Homework 2

Jean-Philippe Vert

Due March 21, 2007

1 2-SVM

The 2-SVM algorithm is a method for supervised binary classification. Given a training set $(x_i, y_i)_{i=1,\ldots,n}$ of training patterns $x_1, \ldots, x_n$ in a space X endowed with a positive definite kernel K, and a set of corresponding labels $y_1, \ldots, y_n \in \{-1, 1\}$, it solves the following problem:

$$\min_{f \in H_K} \left\{ \frac{1}{n} \sum_{i=1}^{n} L(f(x_i), y_i) + \lambda ||f||^2 \right\} ,$$

where $||f||$ is the norm of f in the RKHS H_K of the kernel K, and L is the *square* hinge loss function:

$$L(u, y) = \max(1 - uy, 0)^2 .$$

Write the primal and dual problems associated to the 2-SVM, and compare the result with the SVM studied in the course.

2 Implementations

The goal is to play a bit with classical SVMs. Download the spam data from the course homepage. This is a classical pattern recognition problem. Evaluate the performance of a SVM on this dataset by 5-fold cross-validation, with different kernels and different regularization parameters. For each kernel, plot the average training and test errors as a function of the regularization parameter, and comment the results.

Remarks:

- Use the software of your choice, there are many implementations of SVM that you will find on Google. For example, libsvm can be run from the command line, in python, matlab, R etc... Alternatively you can use environments for machine learning such as Spider on Matlab or PyML on Python, where for example cross-validation procedures are already implemented. In R, the package svmpath computes the SVM solutions for all values of the regularization parameter in a single command, which can be useful here.

- N-fold cross-validation is a procedure to evaluate the generalization performance of machine learning algorithms. It works as follows. Randomly split your dataset into N subsets of similar size. Then take each subset in turn as a test set, train a machine learning algorithm on the remaining $N-1$ subsets, test the model of the test set, and evaluate the performance (e.g., number of misclassification). Repeat the process N times (with each subset taken as a test set), and average the performance over the N runs to estimate the generalization error.