Algoritmos e Incerteza **(PUC-Rio INF2979, 2018.2)**

Lecture 11: PAC Learning and VC dimension

October 23, 2018

Lecturer: Marco Molinaro *Scribe: Thiago Milagres*

Section 1 contains a brief recap on the PAC Learning framework, which we've seen in the last lecture. Section 2 extends the results seen to the non-realizable case and to situations in which our family of classifiers is infinite. Section 3 focuses on the concept of VC dimension, and how to apply it to improve our previous results.

1 Recap: PAC Learning

PAC (Probably Approximately Correct) Learning is a framework to analyze machine learning. In our discussion, we will work with binary classification, and we denote $\mathcal{Y}$ as the set of two possible labels. Then, in this model, we have:

- The population, or universe, $\mathcal{X}$, which is known to the learner. This is the set of objects that we wish to classify, and they are usually represented by a vector of features.
- μ, a distribution over $\mathcal{X}$. For $x \in \mathcal{X}$, $\mu(x)$ is the probability to sample x. This distribution μ is unknown to the learner.
- Training data: the learner has access to n instances of x, which are sampled from the population according to the distribution μ. The learner knows the correct classification of these n points, i.e. it has access to the set of pairs:

  $$S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$$

  where $y_i = c^*(x_i)$
- An optimal classifier c^*, also unknown to the learner, that correctly classifies every point of the population $\mathcal{X}$.

Then, we let $\mathcal{C}$ be the family of possible classifiers that the learner considers (for example, every neural network with at most k nodes, or every decision tree with height at most k).

Definition 1.1. As we've seen, each classifier $(c_1, c_2, \ldots, c_k) \in \mathcal{C}$ has an error, which can be defined as the proportion of the instance space in which it incorrectly classifies the data:

$$err(c) = Pr_{\mathcal{X}\sim\mu}\{c(x) \neq c^*(x)\} = \sum_{x\in\mathcal{X}} \mu(x) \cdot \mathbb{1}[c(x) \neq c^*(x)]$$

i.e. we sum $\mu(x)$ only when $c(x)$ is incorrect.

The learner's goal is to, using only the training data, output the best classifier of the family, i.e. to output $\bar{c}$ such that:

$$err(\bar{c}) \approx err^*$$

where $err^* = \min_{c \in \mathcal{C}} err(c)$

Then, the following question naturally comes up:

Question How many samples are needed to guarantee $err(\bar{c}) \leq err^* + \epsilon$ with high probability?

In the last lecture, we started to answer this question, considering the realizable case (which means that $c^* \in \mathcal{C}$). We introduced the following theorem:

Theorem 1.2. *Suppose that the realizability assumption holds, i.e. $c^* \in \mathcal{C}$. It follows that $err^* = 0$.*

Then, if we use ERM (choose $\bar{c}$ as the classifier in $\mathcal{C}$ with the smallest error in the training data), we can guarantee that, "with high probability":

$$err(\bar{c}) \lessapprox \frac{\log |\mathcal{C}|}{n} \tag{1}$$

To run the above algorithm, we need to calculate the in-sample error for each classifier. To do that, we can use the following definition:

$$err_S(c) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}[c(x_i) \neq c^*(x_i)]$$

Although Theorem 1.2 shows an important result, it has strong assumptions that could limit its applications. In today's lecture, we will see some ways to extend its implications into more general problems.

2 Extending previous results

Some of the aspects that we would like to improve in Theorem 1.2 are the following:

(i) It needs the realizability assumption, assuming that $c^* \in \mathcal{C}$.

(ii) It takes the cardinality $|\mathcal{C}|$ into account. This is a problem because in many cases, even in simple ones, the set of possible classifiers is infinite. For example, consider the set of linear separators in d dimensions.

(iii) ERM in general can be computationally intractable. For example, if our classifiers are neural networks, minimizing the empirical error is a non-convex problem.

Problem (iii) is somewhat out of the scope of our lecture, since it widely depends on the structure of the family chosen. However, we can say that with small changes, the guarantees proven here still work even if we are not able to give an exact solution to the problem, but can find an approximate one.

Now, we will focus in (i) and (ii).

2.1 Considering the non-realizable case

Theorem 2.1. *Suppose, for now, that $\mathcal{C}$ is finite. Then, ERM returns $\bar{c}$ that satisfies:*

$$Pr\{err(\bar{c}) \geq err^* + \epsilon\} \lessapprox 2|\mathcal{C}|e^{-n\epsilon^2} \tag{2}$$

Corollary 2.2. *Equivalently, we can reparametrize* (2) *to say the following:*
With probability at least $1-\delta$, we guarantee:

$$err(\bar{c}) \lessapprox err^* + \sqrt{\frac{\log(|\mathcal{C}|)/\delta}{n}} \tag{3}$$

Comparing equations (1) and (3), we see two important differences.

The first, and more natural, is that in (1) we did not have to take into account err^*, since it was zero, which is not necessarily the case in the non-realizable case.

The second difference is that in (1), the error decreases with $\mathcal{O}(1/n)$, and in (3) it decreases with $\mathcal{O}(1/\sqrt{n})$, which is way slower. This is a consequence of the nature of the new problem, and can't be avoided.

We will now prove Theorem 2.1

Proof. (Chernoff + Union Bound)

1. Since $|\mathcal{C}|$ is finite, we can enumerate the classifiers to simplify notation:

   $$\mathcal{C} = \{c_1, c_2, \ldots, c_k\}$$

   We then define the "Bad" events: Let B_i be the event described by

   $$err_S(c_i) \notin [err(c_i) \pm \epsilon/2]$$

   or, equivalently:

   $$|err_S(c_i) - err(c_i)| \geq \epsilon/2 \tag{4}$$

   Intuitively, B_i happens when the error of classifier c_i in the training set is too different from its "true" error in the population.

2. We wish to show:

   $$Pr\{B_1 \text{ or } B_2 \text{ or } \ldots \text{ or } B_k\} \leq 2|\mathcal{C}|e^{-n\epsilon^2}$$

3. Using Union Bound, we know that:

   $$Pr\{B_1 \text{ or } B_2 \text{ or } \ldots \text{ or } B_k\} \leq \sum_i Pr\{B_i\} \tag{5}$$

   We now need to find $Pr\{B_i\}$

4. Define the variable $z_{i,j}$, which indicates that classifier c_i misclassifies the sample j, i.e.:

$$z_{i,j} \coloneqq \begin{cases} 0, & c_i \text{ incorrectly classifies } x_j \\ 1, & \text{otherwise} \end{cases}$$

Note that, for a given i, we have:

$$err_S(c_i) = \frac{1}{n} \sum_{j=1}^{n} z_{i,j} \tag{6}$$

$$err(c_i) = \frac{1}{n} \sum_{j=1}^{n} \mathbb{E}\, z_{i,j} = \frac{1}{n}\, \mathbb{E} \left[\sum_{j=1}^{n} z_{i,j} \right] \tag{7}$$

Here, (7) holds because the definition of the error of a classifier is precisely the probability that it makes a mistake in a random sample.

5. Using (4), (6) and (7) (and multiplying by n in both sides), we have:

$$Pr\{B_i\} = Pr\left(\left| \sum_{j=1}^{n} z_{i,j} - \mathbb{E} \left[\sum_{j=1}^{n} z_{i,j} \right] \right| \geq \frac{n\epsilon}{2} \right)$$

Using Chernoff Bound, we can then bound this probability as:

$$Pr\{B_i\} \lessapprox 2e^{-\epsilon^2 n} \tag{8}$$

6. We then return to the result from union bound, (5). Using it with (8), we have:

$$Pr\{B_1 \text{ or } B_2 \text{ or } \ldots \text{ or } B_k\} \lessapprox 2|\mathcal{C}|e^{-\epsilon^2 n}$$

7. Finally, we want to show that when no B_i happens, ERM returns a classifier $\bar{c}$ such that $err(\bar{c}) \leq err^* + \epsilon$.

Let $\tilde{c}$ be the best classifier in $\mathcal{C}$. The following holds:

$$err_S(\bar{c}) \leq err_S(\tilde{c})$$

because $\bar{c}$ is chosen as to minimize $err_S(\cdot)$. But since no B_i happens, it in particular doesn't happen for the classifier $\tilde{c}$, and therefore:

$$err_S(\bar{c}) \leq err(\tilde{c}) + \frac{\epsilon}{2} = err^* + \frac{\epsilon}{2}$$

Finally, since B_i also doesn't happen for the classifier $\bar{c}$ we have:

$$err(\bar{c}) \leq err_S(\bar{c}) + \frac{\epsilon}{2}$$

And therefore:

$$err(\bar{c}) \leq err^* + \epsilon$$

which concludes our proof.

□

2.2 Considering the case in which $|\mathcal{C}|$ is not finite

We now focus our attention in item (ii).

Question If $|\mathcal{C}| = \infty$, is learning (with guarantees) impossible?

Since this is not an easy question to answer, let us start with an example.

Example 2.3. *We will consider an example with the following characteristics:*

- $\mathcal{X} = [0, 1]$
- $\mu = \mathit{Uniform}(0, 1)$
- $\mathcal{C} = \{c_a : a \in [0, 1]\}$, $c_a(x) = \begin{cases} 1, & \text{if } x \geq a \\ 0, & \text{otherwise} \end{cases}$

  Note that $|\mathcal{C}| = \infty$

For simplicity, we will consider the realizable case, i.e. $c^* = c^*_{a^*} \in \mathcal{C}$

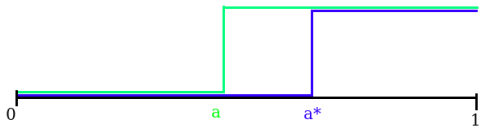

Figure 1: a and a^*

In Figure 1, the only region in which c_a *misses is if the sample falls between* a *and* a^**. Therefore, the error of such classifier (which corresponds to the probability of a misclassification) will be the area between* a *and* a^**.*

Suppose that we have, for example, 10 samples that are representative of the population. Since they are random, let's say that they are approximately equidistant, as in Figure 2

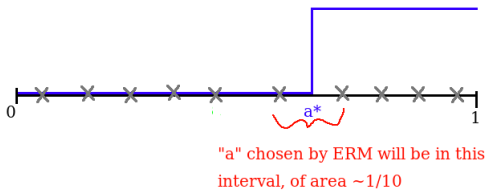

Figure 2: 10 representative samples

It follows that, on average, if our training data is sufficiently representative, we can see that the error is at most approximately $\frac{1}{10}$.

More generally, if we have n samples in this example, we will have:

$$err(ERM) \lessapprox \frac{1}{n}$$

So, at least in this simplified example, we are able to learn with guarantees (although we do not formally prove them here) even when $|\mathcal{C}| = \infty$.

Question Is this always possible?

Answer No. Consider the case in which $\mathcal{C}$ is too flexible (e.g. every possible classifier). We've seen in the previous lecture that in this case, learning with ERM is not possible.

In order to be able to formalize the difference between the two situations above, we need the notion of VC dimension, which will be introduced next.

3 VC dimension

The VC dimension, which stands for Vapnik – Chervonenkis dimension, can be viewed as a measure of the flexibility of a space of functios that can be learned by a statistical classification algorithm. This measure is related to the number of parameters, or the number of degrees of freedom of the family $\mathcal{C}$ of classifiers.

To illustrate: in Example 2.3 we had 1 degree of freedom in our family of classifiers. The VC dimension in this case is 1.

As another example, suppose that $\mathcal{C}$ is a family of linear classifiers in a space of dimension d. In this case, we have approximately $d + 1$ degrees of freedom, which is also the VC dimension.

Even though the VC dimension is related to these degrees of freedom, this is not immediately linked to its formal definition. Before formalizing the notion of VC dimension, we first must define shattering.

3.1 Shattered sets

Definition 3.1. We say that a set $U \subseteq \mathcal{X}$ is shattered by family $\mathcal{C}$ if the classifiers in $\mathcal{C}$ are able to give all the $2^{|U|}$ possible classifications of these points.

Example 3.2. *Suppose that our population $\mathcal{X} = \mathbb{R}^2$ and $\mathcal{C}$ is a family of linear classifiers.*

If U is the set of two points illustrated in Figure 3, we see that using $\mathcal{C}$ we are able to separate our set into every possible combination.

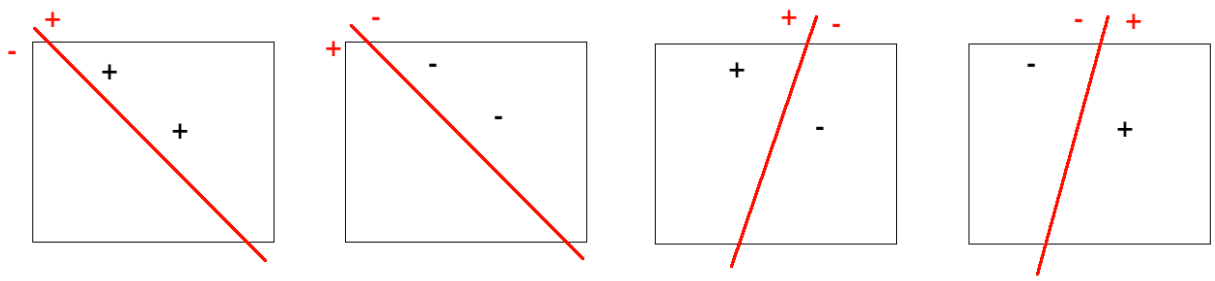

Figure 3: shattering of U using linear classifiers

Therefore, U is shattered by $\mathcal{C}$
This essentially shows that $\mathcal{C}$ shatters any set of two points in $\mathbb{R}^2$.

Example 3.3. *We now proceed to a case in which we are not able to shatter the set U.*

If U is a set of 4 points, $\mathcal{C}$ is not able to shatter it. Even though we won't formally prove this fact, Figure 4 aims to illustrate it for two cases. In both, it is impossible to shatter the set with $\mathcal{C}$.

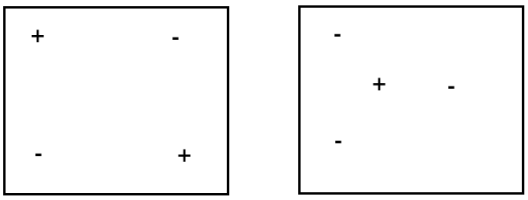

Figure 4: 4 points: impossible to shatter using linear classifiers

Going a little further in these examples, we can check the following results:

- In $\mathbb{R}^2$, $\exists$ a set U of 3 points that is shattered by $\mathcal{C}$, a family of linear classifiers.
- In $\mathbb{R}^3$, $\exists$ a set U of 4 points that is shattered by $\mathcal{C}$, a family of linear classifiers.
- In $\mathbb{R}^2$, $\nexists$ a set U of 4 points that is shattered by $\mathcal{C}$, a family of linear classifiers.
- In $\mathbb{R}^3$, $\nexists$ a set U of 5 points that is shattered by $\mathcal{C}$, a family of linear classifiers.

3.2 Formal definition of VC dimension

Definition 3.4. Let $\mathcal{X}$ be the population and $\mathcal{C}$ be the family of classifiers. The VC dimension is d if:

- $\exists$ set U, with $|U| = d$, shattered by $\mathcal{C}$
- $\nexists$ set U, with $|U| = d + 1$, shattered by $\mathcal{C}$

Example 3.5. *In our previous example, with $\mathcal{X} = \mathbb{R}^2$ and $\mathcal{C}$ the set of linear classifiers, the VC dimension is* 3.

To further understand this definition, we will see the VC dimension for many different families of classifiers.

3.3 Examples of VC dimension

Threshold Functions As in Example 2.3, we are interested in threshold functios as classifiers, i.e.:

$$c_a(x) = \begin{cases} 1, & \text{if } x \geq a \\ 0, & \text{otherwise} \end{cases}$$

There are no sets with 2 points that can be shattered by family $\mathcal{C}$, because the classification is not symmetric.

However, there exists a set U with $|U| = 1$ that can be shattered by $\mathcal{C}$.

Therefore, the VC dimension is 1.

Intervals Now, our classifier has two parameters:

$$c_{a,b}(x) = \begin{cases} 1, & \text{if } x \in [a, b] \\ 0, & \text{otherwise} \end{cases}$$

We can shatter sets of 2 points using this family of classifiers. However, it is not possible to shatter sets of 3 points.

Then, the VC dimension is 2.

Note that, once again, the VC dimension is equal to the number of parameters. This is not a formal link between the definitions, but often happens.

Linear classifiers in $\mathbb{R}^d$ In this case, the VC dimension is $d + 1$. We explored this idea in lower dimensions, but we won't prove the general result here.

We refer to [2] for a complete proof, which uses Radon's Theorem.

Decision Trees with continuous features, using Threshold functions Consider trees with height at most k.

We can think that, for each node, we have one parameter, which refers to the threshold. Note that we actually have more parameters, because in each node we also have the choice of which feature is being tested at each node, but for simplicity we will consider 1 parameter per node.

Therefore, $VCdimension \approx$ # nodes $= 2^k$

Finite $\mathcal{C}$ When our family of classifiers is finite, we will not be able to give an exact value for the VC dimension, but can we give an upper bound?

Suppose that we wish to shatter a set of m points. We then need at least 2^m possible classifications in our family. Therefore:

$$|\mathcal{C}| \geq 2^m$$

$$m \leq \log_2(|\mathcal{C}|)$$

Therefore, for finite $\mathcal{C}$, VC dimension $\leq \log_2(|\mathcal{C}|)$

$\mathcal{C}$ = every possible classifier, $\mathcal{X} = [0, 1]$ Note that since every classifier is in $\mathcal{C}$, this family shatters sets of any size.

This means that, in this case, $VCdimension = \infty$.

In this case, the family of classifier is so flexible that it becomes impossible to learn (with guarantees), at least using ERM.

Neural Networks with activation function ReLu and k parameters We won't prove here, but for this family of classifiers, $VCdimension \approx k$.

For proofs and further investigation on the VC dimension of neural networks, we refer to [1] and the lecture [3]

3.4 Sauer-Shelah lemma

So far, we have an intuition that the VC dimension is related to the degrees of freedom, and the examples seen support this idea. However, we still haven't answered the following question:

Question How does the VC dimension capture the flexibility of our classifiers? How can we use it to prove guarantees for the ERM?

Lemma 3.6. *(Sauer-Shelah)*
Suppose that the size of the universe is finite, i.e. $|\mathcal{X}| = k \leq \infty$, and that the VC dimension of our family of classifiers $\mathcal{C}$ is finite. Then, $|\mathcal{C}|$ is bounded as follows:

$$|\mathcal{C}| \leq \mathcal{O}(k)^{VCdim(\mathcal{C})} < \infty$$

Note that since $|\mathcal{X}| = k$, a simple, more direct result would be $|\mathcal{C}| \leq 2^k$.

Therefore, the lemma gives a much tighter bound: it is polynomial in k.

We will not prove the lemma here. The reader can check the proof in section $5.6.1$ of the book [5]. We also refer to a lecture that shows 3 different types of proof for the lemma [4].

Question How can we use this lemma if $|\mathcal{X}| = \infty$, such as $\mathcal{X} = \mathbb{R}^d$?

Answer The idea is that we only need to control the family $\mathcal{C}$ in the n samples of the population, with n finite. In the n samples, there are only $n^{VCdim(\mathcal{C})}$ different classifiers of $\mathcal{C}$. In general, $n^{VCdim(\mathcal{C})} \ll 2^n$, and thus this is a non-trivial and important result.

Recall from (3) that our guarantee for a solution $\bar{c}$ found using ERM was as follows:

$$err(ERM) \leq err^* + \sqrt{\frac{\log(|\mathcal{C}|)/\delta}{n}}$$

We can now substitute $|\mathcal{C}|$ by its upper bound, $n^{VCdim(\mathcal{C})}$

This remark informally leads us to the following theorem

Theorem 3.7. *(Vapnik-Chervonenkis) If $VCdim(\mathcal{C}) = d$, then with high probability we have the following (approximate) bound:*

$$err(ERM) \lessapprox err^* + \sqrt{\frac{d \cdot \log(n)}{n}}$$

Finally, we will give two remarks and a final example.

Remark 3.8. The $\log(n)$ term in Theorem 3.7 can in fact be removed, with the used of more advanced techniques that mix covering bounds and chaining methods (e.g. Dudley Chaining)

Remark 3.9. After such improvement, and by noting that $VCdim \leq \log(|\mathcal{C}|)$, this bounds becomes the bound we previously obtained for finite sets, given in equation (3). Hence, what we are doing is essentialy a generalization.

Example 3.10. *Linear classification in $\mathbb{R}^d$*

The error obtained will be

$$err(ERM) \lessapprox err^* + \sqrt{\frac{d}{n}}$$

As an example, with $n \approx 100d$, our approximation error would be around 0.1. If there are too many features (d is big), the n needed to guarantee a small error could be big.

However, on a more positive note, we note that if we have a margin γ (i.e. if it's possible to linearly classify the data using margin γ), our guarantee would be:

$$err(ERM) \lessapprox err^* + \frac{1}{\gamma}\sqrt{\frac{1}{n}}$$

which does not depend on the dimension d. Adding more dimensions (for example, using the Kernel Trick), the error does not increase.

This illustrates why SVM works very well in practice.

References

[1] P. L. Bartlett, V. Maiorov, and R. Meir. Almost linear vc dimension bounds for piecewise polynomial networks. In *Advances in Neural Information Processing Systems*, pages 190–196, 1999.

[2] S. Kakade and A. Tewar. Vc dimension and sauer's lemma. `http://ttic.uchicago.edu/~tewari/lectures/lecture11.pdf`, 2008.

[3] S. Kakade and A. Tewari. Vc dimension of multilayer neural networks, range queries. `http://ttic.uchicago.edu/~tewari/lectures/lecture12.pdf`, 2008.

[4] H. Q. Ngo. Three proofs of sauer-shelah lemma. `https://cse.buffalo.edu/~hungngo/classes/2010/711/lectures/sauer.pdf`, 2010.

[5] S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.