Deep Learning

6. Recurrent Neural Networks (RNNs)

Lars Schmidt-Thieme

Information Systems and Machine Learning Lab (ISMLL)
Institute for Computer Science
University of Hildesheim, Germany

Syllabus

Tue. 21.4.	(1)	1. Supervised Learning (Review 1)
Tue. 28.4.	(2)	2. Neural Networks (Review 2)
Tue. 5.5.	(3)	3. Regularization for Deep Learning
Tue. 12.5.	(4)	4. Optimization for Training Deep Models
Tue. 19.5.	(5)	5. Convolutional Neural Networks
Tue. 26.5.	(6)	6. Recurrent Neural Networks
Tue. 2.6.	—	— *Pentecoste Break* —
Tue. 9.6.	(7)	7. Autoencoders
Tue. 16.6.	(8)	8. Generative Adversarial Networks
Tue. 23.6.	(9)	9. Recent Advances
Tue. 30.6.	(10)	10. Engineering Deep Learning Models
Tue. 7.7.	(11)	tbd.
Tue. 14.7.	(12)	Q & A

Outline

1. Sequence Data and Problems
2. Recurrent Neural Networks
3. Back Propagation Through Time
4. Gated Units and Long Short-Term Memory (LSTM)
5. Time Series Classification and Forecasting

Outline

Sequences

- let X be any set, often called **alphabet**
- **sequences in X**:

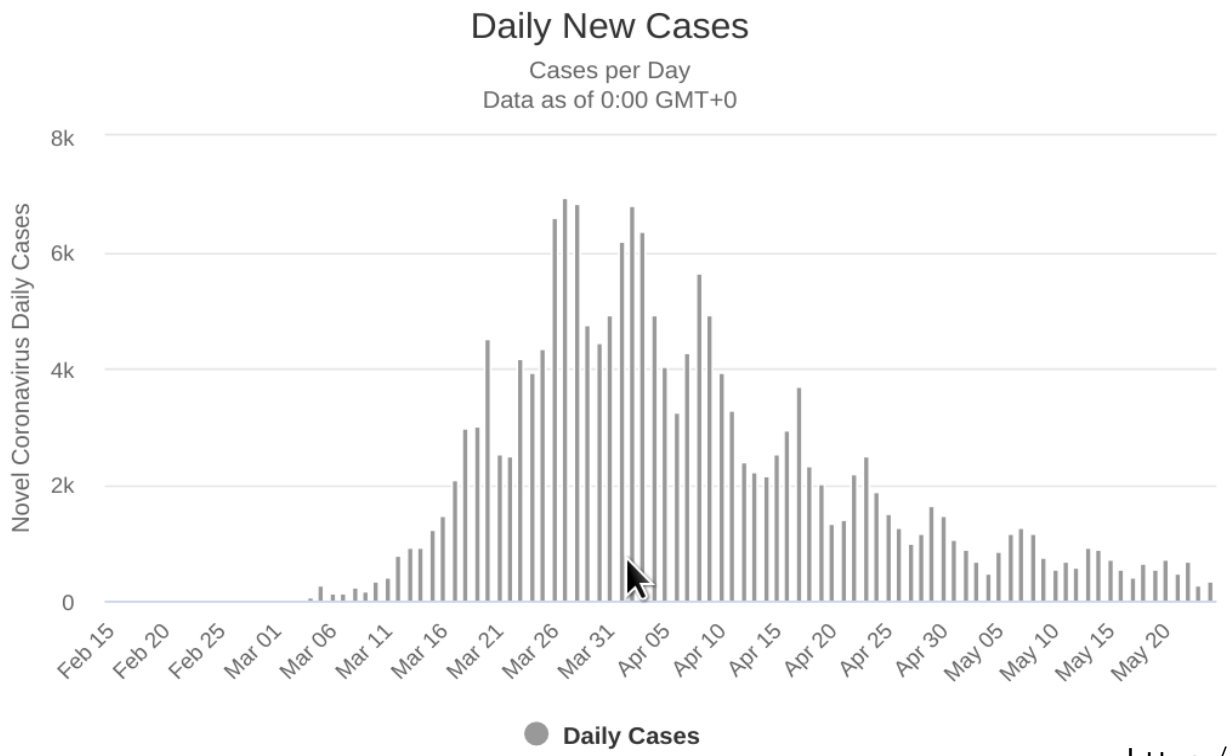
$$X^* := \bigcup_{t \in \mathbb{N}} X^t$$

  - aka **time series**
  - e.g., $(\text{"h"}, \text{"e"}, \text{"l"}, \text{"l"}, \text{"o"}) \in X^*$ for alphabet $X := \{\text{"a"}, \text{"b"}, \ldots, \text{"z"}\}$: strings.

- $|x| := t$ if $x \in X^t$ **length** of sequence $x \in X^*$

Sequences / Example

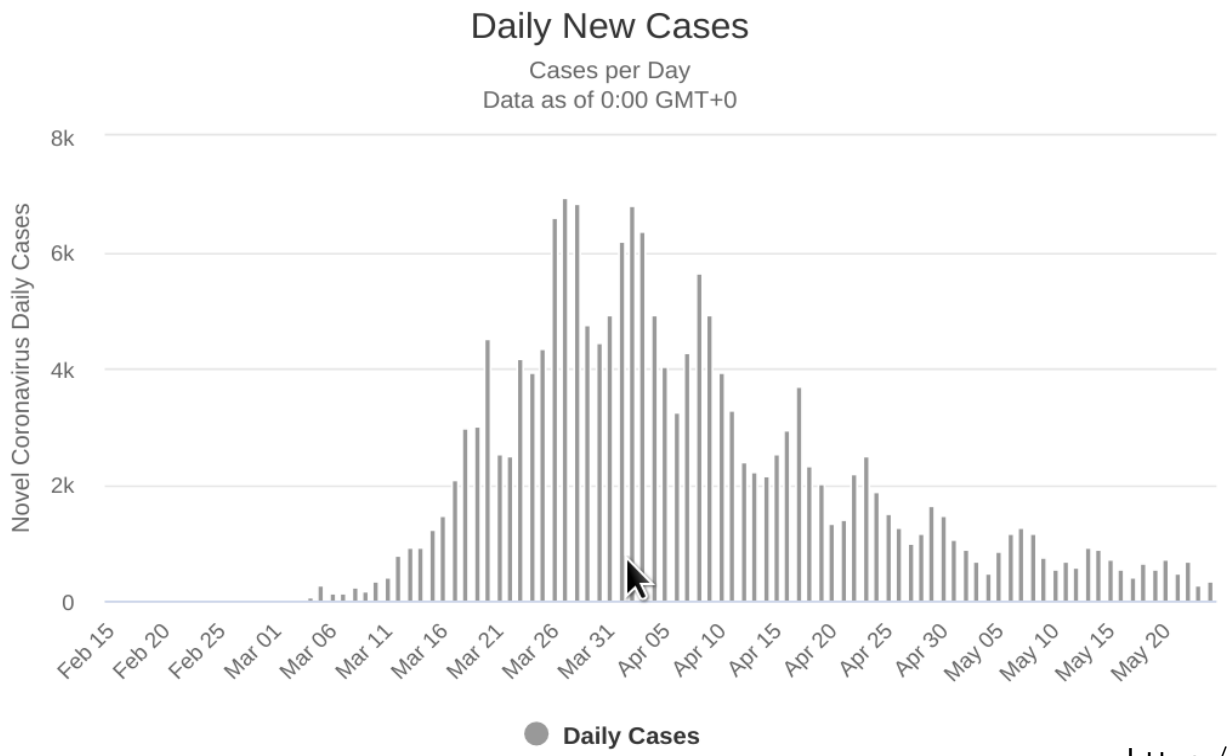
- new corona infections in Germany
- alphabet: — Q: what is the alphabet?
- example: $(490, 692, 273, 342) \in X^*$



https://www.worldometers.info/c

Sequences / Example

- new corona infections in Germany
- alphabet: $X := \mathbb{N}$
- example: $(490, 692, 273, 342) \in X^*$



https://www.worldometers.info/c

Sequences / Example

- German stock index DAX
- alphabet: $X := \mathbb{R}_0^+$
- example: $(11223.71, 11065.93, 11073.87, 11259.11) \in X^*$

https://www.google.com/search?

Sequences of Vectors

- $X := \mathbb{R}^M$, M called **channels**
- e.g., EEG data: measurement at $M := 16$ electrodes.

[source: https://en.wikipedia.org/wiki/Electroencephalography]

Sequences of Vectors / More Examples

- weather data: temperature, precipitation, wind speed, pressure, humidity etc.
- infections: new infections, new recoveries, new deaths
- infections: in Lower Saxony, Hamburg, Bremen, Berlin, Hessen etc.
- texts: 60.000 binary word indicators

⋮

Sequence Problems / 1. Sequence Prediction

- supervised problems with pairs
  $\mathcal{D}^{\text{train}} := \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\} \subseteq \mathcal{X} \times \mathcal{Y}$

1. **sequence / time series classification / regression / prediction**:
   - input: $\mathcal{X} := (\mathbb{R}^M)^*$ sequences / time series
     - usually with multiple channels
   - output: $\mathcal{Y} := \mathbb{R}^O$ (regression) or $\mathcal{Y} := \{0, 1\}^O$ (classification)
   - example:
     - predict thunderstorm from weather data
     - predict insolvency from stock quotes
     - predict health conditions from ECG (or other medical time-variant data)
   - special case: **time series forecasting**: $\mathcal{Y} := \mathbb{R}^M$

Sequence Problems / 2. Sequence-to-Sequence Learning

2. **sequence-to-sequence learning**:
   - input: $\mathcal{X} := (\mathbb{R}^M)^*$ sequences / time series
     - usually with multiple channels
   - output: $\mathcal{Y} := (\mathbb{R}^O)^*$ or $\mathcal{Y} := (\{0,1\}^O)^*$ sequence / time series
     - the output sequence is **not aligned** to the input sequence, esp. the output sequence can have a different length as the input sequence.
   - loss: average elementwise loss:

$$\ell(y, \hat{y}) := \frac{1}{|y|} \sum_{t=1}^{|y|} \ell(y_t, \hat{y}_t), \quad \text{e.g.,} = -\frac{1}{|y|} \sum_{t=1}^{|y|} \sum_{o=1}^{O} y_{t,o} \log(\hat{y}_{t,o})$$

   - example: translation:

input x:	Machen wir es kurz.
output y:	Let's make it short
input x:	Auf Wiedersehen!
output y:	Goodbye!

Sequence Problems / 3. Sequence Labeling

3. **sequence labeling**:
   - input: $\mathcal{X} := (\mathbb{R}^M)^*$ sequences / time series
     - usually with multiple channels
   - output: $\mathcal{Y} := (\mathbb{R}^O)^*$ or $\mathcal{Y} := (\{0,1\}^O)^*$
     - each output representing further measurements for each input index (**aligned sequences**),
       esp. each output having the same length as the input,
       i.e., $\mathcal{D}^{\text{train}} \in ((\mathbb{R}^M \times \mathbb{R}^O)^*)^* = ((\mathbb{R}^{M+O})^*)^*$
   - loss: average elementwise loss.
   - example: part of speech tagging:

input x:	The	quick	brown	jumps	over	the	lazy	dog.
output y:	DET	ADJ	NOUN	VERB	ADP	DET	ADJ	NOUN

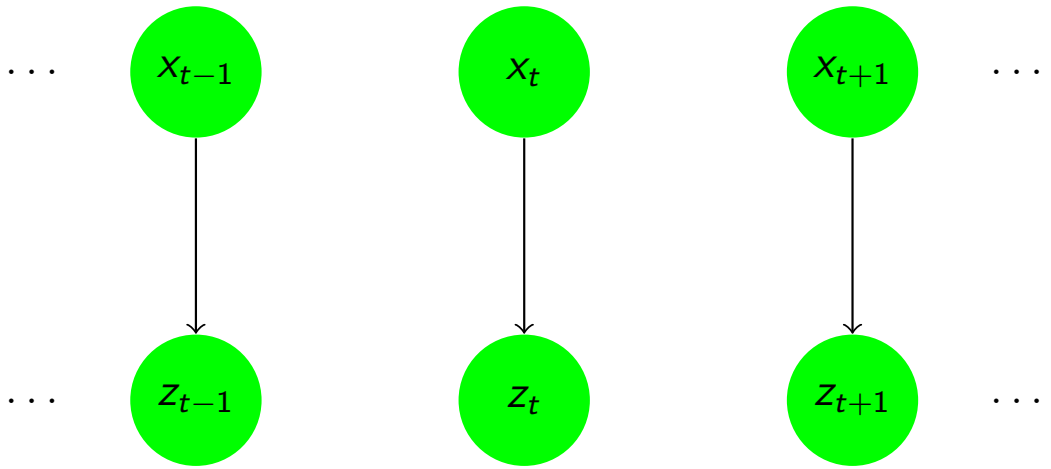
Outline

From Convolutional Layers to Recurrent Layers

- inputs $x \in (\mathbb{R}^M)^*$
- convolutions with kernel size 1 (kernels $1 \times M$):

$$z_t := g(x_t) = a(Wx_t + b), \quad t := 1, \ldots, |x|$$

$$W \in \mathbb{R}^{K \times M}, \quad b \in \mathbb{R}^K, \quad a : \mathbb{R} \to \mathbb{R}$$



From Convolutional Layers to Recurrent Layers

- inputs $x \in (\mathbb{R}^M)^*$
- convolutions with kernel size 1 (kernels $1 \times M$):

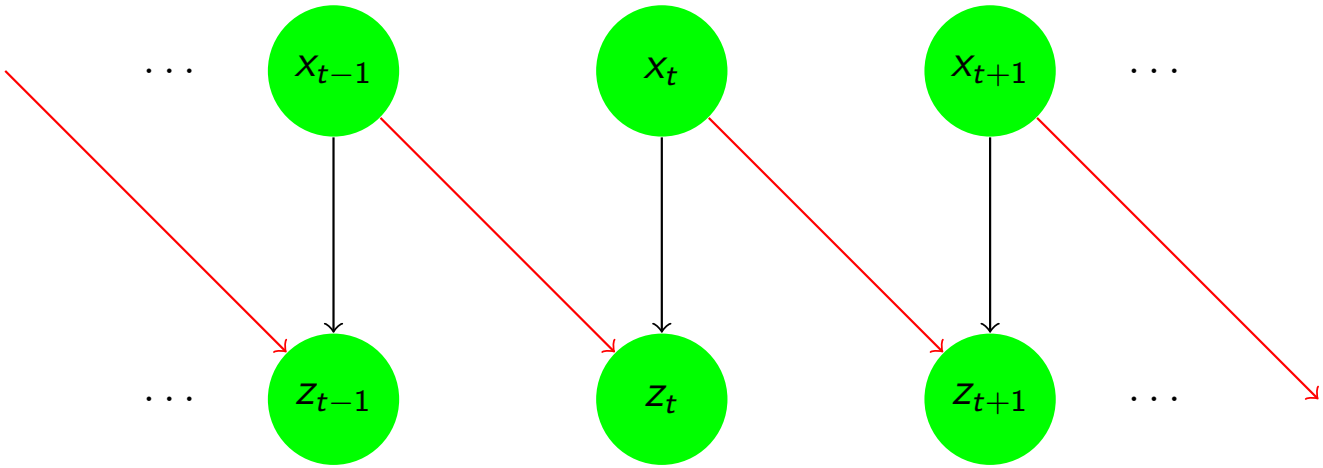
$$z_t := g(x_t) = a(Wx_t + b), \quad t := 1, \ldots, |x|$$

$$W \in \mathbb{R}^{K \times M}, \quad b \in \mathbb{R}^K, \quad a : \mathbb{R} \to \mathbb{R}$$

- convolutions with kernel size 2 (kernels $2 \times M$):

$$z_t := g(x_t, x_{t-1}) = a(W_{.,2,.}x_t + W_{.,1,.}x_{t-1} + b), \quad t := 1, \ldots, |x|$$

$$W \in \mathbb{R}^{K \times 2 \times M}, \quad b \in \mathbb{R}^K, \quad a : \mathbb{R} \to \mathbb{R}$$



From Convolutional Layers to Recurrent Layers

- inputs $x \in (\mathbb{R}^M)^*$
- convolutions with kernel size 1 (kernels $1 \times M$):

$$z_t := g(x_t) = a(Wx_t + b), \quad t := 1, \ldots, |x|$$
$$W \in \mathbb{R}^{K \times M}, \quad b \in \mathbb{R}^K, \quad a : \mathbb{R} \to \mathbb{R}$$

- convolutions with kernel size 2 (kernels $2 \times M$):

$$z_t := g(x_t, x_{t-1}) = a(W_{.,2,.}x_t + W_{.,1,.}x_{t-1} + b), \quad t := 1, \ldots, |x|$$
$$W \in \mathbb{R}^{K \times 2 \times M}, \quad b \in \mathbb{R}^K, \quad a : \mathbb{R} \to \mathbb{R}$$
$$= g(x_t, x_{t-1}) = a(Wx_t + Vx_{t-1} + b), \quad t := 1, \ldots, |x|$$
$$W \in \mathbb{R}^{K \times M}, \quad V \in \mathbb{R}^{K \times M}, \quad b \in \mathbb{R}^K, \quad a : \mathbb{R} \to \mathbb{R}$$

From Convolutional Layers to Recurrent Layers

- inputs $x \in (\mathbb{R}^M)^*$
- convolutions with kernel size 1 (kernels $1 \times M$):

$$z_t := g(x_t) = a(Wx_t + b), \quad t := 1, \ldots, |x|$$
$$W \in \mathbb{R}^{K \times M}, \quad b \in \mathbb{R}^K, \quad a : \mathbb{R} \to \mathbb{R}$$

- convolutions with kernel size 2 (kernels $2 \times M$):

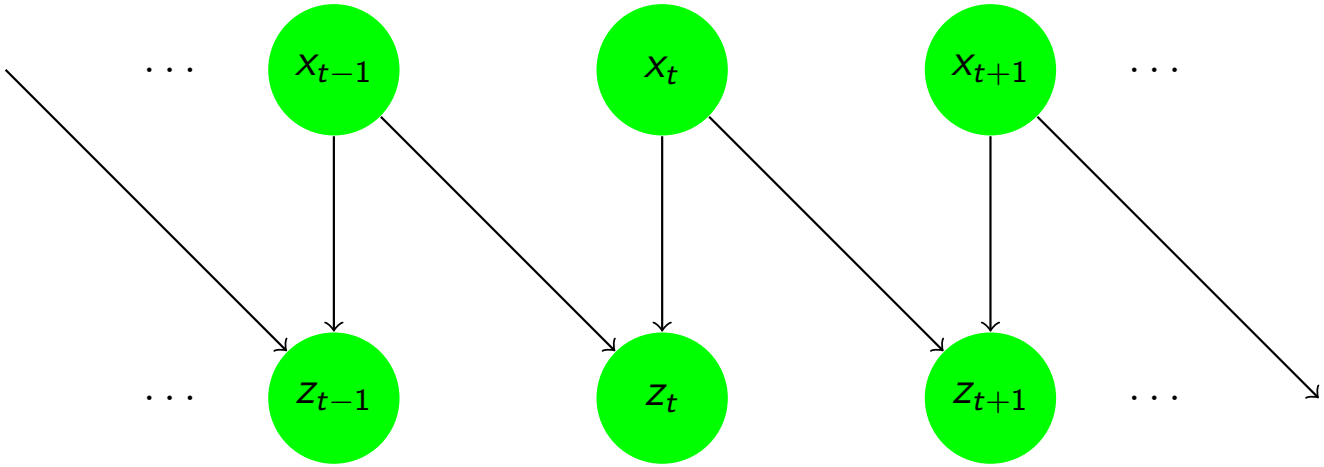
$$z_t = g(x_t, x_{t-1}) = a(Wx_t + Vx_{t-1} + b), \quad t := 1, \ldots, |x|$$
$$W \in \mathbb{R}^{K \times M}, \quad V \in \mathbb{R}^{K \times M}, \quad b \in \mathbb{R}^K, \quad a : \mathbb{R} \to \mathbb{R}$$

- recurrent layer:

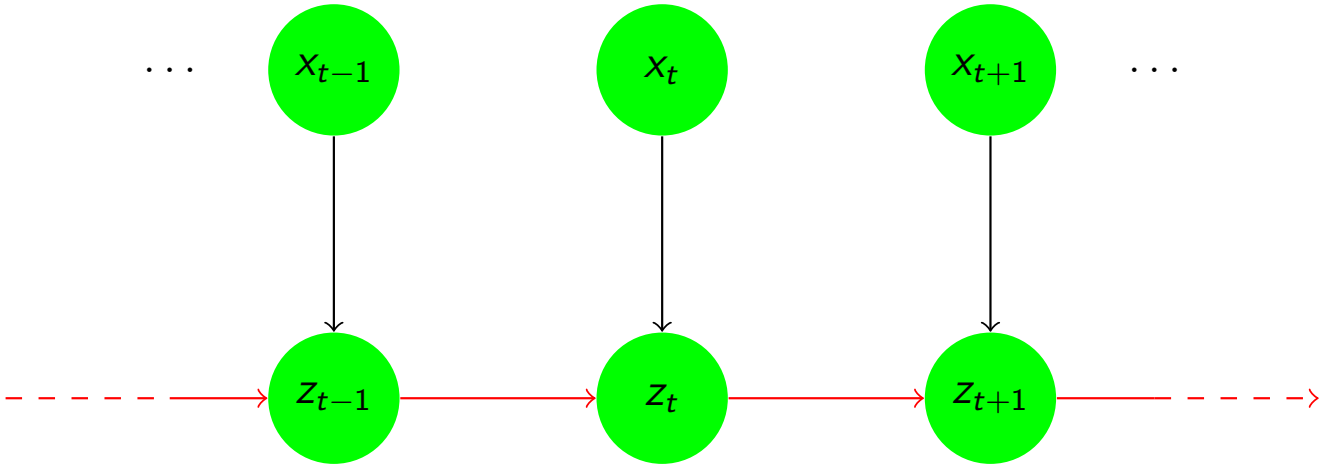
$$z_t := g(x_t, z_{t-1}) = a(Wx_t + Vz_{t-1} + b), \quad t := 1, \ldots, |x|$$
$$W \in \mathbb{R}^{K \times M}, \quad V \in \mathbb{R}^{K \times K}, \quad b \in \mathbb{R}^K, \quad a : \mathbb{R} \to \mathbb{R}, \quad z_0 \in \mathbb{R}^K$$

From Convolutional Layers to Recurrent Layers

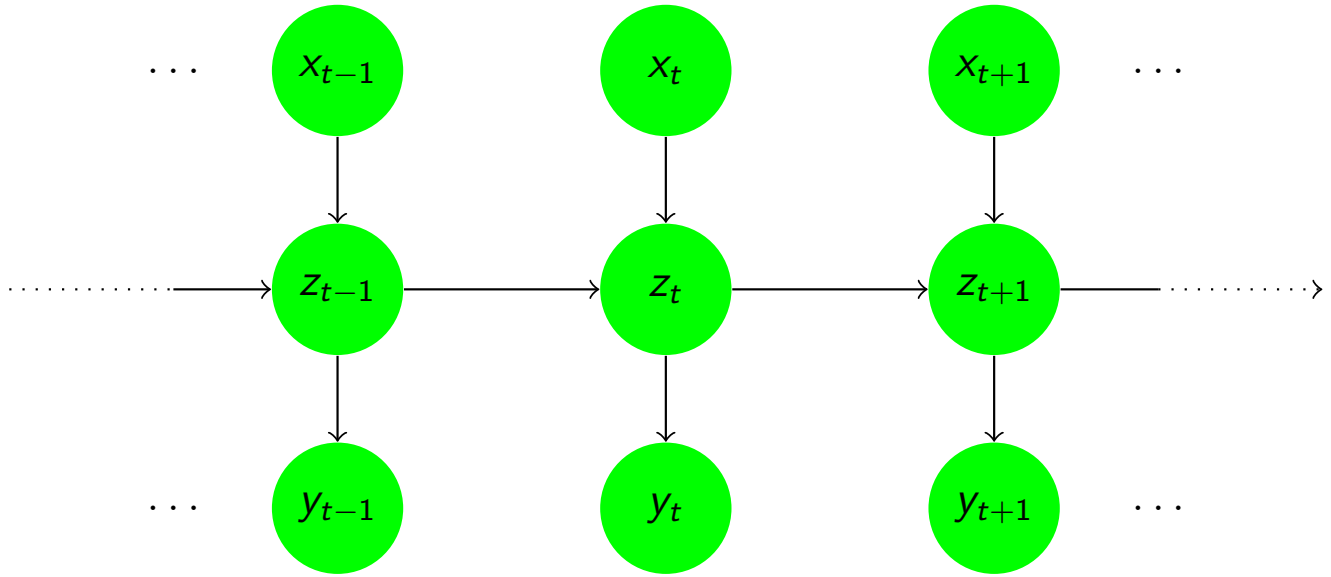
- convolutional layer with kernel size 2:



- recurrent layer:



RNN with one Hidden Layer



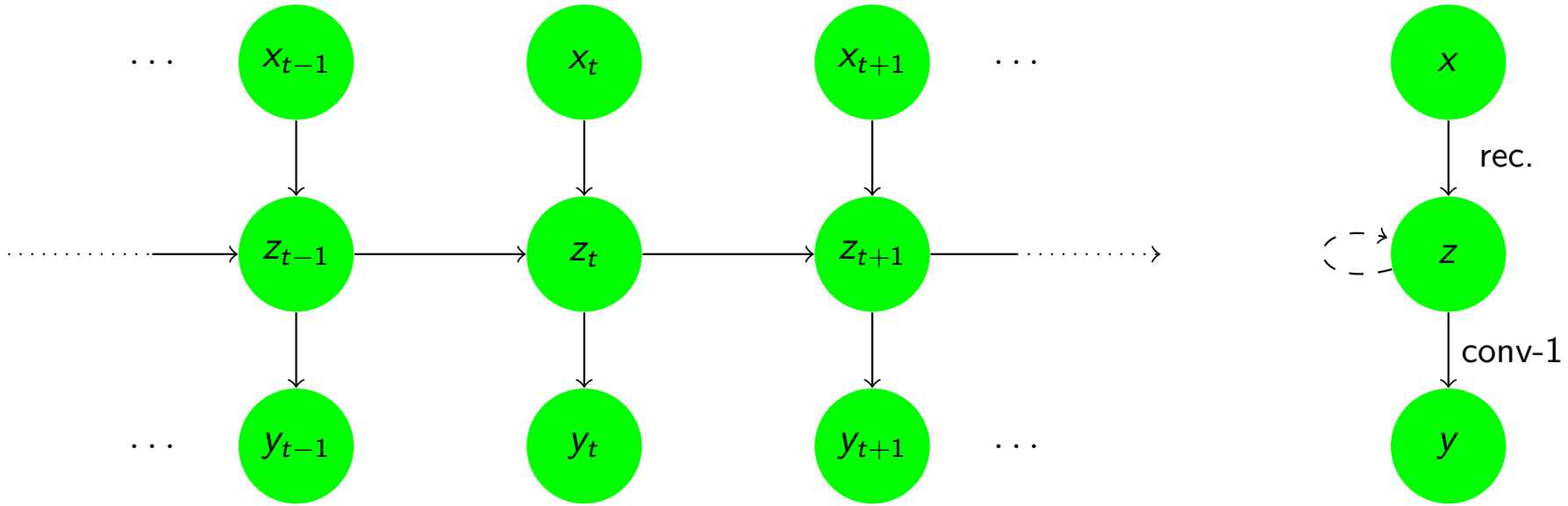
$$z_t := g(x_t, z_{t-1}) := a(W^1 x_t + V^1 z_{t-1} + b^1), \quad t := 1, \ldots, |x|$$

$$W^1 \in \mathbb{R}^{K \times M}, \quad V^1 \in \mathbb{R}^{K \times K}, \quad b^1 \in \mathbb{R}^K, \quad a : \mathbb{R} \to \mathbb{R}, \quad z_0 \in \mathbb{R}^K$$

$$y_t := h(z_t) := a_2(W^2 z_t + b^2), \quad t := 1, \ldots, |x|$$

$$W^2 \in \mathbb{R}^{O \times K}, \quad b^2 \in \mathbb{R}^O, \quad a_2 : \mathbb{R} \to \mathbb{R} \text{ (or } \mathbb{R}^O \to \mathbb{R}^O \text{, e.g., softmax)}$$

RNN with one Hidden Layer



$$z_t := g(x_t, z_{t-1}) := a(W^1 x_t + V^1 z_{t-1} + b^1), \quad t := 1, \ldots, |x|$$

$$W^1 \in \mathbb{R}^{K \times M}, \quad V^1 \in \mathbb{R}^{K \times K}, \quad b^1 \in \mathbb{R}^K, \quad a: \mathbb{R} \to \mathbb{R}, \quad z_0 \in \mathbb{R}^K$$

$$y_t := h(z_t) := a_2(W^2 z_t + b^2), \quad t := 1, \ldots, |x|$$

$$W^2 \in \mathbb{R}^{O \times K}, \quad b^2 \in \mathbb{R}^O, \quad a_2: \mathbb{R} \to \mathbb{R} \text{ (or } \mathbb{R}^O \to \mathbb{R}^O \text{, e.g., softmax)}$$

Q: Is a RNN still a computational acyclic graph?

Note: dashed line: temporal connection.

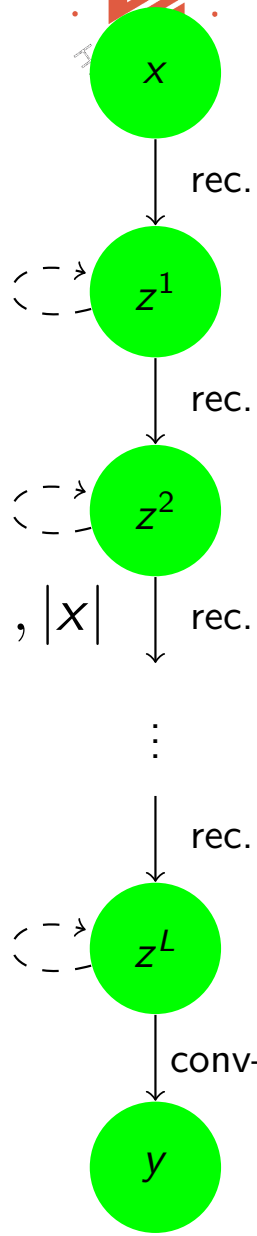
RNN with L Hidden Layers

- RNN with L hidden layers:

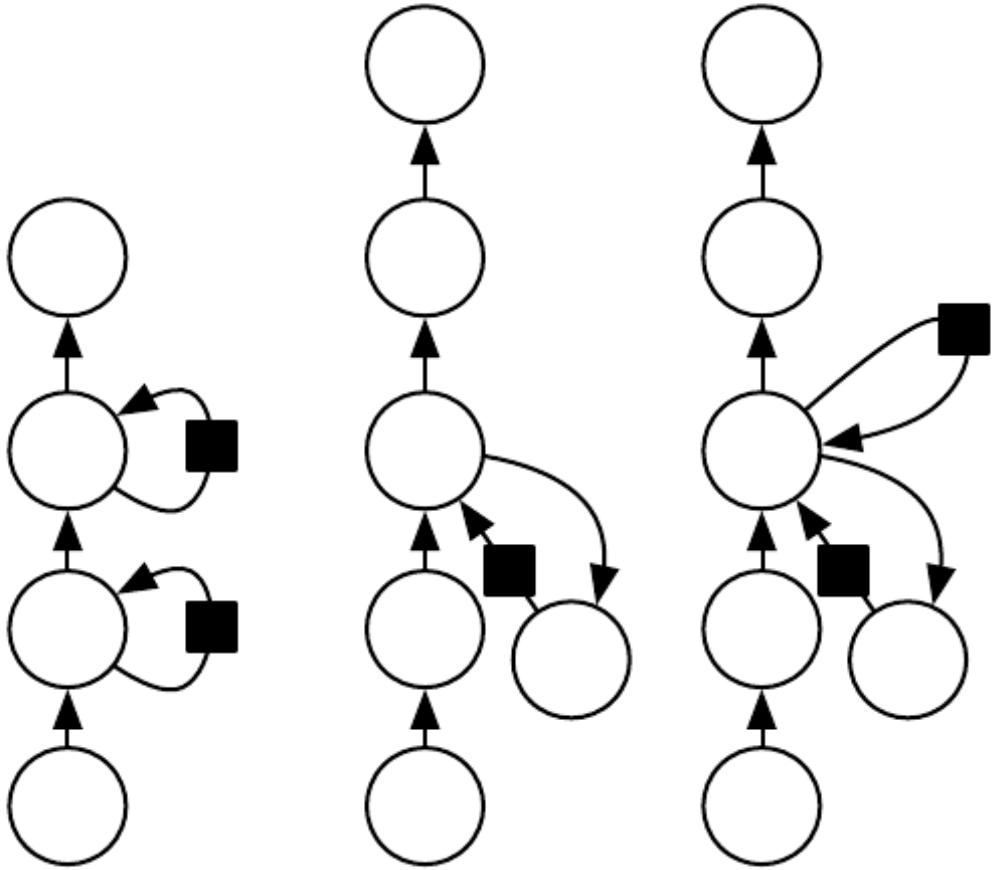
$$z_t^\ell := a(W^\ell z_t^{\ell-1} + V^\ell z_{t-1}^\ell + b^\ell), \quad \ell := 1, \ldots, L+1; t := 1, \ldots, |x|$$

$$W^\ell \in \mathbb{R}^{M_\ell \times M_{\ell-1}}, \quad V^\ell \in \mathbb{R}^{M_\ell \times M_\ell}, \quad b^\ell \in \mathbb{R}^{M_\ell}, \quad a: \mathbb{R} \to \mathbb{R}, \quad z_0^\ell \in \mathbb{R}^{M_\ell}$$

$$z^0 := x, \quad M_0 := M, \quad z^{L+1} =: y, \quad M_{L+1} := O, \quad V_{L+1} := 0$$



Variants of Deep RNN Architectures



[source: Goodfellow et al. 2016, p. 400]

Outline

Gradients w.r.t. Latent Features

- recurrent layer:

$$u_t^\ell := W^\ell z_t^{\ell-1} + V^\ell z_{t-1}^\ell + b^\ell, \quad \ell{:=}1,\ldots,L{+}1; t{:=}1,\ldots,|x|$$
$$z_t^\ell := a(u_t^\ell)$$
$$W^\ell \in \mathbb{R}^{M_\ell \times M_{\ell-1}}, \quad V^\ell \in \mathbb{R}^{M_\ell \times M_\ell}, \quad b^\ell \in \mathbb{R}^{M_\ell}, \quad a{:}\mathbb{R}\to\mathbb{R}, \quad z_0^\ell \in \mathbb{R}^{M_\ell}$$
$$z^0{:=}x, \quad M_0{:=}M, \quad z^{L+1}{=:}y, \quad M_{L+1}{:=}O, \quad V_{L+1}{:=}0$$

- objective:

$$f(x,y;\theta) := \ell(y,\hat{y}(x;\theta)) + \Omega(\theta), \quad \theta := (W^\ell, V^\ell, b^\ell, z_0^\ell)_{\ell=1:L+1}$$

- gradients:

$$\frac{\partial f(x,y;\theta)}{\partial \theta} = \frac{\partial \ell(y,\hat{y}(x;\theta))}{\partial \hat{y}} \frac{\partial \hat{y}(x;\theta)}{\partial \theta} + \frac{\partial \Omega(\theta)}{\partial \theta}$$

Gradients w.r.t. Latent Features

- recurrent layer:

$$u_t^\ell := W^\ell z_t^{\ell-1} + V^\ell z_{t-1}^\ell + b^\ell, \quad \ell:=1,\ldots,L+1; t:=1,\ldots,|x|$$

$$z_t^\ell := a(u_t^\ell)$$

$$W^\ell \in \mathbb{R}^{M_\ell \times M_{\ell-1}}, \quad V^\ell \in \mathbb{R}^{M_\ell \times M_\ell}, \quad b^\ell \in \mathbb{R}^{M_\ell}, \quad a:\mathbb{R}\to\mathbb{R}, \quad z_0^\ell \in \mathbb{R}^{M_\ell}$$

$$z^0:=x, \quad M_0:=M, \quad z^{L+1}=:y, \quad M_{L+1}:=O, \quad V_{L+1}:=0$$

- gradients w.r.t. latent features:

$$\frac{\partial \hat{y}}{\partial z_t^\ell} = \frac{\partial \hat{y}}{\partial u_t^{\ell+1}} \frac{\partial u_t^{\ell+1}}{\partial z_t^\ell} + \frac{\partial \hat{y}}{\partial u_{t+1}^\ell} \frac{\partial u_{t+1}^\ell}{\partial z_t^\ell} = \frac{\partial \hat{y}}{\partial u_t^{\ell+1}} W^{\ell+1} + \frac{\partial \hat{y}}{\partial u_{t+1}^\ell} V^\ell$$

$$\frac{\partial \hat{y}}{\partial u_t^\ell} = \frac{\partial \hat{y}}{\partial z_t^\ell} \frac{\partial z_t^\ell}{\partial u_t^\ell} = \frac{\partial \hat{y}}{\partial z_t^\ell} \operatorname{diag}(a'(u_t^\ell))$$

$$= (\underbrace{\frac{\partial \hat{y}}{\partial u_t^{\ell+1}}}_{\text{bp layers}} W^{\ell+1} + \underbrace{\frac{\partial \hat{y}}{\partial u_{t+1}^\ell}}_{\text{bp time}} V^\ell) \operatorname{diag}(a'(u_t^\ell))$$

Gradients w.r.t. Parameters

- recurrent layer:

$$u_t^\ell := W^\ell z_t^{\ell-1} + V^\ell z_{t-1}^\ell + b^\ell, \quad \ell:=1,\ldots,L+1; t:=1,\ldots,|x|$$
$$z_t^\ell := a(u_t^\ell)$$

- gradients w.r.t. parameters:

$$\frac{\partial \hat{y}}{\partial b^\ell} = \frac{\partial z^{L+1}}{\partial b^\ell}$$

Gradients w.r.t. Parameters

- recurrent layer:

$$u_t^\ell := W^\ell z_t^{\ell-1} + V^\ell z_{t-1}^\ell + b^\ell, \quad \ell:=1,\ldots,L+1; t:=1,\ldots,|x|$$
$$z_t^\ell := a(u_t^\ell)$$

- gradients w.r.t. parameters:

$$\frac{\partial \hat{y}}{\partial b^\ell} = \frac{\partial z^{L+1}}{\partial b^\ell} \stackrel{?}{=} \begin{cases} \text{A.} & \sum_{t=1}^{|x|} \frac{\partial \hat{y}}{\partial u_t^\ell} \frac{\partial u_t^\ell}{\partial b^\ell} \\ \text{B.} & \sum_{t=1}^{|x|} \frac{\partial \hat{y}}{\partial u_t^\ell} \\ \text{C.} & \frac{\partial \hat{y}}{\partial u_t^\ell} \frac{\partial u_t^\ell}{\partial b^\ell} \\ \text{D.} & 0 \end{cases}$$

Gradients w.r.t. Parameters

- recurrent layer:

$$u_t^\ell := W^\ell z_t^{\ell-1} + V^\ell z_{t-1}^\ell + b^\ell, \quad \ell:=1,\ldots,L+1; t:=1,\ldots,|x|$$
$$z_t^\ell := a(u_t^\ell)$$

- gradients w.r.t. parameters:

$$\frac{\partial \hat{y}}{\partial b^\ell} = \frac{\partial z^{L+1}}{\partial b^\ell} = \sum_{t=1}^{|x|} \frac{\partial \hat{y}}{\partial u_t^\ell} \frac{\partial u_t^\ell}{\partial b^\ell} = \sum_{t=1}^{|x|} \frac{\partial \hat{y}}{\partial u_t^\ell}$$

$$\frac{\partial \hat{y}}{\partial W_{.,m}^\ell} = \sum_{t=1}^{|x|} \frac{\partial \hat{y}}{\partial u_t^\ell} \frac{\partial u_t^\ell}{\partial W_{.,m}^\ell} = \sum_{t=1}^{|x|} \frac{\partial \hat{y}}{\partial u_t^\ell} z_{t,m}^{\ell-1}$$

$$\frac{\partial \hat{y}}{\partial V_{.,m}^\ell} = \sum_{t=1}^{|x|} \frac{\partial \hat{y}}{\partial u_t^\ell} \frac{\partial u_t^\ell}{\partial V_{.,m}^\ell} = \sum_{t=1}^{|x|} \frac{\partial \hat{y}}{\partial u_t^\ell} z_{t-1,m}^{\ell}$$

$$\frac{\partial \hat{y}}{\partial z_0^\ell} = \text{already computed above}$$

RNN: Prediction / Forward Computation

predict-rnn$((W^\ell, V^\ell, b^\ell, z_0^\ell)_{\ell=1:L+1}, a, x, \text{mode})$:

$z^0 := x$

for $\ell := 1 : L+1$:

  for $t := 1 : |x|$:

    $u_t^\ell := W^\ell z_t^{\ell-1} + V^\ell z_{t-1}^\ell + b^\ell$

    $z_t^\ell := a(u_t^\ell)$

if mode="prediction":

  return $(z_t^{L+1})_{t=1:|x|}$

else :

  return $(u_t^\ell, z_t^\ell)_{\ell=0:L+1;t=1:|x|}$

Backpropagation Through Time (BPTT)

gradients-rnn$((W^\ell, V^\ell, b^\ell, z_0^\ell)_{\ell=1:L+1}, a, x, y, f)$:

  $(u_t^\ell, z_t^\ell)_{\ell=0:L+1;t=1:|x|}$:= predict-rnn$((W^\ell, V^\ell, b^\ell, z_0^\ell)_{\ell=1:L+1}, a, x, \text{"all"})$

  for $\ell := L+1 : 1$ backwards:

    $\frac{\partial f}{\partial W^\ell} := 0, \frac{\partial f}{\partial V^\ell} := 0, \frac{\partial f}{\partial b^\ell} := 0; \frac{\partial f}{\partial u^\ell_{|x|+1}} := 0$

    for $t := |x| : 1$ backwards:

      if $\ell = L+1$:

        $\frac{\partial f}{\partial u_t^{L+1}} := \frac{\partial f}{\partial \hat{y}}(y, z^{L+1}) \operatorname{diag}(a'(u_t^{L+1}))$

      else :

        $\frac{\partial f}{\partial u_t^\ell} := (\frac{\partial f}{\partial u_t^{\ell+1}} W^{\ell+1} + \frac{\partial f}{\partial u_{t+1}^\ell} V^\ell) \operatorname{diag}(a'(u_t^\ell))$

      $\frac{\partial f}{\partial W^\ell} \mathrel{+}= \frac{\partial f}{\partial u_t^\ell} \cdot (z_t^{\ell-1})^T$

      $\frac{\partial f}{\partial V^\ell} \mathrel{+}= \frac{\partial f}{\partial u_t^\ell} \cdot (z_{t-1}^\ell)^T$

      $\frac{\partial f}{\partial b^\ell} \mathrel{+}= \frac{\partial f}{\partial u_t^\ell}$

    $\frac{\partial f}{\partial z_0^\ell} := \frac{\partial f}{\partial u_1^\ell} V^\ell$

  return $(\frac{\partial f}{\partial W^\ell}, \frac{\partial f}{\partial V^\ell}, \frac{\partial f}{\partial b^\ell}, \frac{\partial f}{\partial z_0^\ell})_{\ell=1:L+1}$

Backpropagation Through Time (BPTT)

1. compute parameter gradients:

$$(\frac{\partial f}{\partial W^\ell}, \frac{\partial f}{\partial V^\ell}, \frac{\partial f}{\partial b^\ell}, \frac{\partial f}{\partial z_0^\ell})_{\ell=1:L+1} := \text{gradients-rnn}(\ldots)$$

2. update parameters:

$$W^\ell := W^\ell - \mu \frac{\partial f}{\partial W^\ell}$$
$$V^\ell := V^\ell - \mu \frac{\partial f}{\partial V^\ell}$$
$$b^\ell := b^\ell - \mu \frac{\partial f}{\partial b^\ell}$$
$$z_0^\ell := z_0^\ell - \mu \frac{\partial f}{\partial z_0^\ell}, \quad \ell = 1 : L+1$$

Backpropagation Through Time (BPTT)

- mind the details:
  - compute gradients for minibatches
  - use weight decay / L2 regularization .
  - possibly use momentum
  - use step length controller for μ, e.g., Adam.

Outline

Long-term Dependencies

- assume no activation functions ($a(u) := u$)
- z_t as a function of x_1:

$$z_t(x_1) \propto V z_{t-1} \propto V^2 z_{t-2} \propto \cdots \propto V^{t-1} z_1 \propto V^{t-1} W x_1$$

- for $V = Q \Lambda Q^T$ with
  orthogonal matrix Q and
  diagonal matrix $\Lambda := \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_K)$ the eigenvalues:

$$z_t(x_1) \propto Q \Lambda Q^T Q \Lambda Q^T \cdots Q \Lambda Q^T W x_1 = Q \Lambda^{t-1} Q^T W x_1$$

- dimensions with eigenvalue $\lambda_k < 1$: $\lambda_k^{t-1} \to 0$ will vanish
- dimensions with eigenvalue $\lambda_k > 1$: $\lambda_k^{t-1} \to \infty$ will explode

Illustrating Vanishing Gradients

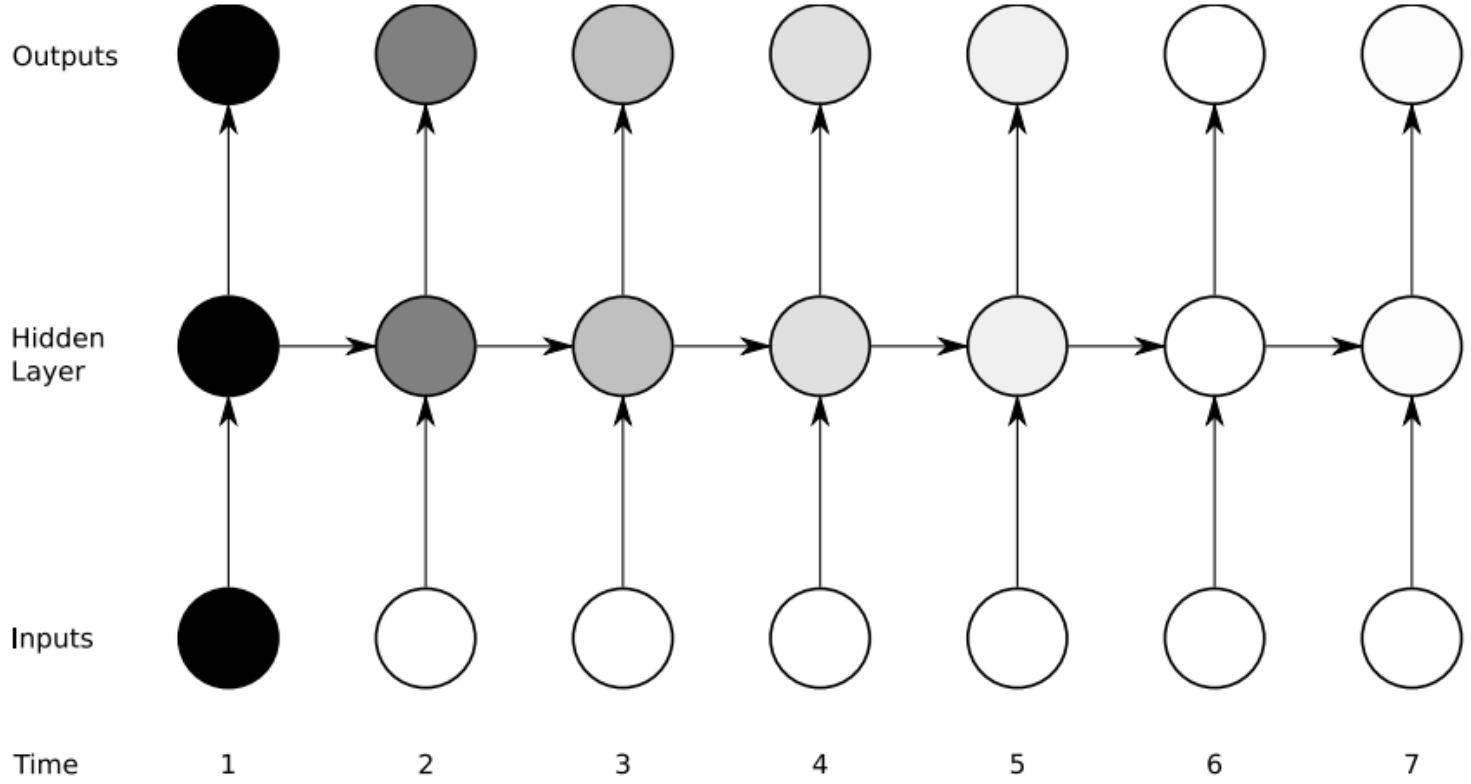


Figure 1: Sensitivity to the input at time one, Source: Graves 2008

Gating against Vanishing Gradients

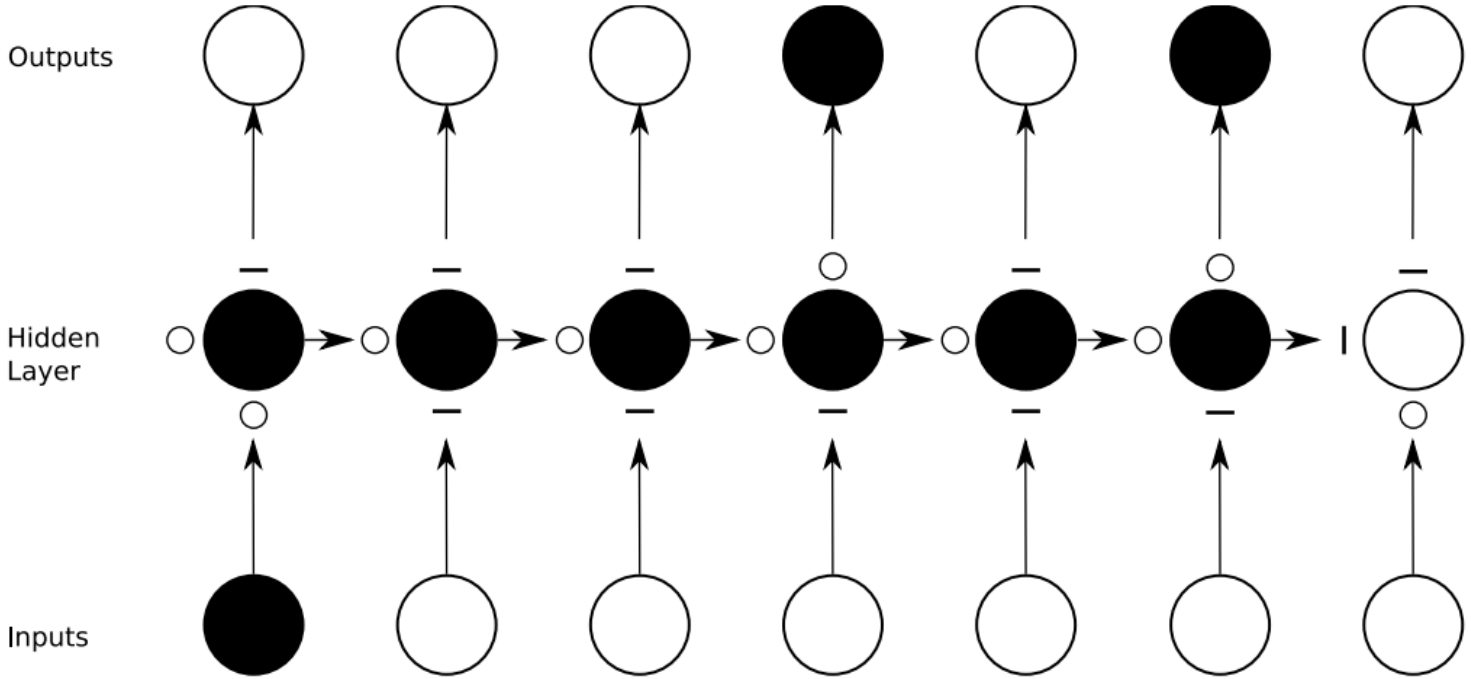


Figure 2: Gating helps to remember, Source: Graves 2008

Gated Recurrent Units (GRUs)

- GRU layer:

$$z^t := v^t \odot z^{t-1} + (1 - v^t) \odot a(W^z x^t + V^z (r^t \odot z^{t-1}))$$

with $v^t := a(W^v x^t + V^v z^{t-1} + b^v)$ **update gate**

$r^t := a(W^r x^t + V^r z^{t-1} + b^r)$ **reset gate**

$$W^z, W^v, W^r \in \mathbb{R}^{K \times M}, \quad V^z, V^v, V^r \in \mathbb{R}^{K \times K}, \quad b^z, b^v, b^r \in \mathbb{R}^K$$

- $v_t = 1$: gate $x_t \rightsquigarrow z_t$ closed
- Q: What happens in the following edge cases?
  Assign edge cases to possible effects.

edge cases		possible effects
a. $\forall t : v^t = 1$		1. usual conv-1 layer
b. $\forall t : v^t = 0, r^t = 1$	$\overset{?}{\rightsquigarrow}$	2. usual recurrent layer
c. $\forall t : v^t = 0, r^t = 0$		3. $z^t = z_0$ constant in t, not dependend on x

Gated Recurrent Units (GRUs)

- GRU layer:

$$z^t := v^t \odot z^{t-1} + (1 - v^t) \odot a(W^z x^t + V^z(r^t \odot z^{t-1}))$$

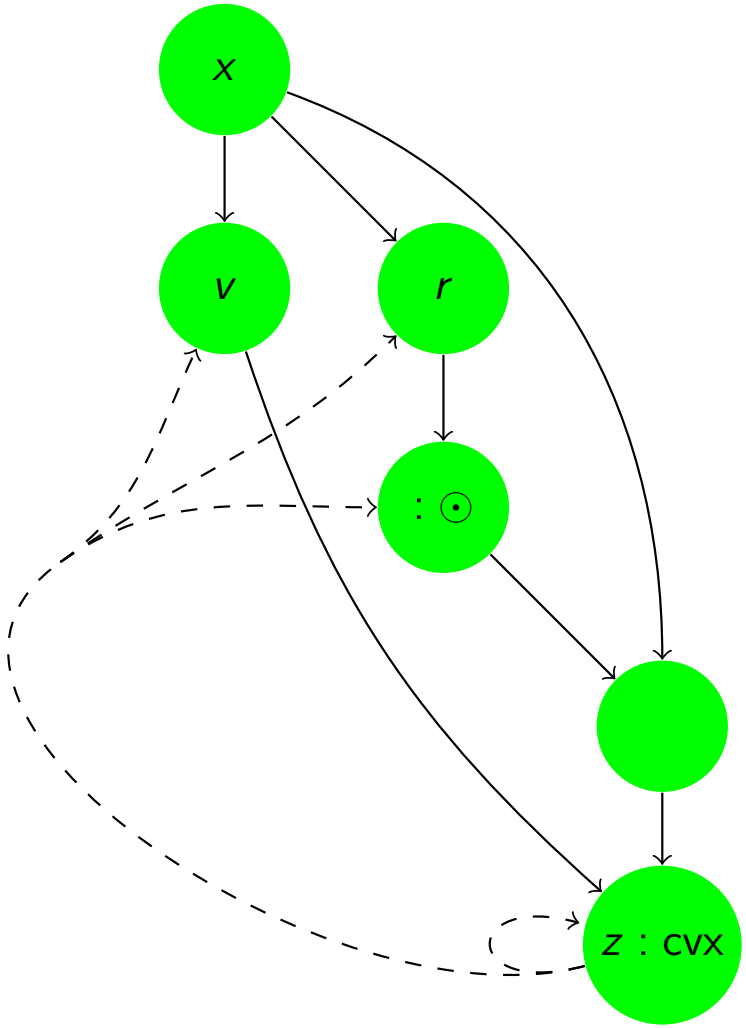
with $v^t := a(W^v x^t + V^v z^{t-1} + b^v)$ **update gate**

$r^t := a(W^r x^t + V^r z^{t-1} + b^r)$ **reset gate**

$$W^z, W^v, W^r \in \mathbb{R}^{K \times M}, \quad V^z, V^v, V^r \in \mathbb{R}^{K \times K}, \quad b^z, b^v, b^r \in \mathbb{R}^K$$

- $v_t = 1$: gate $x_t \rightsquigarrow z_t$ closed
- edge cases:
  - $\forall t : v^t = 1 \quad \rightsquigarrow z^t = z_0$ constant in t, not dependend on x
  - $\forall t : v^t = 0, r^t = 1 \rightsquigarrow$ usual recurrent layer
  - $\forall t : v^t = 0, r^t = 0 \rightsquigarrow$ usual conv-1 layer

The GRU Cell

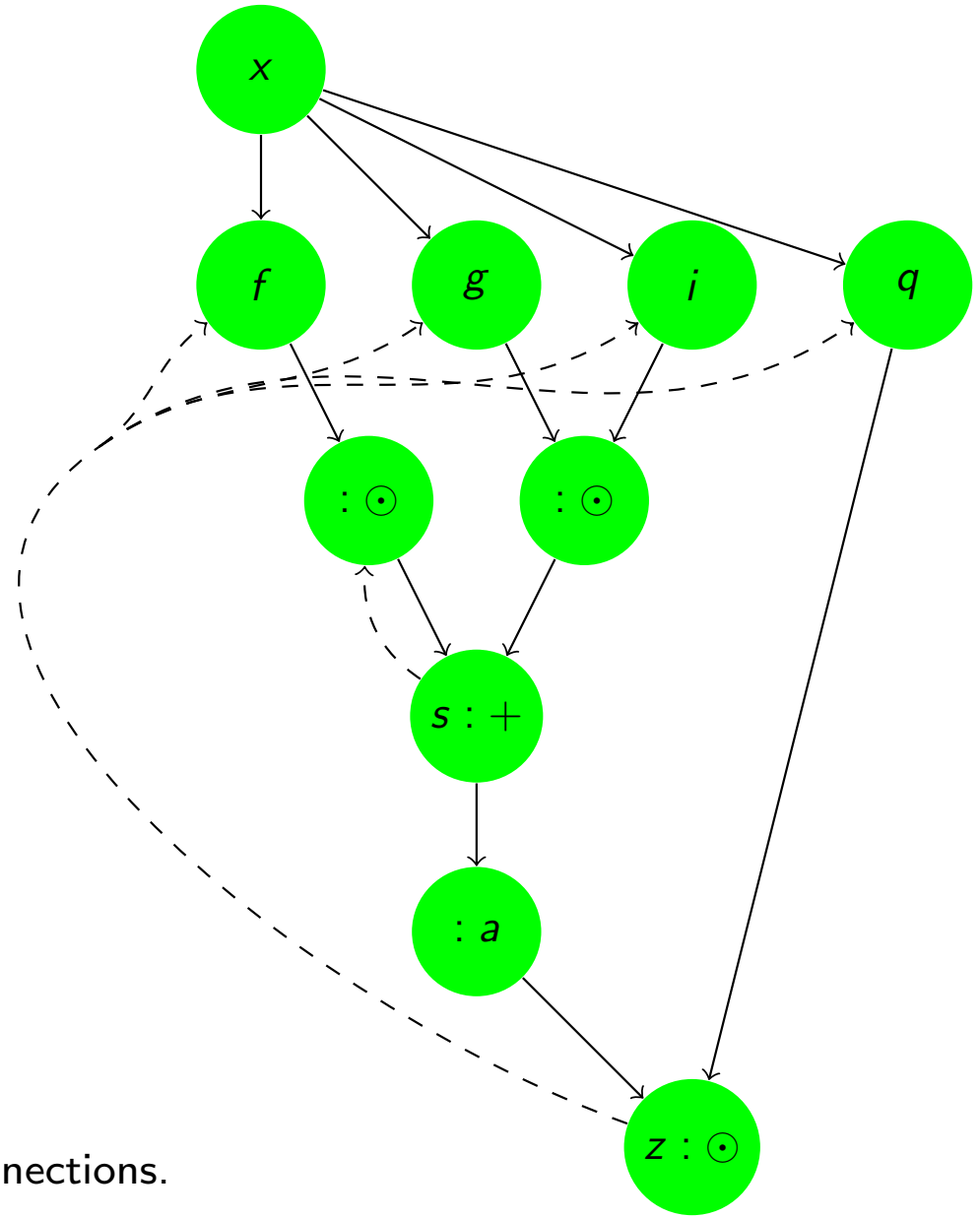


Note: dashed: temporal connections. cvx: convex combination $u \odot v + (1 - u) \odot w$.

Long Short-Term Memory (LSTM)

$$f^t := a(W^f x^t + V^f z^{t-1} + b^f) \quad \textbf{forget gate}$$
$$g^t := a(W^g x^t + V^g z^{t-1} + b^g) \quad \textbf{input gate}$$
$$q^t := a(W^q x^t + V^q z^{t-1} + b^q) \quad \textbf{output gate}$$
$$s^t := f^t \odot s^{t-1} + g^t \odot a(W^s x^t + V^s z^{t-1} + b^s) \quad \textbf{state}$$
$$z^t := a(s^t) \odot q^t \quad \textbf{output}$$

The LSTM Cell



Note: dashed: temporal connections.

Clipping gradients

RNN produces strongly nonlinear loss functions which create cliffs:

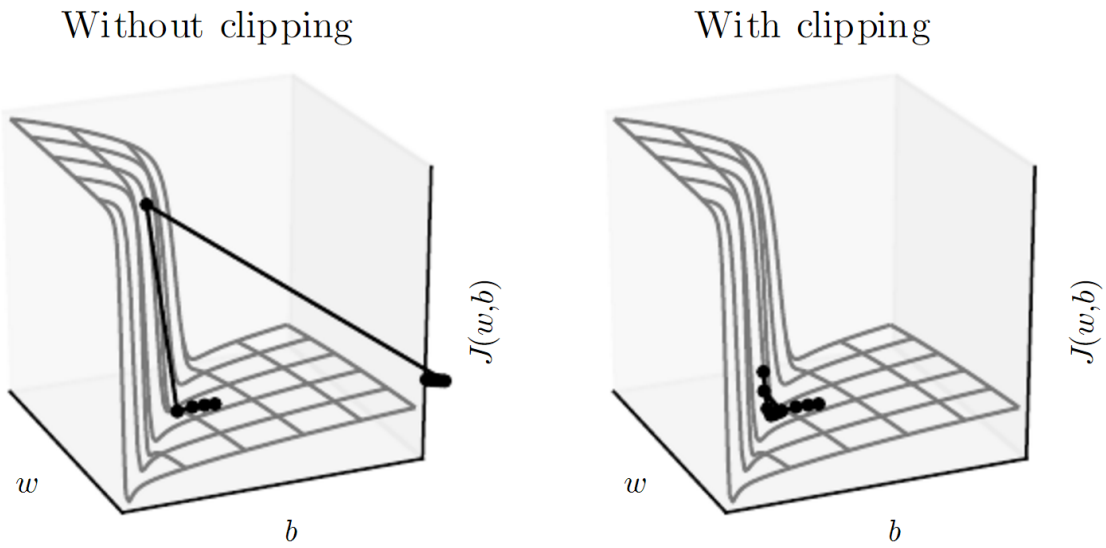


Figure 3: Clipping can avoid exploding gradients, Source: Goodfellow et al., 2016

A simple solution is the gradient clipping heuristic:

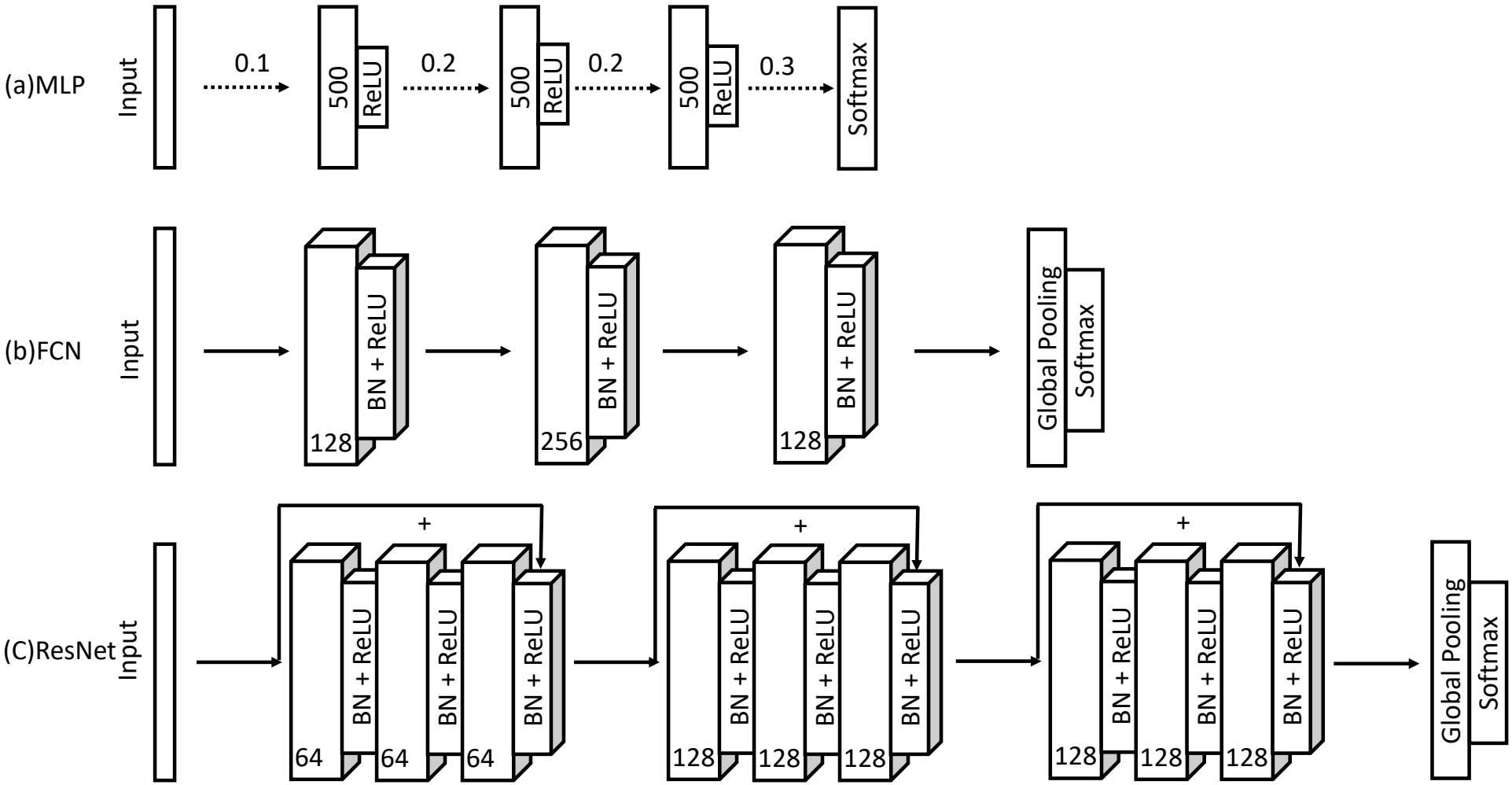
$$\text{if } ||g|| > v \text{ then } g := \frac{gv}{||g||}$$

Outline

RNNs for Time Series Classification

- like convolutional layers, recurrent layers retain the size of the time dimension.
- like for CNNs, also in RNNs **pooling layers** could be used to reduce time resolution
  - esp. **global pooling** for a fixed size of the latent feature vector
- once the time dimension is eliminated, fully-connected layers can be added.
- CNNs are a well-known architecture for time series classification sometimes called **fully convolutional neural networks** [Wang et al., 2017]
- RNNs are used less frequently this way [Ismail Fawaz et al., 2019].

CNNs for Time Series Classification



Note: Here dashed lines for MLP denotes dropout. CNN kernel sizes are 8, 5 and 3.

RNNs for Time Series Forecasting

- use the time shifted input as output:

$$\tilde{x} := x_{1:T-1}, \quad T := |x|y \qquad := x_{2:T}, \quad \text{i.e., } y_t := x_{t+1}$$

  - with forecasting horizion $h \in \mathbb{N}$:

$$\tilde{x} := x_{1:T-h}, \quad T := |x|y \qquad := x_{1+h:T}, \quad \text{i.e., } y_t := x_{t+h}$$

- possible, because RNNs use only information from earlier time slices.
  - works the same for CNNs with kernels having their reference point at the largest index / on the right, not in the center.

Summary (1/2)

- There are several supervised learning problems for sequence data:
  - **sequence prediction**: scalar/vector targets.
  - **sequence-to-sequence prediction**: an unaligned sequence targets.
  - **sequence labeling**: aligned sequence targets.
- A **recurrent layer** consists of
  - a neuron for each time slice,
  - fully connected to the input at the same time and its sibling neuron a timeslice earlier (same layer).
- A recurrent neural network with a single hidden layer consists of
  - a hidden recurrent layer and
  - a convolutional output layer with kernel size 1.
- Recurrent layers can be stacked to **deep recurrent neural networks**.
- In RNNs gradients can be computed and thus parameters learned by **backpropagation through time (bptt)**.

Summary (2/2)

- In RNNs **long-term dependencies** suffer from **vanishing or exploding gradients**.
- **Gating** is used to learn relevant dependencies between inputs and outputs across time.
  - **Gated Recurrent Units (GRUs)** with update and reset gates.
  - **Long Short-Term Memory (LSTM)** with
    - forget, input and output gates and
    - separate states and outputs

Further Readings

- Goodfellow et al. 2016, ch. 10
- Zhang et al. 2020, ch. 8 & 9
- the part of speech example is taken in modified form from [Sarkar, 2016, p. 138].

Acknowledgement: An earlier version of the slides for this lecture have been written by my former postdoc Dr Josif Grabocka.

References

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The Mit Press, Cambridge, Massachusetts, November 2016. ISBN 978-0-262-03561-3.

Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: A review. *Data Mining and Knowledge Discovery*, 33(4):917–963, July 2019. ISSN 1573-756X. doi: 10.1007/s10618-019-00619-1.

Dipanjan Sarkar. *Text Analytics with Python: A Practical Real-World Approach to Gaining Actionable Insights from Your Data*. Apress, 2016. ISBN 978-1-4842-2388-8. doi: 10.1007/978-1-4842-2388-8.

Z. Wang, W. Yan, and T. Oates. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 1578–1585, May 2017. doi: 10.1109/IJCNN.2017.7966039.

Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander Smola. *Dive into Deep Learning*. https://d2l.ai/, 2020.