Assignment 9

Statistical Machine Learning

Moritz Haas / Prof. Ulrike von Luxburg

Summer term 2022 — due on **June 27th at 12:00**

Exercise 1 (Exact recovery from a PCA is impossible, 2+1 points)

Let x be a vector in $\mathbb{R}^n$. A matrix $W \in \mathbb{R}^{d \times n}$, with $d < n$, induces a map $x \to Wx \in \mathbb{R}^d$. Such map can be interpreted as a lower dimension representation of x. We will call such matrix a *compression matrix*. Similarly if $y \in \mathbb{R}^d$, a matrix $U \in \mathbb{R}^{n \times d}$ induces a map $y \to Uy \in \mathbb{R}^n$. Such map can be interpreted as the *reconstruction* of y in $\mathbb{R}^n$. Clearly PCA provides one possible way to define such W and U.

(a) Let $W \in \mathbb{R}^{d \times n}$ be an arbitrary compression matrix. Show that there exist $u, v \in \mathbb{R}^n$ such that $u \neq v$ and $Wu = Wv$.

(b) Conclude that exact recovery from a linear compression scheme is impossible.

Exercise 2 (PCA and Random Projections, 1+1+1+1+2+1+1 points +3 bonus points)

In this exercise we explore linear projections onto l-dimensional subspaces of a d-dimensional Euclidean space ($l \ll d$), using the USPS data set. A recap on projections can be found in the lecture slides, 1349ff.
Given a d-dimensional data set, a very popular projection is the Principal Component Analysis (PCA), which maximizes the recovered variance with respect to the given data set. Another option is to project onto a random l-dimensional subspace. This can be done by independently drawing l random vectors v_i with $\|v_i\| = 1$ uniformly on the sphere S^{d-1}. Optionally, $v_1, \dots, v_l$ can be transformed into orthonormal vectors. Then define the matrix $V \in \mathbb{R}^{d \times l}$, where the i-th column consists of the i-th transformed random vector. Now V^T yields the projection into $\mathbb{R}^l$.
You can either use `PCA` and `GaussianRandomProjections` as provided by `scikit-learn`, or you can implement these methods yourself. If you choose to implement these methods yourself, it will be awarded with 1.5 bonus points per method.

(a) Use PCA and random projections to reduce the dataset's dimensionality to $l = 2$. Visualize the projected data. Use different colors for different digits.

(b) Visualize the principal components and random directions as images in feature space.

We want to compare the reconstruction error of PCA and random projections. To compute the reconstruction error, we have to transform the projected data back into the original space. `PCA` offers a method `inverse_transform` that performs this task. For any two matrices of original points $X \in \mathbb{R}^{n \times d}$ and projected points $X_{projected} \in \mathbb{R}^{n \times l}$, the ipynb-template offers the function `reconstruct` that computes the optimal reconstruction matrix $U \in \mathbb{R}^{d \times l}$ and returns the reconstructed points as a $n \times d$ matrix.

(c) For different values of l, visualize the reconstructed images in the feature space.

(d) Compare the reconstruction error of PCA and random projections. The reconstruction error is the Euclidean distance between an original and a reconstructed point. Plot the average reconstruction error against different dimensions l.

The reconstruction error is not the only measure that aims to describe the quality of an embedding. There are also measures of distance distortion. We now want to compare the average and maximum distance distortion of PCA and random projections. If you are interested, you can read about different measures of distortion in Vankadara & von Luxburg (2018).[1]

(e) The distance distortion for a pair of points (x, y) is the ratio of point distances between the original points (x, y) and the projected points $(\tilde{x}, \tilde{y})$,

$$\rho = \max\left\{\frac{d(x,y)}{d(\tilde{x},\tilde{y})}, \frac{d(\tilde{x},\tilde{y})}{d(x,y)}\right\}.$$

Implement a function `pairwise_distortion` that computes the distance distortion for all pairs of points. Compute and plot the average and maximum distance distortion for PCA and random projections against different dimensions l.

(f) Multiply the first dimension of the data by 500, and repeat (d) and (e) for the modified data set. Can you explain what you observe?

(g) Summarize the advantages and disadvantages of PCA and random projections.

Hint: This exercise relates to the Johnson-Lindenstrauss Lemma (lecture slides 610). For random projections, especially if you want to implement them yourself, refer to the book by Shai Shalev-Shwartz and Shai Ben-David.

Exercise 3 (Implement Kernel PCA, 4 + 2 points) In this exercise you will implement a Kernel Principal Component Analysis (kPCA). You find the data set `data` with 3 labels plotted in the ipynb-template.

(a) Implement kPCA as a function `kernel_PCA(K, l=2)`. It takes as input the $n \times n$-kernel matrix K and the number l of dimensions you want to project on. It returns the low-dimensional representation points in $\mathbb{R}^l$ computed with the kPCA-algorithm on page 579 of the slides.

Note two things:

1) Due to rounding errors, it can happen be that `numpy.linalg.eig` returns complex eigenvectors and eigenvalues. If that is the case a solution is to do
`eigenvalues, eigenvectors = eigenvalues.real, eigenvectors.real`
2) The matrix $\mathbb{1}_n$ is the matrix $n \times n$ with all entries equal to $1/n$.

(b) Use `sklearn.gaussian_process.kernels.RBF` and compute K for `data` using a RBF kernel with $\sigma = 5$. Then compute the kernel PCA for K with $l = 3$. Plot the first 2 dimensions of the 3-dimensional representations in a 2D figure. Then plot the representation points in a 3D figure.

Exercise 4 (Feedback Poll: Week 8, 0 bonus points but 1 gratitude)
Weekly feedback survey on Ilias about the lecture, tutorials and exercise sheets. If you participate, it helps us in understanding what we can improve. Future polls will be anonymous to ensure open feedback. The poll asks about the lectures that cover the topics treated in the respective sheet.

[1] `http://www.tml.cs.uni-tuebingen.de/team/leena_chennuru/VankadaraLuxburg18.pdf`