GradInit: Learning to Initialize Neural Networks for Stable and Efficient Training


КАКОЙ ГРАД?
risovach.ru

Какой может быть начальная инициализация весов

насемплировать веса из какого-нибудь распределения

взять предобученные веса

обучить начальную инициализацию

Какие-нибудь распределения

Распределение с нулевым средним и ограниченной дисперсией

$$w \sim \mathcal{N}(0, 1) \qquad w \sim \mathcal{U}[-1, 1]$$

Инициализация Ксавьера

$$w \sim \mathcal{U}\left[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}\right] \qquad w \sim \mathcal{U}\left[-\frac{\sqrt{6}}{\sqrt{n+m}}, \frac{\sqrt{6}}{\sqrt{n+m}}\right]$$

Инициализация Хе (Кеминга)

$$w \sim \mathcal{N}\left(0, \sqrt{\frac{2}{n}}\right)$$

Обучить инициализацию

Алгоритм MetaInit

Гипотеза: градиентный спуск работает лучше если мы берем начальную точку из области пространства, которая имеет небольшую локальную кривизну

Идея: нормировать веса, чтобы попасть в такую гладкую Область

Плюсы: работает
Минусы: очень долгий

Хотим: быстрый

Обучить инициализацию

Алгоритм GradInit

идея:
нормируем веса в каждом из слоев модели

$$W_1, .., W_M,\ w_{ij} - \text{матрицы весов}$$

$$w_{ij} \sim \mathcal{N}(0, 1)$$

$$\{\alpha_1, .. , \alpha_M\},\ \alpha_i = 1$$

$$\{\alpha_1 W_1, .. , \alpha_M W_M\}, - \text{инициализация}$$

алгоритм:
градиентным спуском найдем такие альфы, чтобы первые шаги градиентного спуска уменьшали функцию потерь настолько, насколько это возможно

в процессе градиентного спуска мы не изменяем матрицы весов

Алгоритм GradInit

Более формально:

$$\begin{cases} L(\tilde{S}, \Theta_m - \eta \cdot \mathcal{A}[g_{s,\theta_m}]) \to \min\limits_{\alpha} \\ \|g_{s,\theta_m}\|_{p_{\mathcal{A}}} \leqslant \gamma \end{cases}$$

$\Theta_m = \{\alpha_1 W_1, .. , \alpha_M W_M\}$

L – функция потерь
$\mathcal{A}$ – метод спуска: SGD или Adam
η – learning rate для метода спуска для исходной модели
g_{s,θ_m} – градиент функции потерь
γ – нормировочная константа для нормы градиента
$p_{\mathcal{A}}$ – норма, соотв. методу спуска в исходной задаче

Алгоритм GradInit

$\boldsymbol{m}_1 \leftarrow \mathbf{1}$
for $t = 1$ **to** T **do**
  Sample S_t from training set.
  $L_t \leftarrow \frac{1}{|S_t|} \sum_{x_k \in S_t} \ell(x_k; \boldsymbol{\theta}_{\boldsymbol{m}_t}),\ \boldsymbol{g}_t \leftarrow \nabla_{\boldsymbol{\theta}} L_t$
  if $\|\boldsymbol{g}_t\|_{p_{\mathcal{A}}} > \gamma$ **then**
    $\boldsymbol{m}_{t+1} \leftarrow \boldsymbol{m}_t - \tau \nabla_{\boldsymbol{m}_t} \|\boldsymbol{g}_t\|_{p_{\mathcal{A}}}$
  else
    Sample $\tilde{S}_t$ from training set.
    $\tilde{L}_{t+1} \leftarrow \frac{1}{|\tilde{S}_t|} \sum_{x_k \in \tilde{S}_t} \ell(x_k; \boldsymbol{\theta}_{\boldsymbol{m}_t} - \eta \mathcal{A}[\boldsymbol{g}_t])$
    $\boldsymbol{m}_{t+1} \leftarrow \boldsymbol{m}_t - \tau \nabla_{\boldsymbol{m}_t} \tilde{L}_{t+1}$
  Clamp $\boldsymbol{m}_{t+1}$ using $\underline{\alpha}$

Алгоритм GradInit

Подробнее про некоторые параметры

- **батчи**
- **шаг алгоритма**
- **нижняя граница для m**
- **шаг спуска модели и гамма**

Зачем нужно ограничение?

Почему ограничение а не штраф?

$$L(\tilde{S}, \Theta_M - \eta \cdot \mathcal{A}[g_{s,\theta_m}]) + \lambda \|g_{s,\theta_m}\|_{p_{\mathcal{A}}} \to \min_m$$

$\boldsymbol{m}_1 \leftarrow \mathbf{1}$
for $t = 1$ **to** T **do**
  Sample S_t from training set.
  $L_t \leftarrow \frac{1}{|S_t|} \sum_{x_k \in S_t} \ell(x_k; \boldsymbol{\theta}_{\boldsymbol{m}_t}),\ \boldsymbol{g}_t \leftarrow \nabla_{\boldsymbol{\theta}} L_t$
  if $\|\boldsymbol{g}_t\|_{p_{\mathcal{A}}} > \gamma$ **then**
    $\boldsymbol{m}_{t+1} \leftarrow \boldsymbol{m}_t - \tau \nabla_{\boldsymbol{m}_t} \|\boldsymbol{g}_t\|_{p_{\mathcal{A}}}$
  else
    Sample $\tilde{S}_t$ from training set.
    $\tilde{L}_{t+1} \leftarrow \frac{1}{|\tilde{S}_t|} \sum_{x_k \in \tilde{S}_t} \ell(x_k; \boldsymbol{\theta}_{\boldsymbol{m}_t} - \eta \mathcal{A}[\boldsymbol{g}_t])$
    $\boldsymbol{m}_{t+1} \leftarrow \boldsymbol{m}_t - \tau \nabla_{\boldsymbol{m}_t} \tilde{L}_{t+1}$
  Clamp $\boldsymbol{m}_{t+1}$ using $\underline{\alpha}$

Model	$\frac{\lvert\tilde{S}\cap S\rvert}{\lvert S\rvert}$	Acc_1	Acc_{best}
VGG-19	0	21.9 ± 4.4	94.5 ± 0.1
w/o BN	0.5	$\mathbf{29.3 \pm 0.6}$	$\mathbf{94.7 \pm 0.02}$
(20.03 M)	1	28.7 ± 1.0	94.5 ± 0.1

Эксперименты

Table 3: First epoch (Acc_1) and best test accuracy over all epochs (Acc_{best}) for models on CIFAR-10. We report the mean and standard error of the test accuracies in 4 experiments with different random seeds. Best results in each group are in bold.

Model (# Params)		VGG-19 w/o BN (20.03M)	VGG-19 w/ BN (20.04M)	ResNet-110 w/o BN (1.72M)	ResNet-110 w/ BN (1.73M)	ResNet-1202 w/ BN (19.42M)
Kaiming	Acc_1	29.1 ± 1.5	12.6 ± 0.6	16.1 ± 2.1	23.2 ± 0.9	12.9 ± 2.8
	Acc_{best}	94.5 ± 0.1	94.4 ± 0.1	94.2 ± 0.1	95.0 ± 0.2	94.4 ± 0.6
+1 epoch (Const. LR)	Acc_1	37.2 ± 1.1	19.6 ± 4.0	21.0 ± 3.8	32.5 ± 3.8	12.6 ± 2.8
	Acc_{best}	94.4 ± 0.1	94.5 ± 0.1	93.9 ± 0.4	94.7 ± 0.3	94.0 ± 0.4
+1 epoch (Warmup)	Acc_1	37.4 ± 1.2	53.5 ± 2.9	19.8 ± 0.5	48.7 ± 1.1	28.1 ± 1.3
	Acc_{best}	94.4 ± 0.1	94.7 ± 0.1	94.1 ± 0.1	95.1 ± 0.1	95.4 ± 0.2
MetaInit	Acc_1	30.5 ± 0.9	35.1 ± 0.6	14.6 ± 2.2	29.0 ± 1.5	11.7 ± 1.6
	Acc_{best}	94.6 ± 0.1	94.6 ± 0.1	94.2 ± 0.1	94.8 ± 0.1	95.0 ± 0.5
GradInit	Acc_1	29.3 ± 0.6	47.8 ± 1.8	36.2 ± 0.8	38.2 ± 0.9	29.0 ± 1.1
	Acc_{best}	$\mathbf{94.7} \pm 0.1$	$\mathbf{95.1} \pm 0.1$	$\mathbf{94.6} \pm 0.1$	$\mathbf{95.4} \pm 0.1$	$\mathbf{96.2} \pm 0.1$

Эксперименты

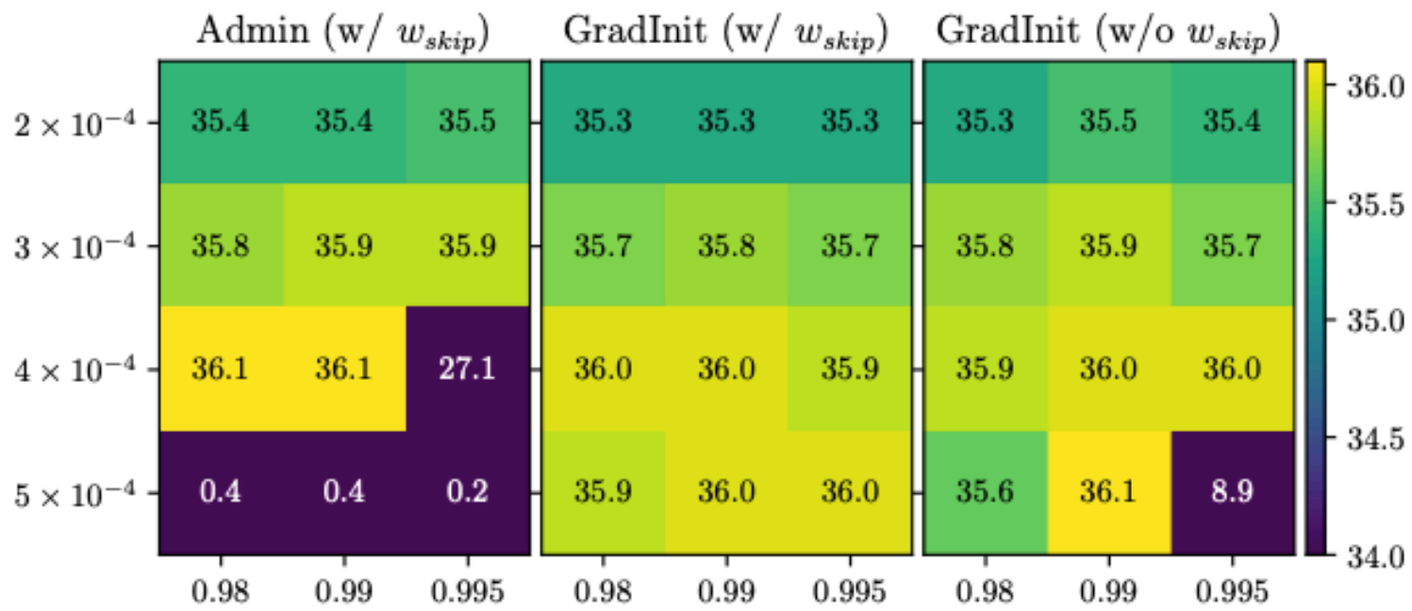

Figure 3: BLEU scores for the Post-LN Transformer without learning rate warmup using Adam on IWSLT-14 DE-EN under different learning rates $\eta_{\max}$ (y axis) and β_2 (x axis). Each result is averaged over 4 experiments.

спасибо за внимание