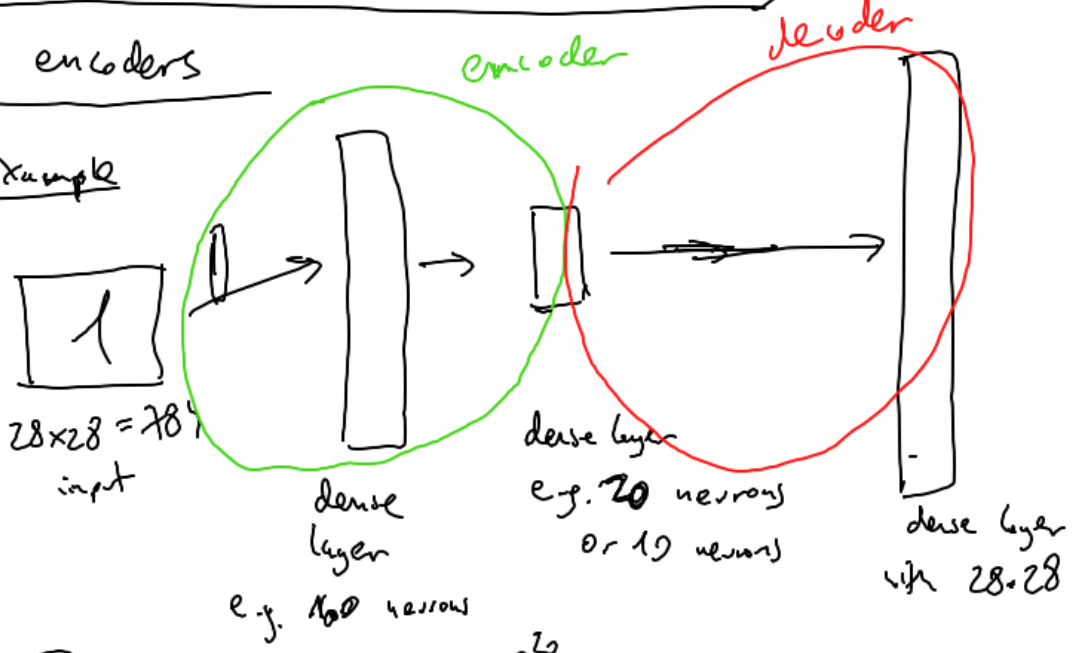
- LSTM (long-short term memory) layers
- GRU (gated recurrent unit) layers (Cho et al. 2014)
  a simplified version of LSTM (smaller number of weights), but still often the same performance

self-supervised learning

Auto encoders

Example



- we train the network so that the output is the same as the input

  In fact, we want to obtain 2 networks which are parts of the above one: encoder & decoder

  Encoder reduces the dimension of the input without losing much information

The layers need not be dense, we could also have convolutional layers etc.

Natural Language Processing (NLP)

Example: . sentiment analysis

movie reviews $\xrightarrow{\text{aim}}$ predict how good the movie is according to the reviewer

'This movie is good' → 4/5

'This movie is not good' → 1/5

Lousy ⇕ Bad

. translations

. suggestion for next words (texting)

Good ... Morning Afternoon

One-hot encoding of words:

- construct a dictionary of words, possibly with some special tokens
  e.g. [UNK] unknown word
  [padding] (if one wants to have sequences of fixed length)
  [eos] end of sentence

- each word is encoded as a 1-hot vector

Sequence of words ⟶ sequence of vectors (with 0 and 1)

The drawback: · the vectors have large dimension
· the training set has to be very large, so that the model 'sees' every word often enough

Models: one of the simplest is the 'bag of words'

One just considers a set of the words that appear, without taking the order into account

Alternative to the one-hot encoding:

use word embeddings:

$$f: \{\text{words}\} \longrightarrow \mathbb{R}^N$$

with $N \ll \#\{\text{words}\}$

GloVe $\#\text{words} \cong 400'000$

$N = 50, 100, 200, 300$

$50 \{-1, 0, 1\}^{3^{50}}$

The function f should catch the meaning of the words

i.e. similar words should have similar values

If we use $f(\text{words})$ instead of the one-hot encodings, then

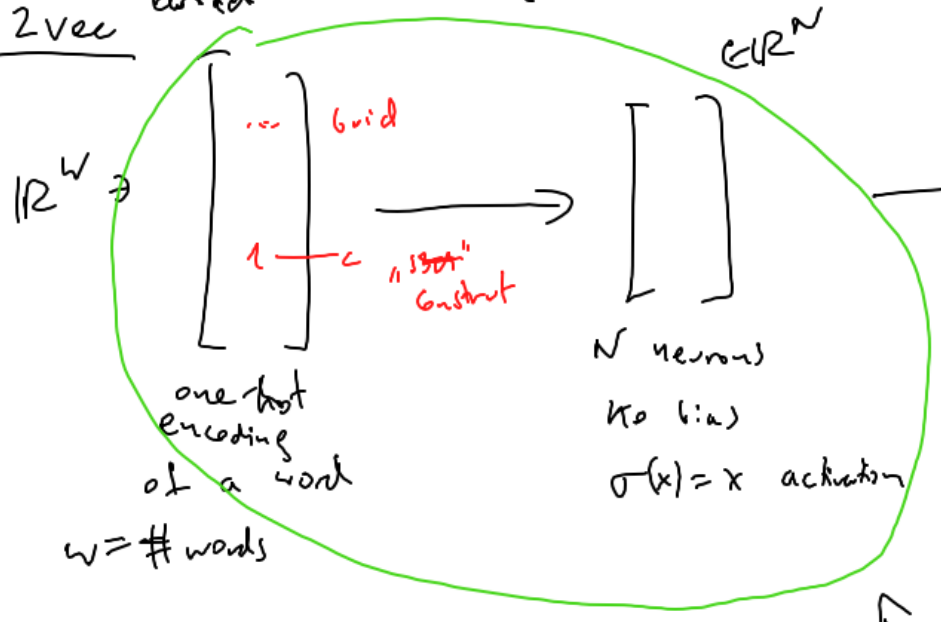
the dimension is not that large anymore

the training set does not need to be that large, because f carries the information about the meaning of the words

How to construct such an embedding?

'build' — t — 'an'

Word2vec



c – word (context), input

t – target word: a word that is found close to the word c

This is not the real objective we care about this part

GloVe

X_{ij} = probability (frequency) of finding word i close to j

objective: find $\theta_i, e_i \in \mathbb{R}^N$, $i = 1, \dots, W$ and b_i, b_i', $i = 1, \dots, W$ to minimise

$$\sum_{i=1}^{W} \sum_{j=1}^{W} f(X_{ij}) \left(\theta_i^{\top} e_j + b_i + b_j' - \log X_{ij} \right)^2$$

i - fixed

j, j'

↑ weight $= 0$ if $X_{ij} = 0$

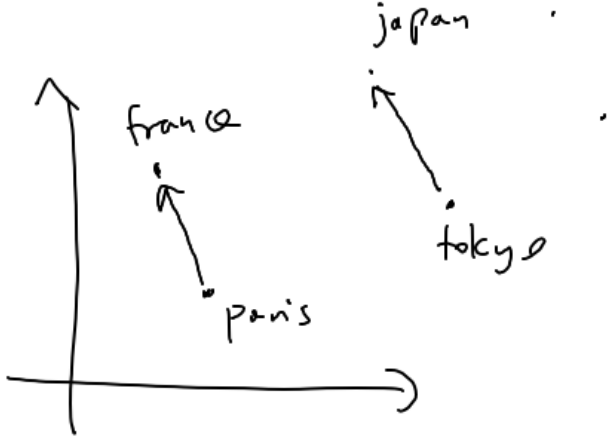
small for words that are very common (e.g. 'the')

	k=solid	k=gas	k=water	k=fashion
P(k|ice)	$1.9 \cdot 10^{-4}$	$6.6 \cdot 10^{-5}$	$3.0 \cdot 10^{-3}$	$1.7 \cdot 10^{-5}$
P(k|steam)	$2.2 \cdot 10^{-5}$	$7.8 \cdot 10^{-4}$	$2.2 \cdot 10^{-3}$	$1.8 \cdot 10^{-5}$
P(k|ice)/P(k|steam)	8.9	0.085	1.36	0.96

↳ embedding as a by-product

These embeddings have some interesting properties:

$$f(\text{'paris'}) - f(\text{'france'}) \approx f(\text{'tokyo'}) - f(\text{'japan'})$$



it catches analogies

The dangers of using word embeddings:

Paper "Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings"

Computer programmer ... (A)

A man ... (B)

Computer programmer ... (B)

A man ... (A)

Jürgen ...

Jamaal ...

Mohammed ...