Exercises for the course
Machine Learning for Data Science
Winter Semester 2024/25

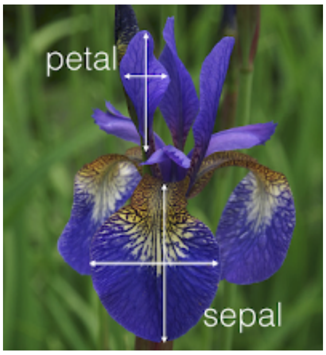
G. Montavon
Institute of Computer Science
Department of Mathematics and Computer Science
Freie Universität Berlin

Exercise Sheet 4 (programming part)

In [1]:
```
import numpy
import scipy
import utils
import sklearn
import sklearn.datasets
import sklearn.decomposition
import matplotlib
%matplotlib inline
from matplotlib import pyplot as plt
lcm = matplotlib.colors.ListedColormap
```

Exercise 2 (30 P)

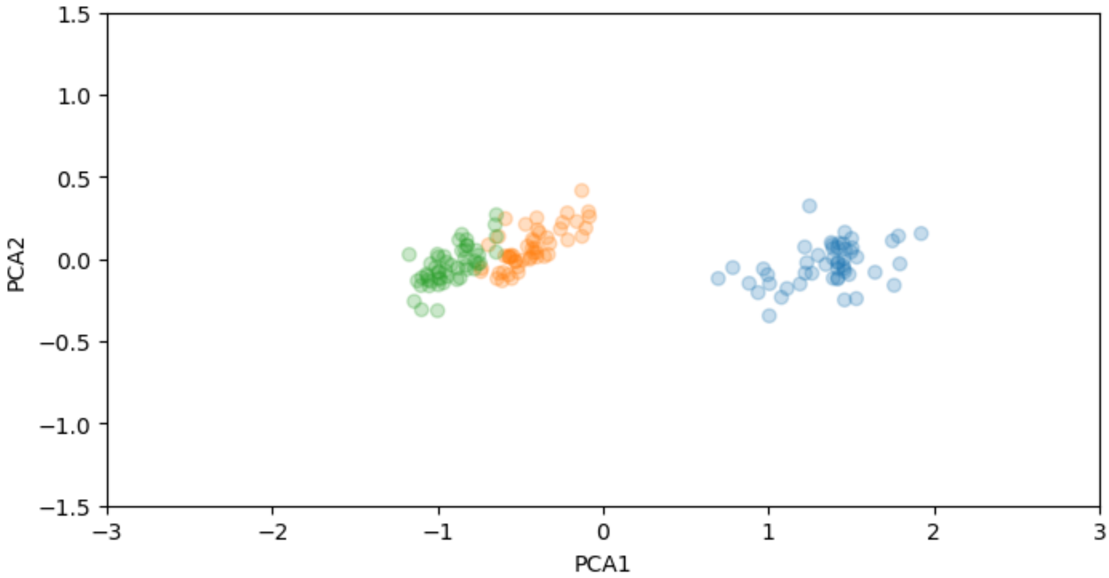
In this exercise we build a model of anomalies for the Iris dataset, i.e. the same dataset we have already considered in Exercise Sheet 3 when studying principal component analysis.



The Iris dataset contains 150 instances of iris plants with 4 measurements each, and can be stored in an array of size 150 x 4. We apply some nonlinear transformation to data in order to emphasize variation of small distances. The following cell loads the dataset and performs the stated normalization. It also generates a PCA representation of the data for plotting purposes.

In [2]:
```
dataset = sklearn.datasets.load_iris()

X,T = numpy.log(0.1+dataset['data']),dataset['target']

Z = sklearn.decomposition.PCA(n_components=2).fit_transform(X)

utils.preparefigure()
plt.scatter(*Z.T,c=T,alpha=0.25,cmap=lcm(['C0','C1','C2']))
```

Out[2]:
```
<matplotlib.collections.PathCollection at 0x7263b11231d0>
```



We will identify anomalies using the SVDD method presented in the lecture. This method builds an enclosing hypersphere of center $\boldsymbol{c} \in \mathbb{R}^d$ and of squared radius S. The dual optimization problem of SVDD, which have derived in the theoretical part of the homework can stated in matrix form as:

$$ \newcommand{\balpha}{\boldsymbol{\alpha}} \max_{\balpha}~~ \balpha^\top D - \balpha^\top XX^\top \balpha $$ where X is a data matrix of size $N \times d$ and where $D = (\|\boldsymbol{x}\|_1^2,\|\boldsymbol{x}\|_2^2,\dots,\|\boldsymbol{x}\|_N^2)$ is the vector containing the squared norm of each data point. This optimization problem is carried out subject to the constraints:

$$ \newcommand{\balpha}{\boldsymbol{\alpha}} \boldsymbol{1}^\top\balpha = 1 \quad \text{and} \quad \begin{pmatrix}-I\\I\end{pmatrix} \balpha \preceq \begin{pmatrix}\boldsymbol{0}\\C\end{pmatrix} $$ To solve this optimization problem, we can use the quadratic solver from `cvxopt`, specifically the function `cvxopt.solvers.qp`, which is described at the page https://cvxopt.org/userguide/coneprog.html. From the solution of the dual $\balpha$, the parameters of the original problem, i.e. the description of the enclosing hypersphere can be recovered as: $\boldsymbol{c} = \sum_i \alpha_i \boldsymbol{x}_i$ for the center of the hypersphere, and its the squared radius S can be recovered as the squared distance of any data point $ \boldsymbol{x}_i$ satisfying $0 < \alpha_i < C$ (note the strict inequalities) from the vector $\boldsymbol{c}$.

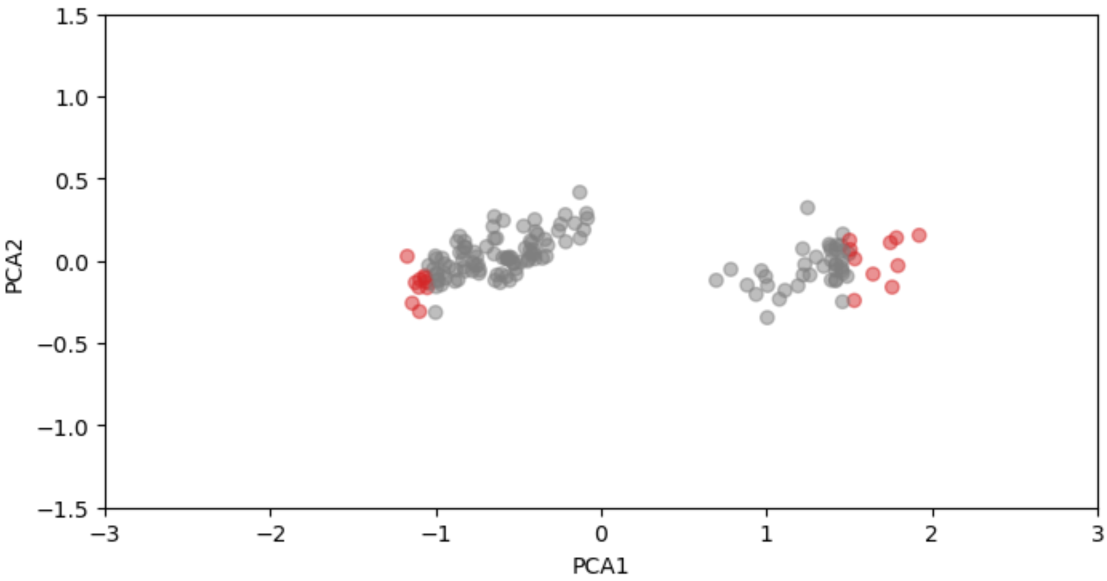
Task: Implement the functions `fit_predict` of the class `SVDD`. It should prepare the matrices and vectors to be fed to `cvxopt`, call `cvxopt` to solve the dual, recover the solution of the primal (the parameters $\boldsymbol{c}$ and S) from the solution of the dual, and return a boolean vector of size `N` indicating which point in the dataset is an outlier.

In [3]:
```
import cvxopt
import cvxopt.solvers

class SVDD:

    def __init__(self,C):
        self.C = C

    def fit_predict(self,X):

        # -----------------------------------
        # TODO: replace by your code
        # -----------------------------------
        import solution
        Y = solution.svdd_fit_predict(self,X)
        # -----------------------------------

        return Y
```

The SVDD can now be applied to the Iris dataset. Here, we consider the hyperparameter $C = 0.05$ which allows for a moderate number of outliers. The predictions are rendered in a PCA plot where each instance is color-coded according to its anomaly decision.

```
In [4]: Y = SVDD(0.05).fit_predict(X)

utils.preparefigure()
plt.scatter(*Z.T,c=Y,alpha=0.5,cmap=lcm(['C7','C3']));
```

```
     pcost       dcost       gap    pres   dres
 0: -1.9189e+01 -1.5132e+01  1e+03  3e+01  4e-15
 1: -2.7941e+00 -1.4237e+01  5e+01  1e+00  4e-15
 2: -1.6590e+00 -7.0095e+00  6e+00  2e-02  5e-16
 3: -1.7616e+00 -2.8439e+00  1e+00  3e-03  4e-16
 4: -1.8501e+00 -2.2643e+00  4e-01  5e-04  4e-16
 5: -1.9165e+00 -2.0969e+00  2e-01  1e-04  4e-16
 6: -1.9640e+00 -1.9932e+00  3e-02  9e-06  4e-16
 7: -1.9750e+00 -1.9773e+00  2e-03  6e-07  4e-16
 8: -1.9760e+00 -1.9760e+00  5e-05  1e-08  4e-16
 9: -1.9760e+00 -1.9760e+00  5e-07  1e-10  4e-16
Optimal solution found.
```



As expected, points that are predicted to be anomalous are at the periphery of the data distribution. Yet, the anomaly model appears to be too rigid and fails to account for anomaly scores along the PCA-2 direction. It also fails to account for anomalies between the two clusters. This result can be attributed to the rigidity of the hypersphere model. To address this limitation, we will generate a more detailed, high-dimensional representation of our data, $ \Phi(\boldsymbol{z})$. specifically, a 1000-dimensional representation of the data, where each feature $i=1\dots 1000$ of this representation is calculated as

$$ \Phi_i(\boldsymbol{z}) = \cos(\boldsymbol{\omega}_i^\top \boldsymbol{x} + \tau_i) $$

where $\boldsymbol{\omega}$ and τ are random parameters specified below. This technique is called 'random features'.

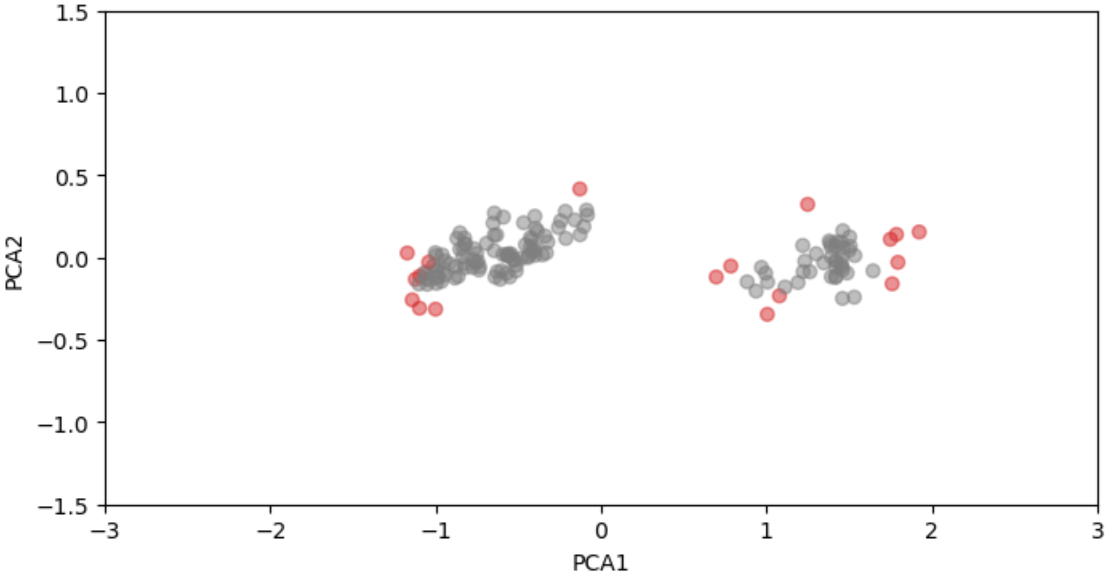
Task: Implement the mapping on this feature representation, specifically, a function that takes a matrix of size $N \times 4$, and that produces as an output the desired matrix of size $N \times 1000$.

```
In [5]: h = 1000
omega = numpy.random.mtrand.RandomState(0).normal(0,1,[4,h])
tau   = numpy.random.mtrand.RandomState(2).uniform(0,2*numpy.pi,[h])

def Phi(X):
    # ------------------------------------
    # TODO: replace by your code
    # ------------------------------------
    import solution
    return solution.phi(X,omega,tau)
    # ------------------------------------
```

The SVDD can now be retrained on this new data representation. The new anomaly predictions can then be visualized.

In [6]:
```
Y = SVDD(0.05).fit_predict(Phi(X))

utils.preparefigure()
plt.scatter(*Z.T,c=Y,alpha=0.5,cmap=lcm(['C7','C3']));
```

```
     pcost       dcost       gap    pres   dres
 0: -4.5730e+02 -4.0862e+02  2e+03  5e+01  2e-15
 1: -4.0680e+02 -3.7177e+02  5e+02  1e+01  2e-15
 2: -3.6309e+02 -3.3813e+02  2e+02  4e+00  1e-15
 3: -3.4141e+02 -3.2118e+02  1e+02  2e+00  8e-16
 4: -3.1167e+02 -3.0264e+02  9e+01  1e+00  5e-16
 5: -2.9440e+02 -2.8999e+02  4e+01  4e-01  4e-16
 6: -2.8894e+02 -2.8574e+02  3e+01  3e-01  5e-16
 7: -2.8301e+02 -2.8073e+02  2e+01  2e-01  4e-16
 8: -2.7918e+02 -2.7776e+02  1e+01  1e-01  3e-16
 9: -2.7628e+02 -2.7592e+02  8e+00  6e-02  3e-16
10: -2.7453e+02 -2.7465e+02  5e+00  3e-02  3e-16
11: -2.7347e+02 -2.7394e+02  4e+00  2e-02  3e-16
12: -2.7199e+02 -2.7270e+02  7e-01  2e-16  4e-16
13: -2.7230e+02 -2.7236e+02  7e-02  4e-16  4e-16
14: -2.7233e+02 -2.7233e+02  7e-04  2e-16  4e-16
15: -2.7233e+02 -2.7233e+02  7e-06  1e-16  4e-16
Optimal solution found.
```



We observe that this nonlinear transformation has made the decision function significantly more flexible, allowing in particular to predict anomalies in a way that better follows the boundaries of the two data clusters.