Convergence of RNN under Unitary Weights

Hoang Chu

April 29, 2024

Introduction

RNNs tend to face a problem of vanishing / exploding gradients: after a number of iterations, the weights approach 0 (vanishing) or ∞ (exploding).

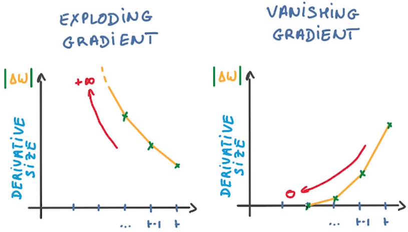


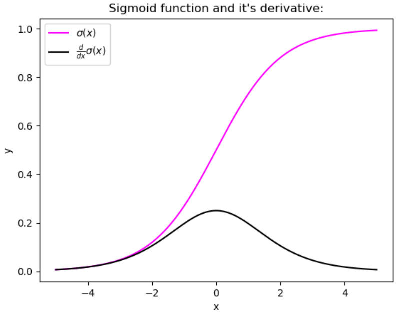
Figure: Problem Visualization

Why?

Backpropagation Formula:

$$\frac{\partial \text{State}_{\text{now}}}{\partial \text{State}_{\text{prev}}} = \prod_{\text{now} \geq i > \text{prev}} (\text{Weight})^T \, diag\left(\sigma'(\text{State}_i)\right)$$

1. The gradients of Weight are controlled by eigenvalues.[2]
2. Choice of Activation Function:



Past Approach

In 2015 [1], 3 researchers devised a solution to solve **BOTH** issues by restricting Weight to be always a Unitary Matrix during backpropagation.

Unitary Evolution Recurrent Neural Networks

Martin Arjovsky * MARJOVSKY@DC.UBA.AR
Amar Shah * AS793@CAM.AC.UK
Yoshua Bengio
Universidad de Buenos Aires, University of Cambridge,
Université de Montréal. Yoshua Bengio is a CIFAR Senior Fellow.

*Indicates first authors. Ordering determined by coin flip.

$$W \in \mathbb{R} \xrightarrow{\text{unitary transform}} X \in \mathbb{C} : X \cdot \text{conj-transposed}(X) = I$$

Past Approach: Advantages

1. Preserve norms: unitary matrices preserve norm of vector it multiplies.
   E.g: given $X = [[0,1],[1,0]], v = [3,4], Xv = [4,3]$:
   $$Norm(X) = Norm(Xv) = 5$$
   $\rightarrow$ Addressed exploding weights.
2. Control eigenvalues: eigenvalues of a unitary matrix have norm $= 1$.
   E.g: given $X = [[0,1],[1,0]]$:
   eigenvalues are 1 and -1, all of which having norm $= 1$.
   $\rightarrow$ Addressed vanishing weights.
3. Easy to Compute Inverse

Past Approach: Challenges

The authors kept using Sigmoid as an activation function, which does not guarantee that the weight matrices will remain unitary after each update, especially when the learning rate is large.

E.g: $X = [[0, 1], [1, 0]]$, learning rate $= 10$.

main.py

Save Run

```python
import numpy as np

def sigmoid(x):
    return 1 / (1 + np.exp(-x))

def is_unitary(matrix):
    return np.allclose(np.eye(len(matrix)), \
            matrix @ matrix.T.conj())

U = np.array([[0, 1], [1, 0]]); lr = 10

i = 0
while is_unitary(U):
    U = U + lr * sigmoid(U)
    i += 1

print(f"Weighted matrix stopped being unitary after {i}
    iterations")
```

Output

Clear

```
Weighted matrix stopped being unitary after 1 iterations

=== Code Execution Successful ===
```

New approach on updating Weight

- By definition, we want $WW* = I$
- $I = \exp\{0_{\text{matrix}}\}$ (due to power series $\exp\{A\} = 1 + A + \frac{1}{2!}A^2$)
- For any matrix X, $W = \exp(X)$ is always unitary.
- So I want to pick exp() as my activation function.
- This means, for any matrix X, I want $W = \exp(X)$ is always unitary.
- After some reverse algebra, I found this that X must be a skew-symmetric (or anti-symmetric) matrix.
- New update rule:

$$B \leftarrow B + \alpha \cdot \left((\text{Gradient(F)})\, W^T - W\, (\text{Gradient(F)})^T \right)$$

$$W \leftarrow \exp(B)$$

where α is the learning rate and F is the activation function.

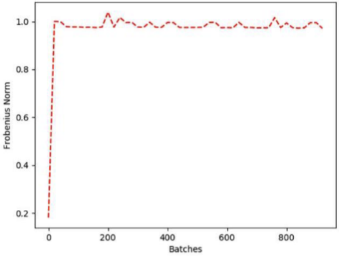
Proof of Unitarity

- What I found basically to make B to be anti-symmetric $(A^T = -A^{-1})$, which has these additional properties:
  1. $B + B^* = 0$
  2. $\exp(-B)^* = \exp(B^*)$
- Compute WW^*:

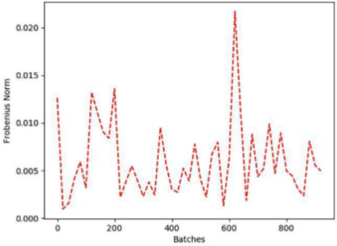
$$\begin{aligned} WW^* &= \exp(B)\exp^{-1}(B)^* \\ &= \exp(B)\exp(-B)^* \\ &= \exp(B)\exp(B^*) \\ &= \exp(B + B^*) \\ &= \exp(0) = I \end{aligned}$$

New Approach: Generate Next Word Implementation

- Training Average Time: 1 hour 52 minutes.



a. Convergence By Batches (uRNN)



c. Convergence By Batches (RNN)

- Accuracy Table:

Acc. (%) / Input length	100	200	400
uRNN	38.12	34.15	15.23
RNN	92.01	80.45	56.73

Table: Accuracy of RNN and uRNN at different input lengths

References

[1] Martın Arjovsky, Amar Shah, and Yoshua Bengio. "Unitary Evolution Recurrent Neural Networks". In: *CoRR* abs/1511.06464 (2015). URL: http://arxiv.org/abs/1511.06464.

[2] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. "Understanding the exploding gradient problem". In: *ArXiv* abs/1211.5063 (2012).