Cheap Checking for Cloud Computing: Statistical Analysis via Annotated Data Streams

Graham Cormode **Christopher Hickey**
University of Warwick

Abstract

As the popularity of outsourced computation increases, questions of accuracy and trust between the client and the cloud computing services become ever more relevant. Our work aims to provide fast and practical methods to verify analysis of large data sets, where the client's computation and memory costs are kept to a minimum. Our verification protocols are based on defining "proofs" which are easy to create and check. These add only a small overhead to reporting the result of the computation itself. We build up a series of protocols for elementary statistical methods, to create more complex protocols for Ordinary Least Squares, Principal Component Analysis and Linear Discriminant Analysis, and show them to be very efficient in practice.

1 Introduction

The massive leap in popularity of machine learning techniques can be attributed in part not simply to novel algorithms, but also to dramatic increases in scale: much larger models with many parameters to set optimally, and much larger training data sets to determine these parameters. However, this presents a challenge to data owners who do not have a convenient data centre at their disposal. The size of data and computational cost in order to extract accurate models begins to look prohibitive. At the same time, a potential answer has emerged, in the form of outsourced computation. That is, instead of building the infrastructure needed to store and analyse large quantities of data, computation can be 'rented' on demand. Initially cloud offerings provided only the barebones of a remote system, but current options provide many tools, libraries and algorithms available to take "off the shelf".

Proceedings of the 21st International Conference on Artificial Intelligence and Statistics (AISTATS) 2018, Lanzarote, Spain. PMLR: Volume 84. Copyright 2018 by the author(s).

One doubt remains. If we send data off to the cloud, and request some analysis to be performed, what guarantee do we get that the processing has been done to our satisfaction? The provider has an economic incentive to cut corners: to perform the computation on only a sample of provided data, or to terminate an iterative parameter search before convergence has occurred, for example. Such short cuts yield plausible but suboptimal models. So how could we be assured that the best model has been found, without repeating the computation ourself or having multiple providers repeat the work, substantially driving up costs?

In this paper, we adopt the Annotated Data Streams approach (Chakrabarti et al., 2009) for verifying the models found by an outsourced provider Rather than repeating all or some of the computation, we instead provide protocols in which the cloud also provides some extra information that allows us to check a strict adherence to the required computation. The overhead for the cloud provider is minimal – often, the required information is a relatively low cost function of the input data or natural by-products of the target computation. These do not restrict the cloud to use any particular implementation or algorithm; just that they demonstrate that the output meets certain necessary properties. The key part of these protocols is that the information required is very easy for the original data owner to check, based on appropriately defined fingerprints of the input. These fingerprints can be computed flexibly and incrementally from the input as it arrives in any order, so the data owner does not even need to retain a complete copy of the input. The overhead for the data owner is therefore low: it is typically dominated by the cost of sending the data to the cloud and receiving the output of the computation. If the data owner's checks pass, then they are assured that the computation has been performed by the cloud satisfactorily, with a very high degree of certainty. One can then think of these protocols as providing effective "checksums for computation".

Work on annotated data streams draw on the theory of Interactive Proofs. This model was developed in the early 1990s as an alternate perspective on computational complexity. An early celebrated result was that the set of computations that could be effectively checked by a "weak" verifier corresponded exactly to the powerful class of computations that could be performed using polynomial space (PSPACE) (Babai, 1985; Goldwasser and Sipser, 1986; Shamir, 1992). Such results were initially thought to be of purely theoretical interest. A decade ago, this topic was revisited from a perspective closer to our own: to what extent could arbitrary programs be checked without fully repeating them (Goldwasser et al., 2008)? Several strong models were proposed, allowing a large class of computations to be checked in this way. However, the costs were still typically large: programs have to be compiled into non-standard formats (such as circuits with gates performing arithmetic operations), and the overheads for the cloud can be very substantial, often hundreds or even millions of times slower than directly performing the computation.

We break away from this paradigm, and achieve protocols that have minimal overheads by deliberately narrowing the scope of the computations considered. By focusing on a collection of important tasks in machine learning based on linear algebra, we can provide bespoke protocols based on "fingerprinting" the input data that take advantage of specific structure in the target problem. These problems are of sufficient generality that our efforts are repaid. After surveying prior work and providing a technical background (Section 2), we begin with primitives for ubiquitous steps in data analysis: matrix multiplication and inversion, Cholesky decomposition and eigenvalue finding (Section 3). In Section 4, we present applications of these to tasks of interest: regression, PCA, and LDA. These core tasks are sufficient to show the power of this paradigm. We provide empirical validation of our claims in Section 5.

This work represents some first steps in verifying outsourced computation of machine learning. The next steps are to extend this work to more complex models and algorithms currently enjoying popularity in machine learning, such as deep learning and beyond. Since, at the risk of drastic oversimplification, almost all of machine learning can be performed via numerical data encodings (vectors, matrices and tensors) combined with optimization, we are optimistic that the foundations laid in this work will naturally extend to further protocols for common mining and modelling tasks.

2 Preliminaries

2.1 Annotated Data Stream Model

In our protocols, we have a data stream $\mathcal{S}$, which is observed by two parties, a "helper" (H) and a "verifier" (V). The data stream will usually be arranged as a sequence of n tuples of elements, where each tuple typically defines an element of a larger structure, such as a matrix or vector. Abstractly, the verifier wishes to compute some function on $\mathcal{S}$, $f(\mathcal{S})$, with assistance from the helper. Typically, the helper will provide the value of $f(\mathcal{S})$, along with a proof of its correctness. This yields the *Annotated Data Stream* model, introduced by Chakrabarti et al. (2009). We formalize the model via the definitions below.

Definition 1. *We have a helper H, and a verifier V, with the aim of cooperating to compute some function $f(\mathcal{S})$ of the stream $\mathcal{S}$. The helper provides a message $M_H(\mathcal{S})$ comprised of , $\hat{f}$, the claimed value $f(\mathcal{S})$, and a proof P_H which supports this claim according to some pre-agreed structure. The output of V based on the stream $\mathcal{S}$, V's randomly chosen bits $\mathcal{R}_V$, and the helper's message is*

$$\mathrm{Out}_V(\mathcal{S}, \mathcal{R}_V, M_H(\mathcal{S})) = \begin{cases} \hat{f} & \textit{If } V \textit{ is convinced} \\ \perp & \textit{Otherwise} \end{cases}$$

*A protocol is defined by the functions M_H and Out_V. We say that a protocol is **complete** if*

$$\exists H : \mathbb{P}[\mathrm{Out}_V(\mathcal{S}, \mathcal{R}_V, M_H(\mathcal{S})) = f(\mathcal{S})] = 1 \qquad (1)$$

*The Verifier's protocol is **sound** if*

$$\forall H', \mathcal{S}' : \mathbb{P}\left[\mathrm{Out}_V(\mathcal{S}', \mathcal{R}_V, M_{H'}(\mathcal{S}')) \notin \{f(\mathcal{S}'), \perp\}\right] \leq \tfrac{1}{3}$$

Intuitively this says that we seek protocols so that an honest helper (one who faithfully follows the protocol) can always persuade the verifier to accept the correct answer, while a dishonest helper cannot persuade the verifier to accept an incorrect result with more than a constant probability. Our protocols allow this probability to be reduced to an arbitrarily small value with minimal cost.

In order to show an annotated data streaming protocol, we need to show completeness and soundness. This is sufficient to check that our protocol will successfully do what we want: verify the computation with a high probability of detecting a malicious helper. Trivial protocols always exist wherein the helper's message is null, and the verifier evaluates the function in full. Consequently, we seek protocols whose costs for the verifier are substantially lower than this. Ideally, the protocol should run in sublinear memory space for the verifier, without the need for intensive computation and the message should be as small as possible. Similarly, we seek protocols where the honest helper does not have to do substantially more work than simply computing $f(\mathcal{S})$. Our focus in this paper are on the memory space for the verifier, and the size of the proof, which we call the communication cost.

Definition 2. *A **(h,v)-protocol** is a valid annotated data streaming protocol using a message of size $O(h)$, and $O(v)$ space cost for the verifier.*

2.2 Fingerprint techniques

To build up complete protocols for the statistical methods, we start with fingerprinting, which provides equality tests on which our subsequent protocols rely.

Finite Fields. In line with prior work, all our protocols rely on computations performed over finite fields. For ease of implementation, we use prime fields. Given a prime q, the finite (prime) field $\mathbb{F}_q$ is the set $\{0 \dots q-1\}$ with addition and multiplication modulo q. Hence, storing field values requires $O(\log q)$ bits. We make use of the fact that in many cases arithmetic in the field and arithmetic over the integers is in exact correspondence. However, as we consider more complicated computations, we encounter situations where we seek solutions over the reals, which do not correspond to solutions in the field. To avoid this, we will use scaling and rounding techniques to approximate using field values. Specifically, we consider the input to be fixed precision rational numbers which can be represented as members of the set $\mathcal{F}_{\rho,M} = \{x \in \mathbb{R} \cap [-M, M] : b^\rho x \in \mathbb{Z}\}$, with respect to a base, b. We then choose the field size q as a function of ρ and M, in order to allow us to maintain the exact correspondence between the field and the fixed precision rationals. We map $y \in \mathcal{F}_{\rho,M}$ to $y' \in \mathbb{F}_q$, where $y' = \mod(xb^\rho, q)$, choosing q to be a prime bigger than $(2M+1)b^\rho$.

Fingerprints. Most of the values stored by our verifier will be *fingerprints* of large objects, such as vectors or matrices. Fingerprints (based on randomly chosen seeds x) have the property that if two fingerprints agree, then with high probability the objects that gave rise to the fingerprints are identical.

Definition 3 (Matrix fingerprint)**.** *For $A \in \mathbb{F}_q^{n\times m}$, the matrix fingerprint of A is $f_x^{\text{mat}}(A)$ with $x \in_R \mathbb{F}_q$ and*

$$f_x^{\text{mat}}(A) = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} A_{ij} x^{in+j}$$

From the Schwartz-Zippel Lemma (Shamir, 1992), we have that for a randomly chosen $x \in \mathbb{F}_q$, given $A, B \in \mathbb{F}_q^{n\times m}$, if $A \neq B$, then $\Pr[f_x(A) = f_x(B)] \leq \frac{nm-1}{q}$. Therefore for sufficently large q (say, greater than $3nm$) we will always have soundness and completeness for verifying equality with fingerprints. We can view vectors as a special case of matrices, so similarly for vectors $u \in \mathbb{F}_q$ we use the notation $f_x^{\text{vec}}(u) = \sum_{i=0}^{n-1} u_i x^i$ to denote a vector fingerprint. Fingerprints have several useful properties, such as linearity, with $f_x(A+B) = f_x(A) + f_x(B)$.

2.3 Related Work

We briefly survey the most related work in this area. Chakrabarti *et al.* (Chakrabarti et al., 2009) introduced the annotated streams model and provided protocols for frequency moments in data streams, and several graph problems, including triangle counting, connectivity and bipartite matchings. They also introduced a square matrix multiplication protocol built extending the classical result of Freivalds (1979). Subsequently, Cormode et al. (2013) provided further protocols for graph problems, using linear and integer programs to validate optimal matchings and shortest paths. More recently, Daruki et al. (2015) extended results on matrix analysis, provided more general protocols for matrix multiplication, and a protocol for eigenvalue (but not eigenpair) checking. For matrices $A \in \mathbb{F}_q^{k\times n}$ and $B \in \mathbb{F}_q^{n\times k'}$, they show an $(kk'h\log(q), v\log(q))$ −protocol, where $hv \geq n$. These protocols are used to perform shape fitting and clustering, although they shift away from annotated data streams and towards an interactive proof model which allows several exchanges of messages between helper and verifier. The annotated data stream model is generalized by definitions of *streaming interactive proofs* (SIPs) (Cormode et al., 2011, 2012). Note that annotated data stream verification protocols can be considered as single message SIPs.

3 Linear Algebraic Checks

In this section, we define protocols for checking a variety of linear algebraic primitives, based on careful use of fingerprints. We use the notation $A_i^{\rightarrow}$ to denote the ith row of the matrix A, and $A_i^{\downarrow}$ to denote the ith column of A.

3.1 Fingerprinting the Gramian Matrix

We first show how to efficiently build a fingerprint of the Gramian matrix $G = A^TA$ given a stream that specifies A.

Lemma 1. *For $A \in \mathbb{F}_q^{n\times m}$,*

$$f_x^{\text{mat}}(A^TA) = \sum_{i=1}^{n} f_{x^m}^{\text{vec}}(A_i^{\rightarrow}) f_x^{\text{vec}}(A_i^{\rightarrow})$$

Proof. Given $p, q \in \mathbb{F}_q^m$,

$$\begin{aligned} f_x^{\text{mat}}(p\otimes q) &= \sum_{i=0}^{m-1}\sum_{j=0}^{m-1} p_i q_j x^{im+j} \\ &= \sum_{i=0}^{m-1} p_i x^{im} \sum_{j=0}^{m-1} q_j x^j \quad , \\ &= f_{x^m}^{\text{vec}}(p) f_x^{\text{vec}}(q) \end{aligned}$$

Using the outer product definition of matrix multiplication;

$$\begin{aligned} f_x^{\text{mat}}(A^TA) &= \sum_{i=0}^{n-1} f_x^{\text{mat}}\left(\left((A^T)_i^{\downarrow}\right)(A_i^{\rightarrow})\right) \\ &= \sum_{i=0}^{n-1} f_{x^m}^{\text{vec}}(A_i^{\rightarrow}) f_x^{\text{vec}}(A_i^{\rightarrow}) \end{aligned}$$ □

Hence, we can compute the fingerprint of A^TA from A row-by-row and summing the product of row vector fingerprints. This immediately implies a protocol to verify that a matrix G provided by H is the Gramian: simply use the above identity to compute $f_x(A^TA)$ from the stream, and check that this is equal to $f_x(G)$. Soundness and completeness follow immediately from properties of fingerprints in Section 2.2, and the cost is $O(m^2\log q)$ communication to specify G, while the verifier can maintain the needed fingerprints in space $O(\log q)$.

3.2 Matrix Multiplication

We generalize the previous protocol to solve matrix multiplication. Given two matrices, $A \in \mathbb{F}_q^{k\times n}$ and $B \in \mathbb{F}_q^{n\times k'}$, a similar proof to Lemma 1 shows

$$f_x^{\mathrm{mat}}(AB) = \sum_{i=0}^{n-1} f_{x^{k'}}^{\mathrm{vec}}(A_i^{\downarrow}) f_x^{\mathrm{vec}}(B_i^{\rightarrow}) \tag{2}$$

Equation (2) allows for an efficient matrix multiplication protocol, where the main work of the helper is to repeat the input matrices in a convenient order. To find the fingerprint of AB we need to see each column of A and row of B interleaved in order, i.e. $M_H = \langle \tilde{A}_0^{\downarrow}, \tilde{B}_0^{\rightarrow}, ..., \tilde{A}_{n-1}^{\downarrow}, \tilde{B}_{n-1}^{\rightarrow} \rangle$. The verifier uses fingerprints to check that the reordered versions of A and B agree with the versions present in the stream $\mathcal{S}$, and that the claimed matrix product C has the same fingerprint as the fingerprint computed via (2).

Theorem 1. *There is a* $(\max(k'k, kn, k'n)\log(q)$, $\log(q))$*-protocol for verifying matrix multiplication with* $A \in \mathbb{F}_q^{k\times n}$, $B \in \mathbb{F}_q^{n\times k'}$.

Soundness and completeness are again immediate from the properties of fingerprints. The verifier keeps a constant number of fingerprints in space $O(\log(q))$, and the communication cost of $O(\max(k'k, kn, k'n)\log(q))$ is due to sending each of A, B and AB.

Matrix multiplication has been considered in prior work on annotated streaming. Daruki et al. (2015) showed that any protocol for this problem must have the product of the communication cost and space cost at least $\Omega((k + k')n)$. Our protocol achieves this lower bound up to logarithmic factors (noting that any protocol which reports (AB) requires $\Omega(kk')$ communication for this step). The previous best rectangular matrix multiplication protocol, achieved by Daruki et al. (2015), was a $\big(kk'h\log(q), v\log(q)\big)$-protocol with $hv \geq n$, that used the inner product protocol of Chakrabarti et al. (2009). Our protocol can be understood as setting $v = 1$, but removing the high overhead factor of n from the communication cost in this case. When our input matrices are constituted of fixed precision rationals, we choose q as follows:

Corollary 1. *Given* $A \in \mathcal{F}_{\rho,M}^{k\times n}$ *and* $B \in \mathcal{F}_{\rho,M}^{n\times k'}$*, the matrix* AB *is in* $\mathcal{F}_{2\rho,nM^2}^{n\times n}$*, and we choose* $q > n(2M+1)^2 2^{2\rho}$ *so that the product can be represented exactly in the field.*

Choosing q to be this large means that when we move A and B from $\mathcal{F}_{\rho,M}$ to $\tilde{A}, \tilde{B} \in \mathbb{F}_q$ by multiplying by 2^ρ, and then compute $\tilde{A}\tilde{B}$ all these values remain in $\mathbb{F}_q$, and by scaling back down by $2^{2\rho}$ we get our result in the desired format, without wraparound or rounding errors. The memory required to store elements of $\mathbb{F}_q$ is $\log(q) = O(\log(M) + \log(b)\rho)$, which is proportional to the space required to store the original matrices in $\mathcal{F}_{\rho,M}$ is $\log(M) + \log(b)\rho$.

Lower bounds on computing matrix fingerprints. The verifier needs very little memory for the above protocol, since the matrix is provided in a convenient order. More generally, we would like to be able to find $f_x^{\mathrm{mat}}(AB)$ from just $f_x^{\mathrm{mat}}(A)$ and $f_x^{\mathrm{mat}}(B)$, without requiring the helper to repeat these. This would reduce communication and simplify the protocol; however, we show this is not possible.

Theorem 2. *Any function* g *with* $f_x^{\mathrm{mat}}(AB) = g(f_x^{\mathrm{mat}}(A),$ $f_x^{\mathrm{mat}}(B))$ *for* $A, B \in \mathbb{F}_q^{n\times n}$ *requires that fingerprints* f_x^{mat} *are at least* $\Omega(n)$ *bits in size.*

Proof. We make use of a hard problem from communication complexity to show the space lower bound. In the DISJOINTNESS problem, two players Alice and Bob each have a bit string, a, $b \in \{0,1\}^n$, and they wish to see whether for any $i \in [n]$ they have $a_i = b_i = 1$. If we had a function $f_x^{\mathrm{mat}} : \mathbb{F}_q^{n\times n} \to \mathbb{F}_q$ they could create $n \times n$ matrices A and B with a and b on the diagonals and 0's elsewhere. Then Alice could send Bob $f_x^{\mathrm{mat}}(A)$, Bob could compute $f_x^{\mathrm{mat}}(B)$, and then find $f_x^{\mathrm{mat}}(AB)$ using g.Observe that $AB = \mathbf{0}$ iff strings a and b are disjoint, and is non-zero otherwise. So by comparing $f_x^{\mathrm{mat}}(AB)$ to $f_x^{\mathrm{mat}}(\mathbf{0})$, we can determine the answer to the disjointness problem. The fingerprints must be at least $\Omega(n)$ bits from the corresponding communication complexity of DISJOINTNESS (Håstad and Wigderson, 2007). $\square$

3.3 Eigenvalue Check

For the subsequent problems, we need to apply scaling and rounding, as mentioned earlier. There is a tension here, since matrix computations can include values which are very large compared to the input values. We also need to ensure that our approximation tolerance always allows an honest helper to find a satisfying answer, but prevents a dishonest helper from getting a wildly wrong answer accepted.

We first work with (approximate) eigenvectors and eigenvalues of a symmetric matrix, i.e. a matrix such that $\forall i, j.\ A_{ij} = A_{ji}$. We wish to find pairs (λ_i, v_i) over the reals, for all $i \in [n]$, such that A acts on each of the vectors by only scaling them by λ_i and that the vectors are orthonormal, i.e.

$$Av_i = \lambda_i v_i \qquad v_i^T v_j = \delta_{ij} = \begin{cases} 0 & i \neq j \\ 1 & i = j \end{cases}$$

Mapping these eigenpairs to the finite field can be delicate, since they may not align with coordinates in the field. Our protocol relies on a scaling factor T, which is used to multiply up values from the original domain. The field size q must grow by a corresponding factor to accommodate the large range of values. To tolerate this, we relax to allow approximate eigenvectors as defined below. We first show that rounding to the scaled field $\mathbb{F}_{Tq}$ is always possible. Consider a particular eigenvector v_i, and write $\widehat{v_i} = Tv_i + r$, with $r_j \in \left[-\frac{1}{2}, \frac{1}{2}\right]$, and $\widehat{\lambda_i} = T\lambda_i + \rho$, where $\rho \in \left[-\frac{1}{2}, \frac{1}{2}\right]$. The following theorem shows that this "rounded" eigenpair acts like the original eigenpair. Proofs for Theorem 3 and 4 are in the supplementary material.

Theorem 3. *For symmetric $A \in \mathbb{F}_q^{n\times n}$ if (λ_i, v_i), for all $i \in [n]$ are the eigenpairs of A and $\widehat{v_i} = Tv_i + r$, with $r \in \left[-\frac{1}{2}, \frac{1}{2}\right]^n$, $\widehat{\lambda_i} = T\lambda_i + \rho$ for $\rho \in \left[-\frac{1}{2}, \frac{1}{2}\right]$, then (in $\mathbb{R}$)*

$$|TA\widehat{v_i} - \widehat{\lambda_i}\widehat{v_i}| \le \left\lceil Tn\|A\|_F + \frac{T\sqrt{n}}{2} + \frac{n}{4}\right\rceil$$
$$|\widehat{v_i}^T\widehat{v_j} - T^2\delta_{ij}| \le \left\lceil T\sqrt{n} + \frac{n}{4}\right\rceil$$

These bounds show how the error scales with the rounding factor T. To better interpret this, we next show that this error can be made arbitrarily small by increasing T.

Theorem 4. *If we have*

$$|TA\widehat{v_i} - \widehat{\lambda_i}\widehat{v_i}| \le \left\lceil Tn\|A\|_F + \frac{T\sqrt{n}}{2} + \frac{n}{4}\right\rceil$$
$$|\widehat{v_i}^T\widehat{v_j} - T^2\delta_{ij}| \le \left\lceil T\sqrt{n} + \frac{n}{4}\right\rceil$$

Then we have (in $\mathbb{R}$)

$$\frac{|T\lambda_i - \widehat{\lambda_i}|}{T} \le \frac{2n\sqrt{n}\|A\|_F}{T} + \frac{n}{T} + \frac{n\sqrt{n}}{2T^2} := E_{A,n}(T)$$

and $E_{A,n}(T) \to 0$ as $T \to \infty$

This means that if we want to find eigenvalues within certain error, we simply have to check that the bounds of Theorem 4 hold, using a T^* satisfying As $\|A\|_F = \Omega(1)$, this expression is $O(\frac{n^{3/2}\|A\|_F}{T})$. Hence it suffices to pick $T = O(n^{3/2}\|A\|_F/\epsilon)$.

Theorem 5. *There is an annotated steaming $\left(n^2 \log\left(qn^{\frac{3}{2}}\|A\|_F/\epsilon\right), \log\left(qn^{\frac{3}{2}}\|A\|_F/\epsilon\right)\right)$ –protocol for finding the eigenvalues of a symmetric matrix $A \in \mathbb{F}_q^{n\times n}$ to a precision of $\epsilon > 0$.*

Proof. The protocol is relatively straightforward. The verifier can compute $\|A\|_F$ and a fingerprint of A as the input is received. The helper provides the claimed matrix of (approximate) eigenvectors V and corresponding eigenvalues D We then make use of the matrix multiplication protocol to compute $(TA - D)V$. If the eigenvectors were exact, this would be $\mathbf{0}$; since they are approximate, we allow each entry in the result matrix to be at most $\lceil Tn\|A\|_F + \frac{1}{2}T\sqrt{n} + \frac{1}{4}n\rceil$ (computed over the reals and rounded to the finite field), and can verify that each entry of the product satisfies this bound. We also need to check that the eigenvectors are (almost) orthogonal, that is that each entry of $(V^TV - T^2I)$ is at most $\lceil T\sqrt{n} + \frac{1}{4}n\rceil$. Thus we need two invocations of the matrix multiplication protocol, evaluated over a sufficiently large finite field. Setting T according to the above bounds gives the claimed costs. $\square$

Once again, we consider the consequences of our input being $A \in \mathcal{F}_{\rho,M}^{n\times n}$, and choose the value of q as follows

Corollary 2. *Given $A \in \mathcal{F}_{\rho,M}^{n\times n}$ and desired precision ϵ, our matrices V and D are in $\mathcal{F}_{2\rho',nM^2}^{n\times n}$, where $\rho' = \max\{\rho, -\log\epsilon\}$. Therefore, to keep a direct correspondence between verification done in the field, and our real values, we choose $q > n^2(2M+1)^2 2^{2\rho'}$.*

This value of q is determined by the matrix multiplication step, and the fact that the largest possible eigenvalue of A is bounded by $n||A||_{\max}$, where $||A||_{\max}$ is M.

3.4 Matrix Inversion

For matrix inversion we again scale the field by a factor T. Over this expanded field, given an $n \times n$ matrix A, we want to find a matrix B such that $AB = TI$. To allow for rounding, we can relax the requirement of exact equality, and instead seek a matrix B that acts approximately like an inverse. We first show that such a matrix is guaranteed to exist.

Lemma 2. *For an invertible $A \in \mathbb{F}_q^{n\times n}$, we can find a matrix $B \in \mathbb{F}_{qT}^{n\times n}$ satisfying*

$$||AB - TI||_{\max} \le T\epsilon \tag{3}$$

if $T \ge \frac{n^2||A||_{\max}}{2\epsilon}$ and $\epsilon > 0$

Proof. Consider the true inverse of A, A^{-1}, computed over the reals. Then we can find $B = TA^{-1} + E$ so that $B \in \mathbb{Z}^{n\times n}$ and $\forall i,j.|E_{i,j}| \le \frac{1}{2}$. We can then map the entries of B directly into the field $\mathbb{F}_{Tq}$. We then have

$$\|AB - TI\|_{\max} \le \|AE\|_2 \le \tfrac{1}{2}n^2\|A\|_{\max}$$

That is, the error is at most $\frac{n^2\|A\|_{\max}}{2T}$. We can set the parameter T to be as large as needed to make this error value some small ϵ at quite modest cost: the field values are represented using $O(\log n + \log\|A\|_{\max} + \log 1/\epsilon)$ bits. $\square$

Theorem 6. *There is a streaming interactive $\left(n^2\log(n||A||_{\max}q/\epsilon), \log(n||A||_{\max}q/\epsilon)\right)$-protocol to invert an invertible matrix $A \in \mathbb{F}_q^{n\times n}$ with the above criteria,* (3).

Proof. The protocol is based on the above Lemma: we require the helper to provide such a matrix B under the extended field $\mathbb{F}_{Tq}$, along with the claimed value of AB. We then run the above protocol for matrix multiplication, and check that each entry of AB meets the required size bound. This ensures that the required condition on entries is met. The cost of storing the fingerprints is $O(\log(n\|A\|_{\max}q/\epsilon))$, and the communication costs come from sending B over $\mathbb{F}_{Tq}$, which is $O(n^2\log(n\|A\|_{\max}q/\epsilon))$. $\square$

In the case that the matrix is singular, the helper could demonstrate this by showing that there is an eigenvalue of A that is 0 to our precision.

Ensuring that everything is representable within finite precision is a little more involved, since A^{-1} can contain very large values, depending on the condition number $\kappa(A) = ||A^{-1}||_2 \cdot ||A||_2$. Since $||A||_2 \geq ||A||_{\max}$, we see that

$$\kappa(A) = ||A||_2||A^{-1}||_2 \geq ||A||_{\max}||A^{-1}||_{\max}$$

So, given a bound on $\kappa(A)$, we choose a field large enough to represent $\frac{\kappa(A)}{||A||_{\max}} \geq ||A^{-1}||_{\max}$.

3.5 Cholesky Decomposition

If we have a positive semi-definite matrix $A \in \mathbb{F}_q^{n\times n}$, the Cholesky Decomposition of A involves finding a lower triangular matrix $L \in \mathbb{R}^{n\times n}$, with $A = LL^T$. As before, we seek an approximate answer, $\widehat{L} \in \mathbb{F}_{qT}^{n\times n}$.

Theorem 7. *Given a positive semi-definite matrix $A \in \mathbb{F}_q^{n\times n}$, and $T \geq \frac{2n\|A\|_F}{\epsilon}$ there is an annotated steaming $(n^2 \log(qn\|A\|_F/\epsilon), \log(qn\|A\|_F/\epsilon))$-protocol to find $\hat{L} \in \mathbb{F}_{qT}^{n\times n}$ satisfying*

$$||\hat{L}\hat{L}^T - T^2A||_{\max} \leq T^2\epsilon$$

Proof. The protocol has the helper provide $\hat{L} \in \mathbb{R}^{n\times n}$ in order, and the helper executes the matrix multiplication protocol. It is straightforward to check that $\hat{L}$ is lower triangular as it is presented. For a $\widehat{L} = TL + E$ with $E_{i,j} \in \left[-\frac{1}{2}, \frac{1}{2}\right]$, we have

$$\begin{aligned}\|\widehat{L}\widehat{L}^T - T^2A\|_{\max} &\leq \|TLE^T + TEL^T + EE^T\|_2 \\ &\leq 2T\|L\|_2\|E\|_2 + \|E\|_2^2 \\ &\leq Tn\|A\|_F + \tfrac{n^2}{4}\end{aligned}$$

So we have an error ϵ at most $\frac{n\|A\|_F}{T} + \frac{n^2}{4T^2}$. Picking $T \geq 2n\|A\|_F/\epsilon$ suffices for any given ϵ (since $\|A\|_F \geq 1$). Via the matrix multiplication protocol, we just check $||\hat{L}\hat{L}^T - T^2A||_{\max} \leq T^2\epsilon$. The resulting protocol has communication cost $n^2\log(\frac{qn\|A\|_F}{\epsilon})$ and memory cost of $\log(\frac{qn\|A\|_F}{\epsilon})$, as claimed. □

3.6 Symmetric Generalised Eigenvalues

The Cholesky Decomposition allows us to solve the symmetric generalised eigenvalue problem for $A, B \in \mathbb{F}_q^{n\times n}$, with A symmetric, and B symmetric positive semi-definite:

$$\text{Find } V, D \in \mathbb{R}^{n\times n} \text{ such that } AV = BVD$$

We do this by finding the Cholesky Decomposition of B, L and then finding the eigenvalues of the symmetric matrix $C = L^{-1}A(L^{-1})^T$ to get matrices V', D' with $CV' = V'D'$. The solutions we want are $D = D'$, and $V = L^{-1}V'$.

Theorem 8. *There is a streaming annotated $(n^2\log(qT)$, $\log(qT))$-protocol with $T = q^2n^4/\epsilon$ for verifying $\hat{V}, \hat{D} \in \mathbb{F}_{qT}^{n\times n}$ approximately solving the generalised eigenvalue problem for $A \in \mathbb{F}_q^{n\times n}$ symmetric, and $B \in \mathbb{F}_q^{n\times n}$ symmetric positive semi-definite:*

$$AV = BVD$$

with ϵ the maximum absolute error between $\hat{D}$ and D.

The proof of Theorem 8 can be found in the supplementary materials.

4 Statistical Analysis

With the primitives outlined above, we can construct protocols for several common statistical analysis tasks.

4.1 Ordinary Least Squares

We first consider ordinary least squares regression (OLS), where we aim to find the optimal linear relationship between a data set X and a set of observations y.

Definition 4. *Let $\mathcal{S} = \langle\{y_1, X_1\}, ..., \{y_n, X_n\}\rangle$ where $y_i \in \mathbb{F}_q$ is an observation of the set of d predictors $X_i \in \mathbb{F}_q^d$. If we define $X \in \mathbb{F}_q^{n\times d+1}$ with the i'th row being $[1 X_i]$, OLS involves finding the linear relationship $y = X\beta + \epsilon$ with $\epsilon \in \mathbb{R}^n$ and $\beta \in \mathbb{R}^{d+1}$ minimizing $\sum_{i=1}^n \epsilon_i^2$.*

The OLS problem is solved with the Moore-Penrose pseudo inverse, $\left(X^TX\right)^{-1}X^T$. The optimal β is then $\left(X^TX\right)^{-1}X^Ty$. This leads directly to a protocol where the helper provides a β which is claimed to be optimal

Theorem 9. *There is an annotated streaming $\left(\max\{(d+1)^2, (d+1)n\}\log(q), \log(q)\right)$-protocol for OLS for $X \in \mathbb{F}_q^{n\times d+1}$, $y \in \mathbb{F}_q^n$ as above.*

Proof. The algorithm requires two applications of our matrix multiplication protocol to check that the β provided satisfies $(X^TX)\beta = X^Ty$. This avoids explicitly inverting X^TX. Via Theorem 1, the costs for X^Ty are $((d+1)n, 1)$ and for $(X^TX)\beta$ are $\left((d+1)^2, 1\right)$. Our total communication and space costs for OLS will therefore be $O\left(\max\{(d+1)^2, (d+1)n\}\log(q)\right)$ and $O(\log(q))$ respectively. As X^TX, X^T and y are in $\mathbb{F}_q$, we can find a $\beta \in \mathbb{F}_q$ and perform an exact equality check. □

4.2 Principal Component Analysis

Given a large data matrix $S \in \mathbb{F}_q^{n\times d}$ where S_{ij} represents the jth observation of the ith variable, PCA finds the principal components of the data, which can be used for dimensionality reduction or classification. PCA maximises the variation captured by successive vectors in $\mathbb{R}^n$, i.e. var $\left(v_1^TS\right) \geq$ var $\left(v_2^TS\right) \geq ... \geq$ var $\left(v_n^TS\right)$, with each $\lambda_i =$ var $\left(v_i^TS\right)$ maximised with respect to λ_j, $\forall j < i$. These v_i are the principal components and we can perform dimensionality reduction on S by choosing the k columns of $V = [v_1, ..., v_n]$ corresponding to the k largest λ_i and forming $V^T_{1\dots k}S = S' \in \mathbb{F}_q^{n\times k}$. This is equivalent to finding vectors corresponding to approximate eigenvalues of the covariance matrix of S.

Definition 5. *For a data set $S \in \mathbb{F}^{n\times d}$, the **Covariance Matrix** of S is*

$$\mathrm{Cov}(S)_{ij} = \frac{1}{n-1}\sum_{k=1}^{d}\left(S_{ik} - \mathbb{E}[S_i^{\downarrow}]\right)\left(S_{jk} - \mathbb{E}[S_j^{\downarrow}]\right)$$

Using the above protocols to check matrix multiplication and approximate eigenvectors (Sections 3.2 and 3.3), we can check PCA results to any desired precision $\epsilon > 0$ with the costs stated in the following theorem.

Theorem 10. *Given $S \in \mathbb{F}_q^{n\times d}$ and $\epsilon > 0$, there is a $(d^2\log(qnd\|S\|_F/\epsilon),\ \log(qnd\|S\|_F/\epsilon))$-protocol for verifying that PCA has been done to the desired precision.*

Proof. We have a primitive for producing S^TS whilst streaming S, and we can adapt this to generate the covariance matrix of S (scaled by $n-1$ to be in the field). This method can be found in the supplementary material and is a $(d^2\log(qn),\ \log(qn))-$protocol. This algorithm allows us to have a $(d^2\log(qnd\|S\|_F/\epsilon), \log(qnd\|S\|_F/\epsilon))$-protocol for PCA.

We use the approximate eigenvalue check (Section 3.3) with scaling factor $T = O(n^{3/2}\|S\|_F^2/\epsilon)$ to ensure that V has the necessary properties, i.e. is almost orthogonal, and each claimed eigenvector acts on the covariance matrix as an approximate stretch. Each principal component v_i corresponds to $\mathrm{var}(\hat{v}_i^TS) = \hat{\lambda}_i$ and therefore $|\mathrm{var}(\hat{v}_i^TS) - T\lambda_i| \le T\epsilon$, allowing us to reduce the dimensionality of S, confident of how much variance we are removing with each principal component. Soundness and correctness then follow from the invoked protocols. □

4.3 Fisher Linear Discriminant Analysis

In Linear Discriminant Analysis (LDA), we again have a large data matrix $S \in \mathbb{F}_q^{n\times d}$, with S_{ij} representing the jth observation of the ith variable, however, we also have a classification for each observation, ω_m for $m \in [k]$, where we have k classes. The aim is to do dimensionality reduction, as in PCA, while maximizing the class discriminatory information between the classes in the reduced dimension.

If we have k classes, we wish to transform S to a new set $S' \in \mathbb{F}_q^{n\times k-1}$, i.e. $S' = W^TS$ where $W \in \mathbb{F}_q^{d\times k-1}$ is a matrix projecting S to S' and S' has the maximum Fisher linear discriminant $J(W) = \frac{\det(W^TS_BW)}{\det(W^TS_WW)}$ where we have

Within-class scatter $S_W = \sum_{i=1}^{k}\mathrm{Cov}(S_i)$

Between-class scatter $S_B = \sum_{i=1}^{k} n_i(\mu_i - \mu)(\mu_i - \mu)^T$

We first treat the two-class case, then generalize to k classes.

4.3.1 Two Classes

To find $w \in \mathbb{F}_q^d$ we split S into two matrices, $S_1 \in \mathbb{F}_q^{n_1\times d}$ holding the observations in class 1 with average $\mu_1 \in \mathbb{F}_q^{n_1}$, and $S_2 \in \mathbb{F}_q^{n_2\times d}$ for the observations in class 2 with average $\mu_2 \in \mathbb{F}_q^{n_2}$. Then our two scatter matrices are:

$$S_W = \mathrm{Cov}(S_1)+\mathrm{Cov}(S_2) \text{ and } S_B = (\mu_1-\mu_2)(\mu_1-\mu_2)^T$$

To ensure that we can easily represent elements in the finite field, we actually find the (scaled-up) matrices $(n_1n_2)S_W$ and $(n_1n_2)S_B$. We wish to maximise the function $J(w) = \frac{w^TS_Bw}{w^TS_Ww}$. As such, these rescalings do not affect the result. Using the KKT conditions, we can shift the optimisation problem above to solving $S_Bw = \lambda S_Ww$, for some λ.

Theorem 11. *There is a $(dn\log(nq), \log(nq))$ −protocol for verifying LDA with 2 classes on $S \in \mathbb{F}_q^{n\times d}$.*

Proof. We can simplify $S_Bw = \lambda S_Ww$ for 2 classes, since for all vectors $v \in \mathbb{R}^d$;

$$S_Bv = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^Tv$$

As $\alpha = (\mu_1 - \mu_2)^Tv$ is just a scalar, S_Bv points in the direction of $(\mu_1 - \mu_2)$. Hence we get that $S_Bw = \lambda S_Ww$ is equivalent to $\alpha(\mu_1 - \mu_2) = \lambda S_Ww$. To verify we have been sent the correct w, all we need do is check that $S_Ww = (\mu_1 - \mu_2)$, and our result will lie in the (scaled-up) field, since all the scalars and vectors involved do, and so we can check this with a simple implementation of matrix multiplication protocol and covariance (Gramian) fingerprinting (Sections 3.1 and 3.2). The helper just has to send S, making the communication cost $O(nd\log(qn))$. The memory space is $O(\log(qn))$, where we have $\log(qn)$ due to the scaling of the scatter matrices. Completeness and Soundness follow from Theorem 1. □

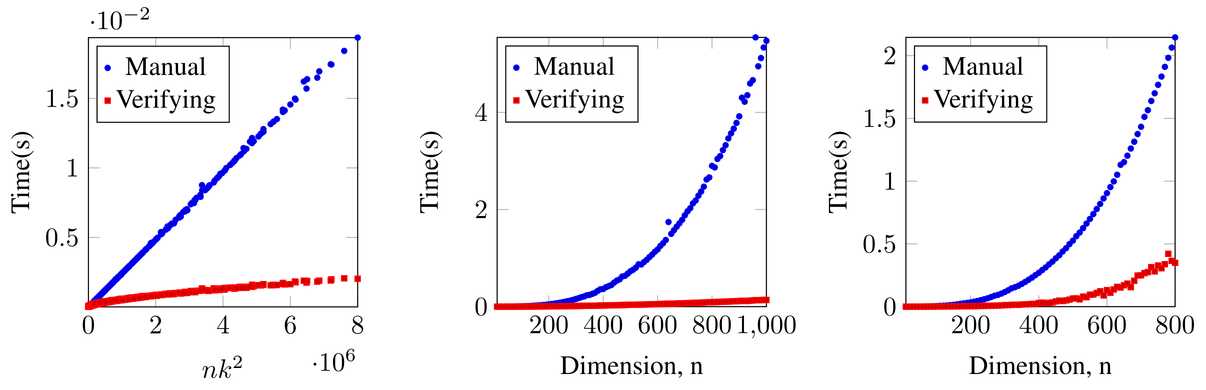
4.3.2 k Classes

With k classes, we now need to find a matrix $W \in \mathbb{F}_q^{d\times k-1}$. We still want to maximise the ratio between the between-class and within-class scatter, $J(W)$. The simplification to an eigenvalue problem is more complex here, but from Devijver and Kittler (1982) we see it reduces to finding (at most) $k-1$ eigenvectors with non-zero real eigenvalues of the matrix $S_W^{-1}S_B$.

Consider $A, B \in \mathbb{F}_q^{n\times n}$, with A symmetric, and B symmetric positive semi-definite (psd) the task is to find $V, D \in \mathbb{R}^{n\times n}$ such that $AV = BVD$ where $A = S_B$ and $B = S_W$. S_W is psd since it is the sum of psd matrices, so we can apply the Symmetric Generalised Eigenvalue Method.

Theorem 12. *There is a $(dn\log(qd/\epsilon),\ \log(qd/\epsilon))$-protocol for verifying LDA with k classes on $S \in \mathbb{F}_q^{n\times d}$.*

Proof. We stream in and fingerprint S, which then allows us to receive in the data class by class to build up



(a) Matrix Multiplication of $A \in \mathbb{F}_q^{k \times n}$ and $B \in \mathbb{F}_q^{n \times k}$ and 500 runs per test, for n and k from 10 to 200.

(b) Eigen-decomposition of symmetric $A \in \mathbb{F}_q^{n \times n}$ with $\epsilon = 0.01$ and 150 runs per test.

(c) Matrix Inversion of $A \in \mathbb{F}_q^{n \times n}$ where $\epsilon = \frac{1}{2}$ and 100 runs per test.

Figure 1: Experimental Results, with time taken for self computation (blue dots), and for verification (red squares).

the scatter matrices, with this we then use the method in Section 3.6 to find $\hat{V}$ and $\hat{D}$ with error ϵ. This is a $\big(d^2 \log(qd/\epsilon), \log(qd/\epsilon)\big)$-protocol, note however that we are only interested in columns of $\hat{V}$ corresponding to non-zero eigenvalues. We can use these get our $r \leq k - 1$ vectors to form W and compute $W^T S$ using the matrix multiplication protocol to get the desired result. Our total protocol will therefore be $(dn \log(nqd/\epsilon), \log(nqd/\epsilon))$ from constructing our scatter matrices. $\square$

5 Practical Results

We performed a series of experiments with a field size of $q = 2^{31} - 1$, using implementations in C of our protocols for matrix multiplication, inversion, and eigen-decomposition. We used a machine with an Intel i7 processor and 16 GB of memory. We worked on matrices up to 1000 by 1000, as this allowed us to get a distinguishing result between the verifier running the computation and the verifier using our checking protocols. We used the GNU Scientfic Library for C, using *gsl_blas_dgemm* for matrix multiplication, *gsl_linalg_LU_invert* to find the inverse and *gsl_eigen_symm* for the eigen-decomposition. The verification was done on the result produced by the self-computation. We created dense random matrices with entries drawn uniformly. Note that our protocols do not depend on any structure in the matrices, so uniform random data suffices to test them. Where we require symmetric matrices, we set $A_{ij} = A_{ji}\ \forall i < j$.

Our verification protocols were built as discussed in the paper, with the time taken to find the fingerprints of all relevant matrices and check the necessary bounds being the verification time seen in Figure 1. We tested the verification protocols against a variety of incorrect matrices, which were successfully rejected. The memory costs for our verification, even in the case of $A \in \mathbb{F}_q^{1000 \times 1000}$ was only a few hundred bytes, a significant reduction over self computation, which would be in the order of several megabytes. The communication cost is the cost of sending the matrix, simply the size of the matrix multiplied by a small constant. Consequently, the main aim of our experiments is to quantify the potential timing gains of verification over self-computation.

For our matrix multiplication protocol (Figure 1a), we consider the number of scalar multiplications required to find the result of multiplying $A \in \mathbb{F}_q^{k \times n}$ and $B \in \mathbb{F}_q^{n \times k}$, that is, $k^2 n$. When we plot the time taken against $k^2 n$, we see the expected linear relationship between self-computation and time taken. The verifier needs to compute the fingerprint of A, B, which will involve nk multiplications, and AB, which involves k^2 multiplications. This agrees with the observation in the graph; that the self-computation time is linear, and the verification time is sublinear.

In both matrix inversion (Figure 1b) and eigen-decomposition (Figure 1c), we see the expected cubic increases in running time for self-computation, with a significantly lower increase for the verification. The verification running time depends primarily on the $O(n^2)$ cost of computing the needed fingerprints, as above, and as such, we see the asymptotically shallower lower curve.

6 Concluding Remarks

We have introduced effective protocols for checking models generated by common tasks, such as Ordinary Least Squares, Principal Components Analysis and Linear Discriminant Analysis. Natural next steps are to study more complex models, such as Support Vector Machines, and (initially shallow but ultimately deep) neural networks, which have high training costs but could be cheap to verify.

Acknowledgements. This work is supported by European Research Council grant ERC-2014-CoG 647557.

References

László Babai. Trading group theory for randomness. In *Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pages 421–429. ACM, 1985.

Amit Chakrabarti, Graham Cormode, and Andrew Mcgregor. Annotations in data streams. *Automata, Languages and Programming*, pages 222–234, 2009.

Graham Cormode, Justin Thaler, and Ke Yi. Verifying computations with streaming interactive proofs. *Proceedings of the VLDB Endowment*, 5(1):25–36, 2011.

Graham Cormode, Michael Mitzenmacher, and Justin Thaler. Practical verified computation with streaming interactive proofs. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 90–112. ACM, 2012.

Graham Cormode, Michael Mitzenmacher, and Justin Thaler. Streaming graph computations with a helpful advisor. *Algorithmica*, 65(2):409–442, 2013.

Samira Daruki, Justin Thaler, and Suresh Venkatasubramanian. Streaming verification in data analysis. In *International Symposium on Algorithms and Computation*, pages 715–726. Springer, 2015.

Pierre A Devijver and Josef Kittler. *Pattern recognition: A statistical approach*. Prentice hall, 1982.

Rūsiņš Freivalds. Fast probabilistic algorithms. *Mathematical Foundations of Computer Science 1979*, pages 57–69, 1979.

Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 59–68. ACM, 1986.

Shafi Goldwasser, Yael Tauman Kalai, and Guy N Rothblum. Delegating computation: interactive proofs for muggles. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 113–122. ACM, 2008.

Johan Håstad and Avi Wigderson. The randomized communication complexity of set disjointness. *Theory of Computing*, 3(1):211–219, 2007.

Adi Shamir. IP=PSPACE. *Journal of the ACM (JACM)*, 39 (4):869–877, 1992.