A Max-Affine Spline Perspective of Recurrent Neural Networks (RNNs)

Zichao (Jack) Wang, Randall Balestriero, Richard Baraniuk

Department of Electrical and Computer Engineering, Rice University

Motivation

- RNNs are widely used in practice, but their inner workings remain a mystery
- Recent work provides new insights into deep feedforward networks using max-affine spline operators (MASOs)
- We focus on RNNs with piecewise affine and convex nonlinearities such as ReLU and study RNNs from a function approximation perspective

Background: RNNs

- RNN formula (ReLU nonlinearity, identity weight initialization):
  \[
  h^{(t)} = \text{ReLU}(Ux^{(t)} + Vh^{(t-1)} + a)
  \]
  \[
  \hat{y} = \text{softmax}(Wh^T + b)
  \]
- We focus on sequence classification tasks (e.g., semantic analysis)

Core Result: RNNs as Piecewise Affine Operators

- A recurrent neural network cell is a max-affine spline operator (MASO)
- A complete recurrent neural network is a composition of MASOs (one for each cell)
  - generalizes to multi-layer recurrent networks

Affine Spline Operator

Input Space Partitioning in RNNs

- RNNs develop input sequence partitioning through time
  - geometric perspective of RNNs (vector quantization, k-means, etc.)
  - insights into RNN dynamics
- Example: partition progress through time in an RNN of MNIST images

Input sequence \( X \) to output \( f_{RNN}(X) \) mapping is a region-dependent affine transform

\[
 f_{RNN}(X) = A_{RNN}[X]X + b_{RNN}[X]
\]

Example: visualization for MNIST. From left to right: input image; each row of \( A_{RNN}[X] \) from 0 to 9 and its inner product with \( X \).

<table>
<thead>
<tr>
<th>k</th>
<th>MNIST, raw pixels</th>
<th>MNIST, VQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.950</td>
<td>0.977</td>
</tr>
<tr>
<td>2</td>
<td>0.936</td>
<td>0.974</td>
</tr>
<tr>
<td>5</td>
<td>0.951</td>
<td>0.977</td>
</tr>
<tr>
<td>10</td>
<td>0.939</td>
<td>0.975</td>
</tr>
</tbody>
</table>

Input Space Partitioning in RNNs

- Adding noise to the initial hidden state links to exploding gradient problem in RNNs and improves RNN performance.

Background: MASOs

- A spline function approximation consists of
  - a partition of the independent variable (input space) into \( R \) regions
  - a (simple) local mapping \((\alpha_r, \beta_r)\) on each region of the partition
- The elegant spline: Max-affine splines automatically find the optimal partition while estimating the local mappings
  \[
  z(x) = \max_{r=1,...,R} (\alpha_r x + \beta_r)
  \]
- Example: to approximate \( e^x \), learn \( \alpha_r \) and \( \beta_r \) by minimizing
  \[
  \| e^x - z(x) \|
  \]
- Concatenation of multiple max-affine splines, we get a max-affine spline operator (MASO)

Improving RNNs Using Noisy Initial Hidden State

- Injecting Gaussian noise into the initial hidden state corresponds to explicit regularization
  - links to exploding gradient problem
  - improves classification accuracy

<table>
<thead>
<tr>
<th>Model</th>
<th>Dataset 1</th>
<th>Dataset 2</th>
<th>Dataset 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN, 1 layer</td>
<td>0.970</td>
<td>0.891</td>
<td>0.871</td>
</tr>
<tr>
<td>RNN, 1 layer, noise</td>
<td>0.981</td>
<td>0.922</td>
<td>0.873</td>
</tr>
<tr>
<td>((\sigma_\alpha = 0.1))</td>
<td>((\sigma_\beta = 0.1))</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RNN, 2 layer</td>
<td>0.969</td>
<td>0.873</td>
<td>0.884</td>
</tr>
<tr>
<td>RNN, 2 layer, noise</td>
<td>0.987</td>
<td>0.927</td>
<td>0.888</td>
</tr>
<tr>
<td>((\sigma_\alpha = 0.5))</td>
<td>((\sigma_\beta = 0.005))</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example: improved classification accuracy in the partition space (vector quantization space) using K-Nearest Neighbors

RNNs as Matched Filter Bank

- Row \( c \) of \( A_{RNN}[X] \) is a matched filter for class \( c \) that is applied to \( X \): largest inner product wins
- Example: visualization for MNIST. From left to right: input image; each row of \( A_{RNN}[X] \) (from 0 to 9) and its inner product with \( X \).