Assignment 4

Silvan Robert Adrian

December 17, 2018

Contents

1 Airline Revisited

So as far as I understand

2 The Growth Function

2.1 Question 1

Let $\mathcal{H}$ be a finite hypothesis set with $|\mathcal{H}| = M$ hypotheses. We wish to derive a bound on the growth function $m_{\mathcal{H}}$.

Since we have a finite hypothesis set it is clear that an upper bound on the growth function is M. Further from Definition 2.2 we now that the growth function $m_{\mathcal{H}}(n)$ is bounded by 2^n. Summarizing these arguments yields that

$$m_{\mathbb{H}}(n) \leq \min\{M, 2^n\}.$$

The VC-dimension, $d_{\mathrm{vc}}(\mathcal{H})$, of a hypothesis set $\mathcal{H}$ is the largest value of N for which $m_{\mathcal{H}}(N) = 2^N$. In other words we get that

$$M = 2^N \Leftrightarrow d_{\mathrm{vc}}(\mathcal{H}) = N = \frac{\log M}{\log 2}$$

2

We have from 1 that $m_{\mathbb{H}}(n) \leq \min\{M, 2^{2n}\}$. We now consider the two possible cases:

In the case when $M \leq 2^{2n}$ then $m_{\mathbb{H}}(n) \leq M \leq M^2$, where M^2 is the bound on $m_{\mathbb{H}}(n)^2$.

In the case when $M > 2^{2n}$ then $m_{\mathbb{H}}(n) \leq 2^{2n}$, but 2^{2n} is in this case also the bound on $m_{\mathbb{H}}(n)^2$.

All in all we must have that

2.2 Question 3

We wish to prove by induction that

$$\sum_{i=0}^{d} \binom{n}{i} \leq n^d + 1.$$

First we show that it holds for $d = 0$. We get

$$\sum_{i=0}^{0} \binom{n}{i} = \binom{n}{0} = \frac{n!}{0!(n-0)!} = \frac{n!}{n!} = 1 < n^0 + 1 = 2$$

We now assume it holds up from 1 up to $d-1$, and wish to show that it also holds for d. We get

$$\sum_{i=0}^{d} \binom{n}{i} = \sum_{i=0}^{d-1} \binom{n}{i} + \binom{n}{d} \overset{ind.ass.}{\leq} n^{d-1} + 1 + \frac{n!}{d!(n-d)!} \leq n^d + 1,$$

where the last inequality follows by using the properties of the binomial coefficient!

2.3 Question 4

From Theorem 2.4 in Learning From Data we have that if $m_{\mathcal{H}}(k) < 2^k$ for some k then it holds for all N that

$$m_{\mathcal{H}}(N) \leq \sum_{i=0}^{k-1} \binom{N}{i}.$$

By using this result together with the result from 3, we get the following bound on $m_{\mathcal{H}}(n)$

$$m_{\mathcal{H}}(n) \leq \sum_{i=0}^{k-1} \binom{n}{i} \leq n^{d-1} + 1.$$

5

By using the bound from 4 we get the following VC generalization bound by inserting into Theorem 2.5 of Learning From Data

$$E_{\text{out}}(g) \leq E_{\text{in}}(g) + \sqrt{\frac{8}{N} \ln \frac{4 m_{\mathcal{H}}(2n)}{\delta}} \leq E_{\text{in}}(g) + \sqrt{\frac{8}{N} \ln \frac{4\left((2n)^{d-1} + 1\right)}{\delta}}$$

Since the bound is derived from using binomial coefficients we must have that $d \leq n$. Otherwise the following quantity is not meaningful (negative factorials are not defined)

$$\binom{n}{d} = \frac{n!}{d!(n-d)!}$$

3 VC Dimension

3.1 Question 1

We consider $\mathcal{H}_+$ to be the class of positive circles in $\mathbb{R}^2$, and wish to determine the VC-dimension of of $\mathcal{H}_+$. The VC-dimension of a hypothesis set $\mathcal{H}$ is the largest value of N for which $m_{\mathcal{H}}(N) = 2^N$. Less formally it is the largest numbers of points we can shatter/classify by a hypothesis $h \in \mathcal{H}_+$.

It is obvious that we can shatter every 2 distinct points by positive circles. To shatter 3 points by positive circles is a bit more difficult! If we consider 3 points they are either colinear or form a triangle. It is clear that we can not shatter 3 points on a line. If the 3 points form an equilateral triangle, we will actually be able to perfectly shatter the points! In other words a lower bound on the VC-dimension is 3. That we cannot find an example on 4 points, which can be shattered is a bit more difficult to show! 4 distinct points either form a line, or the convex hull is a triangle or a quadrilateral. 4 points on a line cannot be shattered, imagine if we wanted to classify point 2 and 4 as +1. If the convex hull is a triangle we will never be able to classify the 3 points forming the triangle as +1 and the point inside as -1. If the convex hull is a equilateral we would not be able to classify the two points on the 'diagonal. Summarizing these observations we must have that $d_{\text{VC}}(\mathcal{H}_+) = 3$

3.2 Question 2

We now consider $\mathcal{H} = \mathcal{H}_+ \cup \mathcal{H}_-$, where $\mathcal{H}_-$ denotes negative circles in $\mathbb{R}^2$. Considering this hypothesis set, we can actually find examples on how to shatter 4 points in $\mathbb{R}^2$. For example if the convex hull forms a triangle, we can now classify the points forming the triangle by +1 by drawing a negative circle around the point inside the triangle. Thus we must at least have that $d_{\text{VC}}(\mathcal{H}) = 4$. That is not bigger than 4 is a bit more difficult to justify, but it is the case! You can verify this by considering the possible ways 5 points can be arranged (i.e colinear, convex hull is a triangle etc.)

4 SVMs

4.1 Data normalization

I define my normalization function $f_{norm} : \mathbb{R}^{22} \to \mathbb{R}^{22}$ by:

$$f_{norm}(x) = \left(f^1_{norm}(x_1), ..., f^{22}_{norm}(x_{22})\right) \tag{1}$$

where

$$f^i_{norm}(x_i) = \frac{x_i - \mu_i}{\sigma_i} \tag{2}$$

μ_i and σ_i are here the empirical mean and empirical standard deviation.

I did do the computation with `sklearn` by using the `StandardScaler` from preprocessing in `sklearn` to normalize the data.

Here we have the table of the mean and standard deviation, before and after the normalization of the training data:

Feature	before mean	normalized mean	before std. deviation	normalized std. deviation
1	155.9604	0.0	44.3036	1.0
2	204.8212	0.0	98.1520	1.0
3	115.0586	0.0	45.7556	1.0
4	0.0060	0.0	0.0040	1.0
5	0.0000	0.0	0.0000	1.0
6	0.0032	0.0	0.0024	1.0
7	0.0033	0.0	0.0023	1.0
8	0.0096	0.0	0.0071	1.0
9	0.0277	0.0	0.0159	1.0
10	0.2624	0.0	0.1627	1.0
11	0.0147	0.0	0.0087	1.0
12	0.0166	0.0	0.0101	1.0
13	0.0220	0.0	0.0133	1.0
14	0.0440	0.0	0.0260	1.0
15	0.0226	0.0	0.0298	1.0
16	22.0007	0.0	4.0632	1.0
17	0.4948	0.0	0.1015	1.0
18	0.7157	0.0	0.0558	1.0
19	-5.7637	0.0	1.0304	1.0
20	0.2148	0.0	0.0758	1.0
21	2.3658	0.0	0.3694	1.0
22	0.1997	0.0	0.0816	1.0

Same for test data, also sowing the before and after mean/standard deviation:

Feature	before mean	normalized mean	before std. deviation	normalized std. deviation
1	152.479041	0.078579	37.909624	0.855678
2	189.309093	0.158042	82.990021	0.845525
3	117.603691	0.055623	40.863362	0.893079
4	0.006445	0.113184	0.005578	1.410816
5	0.000045	0.071574	0.000039	1.290752
6	0.003410	0.086915	0.003456	1.461758
7	0.003579	0.115672	0.003155	1.386457
8	0.010230	0.087016	0.010370	1.462078
9	0.031699	0.248982	0.021164	1.331149
10	0.302299	0.245187	0.220027	1.352389
11	0.016662	0.229566	0.011338	1.310464
12	0.019155	0.250891	0.013502	1.333356
13	0.026196	0.316608	0.019672	1.479942
14	0.049988	0.229603	0.034015	1.310517
15	0.027078	0.149057	0.048595	1.631861
16	21.770062	0.056763	4.740054	1.166577
17	0.502290	0.073568	0.105654	1.040496
18	0.720533	0.086767	0.054445	0.975350
19	-5.604248	0.154772	1.136498	1.102955
20	0.238346	0.310695	0.088486	1.167391
21	2.398055	0.087416	0.393293	1.064668
22	0.213465	0.168577	0.097058	1.189412

So we can see now that each deviation in the training data end up to be 1 and the mean 0. But since we also use the empirical mean and standardized deviation of the training data for the test data we won't end up exactly on 1 or 0. But in the end the normalized means and normalized standard eviations end up still vey much more near to 1 or 0 than the original ones.

4.2 Model selection using grid-search

My selection for the logarithmic scale for y and C, by setting $C = 10$, and $y = 0.1$ in the middle of the scale:

$$\mathcal{C} = \{0.01, 0.1, 1, 10, 100, 1000, 10000\} \tag{3}$$
$$\mathcal{Y} = \{0.0001, 0.001, 0.01, 0.1, 1, 10, 100\} \tag{4}$$

I implemented the 5-cross validation with `GridSearchCV` from the python module `sklearn.model_selection` in the `sklearn` library. Where we calculate each cross validation score for all pairs of (C, y), the pair with the highest score would be then the best hyperparamter pair configuration we are searching for. So that I ended up using a heatmap to show all the cross validation scores from which we can read out the best possible configuration.

	0.0001	0.0010	0.0100	0.1000	1.0000	10.0000	100.0000
0.01	0.734694	0.734694	0.734694	0.734694	0.734694	0.734694	0.734694
0.1	0.734694	0.734694	0.734694	0.734694	0.734694	0.734694	0.734694
1.0	0.734694	0.734694	0.867347	0.897959	0.795918	0.734694	0.734694
10.0	0.734694	0.877551	0.897959	0.908163	0.795918	0.775510	0.734694
100.0	0.877551	0.887755	0.867347	0.908163	0.795918	0.775510	0.734694
1000.0	0.887755	0.846939	0.877551	0.908163	0.795918	0.775510	0.734694
10000.0	0.846939	0.877551	0.877551	0.908163	0.795918	0.775510	0.734694

Figure 1: Showing all the cross validation scores in a table

Either from the table above or getting it from the outputs of my implementation, we will see that that the best validation score we get with the hyperparameters $\{C = 10, y = 0.1\}$. Is the validation score of $0.908163 \approx$. Additionally we can calculate the accuracy of the cross validation which is: $0.907216 \approx$

4.3 Inspecting the kernel expansion

For the purpose of calculating bounded and free bounded vectors I used `sklearn` again, by fitting the data again as in the exercise before and then keep y the same while going through various values of C. Then I came to following solutions:

```
C = 0.1,   bounded support vectors: 54, free support vectors: 0
C = 10,    bounded support vectors: 23, free support vectors: 17
C = 100,   bounded support vectors: 12, free support vectors: 20
C = 1000,  bounded support vectors: 1 , free support vectors: 26
C = 10000, bounded support vectors: 0 , free support vectors: 26
```

As we see above the value of C affects the misclassification rate of training examples. For small values of C a large-margin hyperplane will be still used regardless of the misclassification rate. For larger values of C, the SVM will pick a smaller-margin hyperplane provided it minimizes the misclassification rate.