Natural Language Processing

Lecture VII. Word Embedding – a 30-year journey

Forrest Sheng Bao, Ph.D.

Dept. of Computer Science
Iowa State University
Ames, IA 50011

October 20, 2022

Something about graduate school

“The path to real success is not to compete, but to invent a new game, and then master it.” Reid Hoffman,

https://www.linkedin.com/pulse/dont-just-compete-invent-new-game-master-reid-hoffman

Salute to the NLP pioneers, including but not limited to: Elman (1990), Bengio (2003), Collobert & Weston (2008, look-up table), Mnih & Hinton (2007 & 2009, tree).

Language model review

- BOW or unigram: no order of words
- N-gram where $N > 1$: a sequence of words, some structural information.
- A classical language model estimates a cost function (e.g., likelihood) of word sequences.
- Problem?
  - "curse of dimensionality"

    Too many probabilities!
  - What about unseen combinations (not just words)? Smoothing is not enough.
  - How to plug into neural networks?

Language model review

- BOW or unigram: no order of words
- N-gram where $N > 1$: a sequence of words, some structural information.
- A classical language model estimates a cost function (e.g., likelihood) of word sequences.
- Problem?
  - "curse of dimensionality" $P(w_1, \ldots, w_n) = P(w_2|w_1) \times P(w_3|w_1, w_2) \times \cdots \times P(w_n|w_1, \ldots, w_{n-1}) = \prod_{i=2}^{n} P(w_i|w_1, \ldots, w_{i-1}) \approx \prod_{i=2}^{n} P(w_i|w_{i-N+1}, \ldots, w_{i-1})$. Too many probabilities!
  - What about unseen combinations (not just words)? Smoothing is not enough.
  - How to plug into neural networks?

Language model review

- BOW or unigram: no order of words
- N-gram where $N > 1$: a sequence of words, some structural information.
- A classical language model estimates a cost function (e.g., likelihood) of word sequences.
- Problem?
  - "curse of dimensionality" $P(w_1, \ldots, w_n) = P(w_2|w_1) \times P(w_3|w_1, w_2) \times \cdots \times P(w_n|w_1, \ldots, w_{n-1}) = \prod_{i=2}^{n} P(w_i|w_1, \ldots, w_{i-1}) \approx \prod_{i=2}^{n} P(w_i|w_{i-\tau-1}, \ldots, w_{i-1})$ Too many probablities!
  - What about unseen combinations (not just words)? Smoothing is not enough.
  - How to plug into neural networks?

Language model review

- BOW or unigram: no order of words
- N-gram where $N > 1$: a sequence of words, some structural information.
- A classical language model estimates a cost function (e.g., likelihood) of word sequences.
- Problem?
  - "curse of dimensionality" $P(w_1, \ldots, w_n) = P(w_2|w_1) \times P(w_3|w_1, w_2) \times \cdots \times P(w_n|w_1, \ldots, w_{n-1}) = \prod_{i=2}^{n} P(w_i|w_1, \ldots, w_{i-1}) \approx \prod_{i=2}^{n} P(w_i|w_{i-\tau-1}, \ldots, w_{i-1})$ Too many probablities!
  - What about unseen combinations (not just words)? Smoothing is not enough.
  - How to plug into neural networks?

Language model review

- BOW or unigram: no order of words
- N-gram where $N > 1$: a sequence of words, some structural information.
- A classical language model estimates a cost function (e.g., likelihood) of word sequences.
- Problem?
  - "curse of dimensionality" $P(w_1, \ldots, w_n) = P(w_2|w_1) \times P(w_3|w_1, w_2) \times \cdots \times P(w_n|w_1, \ldots, w_{n-1}) = \prod_{i=2}^{n} P(w_i|w_1, \ldots, w_{i-1}) \approx \prod_{i=2}^{n} P(w_i|w_{i-\tau-1}, \ldots, w_{i-1})$ Too many probablities!
  - What about unseen combinations (not just words)? Smoothing is not enough.
  - How to plug into neural networks?

Language model review

- BOW or unigram: no order of words
- N-gram where $N > 1$: a sequence of words, some structural information.
- A classical language model estimates a cost function (e.g., likelihood) of word sequences.
- Problem?
  - "curse of dimensionality"
    $P(w_1, \ldots, w_n) = P(w_2|w_1) \times P(w_3|w_1, w_2) \times \cdots \times P(w_n|w_1, \ldots, w_{n-1}) = \prod_{i=2}^{n} P(w_i|w_1, \ldots, w_{i-1}) \approx \prod_{i=2}^{n} P(w_i|w_{i-\tau-1}, \ldots, w_{i-1})$ Too many probablities!
  - What about unseen combinations (not just words)? Smoothing is not enough.
  - How to plug into neural networks?

Language model review

- BOW or unigram: no order of words
- N-gram where $N > 1$: a sequence of words, some structural information.
- A classical language model estimates a cost function (e.g., likelihood) of word sequences.
- Problem?
  - "curse of dimensionality" $P(w_1, \ldots, w_n) = P(w_2|w_1) \times P(w_3|w_1, w_2) \times \cdots \times P(w_n|w_1, \ldots, w_{n-1}) = \prod_{i=2}^{n} P(w_i|w_1, \ldots, w_{i-1}) \approx \prod_{i=2}^{n} P(w_i|w_{i-\tau-1}, \ldots, w_{i-1})$ Too many probablities!
  - What about unseen combinations (not just words)? Smoothing is not enough.
  - How to plug into neural networks?

How to plug words into an ANN

- A very good explanation from TF v1 tutorial about why word needs to be vectorized. https://github.com/tensorflow/docs/blob/r1.15/site/en/tutorials/representation/word2vec.md
- How do we send sequences of words into an NN?
- Using ASCII code? Using UTF code?
- Turning words into vectors (the simple ways):
  - one-hot encoder (Finding Structure in Time, Elman 1990: each word is a 31 bit vector)
  - co-occurrence matrix (e.g., P("ice", "solid"), P("ice", "gas"))
  - factorization on the co-occurrence matrix (dimension reduction) such as SVD

How to plug words into an ANN

- A very good explanation from TF v1 tutorial about why word needs to be vectorized. `https://github.com/tensorflow/docs/blob/r1.15/site/en/tutorials/representation/word2vec.md`
- How do we send sequences of words into an NN?
- Using ASCII code? Using UTF code?
- Turning words into vectors (the simple ways):
  - one-hot encoder (Finding Structure in Time, Elman, 1990, Table 5, each word is a 31-bit vector)
  - co-occurrence matrix (e.g., P("fox", "jump"), P("lazy", "dog"))
  - factorization on the co-occurrence matrix (to reduce dimensionality) such as SVD

How to plug words into an ANN

- A very good explanation from TF v1 tutorial about why word needs to be vectorized. https://github.com/tensorflow/docs/blob/r1.15/site/en/tutorials/representation/word2vec.md
- How do we send sequences of words into an NN?
- Using ASCII code? Using UTF code?
- Turning words into vectors (the simple ways):
  - one-hot encoder (Finding Structure in Time, Elman, 1990, Table 5, each word is a 31-bit vector)
  - co-occurrence matrix (e.g., P("fox", "jump"), P("lazy", "dog"))
  - factorization on the co-occurrence matrix (to reduce dimensionality) such as SVD

How to plug words into an ANN

- A very good explanation from TF v1 tutorial about why word needs to be vectorized. https://github.com/tensorflow/docs/blob/r1.15/site/en/tutorials/representation/word2vec.md
- How do we send sequences of words into an NN?
- Using ASCII code? Using UTF code?
- Turning words into vectors (the simple ways):
  - one-hot encoder (Finding Structure in Time, Elman, 1990, Table 5, each word is a 31-bit vector)
  - co-occurrence matrix (e.g., P("fox", "jump"), P("lazy", "dog"))
  - factorization on the co-occurrence matrix (to reduce dimensionality) such as SVD

How to plug words into an ANN

- A very good explanation from TF v1 tutorial about why word needs to be vectorized. https://github.com/tensorflow/docs/blob/r1.15/site/en/tutorials/representation/word2vec.md
- How do we send sequences of words into an NN?
- Using ASCII code? Using UTF code?
- Turning words into vectors (the simple ways):
  - one-hot encoder (Finding Structure in Time, Elman, 1990, Table 5, each word is a 31-bit vector)
  - co-occurrence matrix (e.g., P("fox", "jump"), P("lazy", "dog"))
  - factorization on the co-occurrence matrix (to reduce dimensionality) such as SVD

How to plug words into an ANN

- A very good explanation from TF v1 tutorial about why word needs to be vectorized. `https://github.com/tensorflow/docs/blob/r1.15/site/en/tutorials/representation/word2vec.md`
- How do we send sequences of words into an NN?
- Using ASCII code? Using UTF code?
- Turning words into vectors (the simple ways):
  - one-hot encoder (Finding Structure in Time, Elman, 1990, Table 5, each word is a 31-bit vector)
  - co-occurrence matrix (e.g., P("fox", "jump"), P("lazy", "dog"))
  - factorization on the co-occurrence matrix (to reduce dimensionality) such as SVD

How to plug words into an ANN

- A very good explanation from TF v1 tutorial about why word needs to be vectorized. https://github.com/tensorflow/docs/blob/r1.15/site/en/tutorials/representation/word2vec.md
- How do we send sequences of words into an NN?
- Using ASCII code? Using UTF code?
- Turning words into vectors (the simple ways):
  - one-hot encoder (Finding Structure in Time, Elman, 1990, Table 5, each word is a 31-bit vector)
  - co-occurrence matrix (e.g., P("fox", "jump"), P("lazy", "dog"))
  - factorization on the co-occurrence matrix (to reduce dimensionality) such as SVD

Word representation

- "A word representation is a mathematical object associated with each word, often a vector." [1]
- "Each dimension's value corresponds to a feature and might even have a semantic or grammatical interpretation, so we call it a word feature."
- One-hot (aka 1-of-N) encoding is one, but obviously not good.
- Word embedding: a distributed representation

  Ref [1]: Turian, Ratinov, and Bengio, Word representations: A simple and general method for semi-supervised learning, ACL 2010

Embedding via Factorization

- Intuition: Semantically (dis)similar/(un)related words should co-occur in documents (in)frequently.
- A word-document co-occurence matrix can be considered as a composition of a series of transforms.
  1. Each word is a distribution over given semantic dimensions, e.g., “water” covers “liquid”, “clear”, and “oderless”.
  2. A document is generated by sampling words in different semantic dimensions, thus transform the word probabilities to their distributions in documents.
  3. Some semantic dimensions are more important.
- This is the idea behind Singular Vector Decomposition (SVD): $M = U\Sigma V$, where all rows of U or all columns of V are orthorgonal, and Σ a diagnoal matrix of signular values (importances of semantic dimensions).
- The dimension of U is high. Usually we zero out some dimensions in Σ to focus on only the important ones. And thus, the resulting U is no longer orthornormal.
- SVD on word-document co-occurence [2]. Note that the SVD in this tutorial is called compact SVD.
- The co-occurence counts can be further weighted into other quantities, such as point-wise mutual information (Slide 37 of [2]).

Refs:

Embedding via Factorization

- Intuition: Semantically (dis)similar/(un)related words should co-occur in documents (in)frequently.
- A word-document co-occurence matrix can be considered as a composition of a series of transforms.
  1. Each word is a distribution over given semantic dimensions, e.g., “water” covers “liquid”, “clear”, and “oderless”.
  2. A document is generated by sampling words in different semantic dimensions, thus transform the word probabilities to their distributions in documents.
  3. Some semantic dimensions are more important.
- This is the idea behind Singular Vector Decomposition (SVD): $M = U\Sigma V$, where all rows of U or all columns of V are orthorgonal, and Σ a diagnoal matrix of signular values (importances of semantic dimensions).
- The dimension of U is high. Usually we zero out some dimensions in Σ to focus on only the important ones. And thus, the resulting U is no longer orthornormal.
- SVD on word-document co-occurence [2]. Note that the SVD in this tutorial is called compact SVD.
- The co-occurence counts can be further weighted into other quantities, such as point-wise mutual information (Slide 37 of [2]).

Refs:

Embedding via Factorization

- Intuition: Semantically (dis)similar/(un)related words should co-occur in documents (in)frequently.
- A word-document co-occurence matrix can be considered as a composition of a series of transforms.
  1. Each word is a distribution over given semantic dimensions, e.g., “water” covers “liquid”, “clear”, and “oderless”.
  2. A document is generated by sampling words in different semantic dimensions, thus transform the word probabilities to their distributions in documents.
  3. Some semantic dimensions are more important.
- This is the idea behind Singular Vector Decomposition (SVD): $M = U\Sigma V$, where all rows of U or all columns of V are orthorgonal, and Σ a diagnoal matrix of signular values (importances of semantic dimensions).
- The dimension of U is high. Usually we zero out some dimensions in Σ to focus on only the important ones. And thus, the resulting U is no longer orthornormal.
- SVD on word-document co-occurence [2]. Note that the SVD in this tutorial is called compact SVD.
- The co-occurence counts can be further weighted into other quantities, such as point-wise mutual information (Slide 37 of [2]).

Refs:

Embedding via Factorization

- Intuition: Semantically (dis)similar/(un)related words should co-occur in documents (in)frequently.
- A word-document co-occurence matrix can be considered as a composition of a series of transforms.
  1. Each word is a distribution over given semantic dimensions, e.g., “water” covers “liquid”, “clear”, and “oderless”.
  2. A document is generated by sampling words in different semantic dimensions, thus transform the word probabilities to their distributions in documents.
  3. Some semantic dimensions are more important.
- This is the idea behind Singular Vector Decomposition (SVD): $M = U\Sigma V$, where all rows of U or all columns of V are orthorgonal, and Σ a diagnoal matrix of signular values (importances of semantic dimensions).
- The dimension of U is high. Usually we zero out some dimensions in Σ to focus on only the important ones. And thus, the resulting U is no longer orthornormal.
- SVD on word-document co-occurence [2]. Note that the SVD in this tutorial is called compact SVD.
- The co-occurence counts can be further weighted into other quantities, such as point-wise mutual information (Slide 37 of [2]).

Refs:

Embedding via Factorization

- Intuition: Semantically (dis)similar/(un)related words should co-occur in documents (in)frequently.
- A word-document co-occurence matrix can be considered as a composition of a series of transforms.
  1. Each word is a distribution over given semantic dimensions, e.g., “water” covers “liquid”, “clear”, and “oderless”.
  2. A document is generated by sampling words in different semantic dimensions, thus transform the word probabilities to their distributions in documents.
  3. Some semantic dimensions are more important.
- This is the idea behind Singular Vector Decomposition (SVD): $M = U\Sigma V$, where all rows of U or all columns of V are orthorgonal, and Σ a diagnoal matrix of signular values (importances of semantic dimensions).
- The dimension of U is high. Usually we zero out some dimensions in Σ to focus on only the important ones. And thus, the resulting U is no longer orthornormal.
- SVD on word-document co-occurence [2]. Note that the SVD in this tutorial is called compact SVD.
- The co-occurence counts can be further weighted into other quantities, such as point-wise mutual information (Slide 37 of [2]).

Refs:

Embedding via Factorization

- Intuition: Semantically (dis)similar/(un)related words should co-occur in documents (in)frequently.
- A word-document co-occurence matrix can be considered as a composition of a series of transforms.
  1. Each word is a distribution over given semantic dimensions, e.g., “water” covers “liquid”, “clear”, and “oderless”.
  2. A document is generated by sampling words in different semantic dimensions, thus transform the word probabilities to their distributions in documents.
  3. Some semantic dimensions are more important.
- This is the idea behind Singular Vector Decomposition (SVD): $M = U\Sigma V$, where all rows of U or all columns of V are orthorgonal, and Σ a diagnoal matrix of signular values (importances of semantic dimensions).
- The dimension of U is high. Usually we zero out some dimensions in Σ to focus on only the important ones. And thus, the resulting U is no longer orthornormal.
- SVD on word-document co-occurence [2]. Note that the SVD in this tutorial is called compact SVD.
- The co-occurence counts can be further weighted into other quantities, such as point-wise mutual information (Slide 37 of [2]).

Refs:

Embedding via Factorization

- Intuition: Semantically (dis)similar/(un)related words should co-occur in documents (in)frequently.
- A word-document co-occurence matrix can be considered as a composition of a series of transforms.
  1. Each word is a distribution over given semantic dimensions, e.g., “water” covers “liquid”, “clear”, and “oderless”.
  2. A document is generated by sampling words in different semantic dimensions, thus transform the word probabilities to their distributions in documents.
  3. Some semantic dimensions are more important.
- This is the idea behind Singular Vector Decomposition (SVD): $M = U\Sigma V$, where all rows of U or all columns of V are orthorgonal, and Σ a diagnoal matrix of signular values (importances of semantic dimensions).
- The dimension of U is high. Usually we zero out some dimensions in Σ to focus on only the important ones. And thus, the resulting U is no longer orthornormal.
- SVD on word-document co-occurence [2]. Note that the SVD in this tutorial is called compact SVD.
- The co-occurence counts can be further weighted into other quantities, such as point-wise mutual information (Slide 37 of [2]).

Refs:

Embedding via Factorization

- Intuition: Semantically (dis)similar/(un)related words should co-occur in documents (in)frequently.
- A word-document co-occurence matrix can be considered as a composition of a series of transforms.
  1. Each word is a distribution over given semantic dimensions, e.g., "water" covers "liquid", "clear", and "oderless".
  2. A document is generated by sampling words in different semantic dimensions, thus transform the word probabilities to their distributions in documents.
  3. Some semantic dimensions are more important.
- This is the idea behind Singular Vector Decomposition (SVD): $M = U\Sigma V$, where all rows of U or all columns of V are orthorgonal, and Σ a diagnoal matrix of signular values (importances of semantic dimensions).
- The dimension of U is high. Usually we zero out some dimensions in Σ to focus on only the important ones. And thus, the resulting U is no longer orthornormal.
- SVD on word-document co-occurence [2]. Note that the SVD in this tutorial is called compact SVD.
- The co-occurence counts can be further weighted into other quantities, such as point-wise mutual information (Slide 37 of [2]).

Refs:

Embedding via Factorization

- Intuition: Semantically (dis)similar/(un)related words should co-occur in documents (in)frequently.
- A word-document co-occurence matrix can be considered as a composition of a series of transforms.
  1. Each word is a distribution over given semantic dimensions, e.g., “water” covers “liquid”, “clear”, and “oderless”.
  2. A document is generated by sampling words in different semantic dimensions, thus transform the word probabilities to their distributions in documents.
  3. Some semantic dimensions are more important.
- This is the idea behind Singular Vector Decomposition (SVD): $M = U\Sigma V$, where all rows of U or all columns of V are orthorgonal, and Σ a diagnoal matrix of signular values (importances of semantic dimensions).
- The dimension of U is high. Usually we zero out some dimensions in Σ to focus on only the important ones. And thus, the resulting U is no longer orthornormal.
- SVD on word-document co-occurence [2]. Note that the SVD in this tutorial is called compact SVD.
- The co-occurence counts can be further weighted into other quantities, such as point-wise mutual information (Slide 37 of [2]).

Refs:

Problems of factorization-based embedding

- You have to re-create the co-occurence matrix when updating or fine-tuing.
- A large sparse matrix.
- Starting from scratch in computing SVD each time. Cannot reuse previous results, i.e., fine-tuning.

Problems of factorization-based embedding

- You have to re-create the co-occurence matrix when updating or fine-tuing.
- A large sparse matrix.
- Starting from scratch in computing SVD each time. Cannot reuse previous results, i.e., fine-tuning.

Problems of factorization-based embedding

- You have to re-create the co-occurence matrix when updating or fine-tuing.
- A large sparse matrix.
- Starting from scratch in computing SVD each time. Cannot reuse previous results, i.e., fine-tuning.

History

- “A Neural Probabilistic Language Model”, Bengio et al, JMLR, 2003
- “Three New Graphical Models for Statistical Language Modelling”, Mnih & Hinton, ICML 2007
- “A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning”, Collobert and Weston, ICML 2008
- “Neural network based language models for higly inflective languages”, Mikolov et al., ICASSP 2009, separating the training of embeddings and that of the LM/task NN.
- Finally, Word2vec, “Distributed Representations of Words and Phrases and their Compositionality”, Mikolov et al., NIPS 2013
- And, “GloVe: Global Vectors for Word Representation”, Pennnington et al., EMNLP 2014

History

- “A Neural Probabilistic Language Model”, Bengio et al, JMLR, 2003
- “Three New Graphical Models for Statistical Language Modelling”, Mnih & Hinton, ICML 2007
- “A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning”, Collobert and Weston, ICML 2008
- “Neural network based language models for higly inflective languages”, Mikolov et al., ICASSP 2009, separating the training of embeddings and that of the LM/task NN.
- Finally, Word2vec, “Distributed Representations of Words and Phrases and their Compositionality”, Mikolov et al., NIPS 2013
- And, “GloVe: Global Vectors for Word Representation”, Pennnington et al., EMNLP 2014

History

- “A Neural Probabilistic Language Model”, Bengio et al, JMLR, 2003
- “Three New Graphical Models for Statistical Language Modelling”, Mnih & Hinton, ICML 2007
- “A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning”, Collobert and Weston, ICML 2008
- “Neural network based language models for higly inflective languages”, Mikolov et al., ICASSP 2009, separating the training of embeddings and that of the LM/task NN.
- Finally, Word2vec, “Distributed Representations of Words and Phrases and their Compositionality”, Mikolov et al., NIPS 2013
- And, “GloVe: Global Vectors for Word Representation”, Pennnington et al., EMNLP 2014

History

- "A Neural Probabilistic Language Model", Bengio et al, JMLR, 2003
- "Three New Graphical Models for Statistical Language Modelling", Mnih & Hinton, ICML 2007
- "A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning", Collobert and Weston, ICML 2008
- "Neural network based language models for higly inflective languages", Mikolov et al., ICASSP 2009, separating the training of embeddings and that of the LM/task NN.
- Finally, Word2vec, "Distributed Representations of Words and Phrases and their Compositionality", Mikolov et al., NIPS 2013
- And, "GloVe: Global Vectors for Word Representation", Pennnington et al., EMNLP 2014

History

- “A Neural Probabilistic Language Model”, Bengio et al, JMLR, 2003
- “Three New Graphical Models for Statistical Language Modelling”, Mnih & Hinton, ICML 2007
- “A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning”, Collobert and Weston, ICML 2008
- “Neural network based language models for higly inflective languages”, Mikolov et al., ICASSP 2009, separating the training of embeddings and that of the LM/task NN.
- Finally, Word2vec, “Distributed Representations of Words and Phrases and their Compositionality”, Mikolov et al., NIPS 2013
- And, “GloVe: Global Vectors for Word Representation”, Pennnington et al., EMNLP 2014

History

- “A Neural Probabilistic Language Model”, Bengio et al, JMLR, 2003
- “Three New Graphical Models for Statistical Language Modelling”, Mnih & Hinton, ICML 2007
- “A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning”, Collobert and Weston, ICML 2008
- “Neural network based language models for higly inflective languages”, Mikolov et al., ICASSP 2009, separating the training of embeddings and that of the LM/task NN.
- Finally, Word2vec, “Distributed Representations of Words and Phrases and their Compositionality”, Mikolov et al., NIPS 2013
- And, “GloVe: Global Vectors for Word Representation”, Pennnington et al., EMNLP 2014

First neural language model

- Bengio et al., NIPS 2003, A neural probablistic language model
- Joint probability as a function of words:
  $P(\underbrace{w_i}_{target} | \underbrace{w_{i-\tau-1}, \ldots, w_{i-1}}_{context}) = f(\underbrace{w_i}_{target}, \underbrace{w_{i-\tau-1}, \ldots, w_{i-1}}_{context})$ in 2 steps:
  1. each word $w_{j \in [i-\tau-1, i]}$ into a vector $C(w_j) \in \mathbb{R}^n$ thru a look-up table C, which is also a function, and
- $\mathcal{V}$ is the vocabulary. For computing sake, g is simplifed into $g(x, C(w_{i-\tau-1}), \ldots, C(w_{i-1}))$ where x is the index of the target word (correct or fake) in the vocabulary.

First neural language model

- Bengio et al., NIPS 2003, A neural probablistic language model
- Joint probability as a function of words:
  $\underbrace{P(w_i}_{target} | \underbrace{w_{i-\tau-1},\dots,w_{i-1}}_{context}) = f(\underbrace{w_i}_{target}, \underbrace{w_{i-\tau-1},\dots,w_{i-1}}_{context})$ in 2 steps:
  1. each word $w_{j\in[i-\tau-1..i]}$ into a vector $C(w_j)\in\mathbb{R}^D$ thru a look-up table C, which is also a function, and
  2. a function $g(\underbrace{C(w_x)}_{\text{any word } w_x,}, \underbrace{C(w_{i-\tau-1}),\dots,C(w_{i-1})}_{\text{given the same context}})$ such that $w_i = \operatorname{argmax}_x g$,

     e.g.,

     $$
     \begin{array}{rcr}
     g(\text{“}dog\text{''} | \text{“a brown fox jumps over a lazy”}) & > & g(\text{“}penguin\text{''} | \text{“a brown fox jumps over a lazy”}) \\
     g(\text{“}dog\text{''} | \text{“a brown fox jumps over a lazy”}) & > & g(\text{“}wheel\text{''} | \text{“a brown fox jumps over a lazy”}) \\
     g(\text{“}dog\text{''} | \text{“a brown fox jumps over a lazy”}) & > & g(\text{“}homework\text{''} | \text{“a brown fox jumps over a lazy”}) \\
     g(\text{“}dog\text{''} | \text{“a brown fox jumps over a lazy”}) & > & g(\text{“}salary\text{''} | \text{“a brown fox jumps over a lazy”}) \\
     g(\text{“}dog\text{''} | \text{“a brown fox jumps over a lazy”}) & > & \dots g \text{ on any constructed fake/negative examples } \dots
     \end{array}
     $$

- $\mathcal{V}$ is the vocabulary. For computing sake, g is simplifed into $g(x, C(w_{i-\tau-1}),\dots,C(w_{i-1}))$ where x is the index of the target word (correct or fake) in the vocabulary.

First neural language model

- Bengio et al., NIPS 2003, A neural probablistic language model
- Joint probability as a function of words:
  $P(\underbrace{w_i}_{target} \mid \underbrace{w_{i-\tau-1},\ldots,w_{i-1}}_{context}) = f(\underbrace{w_i}_{target}, \underbrace{w_{i-\tau-1},\ldots,w_{i-1}}_{context})$ in 2 steps:
  1. each word $w_{j\in[i-\tau-1..i]}$ into a vector $C(w_j)\in\mathbb{R}^D$ thru a look-up table C, which is also a function, and
  2. a function $g(\underbrace{C(w_x)}_{\text{any word } w_x,}, \underbrace{C(w_{i-\tau-1}),\ldots,C(w_{i-1})}_{\text{given the same context}})$ such that $w_i = \operatorname*{argmax}_x g$,

     e.g.,

$$
\begin{array}{rcr}
g(\text{“}dog\text{''} \mid \text{“a brown fox jumps over a lazy”}) & > & g(\text{“}penguin\text{''} \mid \text{“a brown fox jumps over a lazy”}) \\
g(\text{“}dog\text{''} \mid \text{“a brown fox jumps over a lazy”}) & > & g(\text{“}wheel\text{''} \mid \text{“a brown fox jumps over a lazy”}) \\
g(\text{“}dog\text{''} \mid \text{“a brown fox jumps over a lazy”}) & > & g(\text{“}homework\text{''} \mid \text{“a brown fox jumps over a lazy”}) \\
g(\text{“}dog\text{''} \mid \text{“a brown fox jumps over a lazy”}) & > & g(\text{“}salary\text{''} \mid \text{“a brown fox jumps over a lazy”}) \\
g(\text{“}dog\text{''} \mid \text{“a brown fox jumps over a lazy”}) & > & \ldots g \text{ on any constructed fake/negative examples} \ldots
\end{array}
$$

- $\mathcal{V}$ is the vocabulary. For computing sake, g is simplifed into $g(x, C(w_{i-\tau-1}),\ldots,C(w_{i-1}))$ where x is the index of the target word (correct or fake) in the vocabulary.

First neural language model

- Bengio et al., NIPS 2003, A neural probablistic language model
- Joint probability as a function of words:
  $P(\underbrace{w_i}_{target} | \underbrace{w_{i-\tau-1}, \ldots, w_{i-1}}_{context}) = f(\underbrace{w_i}_{target}, \underbrace{w_{i-\tau-1}, \ldots, w_{i-1}}_{context})$ in 2 steps:
  1. each word $w_{j\in[i-\tau-1..i]}$ into a vector $C(w_j) \in \mathbb{R}^D$ thru a look-up table C, which is also a function, and
  2. a function $g(\underbrace{C(w_x)}_{\text{any word } w_x,}, \underbrace{C(w_{i-\tau-1}), \ldots, C(w_{i-1})}_{\text{given the same context}})$ such that $w_i = \underset{x}{\operatorname{argmax}}\, g$, e.g.,

$$
\begin{array}{rcr}
g(\text{“}dog'' | \text{“a brown fox jumps over a lazy”}) & > & g(\text{“}penguin'' | \text{“a brown fox jumps over a lazy”}) \\
g(\text{“}dog'' | \text{“a brown fox jumps over a lazy”}) & > & g(\text{“}wheel'' | \text{“a brown fox jumps over a lazy”}) \\
g(\text{“}dog'' | \text{“a brown fox jumps over a lazy”}) & > & g(\text{“}homework'' | \text{“a brown fox jumps over a lazy”}) \\
g(\text{“}dog'' | \text{“a brown fox jumps over a lazy”}) & > & g(\text{“}salary'' | \text{“a brown fox jumps over a lazy”}) \\
g(\text{“}dog'' | \text{“a brown fox jumps over a lazy”}) & > & \ldots g \text{ on any constructed fake/negative examples } \ldots
\end{array}
$$

- $\mathcal{V}$ is the vocabulary. For computing sake, g is simplifed into $g(x, C(w_{i-\tau-1}), \ldots, C(w_{i-1}))$ where x is the index of the target word (correct or fake) in the vocabulary.

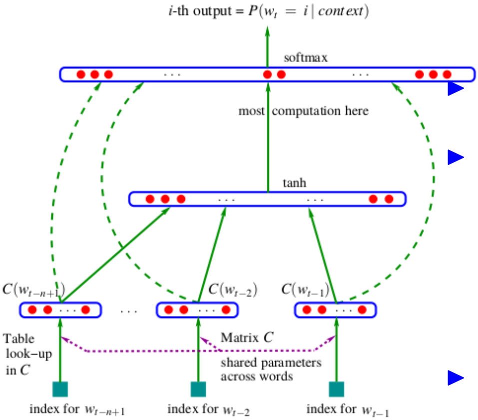
First neural language model

- Bengio et al., NIPS 2003, A neural probablistic language model
- Joint probability as a function of words:
  $P(\underbrace{w_i}_{target} | \underbrace{w_{i-\tau-1}, \dots, w_{i-1}}_{context}) = f(\underbrace{w_i}_{target}, \underbrace{w_{i-\tau-1}, \dots, w_{i-1}}_{context})$ in 2 steps:
  1. each word $w_{j\in[i-\tau-1..i]}$ into a vector $C(w_j) \in \mathbb{R}^D$ thru a look-up table C, which is also a function, and
  2. a function $g(\underbrace{C(w_x)}_{\text{any word } w_x,}, \underbrace{C(w_{i-\tau-1}), \dots, C(w_{i-1})}_{\text{given the same context}})$ such that $w_i = \operatorname*{argmax}_x g$, e.g.,

$$
\begin{array}{rcr}
g(\text{“}dog\text{''} | \text{“a brown fox jumps over a lazy”}) & > & g(\text{“}penguin\text{''} | \text{“a brown fox jumps over a lazy”}) \\
g(\text{“}dog\text{''} | \text{“a brown fox jumps over a lazy”}) & > & g(\text{“}wheel\text{''} | \text{“a brown fox jumps over a lazy”}) \\
g(\text{“}dog\text{''} | \text{“a brown fox jumps over a lazy”}) & > & g(\text{“}homework\text{''} | \text{“a brown fox jumps over a lazy”}) \\
g(\text{“}dog\text{''} | \text{“a brown fox jumps over a lazy”}) & > & g(\text{“}salary\text{''} | \text{“a brown fox jumps over a lazy”}) \\
g(\text{“}dog\text{''} | \text{“a brown fox jumps over a lazy”}) & > & \dots g \text{ on any constructed fake/negative examples} \dots
\end{array}
$$

- $\mathcal{V}$ is the vocabulary. For computing sake, g is simplifed into $g(x, C(w_{i-\tau-1}), \dots, C(w_{i-1}))$ where x is the index of the target word (correct or fake) in the vocabulary.

First neural language model (cont.)



Note the subscripts are different.

- Bengio et al. (2003) used a feedforward network to do it.
- Just 3 layers (see Eq. (1) of Bengio et al. 2003 paper):
  1. input (vectors of context words),
  2. middle,
  3. output (each neuron in the output corresponds to (the vocabulary index of) a word)
- g at “most computation here”.

Cost function of a neural language model

- Maximize the probablities of correct examples, e.g., $g(\text{“}dog''|\text{“a brown fox jumps over a lazy”})$
- Minimize those of all fake exampels, e.g., $g(\text{“}penguin''|\text{“a brown fox jumps over a lazy”})$, $g(\text{“}homework''|\text{“a brown fox jumps over a lazy”})$
- Put them all together: $J = \sum g(\text{correct target}, context) - \sum g(\text{fake target}, context)$ Max it!
- During the training, the g for correct targets grow, while the g for fake targets drop, because the C for them is being updated.
- Bengio et al. 2003 has only the first term, lack of the part for fake targets.
- For computational stablity, usually $\log g$.
- How to generate fake samples? Let all words but “dog” to pair with context “a brown fox jumps over a lazy”?

Cost function of a neural language model

- Maximize the probablities of correct examples, e.g., $g(\text{“}dog''|\text{“a brown fox jumps over a lazy”})$
- Minimize those of all fake exampels, e.g., $g(\text{“}penguin''|\text{“a brown fox jumps over a lazy”})$, $g(\text{“}homework''|\text{“a brown fox jumps over a lazy”})$
- Put them all together: $J = \sum g(\text{correct target}, context) - \sum g(\text{fake target}, context)$ Max it!
- During the training, the g for correct targets grow, while the g for fake targets drop, because the C for them is being updated.
- Bengio et al. 2003 has only the first term, lack of the part for fake targets.
- For computational stablity, usually $\log g$.
- How to generate fake samples? Let all words but “dog” to pair with context “a brown fox jumps over a lazy”?

Cost function of a neural language model

- Maximize the probablities of correct examples, e.g., $g(\text{“}dog'' | \text{“a brown fox jumps over a lazy”})$
- Minimize those of all fake exampels, e.g., $g(\text{“}penguin'' | \text{“a brown fox jumps over a lazy”})$, $g(\text{“}homework'' | \text{“a brown fox jumps over a lazy”})$
- Put them all together: $J = \sum g(\text{correct target}, context) - \sum g(\text{fake target}, context)$ Max it!
- During the training, the g for correct targets grow, while the g for fake targets drop, because the C for them is being updated.
- Bengio et al. 2003 has only the first term, lack of the part for fake targets.
- For computational stablity, usually $\log g$.
- How to generate fake samples? Let all words but “dog” to pair with context “a brown fox jumps over a lazy”?

Cost function of a neural language model

- Maximize the probablities of correct examples, e.g., $g(\text{``}dog''|\text{``a brown fox jumps over a lazy''})$
- Minimize those of all fake exampels, e.g., $g(\text{``}penguin''|\text{``a brown fox jumps over a lazy''})$, $g(\text{``}homework''|\text{``a brown fox jumps over a lazy''})$
- Put them all together: $J = \sum g(\text{correct target}, context) - \sum g(\text{fake target}, context)$ Max it!
- During the training, the g for correct targets grow, while the g for fake targets drop, because the C for them is being updated.
- Bengio et al. 2003 has only the first term, lack of the part for fake targets.
- For computational stablity, usually $\log g$.
- How to generate fake samples? Let all words but "dog" to pair with context "a brown fox jumps over a lazy"?

Cost function of a neural language model

- Maximize the probablities of correct examples, e.g., $g(\text{“}dog''|\text{“a brown fox jumps over a lazy”})$
- Minimize those of all fake exampels, e.g., $g(\text{“}penguin''|\text{“a brown fox jumps over a lazy”})$, $g(\text{“}homework''|\text{“a brown fox jumps over a lazy”})$
- Put them all together: $J = \sum g(\text{correct target}, context) - \sum g(\text{fake target}, context)$ Max it!
- During the training, the g for correct targets grow, while the g for fake targets drop, because the C for them is being updated.
- Bengio et al. 2003 has only the first term, lack of the part for fake targets.
- For computational stablity, usually $\log g$.
- How to generate fake samples? Let all words but “dog” to pair with context “a brown fox jumps over a lazy”?

Cost function of a neural language model

- Maximize the probablities of correct examples, e.g.,
  $g(\text{“}dog''|\text{“a brown fox jumps over a lazy”})$
- Minimize those of all fake exampels, e.g.,
  $g(\text{“}penguin''|\text{“a brown fox jumps over a lazy”})$,
  $g(\text{“}homework''|\text{“a brown fox jumps over a lazy”})$
- Put them all together:
  $J = \sum g(\text{correct target}, context) - \sum g(\text{fake target}, context)$ Max it!
- During the training, the g for correct targets grow, while the g for fake targets drop, because the C for them is being updated.
- Bengio et al. 2003 has only the first term, lack of the part for fake targets.
- For computational stablity, usually $\log g$.
- How to generate fake samples? Let all words but “dog” to pair with context “a brown fox jumps over a lazy”?

Cost function of a neural language model

- Maximize the probablities of correct examples, e.g., $g(\text{“}dog''|\text{“a brown fox jumps over a lazy”})$
- Minimize those of all fake exampels, e.g., $g(\text{“}penguin''|\text{“a brown fox jumps over a lazy”})$, $g(\text{“}homework''|\text{“a brown fox jumps over a lazy”})$
- Put them all together: $J = \sum g(\text{correct target}, context) - \sum g(\text{fake target}, context)$ Max it!
- During the training, the g for correct targets grow, while the g for fake targets drop, because the C for them is being updated.
- Bengio et al. 2003 has only the first term, lack of the part for fake targets.
- For computational stablity, usually $\log g$.
- How to generate fake samples? Let all words but “dog” to pair with context “a brown fox jumps over a lazy”?

Negative sampling

- Too many fake samples. For each correct target (e.g., “a quick brown fox jumps over a lazy dog”), we can have $|\mathcal{V}| - 1$ fake/negative samples (e.g., “a quick brown fox jumps over a lazy cat/penguin/car/code...”).
- A better strategy is to just sample some of them.

Negative sampling

- Too many fake samples. For each correct target (e.g., “a quick brown fox jumps over a lazy dog”), we can have $|\mathcal{V}| - 1$ fake/negative samples (e.g., “a quick brown fox jumps over a lazy cat/penguin/car/code...”).
- A better strategy is to just sample some of them.

Recap: Bengio 2003, first NNLM

- Words are represented into vectors using a look-up table (embedding matrix)
- The look-up table is updated using backpropgragation (thus word embeddings are updated)
- Context words are mapped together into the output layer
- Forward (not to be confused with feedforward): using history words to predict next word

Recap: Bengio 2003, first NNLM

- Words are represented into vectors using a look-up table (embedding matrix)
- The look-up table is updated using backpropgragation (thus word embeddings are updated)
- Context words are mapped together into the output layer
- Forward (not to be confused with feedforward): using history words to predict next word

Recap: Bengio 2003, first NNLM

- Words are represented into vectors using a look-up table (embedding matrix)
- The look-up table is updated using backpropgragation (thus word embeddings are updated)
- Context words are mapped together into the output layer
- Forward (not to be confused with feedforward): using history words to predict next word

Recap: Bengio 2003, first NNLM

- Words are represented into vectors using a look-up table (embedding matrix)
- The look-up table is updated using backpropgragation (thus word embeddings are updated)
- Context words are mapped together into the output layer
- Forward (not to be confused with feedforward): using history words to predict next word

Mnih & Hinton, ICML 2007, bi-linear word-interaction model

- The goal is still to use left history words w_1 to w_{n-1} to predict the n-th word w_n
- $$E(w_n; w_{1:n-1}) = -\left(\sum_{i=1}^{n-1} v_i^T R C_i\right) R^T v_n - bias$$

  where E is not error by energy, v_i is the embedding of the i-th word.
- R is the embedding matrix while C_i's are the weights of the language model.
- Bi-linear: embeddings of context words $v_i^T R$ are linear projected by C_i's, and then the summation of projections dot product with the embedding of the target word $R^T v_n$.
- Purely linear: faster than tanh used in Bengio 2003.

Mnih & Hinton, ICML 2007, bi-linear word-interaction model

- The goal is still to use left history words w_1 to w_{n-1} to predict the n-th word w_n
- $$E(w_n; w_{1:n-1}) = -\left(\sum_{i=1}^{n-1} v_i^T R C_i\right) R^T v_n - bias$$

  where E is not error by energy, v_i is the embedding of the i-th word.
- R is the embedding matrix while C_i's are the weights of the language model.
- Bi-linear: embeddings of context words $v_i^T R$ are linear projected by C_i's, and then the summation of projections dot product with the embedding of the target word $R^T v_n$.
- Purely linear: faster than tanh used in Bengio 2003.

Mnih & Hinton, ICML 2007, bi-linear word-interaction model

- The goal is still to use left history words w_1 to w_{n-1} to predict the n-th word w_n
- $$E(w_n; w_{1:n-1}) = -\left(\sum_{i=1}^{n-1} v_i^T R C_i\right) R^T v_n - bias$$
  where E is not error by energy, v_i is the embedding of the i-th word.
- R is the embedding matrix while C_i's are the weights of the language model.
- Bi-linear: embeddings of context words $v_i^T R$ are linear projected by C_i's, and then the summation of projections dot product with the embedding of the target word $R^T v_n$.
- Purely linear: faster than tanh used in Bengio 2003.

Mnih & Hinton, ICML 2007, bi-linear word-interaction model

- The goal is still to use left history words w_1 to w_{n-1} to predict the n-th word w_n
- $$E(w_n; w_{1:n-1}) = -\left(\sum_{i=1}^{n-1} v_i^T R C_i\right) R^T v_n - bias$$
  where E is not error by energy, v_i is the embedding of the i-th word.
- R is the embedding matrix while C_i's are the weights of the language model.
- Bi-linear: embeddings of context words $v_i^T R$ are linear projected by C_i's, and then the summation of projections dot product with the embedding of the target word $R^T v_n$.
- Purely linear: faster than tanh used in Bengio 2003.

Mnih & Hinton, ICML 2007, bi-linear word-interaction model

- The goal is still to use left history words w_1 to w_{n-1} to predict the n-th word w_n
- $$E(w_n; w_{1:n-1}) = -\left(\sum_{i=1}^{n-1} v_i^T R C_i\right) R^T v_n - bias$$

  where E is not error by energy, v_i is the embedding of the i-th word.
- R is the embedding matrix while C_i's are the weights of the language model.
- Bi-linear: embeddings of context words $v_i^T R$ are linear projected by C_i's, and then the summation of projections dot product with the embedding of the target word $R^T v_n$.
- Purely linear: faster than tanh used in Bengio 2003.

Collobert & Weston, ICML 2008, A Unified Architecture for Natural Language Processing: Deep Neural Network with Multitask Learning

- Lookup-table layer: embedding layer
- Convolutional layers to extract features
- TDNNs to deal with variable lengths of sentences.
- Position encoding: encode the distance between every word in the sentence and the word to be predicted

Collobert & Weston, ICML 2008, A Unified Architecture for Natural Language Processing: Deep Neural Network with Multitask Learning

- Lookup-table layer: embedding layer
- Convolutional layers to extract features
- TDNNs to deal with variable lengths of sentences.
- Position encoding: encode the distance between every word in the sentence and the word to be predicted

Collobert & Weston, ICML 2008, A Unified Architecture for Natural Language Processing: Deep Neural Network with Multitask Learning

- Lookup-table layer: embedding layer
- Convolutional layers to extract features
- TDNNs to deal with variable lengths of sentences.
- Position encoding: encode the distance between every word in the sentence and the word to be predicted

Collobert & Weston, ICML 2008, A Unified Architecture for Natural Language Processing: Deep Neural Network with Multitask Learning

- Lookup-table layer: embedding layer
- Convolutional layers to extract features
- TDNNs to deal with variable lengths of sentences.
- Position encoding: encode the distance between every word in the sentence and the word to be predicted

Separating word embeddings and language models

- All work up to this point tries to learn word embeddings and language models together: one network with both the projection/LUT layer and the language model layer (for the task).
- In Mokolov et al., ICASSP 2009, “Neural network based language models for higly inflective languages”, the authors noticed that separating the two can be better.
- This becomes the foundation of the word2vec.

Separating word embeddings and language models

- All work up to this point tries to learn word embeddings and language models together: one network with both the projection/LUT layer and the language model layer (for the task).
- In Mokolov et al., ICASSP 2009, “Neural network based language models for higly inflective languages”, the authors noticed that separating the two can be better.
- This becomes the foundation of the word2vec.

Separating word embeddings and language models

- All work up to this point tries to learn word embeddings and language models together: one network with both the projection/LUT layer and the language model layer (for the task).
- In Mokolov et al., ICASSP 2009, “Neural network based language models for higly inflective languages”, the authors noticed that separating the two can be better.
- This becomes the foundation of the word2vec.

Mokolov et al., ICASSP 2009, Neural network based language models for higly inflective languages

- Did not refer to Bengio's or Hinton's models at all.
- Words like "embedding" or "look-up table" do not appear in this paper.
- Just brutal force: one-hot encoding as the input, which does the propose of embedding layer (maybe) unintentionally.
- Per the authors, they did so hoping to form a clustering of word representations at the hidden layer, e.g., "see", "saw", "seen" would be mapped to similar vectors.
- Separating the training of word embeddings and language models.
- First train word embeddings by predicting the next word based on the previous word, called bigram network in the paper.
- Then train n-gram LM using the word embeddings just trained.

Mokolov et al., ICASSP 2009, Neural network based language models for higly inflective languages

- Did not refer to Bengio's or Hinton's models at all.
- Words like "embedding" or "look-up table" do not appear in this paper.
- Just brutal force: one-hot encoding as the input, which does the propose of embedding layer (maybe) unintentionally.
- Per the authors, they did so hoping to form a clustering of word representations at the hidden layer, e.g., "see", "saw", "seen" would be mapped to similar vectors.
- Separating the training of word embeddings and language models.
- First train word embeddings by predicting the next word based on the previous word, called bigram network in the paper.
- Then train n-gram LM using the word embeddings just trained.

Mokolov et al., ICASSP 2009, Neural network based language models for higly inflective languages

- Did not refer to Bengio's or Hinton's models at all.
- Words like "embedding" or "look-up table" do not appear in this paper.
- Just brutal force: one-hot encoding as the input, which does the propose of embedding layer (maybe) unintentionally.
- Per the authors, they did so hoping to form a clustering of word representations at the hidden layer, e.g., "see", "saw", "seen" would be mapped to similar vectors.
- Separating the training of word embeddings and language models.
- First train word embeddings by predicting the next word based on the previous word, called bigram network in the paper.
- Then train n-gram LM using the word embeddings just trained.

Mokolov et al., ICASSP 2009, Neural network based language models for higly inflective languages

- Did not refer to Bengio's or Hinton's models at all.
- Words like "embedding" or "look-up table" do not appear in this paper.
- Just brutal force: one-hot encoding as the input, which does the propose of embedding layer (maybe) unintentionally.
- Per the authors, they did so hoping to form a clustering of word representations at the hidden layer, e.g., "see", "saw", "seen" would be mapped to similar vectors.
- Separating the training of word embeddings and language models.
- First train word embeddings by predicting the next word based on the previous word, called bigram network in the paper.
- Then train n-gram LM using the word embeddings just trained.

Mokolov et al., ICASSP 2009, Neural network based language models for higly inflective languages

- Did not refer to Bengio's or Hinton's models at all.
- Words like "embedding" or "look-up table" do not appear in this paper.
- Just brutal force: one-hot encoding as the input, which does the propose of embedding layer (maybe) unintentionally.
- Per the authors, they did so hoping to form a clustering of word representations at the hidden layer, e.g., "see", "saw", "seen" would be mapped to similar vectors.
- Separating the training of word embeddings and language models.
- First train word embeddings by predicting the next word based on the previous word, called bigram network in the paper.
- Then train n-gram LM using the word embeddings just trained.

Mokolov et al., ICASSP 2009, Neural network based language models for higly inflective languages

- Did not refer to Bengio's or Hinton's models at all.
- Words like "embedding" or "look-up table" do not appear in this paper.
- Just brutal force: one-hot encoding as the input, which does the propose of embedding layer (maybe) unintentionally.
- Per the authors, they did so hoping to form a clustering of word representations at the hidden layer, e.g., "see", "saw", "seen" would be mapped to similar vectors.
- Separating the training of word embeddings and language models.
- First train word embeddings by predicting the next word based on the previous word, called bigram network in the paper.
- Then train n-gram LM using the word embeddings just trained.

Mokolov et al., ICASSP 2009, Neural network based language models for higly inflective languages

- Did not refer to Bengio's or Hinton's models at all.
- Words like "embedding" or "look-up table" do not appear in this paper.
- Just brutal force: one-hot encoding as the input, which does the propose of embedding layer (maybe) unintentionally.
- Per the authors, they did so hoping to form a clustering of word representations at the hidden layer, e.g., "see", "saw", "seen" would be mapped to similar vectors.
- Separating the training of word embeddings and language models.
- First train word embeddings by predicting the next word based on the previous word, called bigram network in the paper.
- Then train n-gram LM using the word embeddings just trained.

Some thoughts about research

- Mikolov et al. used extremely simple networks in their ICASSP 2009 paper.
- Nothing fancy like bi-linear interactions.
- Their terminology differs from those appearing in Bengio's or Hinton's.
- Their InterSpeech 2010 paper "Recurrent neural network based language model" is just Elman's network. Again, one-hot encoding as input.
- Hypothetically, if they submitted the papers to ACL/NAACL/EMNLP/COLING, what feedback would they receive? "Trivial model," "nothing new," "lack of comparison with X,Y,Z".
- The beauty of science is to make things simple.

Some thoughts about research

- Mikolov et al. used extremely simple networks in their ICASSP 2009 paper.
- Nothing fancy like bi-linear interactions.
- Their terminology differs from those appearing in Bengio's or Hinton's.
- Their InterSpeech 2010 paper "Recurrent neural network based language model" is just Elman's network. Again, one-hot encoding as input.
- Hypothetically, if they submitted the papers to ACL/NAACL/EMNLP/COLING, what feedback would they receive? "Trivial model," "nothing new," "lack of comparison with X,Y,Z".
- The beauty of science is to make things simple.

Some thoughts about research

- Mikolov et al. used extremely simple networks in their ICASSP 2009 paper.
- Nothing fancy like bi-linear interactions.
- Their terminology differs from those appearing in Bengio's or Hinton's.
- Their InterSpeech 2010 paper "Recurrent neural network based language model" is just Elman's network. Again, one-hot encoding as input.
- Hypothetically, if they submitted the papers to ACL/NAACL/EMNLP/COLING, what feedback would they receive? "Trivial model," "nothing new," "lack of comparison with X,Y,Z".
- The beauty of science is to make things simple.

Some thoughts about research

- Mikolov et al. used extremely simple networks in their ICASSP 2009 paper.
- Nothing fancy like bi-linear interactions.
- Their terminology differs from those appearing in Bengio's or Hinton's.
- Their InterSpeech 2010 paper "Recurrent neural network based language model" is just Elman's network. Again, one-hot encoding as input.
- Hypothetically, if they submitted the papers to ACL/NAACL/EMNLP/COLING, what feedback would they receive? "Trivial model," "nothing new," "lack of comparison with X,Y,Z".
- The beauty of science is to make things simple.

Some thoughts about research

- Mikolov et al. used extremely simple networks in their ICASSP 2009 paper.
- Nothing fancy like bi-linear interactions.
- Their terminology differs from those appearing in Bengio's or Hinton's.
- Their InterSpeech 2010 paper "Recurrent neural network based language model" is just Elman's network. Again, one-hot encoding as input.
- Hypothetically, if they submitted the papers to ACL/NAACL/EMNLP/COLING, what feedback would they receive? "Trivial model," "nothing new," "lack of comparison with X,Y,Z".
- The beauty of science is to make things simple.

Some thoughts about research

- Mikolov et al. used extremely simple networks in their ICASSP 2009 paper.
- Nothing fancy like bi-linear interactions.
- Their terminology differs from those appearing in Bengio's or Hinton's.
- Their InterSpeech 2010 paper "Recurrent neural network based language model" is just Elman's network. Again, one-hot encoding as input.
- Hypothetically, if they submitted the papers to ACL/NAACL/EMNLP/COLING, what feedback would they receive? "Trivial model," "nothing new," "lack of comparison with X,Y,Z".
- The beauty of science is to make things simple.

Word2vec (2013)

- Two models: CBOW (similar to Bengio et al. 2003) and Skip-gram.
- CBOW: use context to predict target word.
- Skip-gram: use target word to predict context words:
- Very simple network architecture: For CBOW, see `https://www.tensorflow.org/tutorials/representation/word2vec` For Skip-gram, see `http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/`
- For math, see `https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-a` `pdf`

Word2vec (2013)

- Two models: CBOW (similar to Bengio et al. 2003) and Skip-gram.
- CBOW: use context to predict target word.
- Skip-gram: use target word to predict context words:
- Very simple network architecture: For CBOW, see https://www.tensorflow.org/tutorials/representation/word2vec For Skip-gram, see http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/
- For math, see https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-a
pdf

Word2vec (2013)

- Two models: CBOW (similar to Bengio et al. 2003) and Skip-gram.
- CBOW: use context to predict target word.
- Skip-gram: use target word to predict context words:
- Very simple network architecture: For CBOW, see `https://www.tensorflow.org/tutorials/representation/word2vec` For Skip-gram, see `http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/`
- For math, see `https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-a` `pdf`

Word2vec (2013)

- Two models: CBOW (similar to Bengio et al. 2003) and Skip-gram.
- CBOW: use context to predict target word.
- Skip-gram: use target word to predict context words:
- Very simple network architecture: For CBOW, see https://www.tensorflow.org/tutorials/representation/word2vec For Skip-gram, see http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/
- For math, see https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-a pdf

Word2vec (2013)

- Two models: CBOW (similar to Bengio et al. 2003) and Skip-gram.
- CBOW: use context to predict target word.
- Skip-gram: use target word to predict context words:
- Very simple network architecture: For CBOW, see `https://www.tensorflow.org/tutorials/representation/word2vec` For Skip-gram, see `http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/`
- For math, see `https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-ar` `pdf`

Why word2vec succeeded

- Separating word embedding learning and language model learning.
- A super simple linear layer – faster training.
- Skip-gram – updating the embedding of only one word each time.
- Bidirectional context: forward and backword

Why word2vec succeeded

- Separating word embedding learning and language model learning.
- A super simple linear layer – faster training.
- Skip-gram – updating the embedding of only one word each time.
- Bidirectional context: forward and backword

Why word2vec succeeded

- Separating word embedding learning and language model learning.
- A super simple linear layer – faster training.
- Skip-gram – updating the embedding of only one word each time.
- Bidirectional context: forward and backword

Why word2vec succeeded

- Separating word embedding learning and language model learning.
- A super simple linear layer – faster training.
- Skip-gram – updating the embedding of only one word each time.
- Bidirectional context: forward and backword

Limitations of word2vec

- It only uses local context information.
- However, some local context words do not contain much semantics of the center word, e.g., “the” in “The cat sat on the mat,” because they have lots co-occurence with other words.
- Solution: remove stop words from the corpus.
- But that’s arbitrary and relies on manual rules.
- Better solution: make use of word-word co-occurence in a global scope.

Limitations of word2vec

- It only uses local context information.
- However, some local context words do not contain much semantics of the center word, e.g., “the” in “The cat sat on the mat,” because they have lots co-occurence with other words.
- Solution: remove stop words from the corpus.
- But that’s arbitrary and relies on manual rules.
- Better solution: make use of word-word co-occurence in a global scope.

Limitations of word2vec

- It only uses local context information.
- However, some local context words do not contain much semantics of the center word, e.g., “the” in “The cat sat on the mat,” because they have lots co-occurence with other words.
- Solution: remove stop words from the corpus.
- But that’s arbitrary and relies on manual rules.
- Better solution: make use of word-word co-occurence in a global scope.

Limitations of word2vec

- It only uses local context information.
- However, some local context words do not contain much semantics of the center word, e.g., “the” in “The cat sat on the mat,” because they have lots co-occurence with other words.
- Solution: remove stop words from the corpus.
- But that’s arbitrary and relies on manual rules.
- Better solution: make use of word-word co-occurence in a global scope.

Limitations of word2vec

- It only uses local context information.
- However, some local context words do not contain much semantics of the center word, e.g., “the” in “The cat sat on the mat,” because they have lots co-occurence with other words.
- Solution: remove stop words from the corpus.
- But that’s arbitrary and relies on manual rules.
- Better solution: make use of word-word co-occurence in a global scope.

GloVe

- We have seen two approaches to word embedding: factorization on co-cooccurence matrixes and neural network-based embedding using local context.
- GloVe combines the benefit of the two.
- Key observation: ratios of co-occurrence probabilities reveal semantics better than co-occurence probabilities.

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

- "water" and "fashion" both have little power to tell the difference between "ice" and "steam": ratios around 1. They are both about water and have little connection with fashion. But "solid" and "gas" can: ratios much larger or smaller than 1.
- $P(k|\text{ice})P(k|\text{steam}) >> 1$ if k is closer to "ice", and $<< 1$ if k is closer to "steam".
- Challenge: how to define a loss function to train the embeddings?
- Starting point: Instead of modeling $g(w_1, w_2)$, let's model $g(w_1, w_2, w_k)$ where w_k can be any word.

GloVe

- We have seen two approaches to word embedding: factorization on co-cooccurence matrixes and neural network-based embedding using local context.
- GloVe combines the benefit of the two.
- Key observation: ratios of co-occurrence probabilities reveal semantics better than co-occurence probabilities.

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

- "water" and "fashion" both have little power to tell the difference between "ice" and "steam": ratios around 1. They are both about water and have little connection with fashion. But "solid" and "gas" can: ratios much larger or smaller than 1.
- $P(k|\text{ice})P(k|\text{steam}) >> 1$ if k is closer to "ice", and $<< 1$ if k is closer to "steam".
- Challenge: how to define a loss function to train the embeddings?
- Starting point: Instead of modeling $g(w_1, w_2)$, let's model $g(w_1, w_2, w_k)$ where w_k can be any word.

GloVe

- We have seen two approaches to word embedding: factorization on co-cooccurence matrixes and neural network-based embedding using local context.
- GloVe combines the benefit of the two.
- Key observation: ratios of co-occurrence probabilities reveal semantics better than co-occurence probabilities.

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

- "water" and "fashion" both have little power to tell the difference between "ice" and "steam": ratios around 1. They are both about water and have little connection with fashion. But "solid" and "gas" can: ratios much larger or smaller than 1.
- $P(k|\text{ice})P(k|\text{steam}) >> 1$ if k is closer to "ice", and $<< 1$ if k is closer to "steam".
- Challenge: how to define a loss function to train the embeddings?
- Starting point: Instead of modeling $g(w_1, w_2)$, let's model $g(w_1, w_2, w_k)$ where w_k can be any word.

GloVe

- We have seen two approaches to word embedding: factorization on co-cooccurence matrixes and neural network-based embedding using local context.
- GloVe combines the benefit of the two.
- Key observation: ratios of co-occurrence probabilities reveal semantics better than co-occurence probabilities.

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

- "water" and "fashion" both have little power to tell the difference between "ice" and "steam": ratios around 1. They are both about water and have little connection with fashion. But "solid" and "gas" can: ratios much larger or smaller than 1.
- $P(k|\text{ice})P(k|\text{steam}) >> 1$ if k is closer to "ice", and $<< 1$ if k is closer to "steam".
- Challenge: how to define a loss function to train the embeddings?
- Starting point: Instead of modeling $g(w_1, w_2)$, let's model $g(w_1, w_2, w_k)$ where w_k can be any word.

GloVe

- We have seen two approaches to word embedding: factorization on co-cooccurence matrixes and neural network-based embedding using local context.
- GloVe combines the benefit of the two.
- Key observation: ratios of co-occurrence probabilities reveal semantics better than co-occurence probabilities.

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

- "water" and "fashion" both have little power to tell the difference between "ice" and "steam": ratios around 1. They are both about water and have little connection with fashion. But "solid" and "gas" can: ratios much larger or smaller than 1.
- $P(k|\text{ice})P(k|\text{steam}) >> 1$ if k is closer to "ice", and $<< 1$ if k is closer to "steam".
- Challenge: how to define a loss function to train the embeddings?
- Starting point: Instead of modeling $g(w_1, w_2)$, let's model $g(w_1, w_2, w_k)$ where w_k can be any word.

GloVe

- We have seen two approaches to word embedding: factorization on co-cooccurence matrixes and neural network-based embedding using local context.
- GloVe combines the benefit of the two.
- Key observation: ratios of co-occurrence probabilities reveal semantics better than co-occurence probabilities.

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

- "water" and "fashion" both have little power to tell the difference between "ice" and "steam": ratios around 1. They are both about water and have little connection with fashion. But "solid" and "gas" can: ratios much larger or smaller than 1.
- $P(k|\text{ice})P(k|\text{steam}) >> 1$ if k is closer to "ice", and $<< 1$ if k is closer to "steam".
- Challenge: how to define a loss function to train the embeddings?
- Starting point: Instead of modeling $g(w_1, w_2)$, let's model $g(w_1, w_2, w_k)$ where w_k can be any word.

GloVe

- We have seen two approaches to word embedding: factorization on co-cooccurence matrixes and neural network-based embedding using local context.
- GloVe combines the benefit of the two.
- Key observation: ratios of co-occurrence probabilities reveal semantics better than co-occurence probabilities.

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

- "water" and "fashion" both have little power to tell the difference between "ice" and "steam": ratios around 1. They are both about water and have little connection with fashion. But "solid" and "gas" can: ratios much larger or smaller than 1.
- $P(k|\text{ice})P(k|\text{steam}) >> 1$ if k is closer to "ice", and $<< 1$ if k is closer to "steam".
- Challenge: how to define a loss function to train the embeddings?
- Starting point: Instead of modeling $g(w_1, w_2)$, let's model $g(w_1, w_2, w_k)$ where w_k can be any word.

GloVe

- We have seen two approaches to word embedding: factorization on co-cooccurence matrixes and neural network-based embedding using local context.
- GloVe combines the benefit of the two.
- Key observation: ratios of co-occurrence probabilities reveal semantics better than co-occurence probabilities.

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

- "water" and "fashion" both have little power to tell the difference between "ice" and "steam": ratios around 1. They are both about water and have little connection with fashion. But "solid" and "gas" can: ratios much larger or smaller than 1.
- $P(k|\text{ice})P(k|\text{steam}) >> 1$ if k is closer to "ice", and $<< 1$ if k is closer to "steam".
- Challenge: how to define a loss function to train the embeddings?
- Starting point: Instead of modeling $g(w_1, w_2)$, let's model $g(w_1, w_2, w_k)$ where w_k can be any word.

The loss of function of GloVe

- GloVe wants to archive a relation F of two words w_i and w_j and a context word w'_k ($'$ indicating the context word, not an operation) such that $F(w_i, w_j, w'_k) = P_{ik}/P_{jk}$ where P_{ik} and P_{jk} are the co-occuring probability of w_i and w'_k and that of w_j and w'_k.
- First, the difference between w_i and w_j is expected to be characterized linearly. The simplest linear difference is vector subtraction. Hence, $F(w_i - w_j, w'_k) = P_{ik}/P_{jk}$. F is overloaded.
- Second, the difference $w_i - w_j$ with respect to the context word w'_k to be linearly characterized as well. The simplest form is dot product. Hence, $F((w_i - w_j)^T w'_k) = P_{ik}/P_{jk}$. F is overloaded again.
- Third, we want to characterize the difference between any two words using their co-occourence, regardless of whether a word is a context word or not. Hence, we want $F((w_i - w_j)^T w'_k) = F(w_i^T w'_k + (-w_j^T w'_k)) = F(w_i^T w'_k) \circ F(-w_j^T w'_k)$ where $\circ$ is an operation to be found. Such an F is known as a group homomorphism in discrete math.

The loss of function of GloVe

- GloVe wants to archive a relation F of two words w_i and w_j and a context word w'_k ($'$ indicating the context word, not an operation) such that $F(w_i, w_j, w'_k) = P_{ik}/P_{jk}$ where P_{ik} and P_{jk} are the co-occuring probability of w_i and w'_k and that of w_j and w'_k.
- First, the difference between w_i and w_j is expected to be characterized linearly. The simplest linear difference is vector subtraction. Hence, $F(w_i - w_j, w'_k) = P_{ik}/P_{jk}$. F is overloaded.
- Second, the difference $w_i - w_j$ with respect to the context word w'_k to be linearly characterized as well. The simplest form is dot product. Hence, $F((w_i - w_j)^T w'_k) = P_{ik}/P_{jk}$. F is overloaded again.
- Third, we want to characterize the difference between any two words using their co-occourence, regardless of whether a word is a context word or not. Hence, we want $F((w_i - w_j)^T w'_k) = F(w_i^T w'_k + (-w_j^T w'_k)) = F(w_i^T w'_k) \circ F(-w_j^T w'_k)$ where $\circ$ is an operation to be found. Such an F is known as a group homomorphism in discrete math.

The loss of function of GloVe

- GloVe wants to archive a relation F of two words w_i and w_j and a context word w'_k ($'$ indicating the context word, not an operation) such that $F(w_i, w_j, w'_k) = P_{ik}/P_{jk}$ where P_{ik} and P_{jk} are the co-occuring probability of w_i and w'_k and that of w_j and w'_k.
- First, the difference between w_i and w_j is expected to be characterized linearly. The simplest linear difference is vector subtraction. Hence, $F(w_i - w_j, w'_k) = P_{ik}/P_{jk}$. F is overloaded.
- Second, the difference $w_i - w_j$ with respect to the context word w'_k to be linearly characterized as well. The simplest form is dot product. Hence, $F((w_i - w_j)^T w'_k) = P_{ik}/P_{jk}$. F is overloaded again.
- Third, we want to characterize the difference between any two words using their co-occourence, regardless of whether a word is a context word or not. Hence, we want $F((w_i - w_j)^T w'_k) = F(w_i^T w'_k + (-w_j^T w'_k)) = F(w_i^T w'_k) \circ F(-w_j^T w'_k)$ where $\circ$ is an operation to be found. Such an F is known as a group homomorphism in discrete math.

The loss of function of GloVe

- GloVe wants to archive a relation F of two words w_i and w_j and a context word w'_k ($'$ indicating the context word, not an operation) such that $F(w_i, w_j, w'_k) = P_{ik}/P_{jk}$ where P_{ik} and P_{jk} are the co-occuring probability of w_i and w'_k and that of w_j and w'_k.
- First, the difference between w_i and w_j is expected to be characterized linearly. The simplest linear difference is vector subtraction. Hence, $F(w_i - w_j, w'_k) = P_{ik}/P_{jk}$. F is overloaded.
- Second, the difference $w_i - w_j$ with respect to the context word w'_k to be linearly characterized as well. The simplest form is dot product. Hence, $F((w_i - w_j)^T w'_k) = P_{ik}/P_{jk}$. F is overloaded again.
- Third, we want to characterize the difference between any two words using their co-occourence, regardless of whether a word is a context word or not. Hence, we want $F((w_i - w_j)^T w'_k) = F(w_i^T w'_k + (-w_j^T w'_k)) = F(w_i^T w'_k) \circ F(-w_j^T w'_k)$ where $\circ$ is an operation to be found. Such an F is known as a group homomorphism in discrete math.

The loss of function of GloVe II

- We can make $\circ$ to be super simple, just multiplication. Thus, $F((w_i - w_j)^T w'_k) = F(w_i^T w'_k) \cdot F(-w_j^T w'_k)$ Then F is a homomorphism between groups $(\mathbb{R}, +)$ and $(\mathbb{R}^+, \times)$.
- Exponential functions are such homomorphism, i.e., $e^{a+b} = e^a \cdot e^b$, thus $F = \exp$.
- Based on the definition, $F((w_i - w_j)^T w'_k) = P_{ik}/P_{jk}$ and $F((w_j - w_i)^T w'_k) = P_{jk}/P_{ik}$ (i and j flipped in the second equation). Their product $F(x)F(-x) = \frac{P_{ik}}{P_{jk}}\frac{P_{jk}}{P_{ik}} = 1$ or $F(-x) = \frac{1}{F(x)}$.
- Using this property, we have $F(w_i - w_j)^T w'_k) = F(w_i^T w'_k) \cdot F(-w_j^T w'_k) = \frac{F(w_i^T w'_k)}{F(w_j^T w'_k)}$.

The loss of function of GloVe II

- We can make $\circ$ to be super simple, just multiplication. Thus, $F((w_i - w_j)^T w'_k) = F(w_i^T w'_k) \cdot F(-w_j^T w'_k)$ Then F is a homomorphism between groups $(\mathbb{R}, +)$ and $(\mathbb{R}^+, \times)$.
- Exponential functions are such homomorphism, i.e., $e^{a+b} = e^a \cdot e^b$, thus $F = \exp$.
- Based on the definition, $F((w_i - w_j)^T w'_k) = P_{ik}/P_{jk}$ and $F((w_j - w_i)^T w'_k) = P_{jk}/P_{ik}$ (i and j flipped in the second equation). Their product $F(x)F(-x) = \frac{P_{ik}}{P_{jk}} \frac{P_{jk}}{P_{ik}} = 1$ or $F(-x) = \frac{1}{F(x)}$.
- Using this property, we have $F(w_i - w_j)^T w'_k) = F(w_i^T w'_k) \cdot F(-w_j^T w'_k) = \frac{F(w_i^T w'_k)}{F(w_j^T w'_k)}$.

The loss of function of GloVe II

- We can make $\circ$ to be super simple, just multiplication. Thus, $F((w_i - w_j)^T w'_k) = F(w_i^T w'_k) \cdot F(-w_j^T w'_k)$ Then F is a homomorphism between groups $(\mathbb{R}, +)$ and $(\mathbb{R}^+, \times)$.
- Exponential functions are such homomorphism, i.e., $e^{a+b} = e^a \cdot e^b$, thus $F = \exp$.
- Based on the definition, $F((w_i - w_j)^T w'_k) = P_{ik}/P_{jk}$ and $F((w_j - w_i)^T w'_k) = P_{jk}/P_{ik}$ (i and j flipped in the second equation). Their product $F(x)F(-x) = \frac{P_{ik}}{P_{jk}} \frac{P_{jk}}{P_{ik}} = 1$ or $F(-x) = \frac{1}{F(x)}$.
- Using this property, we have $F(w_i - w_j)^T w'_k) = F(w_i^T w'_k) \cdot F(-w_j^T w'_k) = \frac{F(w_i^T w'_k)}{F(w_j^T w'_k)}$.

The loss of function of GloVe II

- We can make $\circ$ to be super simple, just multiplication. Thus, $F((w_i - w_j)^T w'_k) = F(w_i^T w'_k) \cdot F(-w_j^T w'_k)$ Then F is a homomorphism between groups $(\mathbb{R}, +)$ and $(\mathbb{R}^+, \times)$.
- Exponential functions are such homomorphism, i.e., $e^{a+b} = e^a \cdot e^b$, thus $F = \exp$.
- Based on the definition, $F((w_i - w_j)^T w'_k) = P_{ik}/P_{jk}$ and $F((w_j - w_i)^T w'_k) = P_{jk}/P_{ik}$ (i and j flipped in the second equation). Their product $F(x)F(-x) = \frac{P_{ik}}{P_{jk}}\frac{P_{jk}}{P_{ik}} = 1$ or $F(-x) = \frac{1}{F(x)}$.
- Using this property, we have $F(w_i - w_j)^T w'_k) = F(w_i^T w'_k) \cdot F(-w_j^T w'_k) = \frac{F(w_i^T w'_k)}{F(w_j^T w'_k)}$.

The loss of function of GloVe III

- Recalling that $F((w_i - w_j)^T w'_k) = P_{ik}/P_{jk}$, we have $F(w_i^T w'_k) = \exp(w_i^T w'_k) = P_{ik} = X_{ik}/Xi$ where X_{ik} is the global cooccurence of w_i and w_k and X_i is the global occurence of w_i.
- Log on both sides, we have $w_i^T w'_k = \log X_{ik} - \log X_i$ where $\log X_i$ has nothing to do with w'_k and hence is absorbed into a bias: $w_i^T w'_k = \log X_i k + b_i$.
- Last tuning: the authors want the formula above to be symmetric to both w'_k and w_i, thus the bias term is not only there for non-context word. Hence they add a bias for the context word: $w_i^T w'_k = \log X_{ik} + b_i + b'_k$.
- Then the loss function is $(\log X_{ik} - w_i^T w'_k - b_i - b'_k)^2$ and the goal is to minimize it.
- Not really. One more thing.

The loss of function of GloVe III

- Recalling that $F((w_i - w_j)^T w'_k) = P_{ik}/P_{jk}$, we have $F(w_i^T w'_k) = \exp(w_i^T w'_k) = P_{ik} = X_{ik}/Xi$ where X_{ik} is the global cooccurence of w_i and w_k and X_i is the global occurence of w_i.
- Log on both sides, we have $w_i^T w'_k = \log X_{ik} - \log X_i$ where $\log X_i$ has nothing to do with w'_k and hence is absorbed into a bias: $w_i^T w'_k = \log X_i k + b_i$.
- Last tuning: the authors want the formula above to be symmetric to both w'_k and w_i, thus the bias term is not only there for non-context word. Hence they add a bias for the context word: $w_i^T w'_k = \log X_{ik} + b_i + b'_k$.
- Then the loss function is $(\log X_{ik} - w_i^T w'_k - b_i - b'_k)^2$ and the goal is to minimize it.
- Not really. One more thing.

The loss of function of GloVe III

- Recalling that $F((w_i - w_j)^T w'_k) = P_{ik}/P_{jk}$, we have $F(w_i^T w'_k) = \exp(w_i^T w'_k) = P_{ik} = X_{ik}/Xi$ where X_{ik} is the global cooccurence of w_i and w_k and X_i is the global occurence of w_i.
- Log on both sides, we have $w_i^T w'_k = \log X_{ik} - \log X_i$ where $\log X_i$ has nothing to do with w'_k and hence is absorbed into a bias: $w_i^T w'_k = \log X_i k + b_i$.
- Last tuning: the authors want the formula above to be symmetric to both w'_k and w_i, thus the bias term is not only there for non-context word. Hence they add a bias for the context word: $w_i^T w'_k = \log X_{ik} + b_i + b'_k$.
- Then the loss function is $(\log X_{ik} - w_i^T w'_k - b_i - b'_k)^2$ and the goal is to minimize it.
- Not really. One more thing.

The loss of function of GloVe III

- Recalling that $F((w_i - w_j)^T w'_k) = P_{ik}/P_{jk}$, we have $F(w_i^T w'_k) = \exp(w_i^T w'_k) = P_{ik} = X_{ik}/Xi$ where X_{ik} is the global cooccurence of w_i and w_k and X_i is the global occurence of w_i.
- Log on both sides, we have $w_i^T w'_k = \log X_{ik} - \log X_i$ where $\log X_i$ has nothing to do with w'_k and hence is absorbed into a bias: $w_i^T w'_k = \log X_i k + b_i$.
- Last tuning: the authors want the formula above to be symmetric to both w'_k and w_i, thus the bias term is not only there for non-context word. Hence they add a bias for the context word: $w_i^T w'_k = \log X_{ik} + b_i + b'_k$.
- Then the loss function is $(\log X_{ik} - w_i^T w'_k - b_i - b'_k)^2$ and the goal is to minimize it.
- Not really. One more thing.

The loss of function of GloVe III

- Recalling that $F((w_i - w_j)^T w'_k) = P_{ik}/P_{jk}$, we have $F(w_i^T w'_k) = \exp(w_i^T w'_k) = P_{ik} = X_{ik}/Xi$ where X_{ik} is the global cooccurence of w_i and w_k and X_i is the global occurence of w_i.
- Log on both sides, we have $w_i^T w'_k = \log X_{ik} - \log X_i$ where $\log X_i$ has nothing to do with w'_k and hence is absorbed into a bias: $w_i^T w'_k = \log X_i k + b_i$.
- Last tuning: the authors want the formula above to be symmetric to both w'_k and w_i, thus the bias term is not only there for non-context word. Hence they add a bias for the context word: $w_i^T w'_k = \log X_{ik} + b_i + b'_k$.
- Then the loss function is $(\log X_{ik} - w_i^T w'_k - b_i - b'_k)^2$ and the goal is to minimize it.
- Not really. One more thing.

The loss of function of GloVe IV

- Word pairs have different frequencies in a corpus. So they should have different contributions to the loss function.
- $J = \sum_{i,j=1}^{|\mathcal{V}|} W(X_{i,j})(\log X_{ij} - w_i^T w_j - b_i - b_j)^2$
- Two goals of the weight function W: $W(X_{i,j})$ cannot be too large if $X_{i,j}$ is small whereas it cannot be too large also for frequenty w_i and w_j pairs.
- An implementation:

$$W(X_{i,j}) = \begin{cases} (X_{i,j}/X_{max})^{\alpha} & if \quad X_{i,j} < X_{max} \\ 1 & o/w \end{cases}$$

  where X_{max} is the maximal cooccurence of two words in the corpus.
- Empirical study finds that $\alpha = 3/4$ is a good number.
- See also: `http://mlexplained.com/2018/04/29/paper-dissected-glove-global-vectors-for-word-representa` and `http://text2vec.org/glove.html`

The loss of function of GloVe IV

- Word pairs have different frequencies in a corpus. So they should have different contributions to the loss function.
- $J = \sum_{i,j=1}^{|\mathcal{V}|} W(X_{i,j})(\log X_{ij} - w_i^T w_j - b_i - b_j)^2$
- Two goals of the weight function W: $W(X_{i,j})$ cannot be too large if $X_{i,j}$ is small whereas it cannot be too large also for frequenty w_i and w_j pairs.
- An implementation:

$$W(X_{i,j}) = \begin{cases} (X_{i,j}/X_{max})^{\alpha} & if \quad X_{i,j} < X_{max} \\ 1 & o/w \end{cases}$$

  where X_{max} is the maximal cooccurence of two words in the corpus.
- Empirical study finds that $\alpha = 3/4$ is a good number.
- See also: `http://mlexplained.com/2018/04/29/paper-dissected-glove-global-vectors-for-word-representa` and `http://text2vec.org/glove.html`

The loss of function of GloVe IV

- Word pairs have different frequencies in a corpus. So they should have different contributions to the loss function.
- $J = \sum_{i,j=1}^{|\mathcal{V}|} W(X_{i,j})(\log X_{ij} - w_i^T w_j - b_i - b_j)^2$
- Two goals of the weight function W: $W(X_{i,j})$ cannot be too large if $X_{i,j}$ is small whereas it cannot be too large also for frequenty w_i and w_j pairs.
- An implementation:

$$W(X_{i,j}) = \begin{cases} (X_{i,j}/X_{max})^{\alpha} & if \quad X_{i,j} < X_{max} \\ 1 & o/w \end{cases}$$

  where X_{max} is the maximal cooccurence of two words in the corpus.
- Empirical study finds that $\alpha = 3/4$ is a good number.
- See also: `http://mlexplained.com/2018/04/29/paper-dissected-glove-global-vectors-for-word-representa` and `http://text2vec.org/glove.html`

The loss of function of GloVe IV

- Word pairs have different frequencies in a corpus. So they should have different contributions to the loss function.
- $J = \sum_{i,j=1}^{|\mathcal{V}|} W(X_{i,j})(\log X_{ij} - w_i^T w_j - b_i - b_j)^2$
- Two goals of the weight function W: $W(X_{i,j})$ cannot be too large if $X_{i,j}$ is small whereas it cannot be too large also for frequenty w_i and w_j pairs.
- An implementation:

$$W(X_{i,j}) = \begin{cases} (X_{i,j}/X_{max})^{\alpha} & if \;\; X_{i,j} < X_{max} \\ 1 & o/w \end{cases}$$

  where X_{max} is the maximal cooccurence of two words in the corpus.
- Empirical study finds that $\alpha = 3/4$ is a good number.
- See also: http://mlexplained.com/2018/04/29/paper-dissected-glove-global-vectors-for-word-representa and http://text2vec.org/glove.html

The loss of function of GloVe IV

- Word pairs have different frequencies in a corpus. So they should have different contributions to the loss function.
- $J = \sum_{i,j=1}^{|\mathcal{V}|} W(X_{i,j})(\log X_{ij} - w_i^T w_j - b_i - b_j)^2$
- Two goals of the weight function W: $W(X_{i,j})$ cannot be too large if $X_{i,j}$ is small whereas it cannot be too large also for frequenty w_i and w_j pairs.
- An implementation:

$$W(X_{i,j}) = \begin{cases} (X_{i,j}/X_{max})^{\alpha} & if \;\; X_{i,j} < X_{max} \\ 1 & o/w \end{cases}$$

  where X_{max} is the maximal cooccurence of two words in the corpus.
- Empirical study finds that $\alpha = 3/4$ is a good number.
- See also: http://mlexplained.com/2018/04/29/paper-dissected-glove-global-vectors-for-word-representa and http://text2vec.org/glove.html

The loss of function of GloVe IV

- Word pairs have different frequencies in a corpus. So they should have different contributions to the loss function.
- $J = \sum_{i,j=1}^{|\mathcal{V}|} W(X_{i,j})(\log X_{ij} - w_i^T w_j - b_i - b_j)^2$
- Two goals of the weight function W: $W(X_{i,j})$ cannot be too large if $X_{i,j}$ is small whereas it cannot be too large also for frequenty w_i and w_j pairs.
- An implementation:

$$W(X_{i,j}) = \begin{cases} (X_{i,j}/X_{max})^\alpha & if \quad X_{i,j} < X_{max} \\ 1 & o/w \end{cases}$$

  where X_{max} is the maximal cooccurence of two words in the corpus.
- Empirical study finds that $\alpha = 3/4$ is a good number.
- See also: `http://mlexplained.com/2018/04/29/paper-dissected-glove-global-vectors-for-word-representa` and `http://text2vec.org/glove.html`

Sentence embedding

- DAN
- Skip-thought
- Transformer