Deep Learning

Übung WS 23/24

Judith Nester (nester@uni-koeln.de)

14-12-2023

Recap

Bag of words

- Count words, disregard their order
- Document $\rightarrow$ vector $\rightarrow$ input for neural network
- scikit-learn: `CountVectorizer`

Recap

Bag of words

- Count words, disregard their order
- Document $\rightarrow$ vector $\rightarrow$ input for neural network
- scikit-learn: `CountVectorizer`

Overfitting

- Good performance on training data
- Less performance on real-world data
- No strict, deterministic decision
- Regularization: Add something to loss function
- Dropout: Randomly remove edges during training, force the network to create redundancies

Recap

Bag of words

- Count words, disregard their order
- Document → vector → input for neural network
- scikit-learn: `CountVectorizer`

Overfitting

- Good performance on training data
- Less performance on real-world data
- No strict, deterministic decision
- Regularization: Add something to loss function
- Dropout: Randomly remove edges during training, force the network to create redundancies

Exercise 7

Today

Section 1

Input Representation

Structured Data - Tables

- i.e. Titanic data set
- Objects (passengers) are described with the help of various properties (name, sex, ticket, age, cabin, ...)
- Number of features gives us the input shape
- Input that is not an integer is converted to an integer
- Input is a vector with the size of the feature count

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket
1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 2117
2	1	1	Cumings, Mrs. John Bradley (Flo	female	38	1	0	PC 17599
3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2
4	1	1	Futrelle, Mrs. Jacques Heath (Lil	female	35	1	0	
5	0	3	Allen, Mr. William Henry	male	35	0	0	
6	0	3	Moran, Mr. James	male		0	0	
7	0	1	McCarthy, Mr. Timothy J	male	54	0	0	
8	0	3	Palsson, Master. Gosta Leonard	male	2	3	1	
9	1	3	Johnson, Mrs. Oscar W (Elisabet	female	27	0	2	
10	1	2	Nasser, Mrs. Nicholas (Adele Ac	female	14	1	0	

• • •

• • •

Last Week: Bag of Words

- Disregards word order and semantic
- A vocabulary is established
- Words are counted in order of vocabulary in a text
- Size of vocabulary gives us the input shape
- Input is a vector with the size of the vocabulary

Abend	Adresse	also	auf	...	bei	beugen	Blume	Brief	...	und	Urlaub	...	Zaun	zeigen
0	0	2	4		3	0	0	1		7	2		0	3

...

Last Week: Bag of Words

» Makes things a lot easier

» But it's not how language works

Last Week: Bag of Words

- Makes things a lot easier
- But it's not how language works
- Example
  - »This remake was even greater than the original.«
  - »Deciding to make the script into a 3D movie led to an even greater failure.«

The problem with natural language

- Metaphors, ambiguities, synonyms, etc.
- "Words that occur in similar contexts tend to have similar meanings." (Jurafski 2021)
- Context is crucial for the meaning of a word

The problem with natural language

- Metaphors, ambiguities, synonyms, etc.
- "Words that occur in similar contexts tend to have similar meanings." (Jurafski 2021)
- Context is crucial for the meaning of a word

Is there a way to represent a word so that the meaning of the word within the context is not lost?

Section 2

Embeddings

Motivation

- An embedding is a mapping of words or documents to vectors
  - Things are ›embedded‹ in a vector space

Motivation

- An embedding is a mapping of words or documents to vectors
  - Things are ›embedded‹ in a vector space
- Why do we need this?
  - Classically, words are discrete symbols
    - For neural networks, each word is replaced by a word index

Motivation

- An embedding is a mapping of words or documents to vectors
  - Things are ›embedded‹ in a vector space
- Why do we need this?
  - Classically, words are discrete symbols
    - For neural networks, each word is replaced by a word index
- We can do better
  - Representing a word as a vector allows calculating similarity between words
  - If the embedding works well, similarity between words has *meaning*

Word Embeddings (after 2013)

Mikolov et al. 2013, Pennington et al. 2014, Bojanowski et al. 2016, ...

- Dense representations of words in vector space
- Word vectors: Weights learned by a simple neural network with a classification target
  - word2vec: Given word w_i, how likely is it that w_j appears in its context?
- Idea
  - Embeddings are learned using a neural network
  - Classification task: Given a word, predict its context words
    - Training data in abundance
  - Use learned weights as embeddings

Embeddings and neural networks

- Existing (pre-trained) embeddings can be plugged in
- Specific embeddings can be trained, just like all other weights

Embeddings and neural networks

- Existing (pre-trained) embeddings can be plugged in
- Specific embeddings can be trained, just like all other weights

Pre-trained embeddings

- Glove (Stanford University): `https://nlp.stanford.edu/projects/glove/`
- FastText (facebook research): `https://fasttext.cc` (multiple languages)

Embeddings and neural networks

- Existing (pre-trained) embeddings can be plugged in
- Specific embeddings can be trained, just like all other weights

Pre-trained embeddings

- Glove (Stanford University): `https://nlp.stanford.edu/projects/glove/`
- FastText (facebook research): `https://fasttext.cc` (multiple languages)

```
adventure 0.0292 -0.0269 0.0273 0.0792 -0.0617 0.1370 -0.0628 0.0420
0.0743 0.0979 -0.0136 0.0488 -0.0267 -0.0227 0.0592 0.0410 0.0314 0.0378
-0.0455 0.0616 -0.0380 0.0232 -0.0218 0.0000 -0.0699 -0.1327 -0.0393
0.0467 0.0413 0.0089 -0.0046 0.0372 -0.0590 0.0740 0.0214 0.0625 0.0067
-0.0063 0.0218 -0.0447 -0.0298 0.0186 -0.0207 0.0158 -0.0508 -0.0297
-0.0807 -0.0619 -0.0194 -0.0153 0.0909 -0.0037 0.0999 -0.0110 ...
```

Embeddings and neural networks

- This changes the input data shape

This	is	a	great	adventure	.

Embeddings and neural networks

» This changes the input data shape

This	is	a	great	adventure	.

Embeddings and neural networks

» This changes the input data shape

This	is	a	great	adventure	.
↓	↓	↓	↓	↓	↓
-0.044	-0.0194	0.0979	-0.044	0.0292	-0.044
-0.029	-0.0153	-0.0136	-0.029	-0.0269	-0.029
0.018	0.0909	0.0488	0.018	0.0273	0.018
-0.020	-0.0037	-0.0267	-0.020	0.0792	-0.020
0.015	0.0999	-0.0227	0.015	-0.0617	0.015
-0.050	-0.0110	0.0592	-0.050	0.1370	-0.050
...	...	...	...	...	...

Embeddings and neural networks

» This changes the input data shape

This	is	a	great	adventure	.
-0.044	-0.0194	0.0979	-0.044	0.0292	-0.044
-0.029	-0.0153	-0.0136	-0.029	-0.0269	-0.029
0.018	0.0909	0.0488	0.018	0.0273	0.018
-0.020	-0.0037	-0.0267	-0.020	0.0792	-0.020
0.015	0.0999	-0.0227	0.015	-0.0617	0.015
-0.050	-0.0110	0.0592	-0.050	0.1370	-0.050
...	...	...	...	...	...

» This is a matrix!

- I.e., by embedding tokens into a vector space, we have changed the shape of our data from 1D to 2D

Fixed Length of Input

- Our input now consists of a matrix (per instance)
- Matrix size needs to be predefined
  - Embedding dimension: Parameter we can set freely
  - Length: To be set on training data

Fixed Length of Input

- Our input now consists of a matrix (per instance)
- Matrix size needs to be predefined
  - Embedding dimension: Parameter we can set freely
  - Length: To be set on training data
- Input length
  - This parameter controls how long sentences (or texts) can be
  - It's a hard limit

Fixed Length of Input

- Our input now consists of a matrix (per instance)
- Matrix size needs to be predefined
  - Embedding dimension: Parameter we can set freely
  - Length: To be set on training data
- Input length
  - This parameter controls how long sentences (or texts) can be
  - It's a hard limit
- Padding
  - Extend shorter inputs so that they have the same length
  - Truncate longer inputs
  - Keras: Function `tensorflow.keras.preprocessing.sequence.pad_sequences(...)`

Section 3

Implementing Embeddings in Keras

Implementing Embeddings in Keras

- Two relevant new layers
  - `tensorflow.python.keras.layers.Embedding()`
  - `tensorflow.python.keras.layers.Flatten()`
- Preparations
  - `tensorflow.keras.preprocessing.text.Tokenizer()`
  - `tensorflow.keras.preprocessing.text.text_to_word_sequence()`
  - `tensorflow.keras.preprocessing.sequence.pad_sequences()`

Embeddings

`tensorflow.python.keras.layers.Embedding(...)`

- Must be the first layer of the model
- Turns positive integers (indexes) into dense vectors of fixed size

Embeddings

`tensorflow.python.keras.layers.Embedding(...)`

- Must be the first layer of the model
- Turns positive integers (indexes) into dense vectors of fixed size
- Parameters
  - `input_dim`: Size of the vocabulary (i.e., how many words to distinguish)
  - `output_dim`: How many elements/dimensions do word vectors have?
  - `input_length`: Length of input vectors (e.g., sentences)

Embeddings

`tensorflow.python.keras.layers.Embedding(...)`

- Must be the first layer of the model
- Turns positive integers (indexes) into dense vectors of fixed size
- Parameters
  - `input_dim`: Size of the vocabulary (i.e., how many words to distinguish)
  - `output_dim`: How many elements/dimensions do word vectors have?
  - `input_length`: Length of input vectors (e.g., sentences)
- Pre-trained embeddings
  - Documentation: `https://keras.io/examples/nlp/pretrained_word_embeddings/`
  - Parameters
    - `embeddings_initializer=keras.initializers.Constant(embedding_matrix)`
    - `trainable=False`
  - `embedding_matrix` is a numpy matrix that contains the vectors loaded from a file

Flatten

- Network structure so far: Only 1-dimensional vectors
- Result of embedding layer: matrices (2D)
- Flatten layer: Combine all rows of a matrix to a long vector
- `layers.Flatten()`
  - Usually used after an embedding layer

Preparations

1. Tokenizer
   - Establish vocabulary
   - Assign each type an integer number
2. Map token sequences to integer sequences
3. Padding
   - Ensure that all sequences have the same length by truncating or adding

Full Example

```
tokenizer = Tokenizer()
tokenizer.fit_on_texts(train_texts)
vocab_size = len(tokenizer.word_index) + 1
train_texts = tokenizer.texts_to_sequences(train_texts)

MAX_LENGTH = max(len(train_ex) for train_ex in train_texts)

train_texts = pad_sequences(train_texts, maxlen=MAX_LENGTH, padding="post")

model = models.Sequential()
model.add(layers.Input(shape=(MAX_LENGTH)))
model.add(layers.Embedding(vocab_size, 200, input_length=MAX_LENGTH))
model.add(layers.Flatten())
model.add(layers.Dense(10, activation="sigmoid"))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(1, activation="sigmoid"))

model.summary()
```

Section 4

Exercise

Exercise 08

https://github.com/IDH-Cologne-Deep-Learning-Uebung/exercise-08