On the difficulty of training Recurrent Neural Networks

By R. Pascanu, T. Mikolov, Yoshua Bengio

Ludwig Winkler

TU Berlin

February 22, 2017

Outline

Notation

Input at time t	$\boldsymbol{x}_t$
Hidden state at time t	$\boldsymbol{h}_t$
Output at time t	$\boldsymbol{y}_t$
Input weights	$\boldsymbol{W}_1$
Recurrent weights	$\boldsymbol{W}_{rec}$
Output weights	$\boldsymbol{W}_2$
Biases	$\boldsymbol{b}_1, \boldsymbol{b}_2$
Model parameters	$\boldsymbol{\theta} = \{\boldsymbol{W}_1, \boldsymbol{W}_2, \boldsymbol{W}_{\text{rec}}, \boldsymbol{b}_1, \boldsymbol{b}_2\}$
Activation function	$\sigma(\boldsymbol{x}_t)$
Cost function at time t	$\varepsilon_{\boldsymbol{t}}$
'Immediate' derivative at time t	$\frac{\partial^+}{\partial \boldsymbol{\theta}}$

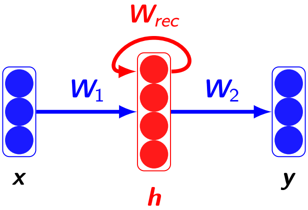
Recurrent Neural Network

- Neural Network with constant connections through time
- Hidden state input $\boldsymbol{x}_t$ and $\boldsymbol{h}_{t-1}$
- Retains past input information in $\boldsymbol{h}_t$
- For a simple three layer network the hidden states are updated as

$$
\begin{aligned}
\boldsymbol{h}_t &= \mathcal{F}\left(\boldsymbol{x}_t, \boldsymbol{h}_{t-1}; \theta\right) \\
\boldsymbol{h}_t &= \sigma\left(\boldsymbol{W}_{rec} \cdot \boldsymbol{h}_{t-1} + \boldsymbol{W}_1 \cdot \boldsymbol{x}_t + \boldsymbol{b}_1\right) \\
\boldsymbol{y}_t &= \sigma\left(\boldsymbol{W}_2 \cdot \boldsymbol{h}_t\right)
\end{aligned}
$$

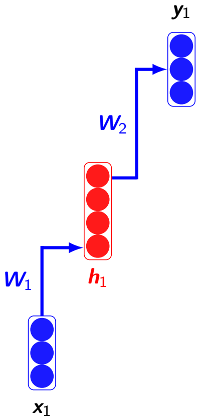
Recurrent Neural Network

- Neural Network with constant connections through time
- Hidden state input $\boldsymbol{x}_t$ and $\boldsymbol{h}_{t-1}$
- Retains past input information in $\boldsymbol{h}_t$
- For a simple three layer network the hidden states are updated as



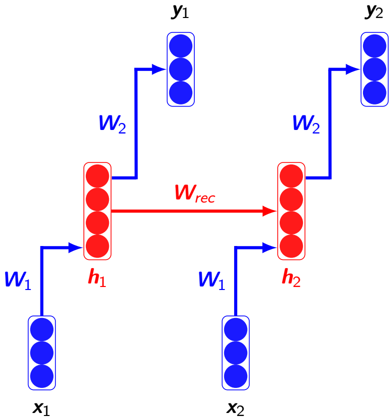
Recurrent Neural Network

- RNNs can be unfolded to visualize their recurrent information flow



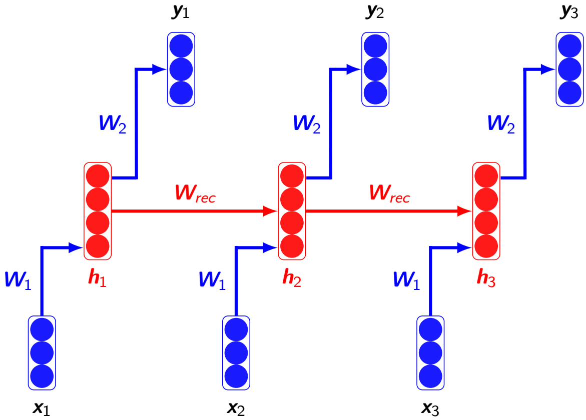
Recurrent Neural Network

- RNNs can be unfolded to visualize their recurrent information flow



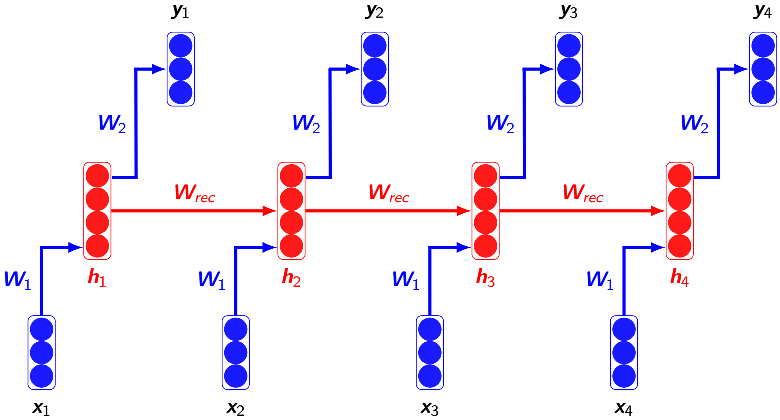
Recurrent Neural Network

- RNNs can be unfolded to visualize their recurrent information flow



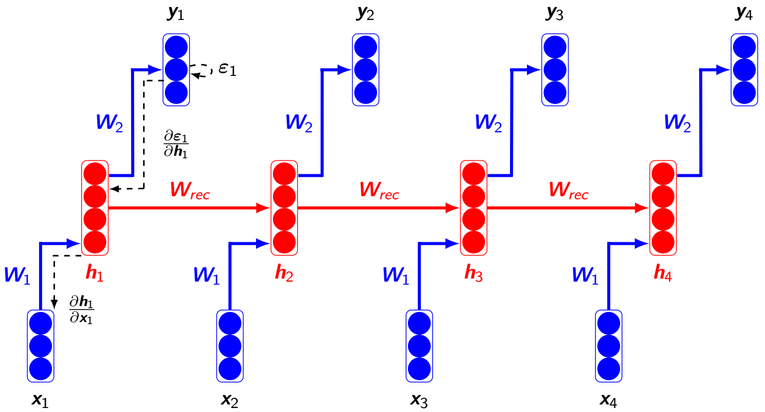
Recurrent Neural Network

- RNNs can be unfolded to visualize their recurrent information flow



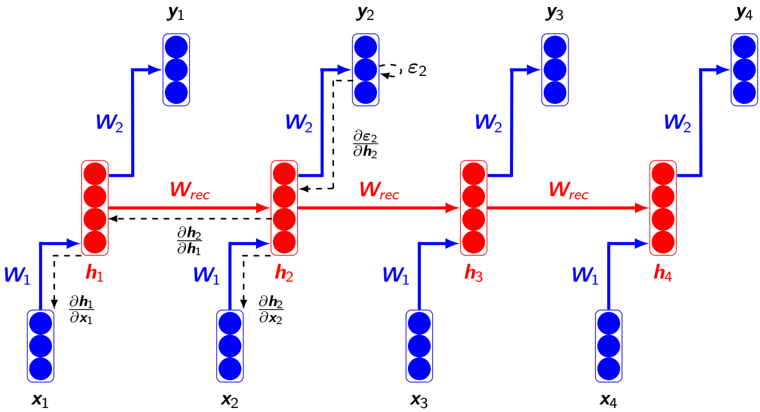
Error Propagation in Unfolded RNN

- Trained with Backpropagation Through Time (BPTT)



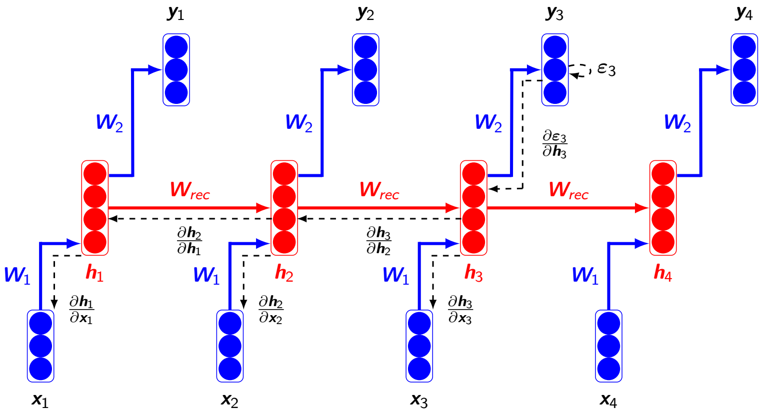
Error Propagation in Unfolded RNN

- Trained with Backpropagation Through Time (BPTT)



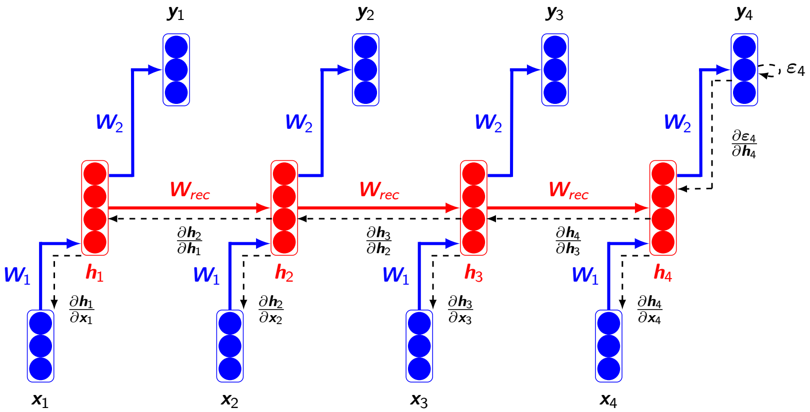
Error Propagation in Unfolded RNN

- Trained with Backpropagation Through Time (BPTT)



Error Propagation in Unfolded RNN

- Trained with Backpropagation Through Time (BPTT)



Error Propagation in RNN

- An unfolded RNN can be interpreted as a deep neural network
- The cost $\mathcal{E}$ can decomposed into its temporal contributions

$$\frac{\partial \mathcal{E}}{\partial \boldsymbol{\theta}} = \sum_{1 \leq t \leq T} \frac{\partial \mathcal{E}_t}{\partial \boldsymbol{\theta}}$$

- Gradient for all 'temporal' and 'spatial' parameters of the model

Error Propagation in RNN

- Chain rule for the gradients over all past time steps k

$$\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{\theta}} = \sum_{1 \leq k \leq t} \left(\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{h}_t} \frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{h}_k} \frac{\partial^+ \boldsymbol{h}_k}{\partial \boldsymbol{\theta}} \right)$$

- Cost function for the temporal and spatial parameters of the model
- $\frac{\partial^+ \boldsymbol{h}_k}{\partial \boldsymbol{\theta}}$ is the 'immediate' derivative disregarding recurrent connections more than one time step away

Error Propagation in RNN

- $\left.\frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{h}_k}\right|_{1 \leq k \leq t}$ calculates the gradients for the time steps $k \leq t$

$$\left.\frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{h}_k}\right|_{1 \leq k \leq t} = \prod_{k<i\leq t} \frac{\partial \boldsymbol{h}_i}{\partial \boldsymbol{h}_{i-1}} = \prod_{k<i\leq t} \boldsymbol{W}_{rec}^T diag(\sigma'(\boldsymbol{h}_{i-1}))$$

- Propagates the error to time step k given the cost at time step t

Analytical Analysis of the Gradients

- Using the identity function for simplification we obtain

$$\boldsymbol{h}_t = \boldsymbol{W}_{rec} \cdot \boldsymbol{h}_{t-1} + \boldsymbol{W}_1 \boldsymbol{x}_t + \boldsymbol{b}_1$$

- The derivative for the $\ell = t - k$ time steps of BPTT is

$$\left.\frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{h}_k}\right|_{1 \leq k \leq t} = \prod_{k < i \leq t} \frac{\partial \boldsymbol{h}_i}{\partial \boldsymbol{h}_{i-1}} = \left(\boldsymbol{W}_{\text{rec}}^T\right)^\ell$$

Power Iteration Method

- Iterative eigenvector algorithm that will result in $\boldsymbol{v}_1$ with $|\lambda_1| > |\lambda_i|$

$$\boldsymbol{b}_{k+1} = \frac{\boldsymbol{A}\boldsymbol{b}_k}{\|\boldsymbol{A}\boldsymbol{b}_k\|}$$

- Taking the recurrent relation into account we can reformulate the method as

$$\boldsymbol{b}_k = \frac{\boldsymbol{A}^k \boldsymbol{b}_0}{\|\boldsymbol{A}^k \boldsymbol{b}_0\|}$$

Power Iteration Method

- Let λ_i and $\boldsymbol{v}_i$ be the ordered eigenvalues and their corresponding eigenvectors of matrix $\boldsymbol{A}$

$$\boldsymbol{b}_0 = c_1 \boldsymbol{v}_1 + c_2 \boldsymbol{v}_2 + \cdots + c_m \boldsymbol{v}_m$$

- The product $\boldsymbol{A}^k \boldsymbol{b}_0$ can be rewritten with $\boldsymbol{A}^k \boldsymbol{v}_i = \lambda_i^k \boldsymbol{v}_i$ as

$$\begin{aligned}
\boldsymbol{A}^k \boldsymbol{b}_0 &= c_1 \boldsymbol{A}^k \boldsymbol{v}_1 + c_2 \boldsymbol{A}^k \boldsymbol{v}_2 + \cdots + c_m \boldsymbol{A}^k \boldsymbol{v}_m \\
&= c_1 \lambda_1^k \boldsymbol{v}_1 + c_2 \lambda_2^k \boldsymbol{v}_2 + \cdots + c_m \lambda_m^k \boldsymbol{v}_m \\
&= c_1 \lambda_1^k \boldsymbol{v}_1 + \lambda_1^k \sum_{j=2}^{m} c_j \underbrace{\left(\frac{\lambda_j}{\lambda_1}\right)^k}_{|\lambda_j/\lambda_1|<1} \boldsymbol{v}_j \approx c_1 \lambda_1^k \boldsymbol{v}_1
\end{aligned}$$

Analytical Analysis

- Eigendecomposition of $\boldsymbol{W}_{rec}^{T}$ gives eigenvalues λ_i and their orthogonal eigenvectors $\boldsymbol{q}_i$
- Using $\boldsymbol{q}_i$ we can linearly decompose $\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{h}_t}$

$$\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{h}_t} = \sum_{i=1}^{N} c_i \boldsymbol{q}_i^T$$

$$\sum_{1 \leq k \leq t} \frac{\partial \mathcal{E}_t}{\partial \boldsymbol{h}_t} \frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{h}_k} = \sum_{i=1}^{N} c_i \boldsymbol{q}_i^T \left(\boldsymbol{W}_{\text{rec}}^{T} \right)^{\ell} = \sum_{i=1}^{N} c_i \Big(\underbrace{\boldsymbol{W}_{\text{rec}}^{\ell} \boldsymbol{q}_i}_{= \lambda_i^{\ell} \boldsymbol{q}_i} \Big)^{T}$$

$$= c_1 \lambda_1^{\ell} \boldsymbol{q}_1^T + \lambda_1^{\ell} \sum_{j=2}^{N} c_j \underbrace{\left(\frac{\lambda_j}{\lambda_1} \right)^{\ell}}_{|\lambda_j / \lambda_1| < 1} \boldsymbol{q}_j^T \approx c_1 \lambda_1^{\ell} \boldsymbol{q}_1^T$$

Analytical Analysis

Non-Unitarian Case

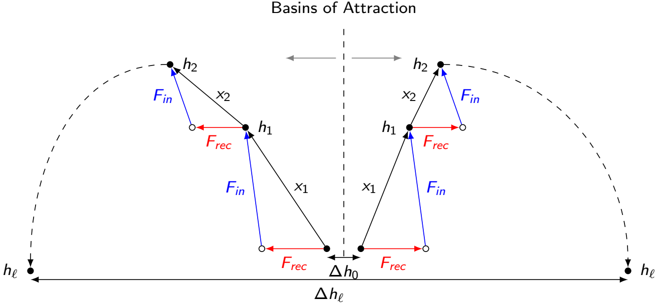
- $\sigma'(\boldsymbol{h}_k)$ and λ_1 of $\boldsymbol{W}_{\text{rec}}$ are bounded by $\|diag\,(\sigma'(\boldsymbol{h}_k))\| \leq \gamma$ and $\lambda_1 < \frac{1}{\gamma}$

$$\left\| \frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{h}_{t-1}} \right\| \leq \|\boldsymbol{W}_{\text{rec}}\| \left\| diag\,\left(\sigma'(\boldsymbol{h}_{t-1})\right) \right\| < \frac{1}{\gamma}\gamma < 1$$

- *Sufficient* that $\lambda_1 < 1$ for gradients to vanish
- *Necessary* that $\lambda_1 > 1$ for gradients to explode

Similarities to Dynamical Systems

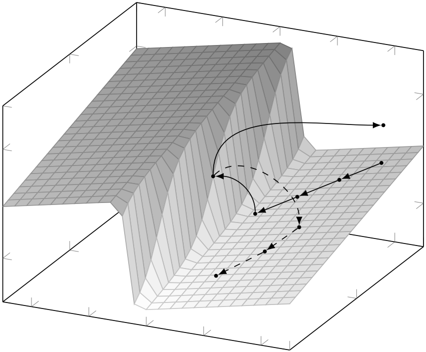
- System approaches attractor asymptotically if in basin of attraction
- Small differences in initialization of $\boldsymbol{W}_{\text{rec}}$ lead to different attractors



$$F_{RNN}(x) = F_{rec}(h) + F_{in}(x)$$

Geometric Interpretation

- $\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{\theta}}$ tends to explode along $\boldsymbol{q}_1$ of $\boldsymbol{W}_{rec}$
- Error surface becomes very steep if $\boldsymbol{q}_1$ and $\frac{\partial \mathcal{E}_t}{\partial \boldsymbol{\theta}}$ are aligned
- Regularization needed to prevent excessive gradient descent steps



Gradient-based Methods

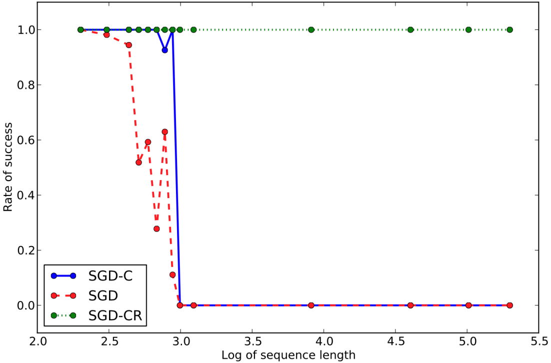
- Clipping exploding gradients if $\|\boldsymbol{g}\| \geq K$ with $\boldsymbol{g} \leftarrow K \frac{\boldsymbol{g}}{\|\boldsymbol{g}\|}$
- Vanishing gradients rescaled with regularization term Ω

$$\Omega = \sum_{1 \leq k \leq t} \Omega_k = \sum_{1 \leq k \leq t} \left(\frac{\left\| \frac{\partial \mathcal{E}_t}{\partial \boldsymbol{h}_k} \frac{\partial \boldsymbol{h}_k}{\partial \boldsymbol{h}_{k-1}} \right\|}{\left\| \frac{\partial \mathcal{E}_t}{\partial \boldsymbol{h}_k} \right\|} - 1 \right)^2$$

- Can result in a 'tug-of-war' scenario where clipping and regularization work diametrically opposed

Experiments and Results

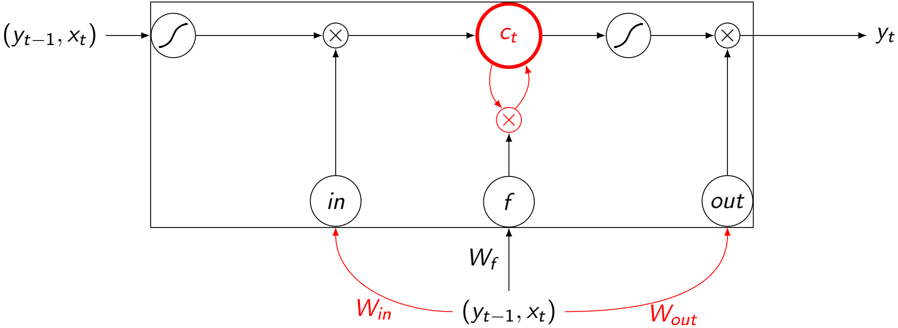
- Sequence with $\{A, B\}$ at beginning and middle
  e.g. $(A, \#, \#, \ldots, \#, \#, B, \#, \#, \ldots, \#, \#) = (A, B)$
- Can generalize to sequences twice as long as the training data



Long Short-Term Memory

- Degenerating gradients are caused by passing the error through many time steps: Can we store the information instead?
- LSTMs store the gradient in a memory cells
- Used nowadays in almost all commercial speech recognition systems
- Come in many different flavors with 'Gated Recurrent Unit' being a very popular one (Reset & Update gate)

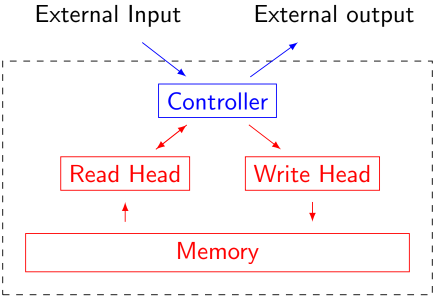
Long Short-Term Memory



He *turned* the radio which was built by Motorola in 1965 *off*.

Neural Turing Machine

- NTM learns to read, write and process information
- Differentiable attention mechanism to interact with memory
- Content and location focusing



Thank You For Your Attention

Sources

- Pascanu, R., Mikolov, T., Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning*, 1310–1318
- Graves, A. (2013). Generating Sequences With Recurrent Neural Networks, arXiv:1308.0850v5
- Graves, A., Wayne, G., Danihelka, I. (2014). Neural Turing Machines, arXiv:1410.5401v2
- Kumar, A. et al (2016). Ask Me Anything: Dynamic Memory Networks for Natural Language Processing. In *Proceedings of the 33rd International Conference on Machine Learning*, 1378–1387