TTIC 31230, Fundamentals of Deep Learning

David McAllester, Autumn 2023

Stochastic Gradient Descent (SGD)

Temperature, Batch Size, Momentum, and Adam

The Classical Convergence Theorem

$$\Phi \texttt{ -= } \eta_t \nabla_\Phi \mathcal{L}(\Phi, x_t, y_t)$$

For "sufficiently smooth" non-negative loss with

$$\eta_t \geq 0 \qquad \lim_{t \to \infty} \eta_t = 0 \qquad \sum_t \eta_t = \infty \qquad \sum_t \eta_t^2 < \infty$$

we have that the training loss $E_{(x,y)\sim\text{Train}} \mathcal{L}(\Phi, x, y)$ converges to a limit and any limit point of the sequence Φ_t is a stationary point in the sense that $\nabla_\Phi E_{(x,y)\sim\text{Train}} \mathcal{L}(\Phi, x, t) = 0$.

Rigor Police: One can construct cases where Φ diverges to infinity, converges to a saddle point, or even converges to a limit cycle.

Physicist's Proof of the Convergence Theorem

Since $\lim_{t \to \infty} \eta_t = 0$ we will eventually get to arbitrarilly small learning rates.

For sufficiently small learning rates any meaningful update of the parameters will be based on an arbitrarily large sample of gradients at essentially the same parameter value.

An arbitrarily large sample will become arbitrarily accurate as an estimate of the full gradient.

But since $\sum_t \eta_t = \infty$, no matter how small the learning rate gets, we still can make arbitrarily large motions in parameter space.

For a rigorous proof see Neuro-Dynamic Programming, Bertsekas and Tsitsiklis, 1996.

Vanilla SGD

$$\hat{g} = E_{(x,y)\sim\text{Batch}} \nabla_{\Phi} \mathcal{L}(\Phi, x, y)$$

$$\Phi_{t+1} = \Phi_t - \eta_t \hat{g}$$

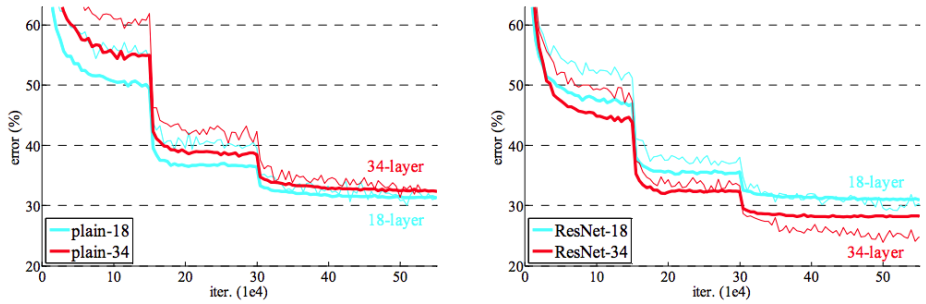
η_t is the learning rate for time t.

The Learning Rate Schedule

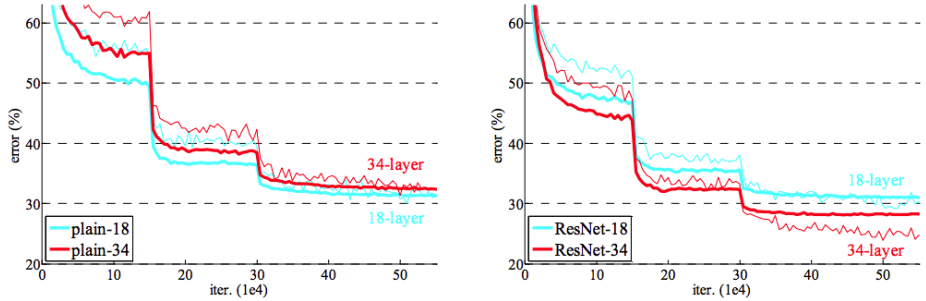
At the beginning a large learning rate reduces the loss faster. Near the end a small learning rate reaches a lower loss.

How the learning rate changes over time is called **the learning rate schedule**.

In the early days of deep learning the learning rate was reduced by a factor of 2 in steps (as above).



Temperature



As the learning rate is decreased (here by factors of 2) the asymptotic loss decreases.

We will say that two setting of hyper-parameters (for example η and the batch size B) run at the same "temperature" if they result in the same asymptotic loss value.

Two setting of hyper-parameters (such as the learning rate η and batch size B) will behave similarly if they have the same **temperature schedule** (as opposed to learning rate schedule).

Physical Temperature

Physical temperature is a relationship between the energy of a system state and its probability.

$$P(x) = \frac{1}{Z}\, e^{\frac{-E(x)}{kT}} \qquad Z = \sum_x e^{\frac{-E(x)}{kT}}$$

This is called the Gibbs or Boltzman distribution.

$E(x)$ is the energy of physical microstate state x.

k is Boltzman's constant.

Z is called the partition function.

We will eventually show that under certain (somewhat special) conditions the asymptotic probability distribution over model parameters is a Gibbs distribution where the energy is the loss.

Temperature

The Gibbs distribution is typically written as

$$P(x) = \frac{1}{Z}\, e^{-\beta E(x)}$$

$\beta = \frac{1}{kT}$ is the (inverse) temperature parameter.

"Hot" is when β is small and "cold" is when β is large (confusing).

Batch Size and Temperature

PyTorch vanilla SGD uses the following update which defines a meaning of the learning rate η.

$$\Phi_{t+1} \texttt{ -= } \eta \hat{g}_t$$

$$\hat{g}_t = \frac{1}{B} \sum_b \hat{g}_{t,b}$$

Here $\hat{g}_t$ is the average gradient over the batch.

For PyTorch vanilla SGD both the batch size and the learning rate influence temperature (not good).

Making Temperature Independent of Batch Size

For batch size 1 with learning rate η_0 we have

$$
\begin{aligned}
\Phi_{t+1} &= \Phi_t - \eta_0 \, \nabla_\Phi \mathcal{L}(t, \Phi_t) \\
\Phi_{t+B} &= \Phi_t - \sum_{b=0}^{B-1} \eta_0 \, \nabla_\Phi \mathcal{L}(t+b, \Phi_{t+b-1}) \\
&\approx \Phi_t - \eta_0 \sum_b \nabla_\Phi \mathcal{L}(t+b, \Phi_t) \\
&= \Phi_t - B\eta_0 \, \hat{g}_t
\end{aligned}
$$

For batch updates $\Phi_{t+1} = \Phi_t - B\eta_0 \, \hat{g}_t$ the temperature is essentially determined by η_0 independent of B.

Making Temperature Independent of B

In 2017 it was discovered that setting $\eta = B\eta_0$ allows very large (highly parallel) batches.

Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour, Goyal et al., 2017.

Momentum

Momentum is another hyper-parameter in Deep Learning.

Momentum uses a moving average of the gradient.

Consider a sequence x_1, x_2, x_3, ...

For $t \geq N$, the **rolling average** of the N most recent values is

$$\overline{x}_t = \frac{1}{N} \sum_{k=0}^{N-1} x_{t-k}$$

This is awkward to compute and is influenced by both the value added and the value removed from the average.

Exponential Moving Average (EMA)

Consider a sequence x_1, x_2, x_3, ...

The corresponding **exponential moving average (EMA)** is

$$\tilde{x}_0 = 0$$

$$\tilde{x}_t = \left(1 - \frac{1}{N}\right) \tilde{x}_{t-1} + \left(\frac{1}{N}\right) x_t$$

This behaves like the rolling average of the last N values but is easy to compute and does not involve removing any particular old value.

Exponential Moving Average (EMA)

$$\tilde{x}_t = \left(1 - \frac{1}{N}\right)\tilde{x}_{t-1} + \left(\frac{1}{N}\right)x_t$$

$$= \frac{1}{N}\sum_{i=0}^{t-1}\left[\left(1 - \frac{1}{N}\right)^i x_{t-i}\right]$$

$$\sum_{i=0}^{\infty}\left(1 - \frac{1}{N}\right)^i = N$$

The Conventional Formulation of EMAs

$$\tilde{x}_t = \left(1 - \frac{1}{N}\right) \tilde{x}_{t-1} + \left(\frac{1}{N}\right) x_t$$

is written as

$$\tilde{x}_t = \beta \tilde{x}_{t-1} + (1 - \beta) x_t$$

where

$$\beta = 1 - 1/N$$

But we can use any $\beta \in [0, 1)$.

In deep learning we typically take β to be .9, .99 or .999 corresponding to N being 10, 100 or 1000.

EMA Momentum

$$\tilde{g}_0 = 0$$

$$\tilde{g}_t = \left(1 - \frac{1}{N}\right) \tilde{g}_{t-1} + \frac{1}{N}\hat{g}_t, \quad N \geq 1$$

$$= \beta \tilde{g}_{t-1} + (1 - \beta)\hat{g}_t, \quad \beta \in [0, 1)$$

$$\Phi_{t+1} = \Phi_t - \eta \tilde{g}_t$$

EMA Momentum: Temperature is Independent of N

Unless the batch size is extremely large (larger than possible for academics), the temperature is determined by the total effect of a single training gradient $\hat{g}$.

With EMA momentum the total contribution of a single $\hat{g}_t$ is

$$\begin{aligned} & \frac{1}{N} \sum_{i=0}^{\infty} \left(1 - \frac{1}{N}\right)^i \\ = \; & (1-\beta) \sum_{i=0}^{\infty} \beta^i \\ = \; & 1 \end{aligned}$$

Hence the temperature is determined by η independent of N.

Heavy Ball Momentum

The PyTorch implementation of vanilla SGD uses "heavy ball momentum".

$$
\begin{aligned}
v_t &= \mu v_{t-1} + \eta * \hat{g}_t \\
&= \mu v_{t-1} + (1-\mu)\frac{\eta}{1-\mu} * \hat{g}_t \\
\Phi_{t+1} &= \Phi_t - v_t
\end{aligned}
$$

In PyTorch vanilla SGD the temperature depends on all three hyper parameters — μ, B, and η (not good).

Heavy Ball Momentum

For PyTorch vanilla SGD we can set η by

$$\eta = (1 - \mu)B\eta_0$$

For academic batch sizes this setting of η makes the temperature depend on η_0 independent of μ and B.

Adaptive SGD with Momentum (Adam)

Adam introduces even more hyper-parameters.

Adam is derived from RMSProp which first appeared in lecture slides by Hinton.

"Adaptive" means that Different learning rates are used for different parameters.

Adam

Adam uses the EMA momentum.

PyTorch RMSProp uses heavy ball parameterization (not good).

The choice of momentum parameterization may be an important reason Adam is prefered over RMSprop in practice — hyper-parameter tuning is much easier when temperature is determined by a single parameter.

Adam (Without Bias Correction)

$$\tilde{g}_t[i] = \beta_g \, \tilde{g}_{t-1}[i] + (1-\beta_g)\hat{g}_t[i] \quad \beta_g \text{ typically .9 or .99}$$

$$s_t[i] = \beta_s s_{t-1}[i] + (1-\beta_s)\hat{g}_t[i]^2 \quad \beta_s \text{ typically .99 or .999}$$

$$\Phi_{t+1}[i] = \Phi_t[i] - \eta \, \frac{\tilde{g}_t[i]}{\sqrt{s_t[i]} + \epsilon}$$

Adam as Gradient Normalization

We can set $\beta_g = 0$ without influencing temperature. Hence Adam should be approximately equivalent to

$$s_t[i] = \beta_s s_{t-1}[i] + (1 - \beta_s)\hat{g}_t[i]^2$$

$$\Phi_{t+1}[i] = \Phi_t[i] - \eta \frac{g_t[i]}{\sqrt{s_t[i]} + \epsilon}$$

While normalization layers normalize the values, Adam normalizes the gradients.

Bias Correction

Consider a standard EMA.

$$\tilde{x}_0 = 0$$

$$\tilde{x}_t = \left(1 - \frac{1}{N}\right) \tilde{x}_{t-1} + \left(\frac{1}{N}\right) x_t$$

For $t < N$ the average $\tilde{x}_t$ will be strongly biased toward zero.

Bias Correction

The following running average maintains the invariant that $\tilde{x}_t$ is exactly the average of $x_1, \ldots, x_t$.

$$\tilde{x}_t = \left(\frac{t-1}{t}\right)\tilde{x}_{t-1} + \left(\frac{1}{t}\right)x_t$$

$$= \left(1 - \frac{1}{t}\right)\tilde{x}_{t-1} + \left(\frac{1}{t}\right)x_t$$

We now have $\tilde{x}_1 = x_1$ independent of any x_0.

But this fails to track a moving average for $t >> N$.

Bias Correction

The following avoids the initial bias toward zero while still tracking a moving average.

$$\tilde{x}_t = \left(1 - \frac{1}{\min(N, t)}\right) \tilde{x}_{t-1} + \left(\frac{1}{\min(N, t)}\right) x_t$$

The PyTorch version of Adam has a more subtle form of bias correction which yields the same effect.

Adam

$$\tilde{g}_t[i] = \left(1 - \frac{1}{\min(t, N_g)}\right) \tilde{g}_{t-1}[i] + \frac{1}{\min(t, N_g)} \hat{g}_t[i]$$

$$s_t[i] = \left(1 - \frac{1}{\min(t, N_s)}\right) s_{t-1}[i] + \frac{1}{\min(t, N_s)} \hat{g}_t[i]^2$$

$$\Phi_{t+1}[i] = \Phi_t - \frac{\eta}{\sqrt{s_t[i]} + \epsilon} \ \tilde{g}_t[i]$$

Decoupling Hyper-Parameters

The following reparameterization should be helpful for hyper-parameter tuning for Adam. Here the independent hyperparameters are N_g^0, N_s^0, B, ϵ_0 and η_0. Temperature should depend primarily on η_0 independent of the other parameters and the procedure gracefully converges to vanilla SGD in the limit as $\epsilon_0 \to \infty$.

$$
\begin{aligned}
N_g &= (1 - \beta_g) = \min(1, N_g^0/B) \\
N_s &= (1 - \beta_s) = \min(1, N_s^0/B) \\
\epsilon &= \epsilon_0 \sqrt{B} \\
\eta &= \epsilon B \eta_0
\end{aligned}
$$

Stochastic Gradient Descent (SGD)

The Classical Convergence Theorem

The Learning Rate as Temperature

Temperature, Batch Size, Momentum, and Adam

END