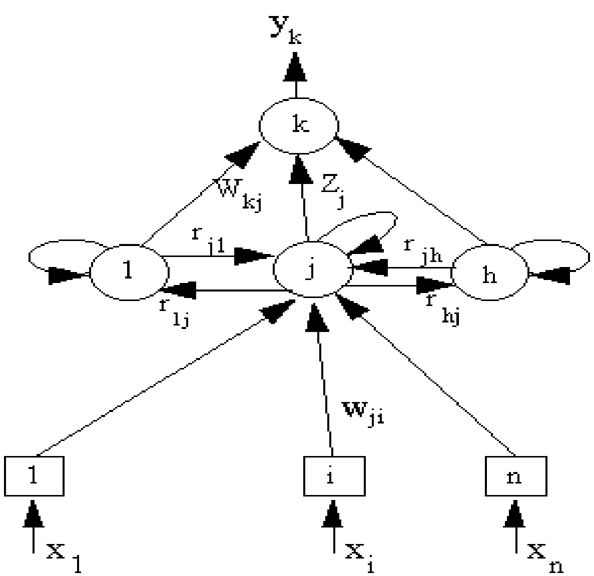
The Difficulty of Learning Long-Term Dependencies with Gradient Descent

Papers by Y. Bengio, P. Simard, P. Frasconi, S. Hochreiter and J. Schmidhuber

Presented by Keerthiram Murugesan

Recurrent Neural Network

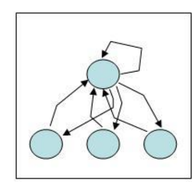
- Neural Network with Feedback
  - Stores information/activations
  - Represents context information

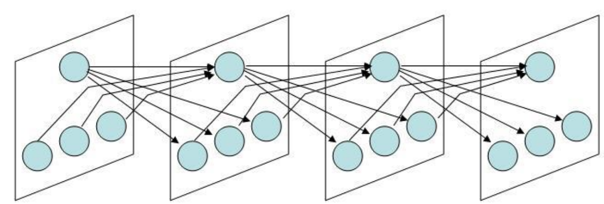


Learning Algorithms

- Compute gradient of a cost function w.r.t. weights
- Back-Propagation Through Time
- Forward Propagation
- For Constrained

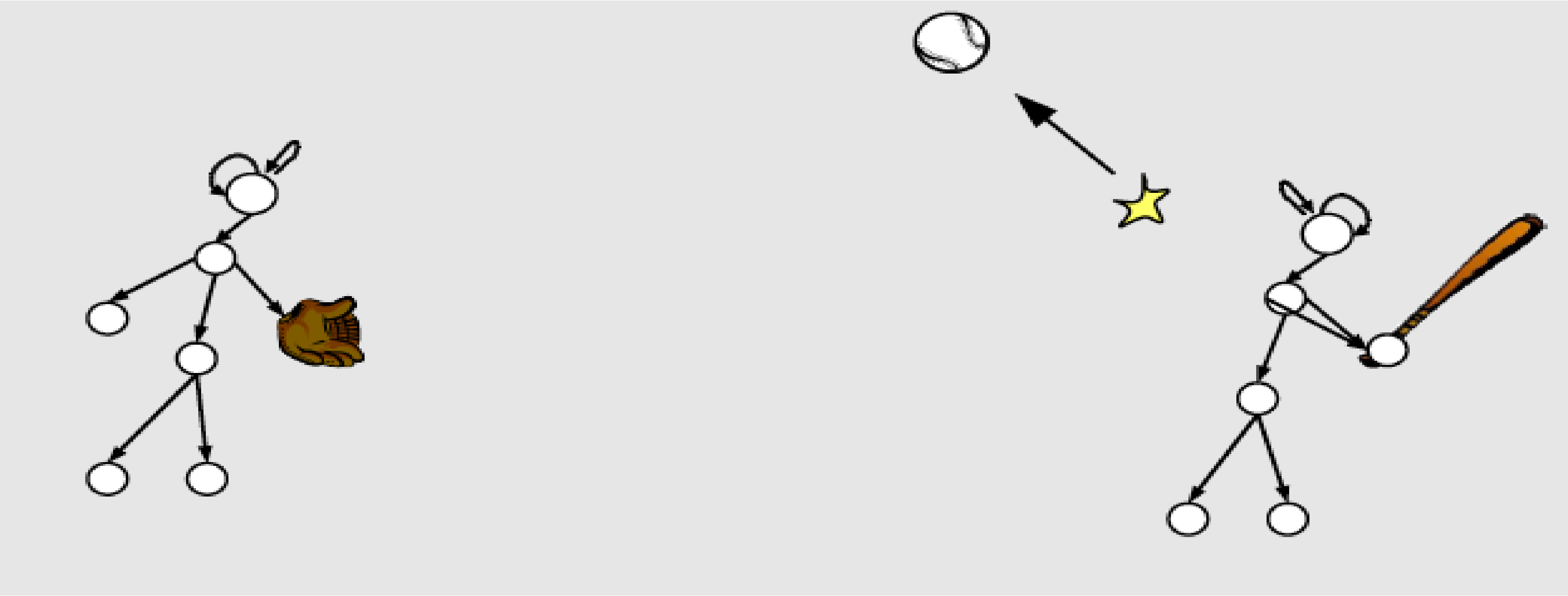
Recurrent Networks





Basic Requirements for a recurrent system

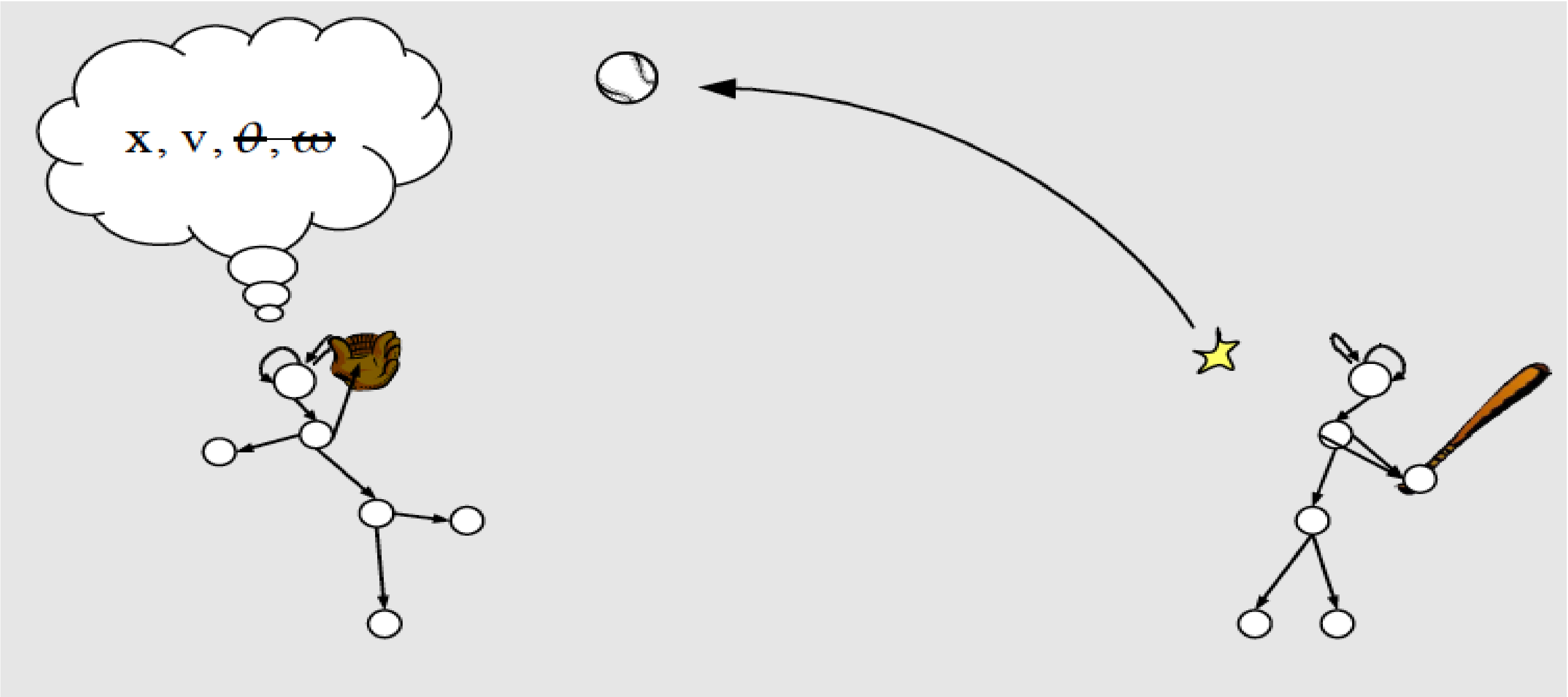
- For a parametric dynamical system that can store relevant state information, it should satisfy 3 basic requirements.



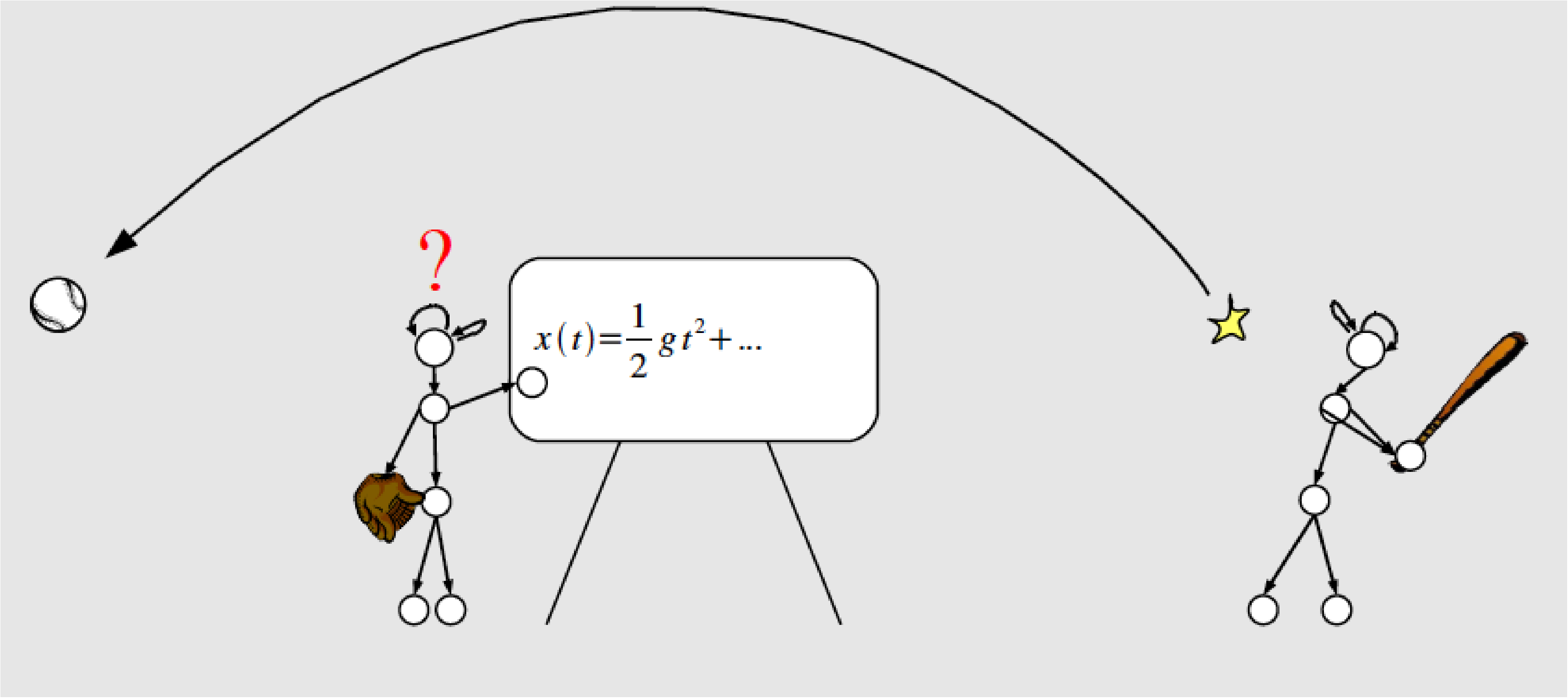
1. System should be able to store information for an arbitrary duration

– e.g. be able to learn constants

2. System should be resistant to noise.



3. System parameters should be learnable in reasonable time



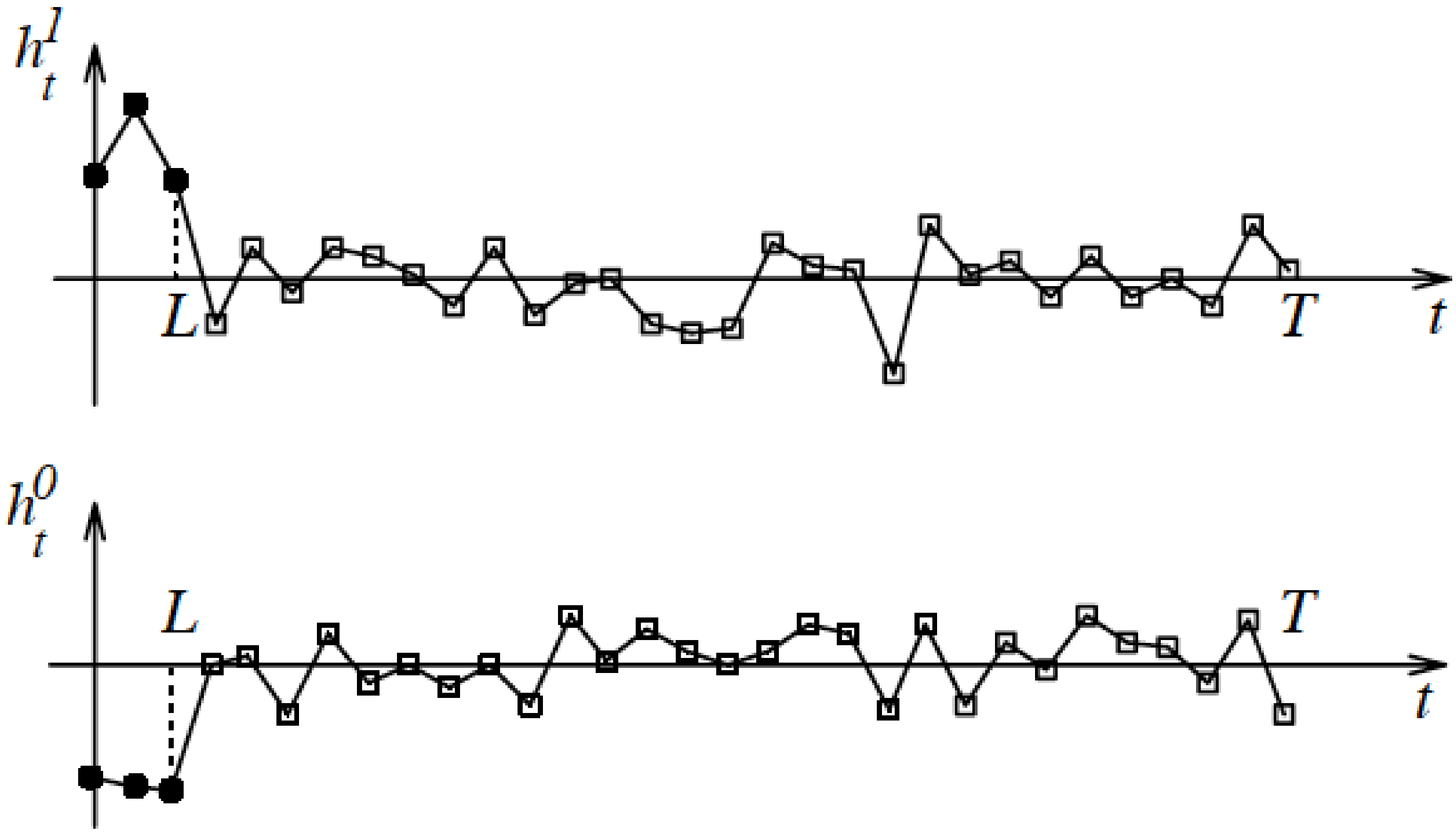
Problem

"Storage" in recurrent neural nets via gradient-based backprop is problematic due to the tradeoff between storing long-term dependencies and efficient learning.

Minimal Task (classification)

- We define a minimal task as a test that must necessarily be passed in order to satisfy the three conditions.
- Goal: Classify 2 different types of sequences of length T;
- $\mathcal{C}(u_1, \ldots, u_T) \in \{0, 1\}$
- $\mathcal{C}(u_1, \ldots, u_T) = \mathcal{C}(u_1, \ldots, u_L).$

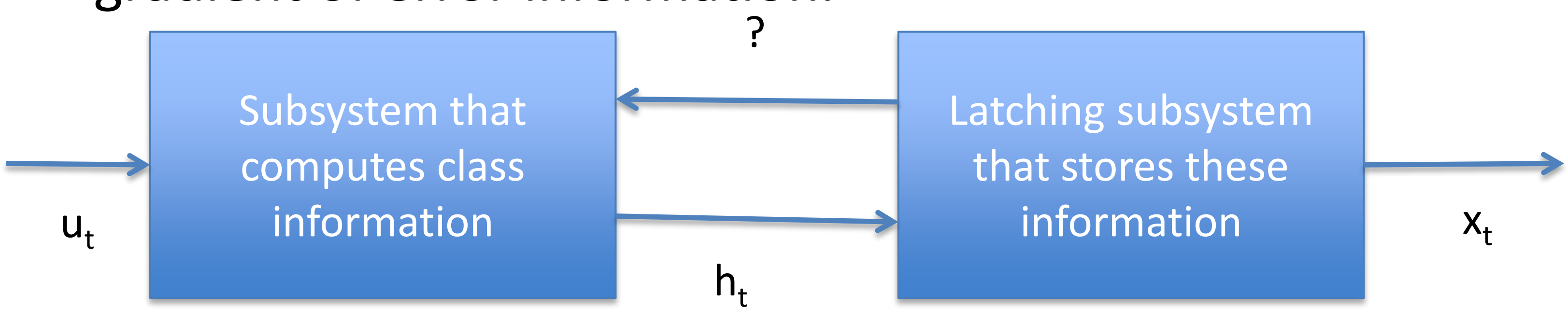
$$\mathcal{C}(u_1, \ldots, u_T) = \mathcal{C}(u_1, \ldots, u_L).$$



Learnability

- Two Subtask:
  - Process input sequence (*u*) in order to extract some information about the class (*h*).
  - Store these information for an arbitrary duration (latching)

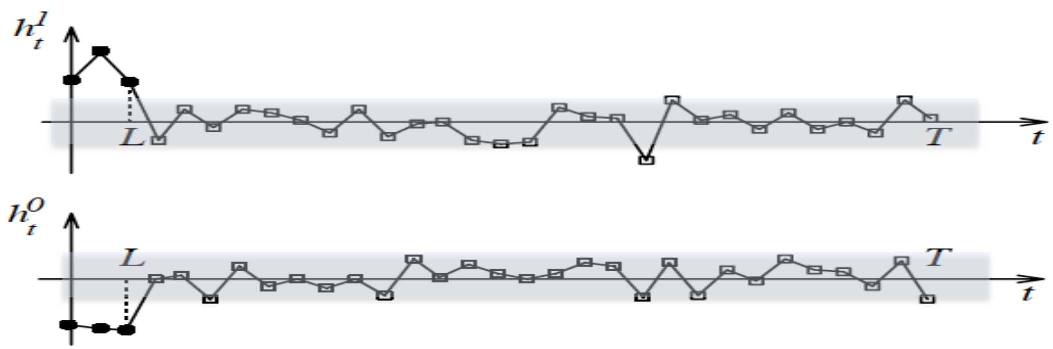
The ability to learn h_t is the measure of effectiveness of the gradient of error information.



h_t is the extracted information about the class.

Non-linear Solution

- Use threshold h* (h* depends on *w*)
  - Keep Inputs ($|h_t|$) larger than h* for a long time
  - Small noisy inputs ($|h_t|$ *smaller than h**) cannot change the sign of activation/state vector (a_t) of the neuron.

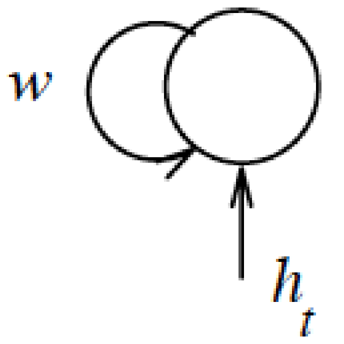


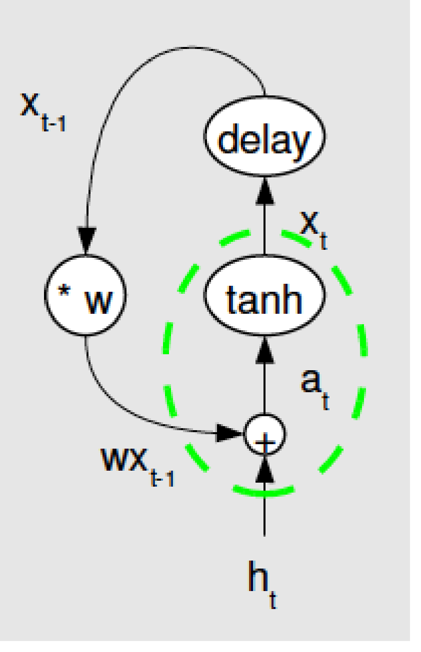
Simple Recurrent Network

- With Single recurrent neuron

$$x_t = f(a_t) = tanh(a_t)$$

$$a_t = wx_{t-1} + h_t$$





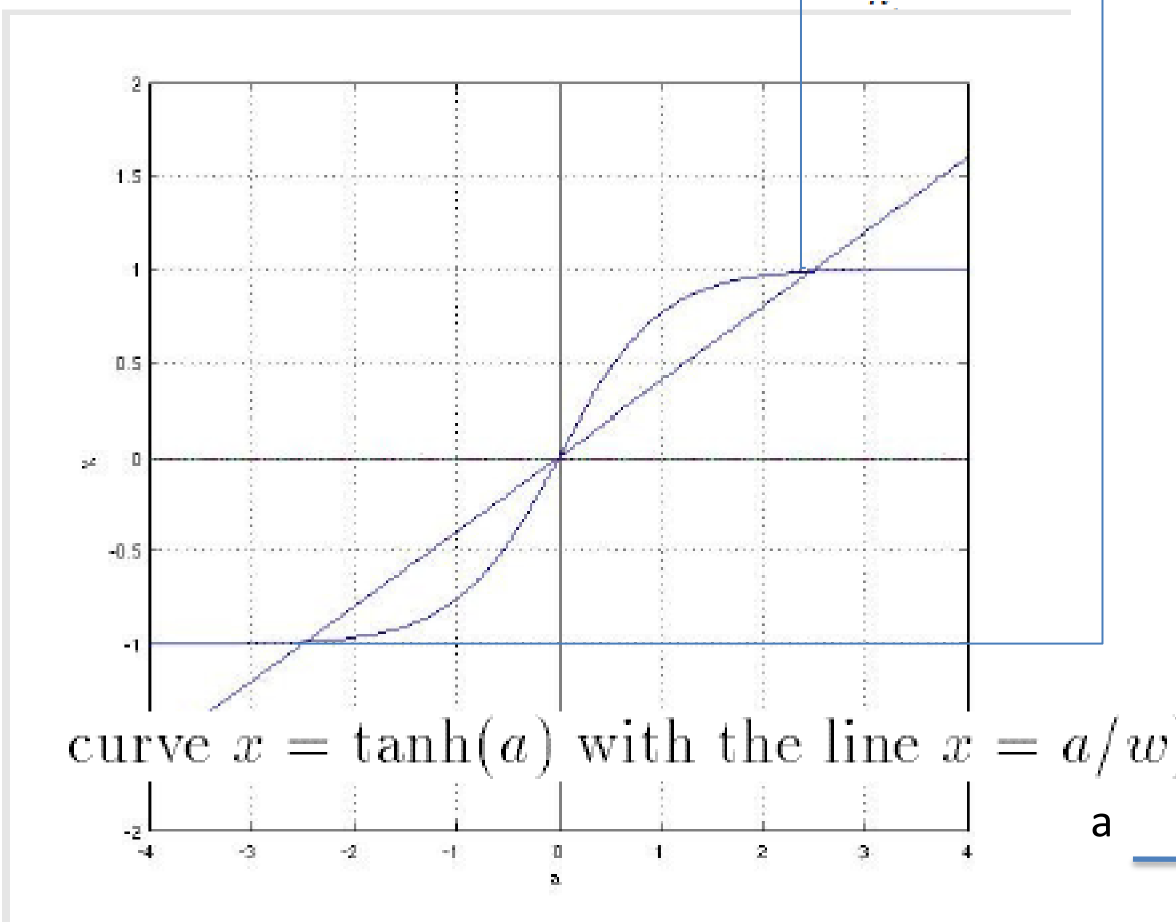
Representation

- $$x_t = f(a_t) = tanh(a_t)$$
  $$a_t = wx_{t-1} + h_t$$

Attractors

Attractor is a point where $a_t=a_{t-1}$

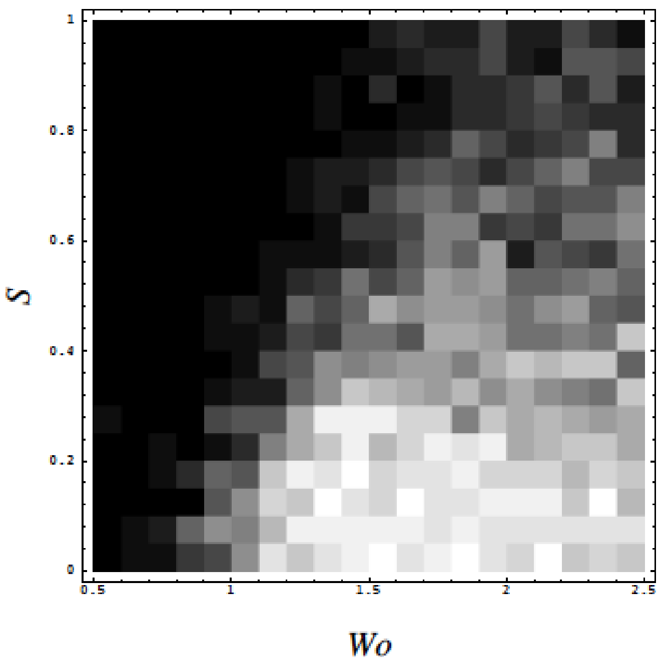
x



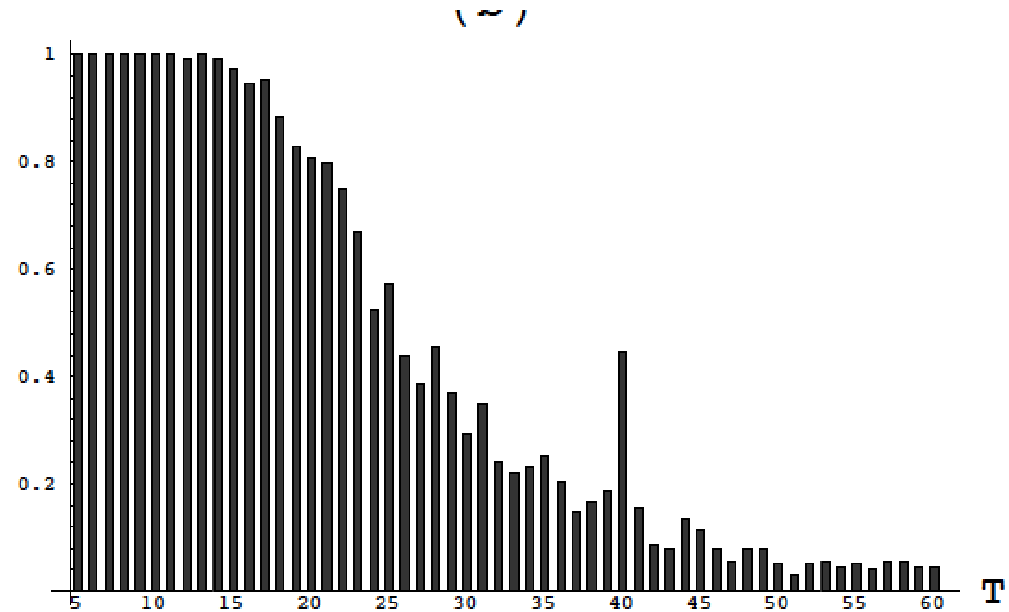
a

Experimental Results

Density Plot of convergence
Over Variance S of Gaussian noise
and initial weight W_0. Fix L=3,T=20



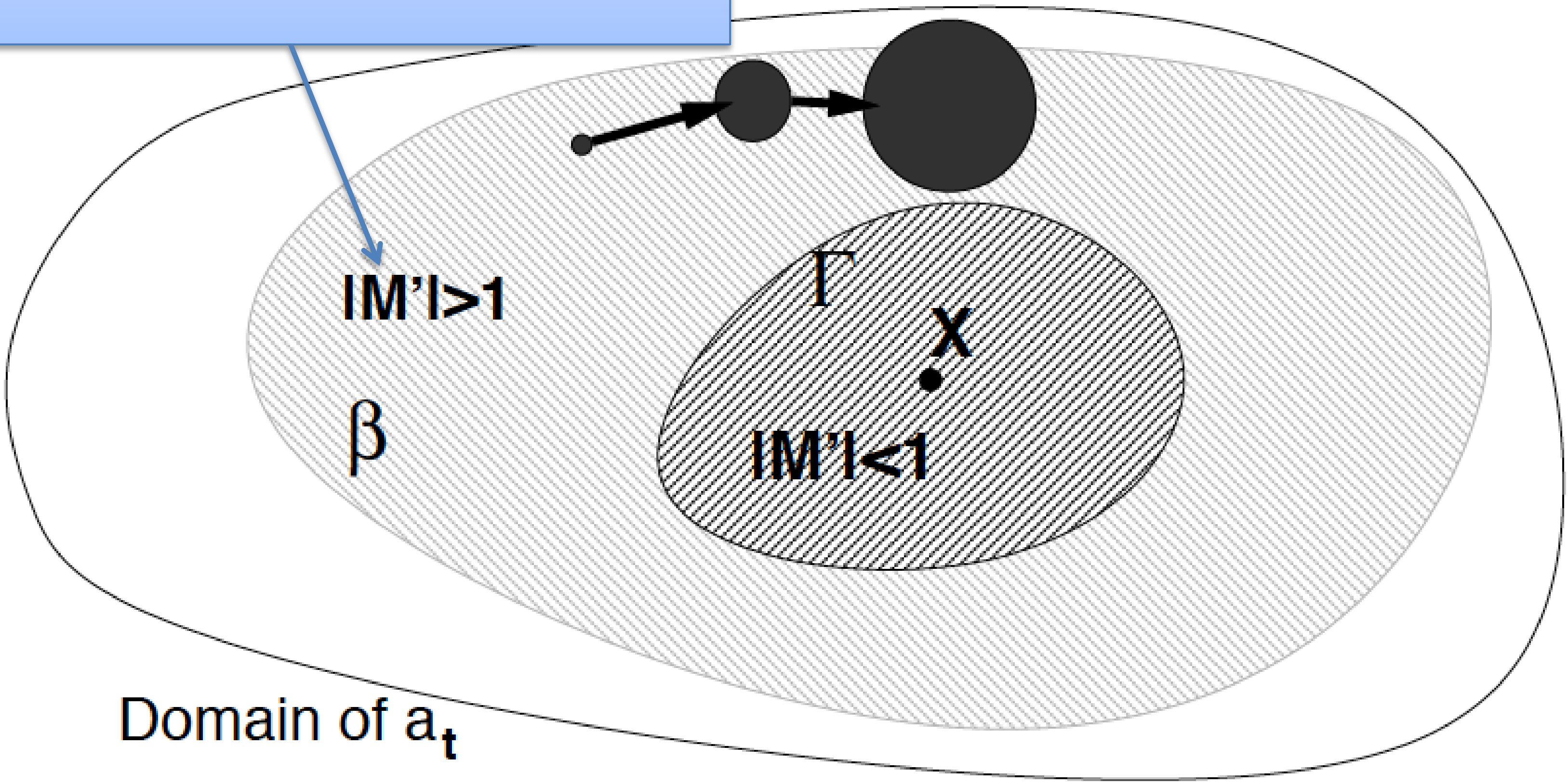
Frequency of convergence on training sets
w.r.t. sequence length T.
Fix S=0.2,W_0=1.25



(White = High Density)
Bigger noise = harder to learn (S is large)
Smaller weight = harder to learn (w_0 is small)

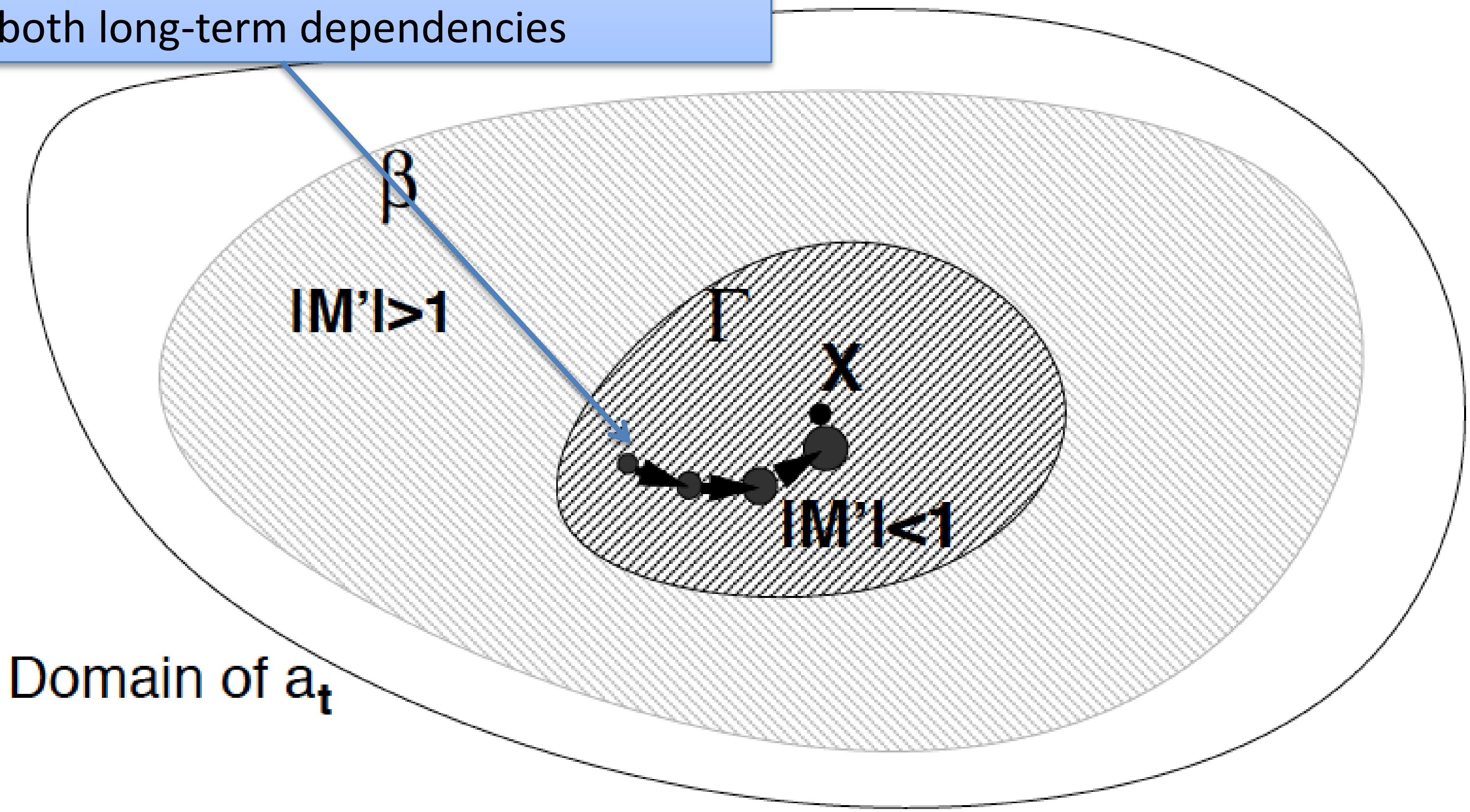
- The above results shows:
  - It doesn’t work even for simple situation (latch one bit of information)
  - Gradient Descent on the output error fails for long-term dependencies, for most of the initial parameter values (S, W_0, T).
  - But why?

No Robust Latching Possible, gradient grows exponentially (it goes to infinity)
So it can't learn both long-term and short-term dependencies



Robust Latching Occurs, gradient decays (short-term dependency has some value where as long-term dependency vanishes-see next slide)

So it can learn short-term dependencies but not both long-term dependencies



Effect on Weight Gradient

- $$\frac{\partial C_t}{\partial W} = \sum_{\tau \leq t} \frac{\partial C_t}{\partial a_\tau} \frac{\partial a_\tau}{\partial W} = \sum_{\tau \leq t} \frac{\partial C_t}{\partial a_t} \frac{\partial a_t}{\partial a_\tau} \frac{\partial a_\tau}{\partial W}$$
- $\frac{\partial a_t}{\partial a_\tau}$ converges exponentially to 0 as t-τ increases.
- $$\frac{\partial C_t}{\partial a_\tau} \frac{\partial a_\tau}{\partial W} \longrightarrow 0$$

Observations

- The sufficient condition to obtain gradient decay is also a necessary condition for the system to robustly store discrete state information for the long term.

- It is difficult to learn long term dependencies because total gradient is the sum of long-term & short-term influences and short-term influences then dominates the total gradient.

Alternative Methods

Simulated annealing

- Good: best overall error.
- Bad: requires ~101.8 times more iterations than other methods.

Multigrid stochastic search

- Good: comparable speed to gradient-based methods.
- Bad: comparable susceptibility to local minima.

Alternative Methods

Standard backprop

- Vanishing gradient problem, local minima.

Timeweighted quasi-newton

- Same problems as backprop, just less bad.

Discrete error propagation

- Usually the fastest, error rate comparable to non-annealing methods.


That's all Folks!
Don't forget
the references

References

- Y. Bengio,P. Simard and P. Frasconi, *Learning Long-Term Dependencies with Gradient Descent is Difficult.*
- S. Hochreiter, Y. Bengio, P. Frasconi and J. Schmidhuber, *Gradient Flow in Recurrent Nets: the Difficulty of Learning Long-Term Dependencies*
- Images adopted from Matt Grimes , Ayse Naz Erkan's presentation on the same paper.