A PARALLEL BATCH TRAINING ALGORITHM FOR DEEP NEURAL NETWORK

Yuping Lin

IFLYTEK Laboratory for Neural Computing for Machine Learning
Department of Electrical Engineering and Computer Science
York University, Toronto

October 13, 2015
What does human brain do?

cat
What does human brain do?

Function → cat
What does neural network do?
Typical structure of NN

- Has multiple layers;
- Each layer has many units (a.k.a. neurons);
- Units are connected by edges;
- Each edge is associated with a weight;
Cycle for NN training

- Forward phase
  - $Z_j^{(l+1)} = F \left( \sum_i w_{ij} \cdot Z_i^{(l)} + b_j \right)$
  - Where $F(x)$ is the nonlinear activation function
Cycle for NN training

- Error back propagation
  - $\delta^{(out)}_k = Z^{(out)}_k - T^{(out)}_k$
  - $\delta^{(l)}_i = F'(Z^{(l)}_i) \cdot \sum_j w_{ij} \cdot \delta^{(l+1)}_j$
  - Where $F'(x)$ is the derivative of the activation function
  - $T$ is the desired output vector

- Weight updating
  - $\Delta w_{ij} = Z^{(l)}_i \cdot \delta^{(l+1)}_j$
  - $w_{ij} = w_{ij} - \gamma \cdot \Delta w_{ij}$
  - Where $\gamma$ is the learning rate
Applications

• Computer vision
  • Multi-column DNN, \(0.23\%\) error rate on MNIST (D. Ciresan et al., 2012)

• Speech recognition
  • Bidirectional LSTM, \(17.7\%\) PER on TIMIT (A. Graves et al., 2013)

• Natural Language Processing
  • S-LSTM, \(81.9\%\) accuracy on Stanford Sentiment Treebank (X. Zhu et al. 2015)
Heavy computation load

- Take as example an simple feed-forward NN with 2 hidden layers of size 100. ([100-100-100-1])
  - Has approximately 20,100 parameters;
  - Perform at least 20,100 multiplications in forward phase, for each train sample;
  - Perform at least 40,300 multiplications in error back propagation phase, for each train sample;
  - Plus other operations;
Heavy computation load

Parallelize
Multi-Layer Perceptron (MLP)

- The authors use an MLP with 2 hidden layers.
- Pre-trained layer by layer with RBMs.
- Then fine-tuned using error back propagation algorithm
Restricted Boltzmann Machine (RBM)

- Has 2 layers of units. Called visible units and hidden units.
- Trained using contrastive divergence algorithm:
  - From inputs $\hat{v}^{(0)}$ compute $\hat{h}^{(1)}$;
  - From $\hat{h}^{(1)}$ compute $\hat{v}^{(1)}$;
  - From $\hat{v}^{(1)}$ compute $\hat{h}^{(2)}$;
  - Compute weight update as: $\Delta w_{ij} = v_i^{(0)} \cdot h_j^{(1)} - v_i^{(1)} \cdot h_j^{(2)}$;
- Use the trained weights to initialize MLP.
Parallel nature of batch training

- For batch training, weight updates only occur at the end of every batch.
- For each train sample in the mini-batch:
  - Feed input into NN and generate output;
  - Back propagate errors;
  - Accumulate update statistics.
- Update weights after the whole mini-batch is traversed.
- The training of each sample within a mini-batch is independent.
Parallel batch training

- The proposed parallel batch training algorithm uses a single Master thread with many Worker threads.
- Within each mini-batch, the Master first distributes train data to Workers. Then, after all the workers finish training, the Master collects training statistics from workers and updates weights.
Parallel batch training

- Need synchronization to ensure:
  - Training statistics are collected after and only after all workers finished their training;
  - All Workers start next training iteration only after Master has updated the weights and distributed new training data to them.
Implementation using a monitor

Train: Monitor

Begin
  Procedure trainSetup (numberOfWorkers : int)

  Procedure finishTraining ()

  Procedure requestUpdate (result : boolean)

End
Implementation plan

• Plan:
  • Implement both the sequential and parallel version of the batch training algorithm.
  • Use training time and testing error rate as comparison criteria.

• Challenges:
  • Need parameter tuning;
  • Shorten the waiting time;
References

