Unbiased Online Recurrent Optimization

Corentin TALLEC

LRI & Paris-Saclay University, France

Supervised by

Yann OLLIVIER

GT Deepnet

February 16, 2017

UORO

- Goal: to train dynamical systems, recurrent networks, reinforcement learning systems... while they are running, without the “backward in time” analysis that comes with backpropagation through time,

UORO

- Goal: to train dynamical systems, recurrent networks, reinforcement learning systems... while they are running, without the "backward in time" analysis that comes with backpropagation through time,

- ... at the same cost as just running the system,

UORO

- Goal: to train dynamical systems, recurrent networks, reinforcement learning systems... while they are running, without the "backward in time" analysis that comes with backpropagation through time,
- ... at the same cost as just running the system,
  ... but with (reasonable) additional noise.

UORO

- Goal: to train dynamical systems, recurrent networks, reinforcement learning systems... while they are running, without the "backward in time" analysis that comes with backpropagation through time,
- ... at the same cost as just running the system,
  ... but with (reasonable) additional noise.
- Not relevant if you have a large number of short training sequences
- Relevant if you have a small number of long training sequences, or only one training sequence (e.g., life).

Algorithms to train dynamical systems

$p = \#\text{params}$

Algorithm	Cost per time step	Store past states	Store past data

Algorithms to train dynamical systems

$p = \#\text{params}$

Algorithm	Cost per time step	Store past states	Store past data
Backprop through time truncated to length T	$O(p)$ biased	$O(\sqrt{T})$	$O(T)$

Algorithms to train dynamical systems

$p = \#\text{params}$

Algorithm	Cost per time step	Store past states	Store past data
Backprop through time truncated to length T	$O(p)$ biased	$O(\sqrt{T})$	$O(T)$
RTRL	$O(p^2)$	No	No
?	$O(p)$	No	No

Recurrent nets as dynamical systems

Problem: how to train a dynamical system defined by

$$s_{t+1} = F(x_{t+1}, s_t, \theta)$$

- s_t: internal state of the system at time t
- x_{t+1}: external input signal
- F: transition function, fixed
- θ: parameter to be trained

Recurrent nets as dynamical systems

Problem: how to train a dynamical system defined by

$$s_{t+1} = F(x_{t+1}, s_t, \theta)$$

- s_t: internal state of the system at time t
- x_{t+1}: external input signal
- F: transition function, fixed
- θ: parameter to be trained

Goal: minimize some loss function $\ell_t = \ell(s_t)$ along the trajectories of the system.

Recurrent nets as dynamical systems

Problem: how to train a dynamical system defined by

$$s_{t+1} = F(x_{t+1}, s_t, \theta)$$

- s_t: internal state of the system at time t
- x_{t+1}: external input signal
- F: transition function, fixed
- θ: parameter to be trained

Goal: minimize some loss function $\ell_t = \ell(s_t)$ along the trajectories of the system.

Examples: RNNs, LSTMs, GRUs, ...

Simple strategy: online gradient descent over the loss at time t,

$$\theta \leftarrow \theta - \eta \frac{\partial \ell_t}{\partial \theta}$$

with learning rate η.

Simple strategy: online gradient descent over the loss at time t,

$$\theta \leftarrow \theta - \eta \, \frac{\partial \ell_t}{\partial \theta}$$

with learning rate η.

Problem: how to compute the derivative $\frac{\partial \ell_t}{\partial \theta}$? Current loss depends on θ via whole past trajectory.

Simple strategy: online gradient descent over the loss at time t,

$$\theta \leftarrow \theta - \eta \frac{\partial \ell_t}{\partial \theta}$$

with learning rate η.

Problem: how to compute the derivative $\frac{\partial \ell_t}{\partial \theta}$? Current loss depends on θ via whole past trajectory.

Standard approach to compute $\frac{\partial \ell_t}{\partial \theta}$: backpropagation through time (BPTT).
Problems: goes back in time, keep track of past history...

Why do we backpropagate through time?

For instance, let us compute how the current loss ℓ_t depends on the starting point s_0:

$$\frac{\partial \ell_t}{\partial s_0} =$$

Why do we backpropagate through time?

For instance, let us compute how the current loss ℓ_t depends on the starting point s_0:

$$\frac{\partial \ell_t}{\partial s_0} = \frac{\partial \ell_t}{\partial s_t} \times \frac{\partial s_t}{\partial s_{t-1}} \times \cdots \times \frac{\partial s_1}{\partial s_0}$$

Why do we backpropagate through time?

For instance, let us compute how the current loss ℓ_t depends on the starting point s_0:

$$\frac{\partial \ell_t}{\partial s_0} = \frac{\partial \ell_t}{\partial s_t} \times \frac{\partial s_t}{\partial s_{t-1}} \times \cdots \times \frac{\partial s_1}{\partial s_0}$$

$$= \text{vector} \times \text{matrix} \times \cdots \times \text{matrix}$$

Why do we backpropagate through time?

For instance, let us compute how the current loss ℓ_t depends on the starting point s_0:

$$\frac{\partial \ell_t}{\partial s_0} = \frac{\partial \ell_t}{\partial s_t} \times \frac{\partial s_t}{\partial s_{t-1}} \times \cdots \times \frac{\partial s_1}{\partial s_0}$$

$$= \text{vector} \times \text{matrix} \times \cdots \times \text{matrix}$$

Left-to-right: OK, only vector×matrix multiplications. That's BPTT.

Right-to-left: matrix-matrix multiplications, must store a matrix.

Why do we backpropagate through time?

For instance, let us compute how the current loss ℓ_t depends on the starting point s_0:

$$\frac{\partial \ell_t}{\partial s_0} = \frac{\partial \ell_t}{\partial s_t} \times \frac{\partial s_t}{\partial s_{t-1}} \times \cdots \times \frac{\partial s_1}{\partial s_0}$$

$$= \text{vector} \times \text{matrix} \times \cdots \times \text{matrix}$$

Left-to-right: OK, only vector$\times$matrix multiplications. That's BPTT.

Right-to-left: matrix-matrix multiplications, must store a matrix.
$\Longrightarrow$ only for small networks. Known as real-time recurrent learning (RTRL).

Why do we backpropagate through time?

For instance, let us compute how the current loss ℓ_t depends on the starting point s_0:

$$\frac{\partial \ell_t}{\partial s_0} = \frac{\partial \ell_t}{\partial s_t} \times \frac{\partial s_t}{\partial s_{t-1}} \times \cdots \times \frac{\partial s_1}{\partial s_0}$$

$$= \text{vector} \times \text{matrix} \times \cdots \times \text{matrix}$$

Left-to-right: OK, only vector×matrix multiplications. That's BPTT.

Right-to-left: matrix-matrix multiplications, must store a matrix. $\Longrightarrow$ only for small networks. Known as real-time recurrent learning (RTRL).

Same forward-backward structure in many problems: hidden Markov models (EM), reinforcement learning and optimal control (Bellman equations)...

If you cannot travel back in time...

Algorithms that go forward in time must maintain the gradient of the current state with respect to the parameters:

$$G_t := \frac{\partial s_t}{\partial \theta}$$

and then compute the gradient of the loss via the chain rule

$$\frac{\partial \ell_t}{\partial \theta} = \frac{\partial \ell_t}{\partial s_t} \frac{\partial s_t}{\partial \theta}$$

If you cannot travel back in time...

Algorithms that go forward in time must maintain the gradient of the current state with respect to the parameters:

$$G_t := \frac{\partial s_t}{\partial \theta}$$

and then compute the gradient of the loss via the chain rule

$$\frac{\partial \ell_t}{\partial \theta} = \frac{\partial \ell_t}{\partial s_t} \frac{\partial s_t}{\partial \theta}$$

G_t is a full object of size dim(state)$\times$dim(param).

If you cannot travel back in time...

Algorithms that go forward in time must maintain the gradient of the current state with respect to the parameters:

$$G_t := \frac{\partial s_t}{\partial \theta}$$

and then compute the gradient of the loss via the chain rule

$$\frac{\partial \ell_t}{\partial \theta} = \frac{\partial \ell_t}{\partial s_t} \frac{\partial s_t}{\partial \theta}$$

G_t is a full object of size dim(state)$\times$dim(param).

Sometimes, cannot even store G_t.

UORO

UORO

- At each time, maintain a search direction $\tilde{\theta}_t$ in parameter space,

UORO

- At each time, maintain a search direction $\tilde{\theta}_t$ in parameter space, together with an estimate $\tilde{s}_t$ of the effect of $\tilde{\theta}$ on the current state s_t.

UORO

- At each time, maintain a search direction $\tilde{\theta}_t$ in parameter space, together with an estimate $\tilde{s}_t$ of the effect of $\tilde{\theta}$ on the current state s_t.
- Have the search direction $\tilde{\theta}_t$ evolve stochastically, but not fully at random, in a way driven by how the transition function F depends on the parameter θ.

UORO

- At each time, maintain a search direction $\tilde{\theta}_t$ in parameter space, together with an estimate $\tilde{s}_t$ of the effect of $\tilde{\theta}$ on the current state s_t.
- Have the search direction $\tilde{\theta}_t$ evolve stochastically, but not fully at random, in a way driven by how the transition function F depends on the parameter θ.
- Can be arranged so that, at each time, $\tilde{s}_t \tilde{\theta}_t^\top$ is an unbiased estimate of $\dfrac{\partial s_t}{\partial \theta}$:

$$\mathbb{E} \tilde{s}_t \tilde{\theta}_t^\top = G_t$$

UORO

- At each time, maintain a search direction $\tilde{\theta}_t$ in parameter space, together with an estimate $\tilde{s}_t$ of the effect of $\tilde{\theta}$ on the current state s_t.
- Have the search direction $\tilde{\theta}_t$ evolve stochastically, but not fully at random, in a way driven by how the transition function F depends on the parameter θ.
- Can be arranged so that, at each time, $\tilde{s}_t\tilde{\theta}_t^\top$ is an unbiased estimate of $\dfrac{\partial s_t}{\partial \theta}$:

$$\mathbb{E}\tilde{s}_t\tilde{\theta}_t^\top = G_t$$

- Unbiased estimate of $G_t \implies$ unbiased estimate of the gradient of the loss function ℓ_t wrt the parameter

UORO

- At each time, maintain a search direction $\tilde{\theta}_t$ in parameter space, together with an estimate $\tilde{s}_t$ of the effect of $\tilde{\theta}$ on the current state s_t.

- Have the search direction $\tilde{\theta}_t$ evolve stochastically, but not fully at random, in a way driven by how the transition function F depends on the parameter θ.

- Can be arranged so that, at each time, $\tilde{s}_t \tilde{\theta}_t^\top$ is an unbiased estimate of $\dfrac{\partial s_t}{\partial \theta}$:
$$\mathbb{E} \tilde{s}_t \tilde{\theta}_t^\top = G_t$$

- Unbiased estimate of $G_t \implies$ unbiased estimate of the gradient of the loss function ℓ_t wrt the parameter

- The estimates are noisy but unbiased $\implies$ over time the parameter evolves in the correct direction.

To understand how to approximate G_t, let us look at its evolution. The evolution equation is

$$s_{t+1} = F(x_{t+1}, s_t, \theta)$$

To understand how to approximate G_t, let us look at its evolution. The evolution equation is

$$s_{t+1} = F(x_{t+1}, s_t, \theta) \qquad\qquad G_t := \frac{\partial s_t}{\partial \theta}$$

To understand how to approximate G_t, let us look at its evolution. The evolution equation is

$$s_{t+1} = F(x_{t+1}, s_t, \theta) \qquad\qquad G_t := \frac{\partial s_t}{\partial \theta}$$

Taking the derivative of the evolution equation wrt θ we get

$$G_{t+1} = \frac{\partial F}{\partial \theta}(x_{t+1}, s_t, \theta) +$$

To understand how to approximate G_t, let us look at its evolution. The evolution equation is

$$s_{t+1} = F(x_{t+1}, s_t, \theta) \qquad\qquad G_t := \frac{\partial s_t}{\partial \theta}$$

Taking the derivative of the evolution equation wrt θ we get

$$G_{t+1} = \frac{\partial F}{\partial \theta}(x_{t+1}, s_t, \theta) + \frac{\partial F}{\partial s}(x_{t+1}, s_t, \theta) \cdot G_t$$

To understand how to approximate G_t, let us look at its evolution. The evolution equation is

$$s_{t+1} = F(x_{t+1}, s_t, \theta) \qquad\qquad G_t := \frac{\partial s_t}{\partial \theta}$$

Taking the derivative of the evolution equation wrt θ we get

$$G_{t+1} = \frac{\partial F}{\partial \theta}(x_{t+1}, s_t, \theta) + \frac{\partial F}{\partial s}(x_{t+1}, s_t, \theta) \cdot G_t$$

This equation is affine.

To understand how to approximate G_t, let us look at its evolution. The evolution equation is

$$s_{t+1} = F(x_{t+1}, s_t, \theta) \qquad\qquad G_t := \frac{\partial s_t}{\partial \theta}$$

Taking the derivative of the evolution equation wrt θ we get

$$G_{t+1} = \frac{\partial F}{\partial \theta}(x_{t+1}, s_t, \theta) + \frac{\partial F}{\partial s}(x_{t+1}, s_t, \theta) \cdot G_t$$

This equation is affine.

$\Longrightarrow$ If $\tilde{G}_t$ is an unbiased approximation of G_t, then

To understand how to approximate G_t, let us look at its evolution. The evolution equation is

$$s_{t+1} = F(x_{t+1}, s_t, \theta) \qquad\qquad G_t := \frac{\partial s_t}{\partial \theta}$$

Taking the derivative of the evolution equation wrt θ we get

$$G_{t+1} = \frac{\partial F}{\partial \theta}(x_{t+1}, s_t, \theta) + \frac{\partial F}{\partial s}(x_{t+1}, s_t, \theta) \cdot G_t$$

This equation is affine.
$\Longrightarrow$ If $\tilde{G}_t$ is an unbiased approximation of G_t, then

$$\frac{\partial F}{\partial \theta} + \frac{\partial F}{\partial s} \cdot \tilde{G}_t$$

is an unbiased approximation of G_{t+1}.

To understand how to approximate G_t, let us look at its evolution. The evolution equation is

$$s_{t+1} = F(x_{t+1}, s_t, \theta) \qquad\qquad G_t := \frac{\partial s_t}{\partial \theta}$$

Taking the derivative of the evolution equation wrt θ we get

$$G_{t+1} = \frac{\partial F}{\partial \theta}(x_{t+1}, s_t, \theta) + \frac{\partial F}{\partial s}(x_{t+1}, s_t, \theta) \cdot G_t$$

This equation is affine.
$\Longrightarrow$ If $\tilde{G}_t$ is an unbiased approximation of G_t, then

$$\frac{\partial F}{\partial \theta} + \frac{\partial F}{\partial s} \cdot \tilde{G}_t$$

is an unbiased approximation of G_{t+1}. Use with $\tilde{G}_t = \tilde{s}_t \tilde{\theta}_t^\top$.

To understand how to approximate G_t, let us look at its evolution. The evolution equation is

$$s_{t+1} = F(x_{t+1}, s_t, \theta) \qquad\qquad G_t := \frac{\partial s_t}{\partial \theta}$$

Taking the derivative of the evolution equation wrt θ we get

$$G_{t+1} = \frac{\partial F}{\partial \theta}(x_{t+1}, s_t, \theta) + \frac{\partial F}{\partial s}(x_{t+1}, s_t, \theta) \cdot G_t$$

This equation is affine.
$\Longrightarrow$ If $\tilde{G}_t$ is an unbiased approximation of G_t, then

$$\frac{\partial F}{\partial \theta} + \frac{\partial F}{\partial s} \cdot \tilde{G}_t$$

is an unbiased approximation of G_{t+1}. Use with $\tilde{G}_t = \tilde{s}_t \tilde{\theta}_t^\top$.

Problem: Even if $\tilde{G}_t = \tilde{s}_t \tilde{\theta}_t^\top$ is rank-one, $\tilde{G}_{t+1}$ is full-rank again.

The rank-one trick

Proposition

Let A be a $k \times l$ matrix,

The rank-one trick

Proposition

Let A be a $k \times l$ matrix,
Let $\nu \in \mathbb{R}^l$ be a column vector of random ± 1 signs. Let

The rank-one trick

Proposition

Let A be a $k \times l$ matrix,
Let $\nu \in \mathbb{R}^l$ be a column vector of random ± 1 signs. Let
Then $(A\nu)\,\nu^\top$ is an unbiased, rank-one estimate of A:

$$\mathbb{E}\,(A\nu)\,\nu^\top = A$$

Proof: $\mathbb{E}\,\nu\,\nu^\top = I$.

The rank-one trick

Proposition

Let A be a $k \times l$ matrix,
Let $\nu \in \mathbb{R}^l$ be a column vector of random ± 1 signs. Let
Then $(A\,\nu)\,\nu^\top$ is an unbiased, rank-one estimate of A:

$$\mathbb{E}\,(A\,\nu)\,\nu^\top = A$$

Proof: $\mathbb{E}\,\nu\,\nu^\top = I$.

Corollary: UORO is unbiased, $\mathbb{E}\tilde{s}_t\tilde{\theta}_t^\top = G_t$.

UORO in equations

Consider a recurrent model $s_{t+1} = F(x_{t+1}, s_t, \theta)$.

UORO in equations

Consider a recurrent model $s_{t+1} = F(x_{t+1}, s_t, \theta)$.

Maintain at each step a vector $\tilde{s}_t \in \mathbb{R}^{\dim s}$ and $\tilde{\theta}_t \in \mathbb{R}^{\dim \theta}$ with:

Then $\theta \leftarrow \theta - \alpha \frac{\partial \ell_t}{\partial s_t} \cdot \tilde{s}_t \cdot \tilde{\theta}_t^\top - \frac{\partial \ell_t}{\partial \theta}\Big|_{s_t}$ is an unbiased gradient descent on θ.

(Unbiased: $\mathbb{E} = \theta - \alpha\, \partial \mathrm{loss}/\partial\theta$.)

UORO in equations

Consider a recurrent model $s_{t+1} = F(x_{t+1}, s_t, \theta)$.

Maintain at each step a vector $\tilde{s}_t \in \mathbb{R}^{\dim s}$ and $\tilde{\theta}_t \in \mathbb{R}^{\dim \theta}$ with:

$$\tilde{\theta}_{t+1} = \tfrac{1}{\rho_0}\tilde{\theta}_t + \tfrac{1}{\rho_1}\left(\frac{\partial F}{\partial \theta}\right)^{\top}\nu$$

Then $\theta \leftarrow \theta - \alpha \frac{\partial \ell_t}{\partial s_t} \cdot \tilde{s}_t \cdot \tilde{\theta}_t^{\top} - \frac{\partial \ell_t}{\partial \theta}\Big|_{s_t}$ is an unbiased gradient descent on θ.

(Unbiased: $\mathbb{E} = \theta - \alpha\, \partial \text{loss}/\partial \theta$.)

UORO in equations

Consider a recurrent model $s_{t+1} = F(x_{t+1}, s_t, \theta)$.

Maintain at each step a vector $\tilde{s}_t \in \mathbb{R}^{\dim s}$ and $\tilde{\theta}_t \in \mathbb{R}^{\dim \theta}$ with:

$$\tilde{\theta}_{t+1} = \tfrac{1}{\rho_0}\tilde{\theta}_t + \tfrac{1}{\rho_1}\left(\frac{\partial F}{\partial \theta}\right)^{\top}\nu$$

$$\tilde{s}_{t+1} = \rho_0 \frac{\partial F}{\partial s}\,\tilde{s}_t + \rho_1\,\nu$$

where $\nu \in \mathbb{R}^{\dim s}$ is a vector of random ± 1 signs,

Then $\theta \leftarrow \theta - \alpha \frac{\partial \ell_t}{\partial s_t} \cdot \tilde{s}_t \cdot \tilde{\theta}_t^{\top} - \left.\frac{\partial \ell_t}{\partial \theta}\right|_{s_t}$ is an unbiased gradient descent on θ.

(Unbiased: $\mathbb{E} = \theta - \alpha\,\partial \mathrm{loss}/\partial\theta$.)

UORO in equations

Consider a recurrent model $s_{t+1} = F(x_{t+1}, s_t, \theta)$.

Maintain at each step a vector $\tilde{s}_t \in \mathbb{R}^{\dim s}$ and $\tilde{\theta}_t \in \mathbb{R}^{\dim \theta}$ with:

$$\tilde{\theta}_{t+1} = \tfrac{1}{\rho_0}\tilde{\theta}_t + \tfrac{1}{\rho_1}(\frac{\partial F}{\partial \theta})^\top \nu$$

$$\tilde{s}_{t+1} = \rho_0 \frac{\partial F}{\partial s} \tilde{s}_t + \rho_1 \nu$$

where $\nu \in \mathbb{R}^{\dim s}$ is a vector of random ± 1 signs, and $\rho_0 := \sqrt{\left\|\tilde{\theta}_t\right\| / \|\partial F / \partial s \, \tilde{s}_t\|}$ and $\rho_1 := \sqrt{\|(\partial F/\partial \theta)^\top \nu\| / \|\nu\|}$.

Then $\theta \leftarrow \theta - \alpha \frac{\partial \ell_t}{\partial s_t} \cdot \tilde{s}_t \cdot \tilde{\theta}_t^\top - \left.\frac{\partial \ell_t}{\partial \theta}\right|_{s_t}$ is an unbiased gradient descent on θ.

(Unbiased: $\mathbb{E} = \theta - \alpha \, \partial \mathrm{loss}/\partial \theta$.)

UORO in equations

Consider a recurrent model $s_{t+1} = F(x_{t+1}, s_t, \theta)$.

Maintain at each step a vector $\tilde{s}_t \in \mathbb{R}^{\dim s}$ and $\tilde{\theta}_t \in \mathbb{R}^{\dim \theta}$ with:

$$\tilde{\theta}_{t+1} = \tfrac{1}{\rho_0}\tilde{\theta}_t + \tfrac{1}{\rho_1}\left(\frac{\partial F}{\partial \theta}\right)^{\top}\nu$$

$$\tilde{s}_{t+1} = \rho_0 \frac{\partial F}{\partial s}\,\tilde{s}_t + \rho_1\,\nu$$

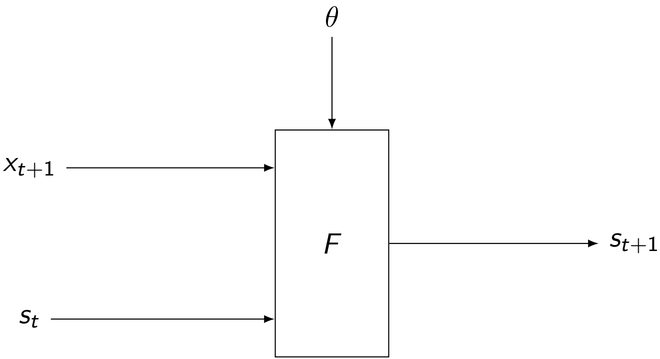
where $\nu \in \mathbb{R}^{\dim s}$ is a vector of random ± 1 signs, and $\rho_0 := \sqrt{\left\|\tilde{\theta}_t\right\| / \|\partial F/\partial s\,\tilde{s}_t\|}$ and $\rho_1 := \sqrt{\|(\partial F/\partial \theta)^{\top}\nu\| / \|\nu\|}$.

Then $\theta \leftarrow \theta - \alpha \dfrac{\partial \ell_t}{\partial s_t} \cdot \tilde{s}_t \cdot \tilde{\theta}_t^{\top} - \dfrac{\partial \ell_t}{\partial \theta}\Big|_{s_t}$ is an unbiased gradient descent on θ.

(Unbiased: $\mathbb{E} = \theta - \alpha\,\partial\mathrm{loss}/\partial\theta$.)

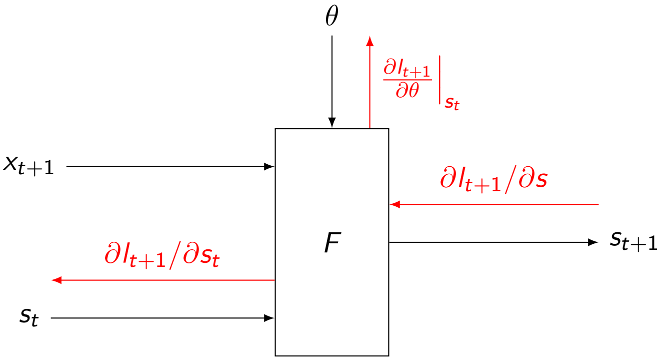
For RNNs, LSTMs, GRUs: same computational cost as running the RNN itself.

UORO graphically



UORO graphically

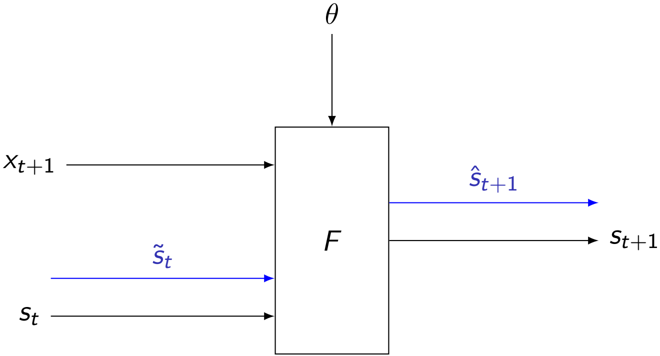
- Backpropagate the loss.



Gradient estimate: $\tilde{g}_{t+1} = \left(\frac{\partial l_{t+1}}{\partial s_t} \cdot \tilde{s}_t\right) \tilde{\theta}_t^\top + \left.\frac{\partial l_{t+1}}{\partial \theta}\right|_{s_t}$

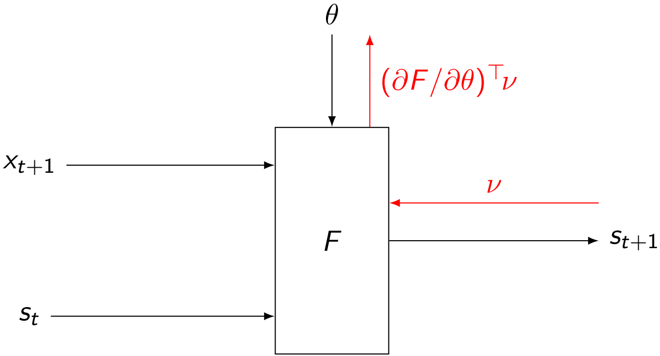
UORO graphically

- Backpropagate the loss.
- Forward differentiate $\tilde{s}_t$ (left multiplication by Jacobian).



UORO graphically

- Backpropagate the loss.
- Forward differentiate $\tilde{s}_t$ (left multiplication by Jacobian).
- Backpropagate random signs.



Aggregation

Aggregation

- $\tilde{g}_{t+1} = \frac{\partial l_{t+1}}{\partial s_t} \cdot \tilde{s}_t + \frac{\partial l_{t+1}}{\partial \theta}\Big|_{s_t}$

Aggregation

- $\tilde{g}_{t+1} = \frac{\partial l_{t+1}}{\partial s_t} \cdot \tilde{s}_t + \frac{\partial l_{t+1}}{\partial \theta}\Big|_{s_t}$
- $\tilde{s}_{t+1} = \rho_0 \, \hat{s}_{t+1} + \rho_1 \, \nu$

Aggregation

- $\tilde{g}_{t+1} = \frac{\partial l_{t+1}}{\partial s_t} \cdot \tilde{s}_t + \frac{\partial l_{t+1}}{\partial \theta}\Big|_{s_t}$
- $\tilde{s}_{t+1} = \rho_0 \, \hat{s}_{t+1} + \rho_1 \, \nu$
- $\tilde{\theta}_{t+1} = \frac{1}{\rho_0} \, \tilde{\theta}_t + \frac{1}{\rho_1} \, \left(\frac{\partial F}{\partial \theta}\right)^{\top} \nu$

UORO: Results

- Datasets:
  - influence balancing
  - distant brackets
  - $a^n b^n$ problem for long dependencies

UORO: Results

- Datasets:
  - influence balancing
  - distant brackets
  - $a^n b^n$ problem for long dependencies
- Model: LSTMs or GRUs, log-loss over probabilistic prediction of next character for character predictions.

UORO: Results

- Datasets:
  - influence balancing
  - distant brackets
  - $a^n b^n$ problem for long dependencies
- Model: LSTMs or GRUs, log-loss over probabilistic prediction of next character for character predictions.
- Baselines: Comparision with Truncated backpropagation through time

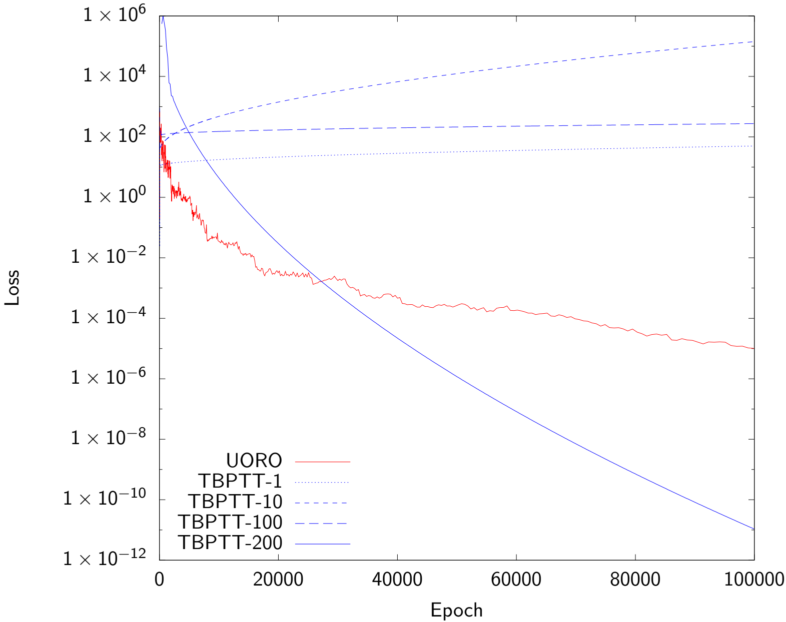
Influence balancing

- Linear model.
- Learn a parameter that has a positive influence in the short term, but a negative influence in the long go.

Influence balancing

- Linear model.
- Learn a parameter that has a positive influence in the short term, but a negative influence in the long go.
- UORO succeeds in balancing dependencies correctly.
- TBPTT fails even when truncation is far above the inherent time range of the model.

Influence balancing, positive influence on 10 steps, negative influence on the 13 next steps.



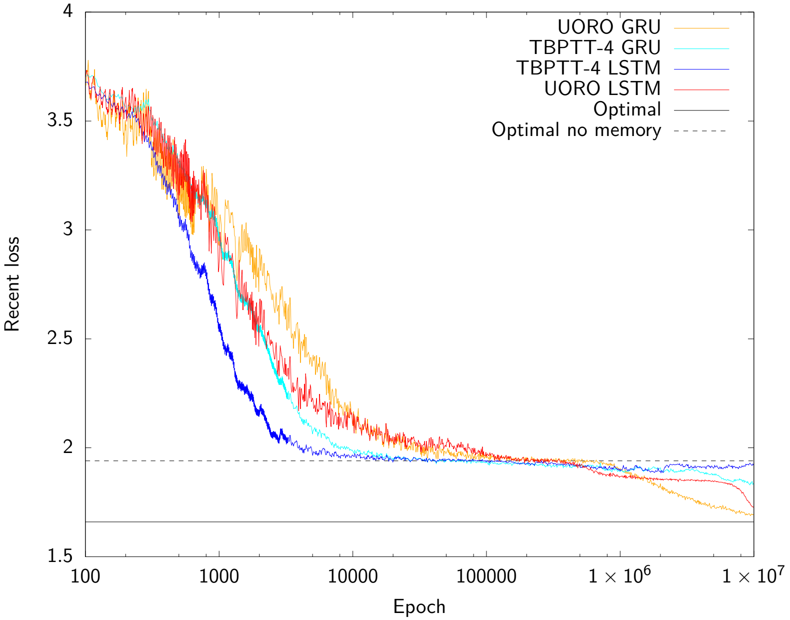
Distant brackets

Sample:

```
[a]lmsle[a]
[c]kopas[c]
[d]llses[d]
[l]oksse[l]
```

Correct prediction requires storing the first character between bracket.

Learning curves on distant brackets



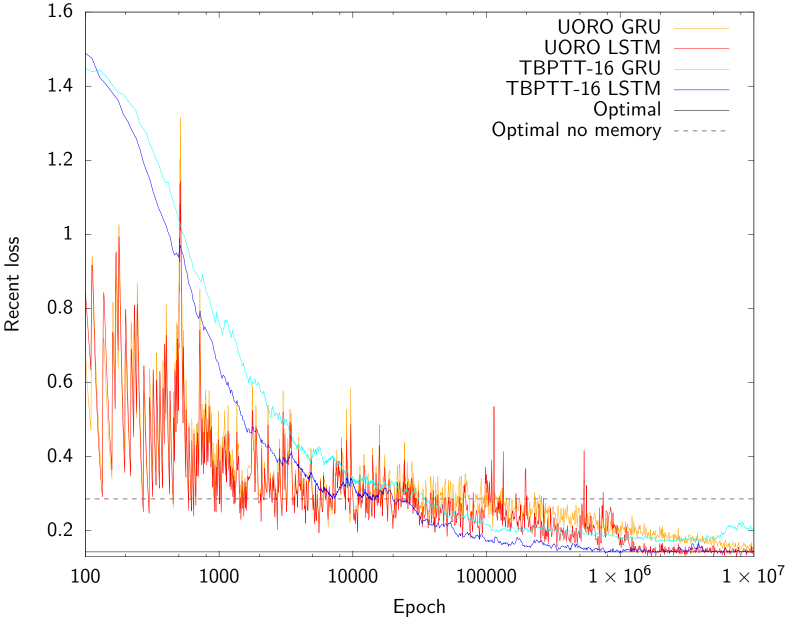
$a^n b^n$

Sample:

```
aaaaa
bbbbb
a
b
aaaaaaaaaaaaaaaa
bbbbbbbbbbbbbbbb
```

Correct prediction requires counting the number of a's.

Learning curves on $a^n b^n$:



Open problems and future work

Open problems and future work

- Convergence theorem: done for simple (linear) cases, in progress for more general cases.

Open problems and future work

- Convergence theorem: done for simple (linear) cases, in progress for more general cases.
- Cleverer variance reduction: noise is put "uniformly" on all components of $\tilde{s}_t$ at each time step. Different units does not respond similarly to noise, how to take this into account?

Open problems and future work

- Convergence theorem: done for simple (linear) cases, in progress for more general cases.
- Cleverer variance reduction: noise is put "uniformly" on all components of $\tilde{s}_t$ at each time step. Different units does not respond similarly to noise, how to take this into account?
- How to deal with parameters that have a non-continuous influence on the trajectory, such as probabilities to make certain discrete choices?

Open problems and future work

- Convergence theorem: done for simple (linear) cases, in progress for more general cases.
- Cleverer variance reduction: noise is put "uniformly" on all components of $\tilde{s}_t$ at each time step. Different units does not respond similarly to noise, how to take this into account?
- How to deal with parameters that have a non-continuous influence on the trajectory, such as probabilities to make certain discrete choices?
- Reinforcement learning:

Open problems and future work

- Convergence theorem: done for simple (linear) cases, in progress for more general cases.
- Cleverer variance reduction: noise is put "uniformly" on all components of $\tilde{s}_t$ at each time step. Different units does not respond similarly to noise, how to take this into account?
- How to deal with parameters that have a non-continuous influence on the trajectory, such as probabilities to make certain discrete choices?
- Reinforcement learning: in real physical systems, response of the environment depends on the actions of the system.

Open problems and future work

- Convergence theorem: done for simple (linear) cases, in progress for more general cases.
- Cleverer variance reduction: noise is put "uniformly" on all components of $\tilde{s}_t$ at each time step. Different units does not respond similarly to noise, how to take this into account?
- How to deal with parameters that have a non-continuous influence on the trajectory, such as probabilities to make certain discrete choices?
- Reinforcement learning: in real physical systems, response of the environment depends on the actions of the system. No gradient available for this $\Longrightarrow$ need to rely on gradients computed from a (learned) model of the world.

Open problems and future work

- Convergence theorem: done for simple (linear) cases, in progress for more general cases.

- Cleverer variance reduction: noise is put "uniformly" on all components of $\tilde{s}_t$ at each time step. Different units does not respond similarly to noise, how to take this into account?

- How to deal with parameters that have a non-continuous influence on the trajectory, such as probabilities to make certain discrete choices?

- Reinforcement learning: in real physical systems, response of the environment depends on the actions of the system. No gradient available for this $\Longrightarrow$ need to rely on gradients computed from a (learned) model of the world.

Thank you!