NPL – Learning through Weak Supervision

J. Rampersad, N. Kushman

Microsoft Research Cambridge

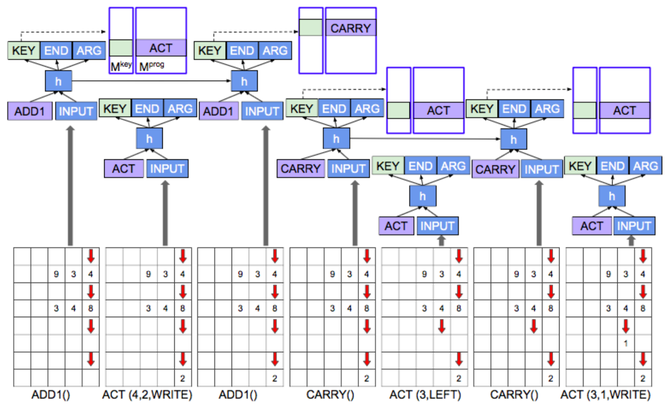
UNIVERSITY OF CAMBRIDGE

Overview

The Neural Program Lattice endows a modified LSTM network with a hierarchy of callable functions. Each function dictates the model's input at the subsequent time-step, determining its interaction with the given environment. The NPL describes a **framework to enable inference** of these latent programs from an observation sequence of a task being completed. Some strong supervision in the form of full program execution traces is also needed.

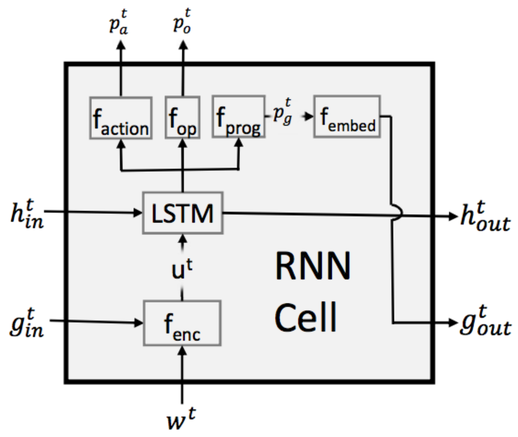
The external hierarchy of learnable programs has three key advantages over standard recurrent neural networks:

1. Facilitates **generalisation** through repeated use of sub-programs, avoids *catastrophic forgetting*.
2. Provides a flexible way to **increase model complexity** by learning new sub-programs.
3. Reduces need for the RNN hidden layer to maintain **long-term dependencies** on observations, instead learns routing between function calls.



Components of the NPL

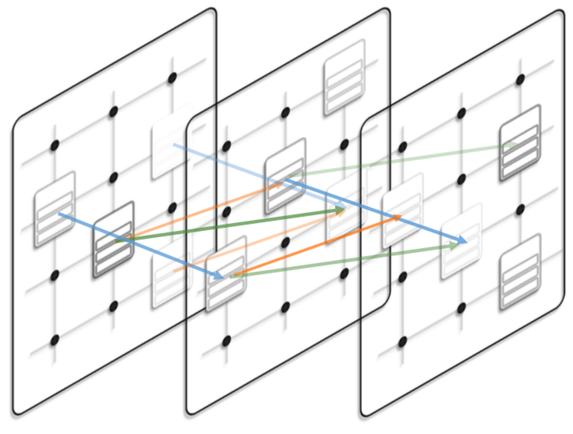
The LSTM controller has several learnable **encoders and decoders** that are tailored to suit the environment (such as a convolutional layer for images). A decoder outputs a program key for content based addressing of the program embeddings which are themselves learnable through back-propagation. Separation of the encoders from the inference core allows the same model to execute entirely different tasks such as bubble-sort, addition and image manipulation.



Inference under Weak Supervision

The Neural Program Lattice enables learning from **mostly** weak supervision (elementary operations) with the need for some strong samples (full-execution traces). We cannot obtain the true likelihood by marginalising over all possible execution traces due to **intractability**. At the other extreme considering a single merged state approximation would yield noisy gradients. Instead a lattice is formed whereby execution paths are grouped by **elementary operation index** and **stack depth**. A recursive variable $y_i^{t,l}$ quantifies the probability that the network has correctly called i elementary operations, is at depth l in the hierarchy at a given time-step t. The **loss** is defined as the negative log probability of paths that correctly execute all elementary operations and are returning from the hierarchy.

$$\mathcal{L}(\theta) = log(\sum_{t<T} p_{a,I}^{t,0}(\texttt{POP}) y_I^{t,0})$$



Extension to Purely Weak Supervision

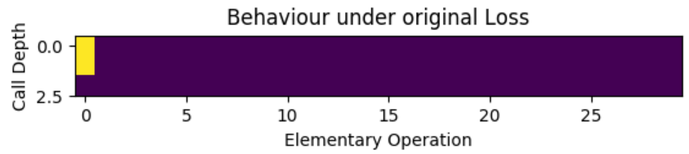
In this work we aim to entirely remove the need for strong supervision, greatly increasing the applicability of the model. Currently, learning can only take place with a small amount of strong supervision as a gradient is only produced once there is a non-zero probability of correctly generating the I elementary operations. We introduce a local loss term to enable learning prior to sufficient probability mass reaching the terminal node.

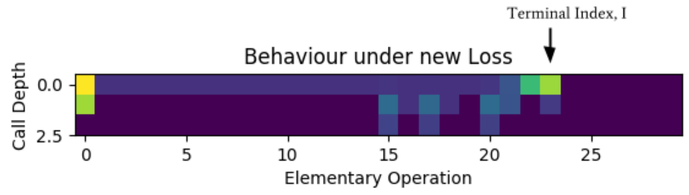
$$\bar{y}_i^{t+1,l} = p(\texttt{OP})_{a,i-1}^{t,l} p_{o,i-1}^{t,l}(\lambda_o^i) y_{i-1}^{t,l} + p(\texttt{POP})_{a,i}^{t,l+1} y_i^{t,l+1} + p(\texttt{PUSH})_{a,i}^{t,l-1} y_i^{t,l-1}$$

With q_i defined as the probability of calling the correct elementary operation, conditional on having called preceding elementary operations.

$$q_i = \sum_{l,t} \bar{y}_{i,l}^t p_a^{t,l,i}(\texttt{OP}) p_o^{t,l,i}(\lambda_o^i)$$

$$\mathcal{L}(\theta) = log \prod_{i \in I} q_i$$





References

1. D. Tarlow, A L. Gaunt, M Brockschmidt, N Kushman, "Neural Program Lattices" ICLR 2017
2. S Reed N Freitas, "Neural Programmer Interpreters" ICLR 2016