Research Plan

Large Scale Submodular Summarization

PhD student: Baharan Mirzasoleiman *
Supervisor: Andreas Krause

October 21, 2014

Data sets are getting larger and more difficult to process. An important goal in machine learning and information retrieval, therefore, is to develop novel techniques that extract and summarize useful information from massive data, while still remaining computationally tractable. A systematic way for data summarization is to turn the problem into selecting a subset of data elements optimizing a utility function that quantifies "representativeness" of the selected set. Often-times, these objective functions satisfy submodularity. Hence, such problems can be reduced to maximizing a submodular set function subject to cardinality or other feasibility constraints; and dealing with big data means we have to solve this problem at scale. The other characteristic common to many large datasets which makes computing with them daunting is that they are evolving over time. The classical computational paradigm, which assumes a fixed data as an input to an algorithm that terminates, is insufficient for many modern data processing needs. In this thesis, we consider the problem of selecting representative elements from very large evolving data sets and develop efficient algorithms with constant factor approximation guarantee to the optimum solution. In our experiments, we extensively evaluate the effectiveness of our approach for several real-world applications on millions of data points.

Keywords: Summarization, Submodular Functions, Approximation Algorithms, Distributed Computing, Greedy Algorithms, Map-Reduce

*Baharan Mirzasoleiman was accepted as a PhD student at ETH Zurich in October 2013. He started to work as a research assistant at ETH Zurich from February 2, 2013.

1 Introduction

The unprecedented growth in modern datasets – coming from different sources and modalities such as images, videos, sensor data, social networks, etc. – demands novel techniques that extract useful information from massive data, while still remaining computationally tractable.

One recent approach for efficiently extract and summarize relevant and useful information gaining some prominence in machine learning is based on submodular functions. Summarization problems can often be reduced to the problem of maximizing a submodular set function subject to cardinality or other feasibility constraints such as matroid, or knapsack constraints [10].

Submodularity is a property of set functions with deep theoretical and practical consequences. Submodular functions exhibit a natural diminishing returns property: the marginal benefit of any given element decreases as we select more and more elements. Submodular maximization generalizes many well-known problems, including maximum weighted matching, maximum coverage, and finds numerous applications in machine learning and social networks: viral marketing, information gathering, document summarization, and active learning. Although maximizing a submodular function is NP-hard in general, a seminal result of Nemhauser et al. [16] states that a simple greedy algorithm produces solutions competitive with the optimal (intractable) solution.

However, two characteristics common to real-world datasets make computing with them daunting: their scale and, often, their evolving and decentralized nature. The classical computational paradigm such as the greedy algorithm (and other standard algorithms for submodular optimization), assumes a fixed data as an input to an algorithm and requires random access to the data. Hence, while it can easily be applied if the data fits in main memory, it is impractical for data residing on disk, or arriving/changing over time at a fast pace. In many domains, data volumes are increasing faster than the ability of individual computers to store them in main memory. In some cases, data may be produced so rapidly that it cannot even be stored. In other cases, underlying data changes over time and an algorithm, aware of these changes, needs to produce a different answer for each time step.

A recent direction in processing large-scale data is to make use of *parallelism*: divide and process the data over many machines and merge the final results in a reliable way. MapReduce [6] is arguably one of the most successful programming models for reliable and efficient parallel computing. Another natural approach to scale up the classical computational paradigm, is to use streaming algorithms. In fact, in applications where data arrives at a pace that does not allow even storing it, this is the only viable option. However, for evolving datasets, the input data changes and the algorithm, aware of these changes, must be able to make appropriate changes in the result, or predict the result of the next time step.

In this thesis, we investigate methods for summarizing large-scale datasets both in distributed and streaming settings, and try to devise methods that are able to make appropriate changes in the summary of evolving datasets, or predict the summary of the next time step. In our experiments, we extensively evaluate the effectiveness of our

approaches on several submodular maximization problems.

2 State-of-the-Art

2.1 Submodularity in data mining

In order to effectively summarize the data, we need to define a measure for the amount of *representativeness* that lies within a selected set. If we think of representative elements as the ones that cover best the items in a dataset (or equivalently the most informative ones) then naturally adding a new element to a set of representatives, say A, is more informative than adding it to its superset, say $B \supseteq A$, as the new element can potentially enclose more uncovered items when considered with elements in A rather than B. This intuitive *diminishing return* property can be systematically formalized through *submodularity* (c.f., [16]). More precisely, a submodular function $f : 2^V \to \mathbb{R}$ assigns a subset $A \subseteq V$ a utility value $f(A)$ –measuring the representativeness of the set A– such that

$$f(A \cup \{i\}) - f(A) \geq f(B \cup \{i\}) - f(B)$$

for any $A \subseteq B \subseteq V$ and $i \in V \setminus B$. Note that $\Delta(i|A) \doteq f(A \cup \{i\}) - f(A)$ measures the marginal *gain* of adding a new element i to a summary A. Of course, the meaning of representativeness (or utility value) depends very much on the underlying application; for a collection of random variables, the utility of a subset can be measured in terms of entropy, and for a collection of vectors, the utility of a subset can be measured in terms of the dimension of a subspace spanned by them. In fact, summarization through submodular functions has gained a lot of interest in recent years with application ranging from exemplar-based clustering, to document and corpus summarization , to recommender systems.

Since we would like to choose a summary of a manageable size, a natural optimization problem is to find a summary A^* of size at most k that maximizes the utility, i.e.,

$$A^* = \arg\max_{A:|A|\leq k} f(A). \tag{2.1}$$

Unfortunately, this optimization problem is NP-hard for many classes of submodular functions. We say a submodular function is *monotone* if for any $A \subseteq B \subseteq V$ we have $f(A) \leq f(B)$. A celebrated result of Nemhauser et al. [16] –with great importance in artificial intelligence and machine learning– states that for non-negative monotone submodular functions a simple *greedy algorithm* provides a solution with $(1 - 1/e)$ approximation guarantee to the optimal (intractable) solution. This greedy algorithm starts with the empty set A_0 and in iteration i, adds an element maximizing the marginal gain $\Delta(e|A_{i-1})$.

2.2 Scaling up: lazy implementation

For a ground set V of size n, the greedy algorithm needs $O(n \cdot k)$ function evaluations in order to find a summarization of size k. However, in many data intensive applica-

tions, evaluating f is expensive and running the standard greedy algorithm is infeasible. Fortunately, submodularity can be exploited to implement an accelerated version of the classical greedy algorithm, usually called LAZY GREEDY [13]. Instead of computing $\Delta(e|A_{i-1})$ for each element $e \in V$, the LAZY GREEDY algorithm keeps an upper bound $\rho(e)$ (initially ∞) on the marginal gain sorted in decreasing order. In each iteration i, the LAZY GREEDY algorithm evaluates the element on top of the list, say e, and updates its upper bound, $\rho(e) \leftarrow \Delta(e|A_{i-1})$. If after the update $\rho(e) \geq \rho(e')$ for all $e' \neq e$, submodularity guarantees that e is the element with the largest marginal gain. Even though the exact cost (i.e., number of function evaluations) of LAZY GREEDY is unknown, this algorithm leads to orders of magnitude speedups in practice. As a result, it has been used as the state-of-the-art implementation in numerous applications including network monitoring, network inference, document summarization, and speech data subset selection, to name a few. However, as the size of the data increases, even for small values of k, running LAZY GREEDY is infeasible. To solve the optimization problem (2.1) at scale, there has been very recent efforts to either make use of parallel computing methods or treat data in a streaming fashion.

2.3 Scaling up: distributed algorithms

Due to the rapid increase in data set sizes, and the relatively slow advances in sequential processing capabilities of modern CPUs, parallel computing paradigms have received much interest. Inhabiting a sweet spot of resiliency, expressivity and programming ease, the MapReduce style computing model [6] has emerged as prominent foundation for large scale machine learning and data mining algorithms. A MapReduce job takes the input data as a set of $< key; value >$ pairs. Each job consists of three stages: the *map* stage, the *shuffle* stage, and the *reduce* stage. The map stage, partitions the data randomly across a number of machines by associating each element with a *key* and produce a set of $< key; value >$ pairs. Then, in the shuffle stage, the value associated with all of the elements with the same key gets merged and send to the same machine. Each reducer then processes the values associated with the same key and outputs a set of new $< key; value >$ pairs with the same key. The reducers' output could be input to another MapReduce job and a program in MapReduce paradigm can consist of multiple rounds of map and reduce stages.

Recent works on submodular optimization in distributed settings has focused on specific instances. Such scenarios often occur in large-scale graph mining problems where the data itself is too large to be stored on one machine. In particular, [5] address the MAX-COVER problem and provide a $(1 - 1/e - \epsilon)$ approximation to the centralized algorithm at the cost of passing over the data set many times. Their result is further improved by [4]. [12] addressed more general graph problems by introducing the idea of filtering, namely, reducing the size of the input in a distributed fashion so that the resulting, much smaller, problem instance can be solved on a single machine. But, there is no general framework for maximizing a submodular function in distributed settings where performance competitive with the centralized setting can be obtained.

2.4 Streaming algorithms for submodular maximization

Another natural approach to scale up submodular optimization, is to use streaming algorithms. In fact, in applications where data arrives at a pace that does not allow even storing it, this is the only viable option. Recently, there have been tries for maximizing submodular functions in streaming fashion. However, they all have some drawbacks. Some of them make strong assumptions about the way the data stream is generated, and unless their assumptions are met, it is fairly easy to construct examples where the performance of the algorithm degrades quickly when compared to the optimum solution. Furthermore, the update time (computational cost to process one data point) of their approach is $\Omega(k)$, which is prohibitive for large k [7]. The claimed guarantees for the other algorithms depend on the maximum increase in the objective any element can offer. Moreover, they have issues with the number of passes they need to make over the datasets, or has a memory requirement depending on the data size n [11].

There is further related work on the *submodular secretary problem* [3, 8]. While also processing elements in a stream, these approaches are different in two important ways: (i) they work in the stronger model where they must either commit to or permanently discard newly arriving elements; (ii) they require *random* arrival of elements, and have a worse approximation ratio (≤ 0.1 vs. $1/2 - \epsilon$). If elements arrive in arbitrary order, performance can degrade arbitrarily. Some approaches also require large (i.e., $O(n)$) memory.

2.5 Algorithms for Evolving Datasets

Most statistical and machine learning algorithms assume that the data is a random sample drawn from a stationary distribution. Unfortunately, most of the large databases available for mining today violate this assumption. They were gathered over months or years, and the underlying processes generating them changed during this time, sometimes radically [9]. Today time series data are being generated at an unprecedented speed from almost every application domain, e.g., daily fluctuations of stock market, traces of dynamic processes and scientific experiments, medical and biological experimental observations, various readings obtained from sensor networks, position updates of moving objects in location-based services etc. As a consequence, in the last decade there has been a dramatically increasing amount of interest in querying and mining such data which, in turn, resulted in a large amount of work introducing new methodologies for indexing, classification, clustering and approximation of time series [17].

One specific domain of interest are graphs. Graphs are ubiquitous in today's world. Such graphs range from social networks to recommendation networks to communication and information networks to hyperlink-induced web graphs. A notable characteristic of these activity networks, is that the structure of the graphs changes over time (e.g.,as people communicate with different friends in a social network, or articles are added to Wikipedia). To address some of the challenges of interest that are not captured by the classical algorithms, computational frameworks have been proposed on graphs where the underlying structure changes over time and an algorithm can only observe these changes

by probing the graph in a limited way [2]. However, the results are limited to two basic graph connectivity questions, namely, path connectivity and finding minimum spanning trees (MST).

3 Goals of Thesis

The main goal of this thesis is to develop efficient methods for summarizing very large datasets. In addition, we try to devise dynamic methods for summarizing large evolving datasets. We develop methods that are able to modify and not recalculate the summary of changing datasets, and use the results to predict their summary in the future. In particular, we are interested in addressing the following two problems:

3.1 Scaling up algorithms for submodular maximization

As discussed in section 2, classical approaches for cardinality-constrained submodular optimization, such as the celebrated greedy algorithm of Nemhauser et al. [16], or its accelerated variants [13] require random access to the data. Once the size of the dataset is very large –possibly increases beyond the memory capacity– (typical in many modern datasets) or the data is arriving incrementally over time, neither the greedy algorithm, nor its accelerated versions can be used.

3.1.1 Distributed submodular maximization

One possible approach to scale up submodular optimization is to distribute data to several machines, and seek parallel computation methods. However, in a distributed environment, a direct adaptation of these algorithms will require the machines to proceed in lock-step to identify the globally best element in each iteration, and hence lose the benefits of multiple machines and parallel processing. This mismatch makes it inefficient to apply classical algorithms directly to parallel setups. Therefore, the first goal of the thesis would be to developed efficient distributed protocols for maximizing a submodular function subject to cardinality constraints. This provides an important step towards solving submodular optimization problems in very large scale, real applications.

3.1.2 Streaming submodular maximization

Another solution is to use streaming algorithms where at any point of time the algorithm has access to only a small fraction of data stored in primary memory. This approach not only avoids the need for vast amounts of random-access memory but also provides predictions in a timely manner based on the data seen so far, facilitating real-time analytics. A naive way to implement it in a streaming fashion, when the data is static and does not increase over time, is to pass k times over the ground set and at each iteration select an element with the maximum marginal gain. However, if the data size increases over time (e.g., new elements are added to a log, video is being recorded, etc.) we can no longer implement this naive approach. Thus, the second goal is to develop

streaming algorithms for submodular maximization subject to cardinality constraints. Given that streaming methods are sometimes the only viable method for solving very large scale or streaming applications, this result would be very useful to numerous data mining and machine learning applications.

3.1.3 Randomized submodular maximization

An even more natural effort to solve submodular maximization at scale is to develop randomized algorithms to speed up the greedy algorithm. Even though the exact cost (i.e., number of function evaluations) of LAZY GREEDY is unknown, this algorithm leads to orders of magnitude speedups in practice. As a result, it has been used as the state-of-the-art implementation in numerous applications. However, as the size of the data increases, even for small values of k, running LAZY GREEDY is infeasible. Therefore, a natural question to ask is whether it is possible to further accelerate LAZY GREEDY by a procedure with a weaker dependency on k. Or even better, is it possible to have an algorithm that does not depend on k at all and scales linearly with the data size n? Our next step would be to develop randomized algorithms for submodular maximization. In general, any distributed algorithm that uses LAZY GREEDY as a sub-routine, can directly benefit from this method and provide even more efficient large-scale algorithmic frameworks.

3.2 Submodular maximization for Evolving datasets

In the second part of the thesis we consider large evolving datasets. Many datasets are evolving, i.e. data elements are being added, removed or changed over time. For these datasets, the standard greedy algorithm which assumes a fixed data as an input cannot be applied, as it needs to process the whole dataset and recalculate the result after each change occurred. This makes it inapplicable for very large datasets.

3.2.1 Dynamic submodular maximization

For evolving or dynamic datasets, the underlying data changes and we need faster algorithms which are able to update and modify the existing summary instead of recalculating it from the scratch. Algorithms that make use of previous solutions and, thus, solve the problem faster than recomputation are called fully dynamic algorithms. Hence, in the second part our goal is to develop fully dynamic algorithms for extracting the summary of large dynamic datasets.

3.2.2 Predictive submodular maximization

Moreover, in many applications dealing with dynamic data we would like to know how the summary looks like in the future. In this setting, we need a probabilistic model and an algorithm able to learn the corresponding parameters and make predictions. In this part, we aim at investigating efficient algorithms able to use the summary of the data in the past (obtained by solving a submodular maximization problem), in order to predict

the result of the submodular summarization problem in the future. The constraint is that at most k points are selected at every point in the time.

4 Detailed Work Plan

1. *Scaling up: Distributed submodular maximization algorithms.* As discussed in section 2, for truly large-scale problems, neither the greedy algorithm, nor its accelerated versions can be used. In the first part of this thesis, we consider the problem of submodular function maximization in a distributed fashion. We develop methods with minimal communication that are easily implemented in MapReduce style parallel computation models. We theoretically analyze our approach and extensively evaluate our algorithm on several large scale machine learning problems using Hadoop.

2. *Scaling up: Streaming submodular maximization algorithms.* In many today's data mining and machine learning applications, data arrives in a stream (e.g., activity logs, video streams), possibly in a pace that precludes storage. Hence, in such applications, we seek methods that can process quickly arriving data in a timely manner. In the second part of this thesis, we develop streaming algorithms with a limited memory available to them (much less than the ground set) and limited processing time per item. Such algorithms access only a small fraction of data at any point in time and provide approximate solutions.

3. *Scaling up: Randomized submodular maximization algorithms.* Another interesting direction is to develop randomized methods for maximizing a monotone submodular function faster than the widely used LAZY GREEDY algorithm (also known as accelerated greedy). In the next part of the thesis, we plan to develop linear-time algorithm for maximizing a general monotone submodular function subject to a cardinality constraint. More specifically, our goal is to develop centralized algorithm whose cost (i.e., number of function evaluations) is independent of the cardinality constraint, which in turn directly addresses the shortcoming of LAZY GREEDY. These methods can be easily integrated into existing distributed methods and enable us to produce the summary of much larger datasets.

4. *Dynamic datasets: Dynamic submodular maximization algorithms.* For evolving and decentralized nature and, often, large scale datasets, such as evolving graphs, changing text corpora or dynamic set of images, none of the above models by itself is sufficient to produce a good solution since they only work on static (i.e., not evolving) input. If the underlying problem instance changes slightly, algorithms are needed that quickly compute the summary in the modified data.To find a summary of evolving and massive datasets, we investigate fast methods able to modify the result of the last time step instead of recalculating the result by running the greedy algorithm or its accelerated versions on the entire data.

5. *Dynamic datasets: Predictive submodular maximization algorithms.* In the last part of this thesis, we try to answer another interesting question: having a large, evolving dataset, is it possible to predict how the summary looks like in the future? More specifically, having the summary of the dataset for a few recent time steps, we want to predict what would be the most representative elements in the next time step. To do so, will try to build appropriate graphical models and predict the result of a submodular maximization problem using techniques from probabilistic modeling.

6. *Dynamic datasets: Evaluating dynamic submodular maximization algorithms on large data sets.* Wikipedia is a free-access, free content Internet encyclopedia. Wikipedia is one of the sixth-most popular website and constitutes the Internet's largest and most popular general reference work. As of February 2014, it had 18 billion page views and nearly 500 million unique visitors each month. Anyone who can access the site can edit almost any of its articles. Wikipedia has earned a reputation as a comprehensive source for knowledge because of its rapid updating of articles. Therefore, it is a good example of a large evolving corpus. In order to empirically test the performance of our algorithms, we calculate the summary of one snapshot and use dynamic algorithms to quickly modify the result and compute the summary of the following snapshots. Then, we use the obtained summaries to build a probabilistic model and predict the important articles in the future.

5 Progress to Date

- To deal with large-scale data, we developed a simple, two-stage protocol GreeDi, that is easily implemented using MapReduce style computations. We theoretically analyze our approach, and show that under certain natural conditions, performance close to the (impractical) centralized approach can be achieved. Beyond monotone submodular maximization subject to a cardinality constraint, We extended our approach to obtain approximation guarantees for (not necessarily monotone) submodular maximization subject to more general constraints including matroid or knapsack constraints. In our extensive experiments, we demonstrated the effectiveness of our approach on several applications, including sparse Gaussian process inference and exemplar based clustering on tens of millions of examples using Hadoop [15].

- To deal with streaming data, we developed the first efficient streaming algorithm with constant factor $1/2 - \varepsilon$ approximation guarantee to the optimum solution, requiring only a single pass through the data, and memory independent of data size. In our experiments, we extensively evaluated the effectiveness of our approach on several applications, including training large-scale kernel methods and exemplar-based clustering, on millions of data points. We observed that our streaming method, while achieving practically the same utility value, runs about 100 times faster than the existing works [1].

- In order to maximize a general monotone submodular function faster than the widely used LAZY GREEDY algorithm we developed the first linear-time algorithm, RAND-GREEDY. We showed that RAND-GREEDY achieves a $(1 - 1/e - \epsilon)$ approximation guarantee to the optimum solution with running time $O(n \log(1/\epsilon))$ (measured in terms of function evaluations) that is *independent* of the cardinality constraint k. Our experimental results on exemplar-based clustering and active set selection in nonparametric learning also confirmed that RAND-GREEDY consistently outperforms LAZY GREEDY by a large margin while achieving practically the same utility value. More surprisingly, in all of our experiments we observed that RAND-GREEDY does not even evaluate all the items and shows a running time that is less than n while still providing solutions close to the ones returned by LAZY GREEDY. Due to its independence of k, RAND-GREEDY is the first algorithm that truly scales to voluminous datasets [14].

6 Time Schedule

The work on this PhD thesis was started in February 2013. We plan to finish the thesis in early 2017. The time schedule of the individual work packages can be found in Figure 1, where the numbering corresponds to the one given in section 4. The winter of 2016 will be mostly devoted to writing up the thesis.

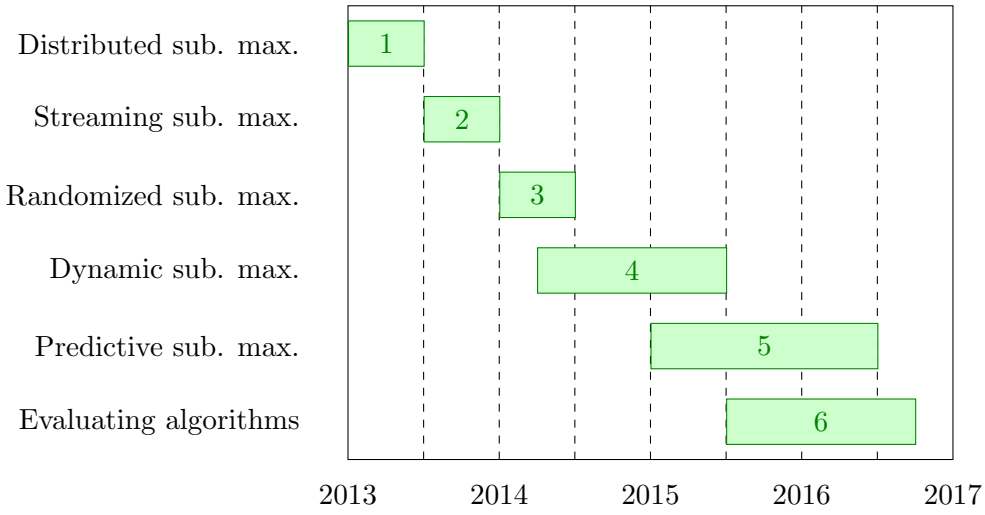

Figure 1: Time schedule of the thesis.

References

[1] A. Badanidiyuru, B. Mirzasoleiman, A. Karbasi, and A. Krause. Streaming submodular maximization: Massive data summarization on the fly. 2014.

[2] B. Bahmani, R. Kumar, M. Mahdian, and E. Upfal. Pagerank on an evolving graph. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 24–32. ACM, 2012.

[3] M. Bateni, M. Hajiaghayi, and M. Zadimoghaddam. Submodular secretary problem and extensions. In *APPROX-RANDOM*, pages 39–52, 2010.

[4] G. E. Blelloch, R. Peng, and K. Tangwongsan. Linear-work greedy parallel approximate set cover and variants. In *SPAA*, 2011.

[5] F. Chierichetti, R. Kumar, and A. Tomkins. Max-cover in map-reduce. In *Proceedings of the 19th international conference on World wide web*, 2010.

[6] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *OSDI*, 2004.

[7] R. Gomes and A. Krause. Budgeted nonparametric learning from data streams. In *Proc. International Conference on Machine Learning (ICML)*, 2010.

[8] A. Gupta, A. Roth, G. Schoenebeck, and K. Talwar. Constrained non-monotone submodular maximization: Offline and secretary algorithms. In A. Saberi, editor, *WINE*, volume 6484 of *Lecture Notes in Computer Science*, pages 246–257. Springer, 2010.

[9] G. Hulten, L. Spencer, and P. Domingos. Mining time-changing data streams. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 97–106. ACM, 2001.

[10] A. Krause and D. Golovin. Submodular function maximization. In *Tractability: Practical Approaches to Hard Problems*. Cambridge University Press, 2012.

[11] R. Kumar, B. Moseley, S. Vassilvitskii, and A. Vattani. Fast greedy algorithms in mapreduce and streaming. In *SPAA*, 2013.

[12] S. Lattanzi, B. Moseley, S. Suri, and S. Vassilvitskii. Filtering: a method for solving graph problems in mapreduce. In *SPAA*, 2011.

[13] M. Minoux. Accelerated greedy algorithms for maximizing submodular set functions. *Optimization Techniques, LNCS*, pages 234–243, 1978.

[14] B. Mirzasoleiman, A. Badanidiyuru, A. Karbasi, J. Vondrak, and A. Krause. Lazier than lazy greedy. In *AAAI*, 2014.

[15] B. Mirzasoleiman, A. Karbasi, R. Sarkar, and A. Krause. Distributed submodular maximization: Identifying representative elements in massive data. In *Advances in Neural Information Processing Systems*, pages 2049–2057, 2013.

[16] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions - I. *Mathematical Programming*, 1978.

[17] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. Keogh. Experimental comparison of representation methods and distance measures for time series data. *Data Mining and Knowledge Discovery*, 26(2):275–309, 2013.

Signatures

The research plan of Baharan Mirzasoleiman is endorsed.

Zürich, Switzerland
October 21, 2014

Baharan Mirzasoleiman

Andreas Krause