Descriptions of Projects for Summer 2020 CSST

HanQin Cai, Yuchen Lou and Daniel McKenzie

2020

Abstract

Let's outline the scope and goals of the various projects we are pursuing this summer. At the moment, I'm focusing on the projects that have not yet been clearly defined. So, I won't go into too much detail on Tree and Block Sparse ZORO.

1 Structured Sparsity Extensions to ZORO

- As discussed, we want to extend ZORO to be able to deal with gradients that exhibit structured sparsity. The two cases we have examined are tree-sparse and block-sparse.

- We note that in both cases, the structure has to be known *a priori*. Clearly, this is unrealistic in some applications. Thus, it is of interest to see whether the tree structure can be *learned*. Specifically, this would mean the following:
  - For the first T iterations, use regular ZORO. Compute the gradient estimators $\hat{\mathbf{g}}_k$ and let $S_k = \text{supp}(\hat{\mathbf{g}}_k)$.
  - Assume that there exists an underlying tree $\mathcal{T}$, such that all gradients for this problem are $\mathcal{T}$-sparse. For now lets assume that all gradients should be s-tree-sparse (*i.e.* have s non-zero entries that form a subtree of $\mathcal{T}$).
  - Observe that each S_k is the set of vertices of an s-vertex subtree of $\mathcal{T}$. Interestingly, observe that S_k alone does not tell us any parent/child relationships.
  - Let's write $\mathcal{T} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ represents the vertices and $\mathcal{E}$ represents the edges. Lets assume that T is large enough that $\cup_{k=1}^{T} S_k = \mathcal{V}$.
  - Here is the interesting question: Can we infer $\mathcal{E}$ from $\{S_1, \ldots, S_T\}$? This seems hard, but one could exploit the tree structure. For example if S_{k_1} and S_{k_2} differ by one index, call it i, then we can infer that i cannot be a parent to any of the vertices in S_{k_1} or S_{k_2}.

Suppose that this problem can be solved, and now let T be the minimum number such that the problem is solvable with $S_1, \ldots, S_T$. Then one can make an adaptive algorithm that runs regular ZORO for T iterations, then reconstructs $\mathcal{T}$, and then switches to tree-sparse ZORO for the remainder of the optimization process.

- Possible applications. We (i.e. HanQin and Daniel) have some expertise in adversarial attacks on classification algorithms. This is quite a trendy application of zeroth-order optimization, so I think it could be a nice way to apply structure-aware ZORO. Three possible ways to do this (and don't worry if this doesn't all make sense to you at this stage):
  - Attacks on decision trees. This paper: [CLC+18] details a way to do this. They use a generic zeroth-order algorithm to do the attack, but it can probably be sped up by using a tree-sparsity-aware zeroth-order algorithm. The downside is that one would need to know the tree, which would mean it's only partially black-box.
  - Attacks that are sparse in the wavelet domain on Convolutional Neural Networks for image classification.
  - Block-sparse attacks on Convolutional Neural Networks for image classification.This paper: [XLZ+18] has some details on this, but I haven't read it closely.

2 Exploiting Block Sparsity for High Dimensional ZORO

- Suppose that we are solving $\text{minimize}_{x \in \mathbb{R}^d} f(x)$ where d is extremely large.
- ZORO generates $z_i \in \mathbb{R}^d$ with $z_{i,j} = \pm 1$ and then uses finite differences to get $y_i \approx z_i^\top \mathbf{g}_k$.
- As we've discussed, the z_i are then stacked into a matrix $Z \in \mathbb{R}^{m \times d}$. By construction, this matrix is now full (*i.e.* not sparse. When d is large this can require too much memory, and operations with this matrix might be too slow.
- So, my idea is the following:
  - Divide $\{1, \ldots, d\}$ into D blocks of size d/D. Lets label the blocks $\mathbf{b}_1, \ldots, \mathbf{b}_D$. Note that $\mathbf{b}_a = \{a * (d/D) + 1, \ldots, (a+1) * (d/D)\}$.
  - Now, only generate z_i that are *supported on only one block*. That is, $\text{supp}(z_i) = \mathbf{b}_{a_i}$ for some a_i. There are several ways one could do it. Given i one could choose the a_i uniformly at random. Alternatively, at the k-th iteration of the gradient descent part of ZORO, one could fix a block $\mathbf{b}_{a_k}$, and then choose all the z_i at this iteration to have

support $\mathbf{b}_{a_k}$. This turns ZORO into a sort of block coordinate descent algorithm, which could be very interesting to analyze.

– That's where this paper: [**?**] comes in. Hopefully we can use their analysis to establish when the matrix Z has the RIP, and hence use the analysis in the ZORO paper to bound $\|\mathbf{g}_k - \hat{\mathbf{g}}_k\|$.

3 Variance Reduction for ZORO

Yuchen's note sent on Friday 12th June summarizes this part quite nicely. The only thing I would add is that I think we can further exploit the structure of the error terms, $e_i = z_i^\top \nabla^2 f(x) z_i$. In particular, one can use the Hanson-Wright inequality to get bounds of the form:

$$\mathbb{P}\left[\left|z_i^\top \nabla^2 f(x) z_i - \mathbb{E}\left[z_i^\top \nabla^2 f(x) z_i\right]\right| \geq t\right] \leq e^{-ct}$$

Of course one would then also need to bound the difference between the true mean, $\mathbb{E}\left[z_i^\top \nabla^2 f(x) z_i\right]$, and the sample mean:

$$\bar{e} = \frac{1}{m}\sum_i e_i = \frac{1}{m}\sum_i z_i^\top \nabla^2 f(x) z_i$$

so getting sharper bounds than those given by Popoviciu's inequality might not be so straightforward.

References

[CLC+18] Minhao Cheng, Thong Le, Pin-Yu Chen, Jinfeng Yi, Huan Zhang, and Cho-Jui Hsieh. Query-efficient hard-label black-box attack: An optimization-based approach. *arXiv preprint arXiv:1807.04457*, 2018.

[XLZ+18] Kaidi Xu, Sijia Liu, Pu Zhao, Pin-Yu Chen, Huan Zhang, Quanfu Fan, Deniz Erdogmus, Yanzhi Wang, and Xue Lin. Structured adversarial attack: Towards general implementation and better interpretability. *arXiv preprint arXiv:1808.01664*, 2018.