Exercises for the course
Machine Learning for Data Science
Winter Semester 2022/23

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

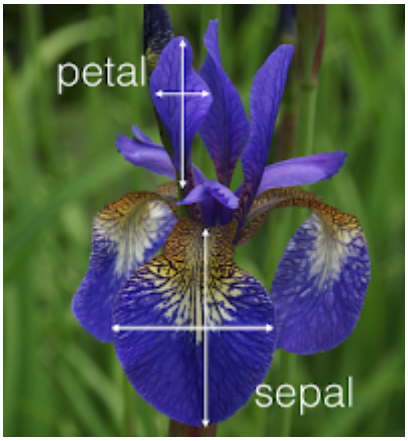
Exercise Sheet 5 (programming part)

In [1]:

```
import numpy
import scipy
import sklearn
import sklearn.datasets
import sklearn.decomposition
import matplotlib
%matplotlib inline
from matplotlib import pyplot as plt
```

Exercise 3 (30 P)

In this exercise we build a model of anomalies for the Iris dataset which we have already considered in Exercise Sheet 4 in the context of clustering.



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. The following cell loads the dataset and performs some standardization.

In [2]:

```
dataset = sklearn.datasets.load_iris()

X = dataset['data']

X = X - X.mean(axis=0)
X = X / X.std()
```

We will identify anomalies in this data using a one-class SVM. The one-class SVM learns a hyperplane that maximally separates the data from the origin. For this approach to be meaningful, the data must be mapped to a feature representation where outliers can be separated from the rest of the data. Such property is observed for certain choices of feature maps such as the 'Gaussian' feature map applied in the cell below:

In [3]:

```
grid = numpy.random.mtrand.RandomState(0).uniform(-3,3,[10000,4])

PhiX = numpy.exp(-scipy.spatial.distance.cdist(X,grid,'sqeuclidean'))

print(PhiX.shape)
```

```
(150, 10000)
```

Note that the feature map is high-dimensional, thereby making it inconvenient to solve the primal form of the one-class SVM. Instead, the dual optimization problem is less affected by such high dimensionality, because the number of parameters to optimize in the dual remains the number of data points. The OC-SVM dual can be stated in matrix form as:

$$\min_{\boldsymbol{\alpha}} \quad \frac{1}{2}\boldsymbol{\alpha}^\top \Phi_X \Phi_X^\top \boldsymbol{\alpha}$$

subject to

$$\mathbf{1}^\top \boldsymbol{\alpha} = 1 \quad \text{and} \quad \begin{pmatrix} -I \\ I \end{pmatrix} \boldsymbol{\alpha} \preceq \begin{pmatrix} \mathbf{0} \\ \mathbf{1}/N\nu \end{pmatrix}$$

where Φ_X is a matrix of size $N \times h$ containing the data mapped to the feature space. To solve this optimization problem, we can use the quadratic solver provided as part of `cvxopt` and the interface of which is shown below:

cvxopt.solvers.qp(*P, q* [, *G, h* [, *A, b* [, *solver* [, *initvals*]]]])

Solves the pair of primal and dual convex quadratic programs

$$\begin{array}{ll} \text{minimize} & (1/2)x^T P x + q^T x \\ \text{subject to} & Gx \preceq h \\ & Ax = b \end{array}$$

(a) Implement the functions `fit_predict` of the class `OCSVM`. 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 $\boldsymbol{w}$ and ρ) from the solution of the dual, and return a vector of size `N` containing the outlier score $y = \rho - \boldsymbol{w}^\top \phi(\boldsymbol{x})$ of each data point (anomalies are then points for which $y > 0$).

In [8]:

```
import cvxopt
import cvxopt.solvers

class OCSVM:

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

    def fit_predict(self,PhiX):
        # Matrices for the optimizer
        P = cvxopt.matrix(PhiX @ PhiX.T)
        q = cvxopt.matrix([0.0,] * PhiX.shape[0])
        G = cvxopt.matrix(numpy.vstack((-numpy.eye(PhiX.shape[0]),
                                        numpy.eye(PhiX.shape[0]))))
        h = cvxopt.matrix([0,] * PhiX.shape[0] + \
                          [1/(self.nu * PhiX.shape[0]),] * PhiX.shape[0])
        A = cvxopt.matrix([1.0,] * PhiX.shape[0], (1, 150))
        b = cvxopt.matrix(1.0)

        # Solution for dual problem
        alpha = numpy.array(cvxopt.solvers.qp(P, q, G, h, A, b)['x'])

        # Retrieving the solution for primal problem
        w = alpha.T @ PhiX

        rho = None
        eps = 0.000001
        for i in range(alpha.shape[0]):
            if alpha[i] > 0 + eps and alpha[i] < 1 / (PhiX.shape[0] * self.nu) - eps:
                rho = (w @ (PhiX[i,:].reshape((-1,1))))[0,0]
                break

        # Computing y
        y = (rho - (w @ PhiX.T)).flatten()
        return y
```

The OC-SVM can now be applied to the Iris dataset. Here, we consider the hyperparameter $\nu = 0.2$ which allows for a moderate number of outliers.

In [5]:

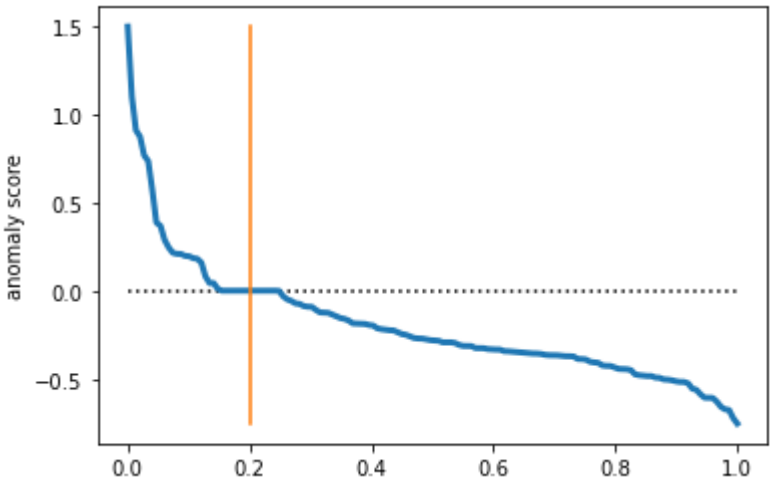
```
nu = 0.2
Y = OCSVM(nu).fit_predict(PhiX)
```

```
     pcost       dcost       gap    pres   dres
 0:  1.0402e+00 -4.6249e+00  5e+02  2e+01  1e-14
 1:  1.3655e+00 -4.1084e+00  3e+01  1e+00  1e-14
 2:  1.8243e+00 -1.6776e+00  5e+00  7e-02  1e-14
 3:  1.8570e+00  1.4515e+00  4e-01  5e-03  1e-14
 4:  1.7898e+00  1.6087e+00  2e-01  2e-03  8e-15
 5:  1.7582e+00  1.7001e+00  6e-02  5e-04  7e-15
 6:  1.7407e+00  1.7303e+00  1e-02  3e-16  8e-15
 7:  1.7372e+00  1.7358e+00  1e-03  2e-16  7e-15
 8:  1.7366e+00  1.7366e+00  3e-05  3e-16  8e-15
 9:  1.7366e+00  1.7366e+00  8e-07  2e-16  8e-15
Optimal solution found.
```

We first look at the distribution of OC-SVM scores.

In [6]:

```
R = numpy.linspace(0,1,len(X))
plt.plot(R,R*0,ls='dotted',color='black')
plt.plot(R,sorted(Y)[::-1],lw=3)
plt.plot([nu]*2,[min(Y),max(Y)])
plt.ylabel('anomaly score')
plt.show()
```



The scores start positive (outliers), then plateau at zero, and finally drop negative (inliers). This plot also allows us to test empirically a theoretical property of ν (orange line), which is that this parameter is an upper-bound to the fraction of points on the wrong side of the boundary (with $\xi_i > 0$) and a lower-bound to the fraction of points that are support vectors (with $\alpha_i > 0$).

Lastly, we make a PCA plot of the Iris data and color-code each data point according to its anomaly score. As expected, points that are predicted to be anomalous are at the periphery of the data distribution, in particular, the two points that were forming their own cluster in Exercise Sheet 4 on Clustering also appear to be outliers, although to a lesser degree than another data point appearing in yellow and that the OC-SVM model predicts to be a stronger outlier.

In [7]:

```
Z = sklearn.decomposition.PCA(2).fit_transform(X)
b= numpy.abs(Y).max()
plt.scatter(*Z.T,c=Y,vmin=-b,vmax=b)
plt.colorbar()
plt.show()
```