Evolutionary learning of recurrent neural networks

(unpublished experiments from 2008)

Tomas Mikolov, 2020

The idea

- stochastic gradient descent had many issues:
  - was believed to not work for deep nets, and also for recurrent nets
  - cannot optimize architecture of the network
  - requires differentiable cost function and strong supervision

- evolutionary optimization can somewhat avoid all these problems
  - very simple, but computationally much more expensive

Evolving recurrent networks

SGD:

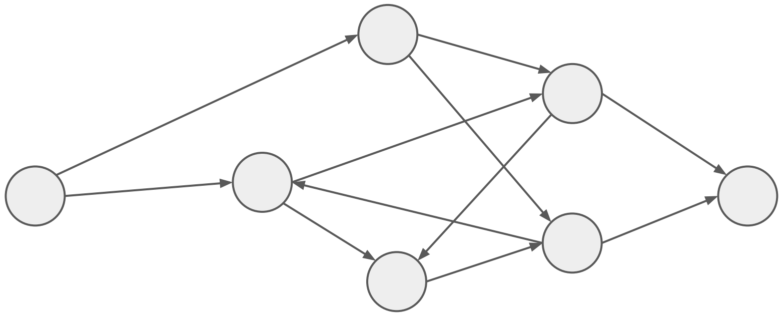
- compute **best** direction how to reduce error for given training example
- make a small step in this direction
- iterate for all training examples

Evolutionary learning:

- compute **random** change of the network weights and architecture
- check if the change is beneficial: if yes, keep it, otherwise reload the previous model

Tips & tricks

- start small
- accept only changes significantly better than threshold T
- reduce T with time



Can this actually work?

- for simple problems: yes! (stack-RNNs, problems like sequence memorization)
- for larger problems: very inefficient and slow

- the largest successful experiment in 2008: character-based language modeling using several tens of thousands of characters, after 2 days of training the same performance as 6-gram model

Why this does not scale?

- the changes are random:
    - less and less likely to be beneficial
    - can be seen as rule-based static learning rule: no adaptation or learning of the training algorithm happens during the evolution

- however, very easy to parallelize
- with some idea how to learn the model updates, there might be some hope
- also combination with SGD might help (-> neural architecture search)