CSCI 5525: Machine Learning (Fall'13)
Homework 3, Due 11/22/13

1. **(25 points)** The Ionosphere dataset has 2 classes (good radar returns, bad radar returns) with 351 samples, each having 34 continuous features. Train and evaluate the following classifiers using 10-fold cross-validation:

   (a) (10 points) Bagged 2-layer decision trees with binary splits using Information Gain with the following number of base classifiers: $B = [5, 10, 15, \dots, 50]$.

   (b) (15 points) Random forest of 100 2-layer decision trees with binary splits using Information Gain with the size of the random feature set $M = [1, 2, \dots, 34]$.

   You will have to submit (i) **summary of methods and results**, and (ii) **code** for each algorithm:

   (i) **Summary of methods and results:** Briefly describe the algorithms in (a) and (b) along with necessary equations, e.g., for splitting on a feature.

   For (a), provide a table of the training and test set error rates for each fold and number of base classifiers. Also provide the training and test set average error rates and standard deviation across all folds for each number of base classifiers. Finally, include a graph of the training and test set average error rates as the number of base classifiers increase.

   For (b), provide a table of the training and test set error rates for each fold and value of M. Also provide the training and test set average error rates and standard deviation across all folds for each value of M. Finally, include a graph of the training and test set average error rates as the value of M increases.

   The graphs *must* have a title, labeled axis, and labeled curves. Finally, thoroughly discuss the results of each method.

   (ii) **Code:** For part (a), you will have to submit code for `myBagging2(filename,B)` (main file). This main file has **input**: filename for the dataset and a vector B for the number of base classifiers, e.g., $B = [5, 10, 15, \dots, 50]$, and **output**: print to the terminal (stdout) the training and test set error rates and number of base classifiers for each fold of the 10-fold cross-validation, along with the average error rate and standard deviation for training and test sets. The function *must* take the inputs in this order and display the output via the terminal.

   For part (b), you will have to submit code for `myRForest2(filename,M)` (main file). This main file has **input**: filename for the dataset and a vector M of the size of the random feature set, e.g., $M = [1, 2, \dots, 34]$, and **output**: print to the terminal (stdout) the training and test set error rates and feature set size for each fold of the 10-fold cross-validation, along with the average error rate and standard deviation for training and test sets. The function *must* take the inputs in this order and display the output via the terminal.

The filename will correspond to a **plain text file** for a dataset, with each line corresponding to a data point: the first entry will be the label and the rest of the entries will be feature values of the data point.

For each part, you can submit additional files/functions (as needed) which will be used by the main file. Put comments in your code so that one can follow the key parts and steps in your code.

2. **(30 points)** This problem considers two different approaches to measuring complexity of a function class: VC dimensions and Rademacher complexity.

   (a) (10 points) Define the VC dimension $VC(\mathcal{F})$ of a function class $\mathcal{F}$. Show that the VC dimension of hyper-plane classifiers in $\mathbb{R}^d$, $\mathcal{F} = \{f : f(x) = \text{sign}(\mathbf{w}^T x + w_0), \mathbf{w} \in \mathbb{R}^d, w_0 \in \mathbb{R}\}$ is $VC(\mathcal{F}) = d + 1$.

   (b) (10 points) Define the Rademacher complexity $R_n(\mathcal{F})$ of a function class $\mathcal{F}$ for n-samples. Show that for any two sets of independent samples x, x' of size n

$$E\left[\sup_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^{n} \rho_i(f(x_i') - f(x_i))\right] \leq 2R_n(\mathcal{F}) ,$$

   where the expectation is over $x = \{x_1, \ldots, x_n\}, x' = \{x_1', \ldots, x_n'\}$, and the independent Rademacher variables $\{\rho_1, \ldots, \rho_n\}$.

   (c) (10 points) For any class of functions $\mathcal{F}$, consider the class $\text{co}_2(\mathcal{F})$ defined as

$$\text{co}_2(\mathcal{F}) = \{f : f(x) = \alpha f_1(x) + (1-\alpha)f_2(x), f_1, f_2 \in \mathcal{F}, \alpha \in [0,1]\} \ .$$

   Assuming $\mathcal{F}$ has a finite set of functions, show that $R_n(\text{co}_2(\mathcal{F})) = R_n(\mathcal{F})$.

3. **(20 points)** This problem considers developing an ADMM (Alternating Direction Method of Multipliers) for sparse 2-class logistic regression. Let $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ be the dataset under consideration, where $\mathbf{x}_i \in \mathbb{R}^d, y_i \in \{0, 1\}$. The objective function to be minimized is given by:

$$f(\mathbf{w}) = h(\mathbf{w}) + \lambda g(\mathbf{w}) = -\sum_{i=1}^{N} \left\{y_i \mathbf{w}^T \mathbf{x}_i - \log(1 + \exp(\mathbf{w}^T \mathbf{x}_i))\right\} + \lambda \|\mathbf{w}\|_1 \ , \qquad (1)$$

   where $\lambda > 0$ is a positive constant and $\|\mathbf{w}\|_1 = \sum_{j=1}^{d} |w_j|$. Further, $h(w)$ refers to the first term and $g(w)$ refers to the second term.

   (a) (10 points) Assuming that the dataset of size N has been grouped into m mini-batches of n points each, i.e., $N = mn$, clearly outline the key steps of the ADMM algorithm for solving the problem.

   (b) (5 points) Which steps can be executed in parallel? Which steps require access to more than one mini-batch? Briefly explain your answers.

   (c) (5 points) Is the resulting algorithm a double-loop algorithm, i.e., an inner loop is required to solve one/more of the key steps of the ADMM algorithm? Briefly explain your answers.

4. **(25 points)** We consider solving the sparse logistic regression problem in (1) using a different algorithm, which we will refer to as COGD (Composite Objective Gradient Descent). Following the notation in (1), COGD does the following update in each iteration:

$$\mathbf{w}_{t+1} = \underset{\mathbf{w}}{\operatorname{argmin}} \left\{ \langle \nabla h(\mathbf{w}_t), \mathbf{w} \rangle + \lambda \|\mathbf{w}\|_1 + \frac{1}{2\eta_t} \|\mathbf{w} - \mathbf{w}_t\|^2 \right\} , \tag{2}$$

where η_t is equivalent to the (adaptive) learning rate.

   (a) (7 points) When $\lambda = 0$, show that the above update is exactly the same as the gradient descent algorithm for logistic regression with learning rate η_t.

   (b) (10 points) Derive the closed form update for $\mathbf{w}_{t+1}$ for any $\lambda > 0$, i.e., express $\mathbf{w}_{t+1}$ as a function of the vectors $(\mathbf{w}_t, \nabla h(\mathbf{w}_t))$ and the scalers (λ, η_t).

   (c) (4 points) What is the per-iteration computational complexity of the closed form update of $\mathbf{w}_{t+1}$ in (b) above in terms of (d, N), where d is the dimensionality of $\mathbf{w}$, and N in the number of data points?

   (d) (4 points) Can you design a stochastic gradient descent algorithm for the problem, i.e., one whose per-iteration complexity has no dependence on N, the number of data points? Briefly justify your answer.

Additional instructions: All outside material used *must* be cited. Code can only be written in C/C++, Java, Matlab, or Python, no other programming languages will be accepted. All programs must be able to be executed from the terminal command prompt. Please specify instructions on how to run your program in the README file.

Evaluation notes: Code will be tested on the provided dataset and possibly other similar (2 class, multivariate, continuous) datasets. Correctness of code, error rates, and standard deviations will be evaluated as well as the discussion of methods used and results.

Instructions

Follow the rules strictly. If we cannot run your functions, you get 0 points.

Things to submit

1. hw3.pdf: A document which contains the solutions to Problems 1, 2, 3, and 4, including the summary of methods and results.

2. `myBagging2` and `myRForest2`: Code for Problem 1.

3. README.txt: README file that contains your name, student ID, email, instructions on how to compile (if necessary) and run your code, any assumptions you are making, and any other necessary details.

4. Any other files, except the data, which are necessary for your program.