Imperial College
London

Machine Learning Coursework 2

Youssef Rizk (00940962)

Examined by
Dr. András György

I, Youssef Rizk, 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.

February 20, 2017

Contents

1 Problem 1

The first problem considers a binary classification context, in which a finite hypothesis set, $\mathcal{H}$, includes a perfect hypothesis, h^*, such that any $x \in \mathcal{X}$ has label $h^*(x)$. The task is to show that with probability $1-\delta$, where $\delta \in (0,1)$, and any ERM hypothesis, g, selected on n i.i.d. points, the test error of g satisfies $R(g) \leq \frac{\log(|\mathcal{H}|/\delta)}{n}$. We now begin to show this.

We first note that g is defined such that $g = \underset{h\in\mathcal{H}}{\operatorname{argmin}} \hat{R_n}(h)$, where $\hat{R_n}$ is the training error for a particular hypothesis. Bearing in mind this definition, it becomes easy to show that $\hat{R_n}(g) = 0$. This follows from the fact that g assumes a hypothesis that achieves the lowest training error. We know that h^* exists in $\mathcal{H}$, and h^* correctly classifies all data points, so there is at least one hypothesis that achieves a zero training error, and g will at least assume that hypothesis. In order for g to assume any other hypothesis, the other hypothesis must also achieve a training error of zero, or else g would have selected the perfect hypothesis h^*. Thus, the probability that the ERM assumes a hypothesis h that achieves a poor test error (i.e. $R(h) > \epsilon > 0$) is bounded by the probability that for any $h \in \mathcal{H}$, $\hat{R_n}(h) = 0$, i.e. $\mathbb{P}(\exists h \in \mathcal{H} \wedge \hat{R_n}(h) = 0)$.

Now, if we consider a given hypothesis $h \in \mathcal{H}$ with $R(h) > \epsilon > 0$, then by definition, h makes an error in classifying a data point with probability $R(h)$. In other words, h does not make an error in classifying a data point with probability $1 - R(h)$. For n i.i.d. data points and the fact that $R(n) > \epsilon$, the probability that h does not make an error in classifying any of them then becomes $\mathbb{P}(\hat{R_n}(h) = 0) = (1 - R(h))^n \leq (1-\epsilon)^n$. Using the inequality $(1-\epsilon)^n \leq e^{-\epsilon n}$, we can bound the previous probability as $\mathbb{P}(\hat{R_n}(h) = 0) \leq (1-\epsilon)^n \leq e^{-\epsilon n}$. Now, if we apply the union bound to the previously obtained probability, namely $\mathbb{P}(\exists h \in \mathcal{H} \wedge \hat{R_n}(h) = 0)$, we obtain the following:

$$\mathbb{P}(\exists h \in \mathcal{H} \wedge \hat{R_n}(h) = 0) \leq \sum_{h\in\mathcal{H}} \mathbb{P}(\hat{R_n}(h) = 0)$$
$$\sum_{h\in\mathcal{H}} \mathbb{P}(\hat{R_n}(h) = 0) \leq |\mathcal{H}| e^{-\epsilon n}$$
$$\overset{\text{def}}{=} \delta$$

If we now rearrange this and solve for ϵ, we obtain that with probability δ:

$$\epsilon = \frac{\log(|\mathcal{H}|/\delta)}{n}$$
$$R(h) > \frac{\log(|\mathcal{H}|/\delta)}{n}$$

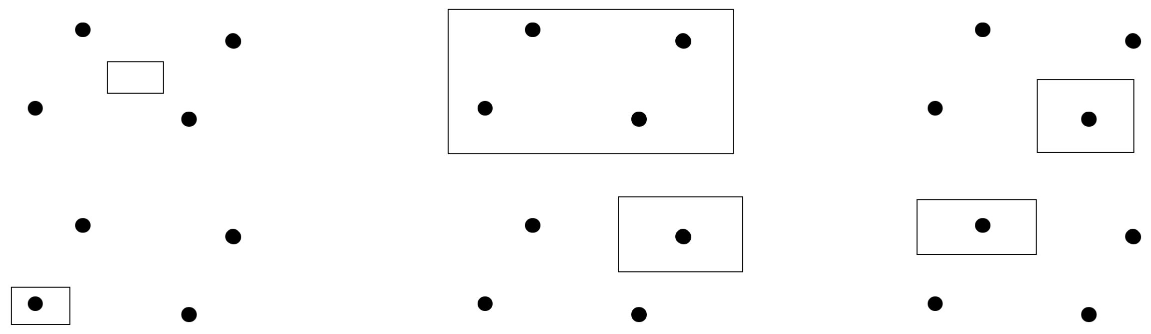
Thus, with probability $1-\delta$:

$$R(h) \leq \frac{\log(|\mathcal{H}|/\delta)}{n}$$

2 Problem 2

This problem examines axis -aligned rectangles. Specifically, consider any real numbers $a_1 \leq a_2$ and $b_1 \leq b_2$ and let

$$R(a_1, a_2, b_1, b_2) = \{(x_1, x_2) \in \mathbb{R}^2 : a_1 \leq x_1 \leq a_2 \text{ and } b_1 \leq x_2 \leq b_2\}$$



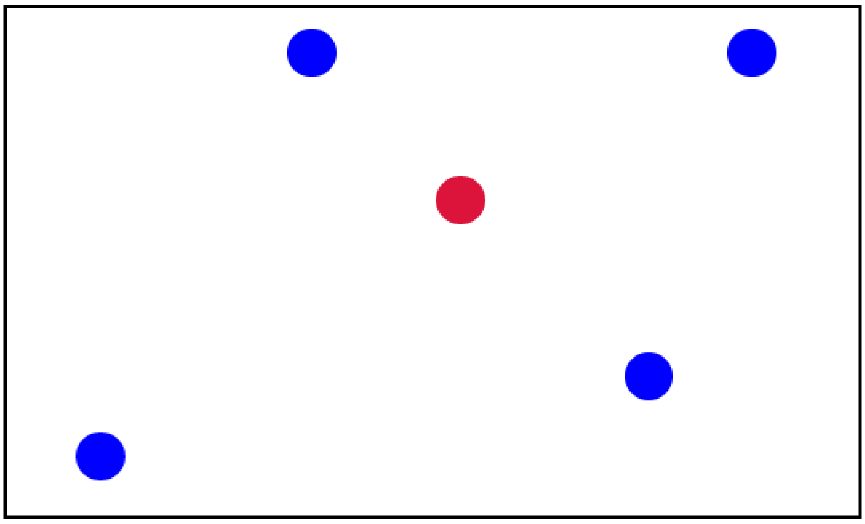
Dichotomies of 4 points

denote an axis-aligned rectangle. Also consider the class of indicator functions of such rectangles:

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

We would like to show that the VC dimension of $\mathcal{H}$ is 4. To show this, I have added several graphics to illustrate my point. We specifically show this in two steps, firstly showing that the VC dimension ≥ 4 and secondly that the VC dimension < 5. We begin with a 4-point example where the points are arranged as shown in figure 1, which shows a subset of the possible dichotomies. It becomes clear that 4 points can be shattered. Since the VC dimension is a measure of the largest number of points that a hypothesis class can shatter, we know that this is now lower bounded by 4 (i.e. VC dimension ≥ 4), since we have just seen that it is possible to shatter 4 points.

When we repeat the same example with 5 points, we specifically look at the arrangement in which the leftmost, rightmost, highest and lowest points have the same label (1 which is here shown in blue), and the remaining point has a different label (0 which is here shown in red). This can be illustrated as in figure 2. Immediately we notice that if we want to group the 4 extreme points using one rectangle, then the middle point with a differing label will also be included, and thus, it is not possible to shatter a data set containing 5-points for $\mathcal{H}$. This shows that the VC dimension < 5. Trivially then, the VC dimension must be equal to 4.



Dichotomies of 5 points

3 Problem 3

It is most illustrative to begin this question by examining a simple case and noting what key features it has. If we consider 5 points and a hypothesis class which is characterized by the below dichotomies:

$$\begin{gathered}[-1,-1,-1,-1,-1]\\ [1,-1,-1,-1,-1]\\ [-1,1,-1,-1,-1]\\ [-1,-1,1,-1,-1]\\ [-1,-1,1,-1,-1]\\ [-1,-1,1,-1,-1]\end{gathered}$$

We can see that in this situation, this hypothesis class does not shatter 2 points. This can be explicitly seen by the fact that no two points in a hypothesis have a label of 1. We also see that $|\mathcal{H}(x_1,...,x_n)| = 6$, which is also equal to $\sum_{i=0}^{k-1}\binom{n}{i}$, where $k = 2$ in this case.

4 Problem 4

4.1 Part A

I proceed to prove this bound by employing Sauer's Lemma, namely that $m_{\mathcal{H}}(n) \leq \sum_{i=0}^{d}\binom{n}{i}$. My goal is to show that $\sum_{i=0}^{d}\binom{n}{i} \leq n^{d_{VC}} + 1$. I assume here that $d_{VC} > 0$.

Let $n = 1$, then

$$\sum_{i=0}^{d_{VC}}\binom{n}{i} \leq n^{d_{VC}} + 1 \rightarrow 2 \leq 2$$

Let $n = k$. Assume

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

We now want to show that

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

To do this, we complete this proof over two cases: $d_{VC} = 1$ & $d_{VC} > 1$. For $d_{VC} = 1$,

$$\begin{gathered}\text{LHS evaluates to:}\sum_{i=0}^{d_{VC}}\binom{k+1}{i} = (k+1)^1 + 1\\ \text{RHS evaluates to:}(k+1)^{d_{VC}} + 1 = (k+1)^1 + 1\\ \text{Thus, LHS} \leq \text{RHS } \forall k.\end{gathered}$$

We now consider the second case. For this, we rewrite the LHS

$$\begin{aligned}
\sum_{i=0}^{d_{VC}} \binom{k+1}{i} &= 1 + \sum_{i=1}^{d_{VC}} \binom{k+1}{i} \\
&= 1 + \sum_{i=1}^{d_{VC}} \binom{k}{i} + \sum_{i=1}^{d_{VC}} \binom{k}{i-1} \\
&= \sum_{i=0}^{d_{VC}} \binom{k}{i} + \sum_{i=1}^{d_{VC}} \binom{k}{i-1} \\
&= \sum_{i=0}^{d_{VC}} \binom{k}{i} + \sum_{i=0}^{d_{VC}-1} \binom{k}{i} \\
&\leq k^{d_{VC}} + 1 + k^{d_{VC}-1} + 1
\end{aligned}$$

We want to show that $k^{d_{VC}} + 1 + k^{d_{VC}-1} + 1 \leq (k+1)^{d_{VC}} + 1$, or simply $k^{d_{VC}} + k^{d_{VC}-1} + 1 \leq (k+1)^{d_{VC}}$. We can rewrite the RHS as

$$\begin{aligned}
(k+1)^{d_{VC}} &= \sum_{i=0}^{d_{VC}} \binom{d_{VC}}{i} k^i \\
&= k^{d_{VC}} + d_{VC} k^{d_{VC}-1} + ... + 1
\end{aligned}$$

Since $d_{VC} > 1$ & $k > 0$,

$$\begin{aligned}
k^{d_{VC}} + k^{d_{VC}-1} + 1 &\leq k^{d_{VC}} + d_{VC} k^{d_{VC}-1} + ... + 1 \\
\therefore \sum_{i=0}^{d_{VC}} \binom{k+1}{i} &\leq (k+1)^{d_{VC}} + 1
\end{aligned}$$

It is trivial to show that $n^{d_{VC}} + 1 \leq (n+1)^{d_{VC}}$, and so, $m_{\mathcal{H}}(n) \leq \sum_{i=0}^{d_{VC}} \binom{n}{i} \leq (n+1)^{d_{VC}}$.

4.2 Part B

For this part of the problem, we are to prove that for all $n \geq d_{VC}$, the following inequality holds: $m_{\mathcal{H}}(n) \leq (\frac{ne}{d_{VC}})^{d_{VC}}$. This can be most easily shown by employing Sauer's Lemma as follows:

$$\begin{aligned}
\sum_{i=0}^{d} \binom{n}{i} &= \left(\frac{n}{d}\right)^d \sum_{i=0}^{d} \binom{n}{i} \left(\frac{d}{n}\right)^d && \text{Rewriting Sauer's Lemma} \\
&\leq \left(\frac{n}{d}\right)^d \sum_{i=0}^{d} \binom{n}{i} \left(\frac{d}{n}\right)^i && \text{Since } i \leq d \\
&\leq \left(\frac{n}{d}\right)^d \sum_{i=0}^{n} \binom{n}{i} \left(\frac{d}{n}\right)^i && \text{Since } n \geq d \\
&= \left(\frac{n}{d}\right)^d \left(1 + \frac{d}{n}\right)^n && \text{By the Binomial Theorem} \\
&\leq \left(\frac{n}{d}\right)^d e^d && \text{Using inequality } \left(1 + \frac{1}{x}\right)^x \leq e, \text{ where } x = \frac{n}{d}
\end{aligned}$$

4.3 Comparing the bounds

The bound from Part B turns out to be much tighter than that of Part A. However, this is because the one in Part B benefits from an added restriction on n, namely that $n \geq d$.

5 Problem 5

In this problem we examine a structural risk minimization context, where the input space is $\mathcal{X} \in \mathbb{R}^2$.

5.1 Part A

Initially, it is instructive to consider a classifier, $h_g : \mathcal{X} \to \{+1, -1\}$, where for any function g $h_g((x_1, x_2)) = 1$ if $x2 \geq g(x_1)$ or $h_g((x_1, x_2)) = -1$ if $x2 < g(x_1)$. Now if we consider the set of classifiers $\mathcal{H}_q = \{h_p | p : \mathbb{R} \to \mathbb{R}$ *is a polynomial of at most degree* $q\}$. Rather than applying the Perceptron to the raw data, it may be more useful to come up with a non-linear transformation to transform points in $\mathcal{X}$ to $\mathcal{Z}$. If we examine the polynomial p, we find that it can be written as

$$p(x) = x^q + a_{q-1}x^{q-1} + ... + a_1 x + a_0 \qquad \text{since it has at most degree } q$$

We notice of course that if $q > 1$, this polynomial ceases to be linear in x. However, we notice immediately, that it remains linear in $[x, x^2, x^3, ..., x^q]$. Thus we can make a non-linear transformation from $\mathcal{X}$ to $\mathcal{Z}$ by applying the following relation

$$(x_1, x_2) \to (1, x_1, x_1^2, x_1^3, ..., x_1^q, x_2)$$

This guarantees that any classifier h in $\mathcal{H}_q$ can be expressed as a linear classifier in $\mathcal{Z}$. We would also like to get a bound on the VC dimension of an individual hypothesis class, $\mathcal{H}_q$. It turns out that the VC dimension is at most $q + 1$. To show this, we notice that the weight corresponding to the x_2 feature is always 1. We can thus note that, effectively, we have $q + 1$ features, since we do not really learn the weight corresponding to x_2. Now, I want to show that we cannot shatter $q + 2$ points, which means that we can shatter at most $q + 1$ points. We can take $q + 2$ $q + 1$ dimensional points. Since there are more points than dimensions, we know that at least one of the points is linearly dependent on the other points. In other words,

$$x_{q+2} = \sum_{i=0}^{q+1} a_i x_i$$

Consider the dichotomy:

$$\begin{aligned} h(x_i) &= sign(a_i) \text{ for } 1 \leq i \leq d + 1, \text{ such that } sign(w^\top x_i) = sign(a_i) \\ h(x_{d+2}) &= -1 \end{aligned}$$

We can also rewrite

$$h(x_{d+2}) = sign(w^\top x_{d+2}) = sign(w^\top \sum_{i=0}^{q+1} a_i x_i) > 0$$

This is a contradiction. Thus, this shows that there is a dichotomy that we can never obtain, and so, we cannot shatter q+2 points.

5.2 Part B

To begin this exploration, I first generate random data points specified by the relationship given in the problem, using a MATLAB function called `generateRandPoints`, together with a classifier function to assign labels according to the probability distributions given to the generated points (`classify`) both of which are available in the appendix. I generate training data sets of size 10, 100, 1000, and 10000. I further modify my existing Perceptron algorithm to ensure that it always produces the ERM hypothesis.

The aim of this section then is to explore the SRM solution. The SRM solution finds the hypothesis within a hypothesis class $\mathcal{H} = \bigcup_{i=0}^{q} \mathcal{H}_i$ which minimizes a certain quantity. Specifically, the SRM solution, g, is defined as

$$g = \operatorname*{argmin}_{h \in \mathcal{H}} \hat{R}_n(h) + \Omega(n, \mathcal{H}_i, w_i\delta)$$

where

$$\Omega(n, \mathcal{H}_i, w_i\delta) = \sqrt{\frac{8d_{VC}}{n} \log \frac{2ne}{d_{VC}(\mathcal{H}_i)} + \frac{8}{n} \log \frac{4}{w_i\delta}}$$

To complement this analysis, I also determine the ERM solution for each of the constituent hypothesis classes, and plot their training and test error. Since these experiments are random in nature due to the random generation of data points, I run this script 100 times and take the average of the relevant quantities, such as test and training error for the ERM solutions and the SRM solution. I also plot the calculated classifier functions that are generated by each hypothesis class and compare them to our original function $f(x)$.

I first examine the results for 10-point training sets. For the various data sizes, we would expect the training error to be of the order of 10% since that is the noise level present within the set. My observations are based around the following points:

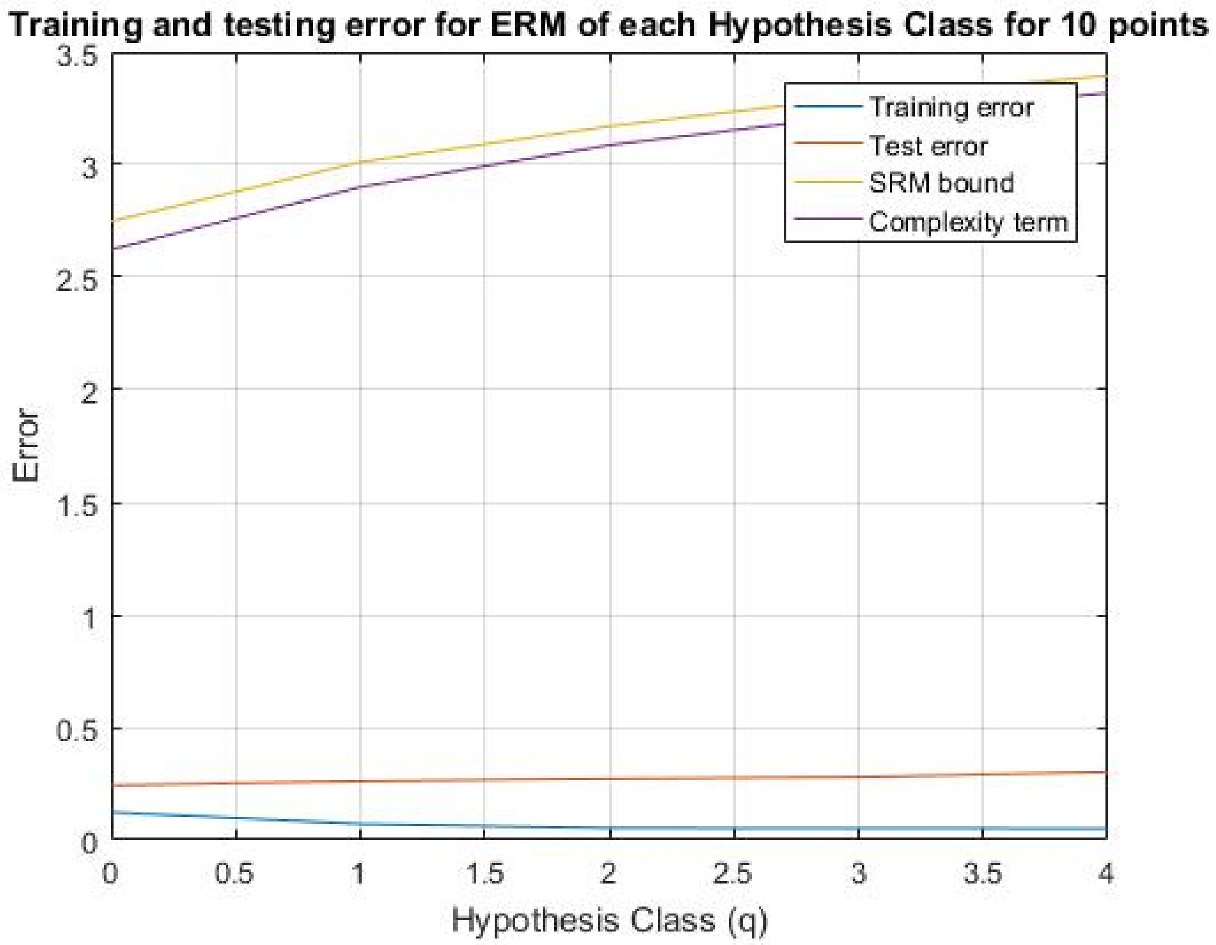
- Training and Test error of ERM solution for each Hypothesis Class
- Average Training and Test error for the SRM solution
- How often the SRM is selected from each Hypothesis Class
- The Average polynomial produced by each Hypothesis Class
- The polynomials produced by $\mathcal{H}_3$ & $\mathcal{H}_4$ versus $f(x)$

I employ a reasonable approximation for the test error. I create a sufficiently large test data set, and calculate the empirical test error, i.e. measuring the number of misclassified test points. This is done using the function `errorprob` again. Of course, there need to be certain restrictions on the number of data points in the test set that we will employ. These can be deduced from Hoeffding's inequality. I assume that I want a 90% confidence interval that the deviation between the empirical test error and the true test error is less than 0.01. Thus we require

$$2e^{-2(0.01)^2 n} = 0.1$$
$$n = 14979$$

Thus, to guarantee a 90% confidence interval that the deviation between the two test errors is no greater than 0.1, we require a test data set consisting of approximately 15,000 data points. In practice, I use 100,000 data points for better guarantees.

5.2.1 10-Point Training Set

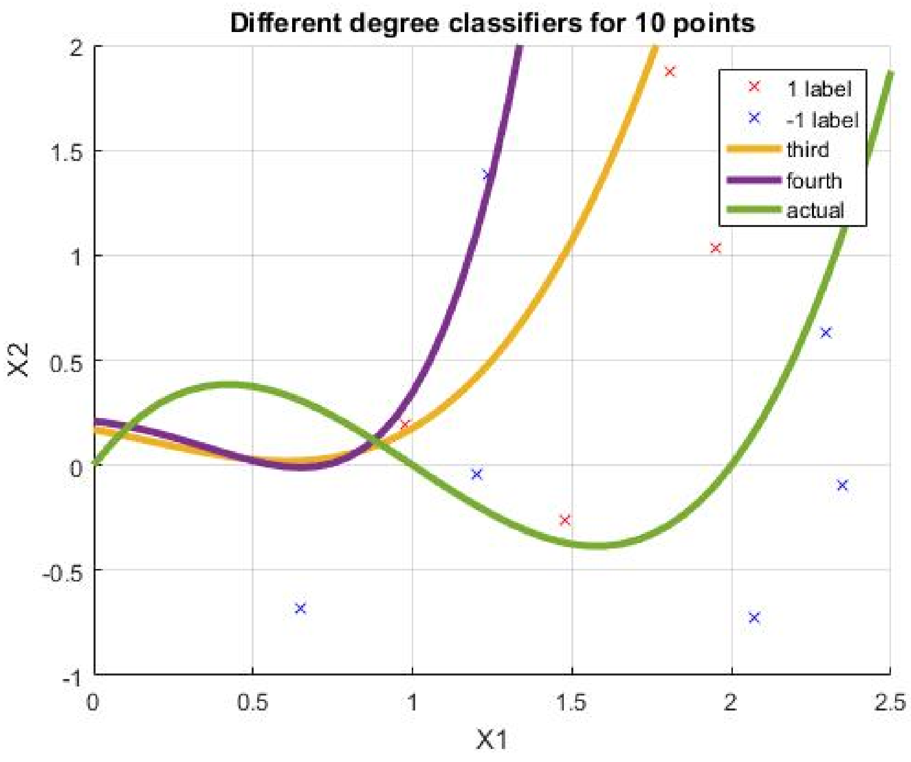


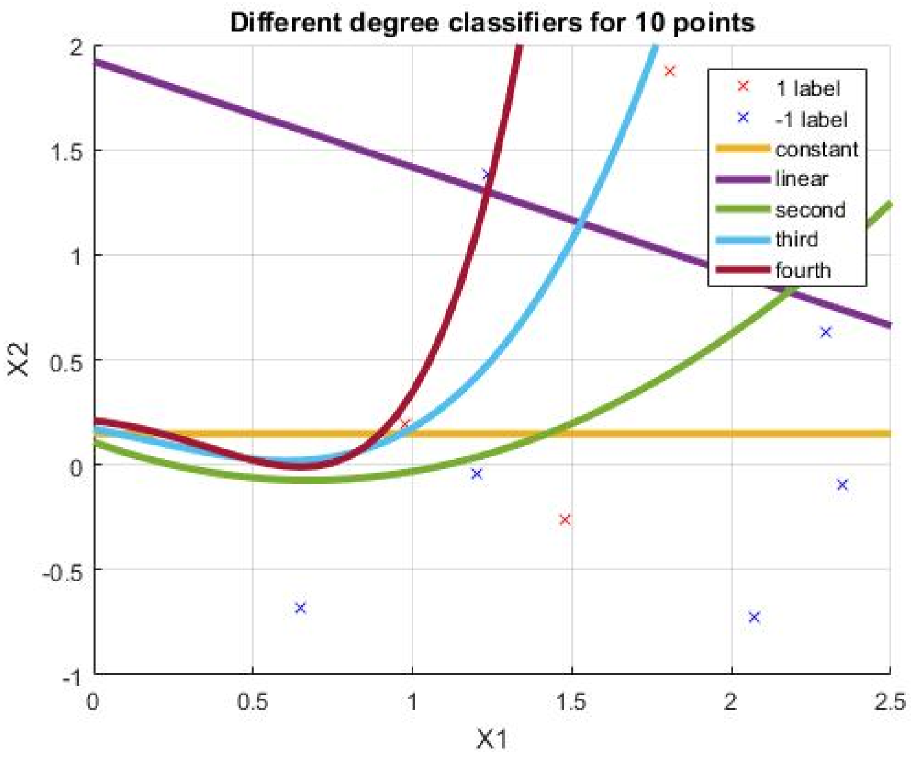
We observe form the above figure that the training error gradually decreases as q increases, indicating that the higher order hypothesis classes manage to fit the training data much better. Nevertheless, we notice that the test error does not decrease with the training error, but rather increases slightly with increasing q, thereby hinting at the over-fitting that is occurring with higher order hypothesis classes. I also plotted the complexity term, and we notice that it is quite large, in fact, much larger than 1. Thus, the bound does not add much value since the test error cannot exceed 1. I particularly note that for the third order hypothesis class, the training error is around 10% which agrees with our expected figure.

We also examine the average training and test errors of the SRM solution. For this number of data points, I observe that the average training error is 11.6%, while the average test error is 24.22%. This discrepancy between training and test errors again points to the over-fitting that is taking place.

Examining the proportion of time were the SRM is chosen from each hypothesis class, I note that 98% of the time, the SRM is chosen from $\mathcal{H}_0$, 1% from $\mathcal{H}_1$, and 1% from $\mathcal{H}_3$. This does not align to our expectations since we would expect the majority of picks to be from $\mathcal{H}_3$ since that is the true distribution of the training data. However, that is not the case due to the small number of training points we are employing.

We now move on to examine the actual polynomials produced by each hypothesis class and how they compare to the actual curve.

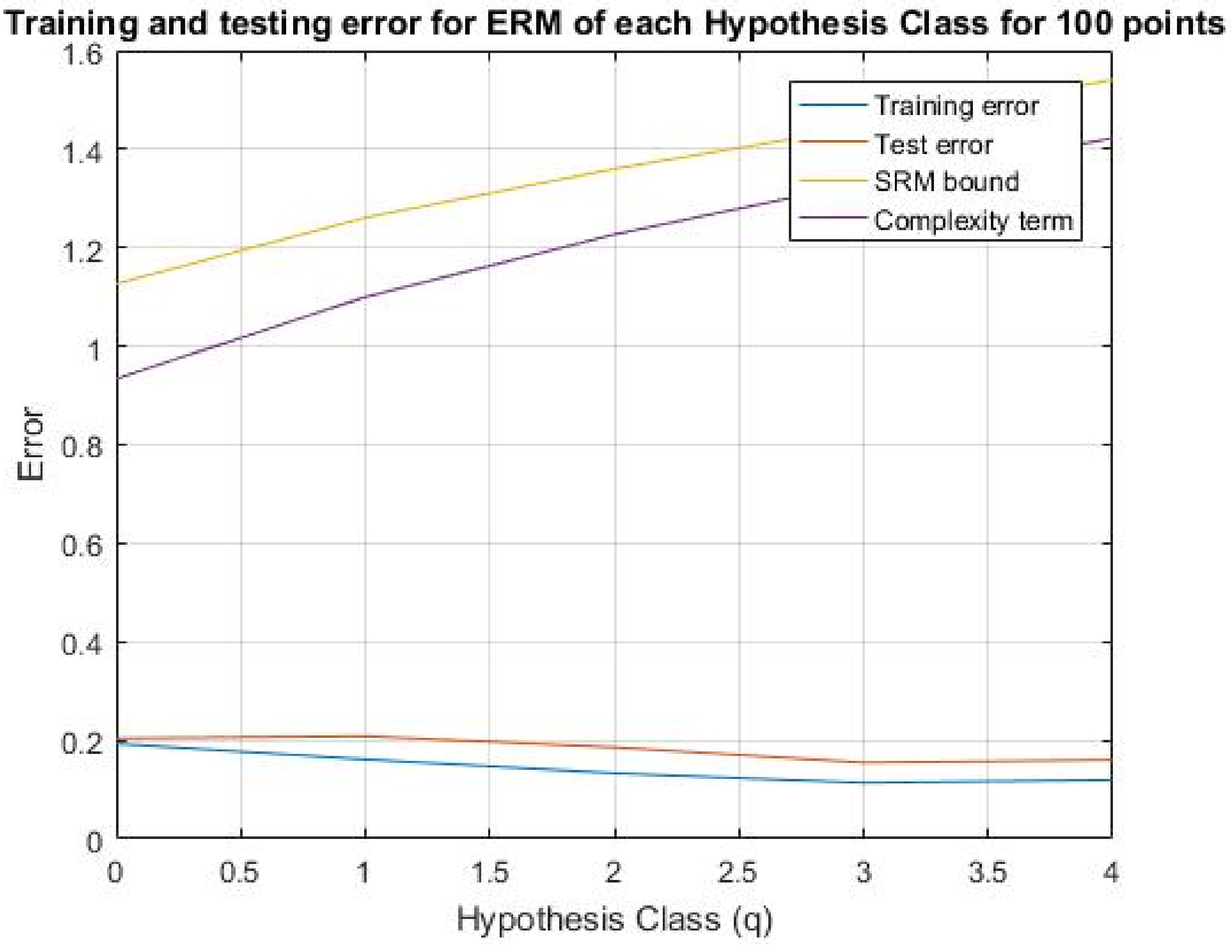




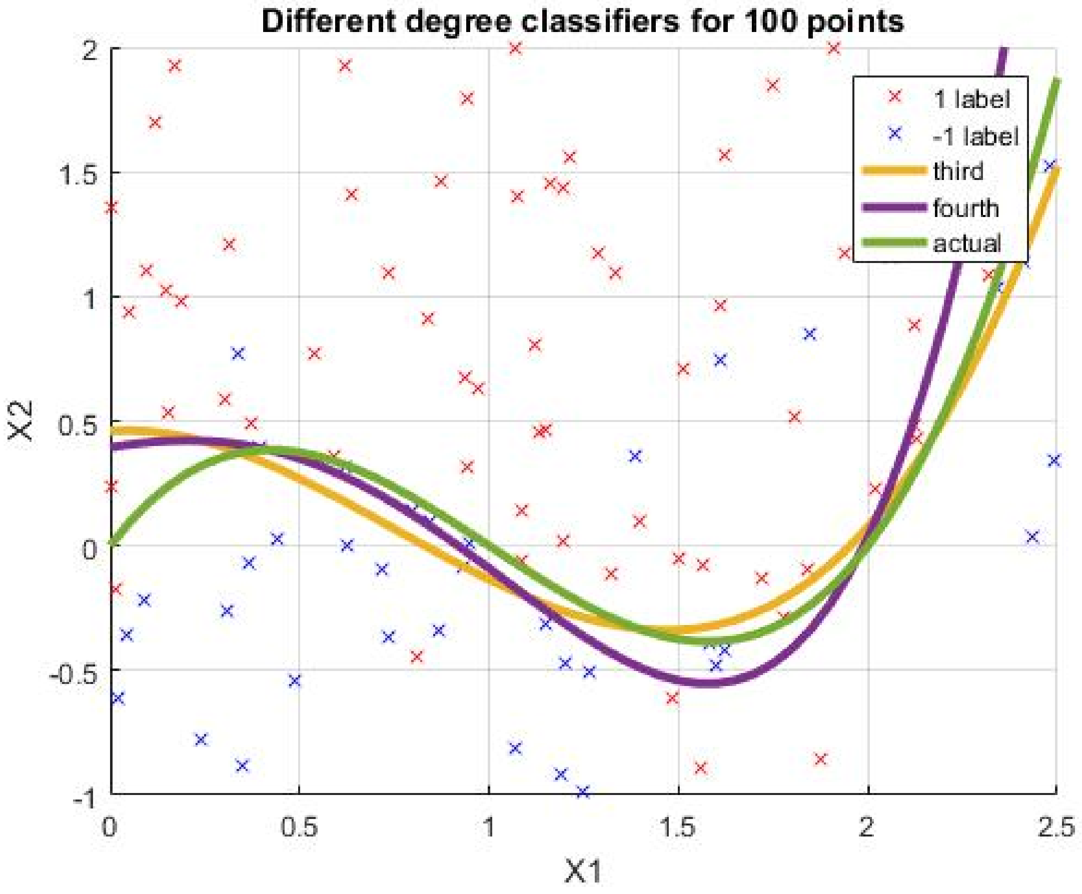
Top: Third & Fourth order polynomials against $f(x)$. Bottom: All generated polynomials from individual $\mathcal{H}_i$

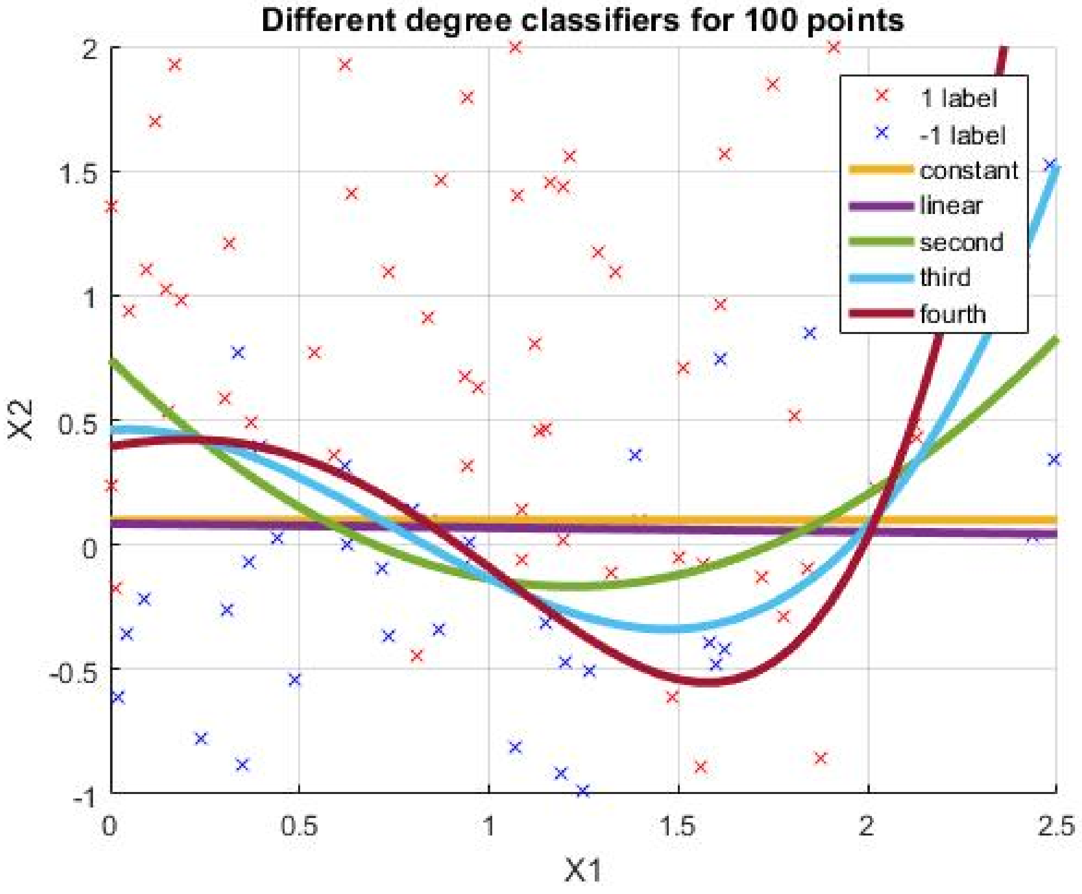
We notice that there is quite a variety in the generated classifiers. More importantly, however, we observe that the third and fourth order hypothesis classes do not produce approximations of the actual curve. Thus, it is possible to conclude that for 10-point training set the ERM solutions of the individual hypothesis classes are not particularly accurate, but this can largely be attributed to the small number of data points.

5.2.2 100-Point Training Set



Similarly to the previous case, we again observe that the training error decreases as the hypothesis class order q increases. This time, however, the test error decreases in a similar fashion, showing a deterioration in the degree of over-fitting. We notice this time that the complexity is significantly lower than in the previous case, but still substantially greater than 1, which again means that it is not too useful. In terms of the training and test performances of the SRM solution, we obtain a training error of 19.29% and a test error of 20.4%. We also not that in this case, the SRM solution is chosen 100% of the time from $\mathcal{H}_0$, which again diverges from our expectations.

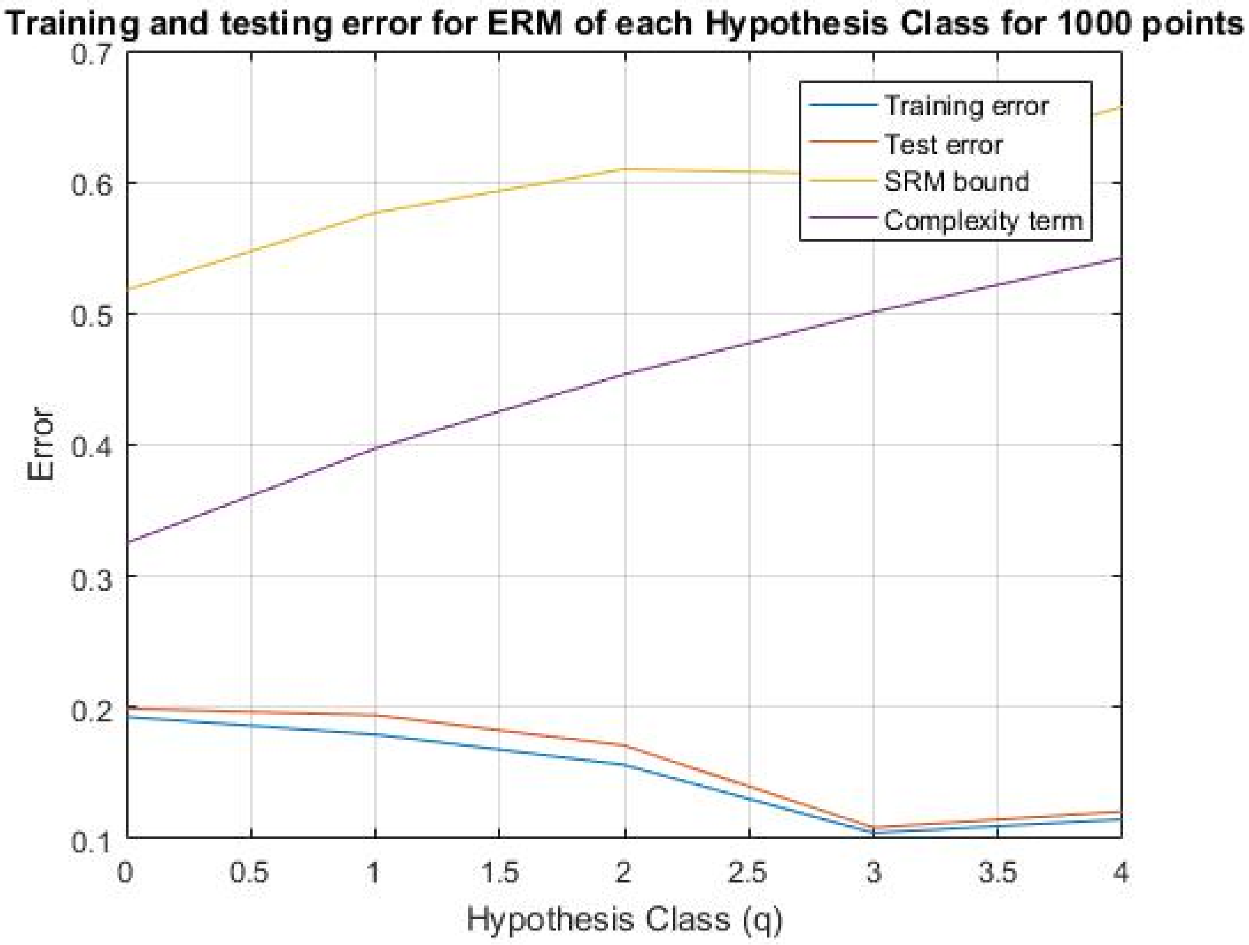




Top: Third & Fourth order polynomials against $f(x)$. Bottom: All generated polynomials from individual $\mathcal{H}_i$

Again, there appears to be a large variety again in the generated classifier polynomials. More importantly, however, it is instructive to observe the variation between those generated by the third and fourth order hypothesis classes, and we see that there is a much closer similarity between them and the actual classifier. Thus, the increase in the number of training points has resulted in an improvement in the calculation of the classifiers, as we would naturally expect.

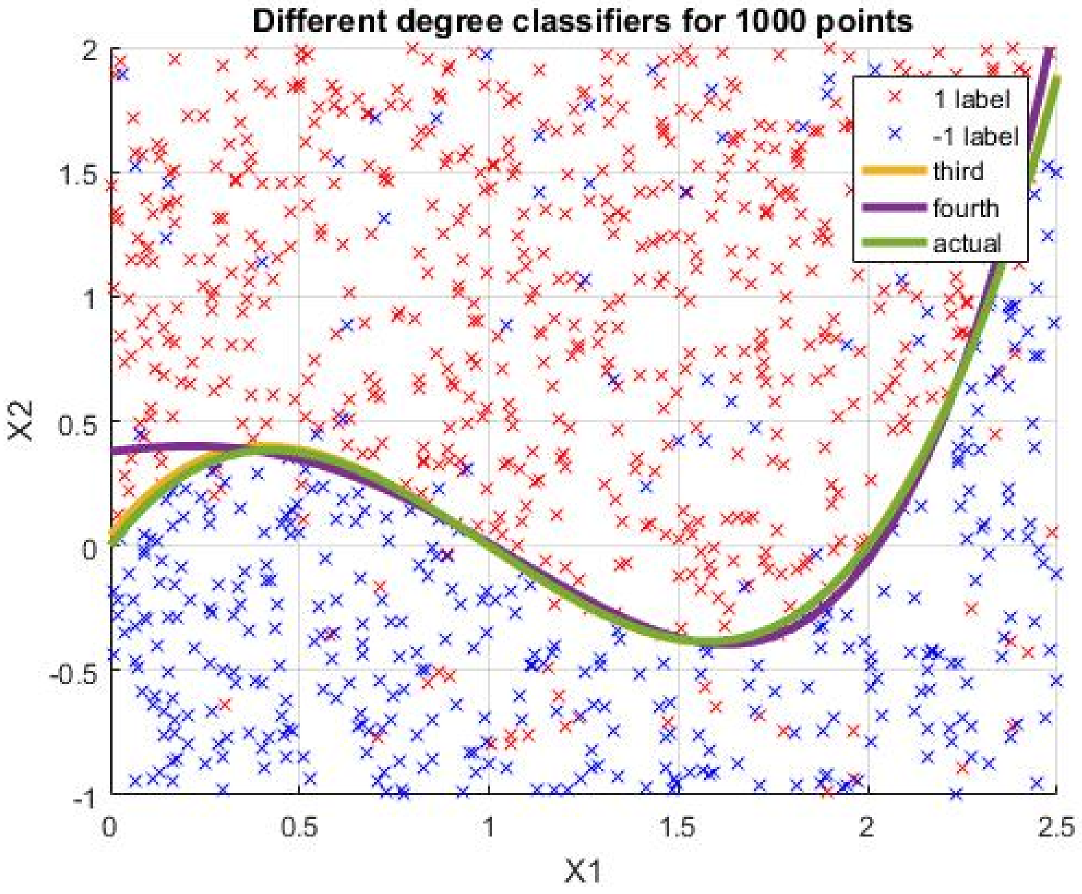
5.2.3 1000-Point Training Set

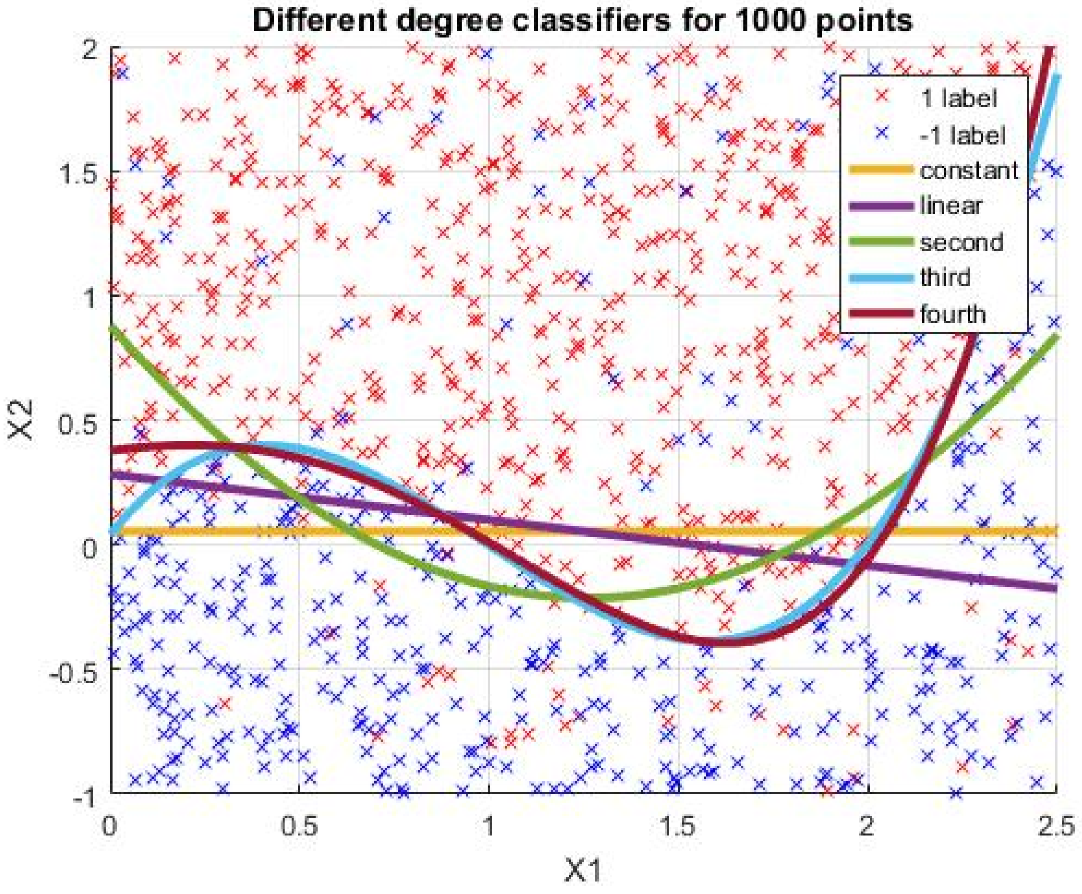


A very similar trend occurs as with the previous cases, again noting that the training error for the third order hypothesis is around 10%. However, the complexity term has now become significantly less than 1, meaning that it can now serve as a somewhat useful bound.

With respect to the training and test errors, we notice that the figures decreases with increasing hypothesis class order q, as seen before. Of particular interest here is that ERM for the third order hypothesis class produces errors that converge to around 10%. We would naturally expect that the errors produced by the fourth order hypothesis class are lower than those by the third order. However, the aberration from our expectations here is explained in a future section.

We note here that the SRM solution is chosen 100% from $\mathcal{H}_0$, and produces an average training and test error of 19.43% and 19.68%, respectively.

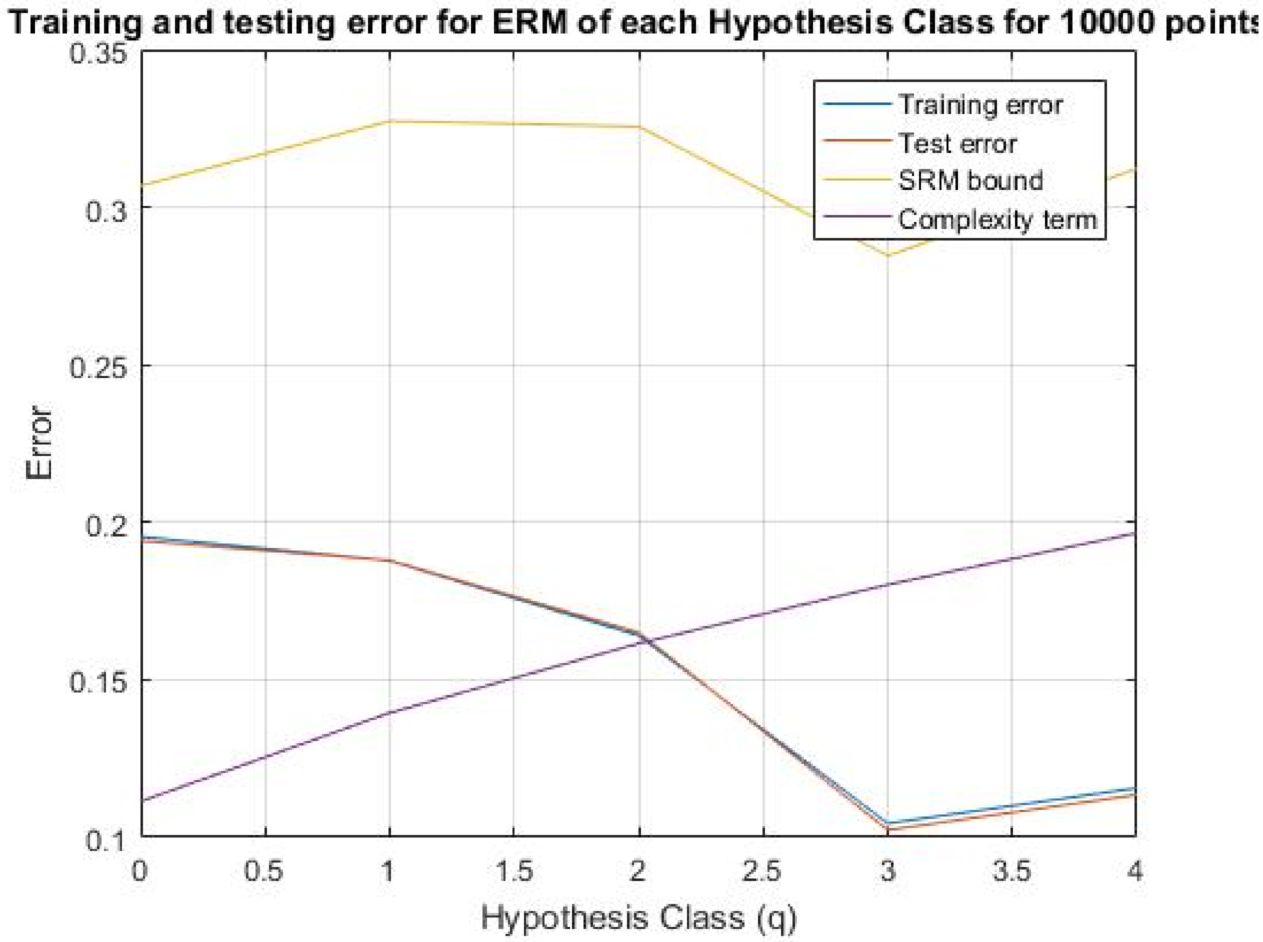




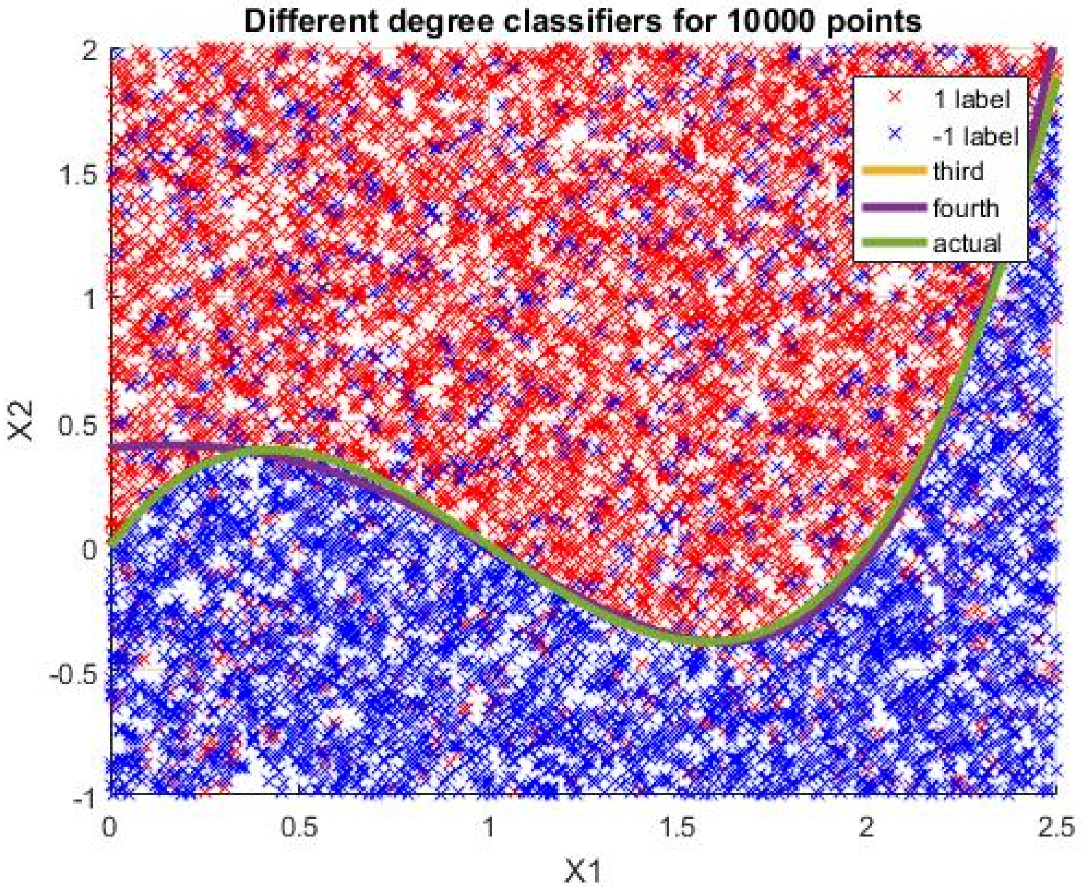
Top: Third & Fourth order polynomials against $f(x)$. Bottom: All generated polynomials from individual $\mathcal{H}_i$

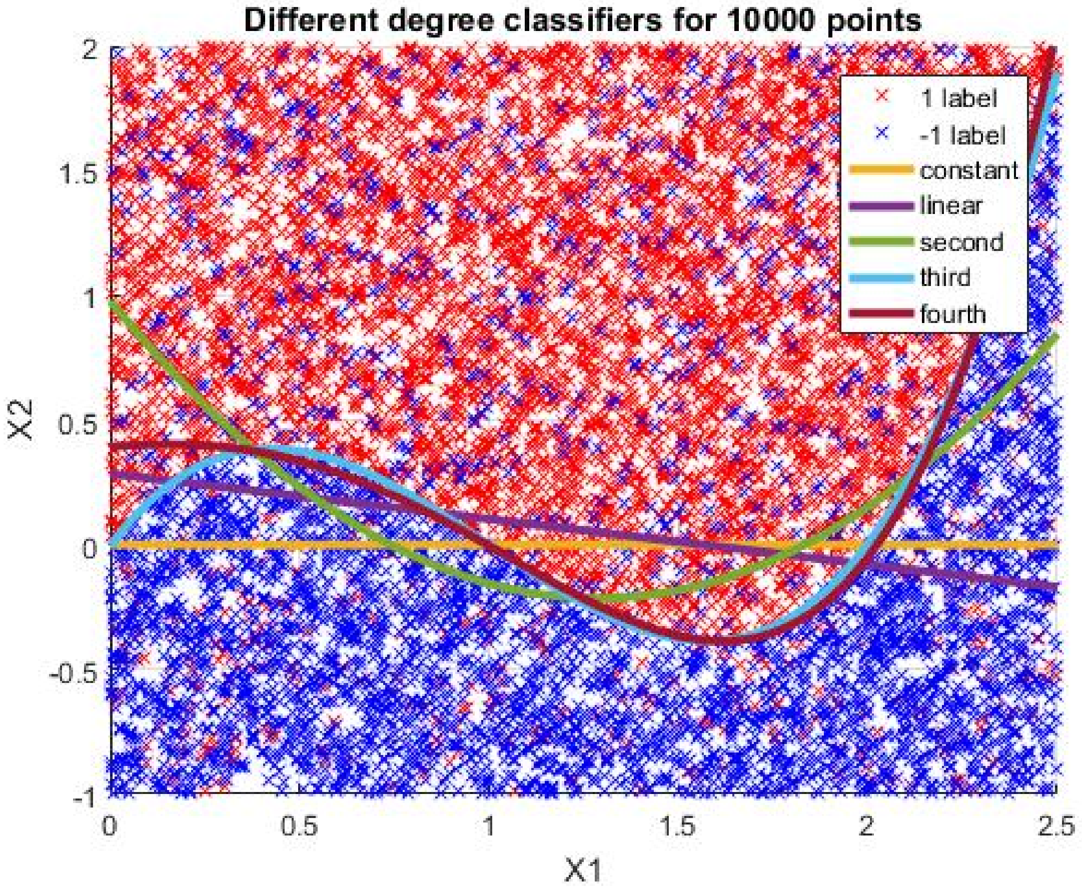
We see now that the third order hypothesis class produces a classifier polynomial that is almost exactly equal to the actual classifier.

5.2.4 10000-Point Training Set



A similar analysis can be done here. Importantly, we note that the training and test errors are 10.44% & 10.23%, respectively. Furthermore, 100% of the SRM solutions were picked from $\mathcal{H}_3$. Regarding the complexity term, we notice that it is lower than in the previous case, thus demonstrating a trend.





Top: Third & Fourth order polynomials against $f(x)$. Bottom: All generated polynomials from individual $\mathcal{H}_i$

5.2.5 General Conclusions

Over the previous sections, we observe a very revealing trend. Firstly, we observe that the SRM solution is predominantly chosen from $\mathcal{H}_0$. Moreover, the general training error produced is around 20%, which diverges significantly from our expected figure of 10%. We note that the SRM performance is quite poor over the 10 & 100 point training sets, less so on the 1000-point set, and actually quite accurate on the 10,000-point set. We also notice that as the number of training points increases, the complexity term decreases. Thus, the phenomenon where the SRM is inaccurately chosen seems to come from the loose bound introduced by the complexity term (a remedy for this will be explored in Part C), where the VC bound essentially penalizes higher order hypothesis too much. We do however note that the VC bound becomes tighter and tighter as the number of training points increases since $\Omega(n, \mathcal{H}_i, w_i\delta) \propto \frac{1}{\sqrt{n}}$.

On another note, I noticed an intuitively strange trend regarding the third & fourth order hypothesis classes. It appears that the training and test errors of the ERM from the third order hypothesis class is always less than that of the fourth order hypothesis class. Intuitively, this did not really make sense to me, since the fourth order hypothesis class is more complex and thus, should be better able to fit the data. I tried running the algorithm on a noise-less set with a Perceptron that has no limit on the number of iterations it is allowed to run. Since this data set is linearly separable, the algorithm should come to a halt. To specifically clarify this trend, I also rewrote the algorithm to output the number of iterations required for it to come to a halt. It turns out that the fourth order hypothesis class requires a number of iterations in the order of 10^5, whereas I run it for 100 iterations. This large number of iterations required can be explained by the complexity of the fourth order over the third order hypothesis class. Thus, the fourth order hypothesis does not generate an accurate classifier because it is not given enough time to run.

5.3 Part C

Although in theory the VC bound provides an upper bound on the test error of an SRM solution, we have seen that in practice, this bound is quite loose, indeed so much so that it is often meaningless and prevents the algorithm from generating acceptable results. A remedy for this is multiplied the complexity term by some attenuating factor (I will use 0.1 here). The idea is that the VC bound is attenuated such that it is below 1 and gives more importance to the empirical training error in determining the SRM solution. As we saw, this will have a greater impact on the 10 & 100 point training sets, since the complexity term was quite large in those cases. When I try this, I obtain several key figures.

Testing this on the 10-point training set, we obtain average SRM training and test errors of 4.8% & 25.12%, respectively. I also notice that the SRM solution is chosen as shown below from the different hypothesis classes.

$\mathcal{H}_0$	$\mathcal{H}_1$	$\mathcal{H}_2$	$\mathcal{H}_3$	$\mathcal{H}_4$
37%	31%	15%	10%	7%

Applying this on the 100-point training set, we obtain a significant improvement in the training and test errors, which are now 10.84% & 15.64%, respectively. We also see that the distribution of how the SRM solution is chosen has also improved, as shown below.

$\mathcal{H}_0$	$\mathcal{H}_1$	$\mathcal{H}_2$	$\mathcal{H}_3$	$\mathcal{H}_4$
3%	14%	14%	39%	30%

Testing this on the 1000-point training set, we obtain average SRM training and test errors of 10.77% & 10.84%, respectively. The SRM is chosen again 100% of the time from $\mathcal{H}_3$.

This attenuation of the complexity term indeed proves to be a remedial solution, as it reduces the training error to around 10%, and allows the algorithm to choose an SRM solution from $\mathcal{H}_3$ a larger proportion of time.

6 Appendix

generateRandPoints:

```
function y = generateRandPoints(n)

    y = [2.5*rand(n,1) 3*rand(n,1)-1];

end
```

classify:

```
function y = classify(x)

    y = zeros(1,size(x,1));

    func = @(x) x*(x-1)*(x-2);

    for i = 1:size(x,1)
        if x(i,2) >= feval(func,x(i,1))
            if rand > 0.1
                y(i) = 1;
            else
                y(i) = -1;
            end
        else
            if rand > 0.1
                y(i) = -1;
            else
                y(i) = 1;
            end

        end
    end
end
```

modpercep:

```
function [wtrue] = modpercep(X,Y,n)

w = zeros(1,size(X,2)); %Weights vector initialization
wtrue = zeros(1,size(X,2));
error = 1;

for i = 1:n %setting a defined number of iterations

    for ii = 1 : size(X,1)

        if (sign(X(ii,:)*w') ~= Y(ii)) %Check if the classification is right
            w = w + X(ii,:) * Y(ii);   %then add (or subtract) this point to w
            newerror = errorprob(sign(w*X').*Y);
            if newerror < error
                error = newerror;
                wtrue = w;
            end
        end
    end

end
```