Distributed Personalization

Xu Miao
LinkedIn Corporate
605 W Maude Ave
Sunnyvale, USA 94085
xmiao@linkedin.com

Yitong Zhou
LinkedIn Corporate
2061 Stierln Court
Mountain View, USA 94043
yizhou@linkedin.com

Chun-Te Chu
Microsoft Corporate
1 Microsoft Way
Redmond, USA
chch@microsoft.com

Joel Young
LinkedIn Corporate
605 W Maude Ave
Sunnyvale, USA 94085
joeyoung@linkedin.com

Lijun Tang
LinkedIn Corporate
605 W Maude Ave
Sunnyvale, USA 94085
litang@linkedin.com

Anmol Bhasin
LinkedIn Corporate
605 W Maude Ave
Sunnyvale, USA 94085
abhasin@linkedin.com

ABSTRACT

Personalization is a long-standing problem in data mining and machine learning. Companies make personalized product recommendations to millions of users every second. In addition to the recommendation problem, with the emerging of personal devices, many conventional problems, e.g., recognition, need to be personalized as well. Moreover, as the number of users grows huge, solving personalization becomes quite challenging. In this paper, we formalize the generic personalization problem as an optimization problem. We propose several ADMM algorithms to solve this problem in a distributed way including a new Asynchronous ADMM that removes all synchronous barriers to maximize the training throughput. We provide a mathematical analysis to show that the proposed Asynchronous ADMM algorithm holds a linear convergence rate which is the best to our knowledge. The distributed personalization allows training to be performed in either a cluster or even on a user's device. This can improve the privacy protection as no personal data is uploaded, while personal models can still be shared with each other. We apply this approach to two industry problems, *Facial Expression Recognition* and *Job Recommendation*. Experiments demonstrate more than 30% relative error reduction on both problems. Asynchronous ADMM allows faster training for problems with millions of users since it eliminates all network I/O waiting time to maximize the cluster CPU throughput. Experiments demonstrate 4 times faster than original synchronous ADMM algorithm.

Categories and Subject Descriptors

H.2.8 [Database Application]: Data mining; H.3.4 [Systems and Software]: Distributed systems

General Terms

Big Data, Distributed Optimization, Personalization

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.

KDD '15, August 10-13, 2015, Sydney, NSW, Australia.

Copyright is held by the owner/author(s). Publication rights licensed to ACM.

ACM 978-1-4503-3664-2/15/08 ...\$15.00.

DOI: <http://dx.doi.org/10.1145/2783258.2788626>.

Keywords

Distributed computing, User modeling, ADMMs

1. INTRODUCTION

Personalization is a key component in many applications such as *advertising* [9, 5, 21], *recommendation* [17, 18], and *personalized medicine* [23], where an impression, a product and a type of drug is predicted according to the user's profile. *Sparse Matrix Factorization* approaches [18, 7, 19] that seek a shared low-dimensional representation for both users and items have inspired significant amount of research. Both theoretical work and empirical work demonstrated that with the *low rank* assumption, i.e., very few dominating factors for individual users, these approaches have achieved state-of-the-arts accuracies on many datasets [7]. However, they are restricted by the matrix formulation, and less flexible to incorporate contextual information, therefore, not general enough to address the personalization aspect in broader machine learning problems. For example, in movie recommendation, extra information about movies like the directors and the budgets are not encoded in the rating matrix. In this paper, we address the personalization differently. Each individual is assigned one model with a regularization term to penalize the difference between individual models and a global consensus model. The regularization term allows individual experiences to be shared among the population, while allowing the models to be different from each other. A parameter in the regularization term controls how much of a difference these models are. This formulation has been proposed under *Multi-task Learning* problem [15, 3] where each individual model is a specific task. These tasks are related to each other, so solving one task provides information to solve others. Our proposed algorithm can be applied to these problem naturally, especially for tasks distributed among a network, e.g., remote sensing and multi-agent planning.

Solving this personalization problem in a distributed environment is quite challenging, especially for data from large companies like LinkedIn and Microsoft. In this paper, we study algorithms based on *alternating direction method of multipliers* (ADMM) [6]. We propose several variants of the original ADMM algorithm including a novel *Asynchronous ADMM* algorithm that frees up the waiting barriers to speed optimization. As the number of worker nodes grows, the processing speed, computation power, network delay usually vary among different worker nodes. Iterating until all nodes finish is expensive as nodes have to wait for the slowest worker node before the next iteration starts. Asynchronous training

addresses the problem and achieves better optimization throughput. We provide a mathematical analysis to prove the proposed *Asynchronous ADMM* converges at a comparable rate as the original *Synchronous ADMM*.

We apply our personalization model to solve two industry problems, *facial expression recognition* and *job recommendation*. The experimental results demonstrate a great gain in accuracy, and the proposed Asynchronous ADMM speeds up the training process dramatically.

In summary, our proposed approach enables systematical training of personalized models jointly in a distributed manner. To our knowledge, this is the first solution of large scale personalization problems via asynchronous distributed machine learning. In addition, the proposed asynchronous ADMM is the first algorithm that demonstrates linear convergence both in theory and empirically.

2. DISTRIBUTED OPTIMIZATION

Before detailing the distributed algorithm, we first formulate our personalization framework as an optimization problem. Equation 1 provides a general formalization where $f_i(\mathbf{w}_i)$ is an arbitrary machine learning task for the i -th user, $\mathbf{w}_i$ is the personal model for that user, and $\mathbf{z}$ is the consensus model. Equation 2 shows an example $f_i(\mathbf{w}_i)$ where l is a convex loss function, $\mathbf{x}_{ij}$ is an input feature vector, and y_{ij} is the target to predict. γ is a regularization term inversely coupling the individual and consensus models. With only a handful of samples per model, we need a relatively large γ to prevent overfitting.

$$\min_{\mathbf{z}, \{\mathbf{w}_1 \dots \mathbf{w}_M\}} \left(\frac{1}{2} \|\mathbf{z}\|^2 + \sum_{i=1}^M (f_i(\mathbf{w}_i) + \frac{\gamma}{2} \|\mathbf{w}_i - \mathbf{z}\|_2^2) \right) \quad (1)$$

$$f_i(\mathbf{w}_i) = \sum_{j=1}^{N_i} l(\mathbf{x}_{ij}, y_{ij}; \mathbf{w}_i) \quad (2)$$

We focus on distributed personalization of classification tasks and as such assume that the number of users is proportional to the total number of samples, and finally, that each user's data is drawn *i.i.d.* from the distribution D_i . These are captured in Assumption 1 where N_i is the number of training samples for user i , N is the total number of training samples, and M is the total number of users.

ASSUMPTION 1. *We make the following assumptions:*

1. Let $N = \sum_i^M N_i$ where $M = \mathcal{O}(N)$.
2. For each user $i \in [1 \dots M]$, $\{\mathbf{x}_{ij}, y_{ij}\} \sim D_i$.
3. Network latency is *i.i.d.* with mean τ .

Applying *distributed stochastic optimization* approaches [1, 10, 2, 12] to this distributed personalization problem is non-trivial. Although the consensus model can be obtained, there aren't enough samples to reliably estimate individual user models. In addition, tracking the learning rate for each user is difficult.

We experiment with a synchronous ADMM algorithm [6, 25] from the literature, a novel versioned adaptation, and propose a faster asynchronous algorithm. A key observation is that alternately optimizing $\mathbf{w}_i$ and $\mathbf{z}$ converges slowly — a low sample count per user drives a large γ large coupling the personal and consensus models which, in turn, leads to hysteresis.

2.1 Synchronous ADMM

We first derive a synchronous ADMM to solve Equation 1. By introducing local consensus variables $\mathbf{q}_i$, we rewrite the problem as in Equation 3. Turning it into the augmented Lagrangian, we have

Equation 4.

$$\begin{aligned} \min_{\mathbf{z}, \mathbf{q}, \{\mathbf{w}_i\}} \quad & \sum_{i=1}^M f_i(\mathbf{w}_i) + \frac{\gamma}{2} \|\mathbf{w}_i - \mathbf{q}_i\|^2 + \frac{1}{2} \|\mathbf{z}\|^2 \quad (3) \\ \text{subject to} \quad & \mathbf{q}_i = \mathbf{z}, \forall i \in [1 \dots M] \end{aligned}$$

$$\begin{aligned} \mathcal{L}(\mathbf{z}, \{\mathbf{q}_i, \mathbf{w}_i, \lambda_i\}) = \quad & \sum_{i=1}^M f_i(\mathbf{w}_i) + \frac{\gamma}{2} \|\mathbf{w}_i - \mathbf{q}_i\|^2 + \\ & \frac{\rho}{2} \|\mathbf{q}_i - \mathbf{z}\|^2 + \lambda_i^T (\mathbf{q}_i - \mathbf{z}) + \frac{1}{2} \|\mathbf{z}\|^2 \end{aligned} \quad (4)$$

An synchronous ADMM solution can be obtained by Equations [5,6,7].

$$\begin{aligned} \mathbf{w}_i^{t+1}, \mathbf{q}_i^{t+1} = \quad & \arg \min_{\mathbf{q}_i, \mathbf{w}_i} f_i(\mathbf{w}_i) + \frac{\gamma}{2} \|\mathbf{w}_i - \mathbf{q}_i\|^2 + \\ & \frac{\rho}{2} \|\mathbf{q}_i - \mathbf{z}^t\|^2 + \lambda_i^t (\mathbf{q}_i - \mathbf{z}^t) \end{aligned} \quad (5)$$

$$\begin{aligned} \mathbf{z}^{t+1} = \quad & \arg \min_{\mathbf{z}} \frac{\rho}{2} \|\mathbf{q}_i^{t+1} - \mathbf{z}\|^2 + \\ & \lambda_i^t (\mathbf{q}_i^{t+1} - \mathbf{z}) + \frac{1}{2} \|\mathbf{z}\|^2 \end{aligned} \quad (6)$$

$$\lambda_i^{t+1} = \lambda_i^t + \rho (\mathbf{q}_i^{t+1} - \mathbf{z}^{t+1}) \quad (7)$$

In Equation 5, $\mathbf{q}_i$ has analytical relation to $\mathbf{w}_i$, we obtain the updating rules in Equation [9,10,11,8], where $\mu = \lambda/\rho$.

$$\forall i, \mu_i^{t+1} = \mu_i^t + \mathbf{q}_i^t - \mathbf{z}^t \quad (8)$$

$$\forall i, \mathbf{w}_i^{t+1} = \arg \min_{\mathbf{w}} f_i(\mathbf{w}) + \frac{\rho\gamma}{2(\rho+\gamma)} \|\mathbf{w} - \mathbf{z}^t + \mu_i^{t+1}\|^2 \quad (9)$$

$$\forall i, \mathbf{q}_i^{t+1} = \frac{\rho}{\rho+\gamma} \mathbf{z}^t + \frac{\gamma}{\rho+\gamma} \mathbf{w}_i^{t+1} - \frac{\rho}{\rho+\gamma} \mu_i^{t+1} \quad (10)$$

$$\mathbf{z}^{t+1} = \frac{\sum_{i=1}^M (\mathbf{q}_i^{t+1} + \mu_i^{t+1})}{M + \frac{1}{\rho}} \quad (11)$$

From Equation 9, even when γ is huge, the regularization is upper bounded by $\frac{\rho}{2}$ that avoids obtaining a $\mathbf{w}$ over-influenced by $\mathbf{z}$, and leads to faster convergence. Note that we update μ_i first, and update $\mathbf{z}$ at the end of the iteration for a simpler derivation.

In synchronous ADMM, Equations [8,9,10] are performed in each worker node, while Equation 11 is performed in the master node. The master node waits for all sub-problems to be solved and their local models submitted, before merging and broadcasting a new consensus model, see Figure 1. Waiting for all nodes to finish slows down the distributed computation process, and each user benefits from the latest consensus less frequently. The asynchronous ADMM essentially unblocks the master and allows the consensus model to be updated freely. Therefore, the aggregated knowledge can be shared between individuals more frequently.

2.2 Versioned Synchronous ADMM

Different users have different numbers of samples to optimize, and different workers have different hardware latency. The submitted local models $\mathbf{q}_i$, and dual variables μ_i come to the master out of order. It is easy to see that $\mathbf{z}$ can be updated whenever the master is ready as long as it is broadcast at the end of the iteration. On the other hand, we can broadcast $\mathbf{z}$ to the workers at any time, and force the remaining workers to restart the sub-problem based on the new consensus $\mathbf{z}$, i.e., at any moment, there is only one version of consensus. Theorem 2 proves that this *Single-Versioned ADMM* converges at the same rate as the original Synchronous ADMM. The convergence analysis for Synchronous ADMM can be found in literature [[16],[22]].

Figure 1: Synchronous ADMM updating schedule. Although all personal models proceed in parallel, the master has to wait for all updates come to merge. This induces a great overhead.

Figure 2: A Single-Versioned ADMM updating schedule. The master updates the consensus periodically to allow faster sharing, but it adds additional barriers that potentially stalls the process more.

THEOREM 2. Let $\Theta_i = (\mathbf{w}_i, \mathbf{q}_i, \boldsymbol{\mu}_i)$, Lagrangian in Equation 4 can be rewritten as $\mathcal{L}(\{\Theta_i\}, \mathbf{z})$. Assuming $\forall i$, f_i is a convex function, at any point, update $\mathbf{z}_k^{t+1} = \arg \min_{\mathbf{z}} \mathcal{L}(\{\Theta_i^{t+1}\} \cup \{\Theta_i^t\}, \mathbf{z})$, then any optimization sequence, i.e., $\Theta_1 \dots \Theta_i^{t+1}, \mathbf{z}_k^{t+1}, \Theta_{i+1}^{t+1} \dots \Theta_M^T$, converges at rate $\mathcal{O}(\frac{1}{T})$ on expectation.

PROOF. For any iteration t , Lagrangian $\mathcal{L}^t(\{\Theta_i^t\}, \mathbf{z}_k^t)$ after k intermediate consensus updates, we can verify that

$$\mathbb{E} [\mathcal{L}(\dots, \mathbf{w}_i^t, \mathbf{q}_i^t, \mathbf{z}_k^t, \boldsymbol{\mu}_i^{t+1})] \leq \mathbb{E} [\mathcal{L}(\dots, \mathbf{w}_i^t, \mathbf{q}_i^t, \boldsymbol{\mu}_i^{t+1})].$$

On the other hand, on expectation, an optimization step $\mathbf{w}_i^t, \mathbf{q}_i^t, \mathbf{z}_k^t, \boldsymbol{\mu}_i^{t+1}$ is equivalent to $\boldsymbol{\mu}_i^t, \mathbf{w}_i^t, \mathbf{q}_i^t$, i.e.,

$$\mathbb{E} [\mathcal{L}(\dots, \boldsymbol{\mu}_i^t, \mathbf{w}_i^t, \mathbf{q}_i^t)] \leq \mathbb{E} [\mathcal{L}(\dots, \mathbf{w}_i^t, \mathbf{q}_i^t, \mathbf{z}_k^t, \boldsymbol{\mu}_i^{t+1})].$$

Combining these two, the optimization sequence for the *Single-Versioned ADMM* is dominated by the sequence of the original *Synchronous ADMM*. If f_i is convex, both sequences arrive at a saddle point. $\square$

If the master and individual workers optimize independently, it is possibly that users' personal models branch out on different versions of the consensus, and request to merge at different pace. These **delayed** updates do not necessarily dominate the synchronous ADMM optimization. However, we can show that they dominate an optimization sequence for a *multiple block ADMM* formulation [13]. The technique is to create the auxiliary variables for these different versions of consensus models. Assuming for the k -th merge, we have a set of users $B_k \subset [1 \dots M]$ submitted local

Figure 3: A Multi-Versioned ADMM updating schedule. Similar to Single-Versioned ADMM, this allows the consensus periodically updated, but no additional barriers are added.

updates. For each one of them $j \in B_k$, the sub-problem is solved by using the consensus model $\tilde{\mathbf{z}}_{h(j)}$, where $h(j) \in [1 \dots k-1]$ is the mapping from the user to the consensus model version. We first update $\tilde{\mathbf{z}}_{h(j)}$ for all $j \in B_k$, then update $\mathbf{z}_k$, and broadcast it. The consensus updating rules for this *multi-versioned ADMM* are defined in Equations 12 and 13.

$$\tilde{\mathbf{z}}_{h(j)}^{t+1} = \tilde{\mathbf{z}}_{h(j)}^t + \frac{1}{M + \frac{1}{\rho}} (\mathbf{q}_j^t + \boldsymbol{\mu}_j^t - \mathbf{q}_j^{t-1} - \boldsymbol{\mu}_j^{t-1}) \quad (12)$$

$$\tilde{\mathbf{z}}_k^{t+1} = \frac{1}{|B_k|} \sum_{j \in B_k} \tilde{\mathbf{z}}_{h(j)}^{t+1} \quad (13)$$

REMARK 3. When $\forall j, k, h(j) = k-1$, multi-versioned ADMM reduces to single-versioned ADMM.

To prove that *multi-versioned ADMM* converges, we first show that a Multi-Block ADMM problem updates using rules in Equations 12 and 13. Consider the optimization problem in Equation 14, where K is the total number of consensus updates. In this problem, we have multiple blocks, $\{\mathbf{q}_i, \mathbf{w}_i\}$, $\tilde{\mathbf{z}}_k$ and $\mathbf{z}$. According to [8], extending the ADMM algorithm directly by alternating optimizations over these blocks achieves the same linear convergence when the equality constraints satisfy the condition in Theorem 4.

$$\begin{aligned} \min_{\mathbf{z}, \{\mathbf{q}_i, \mathbf{w}_i\}, \{\tilde{\mathbf{z}}_k\}} \quad & \sum_{i=1}^M f_i(\mathbf{w}_i) + \frac{\gamma}{2} \|\mathbf{w}_i - \mathbf{q}_i\|^2 + \frac{1}{2} \|\mathbf{z}\|^2 \quad (14) \\ \text{subject to} \quad & \mathbf{q}_i = \tilde{\mathbf{z}}_k, \forall i \in B_k \\ & \tilde{\mathbf{z}}_{h(i)} = \tilde{\mathbf{z}}_k, \forall i \in B_k \\ & \tilde{\mathbf{z}}_K = \mathbf{z} \end{aligned}$$

THEOREM 4. [8] Assuming a constraint optimization has constraint $A_1 x_1 + A_2 x_2 + A_3 x_3 = 0$, if $A_1^T A_2 = 0$ or $A_2^T A_3 = 0$, a direct extension of ADMM converges in $\mathcal{O}(\frac{1}{T})$.

In our case, it is easy to see that A_k for $\tilde{\mathbf{z}}_k$ is canceling each other, and therefore $A_k = 0$. The dual variables for constraints $\tilde{\mathbf{z}}_k = \mathbf{z}$ cancel out, so Equation [12,13] are equivalent to the ADMM updating rules for $\tilde{\mathbf{z}}_k$ and $\mathbf{z}$. Therefore, for any such consensus updating schedules, we can construct a Multi-Block ADMM program that updates correspondingly. The only difference is that Multi-Versioned ADMM also updates $\mathbf{z}$ immediately after $\tilde{\mathbf{z}}$ is updated. Similar to Theorem 2, optimizing $\mathbf{z}$ multiple times within the same iteration dominates the original optimization sequence, so we conclude convergence as stated in Theorem 5.

Figure 4: Asynchronous ADMM updating schedule. This algorithm removes the iteration barriers and free up the consensus updates completely.

THEOREM 5. For any optimization sequence that follows Equation [12,13], $\Theta_1^1, \dots, \Theta_i^{t+1}, \bar{z}_{h(j)}^{t+1}, \mathbf{z}_k^{t+1}, \mu_{h(j)}^{t+1}, \Theta_{i+1}^{t+1} \dots \Theta_M^T$, it converges at rate $\mathcal{O}(\frac{1}{T})$.

2.3 Asynchronous ADMM

For the algorithms described in previous section, the last synchronous property holds that no one proceeds until the end of the current iteration. The master keeps track of the versions of the consensus models, which are the only dependencies to update the new version of the consensus. We can simply remove this iteration lock, and let the whole process runs. This results in the proposed Asynchronous ADMM algorithm as shown in Algorithm 1.

K is the latest version number;

input : $\{\Delta_j, h(j) | j \in B\}$

for $j \in B$ **do**

$$\mathbf{z}_{h(j)} = \mathbf{z}_{h(j)} + \frac{\Delta_j}{M+1};$$

end

$K = K + 1;$

$\mathbf{z}_K = \frac{1}{B} \sum_{j \in B} \mathbf{z}_{h(j)};$

Broadcast ($\mathbf{z}_K, K$);

Algorithm 1: Master for ASYNC-ADMM

$\mathbf{z}_k$ is the current consensus;

input : $\mathbf{x}, \mathbf{y}, i$

Add $\mathbf{x}, \mathbf{y}$ to samples for user i ;

Update $\mu_i, \mathbf{q}_i, \mathbf{w}_i$ according to Equations [8,10,9];

Let μ_i^o and $\mathbf{q}_i^o$ be the original values;

$$\Delta_i = \mu_i + \mathbf{q}_i - \mu_i^o - \mathbf{q}_i^o;$$

Reduce (Δ_i, k)

Algorithm 2: Worker for ASYNC-ADMM

By assuming that networking latency has an identical distribution with mean τ . We can show that this AsynchADMM converges at a similar rate as synchronous ADMM. Note that the concept of iterations does not exist any more. Here, T refers to the number of operations from the master, and is about M times the number of iterations in Synchronous setting. Therefore, the convergence rate is comparable for both of them.

THEOREM 6. With assumptions 1, Asynchronous ADMM Algorithms [1,2] converge at $\mathcal{O}(\frac{M\gamma\tau}{T})$

One interesting fact about the Theorem 6 is that the personalization strength γ influences the convergence too. Intuitively, if the personalized models are independent from the consensus model,

Figure 6: Facial expression of happy emotion on different people's faces.

i.e., $\gamma = 0$, the local model $\mathbf{q}_i$ can be simply assigned to the consensus $\mathbf{z}$, and the iteration stops. On the other hand, if different persons have very different personal preference, enforcing them to reach the consensus is very difficult. Allowing the models to be diverged effectively improves convergence speed.

3. FACIAL EXPRESSION

Human emotion study is a critical topic in several fields, including human-computer interaction, psychology research, medical diagnosis and treatment, entertainment, education, etc. Facial expression is one of the most useful non-verbal information for detecting human emotion. For decades, people have already investigated in automatic facial expression analysis through image, video, or audio [20]. As the camera becomes prevalent in modern devices, like mobile phone, laptop, desktop, and even wearable device, automatically recognizing the facial expression from image or video has gained much attention. Great amount of research [4] and industry products^{1 2} are proposed to solve this problem. The state of the art techniques [4] includes three major steps as shown in figure 5. First of all, human face is detected from the video frame, and the landmark points are located, such as eyes, nose, and mouth. After that, features are extracted from the face image. Popular features are optical flow, Gabor filter, gradient feature, local binary patterns, motion signature, geometry displacement, etc. These features can effectively represent the characteristics of different expressions. Given the feature set extracted from the collected data, classifiers have been trained with various machine learning models such as *linear classifier*, *neural networks*, *hidden Markov model*, *random forests*. However, due to different personalities, cultures, ages, and appearances, the facial expressions of the same emotion vary greatly. For instance, in figure 6, three people performs happy face but with different visual appearance. Hence, using a single model is not sufficient enough to represent all the users. Therefore, personalization will be the key to enhance the accuracy and improve the user experience.

To support this, we consider the data set collected for the facial expression recognition app developed by Microsoft in Xbox One Kinect³. The following experiments are conducted based on a subset of this data set. The subset includes 151,413 samples collected from 250 users. Conventionally, one model is trained for a classifier of a facial expression. In this way, the individual accuracy is sacrificed for obtaining fair average accuracy. With our Asynchronous ADMM framework, we can train a personalized model for each user which can better represent his/her feature distribution. We follow the state-of-the-art techniques in face and landmarks detection

¹FaceReader:<http://www.noldus.com/human-behavior-research/products/facereader>

²Kaggle Challenge:<https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge>

³<http://www.shacknews.com/article/79301/xbox-one-kinect-reading-emotions-and-heart-rate>

Figure 5: Flow chart of facial expression recognition.

Figure 7: ROC curves of personalized model and baseline (consensus) model in Facial expression data set. γ is set as 50 in this case.

and feature extraction. The features we extract are popular features used in computer vision domain, including, spatial filter responses, temporal filter responses, statistical values, and geometry values.

In addition to the accuracy improvement, our framework reduces the privacy concern of the users. People are usually unwilling to share their privacy data to others, especially this kind of visual data from which the subject’s identity can be easily revealed. In the existing framework, in order to train a facial expression classifier, the personal data must be uploaded to somewhere hence loses the privacy. In our distributed framework, training can be performed locally, such as in a smart device or private cluster. The personal data does not need to be uploaded to other places. Although the personal model is still exposed to others, the privacy of the data itself is preserved which largely mitigates the concerns of the privacy.

3.1 Influence of the Personalization

First, we compare the difference between the results with personalization and without personalization in accuracy. For each user, we use 5 positive samples and 5 negative samples for training. Figure 7 shows the ROC curve of the overall accuracy among all the users. When the personalization is considered, the accuracy is better than the one without personalization (baseline model). The baseline model is obtained based on conventional ADMM where all the users train a single consensus model. This result is not surprising since one model does not fit all the users. This result also tells during the real world application, the personalized models would provide better satisfaction to the users than the consensus model.

As we discussed in the previous section, the strength of personalization plays the key role for the generalization ability. To demon-

strate such a behavior, we study the accuracies in different γ settings. The larger the γ is, the more the personal model is towards the consensus model. We show the distribution of the ROC curves across different users. We compute the ROC curves for each individual user, and sample the data points along the curve to compute the mean and the variance. Figure 8 represents a mean ROC curve and the standard deviation interval along the curve to show how consistent the model performs across the population. From the figures, we demonstrate that $\gamma = 50$ (chosen by a validation set) produces the best prediction at the testing set (with better mean ROC curve and narrower variance range), while $\gamma = 10^{-6}$ and $\gamma = 10^6$ produce inferior results. At one extreme side, $\gamma = 10^{-6}$ corresponds to a model trained completely by using personal data. The overfitted model introduces high variance. On the other extreme side, $\gamma \rightarrow \infty$, everyone is using nearly the same model which is too generalized to represent different users’ distribution. By tuning the γ , we find the balance between generalization and personalization. In the future, our vision is γ value can be an user-controlled setting. Users can freely select how personalized their own models should be. Our general recommendation is: when the user has only few data, it is always intuitive to have the consensus model to be the *cold start* model. After the user accumulates the personal data, the model can be relaxed a little bit to support divergence. When the user has enough personal data, the model can be personalized completely, and the consensus model has less and less influence.

3.2 Learning Curves

One of the benefits of the distributed personalization is to leverage community knowledge to improve individual user’s learning curve. In this experiment, we vary the number of training samples to show the progress of the accuracy improvement. Figure 9 and 10 show the testing ROC curves using personalized models trained on different amounts of training samples. The training samples are set as 1 sample per user, 3 samples per user, 5 samples per user, and 10% of an user’s data for each curve, respectively. The corresponding area under curve (AUC) is shown in Table 1. From 1 sample to 3 samples, the learning speed is really fast that increases the accuracy a lot which demonstrates that our framework enables the users to share the knowledge with each other efficiently. On the contrary, if we use consensus model, the improvement of the accuracy is slow compare to the increase of training samples. It is true that the consensus model performs better than the personalized model when using only 1 training sample per user. It means 1 sample of the personal data is not sufficient enough to represent the intra-user variance for each user. With more training data, it covers the variation within each user, i.e., it adapts individual user well, the accuracy of personalized models starts to outperform the consensus model.

This experiment demonstrates that the learning speed of personalized model is fast as the size of training set increases, i.e., users only need few samples to improve their personal models. This is relatively important when we consider the applications in

Figure 8: The distributions of the ROC curves of using personalized models for all the users. For each user, we use 5 positive samples and 5 negative samples for training. Each subfigure corresponds to different personalization strengths ($\gamma = 10^{-6}, 10^{-3}, 1, 50, 10^3, 10^6$). The accuracy is the highest when $\gamma = 50$ and drops when γ is too large or small.

commercial products. In most of the existing personalization products/services, e.g., ads, recommendation system, users are usually asked to provide some personal information, answer questions, or do data collection and tagging. However, users sometimes do not have patience to provide so much information. With our framework, even though users only provide little amount of data, the system can make the models adapt quickly. It encourages the users to provide more data since they can feel the improvement of the accuracy. With more data, the models fit individuals better, and the whole community is also benefited from the ability of knowledge sharing in our framework.

Figure 9: ROC curves of the personalized models trained on different number of samples. $\gamma = 1$ in this experiment.

Figure 10: ROC curves of the baseline (Consensus) models trained on different number of samples. $\gamma = 1$ in this experiment.

4. JOBS YOU MIGHT BE INTERESTED IN (JYMBII)

One of the premium member value propositions of LinkedIn is the jobs recommendation product, Jobs You May Be Interested In (JYMBII), where LinkedIn members can impress active job postings relevant to their professional profiles. It is one of most valuable

Table 1: Area under curve in figures[9,10]

Model	1 sample	3 samples	5 samples	10% of data
Personalized	0.7771	0.8812	0.9084	0.936
Baseline	0.8277	0.8373	0.8383	0.8623

product contributing to member engagement on LinkedIn. Members can consume JYMBII recommendations in different channels, such as LinkedIn homepage feeds, Email, and LinkedIn Jobs Home (i.e. the 'jobs' tab on LinkedIn home page).

As many members maintain an up-to-date professional profile on LinkedIn, it is very important that JYMBII can recommend highly relevant and personalized jobs for members, whether they are active job seekers or passive ones who just want gain some awareness of the job market. Without personalization, there are circumstances that a bad job recommendation can significantly impair user's satisfaction, for example, an executive in a large-size company may feel offended by seeing entry-level job recommendation, an experienced professional may feel uncomfortable receiving intern-level job recommendations. Even though current recommendation engine has already taken into consideration of members' personal career characteristics, such as their professional skills, current job position, and their next job preference (either explicit or implicit), the model coefficients for these features are global as we train a single model for all of our members, which inevitably return less satisfactory recommendations for certain members. The Asynchronous ADMM model we propose, on the other hand, will train a personalized model for each member with sufficient job seeking history, and still maintain a global consensus model for those without enough historical record. In doing so, it can provide additional personalization in a way such that it greatly improve user satisfaction for member's personal job seeking experience.

In order to compare personalization ADMM model against current baseline consensus model trained with synchronized ADMM setting, we have trained our personalization ADMM model with different gammas, which controls the strength of personalization, with larger gamma indicating more personalization. Each example is a (member, job, label) tuple, with label being either 1 or 0 indicating whether or not the member has applied the job. The feature set is the same for baseline and treatment, it mainly includes field based content matching features and member preference features. Each content matching feature is a cosine similarity score between the two fields' tf.idf vectors, with one field from member profile and the other field from job posting, for example, member's experience field matching against job's description field, member's profile summary field matching against job's desired skills field, etc. 11 illustrates the field layout of a (member.job) pair on LinkedIn. The member preference features include job location preference, industry preference, and company size preference. They can be members' explicit preference they set on the web UI, or their latent preference mined by a model [24]. The final model includes one hundred of such features for each example, and the dataset includes about a million samples. Figure 12 illustrates the ROC curves and their AUCs for model comparison.

The personalized model has the best AUC of 0.91 with gamma = 100, comparing to AUC of 0.79 for the most personalization model (with gamma = 1), and AUC of 0.86 for the least personalization model (with gamma = 10000). It suggests that neither personalization nor extreme personalization performs the best, but the optimal strength for personalization is somewhere falls in between.

Lijun (Leo) Tang
Experienced machine learning practitioner
San Francisco Bay Area | Internet

Current

LinkedIn

Previous

Microsoft, Amazon.com, IBM

Education

Columbia University in the City of New York

Send a message

www.linkedin.com/in/lytang

Background

Summary

Machine Learning Researcher/Software Engineer in online industry, IEEE Senior Member.
Specialties: machine learning, software design, video processing, information retrieval

Experience

Senior Research Engineer
LinkedIn
June 2014 – Present (3 months) | Mountain View, California
Working on relevance and recommendation for talent solutions, on the world's largest professional network.

1 project

Architect - Relevance Infrastructure
LinkedIn - San Francisco Bay Area
Posted 9 days ago

Other Details

[View full job listing](#) This is a preview of the Architect - Relevance Infrastructure job at LinkedIn. To view the full job listing, join LinkedIn - it's free!

About this job

Job description

LinkedIn's Relevance Team is responsible for engineering the various software platforms used to connect our 300 million members with the most relevant jobs, content and advertisements. We use the state of the art big data and machine learning technologies, such as ADMM, statistical methods, convex, linear and quadratic programming, Hadoop, Pig, and NoSQL. This team is looking for a Technical Lead/Architect to lead the overall technical direction of our relevance infrastructure platforms. In addition to setting the strategy, you will be responsible for designing the next generation of highly-scalable machine learned systems for recommendations, response prediction, and content matching, and leading and mentoring other software engineers in the implementation of these products.

Desired Skills and Experience

- 10+ years of experience in both individual contributor and technical lead/architect/engineering manager roles
- Exceptional background and knowledge in building large scale distributed systems in a real-time, production serving environment
- Strong background and experience with working on machine learning algorithms and systems, with an emphasis on statistical learning/models
- Significant experience in an engineering leadership role as both a system designer and mentor for other engineers
- Significant experience with solving "big data" problems using clustered computing systems (e.g. Hadoop, Cosmos)
- Significant experience with building both models and systems for recommendation, response prediction (e.g. click prediction), and/or optimization problems
- Familiarity with search engine/information retrieval technology and techniques
- Experience and ability to work with peers in engineering and product management across teams to set technical and product direction
- M.S. (or equiv.) in Computer Science, Ph.D. preferred

Figure 11: A Member Profile (upper) and A Job Posting (lower)

Figure 12: ROC of JYMBII baseline model and personalization models

Comparing to the baseline without personalization ($AUC=0.86$), it is interesting to see all of the personalization models performs much better in the operation area with low false positive rates (e.g. false positive rate less than 0.1), this area is critical as it aligns better with the online system which observes much more negative examples than positive examples, thus it is a necessity to operate with a low false positive rate all the time.

5. CONVERGENCE EXPERIMENT

Due to the asynchronous nature of Algorithm 1&2, it is difficult to implement such algorithms in a typical Map/Reduce architecture [11], which requires a synchronization step between each map/reduce stage. We develop an asynchronous computation framework based on the Publish/Subscribe message paradigm [14] to facilitate the parallel computation interfaces. The framework consists of a master node and multiple worker nodes. Each worker holds a subset of users and their data, initializes a personal model and subscribes a message channel for each user. The training events are triggered through user interactions. The corresponding worker fetches the current best-known consensus model after receiving a training event, performs a local optimization and sends the updated personal model back to the master. The consensus model then gets updated at the reception of each updating event. In synchronous ADMM (SYNC-ADMM), all personal models start training at the same time using the same consensus model, the next round is triggered after all personal models are merged into the consensus. For multi-versioned ADMM (MV-ADMM), all personal models start training at approximately the same time using their current known consensus model. The consensus model does not block any training event in any single round and merges continuously, but it waits for the completion of all user models before triggering another round. In asynchronous ADMM (ASYN-ADMM), each user training event only depends on the completion of the user's previous round. The consensus model merges any reported personal model immediately. During the entire experiment process, the total number of training events triggered is more than 10 million. In Figure 13, for each ADMM model, we divide the

Figure 13: Convergence rate of different ADMM models. Note that the objective function is in log scale.

time window into 300 buckets and only report the average objective function value observed within each time bucket. We choose the starting time when all user models are trained at least once. The flat blank periods within SYNC-ADMM and MV-ADMM models are the blocking periods when they wait for all personal models being merged into the consensus. As demonstrated in Figure 13, ASYNC-ADMM converges at a close-to-identical rate compared to other models, but performs 2~4 times faster by eliminating all blocking period.

6. CONCLUSION AND DISCUSSION

In this paper, we propose a distributed personalization framework and present its successful application to two industry problems. Our approach enables distributed personalization not only for traditional recommendation problems, but also for recognition. Provided each individual task can be formulated as a convex optimization, our framework can scale, for both training and personalization, to millions of simultaneous users. One of the major contribution that enables this distributed learning system is a novel *Asynchronous ADMM* algorithm that removes all network I/O wait barriers to maximize the CPU throughput. One of the interesting features of our publish/subscribe based distributed computation approach allows worker nodes to execute on personal devices. Unlike other popular distributed learning algorithms, we do not rely on a decaying step size to control convergence. Instead, Asynchronous ADMM keeps track of versions of the consensus model, thereby avoiding complicated learning rate adjustments. In the future, we plan to extend our approach to the real online setting. Problems like warm-start, online model adaptation, and in-session personalization will be studied in-depth. Furthermore, we currently employ only one master to update the consensus. This wastes some network bandwidth. Much research has suggested that a *Minimum Spanning Tree* topology can provide optimal network communi-

cation to efficiently leverage computational resources — a natural extension to the proposed algorithm.

References

- [1] A. Agarwal, O. Chapelle, M. Dudík, and J. Langford. A reliable effective terascale linear learning system. *J. Mach. Learn. Res.*, 15(1):1111–1133, Jan. 2014.
- [2] A. Agarwal and J. C. Duchi. Distributed delayed stochastic optimization. In *Proceedings of the 51th IEEE Conference on Decision and Control, CDC 2012, December 10-13, 2012, Maui, HI, USA*, pages 5451–5452, 2012.
- [3] S. Ben-David, J. Gehrke, and R. Schuller. A theoretical framework for learning from a pool of disparate data sources. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '02*, pages 443–449, New York, NY, USA, 2002. ACM.
- [4] V. Bettadapura. Face expression recognition and analysis: the state of the art. *arXiv preprint arXiv:1203.6722*, 2012.
- [5] M. Bilenko and M. Richardson. Predictive client-side profiles for personalized advertising. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '11*, pages 413–421, New York, NY, USA, 2011. ACM.
- [6] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122, Jan. 2011.
- [7] E. Candès and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772, 2009.
- [8] C. Chen, B. He, Y. Ye, and X. Yuan. The direct extension of admm for multi-block convex minimization problems is not necessarily convergent. *Mathematical Programming*, pages 1–23, 2014.
- [9] Y. Chen, D. Pavlov, and J. F. Canny. Large-scale behavioral targeting. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '09*, pages 209–218, New York, NY, USA, 2009. ACM.
- [10] A. Cotter, O. Shamir, N. Srebro, and K. Sridharan. Better mini-batch algorithms via accelerated gradient methods. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 1647–1655. Curran Associates, Inc., 2011.
- [11] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, Jan. 2008.
- [12] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao. Optimal distributed online prediction using mini-batches. *J. Mach. Learn. Res.*, 13:165–202, Jan. 2012.
- [13] W. Deng, M.-J. Lai, Z. Peng, and W. Yin. Parallel Multi-Block ADMM with $\mathcal{O}(1/k)$ Convergence. *ArXiv e-prints*, Dec. 2013.

- [14] G. Eisenhauer, K. Schwan, and F. E. Bustamante. Publish-subscribe for high-performance computing. *Internet Computing, IEEE*, 10(1):40–47, 2006.
- [15] T. Evgeniou and M. Pontil. Regularized multi-task learning. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '04*, pages 109–117, New York, NY, USA, 2004. ACM.
- [16] M. Hong and Z.-Q. Luo. On the Linear Convergence of the Alternating Direction Method of Multipliers. *ArXiv e-prints*, Aug. 2012.
- [17] Y. Koren. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '08*, pages 426–434, New York, NY, USA, 2008. ACM.
- [18] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, Aug. 2009.
- [19] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *In NIPS*, pages 556–562. MIT Press, 2000.
- [20] M. Pantic and L. J. M. Rothkrantz. Automatic analysis of facial expressions: The state of the art. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(12):1424–1445, 2000.
- [21] C. Perlich, B. Dalessandro, T. Raeder, O. Stitelman, and F. Provost. Machine learning for targeted display advertising: Transfer learning in action. *Mach. Learn.*, 95(1):103–127, Apr. 2014.
- [22] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin. On the linear convergence of the admm in decentralized consensus optimization. *Signal Processing, IEEE Transactions on*, 62:1750–1761, February 2014.
- [23] L. Veer and R. Bernards. Enabling personalized cancer medicine through analysis of gene-expression patterns. *Nature*, 452:564–570, 2008.
- [24] J. Wang and D. Hardtke. User latent preference model for better downside management in recommender systems. In *International World Wide Web Conference, WWW '15*, Florence, Italy, May 2015.
- [25] C. Zhang, H. Lee, and K. G. Shin. Efficient distributed linear classification algorithms via the alternating direction method of multipliers. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2012, La Palma, Canary Islands, April 21-23, 2012*, pages 1398–1406, 2012.