CS/CNS/EE 253 – Advanced Topics in Machine Learning
Problem Set 1

Handed out:	15 Jan 2010
Due:	01 Feb 2010

1 A Support Vector Machine without Support Vectors

In this question, you'll be implementing an online support vector machine algorithm to classify handwritten digits, and comparing the performance of the online algorithm with an offline SVM implementation. The data set and MATLAB template code is available in the `HW1.zip` file available on the course webpage.

Dataset

The dataset, `MNISTdata.mat` consists of 11500 training examples , `Xtrain`, from the MNIST handwritten digits dataset (3s and 5s), and 1900 test cases, `Xtest`. The data is stored as a matrix with each row being a vector of pixel intensities. The images can be visualized in MATLAB using

```
colormap(gray); imagesc(reshape(Xtrain(1,:),[28,28])')
```

for example. The labels for the training and test examples (+1 for 3, and -1 for 5) are stored in the vectors `Ytrain` and `Ytest`.

Online SVM

We will use a slightly modified version of the Online SVM. The pseudocode for the algorithm is given below. The algorithm sets the margin to 1, as well as the learning rate. You should normalize the data to have unit norm.

Template code for the online SVM function has been provided in `classification.m` and `onlineSVM.m`.

1. Implement an online SVM by filling out the missing code in `onlineSVM.m`.

2. Plot the classification error on the test set as the dataset size grows i.e. show the number of data points observed, vs. test accuracy. Also plot the running time vs. accuracy. To do this, modify the code in `classification.m`.
   You can look at the weight vector learned during training as an image. Do the weights make sense?

Algorithm 1 Online linear support vector machine

```
1: function onlineSVM(X, Y, $\lambda$)
2: $w \leftarrow 0$
3: for $t = 1$ to $N$ do
4:     $\alpha_t = 1/(\lambda t)$
5:     if $y_t w^T x_t < 1$ then
6:         $w \leftarrow (1 - \alpha_t \lambda) w + \alpha_t y_t x_t$ {Apply Gradient if Margin Violated}
7:     end if
8:     if $w^T w > \frac{1}{\lambda}$ then
9:         $w = \frac{w}{\sqrt{w^T w}\sqrt{\lambda}}$ {Project to ball of size $\sqrt{\lambda}$}
10:    end if
11: end for
12: return $w$ {Return the average of all $w$'s computed during process}
```

Offline SVM

You'll use a standard SVM implementation, *SVMlight*, available at `http://svmlight.joachims.org/`, for the offline classification task. The software and instructions for installation have been provided. We have also included a MATLAB wrapper for *SVMlight*, `svmlight.m`, that can be called using

```
Y = svmlight(Xtrain, Ytrain, Xtest, '-t 0 -c 0.5 ')
```

This code outputs the margins, `Y`, for the test examples `Xtest`, using the weights learned from the training examples. The argument used in the above example specifies the use of a Linear kernel, and the margin/training error trade-off parameter c, having the value of 0.5. This will take about 5 minutes to run on your computer. To classify the test examples, you need to threshold the margin to obtain the labels of +1 or -1 (setting the threshold to 0 is reasonable).

1. Plot the accuracy obtained from using different amounts of training data for the offline SVM, by randomly subsampling the training data and predicting on the test cases. How does the online SVM fare against the offline version when you only use 10, 20, 50, 100, 200, 500, 1000, 2000, 5000, 10000 datapoints? Modify the code in `comparison.m` to do this.

2. Plot the run-time vs. accuracy for the offline SVM for different amounts of training data. How does this compare against the online SVM? You might want to plot the runtime on a log-scale to make the comparison clearer.

Please submit all the functions, including any other scripts or functions you may have written, and specify the parameters you used to create the plots.

2 The ϵ_n-Greedy Policy for the K-armed bandit

In the K-armed bandit problem, we picture a slot machine with K arms. We play a game for T rounds, where at each round n we pull one of the arms, e.g., arm j, and obtain payoff X_{jn}. We assume that the payoffs $X_{11}, \ldots, X_{KT}$ are bounded in $[0, 1]$, and independent across arms and rounds. Each arm j has a potentially different expected payoff $\mathbb{E}[X_{j1}] = \cdots = \mathbb{E}[X_{jT}] = \mu_j$ that is unknown to us. Thus, we need to develop a policy for pulling the arms in order to trade off exploration (learning about payoffs μ_j for the different arms) and exploitation (pull the arm that maximizes the expected revenue μ_j). We will use $\mu^* = \max_j \mu_j$ to refer to the maximum expected payoff across all arms.

Consider the ϵ_n-Greedy policy for the K-armed bandit problem. The algorithm repeats the following procedure at each time n: Play the arm j^* with highest current average reward $j^* = \arg\max_j \bar{X}_j$ with probability $(1 - \epsilon_n)$, where $\bar{X}_j$ is the average reward observed for arm j in the first n rounds (i.e., the cumulative reward for arm j divided by how often the arm was pulled). With probability ϵ_n play an arm chosen uniformly at random. Use $\epsilon_n = \min\{1, \frac{cK}{d^2 n}\}$ where $c > 0$ and $0 < d < \min_{i:\mu_i < \mu^*} \Delta_i$. Hereby, $\Delta_i = \mu^* - \mu_i$ is how much less payoff (in expectation) arm i provides compared to the best arm.

Define $P(I_n = j)$ as the probability that sub-optimal machine j was chosen at time n. We will prove that

$$P(I_n = j) \leq \frac{\epsilon_n}{K} + o(\frac{1}{n}). \tag{1}$$

Hint: You can prove each of the following subproblems independently of the others.

1. Show that:

$$P(I_n = j) \leq \frac{\epsilon_n}{K} + P(\bar{X}_j \geq \bar{X}^*). \tag{2}$$

2. Now we need to bound the second term on the right hand side of eq. 2. We will use tail bounds to do this. First, show that:

$$\begin{aligned} P(\bar{X}_j \geq \bar{X}^*) \leq & P(\bar{X}_j \geq \mu_j + \frac{\Delta_j}{2}) \\ & + P(\bar{X}^* \leq \mu^* - \frac{\Delta_j}{2}) \end{aligned} \tag{3}$$

   Hint: Prove the following:

$$P(A \geq B) \leq P(A \geq a) + P(B \leq a) \tag{4}$$

   for random variables A and B defined on $[0, \infty)$ and any $a > 0$.

3. We can now handle each of the two terms on the right hand side of eq. 3 using the same analysis. Focusing on the first term, we have:

$$P(\bar{X}_j \geq \mu_j + \frac{\Delta_j}{2}) = \sum_{t=1}^{n} P(T_j = t | \bar{X}_{jt} \geq \mu_j + \frac{\Delta_j}{2}) P(\bar{X}_{jt} \geq \mu_j + \frac{\Delta_j}{2}), \tag{5}$$

where T_j indicates the number of times arm j has been pulled and $\bar{X}_{jt} = \frac{1}{t}\sum_{i=1}^{t} X_j^{(i)}$ is the average reward from machine j after it's been pulled t times, i.e., $X_j^{(i)}$ refers to the i-th pull of arm j. Use the Chernoff-Hoeffding bound to show that

$$P(\bar{X}_j \geq \mu_j + \frac{\Delta_j}{2}) \leq \sum_{t=1}^{n} P(T_j = t | \bar{X}_{jt} \geq \mu_j + \frac{\Delta_j}{2}) e^{\frac{-\Delta_j^2 t}{2}}. \tag{6}$$

The exponential terms in the sum will decay rapidly for sufficiently large values of t, but the first exponential terms are potentially large. Our strategy is then to split the sum into two parts and bound each sum seperately. Define the boundary value $\lfloor x_0 \rfloor$ (we will determine x_0 in the next section). Show that

$$\sum_{t=1}^{n} P(T_j = t | \bar{X}_{jt} \geq \mu_j + \frac{\Delta_j}{2}) e^{\frac{-\Delta_j^2 t}{2}} \leq \sum_{t=1}^{\lfloor x_0 \rfloor} P(T_j = t | \bar{X}_{jt} \geq \mu_j + \frac{\Delta_j}{2}) + \frac{2}{\Delta_j^2} e^{\frac{-\Delta_j^2 \lfloor x_0 \rfloor}{2}}. \tag{7}$$

4. We will now bound $P(T_j = t | \bar{X}_{jt} \geq \mu_j + \frac{\Delta_j}{2})$ with $P(T_j^R \leq t)$, which is the probability that arm j has been pulled up to t times at random (the conditioning was dropped since random pulls are independent of the values of the averages). The sum can then be bounded with

$$\sum_{t=1}^{\lfloor x_0 \rfloor} P(T_j = t | \bar{X}_{jt} \geq \mu_j + \frac{\Delta_j}{2}) \leq \sum_{t=1}^{\lfloor x_0 \rfloor} P(T_j^R \leq t) \leq x_0 P(T_j^R \leq x_0). \tag{8}$$

Now, we need a bound for $P(T_j^R \leq x_0)$. Because T_j^R is the sum of binary random variables, we can use the Bernstein inequality to obtain a bound. Consider random variables $A_1, \ldots, A_n$ with range $[0,1]$ and $\sum_{i=1}^{n} \mathrm{Var}[A_i | A_{i-1}, \ldots, A_1] = \sigma^2$. The Bernstein inequality is

$$P(\sum_{i=1}^{n} A_i \leq \sum_{i=1}^{n} E[A_i] - a) \leq e^{-\frac{a^2/2}{\sigma^2 + a/2}} \tag{9}$$

for any $a \geq 0$.

Show that choosing $x_0 = \frac{1}{2}E[T_j^R]$ yields the following exponential bound:

$$P(T_j^R \leq x_0) \leq e^{-\frac{x_0}{5}}. \tag{10}$$

5. Putting everything together we have

$$P(I_n = j) \leq \frac{\epsilon_n}{K} + 2x_0 e^{-\frac{x_0}{5}} + \frac{4}{\Delta_j^2} e^{\frac{-\Delta_j^2 \lfloor x_0 \rfloor}{2}}. \tag{11}$$

The last thing to do is to find a lower bound for x_0 in terms of n. Recall that $x_0 = \frac{1}{2K}\sum_{t=1}^{n} \epsilon_t$. From the definition of the algorithm, $\epsilon_t = 1$ when $t < \frac{cK}{d^2}$ and $\epsilon_t = \frac{cK}{d^2 t}$ otherwise. Prove the following lower bound on x_0:

$$x_0 \geq \frac{c}{d^2} \ln \frac{(n-1)d^2 e^{1/2}}{cK}. \tag{12}$$

6. Substitute the lower bound for x_0 into eq. 11 and show that this implies

$$P(I_n = j) \leq \frac{\epsilon_n}{K} + o(\frac{1}{n}). \tag{13}$$

   Comment on any conditions on the parameters c and d that are necessary for this result to hold.

7. Why does this result (i.e., $P(I_n = j) \leq \frac{\epsilon_n}{K} + o(\frac{1}{n})$) imply that the regret when playing the ϵ_n-Greedy algorithm for T rounds grows as $O(K \log T)$?