CS 446 / ECE 449 — Homework 3

yuhangr2

Version 1.0

Instructions.

- Homework is due **Wednesday, October 13, at noon CST**; you have **3** late days in total for **all Homeworks**.
- Everyone must submit individually at gradescope under `hw3` and `hw3code`.
- The "written" submission at `hw3` **must be typed**, and submitted in any format gradescope accepts (to be safe, submit a PDF). You may use LaTeX, markdown, google docs, MS word, whatever you like; but it must be typed!
- When submitting at `hw3`, gradescope will ask you to **mark out boxes around each of your answers**; please do this precisely!
- Please make sure your NetID is clear and large on the first page of the homework.
- Your solution **must** be written in your own words. Please see the course webpage for full **academic integrity** information. You should cite any external reference you use.
- We reserve the right to reduce the auto-graded score for `hw3code` if we detect funny business (e.g., your solution lacks any algorithm and hard-codes answers you obtained from someone else, or simply via trial-and-error with the autograder).
- When submitting to `hw3code`, only upload `hw3.py` and `hw3_utils.py`. Additional files will be ignored.

Version History.

1. Initial version.

1. Ensemble Methods

In this question, you will implement several ensemble methods including Bagging and AdaBoost on a simple dataset. The methods will learn a binary classification of 2D datapoints in $[-1, 1]^2$.

We have provided a few helper functions in `hw3_utils.py`.

- `visualization()` visualizes a dataset and the ensemble's predictions in 2D space.
- `get_dataset_fixed()` returns a simple dataset with pre-defined datapoints. Please use this dataset for the plot.
- `get_dataset_random()` returns a simple dataset by random construction. You may play with it and test your algorithm.

You will need to implement functions and classes defined in `hw3.py`. When uploading to Gradescope, please pack the two files `hw3_utils.py` and `hw3.py` (without folder) together into one zip file.

(a) **Weak learner**

To begin with, you will implement a weak learner to do the binary classification.

A decision stump is a one-level decision tree. It looks at a single feature, and then makes a classification by thresholding on this feature. Given a dataset with positive weights assigned to each datapoint, we can find a stump that minimizes the weighted error:

$$L = \sum_{i=1}^{n} w^{(i)} \cdot \mathbf{1}(y^{(i)} \neq \hat{y}^{(i)})$$

Here $w^{(i)}$ is the weight of the i-th datapoint, and the prediction $\hat{y}^{(i)}$ is given by thresholding on the k-th feature of datapoint $\boldsymbol{x}^{(i)}$:

$$\hat{y}^{(i)} = \begin{cases} s, & \text{if } x_k^{(i)} \geq t \\ -s, & \text{otherwise} \end{cases}$$

For the 2D dataset we have, the parameters of this stumps are the sign $s \in \{+1, -1\}$, the feature dimension $k \in \{1, 2\}$, and the threshold $t \in [-1, 1]$. In this question, your task is to find out the best stump given the dataset and weights.

Learning a decision stump requires learning a threshold in each dimension and then picking the best one. To learn a threshold in a dimension, you may simply sort the data in the chosen dimension, and calculate the loss on each candidate threshold. Candidates are midpoints between one point and the next, as well as the boundaries of our range of inputs.

Please implement the `Stump` class in `hw3.py`. You may define your own functions inside the class, but do not change the interfaces of `__init__()` and `predict()`. Please read template file for further instructions.

(b) **Weak learner's predictions**

Now test your implementation of `Stump` on the dataset given by `get_dataset_fixed()`. Suppose all the datapoints are equally weighted. Please answer the following questions in your written submission:

- What is your decision function?
- How many datapoints are mis-classified?
- Using the helper function `visualization()`, include a visualization of your stump's predictions.

(c) **Bagging**

As we have learned from the class, we can utilize ensemble methods to create a strong learner from weak learners we have for part (a). Please complete `bagging()` in `hw3.py`. This function should take the whole dataset as input, and sample a subset from it in each step, to build a list of different weak learners.

Please do not change the random sampling of `sample_indices`, and use the default random `seed=0`, so that your code can behave consistently in the autograder.

(d) **AdaBoost**

Now please implement AdaBoost algorithm. As we have learned in class, in each step of AdaBoost, it

- Finds the optimal weak learner according to current data weights
- Acquires the weak learner's predictions
- Calculates the weight for this weak learner
- Updates the weights for datapoints

Complete `adaboost()` in `hw3.py`. It should return a series of weak learners and their weights.

(e) **Visualization**

Run your Bagging and AdaBoost algorithms on the fixed dataset given by `get_dataset_fixed()`. Set the number of weak classifiers to 20, and for Bagging, set the number of samples to 15 for learning each classifier. Please answer the following questions in your written submission:

- Are they performing better than the individual weak learner in (b)?
- Include visualizations for both algorithms in your written submission.

Solution.

(b) My decision function is: $\hat{y} = -1$ if $x_1 \geq -0.6$, 1 otherwise.
Five datapoints are mis-classified.
Visualization is included.

(e) For Bagging, it is not performing better. For AdaBoost, it is performing better. Visualizations are included.

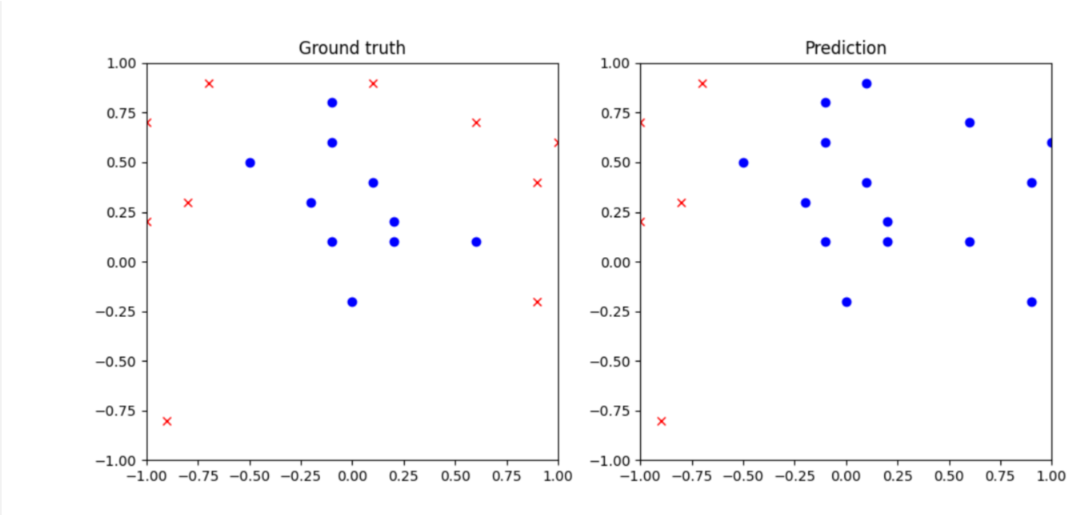

Figure 1: Q2(b): Visualization of a stump decision tree.

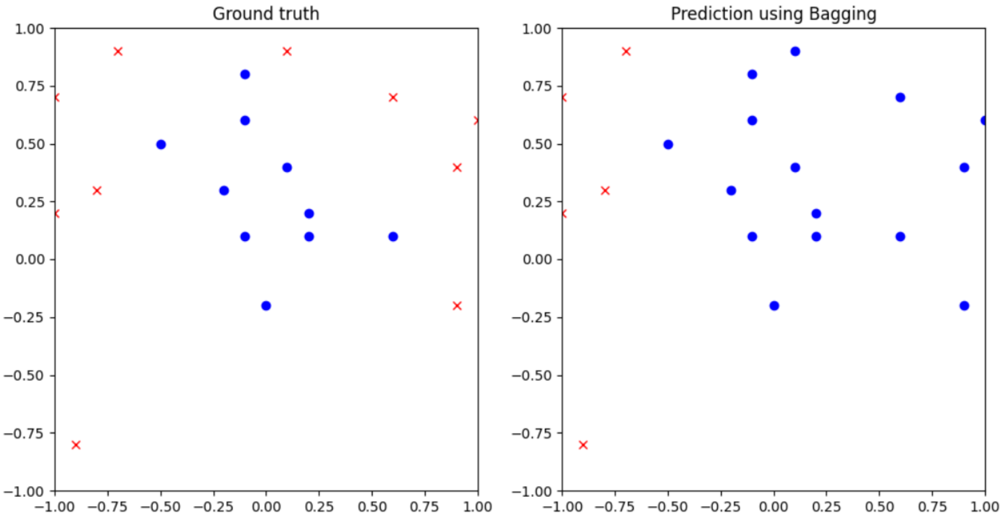

Figure 2: Q2(e): Visualization of Bagging.

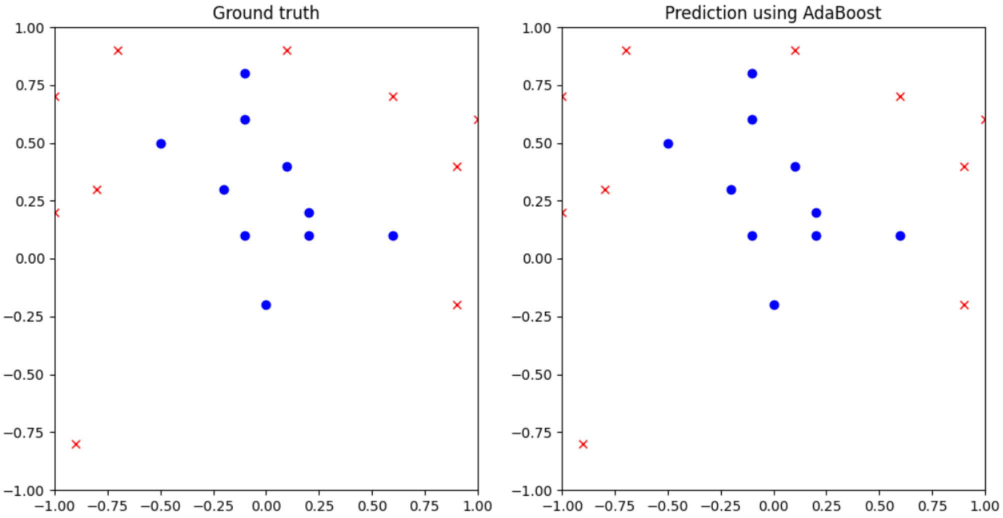

Figure 3: Q2(e): Visualization of AdaBoost.

2. Learning Theory.

(a) **VC Dimensions.** In this problem, we'll explore VC dimensions! First, a few definitions that we will use in this problem. For a feature space $\mathcal{X}$, let $\mathcal{F}$ be a set of binary classifier of the form $f : \mathcal{X} \to \{0, 1\}$. $\mathcal{F}$ is said to **shatter** a set of k distinct points $\{\boldsymbol{x}^{(i)}\}_{i=1}^{k} \subset \mathcal{X}$ if for each set of label assignments $(y^{(i)})_{i=1}^{k} \in \{0,\ 1\}^k$ to these points, there is an $f \in \mathcal{F}$ which makes no mistakes when classifying D.

The VC Dimension of $\mathcal{F}$ is the largest non-negative integer $k \in$ such that there is a set of k points that $\mathcal{F}$ can shatter. Even more formally, let $VC(\mathcal{F})$ denote the VC Dimension of $\mathcal{F}$. It can be defined as:

$$VC(\mathcal{F}) = \max_{k}\ k \qquad \text{s.t. } \exists \{\boldsymbol{x}^{(i)}\}_{i=1}^{k} \subset \mathcal{X}, \forall (y^{(i)})_{i=1}^{k} \in \{0,\ 1\}^k\ , \exists f \in \mathcal{F}, \forall i : f(\boldsymbol{x}^{(i)}) = y^{(i)}$$

The intuition here is that VC dimension captures some kind of complexity or capacity of a set of functions $\mathcal{F}$.

Note: The straightforward proof strategy to show that the VC dimension of a set of classifiers is k is to first show that for a set of k points, the set is shattered by the set of classifiers. Then, show that any set of $k + 1$ points cannot be shattered. You can do that by finding an assignment of labels which cannot be correctly classified using $\mathcal{F}$.

Notation: We denote $\mathbf{1}_{\text{condition}}(\cdot) : \mathcal{X} \to \{0, 1\}$ to be the indicator function, i.e., $\mathbf{1}_{\text{condition}}(x) = 1$ if the condition is true for x and $\mathbf{1}_{\text{condition}}(x) = 0$ otherwise.

We will now find the VC dimension of some basic classifiers.

i. **1D Affine Classifier**

Let's start with a fairly simple problem. Consider $\mathcal{X} = \mathbb{R}$ and $\mathcal{Y} = \{0, 1\}$. Affine classifiers are of the form:

$$\mathcal{F}_{\text{affine}} = \{\mathbf{1}_{wx+b\geq 0}(\cdot) : \mathcal{X} \to \mathbb{R} \mid w, b \in \mathbb{R}\},$$

Show what is $VC(\mathcal{F}_{\text{affine}})$ and prove your result.

Hint: Try less than a handful of points.

ii. **General Affine Classifier**

We will now go one step further. Consider $\mathcal{X} = \mathbb{R}^k$ for some dimensionality $k \geq 1$, and $\mathcal{Y} = \{0, 1\}$. Affine classifiers in k dimensions are of the form

$$\mathcal{F}^{k}_{\text{affine}} = \{\mathbf{1}_{\boldsymbol{w}^\top \boldsymbol{x}+b\geq 0}(\cdot) : \mathcal{X} \to \mathbb{R} \mid \boldsymbol{w} \in \mathbb{R}^k, b \in \mathbb{R}\}$$

Show what is $VC(\mathcal{F}^{k}_{\text{affine}})$ and prove your result.

Hint: Note that $\boldsymbol{w}^\top \boldsymbol{x} + b$ can be written as $[\boldsymbol{x}^\top\ \ 1]\begin{bmatrix} \boldsymbol{w} \\ b \end{bmatrix}$. Moreover, consider to put all data points into a matrix, e.g.,

$$\boldsymbol{X} = \begin{bmatrix} (\boldsymbol{x}^{(1)})^\top & 1 \\ (\boldsymbol{x}^{(2)})^\top & 1 \\ \vdots & \vdots \end{bmatrix}.$$

iii. **Decision Trees**

Consider $\mathcal{X} = \mathbb{R}^k$ for some dimensionality $k \geq 1$, and $\mathcal{Y} = \{0, 1\}$. Show that the VC dimension of the axis-aligned (coordinate-splits) decision trees is infinite.

Hint: Consider an arbitrary dataset, and show that a decision tree can be constructed to exactly fit that dataset.

(b) **Rademacher Complexity.** Recall from class that the generalization error bound scales with the complexity of the function class $\mathcal{F}$, which, in turn, can be measured via Rademacher complexity. In this question we will compute the Rademacher complexity of linear functions step by step. Let's consider a dataset $\{\boldsymbol{x}^{(i)}\}_{i=1}^{n} \subset \mathbb{R}^k$ with the norm bounded by $\|\boldsymbol{x}^{(i)}\|_2 \leq R$ and the set of linear classifiers $\mathcal{F} = \{\boldsymbol{x} \mapsto \boldsymbol{w}^\top \boldsymbol{x}\ \mid \boldsymbol{w} \in \mathbb{R}^k, \|\boldsymbol{w}\|_2 \leq W\}$.

i. For a fixed sign vector $\boldsymbol{\epsilon} = (\epsilon_1, ..., \epsilon_n) \in \{\pm 1\}^n$ show that:

$$\max_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \epsilon_i f(\boldsymbol{x}^{(i)}) \leq W \|\boldsymbol{x}_{\boldsymbol{\epsilon}}\|_2$$

where $\boldsymbol{x}_{\boldsymbol{\epsilon}}$ is defined as $\boldsymbol{x}_{\boldsymbol{\epsilon}} = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{x}^{(i)} \epsilon_i$.
Hint: Cauchy-Schwarz inequality.

ii. Assume ϵ_i is distributed i.i.d. according to $\Pr[\epsilon_i = +1] = \Pr[\epsilon_i = -1] = 1/2$. Show that

$$\mathbb{E}_{\boldsymbol{\epsilon}} \left[\|\boldsymbol{x}_\epsilon\|^2 \right] \leq \frac{R^2}{n}$$

iii. Assume ϵ_i follows the same distribution as previous problem. Recall the definition of Rademacher complexity:

$$\text{Rad}(\mathcal{F}) = \mathbb{E}_{\boldsymbol{\epsilon}} \left[\max_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \epsilon_i f(\boldsymbol{x}^{(i)}) \right]$$

Show that the Rademacher complexity of the set of linear classifiers is:

$$\text{Rad}(\mathcal{F}) \leq \frac{RW}{\sqrt{n}}$$

Hint: Jensen's inequality.

Solution.

(a) i. $VC(\mathcal{F}_{\text{affine}}) = 2$.

For different 2 points, we can always find ω, b, such that the line $y = wx + b$ divides the two points.
For three different points x_1, x_2, x_3 such that $x_1 < x_2 < x_3$ and $x_2 = x_1 + \sigma_1, x_3 = x_2 + \sigma_2 where \sigma_1, \sigma_2 > 0$, if x_1 and x_3 belong to the class 0 and x_2 belongs to the class 1, then $wx_1 + b < 0$ and $wx_2 + b = wx_1 + w\sigma_1 + b \geq 0$. Adding the two inequalities we get $w > 0$. Then $wx_3 + b = wx_2 + b + w\sigma_2 > 0$ and we label x_3 with class 1. However, the true label of x_3 is 0.
Hence, we always get a mistake for any 3 different points with a "0,1,0" label pattern.
Therefore, $VC(\mathcal{F}_{\text{affine}}) = 2$.

ii. $VC(\mathcal{F}_{\text{affine}}) = k + 1$. Proof:

The prediction function can be written as $[\boldsymbol{x}^\top \quad 1] \begin{bmatrix} \boldsymbol{w} \\ b \end{bmatrix} = \boldsymbol{X}\boldsymbol{w}'$, where $\boldsymbol{X} \in \mathbb{R}^{m \times k+1}$ and $\boldsymbol{w}' \in \mathbb{R}^{k+1 \times 1}$. Let the true label to be $\boldsymbol{y} \in \mathbb{R}^{m \times 1}$.
If $m = k + 1$, we can find a set of points such that $\boldsymbol{X}$ has rank $k + 1$. Then, for the learning objective $\boldsymbol{X}\boldsymbol{w}' = \boldsymbol{y}$, we can always find the solution $\boldsymbol{w}' = \boldsymbol{X}^{-1}\boldsymbol{y}$, which predicts perfectly on every training sample.
If $m = k + 2$, $\boldsymbol{X}$ has a maximum rank of $k + 2$. Then the column space of $\boldsymbol{X}$, which is exactly the range of $\boldsymbol{X}\boldsymbol{w}'$, $\forall \boldsymbol{w}' \in \mathbb{R}^{k+1 \times 1}$, would have a maximum dimension of $k + 1$. However, the labels $\boldsymbol{y}$ could span over $\mathbb{R}^{k+2 \times 1}$. Then $\exists \boldsymbol{y}'$ outside the column space of $\boldsymbol{X}$ such that $\boldsymbol{X}\boldsymbol{w}' \neq \boldsymbol{y}' \; \forall \boldsymbol{w}' \in \mathbb{R}^{k+1 \times 1}$. Therefore, the $\mathcal{F}_{\text{affine}})$ is not capable of shattering all $k + 2$ points.
Therefore, $VC(\mathcal{F}_{\text{affine}}) = k + 1$.

iii. The VC dimision of the axis-aligned decision trees is infinite, because decision tree could stop when every leaf is pure, which means no matter how many points there are, the decision tree could grow to an extent where every sample point is perfectly predicted.

(b) (i.) Proof:

$\frac{1}{n}\sum_{i=1}^{n}\epsilon_i f(\boldsymbol{x}^{(i)}) = \frac{1}{n}\sum_{i=1}^{n}\epsilon_i \boldsymbol{w}^\top \boldsymbol{x}^{(i)}$

$= \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{k}\epsilon_i w_j x_{i,j} = \sum_{j=1}^{k}(w_j \frac{1}{n}\sum_{i=1}^{n}\epsilon_i x_{i,j})$

By Cauchy-Schwarz inequality,

$\frac{1}{n}\sum_{i=1}^{n}\epsilon_i f(\boldsymbol{x}^{(i)}) \leq \sqrt{\sum_{j=1}^{k} w_j^2}\sqrt{\sum_{j=1}^{k}(\frac{1}{n}\sum_{i=1}^{n}\epsilon_i x_{i,j})^2}$

$= \|\boldsymbol{w}\|_2 \|\boldsymbol{x}_\epsilon\|_2 \leq W\|\boldsymbol{x}_\epsilon\|_2$

We have shown that for any $f \in \mathcal{F}$, it is true that $\frac{1}{n}\sum_{i=1}^{n}\epsilon_i f(\boldsymbol{x}^{(i)}) \leq W\|\boldsymbol{x}_\epsilon\|_2$. This also holds for $\max_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^{n}\epsilon_i f(\boldsymbol{x}^{(i)})$.

(ii.) Proof:

$\|\boldsymbol{x}_\epsilon\|^2 = \sum_{j=1}^{k}(\frac{1}{n}\sum_{i=1}^{n} x_{i,j}\epsilon_i)^2$

$= \frac{1}{n^2}\sum_{j=1}^{k}(\sum_{i=1}^{n} x_{i,j}\epsilon_i)^2$

$= \frac{1}{n^2}\sum_{j=1}^{k}(\sum_{i=1}^{n}(x_{i,j}\epsilon_i)^2 + \sum_{i,k,i\neq k}^{n} x_{i,j}x_{k,j}\epsilon_i\epsilon_k)$

$\mathbb{E}[\|\boldsymbol{x}_\epsilon\|^2] = \mathbb{E}[\frac{1}{n^2}\sum_{j=1}^{k}\sum_{i=1}^{n}(x_{i,j}\epsilon_i)^2] + \mathbb{E}[\frac{1}{n^2}\sum_{j=1}^{k}\sum_{i,k,i\neq k}^{n} x_{i,j}x_{k,j}\epsilon_i\epsilon_k]$

$= \frac{1}{n^2}\sum_{j=1}^{k}\sum_{i=1}^{n}(x_{i,j})^2\mathbb{E}[\epsilon_i^2] + \frac{1}{n^2}\sum_{j=1}^{k}\sum_{i,k,i\neq k}^{n} x_{i,j}x_{k,j}\mathbb{E}[\epsilon_i\epsilon_k]$

$\mathbb{E}[\epsilon_i^2] = \frac{1}{2}\times 1^2 + \frac{1}{2}\times(-1)^2 = 1$

$\mathbb{E}[\epsilon_i\epsilon_k] = \frac{1}{4}(1)\times 1 + \frac{1}{4}(-1)\times(-1) + \frac{1}{4}(1)\times(-1+)\frac{1}{4}(-1)\times 1 = 0$

$\mathbb{E}[\|\boldsymbol{x}_\epsilon\|^2] = \frac{1}{n^2}\sum_{j=1}^{k}\sum_{i=1}^{n}(x_{i,j})^2 = \frac{1}{n^2}\sum_{i=1}^{n}\|\boldsymbol{x}^{(i)}\|^2 \leq \frac{1}{n^2}\sum_{i=1}^{n}R^2 = \frac{R^2}{n}$

Therefore, $\mathbb{E}[\|\boldsymbol{x}_\epsilon\|^2] \leq \frac{R^2}{n}$.

(iii.) By conclusions from i. and ii., $\mathrm{Rad}(\mathcal{F}) \leq \mathbb{E}[W\|\boldsymbol{x}_\epsilon\|^2] = W\mathbb{E}[\|\boldsymbol{x}_\epsilon\|_2]$.

By Jensen's Inequality, $\mathbb{E}[\|\boldsymbol{x}_\epsilon\|_2] \leq \sqrt{\mathbb{E}[\|\boldsymbol{x}_\epsilon\|^2]}$. So

$\mathrm{Rad}(\mathcal{F}) \leq W\sqrt{\mathbb{E}[\|\boldsymbol{x}_\epsilon\|^2]} \leq W\sqrt{\frac{R^2}{n}} = \frac{RW}{\sqrt{n}}$.