Stratified Sampling meets Machine Learning

Kevin Lang
Yahoo Labs
langk@yahoo-inc.com

Edo Liberty
Yahoo Labs
edo@yahoo-inc.com

Konstantin Shmakov
Yahoo Labs
kshmakov@yahoo-inc.com

ABSTRACT

This paper investigates the practice of non-uniformly sampling records from a database. The goal is to evaluate future aggregate queries as accurately as possible while maintaining a fixed sampling budget.

We formalize this as a machine learning problem in the PAC model. The model learned corresponds to sampling probabilities of individual records and a training set is obtained from previously issued queries to the database. We provide an efficient and simple regularized Empirical Risk Minimization (ERM) algorithm for this problem along with a theoretical generalization result for it.

Our experiments show that model accuracy improves with more training data, that insufficient training data can cause overfitting and that careful regularization is key. These give important practical insights and strengthen the parallels to other machine learning tasks. We report extensive results for both synthetic and real datasets that significantly improve over both uniform sampling and standard stratified sampling.

1. INTRODUCTION

Given a database of n records $1, 2, \ldots, n$ we define the result y of an aggregate query q to be $y = \sum_i q_i$. Here, q_i is the scalar result of evaluating query q on record i.[1] For example, consider a database containing user actions on a popular site such as the Yahoo homepage. Here, each record corresponds to a single user and contains his/her past actions on the site. The value q_i can be the number of times user i read a full news story if they are New York based and $q_i = 0$ otherwise. The result $y = \sum_i q_i$ is the number of articles read by Yahoo's New York based users. In an internal system at Yahoo (YAM+) such queries are performed in rapid succession by advertisers when designing advertising campaigns.

[1]For notational brevity, the index i ranges over $1, 2, \ldots, n$ unless otherwise specified.

Answering such queries efficiently poses a challenge. On the one hand, the number of users n is too large for an efficient linear scan, i.e., evaluating y explicitly. This is both in terms of running time and space (disk) usage. On the other hand, the values q_i could be results of applying arbitrarily complex functions to records. This means no indexing, intermediate pre-aggregation, or sketching based solutions could be applied.

Fortunately, executing queries on a random sample of records can provide an approximate answer which often suffices. The importance of sampling to database algorithms and the history of research in this area are both very significant. See the work of Olken, Rotem and Hellerstein [1, 2, 3, 4] for motivations and efficient algorithms for database sampling. It is well known (and easy to show) that a uniform sample of records provides a provably good solution to this problem.

1.1 Uniform Sampling

Definition: The Horvitz-Thompson estimator for y is given by $\tilde{y} = \sum_{i \in S} q_i / p_i$ where $S \subset \{1, 2, \ldots, n\}$ is the set of sampled records and $\Pr(i \in S) = p_i$.

Definition: The numeric cardinality of a query is defined by $\mathrm{card}(q) := \sum |q_i| / \max |q_i|$. For binary queries ($q_i \in \{0, 1\}$) this coincides with the standard notion of cardinality.

If $p_i \geq \zeta > 0$ the following statements hold true.

- The estimator $\tilde{y}$ is an unbiased estimator of y.
$$\mathbb{E}[\tilde{y} - y] = 0 \ .$$
- The standard deviation, σ, of $\tilde{y} - y$ is bounded.
$$\sigma[\tilde{y} - y] \leq y \sqrt{1/(\zeta \cdot \mathrm{card}(q))} \ .$$
- The probability of large deviation is small.
$$\Pr[|\tilde{y} - y| \geq \varepsilon y] \leq e^{-O(\varepsilon^2 \zeta \cdot \mathrm{card}(q))} \ .$$

The first and second facts follow from direct expectation and variance computations. The third follows from $\tilde{y} - y$ being a sum of independent, mean zero, random variables and the application of Bernstein's inequality to it.

Very selective queries or queries with extreme values could exhibit $\mathrm{card}(q) = O(1)$. To obtain a reasonable variance

and success probability for such queries ζ must be large (a constant) which prohibits any effective sampling. On the other hand, well spread queries that select a large fraction of the records and have moderate values have $\text{card}(q) = O(n)$. That allows for ζ to be inversely proportional to n which means that a constant size sample suffices.

Acharya, Gibbons and Poosala [5] showed that uniform sampling is the optimal strategy against adversarial queries. Later it was shown by Liberty, Mitzenmacher, Thaler and Ullman [6] that uniform sampling is also space optimal in the information theoretic sense for any compression mechanism (not limited to selecting records). That means that no summary of a database can be more accurate than uniform sampling in the worst case.

Nevertheless, in practice, queries and databases are not adversarial. This gives some hope to the idea that a non-uniform sample should produce better results for practically encountered datasets and query distributions. Because high cardinality queries already receive an adequate solution by uniform sampling, the focus is on low cardinality or highly selective queries. This exact scenario motivated several investigations into this problem.

1.2 Prior Art

Sampling populations non-uniformly is a standard technique in statistics. Stratified Sampling is a practice of selecting individual records with probability proportional to the variance of estimating statistics on their strata. An example is known as Neyman allocation [7, 8] which selects records with probability inversely proportional to the size of the stratum they belong to.[2] Strata in this context is a mutually exclusive partitioning of the records which mirrors the structure of future queries. This structure is overly restrictive for our setting where the queries are completely unrestricted.

Acharya et al. [5] introduce *congressional sampling*. This is a hybrid of uniform sampling and *Neyman allocation*. The stratification is performed with respect to the relations in the database. Later Chaudhuri, Das and Narasayya [9] considered the notion of a distribution over queries and assert that the query log is a random sample from that distribution, an assumption we later make as well. They use standard stratified sampling on *fundamental regions* which are sets of records with identical responses to the queries in the query log. It is worth noting that Acharya et al. [5] use a similar *finest partitioning* concept. In our setting, we assume the queries in the query log are rich enough so that *fundamental regions* may degenerate to containing single records. Nevertheless, if the formulation of Chaudhuri et al. [9] is taken to its logical conclusion, their result resembles our Empirical Risk Minimization (ERM) approach. Their solution of the optimization problem however does not carry over. Chaudhuri et al. [10] also use the query log to identify outlier records. Those are indexed separately and not sampled. While their approach mainly focuses on query execution speed, one can distill a sampling scheme from it.

The work of Joshi and Jermaine [11] is closely related to ours. They generate a large number of distributions by taking convex combinations of Neyman allocations of individual strata of single queries. The chosen solution is the one that minimized the observed variance on the query log. Their algorithm reportedly performs well in experiments and is close in spirit to the algorithm we propose. Unfortunately, their methodology falls short on several accounts. Theoretically, they do not argue about future queries which is the main motivation for their work. Practically, they suggest an inefficient algorithm and do not recognize the critical importance of regularization.

Several recent lines of work investigate a dynamic setting where the sampling technique is adaptive to the error bounds sought by the query issuer [12, 13, 14]. These ideas combined with modern data infrastructures [12, 13] lead to impressive practical results.

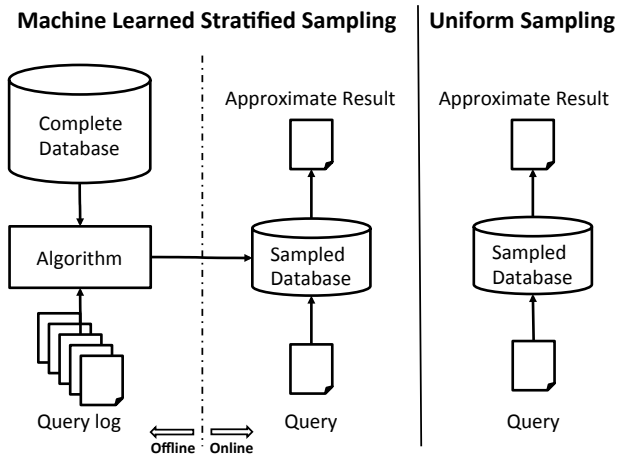

Figure 1: Standard architecture for database sampling. Our algorithm determines sampling probabilities for individual records based solely on a historical query log.

1.3 Our Contributions

In this paper we approach this problem in its fullest generality. We allow each record to be sampled with a different probability. Then, we optimize these probabilities to minimize the expected error of estimating *future* queries. Since future queries are unknown we must make an assumption about their nature. Our only assumption is that past and future queries are drawn independently from the same *unknown* distribution. This allows us to embed the stratification task into a machine learning context and open the door to new algorithms and analyses. Our contributions can be summarized as follows.

1. We formalize stratified sampling as a machine learning problem in the PAC model (Section 2).

2. We propose a simple and efficient one pass algorithm for solving the regularized ERM problem (Section 3). This gives a fully automated solution which obtains provably good results for future queries.

[2]Neyman allocation is also known as Neyman optimal allocation.

3. We report extensive experimental results on both synthetic and real data from Yahoo's systems that showcase the effectiveness of our proposed solution (Section 4).

2. SAMPLING IN THE PAC MODEL

In machine learning as a whole and in the PAC model specifically, one assumes that examples are drawn i.i.d. from an unknown distribution (e.g. [15, 16]). Given a random collection of such samples — a training set — the goal is to train a model that is accurate in expectation for future examples (over the unknown distribution). Our setting is very similar. Let p_i be the probability with which we pick record i. Let q_i denote the query q evaluated for record i and $y = \sum_i q_i$ is the correct exact answer for the query. Let $\tilde{y} = \sum_{i \in S} q_i/p_i$ where $i \in S$ with probability p_i be the Horvitz-Thompson estimator for y. The value y can be thought of as the label for query q and $\tilde{y}$ as the result our sampling would have predicted for it. The model in this analogy is the vector of probabilities p.

A standard objective in machine leaning theory (see for example [17]) is to minimize the risk $R(p)$ of the model p

$$R(p) = \mathbb{E}_q \, \mathrm{L}(p, q) \ .$$

Here, and throughout, $\mathbb{E}_q[\cdot]$ stands for the expectation over the unknown distribution from which queries q are drawn. Note that in Machine Learning, the loss function is usually applied to the predicted value $\mathrm{L}(\tilde{y}, y)$. In our case $\tilde{y}$ itself is a random variable. We therefore overload the notion of the loss with $\mathrm{L}(p, q) = \mathbb{E}_{\tilde{y}} \, \mathrm{L}(\tilde{y}, y)$. Note that the expectation $\mathbb{E}_{\tilde{y}}[\cdot]$ is taken only with respect to the random bits of the sampling procedure.

The simplest and most well studied loss function is the squared loss

$$\mathrm{L}(\tilde{y}, y) = (\tilde{y} - y)^2 \ \rightarrow \ \mathrm{L}(p, q) = \sum q_i^2 (1/p_i - 1) \ .$$

Optimizing for relative squared loss $\mathrm{L}(\tilde{y}, y) = (\tilde{y}/y - 1)^2$ is possible simply by dividing the loss by y^2. For notational brevity, the absolute squared loss is used for the algorithm presentation and mathematical derivations. In the experimental section we report results for the relative loss which turns out to be preferred by most practitioners. The reader should keep in mind that both absolute and relative squared losses fall under the exact same formulation.

The *absolute value loss* $\mathrm{L}(\tilde{y}, y) = |\tilde{y} - y|$ was considered by [9]. While it is a very reasonable measure of loss it is problematic in the context of optimization. First, there is no simple closed form expression for its expectation over $\tilde{y}$. While this does not rule out gradient descent based methods it makes them much less efficient. A more critical issue with setting $\mathrm{L}(\tilde{y}, y) = |\tilde{y} - y|$ is the fact that $\mathrm{L}(p, q)$ is, in fact, not convex in p. To verify, consider a dataset with only two records and a single query $(q_1, q_2) = (1, 1)$. Setting $(p_1, p_2) = (0.1, 0.5)$ or $(p_1, p_2) = (0.5, 0.1)$ gives $\mathbb{E}_{\tilde{y}}[|\tilde{y} - y|] = 1.8$. Setting $(p_1, p_2) = (0.3, 0.3)$ yields $\mathbb{E}_{\tilde{y}}[|\tilde{y} - y|] = 1.96$. This contradicts the convexity of L with respect to p.

3. EMPIRICAL RISK MINIMIZATION

Empirical Risk Minimization (ERM) is a standard approach in machine learning in which the chosen model is the minimizer of the empirical risk. The empirical risk $R_{emp}(p)$ is defined as an average loss of the model over the training set Q. Here Q is a query log containing a random collection of queries q drawn independently from the unknown query distribution.

$$p_{emp} = \arg\min_p R_{emp}(p) = \arg\min_p \frac{1}{|Q|} \sum_{q \in Q} \mathrm{L}(p, q)$$

Notice that, unlike most machine learning problems, one could trivially obtain zero loss by setting all sampling probabilities to 1. This clearly gives very accurate "estimates" but also, obviously, achieves no reduction in the database size. In this paper we assume that retaining record i incurs cost c_i and constrain the sampling to a fixed budget B. The interesting scenario for sampling is when $\sum c_i \gg B$. By enforcing that $\sum p_i c_i \leq B$ the expected cost of the sample fits the budget and the trivial solution is disallowed.

ERM is usually coupled with regularization because aggressively minimizing the loss on the training set runs the risk of overfitting. We introduce a regularization mechanism by enforcing that $p_i \geq \zeta$ for some small threshold $0 \leq \zeta \leq B/\sum_i c_i$. When $\zeta = 0$ no regularization is applied. When $\zeta = B/\sum_i c_i$ the regularization is so severe that uniform sampling is the only feasible solution. This type of regularization both insures that the variance is never infinite and guarantees some accuracy for arbitrary queries (see Section 1.1). To sum up, p_{emp} is the solution to the following constrained optimization problem:

$$\begin{aligned} p_{emp} \quad = \quad & \arg\min_p \frac{1}{|Q|} \sum_{q \in Q} \sum_i q_i^2 (1/p_i - 1) \\ & s.t. \quad \sum_i p_i c_i \leq B \quad \text{and} \quad \forall \, i \ \ p_i \in [\zeta, 1] \end{aligned}$$

This optimization is computationally feasible because it minimizes a convex function over a convex set. Therefore, gradient descent is guaranteed to converge to the global optimal solution.

A (nearly) closed form solution to this constrained optimization problem uses the standard method of Lagrange multipliers. The ERM solution, p_{emp}, minimizes

$$\begin{aligned} \max_{\alpha, \beta, \gamma} [& \frac{1}{|Q|} \sum_{q \in Q} \sum_i q_i^2 (1/p_i - 1) - \sum_i \alpha_i (p_i - \zeta) \\ & - \sum_i \beta_i (1 - p_i) - \gamma (B - \sum_i p_i c_i)] \end{aligned}$$

where α_i, β_i and γ are nonnegative. By complementary slackness conditions, if $\zeta < p_i < 1$ then $\alpha_i = \beta_i = 0$. Taking the derivative with respect to p_i we get that

$$p_i \propto \sqrt{\tfrac{1}{c_i} \tfrac{1}{|Q|} \textstyle\sum_{q \in Q} q_i^2}$$

Combining with the above, for some constant λ we have $p_i = \textsc{clip}_\zeta^1(\lambda z_i)$ where $z_i = \sqrt{\frac{1}{c_i} \frac{1}{|Q|} \sum_{q \in Q} q_i^2}$ and

$$\textsc{clip}_\zeta^1(z) = \max(0, \min(1, z)) \ .$$

The value for λ is the maximal value such that $\sum p_i c_i \leq B$ and can be computed by binary search. This method for

computing p_{emp} is summarized by Algorithm 1, which only makes a single pass over the training data (in Line 3).

Algorithm 1 Train: regularized ERM algorithm

1: **input:** training queries Q,
budget B, record costs c,
regularization factor $\eta \in [0,1]$
2: $\zeta = \eta \cdot (B/\sum_i c_i)$
3: $\forall\, i\ \ z_i = \sqrt{\frac{1}{c_i}\frac{1}{|Q|}\sum_{q\in Q} q_i^2}$
4: Binary search for λ satisfying $\sum_i c_i \,\textsc{clip}^1_\zeta(\lambda z_i) = B$
5: **output:** $\forall\, i\ \ p_i = \textsc{clip}^1_\zeta(\lambda z_i)$

3.1 Model Generalization

The reader is reminded that we would have wanted to minimize the *risk* which means finding

$$p^* = \arg\min_{p'} R(p') = \arg\min_{p'} \mathbb{E}_q\, \mathrm{L}(p', q)\ .$$

However, Algorithm 1 minimizes the *empirical risk* by finding

$$p = \arg\min_{p'} R_{emp}(p') = \arg\min_{p'} \frac{1}{|Q|}\sum_{q\in Q} \mathrm{L}(p', q)\ .$$

Generalization, in this context, refers to the risk associated with the empirical minimizer. That is, $R(p)$. Standard generalization results from machine learning reason about $|R(p) - R(p^*)|$ as a function of the number of training examples and the complexity of the learned concept.

For classification, the most common measure of model complexity is the Vapnik-Chervonenkis (VC) dimension. A comprehensive study of the VC dimension of SQL queries was presented by Riondato et al. [18]. For regression problems, such as ours, Rademacher complexity (see for example [19] and [20]) is a more appropriate measure. Moreover, it is directly measurable on the training set which is of great practical importance.

Luckily, here, we can bound the generalization directly. Let $z_i^* = \sqrt{(1/c_i)\mathbb{E}_q q_i^2}$. Notice that, if we replace z_i by z_i^* in Algorithm 1 we obtain the optimal solution p^*.

The proof begins by showing that z_i^* and z_i are $1\pm\varepsilon$ approximations of one another, and that ε diminishes proportionally to $\sqrt{1/|Q|}$. This will yield that the values of λ and λ^*, p_i and p_i^*, and finally $R(p)$ and $R(p^*)$ are also $1 \pm O(\varepsilon)$ multiplicative approximations of one another which concludes our claim.

For a single record, the variable z_i^2 is a sum of i.i.d. random variables. Moreover, $z_i^{*2} = \mathbb{E}_q z_i^2$. Using Hoeffding's inequality we can reason about the difference between the two values.

$$\Pr\left[\left|z_i^2 - z_i^{*2}\right| \geq \varepsilon z_i^{*2}\right] \leq 2e^{-2|Q|\varepsilon^2\,/\,\mathrm{skew}^2(i)}\ .$$

Definition: The skew of a record is defined as

$$\mathrm{skew}(i) = (\max_q q_i^2)/(\mathbb{E}_q q_i^2)\ .$$

It captures the variability in the values a single record contributes to different queries. Note that $\mathrm{skew}(i)$ is not directly observable. Nevertheless, $\mathrm{skew}(i)$ is usually a small constant times the reciprocal probability of record i being selected by a query.

Taking the union bound over all records, we get the minimal value for ε for which we succeed with probability $1-\delta$.

$$\varepsilon = O\left(\sqrt{\frac{\max_i \mathrm{skew}^2(i)\log(n/\delta)}{|Q|}}\right)$$

From this point on, it is safe to assume $z_i^*/(1+\varepsilon) \leq z_i \leq (1+\varepsilon)z_i^*$ for all records i simultaneously. To prove that $\lambda^* \leq (1+\varepsilon)\lambda$ assume by negation that $\lambda^* > (1+\varepsilon)\lambda$. Because $\textsc{clip}^1_\zeta$ is a monotone non-decreasing function we have that

$$\begin{aligned} B &= \sum c_i \,\textsc{clip}^1_\zeta(\lambda^* z_i^*) > \sum c_i \,\textsc{clip}^1_\zeta(\lambda(1+\varepsilon) z_i^*) \\ &> \sum c_i \,\textsc{clip}^1_\zeta(\lambda z_i) = B \end{aligned}$$

The contradiction proves that $\lambda^* \leq (1+\varepsilon)\lambda$. Using the fact that $\textsc{clip}^1_\zeta(x) \geq \textsc{clip}^1_\zeta(ax)/a$ for all $a \geq 1$ we observe

$$\begin{aligned} p_i &= \textsc{clip}^1_\zeta(\lambda z_i) \geq \textsc{clip}^1_\zeta(\lambda z_i(1+\varepsilon)^2)/(1+\varepsilon)^2 \\ &\geq \textsc{clip}^1_\zeta(\lambda^* z_i^*)/(1+\varepsilon)^2 = p_i^*/(1+\varepsilon)^2 \end{aligned}$$

Finally, a straightforward calculation shows that

$$\begin{aligned} R(p) &= \sum_i (1/p_i - 1)\mathbb{E}_q q_i^2 \\ &\leq \sum_i \left((1+\varepsilon)^2/p_i^* - 1\right)\mathbb{E}_q q_i^2 \\ &\leq (1+3\varepsilon)\sum_i (1/p_i^* - 1)\,\mathbb{E}_q q_i^2 + 3\varepsilon \sum_i \mathbb{E}_q q_i^2 \\ &\leq (1+O(\varepsilon))R(p^*)\ . \end{aligned}$$

The last inequality requires that $\sum_i \mathbb{E}_q q_i^2$ is not much larger than $R(p^*) = \sum_i (1/p_i^* - 1)\,\mathbb{E}_q q_i^2$. This is a very reasonable assumption. In fact, in most cases we expect $\sum_i \mathbb{E}_q q_i^2$ to be much smaller than $\sum_i (1/p_i^* - 1)\,\mathbb{E}_q q_i^2$ because the sampling probabilities tend to be rather small. This concludes the proof of our generalization result

$$R(p) \leq R(p^*)\left(1 + O\left(\sqrt{\frac{\max_i \mathrm{skew}^2(i)\log(n/\delta)}{|Q|}}\right)\right)\ .$$

4. EXPERIMENTS

In the previous section we proved that if ERM is given a sufficiently large number of training queries it will generate sampling probabilities that are nearly optimal for answering future queries.

In this section we present an array of experimental results using our algorithm. We compare it to uniform sampling and stratified sampling. We also study the effects of varying the number of training example and strength of the regularization. This is done for both synthetic and real datasets.

Our experiments focus exclusively on the relative error defined by $\mathrm{L}(\tilde{y}, y) = (\hat{y}/y - 1)^2$. As a practical shortcut, this is achievable without modifying Algorithm 1 at all. The only modification needed is normalizing all training queries such that $y = 1$ before executing Algorithm 1. The reader can easily verify that this is mathematically identical to minimizing the relative error. Algorithm 2 describes the testing phase reported below.

Algorithm 2 Test: measure expected test error.

input: Test queries Q, probability vector p
for $q \in Q$ **do**
  $y_q \leftarrow \sum_i q_i$
  $v_q^2 = \mathbb{E}(\tilde{y}_q/y_q - 1)^2 = (1/y_q^2)\sum_i q_i^2(1/p_i - 1)$
output: $(1/|Q|)\sum_{q\in Q} v_q^2$

4.1 Details of Datasets

Cube Dataset. The Cube Dataset uses synthetic records and synthetic queries which allows us to dynamically generate queries and test the entire parameter space. A record is a 5-tuple $\{x_k; 1 \le k \le 5\}$ of random real values, each drawn uniformly at random from the interval $[0, 1]$. The dataset contained 10000 records. A query $\{(t_k, s_k); 1 \le k \le 5\}$ is a 5-tuple of pairs, each containing a random threshold t_k in $[0, 1]$ (uniformly) and a randomly chosen sign $s_k \in \{-1, 1\}$ with equal probability. We set $q_x = 1$ iff $\forall k, s_k(x_k - t_k) \ge 0$ and zero else. We also set all record costs to $c_i = 1$. The length of the tuples and the number of record is arbitrary. Changing those yields qualitatively similar results.

DBLP Dataset. In this dataset we use a real database from DBLP and synthetic queries. Records correspond to 2,101,151 academic papers from the DBLP public database [21]. From the publicly available DBLP database XML file we selected all papers from the 1000 most populous venues. A venue could be either a conference or a journal. From each paper we extracted the title, the number of authors, and the publication date. From the titles we extracted the 5000 most commonly occurring words (deleting several typical stop-words such as "a", "the" etc.).

Next 50,000 random queries were generated as follows. Select one title word w uniformly at random from the set of 5000 commonly occurring words. Select a year y uniformly at random from $1970, \ldots, 2015$. Select a number k of authors from $1, \ldots, 5$. The query matches papers whose titles contain w and one of the following four conditions (1) the paper was published on or before y (2) the paper was published after y (3) the number of authors is $\le k$ (4) the number of authors is $> k$. Each condition is selected with equal probability. A candidate query is rejected if it was generated already or if it matches fewer than 100 papers. The 50,000 random queries were split into 40,000 for training and 10,000 for testing.

YAM+ Dataset. The YAM+ dataset was obtained from an advertising system at Yahoo. Among its many functions, YAM+ must efficiently estimate the reach of advertising campaigns. It is a real dataset with real queries issued by campaign managers. In this task, each record contains a single user's advertising related actions. The result of a query is the number of users, clicks or impressions that meet some conditions.

In this task, record costs c_i correspond to their volume on disk which varies significantly between records. The budget is the pre-specified allotted disk space available for storing the samples. Moreover, unlike the above two examples, the values q_i often represent the number of matching events for a given user. These are not binary but instead vary between 1 and 10,000. To set up our experiment, 1600 contracts (campaigns) were evaluated on 60 million users, yielding 1.6 billion nonzero values of q_i.

Dataset Sampling Rate	Cube 0.1	DBLP 0.01	YAM+ 0.01
Uniform Sampling	0.664	0.229	0.104
Neyman Allocation	0.643	0.640	0.286
Regularized Neyman	0.582	0.228	0.102
ERM-η, smallest training set	0.637	0.222	0.096
ERM-ρ, smallest training set	0.623	0.213	0.092
ERM-η, largest training set	**0.233**	0.182	0.064
ERM-ρ, largest training set	**0.233**	**0.179**	**0.059**

Figure 2: Average expected relative squared errors on test set for two standard baselines (uniform sampling and Neyman allocation); one novel baseline (Regularized Neyman); and ERM using two regularization methods. Note that Neyman allocation is worse than uniform sampling for two of the three datasets, and that "Regularized Neyman" works better than either of them on all three datasets. The best result for each dataset is shown in bold text. In all cases it is achieved by regularized ERM. Also, more training data reduces the testing error, which is to be expected. Surprisingly, a heuristic variant of the regularization (Section 4.4) slightly outperforms the one analyzed in the paper.

The queries were subdivided to training and testing sets each containing 800 queries. All training queries were chronologically issued before any of the testing queries. The training and testing sets each contained roughly 60 queries that matched fewer than 1000 users. These were discarded since they are considered by YAM+ users as too small to be of any interest. As such, approximating them well is unnecessary.

4.2 Baseline Methods

We used three algorithms to establish baselines for judging the performance of regularized ERM. Both of the first two algorithms, uniform sampling and Neyman allocation, are well known and widely used. The third algorithm was a novel hybrid of uniform sampling and Neyman allocation that was inspired by our best-performing version of regularized ERM.

4.2.1 Standard Baseline Methods

The most important baseline method is uniform random sampling. It is widely used by practitioners and has been proved optimal for adversarial queries. Moreover, as shown in Section 1, it is theoretically well justified.

The second most important (and well known) baseline is Stratified Sampling, specifically Neyman allocation (also known as Neyman optimal allocation). Stratified Sampling as a whole requires the input records to be partitioned into disjoint sets called strata. In the most basic setting, the optimal sampling scheme divides the budget equally between

the strata and then uniformly samples within each stratum. This causes the sampling probability of a given record to be inversely proportional to the cardinality of its stratum. Informally, this works well when future queries correlate with the strata and therefore have plenty of matching samples.

4.2.2 Strata for Neyman Allocation

The difficulty in applying Neyman allocation to a given data system lies in designing the strata. This task incurs a large overhead for developing the necessary insights into the structure of the database and the queries.

Our experiment used the most reasonable strata we could come up with. It turned out that the only dataset where Neyman allocation beat uniform random sampling was the synthetic Cube Dataset, whose structure we understood completely (since we designed it). This, however, does not preclude the possibility that better strata would have produced better results and possibly have improved on uniform random sampling for the other datasets as well.

Strata for Cube Data Set. For the Cube Dataset, we happened to know that good coverage of the corners of the cube is important. We therefore carved out the 32 corners of the cube and assigned them to a separate "corners" stratum as follows. A point was assigned to this stratum if $\forall k \in \{1 \ldots 5\},\ \min(x_k, 1 - x_k) < C$ where the threshold $C = (1/160)^{(1/5)} \approx 0.362$ was chosen so that the total volume of the corners stratum was 20 percent of the volume of the cube. This corners stratum was then allocated 50 percent of the sampling scheme's space budget. This caused the sampling probabilities of points in the corners stratum to be 4 times larger than the probabilities of other points.

Strata for DBLP Data Set. For the DBLP dataset, we experimented with three different stratifications that could plausibly correlate with queries: 1) by paper venue, 2) by number of authors, and 3) by year. Stratification by year turned out to work best, with number of authors a fairly close second.

Strata for YAM+ Data Set. For the YAM+ dataset users were put into separate partitions by the type of device they use most often (smartphone, laptop, tablet etc.) and available ad-formats on these devices. This creates 71 strata. YAM+ supports Yahoo ads across many devices and ad-formats and advertisers often choose one or a few formats for their campaigns. Therefore, this partition respects the structure of most queries. Other reasonable partitions we experimented with did not perform as well. For example, partition by user age and/or gender would have been reasonable but it correlates poorly with the kind of queries issued to the system.

4.2.3 Novel Baseline Method

The winner of a preliminary round of experiments was ERM with mixture regularization (see Section 4.4). We realized that this same regularization method could be applied to probability vectors produced by Neyman allocation, possibly improving the results and providing a stronger baseline. Because this baseline method uses mixture regularization, we defer its detailed description to Section 4.5.

4.2.4 Results for Baseline Methods

Figure 2 tabulates the baseline results against which the accuracy of regularized ERM is judged. The sampling rate is $B/\sum c_i$. The rest of the rows contain the quantity $(1/|Q|)\sum_{q\in Q} v_q^2$, the output of Algorithm 2.

A comparison of the two standard baseline methods shows that uniform random sampling worked better than Neyman allocation for both of the datasets that used real records and whose structure was therefore complex and somewhat obscure.

Results for our novel baseline method appear in the table row labeled "Regularized Neyman", and are discussed in Section 4.5.

4.3 Main Experimental Results

Figure 3 shows the results of applying Algorithm 1 to the three above datasets. There is one plot per dataset. In all three plots the y-axis is the average expected normalized squared error as measured on the testing queries; lower values are better. The different curves in each plot in Figure 3 report the results for a different size of training set. The worst results (highest curve) correspond to the smallest training set. The best results (lowest curve) are for the largest training set. There is also a black line across the middle of the plot showing the performance of uniform random sampling at the same average sampling rate (budget). More training data yields better generalization (and clearly does not affect uniform sampling). This confirms our hypothesis that the right model is learned.

The x-axis in Figure 3 varies with the value of the parameter η which controls the strength of regularization. Moving from left to right means that stronger regularization is being applied. When the smallest training set is used (top curve), ERM only beats uniform sampling when very strong regularization is applied (towards the right side of the plot). However, the larger the training set becomes, the less regularization is needed. This effect is frequently observed in many machine learning tasks where smaller training sets require stronger regularization to prevent overfitting.

It is important to point out that overfitting is a very real concern. Most settings suffer from significant overfitting if not enough regularization is applied. The crucial role of regularization is another novel contribution of this work which was not explained by previous art.

4.4 Mixture Regularization

In Algorithm 1, the amount of regularization is determined by a probability floor whose height is controlled by the user parameter η. We have also experimented with a different regularization that seems to work slightly better. In this method, unregularized sampling probabilities p are generated by running Algorithm 1 with $\eta = 0$. Then, regularized probabilities are computed via the formula $p' = (1-\rho)p + \rho u$ where $u = B/(\sum_i c_i)$ is the uniform sampling rate that

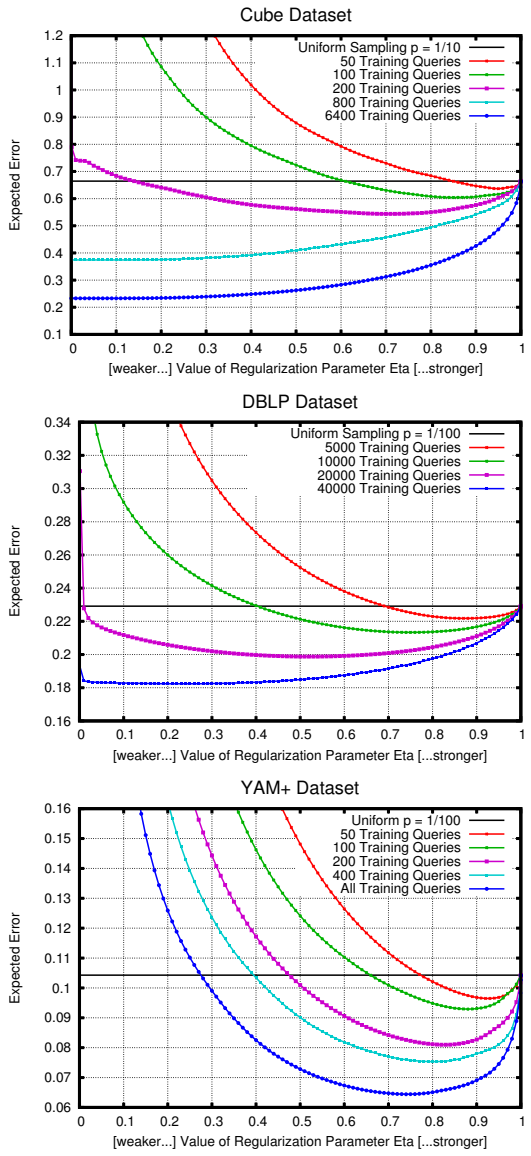

Figure 3: The three plots correspond to the three datasets. The y-axis is the average expected normalized squared error on the testing queries (lower values are better). The different curves in each plot correspond to different sizes of training sets (see the legend). The black horizontal line corresponds to uniform sampling using a similar sampling rate. The value of η (strength of regularization) varies along the x-axis.

would hit the space budget. Note that p' is a convex combinations of two feasible solutions to our optimization problem and is therefore also a feasible solution. Test error as a function of training set size and the value of ρ are almost identical to those achieved by η-regularization (Figure 3). The only difference is that the minimum testing errors achieved by mixture regularization are slightly lower. Some of these minima are tabulated in Figure 2. This behavior could be specific to the data used but could also apply more generally.

4.5 Neyman with Mixture Regularization

The Mixture Regularization method described in Section 4.4 can be applied to any probability vector, including a vector generated by Neyman allocation. The resulting probability vector is a convex combination of a uniform vector and a Neyman vector, with the fraction of uniform controlled by a parameter $\rho \in [0, 1]$. We note that this idea is similar in spirit to *Congressional Sampling* [5].

The estimation accuracy of Neyman with Mixture Regularization is tabulated in the "Regularized Neyman" row of Figure 2. Each number was measured using the best value of ρ for the particular dataset (tested in 0.01 increments). We note that this hybrid method worked better than either uniform sampling or standard Neyman allocation.

4.6 Accuracy as a Function of Query Size

Our main experimental results show that (with appropriate regularization) ERM can work better overall than uniform random sampling. However, there is no free lunch. The method intuitively works by redistributing the overall supply of sampling probability, increasing the probability of records involved in hard queries by taking it away from records that are only involved in easy queries. This decreases the error of the system on the hard queries while increasing its error on the easy queries. This tradeoff is acceptable because easy queries initially exhibit minuscule error rates and remain well below an acceptable error rate even if increased.

We illustrate this phenomenon using scatter plots that have a separate plotted point for each test query showing its expected error as a function of its *numeric cardinality* $\text{card}(q) := \sum |q_i| / \max |q_i|$. As discussed in Section 1.1, the numeric cardinality is a good measure of how hard it is to approximate a query result well using a downsampled database.

These scatter plots appear in Figure 4. There is one plot for each of the three datasets. Also, within each plot, each query is plotted with two points; a blue one showing its error with uniform sampling, and a red one showing its error with regularized ERM sampling.

For high cardinality (easy) queries ERM typically exhibits more error than uniform sampling. For example, the extreme cardinality queries for the Cube dataset experience a 0.001 error rate with uniform random sampling. With our solution the error increases to 0.005. This is a five fold increase but still well below an average 0.25 error in this setting. For low cardinality (hard) queries, ERM typically achieves less error than uniform sampling. However, it does not exhibit lower error on *all* of the hard queries. That is because error is measured on testing queries that were not seen during training. Predicting the future isn't easy.

4.7 Variability Caused by Sampling Choices

The quantity $\frac{1}{|Q|} \sum_{q \in Q} v_q^2$ output by Algorithm 2 is the average expected normalized squared error on the queries of the testing set. While this expected test error is minimized by the algorithm, the actual test error is a random variable that depends on the random bits of the sampling algorithm. Therefore, for any specific sampling, the test error could be either higher or lower than its expected value. The same

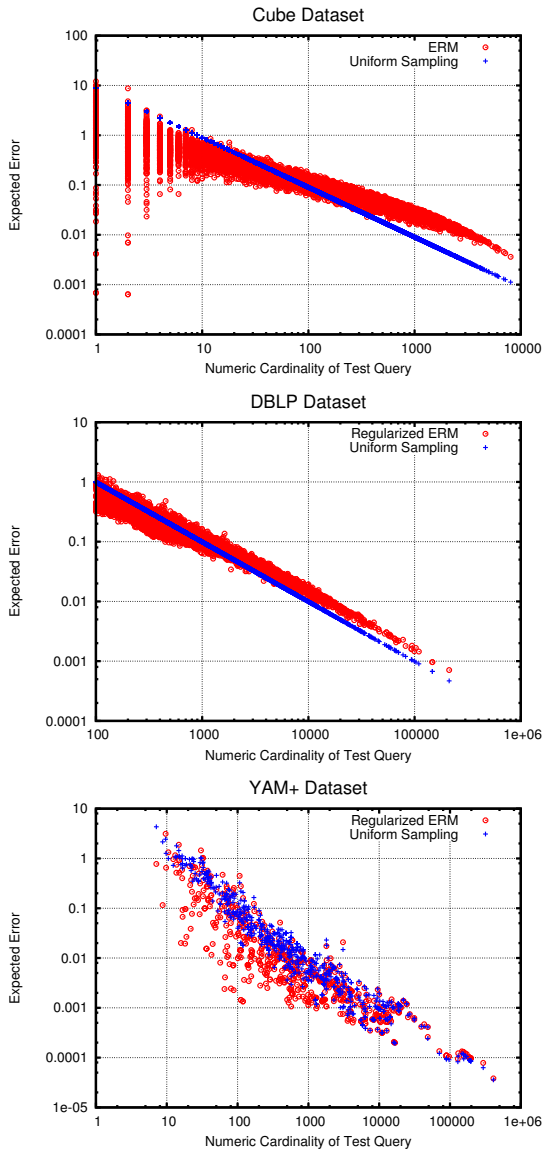

Figure 4: These three plots show the expected error of each test query. Clearly, for all three datasets, error is a generally decreasing function of the numeric cardinality of the queries. The advantage of ERM over uniform random sampling lies primarily at the more difficult low cardinality end of the spectrum.

thing is true for uniform random sampling. Given this additional source of variability, it is possible that a concrete sample obtained using ERM could perform worse than a concrete sample obtained by uniform sampling, even if the *expected* error of ERM is better.

To study the variability caused by sampling randomness, we first computed two probability vectors, p_e and p_u for the YAM+ dataset. The former was the output of ERM with mixture regularization with $\rho = 0.71$ (its best value for this dataset). The latter was a uniform probability vector with the same effective sampling rate (0.01). These were kept fixed throughout the experiment.

Next the following experiment was repeated 3000 times. In each trial a random vector, r, of n random numbers was created. Each of the values r_i was chosen uniformly at random from $[0, 1]$.

The two concrete samples specified by these values are

$$i \in S_e \text{ if } p_{e,i} < r_i \quad \text{and} \quad i \in S_u \text{ if } p_{u,i} < r_i$$

Finally we measured the average normalized squared error over the testing queries for the concrete samples S_e and S_u. The reason for this construction is so that the two algorithms use the exact same random bits.

Smoothed histograms of these measurements for regularized ERM and for uniform sampling appear in Figure 7. For esthetic reasons, these histograms were smoothed by convolving the discrete data points with a narrow gaussian ($\sigma = 0.006$). They approximate the true distribution of concrete outcomes.

The two distributions overlap. With probability 7.2%, a specific ERM outcome was actually worse than the outcome of uniform sampling with the same vector of random numbers. Even so, from Figure 7 we clearly see the distribution for regularized ERM shifted to the left. This corresponds to the reduced expected loss but also shows that the mode of the distribution is lower.

Moreover, the ERM outcomes are more sharply concentrated around their mean, exhibiting standard deviation of 0.049 versus 0.062 using uniform sampling. This is despite the fact that the right tail of the ERM distribution was slightly worse, with 17/3000 outcomes in the interval $[0.4, 0.6]$ versus 11/3000 for uniform. The increased concentration is surprising because usually reducing expected loss comes at the expense of increasing its variance. This should serve as additional motivation for using the ERM solution.

4.8 Effect of Including Costs in Optimization

We now turn to inspecting the benefit of incorporating individual record costs (c_i) in the optimization solution. This is done by comparing Algorithm 1 against a *modified* version of Algorithm 1 that is oblivious to the record costs except when enforcing the space constraint. The modification consists of deleting the $\sqrt{1/c_i}$ term from the formula for z_i in line 3 of Algorithm 1. However, the c_i term in line 4 is retained so that the modified algorithm does not inadvertently gain an unfair advantage.

The results of this experiment appear in Figure 6. For the YAM+ data, record costs corresponded to the size of the record on disk. For the DBLP dataset, we created artificial costs uniformly distributed over $[0, 1]$. The red curves show test error as a function of ρ using our actual ERM system. The blue curves show the same measurements for the modified system that is partially oblivious to the record costs.

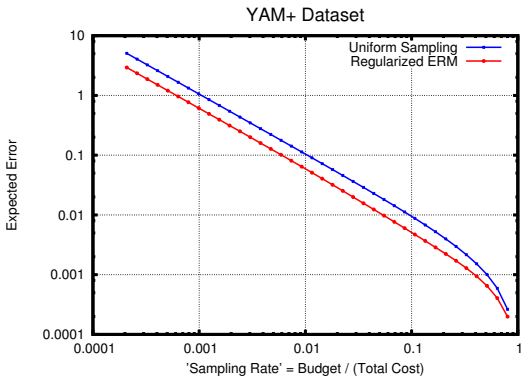

Figure 5: ERM with mixture regularization versus uniform random sampling at various effective sampling rates ($B/\sum c_i$). The gains might appear unimpressive in the log-log scale plot but are, in fact, 40%-50% throughout which is significant.

For the DBLP dataset with artificial costs, the partially oblivious system clearly performed worse. Oddly enough, for the YAM+ dataset, the two systems performed almost the same. We suspect that this has something to do with the fact that the costs c_i and the values q_i are both determined by the number of events for the given user, and hence are heavily correlated. The unmodified system that fully considered the costs c_i managed to fit slightly more records into the sample by favoring the smaller ones. However, these smaller records had less power to reduce the prediction error. We conjecture that these two effects roughly cancelled each other out.

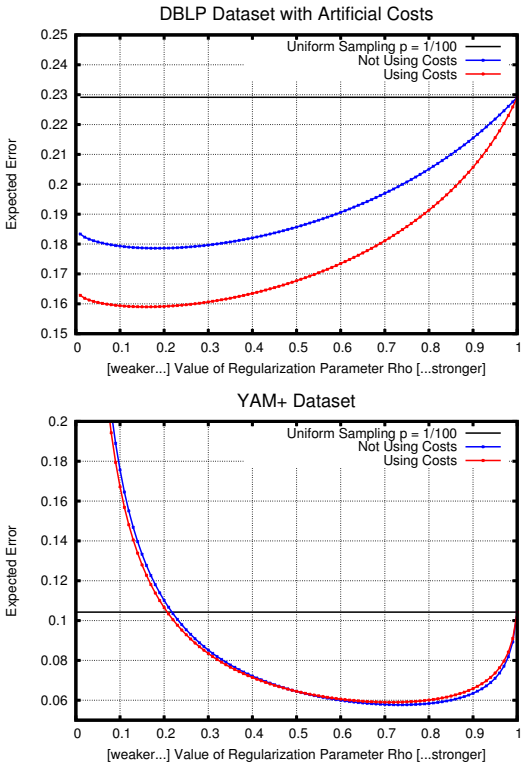

Figure 6: These plots show the effect of taking the costs c_i into account while computing the values z_i. For YAM+, record costs correspond to the size of the record on disk. For the DBLP dataset randomly generated artificial record costs were used.

5. CONCLUDING DISCUSSION

Using three datasets, we demonstrate that our machine learning based sampling and estimation scheme provides a useful level of generalization from past queries to future queries. That is, the estimation accuracy on future queries is better than it would have been had we used any combination of uniform or stratified sampling. Moreover, it is a disciplined approach that does not require any manual labor or data insights such as needed for using Stratified Sampling (creating strata). Since we believe most systems of this nature already store a historical query log, this method should be widely applicable.

The ideas presented extend far beyond the squared loss and the specific ERM algorithm analyzed. Machine learning theory allows us to apply this framework to any convex loss function using gradient descent based algorithms [22]. One interesting function to minimize is the deviation indicator function $\mathrm{L}(\tilde{y}, y) = 1$ if $|\tilde{y} - y| \geq \varepsilon y$ and zero else. This choice does not yield a closed form solution for $\mathrm{L}(p, q)$ but using Bernstein's inequality yields a tight bound that turns out to be convex in p. Online convex optimization [23] could give provably low regret results for any arbitrary sequence of queries. This avoids the i.i.d. assumption and could be especially appealing in situations where the query distribution is expected to change over time.

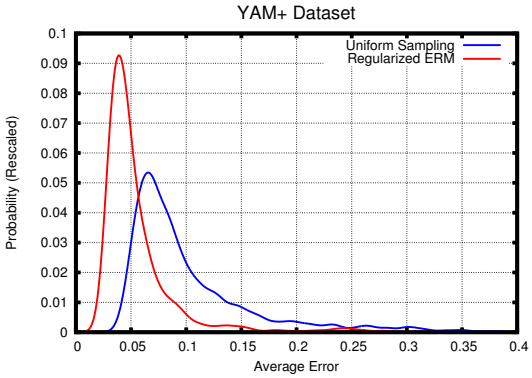

Figure 7: These smoothed histograms show the variability of results caused by random sampling decisions. Clearly, the distribution of outcomes for regularized ERM is preferable to that of uniform random sampling.

6. REFERENCES

[1] Frank Olken and Doron Rotem. Simple random sampling from relational databases. In *Proceedings of the 12th International Conference on Very Large Data Bases*, VLDB '86, pages 160–169, San Francisco, CA, USA, 1986. Morgan Kaufmann Publishers Inc.

[2] Frank Olken and Doron Rotem. Random sampling from database files: A survey. In *Statistical and Scientific Database Management, 5th International Conference SSDBM, Charlotte, NC, USA, April 3-5, 1990, Proccedings*, pages 92–111, 1990.

[3] Frank Olken. *Random Sampling from Databases*. PhD thesis, University of California at Berkeley, 1993.

[4] Joseph M. Hellerstein, Peter J. Haas, and Helen J. Wang. Online aggregation. *SIGMOD Rec.*, 26(2):171–182, June 1997.

[5] Swarup Acharya, Phillip B. Gibbons, and Viswanath Poosala. Congressional samples for approximate answering of group-by queries. *SIGMOD Rec.*, 29(2):487–498, May 2000.

[6] Edo Liberty, Michael Mitzenmacher, Justin Thaler, and Jonathan Ullman. Space lower bounds for itemset frequency sketches. *CoRR*, abs/1407.3740, 2014.

[7] Jerzy Neyman. On the two different aspects of the representative method: the method of stratified sampling and the method of purposive selection. *Journal of the Royal Statistical Society*, pages 558–625, 1934.

[8] William G Cochran. *Sampling techniques*. John Wiley & Sons, 2007.

[9] Surajit Chaudhuri, Gautam Das, and Vivek Narasayya. Optimized stratified sampling for approximate query processing. *ACM Trans. Database Syst.*, 32(2), June 2007.

[10] Surajit Chaudhuri, Gautam Das, Mayur Datar, Rajeev Motwani, and Vivek Narasayya. Overcoming limitations of sampling for aggregation queries. In *Data Engineering, 2001. Proceedings. 17th International Conference on*, pages 534–542. IEEE, 2001.

[11] Shantanu Joshi and Christopher Jermaine. Robust stratified sampling plans for low selectivity queries. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, ICDE '08, pages 199–208, Washington, DC, USA, 2008. IEEE Computer Society.

[12] Sameer Agarwal, Barzan Mozafari, Aurojit Panda, Henry Milner, Samuel Madden, and Ion Stoica. Blinkdb: Queries with bounded errors and bounded response times on very large data. In *Proceedings of the 8th ACM European Conference on Computer Systems*, EuroSys '13, pages 29–42, New York, NY, USA, 2013. ACM.

[13] Nikolay Laptev, Kai Zeng, and Carlo Zaniolo. Early accurate results for advanced analytics on mapreduce. *Proc. VLDB Endow.*, 5(10):1028–1039, June 2012.

[14] Sameer Agarwal, Henry Milner, Ariel Kleiner, Ameet Talwalkar, Michael Jordan, Samuel Madden, Barzan Mozafari, and Ion Stoica. Knowing when you're wrong: Building fast and reliable approximate query processing systems. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 481–492, New York, NY, USA, 2014. ACM.

[15] L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, November 1984.

[16] Michael J Kearns and Umesh Virkumar Vazirani. *An introduction to computational learning theory*. MIT press, 1994.

[17] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012.

[18] Matteo Riondato, Mert Akdere, UÇ§ur ÃÂĞetintemel, StanleyB. Zdonik, and Eli Upfal. The vc-dimension of sql queries and selectivity estimation through sampling. In Dimitrios Gunopulos, Thomas Hofmann, Donato Malerba, and Michalis Vazirgiannis, editors, *Machine Learning and Knowledge Discovery in Databases*, volume 6912 of *Lecture Notes in Computer Science*, pages 661–676. Springer Berlin Heidelberg, 2011.

[19] Peter L. Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *J. Mach. Learn. Res.*, 3:463–482, March 2003.

[20] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, New York, NY, USA, 2014.

[21] DBLP XML database. http://dblp.uni-trier.de/xml/.

[22] Elad Hazan and Satyen Kale. Beyond the regret minimization barrier: Optimal algorithms for stochastic strongly-convex optimization. *J. Mach. Learn. Res.*, 15(1):2489–2512, January 2014.

[23] Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. 2003.