Learning Unitary Operators with Help From $\mathfrak{u}(n)$

@__hylandSL

ETH zürich

Stephanie L. Hyland, Gunnar Rätsch ETH Zürich, Weill Cornell Medicine

Introduction

- Training deep neural networks can be difficult due to the problem of **vanishing/exploding gradients**
- In recurrent neural networks, repeated application of the state evolution operator can be responsible
- Using a **norm-preserving** operator can help!
  - focus on unitary/orthogonal matrices
- Gradient updates in general break unitarity: use **parametrisation** closed under additive updates
- Idea: use **Lie group-Lie algebra correspondence** to define parametrisation

Parametrisation with $\mathfrak{u}(n)$

- Unitary *n x n* matrices form a continuous group, *U(n)* (a **Lie group**), closed under matrix multiplication
- The tangent space to a Lie group at the identity is its **Lie algebra**: a vector space closed under *addition*
  - *U(n)*'s Lie algebra is $\mathfrak{u}(n)$, the space of *n x n* skew-Hermitian matrices ($A^\dagger = -A$)
- The **exponential map** associates elements of the Lie group and Lie algebra:
  - for matrix groups like *U(n)*, the map is the matrix exponential (`expm`)
  - $L \in \mathfrak{u}(n) \Rightarrow \exp(L) \in U(n)$
  - as *U(n)* is compact and connected, the map is **surjective** (not true for the orthogonal group)
- Given a basis $\{ T_i \}$ of $\mathfrak{u}(n)$, $L \in \mathfrak{u}(n)$ is defined by its coefficients $\lambda_i \in \mathbb{R}$
- Thus *U* is parametrised by these n^2 coefficients:

$$U(\lambda_1, \ldots, \lambda_{n^2}) = \exp\left(\sum_{i=1}^{n^2} \lambda_i T_i \right)$$

Thus: we do gradient descent in the **Lie algebra**

- Choice of basis $\{ T_i \}$ is arbitrary: we choose them to be **sparse** (1 or 2 non-zero elements):
  - *n* diagonal imaginary
  - *n(n-1)/2* symmetric imaginary
  - *n(n-1)/2* skew-symmetric real

Learning the Operator

- Discard the rest of RNN machinery (for *now*!)
- Simple **supervised learning** task:
  - sample random ground truth *U*, many random **x**
  - $\mathbf{y} = U\mathbf{x} + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2)$
  - minimise mean $\|\hat{U}\mathbf{x} - \mathbf{y}\|^2$ over batches
- Approaches for learning *U*:
  1. `projection`: after gradient update, project *U* back to closest unitary matrix
  2. `composition`: *U* is a product of parametrisable unitary operators; (with *7n* free parameters) from *Arjovsky, Shah & Bengio (ICLR 2016)*
  3. `lie algebra`: U is defined by the n^2 coefficients of an element of the Lie algebra (my method)

General Unitary RNN

- Define (general) unitary recurrent neural network, with:

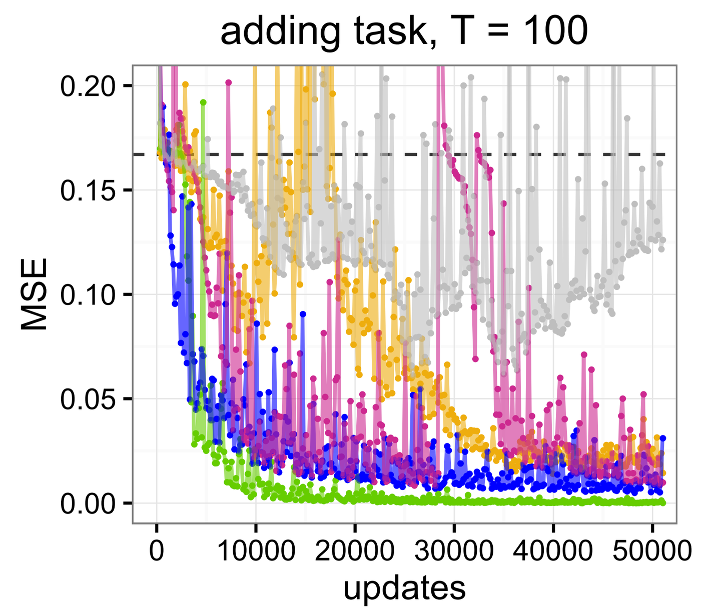
$$\mathbf{h}_t = f\left(U\mathbf{h}_{t-1} + V\mathbf{x}_t + \mathbf{b}\right)$$

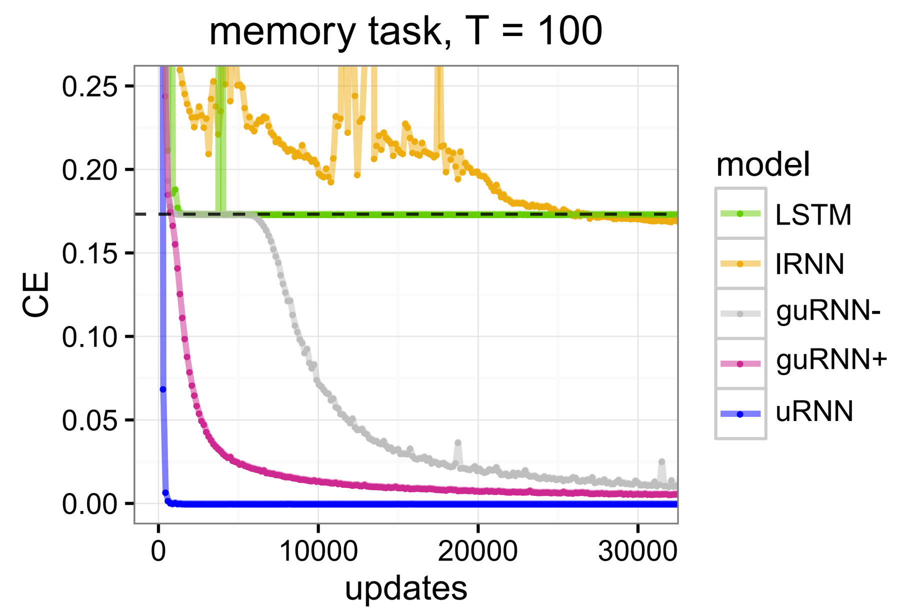
- f is a nonlinearity, **h** is hidden state, U is a **unitary** matrix

Tasks: (classic long-memory RNN problems)

Adding problem**:** add two numbers from long sequence, minimise mean-squared error on result

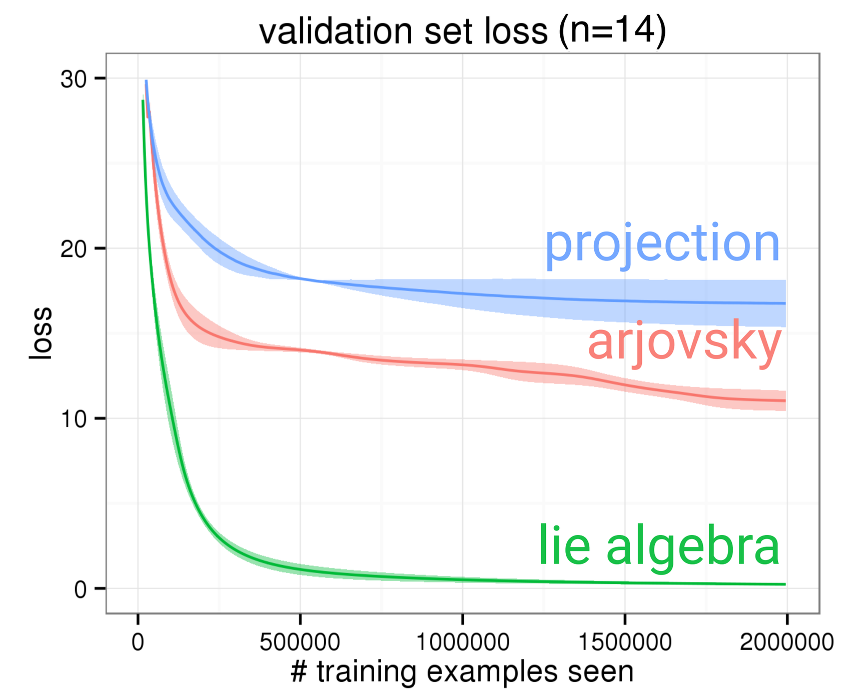
Memory problem: 'remember' a sequence for a long time, minimise categorical cross-entropy over whole sequence





n	true	projection	arjovsky	lie algebra	rand
3	$6.004 \pm 0.005 \times 10^{-4}$	8 ± 1	$\mathbf{6.005 \pm 0.003 \times 10^{-4}}$	$\mathbf{6.003 \pm 0.003 \times 10^{-4}}$	12.5 ± 0.4
6	~ 0.001	15 ± 1	0.09 ± 0.01	$\mathbf{0.03 \pm 0.01}$	24 ± 1
8	~ 0.002	14 ± 1	1.17 ± 0.06	$\mathbf{0.014 \pm 0.006}$	31.6 ± 0.6
14	~ 0.003	24 ± 4	10.8 ± 0.3	$\mathbf{0.07 \pm 0.02}$	52 ± 1
20	~ 0.004	38 ± 3	29.0 ± 0.5	$\mathbf{0.47 \pm 0.03}$	81 ± 2

Table: Mean squared l_2 distance between true **y** and predicted **y**



Additional experiments:

- **restricted**: restrict `lie algebra` to have *7n* learnable parameters
- **basis**: change the basis used by `lie algebra` with random uniform matrix (e.g. between -5 and 5)

✨Further Questions✨

- **Geometry:**
  - Casting the learning problem directly as optimisation on the Lie group (manifold)
  - How does exponential map change cost surface?
- **Deep learning**:
  - understanding relative performance of models
  - nonlinearity has a large role in vanishing gradients - consider it **alongside** transition operator

Conclusions

- Parametrisation using the **Lie algebra** enables the learning of arbitrary unitary operators with simple gradient descent
- We've used our parametrisation to define a **general unitary RNN** and demonstrated it on standard tasks
- The parametrisation of Arjovksy *et al.* struggles to learn matrices for *n > 7*, but still works well in an RNN setting, somehow

- **Code**: https://github.com/corcra/uRNN
- **Paper**: accepted at AAAI-17, available on arXiv: