Dimension Reduction Methods for Collaborative Mobile Gossip Learning

Árpád Berta
University of Szeged, Hungary
Email: berta@inf.u-szeged.hu

István Hegedűs and Márk Jelasity
University of Szeged, MTA-SZTE Research Group on AI, Szeged, Hungary
Email: {ihegedus,jelasity}@inf.u-szeged.hu

***Abstract*—Decentralized learning algorithms are very sensitive to the size of the raw data records due to the resulting large communication cost. This can, in the worst case, even make decentralized learning infeasible. Dimension reduction is a key technique to compress data and to obtain small models. In this paper, we propose a number of robust and efficient decentralized approaches to dimension reduction in the system model where each network node holds only one data record. These algorithms build on searching for good random projections. We present a thorough experimental comparison of the proposed algorithms and compare them with a variant of distributed singular value decomposition (SVD), a state-of-the-art algorithm for dimension reduction. We base our experiments on a trace of real mobile phone usage. We conclude that our method based on selecting good random projections is preferable and provides good quality results when the output is required on a very short timescale, within tens of minutes. We also present a hybrid method that combines the advantages of random projections and SVD. We demonstrate that the hybrid method offers good performance over all timescales.**

***Keywords*—distributed data mining, gossip, dimension reduction**

I. Introduction

Our research targets networked systems where each networked device stores only a small amount of data (typically collected locally), while there are possibly millions of participating devices in the network. This model covers a wide range of applications including smart metering [1], collaborative mobile platforms [2] and Internet of Things platforms [3].

To implement machine learning algorithms in such a system model, we opted for gossip learning algorithms [4]. There are, of course, many ways to perform data mining in a decentralized environment [5]. Gossip learning is attractive because of its natural emphasis on privacy and full decentralization, while being efficient enough for most data mining tasks. Privacy is achieved in part by assuming that devices do not share raw data either with each other or with any external party. In addition, theoretical notions of privacy such as differential privacy can be incorporated in this framework as well [4], [6], [7]. In-place data processing could also provide better scalability for certain tasks in the limit of extremely large systems compared to cloud-based solutions by exploiting local resources and networks, as proposed e.g. by Cisco in its ongoing fog computing initiative [8].

However, one key concern in the above scenario is communication complexity that is often proportional to the size of the raw data. Raw data might be very high dimensional, such as images from a surveillance camera, or text documents containing private communication. It is essential to compress this data locally using a shared method. Algorithms that compress raw data wile preserving its information content are called dimension reduction methods [9]. Our goal in this study is to propose practical distributed dimension reduction algorithms for gossip learning that are efficient yet perform well in learning tasks.

Most known methods for distributed dimension reduction are unsuitable for gossip learning. Often it is assumed that there is an aggregator node that processes and aggregates output from all the networked nodes [10]. It is also often assumed that all the nodes have sufficient data to produce meaningful partial results to be aggregated. These assumptions violate both our system model (in which there is only very little data at each node) and our objective of decentralization. Methods in this class include several feature selection methods that have been implemented in the MapReduce framework [11]. Landmark-based methods also have distributed variants [12]. Here, extremal points are selected from the raw data that are used to encode a lower dimensional distance-preserving representation. Many methods seek to find an optimal linear mapping based on spectral properties of the data such as principal component analysis (PCA) [13].

Known algorithms that are suitable for gossip learning include singular value decomposition (SVD), which is a powerful method that has a distributed implementation compatible with our system model [14].

Our contributions are threefold: (1) we propose a method for dimension reduction based on selecting good random projections using an algorithm compatible with the gossip learning framework, (2) we propose a hybrid algorithm based on SVD and random projections that combines the advantages of both pure algorithms, and (3) we perform an extensive empirical analysis of these algorithms using real smartphone traces over several learning tasks.

II. Background

A. System Model and Data Distribution

We consider a network of a potentially large number of computational units (personal computers, smart sensors,

In: Proc. PDP 2016, pp 393–397, http://dx.doi.org/10.1109/PDP.2016.20.

wearable devices, phones, tablets, etc.), called nodes. These nodes can communicate with their neighbors via messaging. The neighbors are provided by the peer sampling service [15], [16]. Nodes can leave the network and join again without any prior notice. The only assumption is that the node state is unchanged when it is offline. In our experiments we will base our simulated model on a real smartphone trace. Every node is assumed to have only one complete data record, which is also called a training example.

B. Dimension Reduction

In machine learning we are given a set of *training examples* from an unknown distribution. Here we assume that an example is of the form (x, y), where x is a d dimensional real *feature* vector ($x \in \mathbb{R}^d$) and y is the *class* of the example. In the classification task, we look for a function $f(x; w) : \mathbb{R}^d \to C$ that categorizes any feature vector x into a finite number of classes, where C is the set of classes and w is a parameter vector that we wish to learn. The function $f(x; w)$ is often called the *model* of the data. Parameter w is typically found through some local gradient method that optimizes a loss (or error) function, which characterizes the accuracy of the model over a set of classified training examples.

If the number of features d is high, we might experience the curse of dimensionality, which means that classification algorithms will perform poorly [17]. Formally, the dimension reduction methods are functions that transform the data into a lower dimensional space $\mathbb{R}^k$, where $k \ll d$. Here, we focus on linear projection methods where we wish to find a matrix $P \in \mathbb{R}^{k \times d}$ with k rows and d columns. Thus the projection of a training example x is a simple vector-matrix multiplication Px. Here we consider two feature extraction methods: *Random Projection* [18] and *Principal Component Analysis (PCA)* [13].

The simpler method is random projection, where the data is transformed by a random matrix to a lower dimension. The Johnson-Lindenstrauss lemma gives a lower and upper bound on the error of the projection, and the random projection technique exploits this result [18], [19]. PCA is a method to find those k directions in the d dimensional space that span a subspace where the covariance of the data is maximized. As a side effect, PCA optimally preserves the distances between data points. Now, let us define matrix $X = [x_1, \ldots, x_N] \in \mathbb{R}^{d \times N}$ that has the training examples in its columns. The projection matrix here is given by the matrix of the eigenvectors that correspond to the first k eigenvalues of the covariance matrix of the training examples XX^T.

In this paper we will calculate the singular value decomposition (SVD) of X^T using the algorithm in [14] to get the projection matrix of PCA. We first assume that the database is such that all the dimensions (features) have zero mean. When using PCA, one first computes the covariance matrix of the data $XX^T \in \mathbb{R}^{d \times d}$, which is a symmetric, positive semidefinite matrix. Diagonalizing the covariance matrix $XX^T = PDP^T$ gives the matrix P. The projection matrix of PCA is given by keeping the first k rows of P^T. Matrix P can be calculated using the SVD of X^T, where $X^T = U\Sigma V^T$. By substituting this into the covariance matrix we get $XX^T = (U\Sigma V^T)^T(U\Sigma V^T)$. Since U is orthonormalized, we have $XX^T = V\Sigma^2 V^T$. Letting $D = \Sigma^2$ we have $P = V$.

Algorithm 1 Gossip Learning Framework

```
1: (x, y) ← local training example
2: currentModel ← initModel()
3: loop
4:     wait(Δ)
5:     p ← selectPeer()
6:     send currentModel to p
7: end loop
8: procedure ONRECEIVEMODEL(m)
9:     m.updateModel(x, y)
10:    currentModel ← m
11: end procedure
```

C. Gossip Learning

To learn models over a large fully distributed database we are motivated by *Gossip Learning* [4]. This framework works as follows. Every node, when joining the network, initializes the parameters of a model and sets it as its local model. Recall that by model we mean the function $f(x; w)$ with parameter w, which categorizes a given feature vector x. All the participating nodes send their local model periodically to a randomly selected node in the network. The address of the destination node is provided by a peer sampling service [15], [16]. When a node receives a model it updates it by its local data. This learning step is typically implemented by applying a stochastic gradient step on the model using the local example. The node then stores the updated model as its new local model. In effect, the models perform *random walks* in the network and are updated when they visit a node. The skeleton of the framework can be seen in Algorithm 1.

III. Algorithms

Here, we present two algorithms for decentralized dimension reduction. The first one is based on the idea of generating random projections, evaluating them through quickly training a model using only a few updates, and selecting the projection that results in the best preliminary model. The second one is a hybrid algorithm that combines SVD and random projections. The hybrid algorithm inherits the quick convergence of the random projection method and the high quality of SVD.

A. Random Projection Selection

Random projections are very cheap to generate, and a random matrix can be communicated through sending only the corresponding random seed. Due to this, it is cheap to evaluate a lot of different random matrices searching for the best one for the problem at hand. Given a fixed machine learning problem (data and a learning algorithm), our idea is to evaluate a random projection based on the accuracy of the outcome of the learning algorithm, provided that the given random projection is used as the dimension reduction algorithm. To evaluate a random projection, a naive approach would be to train a model based on the given random projection and then evaluate this model on a test set. Instead of this, we train the model using gossip learning and before each update step we evaluate it on the local training example (using it as a test example). We then use the running average of these evaluations as a rough approximation of the true performance of the model.

The skeleton of our algorithm to be run on each node is shown in Algorithm 2. The algorithm resembles the periodic gossip learning algorithm (Algorithm 1) but there is an important difference. Instead of being periodic, the algorithm implements the random walks in a "hot potato" style. We now describe the algorithm in more detail.

Algorithm 2 Random projection selection at node i

```
1: procedure INITNODE
2:     (x, y) ← local training example
3:     initModel(currentModel)
4:     initModel(bestModel)
5:     if selectedAtRandom(π) then
6:         p ← selectPeer()
7:         send currentModel and bestModel to p
8:     end if
9: end procedure
10: procedure ONRECEIVEMODELS(m_c, m_b)
11:     currentModel ← m_c
12:     update(currentModel)
13:     bestModel ← getBetter(bestModel,m_b)
14:     update(bestModel)
15:     if currentModel.age ≥ minAge then
16:         bestModel ← getBetter(bestModel,currentModel)
17:         initModel(currentModel)
18:     end if
19:     p ← selectPeer()
20:     send currentModel and bestModel to p
21: end procedure
22: procedure UPDATE(m)
23:     R ← createSparseRandomMatrix(k, d, m.seed)
24:     x_red ← Rx
25:     ŷ ← predict(m.model, x_red)
26:     m.error ← (1 − λ)m.error + λ|y − ŷ|
27:     m.model ← updateSGD(m.model, x_red, y)
28:     m.age ← m.age + 1
29: end procedure
30: procedure INITMODEL(m)
31:     m.age ← 0
32:     m.error ← 0
33:     m.seed ← getNextRandomSeed()
34:     m.model ← initSGD()
35: end procedure
```

Each node is initialized when joining the network. This consists of setting up the local variables and potentially starting a random walk of a new model to initiate gossip learning. The local data includes the training example (x, y) and models under training. Since we would like to find the best possible projection (that is, the one that results in the best model) and we also want to keep exploring new projections, every node needs to keep track of two different models: BESTMODEL (the best model known by the node) and CURRENTMODEL (the model currently under evaluation, as in gossip learning).

Each model stores the random seed that is used to generate the random projection that in turn is used to compress the local data x. In addition, the model stores the machine learning model itself, which is being trained via gossip learning. We also keep track of the number of updates (age) and the accumulating error that is used to assess the performance of the model, and thus the performance of the random projection.

Every node initially picks a random seed for a new projection matrix.Based on the seed, the projection matrix $R \in \mathbb{R}^{k \times d}$ is always generated in line 23 by randomly setting $r_{ij} = 1$, 0, or -1 with a probability of $1/6$, $2/3$, and $1/6$, respectively. With this choice, the matrices are sparse but still satisfy the Johnson-Lindenstrauss lemma [18], [19]. Obviously, the same pseudo random generator should be used at each node so that the same projection matrix is generated for the same seed.

Finally, with probability π the node initiates a random walk. The probability defines the number of overall random walks in the network, which is πN in expectation. This defines, among other properties, the overall bandwidth utilization.

In method UPDATE, first dimension reduction is performed on the local example that creates vector x_{red} of length k. Next, before performing the model update step, we update the running average of the prediction error using the local example. This is used to approximate the prediction performance of the model (and thus the random projection). The running average has a parameter λ that determines the balance of old and new information. If λ is too large then only the most recent examples will count and the measure will contain too much noise. If it is too low then the old examples will count too much, and the improvement in time is not reflected sufficiently.

We then perform the model update according to gossip learning; that is, we apply a stochastic gradient descent (SGD) update step using the local data. The implementation of the SGD update depends on the learning algorithm of choice; in our experimental evaluation, we will use logistic regression.

Models arrive in pairs. Model m_b is the sending node's approximation of the best model. This model participates in a gossip-based minimum search; that is, we compare it with the local approximation of the best model and keep the better one of the two. Method GETBETTER selects the model with the smaller error, or with the higher age in case at least one of the models is younger than MINAGE steps. Model m_c participates only in the gossip learning algorithm. When it reaches the age (number of updates) of MINAGE (a parameter), it becomes eligible for competing for the title of best model in the network. Since CURRENTMODEL is now participating in the global minimum search, we start a new model to test. This is our exploration strategy: new random projections are tested continuously while the mature ones contribute to the global minimum search. The role of parameter MINAGE is to define the number of updates needed to get a reliable error approximation. Note that our goal is to have an error approximation that is suitable for ordering different candidates as opposed to approximating the exact error, which allows for a relatively small MINAGE value.

B. Communication complexity

As of the size of a single message, the nodes send two models in a message that contains only the seed of the random projection matrix, and the model parameters, which—as we will see—have the size of $O(k)$ in the case of linear learning algorithms. To be more precise, assuming an 8 byte representation of both the floating point and integer parameters, we have a message size of $2 \cdot (8 \cdot k + 3 \cdot 8)$. This leads to a very small, by today's standards practically negligible, message size since k is typically small. In contrast to this, in the SVD protocol the whole projection matrix of size $O(k \cdot d)$ needs to be sent in each message [14]. This can be very large since d is potentially in the order of millions.

In both protocols the overall communication complexity can be controlled by the number of random walks πN in the network that determines the overall bandwidth consumption. Thus, given any bandwidth quota, these protocols can be executed within that quota via setting the number of random walks. Note that, due to the extremely different message sizes, this means that with the same bandwidth quota, the two algorithms will have a very different iteration speed.

C. A hybrid algorithm

As mentioned before, in our evaluation we will consider the distributed SVD implementation described in [14] that also outputs a projection matrix $P \in \mathbb{R}^{k \times d}$. We cannot describe the SVD algorithm in detail here, but a few details are important to understand our hybrid approach.

The motivation is that—although the SVD algorithm is also based on a variant of gossip learning—it operates with messages of size $O(k \cdot d)$ that can be very large for a large d. Thus, with a fixed bandwidth, the SVD algorithm converges much slower than random projection selection. At the same time, SVD provides a very high quality projection that is expected to outperform random projections. Our goal is to combine the advantages of the two algorithms, that is, with a fixed bandwidth we would like to get a good projection at any point in time that eventually converges to the SVD output.

Now, let R_t be the best random projection at time t and P_t be the projection calculated by the SVD algorithm at the same time. Another detail of the SVD algorithm is that the rows of the projection matrix converge in a sequential order, and in the converged state the rows are pairwise orthogonal. Our basic idea is that we run the two algorithms in parallel and define a projection $\hat{P}_t(R_t, P_t)$ that will use the converged rows from P_t and fills in the rest of the rows from R_t.

Thus, all we need is a method to determine the row index up to which P_t is considered to have converged. We first introduce a measure of orthogonality o_i for a given row p_i^T:

$$o_i = \frac{1}{k-1} \sum_{j \neq i} \frac{p_i^T \cdot p_j^T}{\|p_i^T\| \|p_j^T\|},$$

which is the average cosine distance of p_i^T from the rest of the rows. We include in $\hat{P}_t$ the first i rows of P_t where i is such that $o_i \leq \text{MINORT}$ but $o_{i+1} > \text{MINORT}$ (or $i = k$). Here, MINORT is a threshold parameter that defines how aggressively we include SVD vectors. A value close to zero is conservative, while a smaller value is more aggressive.

Although we propose to run the two algorithms in parallel, we do not assign the same bandwidth quota to each. Instead, we allow the SVD algorithm to consume almost all the bandwidth quota, and assign only 1% of it to random projection selection.

IV. Experimental Results

A. Experimental Setup

Here, our goal is to demonstrate that the proposed methods can be applied over real-world datasets under realistic network conditions. The key parameter in dimension reduction is k, that is, the reduced dimensionality. Our motivation is to demonstrate empirically how our distributed algorithms perform with different values of k, so here k is our main free parameter.

We simulate node churn based on a real trace of smartphone user behavior. We also ensure that we use the phone only when it is connected to a charger, so as to save the battery. The trace we used was collected by our smartphone app called STUNner, as described previously [20]. In a nutshell, the app monitors and collects information about charging status, battery level, bandwidth, and NAT type.

TABLE I. The key properties of the data sets

	MNIST	Farm ads	HAR
Training set size	60 000	3 314	7 352
Test set size	10 000	829	2947
Number of features	784	54 877	561
Original number of classes	10	2	6
Positive examples	10%	53%	17%

TABLE II. Parameter settings

Alg.		MNIST	Farm ads	HAR
RPSVD	MINORT	0.5	0.5	0.5
RP	λ	0.05	0.05	0.05
	MINAGE	200	200	200
	message size (Mbit, $k = 1$)	0.0003	0.0003	0.0003
	message size (Mbit, $k = 64$)	0.0084	0.0084	0.0084
SVD	α	10^{-4}	10^{-2}	10^{-2}
	message size (Mbit, $k = 1$)	0.05	3.51	0.03
	message size (Mbit, $k = 64$)	3.21	224.77	2.29
Log. Reg.	α	10^{-2}	10^{-2}	10^{-2}

We have traces of varying lengths taken from 1191 different users. We divided these traces into 2-day segments (with a one-day overlap), resulting in 40,658 segments altogether. With the help of these segments, we were able to simulate a virtual 2-day period by assigning a different segment to each simulated node. When we needed more users than segments, we re-sampled the segments to artificially inflate the number of users.

To ensure our algorithm is phone and user friendly, we defined a user to be online (available) when the user has a network connection and the phone is connected to a charger, thus we never use battery power at all. In addition, we also treated those users as offline who had a bandwidth of less than 1 Mbit/s, and we filtered out the online sessions that lasted less than a minute as well.

In our experiments we used data sets taken from various machine learning domains with different properties. Our first data set, called MNIST [21], contains gray level images of size 28×28 of handwritten digits (from 0 to 9).

The remaining two data sets are part of the UCI machine learning repository [22]. The second data set we chose was the Farm ads data set. This is a text classification data set that has a large number of features. These features include those extracted by the well-known bag-of-words technique from websites as well as higher level features. The task is to decide whether the owner of a Web content approves an ad.

Finally, we used the Human Activity Recognition (HAR) data set as well. Here the features were preprocessed by the owner of the data. The goal is to discriminate different activities based on the data of smart phone sensors (e.g., accelerometer and gyroscope).

We performed binary classification so we had to transform the MNIST and HAR data sets to binary classification problems. To achieve this, we selected the classes "number 7" and "walking" as positive classes from the MNIST and HAR data sets, respectively, and the examples in the remaining classes were treated as negative examples. The key properties of these three data sets are summarized in Table I.

The algorithm parameters are summarized in Table II. From now on, the name RPSVD will denote the hybrid algorithm

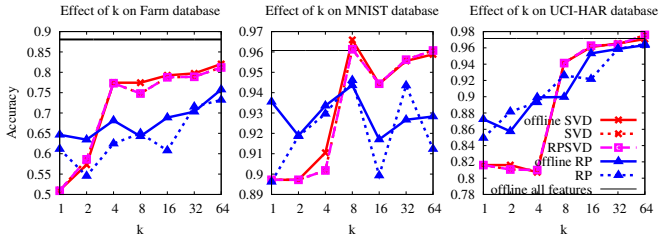

Fig. 1. Accuracy after two days of simulated time as a function of k.

and RP will denote random projection selection. Logistic regression is used inside RP as the gossip learning component.

The communication cost of each protocol was limited so that a node consumes 1 Mbit/s on average. This was achieved by limiting bandwidth in the simulation. In the case of RPSVD we set $\pi = 0.01$ in the RP component and everywhere else we set $\pi = 1$.

Recall that we use only those nodes that are on a charger, and that we include only those nodes that have a larger bandwidth than 1 Mbit/s. As a result, a node typically experiences much less load when averaged over the simulation period unless it is on a charger as well as online all the time. That is, based on the real trace we use, this is a practical setting.

Apart from bandwidth, message latency also limits the speed of the random walks. When the message size is very small (as in the case of RP), this can be the main limiting factor, so the RP algorithms will use much less bandwidth than allowed. We simulated a 100 ms latency. This means that in RP algorithms it takes as little as 20 seconds to evaluate a random projection candidate (assuming MINAGE=200), as the computation time is practically negligible.

The peer sampling service is assumed to be based on a static network, in which every node has 50 random neighbors from the whole population. This is a realistic setup on the real Internet, since in the case of stable connections one needs to perform NAT traversal only once, potentially with the assistance of a server in the connection phase [16].

B. Discussion

Figure 1 illustrates the prediction accuracy of our three distributed algorithms at the end of the simulated two days. Accuracy is defined as the fraction of correctly classified instances. To evaluate our distributed protocols, we chose a uniform random online node from the network and calculated its current dimension reduction offline. This was done by taking the dimension reduction matrix, transforming the training set and then training using R (the OPTIM gradient solver), and calculating the accuracy on the test set. We opted for this methodology because it was not computationally feasible for us to report statistics at every measurement point. However, variance is illustrated by the smoothness of the curves. The algorithms are compared to offline (centralized) variants. The offline SVD was calculated using R (the SVD function) and the offline RP is the best of 10,000 random projections, where the evaluation during the selection process is identical to that of the distributed version. We also include the accuracy based on training a logistic regression model that uses all the features.

We may conclude that the distributed algorithms approximate the offline variants very well at the end of the second day. As we increase k, we can approximate the performance of the full feature set, except in the highest dimensional data set. It is also clear that SVD outperforms random projections for larger k-s. Clearly, the hybrid method, by design, is identical to the SVD algorithm in its converged state.

Figure 2 shows our results as a function of time. Now, we plot the accuracy of the distributed methods for each minute, following the same evaluation methodology as before.

It is clear that, for SVD, the most important parameter is d, the original number of features. In the Farm Ads data set we have the largest number of features and thus SVD converges rather late (note the logarithmic scale of the plots). This is because the message size (hence the number of iterations) depends on d. However, RP converges almost instantly independently of d. This is not surprising as the communication complexity of RP is independent of d and the message size is very small, allowing for a large number of iterations in a very short time. However, the best performance of RP remains below that of SVD, especially for larger values of k.

The hybrid approach SVDRP combines the advantages of the two methods, and provides a good dimension reduction transformation at all timescales, at the same cost as any of the individual algorithms. The significance of this finding is that SVDRP is a method that is more robust than either SVD or RP alone and it can be applied with minimal knowledge about the problem at hand without parameter tuning for a certain wall-clock-time budget. The only exceptions are the smallest values of k where RP is better on its own. However, in practice, one rarely uses such extreme dimension reduction.

V. Conclusions

In this study we presented a decentralized algorithm for selecting a good random projection to be used to reduce the dimensionality of a machine learning problem. We also proposed a hybrid approach which combines random projection selection with a decentralized SVD solver from related work. We evaluated these algorithms over a real smartphone trace over three machine learning data sets. The simulations assumed that we use only phones that are on a charger and that have a bandwidth of at least 1 Mbit/s, thus we took into account the energy problem in mobile computing.

We conclude that the proposed random projection selection algorithm is very fast and efficient, but the quality of the dimension reduction is somewhat lower than that of SVD. However, the SVD algorithm converges in a time proportional to the original dimensionality of the problem, which can be quite slow. Our hybrid approach combines the advantages of the two approaches and (assuming the same communication cost) it can provide a good quality dimension reduction, independently of the time available for convergence.

References

[1] A. Rial and G. Danezis, "Privacy-preserving smart metering," in *Proc. 10th annual ACM workshop on Privacy in the electronic society (WPES'11)*. ACM, 2011, pp. 49–60.

[2] A. S. Pentland, "Society's nervous system: Building effective government, energy, and public health systems," *Computer*, vol. 45, no. 1, pp. 31–38, 2012.

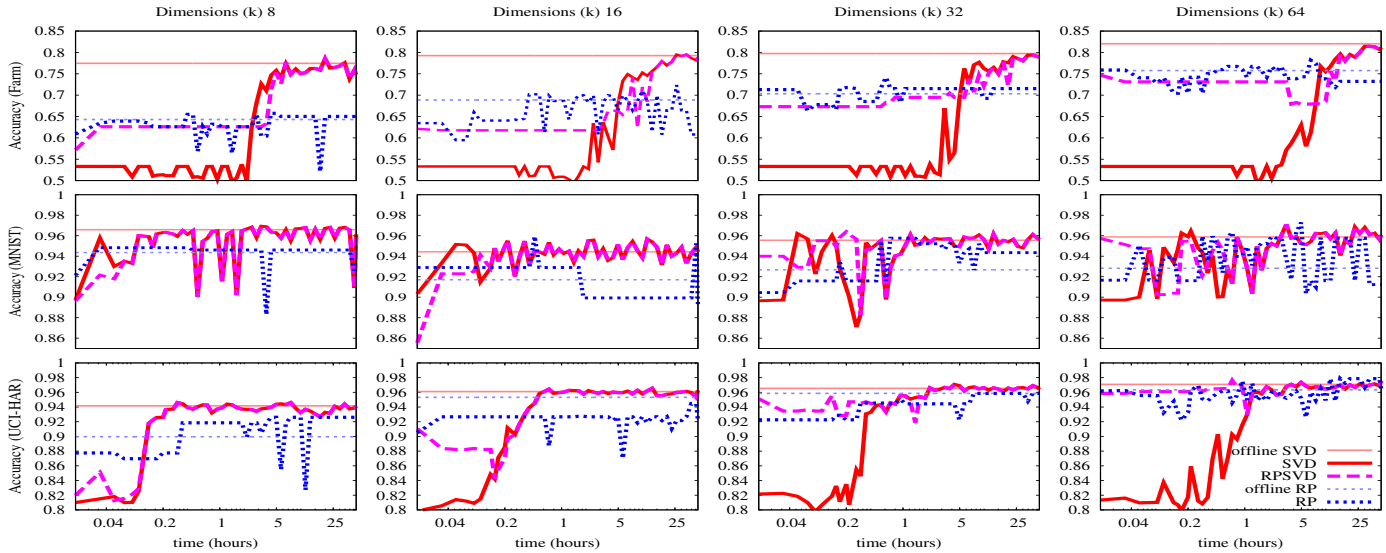

Fig. 2. Experimental results showing prediction accuracy as it evolves in time (time is on a logarithmic scale).

[3] C.-W. Tsai, C.-F. Lai, M.-C. Chiang, and L. Yang, "Data mining for internet of things: A survey," *Communications Surveys Tutorials, IEEE*, vol. 16, no. 1, pp. 77–97, 2014.

[4] R. Ormándi, I. Hegedűs, and M. Jelasity, "Gossip learning with linear models on fully distributed data," *Concurrency and Computation: Practice and Experience*, vol. 25, no. 4, pp. 556–571, 2013.

[5] A. Sayed, "Adaptation, learning, and optimization over networks," *Found. Trends Mach. Learn.*, vol. 7, no. 4-5, pp. 311–801, 2014.

[6] C. Dwork, "A firm foundation for private data analysis," *Commun. ACM*, vol. 54, no. 1, pp. 86–95, 2011.

[7] S. Song, K. Chaudhuri, and A. D. Sarwate, "Stochastic gradient descent with differentially private updates," in *IEEE Global Conf. on Signal and Information Processing (GlobalSIP)*, 2013, pp. 245–248.

[8] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proc. 1st MCC Workshop on Mobile Cloud Computing*, ser. MCC '12. ACM, 2012, pp. 13–16.

[9] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, pp. 1157–1182, 2003.

[10] P. Magdalinos, "Linear and non linear dimensionality reduction for distributed knowledge discovery," Ph.D. dissertation, Athens University of Economics and Business, Greece, 2010.

[11] J. Kubica, S. Singh, and D. Sorokina, "Parallel large-scale feature selection," in *Scaling up Machine Learning: Parallel and Distr. Approaches*, R. Bekkerman, M. Bilenko, and J. Langford, Eds. CUP, 2011.

[12] P. Magdalinos, C. Doulkeridis, and M. Vazirgiannis, "Enhancing clustering quality through landmark-based dimensionality reduction," *ACM Trans. Knowl. Discov. Data*, vol. 5, no. 2, pp. 11:1–11:44, 2011.

[13] I. T. Jolliffe, *Principal Component Analysis*, 2nd ed., ser. Springer Series in Statistics. Springer-Verlag, 2002.

[14] I. Hegedűs, M. Jelasity, L. Kocsis, and A. A. Benczúr, "Fully distributed robust singular value decomposition," in *Proc. 14th IEEE Intl. Conf. on Peer-to-Peer Comp. (P2P 2014)*. IEEE, 2014.

[15] M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, and M. van Steen, "Gossip-based peer sampling," *ACM Transactions on Computer Systems*, vol. 25, no. 3, p. 8, 2007.

[16] R. Roverso, J. Dowling, and M. Jelasity, "Through the wormhole: Low cost, fresh peer sampling for the internet," in *Proc. 13th IEEE Intl. Conf. on Peer-to-Peer Comp. (P2P 2013)*. IEEE, 2013.

[17] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[18] E. Bingham and H. Mannila, "Random projection in dimensionality reduction: Applications to image and text data," in *Proc. 7th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, ser. KDD '01. ACM, 2001, pp. 245–250.

[19] D. Achlioptas, "Database-friendly random projections: Johnson-lindenstrauss with binary coins," *J. Comput. Syst. Sci.*, vol. 66, no. 4, pp. 671–687, 2003.

[20] Á. Berta, V. Bilicki, and M. Jelasity, "Defining and understanding smartphone churn over the internet: a measurement study," in *Proc. 14th IEEE Intl. Conf. on Peer-to-Peer Comp. (P2P 2014)*. IEEE, 2014.

[21] Y. Lecun, C. Cortes, and B. Christopher, J.C., "The MNIST database of handwritten digits." [Online]. Available: http://yann.lecun.com/exdb/mnist/

[22] K. Bache and M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml