Study of Neural Network Size Requirements for Approximating Functions Relevant to HEP

Jessica Stietzel
Kevin Lannon

Data Intensive Scientific Computing REU
University of Notre Dame
Deep Learning

- Neural networks as function approximator
- Learns from map of inputs and outputs
- Outputs given by

\[ y_j = h \left( \sum_{i=0}^{2} A_{ji} x_i + a_j \right) \]

\[ z_k = \sum_{j=0}^{2} B_{kj} y_j + b_k \]

\[ h(x) = \begin{cases} 0 & x \leq 0 \\ x & x > 0 \end{cases} \]
Deep Learning in HEP

Inspiration

- Machine learning approaches are already used to solve classification problems in HEP
- “Searching for Exotic Particles in High-Energy Physics with Deep Learning” (P. Baldi et al., 2014)

Questions

- How big does the network have to be?
  - This effects how effectively we survey hyperparameter space
  - Too big – overfitting
  - Too small – can’t fit
Deep Learning Study

- Examine basic functions
  - Vector to Vector
  - Vector to Vector$^2$
  - Vector to |Vector|$^2$
- Used toy model to generate reasonable parameters for momentum vectors
  - Inputs standardized for training

- Network training
  - GPUs
  - Keras with Theano
  - 5 trials (trained on 1 million samples, validated on 10,000)
  - MSE loss function
  - Adam optimizer with default parameters
  - Batch size: 1000    Epochs: 5000
  - Saved model from best epoch
Vector to Vector

Mapping a vector onto itself

The solution needs to satisfy…

\[ AB = I \quad Ba + b = 0 \]

Obvious solution:

\[
A = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
\end{bmatrix} \quad a = \begin{bmatrix}
-x_{0,\text{min}} \\
-x_{1,\text{min}} \\
-x_{2,\text{min}} \\
\end{bmatrix} \quad B = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
\end{bmatrix} \quad b = \begin{bmatrix}
+x_{0,\text{min}} \\
+x_{1,\text{min}} \\
+x_{2,\text{min}} \\
\end{bmatrix}
\]

What the NN training actually found (one trial):

\[
A = \begin{bmatrix}
0.711 & 0.279 & 0.266 \\
0.023 & 0.494 & -0.102 \\
-0.091 & 0.354 & 0.471 \\
\end{bmatrix} \quad a = \begin{bmatrix}
1.474 \\
1.929 \\
1.448 \\
\end{bmatrix} \quad B = \begin{bmatrix}
1.313 & -0.182 & -0.782 \\
-0.009 & 1.754 & 0.386 \\
0.260 & -1.354 & 1.682 \\
\end{bmatrix} \quad b = \begin{bmatrix}
-2.294 \\
-1.154 \\
-2.203 \\
\end{bmatrix}
\]

NN finds a correct solution even if it is not the simplest solution (fewest non-zero weights).
In networks with few nodes, we can see the segments used to approximate the function.
Vector to Vector$^2$

As the number of nodes increases, the curve becomes smoother. This indicates a better approximation of the function $V^2$. 
Vector to Vector²

- Zero crossings correspond to the segments used to approximate $V^2$
  - Divided among the 3 variables (not always evenly)
  - Suspect this is from ReLU case where all inputs < 0 and gradient becomes 0.
  - Initial weights also play a role

25 Nodes

<table>
<thead>
<tr>
<th>inProf2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entries</td>
</tr>
<tr>
<td>Mean x</td>
</tr>
<tr>
<td>Mean y</td>
</tr>
<tr>
<td>RMS x</td>
</tr>
<tr>
<td>RMS y</td>
</tr>
</tbody>
</table>
Vector to $|\text{Vector}|^2$

- Intuitive Strategy
  - Use the best networks from $V$ to $V^2$
    - One layer with 500 nodes
  - Add a summation layer that adds the $V^2$ outputs
    - One layer with 3 nodes
Vector to $|\text{Vector}|^2$

- **Intuitive Strategy**
  - Use the best networks from $V$ to $V^2$
  - Add a summation layer that adds the $V^2$ outputs

- **Alternative Strategies**
  - Tried many, for example, add a layer, but keep the total nodes constant
Vector to $|\text{Vector}|^2$

Intuitive

Validation Plot for Two Layer Network with 500 Nodes and 3 Nodes

Alternative

Validation Plot for Two Layer Network with 251 Nodes
Future Work

Ask questions similar to Vector to $|\text{Vector}|^2$ experiment, but this time taking multiple four vectors as inputs.

Intuitive

Unstructured

Decoder

Two Body
Conclusions

- NN’s find solutions to problems that aren’t always the most intuitive
- It is important to understand the fundamentals of NN behavior before applying them to more complicated problems
  - We can develop better intuition for creating networks that work for HEP analysis
- Do we guide the network towards the solutions we’d like or give it free reign?
Questions?