Numerical Linear Algebra Methods in Recurrent Neural Networks

Qiang Ye
(Joint works with Kyle Helfrich, Gayan Maduranga, Devin Willmott)

University of Kentucky

July 16, 2020 - S.T. Yau Center, Southeast University

Outline

Supervised Learning

Supervised Learning

Given a labeled data set $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N} \subset \mathbf{R}^m \times \mathbf{R}^n$, fit a parametric family of functions $f : (\mathbf{x}, \theta) \in \mathbf{R}^m \times \mathbf{R}^p \to \mathbf{R}^n$ to the data;

Supervised Learning

Supervised Learning

Given a labeled data set $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N \subset \mathbf{R}^m \times \mathbf{R}^n$, fit a parametric family of functions $f : (\mathbf{x}, \theta) \in \mathbf{R}^m \times \mathbf{R}^p \to \mathbf{R}^n$ to the data;

- Choose $f(x, \theta)$
- Choose a loss function $\mathcal{L}(\theta) = \sum_{i=1}^N L(f(x_i, \theta), y_i)$
- find $\theta \in \mathbf{R}^p$ by minimizing $\mathcal{L}(\theta)$

Supervised Learning

Supervised Learning

Given a labeled data set $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N \subset \mathbf{R}^m \times \mathbf{R}^n$, fit a parametric family of functions $f : (\mathbf{x}, \theta) \in \mathbf{R}^m \times \mathbf{R}^p \to \mathbf{R}^n$ to the data;

- Choose $f(x, \theta)$
- Choose a loss function $\mathcal{L}(\theta) = \sum_{i=1}^N L(f(x_i, \theta), y_i)$
- find $\theta \in \mathbf{R}^p$ by minimizing $\mathcal{L}(\theta)$

Example. Linear regression:

1. $f(x, \theta) = Wx + b$ with $\theta = [W, b]$
2. $\mathcal{L}(\theta) = \sum_{i=1}^N \|f(x_i, \theta) - y_i\|^2$

Deep Neural Network

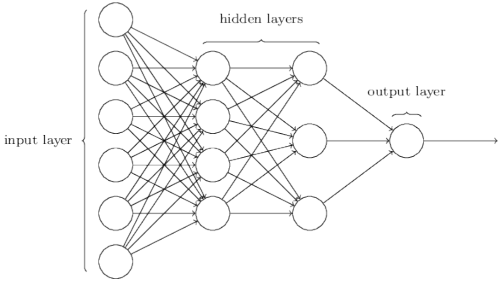


Image source: Goodfellow, et al.

- Composition of L functions:

$$f(\mathbf{x}, \theta) = f^{(3)}(f^{(2)}(f^{(1)}(\mathbf{x})))$$

- hidden variables at ℓ-th layer:

$$\begin{aligned} h^{(\ell)} &= f^{(\ell)}(h^{(\ell-1)}) \\ &:= \sigma(W^{(\ell)} h^{(\ell-1)} + b^{(\ell)}) \end{aligned}$$

- $\sigma(t)$: an elementwise nonlinear activation function:
  - Rectified linear unit (ReLu) $\sigma(t) = \max\{t, 0\}$
  - Logistic sigmoid $\sigma(t) = 1/(1 + e^{-t})$
  - Tanh $\sigma(t) = \tanh(t)$

Loss Function

For the model output $\hat{y}_i := f(x_i, \theta)$, use loss $\mathcal{L}(\theta)$:

- Regression problem: MSE

$$\mathcal{L}(\theta) = \sum_i \|\hat{y}_i - y_i\|^2$$

- Classification problem: Cross-Entropy

$$\mathcal{L}(\theta) = -\sum_i \sum_j y_j^{(i)} \log(\hat{y}_j^{(i)})$$

Optimization/Training

Gradient descent:

$$\theta \leftarrow \theta - \lambda \nabla \mathcal{L}(\theta)$$

- $\lambda > 0$ - learning rate
- Mini-batch training;
- Back-propagation algorithm
- Accelerations: SGD with momentum, Adams, RMSPROP, Batch normalization, ...

Vanishing gradients

$\nabla\mathcal{L}(\theta) \approx 0$ for θ in some large regions not near local minimum.

- Logistic sigmoid and tanh: $\sigma'(t) \approx 0$ for most t;
- ReLU: $\sigma'(t) = 0$ for $t < 0$
- Choice of $\mathcal{L}(\theta)$
- initialization
- depth of the network: multiplications of L weight matrices

Recurrent Neural Network (RNN)

RNN Basics

Sequential data $\mathbf{x} = (\mathbf{x}^{(t)})_{t=1}^{\tau}$.

- Language Processing
- Audio and Video Files

RNN Basics

Sequential data $\mathbf{x} = (\mathbf{x}^{(t)})_{t=1}^{\tau}$.

- Language Processing
- Audio and Video Files

Difficulties with feedforward network models:

- high dimensional inputs
- variable sequence length

RNN Basics

State-space model of input-output systems:

$$\mathbf{h}^{(t)} = \sigma\left(\mathbf{U}^T\mathbf{x}^{(t)} + \mathbf{W}^T\mathbf{h}^{(t-1)} + \mathbf{b}\right)$$
$$\mathbf{o}^{(t)} = \mathbf{V}^T\mathbf{h}^{(t)} + \mathbf{c}$$

- Input: $\mathbf{x} = (\mathbf{x}^{(t)})_{t=1}^{\tau}$ with $\mathbf{x}^{(t)} \in \mathbb{R}^m$
- Output: $\mathbf{o} = (\mathbf{o}^{(t)})_{t=1}^{\tau}$ with $\mathbf{o}^{(t)} \in \mathbb{R}^p$
- State $\mathbf{h} = (\mathbf{h}^{(t)})_{t=1}^{\tau}$ with $\mathbf{h}^{(t)} \in \mathbb{R}^m$
- $\mathcal{L}(\theta) = \sum_i L(\mathbf{o}_i^{(t)}, \mathbf{y}_i^{(t)})$
- Often output at τ only: $\mathbf{o} = \mathbf{V}^T\mathbf{h}^{(\tau)} + \mathbf{c}$

RNN Basics

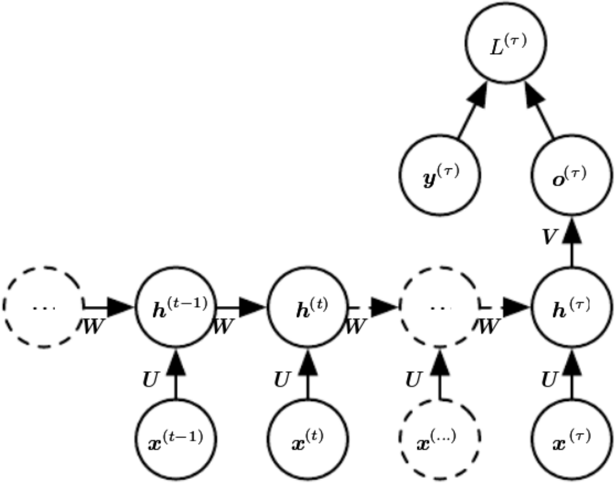


Image source: Goodfellow, et al.

Backpropagation Through Time

$$\frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(t)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(t+1)}} \frac{\partial \mathbf{h}^{(t+1)}}{\partial \mathbf{h}^{(t)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(t+1)}} \mathbf{D}^{(t)} \mathbf{W}^T$$

where $\mathbf{D}^{(t)} = diag\left(\sigma'\left(\mathbf{U}^T \mathbf{x}^{(t)} + \mathbf{W}^T \mathbf{h}^{(t-1)} + \mathbf{b}\right)\right)$

Backpropagation Through Time

$$\frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(t)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(t+1)}} \frac{\partial \mathbf{h}^{(t+1)}}{\partial \mathbf{h}^{(t)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(t+1)}} \mathbf{D}^{(t)} \mathbf{W}^T$$

where $\mathbf{D}^{(t)} = diag\left(\sigma'\left(\mathbf{U}^T \mathbf{x}^{(t)} + \mathbf{W}^T \mathbf{h}^{(t-1)} + \mathbf{b}\right)\right)$

- $0 \leq \sigma'(t) \leq 1$ and $\|\mathbf{D}^{(k)}\| \leq 1$.
- Vanishing (if $\|W\| < 1$) or exploding (if $\|W\| > 1$) gradients:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(t)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(\tau)}}^T \left(\prod_{k=\tau}^{t+1} \mathbf{D}^{(k)} \mathbf{W}^T \right)$$

Backpropagation Through Time

$$\frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(t)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(t+1)}} \frac{\partial \mathbf{h}^{(t+1)}}{\partial \mathbf{h}^{(t)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(t+1)}} \mathbf{D}^{(t)} \mathbf{W}^T$$

where $\mathbf{D}^{(t)} = diag\left(\sigma'\left(\mathbf{U}^T \mathbf{x}^{(t)} + \mathbf{W}^T \mathbf{h}^{(t-1)} + \mathbf{b}\right)\right)$

- $0 \leq \sigma'(t) \leq 1$ and $\|\mathbf{D}^{(k)}\| \leq 1$.
- Vanishing (if $\|W\| < 1$) or exploding (if $\|W\| > 1$) gradients:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(t)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(\tau)}}^T \left(\prod_{k=\tau}^{t+1} \mathbf{D}^{(k)} \mathbf{W}^T \right)$$

- For $t \ll \tau$, $\mathbf{h}^{(t)}$ or $\mathbf{x}^{(t)}$ has little effect on $\mathcal{L}$ or $\mathbf{o}^{(\tau)}$

Long Short Term Memory (LSTM) Network

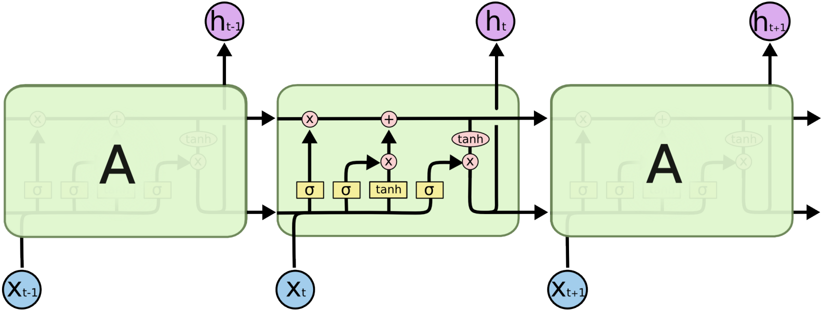


Image source: Colah's blog

Long Short Term Memory (LSTM) Network

- Most popular architecture of RNN
- Complicated network
- a large number of trainable parameters
- Other variants: Gated Recurrent Units (GRUs)

Unitary RNN (uRNN)

Unitary Evolution RNN (uRNN)

Use unitary or orthogonal W in RNN:

- Taking 2-norms

$$
\begin{aligned}
\left\| \frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(t)}} \right\| &\leq \left\| \frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(\tau)}} \right\| \prod_{k=t+1}^{\tau} \left\| \mathbf{D}^{(k)} \right\| \|\mathbf{W}\| \\
&\leq \left\| \frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(\tau)}} \right\|
\end{aligned}
$$

- How to construct W?
- Early attempts: initialize W to be orthogonal

Unitary Evolution RNN (uRNN)

Arjovsky, et al. (2016)

- Use a special unitary matrix:

$$\mathbf{W} = \mathbf{D}_3\mathbf{R}_2\mathcal{F}^{-1}\mathbf{D}_2\mathbf{\Pi}\mathbf{R}_1\mathcal{F}\mathbf{D}_1$$

- $\mathbf{D}_k$ - diagonal matrix with entries $\mathbf{D}_{j,j} = e^{iw_j}$ and $w_j \in \mathbb{R}$ (trainable)
- $\mathbf{R} = \mathbf{I} - 2\frac{\mathbf{v}\mathbf{v}^*}{\|\mathbf{v}\|^2}$ - Householder reflection matrices (trainable $\mathbf{v} \in \mathbb{C}^n$)
- $\mathbf{\Pi}$ - fixed random permutation matrix
- $\mathcal{F}$, $\mathcal{F}^{-1}$ - Discrete Fourier and inverse Fourier transforms
- Requres $7n$ in memory storage.

modReLU

- New activation function:

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

- $\sigma_{\text{modReLU}(z)} = \sigma_{ReLU}(|z| + b)\frac{z}{|z|}$
- Unlike ReLU, for the real case it can have postive and negative activation values.

Full-Capacity uRNN

Full-Capacity uRNN

Wisdom, et al. (2016)

- Find W from Stiefel Manifold $\mathcal{V}_p(\mathbb{C}^n) = \{\mathbf{X} \in \mathbb{C}^{n\times p} | \mathbf{X}^*\mathbf{X} = \mathbf{I}\}$
- Optimize $\min_{W^*W=I} \mathcal{L}(W)$;
- Updates W by moving along a descent curve on $\mathcal{V}_p(\mathbb{C}^n)$ by Wen and Yin (2013):

$$\mathbf{W}^{(k+1)} = \left(\mathbf{I} + \frac{\lambda}{2}\mathbf{A}^{(k)}\right)^{-1}\left(\mathbf{I} - \frac{\lambda}{2}\mathbf{A}^{(k)}\right)\mathbf{W}^{(k)}$$

- λ is the learning rate
- $\mathbf{A}^{(k)} = \mathbf{G}^{(k)}\mathbf{W}^{(k)^*} - \mathbf{W}^{(k)}\mathbf{G}^{(k)^*}$ is a skew-hermitian matrix ($\mathbf{A} = -\mathbf{A}^*$)
- $\mathbf{G}^{(k)} = \left[\frac{\partial \mathcal{L}}{\partial W_{i,j}}\right]_{i,j=1}^{n}$

Limitations of the Full-Capacity uRNN

- The descent curve not necessarily in the steepest descent direction.
- Loss of orthogonality due to repeated matrix multiplications.

Other orthogonal RNNs

oRNN: construct W by Householder reflections
euRNN: construct W by Givens rotations

- Long product $W = H_1 H_2 \cdots H_m$ nonlinearity
- More complicated learning algorithm
- Implemented with small m

Other orthogonal RNNs

oRNN: construct W by Householder reflections
euRNN: construct W by Givens rotations

- Long product $W = H_1 H_2 \cdots H_m$ nonlinearity
- More complicated learning algorithm
- Implemented with small m

expRNN: construct W through exponential of skew-symmetric matrix

- $W = exp(K)$

Scaled Cayley Orthogonal RNN (scoRNN)

Cayley Transform

Every real orthogonal matrix $\mathbf{W}$ that does not have -1 as an eigenvalue can be expressed as:

$$\mathbf{W} = (\mathbf{I} + \mathbf{A})^{-1}(\mathbf{I} - \mathbf{A})$$

where

$$\mathbf{A} = (\mathbf{I} + \mathbf{W})^{-1}(\mathbf{I} - \mathbf{W})$$

is skew-symmetric.

- Unstable when an eigenvalue of $\mathbf{W}$ is close to -1

Scaled Cayley Transform

Theorem 1 (Kahan, O'Dorney)

Every orthogonal $\mathbf{W} \in \mathcal{V}_n(\mathbb{R}^n)$ *can be expressed as*

$$\mathbf{W} = (\mathbf{I} + \mathbf{A})^{-1}(\mathbf{I} - \mathbf{A})\mathbf{D}$$

where $\mathbf{A} = [a_{ij}]$ *is real, skew-symmetric with* $|a_{ij}| \leq 1$*, and* $\mathbf{D}$ *is diagonal with all entries equal to* ± 1.

Every unitary $\mathbf{W} \in \mathcal{V}_n(\mathbb{C}^n)$ *can be expressed as*

$$\mathbf{W} = (\mathbf{I} + \mathbf{A})^{-1}(\mathbf{I} - \mathbf{A})\mathbf{D}$$

where $\mathbf{A} = [a_{ij}]$ *is skew-Hermitian with* $|a_{ij}| \leq 1$*, and* $\mathbf{D} = \mathrm{diag}\{e^{i\theta_1} \cdots, e^{i\theta_n}\}$.

Scaled Cayley Transform

Theorem 1 (Kahan, O'Dorney)

Every orthogonal $\mathbf{W} \in \mathcal{V}_n(\mathbb{R}^n)$ *can be expressed as*

$$\mathbf{W} = (\mathbf{I} + \mathbf{A})^{-1}(\mathbf{I} - \mathbf{A})\mathbf{D}$$

where $\mathbf{A} = [a_{ij}]$ *is real, skew-symmetric with* $|a_{ij}| \leq 1$*, and* $\mathbf{D}$ *is diagonal with all entries equal to* ± 1*.*
Every unitary $\mathbf{W} \in \mathcal{V}_n(\mathbb{C}^n)$ *can be expressed as*

$$\mathbf{W} = (\mathbf{I} + \mathbf{A})^{-1}(\mathbf{I} - \mathbf{A})\mathbf{D}$$

where $\mathbf{A} = [a_{ij}]$ *is skew-Hermitian with* $|a_{ij}| \leq 1$*, and* $\mathbf{D} = \mathrm{diag}\{e^{i\theta_1} \cdots, e^{i\theta_n}\}$*.*

- In practice, only need $|a_{ij}|$ bounded
- Achieved by many D

scoRNN Architecture

- Similar to a standard RNN:

$$\mathbf{z}^{(t)} = \mathbf{U}^T\mathbf{x}^{(t)} + \mathbf{W}^T\mathbf{h}^{(t-1)}$$
$$\mathbf{h}^{(t)} = \sigma_{\text{modReLU}}(\mathbf{z}^{(t)})$$

- $\mathbf{W} = (\mathbf{I} + \mathbf{A})^{-1}(\mathbf{I} - \mathbf{A})\,\mathbf{D}$ where $\mathbf{D}$ has ρ diagonals being -1.
- ρ is a hyperparameter;
- The entries of $\mathbf{A}$ are the trainable parameters.

scoRNN Architecture

Theorem 2

Let $\mathcal{L} = \mathcal{L}(W) : \mathbb{R}^{n \times n} \to \mathbb{R}$ *be some loss function for an RNN and* $\mathbf{W} = \mathbf{W}(\mathbf{A}) := (\mathbf{I} + \mathbf{A})^{-1} (\mathbf{I} - \mathbf{A}) \mathbf{D}$ *Then*

$$\frac{\partial \mathcal{L}}{\partial \mathbf{A}} = \mathbf{V}^T - \mathbf{V} \tag{1}$$

where $\mathbf{V} := (\mathbf{I} - \mathbf{A})^{-1} \frac{\partial \mathcal{L}}{\partial \mathbf{W}} \left(\mathbf{D} + \mathbf{W}^T\right)$,

scoRNN Architecture

Theorem 2

Let $\mathcal{L} = \mathcal{L}(W) : \mathbb{R}^{n \times n} \to \mathbb{R}$ *be some loss function for an RNN and* $\mathbf{W} = \mathbf{W}(\mathbf{A}) := (\mathbf{I} + \mathbf{A})^{-1} (\mathbf{I} - \mathbf{A}) \mathbf{D}$ *Then*

$$\frac{\partial \mathcal{L}}{\partial \mathbf{A}} = \mathbf{V}^T - \mathbf{V} \tag{1}$$

where $\mathbf{V} := (\mathbf{I} - \mathbf{A})^{-1} \frac{\partial \mathcal{L}}{\partial \mathbf{W}} \left(\mathbf{D} + \mathbf{W}^T\right)$,

Update A:

$$A^{(k+1)} = A^{(k)} - \lambda \frac{\partial \mathcal{L}}{\partial A}$$

Training of D

Real case: discrete D determined by ρ (number of -1 in D)

- ρ needs to be tuned.

Complex case: continuous $\mathbf{D} = \mathrm{diag}\{e^{i\theta_1} \cdots, e^{i\theta_n}\}$

- optimize θ_i through gradient descent

Training of D

Scaled Cayley Unitary RNN (scuRNN): train $\mathbf{D} = \mathrm{diag}\{e^{i\theta_1} \cdots, e^{i\theta_n}\}$ by optimizing with respect to $\theta = [\theta_1, \cdots, \theta_n]$.

Theorem 3

Let $\mathcal{L} = \mathcal{L}(W) : \mathbb{C}^{n\times n} \to \mathbb{R}$ be some differentiable loss function for an RNN with the recurrent weight matrix $\mathbf{W} = \mathbf{W}(\mathbf{A}, D) := (\mathbf{I} + \mathbf{A})^{-1}(\mathbf{I} - \mathbf{A})\,\mathbf{D}$. Then the gradient of $L = L(W(A, D))$ with respect to $\theta = [\theta_1, \cdots, \theta_n]$ is

$$\frac{\partial L}{\partial \theta} = 2Re\left(i\left(\left(\frac{\partial L}{\partial W}^T Z\right) \odot I\right) d\right),$$

where $d = \left[e^{i\theta_1}, e^{i\theta_2, \ldots e^{i\theta_n}}\right]^T$

Training of D

Scaled Cayley Unitary RNN (scuRNN): train $\mathbf{D} = \mathrm{diag}\{e^{i\theta_1} \cdots, e^{i\theta_n}\}$ by optimizing with respect to $\theta = [\theta_1, \cdots, \theta_n]$.

Theorem 3

Let $\mathcal{L} = \mathcal{L}(W) : \mathbb{C}^{n\times n} \to \mathbb{R}$ be some differentiable loss function for an RNN with the recurrent weight matrix $\mathbf{W} = \mathbf{W}(\mathbf{A}, D) := (\mathbf{I} + \mathbf{A})^{-1}(\mathbf{I} - \mathbf{A})\,\mathbf{D}$. Then the gradient of $L = L(W(A, D))$ with respect to $\theta = [\theta_1, \cdots, \theta_n]$ is

$$\frac{\partial L}{\partial \theta} = 2Re\left(i\left(\left(\frac{\partial L}{\partial W}^{T} Z\right) \odot I\right) d\right),$$

where $d = \left[e^{i\theta_1}, e^{i\theta_2, \ldots e^{i\theta_n}}\right]^T$

$$\theta^{(k+1)} = \theta^{(k)} - \lambda \frac{\partial \mathcal{L}}{\partial \theta}$$

Eigenvalue Normalized RNN (ENRNN)

ENRNN

- Orthogonal/Unitary RNNs $\rightarrow$ Long term dependency:
  - Unable to "forget" short term information
  - Reduces capacity

ENRNN

- Orthogonal/Unitary RNNs $\rightarrow$ Long term dependency:
  - Unable to "forget" short term information
  - Reduces capacity
- ENRNN:
  - Two states: Long term memory and short-term memory
  - Short-term memory state: Use W with $\rho(W) < 1$

ENRNN Architecture

- Version 1:

$$
\begin{cases}
h_t^{(L)} = \sigma\left(U^{(L)}x_t + W^{(L)}h_{t-1}^{(L)} + b^{(L)}\right) \\
h_t^{(S)} = \sigma\left(U^{(S)}x_t + W^{(S)}h_{t-1}^{(S)} + b^{(S)}\right) \\
y_t = V^{(L)}h_t^{(L)} + V^{(S)}h_t^{(S)} + c
\end{cases}
\tag{2}
$$

$$
W = \left[\begin{array}{c|c} W^{(L)} & \\ \hline & W^{(S)} \end{array}\right]
$$

- $W^{(L)}$ is orthogonal/unitary
- $\rho(W^{(S)}) < 1$

ENRNN Architecture

- Version 2:

$$\begin{cases} h_t^{(L)} = \sigma\left(U^{(L)}x_t + W^{(L)}h_{t-1}^{(L)} + W^{(C)}h_{t-1}^{(S)} + b^{(L)}\right) \\ h_t^{(S)} = \sigma\left(U^{(S)}x_t + W^{(S)}h_{t-1}^{(S)} + b^{(S)}\right) \\ y_t = V^{(L)}h_t^{(L)} + V^{(S)}h_t^{(S)} + c \end{cases} \tag{3}$$

$$W = \left[\begin{array}{c|c} W^{(L)} & W^{(C)} \\ \hline & W^{(S)} \end{array}\right]$$

- $W^{(L)}$ is orthogonal/unitary
- $\rho(W^{(S)}) < 1$

ENRNN Architecture

Theorem 4

With the ReLU nonlinearity, if $\|W^{(S)}\|_2 < 1$ *then*

$$\left\| \frac{\partial h_{t+\tau}^{(S)}}{\partial h_t^{(S)}} \right\| \leq \left\| W^{(S)} \right\|^{\tau} \text{ and } \left\| \frac{\partial h_{t+\tau}^{(S)}}{\partial x_t} \right\| \leq \left\| W^{(S)} \right\|^{\tau} \left\| U^{(S)} \right\|$$

ENRNN Architecture

Construction of $W^{(S)}$:

- Parameterize $W^{(S)}$ by T as

$$W^{(S)} = W^{(S)}(T) := \frac{T}{\rho(T) + \epsilon}$$

ENRNN Architecture

Construction of $W^{(S)}$:

- Parameterize $W^{(S)}$ by T as

$$W^{(S)} = W^{(S)}(T) := \frac{T}{\rho(T) + \epsilon}$$

- Gradient descent in T.

ENRNN Architecture

Theorem 5

Let $L = L(W) : \mathbb{R}^{m\times m} \to \mathbb{R}$ *be some loss function for an RNN and let* $\frac{\partial L}{\partial W} := \left[\frac{\partial L}{\partial W_{i,j}}\right] \in \mathbb{R}^{m\times m}$. *Let* W *be parameterized as* $W = \frac{T}{\rho(T)+\epsilon}$. *If* $\lambda = \alpha + i\beta$ *is a simple eigenvalue of* T *with* $|\lambda| = \rho(T)$ *and if* $Tu = \lambda u$ *and* $v^* T = \lambda v^*$, *then*

$$\frac{\partial L}{\partial T} = \frac{1}{\tilde{\rho}(T)} \left[\frac{\partial L}{\partial W} - \frac{1}{\tilde{\rho}(T)} 1_m^T \left(\frac{\partial L}{\partial W} \odot W\right) 1_m C\right]$$

where $C = \alpha \operatorname{Re}(S) + \beta \operatorname{Im}(S)$ *with* $S = \frac{\overline{v} u^T}{v^* u} \in \mathbb{C}^{m\times m}$, $1_m \in \mathbb{R}^m$ *is a vector consisting of all ones,* $\tilde{\rho}(T) = \rho(T) + \epsilon$.

ENRNN Architecture

- Selecting λ or $\overline{\lambda}$ results in same derivative due to conjugation.
- Repeat eigenvalues unlikely.
- Begin normalization once $\rho(W) > 1$

Gradient Analysis

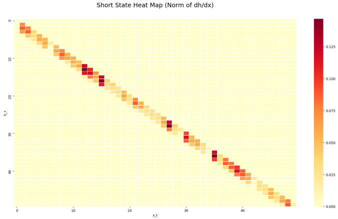


Figure: Gradient norms $\|\frac{\partial h_{\tau}^{(S)}}{\partial x_t}\|$

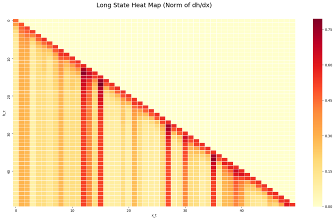


Figure: Gradient norms $\|\frac{\partial h_{\tau}^{(L)}}{\partial x_t}\|$

Experiments

Adding Problem

0.58	0.23	0.84	0.06	0.71		0.35	0.22	0.63	0.14	0.97
0	0	1	0	0		0	1	0	0	0

Figure: The goal of the machine is to output the sum of the entries marked by one, in this case 0.84+0.22 = 1.06

- Two sequences concurrently, each length T
  - First sequence: $\mathcal{U}[0,1)$
  - Second sequence: All zeros except a 1 located uniformly in $[1, \frac{T}{2})$ and a second 1 uniformly in $[\frac{T}{2}, T)$
- Goal: Sum the two entries marked by 1s

Adding Problem

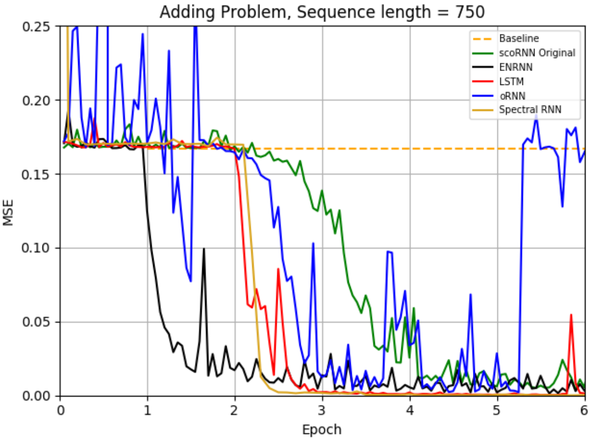


Figure: Test set MSE on the adding problem.

Copying Problem

- Sequence length: $T + 20$
- First ten uniformly sampled from $1 - 8$
- Marker 9 placed ten timesteps from the end
- All other entries 0
- Goal: Output zeros until the 9 then output the first ten elements from the beginning of the sequence.

Copying Problem

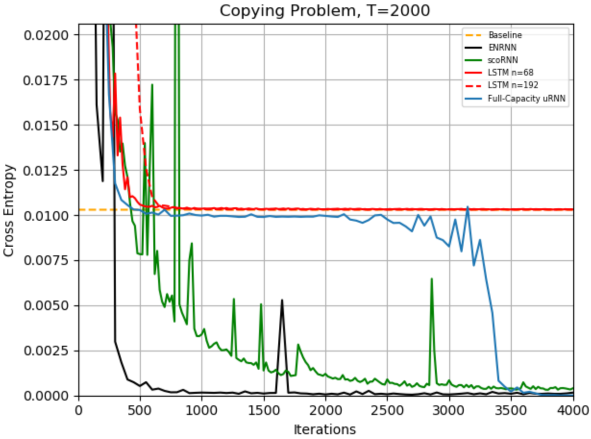


Figure: Cross entropy of each machine on the copying problem.

TIMIT Speech Dataset

- TIMIT dataset - 3,696 training, 400 validation, and 192 testing speech recordings.
- Goal: Predict log-magnitudes of the Fourier amplitudes at frame $t + 1$.

TIMIT Speech Dataset

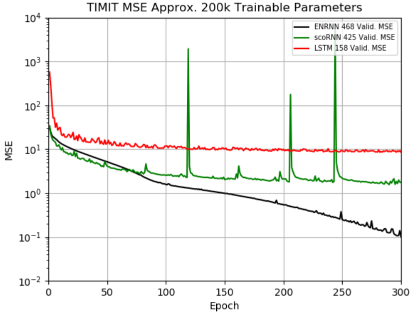


Figure: Validation set MSE for TIMIT

Table: TIMIT: Best validation MSE after 300 epochs

Model	n	#Params	Valid. MSE	Test. MSE
ENRNN	374/94	$\approx$ 200k	**0.13**	**0.13**
scoRNN	425	$\approx$ 200k	1.56	1.52
LSTM	158	$\approx$ 200k	8.53	8.27
LSTM	468	$\approx$ 1200k	5.60	5.42

Model	N	SegSNR (dB)	STOI	PESQ
ENRNN	374/94	**4.84**	**0.83**	**2.75**
scoRNN	425	4.55	0.82	2.72
LSTM	158	4.00	0.79	2.51
LSTM	468	4.82	0.81	2.75

Character PTB

- Character PTB dataset - 10k words, 50 characters
- 5102k training, 400k validation, 450k testing characters
- Goal: Predict next character in the sequence

Character PTB

Table: Best testing MSE in BPC after 20 epochs.

Model	n	# Param	Valid. BPC	Test BPC
ENRNN	310/720	$\approx$ 1016k	**1.475**	**1.429**
LSTM	350	$\approx$ 1016k	1.506	1.461
GRU	415	-	-	1.601*
EURNN	2048	-	-	1.715*
GORU	512	-	-	1.623*
oRNN	512	$\approx$ 183k	1.73**	1.68**
nnRNN	1024	$\approx$ 1320k	-	1.47***
LSTM	1030	$\approx$ 8600k	1.447	1.408

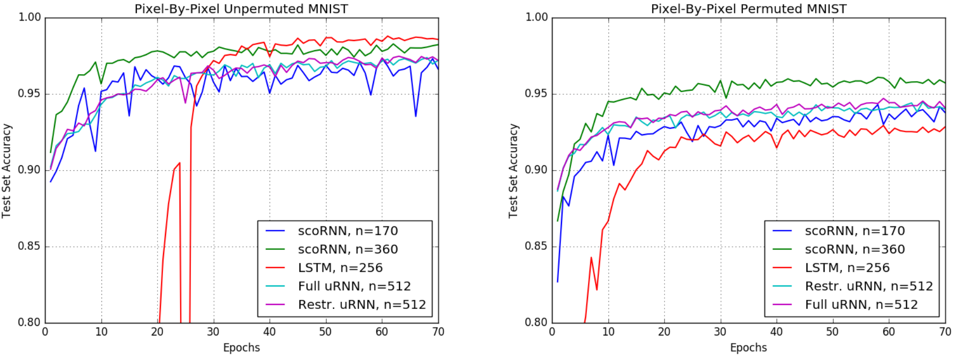
MNIST Problem

- Goal: Classify 28x28 pixel images of handwritten digits (0-9)
- Pixel fed into RNN sequentially - single pixel sequence length of 784
- Unpermuted and fixed permutation
- 55,000 training images and 10,000 testing images

MNIST Problem

Model	n	# parameters	MNIST Test Accuracy	Permuted MNIST Test Accuracy
scoRNN	170	$\approx$ 16k	0.973	0.943
scoRNN	360	$\approx$ 69k	0.983	0.962
LSTM	128	$\approx$ 68k	0.987	0.920
LSTM	256	$\approx$ 270k	0.989	0.929
Restricted-capacity uRNN	512	$\approx$ 16k	0.976	0.945
Full-capacity uRNN	116	$\approx$ 16k	0.947	0.925
Full-capacity uRNN	512	$\approx$ 270k	0.974	0.947

MNIST Problem



Concluding Remarks

Concluding Remarks

- Some difficulties in deep learning pertain to numerical issues
- Robust parametrization improves performance
- Orthogonal RNNs $\rightarrow$ accumulation of long term memory
- enRNN $\rightarrow$ short term memory

References:

1. K. Helfrich and D. Willmott and Q. Ye, Orthogonal Recurrent Neural Networks with Scaled Cayley Transform, ICML 2018.
2. K.D. Maduranga, K. Helfrich, and Q. Ye, Complex Unitary Recurrent Neural Networks using Scaled Cayley Transform, AAAI 2019.
3. K. Helfrich and Q. Ye, Eigenvalue Normalized Recurrent Neural Networks for Short Term Memory, AAAI 2020.

The End