Model Compression

Ivan Skorokhodov

June 19, 2020

Overview

- Accelerating inference or training
- Reducing memory footprint
- Theoretical curiosity

Pruning

- *Pruning* is removing weights/neurons in a model while preserving the accuracy
- It can be done at different stages:
  - before training
  - during training
  - after training
  - iteratively train/prune several times
- Pruning neurons speeds up a model, but:
  - [2] argues that training the pruned model from scratch would give the same performance
  - So the main value is in optimizing the architecture
- Pruning weights (theoretically) reduces the number of FLOPs, but:
  - Resulted sparse matrices are not "sparse enough" to provide practical benefits (sparse matrix-vector multiplications are usually based on non-parallel computations)
  - [1, 2] claim that modern SotA weight-pruning algorithms do not generalize on large datasets

Pruning weights

Simple strategies:

- Apply L_1-regularization during training
- Iterative Magnitude Pruning (IMP): prune weights

Lottery Ticket Hypothesis (LTH)

-

Synaptic Flow [3]

- Pruning algorithms remove weights based on score values associated with each weight
- These scores can usually be represented as

$$\mathcal{S}(\theta) = \frac{\partial \mathcal{R}}{\partial \theta} \odot \theta \tag{1}$$

  for some function $R(\theta)$
- Authors call this function $S(\theta)$ a *synaptic saliency* (importance) function
- Many pruning algorithms have a form similar to (1):
  - Magnitude Pruning:

$$\mathcal{R}(\theta) = \frac{1}{2}\|\theta\|_2^2 \Longrightarrow S(\theta_i) = \theta_i^2 \tag{2}$$

  -

Pruning neurons (*structured pruning*)

-
-

pass

pass

pass

pass

pass

Knowledge distillation

pass

Conditional computation

pass

Architectural tricks

pass

pass