Theoretician's Toolkit **Spring 2019**

Lecture 1: Dimension Argument

Scribe: Jonathan Liu 2/4/2019

In this lecture, we explore how arguments about dimension and linear independence of matrices can be used to prove results not only in random matrix theory but also in polynomials, error correcting codes, and hash functions. Later in the reading group, we will use similar ideas in Spectral Graph Theory.

1.1 Random Matrices

Theorem 1.1. *Let $\boldsymbol{M}$ be a random $n \times n$ matrix over $GF(2)$. Then* $\Pr[\det \boldsymbol{M} \neq 0] \geq 1/4$.

Proof. As stated earlier, the matrix's determinant is not zero when its rank is n, which is when its columns are all linearly independent. Note that for any set of linearly independent vectors $v_1, \ldots, v_m$, the probability that a randomly chosen vector in $\{0,1\}^n$ is linearly dependent to the set of vectors is $\frac{|\text{span}(v_1,\ldots,v_m)|}{|\{0,1\}^n|} = \frac{2^m}{2^n} = 2^{m-n}$.

With this in mind, note that the probability that the vectors are independent can be written as

$$\prod_{i=1}^{n} \Pr[\text{first } i-1 \text{ columns are linearly independent} | \text{first } i \text{ columns are linearly independent}],$$

and evaluation gives us

$$\begin{aligned}
&= \prod_{i=1}^{n} 1 - 2^{(i-1)-n} \\
&\geq \prod_{i=1}^{n} 4^{-2^{i-1-n}} \\
&= 4^{-\left(\sum_{i=1}^{n} 2^{i-1-n}\right)} \\
&\geq 4^{-\left(2^{-n}+\ldots+2^{-1}\right)} \\
&\geq 4^{-1} = \frac{1}{4}
\end{aligned}$$

□

Note that in the first step, we used the fact that $1 - x \geq 4^{-x}$, which is true in the case where $x \in \left[0, \frac{1}{2}\right]$.

Corollary 1.2. *Suppose M is a random matrix over $GF(2)$ and $\vec{b}$ is a random $n \times 1$ vector. Then the probability that $Mx = \vec{b}$ has a solution is at least $\frac{1}{4}$.*

Proof. By Theorem 1.6, M is linearly independent with probability at least 1/4. When it is, its span must contain $\vec{b}$. □

We now move away from linear algebra to look at how linear algebra arguments can be used to prove ideas in other familiar topics in mathematics.

1.2 Polynomials

Polynomials are important in both math and CS, but it's difficult to see exactly how linear algebra techniques can relate. We will see that when we represent polynomials in vector form, we can still use the dimension argument in our arguments

1.2.1 An Unexpected Argument for the Limit on the Roots of a Polynomial

We begin by introducing an interesting matrix, and use its properties to derive a fundamental idea about polynomials: that a polynomial can not have more roots than its degree.

Lemma 1.3. *The Vandermonde matrix, defined as a matrix of the form*

$$V = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-1} \end{bmatrix}$$

has determinant

$$\det(V) = \prod_{i<j}(x_j - x_i).$$

Proof. See Appendix 1.4.1. □

Theorem 1.4. *If a polynomial of degree d has more than d distinct roots, it is the zero vector.*

Proof. Denote the polynomial $p = \sum_{i=0}^{d} c_i x^i$. Arbitrarily select $d+1$ of the roots of p as $r_1, ..., r_{d+1}$. Then the system

$$\begin{bmatrix} 1 & r_1 & r_1^2 & \cdots & r_1^d \\ 1 & r_2 & r_2^2 & \cdots & r_2^d \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & r_{d+1} & r_{d+1}^2 & \cdots & r_{d+1}^d \end{bmatrix} \begin{bmatrix} c_d \\ c_{d-1} \\ \vdots \\ c_0 \end{bmatrix} = \vec{0}$$

must hold true. Notice that the matrix on the left is a Vandermonde matrix and the roots are unique, so by Lemma 1.3 it has positive determinant. Thus, the matrix is linearly independent. This implies that the equation can only hold if the coefficient vector is the zero vector, in which case the polynomial is the zero polynomial. □

1.2.2 Information Dispersal

Polynomials are often useful for sending information because of the ability to extrapolate a polynomial from any combination of d of its points. We will begin by proving that each polynomial can be described with finite amounts of information.

Theorem 1.5. *For any set of pairs $\{(x_1, y_1), \ldots, (x_{d+1}, y_{d+1}\}$ there exists a unique polynomial p of degree at most d such that $p(x_i) = y_i$ for all i.*

Proof. First, we prove existence by construction: the polynomial

$$p(x) = \sum_{i=1}^{d+1} y_i \frac{\prod_{i \neq j}(x - x_j)}{\prod_{i \neq j}(x_i - x_j)}$$

must take the values necessary (note that the fraction on the right side evaluates to 1 at $x = x_i$), and it has degree d because it is the sum of d-degree polynomials.

Next, we show uniqueness. Assume that $s(x)$ and $t(x)$ are d-degree polynomials that satisfy the property. Then for every $x \in \{x_1, \ldots, x_{d+1}\}$, we can see that $s(x)$ and $t(x)$ take the same value. Therefore, $r(x) := s(x) - t(x)$ is a d-degree polynomial with $d + 1$ roots, so by Theorem 1.4 it must be the zero polynomial. In that case, $s(x) = t(x)$. □

From here, we will show how to transmit information via polynomials across untrusted networks that can corrupt or destroy information.

First, we will work with a network that can destroy up to half of the packets sent through it. For any polynomial on the real numbers, there exist an infinite number of (x, y) pairs where $p(x) = y$. Thus, to send a d-degree polynomial, we can simply send $2d + 2$ unique pairs. At least $d + 1$ points are preserved, which we know from Theorem 1.5 is enough to uniquely determine the polynomial.

The more interesting case is when the network corrupts packets, which means that it changes the values in some unknown subset of the packets. In particular, we examine a network that corrupts up to $1/4$ of packets sent through it.

To counteract this, we simply send $4d$ packets of points on the polynomial. We must demonstrate that we can always efficiently retrieve the original polynomial from this set of packets, $1/4$ of which are potentially corrupted. To do so, we will use the observation that because at most d of the packets are corrupted, there exists a polynomial of degree at most d that has zeroes at every corrupted point.

Lemma 1.6. *Denote by $(x_1, y_1), \ldots, (x_{4d}, y_{4d})$ a set of $4d$ points, of which at least $3d$ lie on a polynomial p of degree d. Then there exist polynomials $e(x)$ and $c(x)$ such that*

$$deg\ (e) \leq d$$
$$deg\ (c) \leq 2d$$

and $c(x_i) = y_i e(x_i)$ for all i.

Proof. Because there are at most d points not on the polynomial, we can find $e(x)$ with degree at most d containing zeroes on those corresponding x-values, and select $c(x)$ such that $c(x) = p(x)e(x)$. In this case, $\deg(c) \leq 2d$, and for each packet we have

$$c(x_i) = e(x_i)p(x_i) = \begin{cases} e(x_i)y_i & \text{uncorrupted } x_i \\ 0 \cdot p(x_i) & \text{corrupted } x_i, \end{cases}$$

as desired. □

For any i, the equation $c(x_i) = y_i e(x_i)$ has $3d + 2$ unknowns for the coefficients of c and e. With the $4d$ packets sent, we can set up a system of $4d$ equations with $3d + 2$ coefficients, of which we can find a solution in polynomial time.

There are many possibilities for e and c, so we must also ensure that they always allow us to retrieve p.

Theorem 1.7 (Berlekamp-Welch 1985)**.** *Any polynomials $e(x)$ and $c(x)$ that follow the properties in Lemma 1.6 must follow the property that $p(x) = c(x)/e(x)$.*

Proof. For any c and e, the polynomial $r(x) = c(x) - e(x)p(x)$ is of degree at most 2d, but is zero at each of the $3d$ uncorrupted points. Thus, it must be the zero polynomial, so $c(x) = e(x)p(x)$. □

1.3 Hashing

Hashing schemes are a natural use for the dimension argument, given that they generally send sets to sets of different cardinality. When we design hash functions, we should want to design them in a way that minimizes collisions regardless of input. In particular, for any set of inputs, the hash function should limit the number of times that it sends different inputs to the same output.

Definition 1.8 (Pairwise independence)**.** *A family of hash functions with range $GF(p)$ is **pairwise independent** if for any hash function h in the family,*

$$(\forall x \neq y)\ (\forall u, v)\ \Pr_h[h(x) = u, h(y) = v] = \frac{1}{p^2}.$$

Example 1.9. If we look at the family of hash functions denoted by $h(x) = ax + b$ for some $a, b \in GF(p)$, we can see that for any x, y, u, v the event $h(x) = u, h(y) = v$ occurs only in the case where the following system of equations holds:

$$ax + b = u$$
$$ay + b = v.$$

Solving for a and b, we find that the left hand side is linearly independent when $x \neq y$, so there is a unique pair (a', b') that satisfies the equation. Therefore, these equations are satisfied iff the randomly selected hash function is exactly $h(x) = a'x + b'$, which is $\frac{1}{p^2}$.

Example 1.10. Suppose we want to determine whether a set of numbers contains two identical terms. We can do this by checking all $\binom{n}{2}$ pairs of elements, but we'd like to be able to do it in linear time. To do so, we select a pairwise-independent hash function on $GF(p)$ for $p \approx 2n$, and hash each element with it.

There are p buckets, in each of which we will need to compare elements pairwise to determine whether or not they are different. We will fix one of the buckets u, and use the indicator variable X_i for the event that $h(i) = u$. Then if we define S to be the number of items sent to bucket u, the number of comparisons will be $O(|S|^2)$. Note that $S = \sum X_i$, so

$$\begin{aligned} E[S^2] &= \sum_{i,j} E[X_i X_j] \\ &= \sum_i E[X_i] + \sum_{i \neq j} E[X_i X_j] \\ &= \sum_i (1/p) + \sum_{i \neq j} (1/p^2) \\ &= \frac{n}{p} + \frac{n^2}{p^2} \\ &\approx \frac{1}{2} + \frac{1}{4} = O(1), \end{aligned}$$

which means that across all buckets the expected number of comparisons will be $O(n)$, as desired.

1.3.1 Amplification the Success Probability of a Monte Carlo Algorithm

While the ability to generate pairwise independent variables with such a straightforward hash function is great, it is not immediately clear whether this condition is strong enough. After all, the condition is weaker than mutual independence, which we should expect of any random generators. With this in mind, we will show that pairwise independence is "random enough" for probability amplification of *Monte Carlo algorithms*.

Definition 1.11 (Monte Carlo algorithm)**.** *A **Monte Carlo algorithm** is a randomized algorithm for determining membership in a language where for any input, the output may be incorrect with small probability.*

Let's assume that we have a Monte Carlo algorithm A for a language L with the following correctness and soundness properties:

- Correctness: For all $x \in L$, $\Pr[A \text{ outputs "yes"}] \geq 1/2$.
- Soundness: For all $x \notin L$, $\Pr[A \text{ outputs "no"}] = 1$.

As we can see, if we try any $x \in L$, the error probability after t runs is $\leq 1/2^t$. Now, assume that A requires b random bits. If we want to bound the error probability of A by some probability $1/q$ where $q \leq 2^m$, we can simply run the algorithm $\log m$ times, in which case we need at least $b \log m$ bits of randomness. We would like to show that pairwise independent variables use less inherent "randomness," but will still provide the same bounds on the probability.

Theorem 1.12 (Chor/Goldreich 89)**.** *For $q \leq 2^m$, we can achieve an error bound of $1/q$ with $2m$ random bits.*

Proof. We select q pairwise independent samples from $\{0,1\}^m$, which we can do with our hash family or any other sampling method. Let X_i be the indicator for outputting "yes" on the ith sample, so an error occurs

whenever for an input in the language when $\sum_i X_i = 0$. We denote $X = \sum_i X_i$, and turn to Chebyshev's inequality:

$$\Pr[X = 0] \leq \Pr[|X - E[X]| \geq E[X]] \leq \frac{\mathrm{Var}(X)}{(E[X])^2}.$$

The variance for X is given by

$$\mathrm{Var}(X) = \sum_i \mathrm{Var}(X)_i + \sum_{i \neq j} \mathrm{Cov}(X_i, X_j).$$

Each outcome is pairwise independent from other outcomes, so each covariance is 0. Furthermore, the variance of each indicator variable is simply $p(1-p)$. Then

$$\frac{\mathrm{Var}(X)}{(E[X])^2} = \frac{qp(1-p)}{(qp)^2} = \frac{1-p}{p}\frac{1}{q} \leq \frac{1}{q}.$$

Therefore, the probability that $X = 0$ is less than $\frac{1}{q}$. □

1.4 Appendix

1.4.1 Proof of Lemma 1.3

Proof. Note that adding a column to another column of the matrix and adding a row to another row of the matrix are both operations that do not affect the determinant of the matrix. If we subtract the top row of the matrix from the rest of the matrix, we get

$$\begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 0 & x_2 - x_1 & x_2^2 - x_1^2 & \cdots & x_2^{n-1} - x_1^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & x_n - x_1 & x_n^2 - x_1^2 & \cdots & x_n^{n-1} - x_1^{n-1} \end{bmatrix}$$

At this point, note that the determinant of V is equal to the determinant of the cofactor matrix of the top left entry, because it is a column with all zeroes outside of the 1. Thus, we are looking for the determinant of the matrix

$$\begin{bmatrix} x_2 - x_1 & x_2^2 - x_1^2 & \cdots & x_2^{n-1} - x_1^{n-1} \\ x_3 - x_1 & x_3^2 - x_1^2 & \cdots & x_3^{n-1} - x_1^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ x_n - x_1 & x_n^2 - x_1^2 & \cdots & x_n^{n-1} - x_1^{n-1} \end{bmatrix},$$

which we can factor into

$$\begin{bmatrix} x_2 - x_1 & (x_2 - x_1)(x_2 + x_1) & \cdots & (x_2 - x_1)(\sum_{i=0}^{n-2} x_1^i x_2^{n-2-i}) \\ x_3 - x_1 & (x_3 - x_1)(x_3 + x_1) & \cdots & (x_3 - x_1)(\sum_{i=0}^{n-2} x_1^i x_3^{n-2-i}) \\ \vdots & \vdots & \ddots & \vdots \\ x_n - x_1 & (x_n - x_1)(x_n + x_1) & \cdots & (x_n - x_1)(\sum_{i=0}^{n-2} x_1^i x_n^{n-2-i}) \end{bmatrix}.$$

Multiplying a row of the matrix by a scalar multiplies the determinant by that same scalar, so we can see that the determinant of this matrix is equal to

$$\left(\prod_{i=2}^{n}(x_n - x_1)\right) \det\left(\begin{bmatrix} 1 & x_2 + x_1 & x_2^2 + x_1x_2 + x_1^2 & \cdots & \sum_{i=0}^{n-2} x_1^i x_2^{n-2-i} \\ 1 & x_3 + x_1 & x_3^2 + x_1x_3 + x_1^2 & \cdots & \sum_{i=0}^{n-2} x_1^i x_3^{n-2-i} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n + x_1 & x_n^2 + x_1x_n + x_1^2 & \cdots & \sum_{i=0}^{n-2} x_1^i x_n^{n-2-i} \end{bmatrix}\right).$$

Now, notice that if we denote column i as of the matrix V_i, then $V_i = x_1 V_{i-1} + x_r^i$ where r is the row of the entry. Thus, if we take each column, multiply it by x_1, and subtract it from the column to the right of it, the matrix becomes

$$\begin{bmatrix} 1 & x_2 + x_1 - x_1 & x_2^2 + x_1x_2 + x_1^2 - (x_1x_2 + x_1^2) & \cdots & \sum_{i=0}^{n-2} x_1^i x_2^{n-2-i} - \sum_{i=1}^{n-2} x_1^i x_2^{n-2-i} \\ 1 & x_3 + x_1 - x_1 & x_3^2 + x_1x_3 + x_1^2 - (x_1x_3 + x_1^2) & \cdots & \sum_{i=0}^{n-2} x_1^i x_3^{n-2-i} - \sum_{i=1}^{n-2} x_1^i x_2^{n-2-i} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n + x_1 - x_1 & x_n^2 + x_1x_n + x_1^2 - (x_1x_n + x_1^2) & \cdots & \sum_{i=0}^{n-2} x_1^i x_n^{n-2-i} - \sum_{i=1}^{n-2} x_1^i x_2^{n-2-i} \end{bmatrix},$$

which simplifies very nicely to

$$\begin{bmatrix} 1 & x_2 & x_2^2 & \cdots & x_2^{n-2} \\ 1 & x_3 & x_3^2 & \cdots & x_3^{n-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-2} \end{bmatrix}$$

As a result, we can see that

$$\det(V) = \left(\prod_{i=2}^{n}(x_n - x_1)\right) \det\left(\begin{bmatrix} 1 & x_2 & x_2^2 & \cdots & x_2^{n-2} \\ 1 & x_3 & x_3^2 & \cdots & x_3^{n-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-2} \end{bmatrix}\right).$$

The matrix on the right is a Vandermonde matrix! By an inductive argument, we can see that

$$\det(V) = \left(\prod_{i=2}^{n}(x_n - x_1)\right)\left(\prod_{i=3}^{n}(x_n - x_2)\right)\cdots\left(\prod_{i=n}^{n}(x_n - x_{n-1})\right),$$

which simplifies to $\det(V) = \prod_{j>i}(x_j - x_i)$ as required. □