LEARNING CONTEXT SENSITIVE LANGUAGES WITH LSTM TRAINED WITH KALMAN FILTERS

Felix A. Gers (Mantik, Germany)

Juan Antonio Pérez-Ortiz (U. Alacant, Spain) *

Douglas Eck (IDSIA, Switzerland)

Jürgen Schmidhuber (IDSIA, Switzerland)

Overview

- The problem: learning the $a^n b^n c^n$ language with a recurrent neural network
- Previous works used:
  - Some traditional recurrent networks with poor results
  - Long Short-Term Memory (LSTM), a novel recurrent architecture, with much better results
- Training algorithm was gradient descent in all cases
- What about an algorithm based on the Kalman filter applied to the LSTM neural network?

The language

- $a^n b^n c^n$: a context sensitive language far from the regular languages
- Main difficulty: long-term dependencies $\rightarrow$ vanishing gradients in traditional recurrent nets
- Does the network learn the training set? Does it generalize?

Language learning

- Two approaches: classification vs prediction
- Classification task:
  - Training set consists of positive and negative samples
  - The net is trained to accept or reject them
- Prediction task:
  - Training set consists of positive samples only
  - The net is trained to predict the next-symbol

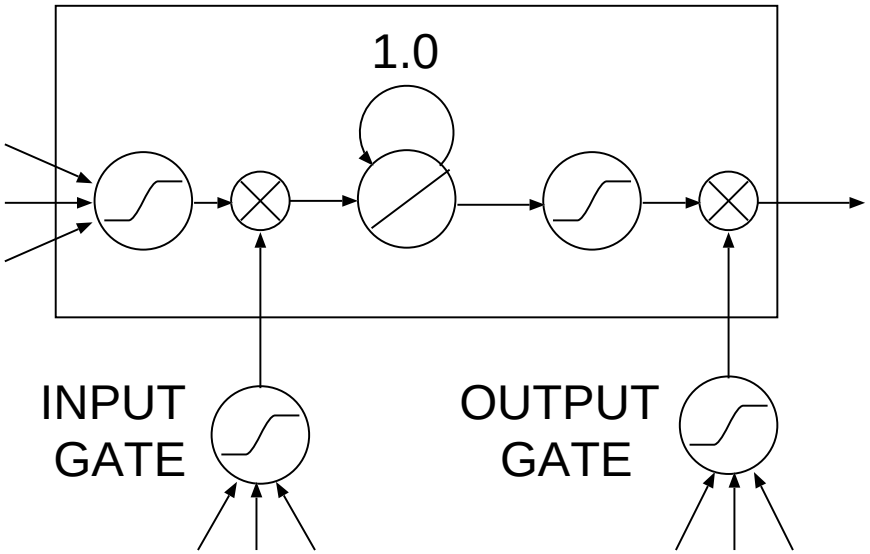
Previous work on $a^n b^n c^n$

- Traditional recurrent networks:
  - Second-order sequential cascaded network (Bodén and Wiles, 2000) and simple recurrent net (Chalup and Blair, 1999)
  - $a^n b^n c^n$ task: hard training, poor generalization
- LSTM (Gers and Schmidhuber, 2001):
  - $a^n b^n c^n$ task: easier learning, good generalization
  - LSTM solves lot of problems unsolvable by traditional nets
  - Architecture and computation of derivatives make long-term dependencies much easier to deal with

LSTM memory cells

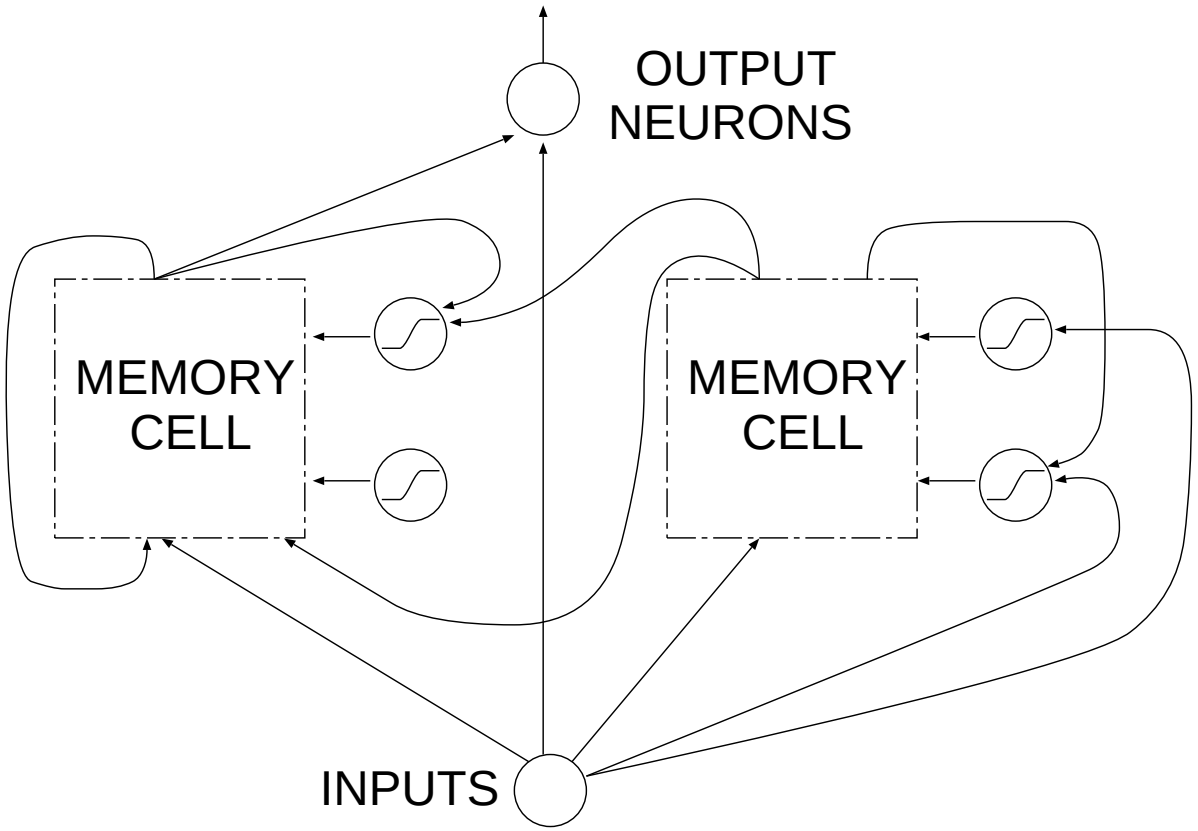
- The memory cell is the basic component of LSTM: a self-recurrent linear unit surrounded by a set of gates

- The linear unit of the cell can latch information during long periods of time: the vanishing gradient problem vanishes



LSTM network

- Hidden layer of a LSTM network consists of a set of memory cells; output neurons are classical neurons



Our contribution

- All previous approaches considered gradient descent (GD) for weight updating
- The decoupled extended Kalman filter (DEKF) improved LSTM performance in all the tasks we studied before
- What are the results of DEKF when applied to LSTM in the $a^n b^n c^n$ task?

Decoupled extended Kalman filter (DEKF)

- Gradient descent (GD) is usually slow because it depends on instantaneous estimations of the gradient: training history is not taken into account

- Training algorithms based on the Kalman filter consider recursively and efficiently all the information computed until now

- DEKF (Puskurious and Feldkamp, 1994) is based on the extended (nonlinear) Kalman filter

Results for $n \in [1, 10]$

- Architectures with roughly the same number of weights (≈ 70)

- Training set: $a^n b^n c^n$ with $n \in [1, 10]$

Model	Training strings	% learnt	Mean generalization	Best generalization
Traditional + GD	20 000	8	[1,12]	[1,18]
LSTM + GD	45 000	90	[1,28]	[1,52]
LSTM + DEKF	2 000	100	[1,434]	[1,2743]

Results for $n \in [20, 21]$

- A harder task (without the support of shortest strings)

- Training set: $a^n b^n c^n$ with $n \in [20, 21]$

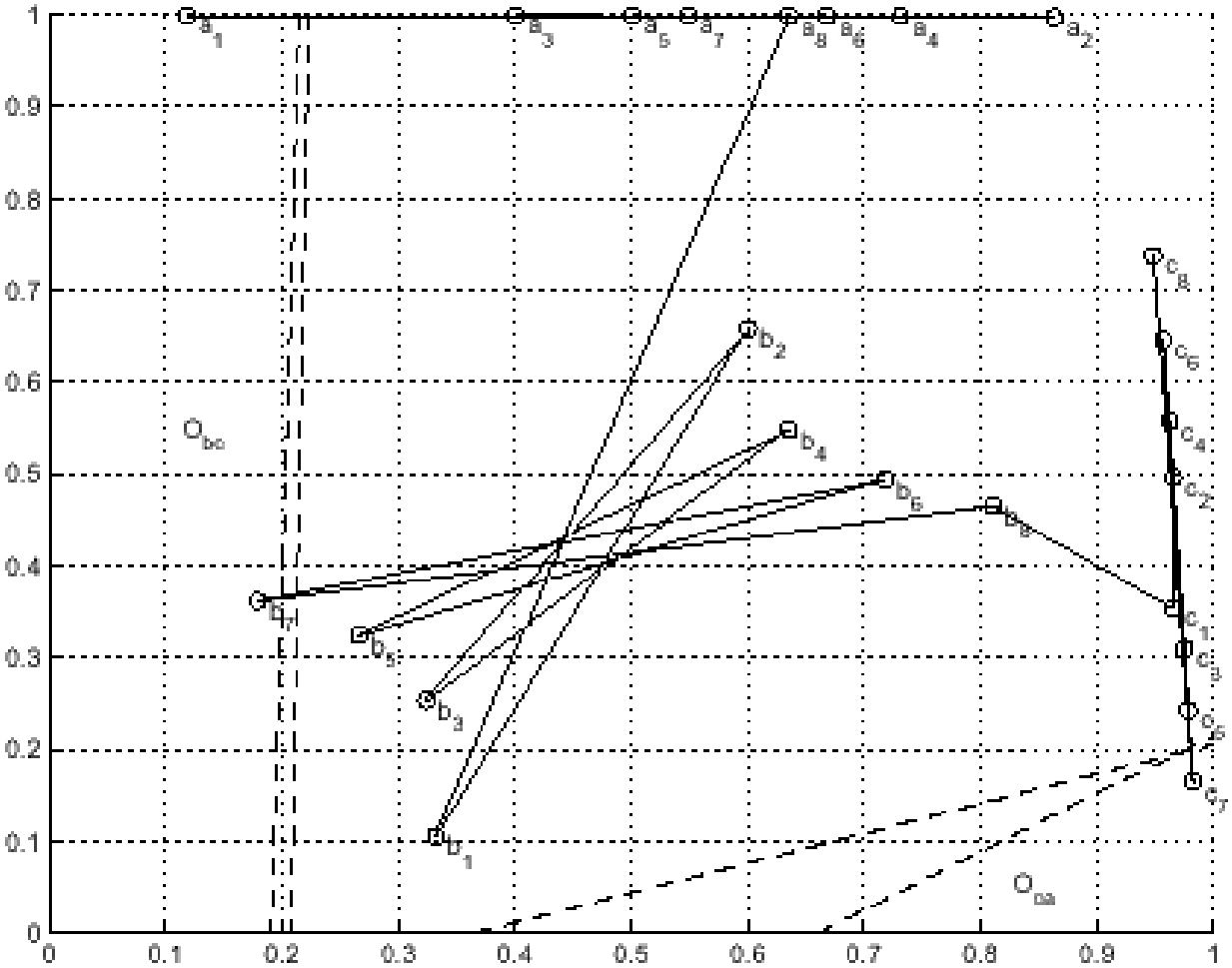
Model	Training strings	% learnt	Mean generalization	Best generalization
LSTM + GD	127 000	20	[17,23]	[10,27]
LSTM + DEKF	5 000	50	[12,29]	[8,34]

Best result for $n \in [1, 10]$

- Best result over the LSTM networks trained on the strings from $a^n b^n c^n$ with $n \in [1, 10]$
- Training set completely learnt after only 2 000 training strings
- Generalization: $n \in [1, 22\,463\,683]$
- An extremely good result!

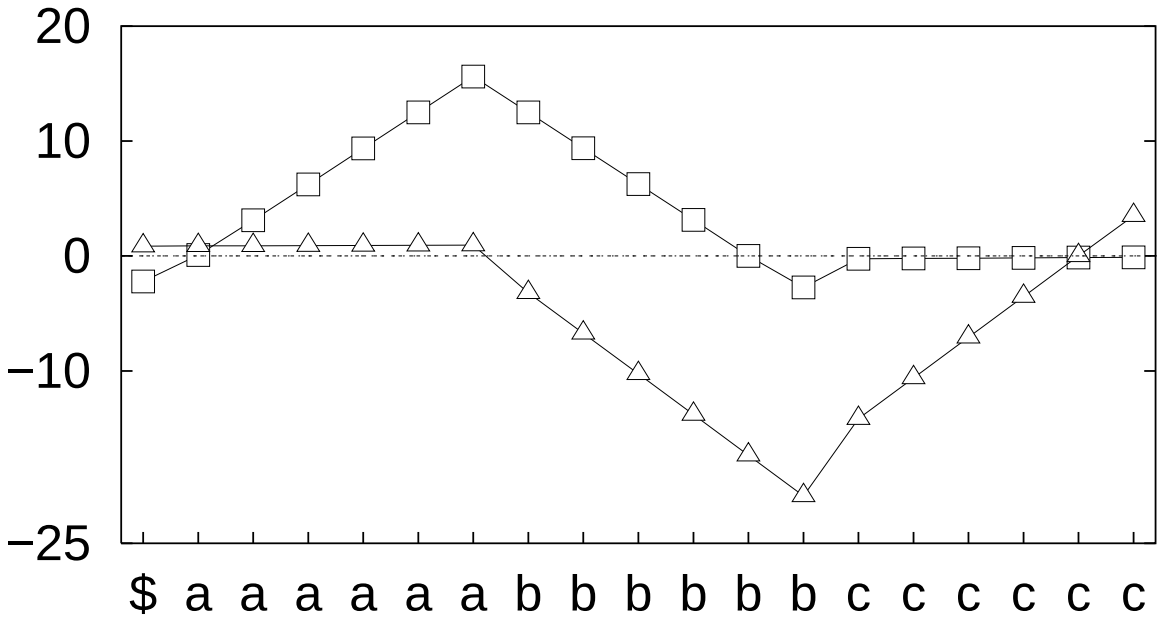
Solution found by traditional recurrent networks

- Oscillations around fixed points in the bounded state space



Solution found by LSTM

- Monotonic counters in the unbounded state space
- One memory cell solves $a^n b^n$ and the other solves $b^n c^n$



Conclusions

- LSTM with DEKF needs orders of magnitude fewer training sequences, learns more often, and generalizes even better than standard LSTM when learning $a^n b^n c^n$

- The additional time complexity of DEKF is largely compensated for by the faster learning velocity

- DEKF takes one step further LSTM performance with respect to traditional recurrent networks