Efficient Full-Matrix Adaptive Regularization

Naman Agarwal, Brian Bullins, Xinyi Chen, Elad Hazan, Karan Singh, Cyril Zhang, Yi Zhang

Princeton University
Google AI Princeton
Adaptive Preconditioning in ML

- Optimization in ML: training neural nets → minimizing non-convex losses

- Diagonal Adaptive Optimizers: each coordinate has a different learning rate according to past gradients
  - AdaGrad, Adam, RMSProp
  - Works well in practice

Theory is only known for convex losses at the time
Adaptive Preconditioning: Intuition

Doesn’t adapt to a rotated basis

\[ x_{t+1} \leftarrow x_t - \text{diag} \left[ \sum_{s=1}^{t} g_s g_s^\top \right]^{-1/2} \cdot g_t \]

Learns the correct basis, faster optimization

\[ x_{t+1} \leftarrow x_t - \left[ \sum_{s=1}^{t} g_s g_s^\top \right]^{-1/2} \cdot g_t \]

Expensive!

Can we have a linear time algorithm?
Our Results

- **GGT**: a new adaptive optimizer
  Efficient full-matrix (low-rank) AdaGrad

- **Experiments**: faster training and sometimes better generalization on vision and language tasks
- **GPU-friendly Implementation**

- **Theory**: “adaptive” convergence rate on convex and non-convex functions
- **Up to $O(1/\sqrt{d})$ faster than SGD**
The GGT Trick

- Scalar Case:

\[(a \times a)^{-1/2} = a \times (a \times a)^{-3/2} \times a\]

- Matrix Case:
The GGT Trick

- Scalar Case:

\[(a \times a)^{-1/2} = a \times (a \times a)^{-3/2} \times a\]

- Matrix Case:
The GGT Trick

- Scalar Case:

\[(a \times a)^{-1/2} = a \times (a \times a)^{-3/2} \times a\]

- Matrix Case:

\[
\begin{bmatrix}
G_t \\
\end{bmatrix}
\begin{bmatrix}
G_t^T \\
\end{bmatrix}
\begin{bmatrix}
\frac{1}{2} \\
\end{bmatrix}
\begin{bmatrix}
\frac{3}{2} \\
\end{bmatrix}
\begin{bmatrix}
G_t^T G_t \\
\end{bmatrix}
\]

Efficient implementation on the GPU!
Large-Scale Experiments (CIFAR-10, PTB)

- Resnet-26 for CIFAR-10 and LSTM for PTB
- Better and faster training
- Initial acceleration in optimizing the LSTM
- Better validation ppl for the LSTM
Theory

- Define the *adaptivity ratio*:

\[ \mu^2 = \frac{\text{AdaGrad Regret}}{\text{worst-case OGD Regret}} \]

[DHS10]: \( \mu^2 \in \left[ \frac{1}{\sqrt{d}}, \sqrt{d} \right] \) for diagonal AdaGrad, sometimes smaller for full-matrix AdaGrad

- **Non-Convex reduction**: GGT* converges in \( \tilde{O}\left(\frac{\mu^2 \sigma^2}{\epsilon^4}\right) \) steps

- First step towards analyzing adaptive methods in non-convex optimization

* Idealized modification of GGT for analysis. See paper for details.
A note on the important parameters

- Improving dependence on epsilon: \( \frac{1}{\epsilon^4} \rightarrow \frac{1}{\epsilon^{3.5}} \)

  In practice \( \epsilon \sim 0.1 \), leading to an improvement of about 3.1

- Instead our improvement can be as large as the dimension, which can be 1e7 for language models

- Huge untapped potential for large-scale optimization!
Thank You!

Poster #209

xinyic@google.com