Principled Weight Initialization for Hypernetworks [1]

April 16, 2020

[1] "Principled Weight Initialization for Hypernetworks" by Chang, Flokas, and Lipson

Overview

- *Hypernetwork* is a model that produces weights for another model (*target model*)
- Hypernetworks are used in a lot of applications: mutli-task learning, continual learning, weights compression, etc.
- People use traditional initialization strategies (Xavier, He, etc) for hypernetworks
- But these traditional initializations of a hypernetwork lead to a bad initialization of a target model
- Authors fixed initialization for a hypernetwork to make the training more stable

What is a hypernetwork?

- Imagine we have a target model $f_\theta(x)$ which solves the task of interest
- A hypernetwork H_ϕ produces the parameters for each layer l of f_θ given an embedding e_l: $\theta_l = H_\phi(e_l)$.
- During training we optimze parameters ϕ instead of θ by backpropagating $\Delta\theta$ further to $\Delta\phi$.
- People usually use very small hypernetworks (1 or 2 hidden layers) since our output is very large
- For example, for a dense layer of input-output sizes 500×500, our output layer for H_ϕ is $250,000$ neurons.

Notation

We will use the following notation:

- Einstein summation:
$$\alpha_i \beta^i = \sum_{i=1}^{n} \alpha_i \beta^i$$
- Denote by $W[t]$ a weight matrix in t-th layer for the main model
- Then transformation $y^i = W[t]^i_j x^j + b^i$ means $y^i = W^{(t)}_i x + b^i$, i.e. t_i is the i-th neuron and $W^{(t)}_i$ is the i-th row in the t-th layer's matrix $W^{(t)}$.
- $\partial_x f = \frac{\partial f}{\partial x}$
- Hypernetwork H_ϕ produces W^i_j by:
$$W^i_j = H^i_{jk} h(e)^k + \beta^i_j,$$
where e is a layer's embedding, h is the hypernetwork's body, H^i_j, β^i_j are output weights and biases (H is a 3-dimensional tensor).

Xavier fan-in init

Suppose we have a transformation $y^i = W^i_j x^j + b^i$. Xavier init is based on the following assumptions:

1. $\forall i, j : W^i_j, x^j, b^i$ are independent from each other.
2. $\forall i, j : \mathbb{E}\left[W^i_j\right] = 0$
3. $\forall j : \mathbb{E}\left[x^j\right] = 0$
4. $\forall i : \mathbb{E}\left[b^i\right] = 0$.

If these assumptions hold, then $\mathbb{E}\left[y^i\right] = 0$ and:

$$\mathrm{Var}\left[y^i\right] = d_j \mathrm{Var}\left[W^i_j\right] \mathrm{Var}\left[x^j\right]$$

- The goal of a good initialization is to make $\mathrm{Var}\left[y^i\right] = \mathrm{Var}\left[x^j\right]$.
- This gives us $\mathrm{Var}\left[W^i_j\right] = \frac{1}{d_j}$

Fan-out init and Kaiming inits

- On the previous slide, we tried to have $\mathrm{Var}\left[y^i\right] = \mathrm{Var}\left[x^j\right]$ during a forward pass.
- But what if we want a similar property for a backward pass?
- Since backprop for a linear layer gives $\partial_x \mathcal{L} = \partial_y \mathcal{L} W$ then the similar reasoning leads to $\mathrm{Var}\left[W^i_j\right] = \frac{1}{d_i}$. This is a *fan-out* init.
- Since we want both $\mathrm{Var}\left[W^i_j\right] = \frac{1}{d_j}$ and $\mathrm{Var}\left[W^i_j\right] = \frac{1}{d_i}$, which is impossible, let's just take their harmonic mean and set $\mathrm{Var}\left[W^i_j\right] = \frac{2}{d_i + d_j}$
- Kaiming inits are very similar, but we consider transformations like $y^i = W^i_j \sigma(x^j) + b^i$ for different non-linearities σ (ReLU, LeakyReLU, tanh, etc), obtaining different initialization schemes.

Modern (ad-hoc) techniques of initializing hypernetworks

Currently, people use the following schemes to initialize their hypernetworks:

- M1: Xavier or Kaiming inits
- M2: Small random values
- M3: Kaiming init, but with scaling output layer by 0.1
- M4: Kaiming init, but with scaling hypernetwork embeddings.

A problem: using these initialization schemes in a hypernetwork produces such weights $W[t]$ in a target network that have bad variances $\mathrm{Var}\left[W_j^i\right]$, making training difficult and unstable.

Hyperfan-in assumptions

First, authors make the following assumptions:

- (1-4) Xavier assumptions for all the layers in the hypernet $h(e)$
  - This would give us $\mathrm{Var}\left(h(e)^k\right) = \mathrm{Var}\left(e^l\right)$
- (5) ...everything is independent of each other in the output layer
- (6-8) ...everything has zero mean in the output layer

Hyperfan-in init

Now, we want a transformation $y^i = W^i_j x^j + b^i$ not to change variance:

- i.e, we want $\text{Var}\left[y^i\right] = \text{Var}\left[x^j\right]$
- Using derivations similar to Xavier's ones, we can obtain:

$$\text{Var}\left[H^i_{jk}\right] = \frac{1}{2\mathrm{d}_j \mathrm{d}_k \,\text{Var}\left[e^m\right]} \tag{1}$$

- This is a hyperfan-in init for hypernetworks.
- If we want to generate biases for a target model as well, we would need to derive init for the corresponding component as well (you check the paper)
- If we want hyperfan-out, this can be done in a similar manner

HyperKaiming inits

- On the previous slide, we had Xavier inits. But what if we care about activation functions?
- Authors give an derivation for ReLU:

Initialization	Variance Formula	Initialization	Variance Formula
Hyperfan-in	$\mathrm{Var}(H^i_{jk}) = \frac{2^{\mathbb{1}_{\mathrm{ReLU}}}}{2^{\mathbb{1}_{\mathrm{HBias}}} \mathrm{d}_j \mathrm{d}_k \mathrm{Var}(e[1]^m)}$	Hyperfan-out	$\mathrm{Var}(H^i_{jk}) = \frac{2^{\mathbb{1}_{\mathrm{ReLU}}}}{\mathrm{d}_i \mathrm{d}_k \mathrm{Var}(e[1]^m)}$
Hyperfan-in	$\mathrm{Var}(G^i_l) = \frac{2^{\mathbb{1}_{\mathrm{ReLU}}}}{2\mathrm{d}_l \mathrm{Var}(e[2]^n)}$	Hyperfan-out	$\mathrm{Var}(G^i_l) = \max(\frac{2^{\mathbb{1}_{\mathrm{ReLU}}}(1-\frac{\mathrm{d}_j}{\mathrm{d}_i})}{\mathrm{d}_l \mathrm{Var}(e[2]^n)}, 0)$

- For other activations, it should be derivable in a similar manner

Experiments: FeedForward Network on MNIST (1/2)

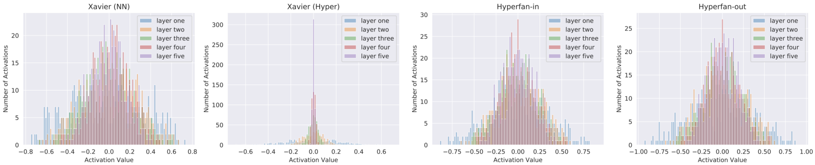

Figure: MNIST mean activations before training

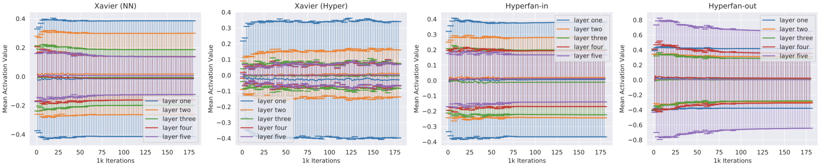

Figure: MNIST activations evolution

Experiments: FeedForward Network on MNIST (2/2)

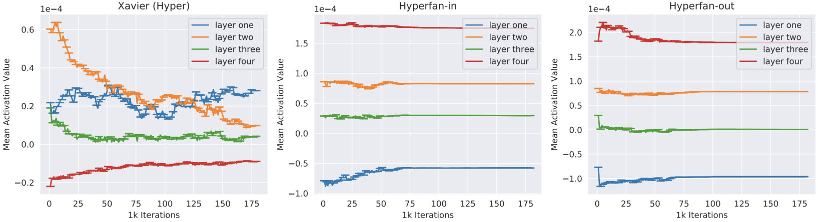

Figure: MNIST mean output value of $H_\phi(e)$

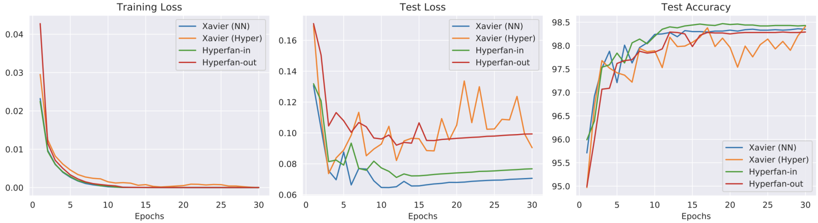

Figure: MNIST train/test scores

A bunch of final thoughts

- Proper initialization is very important.
- The paper rises an important question of initialization for new architectures
- It is possible to derive Hyperkaiming inits for other non-linearities
- Authors' results on ImageNet for a bayesian NN looks suspicious ($\sim$17% Top-5 accuracy for MobileNet after 25 epochs)