Logarithmic Pruning is All You Need

Laurent Orseau
Marcus Hutter
Omar Rivasplata

DeepMind

NEURAL INFORMATION PROCESSING SYSTEMS

OVERVIEW

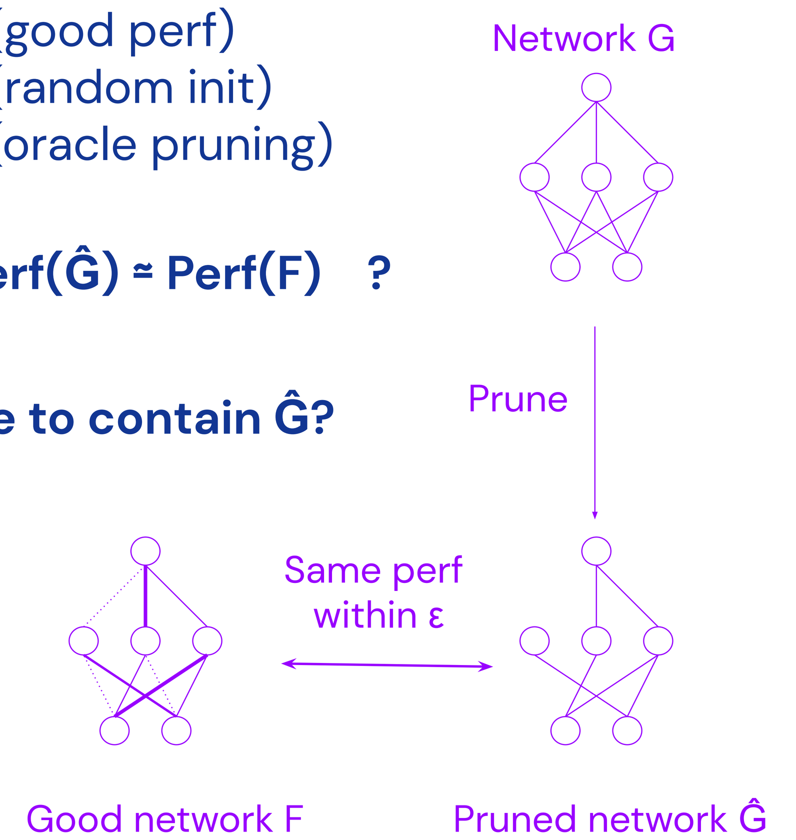
Problem description

- Conjecture by Ramanujan et al. 2019

- **F**: Small **ReLU** NN (good perf)
- **G**: **Big** ReLU NN (random init)
- **Ĝ** = **Prune**(G) (oracle pruning)

∃G, ∀F, ∃Ĝ s.t. **Perf(Ĝ) ≈ Perf(F)** ?

- **How big must G be to contain Ĝ?**



Previous result (Malach et al., ICML 2020)

- **Idea: Add an intermediate ReLU layer**
  #layers(G) = 2ℓ
  #neurons in G per target weight of F:

$$O\left(\frac{n^3\ell^2}{\varepsilon^2}\log\frac{n\ell}{\delta}\right)$$

- **Strong assumptions**
  - $\|W_i\|_2 \le 1$ (weights at layer i)
  - $\|W_i\|_\infty \le 1/\sqrt{n}$
  - $\|W_i\|_0 \le n$
  - $\|x\|_2 \le 1$ (inputs)

Our result

- Assumptions
  - Same as Malach et al.
  - **Hyperbolic** distribution of the initial weights

#neurons in G per target weight of F:

$$\boxed{\tilde{O}\left(\log\frac{n\ell}{\varepsilon}\right)}$$

n: #neurons(F) per layer
ℓ: #layers(F)
ε: approximation error

Our result (more general)

- Assumptions
  - **Hyperbolic** distribution of the initial weights

#neurons in G per target weight of F:

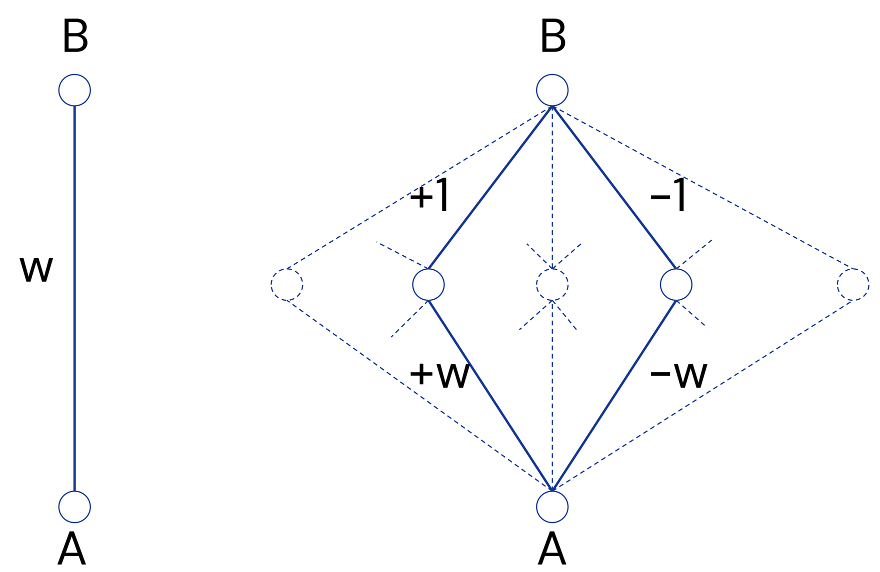
$$\tilde{O}\left(\log\left(\frac{n\ell}{\varepsilon}w_{\max}F_{\max}\right)+\sum_{i=1}^{\ell}\log\max\{1,\|W_i\|_2\}\right)$$

F_{max}: max activation of any neuron
w_{max}: max weight
W_i: matrix weight at layer i

TECHNICAL IDEAS

Weight Decomposition (Malach et al. 2020)

- Simulate one weight with 2 ReLU neurons



Target weight in F

Decomposition in G

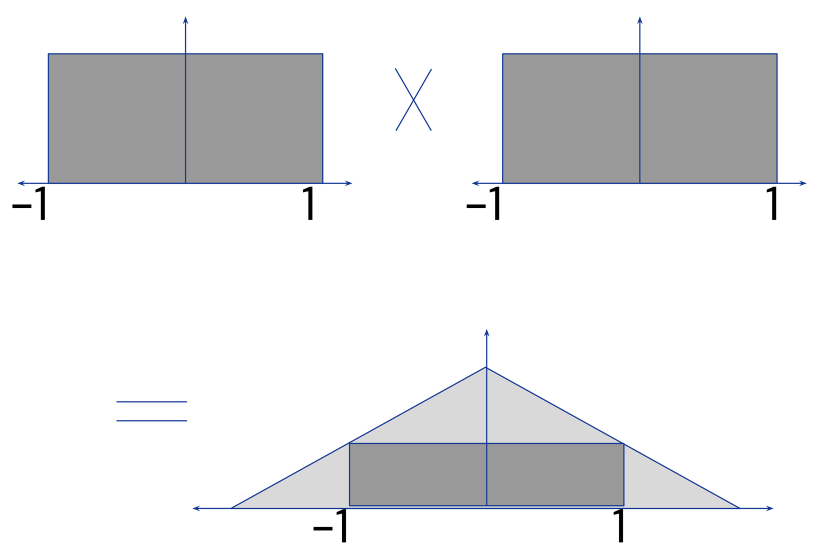
w = 1 × σ(w) + (−1) × σ(−w)

σ(x) = max{0, x}

Sample intermediate neurons until
Input and outputs are all within ε

Takes O(1/ε² log 1/δ) samples

Product weights



w ≈ wa σ(wb) + wc × σ(wd)

Sample intermediate neurons until

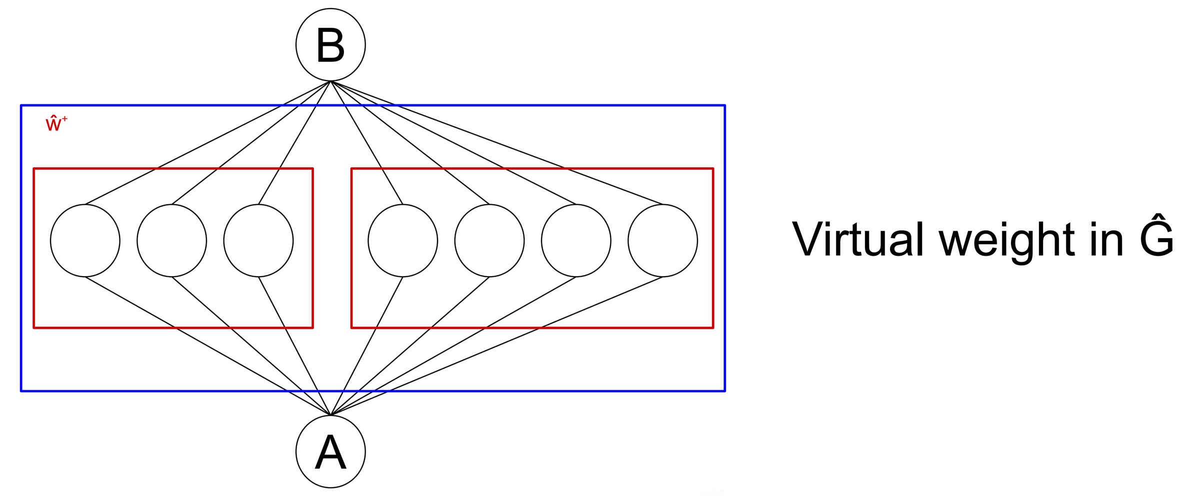
wa × wb ≈ +w & sgn(wa) = +1
wc × wd ≈ −w & sgn(wc) = −1

Requires O(1/ε log 1/δ) samples

(similar for hyperbolic distribution
by a change of variable argument)

Golden Ratio Decomposition

- Take advantage of the sum in the neuron function
- Binary decomposition: requires log 1/ε intermediate neurons
- Weight are sampled from hyperbolic P(w) ≈ 1/w
- Base 2 not possible, use base 3/2 instead (or φ=(1+√5)/2)



Virtual weight in Ĝ

Binary decomposition: $\sum_{i=1}^{k} b_i\, 2^{-i}$

Golden-ration decomposition: $\sum_{i=1}^{k} b_i\, x_i^{-i}\,,\quad x_i\in[\varphi^{-i-1},\varphi^{-i}]$

$\varphi=\frac{1+\sqrt{5}}{2}$ or $\varphi=\frac{3}{2}$

→ Hyperbolic sampling: $P([\varphi^{-i-1},\varphi^{-i}]) \ge c \quad \forall i\,;\quad P_w(w) \propto 1/w$

Need only Õ(log 1/ε log 1/δ) samples

Batch sampling

- Don't throw away samples that can be reused elsewhere
- Fill k **disjoint** categories each with n samples w.p. 1−δ
  - P(any cat.) ≥ c
- Needs #samples M: (Chernoff-Hoeffding)

$$M=\left\lceil\frac{2}{c}\left(m+\ln\frac{k}{\delta}\right)\right\rceil$$

m: #weights(F) per layer
k: #neurons to decompose a weight = O(log 1/ε)
c: probability of one of the k segments

CONCLUSION

- Is hyperbolic sampling worth trying in practice?

- What about uniform sampling? A lower bound?
  - *Pensia et al., Neurips 2020*
    "Optimal lottery tickets via subset-sum: Logarithmic over-parameterization is sufficient."