NPFL116 Compendium of Neural Machine Translation

Recurrent Neural Networks

March 1 and 8, 2017

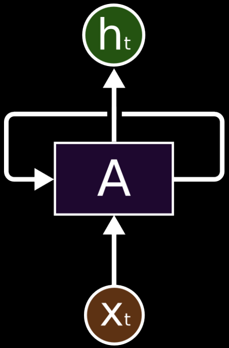
Jindřich Libovický, Jindřich Helcl

Charles Univeristy in Prague
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics

Why RNNs

- `for` loops over sequential data
- the most frequently used type of network in NLP

General Formulation



- inputs: $x_{,} \ldots, x_T$
- initial state $h_0 = \mathbf{0}$, a result of previous computation, trainable parameter
- recurrent computation: $h_t = A(h_{t-1}, x_t)$

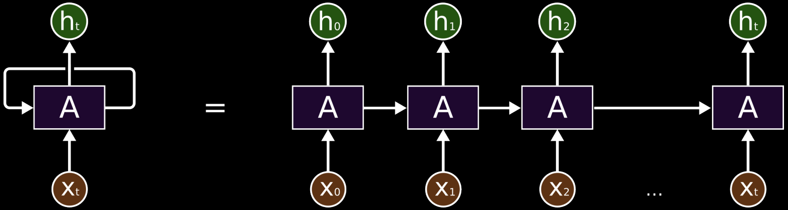
RNN as Imperative Code

```
def rnn(initial_state, inputs):
  prev_state = initial_state
  for x in inputs:
    new_state, output = rnn_cell(x, prev_state)
    prev_state = new_state
    yield output
```

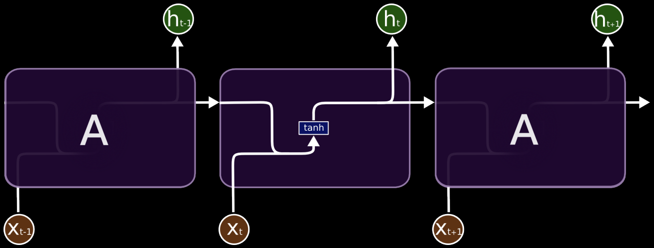
RNN as Functional Code

```
def rnn(state, inputs) = inputs match {
  case Nil => Nil
  case x :: rest => {
    new_state, output = rnn_cell(x, state)
    output :: rnn(new_state, rest)
  }
}
```

RNN as a Fancy Image



Vanilla RNN

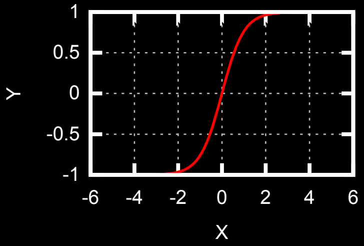


$$h_t = \tanh\left(W[h_{t-1}; x_t] + b\right)$$

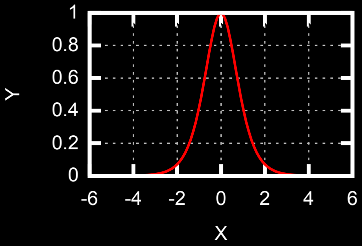
- cannot propagate long-distance relations
- vanishing gradient problem

Vanishing Gradient Problem (1)

$$\tanh x = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$

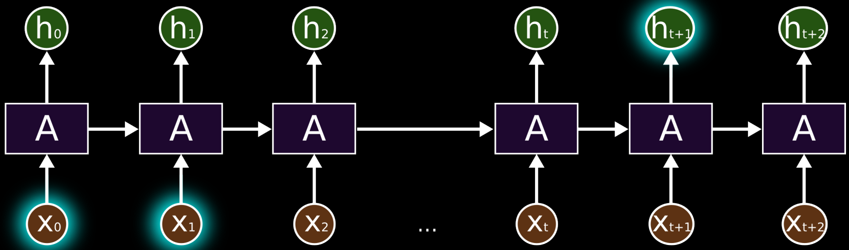


$$\frac{d \tanh x}{dx} = 1 - \tanh^2 x \in (0, 1]$$



Weight initialized $\sim \mathcal{N}(0, 1)$ to have gradients further from zero.

Vanishing Gradient Problem (2)



$$\frac{\partial E_{t+1}}{\partial b} = \frac{\partial E_{t+1}}{\partial h_{t+1}} \cdot \frac{\partial h_{t+1}}{\partial b}$$ (chain rule)

Vanishing Gradient Problem (3)

$$\frac{\partial h_t}{\partial b} = \frac{\partial \tanh \overbrace{(W_h h_{t-1} + W_x x_t + b)}^{=z_t \text{ (activation)}}}{\partial b}$$

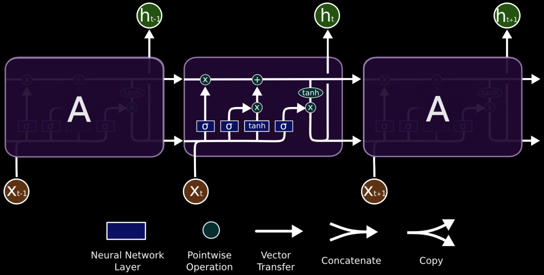
(tanh′ is derivative of tanh)

$$= \tanh'(z_t) \cdot \left(\frac{\partial W_h h_{t-1}}{\partial b} + \underbrace{\frac{\partial W_x x_t}{\partial b}}_{=0} + \underbrace{\frac{\partial b}{\partial b}}_{=1} \right)$$

$$= \underbrace{W}_{\sim \mathcal{N}(0,1)} \underbrace{\tanh'(z_t)}_{\in (0;1]} \frac{\partial h_{t-}}{\partial b} + \tanh'(z_t)$$

LSTMs

LSTM = Long short-term memory

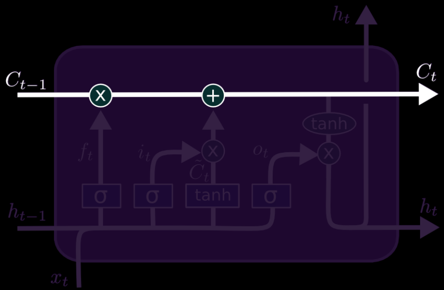


Control the gradient flow by explicitly gating:

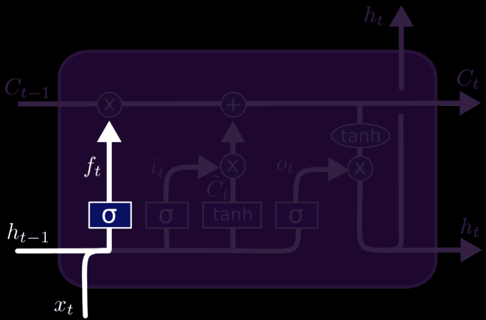
- what to use from input,
- what to use from hidden state,
- what to put on output

Hidden State

- two types of hidden states
- h_t — "public" hidden state, used an output
- c_t — "private" memory, no non-linearities on the way
  - direct flow of gradients (without multiplying by $\leq$ derivatives)
  - only vectors guaranteed to live in the same space are manipulated
- information highway metaphor



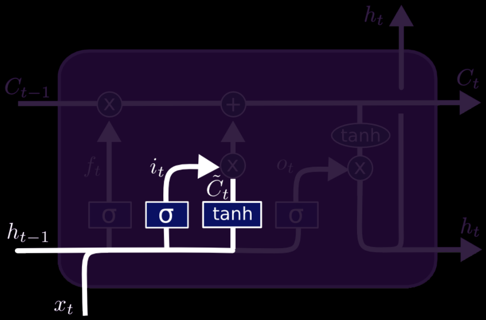
Forget Gate



$$f_t = \sigma\left(W_f[h_{t-1}; x_t] + b_f\right)$$

- based on input and previous state, decide what to forget from the memory

Input Gate

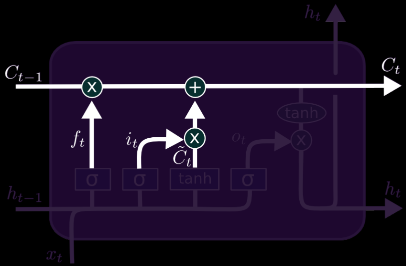


$$i_t = \sigma\left(W_i \cdot [h_{t-1}; x_t] + b_i\right)$$

$$\tilde{C}_t = \tanh\left(W_c \cdot [h_{t-1}; x_t] + b$$

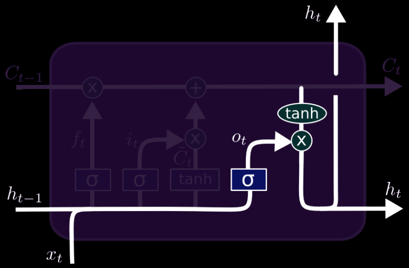
- $\tilde{C}$ — candidate what may want to add to the memory
- i_t — decide how much of the information we want to store

Cell State Update



$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$

Output Gate



$$o_t = \sigma\left(W_o \cdot [h_{t-1}; x_t] + b_o\right)$$

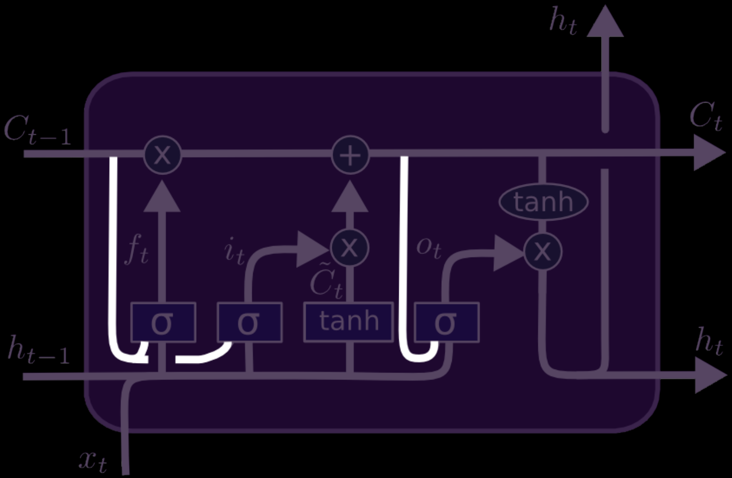
$$h_t = o_t \odot \tanh C_t$$

Here we are!

$$
\begin{aligned}
f_t &= \sigma\left(W_f[h_{t-1}; x_t] + b_f\right) \\
i_t &= \sigma\left(W_i \cdot [h_{t-1}; x_t] + b_i\right) \\
o_t &= \sigma\left(W_o \cdot [h_{t-1}; x_t] + b_o\right) \\
\tilde{C}_t &= \tanh\left(W_c \cdot [h_{t-1}; x_t] + b_C\right) \\
C_t &= f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \\
h_t &= o_t \odot \tanh C_t
\end{aligned}
$$

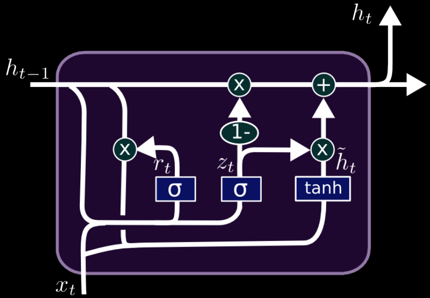
Exercise: How would you implement it efficiently?
Compute all gates in a single matrix multiplication.

LSTM with Peephole Connections



Just add h_{t-1} to all linear combinations.

Gated Recurrent Units



$$z_t = \sigma\left(W_z[h_{t-1}; x_t] + b_z\right)$$

$$r_t = \sigma\left(W_r[h_{t-1}; x_t] + b_r\right)$$

$$\tilde{h}_t = \tanh\left(W[r_t \odot h_{t-1}; x_t]\right)$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$$

GRU or LSTM?

- GRU preserved the information highway property
- less parameters, should learn faster
- LSTM more general (although both Turing complete)
- empirical results: it's task-specific

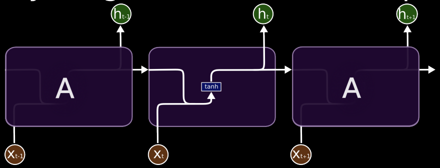
Exercise: Are GRUs special case of LSTMs?

No, you cannot lay $C_t \equiv h_t$ because of the additional non-linearity in LSTMs.

Chung, Junyoung, et al. "Empirical evaluation of gated recurrent neural networks on sequence modeling." arXiv preprint arXiv:412.3555 (204).
Irie, Kazuki, et al. "LSTM, GRU, highway and a bit of attention: an empirical overview for language modeling in speech recognition." Interspeech, San Francisco, CA, USA (206).

Type Theory View on Gated RNNs

- every projection is type transformation (vector live in different space)
- vanilla RNNs forces to project h_t, s.t. it stays in the same space — difficult with non-linearities
- in LSTM everything lives in the same space



- successfully training a vanilla RNN = getting the very fragile type-preserving transformation (e.g., as Mikolov in his first recurrent LM)

Balduzzi, David, and Muhammad Ghifary. ”Strongly-typed recurrent neural networks.” arXiv preprint arXiv:602.02218 (2016).

Language Model

Estimate probability of a sentence using chain rule:

$$\Pr\left(w_{,}w_2, \ldots, w_n\right) = \prod_{i=}^{n} \Pr\left(w_i | w_{i-1}, w_{i-2}, \ldots, w_1\right) \qquad (1)$$

Exercise: How would you do that?

(i) What would be the output of the network?
(ii) What do you expect some problems with network inputs?

Output – softmax over the Vocabulary

$$l(w)_t = W_l h_t + b_l$$

$$p(w)_t = \frac{\exp l(w)_t}{\sum_{w' \in V} \exp l(w')_t}$$

V is vocabulary

- $l(w)_t$ — logits/energies for word w in time t
- weight matrix: hidden state $\times$ vocabulary size
- big weight matrix + costly normalization
- tricks what to do with it (negative sampling, hierarchical softmax) — not frequently used

From RNN Input to Word Embeddings

- input in one hot representation $(0, \ldots, 0, 1, 0, \ldots, 0)$ — length of vocabulary
- matrices projecting input ($W_f^{(x)}$, $W_i^{(x)}$, $W_o^{(x)}$) into hidden states are huge and do pretty much the same
- we can factorize it: $W_f^{(x)} = E \cdot W_f'^{(x)}$
- E — matrix of word embeddings (one hot $\rightarrow$ one row for each word)

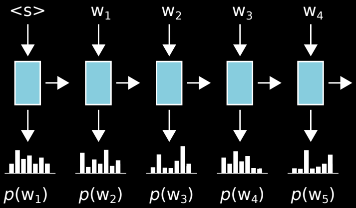
Word Embeddings

- word representations E have interesting properties (semantic, morphological clusters)
- representations from Tomáš Mikolov's PhD theses first had the interesting algebraic properties
- `word2vec` – how to propagate the same information as in the RNN LM with much simple architecture
- later shown to be a factorization of PMI matrix

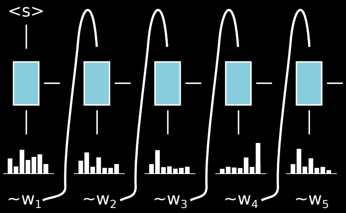
Tomáš Mikolov's PhD. Thesis (`http://www.fit.vutbr.cz/~imikolov/rnnlm/thesis.pdf`)
Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." arXiv preprint arXiv:1301.3781 (2013).
Goldberg, Yoav, and Omer Levy. "word2vec explained: Deriving Mikolov et al.'s negative-sampling word-embedding method." arXiv preprint arXiv:1402.3722 (2014)."'"

Sampling from a Model

Model computing probability of a sentence:



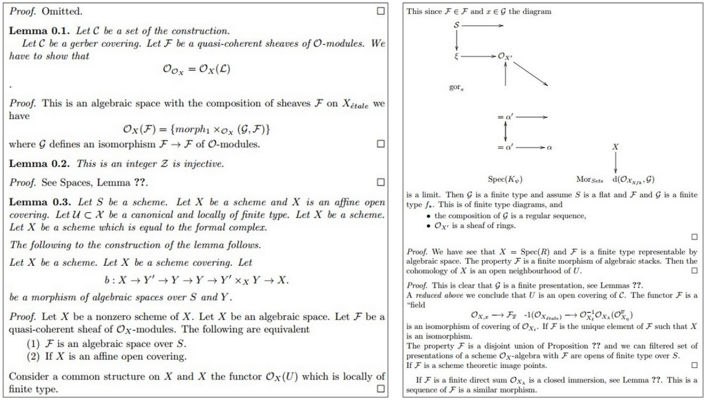
How would you sample from the model?



…and this is basically the recurrent decoder

Other tricks

- multi-layer networks, character level



Proof. Omitted. □

Lemma 0.1. *Let $\mathcal{C}$ be a set of the construction.*

Let $\mathcal{C}$ be a gerber covering. Let $\mathcal{F}$ be a quasi-coherent sheaves of $\mathcal{O}$-modules. We have to show that

$$\mathcal{O}_{\mathcal{O}_X} = \mathcal{O}_X(\mathcal{L})$$

.

Proof. This is an algebraic space with the composition of sheaves $\mathcal{F}$ on $X_{\acute{e}tale}$ we have

$$\mathcal{O}_X(\mathcal{F}) = \{morph_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

where $\mathcal{G}$ defines an isomorphism $\mathcal{F} \to \mathcal{F}$ of $\mathcal{O}$-modules. □

Lemma 0.2. *This is an integer $\mathcal{Z}$ is injective.*

Proof. See Spaces, Lemma ??. □

Lemma 0.3. *Let S be a scheme. Let X be a scheme and X is an affine open covering. Let $\mathcal{U} \subset \mathcal{X}$ be a canonical and locally of finite type. Let X be a scheme. Let X be a scheme which is equal to the formal complex.*

The following to the construction of the lemma follows.

Let X be a scheme. Let X be a scheme covering. Let

$$b : X \to Y' \to Y \to Y \to Y' \times_X Y \to X.$$

be a morphism of algebraic spaces over S and Y.

Proof. Let X be a nonzero scheme of X. Let X be an algebraic space. Let $\mathcal{F}$ be a quasi-coherent sheaf of $\mathcal{O}_X$-modules. The following are equivalent

(1) $\mathcal{F}$ is an algebraic space over S.
(2) If X is an affine open covering.

Consider a common structure on X and X the functor $\mathcal{O}_X(U)$ which is locally of finite type. □

This since $\mathcal{F} \in \mathcal{F}$ and $x \in \mathcal{G}$ the diagram

$$\begin{tikzcd}
\mathcal{S} \arrow[r] \arrow[d] & {} & & \\
\xi \arrow[r] & \mathcal{O}_{X'} \arrow[dr] & & \\
\mathrm{gor}_s & {} \arrow[u] & {} & \\
& =\alpha' \arrow[r] \arrow[d, leftrightarrow] & {} & \\
& =\alpha' \arrow[r] & \alpha & X \arrow[d] \\
& \mathrm{Spec}(K_\psi) & \mathrm{Mor}_{Sets} & \mathrm{d}(\mathcal{O}_{\mathcal{X}_{X/k}}, \mathcal{G})
\end{tikzcd}$$

is a limit. Then $\mathcal{G}$ is a finite type and assume S is a flat and $\mathcal{F}$ and $\mathcal{G}$ is a finite type f_*. This is of finite type diagrams, and

- the composition of $\mathcal{G}$ is a regular sequence,
- $\mathcal{O}_{X'}$ is a sheaf of rings.

□

Proof. We have see that $X = \mathrm{Spec}(R)$ and $\mathcal{F}$ is a finite type representable by algebraic space. The property $\mathcal{F}$ is a finite morphism of algebraic stacks. Then the cohomology of X is an open neighbourhood of U. □

Proof. This is clear that $\mathcal{G}$ is a finite presentation, see Lemmas ??.
A *reduced above* we conclude that U is an open covering of $\mathcal{C}$. The functor $\mathcal{F}$ is a "field

$$\mathcal{O}_{X,x} \longrightarrow \mathcal{F}_{\overline{x}} \ \ \text{-1}(\mathcal{O}_{X_{\acute{e}tale}}) \longrightarrow \mathcal{O}_{X_\ell}^{-1}\mathcal{O}_{X_\lambda}(\mathcal{O}_{X_\eta}^{\overline{v}})$$

is an isomorphism of covering of $\mathcal{O}_{X_i}$. If $\mathcal{F}$ is the unique element of $\mathcal{F}$ such that X is an isomorphism.
The property $\mathcal{F}$ is a disjoint union of Proposition ?? and we can filtered set of presentations of a scheme $\mathcal{O}_X$-algebra with $\mathcal{F}$ are opens of finite type over S.
If $\mathcal{F}$ is a scheme theoretic image points. □

If $\mathcal{F}$ is a finite direct sum $\mathcal{O}_{X_\lambda}$ is a closed immersion, see Lemma ??. This is a sequence of $\mathcal{F}$ is a similar morphism.

Source: Andrej Karpathy's blog (`http://karpathy.github.io/2015/05/21/rnn-effectiveness/`)

Reading for the Next Week

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. ”Neural machine translation by jointly learning to align and translate.” arXiv preprint arXiv:1409.0473 (2014).

`https://arxiv.org/pdf/1409.0473.pdf`

Question:

What do you think is the main difference between Bahdanau’s attention model and concept of alignment in statistical MT?