Seq2seq for Polynomial Roots Finding

Hoang Duong Joan Bruna

ST212B Deep Learning Final Project, 2016

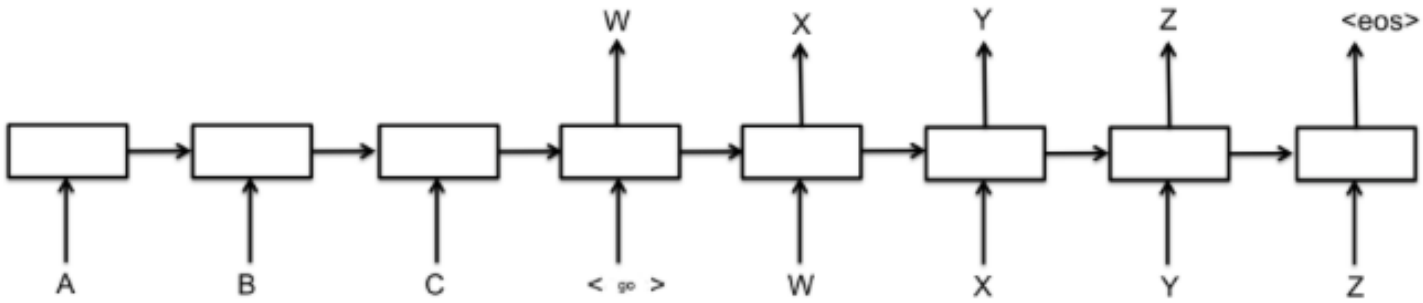
Modelling Sequences

1. Sequence to Sequence models, for variable sequence input size, and variable output size (hard for traditional machine learning models, e.g. Random Forest)
2. Train on short sequences, and hopefully generalize on longer sequences
3. Current state: can solve a few algorithmic tasks: addition, multiplication, sorting, reversing,

Outline

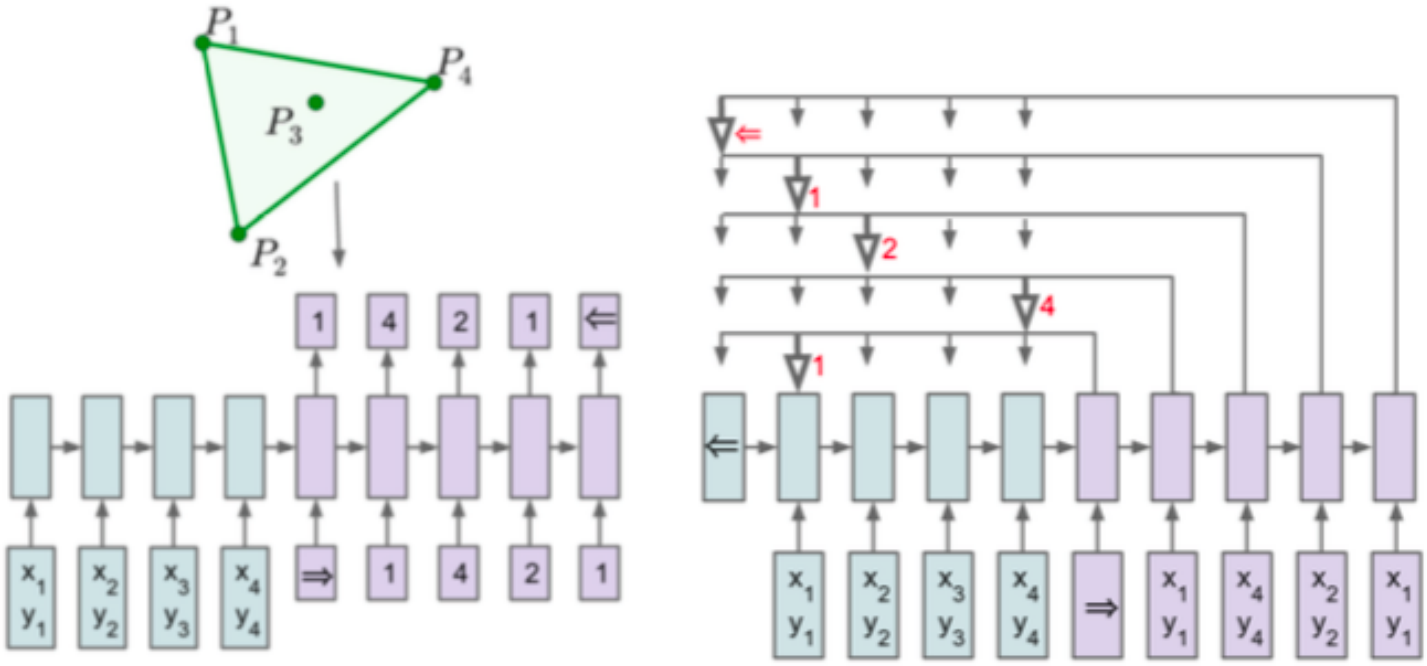
1. Literature Overview: Neural Encoder Decoder, Pointer Network, Neural GPU
2. Describing Problems: Polynomial Tasks
3. Proposed Model: Recursive Root Finding Model

Neural Encoder Decoder



1. Machine Translation: French to English
2. Can do attention mechanism
3. Also use word embedding

Pointer Network

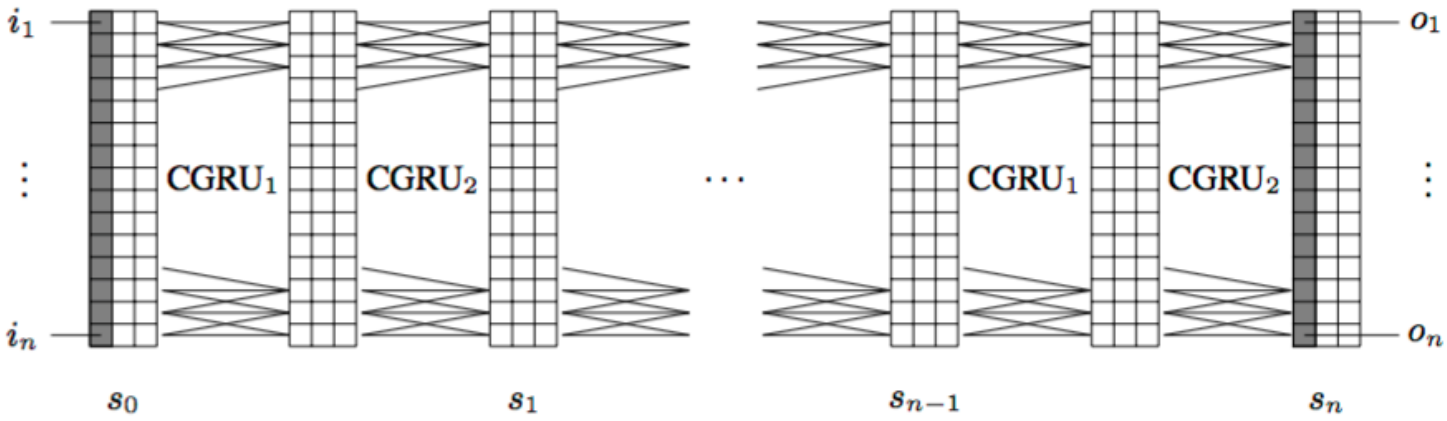


(a) Sequence-to-Sequence

(b) Ptr-Net

1. Tasks: convex hull, Travelling Salesman Problem, Delaunay Triangulations
2. Target must be a subset of inputs

Neural GPU



1. Tasks: long binary addition, long multiplication, copying, reversing
2. Can potentially work for our problem

Polynomial Tasks

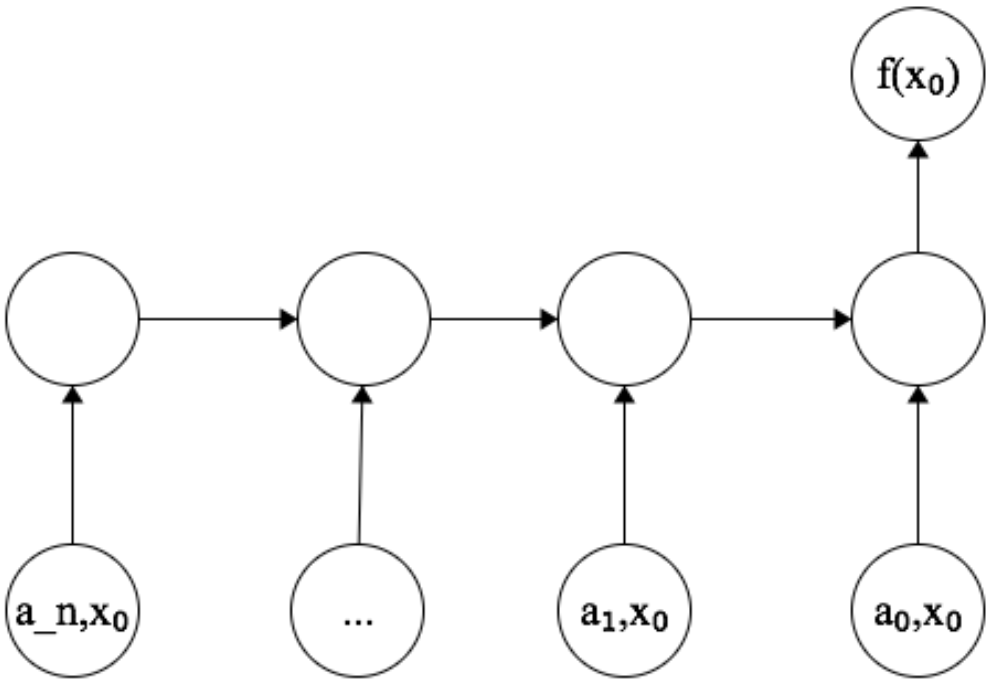
The easier problems: given coefficients $c_0, c_1, ..., c_d \sim^{i.i.d} \mathscr{U}[-1, 1]$, denote p(x) the corresponding polynomial, and an $x_0 \sim \mathscr{U}[-1, 1]$, find a mapping $\mathbb{R}^{d+1} \to \mathbb{R}$

1. $p(x_0)$
2. $p'(x_0)$
3. coefficients and remainder of $p(x)/(x - x_0)$

The harder problem: we generate the roots first $z_0, z_1, ..., z_{d-1} \sim^{i.i.d} \mathscr{U}[-1, 1]$, denote the coefficients of the corresponding monic polynomial $c_0, c_1, ..., c_{d-1}$, find a mapping $\mathbb{R}^d \to \mathbb{R}^d$ from the coefficients to roots

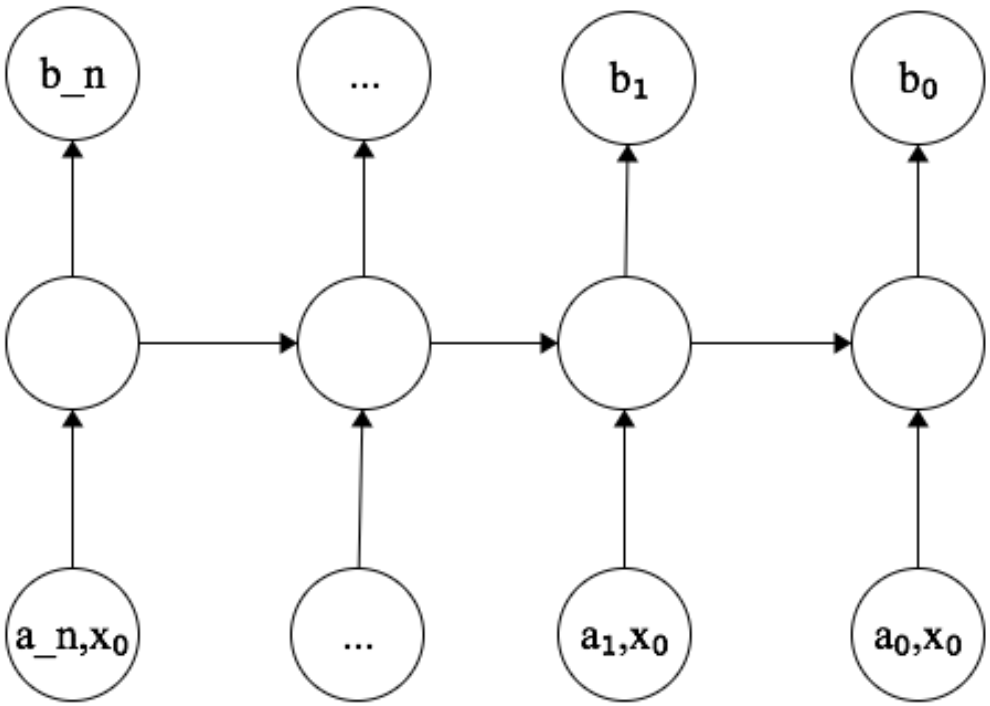
Easy Polynomial Tasks

We use simple LSTM. Train on polynomials degree 5, 10, 15; test on polynomials degree 10, 15, 20, 25, 35, 50. For polynomial evaluation:

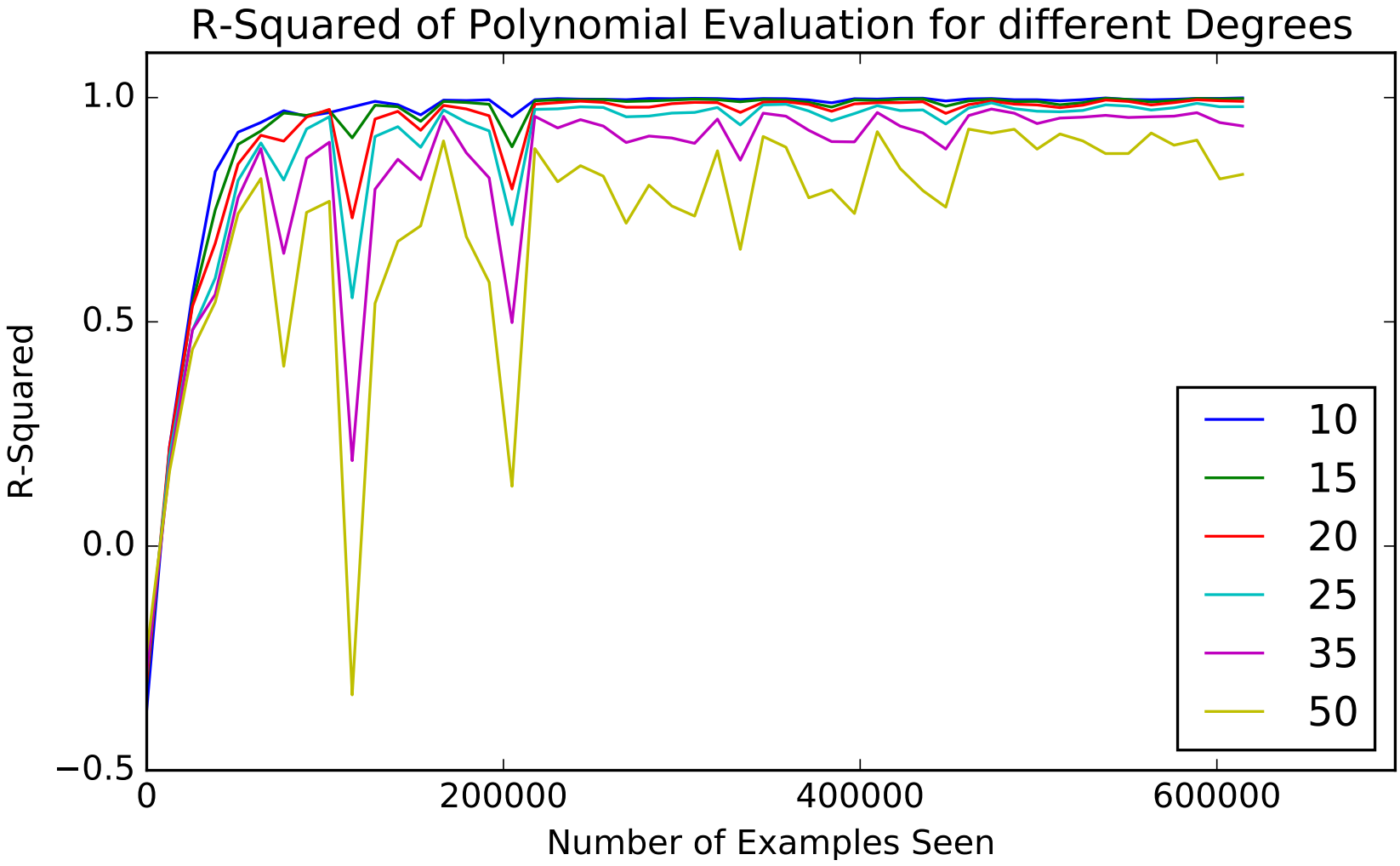


Easy Polynomial Tasks

We use simple LSTM. Train on polynomials degree 5, 10, 15; test on polynomials degree 10, 15, 20, 25, 35, 50. For polynomial division by $x - z_0$:



Result


R-Squared of Polynomial Evaluation for different Degrees
R-Squared
1.0
0.5
0.0
−0.5
0
200000
400000
600000
Number of Examples Seen
10
15
20
25
35
50

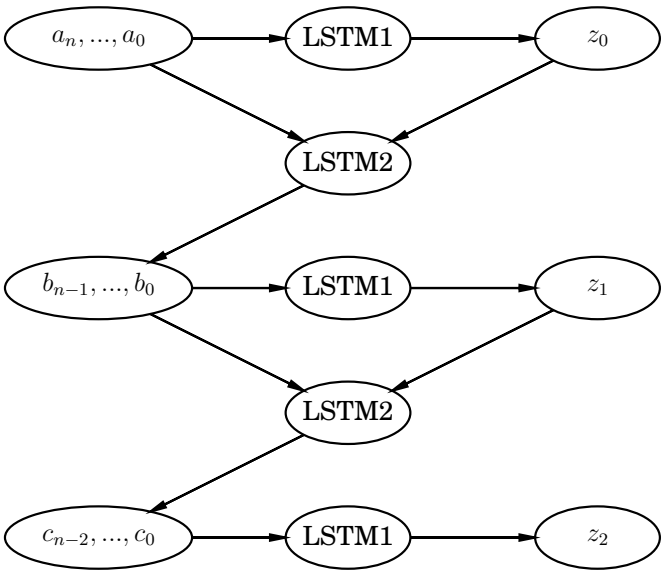
Taxonomy

For the harder problem of finding roots, if we are training on one degree, and testing on the same degree, traditional ML models perform very well.

	Easy (Evaluation)	Hard (Finding roots)
Fixed Length	definitely not ML	Random Forest, FNN
Variable Length	LSTM	??

We try Neural Encoder Decoder on this problem, and it didn't work (i.e. no generalization to higher degrees)

Recursive Polynomial Roots Model



1. Iteratively solve for one root, factorize the polynomial, and solve for next root.
2. Use one LSTM1 for solving one root, and one LSTM2 for factorization

Recursive Polynomial Roots Model

1. $z_0 = LSTM1(a_n, ..., a_0)$
2. $b_{n-1}, ..., b_0 = LSTM2((a_n, z_0), (a_{n-1}, z_0), ..., (a_1, z_0))$
3. $L(z, \hat{z}) = (z_0 - \hat{z}_0)^2 + \frac{1}{2}(z_1 - \hat{z}_1)^2 + \frac{1}{3}(z_2 - \hat{z}_2)^2$
4. Currently learning end-to-end, i.e. only feeding in coefficients, and roots, without any intermediate root (for predicting next root), or coefficients (b_n) to guide the model.

Result


R-Squared of Finding 3 Smallest Roots
R-Squared - Minimum out of 3
1.0
0.5
0.0
−0.5
−1.0
0
1
2
3
4
1e7
Number of Examples Seen
Deg17
Deg19
Deg22
Deg26
Deg30

Future

1. How to improve generalization? Guiding with the true labels?
2. Better component than LSTM?
3. What are other similar problems?