Exercises for the course
Machine Learning 2
Summer semester 2021

Abteilung Maschinelles Lernen
Institut für Softwaretechnik und theoretische Informatik
Fakultät IV, Technische Universität Berlin
Prof. Dr. Klaus-Robert Müller
Email: klaus-robert.mueller@tu-berlin.de

Exercise Sheet 1

Exercise 1: Symmetries in LLE (25 P)

The Locally Linear Embedding (LLE) method takes as input a collection of data points $\vec{x}_1, \ldots, \vec{x}_N \in \mathbb{R}^d$ and embeds them in some low-dimensional space. LLE operates in two steps, with the first step consisting of minimizing the objective

$$\mathcal{E}(w) = \sum_{i=1}^{N} \left\| \vec{x}_i - \sum_j w_{ij} \vec{x}_j \right\|^2$$

where w is a collection of reconstruction weights subject to the constraint $\forall i : \sum_j w_{ij} = 1$, and where $\sum_j$ sums over the K nearest neighbors of the data point $\vec{x}_i$. The solution that minimizes the LLE objective can be shown to be invariant to various transformations of the data.

Show that invariance holds in particular for the following transformations:

(a) Replacement of all $\vec{x}_i$ with $\alpha \vec{x}_i$, for an $\alpha \in \mathbb{R}^+ \setminus \{0\}$,

(b) Replacement of all $\vec{x}_i$ with $\vec{x}_i + \vec{v}$, for a vector $\vec{v} \in \mathbb{R}^d$,

(c) Replacement of all $\vec{x}_i$ with $U\vec{x}_i$, where U is an orthogonal $d \times d$ matrix.

Exercise 2: Closed form for LLE (25 P)

In the following, we would like to show that the optimal weights w have an explicit analytic solution. For this, we first observe that the objective function can be decomposed as a sum of as many subobjectives as there are data points:

$$\mathcal{E}(w) = \sum_{i=1}^{N} \mathcal{E}_i(w) \qquad \text{with} \quad \mathcal{E}_i(w) = \left\| \vec{x}_i - \sum_j w_{ij} \vec{x}_j \right\|^2$$

Furthermore, because each subobjective depends on different parameters, they can be optimized independently. We consider one such subobjective and for simplicity of notation, we rewrite it as:

$$\mathcal{E}_i(w) = \left\| \vec{x} - \sum_{j=1}^{K} w_j \vec{\eta}_j \right\|^2$$

where $\vec{x}$ is the current data point (we have dropped the index i), where $\eta = (\vec{\eta}_1, \ldots, \vec{\eta}_K)$ is a matrix of size $K \times d$ containing the K nearest neighbors of $\vec{x}$, and w is the vector of size K containing the weights to optimize and subject to the constraint $\sum_{j=1}^{K} w_j = 1$.

(a) *Prove* that the optimal weights for $\vec{x}$ are found by solving the following optimization problem:

$$\min_w \quad w^\top C w \qquad \text{subject to} \quad w^\top \mathbb{1} = 1.$$

where $C = (\mathbb{1}\vec{x}^\top - \eta)(\mathbb{1}\vec{x}^\top - \eta)^\top$ is the covariance matrix associated to the data point $\vec{x}$ and $\mathbb{1}$ is a vector of ones of size K.

(b) *Show* using the method of Lagrange multipliers that the minimum of the optimization problem found in (a) is given analytically as:

$$w = \frac{C^{-1}\mathbb{1}}{\mathbb{1}^\top C^{-1}\mathbb{1}}.$$

(c) *Show* that the optimal w can be equivalently found by solving the equation $Cw = \mathbb{1}$ and then rescaling w such that $w^\top \mathbb{1} = 1$.

Exercise 3: SNE and Kullback-Leibler Divergence (25 P)

SNE is an embedding algorithm that operates by minimizing the Kullback-Leibler divergence between two discrete probability distributions p and q representing the input space and the embedding space respectively. In 'symmetric SNE', these discrete distributions assign to each pair of data points (i, j) in the dataset the probability scores p_{ij} and q_{ij} respectively, corresponding to how close the two data points are in the input and embedding spaces. Once the exact probability functions are defined, the embedding algorithm proceeds by optimizing the function:

$$\begin{aligned} C &= D_{\mathrm{KL}}(p \,\|\, q) \\ &= \sum_{i=1}^{N} \sum_{j=1}^{N} p_{ij} \log\left(\frac{p_{ij}}{q_{ij}}\right) \end{aligned}$$

where p and q are subject to the constraints $\sum_{i=1}^{N} \sum_{j=1}^{N} p_{ij} = 1$ and $\sum_{i=1}^{N} \sum_{j=1}^{N} q_{ij} = 1$. Specifically, the algorithm minimizes q which itself is a function of the coordinates in the embedded space. Optimization is typically performed using gradient descent.

In this exercise, we derive the gradient of the Kullback-Leibler divergence, first with respect to the probability scores q_{ij}, and then with respect to the embedding coordinates of which q_{ij} is a function.

(a) *Show* that

$$\frac{\partial C}{\partial q_{ij}} = -\frac{p_{ij}}{q_{ij}}. \tag{1}$$

(b) The probability matrix q is now reparameterized using a 'softargmax' function:

$$q_{ij} = \frac{\exp(z_{ij})}{\sum_{k=1}^{N} \sum_{l=1}^{N} \exp(z_{kl})}$$

The new variables z_{ij} can be interpreted as unnormalized log-probabilities. *Show* that

$$\frac{\partial C}{\partial z_{ij}} = -p_{ij} + q_{ij}. \tag{2}$$

(c) *Explain* which of the two gradients, (1) or (2), is the most appropriate for practical use in a gradient descent algorithm. Motivate your choice, first in terms of the stability or boundedness of the gradient, and second in terms of the ability to maintain a valid probability distribution during training.

(d) The scores z_{ij} are now reparameterized as

$$z_{ij} = -\|\vec{y}_i - \vec{y}_j\|^2$$

where the coordinates $\vec{y}_i, \vec{y}_j \in \mathbb{R}^h$ of data points in embedded space now appear explicitly. *Show* using the chain rule for derivatives that

$$\frac{\partial C}{\partial \vec{y}_i} = \sum_{j=1}^{N} 4\,(p_{ij} - q_{ij}) \cdot (\vec{y}_i - \vec{y}_j).$$

Exercise 4: Programming (25 P)

Download the programming files on ISIS and follow the instructions.

Implementing Locally Linear Embedding (25 P)

In this programming homework we will implement locally linear embedding (LLE) and experiment with it on the swiss roll dataset. In particular, the effects of neighbourhood size and noise on the quality of the embedding will be analyzed.

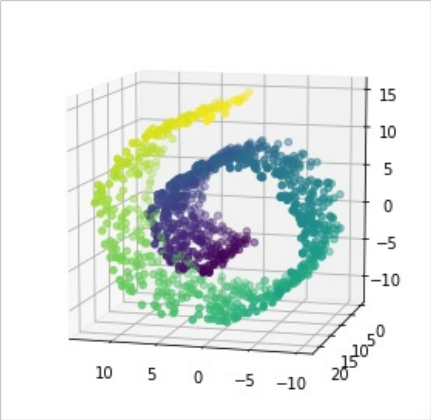
In [1]:

```
import numpy as np
import matplotlib
%matplotlib inline
from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d.axes3d import Axes3D
import sklearn,sklearn.datasets
```

The following code plots the swiss roll dataset (a commonly used dataset to test LLE) with `N=1000` data points and a noise parameter of `0.25` .

In [2]:

```
X,T = sklearn.datasets.make_swiss_roll(n_samples=1000, noise=0.25)
plt.figure(figsize=(5,5))
ax = plt.gca(projection='3d')
ax.view_init(elev=10., azim=105)
ax.scatter(X[:,0],X[:,1],X[:,2],c=T)
plt.show()
```



Although the dataset is in three dimensions, the points follow a two-dimensional low-dimensional structure. The goal of embedding algorithms is to extract this underlying structure, in this case, unrolling the swiss roll into a two-dimensional Euclidean space.

In the following, we consider a simple implementation of LLE. You are required to complete the code by writing the portion where the optimal reconstruction weights are extracted. *(Hint: During computation, you need to solve an equation of the type Cw=**1**, where **1** is a column vector (1,1,...,1). In case k>d i.e. the size of the neighbourhood is larger than the number of dimensions of the input space, it is necessary to regularize the matrix C. You can do this by adding positive terms on the diagonal. A good starting point is 0.05.)*

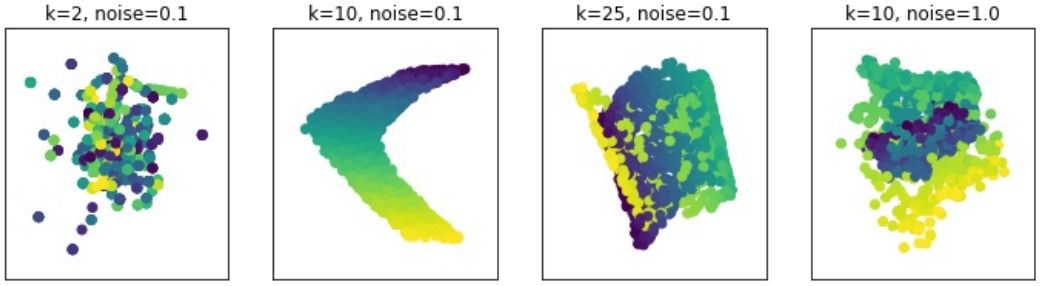
In [3]:

```
def LLE(X,k):
    N = len(X)
    W = np.zeros([N,N])

    for i in range(N):
        # ---------------------------------------
        # TODO: Replace by your code
        # ---------------------------------------
        import solution
        w,ind = solution.lle(X,i,k)
        # ---------------------------------------
        W[i,ind] = w

    M = np.identity(N) - W - W.T + np.dot(W.T,W)
    E = np.linalg.svd(M)[0][:,-3:-1]

    return E
```

You can now test your implementation on the swiss roll dataset and vary the noise in the data and the parameter k of the LLE algorithm. Results are shown below:

In [4]:

```
f = plt.figure(figsize=(12,3))
for t,(k,noise) in enumerate([(2,0.1),(10,0.1),(25,0.1),(10,1)]):
    X,T = sklearn.datasets.make_swiss_roll(n_samples=1000, noise=noise)
    embedding = LLE(X,k=k)
    ax = f.add_subplot(1,4,t+1)
    ax.set_title('k=%d, noise=%.1f'%(k,noise))
    ax.set_xticks([])
    ax.set_yticks([])
    ax.scatter(embedding[:,0],embedding[:,1],c=T)
```



It can be observed that the parameter k must be carefully tuned to have sufficiently many neighbors for stability but also not too many. We can further observe that LLE works well as long as the noise in the data remains low enough.

Exercise Sheet 1

1 Symmetries in LLE

a) $$\mathcal{E}'(W) = \sum_{i=1}^{N} \| \alpha \vec{x}_i - \sum_j w_{ij} \alpha \vec{x}_j \|^2 = \sum_{i=1}^{N} \| \alpha (\vec{x}_i - \sum_j w_{ij} \vec{x}_j) \|^2$$

$$= \alpha^2 \sum_{i=1}^{N} \| \vec{x}_i - \sum_j w_{ij} \vec{x}_j \|^2 = \alpha^2 \mathcal{E}(W)$$

$$\operatorname{argmin} \mathcal{E}'(W) = \operatorname{argmin} \alpha^2 \mathcal{E}(W)$$

b) $$\mathcal{E}'(W) = \sum_{i=1}^{N} \| \vec{x}_i + \vec{v} - \sum_j w_{ij} (\vec{x}_j + \vec{v}) \|^2$$

$$= \sum_{i=1}^{N} \| \vec{x}_i + \vec{v} - \sum_j w_{ij} \vec{x}_j - \sum_j w_{ij} \vec{v} \|^2$$

$$= \sum_{i=1}^{N} \| \vec{x}_i + \vec{v} - \sum_j w_{ij} \vec{x}_j - \vec{v} \underbrace{\sum_j w_{ij}}_{=1} \|^2$$

$$= \sum_{i=1}^{N} \| \vec{x}_i + \vec{v} - \vec{v} - \sum_j w_{ij} \vec{x}_j \|^2$$

$$= \sum_{i=1}^{N} \| \vec{x}_i - \sum_j w_{ij} \vec{x}_j \|^2 = \mathcal{E}(W)$$

c) $$\mathcal{E}'(W) = \sum_{i=1}^{N} \| U \vec{x}_i - \sum_j w_{ij} U \vec{x}_j \|^2 = \sum_{i=1}^{N} \| U (\vec{x}_i - \sum_j w_{ij} \vec{x}_j) \|^2$$

$$= \sum_{i=1}^{N} \left(U (\vec{x}_i - \sum_j w_{ij} \vec{x}_j) \right)^T \left(U (\vec{x}_i - \sum_j w_{ij} \vec{x}_j) \right)$$

$$= \sum_{i=1}^{N} (\vec{x}_i - \sum_j w_{ij} \vec{x}_j)^T \underbrace{U^T \cdot U}_{=\mathbb{1}} (\vec{x}_i - \sum_j w_{ij} \vec{x}_j)$$

$$= \sum_{i=1}^{N} (\vec{x}_i - \sum_j w_{ij} \vec{x}_j)^T (\vec{x}_i - \sum_j w_{ij} \vec{x}_j)$$

$$= \sum_{i=1}^{N} \| \vec{x}_i - \sum_j w_{ij} \vec{x}_j \|^2 = \mathcal{E}(W)$$

2 Closed Form for LLE

a) $\mathcal{E}_i(w) = \| \vec{x} - \sum_{j=1}^{K} w_j \vec{\eta}_j \|^2 = \| \vec{x}\, w^T \underline{1} - \eta^T w \|^2$

$= \| (\vec{x}\, \underline{1}^T w - \eta^T w) \|^2 = \| (\vec{x}\, \underline{1}^T - \eta^T)\, w \|^2$

$= \left((\vec{x}\, \underline{1}^T - \eta^T) w \right)^T \left((\vec{x}\, \underline{1}^T - \eta^T) w \right)$

$= w^T (\vec{x}\, \underline{1}^T - \eta^T)^T (\vec{x}\, \underline{1}^T - \eta^T) w$

$= w^T (\underline{1}\, \vec{x}^T - \eta)(\vec{x}\, \underline{1}^T - \eta^T) w$

$= w^T (\underline{1}\, \vec{x}^T - \eta)(\underline{1}\, \vec{x}^T - \eta)^T w = w^T C w$

b) $\mathcal{L}(w, \lambda) = w^T C w + \lambda (w^T \underline{1} - 1) = w^T C w + \lambda w^T \underline{1} - \lambda$

$\frac{\partial \mathcal{L}(w,\lambda)}{\partial \lambda} = w^T \underline{1} - 1 = 0 \Leftrightarrow w^T \underline{1} = 1$

$\frac{\partial \mathcal{L}(w,\lambda)}{\partial w} = \frac{\partial w^T C w}{\partial w} + \frac{\partial \lambda \underline{1}^T w}{\partial w}$

$= \frac{\partial w^T C w}{\partial w^T} + \frac{\partial w^T C w}{\partial w} + \lambda \underline{1}^T$

$= \frac{\partial w C^T w^T}{\partial w} + \frac{\partial w^T C w}{\partial w} + \lambda \underline{1}^T$

$= C^T w^T + C w + \lambda \underline{1}^T = w^T C^T + w^T C + \lambda \underline{1}^T = w^T (C + C^T) + \lambda \underline{1}^T = 0$

C symmetric $\Rightarrow C^T = C$

$\Rightarrow w^T \cdot 2C + \lambda \underline{1}^T = 0 \Leftrightarrow 2(w^T C) + \lambda \underline{1}^T = 2(C^T w)^T + \lambda \underline{1}^T$

$= (2(C^T w)^T + \lambda \underline{1}^T)^T = 2Cw + \lambda \underline{1} \Leftrightarrow 2Cw = -\lambda \underline{1}$

$\Leftrightarrow Cw = -\frac{\lambda}{2} \underline{1} \Leftrightarrow w = -\frac{\lambda}{2} C^{-1} \underline{1} \Leftrightarrow 1 = w^T \underline{1} \Leftrightarrow \underline{1}^T w = -\frac{\lambda}{2} \underline{1}^T C^{-1} \underline{1}$

$\Leftrightarrow 1 = -\frac{\lambda}{2} \underline{1}^T C^{-1} \underline{1} \Leftrightarrow \lambda = -\frac{2}{\underline{1}^T C^{-1} \underline{1}}$

$\Leftrightarrow w = \frac{C^{-1} \underline{1}}{\underline{1}^T C^{-1} \underline{1}}$

c) $Cw = \mathbb{1}$, $w^T \mathbb{1} = 1$

$\Rightarrow w = C^{-1}\mathbb{1}$

$\Leftrightarrow \frac{w}{w^T\mathbb{1}} = \frac{C^{-1}\mathbb{1}}{\mathbb{1}^T C^{-1}\mathbb{1}}$

3 SNE and Kullback-Leibler Divergence

a) $$\frac{\partial C}{\partial q_{ij}} = \frac{\partial}{\partial q_{ij}} \sum_{i=1}^{N} \sum_{j=1}^{N} p_{ij} \log\left(\frac{p_{ij}}{q_{ij}}\right) = \frac{\partial}{\partial q_{ij}} p_{ij} \log\left(\frac{p_{ij}}{q_{ij}}\right)$$

$$= \frac{\partial}{\partial q_{ij}} p_{ij}\left(\log(p_{ij}) - \log(q_{ij})\right) = -p_{ij}\frac{1}{q_{ij}} = -\frac{p_{ij}}{q_{ij}}$$

b) $$\frac{\partial}{\partial z_{ij}} \sum_{i=1}^{N}\sum_{j=1}^{N} p_{ij} \log\left(\frac{p_{ij}\sum_{k=1}^{N}\sum_{l=1}^{N} e^{z_{kl}}}{e^{z_{ij}}}\right)$$

$$= \frac{\partial}{\partial z_{ij}} \sum_{i=1}^{N}\sum_{j=1}^{N} p_{ij}\left(\log\left(p_{ij}\sum_{k=1}^{N}\sum_{l=1}^{N} e^{z_{kl}}\right) - \log e^{z_{ij}}\right)$$

$$= \frac{\partial}{\partial z_{ij}} \sum_{i=1}^{N}\sum_{j=1}^{N} p_{ij}\left(\log(p_{ij}) + \log\left(\sum_{k=1}^{N}\sum_{l=1}^{N} e^{z_{kl}}\right) - z_{ij}\right)$$

$$= \frac{\partial}{\partial z_{ij}} \sum_{i=1}^{N}\sum_{j=1}^{N} p_{ij} \log\left(\sum_{k=1}^{N}\sum_{l=1}^{N} e^{z_{kl}}\right) + \sum_{i=1}^{N}\sum_{j=1}^{N}(-p_{ij} z_{ij})$$

$$= \frac{\partial}{\partial z}\left(\log\left(\sum_{k=1}^{N}\sum_{l=1}^{N} e^{z_{kl}}\right) - p_{ij} z_{ij}\right)$$

$$= \frac{\partial}{\partial z_{ij}} \log\left(\sum_{k=1}^{N}\sum_{l=1}^{N} e^{z_{kl}}\right) - p_{ij} z_{ij} = q_{ij} - p_{ij} = -p_{ij} + q_{ij}$$

c) $q_{ij} \in q_{ij} - \beta \frac{\partial C}{\partial q_{ij}} \rightarrow -\frac{p_{ij}}{q_{ij}}$ If the probability of the embedded space gets very close to 0, the gradient becomes very large
→ not stable, not bounded

$z_{ij} \in z_{ij} - \beta \frac{\partial C}{\partial z_{ij}} \rightarrow -p_{ij} + q_{ij}$ → The gradient will never become infinite → stable and bounded

d) $$\frac{\partial C}{\partial \vec{y}_i} = \sum_{j=1}^{N} \frac{\partial C}{\partial z_{ij}} \frac{\partial z_{ij}}{\partial \vec{y}_i} + \frac{\partial C}{\partial z_{ji}} \cdot \frac{\partial z_{ji}}{\partial \vec{y}_i}$$

$$= \sum_{j=1}^{N} (-p_{ij} + q_{ij}) \frac{\partial - \| \vec{y}_i - \vec{y}_j \|^2}{\partial \vec{y}_i} + (-p_{ji} + q_{ji}) \frac{\partial - \| \vec{y}_j - \vec{y}_i \|^2}{\partial \vec{y}_i}$$

$$= \sum_{j=1}^{N} (-p_{ij} + q_{ij})(-2(\vec{y}_i - \vec{y}_j)) + (-p_{ji} + q_{ji})(-2(\vec{y}_j - \vec{y}_i))$$

$$= \sum_{j=1}^{N} 2(-(p_{ij} - q_{ij}))(-2(\vec{y}_i - \vec{y}_j)) = \sum_{j=1}^{N} 4(p_{ij} - q_{ij})(\vec{y}_i - \vec{y}_j)$$