Explanation of algorithm version a (doesn't work)

We have a mixture of r product distributions on n coordinates.

Let $p_j^{(i)}$ be the probability that the i-th product measure has a $+1$ in the coordinate j. If $x \in \{-1,1\}^n$, let μ_i be the measure of the i-th product

$$\mu_i(x) := \prod \left(\frac{1}{2} + \left(p_j^{(i)} - \frac{1}{2} \right) x_j \right)$$

and $M(x) = \sum w_i \mu_i(x)$.

If $\mu : \{-1,1\}^n \to [0,1]$ is a measure, we represent it by a polynomial

$$G_\mu(t_1, \ldots, t_n) := \sum_{x \in \{-1,1\}^n} \mu(x) \prod_{j=1}^{n} \left(\frac{1}{2} + \frac{1}{2} t_j x_j \right).$$

If you have a point $x \in \{-1,1\}^n$, then $G(x_1, \ldots, x_n) = \mu(x_1, \ldots, x_n)$.

Let $h_i := G_{\mu_i} = \prod_{j=1}^n \left(\frac{1}{2} + (p_j^{(i)} - \frac{1}{2}) t_j \right)$ be the representation of the i-th product distribution, and let $H := G_M = \sum_{i=1} w_i h_i$.

We take a large number of samples M from the mixture of product distributions. Call them $x^1, \ldots, x^M \in \{-1,1\}^n$. Let $G(t_1, \ldots, t_n) := \frac{1}{M} \sum_{m=1}^M \prod(\frac{1}{2} + \frac{1}{2} t_j x_j^m)$ be the representation of the empirical distribution.

We are given a coordinate j and parameters n, r. We are supposed to recover the set $\{p_j^{(i)}\}$ for one coordinate. The algorithm works conceptually as follows.

Step 1. Choose an integer $\rho \approx 2 + \lg r$. Pick ρ coordinates, one of which must be j. (Suppose $j = 1$ and the remaining coordinates are $2, \ldots, \rho$.)

Step 2. Let $\vec{G}(x_1, \ldots, x_\rho) := G(x_1, \ldots, x_\rho, t_{\rho+1}, \ldots, t_n)$. This is a vector in the $2^{n-\rho}$-dimensional space of polynomials in $t_{\rho+1}, \ldots, t_n$ which have got degree at most one in each coordinate.

Project that onto the $1 + n - \rho$-dimensional space of polynomials in $t_{\rho+1}, \ldots, t_n$ which have total degree at most one, by throwing out all the terms of degree > 1. That is, $1 \mapsto 1$, $t_j \mapsto e_j$, everything else goes to zero. We get a new list of vectors

$$\vec{g}(x_1, \ldots, x_\rho) \in \mathbb{C}[t_{\rho+1}, \ldots, t_n]/(\text{monomials of degree} > 1) \cong \mathbb{R}^{1+n-\rho}.$$

At this point, $\vec{g}(x_1, \ldots, x_\rho)$ is $1/M$ times the sum of $(1, x_{\rho+1}^m, \ldots, x_n^m)$ for each sample x^m where the bits $1, \ldots, \rho$ matched the bit string $x_1, \ldots, x_\rho$. So, basically, it's an average of the samples that started with $x_1, \ldots, x_\rho$.

(Note: In the real algorithm, we don't compute $\vec{G}$, only the projection $\vec{g}$. You might get somewhere by keeping monomials of higher degree, but the coefficients are noisy, and the signal-to-noise ratio goes up with the degree, I think.)

Step 3. Let $q = 0, 0.0001, \ldots, 1$. Set

$$\mathcal{O}_q(x_2, \ldots, x_\rho) = q \times \vec{g}(0, x_2, \ldots, x_\rho) - (1-q) \times \vec{g}(1, x_2, \ldots, x_\rho).$$

This gives us a list of $2^{\rho-1}$ vectors. We think of this as a matrix $\mathcal{O}(q)$.

Step 4. We *hope* that the rank of $\mathcal{O}(q)$ is r. Use SVD to get the r-th singular value of $\mathcal{O}(q)$, and call it $s_r(q)$. Find the local minima of $s_r(q)$, i.e. the values of q with $s_r(q - 0.0001) > s_r(q) < s_r(q + 0.0001)$. These numbers are the estimates for the values of $p_j^{(i)}$.

Here is the idea of the algorithm. We have

$$\vec{g}(x_1, \dots, x_\rho) \approx \sum_{i=1}^{r} w_i \left[\prod_{j=1}^{\rho} \left(\frac{1}{2} + \left(p_j^{(i)} - \frac{1}{2} \right) x_j \right) \right] \vec{h}^{(i)}$$

where $\vec{h}^{(i)}$ are projections of $h_i(x_1, \dots, x_\rho, t_{\rho+1}, \dots, t_n)$ onto the space of polynomials in $n - \rho$ variables with degree at most 1 as above, and we *hope* very much that $\operatorname{span} \vec{h}^{(i)}$ is as large as possible, r. This isn't true in general, but it is true generically. If it is true, then

$$\mathcal{O}_q(x_2, \dots, x_\rho) \approx \sum_{i=1}^{r} w_i (q - p_1^{(i)}) \left[\prod_{j=2}^{\rho} \left(\frac{1}{2} + \left(p_j^{(i)} - \frac{1}{2} \right) x_j \right) \right] \vec{h}^{(i)}.$$

The i-th term in this sum will be small when $q \approx p_1^{(i)}$.

If the span of $\vec{h}^{(i)}$ is r and the polynomials in brackets are not linearly dependent, then $\mathcal{O}_q$ will have rank r for most q. If q is close to the parameter $p_j^{(i)}$, then $\vec{h}^{(i)}$ will vanish, the approximate rank will drop, and the r-th singular value, which is the last "real" one that doesn't just come from noise, will reach a minimum.