Parallel Methods for Neuron Network

Haihao Lu

Problem of Interest: ANN

$$\min_{W,b} \frac{1}{n} \sum_{k=1}^{n} l(\phi_l(W_l \phi_{l-1}(\cdots \phi_1(W_1 x_k + b_1) \cdots + b_l)), y_k)$$

- W is the weight
- b is the bias
- ϕ is activation function
- x is feature
- y is label
- l is loss function

Drawback of SGD

- SGD is a first-order method, thus it converges slow.
- SGD suffers from vanishing gradient problem
- Most importantly, it is hard to parallelize SGD

$$\min_{W,b} \frac{1}{n} \sum_{k=1}^{n} l(\phi_l(W_l \phi_{l-1}(\cdots \phi_1(W_1 x_k + b_1) \cdots + b_l)), y_k)$$

- Equivalent Problem

$$\min_{W,b,z} \frac{1}{n} \sum_{k=1}^{n} l(\phi_l(z_l^k), y_k)$$

$$s.t. z_l^k = W_l \phi_{l-1}(z_{l-1}^k) + b_l$$

$$\cdots$$

$$z_2^k = W_2 \phi_1(z_1^k) + b_2$$

$$z_1^k = W_1 x^k + b_1$$

- Relaxed Problem

$$\min_{W,b,z} \frac{1}{n} \sum_{k=1}^{n} l(\phi_l(z_l^k), y_k) + \sum_{i=1}^{l} \mu_i \| z_i^k - W_i \phi_{i-1}(z_{i-1}^k) - b_i \|_2^2$$

$$\min_{W,b,z} \frac{1}{n} \sum_{k=1}^{n} l(\phi_l(z_l^k), y_k) + \sum_{i=1}^{l} \mu_i \| z_i^k - W_i \phi_{i-1}(z_{i-1}^k) - b_i \|_1$$

Alternating Minimization

- W-update

$$[W_i, b_i] = \left(\sum_k z_i^k [\phi_{i-1}(z_{i-1}^k; 1)]\right) \times \left(\sum_k [\phi_{i-1}(z_{i-1}^k; 1)][\phi_{i-1}(z_{i-1}^k; 1)]^T\right)^{-1}$$

- z-update

$$\min_z l(\phi_l(z_l^k), y_k) + \sum_{i=1}^{l} \mu_i \|z_i^k - W_i \phi_{i-1}(z_{i-1}^k) - b_i\|_2^2$$

one or more steps of damped Newton

cheap to compute Hessian

parallel computing

RNN

$$\min_{W,V,b} \frac{1}{n} \sum_{k=1}^{n} l(\phi_l(W_l \phi_{l-1}(\cdots \phi_1(W_1 x_k + V_1 s_0 + b_1) \cdots + V_l s_l + b_l)), y_k)$$

- Equibalent Problem

$$\begin{aligned} &\min_{W,b,z} \frac{1}{n} \sum_{k=1}^{n} l(\phi_l(z_l^k), y_k) \\ &s.t. z_l^k = W_l \phi_{l-1}(z_{l-1}^k) + V_l s_{l-1}^k + b_l \\ &\quad \cdots \\ &\quad z_2^k = W_2 \phi_1(z_1^k) + V_2 s_1^k + b_2 \\ &\quad z_1^k = W_1 x^k + V_1 s_0^k + b_1 \end{aligned}$$

- Relaxed Problem

$$\min_{W,b,z} \frac{1}{n} \sum_{k=1}^{n} l(\phi_l(z_l^k), y_k) + \sum_{i=1}^{l} \mu_i \| z_i^k - W_i \phi_{i-1}(z_{i-1}^k) - V_i s_{i-1}^k - b_i \|_2^2$$

With L_2 Regulation

- W-update

$$[W_i, b_i] = \left(\sum_k z_i^k [\phi_{i-1}(z_{i-1}^k; 1)]\right) \times \left(\sum_k [\phi_{i-1}(z_{i-1}^k; 1)][\phi_{i-1}(z_{i-1}^k; 1)]^T + \frac{\lambda}{\mu_i} I\right)^{-1}$$

- z-update

$$\min_z l(\phi_l(z_l^k), y_k) + \sum_{i=1}^{l} \mu_i \| z_i^k - W_i \phi_{i-1}(z_{i-1}^k) - b_i \|_2^2$$

one or more steps of damped Newton

cheap to compute Hessian

parallel computing

Stochastic Method

- W-update

$$[nW_i, nb_i] = \left(\sum_{k\in I} z_i^k[\phi_{i-1}(z_{i-1}^k;1)]\right) \times \left(\sum_{k\in I}[\phi_{i-1}(z_{i-1}^k;1)][\phi_{i-1}(z_{i-1}^k;1)]^T\right)^{-1}$$

$$[W_i, b_i] = s[W_i, b_i] + (1-s)[nW_i, nb_i]$$

- z-update

$$\min_z l(\phi_l(z_l^k), y_k) + \sum_{i=1}^{l} \mu_i \|z_i^k - W_i\phi_{i-1}(z_{i-1}^k) - b_i\|_2^2$$

Just compute all k in next batch

Compared with: Training Neural Networks Without Gradients: A Scalable ADMM Approach (ICML,2016)

Ours

initial problem

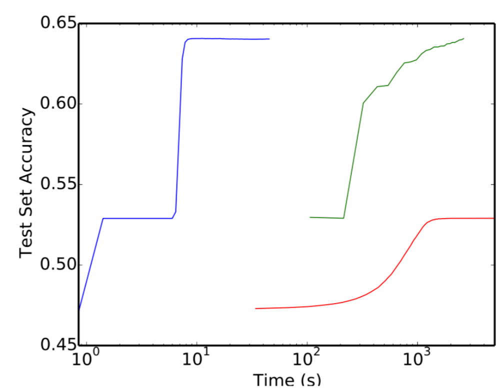
$$\min_{W,b,z} \frac{1}{n}\sum_{k=1}^{n} l(\phi_l(z_l^k), y_k)$$
$$s.t. z_l^k = W_l\phi_{l-1}(z_{l-1}^k) + b_l$$
$$\cdots$$
$$z_2^k = W_2\phi_1(z_1^k) + b_2$$
$$z_1^k = W_1 x^k + b_1$$

relaxed problem $\min_{W,b,z} \frac{1}{n}\sum_{k=1}^{n} l(\phi_l(z_l^k), y_k) + \sum_{i=1}^{l} \mu_i \|z_i^k - W_l\phi_{i-1}(z_{i-1}^k) - b_i\|_2^2$

Theirs

$$\min_{W,b,t} \frac{1}{n}\sum_{k=1}^{n} l(t_l^k, y_k)$$
$$s.t. t_l^k = \phi_l(z_l^k)$$
$$\cdots$$
$$t_1^k = \phi_1(z_1^k)$$
$$z_l^k = W_l t_{l-1}^k + b_l$$
$$\cdots$$
$$z_1^k = W_1 x^k + b_1$$

Three block ADMM: unstable



(b) **Test set predictive accuracy as a function of time** for ADMM on 7200 cores (blue), conjugate gradients (green), and SGD (red). Note the x-axis is scaled logarithmically.