Practical problem for big n on average of finite sum

The limitation of machine precision on the problem $\min_{\mathbf{x}} \frac{1}{n} \sum_i f_i(\mathbf{x})$

Andersen Ang

Mathématique et recherche opérationnelle
UMONS, Belgium

manshun.ang@umons.ac.be Homepage: angms.science

First draft : June 21, 2019
Last update : June 21, 2019

The minimization problem : average of finite sum

$$(\mathcal{P}) \quad : \quad \min_{\mathbf{x}\in\mathcal{Q}} F(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x}).$$

where

- $\mathbf{x}$ is the optimization variable
- n is the number of sum ($n \neq \infty$)
- f_i is the ith component of F

The key point of the algorithms that specifically designed to solve $(\mathcal{P})$ is that n is "big", so computing the full gradient

$$\nabla F(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\mathbf{x})$$

is prohibitive : too big to compute, too big to be stored, too be to be transported. Hence, algorithms that make use of partial gradient is used.

The problem of machine precision for big n

In theory, as long as $n < +\infty$, everything works.
However, in practise, when n is too big, problems appear.
Suppose the computing environment you are using provide 7 digits of precision. That is, things after the 7th digit become untrustworthy.

Now suppose the problem size is $n = 10^9$, hence

$$F(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x}) = \frac{1}{10^9}\sum_{i=1}^{n} f_i(\mathbf{x}).$$

In this case, if $\sum_{i=1}^{n} f_i(\mathbf{x})$ is in the order of 10^0, then even stochastic methods may not be able to correctly handle such big data since

$$F(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x}) = \frac{1}{10^9}[c \times 10^0] = \frac{c}{10^9},$$

which is outside the machine precision and the c is meaningless.

In other words, here the result is not trustworthy since the $\frac{1}{n}$ part is so big, making the $\sum_{i=1}^{n} f_i(\mathbf{x})$ part meaningless.

Big n causes problem when comparing algorithm

Suppose we compare two algorithms $\mathcal{A}_1$, $\mathcal{A}_2$ on the minimization problem

$$\min_{\mathbf{x}\in\mathcal{Q}} F(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x}).$$

Let $\mathbf{x}_{\mathcal{A}_1}$ and $\mathbf{x}_{\mathcal{A}_2}$ be the outputs of the algorithms respectively. We have

$$F(\mathbf{x}_{\mathcal{A}_1}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x}_{\mathcal{A}_1}), \quad F(\mathbf{x}_{\mathcal{A}_2}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x}_{\mathcal{A}_2}).$$

Suppose within the same amount of computational time, same amount of computational resources, and same number of iterations, we have

$$F(\mathbf{x}_{\mathcal{A}_1}) < F(\mathbf{x}_{\mathcal{A}_2}).$$

Can we say $\mathcal{A}_1$ is a better algorithm than $\mathcal{A}_2$ (in terms of convergence)?

Answers : it depends on n. We cannot say so if the problem of machine precision occur – if $\frac{1}{n}$ is so big that both $F(\mathbf{x}_{\mathcal{A}_1})$ and $F(\mathbf{x}_{\mathcal{A}_2})$ are outside machine precision, the numerical result is not trustworthy anymore.

Big n is common : the curse of dimensionality

The problem becomes much serious if the data is matrix or tensor.

For a vector problem, it has n elements in total, we may have

$$F(\mathbf{x}) = \frac{1}{n} \sum_i f_i(\mathbf{x})$$

For a matrix problem, it has $n_1 \times n_2$ elements in total, we may have

$$F(\mathbf{x}) = \frac{1}{n_1 n_2} \sum_{ij} f_{ij}(\mathbf{x})$$

For a tensor problem, it has $n_1 \times n_2 \times n_3$ elements in total, we may have

$$F(\mathbf{x}) = \frac{1}{n_1 n_2 n_3} \sum_{ijk}^{n} f_{ijk}(\mathbf{x})$$

These denominators can be big and cause the machine precision problem.

Practical solution to big n problem on machine precision

Solution 1. Increase the number of precision digits (p).
Simply use a better compute and better computing environment. Use `double` (16) or `quadruple` (34) instead of `single` (7).

Solution 2. Split the data into mini-batches.
To handle the problem

$$\min_{\mathbf{x}\in\mathcal{Q}} F(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x}),$$

split the data into G groups with size $n_1, \ldots, n_G$ with $n_1 + \cdots + n_G = n$, then solve for each

$$\min_{\mathbf{x}\in\mathcal{Q}} F_g(\mathbf{x}) = \frac{1}{n_g}\sum_{i=1}^{n^g} f_i(\mathbf{x}), \quad g = 1, 2, \ldots, G$$

and finally take the average among the G computed $F_g(\mathbf{x})$.

-END-