Data Driven Resource Allocation for Distributed Learning

Travis Dick
Carnegie Mellon University

Mu Li
Carnegie Mellon University

Venkata Krishna Pillutla
University of Washington

Colin White
Carnegie Mellon University

Maria Florina Balcan
Carnegie Mellon University

Alex Smola
Carnegie Mellon University
and AWS Deep Learning

Abstract

In distributed machine learning, data is dispatched to multiple machines for processing. Motivated by the fact that similar data points often belong to the same or similar classes, and more generally, classification rules of high accuracy tend to be “locally simple but globally complex” (Vapnik and Bottou, 1993), we propose data dependent dispatching that takes advantage of such structure. We present an in-depth analysis of this model, providing new algorithms with provable worst-case guarantees, analysis proving existing scalable heuristics perform well in natural non worst-case conditions, and techniques for extending a dispatching rule from a small sample to the entire distribution. We overcome novel technical challenges to satisfy important conditions for accurate distributed learning, including fault tolerance and balancedness. We empirically compare our approach with baselines based on random partitioning, balanced partition trees, and locality sensitive hashing, showing that we achieve significantly higher accuracy on both synthetic and real world image and advertising datasets. We also demonstrate that our technique strongly scales with the available computing power.

1 INTRODUCTION

Motivation and Overview: Distributed computation is playing a major role in modern large-scale machine learning practice with a lot of work in this direction in the last few years (Balcan et al., 2012b, 2013b, 2014; Li et al., 2014; Zhang et al., 2013, 2012). This tends to take two high-level forms. The first is when the data itself is collected in a distributed manner, whether from geographically-distributed experiments, distributed sensors, distributed click data, etc., and the goal is to take advantage of all this data without incurring the substantial overhead of first communicating it all to some central location. The second high-level form is where massive amounts of data are collected centrally, and for space and efficiency reasons this data must be dispatched to distributed machines in order to perform the processing needed (Li et al., 2014; Zhang et al., 2012). It is this latter form that we address here.

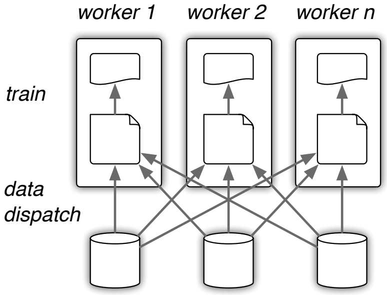


Figure 1: Data is partitioned and dispatched into multiple workers. Each worker then trains a local model using its local data. There is no communication between workers during training.

When data is dispatched to distributed machines, the simplest approach and what past work (both theoretical and empirical) has focused on is to perform the dispatching randomly (Zhang et al., 2012, 2013). Random dispatching has the advantage that dispatching is easy, and because each machine receives data from the same distribution, it is rather clean to analyze theoretically. However, since the distributions of the data on each machine are identical statistically, such techniques could lead to sub-optimal results in practice in terms of the accuracy of the resulting learning rule. Motivated by the fact that in practice, similar data points tend to have the same or similar classification, and more generally, classification rules of high accuracy tend to be “locally simple but globally complex” (Vapnik and Bottou, 1993), we propose a new paradigm for performing *data-dependent dispatching* that takes advantage of such structure by sending similar datapoints to similar machines. For example, a

Proceedings of the 20^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2017, Fort Lauderdale, Florida, USA. JMLR: W&CP volume 54. Copyright 2017 by the author(s).

globally accurate classification rule may be complicated, but each machine can accurately classify its *local* region with a simple classifier.

We introduce and analyze dispatching techniques that partition a set of points such that similar examples end up on the same machine/worker, while satisfying key constraints present in a real world distributed system including balancedness and fault-tolerance. Such techniques can then be used within a simple, but highly efficient distributed system that first partitions a small initial segment of data into a number of sets equal to the number of machines. Then each machine locally and independently applies a learning algorithm, with no communication between workers at training. In other words, the learning is embarrassingly parallel. See Figure 1 for a schematic representation. At the prediction time, we use a super-fast sublinear algorithm for directing new data points to the most appropriate machine.

Our Contributions: We propose a novel scheme for partitioning data which leads to better accuracy in distributed machine learning tasks, and we give a theoretical and experimental analysis of this approach. We present new algorithms with provable worst-case guarantees, analysis proving existing scalable heuristics perform well in natural non worst-case conditions, techniques for extending a dispatching rule from a small sample to the entire distribution, and an experimental evaluation of our proposed algorithms and several baselines on both synthetic and real-world image and advertising data. We empirically show that our method strongly scales and that we achieve significantly higher accuracy over baselines based on random partitioning, balanced partition trees, and locality-sensitive hashing.

In our framework, a central machine starts by clustering a small sample of data into roughly equal-sized clusters, where the number of clusters is equal to the number of available machines. Next, we extend this clustering into an efficient dispatch rule that can be applied to new points. This dispatch rule is used to send the remaining training data to the appropriate machines and to direct new points at prediction time. In this way, similar datapoints wind up on the same machine. Finally, each machine independently learns a classifier using its own data (in an embarrassingly parallel manner). To perform the initial clustering used for dispatch, we use classic clustering objectives (k-means, k-median, and k-center). However, we need to add novel constraints to ensure that the clusters give a data partition that respects the constraints of real distributed learning systems:

Balancedness: We need to ensure our dispatching procedure balances the data across the different machines. If a machine receives much more data than other machines, then it will be the bottleneck of the algorithm. If any machine receives very little data, then its processing power is wasted. Thus, enforcing upper and lower bound constraints on the cluster sizes leads to a faster, more efficient setup.

Fault-Tolerance: In order to ensure that our system is robust to machine failures, we assign each point to multiple distinct clusters. This way, even if a machine fails, the data on that machine is still present on other machines. Moreover, this has the added benefit that our algorithms behave well on points near the boundaries of the clusters. We say a clustering algorithm satisfies p-replication if each point is assigned to p distinct clusters.

Efficiency: To improve efficiency, we apply our clustering algorithms to a small sample of data. Therefore, we need to be able to extend the clustering to new examples from the same distribution while maintaining a good objective value and satisfying all constraints. It is important that the extension technique be efficient for both the initial partitioning and when we dispatch examples at prediction time.

When designing clustering algorithms, adding balancedness and fault tolerance makes the task significantly harder. Prior work has considered upper bounds on the cluster sizes (Li, 2014b; Byrka et al., 2015b; Li, 2014a; An et al., 2014; Khuller and Sussmann, 1996; Cygan et al., 2012) [1] and lower bounds (Aggarwal et al., 2006; Ahmadian and Swamy, 2016), but no prior work has shown provable guarantees with upper and lower bounds on the cluster sizes simultaneously. While capacitated clustering objective functions are nondecreasing as the number of clusters k increases, with lower bounds on the cluster sizes, we show the objective function can oscillate arbitrarily with respect to k. This makes the problem especially challenging from a combinatorial optimization perspective. Existing capacitated clustering algorithms work by rounding a fractional linear program solution, but the erratic nature of the objective function makes this task more difficult for us.

The balance constraints also introduce challenges when extending a clustering-based partitioning from a small sample to unseen data. The simple rule that assigns a new point to the cluster with the nearest center provides the best objective value on new data, but it can severely violate the balance constraints. Therefore, any balanced extension rule must take into account the distribution of data.

We overcome these challenges, presenting a variety of complementary results, which together provide strong justification for our distributed learning framework. We summarize each of our main results below.

• **Balanced fault-tolerant clustering:** We provide the first algorithmic results with provable guarantees that simultaneously handle upper and lower bounds on the cluster sizes, as well as fault tolerance. Clustering is NP-hard and adding more constraints makes it significantly harder, as we will see in Section 2. For this reason, we first devise approximation

[1] Note that enforcing only upper (resp. lower) bounds implies a weak lower (resp. upper) bound on the cluster sizes, but this is only nontrivial if the upper (resp. lower) bounds are extremely tight or the number of clusters is a small constant.

algorithms with strong worst-case guarantees, demonstrating this problem is tractable. Specifically, in Section 2 we provide an algorithm that produces a fault-tolerant clustering that approximately optimizes k-means, k-median, and k-center objectives while also roughly satisfying the given upper and lower bound constraints. At a high level, our algorithm proceeds by first solving a linear program, followed by a careful balance and replication aware rounding scheme. We use a novel min-cost flow technique to finish off rounding the LP solution into a valid clustering solution.

• **k-means++ under stability:** In addition to these algorithms which give provably strong guarantees in the worst-case, we give complementary results which show that for 'typical' problem instances, it is possible to achieve better guarantees with simpler, more scalable algorithms. Specifically, in Section 3 we show the popular k-means++ algorithm outputs a balanced clustering with stronger theoretical guarantees, provided the data satisfies a natural notion of stability. We make nontrivial extensions of previous work to ensure the upper and lower size constraints on the clusters are satisfied. No previous work gives provable guarantees while satisfying both upper and lower bounds on the cluster sizes, and Sections 2 and 3 may be of independent interest beyond distributed learning.

• **Efficient clustering by subsampling:** For datasets large enough to require distributed processing, clustering the entire dataset is prohibitively expensive. A natural way to avoid this cost is to only cluster a small subset of the data and then efficiently extend this clustering to the entire dataset. The simple extension that assigns each new point to the p clusters with the closest centers does not satisfy the balance constraints. Instead, in Section 4 we show that assigning a new example to the same p clusters as its nearest neighbor in the clustered subsample approximately preserves both the objective value and all constraints. We also use this technique at prediction time to send new examples to the most appropriate machines.

• **Experimental results:** We conduct experiments with both our LP rounding algorithms and k-means++ together with our nearest neighbor extension technique. We include empirical (and theoretical) comparisons which show the effectiveness of both algorithms in different situations. The k-means++ algorithm is competitive on real world image and advertising datasets, complementing the results of Section 3 by showing empirically that k-means++ produces high-quality balanced clusterings for 'typical' datasets. We then compare the performance of our framework (using k-means++ with nearest neighbor extension) against three baseline methods (random partitioning, balanced partition trees, and locality sensitive hashing) in large scale learning experiments where each machine trains an SVM classifier. We find that for all datasets and across a wide range of k values, our algorithm achieves higher accuracy than any of the baselines. Finally, we show that our technique strongly scales, meaning that doubling the available computational power while keeping the workload fixed reduces the running time by a constant factor, demonstrating that our method can scale to very large datasets.

Related Work: Currently, the most popular method of dispatch in distributed learning is random dispatch (Zhang et al., 2013, 2012). This may not produce optimal results because each machine must learn a global model. Previous work has studied partitioning for distributed machine learning (Wei et al., 2015; You et al., 2015; Delling et al., 2011; Bourse et al., 2014; Aydin et al., 2016), but none simultaneously achieve load-balancing guarantees and approximation guarantees for k-median, k-means, or k-center.

Previous work in theoretical computer science has considered capacitated clustering, or clustering with upper bounds (Li, 2014b; Byrka et al., 2015b; Li, 2014a; Cygan et al., 2012), and lower bounds (Aggarwal et al., 2006; Ahmadian and Swamy, 2016), but our algorithm is the first to solve a more general and challenging question of simultaneously handling upper and lower bounds on the cluster sizes, and p-replication. See Section 7 in the supplementary material for a more detailed discussion about related work.

2 FAULT TOLERANT BALANCED CLUSTERING

In this section, we give an algorithm to cluster a small initial sample of data to create a dispatch rule that sends similar points to the same machine. There are many ways to measure the similarity of points in the same cluster. We consider three classic clustering objectives, k-means, k-median, and k-center clustering while imposing upper and lower bounds on the cluster sizes and replication constraints. This is the first algorithm with provable guarantees to simultaneously handle both upper and lower bounds on the cluster sizes.

A clustering instance consists of a set V of n points, and a distance metric d. Given two points i and j in V, denote the distance between i and j by $d(i,j)$. The task is to find a set of k centers $C = \{c_1, \ldots, c_k\} \subset V$ and assignments of each point to p of the centers $f : V \to \binom{C}{p}$, where $\binom{C}{p}$ represents the subset of C^p with no duplicates. In this paper, we study three popular clustering objectives:

(1) k-median: $\min_{C,f} \sum_{i \in V} \sum_{j \in f(i)} d(i,j)$

(2) k-means: $\min_{C,f} \sum_{i \in V} \sum_{j \in f(i)} d(i,j)^2$

(3) k-center: $\min_{C,f} \max_{i \in V} \max_{j \in f(i)} d(i,j)$

We add size constraints $0 < \ell \leq L < 1$, also known as capacity constraints, so each cluster must have a size between $n\ell$ and nL. For simplicity, we assume these values are integral (or replace them by $\lceil n\ell \rceil$ and $\lfloor nL \rfloor$ respectively). Before we present our approximation algorithm, we discuss the challenges introduced by these size constraints.

Structure of Balanced Clustering: It is well-known that

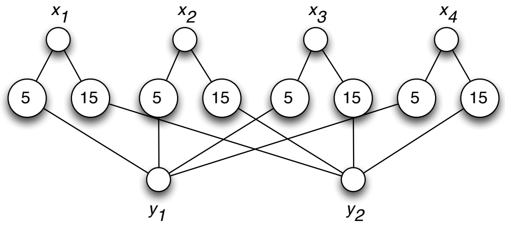


Figure 2: Each edge signifies distance 1, and all other distances are 2. The middle points are replicated as many times as their label suggests (but each pair of replicated points are still distance 2 away). Finally, add length 1 edges between all pairs in $\{x_1, x_2, x_3, x_4\}, \{y_1, y_2\}$.

solving the objectives optimally are NP-hard (even without the capacity and fault tolerance generalizations) (Jain et al., 2003). In fact, with the addition of lower bounds, the value of the optimal clustering objective $\mathcal{OPT}$ as a function of k behaves erratically. In uncapacitated clustering and clustering with upper bounds only, given a problem instance, the cost of the optimal solution always decreases as k increases. This is easy to see: given a set of optimal centers, if we add another center v, at the very least v is now distance 0 from a center, which decreases the cost.

However, when there are lower bounds on the cluster sizes, there are simple examples in which the value of the optimal solution as a function of k contains a local minimum. For instance, the star graph has this property (see Section 11 in the supplementary material). A much more subtle question is whether there exists a clustering instance with a local *maximum*. We confirm such clusterings do exist; see Figure 2. We give the idea here and defer the formal proof to Section 8 in the supplementary material.

Lemma 1. *There exists a balanced clustering instance with $p = 1$ for which the k-center, k-median, and k-means objectives contain a local maximum with respect to k.*

Proof sketch. Consider Figure 2, where $n = 86$, and set $n\ell = 21$. Since the distances are all 1 or 2, this construction is trivially a valid distance metric. From Figure 2, we see that $k = 2$ and $k = 4$ have valid clusterings using only length 1 edges, using centers $\{y_1, y_2\}$ and $\{x_1, x_2, x_3, x_4\}$, respectively. But now consider $k = 3$. The crucial property is that by construction, y_1 and any x_i cannot simultaneously be centers and each satisfy the capacity to distance 1 points, because the union of their distance 1 neighborhoods is less than $2n\ell$. In the supplementary material, we carefully check all other sets of 3 centers do not achieve a clustering with distance 1 edges, which completes the proof. □

In fact, with a more intricate clustering instance, we are able to show (in Theorem 8 in the supplementary material) that *any number* of local maxima may exist!

Approximation Algorithm: In light of these difficulties, one might ask whether any approximation algorithm exists for this problem. We answer affirmatively, by extending previous work (Li, 2014a) to fit our more challenging constrained optimization problem. Our algorithm returns a clustering whose cost is at most a constant factor multiple of the optimal solution, while violating the capacity and replication constraints by a small constant factor.

Theorem 2. *Algorithm 1 returns a constant factor approximate solution for the balanced k-clustering with p-replication problem for $p > 1$, where the upper capacity constraints are violated by at most a factor of $\frac{p+2}{p}$, and each point can be assigned to each center at most twice.*

At a high level, our algorithm proceeds by first solving a linear program, followed by careful rounding. The key insight is that p-replication helps to mitigate the capacity violation in the rounding phase. Together with a novel min-cost flow technique, this allows us to simultaneously handle upper and lower bounds on the cluster sizes. The procedure is summarized in Algorithm 1, and below we provide details, together with the key ideas behind its correctness (see Section 9 in the supplementary material for the full details).

Step 1: Linear Program The first step is to solve an linear program (LP) relaxation of the integer program (IP) formulation of our constrained clustering problem. The variables are as follows: for each $i \in V$, let y_i be an indicator for whether i is opened as a center. For $i, j \in V$, let x_{ij} be an indicator for whether point j is assigned to center i. In the LP, the variables may be fractional, so y_i represents the fraction to which a center is opened (we will refer to this as the "opening" of i), and x_{ij} represents the fractional assignment of j to i. One can use an LP solver to get a fractional solution which must then be rounded (i.e., the LP may open up $2k$ 'half' centers). Let (x, y) denote an optimal solution to the LP. For any points i and j, let c_{ij} be the cost of assigning point j to center i. That is, for k-median, $c_{ij} = d(i, j)$, and for k-means $c_{ij} = d(i, j)^2$ (we discuss k-center in the supplementary material). Define $C_j = \sum_i c_{ij}x_{ij}$, the average cost from point j to its centers in the LP solution (x, y).

It is well-known that the LP in Algorithm 1 has an unbounded integrality gap (the ratio of the optimal LP solution over the optimal integral LP solution), even when the capacities are violated by a factor of $2 - \epsilon$ (Li, 2014a). However, with fault tolerance, the integrality is only unbounded when the capacities are violated by a factor of $\frac{p}{p-1}$ (see the supplementary material for the integrality gap). Intuitively, this is because the p centers can 'share' this violation.

Step 2: Monarch Procedure Next, partition the points into "empires" such that every point is $\leq 4C_j$ from the center of its empire (the "monarch") by using a greedy procedure from Charikar et al. (1999) (for an informal description, see step 2 of Algorithm 1). By Markov's inequality, every empire has total opening $\geq p/2$, which is crucially ≥ 1 for $p \geq 2$ under our model of fault tolerance.

1. Find a solution to the following linear program:
$$\min_{x,y} \sum_{i,j \in V} c_{ij} x_{ij} \quad \text{s.t.}$$
$$\textbf{(a)}\, \forall j \in V : \sum_{i \in V} x_{ij} = p; \quad \textbf{(b)} \sum_{i \in V} y_i \leq k;$$
$$\textbf{(c)}\, \forall i \in V : \ell y_i \leq \sum_{j \in V} \frac{x_{ij}}{n} \leq L y_i;$$
$$\textbf{(d)}\, \forall i, j \in V : 0 \leq x_{ij} \leq y_i \leq 1.$$
2. Greedily place points into a set $\mathcal{M}$ from lowest C_j to highest (called "monarchs"), adding point j to $\mathcal{M}$ if it is not within distance $4C_j$ of any monarch. For each monarch u, let $\mathcal{E}_u$ be the points closest to u, called u's empire.
3. For empire $\mathcal{E}_u$ with total fractional opening $Y_u \triangleq \sum_{i \in \mathcal{E}_u} y_i$, give opening $Y_u / \lfloor Y_u \rfloor$ to the $\lfloor Y_u \rfloor$ closest points to u and all other points opening 0.
4. Round the x_{ij}'s by constructing a minimum cost flow problem on a bipartite graph of centers and points, setting up demands and capacities to handle the bounds on cluster sizes.

Algorithm 1: Balanced clustering with fault tolerance

Step 3: Aggregation The point of this step is to end up with $\leq k$ centers total. Since each empire has total opening at least 1, we can aggregate openings within each empire. For each empire $\mathcal{E}_u$, we move the openings to the $\lfloor Y_u \rfloor$ innermost points of $\mathcal{E}_u$, where $Y_u = \sum_{i \in \mathcal{E}_u} y_i$. We accomplish this using an iterative greedy procedure, similar to (Li, 2014a) (we give details in the supplementary material). We preserve all LP constraints, except we may incur a factor $\frac{p+2}{p}$ increase to the capacity constraints. At the end of the procedure, there are $\leq k$ points with nonzero opening, so we can set them all to 1 to round the y's. The cost incurred in this step can be bounded using the triangle inequality.

Step 4: Min cost flow Now we must round the x's. We set up a min cost flow problem, where an integral solution corresponds to an assignment of points to centers. We create a bipartite graph with V on the left (each with supply p) and the k centers on the right (each with demand $n\ell$), and a sink vertex with demand $np - kn\ell$. We carefully set the edge weights so that the minimum cost flow that satisfies the capacities corresponds to an optimal clustering assignment. Then using the Integral Flow Theorem, we are guaranteed there is an *integral* assignment that achieves the same optimal cost (and finding the min cost flow is a well-studied polynomial time problem (Papadimitriou and Steiglitz, 1998)). Thus, we can round the x's without incurring any additional cost to the approximation factor. This is the first time this technique has been used in the setting of clustering.

In Section 10 of the supplementary material, we show a more involved algorithm specifically for k-center which achieves a 6-approximation with *no violation* to the capacity or replication constraints.

3 BALANCED CLUSTERING UNDER STABILITY

In the previous section, we showed an LP-based algorithm which provides theoretical guarantees even on adversarially chosen data. Often real-world data has inherent structure that allows us to use more scalable algorithms and achieve even better clusters (Balcan et al., 2013a; Ostrovsky et al., 2006). In our distributed ML framework, this translates to being able to use a larger initial sample for the same computational power (Section 4 analyzes the effect of sample size). In this section, we prove the popular k-means++ algorithm as well as a greedy thresholding algorithm output clusters very close to the optimal solution, provided the data satisfies a natural notion of stability called *approximation stability* (Balcan et al., 2013a; Agarwal et al., 2015; Balcan and Braverman, 2010; Balcan et al., 2016; Gupta et al., 2014).

Specifically, we show that given a balanced clustering instance in which clusterings close in *value* to $\mathcal{OPT}$ are also close in terms of the clusters themselves, assuming $L \in O(\ell)$, then k-means++ with a simple pruning step (Ostrovsky et al., 2006) outputs a solution close to optimal. We overcome key challenges that arise when we add upper and lower bounds to the cluster sizes. We include the details in Section 11 of the supplementary material.

Approximation Stability: Given a clustering instance (S, d) and inputs ℓ and L, and let $\mathcal{OPT}$ denote the cost of the optimal balanced clustering. Two clusterings $\mathcal{C}$ and $\mathcal{C}'$ are ϵ-*close*, if only an ϵ-fraction of the input points are clustered differently in the two clusterings, i.e., $\min_\sigma \sum_{i=1}^{k} |C_i \setminus C'_{\sigma(i)}| \leq \epsilon n$, where σ is a permutation of $[k]$.

Definition 1 (Balcan et al. (2013a))**.** *A clustering instance* (S, d) *satisfies* $(1 + \alpha, \epsilon)$-approximation stability *with respect to balanced clustering if all clusterings* $\mathcal{C}$ *with cost*$(\mathcal{C}) \leq (1 + \alpha) \cdot \mathcal{OPT}$ *are* ϵ*-close to* $\mathcal{C}$*.*

k**-means:** We show that sampling $k \log k$ centers using k-means++, followed by a greedy center-pruning step, (introduced by Ostrovsky et al. (2006)) is sufficient to cluster well with high probability, assuming (α, ϵ)-approximation stability for balanced clustering. Our results improve over Agarwal et al. (2015), who showed this algorithm outputs a good clustering with probability $\Omega(\frac{1}{k})$ for standard (unbalanced) clustering under approximation stability. Formally, our result is the following.

Theorem 3. *Given* $\frac{\epsilon \cdot k}{\alpha} < \rho < 1$*, k-means++ seeding with a greedy pruning step outputs a solution that is* $\frac{1}{1-\rho}$ *close to the optimal solution with probability* $> 1 - O(\rho)$*, for clustering instances satisfying* $(1+\alpha, \epsilon)$*-approximation stability for the balanced* k*-means objective, with* $\frac{L}{\ell} \in O(1)$*.*

Intuitively, (α, ϵ)-approximation stability forces the clusters to become "spread out", i.e., the radius of any cluster is much smaller than the inter-cluster distances. This allows us to show for 2-means clustering, the k-means++ seeding procedure will pick one point from each cluster with high probability. However, if we induct on the number of clusters, the probability of success becomes exponentially small in k. We circumvent this issue in a manner similar to Ostrovsky et al. (2006), by sampling $k \log k$ centers, and carefully deleting centers greedily, until we are left with one center per cluster with high probability.

k-median and k-center: We show that the greedy thresholding algorithm of Balcan et al. (2013a) is sufficient to give a good clustering even for the balanced k-median or k-means objective, under approximation stability. At a high level, their algorithm works by first creating a threshold graph for a specific distance, and then iteratively picking the node with the highest degree in the threshold graph and removing its neighborhood. We show balanced clustering instances where the analysis in Balcan et al. (2013a) is not sufficient to guarantee good clusterings are outputted. We provide a new technique which overcomes the difficulties in adding upper and lower balance constraints. The technique involves showing there cannot be too many distinct pairs of points from different clusters which are close together, otherwise swapping these points between clusters would conserve the balance constraints and contradict approximation stability. We obtain the following theorem.

Theorem 4. *(1) There is an efficient algorithm which returns a valid clustering that is $O(\frac{\epsilon}{\alpha})$-close to the optimal, for balanced k-median or k-means clustering under $(1 + \alpha, \epsilon)$-approximation stability, provided all clusters are size $\geq 3\epsilon n(1 + \frac{3}{\alpha})$. (2) There is an efficient algorithm which returns the optimal clustering for balanced k-center under $(2, 0)$-approximation stability. (3) For $\epsilon > 0$, there does not exist an efficient algorithm which returns the optimal clustering for balanced k-center under $(2 - \epsilon, 0)$-approximation stability, unless $NP = RP$.*

4 EFFICIENT CLUSTERING BY SUBSAMPLING

For datasets large enough to require a distributed learning system, it is expensive to apply a clustering algorithm to the entire dataset. In this section, we show that we can first cluster a small subsample of data and then efficiently extend this clustering to the remaining data. In our technique, each point in the dataset is assigned to the same p clusters as its nearest neighbor in the clustered subsample. In fact, this dispatch rule extends the clustering to the entire space $\mathcal{X}$ (not just to the unused portion of the training set), so at prediction time it can be used to send query points to the appropriate machines. We show that the clustering induced over $\mathcal{X}$ approximately inherits all of the desirable properties of the clustered subsample: good objective value, balanced clusters, and replication.

We measure the quality of a clustering of $\mathcal{X}$ as follows: given a data distribution μ over $\mathcal{X}$, our goal is to find a clustering with centers $C = \{c_1, \ldots, c_k\}$ and an assignment function $f : \mathcal{X} \to \binom{C}{p}$ for the entire space that minimizes $Q(f, C) = \mathbb{E}_{x \sim \mu}[\sum_{c_j \in f(x)} d(x, c_j)]$ subject to the balance constraints $\mathbb{P}_{x \sim \mu}(c_j \in f(x)) \in [\ell, L]$ for all j. In this section we focus on the k-median objective, but similar results for k-means are given in Section 12 of the supplementary material.

The simplest approach to extend a clustering of small subsample of data is to assign a new example x to the p clusters with the closest centers. This strategy incurs the lowest cost for new examples, but it may severely violate the balance constraints if the distribution is concentrated near one center.

Instead, given a clustering of the subsample S, our technique assigns a new example x to the same p clusters as its nearest neighbor in the set S, denoted by $NN_S(x)$. We use the fact that the clustering of S is balanced to show that the extended clustering is also balanced. Some points in S will represent more probability mass of μ than others, so we use a second independent sample S' to estimate weights for each point in S, which are used in a weighted version of the objective and balance constraints. Pseudocode is given in Algorithm 2. We obtain the following guarantees for k-median.

Theorem 5. *For any $\epsilon, \delta > 0$, let $(\bar{g}_S, C_S)$ be the output of Algorithm 2 with parameters k, p, ℓ, L and second sample size $n' = O\big(\frac{1}{\epsilon^2}(n + \log \frac{1}{\delta})\big)$. Let (f^*, C^*) be any clustering of $\mathcal{X}$ and (g^*_S, C^*_S) be an optimal clustering of S under Q_S satisfying the weighted balance constraints (ℓ, L). Suppose that $Q_S(g_S, C_S) \leq r \cdot Q_S(g^*_S, C^*_S) + s$. Then w.p. $\geq 1 - \delta$ over the second sample, the output $(\bar{g}_S, C_S)$ satisfies the balance constraints with $\ell' = \ell - \epsilon$ and $L' = L + \epsilon$ and*

$$\begin{aligned} Q(\bar{g}_S, C_S) \leq r \cdot Q(f^*, C^*) + s + 2(r + 1)pD\epsilon \\ + p(r + 1)\alpha(S) + r\beta(S, \ell + \epsilon, L - \epsilon), \end{aligned}$$

where D is the diameter of $\mathcal{X}$, the quantity $\alpha(S) = \mathbb{E}_{x \sim \mu}[d(x, \mathrm{NN}_S(x))]$ measures how well μ is approximated by S, and $\beta(S, \ell, L) = \min_{h,C} \{Q(\bar{h}, C) - Q(f^*, c^*)\}$ measures the loss incurred by restricting to clusterings that are constant over the Voronoi tiles of S.

The terms $\alpha(S)$, $\beta(S)$ can be bounded in terms of the size of S under natural conditions on the distribution μ. For example, when the distribution has doubling dimension d_0 and the optimal clustering of $\mathcal{X}$ is ϕ-probabilistically Lipschitz (Urner et al., 2011, 2013) (intuitively requiring that the probability mass close to the cluster boundaries is small) then for $n = \tilde{O}((\frac{1}{\epsilon \phi^{-1}(\epsilon)})^{d_0} d_0)$ we will have $\alpha(S) < D\epsilon$ and $\beta(S) < pD\epsilon$ with high probability. See Section 12 in the supplementary material for details.

5 EXPERIMENTS

In this section, we present an empirical study of the accuracy and scalability of our technique using both the LP rounding algorithms and k-means++ together with the nearest neighbor extension. We compare against three baselines: random

```
Input: Dataset $S = \{x_1, \dots, x_n\}$, cluster parameters $(k, p, \ell, L)$, second sample size $n'$.
1. Draw second sample $S'$ of size $n'$ iid from $\mu$.
2. For each point $x_i$, set $\hat{w}_i = |S'_i|/n'$, where $S'_i = \{x' \in S' : NN_S(x') = x_i\}$
3. Let $C_S = (c_1, \dots, c_k)$ and $g_S : S \to \binom{C}{p}$ be a clustering of $S$ obtained by minimizing
       $Q_S(g, C) = \sum_{i=1}^{n} \hat{w}_i \sum_{c_j \in g(x)} d(x_i, c_j)$
   subject to $\sum_{i : c_j \in g_n(x_i)} \hat{w}_i \in [\ell, L]$ for all $j = 1, \dots, k$.
4. Return $\bar{g}_S(x) = g_S(NN_S(x))$ and centers $C_S$.
```

Algorithm 2: Nearest neighbor clustering extension.

partitioning, balanced partition trees, and locality sensitive hashing (LSH) on both synthetic and real world image and advertising datasets. Our findings are summarized below:

• Using small samples of the given datasets, we compare the clusterings produced by our LP rounding algorithms[2] and k-means++ (with balancing heuristics described shortly). We find that clusterings produced by k-means++ and the LP rounding algorithms have similar objective values and correlate well with the underlying class labels. These results complement the results of Section 3, showing that k-means++ produces high quality balanced clusterings for 'typical' data. This comparison is detailed in Sections 13 and 14 of the supplementary material. Based on this observation, our further empirical studies use k-means++.

• We compare the accuracy of our technique (using k-means++ and the nearest neighbor extension) to the three baselines for a wide range of values of k in large-scale learning tasks where each machine learns a local SVM classifier. For all values of k and all datasets, our algorithm achieves higher accuracy than all our baselines.

• We show that our framework exhibits strong scaling, meaning that if we double the available computing power, the total running time reduces by a constant fraction.

Experimental Setup: In each run of our experiment, one of the partitioning algorithms produces a dispatch rule from $10,000$ randomly sampled training points. This dispatch rule is then used to distribute the training data among the available worker machines. If the parameter k exceeds the number of machines, we allow each machine to process multiple partitions independently. Next we train a one-vs-all linear separator for each partition in parallel by minimizing the L2-regularized L2-loss SVM objective. This objective is minimized using Liblinear (Fan et al., 2008) when the data is small enough to fit in the each worker's memory, and L-BFGS otherwise (note that both solvers will converge to the same model). The regularization parameter is chosen via 5-fold cross validation. To predict the label of a new example, we use the dispatch rule to send it to the machine with the most appropriate model. All experimental results are averaged over 10 independent runs.

Details for our technique: Our method builds a dispatch rule by clustering a small sample of data using k-means++ and uses the nearest neighbor dispatch rule in order to dispatch both the training and testing data. To ensure a balanced partitioning, we apply the following simple balancing heuristics: while there is any cluster smaller than ℓn points, pick any such cluster and merge it with the cluster whose center is nearest. Then each cluster that is larger than Ln points is randomly partitioned into evenly sized clusters that satisfy the upper capacity constraint. This guarantees every cluster satisfies the capacity constraints, but the number of output clusters may differ from k. For the nearest neighbor dispatch, we use the random partition tree algorithm of Dasgupta and Sinha (2015) for efficient approximate nearest neighbor search. We set $\ell = 1/(2k)$ and $L = 2/k$ and $p = 1$, since our baselines do not support replication.

Baselines: We compare against the following baselines.[3]

Random Partitioning: Points are dispatched uniformly at random. This baseline produces balanced partitions but does not send similar examples to the same machine.

Balanced Partition Trees: Similarly to a kd-tree, this partitioning rule recursively divides the dataset by splitting it at the median point along a randomly chosen dimension. This is repeated until the tree has k leaves (where we assume k is a power of 2). This baseline produces balanced partitions and improves over random partitioning because each machine learns a local model for a different subset of the space. The drawback is that the partitioning may result in subsets that do not contain similar data points.

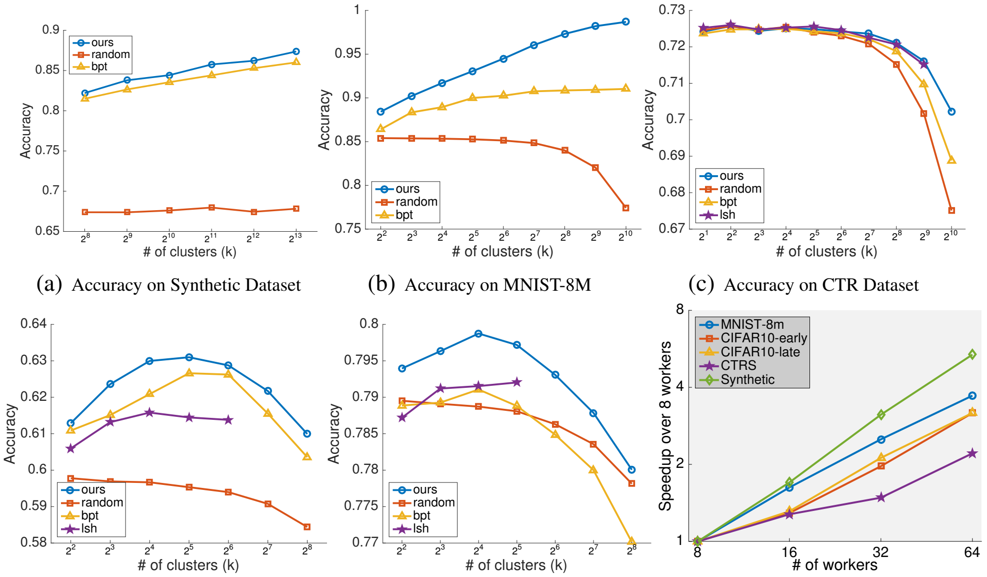
LSH Partitioning: This baseline uses locality sensitive hash functions (Andoni and Indyk, 2006) to dispatch similar points to the same machine. Given an LSH family H, we pick a random hash $h : \mathbb{R}^d \to \mathbb{Z}$. Then a point x is assigned to cluster $h(x) \mod k$. In our experiments, we use the concatenation of 10 random projections followed by binning (Datar et al., 2004). See Section 13 for details of the construction. This baseline sends similar examples to the same machine, but does not balance the cluster sizes (which is essential for practical data distribution).

Datasets: We use the following datasets:

Synthetic: We use a 128 GB synthetic dataset with 30 classes and 20 features. The data distribution is a mixture of 200 Gaussians with uniformly random centers in $[0, 1]^{20}$ with covariance $0.09I$. Labels are assigned so that nearby Gaus-

[2] We can run the LP rounding algorithm for small n, even though there are $O(n^2)$ variables.

[3] Since our framework does not communicate during training, we do not compare against algorithms that do, e.g. boosting (Balcan et al., 2012a).



(a) Accuracy on Synthetic Dataset (b) Accuracy on MNIST-8M (c) Accuracy on CTR Dataset

(d) Accuracy on CIFAR-10 (in3c) (e) Accuracy on CIFAR-10 (in4d) (f) **Strong Scaling**

Figure 3: Figures (a) through (e) show the effect of k on the classification accuracy. Figure (f) shows the speedup factor as we increase the number of workers from 8 to 64 for each dataset.

sians have the same label.

MNIST-8M: We use the raw pixels of the MNIST-8M dataset (Loosli et al., 2007). It has $8M$ examples and 784 features.

CIFAR-10: The CIFAR-10 dataset (Krizhevsky, 2009) is an image classification task with 10 classes. Following Krizhevsky et al. (2012) we include 50 randomly rotated and cropped copies of each training example to get a training set of 2.5 million examples. We extract the features from the Google Inception network (Szegedy et al., 2015) by using the output of an early layer (in3c) and a later layer (in4d).

CTR: The CTR dataset contains ad impressions from a commercial search engine where the label indicates whether the ad was clicked. It has 860K examples with 232 features.

Results: Our empirical results are shown in Figure 3. We do not report accuracies when the partitioning is imbalanced, specifically when the largest $k/2$ clusters contain more than 98% of the data. For all values of k and all datasets, our method has higher accuracy than all three baselines. The balanced partition tree is the most competitive baseline, but in Section 13 we present an additional synthetic distribution for which our algorithm drastically outperforms the balanced partition tree. For all datasets except CTR, the accuracy of our method increases as a function of k, until k is so large that each cluster becomes data starved. Our method combines the good aspects of both the balanced partition tree and LSH baselines by simultaneously sending similar examples to the same machines and ensuring that every machine gets roughly the same amount of data.

Figure 3(f) shows the speedup obtained when running our system using 16, 32, or 64 workers compared to using 8. We clock the time taken for the entire experiment: the time for clustering a subsample, dispatch, training and testing. In all cases, doubling the number of workers reduces the total time by a constant factor, showing that our framework strongly scales and can be applied to very large datasets.

6 CONCLUSION

In this work, we propose and analyze a new framework for distributed learning. Given that similar points tend to have similar classes, we partition the data so that similar examples go to the same machine. We cast the dispatching step as a clustering problem combined with novel fault tolerance and balance constraints necessary for distributed systems. We show the added constraints make the objective highly nontrivial, yet we provide LP rounding algorithms with provable guarantees. This is complemented by our results showing that the k-means++ algorithm is competitive on ‘typical’ datasets. These are the first algorithms with provable guarantees under both upper and lower capacity constraints, and may be of interest beyond distributed learning. We show that it is sufficient to cluster a small subsample of data and use a nearest neighbor extension technique to efficiently dispatch the remaining data. Finally, we conduct experiments for all our algorithms that support our theoretical claims, show that our framework outperforms several baselines and strongly scales.

Acknowledgements

This work was supported in part by NSF grants CCF-1451177, CCF-1422910, CCF-1535967, IIS-1618714, IIS-1409802, a Sloan Research Fellowship, a Microsoft Research Faculty Fellowship, a Google Research Award, Intel Research, Microsoft Research, and a National Defense Science & Engineering Graduate (NDSEG) fellowship.

References

Karen Aardal, Pieter L van den Berg, Dion Gijswijt, and Shanfei Li. Approximation algorithms for hard capacitated k-facility location problems. *European Journal of Operational Research*, (2):358–368, 2015.

Manu Agarwal, Ragesh Jaiswal, and Arindam Pal. k-means++ under approximation stability. *Theoretical Computer Science*, 588:37–51, 2015.

Gagan Aggarwal, Tomás Feder, Krishnaram Kenthapadi, Samir Khuller, Rina Panigrahy, Dilys Thomas, and An Zhu. Achieving anonymity via clustering. In *Proceedings of the twenty-fifth ACM symposium on Principles of database systems*, pages 153–162, 2006.

Sara Ahmadian and Chaitanya Swamy. Approximation algorithms for clustering problems with lower bounds and outliers. In *Proceedings of the 43rd annual International Colloquium on Automata, Languages, and Programming*, 2016.

Hyung-Chan An, Aditya Bhaskara, Chandra Chekuri, Shalmoli Gupta, Vivek Madan, and Ola Svensson. Centrality of trees for capacitated k-center. In *Integer Programming and Combinatorial Optimization*, pages 52–63. Springer, 2014.

Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 459–468. IEEE, 2006.

Kevin Aydin, MohammadHossein Bateni, and Vahab Mirrokni. Distributed balanced partitioning via linear embedding. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, pages 387–396. ACM, 2016.

Maria-Florina Balcan and Mark Braverman. Approximate nash equilibria under stability conditions. Technical report, 2010.

Maria-Florina Balcan, Avrim Blum, Shai Fine, and Yishay Mansour. Distributed learning, communication complexity and privacy. *arXiv preprint arXiv:1204.3514*, 2012a.

Maria-Florina Balcan, Avrim Blum, Shai Fine, and Yishay Mansour. Distributed learning, communication complexity, and privacy. In *Conference on Learning Theory*, 2012b.

Maria-Florina Balcan, Avrim Blum, and Anupam Gupta. Clustering under approximation stability. *J. ACM*, 60(2):8:1–8:34, May 2013a. ISSN 0004-5411. doi: 10.1145/2450142.2450144. URL http://doi.acm.org/10.1145/2450142.2450144.

Maria-Florina Balcan, Steven Ehrlich, and Yingyu Liang. Distributed k-means and k-median clustering on general communication topologies. In *Advances in Neural Information Processing Systems*, 2013b.

Maria-Florina Balcan, Vandana Kanchanapally, Yingyu Liang, and David Woodruff. Improved distributed principal component analysis. In *Advances in Neural Information Processing Systems*, 2014.

Maria-Florina Balcan, Nika Haghtalab, and Colin White. k-center clustering under perturbation resilience. In *Proceedings of the 43rd annual International Colloquium on Automata, Languages, and Programming*, 2016.

J. Barilan, G. Kortsarz, and D. Peleg. How to allocate network centers. *Journal of Algorithms*, 15(3):385 – 415, 1993. ISSN 0196-6774. doi: http://dx.doi.org/10.1006/jagm.1993.1047. URL http://www.sciencedirect.com/science/article/pii/S0196677483710473.

Florian Bourse, Marc Lelarge, and Milan Vojnovic. Balanced graph edge partition. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1456–1465. ACM, 2014.

Jarosław Byrka, Krzysztof Fleszar, Bartosz Rybicki, and Joachim Spoerhase. Bi-factor approximation algorithms for hard capacitated k-median problems. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 722–736. SIAM, 2015a.

Jarosław Byrka, Krzysztof Fleszar, Bartosz Rybicki, and Joachim Spoerhase. Bi-factor approximation algorithms for hard capacitated k-median problems. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 722–736. SIAM, 2015b.

Moses Charikar, Sudipto Guha, Éva Tardos, and David B Shmoys. A constant-factor approximation algorithm for the k-median problem. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 1–10. ACM, 1999.

Brian F Cooper, Raghu Ramakrishnan, Utkarsh Srivastava, Adam Silberstein, Philip Bohannon, Hans-Arno Jacobsen, Nick Puz, Daniel Weaver, and Ramana Yerneni. Pnuts: Yahoo!'s hosted data serving platform. *Proceedings of the VLDB Endowment*, 1 (2):1277–1288, 2008.

Marek Cygan, MohammadTaghi Hajiaghayi, and Samir Khuller. Lp rounding for k-centers with non-uniform hard capacities. In *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*, pages 273–282. IEEE, 2012.

Sanjoy Dasgupta and Kaushik Sinha. Randomized partition trees for exact nearest neighbor search. *Algorithmica*, 72(1):237–263, 2015.

M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262, 2004.

Daniel Delling, Andrew V Goldberg, Ilya Razenshteyn, and Renato F Werneck. Graph partitioning with natural cuts. In *Parallel & Distributed Processing Symposium (IPDPS), 2011 IEEE International*, pages 1135–1146. IEEE, 2011.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874, 2008.

Sudipto Guha, Adam Meyerson, and Kamesh Munagala. Hierarchical placement and network design problems. In *FOCS*, pages 603–612. IEEE Computer Society, 2000. ISBN 0-7695-0850-2. URL http://dblp.uni-trier.de/db/conf/focs/focs2000.html#GuhaMM00.

Rishi Gupta, Tim Roughgarden, and C Seshadhri. Decompositions of triangle-dense graphs. In *Proceedings of the 5th conference on Innovations in theoretical computer science*, pages 471–482. ACM, 2014.

Kamal Jain, Mohammad Mahdian, Evangelos Markakis, Amin Saberi, and Vijay V Vazirani. Greedy facility location algorithms analyzed using dual fitting with factor-revealing lp. *Journal of the ACM (JACM)*, 50(6):795–824, 2003.

David R. Karger and Maria Minkoff. Building steiner trees with incomplete global knowledge. In *FOCS*, pages 613–623. IEEE Computer Society, 2000. ISBN 0-7695-0850-2. URL http://dblp.uni-trier.de/db/conf/focs/focs2000.html#KargerM00.

Samir Khuller and Yoram J. Sussmann. The capacitated k-center problem. In *In Proceedings of the 4th Annual European Symposium on Algorithms, Lecture Notes in Computer Science 1136*, pages 152–166. Springer, 1996.

Samory Kpotufe. The curse of dimension in nonparametric regression. 2010.

Robert Krauthgamer and James R Lee. Navigating nets: simple algorithms for proximity search. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 798–807. Society for Industrial and Applied Mathematics, 2004.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

Mu Li, David G Andersen, Alex J Smola, and Kai Yu. Communication efficient distributed machine learning with the parameter server. In *Advances in Neural Information Processing Systems*, pages 19–27, 2014.

Shanfei Li. An improved approximation algorithm for the hard uniform capacitated k-median problem. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2014, September 4-6, 2014, Barcelona, Spain*, pages 325–338, 2014a. doi: 10.4230/LIPIcs.APPROX-RANDOM.2014.325. URL http://dx.doi.org/10.4230/LIPIcs.APPROX-RANDOM.2014.325.

Shi Li. Approximating capacitated k-median with $(1+\epsilon)k$ open facilities. *arXiv preprint arXiv:1411.5630*, 2014b.

Gaëlle Loosli, Stéphane Canu, and Léon Bottou. Training invariant support vector machines using selective sampling. *Large scale kernel machines*, pages 301–320, 2007.

Mohammad Mahdian and Martin Pál. Universal facility location. In *Algorithms-ESA 2003*, pages 409–421. Springer, 2003.

Rafail Ostrovsky, Yuval Rabani, Leonard J Schulman, and Chaitanya Swamy. The effectiveness of lloyd-type methods for the k-means problem. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 165–176. IEEE, 2006.

Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

Ruth Urner, Shai Shalev-Shwartz, and Shai Ben-David. Access to unlabeled data can speed up prediction time. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 641–648, 2011.

Ruth Urner, Sharon Wulff, and Shai Ben-David. Plal: Cluster-based active learning. In *Conference on Learning Theory*, pages 376–397, 2013.

Vladimir N. Vapnik and Leon Bottou. Local algorithms for pattern recognition and dependencies estimation. *Neural Computation*, 1993.

Kai Wei, Rishabh K Iyer, Shengjie Wang, Wenruo Bai, and Jeff A Bilmes. Mixed robust/average submodular partitioning: Fast algorithms, guarantees, and applications. In *Advances in Neural Information Processing Systems*, pages 2233–2241, 2015.

Y. You, J. Demmel, K. Czechowski, L. Song, and R. Vuduc. CA-SVM: Communication-avoiding support vector machines on clusters. In *IEEE International Parallel and Distributed Processing Symposium*, 2015.

Yuchen Zhang, John C. Duchi, and Martin Wainwright. Communication-efficient algorithms for statistical optimization. In *Neural Information Processing Systems*, 2012.

Yuchen Zhang, John Duchi, Michael Jordan, and Martin Wainwright. Information-theoretic lower bounds for distributed statistical estimation with communication constraints. In *Neural Information Processing Systems*, 2013.