10-601 Machine Learning: Homework 4

Due 5:30 p.m. Thursday, March 3, 2016
TAs: Han Zhao and Tianshu Ren

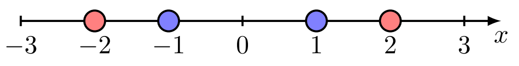
Instructions

- **Late homework policy**: Homework is worth full credit if submitted before the due date, half credit during the next 48 hours, and zero credit after that. You must turn in at least $n-1$ of the n homeworks to pass the class, even if for zero credit.
- **Collaboration policy**: Homeworks must be done individually, except where otherwise noted in the assignments. "Individually" means each student must hand in their own answers, and each student must write and use their own code in the programming parts of the assignment. It is acceptable for students to collaborate in figuring out answers and to help each other solve the problems, though you must in the end write up your own solutions individually, and you must list the names of students you discussed this with. We will be assuming that, as participants in a graduate course, you will be taking the responsibility to make sure you personally understand the solution to any work arising from such collaboration.
- **Submission**: You must submit your solutions on time **BOTH** electronically by submitting to autolab AND by dropping off a hardcopy in the bin outside Gates 8221 by 5:30 p.m. Thursday, March 3, 2016. We recommend that you use LaTeX, but we will accept scanned solutions as well. On the Homework 4 autolab page, you can click on the "download handout" link to download the submission template, which is a tar archive containing a blank placeholder pdf for your written solutions, and an Octave .m file for each programming question. Replace each of these files with your solutions for the corresponding problem, create a new tar archive of the top-level directory, and submit your archived solutions online by clicking the "Submit File" button. Please remember to **include your Andrew ID** on **both** your electronic and hardcopy homework solution.

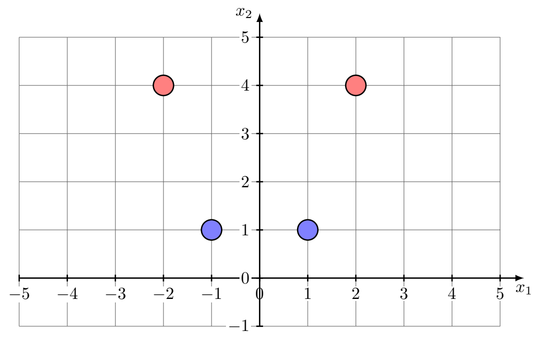
1 Support Vector Machines [30 pts] (Han)

1.1 Feature Map [8 pts]

Consider a binary classification problem in one-dimensional space where the sample contains four data points $S = \{(1, -1), (-1, -1), (2, 1), (-2, 1)\}$ as shown in Fig. 1a.



(a) Red points represent instances from class +1 and blue points represent instances from class -1.



(b) Data instances after the feature map transformation.

1. [3 pts] Define $H_t = [t, \infty)$. Consider a class of linear separators $\mathcal{H} = \{H_t : t \in \mathbb{R}\}$, i.e., for $\forall H_t \in \mathcal{H}$, $H_t(x) = 1$ if $x \geq t$ otherwise -1. Is there any linear separator $H_t \in \mathcal{H}$ that achieves 0 classification error on this example? If yes, show one of the linear separators that achieves 0 classification error on this example. If not, briefly explain why there cannot be such linear separator.

2. [3 pts] Now consider a feature map $\phi : \mathbb{R} \to \mathbb{R}^2$ where $\phi(x) = (x, x^2)$. Apply the feature map to all the instances in sample S to generate a transformed sample $S' = \{(\phi(x), y) : (x, y) \in S\}$, shown in Fig. 1b. Let $\mathcal{H}' = \{ax_1 + bx_2 + c \geq 0 : a^2 + b^2 \neq 0\}$ be a collection of half-spaces in $\mathbb{R}^2$. More specifically, $H_{a,b,c}((x_1, x_2)) = 1$ if $ax_1 + bx_2 + c \geq 0$ otherwise -1. Is there any half-space $H' \in \mathcal{H}'$ that achieves 0 classification error on the transformed sample S'? If yes, give the equation of the max-margin linear separator and compute the corresponding margin. For this question, you can give the equation directly by inspection of Fig. 1b.

3. [2pts] What is the kernel corresponding to the feature map $\phi(\cdot)$ in the last question, i.e., give the kernel function $K(x, z) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$.

1.2 Constructing Kernels [12 pts]

In this question you will be asked to construct new kernels from existing kernels. Suppose $K_1(\mathbf{x}, \mathbf{z}) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ and $K_2(\mathbf{x}, \mathbf{z}) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ are both kernels, show the following functions are also kernels:

1. [4 pts] $K(\mathbf{x}, \mathbf{z}) = c_1 K_1(\mathbf{x}, \mathbf{z}) + c_2 K_2(\mathbf{x}, \mathbf{z})$ with $c_1, c_2 \geq 0$.

2. [4 pts] $K(\mathbf{x}, \mathbf{z}) = K_1(\mathbf{x}, \mathbf{z}) \cdot K_2(\mathbf{x}, \mathbf{z})$.

3. [4 pts] Let $q(t) = \sum_{i=0}^{p} c_i t^i$ be a polynomial function with nonnegative coefficients, i.e., $c_i \geq 0, \forall i$. Show that $K(\mathbf{x}, \mathbf{z}) = q(K_1(\mathbf{x}, \mathbf{z}))$ is a kernel. (**Hint**: You can use the conclusions from last two questions to prove this one.)

4. (Bonus, [2 pts]) $K(\mathbf{x}, \mathbf{z}) = \exp(K_1(\mathbf{x}, \mathbf{z}))$. (**Hint**: You can use the conclusion from the last question to prove this one.)

5. (Bonus, [2 pts]) Let A be a positive semidefinite matrix and define $K(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T A \mathbf{z}$.

1.3 Support Vectors [10 pts]

In question 1 we explicitly construct the feature map and find the corresponding kernel to help classify the instances using linear separator in the feature space. However in most cases it is hard to manually construct the desired feature map, and the dimensionality of the feature space can be very high, even infinity, which makes explicit computation in the feature space infeasible in practice. In this question we will develop the dual of the primal optimization problem to avoid working in the feature space explicitly. Suppose we have a sample set $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ of labeled examples in $\mathbb{R}^d$ with label set $\{1, -1\}$. Let $\phi : \mathbb{R}^d \to \mathbb{R}^D$ be a feature map that transform each input example to a feature vector in $\mathbb{R}^D$. Recall from the lecture notes that the primal optimization of SVM is given by

$$\begin{aligned} \underset{\mathbf{w},\xi_i}{\text{minimize}} \quad & \frac{1}{2}||\mathbf{w}||_2^2 + C\sum_{i=1}^n \xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^T\phi(\mathbf{x}_i)) \geq 1 - \xi_i \quad \forall i = 1, \ldots, n \\ & \xi_i \geq 0 \quad \forall i = 1, \ldots, n \end{aligned}$$

which is equivalent to the following dual optimization

$$\begin{aligned} \underset{\alpha_i}{\text{minimize}} \quad & \sum_{i=1}^n \alpha_i - \frac{1}{2}\sum_{i=1}^n\sum_{j=1}^n \alpha_i\alpha_j y_i y_j \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j) \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C \quad \forall i = 1, \ldots, n \\ & \sum_{i=1}^n \alpha_i y_i = 0 \end{aligned}$$

Recall from the lecture notes $\xi_1, \ldots, \xi_n$ are called slack variables. The optimal slack variables have intuitive geometric interpretation as shown in Fig. 2. Basically, when $\xi_i = 0$, the corresponding feature vector $\phi(\mathbf{x}_i)$ is correctly classified and it will either lie on the margin of the separator or on the correct side of the margin. Feature vector with $0 < \xi_i \leq 1$ lies within the margin but is still be correctly classified. When $\xi_i > 1$, the corresponding feature vector is misclassified. Support vectors correspond to the instances with $\xi_i > 0$ or instances that lie on the margin. The optimal vector $\mathbf{w}$ can be represented in terms of $\alpha_i, i = 1, \ldots, n$ as $\mathbf{w} = \sum_{i=1}^n \alpha_i y_i \phi(\mathbf{x}_i)$.

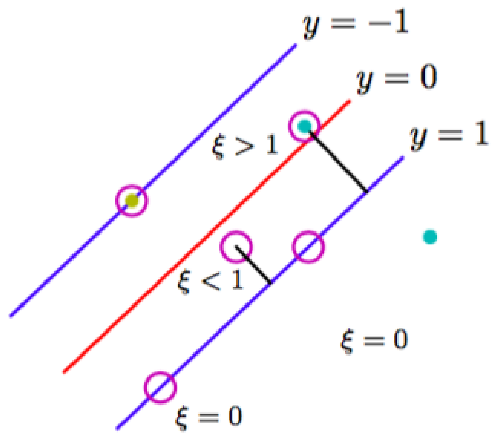


Figure 2: The relationship between the optimal slack variables and the optimal linear separator in the feature space. Support vectors are surrounded with circles.

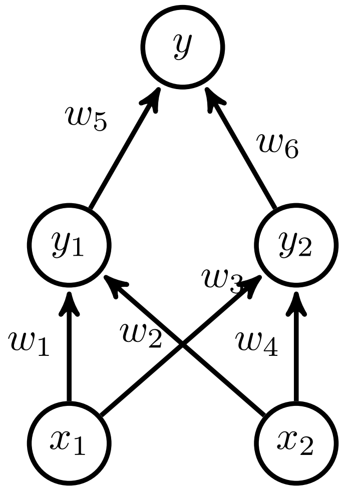
1. [2 pts] Suppose the optimal $\xi_1, \dots, \xi_n$ have been computed. Use the ξ_i to obtain an upper bound on the number of misclassified instances.

2. [4 pts] In the primal optimization of SVM, what's the role of the coefficient C? Briefly explain your answer by considering two extreme cases, i.e., $C \to 0$ and $C \to \infty$.

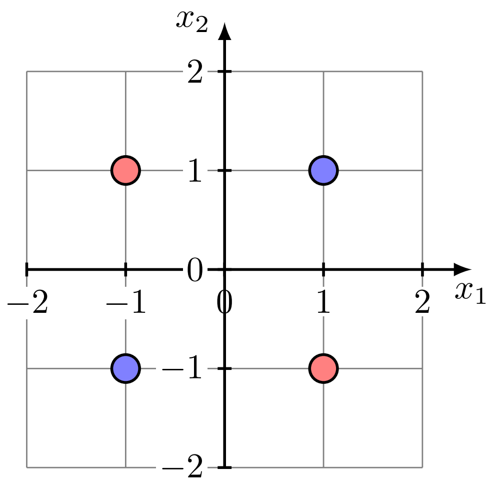
3. [4 pts] Explain how to use the kernel trick to avoid the explicit computation of the feature vector $\phi(\mathbf{x}_i)$? Also, given a new instance $\mathbf{x}$, how to make prediction on the instance without explicitly computing the feature vector $\phi(\mathbf{x})$?

2 Feed-forward Neural Networks [30 pts] (Han)

Consider a feed-forward neural network (FNN) with one input layer, one hidden layer and one output layer shown in Fig. 3a. In this problem we will use the FNN to classify the XOR pattern shown in Fig. 3b. Suppose that the activation functions at every unit in the FNN are linear, i.e., $y_1 = w_1x_1 + w_2x_2$, $y_2 = w_3x_1 + w_4x_2$ and $y = w_5y_1 + w_6y_2 + w_0$. Classify the input instance (x_1, x_2) as $+1$ if $y(x_1, x_2) \geq 0$ otherwise -1.



(a) A three-layer feed-forward neural network with two inputs.



(b) XOR pattern. Instances in blue have label +1 while instances in red have label -1.

1. [10 pts] For any weight configuration $(w_0, w_1, w_2, w_3, w_4, w_5, w_6)$ of the FNN shown above, can you find another FNN with only two layers, i.e., one input layer with two inputs x_1, x_2 and one output layer with one output unit y that computes the same function? If yes, describe such two-layer FNN and give the weights as functions of $(w_0, w_1, w_2, w_3, w_4, w_5, w_6)$. If not, briefly describe your reasoning.

2. [15pts] Show that there is no weight configuration $(w_0, w_1, w_2, w_3, w_4, w_5, w_6)$ under which the FNN shown in Fig. 3a can classify the XOR pattern with no error.

3. [5 pts] For the FNN shown in Fig. 3a, will changing the activation functions at y_1 and y_2 to be nonlinear help classify the XOR pattern? If yes, construct a nonlinear activation function $f(\cdot)$ to be applied only at y_1 and y_2, i.e., $y_1(x_1, x_2) = f(w_1x_1 + w_2x_2)$, $y_2(x_1, x_2) = f(w_3x_1 + w_4x_2)$, such that the new FNN can perfectly classify the XOR pattern. If not, briefly describe your reasoning why it is not possible.

3 AdaBoost [40 pts] (Tianshu)

In this problem, you will explore some interesting properties of AdaBoost through both intuitive examples and mathematical analysis. Then, you will be asked to implement the AdaBoost algorithm with decision stumps as base learners to perform classification on a synthetic dataset.

A decision stump is an axis-aligned linear separator that classifies samples to one side as positive, and samples to the other side as negative. Suppose the input is $\mathbf{x} = (x_1, x_2) \in \mathbb{R}^2$, an example of decision stump is

$$h(\mathbf{x}) = \begin{cases} 1, & \text{if } x_1 \geq 1 \\ -1, & \text{otherwise} \end{cases}$$

An outline of AdaBoost algorithm is provided below.

Given: $(x_1, y_1), ..., (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$
Initialize $D_1(i) = \frac{1}{m}$
For $t = 1, ..., T$:

- Train base learner using distribution D_t
- Get base classifier $h_t : X \to \{-1, +1\}$
- Compute
$$\epsilon_t = \Pr_{i \sim D_t}(y_i \neq h_t(x_i)) = \sum_{i=1}^{m} D_t(i) I(h_t(x_i) \neq y_i) \tag{1}$$
- Compute
$$\alpha_t = \frac{1}{2} \ln \frac{1 - \epsilon_t}{\epsilon_t} \tag{2}$$
- Update
$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t} \tag{3}$$
where Z_t is a normalization factor

Output the final classifier

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right) \tag{4}$$

3.1 Empirical Weak Learnability [15 pts]

In AdaBoost, we assume that at each iteration t we are always able to learn a weak hypothesis h_t in the class of hypotheses $\mathcal{H}$ that is better than random guessing, i.e. $\exists \gamma > 0$ s.t. $\epsilon_t = \frac{1}{2} - \gamma$. This assumption is referred to as empirical γ-weak learnability and is essential to ensure that AdaBoost will drive down training error quickly. This assumption does not hold all the time. In this problem, we will explore the conditions under which this assumption holds.

1. [5 pts] Consider the data in Figure 4. Let the base learners be decision stumps. What is the minimum error rate a single decision stump will achieve on this training set?
2. [5 pts] Consider another dataset in Figure 5. Let the base learners be decision stumps.

   What is a valid classifier h_1 that the algorithm could output in the first iteration? What is its error rate on the training samples?

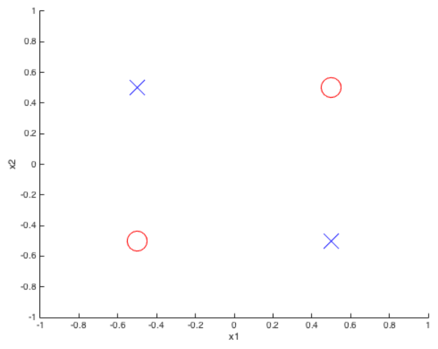


Figure 4: AdaBoost heuristic

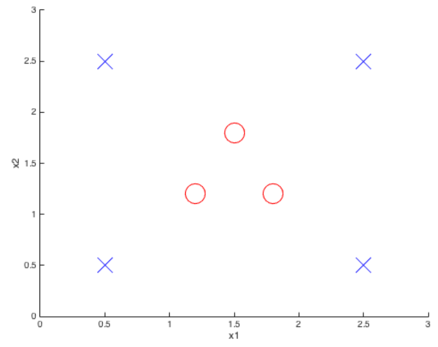


Figure 5: AdaBoost heuristic 2

3. [5 pts] Suppose that the true data generation process is

$$y = \begin{cases} 1, & \text{if } x_1 \in [1,2] \text{ and } x_2 \in [1,2] \\ -1, & \text{otherwise} \end{cases}$$

Consider $h_1(x) = I^*[x_1 \geq 1]$, $h_2(x) = I^*[x_1 \leq 2]$, $h_3(x) = I^*[x_2 \geq 1]$, $h_4(x) = I^*[x_2 \leq 2]$ where $I^*[\cdot] = +1$ if argument is true and -1 otherwise. Let $f(x) = \frac{1}{7}(\sum_{k=1}^{4} h_k(x) - 3)$. Show that for any $x_i \in \mathbb{R}^2$, $y_i f(x_i) \geq \frac{1}{7}$.

4. [Bonus 5 pts] Suppose we are given a training set $S = \{(x_1, y_1), ..., (x_m, y_m)\}$, such that for some weak hypotheses $g_1, ..., g_k$ from the hypothesis space $\mathcal{H}$, and some non-negative coefficients $a_1, ..., a_k$ with $\sum_{j=1}^{k} a_j = 1$, there exists $\theta > 0$ such that

$$\forall (x_i, y_i) \in S, y_i \sum_{j=1}^{k} a_j g_j(x_i) \geq \theta$$

Here, we will show that if the condition above is satisfied, then **for any** distribution D over S, there exists a hypothesis $g_l \in \mathcal{H}$ with error at most $\frac{1}{2} - \frac{\theta}{2}$ over the distribution D. In other words, the training set S is empirically γ-weak learnable with $\gamma = \frac{\theta}{2}$.

(a) Show that there exists a weak hypothesis $g_l \in \{g_1, ..., g_k\}$ such that $\mathbf{E}_{i \sim D}[y_i g_l(x_i)] \geq \theta$.

(b) Show that there exists a weak hypothesis $g_j \in \{g_1, ..., g_k\}$ such that $\Pr_{i \sim D}[y_i \neq g_j(x_i)] \leq \frac{1}{2} - \frac{\theta}{2}$

Hint 1: Taking expectation under the same distribution does not change the inequality conditions.
Hint 2: Consider the property of weighted average

3.2 Complexity and Overfitting [Bonus 5 pts]

From the last section, we have established a sufficient condition for empirical weak learnability. Under those conditions, AdaBoost is guaranteed to have low training error. Now we consider the generalization error of AdaBoost. A typical run of AdaBoost is shown in Figure 6.

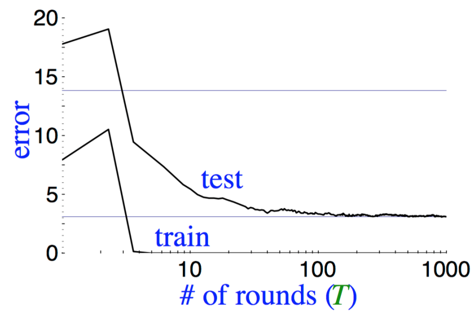


Figure 6: AdaBoost typical run

1. [2 pts] Suppose you want to use decision stumps as the base learners for classifying n data points in $\mathbb{R}^d$. After the first iteration of AdaBoost, how many parameters are there in the classifier $H_1(x) = \text{sign}(\alpha_1 h_1(x))$?
2. [2 pts] Suppose you want to use decision stumps as the base learners for classifying n data points in $\mathbb{R}^d$. After AdaBoost terminates after T iterations, how many parameters are there in the final classifier $H(x)$?
3. [1 pts] Does the Figure 6 indicate overfitting? Does the result contradict your intuition?

3.3 Margin [Bonus 5 pts]

Despite that model complexity increases with each iteration, AdaBoost does not usually overfit. The reason behind this is that the model becomes more "confident" as we increase the number of iterations. The "confidence" can be expressed mathematically as the "margin". Recall that the classifier after the algorithm terminates with T iterations is

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

Similarly, we can define the intermediate weighted classifier after k iterations as:

$$H_k(x) = \text{sign}\left(\sum_{t=1}^{k} \alpha_t h_t(x)\right)$$

As its output is either -1 or 1, it does not tell the confidence of its judgement. Here, without changing the decision rule, let

$$H_k(x) = \text{sign}\left(\sum_{t=1}^{k} a_t h_t(x)\right)$$

where $a_t = \alpha_t / \sum_{t'=1}^{k} \alpha_{t'}$ so that the weights on each weak classifier are normalized. Define the margin after the k-th iteration as the weights of h_t voting correctly minus the weights of h_t voting incorrectly.

$$\text{Margin}_k(x) = \sum_{t:h_t(x)=y} a_t - \sum_{t:h_t(x)\neq y} a_t$$

1. [2 pts] Let $f_k(x) = \sum_{t=1}^{k} a_t h_t(x)$. Show that $\text{Margin}_k(x) = y f_k(x)$.

   Hint: $y \in \{-1, 1\}$ and $h_t(x) \in \{-1, 1\}$

2. [2 pts] Consider Equation (3). Show that

$$D_{k+1}(i) = \frac{\exp(-y_i f_k(x_i))}{m \cdot \prod_{t=1}^{k} Z_t}$$

3. [1 pts] From the result above, if $\text{Margin}_k(x_i) > \text{Margin}_k(x_j)$, which of the samples x_i and x_j will receive a higher weight in iteration $k+1$?

3.4 Remarks

With further mathematical details, it can be shown that boosting increases margin on training examples and that large margin on training examples reduces the generalization error. Thus we have provided an explanation to the observations in Section 3.2.

3.5 Programming [25 pts]

In this section, you will need to implement the AdaBoost algorithm with decision stump as base learners on synthetic datasets. You will be provided with the following files:

- **adaboost_main.m**: A code skeleton that provides general guidelines on how you should write and report your program
- **gen_sample.m**: A helper function that is used to generate synthetic dataset. The generated dataset has two entries **X** and **y**, where $\mathbf{X}_{ij} \in [-2, 2]$ corresponds to the j-th feature of the i-th sample and $\mathbf{y}_i \in \{-1, 1\}$ corresponds to the label of the i-th sample.

The dataset looks like below (Since it is randomly generated, you will not get exactly the same dataset as the one plotted in Figure 7)

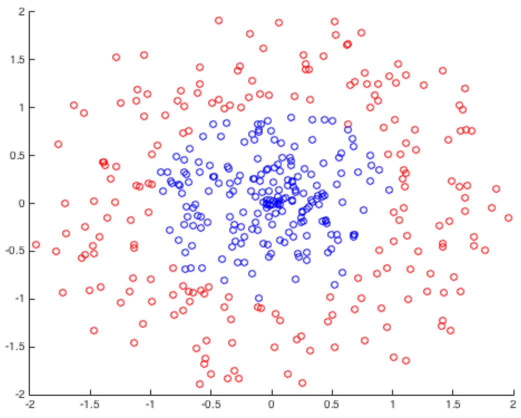


Figure 7: AdaBoost synthetic dataset

You are free to design your own code, but you are required to submit a function **adaboost.m** with the following signature: (along with other supporting files you wrote)

[pred, other_outputs...] = adaboost(trainX, trainY, testX, T)

Here, **trainX** is $m \times 2$ matrix where row i corresponds to the feature values of the ith sample; **trainY** is $m \times 1$ vector where row i corresponds to the label of the ith sample; **testX** is $n \times 2$ vector where row i

corresponds to the feature values of the ith sample. m is the size of training set, n is the size of testing set; **T** is the number of iterations your AdaBoost algorithm will run. The output **pred** should be a $n \times 1$ vector where the ith entry is your label prediction on the ith test sample. Note that your prediction needs to be either $+\mathbf{1}$ or $-\mathbf{1}$.

Recommendations on Implementation:

1. Start with a base learner function that takes in training data and weights of each sample, and outputs the specifications of the best decision stump under the current AdaBoost distribution D_t, e.g. the feature index that your decision stump is separating (x_1 or x_2), the position of the decision stump, and to which side your decision stump will label as 1.
2. Write a function to update the AdaBoost distribution after each iteration.
3. Write a function to predict the labels $\mathbf{y}$ given matrix $\mathbf{X}$. This function should take in the decision stumps that you learned in previous iterations.
4. You are free to use **other_outputs...** to return additional variables that you may need for the report questions. **However, make sure that the first returned value is the vector of predictions**.

3.5.1 Report Questions [10 pts]

Using the functions you wrote and **adaboost_main.m**, do the following experiments and report the results. For these experiments, set size of training set to 400 and size of testing set to 100.

1. [3 pts] At each iteration t, compute the weighted error rate ϵ_t of your weak learner h_t on the training set. For $T = 20$, plot ϵ_t as a function of t. What is the highest weighted error rate a weak learner ever achieves?
2. [2 pts] At each iteration t, compute the error rate of your combined learner H_t on the training set. For $T = 20$, plot your training error as a function of current iteration number t.
3. [2 pts] At each iteration t, compute the error rate of your combined learner H_t on the testing set. For $T = 20$, on the same graph as **3.5.1.2**, plot your testing error as a function of current iteration number t.
4. [3 pts] Set $T = [20, 50, 100, 200]$ respectively. Does the training error decrease with the number of iterations? Does the testing error increase with the number of iterations?
5. [Bonus 5 pts] At each iteration t, compute the minimum margin of H_t on training sample X. Plot the margin as a function of current number of iteration t.

3.6 Remarks

You may notice that in the plot you produce, both the training error and the testing error decreases with the number of iterations, which contradicts our intuition that after some number of iterations, testing error will increase due to overfitting. The main reason is that instead of optimizing an objective function over all the parameters in the model, i.e. $\alpha_1, ..., \alpha_t, h_1, ..., h_t$, AdaBoost only decides the best parameters for the current weak learner. Further, as discussed in Section 3.3, Section 3.4, and Section 3.5.1.5, the margin on training examples increase with number of iterations. It can be shown that the generalization error, i.e. $\Pr_{\mathscr{D}}[yf(x) \leq 0]$ where $\mathscr{D}$ is the true target distribution, will be small if the margin is large.

4 Submission Instructions

Please submit your report and your codes for programming in a tar file and submit to Autolab. Make sure you put all your files in a directory called **hw4**, and run the following command in its top-directory enclosing the folder:

```
>> tar -cvf hw4.tar hw4
```

Then submit **hw4.tar**