Project Ideas for 236779
Foundations of Algorithms for Massive Datasets
Fall 2015-16

Nir Ailon
Compiled from an earlier version by Nir Ailon and Elad Hofer
November 18, 2015

1 General

This document might help you find a project for the course. An ideal project should focus on a research topic (either one of the topics below, or your own topic), and should generally include the following parts:

1. A summary of what’s known about the topic, written by you after reading the relevant literature. This summary should follow strict academic writing guidelines, including proper citation and clarity.

2. An experimental part, which can serve as an empirical confirmation (or refutation!?) of a known theoretical result or conjecture. Yet more exciting is an experiment that gives rise to new conjectures that turn into a research problem!

Projects do not have to include both parts necessarily, but they should require a proper amount of effort, and most importantly: They should be interesting. For example, if you write a brilliant survey of a topic that is worth posting somewhere and could become the "standard text" that people read for that topic, then it’s great. If you make interesting progress in an open question but do not experiment, then it’s wonderful. If you write a short topic summary but implement a huge, large scale project with interesting empirical findings, you have our blessing.

You don’t have to decide in advance whether you’re going to spend more time on the theoretical or on the experimental side. It’s best to start reading about a topic you find interesting, to think about it, and see where it takes you. And always remember:

Don’t be afraid to tackle big problems.

You never know.
2 Sublinear Algorithms

Roughly speaking, sublinear algorithms include any algorithm that runs in time $o(n)$ for input of size $n$. Often, you don’t even see the entire input. If you’ve ever run into any of the following terms: ”property testing”, ”property reconstruction (i– ask Nir about this)”, ”streaming”, ”stochastic gradient descent”, ”matrix completion”, and also ”dimensionality reduction”, then you’ve heard of sublinear algorithms in some form.

There is a nice list of open problems in sublinear algorithms (and some of these problems involve streaming and dimensionality reduction): [http://sublinear.info/index.php?title=Open_Problems:By_Number](http://sublinear.info/index.php?title=Open_Problems:By_Number) compiled from various workshops. It’s a great place to get ideas, together with relevant literature.

3 Clustering

Roughly speaking, clustering data means to partition so that all data points in each partition are “similar” or “close” to each other, while all those between two clusters are “dissimilar” or “far”. The exact definition of similarity (or distance) depends on the application, but it is almost always an NP-Hard problem. (Nevertheless, it is usually easy to well approximate it). In learning theory, clustering is considered the most important example of unsupervised learning.

There are many flavours of clustering, and the literature is too vast to mention here. Some interesting results related to the famous Lloyd’s algorithm (for $k$-means, which is a variant of clustering for Euclidean data) can be found in [6] and in [35]. ExampleLiterature for streaming algorithms for clustering data can be found in [16], [1], [4] and many references from and to these papers.

An interesting version of clustering problem known as correlation clustering was defined here [8]. In certain settings can be seen as a special case of matrix recovery [3] (see section below). It can be viewed as a supervised version of clustering, from a learning theoretical perspective. Practical work on correlation clustering can be found, eg here [11].

4 Nearest neighbor search

Nearest neighbor searching is the task of preprocessing a dataset so that, given a new (query) data point, the closest (or almost closest) datapoint can be found (with high probability). Aside from being an interesting problem from an algorithm and data structure design point of view, it is also considered widely in machine learning as a non-parameteric learning algorithm (given a new instance point, assign to it the label of the closest instance you have seen in the training set).

The literature is as vast as that of clustering. For theoretical papers, you can find some upper bounds in [39], [23], [33], and some lower bounds in [34] [36].

There is also much practical work in nearest neighbor searching, eg [32] (just as an example).
5 Sparse Recovery

Sparse recovery is the idea that, if you have a (approximate) sparse prior on your data, then you need only few (random) measurements in order to faithfully represent it. The theory of applications of sparse recovery (or, “compressed sensing”, as some engineers would refer to it) is theoretically deep and also exciting from a practical point of view. It turns out that this subject is intimately related to random projections, also a topic in this course.

If you want to get into the subject, you should read the classic [13] and go from there. To understand the connection with random projections, you can start from [44]. Some more recent stuff on adaptive sparse recovery can be found here: [24] (for vector signals) and also [3] (for matrices).

6 MapReduce

This large scale distributed computational architecture is widely used in places like Google (although word has it that today a more sophisticated architecture is used there). The most commonly used open source version is called Hadoop (developed at Yahoo!). See [21, 27, 30, 29]. The paper A bridging model for multi-core computing [43] by Leslie Valiant is also worth reading.

7 Matrix Sampling and Sketching, Large Scale Numerical Linear Algebra

How do you sketch large matrices, and what operations can your sketch support? See Frieze et al 2004 [20], Drineas et al 2006 [19], Deshpande et al 2006 [17], Sarlos et al 2006 [42], Boutsidis et al 2011 [12], Drineas et al 2011 [18]. An amazingly cute sketching algorithm that is inspired by the “heavy hitters” algorithm that we learned in class on October 29 can be found here: [31].

8 Matrix Completion (The ”Netflix” Problem)

The “Netflix” problem is an instantiation of the problem of collaborative filtering, which is an approach for building a recommender system. It became famous because it involved a lot of money and news coverage.

The idea of matrix completion is, that if you only view a tiny fraction of elements of a gigantic matrix, then you might be able to approximately complete it (that is, to build an efficient oracle that returns the value of the matrix in row $i$ column $j$, given a query $(i, j)$) if you assume certain priors on the matrix (e.g. low rank, sparsity etc).

This subject has much work in both the practical and theoretical extremes.
For practical stuff, perhaps you should refer to the work of Yehuda Koren’s team who won the Netflix challenge. Here they explain how: \[10, 9, 28\]. Aside from that, there is vast literature in practical machine learning conferences and journals.

Much dep theory has been inspired by the Netflix contest. See \[40, 22, 41, 15, 14, 22\]. (The first two in the list are easier than the others). Some good links for more literature: http://web.engr.illinois.edu/~swoh/software/optspace/papers.html
http://terrytao.wordpress.com/tag/matrix-completion/

9 Random Projections

Random projections refers to a method of reducing dimensionality of data that is oblivious to the data. For Euclidean metrics, a name that is almost synonymous with “random projections” is “Johnson-Lindenstrauss”, coined after this classic \[25\]. More computationally efficient versions can be found here: \[2, 5, 26\]. The latter resource is streaming friendly.

Aside from an algorithm (or rather, a method) that is interesting in its own right, random projections have been studied as preprocessing steps for other more sophisticated algorithms that have a heavy dependence on input dimensionality (SVM, linear regression, SVD decomposition). A Matlab library called BLENDENPIK, for example, uses such ideas to achieve so-called well conditionedness for linear regression. Read about it here (see \[7\]). It is claimed that this library beats the classic Matlab library LAPACK for various scenarios. (Dare to check it out?) I’ve been told that \[38\] is a paper that uses random projection-like tricks for learning with kernels, for those who love learning theory. Also for learning aficionados, in \[37\], you can find both theoretical and experimental justification for random projections as preprocessing for SVM’s (Support Vector Machines).

10 Other Resources

10.1 Venues

You can also search proceedings of conferences that publish work on algorithms for ”big data”, in various forms. If you are interested in theoretical aspects, you should check out (those with a ‘*’ are considered more prestigious). You might have to skim through much unrelated work (by reading title/abstract), because the “big data” is not the main theme of these conference.

- COCOA. Combinatorial Optimization and Applications.
- COCOON. Computing and Combinatorics.
- ESA. European Symposium on Algorithms.
- * FOCS. Foundations of Computer Science.
• * ICALP (track A). International Colloquium on Automata, Languages and Programming.
• ICDE. International Conference on Data Engineering.
•ISAAC. International Symposium on Algorithms and Computation.
• LATIN. Latin American Symposium on Theoretical Informatics.
•MFCS. Mathematical Foundations of Computer Science.
•* RANDOM. International Workshop on Randomization and Computation.
•* SODA. Symposium on Discrete Algorithms.
•STACS. Symposium on Theoretical Aspects of Computer Science.
•SWAT. Scandinavian Workshop on Algorithm Theory.
•TAMC. Theory and Applications of Models of Computation.
•WADS. Workshop on Algorithms and Data Structures.

Conferences that are considered more experimental (although some theory can be found there as well):

• KDD. Knowledge Discovery and Data Mining. (data mining)
• ICML. International Conference on Machine Learning. (machine learning)
• NIPS. Neural Information Processing Systems. (machine learning)
• SIGMOD. ACM SIGMOD International Conference on Management of Data. (databases)
• VLDB. Very Large Data Bases. (databases)

## 10.2 People

• **Dimensionality reduction.** Nir Ailon, Felix Kramer, Edo Liberty, Jan Vybíral, Rachel Ward, Jelani Nelson
• **Numerical Linear Algebra.** Christos Boutsidis, Ken Clarkson, James Demmel, Amit Deshpande, Petros Drineas, Ravi Kannan, Malik Magdon-Ismail, Michael Mahoney, Richard Peng, Vladimir Rokhlin, Roman Vershynin, David Woodruff.
• **Compressed sensing.** Richard Baraniuk, Emmanuel Candès, David Donoho, Anna Gilbert, Piotr Indyk, Deanna Needell, Eric Price, Justin Romberg, Martin Strauss, Terence Tao, Joel Tropp, David Woodruff.
• **External memory algorithms.** Peyman Afshani, Pankaj Agarwal, Lars Arge, Michael Bender, Gerth Stølting Brodal, Martin Farach-Colton, Jeremy Fineman, John Iacono, Bradley Kuszmaul, Charles Leiserson, S. Srinivasa Rao, Norbert Zeh, Qin Zhang.
• **MapReduce/Hadoop.** Michael Goodrich, Siddharth Suri, Sergei Vassilvitski. The paper “A bridging model for multi-core computing” by Leslie Valiant is also worth reading.
References


