Recurrent Neural Networks

Peter Tino
School of Computer Science
University of Birmingham
UK

Key idea

State space formulation

- (Input conditional) state transitions: $\mathbf{x}(t) = f(\mathbf{u}(t), \boldsymbol{x}(t-1))$
- Output readout from the state: $\mathbf{y}(t) = h(\mathbf{x}(t))$

Realise the mappings f and h as parts of a neural network:

$$\textbf{State} \quad x_i(t) = \sigma\left(\sum_j W_{ij}^{RI} \cdot u_j(t) + \sum_j W_{ij}^{RC} \cdot x_j(t-1) + T_i^R\right)$$

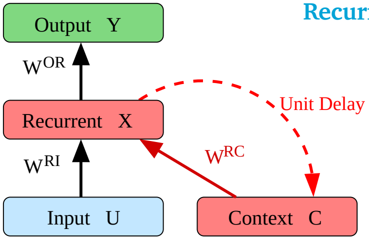
$$\textbf{Output} \quad y_i(t) = \sigma\left(\sum_j W_{ij}^{OR} \cdot x_j(t) + T_i^O\right)$$

$\boldsymbol{W}^{RI}$, $\boldsymbol{W}^{RC}$, $\boldsymbol{W}^{OR}$ and $\boldsymbol{T}^R$, $\boldsymbol{T}^O$ are real-valued connection weight matrices and threshold vectors, respectively, σ is a sigmoid activation function, e.g. $\sigma(a) = (1 + \exp(-a))^{-1}$

Recurrent Neural Network (Elman)

Normal NN: Feed-forward NN

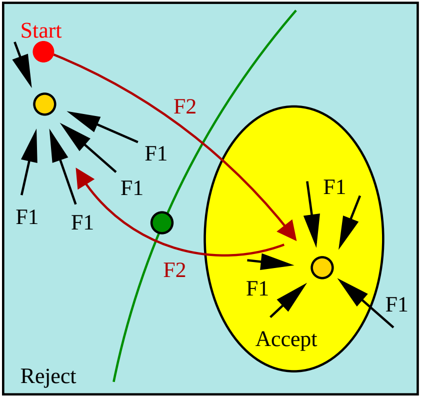
Recurrent NN is a bit different

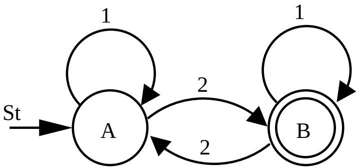


- Neurons organized in 4 groups: Input (U), Context (C), Recurrent (X) and Output (Y)
- X+C can be considered an "extended" input to the network
- C is an exact copy of X from the previous time step

Attractive sets

To latch a piece of information for a potentially unbounded number of time steps we need attractive sets





Grammatical:

all strings containg odd number of 2's

Information latching problem

1. To latch an important piece of information for future reference we need to create an attractive set
2. But derivatives of the state-transition map are small in the neighborhood of such an attractive set
3. We cannot propagate error information when training RNN via a gradient-based procedure – derivatives decrease exponentially fast through time

Curse of long time spans

- As soon as we try to create a mechanism for keeping important bits of information from the past, we lose the ability to set the parameters to the appropriate values since the error information gets lost very fast
- It is actually quite non-trivial (although not impossible) to train a parametrized state space model (e.g. RNN) beyond finite memory on non-toy problems

Dealing with vanishing gradients

- The problems arise from
  1. the unobservable nature of states in the hidden recurrent layer(s) of the RNN (we do not know from the series of inputs what they ideally should be - exponentially exploding set of possibilities)
  2. information latching problem.

- Several attempts to allow efficient propagation of gradients through time - most popular is the Long Short Term Memory (LSTM) model

- Implement internal mechanism to manipulate gradient flow through time. Closely related to gradient propagation through deep neural network architectures.

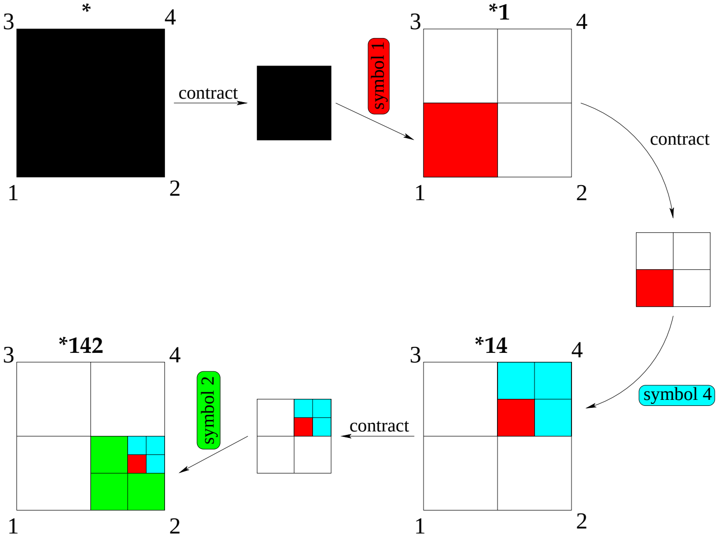
Fixed simple input-driven dynamics

Four input symbols coded as corners of the unit square, $\mathbf{u}(t) \in \{0, 1\}^2$.

States evolve in $[0, 1]^2$ according to

$$\begin{aligned} \mathbf{x}(t) &= f(\mathbf{u}(t), \mathbf{x}(t-1)) \\ &= \frac{1}{2} \cdot \mathbf{x}(t-1) + \frac{1}{2} \cdot \mathbf{u}(t). \end{aligned}$$

The power of contractive dynamics



Mapping sequences with contractions

- Mapping symbolic sequences into Euclidean space
- Coding strategy is Markovian
- Sequences with shared histories of recently observed symbols have close images
- The longer is the common suffix of two sequences, the closer are they mapped to each other
- Naturally implements VLMM

Reservoir computation

- Fixed "contractive" input driven dynamics - no need to train!
- Only linear static readout is trained - very efficient
- different flavours:
  - Echo State Network
  - Liquid State Machine
  - Fractal Prediction Machine