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.

Solution: We will consider the set $S = \{e_1 \dots e_d\}$ consisting of the set of all basis vectors in $\mathbb{R}^d$. We will allow a mapping $S \to \{1, -1\}$ such that each basis vector corresponds to a label. We will show that we can find a vector $w \in \mathbb{R}^d$ to construct a hypothesis for any arbitrary mapping where $f_w \in \mathcal{H}_d$ and it maps S to the correct set of labels $\{-1, 1\}$ provided.

Let us represent $y = \{-1, 1\}^d$ as the provided set of labels corresponding to our set S. We want to find a w such that:

$$\text{sign}(w \cdot e_i) = y_i$$

We can simply set $w[i] = y_i$ and notice that this makes the above equality true. This means that $\mathcal{H}_d$ shatters the set S since we can choose a different w to satisfy any one of the possibilities of y, of which there are 2^d different combinations. Thus, $\left|\mathcal{H}_{|S}\right| = |y| = 2^{|S|}$.

(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

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

Solution: Since there are $d + 1$ vectors in T, then that means that at least one of the vectors are not linearly independent in T. This means that for not all $c_i = 0$, we have:

$$\sum_{k=1}^{d+1} c_k x^{(k)} = 0$$

We can then construct two different sets: $I = \{i \ : \ c_i > 0\}$ and $J = \{i \ : \ c_i < 0\}$. Then we can rearrange the above into:

$$\sum_{k=1}^{d+1} c_k x^{(k)} = 0$$
$$\sum_{i \in I} c_i x^{(i)} + \sum_{j \in I} c_i x^{(i)} = 0$$
$$\sum_{i \in I} c_i x^{(i)} = \sum_{j \in J} -c_j x^{(j)}$$

This means that we can set $a_k = c_i$ for $i \in I$ and $a_k = -c_j$ for $j \in J$ and all these coefficients are guaranteed to be positive.

Note that it is never the case that both of these sets are empty, because that would mean that all $c_k = 0$ and the $d + 1$ vectors in T are linearly independent, which is a contradiction.

This proof is similar to the proof of **Radon's Theorem**, which was mentioned in lecture, except Radon's Theorem handles the more general case of hyperplanes that do not necessarily pass through the origin. The core idea of linear combinations with nonnegative coefficients is still there, but this case is a bit simpler.

(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

Solution: For contradiction, assume that $\mathcal{H}$ shatters T. From the previous part, we showed that there exist some constants $a_i \ldots a_n > 0$ such that

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

Since $\mathcal{H}$ shatters T, we know that it is possible to find a w such that

$$w \cdot x^{(i)} > 0 \quad \forall i \in I \qquad w \cdot x^{(j)} < 0 \quad \forall j \in J$$

Given that none of the sets are non-empty, we can use the linearity of the inner product to write:

$$w \cdot \sum_{i \in I} a_i x^{(i)} = \sum_{i \in I} a_i \left(w \cdot x_i\right) > 0$$
$$w \cdot \sum_{j \in J} a_j x^{(j)} = \sum_{j \in J} a_j \left(w \cdot x_j\right) < 0$$

This is a contradiction since

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

Note that even if one of the sets are nonempty, then the corresponding inequality > 0 or < 0 would become an equality, from which the contradiction still holds. This means that $\mathcal{H}$ does not shatter T, which implies that $\mathcal{H}_d$ cannot shatter a set of dimension $d+1$ so the VC dimension of $\mathcal{H}_d$ is 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.

Solution: We have

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

(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 $\text{err}_{\min} \leq \text{err}_t \leq \text{err}_{\max}$?

Solution: The minimum of err_t is attained when G_t classifies all examples correctly, and maximum when G_t classifies all examples incorrectly, so $0 \leq \text{err}_t \leq 1$.

(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?

Solution: $\nabla_{\text{err}_t} \beta_t = \frac{1}{2\,\text{err}_t(1-\text{err}_t)}$. When err_t is near its bounds, the rate of change blows up. This means that really good classifiers are given a lot of weight, and really bad ones are given very little weight.

Fun fact: The logistic function is the inverse-logit function, and $\phi(x) = \frac{1}{1+e^x}$ is the inverse-negative-logit function, so $\phi(\beta_t) = \text{err}_t$.

(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?

Solution:

(a) Since this is the first iteration, and j is the only observation that is incorrectly classified, we can simplify $\text{err}_1 = \left(\frac{w_j^1}{\sum_i^{|S|} w_i^{(1)}}\right) = w_j^{(1)} = 1/|S|$. After 1 iteration, $w_j^{(2)} \leftarrow w_j^{(1)}(1 - w_j^{(1)})/w_j^{(1)} = (1 - 1/|S|) = (|S| - 1)/|S|$. Note that for $i \neq j$, $w_i^{(2)} \leftarrow w_i^{(1)}$.

(b) Because $w_j^{(2)}$ is $|S| - 1$ times larger than $w_j^{(1)}$, the "proportion" p_{y_j} will be much larger in iteration 2. Separating y_j from the other class will result in a significantly greater drop in entropy than any other observation that has the same class as y_j. You can think of separating y_j as isolating $N - 1$ labels of the same class.

(c) $\text{err}_2 = \frac{N-1}{N-1+N-1} = \frac{1}{2}, \beta_2 = \frac{1}{2}\ln\left(\frac{1/2}{1/2}\right) = 0$, so $w_j^{(3)} = w_j^{(2)}$.

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 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.

Solution: A decision stump essentially splits $\mathbb{R}^d$ into two axis-aligned regions that correspond to each class (+1 or -1). Then, our classifier is:

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

(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?

Solution: If our error is greater than 50%, then the range of β_i is $(-\infty, 0]$. This means that the model weight for decision stump G_i is negative.

The metalearner in AdaBoost can be thought of doing the following during classification. We add up the model weights for all the individual decision stumps that classify a test point as +1,

and do the same thing for stumps that classify as -1. If the summed model weights for +1 are greater than the summed model weights for -1, then the metalearner classifies the test point as +1, and vice versa.

Thus, when a stump has negative model weight, it is acting adversarially: it is consistently classifying points as the inverse of what they should be. Thus, we can invert the classification of this adversarial stump, which means that its model weight should go the opposite class. I.e. if the stump has model weight β and adversarially classifies a point as +1, then we add β to the summed model weights for -1.

This is equivalent to leaving the summed model weights for -1 the same and subtracting β from the summed model weights for +1.

(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.)

Solution: If Bike Miles Driven > 3, we classify the point as -1 (Stanford), otherwise +1 (Cal). This leads to points 1 and 2 being correctly classified and point 3 being misclassified.

This means that $\texttt{err}_1 = \frac{1}{3}$ and $\beta_1 = \frac{1}{2}\ln\left(\frac{1-\texttt{err}_1}{\texttt{err}_1}\right) \approx 0.347$.

(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 $w_3^{(1)}$.

Solution: Using the weight update formula provided in the problem statement above, we have:

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

(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.)

Solution: If Elevation Climbed > 3.5, we classify the point as +1 (Cal), otherwise -1 (Stanford). This leads to points 2 and 3 being correctly classified and point 1 being misclassified.

This means that $\texttt{err}_2 = \frac{0.236}{0.236+0.236+0.471} = \frac{1}{4}$ and $\beta_2 = \frac{1}{2}\ln\left(\frac{1-\texttt{err}_2}{\texttt{err}_2}\right) \approx 0.549$.

(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?

Solution: Using the weight update formula provided in the problem statement above, we have:

$$w_1^{(2)} = 0.236e^{0.549} = 0.409$$

$$w_2^{(2)} = 0.236e^{-0.549} = 0.136$$
$$w_3^{(2)} = 0.471e^{-0.549} = 0.272$$

G_3 will likely prioritize getting point 1's classification correct.

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

Solution: G_1: Because Bike Miles Driven > 3, we classify as -1.

G_2: Because Elevation Climbed > 3.5, we classify as +1.

$M(x) = \text{sign}(\beta_1 G_1(x) + \beta_2 G_2(x)) = \text{sign}(-0.347 + 0.549) = \text{sign}(0.202) = +1$. The test point is classified as Cal. This makes sense because Stanford's campus is very flat :)