Übung zur Vorlesung

Maschinelles Lernen 2

Sommersemester 2013

Abteilung Maschinelles Lernen
Institut für Softwaretechnik und
theoretische Informatik
Fakultät IV, Technische Universität Berlin
Prof. Dr. Klaus-Robert Müller
Email: klaus-robert.mueller@tu-berlin.de

Exercise Sheet 5

Due Tuesday **May 21**, 9am local time, as a digital submission on ISIS

1. **One-class-SVM:Theory (30+10 points)**
(a) *Derive* the dual program for the one-class-SVM (page 26 of the slides). *Show* that it has the form

$$\max_{\alpha} \quad \sum_{i=1}^{N} \alpha_i k(x_i, x_i) - \sum_{i,j=1}^{N} \alpha_i \alpha_j k(x_i, x_j)$$
$$\text{subject to} \quad \sum_{i=1}^{N} \alpha_i = 1 \quad \text{and} \quad 0 \leq \alpha_i \leq C \quad \text{for } i = 1, \ldots, N,$$

where $x_1, \ldots, x_N \in \mathbb{R}^n$ are the training data, and k is some kernel function.

(b) Show that the dual program derived in (a) is a linearly constrained quadratic program, by writing it in the form

$$\begin{aligned} \max_{\alpha} \quad & \alpha^\top A \alpha + b^\top \alpha \\ \text{subject to} \quad & u^\top \alpha \leq v. \\ \forall i : & l_i \leq \alpha_i \leq m_i \end{aligned}$$

with vectors b, u, l, m, matrix A and a scalar v, and where $\leq$ denotes component-wise inequality. That is, explicitly express the b, u, l, m, A and v in terms of (a).

2. **One-class-SVM:Implementation (20+10 points)**
(a) *Write* a MATLAB function `oneclass.m`, implementing one-class-SVM. The program should compute, given a kernel matrix $K \in \mathbb{R}^{N \times N}$ and a regularizing constant $C \geq 0$, the dual solution vector $\alpha \in \mathbb{R}^N$. For solving the quadratic program, use the function `pr_loqo2` by R. Vanderbei.

(b) The learnt SVM given by α needs to be evaluated on test data points $z \in \mathbb{R}^n$. As shown in the lecture, the distance of the test point z to the center of the classifier hypersphere given by α is

$$a(z) = k(z, z) - 2 \sum_{i=1}^{N} \alpha_i k(x_i, z) + \sum_{i,j=1}^{N} \alpha_i \alpha_j k(x_i, x_j).$$

Implement this as a MATLAB function `compute_scores.m`. As in 1(b), it is helpful to use matrix/vector notation.

3. **One-class-SVM vs hackers (30 points)**
Use your implementation of one-class-SVM from exercise 2 to find hacker attacks in a preprocessed data set of network traffic. The data set is provided on ISIS. It is divided in a training and a test partition; as in reality, both partitions contain hacker attacks. *Complete* the MATLAB script `hacker_detection.m` provided on ISIS which: learns the SVM on the training data partition, applies the learnt SVM classifier to the test partition, and returns the exact positions of the detected hacker attacks in the training data set. (if you did not manage to do exercise 2, you may use any implementation of SVM, providing a citation or link where you found it)

The test data set has been generated by K. Rieck from real HTTP connections and hacker attacks. Each connection x has been encoded by 3-grams in a feature vector $\phi(x)$. The data are sum-normalized, i.e., $||\phi(x)||_1 = 1$.

The regularizing constant is a free parameter which can be determined by guessing, cross-validation, or other model selection techniques. *Document* your choice of C; that is, document by which means you arrived at which value of C.

Please ask questions in the ISIS discussion forums for Machine Learning 2:
`https://www.isis.tu-berlin.de/course/view.php?id=6602`