Generating 3 Qubit Quantum Circuits with Neural Networks

Journal Club, 20 Apr 2017

Zhang Chuheng, CQI, IIIS, THU

Preface

- Quantum Machine Learning[1]
  - Machine learning algorithm find application in understanding and controlling quantum systems
    - Eg. Solving quantum many-body system by ML
  - Quantum computational devices promote the performance of machine learning algorithms
    - Eg. Quantum version of SVM[2]

[1] Biamonte, Jacob, et al. "Quantum Machine Learning." *arXiv preprint arXiv:1611.09347* (2016).
[2] Rebentrost, Patrick, Masoud Mohseni, and Seth Lloyd. "Quantum support vector machine for big data classification." *Physical review letters* 113.13 (2014): 130503.

Abstract

Generating 3 qubit quantum circuits with neural networks

Michael Swaddle,[1,*] Lyle Noakes,[2] Liam Salter,[3] Harry Smallbone,[3] and Jingbo Wang[1]

[1]*School of Physics, The University of Western Australia, Crawley 6009, Australia*
[2]*Faculty of Engineering, Computing and Mathematics, The University of Western Australia, Crawley 6009, Australia*
[3]*The University of Western Australia, Crawley 6009, Australia*

(Dated: April 3, 2017)

A new method for compiling quantum algorithms is proposed and tested for a three qubit system. The proposed method is to decompose a a unitary matrix U, into a product of simpler U_j via a neural network. These U_j can then be decomposed into product of known quantum gates. Key to the effectiveness of this approach is the restriction of the set of training data generated to paths which approximate minimal normal subRiemannian geodesics, as this removes unnecessary redundancy and ensures the products are unique. The two neural networks are shown to work effectively, each individually returning low loss values on validation data after relatively short training periods. The two networks are able to return coefficients that are sufficiently close to the true coefficient values to validate this method as an approach for generating quantum circuits. There is scope for more work in scaling this approach for larger quantum systems.

[1] Swaddle, Michael, et al. "Generating 3 qubit quantum circuits with neural networks." arXiv preprint arXiv:1703.10743 (2017).

Overview

- Problem Description
- Proposed Neural Network Solution
  - Reformulation of the problem
  - Generation of training data
  - Neural network design
- Results

Problem Description

- Statement I: Given an arbitrary unitary operation $U \in SU(2^n)$, find an optimal quantum circuit approximates U.

- Statement II: Suppose U is generated by some time dependent Hamiltonian $\frac{dU}{dt} = -iH(t)U(t)$, $U(0) = I$, $U(t_F) = U$, find the optimal control function $H(t)$ for synthesizing U. $H(t)$ should be easy-to-implement, finite number of quantum gates.

Problem Description

- Remark: the minimal geodesic distance between the identity operation I and U is essentially equivalent to the number of gates required to synthesize U.[1] (Geometric view)

- Geodesic: given the initial point and initial velocity, the rest of geodesic is determined by geodesic equation $\frac{d^2x^j}{dt^2} + \Gamma_{kl}^j \frac{dx^k}{dt}\frac{dx^l}{dt} = 0, \text{with } \Gamma_{kl}^j$ Christoffel symbol representing local geometry.

- Like Newtonian and Lagrangian Formulations in Mechanics / geometric optics and Fermat's principle in optics

[1] Nielsen, Michael A. "A geometric approach to quantum circuit lower bounds." arXiv preprint quant-ph/0502070 (2005).

Problem Description

- Remark: $H(t)$ could be found via finding minimal geodesics of the Riemannian geometry from I to U.

- Remark: Computing geodesics requires one to solve a boundary value problem in a high dimensional space, which is hard.

Problem Description

- Remark: using one- and two-qubit gates, it is possible to well approximate U (well approximate the geodesics)[1].

- Remark: One- and two-qubit gates are easy to implement.

[1] Nielsen, Michael A., et al. "Quantum computation as geometry." *Science* 311.5764 (2006): 1133-1135.

Proposed Solution

- Use neural network (NN), supervised learning scheme

- find $U \approx \boldsymbol{E}(\boldsymbol{c}) = \exp(c_1^1\tau_1)\cdots\exp(c_m^1\tau_m)$ ← segment

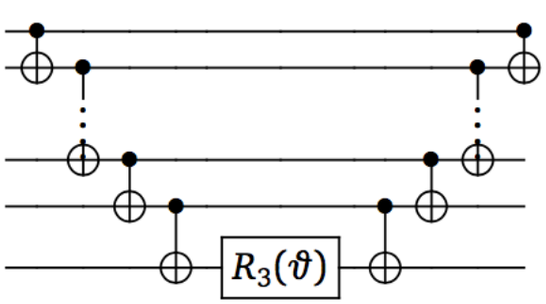
$$\cdots\cdots\cdots$$

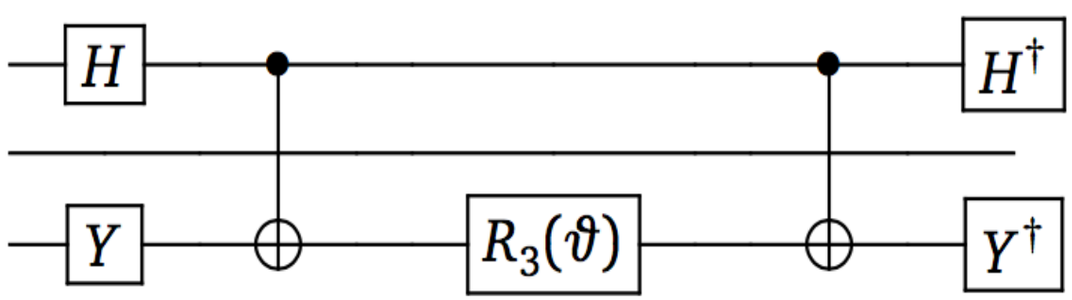
$$\exp(c_1^N\tau_1)\cdots\exp(c_m^N\tau_m)$$

- $\tau_i(i = 1,\cdots,m)$ is basis of $\Delta \subset su(2^n) = span\{\frac{i}{\sqrt{2^n}}\sigma_i^j, \frac{i}{\sqrt{2^n}}\sigma_i^k\sigma_j^l\}$, where σ_i^j represents the n fold Kronecker product $\sigma_i^j = I \otimes \cdots \otimes \sigma_i \otimes \cdots \otimes I$, with σ_i on the j-th slot.

Proposed Solution

- Quantum circuit construction of $\exp(i\theta\tau)$.
  - When $\tau = I \otimes \cdots \otimes \sigma_i \otimes \cdots \otimes I$, it is just a single qubit gate $\exp(i\,\theta\,\sigma_i)$, it can be constructed by $H\,R_3(\theta)H^+$ (for $i = 1$), $\mathrm{Y}\,R_3(\theta)Y^+$ (for $i = 2$) or $R_3(\theta)$ (for $i = 3$)
  - When $\tau = I \otimes \cdots \sigma_i \cdots \sigma_j \cdots \otimes I$, it's a two qubit gate $\exp(i\,\theta\,\sigma_i \otimes \sigma_j)$. It can also be generated similarly, for example, $\exp(i\theta\sigma_1 \otimes I \otimes \sigma_2)$





Proposed Solution

- Supervised learning scheme
- Training data $\mathfrak{D} = \{(x_i, y_i)\}$, find parameters $\theta = \arg\min loss(y_i, f(x_i, \theta))\,|_{(x_i,y_i)\in\mathfrak{D}}$
- Test data $\mathcal{T} = \{(x_i, y_i)\}$, not seen by the model in the training phase, is used to test the efficiency of the model.

Proposed Solution

- $U \approx \boldsymbol{E}(\boldsymbol{c}) = \exp(c_1^1\tau_1)\cdots\exp(c_m^1\tau_m)$ ← segment

$$\cdots\cdots\cdots$$

$$\exp(c_1^N\tau_1)\cdots\exp(c_m^N\tau_m)$$

- Global decomposition: factor U into segments.

$$U \approx U_1U_2\cdots U_j\cdots U_N$$

- Local decomposition: factor U_j into sequence of one- and two-qubit gates.

$$U_j \approx \exp\left(c_1^j\tau_1\right)\cdots\exp\left(c_m^j\tau_m\right)$$

Proposed Solution

- Generation of training data for global decomposition
- There are infinitely many ways to factor U, training data should be generated in a unique way.
- These paths should be chosen to be minimal normal subRiemannian geodesics. By Pontryagin Maximum Principle, obtain geodesic equation.

$$\dot{x} = ux$$

- $$\dot{\Lambda} = [\Lambda, u] \implies \dot{x} = proj_{\Delta}(x\Lambda_0 x^{+})x$$

$u = proj_{\Delta}(\Lambda)$

- $x\colon [0,1] \to SU(2^n), \Lambda\colon [0,1] \to su(2^n), u\colon [0,1] \to \Delta \subset su(2^n)$

Proposed Solution

- Generation of training data for global decomposition
  - Randomly select a Λ_0
  - Iteratively repeat $x(t_{j+1}) = U_j x(t_j)$, $U_j = \exp(h\, proj_\Delta(x_j \Lambda_0 x_j^+))$
  - Obtain a training data $\{U, \{U_j\}\}$, where $U = U_1 U_2 \cdots U_j \cdots U_N$
- Distance between U and I approximates the complexity to implement U. In order U be well approximated by the limited sequence, they bounded the norm $||proj_\Delta(\Lambda_0)|| = ||u_0||$, which determines distance.

Proposed Solution

- Generation of training data for local decomposition
  - $\mathrm{U_j} = \exp\left(c_1^j \tau_1\right) \cdots \exp\left(c_m^j \tau_m\right)$
  - Simply randomly generates data $\{c_1, \cdots, c_m\}$ ($c_i \in [-h\pi, +h\pi]$, i.e. should be small) and construct $\mathrm{U_j} = \exp\left(c_1^j \tau_1\right) \cdots \exp\left(c_m^j \tau_m\right)$

Proposed Solution

- $U \approx \boldsymbol{E}(\boldsymbol{c}) = \exp(c_1^1 \tau_1) \cdots \exp(c_m^1 \tau_m)$ ← segment

$$\cdots\cdots\cdots$$

$$\exp(c_1^N \tau_1) \cdots \exp(c_m^N \tau_m)$$

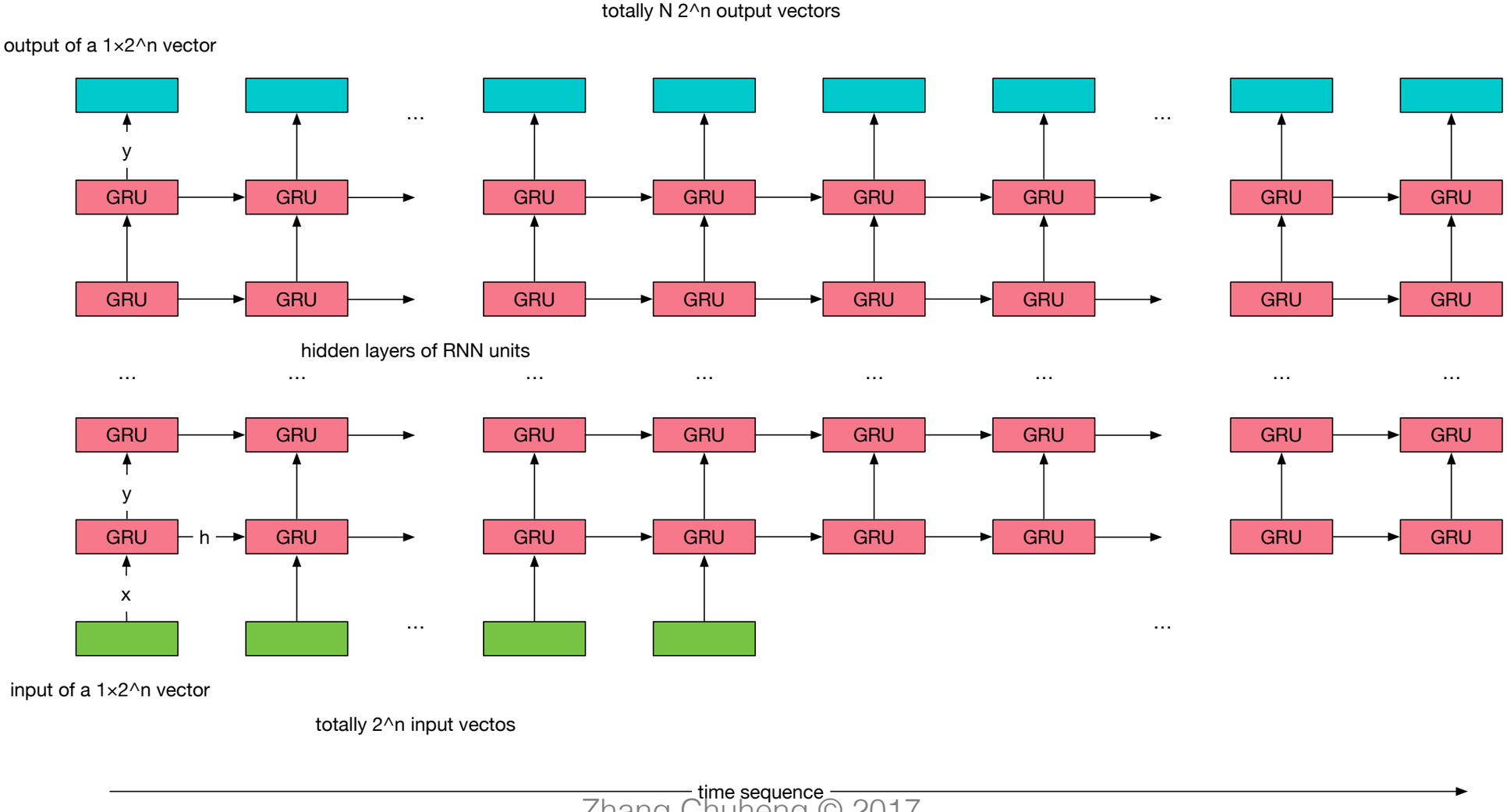
- Global decomposition: Use GRU (Gated Recurrent Unit) network to factor U into segments. $U \approx U_1 U_2 \cdots U_j \cdots U_N$

- Input: $U_{2^n \times 2^n}$ - 2^n real vectors representing rows

- Output: $U_1 U_2 \cdots U_j \cdots U_N$ - $N2^n$ row vectors

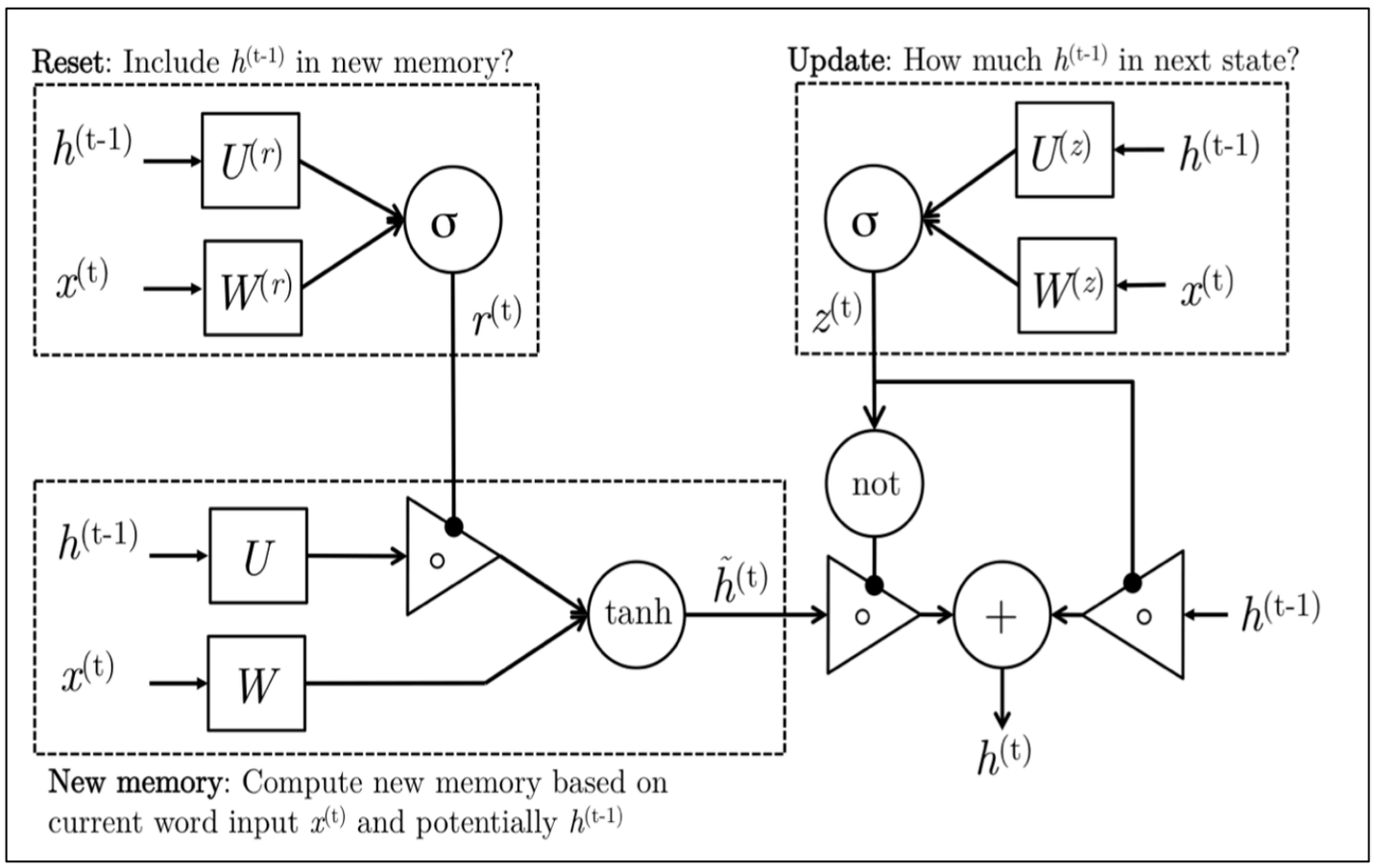
Proposed Solution

- GRU network is a kind of recurrent neural network (RNN)



Proposed Solution

- GRU



$$z^{(t)} = \sigma(W^{(z)}x^{(t)} + U^{(z)}h^{(t-1)}) \quad \text{(Update gate)}$$
$$r^{(t)} = \sigma(W^{(r)}x^{(t)} + U^{(r)}h^{(t-1)}) \quad \text{(Reset gate)}$$
$$\tilde{h}^{(t)} = \tanh(r^{(t)} \circ Uh^{(t-1)} + Wx^{(t)}) \quad \text{(New memory)}$$
$$h^{(t)} = (1 - z^{(t)}) \circ \tilde{h}^{(t)} + z^{(t)} \circ h^{(t-1)} \quad \text{(Hidden state)}$$

$$\leftarrow x^{(t)}, h^{(t-1)}$$
$$\rightarrow h^{(t)}, y^{(t)} = f(W^{(y)}h^{(t)})$$

Image from http://cs224n.stanford.edu

Proposed Solution

- $U \approx \boldsymbol{E}(\boldsymbol{c}) = \exp(c_1^1\tau_1)\cdots\exp(c_m^1\tau_m)$ ← segment

$$\cdots\cdots\cdots$$

$$\exp(c_1^N\tau_1)\cdots\exp(c_m^N\tau_m)$$

- Local decomposition: Use a four-layer fully connected network to decompose $U_j \approx \exp(c_1^j\tau_1)\cdots\exp(c_m^j\tau_m)$

- Input: $U_{j_{2^n\times 2^n}}$ - $2^n\times 2^n$ numbers

- Output: coefficients $(c_1^j,\cdots,c_m^j)$

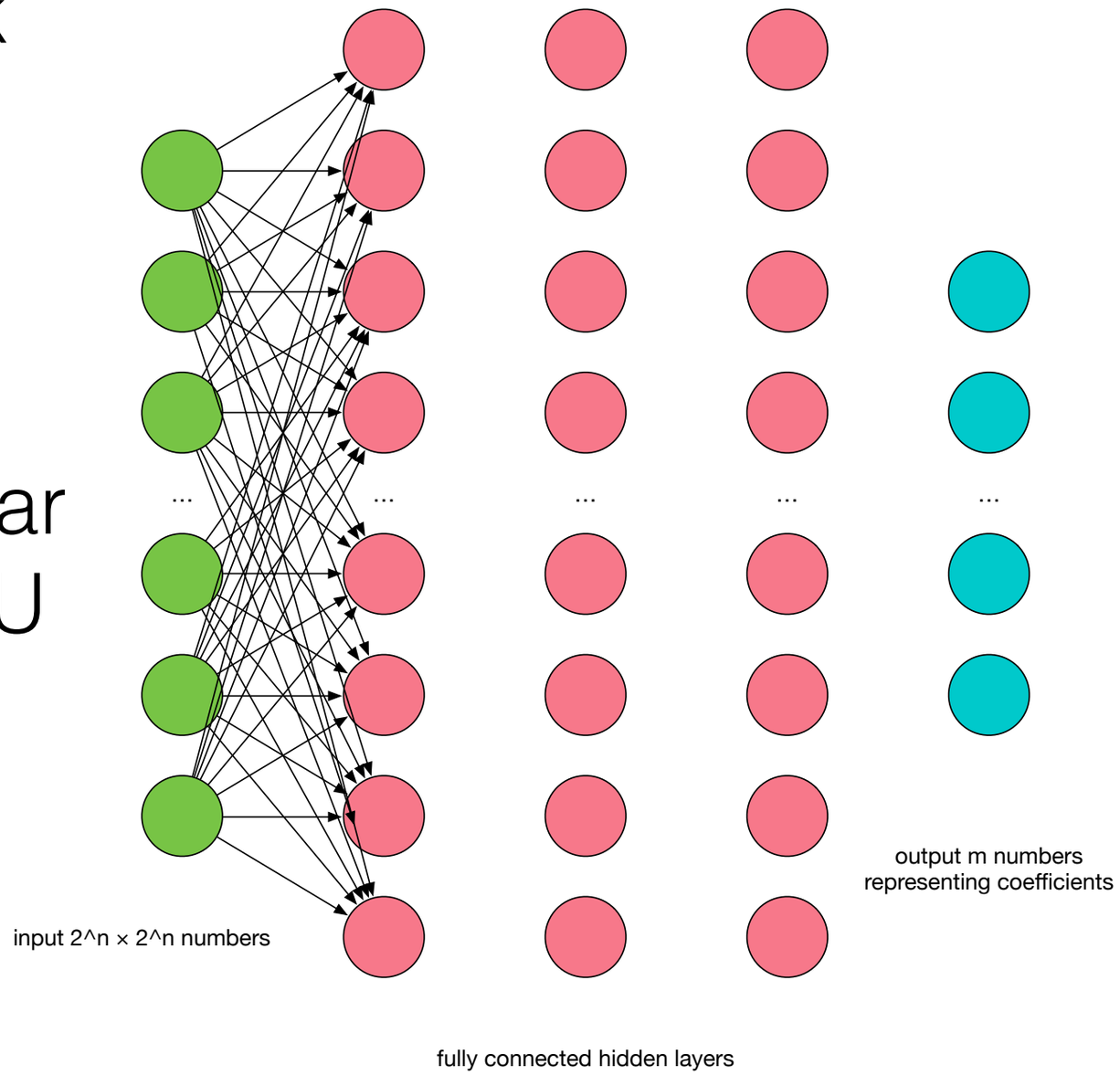
Proposed Solution

- Fully connect network

For each layer

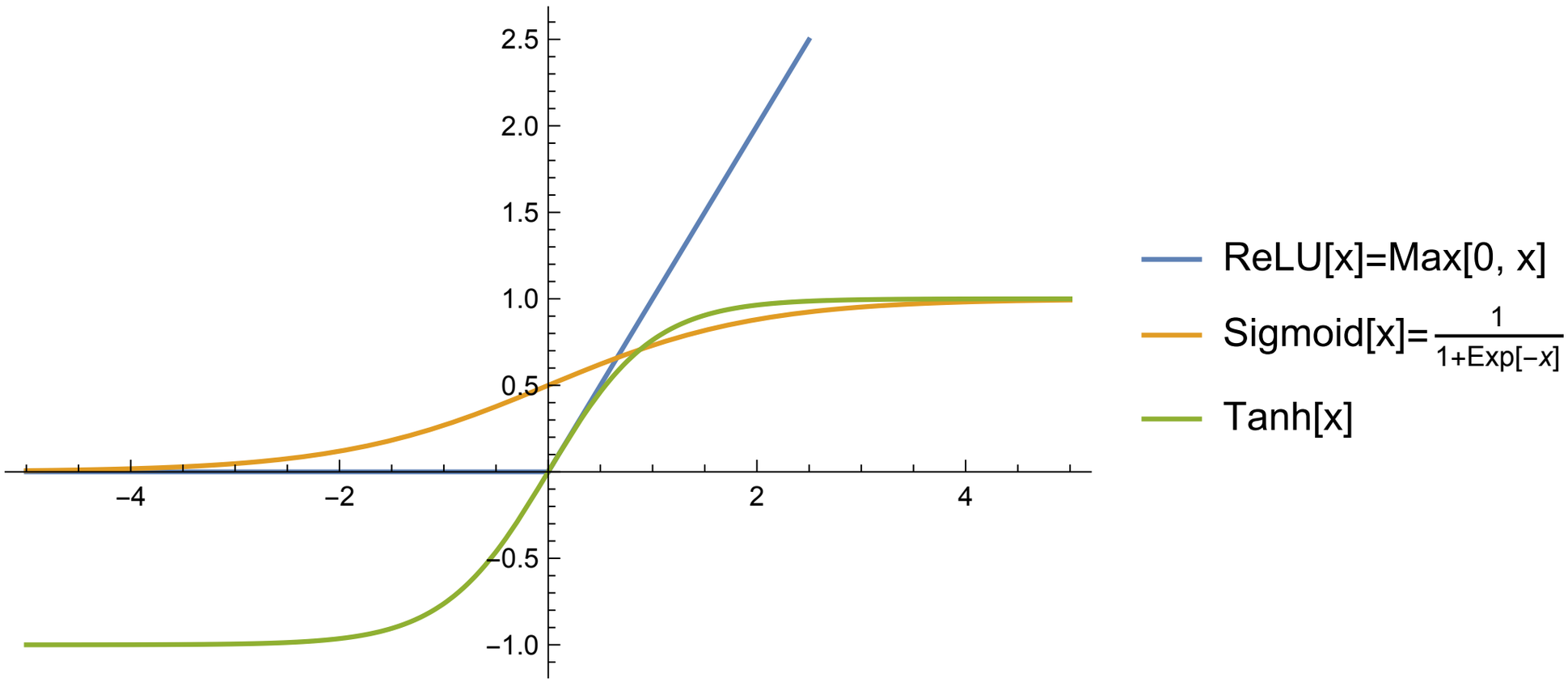
$$y_i = f(W_i x_i)$$

Where $f(\cdot)$ is a non-linear function – they use ReLU here



Proposed Solution

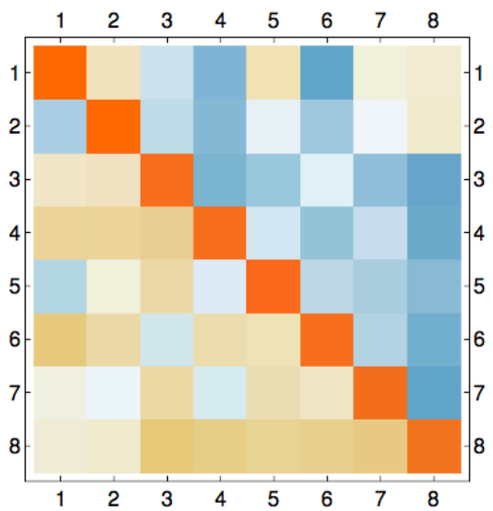
- Non-linear functions



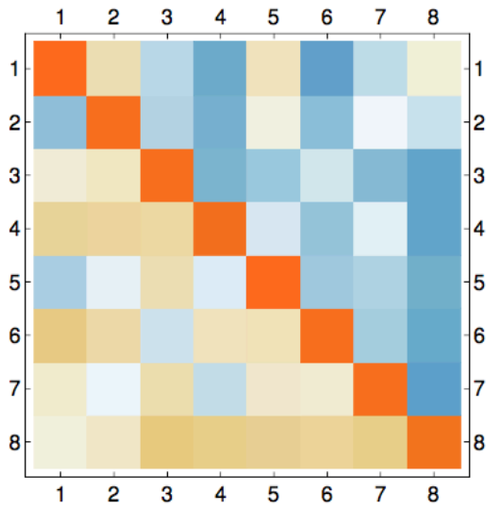
Proposed Solution

- Problem:
  - Not well scaled, also exponentially increasing
- Future:
  - U is always sparse, make full use sparsity

Results

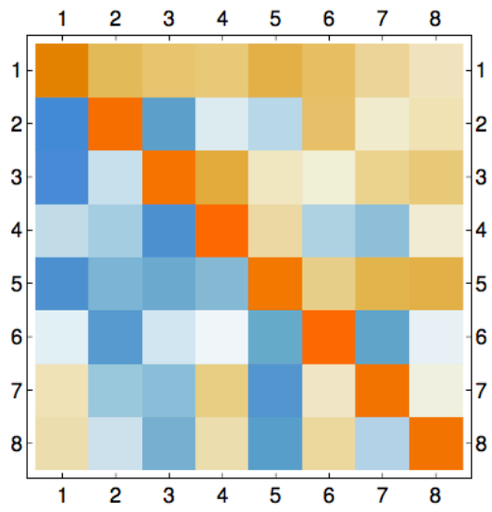


(a) Real components of a U_j generated by the NN.

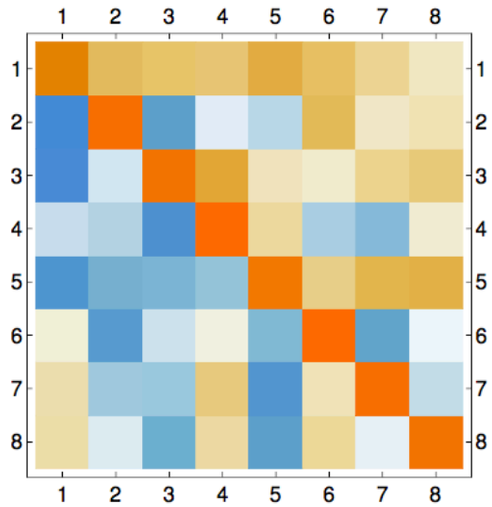


(b) The respective known real components of a U_j from the validation dataset.

Global Decomposition



(a) Real components of a U_j generated by the NN.



(b) The respective known real components of a U_j from the validation dataset.

Local Decomposition

Thanks for your attention