MATH858D

AN INTRODUCTION TO DATA ANALYSIS

MARIA CAMERON

Contents

1. Principal Component Analysis

Principal Component Analysis (PCA) is a widely used tool for data analysis in both natural and social sciences. Besides, it is a building block for many methods for data analysis introduced later. A classic reference for the PCA is the book by I. T. Jolliffe [1] available online. Here I give just a brief overview.

Let $\eta \in \mathbb{R}^D$ be a vector random variable. Let

$$X = \begin{bmatrix} x_1^\top & \to \\ \vdots & \\ x_N^\top & \to \end{bmatrix}$$

be an $N \times D$ matrix of samples of η. The goal of the PCA is to map the samples of η from the high-dimensional space $\mathbb{R}^D$ into a low-dimensional space $\mathbb{R}^d$, while retaining as much variation present in the samples as possible. This is achieved by calculating the singular value decomposition (SVD) of the centered sample matrix whose column means are equal to zero. Recall that an SVD of an $m \times n$, $m \geq n$, matrix A is a decomposition of the

form $A = U\Sigma V^\top$ where U is orthogonal $m \times n$, i.e., $U^\top U = I$, Σ is diagonal $n \times n$ with $\sigma_1 \geq \ldots \geq \sigma_n \geq 0$, and V is orthogonal $n \times n$. The columns of U are called left singular vectors, the diagonal entries of Σ are called singular values, and the columns of V are called the right singular vectors. If $m < n$, we consider such a decomposition for $X^\top$. Next we will derive the formulas for the principal components and understand why computing SVD is helpful.

1.1. **Derivation.** We would like to replace η with a random variable $\xi \in \mathbb{R}^d$ whose components are linear combinations of the components of η,

$$\xi = (w_1^\top \eta, \ldots, w_d^\top \eta)$$

where the coefficients w_i, $i = 1, \ldots, d$ maximize the variance $\mathrm{Var}(w_i^\top \eta)$ while satisfy the constraints

$$w_i^\top w_i = 1, \quad i = 1, \ldots, d, \quad \text{and} \quad \mathrm{Cov}(w_i\eta, w_j\eta) = 0.$$

The condition $\mathrm{Cov}(w_i\eta, w_j\eta) = 0$ means that the components of ξ should be uncorrelated.

Let us first find PCA 1, $\xi_1 = w_1\eta$ using Lagrange multipliers. Let

$$C := \mathrm{Cov}(\eta) = (E[(\eta_i - E[\eta_i])(\eta_j - E[\eta_j])])_{i,j=1}^D$$

be the covariance matrix of η. It is $D \times D$, symmetric positive definite provided that all components of η have nonzero variances. Then

$$\mathrm{Var}(\xi_1) = E[(w_1^\top \eta - w_1^\top E[\eta])^2] = w_1^\top C w_1.$$

The Lagrange function is given by

$$L(w, \lambda) = w_1^\top C w_1 - \lambda(w_1^\top w_1 - 1),$$

where λ is the Lagrange multiplier. Differentiating L with respect to w_1 we get

$$Cw_1 - \lambda w_1 = (C - I\lambda)w_1 = 0.$$

Hence w_1 must be an eigenvector of C corresponding to the eigenvalue λ. To decide which eigenvalue we pick, we recall that we are maximizing

$$w_1^\top C w_1 = \lambda w_1^\top w = \lambda.$$

Hence, we pick λ_1, the largest eigenvalue of C, and the corresponding eigenvector w_1.

Next we will look for w_2. The zero covariance condition gives:

$$\mathrm{Cov}(w_1\eta, w_2\eta) = w_1^\top C w_2 = \lambda_1 w_1^\top w_2 = 0.$$

Hence w_2 must be orthogonal to w_1. The Lagrangian function is

$$L(w_2, \lambda, \phi) = w_2^\top C w_2 - \lambda(w_2^\top w_2 - 1) - \phi w_1^\top w_2.$$

Differentiating it with respect to w_2 we get:

$$Cw_2 - \lambda w_2 - \phi w_1 = (C - I\lambda)w_2 - \phi w_1 = 0.$$

Taking a dot product of this equation with w_1 we get: $0 - 0 - \phi = 0$ which forces ϕ to be zero. Hence, λ and w_2 must be an eigenpair corresponding to the second largest eigenvalue of C.

Proceeding in a similar manner, we find that w_k is the eigenvector of C corresponding to its kth largest eigenvalue.

The variables $\xi_i = w_i^\top \eta$ where w_i, $i = 1, \ldots, d$ are the eigenvectors of the covariance matrix C corresponding to the d largest eigenvalues, are called *the principal components*. The vector w_i, $i = 1, \ldots, d$, is called *the vector of coefficients or loadings for the ith principal component*.

1.2. **Calculation in practice.** In practice, when we are dealing with data, the data points $x_i \in \mathbb{R}^D$ are interpreted as samples of a vector random variable. The covariance matrix is not known. To approximate it, we compute the $D \times D$ data covariance matrix. First we need to center the data so that column means are all zero:

$$Y := X - \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \left[\frac{1}{N} \sum_{i=1}^{N} x_{i1}, \ldots, \frac{1}{N} \sum_{i=1}^{N} x_{iD} \right].$$

Then the covariance matrix is given by

$$C := \frac{1}{N} Y^\top Y.$$

Its eigendecomposition with eigenvalues ordered in the decreasing order is

$$C = W \Lambda W^\top, \quad \text{where} \quad \Lambda = \operatorname{diag}\{\lambda_1, \ldots, \lambda_D\}$$

and $\lambda_1 \geq \ldots \geq \lambda_D \geq 0$. The d eigenvectors corresponding to the d largest eigenvalues will be the desired loadings. The coordinates of the data points in the principal component space will be $z_i := Y w_i$, $i = 1, \ldots, d$.

Now let us connect the PCA with the SVD. First we consider the case $N > D$. Let

$$Y = U \Sigma W^\top$$

be an SVD of Y. Then

$$Y^\top Y = W \Sigma^2 W^\top \equiv NC = NW \Lambda W^\top.$$

Hence the first d columns of $W = [w_1, \ldots, w_D]$ are the vectors of coefficients for the first d principal components, and the principal components are $Y w_i$

Now suppose that $D > N$. Then

$$Y^\top = U \Sigma V^\top, \quad Y^\top Y = U \Sigma^2 U^\top.$$

The first d columns of $U = [u_1, \ldots u_D]$ are the vectors of coefficients for the first d principal components, and and the principal components are $Y u_i$. Note that in the case where $D > N$, the matrix $Y^\top Y$ is larger than Y and your computer might run out of memory if you try to calculate it, while the calculation of the SVD would not create a problem.

2. Multidimensional scaling

Multidimensional scaling (MDS)[2] aims at finding a low-dimensional representation of data given an $N \times N$ matrix Δ of squared distances between the data points. As we will show below, if the distances are Euclidean, MDS is equivalent to PCA. However, in many applications coming from social sciences (see examples in [2]) or in medicine (e.g., see [3]) the distances are non-Euclidean.

The classic multidimensional scaling algorithm is the following. Let d be the desired output dimension.

(1) Compute row means of the matrix Δ:

$$\mu_i := \frac{1}{N}\sum_{j=1}^{N} \Delta_{ij}, \quad i = 1, \dots, N.$$

(2) Set the mean of row means:

$$\mu := \frac{1}{N}\sum_{i=1}^{N} \mu_i.$$

(3) Define the $N \times N$ matrix $B = (B_{ij})$ by

$$B_{ij} := \frac{1}{2}\left(\mu_i + \mu_j - \Delta_{ij} - \mu\right).$$

(4) Compute the eigendecomposition of B, $B = V\Lambda V^\top$ where the eigenvalues λ_i are ordered in the decreasing order. If the d largest eigenvalues are positive, take the corresponding eigenpairs (v_i, λ_i), $i = 1, \dots, d$. Otherwise, take the set of m_+ eigenpairs corresponding to positive eigenvalues. Set $d_+ := \min\{m_+, d\}$ where m_+ is the number of positive eigenvalues.

(5) Define the embedding of the dataset into $\mathbb{R}^{d_+}$ by

$$Z = \left[v_1, \dots v_{d_+}\right] \begin{bmatrix} \sqrt{\lambda_1} & & \\ & \ddots & \\ & & \sqrt{\lambda_{d_+}} \end{bmatrix}.$$

The matrix Z is $N \times d_+$. Row i of Z corresponds to the embedding of the ith data point into $\mathbb{R}^{d_+}$.

If the distance is Euclidean, the MDS is equivalent to the PCA. Indeed, let Y be an $N \times D$ matrix of data centered so that the column means are zeros. Then

$$\Delta_{ij} = \sum_{k=1}^{D}(y_{ki} - y_{kj})^2 = \sum_{k=1}^{D} y_{ki}^2 + \sum_{k=1}^{D} y_{kj}^2 - 2\sum_{k=1}^{D} y_{ki}y_{kj}.$$

Hence

$$(YY^\top)_{ij} = \frac{1}{2}\left(\sum_{k=1}^{D} y_{ki}^2 + \sum_{k=1}^{D} y_{kj}^2 - (\Delta)_{ij}\right)$$

are the matrix elements of $YY^\top$. Let us show that the matrix elements of the matrix B defined in Step (3) of the MDS coincide with those of $YY^\top$. First we expand the row means of B:

$$\begin{aligned}\mu_i &= \frac{1}{N}\sum_{j=1}^{N}\Delta_{ij} = \frac{1}{N}\sum_{j=1}^{N}\sum_{k=1}^{D}(y_{ki}-y_{kj})^2 \\ &= \sum_{k=1}^{D} y_{ki}^2 + \frac{1}{N}\sum_{j=1}^{N}\sum_{k=1}^{D} y_{kj}^2 - 2\frac{1}{N}\sum_{k=1}^{D} y_{ki}\sum_{j=1}^{N} y_{kj} \\ &= \sum_{k=1}^{D} y_{ki}^2 + \frac{1}{N}\sum_{j=1}^{N}\sum_{k=1}^{D} y_{kj}^2.\end{aligned}$$

Here we have used the fact that the column means of Y are zeros. Now we expand μ:

$$\begin{aligned}\mu &= \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{k=1}^{D} y_{ki}^2 + \frac{1}{N}\sum_{j=1}^{N}\sum_{k=1}^{D} y_{kj}^2\right) \\ &= \frac{2}{N}\sum_{i=1}^{N}\sum_{k=1}^{D} y_{ki}^2.\end{aligned}$$

Plugging in these expressions for μ_i, μ_j, and μ into the formula for B_{ij} we get:

$$\begin{aligned}B_{ij} &= \frac{1}{2}\left(\mu_i+\mu_j-\Delta_{ij}-\mu\right) \\ &= \frac{1}{2}\left(\sum_{k=1}^{D} y_{ki}^2 + \frac{1}{N}\sum_{l=1}^{N}\sum_{k=1}^{D} y_{kl}^2 + \sum_{k=1}^{D} y_{kj}^2 + \frac{1}{N}\sum_{l=1}^{N}\sum_{k=1}^{D} y_{kl}^2 - \frac{2}{N}\sum_{l=1}^{N}\sum_{k=1}^{D} y_{kl}^2 - (\Delta)_{ij}\right) \\ &= \frac{1}{2}\left(\sum_{k=1}^{D} y_{ki}^2 + \sum_{k=1}^{D} y_{kj}^2 - (\Delta)_{ij}\right) = (YY^\top)_{ij}.\end{aligned}$$

Now, if

$$Y = P\Sigma Q^\top$$

is an SVD of Y, then

$$YY^\top = P\Sigma^2 P^\top,$$

hence $\Lambda = \Sigma^2$ and $V = P$. Now,

$$Y[q_1,\dots q_d] = P\Sigma Q^\top[q_1,\dots q_d] = [p_1,\dots,p_d]\begin{bmatrix}\sqrt{\lambda_1} & & \\ & \ddots & \\ & & \sqrt{\lambda_{d_+}}\end{bmatrix},$$

which coincides with the output of the MDS.

Remark If the squared distances forming the matrix Δ are non-Euclidean, the matrix B might have some negative eigenvalues.

Remark If the number of data points is large, the computation of the MDS becomes expensive: its cost scales as $O(N^2)$. To fight this issue the so-called landmark multidimensional scaling (LMDS) has been introduced [4] with cost $O(nN)$ where $n \ll N$ is the number of landmarks.

3. Diffusion maps

While the PCA is a power tool when the data points lie near a d-dimensional hyperplane in $\mathbb{R}^D$, it might fail to give a nice embedding if the data are instead located near some d-dimensional curved manifold. To handle this case, Coifman and Lafon (Yale University, 2006) introduced the so-called *diffusion maps* [5]. The key idea of this approach is to devise a discrete-time Markov chain on the data points and define the distances between remote points using the stochastic matrix of this Markov chain. This approach is robust to noisy data and is capable of adequately representing complex geometries of data structures. This dimensional reduction technique has been successfully applied to problems arising in protein dynamics (e.g. [7, 8]). This approach is not a universal solution. One of the implementational problems consists in that it requires to provide a scaling parameter ϵ a reasonable value for which might be hard-to-find. Nonetheless, it is a beautiful idea worth going over. One approach to tackle the problem of choosing ϵ was proposed by M. Maggioni and collaborators and will be discussed in Section 4.

3.1. Calculation of a diffusion map.

Here we present the most basic diffusion map algorithm corresponding to $\alpha = 0$ in [5]. Let $X = (x_{ik})$ be an $N \times D$ matrix of data. The rows x_i, $i = 1, \dots, N$, of X represent data points lying in $\mathbb{R}^D$.

- First we compute the squared-distance matrix between the data points:

$$\Delta(i,j) = \sum_{k=1}^{D} (x_{ik} - x_{jk})^2.$$

- Next, we need to pick the scaling parameter ϵ and define the diffusion kernel, an $N \times N$ matrix $K = (k_{ij})$ where

$$k_{ij} = \exp\left(-\frac{\Delta(i,j)}{\epsilon}\right).$$

  A good choice of ϵ is very important. ϵ should be comparable to squared distances from the data points to their neighbors. To get an idea what ϵ should be, look as a row of the matrix Δ to get an initial guess for ϵ and then tune it experimentally.
- Convert the diffusion kernel K into a stochastic matrix $P = (p_{ij})$ by normalizing the rows of K:

$$P = Q^{-1}K \quad \text{where} \quad Q := \operatorname{diag}\left\{\sum_{j=1}^{N} k_{1j}, \dots, \sum_{j=1}^{N} k_{Nj}\right\} := \operatorname{diag}\{q_1, \dots, q_N\}.$$

Note that the Markov chain with the stochastic matrix P is time-reversible, and the diagonal entries of Q constitute the invariant probability measure: check that

$$[q_1, \ldots, q_N]Q^{-1}K = [q_1, \ldots, q_N].$$

The invariant probability distribution in this Markov chain is

$$\pi = \frac{q}{\sum_{i=1}^{N} q_i} \quad \text{where} \quad q := [q_1, \ldots, q_N].$$

- Recall that the entries p_{ij}^t of the tth power of the stochastic matrix P are the probabilities to transition from i to j in t steps, $t \in \mathbb{N}$. A family of *diffusion distances* indexed by $t \in \mathbb{N}$ is defined by

$$D_t(x_i, x_j) := \sum_{m=1}^{N} \frac{1}{\pi_m} \left| p_{im}^t - p_{jm}^t \right|^2. \tag{1}$$

The diffusion distance $D_t(x_i, x_j)$ between the data points x_i and x_j is small if the transition probabilities from x_i and x_j to all other data points in the defined diffusion process are close.

- The family of *diffusion maps* $\Psi_t : \mathbb{R}^D \to \mathbb{R}^{N-1}$ indexed by $t \in \mathbb{N}$ from the data space $\mathbb{R}^D$ to the diffusion space $\mathbb{R}^{N-1}$ is defined so that the squared Euclidean distances $\|\Psi_t(x_i) - \Psi_t(x_j)\|$ in the diffusion space are equal to the diffusion distances $D_t(x_i, x_j)$. The diffusion map Ψ_t is defined by:

$$\Psi_t(x_i) := \begin{bmatrix} \lambda_1^t r_1(i) \\ \vdots \\ \lambda_{N-1}^t r_{N-1}(i) \end{bmatrix}, \tag{2}$$

where $R := [r_0, r_1, \ldots, r_{N-1}]$ is the matrix of right eigenvectors of P normalized so that $R^\top \Pi R = I$, and $1 = \lambda_0 > \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_{N-1}$ are the ordered eigenvalues of P. Respectively, λ_m^t, is the tth power of λ_m, $m = 1, \ldots, N-1$. Note that since P is irreducible and aperiodic (as $P_{ii} > 0$, $i = 1, \ldots, N$) by construction, $\lambda_0 = 1$ and $r_0 = [1, \ldots, 1]^\top$. In Eq. (2), $r_m(i)$ denotes the ith entry of the vector r_k. The fact that one can normalize R so that $R^\top \Pi R = I$ will be shown within the proof of the proposition below.

Proposition 1.

$$D_t(x_i, x_j) = \sum_{m=1}^{N-1} \lambda_m^{2t} |r_m(i) - r_m(j)|^2,$$

i.e., the diffusion distance in the data space equals the Euclidean distance in the diffusion space.

We will prove this proposition after we finish the description of the construction.

- The diffusion maps allow us to do dimensional reduction by keeping only the first few components of $\Psi_t(\cdot)$. Often it is desirable to keep only the first two or three entries of $\Psi_t(\cdot)$ as then the diffusion map is readily visualizable. Thus, we can define diffusion maps (abusing the term) to 2D or 3D diffusion spaces by

$$\Psi_t(x_i) = \begin{bmatrix} \lambda_1^t r_1(i) \\ \lambda_2^t r_2(i) \end{bmatrix} \quad \text{and} \quad \Psi_t(x_i) = \begin{bmatrix} \lambda_1^t r_1(i) \\ \lambda_2^t r_2(i) \\ \lambda_3^t r_3(i) \end{bmatrix}. \tag{3}$$

Now let us prove Proposition 1.

Proof. Let us redefine the diffusion kernel K as

$$K \to \left(\sum_{i=1}^{N} q_i \right)^{-1} K.$$

Then the stochastic matrix P with the new K can be decomposed as

$$P = \Pi^{-1} K.$$

P is similar to the symmetric matrix

$$A := \Pi^{1/2} P \Pi^{-1/2} = \Pi^{1/2} \Pi^{-1} K \Pi^{-1/2} = \Pi^{-1/2} K \Pi^{-1/2}.$$

Hence, the eigenvalues of A coincide with those of P. Let

$$A = \Phi \Lambda \Phi^\top = \sum_{n=0}^{N-1} \lambda_n \phi_n \phi_n^\top.$$

be an eigendecomposition of A where Φ is orthogonal, and the diagonal entries of Λ, the eigenvalues, are ordered in the decreasing order. Then the desired eigendecomposition of P can be obtained as follows:

$$P = \Pi^{-1/2} A \Pi^{1/2} = \Pi^{-1/2} \Phi \Lambda \Phi^\top \Pi^{1/2} =: R \Lambda L = \sum_{n=0}^{N-1} \lambda_n r_n l_n, \tag{4}$$

where $r_n := \Pi^{-1/2} \phi_n$, the columns of R, are the right eigenvectors of P, and $l_n := \phi_n^\top \Pi^{1/2}$, the rows of L, are the left eigenvectors of P. It can be readily verified that the left and right eigenvectors satisfy the following conjugacy relationships:

$$\sum_{m=1}^{n} \pi_m r_i(m) r_j(m) = r_i^\top \Pi r_j = \phi_i^\top \Pi^{-1/2} \Pi \Pi^{-1/2} \phi_j = \phi_i \phi_j = \delta_{i,j}, \tag{5}$$

$$\sum_{m=1}^{n} \frac{l_i(m) l_j(m)}{\pi_m} = l_i \Pi^{-1} l_j^\top = \phi_i^\top \Pi^{1/2} \Pi^{-1} \Pi^{1/2} \phi = \phi_i \phi_j = \delta_{i,j}. \tag{6}$$

Eq. (4) allows us to write entries of P^t as

$$P_{im}^t = \sum_{n=0}^{N-1} \lambda_n^t r_n(i) l_n(m). \tag{7}$$

Plugging P^t_{im} and P^t_{jm} into the formula for $D_t(i,j)$ we get:

$$\begin{aligned} D_t(i,j) &= \sum_{m=1}^{N} \frac{1}{\pi_m} \sum_{n=0}^{N-1} \left[\lambda_n^t r_n(i) l_n(m) - \lambda_n^t r_n(j) l_n(m)\right]^2 \\ &= \sum_{n=0}^{N-1} \lambda_n^{2t} \left[r_n(i) - r_n(j)\right]^2 \sum_{m=1}^{N} \frac{[l_n(m)]^2}{\pi_m}. \end{aligned}$$

Eq. (6) implies that the inner sum in the last expression is 1. Furthermore, since $r_0 = [1,\ldots,1]^\top$, $r_0(i) - r_0(j) = 0$. Therefore,

$$D_t(i,j) = \sum_{n=1}^{N-1} \lambda_n^{2t} \left[r_n(i) - r_n(j)\right]^2$$

as desired. □

3.2. **Illustrative examples.** Let us consider a data set consisting of 200 images depicting the Pacman. This example is similar to the one in [6][1]. Each image is 65×65 pixels either black (color = 0) or white (color = 255). The images differ from each other by the angle of rotation of the Pacman around the center of the image. The angles of rotation are drawn from the uniform distribution on $(0, 2\pi)$. A sample of 20 such images is shown in Fig. 1(a). This dataset is naturally embedded into $\mathbb{R}^{65^2} = \mathbb{R}^{4225}$ space. Note that $D > N$ in this case. The PCA mapping into 2D applied to this dataset is shown in Fig. 1(b). The diffusion maps with $\epsilon = 5 \cdot 10^7$ and $t = 1$ into 2D and 3D diffusion spaces are shown in Figs. 1(c) and (d) respectively. Both the PCA and the diffusion mappings show that the set of images is well-approximated by a 1D manifold homeomorphic to the circle as we would expect.

A similar example with a more complex image of the Cat-in-the-hat is shown in Fig. 2. Each image is 500×500. The double-loop formed by the mapped data is caused by the fact that the image rotated by π is closer to the original image than, say the one rotated by an angle between $\pi/6$ and $5\pi/6$.

4. Multiscale SVD

While the diffusion maps approach to data analysis is an appealing idea, the choice of the scaling parameter ϵ might be very challenging in problems associated with data coming from molecular simulation. Questions about choosing ϵ asked by chemical physicists inspired Prof. M. Maggioni[2] (JHU) to develop a routine procedure for the determination of the scaling parameter. The method A. Little, M. Maggioni , and L. Rosasco came up with kills two birds with one shot [9, 10]: it determines the dimension of the manifold near which noisy data are located and gives a range of good values of the scaling parameter that can be

[1]While Ref. [6] offers a nice exposition, I do not recommend to rely on it as it contains a number of errors in important formulas. For example, Eqs. (6), (7) contain errors, the comments following Eq. (9) are misleading, etc.

[2]M. Maggioni said this in the CSCAMM seminar on Nov. 30, 2016.

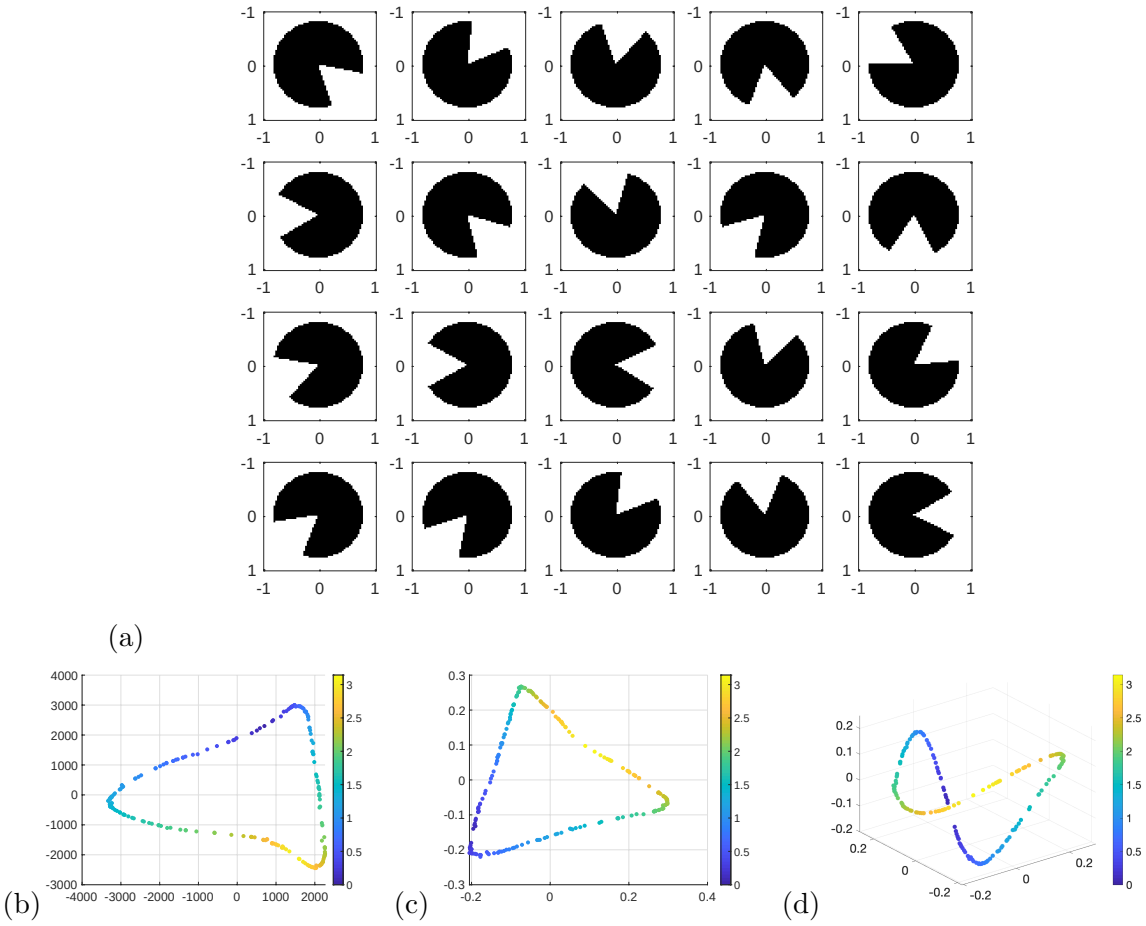
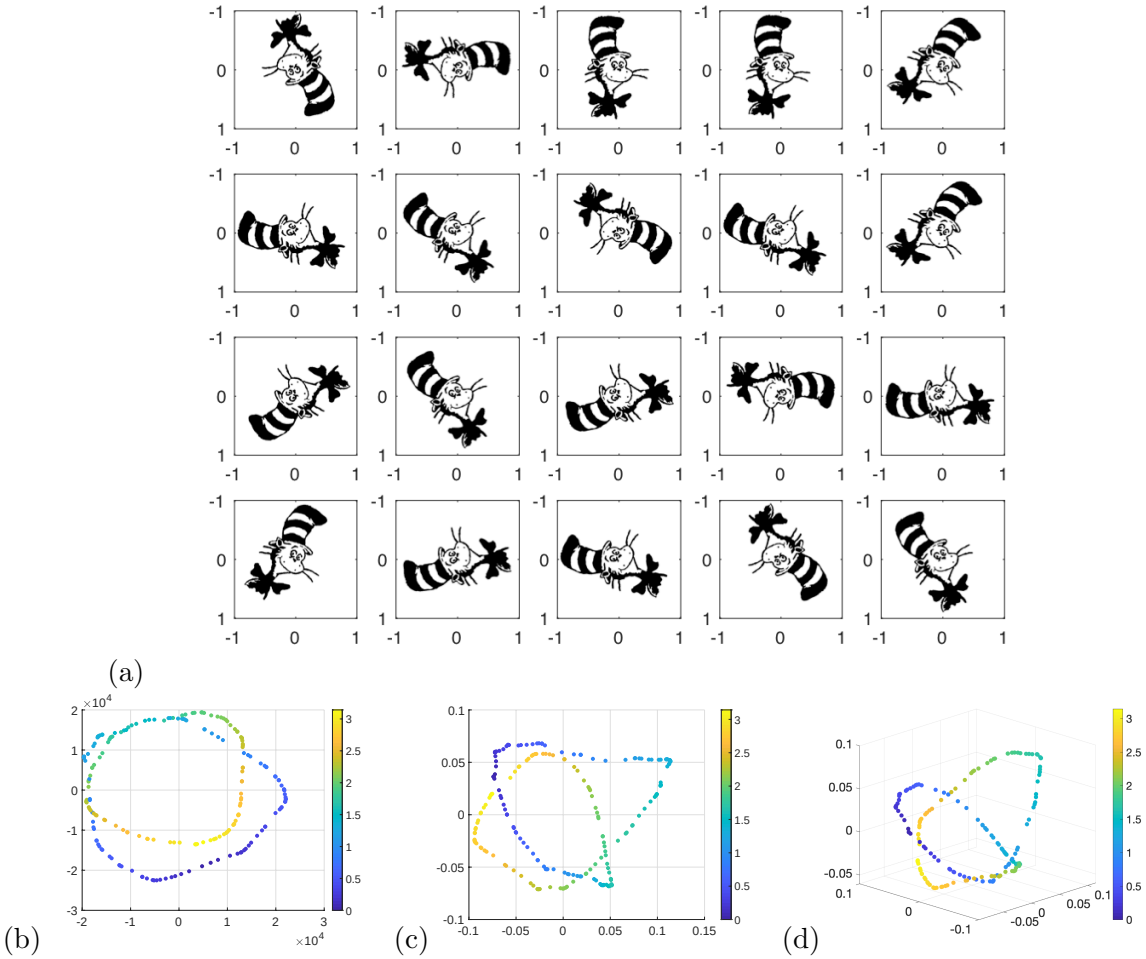


FIGURE 1. The dataset consists of 200 images of the Pacman rotated around the center of the image by angles uniformly distributed in $(0, 2\pi)$. (a): A sample of 20 data points. (b): The PCA mapping into 2D. (c) and (d): Diffusion maps to 2D and 3D respectively.

used for diffusion maps. Below I will outline the method. Please read [9] for more details. Although this method requires a number of hard-to-check-in-advance assumptions to be satisfied, this is the only method I am aware of that addresses the problem of determination of intrinsic dimension of noisy dataset and finding the appropriate scaling parameter.

Suppose we are given a dataset X embedded into $\mathbb{R}^D$ where D is large. $X = (x_{ij})$ is an $N \times D$ matrix, i.e., each row of X represents a data point in $\mathbb{R}^D$. The rows x_i of X can be interpreted as samples of a vector random variable ξ with D components. Let



(a) (b) (c) (d)

FIGURE 2. The dataset consists of 200 images of the Cat-in-the-hat rotated around the center of the image by angles uniformly distributed in $(0, 2\pi)$. (a): A sample of 20 data points. (b): The PCA mapping into 2D. (c) and (d): Diffusion maps to 2D and 3D respectively.

$\mu = [\mu_1, \ldots, \mu_D]$ be the vector of column means of X, i.e., $\mu \approx E[\xi]$:

$$\mu_j = \frac{1}{N} \sum_{i=1}^{N} x_{ij}, \quad j = 1, \ldots, N.$$

The covariance matrix of η is estimated by the covariance of the dataset:

$$\mathrm{Cov}(\xi) \approx C := \mathrm{Cov}(X) = \frac{1}{N}(X - e\mu)^\top (X - e\mu), \quad \text{a } D \times D \text{ matrix},$$

where e is $N \times 1$ array of ones. Let $\lambda_1 \geq \ldots \geq \lambda_D$ be the eigenvalues of C.

- If the rows x_i, $i = 1, \ldots, N$, lie at a d-dimensional hyperplane, then the first d eigenvalues of C will be nonzero, while the rest $D - d$ eigenvalues will be zeros. In this case, the standard PCA will give an excellent embedding into $\mathbb{R}^d$.
- If $x_i = y_i + \sigma\eta_i$ where η_i are i.i.d. RVs with with mean 0 and variance 1, σ is small enough, and y_i lie at a d-dimensional hyperplane, then the first d eigenvalues of C will be approximately equal to variances of y_i along the d principle axes of the spread of y_i's in the hyperplane, while the rest $D - d$ eigenvalues will be approximately equal to σ. Therefore, the PCA will correctly identify the dimension if the noiseless dataset Y provided that the covariance matrix $\mathrm{Cov}(Y)$ has all nonzero eigenvalues larger than σ^2.
- Now suppose that $x_i = y_i + \sigma\eta_i$ where η_i and σ are as described in the previous item, while y_i belong to a d-dimensional curved manifold $\mathcal{M}$. In this case, the number of nonzero eigenvalues of $\mathrm{Cov}(Y)$ will be equal to the minimal space dimension where the manifold $\mathcal{M}$ can be embedded into, which is larger than d.

The key idea of the multiscale SVD as a tool for identifying the true dimensions of such manifolds $\mathcal{M}$ is that $\mathcal{M}$ can be well-approximated by a hyperplane in a small enough neighborhood of each its point. Hence, if we consider a ball of radius r surrounding a point $y \in \mathcal{M}$, where r is small enough so that $B_r(y) \cap \mathcal{M}$ is close to a hyperplane and, at the same time, r is notably greater than σ, the size of the noise, then the eigenvalues of C corresponding to the true dimension of $\mathcal{M}$ will scale as r^2, while the ones related to the curvature of $\mathcal{M}$ will scale as r^4. Here $B_r(y)$ denotes the ball of radius r centered at y. Finally, the eigenvalues corresponding to noise will be approximately equal to σ^2.

Ref. [9] states a number of assumptions on the geometric properties of the given dataset X as well as a theorem validating the MSVD (multiscale SVD) algorithm outlined below.

Algorithm: MSVD
Input: Dataset matrix X, $N \times D$.
Parameters provided by user:
• $N_{repeat} \leq N$ is the number of points of X around which we consider sequences of $B_r(x_i)$.
• The set of values of r: $r_1 < \ldots < r_{N_r}$, the radii of the neighborhoods.
Output: The set of mean singular values (averaged over N_{repeat} points of X) as functions of r, i.e., an $D \times N_r$ matrix MSV.

The main body
for $k = 1, \ldots, N_{repeat}$
(1) Pick a random index $i \in \{1, \ldots, N\}$;
(2) Calculate the vector of distances from x_i to all points in X;
for $m = 1, \ldots, N_r$
(3) Find the subset of points in X lying in the ball $B_{r_m}(x_i)$: $X_1 := X \cap B_{r_m}(x_i)$;
(4) Find column means of X_1 and calculate the covariance matrix C of X_1;

(5) Find eigenvalues of C, sort them in the decreasing order and take their square roots. These square roots are the singular values of the centered sample X_1 (i.e., column means are subtracted from each column of X_1). The corresponding Matlab command is:

$$\texttt{SV(k,m,:)} = \texttt{sqrt(sort(real(eig(C)),'descend'));}$$

end for
end for
(6) Average the singular values over all k's. the corresponding Matlab command is:

$$\texttt{MSV} = \texttt{squeeze(mean(SV,1));}$$

As the mean singular values are found, the user is supposed to plot them as functions of r and identify those that are approximately equal to σ, the ones that grow linearly with r, and the ones that grow quadratically with r. The first group corresponds to noise, the second group corresponds to the true dimension of $\mathcal{M}$, and the third group corresponds to the curvature of $\mathcal{M}$. The range of good parameter values for the scaling coefficient ϵ in the diffusion maps is the range of values of r where these three groups are the most distinct.

Let us consider an application of the MSVD algorithm to an example discussed in [9]. The dataset X is generated as follows. We pick $N = 1000$, $D = 100$, $\sigma = 0.1$ and set $X = Y + \sigma Z$, where Z is an $N \times D$ matrix of samples of i.i.d. Gaussian RVs with mean 0 and variance 1, and $Y = [Y_1, Y_2]$ is an $N \times D$ matrix where Y_1 is an $N \times 10$ matrix each row of which is a sample of the RV uniformly distributed on the unit 9-dimensional sphere $\mathbb{S}_9$ embedded into $\mathbb{R}^{10}$, and all entries of Y_2 are zeros. Here I used $N_{repeat} = 1000$ and r ranging from 1.1 to 2.2 with step 0.01.

The mean singular values, i.e., $E[\lambda^{1/2}(C)]$ plotted versus r are shown in Fig. 3. The green shaded region indicates the range of good values for the scaling coefficient ϵ.

5. Data Assimilation

Data assimilation is an iterative approach to the problem of estimating the state of a system in-hand given both current and past observations and a model for the time evolution of the system. This approach is widely used in meteorology (e.g. weather prediction), finance (e.g. stock prices prediction), geophysics (e.g. prediction for the magnetic field of the Earth). There are two different problems that are addressed in the framework of data assimilation: (i) t*he smoothing problem*, when the goal is to estimate the whole sequence of past states of the system up to the present, and (ii) *the filtering problem*, when the goal is to assess the current state. In this course, we will focus only on the filtering problem. This review is mostly based on [11] which is available online via arXiv:1506.07825.

5.1. **Filtering problem.**

We will consider the following discrete time filtering problem. Suppose the system we are interested in is governed by the following model giving its time

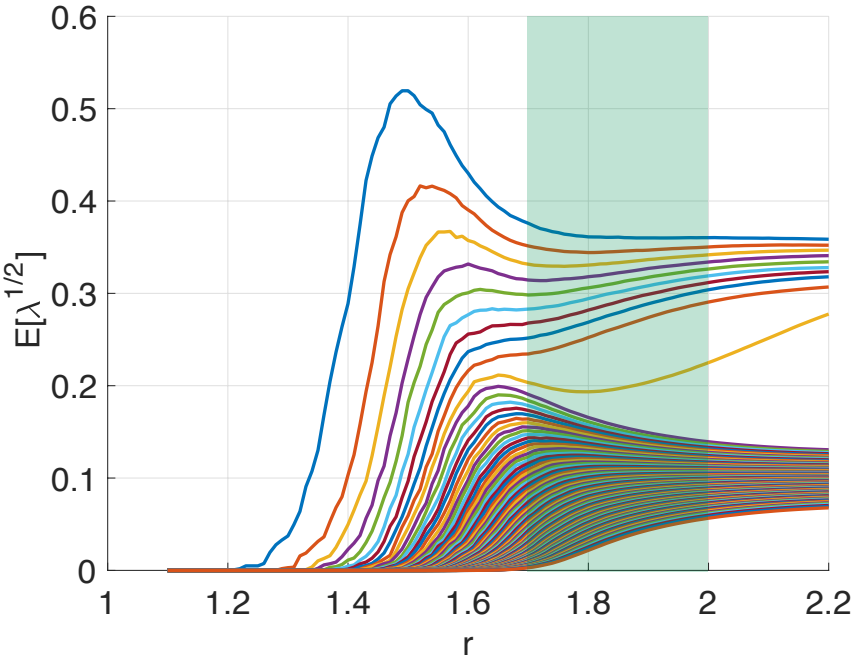


FIGURE 3. The output of the MSVD algorithm applied to a noisy dataset lying near $\mathbb{S}_9$ embedded into $\mathbb{R}^{100}$. The noise size is $\sigma = 0.1$. This example is reproduced from [9].

evolution:

$$v_{j+1} = \Psi(v_j) + \xi_j, \quad v_j, \xi_j \in \mathbb{R}^n, \quad j = 0, 1, 2, \ldots, \tag{8}$$
$$v_0 \sim \mathcal{N}(m_0, C_0),$$
$$\xi_j \sim \mathcal{N}(0, \Sigma) \text{ are i.i.d. RVs independent of } v_0;$$
$$y_{j+1} = h(v_{j+1}) + \eta_{j+1}, \quad y_j, \eta_j \in \mathbb{R}^m, \ m \le n, \quad j = 0, 1, 2, \ldots, \tag{9}$$
$$\eta_j \sim \mathcal{N}(0, \Gamma) \text{ are i.i.d. RVs independent of } v_0 \text{ and } \xi_j\text{'s}.$$

Here v_j's are the states of the system at times $j \in \mathbb{N} \cup \{0\} =: \mathbb{Z}_+$ that we do not know but want to estimate. We have a pdf $\mathcal{N}(m_0, C_0)$ for the initial state v_0, where m_0 is the mean and C_0 is an $n \times n$ covariance matrix. We have a model (1) for the time evolution of v that is generally stochastic. To make it deterministic, just set the covariance matrix Σ to zero. Besides, we collect observations y_j's at times $j \in \mathbb{N}$ that depend on the current state of the system and are corrupted by noise. In Eq. (9), $h : \mathbb{R}^n \to \mathbb{R}^m$ is the observation operator. For example, y can be the first component of v, then $h(v) = [1, 0, \ldots, 0]v = v_1$. The task is to find

$$E[v_{j+1} \mid Y_{j+1}] \quad \text{and} \quad \mathrm{Cov}(v_{j+1} \mid Y_{j+1}), \text{ where } Y_{j+1} = \{y_1, \ldots, y_{j+1}\}. \tag{10}$$

The assumption that v_0, ξ_j's, and η_j's are Gaussian are made here for simplicity. They can be relaxed for some filters.

Filtering is a two-step iterative process.

(1) **Prediction.** Find the conditional pdf $f(v_{j+1}|Y_j)$ given $f(v_j|Y_j)$.
(2) **Analysis.** Find $f(v_{j+1}|Y_{j+1})$ given $f(v_{j+1}|Y_j)$, i.e., use the new piece of information y_{j+1} to update $f(v_{j+1}|Y_j)$.

Step 1 is accomplished by using the model update (8). In Step 2, y_{j+1} is incorporated via Bayesian estimation. Various filters differ in how steps 1 and 2 are performed. Prior to introduce them, we will go over Bayesian formulas.

5.2. **Bayesian estimation.**

5.2.1. *Discrete setting.* Bayes's theorems allow us to make corrections to our a priori estimations based on new information. They might seem trivial because their proofs are very simple, however, they are very important. Their power is demonstrated by the following problem from D. Kahneman's book "Thinking Fast and Slow" [12]. In this book, D. Kahneman shows that humans are poor intuitive statisticians.

`A cab was involved in a hit-and-run accident at night. Two cab companies, the Green and the Blue, operate in the city. You are given the following data:`

- `85% of the cabs in the city are Green and 15% are Blue.`
- `A witness identified the cab as Blue. The court tested the reliability of the witness under the circumstances that existed on the night of the accident and concluded that the witness correctly identified each one of the two colors 80% of the time and failed 20% of the time.`

`What is the probability that the cab involved in the accident was Blue rather than Green?`

In order to solve this problem correctly we need to use Bayes's theorem. First consider two events A and B with $P(A) \neq 0$ and $P(B) \neq 0$. Then recall that

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \text{ and } P(B|A) = \frac{P(A \cap B)}{P(A)}.$$

Hence

$$\boxed{P(B|A) = \frac{P(A|B)P(B)}{P(A)}.} \tag{11}$$

This is the first form of Bayes's theorem.

Now suppose that we are not given the probability of A explicitly. However, the set of outcomes Ω can be partitioned into a finite or countable number of disjoint subsets Z_i:

$$\Omega = \bigcup_i Z_i, \ Z_i \cap Z_j = \emptyset.$$

Suppose $P(Z_i) \neq 0$ for all i. And, for each i, we know the conditional probability $P(A|Z_i)$. Then we can express $P(A)$ as

$$P(A) = \sum_i P(A \cap Z_i) = \sum_i \frac{P(A \cap Z_i)}{P(Z_i)} P(Z_i) = \sum_i P(A|Z_i)P(Z_i). \tag{12}$$

Using Eq. (11) for Z_j and A and Eq. (12) we obtain

$$\boxed{P(Z_j|A) = \frac{P(A|Z_j)P(Z_j)}{\sum_i P(A|Z_i)P(Z_i)}.} \tag{13}$$

Eq. (13) is the second form of Bayes's theorem.

Now we will solve the problem above using Eq. (13). The set of outcomes Ω is the set of all taxis in the city. It is partitioned into two subsets B (Blue) and G (Green). We are given: $P(B) = 0.15$, $P(G) = 0.85$. If only the first piece of information would be available, i.e., if there would be no witness, the best answer we could give would be that the probability that a Blue cab was responsible for the crime is 15%.

Now let us incorporate the evidence provided by the witness. Intuitively, many people, while reading such a police report, neglect the base rates 15% and 85% and jump to the conclusion that a Blue cab is responsible with probability 80%. However, it is wrong, because there are much more Green cabs that could be misidentified as Blue rather than vice verse.

Denote the event that the witness saw a Blue cab by WB. We want to find the conditional probability

$$P(B|WB), \text{ the cab was Blue provided that the witness saw a Blue cab.}$$

Bayes's theorem in form 1 is not directly applicable because we do not know $P(WB)$. However, we do know from the police experiment that $P(WB|B) = 0.8$, $P(WB|G) = 0.2$. Substituting $j = 1$, $i = 1, 2$, $WB \to A$, $B \to Z_1$, $G \to Z_2$ into Eq. (13) we calculate

$$P(B|WB) = \frac{P(WB|B)P(B)}{P(WB|B)P(B) + P(WB|G)P(G)} = \frac{0.8 \cdot 0.15}{0.8 \cdot 0.15 + 0.2 \cdot 0.85} \approx 0.41.$$

Thus, the probability the cab involved into the accident was Blue rather than Green is about 41%.

5.2.2. *Continuous setting.* Recall that the conditional pdf is defined by

$$f(x|y) = \frac{f(x,y)}{f(y)}.$$

Hence, the continuous analog of the first form of Bayes's theorem is

$$\boxed{f(y|x) = \frac{f(x|y)f(y)}{f(x)}.} \tag{14}$$

Also note that, similarly to the discrete case, $f(x)$ can be expressed via a conditional pdf of another RV Z:

$$f(x) = \int_{\mathbb{R}^n} f(x|z)f(z)dz. \tag{15}$$

5.3. **Filtering: prediction and analysis steps.** Now we are quipped to elaborate on the prediction and analysis steps.

(1) **Prediction.**

$$f(v_{j+1}|Y_j) = \int_{\mathbb{R}^n} f(v_{j+1}|Y_j, v_j)f(v_j|Y_j)dv_j = \int_{\mathbb{R}^n} f(v_{j+1}|v_j)f(v_j|Y_j)dv_j. \tag{16}$$

Here we have used Eq. (15) and the fact that $f(v_{j+1}|Y_j, v_j) = f(v_{j+1}|v_j)$. The latter is true because Y_j contains an indirect information on v_j and cannot improve upon perfect knowledge of v_j. The pdf $f(v_{j+1}|v_j)$ is determined by the model equation .

(2) **Analysis.** Here we will use Bayes's formula (14).

$$\begin{aligned} f(v_{j+1}|Y_{j+1}) = f(v_{j+1}|Y_j, y_{j+1}) &= \frac{f(y_{j+1}|v_{j+1}, Y_j)f(v_{j+1}|Y_j)}{f(y_{j+1}|Y_j)} \\ &= \frac{f(y_{j+1}|v_{j+1})f(v_{j+1}|Y_j)}{f(y_{j+1}|Y_j)}. \end{aligned} \tag{17}$$

The pdf $f(y_{j+1}|v_{j+1})$ is determined by the observation equation (9).

5.4. **Linear Gaussian Problems: the Kalman Filter.** Kalman Filter was introduced by R. Kalman in 1960 [13]. It is applicable only for the case where $\Psi(v) = Mv$, $h(v) = Hv$, where M in an $n \times n$ matrix and and H is an $m \times n$ matrix, and and ξ_j, η_j, v_0 are Gaussian RVs. More importantly, the Kalman filter serves as a building block for many other filtering algorithms that do not require linearity or Gaussianity.

Let us restate the settings for the linear Gaussian case.

$$\begin{aligned} &v_{j+1} = Mv_j + \xi_j, \quad v_j, \xi \in \mathbb{R}^n, \quad j \in \mathbb{Z}_+, \\ &v_0 \sim \mathcal{N}(m_0, C_0), \\ &\xi_j \sim \mathcal{N}(0, \Sigma) \text{ are i.i.d. RVs independent of } v_0; \end{aligned} \tag{18}$$

$$\begin{aligned} &y_{j+1} = Hv_{j+1} + \eta_{j+1}, \quad y_j, \eta_j \in \mathbb{R}^n, \ m \le n, \quad j\mathbb{Z}_+, \\ &\eta_j \sim \mathcal{N}(0, \Gamma) \text{ are i.i.d. RVs independent of } v_0 \text{ and } \xi_j\text{'s}. \end{aligned} \tag{19}$$

We assume that $\mathrm{Rank}(H) = m$. It can be proven that all v_j's are Gaussian. A simple proof of this fact involves characteristic functions. A characteristic function of a RV η is defined by $\mathsf{cf}(z) := E[\exp(iz\eta)]$ where i is the imaginary unit. It can be proven that a characteristic function uniquely determines the distribution of a RV. The proof of the fact that v_j are Gaussian is done by induction. The induction step is proved by showing using the characteristic function that a linear transform of a Gaussian RV is Gaussian and a sum of Gaussian RVs is Gaussian (see [11]).

We will use the following notations:

$$
\begin{aligned}
&m_j := E[v_j|Y_j], && C_j := \mathrm{Cov}(v_j|Y_j),\\
&\hat{m}_{j+1} := E[v_{j+1}|Y_j], && \hat{C}_{j+1} := \mathrm{Cov}(v_{j+1}|Y_j),\\
&m_{j+1} := E[v_{j+1}|Y_{j+1}], && C_{j+1} := \mathrm{Cov}(v_{j+1}|Y_{j+1}).
\end{aligned} \tag{20}
$$

Note that the mean and covariance completely define a Gaussian RV as the pdf of $\eta \sim \mathcal{N}(m, C)$ is

$$
f_\eta(x) = \frac{1}{(2\pi)^{n/2}|\det(C)|^{1/2}} e^{-\frac{1}{2}(x-m)^\top C^{-1}(x-m)}.
$$

We adopt the standard assumption that the covariance matrices C_0, Γ, and Σ are symmetric positive definite (SPD). We will show that all C_j's are SPD by induction and derive the formulas for the map

$$
(m_j, C_j) \mapsto (m_{j+1}, C_{j+1})
$$

that is central to the Kalman filter.

The prediction step is done by

$$
v_{j+1} = Mv_j + \xi_j.
$$

Hence

$$
\hat{m}_{j+1} = E[v_{j+1}|Y_j] = E[Mv_{j+1}|Y_j] + E[\xi_j|Y_j] = Mm_j,
$$

as ξ_j has mean 0 and is independent of Y_j. For the covariance $\hat{C}_{j+1}$ we have:

$$
\begin{aligned}
\hat{C}_{j+1} &= E[(v_{j+1} - \hat{m}_{j+1})(v_{j+1} - \hat{m}_{j+1})^\top|Y_j]\\
&= E[M(v_j - m_j)(v_j - \hat{m}_j)^\top M^\top|Y_j] + E[\xi_j\xi_j^\top|Y_j]\\
&+ E[M(v_j - m_j)\xi_j^\top|Y_j] + E[\xi_j(v_j - \hat{m}_j)^\top M^\top|Y_j]\\
&= MC_jM^\top + \Sigma.
\end{aligned}
$$

The two terms in the third line of this derivation are zeros as ξ_j is independent of Y_j and v_j. Note that $\hat{C}_{j+1} = MC_jM^\top + \Sigma$ is SPD as it is a sum of two SPD matrices. The matrix C_j is SPD by induction assumption.

Next comes the analysis step. By Eq. (17) and the fact that v_{j+1} is Gaussian, we have:

$$
\begin{aligned}
&\exp\left\{-\tfrac{1}{2}(v_{j+1} - m_{j+1})^\top C_{j+1}^{-1}(v_{j+1} - m_{j+1})\right\} \propto\\
&\exp\left\{-\tfrac{1}{2}(y_{j+1} - Hv_{j+1})^\top \Gamma^{-1}(y_{j+1} - Hv_{j+1})\right\} \exp\left\{-\tfrac{1}{2}(v_{j+1} - \hat{m}_{j+1})^\top \hat{C}_{j+1}^{-1}(v_{j+1} - \hat{m}_{j+1})\right\}.
\end{aligned}
$$

Equating the coefficients in the quadratic polynomial in v_{j+1} we get:

$$
C_{j+1}^{-1} = \hat{C}_{j+1}^{-1} + H^\top\Gamma^{-1}H = \left(MC_jM^\top + \Sigma\right)^{-1} + H^\top\Gamma^{-1}H, \tag{21}
$$

$$
C_{j+1}^{-1}m_{j+1} = H^\top\Gamma^{-1}y_{j+1} + \hat{C}_{j+1}^{-1}\hat{m}_{j+1} = H^\top\Gamma^{-1}y_{j+1} + \left(MC_jM^\top + \Sigma\right)^{-1} Mm_j. \tag{22}
$$

Eq. (21) implies that C_{j+1} is SPD as C_{j+1}^{-1} is a sum of two SPD matrices.

Eqs. (21) and (22) allow us to establish a computational algorithm for computing the map $(m_j, C_j) \mapsto (m_{j+1}, C_{j+1})$. Let us express m_{j+1} and C_{j+1} via $\hat{m}_{j+1}$ and $\hat{C}_{j+1}$. We will use the following matrix identity.

Lemma 1. Woodbury Matrix Identity *Let $A \in \mathbb{R}^{p\times p}$, $U \in \mathbb{R}^{p\times q}$, $C \in \mathbb{R}^{q\times q}$, and $V \in \mathbb{R}^{q\times p}$. If A and C are SPD then $A+UCV$ is invertible and*

$$(A+UCV)^{-1} = A^{-1} - A^{-1}U\left(C^{-1}+VA^{-1}U\right)^{-1}VA^{-1}. \tag{23}$$

This lemma can be proven by direct verification. Multiply $A+UCV$ by the right-hand side and you will get I.

Eq.(23) gives the following expression for C_{j+1}:

$$\begin{aligned} C_{j+1} &= \left(\hat{C}_{j+1}^{-1} + H^\top\Gamma^{-1}H\right)^{-1} \\ &= \hat{C}_{j+1} - \hat{C}_{j+1}H^\top\left(\Gamma + H\hat{C}_{j+1}H^\top\right)^{-1}H\hat{C}_{j+1} \\ &= \left(I - \hat{C}_{j+1}H^\top\left(\Gamma + H\hat{C}_{j+1}H^\top\right)^{-1}H\right)\hat{C}_{j+1} \\ &= (I-KH)\,\hat{C}_{j+1}, \quad \text{where} \qquad (24) \\ K &:= \hat{C}_{j+1}H^\top\left(\Gamma + H\hat{C}_{j+1}H^\top\right)^{-1} \text{ is called Kalman gain.} \end{aligned}$$

Multiplying Eq.(22) by C_{j+1} we get:

$$\begin{aligned} m_{j+1} &= (I-KH)\,\hat{C}_{j+1}\left(H^\top\Gamma^{-1}y_{j+1} + \hat{C}_{j+1}^{-1}\hat{m}_{j+1}\right) \\ &= (I-KH)\,\hat{m}_{j+1} + C_{j+1}H^\top\Gamma^{-1}y_{j+1}. \qquad (25) \end{aligned}$$

To further simplify the expression for m_{j+1}, we note that Eq. (24) implies that

$$KH = I - C_{j+1}\hat{C}_{j+1}^{-1}.$$

On the other hand, Eq. (21) implies that

$$C_{j+1}H^\top\Gamma^{-1}H = I - C_{j+1}\hat{C}_{j+1}^{-1} = KH.$$

Since H is full rank, we conclude that

$$K = C_{j+1}H^\top\Gamma^{-1}. \tag{26}$$

Plugging this into Eq. (25) we get:

$$m_{j+1} = \hat{m}_{j+1} + K(y_{j+1} - H\hat{m}_{j+1}). \tag{27}$$

Now we are ready to write out the Kalman filter algorithm. We omit the subscripts in $\hat{m}$ and $\hat{C}$ because we recompute them at every step from scratch and do not store them.

Kalman filter

for $j = 0, 1, 2, \ldots$ **do**

Prediction step

a. $\hat{m} = Mm_j$

b. $\hat{C} = MC_jM^\top + \Sigma$

Analysis step

c. $d = y_{j+1} - H\hat{m}$ \\ Kalman innovation

d. $K = \hat{C}H^\top \left(\Gamma + H\hat{C}H^\top\right)^{-1}$ \\ Kalman gain

e. $m_{j+1} = \hat{m}_{j+1} + Kd$ \\ estimator update

f. $C_{j+1} = (I - KH)\,\hat{C}$ \\ covariance update

end for

The Kalman filter, while applicable in its authentic form only for linear Gaussian problems, has been adjusted for nonlinear problems as well. Two such adjustments called "3DVAR" and "Extended Kalman Filter (ExKF)" are described in [11]. 3DVAR is based on fixing the model covariance $\hat{C}_{j+1}$. The ExKF is based on linearization of Ψ. We are going to skip these adjustments and discuss other adjustments based on the introduction of an ensemble of particles with the goal of estimation of the mean and the covariance $\hat{m}_{j+1}$ and $\hat{C}_{j+1}$.

5.5. **Ensemble Kalman Filters.** Here we consider two variants of Ensemble Kalman Filter, a basic one denoted EnKF, and the so-called "Square Root Ensemble Kalman Filter" or "Ensemble Transform Kalman Filter (ETKF)".

5.5.1. *Basic Ensemble Kalman Filter (EnKF).* **Ensemble Kalman filter (EnKF)**

Setup

Pick N, the number of particles in the ensemble.

Initialization

$U = [U_1, \dots, U_N]$ is $n \times N$ matrix with the positions of ensemble members.

for $i = 1, \dots, N$ **do**

$U_i = m_0 + C_0^{1/2}\xi_i$, where $\xi_i \sim \mathcal{N}(0,1)$

end for

Main body

for $j = 0, 1, 2, \dots$ **do**

Prediction step: evolve the ensemble and estimate its mean and covariance

a. $\hat{U} = \Psi(U) + \Sigma^{1/2}\zeta$ where ζ is $n \times N$ matrix of i.i.d. RVs $\zeta_{ij} \sim \mathcal{N}(0,1)$ \\ ensemble predict

b. $\hat{m} = \frac{1}{N}\sum_{i=1}^N \hat{U}_i$ \\ estimator predict

c. $\hat{C} = \frac{1}{N-1}\sum_{i=1}^N (\hat{U}_i - \hat{m})(\hat{U}_i - \hat{m})^\top$ \\ covariance predict

Analysis step

d. $d = y_{j+1} + \Gamma^{1/2}\eta - H\hat{U}$ where η is $m \times N$ matrix of i.i.d. RVs $\eta_{ij} \sim \mathcal{N}(0,1)$ \\ innovation

e. $K = \hat{C}H^\top \left(\Gamma + H\hat{C}H^\top\right)^{-1}$ \\ Kalman gain

f. $U = \hat{U} + Kd$ \\ ensemble update

g. $m_{j+1} = \frac{1}{N}\sum_{i=1}^N U_i$ \\ estimator update

f. $C_{j+1} = \frac{1}{N-1}\sum_{i=1}^{N}(U_i - m_{j+1})(U_i - m_{j+1})^\top$ \\ covariance update
end for

5.5.2. *Ensemble Transform Kalman Filter (ETKF).* The key idea of the ETKF is to transform the ensemble so that the Kalman identity

$$C_{j+1} = (I - KH)\hat{C} \tag{28}$$

is enforced. This is done by mapping the predicted mean $\hat{m}$ according to the standard Kalman update and introduce a deterministic linear transformation of the difference between the particles and their mean to enforce Eq. (28). This step is supposed to eliminate the sampling error. More precisely, at each step j, we define

$$\hat{X} = \frac{1}{\sqrt{N-1}}\left[\hat{U}_1 - \hat{m}, \ldots, \hat{U}_N - \hat{m}\right], \text{ then } \hat{C} = \hat{X}\hat{X}^\top,$$

and look for an SPD $N \times N$ matrix T to transform $\hat{X}$ according to $X := \hat{X}T^{1/2}$ so that

$$C_{j+1} := XX^\top = (I - KH)\hat{C} = (I - KH)\hat{X}\hat{X}^\top.$$

Let us show that

$$T := \left[I + \left(H\hat{X}\right)^\top \Gamma^{-1}\left(H\hat{X}\right)\right]^{-1} \tag{29}$$

is a suitable choice. To invert the matrix expression in Eq. (29) we use Woodbury matrix identity (23).

$$\begin{aligned}
XX^\top = \hat{X}T\hat{X}^\top &= \hat{X}\left[I + \left(H\hat{X}\right)^\top \Gamma^{-1}\left(H\hat{X}\right)\right]^{-1}\hat{X}^\top \\
&= \hat{X}\left\{I - \left(H\hat{X}\right)^\top\left[\left(H\hat{X}\right)\left(H\hat{X}\right)^\top + \Gamma\right]^{-1}\left(H\hat{X}\right)\right\}\hat{X}^\top \\
&= \hat{X}\left\{I - \left(H\hat{X}\right)^\top\left[H\hat{C}H^\top + \Gamma\right]^{-1}\left(H\hat{X}\right)\right\}\hat{X}^\top \\
&= \hat{C} - \hat{C}H^\top\left[H\hat{C}H^\top + \Gamma\right]^{-1}H\hat{C} \\
&= (I - KH)\,\hat{C},
\end{aligned}$$

as required.

The ETKF algorithm is summarized below.

Ensemble Transform Kalman filter (ETKF)

Setup

Pick N, the number of particles in the ensemble.

Initialization

$U = [U_1, \ldots, U_N]$ is $n \times N$ matrix with the positions of ensemble members.

for $i = 1, \ldots, N$ **do**

$U_i = m_0 + C_0^{1/2}\xi_i$, where $\xi_i \sim \mathcal{N}(0, 1)$

end for

Main body
for $j = 0, 1, 2, \ldots$ **do**
Prediction step: evolve the ensemble and estimate its mean and covariance
a. $\hat{U} = \Psi(U) + \Sigma^{1/2}\zeta$ where ζ is $n \times N$ matrix of i.i.d. RVs $\zeta_{ij} \sim \mathcal{N}(0,1)$ \\ ensemble predict
b. $\hat{m} = \frac{1}{N}\sum_{i=1}^{N} \hat{U}_i$ \\ estimator predict
c. $\hat{X} = \frac{1}{\sqrt{N-1}}(\hat{U}_i - \hat{m})$
d. $\hat{C} = \hat{X}\hat{X}^\top$ \\ covariance predict
Analysis step
e. $T = \left(I + (H\hat{X})^\top \Gamma^{-1} (H\hat{X})^\top\right)^{-1/2}$ \\ ensemble transform matrix
f. $X = \hat{X}T$ \\ ensemble transform
g. $d = y_{j+1} - H\hat{m}$ \\ innovation
h. $K = \hat{C}H^\top \left(\Gamma + H\hat{C}H^\top\right)^{-1}$ \\ Kalman gain
i. $m_{j+1} = \hat{m} + Kd$ \\ estimator update
j. $U = m_{j+1} + (N-1)^{1/2}X$ \\ ensemble update
k. $C_{j+1} = XX^\top$ \\ covariance update
end for

Experimenting with EnKF and ETKF you can observe that ETKF is notably slower as the ensemble transform requires an inversion of an $N \times N$ matrix.

Remark The further development along these lines is the filter referred to as LETKF (Local Ensemble Transform Kalman Filter) introduced by B. Hunt (UMD), E. Kostelich (U Arizona), and I. Szunyogh (UMD) in 2007 [14]. This is a more sophisticated filter, the best one available so far for attacking problems coming from real-life applications such as weather forecasting to the best of my knowledge. Its consideration lies beyond the scope of our introduction to data assimilation.

5.6. **Particle Filters.** Particle filters are an important class of filters as they are *provably* capable of reproducing the pdf $f(v_{j+1}|Y_{j+1})$ in the large particle limit. This is not true for the ensemble Kalman filters considered here because they treat the distributions $f(v_{j+1}|Y_{j+1})$ as Gaussian while it is not true. On the other hand, particle filters even in their current state-of-art forms are not suitable for geophysical applications such as weather forecasting due to the lack of robustness. In contrast, ensemble Kalman filters are proven to be robust and efficient for problems arising in geophysical applications.

Our excursion into particle filters will include the basic particle filter based on *sequential importance resampling (SIRS)*. I am referring an interested reader to Ref. [11] for the proof that the particle filter approximates $f(v_{j+1}|Y_{j+1})$ in the limit $N \to \infty$, i.e. the number of particles tends to infinity.

Particle filters are used, e.g., in robotic navigation (see Wiki).

In the ensemble-based filters EnKF and ETKF, the pdfs $\mu_j := f(v_j|Y_j)$ and $\hat{\mu} := f(v_{j+1}|Y_j)$ were approximated by those of the ensemble:

$$\mu_j \approx \mu_j^{(N)} := \frac{1}{N}\sum_{i=1}^{N}\delta(x-U_i), \quad \hat{\mu} \approx \hat{\mu}^{(N)} := \frac{1}{N}\sum_{i=1}^{N}\delta(x-\hat{U}_i),$$

where δ is the Dirac δ-function:

$$\delta(x) = \begin{cases} +\infty, & x=0, \\ 0, & x\neq 0, \end{cases}, \quad \int_{\mathbb{R}^n}\delta(x)dx = 1.$$

The key difference between the EnKF and its extensions and the particle filter is that the weights in the particle filter are allowed to vary. As a result, the pdfs $f(v_j|Y_j)$ and $f(v_{j+1}|Y_j)$ are approximated by

$$\mu_j \approx \mu_j^{(N)} := \sum_{i=1}^{N} w_i^j \delta(x-U_i), \quad \hat{m} \approx \hat{\mu}^{(N)} \sum_{i=1}^{N}\hat{w}_i\delta(x-\hat{U}_i). \tag{30}$$

The objective of the particle filter is to set up appropriate update rules for the prediction and analysis steps respectively:

(31)

$$\left\{U_i^{(j)}, w_i^{(j)}\right\}_{i=1}^{N} \mapsto \left\{\hat{U}_i^{(j+1)}, \hat{w}_i^{(j+1)}\right\}_{i=1}^{N}, \quad \left\{\hat{U}_i^{(j+1)}, \hat{w}_i^{(j+1)}\right\}_{i=1}^{N} \mapsto \left\{U_i^{(j+1)}, w_i^{(j+1)}\right\}_{i=1}^{N}.$$

Here the superscripts indicate that the ensemble members and their weights depend on j (j is the step number).

5.6.1. *Basic particle filter.* The basic particle filter, the so-called *sequential importance resampling (SIRS)* is the following algorithm.

Particle Filter (basic)

Setup

Pick N, the number of particles in the ensemble.

Initialization

$U = [U_1, \ldots, U_N]$ is $n \times N$ matrix with the positions of ensemble members.

for $i = 1, \ldots, N$ **do**

$U_i = m_0 + C_0^{1/2}\xi_i$, where $\xi_i \sim \mathcal{N}(0,1)$

end for

Main body

for $j = 0, 1, 2, \ldots$ **do**

Prediction step: evolve the ensemble

a. $\hat{U} = \Psi(U) + \Sigma^{1/2}\zeta$ where ζ is $n \times N$ matrix of i.i.d. RVs $\zeta_{ij} \sim \mathcal{N}(0,1)$ \\ ensemble predict

Analysis step

b. $d = y_{j+1} - H\hat{U}$ \\ ensemble innovation

c. $\hat{w} = \exp\left(-\frac{1}{2}d^\top\Gamma^{-1}d\right)$ \\ propose weight update

d. $w = \left(\sum_{i=1}^{n}\hat{w}_i\right)^{-1}\hat{w}$ \\ normalize weight update

e. Resample the ensemble according to the CDF of weights. Here is a Matlab code for this resampling:

```
ws = cumsum(w); % compute CDF of weights
for i = 1 : N
     ix = [];
     while isempty(ix)
            ix = find(ws > rand,1,'first'); % resample: draw rand ~ U[0,1]
     end
     % and find the index of the particle corresponding to the
     % first time the CDF of the weights exceeds rand
     U(:,i) = Uhat(:,ix); % reset the n-th particle to the one with index ix
end
```

f. $m_{j+1} = \frac{1}{N}\sum_{i=1}^N U_i$ \\ estimator update

g. $C_{j+1} = \frac{1}{N-1}\sum_{i=1}^N (U_i - m_{j+1})(U_i - m_{j+1})^\top$ \\ covariance update

end for

Let us motivate the analysis step. See [11] for a more detailed explanation. We observe that

$$d := y_{j+1} - Hv_{j+1} \sim \mathcal{N}(0,\Gamma), \text{ i.e. } f_d(x) = Z^{-1}\exp\left(-\tfrac{1}{2}x^\top \Gamma^{-1} x\right).$$

Now recall Eq. (17):

$$f(v_{j+1}|Y_{j+1}) = \frac{f(y_{j+1}|v_{j+1})f(v_{j+1}|Y_j)}{f(y_{j+1}|Y_j)}.$$

It says that

$$\frac{f(v_{j+1}|Y_{j+1})}{f(v_{j+1}|Y_j)} \propto f(y_{j+1}|v_{j+1}) = f_d(x) = Z^{-1}\exp\left(-\tfrac{1}{2}x^\top \Gamma^{-1} x\right). \tag{32}$$

The left-hand side of Eq. (32) is the ratio of the pdfs approximated by the ensembles $U^{(j+1)}$ and $\hat{U}$. Hence, Eq. (32) suggests to pick weights to the updated distribution $f(v_{j+1}|Y_{j+1})$ by the updated ensemble $\hat{U}$ proportional to $f_d(x)$. Then we resample the ensemble in order to approximate the resulting distribution using an ensemble with uniform weights.

Remark The choice of weights in the basic particle filter algorithm can be improved by doind the so-called *optimal importance resampling*. See Section 4.3.3 in [11].

References

[1] I.T. Jolliffe. Principal component analysis. Springer-Verlag New York, 1986

[2] I. Borg, P. J. F. Groenen, Modern multidimensional scaling: theory and applications, Springer e-book, 2005

[3] M. Hervella, N. Izagirre, S. Alonso, M. Ioana, M. Netea, and C. de-la-Rua, BMC Genetics, 2014,15, 56 http://www.biomedcentral.com/1471-2156/15/56

[4] V. de Silva and J. B. Tenenbaum, Sparse multidimensional scaling using landmark points, Technical report, Stanford University, Stanford, CA, 2004

[5] R. Coifman and S. Lafon, Diffusion Maps, Appl. Comput. Harmon. Anal. 21 (2006) 5–30

[6] J. de la Porte, B. M. Herbst, W. Hereman, S. J. van der Walt, An Introduction to Diffusion Maps

[7] A. L. Fergusona, A. Z. Panagiotopoulosa, P. G. Debenedettia, and I. G. Kevrekidis, Systematic determination of order parameters for chain dynamics using diffusion maps, PNAS, August 3, 2010, vol. 107, no. 31, 13597?13602
[8] S. B. Kim, C. J. Dsilva, I. G. Kevrekidis, and P. G. Debenedetti, Systematic characterization of protein folding pathways using diffusion maps: Application to Trp-cage miniprotein, J. Chem. Phys. 142, 085101 (2015)
[9] A. V. Little, M. Maggioni, and L. Rosasco, Multiscale geometric methods for estimating intrinsic dimensions, SampTA 2011
[10] A. V. Little, Estimating the Intrinsic Dimension of High-Dimensional Data Sets: A Multiscale, Geometric Approach, Dissertation, Department of Mathematics, Duke University, 2011
[11] K. J. H. Law, A. M. Stuart, K. C. Zygalakis, Data Assimilation: A Mathematical Introduction, Springer, 2015, arXiv:1506.07825
[12] Daniel Kahneman, "Thinking Fast and Slow", Farrar, Straus and Giroux, New York, 2011
[13] Kalman, R. E. (1960). "A New Approach to Linear Filtering and Prediction Problems". Journal of Basic Engineering. 82: 35–45
[14] B. R. Hunt, E. J. Kostelichb, I. Szunyogh, Efficient data assimilation for spatiotemporal chaos: A local ensemble transform Kalman filter, Physica D, 230 (2007) 112–126, arXiv