IN5550: Neural Methods in Natural Language Processing
Sub-lecture 4.1
One-hot and dense representations

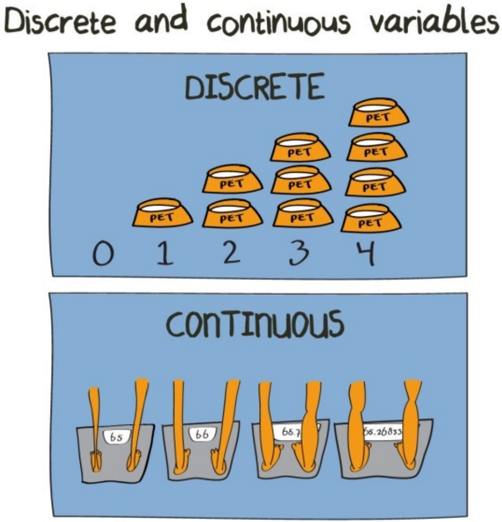
Andrey Kutuzov

University of Oslo

14 February 2023

Contents

How to make the world continuous?



(by Luis Fok on Quora)

Representations

- In the obligatory 1, you train neural document classifiers...

Dense Representations of Linguistic Features

Representations

- In the obligatory 1, you train neural document classifiers...
- ...using bags of words as features.

Dense Representations of Linguistic Features

Representations

- In the obligatory 1, you train neural document classifiers...
- ...using bags of words as features.
- Documents are represented as sparse vocabulary vectors.

Dense Representations of Linguistic Features

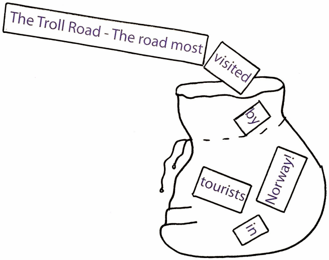
Representations

- In the obligatory 1, you train neural document classifiers...
- ...using bags of words as features.
- Documents are represented as sparse vocabulary vectors.
- Core elements of this representation are words,

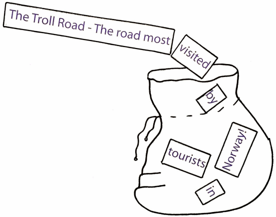
Dense Representations of Linguistic Features

Representations

- In the obligatory 1, you train neural document classifiers...
- ...using bags of words as features.
- Documents are represented as sparse vocabulary vectors.
- Core elements of this representation are words,
- and they are in turn represented with one-hot vectors.

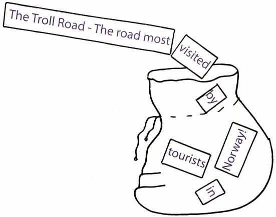


One-hot representations: let's recall



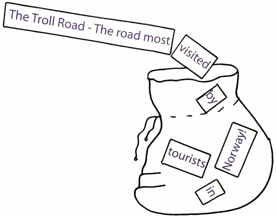
- BOW feature vector of the document i can be interpreted as a sum of one-hot vectors ($\boldsymbol{o}$) for each token in it:

One-hot representations: let's recall



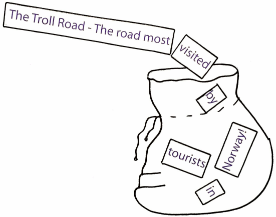
- BOW feature vector of the document *i* can be interpreted as a sum of one-hot vectors ($\boldsymbol{o}$) for each token in it:
  - Vocabulary *V* from the picture above contains 10 words (lowercased): *['-', 'by', 'in', 'most', 'norway', 'road', 'the', 'tourists', 'troll', 'visited']*.

One-hot representations: let's recall



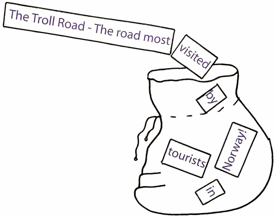
- BOW feature vector of the document *i* can be interpreted as a sum of one-hot vectors ($\boldsymbol{o}$) for each token in it:
  - Vocabulary *V* from the picture above contains 10 words (lowercased): *['-', 'by', 'in', 'most', 'norway', 'road', 'the', 'tourists', 'troll', 'visited'].*
  - $\boldsymbol{o}^0 = [0, 0, 0, 0, 0, 0, 1, 0, 0, 0]$ ('The')

One-hot representations: let's recall



- BOW feature vector of the document i can be interpreted as a sum of one-hot vectors ($\boldsymbol{o}$) for each token in it:
  - Vocabulary V from the picture above contains 10 words (lowercased): *['-', 'by', 'in', 'most', 'norway', 'road', 'the', 'tourists', 'troll', 'visited'].*
  - $\boldsymbol{o}^0 = [0, 0, 0, 0, 0, 0, 1, 0, 0, 0]$ ('The')
  - $\boldsymbol{o}^1 = [0, 0, 0, 0, 0, 0, 0, 0, 1, 0]$ ('Troll')
  - etc...

One-hot representations: let's recall



- BOW feature vector of the document i can be interpreted as a sum of one-hot vectors ($\boldsymbol{o}$) for each token in it:
  - Vocabulary V from the picture above contains 10 words (lowercased): *['-', 'by', 'in', 'most', 'norway', 'road', 'the', 'tourists', 'troll', 'visited'].*
  - $\boldsymbol{o}^0 = [0, 0, 0, 0, 0, 0, 1, 0, 0, 0]$ ('The')
  - $\boldsymbol{o}^1 = [0, 0, 0, 0, 0, 0, 0, 0, 1, 0]$ ('Troll')
  - etc...
  - $\boldsymbol{i} = [1, 1, 1, 1, 1, 2, 2, 1, 1, 1]$ ('*the*' and '*road*' occurred 2 times)

One-hot representations: let's recall

- The network is trained on words represented with integer identifiers:
  - '*the*' is the word number 6 in the vocabulary
  - '*most*' is the word number 3 in the vocabulary
  - '*visited*' is the word number 9 in the vocabulary
  - etc.

One-hot representations: let's recall

- The network is trained on words represented with integer identifiers:
  - '*the*' is the word number 6 in the vocabulary
  - '*most*' is the word number 3 in the vocabulary
  - '*visited*' is the word number 9 in the vocabulary
  - etc.
- Such features are discrete (categorical).
  - a.k.a. one-hot.

One-hot representations: let's recall

- The network is trained on words represented with integer identifiers:
  - '*the*' is the word number 6 in the vocabulary
  - '*most*' is the word number 3 in the vocabulary
  - '*visited*' is the word number 9 in the vocabulary
  - etc.
- Such features are discrete (categorical).
  - a.k.a. one-hot.
- Each word is a feature on its own, completely independent from other words.

One-hot representations: let's recall

- The network is trained on words represented with integer identifiers:
  - '*the*' is the word number 6 in the vocabulary
  - '*most*' is the word number 3 in the vocabulary
  - '*visited*' is the word number 9 in the vocabulary
  - etc.
- Such features are discrete (categorical).
  - a.k.a. one-hot.
- Each word is a feature on its own, completely independent from other words.
- Other NLP tasks: categorical features for PoS tags, dependency labels, etc.

One-hot representations: let's recall

Why discrete features might be bad?

- Features for words are extremely sparse:

One-hot representations: let's recall

Why discrete features might be bad?

- Features for words are extremely sparse:
  - the feature '*word form*' can take any of tens or hundreds of thousands categorical values...

One-hot representations: let's recall

Why discrete features might be bad?

- Features for words are extremely sparse:
  - the feature '*word form*' can take any of tens or hundreds of thousands categorical values...
  - ...each absolutely unique and not related to each other.

One-hot representations: let's recall

Why discrete features might be bad?

- Features for words are extremely sparse:
  - the feature '*word form*' can take any of tens or hundreds of thousands categorical values...
  - ...each absolutely unique and not related to each other.
- Classifiers have to learn weight matrices with $dim = |V|$.

One-hot representations: let's recall

Why discrete features might be bad?

- Features for words are extremely sparse:
  - the feature '*word form*' can take any of tens or hundreds of thousands categorical values...
  - ...each absolutely unique and not related to each other.
- Classifiers have to learn weight matrices with $dim = |V|$.
- Not efficient:
  - A 50-words text is $\boldsymbol{x} \in \mathbb{R}^{100000}$, because 100K words in our vocabulary!

One-hot representations: let's recall

Why discrete features might be bad?

- Features for words are extremely sparse:
  - the feature '*word form*' can take any of tens or hundreds of thousands categorical values...
  - ...each absolutely unique and not related to each other.
- Classifiers have to learn weight matrices with $dim = |V|$.
- Not efficient:
  - A 50-words text is $\boldsymbol{x} \in \mathbb{R}^{100000}$, because 100K words in our vocabulary!
- A bit easier for other linguistic entities (parts of speech, etc)...

One-hot representations: let's recall

Why discrete features might be bad?

- Features for words are extremely sparse:
  - the feature '*word form*' can take any of tens or hundreds of thousands categorical values...
  - ...each absolutely unique and not related to each other.
- Classifiers have to learn weight matrices with $dim = |V|$.
- Not efficient:
  - A 50-words text is $\boldsymbol{x} \in \mathbb{R}^{100000}$, because 100K words in our vocabulary!
- A bit easier for other linguistic entities (parts of speech, etc)...
- ...but their feature combinations yield millions of resulting features.

One-hot representations: let's recall

Why discrete features might be bad?

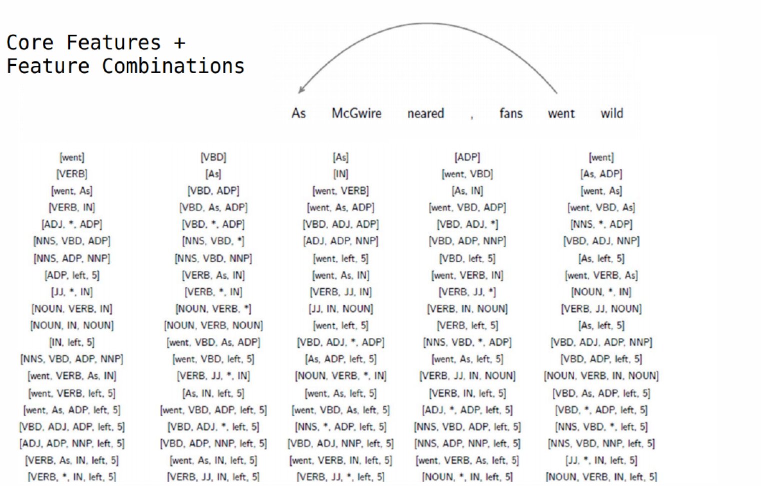
- Features for words are extremely sparse:
  - the feature '*word form*' can take any of tens or hundreds of thousands categorical values...
  - ...each absolutely unique and not related to each other.
- Classifiers have to learn weight matrices with $dim = |V|$.
- Not efficient:
  - A 50-words text is $\boldsymbol{x} \in \mathbb{R}^{100000}$, because 100K words in our vocabulary!
- A bit easier for other linguistic entities (parts of speech, etc)...
- ...but their feature combinations yield millions of resulting features.
- It's very difficult to learn good weights for them all.

One-hot representations: let's recall

Why discrete features might be bad?

- Features for words are extremely sparse:
  - the feature '*word form*' can take any of tens or hundreds of thousands categorical values...
  - ...each absolutely unique and not related to each other.
- Classifiers have to learn weight matrices with $dim = |V|$.
- Not efficient:
  - A 50-words text is $\boldsymbol{x} \in \mathbb{R}^{100000}$, because 100K words in our vocabulary!
- A bit easier for other linguistic entities (parts of speech, etc)...
- ...but their feature combinations yield millions of resulting features.
- It's very difficult to learn good weights for them all.
- Feature extraction step haunted NLP practitioners for several decades.

One-hot representations: let's recall



Example from slides of Rush and Petrov (2012)

Feature model for parsing

'Is the 1st word to the right *wild*, and the 3rd word to the left a *verb*?'

One-hot representations: let's recall

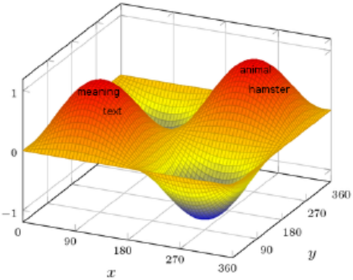
We can do better

- Is there a way to avoid using multitudes of discrete categorical features?

One-hot representations: let's recall

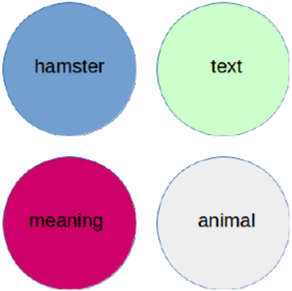
We can do better

- Is there a way to avoid using multitudes of discrete categorical features?
- Yes.
- Use dense continuous features.

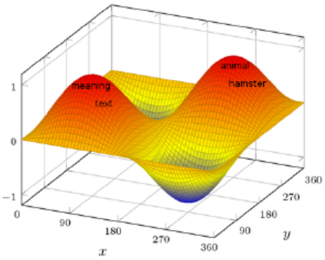


Dense representations (embeddings)

Discrete representations

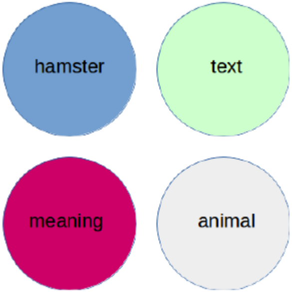


Continuous representations

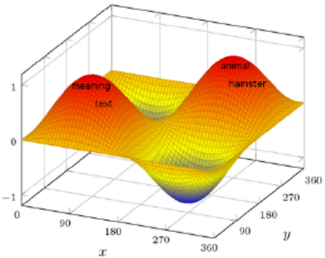


Dense representations (embeddings)

Discrete representations



Continuous representations



- We would like linguistic entities to be represented with some meaningful ‘coordinates’.
- It would allow our models to understand whether entities (for example, words) are more or less similar with respect to the current task at hand.

Dense representations (embeddings)

Vectors as coordinates

- A vector is a sequence or an array of n real values:
  - $[0, 1, 2, 4]$ is a vector with 4 components/entries ($\in \mathbb{R}^4$);
  - $[200, 300, 1]$ is a vector with 3 components/entries ($\in \mathbb{R}^3$);

Dense representations (embeddings)

Vectors as coordinates

- A vector is a sequence or an array of n real values:
  - $[0, 1, 2, 4]$ is a vector with 4 components/entries ($\in \mathbb{R}^4$);
  - $[200, 300, 1]$ is a vector with 3 components/entries ($\in \mathbb{R}^3$);
- Components can be viewed as coordinates in an n-dimensional space;

Dense representations (embeddings)

Vectors as coordinates

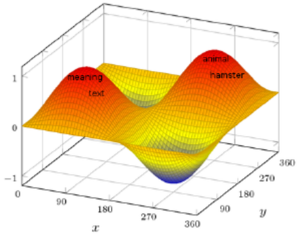
- A vector is a sequence or an array of n real values:
  - $[0, 1, 2, 4]$ is a vector with 4 components/entries ($\in \mathbb{R}^4$);
  - $[200, 300, 1]$ is a vector with 3 components/entries ($\in \mathbb{R}^3$);
- Components can be viewed as coordinates in an n-dimensional space;
- then a vector is a point in this space.

Dense representations (embeddings)

Vectors as coordinates

- A vector is a sequence or an array of n real values:
  - $[0, 1, 2, 4]$ is a vector with 4 components/entries ($\in \mathbb{R}^4$);
  - $[200, 300, 1]$ is a vector with 3 components/entries ($\in \mathbb{R}^3$);
- Components can be viewed as coordinates in an n-dimensional space;
- then a vector is a point in this space.

3-dimensional space:



Dense representations (embeddings)

Feature embeddings

- Say we have a vocabulary of size $|V|$;

Dense representations (embeddings)

Feature embeddings

- Say we have a vocabulary of size $|V|$;
- Let's embed these words into a d-dimensional space.

Dense representations (embeddings)

Feature embeddings

- Say we have a vocabulary of size $|V|$;
- Let's embed these words into a d-dimensional space.
- $d \ll |V|$
- e.g., $d = 100$ for a vocabulary of 100 000 words.

Dense representations (embeddings)

Feature embeddings

- Say we have a vocabulary of size $|V|$;
- Let's embed these words into a d-dimensional space.
- $d \ll |V|$
- e.g., $d = 100$ for a vocabulary of 100 000 words.
- Each word is associated with its own d-dimensional embedding vector;

Dense representations (embeddings)

Feature embeddings

- Say we have a vocabulary of size $|V|$;
- Let's embed these words into a d-dimensional space.
- $d \ll |V|$
- e.g., $d = 100$ for a vocabulary of 100 000 words.
- Each word is associated with its own d-dimensional embedding vector;
- These embeddings are part of θ;
- can be trained together with the rest of the network.

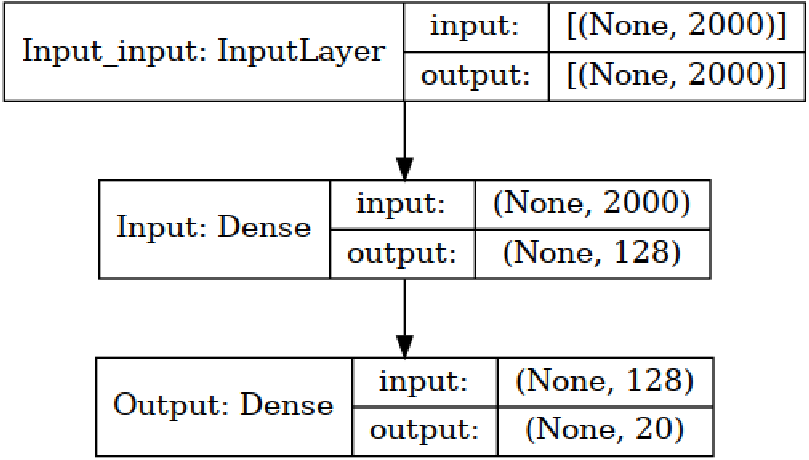
Dense representations (embeddings)

Feature embeddings

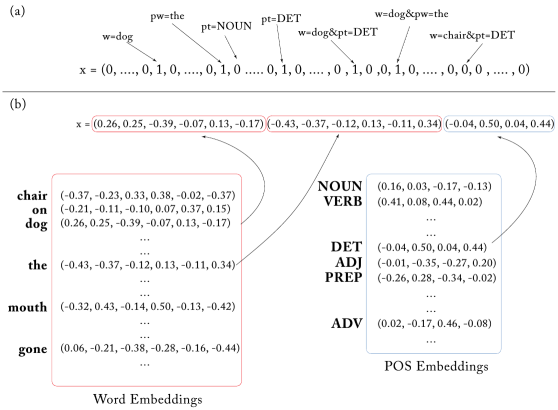
- Say we have a vocabulary of size $|V|$;
- Let's embed these words into a d-dimensional space.
- $d \ll |V|$
- e.g., $d = 100$ for a vocabulary of 100 000 words.
- Each word is associated with its own d-dimensional embedding vector;
- These embeddings are part of θ;
- can be trained together with the rest of the network.

Your models in the Obligatory 1 implicitly do this during the training.
This is representation learning.

Representation learning

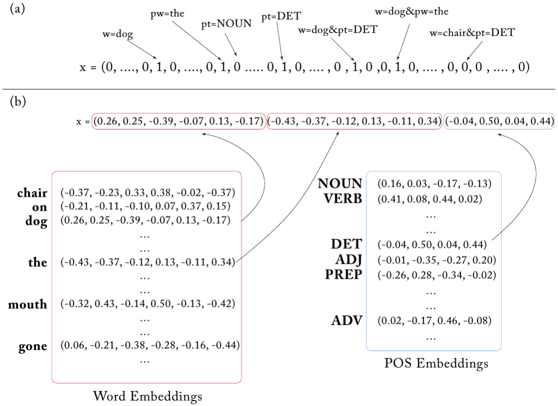


Sparse (a) and dense (b) feature representations for
'the_DET dog'
(part of speech tagging task)



Representation learning

Sparse (a) and dense (b) feature representations for
'the_DET dog'
(part of speech tagging task)



Q: what are the dimensionalities of word and PoS embeddings here?

Dense representations (embeddings)

Main benefits of continuous features

- Dimensionality of representations usually low (50-300).

Dense representations (embeddings)

Main benefits of continuous features

- Dimensionality of representations usually low (50-300).
- Feature vectors are dense, not sparse (more computationally efficient).

Dense representations (embeddings)

Main benefits of continuous features

- Dimensionality of representations usually low (50-300).
- Feature vectors are dense, not sparse (more computationally efficient).
- Generalization power: similar entities get similar embeddings (hopefully).
  - '*town*' vector is closer to '*city*' vector than to '*banana*' vector (semantics);

Dense representations (embeddings)

Main benefits of continuous features

- Dimensionality of representations usually low (50-300).
- Feature vectors are dense, not sparse (more computationally efficient).
- Generalization power: similar entities get similar embeddings (hopefully).
  - '*town*' vector is closer to '*city*' vector than to '*banana*' vector (semantics);
  - *NOUN* vector is closer to *ADJ* vector than to *VERB* vector (grammar);

Dense representations (embeddings)

Main benefits of continuous features

- Dimensionality of representations usually low (50-300).
- Feature vectors are dense, not sparse (more computationally efficient).
- Generalization power: similar entities get similar embeddings (hopefully).
  - '*town*' vector is closer to '*city*' vector than to '*banana*' vector (semantics);
  - *NOUN* vector is closer to *ADJ* vector than to *VERB* vector (grammar);
  - *iobj* vector is closer to *obj* vector than to *punct* vector (syntax).

Dense representations (embeddings)

Main benefits of continuous features

- Dimensionality of representations usually low (50-300).
- Feature vectors are dense, not sparse (more computationally efficient).
- Generalization power: similar entities get similar embeddings (hopefully).
  - '*town*' vector is closer to '*city*' vector than to '*banana*' vector (semantics);
  - *NOUN* vector is closer to *ADJ* vector than to *VERB* vector (grammar);
  - *iobj* vector is closer to *obj* vector than to *punct* vector (syntax).
- Same features in different positions can share statistical strength:
  - A token 2 words to the right and a token 2 words to the left can be one and the same word. Would be good for the model to use this knowledge.
  - Not important for the Obligatory 1, but can be critical for other tasks.

Demo web service

Word vectors for English and Norwegian online

You can try the *WebVectors* service developed by our Language Technology group

Word vectors for English and Norwegian online

You can try the *WebVectors* service developed by our Language Technology group

`http://vectors.nlpl.eu/explore/embeddings/`

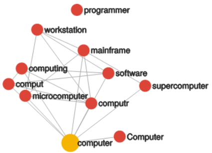
What words are related to «**computer**» in model «English Wikipedia»?

Word frequency

High Medium Low

1. comput r NOUN 0.7758
2. Computer NOUN 0.7665
3. microcomputer NOUN 0.6608
4. computing NOUN 0.6534
5. software NOUN 0.6387
6. mainframe NOUN 0.6324
7. comput NOUN 0.6314
8. supercomputer NOUN 0.6026
9. workstation NOUN 0.5905
10. programmer NOUN 0.5837

- *We show only the associates of the same part of speech as your query. All associates can be found at the Similar Words tab.*



0.6 Similarity threshold Show tags

This word in other models

- British National Corpus
- Google News
- English Gigaword
- Norsk Aviskorpus

Show the raw vector of «computer» in model