CSCI567: Machine Learning USC, Fall 2022

Homework 1

Instructor: Vatsal Sharan *Due: September 14 by 2:00 pm PST*

Problem 1: Perceptron Convergence (15pts)

Recall the perceptron algorithm that we saw in class. The perceptron algorithm comes with strong theory, and you will explore some of this theory in this problem. We begin with some remarks related to notation which are valid throughout the homework. Unless stated otherwise, scalars are denoted by small letters in normal font, vectors are denoted by small letters in bold font, and matrices are denoted by capital letters in bold font.

Problem 1 asks you to show that when the two classes in a binary classification problem are linearly separable, then the perceptron algorithm will *converge*. For the sake of this problem, we define convergence as predicting the labels of all training instances perfectly. The perceptron algorithm is described in Algorithm 3. It gets access to a dataset of n instances $(\mathbf{x}_i, y_i)$, where $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$. It outputs a linear classifier $\mathbf{w}$.

Algorithm 1 Perceptron

```
while not converged do
    Pick a data point $(\mathbf{x}_i, y_i)$ randomly
    Make a prediction $\hat{y} = \mathrm{sgn}(\mathbf{w}^T \mathbf{x}_i)$ using current $\mathbf{w}$
    if $\hat{y} \neq y_i$ then
        $\mathbf{w} \leftarrow \mathbf{w} + y_i \mathbf{x}_i$
    end if
end while
```

Assume there exists an optimal hyperplane $\mathbf{w}_{\text{opt}}$, $\|\mathbf{w}_{\text{opt}}\|_2 = 1$ and some $\gamma > 0$ such that $y_i(\mathbf{w}_{\text{opt}}^T \mathbf{x}_i) \geq \gamma, \forall i \in \{1, 2, \ldots, n\}$. Additionally, assume $\|\mathbf{x}_i\|_2 \leq R, \forall i \in \{1, 2, ..., n\}$. Following the steps below, show that the perceptron algorithm makes at most $\frac{R^2}{\gamma^2}$ errors, and therefore the algorithm must converge.

1.1 (5pts) Show that if the algorithm makes a mistake, the update rule moves it towards the direction of the optimal weights $\mathbf{w}_{\text{opt}}$. Specifically, denoting explicitly the updating iteration index by k, the current weight vector by $\mathbf{w}_k$, and the updated weight vector by $\mathbf{w}_{k+1}$, show that, if $y_i(\mathbf{w}_k^T \mathbf{x}_i) < 0$, we have

$$\mathbf{w}_{k+1}^T \mathbf{w}_{\text{opt}} \geq \mathbf{w}_k^T \mathbf{w}_{\text{opt}} + \gamma \|\mathbf{w}_{\text{opt}}\|_2. \tag{1}$$

*The geometric meaning of **1.1**: although we cannot say that the update moves $\mathbf{w}$ to exactly the optimal $\mathbf{w}_{opt}$, we can say that the magnitude of the projection of $\mathbf{w}$ on $\mathbf{w}_{opt}$ ($\mathbf{w}_{k+1}^T\mathbf{w}_{opt}$) increases by each round of updating. That is: the component of weights in the optimal direction increases.*

Algorithm 2 Perceptron

```
while not converged do
    Pick a data point $(\mathbf{x}_i, y_i)$ randomly
    Make a prediction $\hat{y} = \mathrm{sgn}(\mathbf{w}^T\mathbf{x}_i)$ using current $\mathbf{w}$
    if $\hat{y} \neq y_i$ then
        $\mathbf{w} \leftarrow \mathbf{w} + y_i\mathbf{x}_i$
    end if
end while
```

Assume there exists an optimal hyperplane $\mathbf{w}_{\text{opt}}, \|\mathbf{w}_{\text{opt}}\|_2 = 1$ and some $\gamma > 0$ such that $y_i(\mathbf{w}_{\text{opt}}^T\mathbf{x}_i) \geq \gamma, \forall i \in \{1, 2, \ldots, n\}$. Additionally, assume $\|\mathbf{x}_i\|_2 \leq R, \forall i \in \{1, 2, ..., n\}$. Following the steps below, show that the perceptron algorithm makes at most $\dfrac{R^2}{\gamma^2}$ errors, and therefore the algorithm must converge.

1.2 (4pts) Show that the length of updated weights does not increase by a large amount. In particular, show that if $y_i(\mathbf{w}_k^T\mathbf{x}_i) < 0$, then

$$\|\mathbf{w}_{k+1}\|_2^2 \leq \|\mathbf{w}_k\|_2^2 + R^2. \tag{2}$$

Algorithm 3 Perceptron

```
while not converged do
    Pick a data point $(\mathbf{x}_i, y_i)$ randomly
    Make a prediction $\hat{y} = \mathrm{sgn}(\mathbf{w}^T \mathbf{x}_i)$ using current $\mathbf{w}$
    if $\hat{y} \neq y_i$ then
        $\mathbf{w} \leftarrow \mathbf{w} + y_i \mathbf{x}_i$
    end if
end while
```

Assume there exists an optimal hyperplane $\mathbf{w}_{\text{opt}}, \|\mathbf{w}_{\text{opt}}\|_2 = 1$ and some $\gamma > 0$ such that $y_i(\mathbf{w}_{\text{opt}}^T \mathbf{x}_i) \geq \gamma, \forall i \in \{1, 2, \ldots, n\}$. Additionally, assume $\|\mathbf{x}_i\|_2 \leq R, \forall i \in \{1, 2, ..., n\}$. Following the steps below, show that the perceptron algorithm makes at most $\dfrac{R^2}{\gamma^2}$ errors, and therefore the algorithm must converge.

1.3 (5pts) Assume that the initial weight vector $\mathbf{w}_0 = \mathbf{0}$ (an all-zero vector). Using results from the previous two parts, show that for any iteration $k+1$, with M being the total number of mistakes the algorithm has made for the first k iterations, we have

$$\gamma M \leq \|\mathbf{w}_{k+1}\|_2 \leq R\sqrt{M}. \tag{3}$$

Hint: use the Cauchy-Schwartz inequality: $\mathbf{a}^T \mathbf{b} \leq \|\mathbf{a}\|_2 \|\mathbf{b}\|_2$.

1.4 (1pts) Use **1.3** to conclude that $M \leq R^2/\gamma^2$. Therefore the algorithm can make at most R^2/γ^2 mistakes (note that there is no direct dependence on the dimensionality of the datapoints).

Problem 2: Logistic Regression (10pts)

This problems explores some properties of the logistic function to get you more comfortable with it. Consider the following univariate function first:

$$F(x; A, k, b) = \frac{A}{1 + e^{-k(x-b)}} \tag{4}$$

2.1 (3pts) Describe with words how A, k and b affect or are related to the shape of $F(x; A, k, b)$ by using the plot at `https://www.geogebra.org/graphing/mxw7wp8b`. Note: We are trying to learn the parameters of a logistic function that can fit the dataset. For example, see the figure below and guess which of the four functions fit well.

Ans. A controls the range/height of the function i.e $F \in (0, A)$.
k controls the steepness of transition from 0 to A. This is most prominent at the point of inflection $(b, A/2)$.
b controls the point of inflection and shifts the graph along the x-axis.

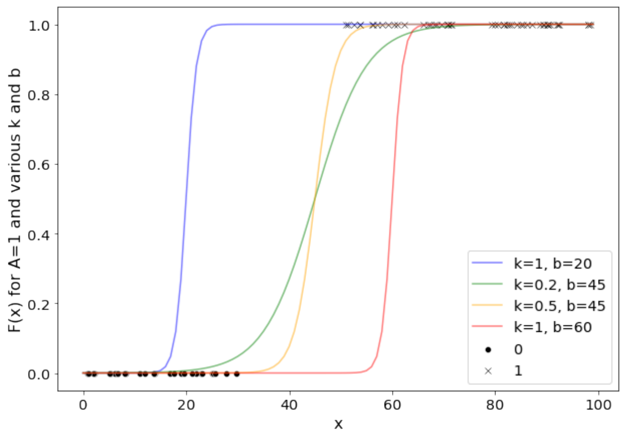


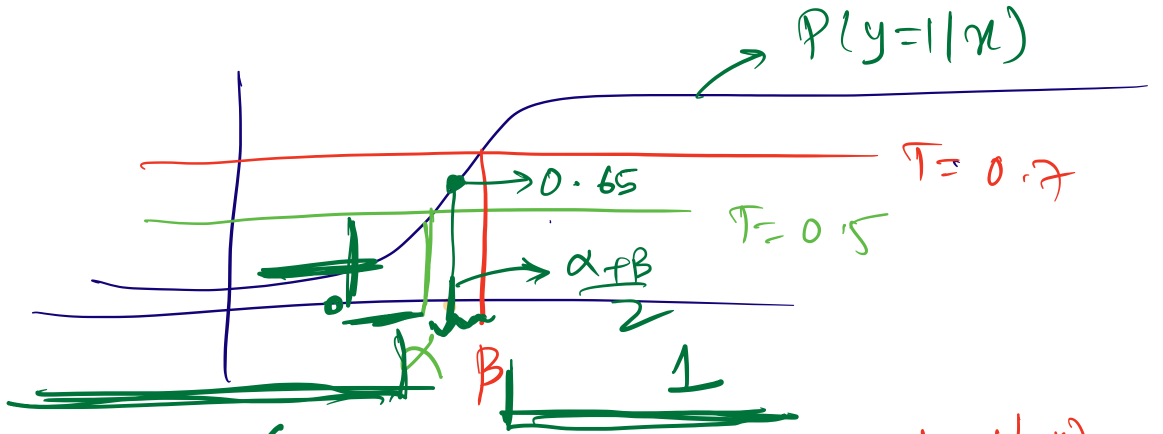
Figure 1: Learning the parameters of logistic function for a dataset.

2.2 (1pt) For what values of A, k and b is $F(x; A, k, b)$ a cumulative distribution function (CDF)?

Ans. For $A = 1$ and $k > 0$.

2.3 (2pts) When $F(x; A, k, b)$ is a CDF, we can interpret $F(x; A, k, b)$ as the probability of x belonging to the class 1 (in a binary classification problem). Suppose we know $F(x; A, k, b)$ and want to predict the label of a datapoint x. We need to decide on a threshold value for $F(x; A, k, b)$, above which we will predict the label 1 and below which we will predict -1. Show that setting the threshold to be $F(x; A, k, b) \geq 1/2$ minimizes the classification error.

The value $x = x_0$ for which $F(x_0) = 0.5$ is called the decision boundary. In the univariate case, it is a point in on the x-axis, in the multivariate case (next question), it is a hyperplane. There is a rich literature on decision theory which studies how to make decisions if we know the probabilities of the underlying events (see https://mlstory.org/pdf/prediction.pdf to learn more if you're interested).



$y=1|x) = F(x) = 0.7$

1 ✓ $P_{T=0.5}(x) = 1 \quad \forall x > \alpha$ ✓ $\quad P(y=0|x) = 1-F(x) = 0.3$

2 ✓ $P_{T=0.7}(x) = 1 \quad \forall x > ~~\alpha~~ \beta$ ✓

$P_{T=0.3}$

$$\hat{y}_1 = P_{T=0.5}\left(\frac{\alpha+\beta}{2}\right) = 1 \rightarrow y \begin{cases} \nearrow 1 & 0.65 \\ \searrow 0 & \to 0.35 \end{cases}$$

$$\hat{y}_2 = P_{T=0.7}\left(\frac{\alpha+\beta}{2}\right) = 0$$

$E(L(\hat{y}_1, y)) = 0.35 \cdot 1 = 0.35$ ✓

$E(L(\hat{y}_2, y)) = 0.65$ ✓

$E(L(\hat{y}, y)) = P(y=0|x)\,\mathbb{I}(F(x) \geq T)$ $\quad T=0.5$

[illegible]

$$+ P(\hat{y}=1|x) I(F(x) \leq T)$$

2.4 (4pts) We now consider the multivariate version. Consider the function:

$$F(\mathbf{x}; \mathbf{w}, b) = \frac{1}{1 + e^{-\mathbf{w}^{\mathbf{T}}\mathbf{x}+b}}. \tag{5}$$

You can explore a 2D version of this function at `https://www.geogebra.org/3d/g9fjtdeh` where $\mathbf{w} = (w_1, w_2)$. Match the items in the left column with those on the right. It will help if you can get the expression for the gradient of $F(\mathbf{x}; \mathbf{w}, b)$ w.r.t $\mathbf{x}$. Provide a short explanation of your answers.

1) level sets of $F(\mathbf{x}; \mathbf{w}, b)$	a) parallel to $\mathbf{w}$
2) direction of gradient at any point	b) $\|\|\mathbf{w}\|\|/4$
3) distance of the level set $F(\mathbf{x}) = 1/2$ from the origin	c) $\|b\|/\|\|\mathbf{w}\|\|$
	d) $\|b\|$
	e) orthogonal to $\mathbf{w}$

Problem 3: Learning rectangles (15pts)

An axis aligned rectangle classifier is a classifier than assigns the value 1 to a point if and only if it is inside a certain rectangle. Formally, given real numbers $a_1 \le b_1, a_2 \le b_2$, define the classifier $f_{(a_1,b_1,a_2,b_2)}$ on an input $\mathbf{x}$ with coordinates (x_1, x_2) by

$$f_{(a_1,b_1,a_2,b_2)}(x_1,x_2) = \begin{cases} 1 & \text{if } a_1 \le x_1 \le b_1 \text{ and } a_2 \le x_2 \le b_2 \\ -1 & \text{otherwise.} \end{cases}$$

The function class of all axis-aligned rectangles in the plane is defined as

$$\mathcal{F}^2_{\text{rec}} = \{f_{(a_1,b_1,a_2,b_2)}(x_1,x_2) : a_1 \le b_1, a_2 \le b_2\}.$$

We will assume that the true labels y of the datapoints $(\mathbf{x}, y)$ are given by some axis-aligned rectangle (this is the realizability assumption discussed in class). The goal of this question is to come up with an algorithm which gets small classification error with respect to any distribution D over $(\mathbf{x}, y)$ with good probability.

The loss function we use throughout the question is the 0-1 loss. It will be convenient to denote a rectangle marked by corners (a_1, b_1, a_2, b_2) as $B(a_1, b_1, a_2, b_2)$. Let $B^* = B(a_1^*, b_1^*, a_2^*, b_2^*)$ be the rectangle corresponding to the function $f_{(a_1^*,b_1^*,a_2^*,b_2^*)}$ which labels the datapoints. Let $S = \{(\mathbf{x}_i, y_i), i \in [n]\}$ be a training set of n samples drawn i.i.d. from D. Please see Fig. 5 for an example.

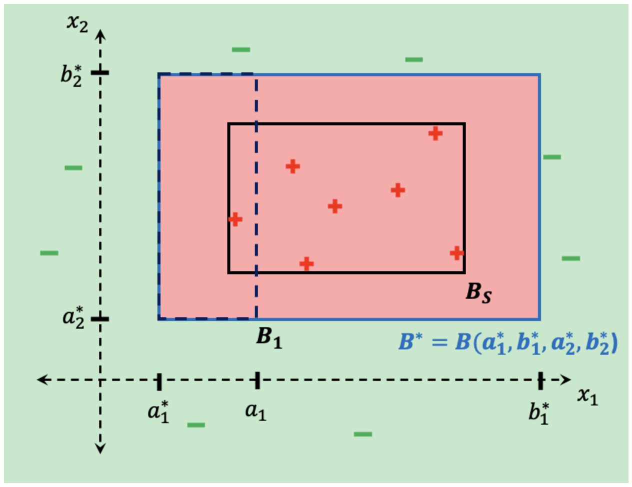


Figure 2: Learning axis-aligned rectangles in two dimensions, here $+$ (red) denotes datapoints in training set S with label 1 and $-$ (green) denotes datapoints with label -1. The true labels are given by rectangle B^* (solid blue line), with everything outside B^* being labelled negative and inside being labelled positive. B_S (solid black line) is the rectangle learned by the algorithm in Part (a). B_1 (dashed black line) is defined in Part (c).

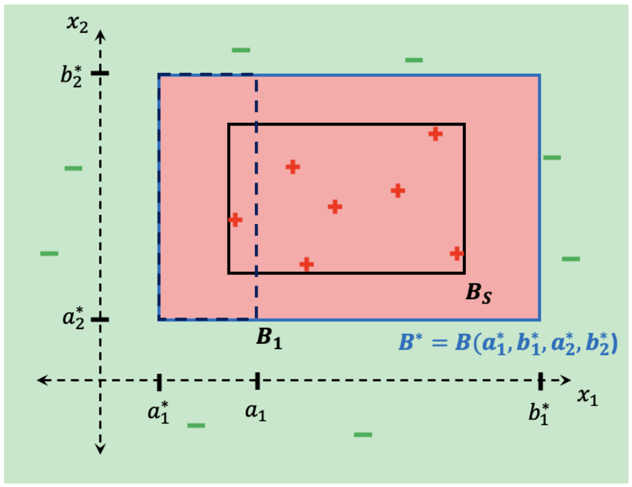


Figure 3: Learning axis-aligned rectangles in two dimensions, here $+$ (red) denotes datapoints in training set S with label 1 and $-$ (green) denotes datapoints with label -1. The true labels are given by rectangle B^* (solid blue line), with everything outside B^* being labelled negative and inside being labelled positive. B_S (solid black line) is the rectangle learned by the algorithm in Part (a). B_1 (dashed black line) is defined in Part (c).

3.1 (3pts) We will follow the general supervised learning framework from class. Given the 0-1 loss, and the function class of axis-aligned rectangles, we want to find an empirical risk minimizer. Consider the algorithm which returns the smallest rectangle enclosing all positive examples in the training set. Prove that this algorithm is an empirical risk minimizer.

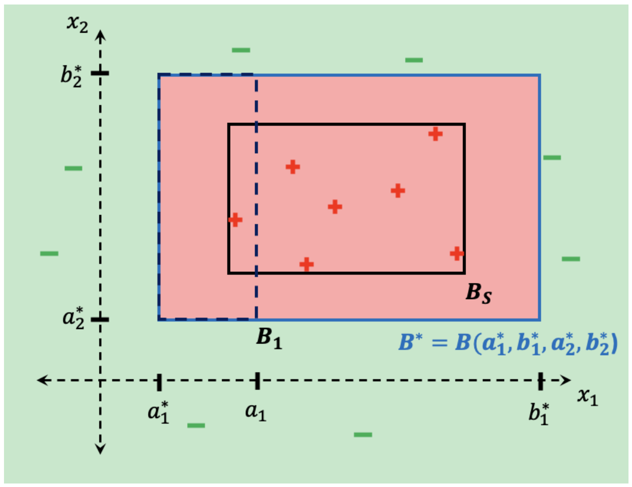
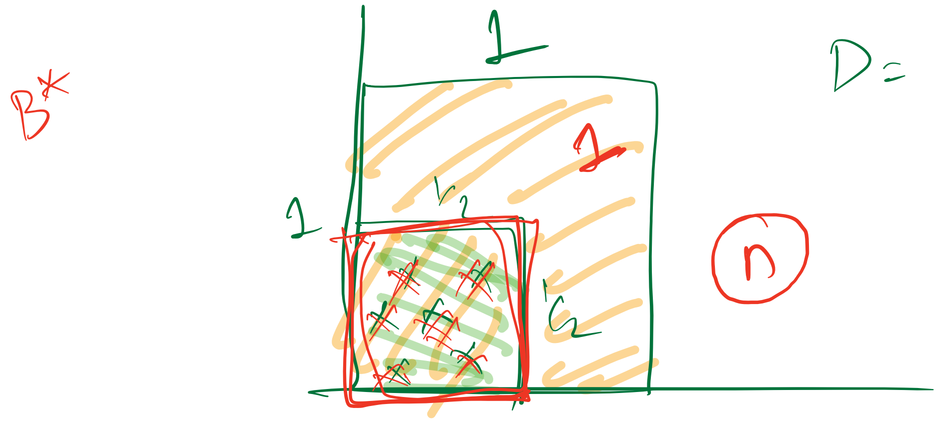


Figure 4: Learning axis-aligned rectangles in two dimensions, here $+$ (red) denotes datapoints in training set S with label 1 and $-$ (green) denotes datapoints with label -1. The true labels are given by rectangle B^* (solid blue line), with everything outside B^* being labelled negative and inside being labelled positive. B_S (solid black line) is the rectangle learned by the algorithm in Part (a). B_1 (dashed black line) is defined in Part (c).

3.2 (2pts) Our next task is to show that the algorithm from the previous part not only does well on the training data, but also gets small classification error with respect to the distribution D. Let B_S be the rectangle returned by the algorithm in the previous part on training set S, and let f_S^{ERM} be the corresponding hypothesis. First, we will convince ourselves that generalization is inherently a probabilistic statement. Let a *bad* training set S' be a training set such that $R(f_{S'}^{ERM}) \geq 0.5$. Pick some simple distribution D and ground-truth rectangle B^*, and give a short explanation for why there is a non-zero probability of seeing a bad training set.



$D = U(x, y)$

$x \in [0,1]$

$y \in [0,1]$

$P(\text{bad Sample}) = \left(\frac{1}{4}\right)^n$

$E(L(f_{B_S}^{ERM})) = 0 \cdot P(\text{green})$ 0.75

$+ 1 \cdot P(\text{orange})$

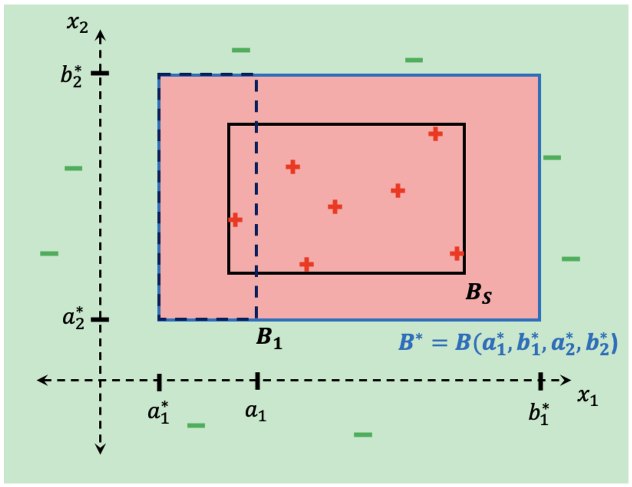
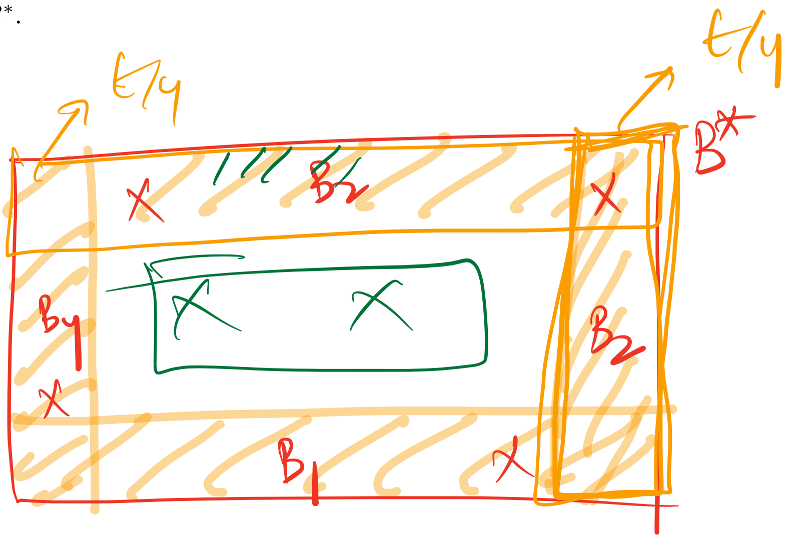


Figure 5: Learning axis-aligned rectangles in two dimensions, here $+$ (red) denotes datapoints in training set S with label 1 and $-$ (green) denotes datapoints with label -1. The true labels are given by rectangle B^* (solid blue line), with everything outside B^* being labelled negative and inside being labelled positive. B_S (solid black line) is the rectangle learned by the algorithm in Part (a). B_1 (dashed black line) is defined in Part (c).

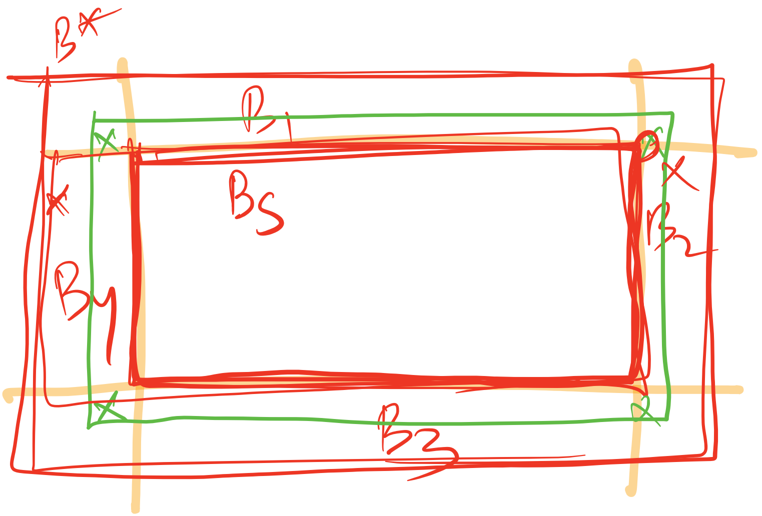
3.3 (5pts) We will now show that with good probability over the training dataset S, f_S^{ERM} does get small error. Show that if $n \geq \frac{4\log(4/\delta)}{\epsilon}$ then with probability at least $1-\delta$, $R(f_S^{ERM}) \leq \epsilon$.

To prove this follow the following steps. Let $a_1 \geq a_1^*$ be such that the probability mass (with respect to D) of the rectangle $B_1 = B(a_1^*, a_1, a_2^*, b_2^*)$ is exactly $\epsilon/4$. Similarly, let b_1, a_2, b_2 be numbers such that the probability mass (with respect to D) of the rectangles $B_2 = B(b_1, b_1^*, a_2^*, b_2^*), B_3 = B(a_1^*, b_1^*, a_2^*, a_2), B_4 = B(a_1^*, b_1^*, b_2, b_2^*)$ are all exactly $\epsilon/4$.

- Show that $B_S \subseteq B^*$.



- Show that if S contains (positive) examples in all of the rectangles B_1, B_2, B_3, B_4 then $R(f_S^{ERM}) \leq \epsilon$.

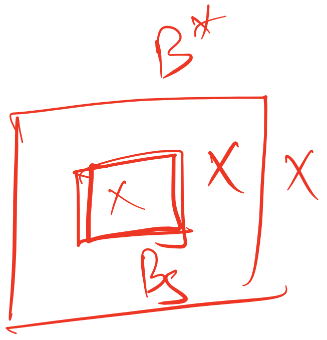


$1-\epsilon$

$$B^* = B_1 \cup B_2 \cup B_3 \cup B_4 \cup B_S$$

$$R(f_{B_S}^{ERM}) < \epsilon$$

$$= E\left[\underline{\mathbb{I}}\left(\underline{B^*(\vec{x})} \neq B_S(\vec{x})\right)\right]$$



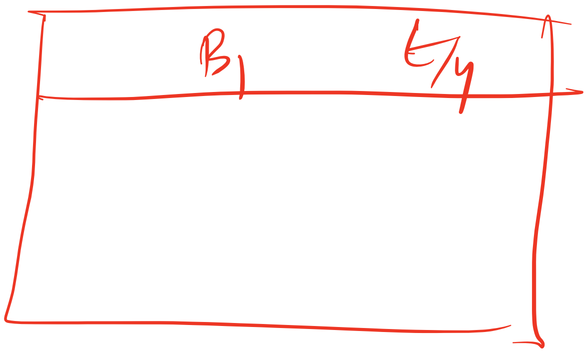
$$= \underline{P\left(B^*(x) \neq \overline{B_S(\vec{x})}\right)}$$

$$= P(x \in B^* \setminus B_S)$$

$$\leq P(x \in B_1 \cup B_2 \cup B_3 \cup B_4)$$

$$\leq P(\underline{x \in B_1}) + P(\underline{x \in B_2}) \ldots$$

- For each $i \in \{1, \ldots, 4\}$ upper bound the probability that S does not contain an example from B_i.

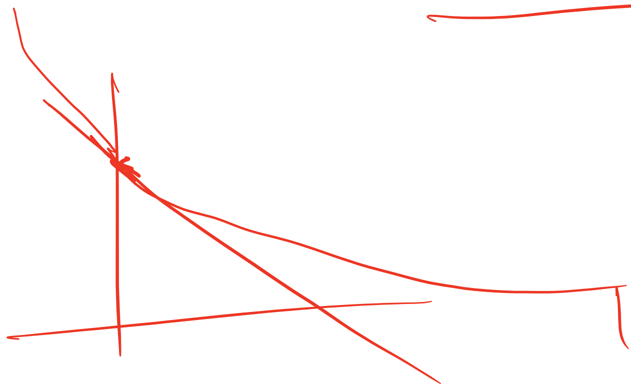


$$P(x \notin B_1) = 1 - \epsilon/4$$

$$S = \{x_1, x_2, \ldots x_n\}$$

$$P(E_1) = P(\underline{\forall x \in S, x \notin B_1}) = \underline{(1 - \epsilon/4)^n}$$

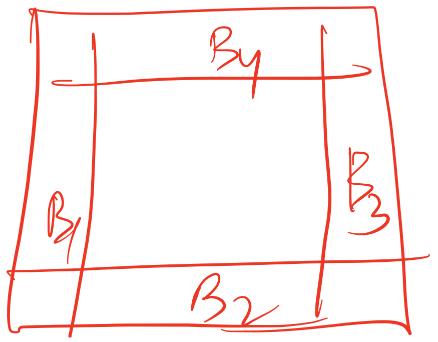
$$\leq \underline{e^{-n\epsilon/4}}$$



$$1 - \underline{x} \leq e^{-x}.$$

- Use the union bound to conclude the argument.

$R(f_S^{ERM}) < \epsilon$



$$\underline{P(R < \epsilon)} = P(\text{good sample})$$

$$1 - P(\text{bad sample})$$

$$\underline{1 - P(E_1 \cup E_2 \cup E_3 \cup E_4)}$$

$$\geq \underline{1 - \Sigma P(E_i)}$$

$E_1 \rightarrow$ event that sample is not from B_1

$$P(R < \epsilon) \geq 1 - \sum_{i=1}^{4} P(E_i)$$

$$\geq 1 - \underline{4e^{-n\epsilon/4}}$$

$$P(R < \epsilon) \geq \underline{1 - 4e^{-n\epsilon/4}} \geq \underline{1 - \delta}$$

$$1 - 4e^{-n\epsilon/4} > 1 - \delta$$

$$n \geq \frac{4 \ln (4/\delta)}{\epsilon}$$

3.4 (5pts) Repeat the previous question for the function class of axis-aligned rectangles in $\mathbb{R}^d$. To show this, try to generalize all the steps in the above question to higher dimensions, and find the number of training samples n required to guarantee that $R(f_S^{ERM}) \leq \epsilon$ with probability at least $1 - \delta$.

d

$\epsilon/4$

$\boxed{\epsilon/2d}$

$$\boxed{\frac{2d \ln (2d/\delta)}{\epsilon}}$$

$\underline{\underline{R < \epsilon}}$