Deep Learning

Übung WS 23/24

Judith Nester (nester@uni-koeln.de)

11-01-2024

Evaluation der Veranstaltung

Zugänglich bis 17.01.2024 23:59:00 Uhr!

`https://uzk-evaluation.uni-koeln.de/evasys/online.php?pswd=6K8QK`

Semesterende-Party

Hoch die Hände, Semesterende!

Kommt und feiert mit uns das Ende der Vorlesungszeit 2023/24!
Wann: Do., 01.02.2024 ab 18 Uhr
Wo: IDH, Universitätsstr. 22, 1.OG

Für Bier und ein paar Snacks ist gesorgt.

A group of students and professors having a party to celebrate the end of the semester, pop art

Today

Sequential Data

Recurrent Neural Networks

Long Short-Term Memory (LSTM)

Exercise

Section 1

Sequential Data

Introduction

- So far: ›bag of words‹
- We count the number of times a word appears in a text

Introduction

- So far: ›bag of words‹
- We count the number of times a word appears in a text
- This is not how language works
  - Example
    - After the bad predecessor, this movie was very good.
    - After the good predecessor, this movie was very bad.
  - Both sentences have the same feature vector, but different meanings
- Convolution: Take multi-word structures into account

Introduction

- So far: ›bag of words‹
- We count the number of times a word appears in a text
- This is not how language works
    - Example
        - After the bad predecessor, this movie was very good.
        - After the good predecessor, this movie was very bad.
    - Both sentences have the same feature vector, but different meanings
- Convolution: Take multi-word structures into account
- CNNs mostly used for image recognition

Intuition

- Moving window over the input
- For each window, we apply logistic regression
- And continue with a slightly shorter vector

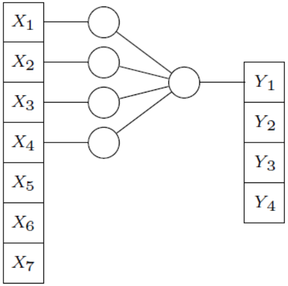


Figure: A 1D convolutional layer of size 4 (strides = 1)

Intuition

- Moving window over the input
- For each window, we apply logistic regression
- And continue with a slightly shorter vector

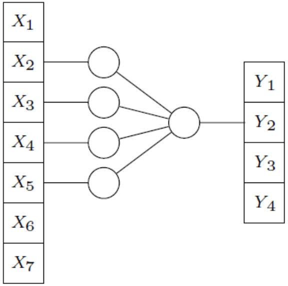


Figure: A 1D convolutional layer of size 4 (strides $= 1$)

Intuition

- Moving window over the input
- For each window, we apply logistic regression
- And continue with a slightly shorter vector

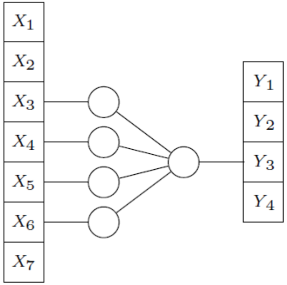


Figure: A 1D convolutional layer of size 4 (strides $= 1$)

Intuition

- Moving window over the input
- For each window, we apply logistic regression
- And continue with a slightly shorter vector

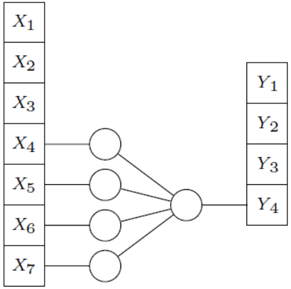


Figure: A 1D convolutional layer of size 4 (strides = 1)

1D Convolution

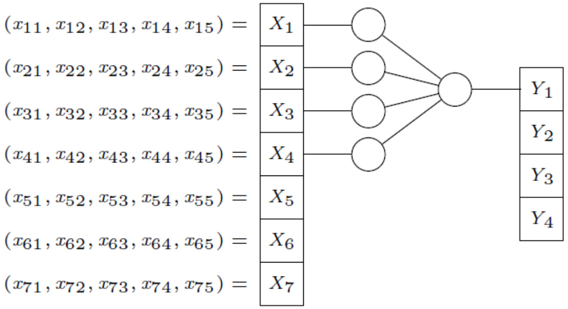

$(x_{11}, x_{12}, x_{13}, x_{14}, x_{15}) =$
$(x_{21}, x_{22}, x_{23}, x_{24}, x_{25}) =$
$(x_{31}, x_{32}, x_{33}, x_{34}, x_{35}) =$
$(x_{41}, x_{42}, x_{43}, x_{44}, x_{45}) =$
$(x_{51}, x_{52}, x_{53}, x_{54}, x_{55}) =$
$(x_{61}, x_{62}, x_{63}, x_{64}, x_{65}) =$
$(x_{71}, x_{72}, x_{73}, x_{74}, x_{75}) =$
X_1
X_2
X_3
X_4
X_5
X_6
X_7
Y_1
Y_2
Y_3
Y_4

Figure: A 1D convolutional layer of size 4 (strides = 1)

- » Requires a 2D input shape – because of embeddings!
  - (if you're not using embeddings, each token can be coded as a vector of length 1)

Convolution in Two Dimensions (e.g., images)

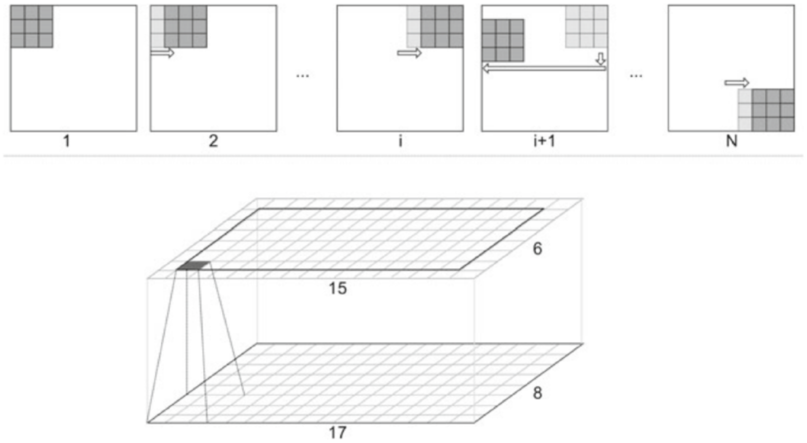


Figure: 2D convolutional layer (Skansi, 2018, p. 124)

Sequential Text Data

- Language works sequentially
  - Word meaning depends on context

Sequential Text Data

- Language works sequentially
  - Word meaning depends on context
- Feedforward neural networks
  - One instance (titanic passenger, document, ...) at a time
- Convolutional neural networks
  - Windows of fixed lengths over the data

Sequential Text Data

- Language works sequentially
  - Word meaning depends on context
- Feedforward neural networks
  - One instance (titanic passenger, document, . . .) at a time
- Convolutional neural networks
  - Windows of fixed lengths over the data
- Both are conceptually not adequate for natural language
- Length of influencing context varies

Sequential Text Data

- Language works sequentially
  - Word meaning depends on context
- Feedforward neural networks
  - One instance (titanic passenger, document, . . .) at a time
- Convolutional neural networks
  - Windows of fixed lengths over the data
- Both are conceptually not adequate for natural language
- Length of influencing context varies
- Recurrent neural networks are one solution to this problem

Section 2

Recurrent Neural Networks

Sequence Labeling

- So far: Classification
- Sequence labeling
  - Special case of classification
  - Instances are organized sequentially and dependent of each other
    - I.e.: The prediction for one class influences the next

Sequence Labeling

- So far: Classification
- Sequence labeling
  - Special case of classification
  - Instances are organized sequentially and dependent of each other
    - I.e.: The prediction for one class influences the next

Examples

- Part of speech tagging
  - »the dog barks« → »DET NN VBZ«
- Named entity recognition, mention detection
  - »John Bercow says he has changed allegiances to join Labour«
    → »B-PER I-PER O O O O O O O B-ORG«

BIO Scheme

- Named entity recognition is complicated
  - Not every token is part of a named entity (NE)
  - Many named entities span multiple tokens
  - We distinguish NEs based on the ontological type of the referent
    - PERson, ORGanization, LOCation, . . .

BIO Scheme

- Named entity recognition is complicated
  - Not every token is part of a named entity (NE)
  - Many named entities span multiple tokens
  - We distinguish NEs based on the ontological type of the referent
    - PERson, ORGanization, LOCation, . . .
- BIO scheme
  - How to represent NE annotations token-wise
  - Each token gets a label
    - B: Beginning of a NE
    - I: Inside of a NE
    - O: Outside of a NE (the majority of tokens)

BIO Scheme

- Named entity recognition is complicated
  - Not every token is part of a named entity (NE)
  - Many named entities span multiple tokens
  - We distinguish NEs based on the ontological type of the referent
    - PERson, ORGanization, LOCation, . . .
- BIO scheme
  - How to represent NE annotations token-wise
  - Each token gets a label
    - B: Beginning of a NE
    - I: Inside of a NE
    - O: Outside of a NE (the majority of tokens)
- Why B: Marking the beginning allows to recognize multiple multi-word NEs in direct sequence
  - ». . . hat Peter Schneider Maria Müller geküsst« $\rightarrow$ »O B-PER I-PER B-PER I-PER O«

Towards Recurrent Neural Networks

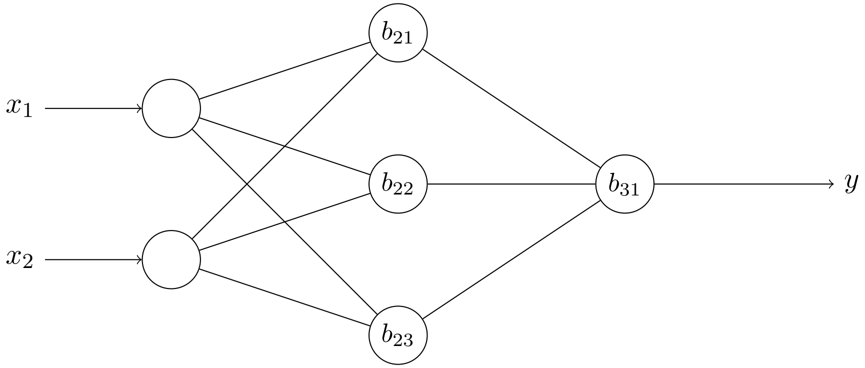


Figure: A feedforward neural network with 1 hidden layer (same picture as in Session 2)

Towards Recurrent Neural Networks

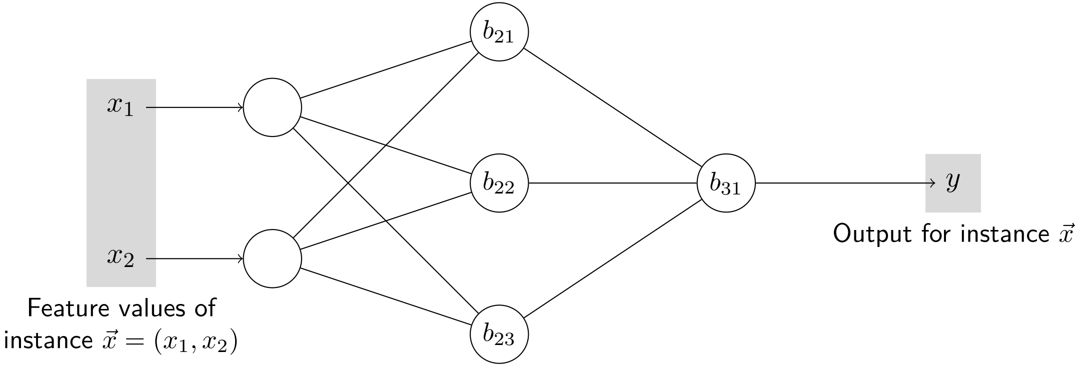

x_1
x_2
Feature values of instance $\vec{x} = (x_1, x_2)$
b_{21}
b_{22}
b_{23}
b_{31}
y
Output for instance $\vec{x}$

Figure: A feedforward neural network with 1 hidden layer (same picture as in Session 2)

Towards Recurrent Neural Networks

» To work with sequences, we need to include the sequence into the model

Notation

$X = (X_1, X_2, \dots)$	The input data set with instances
$X_i = (x_1, x_2, \dots)$	One instance with feature values
Y_i	Output for instance X_i

Towards Recurrent Neural Networks

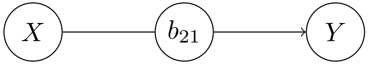


Figure: A simple neural network with 1 hidden layer

Recurrent Neural Networks

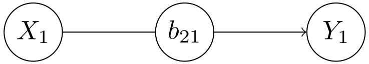


Figure: Recurrent Neural Network (unfolded)

Recurrent Neural Networks

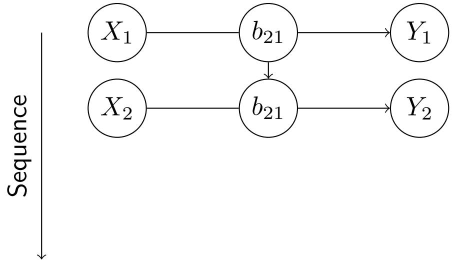


Figure: Recurrent Neural Network (unfolded)

Recurrent Neural Networks

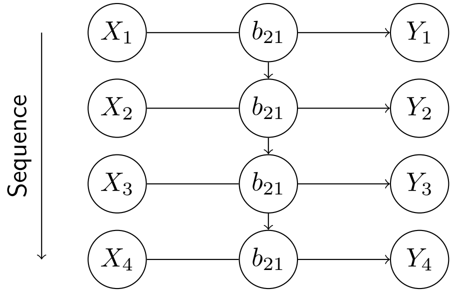


Figure: Recurrent Neural Network (unfolded)

Recurrent Neural Networks

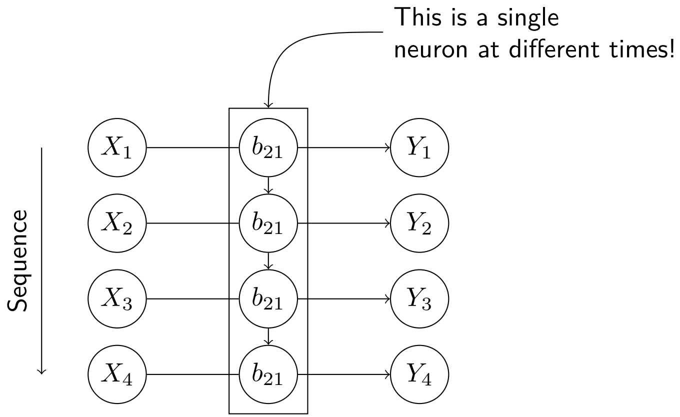


Figure: Recurrent Neural Network (unfolded)

Recurrent Neural Networks

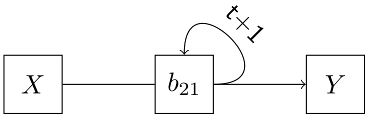
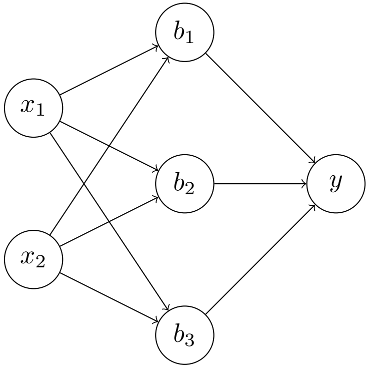


Figure: A recurrent neural network with 1 hidden layer (folded). Squares represent sequentially used neurons.

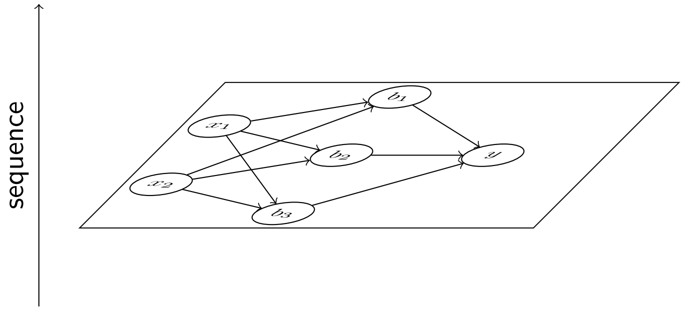
Recurrent Neural Networks

Example with multiple features per instance



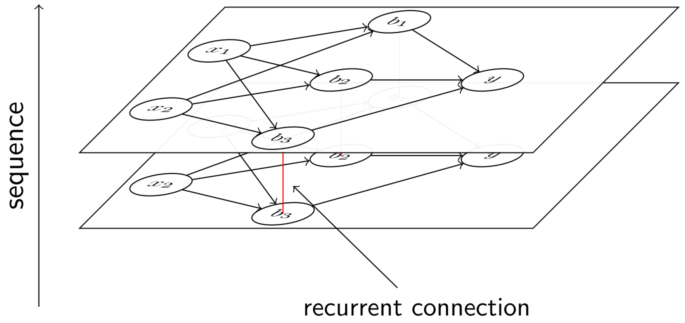
Recurrent Neural Networks

Example with multiple features per instance



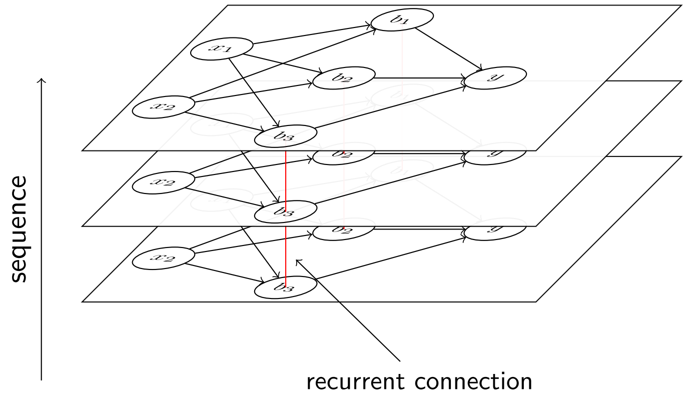
Recurrent Neural Networks

Example with multiple features per instance



Recurrent Neural Networks

Example with multiple features per instance



Recurrent Neural Networks

- FFNN, CNN: Weights between neurons
- RNN
  - Weights between neurons
  - Weight(s) for recurrent connections

Recurrent Neural Networks

- FFNN, CNN: Weights between neurons
- RNN
  - Weights between neurons
  - Weight(s) for recurrent connections

Input shape

RNN layers need 2D input:

- Length of input sequences (if needed, padded)
- Number of features (dimensions)
  - (this is where embeddings would go)

Implementation in keras

- `tf.keras.layers.SimpleRNN`
  - Documentation: `https://keras.io/api/layers/recurrent_layers/simple_rnn/`
    Selected parameters:
  - `recurrent_dropout=0.0` Dropout for recurrent links
  - `return_sequences=False` Wether to return the entire sequence or just the last element

```
model.add(layers.SimpleRNN(...))
```

Section 3

Long Short-Term Memory (LSTM)

Issues with RNNs

- Single neuron that transmits information along the sequence
- Long-distance information gets lost, because short-distance information is more prominent
- Slow because of the increased complexity
- Problem of vanishing and exploding gradients
- But: First architecture to process sequences as sequences

Long Short-Term Memory (LSTM)

- Most often used architecture for sequence labeling tasks
- Sub type of a recurrent layer
- Recurrent layer
  - Simple neuron, one connection along the sequence
- LSTM
  - Hochreiter and Schmidhuber (1997)
  - A neuron with more internal structure (often called »cell« or »unit«)
  - Two connections along the sequence

Recurrent Layer

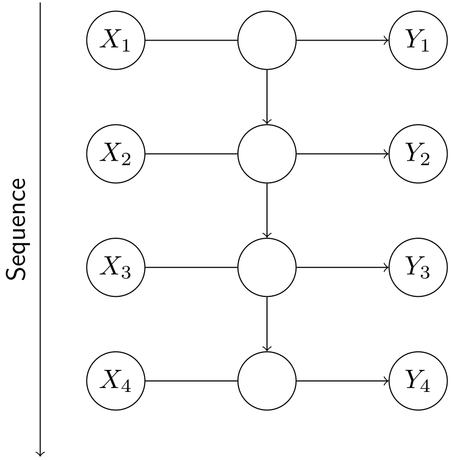


Figure: Recurrent Neural Network

LSTM Layers

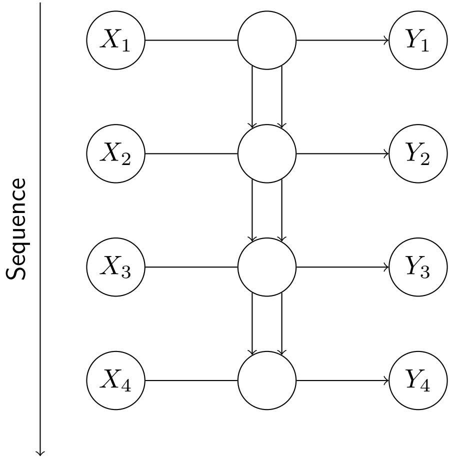


Figure: Neural Network with an LSTM Layer

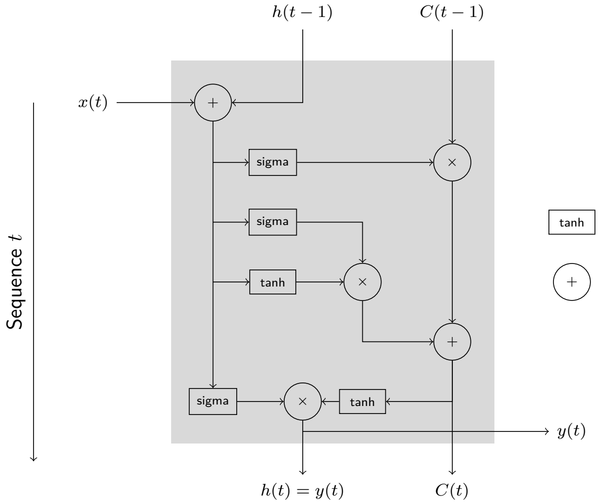
LSTM Cells

- Two connections along the sequence
  - h: The regular history of outcomes
    - I.e., the outcome of a neuron is passed into the neuron for the next sequence element
  - C: A state for the cell
    - Allows long-term storage

LSTM Cells

- Two connections along the sequence
  - h: The regular history of outcomes
    - I.e., the outcome of a neuron is passed into the neuron for the next sequence element
  - C: A state for the cell
    - Allows long-term storage
- Cell state is controlled within the cell
  - Forget: Previous state is removed
  - Input: Current input is (partially) stored in the cell state
  - Output: How much of the cell state is added to the cell output
- All ›gates‹ are controlled by weights, learned during training

An LSTM Cell

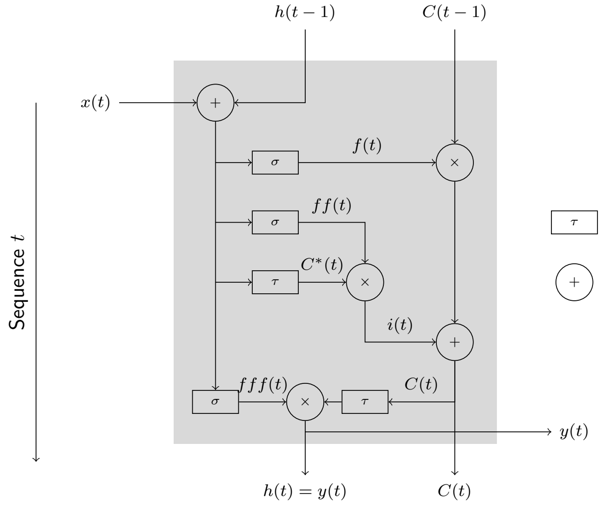


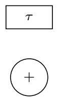
activation function
sigmoid: $[0;1]$
tanh: $[-1;1]$

arithmetical operator

An LSTM Cell

with labeled connections





activation function
$\sigma : [0;1]$
$\tau : [-1;1]$

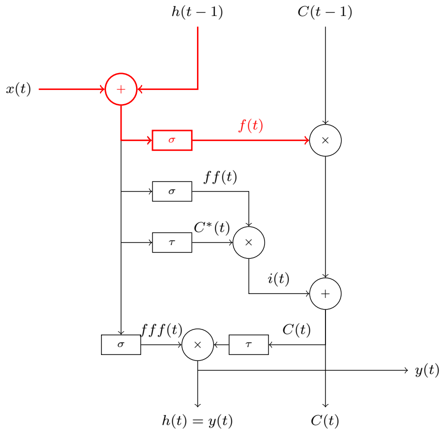
arithmetical operator

An LSTM Cell

Forget Gate

$$f(t) = \sigma\left(\vec{w}_f \times (x_t + h(t-1))\right)$$

- How much of the cell state do we forget?
- If $f(t) = 0$, cell state is emptied
- $\vec{w}_f$: Trainable weights for this gate



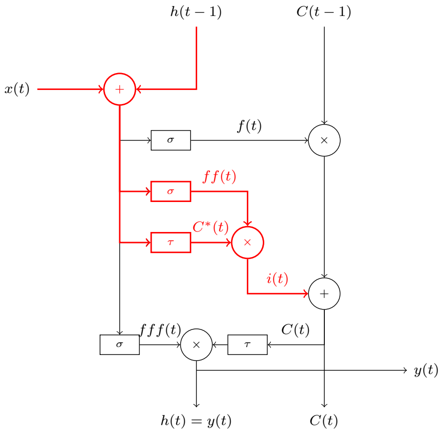
An LSTM Cell

Input Gate

How much of the current value is put into the cell state?

$$
\begin{aligned}
ff(t) &= \sigma\left(\vec{w}_{ff} \times (x_t + h(t-1))\right) \\
C^*(t) &= \tau\left(\vec{w}_C \times (x_t + h(t-1))\right) \\
i(t) &= ff(t) \times C^*(t)
\end{aligned}
$$

» $\vec{w}$: trainable weights



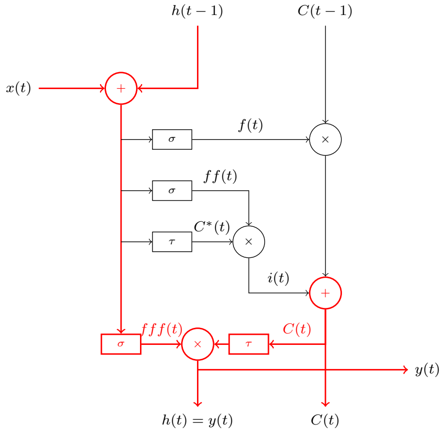
An LSTM Cell

Output Gate

How do we calculate the output(s) of the cell?

- Three outputs:
  - $y(t)$: regular output for the next layer
  - $h(t)$: passed on to the next sequence element
  - $C(t)$: new cell state

$$\begin{aligned} C(t) &= f(t) \times C(t-1) + i(t) \\ fff(t) &= \sigma\left(\vec{w}_{fff} \times (x_t + h(t-1))\right) \\ y(t) &= fff(t) \times \tau(C(t)) \end{aligned}$$



An LSTM Unit

Cell state $C(t)$

- A LSTM unit has a cell state (used for the long-term memory)
- Three gates control the state of the cell – each with its own weight
  - Forget gate $f(t)$: How much of the previous state is kept
    - $f(t) = \sigma(\vec{w}_f \times (x(t) + h(t\breve{}1)))$

An LSTM Unit

Cell state $C(t)$

- A LSTM unit has a cell state (used for the long-term memory)
- Three gates control the state of the cell – each with its own weight
  - Forget gate $f(t)$: How much of the previous state is kept
    - $f(t) = \sigma(\vec{w}_f \times (x(t) + h(t\breve{}1)))$
  - Input gate $ff(t)$, $C^*(t)$, $i(t)$: How much of the current state is stored
    - $ff(t) = \sigma(\vec{w}_{ff} \times (x(t) + h(t\breve{}1)))$, $C^*(t) = \tau(\vec{w}_C \times (x(t) + h(t\breve{}1)))$, $i(t) = ff(t) \times C^*(t)$

An LSTM Unit

Cell state $C(t)$

- A LSTM unit has a cell state (used for the long-term memory)
- Three gates control the state of the cell – each with its own weight
  - Forget gate $f(t)$: How much of the previous state is kept
    - $f(t) = \sigma(\vec{w}_f \times (x(t) + h(t\breve{}1)))$
  - Input gate $ff(t)$, $C^*(t)$, $i(t)$: How much of the current state is stored
    - $ff(t) = \sigma(\vec{w}_{ff} \times (x(t) + h(t\breve{}1)))$, $C^*(t) = \tau(\vec{w}_C \times (x(t) + h(t\breve{}1)))$, $i(t) = ff(t) \times C^*(t)$
  - Output gate $fff(t)$: What do we push to the next unit and what do we give out
    - $fff(t) = \sigma(\vec{w}_{fff}(x(t) + h(t\breve{}1)))$
    - $C(t) = f(t) \times C(t\breve{}1) + i(t)$
    - $h(t) = fff(t) \times \tau(C(t))$

An LSTM Unit

Cell state $C(t)$

- A LSTM unit has a cell state (used for the long-term memory)
- Three gates control the state of the cell – each with its own weight
  - Forget gate $f(t)$: How much of the previous state is kept
    - $f(t) = \sigma(\vec{w}_f \times (x(t) + h(t\breve{\ }1)))$
  - Input gate $ff(t)$, $C^*(t)$, $i(t)$: How much of the current state is stored
    - $ff(t) = \sigma(\vec{w}_{ff} \times (x(t) + h(t\breve{\ }1)))$, $C^*(t) = \tau(\vec{w}_C \times (x(t) + h(t\breve{\ }1)))$, $i(t) = ff(t) \times C^*(t)$
  - Output gate $fff(t)$: What do we push to the next unit and what do we give out
    - $fff(t) = \sigma(\vec{w}_{fff}(x(t) + h(t\breve{\ }1)))$
    - $C(t) = f(t) \times C(t\breve{\ }1) + i(t)$
    - $h(t) = fff(t) \times \tau(C(t))$
- Weights to be learned: $\vec{w}_f$, $\vec{w}_{ff}$, $\vec{w}_{fff}$, $\vec{w}_C$

LSTM in Keras

`layers.LSTM`

- Docs: `https://keras.io/api/layers/recurrent_layers/lstm/`
- `units` – Number of LSTM units to create
  - corresponds to `timesteps` for RNNs

LSTM in Keras

`layers.LSTM`

- Docs: `https://keras.io/api/layers/recurrent_layers/lstm/`
- `units` – Number of LSTM units to create
  - corresponds to `timesteps` for RNNs
- Bi-LSTM
  - Best performance for many tasks
  - `model.add(layers.Bidirectional(layers.LSTM(...)))`

Section 4

Exercise

Exercise 09

`https://github.com/IDH-Cologne-Deep-Learning-Uebung/exercise-09`