逆核分解

李宬睿，吴安琪
计算科学与工程 School of Computational Science & Engineering
佐治亚理工学院 Georgia Institute of Technology
{cnlichengrui,anqiwu}@gatech.edu

2024 年 3 月 25 日

摘要

目前最先进的降维方法严重依赖于复杂的优化过程. 另一方面，仅通过特征分解的解析解（闭式解）往往没有充分的复杂性与非线性. 本文我们提出了一个新的基于样本协方差矩阵的特征分解的非线性降维方法—逆核分解 (Inverse Kernel Decomposition, IKD). 这个方法是启发自高斯过程隐变量模型 (Gaussian process latent variable models, GPLVM) 的，并且比传统的 GPLVMs 性能更好. 为了处理噪声很大且相关性很弱的情形，我们提出了两种方案—分块解和测地解—来利用局部相关的数据点以得到更好的且数值稳定的隐变量估计. 我们用合成数据集和四个真实世界数据集来展示，IKD 比其它基于特征分解的降维方法更好，并且能以更快的速度达到与基于优化的降维方法相当的效果. Python 实现的 IKD 开源在 `https://github.com/JerrySoybean/ikd`.

1 引言

降维问题在机器学习领域已经研究了很多年了，广泛地应用在隐变量估计 [Wu et al., 2017, 2018]、去噪 [Sheybani and Javidi, 2009]、聚类分析 [Bakrania et al., 2020]、数据可视化 [Van der Maaten and Hinton, 2008a] 等等. 最常用的是主成分分析 (PCA)，一个线性降维方法. 它好用是因为只需要做一步特征分解. 然而它的线性假设太简单，限制了适用性，尤其是高度非线性的情形. 另一方面，像自编码器 [Kramer, 1991]、变分自编码器 (VAE) [Kingma and Welling, 2013]、t-SNE [Van der Maaten and Hinton, 2008b]、UMAP [McInnes et al., 2020] 以及高斯过程隐变量模型 (GPLVMs) [Lawrence, 2003, 2005] 这些分线性降维方法能在发现极优低维隐变量上达到非常好的效果，使得之后的分析效果非常好（比如，可视化、预测、分类）. 然而，这些非线性方法涉及到复杂的优化，非常浪费时间且容易陷入局部最优，并且对初始条件非常敏感. 本文我们提出一种新的非线性的基于特征分解的降维方法，用以寻找复杂非线性且具有解析解的低维隐变量.

提出的方法称为逆核分解 (Inversed Kernel Decomposition, IKD)，启发自 GPLVMs. GPLVMs 是一个用于降维的概率生成模型. 其使用高斯过程 (GPs) 来找高维数据在低维的非线性嵌入. GPLVM 及其众多变种在很多领域都有涉猎 [Bui and Turner, 2015, Wang et al., 2005, 2008, Urtasun et al., 2006, Wu et al., 2017] 并且被认为是强有力的非线性降维方法和隐变量模型. 然而，由于其中的 GP 成分，

GPLVMs 是高度非线性、非凸的，这就导致优化时有比较现实的困难．而 IKD 通过推导 GPLVMs 中核函数的关系，能够用特征分解的方法解决 GPLVM 问题．相比于传统的基于优化的 GPLVM 求解方法，IKD 可以可以在更短时间内得到稳定的隐变量估计（见 Sec. 3.1.1）．

在实验部分，我们将 IKD 和四个基于特征分解的降维方法和四个基于优化的降维方法在一个合成数据集和四个真实世界数据集上进行比较．我们将 IKD 的贡献总结成如下四点：

• 作为一个基于特征分解的方法，IKD 相较于其它基于特征分解的方法，可以得到更合理的特征表示．在之后的分类任务中可以取得更好的分类准确率．IKD 的运行时间则和其它基于特征分解的方法差不多．

• IKD 可以用更短的时间，给出和最先进的基于优化的方法相当结果．

• IKD 能够在仿射变换下保证最优解的稳定性和唯一性．相比之下，基于特征分解的方法无法保证最优解的唯一性，并且由于非线性优化过程，得到的解可能数值上不稳定．因此 IKD 有时可以得到比基于优化的方法（比如 GPLVM 和 VAE）更好的隐变量表示和分类效果．

• 当观测维度较大时（即观测变量是很高维的），很多方法有明显的缺陷．比如，t-SNE、UMAP 和 VAE 可能会遇到维度灾难．高的维度不仅导致更长的运行时间，还影响了降维效果．相比之下，随着维度升高，IKD 总能得取得更优的效果，这表明了它在高维问题上的绝对优越性．

注意我们不是在声称 IKD 比其它先进的降维方法更优．我们提出的基于特征分解的 IKD，(1) 可以在同样的时间内在合成数据和真实世界应用上取得比其它基于特征分解更好的结果，(2) 能够以更快的速度达到和基于优化的方法相当的效果．

1.1 相关工作

作为 GPLVM 的基于特征分解的求解器, IKD 从数据生成过程开始分析, 而非观测数据. 尽管 IKD 利用了特征分解和核函数，它也和其它基于特征分解的方法不同．比如 GPLVM 用核函数来形成从嵌入的隐变量空间到数据空间的映射，这与核-PCA [Schölkopf et al., 1997] 中的核的用法是相反的．另一个相似的方法是 Isomap [Tenenbaum et al., 2000]．Isomap 是一个推广的多维标度 (multidimensional scaling, MDS) [Borg and Groenen, 2005] 方法．从 2.2 节推导出的 IKD 算法的骨架形式上来看，IKD 将数据的协方差对作为目标相似矩阵，而 Isomap 将数据的广义距离对作为目标相似矩阵．然而，IKD 和 Isomap 最初的动机是不同的．Isomap 欲将数据放在一个低维空间中，同时尽可能使得距离对在高维空间和低维空间中是同比例的．而 IKD 用特征分解的方法求解由 GPLVM 生成的数据．在第 2 节中，我们会先介绍生成模型 GPLVM，然而把这个问题一步步转化成一个特征分解问题，最后对其求解.

2 方法

2.1 高斯过程隐变量模型

生成模型. 令 $\boldsymbol{X} \in \mathbb{R}^{T\times N}$ 为观测数据，其中 T 为观测量（样本个数），N 为每个观测数据向量的维度．令 $\boldsymbol{Z} \in \mathbb{R}^{T\times M}$ 表示对应的隐变量，其中 M 是隐变量的维度．通常来说，我们认为隐变量空间相比原始观测变量空间是低维的，也就是 $M < N$．$\boldsymbol{X}$ 的每个维度记为 $\boldsymbol{X}_{:,n} \in \mathbb{R}^T,\ \forall n \in \{1,\dots,N\}$.

GPLVM 定义了一个映射函数，将隐变量映射到观测变量，而这个过程是有高斯过程 (GP) 先验的．因此，给定观测个数，我们有

$$\boldsymbol{X}_{:,n} \overset{\text{i.i.d}}{\sim} \mathcal{N}(\mathbf{0}, \boldsymbol{K}), \quad \forall n \in \{1, 2, ..., N\}. \tag{1}$$

其中 $\boldsymbol{K}$ 是个 $T \times T$ 的协方差矩阵，由 GP 的核函数 k 计算 $\boldsymbol{Z}$ 的两两行对得到，即 $k_{i,j} = k(\boldsymbol{z}_i, \boldsymbol{z}_j)$ 其中 $\boldsymbol{z}_i$ 和 $\boldsymbol{z}_j$ 是 $\boldsymbol{Z}$ 的第 i 行和第 j 行．

问题设定. GPLVM 的目标是从观测变量 $\boldsymbol{X}$ 估计未知的用于生成协方差矩阵 $\boldsymbol{K}$ 的隐变量 $\boldsymbol{Z}$．注意我们在推导 IKD 时只考虑无噪声的 GP，但 IKD 实际可以处理有噪声的观测变量．

2.2 逆核分解

本节中，我们推导这个新的非线性分解方法，逆核分解 (IKD)，启发自 GPLVM．之间的工作已经通过最大化对数似然，一步从 $\boldsymbol{X}$ 求解出 $\boldsymbol{Z}$ 的．而 IKD 把求解分解成两部：(1) 从 $\boldsymbol{X}$ 估计 $\boldsymbol{Z}$，然后 (2) 从估计的协方差矩阵 $\boldsymbol{K}$ 估计隐变量 $\boldsymbol{Z}$．第一步可以通过 $\boldsymbol{K}$ 的无偏估计得到，也就是样本协方差矩阵 $\boldsymbol{S} \coloneqq \frac{1}{N-1}\left(\boldsymbol{X} - \bar{\boldsymbol{X}}\mathbf{1}^{\mathrm{T}}\right)\left(\boldsymbol{X} - \bar{\boldsymbol{X}}\mathbf{1}^{\mathrm{T}}\right)^{\mathrm{T}} \approx \frac{1}{N-1}\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}$，其中 $\bar{\boldsymbol{X}} = \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{X}_{:,n}$ 理应为 $\mathbf{0}$，因为 $\boldsymbol{X}_{:,n}$ 是来自零均值高斯（公式 1）的独立同分布样本．

因此，我们这里着重看第二步如何用 $\boldsymbol{S}$ 估计出 $\boldsymbol{Z}$．下面我们先用最常用的平稳核，也就是平方指数 (squared exponential, SE) 核来推导．在 2.3 节中我们会说明 IKD 适用于众多平稳核上．

SE 核的定义是 $k(\boldsymbol{z}_i, \boldsymbol{z}_j) = \sigma^2 \exp\left(-\frac{\|\boldsymbol{z}_i - \boldsymbol{z}_j\|^2}{2l^2}\right)$，其中 σ^2 是边缘方差 (marginal variance)，l 是长度刻度 (length-scale)．注意到 $\sigma^2 = k(\boldsymbol{z}_i, \boldsymbol{z}_i) = k_{i,i},\ \forall i \in \{1, \cdots, T\}$．令 f 为将放缩后的隐变量对 $(\boldsymbol{z}_i, \boldsymbol{z}_j)$ 的平方距离 $d_{i,j} \coloneqq \frac{\|\boldsymbol{z}_i - \boldsymbol{z}_j\|^2}{l^2}$ 映射到协方差 $k_{i,j}$ 的标量函数，即 $k_{i,j} = f(d_{i,j}) = \sigma^2 \exp\left(-\frac{d_{i,j}}{2}\right)$．先假设我们知道真实的 $\boldsymbol{K}$．因为 $f(\cdot)$ 是严格单调的，所以我们可以得到 $d_{i,j} = f^{-1}(k_{i,j}) = -2\ln\left(\frac{k_{i,j}}{\sigma^2}\right)$．把 $d_{i,j}$ 的矩阵形式表示为 $\boldsymbol{D} = (d_{i,j})_{T\times T}$，就有

$$\begin{aligned}\boldsymbol{D} =& \frac{1}{l^2}\begin{bmatrix} 0 & (\boldsymbol{z}_1 - \boldsymbol{z}_2)^{\mathrm{T}}(\boldsymbol{z}_1 - \boldsymbol{z}_2) & \cdots & (\boldsymbol{z}_1 - \boldsymbol{z}_T)^{\mathrm{T}}(\boldsymbol{z}_1 - \boldsymbol{z}_T) \\ (\boldsymbol{z}_2 - \boldsymbol{z}_1)^{\mathrm{T}}(\boldsymbol{z}_2 - \boldsymbol{z}_1) & 0 & \cdots & (\boldsymbol{z}_2 - \boldsymbol{z}_T)^{\mathrm{T}}(\boldsymbol{z}_2 - \boldsymbol{z}_T) \\ \vdots & \vdots & \ddots & \vdots \\ (\boldsymbol{z}_T - \boldsymbol{z}_1)^{\mathrm{T}}(\boldsymbol{z}_T - \boldsymbol{z}_1) & (\boldsymbol{z}_T - \boldsymbol{z}_2)^{\mathrm{T}}(\boldsymbol{z}_T - \boldsymbol{z}_2) & \cdots & 0 \end{bmatrix} \\ =& f^{-1}(\boldsymbol{K}) = \left[f^{-1}(k_{i,j})\right]_{T\times T},\end{aligned} \tag{2}$$

其中 f^{-1} 把 $\boldsymbol{K}$ 逐元素的映射到 $\boldsymbol{D}$．定义 $\tilde{\boldsymbol{z}} = \frac{\boldsymbol{z} - \boldsymbol{z}_1}{l}$，同时有 $\tilde{\boldsymbol{z}}_1 = \mathbf{0}$．则

$$\begin{aligned} d_{i,j} =& \frac{1}{l^2}(\boldsymbol{z}_i - \boldsymbol{z}_j)^{\mathrm{T}}(\boldsymbol{z}_i - \boldsymbol{z}_j) = \frac{1}{l^2}\left[(\boldsymbol{z}_i - \boldsymbol{z}_1) - (\boldsymbol{z}_j - \boldsymbol{z}_1)\right]^{\mathrm{T}}\left[(\boldsymbol{z}_i - \boldsymbol{z}_1) - (\boldsymbol{z}_j - \boldsymbol{z}_1)\right] \\ =& (\tilde{\boldsymbol{z}}_i - \tilde{\boldsymbol{z}}_j)^{\mathrm{T}}(\tilde{\boldsymbol{z}}_i - \tilde{\boldsymbol{z}}_j) = \tilde{\boldsymbol{z}}_i^{\mathrm{T}}\tilde{\boldsymbol{z}}_i + \tilde{\boldsymbol{z}}_j^{\mathrm{T}}\tilde{\boldsymbol{z}}_j - 2\tilde{\boldsymbol{z}}_i^{\mathrm{T}}\tilde{\boldsymbol{z}}_j. \end{aligned} \tag{3}$$

因为 $\tilde{\boldsymbol{z}}_1 = \mathbf{0}$，所以有 $d_{1,j} = \tilde{\boldsymbol{z}}_1^{\mathrm{T}}\tilde{\boldsymbol{z}}_1 + \tilde{\boldsymbol{z}}_j^{\mathrm{T}}\tilde{\boldsymbol{z}}_j - 2\tilde{\boldsymbol{z}}_1^{\mathrm{T}}\tilde{\boldsymbol{z}}_j \implies \tilde{\boldsymbol{z}}_j^{\mathrm{T}}\tilde{\boldsymbol{z}}_j = d_{1,j},\ \forall j \in \{1, ..., T\}$．因此，就得到了表达式 $\tilde{\boldsymbol{z}}_i^{\mathrm{T}}\tilde{\boldsymbol{z}}_j$ 的表达式 $\tilde{\boldsymbol{z}}_i^{\mathrm{T}}\tilde{\boldsymbol{z}}_j = \frac{1}{2}(d_{i,1} + d_{1,j} - d_{i,j})$．注意到这里我们有对称性 $d_{1,i} = d_{i,1}$．记

$\tilde{\boldsymbol{Z}} = [\tilde{\boldsymbol{z}}_1, \tilde{\boldsymbol{z}}_2, \cdots, \tilde{\boldsymbol{z}}_T]^{\mathrm{T}} = [\boldsymbol{0}, \tilde{\boldsymbol{z}}_2, \cdots, \tilde{\boldsymbol{z}}_T]^{\mathrm{T}} \in \mathbb{R}^{T\times M}$，就可以写出 $\tilde{\boldsymbol{Z}}\tilde{\boldsymbol{Z}}^{\mathrm{T}} = \left(\tilde{\boldsymbol{z}}_i^{\mathrm{T}}\tilde{\boldsymbol{z}}_j\right)_{T\times T}$ 的矩阵形式

$$\tilde{\boldsymbol{Z}}\tilde{\boldsymbol{Z}}^{\mathrm{T}} = \left[\begin{array}{c|ccc} 0 & 0 & \cdots & 0 \\ \hline 0 & d_{2,1} & \cdots & \frac{1}{2}(d_{2,1}+d_{1,T}-d_{2,T}) \\ 0 & \frac{1}{2}(d_{3,1}+d_{1,2}-d_{3,2}) & \cdots & \frac{1}{2}(d_{3,1}+d_{1,T}-d_{3,T}) \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \frac{1}{2}(d_{T,1}+d_{1,2}-d_{T,2}) & \cdots & d_{T,1} \end{array}\right] =: g(\boldsymbol{D}) = g(f^{-1}(\boldsymbol{K})), \tag{4}$$

这是一个秩为 M 的对称半正定矩阵，因为 $M < T$. g 是将 $\boldsymbol{D}$ 映射到 $\tilde{\boldsymbol{Z}}\tilde{\boldsymbol{Z}}^{\mathrm{T}}$ 的函数. 那么，公式 4 就有唯一的规约的矩阵分解

$$g(f^{-1}(\boldsymbol{K})) = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{\mathrm{T}} = \left(\sqrt{\lambda_1}\boldsymbol{U}_{:,1}, \ldots, \sqrt{\lambda_M}\boldsymbol{U}_{:,M}\right)\left(\sqrt{\lambda_1}\boldsymbol{U}_{:,1}, \ldots, \sqrt{\lambda_M}\boldsymbol{U}_{:,M}\right)^{\mathrm{T}} =: \tilde{\boldsymbol{U}}\tilde{\boldsymbol{U}}^{\mathrm{T}}, \tag{5}$$

其中 $\boldsymbol{U}_{:,m} = [0, u_{2,m}, u_{3,m}, \ldots, u_{T,m}]^{\mathrm{T}} \in \mathbb{R}^T$ 是 $\boldsymbol{U} \in \mathbb{R}^{T\times M}$ 的第 m^{th} 列，$\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_M)$ $(\lambda_1 > \lambda_2 > \cdots > \lambda_M > \lambda_{M+1} = \cdots = \lambda_T = 0)$. 而由于 $\tilde{\boldsymbol{Z}}$ 的唯一规约的奇异值分解为

$$\tilde{\boldsymbol{Z}} = \boldsymbol{U}\boldsymbol{\Lambda}^{\frac{1}{2}}\boldsymbol{V}^{\mathrm{T}} = \tilde{\boldsymbol{U}}\boldsymbol{V}^{\mathrm{T}} \implies \boldsymbol{z}_t = l\boldsymbol{V}\tilde{\boldsymbol{U}}_{t,:}^{\mathrm{T}} + \boldsymbol{z}_1, \quad \forall t \in \{1, \ldots, T\}, \tag{6}$$

其中 $\boldsymbol{z}_1$ 为参考平移量，长度刻度 l 是放缩因子，$\boldsymbol{V}$ 是用于旋转和翻转的正交矩阵. 由于 $\boldsymbol{Z}$ 和 $\tilde{\boldsymbol{U}}$ 张成同样的列空间，满足

$$\begin{aligned} k(\boldsymbol{z}_i, \boldsymbol{z}_j) =& \sigma^2 \exp\left(-\frac{\|\boldsymbol{z}_i - \boldsymbol{z}_j\|^2}{2l^2}\right) = \sigma^2 \exp\left(-\frac{\|l\boldsymbol{V}\tilde{\boldsymbol{U}}_{i,:}^{\mathrm{T}} - l\boldsymbol{V}\tilde{\boldsymbol{U}}_{j,:}^{\mathrm{T}}\|^2}{2l^2}\right) \\ =& \sigma^2 \exp\left(-\frac{\|\tilde{\boldsymbol{U}}_{i,:} - \tilde{\boldsymbol{U}}_{j,:}\|^2}{2}\right) = k_{l=1}\left(\tilde{\boldsymbol{U}}_{i,:}, \tilde{\boldsymbol{U}}_{j,:}\right), \end{aligned} \tag{7}$$

$\tilde{\boldsymbol{U}}$ 包含了 $\boldsymbol{Z}$ 所有的低维信息. 因此我们可以把 $\tilde{\boldsymbol{U}}$ 视为 $\boldsymbol{Z}$ 的一个估计. 公式 5 和公式 6 总结了从 GP 核协方差矩阵到隐变量的这一逆关系.

目前为止，我们已经能在给定真正的由 $\boldsymbol{Z}$ 生成的 GP 协方差核 $\boldsymbol{K}$（公式 7）的情况下，找到精准估计 $\tilde{\boldsymbol{U}}$ 了. 而实际情况下，我们只有样本协方差估计量 $\boldsymbol{S}$，那么 $g(f^{-1}(\boldsymbol{S}))$ 既不是秩为 M，也不保证半正定. 因此我们需要找它的最优秩为 M 的半正定近似，即

$$\underset{\tilde{\boldsymbol{U}}\in\mathbb{R}^{T\times M}}{\text{minimize}} \left\|\boldsymbol{g}(f^{-1}(\boldsymbol{S})) - \tilde{\boldsymbol{U}}\tilde{\boldsymbol{U}}^{\mathrm{T}}\right\|. \tag{8}$$

Dax et al. [2014] 表明

$$\tilde{\boldsymbol{U}} = \left(\sqrt{\lambda_1}\boldsymbol{U}_{:,1}, \ldots, \sqrt{\lambda_M}\boldsymbol{U}_{:,M}\right) \tag{9}$$

在任意酉不变矩阵范数下为最优估计，其中 $\lambda_1, \ldots, \lambda_M$ 是 $g(f^{-1}(\boldsymbol{S}))$ 的前 $\boldsymbol{M}$ 个最大的正特征值，$\boldsymbol{U}_{:,1}, \ldots, \boldsymbol{U}_{:,M}$ 为对应的特征向量. 通过目标损失函数公式 8 估计的隐变量的优度可以通过解释方差比（解释变异比, explained variance ratio）$\frac{\sum_{t=1}^{M}\lambda_t^2}{\sum_{t=1}^{T}\lambda_t^2}$ 来量化. 我们把 IKD 算法总结在算法 1 中.

Algorithm 1 Inverse kernel decomposition

function IKD($\boldsymbol{X} \in \mathbb{R}^{T\times N}, f$)
  $\boldsymbol{S} \leftarrow \frac{1}{N-1}\left(\boldsymbol{X}-\bar{\boldsymbol{X}}\mathbf{1}^{\mathrm{T}}\right)\left(\boldsymbol{X}-\bar{\boldsymbol{X}}\mathbf{1}^{\mathrm{T}}\right)^{\mathrm{T}}$ ▷ $\boldsymbol{S}$ 作为协方差矩阵 $\boldsymbol{K}$ 的估计
  $\sigma^2 \leftarrow \frac{1}{T}\sum_{i=1}^{T} s_{i,i}$ ▷ 通过 $\boldsymbol{S}$ 的对角的某种统计量来估计 σ^2
  $\hat{\boldsymbol{D}} = (\hat{d}_{i,j})_{T\times T} \leftarrow f^{-1}(\boldsymbol{S})$ ▷ $\hat{\boldsymbol{D}}$ 作为 $\boldsymbol{D}$ 的估计（公式 2）
  $\boldsymbol{U}, \boldsymbol{\Lambda} \leftarrow$ 对 $g(\hat{\boldsymbol{D}})$ 做特征分解 ▷ 公式 5
  用 $\boldsymbol{U}$ 和 $\boldsymbol{\Lambda}$ 形成最优隐变量解 $\tilde{\boldsymbol{U}}$ ▷ 公式 5
  return $\tilde{\boldsymbol{U}}$
end function

表 1: 适用于 IKD 的平稳核.

kernel	f	f^{-1}
squared exponential	$f(d)=\sigma^2\exp\left(-\frac{d}{2}\right)$	$f^{-1}(k)=-2\ln\left(\frac{k}{\sigma^2}\right)$
rational quadratic	$f(d)=\sigma^2\left(1+\frac{d}{2\alpha}\right)^{-\alpha}$	$f^{-1}(k)=2\alpha\left[\left(\frac{k}{\sigma^2}\right)^{-\frac{1}{\alpha}}-1\right]$
γ-exponential	$f(d)=\sigma^2\exp\left(-d^{\frac{\gamma}{2}}\right)$	$f^{-1}(k)=\left(-\ln\frac{k}{\sigma^2}\right)^{\frac{2}{\gamma}}$
Matérn	$f(d)=\sigma^2\frac{2^{1-\nu}}{\Gamma(\nu)}\left(\sqrt{2\nu}\sqrt{d}\right)^{\nu}K_\nu\left(\sqrt{2\nu}\sqrt{d}\right)$	没解析解但可以用求根算法求解

2.3 推广到带有平稳核的 IKD

除了 SE 核，IKD 还可以用于接大多数常用的平稳核，只要核函数 f 可逆（也就是 f 在 $[0,\infty)$ 上要严格单调），就可以为 $d=f^{-1}(k)$ 找到唯一的非负解．我们把这些核总结在表 1 中.

对于 SE 核，我们可以将其推广到 ARD 核 $k(\boldsymbol{z}_i,\boldsymbol{z}_j)=\sigma^2\exp\left(-\frac{1}{2}\sum_{m=1}^{M}\frac{1}{l_m^2}(z_{i,m}-z_{j,m})^2\right)$ 和高斯核 $k(\boldsymbol{z}_i,\boldsymbol{z}_j)=\sigma^2\exp\left(-\frac{1}{2}(\boldsymbol{z}_i-\boldsymbol{z}_j)^{\mathrm{T}}\boldsymbol{L}^{-1}(\boldsymbol{z}_i-\boldsymbol{z}_j)\right)$，其中需要一个额外的仿射变换 $\boldsymbol{L}^{\frac{1}{2}}$ 而非简单的一个放缩常数 l.

对于由 ν 参数化的 Matérn 核，其本质是一个第二类修正贝塞尔函数 (modified Bessel function of the second kind)．尽管得到 Matérn 核 $f^{-1}(\cdot)$ 的解析解比较困难，但解 $f^{-1}(\cdot)$ 总是存在的，因为 $f(\cdot)$ 在 $[0,\infty)$ 上是严格单调递减的，对所有 $\nu>0$ 恒成立．注意到对于常用的 $\nu=p+\frac{1}{2},\ p\in\mathbb{N}$，$f'(\cdot)$ 很好算．比如，当 $\nu=\frac{3}{2}$, $f(d)=\sigma^2\left(1+\frac{\sqrt{3}d}{l}\right)\exp\left(-\frac{\sqrt{3}d}{l}\right)$，那么 $f'(d)=-\frac{3d\sigma^2}{l^2}\exp\left(-\frac{\sqrt{3}d}{l}\right)$．在这些情况下，高阶求根算法（如，牛顿法）可以用于求解 $d=f^{-1}(k)$.

IKD 的直观做法是在找非线性映射，使得在高维上强相关的点 $\boldsymbol{x}_i,\boldsymbol{x}_j$ 在低维隐变量空间中也相近．这很好的由平稳核的形状表现出来，也就在 $d=0$ 处单调递减，然后当 $d\to\infty$ 的过程中专拣边平缓，收敛到 0．鉴于这些平稳核形状都差不多，不同核估计出的隐变量也相似（见真实世界实验的最后一自然段，3.2 节）．由于这些核都是可你的，我们可以得到两个不同的核 f_1 和 f_2 得到的两个隐变量 $\tilde{\boldsymbol{U}}^{(1)}$ 和 $\tilde{\boldsymbol{U}}^{(2)}$ 的精准关系．记 $\tilde{D}^{(1)}=\left[(\tilde{U}^{(1)}_{i,:}-\tilde{U}^{(1)}_{j,:})^{\mathrm{T}}(\tilde{U}^{(1)}_{i,:}-\tilde{U}^{(1)}_{j,:})\right]_{T\times T}$．$\tilde{D}^{(2)}$ 类似，那么

$$f_1\left(\tilde{D}^{(1)}\right)=\boldsymbol{S}=f_2\left(\tilde{D}^{(2)}\right). \tag{10}$$

所以从 $\tilde{D}^{(1)}$ 到 $\tilde{D}^{(2)}$ 的变换是一个微分同胚（映射）$f_2^{-1}f_1$.

2.4 IKD 的误差分析

IKD 在 $g(f^{-1}(\boldsymbol{S}))$ 上做特征分解（公式 5），这个用到了 $\boldsymbol{K}$ 的经验估计，也就是样本协方差 $\boldsymbol{S}$. 实际情况下，样本协方差 $\boldsymbol{S}$ 中的值 $s_{i,j}$ 可能因为数据本身的噪声以及不足的观测维度 $\boldsymbol{N}$ 含有非常大的噪声. 可能还有非正的或非常接近零的正值，导致没办法精准计算 $\hat{d}_{i,j}=f^{-1}(s_{i,j})$. 非正的 $s_{i,j}$ 落入了 f^{-1} 的输入范围，也就是 $(0,\sigma^2]$. 而对接近零的正值 $s_{i,j}$，估计值 $\hat{d}_{i,j}=f^{-1}(s_{i,j})$ 和真实值 $d_{i,j}=f^{-1}(k_{i,j})$ 之间的误差就会非常大，而且对 $s_{i,j}$ 很敏感. 可以用 f^{-1} 在 $s_{i,j}$ 处的泰勒展开对 SE 核简单的误差分析：

$$\begin{aligned} d_{i,j} =& f^{-1}(k_{i,j}) = -2\ln\frac{s_{i,j}+(k_{i,j}-s_{i,j})}{\sigma^2} = -2\ln\frac{s_{i,j}}{\sigma^2} - 2\frac{k_{i,j}-s_{i,j}}{s_{i,j}} + O((k_{i,j}-s_{i,j})^2) \\ =& f^{-1}(s_{i,j}) + \frac{O(k_{i,j}-s_{i,j})}{s_{i,j}} = \hat{d}_{i,j} + \frac{O(k_{i,j}-s_{i,j})}{s_{i,j}}. \end{aligned} \tag{11}$$

定义估计误差为 $|d_{i,j}-\hat{d}_{i,j}| = \frac{O(|k_{i,j}-s_{i,j}|)}{s_{i,j}}$. 对于较大的 $s_{i,j}$，误差比较小；但对于较小的 $s_{i,j}$，误差就对协方差误差 $|k_{i,j}-s_{i,j}|$ 很敏感. 为了解决这个问题，有两种办法：

分块解. 首先先用 s_0 作为阈值，滤掉太差的 $s_{i,j}$，得到了一个阈值化的协方差矩阵 $\tilde{\boldsymbol{S}} = (s_{i,j}\cdot\mathbb{1}[s_{i,j}>s_0])_{T\times T}$. $\tilde{\boldsymbol{S}}$ 由于很多零的存在可能不是个全连通图. 没法直接将 IKD 用在 $\tilde{\boldsymbol{S}}$ 上估计隐变量. 所以我们可以用像 Bron–Kerbosch [Bron and Kerbosch, 1973] 这样的算法，找 $\tilde{\boldsymbol{S}}$ 中的极大团. 之后每个团都是 $\tilde{\boldsymbol{S}}$ 的一个全连通子图（块），可以用 IKD 做特征分解. 得到每个团的隐变量后，再利用每对团之间共享的点，把这些估计的隐变量拼起来. 只要两个团之间共享的隐变量个数大于 M，就能找到唯一最优的刚性（刚体）变换 (rigid transformation) 把两个团正确地匹配起来.

下面我们把分块解中合并团的过程阐述清楚. 首先，Bron-Kerbosch 算法给出了一个团的集合 $\{C_1,C_2,\ldots,C_L\}$，满足 $\bigcup_{i=1}^{L}C_i=\{\boldsymbol{z}_t\}_{t=1}^{T}$. 首先从共享最多个点的两个团 $\arg\max_{C_i,C_j}|C_i\cap C_j|$ 开始. 比如，有 $C_i=\{\boldsymbol{z}_1,\boldsymbol{z}_2,\boldsymbol{z}_3,\boldsymbol{z}_4\}$ 和 $C_2=\{\boldsymbol{z}_2,\boldsymbol{z}_3,\boldsymbol{z}_4,\boldsymbol{z}_5\}$（图 1）. 他们共享的点是 $\{\boldsymbol{z}_2,\boldsymbol{z}_3,\boldsymbol{z}_4\}$. 由于隐变量在 $\mathbb{R}^M=\mathbb{R}^2$ 中，而 $|C_i\cap C_j|=3\geqslant M+1$，我们就能通过 SVD 找到唯一的反转 + 旋转 + 平移变换将团 C_2 中的 $\{\boldsymbol{z}_2,\boldsymbol{z}_3,\boldsymbol{z}_4\}$ 对齐到团 C_1 中的 $\{\boldsymbol{z}_2,\boldsymbol{z}_3,\boldsymbol{z}_4\}$. 若是它们之间共享的点少于 $M+1$，对齐的方式就有多种. 类似的，在 $\mathbb{R}^M$ 中，唯一的最优对齐变换需要至少 $M+1$ 个共享点. 一旦两个团合并上了，我们就把 C_i,C_j 从原始集中去掉，把合并后的 $C_i\cap C_j$ 放回来. 不断重复这些合并过程，直至最大的团包含所有点，就得到了最终的隐变量.

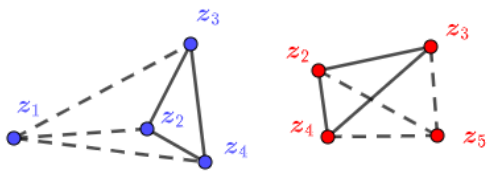

图 1: A clique pair example that shares three points $\{\boldsymbol{z}_2,\boldsymbol{z}_3,\boldsymbol{z}_4\}$.

尽管用 Bron-Kerbosch 算法找极大团的复杂度为 $\mathcal{O}(3^T)$，只要现有团的并能构成整个数据集，算法就可以停止．换句话说，我们最多需要 T 个极大团，所以找团的时间可以由 $\mathcal{O}(T^2)$ 界定．之后为前 T 个解前 M 个特征的特征分解算法需要画 $\mathcal{O}(T \times (MT^2))$．因此，整个过程的复杂度可以被 $\mathcal{O}(MT^3)$ 界定．

测地解. 由于 $s_{i,j} < s_0$ 的值会严重影响特征分解，我们可以把那些值小于 s_0 的 $s_{i,j}$ 换成测地协方差 $s_{i,j} \leftarrow \max_{(t_1,t_2,...,t')} s_{i,t_1} \cdot s_{t_1,t_2} \cdots s_{t',j}$，其中 $i \to t_1 \to \cdots \to t' \to j$ 是从 i 到 j 的测地路径，用 Dijkstra [Dijkstra et al., 1959] 算法就可以找到．这个方法的复杂度由 Dijkstra 算法的复杂度界定，即 $\mathcal{O}(T^2 \log(T))$．由于测地方法的复杂度在 T 较大（在实验中大于 1000 时）时小于分块方法的复杂度，我们就选测地方法．实验部分也会比较两种解.

2.5 参考点的选择

在公式 3 中，我们选了 $\boldsymbol{z}_1$ 作为参考点计算 $d_{i,j}$．但是参考点其实可以是 $\{\boldsymbol{z}_t\}_{t=1}^T$ 中的任意一个点．如果从 $r \in \{1, ..., T\}$ 中任选一个 $\boldsymbol{z}_r$ 作为参考点，那么和公式 4 类似，$g(\boldsymbol{D})$ 的第 r 行和第 r 列都为 0，起源元素为 $g(\boldsymbol{D})_{i,j} = \frac{1}{2}(d_{i,r} + d_{r,j} - d_{i,j}) \neq 0,\ \forall i \neq r, j \neq r$．注意到每一个 $g(\boldsymbol{D})_{i,j}$ 都含有 $\{d_{r,i}\}_{i=1}^T$ 中的点．所以 $\{d_{r,i}\}_{i=1}^T$ 的质量对隐变量估计很重要．基于公式 11 的分析，我们在实际中需要选一个好的参考 r 使得 $\{\hat{d}_{r,i}\}_{i=1}^T$ 相对较小（即，$\{s_{r,i}\}_{i=1}^T$ 较大，这就意味着第 r 个数据点和其它点都很相关．注意到 MDS [Kruskal and Wish, 1978] 也是解类似的特征分解问题，也就是从距离矩阵 $\boldsymbol{D}$ 中找坐标 $\boldsymbol{Z}$．它用的是中心化的方法，相当于用所有隐变量的平均 $\frac{1}{T}\sum_{t=1}^T \boldsymbol{z}_t$ 作为参考点．我们则是选最好的参考点，这样可以尽可能减小公式 11 中的估计误差，从而目标函数 8 可以被尽可能地最小化．数学语言表述就是，选 r，满足 $\|\hat{\boldsymbol{d}}_r\|_\infty \leqslant \|\hat{\boldsymbol{d}}_i\|_\infty,\ \forall i \in \{1, 2, ..., T\}$，其中 $\hat{\boldsymbol{d}}_i$ 是 $\hat{\boldsymbol{D}} = (\hat{d}_{i,j})_{T\times T}$ 的第 ri 行，即 $r = \arg\min_i \|\hat{\boldsymbol{d}}_i\|_\infty = \arg\min_i \left\{\max_j \{\hat{d}_{i,j}\}\right\}$.

3 实验

本节中，我们用最常用的 SE 核作为默认核求解 IKD．实验包括三个合成数据集（知道真实的隐变量）和四个真实世界数据集.

用于比较的基线方法:

- **PCA**：主成分分析．最常用的线性降维方法.
- **KPCA**：核 PCA．我们尝试了不同的核，包括多项式、SE、sigmoid 和 cosine，然后用最好的那个.
- **LE** [Belkin and Niyogi, 2003]：拉普拉斯特征图，对近似矩阵的图拉普拉斯做谱分解.
- **Isomap** [Tenenbaum et al., 2000]：MDS 下的等距特征映射，用的是测地距离．我们用 `sklearn` 的默认（五个近邻）设置构建距离矩阵.
- **t-SNE** [Van der Maaten and Hinton, 2008b]：t 分布随机近邻嵌入．我们用 `sklearn` 的默认超参数设置拟合模型.
- **UMAP** [McInnes et al., 2020]：Uniform Manifold Approximation and Projection．我们用 UMAP 的官方实现 [McInnes et al., 2018] 及其默认超参数拟合模型.

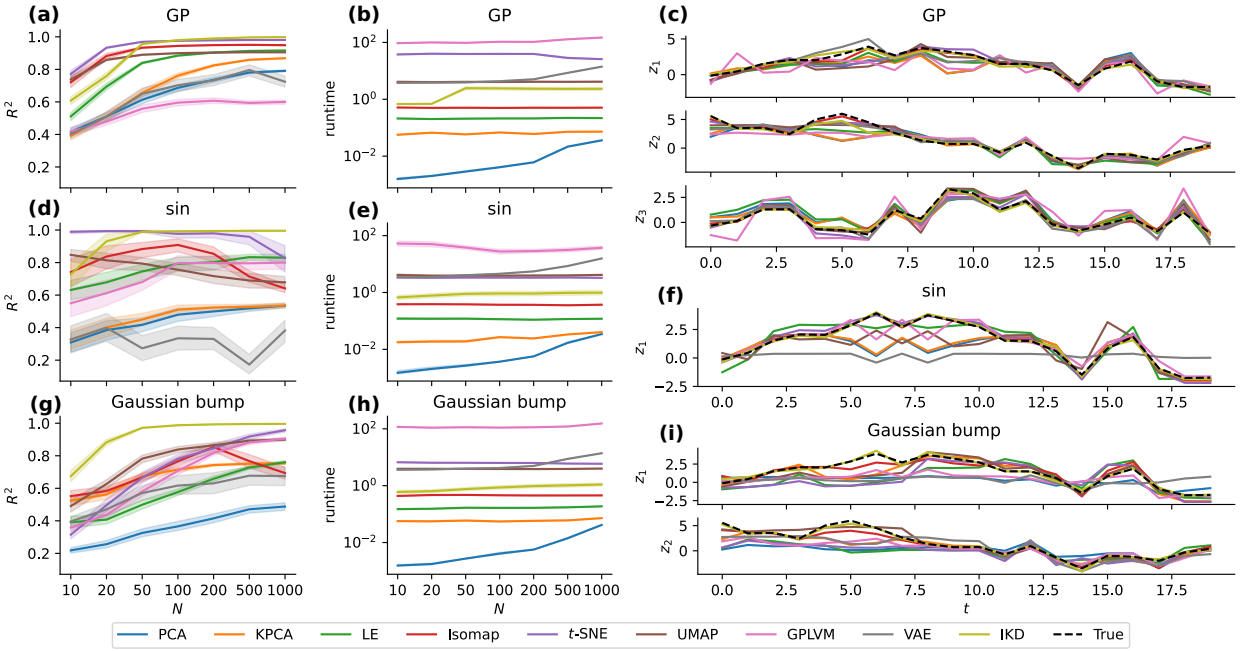

图 2: GP、正弦和高斯隆起的 R^2 值 (a,d,g) 和运行时间 (b,e,h) 关于 N 的变化. (c,f,i): GP、正弦和高斯隆起映射函数在 $N = 100$ 的某一个试次的隐变量恢复，只画了前 20 个隐变量.

- **GPLVM** [Lawrence, 2003, 2005]: 传统的基于优化的高斯过程隐变量模型 (GPLVM) 求解器. 我们用 `GPy` [GPy, since 2012] 这个包的 GPLVM 模块及其默认参数你和模型. 和 IKD 一样，不加说明的话，我们选择最常用的 SE 核作为默认核.
- **VAE** [Kingma and Welling, 2013] 变分自编码器.

前四个是基于特征分解的方法；后四个是基于优化的方法.

3.1 合成数据

实验设置. 首先在三个合成数据集上比较不同方法. 所有实验重复 50 次. 每个试次，从

$$\boldsymbol{Z}_{m,1:T} \sim \mathcal{N}\left(\mathbf{0}, \left(6\mathrm{e}^{-\frac{|i-j|}{5}}\right)_{T\times T}\right), \quad \forall m \in \{1, ..., M\} \tag{12}$$

生成隐变量，其中 M 是隐变量维度，不同数据集的 M 不一样. 然后我们从 GP、正弦和高斯隆起映射函数采没噪声的隐变量. 然后，再在每个样本上加上独立同分布的高斯噪声，得到最终带噪声的观测数据 $\boldsymbol{X}$.

评价方法. 我们用 R^2 评价效果. 计算 R^2 时，首先用一个仿射变换（也就是一个线性解码器）将估计的隐变量对齐到真实的隐变量；之后在每个维度上计算 R^2，然后对所有维度 $m \in \{1, 2, ..., M\}$ 求平均得到最终结果. 选择仿射变换的原因有二: (1) 除了 IKD 以外的方法（比如 PCA），如果只用刚性变换可能会导致非常负的 R^2; (2) 仿射变换在隐变量估计和对齐上时最常用的.

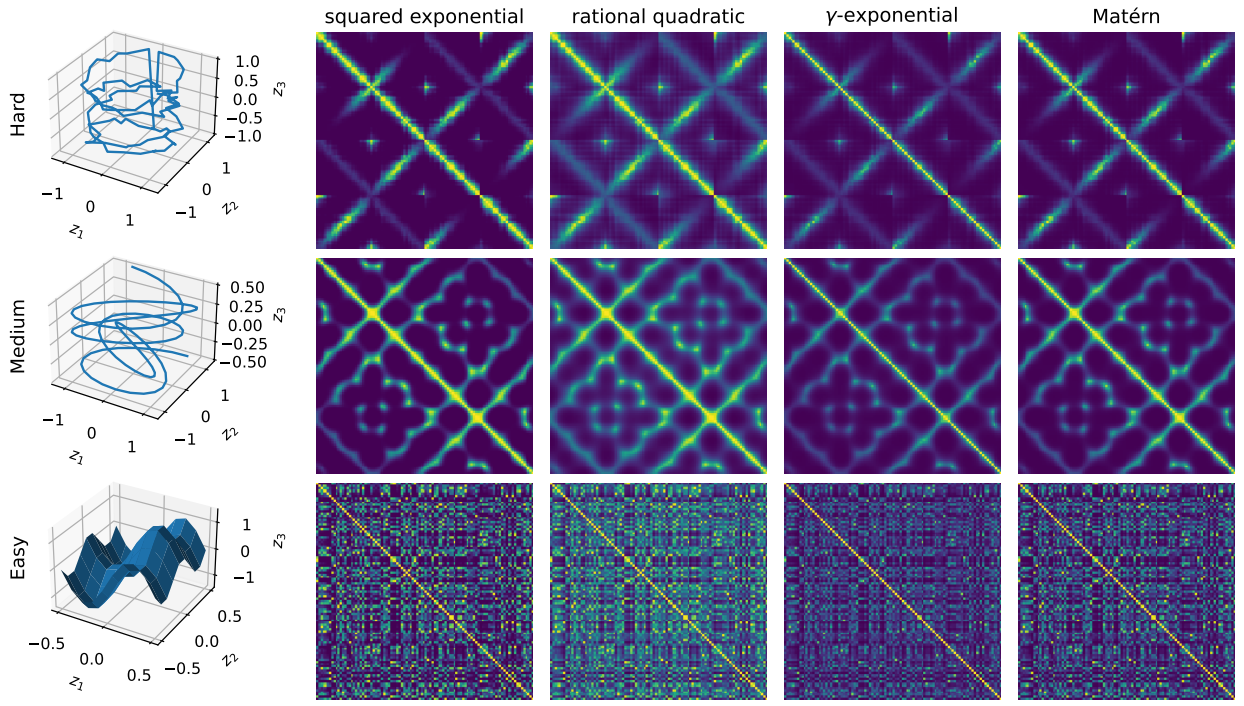

图 3: 左：真实的隐变量 $\boldsymbol{Z}$. 右：四个不同的核的隐变量得到的核协方差矩阵 $\boldsymbol{K}$，边缘方差为 $\sigma = 1$，长度刻度 $l = 0.5$；在 rational quadratic 核中取 $\alpha = 1$，γ-exponential 中 $\gamma = 1$，Matérn 核中 $\nu = 1.5$. 三个数据集从下到上越来越难. 对于简单数据集（下），数据点是从曲面上的网格中选出来然后随机打乱顺序的.

数据集 1：GP 映射函数. 首先先看 GP 映射函数. 每个试次中，我们根据公式 12生成一个 3D 隐变量 $\boldsymbol{Z} \in \mathbb{R}^{1000\times 3}$（即 $M = 3$），然后从公式 1 生成 $\boldsymbol{X} \in \mathbb{R}^{1000\times N}$，用 $\sigma^2 = 1$ 和 $l = 3$. 之后加上高斯噪声：$x_{t,n} \leftarrow x_{t,n} + \varepsilon_{t,n}$，$\forall (t,n) \in \{1,...,1000\} \times \{1,...,N\}$，其中噪声是 $\varepsilon_{t,n} \sim \mathcal{N}(0, 0.05^2)$. 注意这个数据生成过程是和 GPLVM 的生成过程一致的. 所以这是和 IKD 的模型假设一致的情况，视为数据匹配情况. 图 2(a) 是结果. 对于 $N = 10$ 和 $N = 20$，Isomap 是最好的；但是当 $N > 50$ 时，IKD 就是最好的了，R^2 随 N 的增加收敛到 1. 当 $N = 100$ 时，前 20 个点的隐变量恢复结果是图 2(c)，其中 Isomap 和 IKD 匹配真实隐变量最好.

数据集 2：正弦映射函数. 每个试次里，我们根据公式 12 生成一个 1D 隐变量（即 $M = 1$），然后用 $\boldsymbol{x}_t = \sin(\boldsymbol{\Omega}\boldsymbol{z}_t + \boldsymbol{\varphi}) + \boldsymbol{\varepsilon}_t$，$\forall t \in \{1,...,1000\}$ 生成带噪声的观测变量 $\boldsymbol{X} \in \mathbb{R}^{1000\times N}$，其中 $\boldsymbol{\Omega} = (\omega_{n,m})_{N\times M}$ with $\omega_{n,m} \sim \mathcal{U}(-1,1)$，$\boldsymbol{\varphi} = [\varphi_1,...,\varphi_N]^{\mathrm{T}}$ with $\varphi_n \sim \mathcal{U}(-\boldsymbol{\pi},\boldsymbol{\pi})$，噪声 $\boldsymbol{\varepsilon}_t \sim \mathcal{N}(\boldsymbol{0}, 0.1^2\boldsymbol{I})$. 图 2(d) 所示的结果表明，即便观测数据不是来自 GP（数据不匹配情况），IKD 还是能找到比其它方法好的隐变量，除了在 $N = 10$ 和 $N = 20$ 的时候 Isomap 最好. 当观测维度 $N > 50$ 时，IKD 的 R^2 接近 1，而 Isomap、t-SNE 和 UMAP 由于维度灾难表现都在下降. 在 $N = 100$ 的情况下，不同方法的隐变量恢复结果（图 2(f)）表明 Isomap 和 IKD 和真实的隐变量匹配的最好.

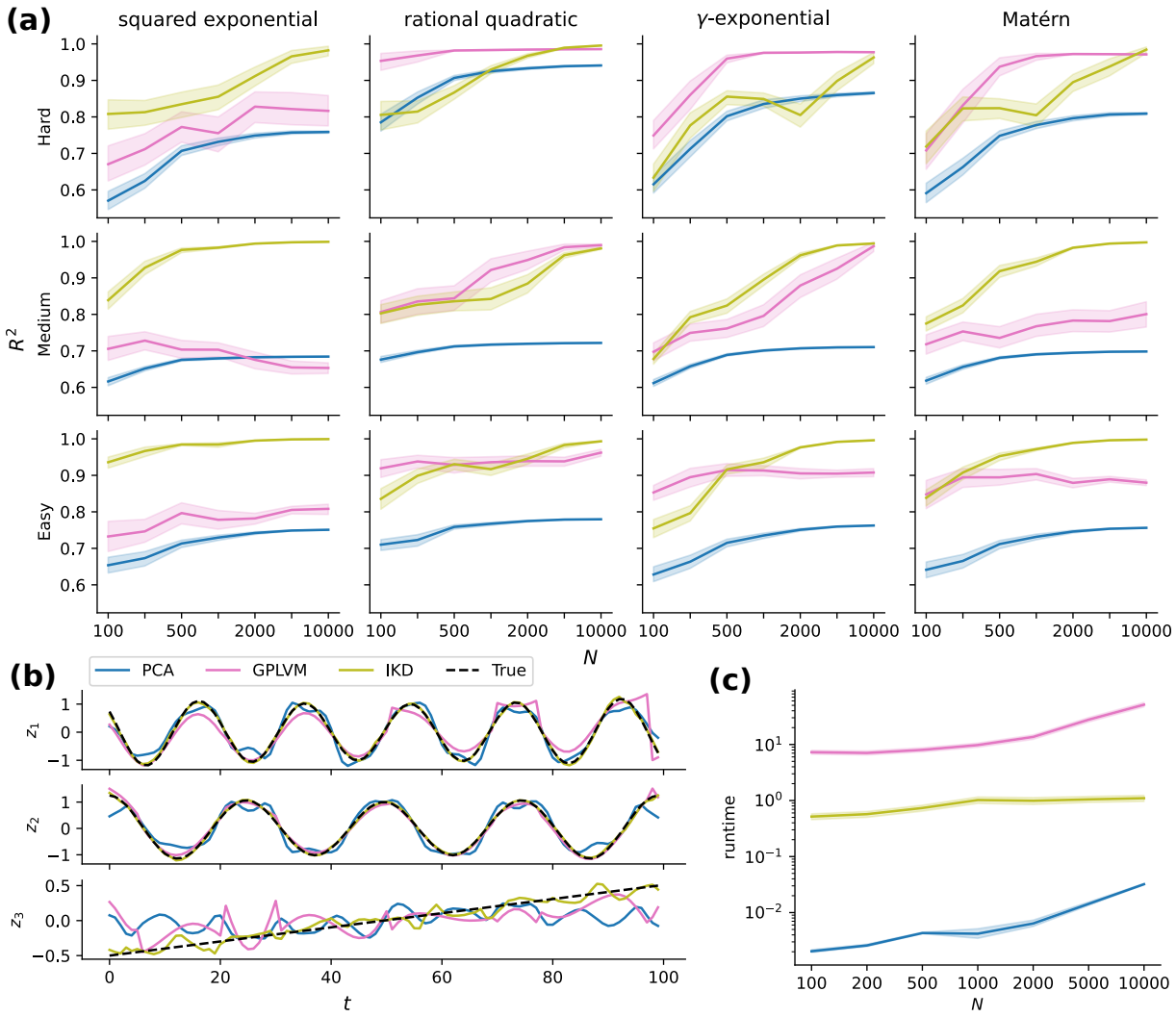

图 4: (a)：PCA、GPLVM 和 IKD 的 R^2 关于 N 的变化. (b)：中等难度数据集, 用 SE 核在 $N = 10000$ 时的隐变量恢复. (c)：平均所有核、所有数据集以及 50 个独立试次后的平均运行时间关于 N 的变化. 对不同核和不同数据集上平均是因为它们运行时间差不多.

数据集 3：高斯隆起映射函数. 每个试次，我们用公式 12 生成 2D 隐变量（即 $M = 2$），并用

$$x_{t,n} = 20 \exp\left(-\|\boldsymbol{z}_t - \boldsymbol{c}_n\|_2^2\right) + \varepsilon_{t,n}, \quad \forall (t, n) \in \{1, ..., 1000\} \times \{1, ..., N\} \tag{13}$$

生成 $\boldsymbol{X} \in \mathbb{R}^{1000 \times N}$，其中 $\boldsymbol{c}_n \in \mathbb{R}^2$ 是第 n 个高斯隆起，这些点是从 10000 个 $[-6, 6]^2$ 的均匀分布的网格点中随机选取出来的，噪声是 $\varepsilon_{t,n} \sim \mathcal{N}(0, 0.05^2)$. 这还是个数据不匹配的例子. 图 2(g) 表明这种情况下，IKD 在所有观测维度 N 所有方法下是最好的. 图 2(i) 也表明只有 IKD 精准的和真实隐变量匹配得上.

IKD 在三个合成数据集上的运行时间和其它基于特征分解的方法差不多（图 2(b,e,h)），显著小于基于优化的方法. 特别的，GPLVM 总是要花很长的是缠绵；t-SNE 在隐变量维度大于 2 的时候需要

非常长的时间；VAE 则是在观测维度比较高的时候要花很长时间．注意这里是在变化观测维度 N 而不是观测个数 T．当 T 增加的时候，所有办法的复杂度都以多项式的复杂度增加．对于基于优化的方法，随即优化方法使得大数据集也能跑．若是变化数据集的大小，大型矩阵分解方法则需要额外的大尺度技巧，不在本文考虑之中．

总的来说，IKD 在所有三个映射函数数据集上表现得最好，尤其是当观测维度 N 很大的时候．除了能正确恢复隐变量的整体结构外，IKD 在捕捉隐变量细节上也非常有效．和其它基于特征分解的方法一样，IKD 在这些合成数据集问题上比其它基于优化的方法要快．

3.1.1 变化维度、核和隐变量结构

我们在 GP 映射函数（公式 1）上测试 IKD 的表现．这里我们变化观测维度 $N \in \{100, 200, 500, 1000, 2000, 5000, 10000\}$、核函数（表 1）、以及隐变量结构（困难、中等、简单，如图 3所示）．这里只讲 IKD 和 PCA 以及 GPLVM 比较．PCA 是最常用的线性方法，所以可以视为基线模型．GPLVM 则是传统的基于优化的求解器．IKD 是我们新给出的基于特征分解的求解器，用于求解来自 GP 映射函数的数据．

图 4(b) 表明 IKD 在最常用的 SE 核上效果最好．和 PCA 以及 GPLVM 比较，IKD 效果非常好，尤其是在 N 很大的时候，IKD 的 R^2 非常接近 1．图 4(c) 展示了在中等数据集 $N = 1000$ 的情况下，某个试次的隐变量的恢复情况．这里的核还是 SE．这里可以看出 IKD 还是和真实的隐变量匹配的最好．尤其是第三个维度，只有 IKD 恢复的隐变量能正确地反映出线性上升的趋势．从复杂度的角度讲，GPLVM 和 IKD 比非常费时间（图 4(d)）．这些结果表明 IKD 能快速精准地恢复由 GP 映射函数生成的数据．

3.1.2 变化噪声等级

为理解 IKD 求解不同等级噪声的 GPLVM, 我们用中等难度数据集（图 3）以及 SE 核作为 GPLVM 来生成无噪声的观测量 $\boldsymbol{X}$（公式 1）．然后我们加上不同等级的噪声，也就是 $x_{t,n} \leftarrow x_{t,n} + \varepsilon_{t,n}$，其中 $\varepsilon_{t,n} \sim \mathcal{N}(0, \mathrm{sd}^2)$，这里变化 $\mathrm{sd} \in \{0, 0.1, \ldots, 0.9, 1\}$．高斯噪声的标准差“sd”代表了噪声等级．

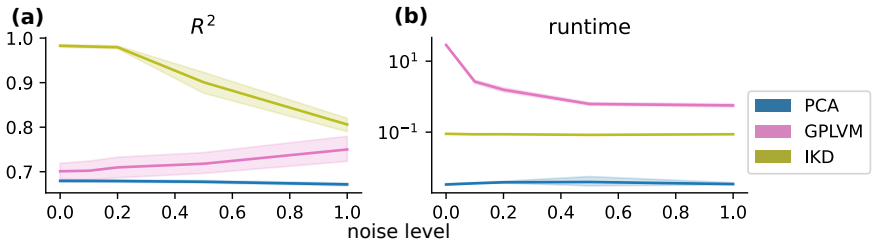

图 5: (a): R^2 performances of PCA, GPLVM, and IKD on datasets with different noise levels. (b): The corresponding running time.

图 5 表明 IKD 在噪声较低的情况下总能估计得很好．IKD 的 R^2 随噪声等级的增加逐渐减小，但仍然在直至 $\mathrm{sd} = 1$ 的时候还比 GPLVM 好．传统的基于优化方法的 GPLVM 求解算法显式的考虑了噪声项，导致性能甚至随噪声增加而有一点点增大．此外，当噪声较大时，GPLVM 运行得甚至更快．

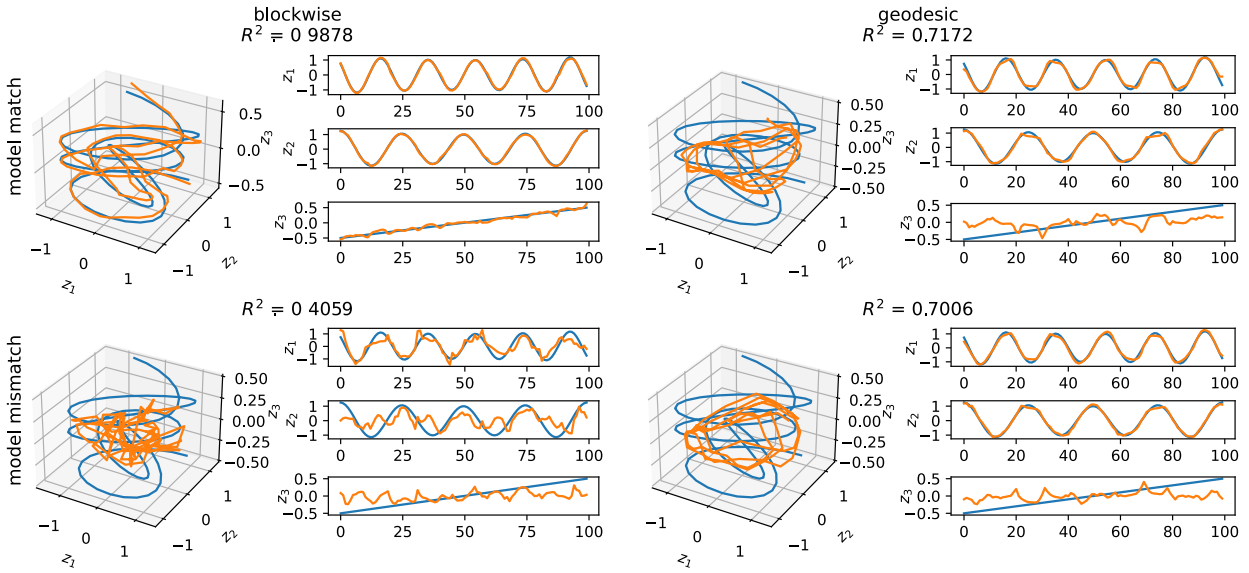

图 6: 分块（左）和测地（右）解在模型匹配（上）和模型不匹配（下）情况下的结果. 蓝色曲线时真实的隐变量，橙色曲线时估计的隐变量. 在模型不匹配情况下，尽管真实生成模型不是 SE 核的 GP，我们还是用一个简单的线性解码器（也就是仿射变换）讲估计的隐变量核真实隐变量对齐，然后计算 R^2.

3.1.3 比较分块解和测地解

为了理解分块解和测地解的区别，我们还是在中等难度数据集上（图 3）分别跑 IKD 的分块解和测地解，并画出这俩结果. 有两组观测，一组从 SE 核来，另一组从 rational quadratic 核来. 但求解这两组观测值时只用 SE 核，也就是模型匹配和模型不匹配两种情况.

在模型匹配情况，分块解能解决 GPLVM，但测地解不能. 因此在模型匹配情况下，当数据点不多的时候，Bron-Kerbosh 算法能很块完成，分块解就可以很快收敛到真实隐变量（最多差一个仿射变换，图 6 中的模型匹配情况）.

在模型不匹配情况，团合并过程就比较脆弱，容易受到数值误差的影响. 并且小误差可能会在合并过程中累计，尤其是当团特别多的时候. 如果团很多，而点又很分散，可能就需要找很多团，导致 Bron-Kerbosch 算法的时间过长. 此外，我们其实也不知道真实的数据生成模型是不是 SE 核的 GP（其实很有可能不是，就像 PCA 中我们就知道数据生成模型肯定不是线性那么简单一样）. 在这种情况下，我们就把所有数据点视为一个整体，然后用测地最短路径计算点与点之间的经验相关性，所以此时用测地法更直观，更全局稳定，也更有效（图 6 的模型不匹配情况）.

3.2 真实世界数据

数据集. 这里我们在四个真实世界数据集上比较 IKD 和其它方法:

• 单细胞 qPCR (PRC) [Guo et al., 2010]: 单细胞 48 个基因在 10 个不同阶段的规范化的测量数据. 总共有 437 个数据点，数据集就是 $\boldsymbol{X} \in \mathbb{R}^{437\times 48}$.

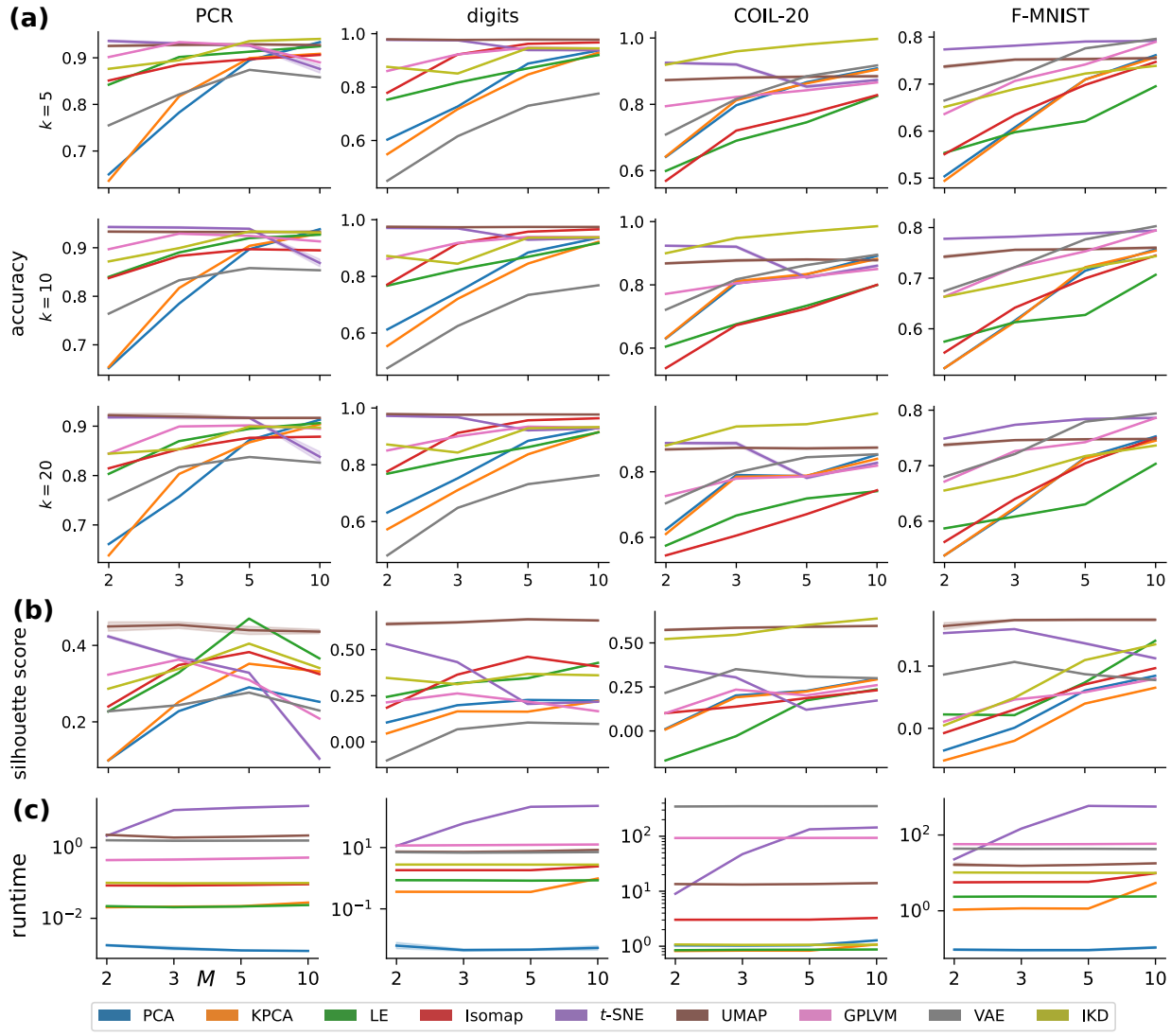

图 7: (a) k-NN 5 折交叉验证、(b) silhouette score 以及 (c) 运行时间，在不同方法、不同隐变量维度 M、不同数据集以及不同选择 k 上的结果.

• 手写体数字 (digits) [Dua and Graff, 2017]：包括 1797 个数字手写体的灰度图．每个都是一个 8×8 的图像，数据集就是 $\boldsymbol{X} \in \mathbb{R}^{1797 \times 64}$.

• COIL-20 [Nene et al., 1996]：包括 1440 个灰度照片．一共有 20 个物体，每个物体有 72 个角度的照片．每个照片都是 128×128 图像，数据集就是 $\boldsymbol{X} \in \mathbb{R}^{1440 \times 16384}$.

• Fashion MNIST (F-MNIST) [Xiao et al., 2017]：包括 10 个流行物品（衣服、包等）的 70000 个灰度图．我们用其中的一个子集，数据集就是 $\boldsymbol{X} \in \mathbb{R}^{3000 \times 784}$.

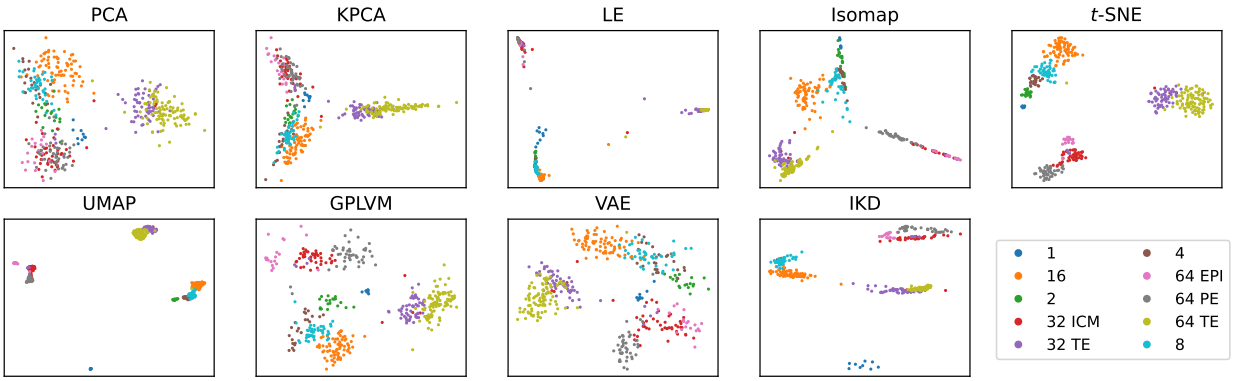

图 8: 不同方法在 PCR 数据集上的降维结果.

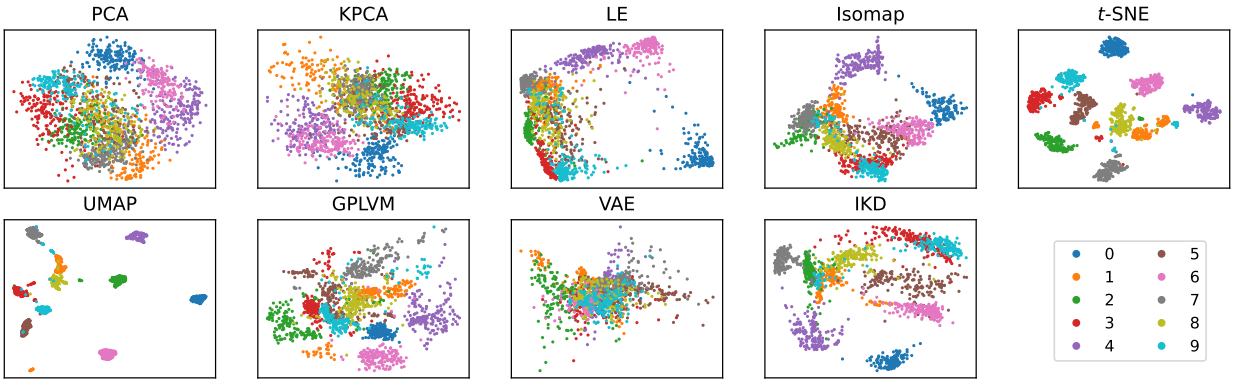

图 9: 不同方法在 digits 数据集上的降维结果.

评价方法. 由于没有真实隐变量可供比较,我们先在 $\{2,3,5,10\}$ 维隐变量空间中估计隐变量,然后用 k-近邻 (k-NN) 分类器来评价不同降维方法的效果. 具体来说, 我们用 5 折交叉验证 k-NN ($k \in \{5,10,20\}$) 在估计的 $\{2,3,5,10\}$ 维隐变量上, 来评估每个方法在每个数据集上的效果. k-NN 在不同方法、不同隐变量维度 M、不同数据集以及不同 k 的选择上的分类结果如图 7(a) 所示. 不同方法在不同数据集上的 silhouette score 在图 7(b) 中.

结果 对比 IKD 和其它特征分解方法（PCA、KPCA、LE、Isomap），我们可以得出结论 IKD 几乎在所有四个数据集上表现都是最好的，除了在 digits 数据集上 $M \in \{3,5,10\}$ 时，Isomap 比 IKD 好. 讲 IKD 和 GPLVM 比较，我发现 GPLVM 在 PCR、digits 和 F-MNIST 数据集上比 IKD 稍微好点，不过 GPLVM 花的时间非常长. 特别的，在 COIL-20 数据集上，IKD 比 GPLVM 显著的好，而在其它数据集上 IKD 只比 GPLVM 差一点点. VAE 只在最复杂的 F-MNIST 上表现得较好，在另外三个数据集上明显不如 IKD. 尽管 IKD 不如剩下两个基于优化的方法（t-SNE 和 UMAP），IKD 还是在 COIL-20 数据集上最好. 原因是 COIL-20 数据集的观测维度非常高（$N = 16384$），而正如合成数据集的结果（图 2(a)）所示，IKD 在高维问题上效果特别好.

四个数据集的 2D 降维可视化结果分别在图 8、9、10 和 11 中. 定性看，可以看出在所有四个数据集上，IKD 一致地比其它基于特征分解的方法和两个基于优化的方法（GPLVM 和 VAE）讲不同的

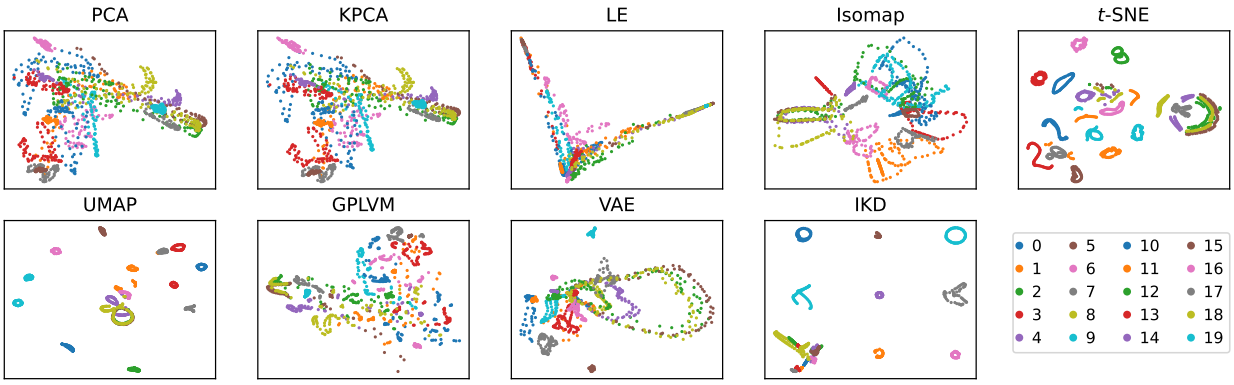

图 10: 不同方法在 COIL-20 数据集上的降维结果.

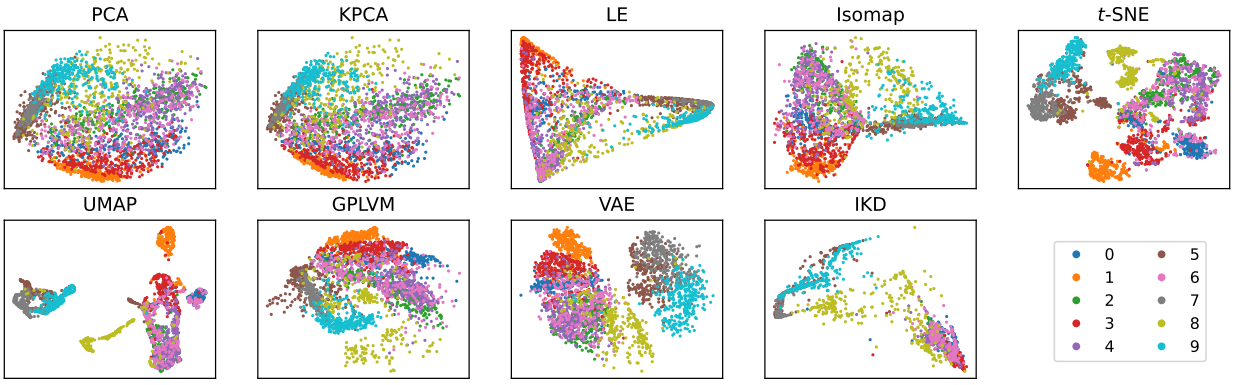

图 11: 不同方法在 F-MNIST 数据集上的降维结果.

簇分得更开. 因此, 尽管 IKD 是基于特征分解的方法, 其在四个真实世界数据集上的表现比其它基于特征分解的方法显著的好, 有时还比基于优化的方法好.

从运行时间看（图 7(b)), IKD 和 Isomap 以及其它基于特征分解的方法显著比其它基于优化的方法快. 注意到如果目标隐变量维度 $M > 2$, 那么 t-SNE 的运行时间仅是勉强可以接受. 而像 COIL-20 这种高维数据集, VAE 和 GPLVM 的运行时间非常高, 超出了对应坐标轴的上限.

不同核的 IKD 降维结果. 正如正文公式 10 中推导的一样, 选不同的核基本不影响降维结果. 区别主要在于不同核的形状. 尤其是在分类任务上, 尽管不同核估计的隐变量结果不完全一致, 同属于一个类别的点基本还是在隐变量空间（低维空间）中类似的聚在一起. 图 12 表明不同核的降维结果（在翻转、旋转和平移后）基本一样. 表 2 表明 k-NN 的 5 折交叉验证分类精度在不同核下都差不多.

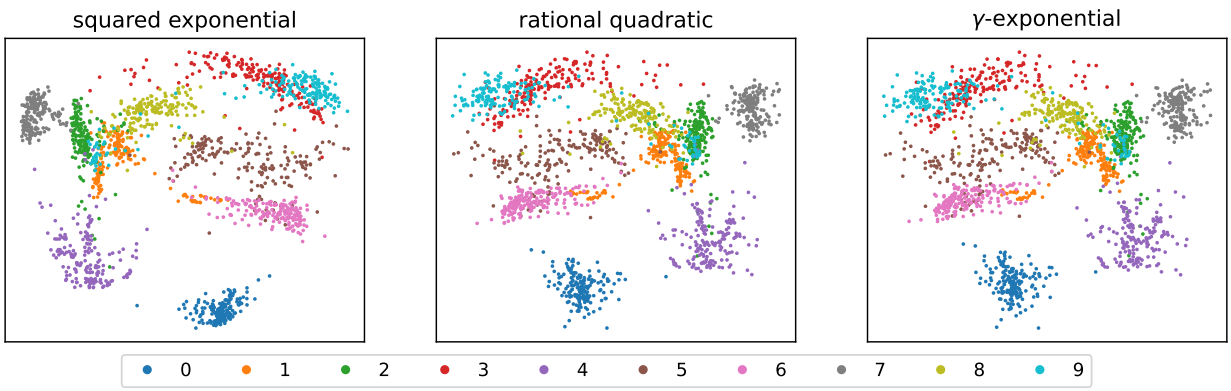

图 12: IKD 在 digits 数据集上选择不同的核的降维（$M = 2$）结果. rational quadratic 核中 $\alpha = 1$, γ-exponential 核中 $\gamma = 1$.

表 2: 5-NN 5-fold cross-validation classification accuracies under different kernels and latent dimensionalities $M \in \{2, 3, 5, 10\}$ for the digits dataset.

k	kernels	2	3	5	10
5	squared exponential	0.875899	0.85085	0.946049	0.944937
	rational quadratic	0.841382	0.821323	0.931574	0.935474
	γ-exponential	0.837478	0.806288	0.930458	0.933807
10	squared exponential	0.872006	0.844732	0.936592	0.937696
	rational quadratic	0.857527	0.825235	0.92212	0.943258
	γ-exponential	0.854737	0.81242	0.919336	0.93992
20	squared exponential	0.871453	0.843067	0.928804	0.932683
	rational quadratic	0.857521	0.822467	0.906541	0.929341
	γ-exponential	0.856408	0.817457	0.908767	0.928231

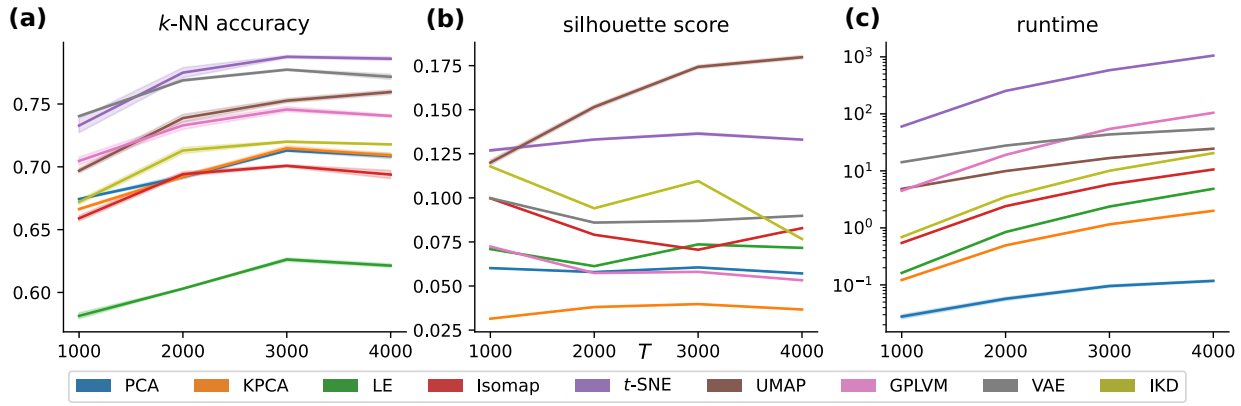

图 13: 不同方法的 (a) k-NN 5 折交叉验证的平均结果、(b) silhouette score 以及 (c) 运行时间，在 F-MNIST 数据集上关于 T 的变化.

变化样本数 T. 尽管 IKD 的时间复杂度的理论结果关于样本数 T 已经在 2.4 节中分析了，且在 3.1 中讨论过了，我们还是想看一下变化样本数 T 不同方法会怎么样. 因此，我们在 F-MNIST 数据集上尝试不同样本数 $T \in \{1000, 2000, 3000, 4000\}$. k-NN 准确率和 silhouette scores 的结果在图 13(a) 和 (b) 中. 结果表明，IKD 一致得比其它基于特征分解的方法好，除了 silhouette score 在 $T = 4000$ 时. 从运行时间看，IKD 比其它基于优化的方法快. 然而，和 UMAP 以及 VAE 比较 IKD 的运行时间的增长率，从图 13(c) 中可以看出，由于 (1) IKD 和其它基于特征分解方法的较差的关于 T 的时间复杂度，以及 (2) 基于优化的方法能够应对大规模数据集，样本量 T 对 IKD 和其它基于特张分解的方法的影响，会随着 T 的增加越来越大.

4 讨论

总结来说，IKD 作为基于特征分解的方法，以较少的运行时间，就能够得到比其它基于特征分解方法更好的降维结果. 当面对高维观测数据时，IKD 明显比其它所有方法都快都好.

尽管基于特征分解的方法不如基于优化的方法，但却能快速得到一个结果，作为那些复杂非线性优化问题的初始化，从而减轻 GPLVM 和 VAE 这种方法中常遇到的数值不稳定问题和多峰问题.

参考文献

Anqi Wu, Nicholas A. Roy, Stephen Keeley, and Jonathan W Pillow. Gaussian process based nonlinear latent structure discovery in multivariate spike train data. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/hash/b3b4d2dbedc99fe843fd3dedb02f086f-Abstract.html.

Anqi Wu, Stan Pashkovski, Sandeep R Datta, and Jonathan W Pillow. Learning a latent manifold of odor representations from neural responses in piriform cortex. *Advances in Neural Information Processing Systems*, 31, 2018.

Ehsan Sheybani and Giti Javidi. Dimensionality reduction and noise removal in wireless sensor network datasets. In *2009 Second International Conference on Computer and Electrical Engineering*, volume 2, pages 674–677. IEEE, 2009.

Mayur R Bakrania, I Jonathan Rae, Andrew P Walsh, Daniel Verscharen, and Andy W Smith. Using dimensionality reduction and clustering techniques to classify space plasma regimes. *Frontiers in Astronomy and Space Sciences*, page 80, 2020.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008a.

Mark A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2):233–243, 1991. ISSN 1547-5905. doi: 10.1002/aic.690370209. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/aic.690370209. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/aic.690370209.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of machine learning research*, 9(11), 2008b.

Leland McInnes, John Healy, and James Melville. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *arXiv:1802.03426 [cs, stat]*, September 2020. URL http://arxiv.org/abs/1802.03426. arXiv: 1802.03426.

Neil Lawrence. Gaussian Process Latent Variable Models for Visualisation of High Dimensional Data. In *Advances in Neural Information Processing Systems*, volume 16. MIT Press, 2003. URL https://proceedings.neurips.cc/paper/2003/hash/9657c1fffd38824e5ab0472e022e577e-Abstract.html.

Neil Lawrence. Probabilistic Non-linear Principal Component Analysis with Gaussian Process Latent Variable Models. *Journal of machine learning research*, 6(11), 2005.

Thang D Bui and Richard E Turner. Stochastic variational inference for Gaussian process latent variable models using back constraints. In *Black Box Learning and Inference NIPS workshop*, 2015.

Jack Wang, Aaron Hertzmann, and David J Fleet. Gaussian Process Dynamical Models. In *Advances in Neural Information Processing Systems*, volume 18. MIT Press, 2005. URL `https://proceedings.neurips.cc/paper/2005/hash/ccd45007df44dd0f12098f486e7e8a0f-Abstract.html`.

Jack M. Wang, David J. Fleet, and Aaron Hertzmann. Gaussian Process Dynamical Models for Human Motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):283–298, 2008. ISSN 1939-3539. doi: 10.1109/TPAMI.2007.1167. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

R. Urtasun, D.J. Fleet, and P. Fua. 3D People Tracking with Gaussian Process Dynamical Models. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 238–245, June 2006. doi: 10.1109/CVPR.2006.15. ISSN: 1063-6919.

Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Kernel principal component analysis. In Wulfram Gerstner, Alain Germond, Martin Hasler, and Jean-Daniel Nicoud, editors, *Artificial Neural Networks —ICANN'97*, Lecture Notes in Computer Science, pages 583–588, Berlin, Heidelberg, 1997. Springer. ISBN 978-3-540-69620-9. doi: 10.1007/BFb0020217.

Joshua B Tenenbaum, Vin de Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

Ingwer Borg and Patrick JF Groenen. *Modern multidimensional scaling: Theory and applications*. Springer Science & Business Media, 2005.

Achiya Dax et al. Low-rank positive approximants of symmetric matrices. *Advances in Linear Algebra & Matrix Theory*, 4(03):172, 2014.

Coen Bron and Joep Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *Communications of the ACM*, 16(9):575–577, September 1973. ISSN 0001-0782, 1557-7317. doi: 10.1145/362342.362367. URL `https://dl.acm.org/doi/10.1145/362342.362367`.

Edsger W Dijkstra et al. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

Joseph B Kruskal and Myron Wish. *Multidimensional scaling*. Number 11. Sage, 1978.

Mikhail Belkin and Partha Niyogi. Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. *Neural Computation*, 15(6):1373–1396, June 2003. ISSN 0899-7667. doi: 10.1162/089976603321780317. Conference Name: Neural Computation.

Leland McInnes, John Healy, Nathaniel Saul, and Lukas Grossberger. Umap: Uniform manifold approximation and projection. *The Journal of Open Source Software*, 3(29):861, 2018.

GPy. GPy: A gaussian process framework in python. `http://github.com/SheffieldML/GPy`, since 2012.

Guoji Guo, Mikael Huss, Guo Qing Tong, Chaoyang Wang, Li Li Sun, Neil D Clarke, and Paul Robson. Resolution of cell fate decisions revealed by single-cell gene expression analysis from zygote to blastocyst. *Developmental cell*, 18(4):675–685, 2010.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Sameer A Nene, Shree K Nayar, Hiroshi Murase, et al. Columbia object image library (coil-100). 1996.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.