From TCS to Learning Theory

Kasper Green Larsen
Aarhus University, Denmark

Abstract

While machine learning theory and theoretical computer science are both based on a solid mathematical foundation, the two research communities have a smaller overlap than what the proximity of the fields warrant. In this invited abstract, I will argue that traditional theoretical computer scientists have much to offer the learning theory community and vice versa. I will make this argument by telling a personal story of how I broadened my research focus to encompass learning theory, and how my TCS background has been extremely useful in doing so. It is my hope that this personal account may inspire more TCS researchers to tackle the many elegant and important theoretical questions that learning theory has to offer.

2012 ACM Subject Classification Theory of computation

Keywords and phrases Theoretical Computer Science, Learning Theory

Digital Object Identifier 10.4230/LIPIcs.MFCS.2024.4

Category Invited Paper

Funding *Kasper Green Larsen*: Supported by a DFF Sapere Aude Research Leader Grant No. 9064-00068B.

1 A Personal Story

My goal with this invited abstract, is to inspire more theoretical computer science (TCS) researchers to engage in topics in machine learning theory. Being a TCS researcher myself, I hope that by telling my own personal story – of how I adopted the field of learning theory – may provide a unique and more compelling argument than if a core learning theorist was to attempt to convince you. Being a personal story, I will focus on some of the research results I obtained along the way, and not so much on the broader learning theory and TCS literature. The results I present have been chosen because they tell an interesting story and showcase how core TCS concepts have proven useful in learning theory. Along the way, I will tell some anecdotes on how these research projects started, and I will try to give you an idea of some of the thought processes I initially went through when trying to enter a new field. I have also tried to sprinkle in some advice for more junior researchers based on my own experiences. Finally, I will also formally define some of the learning theory questions I have studied, giving you an idea of what such questions may look like. I sincerely hope that you find this somewhat unusual abstract refreshing.

1.1 My TCS Background and Connections Between Fields

For those who do not know me, let me start by briefly giving you my background, while also trying to convince you that many fields of TCS are deeply connected, and that it pays off to have a broad interest in TCS.

I started my Ph.D. studies in 2008 under the supervision of Professor Lars Arge at Aarhus University, Denmark. My initial research was on data structures, in particular so-called *I/O-efficient* data structures [2, 5], that take the memory hierarchy of modern hardware into account. Ever since I started my studies, I have always been curious and eager to learn new areas by interacting with great colleagues in the TCS community. This openness to collaborations and sharing of ideas is one of the strongest and most rewarding traits of our community.

Initially, my interactions with other researchers were through fellow students at the department. I explicitly remember Allan Grønlund coming to my office with one of Mihai Pătraşcu's beautiful papers [33] on data structure lower bounds in the cell probe model [35], excitedly proclaiming that "*this is so cool, we need to do this too!*". At the time, I had never had a complexity course or seen a real lower bound proof, and I was immediately hooked. This led me on a long research journey trying to understand and develop the techniques for proving lower bounds on the efficiency (space, query time, update time) of data structures, see e.g. [23, 24, 29] for some highlights.

While cell probe lower bounds were my main focus for years, I always found research that bridges several fields the most exciting. The earliest such example in my own research started with a visit by Jeff M. Phillips in our research group. During his visit, Jeff introduced me to *discrepancy theory* while explaining some of his recent work. As discrepancy theory will be important later, let me give a rough definition of it here. At its core, discrepancy theory studies the following problem: Given a matrix $A \in \mathbb{R}^{m \times n}$, compute a *coloring* $x \in \{-1, 1\}^n$ minimizing the discrepancy $\|Ax\|_\infty$. Understanding the best achievable discrepancy for various types of matrices is an old and very well-studied topic in TCS [3, 34, 10, 7, 31]. In particular, I recall Jeff telling me about a result of Banaszczyk [6] showing that matrices with sparse columns always have low discrepancy colorings. While being intellectually intrigued by this result, I did not see any immediate applications of it in my own research. However, a couple of months later, Elad Verbin, a post doc in our group at the time, suggested to me that I should try to prove data structure lower bounds in the *group model*. Without going into details here, it turns out that Banaszczyk's discrepancy result could be used to directly translate decades of research on discrepancy lower bounds into group model data structure lower bounds [25]. The connection even improved some of the discrepancy upper bounds using data structure upper bounds as well.

This theme of connecting areas has proven very useful over the years. It for instance got me started on *streaming algorithms* when Jelani Nelson and Huy L. Nguyen asked whether we could use cell probe lower bound techniques to prove *time* lower bounds for streaming algorithms [27]. In later work with Jelani [26], we similarly proved optimality of the Johnson-Lindenstrauss (JL) lemma [21] in *dimensionality reduction* using an encoding-based argument. Such arguments were typically used in cell probe data structure lower bounds. As it will be important later, let us recall that the JL lemma says that any set of n points in $\mathbb{R}^d$ can be embedded into $O(\varepsilon^{-2} \log n)$ dimensions while preserving all pairwise distances to within a factor $1 \pm \varepsilon$.

Another interesting example was initiated during a visit at MIT in 2017 where Vinod Vaikuntanathan was telling me how "*data structures and crypto are a match made in heaven*". This claim got me curious, and after returning home, I read a bit on data structure in cryptography and stumbled on Oblivious RAMs (ORAMs) [17]. An ORAM is basically a primitive for obfuscating the memory access pattern of an algorithm, such that the memory accesses reveal nothing about the data stored. Checking the references of the papers I was reading, it turned out that a crypto Professor sitting just down stairs, Jesper Buus Nielsen, had made multiple contributions to ORAMs. After some brief initial discussions, we quickly realized that cell probe lower bound techniques could be tweaked to prove strong lower bounds for ORAMs [28]. This connection started a whole line of research into lower bounds for ORAMs and related primitives in crypto.

Finally, let me mention one last example outside learning theory. Back in 2017, I was attending a communication complexity workshop in Banff, where Mark Braverman was giving a talk about some work of his [11] on a conjecture in information theory/network coding [30]. The conjecture relates to communication networks, where a network is modeled by a graph with capacities on the edges, designating how many bits may be sent across. The conjecture then says that for undirected graphs (messages may be sent in both directions), if we are given k source-sink pairs that each need to send an r bit message from source to sink, then the largest number of bits r we can handle without violating capacity constraints, is equal to the multi-commodity flow rate. Intuitively, this means that the best you can do is to simply forward your messages as indivisible bits. Mark's talk was excellent, but at the time, I had no immediate applications of his result. However, while spending a semester at the Simons Institute in 2018, a coincidental conversation with MohammadTaghi Hajiaghayi led us [15] to proving conditional lower bounds for I/O-efficient algorithms using the information theory conjecture that Mark presented. Needless to say, it was quite satisfactory to come full circle and address the topic that started my Ph.D. studies. In addition to the lower bounds for I/O-efficient algorithms, the conjecture also gave a clean conditional $\Omega(n \log n)$ lower bound for constant degree boolean circuits for multiplication [1]. I find it quite fascinating how two such seemingly different problems of multiplication and communication in graphs may prove to be so intimately connected.

If there is one message to take away from my own experiences, in particular for junior researchers, it must be that many things are deeply connected and you never know when results in one branch of TCS may prove useful in another. That is one of its beauties. I would thus strongly recommend that you attend talks, seminars, and generally engage in discussions with the many brilliant TCS researchers, also in fields outside your own immediate interest. Build your expertise in one area, but always be curious and look for connections and inspiration from others.

1.2 Entering Learning Theory

Now let me get to the promised topic of entering learning theory from a TCS background, and how this background proved extremely useful. For me, this journey started around 2019. As you all know, machine learning was the big hype at the time, and probably still is. Like many others, I initially found deep neural networks and what they could do quite fascinating. This led me to consider whether I should get into machine learning and work on this extremely hot topic. However, after getting a slightly better understanding of the field, I quickly found that training a deep neural net and running some experiments on a benchmark data set, to be much more engineering and craftsmanship than deep stimulating thoughts. And with all the obligations that come with a faculty position, I just did not have the time to learn the practical skills that it takes to make efficient and competitive implementations. And probably was not that interested in it after all. I almost abandoned machine learning, had it not been for a student of mine, Alexander Mathiasen. Alexander was very independent and eager to get into machine learning. He had realized that I was more interested in theory questions and had on his own discovered the great source of open problems in learning theory that are published with the COLT proceedings. The open problem Alexander had found was related to a technique called *boosting*. Let me start by introducing the general idea in boosting and the setup for binary classification in supervised learning.

Supervised Learning and Boosting

In supervised learning, a binary classification problem is specified by an input domain $\mathcal{X}$, an unknown target function $f : \mathcal{X} \to \{-1, 1\}$ and an unknown data distribution $\mathcal{D}$ over $\mathcal{X}$. A learning algorithm receives as input a training data set of n i.i.d. samples $(x_i, f(x_i))$ with each $x_i \sim \mathcal{D}$. The goal of the learning algorithm is to output a classifier $h : \mathcal{X} \to \{-1, 1\}$ minimizing the probability of misprediction the label of a new sample from $\mathcal{D}$, i.e. where $\mathrm{er}_{\mathcal{D}}(h) := \Pr_{x \sim \mathcal{D}}[h(x) \neq f(x)]$ is as small as possible. To have an example in mind, think of $\mathcal{X}$ as the set of all images of a particular size and f as the hard-to-specify function that maps every image of a mammal to -1 and every other image to 1. The learning problem is thus to train a classifier that can detect whether an image contains a mammal or not.

Boosting is then a powerful and elegant idea that allows one to combine multiple inaccurate classifiers into a highly accurate *voting classifier*. Boosting algorithms such as AdaBoost [16] work by iteratively running a base learning algorithm on reweighed versions of the training data to produce a sequence of classifiers $h_1, \dots, h_t$. After obtaining h_i, the weighting of the training data is updated to put larger weights on samples misclassified by h_i, and smaller weights on samples classified correctly. This effectively forces the next training iteration to focus on points with which the previous classifiers struggle. After sufficiently many rounds, the classifiers $h_1, \dots, h_t$ are finally combined by taking a (weighted) majority vote among their predictions. This idea also goes under the name of *multiplicative weight updates* and has many applications in TCS.

Boosting works exceptionally well in practice and much theoretical work has gone into explaining its huge success. One important line of work in this direction is based on the notion of *margins* [8]. A voting classifier g may be written as $g(x) = \mathrm{sign}(\sum_{i=1}^t \alpha_i h_(x))$ with $\alpha_i \in \mathbb{R}$. If we normalize the weights so that $\sum_i |\alpha_i| = 1$, then $\sum_{i=1}^t \alpha_i h_i(x) \in [-1, 1]$. The margin of g on a sample $(x, y) \in \mathcal{X} \times \{-1, 1\}$ is then defined as $y \sum_{i=1}^t \alpha_i h_i(x)$. The margin is thus a number in $[-1, 1]$ and can be thought of as a confidence in the prediction. If the margin is 1, then all classifiers combined by g agree and are correct. If the margin is 0, then we have a (weighted) 50-50 split between the two possible predictions $-1/1$, and so on. It has then been proved that voting classifiers g with large margins on all training samples (x_i, y_i) have a small $\mathrm{er}_{\mathcal{D}}(g)$ [8].

Boosting and Discrepancy Theory

Now let me return to the open problem Alexander approached me with. Since voting classifiers with large margins have a small $\mathrm{er}_{\mathcal{D}}(g)$, it seems natural to develop boosting algorithms that obtain large margins by combining few base classifiers, i.e. with a small t. This leads to faster predictions and possibly also faster training. The question now is as follows: If the best possible margin on all training samples is $\gamma^\star$ when one is allowed to combine arbitrarily many base classifiers h_i, then how large margins can we obtain by combining t base classifiers? The best known algorithm obtained margins of $\gamma^\star - O(\sqrt{\log(n)/t})$ with n training samples, and the best known lower bound showed that this is tight whenever $t \leq n^{1/2}$. The conjecture stated that this remains tight for $t \leq n \log n$.

In my first learning theory paper [32], we observed an interesting connection between this question and discrepancy theory. In more detail, we considered the matrix A obtained by forming a row for each training sample (x_i, y_i) and a column for each h_i in a voting classifier $g(x) = \mathrm{sign}(\sum_{j=1}^t \alpha_j h_j(x))$. The entry corresponding to (x_i, y_i) and h_j stores the value $y_i \alpha_j h_j(x)$. Notice that the sum of the entries in the i'th row is precisely the margin of g on (x_i, y_i). Next, if we can find a coloring $x \in \{-1, 1\}^t$ such that $\|Ax\|_\infty \approx 0$, then this means that for every single row of A, the sum over the columns j where $x_j = 1$ is almost exactly the same as the sum over columns where $x_j = -1$. Thus if we replace g by a new voting classifier where we set all α_j to 0 when x_j equals the majority of $-1/1$'s in x, and to 2 when x_j is the minority, then the new g' has almost exactly the same margins as before but uses only half as many base classifiers. Recursively finding a low discrepancy coloring x using the celebrated *Spencer's six standard deviations suffice* result [34], we were able to refute the conjecture on the tradeoff between t and the achievable margin while establishing yet an interesting connection between discrepancy theory and other areas.

Support Vector Machines, Sketching and Teaching

Around the same time as our first learning theory results, I had voluntarily started teaching the Machine Learning undergraduate course at Aarhus University. Never having followed such a course myself, this was a great way of forcing myself to learn the basics. But beyond learning the basics, teaching also led to new research questions. After having completed the first project on boosting, we naturally started reading up on the references proving that large margins lead to small $\text{er}_{\mathcal{D}}(g)$. Such results are referred to as *generalization bounds* and are a cornerstone of learning theory. Studying the classic proofs of generalization for large margin voting classifiers [8], there seemed to be an underlying idea of *sketching* or *compressing* the voting classifier g while exploiting that it has large margins. Intuitively, if there is a small-bit representation of g, then there are only few choices for g. A union bound over all these choices shows that it is unlikely that there is even a single g that performs great on the training data (has large margins) and poor on new data (has large $\text{er}_{\mathcal{D}}(g)$).

Around the same time, I had just taught Support Vector Machines (SVMs) [14]. SVMs are another type of learning algorithm where the input domain $\mathcal{X}$ is $\mathbb{R}^d$. In the simplest setup, when given a training set of n samples $(x_i, y_i) \in \mathbb{R}^d \times \{-1, 1\}$ with $\|x_i\| \leq 1$ for all i, SVM searches for the separating hyperplane with the largest *margin* to the data. In the context of SVMs, if we assume for simplicity that a hyperplane passes through the origin and has unit length normal vector $w \in \mathbb{R}^d$, then we predict the label of a new point $x \in \mathbb{R}^d$ by returning $\text{sign}(w^T x)$. The margin of the hyperplane on a labeled point (x, y) is then defined as $yx^T w$. The margin is positive if the hyperplane correctly predicts the label of (x, y) and the absolute value of the margin measures how far the point is from the hyperplane itself.

Similarly to the case for boosting, there were known generalization bounds [9] stating that if a hyperplane h has large margins, then $\text{er}_{\mathcal{D}}(h)$ is small. However, these bounds were proved in a completely different way than the boosting bounds and seemed to be sub-optimal. In our work [18], we improved these generalization bounds by observing a connection to the Johnson-Lindenstrauss (JL) lemma [21] that I previously worked on [26]. Shortly after the lower bound for JL by Jelani and I, Noga Alon and Bo'az Klartag [4] presented an alternative proof that at its core gives a randomized sketch for representing a set of n unit length vectors in $\mathbb{R}^d$ using $O(\varepsilon^{-2} \log n)$ bits for each vector, while preserving their pairwise inner products up to additive ε. Our idea was then to apply this sketch to w and the training data. Since $w^T x$ changes by only additive ε, the sketched hyperplane $\tilde{w}$ remains correct (i.e. $\text{sign}(\tilde{w}^T x) = \text{sign}(w^T x) = y$) if the margin was larger than ε before sketching. In this way, we get a compression whose size depends on the margin. Combining this with the union bound idea mentioned earlier led to improved generalization bounds for SVMs.

Replicable Learning and Sketching

Let me give another example where the JL lemma proved useful in learning theory. In a very exciting line of work, starting with [20], core TCS researchers introduced the notion of *replicable learning* as an attempt at addressing issues with reproducibility of results in machine learning. Here the basic setup is that you want to develop learning algorithms $\mathcal{A}$, such that if $\mathcal{A}$ is run on two sets S and S' of n i.i.d. samples from the same distribution $\mathcal{D}$, then we should get the same output with high probability. The intuition is thus, if you rerun someones replicable algorithm on your own data set, you will get the same results as they did on their data set, provided that the data set is sampled from the same distribution. To help you in this seemingly difficult task, the two executions of $\mathcal{A}$ share a random seed. This may be justified in practice by publishing the seed of a pseudorandom generator along with your machine learning paper and experimental results. There is naturally a ton of problems to revisit in replicable learning and some strong connections with *differential privacy* have already been established [12].

In a recent result [22], we considered SVMs in the replicable setting. Here we are promised that there exists a hyperplane with all margins at least γ on the support of the data distribution $\mathcal{D}$, and must replicably compute a hyperplane h with small $\mathrm{er}_{\mathcal{D}}(h)$. The key idea in our work is quite simple: partition the training data into t chunks and run SVM on each to obtain hyperplanes with normal vectors $w_1, \dots, w_t$. Average these normal vectors $w = t^{-1} \sum_i w_i$ and apply the sketching technique by Alon and Klartag [4] to w. The first main observation is that computing the average w reduces variance compared to each w_i. Thus for two independent but identically distributed training data sets S and S', the resulting w and w' will be close. The key property of the sketching technique of Alon and Klartag is that such close vectors are very likely to be "rounded" to the same hyperplane/sketch if we share a carefully chosen random seed.

Multi-Distribution Learning, Discrepancy Theory and NP-Hardness

Let me conclude with one last example of TCS techniques playing a key role in learning theory results. A recent line of work in learning theory has studied learning in a setup with multiple data distributions. Formally, we have k distributions $\mathcal{D}_1, \dots, \mathcal{D}_k$ over $\mathcal{X} \times \{-1, 1\}$ and the goal is to train a classifier that performs well on all k distributions simultaneously. For this purpose, we are given a hypothesis set $\mathcal{H}$ containing hypotheses $h : \mathcal{X} \to \{-1, 1\}$. As an example, think of $\mathcal{H}$ as all hyperplanes in $\mathbb{R}^d$. If we let $\tau = \min_{h \in \mathcal{H}} \max_i \mathrm{er}_{\mathcal{D}_i}(h)$ denote the best achievable max-error of any $h \in \mathcal{H}$, then the goal is produce a classifier $f : \mathcal{X} \to \{-1, 1\}$ with $\max_i \mathrm{er}_{\mathcal{D}_i}(f) \leq \tau + \varepsilon$ using as few training samples as possible.

It is known that for a single distribution $\mathcal{D}$, this task requires $\Theta(d/\varepsilon^2)$ samples and it is straight forward to generalize the algorithm to $O(dk/\varepsilon^2)$ samples for k distributions. Very surprisingly, it turns out that this can be improved to essentially $O((d+k)/\varepsilon^2)$ samples using boosting ideas [19]. However, the upper bounds achieving this number of samples are cheating a little. Concretely, they do not output a single hypothesis f with $\max_i \mathrm{er}_{\mathcal{D}_i}(f) \leq \tau + \varepsilon$. Instead, they output a distribution $\mathcal{D}_f$ over hypotheses such that $\max_i \mathbb{E}_{f \sim \mathcal{D}_f}[\mathrm{er}_{\mathcal{D}_i}(f)] \leq \tau + \varepsilon$. Attempting to *derandomize* this does not seem to work as we only have Markov's inequality to argue that with constant probability, a random $f \sim \mathcal{D}_f$ has $\max_i \mathrm{er}_{\mathcal{D}_i}(f) = O(k(\tau + \varepsilon))$. In a current manuscript, we show that there is a good reason why this is the case. Concretely, we prove via a reduction from discrepancy minimization that it is NP-hard to compute an $f : \mathcal{X} \to \{-1, 1\}$ with $\max_i \mathrm{er}_{\mathcal{D}_i}(f) \leq \tau + \varepsilon$. Here we use a result of Charikar et al. [13] stating that it is NP-hard to distinguish whether, for a given a 0/1 matrix A, there exists an $x \in \{-1,1\}^n$ with $Ax = 0$, or whether all $x \in \{-1,1\}^n$ have $\|Ax\|_\infty = \Omega(\sqrt{n})$. In our reduction, we think of the columns as the input domain $\mathcal{X}$ and we form a distribution for each row of A. The key idea is to show that if a learning algorithm computes an $f : \mathcal{X} \to \{-1,1\}$ with $\max_i \mathrm{er}_{\mathcal{D}_i}(f) \leq \tau + \varepsilon$, then the predictions made by f on the columns must give a low discrepancy coloring that can distinguish the two cases.

While showing NP-hardness is by now completely standard in TCS, let me add that my two coauthors are both from Statistics. They had heard NP-hardness mentioned before, but never used it, and found quite some joy in getting to apply it on a relevant problem. I guess the experience shows that what may be common tools in one discipline may be highly novel in another.

1.3 Conclusion and Thoughts

To summarize my research journey into learning theory, I hope that I convinced you that many techniques and tricks from TCS are extremely useful. Not only do they help in addressing learning theory questions, but they also bring an interesting new direction to the field, such as e.g. arguing computational hardness of training a classifier. Furthermore, this transfer of techniques is not isolated to learning theory, but has been a guiding principle in all my research since the early days of my Ph.D. studies. Keep your eyes open for surprising connections between seemingly disjoint topics - they are often hiding just below the surface.

To me, the questions studied in learning theory are as clean, elegant, beautiful and well-defined as any TCS topic, and are very well fit for anyone with a TCS background. What has worked well for me, has been to start in some corner (coincidentally boosting for me) and get to know the literature and related questions. Then as you have a better understanding and overview, you can start approaching neighboring questions and growing your focus. I suppose this is not much different from starting your Ph.D. studies by working on a narrow topic and broadening out as you mature. Finally, do not underestimate the value of talking to peers and attending talks for inspiration. As a newcomer, it is sometimes hard to ask the right question, and calibrating with an expert is very useful. Also, the COLT open problems have been a good source of inspiration as you at least know that some experts in the field find the question interesting.

I strongly hope that this abstract may inspire some of you to work on learning theory, or give you the courage to enter a new field that you find intriguing, knowing that your TCS skills are useful in many surprising places.

References

1 Peyman Afshani, Casper Benjamin Freksen, Lior Kamma, and Kasper Green Larsen. Lower bounds for multiplication via network coding. In *ICALP*, volume 132 of *LIPIcs*, pages 10:1–10:12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019.

2 Alok Aggarwal and S. Vitter, Jeffrey. The input/output complexity of sorting and related problems. *Commun. ACM*, 31(9):1116–1127, September 1988. doi:10.1145/48529.48535.

3 R. Alexander. Geometric methods in the study of irregularities of distribution. *Combinatorica*, 10(2):115–136, 1990.

4 Noga Alon and Bo'az Klartag. Optimal compression of approximate inner products and dimension reduction. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 639–650, 2017. doi:10.1109/FOCS.2017.65.

5 Lars Arge. The buffer tree: A new technique for optimal i/o-algorithms. In Selim G. Akl, Frank Dehne, Jörg-Rüdiger Sack, and Nicola Santoro, editors, *Algorithms and Data Structures*, pages 334–345, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg.

6 Wojciech Banaszczyk. Balancing vectors and gaussian measures of n-dimensional convex bodies. *Random Structures & Algorithms*, 12:351–360, July 1998.

7 Nikhil Bansal. Constructive algorithms for discrepancy minimization. In *Proc. 51th Annual IEEE Symposium on Foundations of Computer Science (FOCS'10)*, pages 3–10, 2010.

8 Peter Bartlett, Yoav Freund, Wee Sun Lee, and Robert E. Schapire. Boosting the margin: a new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686, 1998.

9 Peter Bartlett and John Shawe-Taylor. *Generalization performance of support vector machines and other pattern classifiers*, pages 43–54. MIT Press, Cambridge, MA, USA, 1999.

10 J. Beck and T. Fiala. Integer-making theorems. *Discrete Applied Mathematics*, 3:1–8, February 1981.

11 Mark Braverman, Sumegha Garg, and Ariel Schvartzman. Coding in undirected graphs is either very helpful or not helpful at all. In *8th Innovations in Theoretical Computer Science Conference, ITCS 2017, January 9-11, 2017, Berkeley, CA, USA*, pages 18:1–18:18, 2017.

12 Mark Bun, Marco Gaboardi, Max Hopkins, Russell Impagliazzo, Rex Lei, Toniann Pitassi, Satchit Sivakumar, and Jessica Sorrell. Stability is stable: Connections between replicability, privacy, and adaptive generalization. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 520–527, 2023.

13 Moses Charikar, Alantha Newman, and Aleksandar Nikolov. Tight hardness results for minimizing discrepancy. In Dana Randall, editor, *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 1607–1614. SIAM, 2011. doi:10.1137/1.9781611973082.124.

14 Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20:273–297, 1995.

15 Alireza Farhadi, MohammadTaghi Hajiaghayi, Kasper Green Larsen, and Elaine Shi. Lower bounds for external memory integer sorting via network coding. In *STOC*, pages 997–1008. ACM, 2019.

16 Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

17 Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious RAMs. *Journal of the ACM (JACM)*, 1996.

18 Allan Grønlund, Lior Kamma, and Kasper Green Larsen. Near-tight margin-based generalization bounds for support vector machines. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 3779–3788. PMLR, 2020.

19 Nika Haghtalab, Michael I. Jordan, and Eric Zhao. On-demand sampling: Learning optimally from multiple distributions. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL: http://papers.nips.cc/paper_files/paper/2022/hash/02917acec264a52a729b99d9bc857909-Abstract-Conference.html.

20 Russell Impagliazzo, Rex Lei, Toniann Pitassi, and Jessica Sorrell. Reproducibility in learning. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 818–831. ACM, 2022. doi:10.1145/3519935.3519973.

21 William Johnson and Joram Lindenstrauss. Extensions of lipschitz maps into a hilbert space. *Contemporary Mathematics*, 26:189–206, January 1984. doi:10.1090/conm/026/737400.

22 Alkis Kalavasis, Amin Karbasi, Kasper Green Larsen, Grigoris Velegkas, and Felix Zhou. Replicable learning of large-margin halfspaces. In *ICML*, Proceedings of Machine Learning Research. PMLR, 2024. To appear.

23 Kasper Green Larsen. The cell probe complexity of dynamic range counting. In *STOC*, pages 85–94. ACM, 2012.

24 Kasper Green Larsen. Higher cell probe lower bounds for evaluating polynomials. In *FOCS*, pages 293–301. IEEE Computer Society, 2012.

25 Kasper Green Larsen. On range searching in the group model and combinatorial discrepancy. *SIAM J. Comput.*, 43(2):673–686, 2014.

26 Kasper Green Larsen and Jelani Nelson. Optimality of the johnson-lindenstrauss lemma. In *FOCS*, pages 633–638. IEEE Computer Society, 2017.

27 Kasper Green Larsen, Jelani Nelson, and Huy L. Nguyên. Time lower bounds for nonadaptive turnstile streaming algorithms. In *STOC*, pages 803–812. ACM, 2015.

28 Kasper Green Larsen and Jesper Buus Nielsen. Yes, there is an oblivious RAM lower bound! In *CRYPTO (2)*, volume 10992 of *Lecture Notes in Computer Science*, pages 523–542. Springer, 2018.

29 Kasper Green Larsen, Omri Weinstein, and Huacheng Yu. Crossing the logarithmic barrier for dynamic boolean data structure lower bounds. In *STOC*, pages 978–989. ACM, 2018.

30 Zongpeng Li and Baochun Li. Network coding: the case of multiple unicast sessions. In *Proceedings of the 42nd Allerton Annual Conference on Communication, Control and Computing*, Allerton'04, 2004.

31 Shachar Lovett and Raghu Meka. Constructive discrepancy minimization by walking on the edges. *SIAM Journal on Computing*, 44(5):1573–1582, 2015.

32 Alexander Mathiasen, Kasper Green Larsen, and Allan Grønlund. Optimal minimal margin maximization with boosting. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 4392–4401. PMLR, 2019.

33 Mihai Pătraşcu. Unifying the landscape of cell-probe lower bounds. *SIAM Journal on Computing*, 40(3):827–847, 2011. doi:10.1137/09075336X.

34 Joel Spencer. Six standard deviations suffice. *Trans. Amer. Math. Soc.*, 289:679–706, 1985.

35 Andrew Chi-Chih Yao. Should tables be sorted? *J. ACM*, 28(3):615–628, July 1981. doi:10.1145/322261.322274.