CS492(E) Homework 2

Logistic Regression and Support Vector Machine

(Due date : 2017-11-17 11:59 pm)

Dataset.

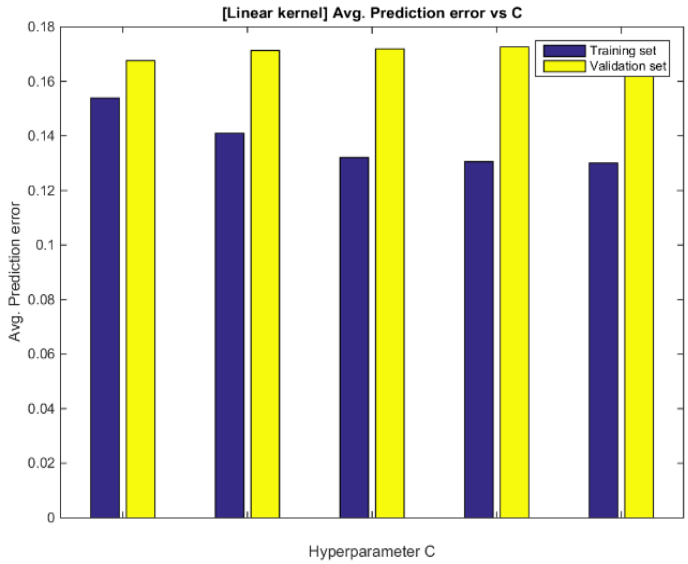
In this programming homework, you use a LIBSVM dataset which are pre-processed data originally from UCI data

- **Adult** dataset (binary classification dataset with 123 features). You can download 'a2a' and 'a2a.t' datasets from https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html#a2a
  - **a2a** is the training dataset with 2,265 data points
  - **a2a.t** is the test dataset with 30,296 data points
- Cross-validation (https://en.wikipedia.org/wiki/Cross-validation_(statistics))
  Since there is no additional validation dataset for hyper-parameter tuning, prepare 5 training and validating pairs for 5-fold cross validation. Randomly partition the training data 'a2a' into 5 (almost) equal sized subsets. A single subset is retained as the validation and the remaining 4 are used for training, at a time. Keep these training and validation pairs for problems 2 and 3.
- Reporting the prediction error on the test dataset.
  (prediction error = 1 - (the number of correct prediction) / (the number of prediction)
  Since the test dataset 'a2a.t' contains more than 30,000 data points, we will use the first 1,000 data points when compute the prediction accuracy on the test dataset **for all problems below**.

Problem.

1. *Logistic Regression* (20 points)
   Using any existing implementation of Logistic regression is not allowed.
   a. Prove that the loss function (or the objective function) of logistic regression $\mathrm{J}(\beta) = \sum_{i=1}^{N}\left(-y_i \log\left(p_i(\beta)\right) - (1-y_i)\log\left(1-p_i(\beta)\right)\right)$ is convex.
   where $p_i(\beta) = \frac{1}{1+e^{-\beta^T x_i}}$.
   (You can find $\mathrm{J}(\beta)$ at page 41 in Logistic Regression lecture note.)
   b. Train your logistic regression by using **a2a** dataset only. Draw a plot showing objective function value vs. iterations of gradient descent. Report the prediction error on the test dataset
   c. Repeat (b) with 3 different step size (learning rate).

2. *Support Vector Machine with the linear kernel* (40 points)
   linear kernel : $\mathrm{K}(\mathrm{x}, \mathrm{y}) = x^T y$
   Using any existing implementation of SVM is not allowed. However, using a quadratic solver (QP) is fine. For example, you can use the built-in function **quadprog** in MATLAB. In Python, you can use a free Python package, **CVXOPT** (http://cvxopt.org).
   a. Train your SVM and tune the hyper-parameter *C* by using 5-fold cross validation. (You can find C at 27 page in SVM lecture note).
   b. Draw the plot on the average of prediction errors on the five training sets and validation sets vs. *C* (for each 5 different *C*).



< Figure 1. plot example >

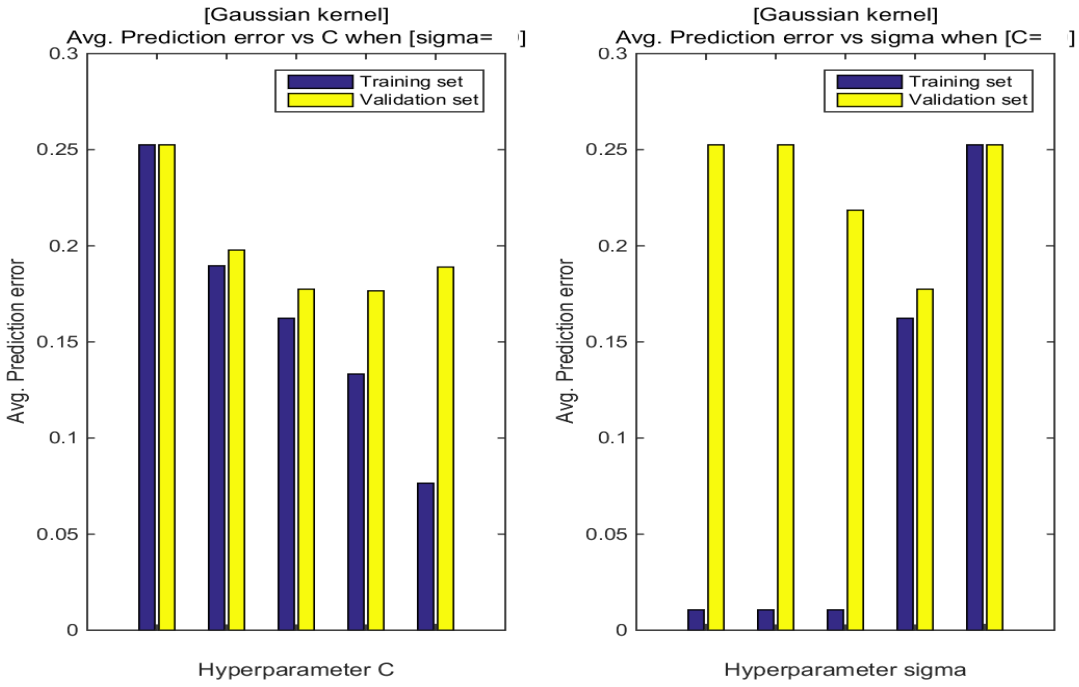
   c. For the best performing C, report the test prediction error.

3. *Support Vector Machine with the Gaussian kernel* (30 points)
   In this problem, we implement SVM that supports the kernel trick.

   Gaussian kernel : $\mathrm{K}(\mathrm{x},\mathrm{y}) = \exp\left(-\frac{||x-y||^2}{2\sigma^2}\right)$

   a. Train your SVM with the Gaussian kernel and tune the hyper-parameters C and σ by using 5-fold cross validation.

   b. Using your best performing C and σ that you've found in (a), draw the plot on the average of prediction errors on the five training sets and validation sets vs. σ (for each 5 different σ when C is fixed as the best performing C) and the plot on the average of prediction errors on the five training sets and validation sets vs. C (for each 5 different C when σ is fixed as the best performing σ).



< Figure 2. plot example >

   c. For the best performing C and σ, report test prediction error.

4. Provide the report with the details (in Korean or in English; either is fine). Try to include discussions and your impressions on your work. (10 points)