



UNIVERSITÀ
DEGLI STUDI
FIRENZE



UNIVERSIDADE DE COIMBRA

Optimization methods for Recurrent Neural Networks training

Giulio Galvan

Relatori:

Prof. Marco Sciandrone

Prof. Luís Nunes Vicente

Correlatore:

Prof. Fabio Schoen

20th April 2016

Introduction

Definition of RNN

Given an input sequence $\{\mathbf{u}^t\}_{t=1,\dots,T}$, with $\mathbf{u}^t \in \mathbb{R}^p$, an RNN yields the output sequence $\{\mathbf{y}^t\}_{t=1,\dots,T}$, with $\mathbf{y}^t \in \mathbb{R}^o$, defined by the following:

$$\mathbf{y}^t \triangleq F(\mathbf{z}^t) \quad (1)$$

$$\mathbf{z}^t \triangleq W^{out} \cdot \mathbf{a}^t + \mathbf{b}^{out} \quad (2)$$

$$\mathbf{a}^t \triangleq W^{rec} \cdot \mathbf{h}^{t-1} + W^{in} \cdot \mathbf{u}^t + \mathbf{b}^{rec} \quad (3)$$

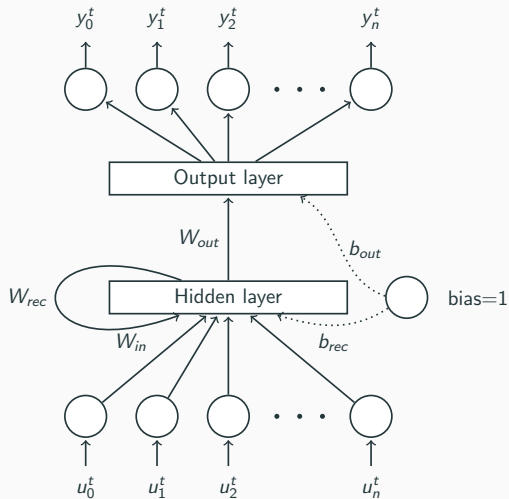
$$\mathbf{h}^t \triangleq \sigma(\mathbf{a}^t) \quad (4)$$

$$\mathbf{h}^0 \triangleq \mathbf{0}, \quad (5)$$

where $\sigma(\cdot) : \mathbb{R} \rightarrow \mathbb{R}$ is a non linear function applied element-wise called **activation function**, $F(\cdot)$ is called **output function**.

The **parameters** of the net are $\{W^{out}, W^{in}, W^{rec}, \mathbf{b}^{rec}, \mathbf{b}^{out}\}$.

The model



The optimization problem

Given a dataset D :

$$D \triangleq \{ \{ \bar{\mathbf{u}}^t \}_{t=1, \dots, T(i)}^{(i)}, \{ \bar{\mathbf{y}}^t \}_{t=1, \dots, T(i)}^{(i)}; i = 1, \dots, N \} \quad (6)$$

we define a loss function $L_D(\mathbf{x})$ over D as

$$L_D(\mathbf{x}) \triangleq \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^{T(i)} L_t(\bar{\mathbf{y}}_t^{(i)}, \mathbf{y}_t^{(i)}(\mathbf{x})), \quad (7)$$

where $L_t(\cdot, \cdot)$ is an arbitrary loss function for the time step t and \mathbf{x} represents all the parameters of the network. The problem is

$$\min_{\mathbf{x}} L_D(\mathbf{x}). \quad (8)$$

Some learning examples

- Regression: mean squared error, linear output

$$L(\mathbf{y}, \bar{\mathbf{y}}) = \frac{1}{M} \sum_{i=1}^M (y_i - \bar{y}_i)^2, \quad F(\mathbf{z}) = \mathbf{z}. \quad (9)$$

- Binary classification: hinge loss, linear output

$$L(y, \bar{y}) = \max(0, 1 - \bar{y} \cdot y), \quad F(z) = z. \quad (10)$$

- Multi-way classification: cross entropy loss, softmax output

$$L(\mathbf{y}, \bar{\mathbf{y}}) = -\frac{1}{M} \sum_{i=1}^M \log(y_i) \cdot \bar{y}_i, \quad F(z_j) = \frac{e^{z_j}}{\sum_{i=1}^M e^{z_i}}. \quad (11)$$

Stochastic gradient descent (SGD)

Algorithm 1: Stochastic gradient descent

Data:

$D = \{\langle \bar{\mathbf{u}}^{(i)}, \bar{\mathbf{y}}^{(i)} \rangle\}$: training set

\mathbf{x}_0 : candidate solution

m : size of each mini-batch

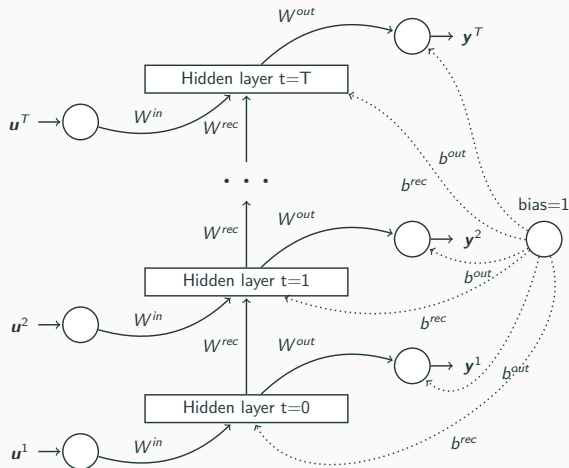
Result:

\mathbf{x} : solution

```
1  $\mathbf{x} \leftarrow \mathbf{x}_0$ 
2 while stop criterion do
3    $I \leftarrow$  select  $m$  training example  $\in D$ 
4    $\alpha \leftarrow$  compute learning rate
5    $\mathbf{x} \leftarrow \mathbf{x} - \alpha \sum_{i \in I} \nabla_{\mathbf{x}} L(\mathbf{x}; \bar{\mathbf{u}}^{(i)}, \bar{\mathbf{y}}^{(i)})$ 
6 end
```

Gradient of a RNN

Gradient structure: unfolding



Consider the case where the loss function $L(\mathbf{y}, \bar{\mathbf{y}})$ is defined only on the last step τ . Let $g(\mathbf{x}) : \mathbb{R}$ be the function defined by

$$g(\mathbf{x}) \triangleq L(F(\mathbf{z}^\tau(\bar{\mathbf{u}}; \mathbf{x}), \bar{\mathbf{y}}^\tau)).$$

We compute the gradient as:

$$\frac{\partial g}{\partial W^{rec}} = \frac{\partial g}{\partial \mathbf{a}^\tau} \cdot \frac{\partial \mathbf{a}^\tau}{\partial W^{rec}} \quad (12)$$

$$= \nabla L^\tau \cdot J(F) \cdot \frac{\partial \mathbf{z}^\tau}{\partial \mathbf{a}^\tau} \cdot \frac{\partial \mathbf{a}^\tau}{\partial W^{rec}}. \quad (13)$$

In matrix notation we have:

$$\frac{\partial \mathbf{a}^t}{\partial W^{rec}} = \sum_{k=1}^t \frac{\partial \mathbf{a}^t}{\partial \mathbf{a}^k} \cdot \frac{\partial^+ \mathbf{a}^k}{\partial W^{rec}} \quad (14)$$

$$\frac{\partial^+ \mathbf{a}^k}{\partial W_j^{rec}} = \begin{bmatrix} h_j^k & 0 & \dots & \dots & 0 \\ 0 & h_j^k & \dots & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & \dots & \dots & h_j^k \end{bmatrix} \quad (15)$$

$$\frac{\partial \mathbf{a}^t}{\partial \mathbf{a}^k} = \frac{\partial \mathbf{a}^t}{\partial \mathbf{a}^{k+1}} \cdot \text{diag}(\sigma'(\mathbf{a}^k)) \cdot W^{rec} \quad (16)$$

$$= \prod_{i=t-1}^k \text{diag}(\sigma'(\mathbf{a}^i)) \cdot W^{rec}. \quad (17)$$

The derivatives with respect to the other variables are computed in a similar fashion.

Gradient structure: temporal components

Putting all together:

$$\nabla_{W^{rec}} g = \sum_{k=1}^{\tau} \frac{\partial g}{\partial \mathbf{a}^{\tau}} \cdot \frac{\partial \mathbf{a}^{\tau}}{\partial \mathbf{a}^k} \cdot \frac{\partial^+ \mathbf{a}^k}{\partial W^{rec}} \quad (18)$$

$$\triangleq \sum_{k=1}^{\tau} \nabla_{W^{rec}} L_{|k}. \quad (19)$$

We refer to $\nabla_{\mathbf{x}} g_{|k}$ as the **temporal gradient** for time step k w.r.t. the variable \mathbf{x} .

It is the gradient we would compute if we replicated the variable \mathbf{x} for each time step and took the derivatives w.r.t. to its k -th replicate.

The vanishing gradient problem

A pathological problem example

An input sequence:

marker	0	1	0	...	0	1	0	0
value	0.3	0.7	0.1	...	0.2	0.4	0.6	0.9

The predicted output should be the sum of the two one-marked positions (1.1).

Why is this a difficult problem? Because of its long time dependencies.

A pathological problem example

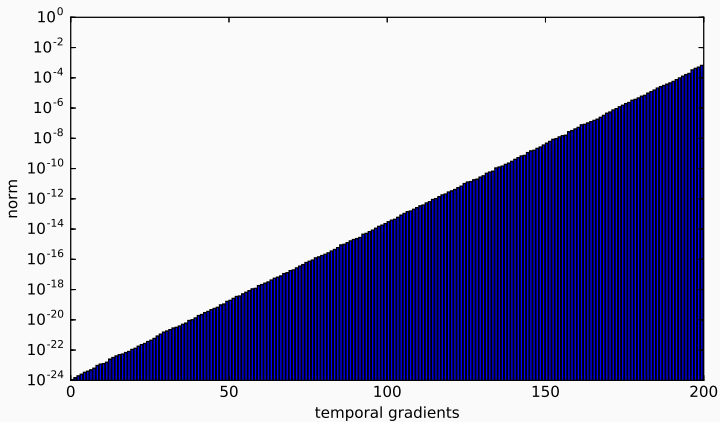
An input sequence:

marker	0	1	0	...	0	1	0	0
value	0.3	0.7	0.1	...	0.2	0.4	0.6	0.9

The predicted output should be the sum of the two one-marked positions (1.1).

Why is this a difficult problem? Because of its long time dependencies.

Vanishing gradient: an illustration



Vanishing gradient: a sufficient condition

$$\frac{\partial \mathbf{a}^t}{\partial \mathbf{a}^k} = \prod_{i=t-1}^k \text{diag}(\sigma'(\mathbf{a}^i)) \cdot W^{rec}. \quad (20)$$

Taking the singular value decomposition of W^{rec} :

$$W^{rec} = S \cdot D \cdot V^T \quad (21)$$

where S, V^T are squared orthogonal matrices and

$D \triangleq \text{diag}(\mu_1, \mu_2, \dots, \mu_r)$ is the diagonal matrix containing the singular values of W^{rec} . Hence:

$$\frac{\partial \mathbf{a}^t}{\partial \mathbf{a}^k} = \prod_{i=t-1}^k \text{diag}(\sigma'(\mathbf{a}^i)) \cdot S \cdot D \cdot V^T \quad (22)$$

Since U and V are orthogonal matrix, hence

$$\|U\|_2 = \|V^T\|_2 = 1,$$

and

$$\|diag(\lambda_1, \lambda_2, \dots, \lambda_r)\|_2 = \lambda_{max},$$

we get

$$\left\| \frac{\partial \mathbf{a}^t}{\partial \mathbf{a}^k} \right\|_2 = \left\| \left(\prod_{i=t-1}^k diag(\sigma'(\mathbf{a}^i)) \cdot S \cdot D \cdot V^T \right) \right\|_2 \quad (23)$$

$$\leq (\sigma'_{max} \cdot \mu_{max})^{t-k-1}. \quad (24)$$

A new SGD approach for training RNNs

The matrix W^{rec} is **scaled** to have a desired spectral radius.

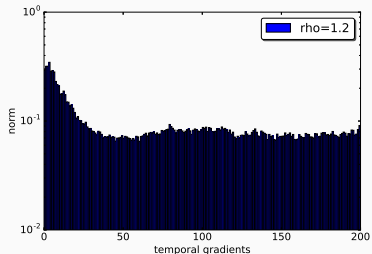
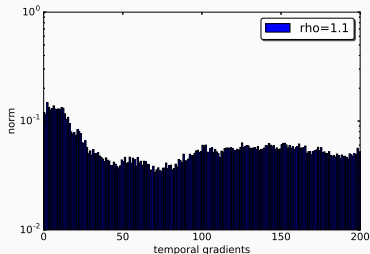
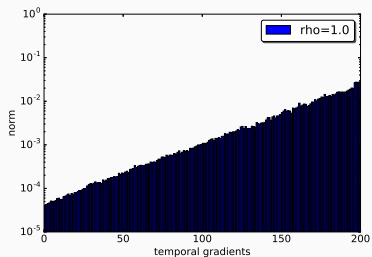
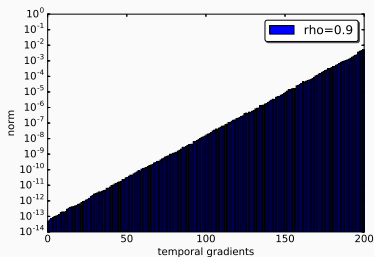
Algorithm 2: Recurrent weight matrix initialization scheme

Data:

ρ = desired spectral radius

- 1 $W_{rec} \sim \mathcal{N}(0, \sigma^2)$
 - 2 $r \leftarrow \text{spectral_radius}(W_{rec})$
 - 3 $W_{rec} \leftarrow \frac{\rho}{r} \cdot W_{rec}$
 - 4 **return** W_{rec}
-

The effect of initialization on the temporal gradients



The effect of initialization on the rate of success

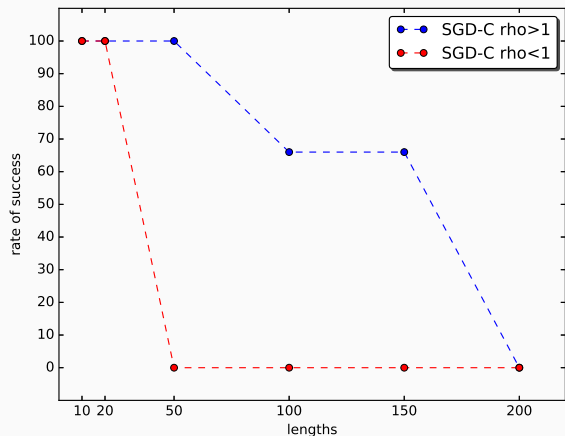


Figure 1: Rate of success for the temporal order task.

A different descent direction

Combine the temporal gradients to obtain a descent direction which does not suffer from the vanishing gradient problem.

- Normalize the temporal gradients:

$$s_t(\mathbf{x}) = \frac{\nabla L_t(\mathbf{x})}{\|\nabla L_t(\mathbf{x})\|}. \quad (25)$$

- Combine the normalized gradients in a convex way:

$$s(\mathbf{x}) = \sum_{t=1}^T \beta_t \cdot s_t(\mathbf{x}). \quad (26)$$

with $\sum_{t=1}^T \beta_t = 1, \beta_t > 0$ (randomly picked at each iteration).

- Introduce the gradient norm:

$$d(\mathbf{x}) = - \|\nabla L(\mathbf{x})\| \frac{s(\mathbf{x})}{\|s(\mathbf{x})\|}. \quad (27)$$

Algorithm 3: RNN training

Data:

$D = \{\langle \mathbf{x}^{(i)}, \mathbf{y}^{(i)} \rangle\}$: training set

m : size of each mini-batch

μ : constant learning rate

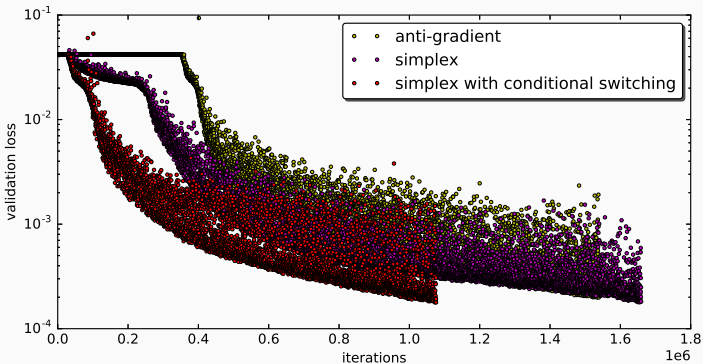
τ : gradient clipping threshold

ρ : initial spectral radius

ψ threshold for the direction norm

```
1  $W_{rec}, W_{in}, W_{out} \sim \mathcal{N}(0, \sigma^2)$ 
2  $\mathbf{b}_{out}, \mathbf{b}_{rec} \leftarrow 0$ 
3  $r \leftarrow \text{spectral\_radius}(W_{rec})$ 
4  $W_{rec} \leftarrow \frac{\rho}{r} \cdot W_{rec}$ 
5  $\theta_0 = [W_{rec}, W_{in}, W_{out}, \mathbf{b}_{out}, \mathbf{b}_{rec}]$ 
6 while stop criterion do
7    $I \leftarrow$  sample  $m$  training example  $\in D$ 
8    $\{\nabla_{\theta} L_t\}_{t=1}^T \leftarrow \text{compute\_temporal\_gradients}(\theta_k, I)$ 
9    $\mathbf{d}_k \leftarrow \text{simplex\_combination}(\{\nabla_{\theta} L_t\})$ 
10  if  $\|\nabla_{\theta} L(\theta_k)\|_2 > \psi$  then
11     $\mathbf{d}_k \leftarrow \nabla_{\theta} L(\theta_k)$ 
12  end
13   $\alpha_k = \begin{cases} \mu & \text{if } \|\mathbf{d}_k\|_2 \leq \tau \\ \frac{\mu \cdot \tau}{\|\mathbf{d}_k\|_2} & \text{otherwise} \end{cases}$ 
14   $\theta_{k+1} \leftarrow \theta_k + \alpha_k \mathbf{d}_k$ 
15   $k \leftarrow k + 1$ 
16 end
17 return  $\theta_k$ 
```

Effect of the simplex direction



(a) Loss for the addition task during training.

	anti-gradient	simplex with conditional switching
addition	1807466	1630666
temporal order	2164800	1010000

(b) Average number of iterations to converge.

A case study: Lupus disease prediction

Lupus disease prediction

The **goal** is to predict whether a patient will develop the lupus disease in the near future given its medical history.

Dataset:

- ▶ Gathered by the “Lupus Clinic”, Reumatologia, Università Sapienza, Rome.
- ▶ Patients have a different number of visits.
- ▶ Visits are not equally spaced over time.
- ▶ Small dataset (~ 100 negatives, 40 positives).

An example of a record of a patient

	Visit 0	Visit 1	Visit 2	Visit 3	Visit 4
age	44.23	44.63	44.77	44.98	45.58
MyasteniaGravis	0	0	0	0	0
arthritis	1	0	1	1	0
c3level	119	96	85.42	76	76
c4level	9	7	6	6	6
hematological	0	0	6	6	6
skinrash	0	0	0	0	0
sledai2kInferred	12	2	2	2	0
...
SDI	0	0	0	0	1

Numerical results for the lupus disease prediction

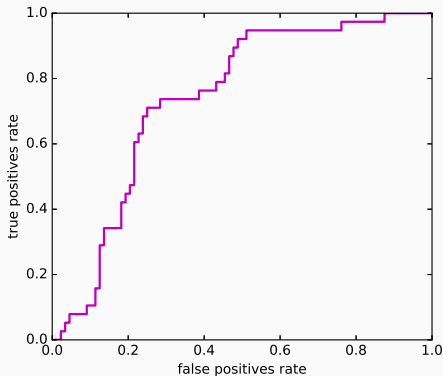





Figure 2: ROC curve. AUC score is 0.74

- Results considered promising by medical experts.
- Improvable when more data will be available.



-  J. Bayer, C. Osendorfer, N. Chen, S. Urban, and P. van der Smagt.
On fast dropout and its applicability to recurrent networks.
CoRR, abs/1311.0701, 2013.
-  Y. Bengio, N. Boulanger-Lewandowski, and R. Pascanu.
Advances in optimizing recurrent networks.
In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2013, Vancouver, BC, Canada, May 26-31, 2013*, pages 8624–8628. IEEE, 2013.
-  Y. Bengio, J. Louradour, R. Collobert, and J. Weston.
Curriculum learning.
In *Proceedings of the 26th Annual International Conference on Machine Learning, ICML '09*, pages 41–48, New York, NY, USA, 2009. ACM.



Y. Bengio, P. Simard, and P. Frasconi.

Learning long-term dependencies with gradient descent is difficult.



K. Cho, B. van Merriënboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio.

Learning phrase representations using RNN encoder-decoder for statistical machine translation.

In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL, pages 1724–1734, 2014.



J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio.

Empirical evaluation of gated recurrent neural networks on sequence modeling.

CoRR, abs/1412.3555, 2014.



J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio.

Gated feedback recurrent neural networks.

CoRR, abs/1502.02367, 2015.



A. Graves.

Generating sequences with recurrent neural networks.

CoRR, abs/1308.0850, 2013.



S. Hochreiter and J. Schmidhuber.

Long short-term memory.

Neural Comput., 9(8):1735–1780, Nov. 1997.



K. Hornik, M. Stinchcombe, and H. White.

Multilayer feedforward networks are universal approximators.

Neural Netw., 2(5):359–366, July 1989.



H. Hyotyniemi.

Turing machines are recurrent neural networks, 1996.



H. Jaeger, M. Lukosevicius, D. Popovici, and U. Siewert.

Optimization and applications of echo state networks with leaky- integrator neurons.

Neural Networks, 20(3):335–352, 2007.



R. Józefowicz, W. Zaremba, and I. Sutskever.

An empirical exploration of recurrent network architectures.

In F. R. Bach and D. M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Proceedings*, pages 2342–2350. JMLR.org, 2015.



M. Lukosevicius and H. Jaeger.

Reservoir computing approaches to recurrent neural network training.

Computer Science Review, 3(3):127–149, 2009.



M. Lukoševičius and H. Jaeger.

Survey: Reservoir computing approaches to recurrent neural network training.

Comput. Sci. Rev., 3(3):127–149, Aug. 2009.



W. Maass, T. Natschlaeger, and H. Markram.

Real-time computing without stable states: A new framework for neural computation based on perturbations.

Neural Computation, 14(11):2531–2560, 2002.



J. Martens and I. Sutskever.

Training deep and recurrent networks with hessian-free optimization.

In G. Montavon, G. B. Orr, and K. Müller, editors, *Neural Networks: Tricks of the Trade - Second Edition*, volume 7700 of *Lecture Notes in Computer Science*, pages 479–535. Springer, 2012.



T. Mikolov.

Statistical Language Models Based on Neural Networks.

PhD thesis, 2012.



T. Mikolov, A. Joulin, S. Chopra, M. Mathieu, and M. Ranzato.
Learning longer memory in recurrent neural networks.
CoRR, abs/1412.7753, 2014.



A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro.
Robust stochastic approximation approach to stochastic programming.
SIAM Journal on Optimization, 19(4):1574–1609, 2009.



Y. Nesterov.
A method of solving a convex programming problem with convergence rate $O(1/\sqrt{k})$.
Soviet Mathematics Doklady, 27:372–376, 1983.



R. Pascanu, T. Mikolov, and Y. Bengio.

On the difficulty of training recurrent neural networks.

In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, pages 1310–1318, 2013.



B. Polyak.

Some methods of speeding up the convergence of iteration methods.

USSR Computational Mathematics and Mathematical Physics, 4(5):1 – 17, 1964.



D. E. Rumelhart, G. E. Hinton, and R. J. Williams.

Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1.

chapter Learning Internal Representations by Error Propagation, pages 318–362. MIT Press, Cambridge, MA, USA, 1986.



F. Scarselli and A. C. Tsoi.

Universal approximation using feedforward neural networks: A survey of some existing methods, and some new results.

Neural Netw., 11(1):15–37, Jan. 1998.



N. N. Schraudolph.

Fast curvature matrix-vector products for second-order gradient descent.

Neural Computation, 14(7):1723–1738, 2002.



H. T. Siegelmann and E. D. Sontag.

Turing computability with neural nets.

Applied Mathematics Letters, 4:77–80, 1991.



N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov.

Dropout: a simple way to prevent neural networks from overfitting.

Journal of Machine Learning Research, 15(1):1929–1958, 2014.



I. Sutskever, J. Martens, G. E. Dahl, and G. E. Hinton.

On the importance of initialization and momentum in deep learning.

In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, volume 28 of *JMLR Proceedings*, pages 1139–1147. JMLR.org, 2013.



R. J. Williams and J. Peng.

An efficient gradient-based algorithm for on-line training of recurrent network trajectories.

Neural Computation, 2:490–501, 1990.



W. Zaremba and I. Sutskever.

Learning to execute.

CoRR, abs/1410.4615, 2014.



W. Zaremba, I. Sutskever, and O. Vinyals.

Recurrent neural network regularization.

CoRR, abs/1409.2329, 2014.