feminism-conservatism	cosmetics-pharmaceuticals
giggle-chuckle	vocalist-guitarist	petite-lanky
sassy-snappy	diva-superstar	charming-affable
volleyball-football	cupcakes-pizzas	lovely-brilliant

Gender appropriate *she-he* analogies

queen-king	sister-brother	mother-father
waitress-waiter	ovarian cancer-prostate cancer	convent-monastery

Figure 1: **Left** The most extreme occupations as projected on to the *she*−*he* gender direction on w2vNEWS. Occupations such as *businesswoman*, where gender is suggested by the orthography, were excluded. **Right** Automatically generated analogies for the pair *she-he* using the procedure described in text. Each automatically generated analogy is evaluated by 10 crowd-workers to whether or not it reflects gender stereotype.

Word embeddings and stereotypes

- Garg et al. (2018), PNAS: *Word embeddings quantify 100 years of gender and ethnic stereotypes*.
- Use embeddings to study temporal dynamics in stereotypes and attitudes toward women and ethnic minorities in the 20th and 21st centuries in the United States.
- Captures societal shifts (the women's movement in the 1960s and Asian immigration) and how specific adjectives and occupations became more closely associated with populations over time.

Word embeddings and stereotypes

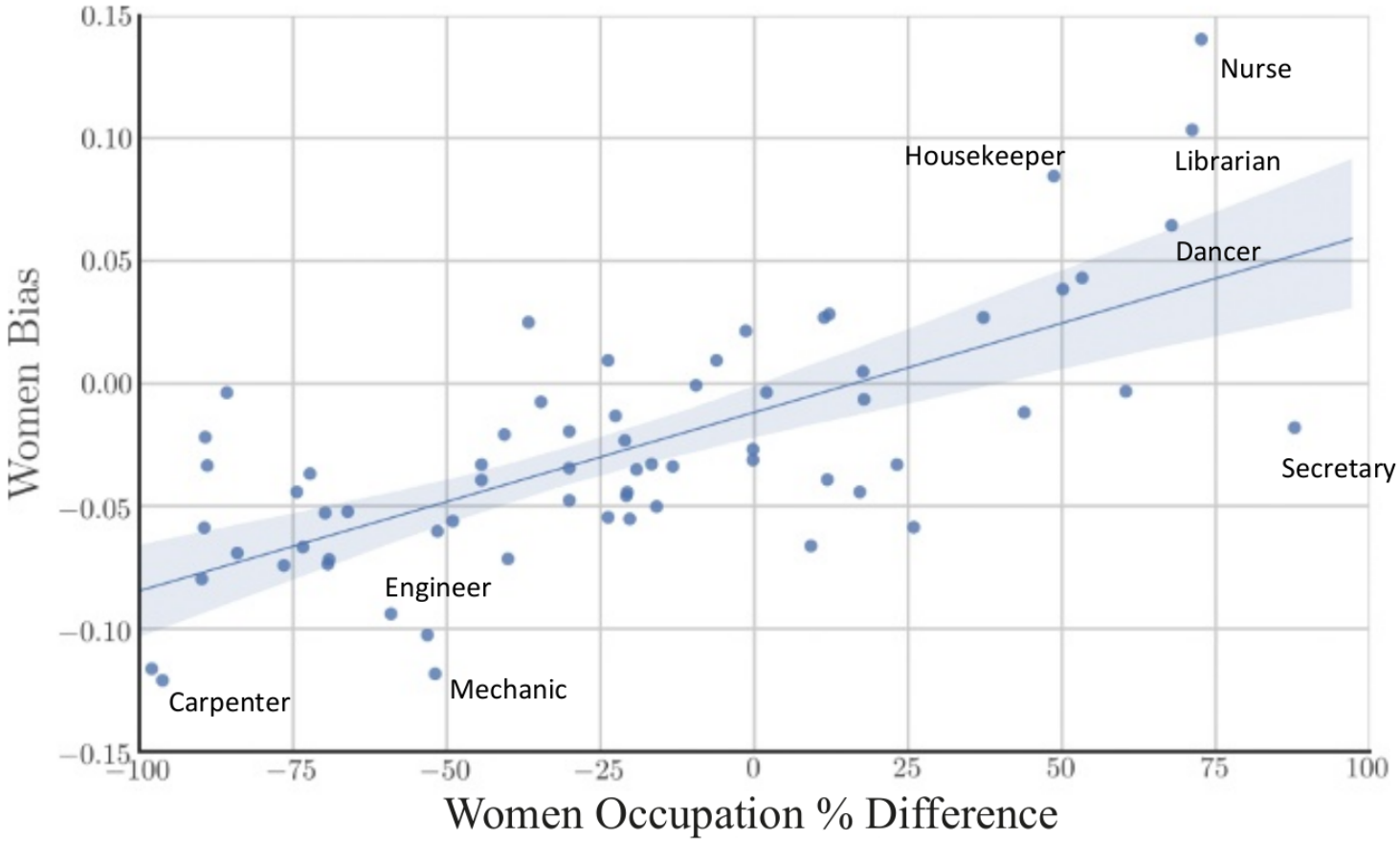


Fig. 1. Women's occupation relative percentage vs. embedding bias in Google News vectors. More positive indicates more associated with women on both axes. $P < 10^{-10}$, $r^2 = 0.499$. The shaded region is the 95% bootstrapped confidence interval of the regression line. In this single embedding, then, the association in the embedding effectively captures the percentage of women in an occupation.

Word embeddings and stereotypes

Table 1. The top 10 occupations most closely associated with each ethnic group in the Google News embedding

Hispanic	Asian	White
Housekeeper	Professor	Smith
Mason	Official	Blacksmith
Artist	Secretary	Surveyor
Janitor	Conductor	Sheriff
Dancer	Physicist	Weaver
Mechanic	Scientist	Administrator
Photographer	Chemist	Mason
Baker	Tailor	Statistician
Cashier	Accountant	Clergy
Driver	Engineer	Photographer

Word embeddings and stereotypes

Table 2. Top adjectives associated with women in 1910, 1950, and 1990 by relative norm difference in the COHA embedding

1910	1950	1990
Charming	Delicate	Maternal
Placid	Sweet	Morbid
Delicate	Charming	Artificial
Passionate	Transparent	Physical
Sweet	Placid	Caring
Dreamy	Childish	Emotional
Indulgent	Soft	Protective
Playful	Colorless	Attractive
Mellow	Tasteless	Soft
Sentimental	Agreeable	Tidy

Word embeddings and stereotypes

Table 3. Top Asian (vs. White) adjectives in 1910, 1950, and 1990 by relative norm difference in the COHA embedding

1910	1950	1990
Irresponsible	Disorganized	Inhibited
Envious	Outrageous	Passive
Barbaric	Pompous	Dissolute
Aggressive	Unstable	Haughty
Transparent	Effeminate	Complacent
Monstrous	Unprincipled	Forceful
Hateful	Venomous	Fixed
Cruel	Disobedient	Active
Greedy	Predatory	Sensitive
Bizarre	Boisterous	Hearty

Word embeddings and stereotypes

Fig. 4. Pearson correlation in embedding bias scores for adjectives over time between embeddings for each decade. The phase shift in the 1960s–1970s corresponds to the US women's movement.

Word embeddings and stereotypes

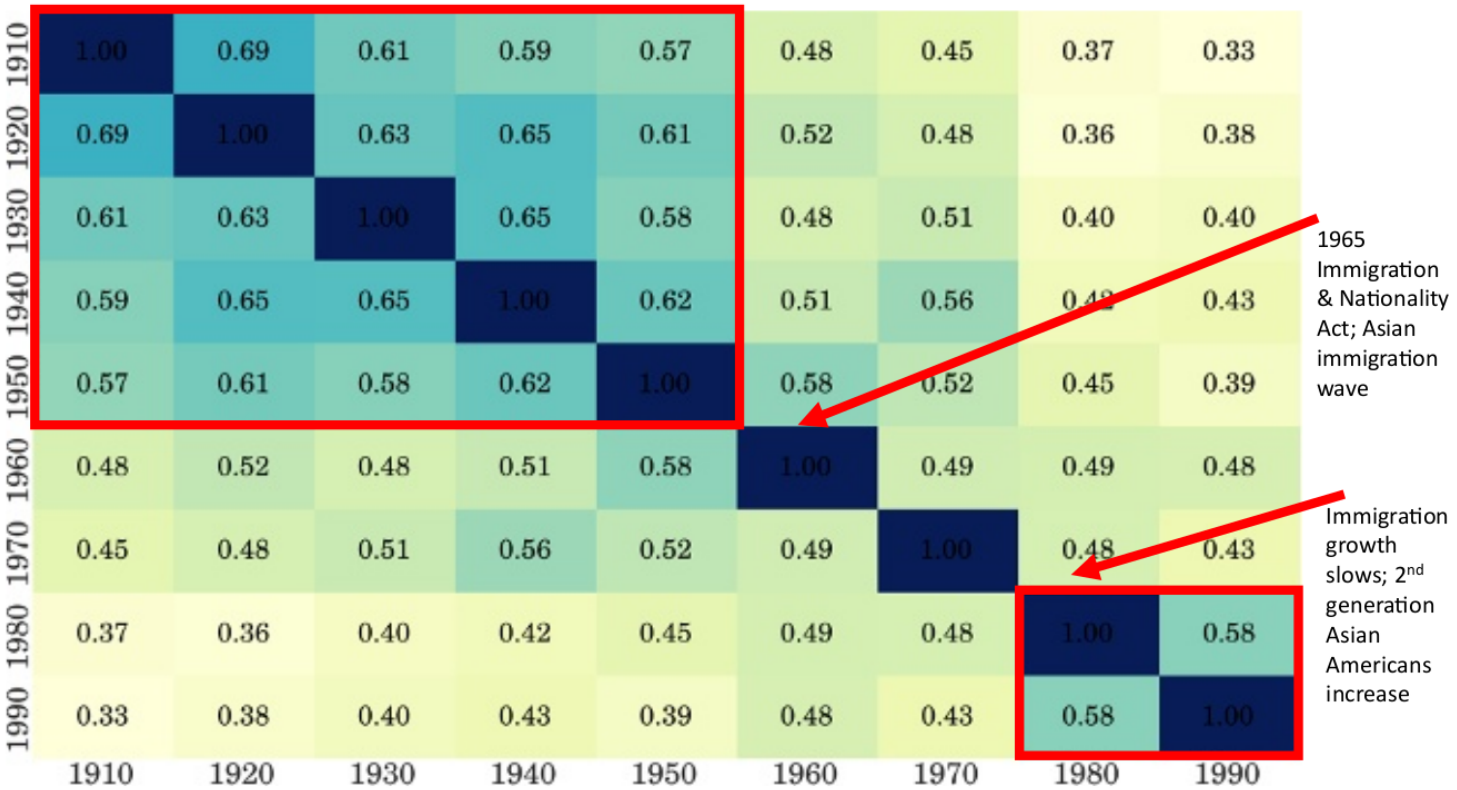


Fig. 5. Pearson correlation in embedding Asian bias scores for adjectives over time between embeddings for each decade.

Word embeddings and stereotypes

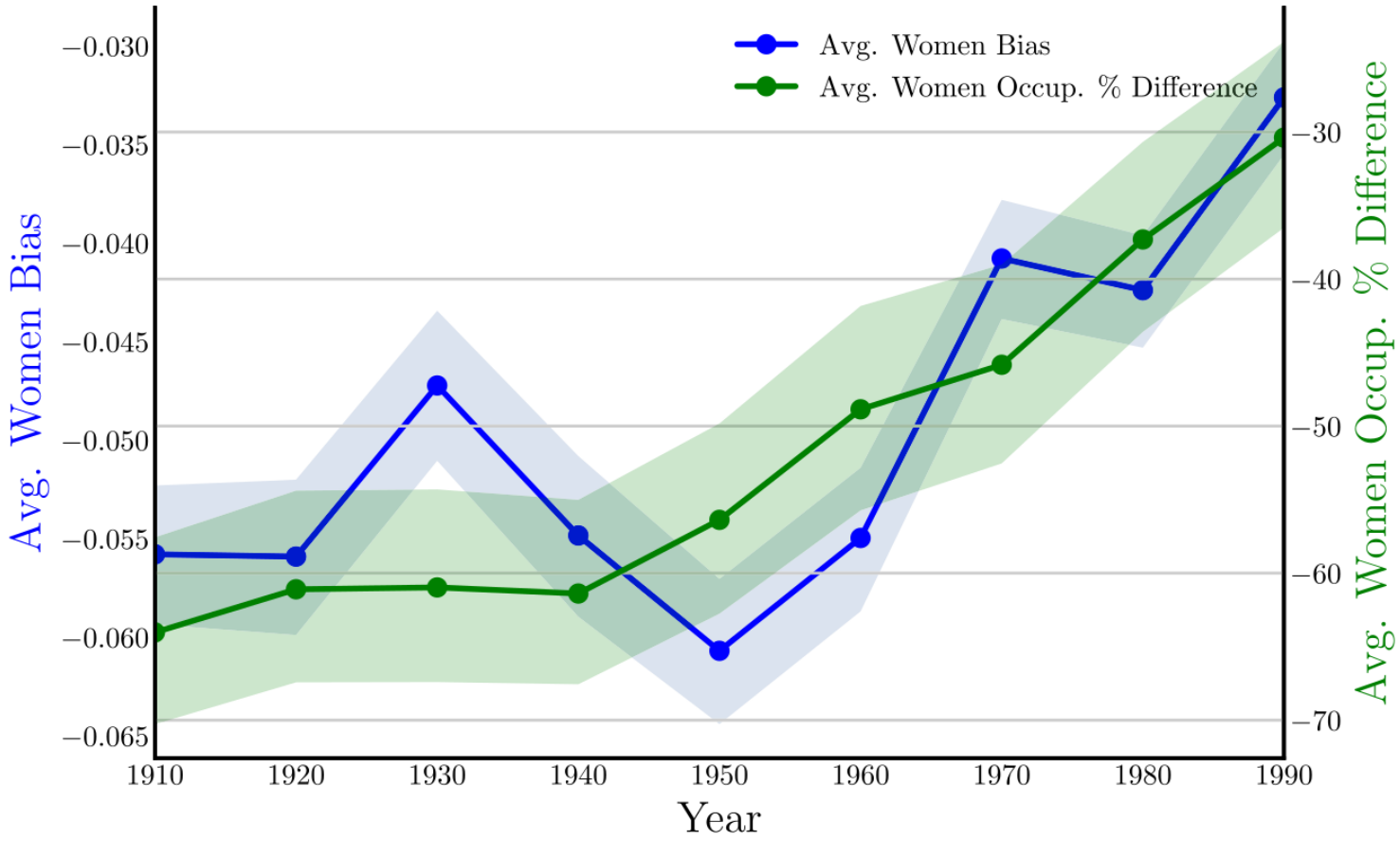


Fig. 2. Average gender bias score over time in COHA embeddings in occupations vs. the average percentage of difference. More positive means a stronger association with women. In blue is relative bias toward women in the embeddings, and in green is the average percentage of difference of women in the same occupations. Each shaded region is the bootstrap SE interval.

Word embeddings and stereotypes

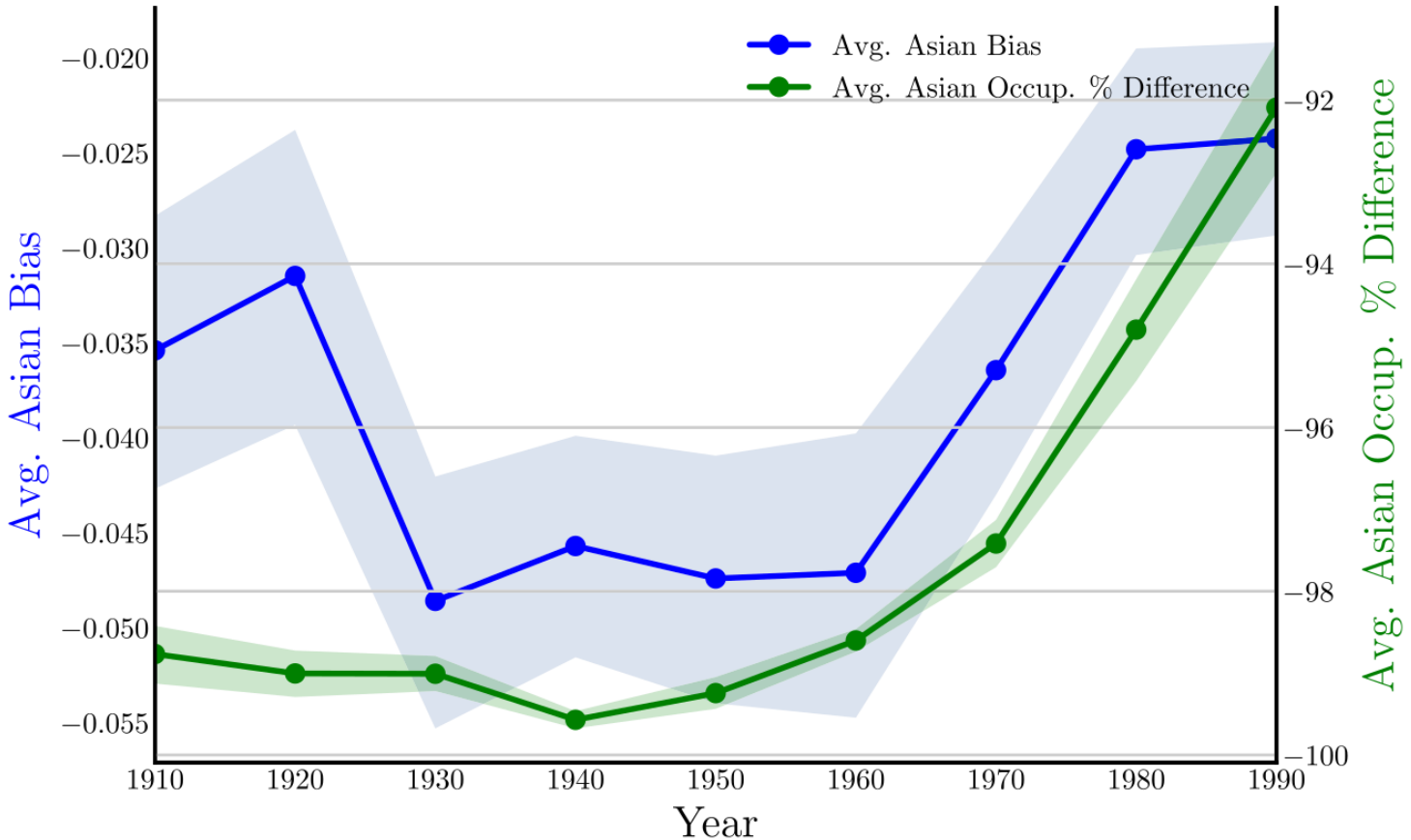


Fig. 3. Average ethnic (Asian vs. White) bias score over time for occupations in COHA (blue) vs. the average percentage of difference (green). Each shaded region is the bootstrap SE interval.

Word embeddings and stereotypes

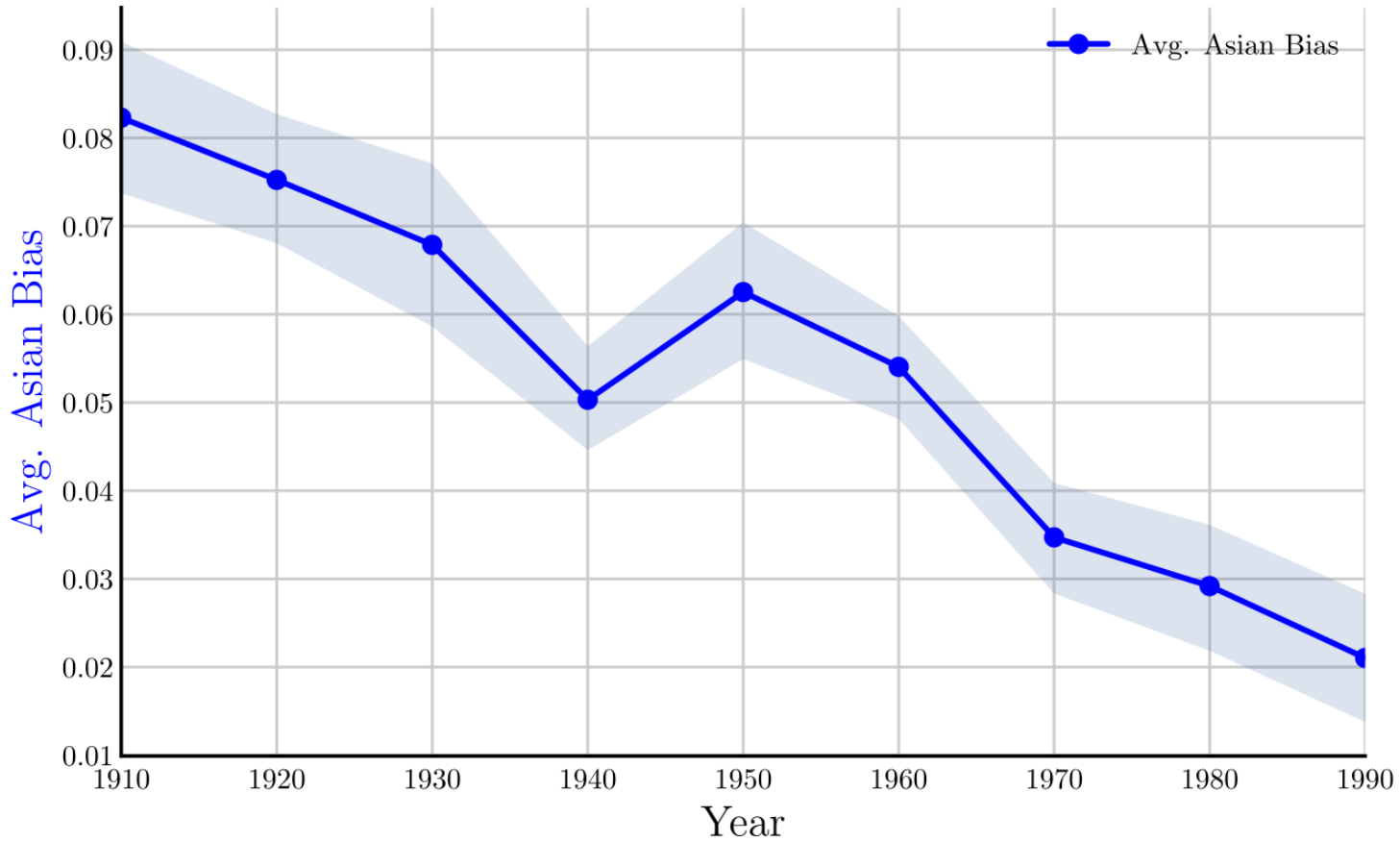


Fig. 6. Asian bias score over time for words related to outsiders in COHA data. The shaded region is the bootstrap SE interval.

Word embeddings and stereotypes

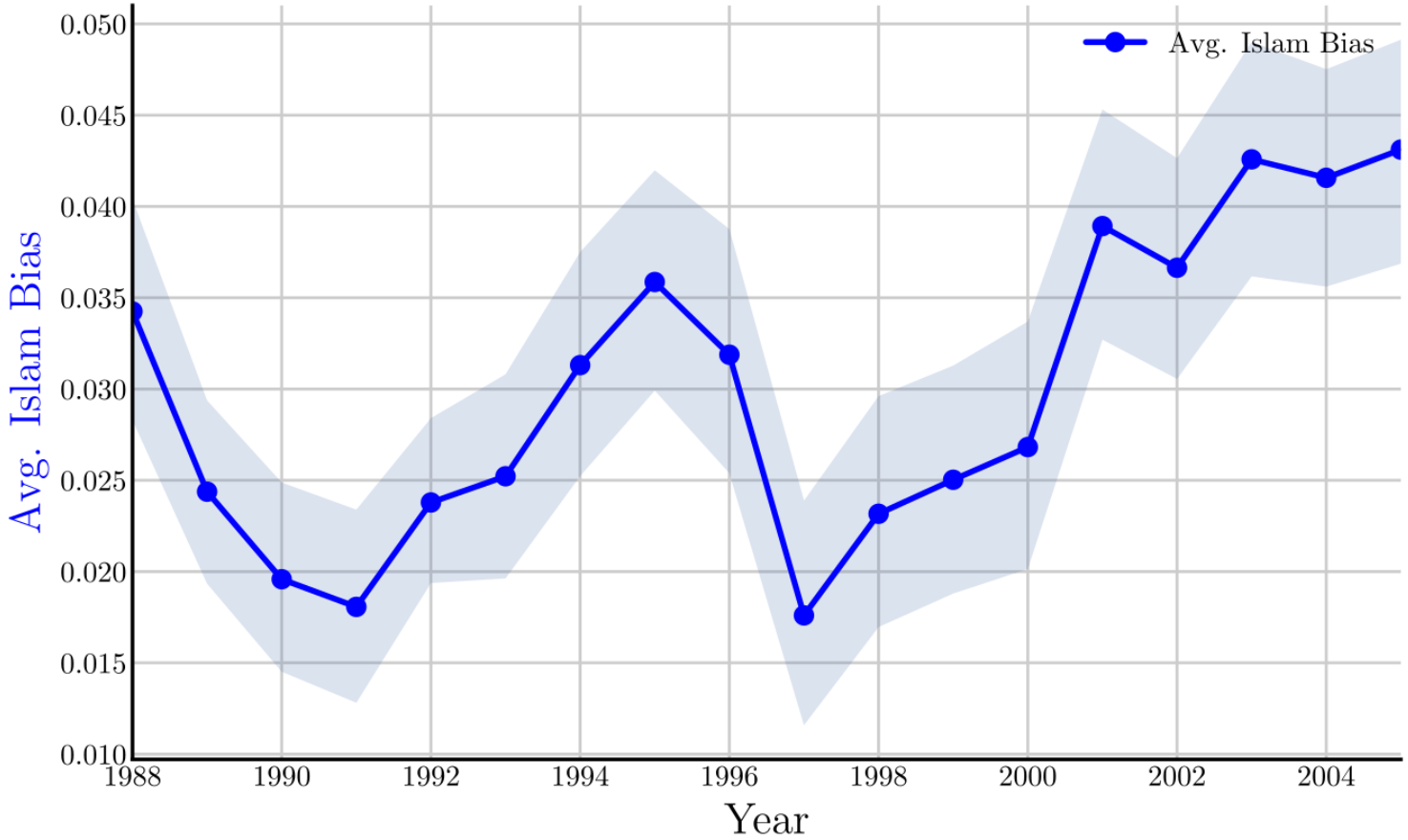


Fig. 7. Religious (Islam vs. Christianity) bias score over time for words related to terrorism in *New York Times* data. Note that embeddings are trained in 3-y windows, so, for example, 2000 contains data from 1999–2001. The shaded region is the bootstrap SE interval.

GloVe

- Pennington et al. 2014, *GloVe: Global Vectors for Word Representation*.
- Shallow window-based methods scan context windows across the corpus, failing to take advantage of repetition in the data.
- Matrix factorization methods (like SVD) operate directly on the co-occurrence statistics of the corpus.
- GloVe operates on the global co-occurrence statistics.
- GloVe also creates embeddings which capture semantic relations and allow for meaningful arithmetic operations.

GloVe: Example and intuition

Probability and ratio	k = solid	k = gas	k = water	k = fashion
$p(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$p(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$p(k\|ice)/p(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

- Consider two words, e.g. i = ice and j = steam.
- *Ice* co-occurs more frequently with *solid*, *steam* more frequently with *gas*. Both co-occur with *water* frequently, and rarely with *fashion*.

➥ Relationship between words can be examined by studying the *ratio* of their co-occurence probabilities with various probe words k.

GloVe: Example and intuition

Probability and ratio	k = solid	k = gas	k = water	k = fashion
$p(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$p(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$p(k\|ice)/p(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

- By taking the ratio, the noise from non-discriminative words cancels out.
- Large values ($\gg 1$) correlate with properties specific to ice.
- Small values ($\ll 1$) correlate with properties specific to steam.
- Ratio of probabilities encodes meaning.
- Starting point for word vector learning should be ratios rather than co-occurrence probabilities themselves.

GloVe

- Let c_{ij} be the number of times word j appears in the context of word i, and $c_i = \sum_k c_{ik}$ be the number of times any word appears in the context of word i. Probability that word j appears in the context of word i is $p_{ij} = p(j|i) = c_{ij}/c_i$.
- Ratio p_{ik}/p_{jk} depends on three words. Let $w \in \mathbb{R}^d$ and $\tilde{w} \in \mathbb{R}^d$ denote word and context vectors. Most general model takes the form

$$F(w_i, w_j, \tilde{w}_k) = \frac{p_{ik}}{p_{jk}}$$

- Natural to encode the information in the ratio as a vector difference.

$$F((w_i - w_j), \tilde{w}_k) = \frac{p_{ik}}{p_{jk}}$$

- Use the dot product to map vector space to a scalar.

$$F((w_i - w_j)^T \tilde{w}_k) = \frac{p_{ik}}{p_{jk}}$$

GloVe

- Model should be invariant to relabeling words and contexts. Require F to be a group homomorphism between $(\mathbb{R}, +)$ and $(\mathbb{R}_{>0}, \times)$.

$$F((w_i - w_j)^T \tilde{w}_k) = \frac{F(w_i^T \tilde{w}_k)}{F(w_j^T \tilde{w}_k)} = \frac{p_{ik}}{p_{jk}}$$

- Simplification (omitting a constant scaling factor)

$$F(w_i^T \tilde{w}_k) = p_{ik} = \frac{c_{ik}}{c_i}$$

- Solution is given by the exponential function for F.

$$\exp(w_i^T \tilde{w}_k) = \frac{c_{ik}}{c_i}$$

$$w_i^T \tilde{w}_k = \log(c_{ik}) - \log(c_i)$$

GloVe

- Symmetric with the exception of the final term.
- Add a bias term for each word. Absorb final term into first bias term (independent of k) to capture differences in base frequency. Add an additional bias term to restore symmetry (i and k interchangeable).

$$w_i^T \tilde{w}_k + b_i + \tilde{b}_k = \log(c_{ik})$$

- Potentially use Laplace smoothing since logarithm ill-defined for zero: $\log(1 + c_{ik})$ instead of $\log(c_{ik})$.
- Similar to LSA: Factorizing of co-occurrence statistics.

GloVe: Objective

- Minimize a weighted least squares regression model.

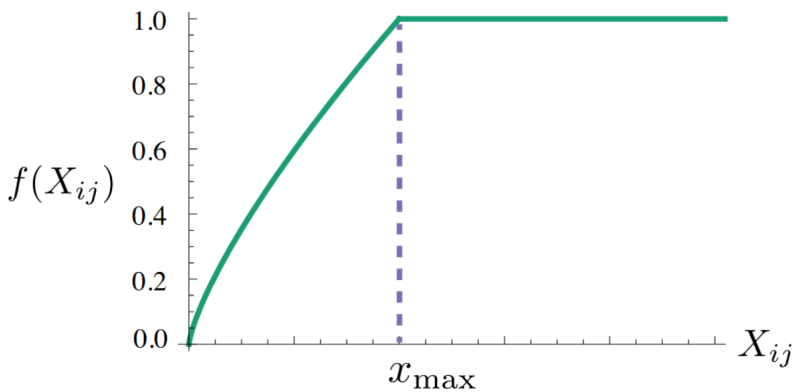
$$J = \sum_{i,j=1}^{V} f(c_{ij})(w_i^T \tilde{w}_k + b_i + \tilde{b}_k - \log(c_{ik}))^2$$

- Estimation using stochastic gradient descent.
- Drawback: Model weighs all co-occurrences equally.
- Rare co-occurrences are noisy and carry less information. Much of the data in $\boldsymbol{C}$ (regularly 75%–95%, depending on the corpus) consists of zeros.
- Add a weighting function $f(c_{ij})$ to the cost function.

GloVe: Weight function

- Weighting function $f(c)$ should satisfy following properties:
  1. $f(0) = 0$
  2. Non-decreasing (small co-occurences not overweighted).
  3. Relatively small for large c (frequent co-occurences not overweighted).
- In practice, good performance given by

$$f(x) = \begin{cases} (c/c_{max})^{\alpha} & \text{if } x < x_{max} \\ 1 & \text{otherwise} \end{cases}$$



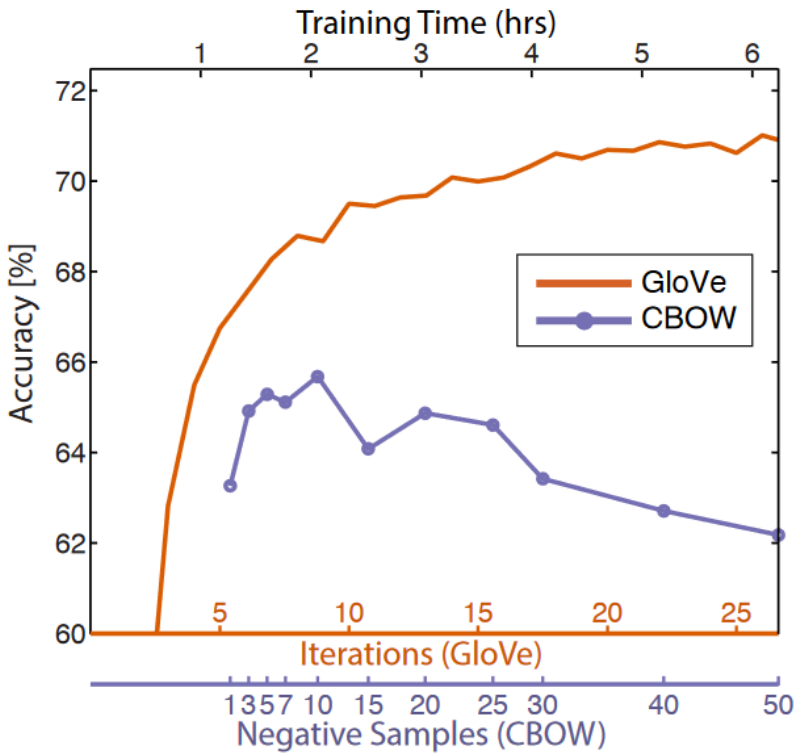
with $\alpha = 3/4$ and $x_{max} = 100$.

- Motivation purely empirical.
- Similar 0.75 fractional power scaling works well for word2vec.

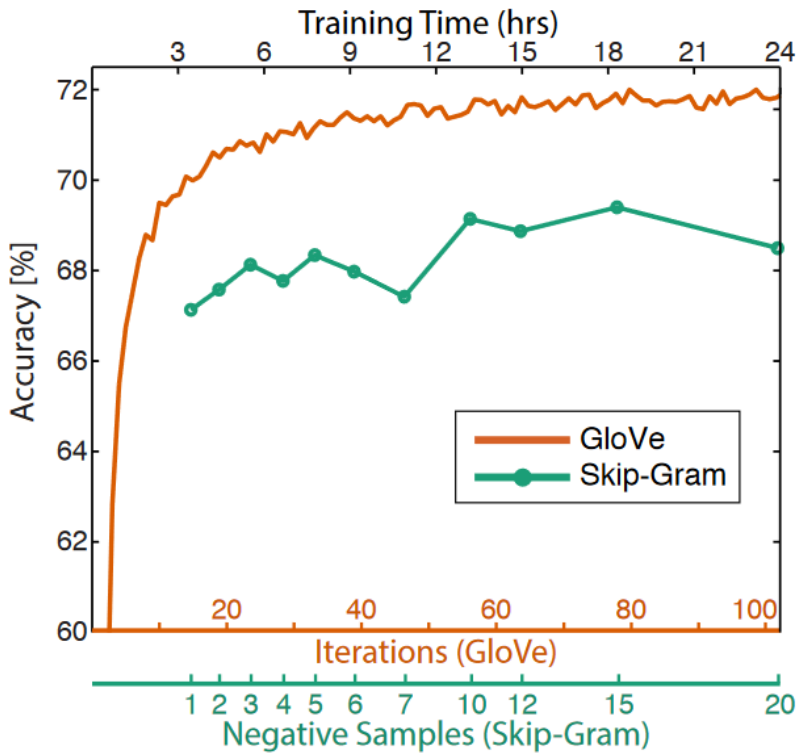
GloVe

- Captures both global count statistics and meaningful linear substructures.
- Performs well with similarity and analogy tasks.
- Fast to train and good performance even with small corpora.
- Like SGNS, no expensive evaluation of a softmax normalization factor.
- Uses a lot of memory—fastest way is to keep co-occurence matrix in memory.
- Pre-trained embeddings can be downloaded.

GloVe vs. word2vec



(a) GloVe vs CBOW



(b) GloVe vs Skip-Gram

FastText

- Bojanowski et al. (2017), *Enriching Word Vectors with Subword Information*.
- Developed at Facebook. Code and pre-trained models available.
- word2vec and GloVe consider words to be the smallest entities.
- fastText can be considered an extension of word2vec.
- Also considers words to be composed of character n-grams (in addition to word context).

Remarks

- Dense embeddings better to include as features in machine learning methods.
- Unsupervised: Embeddings can be learned from any text.
- Learning embeddings is pointless without substantial data.
- And even with sufficient data often unnecessary, unless the application domain is very specific.
- Plenty of pre-trained word embeddings available (e.g. GloVe, fasttext).
- Trained on Wikipedia, Common Crawl or Twitter.

Caveat

- We are typically more interested in *document embeddings*.
- Document-level embeddings could be used as input for selection-on-observables methods or for prediction/supervised learning.

➥ How to obtain an embedding representation on the document level?

Caveat

- Simple approach: Choose an aggregation function (e.g. the mean) and apply it on the document level.
- Alternative: Extend method to learn embeddings at the document level.
- Example: Paragraph vectors/doc2vec (Le and Mikolov 2014). Learns document embedding by adding document-specific encoding vector (predict center word using context words plus document id, or predict output word sampled from document with id).
- Can be used with pre-trained word embeddings.
- Much more work in this area (e.g. paragraph vectors, Dai et al. 2015; skip-thought, Kiros et al. 2015, BERT at Google Research).

Concluding remarks

Concluding remarks

- Few comprehensive textbooks on this subject.
- Field develops fast, many possible applications outside economics.
- Solid simulation evidence not always available.
- Potentially useful for economics—*how* useful depends on application.
- Methods need to be tailored to the application.
- Pre-processing matters. Domain-specific prior information helps.

Concluding remarks

- Penalized regression/naive Bayes is simple to apply.
- Latent classification methods can be illustrative. May also help obtaining measures for things which are difficult to measure otherwise.
- Embedding methods potentially useful in many contexts.
- Causal inference: Vector space representation may be helpful to identify control group (if text data is detailed enough to assume unconfoundedness).
- But using text for identification can be problematic: Often text is an outcome by itself rather than an exogenous covariate.

What next?

- What you should learn next: Deep learning.
- Neural networks are incredibly flexible and powerful tools.
- Do not suffer from the curse of dimensionality under certain conditions.
- Universal function approximators.
- Double descent phenomenon.
- Language modeling today is almost exclusively deep learning based.
- Pre-trained models are widely available. Make sure you understand the basics first.

Appendix

Digression: Gradient descent

Gradient descent

- Consider unconstrained optimization of some smooth convex differentiable function f with domain $\mathcal{R}^n$:

$$\min_x \ f(x)$$

- Choose initial point $x^{(0)} \in \mathcal{R}^n$ and step size γ_k, repeat:

$$x^{(k)} = x^{(k-1)} - \gamma_k \cdot \nabla f(x^{(k-1)}) \ , \ k = 1, 2, 3, \ldots$$

- Stop at some point when changes are small, $\|\nabla f(x)\|_2 \leq \epsilon$.

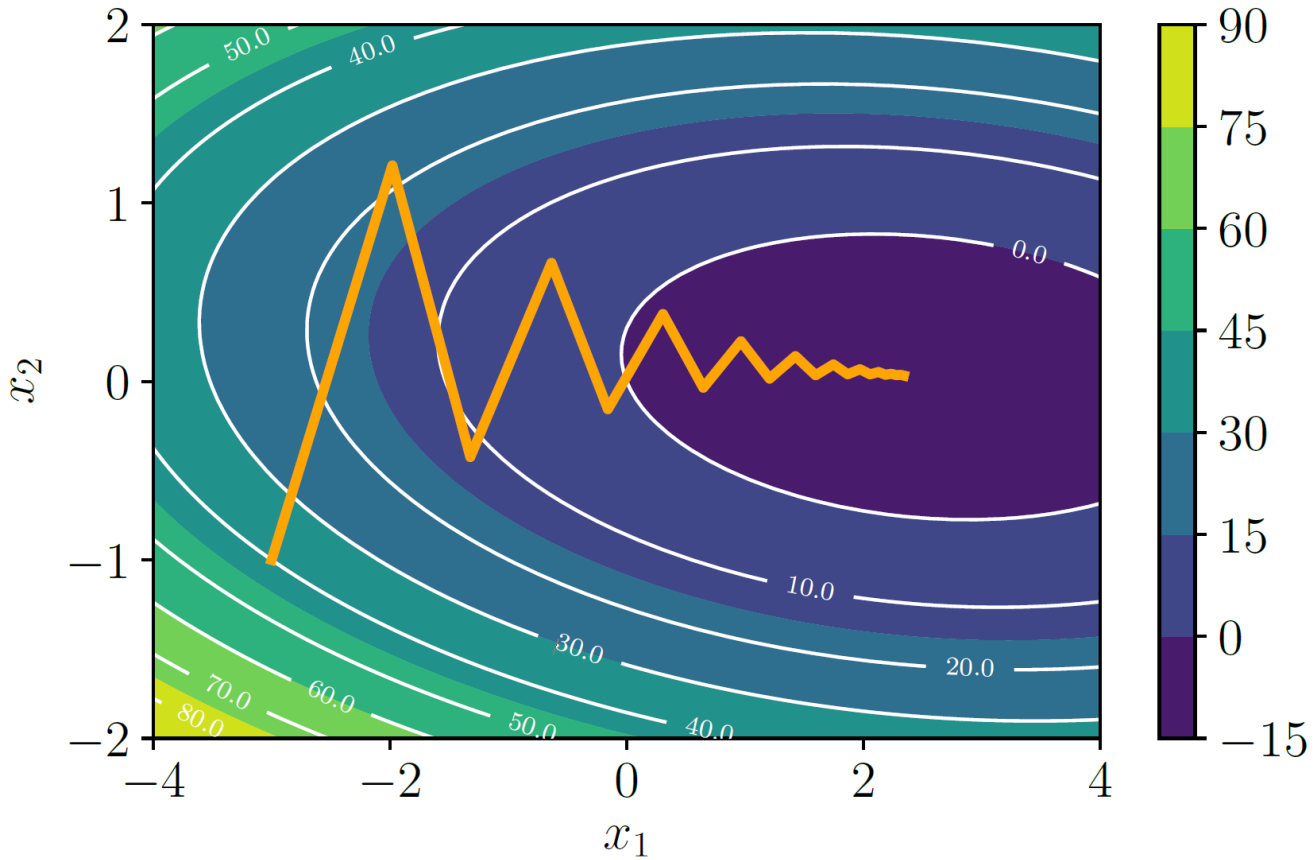
Digression: Gradient descent

Main

$$x^{(k)} = x^{(k-1)} - \gamma_k \cdot \nabla f(x^{(k-1)})$$

- Update parameter by taking a small step in the direction of the negative gradient. Note $f(x^{(k)}) < f(x^{(k-1)})$.
- $-\nabla f(x^{(k-1)})$ is the step/descent direction.
- γ_k is the step size (fixed if $\gamma_k = \gamma$).
- Too small step size will slow convergence.
- Too large step size could cause overshooting of minima and divergence.

Digression: Gradient descent



Digression: Gradient descent

Main

- Generic optimization method.
- Gradient descent can be used to minimize negative log-likelihood or other loss functions (e.g. squared error loss for linear models).
- ‘Back propagation’ in neural networks is the same as gradient descent.
- Computing complicated gradients for large data sets is expensive.

Digression: Gradient descent

Main

- Consider the objective function as a sum of losses over all data points, and θ as the vector of parameters of interest:

$$L(\theta) = \sum_{i=1}^{N} L_i(\theta)$$

- Regression example: Minimizing the negative log-likelihood.

$$L(\theta) = -\sum_{i=1}^{N} \log p(y_i|x_i, \theta).$$

- Update the parameter vector according to

$$\theta^k = \theta^{(k-1)} - \gamma^k \nabla L(\theta) = \theta^{(k-1)} - \gamma^k \sum_{i=1}^{N} \nabla L_i(\theta)$$

Digression: Gradient descent

Main

- Evaluating the sum gradient requires expensive evaluation of the gradients from all individual functions L_i.
- Batch gradient descent uses all L_i for $i = 1, \ldots, N$.
- Amount of computation can be reduced by taking a sum over a smaller set of L_i.
- Any unbiased estimate of the expected value of the gradient suffices.
- Sufficient to compute the gradient for a small sample (or even a single observation).
- Mini-batch/stochastic gradient descent uses only a random subset of L_i.
- Only stochastic gradient descent used in practice.

Digression: Gradient descent

- Simple and easy to implement.
- Versatile, can be applied to many different problems.

- Local optima problem.
- Slow close to mimimum.
- Ill-defined for non-differentiable functions.

References

References

Bojanowski, P., E. Grave, A. Joulin, and T. Mikolov (2017). Enriching Word Vectors with Subword Information. *arXiv:1607.04606 [cs]*.

Bolukbasi, T., K.-W. Chang, J. Zou, V. Saligrama, and A. Kalai (2016). Man Is to Computer Programmer as Woman Is to Homemaker? Debiasing Word Embeddings. *arXiv:1607.06520 [cs, stat]*.

Dai, A. M., C. Olah, and Q. V. Le (2015). Document Embedding with Paragraph Vectors. *arXiv:1507.07998 [cs]*.

Deerwester, S., S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman (1990). Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science* 41(6), 391–407. DOI: 10.1002/(sici)1097-4571(199009)41:6<391::aid-asi1>3.0.co;2-9.

Deisenroth, M. P., A. A. Faisal, and C. S. Ong (2020). *Mathematics for Machine Learning*. Cambridge ; New York, NY: Cambridge University Press.

Fano, R. M. (1963). *Transmission of Information: A Statistical Theory of Communications*. M.I.T. Press.

Firth, J. R. (1957). A Synopsis of Linguistic Theory 1930-55. 1952-59, 1–32.

References

Garg, N., L. Schiebinger, D. Jurafsky, and J. Zou (2018). Word Embeddings Quantify 100 Years of Gender and Ethnic Stereotypes. *Proceedings of the National Academy of Sciences* 115(16), E3635–E3644. DOI: 10/gdfq9g.

Goldberg, Y. (2017). *Neural Network Methods for Natural Language Processing*. Synthesis Lectures on Human Language Technologies 37. San Rafael: Morgan & Claypool Publishers.

Goldberg, Y. and O. Levy (2014). Word2vec Explained: Deriving Mikolov et al.'s Negative-Sampling Word-Embedding Method.

Hotelling, H. (1933). Analysis of a Complex of Statistical Variables into Principal Components. *Journal of Educational Psychology* 24(6), 417–441. DOI: 10/fb5435.

Kiros, R., Y. Zhu, R. Salakhutdinov, R. S. Zemel, A. Torralba, R. Urtasun, and S. Fidler (2015). Skip-Thought Vectors. *arXiv:1506.06726 [cs]*.

Le, Q. V. and T. Mikolov (2014). Distributed Representations of Sentences and Documents. *arXiv:1405.4053 [cs]*.

References


Levy, O. and Y. Goldberg (2014). Neural Word Embedding as Implicit Matrix Factorization. In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger. Curran Associates, Inc., 2177–2185.


Mikolov, T., K. Chen, G. Corrado, and J. Dean (2013). Efficient Estimation of Word Representations in Vector Space. *arXiv:1301.3781 [cs]*.


Mikolov, T., I. Sutskever, K. Chen, G. S. Corrado, and J. Dean (2013). Distributed Representations of Words and Phrases and Their Compositionality. In: *Advances in Neural Information Processing Systems 26*. Ed. by C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger. Curran Associates, Inc., 3111–3119.


Pearson, K. (1901). LIII. On Lines and Planes of Closest Fit to Systems of Points in Space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2(11), 559–572. DOI: 10/dd63n4.

References

Pennington, J., R. Socher, and C. Manning (2014). “GloVe: Global Vectors for Word Representation”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, 1532–1543. DOI: 10/gfshwg.