


Recurrent Neural Networks

Dr. Grégoire Montavon | Technische Universität Berlin | Machine Learning Group

Outline

Recurrent Neural Networks

Overview & Applications

RNNs from differential equations (Unfolding RNNs)

Statistics of RNNs

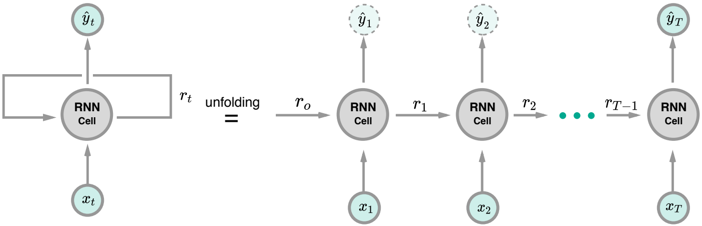
Exploding and Varnishing Gradient Problems

Echo State Networks (ESNs)

Gradient Descent with Momentum

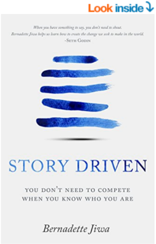
Long Short-Term Memory & Gated RNNs

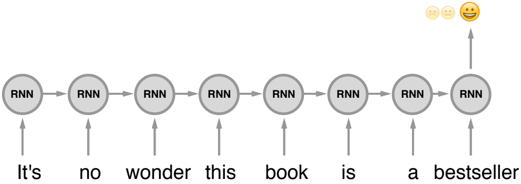
What are Recurrent Neural Networks (RNNs)?



What are Recurrent Neural Networks (RNNs)?

Example Applications

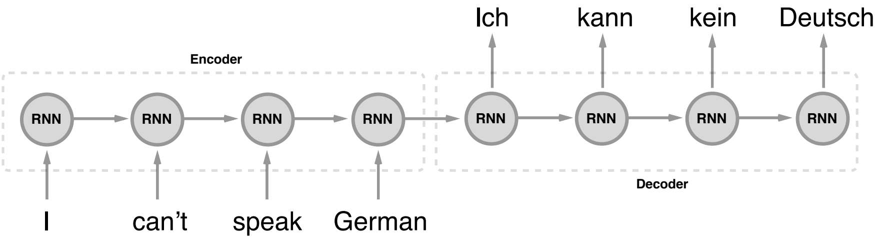




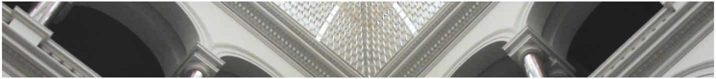
Sentiment Analysis

What are Recurrent Neural Networks (RNNs)?

Example Applications



Machine Translation (seq2seq architecture)



Technische Universität Berlin

What are Recurrent Neural Networks (RNNs)?

Example Applications

Proof. Omitted. □

Lemma 0.1. *Let $\mathcal{C}$ be a set of the construction.*
Let $\mathcal{C}$ be a gerber covering. Let $\mathcal{F}$ be a quasi-coherent sheaves of $\mathcal{O}$-modules. We have to show that

$$\mathcal{O}_{\mathcal{O}_X} = \mathcal{O}_X(\mathcal{L})$$

.

Proof. This is an algebraic space with the composition of sheaves $\mathcal{F}$ on $X_{étale}$ we have

$$\mathcal{O}_X(\mathcal{F}) = \{morph_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

where $\mathcal{G}$ defines an isomorphism $\mathcal{F} \to \mathcal{F}$ of $\mathcal{O}$-modules. □

Lemma 0.2. *This is an integer $\mathcal{Z}$ is injective.*

Proof. See Spaces, Lemma ??. □

Lemma 0.3. *Let S be a scheme. Let X be a scheme and X is an affine open covering. Let $\mathcal{U} \subset \mathcal{X}$ be a canonical and locally of finite type. Let X be a scheme. Let X be a scheme which is equal to the formal complex.*

The following to the construction of the lemma follows.

Let X be a scheme. Let X be a scheme covering. Let

$$b : X \to Y' \to Y \to Y \to Y' \times_X Y \to X.$$

be a morphism of algebraic spaces over S and Y.

Proof. Let X be a nonzero scheme of X. Let X be an algebraic space. Let $\mathcal{F}$ be a quasi-coherent sheaf of $\mathcal{O}_X$-modules. The following are equivalent

(1) $\mathcal{F}$ is an algebraic space over S.
(2) If X is an affine open covering.

Consider a common structure on X and X the functor $\mathcal{O}_X(U)$ which is locally of finite type. □

An RNN learns to create a LaTeX document.

`http://karpathy.github.io/2015/05/21/rnn-effectiveness/`

Unfolded computations of an RNN

Assume that $\theta = \{W^{\text{in}}, W^{\text{rec}}\}$, $\boldsymbol{r}_0 = \mathbf{0}$, and the **forward pass** is constructed as follows:

$$\boldsymbol{r}_1 = W^{\text{in}}\boldsymbol{x}_1 + W^{\text{rec}}\sigma(\boldsymbol{r}_0)$$

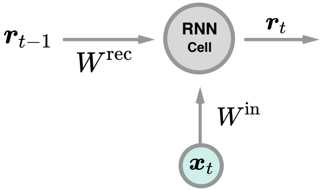
$$\vdots$$

$$\boldsymbol{r}_t = W^{\text{in}}\boldsymbol{x}_t + W^{\text{rec}}\sigma(\boldsymbol{r}_{t-1})$$

$$\vdots$$

$$\boldsymbol{r}_T = W^{\text{in}}\boldsymbol{x}_T + W^{\text{rec}}\sigma(\boldsymbol{r}_{T-1})$$

$$\hat{\boldsymbol{y}} = \boldsymbol{r}_T$$



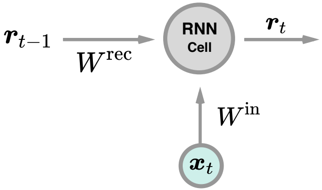
Unfolded computations of an RNN

Backpropagation Through Time can be derivsed by

$$\frac{\partial E}{\partial \theta} = \sum_{k=1}^{T} \frac{\partial E}{\partial \boldsymbol{r}_T} \frac{\partial \boldsymbol{r}_T}{\partial \boldsymbol{r}_k} \frac{\partial^+ \boldsymbol{r}_k}{\partial \theta}$$

$$\frac{\partial \boldsymbol{r}_T}{\partial \boldsymbol{r}_k} = \prod_{T \geq i > k} \frac{\partial \boldsymbol{r}_i}{\partial \boldsymbol{r}_{i-1}}$$

$$= \prod_{T \geq i > k} \underbrace{(W^{\text{rec}})^{T}}_{!} \sigma'(\boldsymbol{r}_{i-1})$$



$\frac{\partial^+ \boldsymbol{r}_{t+1}}{\partial \theta}$ is the “immediate”partial derivative. For example, any row in the matrix $\frac{\partial^+ \boldsymbol{r}_k}{\partial W^{\text{rec}}}$ is $\sigma(\boldsymbol{r}_t)$ [PMB13].

Exploding and Varnishing Gradient Problems

– The exponential term of W^{rec} will dominate the calculation of $\frac{\partial E}{\partial \theta}$.

Exploding and Varnishing Gradient Problems

- The exponential term of W^{rec} will dominate the calculation of $\frac{\partial E}{\partial \theta}$.
- Two problems can happen:
  - Exploding gradient ($\|\nabla_{\boldsymbol{\theta}} E\| \to \infty$)
  - Varnishing gradient ($\|\nabla_{\boldsymbol{\theta}} E\| \to 0$)

Exploding and Varnishing Gradient Problems

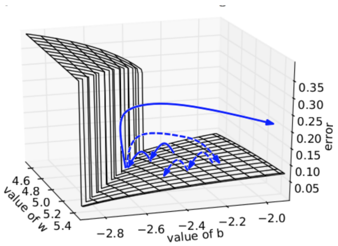
- The exponential term of W^{rec} will dominate the calculation of $\frac{\partial E}{\partial \theta}$.
- Two problems can happen:
  - Exploding gradient ($\|\nabla_{\boldsymbol{\theta}} E\| \rightarrow \infty$)
  - Varnishing gradient ($\|\nabla_{\boldsymbol{\theta}} E\| \rightarrow 0$)
- The situation is determined by **spectral radius** of the matrix $\rho(W^{\text{rec}})$.
  - *Spectral radius* of a matrix is its largest absolute eigenvalue.

Exploding Gradient

- happens when $\rho(W^{\text{rec}}) > 1$.

Exploding Gradient

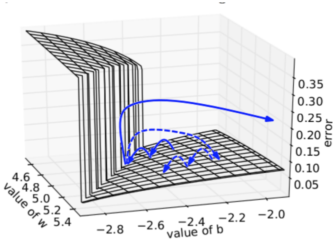
- happens when $\rho(W^{\text{rec}}) > 1$.



Error Function [PMB13].

Exploding Gradient

– happens when $\rho(W^{\text{rec}}) > 1$.



Error Function [PMB13].

– causes training *unstable*.

Exploding Gradient

Exploding gradient can be alleviated by **clipping** the norm of the gradient.

$\epsilon \leftarrow$ *threshold* ;

$\hat{\boldsymbol{g}} \leftarrow \nabla_{\boldsymbol{\theta}} J$;

if $||\hat{\boldsymbol{g}}|| \geq$ *threshold* **then**

$\quad \hat{\boldsymbol{g}} \leftarrow \frac{\epsilon}{||\hat{\boldsymbol{g}}||} \hat{\boldsymbol{g}}$;

end

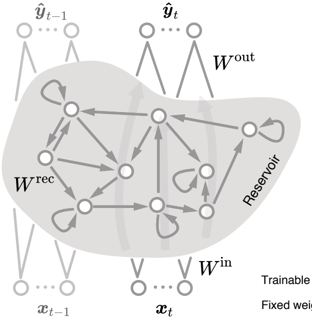
Algorithm 1: Gradient Clipping [PMB13].

Varnishing Gradient

- happens when $\rho(W^{\text{rec}}) < 1$.
- causes RNNs to learn long-term dependencies with a slow progress.
- **Solutions**
  - Echo State Networks
  - Momentum Technique
  - Long Short-Term Memory Networks

Echo State Networks [Jae01]

The idea of ESNs is to create a large random reservoir RNN (W^{rec}, W^{in}). The matrix W^{rec} is carefully constructed such that reservoir can keep signal **echoed** in the network for long time.



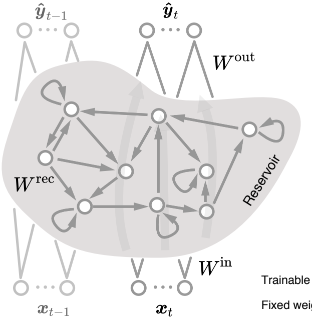
Trainable weights : W^{out}

Fixed weights : $W^{\text{rec}}, W^{\text{in}}$

Echo State Networks [Jae01]

The idea of ESNs is to create a large random reservoir RNN (W^{rec}, W^{in}). The matrix W^{rec} is carefully constructed such that reservoir can keep signal **echoed** in the network for long time.

– $W^{\text{rec}} \leftarrow \frac{\xi}{\rho(W^{\text{rec}})} W^{\text{rec}}$



Trainable weights : W^{out}

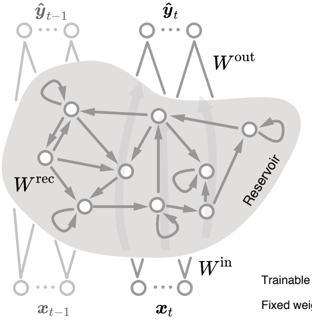
Fixed weights : $W^{\text{rec}}, W^{\text{in}}$

Echo State Networks [Jae01]

The idea of ESNs is to create a large random reservoir RNN (W^{rec}, W^{in}). The matrix W^{rec} is carefully constructed such that reservoir can keep signal **echoed** in the network for long time.

- $W^{rec} \leftarrow \frac{\xi}{\rho(W^{rec})} W^{rec}$

ESNs can be viewed as a kernel trick mapping input to a high-dimensional space and learn a linear classifier (W^{out}) there.



Trainable weights : W^{out}

Fixed weights : W^{rec}, W^{in}

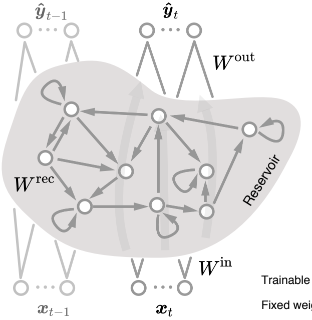
Echo State Networks [Jae01]

Advantages

- Can be trained very fast.
- Work well for low dimensional input.

Disadvantages

- Require experiences to initialize W^{rec} sensibly.
- Require a big reservoir to solve complex problems.



Trainable weights : W^{out}

Fixed weights : $W^{\text{rec}}, W^{\text{in}}$

Gradient Descent with Momentum

– Standard Gradient Descent

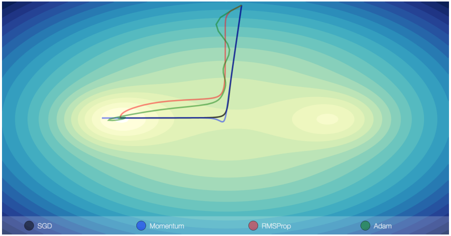
$$w \leftarrow w - \lambda \frac{\partial J}{\partial w}$$

– Adam [KB14]

$$w \leftarrow w - \lambda \frac{\partial J}{\partial w} + someformular$$

– RMSProp [TH]

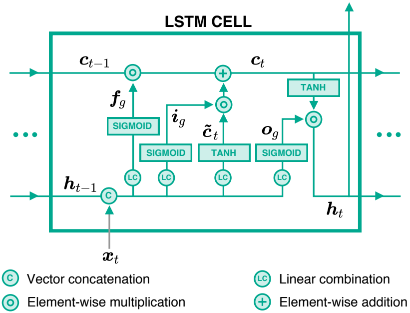
Gradient Descent with Momentum



https://cdn.rawgit.com/EmilienDupont/aaf429be5705b219aaaf8d691e27ca87/raw/d5c6bd2483479c8187f5b49e329d84651b930e09/index.html

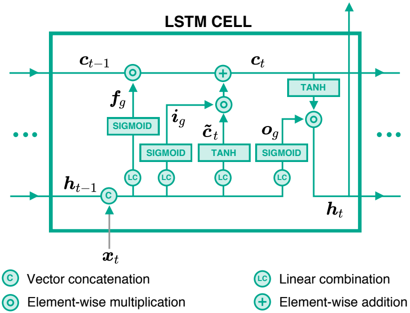
Long Short-Term Memory (LSTM) [HS97]

- Use gating mechanisms and additive updates for $\boldsymbol{c}_t$.
  - Eliminate exponential decay factors in the gradient computation

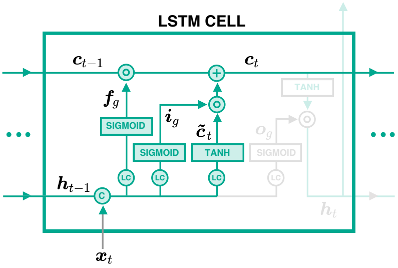


Long Short-Term Memory (LSTM) [HS97]

- Use gating mechanisms and additive updates for $\boldsymbol{c}_t$.
  - Eliminate exponential decay factors in the gradient computation
- Employ 3 gates:
  - Input gate $\boldsymbol{i}_g$
  - Forget gate $\boldsymbol{f}_g$
  - Output gate $\boldsymbol{o}_g$



Computations in an LSTM cell



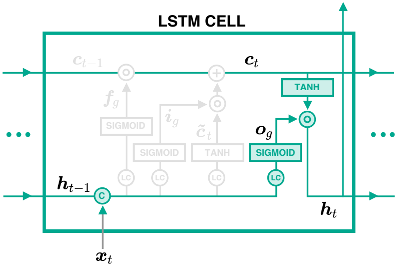
$$\boldsymbol{f}_g = \mathrm{sigm}(U^{(f)}\boldsymbol{x}_t + V^{(f)}\boldsymbol{h}_{t-1})$$

$$\boldsymbol{i}_g = \mathrm{sigm}(U^{(i)}\boldsymbol{x}_t + V^{(i)}\boldsymbol{h}_{t-1})$$

$$\tilde{\boldsymbol{c}}_t = \tanh(U^{(c)}\boldsymbol{x}_t + V^{(c)}\boldsymbol{h}_{t-1})$$

$$\boldsymbol{c}_t = \boldsymbol{f}_g \circ \boldsymbol{c}_{t-1} + \boldsymbol{i}_g \circ \tilde{\boldsymbol{c}}_t$$

Computations in an LSTM cell



$$\boldsymbol{o}_g = \mathrm{sigm}(U^{(o)}\boldsymbol{x}_t + V^{(o)}\boldsymbol{h}_{t-1})$$

$$\boldsymbol{h}_t = \boldsymbol{o}_g \circ \tanh(\boldsymbol{c}_t)$$

References I

[HS97] Sepp Hochreiter and Jürgen Schmidhuber, **Long short-term memory**, Neural Comput. **9** (1997), no. 9, 1735–1780.

[Jae01] Herbert Jaeger, **The "echo state" approach to analysing and training recurrent neural networks-with an erratum note'**.

[KB14] Diederik P. Kingma and Jimmy Ba, **Adam: A method for stochastic optimization**, CoRR **abs/1412.6980** (2014).

[PMB13] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio, **On the difficulty of training recurrent neural networks**, Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013, 2013, pp. 1310–1318.

[TH] T. Tieleman and G. Hinton, **RMSprop Gradient Optimization**.