Machine learning 2 - Final Assignment

Frans de Boer
frans.deboer
5661439

June 28, 2022

The git repository for this assignment (including code and the report) can be found at github.com. I will make this repository public after the Brightspace deadline has passed.

1 PAC Learning

1.a Extend the proof that was given in the slides for the PAC-learnability of hyper-rectangles: show that axis-aligned hyper-rectangles in n-dimensional feature spaces (n > 2) are PAC learnable.

We can extend the proof by taking the number of sides/faces of the hyper-rectangle into account.

- A hyper-rectangle in 2-dimensional space is simply a rectangle and has 4 sides.
- A hyper-rectangle in 3-dimensional space is a rectangular box and has 6 sides.

For now I will assume the number of sides on a n-dimensional hyper-rectangle to be $F(n)$, and I will get back later to the calculation of $F(n)$.

To start we can follow the steps from the lecture. We have the ground truth n-dimensional hyper-rectangle R, and the current tightest fit rectangle R'. The error $(R - R')$ can be split into $F(n)$ strips T'. On each of these strips we can now grow a new strip T such that the probability mass of T is $\frac{\varepsilon}{F(n)}$.

If T covers T' for all strips then the probability of an error is

$$P[error] \leq \sum_{i=0}^{F(n)-1} P[T_i] = F(n)\frac{\varepsilon}{F(n)} = \varepsilon \tag{1}$$

We can now estimate the probability that T does not cover T'

$$P[\textit{random x hits T}] = \frac{\varepsilon}{F(n)}$$
$$P[\textit{random x misses T}] = 1 - \frac{\varepsilon}{F(n)}$$
$$P[\textit{m random x's misses T}] = (1 - \frac{\varepsilon}{F(n)})^m$$

Since we have $F(n)$ strips:

$$P[\textit{m random x's miss any Ts}] \leq F(n)(1 - \frac{\varepsilon}{F(n)})^m$$
$$P[\textit{R' has larger error than } \varepsilon] \leq F(n)(1 - \frac{\varepsilon}{F(n)})^m \ < \delta$$

Bounding the chance that our R' has an error larger than ε by δ:

$$F(n)(1 - \frac{\varepsilon}{F(n)})^m \ < \delta$$

Using $e^{-x} \geq (1-x)$

$$F(n)e^{-m\varepsilon/F(n)} \geq F(n)(1 - \frac{\varepsilon}{F(n)})^m$$

So instead we can use:

$$F(n)e^{-m\varepsilon/F(n)} < \delta$$
$$-m\varepsilon/F(n) < \log(\delta/F(n))$$
$$m\varepsilon/F(n) > \log(F(n)/\delta)$$
$$m > (F(n)/\varepsilon)\log(F(n)/\delta)$$

So any n-dimensional hyper-rectangle is learnable.

1.b Assume we have a 2-dimensional feature space R2, and consider the set of concepts that are L1-balls: $c = \{(x,y) : |x| + |y| \leq r\}$ (basically, all L1-balls centered around the origin). Use a learner that fits the tightest ball. Show that this class is PAC-learnable from training data of size m.

Same thing? what. literally just question a but now rotated 45 degrees

1.c Now we extend the previous class by allowing the varying center: $c = \{(x,y) : |x-x0| + |y-y0| \leq r\}$. Is this class PAC-learnable? Proof if it is, or otherwise disproof it.

maybe, maybe not. can't you still just do the tightest fit L1 ball? consistent learner so PAC learnable?

2 VC Dimension

Let us assume we are dealing with a two-class classification problem in d-dimensions. Let us start off with refreshing our memories. Consider the class of linear hypotheses (i.e., all possible linear classifiers) in d-dimensional space.

2.a Given N data points, how many possible labelings does this data set have?

Each data point can have 2 labels, so 2^n

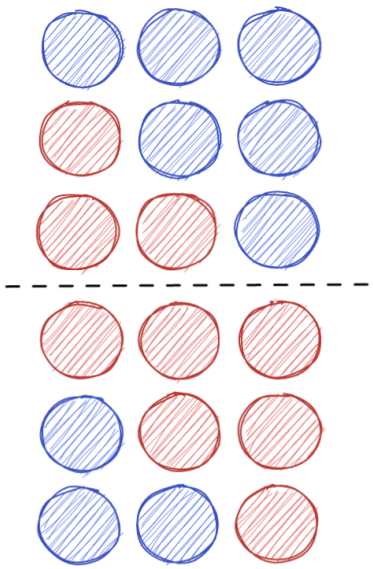
2.b What is the dimensionality d we need, at the least, for our class of linear hypotheses to be able to find perfect solutions to all possible labelings of a data set of size N? Stated differently, what is the smallest d that allows us to shatter N points?

$d - 1$

2.c Consider d = 1 and a data set of size N. At maximum, how many different labelings of such a data set can decision stumps solve perfectly, i.e., with zero training error?

This would be all cases where there is no overlap, i.e. all points with class 1 (or 2) are either all on the left or right side. See Figure 1 for an example with N=3. Note that the first 3 and last 3 examples are the same, just with inverted labelings.

Figure 1: Example of perfectly solvable labels with $N = 3$



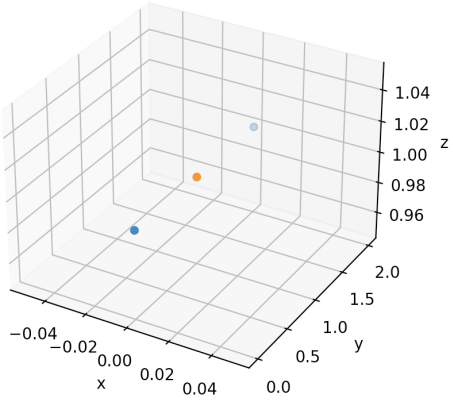
With a dataset of size N, there are N ways in which we can have class 1 be on the left side and have a decision stump be able to solve it perfectly. There can be $1, 2, ..., N$ points with class 1 in a row. Because each of these possible label assignments has a compliment (just invert the classes), we end up with a total of $2N$ possible labelings that a decision stump can solve perfectly.

2.d With the answer from 3.c, give an upper-bound on the maximum number of different labelings a data set of size N in d dimensions can get by means of decision stumps.

I'll be answering this question a bit more formally because finding the answer took a lot more exploring (as of writing this I still do not have the final answer).

One initial observation is that in most cases (I'll discuss some cases later where this is not true) adding an extra data point simply adds 2 to the total number of possible labelings. See Figure 2 for an example. If we simply add the new point far away from the original data set such that each stump is able to classify it by itself, then this new point can be classified in 2 possible ways, and the original data set can be classified in the same number of ways as before this point was added. Note that the cases here are additive, not multiplicative.

Figure 2: Adding a point to a data set with N points



We can once again look at Figure 1, but now instead of true labels we can view the labels (red and blue) as labels predicted by the decision stump. The decision stump either assigns all points to the left of it as class 1 or as class 2. So in $d = 1$ we know the maximum number of different labelings a data set of size N can get is $2N$.

2.e Using the bound from Exercise 3.d, determine the smallest upper-bound on the VC-dimension for decision stumps for dimensionalities $d \in \{1, 2, 3, 4, 5, 6, 7, 8, 1024, 2^{100}\}$

VC-Dimension: The largest number of nodes N that a classifier can shatter in all possible 2^N ways for any possible data set. **Smallest upper-bound**: First size N in dimension d where decision stumps cannot shatter any data set