Exercises for
Pattern Recognition
Peter Fischer, Shiyang Hu
Assignment 11, 20/23.01.2015

SOLUTION

General Information:

Exercises (1 SWS):	Tue 12:15 – 13:45 (0.154-115) and Fri 08:15 – 09:45 (0.151-115)
Certificate:	Oral exam at the end of the semester
Contact:	peter.fischer@fau.de, shiyang.hu@fau.de

Kernel SVM and EM Algorithm

Exercise 1 In the last exercise, we created a *Support Vector Machine* (SVM) classifier for a two-class problem that was assumed to be separable within the feature space. For this exercise, we apply a kernel approach and classify the features in higher dimensional space. The class for a test sample using Kernel SVM can be computed using the following model:

$$\hat{y}(\boldsymbol{x}) = \sum_{i=1}^{N} \lambda_i y_i k(\boldsymbol{x}_i, \boldsymbol{x}) + \alpha_0 \tag{1}$$

For training, optimize the Lagrange dual problem of the kernel SVM:

$$\begin{aligned} \max \quad & -\frac{1}{2}\sum_i \sum_j \lambda_i \lambda_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) + \sum_i \lambda_i \\ \text{s.t.} \quad & \lambda_i \geq 0 \ , \ \forall i \\ & \sum_i \lambda_i y_i = 0 \ , \ \forall i \end{aligned} \tag{2}$$

where λ_i is the i-th Lagrangian and K the kernel matrix.

Kernel SVM allows for **non-linear** decision boundaries that are not tractable by linear SVM. In addition, the optimization is independent of the feature dimensionality. This can be a huge advantage e.g. in the case of images.

Basics: Lagrangian optimization
Primal problem: $\min_{\boldsymbol{x}} f(\boldsymbol{x})$ s.t. $g(\boldsymbol{x}) \leq 0$
$\Rightarrow$ the Lagrangian is $L(\boldsymbol{x}, \lambda) = f(\boldsymbol{x}) + \lambda g(\boldsymbol{x})$.
We solve the dual problem: $g(\lambda) = \max_\lambda \inf_{\boldsymbol{x}} L(\boldsymbol{x}, \lambda)$

Kernel SVM
In the dual formulation of the SVM, feature vectors $\boldsymbol{x}$ only appear in inner products during training (see Equation (2)) and testing $\hat{y}(\boldsymbol{x}) = \boldsymbol{\alpha}^T\boldsymbol{x} + \alpha_0 = \sum_{i=1}^{N} \lambda_i y_i k(\boldsymbol{x}_i, \boldsymbol{x}) + \alpha_0$. The 'kernel trick' is to replace the inner products by non-linear kernel functions with similar mathematical properties. The kernels correspond to inner products in high-dimensional space.

(a) Compute the kernel matrix K using a Gaussian radial basis function:

$$k(\boldsymbol{x}, \boldsymbol{x}_i) = e^{-\frac{1}{2}\frac{\|\boldsymbol{x}-\boldsymbol{x}_i\|_2^2}{\sigma^2}}$$

The kernel matrix is a $N \times N$ matrix of the kernel function evaluations between each pair of training samples. You can use a kernel width of $\sigma = 0.1$ for the experiments. When experimenting with this value, you will see that this value acts as a smoother for the input features.

(b) Solve equation (2) using constrained minimization. Possible solvers in Matlab are `fmincon` or `quadprog`.

The objective function and the constraints must be standardized to MATLAB conventions.
fmincon: minimization instead of maximization $\Rightarrow \times -1$
equality constraints: $= 0 \Rightarrow$ OK
inequality constraints: $\leq 0 \Rightarrow \times -1$

Interface: `quadprog(H,f,A,b,Aeq,beq, lb, ub)` minimization of $\frac{1}{2}\boldsymbol{x}^T H \boldsymbol{x} + \boldsymbol{f}^T \boldsymbol{x}$. H is positive definite.
Not necessary: Inequality constraints in matrix notation: $A\boldsymbol{x} \leq \boldsymbol{b}$
Equality constraints in matrix notation: $A_{eq}\boldsymbol{x} = \boldsymbol{b}_{eq}$
Simple bound constraints: $\boldsymbol{lb} \leq \boldsymbol{x} \leq \boldsymbol{ub}$
In the dual problem, the parameters are λ_i!

(c) The dual formulation does not allow to compute the bias parameter α_0 directly. It can be estimated using the average bias of the training samples. Thus, estimate α_0 using the set of support vectors $\mathcal{S}$, which has size $N_{\mathcal{S}}$:

$$\alpha_0 = \frac{1}{N_{\mathcal{S}}} \sum_{n \in \mathcal{S}} \left(y_n - \sum_{m \in \mathcal{S}} \lambda_m y_m K_{nm} \right)$$

(d) Test your kernel SVM in the classification toolbox on the input pattern
`KernelSVMData.mat`
that is provided on the website. This classification problem cannot be solved using the SVM classifier of the previous exercise.

Exercise 2 The EM algorithm was introduced in the lecture and a general example for Gaussian mixture models (GMMs) was given. The goal of this exercise is to implement EM to estimate the parameters of a GMM iteratively. We assume that the number of Gaussians in the mixture model is known.

(a) Provide an initialization for the EM algorithm using k-means clustering. Use `k_means.m` from the homepage.

(b) Implement the E- and M-step of the EM algorithm to estimate GMM parameters.

(c) Implemented a Bayes classifier based on GMM class-conditional densities. The Bayes classifier classifies samples based on maximum posterior probability. Use your implementation of the EM algorithm to estimate a GMM for each class.