EE6550 Machine Learning, Spring 2017

Homework Assignment #2

Please submit your solutions of the word problems in class (during the first recess time) on March 27th Monday and your programs, including source code, report (in pdf format) and user manual (in pdf format), to iLMS by 23:59 on March 29th Wednesday. Late submission will not be accepted unless the instructor gives a pre-approval. You are encouraged to consult or collaborate with other students while solving the problems, but you will have to turn in your own solutions and programs with your own words and work. If you find any resources in the internet to assist you, you should understand but not copy them. Copying will not be tolerated.

Part I: Word problem set.

1. In this problem, you will derive the update rule of the sequential minimal optimization (SMO) algorithm which is used to implement an efficient SVM-learning algorithm for binary classification. SMO reduces a (potentially) large quadratic programming (QP) optimization problem into a series of small optimizations involving only two Lagrange multipliers. Please see the article *A Supplement of SMO Algorithmfor HW#2 Programming Problem*, which will be referred to as *the article*.

   (a) (0.5%) Assume that we want to solve the Lagrangian dual problem in Eq. (17) of *the article* only over two Lagrangian multipliers λ_i and λ_j, by fixing the values of other Lagrangian multipliers $\lambda_k, k \neq i, j$ to their most recently updated values. Please show that the dual problem in Eq. (17) of *the article* can be reduced to the dual problem in Eq. (18) of *the article*.

   (b) (0.5%) By substituting $\lambda_i = \gamma_{ij}^* - s_{ij}\lambda_j$, the dual problem in Eq. (18) becomes the dual problem in Eq. (19). Please show that without the inequality constraints in Eq. (19), the λ_j^{new} in Eq. (20) maximizes the object function Ψ_2 when $\eta_{ij} > 0$.

   (c) (0.5%) Please show that the $\lambda_j^{new,clip}$ in Eq. (26) solves the optimization problem with inequality constraints in Eq. (19) when $\eta_{ij} > 0$.

   (d) (0.5%) When $\eta_{ij} = 0$, please show that the dual problem in Eq. (19) becomes the dual problem in Eq. (27) and the $\lambda_j^{new,clip}$ in Eq. (28) solves this problem.

2. Let $\Phi : \mathscr{I} \to \mathbb{R}^N$ be a feature mapping such that the dimension N of the feature space is very large. Define a kernel $K : \mathscr{I} \times \mathscr{I} \to \mathbb{R}$ by

$$K(\omega, \omega') \triangleq \sum_{i=1}^{N} D(i)[\Phi(\omega)]_i \, [\Phi(\omega')]_i,$$

   where $D(i)$ is the weighting for the product of the i-th components $[\Phi(\omega)]_i$ and $[\Phi(\omega')]_i$ of the two feature vectors $\Phi(\omega)$ and $\Phi(\omega')$ with $D(i) \geq 0 \; \forall \; i$ and $\sum_{i=1}^{N} D(i) = 1$. We assume that $|[\Phi(\omega)]_i| \leq r$ for all $\omega \in \mathscr{I}$ and $i \in [1, N]$.

   (a) (0.5%) Please show that K is a PDS kernel over the input space $\mathscr{I}$.

(b) (0.5%) To reduce the computational complexity, an approximation of $K(\omega, \omega')$ can be computed based on the random selection of a set $I = \{i_1, i_2, \ldots, i_n\}$ of indices of size n according to the distribution $D = \{D(1), D(2), \ldots, D(N)\}$ on the index set $[1, N]$, that is,

$$\hat{K}(\omega, \omega') \triangleq \frac{1}{n} \sum_{k=1}^{n} [\Phi(\omega)]_{i_k} \, [\Phi(\omega')]_{i_k}.$$

Fix ω and ω' in $\mathscr{I}$. Prove that

$$\Pr_{I \sim D^n} \left(\left| K(\omega, \omega') - \hat{K}(\omega, \omega') \right| > \epsilon \right) < 2e^{-\frac{n\epsilon^2}{2r^4}}.$$

(Hint: use McDiarmid's inequality.)

(c) (0.5%) Let $S = (\omega_1, \omega_2, \ldots, \omega_m)$ be a sample of size m drawn i.i.d. from the input space $\mathscr{I}$ according to an unknown distribution. Let $\mathbf{K} = [K(\omega_i, \omega_j)]$ and $\hat{\mathbf{K}} = [\hat{K}(\omega_i, \omega_j)]$ be the kernel matrices associated to K and $\hat{K}$ respectively, and the sample S. Show that for any $\epsilon, \delta > 0$, for $n \geq \frac{2r^4}{\epsilon^2} \ln \frac{m(m+1)}{\delta}$, with probability at least $1 - \delta$,

$$\left| K(\omega_i, \omega_j) - \hat{K}(\omega_i, \omega_j) \right| \leq \epsilon \; \forall \; i, j \in [1, m].$$

Part II: (11.5%) Programming problem: Implementation of an SVM-learning algorithm $\mathbb{A}$ for binary classification by using the sequential minimal optimization (SMO) algorithm. Please refer to *the article*.

Input:

1. A data file which contains a labeled training sample S. This labeled training sample is used to train the SVM-learning algorithm which will return a hypothesis h_S^{SVM} after n-fold cross-validation.
2. A data file which contains a labeled testing sample $\tilde{S}$. This labeled testing sample is used to evaluate the performance of the returned hypothesis h_S^{SVM} from the SVM-learning algorithm based on the labeled training sample S.
3. Choice of n-fold cross-validation, where $n = 5$ or $n = 10$. The free parameter is C. As discussed in Lecture 1, we use n-fold cross-validation to determine the best value of the free parameter C.
   - Randomly partition a given training sample S of m labeled items into n subsamples or folds.
   - $((\omega_{i1}, c(\omega_{i1})), ..., (\omega_{im_i}, c(\omega_{im_i})))$: the ith fold of size m_i, $1 \leq i \leq n$.
     - Usually $m_i = \frac{m}{n}$ for all i.
   - For any $i \in [1, n]$, the SVM-learning algorithm is trained on all but the ith fold to generate a hypothesis h_i, and the performance of h_i is tested on the ith fold.
   - $\hat{R}_{CV}(C)$: the cross-validation error.

$$\hat{R}_{CV}(C) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{m_i} \sum_{j=1}^{m_i} 1_{h_i(\omega_{ij}) \neq c(\omega_{ij})} = \frac{1}{m} \sum_{i=1}^{n} \sum_{j=1}^{m_i} 1_{h_i(\omega_{ij}) \neq c(\omega_{ij})}.$$

- Choose a parameter value C^* which minimizes the cross-validation error $\hat{R}_{CV}(C)$.
- Train the SVM-learning algorithm with the best parameter setting C^* over the full training sample S of size m. The resulted hypothesis will be the returned hypothesis h_S^{SVM} from the SVM-learning algorithm.

Output:

1. The optimal value of the free parameter C up to the 2nd decimal point.
2. The hypothesis h_S^{SVM} returned by the SVM-learning algorithm,

   $$h_S^{SVM}(\mathbf{x}) = \text{sgn}(\mathbf{w}^{SVM} \cdot \mathbf{x} + b^{SVM}),$$

   where $\mathbf{w}^{SVM}$ is the weight vector returned by the SMO algorithm and

   $$b^{SVM} = c(\mathbf{x}_j) - \mathbf{w}^{SVM} \cdot \mathbf{x_j} = -F_j^{SVM}$$

   for any support vector $\mathbf{x}_j$ with $0 < \lambda_j^{SVM} < C$. Thus we have

   $$h_S^{SVM}(\mathbf{x}) = \text{sgn}\left(c(\mathbf{x}_j) + \mathbf{w}^{SVM} \cdot (\mathbf{x} - \mathbf{x}_j)\right)$$

   for any support vector $\mathbf{x}_j$ with $0 < \lambda_j^{SVM} < C$.
3. Performance of the returned hypothesis h_S^{SVM} on the labeled testing sample $\tilde{S}$.

What to submit? You should submit the following items:

1. The source code of your SVM-learning algorithm based on SMO.
   Please indicate which programming language you have used to write the programs in the title of the submission entry with one of the following formats:
   - HW2_yourstudentid_yourname_matlab (The environment you use should be compatible with the version licensed to the NTHU Computer Center.)
   - HW2_yourstudentid_yourname_python36
   - HW2_yourstudentid_yourname_cpp14
   - HW2_yourstudentid_yourname_c11

   This will facilitate the distribution of homeworks to graders for grading. Please have your code compilable/interpretable by a standard compiler/interpreter environment: Matlab (NTHU CC), Python3.x, C++14, and C11.
2. A report consisting of at least:
   (a) a description, at least with a table, of the n-fold cross-validation results to determine the optimal value C^* of the free parameter C up to the 2nd decimal point with $n = 5$;
   (b) a description, at least with a table, of the n-fold cross-validation results to determine the optimal value C^* of the free parameter C up to the 2nd decimal point with $n = 10$;
   (c) the hypothesis h_S^{SVM} returned by the SVM-learning algorithm with $n = 5$;
   (d) the hypothesis h_S^{SVM} returned by the SVM-learning algorithm with $n = 5$;
   (e) the performance of the returned hypothesis h_S^{SVM} on the labeled testing sample $\tilde{S}$ with $n = 5$;

(f) the performance of the returned hypothesis h_S^{SVM} on the labeled testing sample $\tilde{S}$ with $n = 10$.

Please submit your report in pdf format (please do not submit word files).

3. A user manual which should include instructions of

   (a) how to compile the source code with a standard compiler/interpreter;

   (b) how to run, i.e., execute the SVM-learning algorithm, including the required formats of input parameters or data files;

   (c) what results are reported.

   These instructions should support the test scenarios in the grading session. Please submit your manual in pdf format (please do not submit word files).

Test scenario in the grading session: the grader will test your algorithm with your source code by

1. inputting a training data file, which contains a labeled training sample S. This labeled training sample is used to train the SVM-learning algorithm which will return a hypothesis h_S^{SVM} after n-fold cross-validation.
2. inputting a testing data file, which contains a labeled testing sample $\tilde{S}$. This labeled testing sample is used to evaluate the performance of the returned hypothesis h_S^{SVM} from the SVM-learning algorithm based on the labeled training sample S.
3. inputting a positive integer n to perform n-fold cross-validation to determine the optimal value C^* of the free parameter C to minimize the cross-validation error $\hat{R}_{CV}(C)$.
4. checking the obtained optimal value C^* of the free parameter C with a prepared program up to the 2nd decimal point.
5. checking the hypothesis h_S^{SVM} returned by the SVM-learning algorithm with a prepared program.
6. checking the performance of the returned hypothesis h_S^{SVM} on the labeled testing sample $\tilde{S}$ with a prepared program.