On the Importance of Initialization and Momentum in Deep Learning

Ilya Sutskever, James Martens, George Dahl, Geoffrey Hinton

Presenters: Yuwen Xiong, Andrew Liao, Jingkang Wang

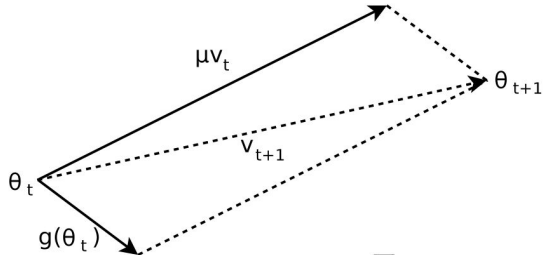
Background

- 2010-era deep networks (Lecture 4 & 7)
  - The difficulty of training DNNs had prevented the widespread use
  - Attract renewed attention following Hinton et al., (2006) - pre-training
  - Hessian-free Optimization (HF) works well without use of pre-training
  - In general, DNNs or RNNs were believed difficult to train from scratch using first-order methods due to various difficulties => **ill-conditioned cost functions**
- **This paper:**
  - More thorough investigation of training DNNs/RNNs
  - Rewrote the Nesterov's Accelerated Gradient (NAG) update rule and emphasize the connection to Classic Momentum (CM)
  - Discussion the relation of momentum and HF

CM and NAG

- Classic Momentum (CM)

$$v_{t+1} = \mu v_t - \varepsilon \nabla f(\theta_t)$$
$$\theta_{t+1} = \theta_t + v_{t+1}$$



learning rate ε, momentum coefficient μ

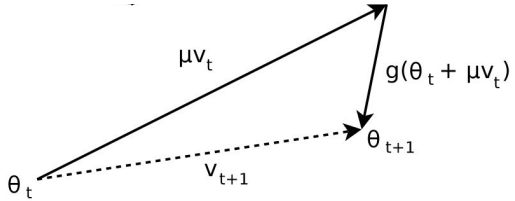
Why Momentum works?

In the high curvature directions, the gradients cancel each other out, so momentum dampens the

In the low curvature directions, the gradients are amplified, so momentum speeds up the update.

- Nesterov's Accelerated Gradient (NAG)

$$v_{t+1} = \mu v_t - \varepsilon \nabla f(\theta_t + \mu v_t)$$
$$\theta_{t+1} = \theta_t + v_{t+1}$$



Accelerated Convergence

- Deterministic gradients (general smooth convex function)

NAG $O(1/T^2)$ **vs. Gradient Descent** $O(1/T)$

- Stochastic gradients (general smooth convex function)

SGD $O(L/T + \sigma/\sqrt{T})$

NAG $O(L/T^2 + \sigma/\sqrt{T})$

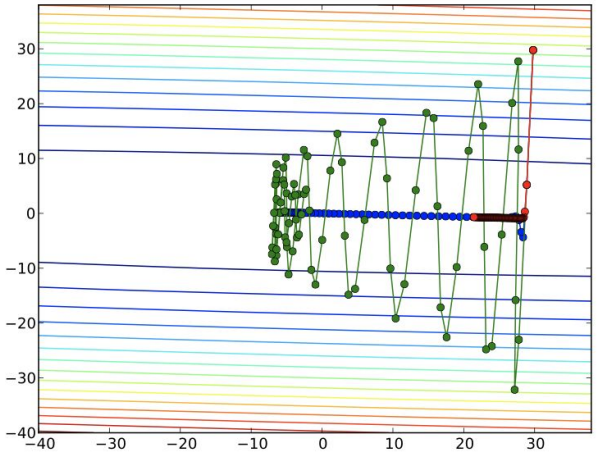
L : Lipschitz coefficient of gradients ∇f

σ : Variance in the gradient estimate

Relationship between CM and NAG

- Calculating velocity by applying a gradient-based correction to the previous velocity vector
  - CM: computes the gradient update from the current position
  - NAG: use $\theta_t + \mu v_t$ to approximate the next position θ_{t+1}
  - NAG allows for changing velocity in a quicker and more responsive way

- Two-dimensional oblong quadratic objective
  - Blue: NAG, Green: CM
  - CM exhibits large oscillations
  - NAG is able to avoid these oscillations almost entirely
  - NAG more tolerant of larges of μ compared to CM



Relationship between CM and NAG

- Positive definite quadratic objective

$$q(x) = x^\top Ax/2 + b^\top x$$

- Reparameterize $q(x)$ under the basis of eigenvectors of A

$$A = U^\top DU$$

$$p(y) \equiv q(x) = q\left(U^\top y\right) = y^\top U\left(U^\top DU\right)U^\top y/2 + b^\top U^\top y$$

$$= y^\top Dy/2 + c^\top y \quad \text{where } c = Ub$$

$$p(y) = \sum_{i=1}^{n} [p]_i([y]_i), \text{ where } [p]_i(t) = \lambda_i t^2/2 + [c]_i t$$

Relationship between CM and NAG

- **Theorem 2.1.** *Let* $p(y) = \sum_{i=1}^{n} [p]_i([y]_i)$ *such that* $[p]_i(t) = \lambda_i t^2/2 + c_i t$. *Let* ε *be arbitrary and fixed. Denote by* $CM_{\mathrm{x}}(\mu, p, y, v)$ *and* $CM_{\mathrm{v}}(\mu, p, y, v)$ *the parameter vector and the velocity vector respectively, obtained by applying one step of CM (i.e., Eq. 1 and then Eq. 2) to the function* p *at point* y, *with velocity* v, *momentum coefficient* μ, *and learning rate* ε. *Define* NAG_{x} *and* NAG_{v} *analogously. Then the following holds for* $\mathrm{z} \in \{\mathrm{x}, \mathrm{v}\}$*:*

$$CM_{\mathrm{z}}(\mu, p, y, v) = \begin{bmatrix} CM_{\mathrm{z}}(\mu, [p]_1, [y]_1, [v]_1) \\ \vdots \\ CM_{\mathrm{z}}(\mu, [p]_n, [y]_n, [v]_n) \end{bmatrix}$$

$$NAG_{\mathrm{z}}(\mu, p, y, v) = \begin{bmatrix} CM_{\mathrm{z}}(\mu(1-\lambda_1\varepsilon), [p]_1, [y]_1, [v]_1) \\ \vdots \\ CM_{\mathrm{z}}(\mu(1-\lambda_n\varepsilon), [p]_n, [y]_n, [v]_n) \end{bmatrix}$$

- CM and NAG become equivalent when ϵ is small
- NAG uses smaller effective momentum for the high-curvature eigen-directions

Proof

- Induction for rotation invariant - operating independently over the different eigen directions
- Consider one step of CM and NAG

$$
\begin{aligned}
&CM_{\mathrm{v}}(\mu, p, y, v) = \mu v - \varepsilon \nabla p(y) \\
&= (\mu[v]_1 - \varepsilon \nabla_{[y]_1} p(y), \ldots, \mu[v]_n - \varepsilon \nabla_{[y]_n} p(y)) \\
&= (\mu[v]_1 - \varepsilon \nabla [p]_1([y]_1), \ldots, \mu[v]_n - \varepsilon \nabla [p]_n([y]_n)) \\
&= (CM_{\mathrm{v}}(\mu, [p]_1, [y]_1, [v]_1), \ldots, CM_{\mathrm{v}}(\mu, [p]_n, [y]_n, [v]_n))
\end{aligned}
$$

$$
\begin{aligned}
NAG_{\mathrm{v}}(\mu, [q]_i, y, v) &= \mu v - \varepsilon \nabla [p]_i (y + \mu v) \\
&= \mu v - \varepsilon(\lambda_i (y + \mu v) + c_i) \\
&= \mu v - \varepsilon \lambda_i \mu v - \varepsilon(\lambda_i y + c_i) \\
&= \mu(1 - \varepsilon \lambda_i) v - \varepsilon \nabla [p]_i (y) \\
&= CM_{\mathrm{v}}(\mu(1 - \varepsilon \lambda_i), [p]_i, v, y)
\end{aligned}
$$

Momentum and HF

- **Hessian-free optimization** (HF) minimizes the quadratic approximation using conjugate gradient (CG)

$$\begin{aligned}\mathcal{J}_{\text{loss}}(\mathbf{w}) &\approx \mathcal{J}(\mathbf{w}_0) + \nabla\mathcal{J}(\mathbf{w}_0)^\top(\mathbf{w}-\mathbf{w}_0) + \tfrac{1}{2}(\mathbf{w}-\mathbf{w}_0)^\top\mathbf{H}(\mathbf{w}-\mathbf{w}_0) \\ &= \nabla\mathcal{J}(\mathbf{w}_0)^\top\mathbf{w} + \tfrac{1}{2}(\mathbf{w}-\mathbf{w}_0)^\top\mathbf{H}(\mathbf{w}-\mathbf{w}_0) + \text{const} \\ \mathcal{Q}_{\text{quad}}(\mathbf{w}) &= \nabla\mathcal{J}(\mathbf{w}_0)^\top\mathbf{w} + \tfrac{1}{2}(\mathbf{w}-\mathbf{w}_0)^\top(\mathbf{G}+\lambda\mathbf{I})(\mathbf{w}-\mathbf{w}_0).\end{aligned}$$

- CG enjoys faster convergence than GD
  - works better when the cost function are more ill-conditioned
  - CG terminated after just 1 step, HF becomes equivalent to NAG - except that it uses a special formula based on the curvature matrix for the learning rate instead of a fixed constant
- Hybrid HF-Momentum (fixed learning rate, decaying ϵ and increasing μ, smaller mini-batch)

Training DNNs

- Deep Autoencoders
  - Sparse initialization (Martens, 2010)
    i. each random unit is connect to 15 randomly chosen units in the previous layer (weights drawn from a unit Gaussian, the biases are set to zero, for tanh units, the biases are set 0.5)
    ii. not easy to saturate, more diverse, better differentiation between units
  - Schedule for dampling parameter

$$\mu_t = \min(1 - 2^{-1-\log_2(\lfloor t/250 \rfloor + 1)}, \mu_{\max}) \qquad \mu_{max} \in \{0.999, 0.995, 0.99, 0.9, 0\}$$

  - NAG vs. HB momentum
    i. NAG outperforms CM (especially with large μ_{max}). NAG achieves results that are comparable with HF

task	$0_{(\mathrm{SGD})}$	0.9N	0.99N	0.995N	0.999N	0.9M	0.99M	0.995M	0.999M	$\mathrm{SGD_C}$	$\mathrm{HF}^{\dagger}$	HF^{*}
Curves	0.48	0.16	0.096	0.091	**0.074**	0.15	0.10	0.10	0.10	0.16	0.058	0.11
Mnist	2.1	1.0	**0.73**	0.75	0.80	1.0	0.77	0.84	0.90	0.9	0.69	1.40
Faces	36.4	14.2	8.5	7.8	**7.7**	15.3	8.7	8.3	9.3	NA	7.5	12.0

Training DNNs

- Deep Autoencoders
  - Reducing the momentum coefficient at the final stage
    i. Reducing to 0.9 during the final 1000 parameter updates with learning rate unchanged
    ii. reducing the momentum coecient allows for finer convergence to take place
    iii. Consistent with Darken & Moody 1993

problem	before	after
Curves	0.096	0.074
Mnist	1.20	0.73
Faces	10.83	7.7

Table 2. The effect of low-momentum finetuning for NAG. The table shows the training squared errors before and after the momentum coefficient is reduced. During the primary ("transient") phase of learning we used the optimal momentum and learning rates.

Training RNNs

- Echo-State Networks (ESNs)
  - A family of RNNs with an unusually simple training method
    i. Recurrent connections are fixed and not learned (random initialization)

$$h_t = W_{ih}^\top x_t + W_{hh}^\top h_{t-1}$$
$$o_t = tanh(h_t)$$

  - Initialization of recurrent connection (Jaeger & Haas, 2004) - spectral radius of
    i. **smaller than 1**: tendency to “forget” whatever input signal exposed
    ii. **much larger than 1**: oscillatory and chaotic, severe exploding gradients
    iii. **slightly larger than 1**: generate response for varied input histories, set 1.1 in experiment
  - Initial scale of input-to-hidden connections
    i. **too large**: “overwritten” by irrelevant signals
    ii. **too small**: significantly slower learning
    iii. good choice depends on particular task

connection type	sparsity	scale
in-to-hid (add,mul)	dense	$0.001 \cdot N(0,1)$
in-to-hid (mem)	dense	$0.1 \cdot N(0,1)$
hid-to-hid	15 fan-in	spectral radius of 1.1
hid-to-out	dense	$0.1 \cdot N(0,1)$
hid-bias	dense	0
out-bias	dense	average of outputs

Training RNNs

- Echo-State Networks (ESNs)
  - Bit memorization, addition and multiplication problem
    i. $\mu = 0.9$ for the first 1000 iterations, after which $\mu \in \{0, 0.9, 0.98, 0.995\}$
    ii. for each setting, use the empirically best learning rate chosen from $\{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$
  - Results
    i. RNNs can be successfully and robustly trained with appropriate initialization & momentum
    ii. Specific initialization, NAG, large momentum, small learning rate
    iii. Spectral radius of recurrent units is set to be around 1.
    iv. Proper scale of the input-to-hidden connections
    v. Initialize the output bias such that the output is centered

Thank you!