Asynchronous Multi-Task Learning

Inci M. Baytas, Ming Yan, Anil K. Jain and Jiayu Zhou

December 14th, 2016

Outline

1. Introduction
2. Solving Regularized MTL
3. Distributed MTL
4. Asynchronous Multi-Task Learning (AMTL)
5. Experimental Results
6. Conclusion and Future Work

Introduction

- **Many real-world machine learning applications involve multiple learning tasks**.
- **Tasks are related**.
  - Shared hidden layers
  - Shared parameters
  - Regularization based; shared subspace, joint feature.
- **Multi-Task Learning (MTL)**
  - Simultaneously learn related tasks.
  - Perform inductive knowledge transfer among tasks.

Introduction

- **How to pose task relatedness?**
  - Subspace learning; regularized MTL.

$$\min_{W} \sum_{t=1}^{T} \ell_t(w_t) + \lambda \|W\|_*$$

where $W \in \mathbb{R}^{d \times T}$ is the model matrix, and λ is the regularization parameter.

Solving Regularized MTL

- **Centralize the data**
- **Apply forward-backward splitting**
    - Calculate gradient of loss function ($\sum_{t=1}^{T} \ell_t(w_t)$)
    - Move in the negative gradient direction ($\hat{W} = W^k - \eta \nabla_{W^k} \sum_{t=1}^{T} \ell_t(w_t)$).
    - **Apply proximal mapping to obtain the updated model matrix.**
        - Project the intermediate model matrix ($\hat{W}$) to the solution domain.
- **Limitation:** Data centralization is not always possible!

Regularized MTL: Limitations

- Data servers are in different locations.
- **Data cannot be frequently transferred over the network.**
  - Privacy reasons
  - Communication cost; large scale datasets and limited bandwidth
- **Solution:**
  - Distributed MTL

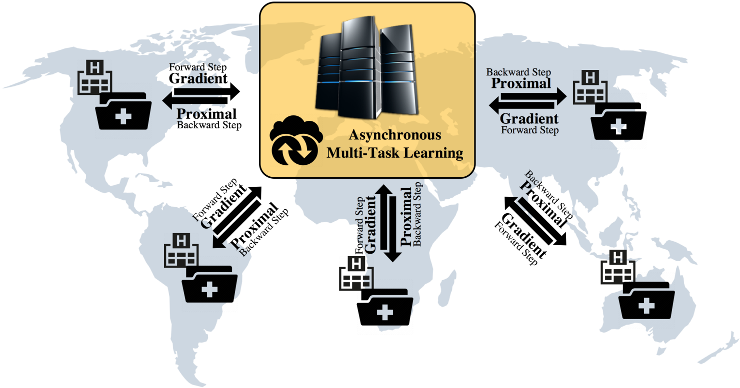
Distributed MTL

- **Synchronized distributed MTL (SMTL)**
  - One central node and T task nodes.
  - **In each task node:**
    - $\hat{w}_t = w_t^k - \eta \nabla \ell_t(w_t^k)$
    - Send $\hat{w}_t$ to central node.
  - **In central node**:
    - Wait for each task node to finish its computations and send $\hat{w}_t$.
    - Construct the model matrix $\hat{W}$ and perform proximal mapping.

Synchronous MTL: Limitations

- Slow due to data imbalance.
- Slow due to communication delays.
- Not robust against network failures.
- **Proposed Solution:** Asynchronous MTL (AMTL)
  - Central node does not wait for all task nodes.
  - Robust against communication delays.
  - Linear convergence is guaranteed.

Asynchronous MTL Overview



Asynchronous MTL Framework

- AMTL framework is based on AROCK [1], an asynchronized coordinate descent framework.
- In AMTL, each model is a block of coordinates.
- A generic Krasnoselskii−Mann (KM) iteration framework to solve fixed point problems. $\Rightarrow$ Linear convergence
- KM iteration update rule:

$$x^{k+1} = x^k + \eta_k \left(F\left(x^k\right) - x^k \right) \tag{1}$$

where F is a fixed point operator.

[1] Z. Peng et.al, "ARock: An algorithmic framework for asynchronous parallel coordinate updates", SIAM, vol. 38, no. 5, pp. A2851-A2879, 2016.

Optimization

- AMTL uses backward-forward splitting (BFS) as the fixed point operator.

$$w_t^{k+1} = w_t^k + \eta_k \left(F_{BFS}\left(\hat{w}^k\right) - w_t^k \right)$$

$$F_{BFS}\left(\hat{w}^k\right) = \left(\mathsf{Prox}_{\eta\lambda}\left(\hat{w}^k\right)\right)_t - \eta \nabla \ell_t \left(\left(\mathsf{Prox}_{\eta\lambda}(\hat{w}^k) \right)_t \right)$$

- **Why BFS?**
  - Reduced communication between central and task nodes in each iteration.
- Changing order does not effect convergence.

Backward-Forward Splitting

- Forward step (gradient step) can be decoupled.

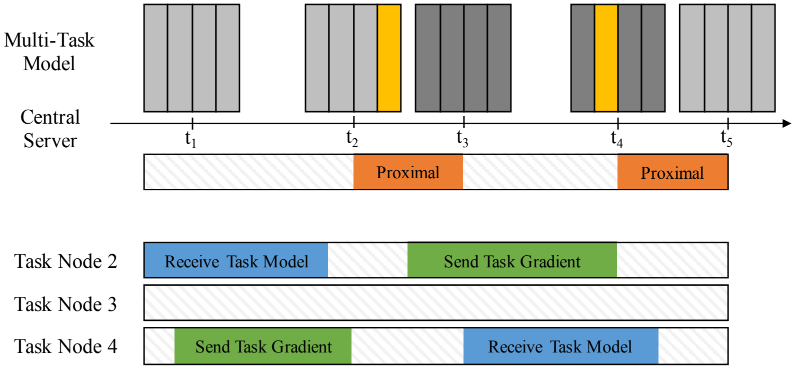
$$\nabla f(W) = \nabla \sum_{t=1}^{T} \ell_t(w_t) = [\nabla \ell_1(w_1), \ldots, \nabla \ell_T(w_T)].$$

- However, backward step (proximal maping) is not decoupled.

$$\mathsf{prox}_{\eta\lambda}(\hat{W}) = \arg\min_{W} \frac{1}{2\eta} \|W - \hat{W}\|_F^2 + \lambda \|W\|_*$$

- Backward step should be performed in the central node.
- One extra backward step is needed at the end of the last iteration.

AMTL: Update Mechanism



Dynamic Step Size

- In reality, each task node does not have the same activation rate; **network delays!**

$$w_t^{k+1} = w_t^k + c_{(t,k)} \eta_k \left(F_{BFS} \left(\hat{w}^k \right) - w_t^k \right)$$

- Longer the delay, larger the step size (η_k); a heuristic approach:

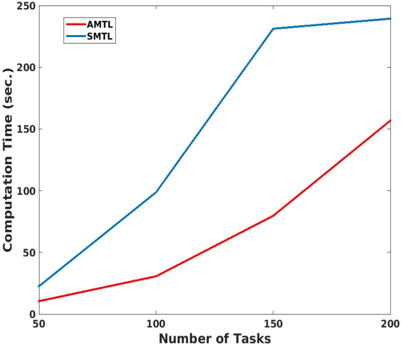
$$c_{(t,k)} = \log \left(\max \left(\bar{\nu}_{t,k}, 10 \right) \right)$$

where $\bar{\nu}_{t,k} = \frac{1}{k} \sum_{i=z-k}^{z} \nu_t^{(i)}$; history of delays.

Experimental Results

- AMTL is implemented by using AROCK framework (MPI in C++).
- Distributed environment is simulated by using shared memory architecture; network delays are artificially introduced.
- Experiments were conducted using an Intel Core i5-5200U CPU (2.20GHz x 4) machine.
- Performance is limited by the hardware specifications of the machine.

Computation Time: AMTL v. SMTL



- Synthetic data
- Square loss

Datasets

- 5 binary classification tasks for MNIST such as 0 vs 9, 1 vs 8, ...
- 4 binary classification tasks for MTFL(Multi-Task Facial Landmark) such as male vs female, smiling vs not smiling, ...

Table: Datasets used in this paper.

Data set	Number of tasks	Sample sizes	Dimensionality
School	139	22-251	28
MNIST	5	13137-14702	100
MTFL	4	2224-10000	10

AMTL and SMTL Comparison

Table: Training time (sec.) comparison of AMTL and SMTL for different datasets. Training time of AMTL is less than the training time of SMTL with different network settings.

Network	School	MNIST	MTFL
SMTL-1	299.79	57.94	50.59
AMTL-1	**194.22**	**54.96**	**50.40**
SMTL-2	298.42	114.85	92.84
AMTL-2	**231.58**	**83.17**	**77.44**
SMTL-3	593.36	161.67	146.87
AMTL-3	**460.15**	**115.46**	**103.45**

Effect of Dynamic Step Size

Table: Final objective values of the synthetic dataset with 5 tasks under different network settings.

Network	Without dynamic step size	Dynamic step size
AMTL-5	163.62	**144.83**
AMTL-10	163.59	**144.77**
AMTL-15	163.56	**143.82**
AMTL-20	168.63	**143.50**

Conclusion and Future Work

- AMTL is a time efficient distributed MTL framework.
- Dynamic step size can boost the convergence in real world network settings.
- AMTL implementation for real world settings.[2]
- Stochastic gradient framework will also be investigated for AMTL.

[2] https://github.com/illidanlab/AMTL

Acknowledgements

This research is supported by:

NSF under grant IIS-1565596, IIS-1615597, and DMS-1621798

ONR under grant N00014-14-1-0631

Thank you!
Questions?