CS6501: Advanced Machine Learning	**2017**
Problem Set 1	
Handed Out: Feb 10, 2017	*Due: Feb 17, 2017*

- Feel free to talk to other members of the class in doing the homework. You should, however, write down your solution yourself. Please try to keep the solution brief and clear.
- Please use Piazza first if you have questions about the homework. Also feel free to come to office hours.
- Please, no handwritten solutions. You will submit your solution manuscript as a single pdf file. Please use LaTeXto typeset your solutions.
- The homework is due at 11:59 PM on the due date. We will be using Collab for collecting the homework assignments. Please submit your solution manuscript as a pdf file and your code as a zip file. Please do NOT hand in a hard copy of your write-up. Contact the TAs if you are having technical difficulties in submitting the assignment.

1 Binary Classification

A *linear program* can be stated as follows:

Definition 1 *Let A be an $m \times n$ real-valued matrix, $\vec{b} \in \mathbb{R}^m$, and $\vec{c} \in \mathbb{R}^n$. Find a $\vec{t} \in \mathbb{R}^n$, that minimizes the linear function*

$$\begin{aligned} & z(\vec{t}) = \vec{c}^{\,T}\vec{t} \\ \textit{subject to} \quad & A\vec{t} \geq \vec{b} \end{aligned}$$

In the linear programming terminology, $\vec{c}$ is often referred to as the *cost vector* and $z(\vec{t})$ is referred to as the *objective function*.[1] We can use this framework to define the problem of learning a linear discriminant function.[2]

[1]Note that you don't need to really know how to solve a linear program since you can use Matlab or Scipy to obtain the solution (although you may wish to brush up on Linear Algebra).

[2]This discussion closely parallels the linear programming representation found in *Pattern Classification*, by Duda, Hart, and Stork.

The Learning Problem:[3] Let $\vec{x_1}, \vec{x_2}, \ldots, \vec{x_m}$ represent m samples, where each sample $\vec{x_i} \in \mathbb{R}^n$ is an n-dimensional vector, and $\vec{y} \in \{-1, 1\}^m$ is an $m \times 1$ vector representing the respective labels of each of the m samples. Let $\vec{w} \in \mathbb{R}^n$ be an $n \times 1$ vector representing the weights of the linear discriminant function, and θ be the threshold value.

We *predict* $\vec{x_i}$ to be a *positive* example if $\vec{w}^T\vec{x_i} + \theta \geq 0$. On the other hand, we *predict* $\vec{x_i}$ to be a *negative* example if $\vec{w}^T\vec{x_i} + \theta < 0$.

We hope that the learned linear function can separate the data set. That is, for the true labels we hope

$$y_i = \begin{cases} 1 & \text{if } \vec{w}^T\vec{x_i} + \theta \geq 0 \\ -1 & \text{if } \vec{w}^T\vec{x_i} + \theta < 0. \end{cases} \tag{1}$$

In order to find a good linear separator, we propose the following linear program:

$$\min_{\vec{w},\theta,\delta} \quad \delta \tag{2}$$

$$\text{subject to} \quad y_i(\vec{w}^T\vec{x_i} + \theta) \geq 1 - \delta, \qquad \forall(\vec{x_i}, y_i) \in D \tag{3}$$

$$\delta \geq 0 \tag{4}$$

where D is the data set of all training examples.

a. *[10 points]* A data set $D = \{(\vec{x_i}, y_i)\}_{i=1}^m$ that satisfies condition (1) above is called *linearly separable*. **Show that the data set D is linearly separable if and only if there exists a hyperplane $(\vec{w'}, \theta')$ that satisfies condition** (3) **with** $\delta = 0$ (Need a hint?[4]). Conclude that D is linearly separable iff the optimal solution to the linear program [(2) to (4)] attains $\delta = 0$. **What can we say about the linear separability of the data set if there exists a hyperplane that satisfies condition** (3) **with** $\delta > 0$**?**

b. *[5 points]* **Show that there is a trivial optimal solution for the following linear program:**

$$\min_{\vec{w},\theta,\delta} \quad \delta$$

$$\text{subject to} \quad y_i(\vec{w}^T\vec{x_i} + \theta) \geq -\delta, \qquad \forall(x_i, y_i) \in D$$

$$\delta \geq 0$$

Show the optimal solution and use it to (very briefly) explain why we chose to formulate the linear program as [(2) **to** (4)] **instead.**

[3] Note that the notation used in the Learning Problem is **unrelated** to the one used in the Linear Program definition. You may want to figure out the correspondence.

[4] **Hint:** Assume that D is linearly separable. D is a finite set, so it has a positive example that is *closest* to the hyperplane among all positive examples. Similarly, there is a negative example that is *closest* to the hyperplane among negative examples. Consider their distances and use them to show that condition (3) holds. Then show the other direction.

c. *[10 points]* Let $\vec{x_1} \in \mathbb{R}^n$, $\vec{x_1}^T = \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}$ and $y_1 = 1$. Let $\vec{x_2} \in \mathbb{R}^n$, $\vec{x_2}^T = \begin{bmatrix} -1 & -1 & \dots & -1 \end{bmatrix}$ and $y_2 = -1$. The data set D is defined as

$$D = \{(\vec{x_1}, y_1), (\vec{x_2}, y_2)\}.$$

Consider the formulation in [(2) to (4)] applied to D. **Show the set of all possible optimal solutions (solve this problem by hand)**.

2 Multiclass Classification

In this problem set, we will derive, implement, and test an SGD method for a multi-class SVM classification model, and then we will compare it with the one-vs-all approach.

Given a data set $D = \{x_i, y_i\}_{i=1}^N$ with K classes, the multi-class SVM can be written as the following unconstrained optimization problem:

$$\min_{w=\{w_k\}} \quad \frac{1}{2}\sum_{k=1}^{K} w_k^T w_k + C\sum_{i=1}^{N}\left(\max_{k=1\dots K}\left(\Delta(y_i,k) + w_k^T x_i\right) - w_{y_i}^T x_i\right), \tag{5}$$

where w_k is the weight vector for class k, (x_i, y_i) is the i^{th} instance and

$$\Delta(y,k) = \begin{cases} 1 & \text{if } y \neq k, \\ 0 & \text{otherwise.} \end{cases}$$

To derive the SGD algorithm, we first need to rewrite the objective function in Eq. (5) into the form of $\sum_i g_i(w)$:

$$\min_{w=\{w_k\}} \quad \sum_{i=1}^{N}\left(\frac{1}{2N}\sum_{k=1}^{K} w_k^T w_k + C\left(\max_{k=1\dots K}\left(\Delta(y_i,k) + w_k^T x_i\right) - w_{y_i}^T x_i\right)\right),$$

so that

$$g_i(w) = \left(\frac{1}{2N}\sum_{k=1}^{K} w_k^T w_k + C\left(\max_{k=1\dots K}\left(\Delta(y_i,k) + w_k^T x_i\right) - w_{y_i}^T x_i\right)\right).$$

Now let's derive the sub-gradient of $g_i(w)$. Consider $\partial g_i(w)/\partial w_k$, the partial sub-gradient of the first term of $g_i(w)$ is w_k/N. Regarding the second term, we have to consider several cases. Let

$$\tilde{y} = \arg\max_k \left(\Delta(y_i,k) + w_k^T x_i\right)$$

. If $\tilde{y} = k$ and $y_i = k$, the partial sub-gradient of the second term is $C(x - x) = 0$. Therefore, $\partial g_i(w)/\partial w_k = w_k/N$ when $\tilde{y} = k$ and $y_i = k$.

How about other cases?

a. [15 points] Please complete the following formulation

$$\frac{\partial g_i(w)}{\partial w_k} = \begin{cases} w_k & \text{if } \tilde{y} = k \text{ and } y_i = k \\ \_\_\_\_\_ & \text{if } \tilde{y} \neq k \text{ and } y_i = k \\ \_\_\_\_\_ & \text{if } \tilde{y} = k \text{ and } y_i \neq k \\ \_\_\_\_\_ & \text{if } \tilde{y} \neq k \text{ and } y_i \neq k \end{cases} \tag{6}$$

b. [15 points] The SGD algorithm runs as follows:

for epoch = 0 ... T **do**
  for $(\vec{x_i}, y_i)$ in D **do**
    for k= 1 ... K **do**
      $w_k \leftarrow w_k - \eta \; \partial g_i(w_k)/\partial w_k$
    end for
  end for
end for

η is the step size. The step size is a user specified parameter. Here, we can set it with a default value 0.01.

Plug Eq. (6) in, we have:

for epoch = 0 ... T **do**
  for $(\vec{x_i}, y_i)$ in D **do**
    $\tilde{y} = \arg\max_k \left(\Delta(y_i, k) + w_k^T x_i\right)$
    for k= 1 ... K **do**

      $w_k \leftarrow w_k - \eta$_____

    end for
    if $\tilde{y} \neq y_i$ **then**

      $w_{y_i} \leftarrow w_{y_i} - \eta$_____

      $w_{\tilde{y}} \leftarrow w_{\tilde{y}} - \eta$_____

    end if
  end for
end for

Please, complete the above algorithm.

c. [20points] Let's implement a baseline one-vs-all multiclass algorithm and conduct experiments on MNIST dataset [5]. We provide a basic code framework at `https:`

[5] http://yann.lecun.com/exdb/mnist/

//github.com/uvanlp/HW1 to facilitate the implementation. However, you're welcomed to write your own code. If you want to use the code we provided, implement the one-vs-all algorithm in the main function of one_vs_all_with_data_reader.py. Then, conduct cross-validation to find the best C for all underlying binary classifier (assume the same C is used for all the binary classifiers). Note: You can use the LinearSVC class at Scikit-Learn, `http://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html`, to implement the underlying classifiers.

Report the best C and the final accuracy.

d. [25points] Implement the SGD algorithm for multi-class SVM. We provide 'multi-class_svm.py' as the code framework. However, you have to implement the update rules. Conduct cross-validation to find the best C and report the result on the test set.[6]

e. [0-10 points] We will give bonus points for additional experiments.

[6] To test if your algorithm implementation is correct, you can check if the objective function value of the SGD algorithm matches with the score from liblinear https://www.csie.ntu.edu.tw/ cjlin/liblinear. Use the following command line to train a multi-class SVM model "./train -s 4 -B -1 -e 0.01 -C 1 train.data" with C=1. There is a function in the sample code allows you to print data in liblinear format. Check if the objective function values of your implementation is the same as that of liblinear on a small subset (e.g., run on only the first 100 samples).