Lecture 1: Decoding Polynomial Codes

Topics in Pseudorandomness and Complexity Theory (Spring 2017)
Rutgers University
Swastik Kopparty
Scribe: Aditya Potukuchi

1 Overview

Our main goal (for now) is the following:

Goal: Given a function $f : \{0,1\}^n \to \{0,1\}$ such that for every boolean circuit C of size S,

$$\Pr_x[C(x) = f(x)] < 1$$

i.e., the function is hard (in the *worst case*) for circuits of size S, and any $\epsilon > 0$, we want to construct a function $f' : \{0,1\}^n \to \{0,1\}$ such that for every circuit C' of size $O\left(\frac{S}{poly(n,1/\epsilon)}\right)$, we have:

$$\Pr_x[C(x) = f(x)] \leq \frac{1}{2} + \epsilon$$

i.e., the function is *very* hard (hard in the *average case*) for circuits of size $O\left(\frac{S}{poly(n,1/\epsilon)}\right)$.

We have seen such constructions in the previous lectures. However, the range of parameters we are now interested in, dictates that we make a few modifications to the previous construction, which involves lots (more!) of Error Correcting Codes.

We eventually want to construct functions f' such that for every circuit C' of size less than $2^{\delta n}$, we have:

$$\Pr_x[f'(x) = C'(x)] \leq \frac{1}{2} + 2^{-\epsilon n}$$

for some constants $\epsilon, \delta > 0$.

2 The Hadamard Code

(Recall) The Hadamard code is a linear code given by the encoding map $H : \{0,1\}^m \to \{0,1\}^{2^m}$ where $H(x) = (\langle x, y \rangle)_{y \in \{0,1\}^m}$ where the inner product $\langle x, y \rangle$ is taken mod 2. Note that this is exactly the polynomial code from the previous lecture, over $\mathbb{F}_2$, with degree 1. For convenience, it

is often helpful to identify $\{0,1\}^n$ with $\mathbb{F}_2^n$, and talk about H (also) as a linear map from $\mathbb{F}_2^m$ to $\mathbb{F}_2^{2^m}$.

This is a code with very low rate, and is typically useful in settings where this can be used as an 'inner code' along with more efficient 'outer codes' (we shall see an example of this in the next lecture). However this code has very large distance $\delta = \frac{1}{2}$.

Claim 1. *For any two $x, x' \in \{0,1\}^m$, $\Delta(H(x), H(x')) = 2^{m-1}$, where $\Delta(u,v)$ denotes the hamming distance between u, and v.*

Proof. Notice that $\Delta(H(x), H(x')) = |H(x \oplus x')|$, where $|w|$ denotes the *weight* of a vector, or the number of nonero coordinates. Since $x \neq x'$, we have that $x \oplus x'$ is nonzero. Therefore, it is enough to show that for a nonzero r, and y chosen uniformly from $\mathbb{F}_2^n$,

$$\Pr[\langle r, y \rangle = 0] = \frac{1}{2}$$

Indeed, let r_i, the i'th coordinate be nonzero in r. Then, for every $y \in \mathbb{F}_2^n$, exactly one of $\langle r, y \rangle$ and $\langle r, y + e_i \rangle$ is zero. □

3 The Goldreich-Levin Theorem and Local List Decodability

Since the Hadamard code has distance $\frac{1}{2}$, it can be uniquely decoded from less than $\frac{1}{4}$ fraction of errors using the polynomial decoding algorithm. However here we see that something much more interesting is true. The Goldreich-Levin theorem says that one can decode the Hamming code *locally*, i.e., with only a few queries to the bits of the received string, without looking at it completely. Moreover, one can decode from $\frac{1}{2} - \epsilon$ fraction of errors. Of course, this means that the decoded message is no longer unique. However, there are at most $poly(\frac{1}{\epsilon})$ such messages, and this algorithm outputs all of them.

[Remark: It is not hard to see that there are at most $\frac{1}{\epsilon^2}$ such messages, for example, via. fourier analysis. The main observation is that for a function $f : \mathbb{F}_2^n \to \{-1,1\}^n$, (we look at f both as a function, and its evaluation vector depending on the context. Also, we identify $\mathbb{F}_2$ with $\{-1,1\}$ in the usual way.) $\Delta(H(y), f) \leq \frac{1}{2} - \epsilon$ is equivalent to the condition $\hat{f}(y) \geq \epsilon$, and by Parseval, the number of such y's is at most $\frac{1}{\epsilon^2}$. So what we are stating with this theorem is that there is not much wastage (upto polynomial factors) in the worst case.]

We now state the Goldreich-Levin Theorem:

Theorem 2 (Goldreich-Levin Theorem)**.** *For every $\delta > 0$, there is a randomized algorithm $\mathcal{A}$ such that for any function $g : \{0,1\}^m \to \{0,1\}^n$, if there is an $x \in \{0,1\}^m$ that satisfies*

$$\Pr_{y \sim \{0,1\}^m}[g(y) = \langle x, y \rangle] \geq \frac{1}{2} + \epsilon$$

the algorithm $\mathcal{A}$ makes $poly(m/\epsilon)$ queries to g, and outputs a list $L \subseteq \{0,1\}^n$ with the following properties:

1. $|L| \leq poly(1/\epsilon)$
2. $x \in L$

with probability at least $1 - \delta$

In the language of coding theory, the above theorem says the following. We can *locally list decode* the Hadamard code from $\frac{1}{2} - \epsilon$ fraction of errors. We shall eventually arrive at the main algorithm given in the theorem. We first start off with the relatively simple case when we are to decode from $\frac{1}{4} - \epsilon$ fraction errors. This is indeed a simpler task, as one can locally uniquely decode the code, as shown in the following simple algorithm.

3.1 Baby Goldreich-Levin

Here, we will solve the same problem if $\Pr_y[g(y) \neq \langle x, y \rangle] \leq \frac{1}{4} - \epsilon$. Choose r uniformly at random from $\{0,1\}^m$. We notice that both r, and $r + e_i$ are uniformly distributed in $\{0,1\}^m$. Therefore, we have, by union bound:

$$\Pr_r[g(r) \oplus g(r + e_i) = x_i] \geq \frac{1}{2} + 2\epsilon$$

So, we make t such random queries $r_1, r_2, \ldots, r_t$, and take the majority vote of the $g(r_j) \oplus g(r_j + e_i)$'s. We have, by Chernoff Bound:

$$\Pr_{r_1,\ldots,r_t}[\text{MAJ}((g(r_j) \oplus g(r_j + e_i))_{j \in [t]}) \neq x_i] \leq \exp(-2\epsilon^2 t)$$

For $t = \frac{\ln m}{\epsilon^2}$, this probability is upper bounded by $\frac{1}{m^2}$.

Next, repeat the same algorithm for every coordinate. The probability that anyone of them is incorrect is upper bounded, by union bound, by $\frac{1}{m}$.

Thus, making $\frac{m \ln m}{\epsilon^2}$ queries, we recovered the value of (the unique) x with probability at least $1 - \frac{1}{m}$. This gives us our algorithm for the simple case

Algorithm 1 Baby Goldreich-Levin

```
1: procedure BabyGL(g)
2:     Choose r_1, ..., r_t uniformly and independently from {0,1}^m
3:     for i ∈ [m] do
4:         x_i ← MAJ(g(r_j) ⊕ g(r_j + e_i))_{j∈[t]}
       return (x_1, ..., x_m)
```

Next, we will see an algorithm that will locally list decode from $\frac{1}{2} - \epsilon$ fraction errors. The catch is that the list size is a lot more than what is guaranteed by the theorem, but it will demonstrate a crucial idea behind the final algorithm.

3.2 Almost Goldreich-Levin

Towards this improvement of the previous algorithm, the main idea is as follows: For some $a_1, a_2, \ldots, a_t$ chosen uniformly from $\{0,1\}^m$, suppose we knew the values of $\langle x, a_i \rangle$ for every $i \in [l]$. Then, to compute $\langle x, r \rangle$, we simply return $g(r + a_i) \oplus \langle x, a_i \rangle$. What we have gained is that we need to only guess $\langle x, r + a_i \rangle$ correctly (whereas in BabyGL(), we needed to guess two values correctly). More precisely, for every r,

$$\Pr_a[\langle x, r \rangle \neq g(r + a) \oplus \langle x, a \rangle] \leq \frac{1}{2} - \epsilon$$

Therefore, by Chernoff bound,

$$\Pr_{a_1,\ldots,a_t}[\langle x, r \rangle \neq \mathrm{MAJ}((g(r + a_i) \oplus \langle x, a_i \rangle)_{i \in [t]})] \leq \exp(-2\epsilon^2 t)$$

And in particular,

$$\begin{aligned} &\mathbf{E}_{a_1,\ldots,a_t}[\Pr_r[\langle x, r \rangle \neq \mathrm{MAJ}((g(r + a_i) \oplus \langle x, a_i \rangle)_{i \in [t]})]] \\ &= \Pr_{r,a_1,\ldots,a_t}[\langle x, r \rangle \neq \mathrm{MAJ}((g(r + a_i) \oplus \langle x, a_i \rangle)_{i \in [t]})] \\ &\leq \exp(-2\epsilon^2 t) \end{aligned}$$

Therefore, we have, by Markov:

$$\begin{aligned} \Pr_{a_1,\ldots,a_t}[\Pr_r[\langle x, r \rangle \neq \mathrm{MAJ}((g(r + a_i) \oplus \langle x, a_i \rangle)_{i \in [t]})] > \frac{1}{8}] \\ < 8 \exp(-2\epsilon^2 t) \end{aligned}$$

Let $g'(r) := \mathrm{MAJ}((g(r + a_i) \oplus \langle x, a_i \rangle)_{i \in [t]})$ Therefore, if t is big enough (but still $O(\frac{1}{\epsilon^2})$), we have that with high probability,

$$\Pr_r[g'(r) \neq \langle x, r \rangle] \leq \frac{1}{8}$$

which puts us in range to use BabyGL() ($\frac{1}{8}$ was chosen as some arbitrary number less than $\frac{1}{4}$). The only piece that is missing at this point is that we have assumed we know what $\langle x, a_i \rangle$ is for $i \in [t]$. The fix to this is simple: repeat this for every possible guess $(b_i)_{i \in [t]} := (\langle x, a_i \rangle)_{i \in [t]}$. This is the reason we get a list of possible values of x.

The size of the list the the total number of possible guesses. If we assume $t = \Theta\left(\frac{1}{\epsilon^2}\right)$, the list size is $2^{\Theta\left(\frac{1}{\epsilon^2}\right)}$. While not optimal, it is still worth noting that this list already has constant size (we

Algorithm 2 Almost Goldreich-Levin

procedure ALMOSTGL(g)
  $List \leftarrow \emptyset$
  Choose $a_1, \ldots, a_t$ uniformly and independently from $\{0,1\}^m$
  for $(b_1, \ldots, b_t) \in \{0,1\}^t$ **do**
    Define $g'(r) := \mathrm{MAJ}((g(r + a_i) \oplus b_i)_{i \in [t]})$
    $List \leftarrow List \cup \{\text{BABYGL}(g')\}$
  return $List$

will not compute the other parameters for this algorithm in this section). Next, we will reduce the number of a_i's that we need to pick with another simple observation. This will give us the final algorithm.

The following subsection gives the necessary prerequisites (not already covered in this course) to analyze the main algorithm, and may be skipped if the reader is familiar with it.

Aside: Pairwise Independence and Chebyshev Inequality

Definition 3 (Pairwise Independence)**.** *A joint distribution on random variables $(X_1, \ldots, X_m)$ is said to be* pairwise independent *if the marginal restricted to any two variables is fully independent, i.e., for every distinct $i, j \in [m]$, $\Pr[X_i = v_i \wedge X_j = v_j] = \Pr[X_i = v_i] \Pr[X_j = v_j]$.*

Often, we will just say '$X_1, \ldots, X_m$ are pairwise independent' when we refer to an underlying joint distribution on them.

Proposition 4. *For pairwise independent real valued random variables $X_1, \ldots, X_m$, we have for any distinct $i, j \in [m]$, $\mathbf{E}[X_i X_j] = \mathbf{E}[X_i]\mathbf{E}[X_j]$.*

Let $X := \sum_{i \in [m]} X_i$ where all the X_i's are pairwise independent real valued random variables. Then we have:

$$\begin{aligned}
\mathbf{E}[X^2] &= \mathbf{E}[(\sum_{i \in [m]} X_i)^2] \\
&= \sum_{i \in [m]} \mathbf{E}[X_i^2] + \sum_{(i,j)|i \neq j} \mathbf{E}[X_i X_j] \\
&= \sum_{i \in [m]} \mathbf{E}[X_i^2] + \sum_{(i,j)|i \neq j} \mathbf{E}[X_i]\mathbf{E}[X_j] && \text{by proposition ??}
\end{aligned}$$

Therefore, we have $\mathrm{Var}(X) = \mathbf{E}[X^2] - \mathbf{E}^2[X] = \sum_{i \in [m]} (\mathbf{E}[X_i^2] - \mathbf{E}^2[X_i]) = \sum_{i \in [m]} \mathrm{Var}(X_i)$. Thus, the variance of the sum of pairwise independent random variables is just the sum of their variances.

Now that we have a way of getting a handle on variance, the following well known inequality is often useful to get concentration bounds around the means of such random variables.

Lemma 5 (Chebyshev Inequality)**.** *For a real valued random variable X with mean μ, we have:*

$$\Pr[|X - \mu| \geq d] \leq \frac{\mathrm{Var}(X)}{d^2}$$

Proof. We have $\Pr[|X-\mu| \geq d] = \Pr[(X-\mu)^2 \geq d^2]$. Since $(X-\mu)^2$ is always positive, by Markov, we have $\Pr[(X-\mu)^2 \geq d^2] \leq \frac{\mathbf{E}[(X-\mu)^2]}{d^2} = \frac{Var(X)}{d^2}$ □

We are now ready to study the main algorithm that achieves the guarantees claimed in theorem **??**.

3.3 Goldreich-Levin

In this section, we finally prove the Goldreich-Levin Theorem by exhibiting the randomized algorithm $\mathcal{A}$.

The main observation that leads to a reduced number of a_i's (and hence reduced list size), compared to AlmostGL() is the following: Once we have (correctly) guessed $\langle x, a_i \rangle$, and $\langle x, a_j \rangle$, we have also basically (correctly) guessed $\langle x, a_i + a_j \rangle$. In general, once we have guessed $\langle x, a_i \rangle$ for $i \in [t]$, we also have the values of $\langle x, a_S \rangle$, where $a_S = \sum_{i \in S} a_i$ for every $S \subseteq [t]$. So we can use these in the majority vote to compute g'. However, the problem is that the events $\{[\langle x, r \rangle \neq g(r + a_S) \oplus \langle x, a_S \rangle]\}_{S \subseteq [t]}$ are not all independent, and so we cannot use the Chernoff bound. However, *they are pairwise independent*, and it turn out that this is enough to give a good guarantee on the choice of the a_i's.

Claim 6. *For $S \subseteq [t]$, denote A_S to be the event $[\langle x, r \rangle \neq g(r + a_S) \oplus \langle x, a_S \rangle]$. Then for any $T \neq T'$, both nonempty, A_T, and $A_{T'}$ are independent.*

Proof. The proof is by observing that a_T, and $a_{T'}$ are independently distributed, i.e., for any $u, v \in \{0,1\}^n$, $\Pr[a_{T'} = u | a_T = v] = \frac{1}{2^m}$. Indeed, this is the case, let (w.l.o.g) $l \in T' \setminus T$ be the last set chosen. Then, we have that the event $[a_{T'} = v | a_T = u]$ depends only on l, and $\Pr[a_{T'} = v | a_T = u] = \frac{1}{2^m}$. □

For a fixed $r \in \{0,1\}^m$, let the random variable $X := \sum X_S$, where $X_S := \mathbb{1}_{\langle x,r \rangle = g(r + a_S) \oplus \langle x, a_S \rangle}$ for every nonempty $S \subseteq [t]$. We have $\mathbf{E}[X] \geq (2^t - 1)\left(\frac{1}{2} + \epsilon\right)$, and $\mathrm{Var}(X) = \sum_S \mathrm{Var}(X_S) \leq (2^t - 1)\left(\frac{1}{4} - \epsilon^2\right)$. Again, we compute the probability that the majority vote gives the wrong answer. By Cebyshev Inequality:

$$\Pr[X \leq 2^{t-1}] \leq \frac{1}{4\epsilon^2 2^t}$$

We carry out a similar computation as above:

$$\begin{aligned}\mathbf{E}_{a_1,\ldots,a_t}[\Pr_r[\langle x,r\rangle \neq \mathrm{MAJ}((g(r+a_S)\oplus\langle x,a_S\rangle)_{S\subseteq[t]})]] \\ = \Pr_{r,a_1,\ldots,a_t}[\langle x,r\rangle \neq \mathrm{MAJ}((g(r+a_S)\oplus\langle x,a_S\rangle)_{S\subseteq[t]})] \\ \leq \frac{1}{4\epsilon^2 2^t}\end{aligned}$$

Therefore, again, by Markov:

$$\begin{aligned}\Pr_{a_1,\ldots,a_t}[\Pr_r[\langle x,r\rangle \neq \mathrm{MAJ}((g(r+a_S)\oplus\langle x,a_S\rangle)_{S\subseteq[t]})] > \frac{1}{8}] \\ < \frac{2}{\epsilon^2 2^t}\end{aligned}$$

And we have a similar algorithm as above, except with a smaller list:

Algorithm 3 Goldreich-Levin

procedure GL(g)
  List $\leftarrow \emptyset$
  Choose $a_1,\ldots,a_t$ uniformly and independently from $\{0,1\}^m$
  for $(b_1,\ldots,b_t)\in\{0,1\}^t$ **do**
    Define $g'(r) := \mathrm{MAJ}((g(r+a_S)\oplus b_S)_{S\subseteq[t]})$
    List $\leftarrow$ *List* $\cup\{\textsc{BabyGL}(g')\}$
  return *List*

First, we compute the probability of outputting the correct answer. Recall that there are at most $\frac{1}{\epsilon^2}$ values of $(b_i)_{i\in[t]}$ which can potentially belong to the 'correct' x, and our a_i's must work for all of them. Also, BabyGL() must output correctly for all of these. So, the probability of error, by either of these process is at most (by union bound):

$$\frac{1}{\epsilon^2}\left(\frac{1}{m}+\frac{2}{\epsilon^2 2^t}\right)$$

For $t=\frac{1}{\epsilon^4}+\log(4/\delta)$, this value is at most δ.

The size of the list is equal to $2^t = O\left(\frac{1}{\epsilon^4}\right)$. The number of queries is equal to the number of queries made by all the calls to the BabyGL() algorithm, which is $O(2^t\cdot m\log m) = O\left(\frac{m\log m}{\epsilon^4}\right)$

This completes the proof of theorem **??**.

[Remark: In the analysis, we needed the fact that number of x's such that $\Pr_y[g(y)\neq\langle x,y\rangle]\leq\frac{1}{2}-\epsilon$ is at most $\frac{1}{\epsilon^2}$. Moreover, the list returned by the algorithm as is, is only guaranteed to be of size $O\left(\frac{1}{\epsilon^4}\right)$. This can easily be 'improved' by going through the list and testing if every point x in it does indeed satisfy the condition that $\Pr_y[g(y)\neq\langle x,y\rangle]\leq\frac{1}{2}-\epsilon$. This would take another poly $\left(\frac{1}{\epsilon^2}\right)$ queries, and in return, we get a list of the 'right' size. However, it is not really needed in our application, and so, we omit the details.]

4 Application in Hardness Amplification

As a corollary of the Goldreich-Levin theorem, we have the following lemma which provides another key step in our journey towards constructing functions that are very hard on average:

Lemma 7. *Let $f : \{0,1\}^n \to \{0,1\}^n$ be a function. Define $\tilde{f} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ given by*

$$\tilde{f}(x,y) = \langle f(x), y \rangle$$

Suppose there was a circuit $\tilde{C}$ of size S such that

$$\Pr_u[\tilde{C}(u) = \tilde{f}(u)] \geq \frac{1}{2} + \epsilon$$

Then there is a circuit $C : \{0,1\}^n \to \{0,1\}^n$ of size at most poly $\left(\frac{S}{\epsilon^2}\right)$ such that

$$\Pr_u[C(u) = f(u)] \geq poly(\epsilon)$$

Proof. We have:

$$\begin{aligned} & \mathbf{E}_x[\Pr_y[\tilde{C}(x,y) = \langle f(x), y \rangle]] \\ = & \Pr_{x,y \sim \{0,1\}^n}[\tilde{C}(x,y) = \tilde{f}(x,y)] \\ = & \Pr_{w \in \{0,1\}^{2n}}[\tilde{C}(u) = \tilde{f}(u)] \\ & \geq \frac{1}{2} + \epsilon \end{aligned}$$

Therefore,

$$\Pr_x[\Pr_y[\tilde{C}(x,y) = \langle f(x), y \rangle] \geq (1/2) + (\epsilon/2)] \geq \frac{\epsilon}{2}$$

Therefore, at at least $\frac{\epsilon}{2}$ fraction of points, we have enough agreement to use the Goldreich-Levin algorithm. However, the algorithm returns a list of possible outputs. Since this list is of size *poly* $\left(\frac{1}{\epsilon}\right)$, outputting one at random still gives a probability at least *poly* $\left(\frac{1}{\epsilon}\right)$ of being right. Further, it can be verified that all this computation can be done with a circuit of size *poly* $\left(\frac{S}{\epsilon}\right)$. Given this randomized circuit, we use the usual error reduction, and non uniformity to obtain a deterministic circuit with *poly* $\left(\frac{1}{\epsilon}\right)$ factor blowup in size.

□

[Remark/A bit of perspective as to where this fits in: Informally, we take a function $g : \{0,1\}^n \to \{0,1\}$ that is worst case hard. From g, we construct a function $f : \{0,1\}^n \to \{0,1\}^m$ (think of $m = \tilde{O}(n)$) that is much harder (as hard as given in the hypothesis of the above lemma). Finally, we construct a function $\tilde{f}$ as above to get a boolean function that is very hard on average.]

5 Decoding Reed-Solomon Codes

We continue along our journey to constructing functions that are hard on average. We are one step away (see the remark at the end of the previous section), and the main problem we will encounter in this step is the local list decoding of general low degree polynomial codes (we have done the case where the degree is 1). We will encounter the 'sub-task' of list decoding univariate polynomial codes. Here, we will start with uniquely decoding univariate polynomial (Reed-Solomon) Codes.

Decoding from errors in a Reed-Solomon Code, is stated equivalently, in the language of polynomial interpolation as follows:

Problem: Given (distinct) points $(x_1, y_1), \ldots, (x_n, y_n)$ in $\mathbb{F}_q^2$ such that there is a nonzero polynomial p of degree $d \leq n/2$ that passes through more than $(n+d)/2$ of them, find p.

Is is easy to see that there is at most one such nonzero polynomial. Otherwise, let p and q be nonzero polynomials with the above property. Then the polynomial $p - q$, is of degree at most d and has more than than $2(n+d)/2 - n = d$ zeroes, which is a contradiction.

5.1 Berlekamp-Welch Algorithm

The main idea behind the solution to the above problem is quite simple:

1. Try to fit a rational function $\frac{N(X)}{E(X)}$ to the given data.
2. Hope that $E(X)$ divides $N(X)$ (as a polynomial) and that $\frac{N(X)}{E(X)}$ is the required polynomial.

The algorithm is given as follows:

Algorithm 4 Berlekamp-Welch

procedure BW($\{(x_i, y_i)\}_{i \in [n]}$)
    Find polynomials $N(X)$ and $E(X)$ of degrees at most $d + e$,and e respectively such that $N(x_i) = E(x_i)y_i$ for all $i \in [n]$.
    $p(X) \leftarrow N(X)/E(X)$ **return** $p(X)$

5.2 Analysis of Berlekamp-Welch

First, we note that the first step in the algorithm is just solving a bunch of linear equations, ad the second is polynomial division over a finite field. Both these operations can be carried out efficiently.

Let p be the unique polynomial of degree at most d that fits the data. By assumption, p is nonzero. The algorithm returns nonzero polynomials $N(X)$ and $E(X)$ of degree $d+e$, and e respectively such that for every given tuple (x_i, y_i), we have $N(x_i) = E(x_i)y_i$. Here, e is chosen to be $< (n-d)/2$

Such an N and an E must exist. We can see this by choosing $E(X) = \prod_{p(x_i \neq y_i)}(X - x_i)$, and $N(X) = E(X) \cdot p(X)$ (for this reason, $E(X)$ is often called the *error locator polynomial*). The following claim proves the correctness of the algorithm

Claim 8. *For every $N(X)$, and $E(X)$ returned by the algorithm, $E(X)|N(X)$, and $p(X) = N(X)/E(X)$.*

Proof. Suppose not, consider the nonzero polynomial $N(X) - p(X)E(X)$. This polynomial has degree at most $d + e$. Therefore, it vanishes at at most $d + e$ points. However, from the way these polynomials were obtained, it vanishes whenever $y_i = p(x_i)$, so it vanishes more than $(n + d)/2$ points. Therefore, we have $(n + d)/2) < d + e < d + (n - d)/2$ which is a contradiction. □

The range of parameters that will be interesting for us is as follows: $d \leq 0.5n$, the number of points where the polynomial matches the given data is $0.99n$. It is easy to check that the above algorithm works.