On Data Summarization for Machine Learning in Multi-Organization Federations

Bong Jun Ko*, Shiqiang Wang*, Ting He†, Dave Conway-Jones‡
*IBM T. J. Watson Research Center, Yorktown Heights, NY, USA
†Pennsylvania State University, University Park, PA, USA
‡Emerging Technology, IBM UK, Hursley Park, Winchester, UK

***Abstract*—Machine learning is a promising technology for many modern applications. To train an effective machine learning model, a large amount of data is required. However, data may be created in different organizations and sharing data across organizational boundaries is difficult due to privacy concerns and communication bandwidth limitations. Data summarization is a technique for reducing the amount of data that needs to be shared, while preserving characteristics in the data that are useful for training machine learning models. In this paper, we present an overview of data summarization techniques, which can be useful for machine learning across organizational boundaries. We also discuss some possible applications related to these data summarization techniques and challenges for future research.**

***Index Terms*—Data summarization, federation, machine learning**

I. Introduction

One of the main drivers for the recent advances in artificial intelligence and their applications is the availability of massive amount of data, with which machine learning (ML) algorithms, in particular deep learning (DL) techniques, can learn complex patterns and relationship within the data and apply the learned knowledge for previously unobserved data. However, the sheer volume of large data sets often renders the training of machine learning models difficult, if not infeasible, for the following reasons:

- Training deep learning models with a large amount of data requires equally large amount of computing power, which is not always available for the training process to finish within a reasonable time.
- The network bandwidth may not be sufficient to transfer a large volume of data from the point of data generation (typically at network edge) to where adequate amount of computing power is (typically at some centralized location with consolidated compute servers).
- It takes a lot of human efforts and time to create the training data for ML/DL by assigning the labels to the raw data set.

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

In multi-organizational coalition, where multiple parties in the coalition would like to share with one another their respective data sets and the knowledge learned from them, these problems are even more pronounced since the data exchange across the coalition boundaries are subject to additional constraints such as regulatory requirements and privacy/security issues.

There are three broad categories of machine learning federation, including output federation, model federation, and data federation. *Output federation* is the aggregation of outputs from multiple machine learning models which may be owned by different organizations. First, a data sample is sent to each model which computes the results locally. Then, the local outputs from all models are sent to an entity that aggregates these outputs and produces the final result. For instance, in the case of supervised learning, the aggregation can be performed by computing the average of the scores of each output label. In the output federation approach, the model always remains local, but there is no protection of the privacy of the data samples in the inference task. *Model federation* is to share and combine different pre-trained models. The combined model can then be sent to the party that needs it. In this way, the raw data samples used in inference can remain local. However, it is generally difficult to merge models after they are trained, and a model trained only using an organization's local dataset may be less accurate than a model that is trained using all organizations' datasets. *Data federation* extracts useful information from each organization's local dataset. Such information is usually less sensitive than the raw data, which can be shared with other organizations and used for training models collectively.

In this paper, we are primarily concerned with data federation, and explore data summarization approaches, generally refering to methods for representing a data set by a smaller amount of data than the original one, as remedies for the above issues. While the precise definition, the form of summarized data, and the techniques used vary by the objective and the application, we will focus on the aspects of different data summarization methods and their applicability to machine learning tasks in federated environments.

The rest of this paper is organized as follows. In Section II, we will review a few categories of data summarization methods in literature in the context of data federation for machine learning. Then, in Section III, we present how these data summarization methods can be applied to various scenarios of

multi-organizational federations of data and machine learning models, such as cross-organizational data brokerage, federated learning, and model searching. We outline some open research challenges in Section IV and draw the conclusion in Section V.

II. Data summarization techniques

There has been a wide variety of data summarization techniques proposed in literature and used in practice. This section serves as a brief survey of some of the representative approaches, with the focus on their relevance in machine learning in federated environments. We put them in three broad categories: (i) statistical summaries, (ii) dimensionality reduction, and (iii) sampling-based approaches.

A. Statistical summaries

This category of methods is typically originated from the need of summarizing data for efficiently exploring and analyzing a large amount of data. The focus is on time- and space-efficiency in creating and updating the aggregate summary, and the basic premise is that the summaries are created for specific types of queries on the data set, such as means, variance, count, and other statistical information.

The benefit of these approaches is that it requires only a small amount of time and space to generate the summary information: they can be typically created through only a single pass of the entire data set (and therefore adequate for streaming data as well), and have small memory footprints. The downside is that each summary can answer only a specific type of queries that it is designed for.

1) Summary statistics: Obviously the most basic form of statistical summarization is to aggregate the entire data set into a single number, such as sum, average, count, variance, median, mode, etc. If the entire data set is used to compute these statistical summaries, they provide the exact answer to the query that they are designed for. The time requirement is minimal, as these numbers are mostly calculated by a single pass through the data set and also can be updated incrementally.

The space requirement, however, varies by the type of the summary statistics. Some of them (e.g., sum, average, count, variance, min, max) can be trivially computed by simply updating a single number (or their combination) each time a new data point is encountered. However, finding exact values of some others requires maintaining larger space data structures and more involved operations on them; for example, the median or quartiles of a (multi)set of numbers shall be found in their sorted list, the mode (i.e., the most frequently occurring item) requires maintaining the a table of the items' occurrences.

Other statistical summaries are less trivial to obtain exactly in terms of their time and space requirements, for example, calculating range-sum, which is the sum (or the number) of items in a specific groups (or ranges), or the density function of the entire data set. For this reason, many approximate methods that estimate such summary statistics have been proposed in literature.

2) Histogram: The histogram, which summarize a dataset by putting them into discrete groups (or "buckets") and calculates/maintains some summary statistics of each group, has a long history as a method for data summarization and visualization. From the statistical viewpoint, a histogram can be viewed as a non-parametric, piece-wise continuous estimator of the density function of a dataset modeled as a collection of i.i.d. samples [1], and hence serves well as a general basis for estimating any summary statistics for the entire dataset based on the probability distribution.

If the bucket boundaries are known a priori, a histogram can be constructed in a single pass of the dataset, requiring $O(N)$ time in total for a data with N items, and $O(B)$ space with B buckets. Also, updating and deleting data items can be done in a straightforward manner. A classical, the most basic scheme for bucketizing is by dividing the entire range of the dataset into groups of equal width. While such a simple histogram can be constructed and maintained very efficiently, it can perform quite poorly in approximately answering the statistical queries, especially when the distribution of the data items within a single bucket is very skewed. An alternative to this basic equal-width model for avoiding the above issue is to construct the bucket boundaries so that each bucket contains the same number of items [2], generally called "equal-depth" histogram, albeit at the expense of additional computation required to sort the data or compute quantiles.

3) Sketches: Another, more recent class of approximately answering statistical queries on a large data set is *sketches*. The term itself actually is a loosely defined one, and generally refers to "a compressed mapping of the full data set onto a data structure which is easy to update with new or changed data, and allows certain queries whose results approximate queries on the full data set" [3]. In this notion, the summary statistics described above can be considered as a type of sketches, of which some statistical values like sum and average, as well as histograms with fixed bucket boundaries, are types of *linear* sketches (i.e., the mapping is a linear function of each data point) while some others are non-linear ones, like min and max.

Of particular interests to research work in the literature are sketches that require more involved and sophisticated methods and data structure for creating and maintaining the summary and answering (approximately) the queries. A classic example is Bloom Filter [4], which answers membership queries (i.e., "Is an item X is in the set S?") via a collection of bit-vectors constructed through a set of hash functions. Other, more advanced types of sketches have also been developed for solving a variety of problems that require the estimation of the frequencies of items in the set, such as Count Sketch [5], Count-Min Sketch [6], and AMS Sketch [7].

While these methods differ by the data structure employed, space requirements, supported types of queries, and estimation error bounds, they share the following common properties [8]:

- They have constant-size data structure, which is determined by one or more parameters related to the expected approximation error.

- Each update to the summary is independent of the past history, hence requiring each data sample needs to be seen only once.
- Most of the sketches, especially the non-trivial ones, involve some form of randomization in the form of random choice of hash functions, and hence the accuracy guarantees hold only in probabilistic sense.

Despite their benefits in space requirements and speed of answering queries, the statistical summaries of a data set described above–summary statistics, histogram, and sketches–have a major limitation in serving as the data summary for machine learning: Each of these methods can serve only a specific type of queries on the data set. In what follows, we review other classes of techniques that can serve as bases for more general ways of summarizing a large data set.

B. Dimensionality reduction

Data summarization through dimensionality reduction includes classes of approaches that map high-dimensional data onto a lower dimensional space such that certain properties of the original data set are preserved in the mapped space. Reducing dimension of the data does not decrease the cardinality of a data set; instead, it has the effect of reducing the total amount of data (i.e., the total number of bytes). Yet they also enable certain operations on the data set to be executed more efficiently (in time and space), including further summarization through other techniques in this section.

1) Principle component analysis (PCA): Probably the most well-known dimensionality reduction technique is Principal Component Analysis (PCA) [9], which converts a set of possibly correlated variables into a set of linearly uncorrelated ones. Though it is widely used in practice for data exploration as well as a pre-processing step in many data analysis tasks, its main limitation is the reliance on the linearity assumption and applicability only to numerical data, often rendering it inadequate for complex, unstructured data sets.

2) Locality-sensitive hashing (LSH): Locality-sensitive hashing refers to a class of data hashing techniques that map, with high probability, "similar" data items into the same "buckets" and "different" items into different buckets, to the effect of reducing the dimension of the data when the number of buckets is smaller than the that of data items. In particular, they enable approximate solutions to nearest neighbor (NN) search problems, which has linear complexity w.r.t. the number of items in a set and generally suffers from the curse of dimensionality. With LSH, an approximate nearest neighbor (ANN) search can be executed in sub-linear time, by limiting the search to the items within the same hash bucket for the given queried item.

One popular LSH method, called SimHash [10], maps similar data items in Euclidean space to lower-dimensional binary vectors (codewords) through a set of random projection, such that the Hamming distance between the binary codewords is small. Another popular method is via MinHash algorithm [11], which uses a signature vector from a set of randomly selected hash functions (from a family of hash functions) to map the high-dimensional data; The LSH is constructed then by a "banding" technique applied to this hash vector signature.[1]

The above hashing-based methods are data-independent, as random projections (for [10]) or random hash functions (for [11]) are selected independently of the data set being mapped to hash vectors. More recently, data-dependent methods are proposed, in which mapping functions of data samples to binary codes are learned by machine learning approaches. In particular, a deep learning model is used in [12] to learn the compact binary codes that can account for non-linear structure and relationship within the original data.

3) Feature extraction: The third class of dimensionality reduction methods is by using feature extraction functions learned from the given data set as the data summarization techniques; for example, a few layers of a convolutional neural network (CNN) model trained with the data set of interests, or the encoding layers of auto-encoders. The data summary in reduced dimension is therefore the set of feature maps corresponding to the original data set.

Strictly speaking, these feature extraction functions differ from the other approaches to data summarization in this section, in that the resulting set of feature vectors do not necessarily preserve the relationship between the data items in their original form; rather, the feature vectors represent their relationship w.r.t. minimizing a certain loss function of the labels associated with the raw data. Nonetheless, they are still useful, low-dimensional representations of the data for machine learning in distributed, federated environment, in which the feature maps in reduced dimension, instead of the raw data set, can be sent to and used by machine learning tasks.

C. Sampling from the original data space

The above two categories of data summarization converts the original data set to some lower-dimensional data space, with the statistical summaries being extreme cases of dimensionality reduction. In contrast, the methods in the third category presented here construct a small set of the data samples within the sample space of the original data set, hence enabling the use of a small data set as a proxy for the original one in machine learning tasks.

1) Random sampling: A straightforward approach is to randomly sample from the original dataset, where the sampling can be uniform, stratified, etc. Despite being simple, this approach does not provide any guarantee on how the sampling affects the resulting machine learning model trained on the sampled data.

2) Model consistent sampling: An improvement to the random sampling is to strategically select the samples with respect to their performance under the model that uses the samples as the training set. An early work of this concept,

[1] MinHash itself is used to approximately computing the similarity between two sets, which in itself is useful in some of the applications we discuss in Section III-C

called Condensed Nearest Neighbors (CNN) [13],[2] incrementally selects a training data set from the original data set by repeatedly selecting samples mis-classifed by the assumed model (k-Nearest Neighbor in this particular case) trained with already-selected samples, until no more samples can be moved into the training set. This work and its subsequent variants would always form a compressed training set that is always a subset of the original data set. More recently, approaches that learn synthetic data set as the representative summary are also proposed, such as Stochastic Neighbor Compression (SNC) [14], which uses stochastic neighborhood [15], instead of deterministic nearest neighbor models, as a probabilistic model for assessing correct/incorrect classification of the data samples in the process of data selection process.

3) Coreset: Coreset construction is an approach that selects a subset of samples from the training dataset, so that training a machine learning model only using this subset (coreset) of samples (where weights are given to each sample in the coreset) approximates the training with all the data samples with a theoretically provable error [16], [17]. Intuitively, each data sample in the coreset represents a group of data samples in the original dataset. For a given type of machine learning model, there is a specific way of constructing the coreset. This approach is most effective with shallow models such as K-means and K-median, where the cost function used in model training is largely based on the Euclidean distance between data samples.

4) Active learning: For supervised learning tasks, it is often labor intensive to collect labeled training data. Active learning is an approach that interacts between the machine learning task and the human labeling effort [18]. Assume that all the data samples are unlabeled at the beginning. The system first randomly selects a small subset of samples for the human to label. It learns an initial model based on this small subset of labeled data. Using this model, the system identifies which other samples among the unlabeled data samples most likely contain useful information for improving the model's accuracy, and it asks the human to label these data samples. Then the model is updated using the newly labeled data. This process repeats until some desired model accuracy has been reached. In this way, the amount of labeled (sampled) data increases over time up to a certain amount. Depending on the desired model accuracy, the amount of labeled data may be significantly less than the total amount of data, which saves human efforts of data labeling.

III. Applications of data summarization

In the following, we summarize some applications of data summarization and related aspects for machine learning in federated environments.

A. Federated learning

Federated learning is a way of training machine learning models from multiple decentralized clients [19], such as mobile phones, micro-servers, etc. Here, different clients may belong to different users or different organizations. The data is stored at each client locally, and a server coordinates the distributed training process. When training a model using federated learning, only the model parameters (or gradients of model parameters) are shared between each client and the server. The raw data at each local client does not need to be shared with any other entity, which preserves the privacy of raw data and often reduces the communication bandwidth requirement as the size of raw data is usually very large.

The basic procedure of federated learning is as follows:

1) The server sends the model parameter vector (e.g., weights of a neural network) to all the clients. Upon initialization, the model parameter is initialized either randomly or as a constant. In all other rounds, the model parameter is the aggregated value obtained in the previous round (Step 4).
2) Each client computes the gradient of the model parameter (with respect to the loss function defined for the machine learning model), based on the most recent model parameter and its own local dataset. Then, it performs one step of gradient descent iteration and updates its own (local) model parameter. This step can be repeated for τ times where τ is either a predefined parameter or can be chosen using an optimization procedure [20], [21].
3) Each client transmits its most recent local model parameter to the server.
4) The server computes the aggregated (global) model parameter, often as the simple average or weighted average of local model parameters.
5) Repeat from Step 1.

Data summarization can assist federated learning in several ways. As discussed in [22], [23], a summary (or small subset) of the local raw dataset may be shared with other clients, to improve the training efficiency when the datasets at different clients are non-i.i.d. distributed. In addition, federated learning may be performed on a summary of local datasets of each client, instead of performing on the raw dataset directly. This can improve the efficiency and stability of federated learning. For example, if the goal is image classification, one may use a common feature extractor for images as a dimension reduction technique, and train the machine learning model using federated learning only on the features. This approach is also related to federated transfer learning [24]. If some data collection devices (such as IoT sensors) are not capable of training advanced machine learning models, they may send a summary of their data to a more powerful device (such as a home gateway) which will act as a client in federated learning.

Furthermore, for a given global model parameter, the incremental update computed at each client (i.e., the difference between the updated local model parameter and the original global model parameter) can also be seen as a summarization of the client's local dataset, for the purpose of training the machine learning model, where the structure (logic) of the machine learning model is predefined.

[2]Not to be confused with Convolutional Neural Network

B. Data brokerage

In environments where datasets are owned by different organizations, there needs to be a mechanism to reward each organization that provides data, if the data has provided value for some machine learning task. A data broker that coordinates the data summarization and sharing among different clients can be useful in this setting. The data broker receives a request from a task requester, which can be a representative from an organization, and selects one or multiple participating clients which have data that is potentially valuable for the requested task. It then coordinates these participating clients towards the completion of the task.

An important functionality of a data broker is to quantify the value of a dataset provided by a client. Such values are often task-dependent and also depends on the datasets (and the values of those datasets) provided by other clients. For example, in federated learning which is based on gradient descent, the value of each client can be quantified by how much the aggregated gradient changes if this client is removed. If the gradient does not change at all after removing a particular client, then it is very likely that this client has a dataset that is similar to one of the other clients, and excluding this client from the task would not cause too much adverse effect, because the learning would still progress in a similar manner since the gradient remains similar.

Each client may offer a price that the task requester has to pay in order for this client to participate in the task. There can be price negotiations between the requester and each client, which are coordinated by the data broker. The price negotiation can depend on the client's value to the task as explained above.

The value of each client's dataset may change during the progression of the task. For example, in federated learning, the model may first learn the coarse differences between different images, for which minor differences in different clients' datasets may not matter much. So, initially, the system may want to select a small subset of clients to participate in the task and pay a low cost. As the learning progresses, the diversity of data may become more important, and the system may need to select a larger subset of clients and pay a higher cost. For this reason, the valuation of different clients and price negotiation may happen for multiple times during task execution. The data broker needs to continuously monitor the changes in values and trigger a new round of negotiation when necessary. Data summarization helps the data broker and clients perform these tasks in a manner that requires less resources (in network and computation) and preserves the sensitivity of the raw data.

C. Model searching

In a multi-organizational environment, each organization may train and store different machine learning models using different datasets for different purposes. The trained models can be useful for other organizations as well, in which case the model created by one organization may be shared with other organizations and the provider of the model would get rewarded in some way. In order for this kind of model sharing to work, there needs to be some mechanism for figuring out which model is the most suitable for a given task.

The first step of searching potentially suitable models is to match some basic specifications of each model with the task. For example, it is obvious that a model trained for sound classification cannot be used for image classification, or vice versa. A subset of models (possibly belonging to different organizations) that process the same type of data and give the same or similar sets of labels as output can be selected from this process. These models will be further considered for use with the given task.

The task specification should include a test dataset which includes pairs of input (raw) data sample and desired output label, because without this test dataset, all the models that match with the high-level specification of the task would be considered as having similar values, as there is no way to quantify the difference of these models. With the test dataset available, there are several ways of quantifying the values of different models for this task.

One straightforward way is to evaluate the model directly with the test data and compare the accuracy provided by different models. However, if the test dataset is not large enough, the accuracy results may not have enough statistical significance and comparing them is not meaningful.

Another approach is to compute summaries of both the test data of the task and the training data used to train the machine learning model. Then, compare the summary of the training data for each model with the summary of the test data. As explained earlier in this paper, some types of data summarization techniques such as projection to lower dimensional space transform the raw data into a space where the distance reflects the inherent similarity between different data samples better than the distance on the raw data itself.

A third approach for quantifying the suitability of a model to the test data is to compute the gradient of the model parameter with respect to the loss function of the model on the test dataset. The gradient computation is similar to what is usually done in model training (see Section III-A), where the only difference is that the gradient is computed on the training dataset for model training whereas we compute it on the test dataset here. Since a model is trained well when the gradient tends to vanish, the model is suitable to the test dataset if the norm of the gradient is small.

IV. Research challenges

Data summarization is a useful technique for sharing data across organizational boundaries, so that the data collected by different organizations can be utilized for machine learning. However, there exist several challenges ahead.

As summarized in Section II, there exist a large variety of data summarization techniques with different complexities. For a given machine learning problem, one needs to choose suitable data summarization techniques that can be useful for this problem. Currently, this choice is usually made by a human based on his/her understanding of the problem characteristics. In a large-scale distributed machine learning platform,

there are many participating organizations and different types of machine learning models. Relying on human efforts to choose appropriate data summarization approaches for each individual machine learning task that occurs in such a platform is infeasible. Therefore, future research can investigate on how to automate the choice of data summarization for a given machine learning task and a family of available datasets possibly from different organizations.

A main promise of data summarization is to preserve the privacy of sensitive information in each organization's raw data. Therefore, a natural question that one would ask is to what the degree does data summarization provide privacy protection. While there has been some efforts towards this direction from a differential privacy's point of view [25], [26], more needs to be done so that users can clearly understand the privacy implications of different data summarization approaches and use this knowledge to properly configure a system for distributed data sharing and machine learning. A related goal here is to provide interpretable metrics for different privacy characterizations that exist in the literature [27].

Lastly, we can intuitively expect that in general, if we share less amount of data (i.e., the degree of compression of the data summarization technique is higher), we preserve more privacy and save more communication bandwidth, but the utility for the machine learning problem also decreases. There exists a complex interplay among the amount of shared data (which is equal to the data transmission size), the degree of privacy protection, and the utility for machine learning. Theoretical or empirical models that capture this tradeoff would be useful for choosing the best settings for a given set of privacy and communication constraints. It is challenging to obtain such models due to the large variety of data summarization techniques and machine learning models.

V. Conclusion

In this paper, we have presented an overview of different data summarization techniques and their applications to machine learning. In general, data summarization can be useful for machine learning with data collected by multiple organizations, which preserves the privacy of organizational data, saves the communication bandwidth, and may also improve the computational efficiency. There exist several challenges before reaching the ultimate goal of developing a large-scale data sharing and distributed learning platform. This paper has provided some insight towards this goal.

References

[1] D. W. Scott, *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons, 2015.

[2] G. Piatetsky-Shapiro and C. Connell, "Accurate estimation of the number of tuples satisfying a condition," *ACM Sigmod Record*, vol. 14, no. 2, pp. 256–276, 1984.

[3] J. M. Phillips, "Coresets and sketches," *arXiv preprint arXiv:1601.00617*, 2016.

[4] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[5] M. Charikar, K. Chen, and M. Farach-Colton, "Finding frequent items in data streams," in *International Colloquium on Automata, Languages, and Programming*. Springer, 2002, pp. 693–703.

[6] G. Cormode and S. Muthukrishnan, "An improved data stream summary: the count-min sketch and its applications," *Journal of Algorithms*, vol. 55, no. 1, pp. 58–75, 2005.

[7] N. Alon, Y. Matias, and M. Szegedy, "The space complexity of approximating the frequency moments," *Journal of Computer and system sciences*, vol. 58, no. 1, pp. 137–147, 1999.

[8] G. Cormode, M. Garofalakis, P. J. Haas, C. Jermaine *et al.*, "Synopses for massive data: Samples, histograms, wavelets, sketches," *Foundations and Trends® in Databases*, vol. 4, no. 1–3, pp. 1–294, 2011.

[9] I. Jolliffe, *Principal component analysis*. Springer, 2011.

[10] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," *Communications of the ACM*, vol. 51, no. 1, p. 117, 2008.

[11] A. Z. Broder, "On the resemblance and containment of documents," in *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171)*. IEEE, 1997, pp. 21–29.

[12] V. Erin Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou, "Deep hashing for compact binary codes learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 2475–2483.

[13] P. Hart, "The condensed nearest neighbor rule (corresp.)," *IEEE transactions on information theory*, vol. 14, no. 3, pp. 515–516, 1968.

[14] M. Kusner, S. Tyree, K. Weinberger, and K. Agrawal, "Stochastic neighbor compression," in *International Conference on Machine Learning*, 2014, pp. 622–630.

[15] G. E. Hinton and S. T. Roweis, "Stochastic neighbor embedding," in *Advances in neural information processing systems*, 2003, pp. 857–864.

[16] D. Feldman and M. Langberg, "A unified framework for approximating and clustering data," in *Proceedings of the forty-third annual ACM symposium on Theory of computing*. ACM, 2011, pp. 569–578.

[17] V. Braverman, D. Feldman, and H. Lang, "New frameworks for offline and streaming coreset constructions," *arXiv preprint arXiv:1612.00889*, 2016.

[18] B. Settles, "Active learning literature survey," University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 2009.

[19] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2016.

[20] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, pp. 1–1, 2019.

[21] T. Tuor, S. Wang, T. Salonidis, B. J. Ko, and K. K. Leung, "Demo abstract: Distributed machine learning at resource-limited edge nodes," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, April 2018, pp. 1–2.

[22] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[23] J. Park, S. Samarakoon, M. Bennis, and M. Debbah, "Wireless network intelligence at the edge," *arXiv*, Dec. 2018. [Online]. Available: http://arxiv.org/abs/1812.02858

[24] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, p. 12, 2019.

[25] Z. Ji, Z. C. Lipton, and C. Elkan, "Differential privacy and machine learning: a survey and review," *arXiv preprint arXiv:1412.7584*, 2014.

[26] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 308–318.

[27] C. Liu, X. He, T. Chanyaswad, S. Wang, and P. Mittal, "Investigating statistical privacy frameworks from the perspective of hypothesis testing," in *Privacy Enhancing Technologies Symposium (PETS)*, Jul. 2019.