

# HOMEWORK 9: LEARNING THEORY & ENSEMBLE METHODS

10-301/10-601 Introduction to Machine Learning (Summer 2024)  
<https://www.cs.cmu.edu/~hchai2/courses/10601/>

OUT: Thursday, July 18th

DUE: Thursday, July 25th

## START HERE: Instructions

- **Collaboration policy:** Collaboration on solving the homework is allowed, after you have thought about the problems on your own. It is also OK to get clarification (but not solutions) from books or online resources, again after you have thought about the problems on your own. There are two requirements: first, cite your collaborators fully and completely (e.g., “Jane explained to me what is asked in Question 2.1”). Second, write your solution *independently*: close the book and all of your notes, and send collaborators out of the room, so that the solution comes from you only. See the Academic Integrity Section on the course site for more information: <https://www.cs.cmu.edu/~hchai2/courses/10601/#Syllabus>
- **Late Submission Policy:** See the late submission policy here: <https://www.cs.cmu.edu/~hchai2/courses/10601/#Syllabus>
- **Submitting your work:**
  - **Programming:** You will submit your code for programming questions on the homework to Gradescope (<https://gradescope.com>). After uploading your code, our grading scripts will autograde your assignment by running your program on a virtual machine (VM). When you are developing, check that the version number of the programming language environment (e.g. Python 3.9.12) and versions of permitted libraries (**only** numpy 1.23.0 for this assignment) match those used on Gradescope. You have a **total of 10 Gradescope programming submissions**. Use them wisely. In order to not waste code submissions, we recommend debugging your implementation on your local machine (or the linux servers) and making sure your code is running correctly first before any Gradescope coding submission.
  - **Written:** For written problems such as short answer, multiple choice, derivations, proofs, or plots, we will be using Gradescope (<https://gradescope.com/>). Please use the provided template. Submissions must be written in LaTeX. Regrade requests can be made, however this gives the TA the opportunity to regrade your entire paper, meaning if additional mistakes are found then points will be deducted. Each derivation/proof should be completed on a separate page. For short answer questions you **should not** include your work in your solution. If you include your work in your solutions, your assignment may not be graded correctly by our AI assisted grader.
- **Materials:** The data that you will need in order to complete this assignment is posted along with the writeup and template on the course website.

For multiple choice or select all that apply questions, shade in the box or circle in the template document corresponding to the correct answer(s) for each of the questions. For LaTeX users, replace `\choice` with

\CorrectChoice to obtain a shaded box/circle, and don't change anything else.

## Instructions for Specific Problem Types

For “Select One” questions, please fill in the appropriate bubble completely:

**Select One:** Who taught this course?

- ☒ Henry Chai
- ☐ Marie Curie
- ☐ Noam Chomsky

If you need to change your answer, you may cross out the previous answer and bubble in the new answer:

**Select One:** Who taught this course?

- ☒ Henry Chai
- ☐ Marie Curie
- ☒ Noam Chomsky

For “Select all that apply” questions, please fill in all appropriate squares completely:

**Select all that apply:** Which are scientists?

- ☒ Stephen Hawking
- ☒ Albert Einstein
- ☒ Isaac Newton
- ☐ I don't know

Again, if you need to change your answer, you may cross out the previous answer(s) and bubble in the new answer(s):

**Select all that apply:** Which are scientists?

- ☒ Stephen Hawking
- ☒ Albert Einstein
- ☒ Isaac Newton
- ☒ I don't know

For questions where you must fill in a blank, please make sure your final answer is fully included in the given space. You may cross out answers or parts of answers, but the final answer must still be within the given space.

**Fill in the blank:** What is the course number?

10-601

10-~~6~~301

## Written Problems (55 points)

### 1 Learning Theory (19 points)

1. Neural the Narwhal is given a classification task to solve, which he decides to use a decision tree learner with 2 binary features  $X_1$  and  $X_2$ . On the other hand, you think that Neural should not have used a decision tree. Instead, you think it would be best to use logistic regression with 16 real-valued features in addition to a bias term. You want to use PAC learning to check whether you are correct. You first train your logistic regression model on  $N$  examples to obtain a training error  $\hat{R}$ .

(a) (1 point) Which of the following case of PAC learning should you use for your logistic regression model?

- ☐ Finite and realizable
- ☐ Finite and agnostic
- ☐ Infinite and realizable
- ☐ Infinite and agnostic

(b) (2 points) What is the upper bound on the true error  $R$  in terms of  $\hat{R}$ ,  $\delta$ , and  $N$ ? You may use big- $\mathcal{O}$  notation if necessary.

Your Answer

(c) (3 points) **Select one:** You want to argue your method has a lower bound on the true error. Assume that you have obtained enough data points to satisfy the PAC criterion with the same  $\epsilon$  and  $\delta$  as Neural. Which of the following is true?

- ☐ Neural's model will always classify unseen data more accurately because it only needs 2 binary features and therefore is simpler.
- ☐ You must first regularize your model by removing 14 features to make any comparison at all.
- ☐ It is sufficient to show that the VC dimension of your classifier is higher than that of Neural's, therefore having a lower bound for the true error.
- ☐ It is necessary to show that the training error you achieve is lower than the training error Neural achieves.

2. In lecture, we saw that we can use our sample complexity bounds to derive bounds on the true error for a particular algorithm. Consider the sample complexity bound for the infinite, agnostic case:

$$N = O\left(\frac{1}{\epsilon^2} \left[ \text{VC}(\mathcal{H}) + \log \frac{1}{\delta} \right]\right).$$

- (a) (2 points) Rewrite the big- $\mathcal{O}$  bound in terms of  $N$  and  $\delta$  using the definition of big- $\mathcal{O}$  notation (i.e. if  $N = O(M)$  (for some value  $M$ ), then there exists a constant  $c \in \mathbb{R}$  such that  $N \leq cM$ ).

Your Answer

- (b) (2 points) Now, using the definition of  $\epsilon$  (i.e.  $|R(h) - \hat{R}(h)| \leq \epsilon$ ), show that with probability at least  $(1 - \delta)$ :

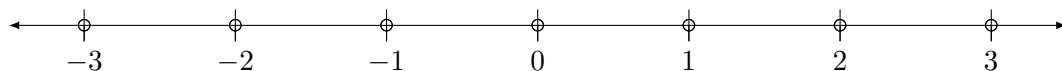
$$R(h) \leq \hat{R}(h) + O\left(\sqrt{\frac{1}{N} \left[ \text{VC}(\mathcal{H}) + \log \frac{1}{\delta} \right]}\right).$$

Your Answer

3. (3 points) Consider the hypothesis space of functions that map  $M$  binary attributes to a binary label. A function  $f$  in this space can be characterized as  $f : \{0, 1\}^M \rightarrow \{0, 1\}$ . Neural the Narwhal says that regardless of the value of  $M$ , a function in this space can always shatter  $2^M$  points. Is Neural wrong? If so, provide a counterexample. If Neural is right, briefly explain why in 1-2 *concise* sentences.

Your Answer

4. Consider an instance space  $\mathcal{X}$  which is the set of real numbers.
- (a) (3 points) **Select one:** What is the VC dimension of hypothesis class  $H$ , where each hypothesis  $h$  in  $H$  is of the form “if  $a < x < b$  or  $c < x < d$  then  $y = 1$ ; otherwise  $y = 0$ ”? (i.e.,  $H$  is an infinite hypothesis class where  $a, b, c$ , and  $d$  are arbitrary real numbers).
- ☐ 2  
☐ 3  
☐ 4  
☐ 5  
☐ 6
- (b) (3 points) Given the set of points in  $\mathcal{X}$  below, construct a labeling of some subset of the points to show that any dimension larger than the VC dimension of  $H$  by *exactly* 1 is incorrect (e.g. if the VC dimension of  $H$  is 3, only fill in the answers for 4 of the points). Fill in the boxes such that for each point in your example, the corresponding label is either 0 or 1. For points you are not using in your example, write N/A.



Answer for -3	Answer for -2	Answer for -1	
Answer for 0	Answer for 1	Answer for 2	Answer for 3

## 2 Ensemble Methods (18 points)

### 2.1 Random Forests

1. (1 point) **True or False:** In a random forest, it is generally better for the trees to be highly correlated, as this reduces variability.  
  - ☐ True
  - ☐ False
2. (2 points) **Select all that apply:** Which of the following is true about OOB error?
  - ☐ OOB error is calculated on a held-out dataset separate from the dataset used to generate bootstrap samples
  - ☐ OOB error is the aggregated value of the errors of subsets of the ensemble on samples those subsets were not trained on
  - ☐ Cross-validation error is the same as OOB error
  - ☐ OOB error is a valid method of estimating true error
  - ☐ None of the above
3. (2 points) **Select all that apply:** Which of the following are hyperparameters that can be tuned in a random forest?
  - ☐ Number of trees trained
  - ☐ Number of points used to train each decision tree
  - ☐ Size of feature subsets used to train each decision tree
  - ☐ Which features are used for splits in each decision tree
  - ☐ None of the above

4. In this question, we will now derive an error bound for random forests in the case of **binary classification**. Given a random forest of  $B$  trees  $\{h_i(x)\}_{i=1}^B$  and a sample  $(x, y)$  drawn from some data distribution  $\mathcal{D}$ , define the classification margin as:

$$m(x, y) = \frac{1}{B} \left( \sum_{i=1}^B \mathbb{I}[h_i(x) = y] - \sum_{i=1}^B \mathbb{I}[h_i(x) \neq y] \right)$$

In words, the margin  $m(x, y)$  is the difference between the average vote for the correct label and the average vote for the incorrect label.

- (a) (1 point) **Fill in the blank:** For any example  $(x, y)$ , the example is classified incorrectly if and only if  $m(x, y) \leq \underline{\hspace{1cm}}$ . Assume majority vote ties are classified incorrectly.

Answer

- (b) (2 points) Observe that  $P_{(x,y) \sim \mathcal{D}}(m(x, y) \leq c)$ , where  $c$  is your answer to part (a), corresponds to the generalization error of the ensemble. Additionally, for an ensemble  $\{h_i(x)\}_{i=1}^B$ , define the strength of the ensemble as  $s = \mathbb{E}_{(x,y) \sim \mathcal{D}}[m(x, y)]$ .

Assume  $s > 0$ . Derive a bound for the generalization error in the form  $P(m(x, y) < c) \leq d$ , where  $d$  is an expression in terms of  $s$  and  $\text{Var}(m(x, y))$ . Show your work.

*Hint:* You should use Chebyshev's inequality, which states that for any random variable  $X$  with finite expectation and variance and any constant  $a > 0$ , we have

$$P(|X - \mathbb{E}[X]| \geq a) \leq \frac{\text{Var}(X)}{a^2}$$

Generalization error bound



(c) (1 point) **Select one:** Through some additional manipulation, it is possible to show that  $\text{Var}(m(x, y)) \leq \bar{\rho}(1 - s^2)$ , where  $\bar{\rho}$  is the mean correlation between trees in the ensemble. Substitute this into your bound from part (b). Which of the following describes how the error bound is affected by  $s$  and  $\bar{\rho}$ ?

- ☐ The error bound gets smaller as  $\bar{\rho}$  increases and  $s$  increases.
- ☐ The error bound gets smaller as  $\bar{\rho}$  increases and  $s$  decreases.
- ☐ The error bound gets smaller as  $\bar{\rho}$  decreases and  $s$  increases.
- ☐ The error bound gets smaller as  $\bar{\rho}$  decreases and  $s$  decreases.

## 2.2 AdaBoost

5. (1 point) **True or False:** Consider some training point  $(x^{(i)}, y^{(i)})$  to the AdaBoost algorithm. If for all  $t$ , the weak learner  $h_t$  learned during training at time  $t$  correctly classifies  $h_t(x^{(i)}) = y^{(i)}$ , there will eventually be a finite time  $t$  such that the weight assigned to  $x^{(i)}$  in the training distribution  $\omega_t$  reaches exactly 0.
- ☐ True
- ☐ False
6. (1 point) **True or False:** If the ensemble learned by AdaBoost reaches perfect training accuracy, all weak learners created in subsequent iterations will be identical (i.e., they will produce the same output on any input). Assume we are using deterministically selected weak learners.
- ☐ True
- ☐ False
7. Assume we use a deterministic training procedure for weak learners. Suppose for some iteration  $t'$  of AdaBoost we find that the weak classifier learned by the algorithm at time  $t'$  has error  $\epsilon_{t'} = 0.5$  of the weak learner  $h_{t'}$  on the training distribution weighted by  $\omega_{t'}$ .
- (a) (1 point) What weight  $\alpha_{t'}$  will AdaBoost assign to the classifier  $h_{t'}$  from above?

$\alpha_{t'}$

- (b) (1 point) **Select all that apply:** In which of the following cases will  $\omega_{t'+1}^{(i)} > \omega_{t'}^{(i)}$  (in other words, in which of the following cases will the weight of training sample  $(x^{(i)}, y^{(i)})$  *strictly* increase from time step  $t'$  to  $t' + 1$ )?
- ☐  $h_{t'}(x^{(i)}) = y^{(i)}$  ( $h_{t'}$  classifies  $x^{(i)}$  correctly)
- ☐  $h_{t'}(x^{(i)}) \neq y^{(i)}$  ( $h_{t'}$  classifies  $x^{(i)}$  incorrectly)
- ☐ None of the above.
- (c) (1 point) **Select all that apply:** Which of the following are true about the next iteration of the AdaBoost algorithm?
- ☐ The errors  $\epsilon_{t'+1}$  and  $\epsilon_{t'}$  are equivalent
- ☐ The weak learners  $h_{t'+1}$  and  $h_{t'}$  will be equivalent (i.e., they will have the same output for every input)
- ☐ None of the above

8. In the following question, we will examine the generalization error of AdaBoost using a concept known as the *classification margin*.

Throughout the question, use the following definitions:

- $T$ : The number of iterations used to train AdaBoost.
- $N$ : The number of training samples.
- $S = \{(x^{(1)}, y^{(1)}), \dots, (x^{(N)}, y^{(N)})\}$ : The training samples with binary labels ( $y^{(i)} \in \{-1, +1\}$ ).
- $\omega_t^{(i)}$ : The weight assigned to training example  $i$  at time  $t$ . Note that  $\sum_i \omega_t^{(i)} = 1$ .
- $h_t$ : The weak learner constructed at time  $t$  (a function  $X \rightarrow \{-1, +1\}$ ).
- $\epsilon_t$ : The weighted (by  $\omega_t$ ) error of  $h_t$ .
- $Z_t = 2\sqrt{\epsilon_t(1 - \epsilon_t)}$ : The normalization factor for the distribution update at time  $t$ .
- $\alpha_t = \frac{1}{2} \ln((1 - \epsilon_t)/\epsilon_t)$ : The weight assigned to the learner  $h_t$  in the composite hypothesis.
- $H_t(x) = (\sum_{t'=1}^t \alpha_{t'} h_{t'}(x)) / (\sum_{t'=1}^t \alpha_{t'})$ : The majority vote of the weak learners, rescaled based on the total weights.
- $g_t(x) = \text{sign}(H_t(x))$ : The voting classifier decision function.

For a binary classification task, assume that we use a probabilistic classifier that provides a probability distribution over the possible labels (i.e.  $p(y|x)$  for  $y \in \{+1, -1\}$ ). The classifier output is the label with highest probability. We define the *classification margin* for an input as the signed difference between the probability assigned to the correct label and the incorrect label  $p_{\text{correct}} - p_{\text{incorrect}}$ , which takes on values in the range  $[-1, 1]$ . Recall from recitation that  $\text{margin}_t(x^{(i)}, y^{(i)}) = y^{(i)} H_t(x^{(i)})$ .

**Note:** Consistency points will not be awarded for this question.

- (a) (1 point) Recall the update AdaBoost performs on the distribution of weights:

- $\omega_1^{(i)} = 1/N$
- $\omega_{t+1}^{(i)} = \omega_t^{(i)} \frac{\exp(-y^{(i)} \alpha_t h_t(x^{(i)}))}{Z_t} = \frac{1}{N} \left( \prod_{t'=1}^t \frac{1}{Z_{t'}} \right) \exp(-\sum_{t'=1}^t y^{(i)} \alpha_{t'} h_{t'}(x^{(i)}))$

We define  $C_{t+1} = \frac{1}{N} \left( \prod_{t'=1}^t \frac{1}{Z_{t'}} \right)$  and  $M_{t+1}(i) = -\sum_{t'=1}^t y^{(i)} \alpha_{t'} h_{t'}(x^{(i)})$ . We then have

$$\omega_{t+1}^{(i)} = C_{t+1} \exp(M_{t+1}(i))$$

Let  $\alpha = \sum_{t'=1}^t \alpha_{t'}$ . Rewrite  $M_{t+1}(i)$  in terms of  $\text{margin}_t(x^{(i)}, y^{(i)})$  and  $\alpha$ . (Hint: first rewrite  $M_{t+1}(i)$  in terms of  $y^{(i)}, \alpha, H_t, x^{(i)}$ , then apply our given formula for the margin).

Your Answer

- (b) (1 point) **Select one:** Note that  $C_{t+1}, \alpha$  are treated as positive constants with respect to the input points. Using the classification margin and the above formulation of the weights assigned by AdaBoost, fill in the blanks to describe which points AdaBoost assigns high weight to at time  $t$ .

At time  $t$ , AdaBoost assigns higher weight to points  $x^{(i)}$  with \_\_\_\_\_ value of margin on the current ensemble classifier (i.e.,  $\text{margin}_t(x^{(i)}, y^{(i)})$ ).

- ☐ higher absolute
  - ☐ higher signed
  - ☐ lower absolute
  - ☐ lower signed
- (c) (1 point) **Select one:** This weighting behavior causes the margins of the points you chose in part (b) to \_\_\_\_\_.
- ☐ increase
  - ☐ decrease
  - ☐ stay the same
- (d) (1 point) **Select all that apply:** How does this change in the margins explain the empirical result of test error continuing to decrease after training error has converged?
- ☐ AdaBoost can continue to increase the confidence of its predictions, particularly on lower confidence training examples, which makes it less likely at test time to misclassify points similar to those it has seen in the training set.
  - ☐ Adaboost continues to increase confidence of its predictions on high confidence training examples only, leading to a highly accurate classifier which, at test time, will outperform the training error.
  - ☐ Adaboost lowers the confidence in high confidence areas by continuing to train on low confidence training examples. This averages out the confidence between high and low confidence training examples, overall creating a more generalizable classifier.
  - ☐ The test error does not continue to decrease after the training error has converged as the model stops updating once we reach convergence.
  - ☐ None of the above.

### 3 Empirical Question (18 points)

1. (12 points) In order to visualize the changing weights in the AdaBoost algorithm, you will first recreate the figures presented in lecture using a subset of the training data plotted in Figure 1. Specifically, for each of the first three iterations of AdaBoost, you should generate two plots (six total plots):

1. A scatter plot of the first 20 training data points, where points with label +1 are blue +’s, points with label -1 are red -’s and each point’s size is proportional to its weight *squared*. Specifically, the formula you should use for a point’s size is

$$100(N\omega_i^{(t)})^2$$

where  $N$  is the number of training data points. This scaling is just done for the purposes of plotting, to ensure that the data points are reasonably sized in the figures.

2. The same scatter plot except with the decision stump learned on all of the training data overlaid on top of it. The decision boundary should be represented as a solid, black line and the regions that the decision stump would classify as +1 and -1 shaded in blue and red respectively.

These figures should all be generated *using the weights at the start of the iteration* before any updating, e.g., in the first two figures you generate, you should use the weights  $\omega_i^{(0)} = 1/n$  and the corresponding stump you would learn with those weights.

For your convenience, we have provided the following link to a Jupyter notebook with Python code that plots the figures we are requesting with dummy data: [AdaBoost\\_plots.ipynb](#)



Weighted plot start of iteration 2



Weighted plot start of iteration 2 with stump



Weighted plot start of iteration 3



Weighted plot start of iteration 3 with stump



2. (6 points) Test your classifier by training with varying the number of iteration(num\_iter) from 1 to 50. Make a plot of the test accuracy versus number of iterations from 1 to 50. Be sure to label your graph sufficiently.

Your Answer

## 4 Collaboration Questions

After you have completed all other components of this assignment, report your answers to these questions regarding the collaboration policy. Details of the policy can be found [here](#).

1. Did you receive any help whatsoever from anyone in solving this assignment? If so, include full details.
2. Did you give any help whatsoever to anyone in solving this assignment? If so, include full details.
3. Did you find or come across code that implements any part of this assignment? If so, include full details.

Your Answer



## 5 Programming (41 points)

### 5.1 Model Definition

In this section, you will implement the AdaBoost algorithm with decision stumps, i.e., 1-level decision trees. You will need to implement the AdaBoost algorithm from scratch and train and test the classifier on the datasets provided.

The 1-level decision stumps are just binary questions asking whether the point is greater or less than a particular dividing line—the dividing line can be horizontal or vertical. Each potential dividing line for the decision stumps must come from the training data values, i.e., when minimizing the weighted training error, you should iterate through all possible decision stumps of the form:

$$h(\mathbf{x}) = \begin{cases} +1 & \text{if } x_d > x_d^{(i)} \\ -1 & \text{otherwise} \end{cases} \quad \text{and} \quad h(\mathbf{x}) = \begin{cases} -1 & \text{if } x_d > x_d^{(i)} \\ +1 & \text{otherwise} \end{cases}$$

where  $x_d^{(i)}$  is the value in dimension  $d$  of the  $i^{\text{th}}$  training data point. You should also consider the decision stumps:

$$h(\mathbf{x}) = \begin{cases} +1 & \text{if } x_1 > \min(x_1^{(i)}) - \epsilon \\ -1 & \text{otherwise} \end{cases} \quad \text{and} \quad h(\mathbf{x}) = \begin{cases} -1 & \text{if } x_1 > \min(x_1^{(i)}) - \epsilon \\ +1 & \text{otherwise} \end{cases}$$

where  $\epsilon = 0.000001$ , i.e., a stump that classifies every training data point as positive and a stump that classifies every training data point as negative (note that there are a few possible stumps that can achieve these classifications using the training data values and we have arbitrarily broken the tie for you).

We recommend designing your code in the following way. However you can write your program according to your design flow.

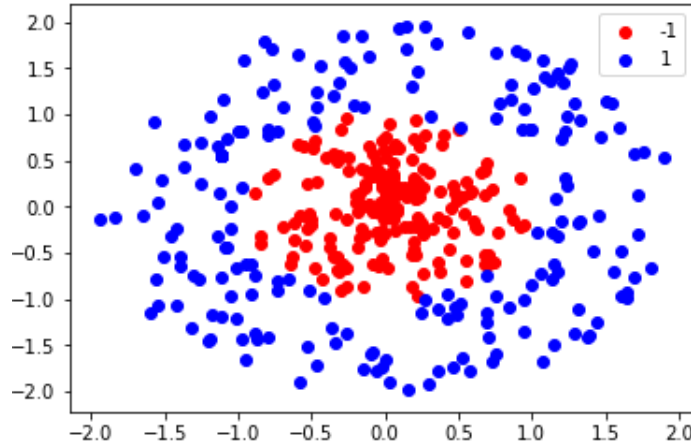


Figure 1: Example dataset. Each point  $\mathbf{x} \in [-2, 2] \times [-2, 2]$  and  $y \in \{-1, 1\}$

### 5.2 The Datasets

The dataset for this task is synthetically generated and has two entries  $\mathbf{x}^{(i)} = (x_1^{(i)}, x_2^{(i)})$  and  $y^{(i)}$ . Here  $\mathbf{x}^{(i)} = (x_1, x_2) \in [-2, 2] \times [-2, 2]$  is the  $i$ -th instance and  $y^{(i)} \in \{-1, 1\}$  is its corresponding label. All the datasets are in the Data file

### 5.3 Structure for the adaboost.py

1. **read\_data**: This function reads the data from the input file.
2. **classify**: This function returns the predictions of a 1-level decision tree (stump).
3. **weak\_classifier**: This function finds the best weak classifier, which is a 1-level decision tree. Defining  $N$  as number of training points and  $D = 2$  as the number of features per data point, the inputs to the function should be the input data ( $N \times D$  array), the true labels ( $N \times 1$  array) and current point weights  $\omega_i^{(t)} \in \Omega^t$  ( $N \times 1$  array) and you should return the best weak classifier with the best split based on the weighted error. Some of the things to include in the output can be: best feature index, best split value, label, value of  $\alpha_t$  and predicted labels by the best weak classifier. Note:  $\alpha_t$  should be a ( $T \times 1$  array, for  $t = 1 \dots T$ , and  $T$  denotes the number of iterations that you choose to run the boosting algorithm)
4. **update\_weights**: This function computes the updated weights  $\Omega^{t+1}$ , The inputs to this function should be current weights  $\Omega^t$  ( $N \times 1$  array), the value of  $\alpha_t$ , the true target values ( $N \times 1$  array) and the predicted target values ( $N \times 1$  array). And the function should output the updated distribution  $\Omega^{t+1}$ .
5. **train**: This function trains the model by using AdaBoost algorithm. It will be tested and needs to output: the final list of weak classifiers ( $T \times 3$ ), alphas ( $T \times 1$ ), weights ( $N \times 1$ ), and test accuracies across iterations ( $T \times 1$ ).
6. **evaluate**: This function evaluates the model with test data by measuring the accuracy. Assuming we have  $M$  test points, the inputs should be the test data ( $M \times D$  array), the true labels for test data ( $M \times 1$  array), the array of weak classifiers ( $T \times 3$  array), and the array of  $\alpha_t$  for  $t = 1 \dots T$  ( $T \times 1$  array). This function will be tested and needs to output: the predicted labels ( $M \times 1$  array) and the accuracy.
7. **main**: Here is an example main function that you can use to get started.

```
def main(args):
    num_iters = args.num_iters
    train_data, train_labels, test_data, test_labels = \
        read_data(args)

    classifier_list, alpha_list, weights, test_accuracies = \
        train(num_iters, train_data, train_labels, test_data,
              test_labels)

    accuracy, predictions = predict(test_data, test_labels,
                                    classifier_list, alpha_list)
```

### 5.4 Details on Implementation:

- Start with a base learner function that takes in training data and weights of each sample, and outputs 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**.
- Your preliminary prediction might belongs to  $\{0, 1\}$ , in order to transform it to **+1, -1** you can consider using the following suggestion:

$$prediction = preliminary\_prediction * 2 - 1, \quad prediction \in \{-1, +1\}$$

- Make sure to use the natural log for log.

## 5.5 Command Line Arguments

The autograder will use the following commands to call your function:

```
$ python adaboost.py [args...]
```

where above `[args...]` is a placeholder for command-line arguments: `<train_data>` `<test_data>` `<num_iters>` `<metrics_out>` `<test_out>` `<weights_out>`. These arguments are described in detail below:

1. `<train_data>`: path to the training input .csv file
2. `<test_data>`: path to the test input .csv file
3. `<num_iters>`: number of iterations of adaboost
4. `<metrics_out>`: path to the output .out file which includes test accuracies over iterations
5. `<test_out>`: path to the output .out file which includes final predictions on test data
6. `<weights_out>`: path to the output .out file which includes final point weights over train data

Example command:

```
$ python3 adaboost.py --train_data data/train_adaboost.csv \  
--test_data data/test_adaboost.csv --num_iters 5 \  
--metrics_out metrics.out --test_out test.out \  
--weights_out weights.out
```

The result outputs of this example command are given in the result file in the handout.

## 5.6 Gradescope Submission

You should submit your `adaboost.py` to Gradescope. **Any other files will be deleted.** Please do not use any other file name for your implementation. This will cause problems for the autograder to correctly detect and run your code.

**Note:** For this assignment, you may make up to **10** submissions to Gradescope before the deadline, but only your last submission will be graded.