Supplementary Material

A. Derivation of Eq. 5

We have samples $(\mathbf{x}^{(i)}, y^{(i)})$, with $\mathbf{x} \in \mathbb{R}^d$ and $y \in \{1, 2, \ldots, l\}$, and we would like to evaluate Eq. 4 or $\hat{H}(Y|X)$. We define this estimator in terms of existing entropy estimators, $\hat{H}(Y|X) \equiv \hat{H}(Y) + \hat{H}(X|Y) - \hat{H}(X)$. The definition of the differential entropy estimators is given by Eq. 1.

To write down the standard plug-in estimator for discrete entropy, we first define $n_j \equiv \sum_{i=1}^{N} \delta_{y^{(i)},j}$.

$$\hat{H}(Y) = -\sum_{j=1}^{l} n_j/N \log(n_j/N)$$

Next we should unpack the second term, using the definition that $\bar{\epsilon}_{i,k}$ denotes the distance to the k-th nearest neighbor to point $x^{(i)}$ that is in the same cluster.

$$\begin{aligned}
\hat{H}(X|Y) &= \sum_{j=1}^{l} p(y=j)\hat{H}(X|y=j) \\
&= \sum_{j=1}^{l} n_j/N \left(\log(n_j/k) + c_{k,n_j} + \frac{d}{n_j} \sum_{i=1}^{N} \delta_{y^{(i)},j} \log \bar{\epsilon}_{i,k} \right)
\end{aligned}$$

We have used Eq. 1 in the second line. Next we expand and perform the sum over j for the last term only, eliminating the delta function.

$$\begin{aligned}
&= \sum_{j=1}^{l} n_j/N \left(\log(n_j/N) + \log N/k + c_{k,n_j} \right) + \frac{d}{N} \sum_{i=1}^{N} \log \bar{\epsilon}_{i,k} \\
&= -\hat{H}(Y) + \log N/k + \sum_{j=1}^{l} c_{k,n_j} n_j/N + \frac{d}{N} \sum_{i=1}^{N} \log \bar{\epsilon}_{i,k}
\end{aligned}$$

Putting this all together and using the nearest neighbor for estimation, or $k = 1$.

$$\begin{aligned}
\hat{H}(Y|X) &= \hat{H}(Y) + \hat{H}(X|Y) - \hat{H}(X) \\
&= \log N + \sum_{j=1}^{l} c_{1,n_j} n_j/N + \frac{d}{N} \sum_{i=1}^{N} \log \bar{\epsilon}_{i,1} - \log N - c_{1,N} - \frac{d}{N} \sum_{i=1}^{N} \log \epsilon_{i,1} \\
&= \left(\sum_{j=1}^{l} c_{1,n_j} n_j/N - c_{1,N} \right) + \frac{d}{N} \sum_{i=1}^{N} \log \frac{\bar{\epsilon}_{i,1}}{\epsilon_{i,1}} \\
&\approx \frac{d}{N} \sum_{i=1}^{N} \log \frac{\bar{\epsilon}_{i,1}}{\epsilon_{i,1}}
\end{aligned}$$

The constant quickly goes to zero for reasonable values of N and we neglect it. Recalling the definition, $c_{k,N} = \psi(N) - \psi(k) + \log(2k/N)$. The constant term has the form $\sum_{j=1}^{l} (\psi(n_j) - \log(n_j)) n_j/N - (\psi(N) - \log(N))$. Using a well-known expansion for the digamma function, $\psi(n) - \log(n) = -1/(2n) + O(1/(2n)^2)$. Applying this first order expansion to our expression gives a single term $(1 - l)/N$.

There is an ambiguity in applying our entropy estimator in Eq. 1 to some subset of points defined by a cluster if there are only k or fewer points in a cluster. Then we define, $\bar{\epsilon}_{i,k} = \epsilon_{i,N-1}$ if $k > n_{y^{(i)}} + 1$. This definition reflects the fact that our uncertainty is maximal if we are not given sufficient data. For instance, when $k = 1$, that means we have a cluster that contains a single sample point. In that case, we estimate the entropy of this single point as maximal with respect to the full dataset. In principle, the maximum length scale could be set by other prior information, but we strive to make data-driven choices whenever possible. In practice, this choice penalizes very small clusters, and the details of the penalty makes little difference in the outcome.

B. Heuristic Optimization

The goal is to search for a natural coarse-graining, as defined in Eq. 7. The number of ways of partitioning N points into groups is very large and evaluating CV for any partitioning requires calculating all pairwise distances. Even calculating the change in CV from altering the cluster membership of a single point may require $O(N)$ operations. Finally, the figures in Sec. 4 suggest that our optimization landscape is very rugged, so gradient-based methods are unlikely to succeed. Because of these difficulties, we suggest a heuristic optimization below.

Our optimization proceeds first by generating a small number of candidate partitions that are likely to have small CV using a tractable semidefinite program. Then we rank the candidate partitions according to CVR.

$$\hat{H}_T(Y|X) = \int_0^1 d\alpha\, \hat{H}_\alpha(Y|X) = \frac{d}{N}\sum_{i=1}^{N}\sum_{k=1}^{N-1}\frac{1}{k(k+1)}\log\frac{\bar{\epsilon}_{i,k}}{\epsilon_{i,k}},$$

Clearly, the contribution from terms coming from k-th nearest neighbors quickly decreases with k. If we ignore $k \geq k_{max}$, then the objective is clearly minimized as long as the k-th nearest neighbors for point $x^{(i)}$ are all in the same cluster. We begin by relaxing our discrete cluster variable $y^{(i)}$ to be a continuous variable lying on a hypersphere, i.e., $\mathbf{y}^{(i)} \in \mathbb{R}^{d'}$ and $\mathbf{y}^{(i)} \cdot \mathbf{y}^{(i)} = 1$. If two points are close together in the $\mathbf{x}$ space, we want them to be close together in $\mathbf{y}$ space as well. We define a weighted adjacency matrix, $A_{i,j} = 1/(k(k+1))$ if j is the k-th nearest neighbor to i.

$$\min_{|\mathbf{y}^{(i)}|=1}\sum_{i,j}A_{i,j}(\mathbf{y}^{(i)}-\mathbf{y}^{(j)})^2 = \max_{|\mathbf{y}^{(i)}|=1}\sum_{i,j}A_{i,j}\mathbf{y}^{(i)}\cdot\mathbf{y}^{(j)}$$

The optimal value of this optimization is for all $\mathbf{y}^{(i)}$ to be equal. We need to add a term that forces all the $\mathbf{y}^{(i)}$ as far apart as possible.

$$\max_{|\mathbf{y}^{(i)}|=1}\sum_{i,j}(A_{i,j}-\beta)\mathbf{y}^{(i)}\cdot\mathbf{y}^{(j)}$$

We represent the Gram matrix with components $M_{ij} = \mathbf{y}^{(i)} \cdot \mathbf{y}^{(j)}$, and define $\bar{A}_{i,j} = A_{i,j} - \beta$. Then our optimization takes the form:

$$\max_{M \text{ is } p.s.d.} \mathrm{Tr}\bar{A} \bullet M, \tag{8}$$

where p.s.d. denotes that M is positive semidefinite, a necessary and sufficient condition for it to be a Gram matrix. This optimization is a semidefinite program and can be efficiently solved using convex optimization techniques. We used $k_{max} = 10$ and $\beta = 1/(k_{max}(k_{max}+1))$ and made no effort to optimize these parameters.

Once the optimal M has been found, a discrete clustering can be found using the rounding method of Goemans and Williamson (Goemans & Williamson, 1995). First, taking the Cholesky decomposition of M recovers the vectors $\mathbf{y}^{(i)}$. Now to recover a discrete partition (into two groups) from these vectors we pick a random unit vector $\mathbf{u}$, and partition the data according to $y^{(i)} \equiv \mathrm{sign}(\mathbf{u} \cdot \mathbf{y}^{(i)})$. We generated 200 candidate partitions this way.

Next, we calculate the CVR for each partition and pick the best (lowest) one. In the event we want multiple clusters, we combine the partition with the lowest CVR with each of the top 25 remaining candidate partitions and then we chose the one with the smallest overlap with the original partition (according to the Rand index). We continue this procedure until we have the desired number of clusters.

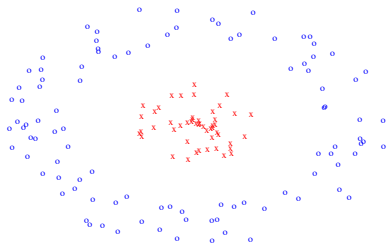

Figure 9. An example of a non-convex clustering produced by our heuristic optimizer for Eq. 7. The correct partition is found despite unbalanced cluster sizes.