GlobalDataLoader in Multi DeepLearning Task

Xie Jian

I2EC, ICS, NJU

March 10, 2021

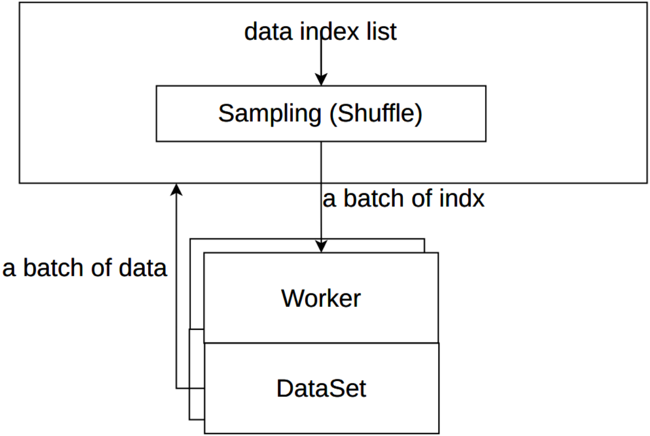
Table of Contents

1. Introduction
2. Global DataLoader
3. Experiment

Table of Contents

1. Introduction
2. Global DataLoader
3. Experiment

DataLoader in Pytorch



Problem: Repeated Reading and Processing

Situation

To compare the performance of different algorithms, Many DeepLearning tasks are training in the same Dataset.

Problem

Every task has its own DataLoader. So the data will be repeatedly read and processed by different tasks.

Result

As the number of tasks increases, so does the training time. And what increases is the time to load the data

Experiment

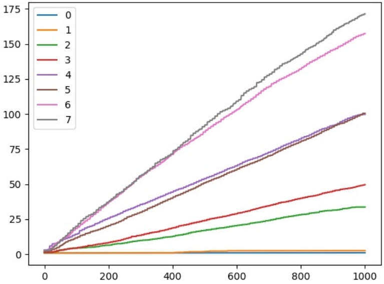


Figure: data loading time

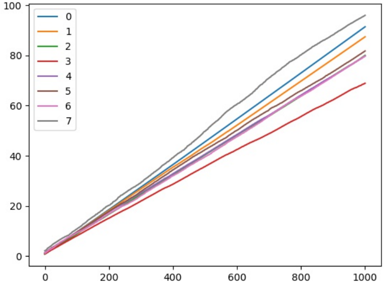
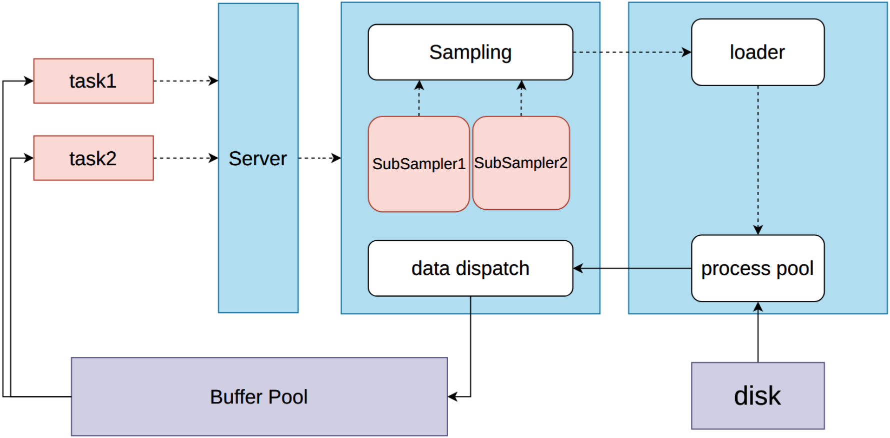


Figure: data training time

Table of Contents

1. Introduction
2. Global DataLoader
3. Experiment

Architecture



Sampling: problem description

Defination

For a single task, the sampler needs to select an element from the index set S.
Similarly, for multiple tasks, the sampler needs to select some elements $\{s_1, s_2, ...\}$ from multiple sets $\{S_1, S_2, ...\}$

Requirments

- The index in the set S should be randomly sampled. $p(s_i) = \frac{1}{|S_i|}$
- Duplicate indexes need to be merged. Maximize the probability $p(s_i = s_j)$
- There can be no problem of "starvation"

Independently Sampling Algorithm

Assumption

There are two sets: S_1, S_2, and their length is n_1, n_2
The intersection set of them is S_i, whose length is n_i
We divide the set S_1 into S_i and $S_{d1} = S_1 - S_i$
We divide the set S_2 into S_i and $S_{d2} = S_2 - S_i$

Example

$S_1 = \{1, 2, 3, 4, 5\} = \{1, 2, 3\} \cup \{4, 5\}$
$S_2 = \{1, 2, 3, 6, 7\} = \{1, 2, 3\} \cup \{6, 7\}$

Independently Sampling Algorithm

Algorithm

- S_1:
  - $step_{11}$: randomly select a set from S_i and S_{d1}
  - $step_{12}$: if the set is S_{d1}, randomly select a element1 from S_{d1}
  - $step_{13}$: if the set is S_i, randomly select a element1 from S_i
- S_2:
  - $step_{21}$: randomly select a set from S_i and S_{d2}
  - $step_{22}$: if the set is S_{d2}, randomly select a element2 from S_{d2}
  - $step_{23}$: if the set is S_i, randomly select a element2 from S_i

Probability

$$\begin{gathered} p(element1) = \frac{1}{n_1}, p(element2) = \frac{1}{n_2} \\ p(element1 = element2) = \frac{n_i}{n_1 * n_2} \end{gathered} \tag{1}$$

Dependently Sampling Algorithm I

Idea

The $step_{13}$ is same as $step_{23}$. We can merge them.

Algorithm

- S_1:
  - $step_{11}$: randomly select a set from S_i and S_{d1}
  - $step_{12}$: if the set is S_{d1}, randomly select a element1 from S_{d1}
  - $step_{13}$: if the set is S_i, go to $step_{33}$
- S_2:
  - $step_{21}$: randomly select a set from S_i and S_{d2}
  - $step_{22}$: if the set is S_{d2}, randomly select a element2 from S_{d2}
  - $step_{23}$: if the set is S_i, go to $step_{33}$
- $step_{33}$: randomly select a element from S_i

Dependently Sampling Algorithm I

Probability

$$\begin{aligned} p(element1) &= \frac{1}{n_1} \\ p(element2) &= \frac{1}{n_2} \\ p(element1 = element2) &= \frac{n_i}{n_1} * \frac{n_i}{n_1} = \frac{n_i^2}{n_1 * n_2} \end{aligned} \tag{2}$$

Example

$S_1 = \{1, 2, 3\} \cup \{4, 5\}$; $S_2 = \{1, 2, 3\} \cup \{6, 7\}$

- As for S_1, select set $\{1, 2, 3\}$ with probability 0.6
- As for S_2, select set $\{1, 2, 3\}$ with probability 0.6
- Finally, randomly sampling element in $\{1, 2, 3\}$
- $p(element1 = element2) = 0.6 * 0.6 = 0.36$

Dependently Sampling Algorithm I

Idea

The $step_{11}$ and $step_{21}$ are similar. We can merge them.

Algorithm

- $step_1$: randomly select a set from S_i and S_{d1}
- S_1:
  - $step_{12}$: if the set is S_{d1}, randomly select a element1 from S_{d1}
  - $step_{13}$: if the set is S_i, go to $step_{33}$
- S_2:
  - $step_{22}$: if the set is not S_{d1}, randomly select a element2 from S_{d2}
  - $step_{23}$: if the set is S_i, go to $step_{33}$
- $step_{33}$: randomly select a element from S_i

Problem

Probability

As for S_1:

$$\begin{aligned} p_1(S_i) &= \frac{n_i}{n_1} \\ p_1(S_{d1}) &= 1 - \frac{n_i}{n_1} \end{aligned} \tag{3}$$

As for S_2:

$$\begin{aligned} p_2(S_i) &= \frac{n_i}{n_2} \\ p_2(S_{d2}) &= 1 - \frac{n_i}{n_2} \end{aligned} \tag{4}$$

Problem

if $n_1 \neq n_2$,
then $p_1(S_i) \neq p_2(S_i)$ and $p_1(S_{d1}) \neq p_2(S_{d2})$

Case1: n1 < n2

Problem

$$(p_1(S_i) = \frac{n_i}{n_1}) > (p_2(S_i) = \frac{n_i}{n_2}) \tag{5}$$

Approach

So in $step_1$, when select S_i, it should be changed S_{d2} in probability of p. The equation is

$$\begin{aligned} p_2(S_i) &= p_1(S_i) * (1-p) = \frac{n_i}{n_2} \\ p_2(S_{d2}) &= p_2(S_{d1}) + p_1(S_i) * p = 1 - \frac{n_i}{n_2} \end{aligned} \tag{6}$$

then

$$p = 1 - \frac{n_1}{n_2}$$

Dependently Sampling Algorithm II

Algorithm

- $step_1$: randomly select a set S from S_i and S_{d1}
- S_1:
  - $step_{12}$: if the S is S_{d1}, randomly select a element1 from S_{d1}
  - $step_{13}$: if the S is S_i, go to $step_3$
- S_2:
  - $step_{22}$: if the S is S_{d2}, randomly select a element2 from S_{d2}
  - $step_{23}$: if the S is S_i
    - randomly let $S = S_{d1}$ in probability of $1 - \frac{n_1}{n_2}$
    - if S is S_{d2}, randomly select a element2 from S_{d2}
    - if S is S_i, go to $step_3$
- $step_3$: randomly select a element from S_i

Dependently Sampling Algorithm II

Probability

$$p(element1 = element2) = p_1(S_i) * (\frac{n_1}{n_2}) = \frac{n_i}{n_2}$$

Example

$S_1 = \{1, 2, 3\} \cup \{4\}$; $S_2 = \{1, 2, 3\} \cup \{6, 7\}$

- In $step_1$, select set $\{1, 2, 3\}$ with probability $\frac{3}{5} = 0.75$
- Then as for S_2, select set $\{1, 2, 3\}$ with probability $\frac{4}{5} = 0.8$
- Finally, randomly sampling element in $\{1, 2, 3\}$
- $p(element1 = element2) = 0.75 * 0.8 = 0.6$

Case1: n1 > n2

Problem

$$(p_1(S_{d1}) = 1 - \frac{n_i}{n_1}) > (p_2(S_{d_2}) = 1 - \frac{n_i}{n_2}) \tag{7}$$

Approach

So in $step_1$, when select S_{d1}, it should be changed S_i in probability of p. The equation is

$$\begin{aligned} p_2(S_i) = p_1(S_i) + p_2(S_{d1}) * p = \frac{n_i}{n_2} \\ p_2(S_{d2}) = p_2(S_{d1}) * (1 - p) = 1 - \frac{n_i}{n_2} \end{aligned} \tag{8}$$

then

$$p = 1 - \frac{n1 * (n2 - n_c)}{n2 * (n1 - n_c)}$$

Dependently Sampling Algorithm II

Algorithm

- $step_1$: randomly select a set S from S_i and S_{d1}
- S_1:
  - $step_{12}$: if the S is S_{d1}, randomly select a element1 from S_{d1}
  - $step_{13}$: if the S is S_i, go to $step_3$
- S_2:
  - $step_{22}$: if the S is S_{d2}, randomly select a element2 from S_{d2}
    - randomly let $S = S_i$ in probability of $1 - \frac{n1*(n2-n_c)}{n2*(n1-n_c)}$
    - if S is S_{d2}, randomly select a element2 from S_{d2}
    - if S is S_i, randomly select a element2 from S_i
  - $step_{23}$: if the S is S_i, go to $step_3$
- $step_3$: randomly select a element from S_i

Dependently Sampling Algorithm II

Probability

$$p(element1 = element2) = p_1(S_i)) = \frac{n_i}{n_1}$$

Example

$S_1 = \{1, 2, 3\} \cup \{4, 5\}$; $S_2 = \{1, 2, 3\} \cup \{6\}$

- In $step_1$, select set $\{1, 2, 3\}$ with probability $\frac{3}{4} = 0.6$
- Then S_2 must select $\{1, 2, 3\}$
- Finally, randomly sampling element in $\{1, 2, 3\}$
- $p(element1 = element2) = 0.6$

Why not shuffle $S_1 \cup S_2$

Probability

$p(element1 = element2) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|} < \frac{|S_1 \cap S_2|}{max(|S_1|,|S_2|)}$

randomly select in $S_1 \cup S_2$: "starvation"

if $|S_1| = 99, |S_2| = 1$, then
$p(element \in S_1) = 0.99, p(element \in S_2) = 0.01$
So task2 may be starving

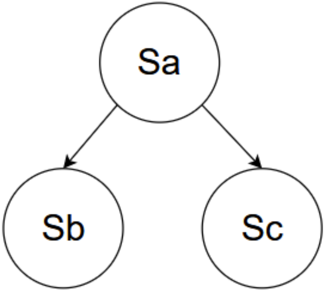
shuffle $S_1 \cup S_2$: offset

$S_1 = \{3, 4\}, S_2 = \{1, 3, 4\}$
$\{\quad 3,4\}$
$\{1,2,3,4\}$
$\{1,\ \ 3,4\}$

Sampling Tree

Attributes

- There are two sets S_1, S_2 and $n_1 > n_2$
- $S_a = S_a \cap S_b$, whose length is n_a
- $S_b = S_1 - S_a$, whose length is n_b
- $S_c = S_2 - S_a$, whose length is n_c



case1 and case2

Algorithm

- case1: $p_1 \leq n_a/n_1$ and $p_2 \leq n1/n_2$
  - $S_a = S_a - \{a\}$
- case2: $p_1 > n_a/n_1$:
  - $S_b = S_a - \{b\}$
  - $S_c = S_c - \{c\}$

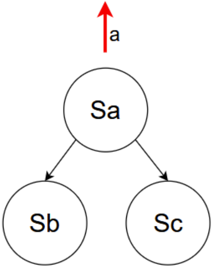


Figure: case1

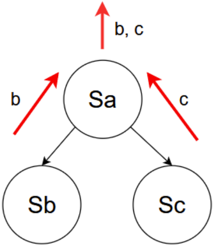


Figure: case2

case3 and case4

Algorithm

- case3: $p_1 \leq n_a/n_1$ and $p_2 > n1/n_2$
  - $S_b = S_b - \{b\}$
  - $S_c = S_c \cup \{a\} - \{c\}$

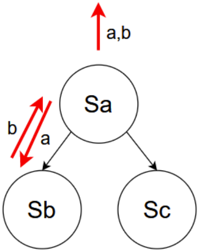


Figure: impossible

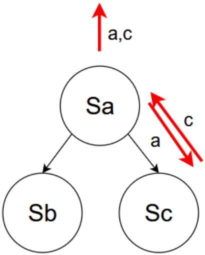
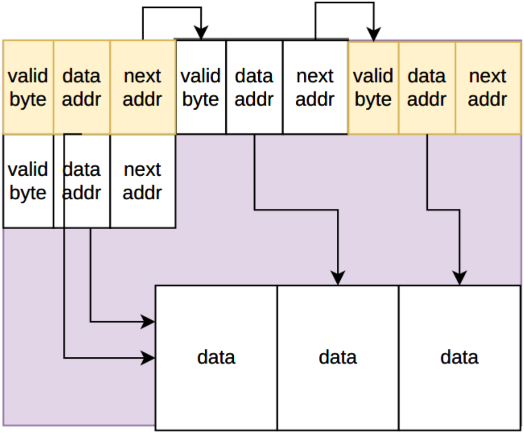


Figure: case3

Buffer Pool: Data Structure

data

- There are two kinds of nodes: inode and datanode
- Every task has a head inode address



Buffer Pool: Valid Byte

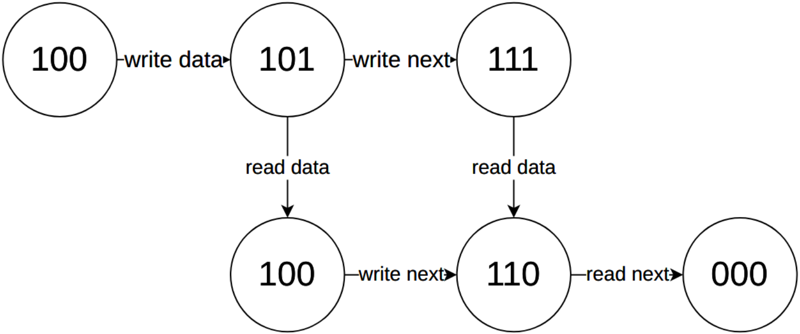
valid byte

- data bit: If the data bit is equal to 1, the data addr is valid. Otherwise invalid
- next bit: If the next bit is equal to 1, the next addr is valid. Otherwise invalid
- used bit: If the used bit is equal to 1, this inode is used by some tasks

Buffer Pool: Automata

valid byte

| used bit | next bit | data bit |



Buffer Pool: allocate inode

case1

There is enough free space to allocate

```
    if inode_tail + inode_size > data_head:
        return inode_tail
```

case2

Free Some unused inode

```
    for head in all_heads:
        if check_free(head) is True:
            return head
```

Buffer Pool: allocate data node

case1

There is enough free space to allocate

```
if inode_tail + inode_size > data_head:
    return inode_tail
```

case2

Free Some unused datanode

```
free = True
for datanode in all_datanodes:
    for ref in refs of datanode:
        if databit(ref) == 0 && dataaddr(ref) == datanode:
            free = False
            break
    if free is True:
        return datanode
```

Table of Contents

1 Introduction

2 Global DataLoader

3 Experiment

Experiment

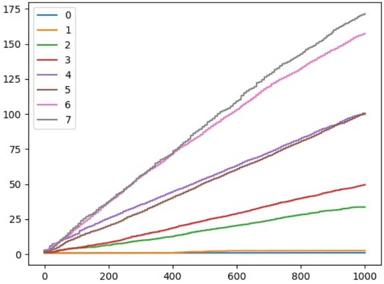


Figure: time

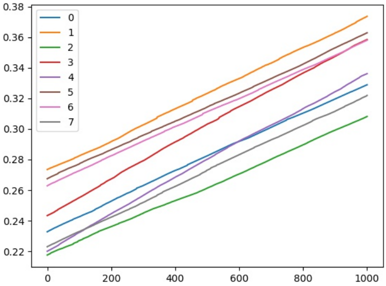


Figure: time with GlobalDataLoader