Assignment 4

Silvan Robert Adrian - zlp432

December 18, 2018

Contents

1 Airline Revisited

Since we work with a bernulli distribution here we have probability p for showing up and not showing up $1-p$ So from the question we have that 0.95 passengers show up while ther is a 0.05 percentage who is not showing up. Also we have 2 Events, and as described we seem to want both Events happening at the same time so $P_p(A \wedge B)$:

- Event A: An i.i.d sample of 10000 flight reservations, and 95% show up in them
- Event B:An i.i.d sample of 100 flights, on which all passengers show up (100 of 99)

We know that the 2 Events are independent which means $P_p(A \wedge B) = P_p(A)P_(B)$. Where p is the probability of a passenger is showing up for his reservation.

Further we would need to multiplicate those 2 probabilities etc.

But I ended up not doing this exercise, any input if the start is right or not would be still appreciated.

2 The Growth Function

2.1 Question 1

Let $\mathcal{H}$ be a finite hypothesis set with $|\mathcal{H}| = M$ hypotheses.

Because we have a finite hypothesis set, we know that an upper bound on the growth function is M.

We also know that the growth function $m_{\mathcal{H}}(n)$ is bounded by 2^n from definition 2.2 (learning from data book). So from these arguments we can say that it holds:

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

2.2 Question 2

skipped

2.3 Question 3

We do know have from question 1 that $m_{\mathbb{H}}(n) \leq \min\{M, 2^{2n}\}$.

So we have to consider 2 cases:

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

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

So over all we must have:

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

2.4 Question 4

Since I haven't done the prove before, here I have done the prove by induction:

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

First show that it holds for $d = 0$:

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

Now we want to show that is also holds for 1 up to $d - 1$:

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

Now we want to derive the bound on $m_{\mathcal{H}}(n)$. From theorem 2.4 (Learning From Data) we know that if $m_{\mathcal{H}}(k) < 2^k$ for some value k then for all N:

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

Now we use the result of the induction above and the theorem 2.4 from the book, so that we end up on the following bound:

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

2.5 Question 5

Now using the bound from 4 we will get the following VC generalization bound, by inserting it into theorem 2.5 from the book (learning from data):

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

2.6 Question 6

Since the bound is derived from using binomial coefficients we must have that $d \leq n$. Since negative factorials are not defined which would not be meaningful.

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

2.7 Question 7

skipped

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 $m_{\mathcal{H}}(N) = 2^N$. So in short we want to find the largest numbers of points we can shatter/classify by $h \in \mathcal{H}_+$.

Well 2 points is obviously possible to shatter. More interesting are 3 points for which I created a figure to show all possible triangle arrangements 2^3 arrangements:

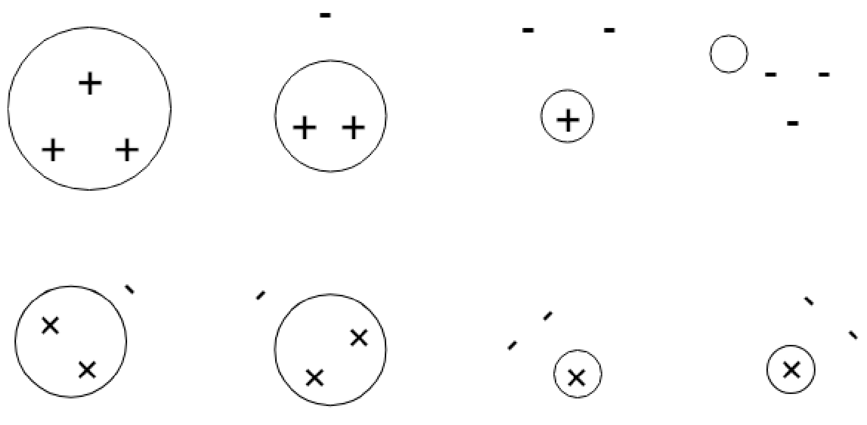

Figure 1: shattering 3 points

So we now we would need to shatter 4 points, which is not in all cases possible, following a working case and a not working case:

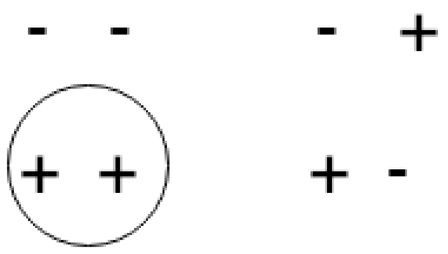

Figure 2: shattering 4 points

So here the right case wouldn't be possible to shatter with a circle. So therefore we need to have: $d_{\mathrm{VC}}(\mathcal{H}_+) = 3$.

3.2 Question 2

In this question we consider $\mathcal{H} = \mathcal{H}_+ \cup \mathcal{H}_-$, where $\mathcal{H}_-$ denotes negative circles in $\mathbb{R}^2$. In this case we are able to find examples which actually work with shattered 4 points:

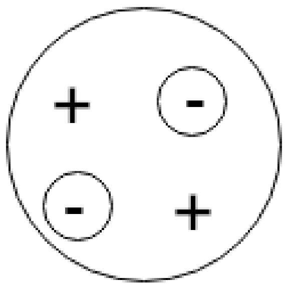

Figure 3: shattering 4 points

So now we can draw negative circles around negatives, since the negative circles are positive on the outside. This way we can say $d_{\mathrm{VC}}(\mathcal{H}) = 4$, for all points more than 4 probably can also be shown that it would be possible but I only proved it for 4 now.

4 SVMs

4.1 Data normalization

We get following normalization function from the assignment text: $f_{norm} : \mathbb{R}^{22} \rightarrow \mathbb{R}^{22}$ which means:

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

from which we can say:

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

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 showing 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:

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

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.

	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 4: Showing all the cross validation scores

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 $\{C = 10, y = 0.1\}$.

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
```

We see that the number of bounded support vectors increase when we decrease C, and decrease when we increase C. This does make sense since C has the role to penalise misclassifications. So when we increase C the model gets more complex, which mean it tries to fit all data points.