Variable Metric Subgradient Methods: From Adagrad to Adam

Mert Pilanci
(based on material from Stephen Boyd and John Duchi)

EE364b, Stanford University

Outline

- Variable metric subgradient methods
- Projected subgradient method
- Adagrad
- RMSProp
- Adam
- Adamw
- (Non)convergence of adam

Variable metric subgradient methods

subgradient method with variable metric $H_k \succ 0$:

(1) get subgradient $g^{(k)} \in \partial f(x^{(k)})$
(2) update (diagonal) metric H_k
(3) update $x^{(k+1)} = x^{(k)} - H_k^{-1} g^{(k)}$

- matrix H_k generalizes step-length α_k

there are many such methods (Ellipsoid method, AdaGrad, . . .)

Variable metric projected subgradient method

same, with projection carried out in the H_k metric:

(1) get subgradient $g^{(k)} \in \partial f(x^{(k)})$
(2) update (diagonal) metric H_k
(3) update $x^{(k+1)} = P_{\mathcal{X}}^{H_k}\left(x^{(k)} - H_k^{-1} g^{(k)}\right)$

where

$$\Pi_{\mathcal{X}}^{H}(y) = \operatorname*{argmin}_{x \in \mathcal{X}} \|x - y\|_H^2$$

and $\|x\|_H = \sqrt{x^T H x}$.

AdaGrad

AdaGrad: adaptive subgradient method

(1) get subgradient $g^{(k)} \in \partial f(x^{(k)})$

(2) choose metric H_k:

- set $S_k = \sum_{i=1}^{k} \mathbf{diag}(g^{(i)})^2$
- set $H_k = \frac{1}{\alpha} S_k^{\frac{1}{2}}$

(3) update $x^{(k+1)} = P_{\mathcal{X}}^{H_k}\left(x^{(k)} - H_k^{-1} g^{(k)}\right)$

where $\alpha > 0$ is step-size

AdaGrad: Duchi, Hazan, Singer (2011).

AdaGrad: derivation

- for fixed $H_k = H$ we have estimate:

$$f_{\text{best}}^{(k)} - f^\star \leq \frac{1}{2k}(x^{(1)} - x^\star)^T H (x^{(1)} - x^\star) + \frac{1}{2k}\sum_{i=1}^{k} \|g^{(i)}\|_{H^{-1}}^2$$

- **idea**: Choose *diagonal* $H_k \succ 0$ that minimizes this estimate in hindsight:

$$H_k = \operatorname*{argmin}_h \max_{x,y \in C} (x-y)^T \mathbf{diag}(h)(x-y) + \sum_{i=1}^{k} \|g^{(i)}\|_{\mathbf{diag}(h)^{-1}}^2$$

- optimal $H_k = \frac{1}{R_\infty}\mathbf{diag}\left(\sqrt{\sum_{i=1}^k (g_1^{(i)})^2}, \ldots, \sqrt{\sum_{i=1}^k (g_n^{(i)})^2}\right)$
- **intuition**: adapt step-length based on historical step lengths

From AdaGrad to Adam: what changed?

AdaGrad uses all past squared gradients:

$$s^{(k)} = \sum_{i=1}^{k} g^{(i)} \circ g^{(i)}, \quad x^{(k+1)} = x^{(k)} - \alpha \frac{g^{(k)}}{\sqrt{s^{(k)}} + \epsilon}$$

where division and square root are componentwise.

- coordinate j learning rate:

$$\eta^{(k)} j = \frac{\alpha}{\sqrt{\sum i = 1^{k} (g_j^{(i)})^2} + \epsilon}$$

- $s^{(k)}$ is monotone increasing, so $\eta_j^{(k)}$ is monotone decreasing
- good for sparse gradients: rarely used coordinates keep large steps
- problem in dense/nonstationary problems: old gradients are remembered forever, so steps can become too small

Step 1: finite memory instead of full memory

Replace AdaGrad's cumulative sum by an exponential moving average:

$$s^{(k)} = \sum_{i=1}^{k} g^{(i)} \circ g^{(i)} \qquad \Longrightarrow \qquad v^{(k)} = \beta_2 v^{(k-1)} + (1-\beta_2) g^{(k)} \circ g^{(k)}.$$

- $\beta_2 \in [0, 1)$ controls memory

$$v^{(k)} = (1-\beta_2) \sum_{i=1}^{k} \beta_2^{k-i} g^{(i)} \circ g^{(i)}$$

- effective window length is roughly $1/(1-\beta_2)$
- update becomes RMSProp:

$$x^{(k+1)} = x^{(k)} - \alpha \frac{g^{(k)}}{\sqrt{v^{(k)}} + \epsilon}$$

- unlike AdaGrad, $v^{(k)}$ is not monotone; coordinate learning rates can increase

Step 2: smooth the gradient

Adam adds momentum to RMSProp:

$$m^{(k)} = \beta_1 m^{(k-1)} + (1-\beta_1) g^{(k)} \quad v^{(k)} = \beta_2 v^{(k-1)} + (1-\beta_2) g^{(k)} \circ g^{(k)}$$

with $m^{(0)} = 0$, $v^{(0)} = 0$.

- $m^{(k)}$ estimates the first moment of the gradient
- $v^{(k)}$ estimates the second raw moment of the gradient
- β_1 controls momentum memory, e.g. 0.9
- β_2 controls squared-gradient memory, e.g. 0.999

$$x^{(k+1)} = x^{(k)} - \alpha \frac{m^{(k)}}{\sqrt{v^{(k)}} + \epsilon}$$

This is RMSProp with a smoothed direction.

Adam: Kingma and Ba (2015).

Adam: bias correction

Because $m^{(0)} = 0$ and $v^{(0)} = 0$, the moving averages are biased toward zero at early iterations:

$$m^{(k)} = (1-\beta_1)\sum_{i=1}^{k}\beta_1^{k-i}g^{(i)}, \quad v^{(k)} = (1-\beta_2)\sum_{i=1}^{k}\beta_2^{k-i}g^{(i)} \circ g^{(i)}.$$

If the gradients were stationary, then

$$\mathbb{E}[m^{(k)}] \approx (1-\beta_1^k)\mathbb{E}[g], \qquad \mathbb{E}[v^{(k)}] \approx (1-\beta_2^k)\mathbb{E}[g \circ g].$$

Adam uses the corrected estimates

$$\hat{m}^{(k)} = \frac{m^{(k)}}{1-\beta_1^k}, \qquad \hat{v}^{(k)} = \frac{v^{(k)}}{1-\beta_2^k}.$$

Adam algorithm

Adam — adaptive moments

(1) get stochastic gradient $g^{(k)} = \nabla f_{i_k}(x^{(k)})$

(2) update moments:

$$m^{(k)} = \beta_1 m^{(k-1)} + (1-\beta_1) g^{(k)} \quad v^{(k)} = \beta_2 v^{(k-1)} + (1-\beta_2) g^{(k)} \circ g^{(k)}$$

(3) correct initialization bias:

$$\hat{m}^{(k)} = \frac{m^{(k)}}{1-\beta_1^k}, \qquad \hat{v}^{(k)} = \frac{v^{(k)}}{1-\beta_2^k}$$

(4) update:

$$x^{(k+1)} = x^{(k)} - \alpha_k \frac{\hat{m}^{(k)}}{\sqrt{\hat{v}^{(k)}} + \epsilon}$$

Adam as a variable metric method

Adam update can be written as

$$x^{(k+1)} = x^{(k)} - H_k^{-1} \hat{m}^{(k)}$$

with diagonal metric

$$H_k = \frac{1}{\alpha_k} \operatorname{diag}\left(\sqrt{\hat{v}^{(k)}} + \epsilon\right).$$

- $\hat{m}^{(k)}$ replaces $g^{(k)}$ by a momentum direction
- $\hat{v}^{(k)}$ chooses the coordinate-wise metric
- $\epsilon > 0$ prevents division by zero and limits very large steps
- coordinate-wise learning rate:

$$\eta_j^{(k)} = \frac{\alpha_k}{\sqrt{\hat{v}_j^{(k)}} + \epsilon}$$

With constraints,

$$x^{(k+1)} \;=\; \Pi_{\mathcal{X}}^{H_k}\left(x^{(k)} - H_k^{-1} \hat{m}^{(k)}\right).$$

Limiting cases

Adam connects the earlier methods:

- $\beta_1 = 0$: no momentum, giving RMSProp-type updates

$$x^{(k+1)} = x^{(k)} - \alpha_k \frac{g^{(k)}}{\sqrt{\hat{v}^{(k)}} + \epsilon}$$

- replace exponential average by a cumulative sum:

$$v^{(k)} \approx \sum_{i=1}^{k} g^{(i)} \circ g^{(i)}$$

  and recover AdaGrad
- $\beta_1 = \beta_2 = 0$:

$$x^{(k+1)} = x^{(k)} - \alpha_k \frac{g^{(k)}}{|g^{(k)}| + \epsilon}$$

  a sign/normalized-gradient type step
- large β_2: closer to long-memory AdaGrad; small β_2: more reactive, but less stable

Adam and L_2 regularization

Suppose we want to solve

$$\min_x \quad f(x) + \frac{\lambda}{2}\|x\|_2^2.$$

This is called weight decay regularization in ML. Using Adam on the regularized objective means replacing

$$g^{(k)} \quad \text{by} \quad g^{(k)} + \lambda x^{(k)}$$

inside both moment estimates:

$$\begin{aligned} m^{(k)} &= \beta_1 m^{(k-1)} + (1-\beta_1)(g^{(k)} + \lambda x^{(k)}) \\ v^{(k)} &= \beta_2 v^{(k-1)} + (1-\beta_2)(g^{(k)} + \lambda x^{(k)}) \circ (g^{(k)} + \lambda x^{(k)}). \end{aligned}$$

- for ordinary SGD, this is equivalent to weight decay
- for Adam, it is not: the $\lambda x^{(k)}$ term is scaled by the adaptive denominator
- regularization is mixed into both the direction and the metric

AdamW

AdamW decouples weight decay from the Adam gradient step.

(1) compute $g^{(k)}$ using the loss only, not including $\lambda x^{(k)}$

(2) form Adam moments $m^{(k)}$, $v^{(k)}$, $\hat{m}^{(k)}$, $\hat{v}^{(k)}$

(3) take the Adam step and separately decay the weights:

$$x^{(k+1)} = x^{(k)} - \alpha_k \frac{\hat{m}^{(k)}}{\sqrt{\hat{v}^{(k)}} + \epsilon} - \alpha_k \lambda x^{(k)}$$

equivalently,

$$x^{(k+1)} = (1 - \alpha_k \lambda) x^{(k)} - \alpha_k \frac{\hat{m}^{(k)}}{\sqrt{\hat{v}^{(k)}} + \epsilon}.$$

- λ controls shrinkage directly
- α_k controls optimization step length
- the decay is not divided by $\sqrt{\hat{v}^{(k)}} + \epsilon$

AdamW: Loshchilov and Hutter (2019).

Adam vs AdamW

Adam with L_2 regularization:

$$x^{(k+1)} = x^{(k)} - \alpha_k \frac{\widehat{m(g + \lambda x)}^{(k)}}{\sqrt{\widehat{v(g + \lambda x)}^{(k)}} + \epsilon}$$

AdamW:

$$x^{(k+1)} = (1 - \alpha_k \lambda) x^{(k)} - \alpha_k \frac{\widehat{m(g)}^{(k)}}{\sqrt{\widehat{v(g)}^{(k)}} + \epsilon}.$$

- Adam: regularization is adapted coordinate-wise
- AdamW: regularization is a uniform multiplicative shrinkage
- AdamW keeps the adaptive metric focused on the loss gradients
- AdamW usually makes λ easier to tune separately from α_k

Convergence: Adam can diverge

For AdaGrad,

$$H_k = \frac{1}{\alpha} \operatorname{diag}\left(\sqrt{\sum_{i=1}^{k} g^{(i)} \circ g^{(i)}}\right)$$

is monotone increasing, so this term is controlled.

For Adam/RMSProp,

$$H_k = \frac{1}{\alpha_k} \operatorname{diag}(\sqrt{v^{(k)}} + \epsilon), \qquad v^{(k)} = \beta_2 v^{(k-1)} + (1-\beta_2) g^{(k)} \circ g^{(k)}$$

need not be monotone.

- There is a simple convex optimization problem where Adam does not converge to the optimal solution (Reddi et al., On the Convergence of Adam and Beyond, 2019)

Convergence fix

Change only the second-moment denominator:

$$v^{(k)} = \beta_2 v^{(k-1)} + (1-\beta_2) g^{(k)} \circ g^{(k)}$$
$$\bar{v}^{(k)} = \max\{\bar{v}^{(k-1)}, v^{(k)}\}$$

where the maximum is componentwise.

Then update with

$$x^{(k+1)} = x^{(k)} - \alpha_k \frac{m^{(k)}}{\sqrt{\bar{v}^{(k)}} + \epsilon}.$$

- $\bar{v}^{(k)}$ is monotone increasing componentwise
- so the coordinate inverse learning rates are monotone
- this restores the AdaGrad-like long-term memory needed in the proof

For convex online optimization, AMSGrad obtains a similar bound of the same flavor as AdaGrad.

Reddi et al.

Takeaways

- AdaGrad:

  full memory of g^2monotone metric, clean convex theory

- RMSProp:

  exponential memory of g^2better adaptation, but nonmonotone metric

- Adam:

  RMSProp metric + momentum + bias correction

- AdamW:

  Adam + decoupled weight decay

- for convex convergence guarantees, the key issue is not momentum alone, but whether the adaptive denominator has enough long-term memory