Discussion on the Project of Analyzing Running Time of Stochastic Gradient Methods
The Analysis I

- You should identify the major operations
- Then check percentage of these operations
- From the results you draw some conclusions
- A typical analysis is as follows. In the forward procedure,

\[ W^m_{\text{mat}}(P^m_{\phi} P^m_{\text{pad}} \text{vec}(Z^{m,i})) = W^m_{\phi}(\text{pad}(Z^{m,i})) \]

\[ \phi(\text{pad}(Z^{m,i})) : \mathcal{O}(l \times h^m h^m d^m a_{\text{conv}}^m b_{\text{conv}}^m) \]

\[ W^m_{\phi}(\cdot) : \mathcal{O}(l \times h^m h^m d^m d^{m+1} a_{\text{conv}}^m b_{\text{conv}}^m) \]
The Analysis II

\[ Z^{m+1,i} = \text{mat}(P_{\text{pool}}^{m,i} \text{vec}(\sigma(S^{m,i}))) \]

\[ \mathcal{O}(l \times h^m h^m d^{m+1} a_{\text{conv}}^m b_{\text{conv}}^m) \]

- From this complexity analysis, matrix-matrix products are the bottleneck.
- Roughly, \( d^m \) times more than others at layer \( m \).
- However, we see that matrix-matrix products take less than half (feedforward part).
- Thus optimized BLAS is very effective.
It is easier to optimize the computationally heavy part.

But this also means that probably there is still room to improve other operations such as

$$\phi (\text{pad}(Z^{m,i}))$$

You should compare practical time and theoretical complexity, otherwise we do not know if you have understood the derived operations.
Why is Tensorflow Faster? I

Before a rough answer of this issue, let’s discuss some differences between Tensorflow and our MATLAB-based implementation

- Automatic differentiation
- Computational graphs
We will discuss this in regular lectures
Let’s borrow a description from https://deepnotes.io/tensorflow “Tensorflow approaches series of computations as a flow of data through a graph with nodes being computation units and edges being flow of Tensors (multidimensional arrays).”

This means that we must build a graph first before the execution.

This is a bit unnatural.

For example, to do a matrix product.
C = A*B;
we cannot just write the above statement like in MATLAB.
We need two steps
- But using a graph does have some advantages
- One is the effective parallel computation
- Operations that are independent to each other (e.g., they need different input data) can be conducted in parallel
- Therefore, operations can be scheduled in a more efficient manner.
Computational Graphs III

- In contrast, our MATLAB code is a procedural setting.

- For example, in the feed forward process for function evaluation we have

\[
S^{m,i} = W^m \phi(pad(Z^{m,i}))_{h^m h^m d^m \times a^m_{\text{conv}} b^m_{\text{conv}}} + b^m_{1 \text{ } T} a^m_{\text{conv}} b^m_{\text{conv}}, \quad i = 1, \ldots, l
\]

(1)

and

\[
Z^{m+1,i} = \text{mat}(P_{\text{pool}}^m,i \text{ vec}(\sigma(S^{m,i})))_{d^{m+1} \times a^{m+1} b^{m+1}}, \quad i = 1, \ldots, l
\]

(2)

- Remember that this is over all data (in a batch).
Thus if a graph has been constructed, easily (1)-(2) can be done in parallel
Why is Tensorflow Faster? I

- Now let’s go back to this issue
- We think there are some possible reasons
- Some MATLAB operations are not efficiently implemented

You have seen that index manipulation is time consuming

We will try to make improvements in the next project

- Tensorflow’s setting by computational graph leads to better overall optimization?
Tensorflow may have used some optimized packages dedicated to neural networks.

For example, they may use

Intel MKL-DNN:
https://github.com/intel/mkl-dnn