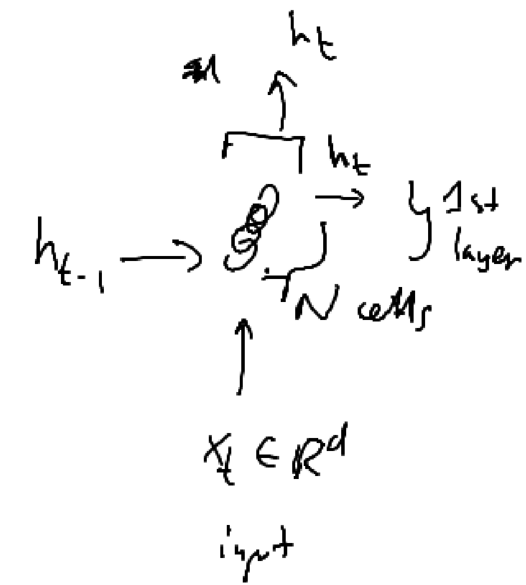
LSTM – before

GRU – Gated Recurrent Unit (Cho et al., 2014)




$$z_t = \sigma_g(\overset{N\times d}{W_z} \cdot \overset{\in \mathbb{R}^d}{x_t} + \overset{N \times N}{U_z} \overset{\in \mathbb{R}^N}{h_{t-1}} + \overset{\in \mathbb{R}^N}{b_z})$$ update gate

$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r)$$ reset gate

$$\hat{h}_t = \phi_h(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h)$$ candidate activation

$$h_t = (1-z_t) \odot h_{t-1} + z_t \odot \hat{h}_t$$ – output & new hidden state

LSTM:

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

| LSTM | GRU |
| --- | --- |
| $4(Nd + N^2 + N)$ weights | $3(Nd + N^2 + N)$ weights |

simple RNN

$Nd + N^2 + N$

## Example

1 LSTM cell

input $(x_1, x_2, \ldots, x_t) \in \mathbb{R}^t$

aim: predict $\hat{x}_{t+1} = x_1$

$$\Big( \underbrace{(x_1, x_2, \ldots, x_t)}_{\text{randomly sampled}} , x_1 \Big)$$

# Natural Language Processing

**Example:** sentiment analysis

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

'This movie is great!' $\longrightarrow$ 5/5

'This movie is horrible.' 1/5

---

one-hot encoding of words:

1) construct a dictionary of words, maybe with some special tokens e.g. [UNK] - unknown word token
   [padding]
   [EOS]

2) each word is encoded as a 1-hot vector $\begin{bmatrix}0\\0\\\vdots\\1\\\vdots\\0\end{bmatrix}$. $\begin{bmatrix}1\\0\\\vdots\\0\end{bmatrix} \rightsquigarrow a$

Drawbacks:

- the dimension of the input is large
  - → performance problems
  - → training set needs to be very large

Solution: use word embeddings

i.e. a function

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

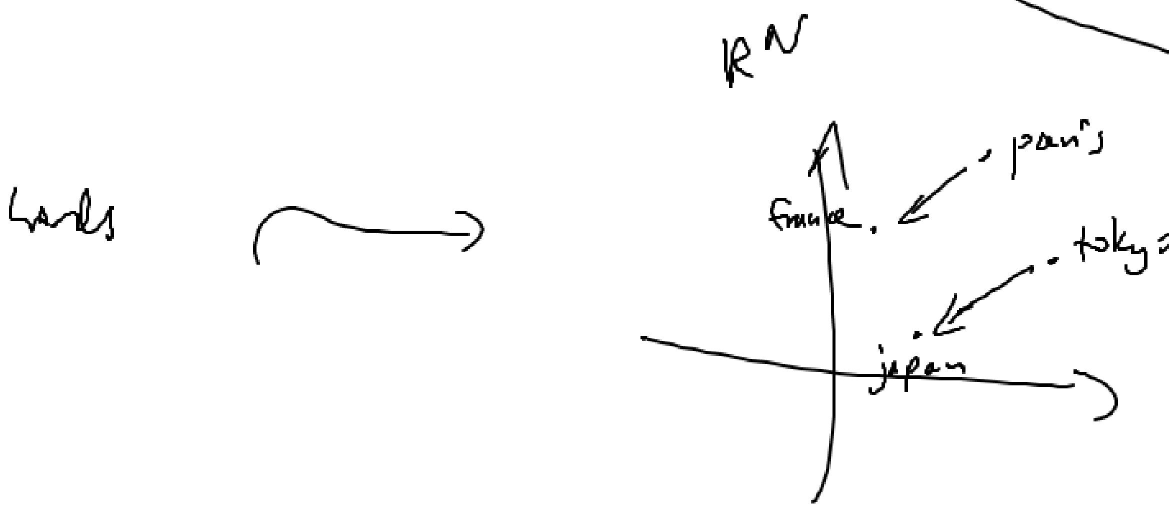
e.g. GloVe: $\#words = 400'000$

$N = 50, 100, 200, 300$

that can catch the meaning of words, for example f should have similar values for words with similar meaning

In word2vec and GloVe the embedding function f has also this property, e.g.

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




similarity measure usually are uses:

(minus) Euclidean distance

· $u \approx v$ : measure:

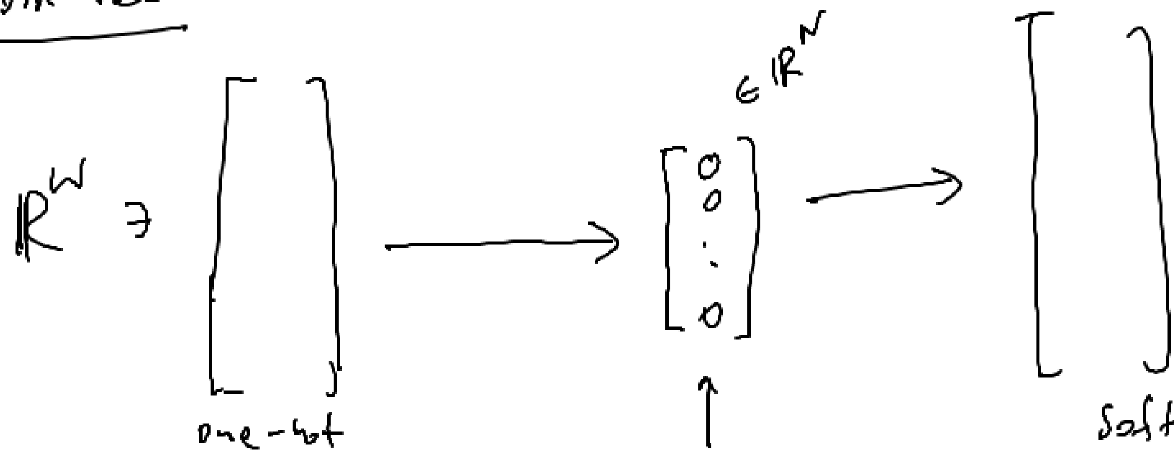
$$\frac{u \circ v}{\|u\| \cdot \|v\|} \quad (= \cos \measuredangle(u,v))$$

By using such embedding, the dim. of input is reduced and also we can have a smaller training set.

How to $\underbrace{\text{construct}}_{c}$ $\underbrace{\text{such}}_{t}$ $\underbrace{\text{an}}_{t'}$ $\underbrace{\text{embedding}}_{t''}$?

Word2vec

We feed a word $c$ (context) and ~~ask~~ check the prob. of word $t$ (target) appearing close to $c$ in the corpus.



$\mathbb{R}^W \ni$ one-hot encoding of a word

$W = \#$ words in the dict.

$\in \mathbb{R}^N$ $\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$

$N$ neurons here, no bias, activation function: $\sigma(x) = x$

Softmax layer with $W$ neurons

$$\frac{\exp(v_j^T \cdot e_i)}{\sum_{j=1}^{W} \exp(v_j^T \cdot e_i)}$$

$o_w \longrightarrow W \cdot o_w = e_w$

$o_w$: one hot encoding of the word

$W$: weights

$\sigma(W \cdot o_w + \cancel{b})$

$e_w$: encoding of the word $w$

$$\mathcal{L}(y, \hat{y}) = -\sum_{j=1}^{W} y_j \cdot \log \hat{y}_j$$

E.g. $c$ – context word

$t$ – target word chosen in a close proximity of $c$ (say, $\pm 5$ words)

One may want to sample the most common words less often,

the least –||– more often than we would

with a uniform distribution.

# GloVe

construct a matrix of word-word occurrence probabilities (occurrence of cases where $w_1$ and $w_2$ are close):

$X_{ij}$ = probab. (frequency) of finding word $i$ close to $j$

| Prob./ratio | $k$ = solid | $k$ = gas | $k$ = water | $k$ = fashion |
|---|---|---|---|---|
| $P(k \mid ice)$ | $1.9 \cdot 10^{-4}$ | $6.6 \cdot 10^{-5}$ | $3.0 \cdot 10^{-3}$ | $1.7 \cdot 10^{-5}$ |
| $P(k \mid steam)$ | $2.2 \cdot 10^{-5}$ | $7.8 \cdot 10^{-4}$ | $2.2 \cdot 10^{-3}$ | $1.8 \cdot 10^{-5}$ |
| $P(k \mid ice)/P(k \mid steam)$ | $8.9$ | $8.5 \cdot 10^{-2}$ | $1.36$ | $0.96$ |

$\theta_i \circ e_j \approx \log X_{ij}$

$\theta_i \circ e_k \approx \log X_{ik}$

$\theta_i \circ (e_j - e_k) = \log \frac{X_{ij}}{X_{ik}}$

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:

by-product: embedding

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

weight, $= 0$ if $X_{ij} = 0$, smaller for words which are very common, larger for less common