Distributed Rating Prediction in User Generated Content Streams

Sibren Isaacman*, Stratis Ioannidis†, Augustin Chaintreau‡, and Margaret Martonosi*
*Princeton University, †Technicolor, ‡Columbia University
isaacman@princeton.edu, stratis.ioannidis@technicolor.com,
augustin@cs.columbia.edu, mrm@princeton.edu

ABSTRACT

Recommender systems predict user preferences based on a range of available information. For systems in which users generate streams of content (*e.g.*, blogs, periodically-updated newsfeeds), users may rate the produced content that they read, and be given accurate predictions about future content they are most likely to prefer. We design a distributed mechanism for predicting user ratings that avoids the disclosure of information to a centralized authority or an untrusted third party: users disclose the rating they give to certain content only to the user that produced this content.

We demonstrate how rating prediction in this context can be formulated as a matrix factorization problem. Using this intuition, we propose a distributed gradient descent algorithm for its solution that abides with the above restriction on how information is exchanged between users. We formally analyse the convergence properties of this algorithm, showing that it reduces a weighted root mean square error of the accuracy of predictions. Although our algorithm may be used many different ways, we evaluate it on the Neflix data set and prediction problem as a benchmark. In addition to the improved privacy properties that stem from its distributed nature, our algorithm is competitive with current centralized solutions. Finally, we demonstrate the algorithm's fast convergence in practice by conducting an online experiment with a prototype user-generated content exchange system implemented as a Facebook application.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems

General Terms

Algorithms

1. INTRODUCTION

A considerable portion of web activity today can be attributed to direct interactions between online users. Blogs, social networks such as Facebook, micro-blogging applications such as Twitter, and video sharing sites such as YouTube have provided online platforms through which users author and share original content, establishing an online following that often overlaps with their real-life social circles.

The sheer volume of content generated daily in the blogosphere and on social networks makes identifying relevant and interesting content a challenging task. At present, providers of these services have deployed rating mechanisms though which a user can give feedback on content generated by other users. Facebook and Twitter have expanded their rating mechanisms to the blogosphere, competing with other traditional aggregators such as Digg, Reddit and StumbleUpon that offer similar rating interfaces. Access to such ratings allow such companies to improve their recommendations but also to profile users; such profiles are a resource that companies monetize, *e.g.*, through advertising.

On the other hand, the increased monetization of private data has been met by a sharp rise in privacy concerns within advocacy groups like the Electronic Frontier Foundation and regulatory bodies like the US Congress [2]. Privacy in general, and online privacy in particular, is recognized as a fundamental human right by laws such as the European Directive on Protection of Personal Data and the Electronic Communications Privacy Act in the U.S. Nevertheless, there are many reasons why online users readily disclose private information to the above companies. Behavioral economists have identified bounded rationality, immediate gratification, underestimating risk [1] and the paradox of control [4] as some of them. Nevertheless, the importance of respecting the privacy of online users has been recognized not only by the authorities and non-profit organizations but by companies as well. For example, Google and Microsoft have struggled to find compromises that respect the privacy of their users without significantly undermining their profits [23, 25].

The challenge thus arising from this state of affairs is enabling users to view and access relevant, interesting, user-generated content without releasing their private information to untrusted third parties. In this paper, we propose a solution to this problem by designing a distributed rating prediction mechanism that allows users to restrict information sharing only among trusted parties.

More specifically, we consider a general system in which users generate and share streams of content items. For example, a content producer in such a system may maintain a blog, a Facebook wall or a Twitter feed. Updates generated by users (*i.e.*, new blog posts, wall entries or tweets) are shared with select subscribers, who may respond to received content by rating it. Note that, at present, such systems are centralized. However, this need not be the case in the sys-

RecSys'11, October 23–27, 2011, Chicago, Illinois, USA.

tem we consider here: users may generate and share both their content as well as their ratings with each other in a peer-to-peer fashion.

Our goal is to design a fully distributed mechanism that learns and predicts the ratings of content consumers—*i.e.*, the subscribers. More specifically, we would like to devise a collaborative filtering scheme operating under the constraint that information is shared only between (a) a content producer and (b) its subscribers. For example, a user rating certain content should share this rating *only* with the user that generated the content.

Again, the main reason motivating our focus on distributed rating prediction is privacy: we wish to avoid the disclosure of ratings or, in fact, any user information (such as profiles), to a central authority or an untrusted third party. Another reason is scalability, as a distributed approach can scale as the number of content producers and consumers increases. Furthermore, since ratings are exchanged among content producer/consumer pairs, our algorithm naturally exploits the social relationships between them: if two friends subscribe to each other's feed, training our predictor based on content exchanges between them can exploit latent similarities between their interests.

Our contributions can be summarized as follows:

- We propose a mathematical model of a system for distributed sharing of user-generated content streams. Our model captures a variety of different applications, and incorporates correlations both on how producers deliver content and how consumers rate it.
- We illustrate that estimating the *probability distribution* of content ratings can be naturally expressed as a Matrix Factorization (MF) problem. This is in contrast to standard MF formulations that focus on estimating ratings directly, rather than their distribution. To the best of our knowledge, our work is the first to apply a MF technique in the context of rating prediction in user-generated content streams.
- Using the above intuition, we propose a decentralized rating prediction algorithm in which information is exchanged only across content producer/consumer pairs. Producers and consumers maintain their own individual profiles; a producer shares its profile *only* with consumers to which it delivers content, and consumers share a rating they give to an item, as well as their profile, *only* with the producer that generated it.
- In spite of the above restriction on how information is exchanged among users, our distributed prediction algorithm optimizes a global performance objective. In particular, we formally characterize the algorithm's convergence properties under our model, showing that it reduces a weighted mean square error of its rating distribution estimates.
- We validate our algorithm empirically. First, we use the Netflix data set as a benchmark to compare the performance of our distributed approach to offline centralized algorithms. Second, we developed a Facebook application that reproduces the main features of a peer-to-peer content exchange environment. Using a month-long experiment with 43 users we show that our algorithm predicts ratings accurately with limited user feedback.

The remainder of this paper is organized as follows. In Section 2 we briefly describe the relationship of our work to previous rating prediction mechanisms. In Section 3 we introduce our mathematical model, and in Section 4 we present our distributed prediction algorithm as well as our main results on its convergence. Our numerical evaluation over the Netflix data set and our case study using a FaceBook application are in Sections 5 and 6, respectively. Finally, we conclude in Section 7.

2. RELATED WORK

A traditional approach to distributed rating prediction in peer-to-peer networks involves computing a similarity metric (*e.g.*, Pearson correlation) among neighboring peers; predicted ratings are obtained as a function of the ratings in a peer's neighborhood, taking into account its similarity to its neighbors [8, 15, 18, 20, 24]. Neighbor selection is part of system design, as predictions improve when neighbors have high similarity scores. We depart considerably from these works by focusing on dynamic user generated content, rather than static content, as well as by providing formal performance guarantees on the prediction error.

Matrix factorization is popular among collaborative filtering methods due to its ability to scale over large datasets [3, 13, 22]. A simple method for obtaining a low-rank factorization of a rating matrix is gradient descent on the prediction RMSE, potentially with added regularization terms (see, *e.g.*, [22], and the references therein). Assuming that ratings follow the Gaussian distribution, minimizing the RMSE is equivalent to maximizing the likelihood of observed entries [21]. In this work, we do not rely on a Gaussian assumption; instead, we directly apply MF to the *probability distribution* of ratings in dynamic user generated streams. Moreover, our distributed algorithm behaves as gradient descent over a *weighted* RMSE metric, whose weights correspond to content delivery rates across different consumers.

Recent work on MF has demonstrated that if entries of a low-rank matrix are removed uniformly at random, the matrix can be reconstructed accurately with high probability [5, 6, 11]. These approaches yield stronger results than [22] as gradient descent does not always converge to the original matrix, or even a minimizer of the RMSE. Nevertheless, gradient descent methods reduce the RMSE even when entries are not removed uniformly at random—which is true for the system we study. Moreover, the algorithms in [5, 6, 11] are centralized and cannot be directly applied to the distributed setting we consider.

There are known distributed algorithms for principal component analysis (PCA) of the adjacency matrix of a graph permitting message exchanges only across adjacent nodes [10,12]. These relate to our work as PCA can be used to construct an optimal low-rank approximation of the adjacency matrix. We cannot apply such approaches as we *do not* factorize the random adjacency matrix restricting communication between nodes/users (matrix $A(k)$ in our model): we factorize the distribution of ratings, which are decoupled from the communication process. The above methods are also not fully distributed, as ortho-normalizing principal components requires broadcasting or gossiping normalization factors across all nodes.

Privacy in collaborative filtering was previously addressed using homomorphic encryption [7], randomized perturbation [19] and concordance measure [14] techniques. Our approach, in contrast, relies on trust between participants: we restrict information exchanges only between trusted content producer-consumer pairs. Within this context, secure multiparty computation is not required to predict ratings without disclosing user ratings publicly.

3. SYSTEM MODEL

In this section, we present the mathematical model that we use in our analysis.

3.1 Content Sharing

We consider a set $\mathcal{U}$ of users generating and sharing content in a peer-to-peer manner. A subset $\mathcal{N} \subseteq \mathcal{U}$ of all users, whom we call *producers*, generate a stream of items. For example, producers could maintain a blog, a news-feed or a twitter-feed. Time is divided into timeslots, and in each timeslot every producer generates a new content item (*i.e.*, a blog entry or a tweet) that is added to her locally-maintained feed; this is subsequently shared with other users in a set $\mathcal{M} \subseteq \mathcal{U}$, which we call *consumers*. Note that $\mathcal{M}$ and $\mathcal{N}$ may intersect, as a user may both produce and consume content.

Producers share items only with consumers that belong to their social circle and/or subscribe to their feed. Even so, items may fail to reach all such consumers, either because the producer shares them selectively or because the consumers fail to receive them (*e.g.*, because they do not observe the feed continuously).

We model this as follows. Let $a_{i,j}(k) \in \{0,1\}$ be a binary random variable indicating whether the item generated by $i \in \mathcal{N}$ at timeslot k is delivered to $j \in \mathcal{M}$, and let $A(k) = [a_{i,j}(k)]_{i\in\mathcal{N},j\in\mathcal{M}}$ be the corresponding $|\mathcal{N}|\times|\mathcal{M}|$ matrix. We make the following assumption:

ASSUMPTION 1. *$\{A(k)\}_{k\in\mathbb{N}}$ is an i.i.d. sequence.*

I.e., deliveries are independent and identically distributed across time. Let $\lambda_{i,j} = \mathbb{E}[a_{i,j}]$ be the probability that i delivers an item to j. If j does not subscribe to i's feed, then $\lambda_{i,j} = 0$. Different consumers may receive items from a feed with different probabilities; our model thus allows heterogeneity in how content is targeted by producers and how often consumers fail to observe it. Note that Assumption 1 *does not imply* that deliveries between different user pairs are independent. *E.g.*, $A(k)$ may be such that an item delivered to Alice is always also delivered to Bob.

3.2 Content Ratings

Whenever a producer i delivers content to a consumer j, the consumer provides some feedback to i in the form of a *rating*. We denote by $\mathcal{O}$ the set of possible ratings provided by consumers. In general, ratings are application-dependent. For example, consumers may indicate through an appropriate interface whether they liked, disliked or were neutral towards the content, so that $\mathcal{O} = \{+,-,\varnothing\}$. Consumers may also indicate their interest on a scale from 1 (lowest interest) to 5 (highest interest), *i.e.*, $\mathcal{O} = \{1,2,3,4,5\}$.

Let $\mathcal{I}_{i,j} = \{k \in \mathbb{N} : a_{i,j}(k) = 1\}$ be the set of timeslots at which i delivers content to j. W.l.o.g., for all $k \in \mathcal{I}_{i,j}$, j provides a rating to i within the duration of the timeslot k. Depending on the application, lack of feedback can be modelled either as a failed delivery ($a_{i,j}(k) = 0$) or an additional rating (element in $\mathcal{O}$). For $k \in \mathcal{I}_{i,j}$, let $r_{i,j}(k) \in \mathcal{O}$ be the the rating given by j to i's content. We assume that:

ASSUMPTION 2. *$\{r_{i,j}(k)\}_{k\in\mathcal{I}_{i,j}}$ is an i.i.d. sequence.*

Note that ratings *need not* be independent across users. For example, Alice and Charlie may always give the same rating to an item from Bob.

3.3 Rating Distributions

Denote by $\tilde{\pi}^o_{i,j}$, $o \in \mathcal{O}$, the probability that $r_{i,j}(k) = o$ for $k \in \mathcal{I}_{i,j}$. For every rating $o \in \mathcal{O}$, let

$$\tilde{\Pi}^o = [\tilde{\pi}^o_{i,j}]_{i\in\mathcal{N},j\in\mathcal{M}}. \quad (1)$$

This is a $|\mathcal{N}| \times |\mathcal{M}|$ matrix; each element corresponds to a producer/consumer pair i, j, and contains the probability that j gives rating o to an item from i. Our goal is to correctly estimate the probability matrices $\tilde{\Pi}^o$, for all $o \in \mathcal{O}$. *I.e.*, for any producer/consumer pair and any rating, we wish to find the probability that the consumer will react to content generated by the producer by providing this rating. Most importantly, we wish to do so in a distributed fashion, by restricting information exchanges only directly between producers and consumers.

Note that every time a consumer rates a content item the rating is in effect a sample from the distribution defined by the matrices $\tilde{\Pi}^o$, $o \in \mathcal{O}$. However, due to the heterogeneity of the process $A(k)$, samples are not obtained at the same rate. In fact, if $\lambda_{i,j} = 0$, the distribution of ratings of the (i, j) producer/consumer pair is *never* sampled; this relates the estimation of $\tilde{\Pi}^o$ to matrix completion, as certain entries of $\tilde{\Pi}^o$ are missing and cannot be directly observed.

Contrary to traditional matrix completion, missing entries of $\tilde{\Pi}^o$ are not selected uniformly at random—their absence is determined by, *e.g.*, the feeds to which a consumer subscribes. However, just as in traditional matrix completion, it is very natural to make the following assumption:

ASSUMPTION 3. *The probability matrices $\tilde{\Pi}^o$ are low-rank.*

This assumption implies that MF techniques can be applied to estimate these matrices; indeed our algorithm, presented in Section 4, exploits this relationship. Note that the low-rank property holds for the *rating distribution*, rather than the ratings themselves—as each producer generates an infinite number of items, this distinction is necessary.

Assumption 3 can be interpreted as a consequence of a generative latent factor model and the total probability theorem. As such, it is indeed very natural in the context of our system. We illustrate this below.

3.4 A Low-Rank Latent Factor Model

In this section we give an example of how the low-rank property of the matrices $\tilde{\Pi}^o$ may manifest by making additional assumptions on how users generate and rate content. This is only for the sake of illustration and to motivate our approach: our main results (Theorems 1 and 2) rely only on the assumptions we have made so far (in particular, Assumptions 1 and 2).

Suppose that content items generated by producers are grouped by similarity with respect to some features, thus forming a partition of the "content universe" into categories. For example, categories may pertain to topics (*e.g.*, news, music, sports, *etc.*). Formally, assume the existence of a set $\tilde{\mathcal{F}}$, whose elements we refer to as categories, such that every content item generated by a producer belongs to a single category. When $i \in \mathcal{N}$ generates new item, this item belongs to category $f \in \tilde{\mathcal{F}}$ with probability

$$\tilde{p}_{i,f} \geq 0, \quad \text{where } \textstyle\sum_{f\in\tilde{\mathcal{F}}} \tilde{p}_{i,f} = 1, \quad (2)$$

independently of any categories of items the producer has generated in the past. Moreover, when $j \in \mathcal{M}$ views an item in category $f \in \tilde{\mathcal{F}}$, j provides rating $o \in \mathcal{O}$ with probability

$$\tilde{q}^o_{j,f} \geq 0, \quad \text{where } \textstyle\sum_{o\in\mathcal{O}} \tilde{q}^o_{j,f} = 1, \quad (3)$$

independently of any ratings the user has given in the past.

Then, from the total probability theorem, the probability that j gives rating o when viewing content from i is:

$$\tilde{\pi}^o_{i,j} = \textstyle\sum_{f\in\tilde{\mathcal{F}}} \tilde{p}_{i,f}\tilde{q}^o_{j,f} = \langle \tilde{p}_i, \tilde{q}^o_j \rangle, \quad o \in \mathcal{O}, \quad (4)$$

or, in matrix form,

$$\tilde{\Pi}^o = \tilde{P} \cdot (\tilde{Q}^o)^T, \quad o \in \mathcal{O}, \tag{5}$$

where $\tilde{P} = [\tilde{p}_{i,f}]_{i\in\mathcal{N},f\in\mathcal{F}}$ and $\tilde{Q}^o = [\tilde{q}^o_{j,f}]_{j\in\mathcal{M},f\in\mathcal{F}}$. *I.e.*, the matrices $\tilde{\Pi}^o$, $o \in \mathcal{O}$, admit a $|\tilde{\mathcal{F}}|$-rank decomposition and, as such, their ranks are at most $|\tilde{\mathcal{F}}|$. Thus, *if the number of categories is small, the matrices* $\tilde{\Pi}^o$ *are low-rank*. Moreover, the l.h.s. matrix $\tilde{P}$ is the same in all $|\mathcal{O}|$ decompositions. The low-rank property is thus a consequence of the existence of content categories and the total probability theorem.

4. DISTRIBUTED RATING PREDICTION

The rating prediction problem in the context of this work is to correctly estimate the probability matrices $\tilde{\Pi}^o$, $o \in \mathcal{O}$, in a distributed fashion. Below, we first formulate this problem as a MF problem and then present our distributed gradient-descent mechanism for its solution.

4.1 Formulation as Matrix Factorization

To estimate $\tilde{\Pi}^o$ through MF, we construct low-dimensional *profiles* of producers and consumers. The inner product of two such profiles yields an estimate of the rating probabilities $\tilde{\pi}^o_{i,j}$. More specifically, let $d \in \mathbb{N}$ be a small integer such that $d \ll \min(|\mathcal{N}|, |\mathcal{M}|)$ and denote by $\mathcal{F}$ the set $\{1, 2, \ldots, d\}$. For now, we make no assumptions on how d relates to the rank of $\tilde{\Pi}^o$ (*i.e.*, the number of "categories" $|\tilde{\mathcal{F}}|$ in the example given by (5)).

Each producer $i \in \mathcal{N}$ maintains a vector $p_i \in [0, 1]^d$ that satisfies (2), which we call the *production profile* of i. Similarly, each consumer $j \in \mathcal{M}$ maintains $|\mathcal{O}|$ vectors $q^o_j \in \mathbb{R}^d_+$, one for every rating $o \in \mathcal{O}$, that satisfy (3). These $|\mathcal{O}|$ vectors constitute the *consumption profile* of j:

$$q_j = (q_j^{o_1}, q_j^{o_2}, \ldots, q_j^{o_{|\mathcal{O}|}}).$$

Given the above profiles, our estimate of the probability $\tilde{\pi}^o_{i,j}$ (the probability that when j views content generated by i it gives rating o)—is computed as follows:

$$\pi^o_{i,j} = \textstyle\sum_{f\in\mathcal{F}} p_{i,f} q^o_{j,f} = \langle p_i, q^o_j \rangle, \quad o \in \mathcal{O}. \tag{6}$$

Note the similarity between Equations (6) and (4). The constraints (2) and (3) immediately imply that the inner products in (6) are non-negative and $\sum_{o\in\mathcal{O}} \pi^o_{i,j} = 1$, *i.e.*, the latter indeed constitute a probability distribution.

Our goal is then to devise an algorithm for finding profiles p_i, q_j such that the prediction $\pi^o_{i,j}$ is as close to $\tilde{\pi}^o_{i,j}$ as possible. More formally, we wish to solve the following optimization problem:

RATING PREDICTION

$$\text{Minimize} \quad E = \textstyle\sum_{i\in\mathcal{N},j\in\mathcal{M}} \lambda_{i,j} \sum_{o\in\mathcal{O}} |\tilde{\pi}^o_{i,j} - \pi^o_{i,j}|^2, \tag{7a}$$

$$\text{subject to:} \quad p_i \in D_1, \quad i \in \mathcal{N}, \text{ and} \tag{7b}$$

$$q_j \in D_2, \quad j \in \mathcal{M}, \tag{7c}$$

where D_1 and D_2 are the sets of profiles that satisfy (2) and (3), respectively. We call the objective function E in (7a) the *error* of our estimate. It corresponds to the error of a rating selected uniformly at random among ratings within a timeslot. Its minimization is equivalent to minimizing a weighted root mean square error (RMSE), with weights equal to the delivery rates $\lambda_{i,j}$. In particular, when $\lambda_{i,j} = 0$ (*i.e.*, when i never delivers content to j), then E does not account for the distance between $\pi^o_{i,j}$ and $\tilde{\pi}^o_{i,j}$. This is not a bug but a useful feature: we never have to predict how j would react to content from i unless it receives such content.

Assumption 3 implies that there exists a small dimension, namely $|\tilde{\mathcal{F}}|$, such that if $d \geq |\tilde{\mathcal{F}}|$ the minimum error will be zero. When $d < |\tilde{\mathcal{F}}|$, there may not exist profiles yielding $E = 0$. In other words, *underestimating* the number of categories may preclude achieving a zero error; this is possible as the number of "categories" is often unknown.

4.2 A Distributed Learning Algorithm

Our distributed algorithm for solving RATING PREDICTION is specified in Figure 1. It is fully distributed, and ensures that a consumer discloses the rating of a content item only to the producer that generated it. Moreover, producers share their profiles only with consumers that subscribe to their feeds, and vice-versa.

In more detail, whenever producer $i \in \mathcal{N}$ delivers content to consumer $j \in \mathcal{M}$, the following interactions take place. First, in addition to the content item, i sends to j its profile p_i. Second, the consumer views and rates the content item with a rating $r_{i,j} \in \mathcal{O}$. Third, the consumer reports to the producer (a) the rating $r_{i,j}$ for this content, as well as (b) its consumption profile q_j. We assume that consumer j reports its rating and consumption profile to i instantaneously, upon receipt of the item. Nevertheless, our results can be directly extended to the case where these are reported with an arbitrary delay within the current timeslot (see [9]).

Upon exchanging the above information, i and j update their profiles as follows:

$$p_i \leftarrow p_i + \gamma \textstyle\sum_{o\in\mathcal{O}} (\mathbb{1}_{r_{i,j}=o} - \langle p_i, q^o_j \rangle) q^o_j \tag{8a}$$

$$q^o_j \leftarrow q^o_j + \gamma (\mathbb{1}_{r_{i,j}=o} - \langle p_i, q^o_j \rangle) p_i, \quad o \in \mathcal{O} \tag{8b}$$

where $\gamma = \gamma(k)$ is the learning rate. We assume that $\gamma(k) \geq 0$, $\sum_{k=1}^{\infty} \gamma(k) = \infty$, and $\sum_{k=1}^{\infty} (\gamma(k))^2 < \infty$; these conditions hold if, *e.g.*, $\gamma = 1/k$.

Finally, at the end of the timeslot, after (8a) and (8b) have been applied at every encounter the producers and consumers further update their profiles by forcing them to satisfy (2) and (3):

$$p_i \leftarrow \Pi_{D_1}(p_i), \quad i \in \mathcal{N} \tag{9a}$$

$$q_j \leftarrow \Pi_{D_2}(q_j), \quad j \in \mathcal{M} \tag{9b}$$

where $\Pi_{D_1} : \mathbb{R}^{|\mathcal{F}|} \to D_1$, $\Pi_{D_2} : \mathbb{R}^{|\mathcal{O}|\times|\mathcal{F}|} \to D_2$ are the orthogonal projections to D_1 and D_2, respectively. Due to the simple geometry of D_1, D_2, there are known algorithms that compute these projections efficiently [16, 17]. In particular, Π_{D_1} can be computed in $O(|\mathcal{F}|)$ steps, while Π_{D_2} can be computed in $O(|\mathcal{O}||\mathcal{F}|)$ steps.

4.3 Convergence Properties

In this section, we state our main result: the dynamics of our algorithm *always* lead to a decrease in E. This important property ensures that updating the production and consumption profiles improves our predictions, even if we underestimate the number of dimensions (*i.e.*, when $d < |\tilde{\mathcal{F}}|$). The proof is omitted for brevity, and can be found in our technical report [9], which also includes additional results regarding the stability of (8) and (9) around local minima.

Our argument is as follows. First, we show that the trajectories defined by the adaptations (8) and (9) asymptotically converge to the solution of a system of ordinary differential equations. Second, we prove that the error E always decreases. Our result relies on Assumptions 1 and 2 but not on Assumption 3: in particular, the weighted error E always decreases *even if the matrices* $\tilde{\Pi}^o$, $o \in \mathcal{O}$, *are not low-rank*. Nevertheless, as discused in Section 4.1, when Assumption 3 fails to hold, profiles under which E is zero may not exist.

Producer i at timeslot k:
  i generates new content item
  $\gamma \leftarrow \gamma(k)$
  for every pair i, j s.t. $a_{i,j}(k) = 1$:
    i sends its item and p_i to j.
    i receives $r_{i,j}$ and q_j from j.
    $p_i \leftarrow p_i + \gamma(k) \sum_{o \in \mathcal{O}} (\mathbb{1}_{r_{i,j}=o} - \langle p_i, q_j^o \rangle) q_j^o$
  $p_i \leftarrow \Pi_{D_1}(p_i)$

Consumer j at timeslot k:
  $\gamma \leftarrow \gamma(k)$
  for every pair i, j s.t. $a_{i,j}(k) = 1$:
    j receives item and p_i from i.
    j rates item with $r_{i,j} \in \mathcal{O}$.
    j sends $r_{i,j}$ and q_j to i.
    for every $o \in \mathcal{O}$:
      $q_j^o \leftarrow q_j^o + \gamma(\mathbb{1}_{r_{i,j}=o} - \langle p_i, q_j \rangle) p_i$.
  $q_j \leftarrow \Pi_{D_2}(q_j)$.

Figure 1: Distributed learning algorithm.

Consider the partial gradients

$$\nabla_{p_i} E = [\partial E / \partial p_{i,f}]_{f \in \mathcal{F}}, \quad \nabla_{q_j} E = \left[\partial E / \partial q_{j,f}^o\right]_{f \in \mathcal{F}, o \in \mathcal{O}},$$

and the following system of ODEs:

$$\frac{dp_i}{dt} = -\nabla_{p_i} E + z_{D_1}, \qquad i \in \mathcal{N}, \qquad \text{(10a)}$$

$$\frac{dq_j}{dt} = -\nabla_{q_j} E + z_{D_2}, \qquad j \in \mathcal{M}, \qquad \text{(10b)}$$

where z_{D_1}, z_{D_2}, defined formally in [9], are the minimum forces required to keep the evolution of the system within the constraint set $D_1 \times D_2$. In other words, the ODE (10) can be viewed as a *projected gradient descent* for the RATING PREDICTION problem.

Let $p_i(k)$, $q_j(k)$, be the production and consumption profiles of $i \in \mathcal{N}$, $j \in \mathcal{M}$, respectively, at timeslot $k \in \mathbb{N}$. We extend these to continuous-time functions $p_i : \mathbb{R}_+ \to D_1$, $q_j : \mathbb{R}_+ \to D_2$ as follows. Let $t_k = \sum_{i=1}^{k} \gamma(k)$ and define for all i and j the continuous-time interpolated processes:

$$p_i(s) = p_i(k), \quad q_j(s) = q_j(k), \text{ for } s \in [t_k; t_{k+1}[.$$

Theorem 1 below establishes that after sufficiently long time, the continuous-time interpolated trajectories of our system can be arbitrarily well approximated by solutions to (10). Moreover, any limit points of our system must also be limit points of the ODE (10).

THEOREM 1. *Consider the interpolated processes*

$$[p_i(t), q_j(t)]_{i \in \mathcal{N}, j \in \mathcal{M}}, \quad t \in \mathbb{R}_+.$$

Then, for any $T > 0$ there exist solutions of (10)

$$[p_i^*(t), q_j^*(t)]_{i \in \mathcal{N}, j \in \mathcal{M}}, \quad t \in \mathbb{R}_+,$$

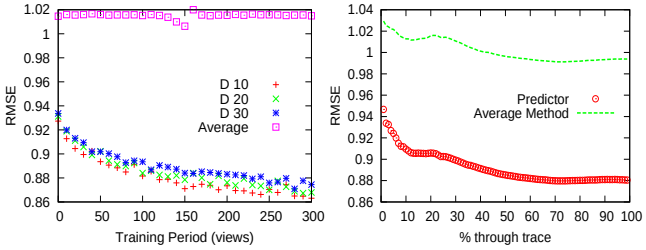
such that, as $k \to \infty$, the quantity

$$\sup_{t_k \le \tau \le t_k + T} \Big(\sum_i \|p_i(\tau) - p_i^*(\tau)\|_\infty + \sum_j \|q_j(\tau) - q_j^*(\tau)\|_\infty \Big)$$

converges to 0, in probability. In addition, $[p_i, q_j]_{i \in \mathcal{N}, j \in \mathcal{M}}$ converge, in probability, to the limit set of ODE (10).

Armed with the above description of system dynamics, the following theorem establishes that under any solution of (10) the error E decreases with time.

THEOREM 2. *If, $p_i(t)$, $q_j(t)$, $t \in \mathbb{R}_+$, evolve according to* (10)*, then $\frac{dE}{dt} \le 0$.*



(a) RMSE v. training time (b) RMSE Evolution

Figure 2: (a) RMSE as a function of the training period. (b) RMSE evolution through the course of the Netflix trace. Most of the drop in RMSE occurs in the first 10% of the trace.

Hence, the evolution of (10) pushes the system in the right direction, reducing the error function E. The above result is true *irrespective* of whether the user profiles have the same dimension as the rank of $\tilde{\Pi}^o$. Even if the ranks of the probability matrices are underestimated, the dynamics of (10) still push towards a decrease.

5. COMPARISON ON NETFLIX DATA SET

In this section, we test our rating prediction algorithm on the Netflix dataset. The wide use of the dataset as a benchmark allows us to implicitly compare the performance of our algorithm to the one achieved by state-of-the-art algorithms.

5.1 Netflix Data Set

In 2006, Netflix announced a competition for recommendation systems, and released a dataset on which competitors could train their algorithms. The Netflix dataset consists of pairs of anonymized movies and anonymized users. Each trace entry includes a timestamp, the user ID, and the user's rating (on an integer scale of 1 to 5).

The dataset includes both publicly-available training data, for which ratings were provided, and a testing dataset, for which ratings were not disclosed. If for every movie in the test set, we simply always predict its average rated value from the training set, this approach would yield a root mean square error of 1.0540 on the test set. The winning team of the Netflix Prize challenge generated predictions with RMSE of 0.8572 [3], a 10% improvement of the RMSE of Cinematch (0.9525), the algorithm designed by Netflix engineers.

We apply our algorithm as follows. Each movie is given a production profile $p_m \in [0,1]^d$. Similarly, each user is given an consumption profile $q_u \in [0,1]^{5 \times d}$, corresponding to ratings with one, two, three, four, or five stars respectively (*i.e.*, $\mathcal{O} = \{1,2,3,4,5\}$). The Netflix dataset is arranged chronologically and as users rate movies, p for the movie and q for the user are updated according to (8a) and (8b) with each rating and are subsequently projected to D_1 and D_2 according to (9).

5.2 Evaluating our Predictions on Netflix

We evaluate the performance of our predictions of user behavior in two ways, as discussed below.

RMSE of rating prediction.

Prior to each rating event, we predict the rating that a user, u, will give to a movie, m. We make this prediction by reporting the expected rating based on our estimation of

the rating probabilities, *i.e.*,

$$PredictedRating(u,m) = \sum_{o\in\mathcal{O}} o \cdot \langle q_u^o, p_m \rangle.$$

We can then compare our prediction to the user's rating for this movie as recorded in the dataset, to calculate an RMSE.

Our algorithm starts with a randomly selected p for each movie and a random q for each user and adapts them as users rate movies. To account for a training period, we discard some of the early predictions before computing the RMSE. More precisely, we compute an RMSE for predictions with training period more than k by including predictions for which either the movie or the user profile has been adapted at least k times.

Although we do not know how the Cinematch algorithm (Netflix's baseline) performs on the *training* dataset, we know that on the test set it performs roughly 10% better than the naïve algorithm that guesses the average rating for each movie. Therefore, to evaluate our effectiveness, we compare our RMSE against this "average" algorithm applied to the training dataset (100,480,507 ratings that 480,189 users gave to 17,770 movies). We know the RMSE of this "average" algorithm for both training (1.015) and test (1.05) datasets, and so we can compare to these numbers.

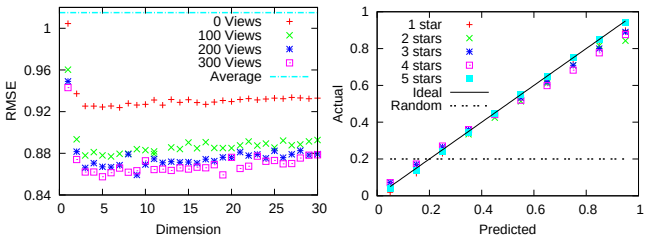
Figure 2(a) shows the improvement in RMSE as we vary the training period for different numbers of dimensions d of the vectors. Regardless of the length of the training period, the RMSE of the "average" method remains at 1.015. Again, though we do not know how the original Netflix algorithm (Cinematch) functioned and how it performs on the training dataset, we expect this centralized solution to achieve roughly 10% improvement (9.18 RMSE over the training dataset). Our algorithm ourperforms this value with a training period of only 10 iterations; a 15% improvement can be reached with training period of 300 views. Even when the training period is set to zero (*i.e.*, we include all predictions in the RMSE), it improves on the "average" method by 8% for all values of d, only slightly worse than Cinematch. Thus, our technique performs at least as well as some of the leading systems, even though our algorithm functions in a completely distributed manner.

Figure 2(b) shows how the RMSE evolves throughout the course of the netflix trace. Our technique makes 50% of it's improvements in RMSE during the first 10% of the trace, indicating that it performs well even in cold start situations. The algorithm consistantly tracks the performance of the "average" method while demonstrating a consistant improvement of 8%-12%. Figure 3(a) shows the effect of modifying the dimensionality of the prediction vectors on the RMSE for different training periods. As the dimensionality increases, we see a large drop; the RMSE quickly reaches a plateau between $d = 3$ and $d = 10$, and then increases slowly. We see the same trend for all training periods, with longer training periods tending to be flatter.

Thus, choosing the dimensionality is a tradeoff. On the one hand, a minimum number is needed to capture the diversity of the system, while on the other hand, too many dimensions makes the algorithm slow to learn and requires a longer training period. In practice, the best number of dimensions can be small (typically, 5 or 10), which also means that very little meta-data (5 to 10 floating-point numbers) must be transferred with each piece of content.

Predicted vs. empirical rating distributions.

We have seen that our algorithm performs well in terms of rating prediction RMSE. However, it can also be used to estimate the rating probability distribution. We now evaluate



(a) RMSE v. dimension (b) Rating Distribution

Figure 3: (a) Effect of dimensionality on the RMSE, which rapidly decreases until the time is insufficient to train all dimensions. (b) Comparison between predicted and empirical rating distributions.

whether our estimated distribution is observed in practice across all users and movies: *e.g.*, for all events where we predict that rating o is obtained with probability 0.3, do we observe that this rating is seen 30% of the time? This is, in effect, a measure of the precision of the rating predictor.

We measure the goodness of fit of our distribution to the empirical distribution in Figure 3(b). To compute it, we first bin our predicted probability into 10 bins (0 to 0.1, 0.1 to 0.2, etc.). We then compare the bin value to the actual rate of occurrence in the bin, for different ratings. Note that for all ratings, the square correlation coefficient R^2 is above 0.98 indicating a very good match. The slope of the best fit line is thus nearly 1. Another metric is the distance of our result to the line $y = x$ which would represent an ideal predictor. Random guessing, for instance, has a mean distance of 0.415. Our rating predictor brings a large improvement, with distance errors as small as 6×10^{-4}.

This accuracy implies that the system can do more than provide a single estimated rating for a specific movie and user, it can actually characterize its behavior on a finer grain which may help provide additional confidence. As an example, it is interesting to note that users are more predictable when they express strong opinions (5 starts, 1 stars), slightly less predictable when they are neutral, and even less predictable when they indicate 4 or 2 stars.

6. CASE STUDY: WEBDOSE

The evaluation of our rating prediction algorithm on the Netflix data set demonstrates that it can, indeed, be used to make accurate predictions. However, it does not illustrate its behavior in a distributed, heterogeneous environment.

Therefore, we implemented our system as a Facebook application ("WebDose"), which allowed Facebook users to view, rate, and share content in the form of web pages. We used Facebook as a distribution platform to ensure that the application would be immediately available to a wide range of users without requiring extensive development overhead.

6.1 Application Description

User interface and content propagation..

When users log into the system, WebDose presents them with a screenshot of a web page and three rating icons: "thumbs up," "thumbs down," or "next". We interpret each of these ratings as positive, negative or neutral, respectively (*i.e.*, $\mathcal{O} = \{+, -, \varnothing\}$). By design, users are not shown a new page until they have clicked a button and thus rated the page they are currently viewing. Additionally, users

have the option to "Add" (*i.e.*, produce) content to the system by typing a url into a text box at the bottom of their screen. WEBDOSE, thus, follows the same producer/consumer model used throughout this paper.

We emulate content sharing as follows. When a user logs in (or refreshes the page), she or he "contacts" another user that has also logged in within the previous three hours. If no such user exists, no contact occurs. A user may experience many contacts during a browsing session as other users log in and trigger their own "contact events." At a contact, users swap pages they have generated or received from other users, thereby propagating them through WEBDOSE. To account for social relationships, when multiple people have logged in the system in the past 3 hours, we bias contact events between users that are Facebook friends.

Behind-the-scenes prediction.

The first time a user u logs into the system, she is assigned a random production profile p_u and a random consumption profile q_u. The dimension of these profiles is set to $d = 10$; this will be examined in Section 6.3.

We also incorporate item profiles in WEBDOSE. When an item is propagated among several consumers, rather than modifying these profiles, it is preferable to adapt them *as the item is propagated from one consumer to the next.* The system is extended so that content items generated by a producer a are associated with a profile vector $t \in [0,1]^d$, that is initialized to $t = p_a$ when the item is generated. This profile is delivered along with the item to every consumer that it passes through. However, instead of remaining static, the item profile is adapted through (8a) and (9). A formal analysis of joint dynamics of a system in which publisher, consumer *and* item profiles are adapted is quite intricate and beyond the scope of the present paper. Nevertheless, we verify the performance of such a combined evolution through the WEBDOSE experiment.

As web pages are rated by other users of the system, t_w and q_u are updated according to Eq. (8a) and (8b) respectively. Both the predicted as well as the actual rating are logged. Web pages in the system carry an identifier of the web page's producer, *i.e.*, the user that added it to WEBDOSE. When a user rates a page, the system stores the producer and rating locally. Subsequently, when the two users have a "contact", the web page consumer informs the web page producer of the rating given to its content. The producer then uses this information to update its profile p_u according to Algorithm (8a). All updates are followed by an immediate projection to D_1, D_2, as in (9).

WEBDOSE uses our prediction algorithm to select pages to show to a user. Each time a page is shown, WEBDOSE first calculates π^+ for each as-yet-unrated web page in the user's cache. The page with the highest π^+ is then shown to the user. Once that page has been rated, π^+ is recalculated for each unrated page and the new highest page is shown. We chose this form of recommendation to ensure that a user is always able to view all of the content in their cache, and that WEBDOSE would not be hindered from training user's profiles through pages not viewed.

6.2 Webdose Experiment

During a 33 day period, 43 users spanning 12 time zones registered for WEBDOSE and viewed or added 326 web pages. Although these small numbers make learning user preferences difficult, we will show in Section 6.3 that the system was able to adapt well. Users were free to log in as frequently as they wished and could rate as many pages as they wanted. Once a page was rated, it was never shown to that user again. As an incentive to add quality content to the system, we provided each user with statistics of the ratings their content received, as well as their ranking in "thumbs-up" and "thumbs-down" received.

Users of the system had a wide range of usage behaviors. Half of the users had more than 10 contacts and 20% had over 50. The most active user rated over 180 pages; 25% of users rated more than 20 pages. This is consistent with a real environment: most users look for web pages occasionally, while an active few regularly check for pages.

For the system to function as intended the pages must be viewed frequently enough that the item profile t is trained. The high number of pages in the system (over 300) compared to the number of users means that each page is viewed by only a limited number of users. Over the course of our experiment, no page was viewed more than 11 times. However, more than half the pages were viewed more than 4 times, which proves to be sufficient.

It is not necessarily true that the highest producers are also the highest consumers. The relatively low number of high producers indicate that it is crucial that item profiles are introduced, otherwise all pages from the same producer will have the same profile, slowing learning. Item profiles allow the system to distinguish different pages from the same producer, accelerating convergence by training individual web pages while the production profiles train.

The majority of users favor neutral ratings, followed by those that mostly rate pages thumbs down. With most people being generally ambivalent or negative on most content, it is important that the system make accurate predictions.

6.3 Predictive Power

WEBDOSE *prediction accuracy.*

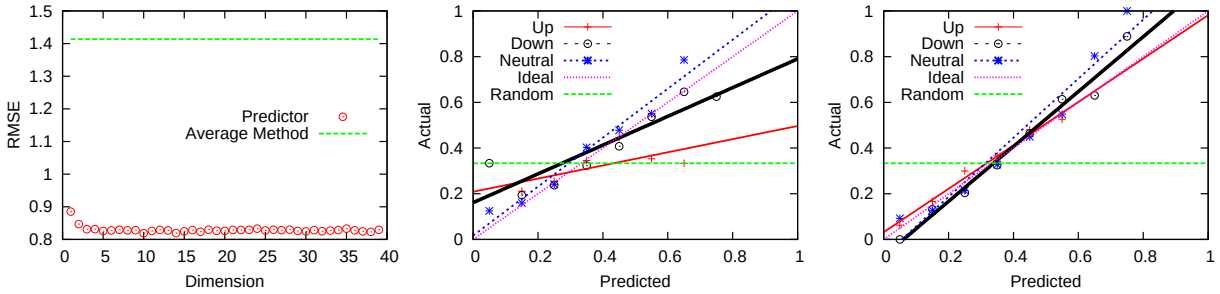
To assess the correctness of predictions in WEBDOSE, we repeated the evaluation of our predictors with respect to the two metrics we considered over the Netflix data set, the RMSE and the goodness of fit of the predicted distribution to the empirical distribution. To compute the RMSE, we associate the values -1,0,+1 to $-, \varnothing, +$ respectively. Before the rating of a web page w by a user u, we again generate our estimate of the rating as

$$PredictedRating(u,w) = \textstyle\sum_{o\in\{-1,0,+1\}} o \cdot \langle t_w, q_u^o \rangle,$$

where t_w, q_u the item and consumption profiles, respectively.

Figure 4(a) displays the RMSE of our predictions as a function of the dimension d of our profiles. The dimension 10 was used in the actual experiment and other values were obtained by running our algorithm on the collected trace. As in Netflix, we again observe the error quickly decreases up to a dimension of 10, after which it remains constant. We note that the RMSE for $d = 10$ is 0.819, 42% less than the RMSE obtained by predictions obtained by the running average ratio of each rating.

Figure 4(b) shows that the outcomes "Thumb down" and "neutral" are both predicted accurately, with slopes above 0.63 and exhibiting good fits (R^2 above 0.8). The "thumbs-up" predictions are considerably worse, with a slope of 0.29. This is not surprising, given that most ratings in the system were for "thumbs-down" and "neutral": the experiment does not provide enough data to train quickly enough for "thumbs-up" ratings. For this reason, we repeated the goodness of fit test using instead the content and consumption profiles *at the end* of the experiment. Figure 4(c) demon-



(a) RMSE (b) *a priori* prediction (c) convergence

Figure 4: Predictions of the thumb rate of pages. (a) shows the effect of the dimension chosen at the outset of the experiment. As the dimension increases, the error decreases. Our selection of 10 occurs below the knee of the curve. In (b) and (c) pages are binned according to estimated percentage of "thumbing" them in a given direction. (b) shows *a priori* prediction and moderate accuracy, with slopes from 0.29 to 1.07. (c) shows convergence of π at the end of the experiment and has slopes from 0.95 to 1.3. In all figures, the green line gives a comparison to other predictors.

strates that, after convergance, the algorithm makes high accuracy predictions, with slopes between 0.95 and 1.3.

7. CONCLUSIONS

User generated content is a primary factor to the success of the social web as it connects users through content sharing. It presents unique challenges for collaborative filtering due to a large volume and the sensitivity of sharing information beyond one's immediate circle of acquaintances. This paper proves that information exchange can be deployed only among trusted pairs while providing strong guarantees on the rating prediction error, thanks to a fully distributed and asynchronous learning algorithm.

Our results expand the scope of recommender systems to operate on top of any social content sharing platform, which unveils important research questions. The case where trust is not reciprocal (such as the follower relationship within Twitter) remains an interesting open case. Our algorithm could leverage secure multiparty computation, to provide at a smaller cost the privacy guarantee offered by centralized schemes. Finally, our model can be used to analyze how social proximity, captured through "rate of delivery" for producer-consumer pairs, impacts the efficiency of learning. Both questions are interesting open problems.

8. REFERENCES

[1] Acquisti, A., and Grossklags, J. What can behavioral economics teach us about privacy? In *Digital Privacy: Theory, Technologies and Practices* (2007), pp. 363–377.

[2] Angwin, J. US seeks web privacy 'bill of rights'. *Wall Street Journal* (Dec. 17th 2010).

[3] Bell, R. M., and Koren, Y. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *Proc. IEEE ICDM* (2007).

[4] Brandimarte, L., Acquisti, A., and Loewenstein, G. Privacy concerns and information disclosure: An illusion of control hypothesis. In *Proc. CIST* (2010).

[5] Candès, E., and Tao, T. The power of convex relaxation: Near-optimal matrix completion. *IEEE Trans. Inform. Theory 56*, 5 (2009), 2053–2080.

[6] Candès, E. J., and Recht, B. Exact matrix completion via convex optimization. *Found. of Comput. Math. 9* (2008), 717–772.

[7] Canny, J. Collaborative filtering with privacy via factor analysis. In *Proc. ACM SIGIR* (2002).

[8] Castagnos, S., and Boyer, A. Personalized Communities in a Distributed Recommender System. In *Proc. ECIR* (2007).

[9] Isaacman, S., Ioannidis, S., Chaintreau, A., and Martonosi, M. Distributed rating prediction in user generated content streams. Tech. Rep. PAC-2011-05-01, Technicolor.

[10] Kempe, D., and McSherry, F. A decentralized algorithm for spectral analysis. *Journal of Computer and System Sciences 74*, 1 (feb 2008).

[11] Keshavan, R., Montanari, A., and Oh, S. Matrix completion from noisy entries. *JMLR 11* (2010), 2057–2078.

[12] Korada, S. B., Montanari, A., and Oh, S. Gossip PCA. *Proc. ACM SIGMETRICS* (2011).

[13] Koren, Y. Collaborative filtering with temporal dynamics. In *Proc. ACM KDD* (2009).

[14] Lathia, N., Hailes, S., and Capra, L. Private distributed collaborative filtering using estimated concordance measures. In *Proc. ACM RecSys* (Oct. 2007).

[15] Liu, K., Bhaduri, K., Das, K., Nguyen, P., and Kargupta, H. Client-side web mining for community formation in peer-to-peer environments. *Proc. WEBKDD* (2006).

[16] Maculan, N., Santiago, C., Macambira, E., and Jardim, M. An $O(n)$ algorithm for projecting a vector on the intersection of a hyperplane and a box in $\mathbb{R}^n$. *J. of Opt. Th. and App. 117*, 3 (2003), 553–574.

[17] Michelot, C. A finite algorithm for fiding the projection of a point onto the canonical simplex of $\mathbb{R}^n$. *J. of Opt. Th. and App. 50*, 1 (1986), 195–200.

[18] Miller, B., Konstan, J., and Riedl, J. PocketLens. *ACM Transactions on Information Systems 22* (2004), 437–476.

[19] Polat, H., and Du, W. Privacy-preserving collaborative filtering using randomized perturbation techniques. In *Proc. IEEE ICDM* (2003).

[20] Ruffo, G., and Schifanella, R. A peer-to-peer recommender system based on spontaneous affinities. *ACM Transactions on Internet Technology 9* (2009), 1–34.

[21] Salakhutdinov, R., and Mnih, A. Probabilistic matrix factorization. *Advances in Neural Information Processing Systems 20* (2008).

[22] Takács, G., Pilászy, I., Németh, B., and Tikk, D. Scalable collaborative filtering approaches for large recommender systems. *JMLR 10* (2009), 623–656.

[23] Vascellaro, J. E. Google agonizes on privacy as ad world vaults ahead. *Wall Street Journal* (Aug. 10th 2010).

[24] Wang, J., Pouwelse, J., Lagendijk, R., and Reinders, M. Distributed collaborative filtering for peer-to-peer file sharing systems. In *Proc. ACM SAC* (2006).

[25] Wingfield, N. Microsoft quashed effort to boost online privacy. *Wall Street Journal* (Aug. 2nd 2010).