The Vapnik–Chervonenkis Dimension: Information versus Complexity in Learning

Yaser S. Abu-Mostafa
California Institute of Technology, Pasadena, CA 91125 USA

When feasible, learning is a very attractive alternative to explicit programming. This is particularly true in areas where the problems do not lend themselves to systematic programming, such as pattern recognition in natural environments. The feasibility of learning an unknown function from examples depends on two questions:

1. **Do the examples convey enough information to determine the function?**
2. **Is there a speedy way of constructing the function from the examples?**

These questions contrast the roles of information and complexity in learning. While the two roles share some ground, they are conceptually and technically different. In the common language of learning, the information question is that of generalization and the complexity question is that of scaling. The work of Vapnik and Chervonenkis (1971) provides the key tools for dealing with the information issue. In this review, we develop the main ideas of this framework and discuss how complexity fits in.

1 Introduction

We start by formalizing a simple setup for learning from examples. We have an *environment* such as the set of visual images, and we call the set X. In this environment we have a *concept* defined as a function $f: X \rightarrow \{0,1\}$, such as the presence or absence of a tree in the image. The goal of learning is to produce a *hypothesis*, also defined as a function $g: X \rightarrow \{0,1\}$, that approximates the concept f, such as a pattern recognition system that recognizes trees. To do this, we are given a number of examples $(x_1, f(x_1)), \ldots, (x_N, f(x_N))$ from the concept, such as images with trees and images without trees.

In generating the examples, we assume that there is an unknown probability distribution P on the environment X. We pick each example independently according to this probability distribution. The statements in the paper hold true for any probability distribution P, which sounds

Neural Computation **1, 312–317** (1989) © 1989 Massachusetts Institute of Technology

very strong indeed. The catch is that the same P that generated the example is the one that is used to test the system, which is a plausible assumption. Thus, we learn the tree concept by being exposed to "typical" images. While X can be finite or infinite (countable or uncountable), we shall use a simple language that assumes no measure-theoretic complications.

The hypothesis g that we produce approximates f in the sense that g would rarely be significantly different from f (Valiant 1984). This definition allows for two tolerance parameters ϵ and δ. With probability $\geq 1 - \delta$, g will differ from f at most ϵ of the time. The δ parameter protects against the small, but nonzero, chance that the examples happen to be very atypical.

A learning algorithm is one that takes the examples and produces the hypothesis. The performance is measured by the number of examples needed to produce a good hypothesis as well as the running time of the algorithm.

2 Generalization

We start with a simple case that may look at first as having little to do with what we think of as generalization. Suppose we make a blind guess of a hypothesis g, without even looking at any examples of the concept f. Now we take some examples of f and test g to find out how well it approximates f. Under what conditions does the behavior of g on the examples reflect its behavior in general?

This turns out to be a very simple question. On any point in X, f and g either agree or disagree. Define the agreement set

$$A = \{x \in X \ : \ f(x) = g(x)\}.$$

The question now becomes: How does the frequency of the examples in A relate to the probability of A? Let π be the probability of A, i.e., the probability that $f(x) = g(x)$ on a point x picked from X according to the probability distribution P. We can consider each example as a Bernoulli trial (coin flip) with probability π of success ($f = g$) and probability $1 - \pi$ of failure ($f \neq g$).

With N examples, we have N independent, identically distributed, Bernoulli trials. Let n be the number of successes (n is a random variable), and let $\nu = n/N$ be the frequency of success. Bernoulli's theorem states that, by taking N sufficiently large, ν can be made arbitrarily close to π with very high probability. In other words, if you take enough examples, the frequency of success will be a good estimate of the probability of success.

Notice that this does not say anything about the probability of success itself, but rather about how the probability of success can be estimated from the frequency of success. If on the examples we get 90% right, we

should get about 90% right overall. If we get only 10% right, we should continue to get about the same. We are only predicting that the results of the experiment with the examples will persist, provided there are enough examples.

How does this case relate to learning and generalization? After all, we do not make a blind guess when we learn, but rather construct a hypothesis from the examples. However, at a closer look, we find that we make a guess, not of a hypothesis but of a set of hypotheses. For example, when the backpropagation algorithm (Rumelhart *et al.* 1986) is used in a feedforward network, we are implicitly guessing that there is a good hypothesis among those that are obtained by setting the weights of the given network in some fashion. The set of hypotheses G would then be the set of all functions g that are obtained by setting the weights of the network in any fashion.

Therefore, when learning deals with a limited domain of representation, such as a given network with free weights, we in effect make a guess G of hypotheses. The learning algorithm then picks a hypothesis $g \in G$ that mostly agrees with f on the examples. The question of generalization now becomes: Does this choice, which is based on the behavior on the examples, hold in general?

We can approach this question in a way similar to the previous case. We define, for every $g \in G$, the agreement set

$$A_g = \{x \in X \mid f(x) = g(x)\}.$$

These sets are different for different gs. Let π_g be the probability of A_g, i.e., the probability that $f(x) = g(x)$ on a point x picked from X according to the probability distribution P, for the particular $g \in G$ in question. We can again define random variables n_g (the number of successes with respect to different gs) and the frequencies of success $\nu_g = n_g/N$. At this point the problem looks exactly the same as the previous one and one may expect the same answer.

There is one important difference. In the simple Bernoulli case, the issue was whether ν converged to π. In the new case, the issue is whether the ν_gs converge to the π_gs *in a uniform manner* as N becomes large. In the learning process, we decide on one g but not the other based on the values of ν_g. If we had the ν_gs converge to the π_gs, but not in a uniform manner, we could be fooled by one erratic g. For example, we may be picking the hypothesis g with the maximum ν_g. With nonuniform convergence, the g we pick can have a poor π_g. We want the probability that *there is some* $g \in G$ such that ν_g differs significantly from π_g be very small. This can be expressed formally as

$$\Pr\left[\sup_{g \in G} |\nu_g - \pi_g| > \epsilon\right] \leq \delta$$

where sup denotes the supremum.

3 The V–C Dimension

A condition for uniform convergence, hence generalization, was found by Vapnik and Chervonenkis (1971). The key is the inequality

$$\Pr\left[\sup_{g\in G}|\nu_g - \pi_g| > \epsilon\right] \leq 4m(2N)e^{-\epsilon^2 N/8},$$

where m is a function that depends on G. We want the right-hand side of the inequality to be small for large N, in order to achieve uniform convergence. The factor $e^{-\epsilon^2 N/8}$ is very helpful, since it is exponentially decaying in N. Unless the factor $m(2N)$ grows too fast, we should be OK. For example, if $m(2N)$ is polynomial in N, the right-hand side will go to zero as N goes to infinity.

What is the function m? It depends on the set of hypotheses G. Intuitively, $m(N)$ measures the flexibility of G in expressing an arbitrary concept on N examples. For instance, if G contains enough hypotheses to be able to express any concept on 100 examples, one should not really expect any generalization with only 100 examples, but rather a memorization of the concept on the examples. On the other hand, if gradually more and more concepts cannot be expressed by any hypothesis in G as N grows, then the agreement on the examples means something, and generalization is probable. Formally, $m(N)$ measures the maximum number of *different* binary functions on the examples $x_1, \ldots, x_N$ induced by the hypotheses $g_1, g_2, \ldots \in G$.

For example, if X is the real line and G is the set of rays of the form $x \leq a$, i.e., functions of the form

$$g(x) = \begin{cases} 0 & x \leq a \\ 1 & x > a \end{cases},$$

then $m(N) = N + 1$. The reason is that on N points one can define only $N+1$ different functions of the above form by sliding the value of a from left of the leftmost point all the way to right of the rightmost point.

There are two simple facts about the function m. First, $m(N) \leq |G|$ (where $|\cdot|$ denotes the cardinality), since G cannot induce more functions that it has. This fact is useful only when G is a finite set of hypotheses. The second fact is that $m(N) \leq 2^N$, since G cannot induce more binary functions on N points than there are binary functions on N points. Indeed, there are choices of G (trivially the set of all hypotheses on X) for which $m(N) = 2^N$. For those cases, the V–C inequality does not guarantee uniform convergence.

The main fact about $m(N)$ that helps the characterization of G as far as generalization is concerned is that $m(N)$ is either identically equal to 2^N for all N, or else is bounded above by $N^d + 1$ for a constant d. This striking fact can be proved in a simple manner (Cover 1965; Vapnik and Chervonenkis 1971). The latter case implies a polynomial $m(N)$

and guarantees generalization. The value of d matters only in how fast convergence is achieved. This is of practical importance because this determines the number of examples needed to guarantee generalization within given tolerance parameters.

The value of d turns out to be the smallest N at which G starts failing to induce all possible 2^N binary functions on any N examples. Thus, the former case can be considered the case $d = \infty$. d is called the V–C dimension (Baum and Haussler 1989; Blumer *et al.* 1986).

4 Interpretation

Training a network with a set of examples can be thought of as a process for selecting a hypothesis g with a favorable performance on the examples (large ν_g) from the set G. Depending on the characteristics of G, one can predict how this performance will generalize. This aspect of the characteristics of G is captured by the parameter d, the V–C dimension. If the number of examples N is large enough with respect to d, generalization is expected. This means that maximizing ν_g will approximately maximize π_g, the real indicator of how well the hypothesis approximates the concept.

In general, the more flexible (expressive, large) G is, the larger its V–C dimension d. For example, the V–C dimension of feedforward networks grows with the network size (Baum and Haussler 1989). For example, the total number of weights in a one-hidden-layer network is an approximate lower bound for the V–C dimension of the network. While a bigger network stands a better chance of being able to implement a given function, its demands on the number of examples needed for generalization is bigger. These are often conflicting criteria. The V–C dimension indicates only the likelihood of generalization. This means, for better or for worse, whether the behavior on the examples is going to persist. The ability of the network to approximate a given function in principle is a separate issue.

The running time of the learning algorithm is a key concern (Judd 1988; Valiant 1984). As the number of examples increases, the running time generally increases. However, this dependency is a minor one. Even with few examples, an algorithm may need an excessive amount of time to manipulate the examples into a hypothesis. The independence of this complexity issue from the above discussion regarding information is apparent. Without a sufficient number of examples, no algorithm slow or fast can produce a good hypothesis. Yet a sufficient number of examples is of little use if the computational task of digesting the examples into a hypothesis proves intractable.

Acknowledgments

The support of the Air Force Office of Scientific Research under Grant AFOSR-88-0213 is gratefully acknowledged.

References

Baum, E.B., and Haussler, D. 1989. What size network gives valid generalization. *Neural Comp.* **1**, 151–160. MIT Press, Cambridge, MA.

Blumer, A., Ehrenfeucht, A., Haussler, D., and Warmuth, M. 1986. Classifying learnable geometric concepts with the Vapnik–Chervonenkis dimension. *Proc. ACM Symp. Theory Computing* **18**, 273–282.

Cover, T.M. 1965. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Trans. Electronic Comput.* 326–334.

Judd, J.S. 1988. On the complexity of loading shallow neural networks. *J. Complex.* **4**, 177–192.

Rumelhart, D.E., Hinton, G.E., and Williams, R.J. 1986. Learning internal representations by error propagation. In *Parallel Distributed Processing*, Vol. 1. MIT Press, Cambridge, MA.

Valiant, L.G. 1984. A theory of the learnable. *Commun. ACM* **27**, 1134–1142.

Vapnik, V.N., and Chervonenkis, A. 1971. On the uniform convergence of relative frequencies of events to their probabilities. *Theory Prob. Appl.* **16**, 264–280.

Received 6 July 1989; accepted 25 July 1989.