Modelling Time Series with Neural Networks

Volker Tresp

Summer 2017

Modelling of Time Series

- The next figure shows a time series (DAX)
- Other interesting time-series: energy prize, energy consumption, gas consumption, copper prize, ...

DAX Performance-Index

01.06.07 17:45 Uhr • 7.987,85 +1,33 % [+104,81]	Enthaltene Werte: **30** Tages-Vol.: **9,12 Mrd.**	Typ: **Index** Börse: XETRA

Neural Networks for Time-Series Modelling

- Let $z_t, t = 1, 2, \ldots$ be the time-discrete time-series of interest (example: DAX)
- Let $x_t, t = 1, 2, \ldots$ denote a second time-series, that contains information on z_t (Example: Dow Jones)
- For simplicity, we assume that both z_t and x_t are scalar. The goal is the prediction of the next value of the time-series
- We assume a system of the form

$$z_t = f(z_{t-1}, \ldots, z_{t-T}, x_{t-1}, \ldots, x_{t-T}) + \epsilon_t$$

with i.i.d. random numbers $\epsilon_t, t = 1, 2, \ldots$ which model unknown disturbances.

Neural Networks for Time-Series Modelling (cont'd)

- We approximate, using a neural network,

$$f(z_{t-1}, \ldots, z_{t-T}, x_{t-1}, \ldots, x_{t-T})$$

$$\approx f_{\mathbf{w},V}(z_{t-1}, \ldots, z_{t-T}, x_{t-1}, \ldots, x_{t-T})$$

  and obtain the cost function

$$\text{cost}(\mathbf{w}, V) = \sum_{t=1}^{N} (z_t - f_{\mathbf{w},V}(z_{t-1}, \ldots, z_{t-T}, x_{t-1}, \ldots, x_{t-T}))^2$$

- The neural network can be trained as before with simple back propagation *if in training all z_t and all x_t are known*!

- This is a NARX model: **N**onlinear **A**uto **R**egressive Model with e**x**ternal inputs. Another name: TDNN (time-delay neural network).

- Note the "convolutional" idea in TDNNs

- Language model: last T words as input. The task is to predict the next word in a sentence. One-hot encoding.

Mutiple-Step Prediction

- Predicting more than one time step in the future is not trivial
- The future inputs are not available
- The model noise needs to be properly considered in multiple step prediction (for example by a stochastic simulation); if possible one could also simulate future inputs (multivariate prediction)
- Teacher forcing: Use the predicted output as input (can be risky)

Recurrent Neural Network

- Recurrent Neural Networks are powerful methods for time series and sequence modelling

Generic Recurrent Neural Network Architecture

- Consider a feedforward neural network where there are connections between the hidden units

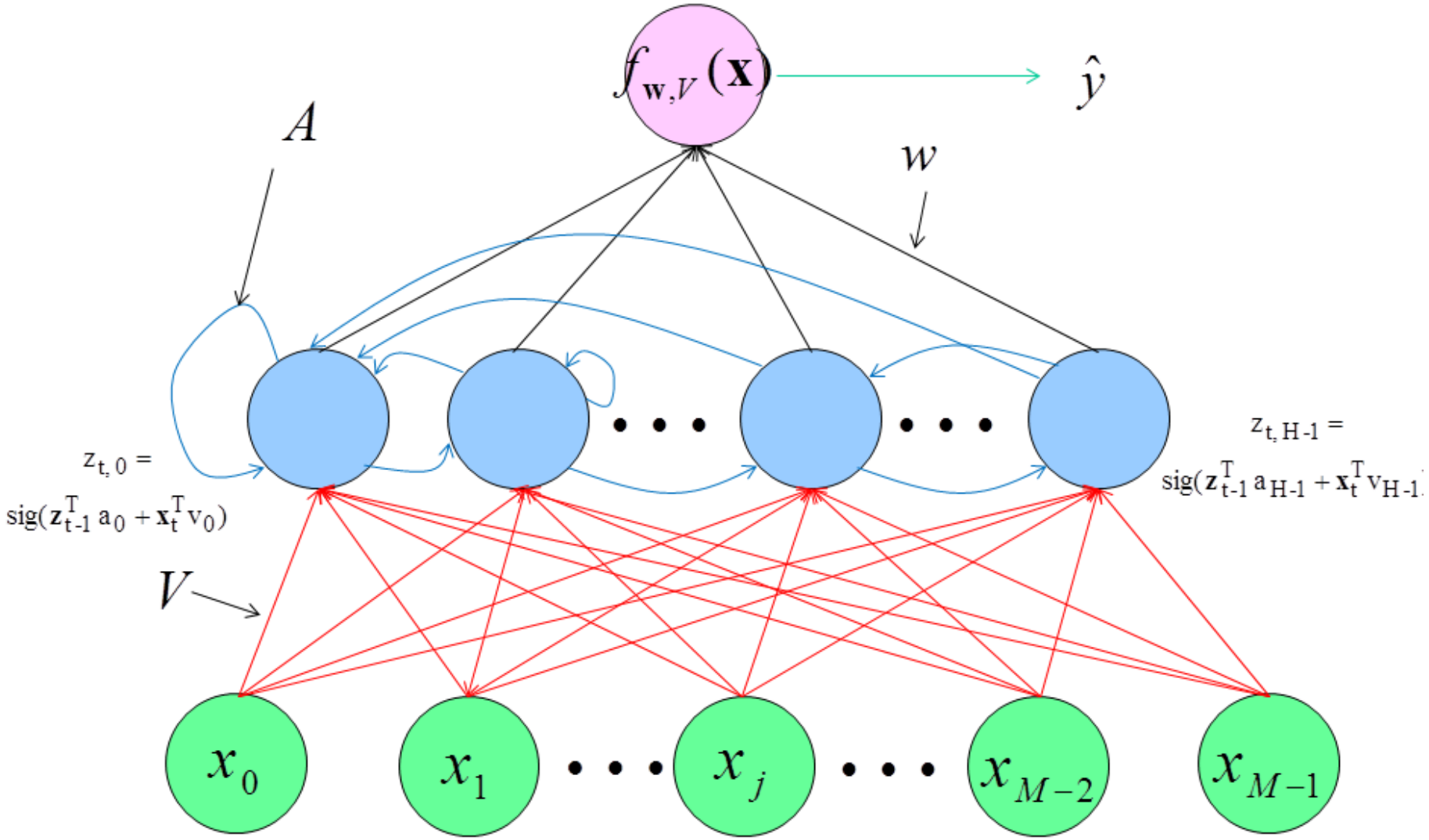
$$z_{t,h} = \text{sig}(\mathbf{z}_{t-1}^T \mathbf{a}_h + \mathbf{x}_t^T \mathbf{v}_h)$$

and, as before,

$$\widehat{y}_t = \text{sig}(\mathbf{z}_t^T \mathbf{w})$$

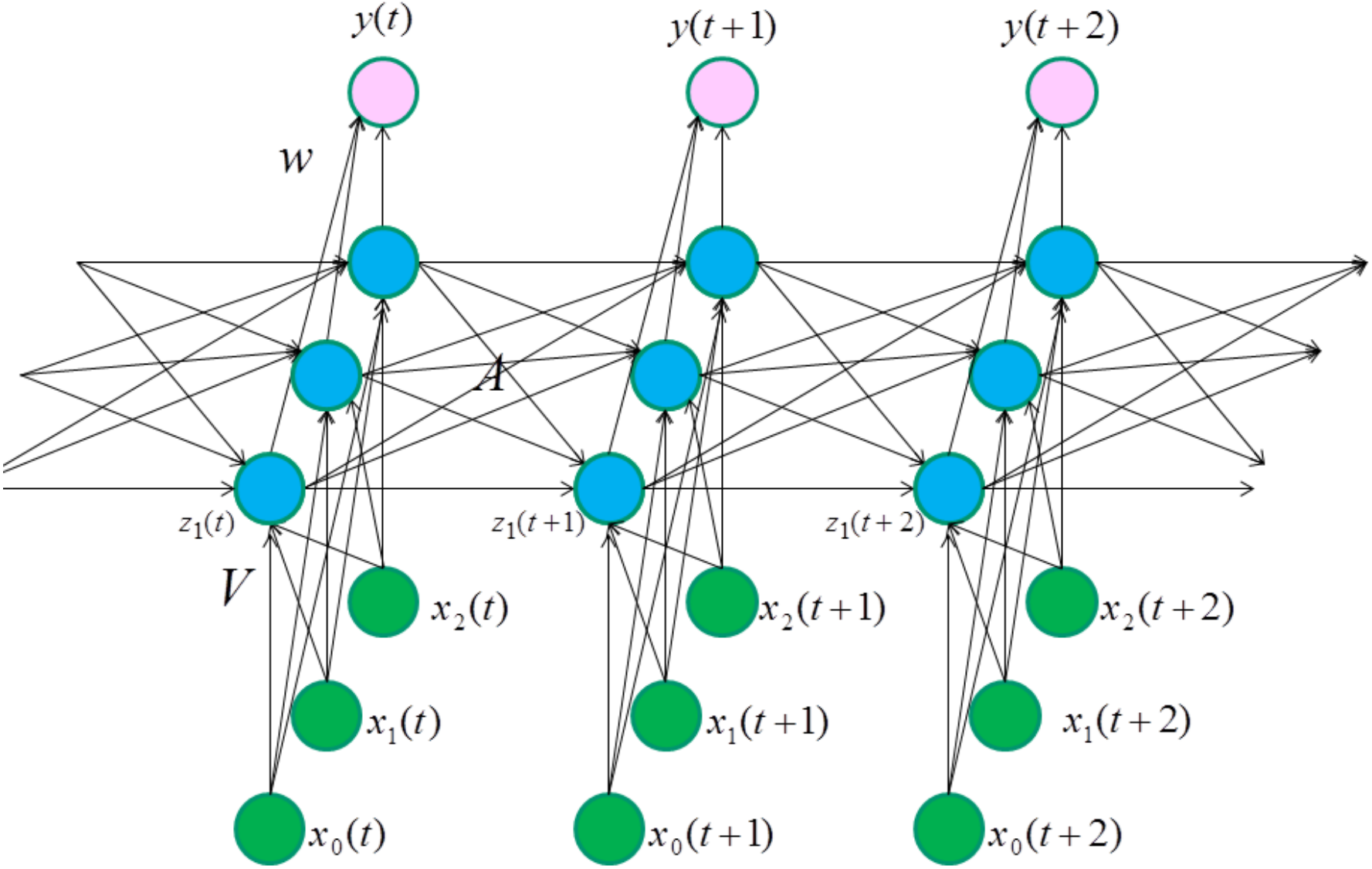
- Here, $\mathbf{z}_t = (z_{t,1}, z_{t,2}, \ldots, z_{t,H})^T$, $\mathbf{x}_t = (x_{t,0}, x_{t,1}, \ldots, x_{t,M-1})^T$
- In Recurrent Neural Networks (RNNs) the next state of the neurons in the hidden layer depends on their last state and both are not directly measured
- $\mathbf{a}_h, \mathbf{w}, \mathbf{v}_h$ are weight vectors
- Note that in must applications, one is interested in the output y_t (and not in $z_{t,h}$)
- The next figure shows an example. Only some of the recurrent connections are shown (blue). The blue connections also model a time lag. Without recurrent connections ($\mathbf{a}_h = 0, \forall h$), we obtain a regular feedforward network

- Note, that a recurrent neural network has an internal memory


$f_{\mathbf{w},V}(\mathbf{x})$
$\hat{y}$
A
w
V
$z_{t,0} = \text{sig}(\mathbf{z}_{t-1}^T a_0 + \mathbf{x}_t^T v_0)$
$z_{t,H-1} = \text{sig}(\mathbf{z}_{t-1}^T a_{H-1} + \mathbf{x}_t^T v_{H-1})$
x_0
x_1
x_j
x_{M-2}
x_{M-1}

A Recurrent Neural Network Architecture unfolded in Time

- The same RNN but with a different intuition
- Consider that at each time-step a feedforward Neural Network predicts outputs based on some inputs
- In addition, the hidden layer also receives input from the hidden layer of the previous time step
- Without the nonlinearities in the transfer functions, this is a linear state-space model; thus a RNN is a nonlinear state-space model


y(t)
y(t+1)
y(t+2)
w
A
V
z1(t)
z1(t+1)
z1(t+2)
x2(t)
x2(t+1)
x2(t+2)
x1(t)
x1(t+1)
x1(t+2)
x0(t)
x0(t+1)
x0(t+2)

Training of Recurrent Neural Network Architecture

- Backpropagation through time (BPTT): essentially backpropagation applied to the unfolded network; note that all that happened before time t influences $\widehat{y}_t$, so the error needs to be backpropagated backwards in time, in principle until the beginning of the experiment! In reality, one typically truncates the gradient calculation (review in: Werbos (1990))
- Real-Time Recurrent Learning (RTRL) (Williams and Zipser (1989))
- Time-Dependent Recurrent Back-Propagation: learning with continuous time (Lagrangian approach) (Pearlmutter 1998)

Echo-State Network

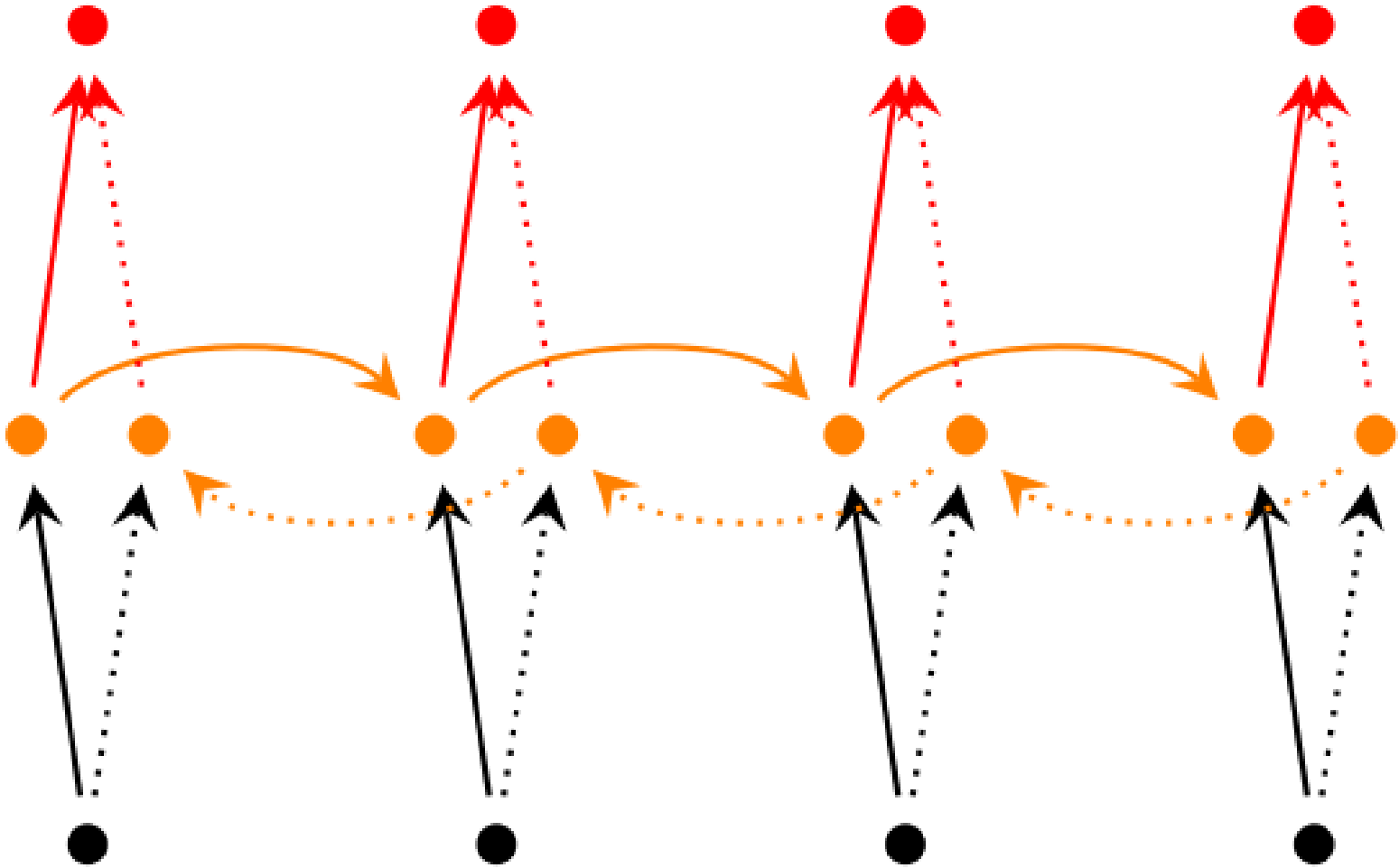
- Recurrent Neural Networks are notoriously difficult to train
- A simple alternative is to initialize A and V randomly (according to some recipe) and only train $\mathbf{w}$, e.g., with the ADALINE learning rule
- This works surprisingly well and is done in the Echo-State Network (ESN)

Iterative Prediction

- Assume a trained model where the prediction is: $\widehat{y}_t \rightarrow (x_t, y_t) \rightarrow \widehat{y}_{t+1}, \ldots$
- Thus we predict (e.g., the DAX of the next day) and the obtain a measurement of the next day
- A RNN would ignore the new measurement. What can be done
- 1: In probabilistic models, the measurement can change the hidden state estimates accordingly (HMM, Kalman filter, particle filter, ...)
- 2: We can use y_t as an input to the RNN (as in TDNN)
- 3: We add a (linear) noise model

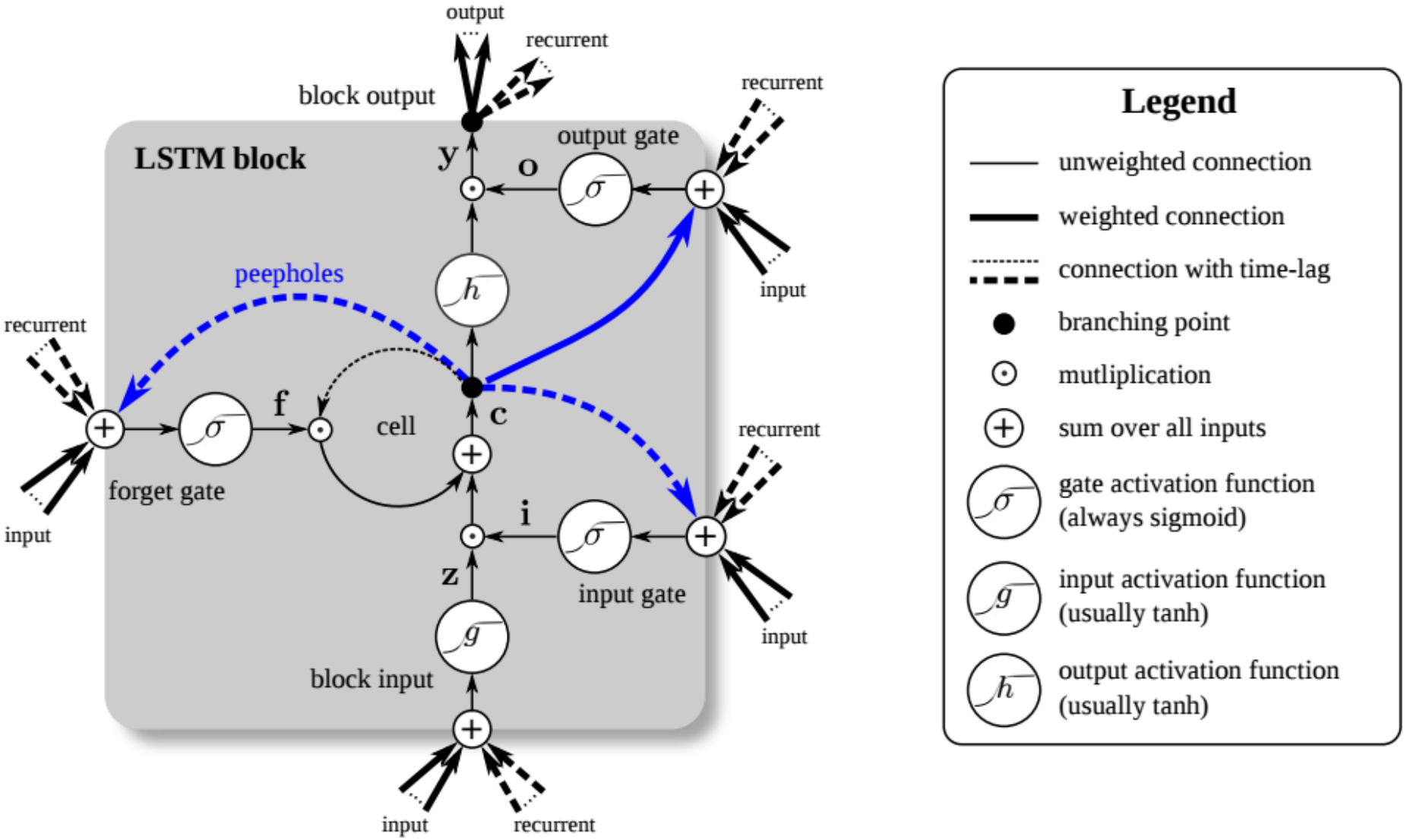
Bidirectional RNNs

- The predictions in bidirectional RNNs depend on past and future inputs
- Useful for sequence labelling problems: handwriting recognition, speech recognition, bioinformatics, ...



Long Short Term Memory (LSTM)

- As a recurrent structure the Long Short Term Memory (LSTM) approach has been very successful
- Basic idea: at time T a newspaper announces that the Siemens stock is labelled as "buy". This information will influence the development of the stock in the next days. A standard RNN will not remember this information for very long. One solution is to define an extra input to represent that fact and that is on as along as "buy" is valid. But this is handcrafted and does not exploit the flexibility of the RNN. A flexible construct which can hold the information is a long short term memory (LSTM) block.
- The LSTM was used very successful for reading handwritten text and is the basis for many applications involving sequential data (NLP, translation of text, ...)


output
recurrent
block output
recurrent
output gate
LSTM block
y
o
peepholes
input
recurrent
h
recurrent
f
cell
c
forget gate
input
recurrent
i
input gate
z
input
g
block input
input
recurrent
Legend
unweighted connection
weighted connection
connection with time-lag
branching point
mutliplication
sum over all inputs
gate activation function (always sigmoid)
input activation function (usually tanh)
output activation function (usually tanh)

LSTM in Detail

- The LSTM block replaces one hidden unit z_h, together with its input weights $\mathbf{a}_h$ and $\mathbf{v}_h$. In general all H hidden units are replaced by H LSTM blocks. It produces one output z_h (in the figure it is called y)
- All inputs in the figure are weighted inputs
- Thus in the figure z would be the regular RNN-neuron output with a $\tanh$ transfer function
- Three gates are used that control the information flow
- The input gate (one parameter) determines if z should be attenuated
- The forget gate (one parameter) determines if the last z should be added and with which weight
- Then another $\tanh$, modulated by the output gate
- See http://www.wildml.com/2015/10/recurrent-neural-network-tutorial-part-4-implementing-a-grulstm-rnn-with-python-and-theano/

LSTM Applications

- Wiki: LSTM achieved the best known results in unsegmented connected handwriting recognition, and in 2009 won the ICDAR handwriting competition. LSTM networks have also been used for automatic speech recognition, and were a major component of a network that in 2013 achieved a record 17.7% phoneme error rate on the classic TIMIT natural speech dataset
- Applications: Robot control, Time series prediction, Speech recognition, Rhythm learning, Music composition, Grammar learning, Handwriting recognition, Human action recognition, Protein Homology Detection

Gated Recurrent Units (GRUs)

- Some people found LSTMs too complicated and invented GRUs with fewer gates

Encoder Decoder Architecture

- For example, used in machine translation