DS-MLR: Scaling Multinomial Logistic Regression via Hybrid Parallelism

Parameswaran Raman[1], Sriram Srinivasan[1], Shin Matsushima[2], Xinhua Zhang[3], Hyokun Yun[4], S.V.N. Vishwanathan[4]

[1]University of California at Santa Cruz [2]The University of Tokyo [3]University of Illinois at Chicago [4]Amazon

Code: https://bitbucket.org/params/dsmlr

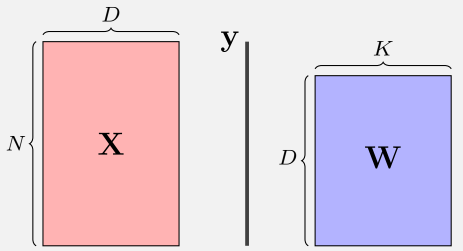
Multinomial Logistic Regression (MLR)

Given:

Training data and labels

Goal:

Learn a model W

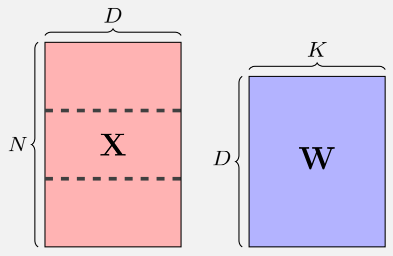


$$p(y_i = k|\mathbf{x}_i, W) = \frac{\exp(\mathbf{w}_k{}^T\mathbf{x}_i)}{\sum_{j=1}^{K}\exp(\mathbf{w}_j{}^T\mathbf{x}_i)}$$

Assume: N, D and K are large $(N >>> D >> K)$

Popular ways to distribute MLR

Data parallel (partition data, duplicate parameters)

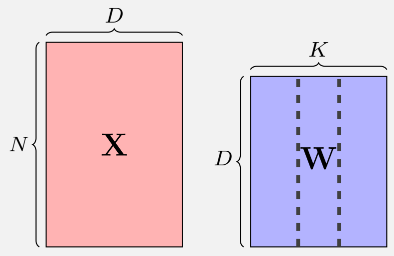


Storage Complexity:

$O(\frac{ND}{\mathbf{P}})$ data, $O(KD)$ model

e.g. L-BFGS

Model parallel (partition parameters, duplicate data)



Storage Complexity:

$O(ND)$ data, $O(\frac{KD}{\mathbf{P}})$ model

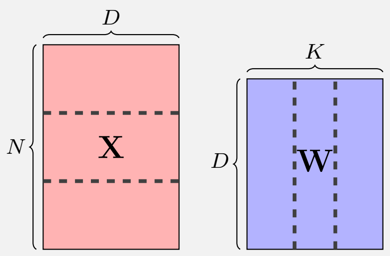
e.g. LC [Gopal et al 2013]

Can we get the best of both worlds?

Our Solution: Doubly-Separable MLR (DS-MLR)

- Double Separability naturally leads to **Hybrid Parallelism**
- **Asynchronous** and fully **Decentralized** algorithm
- Avoids expensive **bulk-synchronization**
- Scales to Reddit-Full dataset (**211 million** data points and **44 billion** parameters)

Hybrid Parallel



Storage Complexity:

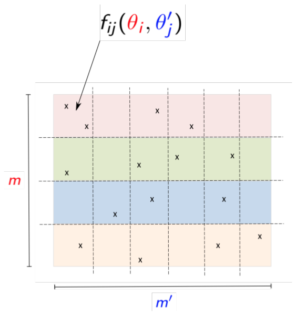
$O(\frac{ND}{\mathbf{P}})$ data, $O(\frac{KD}{\mathbf{P}})$ model

Bottleneck to Model Parallelism in MLR

$$\min_{W} L(W) = \frac{\lambda}{2}\sum_{k=1}^{K}\|\mathbf{w}_k\|^2 - \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} y_{ik}\mathbf{w}_k{}^T\mathbf{x}_i + \frac{1}{N}\sum_{i=1}^{N}\underbrace{\log\left(\sum_{k=1}^{K}\exp(\mathbf{w}_k{}^T\mathbf{x}_i)\right)}_{\text{makes model parallelism hard}}$$

Reformulation into Doubly-Separable form

$$f(\theta_1, \theta_2, \ldots, \theta_m, \theta'_1, \theta'_2, \ldots, \theta'_{m'}) = \sum_{i=1}^{m}\sum_{j=1}^{m'} f_{ij}(\theta_i, \theta'_j)$$



Each sub-function f_{ij} can be computed **independently** and in **parallel**

Step 1: Introduce redundant constraints (new parameters A) into the original MLR problem

$$\min_{W,A} \quad L_1(W, A) = \frac{\lambda}{2}\sum_{k=1}^{K}\|\mathbf{w}_k\|^2 - \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} y_{ik}\mathbf{w}_k{}^T\mathbf{x}_i - \frac{1}{N}\sum_{i=1}^{N}\log a_i$$

$$\text{s.t.} \quad a_i = \frac{1}{\sum_{k=1}^{K}\exp(\mathbf{w}_k{}^T\mathbf{x}_i)}$$

Step 2: Turn the problem to unconstrained min-max problem by introducing Lagrange multipliers $\beta_i, \forall i = 1, \ldots, N$

$$\min_{W,A}\max_{\beta} \quad L_2(W, A, \beta) = \frac{\lambda}{2}\sum_{k=1}^{K}\|\mathbf{w}_k\|^2 - \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} y_{ik}\mathbf{w}_k{}^T\mathbf{x}_i - \frac{1}{N}\sum_{i=1}^{N}\log a_i + \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K}\beta_i\, a_i \exp(\mathbf{w}_k{}^T\mathbf{x}_i) - \frac{1}{N}\sum_{i=1}^{N}\beta_i$$

Step 3: Observations in the Primal-Dual updates

- When a_i^{t+1} is solved to optimality, it admits an exact closed-form solution given by $a_i^* = \frac{1}{\beta_i \sum_{k=1}^{K}\exp(\mathbf{w}_k{}^T\mathbf{x}_i)}$.
- Dual-ascent update for β_i is no longer needed, since the penalty is always zero if β_i is set to a constant equal to 1.

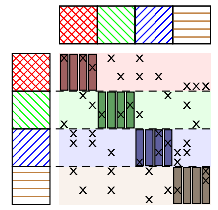
DS-MLR

$$\min_{W,A} \sum_{i=1}^{N}\sum_{k=1}^{K}\left(\frac{\lambda\|\mathbf{w}_k\|^2}{2N} - \frac{y_{ik}\mathbf{w}_k{}^T\mathbf{x}_i}{N} - \frac{\log a_i}{NK} + \frac{\exp(\mathbf{w}_k{}^T\mathbf{x}_i + \log a_i)}{N} - \frac{1}{NK}\right)$$

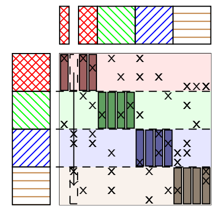
Doubly-Separable form for MLR

Parallelization

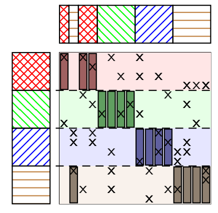
NOMAD [Yun et al 2014]



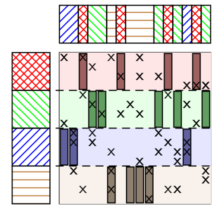
Initial Assignment of W and A



worker 1 updates $\mathbf{w}_2$ and communicates it to worker 4

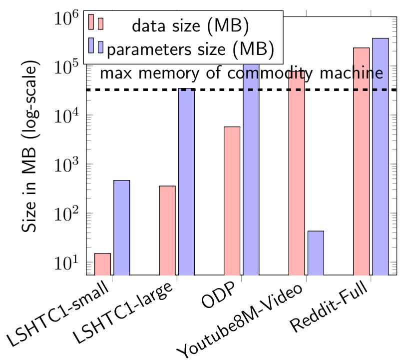


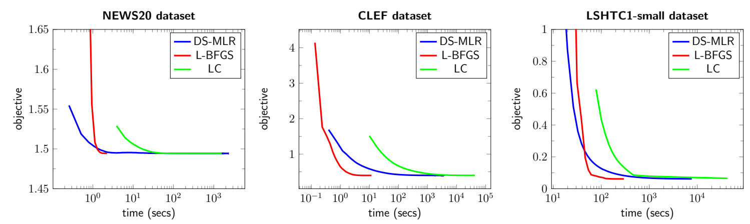
worker 4 can now update $\mathbf{w}_2$



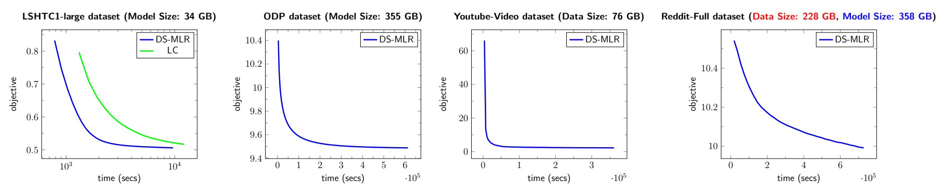
Ownership of $\mathbf{w}_k$ changes continuously.

Experiments





Single-Machine experiments (Data and Model both fit in memory).



Multi-Machine experiments.