Learning Distributed Document Representations for Multi-Label Document Categorization

Nitish Gupta

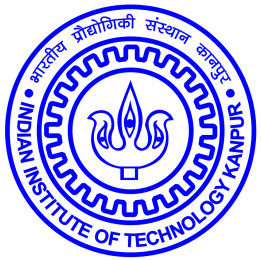
B.Tech - M.Tech Dual Degree

Thesis Defense

Electrical Engineering

IIT Kanpur

May 16, 2015



Outline

Intoduction to Multi-Label Document Categorization

- Text Documents usually belong to more than one conceptual class. For E.g. an article on Music Piracy

Intoduction to Multi-Label Document Categorization

- Text Documents usually belong to more than one conceptual class.
  For E.g. an article on Music Piracy
- Task of assigning documents to one or more predefined categories is called *Multi-Label Document Categorization*

Intoduction to Multi-Label Document Categorization

- Text Documents usually belong to more than one conceptual class.
  For E.g. an article on Music Piracy
- Task of assigning documents to one or more predefined categories is called *Multi-Label Document Categorization*
- Wide range real-world applications :
  - Web-page tagging
  - Medical Patient Record Management
  - Wikipedia Article Management
  - Document Recommendation etc.

Intoduction to Multi-Label Document Categorization

- Text Documents usually belong to more than one conceptual class.
  For E.g. an article on Music Piracy
- Task of assigning documents to one or more predefined categories is called *Multi-Label Document Categorization*
- Wide range real-world applications :
  - Web-page tagging
  - Medical Patient Record Management
  - Wikipedia Article Management
  - Document Recommendation etc.
- Multi-label classification belongs to a general class of supervised learning algorithms where :

Intoduction to Multi-Label Document Categorization

- Text Documents usually belong to more than one conceptual class.
  For E.g. an article on Music Piracy
- Task of assigning documents to one or more predefined categories is called *Multi-Label Document Categorization*
- Wide range real-world applications :
  - Web-page tagging
  - Medical Patient Record Management
  - Wikipedia Article Management
  - Document Recommendation etc.
- Multi-label classification belongs to a general class of supervised learning algorithms where :
  - Training instances in the form of document-category pairs are used to learn a classifier $\mathcal{H}$

Intoduction to Multi-Label Document Categorization

- Text Documents usually belong to more than one conceptual class.
  For E.g. an article on Music Piracy
- Task of assigning documents to one or more predefined categories is called *Multi-Label Document Categorization*
- Wide range real-world applications :
  - Web-page tagging
  - Medical Patient Record Management
  - Wikipedia Article Management
  - Document Recommendation etc.
- Multi-label classification belongs to a general class of supervised learning algorithms where :
  - Training instances in the form of document-category pairs are used to learn a classifier $\mathcal{H}$
  - Learned classifier $\mathcal{H}$ is used to assign categories to new test documents

Introduction to Multi-Label Document Categorization

Given,

- A set of documents $D = \{d_1, \ldots, d_{|D|}\}$

Introduction to Multi-Label Document Categorization

Given,

- A set of documents $D = \{d_1, \ldots, d_{|D|}\}$
- A set of categories $C = \{c_1, \ldots, c_{|C|}\}$

Introduction to Multi-Label Document Categorization

Given,

- A set of documents $D = \{d_1, \ldots, d_{|D|}\}$
- A set of categories $C = \{c_1, \ldots, c_{|C|}\}$
- Training data for n ($n < |D|$) documents, $\mathcal{T} = \{l_{d_1}, \ldots, l_{d_n}\}$

Introduction to Multi-Label Document Categorization

Given,

- A set of documents $D = \{d_1, \ldots, d_{|D|}\}$
- A set of categories $C = \{c_1, \ldots, c_{|C|}\}$
- Training data for n ($n < |D|$) documents, $\mathcal{T} = \{l_{d_1}, \ldots, l_{d_n}\}$
  Each label vector $l_{d_i} \in \{0, 1\}^{|C|}$ denotes relevance of categories to the document d_i

Introduction to Multi-Label Document Categorization

Given,

- A set of documents $D = \{d_1, \ldots, d_{|D|}\}$
- A set of categories $C = \{c_1, \ldots, c_{|C|}\}$
- Training data for n ($n < |D|$) documents, $\mathcal{T} = \{l_{d_1}, \ldots, l_{d_n}\}$
  Each label vector $l_{d_i} \in \{0, 1\}^{|C|}$ denotes relevance of categories to the document d_i

Example :

Documents	Sports	Music	Arts	Technology	Literature	Politics
d_1	0	0	1	0	1	0
d_2	0	1	1	0	0	1
d_3	1	0	0	1	0	1
d_4	×	×	×	×	×	×
d_5	×	×	×	×	×	×

Introduction to Multi-Label Document Categorization

Given,

- A set of documents $D = \{d_1, \ldots, d_{|D|}\}$
- A set of categories $C = \{c_1, \ldots, c_{|C|}\}$
- Training data for n ($n < |D|$) documents, $\mathcal{T} = \{l_{d_1}, \ldots, l_{d_n}\}$
  Each label vector $l_{d_i} \in \{0, 1\}^{|C|}$ denotes relevance of categories to the document d_i

Example :

Documents	Sports	Music	Arts	Technology	Literature	Politics
d_1	0	0	1	0	1	0
d_2	0	1	1	0	0	1
d_3	1	0	0	1	0	1
d_4	×	×	×	×	×	×
d_5	×	×	×	×	×	×

Using $\mathcal{T}$, D and C the learning algorithm learns a multi-label classifier $\mathcal{H}$ to estimate category label vectors, l_{d_j} ($j > n$) for the test documents.

Introduction to Multi-Label Document Categorization

Document Categorization task has the following two components :

Introduction to Multi-Label Document Categorization

Document Categorization task has the following two components :

1. *Learning Document Representations* : Representing text documents using numerical vectors that are inputs to the multi-label classifier $\mathcal{H}$

Introduction to Multi-Label Document Categorization

Document Categorization task has the following two components :

1. *Learning Document Representations* : Representing text documents using numerical vectors that are inputs to the multi-label classifier $\mathcal{H}$
   - Each document $d_i \in D$ is represented using a vector $v_{d_i} \in \mathbb{R}^k$

Introduction to Multi-Label Document Categorization

Document Categorization task has the following two components :

1. *Learning Document Representations* : Representing text documents using numerical vectors that are inputs to the multi-label classifier $\mathcal{H}$
   - Each document $d_i \in D$ is represented using a vector $v_{d_i} \in \mathbb{R}^k$
   - Vectors (v_{d_i}) should encode the semantic content of the documents

Introduction to Multi-Label Document Categorization

Document Categorization task has the following two components :

1. *Learning Document Representations* : Representing text documents using numerical vectors that are inputs to the multi-label classifier $\mathcal{H}$
   - Each document $d_i \in D$ is represented using a vector $v_{d_i} \in \mathbb{R}^k$
   - Vectors (v_{d_i}) should encode the semantic content of the documents
   - Encoding documents in a k-dimensional space using such represenation is called the *Vector Space Model*

Introduction to Multi-Label Document Categorization

Document Categorization task has the following two components :

1. *Learning Document Representations* : Representing text documents using numerical vectors that are inputs to the multi-label classifier $\mathcal{H}$
   - Each document $d_i \in D$ is represented using a vector $v_{d_i} \in \mathbb{R}^k$
   - Vectors (v_{d_i}) should encode the semantic content of the documents
   - Encoding documents in a k-dimensional space using such represenation is called the *Vector Space Model*
   - The complete document set D can be represented by a document representation matrix $\mathrm{D} \in \mathbb{R}^{k \times |D|}$

Introduction to Multi-Label Document Categorization

Document Categorization task has the following two components :

1. *Learning Document Representations* : Representing text documents using numerical vectors that are inputs to the multi-label classifier $\mathcal{H}$
   - Each document $d_i \in D$ is represented using a vector $v_{d_i} \in \mathbb{R}^k$
   - Vectors (v_{d_i}) should encode the semantic content of the documents
   - Encoding documents in a k-dimensional space using such represenation is called the *Vector Space Model*
   - The complete document set D can be represented by a document representation matrix $\mathrm{D} \in \mathbb{R}^{k \times |D|}$

In this thesis, we focus on learning efficient document representations, D

Introduction to Multi-Label Document Categorization

Document Categorization task has the following two components :

1. *Learning Document Representations* : Representing text documents using numerical vectors that are inputs to the multi-label classifier $\mathcal{H}$
   - Each document $d_i \in D$ is represented using a vector $v_{d_i} \in \mathbb{R}^k$
   - Vectors (v_{d_i}) should encode the semantic content of the documents
   - Encoding documents in a k-dimensional space using such represenation is called the *Vector Space Model*
   - The complete document set D can be represented by a document representation matrix $\mathrm{D} \in \mathbb{R}^{k \times |D|}$

In this thesis, we focus on learning efficient document representations, D

2. *Learning Algorithm* : Algorithm to learn the multi-label classifier $\mathcal{H}$

Background on Learning Algorithms

Background on Learning Algorithms

1. *Learning Multiple Binary Classifiers* :

Background on Learning Algorithms

1. *Learning Multiple Binary Classifiers* :
   Algorithms that treat each category assignment independently and learn multiple binary classifiers, one for each category, to make the category assignments

Background on Learning Algorithms

1. *Learning Multiple Binary Classifiers* :
   Algorithms that treat each category assignment independently and learn multiple binary classifiers, one for each category, to make the category assignments
   - Logistic Regression

Background on Learning Algorithms

1. *Learning Multiple Binary Classifiers* :
   Algorithms that treat each category assignment independently and learn multiple binary classifiers, one for each category, to make the category assignments
   - Logistic Regression
   - Support Vector Machines (SVM)

Background on Learning Algorithms

1. *Learning Multiple Binary Classifiers* :
   Algorithms that treat each category assignment independently and learn multiple binary classifiers, one for each category, to make the category assignments
   - Logistic Regression
   - Support Vector Machines (SVM)
   - Neural Networks

Background on Learning Algorithms

1. *Learning Multiple Binary Classifiers* :
   Algorithms that treat each category assignment independently and learn multiple binary classifiers, one for each category, to make the category assignments
   - Logistic Regression
   - Support Vector Machines (SVM)
   - Neural Networks
   - Naive Bayes

Background on Learning Algorithms

1. *Learning Multiple Binary Classifiers* :
   Algorithms that treat each category assignment independently and learn multiple binary classifiers, one for each category, to make the category assignments
   - Logistic Regression
   - Support Vector Machines (SVM)
   - Neural Networks
   - Naive Bayes
2. *Learning Single Joint Classifier* :

Background on Learning Algorithms

1. *Learning Multiple Binary Classifiers* :
   Algorithms that treat each category assignment independently and learn multiple binary classifiers, one for each category, to make the category assignments
   - Logistic Regression
   - Support Vector Machines (SVM)
   - Neural Networks
   - Naive Bayes
2. *Learning Single Joint Classifier* :
   Algorithms that jointly assign all the categories to a document d_i, i.e. estimate the complete label vector l_{d_i} using a single classifier

Background on Learning Algorithms

1. *Learning Multiple Binary Classifiers* :
   Algorithms that treat each category assignment independently and learn multiple binary classifiers, one for each category, to make the category assignments
   - Logistic Regression
   - Support Vector Machines (SVM)
   - Neural Networks
   - Naive Bayes
2. *Learning Single Joint Classifier* :
   Algorithms that jointly assign all the categories to a document d_i, i.e. estimate the complete label vector l_{d_i} using a single classifier
   - k-Nearest Neighbor (k-NN)

Background on Learning Algorithms

1. *Learning Multiple Binary Classifiers* :
   Algorithms that treat each category assignment independently and learn multiple binary classifiers, one for each category, to make the category assignments
   - Logistic Regression
   - Support Vector Machines (SVM)
   - Neural Networks
   - Naive Bayes
2. *Learning Single Joint Classifier* :
   Algorithms that jointly assign all the categories to a document d_i, i.e. estimate the complete label vector l_{d_i} using a single classifier
   - k-Nearest Neighbor (k-NN)
   - Linear Least Square Fit

Background on Learning Algorithms

1. *Learning Multiple Binary Classifiers* :
   Algorithms that treat each category assignment independently and learn multiple binary classifiers, one for each category, to make the category assignments
   - Logistic Regression
   - Support Vector Machines (SVM)
   - Neural Networks
   - Naive Bayes
2. *Learning Single Joint Classifier* :
   Algorithms that jointly assign all the categories to a document d_i, i.e. estimate the complete label vector l_{d_i} using a single classifier
   - k-Nearest Neighbor (k-NN)
   - Linear Least Square Fit
   - Decision Trees

Background on Learning Algorithms

1. *Learning Multiple Binary Classifiers* :
   Algorithms that treat each category assignment independently and learn multiple binary classifiers, one for each category, to make the category assignments
   - Logistic Regression
   - Support Vector Machines (SVM)
   - Neural Networks
   - Naive Bayes
2. *Learning Single Joint Classifier* :
   Algorithms that jointly assign all the categories to a document d_i, i.e. estimate the complete label vector l_{d_i} using a single classifier
   - k-Nearest Neighbor (k-NN)
   - Linear Least Square Fit
   - Decision Trees
   - Generative Probabilistic Models

Background on Text Representation

Bag of Words Model

Background on Text Representation

Bag of Words Model

- Document d_i represented by $v_{d_i} \in \mathbb{R}^{|V|}$

Background on Text Representation

Bag of Words Model

- Document d_i represented by $v_{d_i} \in \mathbb{R}^{|V|}$
- Each element in v_{d_i} denotes presence/absence of each word

Background on Text Representation

Bag of Words Model

- Document d_i represented by $v_{d_i} \in \mathbb{R}^{|V|}$
- Each element in v_{d_i} denotes presence/absence of each word
- Weighing techniques employed to give importance to important terms

Background on Text Representation

Bag of Words Model

- Document d_i represented by $v_{d_i} \in \mathbb{R}^{|V|}$
- Each element in v_{d_i} denotes presence/absence of each word
- Weighing techniques employed to give importance to important terms
  - Term Frequency (tf)
  - Inverse Document Frequency (idf)
  - Term Frequency - Inverse Document Frequency (tf-idf) : $tf \times idf$

Background on Text Representation

Bag of Words Model

- Document d_i represented by $v_{d_i} \in \mathbb{R}^{|V|}$
- Each element in v_{d_i} denotes presence/absence of each word
- Weighing techniques employed to give importance to important terms
  - Term Frequency (tf)
  - Inverse Document Frequency (idf)
  - Term Frequency - Inverse Document Frequency (tf-idf) : $tf \times idf$

Drawbacks of the Bag-of-Words model

Background on Text Representation

Bag of Words Model

- Document d_i represented by $v_{d_i} \in \mathbb{R}^{|V|}$
- Each element in v_{d_i} denotes presence/absence of each word
- Weighing techniques employed to give importance to important terms
  - Term Frequency (tf)
  - Inverse Document Frequency (idf)
  - Term Frequency - Inverse Document Frequency (tf-idf) : $tf \times idf$

Drawbacks of the Bag-of-Words model

- High-dimensionality

Background on Text Representation

Bag of Words Model

- Document d_i represented by $v_{d_i} \in \mathbb{R}^{|V|}$
- Each element in v_{d_i} denotes presence/absence of each word
- Weighing techniques employed to give importance to important terms
  - Term Frequency (tf)
  - Inverse Document Frequency (idf)
  - Term Frequency - Inverse Document Frequency (tf-idf) : $tf \times idf$

Drawbacks of the Bag-of-Words model

- High-dimensionality
- Sparsity

Background on Text Representation

Bag of Words Model

- Document d_i represented by $v_{d_i} \in \mathbb{R}^{|V|}$
- Each element in v_{d_i} denotes presence/absence of each word
- Weighing techniques employed to give importance to important terms
  - Term Frequency (tf)
  - Inverse Document Frequency (idf)
  - Term Frequency - Inverse Document Frequency (tf-idf) : $tf \times idf$

Drawbacks of the Bag-of-Words model

- High-dimensionality
- Sparsity
- Inability to encode word contexts

Background on Text Representation

Bag of Words Model

- Document d_i represented by $v_{d_i} \in \mathbb{R}^{|V|}$
- Each element in v_{d_i} denotes presence/absence of each word
- Weighing techniques employed to give importance to important terms
  - Term Frequency (tf)
  - Inverse Document Frequency (idf)
  - Term Frequency - Inverse Document Frequency (tf-idf) : $tf \times idf$

Drawbacks of the Bag-of-Words model

- High-dimensionality
- Sparsity
- Inability to encode word contexts
- Ignores word order

Background on Feature Selection / Dimensionality Reduction

Techniques to deal with sparsity and high-dimensionality in BOW

Background on Feature Selection / Dimensionality Reduction

Techniques to deal with sparsity and high-dimensionality in BOW

- Information Gain

$$G(t) = -\sum_{i=1}^{|C|} P(c_i) \log P(c_i) + P(t) \sum_{i=1}^{|C|} P(c_i|t) \log P(c_i|t) + P(\sim t) \sum_{i=1}^{|C|} P(c_i| \sim t) \log P(c_i| \sim t) \quad (1)$$

Background on Feature Selection / Dimensionality Reduction

Techniques to deal with sparsity and high-dimensionality in BOW

- Information Gain

$$G(t) = -\sum_{i=1}^{|C|} P(c_i) \log P(c_i) + P(t) \sum_{i=1}^{|C|} P(c_i|t) \log P(c_i|t) + P(\sim t) \sum_{i=1}^{|C|} P(c_i| \sim t) \log P(c_i| \sim t) \quad (1)$$

- Mutual Information

$$I(t, c) = \log \frac{P(t \wedge c)}{P(t) \times P(c)}, \qquad I_{avg}(t) = \sum_{i=1}^{|C|} P(c_i) I(t, c_i) \quad (2)$$

Background on Feature Selection / Dimensionality Reduction

Techniques to deal with sparsity and high-dimensionality in BOW

- Information Gain

$$G(t) = -\sum_{i=1}^{|C|} P(c_i) \log P(c_i) + P(t) \sum_{i=1}^{|C|} P(c_i|t) \log P(c_i|t) + P(\sim t) \sum_{i=1}^{|C|} P(c_i| \sim t) \log P(c_i| \sim t) \quad (1)$$

- Mutual Information

$$I(t, c) = \log \frac{P(t \wedge c)}{P(t) \times P(c)}, \qquad I_{avg}(t) = \sum_{i=1}^{|C|} P(c_i) I(t, c_i) \quad (2)$$

- Latent Semantic Indexing (LSI)

$$X = TSD^T \quad (3)$$

Distributed Word Representations

Representating each word w_i using vector $v_{w_i} \in \mathbb{R}^k$ ($k \in [50, 300]$)

Distributed Word Representations

Representating each word w_i using vector $v_{w_i} \in \mathbb{R}^k$ ($k \in [50, 300]$)

Need for Distributed Word Representations

Distributed Word Representations

Representating each word w_i using vector $v_{w_i} \in \mathbb{R}^k$ ($k \in [50, 300]$)

Need for Distributed Word Representations

- Curse of Dimensionality

Distributed Word Representations

Representating each word w_i using vector $v_{w_i} \in \mathbb{R}^k$ ($k \in [50, 300]$)

Need for Distributed Word Representations

- Curse of Dimensionality
  - One-hot representations grow with the size of vocabulary

Distributed Word Representations

Representating each word w_i using vector $v_{w_i} \in \mathbb{R}^k$ ($k \in [50, 300]$)

Need for Distributed Word Representations

- Curse of Dimensionality
  - One-hot representations grow with the size of vocabulary
  - Parameters in language modelling grow exponentially with the size of vocabulary

Distributed Word Representations

Representating each word w_i using vector $v_{w_i} \in \mathbb{R}^k$ ($k \in [50, 300]$)

Need for Distributed Word Representations

- Curse of Dimensionality
  - One-hot representations grow with the size of vocabulary
  - Parameters in language modelling grow exponentially with the size of vocabulary
- No Word Similarity Measure

Distributed Word Representations

Representating each word w_i using vector $v_{w_i} \in \mathbb{R}^k$ ($k \in [50, 300]$)

Need for Distributed Word Representations

- Curse of Dimensionality
  - One-hot representations grow with the size of vocabulary
  - Parameters in language modelling grow exponentially with the size of vocabulary
- No Word Similarity Measure
  - One-hot representations are orthogonal representations

Distributed Word Representations

Representating each word w_i using vector $v_{w_i} \in \mathbb{R}^k$ ($k \in [50, 300]$)

Need for Distributed Word Representations

- Curse of Dimensionality
  - One-hot representations grow with the size of vocabulary
  - Parameters in language modelling grow exponentially with the size of vocabulary
- No Word Similarity Measure
  - One-hot representations are orthogonal representations
  - Cannot capture semantic similarity between words

Neural Probabilistic Language Model

Bengio et al. [2] developed *Neural Probabilistic Language Model (NPLM)* to learn

Neural Probabilistic Language Model

Bengio et al. [2] developed *Neural Probabilistic Language Model (NPLM)* to learn

1. Distributed word vectors

Neural Probabilistic Language Model

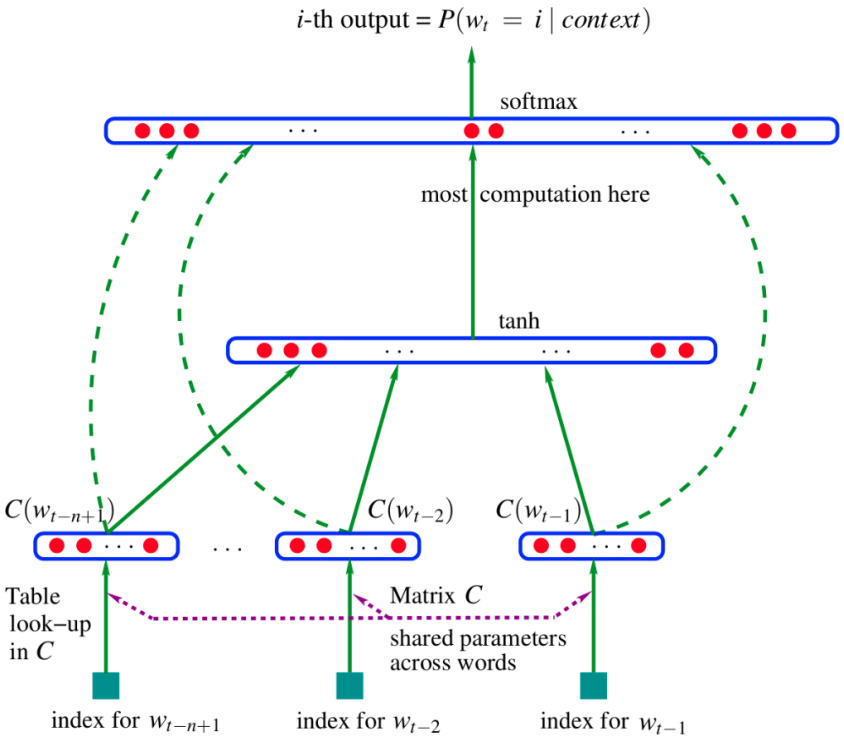
Bengio et al. [2] developed *Neural Probabilistic Language Model (NPLM)* to learn

1. Distributed word vectors
2. A probability function that, using these word vectors, learns a statistical model of language

Neural Probabilistic Language Model

Bengio et al. [2] developed *Neural Probabilistic Language Model (NPLM)* to learn

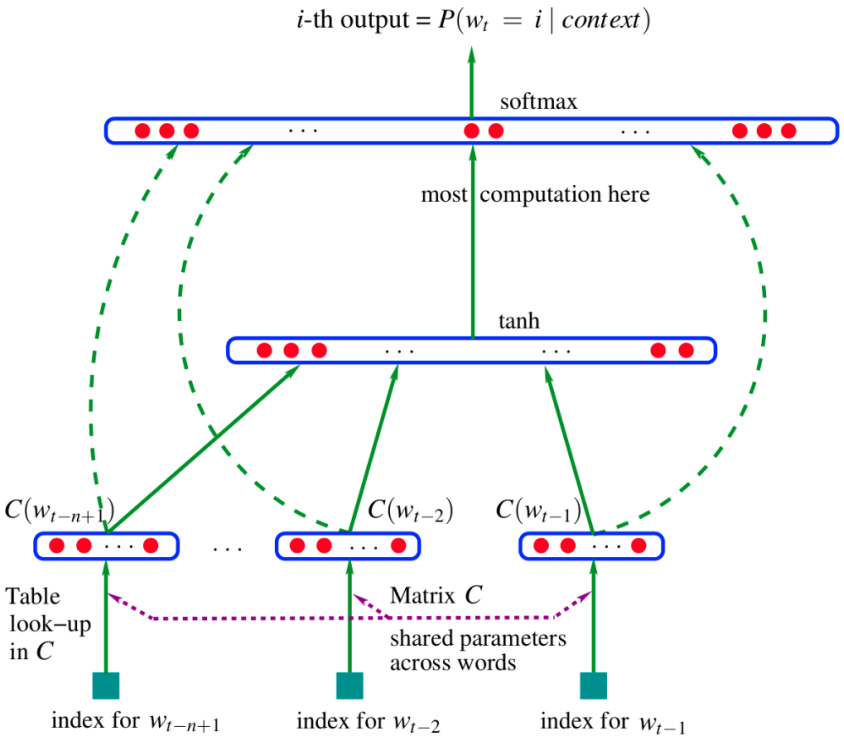
1. Distributed word vectors
2. A probability function that, using these word vectors, learns a statistical model of language



Neural Probabilistic Language Model

Bengio et al. [2] developed *Neural Probabilistic Language Model (NPLM)* to learn

1. Distributed word vectors
2. A probability function that, using these word vectors, learns a statistical model of language

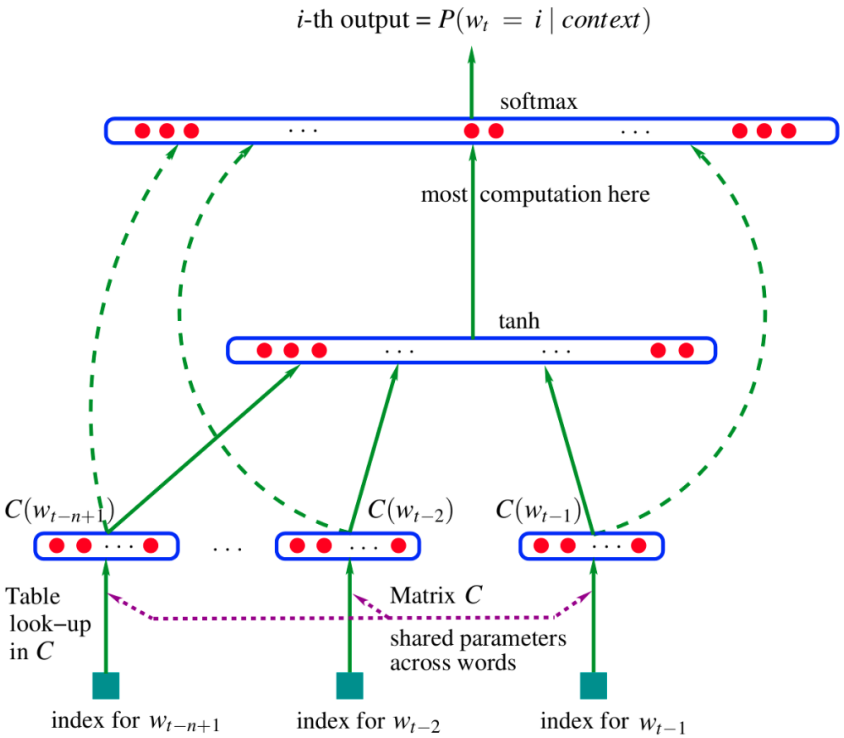


$$P(w_t|w_1^{t-1}) \approx P(w_t|w_{t-n+1}^{t-1}) \quad (4)$$

Neural Probabilistic Language Model

Bengio et al. [2] developed *Neural Probabilistic Language Model (NPLM)* to learn

1. Distributed word vectors
2. A probability function that, using these word vectors, learns a statistical model of language



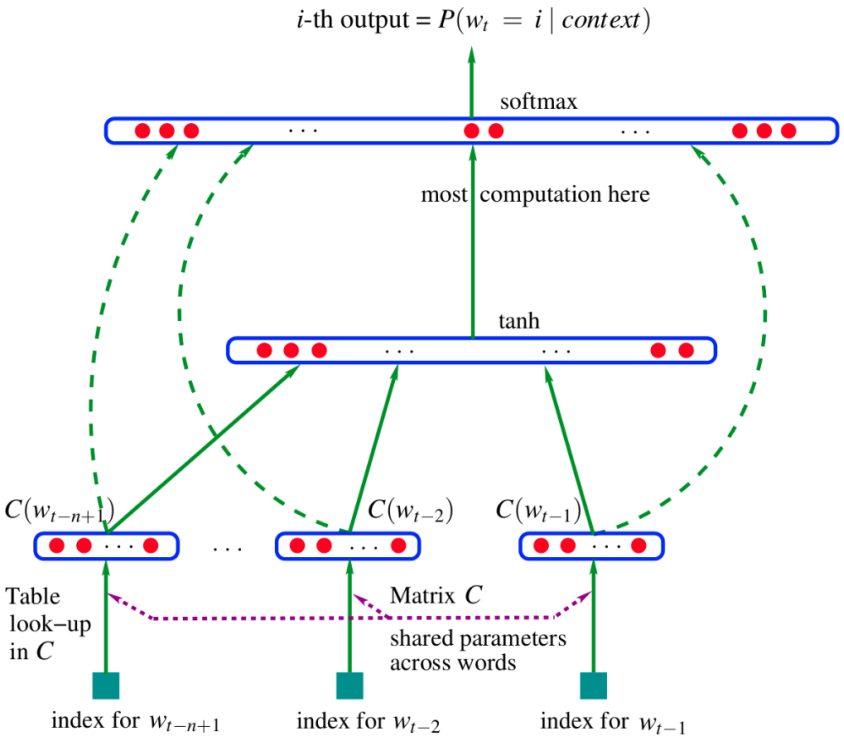
$$P(w_t|w_1^{t-1}) \approx P(w_t|w_{t-n+1}^{t-1}) \tag{4}$$

$$y = b + Utanh(d + Hx), \quad y \in \mathbb{R}^{|V|} \tag{5}$$

Neural Probabilistic Language Model

Bengio et al. [2] developed *Neural Probabilistic Language Model (NPLM)* to learn

1. Distributed word vectors
2. A probability function that, using these word vectors, learns a statistical model of language



$$P(w_t|w_1^{t-1}) \approx P(w_t|w_{t-n+1}^{t-1}) \tag{4}$$

$$y = b + Utanh(d + Hx), \quad y \in \mathbb{R}^{|V|} \tag{5}$$

$$P(w_t = i|w_{t-1}, \ldots, w_{t-n+1}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}} \tag{6}$$

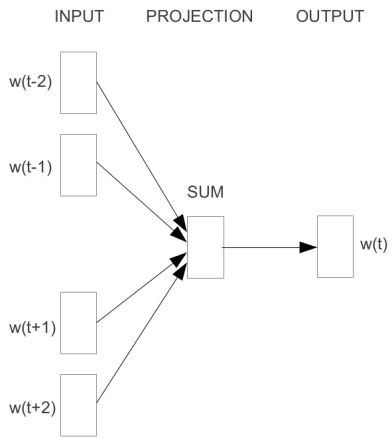
Log-Linear Models

Log-Linear Models for learning distributed word vectors are proposed in Mikolov et al. [7]. These models use word vectors to predict other words in the context.

Log-Linear Models

Log-Linear Models for learning distributed word vectors are proposed in Mikolov et al. [7]. These models use word vectors to predict other words in the context.

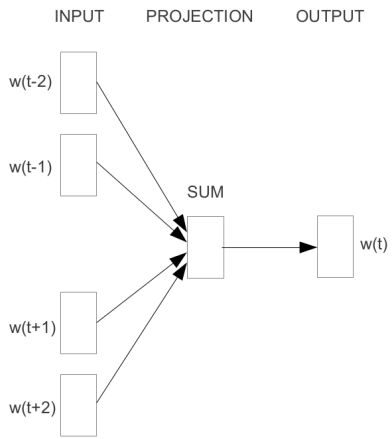
1. Continuous Bag-of-Words Model



Log-Linear Models

Log-Linear Models for learning distributed word vectors are proposed in Mikolov et al. [7]. These models use word vectors to predict other words in the context.

1. Continuous Bag-of-Words Model

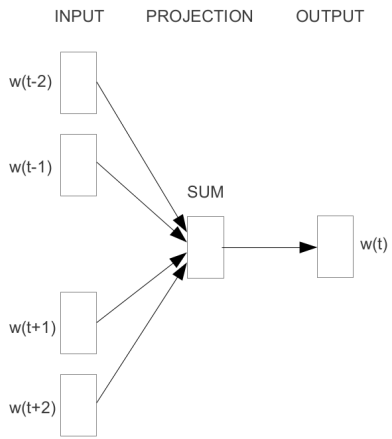


$$h = w_{t-k} + \ldots + w_{t-1} + w_{t+1} + \cdots + w_{t+k} \quad (7)$$

Log-Linear Models

Log-Linear Models for learning distributed word vectors are proposed in Mikolov et al. [7]. These models use word vectors to predict other words in the context.

1. Continuous Bag-of-Words Model



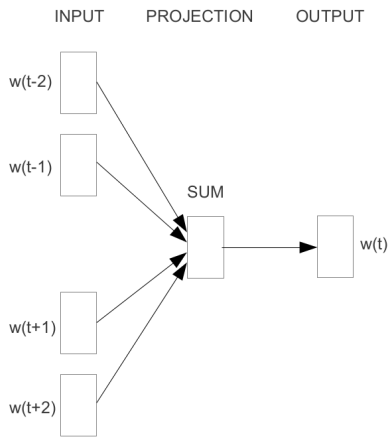
$$h = w_{t-k} + \ldots + w_{t-1} + w_{t+1} + \cdots + w_{t+k} \quad (7)$$

$$y = b + Uh, \quad y \in \mathbb{R}^{|V|} \quad (8)$$

Log-Linear Models

Log-Linear Models for learning distributed word vectors are proposed in Mikolov et al. [7]. These models use word vectors to predict other words in the context.

1. Continuous Bag-of-Words Model



$$h = w_{t-k} + \ldots + w_{t-1} + w_{t+1} + \cdots + w_{t+k} \quad (7)$$

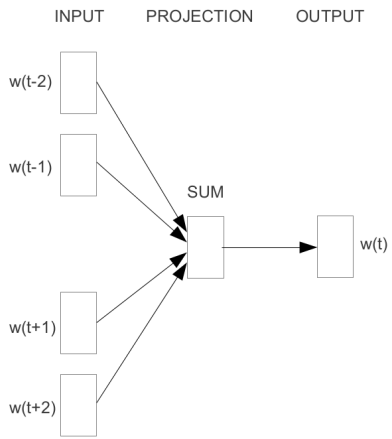
$$y = b + Uh, \quad y \in \mathbb{R}^{|V|} \quad (8)$$

$$P(w_t | w_{t-k}, \ldots, w_{t+k}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}} \quad (9)$$

Log-Linear Models

Log-Linear Models for learning distributed word vectors are proposed in Mikolov et al. [7]. These models use word vectors to predict other words in the context.

1. Continuous Bag-of-Words Model

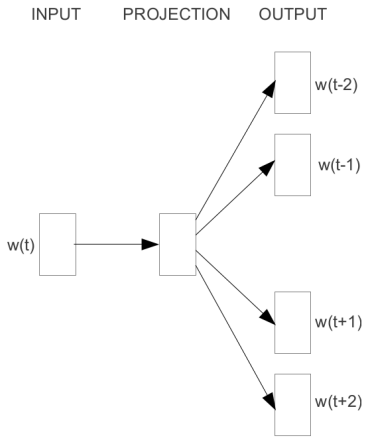


$$h = w_{t-k} + \ldots + w_{t-1} + w_{t+1} + \cdots + w_{t+k} \quad (7)$$

$$y = b + Uh, \quad y \in \mathbb{R}^{|V|} \quad (8)$$

$$P(w_t | w_{t-k}, \ldots, w_{t+k}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}} \quad (9)$$

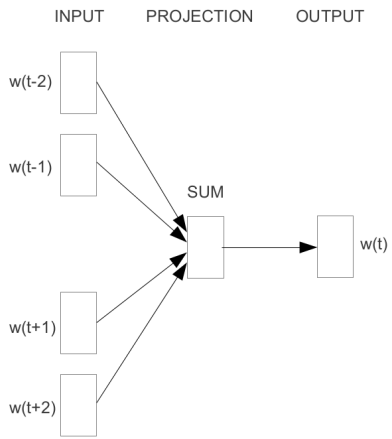
2. Skip-Gram Model



Log-Linear Models

Log-Linear Models for learning distributed word vectors are proposed in Mikolov et al. [7]. These models use word vectors to predict other words in the context.

1. Continuous Bag-of-Words Model

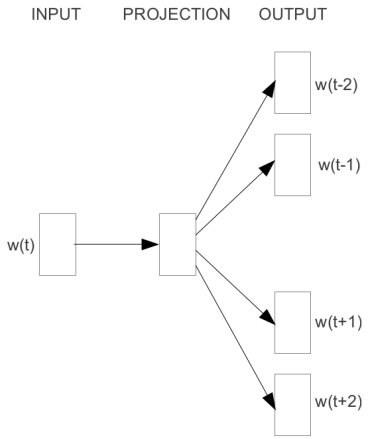


$$h = w_{t-k} + \ldots + w_{t-1} + w_{t+1} + \cdots + w_{t+k} \quad (7)$$

$$y = b + Uh, \quad y \in \mathbb{R}^{|V|} \quad (8)$$

$$P(w_t | w_{t-k}, \ldots, w_{t+k}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}} \quad (9)$$

2. Skip-Gram Model



$$P(w_{t+j} | w_t) = \frac{e^{(v_{w_t} \cdot v_{w_{t+j}})}}{\sum_i e^{(v_{w_t} \cdot v_{w_i})}} \quad (10)$$

Distributed Document Representations

Motivation for learning distributed document representations

Distributed Document Representations

Motivation for learning distributed document representations

1. Traditional representations do not encode semantic similarity between documents. Therefore, cannot handle synonyms

Distributed Document Representations

Motivation for learning distributed document representations

1. Traditional representations do not encode semantic similarity between documents. Therefore, cannot handle synonyms
2. Drawbacks in BOW like sparsity, high-dimensionality, inability to encode context information and consider word ordering

Distributed Document Representations

Motivation for learning distributed document representations

1. Traditional representations do not encode semantic similarity between documents. Therefore, cannot handle synonyms
2. Drawbacks in BOW like sparsity, high-dimensionality, inability to encode context information and consider word ordering
3. Compositionality of word vectors beyond weighted average [9, 15, 14, 4, 8] is not simple

Distributed Document Representations

Motivation for learning distributed document representations

1. Traditional representations do not encode semantic similarity between documents. Therefore, cannot handle synonyms
2. Drawbacks in BOW like sparsity, high-dimensionality, inability to encode context information and consider word ordering
3. Compositionality of word vectors beyond weighted average [9, 15, 14, 4, 8] is not simple
4. Socher et al. [13] propose a Recursive Tensor Neural Network (RTNN) to compose word vectors for learning sentence representations using the parse-tree of the sentence in a bottom-up fashion

Distributed Document Representations

Motivation for learning distributed document representations

1. Traditional representations do not encode semantic similarity between documents. Therefore, cannot handle synonyms
2. Drawbacks in BOW like sparsity, high-dimensionality, inability to encode context information and consider word ordering
3. Compositionality of word vectors beyond weighted average [9, 15, 14, 4, 8] is not simple
4. Socher et al. [13] propose a Recursive Tensor Neural Network (RTNN) to compose word vectors for learning sentence representations using the parse-tree of the sentence in a bottom-up fashion
    - Parsing, a computationally expensive step required for each sentence

Distributed Document Representations

Motivation for learning distributed document representations

1. Traditional representations do not encode semantic similarity between documents. Therefore, cannot handle synonyms
2. Drawbacks in BOW like sparsity, high-dimensionality, inability to encode context information and consider word ordering
3. Compositionality of word vectors beyond weighted average [9, 15, 14, 4, 8] is not simple
4. Socher et al. [13] propose a Recursive Tensor Neural Network (RTNN) to compose word vectors for learning sentence representations using the parse-tree of the sentence in a bottom-up fashion
    - Parsing, a computationally expensive step required for each sentence
    - Composing sentence vectors to represent documents is not straight-forward

Our Model for Learning Document Representations

Inspired by the log-linear models to learn word vectors, we present model, to learn universal distributed representations for documents and words

Our Model for Learning Document Representations

Inspired by the log-linear models to learn word vectors, we present model, to learn universal distributed representations for documents and words

Hypothesis

> *Document Representations that encode semantic content of the document should be able to predict words in the document*

Our Model for Learning Document Representations

Inspired by the log-linear models to learn word vectors, we present model, to learn universal distributed representations for documents and words

Hypothesis

> *Document Representations that encode semantic content of the document should be able to predict words in the document*

Our model,

Our Model for Learning Document Representations

Inspired by the log-linear models to learn word vectors, we present model, to learn universal distributed representations for documents and words

Hypothesis

> *Document Representations that encode semantic content of the document should be able to predict words in the document*

Our model,

1. Learns distributed representations for document (and words) that encode the different semantic content in the documents

Our Model for Learning Document Representations

Inspired by the log-linear models to learn word vectors, we present model, to learn universal distributed representations for documents and words

Hypothesis

> *Document Representations that encode semantic content of the document should be able to predict words in the document*

Our model,

1. Learns distributed representations for document (and words) that encode the different semantic content in the documents
2. Embeds documents and words in the same k-dimensional space such that semantically similar entities have similar vector representations

Our Model for Learning Document Representations

We present an unsupervised neural network model that,

Our Model for Learning Document Representations

We present an unsupervised neural network model that,

1. Represents each document $d_i \in D$ by a vector $\mathrm{v}_i^D \in \mathbb{R}^k$
   Vectors are stored as columns of the matrix $\mathrm{D} = \left[\mathrm{v}_1^D, \ldots, \mathrm{v}_{|D|}^D\right] \in \mathbb{R}^{k \times |D|}$

Our Model for Learning Document Representations

We present an unsupervised neural network model that,

1. Represents each document $d_i \in D$ by a vector $\mathrm{v}_i^D \in \mathbb{R}^k$
   Vectors are stored as columns of the matrix $\mathrm{D} = \left[\mathrm{v}_1^D, \ldots, \mathrm{v}_{|D|}^D\right] \in \mathbb{R}^{k \times |D|}$

2. Each word $w_i \in W$, is represented by a vector $\mathrm{v}_i^W \in \mathbb{R}^k$ Vectors are stored as columns of the matrix $\mathrm{W} = \left[\mathrm{v}_1^W, \ldots, \mathrm{v}_{|V|}^W\right] \in \mathbb{R}^{k \times |V|}$

Our Model for Learning Document Representations

We present an unsupervised neural network model that,

1. Represents each document $d_i \in D$ by a vector $\mathrm{v}_i^D \in \mathbb{R}^k$
   Vectors are stored as columns of the matrix $\mathrm{D} = \left[\mathrm{v}_1^D, \ldots, \mathrm{v}_{|D|}^D\right] \in \mathbb{R}^{k \times |D|}$
2. Each word $w_i \in W$, is represented by a vector $\mathrm{v}_i^W \in \mathbb{R}^k$ Vectors are stored as columns of the matrix $\mathrm{W} = \left[\mathrm{v}_1^W, \ldots, \mathrm{v}_{|V|}^W\right] \in \mathbb{R}^{k \times |V|}$
3. Given a sequence of words, $(w_{t-c}, \ldots, w_{t+c})$ in document d_i, estimates

$$p(w_t | d_i, w_{t-c}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+c})$$

Our Model for Learning Document Representations

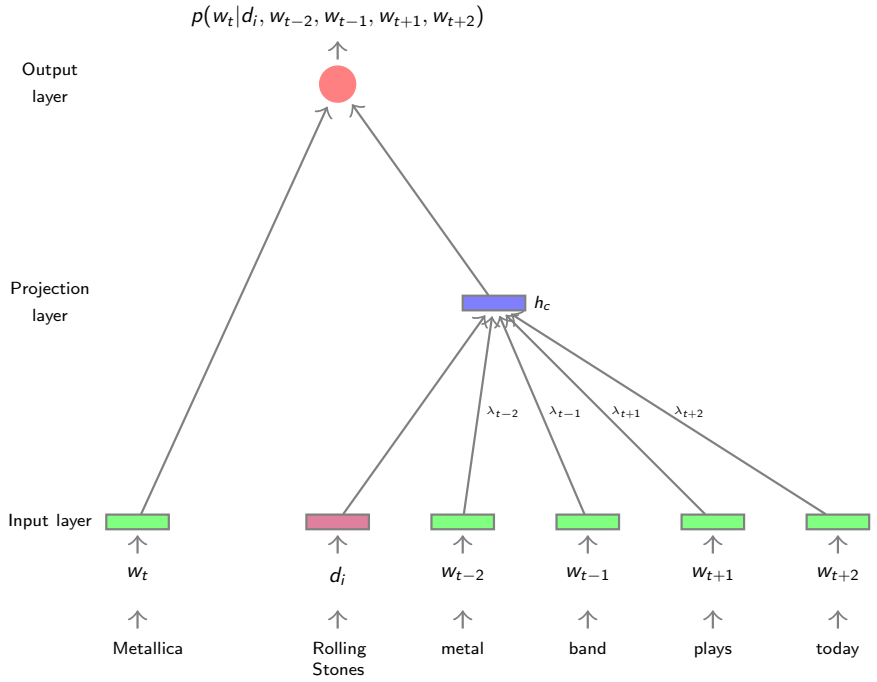
We present an unsupervised neural network model that,

1. Represents each document $d_i \in D$ by a vector $\mathrm{v}_i^D \in \mathbb{R}^k$
   Vectors are stored as columns of the matrix $\mathrm{D} = \left[\mathrm{v}_1^D, \ldots, \mathrm{v}_{|D|}^D\right] \in \mathbb{R}^{k \times |D|}$
2. Each word $w_i \in W$, is represented by a vector $\mathrm{v}_i^W \in \mathbb{R}^k$ Vectors are stored as columns of the matrix $\mathrm{W} = \left[\mathrm{v}_1^W, \ldots, \mathrm{v}_{|V|}^W\right] \in \mathbb{R}^{k \times |V|}$
3. Given a sequence of words, $(w_{t-c}, \ldots, w_{t+c})$ in document d_i, estimates

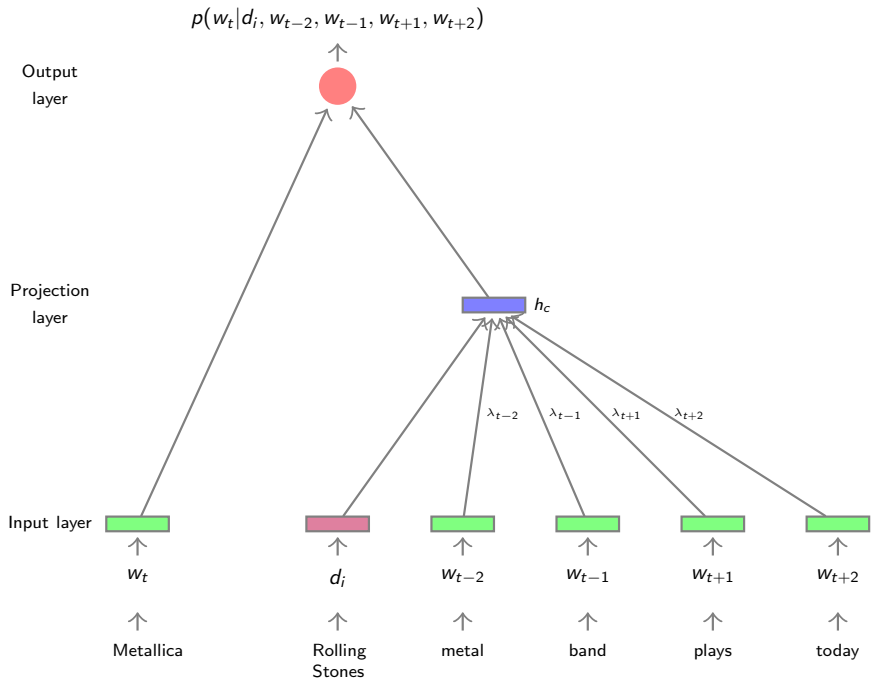
$$p(w_t | d_i, w_{t-c}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+c})$$

4. Maximizes the probability of predicting the words correctly to learn D and W and the parameters of the probability function

Our Model for Learning Document Representations



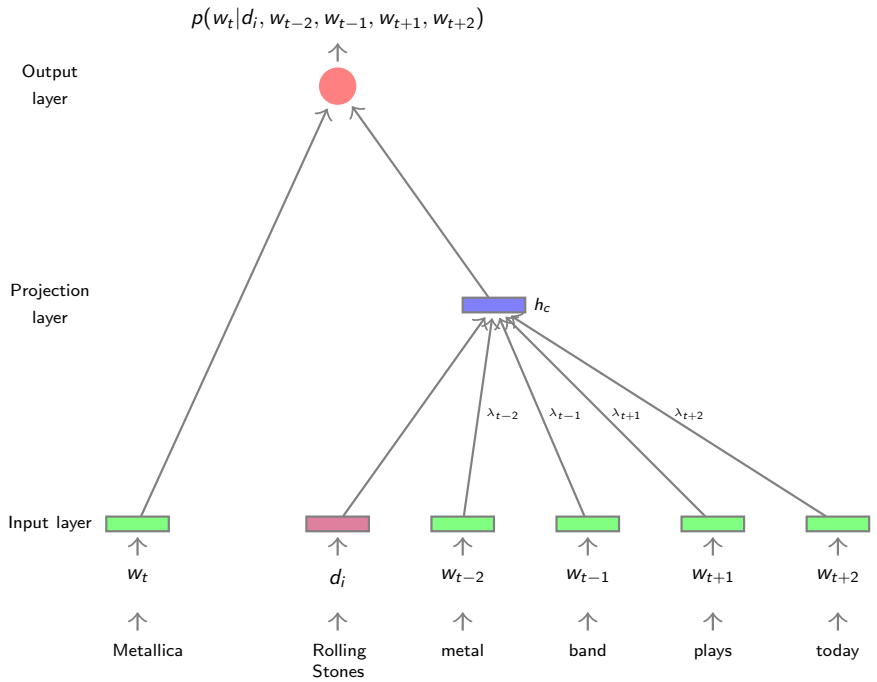
Our Model for Learning Document Representations



Context Representation :

$$h_c = \mathrm{v}^D_{d_i} + \lambda_{t-c}\mathrm{v}^W_{w_{t-c}} + \ldots + \lambda_{t-1}\mathrm{v}^W_{w_{t-1}} + \lambda_{t+1}\mathrm{v}^W_{w_{t+1}} + \ldots + \lambda_{t+c}\mathrm{v}^W_{w_{t+c}} \tag{11}$$

Our Model for Learning Document Representations



Context Representation :

$$h_c = \mathrm{v}^D_{d_i} + \lambda_{t-c}\mathrm{v}^W_{w_{t-c}} + \ldots + \lambda_{t-1}\mathrm{v}^W_{w_{t-1}} + \lambda_{t+1}\mathrm{v}^W_{w_{t+1}} + \ldots + \lambda_{t+c}\mathrm{v}^W_{w_{t+c}} \tag{11}$$

Probability Estimation :

$$s_{w_i} = \sigma(\mathrm{v}^W_{w_i} \cdot h_c), \quad \sigma(x) = \frac{1}{1+e^{-x}} \tag{12}$$

$$p(w_t|d_i, w_{t-c}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+c}) = \frac{e^{s_{w_t}}}{\sum_{i \in V} e^{s_{w_i}}} \tag{13}$$

Training Objective

1. Training data $\mathcal{T} = \{d_i^{(m)}, w_{t-c}^{(m)}, \ldots, w_{t+c}^{(m)}\}_{m=1}^{m=M}$

Training Objective

1. Training data $\mathcal{T} = \{d_i^{(m)}, w_{t-c}^{(m)}, \ldots, w_{t+c}^{(m)}\}_{m=1}^{m=M}$
2. Learn optimum parameter set $\Theta = (\mathrm{D}, \mathrm{W}, \Lambda)$, i.e. document and word vectors and the neural network weights Λ

Training Objective

1. Training data $\mathcal{T} = \{d_i^{(m)}, w_{t-c}^{(m)}, \ldots, w_{t+c}^{(m)}\}_{m=1}^{m=M}$
2. Learn optimum parameter set $\Theta = (\mathrm{D}, \mathrm{W}, \Lambda)$, i.e. document and word vectors and the neural network weights Λ
3. Maximize average log-probability of predicting w_t correctly in each sequence in $\mathcal{T}$

$$\hat{\Theta} = \arg\max_{\Theta} \; I(\mathcal{T}, \Theta) \tag{14}$$

$$I(\mathcal{T}, \Theta) = \frac{1}{M} \sum_{m=1}^{M} \log \left[p(w_t^{(m)} | d_i^{(m)}, w_{t-c}^{(m)}, \ldots, w_{t-1}^{(m)}, w_{t+1}^{(m)}, \ldots, w_{t+c}^{(m)}) \right] \tag{15}$$

Training Objective

1. Training data $\mathcal{T} = \{d_i^{(m)}, w_{t-c}^{(m)}, \ldots, w_{t+c}^{(m)}\}_{m=1}^{m=M}$
2. Learn optimum parameter set $\Theta = (\mathrm{D}, \mathrm{W}, \Lambda)$, i.e. document and word vectors and the neural network weights Λ
3. Maximize average log-probability of predicting w_t correctly in each sequence in $\mathcal{T}$

$$\hat{\Theta} = \arg\max_{\Theta} \; l(\mathcal{T}, \Theta) \tag{14}$$

$$l(\mathcal{T}, \Theta) = \frac{1}{M} \sum_{m=1}^{M} \log \left[p(w_t^{(m)} | d_i^{(m)}, w_{t-c}^{(m)}, \ldots, w_{t-1}^{(m)}, w_{t+1}^{(m)}, \ldots, w_{t+c}^{(m)}) \right] \tag{15}$$

4. Use Stochastic Gradient Descent (SGD) to update parameters

$$\theta_i^{(x)} = \theta_i^{(x-1)} + \gamma \frac{\partial l(\mathcal{T}, \Theta)}{\partial \theta_i} \tag{16}$$

Noise Contrastive Estimation

1. Soft-max computation is expensive, $\mathcal{O}(V)$

Noise Contrastive Estimation

1. Soft-max computation is expensive, $\mathcal{O}(V)$
2. Speed-ups using **Hierarchical soft-max** [12] and **Importance sampling** to approximate the likelihood gradient [3, 1]

Noise Contrastive Estimation

1. Soft-max computation is expensive, $\mathcal{O}(V)$
2. Speed-ups using **Hierarchical soft-max** [12] and **Importance sampling** to approximate the likelihood gradient [3, 1]
    - Finding well-performing trees in Hierarchical soft-max is not trivial

Noise Contrastive Estimation

1. Soft-max computation is expensive, $\mathcal{O}(V)$
2. Speed-ups using **Hierarchical soft-max** [12] and **Importance sampling** to approximate the likelihood gradient [3, 1]
    - Finding well-performing trees in Hierarchical soft-max is not trivial
    - Importance sampling suffers from stability issues

Noise Contrastive Estimation

1. Soft-max computation is expensive, $\mathcal{O}(V)$
2. Speed-ups using **Hierarchical soft-max** [12] and **Importance sampling** to approximate the likelihood gradient [3, 1]
   - Finding well-performing trees in Hierarchical soft-max is not trivial
   - Importance sampling suffers from stability issues
3. **Noise Contrastive Estimation** (NCE) [6] fits unnormalized probabilities

Noise Contrastive Estimation

1. Soft-max computation is expensive, $\mathcal{O}(V)$
2. Speed-ups using **Hierarchical soft-max** [12] and **Importance sampling** to approximate the likelihood gradient [3, 1]
    - Finding well-performing trees in Hierarchical soft-max is not trivial
    - Importance sampling suffers from stability issues
3. **Noise Contrastive Estimation** (NCE) [6] fits unnormalized probabilities
    - Reduces the problem of *probability density estimation* to *probabilistic binary classification*

Noise Contrastive Estimation

1. Soft-max computation is expensive, $\mathcal{O}(V)$
2. Speed-ups using **Hierarchical soft-max** [12] and **Importance sampling** to approximate the likelihood gradient [3, 1]
   - Finding well-performing trees in Hierarchical soft-max is not trivial
   - Importance sampling suffers from stability issues
3. **Noise Contrastive Estimation** (NCE) [6] fits unnormalized probabilities
   - Reduces the problem of *probability density estimation* to *probabilistic binary classification*
   - Adaptation to NPLM [11] and learning word embeddings [10] show significant training time speed-ups

Noise Contrastive Estimation (contd.)

1. Given a sequence of words $(w_{t-c}, \ldots, w_{t+c})$ in document d_i

Noise Contrastive Estimation (contd.)

1. Given a sequence of words $(w_{t-c}, \ldots, w_{t+c})$ in document d_i
   - *Earlier objective* : Maximize $p(w_t | d_i, w_{t-c}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+c})$

Noise Contrastive Estimation (contd.)

1. Given a sequence of words $(w_{t-c}, \ldots, w_{t+c})$ in document d_i
   - *Earlier objective* : Maximize $p(w_t | d_i, w_{t-c}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+c})$
   - *New objective* : Build binary classifier to distinguish between correct middle word w_t and random corrupt word

Noise Contrastive Estimation (contd.)

1. Given a sequence of words $(w_{t-c}, \ldots, w_{t+c})$ in document d_i
   - *Earlier objective* : Maximize $p(w_t | d_i, w_{t-c}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+c})$
   - *New objective* : Build binary classifier to distinguish between correct middle word w_t and random corrupt word

For NCE Binary Classification Objective :

Noise Contrastive Estimation (contd.)

1. Given a sequence of words $(w_{t-c}, \ldots, w_{t+c})$ in document d_i
   - *Earlier objective* : Maximize $p(w_t | d_i, w_{t-c}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+c})$
   - *New objective* : Build binary classifier to distinguish between correct middle word w_t and random corrupt word

For NCE Binary Classification Objective :

1. New labeled training data : $\mathcal{T} = \{d_i^{(m)}, w_{t-c}^{(m)}, \ldots, w_{t+c}^{(m)}, Y^{(m)} = 1\}_{m=1}^{m=M}$

Noise Contrastive Estimation (contd.)

1. Given a sequence of words $(w_{t-c}, \ldots, w_{t+c})$ in document d_i
   - *Earlier objective* : Maximize $p(w_t | d_i, w_{t-c}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+c})$
   - *New objective* : Build binary classifier to distinguish between correct middle word w_t and random corrupt word

For NCE Binary Classification Objective :

1. New labeled training data : $\mathcal{T} = \{d_i^{(m)}, w_{t-c}^{(m)}, \ldots, w_{t+c}^{(m)}, Y^{(m)} = 1\}_{m=1}^{m=M}$
2. For every positive training sequence, n negative training sequences introduced where,

Noise Contrastive Estimation (contd.)

1. Given a sequence of words $(w_{t-c}, \ldots, w_{t+c})$ in document d_i
   - *Earlier objective* : Maximize $p(w_t|d_i, w_{t-c}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+c})$
   - *New objective* : Build binary classifier to distinguish between correct middle word w_t and random corrupt word

For NCE Binary Classification Objective :

1. New labeled training data : $\mathcal{T} = \{d_i^{(m)}, w_{t-c}^{(m)}, \ldots, w_{t+c}^{(m)}, Y^{(m)} = 1\}_{m=1}^{m=M}$
2. For every positive training sequence, n negative training sequences introduced where,
   - The observed middle word w_t is replaced by a corrupt word w_x drawn from a noise distribution $P_n(w)$

Noise Contrastive Estimation (contd.)

1. Given a sequence of words $(w_{t-c}, \ldots, w_{t+c})$ in document d_i
   - *Earlier objective* : Maximize $p(w_t|d_i, w_{t-c}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+c})$
   - *New objective* : Build binary classifier to distinguish between correct middle word w_t and random corrupt word

For NCE Binary Classification Objective :

1. New labeled training data : $\mathcal{T} = \{d_i^{(m)}, w_{t-c}^{(m)}, \ldots, w_{t+c}^{(m)}, Y^{(m)} = 1\}_{m=1}^{m=M}$
2. For every positive training sequence, n negative training sequences introduced where,
   - The observed middle word w_t is replaced by a corrupt word w_x drawn from a noise distribution $P_n(w)$
   - The label for the negative sequence is $Y = 0$

Noise Contrastive Estimation (contd.)

1. Given a sequence of words $(w_{t-c}, \ldots, w_{t+c})$ in document d_i
   - *Earlier objective* : Maximize $p(w_t|d_i, w_{t-c}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+c})$
   - *New objective* : Build binary classifier to distinguish between correct middle word w_t and random corrupt word

For NCE Binary Classification Objective :

1. New labeled training data : $\mathcal{T} = \{d_i^{(m)}, w_{t-c}^{(m)}, \ldots, w_{t+c}^{(m)}, Y^{(m)} = 1\}_{m=1}^{m=M}$
2. For every positive training sequence, n negative training sequences introduced where,
   - The observed middle word w_t is replaced by a corrupt word w_x drawn from a noise distribution $P_n(w)$
   - The label for the negative sequence is $Y = 0$
3. Complete training data : $\mathcal{T} = \{d_i^{(m)}, w_{t-c}^{(m)}, \ldots, w_{t+c}^{(m)}, Y^{(m)}\}_{m=1}^{m=M+nM}$

Noise Contrastive Estimation (contd.)

Given a sequence of words $(w_{t-c}, \ldots, w_{t+c})$ in document d_i, our new objective is to predict whether the sequence is legitimate

Noise Contrastive Estimation (contd.)

Given a sequence of words $(w_{t-c}, \ldots, w_{t+c})$ in document d_i, our new objective is to predict whether the sequence is legitimate

- We build a probabilistic binary classifier to predict the label Y

Noise Contrastive Estimation (contd.)

Given a sequence of words $(w_{t-c}, \ldots, w_{t+c})$ in document d_i, our new objective is to predict whether the sequence is legitimate

- We build a probabilistic binary classifier to predict the label Y

$$P(Y = 1 | d_i, w_{t-c}, \ldots, w_{t+c}, \Theta) = \sigma(\mathrm{v}_{w_t}^{W} \cdot h_c) \tag{17}$$

Noise Contrastive Estimation (contd.)

Given a sequence of words $(w_{t-c}, \ldots, w_{t+c})$ in document d_i, our new objective is to predict whether the sequence is legitimate

- We build a probabilistic binary classifier to predict the label Y

$$P(Y = 1 | d_i, w_{t-c}, \ldots, w_{t+c}, \Theta) = \sigma(\mathrm{v}_{w_t}^W \cdot h_c) \tag{17}$$

$$P(Y = 0 | d_i, w_{t-c}, \ldots, w_{t+c}, \Theta) = 1 - \sigma(\mathrm{v}_{w_t}^W \cdot h_c) \tag{18}$$

Noise Contrastive Estimation (contd.)

Given a sequence of words $(w_{t-c}, \ldots, w_{t+c})$ in document d_i, our new objective is to predict whether the sequence is legitimate

- We build a probabilistic binary classifier to predict the label Y

$$P(Y = 1 | d_i, w_{t-c}, \ldots, w_{t+c}, \Theta) = \sigma(\mathrm{v}_{w_t}^W \cdot h_c) \tag{17}$$

$$P(Y = 0 | d_i, w_{t-c}, \ldots, w_{t+c}, \Theta) = 1 - \sigma(\mathrm{v}_{w_t}^W \cdot h_c) \tag{18}$$

$$P(Y | d_i, w_{t-c}, \ldots, w_{t+c}, \Theta) = [\sigma(\mathrm{v}_{w_t}^W \cdot h_c)]^Y [1 - \sigma(\mathrm{v}_{w_t}^W \cdot h_c)]^{1-Y} \tag{19}$$

Learning Objective with NCE

Given the training data $\mathcal{T} = \{d_i^{(m)}, w_{t-c}^{(m)}, \ldots, w_{t+c}^{(m)}, Y^{(m)}\}_{m=1}^{m=M+nM}$, we maximize the log-likelihood of observing it

---

$P_\Theta(Y_m)$ is a shorthand notation for $P(Y_m|d_i^{(m)}, w_{t-c}^{(m)}, \ldots, w_{t+c}^{(m)}, \Theta)$
Y_m is the predicted label

Learning Objective with NCE

Given the training data $\mathcal{T} = \{d_i^{(m)}, w_{t-c}^{(m)}, \ldots, w_{t+c}^{(m)}, Y^{(m)}\}_{m=1}^{m=M+nM}$, we maximize the log-likelihood of observing it

$$\hat{\Theta} = \arg\max_{\Theta} \; l(\mathcal{T}, \Theta) \tag{20}$$

$$l(\mathcal{T}, \Theta) = \sum_{m=1}^{M+nM} \log P_\Theta(Y_m = Y^{(m)}) \tag{21}$$

$P_\Theta(Y_m)$ is a shorthand notation for $P(Y_m | d_i^{(m)}, w_{t-c}^{(m)}, \ldots, w_{t+c}^{(m)}, \Theta)$

Y_m is the predicted label

Learning Objective with NCE

Given the training data $\mathcal{T} = \{d_i^{(m)}, w_{t-c}^{(m)}, \ldots, w_{t+c}^{(m)}, Y^{(m)}\}_{m=1}^{m=M+nM}$, we maximize the log-likelihood of observing it

$$\hat{\Theta} = \arg\max_{\Theta} \; l(\mathcal{T}, \Theta) \tag{20}$$

$$l(\mathcal{T}, \Theta) = \sum_{m=1}^{M+nM} \log P_\Theta(Y_m = Y^{(m)}) \tag{21}$$

The logarithm of the probability estimate is given by,

$$\log P_\Theta(Y_m = Y^{(m)}) = Y^{(m)} \log \sigma(\mathrm{v}^W_{w_t^{(m)}} \cdot h_c^{(m)}) + (1 - Y^{(m)}) \log(1 - \sigma(\mathrm{v}^W_{w_t^{(m)}} \cdot h_c^{(m)})) \tag{22}$$

$P_\Theta(Y_m)$ is a shorthand notation for $P(Y_m | d_i^{(m)}, w_{t-c}^{(m)}, \ldots, w_{t+c}^{(m)}, \Theta)$

Y_m is the predicted label

Parameter Estimation

We use SGD to learn parameters i.e. document and word vectors and the neural network weights

$$\theta_i^{(x)} = \theta_i^{(x-1)} + \gamma \frac{\partial l(\mathcal{T}, \Theta)}{\partial \theta_i} \tag{23}$$

$d = \mathrm{v}_{w_t}^W \cdot h_c$, is the pre-sigmoid activation

Parameter Estimation

We use SGD to learn parameters i.e. document and word vectors and the neural network weights

$$\theta_i^{(x)} = \theta_i^{(x-1)} + \gamma \frac{\partial l(\mathcal{T}, \Theta)}{\partial \theta_i} \tag{23}$$

Gradient of $\log P_\Theta(Y_m = Y^{(m)})$ with respect to parameter θ,

$d = \mathrm{v}_{w_t}^W \cdot h_c$, is the pre-sigmoid activation

Parameter Estimation

We use SGD to learn parameters i.e. document and word vectors and the neural network weights

$$\theta_i^{(x)} = \theta_i^{(x-1)} + \gamma \frac{\partial l(\mathcal{T}, \Theta)}{\partial \theta_i} \tag{23}$$

Gradient of $\log P_\Theta(Y_m = Y^{(m)})$ with respect to parameter θ,

$$\frac{\partial \log P_\Theta(Y_m = Y^{(m)})}{\partial \theta} = \left[Y^{(m)} \frac{1}{\sigma(d^{(m)})} - (1 - Y^{(m)}) \frac{1}{(1 - \sigma(d^{(m)}))} \right] \frac{\partial \sigma(d^{(m)})}{\partial \theta} \tag{24}$$

$d = \mathrm{v}_{w_t}^W \cdot h_c$, is the pre-sigmoid activation

Parameter Estimation

We use SGD to learn parameters i.e. document and word vectors and the neural network weights

$$\theta_i^{(x)} = \theta_i^{(x-1)} + \gamma \frac{\partial l(\mathcal{T}, \Theta)}{\partial \theta_i} \tag{23}$$

Gradient of $\log P_\Theta(Y_m = Y^{(m)})$ with respect to parameter θ,

$$\frac{\partial \log P_\Theta(Y_m = Y^{(m)})}{\partial \theta} = \left[Y^{(m)} \frac{1}{\sigma(d^{(m)})} - (1 - Y^{(m)}) \frac{1}{(1 - \sigma(d^{(m)}))} \right] \frac{\partial \sigma(d^{(m)})}{\partial \theta} \tag{24}$$

$$\frac{\partial \log P_\Theta(Y_m = Y^{(m)})}{\partial \theta} = \left[Y^{(m)} \frac{1}{\sigma(d^{(m)})} - (1 - Y^{(m)}) \frac{1}{(1 - \sigma(d^{(m)}))} \right] \left[\sigma(d^{(m)})(1 - \sigma(d^{(m)})) \right] \frac{\partial d^{(m)}}{\partial \theta} \tag{25}$$

$d = v_{w_t}^W \cdot h_c$, is the pre-sigmoid activation

Parameter Estimation

We use SGD to learn parameters i.e. document and word vectors and the neural network weights

$$\theta_i^{(x)} = \theta_i^{(x-1)} + \gamma \frac{\partial l(\mathcal{T}, \Theta)}{\partial \theta_i} \tag{23}$$

Gradient of $\log P_\Theta(Y_m = Y^{(m)})$ with respect to parameter θ,

$$\frac{\partial \log P_\Theta(Y_m = Y^{(m)})}{\partial \theta} = \left[Y^{(m)} \frac{1}{\sigma(d^{(m)})} - (1 - Y^{(m)}) \frac{1}{(1 - \sigma(d^{(m)}))} \right] \frac{\partial \sigma(d^{(m)})}{\partial \theta} \tag{24}$$

$$\frac{\partial \log P_\Theta(Y_m = Y^{(m)})}{\partial \theta} = \left[Y^{(m)} \frac{1}{\sigma(d^{(m)})} - (1 - Y^{(m)}) \frac{1}{(1 - \sigma(d^{(m)}))} \right] \left[\sigma(d^{(m)})(1 - \sigma(d^{(m)})) \right] \frac{\partial d^{(m)}}{\partial \theta} \tag{25}$$

$$\frac{\partial \log P_\Theta(Y_m = Y^{(m)})}{\partial \theta} = \left[Y^{(m)} - \sigma(d^{(m)}) \right] \frac{\partial d^{(m)}}{\partial \theta} \tag{26}$$

$d = \mathrm{v}_{w_t}^W \cdot h_c$, is the pre-sigmoid activation

Parameter Estimation

We use SGD to learn parameters i.e. document and word vectors and the neural network weights

$$\theta_i^{(x)} = \theta_i^{(x-1)} + \gamma \frac{\partial l(\mathcal{T}, \Theta)}{\partial \theta_i} \tag{23}$$

Gradient of $\log P_\Theta(Y_m = Y^{(m)})$ with respect to parameter θ,

$$\frac{\partial \log P_\Theta(Y_m = Y^{(m)})}{\partial \theta} = \left[Y^{(m)} \frac{1}{\sigma(d^{(m)})} - (1 - Y^{(m)}) \frac{1}{(1 - \sigma(d^{(m)}))} \right] \frac{\partial \sigma(d^{(m)})}{\partial \theta} \tag{24}$$

$$\frac{\partial \log P_\Theta(Y_m = Y^{(m)})}{\partial \theta} = \left[Y^{(m)} \frac{1}{\sigma(d^{(m)})} - (1 - Y^{(m)}) \frac{1}{(1 - \sigma(d^{(m)}))} \right] \left[\sigma(d^{(m)})(1 - \sigma(d^{(m)})) \right] \frac{\partial d^{(m)}}{\partial \theta} \tag{25}$$

$$\frac{\partial \log P_\Theta(Y_m = Y^{(m)})}{\partial \theta} = \left[Y^{(m)} - \sigma(d^{(m)}) \right] \frac{\partial d^{(m)}}{\partial \theta} \tag{26}$$

$$\frac{\partial \log P_\Theta(Y_m = Y^{(m)})}{\partial \theta} = \left[Y^{(m)} - \sigma(\mathrm{v}^W_{w_t^{(m)}} \cdot h_c^{(m)}) \right] \frac{\partial (\mathrm{v}^W_{w_t^{(m)}} \cdot h_c^{(m)})}{\partial \theta} \tag{27}$$

$d = \mathrm{v}^W_{w_t} \cdot h_c$, is the pre-sigmoid activation

Update rule for Parameters

1. Document Vector :

$$(\mathrm{v}_{d_i}^{D(m)})^{(i+1)} = (\mathrm{v}_{d_i}^{D(m)})^{(i)} + \gamma \left[(Y^{(m)} - \sigma(\mathrm{v}_{w_t}^{W(m)} \cdot h_c^{(m)}))\mathrm{v}_{w_t}^{W(m)} - \beta \mathrm{v}_{d_i}^{D(m)} \right] \tag{28}$$

Update rule for Parameters

1. Document Vector :

$$(\mathrm{v}_{d_i^{(m)}}^{D})^{(i+1)} = (\mathrm{v}_{d_i^{(m)}}^{D})^{(i)} + \gamma \left[(Y^{(m)} - \sigma(\mathrm{v}_{w_t^{(m)}}^{W} \cdot h_c^{(m)}))\mathrm{v}_{w_t^{(m)}}^{W} - \beta \mathrm{v}_{d_i^{(m)}}^{D} \right] \tag{28}$$

2. Middle Word Vector :

$$(\mathrm{v}_{w_t^{(m)}}^{W})^{(i+1)} = (\mathrm{v}_{w_t^{(m)}}^{W})^{(i)} + \gamma \left[(Y^{(m)} - \sigma(\mathrm{v}_{w_t^{(m)}}^{W} \cdot h_c^{(m)}))h_c^{(m)} - \beta \mathrm{v}_{w_t^{(m)}}^{W} \right] \tag{29}$$

Update rule for Parameters

1. Document Vector :

$$(\mathrm{v}_{d_i^{(m)}}^{D})^{(i+1)} = (\mathrm{v}_{d_i^{(m)}}^{D})^{(i)} + \gamma \left[(Y^{(m)} - \sigma(\mathrm{v}_{w_t^{(m)}}^{W} \cdot h_c^{(m)}))\mathrm{v}_{w_t^{(m)}}^{W} - \beta \mathrm{v}_{d_i^{(m)}}^{D} \right] \tag{28}$$

2. Middle Word Vector :

$$(\mathrm{v}_{w_t^{(m)}}^{W})^{(i+1)} = (\mathrm{v}_{w_t^{(m)}}^{W})^{(i)} + \gamma \left[(Y^{(m)} - \sigma(\mathrm{v}_{w_t^{(m)}}^{W} \cdot h_c^{(m)}))h_c^{(m)} - \beta \mathrm{v}_{w_t^{(m)}}^{W} \right] \tag{29}$$

3. Context Word Vectors :

$$(\mathrm{v}_{w_{t+j}^{(m)}}^{W})^{(i+1)} = (\mathrm{v}_{w_{t+j}^{(m)}}^{W})^{(i)} + \gamma \left[(Y^{(m)} - \sigma(\mathrm{v}_{w_t^{(m)}}^{W} \cdot h_c^{(m)}))\lambda_{t+j}\mathrm{v}_{w_t^{(m)}}^{W} - \beta \mathrm{v}_{w_{t+j}^{(m)}}^{W} \right] \tag{30}$$

Update rule for Parameters

1. Document Vector :

$$(\mathrm{v}_{d_i^{(m)}}^{D})^{(i+1)} = (\mathrm{v}_{d_i^{(m)}}^{D})^{(i)} + \gamma \left[(Y^{(m)} - \sigma(\mathrm{v}_{w_t^{(m)}}^{W} \cdot h_c^{(m)}))\mathrm{v}_{w_t^{(m)}}^{W} - \beta \mathrm{v}_{d_i^{(m)}}^{D} \right] \tag{28}$$

2. Middle Word Vector :

$$(\mathrm{v}_{w_t^{(m)}}^{W})^{(i+1)} = (\mathrm{v}_{w_t^{(m)}}^{W})^{(i)} + \gamma \left[(Y^{(m)} - \sigma(\mathrm{v}_{w_t^{(m)}}^{W} \cdot h_c^{(m)}))h_c^{(m)} - \beta \mathrm{v}_{w_t^{(m)}}^{W} \right] \tag{29}$$

3. Context Word Vectors :

$$(\mathrm{v}_{w_{t+j}^{(m)}}^{W})^{(i+1)} = (\mathrm{v}_{w_{t+j}^{(m)}}^{W})^{(i)} + \gamma \left[(Y^{(m)} - \sigma(\mathrm{v}_{w_t^{(m)}}^{W} \cdot h_c^{(m)}))\lambda_{t+j}\mathrm{v}_{w_t^{(m)}}^{W} - \beta \mathrm{v}_{w_{t+j}^{(m)}}^{W} \right] \tag{30}$$

4. Neural Network Weights :

$$\lambda_{t+j}^{(i+1)} = \lambda_{t+j}^{(i)} + \gamma \left[(Y^{(m)} - \sigma(\mathrm{v}_{w_t^{(m)}}^{W} \cdot h_c^{(m)}))(\mathrm{v}_{w_t^{(m)}}^{W} \cdot \mathrm{v}_{w_{t+j}^{(m)}}^{W}) - \beta \lambda_{t+j} \right] \tag{31}$$

Algorithm for learning Document Representations

1: **Input:** D, k, c, n, β, γ, *epochs*
2: **Output:** Document Vectors D, Word Vectors W

Algorithm for learning Document Representations

Input: D, k, c, n, β, γ, *epochs*
Output: Document Vectors D, Word Vectors W
$V \leftarrow Extractfrom(D)$
$\mathrm{D} \leftarrow random(\mathbb{R}^{k \times |D|})$
$\mathrm{W} \leftarrow random(\mathbb{R}^{k \times |V|})$
$\mathcal{T} \leftarrow Extractfrom(D, c, n)$ ▷ $|\mathcal{T}| = M + nM$
$\Lambda \leftarrow \mathbf{1}^{2c}$ ▷ $2c$-sized vector of 1s

Algorithm for learning Document Representations

Input: D, k, c, n, β, γ, *epochs*
Output: Document Vectors D, Word Vectors W
$V \leftarrow Extractfrom(D)$
$\mathrm{D} \leftarrow random(\mathbb{R}^{k \times |D|})$
$\mathrm{W} \leftarrow random(\mathbb{R}^{k \times |V|})$
$\mathcal{T} \leftarrow Extractfrom(D, c, n)$ ▷ $|\mathcal{T}| = M + nM$
$\Lambda \leftarrow \mathbf{1}^{2c}$ ▷ $2c$-sized vector of 1s
while *epochs* ≥ 1 **do**
    for all $\{d_i, w_{t-c}, \ldots, w_{t+c}, Y\} \in \mathcal{T}$ **do**
        $h_c \leftarrow \mathrm{v}^D_{d_i} + \lambda_{t-c}\mathrm{v}^W_{w_{t-c}} + \ldots + \lambda_{t+c}\mathrm{v}^W_{w_{t+c}}$

Algorithm for learning Document Representations

Input: D, k, c, n, β, γ, *epochs*
Output: Document Vectors D, Word Vectors W
$V \leftarrow Extractfrom(D)$
$\mathrm{D} \leftarrow random(\mathbb{R}^{k \times |D|})$
$\mathrm{W} \leftarrow random(\mathbb{R}^{k \times |V|})$
$\mathcal{T} \leftarrow Extractfrom(D, c, n)$ ▷ $|\mathcal{T}| = M + nM$
$\Lambda \leftarrow \mathbf{1}^{2c}$ ▷ $2c$-sized vector of 1s
while *epochs* ≥ 1 **do**
    for all $\{d_i, w_{t-c}, \ldots, w_{t+c}, Y\} \in \mathcal{T}$ **do**
        $h_c \leftarrow \mathrm{v}^D_{d_i} + \lambda_{t-c}\mathrm{v}^W_{w_{t-c}} + \ldots + \lambda_{t+c}\mathrm{v}^W_{w_{t+c}}$
        $\mathrm{v}^D_{d_i} \leftarrow \mathrm{v}^D_{d_i} + \gamma\left[(Y - \sigma(\mathrm{v}^W_{w_t} \cdot h_c))\mathrm{v}^W_{w_t} - \beta\mathrm{v}^D_{d_i}\right]$

Algorithm for learning Document Representations

Input: D, k, c, n, β, γ, *epochs*
Output: Document Vectors D, Word Vectors W
$V \leftarrow Extractfrom(D)$
$\mathrm{D} \leftarrow random(\mathbb{R}^{k \times |D|})$
$\mathrm{W} \leftarrow random(\mathbb{R}^{k \times |V|})$
$\mathcal{T} \leftarrow Extractfrom(D, c, n)$ ▷ $|\mathcal{T}| = M + nM$
$\Lambda \leftarrow \mathbf{1}^{2c}$ ▷ $2c$-sized vector of 1s
while *epochs* ≥ 1 **do**
    for all $\{d_i, w_{t-c}, \ldots, w_{t+c}, Y\} \in \mathcal{T}$ **do**
        $h_c \leftarrow \mathrm{v}^D_{d_i} + \lambda_{t-c}\mathrm{v}^W_{w_{t-c}} + \ldots + \lambda_{t+c}\mathrm{v}^W_{w_{t+c}}$
        $\mathrm{v}^D_{d_i} \leftarrow \mathrm{v}^D_{d_i} + \gamma\left[(Y - \sigma(\mathrm{v}^W_{w_t} \cdot h_c))\mathrm{v}^W_{w_t} - \beta\mathrm{v}^D_{d_i}\right]$
        $\mathrm{v}^W_{w_t} \leftarrow \mathrm{v}^W_{w_t} + \gamma\left[(Y - \sigma(\mathrm{v}^W_{w_t} \cdot h_c))h_c - \beta\mathrm{v}^W_{w_t}\right]$

Algorithm for learning Document Representations

Input: D, k, c, n, β, γ, *epochs*
Output: Document Vectors D, Word Vectors W
$V \leftarrow \mathit{Extractfrom}(D)$
$\mathrm{D} \leftarrow \mathit{random}(\mathbb{R}^{k \times |D|})$
$\mathrm{W} \leftarrow \mathit{random}(\mathbb{R}^{k \times |V|})$
$\mathcal{T} \leftarrow \mathit{Extractfrom}(D, c, n)$ ▷ $|\mathcal{T}| = M + nM$
$\Lambda \leftarrow \mathbf{1}^{2c}$ ▷ $2c$-sized vector of 1s
while *epochs* ≥ 1 **do**
    for all $\{d_i, w_{t-c}, \ldots, w_{t+c}, Y\} \in \mathcal{T}$ **do**
        $h_c \leftarrow \mathrm{v}^D_{d_i} + \lambda_{t-c}\mathrm{v}^W_{w_{t-c}} + \ldots + \lambda_{t+c}\mathrm{v}^W_{w_{t+c}}$
        $\mathrm{v}^D_{d_i} \leftarrow \mathrm{v}^D_{d_i} + \gamma\left[(Y - \sigma(\mathrm{v}^W_{w_t} \cdot h_c))\mathrm{v}^W_{w_t} - \beta\mathrm{v}^D_{d_i}\right]$
        $\mathrm{v}^W_{w_t} \leftarrow \mathrm{v}^W_{w_t} + \gamma\left[(Y - \sigma(\mathrm{v}^W_{w_t} \cdot h_c))h_c - \beta\mathrm{v}^W_{w_t}\right]$
        for all $j \in \{t-c, \ldots, t-1, t+1, \ldots, t+c\}$ **do**
            $\mathrm{v}^W_{w_{t+j}} \leftarrow \mathrm{v}^W_{w_{t+j}} + \gamma\left[(Y - \sigma(\mathrm{v}^W_{w_t} \cdot h_c))\lambda_{t+j}\mathrm{v}^W_{w_t} - \beta\mathrm{v}^W_{w_{t+j}}\right]$

Algorithm for learning Document Representations

Input: D, k, c, n, β, γ, *epochs*
Output: Document Vectors D, Word Vectors W
$V \leftarrow Extractfrom(D)$
$\mathrm{D} \leftarrow random(\mathbb{R}^{k \times |D|})$
$\mathrm{W} \leftarrow random(\mathbb{R}^{k \times |V|})$
$\mathcal{T} \leftarrow Extractfrom(D, c, n)$ ▷ $|\mathcal{T}| = M + nM$
$\Lambda \leftarrow \mathbf{1}^{2c}$ ▷ $2c$-sized vector of 1s
while *epochs* ≥ 1 **do**
  for all $\{d_i, w_{t-c}, \ldots, w_{t+c}, Y\} \in \mathcal{T}$ **do**
    $h_c \leftarrow \mathrm{v}^D_{d_i} + \lambda_{t-c}\mathrm{v}^W_{w_{t-c}} + \ldots + \lambda_{t+c}\mathrm{v}^W_{w_{t+c}}$
    $\mathrm{v}^D_{d_i} \leftarrow \mathrm{v}^D_{d_i} + \gamma\left[(Y - \sigma(\mathrm{v}^W_{w_t} \cdot h_c))\mathrm{v}^W_{w_t} - \beta\mathrm{v}^D_{d_i}\right]$
    $\mathrm{v}^W_{w_t} \leftarrow \mathrm{v}^W_{w_t} + \gamma\left[(Y - \sigma(\mathrm{v}^W_{w_t} \cdot h_c))h_c - \beta\mathrm{v}^W_{w_t}\right]$
    for all $j \in \{t-c, \ldots, t-1, t+1, \ldots, t+c\}$ **do**
      $\mathrm{v}^W_{w_{t+j}} \leftarrow \mathrm{v}^W_{w_{t+j}} + \gamma\left[(Y - \sigma(\mathrm{v}^W_{w_t} \cdot h_c))\lambda_{t+j}\mathrm{v}^W_{w_t} - \beta\mathrm{v}^W_{w_{t+j}}\right]$
      $\lambda_{t+j} \leftarrow \lambda_{t+j} + \gamma\left[(Y - \sigma(\mathrm{v}^W_{w_t} \cdot h_c))(\mathrm{v}^W_{w_t} \cdot \mathrm{v}^W_{w_{t+j}}) - \beta\lambda_{t+j}\right]$

Algorithm for learning Document Representations

Input: D, k, c, n, β, γ, *epochs*
Output: Document Vectors D, Word Vectors W
$V \leftarrow \mathit{Extractfrom}(D)$
$\mathrm{D} \leftarrow \mathit{random}(\mathbb{R}^{k \times |D|})$
$\mathrm{W} \leftarrow \mathit{random}(\mathbb{R}^{k \times |V|})$
$\mathcal{T} \leftarrow \mathit{Extractfrom}(D, c, n)$ ▷ $|\mathcal{T}| = M + nM$
$\Lambda \leftarrow \mathbf{1}^{2c}$ ▷ $2c$-sized vector of 1s
while *epochs* ≥ 1 **do**
  for all $\{d_i, w_{t-c}, \ldots, w_{t+c}, Y\} \in \mathcal{T}$ **do**
    $h_c \leftarrow \mathrm{v}^D_{d_i} + \lambda_{t-c}\mathrm{v}^W_{w_{t-c}} + \ldots + \lambda_{t+c}\mathrm{v}^W_{w_{t+c}}$
    $\mathrm{v}^D_{d_i} \leftarrow \mathrm{v}^D_{d_i} + \gamma\left[(Y - \sigma(\mathrm{v}^W_{w_t} \cdot h_c))\mathrm{v}^W_{w_t} - \beta\mathrm{v}^D_{d_i}\right]$
    $\mathrm{v}^W_{w_t} \leftarrow \mathrm{v}^W_{w_t} + \gamma\left[(Y - \sigma(\mathrm{v}^W_{w_t} \cdot h_c))h_c - \beta\mathrm{v}^W_{w_t}\right]$
    for all $j \in \{t-c, \ldots, t-1, t+1, \ldots, t+c\}$ **do**
      $\mathrm{v}^W_{w_{t+j}} \leftarrow \mathrm{v}^W_{w_{t+j}} + \gamma\left[(Y - \sigma(\mathrm{v}^W_{w_t} \cdot h_c))\lambda_{t+j}\mathrm{v}^W_{w_t} - \beta\mathrm{v}^W_{w_{t+j}}\right]$
      $\lambda_{t+j} \leftarrow \lambda_{t+j} + \gamma\left[(Y - \sigma(\mathrm{v}^W_{w_t} \cdot h_c))(\mathrm{v}^W_{w_t} \cdot \mathrm{v}^W_{w_{t+j}}) - \beta\lambda_{t+j}\right]$
    epochs $\leftarrow$ *epochs* $- 1$
return D, W

Hyper-parameters of the Model

1. Embedding Dimensionality (k)

Hyper-parameters of the Model

1. Embedding Dimensionality (k)
2. Window Size (c)

Hyper-parameters of the Model

1. Embedding Dimensionality (k)
2. Window Size (c)
3. Number of Negative Samples (n)

Hyper-parameters of the Model

1. Embedding Dimensionality (k)
2. Window Size (c)
3. Number of Negative Samples (n)
4. Number of Epochs (*epochs*)

Hyper-parameters of the Model

1. Embedding Dimensionality (k)
2. Window Size (c)
3. Number of Negative Samples (n)
4. Number of Epochs (*epochs*)
5. Learning Rate (γ)

Hyper-parameters of the Model

1. Embedding Dimensionality (k)
2. Window Size (c)
3. Number of Negative Samples (n)
4. Number of Epochs (*epochs*)
5. Learning Rate (γ)
6. Regularization Constant (β)

Document Categorization using Logistic Regression

Given,

1. Set of documents, $D = \{d_1, \ldots, d_{|D|}\}$

Document Categorization using Logistic Regression

Given,

1. Set of documents, $D = \{d_1, \ldots, d_{|D|}\}$
2. Set of categories, $C = \{c_1, \ldots, c_{|C|}\}$

Document Categorization using Logistic Regression

Given,

1. Set of documents, $D = \{d_1, \ldots, d_{|D|}\}$
2. Set of categories, $C = \{c_1, \ldots, c_{|C|}\}$
3. Training Data, $\mathcal{T} = \{d_i^{(m)}, c_j^{(m)}, y^{(m)}\}_{m=1}^{m=T}$, $y^{(m)} \in \{0, 1\}$

Document Categorization using Logistic Regression

Given,

1. Set of documents, $D = \{d_1, \ldots, d_{|D|}\}$
2. Set of categories, $C = \{c_1, \ldots, c_{|C|}\}$
3. Training Data, $\mathcal{T} = \{d_i^{(m)}, c_j^{(m)}, y^{(m)}\}_{m=1}^{m=T}$, $y^{(m)} \in \{0, 1\}$

The task is to assign categories to a new document d_x
To model document category relation, we

Document Categorization using Logistic Regression

Given,

1. Set of documents, $D = \{d_1, \ldots, d_{|D|}\}$
2. Set of categories, $C = \{c_1, \ldots, c_{|C|}\}$
3. Training Data, $\mathcal{T} = \{d_i^{(m)}, c_j^{(m)}, y^{(m)}\}_{m=1}^{m=T}$, $y^{(m)} \in \{0, 1\}$

The task is to assign categories to a new document d_x
To model document category relation, we

1. Represent each $c_i \in C$ using $\mathrm{v}_{c_i}^{C} \in \mathbb{R}^k$

Document Categorization using Logistic Regression

Given,

1. Set of documents, $D = \{d_1, \ldots, d_{|D|}\}$
2. Set of categories, $C = \{c_1, \ldots, c_{|C|}\}$
3. Training Data, $\mathcal{T} = \{d_i^{(m)}, c_j^{(m)}, y^{(m)}\}_{m=1}^{m=T}$, $y^{(m)} \in \{0, 1\}$

The task is to assign categories to a new document d_x
To model document category relation, we

1. Represent each $c_i \in C$ using $\mathrm{v}_{c_i}^{C} \in \mathbb{R}^k$
2. Learn a probabilistic logistic classifier to assign categories

Logistic Classifier for Categorization

Given document category pair, $\{d_i, c_j\}$,

Logistic Classifier for Categorization

Given document category pair, $\{d_i, c_j\}$,

- We build a probabilistic logistic classifier to predict the label y

Logistic Classifier for Categorization

Given document category pair, $\{d_i, c_j\}$,

- We build a probabilistic logistic classifier to predict the label y

$$P(y = 1 | d_i, c_j, \mathrm{D}, \mathrm{C}) = \sigma(\mathrm{v}_{d_i}^{D} \cdot \mathrm{v}_{c_j}^{C}) \tag{32}$$

Logistic Classifier for Categorization

Given document category pair, $\{d_i, c_j\}$,

- We build a probabilistic logistic classifier to predict the label y

$$P(y = 1|d_i, c_j, \mathrm{D}, \mathrm{C}) = \sigma(\mathrm{v}_{d_i}^{D} \cdot \mathrm{v}_{c_j}^{C}) \tag{32}$$

$$P(y = 0|d_i, c_j, \mathrm{D}, \mathrm{C}) = 1 - \sigma(\mathrm{v}_{d_i}^{D} \cdot \mathrm{v}_{c_j}^{C}) \tag{33}$$

Logistic Classifier for Categorization

Given document category pair, $\{d_i, c_j\}$,

- We build a probabilistic logistic classifier to predict the label y

$$P(y = 1|d_i, c_j, \mathrm{D}, \mathrm{C}) = \sigma(\mathrm{v}_{d_i}^D \cdot \mathrm{v}_{c_j}^C) \tag{32}$$

$$P(y = 0|d_i, c_j, \mathrm{D}, \mathrm{C}) = 1 - \sigma(\mathrm{v}_{d_i}^D \cdot \mathrm{v}_{c_j}^C) \tag{33}$$

$$P(y|d_i, c_j, \mathrm{D}, \mathrm{C}) = \sigma(\mathrm{v}_{d_i}^D \cdot \mathrm{v}_{c_j}^C)^y (1 - \sigma(\mathrm{v}_{d_i}^D \cdot \mathrm{v}_{c_j}^C))^{1-y} \tag{34}$$

Logistic Classifier for Categorization

Given document category pair, $\{d_i, c_j\}$,

- We build a probabilistic logistic classifier to predict the label y

$$P(y = 1|d_i, c_j, \mathrm{D}, \mathrm{C}) = \sigma(\mathrm{v}_{d_i}^D \cdot \mathrm{v}_{c_j}^C) \tag{32}$$

$$P(y = 0|d_i, c_j, \mathrm{D}, \mathrm{C}) = 1 - \sigma(\mathrm{v}_{d_i}^D \cdot \mathrm{v}_{c_j}^C) \tag{33}$$

$$P(y|d_i, c_j, \mathrm{D}, \mathrm{C}) = \sigma(\mathrm{v}_{d_i}^D \cdot \mathrm{v}_{c_j}^C)^y (1 - \sigma(\mathrm{v}_{d_i}^D \cdot \mathrm{v}_{c_j}^C))^{1-y} \tag{34}$$

$$\log P(y|d_i, c_j, \mathrm{D}, \mathrm{C}) = y \log \sigma(\mathrm{v}_{d_i}^D \cdot \mathrm{v}_{c_j}^C) + (1 - y) \log(1 - \sigma(\mathrm{v}_{d_i}^D \cdot \mathrm{v}_{c_j}^C)) \tag{35}$$

Learning Category Embeddings

Given the training data $\mathcal{T} = \{d_i^{(m)}, c_j^{(m)}, y^{(m)}\}_{m=1}^{m=T}$, learn category embeddings ($\Theta = C$) by maximizing log-likelihood of training data

---

$P_{\mathrm{D,C}}(y_m = y^{(m)})$ is a shorthand notation for $P(y_m = y^{(m)}|d_i, c_j, \mathrm{D}, \mathrm{C})$
y_m is the predicted label

Learning Category Embeddings

Given the training data $\mathcal{T} = \{d_i^{(m)}, c_j^{(m)}, y^{(m)}\}_{m=1}^{m=T}$, learn category embeddings ($\Theta = C$) by maximizing log-likelihood of training data

$$\hat{\Theta} = \arg\max_{\Theta} \; l(\mathcal{T}, \Theta) \tag{36}$$

$$l(\mathcal{T}, \Theta) = \sum_{m=1}^{T} \log P_{\mathrm{D,C}}(y_m = y^{(m)}) \tag{37}$$

$P_{\mathrm{D,C}}(y_m = y^{(m)})$ is a shorthand notation for $P(y_m = y^{(m)} | d_i, c_j, \mathrm{D}, \mathrm{C})$
y_m is the predicted label

Learning Category Embeddings

Given the training data $\mathcal{T} = \{d_i^{(m)}, c_j^{(m)}, y^{(m)}\}_{m=1}^{m=T}$, learn category embeddings ($\Theta = C$) by maximizing log-likelihood of training data

$$\hat{\Theta} = \arg\max_{\Theta} \; l(\mathcal{T}, \Theta) \tag{36}$$

$$l(\mathcal{T}, \Theta) = \sum_{m=1}^{T} \log P_{\mathrm{D,C}}(y_m = y^{(m)}) \tag{37}$$

Similar to learning document embeddings, category embeddings updates are given by,

$$(\mathrm{v}^{C}_{c_j^{(m)}})^{(i+1)} = (\mathrm{v}^{C}_{c_j^{(m)}})^{(i)} + \gamma \left[(y^{(m)} - \sigma(\mathrm{v}^{D}_{d_i^{(m)}} \cdot (\mathrm{v}^{C}_{c_j^{(m)}})^{(i)}) \mathrm{v}^{D}_{d_i^{(m)}} - \beta (\mathrm{v}^{C}_{c_j^{(m)}})^{(i)} \right] \tag{38}$$

$P_{\mathrm{D,C}}(y_m = y^{(m)})$ is a shorthand notation for $P(y_m = y^{(m)} | d_i, c_j, \mathrm{D}, \mathrm{C})$
y_m is the predicted label

Algorithm for learning Document Representations

Algorithm 1 Learning Category Vector Representations

Input: D, C, $\mathcal{T}$, k, β, γ
Output: Category Vectors C
$\mathrm{C} \leftarrow \mathit{random}(\mathbb{R}^{k \times |C|})$
while not converged **do**
    for all $\{d_i, c_j, y\} \in \mathcal{T}$ **do**
        $$\mathrm{v}_{c_j}^{C} \leftarrow \mathrm{v}_{c_j}^{C} + \gamma \left[(y - \sigma(\mathrm{v}_{d_i}^{D} \cdot \mathrm{v}_{c_j}^{C}))\mathrm{v}_{d_i}^{D} - \beta \mathrm{v}_{c_j}^{C} \right]$$
return C

Advantages of Multinomial Logistic Regression Algorithm

1. Learns embeddings for categories in the same space as words and documents

Advantages of Multinomial Logistic Regression Algorithm

1. Learns embeddings for categories in the same space as words and documents
2. Though learns multiple category vectors, exploits the low-rank structure

Advantages of Multinomial Logistic Regression Algorithm

1. Learns embeddings for categories in the same space as words and documents
2. Though learns multiple category vectors, exploits the low-rank structure
3. Easy incorporation of additional relational data of documents for better categorization as shown in Gupta and Singh [5]

Advantages of Multinomial Logistic Regression Algorithm

1. Learns embeddings for categories in the same space as words and documents
2. Though learns multiple category vectors, exploits the low-rank structure
3. Easy incorporation of additional relational data of documents for better categorization as shown in Gupta and Singh [5]
4. Usage of SGD makes algorithm completely online

References

[1] Y. Bengio and J.-S. Senecal. Adaptive importance sampling to accelerate training of a neural probabilistic language model. *Neural Networks, IEEE Transactions on*, 19(4):713–722, 2008.

[2] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155, 2003.

[3] Y. Bengio, J.-S. Senécal, et al. Quick training of probabilistic neural nets by importance sampling. In *AISTATS Conference*, 2003.

[4] E. Grefenstette, G. Dinu, Y.-Z. Zhang, M. Sadrzadeh, and M. Baroni. Multi-step regression learning for compositional distributional semantics. *arXiv preprint arXiv:1301.6939*, 2013.

[5] N. Gupta and S. Singh. Collectively embedding multi-relational data for predicting user preferences. *arXiv preprint arXiv:1504.06165*, 2015.

[6] M. U. Gutmann and A. Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *The Journal of Machine Learning Research*, 13(1):307–361, 2012.

[7] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[8] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed