Kernel Principal Component Analysis

Exercise T3.1: Kernel PCA for non-linear PCA (tutorial)

(a) How can a non-linear transformation allow PCA to capture non-linear correlations between the variables?

(b) What is the kernel trick and what is the kernel matrix?

(c) How do we construct the transformed eigenvalue problem for Kernel PCA?

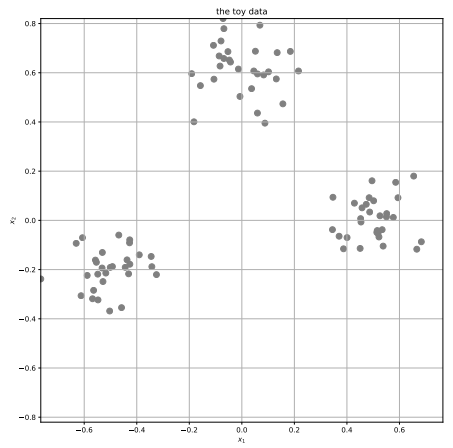
(d) How do we find non-linear structure in data using Kernel PCA?

Exercise H3.1: Kernel PCA: Toy Data (homework, 10 points)

(a) Create a dataset of 2-dimensional data points $\underline{\mathbf{x}}^{(\alpha)} = (x_1^{(\alpha)}, x_2^{(\alpha)})^\top$, $\alpha = 1, \ldots, 90$. The points represent iid samples from 3 different distributions with uncorrelated, normally distributed (SD=0.1) coordinate values. Draw 30 samples from each distribution. The only difference between the distributions is their mean:

- The first 30 samples (i.e. $\alpha = 1, \ldots, 30$) should be centered on $\langle \underline{\mathbf{x}}^{(\alpha)} \rangle_1 = (-0.5, -0.2)^\top$.
- The second subset (i.e. $\alpha = 31, \ldots, 60$) should have $\langle \underline{\mathbf{x}}^{(\alpha)} \rangle_2 = (\,0\,, 0.6)^\top$.
- The third subset (i.e. $\alpha = 61, \ldots, 90$) should have $\langle \underline{\mathbf{x}}^{(\alpha)} \rangle_3 = (0.5,\, 0\,)^\top$.

Solution



(b) Apply Kernel PCA using an RBF kernel with a suitable value for the width σ of the kernel[1] and calculate the coefficients for the representation of the eigenvectors (PCs) in the space spanned by the transformed data points.

$$k_{RBF}(\underline{\mathbf{x}}, \underline{\mathbf{x}}') = \exp\left(-\frac{\|\underline{\mathbf{x}} - \underline{\mathbf{x}}'\|_2^2}{2\sigma^2}\right)$$

Solution

```
# Assuming X is a matrix with 90 samples x 2 dimensions
# define RBF kernel,
# choose sigma the same as sigma for the random samples.
sigma = 0.1
kfunc = lambda x_a, x_b: exp(-norm(x_a-x_b)**2/(2*sigma**2))

# centering a given matrix
# K - row avg - col avg + matrix avg
def center_K(_K):
nrows, ncols = _K.shape
        return _K \
        - _K.mean(axis=1).reshape(nrows, 1) \
        - _K.mean(axis=0).reshape(1, ncols) \
        + _K.mean() # row, col, matrix average

# define the kernel matrix for a sample, given the reference
def calculate_K(new_sample,reference):
    p = reference.shape[0]
    q = new_sample.shape[0]
    K = zeros([q,p])
    for a in range(q):
        for b in range(p):
            K[a,b] = kfunc(new_sample[a,:],reference[b,:])
    return center_K(K)

# Calculate the kernel matrix
K = calculate_K(X, X)
# Solve the transformed eigenvalue problem
lamb, eig = eig(K/len(X))
# sort and normalize before projection...
```

(c) Visualize the first 8 PCs in the original 2-dimensional input space in the following way:

(i) Use equally spaced "test" gridpoints (in a rectangle $[\underline{\mathbf{a}}, \underline{\mathbf{b}}] \times [\underline{\mathbf{c}}, \underline{\mathbf{d}}] \subset \mathbb{R}^2$ which contains all training samples from (a) above).

(ii) Project each test point onto the first 8 eigenvectors in *feature space*[2].

[1] You might not be able to assess if one value for σ is better than another until after you've computed the eigenvalues or set up the subsequent visualization step. However, this shouldn't stop you from making an educated guess on what value to start with.

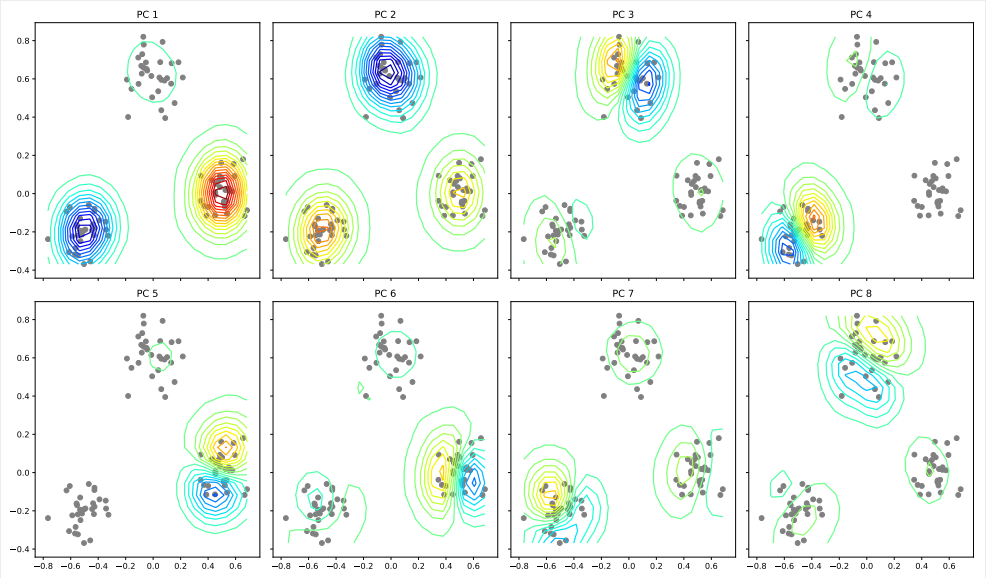
[2] i.e. *after* applying the kernel trick. Which pairs do you select for constructing the kernel matrix?

Hint: You have to ensure that the feature vectors of the test points are centered when calculating their PC projections, i.e. the kernel matrix *used for the projections* is centered.

(iii) Visualize the projections of the test points using contour lines in the original 2-dimensional input space. The contours should indicate points that yield the same projection onto the respective PC. You may also use a heat map or pseudo color plot (e.g. `pcolor`) to distinguish the different regions.

(iv) Add the 90 training points in the same plot (e.g. using small gray circles).

Solution



(v) What kind of roles do the different PCs play?

Solution

The first 2 PCs are sensitive to "which distribution" a point belongs to. After that the PCs become sensitive to the location of the points *within* a distribution (local information).

(vi) Discuss suitable applications for Kernel PCA.

Solution

Some applications for Kernel PCA:

- Visualization
- Denoising
- pre-processing for supervised methods (if you interpret the distributions above as categories, the first 2 PCs can be used to classify which category a point belongs to).

Total 10 points.