On the Difficulty of Training Recurrent Neural Networks

Razvan Pascanu, Tomas Mikolov, Yoshua Bengio

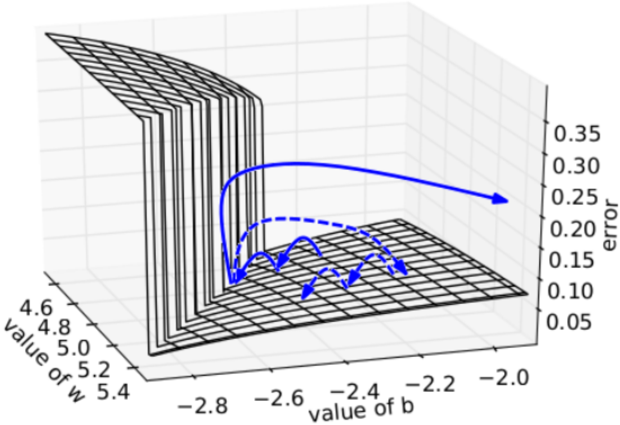
December 4, 2015

Overview

Contents

- Exploding Gradients
- Vanishing Gradients
- Related Work
- Paper Contribution
- Experimental Results

Exploding Gradients



Exploding Gradients

Gradient Descent Method:

$$w_{n+1} = w_n - \eta \nabla f(w_n)$$

- w: weight
- n: n-th iteration
- η: learning rate
- $f(w)$: cost function

Vanishing Gradients

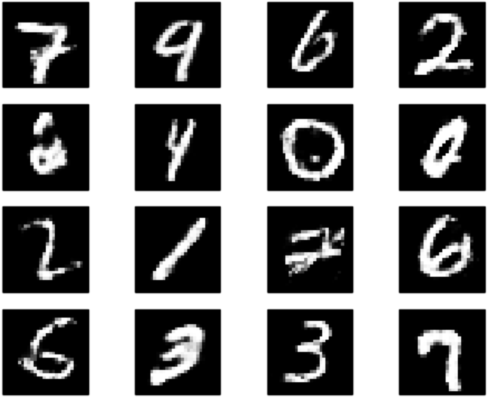
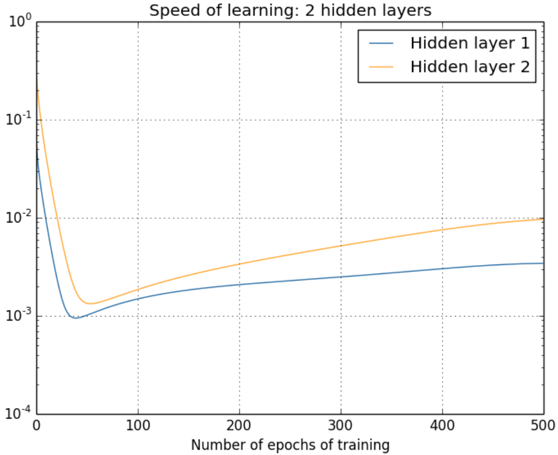
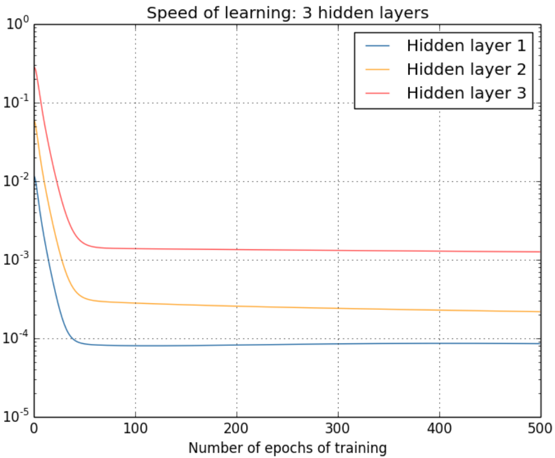


Figure: Sample of MNIST dataset

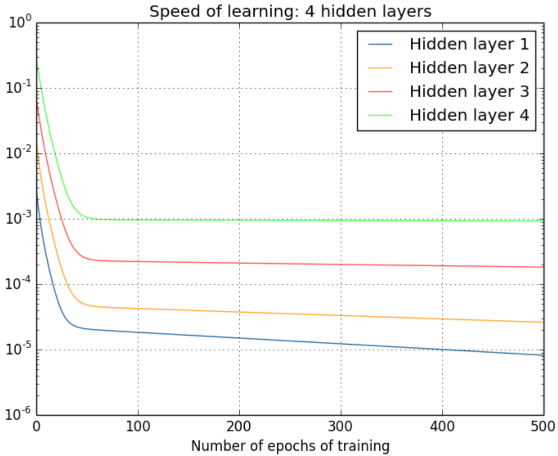
Vanishing Gradients



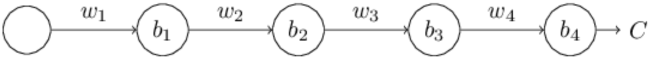
Vanishing Gradients



Vanishing Gradients

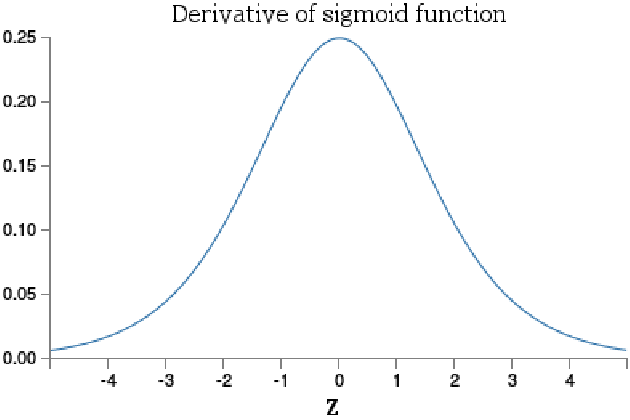


Vanishing Gradients



$$\frac{\partial C}{\partial b_1} = \sigma'(z_1)\, w_2 \sigma'(z_2)\, w_3 \sigma'(z_3)\, w_4 \sigma'(z_4)\, \frac{\partial C}{\partial a_4}$$

Vanishing Gradients

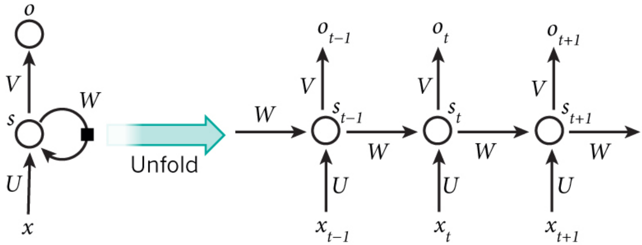


Vanishing Gradients

$$\frac{\partial C}{\partial b_1} = \sigma'(z_1)\overbrace{w_2\sigma'(z_2)}^{<\frac{1}{4}}\overbrace{w_3\sigma'(z_3)}^{<\frac{1}{4}}\underbrace{w_4\sigma'(z_4)\frac{\partial C}{\partial a_4}}_{\text{common terms}}$$

$$\frac{\partial C}{\partial b_3} = \sigma'(z_3)\overbrace{w_4\sigma'(z_4)\frac{\partial C}{\partial a_4}}^{\text{common terms}}$$

Vanishing Gradients



Vanishing Gradients

$$\frac{\partial E}{\partial W} = \sum_t \frac{\partial E_t}{\partial W}$$

$$\frac{\partial E_t}{\partial W} = \frac{\partial E_t}{\partial \hat{o}_t} \frac{\partial \hat{o}_t}{\partial s_t} \frac{\partial s_t}{\partial W}$$

$$\frac{\partial E_t}{\partial W} = \sum_{k=0}^{t} \frac{\partial E_t}{\partial \hat{o}_t} \frac{\partial \hat{o}_t}{\partial s_t} \frac{\partial s_t}{\partial s_k} \frac{\partial s_k}{\partial W}$$

$$\frac{\partial E_t}{\partial W} = \sum_{k=0}^{t} \frac{\partial E_t}{\partial \hat{o}_t} \frac{\partial \hat{o}_t}{\partial s_t} \left(\prod_{j=k+1}^{t} \frac{\partial s_j}{\partial s_{j-1}} \right) \frac{\partial s_k}{\partial W}$$

Related Work

Teacher Forcing

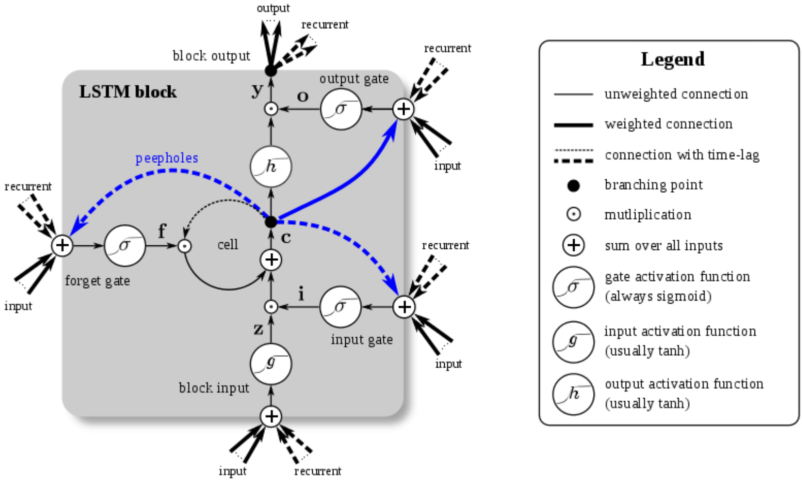
- Proposed by Doya (1993)
- Use targets for some or all hidden units to converge towards
- Assumes model asymptotic behaviour is the same required by the target
- Requires target to be defined at every time step
- Reduces exploding gradients
- Not practical and difficult

Related Work

Long Short Term Memory Architecture (LSTM)

- Proposed by Hochreiter and Schmidhuber (1997)
- Introduces Input, Ouput and Forget gates
- linear unit with self connection of value 1
- Deals with vanishing but not exploding graidents

Related Work



Related Work

$$
\begin{aligned}
\mathbf{z}^t &= g(\mathbf{W}_z\mathbf{x}^t + \mathbf{R}_z\mathbf{y}^{t-1} + \mathbf{b}_z) && \textit{block input} \\
\mathbf{i}^t &= \sigma(\mathbf{W}_i\mathbf{x}^t + \mathbf{R}_i\mathbf{y}^{t-1} + \mathbf{p}_i \odot \mathbf{c}^{t-1} + \mathbf{b}_i) && \textit{input gate} \\
\mathbf{f}^t &= \sigma(\mathbf{W}_f\mathbf{x}^t + \mathbf{R}_f\mathbf{y}^{t-1} + \mathbf{p}_f \odot \mathbf{c}^{t-1} + \mathbf{b}_f) && \textit{forget gate} \\
\mathbf{c}^t &= \mathbf{i}^t \odot \mathbf{z}^t + \mathbf{f}^t \odot \mathbf{c}^{t-1} && \textit{cell state} \\
\mathbf{o}^t &= \sigma(\mathbf{W}_o\mathbf{x}^t + \mathbf{R}_o\mathbf{y}^{t-1} + \mathbf{p}_o \odot \mathbf{c}^t + \mathbf{b}_o) && \textit{output gate} \\
\mathbf{y}^t &= \mathbf{o}^t \odot h(\mathbf{c}^t) && \textit{block output}
\end{aligned}
$$

Related Work

Hessian-Free Optimizer with Structural Damping

- Proposed by Sutskever et al (2011)

(For Vanishing Problem) "Presumably this method works because in high dimensional spaces there is a high probability for long term components to be orthogonal to short term ones. This would allow the Hessian to rescale these components independently."

Echo State Networks

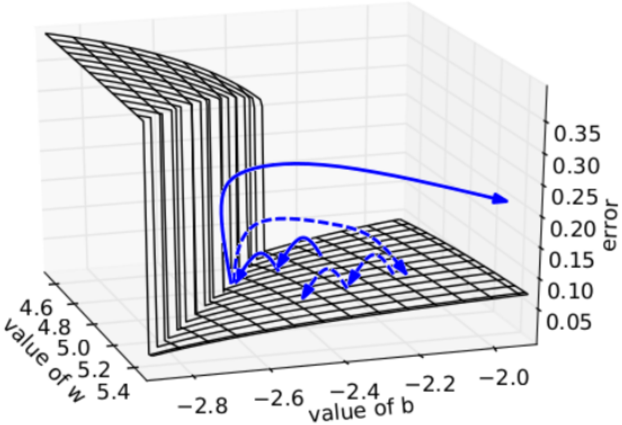
- Proposed by Jaeger and Haas (2014)
- Avoid exploding and vanishing problem by not learning W_{rec} and W_{in}
- Sparsely connected hidden layer (typically 1%)
- Connectivity and weights are fixed and randomly assigned
- Only weights of output are learned

Scaling Gradient

Algorithm 1 Pseudo-code for norm clipping

$\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$
if $\|\hat{\mathbf{g}}\| \geq threshold$ **then**
  $\hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|} \hat{\mathbf{g}}$
end if

Scaling Gradient



Vanishing Gradient Regularizer

- Vanishing gradients prevents long term latching
- Increasing the norm $\frac{\partial s_t}{\partial s_k}$ will increase the sensitivity to all inputs.
- Creates larger errors
- In turn causes convergence to suffer
- Solution is to create a regularizer

Vanishing Gradient Regularizer

$$\frac{\partial E_t}{\partial W} = \sum_{k=0}^{t} \frac{\partial E_t}{\partial \hat{o}_t} \frac{\partial \hat{o}_t}{\partial s_t} \frac{\partial s_t}{\partial s_k} \frac{\partial s_k}{\partial W}$$

$$\frac{\partial E_t}{\partial W} = \sum_{k=0}^{t} \frac{\partial E_t}{\partial \hat{o}_t} \frac{\partial \hat{o}_t}{\partial s_t} \left(\prod_{j=k+1}^{t} \frac{\partial s_j}{\partial s_{j-1}} \right) \frac{\partial s_k}{\partial W}$$

Vanishing Gradient Regularizer

$$\Omega = \sum_k \Omega_k = \sum_k \left(\frac{\left\| \frac{\delta \varepsilon}{\delta x_{k+1}} \frac{\delta x_{k+1}}{\delta x_k} \right\|}{\left\| \frac{\delta \varepsilon}{\delta x_{k+1}} \right\|} - 1 \right)^2$$

- Prefers solutions which the error preserves the norm as it travels back in time

The Temporal Order Problem

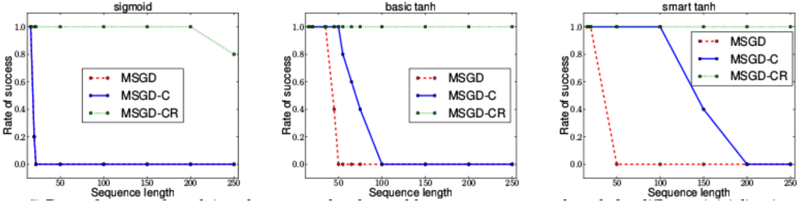
- Long random sequence of discrete symbols
- **Beginning** and **middle** of sequnence will contain one of $\{A, B\}$
- Task is to classify order of A, B
- Task successful if only 1% of $10,000$ random sequences are misclassified

Three different RNN intializations were performed for the experiment:

- **sigmoid unit network**: $W_{rec}, W_{in}, W_{out} \sim \mathcal{N}(0, 0.01)$
- **basic tanh unit network**: $W_{rec}, W_{in}, W_{out} \sim \mathcal{N}(0, 0.1)$
- **smart tanh unit network**: $W_{rec}, W_{in}, W_{out} \sim \mathcal{N}(0, 0.01)$

Of the three RNN networks three different optimizer configurations were used:

- **MSGD**: Mini-batch Stochastic Gradient Decent
- **MSGD-C**: MSGD with Gradient Clipping
- **MSGD-CR**: MSGD-C with Regularization

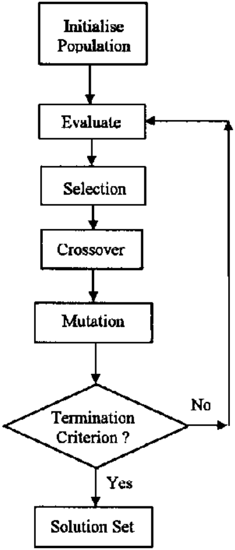


- 5 runs
- 50 hidden unit model
- Learning rate of 0.01
- Threshold of 1.0 (for gradient clipping)

Data set	Data fold	MSGD	MSGD+C	MSGD+CR	State of the art for RNN	State of the art
Piano-midi.de (NLL)	train	6.87	6.81	**7.01**	7.04	6.32
	test	7.56	7.53	**7.46**	7.57	7.05
Nottingham (NLL)	train	3.67	3.21	**2.95**	3.20	1.81
	test	3.80	3.48	**3.36**	3.43	2.31
MuseData (NLL)	train	8.25	6.54	**6.43**	6.47	5.20
	test	7.11	7.00	**6.97**	6.99	5.60
Penn Treebank 1 step (bits/char)	train	1.46	**1.34**	1.36	N/A	N/A
	test	1.50	1.42	**1.41**	**1.41**	1.37
Penn Treebank 5 steps (bits/char)	train	N/A	3.76	**3.70**	N/A	N/A
	test	N/A	3.89	**3.74**	N/A	N/A

Questions?

The Genetic Approach



The Genetic Approach

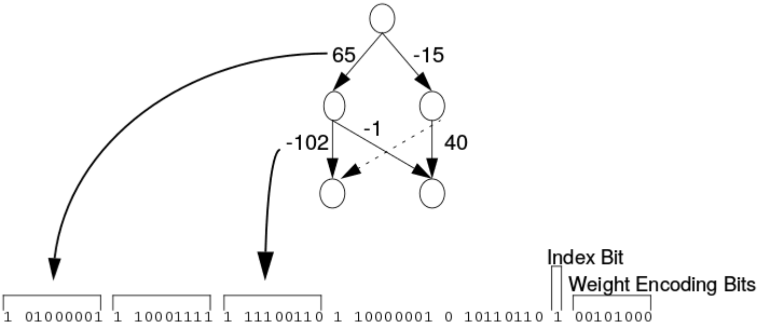


Figure: Whitley's GENITOR Algorithm

The Genetic Approach

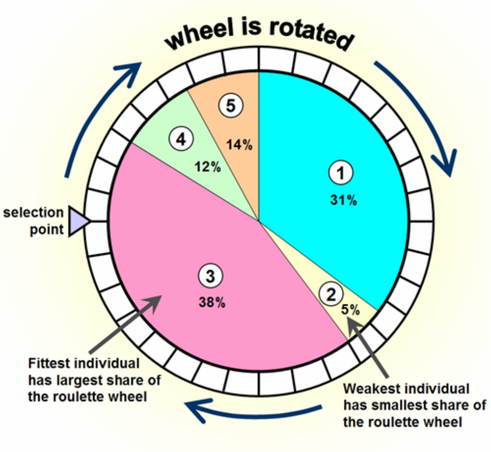


Figure: Fitness Proportionate Selection

The Genetic Approach

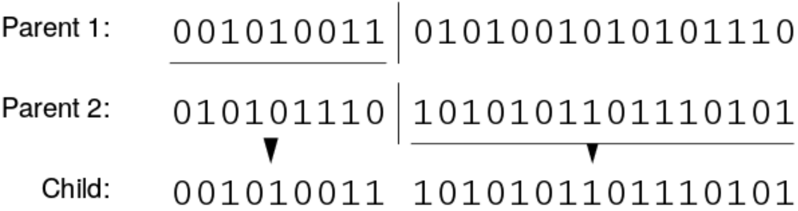


Figure: Point Crossover

The Genetic Approach

Example Applications:

- MarI/O (NEAT Algorithm): Plays a level of Mario ▸ YouTube
- Evolving Gaits for Legged Robots (HyperNEAT) ▸ YouTube