U

Underlying Objective

The term *objective* used in Evolutionary Multi-Objective Optimization refers to an indicator of quality returning an element from an ordered set of scalar values, such as a real number. For any test-based coevolutionary problem, a set of underlying objectives exists such that knowledge of the objective values of an individual is sufficient to determine the outcomes of all possible tests. The existence of a set of underlying objectives is guaranteed, as the set of possible tests itself satisfies this property.

Unit

►Neuron

Universal Learning Theory

Marcus Hutter
Australian National University, Canberra, Australia

Definition, Motivation, and Background

Universal (machine) learning is concerned with the development and study of algorithms that are able to learn from data in a very large range of environments with as few assumptions as possible. The class of environments typically considered includes all computable stochastic processes. The investigated learning tasks range from ►inductive inference, sequence prediction, sequential decisions, to (re)active problems like ►reinforcement learning (Hutter, 2005), but also include ►clustering, ►regression, and others (Li & Vitányi, 2008). Despite various no-free-lunch theorems (Wolpert & Macready, 1997), universal learning is *possible* by assuming that the data possess *some* effective structure, but without specifying any further, *which* structure. Learning algorithms that are universal (at least to some degree) are also *necessary* for developing autonomous general intelligent systems, required, for example, for exploring other planets, as opposed to decision *support* systems which keep a human in the loop. There is also an *intrinsic* interest in striving for generality: Finding new learning algorithms for every particular (new) problem is possible but cumbersome and prone to disagreement or contradiction. A sound, formal, general and ideally complete theory of learning can unify existing approaches, guide the development of practical learning algorithms, and last but not least lead to novel and deep insights.

Deterministic Environments

Let $t, n \in \mathbb{N}$ be natural numbers, $\mathcal{X}^*$ be the set of finite strings and $\mathcal{X}^\infty$ be the set of infinite sequences over some alphabet $\mathcal{X}$ of size $|\mathcal{X}|$. For a string $x \in \mathcal{X}^*$ of length $\ell(x) = n$ we write $x_1 x_2 \ldots x_n$ with $x_t \in \mathcal{X}$, and further abbreviate $x_{t:n} := x_t x_{t+1} \ldots x_{n-1} x_n$ and $x_{<n} := x_1 \ldots x_{n-1}$, and $\epsilon = x_{<1}$ for the empty string. Consider a countable class of hypotheses $\mathcal{M} = \{H_1, H_2, \ldots\}$. Each hypothesis $H \in \mathcal{M}$ (also called model) shall describe an infinite sequence $x^H_{1:\infty}$, for example, like in IQ test questions "2, 4, 6, 8, …." In online learning, for $t = 1, 2, 3, \ldots$, we predict x_t based on past observations $\dot{x}_{<t}$, then nature reveals $\dot{x}_t$, and so on, where the dot above x indicates the true observation. We assume that the true hypothesis is in $\mathcal{M}$, that is, $\dot{x}_{1:\infty} = x^{H_m}_{1:\infty}$ for some $m \in \mathbb{N}$. Goal is to ("quickly") identify the unknown H_m from the observations.

Learning by enumeration works as follows: Let $\mathcal{M}_t = \left\{H \in \mathcal{M} : x^H_{<t} = \dot{x}_{<t}\right\}$ be the set of hypotheses consistent with our observations $\dot{x}_{<t}$ so far. The

Claude Sammut & Geoffrey I. Webb (eds.), *Encyclopedia of Machine Learning*, DOI 10.1007/978-0-387-30164-8,

hypothesis in $\mathcal{M}_t$ with smallest index, say m'_t, is selected and used for predicting x_t. Then x_t is observed and all $H \in \mathcal{M}_t$ inconsistent with x_t are eliminated, that is, they are not included in $\mathcal{M}_{t+1}$. Every prediction error results in the elimination of at least $H_{m'_t}$, so after at most $m-1$ errors, the true hypothesis H_m gets selected forever, since it never makes an error ($H_m \in \mathcal{M}_t \; \forall t$). This identification may take arbitrarily long (in t), but the number of errors on the way is bounded by $m-1$, and the latter is often more important. As an example for which the bound is attained, consider H_i with $x^{H_i}_{1:\infty} := 1^{f(i)}0^\infty \; \forall i$ for any strictly increasing function f, for example, $f(i) = i$. But we now show that we can do much better than this, at least for finite $\mathcal{X}$.

Majority learning:

Consider (temporarily in this paragraph only) a binary alphabet $\mathcal{X} = \{0,1\}$ and a *finite* deterministic hypothesis class $\mathcal{M} = \{H_1, H_2, \ldots, H_N\}$. H_m and $\mathcal{M}_t$ are as before, but now we take a majority vote among the hypotheses in $\mathcal{M}_t$ as our prediction of x_t. If the prediction turns out to be wrong, then at least half (the majority) of the hypotheses get eliminated from $\mathcal{M}_t$. Hence after at most $\log N$ errors, there is only a single hypothesis, namely H_m, left over. So this majority predictor makes at most $\log N$ errors. As an example where this bound is essentially attained, consider $m = N = 2^n - 1$ and let $x^{H_i}_{1:\infty}$ be the digits after the comma of the binary expansion of $(i-1)/2^n$ for $i = 1, \ldots, N$.

Weighted majority for countable classes:

Majority learning can be adapted to denumerable classes $\mathcal{M}$ and general finite alphabet $\mathcal{X}$ as follows: Each hypothesis H_i is assigned a weight $w_i > 0$ with $\sum_i w_i \leq 1$. Let $W := \sum_{i:H_i \in \mathcal{M}_t} w_i$ be the total weight of the hypotheses in $\mathcal{M}_t$. Let $\mathcal{M}^a_t := \{H_i \in \mathcal{M}_t : x^{H_i}_t = a\}$ be the consistent hypotheses predicting $x_t = a$, and W_a their weight, and take the weighted majority prediction $x_t = \arg\max_a W_a$. Similarly as above, a prediction error decreases W by a factor of $1 - 1/|\mathcal{X}|$, since $\max_a W_a \geq W/|\mathcal{X}|$. Since $w_m \leq W \leq 1$, this algorithm can at most make $\log_{1-1/|\mathcal{X}|} w_m = O\left(\log w_m^{-1}\right)$ prediction errors. If we choose, for instance, $w_i = (i+1)^{-2}$, the number of errors is $O(\log m)$, which is an exponential improvement over the Gold-style learning by enumeration above.

Algorithmic Probability

Algorithmic probability has been founded by Solomonoff (1964). The so-called universal probability or a-priori probability is the key quantity for universal learning. Its philosophical and technical roots are ►Ockham's razor (choose the simplest model consistent with the data), Epicurus' principle of multiple explanations (keep all explanations consistent with the data), (Universal) Turing machines (to compute, quantify and assign codes to all quantities of interest), and Kolmogorov complexity (to define what simplicity/complexity means). This section considers deterministic computable sequences, and the next section the general setup of computable probability distributions.

(Universal) monotone Turing machines: Since we consider infinite computable sequences, we need devices that convert input data streams to output data streams. For this we define the following variants of a classical deterministic Turing machine: A monotone Turing machine T is defined as a Turing machine with one unidirectional input tape, one unidirectional output tape, and some bidirectional work tapes. The input tape is binary (no blank) and read only; the output tape is over finite alphabet $\mathcal{X}$ (no blank) and write only; unidirectional tapes are those where the head can only move from left to right; work tapes are initially filled with zeros and the output tape with some fixed element from $\mathcal{X}$. We *say* that *monotone Turing machine* T outputs/computes a string starting with x on input p, and write $T(p) = x*$ if p is to the left of the input head when the last bit of x is output (T reads all of p but no more). T may continue operation and need not halt. For a given x, the set of such p forms a prefix code. Such codes are called *minimal* programs. Similarly, we write $T(p) = \omega$ if p outputs the infinite sequence ω. A *prefix code* $\mathcal{P}$ is a set of binary strings such that no element is proper prefix of another. It satisfies *Kraft's inequality* $\sum_{p \in \mathcal{P}} 2^{-\ell(p)} \leq 1$.

The table of rules of a Turing machine T can be prefix encoded in a canonical way as a binary string, denoted by $\langle T \rangle$. Hence, the set of Turing machines $\{T_1, T_2, \ldots\}$ can be effectively enumerated. There are so-called universal Turing machines that can "simulate" all other Turing machines. We define a particular one which simulates monotone Turing machine $T(q)$ if fed

with input $\langle T\rangle q$, that is, $U(\langle T\rangle q) = T(q)\ \forall T, q$. Note that for p not of the form $\langle T\rangle q$, $U(p)$ does not output anything. We call this particular U the *reference universal Turing machine*.

Universal weighted majority learning: $T_1(\epsilon), T_2(\epsilon), \ldots$ constitutes an effective enumeration of all finite and infinite computable sequences, hence also monotone $U(p)$ for $p \in \{0,1\}^*$. As argued below, the class of computable infinite sequences is conceptually very interesting. The halting problem implies that there is no recursive enumeration of all partial-recursive functions with infinite domain; hence we cannot remove the finite sequences algorithmically. It is very fortunate that we don't have to. Hypothesis H_p is identified with the sequence $U(p)$, which may be finite, infinite, or possibly even empty. The class of considered hypotheses is $\mathcal{M} := \{H_p : p \in \{0,1\}^*\}$.

The weighted majority algorithm also needs weights w_p for each H_p. Ockham's razor combined with Epicurus' principle demand to assign a high (low) prior weight to a simple (complex) hypothesis. If complexity is identified with program length, then w_p should be a decreasing function of $\ell(p)$. It turns out that $w_p = 2^{-\ell(p)}$ is the "right" choice, since minimal p forms a prefix code and therefore $\sum_p w_p \leq 1$ as required.

Using H_p for prediction can now fail in two ways. H_p may make a wrong prediction or no prediction at all for x_t. The true hypothesis H_m is still assumed to produce an infinite sequence. The weighted majority algorithm in this setting makes at most $O\left(\log w_p^{-1}\right) = O(\ell(p))$ errors. It is also plausible that learning $\ell(p)$ bits requires $O(\ell(p))$ "trials."

Universal mixture prediction: Solomonoff (1978) defined the following universal a-priori probability

$$M(x) := \sum_{p:U(p)=x*} 2^{-\ell(p)}. \qquad (1)$$

That is, $M(x) = W$ is the total weight of the computable deterministic hypotheses consistent with x for the universal weight choice $w_p = 2^{-\ell(p)}$. The universal weighted majority algorithm predicted $\arg\max_a M(\dot{x}_{<t}a)$. Instead, one could also make a probability prediction $M(a|\dot{x}_{<t}) := M(\dot{x}_{<t}a)/M(\dot{x}_{<t})$, which is the relative weight of hypotheses in $\mathcal{M}_t$ predicting a. The higher the probability $M(\dot{x}_t|\dot{x}_{<t})$ that is assigned to the true next observation $\dot{x}_t$, the better. Consider the absolute prediction error $|1 - M(\dot{x}_t|\dot{x}_{<t})|$ and the logarithmic error $-\log M(\dot{x}_t|\dot{x}_{<t})$. The cumulative logarithmic error is bounded by $\sum_{t=1}^{n} -\log M(\dot{x}_t|\dot{x}_{<t}) = -\log M(\dot{x}_{1:n}) \leq \ell(p)$ for any program p that prints $\dot{x}*$. For instance, p could be chosen as the shortest one printing $\dot{x}_{1:\infty}$, which has length $Km(\dot{x}_{1:\infty}) := \min\{\ell(p) : U(p) = \dot{x}_{1:\infty}\}$. Using $1 - z \leq -\log z$ and letting $n \to \infty$ we get

$$\sum_{t=1}^{\infty} |1 - M(\dot{x}_t|\dot{x}_{<t})| \leq \sum_{t=1}^{\infty} -\log M(\dot{x}_t|\dot{x}_{<t}) \leq Km(\dot{x}_{1:\infty}).$$

Hence again, the cumulative absolute and logarithmic errors are bounded by the number of bits required to describe the true environment.

Universal Bayes

The exposition so far has dealt with deterministic environments only. Data sequences produced by real-world processes are rarely as clean as IQ test sequences. They are often noisy. This section deals with stochastic sequences sampled from computable probability distributions. The developed theory can be regarded as an instantiation of Bayesian learning. Bayes' theorem allows to update beliefs in face of new information but is mute about how to choose the prior and the model class to begin with. Subjective choices based on prior knowledge are informal, and traditional "objective" choices like Jeffrey's prior are not universal. Machine learning, the computer science branch of statistics, develops (fully) automatic inference and decision algorithms for very large problems. Naturally, machine learning has (re)discovered and exploited different principles (Ockham's and Epicurus') for choosing priors, appropriate for this situation. This leads to an alternative representation of universal probability as a mixture over all lower semi-computable semimeasures with Kolmogorov complexity-based prior as described below.

Bayes

Sequences $\omega = \omega_{1:\infty} \in \mathcal{X}^\infty$ are now assumed to be sampled from the "true" probability measure μ, that is, $\mu(x_{1:n}) := \mathrm{P}[\omega_{1:n} = x_{1:n}|\mu]$ is the μ-probability that ω starts with $x_{1:n}$. Expectations w.r.t. μ are denoted by $\mathbf{E}$. In particular for a function $f : \mathcal{X}^n \to \mathbb{R}$, we have $\mathbf{E}[f] = \mathbf{E}[f(\omega_{1:n})] = \sum_{x_{1:n}} \mu(x_{1:n}) f(x_{1:n})$.

Note that in Bayesian learning, measures, environments, and models are the same objects; let $\mathcal{M} = \{\nu_1, \nu_2, \ldots\} \equiv \{H_{\nu_1}, H_{\nu_2}, \ldots\}$ denotes a countable class of these measures≡hypotheses. Assume that μ is unknown but known to be a member of $\mathcal{M}$, and $w_\nu := \mathrm{P}[H_\nu]$ is the given prior belief in H_ν. Then the Bayesian mixture

$$\begin{aligned}\xi(x_{1:n}) &:= \mathrm{P}[\omega_{1:n} = x_{1:n}] \\ &= \sum_{\nu\in\mathcal{M}} \mathrm{P}[\omega_{1:n} = x_{1:n}|H_\nu]\mathrm{P}[H_\nu] \\ &\equiv \sum_{\nu\in\mathcal{M}} \nu(x_{1:n})w_\nu\end{aligned}$$

must be our a-priori belief in $x_{1:n}$, and $\mathrm{P}[H_\nu|\omega_{1:n} = x_{1:n}] = w_\nu\nu(x_{1:n})/\xi(x_{1:n})$ be our posterior belief in ν by Bayes' rule.

Universal Choice of $\mathcal{M}$

Next, we need to find a universal class of environments $\mathcal{M}_U$. Roughly speaking, Bayes' works if $\mathcal{M}$ contains the true environment μ. The larger $\mathcal{M}$, the less restrictive is this assumption. The class of all computable distributions, although only countable, is pretty large from a practical point of view, since it includes, for instance, all of today's valid physics theories. (Finding a non-computable physical system would indeed overturn the generally accepted Church-Turing thesis.) It is the largest class, relevant from a computational point of view. Solomonoff (1964, Eq. (13)) defined and studied the mixture over this class.

One problem is that this class is not (effectively = recursively) enumerable, since the class of computable functions is not enumerable due to the halting problem, nor is it decidable whether a function is a measure. Hence ξ is completely incomputable. Leonid Levin (Zvonkin & Levin, 1970) had the idea to "slightly" extend the class and include also lower semi-computable semimeasures.

A function $\nu : \mathcal{X}^* \to [0,1]$ is called a semimeasure iff $\nu(x) \geq \sum_{a\in\mathcal{X}} \nu(xa)\ \forall x \in \mathcal{X}^*$. It is a proper probability measure iff equality holds and $\nu(\epsilon) = 1$. $\nu(x)$ still denotes the ν-probability that a sequence starts with string x. A function is called lower semi-computable, if it can be approximated from below. Similarly to the fact that the class of partial recursive functions is recursively enumerable, one can show that the class $\mathcal{M}_U = \{\nu_1, \nu_2, \ldots\}$ of lower semi-computable semimeasures is recursively enumerable. In some sense $\mathcal{M}_U$ is the largest class of environments for which ξ is in some sense computable, but even larger classes are possible (Schmidhuber, 2002).

Kolmogorov Complexity

Before we can turn to the prior w_ν, we need to quantify complexity/simplicity. Intuitively, a string is simple if it can be described in a few words, like "the string of one million ones," and is complex if there is no such short description, like for a random object whose shortest description is specifying it bit by bit. We are interested in effective descriptions, and hence restrict decoders to be Turing machines. One can define the *prefix Kolmogorov complexity* of string x as the length ℓ of the shortest *halting* program p for which U outputs x:

$$K(x) := \min_p\{\ell(p) : U(p) = x \text{ halts}\}.$$

Simple strings like 000…0 can be generated by short programs, and hence have low Kolmogorov complexity, but irregular (e.g., random) strings are their own shortest description, and hence have high Kolmogorov complexity. For non-string objects o (like numbers and functions) one defines $K(o) := K(\langle o\rangle)$, where $\langle o\rangle \in \mathcal{X}^*$ is some standard code for o. In particular, $K(\nu_i) = K(i)$.

To be brief, K is an excellent universal complexity measure, suitable for quantifying Ockham's razor.

The Universal Prior

We can now quantify a prior biased toward simple models. First, we quantify the complexity of an environment ν or hypothesis H_ν by its Kolmogorov complexity $K(\nu)$. The universal prior should be a decreasing function in the model's complexity, and of course sum to (less than) one. Since $\sum_x 2^{-K(x)} \leq 1$ by the prefix property and Kraft's inequality, this suggests the choice

$$w_\nu = w_\nu^U := 2^{-K(\nu)}. \tag{2}$$

Since $\log i \leq K(\nu_i) \leq \log i + 2\log\log i$ for "most" i, most ν_i have prior approximately reciprocal to their index i as also advocated by Jeffreys and Rissanen.

Representations

Combining the universal class $\mathcal{M}_U$ with the universal prior 2, we arrive at the universal mixture

$$\xi_U(x) := \sum_{\nu\in\mathcal{M}_U} 2^{-K(\nu)}\nu(x) \tag{3}$$

which has remarkable properties. First, it is itself a lower semi-computable semimeasure, that is $\xi_U \in \mathcal{M}_U$, which is very convenient. Note that for most classes, $\xi \notin \mathcal{M}$.

Second, ξ_U coincides with M within an irrelevant multiplicative constant, and $M \in \mathcal{M}_U$. This means that the mixture over deterministic computable sequences is as rich as the mixture over the much larger class of semi-computable semimeasures. The intuitive reason is that the probabilistic semimeasures are in the convex hull of the deterministic ones, and therefore need not be taken extra into account in the mixture.

There is another, possibly the simplest, representation: One can show that $M(x)$ is equal to the probability that U outputs a string starting with x when provided with uniform random noise on the program tape. Note that a uniform distribution is also used in many no-free-lunch theorems to prove the impossibility of universal learners, but in our case the uniform distribution is piped through a universal Turing machine, which defeats these negative implications as we will see in the next section.

Applications

In the stochastic case, identification of the true hypothesis is problematic. The posterior $P[H|x]$ may not concentrate around the true hypothesis H_μ if there are other hypotheses H_ν that are not asymptotically distinguishable from H_μ. But even if model identification (*induction* in the narrow sense) fails, *predictions*, *decisions*, and *actions* can be good, and indeed, for universal learning this is generally the case.

Universal Sequence Prediction

Given a sequence $x_1 x_2 \ldots x_{t-1}$, we want to predict its likely continuation x_t. We assume that the strings which have to be continued are drawn from a computable "true" probability distribution μ. The maximal prior information a prediction algorithm can possess is the exact knowledge of μ, but often the true distribution is unknown. Instead, prediction is based on a guess ρ of μ. Let $\rho(a|x) := \rho(xa)/\rho(x)$ be the "predictive" ρ-probability that the next symbol is $a \in \mathcal{X}$, given sequence $x \in \mathcal{X}^*$. Since $\mu \in \mathcal{M}_U$ it is natural to use ξ_U or M for prediction.

Solomonoff's (Hutter, 2005; Solomonoff, 1978) celebrated result indeed shows that M converges to μ. For general alphabet it reads

$$\sum_{t=1}^{\infty} \mathbf{E}\left[\sum_{a\in\mathcal{X}} \left(M(a|\omega_{<t}) - \mu(a|\omega_{<t})\right)^2\right] \leq K(\mu)\ln 2 + O(1). \tag{4}$$

Analogous bounds hold for ξ_U and for other than the Euclidean distance, for example, the Hellinger and the absolute distance and the relative entropy.

For a sequence $a_1, a_2, \ldots$ of random variables, $\sum_{t=1}^{\infty} \mathbf{E}\left[a_t^2\right] \leq c < \infty$ implies $a_t \to 0$ for $t \to \infty$ with μ-probability 1 (w.p.1). Convergence is rapid in the sense that the probability that a_t^2 exceeds $\varepsilon > 0$ at more than $c/\varepsilon\delta$ times is bounded by δ. This might loosely be called the number of errors. Hence Solomonoff's bounds implies

$$M(x_t|\omega_{<t}) - \mu(x_t|\omega_{<t}) \longrightarrow 0 \text{ for any } x_t \text{ rapid w.p.1 for } t \to \infty.$$

The number of times M deviates from μ by more than $\varepsilon > 0$ is bounded by $O(K(\mu))$, that is, proportional to the complexity of the environment, which is again reasonable. A counting argument shows that $O(K(\mu))$ errors for most μ are unavoidable. No other choice for w_ν would lead to significantly better bounds. Again, in general it is not possible to determine *when* these "errors" occur. Multi-step lookahead convergence $M(x_{t:n_t}|\omega_{<t}) - \mu(x_{t:n_t}|\omega_{<t}) \to 0$ even for unbounded lookahead $n_t - t \geq 0$, relevant for delayed sequence prediction and in reactive environments, can also be shown.

In summary, M is an excellent sequence predictor under the only assumption that the observed sequence is drawn from some (unknown) computable probability distribution. No ergodicity, stationarity, or identifiability or other assumption is required.

Universal Sequential Decisions

Predictions usually form the basis for decisions and actions, which result in some profit or loss. Let $\ell_{x_t y_t} \in [0,1]$ be the received loss for decision $y_t \in \mathcal{Y}$ when $x_t \in \mathcal{X}$ turns out to be the true tth symbol of the sequence. The ρ-optimal strategy

$$y_t^{\Lambda_\rho}(\omega_{<t}) := \arg\min_{y_t} \sum_{x_t} \rho(x_t|\omega_{<t})\ell_{x_t y_t} \tag{5}$$

minimizes the ρ-expected loss. For instance, if we can decide among $\mathcal{Y} = \{sunglasses, umbrella\}$ and it turns out to be $\mathcal{X} = \{sun, rain\}$, and our personal loss matrix is $\ell = \left(\begin{smallmatrix} 0.0 & 0.1 \\ 1.0 & 0.3 \end{smallmatrix}\right)$, then Λ_ρ takes $y_t^{\Lambda_\rho} = sunglasses$ if $\rho(rain|\omega_{<t}) < 1/8$ and an *umbrella* otherwise. For $\mathcal{X} = \mathcal{Y}$ and 0–1 loss $\ell_{xy} = 0$ for $x = y$ and 1 else, Λ_ρ predicts the most likely symbol $y_t^{\Lambda_\rho} = \arg\max_a \rho(a|\omega_{<t})$ as in Sect. 2.

The cumulative μ(=true)-expected loss of Λ_ρ for the first n symbols is

$$\mathrm{Loss}_n^{\Lambda_\rho} := \sum_{t=1}^{n} \mathbf{E}\left[\ell_{\omega_t y_t^{\Lambda_\rho}(\omega_{<t})}\right] \equiv \sum_{t=1}^{n}\sum_{x_{1:t}} \mu(x_{1:t})\ell_{x_t y_t^{\Lambda_\rho}(x_{<t})}.$$

If μ is known, Λ_μ obviously results in the best decisions in the sense of achieving minimal expected loss among all strategies. For the predictor Λ_M based on M (and similarly ξ_U), one can show

$$\sqrt{\mathrm{Loss}_n^{\Lambda_M}} - \sqrt{\mathrm{Loss}_n^{\Lambda_\mu}} \le \sqrt{2K(\mu)\ln 2 + O(1)} \quad (6)$$

This implies that $\mathrm{Loss}_n^{\Lambda_M}/\mathrm{Loss}_n^{\Lambda_\mu} \to 1$ for $\mathrm{Loss}_n^{\Lambda_\mu} \to \infty$, or if $\mathrm{Loss}_\infty^{\Lambda_\mu}$ is finite, then also $\mathrm{Loss}_\infty^{\Lambda_M} < \infty$. This shows that M (via Λ_M) also performs excellent from a decision-theoretic perspective, that is, suffers loss only slightly larger than the optimal Λ_μ strategy.

One can also show that Λ_M is pareto-optimal (admissible) in the sense that every other predictor with smaller loss than Λ_M in some environment $\nu \in \mathcal{M}_U$ must be worse in another environment.

Universal Classification and Regression

The goal of classification and regression is to infer the functional relationship $f : \mathcal{Y} \to \mathcal{X}$ from data $\{(y_1, x_1), \ldots, (y_n, x_n)\}$. In a predictive online setting one wants to "directly" infer x_t from y_t given $(y_{<t}, x_{<t})$ for $t = 1, 2, 3, \ldots$. The universal induction framework has to be extended by regarding $y_{1:\infty}$ as independent side-information presented in the form of an oracle or extra tape information or extra parameter. The construction has to ensure that $x_{1:n}$ depends only on $y_{1:n}$ but is (functionally or statistically) independent of $y_{n+1:\infty}$.

First, we augment a monotone Turing machine with an extra input tape containing $y_{1:\infty}$. The Turing machine is called chronological if it does not read beyond $y_{1:n}$ before $x_{1:n}$ has been written. Second, semimeasures $\rho = \mu, \nu, M, \xi_U$ are extended to $\rho(x_{1:n}|y_{1:\infty})$, that is, one semimeasure $\rho(\cdot|y_{1:\infty})$ for each $y_{1:\infty}$ (no distribution over y is assumed). Any such semimeasure must be chronological in the sense that $\rho(x_{1:n}|y_{1:\infty})$ is independent of y_t for $t > n$, hence we can write $\rho(x_{1:n}|y_{1:n})$. In classification and regression, ρ is typically (conditionally) i.i.d., that is, $\rho(x_{1:n}|y_{1:n}) = \prod_{t=1}^{n}\rho(x_t|y_t)$, which is chronological, but note that the Bayesian mixture ξ is *not* i.i.d. One can show that the class of lower semi-computable chronological semimeasures $\mathcal{M}_U^{|} = \{\nu_1(\cdot|\cdot), \nu_2(\cdot|\cdot), \ldots\}$ is effectively enumerable.

The generalized universal a-priori semimeasure also has two equivalent definitions:

$$M(x_{1:n}|y_{1:n}) := \sum_{p:U(p,y_{1:n})=x_{1:n}} 2^{-\ell(p)} = \sum_{\nu\in\mathcal{M}} 2^{-K(\nu)}\nu(x_{1:n}|y_{1:n}) \quad (7)$$

which is again in $\mathcal{M}_U^{|}$. In case of $|\mathcal{Y}| = 1$, this reduces to (1) and (3). The bounds (4) and (6) and others continue to hold, now for all individual ys, that is, M predicts asymptotically x_t from y_t and $(y_{<t}, x_{<t})$ for any y, provided x is sampled from a computable probability measure $\mu(\cdot|y_{1:\infty})$. Convergence is rapid if μ is not too complex.

Universal Reinforcement Learning

The generalized universal a-priori semimeasure (7) can be used to construct a universal reinforcement learning agent, called AIXI. In reinforcement learning, an *agent* interacts with an *environment* in cycles $t = 1, 2, \ldots, n$. In cycle t, the agent chooses an *action* y_t (e.g., a limb movement) based on past *perceptions* $x_{<t}$ and past actions $y_{<t}$. Thereafter, the agent perceives $x_t \equiv o_t r_t$, which consists of a (regular) *observation* o_t (e.g., a camera image) and a real-valued *reward* r_t. The reward may be scarce, for example, just +1 (−1) for winning (losing) a chess game, and 0 at all other times. Then the next cycle $t + 1$ starts. The goal of the agent is to maximize its expected reward over its lifetime n. Probabilistic planning deals with the situation in which the environmental probability distribution $\mu(x_{1:n}|y_{1:n})$ is known. Reinforcement learning deals with the case of unknown μ. In universal reinforcement learning, the unknown μ is replaced

by M similarly to the prediction, decision, and classification cases above. The universally optimal action in cycle t is (Hutter, 2005)

$$y_t := \arg\max_{y_t} \sum_{x_t} \ldots \max_{y_n} \sum_{x_n} [r_t + \cdots + r_n] M(x_{1:n}|y_{1:n}). \tag{8}$$

The expectations (Σ) and maximizations (max) over future x and y are interleaved in chronological order to form an expectimax tree similarly to minimax decision trees in extensive zero-sum games like chess. Optimality and universality results similar to the prediction case exist.

Approximations and Practical Applications

Since K and M are only semi-computable, they have to be approximated in practice. For instance, $-\log M(x) = K(x) + O(\log l(x))$, and $K(x)$ can be and has been approximated by off-the-shelf compressors like Lempel-Ziv and successfully applied to a plethora of clustering problems (Cilibrasi & Vitányi, 2005). The approximations upper-bound $K(x)$ and, for example, for Lempel-Ziv converge to $K(x)$ if x is sampled from a context tree source. The ▶Minimum Description Length principle (Grünwald, 2007) also attempts to approximate $K(x)$ for stochastic x. The Context Tree Weighting algorithm considers a relatively large subclass of $\mathcal{M}_U$ that can be summed over efficiently. This can be and has been combined with Monte-Carlo sampling to efficiently approximate AIXI 8 (Veness, Ng, Hutter, & Silver, 2010). The time-bounded versions of K and M, namely Levin complexity Kt and the speed prior S have also been applied to various learning tasks (Gaglio, 2007).

Other Applications

Continuously parameterized model classes are very common in statistics. Bayesian's usually assume a-prior *density* over some parameter $\theta \in \mathbb{R}^d$, which works fine for many problems, but has its problems. Even for continuous classes $\mathcal{M}$, one can assign a (proper) universal prior (not density) $w_\theta^U := 2^{-K(\theta)} > 0$ for computable θ (and ν_θ), and 0 for uncomputable ones. This effectively reduces $\mathcal{M}$ to a discrete class $\{\nu_\theta \in \mathcal{M} : w_\theta^U > 0\} \subseteq \mathcal{M}_U$ which is typically dense in $\mathcal{M}$. There are various fundamental philosophical and statistical problems and paradoxes around (Bayesian) induction, which nicely disappear in the universal framework. For instance, universal induction has no zero and no improper p(oste)rior problem, that is, can confirm universally quantified hypotheses, is reparametrization and representation invariant, and avoids the old-evidence and updating problem, in contrast to most classical continuous prior densities. It even performs well in incomputable environments, actually better than the latter (Hutter, 2007).

Discussion and Future Directions

Universal learning is designed to work for a wide range of problems without any a-priori knowledge. In practice, we often have extra information about the problem at hand, which could and should be used to guide the forecasting. One can incorporate it by explicating all our prior knowledge z, and place it on an extra input tape of our universal Turing machine U, or prefix our observation sequence x by z and use $M(zx)$ for prediction.

Another concern is the dependence of K and M on U. The good news is that a change of U changes $K(x)$ only within an additive and $M(x)$ within a multiplicative constant independent of x. This makes the theory practically immune to any "reasonable" choice of U for large data sets x, but predictions for short sequences (shorter than typical compiler lengths) can be arbitrary. One solution is to take into account our (whole) scientific prior knowledge z (Hutter, 2006), and predicting the now long string zx leads to good (less sensitive to "reasonable" U) predictions. This is a kind of grand transfer learning scheme. It is unclear whether a more elegant theoretical solution is possible.

Finally, the incomputability of K and M prevents a *direct* implementation of Solomonoff induction. Most fundamental theories have to be approximated for practical use, sometimes systematically like polynomial time approximation algorithms or numerical integration, and sometimes heuristically like in many AI-search problems or in non-convex optimization problems. Universal machine learning is similar, except that its core quantities are only semi-computable. This makes them often hard, but as described in the previous section, not impossible, to approximate.

In any case, universal induction can serve as a "gold standard" which practitioners can aim at. Solomonoff's

theory considers the class of all computable (stochastic) models, and a universal prior inspired by Ockham and Epicurus, quantified by Kolmogorov complexity. This leads to a universal theory of induction, prediction, decisions, and, by including Bellman, to universal actions in reactive environments. Future progress on the issues above (incorporating prior knowledge, getting rid of the compiler constants, and finding better approximations) will lead to new insights and will continually increase the number of applications.

Cross References

- ►Bayes Rule
- ►Bayesian Methods
- ►Bayesian Reinforcement Learning
- ►Classification
- ►Data Set
- ►Discriminative Learning
- ►Hypothesis Space
- ►Inductive Inference
- ►Loss
- ►Minimum Description Length
- ►On-line Learning
- ►Prior Probability
- ►Reinforcement Learning
- ►Time Series

Recommended Reading

Cilibrasi, R., & Vitányi, P. M. B. (2005). Clustering by compression. *IEEE Transactions on Information Theory, 51*(4), 1523–1545.

Gaglio, M. (2007). Universal search. *Scholarpedia, 2*(11), 2575.

Grünwald, P. D. (2007). *The minimum description length principle*. Cambridge: The MIT Press.

Hutter, M. (2005). *Universal artificial intelligence: Sequential decisions based on algorithmic probability*. Berlin: Springer.

Hutter, M. (2006). Human knowledge compression prize. open ended, http://prize.hutter1.net/.

Hutter, M. (2007). On universal prediction and Bayesian confirmation. *Theoretical Computer Science, 384*(1), 33–48.

Li, M., & Vitányi, P. M. B. (2008). *An introduction to Kolmogorov complexity and its applications* (3rd ed.). Berlin: Springer.

Schmidhuber, J. (2002). Hierarchies of generalized Kolmogorov complexities and nonenumerable universal measures computable in the limit. *International Journal of Foundations of Computer Science, 13*(4), 587–612.

Solomonoff, R. J. (1964). A formal theory of inductive inference: Parts 1 and 2. *Information and Control, 7*, 1–22 and 224–254.

Solomonoff, R. J. (1978). Complexity-based induction systems: Comparisons and convergence theorems. *IEEE Transactions on Information Theory, IT-24*, 422–432.

Veness, J., Ng, K. S., Hutter, M., & Silver, D. (2010). Reinforcement learning via AIXI approximation. In *Proceedings of 24th AAAI conference on artificial intelligence*, Atlanta. AAAI Press. 605–611.

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation, 1*(1), 67–82.

Zvonkin, A. K., & Levin, L. A. (1970). The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms. *Russian Mathematical Surveys, 25*(6), 83–124.

Unknown Attribute Values

►Missing Attribute Values

Unknown Values

►Missing Attribute Values

Unlabeled Data

Unlabeled data are ►data for which there are no target values. Unlabeled data are used in ►unsupervised learning. They stand in contrast to *labeled data* that have target values and are used in ►supervised learning.

Unsolicited Commercial Email Filtering

►Text Mining for Spam Filtering

Unstable Learner

An *unstable learner* produces large differences in generalization patterns when small changes are made to its initial conditions. The obvious initial condition is the set of training data used – for an unstable learner, sampling

a slightly different training set produces a large difference in testing behavior. Some models can be unstable in additional ways, for example ►neural networks are unstable with respect to the initial weights. In general this is an undesirable property – high sensitivity to training conditions is also known as high ►variance, which results in higher overall mean squared error. The flexibility enabled by being sensitive to data can thus be a blessing or a curse. Unstable learners can however be used to an advantage in ►ensemble learning methods, where large variance is "averaged out" across multiple learners.

Examples of unstable learners are: neural networks (assuming gradient descent learning), and ►decision trees. Examples of stable learners are ►support vector machines, ►K-nearest neighbor classifiers, and ►decision stumps. It should of course be recognized that there is a continuum between "stable" and "unstable," and the opinion of whether something is "sensitive" to initial conditions is somewhat of a subjective one. See also ►bias-variance decomposition for a more formal interpretation of this concept.

Unsupervised Learning

Unsupervised learning refers to any machine learning process that seeks to learn structure in the absence of either an identified output (cf. ►supervised learning) or feedback (cf. ►reinforcement learning). Three typical examples of unsupervised learning are ►clustering, ►association rules, and ►self-organizing maps.

Unsupervised Learning on Document Datasets

►Document Clustering

Utility Problem

►Explanation-Based Learning