MIT 6.7220/15.084 — Nonlinear Optimization Thu, Apr 11th 2024

Lecture 13
Adaptive preconditioning: AdaGrad and ADAM

Instructor: Prof. Gabriele Farina (✉ gfarina@mit.edu)*

In Lecture 12, we have seen how preconditioning the gradient with the inverse of the Hessian at the current point can significantly improve convergence, at least in the vicinity of a minimum with strong curvature. One of the effects of preconditioning with the Hessian was to make the descent direction invariant to linear transformations of the variables.

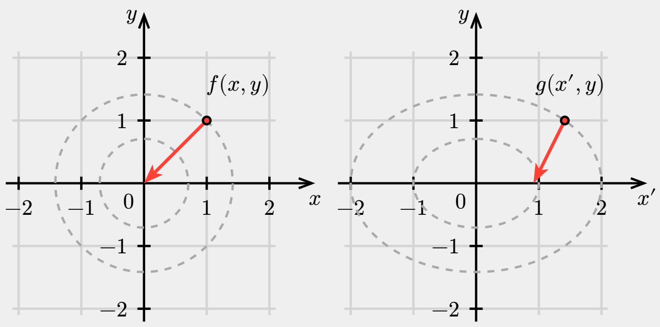
■ **A second look at preconditioning.** The affine-invariance property of Hessian preconditioning is extremely desirable. For example, consider the case of an optimization problem in which we have variables with physical meaning: maybe we are trying to design a bridge, and we have variables that denote lengths, wind strengths, weights, *et cetera*. In our modeling of the problem, we might have decided all length variables are in meters. If we now were to change our mind and decide to use centimeters instead, we would like the optimization algorithm to be invariant to this change, that is, produce the same sequence of iterates regardless of the units we use. While this would be guaranteed by the Hessian preconditioner, the same cannot be said for gradient descent.

Indeed, let us consider what would happen if we were to move all of our length variables from meters to centimeters. Since a centimeter is a tenth of a meter, the objective is now 10 times less sensitive to a change in length. This means that the gradient of the objective with respect to the new parameterization of lengths would now be 10 times smaller than before. And yet, all values of the length variables would be 10 times larger than before, producing a net effect of slowing down the gradient descent update on each of the variables by a factor of 100 if using the same learning rate! Adjusting the learning rate might compensate for this, at the expense of now affecting the speed of change of all other variables, even if they had been left untouched by the "meter → centimeter" reparameterization.

Example. As a small numerical example, consider the objective function (say, parameterized in meters) $f(x,y) = \frac{1}{2}x^2 + \frac{1}{2}y^2$. After a change of units of x, consider now the reparameterized objective

$$g(x', y) = f\left(\frac{x'}{\sqrt{2}}, y\right) = \frac{1}{4}{x'}^2 + \frac{1}{2}y^2,$$

plotted on the right.



The two plots show contour lines for the respective objectives, together with an identical initial point (up to reparameterization). The red arrows show the gradient descent direction at the initial point. It is clear that after one step of gradient descent, the two points will no longer be equivalent.

*These notes are class material that has not undergone formal peer review. The TAs and I are grateful for any reports of typos.

■ **Today's lecture.** In this lecture, we look at preconditioning algorithms that are not based on using the Hessian. Rather, they are based on the idea of adapting the learning rate for each parameter based on the historical gradients. Compared to the Hessian preconditioner, the approach in this lecture is significantly more scalable, and it is particularly useful for large-scale optimization problems.

In fact, the algorithms we will discuss today currently include the most-used first-order optimization algorithm in machine learning, ADAM.

1 The AdaGrad algorithm

The AdaGrad algorithm—introduced by Duchi, J., Hazan, E., & Singer, Y. [DHS11]—is a gradient-based optimization algorithm that adapts the learning rate for each variable based on the historical gradients.[1]

The main idea behind AdaGrad is to scale the learning rate of each variable *based on the sum of the squared gradients accumulated over time*. This allows AdaGrad to give smaller learning rates to frequently updated variables and larger learning rates to variables with infrequent updates. Going back to the example of the bridge design, this means that if we were to change the units of the length variables, AdaGrad would automatically adjust the learning rates to compensate for the change in scale.

In particular, at each iteration t, AdaGrad keeps a tally of the sum of the squared gradients up to time t for each variable. This is done by maintaining a vector s_t of components

$$[s_t]_i := \sqrt{\sum_{\tau=0}^{t} [\nabla f(x_\tau)]_i^2},$$

where $[\nabla f(x_t)]_i$ is the i-th component of the gradient at time t. The update rule for AdaGrad is then

$$\boxed{x_{t+1} = x_t - \eta M_t^{-1} \nabla f(x_t), \quad \text{where } M_t := \operatorname{diag}\left([s_t]_i := \sqrt{\sum_{\tau=0}^{t} [\nabla f(x_\tau)]_i^2} : i = 1, ..., n \right)}. \tag{AdaGrad}$$

We assume that $[\nabla f(x_0)]_i \neq 0$ for all i, so that M_t is invertible at all times $t = 0, 1, 2, ...$.

Remark 1.1. The same algorithm can be used in the **stochastic** setting, where as usual the gradient is replaced by a stochastic gradient,

$$x_{t+1} = x_t - \eta M_t^{-1} \tilde{\nabla} f(x_t), \quad \text{where } M_t := \operatorname{diag}\left([s_t]_i := \sqrt{\sum_{\tau=0}^{t} \left[\tilde{\nabla} f(x_\tau)\right]_i^2} : i = 1, ..., n \right)$$

It can also be used in the projected setting, where the update is projected onto a feasible set. For simplicity, in this lecture, we will focus on the deterministic, unconstrained setting for our analysis.

[1]Today we will consider the version of AdaGrad that only uses diagonal preconditioners, which is the version that is typically preferred in practice for large problems, including in machine learning applications (we will still refer to the algorithm with the generic name "AdaGrad").

2 ADAM: AdaGrad with momentum

In practice, people often use a variant of AdaGrad called ADAM, introduced by Kingma, D. P., & Ba, J. [KB15]. ADAM combines the adaptive learning rate of AdaGrad with the idea of momentum we already saw in Lecture 8. In particular, at each iteration t, ADAM keeps track of the momentum (discounted sum of past gradients)

$$g_t = \gamma g_{t-1} + (1-\gamma)\nabla f(x_t); \qquad g_{-1} \coloneqq 0.$$

The scaling factors s_t are also accumulated with a discount rate β as

$$[s_t]_i^2 = \beta [s_{t-1}]_i^2 + (1-\beta)[\nabla f(x_t)]_i^2 \qquad i = 1, \dots, n; \qquad s_{-1} \coloneqq 0.$$

Finally, ADAM updates the iterate as follows:

$$\boxed{x_{t+1} = x_t - \eta\, M_t^{-1}\, g_t, \quad \text{where } M_t \coloneqq \operatorname{diag}(s_t)}\,. \tag{ADAM}$$

The hyperparameters η, γ, and β in ADAM are *typically* set to 0.001, 0.9, and 0.999 respectively (this is `pytorch`'s default behavior).

Remark 2.1. The ADAM algorithm is widely used in practice and is known to work well for a wide range of optimization problems. It is particularly useful for training deep neural networks. However, ADAM does not have theoretical guarantees like AdaGrad. It is even known to diverge in some cases, even with convex objectives [RKK18].

3 Analysis of AdaGrad

In this section, we will analyze the AdaGrad algorithm. For simplicity, we will focus on the non-stochastic version, though the analysis in the presence of stochastic gradients is analogous.

The main result we will prove is that AdaGrad is competitive with the best possible preconditioner in hindsight, as we make precise in the next theorem.

Theorem 3.1. Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex and differentiable function. AdaGrad is competitive with the best preconditioner in hindsight. More precisely, for any choice of coefficients $\lambda_i \geq 0, i = 1, \dots, n$, the AdaGrad algorithm with stepsize $\eta = D/\sqrt{2}$ satisfies

$$\frac{1}{T}\sum_{t=0}^{T-1} f(x_t) \leq f(x_\star) + \frac{\sqrt{2n}D}{T}\sqrt{\min_{\lambda \in \mathbb{R}^n_{\geq 0}, \|\lambda\|_1 = n} \sum_{t=0}^{T-1} \nabla f(x_t)^\top \operatorname{diag}(\lambda)^{-1} \nabla f(x_t)},$$

where $D \coloneqq \max_{t=0}^{T} \|x_t - x_\star\|_\infty$ is the maximum distance from the optimal solution at all times T.

In the rest of this section, we prove the above result.

3.1 AdaGrad as an instance of mirror descent

The main idea behind the proof is to show that AdaGrad is a form of mirror descent algorithm (Lecture 9), with the twist that the distance-generating function is *time-dependent*. In particular, we will use as our distance-generating function the (strongly convex) function

$$\varphi_t(x) \coloneqq \frac{1}{2}x^\top M_t x.$$

The induced Bregman divergence is

$$\mathrm{D}_{\varphi_t}(x \,\|\, y) = \varphi_t(x) - \varphi_t(y) - \langle \nabla\varphi_t(y), x - y\rangle = \frac{1}{2}(x-y)^\top M_t (x-y).$$

We make the connection formal in the following lemma.

Theorem 3.2. The AdaGrad update rule is equivalent to the mirror descent update rule with the distance-generating function $\varphi_t(x) = \frac{1}{2}x^\top M_t x$, where $M_t \coloneqq \mathrm{diag}(s_t)$.

Proof. Remember that the mirror descent update rule is

$$\begin{aligned} x_{t+1} &= \arg\min_{x\in\mathbb{R}^n} \Big\{ \eta\langle \nabla f(x_t), x\rangle + \mathrm{D}_{\varphi_t}(x \,\|\, x_t) \Big\} \\ &= \arg\min_{x\in\mathbb{R}^n} \left\{ \eta\langle \nabla f(x_t), x\rangle + \frac{1}{2}(x - x_t)^\top M_t (x - x_t) \right\}. \end{aligned}$$

Setting the gradient of the above objective to zero and solving for x yields

$$\eta \nabla f(x_t) + M_t(x - x_t) = 0 \qquad \Longrightarrow \qquad x = x_t - \eta M_t^{-1} \nabla f(x_t),$$

which is exactly the AdaGrad update rule. □

From the mirror descent lemma we saw in Lecture 9, we can write

$$f(x_t) \le f(x_\star) + \frac{1}{\eta}\Big(\mathrm{D}_{\varphi_t}(x_\star \,\|\, x_t) - \mathrm{D}_{\varphi_t}(x_\star \,\|\, x_{t+1}) + \mathrm{D}_{\varphi_t}(x_t \,\|\, x_{t+1}) \Big).$$

Summing the above inequalities over $t = 0, 1, ..., T-1$ and using the fact that $M_0 = 0$ yields

$$\sum_{t=0}^{T-1} f(x_t) \le T f(x_\star) + \frac{1}{\eta}\left(\underbrace{\sum_{t=0}^{T-2}\Big(\mathrm{D}_{\varphi_{t+1}}(x_\star \,\|\, x_{t+1}) - \mathrm{D}_{\varphi_t}(x_\star \,\|\, x_{t+1}) \Big)}_{\text{Ⓐ}} + \underbrace{\sum_{t=0}^{T-1} \mathrm{D}_{\varphi_t}(x_t \,\|\, x_{t+1})}_{\text{Ⓑ}} \right).$$

We will now proceed, in the next two subsections, to bound the two summations Ⓐ and Ⓑ separately.

3.2 Bounding the "almost-telescopic" terms Ⓐ

Theorem 3.3. Let T be arbitrary and assume that $D \coloneqq \max_{t=0}^{T} \|x_t - x_\star\|_\infty$ is finite. Then, at all times T, the sum Ⓐ satisfies the inequality

$$\sum_{t=0}^{T-2}\Big(\mathrm{D}_{\varphi_{t+1}}(x_\star \,\|\, x_{t+1}) - \mathrm{D}_{\varphi_t}(x_\star \,\|\, x_{t+1}) \Big) \le \frac{D^2}{2}\|s_{T-1}\|_1.$$

Proof. It is easy to see that

$$\mathrm{D}_{\varphi_{t+1}}(x_\star \,\|\, x_{t+1}) - \mathrm{D}_{\varphi_t}(x_\star \,\|\, x_{t+1}) = \frac{1}{2}(x_{t+1} - x_\star)^\top (M_{t+1} - M_t)(x_{t+1} - x_\star).$$

(The above is a generalization of the result we saw in Lecture 9, which established that the Bregman divergence induced by the squared Euclidean norm is the squared Euclidean distance.)

Using now the definition of D together with the Cauchy-Schwarz inequality, we can write

$$\begin{aligned}\frac{1}{2}(x_{t+1}-x_\star)^\top(M_{t+1}-M_t)(x_{t+1}-x_\star) &\le \frac{1}{2}\|x_{t+1}-x_\star\|_\infty^2\|s_{t+1}-s_t\|_1\\ &\le \frac{D^2}{2}\|s_{t+1}-s_t\|_1.\end{aligned}$$

On the other hand, since $s_{t+1} \ge s_t \ge 0$ componentwise at all times t, then $\|s_{t+1}-s_t\|_1 = \|s_{t+1}\|_1 - \|s_t\|_1$ and we can write

$$\begin{aligned}\sum_{t=0}^{T-2}\Big(\mathrm{D}_{\varphi_{t+1}}(x_\star \,\|\, x_{t+1}) - \mathrm{D}_{\varphi_t}(x_\star \,\|\, x_{t+1})\Big) &\le \frac{D^2}{2}\sum_{t=0}^{T-2}\Big(\|s_{t+1}\|_1 - \|s_t\|_1\Big)\\ &= \frac{D^2}{2}\big(\|s_{T-1}\|_1 - \|s_0\|_1\big)\\ &\le \frac{D^2}{2}\|s_{T-1}\|_1,\end{aligned}$$

which is the statement. □

3.3 Bounding the summation Ⓑ

We now shift our attention to the summation in Ⓑ, where we will establish the following bound.

Theorem 3.4. At all times T, the sum Ⓑ satisfies the inequality

$$\sum_{t=0}^{T-1}\mathrm{D}_{\varphi_t}(x_t \,\|\, x_{t+1}) \le \eta^2\|s_{T-1}\|_1.$$

Proof. Expanding the expression for the Bregman divergence $\mathrm{D}_{\varphi_t}(x \,\|\, y) = \frac{1}{2}(x-y)^\top M_t(x-y)$, we have

$$\mathrm{D}_{\varphi_t}(x_t \,\|\, x_{t+1}) = \frac{1}{2}(x_{t+1}-x_t)^\top M_t(x_{t+1}-x_t) = \frac{\eta^2}{2}\nabla f(x_t)^\top M_t^{-1}\,\nabla f(x_t).$$

Using the definition of $M_t \coloneqq \mathrm{diag}(s_t)$ where $[s_t]_i \coloneqq \sqrt{\sum_{\tau=0}^{t}[\nabla f(x_\tau)]_i^2}$, we can then write

$$\begin{aligned}\nabla f(x_t)^\top M_t^{-1}\,\nabla f(x_t) &= \sum_{i=1}^{n}\frac{[\nabla f(x_t)]_i^2}{[s_t]_i}\\ &= \sum_{i=1}^{n}\frac{[s_t]_i^2 - [s_{t-1}]_i^2}{[s_t]_i}\\ &\le 2\sum_{i=1}^{n}\frac{[s_t]_i^2 - [s_{t-1}]_i^2}{[s_t]_i + [s_{t-1}]_i} = 2\sum_{i=1}^{n}\Big([s_t]_i - [s_{t-1}]_i\Big) = 2\Big(\|s_t\|_1 - \|s_{t-1}\|_1\Big).\end{aligned}$$

Summing over $t = 0, 1, ..., T-1$ yields the result. □

3.4 Finale: Bounding the norm of the scaling factors

The two bounds above show that

$$\frac{1}{T}\sum_{t=0}^{T-1} f(x_t) \le f(x_\star) + \frac{1}{T}\left(\frac{D^2}{2\eta} + \eta\right)\|s_{T-1}\|_1.$$

Hence, setting $\eta = D/\sqrt{2}$ yields

$$\frac{1}{T}\sum_{t=0}^{T-1} f(x_t) \le f(x_\star) + \frac{D\sqrt{2}}{T}\|s_{T-1}\|_1. \tag{3}$$

So, to complete the proof, we only need to provide a bound on the norm of the scaling factors s_{T-1}. This is the content of the following theorem.

Theorem 3.5. The norm of the scaling factors s_{T-1} satisfies the inequality

$$\|s_{T-1}\|_1 \le \sqrt{n} \cdot \sqrt{\min_{\lambda \in \mathbb{R}^n_{\ge 0}, \|\lambda\|_1 = n} \sum_{t=0}^{T-1} \nabla f(x_t)^\top \operatorname{diag}(\lambda)^{-1} \nabla f(x_t)}.$$

Proof. Pick any $\lambda \in \mathbb{R}^n_{\ge 0}$, $\|\lambda\|_1 = n$; we will use Cauchy-Scharz to bound $\|s_{T-1}\|_1^2$ as follows:

$$\begin{aligned}
\|s_{T-1}\|_1^2 &= \left(\sum_{i=1}^{n} \sqrt{\sum_{t=0}^{T-1} [\nabla f(x_t)]_i^2}\right)^2 \\
&= \left(\sum_{i=1}^{n}\left[\sqrt{\lambda_j} \cdot \left(\frac{1}{\sqrt{\lambda_j}}\sqrt{\sum_{t=0}^{T-1} [\nabla f(x_t)]_i^2}\right)\right]\right)^2 \\
&\le \left(\sum_{i=1}^{n} \lambda_i\right) \cdot \left(\sum_{i=1}^{n}\left(\frac{1}{\lambda_i}\sum_{t=0}^{T-1} [\nabla f(x_t)]_i^2\right)\right) && \text{(from the Cauchy-Schwarz inequality)} \\
&= n \cdot \left(\sum_{t=0}^{T-1} \nabla f(x_t)^\top \operatorname{diag}(\lambda)^{-1} \nabla f(x_t)\right).
\end{aligned}$$

Taking square roots and using the fact that λ was arbitrary yields the statement. □

Plugging the bound of Theorem 3.5 into (3) proves Theorem 3.1.

Bibliography

[DHS11] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization.," *Journal of machine learning research*, vol. 12, no. 7, 2011.

[KB15] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *International Conference on Learning Representations (ICLR)*, 2015. [Online]. Available: `http://arxiv.org/abs/1412.6980`

[RKK18] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of Adam and beyond," in *International Conference on Learning Representations (ICLR)*, 2018.