

# Semi-Parametric Manifold Clustering

## Estimating Polynomial Curves

### Problem Setup

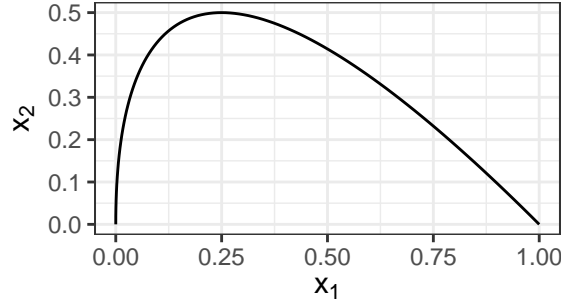
Let:

- $T_1, \dots, T_n \stackrel{\text{iid}}{\sim} F$  with support  $[0, 1]$ .
- $g(\cdot, \theta) : [0, 1] \mapsto \mathcal{X} \subset \mathbb{R}^d$ .
- $X_1, \dots, X_n = g(T_1), \dots, g(T_n)$

Assuming some parametric form of  $g$  with parameters  $\theta$ , we want to find  $\hat{\theta}$ , some “reasonable” estimate for  $\theta$ . We observe  $X_i$  but not  $T_i$ .

For now, we limit  $d = 2$  and  $g$  to quadratic functions.

**Example 1.** Let  $g(t) = (t^2, 2t(1-t)) = (0 + 0t + t^2, 0 + 2t - 2t^2)$ . (This is the first two dimensions of the Hardy-Weinberg curve). Then  $\theta = (0, 0, 1, 0, 2, -2)$ .



If we observe the  $T_i$ 's, then we can use a standard polynomial regression method to obtain  $\hat{\theta}$ . Since we do not observe them, the proposed iterative method is as follows:

1. Initialize  $\hat{\theta}^{(0)}$  (e.g., randomly).
2. Estimate each  $\hat{t}_i^{(s)}$  by minimizing  $L(t_i, \hat{\theta}^{(s)} | x_i) = L_i = \|x_i - g(t_i | \hat{\theta}^{(s)})\|^2$ .
3. Compute each  $\hat{x}_i^{(s)} = g(\hat{t}_i^{(s)} | \hat{\theta}^{(s)})$
4. Estimate  $\hat{\theta}^{(s+1)}$  by minimizing  $L(\{\hat{t}_i^{(s)}\}, \theta | X) = \sum_i \|x_i - g(\hat{t}_i^{(s)} | \theta)\|^2$ .
5. Repeat steps 2-4 until convergence.

If we restrict  $g$  to be polynomials, then steps (2) and (4) have closed-form solutions. Alternatively, we can estimate  $g$  using more general forms, e.g., splines, which may require approximation.

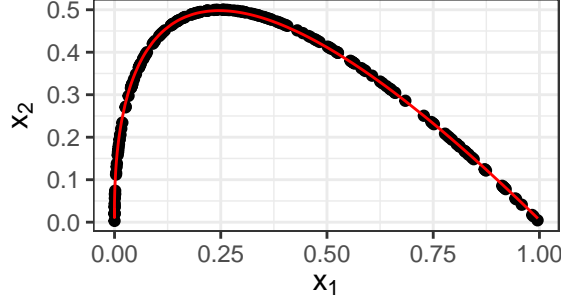
**Example 2.** Write  $g(t|\theta) = (g_1(t|\theta_1), \dots, g_d(t|\theta_d))$  where  $g_r(t|\theta_r)$  is the component of  $g$  in the  $r^{\text{th}}$  dimension and  $\theta_r$  is the vector of parameters for the  $r^{\text{th}}$  dimension. If  $g_r$  are polynomials of degree  $p$ , then each  $\theta_r$  contains up to  $p+1$  entries.

Given the observed points  $x_1, \dots, x_n \in \mathbb{R}^d$  and their corresponding index points  $t_1, \dots, t_n \in \mathbb{R}$ , we can find each  $\hat{\theta}_r$  individually by  $\hat{\theta}_r = A^{-1}b$  where  $b \in \mathbb{R}^{p+1}$  and  $b_k = \sum_i x_i t_i^k$  and  $A \in \mathbb{R}^{(p+1) \times (p+1)}$  and

$$A_{kl} = \sum_i t^{(k-1)(l-1)}.$$

On the other hand, if we have parameters  $\theta$  but not the index points  $t_i$ , we can minimize each  $t_i$  individually by finding the roots of a  $p + 1$  polynomial with coefficients that depend on  $x_1, \dots, x_n$  and  $\theta$ .

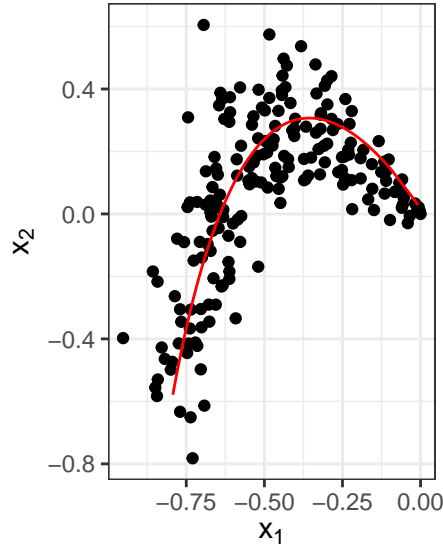
In the following plot, we drew  $n = 200$  points from the 2D H-W curve with  $T_1, \dots, T_n \stackrel{\text{iid}}{\sim} \text{Uniform}(0, 1)$ . The red line is the curve that was fit using the above method.



Note: the parameterization of the curve is not unique.

## Estimation with Noise

**Example 3.** In the next example, we draw  $A \sim \text{RDPG}(X)$  using the same H-W curve and sample size as above and estimate the true latent positions (up to rotation).



A modification to this that is possibly more robust is to use Bezier curves for  $g$ . This is the same functional form as the polynomial curves used before, but with orthogonal bases:

$$g(t|p) = \sum_{r=0}^R p_r \binom{R}{r} (1-t)^{R-r} t^r$$

where  $R$  is the order of the Bezier curve, each  $p_s$  is a vector of length  $d$ , and, as before,  $g : [0, 1] \mapsto \mathbb{R}^d$ . Thus, if we fit each  $p_r$ , then the procedure is the same as before.

The least squares estimate for  $p \in \mathbb{R}^{R \times d}$  is

$$\hat{p} = (T^\top T)^{-1} T^\top X$$

where  $T = \begin{bmatrix} t^{(1)} & \dots & t^{(R)} \end{bmatrix}$  and each  $t_i^{(r)} = \binom{R}{r} (1 - t_i)^{R-r} t_i^r$ , and  $X \in \mathbb{R}^{n \times d}$ . The same procedure for estimating  $t_1, \dots, t_n$  can be applied here.

The parameterization for a given curve is not unique. In particular, the above procedure will not necessarily provide  $t_1, \dots, t_n \in [0, 1]$ . One possible remedy for this is to, after estimating the  $t_i$ 's, normalize them to the unit interval. If we assume  $t_1, \dots, t_n \stackrel{\text{iid}}{\sim} \text{Uniform}(a, b)$ , then the UMVUE are

$$\hat{a} = \frac{nt_{(1)} - t_{(n)}}{n - 1}, \hat{b} = \frac{nt_{(n)} - t_{(1)}}{n - 1}$$

which yields the normalization transformation  $t \leftarrow (t - \hat{a})/(\hat{b} - \hat{a})$ .

Alternatively, we can force the  $t_i$ 's to be approximately uniform on the unit interval by the transformation  $t \leftarrow \hat{F}(t)$ , where  $\hat{F}$  is the empirical CDF of  $t_1, \dots, t_n$ .

For initialization, we can use a one-dimensional Isomap embedding to estimate the  $t_i$ 's and use that to estimate  $\hat{p}$ . Experiments suggest that if the data are well-behaved (i.e., it looks like the curve we are trying to fit), this results in much faster convergence.

## Theoretical Framework

If we assume that the points are distributed as  $X_i \stackrel{\text{ind}}{\sim} \mathcal{N}(f(t_i|p), \Sigma(t_i|\phi))$ , then we can write the (incomplete) log likelihood as:

$$\ell(p, \phi) = -\frac{1}{2} \sum_i |\Sigma(t_i|\phi)| - \frac{1}{2} \sum_i (x_i - f(t_i|p))^\top (\Sigma(t_i|\phi))^{-1} (x_i - f(t_i|p))$$

In the case  $\Sigma(t_i|\phi) = \sigma^2 I$  and  $f(t_i|p) = p^\top t_i$  where  $t_i \in \mathbb{R}^{R+1}$  is the Bezier polynomial expansion of order  $R$ , then this becomes

$$\ell(p, \sigma^2) = -\frac{nd}{2} \log \sigma^2 - \frac{1}{2\sigma^2} \|X - Tp\|_F^2$$

Then given  $t_1, \dots, t_n$ , the MLE of  $p$  is again  $(T^\top T)^{-1} T^\top X$ , and the MLE of  $\sigma^2$  is just  $\frac{1}{nd} \|X - T\hat{p}\|_F^2$ . And given  $p$ , we can estimate each  $t_i$  in the same way as before to maximize  $\ell$  ( $\sigma^2$  is unnecessary here).

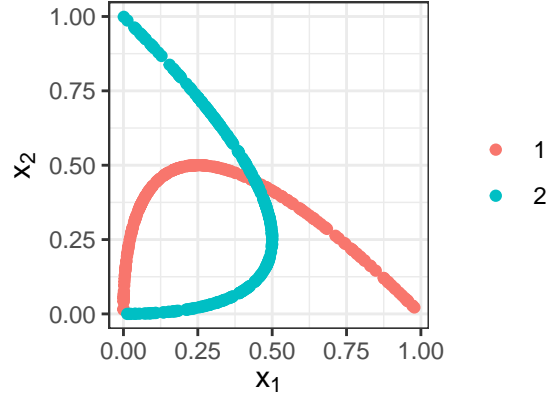
## Clustering

Next, suppose we have  $K$  curves parameterized by  $g^{(k)}$ , with points drawn along these curves. Then one possible clustering technique is as follows:

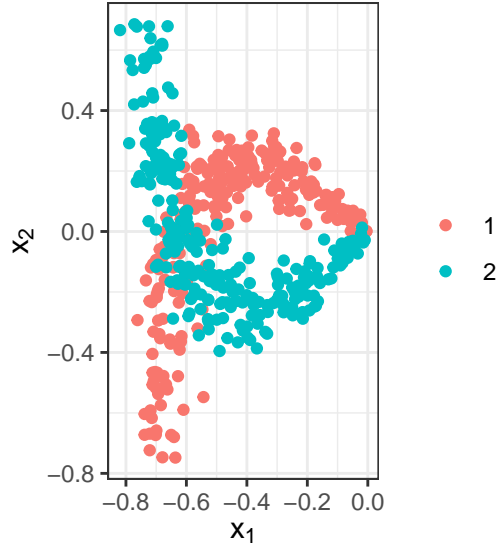
1. Assign an initial clustering (e.g., via spectral clustering).
2. Estimate the curve for each cluster (using the same curve-fitting procedure as before).

3. Reassign the clusters by proximity to each curve.
4. Repeat 2 and 3 until convergence.

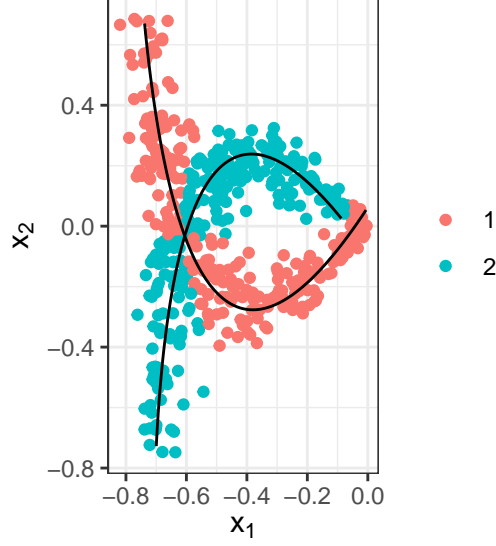
**Example 4.** We have two intersecting curves,  $g_1(t) = \begin{bmatrix} t^2 & 2t(1-t) \end{bmatrix}^\top$  and  $g_2(t) = \begin{bmatrix} 2t(1-t) & (1-t)^2 \end{bmatrix}^\top$ .  $n_1 = n_2 = 256$  points are drawn uniformly from each.



We draw  $A \sim \text{RDPG}(X)$  and obtain the following ASE:

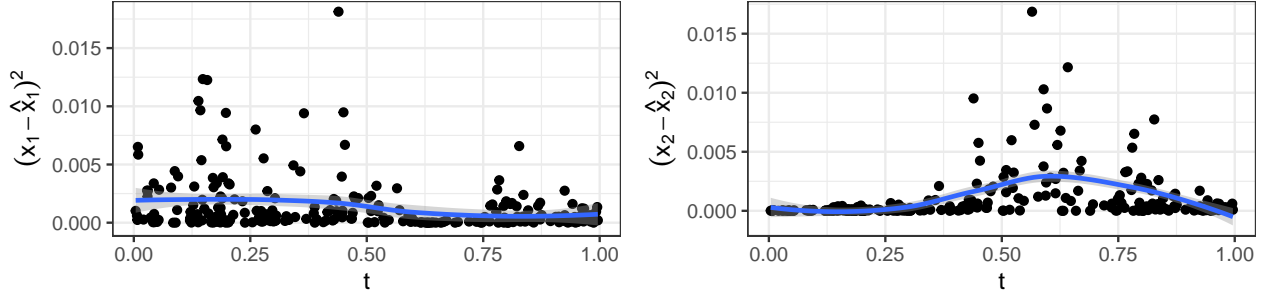


Fitting two quadratic Bezier curves to these data yields a community detection error rate of 10%. In the following plot, the points are labeled according to their estimated labels.



## Model-Based Clustering

The CLT property of the ASE may allow us to approximate  $Wx - \hat{x} \sim \mathcal{N}(f_k(t), \Sigma_k)$ . One possibly reasonable assumption for  $\Sigma_k$  is  $\Sigma_k = \Sigma_k(t) = \text{diag}(\sigma_{k1}^2(t), \dots, \sigma_{kd}^2(t))$ . A plot of the squared errors of cluster 1 in the previous example reveals the following:



Fitting some curve to each  $\sigma_{kr}^2(t)$  allows us to compute estimated probabilities of each point belonging to each cluster/manifold:  $\hat{q}_{ik} = \mathcal{N}(x_i | g_k(\hat{t}_i), \Sigma_k(\hat{t}_i))$ . Then we can use weighted least squares to estimate the parameters of  $g$ :

$$\hat{p}_k = (T^\top Q_k T)^{-1} T^\top Q_k X$$

where  $Q_k = \text{diag}(q_{1k}, \dots, q_{nk})$ .