Applied Machine Learning
Lecture 14-1:
Special types of RNN: LSTM & GRU

Selpi (selpi@chalmers.se)

The slides are further development of Richard Johansson's slides

March 10, 2020

Overview

neural networks for sequential problems

Recurrent Neural Network (RNN)

Examples of RNN architectures

Long short term memory (LSTM)

Gated Recurrent Unit (GRU)

Review of this lecture

overview

- previously, we had an introduction to neural network classifiers
  - key attractions of this type of model is the reduced need of feature engineering
- We briefly discussed **recurrent** neural networks, which we apply to sequential data
  - classifying sentences or documents (e.g., predict positive/negative review)
  - outputting sequences, such as time series prediction
  - "sequence-to-sequence", such as machine translation
- Today: Discuss LSTM models

recap: feedforward NN

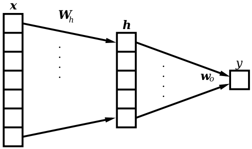
- a **feedforward neural network** or **multilayer perceptron** consists of connected layers of "classifiers"
  - the intermediate classifiers are called **hidden units**
  - the final classifier is called the **output unit**
- each hidden unit h_i computes its output based on its own weight vector w_{h_i}:

$$h_i = f(\mathrm{w}_{h_i} \cdot \mathrm{x})$$

- and then the output is computed from the hidden units:

$$y = f(\mathrm{w}_o \cdot \mathrm{h})$$

- the function f is called the **activation**



Overview

neural networks for sequential problems

Recurrent Neural Network (RNN)

Examples of RNN architectures

Long short term memory (LSTM)

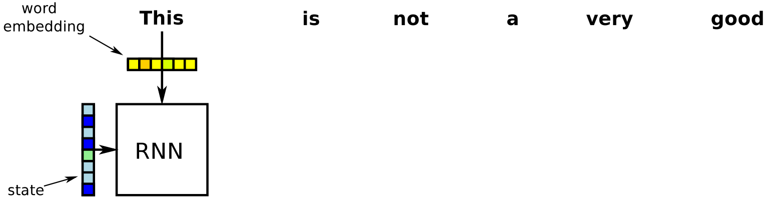
Gated Recurrent Unit (GRU)

Review of this lecture

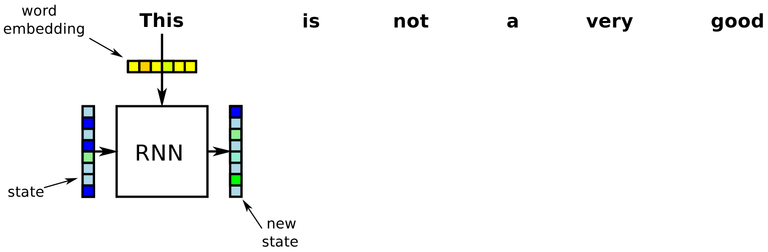
recap: recurrent NN for processing sequences

- **recurrent** NNs (RNNs) are applied in a step-by-step fashion to a sequential input such as a sentence
- RNNs use a **state** vector that represents what has happened previously
- after each step, a new state is computed

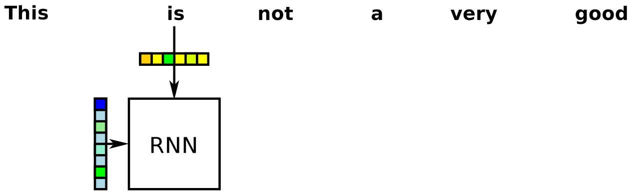
RNN example



RNN example

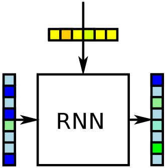


RNN example



RNN example

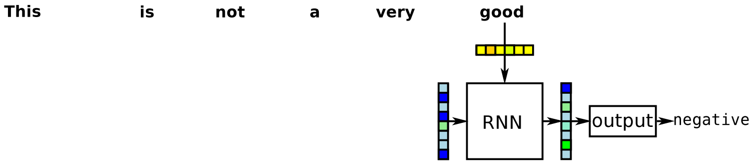
This is not a very good



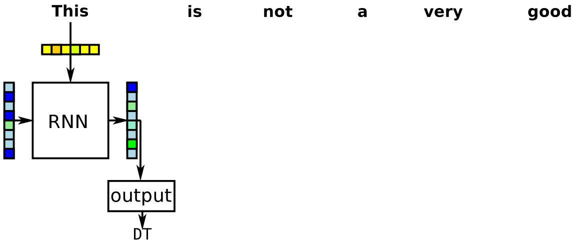
using the RNN output

- the RNN output isn't that interesting on its own ...
- we can use it
  - in sequence classifiers
  - in sequence predictors
  - in sequence taggers
  - in translation

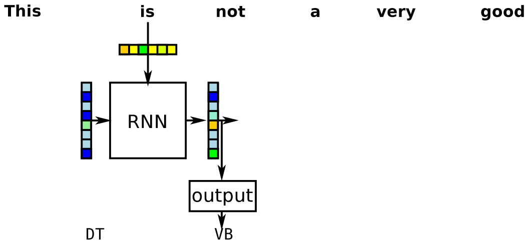
example: using the RNN output in a document classifier



example: using the RNN output in a part-of-speech tagger



example: using the RNN output in a part-of-speech tagger



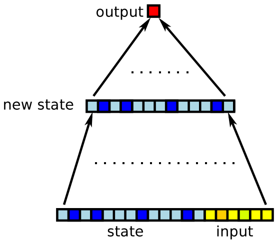
what's in the box? simple RNN implementation

- the simplest type of RNN looks similar to a feedforward NN
    - the next state is computed like a hidden layer in a feedforward NN

$$\text{state}_t = \sigma(W_h \cdot (\text{input} \oplus \text{state}_{t-1}))$$

    - ...and the output from the next state

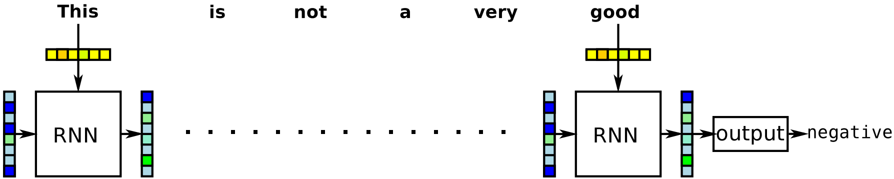
$$\text{output} = \text{softmax}(W_o \cdot \text{state}_t)$$



simple RNNs in Keras

```
model = Sequential()
model.add(SimpleRNN(32))
```

training RNNs: backpropagation through time



Overview

neural networks for sequential problems

Recurrent Neural Network (RNN)

Examples of RNN architectures

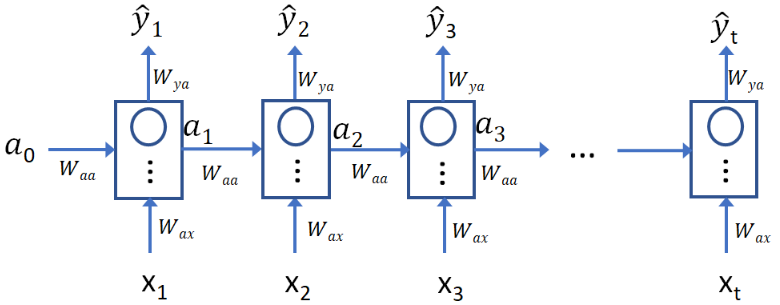
Long short term memory (LSTM)

Gated Recurrent Unit (GRU)

Review of this lecture

Many-to-many architectures(1)

- For input and output sequences of same length

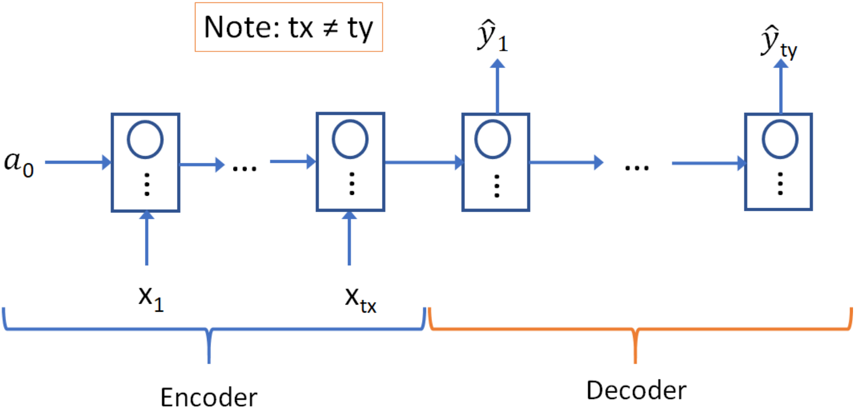


$a_t = g_1(W_{aa}a_{t\text{-}1} + W_{ax}x_t + b_a)$; g1 can be tanh or ReLu
$\hat{y}_t = g_2(W_{ya}a_t + b_y)$; common g2 is sigmoid

$a_t = g_1(W_a[a_{t\text{-}1}, x_t] + b_a)$; g1 can be tanh or ReLu
$\hat{y}_t = g_2(W_y a_t + b_y)$; common g2 is sigmoid

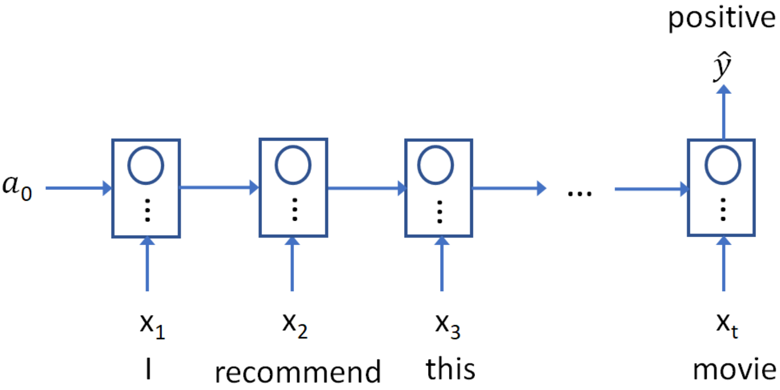
Many-to-many architectures(2)

- For input and output sequences of different lengths
- For example, machine translation for sentiment/document classification, input: sequence, output: positive/negative



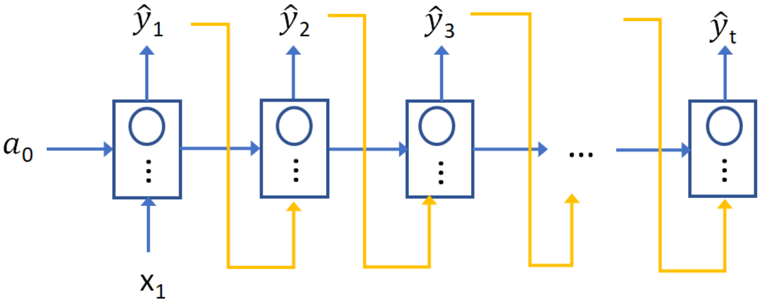
Many-to-one

- For example, for sentiment classification, input: sequence and output: positive/negative. For video-based activity recognition, input: sequence of video frames, output: type of activity



One-to-many

- For example, for music generation, input: genre, output: sequence of notes



simple RNNs have a drawback

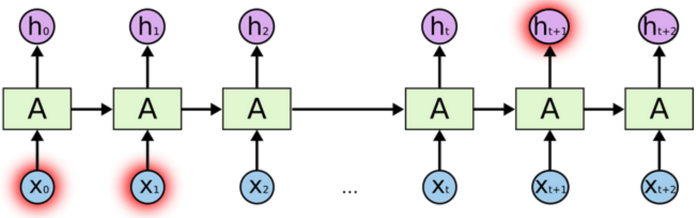


image borrowed from C. Olah's blog

Overview

neural networks for sequential problems

Recurrent Neural Network (RNN)

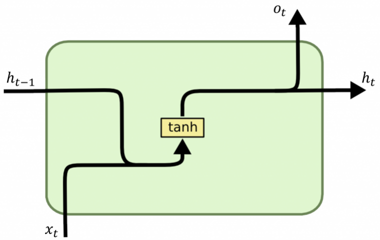
Examples of RNN architectures

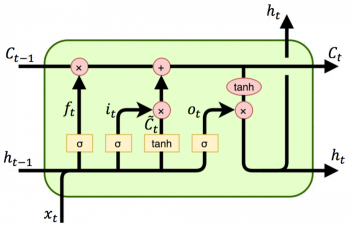
Long short term memory (LSTM)

Gated Recurrent Unit (GRU)

Review of this lecture

Simple RNN vs LSTM





images borrowed from C. Olah's blog

long short-term memory: LSTM

- the **long short-term memory** is a type of RNN that is designed to handle long-term dependencies in a better way

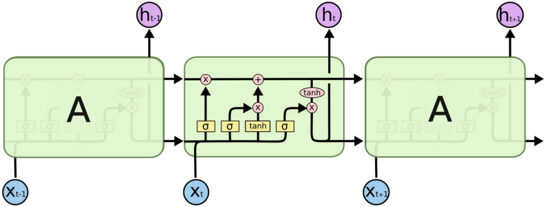
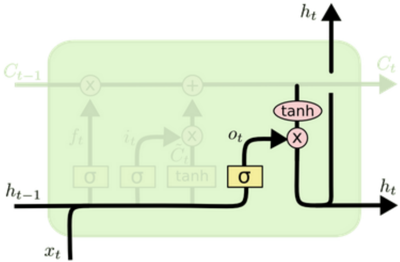


image borrowed from C. Olah's blog

- its state consists of two parts – a **short-term** and a **long-term** part (also known as the **cell state**)
- note that the exact implementation can differ slightly in literature!

LSTM: short-term part and output

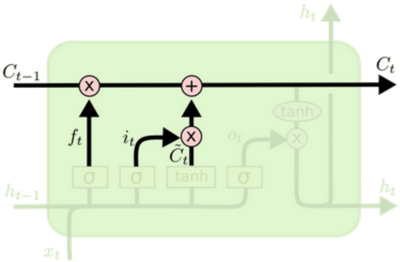


$$o_t = \sigma \left(W_o \; [h_{t-1}, x_t] \; + \; b_o \right)$$
$$h_t = o_t * \tanh \left(C_t \right)$$

image borrowed from C. Olah's blog

- short-term state is a bit similar to what we had in the simple RNN

LSTM: long-term part



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

image borrowed from C. Olah's blog

- the long-term state is controlled by a **forget gate** that decides if information should be passed along or discarded

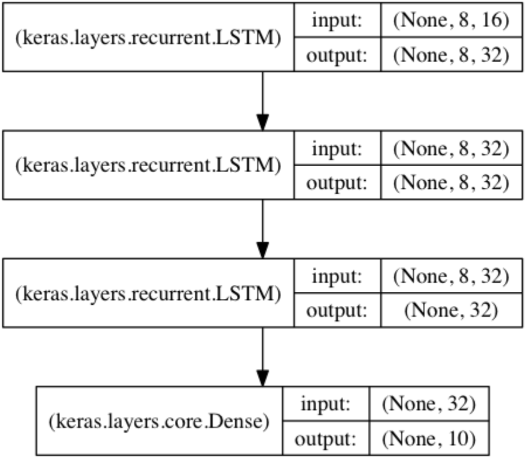
LSTM equations

$$\tilde{C}_t = \tanh(W_c[h_{t-1}, x_t] + b_c)$$
$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$$
$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$$
$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$
$$C_t = i_t * \tilde{C}_t + f_t * C_{t-1}$$
$$h_t = o_t * \tanh(C_t)$$

LSTMs in Keras

```
from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras.layers import Embedding
from keras.layers import LSTM

model = Sequential()
model.add(Embedding(max_features, output_dim=256))
model.add(LSTM(128))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))

model.compile(loss='binary_crossentropy',
              optimizer='adam')

model.fit(x_train, y_train, batch_size=16, epochs=10)
score = model.evaluate(x_test, y_test, batch_size=16)
```

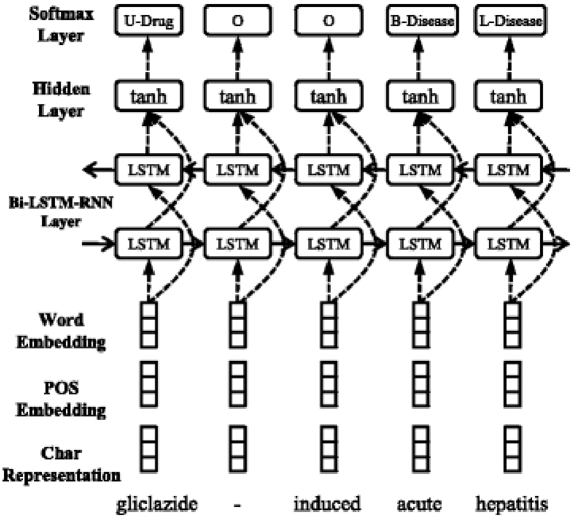
stacked LSTMs



stacked LSTMs in Keras

```
model = Sequential()

# returns a sequence of vectors of dimension 32
model.add(LSTM(32, return_sequences=True,
               input_shape=(timesteps, data_dim)))

# returns a sequence of vectors of dimension 32
model.add(LSTM(32, return_sequences=True))

# return a single vector of dimension 32
model.add(LSTM(32))

model.add(Dense(10, activation='softmax'))
```

Bidirectional LSTM for sequence tagging

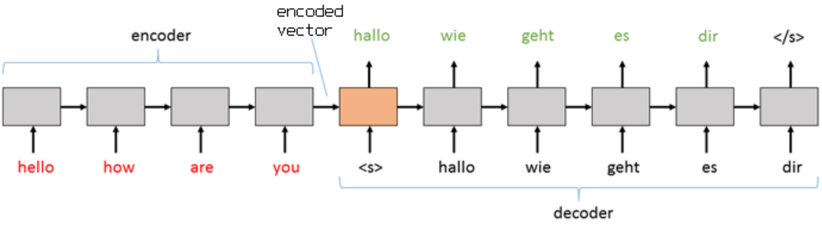


[source]

example: NER using LSTMs

Encoder-Decoder sequence-to-sequence LSTM model for translation

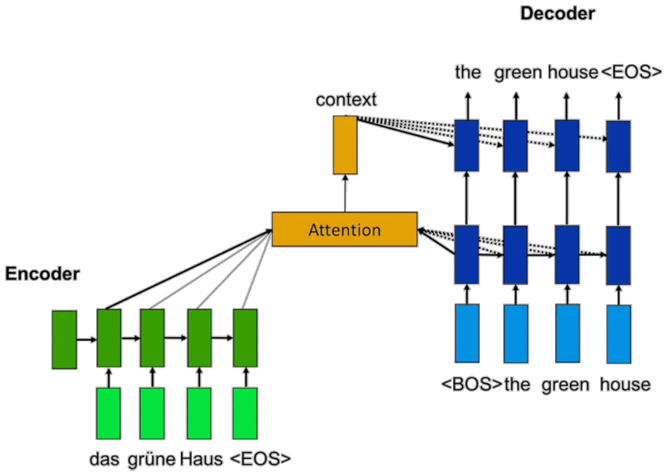
- The whole input is encoded into a single and fixed length context vector.
- The encoder's hidden states are discarded.
- The output of the encoder is more influenced by the recent tokens rather than earlier tokens.



Sutskever et al. (2014) *Sequence to Sequence Learning with Neural Networks*.

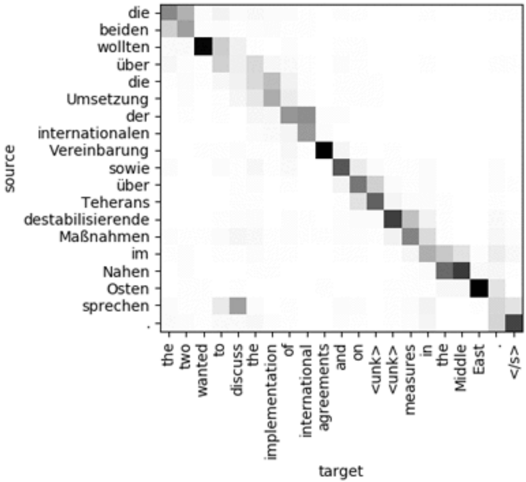
Attention model for translation

- It uses the encoder's hidden states.
- It learns which parts of the input sequence to pay attention to.



[source]

example: visualizing attention

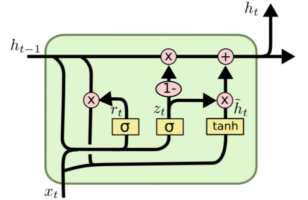


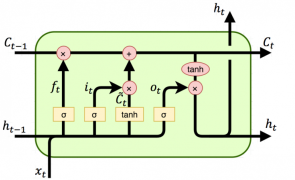
[source]

Overview

GRU vs LSTM - equations

- Gated Recurrent Unit (GRU) is a special type of RNN.





$$\tilde{h}_t = \tanh(W_h[r_t * h_{t\text{-}1}, x_t] + b_h)$$
$$z_t = \sigma(W_z[h_{t\text{-}1}, x_t] + b_z)$$
$$r_t = \sigma(W_r[h_{t\text{-}1}, x_t] + b_r)$$
$$h_t = z_t * \tilde{h}_t + (1 - z_t) * h_{t\text{-}1}$$

$$\tilde{C}_t = \tanh(W_c[h_{t\text{-}1}, x_t] + b_c)$$
$$i_t = \sigma(W_i[h_{t\text{-}1}, x_t] + b_i)$$
$$f_t = \sigma(W_f[h_{t\text{-}1}, x_t] + b_f)$$
$$o_t = \sigma(W_o[h_{t\text{-}1}, x_t] + b_o)$$
$$C_t = i_t * \tilde{C}_t + f_t * C_{t\text{-}1}$$
$$h_t = o_t * \tanh(C_t)$$

images borrowed from C. Olah's blog

Overview

neural networks for sequential problems

Recurrent Neural Network (RNN)

Examples of RNN architectures

Long short term memory (LSTM)

Gated Recurrent Unit (GRU)

Review of this lecture

Review of this lecture

- LSTM and GRU are special types of RNN. How do they differ to the simple RNN? How do they differ from each other?
- Explain the differences between LSTM and GRU! GRU was invented after LSTM. What might be the driving forces for the invention and the usage of GRU, when LSTM is known to be good?
- Explain the different types of RNN architectures and describe the type of application where each type of architecture could be useful!