

## 58147 Machine Learning (Spring 2005)

### Exercise 10 (Wednesday 13 April)

1. The *2-norm soft margin SVM* is like the 1-norm soft margin SVM (Optimisation 4.18, page 280) except that the objective function is

$$-\mu + C \sum_{i=1}^m \xi_i^2$$

(the slack variables are squared). Show that now  $\alpha$  is obtained by maximising

$$W(\alpha) = -\frac{1}{4C} \sum_{i=1}^m \alpha_i^2 - \left( \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j k(x_i, x_j) \right)^{1/2}.$$

2. Show that the value  $\alpha = \alpha_t$  chosen by AdaBoost minimises  $G_t(\alpha)$  given on page 303. Show also that the resulting  $\mathbf{q}^{(t+1)}$  has the property

$$\widehat{\text{err}}(h_t, S, \mathbf{d}^{(t+1)}) = \frac{1}{2}.$$

(This means that the “old” hypothesis  $h_t$  has edge 0 with respect to the “new” distribution  $\mathbf{d}^{(t+1)}$  so the weak learner should not choose it immediately again.)

3. Show that the AdaBoost weight update  $\mathbf{d} = \mathbf{d}^{(t+1)}$  is the solution to the relative entropy minimisation

$$\begin{array}{ll} \text{minimise} & \sum_{i=1}^m d_i \ln \frac{d_i}{d_i^{(t)}} \\ \text{subject to} & d_i \geq 0, \sum_{i=1}^m d_i = 1 \\ & \sum_{i=1}^m d_i y_i h_t(x_i) = 0. \end{array}$$

(See previous problem for the interpretation of the last constraint.)

4. Download the OSU SVM Matlab Toolbox from

[http://www.ece.osu.edu/~maj/osu\\_svm/](http://www.ece.osu.edu/~maj/osu_svm/) .

Download also a simple artificial data set from

<http://www.cs.helsinki.fi/Jyrki.Kivinen/opetus/koppi/k05/balls.mat> .

The data consist of a 2-dimensional circle of negative examples surrounded by positive ones, with a small amount of noise (plot it to see for yourself). Use the data to train an SVM with a polynomial kernel (function `PolySVC`) and visualise the result (function `SVMPLOT2`). The instructions at the beginning of each `.m` file are pretty clear. Try different degree polynomials and see whether you get a nice regular decision boundary. You may also need to play a little with the `C` parameter.

5. We now return to the cars data set from Exercise 5. Split the data into a training set (70% of the data) and a test set (30%). Use the training data to learn an SVM with Gaussian kernel (function `RbfSVC`) and check the training and test errors (function `SVMTTest`). Repeat for a range of kernel parameters `Gamma` and see which values give overfitting and which give bad training errors.