CS 189 Introduction to Machine Learning
Spring 2023 Jonathan Shewchuk

DIS13

1 VC Dimension of Half-Spaces

In lecture, we discussed how the VC dimension of linear classifiers in a plane is 3. In this problem, we will shorten our set of hypotheses to only be those linear classifiers that pass through the origin, and provide a rigorous proof of how the VC dimension of such classifiers in $\mathbb{R}^d$ is d.

We will define a linear classifier as a function $f_w \to \{-1, 1\}$ such that

$$f_w(x) = \text{sign}\left(w^T x\right)$$

and the class of half-spaces of dimension d to be defined as

$$\mathcal{H}_d = \{f_w \ : \ w \in \mathbb{R}^d\}$$

(a) Show that there exists a set of points $S \in \mathbb{R}^d$ of size d such that $\mathcal{H}_d$ shatters S.

(b) We now define a set $T = \{x^{(1)} \dots x^{(d+1)}\}$ of size $d + 1$. Find two sets $I, J \subseteq T$ where $I \cap J = \emptyset$ and with *at least* one of I or J nonempty, as well as positive coefficients $a_1 \dots a_{d+1}$ such that

$$\sum_{i \in I} a_i x^{(i)} = \sum_{j \in J} a_j x^{(j)}$$

(c) Using the last two parts, prove that $\text{VC}(\mathcal{H}_d) = d$

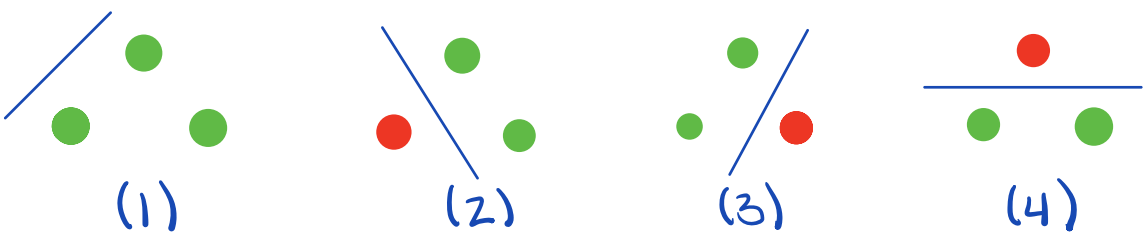
Hint: use a proof by contradiction by assuming that $\mathcal{H}_d$ shatters T and use the linearity of the inner product

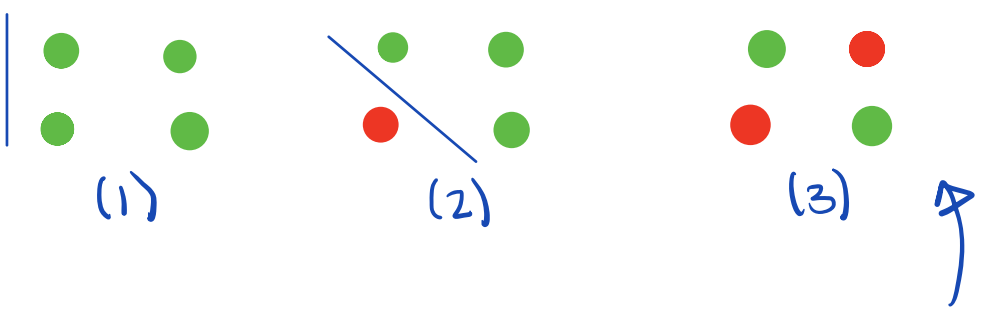
DIS13, ©UCB CS 189, Spring 2023. All Rights Reserved. This may not be publicly shared without explicit permission.

VC dimension - cardinality of the largest set of pts an algorithm can shatter for ≥ 1 configuration of those data pts

shatter - an alg. shatters a set if it can perfectly classify the pts for any labeling

e.g. linear classifier in 2D plane





A linear classifer CANNOT separate this labeling of four pts.

VC dimension = 3
Proof: Radon's theorem

We will define a linear classifier as a function $f_w \to \{-1, 1\}$ such that

$$f_w(x) = \text{sign}\left(w^T x\right)$$

and the class of half-spaces of dimension d to be defined as

$$\mathcal{H}_d = \{f_w \ : \ w \in \mathbb{R}^d\}$$

(a) Show that there exists a set of points $S \in \mathbb{R}^d$ of size d such that $\mathcal{H}_d$ shatters S.

Consider: $S = \{e_1, e_2, \ldots, e_d\}$ ← standard basis vectors

NTS $\exists w \in \mathbb{R}^d$ s.t. $f_w(x)$ corresponds to the correct label $\forall x \in S$ w/ any labeling

Let $y \in \{-1, 1\}^d$ be some label corresponding to the set of data pts S

we want to find w s.t. $\text{sign}(w^T e_i) = y_i$ for all $i = 1, \ldots, d$

Notice that we can simply choose $w = y$

we can choose w for any abitrary labeling y, so H_d shatters the set S

(b) We now define a set $T = \{x^{(1)} \dots x^{(d+1)}\}$ of size $d + 1$. Find two sets $I, J \subseteq T$ where $I \cap J = \emptyset$ and with *at least* one of I or J nonempty, as well as positive coefficients $a_1 \dots a_{d+1}$ such that

$$\sum_{i \in I} a_i x^{(i)} = \sum_{j \in J} a_j x^{(j)}$$

Set T contain d+1 d-dimensional vectors. These vectors cannot all be linearly ind., so $\sum_{k=1}^{d+1} c_k x^{(k)} = 0$ for some set of constants that are not all zero.

Consider the sets: $I = \{ i : c_i > 0 \}$
$J = \{ j : c_j < 0 \}$

$$\sum_{i \in I} c_i x^{(i)} + \sum_{j \in J} c_j x^{(j)} = 0$$

$$\sum_{i \in I} c_i x^{(i)} = - \sum_{j \in J} c_j x^{(j)}$$

Let $a_k = \begin{cases} c_i \text{ for } i \in I \\ -c_j \text{ for } j \in J \end{cases}$

$$\sum_{i \in I} a_i x^{(i)} = \sum_{j \in J} a_j x^{(j)}$$

(c) Using the last two parts, prove that $\text{VC}(\mathcal{H}_d) = d$

Hint: use a proof by contradiction by assuming that $\mathcal{H}_d$ shatters T and use the linearity of the inner product

Assume: H_d shatters T

If H_d shatters T, it can correctly classify <u>any</u> labeling of these pts, so we'd be able to find a w s.t.

$$f_w(x^{(i)}) = \text{sign}(w^T x^{(i)}) > 0 \quad \forall i \in I$$
$$f_w(x^{(j)}) = \text{sign}(w^T x^{(j)}) < 0 \quad \forall j \in J$$

From the linearity of the inner product,

$$w \cdot \sum_{i \in I} a_i x^{(i)} = \sum_{i \in I} a_i w^T x^{(i)} > 0$$

$$w \cdot \sum_{j \in J} a_j x^{(j)} = \sum_{j \in J} a_j w^T x^{(j)} < 0$$

However, in the previous part, we showed

$$\sum_{i \in I} a_i x^{(i)} = \sum_{j \in J} a_j x^{(j)}$$

$\therefore$ H_d can't shatter a set of dimension of $d+1$. From part a, it can shatter a set of dim. d, so $VC(H_d) = d$.

2 Boosted Decision Trees

In this problem, we'll develop the key concepts required to understand boosted decision trees using the AdaBoost algorithm. We are given data $D = \{(X_i, y_i)\}_{i=1}^N$, where $X_i \in \mathbb{R}^d$ and $y_i \in C = \{-1, 1\}$. Recall that, for a node in a decision tree with data $S \subseteq D$, we compute the proportion of each label, then use those proportions to compute the entropy:

$$p_c = \frac{1}{|S|} \sum_{(X,y)\in S} I(y_i = c),$$

$$H(S) = \sum_{c\in C} -p_c \ln p_c.$$

(a) Let w_i be the **weight** of observation (X_i, y_i). The weight of an observation can be thought of as its importance. To incorporate weights into our decision tree, we redefine the way we compute proportions. Let $Z = \sum_{i=1}^{|S|} w_i$, and

$$p_c = \frac{1}{Z} \sum_{i=1}^{|S|} I(y_i = c)\, w_i.$$

Assume $w_i = a$ for all i. Show that $H(S)$ does not change for constant values of a.

(b) Like Random Forest, boosting is an ensemble method. We train several decision trees on weighted observations and combine their predictions to construct an overall improved classifier. The Adaboost algorithm starts by training the first decision tree G_1 using observation weights initialized to $w_i = \frac{1}{|S|}$. The weighted error rate of a trained tree G_t is given by

$$\text{err}_t = \frac{\sum_{i=1}^{|S|} w_i I(y_i \neq G_t(X_i))}{\sum_{i=1}^{|S|} w_i}.$$

Each tree G_t in a boosted ensemble is assigned a weight. The weight is computed using the negative logit function (you will show this is optimal in the lecture on Boosting)

$$\beta_t = \frac{1}{2} \ln\left(\frac{1 - \text{err}_t}{\text{err}_t}\right).$$

What are the minimum and maximum possible values, $\text{err}_{\min}$ and $\text{err}_{\max}$, of err_t so that $\text{err}_{\min} \leq \text{err}_t \leq \text{err}_{\max}$?

(c) After training T decision trees, the AdaBoost algorithm produces the following decision function

$$G(x) = \text{sign}\left[\sum_{t=1}^{T} \beta_t G_t(x)\right],$$

where $\text{sign}[x] = 1$ if $x \geq 0$, and -1 otherwise. Compute $\nabla_{\text{err}_t} \beta_t$. What do you notice about the rate of change of β_t when err_t is near its bounds?

(d) After each decision tree is trained, the weight for each observation $i = 1, ..., |S|$ is updated by the following update rule:

$$w_i \leftarrow w_i \exp\left(-\beta_t y_i G_t(x_i)\right).$$

Note that $-y_i G_t(x_i) = 2I\left(y_i \neq G_t x_i\right) - 1$. Since the -1 becomes a multiplicative constant, we can drop it to obtain

$$w_i \leftarrow w_i \exp\left(2\beta_t I\left(y_i \neq G_t x_i\right)\right).$$

The subsequent tree is trained on these updated weights.

(a) What is the value of $w_j^{(2)}$ if observation j is the only observation that has not been classified correctly after 1 iteration?

(b) How does this influence the optimal split choice for nodes in decision tree G_2?

(c) What is $w_j^{(3)}$ if j is still the only observation which has not been classified correctly?

(a) Let w_i be the **weight** of observation (X_i, y_i). The weight of an observation can be thought of as its importance. To incorporate weights into our decision tree, we redefine the way we compute proportions. Let $Z = \sum_{i=1}^{|S|} w_i$, and

$$p_c = \frac{1}{Z} \sum_{i=1}^{|S|} I(y_i = c)\, w_i.$$

Assume $w_i = a$ for all i. Show that $H(S)$ does not change for constant values of a.

Originally, we had

$$H(S) = \sum_{c \in C} -p_c \ln p_c \text{, where}$$

$$p_c = \frac{1}{|S|} \sum_{(x,y) \in S} I(y_i = c)$$

Now we have the same expression for entropy but define p_c' s.t.

$$p_c' = \frac{1}{Z} \sum_{(x,y) \in S} w_i\, I(y_i = c) \text{, where}$$

$$Z = \sum_{i=1}^{|S|} w_i$$

If $w_i = a$ for all i, then

$$Z = \sum_{i=1}^{|S|} a = |S|a$$

$$p_c' = \frac{1}{|S|a} \sum_{(x,y) \in S} a\, I(y_i = c)$$

$$p_c' = \frac{1}{|S|a} \cdot a \sum_{(x,y) \in S} I(y_i = c)$$

$$= \frac{1}{|S|} \sum_{(x,y) \in S} I(y_i = c) = p_c$$

$\therefore$ $H(S)$ does not change if $w_i = a \ \forall i$ for some constant value of a

(b) Like Random Forest, boosting is an ensemble method. We train several decision trees on weighted observations and combine their predictions to construct an overall improved classifier. The Adaboost algorithm starts by training the first decision tree G_1 using observation weights initialized to $w_i = \frac{1}{|S|}$. The weighted error rate of a trained tree G_t is given by

$$\text{err}_t = \frac{\sum_{i=1}^{|S|} w_i I\left(y_i \neq G_t(X_i)\right)}{\sum_{i=1}^{|S|} w_i}.$$

Each tree G_t in a boosted ensemble is assigned a weight. The weight is computed using the negative logit function (you will show this is optimal in the lecture on Boosting)

$$\beta_t = \frac{1}{2} \ln\left(\frac{1 - \text{err}_t}{\text{err}_t}\right).$$

What are the minimum and maximum possible values, err_{min} and err_{max}, of err_t so that $\text{err}_{\text{min}} \leq \text{err}_t \leq \text{err}_{\text{max}}$?

$$err_t = \frac{1}{Z} \sum_{i=1}^{|S|} w_i I(y_i \neq G_t(X_i))$$

$$I(y_i \neq G_t(X_i)) = \begin{cases} 0 & \text{if } y_i = G_t(X_i) \\ 1 & \text{if } y_i \neq G_t(X_i) \end{cases}$$

err_t is minimized if $y_i = G_t(X_i)\ \forall i \Rightarrow$

$$err_{min} = \frac{1}{Z} \sum_{i=1}^{|S|} w_i(0) = 0$$

└ G_t classifies all examples correctly

err_t is maximized if $y_i \neq G_t(X_i)\ \forall i \Rightarrow$

$$err_{max} = \frac{1}{Z} \sum_{i=1}^{|S|} w_i(1) = \frac{1}{Z} \cdot Z = 1$$

└ G_t classifies all incorrectly

(c) After training T decision trees, the AdaBoost algorithm produces the following decision function

$$G(x) = \text{sign}\left[\sum_{t=1}^{T} \beta_t G_t(x)\right],$$

where $\text{sign}[x] = 1$ if $x \geq 0$, and -1 otherwise. Compute $\nabla_{\text{err}_t}\beta_t$. What do you notice about the rate of change of β_t when err_t is near its bounds?

$$\beta_t = \frac{1}{2}\ln\left(\frac{1-\text{err}_t}{\text{err}_t}\right)$$

$$\nabla_{\text{err}_t}\beta_t = \frac{1}{2}\left(\frac{1-\text{err}_t}{\text{err}_t}\right)^{-1}\left(\frac{\text{err}_t(-1) - (1-\text{err}_t)(1)}{\text{err}_t^2}\right)$$

$$= \frac{1}{2}\left(\frac{\text{err}_t}{1-\text{err}_t}\right)\left(\frac{-1}{\text{err}_t^2}\right)$$

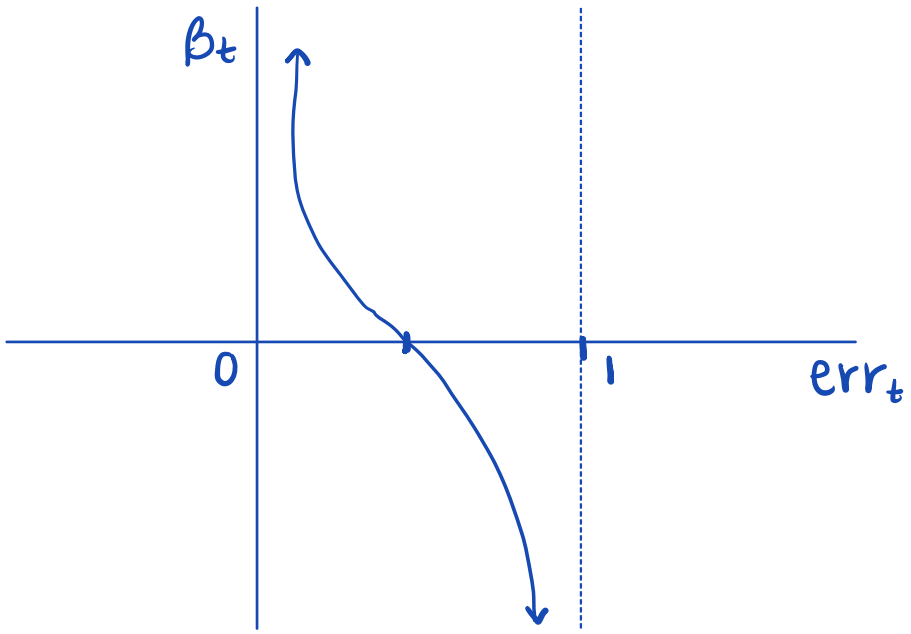
$$= -\frac{1}{2\text{err}_t(1-\text{err}_t)}$$

$$\text{err}_{\min} = 0 \rightarrow \lim_{\text{err}_t \to 0} \nabla_{\text{err}_t}\beta_t = -\infty$$

$$\text{err}_{\max} = 1 \rightarrow \lim_{\text{err}_t \to 1} \nabla_{\text{err}_t}\beta_t = -\infty$$

The rate of change of β_t blows up when err_t approaches its bounds.

The plot of β_t looks like the following:



The weight on a classifier quickly becomes very positive if its error is close to the min. value (0) & quickly becomes very negative if its error is close to the max. value (1).

The weight is approx. 0 for classifiers w/ error rates $\sim \frac{1}{2}$ b/c they have little predictive power.

(d) After each decision tree is trained, the weight for each observation $i = 1, ..., |S|$ is updated by the following update rule:

$$w_i \leftarrow w_i \exp\left(-\beta_t y_i G_t(x_i)\right).$$

Note that $-y_i G_t(x_i) = 2I\left(y_i \neq G_t x_i\right) - 1$. Since the -1 becomes a multiplicative constant, we can drop it to obtain

$$w_i \leftarrow w_i \exp\left(2\beta_t I\left(y_i \neq G_t x_i\right)\right).$$

The subsequent tree is trained on these updated weights.

(a) What is the value of $w_j^{(2)}$ if observation j is the only observation that has not been classified correctly after 1 iteration?

(b) How does this influence the optimal split choice for nodes in decision tree G_2?

(c) What is $w_j^{(3)}$ if j is still the only observation which has not been classified correctly?

a) $$err_1 = \frac{\sum_{i=1}^{|S|} w_i^{(1)} I(y_i \neq G_1(X_i))}{\sum_{i=1}^{|S|} w_i^{(1)}}$$

If j is the only misclassified example,

$$I(y_i \neq G_1(X_i)) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{o.w.} \end{cases}$$

$$err_1 = \frac{w_j^{(1)}}{\sum_{i=1}^{|S|} w_i^{(1)}}$$

We initialize our weights s.t. $w_i^{(1)} = \frac{1}{|S|}$ for all i. Therefore,

$$err_1 = \frac{1/|S|}{\sum_{i=1}^{|S|} \frac{1}{|S|}} = \frac{1/|S|}{|S|\,1/|S|} = \frac{1}{|S|}$$

$$\beta_1 = \frac{1}{2}\ln\left(\frac{1-err_1}{err_1}\right) = \frac{1}{2}\ln\left(\frac{1-\frac{1}{|S|}}{1/|S|}\right)$$
$$= \frac{1}{2}\ln(|S|-1)$$

For $i \neq j$,

$$w_i^{(2)} = w_i^{(1)} \exp\left[2\beta_1 I(y_i \neq G_t(x_i))\right]$$
$$= w_i^{(1)} \exp[2\beta_1 \cdot 0] = w_i^{(1)} = 1/|S|$$

$$w_j^{(2)} = w_j^{(1)} \exp\left[2\beta_1 I(y_i \neq G_t(x_i))\right]$$
$$= w_j^{(1)} \exp[2\beta_1 \cdot 1]$$
$$= 1/|S| \exp\left[2 \cdot \tfrac{1}{2}\ln(|S|-1)\right]$$
$$= 1/|S|(|S|-1)$$

b) Recall that we define entropy s.t.

$H(S) = -p_{-1}\ln p_{-1} - p_1 \ln p_1$, where

$p_{-1} = \sum_{i:y_i=-1} w_i$

$p_1 = \sum_{i:y_i=1} w_i$

Now the j^{th} example has a much greater effect on the entropy than any other example. The optimal split choice will now depend more on this example.

c) $$err_2 = \frac{w_j^{(2)}}{\sum_{i=1}^{|S|} w_i^{(2)}} = \frac{w_j^{(2)}}{w_j^{(2)} + \sum_{i \neq j} w_i^{(2)}}$$

$$= \frac{\frac{1}{|S|}(|S|-1)}{\frac{1}{|S|}(|S|-1) + (|S|-1)\frac{1}{|S|}} = \frac{1}{2}$$

$$\beta_2 = \frac{1}{2} \ln\left(\frac{1 - err_2}{err_2}\right) = \frac{1}{2}\ln(1) = 0$$

For $i \neq j$,

$$w_i^{(3)} = w_i^{(2)} \exp\left[2\beta_2 I(y_i \neq G_t(x_i))\right]$$
$$= w_i^{(2)} \exp[2\beta_2 \cdot 0] = w_i^{(2)} = \frac{1}{|S|}$$

$$w_j^{(3)} = w_j^{(2)} \exp\left[2\beta_2 I(y_i \neq G_t(x_i))\right]$$
$$= w_j^{(2)} \exp(0) = w_j^{(2)} = 1/|S|(|S|-1)$$

3 Cal vs. Stanford Decision Stumps

Recall that **ensembling** is the practice of training several models to perform the same task. A random forest is an example of an ensemble, particularly of decision trees. Also recall that **boosting** is the practice of iteratively training decision trees that learn from the errors of the previous trees. **AdaBoost** is one such boosting algorithm.

In this example, we deal with decision stumps, which are one-level decision trees. AdaBoost is often used to ensemble decision stumps, which we will explore in this problem.

Recall that in AdaBoost, our input is an $n \times d$ design matrix X with n labels $y_i = \pm 1$, and at the end of iteration T the importance of each sample is reweighted as

$$w_i^{(T+1)} = w_i^{(T)} \exp(-\beta_T \, y_i \, G_T(X_i)), \quad \text{where} \quad \beta_T = \frac{1}{2} \ln\left(\frac{1 - \text{err}_T}{\text{err}_T}\right) \quad \text{and} \quad \text{err}_T = \frac{\sum_{y_i \neq G_T(X_i)} w_i^{(T)}}{\sum_{i=1}^{n} w_i^{(T)}}.$$

Note that err_T is the weighted error rate of the classifier G_T. Recall that $G_T(z)$ is ± 1 for all points z, but the metalearner has a non-binary decision function $M(z) = \sum_{t=1}^{T} \beta_t G_t(z)$. To classify a test point z, we calculate $M(z)$ and return its sign.

We went to use AdaBoost to train an ensemble of decision stumps to classify whether a student goes to Stanford ~~and~~ GV Cal, based on their fitness activity. Our training data is shown below, where +1 corresponds to Cal and -1 corresponds to Stanford:

Bike Miles Driven	Elevation Climbed	Cal or Stanford
2	1	+1
4	2	-1
5	5	+1

(a) Write out the decision function for $G_t(X_i)$, which is decision stump t classifying a point X_i. Assume that for this decision stump, the entropy-minimizing feature is feature j, so we use only the scalar value X_{ij} in classification. Assume also that for this entropy-minimizing feature, we split on the threshold α_t.

(b) Assume that decision stump G_i has $\texttt{err}_i \geq 0.5$. What is the range of the resultant value of β_i? What is the significance of this range for a classified test point?

(c) Given the dataset, find decision stump G_1, model error $\texttt{err}_1$, and model weight β_1. Assume that the weights are initialized to $w_1^{(0)} = w_2^{(0)} = w_3^{(0)} = \frac{1}{3}$. (Hint: many thresholds will achieve the lowest loss, so let's use Bike Miles Driven > 3 for the decision stump.)

(d) Compute the weights of the points after the creation of decision stump G_1. In other words, compute $w_1^{(1)}$, $w_2^{(1)}$, and $w_3^{(1)}$.

(e) Find decision stump G_2, model error $\texttt{err}_2$, and model weight β_2. (Hint: many thresholds will achieve the lowest loss, so let's use Elevation Climbed > 3.5 for the decision stump.)

(f) Compute the weights of the points after the creation of decision stump G_2. In other words, compute $w_1^{(2)}$, $w_2^{(2)}$, and $w_3^{(2)}$. Which point do you think decision stump G_3 will prioritize for correct classification?

(g) We halt training after two iterations, i.e. our forest has 2 decision stumps. Classify the following test point: (7, 4).

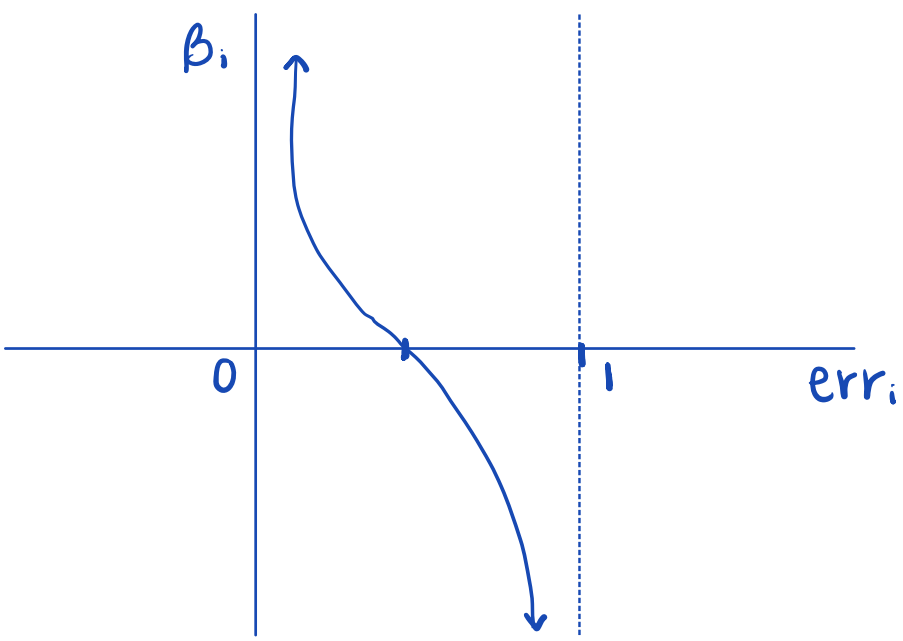
(a) Write out the decision function for $G_t(X_i)$, which is decision stump t classifying a point X_i. Assume that for this decision stump, the entropy-minimizing feature is feature j, so we use only the scalar value X_{ij} in classification. Assume also that for this entropy-minimizing feature, we split on the threshold α_t.

$$G_t(X_i) = \begin{cases} 1 & \text{if } X_{ij} > \alpha_t \\ -1 & \text{if } X_{ij} < \alpha_t \end{cases}$$

$$G_t(X_i) = \text{sign}(X_{ij} - \alpha_t)$$

(b) Assume that decision stump G_i has $\texttt{err}_i \geq 0.5$. What is the range of the resultant value of β_i? What is the significance of this range for a classified test point?

$$\beta_i = \frac{1}{2} \ln\left(\frac{1 - \text{err}_i}{\text{err}_i}\right)$$



If $err_i \geq 0.5$, then $\beta_i \in (-\infty, 0]$. Therefore, the model weight for this classifier is negative. This classifier is more often misclassifying points, so the metalearner inverts its predictions.

(c) Given the dataset, find decision stump G_1, model error $\texttt{err}_1$, and model weight β_1. Assume that the weights are initialized to $w_1^{(0)} = w_2^{(0)} = w_3^{(0)} = \frac{1}{3}$. (Hint: many thresholds will achieve the lowest loss, so let's use Bike Miles Driven < 3 for the decision stump.)

Bike Miles Driven	Elevation Climbed	Cal or Stanford
2	1	+1
4	2	-1
5	5	+1

$$G_1(X_i) = \begin{cases} +1 \text{ (Cal)} & \text{if Bike Miles Driven} < 3 \\ -1 \text{ (Stan.)} & \text{if Bike Miles Driven} > 3 \end{cases}$$

$$G_1(X) = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} \qquad y = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}$$

$$err_1 = \frac{w_3^{(0)}}{\sum_{i=1}^{3} w_i^{(0)}} = \frac{1/3}{3(1/3)} = \frac{1}{3}$$

$$\beta_1 = \frac{1}{2} \ln\left(\frac{1 - err_1}{err_1}\right) = \frac{1}{2}\ln(2) \approx 0.347$$

(d) Compute the weights of the points after the creation of decision stump G_1. In other words, compute $w_1^{(1)}$, $w_2^{(1)}$, and $w_3^{(1)}$.

$$w_i^{(1)} = w_i^{(0)} \exp(-\beta_1 y_i G_1(X_i))$$

$$w_1^{(1)} = w_1^{(0)} \exp(-\beta_1) = \frac{1}{3} e^{-0.347} \approx 0.236$$
$$w_2^{(1)} = w_2^{(0)} \exp(-\beta_1) = \frac{1}{3} e^{-0.347} \approx 0.236$$
$$w_3^{(1)} = w_3^{(0)} \exp(\beta_1) = \frac{1}{3} e^{0.347} \approx 0.471$$

(e) Find decision stump G_2, model error $\texttt{err}_2$, and model weight β_2. (Hint: many thresholds will achieve the lowest loss, so let's use Elevation Climbed > 3.5 for the decision stump.)

Bike Miles Driven	Elevation Climbed	Cal or Stanford
2	1	+1
4	2	-1
5	5	+1

$$G_2(X_i) = \begin{cases} +1 \text{ (Cal)} & \text{if Elevation Climbed} > 3.5 \\ -1 \text{ (Stan.)} & \text{if Elevation Climbed} < 3.5 \end{cases}$$

$$G_2(X) = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} \qquad y = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}$$

$$\text{err}_2 = \frac{w_i^{(1)}}{\sum_{i=1}^{3} w_i^{(1)}} = \frac{0.236}{0.236 + 0.236 + 0.471} = \frac{1}{4}$$

$$\beta_2 = \frac{1}{2} \ln\left(\frac{1 - \text{err}_2}{\text{err}_2}\right) = \frac{1}{2} \ln(3) \approx 0.549$$

(f) Compute the weights of the points after the creation of decision stump G_2. In other words, compute $w_1^{(2)}$, $w_2^{(2)}$, and $w_3^{(2)}$. Which point do you think decision stump G_3 will prioritize for correct classification?

$$w_i^{(2)} = w_i^{(1)} \exp(-\beta_2 y_i G_1(x_i))$$

$$w_1^{(2)} = w_1^{(1)} \exp(\beta_2) = 0.236\, e^{0.549} \approx 0.409$$
$$w_2^{(2)} = w_2^{(1)} \exp(-\beta_2) = 0.236\, e^{-0.549} \approx 0.136$$
$$w_3^{(2)} = w_3^{(1)} \exp(-\beta_2) = 0.471\, e^{-0.549} \approx 0.272$$

G_3 will prioritize correctly classifying the first data pt (w/ the largest weight).

(g) We halt training after two iterations, i.e. our forest has 2 decision stumps. Classify the following test point: (7, 4).

$$G_1(z) = \begin{cases} +1 \text{ (Cal)} & \text{if Bike Miles Driven} < 3 \\ -1 \text{ (Stan.)} & \text{if Bike Miles Driven} > 3 \end{cases}$$

$$G_2(z) = \begin{cases} +1 \text{ (Cal)} & \text{if Elevation Climbed} > 3.5 \\ -1 \text{ (Stan.)} & \text{if Elevation Climbed} < 3.5 \end{cases}$$

$G_1(7,4) = -1$
$G_2(7,4) = 1$

$$M(z) = \text{sign}(\beta_1 G_1(z) + \beta_2 G_2(z))$$
$$= \text{sign}(0.347(-1) + 0.549(1))$$
$$= \text{sign}(0.202) = 1$$