Problem 1: Support Vector Machines

[55 points]

(1) [20 points]

(a) [5 points] By definition, given the distribution $D$, $h^*$ is defined as:

$$ h^* = \arg\min_{h: X \to \{-1, +1\}} error_D(h). \quad (1) $$

$K^*$ is clearly positive definite symmetric since $K^*(x, x')$ is defined as the dot product (in dimension one) of the features vectors $h^*(x)$ and $h^*(x')$.

(b) [15 points] The general expression of the solution is

$$ h(x) = \text{sgn}\left(\sum_{i=1}^{m} \alpha_i K^*(x, x_i) + b\right). \quad (2) $$

Here, it is easy to see both in the separable and non-separable case that the solution is simply:

$$ h(x) = \text{sgn}(K^*(x, x_+)), \quad (3) $$

where $x_+$ is such that $h^*(x_+) = +1$. One support vector is enough. The solution can be rewritten as

$$ h(x) = h^*(x). \quad (4) $$

The generalization error of the solution is thus that of the Bayes classifier (it is optimal). The data is separable iff the Bayes error is zero.

(c) [5 points] A kernel of this type is always positive definite symmetric since $K(x, x')$ is defined as a dot product of the feature vectors $h(x)$ and $h(x')$. 

Figure 1: Error

(2) [35 points] [Thanks to Chien-I Liao for writing the solution for this section.]

(a) [10 points] We need to train once on $m$ points. The test result and the number support vectors $N_{SV}$ for $m$ points are then known. Then, we just need to train and test $N_{SV}$ SVMs on $m-1$ points since the leave-one-out error when excluding a non-support vector point is identical to the original error.

(b) [25 points]
- First rescale all the data:
  $\$ mv positive.dat old-positive.dat
  $\$ ./svmscale old-positive.dat > positive.dat
  $\$ mv negative.dat old-negative.dat
  $\$ ./svmscale old-negative.dat > negative.dat

- Then write a program to split the data into 10 folds. A sample C++ program could be found at http://cs.nyu.edu/~cil217/TA/split.cpp.
• Compile the code:
  
  ```bash
g++ split.cpp -o split
  ```

• Then write a script to repeatedly run svm-train and svm-predict. A sample bash script could be found at http://cs.nyu.edu/~cil217/TA/train_test.sh

• Run the script:
  ```bash
chmod 755 train_test.sh
./train_test.sh
  ```

Figures 1 and 2 show the result with default parameter setting.

**Problem 2: Kernel Methods**

[45 points]

(1) [20 points] $X^* - I$ is a regular language and can be represented by a finite automaton. $K$ can thus be defined by

\[ K(x, y) = [[T \circ T^{-1}]](x, y), \]

(5)
Figure 3: Weighted transducer $T$. $e$ represents the empty string, and $r = \rho$. $X^* - I$ stands for a finite automaton accepting $X^* - I$.

where $T$ is the weighted transducer shown in Figure 3. Thus, $K$ is a rational kernel and in view of the theorem of Lecture 5, it is positive definite symmetric.

(2) [10 points] Let $M_{X^* - I}$ be the minimal automaton representing $X^* - I$. The transducer $T$ of Figure 3 can be constructed using $M_{X^* - I}$. Then, $|T| = |M_{X^* - I}| + 8$. Using composition of weighted transducers, the running time complexity of the computation of the algorithm is:

$$O(|x| |y| |T \circ T^{-1}|) = O(|x| |y| |T|^2) = O(|x| |y| |M_{X^* - I}|^2).$$

(6)

(3) [15 points] The set of strings $Y$ over the alphabet $X$ of length less than $n$ form a regular language since they can be described by:

$$Y = \bigcup_{i=0}^{n-1} X^i.$$  \hspace{1cm} (7)

Thus, $Y_1 = Y \cap (X^* - I)$ and $Y_2 = (X^* - I) - Y_1$ are also regular languages. It suffices to replace in the transducer $T$ of Figure 3 the transition labeled with $X^* - I : X^* - I / \rho$ with two transitions:

- $Y_1 : Y_1 / \rho_1$, and
- $Y_2 : Y_2 / \rho_2$,

with the same origin and destination states and with $Y_1$ and $Y_2$ denoting finite automata representing them. The kernel is thus still rational and PDS since it is of the form $T' \circ T'^{-1}$.