Machine Learning
Exercise 5

Marc Toussaint
Machine Learning & Robotics lab, U Stuttgart
Universitätsstraße 38, 70569 Stuttgart, Germany

May 24, 2017

Principal Component Analysis (PCA) Intro

Dimensionality reduction is the process of projecting data-points $x \in \mathbb{R}^d$ onto a lower-dimensional space $z \in \mathbb{R}^p$, with $p < d$. This procedure serves multiple purposes, in supervised learning, unsupervised learning, data visualization, etc..

A key aspect and objective of all dimensionality reduction algorithms is that they want to preserve as much variance of the original data as possible: the goal is to learn a lower-dimensional representation while retaining the maximum information about the original data.

In standard PCA, we approximate a vector x with an affine projection of its latent representation z, as follows:

$$x \approx Vz + \mu, \tag{1}$$

where $V \in \mathbb{R}^{d \times p}$ is an *orthonormal* matrix which specifies an axis rotation, and $\mu \in \mathbb{R}^d$ which specifies a translation. Given a dataset $D = \{x_i\}_{i=1}^n$, the projection parameters μ, and the projections z_i are obtained by minimizing the difference between the original data and the respective approximations:

$$\widehat{\mu}, \widehat{z}_{1:n} = \operatorname*{argmin}_{\mu, z_{1:n}} \sum_i \|Vz_i + \mu - x_i\|^2 \tag{2}$$

1 PCA optimality principles

a) Find the optimal latent representation z_i, as a function of V and μ.

b) Find *an* optimal offset μ.

(Hint: there is a whole subspace of solutions to this problem. Verify that your solution is consistent with (i.e. contains) $\mu = \frac{1}{n}\sum_i x_i$).

c) Find optimal projection vectors $\{v_i\}_{i=1}^p$, which make up the projection matrix

$$V = \begin{bmatrix} | & & | \\ v_1 & \dots & v_p \\ | & & | \end{bmatrix} \tag{3}$$

Guide:

1. Given a projection V, any vector can be split in orthogonal components which belong to the projected subspace and its complement (which we call W). $x = VV^\top x + WW^\top x$.

2. For simplicity, let us work with the centered datapoints $\tilde{x}_i = x_i - \mu$.

3. The optimal projection V is that which minimizes the discarded components $WW^\top \tilde{x}_i$.

$$\widehat{V} = \operatorname*{argmin}_V \sum_{i=1}^n \|WW^\top \tilde{x}_i\|^2 = \sum_{i=1}^n \|\tilde{x}_i - VV^\top \tilde{x}_i\|^2 \tag{4}$$

4. Don't try to solve computing gradients and setting them to zero. Instead, use the fact that $VV^\top = \sum_{i=1}^p v_i v_i^\top$, and the singular value decomposition of $\sum_{i=1}^n \tilde{x}_i \tilde{x}_i^\top = \tilde{X}^\top \tilde{X} = EDE^T$.

d) In the above, is the orthonormality of V an essential assumption?

e) Prove that you can compute the V also from the SVD of X (instead of $X^\top X$).

2 PCA and reconstruction on the Yale face database

On the webpage find and download the Yale face database http://ipvs.informatik.uni-stuttgart.de/mlr/marc/teaching/data/yalefaces.tgz. (Optionally use `yalefaces_cropBackground.tgz`, which is slightly cleaned version of the same dataset). The file contains gif images of 165 faces.

a) Write a routine to load all images into a big data matrix $X \in \mathbb{R}^{165\times 77760}$, where each row contains a gray image.

In Octave, images can easily be read using `I=imread("subject01.gif");` and `imagesc(I);`. You can loop over files using `files=dir(".");` and `files(:).name`. Python tips:

```
import matplotlib.pyplot as plt
import scipy as sp
plt.imshow(plt.imread(file_name))
u, s, vt = sp.sparse.linalg.svds(X, k=neigenvalues)
```

b) Compute the mean face $\mu = \frac{1}{n}\sum_i x_i$ and center the whole data matrix, $\tilde X = X - \mathbf{1}_n \mu^\top$.

c) Compute the singular value decomposition $\tilde X = UDV^\top$ for the centered data matrix.

In Octave/Matlab, use `[U, S, V] = svd(X, "econ")`, where the `"econ"` ensures you don't run out of memory.

In python, use

```
import scipy.sparse.linalg as sla;
u, s, vt = sla.svds(X, k=num_eigenvalues)
```

d) Find the p-dimensional representations $Z = \tilde X V_p$, where $V_p \in \mathbb{R}^{77760\times p}$ contains only the first p columns of V (Depending on which language / library you use, verify that the eigenvectors are returned in eigenvalue-descending order, otherwise you'll have to find the correct eigenvectors manually). Assume $p = 60$. The rows of Z represent each face as a p-dimensional vector, instead of a 77760-dimensional image.

e) Reconstruct all faces by computing $X' = \mathbf{1}_n \mu^\top + \boldsymbol{Z} V_p^\top$ and display them; Do they look ok? Report the reconstruction error $\sum_{i=1}^n \|x_i - x'_i\|^2$.

Repeat for various PCA-dimensions $p = 5, 10, 15, 20 \dots$.

BONUS) How is the value of p related to the variance / information which is lost through the low-dim projection?