Google Research

Meta-Learning Bidirectional Update Rules

Mark Sandler, Max Vladymyrov, Andrey Zhmoginov, Nolan Miller, Andrew Jackson, Tom Madams, Blaise Agüera y Arcas

ICML International Conference On Machine Learning

Goal

Meta-learn synapse update rules with very mild assumptions on the inner-loop (no loss functions, no gradients) that learns faster than traditional methods.

Motivation

SGD optimization via Backpropagation:

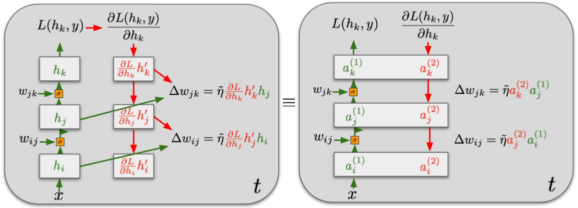
- Uses predefined loss function computed at every iteration.
- The loss is minimized via gradient descent (steepest direction of the current loss).
  - Optimization can use previous iterations (e.g. momentum), but (mostly) can't see forward.
- Optimization procedure is independent from the dataset.

Bidirectional Learning Update Rules (BLUR):

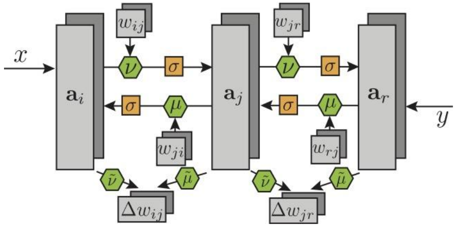
- Synapse updated rules are parametrized and meta-learned via a low-dimensional genome matrix.
- No predefined per-iteration loss function, no explicit gradients.
- Keep bidirectionality of the updates:
  - Input is passed at the forward pass,
  - Labels are passed at the backward pass.
- Metatrain to a given iteration (unroll).

SGD is a special case of two-state neurons

Backpropagation can be equivantly reformulated with generalized two-state neurons a_j^c, where j is a layer and $c \in \{0, 1\}$ is a state.



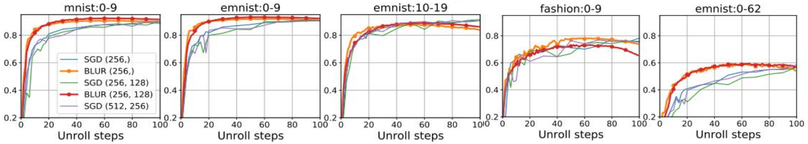
Bidirectional Learning Update Rules (BLUR)



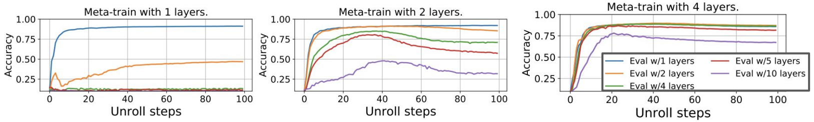
	Backpropagation/SGD	BLUR (Multi-state)
Forward	$a_j^c \leftarrow \phi^c\left(\sum_{i\in I(j),d} w_{ij} a_i^d \nu^{cd}\right)$	$a_j^c \leftarrow \sigma\left(f a_j^c + \eta \sum_{i,d} w_{ij}^c \nu^{cd} a_i^d\right)$
Backward	$a_i^{(2)} \leftarrow a_i^{(2)} \sum_{j\in J(i),d} w_{ij} a_j^d \mu^d$	$a_i^c \leftarrow \sigma\left(f a_i^c + \eta \sum_{j,d} w_{ji}^c \mu^{cd} a_j^d\right)$
Weight update	$w_{ij} \leftarrow w_{ij} - \tilde\eta \sum_{c,d} a_j^c \tilde\mu^c a_i^d \tilde\nu^d$	$w_{ij}^c \leftarrow \tilde f w_{ij}^c + \tilde\eta \sum_{e,d} a_i^e \tilde\nu^{ec} \cdot \tilde\mu^{cd} a_j^d$
States	- Two states neuron: $c, d \in \{1, 2\}$ - Single state synapse.	- k neuron states. - k synapse states (possibly asymmetric).
Feedback	- Derivative of the loss function.	- Passed directly to the final layer.
Forward pass	- Both updates computes from the first state. - Different activation functions for each state.	- All states are updated via transform matrix ν^{cd} - Same activation functions for each state. - Forget f and update η are learned parameters.
Backward pass	- Second state update only multiplicatively. - Linear activation.	- All states are updated via transform matrix μ^{cd} - Same activation for each state. - Forget f and update η are learned parameters.
Synapse update	- Second state of postsynaptic and first state of presynaptic. - Learning rate is a user parameter.	- All states from presynaptic and postsynaptic are mixed together via transform matrices $\tilde\nu^{cd}$ and $\tilde\mu^{cd}$. - Forget $\tilde f$ and update $\tilde\eta$ are learned parameters.

Generalization of a genome

- **Trained** on 10x10 MNIST using 2-layer 4-state architecture. **Validated** on 28x28 digits.



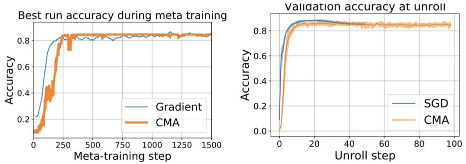
- Train networks with 1,2,4 layers to 10 untrolls and evaluated to 1,2,4,5,10 layers.



Meta-learning the genome

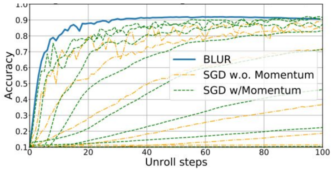
1. Start with a random genome
2. Repeat until meta-convergence:
   a. Apply forward/backward/synapse update for t unroll steps
   b. Measure the quality(*) of the learned synapses
   c. Meta-step: Update genome using ES or SGD

(*) quality can be any fitness functions, e.g. cross-entropy loss or validation accuracy.



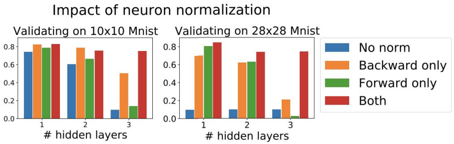
SGD w/ different parameters vs BLUR

Genome learns faster than SGD with any learning rate/momentum.



Role of normalization

Forward and backward (!!) activation normalization is important for good generalization.



Paper:

Code: