15 Dimensionality Reduction

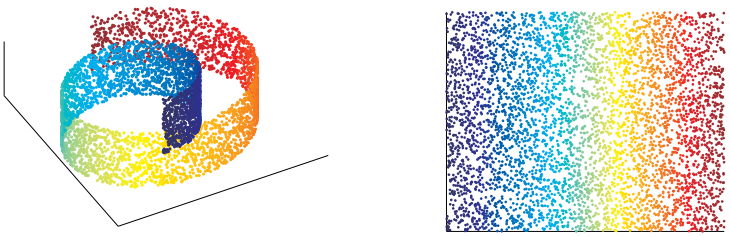
In settings where the data has a large number of features, it is often desirable to reduce its dimension, or to find a lower-dimensional representation preserving some of its properties. The key arguments for dimensionality reduction (or manifold learning) techniques are:

- *Computational*: to compress the initial data as a preprocessing step to speed up subsequent operations on the data.
- *Visualization*: to visualize the data for exploratory analysis by mapping the input data into two- or three-dimensional spaces.
- *Feature extraction*: to hopefully generate a smaller and more effective or useful set of features.

The benefits of dimensionality reduction are often illustrated via simulated data, such as the Swiss roll dataset. In this example, the input data, depicted in figure 15.1a, is three-dimensional, but it lies on a two-dimensional manifold that is "unfolded" in two-dimensional space as shown in figure 15.1b. It is important to note, however, that exact low-dimensional manifolds are rarely encountered in practice. Hence, this idealized example is more useful to illustrate the concept of dimensionality reduction than to verify the effectiveness of dimensionality reduction algorithms.

Dimensionality reduction can be formalized as follows. Consider a sample $S = (x_1, \ldots, x_m)$, a feature mapping $\mathbf{\Phi} \colon \mathcal{X} \to \mathbb{R}^N$ and the data matrix $\mathbf{X} \in \mathbb{R}^{N \times m}$ defined as $(\mathbf{\Phi}(x_1), \ldots, \mathbf{\Phi}(x_m))$. The ith data point is represented by $\mathbf{x}_i = \mathbf{\Phi}(x_i)$, or the ith column of $\mathbf{X}$, which is an N-dimensional vector. Dimensionality reduction techniques broadly aim to find, for $k \ll N$, a k-dimensional representation of the data, $\mathbf{Y} \in \mathbb{R}^{k \times m}$, that is in some way faithful to the original representation $\mathbf{X}$.

In this chapter we will discuss various techniques that address this problem. We first present the most commonly used dimensionality reduction technique called *principal component analysis* (PCA). We then introduce a kernelized version of PCA (KPCA) and show the connection between KPCA and manifold learning



(a) (b)

Figure 15.1
The "Swiss roll" dataset. (a) high-dimensional representation. (b) lower-dimensional representation.

algorithms. We conclude with a presentation of the Johnson-Lindenstrauss lemma, a classical theoretical result that has inspired a variety of dimensionality reduction methods based on the concept of random projections. The discussion in this chapter relies on basic matrix properties that are reviewed in appendix A.

15.1 Principal component analysis

Fix $k \in [N]$ and let $\mathbf{X}$ be a mean-centered data matrix, that is, $\sum_{i=1}^{m} \mathbf{x}_i = \mathbf{0}$. Define $\mathcal{P}_k$ as the set of N-dimensional rank-k orthogonal projection matrices. PCA consists of projecting the N-dimensional input data onto the k-dimensional linear subspace that minimizes *reconstruction error*, that is the sum of the squared L_2-distances between the original data and the projected data. Thus, the PCA algorithm is completely defined by the orthogonal projection matrix solution $\mathbf{P}^*$ of the following minimization problem:

$$\min_{\mathbf{P} \in \mathcal{P}_k} \|\mathbf{PX} - \mathbf{X}\|_F^2 . \tag{15.1}$$

The following theorem shows that PCA coincides with the projection of each data point onto the k top singular vectors of the sample covariance matrix, i.e., $\mathbf{C} = \frac{1}{m}\mathbf{X}\mathbf{X}^\top$ for the mean-centered data matrix $\mathbf{X}$. Figure 15.2 illustrates the basic intuition behind PCA, showing how two-dimensional data points with highly correlated features can be more succinctly represented with a one-dimensional representation that captures most of the variance in the data.

Theorem 15.1 *Let $\mathbf{P}^* \in \mathcal{P}_k$ be the PCA solution, i.e., the orthogonal projection matrix solution of* (15.1)*. Then, $\mathbf{P}^* = \mathbf{U}_k\mathbf{U}_k^\top$, where $\mathbf{U}_k \in \mathbb{R}^{N \times k}$ is the matrix*

formed by the top k singular vectors of $\mathbf{C} = \frac{1}{m}\mathbf{X}\mathbf{X}^\top$, the sample covariance matrix corresponding to $\mathbf{X}$. Moreover, the associated k-dimensional representation of $\mathbf{X}$ is given by $\mathbf{Y} = \mathbf{U}_k^\top \mathbf{X}$.

Proof: Let $\mathbf{P} = \mathbf{P}^\top$ be an orthogonal projection matrix. By the definition of the Frobenius norm, the linearity of the trace operator and the fact that $\mathbf{P}$ is idempotent, i.e., $\mathbf{P}^2 = \mathbf{P}$, we observe that

$$\begin{aligned}\|\mathbf{P}\mathbf{X} - \mathbf{X}\|_F^2 &= \operatorname{Tr}[(\mathbf{P}\mathbf{X} - \mathbf{X})^\top(\mathbf{P}\mathbf{X} - \mathbf{X})] = \operatorname{Tr}[\mathbf{X}^\top\mathbf{P}^2\mathbf{X} - 2\mathbf{X}^\top\mathbf{P}\mathbf{X} + \mathbf{X}^\top\mathbf{X}] \\ &= -\operatorname{Tr}[\mathbf{X}^\top\mathbf{P}\mathbf{X}] + \operatorname{Tr}[\mathbf{X}^\top\mathbf{X}]\,.\end{aligned}$$

Since $\operatorname{Tr}[\mathbf{X}^\top\mathbf{X}]$ is a constant with respect to $\mathbf{P}$, we have

$$\operatorname*{argmin}_{\mathbf{P}\in\mathcal{P}_k} \|\mathbf{P}\mathbf{X} - \mathbf{X}\|_F^2 = \operatorname*{argmax}_{\mathbf{P}\in\mathcal{P}_k} \operatorname{Tr}[\mathbf{X}^\top\mathbf{P}\mathbf{X}]\,. \tag{15.2}$$

By definition of orthogonal projections in $\mathcal{P}_k$, $\mathbf{P} = \mathbf{U}\mathbf{U}^\top$ for some $\mathbf{U} \in \mathbb{R}^{N\times k}$ containing orthogonal columns. Using the invariance of the trace operator under cyclic permutations and the orthogonality of the columns of $\mathbf{U}$, we have

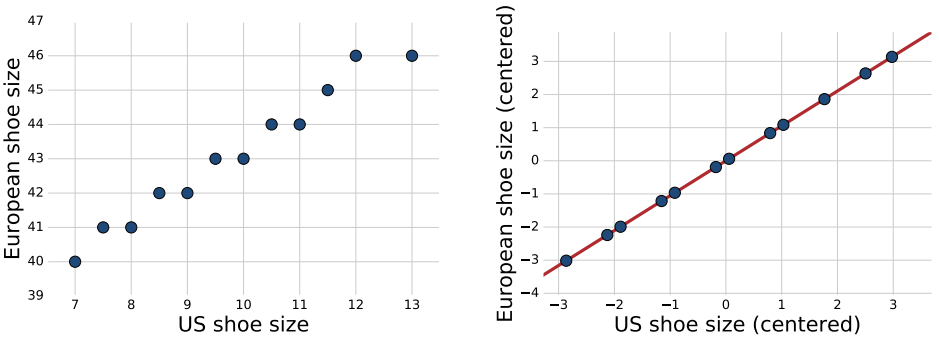
$$\operatorname{Tr}[\mathbf{X}^\top\mathbf{P}\mathbf{X}] = \operatorname{Tr}[\mathbf{U}^\top\mathbf{X}\mathbf{X}^\top\mathbf{U}] = \sum_{i=1}^{k} \mathbf{u}_i^\top\mathbf{X}\mathbf{X}^\top\mathbf{u}_i\,,$$

where $\mathbf{u}_i$ is the ith column of $\mathbf{U}$. By the Rayleigh quotient (section A.2.3), it is clear that the largest k singular vectors of $\mathbf{X}\mathbf{X}^\top$ maximize the rightmost sum above. Since $\mathbf{X}\mathbf{X}^\top$ and $\mathbf{C}$ differ only by a scaling factor, they have the same singular vectors, and thus $\mathbf{U}_k$ maximizes this sum, which proves the first statement of the theorem. Finally, since $\mathbf{P}\mathbf{X} = \mathbf{U}_k\mathbf{U}_k^\top\mathbf{X}$, $\mathbf{Y} = \mathbf{U}_k^\top\mathbf{X}$ is a k-dimensional representation of $\mathbf{X}$ with $\mathbf{U}_k$ as the basis vectors. □

By definition of the covariance matrix, the top singular vectors of $\mathbf{C}$ are the directions of maximal variance in the data, and the associated singular values are equal to these variances. Hence, PCA can also be viewed as projecting onto the subspace of maximal variance. Under this interpretation, the first principal component is derived from projection onto the direction of maximal variance, given by the top singular vector of $\mathbf{C}$. Similarly, the ith principal component, for $1 \le i \le k$, is derived from projection onto the ith direction of maximal variance, subject to orthogonality constraints to the previous $i-1$ directions of maximal variance (see exercise 15.1 for more details).

15.2 Kernel principal component analysis (KPCA)

In the previous section, we presented the PCA algorithm, which involved projecting onto the singular vectors of the sample covariance matrix $\mathbf{C}$. In this section, we



(a) (b)

Figure 15.2
Example of PCA. (a) Two-dimensional data points with features capturing shoe size measured with different units. (b) One-dimensional representation that captures the most variance in the data, generated by projecting onto largest principal component (red line) of the mean-centered data points.

present a kernelized version of PCA, called KPCA. In the KPCA setting, $\boldsymbol{\Phi}$ is a feature mapping to an arbitrary RKHS (not necessarily to $\mathbb{R}^N$) and we work exclusively with a kernel function K corresponding to the inner product in this RKHS. The KPCA algorithm can thus be defined as a generalization of PCA in which the input data is projected onto the top principle components in this RKHS. We will show the relationship between PCA and KPCA by drawing upon the deep connections among the SVDs of $\mathbf{X}$, $\mathbf{C}$ and $\mathbf{K}$. We then illustrate how various manifold learning algorithms can be interpreted as special instances of KPCA.

Let K be a PDS kernel defined over $\mathcal{X} \times \mathcal{X}$ and define the kernel matrix as $\mathbf{K} = \mathbf{X}^\top \mathbf{X}$. Since $\mathbf{X}$ admits the following singular value decomposition: $\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top$, $\mathbf{C}$ and $\mathbf{K}$ can be rewritten as follows:

$$\mathbf{C} = \frac{1}{m}\mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top \qquad \mathbf{K} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^\top, \tag{15.3}$$

where $\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^2$ is the diagonal matrix of the singular values (equivalently eigenvalues) of $m\mathbf{C}$ and $\mathbf{U}$ is the matrix of the singular vectors (equivalently eigenvectors) of $\mathbf{C}$ (and $m\mathbf{C}$).

Starting with the SVD of $\mathbf{X}$, note that right multiplying by $\mathbf{V}\boldsymbol{\Sigma}^{-1}$ and using the relationship between $\boldsymbol{\Lambda}$ and $\boldsymbol{\Sigma}$ yields $\mathbf{U} = \mathbf{X}\mathbf{V}\boldsymbol{\Lambda}^{-1/2}$. Thus, the singular vector $\mathbf{u}$ of $\mathbf{C}$ associated to the singular value λ/m coincides with $\frac{\mathbf{X}\mathbf{v}}{\sqrt{\lambda}}$, where $\mathbf{v}$ is the singular vector of $\mathbf{K}$ associated to λ. Now fix an arbitrary feature vector $\mathbf{x} = \boldsymbol{\Phi}(x)$ for $x \in \mathcal{X}$. Then, following the expression for $\mathbf{Y}$ in theorem 15.1, the one-dimensional

representation of $\mathbf{x}$ derived by projection onto $\mathbf{P}_u = \mathbf{u}\mathbf{u}^\top$ is defined by

$$\mathbf{x}^\top \mathbf{u} = \mathbf{x}^\top \frac{\mathbf{X}\mathbf{v}}{\sqrt{\lambda}} = \frac{\mathbf{k}_x^\top \mathbf{v}}{\sqrt{\lambda}}\,, \tag{15.4}$$

where $\mathbf{k}_x = (K(x_1, x), \ldots, K(x_m, x))^\top$. If $\mathbf{x}$ is one of the data points, i.e., $\mathbf{x} = \mathbf{x}_i$ for $1 \leq i \leq m$, then $\mathbf{k}_x$ is the ith column of $\mathbf{K}$ and (15.4) can be simplified as follows:

$$\mathbf{x}^\top \mathbf{u} = \frac{\mathbf{k}_x^\top \mathbf{v}}{\sqrt{\lambda}} = \frac{\lambda v_i}{\sqrt{\lambda}} = \sqrt{\lambda} v_i\,, \tag{15.5}$$

where v_i is the ith component of $\mathbf{v}$. More generally, the PCA solution of theorem 15.1 can be fully defined by the top k singular vectors (or eigenvectors) of $\mathbf{K}$, $\mathbf{v}_1, \ldots, \mathbf{v}_k$, and the corresponding singular values (or eigenvalues). This alternative derivation of the PCA solution in terms of $\mathbf{K}$ precisely defines the KPCA solution, providing a generalization of PCA via the use of PDS kernels (see chapter 6 for more details on kernel methods).

15.3 KPCA and manifold learning

Several manifold learning techniques have been proposed as non-linear methods for dimensionality reduction. These algorithms implicitly assume that high-dimensional data lie on or near a low-dimensional non-linear manifold embedded in the input space. They aim to learn this manifold structure by finding a low-dimensional space that in some way preserves the local structure of high-dimensional input data. For instance, the Isomap algorithm aims to preserve approximate geodesic distances, or distances along the manifold, between all pairs of data points. Other algorithms, such as Laplacian eigenmaps and locally linear embedding, focus only on preserving local neighborhood relationships in the high-dimensional space. We will next describe these classical manifold learning algorithms and then interpret them as specific instances of KPCA.

15.3.1 Isomap

Isomap aims to extract a low-dimensional data representation that best preserves all pairwise distances between input points, as measured by their geodesic distances along the underlying manifold. It approximates geodesic distance assuming that L_2 distance provides good approximations for nearby points, and for faraway points it estimates distance as a series of hops between neighboring points. The Isomap algorithm works as follows:

1. Find the t nearest neighbors for each data point based on L_2 distance and construct an undirected neighborhood graph, denoted by $\mathcal{G}$, with points as nodes and links between neighbors as edges.
2. Compute the approximate geodesic distances, Δ_{ij}, between all pairs of nodes (i, j) by computing all-pairs shortest distances in $\mathcal{G}$ using, for instance, the Floyd-Warshall algorithm.
3. Convert the squared distance matrix into a $m \times m$ similarity matrix by performing double centering, i.e., compute $\mathbf{K}_{\text{Iso}} = -\frac{1}{2}\mathbf{H}\Delta\mathbf{H}$, where Δ is the squared distance matrix, $\mathbf{H} = \mathbf{I}_m - \frac{1}{m}\mathbf{1}\mathbf{1}^\top$ is the centering matrix, $\mathbf{I}_m$ is the $m \times m$ identity matrix and $\mathbf{1}$ is a column vector of all ones (for more details on double centering see exercise 15.2).
4. Find the optimal k-dimensional representation, $\mathbf{Y} = \{\mathbf{y}_i\}_{i=1}^n$, such that $\mathbf{Y} = \text{argmin}_{\mathbf{Y}'} \sum_{i,j} \left(\|\mathbf{y}'_i - \mathbf{y}'_j\|_2^2 - \Delta_{ij}^2\right)$. The solution is given by,

$$\mathbf{Y} = (\mathbf{\Sigma}_{\text{Iso},k})^{1/2}\mathbf{U}_{\text{Iso},k}^\top \tag{15.6}$$

   where $\mathbf{\Sigma}_{\text{Iso},k}$ is the diagonal matrix of the top k singular values of $\mathbf{K}_{\text{Iso}}$ and $\mathbf{U}_{\text{Iso},k}$ are the associated singular vectors.

$\mathbf{K}_{\text{Iso}}$ can naturally be viewed as a kernel matrix, thus providing a simple connection between Isomap and KPCA. Note, however, that this interpretation is valid only when $\mathbf{K}_{\text{Iso}}$ is in fact positive semidefinite, which is indeed the case in the continuum limit for a smooth manifold.

15.3.2 Laplacian eigenmaps

The *Laplacian eigenmaps* algorithm aims to find a low-dimensional representation that best preserves neighborhood relations as measured by a weight matrix $\mathbf{W}$. The algorithm works as follows:

1. Find t nearest neighbors for each point.
2. Construct $\mathbf{W}$, a sparse, symmetric $m \times m$ matrix, where $\mathbf{W}_{ij} = \exp\left(-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2/\sigma^2\right)$ if $(\mathbf{x}_i, \mathbf{x}_j)$ are neighbors, 0 otherwise, and σ is a scaling parameter.
3. Construct the diagonal matrix $\mathbf{D}$, such that $\mathbf{D}_{ii} = \sum_j \mathbf{W}_{ij}$.
4. Find the k-dimensional representation by minimizing the weighted distance between neighbors as,

$$\mathbf{Y} = \underset{\mathbf{Y}'}{\text{argmin}} \sum_{i,j} \mathbf{W}_{ij}\|\mathbf{y}'_i - \mathbf{y}'_j\|_2^2. \tag{15.7}$$

   This objective function penalizes nearby inputs for being mapped to faraway outputs, with "nearness" measured by the weight matrix $\mathbf{W}$. The solution to the minimization in (15.7) is $\mathbf{Y} = \mathbf{U}_{\mathbf{L},k}^\top$, where $\mathbf{L} = \mathbf{D} - \mathbf{W}$ is the graph Laplacian

and $\mathbf{U}_{\mathbf{L},k}^{\top}$ are the bottom k singular vectors of $\mathbf{L}$, excluding the last singular vector corresponding to the singular value 0 (assuming that the underlying neighborhood graph is connected).

The solution to (15.7) can also be interpreted as finding the largest singular vectors of $\mathbf{L}^{\dagger}$, the pseudo-inverse of $\mathbf{L}$. Defining $\mathbf{K}_{\mathbf{L}} = \mathbf{L}^{\dagger}$ we can thus view Laplacian Eigenmaps as an instance of KPCA in which the output dimensions are normalized to have unit variance, which corresponds to setting $\lambda = 1$ in (15.5). Moreover, it can be shown that $\mathbf{K}_{\mathbf{L}}$ is the kernel matrix associated with the commute times of diffusion on the underlying neighborhood graph, where the commute time between nodes i and j in a graph is the expected time taken for a random walk to start at node i, reach node j and then return to i.

15.3.3 Locally linear embedding (LLE)

The *locally linear embedding* (LLE) algorithm also aims to find a low-dimensional representation that preserves neighborhood relations as measured by a weight matrix $\mathbf{W}$. The algorithm works as follows:

1. Find t nearest neighbors for each point.
2. Construct $\mathbf{W}$, a sparse, symmetric $m \times m$ matrix, whose ith row sums to one and contains the linear coefficients that optimally reconstruct $\mathbf{x}_i$ from its t neighbors. More specifically, if we assume that the ith row of $\mathbf{W}$ sums to one, then the reconstruction error is
$$\Big(\mathbf{x}_i - \sum_{j \in \mathcal{N}_i} \mathbf{W}_{ij}\mathbf{x}_j\Big)^2 = \Big(\sum_{j \in \mathcal{N}_i} \mathbf{W}_{ij}(\mathbf{x}_i - \mathbf{x}_j)\Big)^2 = \sum_{j,k \in \mathcal{N}_i} \mathbf{W}_{ij}\mathbf{W}_{ik}\mathbf{C}'_{jk} \tag{15.8}$$
where $\mathcal{N}_i$ is the set of indices of the neighbors of point $\mathbf{x}_i$ and $\mathbf{C}'_{jk} = (\mathbf{x}_i - \mathbf{x}_j)^{\top}(\mathbf{x}_i - \mathbf{x}_k)$ the local covariance matrix. Minimizing this expression with the constraint $\sum_j \mathbf{W}_{ij} = 1$ gives the solution
$$\mathbf{W}_{ij} = \frac{\sum_k (\mathbf{C}'^{-1})_{jk}}{\sum_{st} (\mathbf{C}'^{-1})_{st}}\,. \tag{15.9}$$
Note that the solution can be equivalently obtained by first solving the system of linear equations $\sum_j \mathbf{C}'_{kj}\mathbf{W}_{ij} = 1$, for $k \in \mathcal{N}_i$, and then normalizing so that the weights sum to one.
3. Find the k-dimensional representation that best obeys neighborhood relations as specified by $\mathbf{W}$, i.e.,
$$\mathbf{Y} = \operatorname*{argmin}_{\mathbf{Y}'} \sum_i \Big(\mathbf{y}'_i - \sum_j \mathbf{W}_{ij}\mathbf{y}'_j\Big)^2. \tag{15.10}$$

The solution to the minimization in (15.10) is $\mathbf{Y} = \mathbf{U}_{\mathbf{M},k}^\top$, where $\mathbf{M} = (\mathbf{I} - \mathbf{W}^\top)(\mathbf{I} - \mathbf{W}^\top)$ and $\mathbf{U}_{\mathbf{M},k}^\top$ are the bottom k singular vectors of $\mathbf{M}$, excluding the last singular vector corresponding to the singular value 0.

As discussed in exercise 15.5, LLE coincides with KPCA used with a particular kernel matrix $\mathbf{K}_{LLE}$ whereby the output dimensions are normalized to have unit variance (as in the case of Laplacian Eigenmaps).

15.4 Johnson-Lindenstrauss lemma

The Johnson-Lindenstrauss lemma is a fundamental result in dimensionality reduction that states that any m points in high-dimensional space can be mapped to a much lower dimension, $k \geq O(\frac{\log m}{\epsilon^2})$, without distorting pairwise distance between any two points by more than a factor of $(1 \pm \epsilon)$. In fact, such a mapping can be found in randomized polynomial time by projecting the high-dimensional points onto randomly chosen k-dimensional linear subspaces. The Johnson-Lindenstrauss lemma is formally presented in lemma 15.4. The proof of this lemma hinges on lemma 15.2 and lemma 15.3, and it is an example of the "probabilistic method", in which probabilistic arguments lead to a deterministic statement. Moreover, as we will see, the Johnson-Lindenstrauss lemma follows by showing that the squared norm of a random vector is sharply concentrated around its mean when the vector is projected onto a k-dimensional random subspace.

First, we prove the following property of the χ^2 distribution (see definition C.7 in appendix), which will be used in lemma 15.3.

Lemma 15.2 *Let Q be a random variable following a χ^2 distribution with k degrees of freedom. Then, for any $0 < \epsilon < 1/2$, the following inequality holds:*

$$\mathbb{P}[(1-\epsilon)k \leq Q \leq (1+\epsilon)k] \geq 1 - 2e^{-(\epsilon^2-\epsilon^3)k/4}. \tag{15.11}$$

Proof: By Markov's inequality, we can write

$$\begin{aligned}\mathbb{P}[Q \geq (1+\epsilon)k] = \mathbb{P}[\exp(\lambda Q) \geq \exp(\lambda(1+\epsilon)k)] &\leq \frac{\mathbb{E}[\exp(\lambda Q)]}{\exp(\lambda(1+\epsilon)k)} \\ &= \frac{(1-2\lambda)^{-k/2}}{\exp(\lambda(1+\epsilon)k)},\end{aligned}$$

where we used for the final equality the expression of the moment-generating function of a χ^2 distribution, $\mathbb{E}[\exp(\lambda Q)]$, for $\lambda < 1/2$ (equation (C.25)). Choosing $\lambda = \frac{\epsilon}{2(1+\epsilon)} < 1/2$, which minimizes the right-hand side of the final equality, and using the inequality $1 + \epsilon \leq \exp(\epsilon - (\epsilon^2 - \epsilon^3)/2)$ yield

$$\mathbb{P}[Q \geq (1+\epsilon)k] \leq \left(\frac{1+\epsilon}{\exp(\epsilon)}\right)^{k/2} \leq \left(\frac{\exp\left(\epsilon - \frac{\epsilon^2-\epsilon^3}{2}\right)}{\exp(\epsilon)}\right)^{k/2} = \exp\left(-\frac{k}{4}(\epsilon^2-\epsilon^3)\right).$$

The statement of the lemma follows by using similar techniques to bound $\mathbb{P}[Q \leq (1-\epsilon)k]$ and by applying the union bound. □

Lemma 15.3 *Let $\mathbf{x} \in \mathbb{R}^N$, define $k < N$ and assume that entries in $\mathbf{A} \in \mathbb{R}^{k \times N}$ are sampled independently from the standard normal distribution, $N(0,1)$. Then, for any $0 < \epsilon < 1/2$,*

$$\mathbb{P}\left[(1-\epsilon)\|\mathbf{x}\|^2 \leq \|\frac{1}{\sqrt{k}}\mathbf{A}\mathbf{x}\|^2 \leq (1+\epsilon)\|\mathbf{x}\|^2\right] \geq 1 - 2e^{-(\epsilon^2-\epsilon^3)k/4}\,. \tag{15.12}$$

Proof: Let $\widehat{\mathbf{x}} = \mathbf{A}\mathbf{x}$ and observe that

$$\mathbb{E}[\widehat{x}_j^2] = \mathbb{E}\left[\Big(\sum_{i=1}^N A_{ji}x_i\Big)^2\right] = \mathbb{E}\left[\sum_{i=1}^N A_{ji}^2 x_i^2\right] = \sum_{i=1}^N x_i^2 = \|\mathbf{x}\|^2\,.$$

The second and third equalities follow from the independence and unit variance, respectively, of the A_{ij}. Now, define $T_j = \widehat{x}_j/\|\mathbf{x}\|$ and note that the T_js are independent standard normal random variables since the A_{ij} are i.i.d. standard normal random variables and $\mathbb{E}[\widehat{x}_j^2] = \|\mathbf{x}\|^2$. Thus, the variable Q defined by $Q = \sum_{j=1}^k T_j^2$ follows a χ^2 distribution with k degrees of freedom and we have

$$\begin{aligned}\mathbb{P}\left[(1-\epsilon)\|\mathbf{x}\|^2 \leq \frac{\|\widehat{\mathbf{x}}\|^2}{k} \leq (1+\epsilon)\|\mathbf{x}\|^2\right] &= \mathbb{P}\left[(1-\epsilon)k \leq \sum_{j=1}^k T_j^2 \leq (1+\epsilon)k\right] \\ &= \mathbb{P}\left[(1-\epsilon)k \leq Q \leq (1+\epsilon)k\right] \\ &\geq 1 - 2e^{-(\epsilon^2-\epsilon^3)k/4}\,,\end{aligned}$$

where the final inequality holds by lemma 15.2, thus proving the statement of the lemma. □

Lemma 15.4 (Johnson-Lindenstrauss) *For any $0 < \epsilon < 1/2$ and any integer $m > 4$, let $k = \frac{20\log m}{\epsilon^2}$. Then for any set V of m points in $\mathbb{R}^N$, there exists a map $f\colon \mathbb{R}^N \to \mathbb{R}^k$ such that for all $\mathbf{u}, \mathbf{v} \in V$,*

$$(1-\epsilon)\|\mathbf{u}-\mathbf{v}\|^2 \leq \|f(\mathbf{u}) - f(\mathbf{v})\|^2 \leq (1+\epsilon)\|\mathbf{u}-\mathbf{v}\|^2. \tag{15.13}$$

Proof: Let $f = \frac{1}{\sqrt{k}}\mathbf{A}$ where $k < N$ and entries in $\mathbf{A} \in \mathbb{R}^{k\times N}$ are sampled independently from the standard normal distribution, $N(0,1)$. For fixed $\mathbf{u}, \mathbf{v} \in V$, we can apply lemma 15.3, with $\mathbf{x} = \mathbf{u} - \mathbf{v}$, to lower bound the success probability by $1 - 2e^{-(\epsilon^2-\epsilon^3)k/4}$. Applying the union bound over the $\mathrm{O}(m^2)$ pairs in V, setting $k = \frac{20}{\epsilon^2}\log m$ and upper bounding ϵ by $1/2$, we have

$$\mathbb{P}[\mathit{success}] \geq 1 - 2m^2 e^{-(\epsilon^2-\epsilon^3)k/4} = 1 - 2m^{5\epsilon-3} > 1 - 2m^{-1/2} > 0\,.$$

Since the success probability is strictly greater than zero, a map that satisfies the desired conditions must exist, thus proving the statement of the lemma. □

15.5 Chapter notes

PCA was introduced in the early 1900s by Pearson [1901]. KPCA was introduced roughly a century later, and our presentation of KPCA is a more concise derivation of results given by Mika et al. [1999]. Isomap and LLE were pioneering works on non-linear dimensionality reduction introduced by Tenenbaum et al. [2000], Roweis and Saul [2000]. Isomap itself is a generalization of a standard linear dimensionality reduction technique called Multidimensional Scaling [Cox and Cox, 2000]. Isomap and LLE led to the development of several related algorithms for manifold learning, e.g., Laplacian Eigenmaps and Maximum Variance Unfolding [Belkin and Niyogi, 2001, Weinberger and Saul, 2006]. As shown in this chapter, classical manifold learning algorithms are special instances of KPCA [Ham et al., 2004]. The Johnson-Lindenstrauss lemma was introduced by Johnson and Lindenstrauss [1984], though our proof of the lemma follows Vempala [2004]. Other simplified proofs of this lemma have also been presented, including Dasgupta and Gupta [2003].

15.6 Exercises

15.1 PCA and maximal variance. Let $\mathbf{X}$ be an *uncentered* data matrix and let $\bar{\mathbf{x}} = \frac{1}{m}\sum_i \mathbf{x}_i$ be the sample mean of the columns of $\mathbf{X}$.

(a) Show that the variance of one-dimensional projections of the data onto an arbitrary vector $\mathbf{u}$ equals $\mathbf{u}^\top \mathbf{C}\mathbf{u}$, where $\mathbf{C} = \frac{1}{m}\sum_i (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\top$ is the sample covariance matrix.

(b) Show that PCA with $k = 1$ projects the data onto the direction (i.e., $\mathbf{u}^\top \mathbf{u} = 1$) of maximal variance.

15.2 Double centering. In this problem we will prove the correctness of the double centering step in Isomap when working with Euclidean distances. Define $\mathbf{X}$ and $\bar{\mathbf{x}}$ as in exercise 15.1, and define $\mathbf{X}^*$ as the centered version of $\mathbf{X}$, that is, let $\mathbf{x}_i^* = \mathbf{x}_i - \bar{\mathbf{x}}$ be the ith column of $\mathbf{X}^*$. Let $\mathbf{K} = \mathbf{X}^\top \mathbf{X}$, and let $\mathbf{D}$ denote the Euclidean distance matrix, i.e., $\mathbf{D}_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|$.

(a) Show that $\mathbf{K}_{ij} = \frac{1}{2}(\mathbf{K}_{ii} + \mathbf{K}_{jj} + \mathbf{D}_{ij}^2)$.

(b) Show that $\mathbf{K}^* = {\mathbf{X}^*}^\top \mathbf{X}^* = \mathbf{K} - \frac{1}{m}\mathbf{K}\mathbf{1}\mathbf{1}^\top - \frac{1}{m}\mathbf{1}\mathbf{1}^\top\mathbf{K} + \frac{1}{m^2}\mathbf{1}\mathbf{1}^\top\mathbf{K}\mathbf{1}\mathbf{1}^\top$.

(c) Using the results from (a) and (b) show that

$$\mathbf{K}_{ij}^* = -\frac{1}{2}\left[\mathbf{D}_{ij}^2 - \frac{1}{m}\sum_{k=1}^{m}\mathbf{D}_{ik}^2 - \frac{1}{m}\sum_{k=1}^{m}\mathbf{D}_{kj}^2 + \bar{\mathbf{D}}\right],$$

where $\bar{\mathbf{D}} = \frac{1}{m^2}\sum_u \sum_v \mathbf{D}^2_{u,v}$ is the mean of the m^2 entries in $\mathbf{D}$.

(d) Show that $\mathbf{K}^* = -\frac{1}{2}\mathbf{HDH}$, where $\mathbf{H} = \mathbf{I}_m - \frac{1}{m}\mathbf{1}\mathbf{1}^\top$.

15.3 Laplacian eigenmaps. Assume $k = 1$ and we seek a one-dimensional representation $\mathbf{y}$. Show that (15.7) is equivalent to $\mathbf{y} = \text{argmin}_{\mathbf{y}'}\, \mathbf{y}'^\top \mathbf{L}\mathbf{y}'$, where $\mathbf{L}$ is the graph Laplacian.

15.4 Nyström method. Define the following block representation of a kernel matrix:

$$\mathbf{K} = \begin{bmatrix} \mathbf{W} & \mathbf{K}_{21}^\top \\ \mathbf{K}_{21} & \mathbf{K}_{22} \end{bmatrix} \quad \text{and} \quad \mathbf{C} = \begin{bmatrix} \mathbf{W} \\ \mathbf{K}_{21} \end{bmatrix}.$$

The Nyström method uses $\mathbf{W} \in \mathbb{R}^{l \times l}$ and $\mathbf{C} \in \mathbb{R}^{m \times l}$ to generate the approximation $\widetilde{\mathbf{K}} = \mathbf{C}\mathbf{W}^\dagger\mathbf{C}^\top \approx \mathbf{K}$.

(a) Show that $\mathbf{W}$ is SPSD and that $\|\mathbf{K} - \widetilde{\mathbf{K}}\|_F = \|\mathbf{K}_{22} - \mathbf{K}_{21}\mathbf{W}^\dagger\mathbf{K}_{21}^\top\|_F$.

(b) Let $\mathbf{K} = \mathbf{X}^\top\mathbf{X}$ for some $\mathbf{X} \in \mathbb{R}^{N \times m}$, and let $\mathbf{X}' \in \mathbb{R}^{N \times l}$ be the first l columns of $\mathbf{X}$. Show that $\widetilde{\mathbf{K}} = \mathbf{X}^\top\mathbf{P}_{U_{X'}}\mathbf{X}$, where $\mathbf{P}_{U_{X'}}$ is the orthogonal projection onto the span of the left singular vectors of $\mathbf{X}'$.

(c) Is $\widetilde{\mathbf{K}}$ SPSD?

(d) If $\text{rank}(\mathbf{K}) = \text{rank}(\mathbf{W}) = r \ll m$, show that $\widetilde{\mathbf{K}} = \mathbf{K}$. Note: this statement holds whenever $\text{rank}(\mathbf{K}) = \text{rank}(\mathbf{W})$, but is of interest mainly in the low-rank setting.

(e) If $m = 20\text{M}$ and $\mathbf{K}$ is a dense matrix, how much space is required to store $\mathbf{K}$ if each entry is stored as a double? How much space is required by the Nyström method if $l = 10\text{K}$?

15.5 Expression for $\mathbf{K}_{LLE}$. Show the connection between LLE and KPCA by deriving the expression for $\mathbf{K}_{LLE}$.

15.6 Random projection, PCA, and nearest neighbors.

(a) Download the MNIST test set of handwritten digits at:

`http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz.`

Create a data matrix $\mathbf{X} \in \mathbb{R}^{N \times m}$ from the first $m = 2{,}000$ instances of this dataset (the dimension of each instance should be $N = 784$).

(b) Find the ten nearest neighbors for each point in $\mathbf{X}$, that is, compute $\mathcal{N}_{i,10}$ for $1 \le i \le m$, where $\mathcal{N}_{i,t}$ denotes the set of the t nearest neighbors for the

ith datapoint and nearest neighbors are defined with respect to the L_2 norm. Also compute $\mathcal{N}_{i,50}$ for all i.

(c) Generate $\tilde{\mathbf{X}} = \mathbf{A}\mathbf{X}$, where $\mathbf{A} \in \mathbb{R}^{k \times N}$, $k = 100$ and entries of $\mathbf{A}$ are sampled independently from the standard normal distribution. Find the ten nearest neighbors for each point in $\tilde{\mathbf{X}}$, that is, compute $\tilde{\mathcal{N}}_{i,10}$ for $1 \leq i \leq m$.

(d) Report the quality of approximation by computing $\text{score}_{10} = \frac{1}{m}\sum_{i=1}^{m} |\mathcal{N}_{i,10} \cap \tilde{\mathcal{N}}_{i,10}|$. Similarly, compute $\text{score}_{50} = \frac{1}{m}\sum_{i=1}^{m} |\mathcal{N}_{i,50} \cap \tilde{\mathcal{N}}_{i,10}|$.

(e) Generate two plots that show score_{10} and score_{50} as functions of k (i.e., perform steps (c) and (d) for $k = \{1, 10, 50, 100, 250, 500\}$). Provide a one- or two-sentence explanation of these plots.

(f) Generate similar plots as in (e) using PCA (with various values of k) to generate $\tilde{\mathbf{X}}$ and subsequently compute nearest neighbors. Are the nearest neighbor approximations generated via PCA better or worse than those generated via random projections? Explain why.