Backprop Considered Harmful?

Hybrid-Evolution Strategies for Supervised Learning Training

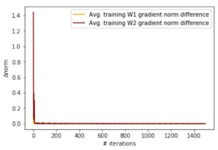
Andrew Bartolo | bartolo@stanford.edu

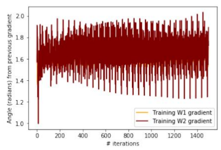
Summary

- Training supervised learning models is computationally intensive and **difficult to parallelize across multiple compute nodes**.
- In particular, batch gradient descent requires memoizing many gradients and potentially **broadcasting parameters** over a network.
- In this project, we assess the feasibility of Evolution Strategies for performing supervised learning training. Evolution Strategies are a stochastic optimization technique most commonly used in reinforcement learning.
- We found that even for simpler nets, effective Hybrid-ES requires extensive hyperparameter tuning, but its potential memory + data savings mean we should keep investigating it.

Investigating the Gradient

- We first characterized the behavior of the full gradient, as we want to mimic it stochastically.



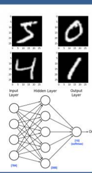


- Periodicity of the figure is (probably) due to cycling over minibatches.
- The norm of the gradients quickly converge, likely due to L2 regularization.

Baseline Network

Most of the algorithm design exploration was done using a multilayer perceptron (MLP) on the MNIST dataset. This allowed for relatively quick iteration and figuring out what worked/what didn't without having to train a huge net.

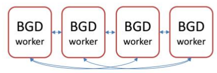
The MNIST MLP is has one input layer, one hidden layer (300 hidden units), and a 10-class softmax output layer. Learning rate was 10^{-4}. It was trained for 30 epochs, each over 50 minibatches of size 1000.



The Hybrid-Evolution Algorithm

Parallel BGD (N worker nodes)
Algorithm:

1. Split training set T into *N* subsets, T_n
2. For every iteration *i*, each worker node:
3. forward_prop(T_n, θ)
4. backward_prop(T_n, θ)
5. $\theta_n := \theta_n - \alpha d\theta_n$
6. **transmit(θ_n)**
7. **receive($\theta_{1\ldots n-1,\ n+1\ldots N}$)**
8. **θ := combine($\theta_{1\ldots N}$)**



Parallel Hybrid-ES (N worker nodes)
Algorithm:

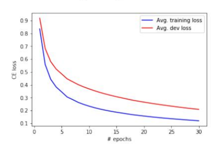
1. If iteration *i* % r == 0:
2. forward_prop(T, θ)
3. backward_prop(T, θ)
4. $\theta := \theta - \alpha d\theta$
5. Else:
6. For *K* attempts, each worker node:
7. $d\theta_{n,k} := d\theta + \boldsymbol{N(0, \sigma^2)}$
8. $\theta_{n,k} := \theta - \alpha\, d\theta_{n,k}$
9. θ_n = argmin forward_prop(T, $\theta_{n,k}$)
10. **transmit(<$rseed_n$, best $cost_n$>)**
11. **receive(<rseeds, best_costs>)**
12. **θ := combine(<rseeds, best_costs>)**

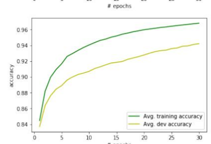
Results + Analysis

	Training Set Accuracy	Dev Set Accuracy
Reference	99.54%	96.57%
Assisted H-ES (r = 1)	99.85%	96.08%
H-ES, r = 2	96.80%	94.21%
H-ES, r = 3	95.34%	93.06%

Training Set	50,000 examples
Dev Set	10,000 examples

Losses and Accuracies, H-ES, r = 2, σ^2 = 1.25



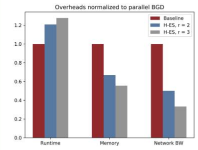


- Goal isn't to beat BGD at its own game, but to parallelize BGD in an approximate but much-lower-overhead way.
  Hybrid-ES can make forward progress without needing to compute the full gradient.
- May be better for driving training progress in later iterations (once gradient has stabilized).
- Optimal $\boldsymbol{\sigma^2}$: empirical gradient *component* var.
- Optimal r: with r too big, H-ES loses information from the full gradient and can't make progress.

Hyperparameters + Savings

Hyperparameters:

- K, the number of random perturbations each worker node makes (multiply by N)
- r, the interval for computing the full gradient (as opposed to a stochastic update)
- σ^2, the variance for the random shift matrices



	Runtime	Memory	Net BW
H-ES, r = 2	120.8%	66.7%	50.0%
H-ES, r = 3	127.7%	55.6%	33.3%

- Model uses the components defined in the algorithms section.
- Runtime doesn't take into account the cost of sending over network! (So this is a conservative estimate.)
- Network BW – not just data: delay, **energy**, etc.
- Backprop expected to be costlier, but wasn't (might be worse for larger nets). Memory + BW benefits increase with net size!

Future Work

- Try adaptively setting the σ^2 variance (shift scaling factor).
- Try stochastically adjusting different **components** of the gradient.
- Try learning some features of the gradient itself… ☺
- Try sampling random shifts from a non-normal distribution.
- End goal: *compress* the weights being sent over the network.
- Simulate across a real cluster, using heterogeneous (CPU, GPU, TPU) HW.

References

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient- based learning applied to document recognition." Proceedings of the IEEE, 86(11):2278-2324, November 1998. Dataset from http://yann.lecun.com/exdb/mnist/.

[2] Evolution Strategies as a Scalable Alternative to Reinforcement Learning. *OpenAI blog*. https://blog.openai.com/evolution-strategies/

[3] Evolution Strategies as a Scalable Alternative to Reinforcement Learning. (full paper) https://arxiv.org/abs/1703.03864

[4] Gradient-Free Optimization. *Stanford AA222*. http://adl.stanford.edu/aa222/lecture notes files/chapter6 gradfree.pdf.