Derivation of the Adam Optimization Algorithm

MHJ

Notes 2018

Overview

Why Combine Momentum and RMSProp?

Motivation for Adam: Adaptive Moment Estimation (Adam) was introduced by Kingma & Ba (2014) to combine the benefits of momentum and RMSProp

- **Momentum:** Fast convergence by smoothing gradients (accelerates in long-term gradient direction).
- **Adaptive rates (RMSProp):** Per-dimension learning rate scaling for stability (handles different feature scales, sparse gradients).
- Adam uses both: maintains moving averages of both first moment (gradients) and second moment (squared gradients)
- Additionally, includes a mechanism to correct the bias in these moving averages (crucial in early iterations)
- Result: Adam is robust, achieves faster convergence with less tuning, and often outperforms SGD (with momentum) in practice

Adam: Exponential Moving Averages (Moments)

Adam maintains two moving averages at each time step t for each parameter w:

First moment (mean) m_t

$$m_t = \beta_1\, m_{t-1} + (1-\beta_1)\, \nabla L(w_t), \quad \text{(Momentum term)}$$

Second moment (uncentered variance) v_t

$$v_t = \beta_2\, v_{t-1} + (1-\beta_2)\, (\nabla L(w_t))^2, \quad \text{(RMS term)}$$

with typical $\beta_1 = 0.9$, $\beta_2 = 0.999$. Initialize $m_0 = 0$, $v_0 = 0$.

These are *biased* estimators of the true first and second moment of the gradients, especially at the start (since m_0, v_0 are zero)

Adam: Bias Correction

To counteract initialization bias in m_t, v_t, Adam computes bias-corrected estimates

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \qquad \hat{v}_t = \frac{v_t}{1-\beta_2^t}.$$

- When t is small, $1-\beta_i^t \approx 0$, so $\hat{m}_t, \hat{v}_t$ significantly larger than raw m_t, v_t, compensating for the initial zero bias.
- As t increases, $1-\beta_i^t \to 1$, and $\hat{m}_t, \hat{v}_t$ converge to m_t, v_t.
- Bias correction is important for Adam's stability in early iterations

Adam: Update Rule Derivation

Finally, Adam updates parameters using the bias-corrected moments:

$$w_{t+1} \;=\; w_t \;-\; \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon}\,\hat{m}_t\,,$$

where ϵ is a small constant (e.g. 10^{-8}) to prevent division by zero
Breaking it down:

1. Compute gradient $\nabla L(w_t)$.
2. Update first moment m_t and second moment v_t (exponential moving averages).
3. Bias-correct: $\hat{m}_t = m_t/(1-\beta_1^t)$, $\hat{v}_t = v_t/(1-\beta_2^t)$.
4. Compute step: $\Delta w_t = \frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon}$.
5. Update parameters: $w_{t+1} = w_t - \alpha\,\Delta w_t$.

This is the Adam update rule as given in the original paper

Adam vs. AdaGrad and RMSProp

- **AdaGrad:** Uses per-coordinate scaling like Adam, but no momentum. Tends to slow down too much due to cumulative history (no forgetting)
- **RMSProp:** Uses moving average of squared gradients (like Adam's v_t) to maintain adaptive learning rates, but does not include momentum or bias-correction.
- **Adam:** Effectively RMSProp + Momentum + Bias-correction
  - Momentum (m_t) provides acceleration and smoother convergence.
  - Adaptive v_t scaling moderates the step size per dimension.
  - Bias correction (absent in AdaGrad/RMSProp) ensures robust estimates early on.
- In practice, Adam often yields faster convergence and better tuning stability than RMSProp or AdaGrad alone

Adaptivity Across Dimensions

- Adam adapts the step size *per coordinate*: parameters with larger gradient variance get smaller effective steps, those with smaller or sparse gradients get larger steps.
- This per-dimension adaptivity is inherited from AdaGrad/RMSProp and helps handle ill-conditioned or sparse problems.
- Meanwhile, momentum (first moment) allows Adam to continue making progress even if gradients become small or noisy, by leveraging accumulated direction.
- **Example:** In a deep network, some weights may receive very noisy or infrequent gradients – Adam will keep their learning rate high (unlike AdaGrad which would have decayed it) and also smooth out the noise with momentum.