PSGD Pytorch implementation updates

Xilin Li

December 21, 2023

Notations:

- H: Hessian
- v: a probe vector for Hessian estimation, typically $v \sim \mathcal{N}(0, I)$
- h: Hessian-vector product, i.e., $h = Hv + \epsilon$, where $\epsilon = 0$ if no gradient noise
- P: preconditioner
- Q: factor of P as $P = Q^T Q$

Preconditioner fitting loss given a single pair of (v, h),

$$\ell(v, h; P) = h^T P h + v^T P^{-1} v$$

Expectation of the above loss assuming $\epsilon = 0$,

$$\ell(P) = \text{tr}(PH^2) + \text{tr}(P^{-1})$$

The optimal P is $|H|^{-1}$. With $\epsilon \neq 0$, the stochastic gradient noise damps the preconditioner estimation.

1 Dec 2023

1.1 Added the gradient whitening preconditioner

When h is set to the stochastic gradient, we have the optimal P as

$$P = \left(E[gg^T]\right)^{-1/2}$$

This preconditioner whitens the gradient. Specifically, for this type, PSGD reduces to RMSProp and Adam (with momentum) when P is diagonal; PSGD of this type resembles Shampoo with affine group preconditioners. By default, we assume the Newton type, i.e., $|H|^{-1}$ as the preconditioner.

1.2 Wrapped affine preconditioner as a class, also support complex matrix

Affine preconditioner design can have many choices. Here, I have implemented a class with preconditioner form $Q = \oplus_i (Q_{2,i}^* \otimes Q_{1,i})$, where $(\cdot)^*$ takes conjugate, and all the $Q_{1\,\text{or}\,2,i}$ are real or complex, triangular or diagonal, matrices. It's possible to replace $Q_{1\,\text{or}\,2,i}$ with many other forms like normalization, LRA, etc. This preconditioner also has two types: Newton or gradient whitening. Its "gradient whitening" type resembles Shampoo, but avoids any matrix inversion by fitting preconditioner on the Lie group of triangular or diagonal matrices.

Let's check the complex matrix case to review the math. Let the ith affine transform be

$$y = \Theta x$$

where Θ can be a real or complex matrix, and x includes feature 1 if bias is used. Let's ignore the 2nd derivative term, $\frac{\partial}{\partial \text{vec}(\Theta)^T} \frac{\partial}{\partial \text{vec}(\Theta^*)}$. Then, the gradient of Θ, say G, can be preconditioned as

$$P\,\text{vec}(G) = (Q_2^T Q_2^* \otimes Q_1^H Q_1)\text{vec}(G) = \text{vec}(Q_1^H Q_1 G Q_2^H Q_2)$$

where $\text{vec}(G^*)$ is ignored, and P can be rewritten as

$$P = Q_2^T Q_2^* \otimes Q_1^H Q_1 = (Q_2^* \otimes Q_1)^H (Q_2^* \otimes Q_1) = Q^H Q$$

Thus, the preconditioned gradient descent formula for Θ with a positive step size μ is

$$\Theta \leftarrow \Theta - \mu Q_1^H Q_1 G Q_2^H Q_2,$$

which is the gradient descent for Θ' in $y' = \Theta' x'$ with $Q' = Q_1^{-H} \Theta Q_2^{-1}$, $x' = Q_2 x$ and $y' = Q_1^{-H} y$. It's always feasible to let the last row of Q_2 be $[0, \ldots, 0, 1]$ when the last element of both x and x' are 1. Thus, Q_2 forms an affine group. Also, Q_1 forms another affine group that keeps 0 as a fixed point. Thus the name affine preconditioner. (see https://arxiv.org/pdf/1809.10232.pdf Section 5).

Let's consider a small perturbation of Θ and its associated gradient change, i.e. pair $(\delta\Theta, \delta G)$. As we ignore the dependence of δG on $\delta\Theta^*$, the preconditioner fitting loss given pair ($\delta\Theta$, δG) is

$$\begin{aligned}
\ell(\delta\Theta, \delta G; Q_1, Q_2) =& \|Q \text{vec}(\delta G)\|^2 + \|Q^{-H} \text{vec}(\delta\Theta)\|^2 \\
=& \|(Q_2^* \otimes Q_1)\text{vec}(\delta G)\|^2 + \|(Q_2^{-T} \otimes Q_1^{-H})\text{vec}(\delta\Theta)\|^2 \\
=& \|\text{vec}(Q_1\, \delta G\, Q_2^H)\|^2 + \|\text{vec}(Q_1^{-H}\, \delta\Theta\, Q_2^{-1})\|^2 \\
=& \text{trace}(Q_1\, \delta G\, Q_2^H Q_2\, \delta G^H\, Q_1^H) + \text{trace}(Q_1^{-H}\, \delta\Theta\, Q_2^{-1} Q_2^{-H}\, \delta\Theta^H\, Q_1^{-1})
\end{aligned}$$

On Lie group $\text{GL}(n, \mathcal{C})$, we let

$$dQ_1 = \mathcal{E}_1 Q_1,\ dQ_2 = \mathcal{E}_2 Q_2$$

and introduce terms A and B as

$$\begin{aligned}
A =& Q_1\, \delta G\, Q_2^H \\
B =& Q_2^{-H}\, \delta\Theta^H\, Q_1^{-1}
\end{aligned}$$

Then, straightforward algebra calculations show that

$$\begin{aligned}
d\ell =& d(\text{trace}(AA^H) + \text{trace}(B^H B)) \\
=& \text{trace}(dA\, A^H + A\, dA^H + dB^H\, B + B^H dB) \\
=& \text{trace}(\mathcal{E}_1 AA^H + A\mathcal{E}_2^H A^H + AA^H \mathcal{E}_1^H + A\mathcal{E}_2 A^H \\
& - \mathcal{E}_1^H B^H B - B^H \mathcal{E}_2 B - B^H B \mathcal{E}_1 - B^H \mathcal{E}_2^H B)
\end{aligned}$$

Thus,

$$\begin{aligned}
\frac{\partial \ell}{\partial \mathcal{E}_1^*} =& AA^H - B^H B \\
\frac{\partial \ell}{\partial \mathcal{E}_2^*} =& A^H A - BB^H
\end{aligned}$$

When we are to fit the preconditioner on the group of triangular or diagonal matrix, we can simply take the triangular or diagonal parts of the above gradients. Feature normalization has a somewhat different form (https://arxiv.org/pdf/1809.10232.pdf Section 5.3). For real valued gradients, the above equations match the ones given in https://arxiv.org/abs/1512.04202 Section V.B. On the group of triangular matrices, $B = Q_2^{-H}\, \delta\Theta^H\, Q_1^{-1}$ can be calculated with backward substitution. Thus, no matrix inverse, which can be nasty, is involved in our preconditioner updating process.

1.3 Tighter lower bound for triangular matrix norm

We update Q on the group of triangular matrix as

$$Q \leftarrow (I + \mu A)Q$$

where the step size μ is small enough such that $\|\mu A\|_2 < 1$, and $-A \in \mathbb{R}^{N \times N}$ is the gradient for preconditioner fitting.

Possible cheap, i.e., $\mathcal{O}(N^2)$ complexity, lower bounds of $\|A\|_2$ for step size normalization are:

- Bound F: $\frac{1}{\sqrt{r}}\|A\|_F \le \|A\|_2 \le \|A\|_F$, where r is the rank of G.
- Bound max: $\|A\|_{\max} \le \|A\|_2 \le N\|A\|_{\max}$, where $\|A\|_{\max} = \max_{ij} a_{ij}$
- Bound 1: $\frac{1}{\sqrt{N}}\|A\|_1 \le \|A\|_2 \le \sqrt{N}\|A\|_1$, where $\|A\|_1 = \max_j \sum_i |a_{ij}|$
- Bound inf: $\frac{1}{\sqrt{N}}\|A\|_\infty \le \|A\|_2 \le \sqrt{N}\|A\|_\infty$, where $\|A\|_\infty = \max_i \sum_j |a_{ij}|$

Bound F is useful only when $r \ll N$. In general, none of the above lower bound is tighter than the rest. In the updated implementation, I replaced lower bound $\|A\|_{\max}$ with the following consistently tighter one.

Let

$$\beta = \sqrt{\max\left(\max_i \sum_j a_{ij}^2,\ \max_j \sum_i a_{ij}^2\right)}$$

Then we have

$$\beta \le \|A\|_2 \le \sqrt{N}\beta$$

Not difficult to prove the following desired properties:

- $\beta \le \|A\|_2$: since $\|A^T\|_2 = \|A\|_2 \ge \|Ax\|/\|x\|$ for any $x \ne 0$, we let x be the columns or rows of A to have this lower bound. This lower bound is tight, e.g., when $A = I$.
- $\beta \ge \frac{1}{\sqrt{N}}\|A\|_F$: obvious.
- $\beta \ge \|A\|_{\max}$: obvious.
- $\beta \ge \frac{1}{\sqrt{N}}\|A\|_1$: due to the Cauchy–Schwarz inequality, $N\sum_i a_{ij}^2 \ge (\sum_i |a_{ij}|)^2$.
- $\beta \ge \frac{1}{\sqrt{N}}\|A\|_\infty$: again, due to the Cauchy–Schwarz inequality.
- $\|A\|_2 \le \sqrt{N}\beta$: as $\|A\|_2 \le \|A\|_F \le \sqrt{N}\beta$. This upper bound is tight, e.g., when A is a matrix with all elements taking the same value.

1.4 Initial scale of the preconditioner can be set to None

I initialize Q as $Q = \alpha I$, where α is the initial scale. By letting

$$\alpha^2 \text{ learning\_rate}_{\text{PSGD}} = \text{learning\_rate}_{\text{SGD}}$$

we can map the initial settings of SGD to PSGD. In general, proper tuning of α is necessary.

If α is set to None, the first pair of (v, h) is used to initialize Q as

$$Q = \left(\frac{v^T v}{h^T h}\right)^{1/4} I$$

However, this can be risky as the Newton's method does not always converge faster than gradient descent out of the basin of attraction. For example, considering convex function $f(x) = x^2 - 4x^{1/2}, x \in \mathbb{R}^+$. We have $f'(x) = 2x - 2x^{-1/2}$, $f''(x) = 2 + x^{-3/2}$, and optimal soltion $x = 1$. The Newton's method can be arbitrarily slower than the gradient descent when $0 < x \ll 1$, as shown by

$$\lim_{x\to 0} \frac{f'(x)}{f''(x)} = \lim_{x\to 0} \frac{2x - 2x^{-1/2}}{2 + x^{-3/2}} = \lim_{x\to 0} \frac{-2x^{-1/2}}{x^{-3/2}} = -2 \lim_{x\to 0} x = 0$$

Thus, for this example, PSGD will get stuck around $x = 0$ if we rely on this strategy to determine α.

1.5 More updates

- Changed class name UVd to LRA (low-rank approximation); keep UVd as an alias of LRA.
- Now LRA reduces to the diagonal preconditioner when the rank of approximation is set to 0.
- Updated hello_psgd.py to show how easy to apply PSGD on minimizing the Rosenbrock function.
- Added misc/preconditioner_fitting_rule_verification.py to show how fitting works on Lie groups.