Machine Learning	Prof. Dr. Bernt Schiele
	Dr. Paul Swoboda & Dr. Gerard Pons-Moll
Exercise 6 - 26.11.2018	due: 03.12.2018, 14:15

Exercise 11 - Construction of Kernels

Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a positive definite kernel. Prove that the following functions $k'(x, y)$ are again positive definite kernels:

a. **(2 points)** $k'(x, y) = \sum_{r=0}^{\infty} \alpha_r k(x, y)^r$ for $\alpha_r \geq 0, r \geq 0$.

b. **(2 points)** $k'(x, y) = e^{-\lambda(d_k^2(x,y))}$, where $\lambda > 0$ and $d_k(x, y)$ is the (semi-)metric induced by the kernel k, $d_k^2(x, y) = k(x, x) + k(y, y) - 2k(x, y)$.

c. **(1 point)** $k'(x, y) = e^{-\frac{\|x-y\|^2}{2\sigma^2}}$.

d. **(2 Bonus points)** $\mathcal{X} = \{x \in \mathbb{R}^d \mid \|x\|_2 < 1\}$ and $k'(x, y) = \frac{1}{1-\langle x,y\rangle}$

Exercise 12 - Multiclass schemes for classification of handwritten digits

In this exercise we are doing handwritten digit classification using multi-class SVM with a Gaussian kernel. In order to solve the optimization problem for the SVM, we are using the MATLAB interface to the LIBSVM package (`http://www.csie.ntu.edu.tw/~cjlin/libsvm/`). Download `DataEx6.zip` from the course webpage. Note: You do not need to download anything from the LIBSVM webpage - everything you need is contained in the zip-file.

- Extract the files in libsvm-3.14.zip somewhere in your home directory.
- Start Matlab and use addpath to add the directory where you have extracted LIBSVM to the MATLAB search path (use savepath in order to add it permanently).
- Go to the subfolder matlab and type make in the Matlab prompt. (Pre-built binary files for Windows 64bit already contained in the folder).
- The matlab function `getKernelSVMSolution` provides a nice interface to the LIBSVM package (use `help getKernelSVMSolution` to see how it works).
- **(4 Points)** The problem deals with the classification of handwritten digits (10 classes). You are supposed to use the SVM with the Gaussian kernel:

  $$k(x, y) = e^{-\lambda \|x-y\|^2}.$$

  The training and test data is in `USPSTrain.mat` and `USPSTest.mat`. The 16×16-images of the digits are represented as 256-dimensional column vectors. Write two matlab scripts:

  – one which solves the multi-class problem using **one-versus-all** (save it in `OneVersusAll.m`),

  – one which solves the multi-class problem using **one-versus-one** (save it in `OneVersusOne.m`),

  In both cases use $C = 100$ and $\lambda = \frac{3}{\gamma}$, where γ is the median of all squared distances between **training** points, as parameters for the binary SVM.

Visually inspect the digits which have been misclassified. How do you judge the result? Compare the quality of the classification obtained by the two multi-class schemes. How do the two multiclass schemes compare in terms of runtime?

Save your prediction on the test set in a file `USPSResults.mat` as `PredOneVersusOne` and `PredOneVersusAll` and report the test error for both cases. Also generate for both cases a figure (`ErrorsOneVersusOne.png` and `ErrorsOneVersusAll.png`) containing the misclassified images in the test set.

- **(2 Points)** Suppose the computation of a binary classifier has complexity $O(n^m)$, where n is the number of training points. Suppose we have k classes and the training set contains $\frac{n}{k}$ points of each class. What is the computational complexity of the one-versus-all and one-versus-one scheme? Which multi-class scheme is better in terms of complexity in terms of n? And in terms of k?

Hints:

- Use `dist_euclidean.m` to compute the squared Euclidean distances between two sets of points (warning: your own code may be too slow for this dataset), `getKernelSVMSolution.m` to obtain the dual variables α and the offset b of the SVM and `VecToImage.m` to plot the images of the digits which have been wrongly classified.
- Given a matrix D, with $D_{ij} = \|X_i - Z_j\|^2$, $i = 1, \ldots, n, j = 1, \ldots, m$, of the squared distances between two point sets $\{X_1, \ldots, X_n\}$ and $\{Z_1, \ldots, Z_m\}$ with $X_i \in \mathbb{R}^d$ and $Z_j \in \mathbb{R}^d$, you can compute the kernel matrix `K` in Matlab as: `K=exp(-lambda*D);`.

Submission instructions

- We accept both handwritten and electronic submissions. So you can choose what is more convenient for you. In any case, you should specify full names and immatriculation IDs of all team members. Obviously, programming tasks you can submit only electronically.
- Handwritten submissions should be submitted in the lecture hall of Monday's lecture (before the lecture starts).
- Electronic submissions should be zipped, containing the m-files (`Basis` etc.), your plots (png files) and the matlab data files (.mat) and emailed to the corresponding tutor:

  a. Apratim Bhattacharyya (Wednesday 8-10): abhattac@mpi-inf.mpg.de

  b. Maksym Andriushchenko (Thursday 8-10): s8mmandr@stud.uni-saarland.de

  c. Max Losch (Friday 16-18): mlosch@mpi-inf.mpg.de

  If not all 3 students belong to the same tutorial group, then you should email your submission to **only** one tutor (e.g. to the tutor of the first author of your homework), so please do not put other tutors in copy of the email.

  The email subject must have the following form: "[ML18/19 Exercise] Sheet X", where X is the number of the current exercise sheet. Then please specify in the email full names and immatriculation IDs of all team members. Then please attach all your files as a single zip archive, which consists of your immatriculation IDs, e.g. "2561234_2561235_2561236.zip".

- Reminder: you should submit in groups of 3. Otherwise, we will later on merge the groups smaller than 3 students.