Introduction to Machine Learning, Assignment 2

Diederik Vink

July 12, 2017

Pledge

I, Diederik Adriaan Vink, pledge that this assignment is completely my own work, and that I did not take, borrow or steal work from any other person, and that I did not allow any other person to use, have, borrow or steal portions of my work. I understand that if I violate this honesty pledge, I am subject to disciplinary action pursuant to the appropriate sections of Imperial College London.

Contents

A makefile is used in this situation and is situated inside the folder "code". Just call make. The current setting take circa 3.8 hours to run. The variables "size", "iterations", and "repeats" control the size of the dataset, the number of perceptron iterations and the overal repetitions to get a good average. These values can be altered to decrease runtime.

1 Problem 1

Problem 1 aimed to show the following:

$$\mathbb{P}\left(R(g) \leq \frac{log(\frac{|H|}{\delta})}{n}\right) \geq 1-\delta \tag{1}$$

where

- $R(g)$ is the test error of g
- $\delta \in (0,1)$
- $g \in argmin\hat{R}_n$
- n is the amount of i.i.d. data points upon which g was chosen

Furthermore, in the problem is based around the set $\mathcal{X}$, that has a perfect hypothesis h^* that is an element of the hypothesis class $\mathcal{H}$. For any $\S \in \mathcal{X}$, the corresponding perfect hypothesis label is $h^*(x)$.

Solving Equation (1) was tackled in two phases. The first phase demonstrates how to solve Equation (1) by using the example of a set with a only two positions for the hypothesis h (and therefore $|\mathcal{H}| = 2$ and the second set expands this for a larger hypothesis class using induction from two sides.

1.1 Phase 1

As can be seen in the image below, there are two locations inidcated for h. In the included image, the red arrows indicate the possible regions that the hypothesis h can reside for $R(n) > \epsilon$. a_1 and b_1 indicate the lower and upper bound of ϵ relative to the perfect hypothesis $h^*(x)$. They also represent the location of the two hypothesis to be considered for Phase 1. Everything to the right of $h^*(x)$ is classified as $+1$ (in blue) and everything to the left as -1 (in yellow).

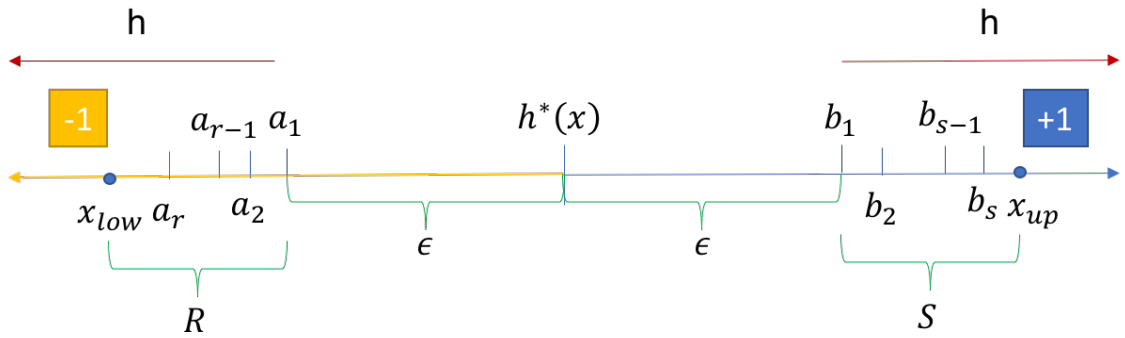


Figure 1: Problem 1 Diagramatic Representation

As per a formal definition:

$$x_{low} = \max x : (x_1, -1) \in \mathcal{X}$$
$$x_{up} = \min x : (x_1, +1) \in \mathcal{X}$$

If one takes an $(x \in (x_{low}, x_{up})) \in \mathcal{X}$, then by definition $\hat{R}_n(x) = 0$. As per the definition of an empirical risk minimizer (ERM), $\hat{R}_n(g)$ is minimized to the smallest possible value, so it can be assumed that $g \in (x_{low}, x_{up})$ and therefore $\hat{R}_n(g) = 0$.

In the situation portrayed in Figure 1, any hypothesis h from the hypothesis class $\mathcal{H}$ has a test error of $R(h)$. For all h's where $\epsilon > 0$, $R(h) > \epsilon$, we know that $h \notin [a_1, b_1]$. Furthermore, for h to

be an ERM, we know that $\hat{R}_n(h) = 0$ and $h \in (x_{low}, x_{up})$. As a result, we can assume that (as is shown in Figure 1):

$$\begin{aligned} x_{low} &< a_1 \\ b_1 &< x_{up} \end{aligned}$$

Based off of all of the previous assumptions, Lecture 3 Slide 10 can be consulted, and the following can be deduced:

$$\begin{aligned} \mathbb{P}(R(h) > \epsilon) &\leq \mathbb{P}(h \notin [a_1, b_1]) \\ &= \mathbb{P}(x_{low} < a_1 \text{ or } x_{up} > b_1) \\ &= \mathbb{P}(x_{low} < a_1) + \mathbb{P}(x_{up} > b_1) \\ &\leq 2(1-\epsilon)^n \\ &\leq 2e^{-\epsilon n} \end{aligned}$$

resulting in Equation 2

$$\mathbb{P}(R(h) > \epsilon) \leq 2e^{-\epsilon n} \tag{2}$$

It is now possible to assume that $\delta = 2e^{-\epsilon n}$. In the problem, it was stated that $\delta \in (0, 1)$. Now that we have determined that it is a valid option to make $\delta = 2e^{-\epsilon n}$ and we can reorganize this in terms of ϵ.

$$\begin{aligned} \delta &= 2e^{-\epsilon n} \\ \frac{\delta}{2} &= e^{-\epsilon n} \\ \log\left(\frac{\delta}{2}\right) &= -\epsilon n \\ \frac{\log\left(\frac{\delta}{2}\right)}{-n} &= \epsilon \end{aligned}$$

which can be simplified into Equation 3

$$\frac{\log\left(\frac{2}{\delta}\right)}{n} = \epsilon \tag{3}$$

Putting together Equations 2 and 3 results in:

$$\mathbb{P}\left(R(h) > \frac{\log\left(\frac{2}{\delta}\right)}{n}\right) \leq \delta \tag{4}$$

As mentioned before, h is an ERM, so we can say that $h = g$, moreover the problem asks that the probability is with regards to $1 - \delta$, resulting in Equation 5:

$$\mathbb{P}\left(R(g) \leq \frac{\log\left(\frac{2}{\delta}\right)}{n}\right) \geq 1 - \delta \tag{5}$$

1.2 Phase 2

Phase 2 of the answer uses induction to prove that Equation 5 can be expanded to Equation 1. Even though Equation 5 is the first step of the induction, the entire induction process will be repeated for clarity and consistency. The induction will occur aim to prove

$$P(R(h) > \epsilon) \leq P(h \notin [a_{rmax}, b_{smax}]) \leq |H|e^{-\epsilon n} \tag{6}$$

where

$$\begin{aligned}
&|\mathcal{R}| + |\mathcal{S}| = |\mathcal{H}| \\
&\{\forall r \in \mathcal{R} \mid x_{low} < a_r \leq a_1\} \\
&\{\forall s \in \mathcal{S} \mid b_1 \leq b_s < x_{up}\} \\
&a_r \geq a_{r-1} \geq \ldots \geq a_2 \geq a_1 \\
&b_s \geq b_{r-1} \geq \ldots \geq b_2 \geq b_1
\end{aligned}$$

and $rmax$ and $smax$ are the greatest values of r and s respectively. Each hypothesis h of the hypothesis class $\mathcal{H}$ is uniquely represented by an element of $\mathcal{R}$ or $\mathcal{S}$. Furthermore $\mathcal{R}$ and $\mathcal{S}$ are the sets that contain all the values of r and s that help all values of a and b meet the conditions stated above. Finally n is the size of the data set. The induction will focus around the values of $rmax$ and $smax$. All of variables mentioned here are depicted in Figure 1.

For purposes of induction and as $h \notin [a_{rmax}, b_{smax}]$ and $h \in (x_{low}, x_{up})$, Equation (6) will be rewritten as the following equivalent equation

$$P(R(h) > \epsilon) \leq \sum_{i=1}^{rmax} \mathbb{P}(x_{low} < a_i) + \sum_{j=1}^{smax} \mathbb{P}(x_{up} > b_j) \tag{7}$$

as is clearly when calculating Equation (2)

1.2.1 Pre-induction

Before the induction to prove Equations (6) and (7) can start, a small detour must be made that will be references later on. This is done head of time so that the main induction does not have to be interrupted.

In this section the aim is prove by induction the following:

$$\sum_{i=1}^{rmax} \mathbb{P}(x_{low} < a_i) \leq (rmax)\mathbb{P}(x_{low} < a_1) \tag{8}$$

Step 1: let $rmax = 1$

$$\begin{aligned}
\sum_{i=1}^{1} \mathbb{P}(x_{low} < a_i) &\leq (1)\mathbb{P}(x_{low} < a_1) \\
\mathbb{P}(x_{low} < a_1) &= (1)\mathbb{P}(x_{low} < a_1)
\end{aligned}$$

Step 1 is clearly true.

Step 2: assume $rmax = t$ is true

$$\sum_{i=1}^{t} \mathbb{P}(x_{low} < a_i) \leq (t)\mathbb{P}(x_{low} < a_1)$$

Step 3: prove Equation (8) holds for $rmax = t + 1$

$$\begin{aligned}
\sum_{i=1}^{t+1} \mathbb{P}(x_{low} < a_i) &= \mathbb{P}(x_{low} < a_{t+1}) + \sum_{i=1}^{t} \mathbb{P}(x_{low} < a_i) \\
&\text{using result from } \textbf{Step 2} \\
&\leq \mathbb{P}(x_{low} < a_{t+1}) + (t)\mathbb{P}(x_{low} < a_1) \\
&\text{considering } \mathbb{P}(x_{low} < a_{t+1}) < \mathbb{P}(x_{low} < a_1) \\
&\leq \mathbb{P}(x_{low} < a_1) + (t)\mathbb{P}(x_{low} < a_1) \\
&= (t+1)\mathbb{P}(x_{low} < a_1) \\
&\text{Q.E.D.}
\end{aligned}$$

which now inductively proves Equation (8). As this applies for $rmax$, the same proof can be used to show that

$$\sum_{i=1}^{smax} \mathbb{P}(x_{up} > b_i) \leq (smax)\mathbb{P}(x_{low} < b_1)$$

1.2.2 First Inductive Step

For the first induction step,

$$\begin{aligned} rmax = smax = 1 \\ \therefore |\mathcal{R}| = |\mathcal{S}| = 1 \\ \therefore |\mathcal{H}| = 2 \end{aligned}$$

With the conditions stated above and as $h \notin [a_{rmax}, b_{smax}]$ and $h \in (x_{low}, x_{up})$:

$$\begin{aligned} x_{low} < a_1 \\ b_1 < x_{up} \end{aligned}$$

$$\begin{aligned} \mathbb{P}(R(h) > \epsilon) &\leq \mathbb{P}(h \notin [a_1, b_1]) \\ &\leq \sum_{i=1}^{1} \mathbb{P}(x_{low} < a_i) + \sum_{j=1}^{1} \mathbb{P}(x_{up} > b_j) \\ &\leq \mathbb{P}(x_{low} < a_1) + \mathbb{P}(x_{up} > b_1) \\ &\leq 2(1-\epsilon)^n \\ &\leq 2e^{-\epsilon n} \end{aligned}$$

as has been demonstrated in Section 1.1 and shows that Equation (6) is true for $rmax = smax = 1$.

1.2.3 Second Inductive Step

Following induction procedure, for the second inductive step we apply the following conditions

$$\begin{aligned} rmax = t, smax = u \\ \therefore |\mathcal{R}| = t, |\mathcal{S}| = u \\ \therefore |\mathcal{H}| = t + u \\ x_{low} < a_t \\ b_u < x_{up} \end{aligned}$$

and then assume that

$$\begin{aligned} P(R(h) > \epsilon) &\leq P(h \notin [a_t, b_u]) \\ &\leq \sum_{i=1}^{t} \mathbb{P}(x_{low} < a_i) + \sum_{j=1}^{u} \mathbb{P}(x_{up} > b_j) \\ &\text{using Equation (8) prove in Section 1.2.1} \\ &\leq t\mathbb{P}(x_{low} < a_1) + u\mathbb{P}(x_{up} > b1) \\ &\leq (t+u)e^{-\epsilon n} \end{aligned}$$

is true.

1.2.4 Third Inductive Step

The final step of the inductive procedure is showing that for

$$\begin{aligned} rmax = t + 1, smax = u + 1 \\ \therefore |\mathcal{R}| = t + 1, |\mathcal{S}| = u + 1 \\ \therefore |\mathcal{H}| = (t + 1) + (u + 1) \end{aligned}$$

and then show that

$$
\begin{aligned}
P(R(h) > \epsilon) &\leq P(h \notin [a_{t+1}, b_{u+1}]) \\
&\leq \sum_{i=1}^{t+1} \mathbb{P}(x_{low} < a_i) + \sum_{j=1}^{u+1} \mathbb{P}(x_{up} > b_j) \\
&\leq \mathbb{P}(x_{low} < a_{t+1}) + \sum_{i=1}^{t} \mathbb{P}(x_{low} < a_i) + \mathbb{P}(x_{up} > b_{s+1}) + \sum_{j=1}^{u} \mathbb{P}(x_{up} > b_j) \\
&\text{using assumption from Second Inductive Setup (Section 1.2.3)} \\
&\leq \mathbb{P}(x_{low} < a_{t+1}) + t\mathbb{P}(x_{low} < a_1) + \mathbb{P}(x_{up} > b_{u+1}) + u\mathbb{P}(x_{up} > b_j) \\
&\text{considering } \mathbb{P}(x_{low} < a_{t+1}) \leq \mathbb{P}(x_{low} < a_1) \\
&\leq (t+1)\mathbb{P}(x_{low} < a_1) + \mathbb{P}(x_{up} > b_{u+1}) + u\mathbb{P}(x_{up} > b_j) \\
&\text{considering } \mathbb{P}(x_{up} > b_{u+1}) \leq \mathbb{P}(x_{up} > b_1) \\
&\leq (t+1)\mathbb{P}(x_{low} < a_1) + (u+1)\mathbb{P}(x_{up} > b_j) \\
&\leq ((t+1) + (u+1))e^{-\epsilon n} \\
&\text{Q.E.D}
\end{aligned}
$$

and as $|H| = rmax + smax = (t+1) + (u+1)$

$$P(R(h) > \epsilon) > |H|e^{-\epsilon n} \tag{9}$$

From Equation (9) it is possible to do the same simplifications as in Section 1.1 as shown below:

$$
\begin{aligned}
\delta &= |H|e^{-\epsilon n} \\
\frac{\delta}{|H|} &= e^{-\epsilon n} \\
\log\left(\frac{\delta}{|H|}\right) &= -\epsilon n \\
\frac{\log\left(\frac{\delta}{|H|}\right)}{-n} &= \epsilon
\end{aligned}
$$

which can be simplified into Equation 10

$$\frac{\log\left(\frac{|H|}{\delta}\right)}{n} = \epsilon \tag{10}$$

Putting together Equations 9 and 10 results in:

$$\mathbb{P}\left(R(h) > \frac{\log\left(\frac{|H|}{\delta}\right)}{n}\right) \leq \delta \tag{11}$$

As mentioned before in Section 1.1, h is an ERM, so we can say that $h = g$. Moreover the question asks for the probability with regards to $1 - \delta$, resulting in Equation 12:

$$\mathbb{P}\left(R(g) \leq \frac{\log\left(\frac{|H|}{\delta}\right)}{n}\right) \geq 1 - \delta \tag{12}$$

which is exactly what the question asked to show.

When comparing Equation (4) to Hoeffding's inequality, it is clear that Equation (4) clearly provides a tighter bound. This is because Hoeffding's inequality is generic for any hypothesis h in the hypothesis class $\mathcal{H}$, whereas Equation 4 only refers to the bound on a specific hypothesis h in the hypothesis class $\mathcal{H}$. Furthermore, it can be seen that that Equation (4) is of complexity $O\left(\frac{1}{n}\right)$, which allows for a far faster convergence than Hoeffding's with it's $O\left(\frac{1}{\sqrt[root]{n}}\right)$ complexity.

2 Problem 2

Problem 2 seeks to demonstrate that the VC dimension (d_{VC}) of a hypothesis class $\mathcal{H}$ is 4 where

$$\mathcal{H} = \{h : \mathbb{R}^2 \to \{0, 1\} : h = \mathbb{I}(x \in R(a_1, a_2, b_1, b_2))\} \tag{13}$$

where $a_1 \leq a_2$ and $b_1 \leq b_2$. When finding the d_{VC} of $\mathcal{H}$, it always follows that $d_{VC} = k - 1$ where k is the amount of points that $\mathcal{H}$ cannot shatter (refer to Lecture 3 slide 23). According to definition, "A hypothesis class H shatters a finite set $\mathcal{C} \subset \mathcal{X}$ if the restriction of $\mathcal{H}$ to $\mathcal{C}$ is the set of all functions from $\mathcal{C}$ to {0, 1}. That is, $|\mathcal{H}_{\mathcal{C}}| = 2^{|\mathcal{C}|}$" [Shalev-Shwartz and Ben-David, 2014]. This means that if any hypothesis $h \in \mathcal{H}$ can produce 2^k combinations of all the k points, then $\mathcal{H}$ can shatter k points. For all values of k, if $\mathcal{H}$ can shatter k points, it can shatter $k - 1$ points, as less points is always easier to shatter.

The d_{VC} of a hypothesis class $\mathcal{H}$ is known as the maximum number of points $\mathcal{H}$ shatters (see Lecture 3 Slide 21)

$$d_{VC}(H) = \max\{n : m_H(n) = 2^n\} \tag{14}$$

A practical explanation of shattering is that a hypothesis class $\mathcal{H}$ can shatter k points if there exists any hypothesis h in $\mathcal{H}$ that can cause any point to become positive or negative. This also implies, that if $\mathcal{H}$ can shatter $k = 3$ points, it can shatter $k = 2$ points. Furthermore the amount of ways in which k points in a set of n can be separated (the number of dichotomies) can be represented as $\binom{n}{k}$. As a result, for $\mathcal{H}$ to shatter k points

$$\sum_{i=0}^{k} \binom{n}{i} = 2^k \tag{15}$$

must hold. Equation (15) is an adaptation of Sauer's Lemma as stated in Lecture 3 Slide 27, to indicate the value of $|\mathcal{H}(x_1, \ldots, x_n)|$ for when $\mathcal{H}$ can shatter k points. This was arrived at in an intuitive way rather than through induction. The remark in Lecture 3 Slide 25 expands Equation (15) to an n and any k, not just where $\mathcal{H}$ can shatter k points.

Problem 2 needs to be solved in two phases. The first phase shows how $\mathcal{H}$ can shatter 4 points and the second that $\mathcal{H}$ cannot shatter 5 points.

2.1 Shattering 4 Points

This section aims to show that utilizing any axis aligned rectangle within the hypothesis class $\mathcal{H}$, 4 points can be shattered. As mentioned, this requires to show that $\sum_{i=0}^{4} \binom{n}{i} = 16$ is the amount of ways that an axis aligned rectangle could separate i points.

For all the diagrams that are going to come in this section, the orange colored dots represent the dots inside the axis aligned rectangle (a hypothesis h from $\mathcal{H}$) and are therefore classified as $+1$, and the blue colored dots represent the dots outside the axis aligned rectangle and are classified as -1.

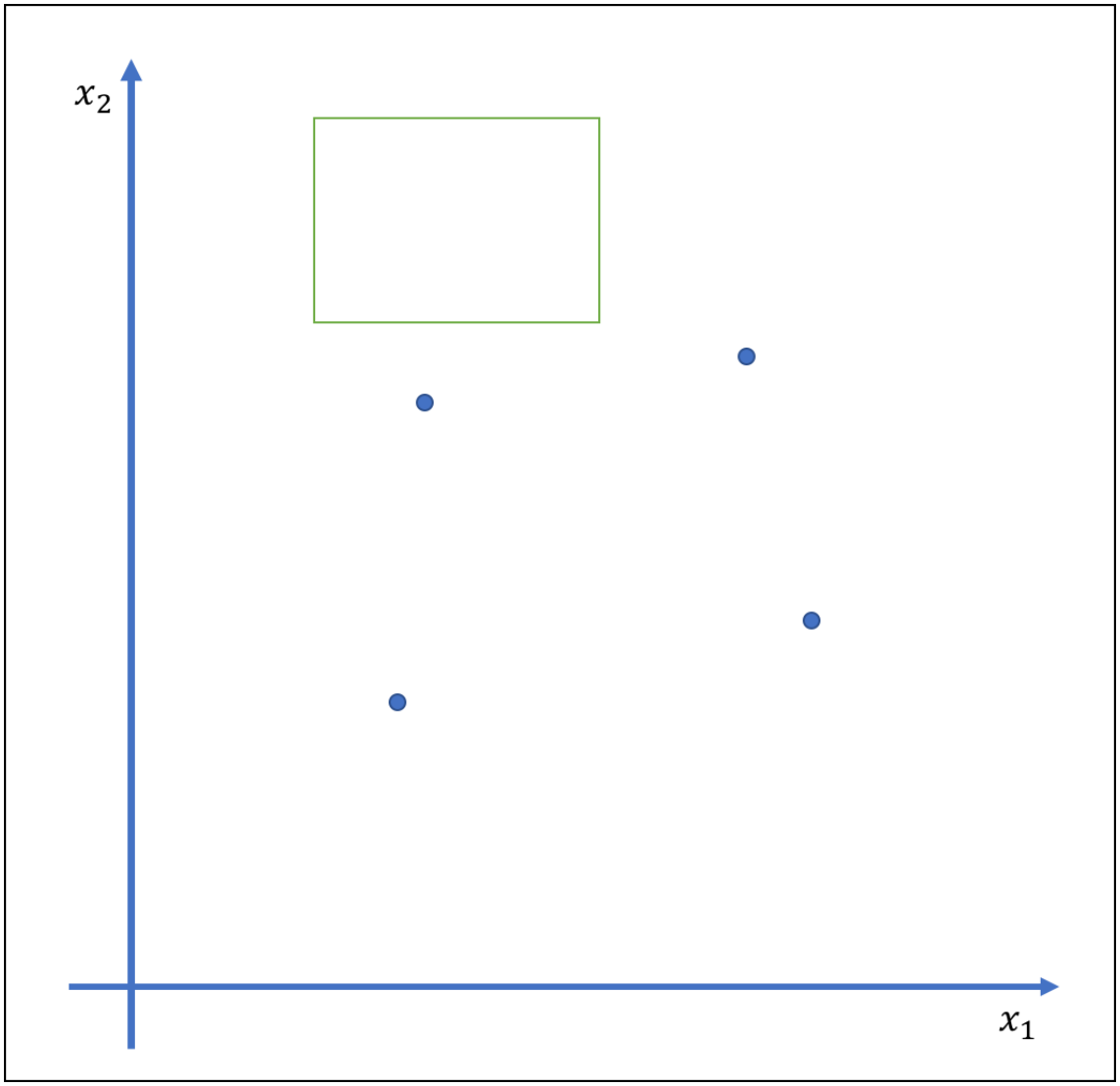
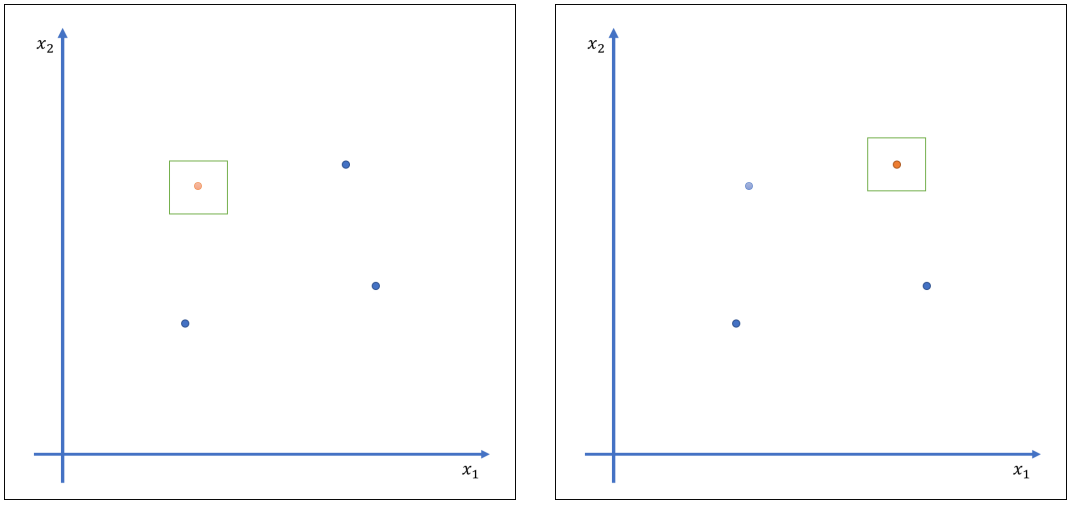


Figure 2: Diagramatic Representation of Shattering $k = 0$ Points

Figure 2 demonstrates, how an axis aligned rectangle would shatter 0 points. In this setup, if it were to be said that everything in the green rectangle is classified as +1, and everything outside as -1, it becomes clear that every combination of classifying 0 points is achieved.

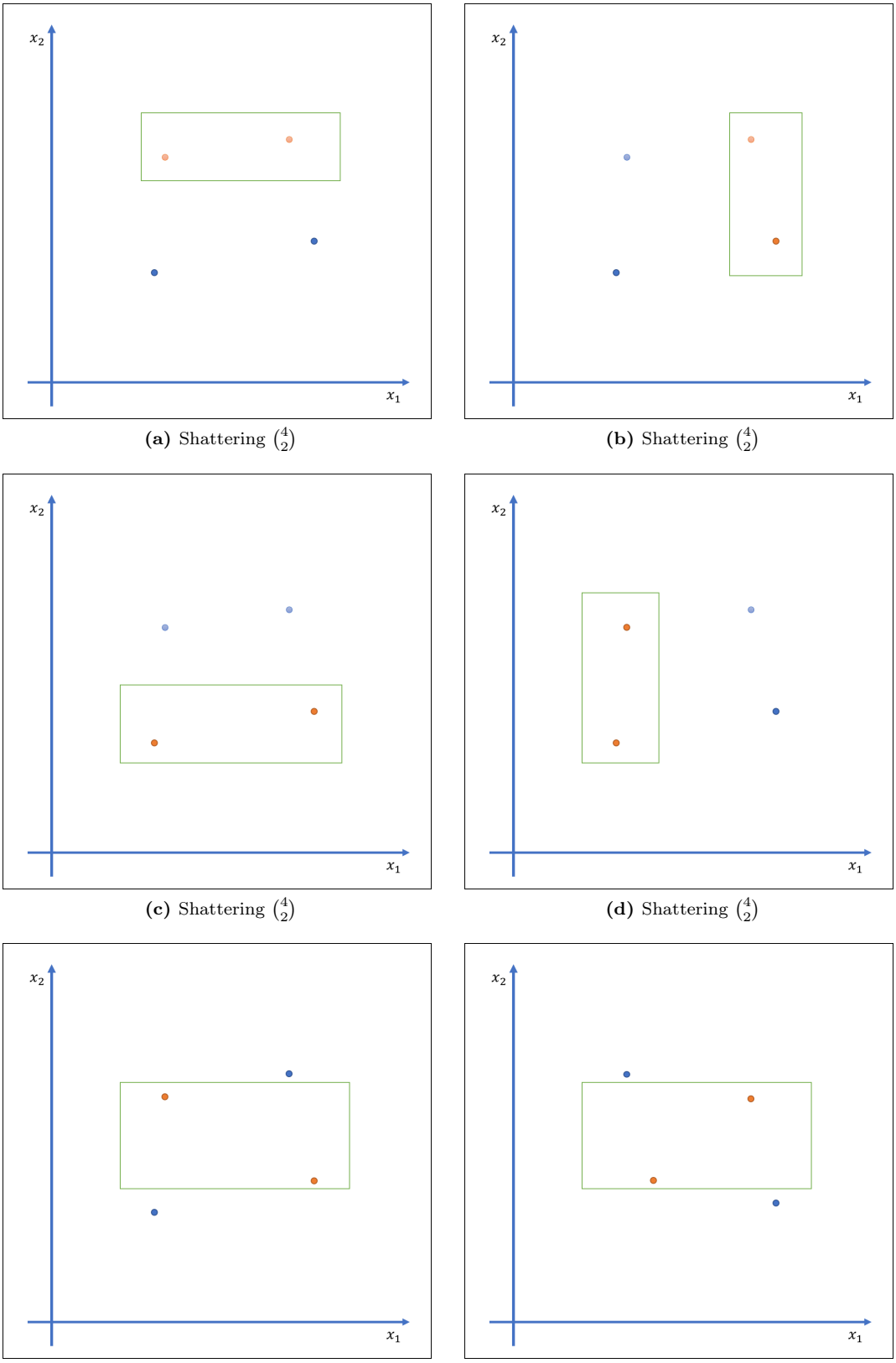


(a) Shattering $\binom{4}{1}$ **(b)** Shattering $\binom{4}{1}$

(c) Shattering $\binom{4}{1}$ **(d)** Shattering $\binom{4}{1}$

Figure 3: Diagramatic Representation of Shattering $k = 1$ Points

Just like Figure 2, Figure 3 demonstrates how for each of the four points, only one point is classified as $+1$. There are clearly 4 possible combinations, and as $\binom{4}{1} = 4$, $\mathcal{H}$ can shatter $k = 1$ points.



(a) Shattering $\binom{4}{2}$

(b) Shattering $\binom{4}{2}$

(c) Shattering $\binom{4}{2}$

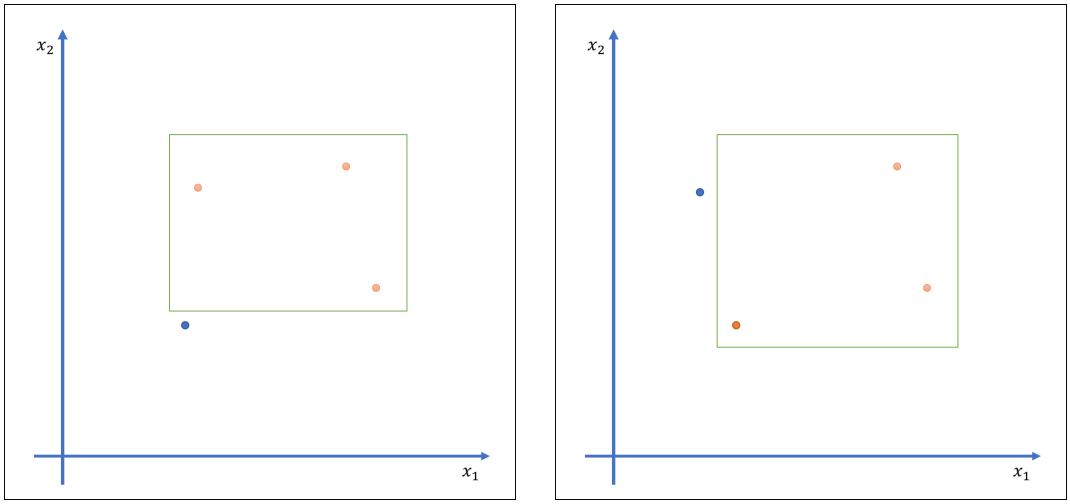
(d) Shattering $\binom{4}{2}$

(e) Shattering $\binom{4}{2}$

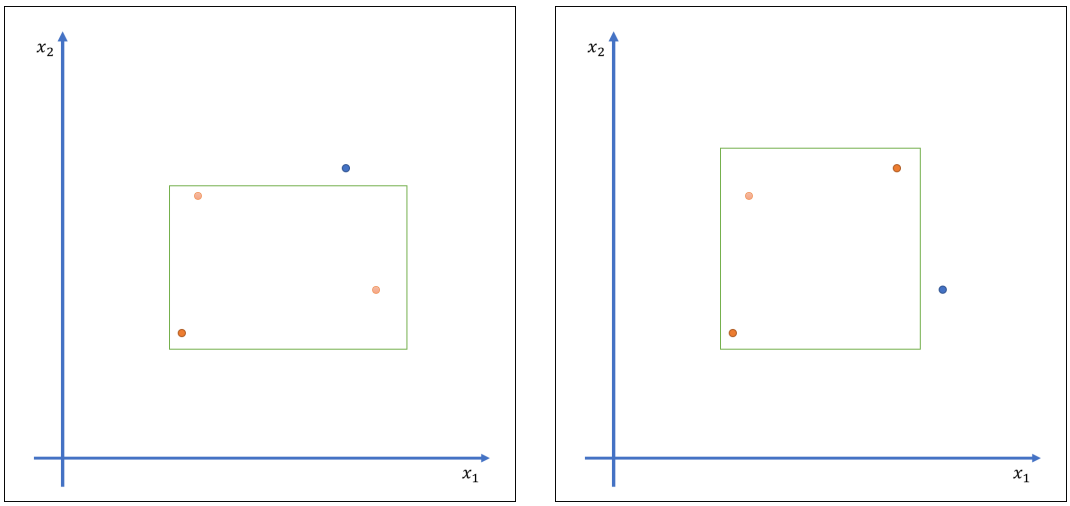
(f) Shattering $\binom{4}{2}$

Figure 4: Diagramatic Representation of Shattering $k = 2$ Points

Figure 4 demonstrates how for classifying any combination of 2 out of the 4 points as $+1$, an axis aligned rectangle can be drawn. There are clearly 6 possible combinations, and as $\binom{4}{2} = 6$, $\mathcal{H}$ can shatter $k = 2$ points.



(a) Shattering $\binom{4}{3}$ **(b)** Shattering $\binom{4}{3}$



(c) Shattering $\binom{4}{3}$ **(d)** Shattering $\binom{4}{3}$

Figure 5: Diagramatic Representation of Shattering $k = 3$ Points

Figure 5 demonstrates how for any classifying combination of 3 out of the 4 points as $+1$, an axis aligned rectangle can be drawn. There are clearly 4 possible combinations, and as $\binom{4}{3} = 4$, $\mathcal{H}$ can shatter $k = 3$ points.

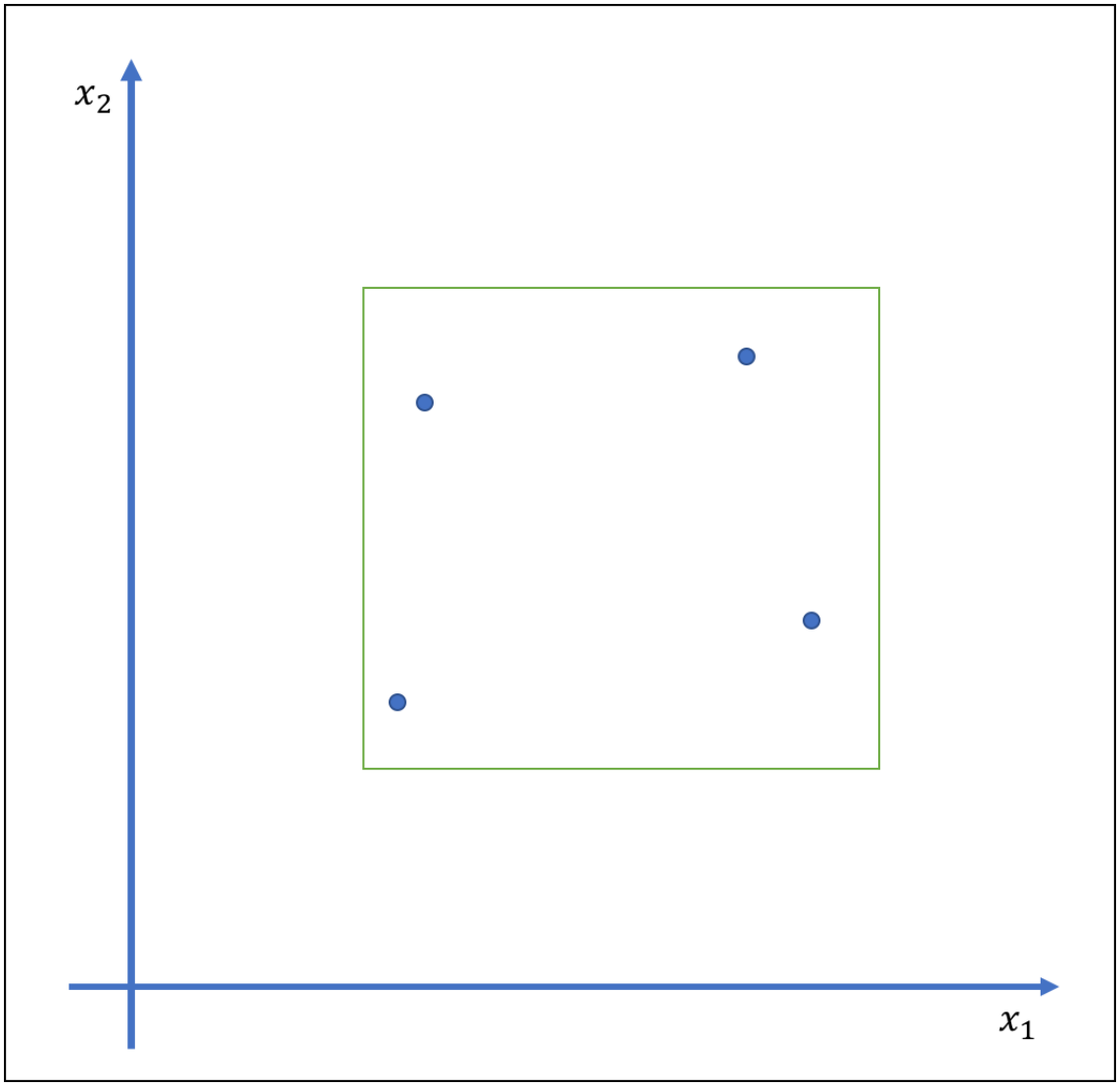


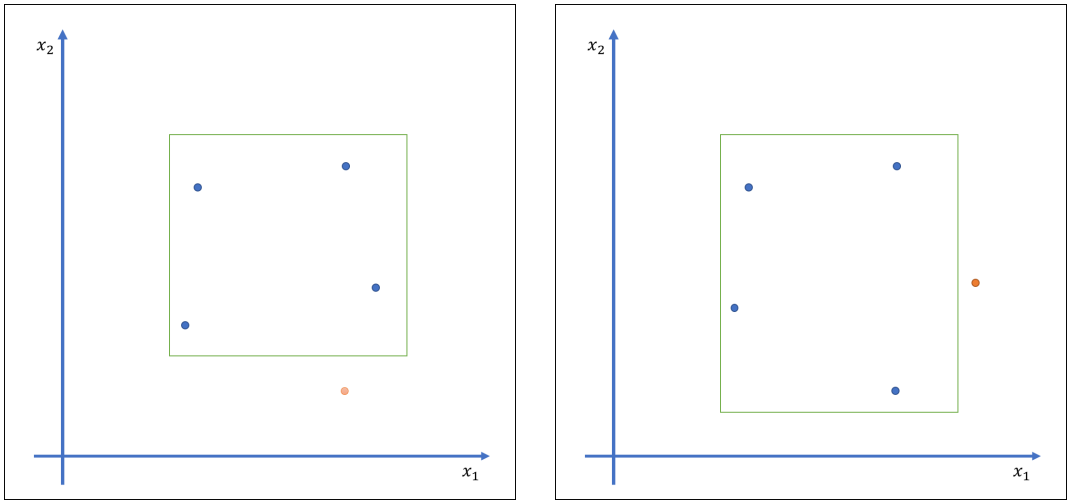
Figure 6: Diagramatic Representation of Shattering $k = 4$ Points

Figure 6 demonstrates how for classifying all the 4 points as $+1$, an axis aligned rectangle can be drawn. There is clearly 1 possible combination, and as $\binom{4}{4} = 1$, $\mathcal{H}$ can shatter $k = 4$ points.

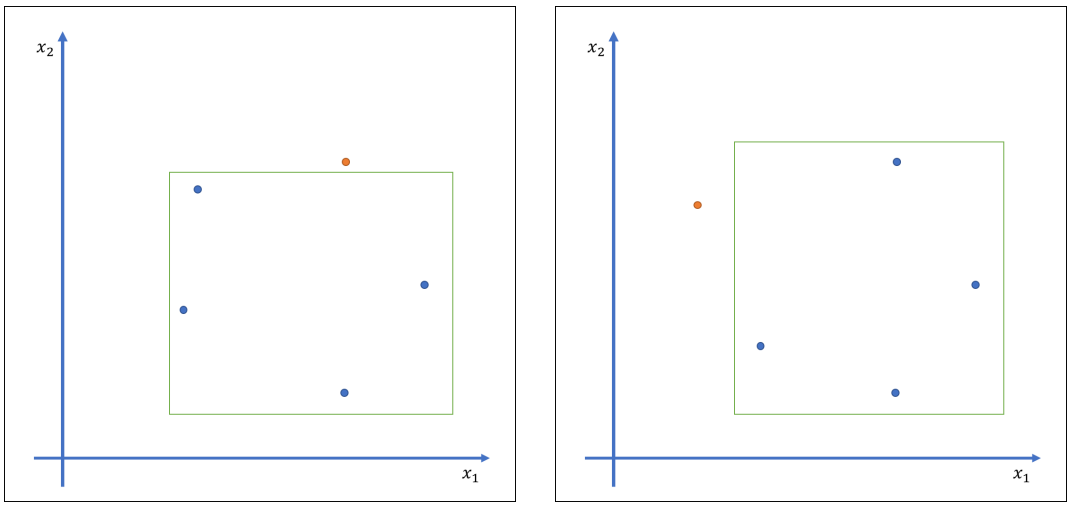
Considering everything that has been presented so far, it is clear that if $\mathcal{H}$ contains axis aligned rectangles, it can clearly shatter 4 points as is shown in the diagrams from above. This is further reinforced by the fact that the adaptation of Sauer's Lemma demonstrated by Equation (15) is met for 4 points, considering the 16 different classification options are demonstrated above and that $\sum_{i=0}^{4} = 16$, indicating that $\mathcal{H}$ can shatter 4 points.

2.2 Shattering 5 Points

The section aims to prove that $\mathcal{H}$ cannot shatter 5 points. Section 2.1 provided a graphical demonstration of shattering and its connection to Sauer's Lemma. Using the same methodology, it can be demonstrated that $\mathcal{H}$ cannot shatter 5 points. For the situations of $\binom{5}{0}, \binom{5}{1}, \binom{5}{2}, \binom{5}{3}, \binom{5}{5}$ similar figures to those presented in Section 2.1 can be produced. The one special case to consider is for $\binom{5}{4}$, or where $n = 5$ and $k = 4$.



(a) Shattering $\binom{5}{4}$ **(b)** Shattering $\binom{5}{4}$



(c) Shattering $\binom{5}{4}$ **(d)** Shattering $\binom{5}{4}$

Figure 7: Diagramatic Representation of Shattering $k = 4$ Points

As Figure 7 demonstrates, there are 4 ways in which an axis aligned rectangle can classify 4 points as +1 and 1 point as -1. The hypothesis in Figure 7a excludes the bottommost point from the rectangle, Figure 7b excludes the rightmost point, Figure 7c excludes the topmost and Figure 7d excludes the left most point. The one option that is missing is the option where the 4 points at the extremities (topmost, rightmost, bottommost and leftmost) are classified as +1 and the last dot (now the dot in the middle) must be classified as -1. This is completely impossible, and as a result, for $n = 5$ points, $\mathcal{H}$ cannot shatter $k = 4$ points. As a result, it can be concluded that $\mathcal{H}$ cannot shatter $n = 5$. Furthermore, the size of $\mathcal{H}$ can be bounded by $\sum_{i=0}^{d_{VC}} \binom{n}{i}$.

According the the definition provided in Lecture 3 Slide 21, d_{VC} is the largest number of points $\mathcal{H}$ shatters, which in this case is 4, so $d_{VC} = 4$.

3 Problem 3

For reference, whenever $\mathcal{H}$ is mentioned, it is a shorthand for $\mathcal{H}(x_1, \ldots, x_n)$

Problem 3 aims to show that for any n and k and dataset $x = \{x_1, \ldots, x_n\} \subset \mathcal{X}$ of size n (for some base set $\mathcal{X}$) with $x_i \neq x_j$, there exists a hypothesis class $\mathcal{H} \subset \{h : \mathcal{X} \to \{-1, +1\}\}$ that cannot shatter any k points form X and $|\mathcal{H}(x_1, \ldots, x_n)| = \sum_{i=0}^{k-1} \binom{n}{i}$.

As demonstrated in Section 2 for Problem 2, for a dataset of size n (like $\mathcal{X}$ in this Problem) with a $d_{VC} = k - 1$, the hypothesis class $\mathcal{H}$ can shatter up to and including $k - 1$ points. Now for any value of $i < d_{VC}$, it is possible to classify i of the total n points as $+1$ and have $n - i$ points classified as -1 and this would still be shattered by $\mathcal{H}$. The total amount of dichotomies that can be formed for every value of i is

$$\sum_{i=0}^{d_{VC}} \binom{n}{i} \tag{16}$$

This number of dichotomies is $|\mathcal{H}|$.

From Section 2 it is clear that the maximum number of points $\mathcal{H}$ can shatter is d_{VC}. So if we say that k is the number of points that $\mathcal{H}$ cannot shatter, then $k - 1$ points can be shattered and $d_{VC} = k - 1$. If we then apply this knowledge to Equation (16) it results in

$$|\mathcal{H}| = \sum_{i=0}^{k-1} \binom{n}{i} \tag{17}$$

4 Problem 4

4.1 Part a

Problem 4a) aims to prove that where $d_{VC}(n) < \infty$, for all n

$$m_{\mathcal{H}}(n) \leq n^{d_{VC}} + 1 \leq (n+1)^{d_{VC}} \tag{18}$$

Utilizing the information provided in Lecture 3 Slide 27 it is known that

$$m_{\mathcal{H}}(n) \leq \sum_{i=0}^{d_{VC}} \binom{n}{i} \tag{19}$$

It is possible to prove Equation (18) through induction that

$$\sum_{i=0}^{d_{VC}} \binom{n}{i} \leq n^{d_{VC}} + 1 \tag{20}$$

4.1.1 Step 1: prove for $n = 1$

For n = 1, $d_{VC} > 0$ (if $d_{VC} < 0$, then there would be no dataset)

$$\begin{aligned}
\sum_{i=0}^{d_{VC}} \binom{1}{i} &\leq 1^{d_{VC}} + 1 \\
\sum_{i=0}^{1} \binom{1}{i} &= 1^1 + 1 \\
\binom{1}{0} + \binom{1}{1} + \ldots \binom{1}{d_{VC}} &= 1 + 1 \\
\binom{1}{0} + \binom{1}{1} &= 1 + 1 \qquad\qquad (\text{as} \binom{1}{n} = 0 \text{when} n > 0) \\
2 &= 2
\end{aligned}$$

proving that it works for $n = 1$.

4.1.2 Step 2: assume correct for $n = k$

$$\sum_{i=0}^{d_{VC}} \binom{k}{i} \leq k^{d_{VC}} + 1 \tag{21}$$

4.1.3 Step 3: prove for $n = k + 1$

$$\begin{aligned}
\sum_{i=0}^{d_{VC}} \binom{k+1}{i} &= 1 + \sum_{i=1}^{d_{VC}} \binom{k+1}{i} \\
&= 1 + \sum_{i=1}^{d_{VC}} \binom{k}{i} + \sum_{i=1}^{d_{VC}} \binom{k}{i-1} \quad \text{(using reverse of last step in proof in Lec 3 Slide 26)} \\
&\leq 1 + k^{d_{VC}} + \sum_{i=1}^{d_{VC}} \binom{k}{i-1} && \left(\text{as} \sum_{i=1}^{d_{VC}} \binom{k}{i} \leq k^{d_{VC}}\right) \\
&= 1 + k^{d_{VC}} + \sum_{j=0}^{d_{VC}-1} \binom{k}{j} && (\text{where} j = i-1) \\
&\leq 1 + k^{d_{VC}} \sum_{j=0}^{d_{VC}-1} k^j && \left(\text{as} \binom{k}{j} \leq k^j\right) \\
&= 1 + \sum_{j=0}^{d_{VC}} k^j \\
&= (k+1)^{d_{VC}} + 1 && \left(\text{as}(k+1)^{d_{VC}} \text{is the binomial expansion of } \sum_{j=0}^{d_{VC}} k^j\right)
\end{aligned}$$

Q.E.D.

This proves that:

$$m_{\mathcal{H}}(n) \leq \sum_{i=0}^{d_{VC}} \binom{n}{i} \leq (n)^{d_{VC}} + 1 \tag{22}$$

4.2 Part b

Problem 4b) aims to prove that where $d_{VC}(n) < \infty$ and for all $n \geq d_{VC}$

$$m_{\mathcal{H}}(n) \leq \left(\frac{ne}{d_{VC}}\right)^{d_{VC}} \tag{23}$$

It merely relies on the following equations

$$\sum_{i=0}^{n} \binom{n}{i} a^i b^{n-i} = (a+b)^n \tag{24}$$

$$\left(1 + \frac{1}{x}\right)^{x d_{VC}} \leq e^{d_{VC}} \tag{25}$$

Equation (25) is an adaptations of the equation provided in the hint and Equation (24) is the equation for a binomial expansion.

What follows is the consecutive proof of Equation (23)

$$
\begin{aligned}
\sum_{i=0}^{d_{VC}} \binom{n}{i} &\leq \sum_{i=0}^{d_{VC}} \binom{n}{i} \left(\frac{n}{d_{VC}}\right)^{d_{VC}-i} && \left(1 \leq \sum_{i=0}^{d_{VC}} \left(\text{as } \frac{n}{d_{VC}}\right)^{d_{VC}-i}\right) \\
&= \sum_{i=0}^{d_{VC}} \binom{n}{i} n^{d_{VC}-i} {d_{VC}}^{i-d_{VC}} \\
&= n^{d_{VC}} {d_{VC}}^{-d_{VC}} \sum_{i=0}^{d_{VC}} \binom{n}{i} n^{-i} {d_{VC}}^{i} \\
&= \left(\frac{n}{d_{VC}}\right)^{d_{VC}} \sum_{i=0}^{d_{VC}} \binom{n}{i} \left(\frac{d_{VC}}{n}\right)^{i} \\
&= \left(\frac{n}{d_{VC}}\right)^{d_{VC}} \sum_{i=0}^{d_{VC}} \binom{n}{i} \left(\frac{d_{VC}}{n}\right)^{i} 1^{n-i} \\
&\leq \left(\frac{n}{d_{VC}}\right)^{d_{VC}} \sum_{i=0}^{n} \binom{n}{i} \left(\frac{d_{VC}}{n}\right)^{i} 1^{n-i} && (n \geq d_{VC}) \\
&= \left(\frac{n}{d_{VC}}\right)^{d_{VC}} \left(1 + \frac{d_{VC}}{n}\right)^{n} && \text{(using Equation (24))} \\
&\leq \left(\frac{n}{d_{VC}}\right)^{d_{VC}} \left(1 + \frac{1}{x}\right)^{x d_{VC}} && \left(\text{substitute } x = \frac{n}{d}\right) \\
&\leq \left(\frac{n}{d_{VC}}\right)^{d_{VC}} e^{d_{VC}} && \text{(using Equation (25))} \\
&= \left(\frac{ne}{d_{VC}}\right)^{d_{VC}}
\end{aligned}
$$

and known form Sauer's Lemma $m_{\mathcal{H}}(n) \leq \sum_{i=0}^{d_{VC}} \binom{n}{i}$, then we can now conclude that $m_{\mathcal{H}}(n) \leq \left(\frac{ne}{d_{VC}}\right)^{d_{VC}}$

Both the result from Problem 4a) and 4b) demonstrate an upper bound on the value of $m_{\mathcal{H}}(n)$. The bound for Problem 4a) represents a generic bound for any value of n that will always hold. The bound for Problem 4b) provides a far tighter bound as it narrows down the bound to hold only for values of $n \geq d_{VC}$. As a result, when possible the bound in Problem 4b) gives a better bound, but with a limited situational applicability.

5 Problem 5

5.1 Part a

For Problem 5a), there are two components to show that $d_{VC}(P_q)$ is at most $q+1$. Generically in the perceptron algorithm until now the feature matrix has been a vector containing all of the features.

The problem can be solved inductively as follows. **Step 1** is showing that it works for $q = 0$, as for a $d_{VC}(P_q) = 1$, only 1 point can be shattered, so the solution holds for $q = 0$.

Step 2 is assuming that for when $q = k$, $d_{VC}(P_q) = k + 1$.

For **Step 3** we must now show that for $q = k + 1$, $d_{VC}(P_{q+1}) = k + 2$. This can be confirmed as true, because when the degree of a polynomial is increased, it is provided with another possible turning point. This additional (and optional) turning point makes it possible to shatter one more point than for P_{q+1}, meaning that the $d_{VC}(P_q)$ is now one higher, and it is inductively proven.

For coursework 1, the transformation was linear so the features were: the constant term (denoted by a row of ones), the x_1 term and the x_2 term. Now that the classification line has become more complex and non-linear, each power of x_1 has become a feature. The size of the transformation vector is at all times $q + 2$ for a polynomial in $\mathbb{R}^q$. If it can be shown that the size of of the feature matrix can be $q + 1$ the proof on Lecture 4 Slide 6 can be applied to prove that $d_{VC}P_q \geq q + 1$ and therefore $d_{VC}P_q = q + 1$.

Decreasing the size of feature matrix is an entire area of research. According to [Mladenic] and [D. Bhattacharjee] there are various strategies that exist to reduce the size of the feature matrix including, but not limited to, standard eigenspace projection, using the energy dimension, the odds ratio, information gain, etc.

From Lecture 4 Slide 6, we can see that the data matrix X that is referred to is the same as the feature matrix discussed so far. The only difference is that $x_2, x_3, x_4, \ldots, x_d, x_{d+1}$ from Slide 6 is $x_1^2, x_1^3, x_1^4, \ldots, x_1^q, x_2$ for the current problem respectively as we have removed on feature. To be able to shatter all the $q+1$ points for $\mathbb{R}^q$, we need $y = sign(Xw)$ where y is the classification vector and w is the perceptron weights vector (like it has been in coursework 1 and in the perceptron code for Section 5.2). As shown on Lecture 4 Slide 6, $w = X^{-1}y$ as X is invertible, so it is possible to shatter $q + 1$ points. As a result, it can be concluded that $d_{VC}(P_q) = q + 1$.

5.2 Part b

Problem 5b) was the only code oriented question present in the coursework. This question was aimed at the implementation of the calculation of the structural risk minimization (SRM). The SRM finds the hypothesis class g as discussed in Equation 26.

$$g = \underset{h \in \mathcal{H}}{argmin}\ \hat{R}_n(h) + \Omega(n, \mathcal{H}(h), w_i\delta) \tag{26}$$

where $\hat{R}_n$ is the training error of that specific hypothesis and Ω is the complexity term. The implementation of finding the SRM required code from the previous coursework with some additional components added on.

The first step that was taken was writing the entire code in C++. Due to the excessive size of the test that needed to be run, working in a more efficient language seemed very logical. The previous coursework was written in R, as it was easy to manipulate matrices and generate graphics. For this coursework graph generation was no longer necessary and matrix operations were easily done through the inclusion of the Eigen library. With the inclusion of this library, writing the coursework in C++ became a logical decision.

The data generation, classification, the perceptron algorithms with a functioning empirical risk

minimizer (ERM) and the calculation of $R_n(g)$ (test error) that were used for the previous coursework are all still valid, and as a result the details of those implementations will not be discussed in this report. The pseudocode demonstrated in Algorithm 1 demonstrates the additions this coursework required.

Algorithm 1 SRM & Test Error Calculation algorithm

procedure FIND MINIMUM ERM OF ALL HYPOTHESIS CLASSES

for all q `in range` $0 \rightarrow 4$ **do**

for all repetitions **do**
Generate uniformly distrubted random data set of specified size (n)
Classify data according to the line $x_2 = x_1(x_1 - 1)(x_1 - 2)$
Perform perceptron algorithm on data set (return ERM hypothesis (h))
$\mathcal{H}_q$ *append* h
$testErrorTotal \leftarrow testErrorTotal + currentTestError$
end for

$\hat{R}_n(\texttt{averageTestError}) \leftarrow \frac{testErrorTotal}{repetitions}$
Calculate: $currentBound = \hat{R}_n + \Omega(n, class_weighting, \delta, d_{vc}(\mathcal{H}))$

if $currentBound \leq minimumBound$ **then**
$minimumBound = currentBound$
$SRMBestClassifier = q$
$g = \mathcal{H}_q$
end if

end for

Calculate Test Error for data set of 10,000,000 points

end procedure

The variable **repetitions** is set to 100 in the code that is provided along side this report. 100 repetitions was judged to be large enough to determine an adequate average value for the training error to acquire the SRM hypothesis class.

Furthermore, the values for **class_weighting** and δ are varied across various test to determine the effect that varying weights has. In all situations, it is ensured that $\sum_{w_i} \leq 1$. Furthermore, as SRM states a bound of $R_n \leq \hat{R_n} + \Omega(n, \mathcal{H}(h), w_i\delta)$ with a probability of $1 - w_i\delta$, which was always 0.9 for the tests ran for this report. As a result, δ is always calculated as a relative value compared to w_i as $\delta = 0.1/w_i$.

5.2.1 Results

In the following section, the results from running various tests calculating the SRM Bound, the best classifier according to the SRM as well as the training and test errors for the SRM solution. The method of calculating and achieving is shown in Section 5.2.

In all of the tables that follow, each column represents the run number or the average of all three runs. The values represented are the result of running 100 repetitions (see Algorithm 1 for reference to repetitions). The "Average" column takes the average values of three of such runs. For this entire coursework, three runs of three various setups were run, for 10,000 data points, a iteration limit of 10,000 for the perceptron algorithm, with 100 repetitions (see Algorithm 1 for reference to repetitions). Doing just one run takes roughly 3.8 hours, even with writing this in C++ and utilizing various optimization strategies. This due to the fact that because of the introduction of noise, the iteration limit will always be reached making the program run for the longest

possible amount of time. It was possible to run all of these tests by simultaneously sshing into nine different Department of Computing computers and running all test concurrently. There is a possible to improve the speed which is utlizing various available libraries to link C++ to Erlang and distribute the workload, but due to time and workload constrictions this was not implemented here.

All test error calculations for both ERM and SRM test error were done on 10,000,000 randomly generated points to get a test error that is as accurate as possible. All the tables containing the test results will be presented and then the presented data will be discussed.

	Run 1	Run 2	Run 3	Average
SRM	3.46675	3.46175	3.45575	3.461417
Best Classifier (q)	0	0	0	0
Training Error	0.143	0.138	0.132	0.137667
Test Error	0.270988	0.288334	0.259091	0.272804

Table 1: Data for 100 repeats, 10,000 perceptron iterations, 10 data points, 10,000,000 test error points

	Run 1	Run 2	Run3	Average
SRM	1.44604	1.44274	1.43984	1.442873
Best Classifier (q)	0	0	0	0
Training Error	0.2152	0.2119	0.209	0.212033
Test Error	0.238543	0.229536	0.237668	0.235249

Table 2: Data for 100 repeats, 10,000 perceptron iterations, 100 data points, 10,000,000 test error points

	Run 1	Run 2	Run 3	Run Average
SRM	0.375748	0.369195	0.371441	0.372128
Best Classifier (q)	0	0	0	0
Training Error	0.223058	0.216505	0.218751	0.219438
Test Error	0.185437	0.204823	0.208556	0.199605

Table 3: Data for 100 repeats, 10,000 perceptron iterations, 10,000 data points, 10,000,000 test error points

As is demonstrated by Tables 1, 2 and 3, it is clear that the SRM calculation is being overpowered by the complexity penalty. All dataset sizes calculated that a 0 degree polynommial would be fit at 3 degere polynomial. With the advantage of knowing what the classification line is, this is clearly a horrifically incorrect assumption. As is stated in Lecture 4 Slide 15 and 17, $test\ error \leq bias + complexity\ penalty$. The complexities for $q = 0$ are as followed:

- For a dataset of 10: 3.323749
- For a dataset of 100: 1.230840
- For a dataset of 10,000: 0.152690

From these complexity terms it becomes clear the colossal effect that the complexity term can have. For a dataset of 10 points, the test errors were more than twice as large as teh training errors and the complexity term is 24.14 times larger than the average training error. Such a large complexity penalty will, in all cases, make the result of the SRM calculation less meaningful as it is overshadowed by the colossal value of the complexity penalty. Furthermore, having a training dataset of 10 will always lead to inaccurate training results with a low training error as the points present will not accurately represent the line that is sought to be acquired. This discrepancy is further reinforced by the large gap between the training and test error shown in Table 1.

For a dataset of 100 points, the training error does represent the test error to a larger degree as demonstrated by Table 2. The difference between training and test error is no longer a factor of two, as the dataset size is able to provide constraints for the line to a far greater degree leading to a more realistic training scenario. Furthermore the complexity is now only roughly 5.81 times larger than the training error. Even though this is more reasonable than 24.14 times, it is still

definitely high enough to overpower the overall SRM calculation leading to the incorrect result for the best classifier.

Finally for a dataset of 10,000 points, the training error on average is very close to the test error, as can be seen in Table 3. This can be attributed to the fact that having a dataset of 10,000 points will allow for a dense generation of points (especially considering the constrained space of the question). Furthermore the complexity penalty is now only 0.7 times the training error. Although this factor difference is clearly preferable to the what has been demonstrated for the other dataset sizes, the range of the complexity values is still greater than the range of training error values produced for each values of q. For evidence, please refer to Table 7 and the raw data presented in Appendix A.

q value	Run 1	Run 2	Run 3	Run Average
0	0.274718	0.286943	0.258539	0.2734
1	0.275937	0.283631	0.275746	0.278438
2	0.291211	0.293083	0.276435	0.28691
3	0.274307	0.316263	0.299518	0.296696
4	0.324403	0.309887	0.302943	0.312411

Table 4: ERM test errors for 100 repeats, 10,000 perceptron iterations, 10 data points, 10,000,000 test error points

q value	Run 1	Run 2	Run3	Run Average
0	0.244418	0.23648	0.236646	0.239181
1	0.226741	0.230184	0.222295	0.226407
2	0.228141	0.235715	0.220402	0.228086
3	0.215973	0.218425	0.216341	0.216913
4	0.222857	0.234444	0.215769	0.224357

Table 5: ERM test errors for 100 repeats, 10,000 perceptron iterations, 100 data points, 10,000,000 test error points

q value	**Run 1**	**Run 2**	**Run 3**	**Run Average**
0	0.243635	0.217495	0.227108	0.229413
1	0.23022	0.220847	0.220855	0.223974
2	0.193995	0.205809	0.235959	0.211921
3	0.197715	0.198874	0.202846	0.199812
4	0.189537	0.206473	0.19132	0.195777

Table 6: ERM test errors for 100 repeats, 10,000 perceptron iterations, 10,000 data points, 10,000,000 test error points

Beyond just looking at the results produced directly by the SRM calculations, the test error for all the ERM solutions was recorded for each value of q as is displayed in Tables 4, 5 and 6. This data is able to further reinforce the analysis made regarding the effectiveness of the training and help in decided the best dataset size to use for training.

When looking at Table 4, it is clear that even though the complexity penalty was extremely high in regards to the training error, using a data set of 10 point is clearly insufficient. When looking at the average value from all three runs for the ERM test error, it shows that $q = 0$ had the lowest ERM test error. This indicates that with so little data points, all other polynomial degree hypotheses were so inaccurate, a 0 degree polynomial hypothesis manages to best represent at third degree polynomial line. In a situation with such results, the perceptron algorithm is clearly not able to function properly.

Table 5 demonstrates the more realistic results when using a training dataset size of 100. The ERM test error decreases as q heads towards 3 and increases again afterwards. This is exactly the behavior that is expected considering the line used to classify the data.

Table 3 demonstrates results that are less favorable than Table 2. For 100 data points $q = 3$ had the best fitting line, whereas for 10,000 data points $q = 4$ seems to have the best possible training error. Due to the very large quantity of data points, is has clearly become possible to very narrowly fit a quartic degree polynomial hypothesis to a cubic line. Even though it is expected that the more datapoints that are present, the more accurate the hypothesis, but considering there is a 10% noise present, there has clearly been a skew towards $q = 4$. Averaging these values with those found in Tables 8 and 9 the data does select $q = 3$ on average, but this is not a guarantee. In a situation where the complexity factor is less overpowering (see Section 5.3), the complexity factor will help guide the SRM to pick the correct q degree polynomial hypothesis.

From the analysis from the data produced using a normal complexity factor, it has become clear that when a hypothesis is to be selected for a noisy data set there are multiple factors to consider when choosing the training dataset size. Clearly the computational power must be considered to ensure enough reliable data can be collected while still being able to run within any time constraints present. Furthermore, it became clear that having a data set size that is too low (10 in this situation) has terrible results and should never be considered as none of the produced data is representative of what is expected. Having a data set that is too large clearly had a slight negative impact as well as was demonstrated by the ERM test error for a dataset of 10,000 points. On the other hand, when performing SRM calculations, it is clear that a larger dataset provides a far better complexity value providing more reasonable and reliable results. The misprediction seen in the ERM test error results for very large datasets is compensated for when applying a proper scaling factor the the complexity term. This leads to the conclusion that the dataset size selection is a trade off, but when performing SRM calculations, larger sets are the only ones that provide reasonable complexity values, and any ERM related inaccuracy are compensated for.

In the hint for this question, it is stated that it may be useful to utilize the bound provided in Problem 4b. Upon analysis of this hint, calculations were done to analyze the effect of using that bound would have upon the total complexity value and the overall effect on the SRM solutions. Due to the insignificant change this introduced, the decision as made to stay with the original equation as this provided computational simplicity and provided a greater degree of accuracy.

5.3 Part c

All the tests for this section were only performed on the dataset size of 10,000 points, as this dataset had the most reasonable complexity penalty to use as a baseline.

Problem 5c) addressed a crucial issue that arose in Section 5.2. In this section, it was demonstrated that even though the lowest training error was always for the hypothesis classes of $\mathcal{H}_3$ or $\mathcal{H}_4$ for reasonable test error situations, the structural risk minimizer (SRM) always turned out to be within the hypothesis class $\mathcal{H}_0$. This is very counterintuitive as the $\mathcal{H}_0$ hypothesis class had, in no way, the lowest training error. After analysis of the formula for SRM, it became clear that the SRM complexity value produces a lower value as q gets smaller for $\mathcal{H}_q$, as is shown in Table 7:

hypothesis class	complexity	complexity $*$ 0.1	complexity $*$ 0.01
$\mathcal{H}_0$	0.0.152689898	0.01526899	0.001526899
$\mathcal{H}_1$	0.178729771	0.017872977	0.001787298
$\mathcal{H}_2$	0.201189494	0.020118949	0.002011895
$\mathcal{H}_3$	0.221234023	0.022123402	0.00221234
$\mathcal{H}_4$	0.239507113	0.023950711	0.002395071

Table 7: SRM Complexity per Hypothesis Class $\mathcal{H}$

To attempt to compensate for the effect that the complexity term had upon the results is demonstrated below.

The tables below are of the same style as in Section 5.2.1. For any clarification regarding their output, please refer back to Section 5.2.1. The data below has only be done for

- sample size $= 10,000$
- perceptron iterations $= 10,000$
- repetitions $= 100$ (see Algorithm 1 for reference to repetitions)

Furthermore, for further details of the results from any of the run see Appendix A (Section 6).

5.3.1 Results

Run	1	2	3	Average
Complexity Mult	0.1	0.1	0.1	0.1
SRM	0.204514	0.2111594	0.215203	0.210292
Best Classifier (q)	3	3	3	3
Training Error	0.182391	0.189036	0.19308	0.188169
Test Error	0.178752	0.185911	0.190955	0.185206

Table 8: Data for a complexity multiplier of 0.1, 100 repeats, 10,000 perceptron iterations, 10,000 data points, 10,000,000 test error points

Run	1	2	3	Average
Complexity Mult	0.01	0.01	0.01	0.01
SRM	0.204677	0.195961	0.197486	0.199375
Best Classifier (q)	3	3	3	3
Training Error	0.202465	0.193749	0.195274	0.197163
Test Error	0.207256	0.18766	0.198494	0.197803

Table 9: Data for a complexity multiplier of 0.01, 100 repeats, 10,000 perceptron iterations, 10,000 data points, 10,000,000 test error points

Tables 3, 8 and 9 demonstrate the performance of various complexity multipliers. As was shown in Section 5.2.1 when using the generically calculated complexity multiplier the SRM calculation is too heavily affected by the complexity penalty. Tables 8 and 9 demonstrate the positive effect decreasing the complexity penalty has. As already mentioned in Section 5.2.1, the ERM calculations may not always pick the correct degree polynomial hypothesis, but with an adequate complexity term, this problem is taken care of. Section 5.2.1 mentions that the range of the complexity penalty values is too large causing the data to be distorted in towards $q = 0$. As can be seen by changing the value the complexity penalty is multiplied by decreases this range and provides the SRM solutions that are expected.

The function of the SRM is to skew the data towards a more favorable decision. In this situation the complexity value cause the SRM procedure to select the optimal hypothesis class. With the data that has been collected there has been no need for this as the test error produced by the ERM has already been ideal. But considering as this is not always the case, performing SRM calculations with a scaled complexity value will ensure more reliable and favorable results

By utilizing a factor of 0.1 or 0.01 it becomes possible to selected a large dataset for training and acquire accurate results. This is reinforced by the extremely small differences between the training and test errors that are seen in Tables 8 and 9 and the fact that we have acquired the correct polynomial for the hypothesis class. As mentioned in the hint, this is due to the fact that the VC bound (using the bound in 4b) or not) is quite loose, some scaling must be done ensure this bound can be empirically tightened.

Run	1	2	3	Average
0	0.226328	0.211574	0.218599	0.218834
1	0.209926	0.213277	0.221149	0.214784
2	0.204419	0.215058	0.243632	0.221036
3	0.190746	0.187599	0.198089	0.192145
4	0.221393	0.218358	0.205556	0.215102

Table 10: ERM test errors for 100 repeats, 10,000 perceptron iterations, 10,000 data points, 10,000,000 test error points

Run	1	2	3	Average
0	0.219998	0.231349	0.225206	0.225518
1	0.22661	0.21736	0.214569	0.219513
2	0.216422	0.201228	0.216819	0.21149
3	0.216361	0.200705	0.199585	0.20555
4	0.21538	0.229996	0.211551	0.218976

Table 11: ERM test errors for 100 repeats, 10,000 perceptron iterations, 10,000 data points, 10,000,000 test error points

In Lecture 4 Slide 16, it was mentioned that the weights can be changed to favor specific values of q. Strictly speaking, altering the calculation procedure because some part of the result is known is technically data snooping. But if the weights are changed, the imbalance produced by lower values of q having a lower complexity term is counteracted, and higher polynomial hypotheses are favored. In the situations seen so far this is definitely a favorable result. This can be show by editing src/main.cpp and changing the "weights" vector initialization.

6 Appendix A

10

Run 1	Run 2	Run 3	Average	
SRM	4.73215	4.71615	4.72815	4.725483
Complexity	4.68615	4.68615	4.68615	4.68615
Training	4.6E-2	0.03	4.2E-2	3.9E-2
ERM	0.324403	0.309887	0.302943	0.312411

Table 12: Data for 100 repeats, 10,000 perceptron iterations, 100 data points, 10,000,000 test error points

Run 1	Run 2	Run 3	Average	
SRM	4.41774	4.44974	4.43174	4.433073
Complexity	4.38974	4.38974	4.38974	4.38974
Training - 3	2.8E-2	0.06	4.2E-2	4.333333E-2
ERM - 3	0.274307	0.316263	0.299518	0.296696

Table 13: Data for 100 repeats, 10,000 perceptron iterations, 100 data points, 10,000,000 test error points

Run 1	Run 2	Run 3	Average	
SRM - 2	4.13126	4.11726	4.11626	4.121593
Complexity - 2	4.06926	4.06926	4.06926	4.06926
Training - 2	6.2E-2	4.8E-2	4.7E-2	5.233333E-2
ERM - 2	0.291211	0.293083	0.276435	0.2869097

Table 14: Data for 100 repeats, 10,000 perceptron iterations, 100 data points, 10,000,000 test error points

Run 1	Run 2	Run 3	Average	
SRM - 1	3.8017	3.8107	3.8067	3.8063667
Complexity - 1	3.7177	3.7177	3.7177	3.7177
Training - 1	8.4E-2	9.3E-2	8.9E-2	8.866667E-2
ERM - 1	0.275937	0.283631	0.275746	0.278438

Table 15: Data for 100 repeats, 10,000 perceptron iterations, 100 data points, 10,000,000 test error points

Run 1	Run 2	Run 3	Average	
SRM - 0	3.46675	3.46175	3.45575	3.461417
Complexity - 0	3.32375	3.32375	3.32375	3.32375
Training - 0	0.143	0.138	0.132	0.137667
ERM - 0	0.274718	0.286943	0.258539	0.2734

Table 16: Data for 100 repeats, 10,000 perceptron iterations, 100 data points, 10,000,000 test error points

100

Run 1	Run 2	Run3	Average	
SRM - 4	2.0196200000000002	2.01762	2.0028199999999998	2.0133533333333333
Complexity - 4	1.83762	1.83762	1.83762	1.83762
Training - 4	0.182	0.18	0.16520000000000001	0.17573333333333332
ERM - 4	0.222857	0.23444400000000001	0.21576899999999999	0.22435666666666668

Table 17: Data for 100 repeats, 10,000 perceptron iterations, 10,000 data points, 10,000,000 test error points

Run 1	Run 2	Run3	Average	
SRM - 3	1.8783300000000001	1.88093	1.8775299999999999	1.8789300000000002
Complexity - 3	1.7080299999999999	1.7080299999999999	1.7080299999999999	1.7080299999999999
Training - 3	0.17030000000000001	0.1729	0.16950000000000001	0.17090000000000002
ERM - 3	0.215973	0.21842500000000001	0.21634100000000001	0.21691299999999999

Table 18: Data for 100 repeats, 10,000 perceptron iterations, 10,000 data points, 10,000,000 test error points

Run 1	Run 2	Run3	Average	
SRM - 2	1.7503599999999999	1.74776	1.74526	1.7477933333333333
Complexity - 2	1.5667599999999999	1.5667599999999999	1.5667599999999999	1.5667599999999997
Training - 2	0.18360000000000001	0.18099999999999999	0.17849999999999999	0.18103333333333335
ERM - 2	0.22814100000000001	0.23571500000000001	0.22040199999999999	0.22808600000000001

Table 19: Data for 100 repeats, 10,000 perceptron iterations, 10,000 data points, 10,000,000 test error points

Run 1	Run 2	Run3	Average	
SRM - 1	1.59762	1.60812	1.5972200000000001	1.6009866666666668
Complexity - 1	1.4099200000000001	1.4099200000000001	1.4099200000000001	1.4099200000000003
Training - 1	0.18770000000000001	0.19819999999999999	0.18729999999999999	0.19106666666666669
ERM - 1	0.226741	0.230184	0.22229499999999999	0.22640666666666667

Table 20: Data for 100 repeats, 10,000 perceptron iterations, 10,000 data points, 10,000,000 test error points

Run 1	Run 2	Run3	Average	
SRM - 0	1.44604	1.4427399999999999	1.43984	1.4428733333333332
Complexity - 0	1.2308399999999999	1.2308399999999999	1.2308399999999999	1.2308399999999999
Training - 0	0.2152	0.21190000000000001	0.20899999999999999	0.21203333333333332
ERM - 0	0.244418	0.23648	0.236646	0.23918133333333333

Table 21: Data for 100 repeats, 10,000 perceptron iterations, 10,000 data points, 10,000,000 test error points

Run	1	2	3	Average
SRM	0.42226	0.434312	0.440413	0.432328
Complexity	0.239507	0.239507	0.239507	0.239507
Training	0.182753	0.194805	0.200906	0.192821
ERM	0.189537	0.206473	0.19132	0.195777

Table 22: $Q = 4$ Data for 100 repeats, 10,000 perceptron iterations, 10,000 data points, 10,000,000 test error points

Run	1	2	3	Average
SRM	0.413385	0.418004	0.422405	0.417931
Complexity	0.221234	0.221234	0.221234	0.221234
Training	0.192151	0.19677	0.201171	0.196697
ERM	0.197715	0.198874	0.202846	0.199812

Table 23: $Q = 3$ Data for 100 repeats, 10,000 perceptron iterations, 10,000 data points, 10,000,000 test error points

Run	1	2	3	Average
SRM	0.408037	0.402375	0.40465	0.405021
Complexity	0.201189	0.201189	0.201189	0.201189
Training	0.206848	0.201186	0.203461	0.203832
ERM	0.193995	0.205809	0.235959	0.211921

Table 24: $Q = 2$, Data for 100 repeats, 10,000 perceptron iterations, 10,000 data points, 10,000,000 test error points

Run	1	2	3	Average
SRM	0.406332	0.390856	0.389646	0.395611
Complexity	0.17873	0.17873	0.17873	0.17873
Training	0.227602	0.212126	0.210916	0.216881
ERM	0.23022	0.220847	0.220855	0.223974

Table 25: $Q = 1$ Data for 100 repeats, 10,000 perceptron iterations, 10,000 data points, 10,000,000 test error points

Run	1	2	3	Average
SRM	0.375748	0.369195	0.371441	0.372128
Complexity	0.15269	0.15269	0.15269	0.15269
Training	0.223058	0.216505	0.218751	0.219438
ERM	0.243635	0.217495	0.227108	0.229413

Table 26: $Q = 0$ Data for 100 repeats, 10,000 perceptron iterations, 10,000 data points, 10,000,000 test error points

Run	1	2	3	Average
SRM	0.218912	0.219631	0.225122	0.221221
Complexity	0.239507	0.239507	0.239507	0.239507
Training	0.194961	0.19568	0.201171	0.197271
ERM	0.221393	0.218358	0.205556	0.215102

Table 27: $Q = 4$ Data for 100 repeats, 10,000 perceptron iterations, 10,000 data points, 10,000,000 test error points

Run	1	2	3	Average
SRM	0.204514	0.2111594	0.215203	0.210292
Complexity	0.221234	0.221234	0.221234	0.221234
Training	0.182391	0.189036	0.19308	0.188169
ERM	0.190746	0.187599	0.198089	0.192145

Table 28: $Q = 3$ Data for 100 repeats, 10,000 perceptron iterations, 10,000 data points, 10,000,000 test error points

Run	1	2	3	Average
SRM	0.2290269	0.2292979	0.230668	0.229664
Complexity	0.201189	0.201189	0.201189	0.201189
Training	0.208908	0.209179	0.210549	0.209545
ERM	0.204419	0.215058	0.243632	0.221036

Table 29: $Q = 2$ Data for 100 repeats, 10,000 perceptron iterations, 10,000 data points, 10,000,000 test error points

Run	1	2	3	Average
SRM	0.226556	0.232427	0.238675	0.232553
Complexity	0.17873	0.17873	0.17873	0.17873
Training	0.208683	0.214554	0.220802	0.21468
ERM	0.209926	0.213277	0.221149	0.214784

Table 30: $Q = 1$ Data for 100 repeats, 10,000 perceptron iterations, 10,000 data points, 10,000,000 test error points

Run	1	2	3	Average
SRM	0.229335	0.224123	0.233949	0.229136
Complexity - 0	0.15269	0.15269	0.15269	0.15269
Training - 0	0.214066	0.208854	0.21868	0.213867
ERM - 0	0.226328	0.211574	0.218599	0.218834

Table 31: $Q = 0$ Data for 100 repeats, 10,000 perceptron iterations, 10,000 data points, 10,000,000 test error points

Run	1	2	3	Average
SRM	0.205887	0.218577	0.211946	0.212137
Complexity	0.239507	0.239507	0.239507	0.239507
Training	0.203492	0.216182	0.209551	0.209742
ERM	0.21538	0.229996	0.211551	0.218976

Table 32: $Q = 4$ Data for 100 repeats, 10,000 perceptron iterations, 10,000 data points, 10,000,000 test error points

Run	1	2	3	Average
SRM	0.204677	0.195961	0.197486	0.199375
Complexity	0.221234	0.221234	0.221234	0.221234
Training	0.202465	0.193749	0.195274	0.197163
ERM	0.216361	0.200705	0.199585	0.20555

Table 33: $Q = 3$ Data for 100 repeats, 10,000 perceptron iterations, 10,000 data points, 10,000,000 test error points

Run	1	2	3	Average
SRM	0.210147	0.201555	0.207416	0.206373
Complexity	0.201189	0.201189	0.201189	0.201189
Training	0.208135	0.199543	0.205404	0.204361
ERM	0.216422	0.201228	0.216819	0.21149

Table 34: $Q = 2$ Data for 100 repeats, 10,000 perceptron iterations, 10,000 data points, 10,000,000 test error points

Run	1	2	3	Average
SRM	0.21456	0.215534	0.20752	0.212538
Complexity	0.17873	0.17873	0.17873	0.17873
Training	0.212773	0.213747	0.205733	0.210751
ERM	0.22661	0.21736	0.214569	0.219513

Table 35: $Q = 1$ Data for 100 repeats, 10,000 perceptron iterations, 10,000 data points, 10,000,000 test error points

Run	1	2	3	Average
SRM	0.221362	0.227078	0.225777	0.224739
Complexity	0.15269	0.15269	0.15269	0.15269
Training	0.219835	0.225551	0.22425	0.223212
ERM	0.219998	0.231349	0.225206	0.225518

Table 36: $Q = 0$ Data for 100 repeats, 10,000 perceptron iterations, 10,000 data points, 10,000,000 test error points

References

D D. Bhattacharjee. *Reduction of Feature Vectors Using Rough Set Theory For Human Face Recognition.*

Dunja Mladenic. *Feature Selection using Linear Classifier Weights: Interaction with Classification Models* .

Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning.* Cambridge University Press, 2014.