1 Boosting

In this problem, you will develop an alternative way of forward stagewise boosting. The overall goal is to derive an algorithm for choosing the best weak learner h_t at each step such that it best approximates the gradient of the loss function with respect to the current prediction of labels. In particular, consider a binary classification task of predicting labels $y_i \in \{+1, -1\}$ for instances $\mathbf{x}_i \in \mathbb{R}^d$, for $i = 1, \ldots, n$. And also a set of weak learners denoted by $\mathcal{H} = \{h_j, j = 1, \ldots, M\}$ that we have access to. In this framework, we first choose a loss function $L(y_i, \hat{y}_i)$ in terms of current prediction $\hat{y}_i$ and the true labels y_i. Here, we use the logistic loss, namely $L(y_i, \hat{y}_i) = \log(1 + \exp(-y_i\hat{y}_i))$. We also consider the gradient g_i of the cost function $L(y_i, \hat{y}_i)$ with respect to the current predictions $\hat{y}_i$ on each instance, i.e. $g_i = \frac{\partial L(y_i, \hat{y}_i)}{\partial \hat{y}_i}$. We take the following steps for boosting:

(a) **Gradient Calculation** In this step, please calculate the gradients $g_i = \frac{\partial L(y_i, \hat{y}_i)}{\partial \hat{y}_i}$.

(b) **Weak Learner Selection** We then choose the next learner to be the one that can best predict these gradients, i.e. we choose

$$h^* = \arg\min_{h \in \mathcal{H}} \left(\min_{\gamma \in \mathbb{R}} \sum_{i=1}^{n} (-g_i - \gamma h(\mathbf{x}_i))^2 \right)$$

Please show that the optimal value of the step size γ can be computed in the closed form in this step, thus the selection rule for h^* can be derived independent of γ.

(c) **Step Size Selection** We then select the step size α^* that minimizes the loss:

$$\alpha^* = \arg\min_{\alpha \in \mathbb{R}} \sum_{i=1}^{n} L(y_i, \hat{y}_i + \alpha h^*(\mathbf{x}_i)).$$

For the logistic loss function, there is no closed form solution for α^*. Instead, we use one step of Newton's method to obtain the step size. That is we start from $\alpha = 0$ and carry out one step of Newton's method to obtain the step size α^*. Finally, we perform the following updating step:

$$\hat{y}_i \leftarrow \hat{y}_i + \alpha^* h^*(\mathbf{x}_i).$$

You are asked to compute the expression of step size analytically in terms of y_i, $\hat{y}_i$, and h^*.

2 SVM

This problem focuses on ϵ-Support Vector Regression (ϵ-SVR). Suppose we are given training data $\{(x_1, y_1), \ldots, (x_l, y_l)\} \in \mathcal{X} \times \mathbb{R}$ where $\mathcal{X}$ denotes the space of the input patterns. In ϵ-SV regression, our goal is to find a functions $f(x)$ that has at most ϵ deviation from the actually obtained targets y_i for all the training data, and at the same time, is as *flat* as possible. In other words, we do not care about error as long as they are less than ϵ but will not accept any deviation larger than this. This may be important if you want to be sure not to lose more than ϵ money when dealing with exchange rates.

(a) Let's start with the linear case where:

$$f(x) = w^T x + b \quad \text{with} \quad w \in \mathcal{X}, b \in \mathbb{R}$$

Flatness in this case means that one seeks small $||w||_2^2$. write down the primal optimization formulation such that it maximized the flatness given the most ϵ deviation for each point.

(b) However there is not always a solution to the optimization that is derived in (a). That is due to the fact that it may not be possible to find a line that guarantees ϵ deviation. In this case we need to bring in a (hinge) loss function to penalize the case when the deviation is greater than ϵ. Assume that the loss function punishes linearly when you have deviation above ϵ (hinge). Modify the above optimization formulation to incorporate the loss function with weight factor $C > 0$. C determines the trade off between the flatness of f and the amount up to which deviations larger than ϵ are tolerated. Plot the hinge loss function you used in this question.

(c) The derived optimization problem is easier to be solve in its dual form and also easier to incorporate nonlinearity into it. Construct the Lagrange function of previous primal that you derived, and then derive the dual form. Show that the weight w only depends on the support vectors.

(d) In the next step we make the SV regression nonlinear. This, for instance, could be achieved by simply preprocessing the training patterns x_i by a map $\phi : \mathcal{X} \to \mathcal{F}$ into some feature space $\mathcal{F}$. Incorporate this nonlinear mapping into the derived dual form in terms of kernel $k(x, x') := \phi^{\mathrm{T}}(x)\phi(x')$ and show that how prediction is derived using the kernel function.

3 Programming

In this part, you will experiment with SVMs on a real-world dataset. You will implement linear SVM (i.e., SVM using the original features, namely the nonlinear mapping is the identity mapping) using Matlab. Also, you will use a widely used SVM toolbox called LIBSVM to experiment with kernel SVMs.

Dataset: We have provided the *Phishing Websites Data Set* from UCI's machine learning data repository (`https://archive.ics.uci.edu/ml/datasets/Phishing+Websites`. The dataset is for a binary classification problem to detect phishing websites. The dimensionality of the input feature is 30, and the training and test sets contain 2,000 and 2,000 samples, respectively (they are provided in Blackboard as `phishing-train.mat` and `phishing-test.mat` which can be directly loaded into Matlab).

3.1 Data preprocessing

All the features in the datasets are categorical. You need to preprocess the training and test data to make features with multiple values to features taking values only zero or one. If a feature f_i have value $\{-1, 0, 1\}$, we create three new features $f_{i,-1}, f_{i,0}$ and $f_{i,1}$. Only one of them can have value 1 and $f_{i,x} = 1$ if and only if $f_i = x$. For example, we transform the original feature 1 into $[0, 0, 1]$. In the given dataset, the features $2, 7, 8, 14, 15, 16, 26, 29$ (index starting from 1) take three different values $\{-1, 0, 1\}$. You need to transform each above feature into three 0/1 features.

3.2 Implement linear SVM

Please write Matlab functions `trainsvm` as `trainsvm.m` and `testsvm` as `testsvm.m`. The input of `trainsvm` contain training feature vectors and labels, as well as the tradeoff parameter C. The output of `trainsvm` contain the SVM parameters (weight vector and bias). In your implementation, you need to solve SVM in its primal form

$$\begin{aligned}
&\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_i \xi_i \\
&\text{s.t.} \quad y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \geq 1 - \xi_i, \forall i \\
&\qquad\;\; \xi_i \geq 0, \forall i
\end{aligned}$$

Please use `quadprog` function in Matlab to solve the above quadratic problem. For `testsvm`, the input contain testing feature vectors and labels, as well as SVM parameters; the output contains the test accuracy.

3.3 Cross validation for linear SVM

Use 3-fold cross validation to select the optimal C for your implementation of linear SVM.

(a) Report the 3-fold cross-valiation accuracy (averaged accuracy over each validation set) and average training time (averaged over each training subset) on different C taken from $\{4^{-6}, 4^{-5}, \cdots, 4, 4^2\}$. How does the value of C affect the cross validation accuracy and average training time? Explain your observation.

(b) Which C do you choose based on the cross validation results?

(c) You need to write a script `own_linear.m` using your `trainsvm`, `testsvm` and the optimal C selected from cross validation. Your script should train a linear SVM with selected C and output the accuracy on the test set.

3.4 Use linear SVM in LIBSVM

LIBSVM is widely used toolbox for SVM and has Matlab interface. Download LIBSVM from `https://www.csie.ntu.edu.tw/~cjlin/libsvm/` and install it according to the README file (make sure to use Matlab interface provided in LIBSVM toolbox). For each C from $\{4^{-6}, 4^{-5}, \cdots, 4, 4^2\}$, apply 3-fold cross validation (use `-v` option in LIBSVM) and report the cross validation accuracy and average training time.

(a) Is the cross validation accuracy the same as that in 3.3? Note that LIBSVM solves linear SVM in dual form while your implementation does it in primal form.

(b) How faster is LIBSVM compared to your implementation, in terms of training?

3.5 Use kernel SVM in LIBSVM

LIBSVM supports a number of kernel types. Here you need to experiment with the polynomial kernel and RBF (Radial Basis Function) kernel.

(a) **Polynomial kernel**. Please tune C and `degree` in the kernel. For each combination of (C, `degree`), where $C \in \{4^{-3}, 4^{-2}, \cdots, 4^6, 4^7\}$ and `degree` $\in \{1, 2, 3\}$, report the 3-fold cross validation accuracy and average training time.

(b) **RBF kernel**. Please tune C and `gamma` in the kernel. For each combination of (C, `gamma`), where $C \in \{4^{-3}, 4^{-2}, \cdots, 4^6, 4^7\}$ and `gamma` $\in \{4^{-7}, 4^{-6}, \cdots, 4^{-2}, 4^{-1}\}$, report the 3-fold cross validation accuracy and average training time.

Based on the cross validation results of Polynomial and RBF kernel, which kernel type and kernel parameters will you choose? You need to write a script `libsvm.m` using the best kernel and the parameters selected from cross-validation. Your script should train a SVM using libsvm with selected kernel and parameters and output the accuracy on the test set.

4 Submission Instructions

Submission Instructions: You need to provide the followings:

- Provide your answers to problems 1-3 in PDF file, named as `CSCI567_hw4_fall15.pdf`. You need to submit the homework in both hard copy (at Locker #19 at PHE, with a box labeled as CSCI567-homework by 6pm of the deadline date) and electronic version as pdf file on Blackboard. If you choose handwriting instead of typing all the answers, you will get 40% points deducted.
- Submit ALL the code and report via Blackboard by 6 pm of the deadline date. The only acceptable language is MATLAB. For your program, you MUST include the main function called `CSCI567_hw4_fall15.m` in the root of your folder. After running this main file, your program should be able to generate all of the results needed for this programming assignment, either as plots or console outputs. Moreover, you SHOULD include scripts `own_linear.m` and `libsvm.m` in the root of your fold. After running these two scrips, your program should be able to generate the results described above. You can have multiple files (i.e your sub-functions), however, the requirement is that once we unzip your folder and execute your main file or the two scripts, your program should execute correctly. Please double-check your program before submitting. You should only submit one `.zip` file. No other formats are allowed except `.zip` file. Also, please name it as `[lastname]_[firstname]_hw4_fall15.zip`.

Collaboration: You may collaborate. However, collaboration has to be limited to discussion only and you need to write your own solution and submit separately. You also need to list with whom you have discussed.