The Forward-Forward Algorithm: Some Preliminary Investigations

https://www.cs.toronto.edu/~hinton/FFA13.pdf

작성자: 이미지처리팀-류채은

References:

- https://towardsdatascience.com/perturbation-theory-in-deep-neural-network-dnn-training-adb4c20cab1b#:~:text=Typically%2C perturbation theory is the,object interacting with the system.

original paper pdf:

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/1d0e146a-82f9-4bd1-872a-e6a308477607/Weight_perturbation__an_optimal_architecture_and_learning_technique_for_analog_VLSI_feedforward_and_recurrent_multilayer_networks.pdf

Related Works

Paper name: **Weight perturbation: An optimal architecture and learning technique for analog vlsi feedforward and recurrent multilayer networks.**

link: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=105429

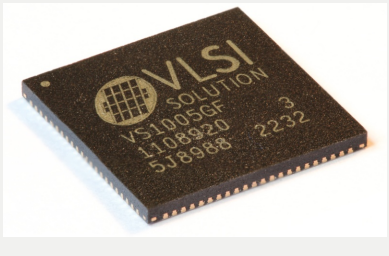
TL;DR

- **Using gradient descent with the direct approximation of the gradient instead of back-propagation (=weight perturbation) is more economical for parallel analog implementations.**

Prerequisites

What is **VLSI?**

VLSI (Very Large Scale Integration): 하나의 칩에 들어가 있는 소자의 수가 10만개 ~ 100만개인 경우.



What is **perturbation in DL or ML**?

A small perturbation or nudge in various parameters/components associated with training such as gradients, weights, inputs etc. can affect DNN training in overcoming some of the issues one might bump into, for example, *vanishing gradient* problem, *saddle point* trap, or creating a robust model to avoid malicious attacks through *adversarial training* etc.

ex)

$$\nabla w_t J \leftarrow \nabla w_t J + \mathcal{N}(0, \sigma_t^2),$$

ref: https://towardsdatascience.com/perturbation-theory-in-deep-neural-network-dnn-training-adb4c20cab1b#:~:text=Typically%2C perturbation theory is the,object interacting with the system.

Main Concept

Weight perturbation, where the gradient is approximated to a finite difference, this paper shows that gradient evaluation using weight perturbation is a cheaper solution with respect to hardware and can be used equally well to train recurrent networks.

Algorithm

```
for each pattern p {
      E = ForwardPass()
      ClearDeltaWeights()
      for each weight w_ij do {
              Epert = ApplyPerturbate(w_ij)
              DeltaError = Epert - E
              DeltaW[i][j] = - η * DeltaError/Perturbation
              RemovePerturbation(w_ij)
      }
}
```

Weight perturbation algorithm in its simplest form.

Gradient Evaluation using Weight Perturbation:

$$\frac{\delta E}{\delta w_{ij}} \approx \frac{(E_{pert} - E)}{\Delta_{pert} w_{ij}}$$

- E: total mean-square-error produced at the output of the network for a given pair of input and training patterns and a given value of the weights.
- $\Delta_{pert} w_{ij}$: perturbation applied at weight w_{ij}

→ **As the gradient is approximated, no backward propagation pass is needed and only forward path is required.**