Breaking the Activation Function Bottleneck

Through Adaptive Parameterization

Sebastian Flennerhag[1,2], **Hujun Yin**[1,2], **John Keane**[1] **& Mark Elliot**[1]

[1]University of Manchester [2]The Alan Turing Institute

Summary

The activation function bottleneck: neural networks are non-linear only by virtue of a simple element-wise activation function.

Our solution: adaptively parameterize the affine pre-activation transformation, conditional on the input at hand.

Results:

- our adaptive feed-forward layer increases model capacity substantially
- our adaptive LSTM advances the state of the art for PTB and WT2, using fewer parameters and converging in less than half as many iterations

The Problem

The feed-forward layer, $f(\mathbf{x}) = \phi(W\,\mathbf{x} + \mathbf{b})$, can be written as a composition of linear maps

$$f(\mathbf{x}) = (G \circ A)(\mathbf{x}),$$

where $A(\mathbf{x}) = W\,\mathbf{x} + \mathbf{b}$ and $G = \mathrm{diag}(\phi(\mathbf{a}) / \mathbf{a})$.

A is fixed for all $\mathbf{x}$, so G encodes the network's adaptation to non-linear patterns. The adaptive capacity of G is limited since:

- ϕ is typically very close to linear over large parts of its domain
- G has no parameters of its own, so A must encode both global and local information

Our Solution

We break the activation function bottleneck by generalizing the feed-forward layer in two ways:

- replace G with a parameterized diagonal matrix $D = \mathrm{diag}(\pi(\mathbf{x}))$
- allow q blocks $(D \circ A)$ in each layer:

$$\tilde{f} = \phi\left(D^{(q)} W^{(q-1)} \cdots W^{(1)} D^{(1)}\,\mathbf{x} + D^{(0)}\,\mathbf{b}\right)$$

π can be any parameterized mapping. We train it jointly with the parameters of $\tilde{f}$.

Variants of Adaptive Layers

Singular Value Adaptation:

$$\tilde{f}(\mathbf{x}) = W^{(2)} D W^{(1)}\,\mathbf{x}\,.$$

Learns a family of composite linear maps diagonalizable in an adaptation basis, along with a policy for selecting singular values.

IO-adaptation:

$$\tilde{f}(\mathbf{x}) = D^{(2)} W D^{(1)}\,\mathbf{x}\,.$$

Learns to adapt the mean and variance of sub-matrices in a default parameter matrix. Can represent any other adaptation policy.

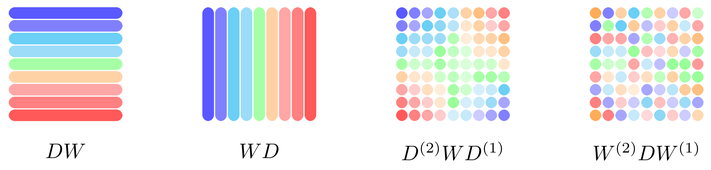

Figure 1: Adaptation policies. *Left:* output adaptation shifts the mean of each row in W; *center left:* input adaptation shifts the mean of each column; *center right:* IO-adaptation shifts mean and variance across sub-matrices; *Right:* SVA scales singular values.

Demonstration: Feed-forward Regression

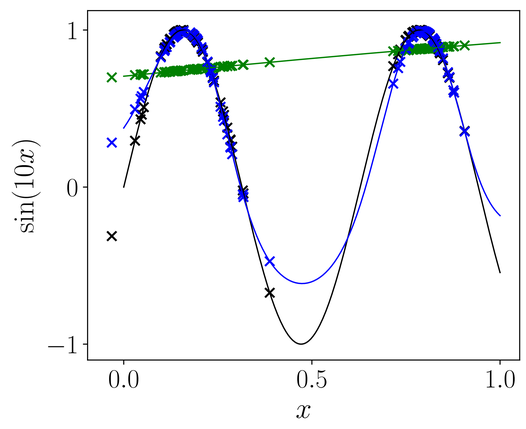

Figure 2: Predict y given x, with $x \sim 0.5\mathcal{N}(0.2, 0.7) + 0.5\mathcal{N}(0.8, 0.4)$ and $y = \sin(10x)$ (*black*). Baseline: 2-layer feed-forward network. Adaptive model: 2-layer feed-forward network, first layer an SVA policy. *Blue:* with adaptive feed-forward layer; *green:* static baseline.

Demonstration: Adaptive LSTM

Replace all affine transformations $\mathbf{u}_t = W\,\mathbf{x}_t + V\,\mathbf{h}_{t-1} + \mathbf{b}$ with IO-adaptation:

$$\mathbf{u}_t^s = D_t^{(s,x,2)} W D_t^{(s,x,1)}\,\mathbf{x}_t + D_t^{(s,h,2)} V D_t^{(s,h,1)}\,\mathbf{h}_{t-1} + D_t^{(s,b,1)}\,\mathbf{b}$$

where the adaptation mechanism is defined by projecting the hidden state $\mathbf{z}_t$ of an adaptive model m:

$$D_t^{(k)} = U^k\,\mathbf{z}_t, \quad \mathbf{z}_t = m(\mathbf{x}_t, \mathbf{z}_{t-1}).$$

Table 1: Perplexities (lower is better) on Penn Treebank (left) and WikiText-2 (right).

Model	Valid	Test
LSTM [4]	82.2	78.4
AWD-LSTM [3]	60.0	57.3
TG-SC LSTM [2]	60.9	58.3
aLSTM	**57.6**	**55.3**

Model	Valid	Test
VD-LSTM [1]	91.5	87.7
AWD-LSTM [3]	68.6	65.8
TG-SC LSTM [2]	69.1	65.9
aLSTM	**67.5**	**64.5**

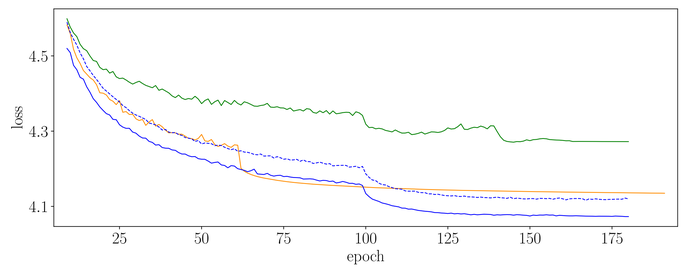

Figure 3: Val. curves on Penn Treebank. The aLSTM (*blue*) outperforms the state of the art AWD-LSTM (*orange*) in 144 epochs, compared to ~500. On Wikitext-2 (not shown), the aLSTM outperforms the AWD-LSTM in 160 epochs, which takes ~700 to converge. *Green:* LSTM; *blue, dashed* aLSTM with m as a feed-forward network.

References

[1] Hakan Inan, Khashayar Khosravi, and Richard Socher. Tying word vectors and word classifiers: A loss framework for language modeling. In *International Conference on Learning Representations*, 2017.

[2] Gábor Melis, Chris Dyer, and Phil Blunsom. On the State of the Art of Evaluation in Neural Language Models. In *International Conference on Learning Representations*, 2018.

[3] Stephen Merity, Nitish Shirish Keskar, and Richard Socher. Regularizing and optimizing LSTM language models. In *International Conference on Learning Representations*, 2018.

[4] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent Neural Network Regularization. In *International Conference on Learning Representations*, 2015.