Learning and Data Selection in Big Datasets

Hossein S. Ghadikolaei [1] **Hadi Ghauch** [1 2] **Carlo Fischione** [1] **Mikael Skoglund** [1]

Abstract

Finding a dataset of minimal cardinality to characterize the optimal parameters of a model is of paramount importance in machine learning and distributed optimization over a network. This paper investigates the compressibility of large datasets. More specifically, we propose a framework that jointly learns the input-output mapping as well as the most representative samples of the dataset (sufficient dataset). Our analytical results show that the cardinality of the sufficient dataset increases sub-linearly with respect to the original dataset size. Numerical evaluations of real datasets reveal a large compressibility, up to 95%, without a noticeable drop in the learnability performance, measured by the generalization error.

1. Introduction

During the last decade, new artificial intelligence methods have offered outstanding prediction performance on complex tasks including face and speech recognition (Zhao et al., 2003; Schalkwyk et al., 2010; Hinton et al., 2012), autonomous driving (Michels et al., 2005), and medicine (Kourou et al., 2015). To achieve such amazing results, state-of-the-art machine learning methods often need to be trained on increasingly large datasets. For example, (MNIST) is a typical dataset for natural image processing of handwritten digits with more than 70,000 samples, and (MovieLens) 20M is a typical dataset for recommendation systems that includes more than 20,000,000 ratings. As we show throughout this paper, most of the samples in these datasets are redundant, carrying almost no additional information for the learning task. A fundamental open question in learning theory is how to characterize and algorithmically identify a small set of critical samples, hereafter called a *small representative dataset*, that best describes an unknown model. Studying its behavior around those critical samples helps us to better understand the unknown model as well as the inefficiencies of the sample acquisition process in the original dataset. For instance, in a multi-agent system, it may be enough to share these small representative datasets among the agents instead of the original large ones, leading to a significant reduction in power consumption and required communication bandwidth (Jiang et al., 2018).

Experiment design (Sacks et al., 1989) or active learning (Settles, 2012) provides algorithmic approaches to obtain a minimal set of samples to be labeled by an "oracle" (e.g., a human annotator). Active learning is well-motivated in many modern machine learning applications where obtaining a new labeled training sample is expensive. The main components of active learning are a parameterized model, a measure of the model's uncertainty, and an acquisition function that decides based on the model's uncertainty the next sample to be labeled. This approach has several challenges including lack of scalability to high-dimensional data (Settles, 2012) (which has been partially addressed in some recent publications (Gal et al., 2017)), lack of theoretical guarantees, and lack of formal uncertainty measure. More importantly, the acquisition function is usually greedy in the sense that it sequentially finds the new samples to be labeled one-by-one. Consequently, the resulting sub-sampled dataset may not necessarily be a small representative dataset due to the greedy nature of active learning.

Core-set selection (Tsang et al., 2005) takes a different approach than active learning. Instead of reducing the total labeling cost (like active learning), it focuses on finding a small representative set of samples with cardinality K in a big dataset of labeled samples with cardinality $N(\gg K)$? The existing results, however, are fairly limited to particular learning algorithms, like support-vector machines. Koh & Liang (2017) proposed a new approach to quantify the importance/influence of every sample of the training dataset in a generic learning task. This approach, however, computes the influence of the training samples one-by-one in isolation, failing to capture the joint effects of the samples.

In this paper, we investigate the core-set selection problem and use optimization theory to establish a scalable algorithm with provable theoretical guarantees to jointly find K most representative samples. We also show fundamental relations

[1]School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Stockholm, Sweden [2]COMELEC Department, Telecom ParisTech, Paris, France. Correspondence to: Hossein S. Ghadikolaei <hshokri@kth.se>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

between K and N to guarantee any arbitrary learning performance. Our framework and algorithmic solution approach can be very useful tools to better understand compressibility of the existing datasets and to improve distributed learning and edge intelligence; see Section 2.4. Moreover, further analysis of the representative dataset with respect to the original big dataset of an unknown model helps understand the sources of inefficiency in the current sampling process and how to improve it for other similar models.

The fundamental research question of this work, and in particular our order analysis, is closely related to the sample complexity (Clarkson et al., 2012), information-theoretic concepts of sampling (Jerri, 1977) like the Shannon-Nyquist sampling, compression (Cover & Thomas, 2012), and compressive sensing (Donoho, 2006) when the function is sparse in some predetermined basis. All these methods address the following question: how many samples are required to reconstruct a function with a predefined error? We show that the size of such a compressed dataset grows sub-linearly with respect to the cardinality of the original dataset.

In this study, we investigate compressibility of large datasets and develop a general framework for function approximation in which choosing the samples that best describe the function is done jointly with learning the function itself. We formulate a corresponding mixed integer non-linear program, and propose an iterative algorithm that alternates between a data selection step and a function approximation step. We show the convergence of the proposed algorithm to a stationary point of the problem, despite the combinatorial nature of the learning task. We then demonstrate that our algorithm outputs a small dataset of carefully chosen samples that solves a learning task, as accurately as if it were solved using original large dataset. Comprehensive numerical analyses on synthetic and real datasets reveal that our algorithm can significantly compress the datasets, by as much as 95%, with almost no noticeable penalty in the learning performance.

The rest of the paper is organized as follows. Section 2 presents the problem setting and our algorithmic solution approach. Section 3 provides main theoretical results. We apply our algorithms on synthetic and real datasets in Section 4, and then conclude the paper in Section 5. Due to lack of space, we have moved all the proofs and some applications to the appendix.

Notation: Normal font a or A, bold font $\boldsymbol{a}$, and calligraphic font $\mathcal{A}$ denote scalar, vector, and set, respectively. $|\mathcal{A}|$ is the cardinality of set $\mathcal{A}$. $\mathbb{I}$ is indicator function. $\boldsymbol{a}^T$ denotes the transpose of $\boldsymbol{a}$, and $\|\boldsymbol{a}\|_0$ and $\|\boldsymbol{a}\|_2$ are its l_0 and l_2 norms. $\mathbf{1}$ is a vector of all ones of proper size. For any integer N, $[N]$ denotes set $\{1, 2, \ldots, N\}$.

2. Setting and Solution Approach

2.1. Problem Setting

Consider input space $\mathcal{X}$, output space $\mathcal{Y}$, an *unknown* function $f : \mathcal{X} \to \mathcal{Y}$ from some function space $\mathcal{F}$, an index set $[N] := \{1, \ldots, N\}$ for $N \in \mathbb{N}$, and a dataset of training samples $\mathcal{D} = \{(\boldsymbol{x}_i, f(\boldsymbol{x}_i))\}_{i\in[N]}$, where $\boldsymbol{x}_i \in \mathcal{X}$. In a classical regression problem, we use the dataset $\mathcal{D}$ to learn f, namely find a function $h : \mathcal{X} \to \mathcal{Y}$ that has a minimal distance (for some distance measure, also called loss) to the true function f. Formally, consider loss function $\ell : \mathcal{X} \times \mathcal{Y} \times \mathcal{Y} \to [0, \infty]$, and let $\ell_i(h) := \ell(\boldsymbol{x}_i, f(\boldsymbol{x}_i), h(\boldsymbol{x}_i))$ denote the loss corresponding to sample $(\boldsymbol{x}_i, f(\boldsymbol{x}_i))$. The regression task solves the following empirical risk minimization problem:

$$(P1): \quad h^\star \in \arg\min_{h\in\mathcal{F}} \frac{1}{N} \sum\nolimits_{i\in[N]} \ell_i(h) \tag{1}$$

Here, we assume that $h \in \mathcal{F}$. However, considering a different function class for h would not change the generality of our results.

In many applications (see Section 2.4) one might not want to work with the entire dataset $\mathcal{D}$, e.g., due to its large size, but rather with a small subset $\mathcal{E} \subseteq \mathcal{D}$, where possibly $|\mathcal{E}| \ll |\mathcal{D}|$. We associate a binary variable z_i with each training sample $(\boldsymbol{x}_i, f(\boldsymbol{x}_i))$ such that $z_i = \mathbb{I}\{(\boldsymbol{x}_i, f(\boldsymbol{x}_i)) \in \mathcal{E}\}$, representing sample $(\boldsymbol{x}_i, f(\boldsymbol{x}_i))$ being selected or dropped. Letting $\boldsymbol{z} = [z_1, \cdots, z_N]^T$, the novel problem of *jointly* learning h and selecting $\mathcal{E}$ can be formulated as

$$(P2): \arg\min_{h\in\mathcal{F},\boldsymbol{z}} \; g(h, \boldsymbol{z}) := \frac{1}{\mathbf{1}^T\boldsymbol{z}} \sum\nolimits_{i\in[N]} z_i\ell_i(h) \tag{2a}$$

$$\text{s.t.} \;\; g_1(h) := \frac{1}{N} \sum\nolimits_{i\in[N]} \ell_i(h) \le \epsilon\,, \tag{2b}$$

$$g_2(\boldsymbol{z}) := \mathbf{1}^T\boldsymbol{z} \ge K\,,\; \boldsymbol{z} \in \{0, 1\}^N, \tag{2c}$$

where constraint (2b) prevents overfitting when "generalizing" from $\mathcal{E}$ to $\mathcal{D}$, and constraint (2c) prevents degenerate/trivial solutions to the problem (e.g., where $\mathcal{E}$ empty). We show later that K is a very important parameter that trades off the compression rate, defined as $1 - |\mathcal{E}|/|D|$, and the generalization error of learning with $\mathcal{E}$.

Followings are some assumptions used throughout the paper, which are prevalent in the learning literature.

Assumption 1. *$\ell_i(h)$ is continuous and convex in h.*

Assumption 2. *The original dataset $\mathcal{D}$ is duplicate-free, namely $\boldsymbol{x}_m \neq \boldsymbol{x}_n$ for all $m, n \in [N]$ and $\boldsymbol{x}_m, \boldsymbol{x}_n \in \mathcal{X}$. Moreover, for all $h \in \mathcal{F}$, $\boldsymbol{x}_m \neq \boldsymbol{x}_n$ implies $\ell_m(h) \neq \ell_n(h)$.*

Assumption 3. *For some sufficiently small $\epsilon > 0$, $\exists h \in \mathcal{F}$ for which* (2b) *is feasible.*

Note that $\ell_i(h)$ does not have to be smooth and differentiable in general. We stress the existence of a large family of loss functions, including L^p spaces, for which both assumptions hold, as exemplified in Appendix B. Moreover, the convexity in Assumption 1 may also be relaxed at the expense of a weaker convergence property using the block successive upper-bound minimization framework (Razaviyayn et al., 2013). Finally, if the dataset contains some duplicated samples, we can add insignificant perturbations to satisfy Assumption 2, without affecting the information content (Bertsekas, 1998). Assumption 3 implies that for some small ϵ, there is a feasible point for optimization problem $(P2)$. This is a natural assumption, as we may not be able to improve the accuracy of the training by sub-sampling the dataset. To choose ϵ, one may run $(P1)$, find the minimal training error, and set ϵ in $(P2)$ to be within a tolerable distance of the minimal error. The higher the distance, the more the achievable compression level, as shown in the numerical results.

2.2. Solution Approach

$(P2)$ is a non-convex combinatorial optimization problem with coupling cost and constraint functions. In the following, we provide a solution approach based on block-coordinate descent (BCD), which splits $(P2)$ into two subproblems: $(P2a)$ for data selection and $(P2b)$ for function approximation. Let $h^{(k)}$ and $\boldsymbol{z}^{(k)}$ be the value of h and $\boldsymbol{z}$ at iteration k. BCD yields the following update rules:

$$(P2a): \quad \boldsymbol{z}^{(k+1)} \in \underset{\boldsymbol{z}\in\{0,1\}^N}{\arg\min} \; g(h^{(k)}, \boldsymbol{z}), \tag{3a}$$

$$\text{s.t. } g_2(\boldsymbol{z}) \geq K\,, \tag{3b}$$

and

$$(P2b): \quad h^{(k+1)} \in \underset{h\in\mathcal{F}}{\arg\min} \; g(h, \boldsymbol{z}^{(k+1)}), \tag{4a}$$

$$\text{s.t. } g_1(h) \leq \epsilon\,. \tag{4b}$$

The data selection (D-)step $(P2a)$ is optimized for a given hypothesis h. Then, in the function approximation (F-)step $(P2b)$, the hypothesis is optimized for the updated compressed dataset $\boldsymbol{z}^{(k+1)}$. Next, we derive solutions to each step.

2.2.1. D-STEP

Given $h^{(k)}$, namely the value of h at iteration k, we let $c_i^{(k)} = \ell(\boldsymbol{x}_i, f(\boldsymbol{x}_i), h^{(k)}(\boldsymbol{x}_i))$ and $\boldsymbol{c}^{(k)} = [c_1^{(k)}, c_2^{(k)}, \ldots, c_N^{(k)}]$, for notation simplicity. Then, the first subproblem is written as

$$\arg\min_{\boldsymbol{z}\in\{0,1\}^N} \; \boldsymbol{z}^T\boldsymbol{c}^{(k)}/\mathbf{1}^T\boldsymbol{z}\,, \text{ s.t. } \boldsymbol{z}^T\mathbf{1} = K, \tag{5}$$

where we have used the fact that $\boldsymbol{z}^T\mathbf{1} \geq K$ holds with equality; shown in Proposition 1. Thus, $\mathbf{1}^T\boldsymbol{z} = K$ can be removed from the denominator to *equivalently*[1] write (5) as

$$(P2a): \; \boldsymbol{z}^{(k+1)} \in \arg\min_{\boldsymbol{z}\in\{0,1\}^N} \; \boldsymbol{z}^T\boldsymbol{c}^{(k)}\,, \text{ s.t. } \boldsymbol{z}^T\mathbf{1} = K\,. \tag{6}$$

Though combinatorial, the form in $(P2a)$ allows to easily derive its solution, as shown by Proposition 1, i.e., $z_i^{(k+1)} = \mathbb{I}\{i \in \mathcal{S}^{(k)}\}$, where $\mathcal{S}^{(k)}$ is the set of K-indices in $\boldsymbol{c}^{(k)}$ with the smallest values. In other words, the optimal solution is obtained by selecting the smallest K elements of $\boldsymbol{c}^{(k)}$, and setting the corresponding indices of $\boldsymbol{z}$ to 1.

2.2.2. F-STEP

Given the updated data selection, $\boldsymbol{z}^{(k+1)}$, we use the fact that $\mathbf{1}^T\boldsymbol{z}^{(k+1)} = K$ and rewrite $(P2b)$ as

$$(P2b): \quad \underset{h\in\mathcal{F}}{\arg\min} \; g(h) := \sum\nolimits_{i=1}^{N} z_i^{(k+1)}\ell_i(h) \tag{7a}$$

$$\text{s.t. } g_1(h) := \sum\nolimits_{i=1}^{N} \ell_i(h) \leq \epsilon\,. \tag{7b}$$

It follows from $z_i^{(k+1)} \in \{0,1\}$ that both the cost and constraints in $(P2b)$ consist of convex combinations of the loss, ℓ_i, assumed convex (Assumption 1): Thus, $(P2b)$ is convex in h (Boyd & Vandenberghe, 2004, Chap. 3). In the following section, we show that there exist loss functions that lead to closed-form solutions.

Algorithm 1 summarizes our alternating data selection and function approximation steps. We establish the algorithm's convergence to a stationary point of $(P2)$ in Proposition 2.

Algorithm 1 Alternating Data Selection and Function Approximation (DF)

Initialize: $\boldsymbol{z}^{(1)} = \mathbf{1}$
for $k = 1, 2, 3, \ldots$ **do**
  // F-step
  Update $h^{(k+1)}$ by solving $(P2b)$ in (7)
  // D-step
  Compute $c_i^{(k)}$, $\forall i \in [N]$
  Update $\boldsymbol{z}^{(k+1)}$ by solving $(P2a)$ in (5), using Proposition 1
  Break if $|g(h^{(k+1)}, \boldsymbol{z}^{(k+1)}) - g(h^{(k)}, \boldsymbol{z}^{(k)})| < \gamma$
end for
Return: $h^{(k+1)}$ and $\boldsymbol{z}^{(k+1)}$

2.3. Special Cases

2.3.1. LINEAR REGRESSION AND DATA SELECTION

We specialize our approach to a linear regression problem with data selection where $h(\boldsymbol{x}_i) = \boldsymbol{x}_i^T\boldsymbol{w}$ for $\boldsymbol{w}, \boldsymbol{x}_i \in \mathbb{R}^d$, d

[1] Two problems are equivalent, if the optimal solution to one can be obtained from the other, and vice-versa (Boyd & Vandenberghe, 2004).

being the dimension of each sample. Optimization problem $(P2)$ reduces to

$$\begin{aligned}\underset{\boldsymbol{w},\boldsymbol{z}}{\arg\min}\ & \sum\nolimits_{i\in[N]} \frac{z_i}{\mathbf{1}^T\boldsymbol{z}} \left(\boldsymbol{x}_i^T\boldsymbol{w} - f(\boldsymbol{x}_i)\right)^2 \\ \text{s.t.}\ & \frac{1}{N}\sum\nolimits_{i\in[N]} \left(\boldsymbol{x}_i^T\boldsymbol{w} - f(\boldsymbol{x}_i)\right)^2 \le \epsilon\,, \\ & \mathbf{1}^T\boldsymbol{z} \ge K\,,\ \boldsymbol{z}\in\{0,1\}^N\,.\end{aligned}$$

The D-step is identical to $(P2a)$, which can be solved by Proposition 1. Given $\boldsymbol{z}^{(k+1)}$, the F-step reduces to the following quadratically-constrained quadratic programming:

$$\begin{aligned}\boldsymbol{w}^{(k+1)} = \underset{\boldsymbol{w}}{\arg\min}\ & \left\|\boldsymbol{A}^{(k+1)}\left(\boldsymbol{X}^T\boldsymbol{w} - f(\boldsymbol{X})\right)\right\|_2^2\,, \\ \text{s.t.}\ & \left\|\boldsymbol{X}^T\boldsymbol{w} - f(\boldsymbol{X})\right\|_2^2 \le \epsilon\,,\end{aligned}$$

where $\boldsymbol{A}^{(k+1)} := \text{diag}\Big(\sqrt{z_1^{(k+1)}},\cdots,\sqrt{z_N^{(k+1)}}\Big)$, $\boldsymbol{X} := [\boldsymbol{x}_1,\cdots,\boldsymbol{x}_N]$, and $f(\boldsymbol{X}) := [f(\boldsymbol{x}_1),\ldots,f(\boldsymbol{x}_N)]^T$. The problem is convex and can be solved using standard Lagrangian techniques to yield

$$\begin{aligned}\boldsymbol{w}^{(k+1)} = & \left(\boldsymbol{X}\left(\left(\boldsymbol{A}^{(k+1)}\right)^2 + \lambda\boldsymbol{I}_N\right)\boldsymbol{X}^T\right)^{-1}\boldsymbol{X} \\ & \times\left(\boldsymbol{A}^{(k+1)} + \lambda\boldsymbol{I}_N\right) f(\boldsymbol{X})\,,\end{aligned}$$

where $\lambda \ge 0$ is a Lagrange multiplier that satisfies the complementary slackness condition and can be found using 1D search methods.

Note that computational complexity of $\boldsymbol{w}^{(k+1)}$ is dominated by the matrix inversion, which is in the order of $\mathcal{O}(d^3)$ and does not scale with the size of the training set, N. However, if it is still considered significant in some applications, $\boldsymbol{w}^{(k+1)}$ can be obtained using stochastic gradient-based methods whose computational complexity is $\mathcal{O}(d/\alpha)$ for accuracy threshold $\alpha > 0$.

2.3.2. Robust Learning

Consider the following continuous relaxation of $(P2)$, where $\boldsymbol{z}\in\{0,1\}^N$ is relaxed to $\boldsymbol{z}\in[0\ ,\ 1]^N$,

$$\begin{aligned}(P3):\quad \underset{h\in\mathcal{F},\boldsymbol{z}}{\arg\min}\ & \frac{1}{\mathbf{1}^T\boldsymbol{z}}\sum\nolimits_{i\in[N]} z_i\,\ell_i(h) \\ \text{s.t.}\ & \frac{1}{N}\sum\nolimits_{i\in[N]}\ell_i(h) \le \epsilon, \\ & \mathbf{1}^T\boldsymbol{z} \ge K, \\ & \boldsymbol{z}\in[0,1]^N\,.\end{aligned}$$

Thus, z_i can be seen as a non-negative weight assigned to sample $(\boldsymbol{x}_i, f(\boldsymbol{x}_i))$ in that training set, representing level of confidence in its quality (higher values of z_i imply better qualities). From this perspective, the resulting problem becomes a robust learning problem, in presence of non-uniform sample quality: learning h, jointly with the *best samples* of the training set. Some applications include denoising sensor measurements and outlier detection.

We can use Algorithm 1 to address $(P3)$, with a minor modification in the D-step. Note that relaxing the binary constraint implies that the D-step cannot be solved using the simple method of Proposition 1. However, we use the linear program relaxation of $(P2a)$,

$$\arg\min\nolimits_{\boldsymbol{z}\in[0,1]^N}\ \boldsymbol{z}^T\boldsymbol{c}^{(k)}\ \text{, s.t. }\ \mathbf{1}^T\boldsymbol{z} = K\,, \tag{8}$$

which can be efficiently solved using standard linear program solvers. In Appendix A.6, we have shown that optimization problem (8) is equivalent to $(P2a)$, and therefore the linear program relaxation is optimal.

2.4. Applications

2.4.1. Multi-agent Systems and Internet-of-Things

Consider a network of agents with some abstract form of very limited networking capacity (the so-called communication-limited networks in optimization and control literature (Smith et al., 2018; Magnússon et al., 2018; Nedic & Bertsekas, 2001; Tsitsiklis & Luo, 1987)). The limitation can be, among others, due to low-power operation of agents (sensor nodes) in Internet-of-Things or low channel capacity in harsh wireless environments, like intra-body networks. Consequently, the data rate among various agents is very limited. Each agent has a local dataset, e.g., some measurements, and they should all share the datasets to jointly run an optimization/learning task within a short deadline (low-latency scenario). Due to the tight deadline and limited communication capability, we cannot send all entries of the datasets to all other agents by the deadline. The central question of this subsection is to decide, locally at every agent, which data should be shared (data selection). To characterize this question, one may consider the existence of an oracle that has access to all the datasets, which finds the minimal representative dataset and then informs all the agents what to share. Clearly, this oracle is not practical: rather it gives a theoretical benchmark on the performance of various solution approaches and on the cardinality of the minimal representative dataset. $(P2)$ models the oracle, and our algorithmic solution approach characterizes its solution. We can also get useful insights on the "optimal" sub-sampling per agent, which can be exploited to develop practical solution approaches for the central question of this subsection. Therefore, our results are of paramount importance to the problem of low-latency learning and inference over a communication-limited network (Jiang et al., 2018).

2.4.2. EDGE COMPUTING

Consider an edge computing scenario where some computations are offloaded to the edge devices. Edge devices can represent some smart meters in smart grids or a camera in a surveillance system. Edge computing enables intelligence closer to the data collections sources (Shi et al., 2016). As a result, it substantially reduces the communications bandwidth by sending only the locally processed data (decisions) instead of raw data (measurements) to the central cloud servers. Assume that the local objective functions need some input parameters $\boldsymbol{\theta}$, given by the cloud. Also consider a natural assumption of a time evolution for $\boldsymbol{\theta}$, determined in part by the measurements from all local edge devices. A prime example is energy pricing in a smart grid, $\boldsymbol{\theta}$, determined in part by the energy consumption of every household (local measurements). To enable edge computing, local devices should frequently send their raw data to the cloud. Thus, $\boldsymbol{\theta}$ will be updated and broadcasted, and then local devices run optimization problems like $(P1)$ with the updated $\boldsymbol{\theta}$. Each device can run $(P2)$ to find a small representative dataset and send that to the cloud, instead of the entire dataset. Our numerical results indicate that some real-world datasets can be compressed as much as 95% without any noticeable drop in the learning performance. This way, we can increase the frequency of re-adjusting $\boldsymbol{\theta}$ with a fixed communication bandwidth budget.

3. Main Theoretical Results

In this section, we present main results of this paper. Detailed proofs are available in Appendix.

Proposition 1 (Solution of $(P2a)$)**.** *Under Assumption 2, we define an index sequence j such that for any $j_m, j_n \in [N]$, $c_{j_m}^{(k)} < c_{j_n}^{(k)}$ iff $m < n$, where $c_i^{(k)}$ is defined in Section 2.2.1. The solution of the data selection subproblem is*

$$z_i^{(k+1)} = \begin{cases} 1, \text{ if } & i = j_1, j_2, \dots, j_K \\ 0, \text{ if } & i = j_{K+1}, \dots, j_N \ , \end{cases} \tag{9}$$

and $\|\boldsymbol{z}^{(k+1)}\|_0 = K$.

Proposition 1 implies that the optimal solution to the binary data selection subproblem is simple: evaluate the loss function for all training samples of $\mathcal{D}$ using $h^{(k)}$, sort the values, and keep K data samples having the smallest losses. Next, we establish the convergence of our BCD-based algorithm to a stationary point of $(P2)$.

Proposition 2 (Convergence)**.** *Let $\{g(h^{(k)}, \boldsymbol{z}^{(k)})\}_{k\geq 1}$ denote the sequence generated by the BCD updates in Algorithm 1. Then, this sequence monotonically decreases with each update, i.e., for $k \in \mathbb{N}$*

$$g(h^{(k)}, \boldsymbol{z}^{(k)}) \geq g(h^{(k)}, \boldsymbol{z}^{(k+1)}) \geq g(h^{(k+1)}, \boldsymbol{z}^{(k+1)})$$

and converges to a stationary point of $(P2)$.

Proposition 3 (Computational Complexity)**.** *The D-step is run in $\mathcal{O}(N)$, and the F-step has the same complexity as of $(P1)$, per iteration of Algorithm 1.*

To analyze the asymptotic behavior of our approach, we make additional assumptions on the class of loss functions, and on $\mathcal{F}$. In particular, we assume that ℓ belongs to the L^p space and $\mathcal{F}$ is the space of L-Lipschitz functions defined on some compact support.

Proposition 4 (Sample Complexity)**.** *Assume that the samples are noiseless, and $\mathcal{F}$ is the set of L-Lipschitz functions defined on interval $[0, T]^d$. Consider optimization problem $(P2)$. Let $g(h^\star, \boldsymbol{z}^\star) := \left(\mathbf{1}^T \boldsymbol{z}^\star\right)^{-1} \sum_{i\in[N]} z_i^\star \ell(\boldsymbol{x}_i, f(\boldsymbol{x}_i), h^\star(\boldsymbol{x}_i))$. For any arbitrary constant $\delta > 0$, the following holds: When $\ell(\boldsymbol{x}_m, f(\boldsymbol{x}_m), h^\star(\boldsymbol{x}_m)) := |f(\boldsymbol{x}_m) - h^\star(\boldsymbol{x}_m)|^2$, $g(h^\star, \boldsymbol{z}^\star) \leq \delta$ if $K \geq \lceil (1 + 2LT\sqrt{d/\delta})^d \rceil$, where $\lceil\cdot\rceil$ is the ceiling function.*

Corollary 1 (Asymptotic Sample Complexity)**.** *Consider the assumptions of Proposition 4. Define compression ratio as $\mathsf{CR} := 1 - K/N$. As N grows large:*

$$\forall \delta > 0, \exists K \leq N \ \text{ s.t. } \ g(h^\star, \boldsymbol{z}^\star) \leq \delta \ , \ \text{and } \mathsf{CR} \to 1 \ .$$

Proposition 4 and Corollary 1 imply that the sufficient dataset $\mathcal{E}^\star$ (which can be used to learn any function f in class $\mathcal{F}$ with any arbitrary accuracy) is the output of a sublinear sub-sampler (our Algorithm 1) of the original big dataset, namely $K/N \to 0$ asymptotically. In other words, as we experimentally show in the next section, most of the existing big datasets are highly redundant, and the redundancy brings almost no additional gain for the accuracy of the learning task. Finding this sufficient dataset is of manageable complexity, as specified in Proposition 3. It is worth noting that from the feasibility constraint of $(P2)$, $\delta \leq \epsilon$. One may observe that if the original problem has no feasible solution for some ϵ, the performance bounds may not hold. However, the additional assumptions of Proposition 4 ensures the feasibility of $(P2)$ for large enough N.

Moreover, notice that our theoretical results go beyond convex functions. Proposition 2) is valid for quasi-convex or invex functions. For multi-convex structure (like a deep neural network training optimization problem), the block successive upper-bound minimization framework (Razaviyayn et al., 2013) under mild condition can ensure Proposition 2, see Corollary 1 of (Razaviyayn et al., 2013) and also (Xu & Yin, 2013).

4. Experimental Results

In this section, we first present two toy examples to illustrate $(P2)$ and algorithmic solution. We then focus on real databases and evaluate the effectiveness of our approach on finding the small representative dataset with a negligible loss in the generalization capability.

4.1. Experimental Setting

In every experiment, we select the input space $\mathcal{X}$, output space $\mathcal{Y}$, mapping f, training dataset $\mathcal{D}$, test dataset $\mathcal{T}$, and hypothesis class $\mathcal{H}$. We then run Algorithm 1 to find the optimal compressed dataset $\mathcal{E}^\star \subseteq \mathcal{D}$. We run this experiment for different values of CR.

To evaluate the true generalization capability of $\mathcal{E}^\star$, we run a conventional regression problem $(P1)$ using both $\mathcal{D}$ and $\mathcal{E}^\star$, find the corresponding optimal approximation function, and then evaluate their accuracy on $\mathcal{T}$. We denote the normalized generalization error by $e(\mathcal{D}, \mathcal{T})$,

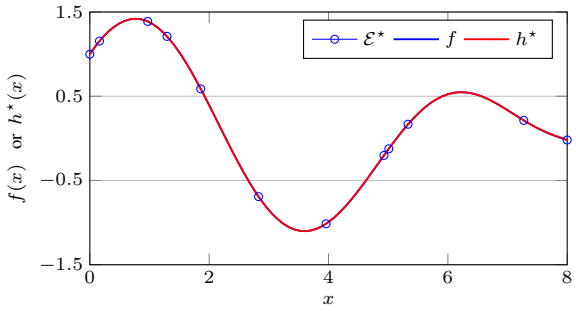
$$\frac{\sum_{i\in\mathcal{T}} \ell\left(\boldsymbol{x}_i, f(\boldsymbol{x}_i), h^\star(\boldsymbol{x}_i)\right)}{\sum_{i\in\mathcal{T}} \ell\left(\boldsymbol{x}_i, f(\boldsymbol{x}_i), 0\right)}, \tag{10}$$

where $h^\star$ is found by running $(P1)$ with $\mathcal{D}$. When we find $h^\star$ by running $(P1)$ with $\mathcal{E}^\star$, (10) shows $e(\mathcal{E}^\star, \mathcal{T})$. When ℓ is the L^2-norm, (10) reduces to the normalized square error, $\sum_{i\in\mathcal{T}} |f(\boldsymbol{x}_i) - h^\star(\boldsymbol{x}_i)|^2 / \sum_{i\in\mathcal{T}} |f(\boldsymbol{x}_i)|^2$. This normalization is to have a fair comparison of the generalization error over various test datasets, which may have different norms. We say that our compressed dataset $\mathcal{E}^\star$ performs as well as $\mathcal{D}$ when $e(\mathcal{E}^\star, \mathcal{T})$ is close to $e(\mathcal{D}, \mathcal{T})$. Throughout the following experimental studies, we observe that the lower the compression ratio, the lower the gap $|e(\mathcal{D}, \mathcal{T}) - e(\mathcal{E}^\star, \mathcal{T})|$. However, for big datasets, we can substantially compress the dataset without any noticeable drop in the gap, indicating a high inefficiency in the data generation process.

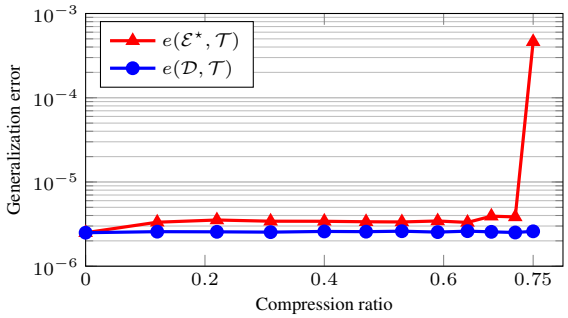
4.2. Illustrative Examples

In our first example, we pick a very smooth function f. Let $\mathcal{X} = [0, 8]$, $\mathcal{F} = \mathcal{H} = \mathrm{Poly}(10)$, where $\mathrm{Poly}(n)$ is polynomial functions of degree n.[2] $f(x)$ is given in figure 1(a). Our original dataset $\mathcal{D}$ is 100 equidistant samples in $\mathcal{X}$. We add i.i.d. Gaussian noise of standard deviation 0.05 to $f(x)$. Figure 1(a) shows an example of an optimal compressed dataset $\mathcal{E}^\star$ computed for $K = 12$, along with the learned hypothesis $h^\star$, which almost perfectly coincide with the true function f. Note that due to the random noise in $\mathcal{D}$, the selected samples would be different in every run. To evaluate the true generalization capability of our algorithm, we run a Monte Carlo simulation for 100 random realizations of the noise on dataset, find $\mathcal{E}^\star$ and $h^\star$ for each realization, and compute $e(\mathcal{D}, \mathcal{T})$ and $e(\mathcal{E}^\star, \mathcal{T})$ from (10). Note that $\mathcal{T}$ is the set of 1000 equidistant examples in $\mathcal{X}$. Figure 1(b) reports the average generalization error against the CR. As expected, the higher the compression ratio the higher the generalization error. The tail drop of this function is at least as a fast as a double exponential function of CR, implying that for a wide range of CR values, the extra generalization error $|e(\mathcal{E}^\star, \mathcal{T}) - e(\mathcal{D}, \mathcal{T})|$ is negligible. In particular, in the example of Figure 1(b) with $N = 100$, 70% compression leads to only 6×10^{-5} extra generalization error.

Figure 2 illustrates the performance of our optimization problem on a less smooth function than that of figure 1(a). In particular, we consider the function of figure 2(a) which is from $\mathcal{F} = \mathrm{Poly}(15)$, $\mathcal{X} = [0, 1]$, $N = 100$ equidistant samples from $\mathcal{X}$ in $\mathcal{D}$, and 1000 equidistant samples from $\mathcal{X}$ in $\mathcal{T}$. We observe a large compression of the dataset in figure 2(b), where only $|e(\mathcal{E}^\star, \mathcal{T}) - e(\mathcal{D}, \mathcal{T})| = 2.3 \times 10^{-5}$ extra true generalization error after 60% compression. Moreover, we can see the double exponential tail of $e(\mathcal{E}^\star, \mathcal{T})$, which implies that only a small dataset of a few carefully



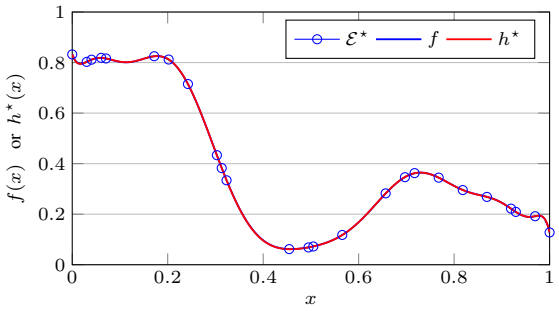
(a) Function f and an example of $\mathcal{E}^\star$ with $K = 12$



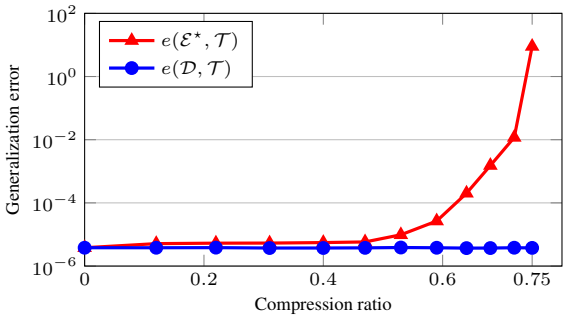
(b) Generalization error

Figure 1: Learning compressed data set $\mathcal{E}^\star$ and optimal hypothesis $h^\star$ with a dataset of size $N = 100$. Function $f \in \mathrm{Poly}(10) +$ noise. Selected samples in $\mathcal{E}^\star (\subset \mathcal{D})$ are denoted by circles. In the example of (a), f and $h^\star$ are visually indistinguishable. The higher the compression ratios the higher the generalization error. $e(\mathcal{E}^\star, \mathcal{T})$ behaves like a double exponential function of CR. For a wide range of CR values there is almost no loss compared to $e(\mathcal{D}, \mathcal{T})$.

[2] To be consistent with the theoretical results, we have added $\mathcal{F} = \mathcal{H}$ assumption. We have also experimented $\mathcal{F} = \mathrm{Poly}(10)$ and $\mathcal{H} = \mathrm{Poly}(15)$ and observed similar insights as of Figure 1.



(a) Function f and an example of $\mathcal{E}^\star$ with $K = 25$.



(b) Generalization error

Figure 2: Learning compressed data set $\mathcal{E}^\star$ and optimal hypothesis $h^\star$ with a dataset of size $N = 100$. Function $f \in \text{Poly}(15) + \text{noise}$. The higher the compression ratios the higher the generalization error. The double exponential behavior of CR, observed also in figure 1(b), is visible in (b).

chosen samples are enough to have a learning with a sufficiently good generalization capability. However, for a fixed error gap $|e(\mathcal{E}^\star, \mathcal{T}) - e(\mathcal{D}, \mathcal{T})|$, functions having more variation are less compressible, since more samples are needed to maintain the same error gap.

4.3. Real Datasets

Motivated by the excellent performance of the proposed algorithm on simple syntectic data, in this section, we apply Algorithm 1 on real databases listed in Table 1, available on Statlib (Sta) and UCI repositories (UCI). These databases have been extensively used in relevant machine learning and signal processing applications (Schapire, 1999; Aharon et al., 2006; Zhang & Li, 2010; Zhou et al., 2014; Tang et al., 2016; Chatterjee et al., 2017).

For the learning task, and without loss of generality, we use the recently proposed extreme learning machine (ELM) architecture (Huang et al., 2006; 2012), due to its implementation efficiency, good regression and classification performance, and convexity of the resulting optimization problem. An ELM typically uses a few hidden layers, each having many nodes, to project the input data vectors to high dimensional feature vectors. Then, a linear projection is used at

Table 1: Databases for regression task. d is the input dimension.

Database	# Training samples	# Test samples	d
Bodyfat	168	84	14
Housing	337	169	13
Space-ga	2,071	1,036	6
YearPredictionMSD	463,715	51,630	90
Power Consumption	1,556,445	518,814	9

the last layer to recover the output vector. An interesting property of ELM is the ability to use instances of random matrices in mapping to feature vectors (as opposed to usual deep neural networks in which these weight matrices should be optimized), and therefore we need to optimize only the weights of the last layer. In our implementation, we have used a single hidden layer, an instance of random matrix between input layer and hidden layer $\boldsymbol{R}$, an element-wise rectifier linear unit (ReLU) function at each hidden node that returns $\sigma(\cdot) = \max(\cdot, 0)$, weights to the output layer $\boldsymbol{W}$. Given dataset $\mathcal{D}$ with N samples and a binary selection vector $\boldsymbol{z}^{(k+1)}$, we use the following optimization problem in the F-step of Algorithm 1 at iterate k:

$$\boldsymbol{W}^{(k+1)} = \arg\min_{\boldsymbol{W}} \frac{\sum_{i\in[N]} z_i^{(k+1)} f_i(\boldsymbol{W})}{\mathbf{1}^T \boldsymbol{z}^{(k+1)}} + \lambda \|\boldsymbol{W}\|_2^2$$
$$\text{s.t. } \frac{1}{N} \sum_{i\in[N]} f_i(\boldsymbol{W}) \leq \epsilon \,,$$

where

$$f_i(\boldsymbol{W}) := \|f(\boldsymbol{x}_i) - \boldsymbol{W}\, \sigma(\boldsymbol{R}\boldsymbol{x}_i)\|_2^2 \,,$$

and the second term in the objective function is the Tikonov regularization with parameter λ that alleviates the overfitting problem. This convex quadratically constrained quadratic program can be efficiently solved by existing optimization toolboxes (Grant & Boyd, 2014). Due to the randomness in $\boldsymbol{R}$ and the chosen small dataset (in one benchmark), we repeat experiments 100 times and report the mean value and standard deviation of the performance results. We also consider two benchmarks: influence-based data selection and random data selection. To have a fair comparison, we first simulate our approach on a given dataset for various CR values (various K). In the first benchmark, we find the influence score (as a measure of their importance on the training process) of all the samples (Koh & Liang, 2017) and run $(P1)$ on a dataset of K samples with the highest scores. In the second benchmark, we randomly choose K samples of the training dataset and run $(P1)$ on those samples.

Table 2 shows the regression performance of various databases, given in Table 1. From this table, our approach can substantially compress the training datasets with a controlled loss on the generalization error. This compressibility

Table 2: Regression performance on real databases. The reported values are "average $\pm$ standard deviation" of the normalized true generalization error. "CR" stands for compression ratio. "Influence" and "Random" show influence-based and random data selection benchmarks. The values on row CR $= 0\%$ corresponds to $e(\mathcal{D}, \mathcal{T})$, and the performance is identical for all approaches.

CR	Algorithm	Bodyfat	Housing	Space-ga	YearPredictionMSD	Power Consumption
0%	Proposed	0.0245 ± 0.0051	0.0301 ± 0.0056	0.1323 ± 0.0134	0.0082 ± 0.0007	0.0142 ± 0.0008
25%	Proposed	0.0294 ± 0.0058	0.0345 ± 0.0071	0.1325 ± 0.0142	0.0083 ± 0.0007	0.0144 ± 0.0008
	Influence	0.0298 ± 0.0055	0.0345 ± 0.0077	0.1326 ± 0.0144	0.0083 ± 0.0007	0.0144 ± 0.0008
	Random	0.0315 ± 0.0081	0.0347 ± 0.0092	0.1329 ± 0.0145	0.0083 ± 0.0008	0.0145 ± 0.0008
50%	Proposed	0.0333 ± 0.0067	0.0348 ± 0.0080	0.1338 ± 0.0175	0.0085 ± 0.0008	0.0144 ± 0.0009
	Influence	0.0342 ± 0.0071	0.0349 ± 0.0078	0.1340 ± 0.0175	0.0087 ± 0.0008	0.0144 ± 0.0008
	Random	0.0370 ± 0.0102	0.0361 ± 0.0105	0.1412 ± 0.0202	0.0162 ± 0.0071	0.0190 ± 0.0124
75%	Ours	0.0360 ± 0.0060	0.0374 ± 0.0076	0.1351 ± 0.0169	0.0086 ± 0.0010	0.0145 ± 0.0008
	Influence	0.0383 ± 0.0067	0.0378 ± 0.0081	0.1519 ± 0.0231	0.0091 ± 0.0008	0.0146 ± 0.0008
	Random	0.0411 ± 0.0166	0.0392 ± 0.0099	0.1901 ± 0.0270	0.0312 ± 0.0156	0.0357 ± 0.0235
80%	Proposed	0.0417 ± 0.0077	0.0382 ± 0.0076	0.1354 ± 0.0171	0.0086 ± 0.0009	0.0145 ± 0.0008
	Influence	0.0446 ± 0.0097	0.0384 ± 0.0080	0.1535 ± 0.0240	0.0099 ± 0.0015	0.0151 ± 0.0011
	Random	0.0461 ± 0.0168	0.0395 ± 0.0107	0.2006 ± 0.0281	0.0370 ± 0.0171	0.0380 ± 0.0254
95%	Proposed	0.0630 ± 0.0134	0.0538 ± 0.0122	0.1386 ± 0.0217	0.0088 ± 0.0012	0.0153 ± 0.0011
	Influence	0.0753 ± 0.0101	0.0565 ± 0.0115	0.1951 ± 0.0329	0.0142 ± 0.0073	0.0216 ± 0.0009
	Random	0.0944 ± 0.0540	0.0744 ± 0.0372	0.3713 ± 0.0714	0.0560 ± 0.0478	0.0749 ± 0.0628

increases by the dataset size and decreases by the input dimension. For instance, 75% compression in Bodyfat results in a noticeable performance drop, while a big dataset like YearPredictionMSD can be compressed by 95% without a significant loss in the learning performance. Moreover, these results empirically show the sub-linear characteristic of the sufficient dataset for any fixed δ, namely $\mathsf{CR} \to 1$ as $N \to \infty$. The results suggest that a small set of carefully selected samples are enough to run the learning task. Notice that for the same compression ratio (same K), our proposed approach outperforms both benchmarks. The difference to the random selection benchmark is more prominent, especially at the higher compression rates. At those regimes, random selection may results in a set of good or completely bad samples, leading to almost unpredictable generalization errors. The influence-based score leads to a better sub-sampling of the training data, compared to the random selection, though its greedy nature (i.e., scoring/selecting samples one-by-one) may results in substantial performance drops at the high compression rates. Our approach, however, does not have this problem by selecting all the samples jointly. As we have discussed in Section 2.4, such extreme level of compressions could be inevitable to implement distributed learning over communication-constrained networks, e.g., Internet-of-Things and edge computing.

4.4. Further Discussions

In all the simulation experiments, we have observed a very fast convergence of the proposed Algorithm 1, usually after a few iterations in small datasets (e.g., Bodyfat) and a few tens of iterations in large datasets (e.g., Abalone) among D-step and F-step. Computational complexity of each step is characterized in Proposition 3. Both bigger datasets and larger $\binom{N}{K}$ correspond to larger search spaces and consequently slower convergence rate.

5. Conclusions

We addressed the compressibility of large datasets, namely the problem of finding dataset of minimal cardinality (small representative dataset) for a learning task. We developed a framework that jointly learns the input-output mapping and the representative dataset. We showed that its cardinality increases sub-linearly with respect to that of the original dataset. While an asymptotic compressibility of almost 100% is available in theory, we have observed that real datasets may be compressed as much as 95% without any noticeable drop in the prediction performance. These results challenge the efficiency and benefits of the existing approaches to create big datasets and serve as guidelines for designing algorithms for distributed learning and edge intelligence over communication-limited networks. The promising initial results are intended a proof of concept of the benefits of our approach. We envisage many extensions and applications in the future, especially where a machine learning algorithm is deployed on the field, such as Internet-of-Things, edge computing, autonomous driving, and smart manufacturing.

Acknowledgements

The work of H. S. Ghadikolaei was supported in part by the Swedish Research Council under Grant 2018-00820.

References

StatLib Repository. [Online] http://lib.stat.cmu.edu/datasets/, Accessed: 2019-01-22.

UCI Machine Learning Repository. [Online] http://mlr.cs.umass.edu/ml, Accessed: 2019-01-22.

Aharon, M., Elad, M., and Bruckstein, A. k-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, November 2006.

Bertsekas, D. *Nonlinear Programming*. Athena Scientific, 2 edition, 1999.

Bertsekas, D. P. *Network optimization: Continuous and discrete models*. Citeseer, 1998.

Boyd, S. and Vandenberghe, L. *Convex Optimization*. Cambridge University Press, 2004. ISBN 0521833787.

Chatterjee, S., Javid, A. M., Sadeghi, M., Mitra, P. P., and Skoglund, M. Progressive learning for systematic design of large neural networks. *arXiv preprint arXiv:1710.08177*, 2017.

Clarkson, K. L., Hazan, E., and Woodruff, D. P. Sublinear optimization for machine learning. *Journal of the ACM*, 59(5):23, 2012.

Cover, T. M. and Thomas, J. A. *Elements of Information Theory*. John Wiley & Sons, 2012.

Donoho, D. L. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, April 2006.

Gal, Y., Islam, R., and Ghahramani, Z. Deep bayesian active learning with image data. *arXiv preprint arXiv:1703.02910*, 2017.

Grant, M. and Boyd, S. CVX: Matlab software for disciplined convex programming, version 2.1, March 2014. [Online] http://cvxr.com/cvx, Accessed: 2019-01-22.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

Huang, G.-B., Zhu, Q.-Y., and Siew, C.-K. Extreme learning machine: Theory and applications. *Neurocomputing*, 70 (1-3):489–501, December 2006.

Huang, G.-B., Zhou, H., Ding, X., and Zhang, R. Extreme learning machine for regression and multiclass classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(2):513–529, April 2012.

Jerri, A. J. The shannon sampling theorem–Its various extensions and applications: A tutorial review. *Proceedings of the IEEE*, 65(11):1565–1596, November 1977.

Jiang, X., Ghadikolaei, H. S., Fodor, G., Modiano, E., Pang, Z., Zorzi, M., and Fischione, C. Low-latency networking: Where latency lurks and how to tame it. *Proceedings of the IEEE*, 2018. Accepted, To Appear.

Koh, P. W. and Liang, P. Understanding black-box predictions via influence functions. In *Proceedings of International Conference on Machine Learning*, volume 70, pp. 1885–1894, Aug 2017.

Kourou, K., Exarchos, T. P., Exarchos, K. P., Karamouzis, M. V., and Fotiadis, D. I. Machine learning applications in cancer prognosis and prediction. *Computational and Structural Biotechnology Journal*, 13:8–17, 2015.

Magnússon, S., Enyioha, C., Li, N., Fischione, C., and Tarokh, V. Convergence of limited communication gradient methods. *IEEE Transactions on Automatic Control*, 63(5):1356–1371, May 2018.

Michels, J., Saxena, A., and Ng, A. Y. High speed obstacle avoidance using monocular vision and reinforcement learning. In *Proceedings of International Conference on Machine Learning*, pp. 593–600. ACM, 2005.

MNIST. MNIST Data Set. [Online] http://yann.lecun.com/exdb/mnist, Accessed: 2019-01-22.

MovieLens. MovieLens Data Set. [Online] https://grouplens.org/datasets/movielens, Accessed: 2019-01-22.

Nedic, A. and Bertsekas, D. P. Incremental subgradient methods for nondifferentiable optimization. *SIAM Journal on Optimization*, 12(1):109–138, 2001.

Razaviyayn, M., Hong, M., and Luo, Z.-Q. A unified convergence analysis of block successive minimization methods for nonsmooth optimization. *SIAM Journal on Optimization*, 23(2):1126–1153, June 2013.

Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P. Design and analysis of computer experiments. *Statistical Science*, pp. 409–423, 1989.

Schalkwyk, J., Beeferman, D., Beaufays, F., Byrne, B., Chelba, C., Cohen, M., Kamvar, M., and Strope, B. "your word is my command": Google search by voice: a case study. In *Advances in Speech Recognition*, pp. 61–90. Springer, 2010.

Schapire, R. E. A brief introduction to boosting. In *International Joint Conference on Artificial Intelligence*, volume 2, pp. 1401–1406, 1999.

Settles, B. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114, 2012.

Shi, W., Cao, J., Zhang, Q., Li, Y., and Xu, L. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5):637–646, October 2016.

Smith, V., Forte, S., Chenxin, M., Takáč, M., Jordan, M. I., and Jaggi, M. CoCoA: A general framework for communication-efficient distributed optimization. *Journal of Machine Learning Research*, 18:230, 2018.

Tang, J., Deng, C., and Huang, G.-B. Extreme learning machine for multilayer perceptron. *IEEE Transactions on Neural Networks and Learning Systems*, 27(4):809–821, April 2016.

Tsang, I. W., Kwok, J. T., and Cheung, P.-M. Core vector machines: Fast SVM training on very large data sets. *Journal of Machine Learning Research*, 6(Apr):363–392, 2005.

Tsitsiklis, J. N. and Luo, Z.-Q. Communication complexity of convex optimization. *Journal of Complexity*, 3(3): 231–243, 1987.

Xu, Y. and Yin, W. A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. *SIAM Journal on imaging sciences*, 6(3):1758–1789, 2013.

Zhang, Q. and Li, B. Discriminative K-SVD for dictionary learning in face recognition. In *in Proceedings of IEEE Computer Vision and Pattern Recognition*, pp. 2691–2698, 2010.

Zhao, W., Chellappa, R., Phillips, P. J., and Rosenfeld, A. Face recognition: A literature survey. *ACM Computing Surveys*, 35(4):399–458, December 2003.

Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., and Oliva, A. Learning deep features for scene recognition using places database. In *or scene recognition using places dataAdvances in Neural Information Processing Systems (NIPS)*, pp. 487–495, 2014.