Nonlinear Dimensionality Reduction by Semidefinite Programming and Kernel Matrix Factorization

Bibliography
Link

The idea is that the full kernel (inner product) matrix K can be accurately reconstructed from a much smaller submatrix of inner products between randomly chosen **landmarks**. We write:

$$K \approx QLQ^T,$$

where L is the $m \times m$ submatrix of inner products between landmarks (with $m \ll n$) and Q is an $n \times m$ linear transformation derived from solving a sparse set of linear equations.

We start by approximately reconstructing inputs $\{x_i\}_{i=1}^m$, denoting the landmark inputs $\{\mu_k\}_{k=1}^m$ (e.g., the first m inputs), as such:

$$\hat{x}_i = \sum_{k=1}^{m} Q_{ik}\mu_k$$

Here, Q is a linear transformation that we derive below. The key idea here is that we can use that same transformation to reconstruct the entire dataset in the lower-dimensional space. In particular, we assume that the transformation from a high to a low-dimensional space is *locally* isometric, so that linear reconstructions are okay. Therefore, after finding the embeddings for the landmarks $\{\ell_k\}_{k=1}^m$, we can reconstruct the entire dataset:

$$y_i = \sum_{k=1}^{m} Q_{ik}\ell_k$$

Sketch of the Algorithm

Step 1 - Compute reconstruction weights $W \in \mathbb{R}^{n\times n}$ that minimize the sum of squares of the reconstruction errors of each input from its r-nearest neighbors. We do that by solving the following optimization problem:

$$\begin{aligned} \min_{W} \quad & \sum_{i=1}^{n} \|x_i - \sum_{j} W_{ij}x_j\|_2^2 \\ \text{s.t.} \quad & \sum_{j} W_{ij} = 1, \quad i = 1, \dots, n \\ & W_{ij} = 0, \quad i \nsim j \end{aligned}$$

We constrain the reconstruction weights for each input to sum to one to ensure that the reconstruction weights are invariant to the choice of origin in the input space. We also constrain the reconstruction weights for x_i to be zero for all x_j that are not r-nearest neighbors of x_i.

Step 2 - Choose m landmarks and compute the transformation:

$$Q = \begin{pmatrix} \mathbb{I}_m \\ (\Phi^{uu})^{-1}\Phi^{ul} \end{pmatrix},$$

where $\Phi = (\mathbb{I}_n - W)^\top(\mathbb{I}_n - W)$ which we partition as:

$$\Phi = \begin{pmatrix} \overbrace{\Phi^{\ell\ell}}^{m} & \overbrace{\Phi^{\ell u}}^{n-m} \\ \Phi^{ul} & \Phi^{uu} \end{pmatrix}$$

Step 3 - Solve MVU for the landmark kernel matrix L and retrieve the low-dimensional embeddings for the landmarks $\{\ell\}_{k=1}^m \in \mathbb{R}^d$ from its eigenvalues:

$$\begin{aligned}\max_{L} \quad & \operatorname{tr}(QLQ^\top) \\ \text{s.t.} \quad & \mathbf{1}^\top (QLQ^\top)\mathbf{1} = 0 \\ & e_i^\top (QLQ^\top) e_i - 2 e_i^\top (QLQ^\top) e_j + e_j^\top (QLQ^\top) e_j = \|x_i - x_j\|_2^2, \quad i \sim j\end{aligned}$$

Step 4 - Reconstruct the outputs $y_i = \sum_{k=1}^m Q_{ik} \ell_k$.

Notes

- L-MVU changes the local isometry constraints to *inequalities*. Because the matrix factorization $K \approx QLQ^\top$ is only approximate, we must relax the constraints to preserve feasibility. This does not appear to change the solutions, since the variance maximization tends to saturate the pairwise distance constraints, even if they are not enforced as strict inequalities.
- MVU and L-MVU have the same number of constraints, and the constraints in the latter are not sparse, so a naive implementation can be much slower than MVU. In practice, the SDP for L-MVU is solved while only explicitly monitoring a small fraction of the original constraints.
  - We start by feeding an initial subset of constraints to the solver, consisting of the SDP constraint, the centering constraint, and the distance constraints between landmarks and their nearest neighbors.
  - If a solution is found that violates some of the unmonitored constraints, these are added to the problem, which is solved again. This process is repeated until all the constraints are satisfied.
  - This incremental scheme is made possible by the relaxation of the distance constraints from equalities to inequalities.

Part 1 - Derive a linear transformation Q for approximately reconstructing the entire dataset from m randomly chosen landmarks.

Denoting the (high dimensional) inputs $X = \{x_i\}_{i=1}^n$ and the landmarks $\{\mu_\alpha\}_{\alpha=1}^n$, the reconstructed inputs $\{\hat{x}_i\}_{i=1}^n$ are given by the linear transformation:

$$\hat{x}_i = \sum_\alpha Q_{i,\alpha} \mu_\alpha$$

(How we actually get Q is explained later.)

Note that the same linear transformation can be used to reconstruct the *unfolded* dataset, that is, after the mapping from inputs $\{x_i\}_{i=1}^n$ to outputs $\{y_i\}_{i=1}^n$. Denoting the *unfolded* landmarks by $\{\ell_\alpha\}_{\alpha=1}^m$ and the reconstructed outputs by $\{\hat{y}_i\}_{i=1}^n$ we argue that $y_i \approx \hat{y}_i$, where:

$$\hat{y} = \sum_\alpha Q_{i,\alpha} \ell_\alpha$$

Part 2 -

Part 3 - The kernel matrix factorization in the beginning follows if we make the approximation:

$$K_{i,j} = y_i^\top y_j \approx \hat{y}_i^\top \hat{y}_j,$$

so that the matrix L from the first equation is:

$$L_{\alpha,\beta} = \ell_\alpha^\top \ell_\beta$$

Reconstructing from Landmarks

We now derive the linear transformation Q.
(...) to a good approximation, we can hope to reconstruct each input by a weighted sum of its nearest neighbors. These reconstruction weights can be found by minimizing the error function:

$$\mathcal{E}(W) = \sum_i \| x_i - \sum_j W_{i,j}\, x_j \|^2$$

This is subject to the constraint that $\sum_j W_{i,j} = 1$, and where $W_{i,j} = 0$ if x_j is not a nearest neighbor of x_i. This constraint ensures that the reconstruction weights are invariant to the choice of the origin in the input space.

Choosing the first m inputs as landmarks, we ask whether it is possible to (approximately) reconstruct the remaining inputs given just the landmarks μ_α and the weights $W_{i,j}$.

We can rewrite the reconstruction error as a function of the inputs:

$$\mathcal{E}(X) = \sum_{i,j} \Phi_{i,j} x_i^\top x_j$$

(Dunno why or how we do this.) Here, $\Phi = (\mathbb{I} - W)^\top (\mathbb{I} - W)$. We can partition Φ into blocks distinguishing the m landmarks from the other (unknown) inputs:

$$\Phi = \begin{bmatrix} \Phi^{\ell\ell} & \Phi^{\ell u} \\ \Phi^{u\ell} & \Phi^{uu} \end{bmatrix},$$

where we use ℓ to index landmarks and u the other inputs.