TTIC 31230, Fundamentals of Deep Learning

David McAllester, Winter 2019

Stochastic Gradient Descent (SGD)

The Classical Convergence Theorem

Momentum, RMSProp, and Adam

Hyperparameter Conjugacy

Vanilla SGD

$$\Phi \texttt{ -= } \eta \hat{g}$$

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

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

Issues

- **Gradient Estimation.** 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.** To converge to a local optimum the learning rate must be gradually reduced toward zero.
- **Exploration.** Since deep models are non-convex we need to search over the parameter space. SGD can behave like MCMC.

A One Dimensional Example

Suppose that y is a scalar, and consider

$$\begin{aligned}
\text{loss}(\beta, y) &= \frac{1}{2}(\beta - y)^2 \\
g &= \nabla_\beta \, E_{y \sim \text{Pop}} \, \frac{1}{2}(\beta - y)^2 \\
&= \beta - E_{y \sim \text{Pop}} \, y \\
\hat{g} &= \beta - E_{y \sim \text{Batch}} \, y
\end{aligned}$$

Even if β is optimal, for a finite batch we will have $\hat{g} \neq 0$.

The Classical Convergence Theorem

$$\Phi \texttt{ -= } \eta_t \nabla_\Phi \text{ loss}(\Phi, x_t, y_t)$$

For "sufficiently smooth" non-negative loss with

$$\eta_t > 0 \quad \text{and} \quad \lim_{t \to \infty} \eta_t = 0 \quad \text{and} \quad \sum_t \eta_t = \infty,$$

we have that the training loss of Φ converges (in practice Φ converges to a local optimum of training loss).

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

Rigor Police: This theorem is usually stated with an additional condition that $\sum_t \eta_t^2 < \infty$. I believe that this additional condition is not required.

Physicist's Proof of the Convergence Theorem

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

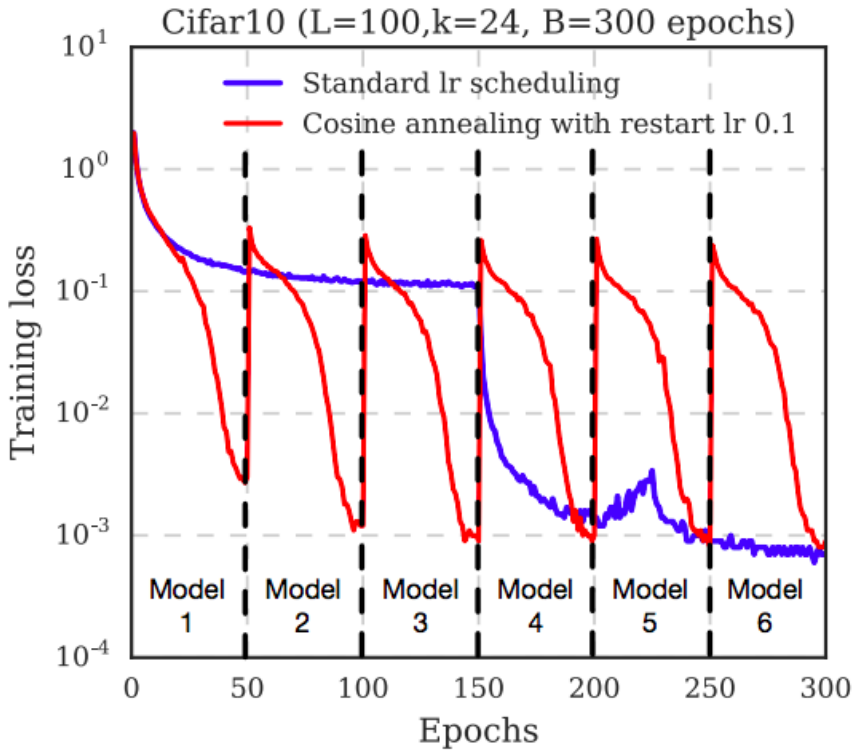
For sufficiently small learning rates any meaningful update of the parameters will be based on an arbitrarily large sample of gradients at essentially the same parameter value.

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.

SGD as a form of MCMC

Learning Rate as a Temperature Parameter



Gao Huang et. al., ICLR 2017

Standard Non-Vanilla SGD Algorithms

Digression on Running Averages

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

For $t \geq N$, consider the average of the N most recent values.

$$\tilde{\mu} = \frac{1}{N} \sum_{s=t-N+1}^{t} x_s$$

This can be approximated more efficiently with

$$\hat{\mu}_0 = 0$$

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

$$\hat{\mu}_t \approx \tilde{\mu}_t \quad t > N$$

Running Averages

More explicitly, for $\hat{\mu}_0 = 0$, the update

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

gives

$$\hat{\mu}_t = \frac{1}{N} \sum_{1 \leq s \leq t} \left(1 - \frac{1}{N}\right)^{t-s} x_s$$

where we have

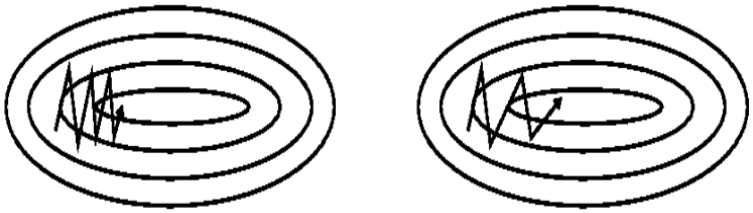
$$\sum_{n \geq 0} \left(1 - \frac{1}{N}\right)^{-n} = N$$

Momentum

$$v_t = \left(1 - \frac{1}{N_g}\right) v_{t-1} + \eta * \hat{g}_t \quad N_g \text{ is typically 10 or 100}$$

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

The theory of momentum is generally given in terms of second order structure.



Rudin's blog

Momentum

However, second order analyses are controversial in Deep Learning.

We can perhaps get insight by reparameterizing the momentum equations.

We are Entering a Twilight Zone (TZ)

The twilight zone is material for which I do not know of a reference.

Conjugacy of SGD Hyperparameters

For an objective function $f(x, y)$ we say that x and y are conjugate if changing x does effect the loss gradient for y.

In other words if

$$\frac{\partial f(x, y)}{\partial x \partial y} = 0$$

We are seeking a nonlinear transformation on hyperparameters yielding robust conjugacy.

Solving for η in terms of N_g (TZ)

Let η' be a learning rate that is appropriate without momentum ($N_g = 1$).

We will solve for a learning rate η as a function of N_g so as to minimize the perturbation of SGD as we increase N_g.

An Equivalent Formulation of Momentum

$$
\begin{aligned}
v_t &= \left(1 - \frac{1}{N_g}\right) v_{t-1} + \eta \hat{g}_t \\
&= \left(1 - \frac{1}{N_g}\right) v_{t-1} + \frac{1}{N_g}(N_g \eta \hat{g}_t)
\end{aligned}
$$

$$
v_t = N_g \eta \mu_t \quad \text{for } \mu_t = \left(1 - \frac{1}{N_g}\right) \mu_{t-1} + \frac{1}{N_g} \hat{g}_t
$$

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

$$
\Phi_{t+1} = \Phi_t - N_g \eta \mu_t
$$

Solving for η (TZ)

$$\mu_t = \left(1 - \frac{1}{N_g}\right) \mu_{t-1} + \frac{1}{N_g} \hat{g}_t$$

$$\text{we have } \Phi_{t+1} = \Phi_t - N_g \eta \, \mu_t$$

$$\text{unperturbed is } \Phi_{t+1} = \Phi_t - \eta' \mu_t$$

This gives $\eta = \eta'/N_g$.

We expect that that η' and N_g exhibit more independence (conjugacy) than η, N_g.

Scaling η and N_g with Batch Size

Recent work has show that scaling hyper-parameters with the batch size can lead to effective learning with very large (highly parallel) batches.

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

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

Solving for η as a Function of B

Let η' be a learning rate appropriate for $N_g = 1$ and $B = 1$.

We will solve for η as a function of η' and B so as to minimize the perturbation of SGD as we increase B.

Solving for η as a function of B

Consider two consecutive updates for a batch size of 1 with learning rate η'.

$$\Phi_{t+1} = \Phi_t - \eta' \nabla_\Phi \text{loss}(\Phi_t, x_t, y_t)$$

$$\Phi_{t+2} = \Phi_{t+1} - \eta' \nabla_\Phi \text{loss}(\Phi_{t+1}, x_{t+1}, y_{t+1})$$

$$\approx \Phi_{t+1} - \eta' \nabla_\Phi \text{loss}(\Phi_t, x_{t+1}, y_{t+1})$$

$$= \Phi_t - \eta' ((\nabla_\Phi \text{loss}(\Phi_t, x_t, y_t)) + (\nabla_\Phi \text{loss}(\Phi_t, x_{t+1}, y_{t+1})))$$

Solving for η as a function of B

Let η be the learning rate for batch size B.

$$\Phi_{t+2} \approx \Phi_t - \eta'((\nabla_\Phi \text{loss}(\Phi_t, x_t, y_t)) + (\nabla_\Phi \text{loss}(\Phi_t, x_{t+1}, y_{t+1})))$$

$$= \Phi_t - 2\eta' \hat{g} \quad \text{for} \quad B = 2$$

We have Φ **-=** $\eta \hat{g}$

unperturbed is Φ **-=** $B\eta' \hat{g}$

This gives $\eta = B\eta'$.

We expect conjugacy for the parameters η' and B.

Solving for N_g as a Function of B

Let N_g' be a momentum parameter appropriate for $B = 1$.

We will solve for N_g as a function of B so as to minimize the perturbation of SGD as we increase B.

For batch size B, $\hat{\mu}_t$ is an average of $\hat{g}$ over $N_g B$ gradient values.

SGD seems minimally perturbed by

$$N_g B = N_g' \quad \text{or} \quad N_g = N_g'/B$$

We expect conjugacy for the parameters N_g' and B.

Putting It Together (TZ)

$$N_g = \min(1, N'_g/B)$$

$$\eta = B\eta'/N_g$$

We expect conjugacy for η', N'_g and B.

RMSProp

RMSProp is based on a running average of $\hat{g}[i]^2$ for each real-valued model parameter i.

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

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

RMSProp

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

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

For $B = 1$ we have $s[i] \approx E\ \hat{g}[i]^2 = g[i]^2 + \sigma[i]^2$

We should expect $\sigma[i] >> g[i]$.

We can therefore think of RMSProp as $\Phi[i]$ `-=` $\eta(\hat{g}[i]/\sigma[i])$

RMSProp is Theoretically Mysterious

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

Although (1) seems to work better, (2) is better motivated theoretically. To see this we can consider units.

If parameters have units of "weight", and loss is in bits, then (2) type checks with η having units of bits — the numerical value of η has no dependence on the choice of the weight unit.

Consistent with the dimensional analysis, many theoretical analyses support (2) over (1) contrary to apparent empirical performance.

Scaling η with Batch Size for RMSProp (TZ)

Current RMSProp framework implementations do not scale well with batch size.

The meaning of $\hat{g}_i^2$ changes when increasing B.

$\hat{g}_i$ is an average over a batch — averaging reduces variance.

$$E\ \hat{g}_i^2 = = \mu_i^2 + \frac{\sigma_i^2}{B}$$

Scaling η with Batch Size for RMSProp (TZ)

$$E\ \hat{g}_i^2 = \mu_i^2 + \frac{\sigma_i^2}{B}$$

To preserve the behavior of RMSProp while increasing B we need a running average of $g_{b,i}^2$ for individual batch elements b.

This requires augmenting backpropagation to compute an average (or sum) of $g_{b,i}^2$ as well as an average of $g_{b,i}$ within each batch.

Adam — Adaptive Momentum

Adam combines momentum and RMSProp.

It also uses "bias correction" of running averages.

A Digression on Bias Correction of Running Averages

Consider a sequence $x_1,\ x_2,\ x_3, \ldots$ and consider the following for N large.

$$\hat{\mu}_0 = 0$$

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

For $\mu \doteq E\ x$ we have $E\ \hat{\mu}_1 = \mu/N$.

For $t << N$ we have $E\ \hat{\mu}_t \approx (t/N)\mu$.

Bias Correction of Running Averages

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

$$\hat{\mu}_0 = 0$$

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

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

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

Bias Correction of Running Averages

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

$$\hat{\mu}_0 = 0$$

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

The published version of Adam has a more obscure form of bias correction which yields essentially the same effect.

Adam (simplified)

$$\mu_0[i] = s_0[i] = 0$$

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

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

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

END