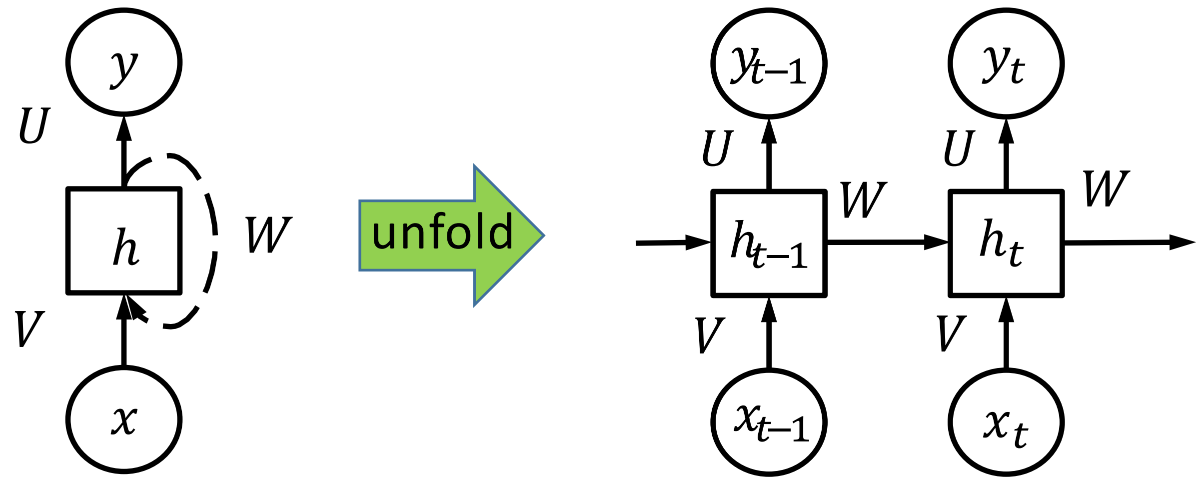
Recurrent neural networks for malware detection

Ekaterina Lobacheva, Dmitry Vetrov

Moscow
2016

Recurrent neural network



$$h_t = g(Vx_t + Wh_{t-1} + b_h)$$
$$y_t = f(Uh_t + b_y)$$

$$\frac{\partial F_T}{\partial h_t} = \frac{\partial F_T}{\partial h_T} \prod_{k=t}^{T-1} \frac{\partial h_{k+1}}{\partial h_k} = \frac{\partial F_T}{\partial h_T} \prod_{k=t}^{T-1} \boxed{diag(g'(\dots))W}$$

exploding or vanishing gradients

no long-range dependencies

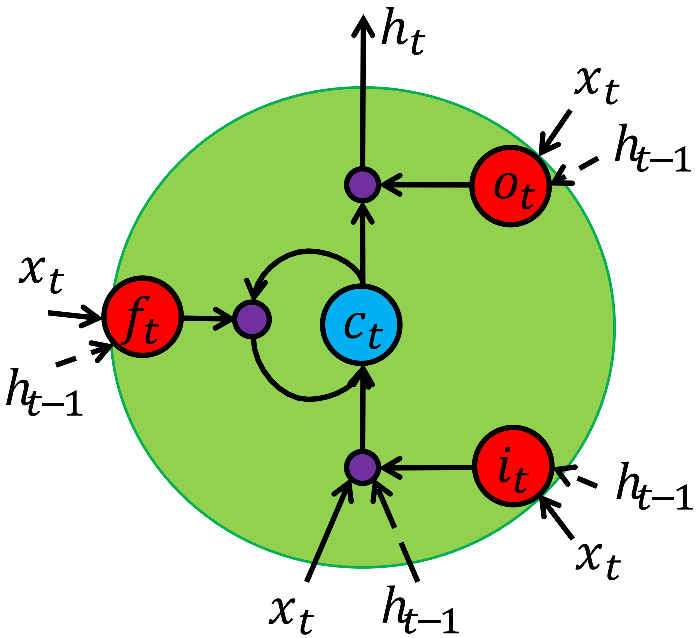
RNN: modifications

- Gradient clipping (Mikolov, 2012; Pascanu et al., 2012)

- Gated models:
  LSTM (Hochreiter and Schmidhuber, 1997)
  GRU (Cho et al., 2014)
  SCRN (Mikolov et al., 2015)
- Orthogonal and unitary matrices in RNN (Saxe et al., 2014; Le et al., 2015; Arjovsky and Shah and Bengio, 2016)

- Echo State Networks (Jaeger and Haas, 2004; Jaeger, 2012)
- Second-order optimization (Martens, 2010; Martens & Sutskever, 2011)
- Regularization (Pascanu et al., 2012)
- Careful initialization (Sutskever et al., 2013)

LSTM



Gate values in [0,1]

i_t, o_t, f_t - input/output/forget gates

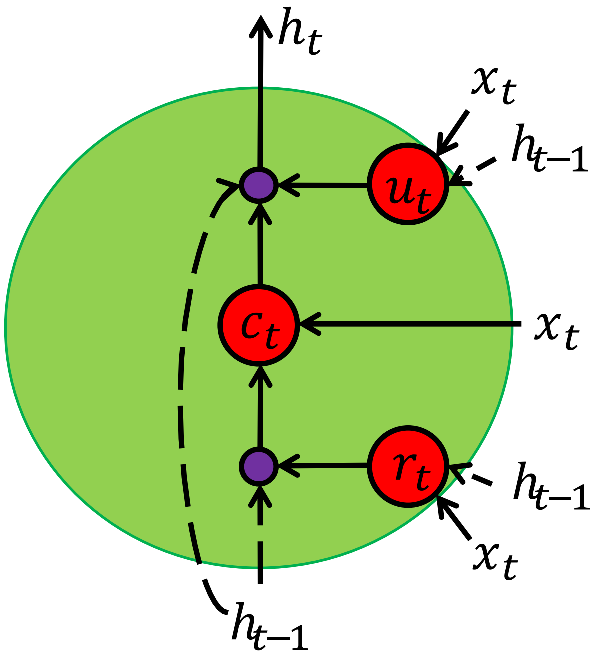
c_t - memory

$$i_t = \sigma(V_i x_t + W_i h_{t-1} + b_i)$$
$$f_t = \sigma(V_f x_t + W_f h_{t-1} + b_f)$$
$$o_t = \sigma(V_o x_t + W_o h_{t-1} + b_o)$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot g(V_c x_t + W_c h_{t-1} + b_c)$$
$$h_t = o_t \cdot g(c_t)$$

$$\frac{\partial h_{k+1}}{\partial h_k} \Rightarrow \frac{\partial c_{k+1}}{\partial c_k} = f_{k+1} \Rightarrow \text{High initial } b_f$$

GRU



Gate values in [0,1]

r_t, u_t - reset/update gates

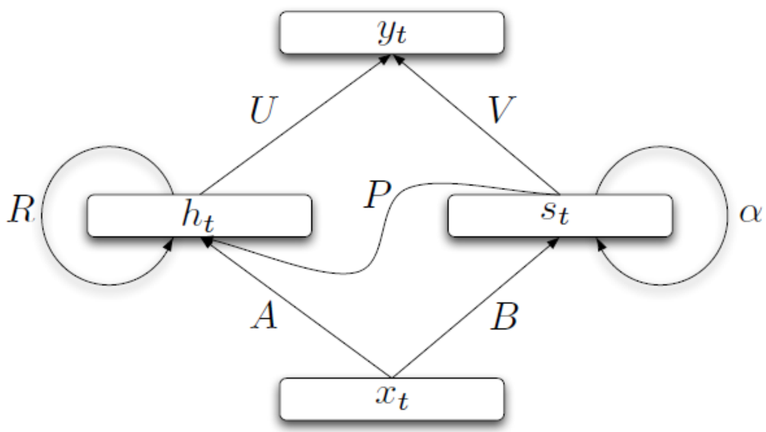
$$u_t = \sigma(V_u x_t + W_u h_{t-1} + b_u)$$
$$r_t = \sigma(V_r x_t + W_r h_{t-1} + b_r)$$

$$c_t = g(V_c x_t + W_c(h_{t-1} \cdot r_t))$$
$$h_t = (1 - u_t) \cdot c_t + u_t \cdot h_{t-1}$$

$$\frac{\partial h_{k+1}}{\partial h_k} = u_{k+1} + (1 - u_{k+1}) \cdot \frac{\partial c_{k+1}}{\partial h_k} \Rightarrow \text{High initial } b_u$$

SCRN

Structurally Constrained Recurrent Network



$$s_t = (1-\alpha)Bx_t + \alpha s_{t-1}$$

$$h_t = g(Ps_t + Ax_t + Rh_{t-1})$$

$$y_t = f(Uh_t + Vs_t)$$

α

- fixed
- trainable parameter Q (diagonal matrix):

$$s_t = (I-Q)Bx_t + Qs_{t-1}$$

$$\frac{\partial F_T}{\partial s_t} = \frac{\partial F_T}{\partial s_T}\prod_{k=t}^{T-1}\frac{\partial s_{k+1}}{\partial s_k} + \cdots = \frac{\partial F_T}{\partial s_T}\alpha^{T-t} + \cdots$$

Orthogonal and unitary matrices

$$\frac{\partial F_T}{\partial h_t} = \frac{\partial F_T}{\partial h_T}\prod_{k=t}^{T-1}\frac{\partial h_{k+1}}{\partial h_k} = \frac{\partial F_T}{\partial h_T}\prod_{k=t}^{T-1}\boxed{diag\big(g'(\ldots)\big)W}$$

$$\left\|\frac{\partial F_T}{\partial h_t}\right\| = \left\|\frac{\partial F_T}{\partial h_T}\prod_{k=t}^{T-1} DW\right\| \leq \left\|\frac{\partial F_T}{\partial h_T}\right\|\prod_{k=t}^{T-1}\|DW\| =$$

W $\Rightarrow$ Orthogonal or unitary $\Rightarrow$ $$= \left\|\frac{\partial F_T}{\partial h_T}\right\|\prod_{k=t}^{T-1}\|D\| =$$

D $\Rightarrow$ ReLU $\Rightarrow$ $$= \left\|\frac{\partial F_T}{\partial h_T}\right\|$$

uRNN

$$\boxed{W = D_3 R_2 F^{-1} D_2 \Pi R_1 F D_1}$$

- $\mathbf{D}$, a diagonal matrix with $\mathbf{D}_{j,j} = e^{iw_j}$, with parameters $w_j \in \mathbb{R}$,
- $\mathbf{R} = \mathbf{I} - 2\frac{vv^*}{\|v\|^2}$, a reflection matrix in the complex vector $v \in \mathbb{C}^n$,
- $\mathbf{\Pi}$, a fixed random index permutation matrix, and
- $\mathcal{F}$ and $\mathcal{F}^{-1}$, the Fourier and inverse Fourier transforms.

Complex:

- hidden units,
- in-to-hidden
- hidden-to-hidden

$$\Rightarrow \quad o_t = f\left(U \begin{pmatrix} Re(h_t) \\ Im(h_t) \end{pmatrix} + b_o\right)$$

$$modReLU(z) = \begin{cases} (|z| + b)\dfrac{z}{|z|} & if\ |z| + b \geq 0 \\ 0 & if\ |z| + b < 0 \end{cases}$$

uRNN

Pros:

- No vanishing or exploding gradients
- Memory: $O(n)$, time: $O(n\ logn)$
- Good parametrization: $O(n)$ parameters $\rightarrow$ more hidden units
- Very long dependencies

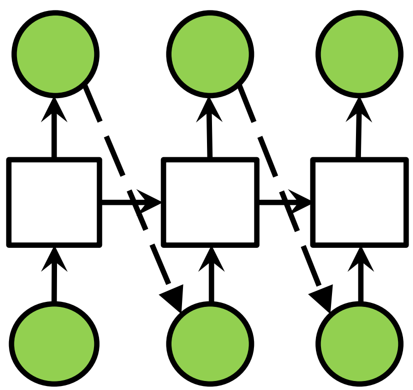
Cons:

- LSTM has stronger local dependencies

RNN for sequence classification

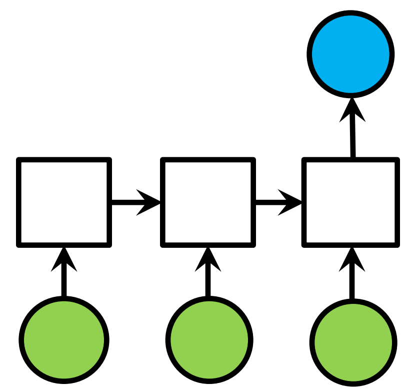
$(x_1, \dots, x_n) \rightarrow y \in \{0,1\}$

Generative model
for each y



Discriminative models

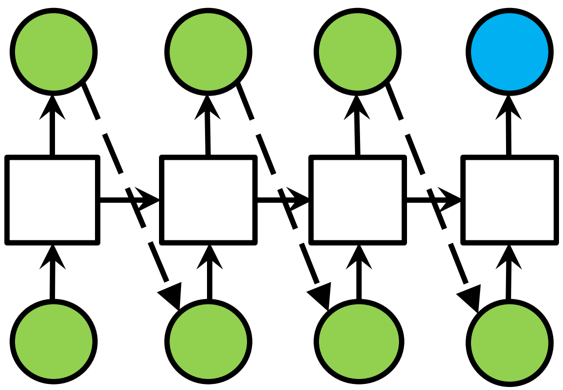
many to one



RNN for sequence classification

$(x_1, \dots, x_n) \rightarrow y \in \{0,1\}$

Generative + Discriminative



URL classification

Malicious URL examples:

- qoog1e.com
- blog.arrowservice.net
- 777nmsc.com

Data: train ≈ 20k, test ≈ 10k

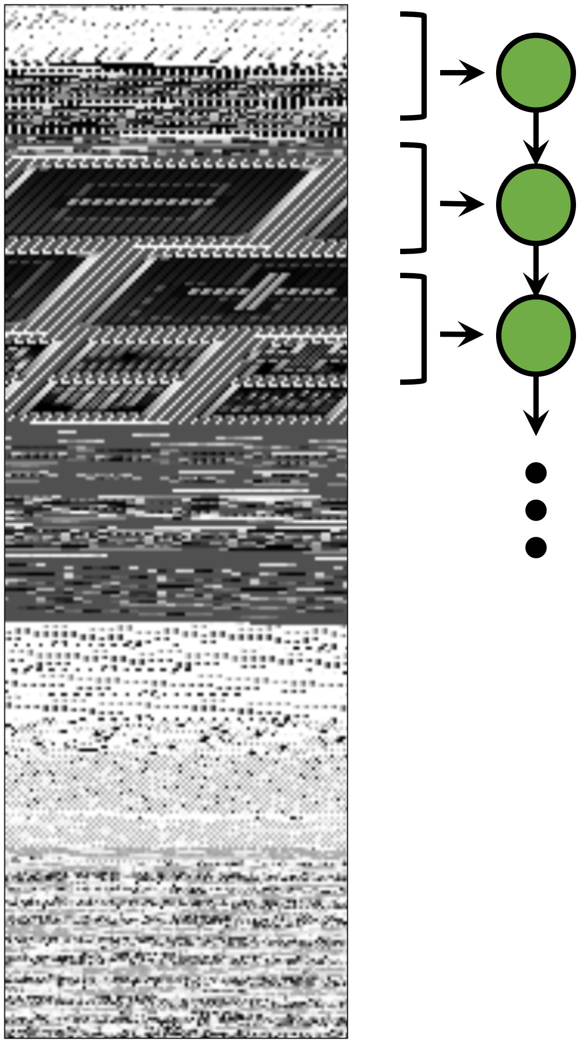
Architecture:

- 2RNN: 500+300,
- 0.3 Dropout

	Error, %
3-grams	13.3
Boosting: 3-grams + other	11.4
G-RNN	**9.8**
D-RNN: many to one	**9.2**
GD-RNN	**9.4**

LSTM, GRU: no improvements

Binary files classification



Why is it hard:

- Non-symmetric task
- Noisy labeled data
- Big data
- Dynamic data distribution

Too much data for RNN:

- millions of files
- average file size ≈ 2MB

We need to compress it:

- Handcrafted features
- Pretrained autoencoder/CNN
- Joint training of RNN and autoencoder/CNN