UNIVERSITY OF CAMBRIDGE

Long Short-Term Memory in Recurrent Neural Network

Shixiang Gu, Andrey Malinin

sg717@cam.ac.uk, am969@cam.ac.uk

November 20, 2014

Outline

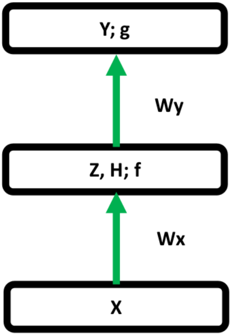
Machine Learning Review

Discrimitive Model learns $P(Y|X)$
Generative Model learns $P(X, Y)$, or $P(X)$
Supervised Learning uses pairs of input X and output Y, and aims to learn $P(Y|X)$ or $f : X \longrightarrow Y$
Unsupervised Learning uses input X only, and aims to learn $P(X)$ or $f : X \longrightarrow H$, where H is "better" representation of X
Models can be **stochastic** or **deterministic**

In this presentation, we focus on **deterministic, supervised, discrimitive model** based on **Recurrent Neural Network (RNN)/Long Short-Term Memory (LSTM)**

Applications: Speech Recognition, Machine Translation, Online Handwriting Recognition, Language Modeling, Music Composition, Reinforcement Learning, etc.

input: $X \in \mathbb{R}^Q$
target output: $Y \in \mathbb{R}^D$
predicted output: $\hat{Y} \in \mathbb{R}^D$
hidden: $H, Z \in \mathbb{R}^P$
weights: $W_X \in \mathbb{R}^{P \times Q}, W_Y \in \mathbb{R}^{D \times P}$
activations: $f : \mathbb{R}^P \longrightarrow \mathbb{R}^P, g : \mathbb{R}^D \longrightarrow \mathbb{R}^D$
loss function: $L : \mathbb{R}^D \times \mathbb{R}^D \longrightarrow \mathbb{R}$
loss: $E \in \mathbb{R}$
objective: minimize total loss E for $(X^{(i)}, Y^{(i)})_{i=1\dots N}$ training examples



Forward Propagation:
$Z = W_X \cdot X$
$H = f(Z)$
$\hat{Y} = g(W_Y \cdot H)$
$E = L(Y, \hat{Y})$

Activation functions

Common activation functions $f : X \longrightarrow Y$ on $X, Y \in \mathbb{R}^D$:

Linear: $f(X) = X$

Sigmoid/Logistic: $f(X) = \frac{1}{1+e^{-X}}$

Rectified Linear (ReLU): $f(X) = max(0, X)$

Tanh: $f(X) = tanh(X) = \frac{e^X - e^{-X}}{e^X + e^{-X}}$

Softmax: $f : Y_i = \frac{e^{X_i}}{\sum_{j=1}^{j=D} e^{X_i}}$

Most activations are **element-wise** and **non-linear**
Derivatives are easy to compute

NN: Backpropagation (1960s-1981)

UNIVERSITY OF CAMBRIDGE

Backpropagation (i.e. chain rule):

assume $g(X) = X$,

$L(Y, \hat{Y}) = \frac{1}{2}\sum_{i=1}^{D}(Y_i - \hat{Y}_i)^2$:

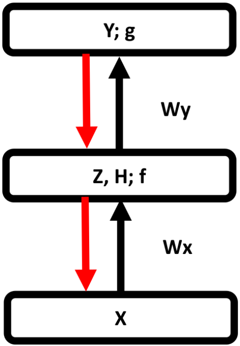
$\frac{\partial E}{\partial \hat{Y}} = \hat{Y} - Y$, $\frac{\partial \hat{Y}}{\partial H} = W_Y$, $\frac{\partial H}{\partial Z} = f'(Z)$

$$\frac{\partial E}{\partial Z} = \frac{\partial E}{\partial \hat{Y}}\frac{\partial \hat{Y}}{\partial H}\frac{\partial H}{\partial Z} = (W_Y^T \cdot (\hat{Y} - Y)) \odot f'(Z)$$

$$\frac{\partial E}{\partial W_Y} = \frac{\partial E}{\partial \hat{Y}}\frac{\partial \hat{Y}}{\partial W_Y} = (\hat{Y} - Y) \otimes H$$

$$\frac{\partial E}{\partial W_X} = \frac{\partial E}{\partial Z}\frac{\partial Z}{\partial W_X} = \frac{\partial E}{\partial Z} \otimes X$$

*abusing matrix calculus notations

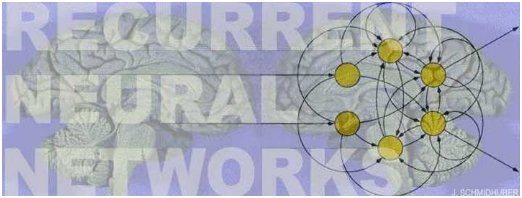


Stochastic Gradient Descent

learning rate: $\epsilon > 0$

$$W = W - \epsilon\frac{\partial E}{\partial W}, \forall W \in \{W_Y, W_X\}$$

RNN (1980s-)



Why Recurrent?

- Human brain is a recurrent neural network, i.e. has feedback connections
- A lot of data is sequential and dynamic (can grow or shrink)
- RNNs are Turing-Complete [Siegelmann and Sontag, 1995, Graves et al., 2014]

Unrolling RNNs

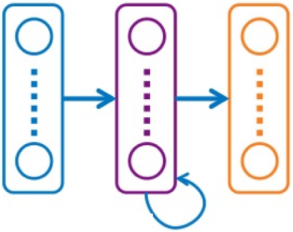


Figure: RNN

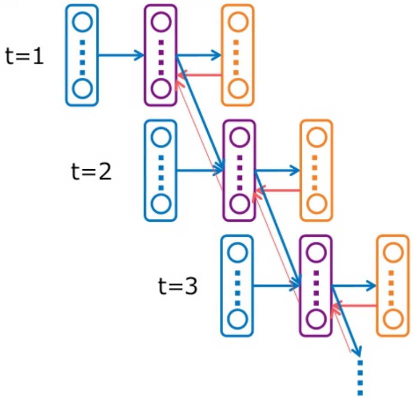


Figure: Unrolled RNN

RNN (1980s-)

UNIVERSITY OF CAMBRIDGE

Forward Propagation:

Given H_0

$Z_t = W_X \cdot X_t + W_H \cdot H_{t-1}$

$H_t = f(Z_t)$

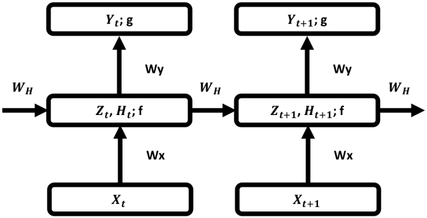
$\hat{Y}_t = g(W_Y \cdot H_t)$

$E_t = L(Y_t, \hat{Y}_t)$

$E = \sum_{t=1}^{T} E_t$

RNN=Very Deep NN w/ tied weights

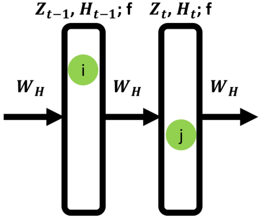
Sequence learning: unknown input len $\longrightarrow$ unknown output len



Vanishing/exploding gradients (1991)

UNIVERSITY OF CAMBRIDGE

Fundamental Problem in Deep Learning [Hochreiter, 1991]:



Let $v_{j,t} = \frac{\partial E}{\partial Z_{j,t}}$

$v_{i,t-1} = f'(Z_{i,t-1}) \cdot \sum_{j=1}^{P} w_{ji} \cdot v_{j,t}$

$$\frac{\partial v_{i,t-q}}{\partial v_{j,t}} = \sum_{l_1=1}^{P} \cdots \sum_{l_{q-1}=1}^{P} \prod_{m=1}^{q} f'_{l_m}(Z_{l_m,t-m}) \cdot w_{l_m,l_{m-1}}$$

$|f'_{l_m}(Z_{l_m,t-m}) \cdot w_{l_m,l_{m-1}}| > 1.0 \forall m \longrightarrow$ explodes

$|f'_{l_m}(Z_{l_m,t-m}) \cdot w_{l_m,l_{m-1}}| < 1.0 \forall m \longrightarrow$ vanishes

Gradient vanishes/increases exponentially in terms of time steps T

e.g. “bufferfly effect”

Early work on RNN (1980s-90s)

UNIVERSITY OF CAMBRIDGE

Exact gradient (TOO HARD!(back then)):

- **Backpropagation Through Time** (BPTT) [Werbos, 1990]

Approximate gradient:

- **Real Time Recurrent Learning** (RTRL) [Robinson and Fallside, 1987, Williams and Zipser, 1989]
- **Truncated BPTT** [Williams and Peng, 1990]

Non-gradient methods:

- **Random Guessing** [Hochreiter and Schmidhuber, 1996]

Unsupervised "pretraining":

- **History Compressor** [Schmidhuber, 1992]

Others:

- **Time-delay Neural Network** [Lang et al., 1990], **Hierarchical RNNs** [El Hihi and Bengio, 1995], **NARX** [Lin et al., 1996] and more...

sources: [Graves, 2006, Sutskever, 2013, Schmidhuber, 2014]

Pretraining RNN (1992)

History compressor (HC) [Schmidhuber, 1992] = unsupervised, greedy layer-wise pretraining of RNN
(Recall in standard DNN: **unsupervised pretraining** (AE,RBM) [Hinton and Salakhutdinov, 2006]$\longrightarrow$ **supervised** [Krizhevsky et al., 2012])

Forward Propagation:

Given $H_0, X_0, Y_t = X_{t+1}$

$Z_t = W_{XH} \cdot X_{t-1} + W_H \cdot H_{t-1}$

$H_t = f(Z_t)$

$\hat{Y}_t = g(W_Y \cdot H_t + W_{XY} \cdot X_t)$

$E_t = L(Y_t, \hat{Y}_t)$

$E = \sum_{t=0}^{T-1} E_t$

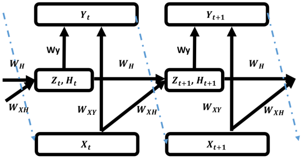


Figure: HC

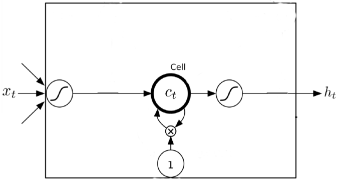
$X' \longleftarrow \{X_0, H_0, (t, X_t) \; \forall \; t \ni X_t \neq \hat{Y}_{t-1}\}$
$\longrightarrow$ train new HC using X'

Greedily pretrain RNN, using "surprises" from previous step

Basic "LSTM" (1991-95)

Idea: Have separate linear units that simply deliver errors [Hochreiter and Schmidhuber, 1997]

Forward Propagation*:
Let $C_t \in \mathbb{R}^P$ = "cell state",
Given H_0, C_0
$C_t = C_{t-1} + f_1(W_X \cdot X_t + W_H \cdot H_{t-1})$
$H_t = f_2(C_t)$
BPTT: Truncate gradients outside the cell
*approximately



Original LSTM (1995-97)

Idea: Have additional controls for R/W access (input/output gates) [Hochreiter and Schmidhuber, 1997]

Forward Propagation*:

Given H_0, C_0

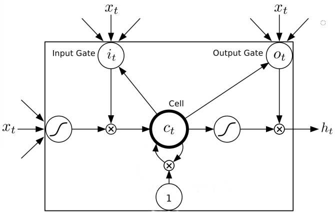
$C_t = C_{t-1} + i_t \odot f_1(W_X \cdot X_t + W_H \cdot H_{t-1})$

$H_t = o_t \odot f_2(C_t)$

$i_t = \sigma(W_{i,X} \cdot X_t + W_{i,C} \cdot C_{t-1} + W_{i,H} \cdot H_{t-1})$

$o_t = \sigma(W_{o,X} \cdot X_t + W_{o,C} \cdot C_t + W_{o,H} \cdot H_{t-1})$

BPTT: Truncate gradients outside the cell



*approximately

Modern LSTM (full version) (2000s-)

Idea: Have control for forgeting cell state (forget gate) [Gers, 2001, Graves, 2006]

Forward Propagation:

Given H_0, C_0

$C_t = f_t \odot C_{t-1} + i_t \odot f_1(W_X \cdot X_t + W_H \cdot H_{t-1})$

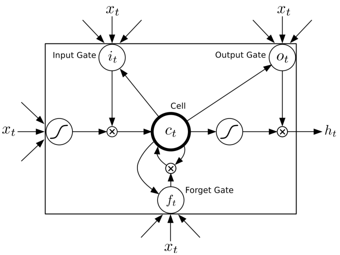
$H_t = o_t \odot f_2(C_t)$

$i_t = \sigma(W_{i,X} \cdot X_t + W_{i,C} \cdot C_{t-1} + W_{i,H} \cdot H_{t-1})$

$o_t = \sigma(W_{o,X} \cdot X_t + W_{o,C} \cdot C_t + W_{o,H} \cdot H_{t-1})$

$f_t = \sigma(W_{f,X} \cdot X_t + W_{f,C} \cdot C_{t-1} + W_{f,H} \cdot H_{t-1})$

BPTT: Use exact gradients



Recent Progress in RNN

Echo-State Network (ESN): [Jaeger and Haas, 2004]

- only train hidden-output weights
- other weights drawn from carefully-chosen distribution and fixed

Hessian-Free (HF) optimization: [Martens and Sutskever, 2011]

- approximate second-order method
- outperformed LSTM on small-scale tasks

Momentum methods: [Sutskever, 2013]

- Could we get **some** good results without HF?
- Yes! With “good” intialization & aggressive momemtum scheduling
- Nesterov’s accelerated gradient?

DEMO!

El Hihi, S. and Bengio, Y. (1995).
Hierarchical recurrent neural networks for long-term dependencies.
In *NIPS*, pages 493–499. Citeseer.

Gers, F. (2001).
Long Short-Term Memory in Recurrent Neural Networks.
PhD thesis, Ecole Polytechnique Federale de Lausanne.

Graves, A. (2006).
Supervised Sequence Labelling with Recurrent Neural Networks.
PhD thesis, Technische Universitat Munchen.

Graves, A., Wayne, G., and Danihelka, I. (2014).
Neural turing machines.
arXiv preprint arXiv:1410.5401.

Hinton, G. E. and Salakhutdinov, R. R. (2006).
Reducing the dimensionality of data with neural networks.
Science, 313(5786):504–507.

Hochreiter, S. (1991).
Untersuchungen zu dynamischen neuronalen Netzen.
PhD thesis, Institut fur Informatik, Technische Universitat Munchen.

Hochreiter, S. and Schmidhuber, J. (1996).
Bridging long time lags by weight guessing and\ long short-term memory.
Spatiotemporal models in biological and artificial systems, 37:65–72.

Hochreiter, S. and Schmidhuber, J. (1997).
Long short-term memory.
Neural computation, 9(8):1735–1780.

Jaeger, H. and Haas, H. (2004).
Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication.
Science, 304(5667):78–80.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012).

Imagenet classification with deep convolutional neural networks.
In *Advances in neural information processing systems*, pages 1097–1105.

Lang, A., Waibel, A., and Hinton, G. E. (1990).
A time-delay neural network architecture for isolated word recognition.
3:23–43.

Lin, T., Horne, B. G., Tino, P., and Giles, C. L. (1996).
Learning long-term dependencies in narx recurrent neural networks.
Neural Networks, IEEE Transactions on, 7(6):1329–1338.

Martens, J. and Sutskever, I. (2011).
Learning recurrent neural networks with hessian-free optimization.
In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1033–1040.

Robinson, A. and Fallside, F. (1987).
The utility driven dynamic error propagation network.
University of Cambridge Department of Engineering.

Schmidhuber, J. (1992).
Learning complex, extended sequences using the principle of history compression.
Neural Computation, 4(2):234–242.

Schmidhuber, J. (2014).
Deep learning in neural networks: An overview.
arXiv preprint arXiv:1404.7828.

Siegelmann, H. T. and Sontag, E. D. (1995).
On the computational power of neural nets.
Journal of computer and system sciences, 50(1):132–150.

Sutskever, I. (2013).
Training Recurrent Neural Networks.
PhD thesis, University of Toronto.

Werbos, P. J. (1990).
Backpropagation through time: what it does and how to do it.
Proceedings of the IEEE, 78(10):1550–1560.

Williams, R. J. and Peng, J. (1990).
An efficient gradient-based algorithm for on-line training of recurrent network trajectories.
Neural Computation, 2(4):490–501.

Williams, R. J. and Zipser, D. (1989).
A learning algorithm for continually running fully recurrent neural networks.
Neural computation, 1(2):270–280.