TTIC 31230, Fundamentals of Deep Learning

David McAllester, Autumn 2023

Stochastic Gradient Descent (SGD)

The Classical Convergence Theorem

The Learning Rate as Temperature

Temperature, Batch Size, Momentum, and Adam

Vanilla SGD

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

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

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

η_t is the learning rate for time t.

Issues

- **Gradient Noise:** The accuracy of $\hat{g}$ as an estimate of g.
- **Gradient Drift:(second order structure)** The fact that g changes as the parameters change.
- **Convergence:** Convergence requires $\eta_t \to 0$.
- **Exploration:** Reducing η slowly allows better exploration of model parameters.

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 $E_{(x,y)\sim\text{Train}} \mathcal{L}(\Phi, x, t)$ 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 $\lim_{t \to 0} \eta_t = 0$ 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.

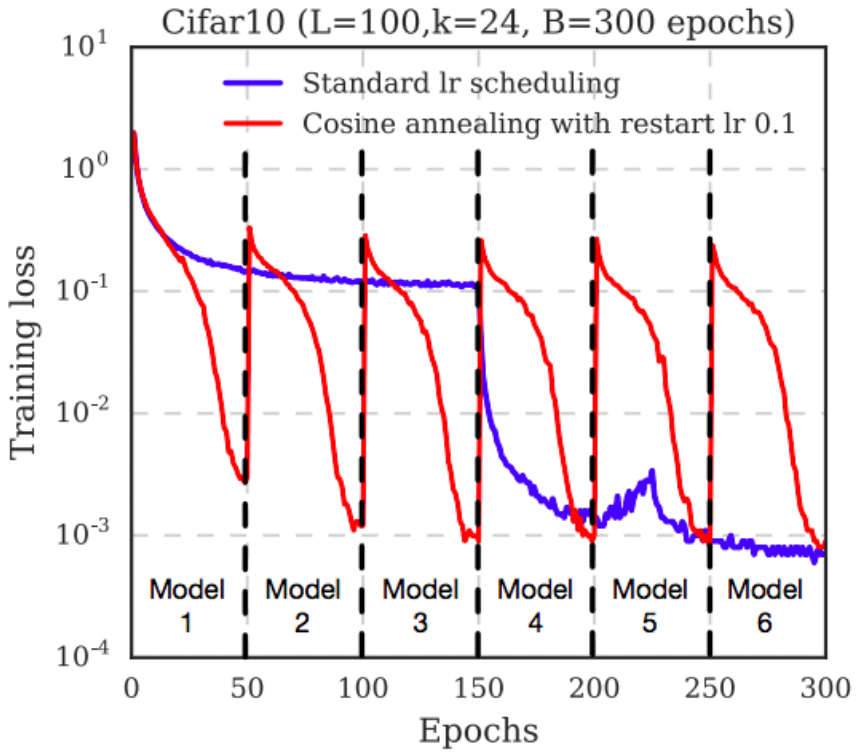
An arbitrarily large sample will become arbitrarily accurate as an estimate of the full gradient.

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.

SGD as a form of MCMC

Learning Rate as a Temperature Parameter



Gao Huang et. al., ICLR 2017

Temperature

Physical temperature is a relationship between the energy and 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.

Temperature

Boltzman's constant can be measured using the ideal gas law.

$$pV = NkT$$

$$\begin{aligned} p &= \text{pressure} \\ V &= \text{volume} \\ N &= \text{the number of molecules} \\ T &= \text{temperature} \\ k &= \text{Boltzman's constant} \end{aligned}$$

We can measure p, V, N and T and solve for k.

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).

Loss as Energy

Learning Rate as Temperature

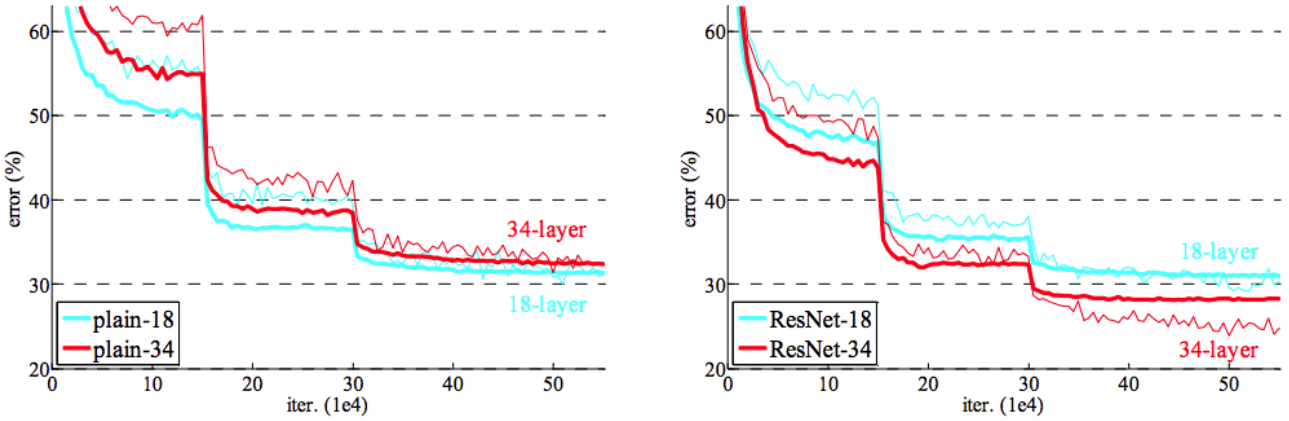
A finite learning rate defines an equalibrium probability distribution (or density) over the model parameters.

Each value of the model parameters has an associated loss.

The distribution over model parameters defines a distribution over loss.

Loss as Energy

Learning Rate as Temperature

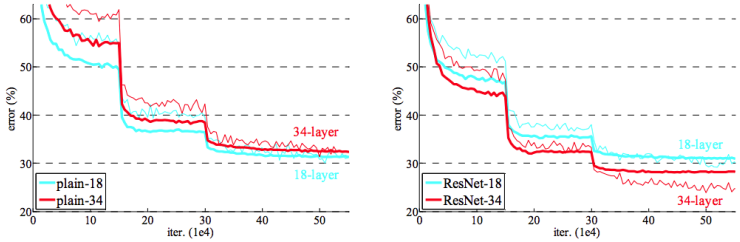


Equalibrium energy (loss) distributions at three different temperatures (learning rates).

Plots are from the ResNet paper. Left plot is for CNNs without residual skip connections, the right plot is ResNet. Thin lines are training error, thick lines are validation error. In all cases η is reduced twice, each time by a factor of 2.

Loss as Energy

Learning Rate as Temperature



The learnng rate is always reduced over time. The profile of learning rate reduction is called **the learning rate schedule**. An older convention (shown here) is to reduce the learning rate by a factor of 2 in steps.

Modern transformer training first quickly and smoothly ramps up the learning rate (the warm-up phase) up and then slowly and smoothly ramps it down.

Batch Size and Temperature

Vanilla SGD with minibatching typically uses the following update which defines the meaning of η.

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

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

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

Under this update **increasing the batch size (while holding η fixed) reduces the temperature.**

Making Temperature Independent of B

For batch size 1 with learning rate η_0 we have

$$\Phi_{t+1} = \Phi_t - \eta_0 \, \nabla_\Phi \mathcal{L}(t, \Phi_t)$$

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

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.

EMA Momentum

Momentum in general is equivalent to using an exponential moving average (EMA) of the gradient.

EMA momentum is parameterized directly as an EMA.

Traditional momentum (momentum in Vanilla SGD in PyTorch) defines the parameters differently.

The Adam optimizer uses the EMA parameterization which is cleaner and which we describe first.

Exponential Moving Average (EMA)

Consider a sequence $x_1, x_2, x_3, \ldots$.

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

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

The corresponding **exponential moving average** is

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

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

Exponential Moving Average (EMA)

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

$$= \frac{1}{N} \sum_{s=1}^{t} \left[\left(1 - \frac{1}{N}\right)^{t-s} x_s \right] \approx \frac{1}{N} \sum_{s=1}^{t} e^{-(s-t)/N} x_s$$

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

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 models typical values for β are .9, .99 or .999 corresponding to N being 10, 100 or 1000.

EMA Momentum:

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

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

$$= \beta \tilde{g}_{t-1} + (1 - \beta)\hat{g}_t$$

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

EMA Momentum: Temperature is Independent of N

The temperature is determined by the total effect of a single training gradient $\hat{g}$. With EMA momentum the totat contribution of a single $\hat{g}_t$ is

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

Hence the gradient estimate $\hat{g}$ has in fluence η for any value of N.

If N is small the model does not change significantly in N updates and temperature is independent of momentum parameter N.

Momentum

The theory of momentum is generally given in terms of second order structure (gradient drift).

Taking second order structure into account (where the model changes over N iterations) momentum can effect the training.

But the EMA parameterization reduces the effect of N on temperature.

Vanilla SGD Momentum Parameterization

The standard (PyTorch) momentum SGD equations are

$$v_t = \mu v_{t-1} + \eta * \hat{g}_t \quad \mu \text{ is typically .9 or .99}$$

$$\Phi_{t+1} = \Phi_t - v_t$$

Here v is velocity, $0 \leq \mu < 1$ represents friction drag and $\eta\hat{g}$ is the acceleration generated by the gradient force.

Decoupling Learning rate, Batch Size, and Momentum (Vanilla SGD parameterization of Momentum)

$$v_t = \mu v_{t-1} + \eta * \hat{g}_t \quad \mu \text{ is typically .9 or .99}$$

$$\Phi_{t+1} = \Phi_t - v_t$$

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

Momentum and Temperature

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

Emprical evidence for this setting of η is given in

Don't Decay the Learning Rate, Increase the Batch Size, Smith et al., 2018

Adam

Adaptive SGD with Momentum (Adam).

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 parameterization of momentum.

PyTorch RMSProp also has momentum but with the vanilla parameterization.

This choice of momentum parameterization may be an important reason Adam is prefered in practice.

Adam (Without Bias Correction)

$$\tilde{g}_t[i] = \left(1 - \frac{1}{N_s}\right) \tilde{g}_{t-1}[i] + \frac{1}{N_s} \hat{g}_t[i] \quad N_s \text{ typically 100 or 1000}$$

$$s_t[i] = \left(1 - \frac{1}{N_s}\right) s_{t-1}[i] + \frac{1}{N_s} \hat{g}_t[i]^2 \quad N_s \text{ typically 100 or 1000}$$

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

Adam as Gradient Normalization

One can think of Adam as gradient normalization.

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

Adam (Centered)

$$\tilde{g}_t[i] = \left(1 - \frac{1}{N_s}\right) tildeg_{t-1}[i] + \frac{1}{N_s}\hat{g}_t[i] \quad N_s \text{ typically 100 or 1000}$$

$$s_t[i] = \left(1 - \frac{1}{N_s}\right) s_{t-1}[i] + \frac{1}{N_s}\hat{g}_t[i]^2 \quad N_s \text{ typically 100 or 1000}$$

$$\Phi_{t+1}[i] = \Phi_t[i] - \frac{\eta}{\sqrt{s_t[i] - \tilde{g}[i]^2 + \epsilon}} \; \tilde{g}_t[i]$$

A Noise Analysis of Centered Adam

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

One interpretation of Adam involves the gradient noise variance $\sigma[i]$.

A low-noise measurement estimate of the gradient is more certain.

A more certain gradient estimate needs less averaging.

Less averaging is equivalent to a larger learning rate.

Problem with the Analysis

$$\Phi[i] \text{ -= } \eta \frac{\tilde{g}[i]}{\sqrt{E\hat{g}^2}} \quad (1) \qquad\qquad \Phi[i] \text{ -= } \eta \frac{\tilde{g}[i]}{\sigma^2[i]} \quad (2)$$

The noise analysis yields (2) but (1) works better.

The "gradient normalization" interpretation seems more relevant to practice.

However, the theory of gradient normalization seems unclear.

Bias Correction

Consider a standard moving average.

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

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

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 published 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 Hyperparameters

The following reparameterization should be helpful for Adam.

$$N_g = \min(1, N_g^0/B)$$

$$N_s = \min(1, N_s^0/B)$$

$$\epsilon = \epsilon_0\sqrt{B}$$

$$\eta = \epsilon B \eta_0$$

Stochastic Gradient Descent (SGD)

The Classical Convergence Theorem

The Learning Rate as Temperature

Temperature, Batch Size, Momentum, and Adam

END